aws

Guida per gli sviluppatori

Amazon Braket

Amazon Braket: Guida per gli sviluppatori

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Cos'è Amazon Braket?

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Amazon Braket è un software completamente gestito Servizio AWS che aiuta ricercatori, scienziati e sviluppatori a iniziare con l'informatica quantistica. L'informatica quantistica ha il potenziale per risolvere problemi computazionali che sfuggono alla portata dei computer classici perché sfrutta le leggi della meccanica quantistica per elaborare le informazioni in modi nuovi.

L'accesso all'hardware di calcolo quantistico può essere costoso e scomodo. L'accesso limitato rende difficile eseguire algoritmi, ottimizzare i progetti, valutare lo stato attuale della tecnologia e pianificare quando investire le risorse per ottenere il massimo beneficio. Braket ti aiuta a superare queste sfide.

Braket offre un unico punto di accesso a una varietà di tecnologie di calcolo quantistico. Con Braket puoi:

- Esplora e progetta algoritmi quantistici e ibridi.
- Testa algoritmi su diversi simulatori di circuiti quantistici.
- Esegui algoritmi su diversi tipi di computer quantistici.
- Crea applicazioni proof of concept.

La definizione di problemi quantistici e la programmazione di computer quantistici per risolverli richiede una nuova serie di competenze. Per aiutarvi ad acquisire queste competenze, Braket offre diversi ambienti per simulare ed eseguire i vostri algoritmi quantistici. Puoi trovare l'approccio più adatto alle tue esigenze e iniziare rapidamente con una serie di ambienti di esempio chiamati notebook.

Lo sviluppo di Braket prevede tre fasi:

- Build - Braket offre ambienti notebook Jupyter completamente gestiti che semplificano l'avvio. I notebook Braket sono preinstallati con algoritmi, risorse e strumenti di sviluppo di esempio, incluso

l'SDK Amazon Braket. Con Amazon Braket SDK, puoi creare algoritmi quantistici e poi testarli ed eseguirli su diversi computer e simulatori quantistici modificando una singola riga di codice.

- Test: Braket fornisce l'accesso a simulatori di circuiti quantistici ad alte prestazioni e completamente gestiti. Puoi testare e convalidare i tuoi circuiti. Braket gestisce tutti i componenti software sottostanti e i cluster Amazon Elastic Compute Cloud (Amazon EC2) per eliminare l'onere della simulazione di circuiti quantistici sulla classica infrastruttura HPC (High Performance Computing).
- Run - Braket fornisce un accesso sicuro e su richiesta a diversi tipi di computer quantistici. Hai accesso ai computer quantistici basati su gate di,, e AQT IonQIQM, nonché a un simulatore Rigetti hamiltoniano analogico di. QuEra Inoltre, non avete alcun impegno anticipato e non è necessario procurarvi l'accesso tramite singoli provider.

Informazioni sull'informatica quantistica e su Braket

L'informatica quantistica è nella sua fase iniziale di sviluppo. È importante capire che al momento non esiste un computer quantistico universale e tollerante ai guasti. Pertanto, alcuni tipi di hardware quantistico sono più adatti per ogni caso d'uso ed è fondamentale avere accesso a una varietà di hardware informatico. Braket offre una varietà di hardware tramite fornitori di terze parti.

L'hardware quantistico esistente è limitato a causa del rumore, che introduce errori. Il settore è nell'era del Noisy Intermediate Scale Quantum (NISQ). Nell'era NISQ, i dispositivi di calcolo quantistico sono troppo rumorosi per sostenere algoritmi quantistici puri, come l'algoritmo di Shor o l'algoritmo di Grover. Fino a quando non sarà disponibile una migliore correzione degli errori quantistici, l'informatica quantistica più pratica richiede la combinazione di risorse di calcolo classiche (tradizionali) con computer quantistici per creare algoritmi ibridi. Braket ti aiuta a lavorare con algoritmi quantistici ibridi.

Negli algoritmi quantistici ibridi, le unità di elaborazione quantistica (QPUs) vengono utilizzate come coprocessori per CPUs, velocizzando così calcoli specifici in un algoritmo classico. Questi algoritmi utilizzano l'elaborazione iterativa, in cui il calcolo si sposta tra computer classici e quantistici. Ad esempio, le attuali applicazioni dell'informatica quantistica in chimica, ottimizzazione e apprendimento automatico si basano su algoritmi quantistici variazionali, che sono un tipo di algoritmo quantistico ibrido. Negli algoritmi quantistici variazionali, le routine di ottimizzazione classiche regolano i parametri di un circuito quantistico parametrizzato in modo iterativo, più o meno allo stesso modo in cui i pesi di una rete neurale vengono regolati iterativamente in base all'errore in un set di addestramento per l'apprendimento automatico. Braket offre l'accesso alla libreria software PennyLane open source, che ti assiste con algoritmi quantistici variazionali.

L'informatica quantistica sta guadagnando terreno per i calcoli in quattro aree principali:

- Teoria dei numeri, inclusi fattorizzazione e crittografia (ad esempio, l'algoritmo di Shor è un metodo quantistico primario per i calcoli della teoria dei numeri)
- Ottimizzazione, che include la soddisfazione dei vincoli, la risoluzione di sistemi lineari e l'apprendimento automatico
- Informatica oracolare, che include la ricerca, i sottogruppi nascosti e la ricerca degli ordini (ad esempio, l'algoritmo di Grover è un metodo quantistico primario per i calcoli oracolari)
- Simulazione, tra cui simulazione diretta, invarianti di nodi e algoritmi di ottimizzazione quantistica approssimativa (QAOA)

Le applicazioni per queste categorie di calcoli si trovano nei servizi finanziari, nelle biotecnologie, nell'industria manifatturiera e nel settore farmaceutico, solo per citarne alcuni. Braket offre funzionalità e quaderni di esempio che possono già essere applicati a molti problemi di proof of concept oltre a determinati problemi pratici.

In questa sezione:

Come funziona Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Amazon Braket fornisce l'accesso su richiesta a dispositivi di elaborazione quantistica, inclusi simulatori di circuiti on-demand e diversi tipi di unità di elaborazione quantistica (). QPUs In Amazon Braket, la richiesta atomica a un dispositivo è un'operazione quantistica. Per i dispositivi basati su

gate, questa richiesta include il circuito quantistico (comprese le istruzioni di misurazione e il numero di scatti) e altri metadati della richiesta. Per i simulatori hamiltoniani analogici, il task quantistico contiene il layout fisico del registro quantistico e la dipendenza dal tempo e dallo spazio dei campi di manipolazione.

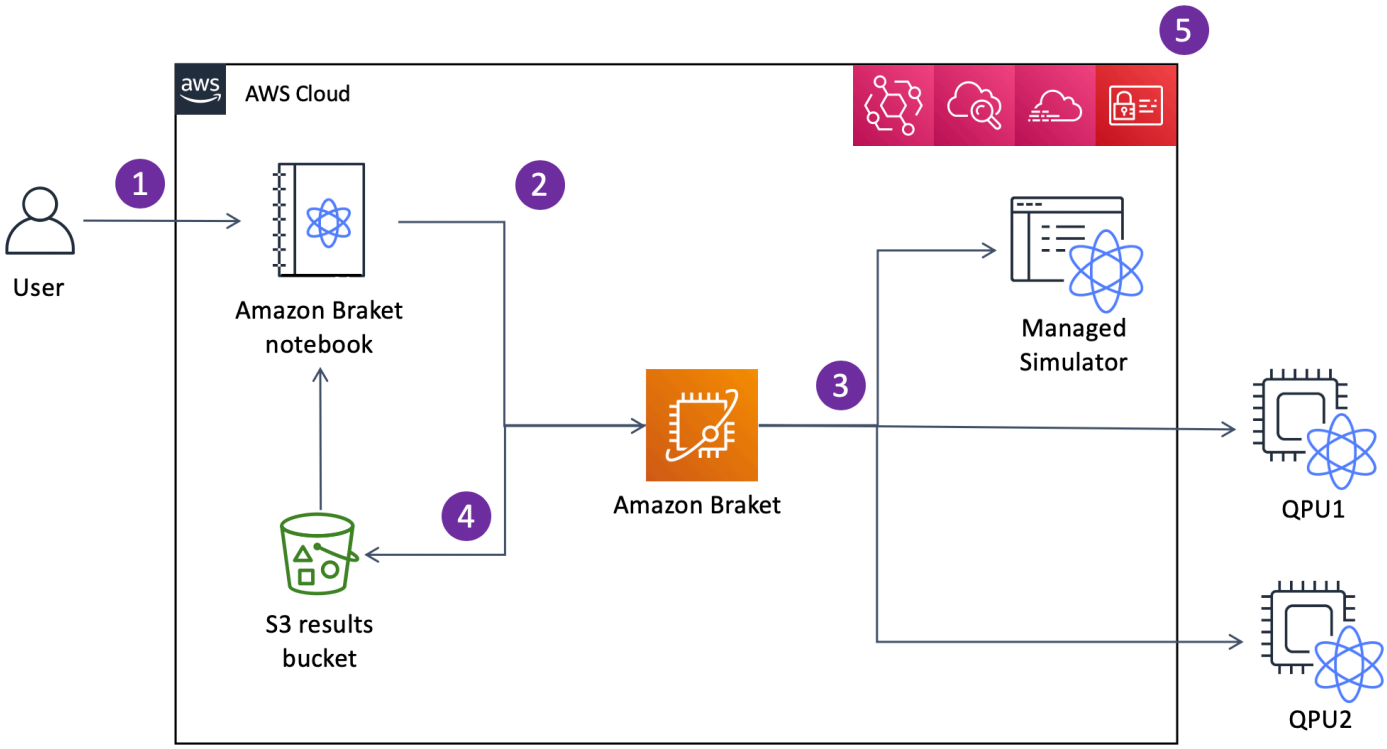
Braket Direct è un programma che amplia il modo in cui esplorare l'informatica quantistica, accelerando la ricerca e l'innovazione. AWS Puoi riservare capacità dedicata su vari dispositivi quantistici, interagire direttamente con specialisti di informatica quantistica e avere accesso anticipato alle funzionalità di nuova generazione, incluso l'ultimo dispositivo a ioni intrappolati di Forte. IonQ

In questa sezione, scopriremo il flusso di alto livello di esecuzione di attività quantistiche su Amazon Braket.

In questa sezione:

- Flusso di attività quantistico di Amazon Braket
- Trattamento dei dati da parte di terze

Flusso di attività quantistico di Amazon Braket



Con Jupyter i notebook, puoi definire, inviare e monitorare le tue attività quantistiche dalla console Amazon Braket o utilizzando l'SDK Amazon Braket. Puoi creare i tuoi circuiti quantistici direttamente nell'SDK. Tuttavia, per Analog Hamiltonian Simulators, dovete definire il layout del registro e i campi di controllo (1). Dopo aver definito l'attività quantistica, puoi scegliere un dispositivo su cui eseguirla e inviarla all'API Amazon Braket (2). A seconda del dispositivo scelto, l'attività quantistica viene messa in coda fino a quando non diventa disponibile e viene inviata alla QPU o al simulatore per l'implementazione (3). Amazon Braket ti dà accesso a una varietà di dispositivi quantistici supportati QPUs, tra cui simulatori on-demand, simulatori locali e un simulatore integrato.

Dopo aver elaborato l'operazione quantistica, Amazon Braket restituisce i risultati in un bucket Amazon S3, dove i dati vengono archiviati nel tuo (4). Account AWS Allo stesso tempo, l'SDK analizza i risultati in background e li carica nel notebook Jupyter al completamento dell'attività quantistica. Puoi anche visualizzare e gestire le tue attività quantistiche nella pagina Quantum Tasks nella console Amazon Braket o utilizzando `GetQuantumTask` il funzionamento di Amazon Braket. API

Amazon Braket è integrato con AWS Identity and Access Management (IAM), Amazon e AWS CloudTrail Amazon EventBridge per la gestione CloudWatch, il monitoraggio e la registrazione degli accessi degli utenti, nonché per l'elaborazione basata su eventi (5).

Trattamento dei dati da parte di terze

Le attività quantistiche inviate a un dispositivo QPU vengono elaborate su computer quantistici situati in strutture gestite da fornitori terzi. Per ulteriori informazioni sulla sicurezza e l'elaborazione di terze parti in Amazon Braket, consulta la sezione Sicurezza dei fornitori di hardware Amazon Braket.

Termini e concetti di Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

In Braket vengono utilizzati i seguenti termini e concetti:

Simulazione hamiltoniana analogica

La simulazione hamiltoniana analogica (AHS) è un paradigma di calcolo quantistico distinto per la simulazione diretta della dinamica quantistica dipendente dal tempo di sistemi a molti corpi. In AHS, gli utenti specificano direttamente un hamiltoniano dipendente dal tempo e il computer quantistico viene regolato in modo tale da emulare direttamente l'evoluzione temporale continua sotto questa forma hamiltoniana. I dispositivi AHS sono in genere dispositivi per scopi speciali e non computer quantistici universali come i dispositivi basati su gate. Sono limitati a una classe di hamiltoniani che possono simulare. Tuttavia, poiché questi hamiltoniani sono naturalmente implementati nel dispositivo, AHS non risente del sovraccarico necessario per formulare algoritmi sotto forma di circuiti e implementare le operazioni di gate.

Staffa

Abbiamo chiamato il servizio Braket con il nome della notazione bra-ket, una notazione standard della meccanica quantistica. È stato introdotto da Paul Dirac nel 1939 per descrivere lo stato dei sistemi quantistici ed è anche noto come notazione di Dirac.

Staffa Direct

Con Braket Direct, puoi prenotare un accesso dedicato a diversi dispositivi quantistici di tua scelta, entrare in contatto con specialisti di informatica quantistica per ricevere indicazioni sul tuo carico di lavoro e ottenere l'accesso anticipato alle funzionalità di nuova generazione, come i nuovi dispositivi quantistici con disponibilità limitata.

Braket Hybrid Job

Amazon Braket ha una funzionalità chiamata Amazon Braket Hybrid Jobs che fornisce esecuzioni completamente gestite di algoritmi ibridi. Un job ibrido Braket è composto da tre componenti:

1. La definizione del tuo algoritmo, che può essere fornita come script, modulo Python o contenitore Docker.
2. L'istanza di lavoro ibrida, basata su Amazon EC2, su cui eseguire l'algoritmo. L'impostazione predefinita è un'istanza ml.m5.xlarge.
3. Il dispositivo quantistico su cui eseguire le attività quantistiche che fanno parte dell'algoritmo. Un singolo lavoro ibrido contiene in genere una raccolta di molte attività quantistiche.

Dispositivo

In Amazon Braket, un dispositivo è un backend in grado di eseguire attività quantistiche. Un dispositivo può essere una QPU o un simulatore di circuiti quantistici. Per ulteriori informazioni, consulta la pagina Dispositivi supportati da Amazon Braket.

Attenuazione degli errori

La mitigazione degli errori implica l'esecuzione di più circuiti fisici e la combinazione delle relative misurazioni per ottenere un risultato migliore. Per ulteriori informazioni, consulta Tecniche di mitigazione degli errori.

Calcolo quantistico basato su Gate

Nel calcolo quantistico basato su gate (QC), chiamato anche QC basato su circuiti, i calcoli sono suddivisi in operazioni elementari (gate). Alcuni set di porte sono universali, il che significa che ogni calcolo può essere espresso come una sequenza finita di tali porte. Le porte sono gli elementi costitutivi dei circuiti quantistici e sono analoghe alle porte logiche dei circuiti digitali classici.

Limite Gateshot

Un limite di gateshot si riferisce al numero totale di gate per colpo (la somma di tutti i tipi di gate) e al numero di colpi per operazione. Matematicamente, il limite di gateshot può essere espresso come:

```
Gateshot limit = (Gate count per shot) * (Shot count per task)
```

hamiltoniano

La dinamica quantistica di un sistema fisico è determinata dalla sua formula hamiltoniana, che codifica tutte le informazioni sulle interazioni tra i componenti del sistema e gli effetti delle forze motrici esogene. L'hamiltoniana di un sistema a N-qubit è comunemente rappresentata come una matrice di numeri complessi 2 N per 2 N su macchine classiche. Eseguendo una simulazione hamiltoniana analogica su un dispositivo quantistico, è possibile evitare questi requisiti esponenziali di risorse.

Impulso

Un impulso è un segnale fisico transitorio trasmesso ai qubit. È descritto da una forma d'onda riprodotta in un frame che funge da supporto per il segnale portante ed è collegato al canale o alla porta hardware. I clienti possono progettare i propri impulsi fornendo l'involucro analogico che modula il segnale portante sinusoidale ad alta frequenza. Il frame è descritto in modo univoco da una frequenza e da una fase che spesso vengono scelte in risonanza con la separazione di energia tra i livelli di energia per |0# e |1# del qubit. I gate vengono quindi attivati come impulsi con una forma predeterminata e parametri calibrati come ampiezza, frequenza e durata. I casi d'uso non coperti dalle forme d'onda modello verranno abilitati tramite forme d'onda personalizzate che verranno specificate alla risoluzione del singolo campione fornendo un elenco di valori separati da un tempo di ciclo fisico fisso.

Circuito quantistico

Un circuito quantistico è il set di istruzioni che definisce un calcolo su un computer quantistico basato su gate. Un circuito quantistico è una sequenza di porte quantistiche, che sono trasformazioni reversibili su un registro, insieme a istruzioni di misurazione. qubit

Simulatore di circuiti quantistici

Un simulatore di circuiti quantistici è un programma per computer che funziona su computer classici e calcola i risultati di misurazione di un circuito quantistico. Per i circuiti generali, il fabbisogno di risorse di una simulazione quantistica cresce esponenzialmente con il numero di risorse da simulare. qubits Braket fornisce l'accesso a simulatori di circuiti quantistici gestiti (accessibili tramite BraketAPI) e locali (parte dell'Amazon Braket SDK).

Computer quantistico

Un computer quantistico è un dispositivo fisico che utilizza fenomeni quantomeccanici, come la sovrapposizione e l'entanglement, per eseguire calcoli. Esistono diversi paradigmi di calcolo quantistico (QC), come il controllo di qualità basato su gate.

Unità di elaborazione quantistica (QPU)

Una QPU è un dispositivo di calcolo quantistico fisico che può essere eseguito su un'attività quantistica. QPUs può essere basato su diversi paradigmi di controllo qualità, come il controllo di qualità basato su gate. Per ulteriori informazioni, consulta la pagina Dispositivi supportati da Amazon Braket.

Porte native QPU

Le porte native QPU possono essere mappate direttamente per controllare gli impulsi dal sistema di controllo QPU. I gate nativi possono essere eseguiti sul dispositivo QPU senza ulteriore compilazione. Sottoinsieme di porte supportate da QPU. Puoi trovare le porte native di un dispositivo nella pagina Dispositivi della console Amazon Braket e tramite l'SDK Braket.

Gate supportate da QPU

Le porte supportate da QPU sono le porte accettate dal dispositivo QPU. Questi gate potrebbero non funzionare direttamente sulla QPU, il che significa che potrebbe essere necessario scomporli in porte native. Puoi trovare le porte supportate di un dispositivo nella pagina Dispositivi della console Amazon Braket e tramite l'SDK Amazon Braket.

Attività quantistica

In Braket, un compito quantistico è la richiesta atomica a un dispositivo. Per i dispositivi QC basati su gate, ciò include il circuito quantistico (comprese le istruzioni di misurazione e il numero di)

e altri metadati di richiesta. shots Puoi creare attività quantistiche tramite l'SDK Amazon Braket o utilizzando CreateQuantumTask API direttamente l'operazione. Dopo aver creato un task quantistico, questo verrà messo in coda fino a quando il dispositivo richiesto non sarà disponibile. Puoi visualizzare le tue attività quantistiche nella pagina Quantum Tasks della console Amazon Braket o utilizzando GetQuantumTask le operazioni o. SearchQuantumTasks API

Qubit

L'unità di informazione di base in un computer quantistico è chiamata qubit (bit quantistico), proprio come un bit nell'informatica classica. A qubit è un sistema quantistico a due livelli che può essere realizzato mediante diverse implementazioni fisiche, come circuiti superconduttori o singoli ioni e atomi. Altri qubit tipi si basano su fotoni, spin elettronici o nucleari o sistemi quantistici più esotici.

Queue depth

Queue depthsi riferisce al numero di attività quantistiche e lavori ibridi in coda per un particolare dispositivo. Le attività quantistiche di un dispositivo e il conteggio delle code di lavoro ibride sono accessibili tramite l'opzione o. Braket Software Development Kit (SDK) Amazon Braket Management Console

1. La profondità della coda delle attività si riferisce al numero totale di attività quantistiche in attesa di essere eseguite con priorità normale.
2. La profondità della coda delle attività prioritarie si riferisce al numero totale di attività quantistiche inviate in attesa di essere eseguite. Amazon Braket Hybrid Jobs Queste attività hanno la priorità rispetto alle attività autonome una volta avviato un lavoro ibrido.
3. La profondità della coda dei lavori ibridi si riferisce al numero totale di lavori ibridi attualmente in coda su un dispositivo. Quantum tasksinviati come parte di un lavoro ibrido hanno la priorità e sono aggregati in. Priority Task Queue

Queue position

Queue positionsi riferisce alla posizione attuale dell'attività quantistica o del lavoro ibrido all'interno di una rispettiva coda di dispositivi. Può essere ottenuto per attività quantistiche o lavori ibridi tramite o. Braket Software Development Kit (SDK) Amazon Braket Management Console

Shots

Poiché l'informatica quantistica è intrinsecamente probabilistica, qualsiasi circuito deve essere valutato più volte per ottenere un risultato accurato. L'esecuzione e la misurazione di un singolo circuito si chiamano colpo. Il numero di scatti (esecuzioni ripetute) per un circuito viene scelto in base alla precisione desiderata per il risultato.

AWS terminologia e suggerimenti per Amazon Braket

politiche IAM

Una policy IAM è un documento che consente o nega autorizzazioni Servizi AWS e risorse. Le policy IAM consentono di personalizzare i livelli di accesso degli utenti alle risorse. Ad esempio, puoi consentire agli utenti di accedere a tutti i bucket Amazon S3 all'interno del tuo Account AWS bucket o solo a un bucket specifico.

- Procedura consigliata: segui il principio di sicurezza del privilegio minimo quando concedi le autorizzazioni. Seguendo questo principio, contribuite a impedire agli utenti o ai ruoli di disporre di più autorizzazioni di quelle necessarie per eseguire le loro attività quantistiche. Ad esempio, se un dipendente deve accedere solo a un bucket specifico, specifica il bucket nella policy IAM invece di concedere al dipendente l'accesso a tutti i bucket del tuo. Account AWS

Ruoli IAM

Un ruolo IAM è un'identità che puoi assumere per ottenere l'accesso temporaneo alle autorizzazioni. Prima che un utente, un'applicazione o un servizio possa assumere un ruolo IAM, è necessario concedere loro le autorizzazioni per passare al ruolo. Quando qualcuno assume un ruolo IAM, abbandona tutte le autorizzazioni precedenti che aveva in un ruolo precedente e assume le autorizzazioni del nuovo ruolo.

- Best practice: i ruoli IAM sono ideali per le situazioni in cui l'accesso ai servizi o alle risorse deve essere concesso temporaneamente, anziché a lungo termine.

Bucket Amazon S3

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) consente di archiviare dati come oggetti in bucket. Servizio AWS I bucket Amazon S3 offrono spazio di archiviazione illimitato. La dimensione massima per un oggetto in un bucket Amazon S3 è di 5 TB. Puoi caricare qualsiasi tipo di file in un bucket Amazon S3, come immagini, video, file di testo, file di backup, file multimediali per un sito Web, documenti archiviati e risultati delle attività quantistiche di Braket.

- Procedura ottimale: puoi impostare le autorizzazioni per controllare l'accesso al tuo bucket S3. Per ulteriori informazioni, consulta le politiche di Bucket nella documentazione di Amazon S3.

Monitoraggio e risparmio dei costi

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Con Amazon Braket, hai accesso a risorse di calcolo quantistico su richiesta senza impegno iniziale. I prezzi sono calcolati solo in base all'uso effettivo. Per ulteriori informazioni sui prezzi, visita la nostra pagina dei prezzi.

In questa sezione:

- Impostazione dei limiti di spesa per Amazon Braket QPUs
- Monitoraggio dei costi quasi in tempo reale
- Le migliori pratiche per il risparmio sui costi

Impostazione dei limiti di spesa per Amazon Braket QPUs

I limiti di spesa di Amazon Braket forniscono controlli opzionali dei costi per dispositivo per le unità di elaborazione quantistica (). QPUs

Come funzionano i limiti di spesa: Amazon Braket tiene traccia della spesa cumulativa e convalida ogni richiesta di creazione di attività rispetto al limite configurato. Se il costo stimato di un'attività supera il limite di spesa residuo, Amazon Braket la rifiuta immediatamente con un errore di convalida. Facoltativamente, puoi configurare un periodo di tempo per il tuo limite di spesa. Configurando un periodo di tempo, puoi assicurarti che le attività possano essere inviate solo in quel periodo specificato. Le attività inviate al di fuori del periodo di tempo verranno rifiutate.

Design opt-in: i flussi di lavoro esistenti rimarranno inalterati a meno che non abiliti esplicitamente i controlli. Puoi rimuovere tutte le restrizioni eliminando il limite di spesa.

> **Note**
>
> I limiti di spesa si applicano solo alle attività QPU per lavori su richiesta e ibridi. Escludono simulatori, notebook gestiti, costi delle istanze Hybrid Job EC2 e prenotazioni Braket Direct. Per una gestione completa dei costi su tutti i servizi AWS, continua a utilizzare Budget AWS.

Elenco delle azioni relative ai limiti di spesa

Cerca

Con il seguente comando AWS CLI, puoi cercare ed elencare i limiti di spesa in una regione AWS specifica e per uno specifico dispositivo Braket.

```
aws --region {device_region} braket search-spending-limits --filters
 name=deviceArn,operator=EQUAL,values={device_arn}
```

Crea

Con il seguente comando AWS CLI, puoi creare un nuovo limite di spesa per un dispositivo quantistico specifico in una regione specifica. La richiesta viene rifiutata se esiste già un limite di spesa per il dispositivo.

```
aws --region {device_region} braket create-spending-limit --device-arn {device_arn}
 --spending-limit {max_spend}
```

Aggiorna

Con il seguente comando AWS CLI, puoi aggiornare un limite di spesa esistente con un nuovo valore massimo di spesa. La richiesta viene rifiutata se la somma della spesa corrente e della spesa in coda è già superiore alla nuova spesa massima richiesta.

```
aws --region {device_region} braket update-spending-limit --spending-limit-arn
 {spending_limit_arn} --spending-limit {new_max_spend}
```

Puoi fornire un periodo di tempo anziché o in aggiunta alla nuova spesa massima, come nell'esempio precedente.

Eliminare

Con il seguente comando AWS CLI, puoi eliminare un limite di spesa esistente.

```
aws --region {device_region} braket delete-spending-limit --spending-limit-arn
 {spending_limit_arn}
```

Puoi fornire un periodo di tempo al posto o in aggiunta alla nuova spesa massima, come nell'esempio precedente.

Sebbene facoltativo, è consigliabile specificare sempre il parametro della regione come procedura consigliata. I comandi eseguiti in un'area diversa da quella del dispositivo falliranno o, nel caso di`SearchSpendingLimits`, restituiranno risultati errati.

Per altri esempi su come utilizzare i limiti di spesa, consulta il taccuino di esempio.

Come funziona la convalida delle attività

Quando l'account AWS invia una `CreateQuantumTask` richiesta altrimenti valida, è soggetto al seguente comportamento di gating. Nota: il budget residuo è la differenza tra il limite di spesa e la somma della spesa in coda e quella corrente. (Vedi la sezione successiva)

- Caso 1: Non esiste un limite di spesa per il dispositivo dell'attività: L'attività viene creata.
- Caso 2: esiste un limite di spesa per il dispositivo di destinazione e l'ora corrente rientra nel periodo di tempo del limite di spesa:
  - Se il costo stimato dell'attività è inferiore o uguale al budget rimanente: CreateQuantumTask ha esito positivo, l'attività viene creata.
  - Se il costo stimato è superiore al budget residuo: `CreateQuantumTask` fallisce e non viene creata alcuna attività.
- Caso 3: esiste un limite di spesa per il dispositivo di destinazione e l'ora corrente non rientra nel periodo di tempo del limite di spesa: `CreateQuantumTask` non riesce e non viene creata alcuna attività.

Come viene calcolato il budget residuo

Il budget residuo è la differenza tra il limite di spesa e la somma della spesa corrente e della spesa in coda.

Quando viene creata un'attività per un dispositivo con un limite di spesa, la spesa in coda viene aumentata del costo stimato dell'attività. Questo evento è elencato nella prima riga della tabella

seguente. La tabella seguente mostra cosa succede alla spesa in coda e alla spesa corrente, a seconda della progressione dell'attività.

Vecchio stato quantistico dell'attività	Nuovo stato quantistico dell'attività	Passa alla spesa in coda	Passa alla spesa corrente
-	created	Aumentata in base al costo stimato	Nessuna modifica
created	IN CODA	Nessuna modifica	Nessuna modifica
Qualsiasi	RUNNING (ESECUZIONE IN CORSO)	Nessuna modifica	Nessuna modifica
Qualsiasi	IN CORSO DI ANNULLAMENTO	Nessuna modifica	Nessuna modifica
IN CORSO DI ANNULLAMENTO	ANNULLATO	Ridotto in base al costo stimato	Nessuna modifica
Qualsiasi	NON RIUSCITO	Ridotto in base al costo stimato	Nessuna modifica
RUNNING (ESECUZIONE IN CORSO)	COMPLETED	Ridotto in base al costo stimato	Aumentato in base al costo stimato (adeguato di conseguenza per le attività parzialmente completate)

Custodie Edge

D: Quando si crea un limite di spesa, le attività già in coda vengono conteggiate ai fini della spesa in coda?

R: No. Le attività già create, in coda o comunque in corso non vengono conteggiate ai fini della spesa in coda di un limite di spesa appena creato.

D: La riduzione del limite di spesa, aggiornandolo, causa la cessazione anticipata di un'attività quantistica creata, in coda o comunque in corso?

R: No.

D: Il raggiungimento dell'ora di fine del limite di spesa comporta l'interruzione anticipata di un'attività quantistica creata, in coda o comunque in corso?

R: No. Le attività create, in coda o comunque in corso possono essere completate indipendentemente dallo stato del limite di spesa.

D: In che modo la mancanza di un limite di spesa è diversa da un limite di spesa di zero dollari?

R: Nessun limite di spesa consente di creare attività quantistiche senza restrizioni. Un limite di spesa pari a zero dollari blocca tutte le attività quantistiche.

D: Un limite di spesa pari a zero nel passato o nel futuro blocca la creazione di tutte le attività quantistiche?

R: Sì.

D: Quando si crea un limite di spesa, il costo stimato delle attività già in coda verrà conteggiato ai fini della spesa corrente una volta completate tali attività?

R: No. Solo le attività inviate mentre è attivo un limite di spesa vengono conteggiate ai fini della spesa accumulata.

Monitoraggio dei costi quasi in tempo reale

L'SDK Braket ti offre la possibilità di aggiungere un monitoraggio dei costi quasi in tempo reale ai tuoi carichi di lavoro quantistici. Ciascuno dei nostri notebook di esempio include un codice di monitoraggio dei costi per fornirti una stima massima dei costi per le unità di elaborazione quantistica () e i simulatori on-demand di Braket. QPUs Le stime dei costi massimi verranno mostrate in USD e non includono crediti o sconti.

> **Note**
>
> I costi indicati sono stime basate sul simulatore Amazon Braket e sull'utilizzo delle attività dell'unità di elaborazione quantistica (QPU). I costi stimati indicati possono differire dagli addebiti effettivi. I costi stimati non tengono conto di sconti o crediti e potrebbero verificarsi

costi aggiuntivi in base all'utilizzo di altri servizi come Amazon Elastic Compute Cloud (Amazon EC2) Elastic Compute Cloud (Amazon EC2).

Monitoraggio dei costi per SV1

Per dimostrare come può essere utilizzata la funzione di tracciamento dei costi, costruiremo un circuito Bell State e lo eseguiremo sul nostro SV1 simulatore. Inizia importando i moduli Braket SDK, definendo un Bell State e aggiungendo la funzione al `Tracker()` nostro circuito:

```
#import any required modules
from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.tracking import Tracker

#create our bell circuit
circ = Circuit().h(0).cnot(0,1)
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
with Tracker() as tracker:
    task = device.run(circ, shots=1000).result()

#Your results
print(task.measurement_counts)
```

```
Counter({'00': 500, '11': 500})
```

Quando eseguite il vostro Notebook, potete aspettarvi il seguente risultato per la simulazione di Bell State. La funzione tracker vi mostrerà il numero di scatti inviati, le attività quantistiche completate, la durata dell'esecuzione, la durata di esecuzione fatturata e il costo massimo in USD. Il tempo di esecuzione può variare per ogni simulazione.

```
import datetime

tracker.quantum_tasks_statistics()
{'arn:aws:braket:::device/quantum-simulator/amazon/sv1':
 {'shots': 1000,
  'tasks': {'COMPLETED': 1},
  'execution_duration': datetime.timedelta(microseconds=4000),
  'billed_execution_duration': datetime.timedelta(seconds=3)}}
```

```
tracker.simulator_tasks_cost()
```

```
Decimal('0.0037500000')
```

Utilizzo del tracker dei costi per impostare i costi massimi

È possibile utilizzare il tracker dei costi per impostare i costi massimi di un programma. Potresti avere una soglia massima per quanto vuoi spendere per un determinato programma. In questo modo, puoi utilizzare il cost tracker per creare una logica di controllo dei costi nel tuo codice di esecuzione. L'esempio seguente utilizza lo stesso circuito su una Rigetti QPU e limita il costo a 1 USD. Il costo per eseguire un'iterazione del circuito nel nostro codice è di 0,30 USD. Abbiamo impostato la logica per ripetere le iterazioni fino a quando il costo totale superi 1 USD; quindi, il frammento di codice verrà eseguito tre volte fino a quando l'iterazione successiva non supererà 1 USD. In genere, un programma continua a iterare fino a raggiungere il costo massimo desiderato, in questo caso, tre iterazioni.

```
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")
with Tracker() as tracker:
    while tracker.qpu_tasks_cost() < 1:
        result = device.run(circ, shots=200).result()
print(tracker.quantum_tasks_statistics())
print(tracker.qpu_tasks_cost(), "USD")
```

```
{'arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3': {'shots': 600, 'tasks':
 {'COMPLETED': 3}}}
1.4400000000 USD
```

> **Note**
>
> Il tracker dei costi non terrà traccia della durata delle attività quantistiche falliteTN1. Durante una TN1 simulazione, se la prova viene completata ma la fase di contrazione fallisce, il costo della prova non verrà visualizzato nel tracker dei costi.

Le migliori pratiche per il risparmio sui costi

Prendi in considerazione le seguenti best practice per l'utilizzo di Amazon Braket. Risparmia tempo, minimizza i costi ed evita gli errori più comuni.

Verifica con i simulatori

- Verifica i circuiti utilizzando un simulatore prima di eseguirlo su una QPU, in modo da poter ottimizzare il circuito senza incorrere in costi per l'utilizzo della QPU.
- Sebbene i risultati dell'esecuzione del circuito su un simulatore possano non essere identici a quelli dell'esecuzione del circuito su una QPU, è possibile identificare errori di codifica o problemi di configurazione utilizzando un simulatore.

Limita l'accesso degli utenti a determinati dispositivi

- È possibile impostare restrizioni che impediscano agli utenti non autorizzati di inviare attività quantistiche su determinati dispositivi. Il metodo consigliato per limitare l'accesso è con IAM. AWS Per ulteriori informazioni su come eseguire questa operazione, consulta Limita l'accesso.
- Ti consigliamo di non utilizzare il tuo account amministratore per concedere o limitare l'accesso degli utenti ai dispositivi Amazon Braket.

Imposta allarmi di fatturazione

- Puoi impostare un allarme di fatturazione per avvisarti quando la fattura raggiunge un limite preimpostato. Il metodo consigliato per impostare un allarme è tramite. Budget AWSÈ possibile impostare budget personalizzati e ricevere avvisi quando i costi o l'utilizzo possono superare l'importo preventivato. Le informazioni sono disponibili all'indirizzo. Budget AWS

Esegui compiti TN1 quantistici con un numero di puntate basso

- I simulatori costano meno di QPUs, ma alcuni simulatori possono essere costosi se le attività quantistiche vengono eseguite con un numero elevato di puntate. Ti consigliamo di testare le tue TN1 attività con un conteggio basso. shot Shotil conteggio non influisce sul costo SV1 e sulle attività del simulatore locale.

Controlla tutte le regioni per le attività quantistiche

- La console mostra le attività quantistiche solo per quelle correnti. Regione AWS Quando cerchi attività quantistiche fatturabili che sono state inviate, assicurati di controllare tutte le regioni.
- È possibile visualizzare un elenco dei dispositivi e delle regioni associate nella pagina della documentazione dei dispositivi supportati.

Referenze e repository API per Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Amazon Braket fornisce un' APIsinterfaccia a riga di comando che puoi utilizzare per creare e gestire istanze di notebook e addestrare e distribuire modelli. SDKs

- SDK Amazon Braket Python (consigliato)
- Riferimento all'API Amazon Braket
- AWS Command Line Interface
- AWS SDK per .NET
- AWS SDK per C++
- AWS SDK per GoAPI Reference
- AWS SDK per Java
- AWS SDK per JavaScript
- AWS SDK per PHP
- AWS SDK per Python (Boto)
- AWS SDK per Ruby

Puoi anche ottenere esempi di codice dal repository Amazon Braket GitHub Tutorials.

- Tutorial Braket GitHub

Repository principali

Di seguito viene visualizzato un elenco di repository principali che contengono pacchetti chiave utilizzati per Braket:

- Braket Python SDK: utilizza l'SDK Braket Python per configurare il codice sui Jupyter notebook nel linguaggio di programmazione Python. Dopo aver configurato i Jupyter notebook, puoi eseguire il codice su dispositivi e simulatori Braket
- Braket Schemas: il contratto tra Braket SDK e il servizio Braket.
- Braket Default Simulator - Tutti i nostri simulatori quantistici locali per Braket (vettore di stato e matrice di densità).

Plugin

Poi ci sono i vari plugin che vengono utilizzati insieme a vari dispositivi e strumenti di programmazione. Questi includono i plug-in supportati da Braket e i plug-in supportati da terze parti, come mostrato di seguito.

Amazon Braket supportato:

- Libreria di algoritmi Amazon Braket: un catalogo di algoritmi quantistici predefiniti scritti in Python. Eseguili così come sono o usali come punto di partenza per creare algoritmi più complessi.
- Braket- PennyLane plugin: da utilizzare PennyLane come framework QML su Braket.

Terze parti (il team di Braket monitora e contribuisce):

- Provider Qiskit-Braket: utilizza l'SDK per accedere alle risorse Braket. Qiskit
- Braket-Julia SDK - (SPERIMENTALE) Una versione nativa di Julia del Braket SDK

Regioni e dispositivi supportati da Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

In Amazon Braket, un dispositivo rappresenta un'unità di elaborazione quantistica (QPU) o un simulatore che puoi chiamare per eseguire attività quantistiche. Amazon Braket fornisce l'accesso ai dispositivi QPU daAQT,, IonQIQM, QuEra e. Rigetti Inoltre, AWS offre l'accesso a simulatori on-

demand, locali e integrati. Per ulteriori informazioni sui simulatori incorporati, vedere Informazioni sui simulatori incorporati.

Per informazioni sui fornitori di hardware quantistico supportati, vedere Invio di attività quantistiche a. QPUs Per informazioni sui simulatori disponibili, vedere Invio di attività quantistiche ai simulatori. La tabella seguente mostra l'elenco dei dispositivi e dei simulatori disponibili.

Provider	Nome dispositivo	Paradigma	Tipo	ARN del dispositivo	Region
AQT	IBEX-Q1	basato su gate	QPU	arn:aws:braket:eu-north-1:: -Q1 device/qpu/aqt/Ibex	eu-north-1
IonQ	Forte-1	basato su gate	QPU	arn:aws:braket:us-east-1:: -1 device/qpu/ionq/Forte	us-east-1
IonQ	Forte-Enterprise-1	basato su gate	QPU	arn:aws:braket:us-east-1:: -Enterprise-1 device/qpu/ionq/Forte	us-east-1
IQM	Garnet	basato su gate	QPU	arn:aws:braket:eu-north-1:: device/qpu/iqm/Garnet	eu-north-1
IQM	Emerald	basato su gate	QPU	arn:aws:braket:eu-north-1:: device/qpu/iqm/Emerald	eu-north-1
QuEra	Aquila	Simulazione hamiltoniana analogica	QPU	arn:aws:braket:us-east-1:: device/qpu/quera/Aquila	us-east-1
Rigetti	Ankaa-3	basato su gate	QPU	arn:aws:braket:us-west-1:: -3 device/qpu/rigetti/Ankaa	us-west-1

Provider	Nome dispositivo	Paradigma	Tipo	ARN del dispositivo	Region
AWS	freno_sv	basato su gate	simulatore locale	N/A (simulatore locale in Braket SDK)	N/D
AWS	braket_dm	basato su gate	simulatore locale	N/A (simulatore locale in Braket SDK)	N/D
AWS	braket_ahs	Simulazione hamiltoniana analogica	Simulatore locale	N/A (simulatore locale in Braket SDK)	N/D
AWS	SV1	basato su gate	simulatore su richiesta	arn:aws:braket::: 1 device/quantum-simulator/amazon/sv	us-east-1, us-west-1, us-west-2, eu-west-2
AWS	DM1	basato su gate	simulatore su richiesta	arn:aws:braket::: 1 device/quantum-simulator/amazon/dm	us-east-1, us-west-1, us-west-2, eu-west-2
AWS	TN1	basato su gate	simulatore su richiesta	arn:aws:braket::: 1 device/quantum-simulator/amazon/tn	us-east-1, us-west-2 e eu-west-2

> **Note**
>
> I dispositivi ARNs sono sensibili alle maiuscole. Ad esempio, quando utilizzi il AQT IBEX-Q1 dispositivo, verifica che l'ARN del dispositivo contenga. 'Ibex-Q1'

Per ulteriori dettagli sull' QPUs utilizzo con Amazon Braket, consulta Amazon Braket Quantum Computers.

Proprietà del dispositivo

Per tutti i dispositivi, puoi trovare ulteriori proprietà del dispositivo, come la topologia del dispositivo, i dati di calibrazione e i set di porte nativi, nella scheda Dispositivi della console Amazon Braket o tramite l'API. `GetDevice` Quando si costruisce un circuito con i simulatori, Amazon Braket richiede l'utilizzo di qubit o indici contigui. Quando si lavora con l'SDK, il seguente esempio di codice mostra come accedere alle proprietà del dispositivo per ogni dispositivo e simulatore disponibile.

```
from braket.aws import AwsDevice
from braket.devices import LocalSimulator

device = AwsDevice('arn:aws:braket:::device/quantum-simulator/amazon/sv1')
 # SV1
# device = LocalSimulator()
 # Local State Vector Simulator
# device = LocalSimulator("default")
 # Local State Vector Simulator
# device = LocalSimulator(backend="default")
 # Local State Vector Simulator
# device = LocalSimulator(backend="braket_sv")
 # Local State Vector Simulator
# device = LocalSimulator(backend="braket_dm")
 # Local Density Matrix Simulator
# device = LocalSimulator(backend="braket_ahs")
 # Local Analog Hamiltonian Simulation
# device = AwsDevice('arn:aws:braket:::device/quantum-simulator/amazon/tn1')
 # TN1
# device = AwsDevice('arn:aws:braket:::device/quantum-simulator/amazon/dm1')
 # DM1
# device = AwsDevice('arn:aws:braket:eu-north-1::device/qpu/aqt/Ibex-Q1')
 # AQT IBEX-Q1
# device = AwsDevice('arn:aws:braket:us-east-1::device/qpu/ionq/Forte-1')
 # IonQ Forte-1
```

```
# device = AwsDevice('arn:aws:braket:us-east-1::device/qpu/ionq/Forte-Enterprise-1')
 # IonQ Forte-Enterprise-1
# device = AwsDevice('arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet')
 # IQM Garnet
# device = AwsDevice('arn:aws:braket:eu-north-1::device/qpu/iqm/Emerald')
 # IQM Emerald
# device = AwsDevice('arn:aws:braket:us-east-1::device/qpu/quera/Aquila')
 # QuEra Aquila
# device = AwsDevice('arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3')
 # Rigetti Ankaa-3

# Get device properties
device.properties
```

Regioni ed endpoint per Amazon Braket

Per un elenco completo delle regioni e degli endpoint, consulta la Guida AWS generale.

Le attività quantistiche eseguite su un dispositivo QPU possono essere visualizzate nella console Amazon Braket nella regione di quel dispositivo. Quando usi l'SDK Amazon Braket, puoi inviare attività quantistiche a qualsiasi dispositivo QPU, indipendentemente dalla regione in cui lavori. L'SDK crea automaticamente una sessione nella regione per la QPU specificata.

Amazon Braket è disponibile nelle seguenti versioni: Regioni AWS

Nome Regione	Regione	Endpoint Braket
Stati Uniti orientali (Virginia settentrionale)	us-east-1	braket.us-east-1.amazonaws.com (solo) IPv4 braket.us-east-1.api.aws (pila doppia)
Stati Uniti occidentali (California settentrionale)	us-west-1	braket.us-west-1.amazonaws.com (solo) IPv4 braket.us-west-1.api.aws (pila doppia)
Stati Uniti occidentali 2 (Oregon)	us-west-2	braket.us-west-2.amazonaws.com (IPv4 solo)

Nome Regione	Regione	Endpoint Braket
		braket.us-west-2.api.aws (pila doppia)
EU North 1 (Stoccolma)	eu-north-1	braket.eu-north-1.amazonaws.com (IPv4 solo) braket.eu-north-1.api.aws (pila doppia)
EU West 2 (Londra)	eu-west-2	braket.eu-west-2.amazonaws.com (IPv4 solo) braket.eu-west-2.api.aws (pila doppia)

> **Note**
>
> L'SDK Amazon Braket non supporta IPv6 solo reti.

Guida introduttiva ad Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Dopo aver seguito le istruzioni in Abilita Amazon Braket, puoi iniziare a usare Amazon Braket.

I passaggi per iniziare includono:

- Abilita Amazon Braket
- Crea un'istanza di notebook Amazon Braket
- Crea un'istanza del notebook Braket usando CloudFormation

Abilita Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Puoi abilitare Amazon Braket nel tuo account tramite la AWS console.

In questa sezione:

- Prerequisiti
- Passaggi per abilitare Amazon Braket

Prerequisiti

Per abilitare ed eseguire Amazon Braket, devi disporre di un utente o di un ruolo con l'autorizzazione per avviare azioni Amazon Braket. Queste autorizzazioni sono incluse nella policy AmazonBraketFullAccess IAM (arn:aws:iam: :aws:policy/). AmazonBraketFullAccess

> **Note**
>
> Se sei un amministratore:
> Per consentire ad altri utenti di accedere ad Amazon Braket, concedi le autorizzazioni agli utenti allegando la AmazonBraketFullAccesspolicy o allegando una policy personalizzata creata da te. Per ulteriori informazioni sulle autorizzazioni necessarie per utilizzare Amazon Braket, consulta Gestire l'accesso ad Amazon Braket.

Passaggi per abilitare Amazon Braket

1. Accedi alla console Amazon Braket con il tuo. Account AWS
2. Apri la console Amazon Braket.
3. Dalla pagina di destinazione di Braket, fai clic su Inizia per accedere alla pagina Service Dashboard. L'avviso nella parte superiore della dashboard del servizio ti guiderà nei tre passaggi seguenti:
   a. Creazione di ruoli collegati ai servizi (SLR)
   b. Abilitazione dell'accesso a computer quantistici di terze parti
   c. Creazione di una nuova istanza del notebook Jupyter

Per utilizzare dispositivi quantistici di terze parti, è necessario accettare determinate condizioni relative al trasferimento di dati tra l'utente e tali dispositivi. AWS I termini e le condizioni del presente contratto sono disponibili nella scheda Generale della pagina Autorizzazioni e impostazioni della console Amazon Braket.

> **Note**
>
> I dispositivi quantistici che non coinvolgono terze parti, come i simulatori locali Braket o i simulatori on-demand, possono essere utilizzati senza accettare il contratto Enable third devices.

L'accettazione di questi termini per consentire l'uso di dispositivi di terze parti deve essere effettuata solo una volta per account se si accede a hardware di terze parti.

Crea un'istanza di notebook Amazon Braket

> **Tip**
>
> Impara le basi dell'informatica quantistica AWS con! Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Amazon Braket offre notebook Jupyter completamente gestiti per aiutarti a iniziare. Le istanze di notebook Amazon Braket si basano su istanze di notebook Amazon SageMaker AI. I passaggi seguenti descrivono come creare una nuova istanza di notebook per clienti nuovi ed esistenti.

Nuovi clienti Amazon Braket:

1. Apri la console Amazon Braket e accedi alla pagina Dashboard nel riquadro a sinistra.
2. Fai clic su Inizia nella modalità Benvenuto in Amazon Braket al centro della pagina del pannello di controllo. Fornisci il nome di un notebook per creare un notebook Jupyter predefinito.
   a. La creazione del notebook potrebbe richiedere alcuni minuti.
   b. Il blocco appunti verrà elencato nella pagina Taccuini con lo stato In sospeso.
   c. Quando l'istanza del notebook è pronta per l'uso, lo stato cambia in. InService
   d. Aggiorna la pagina per visualizzare lo stato aggiornato del notebook.

Clienti Amazon Braket esistenti:

1. Apri la console Amazon Braket e seleziona Notebook nel riquadro a sinistra.
2. Seleziona Crea un'istanza di notebook.
   a. Se non hai notebook, seleziona la configurazione standard per creare un notebook Jupyter predefinito.
3. Immettete il nome dell'istanza di Notebook, utilizzando solo caratteri alfanumerici e trattini, e selezionate la modalità visiva preferita.
4. Abilita o disabilita il gestore dell'inattività di Notebook per il tuo notebook.

a. Se abilitato, seleziona la durata di inattività desiderata prima che il notebook venga ripristinato. Quando un notebook viene ripristinato, i costi di elaborazione smettono di essere addebitati, ma i costi di archiviazione continueranno.

b. Per verificare quanto tempo di inattività rimane per l'istanza del notebook, accedi alla barra dei comandi, seleziona la scheda Braket, quindi la scheda Inactivity Manager.

> **Note**
>
> Per salvare il tuo lavoro, integra l'istanza del tuo notebook SageMaker AI con un repository Git. In alternativa, sposta il lavoro all'esterno delle `/Braket Examples` cartelle `/Braket Algorithms` and in modo che non vengano sovrascritti dal riavvio dell'istanza del notebook.

5. (Facoltativo) Con la configurazione avanzata, puoi creare un notebook con autorizzazioni di accesso, configurazioni aggiuntive e impostazioni di accesso alla rete:

   a. Nella configurazione Notebook scegli il tipo di istanza.

      i. Per impostazione predefinita, viene scelto il tipo di istanza standard ed economico ml.t3.medium. Per ulteriori informazioni sui prezzi delle istanze, consulta i prezzi di Amazon SageMaker AI.

   b. Per associare un repository Github pubblico all'istanza del notebook, fai clic sul menu a discesa del repository Git e seleziona Clona un repository git pubblico dall'URL dal menu a discesa Repository. Inserisci l'URL del repository nella barra di testo dell'URL del repository Git.

   c. In Autorizzazioni di accesso, configura tutti i ruoli IAM opzionali, l'accesso root e le chiavi di crittografia.

   d. In Accesso alla rete, configura le impostazioni di rete e di accesso personalizzate per la tua Jupyter Notebook istanza.

6. Controlla le impostazioni e imposta eventuali tag per identificare l'istanza del tuo notebook. Fai clic su Avvia.

> **Note**
>
> Visualizza e gestisci le istanze dei tuoi notebook Amazon Braket nelle console Amazon Braket e Amazon AI. SageMaker Ulteriori impostazioni del notebook Amazon Braket sono disponibili tramite la SageMaker console.

Se lavori nella console Amazon Braket all'interno dell'SDK Amazon Braket e AWS i plug-in sono precaricati nei notebook che hai creato. Per eseguirlo sul tuo computer, installa l'SDK e i plugin quando esegui il comando o quando esegui il comando per i plugin. `pip install amazon-braket-sdk` `pip install amazon-braket-pennylane-plugin` PennyLane

Crea un'istanza del notebook Braket usando CloudFormation

> **Tip**
>
> Impara le basi dell'informatica quantistica con! AWS Iscriviti all'Amazon Braket Digital Learning Plan e ottieni il tuo badge digitale dopo aver completato una serie di corsi di apprendimento e una valutazione digitale.

Puoi utilizzarle CloudFormation per gestire le istanze dei tuoi notebook Amazon Braket. Le istanze per notebook Braket sono basate su Amazon SageMaker AI. Con CloudFormation, puoi fornire a un'istanza notebook un file modello che descrive la configurazione desiderata. Il file modello è scritto in formato JSON o YAML. È possibile creare, aggiornare ed eliminare le istanze in modo ordinato e ripetibile. Potresti trovarlo utile quando gestisci più istanze di Braket Notebook nel tuo. Account AWS

Dopo aver creato un CloudFormation modello per un notebook Braket, lo usi CloudFormation per distribuire la risorsa. Per ulteriori informazioni, consulta Creazione di uno stack sulla CloudFormation console nella guida per l'CloudFormation utente.

Per creare un'istanza di Braket Notebook utilizzando CloudFormation, esegui questi tre passaggi:

1. Crea uno script di SageMaker configurazione del ciclo di vita AI.
2. Crea un ruolo AWS Identity and Access Management (IAM) che venga assunto dall' SageMaker IA.
3. Crea un'istanza di notebook SageMaker AI con il prefisso **`amazon-braket-`**

Puoi riutilizzare la configurazione del ciclo di vita per tutti i notebook Braket che crei. Puoi anche riutilizzare il ruolo IAM per i notebook Braket a cui assegni le stesse autorizzazioni di esecuzione.

In questa sezione:

- Fase 1: Creare uno script di configurazione del ciclo di vita AI SageMaker
- Fase 2: creare il ruolo IAM assunto da Amazon SageMaker AI

- Passaggio 3: creare un'istanza di notebook SageMaker AI con il prefisso amazon-braket-

Fase 1: Creare uno script di configurazione del ciclo di vita AI SageMaker

Utilizza il seguente modello per creare uno script di configurazione del ciclo di vita SageMaker AI. Lo script personalizza un'istanza di notebook SageMaker AI per Braket. Per le opzioni di configurazione per la CloudFormation risorsa del ciclo di vita, consulta AWS::SageMaker::NotebookInstanceLifecycleConfigla guida per l'utente.CloudFormation

```
BraketNotebookInstanceLifecycleConfig:
    Type: "AWS::SageMaker::NotebookInstanceLifecycleConfig"
    Properties:
      NotebookInstanceLifecycleConfigName: BraketLifecycleConfig-${AWS::StackName}
      OnStart:
      - Content:
          Fn::Base64: |
            #!/usr/bin/env bash
            sudo -u ec2-user -i #EOS
            curl -o braket-notebook-lcc.zip https://d3ded4lzb1lnme.cloudfront.net/
notebook/braket-notebook-lcc.zip
            unzip braket-notebook-lcc.zip
            ./install.sh
            EOS

            exit 0
```

Fase 2: creare il ruolo IAM assunto da Amazon SageMaker AI

Quando utilizzi un'istanza Braket Notebook, l' SageMaker IA esegue operazioni per tuo conto. Ad esempio, supponiamo di utilizzare un notebook Braket utilizzando un circuito su un dispositivo supportato. All'interno dell'istanza del notebook, l' SageMaker intelligenza artificiale esegue l'operazione su Braket per te. Il ruolo di esecuzione del notebook definisce le operazioni esatte che l' SageMaker IA è autorizzata a eseguire per tuo conto. Per ulteriori informazioni, consulta i ruoli SageMaker AI nella guida per sviluppatori di Amazon SageMaker AI.

Usa l'esempio seguente per creare un ruolo di esecuzione del notebook Braket con le autorizzazioni richieste. È possibile modificare le politiche in base alle proprie esigenze.

> **Note**
>
> Assicurati che il ruolo abbia l'autorizzazione per i bucket `s3:ListBucket` e le `s3:GetObject` operazioni su Amazon S3 con il prefisso. `braketnotebookcdk-"` Lo script di configurazione del ciclo di vita richiede queste autorizzazioni per copiare lo script di installazione del notebook Braket.

```
ExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      RoleName: !Sub AmazonBraketNotebookRole-${AWS::StackName}
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        -
          Effect: "Allow"
          Principal:
            Service:
              - "sagemaker.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/service-role/"
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonBraketFullAccess
      Policies:
        -
          PolicyName: "AmazonBraketNotebookPolicy"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - s3:GetObject
                  - s3:PutObject
                  - s3:ListBucket
                Resource:
                  - arn:aws:s3:::amazon-braket-*
                  - arn:aws:s3:::braketnotebookcdk-*
              - Effect: "Allow"
                Action:
                  - "logs:CreateLogStream"
```

```
                - "logs:PutLogEvents"
                - "logs:CreateLogGroup"
                - "logs:DescribeLogStreams"
              Resource:
                - !Sub "arn:aws:logs:*:${AWS::AccountId}:log-group:/aws/sagemaker/*"
            - Effect: "Allow"
              Action:
                - braket:*
              Resource: "*"
```

Passaggio 3: creare un'istanza di notebook SageMaker AI con il prefisso **`amazon-braket-`**

Utilizza lo script del ciclo di vita SageMaker AI e il ruolo IAM creati nei passaggi 1 e 2 per creare un'istanza di notebook SageMaker AI. L'istanza del notebook è personalizzata per Braket ed è accessibile con la console Amazon Braket. Per ulteriori informazioni sulle opzioni di configurazione per questa CloudFormation risorsa, consulta AWS::SageMaker::NotebookInstancela guida per l'CloudFormation utente.

```
BraketNotebook:
    Type: AWS::SageMaker::NotebookInstance
    Properties:
      InstanceType: ml.t3.medium
      NotebookInstanceName: !Sub amazon-braket-notebook-${AWS::StackName}
      RoleArn: !GetAtt ExecutionRole.Arn
      VolumeSizeInGB: 30
      LifecycleConfigName: !GetAtt
 BraketNotebookInstanceLifecycleConfig.NotebookInstanceLifecycleConfigName
```

Sviluppa le tue attività quantistiche con Amazon Braket

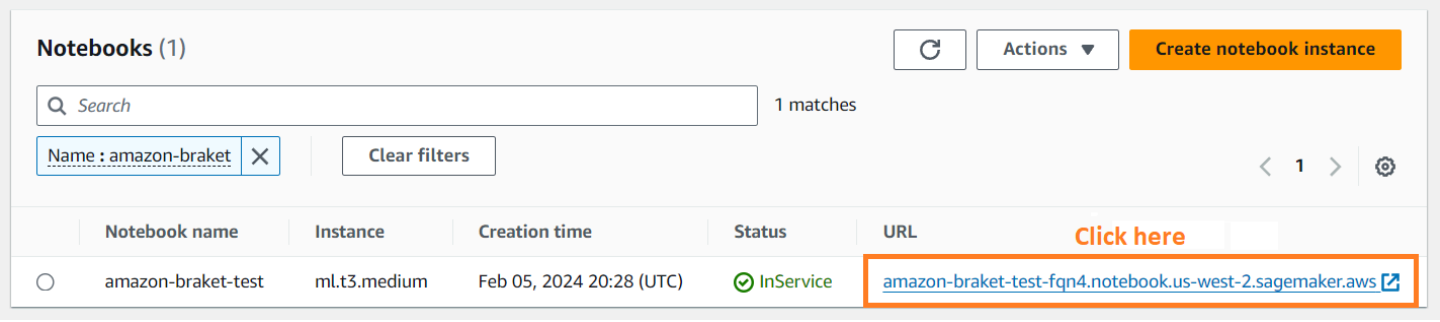
Braket offre ambienti notebook Jupyter completamente gestiti che semplificano l'avvio. I notebook Braket sono preinstallati con algoritmi, risorse e strumenti di sviluppo di esempio, incluso l'SDK Amazon Braket. Con Amazon Braket SDK, puoi creare algoritmi quantistici e poi testarli ed eseguirli su diversi computer e simulatori quantistici modificando una singola riga di codice.

In questa sezione:

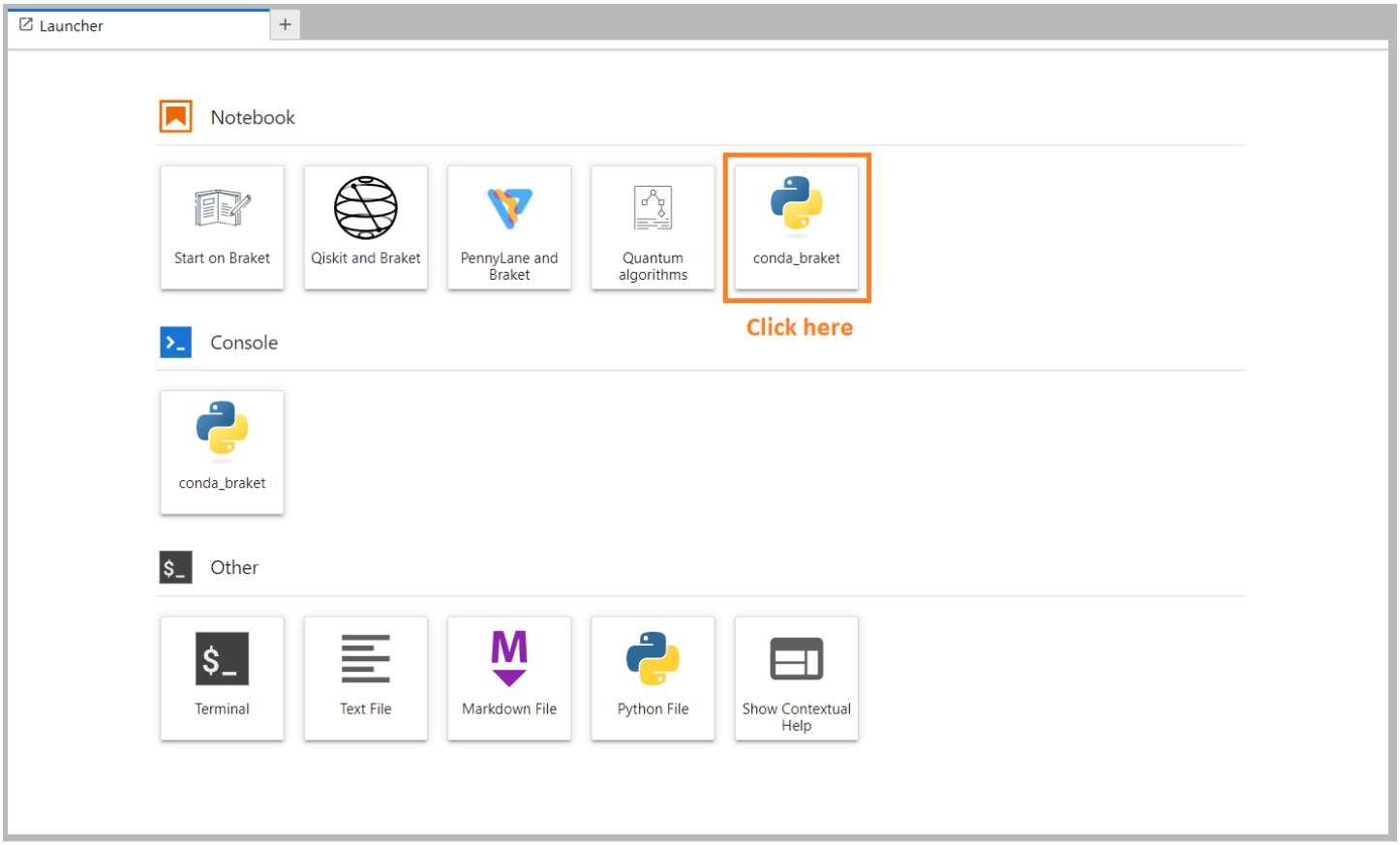
- Costruisci il tuo primo circuito
- Ottenere la consulenza di un esperto
- Esegui i tuoi circuiti con OpenQASM 3.0
- Esplora le funzionalità sperimentali
- Controllo a impulsi su Amazon Braket
- Simulazione hamiltoniana analogica
- Lavorare con AWS Boto3

Costruisci il tuo primo circuito

Dopo l'avvio dell'istanza del notebook, apri l'istanza con un'interfaccia Jupyter standard scegliendo il notebook appena creato.



Le istanze di notebook Amazon Braket sono preinstallate con l'SDK Amazon Braket e tutte le sue dipendenze. Inizia creando un nuovo notebook con kernel. `conda_braket`



Puoi iniziare con un semplice «Hello, world!» esempio. Innanzitutto, costruite un circuito che prepari uno stato di Bell, quindi eseguitelo su dispositivi diversi per ottenere i risultati.

Inizia importando Begin importando i moduli Amazon Braket SDK e definendo un BRAKETlong semplice modulo SDK e definendo un circuito Bell State di base.

```
import boto3
from braket.aws import AwsDevice
from braket.devices import LocalSimulator
from braket.circuits import Circuit

# Create the circuit
bell = Circuit().h(0).cnot(0, 1)
```

Puoi visualizzare il circuito con questo comando:

```
print(bell)
```

```
T  : #  0  #  1  #
```

```
      #####
q0 : ## H ########
      #####   #
            #####
q1 : ######## X ##
            #####
T  : #  0  #  1  #
```

Esegui il tuo circuito sul simulatore locale

Quindi, scegli il dispositivo quantistico su cui eseguire il circuito. L'SDK Amazon Braket è dotato di un simulatore locale per la prototipazione e i test rapidi. Ti consigliamo di utilizzare il simulatore locale per circuiti più piccoli, che possono essere fino a 25 qubits (a seconda dell'hardware locale).

Per creare un'istanza del simulatore locale:

```
# Instantiate the local simulator
local_sim = LocalSimulator()
```

ed esegui il circuito:

```
# Run the circuit
result = local_sim.run(bell, shots=1000).result()
counts = result.measurement_counts
print(counts)
```

Dovresti vedere un risultato simile a questo:

```
Counter({'11': 503, '00': 497})
```

Lo stato specifico di Bell che avete preparato è una sovrapposizione uguale di |00# e |11# e una distribuzione quasi uguale (fino al shot rumore) di 00 e 11 come risultati di misurazione, come previsto.

Esegui il tuo circuito su un simulatore on-demand

Amazon Braket fornisce anche l'accesso a un simulatore on-demand ad alte prestazioni per l'esecuzione di circuiti più grandiSV1. SV1è un simulatore vettoriale di stato su richiesta che consente la simulazione di circuiti quantistici fino a 34. qubits Puoi trovare ulteriori informazioni nella sezione

Dispositivi supportati e SV1 nella console. AWS Quando esegui attività quantistiche su SV1 (e su TN1 qualsiasi QPU), i risultati dell'attività quantistica vengono archiviati in un bucket S3 del tuo account. Se non specifichi un bucket, Braket SDK crea automaticamente un bucket predefinito. `amazon-braket-{region}-{accountID}` Per ulteriori informazioni, consulta Gestire l'accesso ad Amazon Braket.

> **Note**
>
> Inserisci il nome effettivo del bucket esistente, come indicato nell'esempio seguente `amazon-braket-s3-demo-bucket` come nome del bucket. I nomi dei bucket per Amazon Braket iniziano sempre `amazon-braket-` con, seguiti da altri caratteri identificativi che aggiungi. Se hai bisogno di informazioni su come configurare un bucket S3, consulta Guida introduttiva ad Amazon S3.

```
# Get the account ID
aws_account_id = boto3.client("sts").get_caller_identity()["Account"]

# The name of the bucket
my_bucket = "amazon-braket-s3-demo-bucket"

# The name of the folder in the bucket
my_prefix = "simulation-output"
s3_folder = (my_bucket, my_prefix)
```

Per far funzionare un circuitoSV1, devi fornire la posizione del bucket S3 che hai selezionato in precedenza come argomento posizionale nella chiamata. `.run()`

```
# Choose the cloud-based on-demand simulator to run your circuit
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")

# Run the circuit
task = device.run(bell, s3_folder, shots=100)

# Display the results
print(task.result().measurement_counts)
```

La console Amazon Braket fornisce ulteriori informazioni sulla tua attività quantistica. Vai alla scheda Quantum Tasks nella console e il tuo compito quantistico dovrebbe essere in cima all'elenco. In

alternativa, puoi cercare il tuo task quantistico utilizzando l'ID univoco del task quantistico o altri criteri.

> **Note**
>
> Dopo 90 giorni, Amazon Braket rimuove automaticamente tutte le attività quantistiche IDs e gli altri metadati associati alle tue attività quantistiche. Per ulteriori informazioni, consulta Conservazione dei dati.

In esecuzione su una QPU

Con Amazon Braket, puoi eseguire il precedente esempio di circuito quantistico su un computer quantistico fisico semplicemente modificando una singola riga di codice. Amazon Braket fornisce l'accesso a una varietà di dispositivi QPU (Quantum Processing Unit). Puoi trovare informazioni sui diversi dispositivi e sulle finestre di disponibilità nella sezione Dispositivi supportati e nella AWS console nella scheda Dispositivi. L'esempio seguente mostra come creare un'istanza di un IQM dispositivo.

```
# Choose the IQM hardware to run your circuit
device = AwsDevice("arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet")
```

Oppure scegli un IonQ dispositivo con questo codice:

```
# Choose the Ionq device to run your circuit
device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1")
```

Dopo aver selezionato un dispositivo e prima di eseguire il carico di lavoro, puoi interrogare la profondità della coda del dispositivo con il codice seguente per determinare il numero di attività quantistiche o di lavori ibridi. Inoltre, i clienti possono visualizzare le profondità di coda specifiche del dispositivo nella pagina Dispositivi di. Amazon Braket Management Console

```
# Print your queue depth
print(device.queue_depth().quantum_tasks)
# Returns the number of quantum tasks queued on the device
# {<QueueType.NORMAL: 'Normal'>: '0', <QueueType.PRIORITY: 'Priority'>: '0'}

print(device.queue_depth().jobs)
# Returns the number of hybrid jobs queued on the device
```

```
# '2'
```

Quando esegui un'attività, l'SDK Amazon Braket esegue un sondaggio per ottenere un risultato (con un timeout predefinito di 5 giorni). Puoi modificare questa impostazione predefinita modificando il `poll_timeout_seconds` parametro nel `.run()` comando, come mostrato nell'esempio che segue. Tieni presente che se il timeout di polling è troppo breve, i risultati potrebbero non essere restituiti entro il tempo di polling, ad esempio quando una QPU non è disponibile e viene restituito un errore di timeout locale. È possibile riavviare il polling chiamando la funzione. `task.result()`

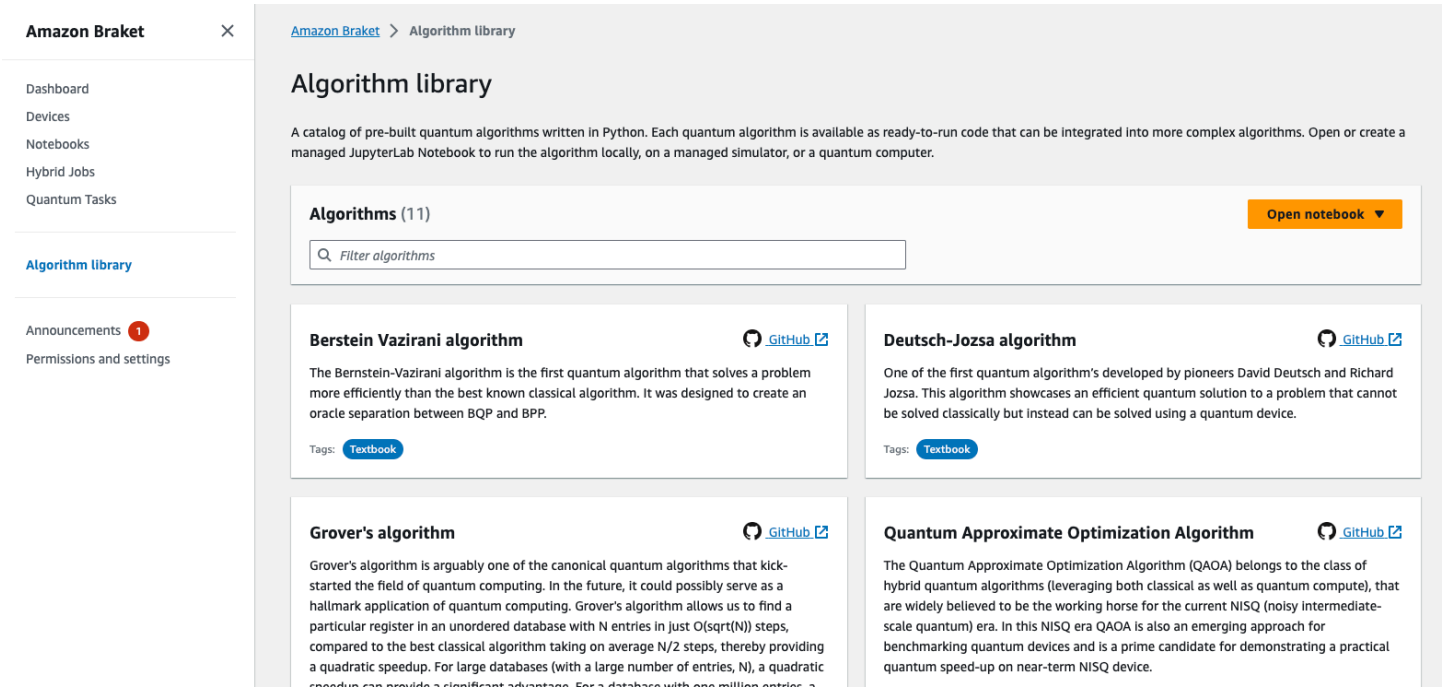
```
# Define quantum task with 1 day polling timeout
task = device.run(bell, s3_folder, poll_timeout_seconds=24*60*60)
print(task.result().measurement_counts)
```

Inoltre, dopo aver inviato l'operazione quantistica o il lavoro ibrido, è possibile richiamare la `queue_position()` funzione per controllare la posizione della coda.

```
print(task.queue_position().queue_position)
# Return the number of quantum tasks queued ahead of you
# '2'
```

Creazione dei primi algoritmi quantistici

La libreria di algoritmi Amazon Braket è un catalogo di algoritmi quantistici predefiniti scritti in Python. Esegui questi algoritmi così come sono o usali come punti di partenza per creare algoritmi più complessi. Puoi accedere alla libreria di algoritmi dalla console Braket. Per ulteriori informazioni, consulta la libreria di algoritmi Braket Github.



La console Braket fornisce una descrizione di ogni algoritmo disponibile nella libreria di algoritmi. Scegliete un GitHub link per visualizzare i dettagli di ogni algoritmo oppure scegliete Apri taccuino per aprire o creare un taccuino che contenga tutti gli algoritmi disponibili. Se scegli l'opzione notebook, puoi trovare la libreria di algoritmi Braket nella cartella principale del tuo notebook.

Costruzione di circuiti nell'SDK

Questa sezione fornisce esempi di definizione di un circuito, visualizzazione delle porte disponibili, estensione di un circuito e visualizzazione delle porte supportate da ciascun dispositivo. Contiene anche istruzioni su come allocare manualmentequbits, indicare al compilatore di eseguire i circuiti esattamente come definito e creare circuiti rumorosi con un simulatore di rumore.

In Braket puoi anche lavorare a livello di impulsi per vari cancelli, con determinati tipi di porte. QPUs Per ulteriori informazioni, consulta Pulse Control su Amazon Braket.

In questa sezione:

- Cancelli e circuiti
- Set di programmi
- Misurazione parziale
- Allocazione manuale qubit
- Compilazione Verbatim

- Simulazione del rumore

Cancelli e circuiti

Le porte e i circuiti quantistici sono definiti nella `braket.circuits`classe dell'SDK Amazon Braket Python. Dall'SDK, puoi creare un'istanza di un nuovo oggetto di circuito chiamando. `Circuit()`

Esempio: definire un circuito

L'esempio inizia definendo un circuito campione di quattro qubits (etichettatiq`0`, `q1q2`, `eq3`) costituiti da porte Hadamard standard a singolo qubit e porte CNOT a due qubit. È possibile visualizzare questo circuito chiamando la funzione come illustrato nell'esempio seguente. `print`

```
# Import the circuit module
from braket.circuits import Circuit

# Define circuit with 4 qubits
my_circuit = Circuit().h(range(4)).cnot(control=0, target=2).cnot(control=1, target=3)
print(my_circuit)
```

```
T  : #  0  #     1     #
      #####
q0 : ## H ##############
      #####   #
      #####   #
q1 : ## H ##############
      #####   #     #
      ##### #####   #
q2 : ## H ### X ########
      ##### #####   #
      #####       #####
q3 : ## H ######### X ##
      #####       #####
T  : #  0  #     1     #
```

Esempio: definire un circuito parametrizzato

In questo esempio, definiamo un circuito con porte che dipendono da parametri liberi. Possiamo specificare i valori di questi parametri per creare un nuovo circuito o, quando inviamo il circuito, per eseguirlo come attività quantistica su determinati dispositivi.

```
from braket.circuits import Circuit, FreeParameter

# Define a FreeParameter to represent the angle of a gate
alpha = FreeParameter("alpha")

# Define a circuit with three qubits
my_circuit = Circuit().h(range(3)).cnot(control=0, target=2).rx(0, alpha).rx(1, alpha)
print(my_circuit)
```

È possibile creare un nuovo circuito non parametrizzato a partire da un circuito parametrizzato fornendo un singolo circuito `float` (che è il valore che assumeranno tutti i parametri liberi) o argomenti di parole chiave che specificano il valore di ciascun parametro al circuito come segue.

```
my_fixed_circuit = my_circuit(1.2)
my_fixed_circuit = my_circuit(alpha=1.2)
print(my_fixed_circuit)
```

Nota che non `my_circuit` è modificato, quindi puoi usarlo per creare istanziazioni di molti nuovi circuiti con valori di parametro fissi.

Esempio: modifica delle porte in un circuito

L'esempio seguente definisce un circuito con porte che utilizzano modificatori di controllo e potenza. È possibile utilizzare queste modifiche per creare nuove porte, come la porta controllata. `Ry`

```
from braket.circuits import Circuit

# Create a bell circuit with a controlled x gate
my_circuit = Circuit().h(0).x(control=0, target=1)

# Add a multi-controlled Ry gate of angle .13
my_circuit.ry(angle=.13, target=2, control=(0, 1))

# Add a 1/5 root of X gate
my_circuit.x(0, power=1/5)

print(my_circuit)
```

I modificatori di porta sono supportati solo sul simulatore locale.

Esempio: visualizza tutte le porte disponibili

L'esempio seguente mostra come visualizzare tutti i gate disponibili in Amazon Braket.

```
from braket.circuits import Gate
# Print all available gates in Amazon Braket
gate_set = [attr for attr in dir(Gate) if attr[0].isupper()]
print(gate_set)
```

L'output di questo codice elenca tutte le porte.

```
['CCNot', 'CNot', 'CPhaseShift', 'CPhaseShift00', 'CPhaseShift01', 'CPhaseShift10',
 'CSwap', 'CV', 'CY', 'CZ', 'ECR', 'GPhase', 'GPi', 'GPi2', 'H', 'I', 'ISwap', 'MS',
 'PRx', 'PSwap', 'PhaseShift', 'PulseGate', 'Rx', 'Ry', 'Rz', 'S', 'Si', 'Swap', 'T',
 'Ti', 'U', 'Unitary', 'V', 'Vi', 'X', 'XX', 'XY', 'Y', 'YY', 'Z', 'ZZ']
```

Ognuna di queste porte può essere aggiunta a un circuito chiamando il metodo per quel tipo di circuito. Ad esempio, chiamate`circ.h(0)`, per aggiungere una porta Hadamard alla prima. qubit

> **Note**
>
> Le porte vengono aggiunte al loro posto e l'esempio che segue aggiunge tutte le porte elencate nell'esempio precedente allo stesso circuito.

```
circ = Circuit()
# toffoli gate with q0, q1 the control qubits and q2 the target.
circ.ccnot(0, 1, 2)
# cnot gate
circ.cnot(0, 1)
# controlled-phase gate that phases the |11> state, cphaseshift(phi) =
 diag((1,1,1,exp(1j*phi))), where phi=0.15 in the examples below
circ.cphaseshift(0, 1, 0.15)
# controlled-phase gate that phases the |00> state, cphaseshift00(phi) =
 diag([exp(1j*phi),1,1,1])
circ.cphaseshift00(0, 1, 0.15)
# controlled-phase gate that phases the |01> state, cphaseshift01(phi) =
 diag([1,exp(1j*phi),1,1])
circ.cphaseshift01(0, 1, 0.15)
# controlled-phase gate that phases the |10> state, cphaseshift10(phi) =
 diag([1,1,exp(1j*phi),1])
circ.cphaseshift10(0, 1, 0.15)
# controlled swap gate
```

```
circ.cswap(0, 1, 2)
# swap gate
circ.swap(0,1)
# phaseshift(phi)= diag([1,exp(1j*phi)])
circ.phaseshift(0,0.15)
# controlled Y gate
circ.cy(0, 1)
# controlled phase gate
circ.cz(0, 1)
# Echoed cross-resonance gate applied to q0, q1
circ = Circuit().ecr(0,1)
# X rotation with angle 0.15
circ.rx(0, 0.15)
# Y rotation with angle 0.15
circ.ry(0, 0.15)
# Z rotation with angle 0.15
circ.rz(0, 0.15)
# Hadamard gates applied to q0, q1, q2
circ.h(range(3))
# identity gates applied to q0, q1, q2
circ.i([0, 1, 2])
# iswap gate, iswap = [[1,0,0,0],[0,0,1j,0],[0,1j,0,0],[0,0,0,1]]
circ.iswap(0, 1)
# pswap gate, PSWAP(phi) = [[1,0,0,0],[0,0,exp(1j*phi),0],[0,exp(1j*phi),0,0],
[0,0,0,1]]
circ.pswap(0, 1, 0.15)
# X gate applied to q1, q2
circ.x([1, 2])
# Y gate applied to q1, q2
circ.y([1, 2])
# Z gate applied to q1, q2
circ.z([1, 2])
# S gate applied to q0, q1, q2
circ.s([0, 1, 2])
# conjugate transpose of S gate applied to q0, q1
circ.si([0, 1])
# T gate applied to q0, q1
circ.t([0, 1])
# conjugate transpose of T gate applied to q0, q1
circ.ti([0, 1])
# square root of not gate applied to q0, q1, q2
circ.v([0, 1, 2])
# conjugate transpose of square root of not gate applied to q0, q1, q2
circ.vi([0, 1, 2])
```

```
# exp(-iXX theta/2)
circ.xx(0, 1, 0.15)
# exp(i(XX+YY) theta/4), where theta=0.15 in the examples below
circ.xy(0, 1, 0.15)
# exp(-iYY theta/2)
circ.yy(0, 1, 0.15)
# exp(-iZZ theta/2)
circ.zz(0, 1, 0.15)
# IonQ native gate GPi with angle 0.15 applied to q0
circ.gpi(0, 0.15)
# IonQ native gate GPi2 with angle 0.15 applied to q0
circ.gpi2(0, 0.15)
# IonQ native gate MS with angles 0.15, 0.15, 0.15 applied to q0, q1
circ.ms(0, 1, 0.15, 0.15, 0.15)
```

Oltre al set di porte predefinito, è possibile applicare al circuito anche porte unitarie autodefinite. Queste possono essere porte a qubit singolo (come mostrato nel codice sorgente seguente) o porte a più qubit applicate al valore definito dal parametro. qubits `targets`

```
import numpy as np

# Apply a general unitary
my_unitary = np.array([[0, 1],[1, 0]])
circ.unitary(matrix=my_unitary, targets=[0])
```

Esempio: estendere i circuiti esistenti

È possibile estendere i circuiti esistenti aggiungendo istruzioni. An `Instruction` è una direttiva quantistica che descrive il compito quantistico da eseguire su un dispositivo quantistico. `Instruction`gli operatori includono solo oggetti di tipo. `Gate`

```
# Import the Gate and Instruction modules
from braket.circuits import Gate, Instruction

# Add instructions directly.
circ = Circuit([Instruction(Gate.H(), 4), Instruction(Gate.CNot(), [4, 5])])

# Or with add_instruction/add functions
instr = Instruction(Gate.CNot(), [0, 1])
circ.add_instruction(instr)
circ.add(instr)
```

```
# Specify where the circuit is appended
circ.add_instruction(instr, target=[3, 4])
circ.add_instruction(instr, target_mapping={0: 3, 1: 4})

# Print the instructions
print(circ.instructions)
# If there are multiple instructions, you can print them in a for loop
for instr in circ.instructions:
     print(instr)

# Instructions can be copied
new_instr = instr.copy()
# Appoint the instruction to target
new_instr = instr.copy(target=[5, 6])
new_instr = instr.copy(target_mapping={0: 5, 1: 6})
```

Esempio: visualizza le porte supportate da ogni dispositivo

I simulatori supportano tutte le porte dell'SDK Braket, ma i dispositivi QPU supportano un sottoinsieme più piccolo. Puoi trovare le porte supportate di un dispositivo nelle proprietà del dispositivo. Di seguito viene mostrato un esempio con un dispositivo IonQ:

```
# Import the device module
from braket.aws import AwsDevice

device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Aria-1")

# Get device name
device_name = device.name
# Show supportedQuantumOperations (supported gates for a device)
device_operations = device.properties.dict()['action']['braket.ir.openqasm.program']
['supportedOperations']
print('Quantum Gates supported by {}:\n {}'.format(device_name, device_operations))
```

```
Quantum Gates supported by Aria 1:
 ['x', 'y', 'z', 'h', 's', 'si', 't', 'ti', 'v', 'vi', 'rx', 'ry', 'rz', 'cnot',
 'swap', 'xx', 'yy', 'zz']
```

Potrebbe essere necessario compilare le porte supportate in porte native prima di poter essere eseguite su hardware quantistico. Quando invii un circuito, Amazon Braket esegue questa compilazione automaticamente.

Esempio: recupera a livello di codice la fedeltà delle porte native supportate da un dispositivo

È possibile visualizzare le informazioni sulla fedeltà nella pagina Dispositivi della console Braket. A volte è utile accedere alle stesse informazioni a livello di programmazione. Il codice seguente mostra come estrarre la fedeltà a due qubit gate tra due gate di una QPU.

```
# Import the device module
from braket.aws import AwsDevice

device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")

# Specify the qubits
a=10
b=11
edge_properties_entry =
 device.properties.standardized.twoQubitProperties['10-11'].twoQubitGateFidelity
gate_name = edge_properties_entry[0].gateName
fidelity = edge_properties_entry[0].fidelity
print(f"Fidelity of the {gate_name} gate between qubits {a} and {b}: {fidelity}")
```

Set di programmi

I set di programmi eseguono in modo efficiente più circuiti quantistici in un'unica attività quantistica. In quell'unica attività, puoi inviare fino a 100 circuiti quantistici o un singolo circuito parametrico con un massimo di 100 set di parametri diversi. Questa operazione riduce al minimo il tempo tra le successive esecuzioni dei circuiti e riduce il sovraccarico di elaborazione delle attività quantistiche. Attualmente, i set di programmi sono supportati su Amazon Braket Local Simulator e sui AQT dispositiviIQM. Rigetti

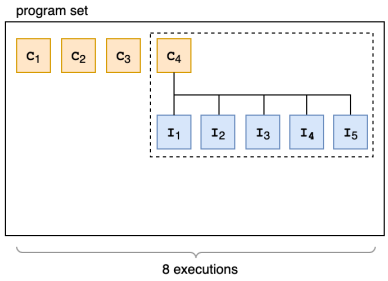
Definizione di un ProgramSet

Il primo esempio di codice seguente mostra come costruire un `ProgramSet` utilizzando sia circuiti parametrizzati che circuiti senza parametri.

```
from braket.aws import AwsDevice
from braket.circuits import Circuit, FreeParameter
from braket.program_sets.circuit_binding import CircuitBinding
from braket.program_sets import ProgramSet

# Initialize the quantum device
device = AwsDevice("arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet")
```

```
# Define circuits
circ1 = Circuit().h(0).cnot(0, 1)
circ2 = Circuit().rx(0, 0.785).ry(1, 0.393).cnot(1, 0)
circ3 = Circuit().t(0).t(1).cz(0, 1).s(0).cz(1, 2).s(1).s(2)
parameterize_circuit = Circuit().rx(0, FreeParameter("alpha")).cnot(0, 1).ry(1,
 FreeParameter("beta"))

# Create circuit bindings with different parameters
circuit_binding = CircuitBinding(
    circuit=parameterize_circuit,
    input_sets={
            'alpha': (0.10, 0.11, 0.22, 0.34, 0.45),
            'beta': (1.01, 1.01, 1.03, 1.04, 1.04),
    })

# Creating the program set
program_set_1 = ProgramSet([
    circ1,
    circ2,
    circ3,
    circuit_binding,
])
```

Questo set di programmi contiene quattro programmi unici:`circ1`,, e. `circ2` `circ3` `circuit_binding` Il `circuit_binding` programma viene eseguito con cinque diverse associazioni di parametri, creando cinque eseguibili. Gli altri tre programmi senza parametri creano ciascuno un file eseguibile. Il risultato è un totale di otto eseguibili, come illustrato nell'immagine seguente.




```
ProgramSet(
    [c1,
     c2,
     c3,
     CircuitBinding(c4, [i1, i2, i3, i4, i5])]
)
```

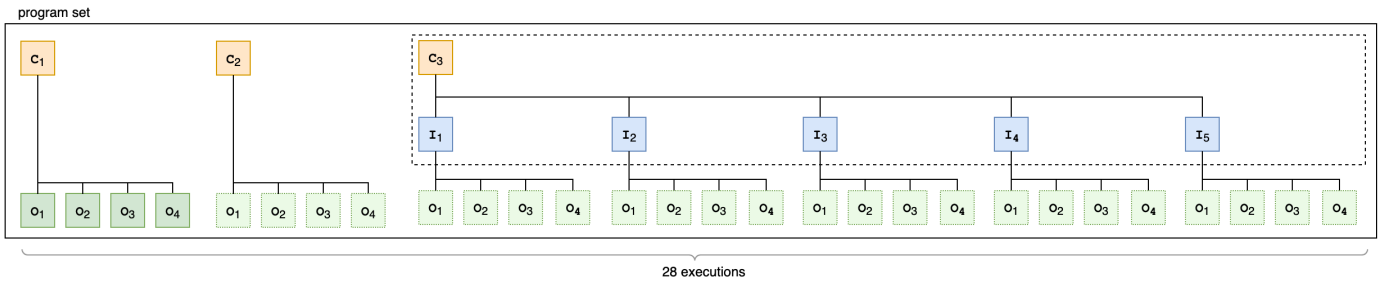
Il secondo esempio di codice seguente mostra come utilizzare il `product()` metodo per collegare lo stesso set di osservabili a ciascun eseguibile del set di programmi.

```
from braket.circuits.observables import I, X, Y, Z
```

```
observables = [Z(0) @ Z(1), X(0) @ X(1), Z(0) @ X(1), X(0) @ Z(1)]

program_set_2 = ProgramSet.product(
    circuits=[circ1, circ2, circuit_binding],
    observables=observables
)
```

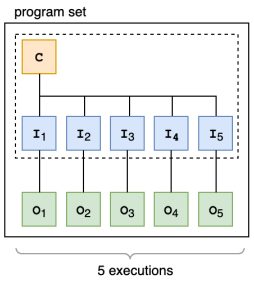
Per i programmi senza parametri, ogni osservabile viene misurato per ogni circuito. Per i programmi parametrici, ogni osservabile viene misurato per ogni set di input, come mostrato nell'immagine seguente.

```
ProgramSet.product(
    circuits=[c1, c2, CircuitBinding(c3, [i1, i2, i3, i4, i5])],
    observables=[o1, o2, o3, o4]
)
```



Il terzo esempio di codice seguente mostra come utilizzare il `zip()` metodo per accoppiare singoli osservabili con set di parametri specifici in. `ProgramSet`

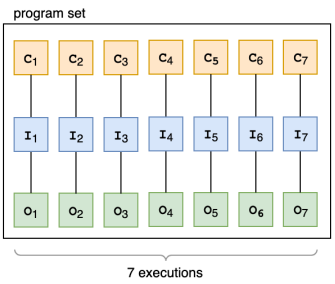
```
program_set_3 = ProgramSet.zip(
    circuits=circuit_binding,
    observables=observables + [Y(0) @ Y(1)]
)
```




```
ProgramSet.zip(
    circuits=CicruitBinding(
        circuit=c,
        input_sets=[i1, i2, i3, i4, i5]
    ),
    observables=[o1, o2, o3, o4, o5]
)
```

Invece`CircuitBinding()`, puoi comprimere direttamente un elenco di osservabili con un elenco di circuiti e set di input.

```
program_set_4 = ProgramSet.zip(
    circuits=[circ1, circ2, circ3],
    input_sets=[{}, {}, {}],
    observables=observables[:3]
)
```




```
ProgramSet.zip(
    circuits=[c1, c2, c3, c4, c5, c6, c7],
    input_sets=[i1, i2, i3, i4, i5, i6, i7],
    observables=[o1, o2, o3, o4, o5, o6, o7]
)
```

Per ulteriori informazioni ed esempi sui set di programmi, consulta la cartella Program set in Github. amazon-braket-examples

Ispeziona ed esegui un programma impostato su un dispositivo

Il numero di file eseguibili di un set di programmi è uguale al numero di circuiti univoci legati a parametri. Calcola il numero totale di eseguibili e scatti del circuito utilizzando il seguente esempio di codice.

```
# Number of shots per executable
shots = 10
num_executables = program_set_1.total_executables

# Calculate total number of shots across all executables
total_num_shots = shots*num_executables
```

> **Note**
>
> Con i set di programmi, si paga una tariffa singola per operazione e una tariffa per ripresa in base al numero totale di scatti su tutti i circuiti di un set di programmi.

Per eseguire il set di programmi, utilizzate il seguente esempio di codice.

```
# Run the program set
```

```
task = device.run(
   program_set_1, shots=total_num_shots,
)
```

Quando si utilizzano Rigetti dispositivi, il set di programmi può rimanere nello `RUNNING` stato mentre le attività sono parzialmente terminate e parzialmente in coda. Per risultati più rapidi, valuta la possibilità di inviare il programma impostato come Hybrid Job.

Analisi dei risultati

Esegui il codice seguente per analizzare e misurare i risultati degli eseguibili in a. `ProgramSet`

```
# Get the results from a program set
result = task.result()

# Get the first executbable
first_program = result[0]
first_executable = first_program[0]

# Inspect the results of the first executable
measurements_from_first_executable = first_executable.measurements
print(measurements_from_first_executable)
```

Misurazione parziale

Invece di misurare tutti i qubit in un circuito quantistico, utilizzate la misurazione parziale per misurare singoli qubit o un sottoinsieme di qubit.

> **Note**
>
> Funzionalità aggiuntive come la misurazione a livello centrale del circuito e le operazioni di feed-forward sono disponibili come funzionalità sperimentali, vedi Accesso ai circuiti dinamici sui dispositivi IQM.

Esempio: misura un sottoinsieme di qubit

Il seguente esempio di codice dimostra la misurazione parziale misurando solo il qubit 0 in un circuito a stato Bell.

```
from braket.devices import LocalSimulator
```

```
from braket.circuits import Circuit

# Use the local state vector simulator
device = LocalSimulator()

# Define an example bell circuit and measure qubit 0
circuit = Circuit().h(0).cnot(0, 1).measure(0)

# Run the circuit
task = device.run(circuit, shots=10)

# Get the results
result = task.result()

# Print the circuit and measured qubits
print(circuit)
print()
print("Measured qubits: ", result.measured_qubits)
```

Allocazione manuale qubit

Quando esegui un circuito quantistico su computer quantistici daRigetti, puoi opzionalmente utilizzare l'qubitallocazione manuale per controllare quali qubits vengono utilizzati per il tuo algoritmo. Amazon Braket Console e Amazon Braket SDK ti aiutano a ispezionare i dati di calibrazione più recenti del dispositivo QPU (Quantum Processing Unit) selezionato, in modo da poter scegliere il migliore per il tuo esperimento. qubits

L'qubitallocazione manuale consente di far funzionare i circuiti con maggiore precisione e di esaminare le singole proprietà. qubit I ricercatori e gli utenti esperti ottimizzano la progettazione dei circuiti sulla base dei più recenti dati di calibrazione dei dispositivi e possono ottenere risultati più accurati.

L'esempio seguente mostra come qubits allocare in modo esplicito.

```
# Import the device module
from braket.aws import AwsDevice

device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")
circ = Circuit().h(0).cnot(0, 7)  # Indices of actual qubits in the QPU

# Set up S3 bucket (where results are stored)
my_bucket = "amazon-braket-s3-demo-bucket" # The name of the bucket
```

```
my_prefix = "your-folder-name" # The name of the folder in the bucket
s3_location = (my_bucket, my_prefix)

my_task = device.run(circ, s3_location, shots=100, disable_qubit_rewiring=True)
```

Per ulteriori informazioni, consulta gli esempi di Amazon Braket su GitHub, o più specificamente, su questo notebook: Allocazione di qubit su dispositivi QPU.

Compilazione Verbatim

Quando si esegue un circuito quantistico su computer quantistici basati su gate, è possibile ordinare al compilatore di eseguire i circuiti esattamente come definito senza alcuna modifica. Utilizzando la compilazione letterale, è possibile specificare che un intero circuito venga preservato esattamente come specificato o che vengano conservate solo parti specifiche di esso (supportata solo da). Rigetti Quando si sviluppano algoritmi per il benchmarking dell'hardware o i protocolli di mitigazione degli errori, è necessario avere la possibilità di specificare esattamente le porte e i layout dei circuiti in esecuzione sull'hardware. La compilazione Verbatim offre il controllo diretto sul processo di compilazione disattivando alcune fasi di ottimizzazione, garantendo così che i circuiti funzionino esattamente come previsto.

La compilazione Verbatim è supportata sui Rigetti dispositiviAQT, IonQIQM, e e richiede l'uso di porte native. Quando si utilizza la compilazione verbatim, è consigliabile controllare la topologia del dispositivo per assicurarsi che le porte vengano richiamate qubits connesse e che il circuito utilizzi le porte native supportate dall'hardware. L'esempio seguente mostra come accedere a livello di codice all'elenco delle porte native supportate da un dispositivo.

```
device.properties.paradigm.nativeGateSet
```

InfattiRigetti, il qubit ricablaggio deve essere disattivato impostandolo `disableQubitRewiring=True` per l'uso con la compilazione letterale. Se `disableQubitRewiring=False` è impostato quando si usano le caselle letterali in una compilazione, il circuito quantistico fallisce la convalida e non viene eseguito.

Se la compilazione verbatim è abilitata per un circuito ed è eseguita su una QPU che non la supporta, viene generato un errore che indica che un'operazione non supportata ha causato il fallimento dell'operazione. Poiché sempre più hardware quantistico supporta nativamente le funzioni del compilatore, questa funzionalità verrà ampliata per includere questi dispositivi. I dispositivi che supportano la compilazione letterale la includono come operazione supportata quando viene richiesta con il codice seguente.

```
from braket.aws import AwsDevice
from braket.device_schema.device_action_properties import DeviceActionType
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")
device.properties.action[DeviceActionType.OPENQASM].supportedPragmas
```

Non sono previsti costi aggiuntivi associati all'utilizzo della compilazione letterale. Continueranno a essere addebitati i costi per le attività quantistiche eseguite su dispositivi Braket QPU, istanze notebook e simulatori on-demand in base alle tariffe correnti, come specificato nella pagina dei prezzi di Amazon Braket. Per ulteriori informazioni, consulta il taccuino di esempio della compilazione Verbatim.

> **Note**
>
> Se si utilizza OpenQASM per scrivere i circuiti per IonQ i dispositivi AQT and e si desidera mappare il circuito direttamente sui qubit fisici, è necessario utilizzare il flag poiché il flag viene ignorato da OpenQASM. `#pragma braket verbatim` `disableQubitRewiring`

Simulazione del rumore

Per creare un'istanza del simulatore di rumore locale è possibile modificare il backend come segue.

```
# Import the device module
from braket.aws import AwsDevice

device = LocalSimulator(backend="braket_dm")
```

È possibile creare circuiti rumorosi in due modi:

1. Costruisci il circuito rumoroso dal basso verso l'alto.
2. Prendi un circuito esistente e privo di rumore e inietta rumore dappertutto.

L'esempio seguente mostra gli approcci che utilizzano un circuito di base con rumore depolarizzante e un canale Kraus personalizzato.

```
import scipy.stats
import numpy as np

# Bottom up approach
```

```
# Apply depolarizing noise to qubit 0 with probability of 0.1
circ = Circuit().x(0).x(1).depolarizing(0, probability=0.1)

# Create an arbitrary 2-qubit Kraus channel
E0 = scipy.stats.unitary_group.rvs(4) * np.sqrt(0.8)
E1 = scipy.stats.unitary_group.rvs(4) * np.sqrt(0.2)
K = [E0, E1]

# Apply a two-qubit Kraus channel to qubits 0 and 2
circ = circ.kraus([0, 2], K)
```

```
from braket.circuits import Noise

# Inject noise approach
# Define phase damping noise
noise = Noise.PhaseDamping(gamma=0.1)
# The noise channel is applied to all the X gates in the circuit
circ = Circuit().x(0).y(1).cnot(0, 2).x(1).z(2)
circ_noise = circ.copy()
circ_noise.apply_gate_noise(noise, target_gates=Gate.X)
```

L'utilizzo di un circuito è la stessa esperienza utente di prima, come illustrato nei due esempi seguenti.

Esempio 1

```
task = device.run(circ, shots=100)
```

Or

Esempio 2

```
task = device.run(circ_noise, shots=100)
```

Per altri esempi, vedi [l'esempio introduttivo del simulatore di rumore Braket]

Ispezione del circuito

I circuiti quantistici di Amazon Braket hanno un concetto di pseudo-tempo chiamato. `Moments` Ognuno qubit può sperimentare un solo gate per. `Moment` Lo scopo `Moments` è rendere i circuiti e le loro porte più facili da indirizzare e fornire una struttura temporale.

> **Note**
>
> I momenti generalmente non corrispondono al tempo reale in cui i gate vengono eseguiti su una QPU.

La profondità di un circuito è data dal numero totale di Momenti in quel circuito. È possibile visualizzare la profondità del circuito chiamando il metodo `circuit.depth` come mostrato nell'esempio seguente.

```
from braket.circuits import Circuit

# Define a circuit with parametrized gates
circ = Circuit().rx(0, 0.15).ry(1, 0.2).cnot(0, 2).zz(1, 3, 0.15).x(0)
print(circ)
print('Total circuit depth:', circ.depth)
```

```
T  : #     0      #        1         #  2  #
      ############                    #####
q0 : ## Rx(0.15) ###################### X ##
      ############   #                #####
      ############   #   ############
q1 : ## Ry(0.20) ######## ZZ(0.15) ########
      ############   #   ############
                  #####      #
q2 : ############### X ######################
                  #####      #
                        ############
q3 : ##################### ZZ(0.15) ########
                        ############
T  : #     0      #        1         #  2  #
Total circuit depth: 3
```

La profondità totale del circuito precedente è 3 (mostrata come momenti `0`1, e2). Puoi controllare il funzionamento del cancello per ogni momento.

`Momentsfunziona` come un dizionario di coppie chiave-valore.

- La chiave è`MomentsKey()`, che contiene pseudo-tempo e informazioni. qubit
- Il valore viene assegnato nel tipo di. `Instructions()`

```
moments = circ.moments
for key, value in moments.items():
    print(key)
    print(value, "\n")
```

```
MomentsKey(time=0, qubits=QubitSet([Qubit(0)]), moment_type=<MomentType.GATE: 'gate'>,
 noise_index=0, subindex=0)
Instruction('operator': Rx('angle': 0.15, 'qubit_count': 1), 'target':
 QubitSet([Qubit(0)]), 'control': QubitSet([]), 'control_state': (), 'power': 1)

MomentsKey(time=0, qubits=QubitSet([Qubit(1)]), moment_type=<MomentType.GATE: 'gate'>,
 noise_index=0, subindex=0)
Instruction('operator': Ry('angle': 0.2, 'qubit_count': 1), 'target':
 QubitSet([Qubit(1)]), 'control': QubitSet([]), 'control_state': (), 'power': 1)

MomentsKey(time=1, qubits=QubitSet([Qubit(0), Qubit(2)]), moment_type=<MomentType.GATE:
 'gate'>, noise_index=0, subindex=0)
Instruction('operator': CNot('qubit_count': 2), 'target': QubitSet([Qubit(0),
 Qubit(2)]), 'control': QubitSet([]), 'control_state': (), 'power': 1)

MomentsKey(time=1, qubits=QubitSet([Qubit(1), Qubit(3)]), moment_type=<MomentType.GATE:
 'gate'>, noise_index=0, subindex=0)
Instruction('operator': ZZ('angle': 0.15, 'qubit_count': 2), 'target':
 QubitSet([Qubit(1), Qubit(3)]), 'control': QubitSet([]), 'control_state': (), 'power':
 1)

MomentsKey(time=2, qubits=QubitSet([Qubit(0)]), moment_type=<MomentType.GATE: 'gate'>,
 noise_index=0, subindex=0)
Instruction('operator': X('qubit_count': 1), 'target': QubitSet([Qubit(0)]), 'control':
 QubitSet([]), 'control_state': (), 'power': 1)
```

È inoltre possibile aggiungere porte a un circuito tramite`Moments`.

```
from braket.circuits import Instruction, Gate

new_circ = Circuit()
instructions = [Instruction(Gate.S(), 0),
                Instruction(Gate.CZ(), [1, 0]),
                Instruction(Gate.H(), 1)
                ]

new_circ.moments.add(instructions)
```

```
print(new_circ)
```

```
T  : #  0  #  1  #  2  #
      ##### #####
q0 : ## S ### Z ########
      ##### #####
              #   #####
q1 : ############## H ##
                  #####
T  : #  0  #  1  #  2  #
```

Elenco dei tipi di risultati

Amazon Braket può restituire diversi tipi di risultati quando un circuito viene misurato utilizzando. `ResultType` Un circuito può restituire i seguenti tipi di risultati.

- `AdjointGradient`restituisce il gradiente (derivata vettoriale) del valore atteso di un osservabile fornito. Questo osservabile agisce su un obiettivo fornito rispetto a parametri specificati utilizzando il metodo di differenziazione aggiuntiva. Puoi usare questo metodo solo quando shots=0.
- `Amplitude`restituisce l'ampiezza degli stati quantistici specificati nella funzione d'onda di uscita. È disponibile solo sui simulatori SV1 e sui simulatori locali.
- `Expectation`restituisce il valore di aspettativa di un dato osservabile, che può essere specificato con la `Observable` classe introdotta più avanti in questo capitolo. L'obiettivo qubits utilizzato per misurare l'osservabile deve essere specificato e il numero di obiettivi specificati deve essere uguale al numero qubits su cui agisce l'osservabile. Se non viene specificato alcun obiettivo, l'osservabile deve operare solo su 1 qubit e viene applicato a tutti qubits in parallelo.
- `Probability`restituisce le probabilità di misurare gli stati di base computazionali. Se non viene specificato alcun obiettivo, `Probability` restituisce la probabilità di misurare tutti gli stati di base. Se vengono specificati obiettivi, vengono restituite solo le probabilità marginali dei vettori di base sull'oggetto specificatoqubits. I simulatori gestiti QPUs sono limitati a un massimo di 15 qubit, mentre i simulatori locali sono limitati alla dimensione della memoria del sistema.
- `Reduced density matrix`restituisce una matrice di densità per un sottosistema di un obiettivo qubits specificato da un sistema di. qubits Per limitare la dimensione di questo tipo di risultato, Braket limita il numero di obiettivi qubits a un massimo di 8.
- `StateVector`restituisce il vettore di stato completo. È disponibile sul simulatore locale.
- `Sample`restituisce i conteggi delle misurazioni di un qubit set di obiettivi specificato e osservabili. Se non viene specificato alcun obiettivo, l'osservabile deve operare solo su 1 qubit e viene

applicato a tutti qubits in parallelo. Se vengono specificati obiettivi, il numero di obiettivi specificati deve essere uguale al numero qubits su cui agisce l'osservabile.

- `Variance`restituisce la variance (`mean([x-mean(x)]`^{2}`)`) del qubit set di obiettivi specificato e osservabile come tipo di risultato richiesto. Se non viene specificato alcun obiettivo, l'osservabile deve operare solo su 1 qubit e viene applicato a tutti qubits in parallelo. Altrimenti, il numero di obiettivi specificati deve essere uguale al numero qubits a cui può essere applicato l'osservabile.

I tipi di risultati supportati per diversi provider:

	Sim locale	SV1	DM1	TN1	AQT	IonQ	IQM	Rigetti
Gradient aggiunto	N	Y	N	N	N	N	N	N
Amplitud	Y	Y	N	N	N	N	N	N
Aspettativa	Y	Y	Y	Y	Y	Y	Y	Y
Probability	Y	Y	Y	N	Y	Y	Y	Y
Matrice a densità ridotta	Y	N	Y	N	N	N	N	N
Vettore di stato	Y	N	N	N	N	N	N	N
Project N.E.M.O	Y	Y	Y	Y	Y	Y	Y	Y
Varianza	Y	Y	Y	Y	Y	Y	Y	Y

È possibile verificare i tipi di risultati supportati esaminando le proprietà del dispositivo, come illustrato nell'esempio seguente.

```
from braket.aws import AwsDevice

device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")

# Print the result types supported by this device
for iter in
 device.properties.action['braket.ir.openqasm.program'].supportedResultTypes:
    print(iter)
```

```
name='Sample' observables=['x', 'y', 'z', 'h', 'i'] minShots=10 maxShots=50000
name='Expectation' observables=['x', 'y', 'z', 'h', 'i'] minShots=10 maxShots=50000
name='Variance' observables=['x', 'y', 'z', 'h', 'i'] minShots=10 maxShots=50000
name='Probability' observables=None minShots=10 maxShots=50000
```

Per chiamare a`ResultType`, aggiungetelo a un circuito, come mostrato nell'esempio seguente.

```
from braket.circuits import Circuit, Observable

circ = Circuit().h(0).cnot(0, 1).amplitude(state=["01", "10"])
circ.probability(target=[0, 1])
circ.probability(target=0)
circ.expectation(observable=Observable.Z(), target=0)
circ.sample(observable=Observable.X(), target=0)
circ.state_vector()
circ.variance(observable=Observable.Z(), target=0)

# Print one of the result types assigned to the circuit
print(circ.result_types[0])
```

> **Note**
>
> Diversi dispositivi quantistici forniscono risultati in vari formati. Ad esempio, Rigetti i dispositivi restituiscono le misurazioni, mentre IonQ i dispositivi forniscono le probabilità. L'SDK Amazon Braket offre una proprietà di misurazione per tutti i risultati. Tuttavia, per i dispositivi che restituiscono probabilità, queste misurazioni vengono post-calcolate e basate sulle probabilità, poiché le misurazioni per colpo non sono disponibili. Per determinare se un risultato è stato post-calcolato, controllate l'oggetto del risultato.

`measurements_copied_from_device` Questa operazione è dettagliata nel file gate_model_quantum_task_result.py nel repository Amazon Braket SDK GitHub .

Osservabili

La `Observable` classe di Amazon Braket ti consente di misurare un osservabile specifico.

Puoi applicare solo un osservabile univoco non identitario a ciascuno. qubit Si verifica un errore se si specificano due o più osservabili non di identità diversi sullo stesso. qubit A tal fine, ogni fattore di un prodotto tensoriale conta come osservabile individuale. Ciò significa che è possibile avere più prodotti tensoriali sullo stessoqubit, purché i fattori che agiscono su di essi rimangano gli stessi. qubit

Un osservabile può essere ridimensionato e aggiungere altri osservabili (ridimensionati o meno). Questo crea un `Sum` che può essere utilizzato nel tipo di risultato. `AdjointGradient`

La `Observable` classe include i seguenti osservabili.

```
import numpy as np

Observable.I()
Observable.H()
Observable.X()
Observable.Y()
Observable.Z()

# Get the eigenvalues of the observable
print("Eigenvalue:", Observable.H().eigenvalues)
# Or rotate the basis to be computational basis
print("Basis rotation gates:", Observable.H().basis_rotation_gates)

# Get the tensor product of the observable for the multi-qubit case
tensor_product = Observable.Y() @ Observable.Z()
# View the matrix form of an observable by using
print("The matrix form of the observable:\n", Observable.Z().to_matrix())
print("The matrix form of the tensor product:\n", tensor_product.to_matrix())

# Factorize an observable in the tensor form
print("Factorize an observable:", tensor_product.factors)

# Self-define observables, given it is a Hermitian
```

```
print("Self-defined Hermitian:", Observable.Hermitian(matrix=np.array([[0, 1], [1,
 0]])))

print("Sum of other (scaled) observables:", 2.0 * Observable.X() @ Observable.X() + 4.0
 * Observable.Z() @ Observable.Z())
```

```
Eigenvalue: [ 1. -1.]
Basis rotation gates: (Ry('angle': -0.7853981633974483, 'qubit_count': 1),)
The matrix form of the observable:
 [[ 1.+0.j  0.+0.j]
 [ 0.+0.j -1.+0.j]]
The matrix form of the tensor product:
 [[ 0.+0.j  0.+0.j  0.-1.j  0.+0.j]
 [ 0.+0.j -0.+0.j  0.+0.j  0.+1.j]
 [ 0.+1.j  0.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.-1.j  0.+0.j -0.+0.j]]
Factorize an observable: (Y('qubit_count': 1), Z('qubit_count': 1))
Self-defined Hermitian: Hermitian('qubit_count': 1, 'matrix': [[0.+0.j 1.+0.j], [1.+0.j
 0.+0.j]])
Sum of other (scaled) observables: Sum(TensorProduct(X('qubit_count': 1),
 X('qubit_count': 1)), TensorProduct(Z('qubit_count': 1), Z('qubit_count': 1)))
```

Parameters

I circuiti possono incorporare parametri liberi. Questi parametri liberi devono essere costruiti una sola volta per essere eseguiti più volte e possono essere utilizzati per calcolare i gradienti.

Ogni parametro gratuito utilizza un nome codificato in una stringa che viene utilizzato per:

- Impostare i valori dei parametri
- Identifica i parametri da utilizzare

```
from braket.circuits import Circuit, FreeParameter, observables
from braket.parametric import FreeParameter

theta = FreeParameter("theta")
phi = FreeParameter("phi")
circ = Circuit().h(0).rx(0, phi).ry(0, phi).cnot(0, 1).xx(0, 1, theta)
```

Aggiungi gradiente

Il SV1 dispositivo calcola il gradiente aggiunto di un valore di aspettativa osservabile, incluso l'hamiltoniano multitermine. Per differenziare i parametri, specificate il loro nome (in formato stringa) o tramite riferimento diretto.

```
from braket.aws import AwsDevice
from braket.devices import Devices

device = AwsDevice(Devices.Amazon.SV1)

circ.adjoint_gradient(observable=3 * Observable.Z(0) @ Observable.Z(1) - 0.5 *
 observables.X(0), parameters = ["phi", theta])
```

Il passaggio di valori di parametri fissi come argomenti a un circuito parametrizzato rimuoverà i parametri liberi. L'esecuzione di questo circuito con `AdjointGradient` produce un errore, poiché i parametri liberi non esistono più. Il seguente esempio di codice dimostra l'utilizzo corretto e scorretto:

```
# Will error, as no free parameters will be present
#device.run(circ(0.2), shots=0)

# Will succeed
device.run(circ, shots=0, inputs={'phi': 0.2, 'theta': 0.2})
```

Ottenere la consulenza di un esperto

Connect con esperti di quantum computing direttamente nella console di gestione Braket per ottenere ulteriori indicazioni sui carichi di lavoro.

Per esplorare le opzioni di consulenza degli esperti tramite Braket Direct, apri la console Braket, scegli Braket Direct nel riquadro di sinistra e vai alla sezione Consigli degli esperti. Sono disponibili le seguenti opzioni di consulenza degli esperti:

- Orari di ufficio Braket: gli orari di ufficio Braket prevedono sessioni individuali, assegnate in base all'ordine di arrivo e si svolgono ogni mese. Ogni fascia oraria d'ufficio disponibile dura 30 minuti ed è gratuita. Parlare con gli esperti di Braket può aiutarti a passare più rapidamente dall'ideazione all'esecuzione esplorando l' use-case-to-deviceadattamento, identificando le opzioni per utilizzare al meglio Braket per il tuo algoritmo e ricevendo consigli su come utilizzare determinate funzionalità di Braket come Amazon Braket Hybrid Jobs, Braket Pulse o Analog Hamiltonian Simulation.

- Per iscriverti agli orari d'ufficio di Braket, seleziona Iscriviti e inserisci le informazioni di contatto, i dettagli del carico di lavoro e gli argomenti di discussione desiderati.
- Riceverai un invito sul calendario per il prossimo slot disponibile tramite la tua e-mail.

> **Note**
>
> Per problemi emergenti o domande rapide sulla risoluzione dei problemi, ti consigliamo di contattare l'AWS assistenza. Per domande non urgenti, puoi anche utilizzare il forum AWS re:POST o il Quantum Computing Stack Exchange, dove puoi sfogliare le domande a cui hai risposto in precedenza e porne di nuove.

- Offerte di fornitori di hardware quantistico: IonQQuEra, e Rigetti ciascuno fornisce servizi professionali tramite. Marketplace AWS
  - Per esplorare le loro offerte, seleziona Connect e sfoglia le loro inserzioni.
  - Per ulteriori informazioni sulle offerte di servizi professionali disponibili su Marketplace AWS, consulta Prodotti per servizi professionali.
- AmazonQuantum Solutions Lab (QSL): il QSL è un team collaborativo di ricerca e servizi professionali composto da esperti di informatica quantistica che possono aiutarti a esplorare efficacemente l'informatica quantistica e valutare le prestazioni attuali di questa tecnologia.
  - Per contattare il QSL, seleziona Connect e inserisci le informazioni di contatto e i dettagli del caso d'uso.
  - Il team QSL ti contatterà tramite e-mail per indicarti i passaggi successivi.

Esegui i tuoi circuiti con OpenQASM 3.0

AmazonBraket ora supporta OpenQASM 3.0 per dispositivi e simulatori quantistici basati su gate. Questa guida per l'utente fornisce informazioni sul sottoinsieme di OpenQASM 3.0 supportato da Braket. I clienti di Braket ora possono scegliere di inviare i circuiti Braket con l'SDK o fornire direttamente le stringhe OpenQASM 3.0 a tutti i dispositivi basati su gate con l'API Amazon Braket e l'SDK Amazon Braket Python.

Gli argomenti di questa guida illustrano vari esempi di come completare le seguenti attività quantistiche.

- Crea e invia attività quantistiche OpenQASM su diversi dispositivi Braket
- Accedi alle operazioni e ai tipi di risultati supportati

- Simula il rumore con OpenQASM
- Usa la compilazione letterale con OpenQASM
- Risolvi i problemi relativi a OpenQASM

Questa guida fornisce anche un'introduzione ad alcune funzionalità specifiche dell'hardware che possono essere implementate con OpenQASM 3.0 su Braket e collegamenti a ulteriori risorse.

In questa sezione:

- Che cos'è OpenQASM 3.0?
- Quando usare OpenQASM 3.0
- Come funziona OpenQASM 3.0
- Prerequisiti
- Quali funzionalità di OpenQASM supporta Braket?
- Crea e invia un esempio di task quantistico OpenQASM 3.0
- Support per OpenQASM su diversi dispositivi Braket
- Simula il rumore con OpenQASM 3.0
- Qubitricablaggio con OpenQASM 3.0
- Compilazione Verbatim con OpenQASM 3.0
- La console Braket
- Risorse aggiuntive
- Calcolo dei gradienti con OpenQASM 3.0
- Misurazione di qubit specifici con OpenQASM 3.0

Che cos'è OpenQASM 3.0?

L'Open Quantum Assembly Language (openQASM) è una rappresentazione intermedia per le istruzioni quantistiche. OpenQASM è un framework open source ed è ampiamente utilizzato per la specifica di programmi quantistici per dispositivi basati su gate. Con OpenQASM, gli utenti possono programmare le porte quantistiche e le operazioni di misurazione che costituiscono gli elementi costitutivi del calcolo quantistico. La versione precedente di OpenQASM (2.0) è stata utilizzata da diverse librerie di programmazione quantistica per descrivere i programmi di base.

La nuova versione di OpenQASM (3.0) estende la versione precedente per includere più funzionalità, come il controllo a livello di impulsi, il gate timing e il flusso di controllo classico per colmare il divario

tra l'interfaccia utente finale e il linguaggio di descrizione dell'hardware. I dettagli e le specifiche sulla versione corrente 3.0 sono disponibili nella OpenQASM 3.x Live Specification. GitHub Lo sviluppo futuro di OpenQASM è governato dal Comitato direttivo tecnico di OpenQASM 3.0, di cui AWS è membro insieme a IBM, Microsoft e all'Università di Innsbruck.

Quando usare OpenQASM 3.0

OpenQASM fornisce un framework espressivo per specificare programmi quantistici attraverso controlli di basso livello che non sono specifici dell'architettura, il che lo rende adatto come rappresentazione su più dispositivi basati su gate. Il supporto Braket per OpenQASM ne promuove l'adozione come approccio coerente allo sviluppo di algoritmi quantistici basati su gate, riducendo la necessità per gli utenti di apprendere e gestire librerie in più framework.

Se disponi di librerie di programmi esistenti in OpenQASM 3.0, puoi adattarle per l'uso con Braket anziché riscrivere completamente questi circuiti. I ricercatori e gli sviluppatori dovrebbero inoltre trarre vantaggio da un numero crescente di librerie di terze parti disponibili con supporto per lo sviluppo di algoritmi in OpenQASM.

Come funziona OpenQASM 3.0

Il supporto per OpenQASM 3.0 di Braket fornisce la parità di funzionalità con l'attuale Intermediate Representation. Ciò significa che tutto ciò che puoi fare oggi su dispositivi hardware e simulatori on-demand con Braket, puoi farlo con OpenQASM usando Braket. API È possibile eseguire i programmi OpenQASM 3.0 fornendo direttamente le stringhe OpenQASM a tutti i dispositivi basati su gate in un modo simile a come i circuiti vengono attualmente forniti ai dispositivi su Braket. Gli utenti di Braket possono anche integrare librerie di terze parti che supportano OpenQASM 3.0. Il resto di questa guida descrive in dettaglio come sviluppare rappresentazioni OpenQASM da utilizzare con Braket.

Prerequisiti

Per utilizzare OpenQASM 3.0 su Amazon Braket, è necessario disporre della versione v1.8.0 degli schemi Amazon Braket Python e v1.17.0 o successiva dell'SDK Amazon Braket Python.

Se utilizzi Amazon Braket per la prima volta, devi abilitare Amazon Braket. Per istruzioni, consulta Abilita Amazon Braket.

Quali funzionalità di OpenQASM supporta Braket?

La sezione seguente elenca i tipi di dati, le istruzioni e le istruzioni pragma di OpenQASM 3.0 supportate da Braket.

In questa sezione:

Tipi di dati OpenQASM supportati

I seguenti tipi di dati OpenQASM sono supportati da Amazon Braket.

- I numeri interi non negativi vengono utilizzati per gli indici di qubit (virtuali e fisici):
  - `cnot q[0], q[1];`
  - `h $0;`
- È possibile utilizzare numeri o costanti a virgola mobile per gli angoli di rotazione del cancello:
  - `rx(-0.314) $0;`
  - `rx(pi/4) $0;`

> **Note**
>
> pi è una costante incorporata in OpenQASM e non può essere utilizzata come nome di parametro.

- Le matrici di numeri complessi (con la `im` notazione OpenQASM per la parte immaginaria) sono consentite nei pragmi di tipo di risultato per la definizione di osservabili hermitiane generali e nei pragmi unitari:
  - `#pragma braket unitary [[0, -1im], [1im, 0]] q[0]`
  - `#pragma braket result expectation hermitian([[0, -1im], [1im, 0]]) q[0]`

Istruzioni OpenQASM supportate

Le seguenti istruzioni OpenQASM sono supportate da Amazon Braket.

- `Header: OPENQASM 3;`
- Dichiarazioni di bit classiche:
  - `bit b1;`(equivalentemente,) `creg b1;`
  - `bit[10] b2;`(equivalentemente,) `creg b2[10];`
- Dichiarazioni Qubit:
  - `qubit b1;`(equivalentemente,) `qreg b1;`
  - `qubit[10] b2;`(equivalentemente,) `qreg b2[10];`
- Indicizzazione all'interno di array: `q[0]`
- Ingresso: `input float alpha;`
- specificazione fisicaqubits: `$0`
- Porte e operazioni supportate su un dispositivo:
  - `h $0;`
  - `iswap q[0], q[1];`

> **Note**
>
> Le porte supportate da un dispositivo sono disponibili nelle proprietà del dispositivo per le azioni OpenQASM; non sono necessarie definizioni di gate per utilizzare queste porte.

- Dichiarazioni letterali. Al momento, non supportiamo la notazione della durata delle caselle. Le porte native e fisiche qubits sono obbligatorie nelle caselle letterali.

```
#pragma braket verbatim
box{
    rx(0.314) $0;
}
```

- Misurazione e assegnazione delle misurazioni su qubits o su un intero registro. qubit
  - `measure $0;`
  - `measure q;`
  - `measure q[0];`
  - `b = measure q;`

  - `measure q # b;`
- Le istruzioni Barrier forniscono un controllo esplicito sulla compilazione e l'esecuzione dei circuiti impedendo il riordino e l'ottimizzazione dei gate attraverso i confini delle barriere. Impongono inoltre un rigoroso ordinamento temporale durante l'esecuzione, garantendo che tutte le operazioni prima che una barriera sia completata prima dell'inizio delle operazioni successive.
  - `barrier;`
  - `barrier q[0], q[1];`
  - `barrier $3, $6;`

Pragmi Braket OpenQASM

Le seguenti istruzioni relative al pragma OpenQASM sono supportate da Amazon Braket.

- Pragmi del rumore
  - `#pragma braket noise bit_flip(0.2) q[0]`
  - `#pragma braket noise phase_flip(0.1) q[0]`
  - `#pragma braket noise pauli_channel`
- Pragmi letterali
  - `#pragma braket verbatim`
- Tipo di risultato: pragmi
  - Tipi di risultati invarianti di base:
    - Vettore di stato: `#pragma braket result state_vector`
    - Matrice di densità: `#pragma braket result density_matrix`
  - Pragmi di calcolo del gradiente:
    - Gradiente aggiunto: `#pragma braket result adjoint_gradient expectation(2.2 * x[0] @ x[1]) all`
  - Tipi di risultati di base Z:
    - Ampiezza: `#pragma braket result amplitude "01"`
    - Probabilità: `#pragma braket result probability q[0], q[1]`
  - Tipi di risultati ruotati di base
    - Aspettativa: `#pragma braket result expectation x(q[0]) @ y([q1])`

- Varianza: `#pragma braket result variance hermitian([[0, -1im], [1im, 0]]) $0`
- Esempio: `#pragma braket result sample h($1)`

> **Note**
>
> OpenQASM 3.0 è retrocompatibile con OpenQASM 2.0, quindi i programmi scritti utilizzando 2.0 possono essere eseguiti su Braket. Tuttavia, le funzionalità di OpenQASM 3.0 supportate da Braket presentano alcune differenze di sintassi minori, come vs e vs. `qreg` `creg` `qubit` `bit` Esistono anche differenze nella sintassi di misurazione, che devono essere supportate con la sintassi corretta.

Supporto di funzionalità avanzate per OpenQASM su Local Simulator

`LocalSimulator`Supporta funzionalità avanzate di OpenQASM che non sono offerte come parte delle QPU o dei simulatori on-demand di Braket. Il seguente elenco di funzionalità è supportato solo in: `LocalSimulator`

- Modificatori di gate
- Porte integrate in OpenQASM
- Variabili classiche
- Operazioni classiche
- Cancelli personalizzati
- Controllo classico
- File QASM
- Sottoroutine

Per esempi di ogni funzionalità avanzata, consultate questo taccuino di esempio. Per le specifiche complete di OpenQASM, consulta il sito Web di OpenQASM.

Operazioni e grammatica supportate con OpenPulse

Tipi di OpenPulse dati supportati

Blocchi Cal:

```
cal {
    ...
}
```

Blocchi Defcal:

```
// 1 qubit
defcal x $0 {
...
}

// 1 qubit w. input parameters as constants
defcal my_rx(pi) $0 {
...
}

// 1 qubit w. input parameters as free parameters
defcal my_rz(angle theta) $0 {
...
}

// 2 qubit (above gate args are also valid)
defcal cz $1, $0 {
...
}
```

Cornici:

```
frame my_frame = newframe(port_0, 4.5e9, 0.0);
```

Forme d'onda:

```
// prebuilt
waveform my_waveform_1 = constant(1e-6, 1.0);

//arbitrary
waveform my_waveform_2 = {0.1 + 0.1im, 0.1 + 0.1im, 0.1, 0.1};
```

Esempio di calibrazione Custom Gate:

```
cal {
```

```
    waveform wf1 = constant(1e-6, 0.25);
}

defcal my_x $0 {
   play(wf1, q0_rf_frame);
}

defcal my_cz $1, $0 {
    barrier q0_q1_cz_frame, q0_rf_frame;
    play(q0_q1_cz_frame, wf1);
    delay[300ns] q0_rf_frame
    shift_phase(q0_rf_frame, 4.366186381749424);
    delay[300ns] q0_rf_frame;
    shift_phase(q0_rf_frame.phase, 5.916747563126659);
    barrier q0_q1_cz_frame, q0_rf_frame;
    shift_phase(q0_q1_cz_frame, 2.183093190874712);
}

bit[2] ro;
my_x $0;
my_cz $1,$0;
c[0] = measure $0;
```

Esempio di impulso arbitrario:

```
bit[2] ro;
cal {
    waveform wf1 = {0.1 + 0.1im, 0.1 + 0.1im, 0.1, 0.1};
    barrier q0_drive, q0_q1_cross_resonance;
    play(q0_q1_cross_resonance, wf1);
    delay[300ns] q0_drive;
    shift_phase(q0_drive, 4.366186381749424);
    delay[300dt] q0_drive;
   barrier q0_drive, q0_q1_cross_resonance;
   play(q0_q1_cross_resonance, wf1);
    ro[0] = capture_v0(r0_measure);
    ro[1] = capture_v0(r1_measure);
}
```

Crea e invia un esempio di task quantistico OpenQASM 3.0

Puoi utilizzare Amazon Braket Python SDK, Boto3 o per inviare attività quantistiche OpenQASM 3.0 AWS CLI a un dispositivo Amazon Braket.

In questa sezione:

Un esempio di programma OpenQASM 3.0

Per creare un'attività OpenQASM 3.0, è possibile iniziare con un programma OpenQASM 3.0 di base (ghz.qasm) che prepara uno stato GHZ, come mostrato nell'esempio seguente.

```
// ghz.qasm
// Prepare a GHZ state
OPENQASM 3;

qubit[3] q;
bit[3] c;

h q[0];
cnot q[0], q[1];
cnot q[1], q[2];

c = measure q;
```

Usa Python SDK per creare attività quantistiche OpenQASM 3.0

Puoi utilizzare l'SDK Amazon Braket Python per inviare questo programma a un dispositivo Amazon Braket con il seguente codice. Assicurati di sostituire l'esempio di posizione del bucket Amazon S3 «amzn-s3-demo-bucket» con il tuo nome di bucket Amazon S3.

```
with open("ghz.qasm", "r") as ghz:
    ghz_qasm_string = ghz.read()

# Import the device module
from braket.aws import AwsDevice
# Choose the Rigetti device
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")
from braket.ir.openqasm import Program
```

```
program = Program(source=ghz_qasm_string)
my_task = device.run(program)

# Specify an optional s3 bucket location and number of shots
s3_location = ("amzn-s3-demo-bucket", "openqasm-tasks")
my_task = device.run(
    program,
    s3_location,
    shots=100,
)
```

Usa Boto3 per creare attività quantistiche OpenQASM 3.0

Puoi anche usare AWS Python SDK for Braket (Boto3) per creare i task quantistici usando le stringhe OpenQASM 3.0, come mostrato nell'esempio seguente. Il seguente frammento di codice fa riferimento a ghz.qasm che prepara uno stato GHZ come mostrato sopra.

```
import boto3
import json

my_bucket = "amzn-s3-demo-bucket"
s3_prefix = "openqasm-tasks"

with open("ghz.qasm") as f:
    source = f.read()

action = {
    "braketSchemaHeader": {
        "name": "braket.ir.openqasm.program",
        "version": "1"
    },
    "source": source
}
device_parameters = {}
device_arn = "arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3"
shots = 100

braket_client = boto3.client('braket', region_name='us-west-1')
rsp = braket_client.create_quantum_task(
    action=json.dumps(
        action
    ),
    deviceParameters=json.dumps(
```

```
        device_parameters
    ),
    deviceArn=device_arn,
    shots=shots,
    outputS3Bucket=my_bucket,
    outputS3KeyPrefix=s3_prefix,
)
```

Usa per creare attività OpenQASM 3.0 AWS CLI

La AWS Command Line Interface (CLI) può essere utilizzata anche per inviare programmi OpenQASM 3.0, come mostrato nell'esempio seguente.

```
aws braket create-quantum-task \
    --region "us-west-1" \
    --device-arn "arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3" \
    --shots 100 \
    --output-s3-bucket "amzn-s3-demo-bucket" \
    --output-s3-key-prefix "openqasm-tasks" \
    --action '{
        "braketSchemaHeader": {
            "name": "braket.ir.openqasm.program",
            "version": "1"
        },
        "source": $(cat ghz.qasm)
    }'
```

Support per OpenQASM su diversi dispositivi Braket

Per i dispositivi che supportano OpenQASM 3.0, il `action` campo supporta una nuova azione tramite la `GetDevice` risposta, come mostrato nell'esempio seguente per i dispositivi and. Rigetti IonQ

```
//OpenQASM as available with the Rigetti device capabilities
{
    "braketSchemaHeader": {
        "name": "braket.device_schema.rigetti.rigetti_device_capabilities",
        "version": "1"
    },
    "service": {...},
    "action": {
        "braket.ir.jaqcd.program": {...},
        "braket.ir.openqasm.program": {
```

```
            "actionType": "braket.ir.openqasm.program",
            "version": [
                "1"
            ],
            ….
        }
    }
}

//OpenQASM as available with the IonQ device capabilities
{
    "braketSchemaHeader": {
        "name": "braket.device_schema.ionq.ionq_device_capabilities",
        "version": "1"
    },
    "service": {...},
    "action": {
        "braket.ir.jaqcd.program": {...},
        "braket.ir.openqasm.program": {
            "actionType": "braket.ir.openqasm.program",
            "version": [
                "1"
            ],
            ….
        }
    }
}
```

Per i dispositivi che supportano il controllo a impulsi, il `pulse` campo viene visualizzato nella `GetDevice` risposta. L'esempio seguente mostra questo `pulse` campo per il Rigetti dispositivo.

```
// Rigetti
{
  "pulse": {
    "braketSchemaHeader": {
      "name": "braket.device_schema.pulse.pulse_device_action_properties",
      "version": "1"
    },
    "supportedQhpTemplateWaveforms": {
      "constant": {
        "functionName": "constant",
        "arguments": [
          {
```

```
            "name": "length",
            "type": "float",
            "optional": false
          },
          {
            "name": "iq",
            "type": "complex",
            "optional": false
          }
        ]
      },
      ...
    },
    "ports": {
      "q0_ff": {
        "portId": "q0_ff",
        "direction": "tx",
        "portType": "ff",
        "dt": 1e-9,
        "centerFrequencies": [
          375000000
        ]
      },
      ...
    },
    "supportedFunctions": {
      "shift_phase": {
        "functionName": "shift_phase",
        "arguments": [
          {
            "name": "frame",
            "type": "frame",
            "optional": false
          },
          {
            "name": "phase",
            "type": "float",
            "optional": false
          }
        ]
      },
     ...
    },
    "frames": {
```

```
      "q0_q1_cphase_frame": {
        "frameId": "q0_q1_cphase_frame",
        "portId": "q0_ff",
        "frequency": 462475694.24460185,
        "centerFrequency": 375000000,
        "phase": 0,
        "associatedGate": "cphase",
        "qubitMappings": [
          0,
          1
        ]
      },
      ...
    },
    "supportsLocalPulseElements": false,
    "supportsDynamicFrames": false,
    "supportsNonNativeGatesWithPulses": false,
    "validationParameters": {
      "MAX_SCALE": 4,
      "MAX_AMPLITUDE": 1,
      "PERMITTED_FREQUENCY_DIFFERENCE": 400000000
    }
  }
}
```

I campi precedenti descrivono in dettaglio quanto segue:

Porte:

Descrive le porte predefinite dei dispositivi esterni (`extern`) dichiarate sulla QPU in aggiunta alle proprietà associate della porta specificata. Tutte le porte elencate in questa struttura sono pre-dichiarate come identificatori validi all'interno del `OpenQASM 3.0` programma inviato dall'utente. Le proprietà aggiuntive di una porta includono:

- ID porta (PortID)
  - Il nome della porta dichiarato come identificatore in OpenQASM 3.0.
- Direzione (direzione)
  - La direzione del porto. Le porte di azionamento trasmettono gli impulsi (direzione «tx»), mentre le porte di misurazione ricevono gli impulsi (direzione «rx»).
- Tipo di porta (PortType)
  - Il tipo di azione di cui è responsabile questa porta (ad esempio, drive, capture o ff - fast-flux).

- Dt (dt)
  - Il tempo in secondi che rappresenta una singola fase temporale di campionamento sulla porta specificata.
- Mappature Qubit (QubitMappings)
  - I qubit associati alla porta specificata.
- Frequenze centrali (CenterFrequencies)
  - Un elenco delle frequenze centrali associate per tutti i frame predichiarati o definiti dall'utente sulla porta. Per ulteriori informazioni, fate riferimento a Frames.
- Proprietà specifiche QHP () qhpSpecificProperties
  - Una mappa opzionale che descrive in dettaglio le proprietà esistenti sulla porta specifica del QHP.

Cornici:

Descrive i frame esterni predefiniti dichiarati sulla QPU e le proprietà associate relative ai frame. Tutti i frame elencati in questa struttura sono pre-dichiarati come identificatori validi all'interno del `OpenQASM 3.0` programma inviato dall'utente. Le proprietà aggiuntive di un frame includono:

- ID del frame (frameID)
  - Il nome del frame dichiarato come identificatore in OpenQASM 3.0.
- ID porta (PortID)
  - La porta hardware associata per il frame.
- Frequenza (frequenza)
  - La frequenza iniziale predefinita del frame.
- Frequenza centrale (CenterFrequency)
  - Il centro della larghezza di banda della frequenza del frame. In genere, i frame possono essere regolati solo su una determinata larghezza di banda attorno alla frequenza centrale. Di conseguenza, le regolazioni della frequenza devono rimanere entro un determinato delta della frequenza centrale. È possibile trovare il valore della larghezza di banda nei parametri di convalida.
- Fase (fase)
  - La fase iniziale predefinita del frame.
- Porta associata (AssociatedGate)

  - Le porte associate al frame specificato.
- Mappature Qubit (QubitMappings)
  - I qubit associati al frame specificato.
- Proprietà specifiche QHP () qhpSpecificProperties
  - Una mappa opzionale che descrive in dettaglio le proprietà esistenti relative al frame specifico del QHP.

SupportsDynamicFrames:

Descrive se un frame può essere dichiarato `cal` o `defcal` bloccato tramite la OpenPulse `newframe` funzione. Se questo è falso, all'interno del programma possono essere utilizzati solo i frame elencati nella struttura del frame.

SupportedFunctions:

Descrive le OpenPulse funzioni supportate dal dispositivo oltre agli argomenti, ai tipi di argomento e ai tipi restituiti associati per le funzioni specificate. Per vedere esempi di utilizzo delle OpenPulse funzioni, consultate le OpenPulsespecifiche. Al momento, Braket supporta:

- shift_phase
  - Sposta la fase di un frame in base a un valore specificato
- set_phase
  - Imposta la fase del frame sul valore specificato
- swap_phases
  - Scambia le fasi tra due frame.
- shift_frequency
  - Sposta la frequenza di un fotogramma in base a un valore specificato
- set_frequency
  - Imposta la frequenza del frame sul valore specificato
- giocare
  - Pianifica una forma d'onda
- capture_v0
  - Restituisce il valore di un frame di acquisizione in un registro di bit

SupportedQhpTemplateWaveforms:

Descrive le funzioni di forma d'onda predefinite disponibili sul dispositivo e gli argomenti e i tipi associati. Per impostazione predefinita, Braket Pulse offre routine di forme d'onda predefinite su tutti i dispositivi, che sono:

Costante

$$Constant(t, \tau, iq) = iq$$

`τ`è la lunghezza della forma d'onda ed è un numero complesso. `iq`

```
def constant(length, iq)
```

gaussiana

$$Gaussian(t, \tau, \sigma, A = 1, ZaE = 0) = \frac{A}{1 - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)}\left[\exp\left(-\frac{1}{2}\left(\frac{t-\frac{\tau}{2}}{\sigma}\right)^2\right) - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)\right]$$

`τ`è la lunghezza della forma d'onda, è la larghezza della gaussiana e l'ampiezza. `σ` `A` Se è impostata `ZaE` su`True`, la gaussiana viene sfalsata e ridimensionata in modo che sia uguale a zero all'inizio e alla fine della forma d'onda e raggiunga il massimo. `A`

```
def gaussian(length, sigma, amplitude=1, zero_at_edges=False)
```

TRASCINA (gaussiano)

$$DRAG\_Gaussian(t, \tau, \sigma, \beta, A = 1, ZaE = 0) = \frac{A}{1 - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)}\left(1 - i\beta\frac{t-\frac{\tau}{2}}{\sigma^2}\right)\left[\exp\left(-\frac{1}{2}\left(\frac{t-\frac{\tau}{2}}{\sigma}\right)^2\right) - ZaE * \exp\left(-\frac{1}{2}\left(\frac{\tau}{2\sigma}\right)^2\right)\right]$$

`τ`è la lunghezza della forma d'onda, è la larghezza della gaussiana, σ è un parametro libero ed è l'ampiezza. `β` `A` Se è impostata `ZaE` su`True`, la Derivative Removal by Adiabatic Gate (DRAG) gaussian viene sfalsata e ridimensionata in modo da essere uguale a zero all'inizio e alla fine della forma d'onda e la parte reale raggiunge il massimo. `A` Per ulteriori informazioni sulla forma d'onda DRAG, vedere il paper Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits.

```
def drag_gaussian(length, sigma, beta, amplitude=1, zero_at_edges=False)
```

Piazza Erf

$$\text{Erf\_Square}(t, L, W, \sigma, A = 1, ZaE = 0) =$$

$$A \times \frac{\text{erf}((t - t_1)/\sigma) + \text{erf}(-(t - t_2)/\sigma)}{2 \times \text{erf}(W/2\sigma))}$$

Dov'è la lunghezza, `L` `W` è la larghezza della forma d'onda, σ definisce la velocità con cui gli spigoli si alzano e si abbassano e, è l'ampiezza. t_1`=(L-W)/2` t_2`=(L+W)/2` `A` Se è impostata `ZaE` su`True`, la gaussiana viene sfalsata e ridimensionata in modo che sia uguale a zero all'inizio e alla fine della forma d'onda e raggiunga il massimo. `A` L'equazione seguente è la versione ridimensionata della forma d'onda.

$$\text{Erf\_Square}(..., ZaE = 1) = (a \times \text{Erf\_Square}(..., ZaE = 0) - bA)/(a - b)$$

`b=erf(-`t_1`/σ)/2+erf(t` $_2$`/σ)/2` Dove e. `a=erf(W/2σ)`

```
def erf_square(length, width, sigma, amplitude=1, zero_at_edges=False)
```

SupportsLocalPulseElements:

Descrive se gli elementi a impulsi, come porte, frame e forme d'onda, possono essere definiti localmente in `defcal` blocchi. Se il valore è`false`, gli elementi devono essere definiti in `cal` blocchi.

SupportsNonNativeGatesWithPulses:

Descrive se è possibile o meno utilizzare porte non native in combinazione con programmi a impulsi. Ad esempio, non è possibile utilizzare una porta non nativa come una `H` porta in un programma senza prima definire la porta passante `defcal` per il qubit utilizzato. Puoi trovare l'elenco delle `nativeGateSet` chiavi dei gate nativi nella sezione delle funzionalità del dispositivo.

ValidationParameters:

Descrive i limiti di convalida degli elementi Pulse, tra cui:

- Valori di scala massima/ampiezza massima per le forme d'onda (arbitrari e predefiniti)
- Larghezza di banda di frequenza massima rispetto alla frequenza centrale fornita in Hz

- Impulso minimo length/duration in secondi
- Impulso massimo length/duration in secondi

Operazioni, risultati e tipi di risultati supportati con OpenQASM

Per scoprire quali funzionalità di OpenQASM 3.0 sono supportate da ciascun dispositivo, è possibile fare riferimento alla `braket.ir.openqasm.program` chiave nel `action` campo sull'output delle funzionalità del dispositivo. Ad esempio, le seguenti sono le operazioni supportate e i tipi di risultati disponibili per il simulatore Braket State Vector. SV1

```
...
  "action": {
    "braket.ir.jaqcd.program": {
      ...
    },
 "braket.ir.openqasm.program": {
      "version": [
        "1.0"
      ],
      "actionType": "braket.ir.openqasm.program",
      "supportedOperations": [
        "ccnot",
        "cnot",
        "cphaseshift",
        "cphaseshift00",
        "cphaseshift01",
        "cphaseshift10",
        "cswap",
        "cy",
        "cz",
        "h",
        "i",
        "iswap",
        "pswap",
        "phaseshift",
        "rx",
        "ry",
        "rz",
        "s",
        "si",
        "swap",
        "t",
```

```
        "ti",
        "v",
        "vi",
        "x",
        "xx",
        "xy",
        "y",
        "yy",
        "z",
        "zz"
      ],
      "supportedPragmas": [
        "braket_unitary_matrix"
      ],
      "forbiddenPragmas": [],
      "maximumQubitArrays": 1,
      "maximumClassicalArrays": 1,
      "forbiddenArrayOperations": [
        "concatenation",
        "negativeIndex",
        "range",
        "rangeWithStep",
        "slicing",
        "selection"
      ],
      "requiresAllQubitsMeasurement": true,
      "supportsPhysicalQubits": false,
      "requiresContiguousQubitIndices": true,
      "disabledQubitRewiringSupported": false,
      "supportedResultTypes": [
        {
          "name": "Sample",
          "observables": [
            "x",
            "y",
            "z",
            "h",
            "i",
            "hermitian"
          ],
          "minShots": 1,
          "maxShots": 100000
        },
        {
```

```
          "name": "Expectation",
          "observables": [
            "x",
            "y",
            "z",
            "h",
            "i",
            "hermitian"
          ],
          "minShots": 0,
          "maxShots": 100000
        },
        {
          "name": "Variance",
          "observables": [
            "x",
            "y",
            "z",
            "h",
            "i",
            "hermitian"
          ],
          "minShots": 0,
          "maxShots": 100000
        },
        {
          "name": "Probability",
          "minShots": 1,
          "maxShots": 100000
        },
        {
          "name": "Amplitude",
          "minShots": 0,
          "maxShots": 0
        }
        {
          "name": "AdjointGradient",
          "minShots": 0,
          "maxShots": 0
        }
      ]
    }
  },
```

```
...
```

Simula il rumore con OpenQASM 3.0

Per simulare il rumore con OpenQASM3, si utilizzano le istruzioni pragma per aggiungere operatori di rumore. Ad esempio, per simulare la versione rumorosa del programma GHZ fornita in precedenza, è possibile inviare il seguente programma OpenQASM.

```
// ghz.qasm
// Prepare a GHZ state
OPENQASM 3;

qubit[3] q;
bit[3] c;

h q[0];
#pragma braket noise depolarizing(0.75) q[0] cnot q[0], q[1];
#pragma braket noise depolarizing(0.75) q[0]
#pragma braket noise depolarizing(0.75) q[1] cnot q[1], q[2];
#pragma braket noise depolarizing(0.75) q[0]
#pragma braket noise depolarizing(0.75) q[1]

c = measure q;
```

Le specifiche per tutti gli operatori di rumore pragma supportati sono fornite nell'elenco seguente.

```
#pragma braket noise bit_flip(<float in [0,1/2]>) <qubit>
#pragma braket noise phase_flip(<float in [0,1/2]>) <qubit>
#pragma braket noise pauli_channel(<float>, <float>, <float>)  <qubit>
#pragma braket noise depolarizing(<float in [0,3/4]>) <qubit>
#pragma braket noise two_qubit_depolarizing(<float in [0,15/16]>) <qubit>, <qubit>
#pragma braket noise two_qubit_dephasing(<float in [0,3/4]>) <qubit>, <qubit>
#pragma braket noise amplitude_damping(<float in [0,1]>) <qubit>
#pragma braket noise generalized_amplitude_damping(<float in [0,1]> <float in [0,1]>)
 <qubit>
#pragma braket noise phase_damping(<float in [0,1]>) <qubit>
#pragma braket noise kraus([[<complex m0_00>, ], ...], [[<complex m1_00>, ], ...], ...)
 <qubit>[, <qubit>]     // maximum of 2 qubits and maximum of 4 matrices for 1 qubit,
 16 for 2
```

Operatore Kraus

Per generare un operatore Kraus, puoi scorrere un elenco di matrici, stampando ogni elemento della matrice come espressione complessa.

Quando usi gli operatori Kraus, ricorda quanto segue:

- Il numero di non qubits deve superare 2. La definizione corrente negli schemi stabilisce questo limite.
- La lunghezza dell'elenco degli argomenti deve essere un multiplo di 8. Ciò significa che deve essere composto solo da matrici 2x2.
- La lunghezza totale non supera 2 matrici 2*num_qubits. Ciò significa 4 matrici per 1 e 16 per 2. qubit qubits
- Tutte le matrici fornite sono CPTP (Completely Positive Trace Preserving).
- Il prodotto degli operatori Kraus con i loro coniugati di trasposizione deve sommarsi a una matrice di identità.

Qubitricablaggio con OpenQASM 3.0

Amazon Braket supporta la qubit notazione fisica all'interno di OpenQASM sui Rigetti dispositivi (per ulteriori informazioni consulta questa pagina). Quando utilizzi sistemi fisici qubits con la strategia di ricablaggio ingenua, assicurati che siano collegati al qubits dispositivo selezionato. In alternativa, se invece vengono utilizzati qubit i registri, la strategia di ricablaggio PARTIAL è abilitata di default sui dispositivi. Rigetti

```
// ghz.qasm
// Prepare a GHZ state
OPENQASM 3;

h $0;
cnot $0, $1;
cnot $1, $2;

measure $0;
measure $1;
measure $2;
```

Compilazione Verbatim con OpenQASM 3.0

Quando si esegue un circuito quantistico su computer quantistici forniti da fornitori comeRigetti, eIonQ, è possibile indicare al compilatore di eseguire i circuiti esattamente come definito, senza alcuna modifica. Questa funzionalità è nota come compilazione letterale. Con i dispositivi Rigetti, è possibile specificare con precisione cosa viene preservato: un intero circuito o solo parti specifiche di esso. Per preservare solo parti specifiche di un circuito, è necessario utilizzare porte native all'interno delle regioni protette. Attualmente, supporta IonQ solo la compilazione letterale per l'intero circuito, quindi ogni istruzione del circuito deve essere racchiusa in una casella letterale.

Con OpenQASM, è possibile specificare esplicitamente un pragma letterale attorno a una casella di codice che viene poi lasciata intatta e non ottimizzata dalla routine di compilazione di basso livello dell'hardware. Il seguente esempio di codice mostra come utilizzare la direttiva per raggiungere questo obiettivo. `#pragma braket verbatim`

```
OPENQASM 3;

bit[2] c;

#pragma braket verbatim
box{
    rx(0.314159) $0;
    rz(0.628318) $0, $1;
    cz $0, $1;
}

c[0] = measure $0;
c[1] = measure $1;
```

Per informazioni più dettagliate sul processo di compilazione letterale, inclusi esempi e best practice, consultate il taccuino di esempio della compilazione Verbatim disponibile nel repository github. amazon-braket-examples

La console Braket

Le attività OpenQASM 3.0 sono disponibili e possono essere gestite all'interno della console Amazon Braket. Sulla console, hai la stessa esperienza nell'invio di attività quantistiche in OpenQASM 3.0 come avevi nell'invio di attività quantistiche esistenti.

Risorse aggiuntive

OpenQASM è disponibile in tutte le regioni Amazon Braket.

Per un notebook di esempio per iniziare a usare OpenQASM su Amazon Braket, consulta Braket Tutorials. GitHub

Calcolo dei gradienti con OpenQASM 3.0

Amazon Braket supporta il calcolo dei gradienti su simulatori on-demand e locali durante l'esecuzione in modalità (esatta). `shots=0` Ciò si ottiene mediante l'uso del metodo di differenziazione aggiuntiva. Per specificare il gradiente da calcolare, è possibile fornire il pragma appropriato, come illustrato nel codice riportato nell'esempio seguente.

```
OPENQASM 3.0;
input float alpha;

bit[2] b;
qubit[2] q;

h q[0];
h q[1];
rx(alpha) q[0];
rx(alpha) q[1];
b[0] = measure q[0];
b[1] = measure q[1];

#pragma braket result adjoint_gradient h(q[0]) @ i(q[1]) alpha
```

Invece di elencare tutti i singoli parametri in modo esplicito, potete anche specificare la parola chiave all'interno del pragma. `all` Questo calcolerà il gradiente rispetto a tutti i `input` parametri elencati, un'opzione utile quando il numero di parametri è molto elevato. In questo caso, il pragma sarà simile al codice nell'esempio seguente.

```
#pragma braket result adjoint_gradient h(q[0]) @ i(q[1]) all
```

Tutti i tipi di osservabili sono supportati nell'implementazione OpenQASM 3.0 di Amazon Braket, inclusi i singoli operatori, i prodotti tensoriali, gli osservabili hermitiani e gli osservabili. `Sum` L'operatore specifico da utilizzare per il calcolo dei gradienti deve essere incluso nella

`expectation()` funzione e i qubit su cui agisce ogni termine dell'osservabile devono essere specificati in modo esplicito.

Misurazione di qubit specifici con OpenQASM 3.0

Il simulatore vettoriale statale locale e il simulatore di matrice di densità locale forniti da Amazon Braket supportano l'invio di OpenQASM programmi in cui è possibile misurare selettivamente un sottoinsieme dei qubit del circuito. Questa funzionalità, spesso denominata misurazione parziale, consente calcoli quantistici più mirati ed efficienti. Ad esempio, nel seguente frammento di codice, è possibile creare un circuito a due qubit e scegliere di misurare solo il primo qubit, lasciando il secondo qubit non misurato.

```
partial_measure_qasm = """
OPENQASM 3.0;
bit[1] b;
qubit[2] q;
h q[0];
cnot q[0], q[1];
b[0] = measure q[0];
"""
```

In questo esempio, abbiamo un circuito quantistico con due qubit `q[0]` eq`[1]`, ma siamo interessati solo a misurare lo stato del primo qubit. Ciò è ottenuto dalla lineab`[0] = measure q[0]`, che misura lo stato del qubit [0] e memorizza il risultato nel classico bit b [0]. Per eseguire questo scenario di misurazione parziale, possiamo eseguire il codice seguente sul simulatore vettoriale dello stato locale fornito da Amazon Braket.

```
from braket.devices import LocalSimulator

local_sim = LocalSimulator()
partial_measure_local_sim_task =
 local_sim.run(OpenQASMProgram(source=partial_measure_qasm), shots = 10)
partial_measure_local_sim_result = partial_measure_local_sim_task.result()
print(partial_measure_local_sim_result.measurement_counts)
print("Measured qubits: ", partial_measure_local_sim_result.measured_qubits)
```

Puoi verificare se un dispositivo supporta la misurazione parziale ispezionando il `requiresAllQubitsMeasurement` campo nelle sue proprietà di azione; in caso affermativo`False`, è supportata la misurazione parziale.

```
from braket.devices import Devices

AwsDevice(Devices.Rigetti.Ankaa3).properties.action['braket.ir.openqasm.program'].requiresAllQu
```

Qui `requiresAllQubitsMeasurement` è`False`, il che indica che non tutti i qubit devono essere misurati.

Esplora le funzionalità sperimentali

Le funzionalità sperimentali forniscono l'accesso all'hardware con disponibilità limitata e a nuove funzionalità software emergenti. Queste funzionalità possono influire sulle prestazioni del dispositivo oltre le specifiche standard. Puoi abilitare automaticamente le funzionalità software sperimentali per ogni attività tramite l'SDK Amazon Braket.

Per utilizzare funzionalità sperimentali, specifica il `experimental_capabilities` parametro quando crei attività quantistiche. Imposta questo parametro per `"ALL"` abilitare tutte le funzionalità sperimentali disponibili per quell'attività. L'esempio seguente mostra come abilitare le funzionalità sperimentali quando si esegue un circuito su un dispositivo:

```
from braket.aws import AwsDevice

device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/quera/Aquila")

task = device.run(
   circuit,
   shots=1000,
   experimental_capabilities="ALL"
)
```

> **Note**
>
> Queste funzionalità sono sperimentali e possono cambiare senza preavviso. Le prestazioni del dispositivo possono differire dalle specifiche pubblicate e i risultati possono variare rispetto alle operazioni standard. È necessario abilitare in modo esplicito le funzionalità sperimentali per ogni attività. Le attività senza questo parametro utilizzeranno solo le funzionalità standard del dispositivo.

In questa sezione:

Accesso alla desintonizzazione locale sull'Aquila QuEra

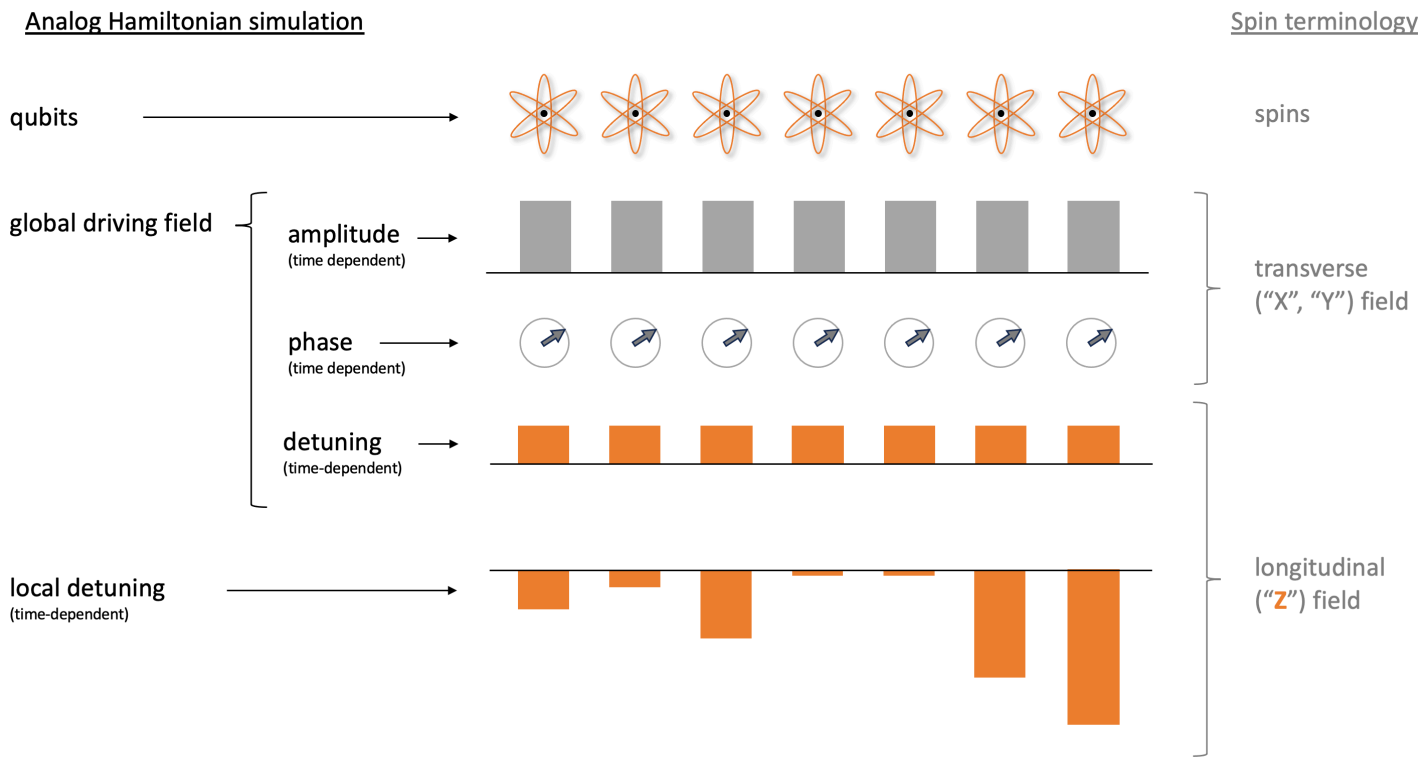
La detuning locale (LD) è un nuovo campo di controllo dipendente dal tempo con uno schema spaziale personalizzabile. Il campo LD influenza i qubit secondo uno schema spaziale personalizzabile, realizzando diverse hamiltoniane per diversi qubit oltre a ciò che il campo di pilotaggio uniforme e l'interazione Rydberg-Rydberg possono creare.

Vincoli:

Il modello spaziale del campo di detuning locale è personalizzabile per ogni programma AHS, ma è costante nel corso di un programma. La serie temporale del campo di detuning locale deve iniziare e terminare da zero con tutti i valori inferiori o uguali a zero. Inoltre, i parametri del campo di detoning locale sono limitati da vincoli numerici, che possono essere visualizzati tramite Braket SDK nella sezione delle proprietà del dispositivo specifico -. `aquila_device.properties.paradigm.rydberg.rydbergLocal`

Limitazioni:

Quando si eseguono programmi quantistici che utilizzano il campo di detonizzazione locale (anche se la sua magnitudine è impostata su zero costante in hamiltoniano), il dispositivo sperimenta una decoerenza più rapida rispetto al tempo T2 elencato nella sezione delle prestazioni delle proprietà di Aquila. Quando non necessario, è consigliabile omettere il campo di detoning locale dall'hamiltoniano del programma AHS.



Esempi:

1. Simulazione dell'effetto di un campo magnetico longitudinale non uniforme nei sistemi di spin

   Sebbene l'ampiezza e la fase del campo pilotante abbiano sui qubit lo stesso effetto del campo magnetico trasverso sugli spin, la somma della detonizzazione del campo pilotante e della detonizzazione locale produce sui qubit lo stesso effetto del campo longitudinale sugli spin. Grazie al controllo spaziale del campo di detunizzazione locale, è possibile simulare sistemi di spin più complessi.
2. Preparazione degli stati iniziali di non equilibrio

   Il quaderno di esempio Simulating retice gauge theory with Rydberg atoms mostra come impedire l'eccitazione dell'atomo centrale di una disposizione lineare a 9 atomi durante la ricottura del sistema verso la fase ordinata Z2. Dopo la fase di preparazione, il campo di detonizzazione locale viene ridotto e il programma AHS continua a simulare l'evoluzione temporale del sistema a partire da questo particolare stato di non equilibrio.
3. Risoluzione di problemi di ottimizzazione ponderata

   L'esempio di notebook Maximum weight independent set (MWIS) mostra come risolvere un problema MWIS su Aquila. Il campo di detoning locale viene utilizzato per definire i pesi sui nodi

del grafico a disco unitario, i cui bordi sono realizzati dall'effetto di blocco Rybderg. Partendo dallo stato fondamentale uniforme e aumentando gradualmente il campo di detonizzazione locale, il sistema passa allo stato fondamentale del MWIS Hamiltonian per trovare soluzioni al problema.

QuEra Accesso alle geometrie alte dell'Aquila

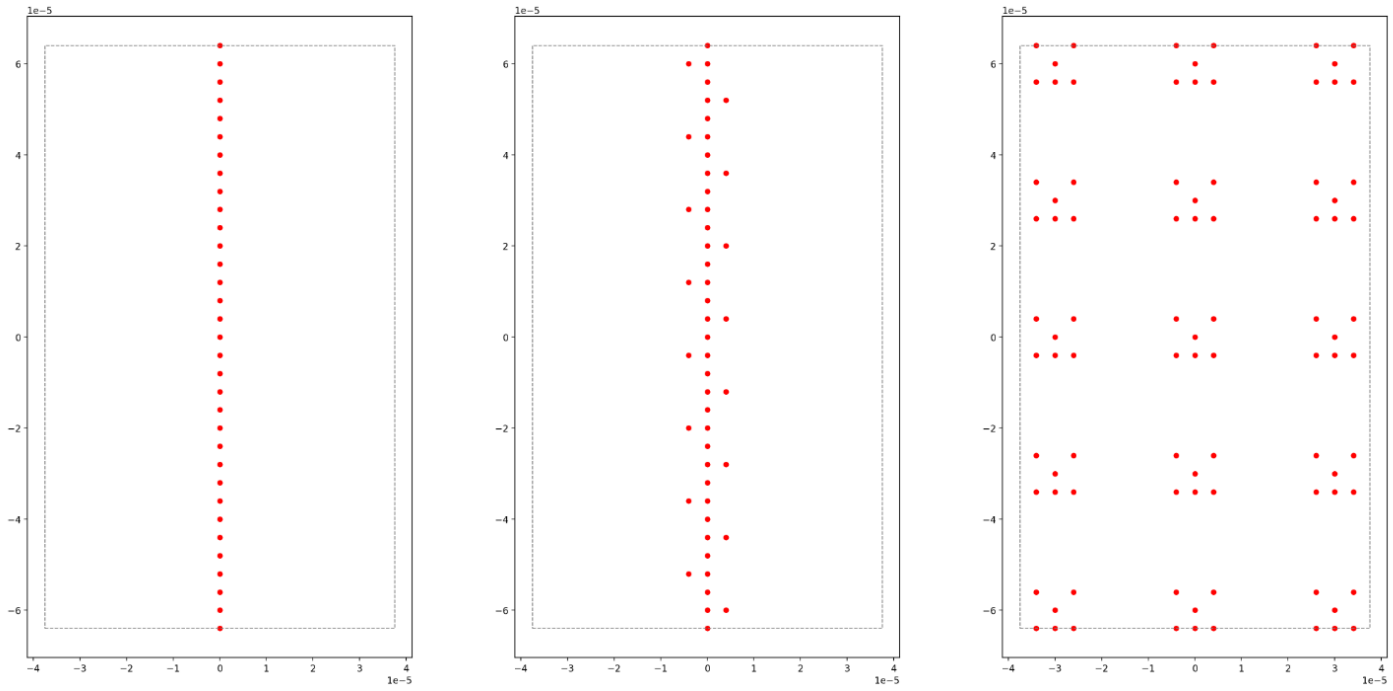
La funzione Tall Geometries consente di specificare geometrie con altezza maggiore. Grazie a questa funzionalità, le disposizioni atomiche dei programmi AHS possono estendersi per una lunghezza aggiuntiva nella direzione y oltre alle normali capacità di Aquila.

Vincoli:

L'altezza massima per geometrie alte è 0,000128 m (128 um).

Limitazioni:

Quando questa funzionalità sperimentale è abilitata per il tuo account, le funzionalità mostrate nella pagina delle proprietà del dispositivo e nella `GetDevice` chiamata continueranno a riflettere il normale limite inferiore di altezza. Quando un programma AHS utilizza arrangiamenti atomici che vanno oltre le normali capacità, si prevede che l'errore di riempimento aumenti. Troverai un numero elevato di 0 imprevisti nella `pre_sequence` parte del risultato dell'attività, che a loro volta riducono la possibilità di ottenere una disposizione perfettamente inizializzata. Questo effetto è più forte nelle file con molti atomi.



Esempi:

1. Arrangiamenti 1D e quasi-1d più grandi

   Le catene atomiche e le disposizioni simili a scale possono essere estese a numeri atomici più elevati. Orientando la direzione lunga parallelamente a y, è possibile programmare istanze più lunghe di questi modelli.

2. Più spazio per la multiplazione dell'esecuzione di attività con geometrie ridotte

   Il taccuino di esempio Parallel quantum tasks on Aquila mostra come sfruttare al meglio l'area disponibile: posizionando copie multiplex della geometria in questione in una disposizione atomica. Con una maggiore area disponibile, è possibile inserire più copie.

Accesso a geometrie strette sull'Aquila QuEra

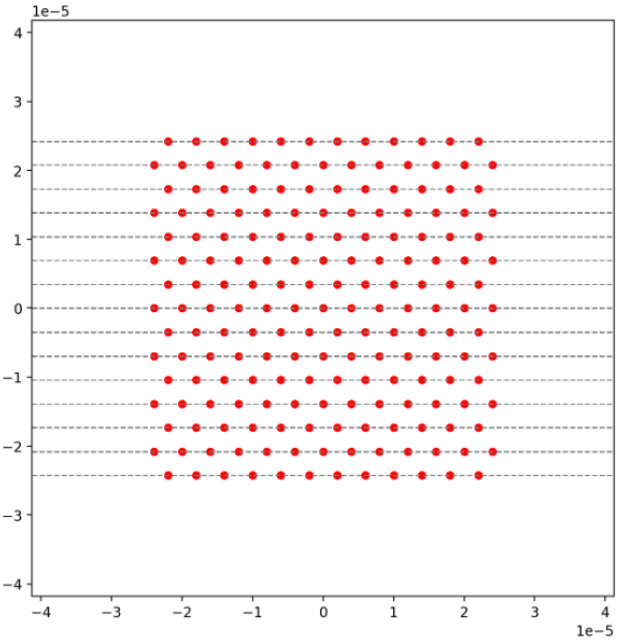
La funzione di geometrie strette consente di specificare geometrie con una spaziatura più breve tra le righe adiacenti. In un programma AHS, gli atomi sono disposti in file, separati da una spaziatura verticale minima. La coordinata y di due siti atomici qualsiasi deve essere zero (stessa riga) o differire di più della distanza minima tra le righe (riga diversa). Grazie alla funzionalità di geometrie strette, la distanza minima tra le file viene ridotta, consentendo la creazione di disposizioni atomiche più strette. Sebbene questa estensione non modifichi il requisito minimo di distanza euclidea tra gli atomi, consente la creazione di reticoli in cui atomi distanti occupano file adiacenti più vicine l'una all'altra, un esempio notevole è il reticolo triangolare.

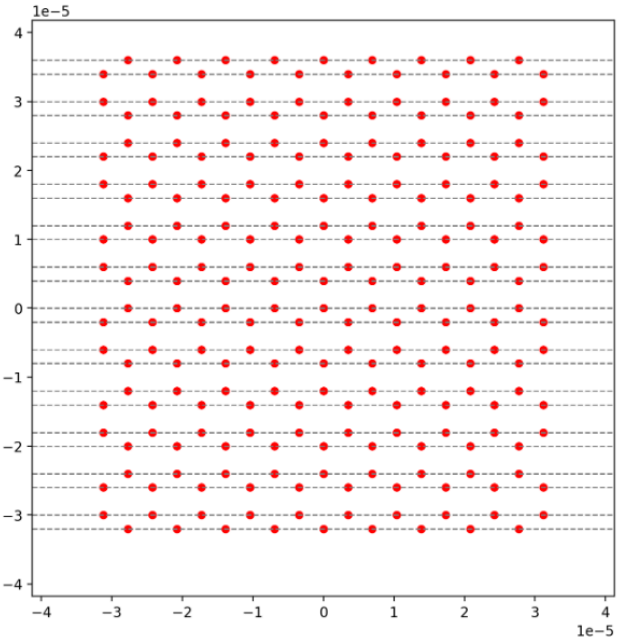
Vincoli:

La distanza minima tra le file per geometrie strette è 0,000002 m (2 um).

Limitazioni:

Quando questa funzionalità sperimentale è abilitata per il tuo account, le funzionalità mostrate nella pagina delle proprietà del dispositivo e nella `GetDevice` chiamata continueranno a riflettere il normale limite inferiore di altezza. Quando un programma AHS utilizza arrangiamenti atomici che vanno oltre le normali capacità, si prevede che l'errore di riempimento aumenti. I clienti riscontreranno un numero elevato di 0 imprevisti nella `pre_sequence` parte del risultato dell'attività, che a loro volta ridurranno la possibilità di ottenere una disposizione perfettamente inizializzata. Questo effetto è più forte nelle file con molti atomi.





Esempi:

1. Reticoli non rettangolari con piccole costanti reticolari

   Una distanza più stretta tra le file consente la creazione di reticoli in cui il vicino più prossimo di alcuni atomi si trova nella direzione diagonale. Esempi degni di nota sono i reticoli triangolari, esagonali e Kagome e alcuni quasi-cristalli.

2. Famiglia di reticoli sintonizzabili

   Nei programmi AHS, le interazioni vengono regolate regolando la distanza tra coppie di atomi. Una spaziatura più stretta tra le righe consente di regolare le interazioni di diverse coppie di atomi l'una rispetto all'altra con maggiore libertà, poiché gli angoli e le distanze che definiscono la struttura dell'atomo sono meno limitati dal vincolo minimo di spaziatura tra le file. Un esempio notevole è la famiglia di reticoli Shastry-Sutherland con lunghezze di legame diverse.

Circuiti dinamici su dispositivi IQM

I circuiti dinamici sui IQM dispositivi consentono misurazioni a medio circuito (MCM) e operazioni di feed-forward. Queste funzionalità consentono ai ricercatori e agli sviluppatori quantistici di implementare algoritmi quantistici avanzati con logica condizionale e capacità di riutilizzo dei qubit.

Questa funzionalità sperimentale aiuta a esplorare algoritmi quantistici con una migliore efficienza delle risorse e a studiare gli schemi di mitigazione e correzione degli errori quantistici.

Istruzioni chiave:

- `measure_ff`: Implementa la misurazione per il controllo del feed-forward, misura un qubit e memorizza il risultato con una chiave di feedback.
- `cc_prx`: Implementa una rotazione controllata in modo classico che si applica solo quando il risultato associato alla chiave di feedback fornita misura uno stato |1#.

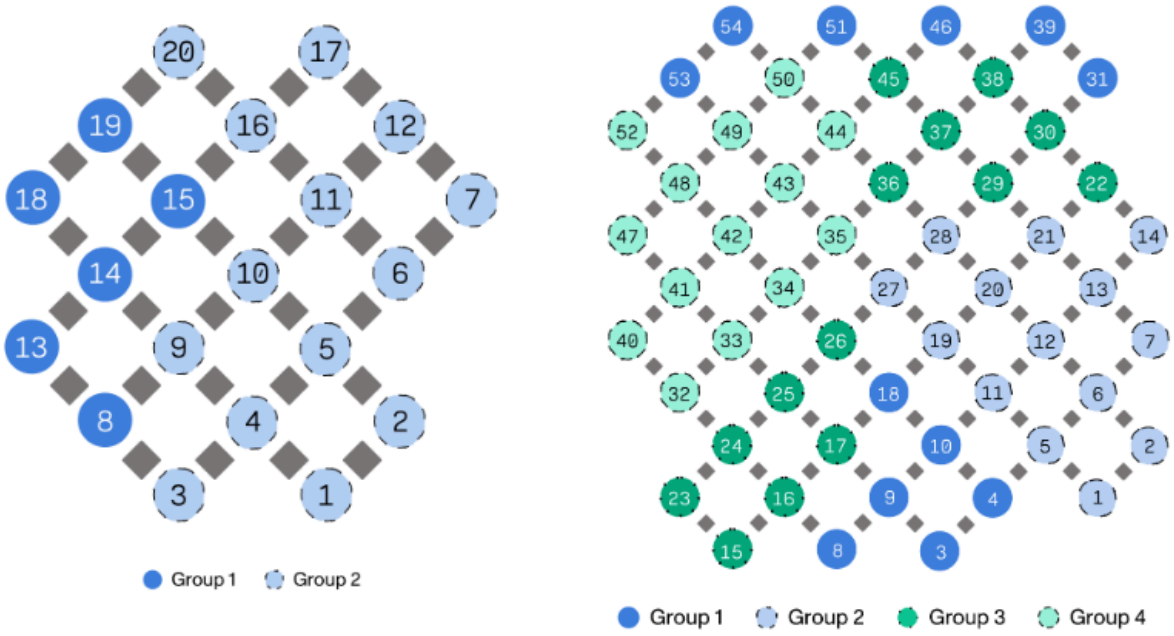
Amazon Braket supporta circuiti dinamici attraversoOpenQASM, ilAmazon Braket SDK, e il. Amazon Braket Qiskit Provider

Vincoli:

1. Le chiavi di feedback nelle `measure_ff` istruzioni devono essere uniche.
2. A `cc_prx` deve succedere dopo `measure_ff` con la stessa chiave di feedback.
3. In un singolo circuito, il feed-forward di un qubit può essere controllato solo da un qubit, da solo o da un altro qubit. In circuiti diversi, è possibile avere diverse coppie di controllo.
   a. Ad esempio, se il qubit 1 è controllato dal qubit 2, non può essere controllato dal qubit 3 nello stesso circuito. Non esiste alcun vincolo sul numero di volte in cui il controllo viene applicato tra qubit 1 e qubit 2. Qubit 2 può essere controllato da qubit 3 (o qubit 1), a meno che non sia stato eseguito un reset attivo sul qubit 2.
4. Il controllo può essere applicato solo ai qubit all'interno dello stesso gruppo. I gruppi di qubit per i Emerald dispositivi IQM Garnet e sono nelle immagini seguenti.
5. I programmi con queste funzionalità devono essere inviati come programmi letterali. Per ulteriori informazioni sui programmi letterali, vedere la compilazione Verbatim con OpenQASM 3.0.

Limitazioni:

MCM può essere utilizzato solo per il controllo feed-forward in un programma. I risultati MCM (0 o 1) non vengono restituiti come parte dei risultati di un'attività.



Queste immagini mostrano IQM i raggruppamenti di qubit per entrambi i dispositivi. Il dispositivo da Garnet 20 qubit contiene 2 gruppi di qubit, mentre il dispositivo da Emerald 54 qubit contiene 4 gruppi di qubit.

Esempi:

1. Riutilizzo dei qubit tramite reset attivo

   MCM con operazioni di ripristino condizionale consente il riutilizzo dei qubit all'interno di un'esecuzione a circuito singolo. Ciò riduce i requisiti di profondità del circuito e migliora l'utilizzo delle risorse dei dispositivi quantistici.

2. Protezione attiva dal bit flip

   I circuiti dinamici rilevano gli errori di bit flip e applicano operazioni correttive in base ai risultati delle misurazioni. Questa implementazione funge da esperimento di rilevamento degli errori quantistici.

3. Esperimenti di teletrasporto

   Il teletrasporto statale trasferisce gli stati dei qubit utilizzando operazioni quantistiche locali e informazioni classiche da. MCMs Il teletrasporto implementa i gate tra i qubit senza operazioni quantistiche dirette. Questi esperimenti dimostrano le subroutine fondamentali in tre aree chiave:

correzione degli errori quantistici, calcolo quantistico basato sulla misurazione e comunicazione quantistica.

4. Simulazione di sistemi quantistici aperti

   I circuiti dinamici modellano il rumore nei sistemi quantistici attraverso il qubit di dati e l'entanglement ambientale e misurazioni ambientali. Questo approccio utilizza qubit specifici per rappresentare dati ed elementi ambientali. Un canale di rumore può essere progettato in base alle porte e alle misurazioni applicate all'ambiente.

Per ulteriori informazioni sull'uso dei circuiti dinamici, consulta altri esempi nell'archivio per notebook Amazon Braket.

Controllo a impulsi su Amazon Braket

Gli impulsi sono i segnali analogici che controllano i qubit in un computer quantistico. Con alcuni dispositivi su Amazon Braket, puoi accedere alla funzione di controllo degli impulsi per inviare circuiti utilizzando impulsi. Puoi accedere al controllo degli impulsi tramite Braket SDK, utilizzando OpenQASM 3.0 o direttamente tramite Braket. APIs Innanzitutto, introduci alcuni concetti chiave per il controllo degli impulsi in Braket.

In questa sezione:

- Frames (Fotogrammi)
- Porte
- Forme d'onda
- Lavorare con Hello Pulse
- Accesso ai gate nativi tramite impulsi

Frames (Fotogrammi)

Un frame è un'astrazione software che funge sia da orologio all'interno del programma quantistico che da fase. L'ora dell'orologio viene incrementata a ogni utilizzo e viene generato un segnale portante statico definito da una frequenza. Quando si trasmettono segnali al qubit, un frame determina la frequenza portante del qubit, l'offset di fase e l'ora in cui viene emesso l'inviluppo della forma d'onda. In Braket Pulse, la costruzione dei frame dipende dal dispositivo, dalla frequenza e dalla fase. A seconda del dispositivo, potete scegliere un frame predefinito o creare un'istanza di nuovi frame fornendo una porta.

```
from braket.aws import AwsDevice
from braket.pulse import Frame, Port

# Predefined frame from a device
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")
drive_frame = device.frames["Transmon_5_charge_tx"]

# Create a custom frame
readout_frame = Frame(frame_id="r0_measure", port=Port("channel_0", dt=1e-9),
 frequency=5e9, phase=0)
```

Porte

Una porta è un'astrazione software che rappresenta qualsiasi input/output componente hardware che controlla i qubit. Aiuta i fornitori di hardware a fornire un'interfaccia con cui gli utenti possono interagire per manipolare e osservare i qubit. Le porte sono caratterizzate da una singola stringa che rappresenta il nome del connettore. Questa stringa mostra anche un incremento di tempo minimo che specifica con quanta precisione possiamo definire le forme d'onda.

```
from braket.pulse import Port

Port0 = Port("channel_0", dt=1e-9)
```

Forme d'onda

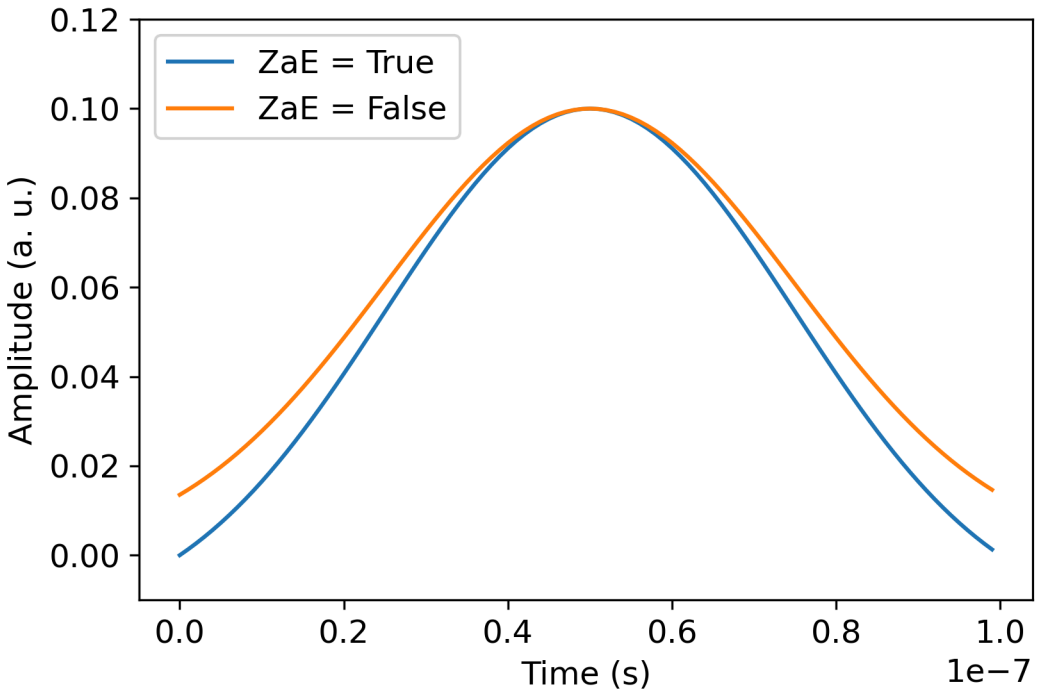
Una forma d'onda è un involucro dipendente dal tempo che possiamo usare per emettere segnali su una porta di uscita o acquisire segnali attraverso una porta di ingresso. È possibile specificare le forme d'onda direttamente tramite un elenco di numeri complessi o utilizzando un modello di forma d'onda per generare un elenco fornito dal fornitore di hardware.

```
from braket.pulse import ArbitraryWaveform, ConstantWaveform
import numpy as np

cst_wfm = ConstantWaveform(length=1e-7, iq=0.1)
arb_wf = ArbitraryWaveform(amplitudes=np.linspace(0, 100))
```

Braket Pulse fornisce una libreria standard di forme d'onda, tra cui una forma d'onda costante, una forma d'onda gaussiana e una forma d'onda Derivative Removal by Adiabatic Gate (DRAG). È

possibile recuperare i dati della forma d'onda tramite la funzione per disegnare la forma della forma d'onda, come mostrato nell'esempio seguente. `sample`

```
from braket.pulse import GaussianWaveform
import numpy as np
import matplotlib.pyplot as plt

zero_at_edge1 = GaussianWaveform(1e-7, 25e-9, 0.1, True)
# or zero_at_edge1 = GaussianWaveform(1e-7, 25e-9, 0.1)
zero_at_edge2 = GaussianWaveform(1e-7, 25e-9, 0.1, False)

times_1 = np.arange(0, zero_at_edge1.length, drive_frame.port.dt)
times_2 = np.arange(0, zero_at_edge2.length, drive_frame.port.dt)

plt.plot(times_1, zero_at_edge1.sample(drive_frame.port.dt))
plt.plot(times_2, zero_at_edge2.sample(drive_frame.port.dt))
```



L'immagine precedente mostra le forme d'onda gaussiane create da. `GaussianWaveform` Abbiamo scelto una lunghezza dell'impulso di 100 ns, una larghezza di 25 ns e un'ampiezza di 0,1 (unità arbitrarie). Le forme d'onda sono centrate nella finestra degli impulsi. `GaussianWaveform`accetta un argomento booleano `zero_at_edges` (zAe nella legenda). Se impostato su`True`, questo argomento

compensa la forma d'onda gaussiana in modo tale che i punti in t=0 e t= `length` siano a zero e ne ridimensiona l'ampiezza in modo che il valore massimo corrisponda all'argomento. `amplitude`

Lavorare con Hello Pulse

In questa sezione, imparerai come caratterizzare e costruire una singola porta qubit direttamente usando pulse su un dispositivo. Rigetti L'applicazione di un campo elettromagnetico a un qubit porta all'oscillazione Rabi, che consente di commutare i qubit tra lo stato 0 e lo stato 1. Con la lunghezza e la fase dell'impulso calibrate, l'oscillazione Rabi può calcolare una singola porta qubit. Qui determineremo la lunghezza ottimale dell'impulso per misurare un impulso pi/2, un blocco elementare utilizzato per costruire sequenze di impulsi più complesse.

Innanzitutto, per creare una sequenza di impulsi, importate la classe. `PulseSequence`

```
from braket.aws import AwsDevice
from braket.circuits import FreeParameter
from braket.devices import Devices
from braket.pulse import PulseSequence, GaussianWaveform

import numpy as np
```

Successivamente, crea un'istanza di un nuovo dispositivo Braket utilizzando l' (Amazon Resource NameARN) della QPU. Il seguente blocco di codice utilizza. Rigetti Ankaa-3

```
device = AwsDevice(Devices.Rigetti.Ankaa3)
```

La seguente sequenza di impulsi include due componenti: riproduzione di una forma d'onda e misurazione di un qubit. La sequenza di impulsi può in genere essere applicata ai fotogrammi. Con alcune eccezioni, come la barriera e il ritardo, che possono essere applicati ai qubit. Prima di costruire la sequenza di impulsi è necessario recuperare i frame disponibili. Il frame di azionamento viene utilizzato per applicare l'impulso per l'oscillazione Rabi e il frame di lettura serve per misurare lo stato del qubit. Questo esempio utilizza i frame del qubit 25.

```
drive_frame = device.frames["Transmon_25_charge_tx"]
readout_frame = device.frames["Transmon_25_readout_rx"]
```

Ora, crea la forma d'onda che verrà riprodotta nel frame del drive. L'obiettivo è caratterizzare il comportamento dei qubit per diverse lunghezze di impulso. Suonerai ogni volta una forma d'onda con lunghezze diverse. Invece di istanziare ogni volta una nuova forma d'onda, usa il supporto Braket

nella sequenza di impulsi. `FreeParameter` È possibile creare la forma d'onda e la sequenza di impulsi una volta con parametri liberi, quindi eseguire la stessa sequenza di impulsi con valori di input diversi.

```
waveform = GaussianWaveform(FreeParameter("length"), FreeParameter("length") * 0.25,
 0.2, False)
```

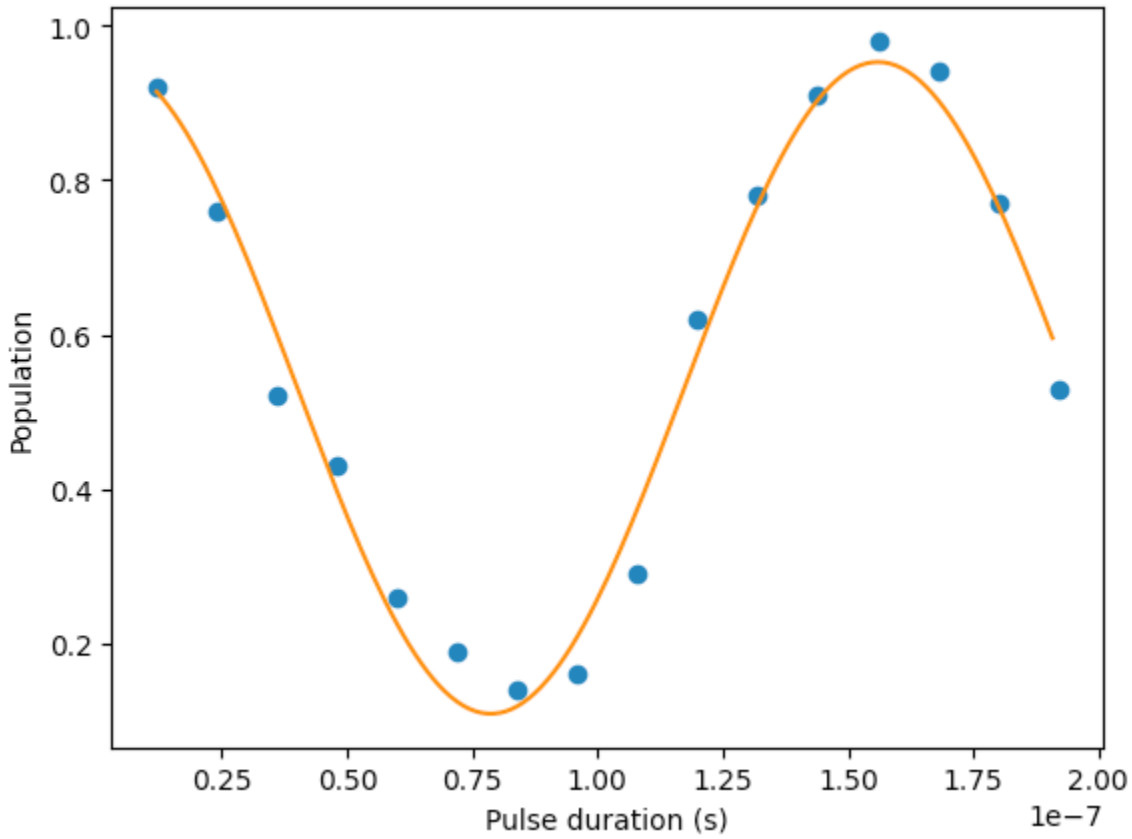
Infine, mettili insieme come sequenza di impulsi. Nella sequenza di impulsi, `play` riproduce la forma d'onda specificata sul frame di trasmissione e quindi `capture_v0` misura lo stato a partire dal riquadro di lettura.

```
pulse_sequence = (
    PulseSequence()
    .play(drive_frame, waveform)
    .capture_v0(readout_frame)
)
```

Esegue la scansione su un intervallo di lunghezza degli impulsi e li invia alla QPU. Prima di eseguire le sequenze di impulsi su una QPU, associa il valore dei parametri liberi.

```
start_length = 12e-9
end_length = 2e-7
lengths = np.arange(start_length, end_length, 12e-9)
N_shots = 100

tasks = [
    device.run(pulse_sequence(length=length), shots=N_shots)
    for length in lengths
]

probability_of_zero = [
    task.result().measurement_counts['0']/N_shots
    for task in tasks
]
```

Le statistiche della misurazione del qubit mostrano la dinamica oscillatoria del qubit che oscilla tra lo stato 0 e lo stato 1. Dai dati di misurazione, è possibile estrarre la frequenza Rabi e regolare con precisione la lunghezza dell'impulso per implementare una particolare porta da 1 qubit. Ad esempio, dai dati riportati nella figura seguente, la periodicità è di circa 154 ns. Quindi una porta di rotazione pi/2 corrisponderebbe alla sequenza di impulsi con lunghezza = 38,5 ns.



Hello Pulse sta usando OpenPulse

OpenPulseè un linguaggio per specificare il controllo a livello di impulsi di un dispositivo quantistico generale e fa parte delle specifiche OpenQASM 3.0. Amazon Braket supporta OpenPulse la programmazione diretta degli impulsi utilizzando la rappresentazione OpenQASM 3.0.

Braket lo utilizza OpenPulse come rappresentazione intermedia sottostante per esprimere gli impulsi nelle istruzioni native. OpenPulsesupporta l'aggiunta di calibrazioni delle istruzioni sotto forma di dichiarazioni `defcal` (abbreviazione di «definisci la calibrazione»). Con queste dichiarazioni, è possibile specificare un'implementazione di un'istruzione gate all'interno di una grammatica di controllo di livello inferiore.

È possibile visualizzare il OpenPulse programma di un Braket `PulseSequence` utilizzando il seguente comando.

```
print(pulse_sequence.to_ir())
```

Puoi anche costruire direttamente un OpenPulse programma.

```
from braket.ir.openqasm import Program

openpulse_script = """
OPENQASM 3.0;
cal {
    bit[1] psb;
    waveform my_waveform = gaussian(12.0ns, 3.0ns, 0.2, false);
    play(Transmon_25_charge_tx, my_waveform);
    psb[0] = capture_v0(Transmon_25_readout_rx);
}
"""
```

Crea un `Program` oggetto con il tuo script. Quindi, invia il programma a una QPU.

```
from braket.aws import AwsDevice
from braket.devices import Devices
from braket.ir.openqasm import Program

program = Program(source=openpulse_script)

device = AwsDevice(Devices.Rigetti.Ankaa3)
task = device.run(program, shots=100)
```

Accesso ai gate nativi tramite impulsi

I ricercatori spesso hanno bisogno di sapere esattamente come le porte native supportate da una particolare QPU vengono implementate come impulsi. Le sequenze di impulsi vengono calibrate con cura dai fornitori di hardware, ma accedervi offre ai ricercatori l'opportunità di progettare gate migliori o di esplorare protocolli per la mitigazione degli errori, come l'estrapolazione a zero rumore mediante l'allungamento degli impulsi di porte specifiche.

Amazon Braket supporta l'accesso programmatico ai gate nativi di Rigetti.

```
import math
from braket.aws import AwsDevice
from braket.circuits import Circuit, GateCalibrations, QubitSet
from braket.circuits.gates import Rx

device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")

calibrations = device.gate_calibrations
```

```
print(f"Downloaded {len(calibrations)} calibrations.")
```

> **Note**
>
> I fornitori di hardware calibrano periodicamente la QPU, spesso più di una volta al giorno. L'SDK Braket consente di ottenere le calibrazioni dei gate più recenti.

```
device.refresh_gate_calibrations()
```

Per recuperare un determinato gate nativo, come il gate RX o XY, è necessario passare l'`Gate`oggetto e i qubit di interesse. Ad esempio, è possibile controllare l'implementazione a impulsi del RX (π /2) applicato su 0. qubit

```
rx_pi_2_q0 = (Rx(math.pi/2), QubitSet(0))

pulse_sequence_rx_pi_2_q0 = calibrations.pulse_sequences[rx_pi_2_q0]
```

È possibile creare un set filtrato di calibrazioni utilizzando la funzione. `filter` Si passa un elenco di porte o un elenco di. `QubitSet` Il codice seguente crea due set che contengono tutte le calibrazioni per RX (π /2) e per 0. qubit

```
rx_calibrations = calibrations.filter(gates=[Rx(math.pi/2)])
q0_calibrations = calibrations.filter(qubits=QubitSet([0]))
```

Ora puoi fornire o modificare l'azione delle porte native collegando un set di calibrazione personalizzato. Ad esempio, si consideri il seguente circuito.

```
bell_circuit = (
    Circuit()
    .rx(0, math.pi/2)
    .rx(1, math.pi/2)
    .iswap(0, 1)
    .rx(1, -math.pi/2)
)
```

Puoi eseguirlo con una calibrazione di gate personalizzata per il `rx` gate on `qubit 0` passando un dizionario di `PulseSequence` oggetti all'argomento della `gate_definitions` parola chiave. È possibile costruire un dizionario dall'attributo `pulse_sequences` dell'`GateCalibrations`oggetto.

Tutte le porte non specificate vengono sostituite con la calibrazione a impulsi del fornitore di hardware quantistico.

```
nb_shots = 50
custom_calibration = GateCalibrations({rx_pi_2_q0: pulse_sequence_rx_pi_2_q0})
task = device.run(bell_circuit, gate_definitions=custom_calibration.pulse_sequences,
 shots=nb_shots)
```

Simulazione hamiltoniana analogica

La simulazione hamiltoniana analogica (AHS) è un paradigma emergente nell'informatica quantistica che si differenzia significativamente dal modello di circuito quantistico tradizionale. Invece di una sequenza di porte, in cui ogni circuito agisce solo su un paio di qubit alla volta. Un programma AHS è definito dai parametri dipendenti dal tempo e dallo spazio dell'hamiltoniano in questione. L'hamiltoniano di un sistema codifica i suoi livelli di energia e gli effetti delle forze esterne, che insieme governano l'evoluzione temporale dei suoi stati. Per un sistema a N-qubit, l'hamiltoniano può essere rappresentato da una matrice quadrata di numeri complessi di 2 N X2 N.

I dispositivi quantistici in grado di eseguire AHS sono progettati per approssimare da vicino l'evoluzione temporale di un sistema quantistico in un sistema hamiltoniano personalizzato regolando attentamente i parametri di controllo interno. Ad esempio, la regolazione dell'ampiezza e la desintonizzazione dei parametri di un campo di guida coerente. Il paradigma AHS è adatto per simulare le proprietà statiche e dinamiche dei sistemi quantistici con molte particelle interagenti, come nella fisica della materia condensata o nella chimica quantistica. Le unità di elaborazione quantistica costruite appositamente (QPUs), come il dispositivo Aquila, sono state sviluppate per sfruttare la potenza dell'AHS e affrontare problemi che esulano dalla QuEra portata degli approcci convenzionali di calcolo quantistico digitale in modi innovativi.

In questa sezione:

- Hello AHS: esegui la tua prima simulazione hamiltoniana analogica
- Invia un QuEra programma analogico usando Aquila

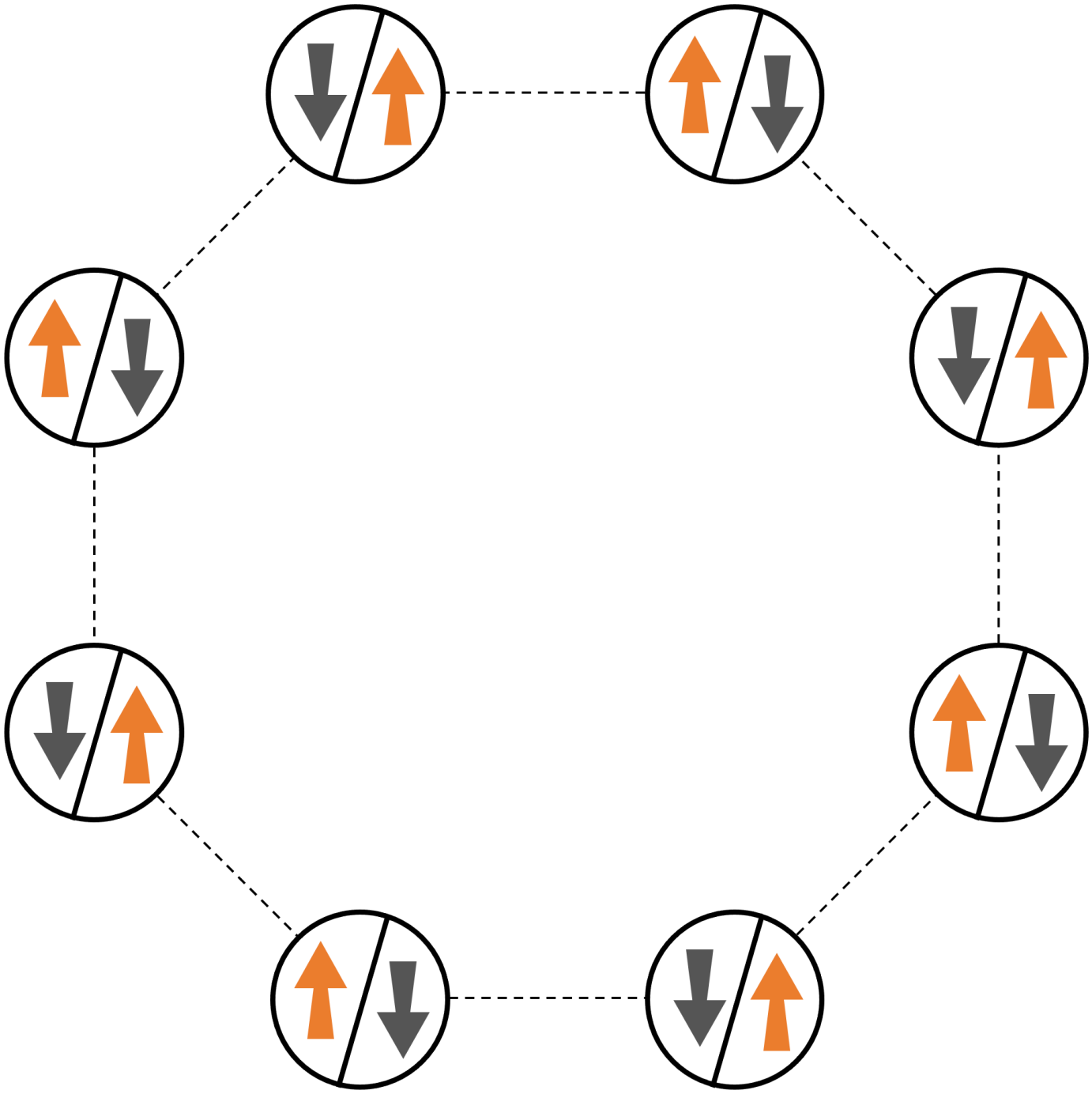
Hello AHS: esegui la tua prima simulazione hamiltoniana analogica

Questa sezione fornisce informazioni sull'esecuzione della prima simulazione hamiltoniana analogica.

In questa sezione:

Catena di rotazione interagente

Per un esempio canonico di un sistema di molte particelle interagenti, consideriamo un anello di otto spin (ognuno dei quali può trovarsi negli stati «su» ↑# e «giù» ↓ #). Sebbene piccolo, questo sistema modello mostra già una manciata di interessanti fenomeni legati ai materiali magnetici presenti in natura. In questo esempio, mostreremo come preparare un cosiddetto ordine antiferromagnetico, in cui gli spin consecutivi puntano in direzioni opposte.



Disposizione

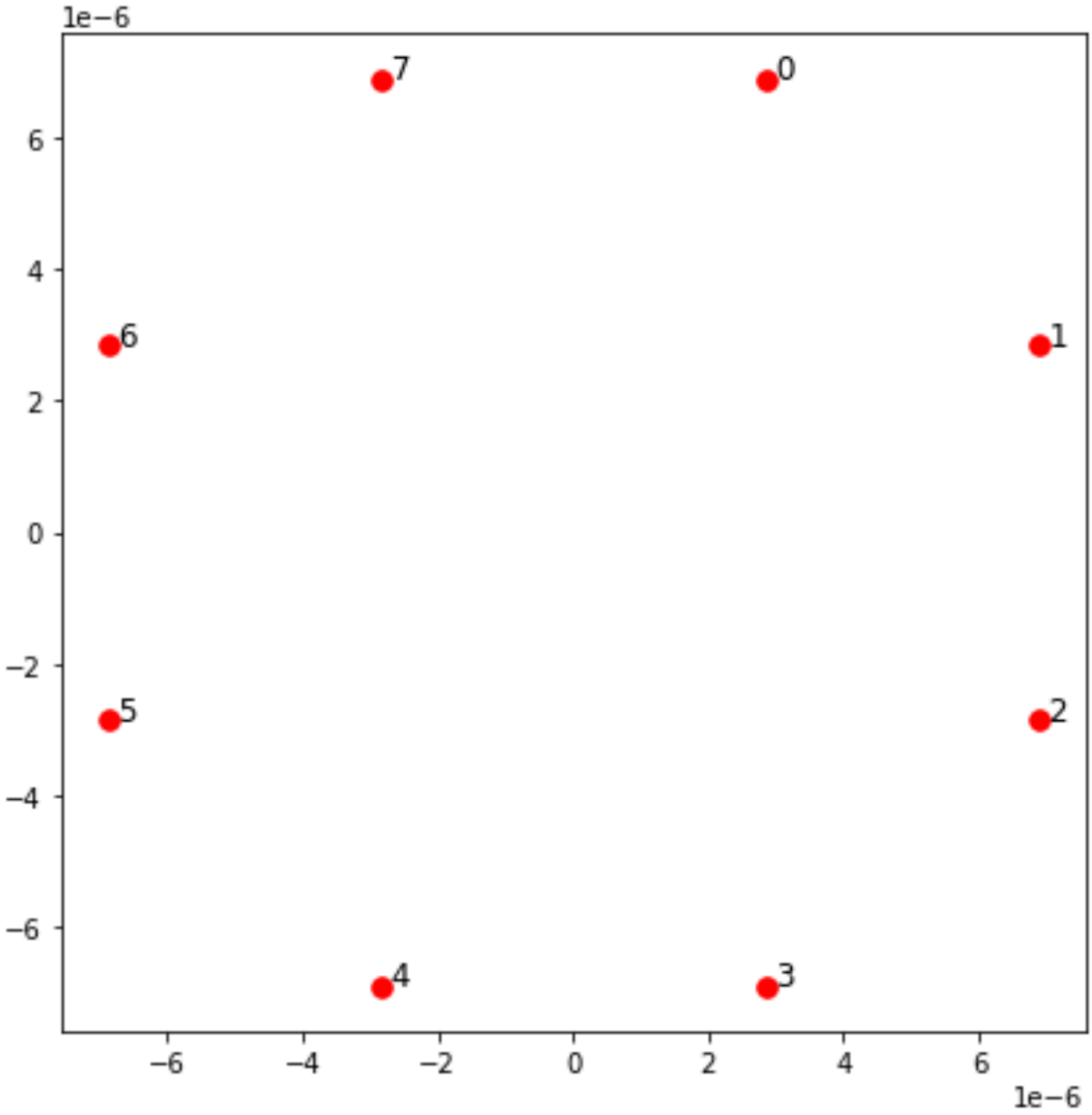
Useremo un atomo neutro per rappresentare ogni spin, e gli stati di spin «su» e «giù» saranno codificati rispettivamente nello stato eccitato di Rydberg e nello stato fondamentale degli atomi. Per prima cosa, creiamo la disposizione bidimensionale. Possiamo programmare il suddetto anello di giri con il seguente codice.

Prerequisiti: è necessario installare pip l'SDK Braket. (Se si utilizza un'istanza di notebook ospitata da Braket, questo SDK viene preinstallato con i notebook.) Per riprodurre i grafici, è inoltre necessario installare separatamente matplotlib con il comando shell. `pip install matplotlib`

```
from braket.ahs.atom_arrangement import AtomArrangement
import numpy as np
import matplotlib.pyplot as plt  # Required for plotting

a = 5.7e-6  # Nearest-neighbor separation (in meters)

register = AtomArrangement()
register.add(np.array([0.5, 0.5 + 1/np.sqrt(2)]) * a)
register.add(np.array([0.5 + 1/np.sqrt(2), 0.5]) * a)
register.add(np.array([0.5 + 1/np.sqrt(2), - 0.5]) * a)
register.add(np.array([0.5, - 0.5 - 1/np.sqrt(2)]) * a)
register.add(np.array([-0.5, - 0.5 - 1/np.sqrt(2)]) * a)
register.add(np.array([-0.5 - 1/np.sqrt(2), - 0.5]) * a)
register.add(np.array([-0.5 - 1/np.sqrt(2), 0.5]) * a)
register.add(np.array([-0.5, 0.5 + 1/np.sqrt(2)]) * a)
```

con cui possiamo anche tracciare

```
fig, ax = plt.subplots(1, 1, figsize=(7, 7))
xs, ys = [register.coordinate_list(dim) for dim in (0, 1)]
ax.plot(xs, ys, 'r.', ms=15)

for idx, (x, y) in enumerate(zip(xs, ys)):
    ax.text(x, y, f" {idx}", fontsize=12)

plt.show()  # This will show the plot below in an ipython or jupyter session
```



Interazione

Per preparare la fase antiferromagnetica, dobbiamo indurre interazioni tra spin adiacenti. A tale scopo utilizziamo l'interazione di van der Waals, che viene implementata nativamente da dispositivi ad atomi neutri (come il dispositivo from). Aquila QuEra Utilizzando la rappresentazione dello spin, il termine hamiltoniano per questa interazione può essere espresso come somma di tutte le coppie di spin (j, k).

$$H_{\text{interaction}} = \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} V_{j,k} n_j n_k$$

In questo caso, nj=^ ↑ è un operatore che assume $_j$ il valore di 1 solo se lo spin j è nello stato $_j$ «alto», e 0 in caso contrario. La forza è V $_{j,k}$ =C_6/($d_{j,k}$) ^{6}, dove C $_6$ è il coefficiente fisso e d è la distanza euclidea tra gli spin j e $_{j,k}$ k. L'effetto immediato di questo termine di interazione è che ogni stato in cui sia lo spin j che lo spin k sono «verso l'alto» ha un'energia elevata (della quantità V). $_{j,k}$ Progettando attentamente il resto del programma AHS, questa interazione eviterà che entrambi gli spin adiacenti si trovino nello stato «attivo», un effetto comunemente noto come «blocco di Rydberg».

Campo di guida

All'inizio del programma AHS, tutti gli spin (per impostazione predefinita) iniziano nel loro stato «inattivo», si trovano in una cosiddetta fase ferromagnetica. Tenendo d'occhio il nostro obiettivo di preparare la fase antiferromagnetica, specifichiamo un campo guida coerente dipendente dal tempo che trasferisce agevolmente gli spin da questo stato a uno stato a molti corpi in cui sono preferiti gli stati «alti». L'hamiltoniano corrispondente può essere scritto come

$$H_{\text{drive}}(t) = \sum_{k=1}^{N} \frac{1}{2}\Omega(t)\left[e^{i\phi(t)} S_{-,k} + e^{-i\phi(t)} S_{+,k}\right] - \sum_{k=1}^{N} \Delta(t) n_k$$

dove Ω (t), 4444t (t), Δ (t) sono l'ampiezza globale (nota anche come frequenza Rabi), la fase e la desintonizzazione del campo di pilotaggio dipendenti dal tempo che influiscono uniformemente su tutti gli spin. Qui S $_{-,k}$ =↓ $_k$ †↑ $_k$ e S $_{+,k}$ =($S_{-,k}$) † =↑ ↑ $_k$ ↓ $_k$ sono rispettivamente gli operatori di abbassamento e innalzamento dello spin k, e $_k$ n $_k$ =↑ ↑ ≤ $_k$ è lo stesso operatore di prima. La parte Ω del campo di pilotaggio accoppia in modo coerente gli stati «giù» e «su» di tutti gli spin contemporaneamente, mentre la parte Δ controlla la ricompensa energetica per gli stati «su».

Per programmare una transizione graduale dalla fase ferromagnetica alla fase antiferromagnetica, specifichiamo il campo di pilotaggio con il seguente codice.

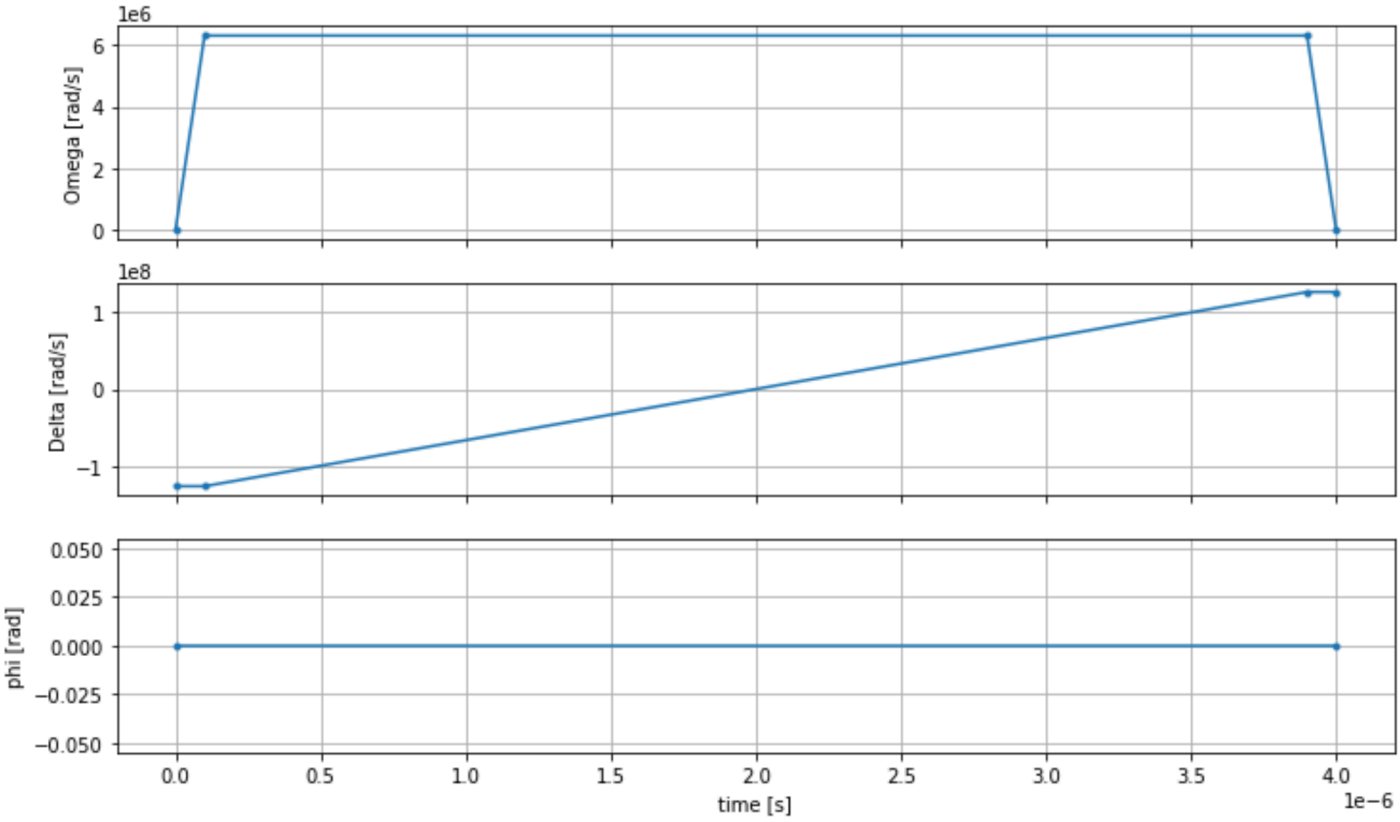
```
from braket.timings.time_series import TimeSeries
from braket.ahs.driving_field import DrivingField

# Smooth transition from "down" to "up" state
time_max = 4e-6  # seconds
time_ramp = 1e-7  # seconds
omega_max = 6300000.0  # rad / sec
```

```
delta_start = -5 * omega_max
delta_end = 5 * omega_max

omega = TimeSeries()
omega.put(0.0, 0.0)
omega.put(time_ramp, omega_max)
omega.put(time_max - time_ramp, omega_max)
omega.put(time_max, 0.0)

delta = TimeSeries()
delta.put(0.0, delta_start)
delta.put(time_ramp, delta_start)
delta.put(time_max - time_ramp, delta_end)
delta.put(time_max, delta_end)

phi = TimeSeries().put(0.0, 0.0).put(time_max, 0.0)

drive = DrivingField(
   amplitude=omega,
   phase=phi,
   detuning=delta
)
```

Possiamo visualizzare le serie temporali del campo di guida con il seguente script.

```
fig, axes = plt.subplots(3, 1, figsize=(12, 7), sharex=True)

ax = axes[0]
time_series = drive.amplitude.time_series
ax.plot(time_series.times(), time_series.values(), '.-')
ax.grid()
ax.set_ylabel('Omega [rad/s]')

ax = axes[1]
time_series = drive.detuning.time_series
ax.plot(time_series.times(), time_series.values(), '.-')
ax.grid()
ax.set_ylabel('Delta [rad/s]')

ax = axes[2]
time_series = drive.phase.time_series
# Note: time series of phase is understood as a piecewise constant function
ax.step(time_series.times(), time_series.values(), '.-', where='post')
```

```
ax.set_ylabel('phi [rad]')
ax.grid()
ax.set_xlabel('time [s]')

plt.show()  # This will show the plot below in an ipython or jupyter session
```



Programma AHS

Il registro, il campo di guida (e le interazioni implicite di van der Waals) costituiscono il programma Analog Hamiltonian Simulation. `ahs_program`

```
from braket.ahs.analog_hamiltonian_simulation import AnalogHamiltonianSimulation

ahs_program = AnalogHamiltonianSimulation(
   register=register,
   hamiltonian=drive
)
```

In esecuzione su un simulatore locale

Poiché questo esempio è piccolo (meno di 15 giri), prima di eseguirlo su una QPU compatibile con AHS, possiamo eseguirlo sul simulatore AHS locale fornito con l'SDK Braket. Poiché il simulatore locale è disponibile gratuitamente con Braket SDK, questa è la migliore pratica per garantire che il nostro codice possa essere eseguito correttamente.

Qui, possiamo impostare il numero di scatti su un valore elevato (ad esempio, 1 milione) perché il simulatore locale traccia l'evoluzione temporale dello stato quantistico e preleva campioni dallo stato finale, aumentando quindi il numero di scatti e aumentando la durata totale solo marginalmente.

```
from braket.devices import LocalSimulator

device = LocalSimulator("braket_ahs")

result_simulator = device.run(
   ahs_program,
   shots=1_000_000
).result()  # Takes about 5 seconds
```

Analisi dei risultati del simulatore

Possiamo aggregare i risultati dei colpi con la seguente funzione che deduce lo stato di ogni rotazione (che può essere «d» per «giù», «u» per «su» o «e» per il sito vuoto) e conta quante volte ogni configurazione si è verificata tra i colpi.

```
from collections import Counter


def get_counts(result):
    """Aggregate state counts from AHS shot results

    A count of strings (of length = # of spins) are returned, where
    each character denotes the state of a spin (site):
      e: empty site
      u: up state spin
      d: down state spin

    Args:
      result
 (braket.tasks.analog_hamiltonian_simulation_quantum_task_result.AnalogHamiltonianSimulationQua
```

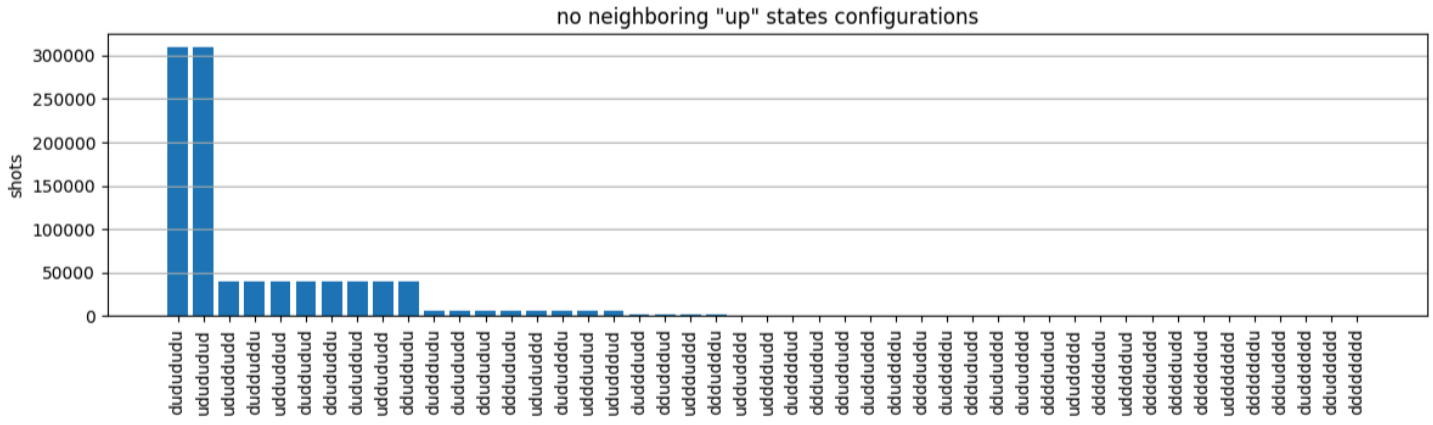
```
    Returns
       dict: number of times each state configuration is measured

    """
    state_counts = Counter()
    states = ['e', 'u', 'd']
    for shot in result.measurements:
        pre = shot.pre_sequence
        post = shot.post_sequence
        state_idx = np.array(pre) * (1 + np.array(post))
        state = "".join(map(lambda s_idx: states[s_idx], state_idx))
        state_counts.update((state,))
    return dict(state_counts)


counts_simulator = get_counts(result_simulator)  # Takes about 5 seconds
print(counts_simulator)
```

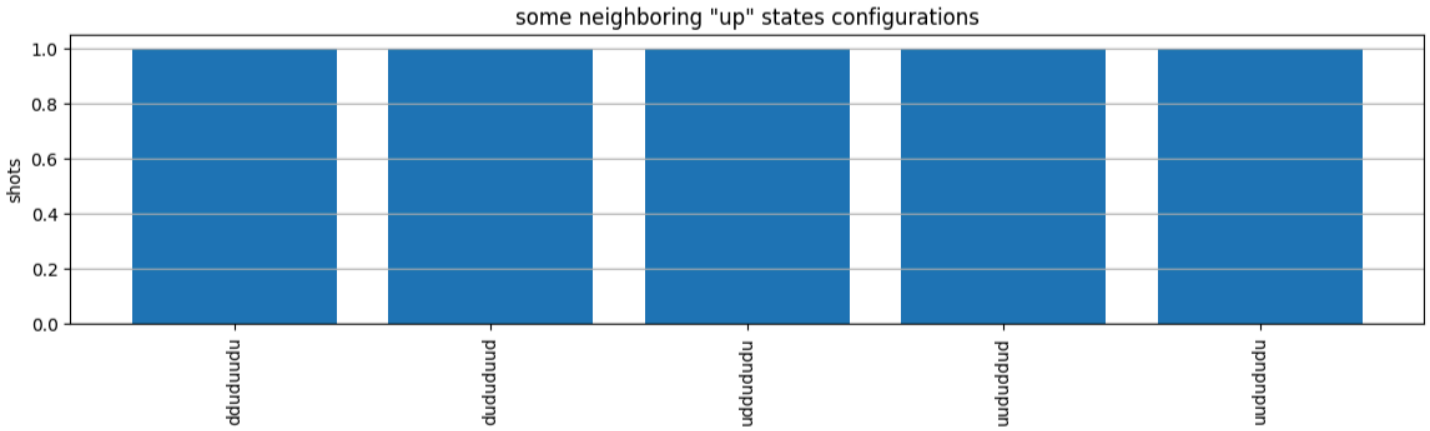
```
*[Output]*
{'dddddddd': 5, 'ddddddddu': 12, 'ddddddud': 15, ...}
```

`counts`Ecco un dizionario che conta il numero di volte in cui ogni configurazione di stato viene osservata tra le riprese. Possiamo anche visualizzarli con il codice seguente.

```
from collections import Counter


def has_neighboring_up_states(state):
    if 'uu' in state:
        return True
    if state[0] == 'u' and state[-1] == 'u':
        return True
    return False


def number_of_up_states(state):
    return Counter(state)['u']


def plot_counts(counts):
    non_blockaded = []
    blockaded = []
```

```
    for state, count in counts.items():
        if not has_neighboring_up_states(state):
            collection = non_blockaded
        else:
            collection = blockaded
        collection.append((state, count, number_of_up_states(state)))

    blockaded.sort(key=lambda _: _[1], reverse=True)
    non_blockaded.sort(key=lambda _: _[1], reverse=True)

    for configurations, name in zip((non_blockaded,
                                     blockaded),
                                    ('no neighboring "up" states',
                                     'some neighboring "up" states')):
        plt.figure(figsize=(14, 3))
        plt.bar(range(len(configurations)), [item[1] for item in configurations])
        plt.xticks(range(len(configurations)))
        plt.gca().set_xticklabels([item[0] for item in configurations], rotation=90)
        plt.ylabel('shots')
        plt.grid(axis='y')
        plt.title(f'{name} configurations')
        plt.show()


plot_counts(counts_simulator)
```





Dai grafici, possiamo leggere le seguenti osservazioni per verificare di aver preparato con successo la fase antiferromagnetica.

1. In genere, gli stati non bloccati (in cui non ci sono due spin adiacenti nello stato «attivo») sono più comuni degli stati in cui almeno una coppia di spin adiacenti si trova entrambi nello stato «positivo».
2. In genere, sono preferiti gli stati con più eccitazioni «in alto», a meno che la configurazione non sia bloccata.
3. Gli stati più comuni sono infatti gli stati antiferromagnetici perfetti e. `"dudududu" "udududud"`
4. I secondi stati più comuni sono quelli in cui ci sono solo 3 eccitazioni «verso l'alto» con separazioni consecutive di 1, 2, 2. Ciò dimostra che l'interazione di van der Waals ha un effetto (anche se molto minore) anche sui vicini più prossimi.

È in esecuzione la QPU Aquila QuEra

Prerequisiti: oltre all'installazione pip dell'SDK Braket, se non utilizzi Amazon Braket, assicurati di aver completato i passaggi introduttivi necessari.

> **Note**
>
> Se utilizzi un'istanza di notebook ospitata da Braket, l'SDK Braket viene preinstallato con l'istanza.

Con tutte le dipendenze installate, possiamo connetterci alla QPU. Aquila

```
from braket.aws import AwsDevice
```

```
aquila_qpu = AwsDevice("arn:aws:braket:us-east-1::device/qpu/quera/Aquila")
```

Per rendere il nostro programma AHS adatto alla QuEra macchina, dobbiamo arrotondare tutti i valori per rispettare i livelli di precisione consentiti dalla Aquila QPU. (Questi requisiti sono regolati dai parametri del dispositivo con «Risoluzione» nel nome. Possiamo vederli `aquila_qpu.properties.dict()` eseguendoli su un notebook. Per ulteriori dettagli sulle funzionalità e sui requisiti di Aquila, vedere il notebook Introduzione ad Aquila.) Possiamo farlo chiamando il `discretize` metodo.

```
discretized_ahs_program = ahs_program.discretize(aquila_qpu)
```

Ora possiamo eseguire il programma (per ora eseguendo solo 100 scatti) sulla Aquila QPU.

> **Note**
>
> L'esecuzione di questo programma sul Aquila processore comporterà un costo. L'SDK Amazon Braket include un Cost Tracker che consente ai clienti di impostare limiti di costo e di tenere traccia dei costi quasi in tempo reale.

```
task = aquila_qpu.run(discretized_ahs_program, shots=100)

metadata = task.metadata()
task_arn = metadata['quantumTaskArn']
task_status = metadata['status']

print(f"ARN: {task_arn}")
print(f"status: {task_status}")
```

```
*[Output]*
ARN: arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef
status: CREATED
```

A causa della grande differenza tra il tempo di esecuzione di un'attività quantistica (a seconda delle finestre di disponibilità e dell'utilizzo della QPU), è una buona idea annotare l'ARN dell'attività quantistica, in modo da poterne controllare lo stato in un secondo momento con il seguente frammento di codice.

```
# Optionally, in a new python session
from braket.aws import AwsQuantumTask

SAVED_TASK_ARN = "arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef"

task = AwsQuantumTask(arn=SAVED_TASK_ARN)
metadata = task.metadata()
task_arn = metadata['quantumTaskArn']
task_status = metadata['status']

print(f"ARN: {task_arn}")
print(f"status: {task_status}")
```

```
*[Output]*
ARN: arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef
status: COMPLETED
```

Una volta che lo stato è COMPLETATO (che può essere verificato anche dalla pagina delle attività quantistiche della console Amazon Braket), possiamo interrogare i risultati con:

```
result_aquila = task.result()
```

Analisi dei risultati della QPU

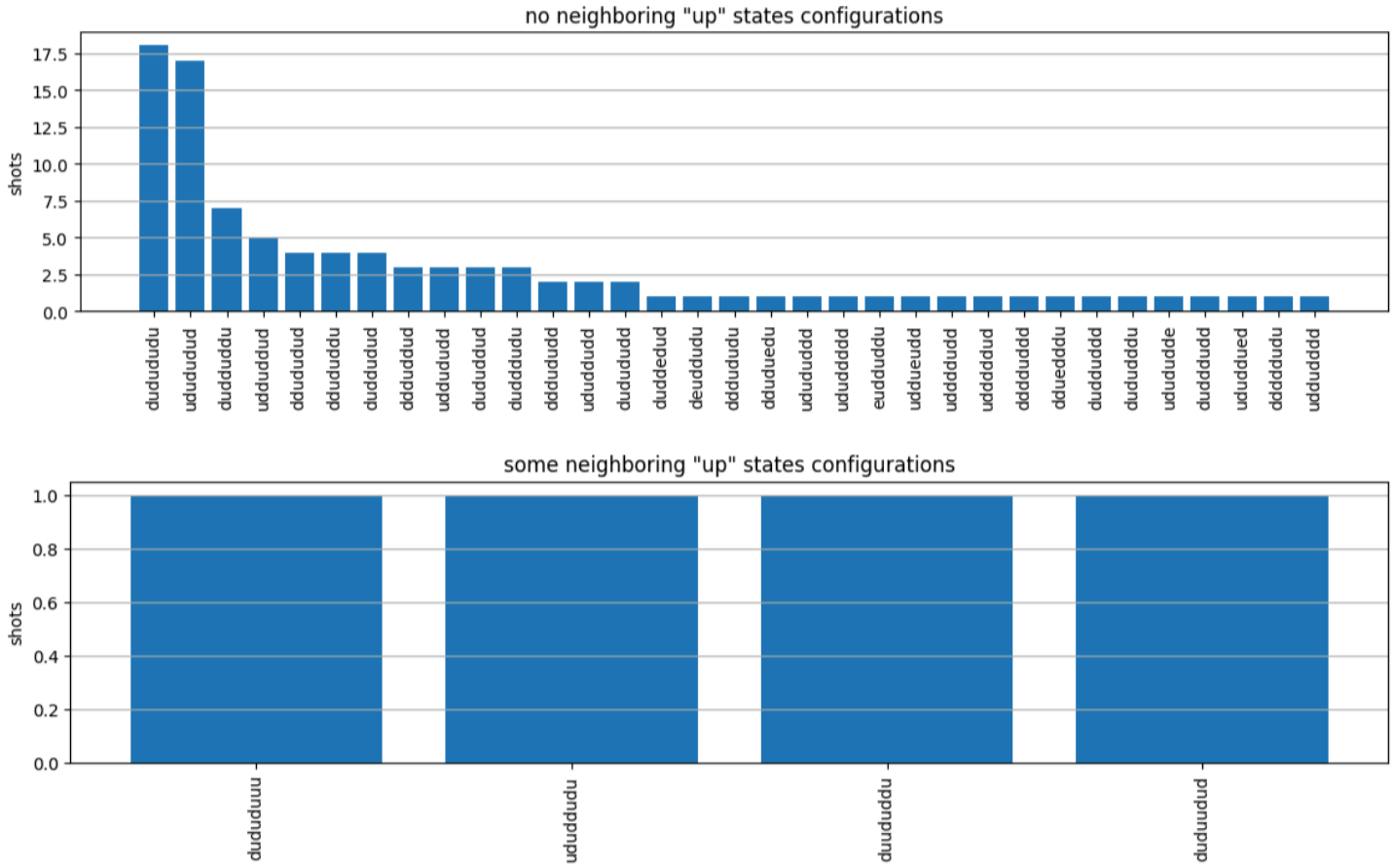
Usando le stesse `get_counts` funzioni di prima, possiamo calcolare i conteggi:

```
counts_aquila = get_counts(result_aquila)
   print(counts_aquila)
```

```
*[Output]*
{'ddddudud': 2, 'dudududu': 18, 'ddududud': 4, ...}
```

e tracciali con`plot_counts`:

```
plot_counts(counts_aquila)
```



Si noti che una piccola parte delle riprese presenta aree vuote (contrassegnate con «e»). Ciò è dovuto a imperfezioni della QPU nella preparazione dell'1-2% per atomo. Aquila Inoltre, i risultati corrispondono alla simulazione nell'ambito della fluttuazione statistica prevista a causa del numero ridotto di scatti.

Fasi successive

Congratulazioni, ora hai eseguito il tuo primo carico di lavoro AHS su Amazon Braket utilizzando il simulatore AHS locale e la QPU. Aquila

Per ulteriori informazioni sulla fisica di Rydberg, sulla simulazione hamiltoniana analogica e sul dispositivo, consulta i nostri taccuini di esempio. Aquila

Invia un QuEra programma analogico usando Aquila

Questa pagina fornisce una documentazione completa sulle funzionalità della Aquila macchina diQuEra. I dettagli trattati qui sono i seguenti:

1. L'hamiltoniano parametrizzato simulato da Aquila

2. Parametri del programma AHS
3. Contenuto dei risultati AHS
4. Aquilaparametro delle capacità

In questa sezione:

- hamiltoniano
- Schema del programma Braket AHS
- schema dei risultati dell'attività Braket AHS
- QuEra schema delle proprietà del dispositivo

hamiltoniano

La Aquila macchina di QuEra simula nativamente il seguente hamiltoniano (dipendente dal tempo):

$$H(t) = \sum_{k=1}^{N} H_{\text{drive},k}(t) + \sum_{k=1}^{N} H_{\text{local detuning},k}(t) + \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} V_{\text{vdw},k,l}$$

> **Note**
>
> L'accesso alla detuning locale è una funzionalità sperimentale ed è disponibile su richiesta tramite Braket Direct.

dove

- H drive,k (t) =($^{1}/_{2}$Ω (t) e i $^{(t)}$ S −,k + $^{1}/_{2}$Ω (t) s) + $^{(-\Delta\ (t_{+,k})}$ n), global k
  - Ω (t) è l'ampiezza di guida globale, dipendente dal tempo (nota anche come frequenza Rabi), espressa in unità di (rad/s)
  - □ (t) è la fase globale, dipendente dal tempo, misurata in radianti
  - S −,k e S +,k sono gli operatori che abbassano e innalzano lo spin dell'atomo k (in base |↓# =|g#, |↑=|r#, sono S =|g86.r|, S =(S) † =|rovolg|) − + −
  - Δ (t) è la desintonizzazione globale dipendente dal tempo global
  - n k è l'operatore di proiezione sullo stato di Rydberg dell'atomo k (ovvero, n=|r86.r|)
- H (t) =-Δ (t) h n local detuning,k local k k

- Δ_{local} (t) è il fattore dipendente dal tempo dello spostamento di frequenza locale, espresso in unità di (rad/s)
- h_k è il fattore dipendente dal sito, un numero adimensionale compreso tra 0,0 e 1,0
- $V_{vdw,k,l} = C_6/(d_{k,l})$ 6 n n, k l
  - C_6 è il coefficiente di van der Waals, espresso in unità di (rad/ s) * (m) ^6
  - $d_{k,l}$ è la distanza euclidea tra l'atomo k e l, misurata in metri.

Gli utenti hanno il controllo sui seguenti parametri tramite lo schema del programma Braket AHS.

- Disposizione bidimensionale degli atomi ($_k$coordinate x $_k$ e y di ciascun atomo k, in unità di um), che controlla le distanze atomiche a coppie $d_{k,l}$ con k, l=1,2,... N
- Ω (t), la frequenza Rabi globale dipendente dal tempo, in unità di (rad/ s)
- □ (t), la fase globale, dipendente dal tempo, in unità di (rad)
- Δ_{global} (t), la detonizzazione globale dipendente dal tempo, in unità di (rad/s)
- Δ_{local} (t), il fattore (globale) dipendente dal tempo dell'entità della detonizzazione locale, in unità di (rad/ s)
- h_k, il fattore (statico) dipendente dal sito dell'entità della detonizzazione locale, un numero adimensionale compreso tra 0,0 e 1,0

> **Note**
>
> L'utente non può controllare quali livelli sono coinvolti (ovvero, gli operatori S, S_{-+}, n sono fissi) né l'intensità del coefficiente di interazione Rydberg-Rydberg (C). $_6$

Schema del programma Braket AHS

Oggetto Braket.ir.ahs.program_v1.program (esempio)

> **Note**
>
> Se la funzione di desintonizzazione locale non è abilitata per il tuo account, usala nell'esempio seguente. `localDetuning=[]`

```
Program(
    braketSchemaHeader=BraketSchemaHeader(
        name='braket.ir.ahs.program',
        version='1'
    ),
    setup=Setup(
        ahs_register=AtomArrangement(
            sites=[
                [Decimal('0'), Decimal('0')],
                [Decimal('0'), Decimal('4e-6')],
                [Decimal('4e-6'), Decimal('0')]
            ],
            filling=[1, 1, 1]
        )
    ),
    hamiltonian=Hamiltonian(
        drivingFields=[
            DrivingField(
                amplitude=PhysicalField(
                    time_series=TimeSeries(
                        values=[Decimal('0'), Decimal('15700000.0'),
 Decimal('15700000.0'), Decimal('0')],
                        times=[Decimal('0'), Decimal('0.000001'), Decimal('0.000002'),
 Decimal('0.000003')]
                    ),
                    pattern='uniform'
                ),
                phase=PhysicalField(
                    time_series=TimeSeries(
                        values=[Decimal('0'), Decimal('0')],
                        times=[Decimal('0'), Decimal('0.000003')]
                    ),
                    pattern='uniform'
                ),
                detuning=PhysicalField(
                    time_series=TimeSeries(
                        values=[Decimal('-54000000.0'), Decimal('54000000.0')],
                        times=[Decimal('0'), Decimal('0.000003')]
                    ),
                    pattern='uniform'
                )
            )
        ],
```

```
        localDetuning=[
            LocalDetuning(
                magnitude=PhysicalField(
                    times_series=TimeSeries(
                        values=[Decimal('0'), Decimal('25000000.0'),
 Decimal('25000000.0'), Decimal('0')],
                        times=[Decimal('0'), Decimal('0.000001'), Decimal('0.000002'),
 Decimal('0.000003')]
                    ),
                    pattern=Pattern([Decimal('0.8'), Decimal('1.0'), Decimal('0.9')])
                )
            )
        ]
    )
)
```

JSON (esempio)

> **Note**
>
> Se la funzione di desintonizzazione locale non è abilitata per il tuo account, usala `"localDetuning": []` nell'esempio seguente.

```
{
    "braketSchemaHeader": {
        "name": "braket.ir.ahs.program",
        "version": "1"
    },
    "setup": {
        "ahs_register": {
            "sites": [
                [0E-7, 0E-7],
                [0E-7, 4E-6],
                [4E-6, 0E-7]
            ],
            "filling": [1, 1, 1]
        }
    },
    "hamiltonian": {
        "drivingFields": [
            {
```

```
                "amplitude": {
                    "time_series": {
                        "values": [0.0, 15700000.0, 15700000.0, 0.0],
                        "times": [0E-9, 0.000001000, 0.000002000, 0.000003000]
                    },
                    "pattern": "uniform"
                },
                "phase": {
                    "time_series": {
                        "values": [0E-7, 0E-7],
                        "times": [0E-9, 0.000003000]
                    },
                    "pattern": "uniform"
                },
                "detuning": {
                    "time_series": {
                        "values": [-54000000.0, 54000000.0],
                        "times": [0E-9, 0.000003000]
                    },
                    "pattern": "uniform"
                }
            }
        ],
        "localDetuning": [
            {
                "magnitude": {
                    "time_series": {
                        "values": [0.0, 25000000.0, 25000000.0, 0.0],
                        "times": [0E-9, 0.000001000, 0.000002000, 0.000003000]
                    },
                    "pattern": [0.8, 1.0, 0.9]
                }
            }
        ]
    }
}
```

Campi principali

Campo del programma	tipo	description
setup.ahs_register.sites	Elenco [Elenco [Decimale]]	Elenco delle coordinate

Campo del programma	tipo	description
		bidimensionali in cui le pinzette intrappolano gli atomi
setup.ahs_register.filling	Elenco [int]	Contrassegna gli atomi che occupano i siti trap con 1 e i siti vuoti con 0
hamiltonian.drivingFields [] .amplitude.time_series.times	Elenco [Decimale]	punti temporali dell'ampiezza di guida, Omega (t)
hamiltonian.drivingFields [] .amplitude.time_series.values	Elenco [Decimale]	valori dell'ampiezza di guida, Omega (t)
hamiltonian.drivingFields [] .amplitude.pattern	str	modello spaziale di ampiezza di guida, Omega (t); deve essere «uniforme»
hamiltonian.drivingFields [] .phase.time_series.times	Elenco [Decimale]	punti temporali della fase di guida, phi (t)
hamiltonian.drivingFields [] .phase.time_series.values	Elenco [Decimale]	valori della fase di guida, phi (t)
hamiltonian.drivingFields [] .phase.pattern	str	modello spaziale della fase di guida, phi (t); deve essere «uniforme»

Campo del programma	tipo	description
hamiltonian.drivingFields [] .detuning.time_series.times	Elenco [Decimale]	punti temporali di desintonizzazione della guida, delta_Global (t)
hamiltonian.drivingFields [] .detuning.time_series.values	Elenco [Decimale]	valori della detonizzazione della guida, delta_Global (t)
hamiltonian.drivingFields [] .detuning.pattern	str	modello spaziale di detuning di guida, delta_global (t); deve essere «uniforme»
hamiltonian.localDeTuning [] .magnitude.time_series.times	Elenco [Decimale]	punti temporali del fattore dipendente dal tempo della magnitudine locale di detonizzazione, delta_local (t)
hamiltonian.localDeTuning [] .magnitude.time_series.values	Elenco [Decimale]	valori del fattore dipendente dal tempo della magnitudine locale di detonizzazione, delta_local (t)

Campo del programma	tipo	description
hamiltonian.localDeTuning [] .magnitude.pattern	Elenco [Decimale]	fattore dipendente dal sito della magnitudine di detonizzazione locale, h_k (i valori corrispondono ai siti in setup.ahs _register.sites)

Campi di metadati

Campo del programma	tipo	description
braketSchemaHeader.name	str	nome dello schema; deve essere 'braket.ir.ahs.program'
braketSchemaHeader.versione	str	versione dello schema

schema dei risultati dell'attività Braket AHS

braket.tasks.analog_hamiltonian_simulation_quantum_task_result.
AnalogHamiltonianSimulationQuantumTaskResult(esempio)

```
AnalogHamiltonianSimulationQuantumTaskResult(
    task_metadata=TaskMetadata(
        braketSchemaHeader=BraketSchemaHeader(
            name='braket.task_result.task_metadata',
            version='1'
        ),
        id='arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef',
        shots=2,
        deviceId='arn:aws:braket:us-east-1::device/qpu/quera/Aquila',
        deviceParameters=None,
        createdAt='2022-10-25T20:59:10.788Z',
        endedAt='2022-10-25T21:00:58.218Z',
        status='COMPLETED',
```

```
        failureReason=None
    ),
    measurements=[
        ShotResult(
            status=<AnalogHamiltonianSimulationShotStatus.SUCCESS: 'Success'>,

            pre_sequence=array([1, 1, 1, 1]),
            post_sequence=array([0, 1, 1, 1])
        ),

        ShotResult(
            status=<AnalogHamiltonianSimulationShotStatus.SUCCESS: 'Success'>,

            pre_sequence=array([1, 1, 0, 1]),
            post_sequence=array([1, 0, 0, 0])
        )
    ]
)
```

JSON (esempio)

```
{
    "braketSchemaHeader": {
        "name": "braket.task_result.analog_hamiltonian_simulation_task_result",
        "version": "1"
    },
    "taskMetadata": {
        "braketSchemaHeader": {
            "name": "braket.task_result.task_metadata",
            "version": "1"
        },
        "id": "arn:aws:braket:us-east-1:123456789012:quantum-task/12345678-90ab-
cdef-1234-567890abcdef",
        "shots": 2,
        "deviceId": "arn:aws:braket:us-east-1::device/qpu/quera/Aquila",

        "createdAt": "2022-10-25T20:59:10.788Z",
        "endedAt": "2022-10-25T21:00:58.218Z",
        "status": "COMPLETED"

    },
    "measurements": [
        {
```

```
            "shotMetadata": {"shotStatus": "Success"},
            "shotResult": {
                "preSequence": [1, 1, 1, 1],
                "postSequence": [0, 1, 1, 1]
            }
        },
        {
            "shotMetadata": {"shotStatus": "Success"},
            "shotResult": {
                "preSequence": [1, 1, 0, 1],
                "postSequence": [1, 0, 0, 0]
            }
        }
    ],
    "additionalMetadata": {
        "action": {...}
        "queraMetadata": {
            "braketSchemaHeader": {
                "name": "braket.task_result.quera_metadata",
                "version": "1"
            },
            "numSuccessfulShots": 100
        }
    }
}
```

Campi principali

Campo dei risultati dell'attività	tipo	description
misure [] .shotResult.presequence	Elenco [int]	Bit di misurazione pre-sequenza (uno per ogni sito atomico) per ogni ripresa: 0 se il sito è vuoto, 1 se il sito è pieno, misurati prima delle sequenze di impulsi che eseguono l'evoluzione quantistica
misure [] .shotResult.postSequence	Elenco [int]	Bit di misurazione post-sequenza per ogni ripresa: 0 se l'atomo è nello stato di Rydberg o il sito è vuoto, 1 se l'atomo è allo stato fondamentale, misurati alla fine

Campo dei risultati dell'attività	tipo	description
		delle sequenze di impulsi che eseguono l'evoluzione quantistica

Campi di metadati

Campo dei risultati dell'attività	tipo	description
braketSchemaHeader.name	str	nome dello schema; deve essere 'braket.task_result.analog_hamiltonian_simulation_task_result'
braketSchemaHeader.versione	str	versione dello schema
Metadati delle attività. braketSchemaHeader.nome	str	nome dello schema; deve essere 'braket.task_result.task_metadata'
Metadati dell'attività. braketSchemaHeader.versione	str	versione dello schema
taskmetadata.id	str	L'ID dell'attività quantistica. Per le attività AWS quantistiche, questo è il compito

Campo dei risultati dell'attività	tipo	description
		quantistico ARN.
TaskMetadata.shots	int	Il numero di scatti per l'operazione quantistica
taskmetadata.shots.deviceID	str	L'ID del dispositivo su cui è stata eseguita l'operazione quantistica. Per AWS i dispositivi, questo è l'ARN del dispositivo.
taskmetadata.shots.createdat at	str	Il timestamp della creazione ; il formato deve essere in ISO-8601/ string format:mm :SS.SSSZ. RFC3339 YYYY-MM-DDTHH L'impostazione predefinita è Nessuno.

Campo dei risultati dell'attività	tipo	description
taskmetadata.shots.EndedAt	str	Il timestamp di quando è terminato il task quantistico; il formato deve essere in ISO-8601/ string format:mm :SS.SSSZ. RFC3339 YYYY-MM-DDTHH L'impostazione predefinita è Nessuno.
taskMetadata.shots.status	str	Lo stato dell'attività quantistica (CREATED, QUEUED, RUNNING, COMPLETED, FAILED). L'impostazione predefinita è Nessuno.

Campo dei risultati dell'attività	tipo	description
taskmetadata.shots.failureReason	str	Il motivo del fallimento del compito quantistico. L'impostazione predefinita è Nessuno.
Metadati aggiuntivi. Action	Braket.ir.ahs.program_v1.program	(Vedi la sezione sullo schema del programma Braket AHS)
Metadati aggiuntivi. Azione. braketSchemaHeader.querametadata.name	str	nome dello schema; deve essere 'braket.task_result.quera_metadata'
Metadati aggiuntivi. Action. braketSchemaHeader.querametadata.version	str	versione dello schema
Metadati aggiuntivi. Action. numSuccessfulShots	int	numero di tiri completamente riusciti; deve essere uguale al numero di tiri richiesto

Campo dei risultati dell'attività	tipo	description
misure [] .shotMetadata.shotStatus	int	Lo stato dello scatto, (Successo , Riuscito parziale, Fallimento); deve essere «Riuscito»

QuEra schema delle proprietà del dispositivo

braket.device_schema.quera.quera_device_capabilities_v1. QueraDeviceCapabilities(esempio)

```
QueraDeviceCapabilities(
    service=DeviceServiceProperties(
        braketSchemaHeader=BraketSchemaHeader(
            name='braket.device_schema.device_service_properties',
            version='1'
            ),
            executionWindows=[
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.MONDAY: 'Monday'>,
                    windowStartHour=datetime.time(1, 0),
                    windowEndHour=datetime.time(23, 59, 59)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.TUESDAY: 'Tuesday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(12, 0)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.WEDNESDAY: 'Wednesday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(12, 0)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.FRIDAY: 'Friday'>,
                    windowStartHour=datetime.time(0, 0),
```

```
                    windowEndHour=datetime.time(23, 59, 59)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.SATURDAY: 'Saturday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(23, 59, 59)
                ),
                DeviceExecutionWindow(
                    executionDay=<ExecutionDay.SUNDAY: 'Sunday'>,
                    windowStartHour=datetime.time(0, 0),
                    windowEndHour=datetime.time(12, 0)
                )
            ],
            shotsRange=(1, 1000),
            deviceCost=DeviceCost(
                price=0.01,
                unit='shot'
            ),
            deviceDocumentation=
                DeviceDocumentation(
                    imageUrl='https://
a.b.cdn.console.awsstatic.com/59534b58c709fc239521ef866db9ea3f1aba73ad3ebcf60c23914ad8c5c5c878/
a6cfc6fca26cf1c2e1c6.png',
                    summary='Analog quantum processor based on neutral atom arrays',
                    externalDocumentationUrl='https://www.quera.com/aquila'
                ),
                deviceLocation='Boston, USA',
                updatedAt=datetime.datetime(2024, 1, 22, 12, 0,
 tzinfo=datetime.timezone.utc),
                getTaskPollIntervalMillis=None
    ),
    action={
        <DeviceActionType.AHS: 'braket.ir.ahs.program'>: DeviceActionProperties(
                version=['1'],
                actionType=<DeviceActionType.AHS: 'braket.ir.ahs.program'>
            )
    },
    deviceParameters={},
    braketSchemaHeader=BraketSchemaHeader(
        name='braket.device_schema.quera.quera_device_capabilities',
        version='1'
    ),
    paradigm=QueraAhsParadigmProperties(
        ...
```

```
        # See https://github.com/amazon-braket/amazon-braket-schemas-python/blob/main/
src/braket/device_schema/quera/quera_ahs_paradigm_properties_v1.py
        ...
    )
)
```

JSON (esempio)

```
{
    "service": {
        "braketSchemaHeader": {
            "name": "braket.device_schema.device_service_properties",
            "version": "1"
        },
        "executionWindows": [
            {
                "executionDay": "Monday",
                "windowStartHour": "01:00:00",
                "windowEndHour": "23:59:59"
            },
            {
                "executionDay": "Tuesday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "12:00:00"
            },
            {
                "executionDay": "Wednesday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "12:00:00"
            },
            {
                "executionDay": "Friday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "23:59:59"
            },
            {
                "executionDay": "Saturday",
                "windowStartHour": "00:00:00",
                "windowEndHour": "23:59:59"
            },
            {
                "executionDay": "Sunday",
                "windowStartHour": "00:00:00",
```

```
                "windowEndHour": "12:00:00"
            }
        ],
        "shotsRange": [
            1,
            1000
        ],
        "deviceCost": {
            "price": 0.01,
            "unit": "shot"
        },
        "deviceDocumentation": {
            "imageUrl": "https://
a.b.cdn.console.awsstatic.com/59534b58c709fc239521ef866db9ea3f1aba73ad3ebcf60c23914ad8c5c5c878/
a6cfc6fca26cf1c2e1c6.png",
            "summary": "Analog quantum processor based on neutral atom arrays",
            "externalDocumentationUrl": "https://www.quera.com/aquila"
        },
        "deviceLocation": "Boston, USA",
        "updatedAt": "2024-01-22T12:00:00+00:00"
    },
    "action": {
        "braket.ir.ahs.program": {
            "version": [
                "1"
            ],
            "actionType": "braket.ir.ahs.program"
        }
    },
    "deviceParameters": {},
    "braketSchemaHeader": {
        "name": "braket.device_schema.quera.quera_device_capabilities",
        "version": "1"
    },
    "paradigm": {
        ...
        # See Aquila device page > "Calibration" tab > "JSON" page
        ...
    }
}
```

Campi delle proprietà del servizio

Campo delle proprietà del servizio	tipo	description
service.executionWindows [] .executionDay	ExecutionDay	Giorni della finestra di esecuzione; deve essere «Tutti i giorni», «Giorni feriali», «Fine settimana », «lunedì», «martedì» , «mercoledì», giovedì», «venerdì», «sabato» o «domenica»
Service.executionWindows []. windowStartHour	datetime.ora	Formato UTC a 24 ore dell'ora di inizio della finestra di esecuzione
Service.ExecutionWindows []. windowEndHour	datetime.ora	Formato UTC a 24 ore dell'ora in cui termina la finestra di esecuzione
service.qpu_capabilities.service.shotsRange	Tupla [int, int]	Numero minimo e massimo di scatti per il dispositivo
service.qpu_capabilities.service.deviceCost.p rice	virgola mobile	Prezzo del dispositivo in dollari USA
service.qpu_capabilities.service.deviceCost.unit	str	unità per l'addebito del prezzo, ad esempio: 'minute', 'hour', 'shot', 'task'

Campi di metadati

Campo di metadati	tipo	description
action [] .version	str	versione dello schema del programma AHS

Campo di metadati	tipo	description
azione [] .actionType	ActionType	nome dello schema del programma AHS; deve essere 'braket.ir.ahs.program'
servizio. braketSchemaHeader.nome	str	nome dello schema; deve essere 'braket.device_schema.device_service_properties'
servizio. braketSchemaHeader.versione	str	versione dello schema
service.deviceDocumentation.imageURL	str	URL per l'immagine del dispositivo
Service.DeviceDocumentation.Summary	str	breve descrizione sul dispositivo
Documentazione Service.Device. externalDocumentationUrl	str	URL della documentazione esterna
service.deviceLocation	str	posizione geografica del dispositivo
Service.UpdateDat	datetime	ora in cui le proprietà del dispositivo sono state aggiornate l'ultima volta

Lavorare con AWS Boto3

Boto3 è l' AWS SDK per Python. Con Boto3, gli sviluppatori Python possono creare, configurare e gestire Servizi AWS, come Braket. Amazon Boto3 fornisce un accesso orientato agli oggetti API e di basso livello ad Amazon Braket.

Segui le istruzioni nella guida rapida di Boto3 per scoprire come installare e configurare Boto3.

Boto3 fornisce le funzionalità di base che funzionano insieme all'SDK Amazon Braket Python per aiutarti a configurare ed eseguire le tue attività quantistiche. I clienti Python devono sempre installare Boto3, perché questa è l'implementazione principale. Se desideri utilizzare metodi di supporto aggiuntivi, devi installare anche l'SDK Amazon Braket.

Ad esempio, quando chiami`CreateQuantumTask`, l'SDK Amazon Braket invia la richiesta a Boto3, che quindi chiama il. AWS API

In questa sezione:

- Attiva il client Amazon Braket Boto3
- Configura AWS CLI i profili per Boto3 e Braket SDK

Attiva il client Amazon Braket Boto3

Per utilizzare Boto3 con Amazon Braket, devi importare Boto3 e quindi definire un client da utilizzare per connetterti ad Amazon Braket. API Nell'esempio seguente, il client Boto3 è denominato. `braket`

```
import boto3
import botocore

braket = boto3.client("braket")
```

> **Note**
>
> Supporti per staffe. IPv6 Se utilizzi IPv6 solo una rete o desideri assicurarti che il tuo carico di lavoro utilizzi il IPv6 traffico, utilizza gli endpoint dual-stack come indicato nella guida agli endpoint Dual-stack e FIPS.

Ora che hai stabilito un `braket` cliente, puoi effettuare richieste ed elaborare risposte dal servizio Amazon Braket. Puoi ottenere maggiori dettagli sui dati di richiesta e risposta nell'API Reference.

Gli esempi seguenti mostrano come lavorare con dispositivi e attività quantistiche.

- Cerca dispositivi
- Recupera un dispositivo
- Crea un'attività quantistica

- Recupera un compito quantistico
- Cerca attività quantistiche
- Annulla attività quantistica

Cerca dispositivi

- `search_devices(**kwargs)`

Cerca dispositivi utilizzando i filtri specificati.

```
# Pass search filters and optional parameters when sending the
# request and capture the response
response = braket.search_devices(filters=[{
    'name': 'deviceArn',
    'values': ['arn:aws:braket:::device/quantum-simulator/amazon/sv1']
}], maxResults=10)

print(f"Found {len(response['devices'])} devices")

for i in range(len(response['devices'])):
    device = response['devices'][i]
    print(device['deviceArn'])
```

Recupera un dispositivo

- `get_device(deviceArn)`

Recupera i dispositivi disponibili in Amazon Braket.

```
# Pass the device ARN when sending the request and capture the repsonse
response = braket.get_device(deviceArn='arn:aws:braket:::device/quantum-simulator/
amazon/sv1')

print(f"Device {response['deviceName']} is {response['deviceStatus']}")
```

Crea un'attività quantistica

- `create_quantum_task(**kwargs)`

Crea un'attività quantistica.

```
# Create parameters to pass into create_quantum_task()
kwargs = {
    # Create a Bell pair
    'action': '{"braketSchemaHeader": {"name": "braket.ir.jaqcd.program", "version":
 "1"}, "results": [], "basis_rotation_instructions": [], "instructions": [{"type": "h",
 "target": 0}, {"type": "cnot", "control": 0, "target": 1}]}',
    # Specify the SV1 Device ARN
    'deviceArn': 'arn:aws:braket:::device/quantum-simulator/amazon/sv1',
    # Specify 2 qubits for the Bell pair
    'deviceParameters': '{"braketSchemaHeader": {"name":
 "braket.device_schema.simulators.gate_model_simulator_device_parameters",
 "version": "1"}, "paradigmParameters": {"braketSchemaHeader": {"name":
 "braket.device_schema.gate_model_parameters", "version": "1"}, "qubitCount": 2}}',
    # Specify where results should be placed when the quantum task completes.
    # You must ensure the S3 Bucket exists before calling create_quantum_task()
    'outputS3Bucket': 'amazon-braket-examples',
    'outputS3KeyPrefix': 'boto-examples',
    # Specify number of shots for the quantum task
    'shots': 100
}

# Send the request and capture the response
response = braket.create_quantum_task(**kwargs)

print(f"Quantum task {response['quantumTaskArn']} created")
```

Recupera un compito quantistico

- `get_quantum_task(quantumTaskArn)`

Recupera il task quantistico specificato.

```
# Pass the quantum task ARN when sending the request and capture the response
response = braket.get_quantum_task(quantumTaskArn='arn:aws:braket:us-
west-1:123456789012:quantum-task/ce78c429-cef5-45f2-88da-123456789012')

print(response['status'])
```

Cerca attività quantistiche

- `search_quantum_tasks(**kwargs)`

Cerca attività quantistiche che corrispondono ai valori di filtro specificati.

```
# Pass search filters and optional parameters when sending the
# request and capture the response
response = braket.search_quantum_tasks(filters=[{
    'name': 'deviceArn',
    'operator': 'EQUAL',
    'values': ['arn:aws:braket:::device/quantum-simulator/amazon/sv1']
}], maxResults=25)

print(f"Found {len(response['quantumTasks'])} quantum tasks")

for n in range(len(response['quantumTasks'])):
    task = response['quantumTasks'][n]
    print(f"Quantum task {task['quantumTaskArn']} for {task['deviceArn']} is
 {task['status']}")
```

Annulla attività quantistica

- `cancel_quantum_task(quantumTaskArn)`

Annulla il task quantistico specificato.

```
# Pass the quantum task ARN when sending the request and capture the response
response = braket.cancel_quantum_task(quantumTaskArn='arn:aws:braket:us-
west-1:123456789012:quantum-task/ce78c429-cef5-45f2-88da-123456789012')

print(f"Quantum task {response['quantumTaskArn']} is {response['cancellationStatus']}")
```

Configura AWS CLI i profili per Boto3 e Braket SDK

L'SDK Amazon Braket si basa sulle AWS CLI credenziali predefinite, a meno che tu non specifichi esplicitamente il contrario. Ti consigliamo di mantenere l'impostazione predefinita quando esegui su un notebook Amazon Braket gestito perché devi fornire un ruolo IAM con le autorizzazioni per avviare l'istanza del notebook.

Facoltativamente, se esegui il codice localmente (su un'istanza Amazon EC2, ad esempio), puoi stabilire AWS CLI profili denominati. Puoi assegnare a ciascun profilo un set di autorizzazioni diverso, anziché sovrascrivere regolarmente il profilo predefinito.

Questa sezione fornisce una breve spiegazione di come configurare tale CLI `profile` e come incorporare quel profilo in Amazon Braket in modo che API le chiamate vengano effettuate con le autorizzazioni di quel profilo.

In questa sezione:

- Fase 1: Configurazione di una AWS CLI locale profile
- Passaggio 2: stabilire un oggetto di sessione Boto3
- Fase 3: incorporare la sessione Boto3 nel Braket AwsSession

Fase 1: Configurazione di una AWS CLI locale **`profile`**

Spiegare come creare un utente e come configurare un profilo non predefinito non rientra nell'ambito di questo documento. Per informazioni su questi argomenti, consulta:

- Nozioni di base
- Configurazione di AWS CLI da utilizzare AWS IAM Identity Center

Per utilizzare Amazon Braket, devi fornire a questo utente e alla CLI `profile` associata le autorizzazioni Braket necessarie. Ad esempio, puoi allegare la policy. AmazonBraketFullAccess

Passaggio 2: stabilire un oggetto di sessione Boto3

Per stabilire un oggetto di sessione Boto3, utilizzate il seguente esempio di codice.

```
from boto3 import Session

# Insert CLI profile name here
boto_sess = Session(profile_name=`profile`)
```

> **Note**
>
> Se le API chiamate previste hanno restrizioni basate sulla regione che non sono allineate con la regione `profile` predefinita, è possibile specificare una regione per la sessione Boto3 come mostrato nell'esempio seguente.

```
# Insert CLI profile name _and_ region
boto_sess = Session(profile_name=`profile`, region_name=`region`)
```

Per l'argomento designato come`region`, sostituite un valore che corrisponda a uno dei valori Regioni AWS in cui Amazon Braket è disponibile, ad esempio, e così via. `us-east-1` `us-west-1`

Fase 3: incorporare la sessione Boto3 nel Braket AwsSession

L'esempio seguente mostra come inizializzare una sessione Boto3 Braket e creare un'istanza di un dispositivo in quella sessione.

```
from braket.aws import AwsSession, AwsDevice

# Initialize Braket session with Boto3 Session credentials
aws_session = AwsSession(boto_session=boto_sess)

# Instantiate any Braket QPU device with the previously initiated AwsSession
sim_arn = 'arn:aws:braket:::device/quantum-simulator/amazon/sv1'
device = AwsDevice(sim_arn, aws_session=aws_session)
```

Una volta completata questa configurazione, è possibile inviare attività quantistiche a quell'`AwsDevice`oggetto istanziato (ad esempio chiamando il comando). `device.run(…)` Tutte le API chiamate effettuate da quel dispositivo possono utilizzare le credenziali IAM associate al profilo CLI precedentemente designato. `profile`

Verifica le tue attività quantistiche con Amazon Braket

Amazon Braket offre una varietà di simulatori di circuiti quantistici ad alte prestazioni per aiutarti a testare e convalidare i tuoi algoritmi quantistici prima di eseguirli su hardware quantistico reale. Questi simulatori gestiscono il complesso software e l'infrastruttura sottostanti e i cluster Amazon Elastic Compute Cloud (Amazon EC2) per eliminare l'onere della simulazione di circuiti quantistici sulla classica infrastruttura HPC (High Performance Computing). Queste risorse ti consentono di concentrarti sullo sviluppo e l'ottimizzazione delle tue applicazioni quantistiche.

Con i simulatori di Braket, potete testare a fondo i vostri circuiti e algoritmi quantistici senza i vincoli e le limitazioni dei dispositivi quantistici fisici. Ciò consente di esplorare un'ampia gamma di concetti di calcolo quantistico, dalle porte e circuiti quantistici di base agli algoritmi quantistici più avanzati e alle tecniche di mitigazione degli errori.

L'SDK Braket semplifica l'invio delle attività quantistiche ai simulatori, consentendovi di controllare i parametri di simulazione, come il numero di scatti e il modello di rumore, per comprendere meglio il comportamento degli algoritmi quantistici. Puoi anche utilizzare le funzionalità di Amazon Braket Hybrid Job per combinare elementi di calcolo classici e quantistici, ampliando ulteriormente l'ambito dei test e della convalida.

Testando a fondo le tue attività quantistiche sui simulatori di Braket, puoi ottenere informazioni preziose, perfezionare gli algoritmi e assicurarne la correttezza prima di implementarli su hardware quantistico reale. Questo aiuta a ridurre i tempi di sviluppo, minimizzare gli errori e, in ultima analisi, accelerare i progressi nel campo dell'informatica quantistica.

In questa sezione:

Invio di attività quantistiche ai simulatori

Amazon Braket fornisce l'accesso a diversi simulatori in grado di testare le tue attività quantistiche. Puoi inviare attività quantistiche singolarmente oppure eseguire più programmi.

Simulatori

- Simulatore di matrici di densità,: DM1 `arn:aws:braket:::device/quantum-simulator/amazon/dm1`

- Simulatore vettoriale di stato,: SV1 `arn:aws:braket:::device/quantum-simulator/amazon/sv1`
- Simulatore di rete Tensor,: TN1 `arn:aws:braket:::device/quantum-simulator/amazon/tn1`
- Il simulatore locale: `LocalSimulator()`

> **Note**
>
> È possibile annullare le attività quantistiche `CREATED` nello stato QPUs e nei simulatori su richiesta. È possibile annullare le attività quantistiche `QUEUED` nello stato nel miglior modo possibile per i simulatori su richiesta e. QPUs Tieni presente che è improbabile che le attività `QUEUED` quantistiche QPU vengano annullate correttamente durante le finestre di disponibilità della QPU.

In questa sezione:

Simulatore vettoriale statale locale () **`braket_sv`**

Il simulatore vettoriale dello stato locale (`braket_sv`) fa parte dell'SDK Amazon Braket che viene eseguito localmente nel tuo ambiente. È ideale per la prototipazione rapida su piccoli circuiti (fino a 25qubits) a seconda delle specifiche hardware dell'istanza del notebook Braket o dell'ambiente locale.

Il simulatore locale supporta tutte le porte dell'SDK Amazon Braket, ma i dispositivi QPU supportano un sottoinsieme più piccolo. Puoi trovare le porte supportate di un dispositivo nelle proprietà del dispositivo.

> **Note**
>
> Il simulatore locale supporta funzionalità avanzate di OpenQASM che potrebbero non essere supportate su dispositivi QPU o altri simulatori. Per ulteriori informazioni sulle funzionalità supportate, vedere gli esempi forniti nel notebook OpenQASM Local Simulator.

Per ulteriori informazioni su come lavorare con i simulatori, consulta gli esempi di Amazon Braket.

Simulatore di matrici di densità locale () `braket_dm`

Il simulatore di matrice di densità locale (`braket_dm`) fa parte dell'SDK Amazon Braket che viene eseguito localmente nell'ambiente. È ideale per la prototipazione rapida su piccoli circuiti con rumore (fino a 12qubits) a seconda delle specifiche hardware dell'istanza del notebook Braket o dell'ambiente locale.

È possibile creare circuiti rumorosi comuni partendo da zero utilizzando operazioni di gate noise come bit-flip e depolarizing error. È inoltre possibile applicare operazioni di rumorosità a porte specifiche qubits e a porte di circuiti esistenti progettati per funzionare sia con che senza rumore.

Il simulatore `braket_dm` locale può fornire i seguenti risultati, dato il numero specificato di: shots

- Matrice a densità ridotta: Shots = 0

> **Note**
>
> Il simulatore locale supporta funzionalità avanzate di OpenQASM, che potrebbero non essere supportate su dispositivi QPU o altri simulatori. Per ulteriori informazioni sulle funzionalità supportate, vedere gli esempi forniti nel notebook OpenQASM Local Simulator.

Per saperne di più sul simulatore di matrice di densità locale, vedi l'esempio del simulatore di rumore introduttivo di Braket.

Simulatore AHS locale () `braket_ahs`

Il simulatore AHS (Analog Hamiltonian Simulation) locale (`braket_ahs`) fa parte dell'SDK Amazon Braket che viene eseguito localmente nel tuo ambiente. Può essere usato per simulare i risultati di un programma AHS. È ideale per la prototipazione su registri di piccole dimensioni (fino a 10-12 atomi) a seconda delle specifiche hardware dell'istanza del notebook Braket o dell'ambiente locale.

Il simulatore locale supporta i programmi AHS con un campo di pilotaggio uniforme, un campo di spostamento (non uniforme) e disposizioni atomiche arbitrarie. Per i dettagli, fate riferimento alla classe Braket AHS e allo schema del programma Braket AHS.

Per ulteriori informazioni sul simulatore AHS locale, consulta la pagina Hello AHS: Run your first Analog Hamiltonian Simulation e i taccuini di esempio Analog Hamiltonian Simulation.

Simulatore vettoriale di stato () SV1

SV1è un simulatore vettoriale di stato universale su richiesta, ad alte prestazioni. Può simulare circuiti fino a 34. qubits Ci si può aspettare che il completamento di un 34-qubit circuito denso e quadrato (profondità del circuito = 34) richieda circa 1-2 ore, a seconda del tipo di porte utilizzate e di altri fattori. I circuiti con all-to-all cancelli sono adatti per. SV1 Restituisce risultati in forme come un vettore di stato completo o una matrice di ampiezze.

SV1ha una durata massima di 6 ore. Ha un valore predefinito di 35 attività quantistiche simultanee e un massimo di 100 (50 in us-west-1 e eu-west-2) attività quantistiche simultanee.

SV1risultati

SV1può fornire i seguenti risultati, dato il numero specificato dishots:

- Esempio: Shots > 0
- Aspettativa: Shots >= 0
- Varianza: >= 0 Shots
- Probabilità: > 0 Shots
- Ampiezza: = 0 Shots
- Gradiente aggiunto: = 0 Shots

Per ulteriori informazioni sui risultati, consulta Tipi di risultati.

SV1è sempre disponibile, fa funzionare i circuiti su richiesta e può far funzionare più circuiti in parallelo. Il runtime si ridimensiona linearmente con il numero di operazioni e in modo esponenziale con il numero di. qubits Il numero di shots ha un impatto minimo sul runtime. Per saperne di più, visita Compare simulators.

I simulatori supportano tutte le porte dell'SDK Braket, ma i dispositivi QPU supportano un sottoinsieme più piccolo. Puoi trovare le porte supportate di un dispositivo nelle proprietà del dispositivo.

Simulatore di matrici di densità () DM1

DM1è un simulatore di matrici di densità su richiesta e ad alte prestazioni. Può simulare circuiti fino a 17. qubits

DM1ha una durata massima di 6 ore, un valore predefinito di 35 attività quantistiche simultanee e un massimo di 50 attività quantistiche simultanee.

DM1risultati

DM1può fornire i seguenti risultati, dato il numero specificato dishots:

- Esempio: Shots > 0
- Aspettativa: Shots >= 0
- Varianza: >= 0 Shots
- Probabilità: > 0 Shots
- Matrice a densità ridotta: Shots = 0, fino a max 8 qubits

Per ulteriori informazioni sui risultati, consulta Tipi di risultati.

DM1è sempre disponibile, fa funzionare i circuiti su richiesta e può far funzionare più circuiti in parallelo. Il runtime si ridimensiona linearmente con il numero di operazioni e in modo esponenziale con il numero di. qubits Il numero di shots ha un impatto minimo sul runtime. Per ulteriori informazioni, consulta Confrontare i simulatori.

Cancelli antirumore e limitazioni

```
AmplitudeDamping
    Probability has to be within [0,1]
BitFlip
    Probability has to be within [0,0.5]
```

```
Depolarizing
    Probability has to be within [0,0.75]
GeneralizedAmplitudeDamping
    Probability has to be within [0,1]
PauliChannel
    The sum of the probabilities has to be within [0,1]
Kraus
    At most 2 qubits
    At most 4 (16) Kraus matrices for 1 (2) qubit
PhaseDamping
    Probability has to be within [0,1]
PhaseFlip
    Probability has to be within [0,0.5]
TwoQubitDephasing
    Probability has to be within [0,0.75]
TwoQubitDepolarizing
    Probability has to be within [0,0.9375]
```

Simulatore di rete Tensor () TN1

TN1è un simulatore di rete tensoriale su richiesta e ad alte prestazioni. TN1può simulare determinati tipi di circuiti con un massimo di 50 qubits e una profondità di circuito pari o inferiore a 100. TN1è particolarmente potente per circuiti sparsi, circuiti con porte locali e altri circuiti con struttura speciale, come i circuiti a trasformata quantistica di Fourier (QFT). TN1funziona in due fasi. Innanzitutto, la fase di prova tenta di identificare un percorso computazionale efficiente per il circuito, in modo da TN1 poter stimare la durata della fase successiva, chiamata fase di contrazione. Se il tempo di contrazione stimato supera il limite di durata della TN1 simulazione, non tenta la contrazione. TN1

TN1ha un limite di durata di 6 ore. È limitato a un massimo di 10 (5 in eu-west-2) compiti quantistici simultanei.

TN1risultati

La fase di contrazione consiste in una serie di moltiplicazioni di matrici. La serie di moltiplicazioni continua fino al raggiungimento di un risultato o fino a quando non si determina che un risultato non può essere raggiunto.

Nota: Shots deve essere > 0.

I tipi di risultati includono:

- Project N.E.M.O.

- Aspettativa
- Varianza

Per ulteriori informazioni sui risultati, consulta Tipi di risultati.

TN1è sempre disponibile, fa funzionare i circuiti su richiesta e può far funzionare più circuiti in parallelo. Per ulteriori informazioni, consulta Confronta i simulatori.

I simulatori supportano tutte le porte dell'SDK Braket, ma i dispositivi QPU supportano un sottoinsieme più piccolo. Puoi trovare le porte supportate di un dispositivo nelle proprietà del dispositivo.

Visita il GitHub repository Amazon Braket per un TN1 esempio di notebook con cui iniziare. TN1

Le migliori pratiche per lavorare con TN1

- Evita i all-to-all circuiti.
- Prova un nuovo circuito o una nuova classe di circuiti con un numero limitato di circuitishots, per conoscere la «durezza» del circuito. TN1
- Suddividi shot simulazioni di grandi dimensioni su più attività quantistiche.

Informazioni sui simulatori incorporati

I simulatori integrati funzionano incorporando la simulazione direttamente nel codice dell'algoritmo. Inoltre, è contenuto nello stesso contenitore ed esegue la simulazione direttamente sull'istanza di lavoro ibrida. Questo approccio è utile per eliminare i colli di bottiglia tipicamente associati alla comunicazione tra la simulazione e un dispositivo remoto. Mantenendo tutti i calcoli in un unico ambiente coeso, i simulatori integrati possono ridurre notevolmente i requisiti di memoria e diminuire il numero di esecuzioni dei circuiti necessarie per raggiungere il risultato desiderato. Ciò può portare a miglioramenti sostanziali delle prestazioni, spesso di dieci o più volte, rispetto alle configurazioni tradizionali che si basano sulla simulazione remota. Per ulteriori informazioni su come i simulatori integrati migliorano le prestazioni e consentono lavori ibridi semplificati, consulta la pagina della documentazione Esegui un lavoro ibrido con Amazon Braket Hybrid Jobs.

PennyLanedei simulatori di fulmini

Puoi usare i simulatori PennyLane di fulmini come simulatori integrati su Braket. Con i simulatori PennyLane di fulmini, puoi utilizzare metodi avanzati di calcolo del gradiente, come la

differenziazione aggiuntiva, per valutare i gradienti più velocemente. Il simulatore lightning.qubit è disponibile come dispositivo tramite Braket NBIs e come simulatore integrato, mentre il simulatore lightning.gpu deve essere eseguito come simulatore integrato con un'istanza GPU. Vedi i simulatori incorporati nel notebook Braket Hybrid Jobs per un esempio di utilizzo di lightning.gpu.

Confronta i simulatori Amazon Braket

Questa sezione ti aiuta a selezionare il simulatore Amazon Braket più adatto alle tue attività quantistiche, descrivendo alcuni concetti, limitazioni e casi d'uso.

Scelta tra simulatori locali e simulatori on-demand (,,) SV1 TN1 DM1

Le prestazioni dei simulatori locali dipendono dall'hardware che ospita l'ambiente locale, ad esempio un'istanza del notebook Braket, utilizzata per eseguire il simulatore. I simulatori on-demand vengono eseguiti nel AWS cloud e sono progettati per scalare oltre i tipici ambienti locali. I simulatori on-demand sono ottimizzati per circuiti più grandi, ma aggiungono un sovraccarico di latenza per attività quantistiche o batch di attività quantistiche. Ciò può comportare un compromesso se sono coinvolte molte attività quantistiche. Date queste caratteristiche prestazionali generali, le seguenti linee guida possono aiutarvi a scegliere come eseguire le simulazioni, comprese quelle con rumore.

Per le simulazioni:

- Se ne impiegate meno di 18qubits, utilizzate un simulatore locale.
- Quando impieghi 18-24 anniqubits, scegli un simulatore in base al carico di lavoro.
- Se ne impieghi più di 24qubits, utilizza un simulatore su richiesta.

Per le simulazioni del rumore:

- Se ne impieghi meno di 9qubits, usa un simulatore locale.
- Se impiegate 9-12 personequbits, scegliete un simulatore in base al carico di lavoro.
- Quando ne impieghi più di 12, usa. qubits DM1

Cos'è un simulatore vettoriale di stato?

SV1è un simulatore vettoriale di stato universale. Memorizza la funzione d'onda completa dello stato quantistico e applica in sequenza le operazioni di gate allo stato. Memorizza tutte le possibilità, anche quelle estremamente improbabili. Il tempo di esecuzione del SV1 simulatore per un'operazione quantistica aumenta linearmente con il numero di porte nel circuito.

Cos'è un simulatore di matrici di densità?

DM1simula circuiti quantistici con rumore. Memorizza la matrice a piena densità del sistema e applica in sequenza le porte e le operazioni di rumorosità del circuito. La matrice di densità finale contiene informazioni complete sullo stato quantistico dopo il funzionamento del circuito. Il runtime generalmente scala linearmente con il numero di operazioni e in modo esponenziale con il numero di. qubits

Cos'è un simulatore di rete tensoriale?

TN1codifica i circuiti quantistici in un grafo strutturato.

- I nodi del grafico sono costituiti da porte quantistiche, o. qubits
- I bordi del grafico rappresentano le connessioni tra le porte.

Come risultato di questa struttura, TN1 può trovare soluzioni simulate per circuiti quantistici relativamente grandi e complessi.

TN1richiede due fasi

In genere, TN1 funziona con un approccio in due fasi alla simulazione del calcolo quantistico.

- La fase di prova: in questa fase, TN1 escogita un modo per attraversare il grafico in modo efficiente, che prevede la visita di ogni nodo in modo da poter ottenere la misurazione desiderata. Come cliente, non vedete questa fase perché TN1 esegue entrambe le fasi contemporaneamente per voi. Completa la prima fase e determina se eseguire la seconda fase da solo in base a vincoli pratici. Una volta iniziata la simulazione, non avete alcun contributo in merito a tale decisione.
- La fase di contrazione: questa fase è analoga alla fase di esecuzione di un calcolo in un computer classico. La fase consiste in una serie di moltiplicazioni matriciali. L'ordine di queste moltiplicazioni ha un grande effetto sulla difficoltà del calcolo. Pertanto, la fase di prova viene completata innanzitutto per trovare i percorsi di calcolo più efficaci sul grafico. Dopo aver trovato il percorso di contrazione durante la fase di prova, unisce le porte del circuito per TN1 produrre i risultati della simulazione.

TN1i grafici sono analoghi a una mappa

Metaforicamente, puoi confrontare il TN1 grafico sottostante con le strade di una città. In una città con una griglia pianificata, è facile trovare un percorso verso la destinazione utilizzando una mappa. In

una città con strade non pianificate, nomi di strade duplicati e così via, può essere difficile trovare un percorso verso la destinazione guardando una mappa.

Se TN1 non eseguissi la fase di prova, sarebbe come camminare per le strade della città per trovare la tua destinazione, invece di guardare prima una mappa. Passare più tempo a guardare la mappa può davvero ripagare in termini di tempo a piedi. Allo stesso modo, la fase di prova fornisce informazioni preziose.

Si potrebbe dire che TN1 ha una certa «consapevolezza» della struttura del circuito sottostante che attraversa. Acquisisce questa consapevolezza durante la fase di prova.

Tipi di problemi più adatti a ciascuno di questi tipi di simulatori

SV1è adatto a qualsiasi classe di problemi che si basano principalmente sull'avere un certo numero di porte qubits e. In genere, il tempo necessario aumenta linearmente con il numero di porte, mentre non dipende dal numero di. shots SV1è generalmente più veloce rispetto TN1 ai circuiti inferiori a 28. qubits

SV1può essere più lento per qubit numeri più alti perché in realtà simula tutte le possibilità, anche quelle estremamente improbabili. Non ha modo di determinare quali risultati siano probabili. Pertanto, per una 30-qubit valutazione, SV1 deve calcolare 2^30 configurazioni. Il limite di 34 qubits per il SV1 simulatore Amazon Braket è un vincolo pratico dovuto alle limitazioni di memoria e archiviazione. Puoi pensarlo in questo modo: ogni volta che aggiungi un qubit aSV1, il problema diventa due volte più difficile.

Per molte classi di problemi, TN1 può valutare circuiti molto più grandi in tempi realistici rispetto a SV1 perché TN1 sfrutta la struttura del grafico. In sostanza tiene traccia dell'evoluzione delle soluzioni dal punto di partenza e conserva solo le configurazioni che contribuiscono a un'attraversamento efficiente. In altre parole, salva le configurazioni per creare un ordinamento di moltiplicazione delle matrici che si traduce in un processo di valutazione più semplice.

Perché TN1 il numero di porte qubits e conta, ma la struttura del grafico conta molto di più. Ad esempio, TN1 è molto bravo a valutare circuiti (grafici) in cui le porte sono a corto raggio (ossia, ciascuna qubit è collegata tramite porte solo alla vicina più vicinaqubits) e circuiti (grafici) in cui le connessioni (o le porte) hanno un intervallo simile. Un intervallo tipico consiste nel far sì che ciascuna qubit parli solo con TN1 altre che si trovano a 5 minuti di distanza. qubits qubits Se la maggior parte della struttura può essere scomposta in relazioni più semplici come queste, che possono essere rappresentate in matrici più, più piccole o più uniformi, TN1 esegue la valutazione in modo efficiente.

Limitazioni di TN1

TN1può essere più lento rispetto SV1 alla complessità strutturale del grafico. Per alcuni grafici, TN1 termina la simulazione dopo la fase di prova e mostra uno stato di, per uno dei `FAILED` due motivi seguenti:

- Impossibile trovare un percorso: se il grafico è troppo complesso, è troppo difficile trovare un buon percorso trasversale e il simulatore rinuncia al calcolo. TN1non può eseguire la contrazione. È possibile che venga visualizzato un messaggio di errore simile a questo: `No viable contraction path found.`
- La fase di contrazione è troppo difficile: in alcuni grafici è TN1 possibile trovare un percorso trasversale, ma la valutazione è molto lunga e richiede molto tempo. In questo caso, la contrazione è così costosa che il costo sarebbe proibitivo e si interrompe invece dopo la fase di prova. TN1 È possibile che venga visualizzato un messaggio di errore simile a questo: `Predicted runtime based on best contraction path found exceeds TN1 limit.`

> **Note**
>
> Ti viene addebitato il costo della fase di prova TN1 anche se non viene eseguita la contrazione e viene visualizzato uno stato. `FAILED`

La durata prevista dipende anche dal conteggio. shot Negli scenari peggiori, il tempo di TN1 contrazione dipende linearmente dal conteggio. shot Il circuito può essere contrattabile con un numero inferiore. shots Ad esempio, potresti inviare un compito quantistico con 100shots, che TN1 decide che non è contrattabile, ma se lo invii nuovamente con solo 10, la contrazione procede. In questa situazione, per ottenere 100 campioni, potreste inviare 10 compiti quantistici su 10 shots per lo stesso circuito e alla fine combinare i risultati.

Come buona prassi, ti consigliamo di testare sempre il tuo circuito o la tua classe di circuito con alcuni shots (ad esempio, 10) per scoprire la resistenza del circuitoTN1, prima di procedere con un numero maggiore di. shots

> **Note**
>
> La serie di moltiplicazioni che forma la fase di contrazione inizia con piccole matrici NxN. Ad esempio, un gate richiede una matrice 4x4. 2-qubit Le matrici intermedie necessarie durante una contrazione giudicata troppo difficile sono gigantesche. Un calcolo di questo

tipo richiederebbe giorni per essere completato. Ecco perché Amazon Braket non tenta contrazioni estremamente complesse.

Concurrency (Simultaneità)

Tutti i simulatori Braket ti danno la possibilità di eseguire più circuiti contemporaneamente. I limiti di concorrenza variano in base al simulatore e alla regione. Per ulteriori informazioni sui limiti di concorrenza, consulta la pagina Quote.

Esempi di attività quantistiche su Amazon Braket

Questa sezione illustra le fasi dell'esecuzione di un'operazione quantistica di esempio, dalla selezione del dispositivo alla visualizzazione del risultato. Come best practice per Amazon Braket, ti consigliamo di iniziare eseguendo il circuito su un simulatore, ad esempio. SV1

In questa sezione:

- Specificare il dispositivo
- Invia un esempio di attività quantistica
- Invia un'attività parametrizzata
- Specifica shots
- Sondaggio per visualizzare i risultati
- Visualizza i risultati di esempio

Specificare il dispositivo

Innanzitutto, selezionate e specificate il dispositivo per il vostro compito quantistico. Questo esempio mostra come scegliere il simulatore,. SV1

```
from braket.aws import AwsDevice

# Choose the on-demand simulator to run the circuit
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
```

È possibile visualizzare alcune delle proprietà di questo dispositivo come segue:

```
print(device.name)
```

```
for iter in device.properties.action['braket.ir.jaqcd.program']:
    print(iter)
```

```
SV1
('version', ['1.0', '1.1'])
('actionType', 'braket.ir.jaqcd.program')
('supportedOperations', ['ccnot', 'cnot', 'cphaseshift', 'cphaseshift00',
 'cphaseshift01', 'cphaseshift10', 'cswap', 'cy', 'cz', 'ecr', 'h', 'i', 'iswap',
 'pswap', 'phaseshift', 'rx', 'ry', 'rz', 's', 'si', 'swap', 't', 'ti', 'unitary', 'v',
 'vi', 'x', 'xx', 'xy', 'y', 'yy', 'z', 'zz'])
('supportedResultTypes', [ResultType(name='Sample', observables=['x', 'y', 'z', 'h',
 'i', 'hermitian'], minShots=1, maxShots=100000), ResultType(name='Expectation',
 observables=['x', 'y', 'z', 'h', 'i', 'hermitian'], minShots=0, maxShots=100000),
 ResultType(name='Variance', observables=['x', 'y', 'z', 'h', 'i', 'hermitian'],
 minShots=0, maxShots=100000), ResultType(name='Probability', observables=None,
 minShots=1, maxShots=100000), ResultType(name='Amplitude', observables=None,
 minShots=0, maxShots=0)])
('disabledQubitRewiringSupported', None)
```

Invia un esempio di attività quantistica

Invia un esempio di attività quantistica da eseguire sul simulatore on-demand.

```
from braket.circuits import Circuit, Observable

# Create a circuit with a result type
circ = Circuit().rx(0, 1).ry(1, 0.2).cnot(0, 2).variance(observable=Observable.Z(),
 target=0)
# Add another result type
circ.probability(target=[0, 2])

# Set up S3 bucket (where results are stored)
my_bucket = "amazon-braket-s3-demo-bucket"  # The name of the bucket
my_prefix = "your-folder-name"  # The name of the folder in the bucket
s3_location = (my_bucket, my_prefix)

# Submit the quantum task to run
my_task = device.run(circ, s3_location, shots=1000, poll_timeout_seconds=100,
 poll_interval_seconds=10)
# The positional argument for the S3 bucket is optional if you want to specify a bucket
 other than the default

# Get results of the quantum task
```

```
result = my_task.result()
```

Il `device.run()` comando crea un'attività quantistica tramite l'API. `CreateQuantumTask` Dopo un breve periodo di inizializzazione, l'attività quantistica viene messa in coda fino a quando non esiste la capacità necessaria per eseguirla su un dispositivo. In questo caso, il dispositivo è. SV1 Dopo che il dispositivo ha completato il calcolo, Amazon Braket scrive i risultati nella posizione Amazon S3 specificata nella chiamata. L'argomento posizionale `s3_location` è obbligatorio per tutti i dispositivi tranne il simulatore locale.

> **Note**
>
> L'azione del task quantistico di Braket ha una dimensione limitata a 3 MB.

Invia un'attività parametrizzata

I simulatori locali e su richiesta di Amazon Braket supportano QPUs anche la specificazione di valori di parametri liberi al momento dell'invio delle attività. Puoi farlo utilizzando l'`inputs`argomento to`device.run()`, come mostrato nell'esempio seguente. `inputs`Deve essere un dizionario di coppie string-float, in cui le chiavi sono i nomi dei parametri.

La compilazione parametrica può migliorare le prestazioni di esecuzione di circuiti parametrici in alcuni casi. QPUs Quando invia un circuito parametrico come attività quantistica a una QPU supportata, Braket compilerà il circuito una sola volta e memorizzerà il risultato nella cache. Non è prevista alcuna ricompilazione per i successivi aggiornamenti dei parametri sullo stesso circuito, con conseguente tempi di esecuzione più rapidi per le attività che utilizzano lo stesso circuito. Braket utilizza automaticamente i dati di calibrazione aggiornati del fornitore di hardware durante la compilazione del circuito per garantire risultati della massima qualità.

> **Note**
>
> La compilazione parametrica è supportata su tutti i moduli superconduttori basati su gate, ad eccezione dei programmi a livello di QPUs impulsi. Rigetti Computing

```
from braket.circuits import Circuit, FreeParameter, Observable

# Create the free parameters
alpha = FreeParameter('alpha')
```

```
beta = FreeParameter('beta')

# Create a circuit with a result type
circ = Circuit().rx(0, alpha).ry(1, alpha).cnot(0, 2).xx(0, 2, beta)
circ.variance(observable=Observable.Z(), target=0)

# Add another result type
circ.probability(target=[0, 2])

# Submit the quantum task to run
my_task = device.run(circ, inputs={'alpha': 0.1, 'beta': 0.2}, shots=100)
```

Specifica shots

L'shotsargomento si riferisce al numero di misurazioni desiderate. shots I simulatori, ad esempio, SV1 supportano due modalità di simulazione.

- Per shots = 0, il simulatore esegue una simulazione esatta, restituendo i valori reali per tutti i tipi di risultati. (Non disponibile suTN1.)
- Per valori diversi da zeroshots, il simulatore campiona dalla distribuzione di uscita per emulare il shot rumore reale. QPUs I dispositivi QPU consentono solo > 0. shots

Per informazioni sul numero massimo di scatti per operazione quantistica, consulta Braket Quotas.

Sondaggio per visualizzare i risultati

Durante l'esecuzione`my_task.result()`, l'SDK inizia a ricercare un risultato con i parametri definiti al momento della creazione dell'attività quantistica:

- `poll_timeout_seconds`è il numero di secondi necessari per eseguire il polling dell'attività quantistica prima che scada quando si esegue l'attività quantistica sul simulatore on-demand e/o sui dispositivi QPU. Il valore predefinito è 432.000 secondi, ovvero 5 giorni.
- Nota: per i dispositivi QPU come Rigetti eIonQ, si consiglia di attendere alcuni giorni. Se il timeout per i sondaggi è troppo breve, è possibile che i risultati non vengano restituiti entro il tempo di polling. Ad esempio, quando una QPU non è disponibile, viene restituito un errore di timeout locale.
- `poll_interval_seconds`è la frequenza con cui viene eseguito il polling dell'attività quantistica. Specifica la frequenza con cui si chiama il Braket API per ottenere lo stato quando l'attività quantistica viene eseguita sul simulatore on-demand e sui dispositivi QPU. Il valore predefinito è 1 secondo.

Questa esecuzione asincrona facilita l'interazione con dispositivi QPU che non sono sempre disponibili. Ad esempio, un dispositivo potrebbe non essere disponibile durante una normale finestra di manutenzione.

Il risultato restituito contiene una serie di metadati associati al task quantistico. È possibile controllare il risultato della misurazione con i seguenti comandi:

```
print('Measurement results:\n', result.measurements)
print('Counts for collapsed states:\n', result.measurement_counts)
print('Probabilities for collapsed states:\n', result.measurement_probabilities)
```

```
Measurement results:
 [[1 0 1]
 [0 0 0]
 [0 0 0]
 ...
 [0 0 0]
 [0 0 0]
 [1 0 1]]
Counts for collapsed states:
 Counter({'000': 766, '101': 220, '010': 11, '111': 3})
Probabilities for collapsed states:
 {'101': 0.22, '000': 0.766, '010': 0.011, '111': 0.003}
```

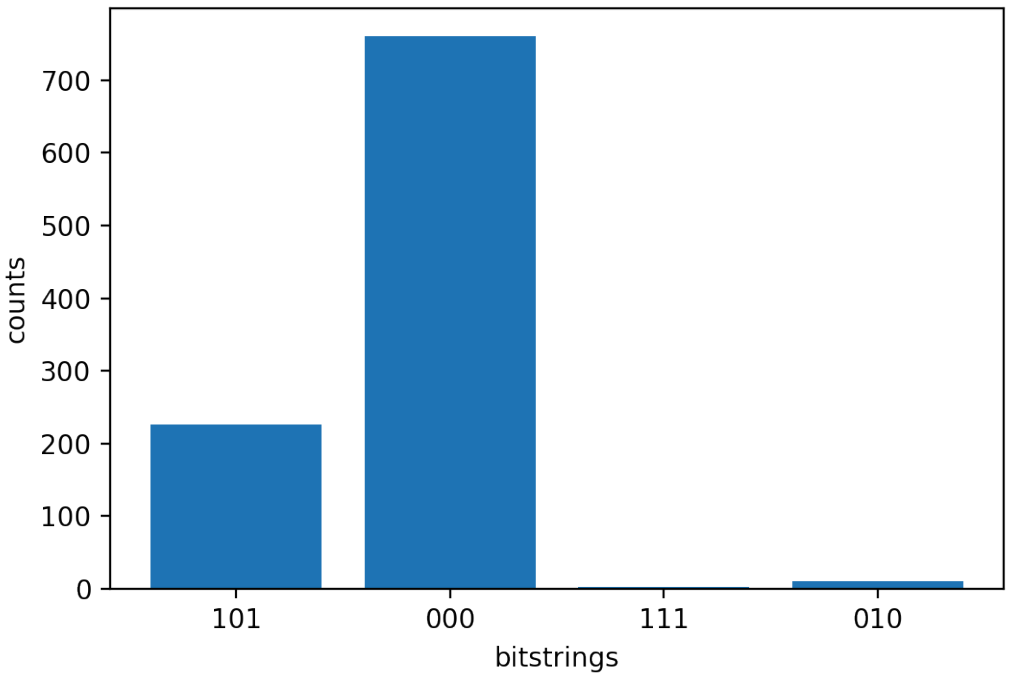
Visualizza i risultati di esempio

Poiché hai anche specificato il`ResultType`, puoi visualizzare i risultati restituiti. I tipi di risultati vengono visualizzati nell'ordine in cui sono stati aggiunti al circuito.

```
print('Result types include:\n', result.result_types)
print('Variance=', result.values[0])
print('Probability=', result.values[1])

# Plot the result and do some analysis
import matplotlib.pyplot as plt
plt.bar(result.measurement_counts.keys(), result.measurement_counts.values())
plt.xlabel('bitstrings')
plt.ylabel('counts')
```

```
Result types include:
```

```
 [ResultTypeValue(type=Variance(observable=['z'], targets=[0], type=<Type.variance:
 'variance'>), value=0.693084), ResultTypeValue(type=Probability(targets=[0, 2],
 type=<Type.probability: 'probability'>), value=array([0.777, 0.   , 0.   , 0.223]))]
Variance= 0.693084
Probability= [0.777 0.    0.    0.223]
Text(0, 0.5, 'counts')
```



Test di un compito quantistico con il simulatore locale

È possibile inviare attività quantistiche direttamente a un simulatore locale per la prototipazione e il test rapidi. Questo simulatore viene eseguito nel tuo ambiente locale, quindi non è necessario specificare una posizione Amazon S3. I risultati vengono calcolati direttamente nella sessione. Per eseguire un'attività quantistica sul simulatore locale, è necessario specificare solo il parametro. shots

> **Note**
>
> La velocità di esecuzione e il numero massimo qubits che il simulatore locale può elaborare dipendono dal tipo di istanza del notebook Amazon Braket o dalle specifiche hardware locali.

I seguenti comandi sono tutti identici e istanziano il simulatore locale State Vector (Noise Free).

```
# Import the LocalSimulator module
from braket.devices import LocalSimulator

# The following are identical commands
device = LocalSimulator()
device = LocalSimulator("default")
device = LocalSimulator(backend="default")
device = LocalSimulator(backend="braket_sv")
```

Quindi esegui un'attività quantistica con quanto segue.

```
my_task = device.run(circ, shots=1000)
```

Per creare un'istanza del simulatore di matrice di densità locale (rumore), i clienti modificano il backend come segue.

```
# Import the LocalSimulator module
from braket.devices import LocalSimulator

device = LocalSimulator(backend="braket_dm")
```

Misurazione di qubit specifici sul simulatore locale

Il simulatore vettoriale statale locale e il simulatore di matrice di densità locale supportano circuiti in esecuzione in cui è possibile misurare un sottoinsieme dei qubit del circuito, operazione spesso denominata misurazione parziale.

Ad esempio, nel codice seguente è possibile creare un circuito a due qubit e misurare solo il primo qubit aggiungendo un'`measure`istruzione con i qubit target alla fine del circuito.

```
# Import the LocalSimulator module
from braket.devices import LocalSimulator

# Use the local simulator device
device = LocalSimulator()

# Define a bell circuit and only measure
circuit = Circuit().h(0).cnot(0, 1).measure(0)

# Run the circuit
task = device.run(circuit, shots=10)
```

```
# Get the results
result = task.result()

# Print the measurement counts for qubit 0
print(result.measurement_counts)
```

Emulatore di dispositivi quantistici locali

Con lo strumento di emulazione locale di Amazon Braket, puoi emulare localmente i tuoi programmi quantistici letterali prima di eseguirli su hardware quantistico reale. L'emulatore utilizza i dati di calibrazione del dispositivo per convalidare i circuiti verbatim, consentendoti di individuare tempestivamente i problemi di compatibilità.

Inoltre, l'emulatore locale simula il rumore hardware quantistico attraverso il seguente processo:

- Utilizzo dei dati di calibrazione del dispositivo per costruire il modello di rumore
- Applicazione di rumore depolarizzante a ciascuna porta del circuito
- Applicazione dell'errore di lettura alla fine del circuito
- Simulazione di un circuito rumoroso con un simulatore di matrice di densità locale

Per ulteriori informazioni sull'utilizzo di un emulatore locale, consulta Emulazione locale per circuiti letterali su Amazon Braket nel repository. amazon-braket-examples GitHub

In questa sezione:

Vantaggi dell'emulazione locale

- Convalida i circuiti testuali in base ai vincoli del dispositivo utilizzando dati di calibrazione storici o in tempo reale.
- Esegui il debug dei problemi prima di inviare attività all'hardware quantistico.
- Confronta le emulazioni silenziose e rumorose con i risultati hardware per comprendere gli effetti del rumore.
- Semplifica il flusso di lavoro per lo sviluppo di algoritmi quantistici sensibili al rumore.

Crea un emulatore locale

Un emulatore di dispositivi quantistici locale può essere creato direttamente da un dispositivo quantistico o da un insieme di proprietà del dispositivo. Quando si emula direttamente un dispositivo, l'emulatore utilizza i dati di calibrazione più recenti del dispositivo istanziato. Il seguente esempio di codice mostra come emulare direttamente il dispositivo. Rigetti's Ankaa-3

```
from braket.aws.aws_device import AwsDevice

ankaa3 = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")
ankaa3_emulator = ankaa3.emulator()
```

L'esempio seguente mostra la creazione di un emulatore di dispositivo locale da un insieme di proprietà del Ankaa-3 dispositivo in formato JSON.

```
from braket.aws import AwsDevice
from braket.emulation.local_emulator import LocalEmulator
import json

# Instantiate the device
ankaa3 = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")
ankaa3_properties = ankaa3.properties

# Put the Ankaa-3 properties in a file named ankaa3_device_properties.json
with open("ankaa3_device_properties.json", "w") as f:
    json.dump(ankaa3_properties.json(), f)

# Load the json into the ankaa3_data_json variable
with open("ankaa3_device_properties.json", "r") as json_file:
    ankaa3_data_json = json.load(json_file)

# Create the Ankaa-3 local emulator from the json file you created
ankaa3_emulator = LocalEmulator.from_json(ankaa3_data_json)
```

È possibile personalizzare l'esempio mediante:

- Utilizzo delle proprietà di un dispositivo QPU diverso
- Specificare un file JSON diverso per l'emulatore
- Modifica dei valori delle proprietà del dispositivo prima di creare un'istanza dell'emulatore

Esegui le tue attività quantistiche con Amazon Braket

Braket fornisce un accesso sicuro e su richiesta a diversi tipi di computer quantistici. Hai accesso ai computer quantistici basati su gate daAQT,, e IonQIQM, nonché a un simulatore Rigetti hamiltoniano analogico di. QuEra Inoltre, non hai alcun impegno anticipato e non devi procurarti l'accesso tramite singoli provider.

- La console Amazon Braket fornisce informazioni e stato del dispositivo per aiutarti a creare, gestire e monitorare le tue risorse e le tue attività quantistiche.
- Invia ed esegui attività quantistiche tramite l'SDK Amazon Braket Python e tramite la console. L'SDK è accessibile tramite notebook Braket preconfigurati. Amazon
- L'API Amazon Braket è accessibile tramite l'SDK e i notebook Amazon Braket Python. Puoi effettuare chiamate direttamente a API se stai creando applicazioni che funzionano con l'informatica quantistica a livello di programmazione.

Gli esempi presenti in questa sezione dimostrano come è possibile lavorare con Amazon Braket API direttamente utilizzando l'SDK Amazon Braket Python insieme all'SDK Python per Braket (AWS Boto3).

Ulteriori informazioni su Amazon Braket Python SDK

Per lavorare con Amazon Braket Python SDK, installa prima l'SDK AWS Python per Braket (Boto3) in modo da poter comunicare con. AWS API Puoi pensare all'SDK Amazon Braket Python come a una comoda soluzione per Boto3 per i clienti quantistici.

- Boto3 contiene interfacce a cui devi attingere. AWS API (Nota che Boto3 è un grande SDK Python che parla con. AWS API La maggior parte Servizi AWS supporta un'interfaccia Boto3.)
- L'SDK Amazon Braket Python contiene moduli software per circuiti, porte, dispositivi, tipi di risultati e altre parti di un'attività quantistica. Ogni volta che crei un programma, importi i moduli necessari per quell'attività quantistica.
- L'SDK Amazon Braket Python è accessibile tramite notebook, che sono precaricati con tutti i moduli e le dipendenze necessari per eseguire attività quantistiche.
- Puoi importare moduli dall'SDK Amazon Braket Python in qualsiasi script Python se non desideri lavorare con i notebook.

Dopo aver installato Boto3, una panoramica dei passaggi per la creazione di un'attività quantistica tramite Braket Amazon Python SDK è simile alla seguente:

1. (Facoltativamente) Apri il tuo notebook.
2. Importa i moduli SDK necessari per i tuoi circuiti.
3. Specificate una QPU o un simulatore.
4. Crea un'istanza del circuito.
5. Avvia il circuito.
6. Raccogli i risultati.

Gli esempi in questa sezione mostrano i dettagli di ogni passaggio.

Per altri esempi, consulta il repository Amazon Braket Examples su. GitHub

In questa sezione:

- Invio di attività quantistiche a QPUs
- Esecuzione di più programmi
- Quando verrà eseguito il mio task quantistico?
- Lavorare con le prenotazioni
- Tecniche di mitigazione degli errori

Invio di attività quantistiche a QPUs

Amazon Braket fornisce l'accesso a diversi dispositivi in grado di eseguire attività quantistiche. Puoi inviare attività quantistiche singolarmente oppure puoi impostare il raggruppamento di attività quantistiche.

Unità di elaborazione quantistica () QPUs

Puoi inviare attività quantistiche a QPUs in qualsiasi momento, ma l'attività viene eseguita entro determinate finestre di disponibilità visualizzate nella pagina Dispositivi della console Amazon Braket. Puoi recuperare i risultati dell'attività quantistica con il quantum task ID, introdotto nella sezione successiva.

- AQT IBEX-Q1 : `arn:aws:braket:eu-north-1::device/qpu/aqt/Ibex-Q1`
- IonQ Forte-1 : `arn:aws:braket:us-east-1::device/qpu/ionq/Forte-1`

- IonQ Forte-Enterprise-1 : `arn:aws:braket:us-east-1::device/qpu/ionq/Forte-Enterprise-1`
- IQM Garnet : `arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet`
- IQM Emerald : `arn:aws:braket:eu-north-1::device/qpu/iqm/Emerald`
- QuEra Aquila : `arn:aws:braket:us-east-1::device/qpu/quera/Aquila`
- Rigetti Ankaa-3 : `arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3`

> **Note**
>
> È possibile annullare le attività quantistiche nei simulatori `CREATED` state for QPUs e on-demand. È possibile annullare le attività quantistiche `QUEUED` nello stato nel miglior modo possibile per i simulatori su richiesta e. QPUs Tieni presente che è improbabile che le attività `QUEUED` quantistiche QPU vengano annullate correttamente durante le finestre di disponibilità della QPU.

In questa sezione:

AQT

AQTIBEX-Q1 QPU si basa su un cristallo di $^{40}Ca^{+}$ ioni in una trappola a radiofrequenza macroscopica situata in una camera ad altissimo vuoto. Il dispositivo funziona a temperatura ambiente e si inserisce in due rack compatibili con data center da 19 pollici.

Le porte ad alta fedeltà sono rese possibili dalle basse velocità di riscaldamento della trappola e dall'uso di una transizione ottica diretta per la rotazione dei qubit. La transizione qubit è guidata da un laser a larghezza di linea stretta con una stabilità di frequenza relativa molto elevata. I qubit sono

inoltre caratterizzati da un'efficiente preparazione e lettura dello stato tramite scaffalature ottiche. All-to-allla connettività è ottenuta mediante l'interazione di Coulomb a lungo raggio nel cristallo ionico. L'indirizzamento e la lettura a ioni singoli si ottengono mediante l'uso di un obiettivo ad alta apertura numerica.

Il AQT dispositivo supporta le seguenti porte quantistiche.

```
'ccnot', 'cnot', 'cphaseshift', 'cphaseshift00', 'cphaseshift01', 'cphaseshift10',
 'cswap', 'swap', 'iswap', 'pswap', 'ecr', 'cy', 'cz', 'xy', 'xx', 'yy', 'zz', 'h',
 'i', 'phaseshift', 'rx', 'ry', 'rz', 's', 'si', 't', 'ti', 'v', 'vi', 'x', 'y', 'z',
 'prx'
```

Con la compilazione letterale, il AQT dispositivo supporta le seguenti porte native.

```
'prx', 'xx', 'rz'
```

> **Note**
>
> Di seguito vengono descritte le porte equivalenti tra porte AQT native e Amazon Braket:
>
> - Il gate AQT Mølmer-Sørensen (MS o RXX) corrisponde al gate di Braket `'xx'`
> - La porta AQT R corrisponde alla porta di `'prx'` Braket
> - Il nome del `'rz'` cancello è lo stesso

IonQ

IonQoffre porte QPUs basate sulla tecnologia delle trappole ioniche. IonQ's QPUs gli ioni intrappolati sono costruiti su una catena di ioni 171Yb+ intrappolati che sono confinati spazialmente mediante una trappola per elettrodi superficiali microfabbricata all'interno di una camera a vuoto.

IonQi dispositivi supportano le seguenti porte quantistiche.

```
'x', 'y', 'z', 'rx', 'ry', 'rz', 'h', 'cnot', 's', 'si', 't', 'ti', 'v', 'vi', 'xx',
 'yy', 'zz', 'swap'
```

Con la compilazione letterale, IonQ QPUs supportano le seguenti porte native.

```
'gpi', 'gpi2', 'ms'
```

Se si specificano solo parametri a due fasi quando si utilizza la porta MS nativa, viene eseguita una porta MS completamente interconnessa. Un gate MS completamente aggrovigliato esegue sempre una rotazione π /2. Per specificare un angolo diverso e far funzionare una porta MS parzialmente aggrovigliata, si specifica l'angolo desiderato aggiungendo un terzo parametro. Per ulteriori informazioni, vedete il modulo braket.circuits.gate.

Queste porte native possono essere utilizzate solo con la compilazione letterale. Per ulteriori informazioni sulla compilazione letterale, consulta Verbatim Compilation.

IQM

IQMi processori quantistici sono dispositivi universali modello gate basati su qubit transmonici superconduttori. È un dispositivo da 20 qubit, IQM Garnet mentre è un dispositivo da 54 qubit. IQM Emerald Entrambi questi dispositivi utilizzano una topologia a reticolo quadrato, nota anche come topologia a reticolo cristallino.

I IQM dispositivi supportano le seguenti porte quantistiche.

```
"ccnot", "cnot", "cphaseshift", "cphaseshift00", "cphaseshift01", "cphaseshift10",
 "cswap", "swap", "iswap", "pswap", "ecr", "cy", "cz", "xy", "xx", "yy", "zz", "h",
 "i", "phaseshift", "rx", "ry", "rz", "s", "si", "t", "ti", "v", "vi", "x", "y", "z"
```

Con la compilazione letterale, IQM i dispositivi supportano le seguenti porte native.

```
'cz', 'prx'
```

Rigetti

Rigettii processori quantistici sono macchine universali modello gate basate su superconduttori completamente sintonizzabili. qubits

- Il Ankaa-3 sistema è un dispositivo da 84 qubit che utilizza una tecnologia multi-chip scalabile.

Il Rigetti dispositivo supporta le seguenti porte quantistiche.

```
'cz', 'xy', 'ccnot', 'cnot', 'cphaseshift', 'cphaseshift00', 'cphaseshift01',
 'cphaseshift10', 'cswap', 'h', 'i', 'iswap', 'phaseshift', 'pswap', 'rx', 'ry', 'rz',
 's', 'si', 'swap', 't', 'ti', 'x', 'y', 'z'
```

Con la compilazione letterale, Ankaa-3 supporta le seguenti porte native.

```
'rx', 'rz', 'iswap'
```

Rigettii processori quantistici superconduttori possono far funzionare la porta 'rx' solo con gli angoli di ±π o ±π.

Il controllo a livello di impulsi è disponibile sui dispositivi Rigetti, che supportano una serie di frame predefiniti dei seguenti tipi per il sistema. Ankaa-3

```
`flux_tx`, `charge_tx`, `readout_rx`, `readout_tx`
```

Il Ankaa-3 dispositivo ha un limite massimo di 20.000 porte per circuito. I circuiti che superano questo limite vengono rifiutati con un errore di convalida. Si tratta di un limite fisso che non può essere aumentato. Il numero di porte si riferisce al circuito compilato, che può differire dal numero di porte del circuito non compilato originale. Per stimare il numero di porte compilate prima dell'invio alla QPU, è possibile utilizzare la compilazione letterale localmente o trasporre il circuito nel set di porte nativo (,,). `rx` `rz` `iswap`

QuEra

QuEraoffre dispositivi basati su atomi neutri in grado di eseguire attività quantistiche di simulazione hamiltoniana analogica (AHS). Questi dispositivi speciali riproducono fedelmente la dinamica quantistica dipendente dal tempo di centinaia di qubit che interagiscono simultaneamente.

È possibile programmare questi dispositivi secondo il paradigma della simulazione hamiltoniana analogica prescrivendo il layout del registro dei qubit e la dipendenza temporale e spaziale dei campi di manipolazione. Amazon Braket fornisce utilità per creare tali programmi tramite il modulo AHS dell'SDK python,. `braket.ahs`

Per ulteriori informazioni, consulta i taccuini di esempio di Analog Hamiltonian Simulation o la pagina Invia un programma analogico utilizzando Aquila. QuEra

Esempio: invio di un'attività quantistica a una QPU

Amazon Braket ti consente di eseguire un circuito quantistico su un dispositivo QPU. L'esempio seguente mostra come inviare un'attività quantistica ai nostri dispositivi. Rigetti IonQ

Scegli il Rigetti Ankaa-3 dispositivo, quindi guarda il grafico di connettività associato

```
# import the QPU module
from braket.aws import AwsDevice
# choose the Rigetti device
device = AwsDevice("arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3")

# take a look at the device connectivity graph
device.properties.dict()['paradigm']['connectivity']
```


```
{'fullyConnected': False,
 'connectivityGraph': {'0': ['1', '7'],
  '1': ['0', '2', '8'],
  '2': ['1', '3', '9'],
  '3': ['2', '4', '10'],
  '4': ['3', '5', '11'],
  '5': ['4', '6', '12'],
  '6': ['5', '13'],
  '7': ['0', '8', '14'],
  '8': ['1', '7', '9', '15'],
  '9': ['2', '8', '10', '16'],
  '10': ['3', '9', '11', '17'],
  '11': ['4', '10', '12', '18'],
  '12': ['5', '11', '13', '19'],
  '13': ['6', '12', '20'],
  '14': ['7', '15', '21'],
  '15': ['8', '14', '22'],
  '16': ['9', '17', '23'],
  '17': ['10', '16', '18', '24'],
  '18': ['11', '17', '19', '25'],
  '19': ['12', '18', '20', '26'],
  '20': ['13', '19', '27'],
  '21': ['14', '22', '28'],
  '22': ['15', '21', '23', '29'],
  '23': ['16', '22', '24', '30'],
  '24': ['17', '23', '25', '31'],
  '25': ['18', '24', '26', '32'],
  '26': ['19', '25', '33'],
  '27': ['20', '34'],
  '28': ['21', '29', '35'],
  '29': ['22', '28', '30', '36'],
  '30': ['23', '29', '31', '37'],
  '31': ['24', '30', '32', '38'],
  '32': ['25', '31', '33', '39'],
  '33': ['26', '32', '34', '40'],
```

```
    '34': ['27', '33', '41'],
    '35': ['28', '36', '42'],
    '36': ['29', '35', '37', '43'],
    '37': ['30', '36', '38', '44'],
    '38': ['31', '37', '39', '45'],
    '39': ['32', '38', '40', '46'],
    '40': ['33', '39', '41', '47'],
    '41': ['34', '40', '48'],
    '42': ['35', '43', '49'],
    '43': ['36', '42', '44', '50'],
    '44': ['37', '43', '45', '51'],
    '45': ['38', '44', '46', '52'],
    '46': ['39', '45', '47', '53'],
    '47': ['40', '46', '48', '54'],
    '48': ['41', '47', '55'],
    '49': ['42', '56'],
    '50': ['43', '51', '57'],
    '51': ['44', '50', '52', '58'],
    '52': ['45', '51', '53', '59'],
    '53': ['46', '52', '54'],
    '54': ['47', '53', '55', '61'],
    '55': ['48', '54', '62'],
    '56': ['49', '57', '63'],
    '57': ['50', '56', '58', '64'],
    '58': ['51', '57', '59', '65'],
    '59': ['52', '58', '60', '66'],
    '60': ['59'],
    '61': ['54', '62', '68'],
    '62': ['55', '61', '69'],
    '63': ['56', '64', '70'],
    '64': ['57', '63', '65', '71'],
    '65': ['58', '64', '66', '72'],
    '66': ['59', '65', '67'],
    '67': ['66', '68'],
    '68': ['61', '67', '69', '75'],
    '69': ['62', '68', '76'],
    '70': ['63', '71', '77'],
    '71': ['64', '70', '72', '78'],
    '72': ['65', '71', '73', '79'],
    '73': ['72', '80'],
    '75': ['68', '76', '82'],
    '76': ['69', '75', '83'],
    '77': ['70', '78'],
    '78': ['71', '77', '79'],
```

```
  '79': ['72', '78', '80'],
  '80': ['73', '79', '81'],
  '81': ['80', '82'],
  '82': ['75', '81', '83'],
  '83': ['76', '82']}}
```

Il dizionario precedente `connectivityGraph` elenca i qubit adiacenti per ogni qubit del dispositivo. Rigetti

Scegli il dispositivo IonQ Forte-Enterprise-1

Per il IonQ Forte-Enterprise-1 dispositivo, `connectivityGraph` è vuoto, come illustrato nell'esempio seguente, perché il dispositivo offre all-to-allconnettività. Pertanto, non `connectivityGraph` è necessario un dettaglio.

```
# or choose the IonQ Forte-Enterprise-1 device
device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Forte-Enterprise-1")

# take a look at the device connectivity graph
device.properties.dict()['paradigm']['connectivity']
```

```
{'fullyConnected': True, 'connectivityGraph': {...}}
```

Come illustrato nell'esempio seguente, hai la possibilità di modificare shots (default=1000), `poll_timeout_seconds` (default = 432000 = 5 giorni), `poll_interval_seconds` (default = 1) e la posizione del bucket S3 (`s3_location`) in cui verranno archiviati i risultati se scegli di specificare una posizione diversa dal bucket predefinito.

```
my_task = device.run(circ, s3_location = 'amazon-braket-my-folder', shots=100,
 poll_timeout_seconds = 100, poll_interval_seconds = 10)
```

Rigettil dispositivi IonQ e compilano automaticamente il circuito fornito nei rispettivi set di porte nativi e mappano qubit gli indici astratti su quelli fisici qubits sulla rispettiva QPU.

> **Note**
>
> I dispositivi QPU hanno una capacità limitata. Quando viene raggiunta la capacità, è possibile aspettarsi tempi di attesa più lunghi.

Amazon Braket può eseguire attività quantistiche QPU entro determinate finestre di disponibilità, ma puoi comunque inviare attività quantistiche in qualsiasi momento (24 ore su 24, 7 giorni su 7) perché tutti i dati e i metadati corrispondenti vengono archiviati in modo affidabile nel bucket S3 appropriato. Come illustrato nella sezione successiva, puoi ripristinare il tuo task quantistico utilizzando e il tuo ID di attività quantistica univoco. `AwsQuantumTask`

Ispezione dei circuiti compilati

Quando un circuito quantistico deve essere eseguito su un dispositivo hardware, come un'unità di elaborazione quantistica (QPU), il circuito deve prima essere compilato in un formato accettabile che il dispositivo possa comprendere ed elaborare. Ad esempio, trasponendo il circuito quantistico di alto livello fino alle porte native specifiche supportate dall'hardware QPU di destinazione. L'ispezione dell'effettivo output compilato del circuito quantistico può essere estremamente utile per scopi di debug e ottimizzazione. Queste conoscenze possono aiutare a identificare potenziali problemi, strozzature o opportunità per migliorare le prestazioni e l'efficienza dell'applicazione quantistica. È possibile visualizzare e analizzare l'output compilato dei circuiti quantistici sia per i dispositivi di elaborazione quantistica che per i dispositivi di elaborazione IQM quantistica Rigetti utilizzando il codice fornito di seguito.

```
task = AwsQuantumTask(arn=task_id, aws_session=session)
# After the task has finished running
task_result = task.result()
compiled_circuit = task_result.get_compiled_circuit()
```

> **Note**
>
> Attualmente, la visualizzazione dell'uscita del circuito compilato per IonQ i dispositivi non è supportata.

Esecuzione di più programmi

Amazon Braket offre due approcci per eseguire più programmi quantistici in modo efficiente: set di programmi e batch di attività quantistiche.

I set di programmi sono il modo preferito per eseguire carichi di lavoro con più programmi. Consentono di raggruppare più programmi in un'unica attività quantistica di Amazon Braket. I set di

programmi offrono miglioramenti delle prestazioni e risparmi sui costi rispetto all'invio di programmi singolarmente, soprattutto quando il numero di esecuzioni di programmi si avvicina a 100.

Attualmente, Rigetti tutti IQM i dispositivi supportano i set di programmi. Prima di inviare i set di programmi a QPUs, si consiglia di eseguire prima il test su Amazon Braket Local Simulator. Per verificare se un dispositivo supporta i set di programmi, puoi visualizzare le proprietà del dispositivo utilizzando Amazon Braket SDK o visualizzare la pagina del dispositivo nella console Amazon Braket.

L'esempio seguente mostra come eseguire un set di programmi.

```
from math import pi
from braket.devices import LocalSimulator
from braket.program_sets import ProgramSet
from braket.circuits import Circuit

program_set = ProgramSet([
    Circuit().h(0).cnot(0,1),
    Circuit().rx(0, pi/4).ry(1, pi/8).cnot(1,0),
    Circuit().t(0).t(1).cz(0,1).s(0).cz(1,2).s(1).s(2),
])

device = LocalSimulator()
result = device.run(program_set, shots=300).result()
print(result[0][0].counts)  # The result of the first program in the program set
```

Per saperne di più sui diversi modi di costruire un set di programmi (ad esempio, costruire un set di programmi da più osservabili o parametri con un singolo programma) e recuperare i risultati dei set di programmi, consulta la sezione Program sets nella Amazon Braket Developer Guide e la cartella program sets nel repository Braket examples Github.

Il quantum task batching è disponibile su tutti i dispositivi Amazon Braket. Il batching è particolarmente utile per le attività quantistiche eseguite sui simulatori on-demand (SV1, DM1 orTN1) perché possono elaborare più attività quantistiche in parallelo. Il batching consente di avviare attività quantistiche in parallelo. Ad esempio, se si desidera eseguire un calcolo che richiede 10 attività quantistiche e i programmi in tali attività quantistiche sono indipendenti l'uno dall'altro, si consiglia di utilizzare il task batching. Utilizzate il quantum task batching quando eseguite carichi di lavoro con più programmi su un dispositivo che non supporta set di programmi.

L'esempio seguente mostra come eseguire un batch di attività quantistiche.

```
from braket.circuits import Circuit
```

```
from braket.devices import LocalSimulator

bell = Circuit().h(0).cnot(0, 1)
circuits = [bell for _ in range(5)]

device = LocalSimulator()
batch = device.run_batch(circuits, shots=100)
print(batch.results()[0].measurement_counts)  # The result of the first quantum task in
 the batch
```

Per informazioni più specifiche sul batching, consulta gli esempi di Amazon Braket su. GitHub

In questa sezione:

- Informazioni sul set e sui costi del programma
- Informazioni sul raggruppamento quantistico delle attività e sui costi
- Quantum task batching e PennyLane
- Task batching e circuiti parametrizzati

Informazioni sul set e sui costi del programma

I set di programmi eseguono in modo efficiente più programmi quantistici raggruppando fino a 100 programmi o set di parametri in un'unica attività quantistica. Con i set di programmi, paghi solo una tariffa per attività più le tariffe per ripresa basate sul totale delle riprese di tutti i programmi, riducendo significativamente i costi rispetto all'invio dei programmi singolarmente. Questo approccio è particolarmente utile per carichi di lavoro con molti programmi e con un numero ridotto di riprese per programma. I set di programmi sono attualmente supportati su Rigetti dispositivi IQM e, oltre che su Amazon Braket Local Simulator.

Per ulteriori informazioni, consulta la sezione Set di programmi per passaggi di implementazione dettagliati, best practice ed esempi di codice.

Informazioni sul raggruppamento quantistico delle attività e sui costi

Alcune avvertenze da tenere a mente per quanto riguarda il raggruppamento quantistico delle attività e i costi di fatturazione:

- Per impostazione predefinita, il quantum task batching riprova ogni volta o fallisce le attività quantistiche 3 volte.

- Un batch di attività quantistiche di lunga durata, ad esempio 34 qubits forSV1, può comportare costi elevati. Assicuratevi di ricontrollare attentamente i valori di `run_batch` assegnazione prima di iniziare una serie di attività quantistiche. Si sconsiglia l'utilizzo conTN1. `run_batch`
- TN1può comportare costi in caso di mancata riuscita delle attività della fase di prova (per ulteriori informazioni, consulta la TN1 descrizione). I nuovi tentativi automatici possono aumentare il costo, quindi consigliamo di impostare il numero di «max_retry» per il batch su 0 quando si utilizza TN1 (vedere Quantum Task Batching, riga 186).

Quantum task batching e PennyLane

Sfrutta i vantaggi del batch quando utilizzi PennyLane Amazon Braket `parallel = True` impostando quando crei un'istanza di un dispositivo Amazon Braket, come illustrato nell'esempio seguente.

```
import pennylane as qml

# Define the number of wires (qubits) you want to use
wires = 2  # For example, using 2 qubits

# Define your S3 bucket
my_bucket = "amazon-braket-s3-demo-bucket"
my_prefix = "pennylane-batch-output"
s3_folder = (my_bucket, my_prefix)

device = qml.device("braket.aws.qubit",
                    device_arn="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
                    wires=wires,
                    s3_destination_folder=s3_folder,
                    parallel=True)
```

Per ulteriori informazioni sul batching with, consulta Ottimizzazione parallelizzata dei circuiti PennyLane quantistici.

Task batching e circuiti parametrizzati

Quando si invia un batch di attività quantistiche che contiene circuiti parametrizzati, è possibile fornire un `inputs` dizionario, che viene utilizzato per tutte le attività quantistiche `list` del batch, o un dizionario di input, nel qual caso il dizionario `i` -th viene abbinato `i` al task -th, come mostrato nell'esempio seguente.

```
from braket.circuits import Circuit, FreeParameter, Observable
from braket.aws import AwsQuantumTaskBatch, AwsDevice

# Define your quantum device
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")

# Create the free parameters
alpha = FreeParameter('alpha')
beta = FreeParameter('beta')

# Create two circuits
circ_a = Circuit().rx(0, alpha).ry(1, alpha).cnot(0, 2).xx(0, 2, beta)
circ_a.variance(observable=Observable.Z(), target=0)

circ_b = Circuit().rx(0, alpha).rz(1, alpha).cnot(0, 2).zz(0, 2, beta)
circ_b.expectation(observable=Observable.Z(), target=2)

# Use the same inputs for both circuits in one batch
tasks = device.run_batch([circ_a, circ_b], inputs={'alpha': 0.1, 'beta': 0.2})

# Or provide each task its own set of inputs
inputs_list = [{'alpha': 0.3, 'beta': 0.1}, {'alpha': 0.1, 'beta': 0.4}]

tasks = device.run_batch([circ_a, circ_b], inputs=inputs_list)
```

È inoltre possibile preparare un elenco di dizionari di input per un singolo circuito parametrico e inviarli come batch di operazioni quantistiche. Se nell'elenco sono presenti N dizionari di input, il batch contiene N task quantistici. Il task quantistico `i` -th corrisponde al circuito eseguito con il dizionario di input -th. `i`

```
from braket.circuits import Circuit, FreeParameter

# Create a parametric circuit
circ = Circuit().rx(0, FreeParameter('alpha'))

# Provide a list of inputs to execute with the circuit
inputs_list = [{'alpha': 0.1}, {'alpha': 0.2}, {'alpha': 0.3}]

tasks = device.run_batch(circ, inputs=inputs_list, shots=100)
```

Quando verrà eseguito il mio task quantistico?

Quando invii un circuito, Amazon Braket lo invia al dispositivo specificato. Le attività quantistiche della Quantum Processing Unit (QPU) e del simulatore su richiesta vengono messe in coda ed elaborate nell'ordine in cui vengono ricevute. Il tempo necessario per elaborare un'attività quantistica dopo l'invio varia a seconda del numero e della complessità delle attività inviate da altri clienti Amazon Braket e della disponibilità della QPU selezionata.

In questa sezione:

Disponibilità, finestre e stato della QPU

La disponibilità delle QPU varia da dispositivo a dispositivo.

Nella pagina Dispositivi della console Amazon Braket, puoi visualizzare le finestre di disponibilità attuali e future e lo stato del dispositivo. Inoltre, ogni pagina del dispositivo mostra le profondità di coda individuali per attività quantistiche e lavori ibridi.

Un dispositivo è considerato offline se non è disponibile per i clienti, indipendentemente dalla finestra di disponibilità. Ad esempio, potrebbe essere offline a causa di manutenzione programmata, aggiornamenti o problemi operativi.

Visibilità della coda

Prima di inviare un'attività quantistica o un lavoro ibrido, puoi visualizzare quante attività quantistiche o ibride hai davanti a te controllando la profondità della coda del dispositivo.

Profondità della coda

Queue depthsi riferisce al numero di attività quantistiche e lavori ibridi in coda per un particolare dispositivo. Le attività quantistiche di un dispositivo e il conteggio delle code di lavoro ibride sono accessibili tramite l'opzione o. Braket Software Development Kit (SDK) Amazon Braket Management Console

1. La profondità della coda delle attività si riferisce al numero totale di attività quantistiche attualmente in attesa di essere eseguite con priorità normale.

2. La profondità della coda delle attività prioritarie si riferisce al numero totale di attività quantistiche inviate in attesa di essere eseguite. Amazon Braket Hybrid Jobs Queste attività vengono eseguite prima delle attività autonome.
3. La profondità della coda dei lavori ibridi si riferisce al numero totale di lavori ibridi attualmente in coda su un dispositivo. Quantum tasksinviati come parte di un lavoro ibrido hanno la priorità e sono aggregati in. Priority Task Queue

I clienti che desiderano visualizzare la profondità della coda tramite il Braket SDK possono modificare il seguente frammento di codice per ottenere la posizione in coda della loro attività quantistica o del loro lavoro ibrido:

```
device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Forte-Enterprise-1")

# returns the number of quantum tasks queued on the device
print(device.queue_depth().quantum_tasks)
{<QueueType.NORMAL: 'Normal'>: '0', <QueueType.PRIORITY: 'Priority'>: '0'}


# returns the number of hybrid jobs queued on the device
print(device.queue_depth().jobs)
'3'
```

L'invio di un'attività quantistica o di un lavoro ibrido a una QPU può comportare lo stato del carico di lavoro. QUEUED Amazon Braket offre ai clienti visibilità sulle attività quantistiche e sulla posizione ibrida nella coda dei lavori.

Posizione in coda

Queue positionsi riferisce alla posizione corrente dell'attività quantistica o del lavoro ibrido all'interno della rispettiva coda del dispositivo. Può essere ottenuto per attività quantistiche o lavori ibridi tramite o. Braket Software Development Kit (SDK) Amazon Braket Management Console

I clienti che desiderano visualizzare la posizione della coda tramite il Braket SDK possono modificare il seguente frammento di codice per ottenere la posizione in coda del loro task quantistico o del loro lavoro ibrido:

```
# choose the device to run your circuit
device = AwsDevice("arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet")
```

```
#execute the circuit
task = device.run(bell, s3_folder, shots=100)

# retrieve the queue position information
print(task.queue_position().queue_position)

# Returns the number of Quantum Tasks queued ahead of you
'2'


from braket.aws import AwsQuantumJob

job = AwsQuantumJob.create(
    "arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet",
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
    wait_until_complete=False
)

# retrieve the queue position information
print(job.queue_position().queue_position)
'3' # returns the number of hybrid jobs queued ahead of you
```

Configurare le notifiche via e-mail o SMS

Amazon Braket invia eventi ad Amazon EventBridge quando la disponibilità di una QPU cambia o quando cambia lo stato dell'attività quantistica. Segui questi passaggi per ricevere notifiche di modifica dello stato del dispositivo e delle attività quantistiche tramite e-mail o messaggio SMS:

1. Crea un argomento Amazon SNS e un abbonamento a e-mail o SMS. La disponibilità di e-mail o SMS dipende dalla tua regione. Per ulteriori informazioni, consulta Guida introduttiva ad Amazon SNS e invio di messaggi SMS.
2. Crea una regola EventBridge che attivi le notifiche sul tuo argomento SNS. Per ulteriori informazioni, consulta Monitoring Amazon Braket with Amazon. EventBridge

(Facoltativo) Configura le notifiche SNS

Puoi configurare le notifiche tramite Amazon Simple Notification Service (SNS) in modo da ricevere un avviso quando l'attività quantistica di Amazon Braket è completata. Le notifiche attive sono utili se prevedi un lungo tempo di attesa, ad esempio quando invii un'attività quantistica di grandi dimensioni

o quando invii un'attività quantistica al di fuori della finestra di disponibilità di un dispositivo. Se non volete attendere il completamento dell'operazione quantistica, potete impostare una notifica SNS.

Un notebook Amazon Braket ti guida attraverso i passaggi di configurazione. Per ulteriori informazioni, consulta gli esempi di Amazon Braket su GitHub e, in particolare, il notebook di esempio per l'impostazione delle notifiche.

Lavorare con le prenotazioni

Le prenotazioni ti danno accesso esclusivo al dispositivo quantistico di tua scelta. Puoi programmare una prenotazione quando preferisci, in modo da sapere esattamente quando inizia e termina l'esecuzione del carico di lavoro. Le prenotazioni sono disponibili con incrementi di 1 ora per tutti i dispositivi Braket e possono essere annullate fino a 48 ore di anticipo, senza costi aggiuntivi. Ti consigliamo di mettere in coda le attività quantistiche e i lavori ibridi per una prenotazione imminente in anticipo, utilizzando l'ARN di Braket Direct Reservation o di inviare carichi di lavoro durante la prenotazione.

Il costo dell'accesso dedicato ai dispositivi si basa sulla durata della prenotazione, indipendentemente dal numero di attività quantistiche e lavori ibridi eseguiti sull'unità di elaborazione quantistica (QPU). Un elenco aggiornato dei computer quantistici disponibili per le prenotazioni è disponibile sulla nostra pagina dei prezzi o tramite la console di gestione Amazon Braket.

> **Note**
>
> In una prenotazione non ci sono limiti per i gateshot. Inoltre, per IonQ i dispositivi, il numero minimo di colpi per le attività di mitigazione degli errori è ridotto a 500 (rispetto ai 2500 per le attività on-demand).

Quando utilizzare una prenotazione

L'utilizzo dell'accesso alle prenotazioni offre la comodità e la prevedibilità di sapere esattamente quando il carico di lavoro quantistico inizia e termina l'esecuzione. Rispetto all'invio di attività e lavori ibridi su richiesta, non è necessario attendere in coda per le attività di altri clienti. Poiché hai accesso esclusivo al dispositivo durante la prenotazione, solo i tuoi carichi di lavoro verranno eseguiti sul dispositivo per l'intera prenotazione.

Ti consigliamo di utilizzare l'accesso su richiesta per la fase di progettazione e prototipazione della ricerca, per consentire un'iterazione rapida ed economica degli algoritmi. Quando siete pronti a

produrre i risultati finali dell'esperimento, prendete in considerazione la possibilità di prenotare un dispositivo in base alle vostre esigenze per assicurarvi di rispettare le scadenze del progetto o della pubblicazione. Ti consigliamo inoltre di utilizzare le prenotazioni quando desideri eseguire le attività in orari specifici, ad esempio quando esegui una demo dal vivo o un workshop su un computer quantistico.

In questa sezione:

Come creare una prenotazione

Per creare una prenotazione, contatta il team di Braket seguendo questi passaggi:

1. Apri la console Amazon Braket.
2. Scegli Braket Direct nel riquadro a sinistra, quindi nella sezione Prenotazioni, scegli Prenota dispositivo.
3. Seleziona il dispositivo che desideri prenotare.
4. Fornisci le tue informazioni di contatto tra cui nome ed e-mail. Assicurati di fornire un indirizzo email valido che controlli regolarmente.
5. Nella sezione Comunicaci il tuo carico di lavoro, fornisci tutti i dettagli sul carico di lavoro da eseguire utilizzando la tua prenotazione. Ad esempio, la durata della prenotazione desiderata, i vincoli pertinenti o la pianificazione desiderata.

Dopo aver inviato il modulo, riceverai un'e-mail dal team di Braket con i passaggi successivi. Una volta confermata la prenotazione, riceverai l'ARN della prenotazione tramite e-mail. È necessario utilizzare l'ARN di prenotazione per creare attività di prenotazione. Le attività create senza l'ARN di prenotazione verranno inviate alla normale coda su richiesta e NON verranno eseguite durante la prenotazione.

> **Note**
>
> La prenotazione è confermata solo dopo aver ricevuto l'ARN di prenotazione.

Le prenotazioni sono disponibili con incrementi minimi di 1 ora e alcuni dispositivi potrebbero avere ulteriori vincoli di durata della prenotazione (inclusa la durata minima e massima della prenotazione). Il team di Braket condivide tutte le informazioni pertinenti con te prima di confermare la prenotazione.

Il team di Braket ti contatterà tramite e-mail per organizzare una sessione di 30 minuti con un esperto di Braket.

Esecuzione di attività quantistiche durante una prenotazione

Dopo aver ottenuto un ARN di prenotazione valido da Crea una prenotazione, puoi creare attività quantistiche da eseguire durante la prenotazione. Le attività quantistiche e i lavori ibridi inviati con un ARN di prenotazione non verranno visualizzati nella coda dei dispositivi. Le attività inviate prima dell'orario di inizio della prenotazione rimarranno nello `QUEUED` stato fino all'inizio della prenotazione.

> **Note**
>
> Le prenotazioni sono AWS specifiche per account e dispositivo. Solo l' AWS account che ha creato la prenotazione può utilizzare l'ARN della prenotazione.
> Durante una prenotazione, è possibile creare sia attività di prenotazione che attività regolari. Per verificare che un'attività quantistica Braket creata sia associata a una prenotazione, controlla il campo «ARN di prenotazione» nella pagina dell'attività quantistica nella console Braket o interroga lo stesso campo nei metadati dell'attività utilizzando l'SDK. Il resto di questa pagina descrive come specificare quali attività sono associate alla prenotazione.

È possibile creare attività quantistiche utilizzando Python SDKs ad esempio Braket,, CUDA-QPennyLaneQiskit, o direttamente con boto3 (Lavorare con Boto3). Per utilizzare le prenotazioni, devi disporre della versione v1.79.0 o successiva di Amazon Braket. Python SDK Puoi eseguire l'aggiornamento all'SDK, al Qiskit provider e PennyLane al plug-in Braket più recenti con il codice seguente.

```
pip install --upgrade amazon-braket-sdk amazon-braket-pennylane-plugin qiskit-braket-provider
```

Esegui attività con il gestore di contesto **`DirectReservation`**

Il modo consigliato per eseguire un'attività all'interno della prenotazione pianificata consiste nell'utilizzare il gestore di `DirectReservation` contesto. Specificando il dispositivo di destinazione e l'ARN di prenotazione, il gestore di contesto assicura che tutte le attività create all'interno dell'istruzione `with` Python vengano eseguite con accesso esclusivo al dispositivo.

Innanzitutto, definisci un circuito quantistico e il dispositivo. Quindi utilizza il contesto di prenotazione ed esegui l'attività. Assicurati che l'intero carico di lavoro venga eseguito all'interno del **`with`** blocco; tutto ciò che non rientra nell'ambito del **`with`** blocco non verrà associato alla tua prenotazione!

```
from braket.aws import AwsDevice, DirectReservation
from braket.circuits import Circuit
from braket.devices import Devices

bell = Circuit().h(0).cnot(0, 1)
device = AwsDevice(Devices.IonQ.ForteEnterprise1)

# run the circuit in a reservation
with DirectReservation(device, reservation_arn="<my_reservation_arn>"):
    task = device.run(bell, shots=100)
```

È possibile creare attività quantistiche in una prenotazione utilizzandoCUDA-Q, e i Qiskit pluginPennyLane, purché il `DirectReservation`on contesto sia attivo durante la creazione di attività quantistiche. Ad esempio, con il Qiskit-Braket provider, è possibile eseguire le attività come segue.

```
from braket.devices import Devices
from braket.aws import DirectReservation
from qiskit import QuantumCircuit
from qiskit_braket_provider import BraketProvider


qc = QuantumCircuit(2)
qc.h(0)
qc.cx(0, 1)

qpu = BraketProvider().get_backend("Forte Enterprise 1")

# run the circuit in a reservation
with DirectReservation(Devices.IonQ.ForteEnterprise1,
 reservation_arn="<my_reservation_arn>"):
```

```
    qpu_task = qpu.run(qc, shots=10)
```

Analogamente, il codice seguente esegue un circuito durante una prenotazione utilizzando il Braket-PennyLane plugin.

```
from braket.devices import Devices
from braket.aws import DirectReservation
import pennylane as qml


dev = qml.device("braket.aws.qubit", device_arn=Devices.IonQ.ForteEnterprise1.value,
 wires=2, shots=10)

@qml.qnode(dev)
def bell_state():
    qml.Hadamard(wires=0)
    qml.CNOT(wires=[0, 1])
    return qml.probs(wires=[0, 1])

# run the circuit in a reservation
with DirectReservation(Devices.IonQ.ForteEnterprise1,
 reservation_arn="<my_reservation_arn>"):
    probs = bell_state()
```

Impostazione manuale del contesto di prenotazione

In alternativa, puoi impostare manualmente il contesto di prenotazione con il seguente codice.

```
# set reservation context
reservation_context = DirectReservation(device,
 reservation_arn="<my_reservation_arn>").start()

# run circuit during reservation
task = device.run(bell, shots=100)
```

Questo è ideale per i notebook Jupyter in cui il contesto può essere eseguito nella prima cella e tutte le attività successive verranno eseguite nella prenotazione.

> **Note**
>
> La cella contenente la *`.start()`* chiamata deve essere eseguita solo una volta.

Per tornare alla modalità su richiesta: riavvia il notebook Jupyter o chiama quanto segue per riportare il contesto alla modalità su richiesta.

```
reservation_context.stop()  # unset reservation context
```

> **Note**
>
> Le prenotazioni hanno un orario di inizio e di fine predeterminati (vedi Creare una prenotazione). I `reservation_context.stop()` metodi `reservation_context.start()` and non iniziano o terminano una prenotazione. Invece, mentre il contesto è attivo, tutte le attività quantistiche create verranno associate alla prenotazione e verranno eseguite solo durante la prenotazione pianificata. Il contesto della prenotazione non ha alcun effetto sull'orario di prenotazione pianificato.

Passa esplicitamente l'ARN di prenotazione durante la creazione dell'attività

Un altro modo per creare attività durante una prenotazione consiste nel trasmettere esplicitamente l'ARN della prenotazione durante la chiamata. `device.run()`

```
task = device.run(bell, shots=100, reservation_arn="<my_reservation_arn>")
```

Questo metodo associa direttamente l'attività quantistica all'ARN di prenotazione, assicurando che venga eseguita durante il periodo riservato. Per questa opzione, aggiungi l'ARN di prenotazione a ogni attività che intendi eseguire durante una prenotazione. Tuttavia, tieni presente che quando utilizzi librerie di terze parti come Qiskit orPennyLane, può essere difficile garantire che le attività inviate utilizzino l'ARN di prenotazione corretto. Per questo motivo, si consiglia di utilizzare il gestore di DirectReservation contesto.

Quando usi direttamente boto3, passa l'ARN di prenotazione come associazione durante la creazione di un'attività.

```
import boto3

braket_client = boto3.client("braket")


kwargs["associations"] = [
```

```
    {
        "arn": "<my_reservation_arn>",
        "type": "RESERVATION_TIME_WINDOW_ARN",
    }
]

response = braket_client.create_quantum_task(**kwargs)
```

Esecuzione di lavori ibridi durante una prenotazione

Una volta che hai una funzione Python da eseguire come lavoro ibrido, puoi eseguire il lavoro ibrido in una prenotazione passando l'argomento della `reservation_arn` parola chiave. Tutte le attività all'interno del lavoro ibrido utilizzano l'ARN di prenotazione. È importante sottolineare che il job ibrido attiva l'elaborazione classica *`reservation_arn`* solo dopo l'inizio della prenotazione.

> **Note**
>
> Un processo ibrido in esecuzione durante una prenotazione esegue correttamente solo attività quantistiche sul dispositivo riservato. Il tentativo di utilizzare un dispositivo Braket on-demand diverso genererà un errore. Se devi eseguire attività sia su un simulatore su richiesta che sul dispositivo riservato all'interno dello stesso lavoro ibrido, utilizza invece. `DirectReservation`

Il codice seguente mostra come eseguire un processo ibrido durante una prenotazione.

```
from braket.aws import AwsDevice
from braket.devices import Devices
from braket.jobs import get_job_device_arn, hybrid_job

@hybrid_job(device=Devices.IonQ.ForteEnterprise1,
 reservation_arn="<my_reservation_arn>")
def example_hybrid_job():
    # declare AwsDevice within the hybrid job
    device = AwsDevice(get_job_device_arn())
    bell = Circuit().h(0).cnot(0, 1)

    task = device.run(bell, shots=10)
```

Per i lavori ibridi che utilizzano uno script Python (vedi la sezione sulla creazione del primo lavoro ibrido nella guida per gli sviluppatori), puoi eseguirli all'interno della prenotazione passando l'argomento della `reservation_arn` parola chiave durante la creazione del lavoro.

```
from braket.aws import AwsQuantumJob
from braket.devices import Devices

job = AwsQuantumJob.create(
    Devices.IonQ.ForteEnterprise1,
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
    reservation_arn="<my_reservation_arn>"
)
```

Cosa succede alla fine della prenotazione

Al termine della prenotazione, non avrai più un accesso dedicato al dispositivo. Tutti i carichi di lavoro rimanenti in coda con questa prenotazione vengono automaticamente annullati.

> **Note**
>
> Qualsiasi lavoro che era in corso al termine della `RUNNING` prenotazione viene annullato. Ti consigliamo di utilizzare i checkpoint per salvare e riavviare i lavori a tuo piacimento.

Una prenotazione in corso, ad esempio dopo l'inizio della prenotazione e prima della fine della prenotazione, non può essere estesa perché ogni prenotazione rappresenta l'accesso indipendente a un dispositivo dedicato. Ad esempio, due back-to-back prenotazioni sono considerate separate e tutte le attività in sospeso dalla prima prenotazione vengono automaticamente annullate. Non riprendono nella seconda prenotazione.

> **Note**
>
> Le prenotazioni rappresentano l'accesso al dispositivo dedicato al tuo AWS account. Anche se il dispositivo rimane inattivo, nessun altro cliente può utilizzarlo. Pertanto, ti verrà addebitato il costo per la durata del tempo prenotato, indipendentemente dal tempo utilizzato.

Annulla o riprogramma una prenotazione esistente

Puoi cancellare la tua prenotazione non meno di 48 ore prima dell'orario di inizio previsto per la prenotazione. Per annullare, rispondi all'e-mail di conferma della prenotazione che hai ricevuto con la richiesta di cancellazione.

Per riprogrammare, devi cancellare la prenotazione esistente e quindi crearne una nuova.

Tecniche di mitigazione degli errori

La mitigazione quantistica degli errori è un insieme di tecniche volte a ridurre gli effetti degli errori nei computer quantistici.

I dispositivi quantistici sono soggetti a rumori ambientali che riducono la qualità dei calcoli eseguiti. Sebbene il calcolo quantistico tollerante ai guasti prometta una soluzione a questo problema, gli attuali dispositivi quantistici sono limitati dal numero di qubit e dai tassi di errore relativamente elevati. Per ovviare a questo problema nel breve termine, i ricercatori stanno studiando metodi per migliorare l'accuratezza dei rumorosi calcoli quantistici. Questo approccio, noto come mitigazione degli errori quantistici, prevede l'utilizzo di varie tecniche per estrarre il segnale migliore da dati di misurazione rumorosi.

In questa sezione:

- Tecniche di mitigazione degli errori sui dispositivi IonQ

Tecniche di mitigazione degli errori sui dispositivi IonQ

La mitigazione degli errori implica l'esecuzione di più circuiti fisici e la combinazione delle relative misurazioni per ottenere un risultato migliore.

> **Note**
>
> Per tutti i IonQ dispositivi: quando si utilizza un modello on demand, è previsto un limite di 1 milione di gateshot e un minimo di 2500 scatti per le attività di mitigazione degli errori. Per una prenotazione diretta, non è previsto un limite di gateshot e un minimo di 500 shot per le attività di mitigazione degli errori.

Debiasing

IonQi dispositivi dispongono di un metodo di mitigazione degli errori chiamato debiasing.

Debiasing mappa un circuito in più varianti che agiscono su diverse permutazioni di qubit o con diverse decomposizioni dei gate. Ciò riduce l'effetto di errori sistematici come le sovratratazioni delle porte o un singolo qubit difettoso, utilizzando diverse implementazioni di un circuito che altrimenti potrebbero alterare i risultati di misurazione. Ciò comporta costi aggiuntivi per la calibrazione di più qubit e gate.

Per ulteriori informazioni sul debiasing, vedere Miglioramento delle prestazioni dei computer quantistici attraverso la simmetrizzazione.

> **Note**
>
> L'uso del debiasing richiede un minimo di 2500 scatti.

È possibile eseguire un'operazione quantistica con debiasing su un IonQ dispositivo utilizzando il codice seguente:

```
from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.error_mitigation import Debias

# choose an IonQ device
device = AwsDevice("arn:aws:braket:us-east-1::device/qpu/ionq/Forte-Enterprise-1")
circuit = Circuit().h(0).cnot(0, 1)

task = device.run(circuit, shots=2500, device_parameters={"errorMitigation": Debias()})

result = task.result()
print(result.measurement_counts)
>>> {"00": 1245, "01": 5, "10": 10 "11": 1240} # result from debiasing
```

Una volta completata l'operazione quantistica, è possibile visualizzare le probabilità di misurazione e tutti i tipi di risultato dell'operazione quantistica. Le probabilità di misurazione e i conteggi di tutte le varianti vengono aggregati in un'unica distribuzione. Tutti i tipi di risultato specificati nel circuito, ad esempio i valori di aspettativa, vengono calcolati utilizzando i conteggi delle misurazioni aggregate.

Filtro di nitidezza

È inoltre possibile accedere alle probabilità di misurazione calcolate con una diversa strategia di post-elaborazione chiamata nitidezza. La nitidezza confronta i risultati di ciascuna variante ed elimina le immagini incoerenti, favorendo il risultato di misurazione più probabile tra le varianti. Per ulteriori informazioni, vedete Migliorare le prestazioni dei computer quantistici attraverso la simmetrizzazione.

È importante sottolineare che la nitidezza presuppone che la forma della distribuzione di output sia scarsa, con pochi stati ad alta probabilità e molti stati a probabilità zero. Se questa ipotesi non è valida, può distorcere la distribuzione delle probabilità.

Puoi accedere alle probabilità da una distribuzione più nitida nel `additional_metadata` campo dell'SDK Braket `GateModelTaskResult` Python. Nota che la nitidezza non restituisce i conteggi delle misurazioni, ma restituisce invece una distribuzione di probabilità rinormalizzata. Il seguente frammento di codice mostra come accedere alla distribuzione dopo la nitidezza.

```
print(result.additional_metadata.ionqMetadata.sharpenedProbabilities)
>>> {"00": 0.51, "11": 0.549} # sharpened probabilities
```

Lavorare con Amazon Braket Hybrid Jobs

Amazon Braket Hybrid Jobs ti offre un modo per eseguire algoritmi ibridi quantistici classici che richiedono sia AWS risorse classiche che unità di elaborazione quantistica (). QPUs Hybrid Jobs è progettato per attivare le risorse classiche richieste, eseguire l'algoritmo e rilasciare le istanze dopo il completamento, in modo da pagare solo per ciò che usi.

Hybrid Jobs è ideale per algoritmi iterativi di lunga durata che prevedono l'uso sia di risorse di calcolo classiche che di risorse di calcolo quantistico. Con Hybrid Jobs, dopo aver inviato l'algoritmo per l'esecuzione, Braket lo eseguirà in un ambiente scalabile e containerizzato. Una volta completato l'algoritmo, puoi recuperare i risultati.

Inoltre, le attività quantistiche create da un lavoro ibrido traggono vantaggio dall'accodamento con priorità più elevata verso il dispositivo QPU di destinazione. Questa prioritizzazione garantisce che i calcoli quantistici vengano elaborati ed eseguiti prima delle altre attività in attesa in coda. Ciò è particolarmente vantaggioso per gli algoritmi ibridi iterativi, in cui i risultati di un'attività quantistica dipendono dai risultati di attività quantistiche precedenti. Esempi di tali algoritmi includono il Quantum Approximate Optimization Algorithm (QAOA), l'autosolver quantistico variazionale o l'apprendimento automatico quantistico. È inoltre possibile monitorare l'avanzamento dell'algoritmo quasi in tempo reale, in modo da tenere traccia dei costi, del budget o di metriche personalizzate come la perdita di allenamento o i valori di aspettativa.

Puoi accedere ai lavori ibridi in Braket utilizzando:

- L'SDK Amazon Braket Python.
- La console Amazon Braket.
- Amazon BraketAPI.

In questa sezione:

- Personalizzazione del tuo Hybrid Job
- Utilizzo PennyLane con Amazon Braket
- Utilizzo di CUDA-Q con Amazon Braket

Quando usare Amazon Braket Hybrid Jobs

Amazon Braket Hybrid Jobs ti consente di eseguire algoritmi ibridi quantistici classici, come Variational Quantum Eigensolver (VQE) e Quantum Approximate Optimization Algorithm (QAOA), che combinano risorse di calcolo classiche con dispositivi di calcolo quantistico per ottimizzare le prestazioni dei sistemi quantistici odierni. Amazon Braket Hybrid Jobs offre tre vantaggi principali:

1. Prestazioni: Amazon Braket Hybrid Jobs offre prestazioni migliori rispetto all'esecuzione di algoritmi ibridi dal tuo ambiente. Mentre il processo è in esecuzione, ha accesso prioritario alla QPU di destinazione selezionata. Le attività del tuo job vengono eseguite prima delle altre attività in coda sul dispositivo. Ciò si traduce in tempi di esecuzione più brevi e prevedibili per gli algoritmi ibridi. Amazon Braket Hybrid Jobs supporta anche la compilazione parametrica. Puoi inviare un circuito utilizzando parametri liberi e Braket lo compila una sola volta, senza la necessità di ricompilarlo per i successivi aggiornamenti dei parametri sullo stesso circuito, con tempi di esecuzione ancora più rapidi.
2. Convenienza: Amazon Braket Hybrid Jobs semplifica la configurazione e la gestione dell'ambiente di calcolo e ne semplifica l'esecuzione durante l'esecuzione dell'algoritmo ibrido. Devi solo fornire lo script dell'algoritmo e selezionare un dispositivo quantistico (un'unità di elaborazione quantistica o un simulatore) su cui eseguire. Amazon Braket attende che il dispositivo di destinazione diventi disponibile, attiva le risorse classiche, esegue il carico di lavoro in ambienti container predefiniti, restituisce i risultati ad Amazon Simple Storage Service (Amazon S3) e rilascia le risorse di elaborazione.
3. Metriche: Amazon Braket Hybrid Jobs on-the-fly fornisce approfondimenti sugli algoritmi in esecuzione e fornisce metriche degli algoritmi personalizzabili quasi in tempo reale ad Amazon e alla console Amazon Braket in modo da poter monitorare CloudWatch l'avanzamento dei tuoi algoritmi.

Esecuzione di un processo ibrido con Amazon Braket Hybrid Jobs

Per eseguire un processo ibrido con Amazon Braket Hybrid Jobs, devi prima definire il tuo algoritmo. Puoi definirlo scrivendo lo script dell'algoritmo e, facoltativamente, altri file di dipendenza utilizzando

Amazon Braket Python SDK o. PennyLane Se desideri utilizzare altre librerie (open source o proprietarie), puoi definire un'immagine del contenitore personalizzata utilizzando Docker, che include queste librerie. Per ulteriori informazioni, consulta Bring your own container (BYOC).

In entrambi i casi, successivamente crei un lavoro ibrido utilizzando Amazon BraketAPI, dove fornisci lo script o il contenitore dell'algoritmo, seleziona il dispositivo quantistico di destinazione che il lavoro ibrido deve utilizzare e quindi scegli tra una serie di impostazioni opzionali. I valori predefiniti forniti per queste impostazioni opzionali funzionano per la maggior parte dei casi d'uso. Affinché il dispositivo di destinazione esegua il tuo Hybrid Job, puoi scegliere tra una QPU, un simulatore on-demand (comeSV1, DM1 oTN1) o la classica istanza di job ibrida stessa. Con un simulatore o una QPU on-demand, il tuo contenitore di job ibrido effettua chiamate API verso un dispositivo remoto. Con i simulatori incorporati, il simulatore è incorporato nello stesso contenitore dello script dell'algoritmo. I simulatori di fulmini di PennyLane sono integrati nel contenitore di job ibrido predefinito e preconfigurato che puoi utilizzare. Se esegui il codice utilizzando un PennyLane simulatore incorporato o un simulatore personalizzato, puoi specificare un tipo di istanza e quante istanze desideri utilizzare. Consulta la pagina dei prezzi di Amazon Braket per i costi associati a ciascuna scelta.

Se il dispositivo di destinazione è un simulatore on-demand o integrato, Amazon Braket inizia subito a eseguire il processo ibrido. Avvia l'istanza del processo ibrido (puoi personalizzare il tipo di istanza nella API chiamata), esegue l'algoritmo, scrive i risultati su Amazon S3 e rilascia le tue risorse. Questa versione di risorse garantisce che paghi solo per ciò che usi.

Il numero totale di lavori ibridi simultanei per unità di elaborazione quantistica (QPU) è limitato. Oggi, su una QPU può essere eseguito solo un processo ibrido alla volta. Le code vengono utilizzate per

controllare il numero di processi ibridi che possono essere eseguiti in modo da non superare il limite consentito. Se il dispositivo di destinazione è una QPU, il processo ibrido entra prima nella coda dei lavori della QPU selezionata. Amazon Braket attiva l'istanza di lavoro ibrida necessaria ed esegue il processo ibrido sul dispositivo. Per tutta la durata dell'algoritmo, il processo ibrido ha accesso prioritario, il che significa che le attività quantistiche del lavoro ibrido vengono eseguite prima delle altre attività quantistiche di Braket in coda sul dispositivo, a condizione che le attività quantistiche del lavoro vengano inviate alla QPU una volta ogni pochi minuti. Una volta completato il lavoro ibrido, vengono rilasciate risorse, il che significa che paghi solo per ciò che utilizzi.

> **Note**
>
> I dispositivi sono regionali e il processo ibrido viene eseguito sullo Regione AWS stesso dispositivo principale.

Sia nello scenario target del simulatore che in quello della QPU, hai la possibilità di definire metriche personalizzate dell'algoritmo, come l'energia della tua Hamiltoniana, come parte dell'algoritmo. Queste metriche vengono segnalate automaticamente ad Amazon CloudWatch e da lì vengono visualizzate quasi in tempo reale nella console Amazon Braket.

> **Note**
>
> Se desideri utilizzare un'istanza basata su GPU, assicurati di utilizzare uno dei simulatori basati su GPU disponibili con i simulatori integrati su Braket (ad esempio,). `lightning.gpu` Se scegli uno dei simulatori integrati basati su CPU (ad esempio, o`braket:default-simulator`)`lightning.qubit`, la GPU non verrà utilizzata e potresti incorrere in costi inutili.

Concetti chiave per Hybrid Jobs

Questa sezione spiega i concetti chiave della `AwsQuantumJob.create` funzione fornita da Amazon Braket Python SDK e la mappatura alla struttura del file contenitore.

Oltre al file o ai file che compongono lo script completo dell'algoritmo, il lavoro ibrido può avere input e output aggiuntivi. All'avvio del processo ibrido, Amazon Braket copia gli input forniti come parte della creazione del lavoro ibrido nel contenitore che esegue lo script dell'algoritmo. Al termine del

processo ibrido, tutti gli output definiti durante l'algoritmo vengono copiati nella posizione Amazon S3 specificata.

> **Note**
>
> Le metriche dell'algoritmo vengono riportate in tempo reale e non seguono questa procedura di output.

Amazon Braket fornisce anche diverse variabili di ambiente e funzioni di supporto per semplificare le interazioni con gli input e gli output dei container. Per ulteriori informazioni, consulta il pacchetto braket.jobs nell'SDK Amazon Braket.

In questa sezione:

- Input
- Output
- Variabili di ambiente
- Funzioni di supporto

Input

Dati di input: i dati di input possono essere forniti all'algoritmo ibrido specificando il file di dati di input, che è impostato come dizionario, con l'`input_data`argomento. L'utente definisce l'`input_data`argomento all'interno della `AwsQuantumJob.create` funzione nell'SDK. Questo copia i dati di input nel file system del contenitore nella posizione indicata dalla variabile `"AMZN_BRAKET_INPUT_DIR"` di ambiente. Per un paio di esempi di come i dati di input vengono utilizzati in un algoritmo ibrido, consulta QAOA con Amazon Braket Hybrid Jobs PennyLane e Quantum machine learning nei notebook Amazon Braket Hybrid Jobs Jobs Jupyter.

> **Note**
>
> Quando i dati di input sono di grandi dimensioni (> 1 GB), ci sarà un lungo tempo di attesa prima che il lavoro ibrido venga inviato. Ciò è dovuto al fatto che i dati di input locali verranno prima caricati su un bucket S3, quindi il percorso S3 verrà aggiunto alla richiesta di lavoro ibrida e, infine, la richiesta di lavoro ibrida verrà inviata al servizio Braket.

Iperparametri: se si passa`hyperparameters`, sono disponibili nella variabile di ambiente. `"AMZN_BRAKET_HP_FILE"`

> **Note**
>
> Per ulteriori informazioni su come creare iperparametri e dati di input e quindi passare queste informazioni allo script di lavoro ibrido, consulta la sezione Use hyperparameters e questa pagina github.

Punti di controllo: per specificare `job-arn` il checkpoint di cui desideri utilizzare in un nuovo lavoro ibrido, usa il comando. `copy_checkpoints_from_job` Questo comando copia i dati del checkpoint nel nuovo processo ibrido, rendendoli disponibili nel percorso indicato dalla variabile di ambiente `AMZN_BRAKET_CHECKPOINT_DIR` durante l'esecuzione del lavoro. `checkpoint_configs3Uri` L'impostazione predefinita è `None` che i dati del checkpoint di un altro lavoro ibrido non verranno utilizzati nel nuovo lavoro ibrido.

Output

Attività quantistiche: i risultati delle attività quantistiche vengono archiviati nella posizione S3. `s3://amazon-braket-<region>-<accountID>/jobs/<job-name>/tasks`

Risultati del lavoro: tutto ciò che lo script dell'algoritmo salva nella directory fornita dalla variabile di ambiente `"AMZN_BRAKET_JOB_RESULTS_DIR"` viene copiato nella posizione S3 specificata in. `output_data_config` Se il valore non è specificato, il valore predefinito è. `s3://amazon-braket-<region>-<accountID>/jobs/<job-name>/<timestamp>/data` Forniamo la funzione di supporto SDK **`save_job_result`**, che puoi utilizzare per archiviare comodamente i risultati sotto forma di dizionario quando richiami dallo script dell'algoritmo.

Punti di controllo: se desideri utilizzare i checkpoint, puoi salvarli nella directory fornita dalla variabile di ambiente. `"AMZN_BRAKET_CHECKPOINT_DIR"` Puoi invece utilizzare anche la funzione di supporto SDK. `save_job_checkpoint`

Metriche dell'algoritmo: puoi definire le metriche dell'algoritmo come parte dello script dell'algoritmo che vengono emesse su Amazon CloudWatch e visualizzate in tempo reale nella console Amazon Braket mentre il processo ibrido è in esecuzione. Per un esempio di come utilizzare le metriche degli algoritmi, consulta Use Amazon Braket Hybrid Jobs per eseguire un algoritmo QAOA.

Per ulteriori informazioni sul salvataggio dei risultati dei lavori, consulta Salvare i risultati nella documentazione di Hybrid Jobs.

Variabili di ambiente

Amazon Braket fornisce diverse variabili di ambiente per semplificare le interazioni con gli input e gli output dei container. Il codice seguente elenca le variabili ambientali utilizzate da Braket.

- `AMZN_BRAKET_INPUT_DIR`— La directory dei dati di input. opt/braket/input/data
- `AMZN_BRAKET_JOB_RESULTS_DIR`— La directory di output opt/braket/model in cui scrivere i risultati del lavoro.
- `AMZN_BRAKET_JOB_NAME`— Il nome del lavoro.
- `AMZN_BRAKET_CHECKPOINT_DIR`— L'elenco dei checkpoint.
- `AMZN_BRAKET_HP_FILE`— Il file contenente gli iperparametri.
- `AMZN_BRAKET_DEVICE_ARN`— L'ARN (AWS Resource Name) del dispositivo.
- `AMZN_BRAKET_OUT_S3_BUCKET`— Il bucket Amazon S3 di output, come specificato nella `CreateJob` richiesta. `OutputDataConfig`
- `AMZN_BRAKET_SCRIPT_ENTRY_POINT`— Il punto di ingresso specificato nella `CreateJob` richiesta. `ScriptModeConfig`
- `AMZN_BRAKET_SCRIPT_COMPRESSION_TYPE`— Il tipo di compressione specificato nella `CreateJob` richiesta`ScriptModeConfig`.
- `AMZN_BRAKET_SCRIPT_S3_URI`— La posizione Amazon S3 dello script dell'utente come specificato nella `CreateJob` richiesta. `ScriptModeConfig`
- `AMZN_BRAKET_TASK_RESULTS_S3_URI`— La posizione Amazon S3 in cui l'SDK archivia i risultati quantistici delle attività per impostazione predefinita per il lavoro.
- `AMZN_BRAKET_JOB_RESULTS_S3_PATH`— La sede Amazon S3 in cui verranno archiviati i risultati del lavoro, come specificato nella `CreateJob` richiesta. `OutputDataConfig`
- `AMZN_BRAKET_JOB_TOKEN`— La stringa che deve essere passata al `CreateQuantumTask` `jobToken` parametro per le attività quantistiche create nel job container.

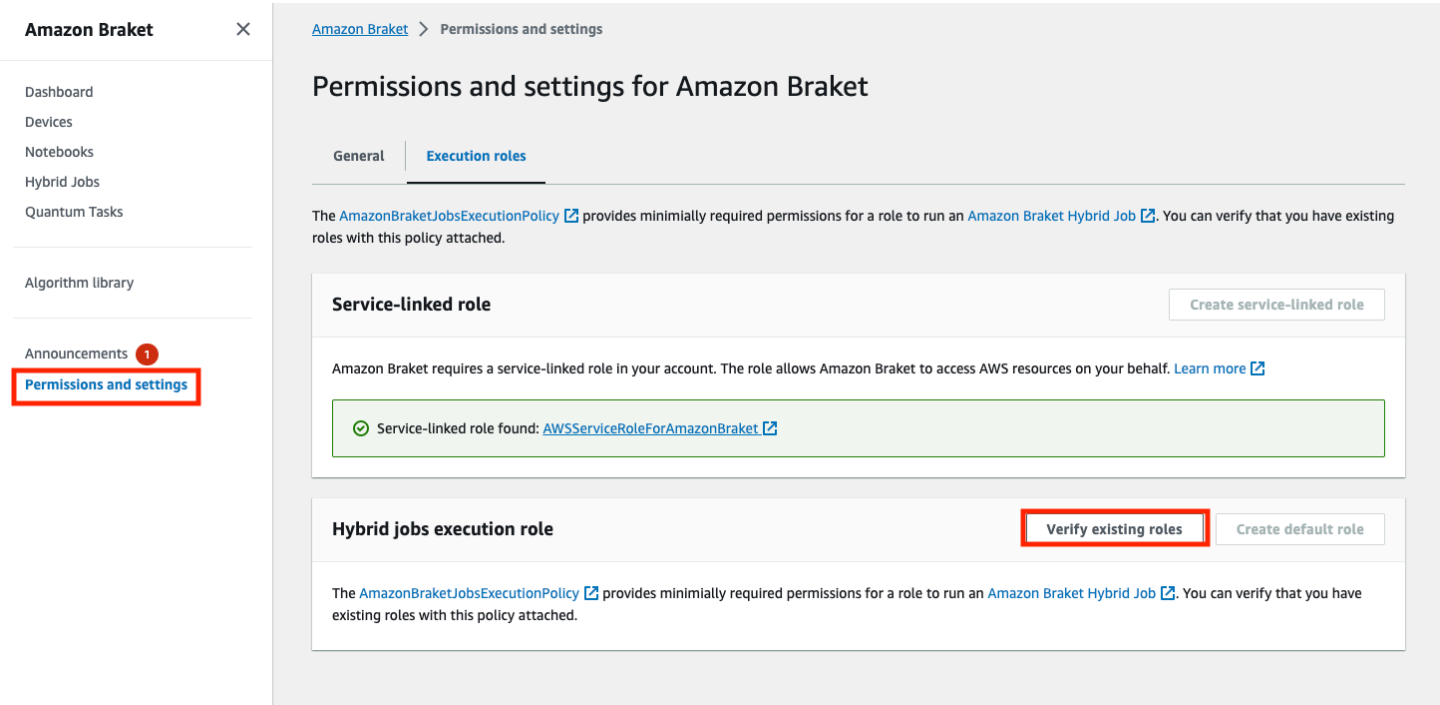
Funzioni di supporto

Amazon Braket offre diverse funzioni di supporto per semplificare le interazioni con gli input e gli output dei container. Queste funzioni di supporto verrebbero richiamate dall'interno dello script dell'algoritmo utilizzato per eseguire Hybrid Job. L'esempio seguente mostra come utilizzarle.

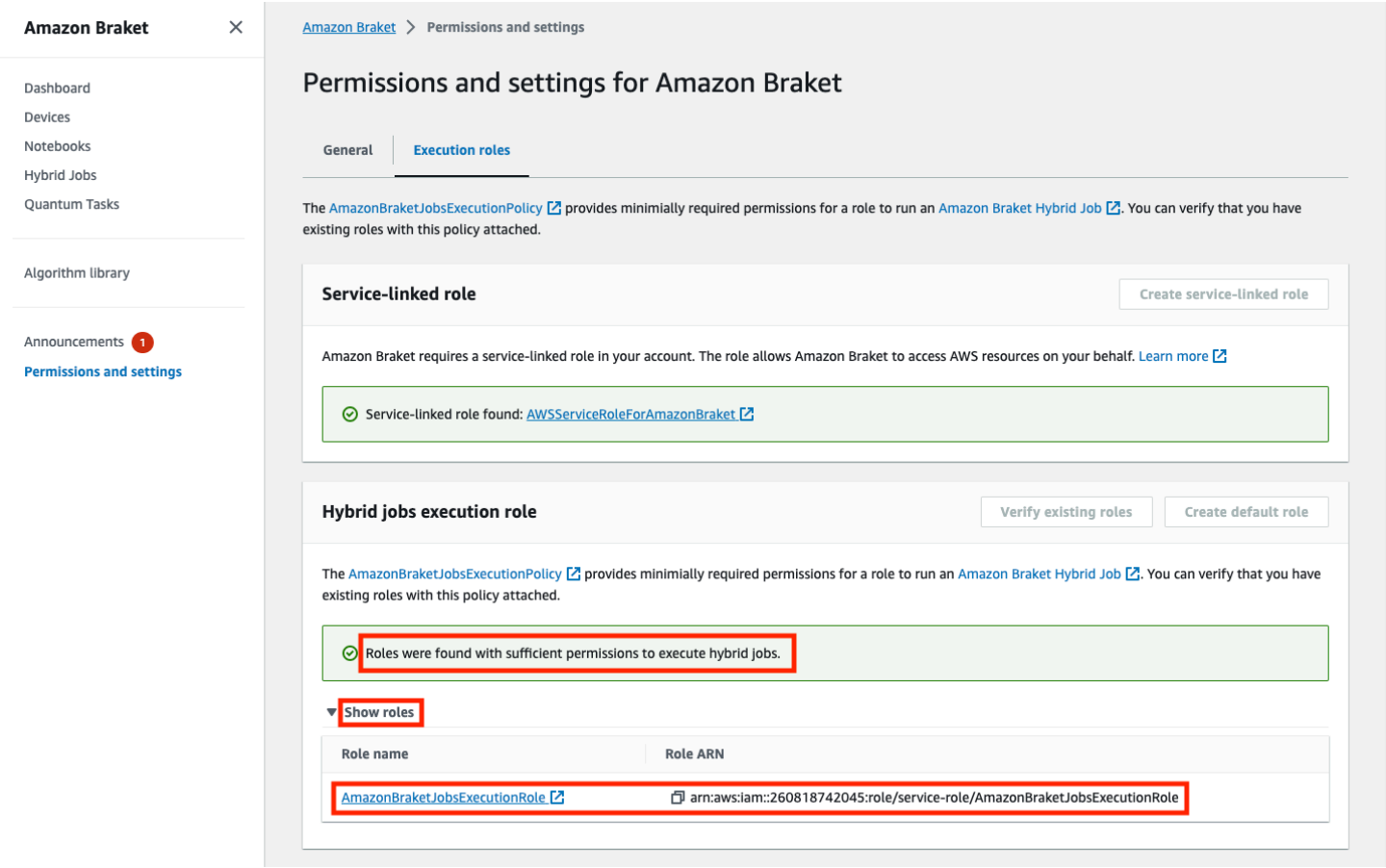
```
from braket.jobs import get_checkpoint_dir, get_hyperparameters, get_input_data_dir,
 get_job_device_arn, get_job_name, get_results_dir, save_job_result,
 save_job_checkpoint, load_job_checkpoint

get_checkpoint_dir() # Get the checkpoint directory
get_hyperparameters() # Get the hyperparameters as strings
get_input_data_dir() # Get the input data directory
get_job_device_arn() # Get the device specified by the hybrid job
get_job_name() # Get the name of the hybrid job.
get_results_dir() # Get the path to a results directory
save_job_result(result_data='data') # Save hybrid job results
save_job_checkpoint(checkpoint_data={'key': 'value'}) # Save a checkpoint
load_job_checkpoint() # Load a previously saved checkpoint
```

Prerequisiti

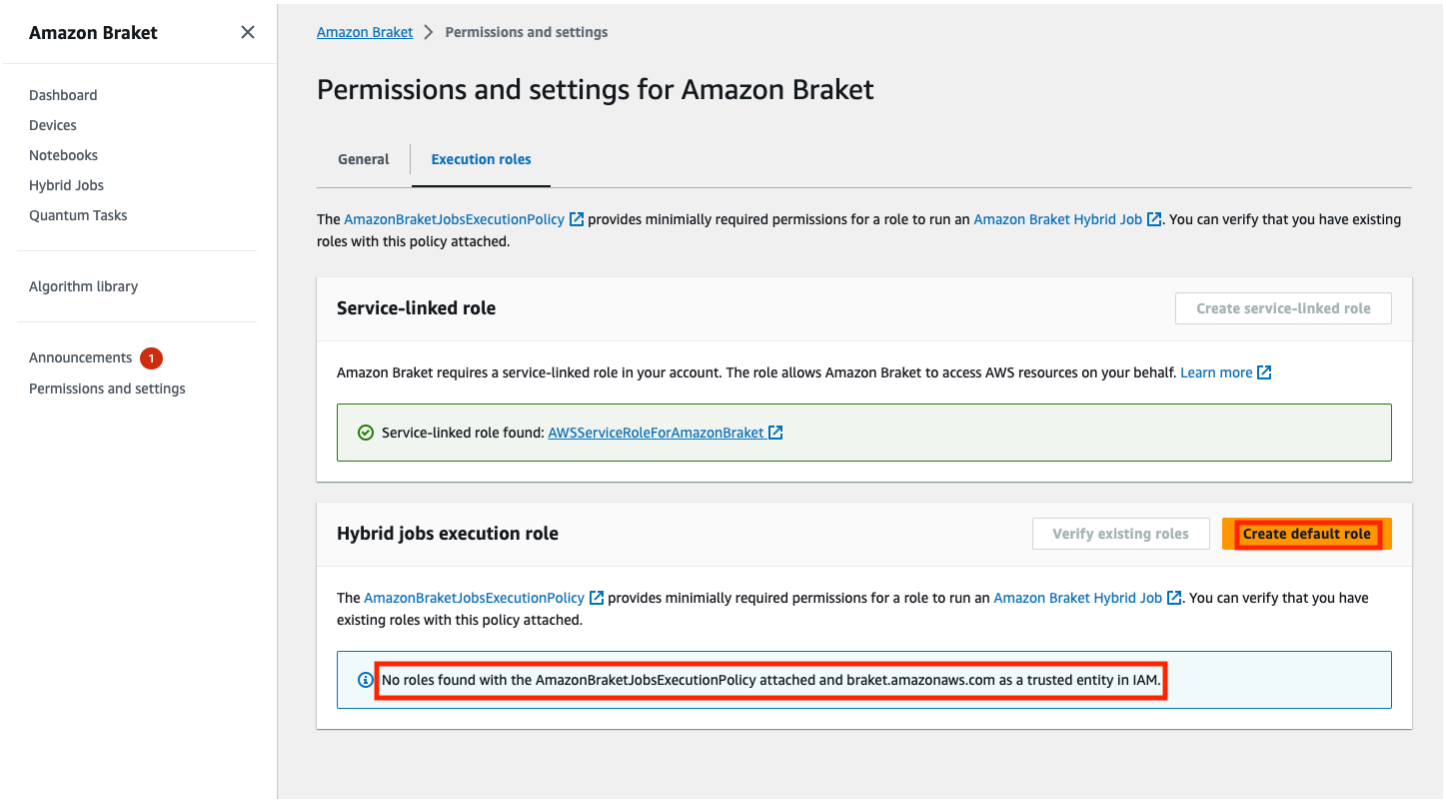
Prima di eseguire il primo lavoro ibrido, è necessario assicurarsi di disporre delle autorizzazioni sufficienti per procedere con questa attività. Per stabilire di disporre delle autorizzazioni corrette, seleziona Autorizzazioni dal menu sul lato sinistro della Braket Console. La pagina Gestione delle autorizzazioni per Amazon Braket ti aiuta a verificare se uno dei tuoi ruoli esistenti dispone di autorizzazioni sufficienti per eseguire il tuo lavoro ibrido o ti guida attraverso la creazione di un ruolo predefinito che può essere utilizzato per eseguire il tuo lavoro ibrido se non disponi già di tale ruolo.



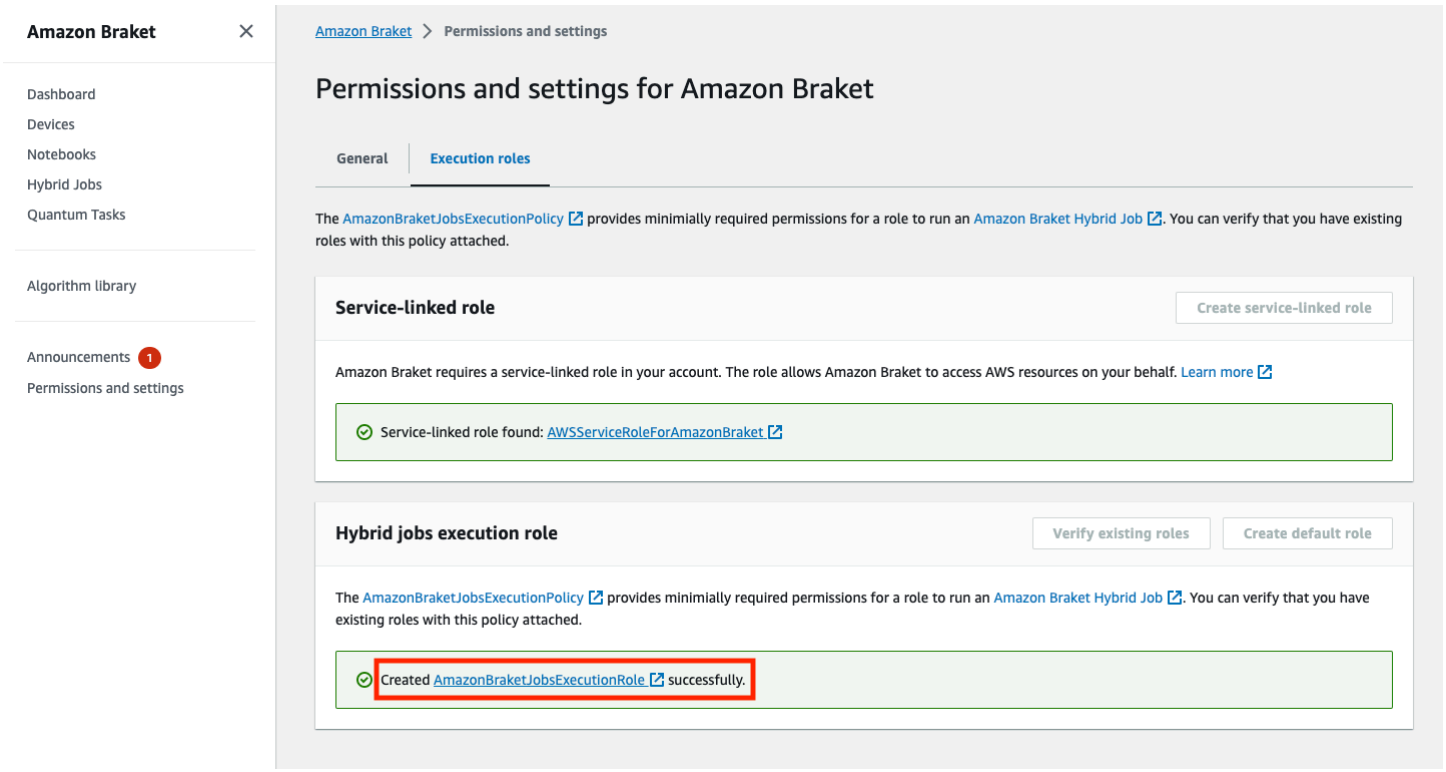
Per verificare di disporre di ruoli con autorizzazioni sufficienti per eseguire un lavoro ibrido, seleziona il pulsante Verifica il ruolo esistente. Se lo fai, ricevi un messaggio che indica che i ruoli sono stati trovati. Per visualizzare i nomi dei ruoli e il relativo ruolo ARNs, seleziona il pulsante Mostra ruoli.



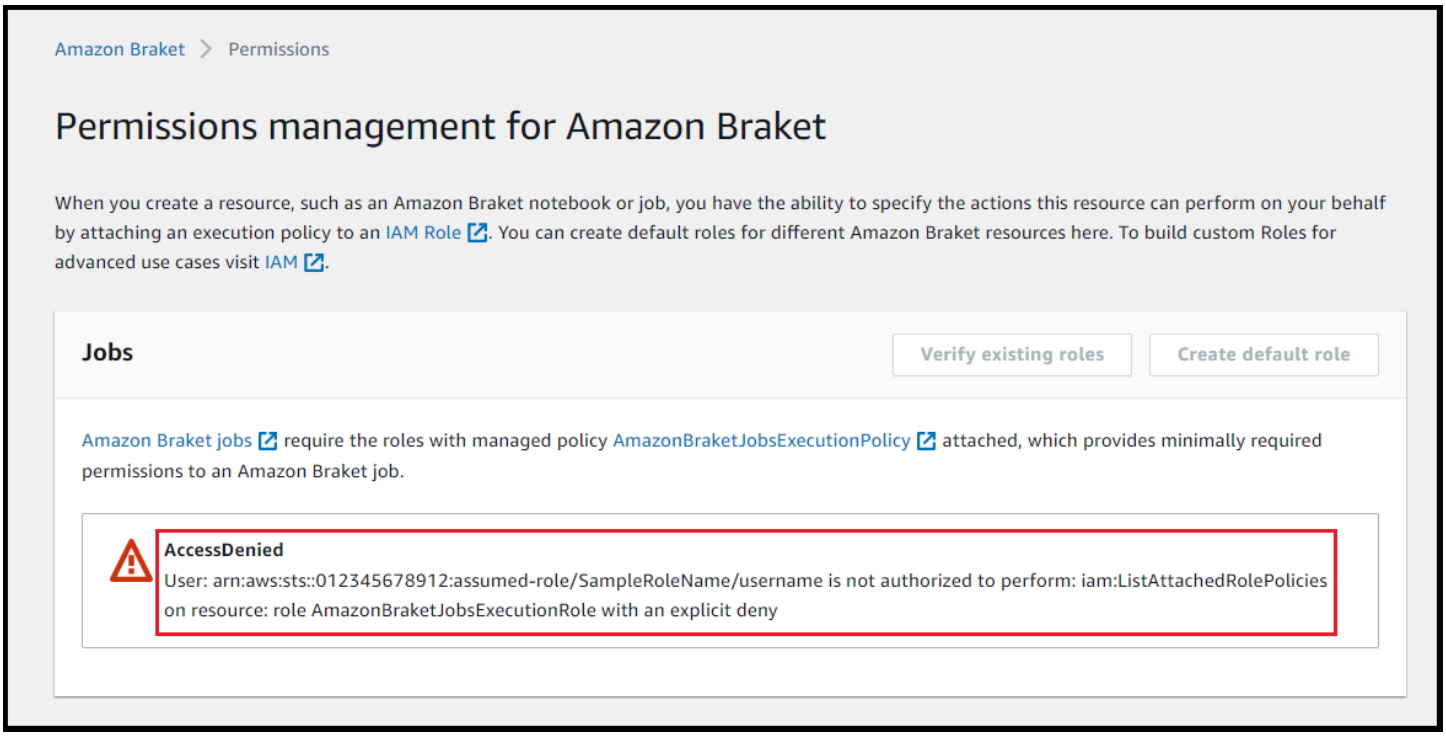
Se non disponi di un ruolo con autorizzazioni sufficienti per eseguire un lavoro ibrido, ricevi un messaggio che indica che tale ruolo non è stato trovato. Seleziona il pulsante Crea ruolo predefinito per ottenere un ruolo con autorizzazioni sufficienti.



Se il ruolo è stato creato correttamente, riceverai un messaggio di conferma.



Se non disponi delle autorizzazioni necessarie per effettuare questa richiesta, ti verrà negato l'accesso. In questo caso, contatta l' AWS amministratore interno.



Crea un Job ibrido

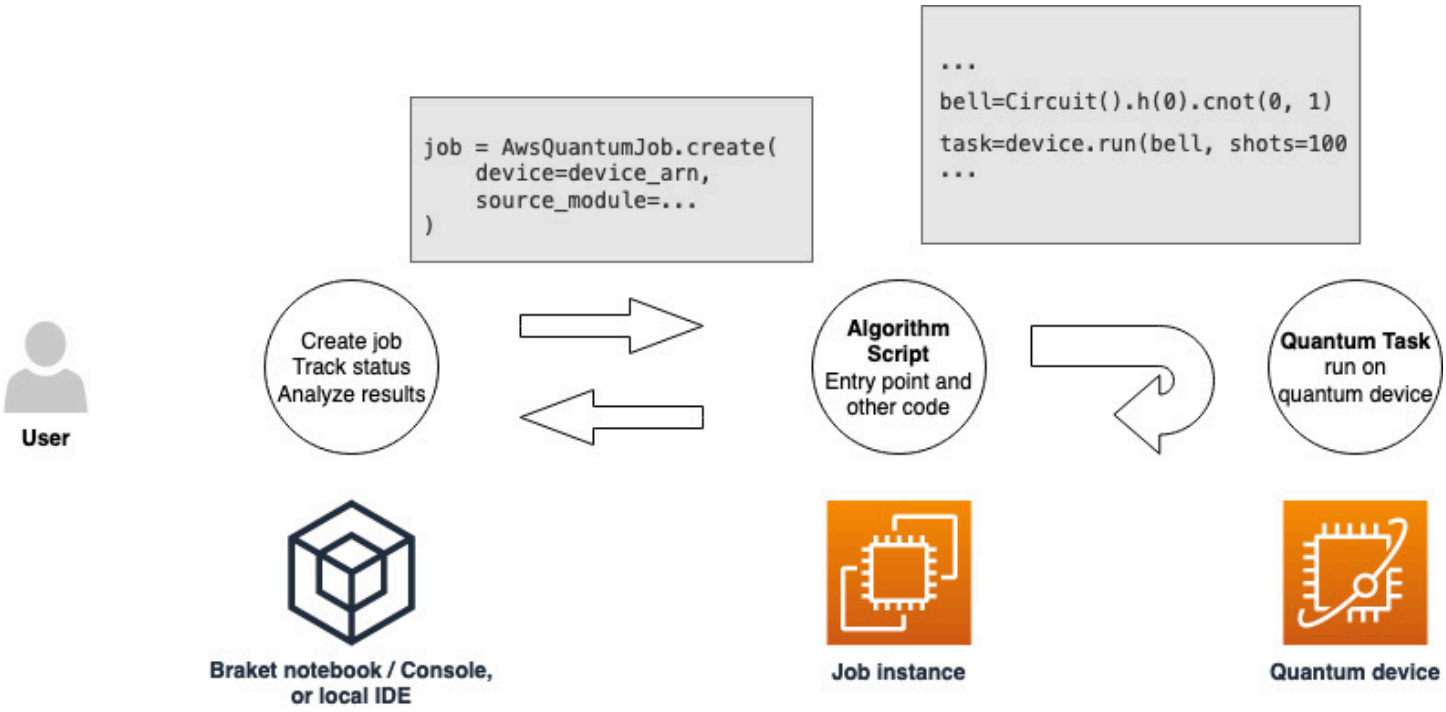
Questa sezione mostra come creare un Hybrid Job usando uno script Python. In alternativa, per creare un lavoro ibrido da codice Python locale, come il tuo ambiente di sviluppo integrato (IDE) preferito o un notebook Braket, vedi. Esegui il codice locale come lavoro ibrido

In questa sezione:

- Crea ed esegui
- Monitora i tuoi risultati
- Salva i risultati
- Utilizzo dei checkpoint
- Esegui il codice locale come lavoro ibrido
- Utilizzo dell'API con Hybrid Jobs
- Crea ed esegui il debug di un processo ibrido con modalità locale

Crea ed esegui

Una volta ottenuto un ruolo con le autorizzazioni per eseguire un lavoro ibrido, sei pronto per procedere. L'elemento chiave del tuo primo lavoro ibrido con Braket è lo script dell'algoritmo. Definisce l'algoritmo da eseguire e contiene le classiche attività logiche e quantistiche che fanno parte dell'algoritmo. Oltre allo script dell'algoritmo, puoi fornire altri file di dipendenza. Lo script dell'algoritmo, insieme alle sue dipendenze, viene chiamato modulo sorgente. Il punto di ingresso definisce il primo file o funzione da eseguire nel modulo di origine all'avvio del processo ibrido.



Innanzitutto, consideriamo il seguente esempio di base di uno script di algoritmo che crea cinque stati a campana e stampa i risultati di misurazione corrispondenti.

```
import os

from braket.aws import AwsDevice
from braket.circuits import Circuit


def start_here():

    print("Test job started!")

    # Use the device declared in the job script
    device = AwsDevice(os.environ["AMZN_BRAKET_DEVICE_ARN"])
```

```
    bell = Circuit().h(0).cnot(0, 1)
    for count in range(5):
        task = device.run(bell, shots=100)
        print(task.result().measurement_counts)

    print("Test job completed!")
```

Salvate questo file con il nome algorithm_script.py nella directory di lavoro corrente sul notebook Braket o nell'ambiente locale. Il file algorithm_script.py ha `start_here()` come punto di ingresso pianificato.

Quindi, crea un file Python o un taccuino Python nella stessa directory del file algorithm_script.py. Questo script avvia il processo ibrido e gestisce qualsiasi elaborazione asincrona, come la stampa dello stato o dei risultati chiave che ci interessano. Come minimo, questo script deve specificare lo script di lavoro ibrido e il dispositivo principale.

> **Note**
>
> Per ulteriori informazioni su come creare un notebook Braket o caricare un file, ad esempio il file algorithm_script.py, nella stessa directory dei notebook, consulta Esegui il tuo primo circuito usando l'SDK Amazon Braket Python

Per questo primo caso di base, scegli come target un simulatore. Indipendentemente dal tipo di dispositivo quantistico scelto come target, un simulatore o un'unità di elaborazione quantistica (QPU) effettiva, il dispositivo specificato `device` nello script seguente viene utilizzato per pianificare il processo ibrido ed è disponibile per gli script dell'algoritmo come variabile di ambiente. `AMZN_BRAKET_DEVICE_ARN`

> **Note**
>
> È possibile utilizzare solo i dispositivi disponibili nel processo ibrido. Regione AWS L'SDK Amazon Braket seleziona automaticamente questa opzione. Regione AWS Ad esempio, un lavoro ibrido in us-east-1 può IonQ utilizzareSV1,, TN1 e dispositiviDM1, ma non dispositivi. Rigetti

Se scegli un computer quantistico anziché un simulatore, Braket pianifica i tuoi lavori ibridi per eseguire tutte le loro attività quantistiche con accesso prioritario.

```
from braket.aws import AwsQuantumJob
from braket.devices import Devices

job = AwsQuantumJob.create(
    Devices.Amazon.SV1,
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
    wait_until_complete=True
)
```

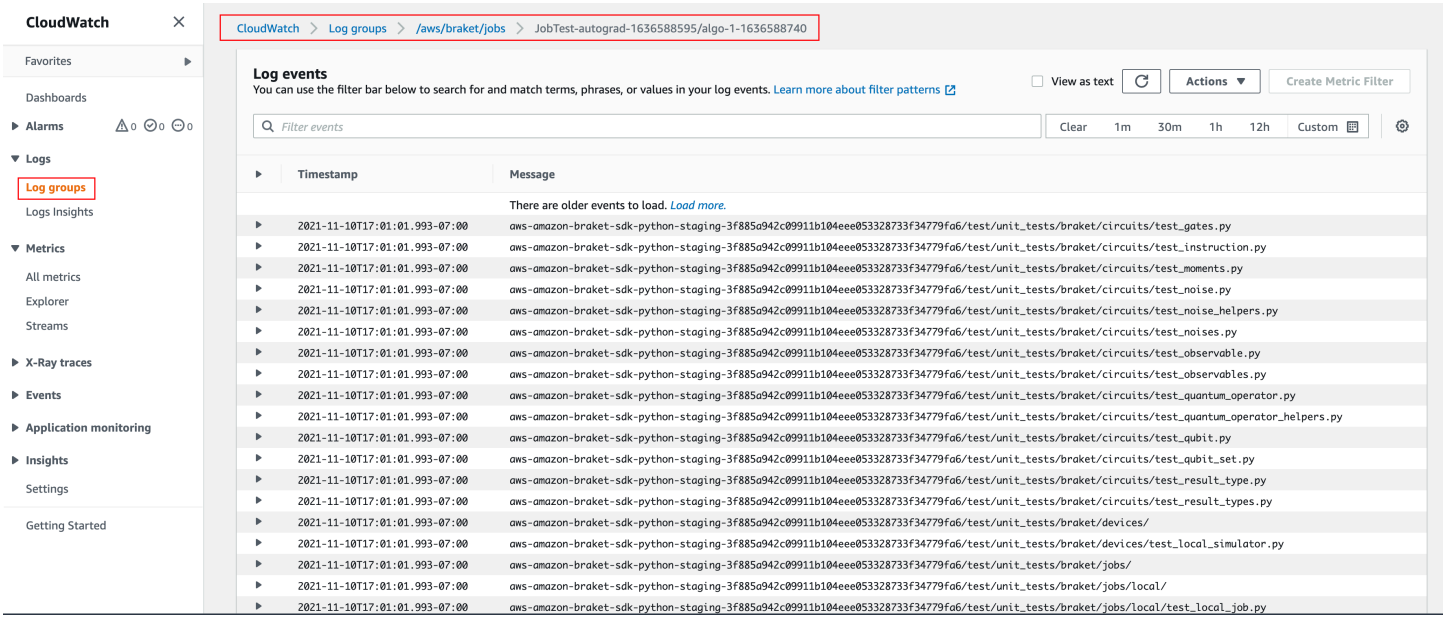
Il parametro `wait_until_complete=True` imposta una modalità dettagliata in modo che il lavoro stampi l'output del lavoro effettivo mentre è in esecuzione. Dovreste vedere un output simile a quello dell'esempio seguente.

```
Initializing Braket Job: arn:aws:braket:us-west-2:111122223333:job/braket-job-
default-123456789012
Job queue position: 1
Job queue position: 1
Job queue position: 1
..............
.
.
.
Beginning Setup
Checking for Additional Requirements
Additional Requirements Check Finished
Running Code As Process
Test job started!
Counter({'00': 58, '11': 42})
Counter({'00': 55, '11': 45})
Counter({'11': 51, '00': 49})
Counter({'00': 56, '11': 44})
Counter({'11': 56, '00': 44})
Test job completed!
Code Run Finished
2025-09-24 23:13:40,962 sagemaker-training-toolkit INFO     Reporting training SUCCESS
```

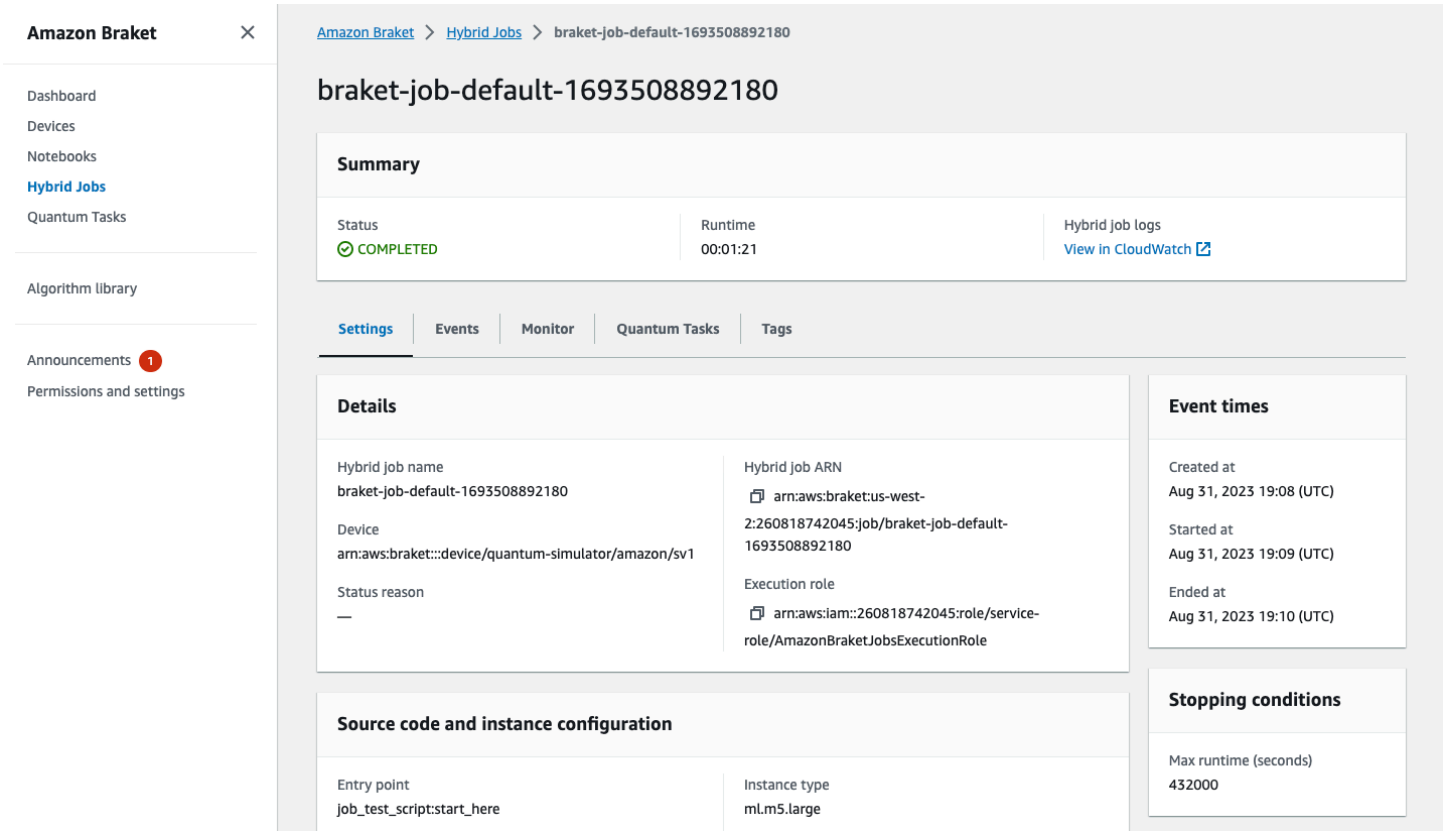
> **Note**
>
> Puoi anche usare il tuo modulo personalizzato con il metodo AwsQuantumJob.create passandone la posizione (il percorso di una directory o di un file locale o un URI S3 di un file tar.gz). Per un esempio funzionante, consulta il file Parallelize_training_for_qml.ipynb nella cartella hybrid jobs nel repository Amazon Braket examples Github.

Monitora i tuoi risultati

In alternativa, puoi accedere all'output del registro da Amazon CloudWatch. Per fare ciò, vai alla scheda Gruppi di log nel menu a sinistra della pagina di dettaglio del lavoro, seleziona il gruppo di log`aws/braket/jobs`, quindi scegli il flusso di log che contiene il nome del lavoro. Nell'esempio precedente è `braket-job-default-1631915042705/algo-1-1631915190`.



Puoi anche visualizzare lo stato del lavoro ibrido nella console selezionando la pagina Hybrid Jobs e quindi scegliendo Impostazioni.



Il tuo lavoro ibrido produce alcuni artefatti in Amazon S3 mentre è in esecuzione. Il nome predefinito del bucket S3 è `amazon-braket-<region>-<accountid>` e il contenuto si trova nella directory. `jobs/<jobname>/<timestamp>` Puoi configurare le posizioni S3 in cui vengono archiviati questi artefatti specificandone una diversa `code_location` quando il lavoro ibrido viene creato con Braket Python SDK.

> **Note**
>
> Questo bucket S3 deve trovarsi nella stessa posizione del job script. Regione AWS

La `jobs/<jobname>/<timestamp>` directory contiene una sottocartella con l'output dello script del punto di ingresso in un file. `model.tar.gz` Esiste anche una directory denominata `script` che contiene gli elementi dello script dell'algoritmo in un file. `source.tar.gz` I risultati delle vostre attività quantistiche effettive si trovano nella directory denominata. `jobs/<jobname>/tasks`

Salva i risultati

È possibile salvare i risultati generati dallo script dell'algoritmo in modo che siano disponibili dall'oggetto di lavoro ibrido nello script del processo ibrido e dalla cartella di output in Amazon S3 (in un file tar-zip denominato model.tar.gz).

L'output deve essere salvato in un file utilizzando un formato JavaScript Object Notation (JSON). Se i dati non possono essere serializzati prontamente in testo, come nel caso di un array numpy, puoi passare un'opzione per serializzare utilizzando un formato di dati selezionato. Vedi il modulo braket.jobs.data_persistence per maggiori dettagli.

Per salvare i risultati dei lavori ibridi, aggiungi le seguenti righe commentate con #ADD al file algorithm_script.py.

```
import os

from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.jobs import save_job_result  # ADD


def start_here():

    print("Test job started!")

    device = AwsDevice(os.environ['AMZN_BRAKET_DEVICE_ARN'])

    results = []  # ADD

    bell = Circuit().h(0).cnot(0, 1)
    for count in range(5):
        task = device.run(bell, shots=100)
        print(task.result().measurement_counts)
        results.append(task.result().measurement_counts)  # ADD

        save_job_result({"measurement_counts": results})  # ADD

    print("Test job completed!")
```

È quindi possibile visualizzare i risultati del lavoro dal proprio script di lavoro aggiungendo la riga **`print(job.result())`**commentata con #ADD.

```
import time
from braket.aws import AwsQuantumJob

job = AwsQuantumJob.create(
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
    device="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
)

print(job.arn)
while job.state() not in AwsQuantumJob.TERMINAL_STATES:
    print(job.state())
    time.sleep(10)

print(job.state())
print(job.result())   # ADD
```

In questo esempio, abbiamo rimosso `wait_until_complete=True` per sopprimere l'output verboso. Puoi aggiungerlo nuovamente per il debug. Quando si esegue questo processo ibrido, vengono emessi l'identificatore e il`job-arn`, seguiti dallo stato del lavoro ibrido ogni 10 secondi fino all'arrivo del lavoro ibrido`COMPLETED`, dopodiché vengono visualizzati i risultati del circuito a campana. Guarda l'esempio seguente.

```
arn:aws:braket:us-west-2:111122223333:job/braket-job-default-123456789012
INITIALIZED
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
RUNNING
...
RUNNING
RUNNING
COMPLETED
{'measurement_counts': [{'11': 53, '00': 47},..., {'00': 51, '11': 49}]}
```

Utilizzo dei checkpoint

Puoi salvare iterazioni intermedie dei tuoi lavori ibridi utilizzando i checkpoint. Nell'esempio di script di algoritmo della sezione precedente, dovresti aggiungere le seguenti righe commentate con #ADD per creare file di checkpoint.

```
from braket.aws import AwsDevice
from braket.circuits import Circuit
from braket.jobs import save_job_checkpoint  # ADD
import os


def start_here():

    print("Test job starts!")

    device = AwsDevice(os.environ["AMZN_BRAKET_DEVICE_ARN"])

    # ADD the following code
    job_name = os.environ["AMZN_BRAKET_JOB_NAME"]
    save_job_checkpoint(checkpoint_data={"data": f"data for checkpoint from
 {job_name}"}, checkpoint_file_suffix="checkpoint-1")  # End of ADD

    bell = Circuit().h(0).cnot(0, 1)
    for count in range(5):
        task = device.run(bell, shots=100)
        print(task.result().measurement_counts)

    print("Test hybrid job completed!")
```

Quando si esegue il job ibrido, viene creato il file -checkpoint-1.json <jobname>negli artefatti del job ibrido nella directory checkpoints con un percorso predefinito. `/opt/jobs/checkpoints` Lo script di lavoro ibrido rimane invariato a meno che non si desideri modificare questo percorso predefinito.

Se si desidera caricare un lavoro ibrido da un checkpoint generato da un precedente lavoro ibrido, lo script dell'algoritmo utilizza. `from braket.jobs import load_job_checkpoint` La logica da caricare nello script dell'algoritmo è la seguente.

```
from braket.jobs import load_job_checkpoint

checkpoint_1 = load_job_checkpoint(
```

```
    "previous_job_name",
    checkpoint_file_suffix="checkpoint-1",
)
```

Dopo aver caricato questo checkpoint, puoi continuare la logica in base al contenuto caricato su. `checkpoint-1`

> **Note**
>
> Il checkpoint_file_suffix deve corrispondere al suffisso precedentemente specificato durante la creazione del checkpoint.

Lo script di orchestrazione deve specificare il precedente lavoro ibrido con la riga commentata con `job-arn` #ADD.

```
from braket.aws import AwsQuantumJob

job = AwsQuantumJob.create(
    source_module="source_dir",
    entry_point="source_dir.algorithm_script:start_here",
    device="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
    copy_checkpoints_from_job="<previous-job-ARN>", #ADD
    )
```

Esegui il codice locale come lavoro ibrido

Amazon Braket Hybrid Jobs fornisce un'orchestrazione completamente gestita di algoritmi ibridi quantistici classici, combinando le risorse di calcolo di Amazon EC2 con l'accesso ad Amazon Braket Quantum Processing Unit (QPU). Le attività quantistiche create in un processo ibrido hanno la priorità di essere messe in coda rispetto alle singole attività quantistiche, in modo che gli algoritmi non vengano interrotti dalle fluttuazioni nella coda delle attività quantistiche. Ogni QPU mantiene una coda di lavori ibridi separata, garantendo che sia possibile eseguire solo un processo ibrido alla volta.

In questa sezione:

- Crea un lavoro ibrido dal codice Python locale
- Installa pacchetti Python e codice sorgente aggiuntivi
- Salva e carica i dati in un'istanza di lavoro ibrida

- Le migliori pratiche per arredatori di lavori ibridi

Crea un lavoro ibrido dal codice Python locale

Puoi eseguire il codice Python locale come Amazon Braket Hybrid Job. Puoi farlo annotando il codice con un `@hybrid_job` decoratore, come mostrato nel seguente esempio di codice. Per gli ambienti personalizzati, puoi scegliere di utilizzare un contenitore personalizzato da Amazon Elastic Container Registry (ECR).

> **Note**
>
> Per impostazione predefinita, è supportato solo Python 3.12.

È possibile utilizzare il `@hybrid_job` decoratore per annotare una funzione. Braket trasforma il codice all'interno del decoratore in uno script di algoritmo di lavoro ibrido Braket. Il job ibrido richiama quindi la funzione all'interno del decoratore su un'istanza Amazon EC2. Puoi monitorare l'avanzamento del lavoro con `job.state()` o con la console Braket. Il seguente esempio di codice mostra come eseguire una sequenza di cinque stati su. State Vector Simulator (SV1) device

```
from braket.aws import AwsDevice
from braket.circuits import Circuit, FreeParameter, Observable
from braket.devices import Devices
from braket.jobs.hybrid_job import hybrid_job
from braket.jobs.metrics import log_metric

device_arn = Devices.Amazon.SV1


@hybrid_job(device=device_arn)  # Choose priority device
def run_hybrid_job(num_tasks=1):
    device = AwsDevice(device_arn)  # Declare AwsDevice within the hybrid job

    # Create a parametric circuit
    circ = Circuit()
    circ.rx(0, FreeParameter("theta"))
    circ.cnot(0, 1)
    circ.expectation(observable=Observable.X(), target=0)

    theta = 0.0  # Initial parameter
```

```
    for i in range(num_tasks):
        task = device.run(circ, shots=100, inputs={"theta": theta})  # Input parameters
        exp_val = task.result().values[0]

        theta += exp_val  # Modify the parameter (possibly gradient descent)

        log_metric(metric_name="exp_val", value=exp_val, iteration_number=i)

    return {"final_theta": theta, "final_exp_val": exp_val}
```

Crei il lavoro ibrido invocando la funzione come faresti con le normali funzioni di Python. Tuttavia, la funzione decorator restituisce l'handle del lavoro ibrido anziché il risultato della funzione. Per recuperare i risultati dopo il completamento, usa. `job.result()`

```
job = run_hybrid_job(num_tasks=1)
result = job.result()
```

L'argomento device nel `@hybrid_job` decoratore specifica il dispositivo a cui il lavoro ibrido ha accesso prioritario, in questo caso, il simulatore. SV1 Per ottenere la priorità QPU, è necessario assicurarsi che l'ARN del dispositivo utilizzato all'interno della funzione corrisponda a quello specificato nel decoratore. Per comodità, è possibile utilizzare la funzione helper per `get_job_device_arn()` acquisire l'ARN del dispositivo dichiarato in. `@hybrid_job`

> **Note**
>
> Ogni processo ibrido ha un tempo di avvio di almeno un minuto poiché crea un ambiente containerizzato su Amazon EC2. Quindi, per carichi di lavoro molto brevi, come un singolo circuito o un batch di circuiti, può essere sufficiente utilizzare attività quantistiche.

Iperparametri

La `run_hybrid_job()` funzione utilizza l'argomento `num_tasks` per controllare il numero di attività quantistiche create. Il processo ibrido lo acquisisce automaticamente come iperparametro.

> **Note**
>
> Gli iperparametri vengono visualizzati nella console Braket come stringhe, limitate a 2500 caratteri.

Metriche e registrazione

All'interno della `run_hybrid_job()` funzione, vengono registrate le metriche degli algoritmi iterativi con. `log_metrics` Le metriche vengono tracciate automaticamente nella pagina della console Braket nella scheda del lavoro ibrido. Puoi utilizzare le metriche per tracciare i costi quantistici delle attività in tempo quasi reale durante l'esecuzione del lavoro ibrido con il tracker dei costi di Braket. L'esempio precedente utilizza il nome della metrica «probabilità» che registra la prima probabilità del tipo di risultato.

Recupero dei risultati

Una volta completato il lavoro ibrido, si utilizza `job.result()` per recuperare i risultati dei lavori ibridi. Tutti gli oggetti nell'istruzione return vengono acquisiti automaticamente da Braket. Nota che gli oggetti restituiti dalla funzione devono essere una tupla con ogni elemento serializzabile. Ad esempio, il codice seguente mostra un esempio funzionante e uno non riuscito.

```
import numpy as np


# Working example
@hybrid_job(device=Devices.Amazon.SV1)
def passing():
    np_array = np.random.rand(5)
    return np_array  # Serializable

# # Failing example
# @hybrid_job(device=Devices.Amazon.SV1)
# def failing():
#     return MyObject() # Not serializable
```

Nome del lavoro

Per impostazione predefinita, il nome di questo lavoro ibrido viene dedotto dal nome della funzione. È inoltre possibile specificare un nome personalizzato lungo fino a 50 caratteri. Ad esempio, nel codice seguente il nome del lavoro è "my-job-name».

```
@hybrid_job(device=Devices.Amazon.SV1, job_name="my-job-name")
def function():
    pass
```

modalità locale

I lavori locali vengono creati aggiungendo l'argomento `local=True` al decoratore. Questo esegue il lavoro ibrido in un ambiente containerizzato sull'ambiente di elaborazione locale, ad esempio il laptop. I lavori locali non prevedono la priorità delle code per le attività quantistiche. Per casi avanzati come multi-nodo o MPI, i job locali possono avere accesso alle variabili di ambiente Braket richieste. Il codice seguente crea un processo ibrido locale con il dispositivo come simulatore. SV1

```
@hybrid_job(device=Devices.Amazon.SV1, local=True)
def run_hybrid_job(num_tasks=1):
    return ...
```

Sono supportate tutte le altre opzioni di lavoro ibride. Per un elenco di opzioni, consultate il modulo braket.jobs.quantum_job_creation.

Installa pacchetti Python e codice sorgente aggiuntivi

Puoi personalizzare il tuo ambiente di runtime per usare i tuoi pacchetti Python preferiti. È possibile utilizzare un `requirements.txt` file, un elenco di nomi di pacchetti o portare il proprio contenitore (BYOC). Ad esempio, il `requirements.txt` file può includere altri pacchetti da installare.

```
qiskit
pennylane >= 0.31
mitiq == 0.29
```

Per personalizzare un ambiente di runtime utilizzando un `requirements.txt` file, fate riferimento al seguente esempio di codice.

```
@hybrid_job(device=Devices.Amazon.SV1, dependencies="requirements.txt")
def run_hybrid_job(num_tasks=1):
    return ...
```

In alternativa, puoi fornire i nomi dei pacchetti come elenco Python come segue.

```
@hybrid_job(device=Devices.Amazon.SV1, dependencies=["qiskit", "pennylane>=0.31",
 "mitiq==0.29"])
def run_hybrid_job(num_tasks=1):
    return ...
```

Il codice sorgente aggiuntivo può essere specificato come elenco di moduli o come singolo modulo come nel seguente esempio di codice.

```
@hybrid_job(device=Devices.Amazon.SV1, include_modules=["my_module1", "my_module2"])
def run_hybrid_job(num_tasks=1):
    return ...
```

Salva e carica i dati in un'istanza di lavoro ibrida

Specificazione dei dati di addestramento in ingresso

Quando crei un lavoro ibrido, puoi fornire un set di dati di addestramento di input specificando un bucket Amazon Simple Storage Service (Amazon S3). Puoi anche specificare un percorso locale, quindi Braket carica automaticamente i dati su Amazon S3 all'indirizzo. `s3://<default_bucket_name>/jobs/<job_name>/<timestamp>/data/<channel_name>` Se specifichi un percorso locale, il nome del canale è predefinito su «input». Il codice seguente mostra un file numpy dal percorso locale. `data/file.npy`

```
import numpy as np


@hybrid_job(device=Devices.Amazon.SV1, input_data="data/file.npy")
def run_hybrid_job(num_tasks=1):
    data = np.load("data/file.npy")
    return ...
```

Per S3, è necessario utilizzare la funzione di `get_input_data_dir()` supporto.

```
import numpy as np
from braket.jobs import get_input_data_dir

s3_path = "s3://amazon-braket-us-east-1-123456789012/job-data/file.npy"


@hybrid_job(device=None, input_data=s3_path)
def job_s3_input():
    np.load(get_input_data_dir() + "/file.npy")


@hybrid_job(device=None, input_data={"channel": s3_path})
def job_s3_input_channel():
    np.load(get_input_data_dir("channel") + "/file.npy")
```

È possibile specificare più fonti di dati di input fornendo un dizionario dei valori dei canali e dei percorsi URIs S3 o locali.

```
import numpy as np
from braket.jobs import get_input_data_dir

input_data = {
    "input": "data/file.npy",
    "input_2": "s3://amzn-s3-demo-bucket/data.json"
}


@hybrid_job(device=None, input_data=input_data)
def multiple_input_job():
    np.load(get_input_data_dir("input") + "/file.npy")
    np.load(get_input_data_dir("input_2") + "/data.json")
```

> **Note**
>
> Quando i dati di input sono di grandi dimensioni (>1 GB), c'è un lungo tempo di attesa prima che il lavoro venga creato. Ciò è dovuto ai dati di input locali che vengono caricati per la prima volta in un bucket S3, quindi il percorso S3 viene aggiunto alla richiesta di lavoro. Infine, la richiesta di lavoro viene inviata al servizio Braket.

Salvataggio dei risultati su S3

Per salvare i risultati non inclusi nell'istruzione return della funzione decorata, è necessario aggiungere la directory corretta a tutte le operazioni di scrittura dei file. L'esempio seguente mostra il salvataggio di un array numpy e di una figura matplotlib.

```
import matplotlib.pyplot as plt
import numpy as np


@hybrid_job(device=Devices.Amazon.SV1)
def run_hybrid_job(num_tasks=1):
    result = np.random.rand(5)

    # Save a numpy array
    np.save("result.npy", result)
```

```
    # Save a matplotlib figure
    plt.plot(result)
    plt.savefig("fig.png")
    return ...
```

Tutti i risultati vengono compressi in un file denominato. `model.tar.gz` Puoi scaricare i risultati con la funzione `job.result()` Python o accedendo alla cartella dei risultati dalla pagina del lavoro ibrido nella console di gestione Braket.

Salvataggio e ripresa dai checkpoint

Per lavori ibridi di lunga durata, si consiglia di salvare periodicamente lo stato intermedio dell'algoritmo. È possibile utilizzare la funzione di `save_job_checkpoint()` supporto integrata o salvare i file nel percorso. `AMZN_BRAKET_JOB_RESULTS_DIR` Quest'ultima è disponibile con la funzione helper. `get_job_results_dir()`

Quello che segue è un esempio di funzionamento minimo per salvare e caricare i checkpoint con un Job Decorator ibrido:

```
from braket.jobs import save_job_checkpoint, load_job_checkpoint, hybrid_job


@hybrid_job(device=None, wait_until_complete=True)
def function():
    save_job_checkpoint({"a": 1})


job = function()
job_name = job.name
job_arn = job.arn


@hybrid_job(device=None, wait_until_complete=True, copy_checkpoints_from_job=job_arn)
def continued_function():
    load_job_checkpoint(job_name)


continued_job = continued_function()
```

Nel primo job ibrido, `save_job_checkpoint()` viene chiamato con un dizionario contenente i dati che vogliamo salvare. Per impostazione predefinita, ogni valore deve essere serializzabile

come testo. Per controllare oggetti Python più complessi, come gli array numpy, puoi impostare. `data_format = PersistedJobDataFormat.PICKLED_V4` Questo codice crea e sovrascrive un file di checkpoint con nome predefinito negli artefatti del job ibrido `<jobname>.json` in una sottocartella chiamata «checkpoints».

Per creare un nuovo lavoro ibrido che prosegua dal checkpoint, dobbiamo indicare `copy_checkpoints_from_job=job_arn` `job_arn` dov'è l'ARN del lavoro ibrido del lavoro precedente. Quindi eseguiamo `load_job_checkpoint(job_name)` il caricamento dal checkpoint.

Le migliori pratiche per arredatori di lavori ibridi

Abbraccia l'asincronicità

I lavori ibridi creati con l'annotazione decorator sono asincroni: vengono eseguiti non appena le risorse classiche e quantistiche sono disponibili. Monitora l'avanzamento dell'algoritmo utilizzando Braket Management Console o Amazon CloudWatch. Quando invii l'algoritmo per l'esecuzione, Braket lo esegue in un ambiente containerizzato scalabile e i risultati vengono recuperati quando l'algoritmo è completo.

Esegui algoritmi variazionali iterativi

Hybrid jobs ti offre gli strumenti per eseguire algoritmi iterativi quantistici classici. Per problemi puramente quantistici, utilizzate attività quantistiche o una serie di attività quantistiche. L'accesso prioritario a determinati QPUs è particolarmente vantaggioso per gli algoritmi variazionali di lunga durata che richiedono più chiamate iterative a o con l'elaborazione classica intermedia. QPUs

Esegui il debug utilizzando la modalità locale

Prima di eseguire un job ibrido su una QPU, si consiglia di eseguirlo prima sul simulatore SV1 per confermare che funzioni come previsto. Per i test su piccola scala, puoi eseguirli in modalità locale per iterazioni e debug rapidi.

Migliora la riproducibilità con Bring your own container (BYOC)

Crea un esperimento riproducibile incapsulando il tuo software e le sue dipendenze in un ambiente containerizzato. Comprimendo tutto il codice, le dipendenze e le impostazioni in un contenitore, si evitano potenziali conflitti e problemi di versione.

Simulatori distribuiti a più istanze

Per eseguire un gran numero di circuiti, prendi in considerazione l'utilizzo del supporto MPI integrato per eseguire simulatori locali su più istanze all'interno di un singolo processo ibrido. Per ulteriori informazioni, consulta Simulatori incorporati.

Usa circuiti parametrici

I circuiti parametrici inviati da un processo ibrido vengono compilati automaticamente su determinati circuiti QPUs utilizzando la compilazione parametrica per migliorare i tempi di esecuzione degli algoritmi.

Checkpoint periodicamente

Per lavori ibridi di lunga durata, si consiglia di salvare periodicamente lo stato intermedio dell'algoritmo.

Per ulteriori esempi, casi d'uso e best practice, consulta gli esempi di Amazon GitHub Braket.

Utilizzo dell'API con Hybrid Jobs

Puoi accedere e interagire con Amazon Braket Hybrid Jobs direttamente utilizzando. API Tuttavia, le impostazioni predefinite e i metodi pratici non sono disponibili quando si utilizza direttamente. API

> **Note**
>
> Ti consigliamo vivamente di interagire con Amazon Braket Hybrid Jobs utilizzando l'SDK Amazon Braket Python. Offre impostazioni predefinite e protezioni convenienti che aiutano i processi ibridi a funzionare correttamente.

In questo argomento vengono illustrate le nozioni di base sull'utilizzo di. API Se scegli di utilizzare l'API, tieni presente che questo approccio può essere più complesso e prepararti a diverse iterazioni per far funzionare il tuo lavoro ibrido.

Per utilizzare l'API, il tuo account deve avere un ruolo nella politica `AmazonBraketFullAccess` gestita.

> **Note**
>
> Per ulteriori informazioni su come ottenere un ruolo con la policy `AmazonBraketFullAccess` gestita, consulta la pagina Abilita Amazon Braket.

Inoltre, è necessario un ruolo di esecuzione. Questo ruolo verrà passato al servizio. Puoi creare il ruolo utilizzando la console Amazon Braket. Utilizza la scheda Ruoli di esecuzione nella pagina Autorizzazioni e impostazioni per creare un ruolo predefinito per i lavori ibridi.

È `CreateJob` API necessario specificare tutti i parametri richiesti per il lavoro ibrido. Per usare Python, comprimi i file di script dell'algoritmo in un pacchetto tar, ad esempio un file input.tar.gz, ed esegui lo script seguente. Aggiorna le parti del codice tra parentesi angolate (<>) in modo che corrispondano alle informazioni dell'account e al punto di ingresso che specificano il percorso, il file e il metodo con cui inizia il processo ibrido.

```
from braket.aws import AwsDevice, AwsSession
import boto3
from datetime import datetime

s3_client = boto3.client("s3")
client = boto3.client("braket")

project_name = "job-test"
job_name = project_name + "-" + datetime.strftime(datetime.now(), "%Y%m%d%H%M%S")
bucket = "amazon-braket-<your_bucket>"
s3_prefix = job_name

job_script = "input.tar.gz"
job_object = f"{s3_prefix}/script/{job_script}"
s3_client.upload_file(job_script, bucket, job_object)

input_data = "inputdata.csv"
input_object = f"{s3_prefix}/input/{input_data}"
s3_client.upload_file(input_data, bucket, input_object)

job = client.create_job(
    jobName=job_name,
    roleArn="arn:aws:iam::<your_account>:role/service-role/
AmazonBraketJobsExecutionRole",  # https://docs.aws.amazon.com/braket/latest/
developerguide/braket-manage-access.html#about-amazonbraketjobsexecution
    algorithmSpecification={
        "scriptModeConfig": {
            "entryPoint": "<your_execution_module>:<your_execution_method>",
            "containerImage": {"uri": "292282985366.dkr.ecr.us-west-1.amazonaws.com/
amazon-braket-base-jobs:1.0-cpu-py37-ubuntu18.04"},   # Change to the specific region
 you are using
            "s3Uri": f"s3://{bucket}/{job_object}",
            "compressionType": "GZIP"
```

```
        }
    },
    inputDataConfig=[
        {
            "channelName": "hellothere",
            "compressionType": "NONE",
            "dataSource": {
                "s3DataSource": {
                    "s3Uri": f"s3://{bucket}/{s3_prefix}/input",
                    "s3DataType": "S3_PREFIX"
                }
            }
        }
    ],
    outputDataConfig={
        "s3Path": f"s3://{bucket}/{s3_prefix}/output"
    },
    instanceConfig={
        "instanceType": "ml.m5.large",
        "instanceCount": 1,
        "volumeSizeInGb": 1
    },
    checkpointConfig={
        "s3Uri":  f"s3://{bucket}/{s3_prefix}/checkpoints",
        "localPath": "/opt/omega/checkpoints"
    },
    deviceConfig={
        "priorityAccess": {
            "devices": [
                "arn:aws:braket:us-west-1::device/qpu/rigetti/Ankaa-3"
            ]
        }
    },
    hyperParameters={
        "hyperparameter key you wish to pass": "<hyperparameter value you wish to
 pass>",
    },
    stoppingCondition={
        "maxRuntimeInSeconds": 1200,
        "maximumTaskLimit": 10
    },
)
```

Dopo aver creato il lavoro ibrido, puoi accedere ai dettagli del lavoro ibrido tramite `GetJob` API o la console. Per ottenere i dettagli del lavoro ibrido dalla sessione Python in cui hai eseguito il `createJob` codice come nell'esempio precedente, usa il seguente comando Python.

```
getJob = client.get_job(jobArn=job["jobArn"])
```

Per annullare un lavoro ibrido, chiamate the `CancelJob` API with the Amazon Resource Name of the job ()'JobArn'.

```
cancelJob = client.cancel_job(jobArn=job["jobArn"])
```

È possibile specificare i checkpoint come parte dell'`createJob`APIutilizzo del `checkpointConfig` parametro.

```
    checkpointConfig = {
        "localPath" : "/opt/omega/checkpoints",
        "s3Uri": f"s3://{bucket}/{s3_prefix}/checkpoints"
    },
```

> **Note**
>
> Il LocalPath di `checkpointConfig` non può iniziare con nessuno dei seguenti percorsi riservati:`/opt/ml`,, `/opt/braket/tmp`, o. `/usr/local/nvidia`

Crea ed esegui il debug di un processo ibrido con modalità locale

Quando si crea un nuovo algoritmo ibrido, la modalità locale consente di eseguire il debug e il test dello script dell'algoritmo. La modalità locale è una funzionalità che ti consente di eseguire il codice che intendi utilizzare in Amazon Braket Hybrid Jobs, ma senza bisogno di Braket per gestire l'infrastruttura per l'esecuzione del lavoro ibrido. Esegui invece lavori ibridi localmente sulla tua istanza Amazon Braket Notebook o su un client preferito, come un laptop o un computer desktop.

In modalità locale, puoi comunque inviare attività quantistiche a dispositivi reali, ma non ottieni vantaggi in termini di prestazioni quando esegui su un'unità di elaborazione quantistica (QPU) effettiva in modalità locale.

Per utilizzare la modalità locale, modificatela `AwsQuantumJob` `LocalQuantumJob` ovunque si verifichi all'interno del programma. Ad esempio, per eseguire l'esempio di Create your first hybrid job, modificate lo script del job ibrido nel codice come segue.

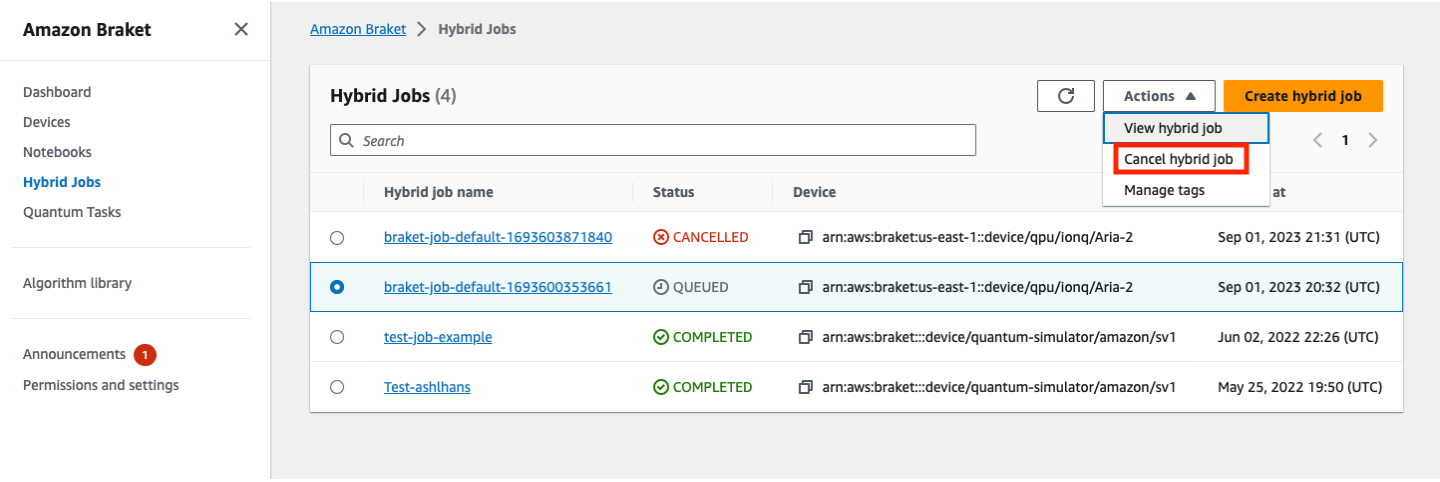
```
from braket.jobs.local import LocalQuantumJob

job = LocalQuantumJob.create(
    device="arn:aws:braket:::device/quantum-simulator/amazon/sv1",
    source_module="algorithm_script.py",
    entry_point="algorithm_script:start_here",
)
```

> **Note**
>
> Docker, che è già preinstallato nei notebook Amazon Braket, deve essere installato nel tuo ambiente locale per utilizzare questa funzionalità. Le istruzioni per l'installazione di Docker sono disponibili nella pagina Get Docker. Inoltre, non tutti i parametri sono supportati in modalità locale.

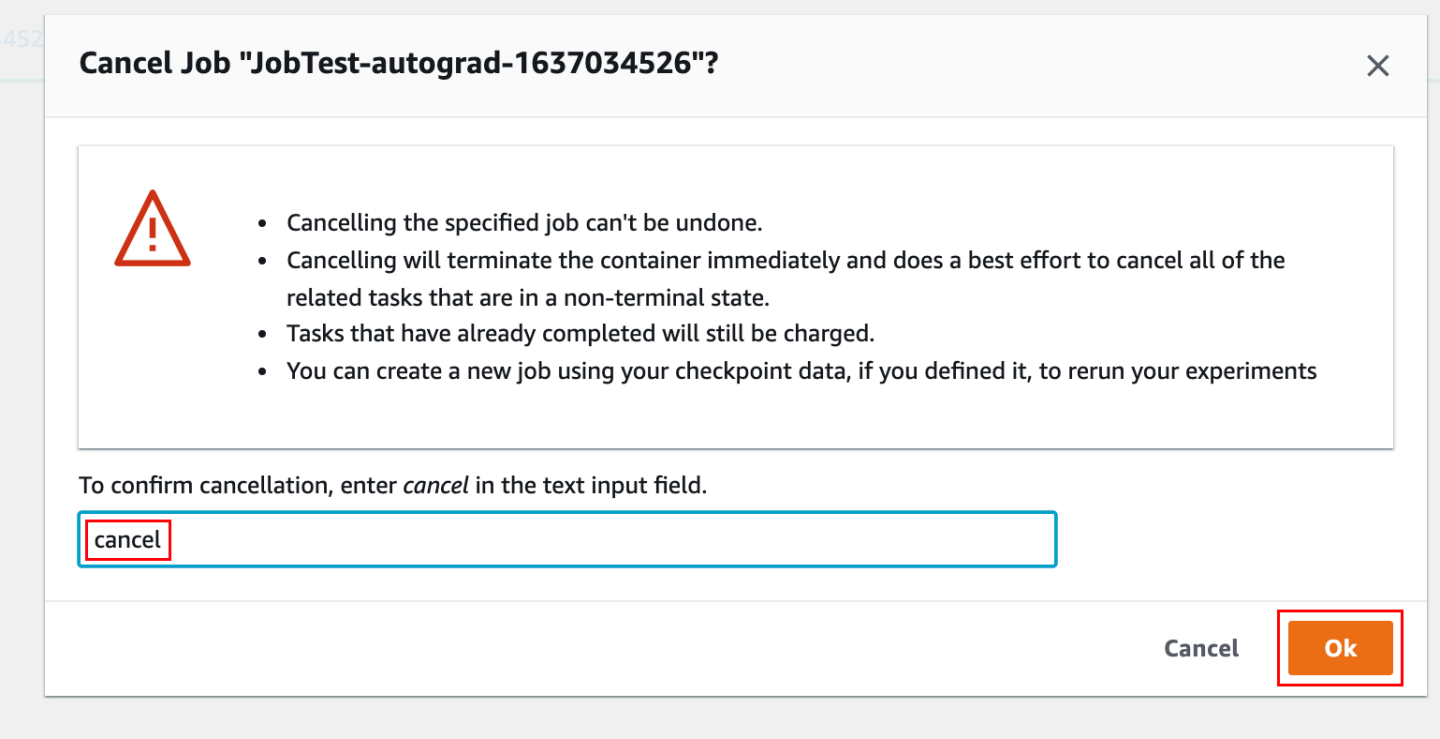
Annullare un Job ibrido

Potrebbe essere necessario annullare un processo ibrido in uno stato non terminale. Questa operazione può essere eseguita nella console o con il codice.

Per annullare il lavoro ibrido nella console, seleziona il lavoro ibrido da annullare dalla pagina Lavori ibridi, quindi seleziona Annulla lavoro ibrido dal menu a discesa Azioni.



Per confermare l'annullamento, inserisci cancel nel campo di immissione quando richiesto, quindi seleziona OK.



Per annullare il lavoro ibrido utilizzando il codice dell'SDK Braket Python, usa per identificare `job_arn` il lavoro ibrido e quindi chiama `cancel` il comando su di esso come mostrato nel codice seguente.

```
job = AwsQuantumJob(arn=job_arn)
job.cancel()
```

Il `cancel` comando termina immediatamente il classico contenitore di lavoro ibrido e fa del suo meglio per annullare tutte le attività quantistiche correlate che sono ancora in uno stato non terminale.

Personalizzazione del tuo Hybrid Job

Amazon Braket offre diversi modi per personalizzare il modo in cui vengono eseguiti i processi ibridi, consentendoti di adattare l'ambiente alle tue esigenze specifiche. Questa sezione esplora le opzioni per personalizzare i lavori ibridi, dalla definizione dell'ambiente di script dell'algoritmo alla creazione di un contenitore personalizzato. Imparerai come ottimizzare il flusso di lavoro utilizzando gli iperparametri, configurare le istanze di lavoro e sfruttare la compilazione parametrica per migliorare le prestazioni. Queste tecniche di personalizzazione ti aiutano a massimizzare il potenziale dei tuoi calcoli quantistici ibridi su Amazon Braket.

In questa sezione:

Definisci l'ambiente per lo script del tuo algoritmo

Amazon Braket supporta ambienti definiti da contenitori per lo script dell'algoritmo:

- Un contenitore di base (predefinito, se non `image_uri` è specificato)
- Un contenitore con CUDA-Q
- Un contenitore con Tensorflow e PennyLane
- Un contenitore con PyTorch, PennyLane e CUDA-Q

La tabella seguente fornisce dettagli sui contenitori e sulle librerie che includono.

Contenitori Amazon Braket

Tipo	Base	CUDA-Q	TensorFlow	PyTorch
URI dell'immagine	292282985366.dkr. ecr.us-west-2.amazonaws.com /:più recente amazon-braket-base-jobs	292282985366.dkr. ecr.us-west-2.amazonaws.com /:più recente amazon-braket-cudaq-jobs	292282985366.dkr. ecr.us-east-1.amazonaws.com /:più recente amazon-braket-tensorflow-jobs	292282985366.dkr. ecr.us-west-2.amazonaws.com /:più recente amazon-braket-pytorch-jobs
Librerie ereditate		• amazon-braket-default-simulator • amazon-braket-pennylane-plugin	• awscli • numpy • pandas • scipy	• awscli • numpy • pandas • scipy

Tipo	Base	CUDA-Q	TensorFlow	PyTorch
		• amazon-braket-schemas • amazon-braket-sdk • awscli • botocore • boto3 • dask • matplotlib • numpy • pandas • PennyLane • PennyLane-Fulmine • qiskit-braket-provider • richieste • formazione da saggista • scikit-learn • scipy		

Tipo	Base	CUDA-Q	TensorFlow	PyTorch
Librerie aggiuntive	• amazon-braket-default-simulator • amazon-braket-pennylane-plugin • amazon-braket-schemas • amazon-braket-sdk • awscli • boto3 • ipykernel • matplotlib • reti x • numpy • babele aperte • pandas • PennyLane • protobuf • psi 4 • RSA • scipy	• cudaq • cudaq-qec • risolutori cudaq	• amazon-braket-default-simulator • amazon-braket-pennylane-plugin • amazon-braket-schemas • amazon-braket-sdk • kernel ipy • keras • matplotlib • reti x • babele aperte • PennyLane • protobuf • psi 4 • RSA • PennyLane-GPU Lightning • CU Quantum	• amazon-braket-default-simulator • amazon-braket-pennylane-plugin • amazon-braket-schemas • amazon-braket-sdk • kernel ipy • keras • matplotlib • reti x • babele aperte • PennyLane • protobuf • psi 4 • RSA • PennyLane-GPU Lightning • CU Quantum • cudaq • cudaq-qec • risolutori cudaq

Puoi visualizzare e accedere alle definizioni dei contenitori open source su aws/. amazon-braket-containers Scegli il contenitore più adatto al tuo caso d'uso. Puoi utilizzare una qualsiasi delle AWS regioni disponibili in Braket (us-east-1, us-west-1, us-west-2, eu-north-1, eu-west-2), ma la regione

del contenitore deve corrispondere alla regione per il tuo lavoro ibrido. Specificate l'immagine del contenitore quando create un lavoro ibrido aggiungendo uno dei seguenti tre argomenti alla chiamata nello script del lavoro ibrido. `create(…)` Puoi installare dipendenze aggiuntive nel contenitore che scegli in fase di esecuzione (al costo dell'avvio o del runtime) perché i contenitori Amazon Braket dispongono di connettività Internet. L'esempio seguente si riferisce alla regione us-west-2.

- Immagine di base: image_uri="292282985366.dkr. ecr.us-west-2.amazonaws.com /:latest» amazon-braket-base-jobs
- Immagine CUDA-Q: image_uri="292282985366.dkr. ecr.us-west-2.amazonaws.com /:latest» amazon-braket-cudaq-jobs
- Immagine Tensorflow: image_uri="292282985366.dkr. ecr.us-west-2.amazonaws.com /:latest» amazon-braket-tensorflow-jobs
- PyTorch immagine: image_uri="292282985366.dkr. ecr.us-west-2.amazonaws.com /:latest» amazon-braket-pytorch-jobs

Possono `image-uris` anche essere recuperati utilizzando la funzione in Braket SDK. `retrieve_image()` Amazon L'esempio seguente mostra come recuperarli da Regione AWS us-west-2.

```
from braket.jobs.image_uris import retrieve_image, Framework

image_uri_base = retrieve_image(Framework.BASE, "us-west-2")
image_uri_cudaq = retrieve_image(Framework.CUDAQ, "us-west-2")
image_uri_tf = retrieve_image(Framework.PL_TENSORFLOW, "us-west-2")
image_uri_pytorch = retrieve_image(Framework.PL_PYTORCH, "us-west-2")
```

Importare un container personalizzato (Bring Your Own Container, BYOC)

Amazon Braket Hybrid Jobs offre tre contenitori predefiniti per l'esecuzione di codice in ambienti diversi. Se uno di questi contenitori supporta il tuo caso d'uso, devi fornire lo script dell'algoritmo solo quando crei un lavoro ibrido. Le dipendenze minori mancanti possono essere aggiunte dallo script dell'algoritmo o da un `requirements.txt` file utilizzando`pip`.

Se nessuno di questi contenitori supporta il tuo caso d'uso o se desideri ampliarli, Braket Hybrid Jobs supporta l'esecuzione di lavori ibridi con la tua immagine del Docker contenitore personalizzata o Bring your own container (BYOC). Assicurati che sia la funzionalità giusta per il tuo caso d'uso.

In questa sezione:

Quando portare il mio container è la decisione giusta?

Bringing your own container (BYOC) a Braket Hybrid Jobs offre la flessibilità di utilizzare il proprio software installandolo in un ambiente impacchettato. A seconda delle esigenze specifiche, potrebbero esserci modi per ottenere la stessa flessibilità senza dover passare attraverso la BYOC Docker build completa - caricamento Amazon ECR - ciclo URI dell'immagine personalizzata.

> **Note**
>
> BYOC potrebbe non essere la scelta giusta se si desidera aggiungere un numero limitato di pacchetti Python aggiuntivi (generalmente meno di 10) disponibili pubblicamente. Ad esempio, se stai usando. PyPi

In questo caso, puoi utilizzare una delle immagini Braket predefinite e quindi includere un `requirements.txt` file nella directory dei sorgenti al momento dell'invio del lavoro. Il file viene letto automaticamente e `pip` installerà i pacchetti con le versioni specificate come di consueto. Se state installando un gran numero di pacchetti, la durata dei vostri job potrebbe aumentare notevolmente. Controlla la versione Python e, se applicabile, CUDA del contenitore precostruito che desideri utilizzare per verificare se il tuo software funzionerà.

Il BYOC è necessario quando si desidera utilizzare un linguaggio non Python (come C++ o Rust) per lo script di lavoro o se si desidera utilizzare una versione Python non disponibile tramite i contenitori predefiniti di Braket. È anche una buona scelta se:

- Stai utilizzando un software con una chiave di licenza e devi autenticarla su un server di licenza per eseguire il software. Con BYOC, puoi incorporare la chiave di licenza nell'Dockerimmagine e includere il codice per autenticarla.
- Stai utilizzando un software che non è disponibile pubblicamente. Ad esempio, il software è ospitato su un archivio privato GitLab o su un GitHub repository a cui è necessaria una particolare chiave SSH per accedere.

- È necessario installare un'ampia suite di software che non sia inclusa nei contenitori forniti da Braket. Il BYOC ti consentirà di eliminare i lunghi tempi di avvio per i contenitori di lavori ibridi dovuti all'installazione del software.

BYOC consente inoltre di mettere a disposizione dei clienti il proprio SDK o algoritmo personalizzato creando un Docker contenitore con il software e rendendolo disponibile agli utenti. Puoi farlo impostando le autorizzazioni appropriate in Amazon ECR.

> **Note**
>
> È necessario rispettare tutte le licenze software applicabili.

Ricetta per portare il proprio contenitore

In questa sezione, forniamo una step-by-step guida su ciò che ti serve bring your own container (BYOC) per Braket Hybrid Jobs: gli script, i file e i passaggi per combinarli per iniziare a utilizzare le tue immagini personalizzateDocker. Le ricette per due casi comuni:

1. Installa software aggiuntivo in un'Dockerimmagine e usa solo script di algoritmi Python nei tuoi lavori.
2. Usa script di algoritmi scritti in un linguaggio non Python con Hybrid Jobs o un'architettura CPU diversa da x86.

La definizione dello script di immissione del contenitore è più complessa nel caso 2.

Quando Braket esegue il tuo Hybrid Job, avvia il numero e il tipo richiesti di istanze Amazon EC2, quindi esegue Docker l'immagine specificata dall'input URI dell'immagine per la creazione di job su di esse. Quando utilizzi la funzionalità BYOC, specifichi un URI di immagine ospitato in un repository Amazon ECR privato a cui hai accesso in lettura. Braket Hybrid Jobs utilizza quell'immagine personalizzata per eseguire il lavoro.

I componenti specifici necessari per creare un'Dockerimmagine che può essere utilizzata con Hybrid Jobs. Se non hai familiarità con la scrittura e la creazione`Dockerfiles`, fai riferimento alla documentazione e alla documentazione di Dockerfile. Amazon ECR CLI

Requisiti:

- Un'immagine di base per il tuo Dockerfile

- (Facoltativo) Uno script modificato per il punto di ingresso del contenitore
- Installa il software e lo script del contenitore necessari con Dockerfile

Un'immagine di base per il tuo Dockerfile

Se utilizzi Python e desideri installare software in aggiunta a quanto fornito nei contenitori forniti da Braket, un'opzione per un'immagine di base è una delle immagini del contenitore Braket, ospitate nel nostro GitHub repository e su Amazon ECR. Dovrai autenticarti su Amazon ECR per estrarre l'immagine e crearla su di essa. Ad esempio, la prima riga del tuo file BYOC potrebbe essereDocker: `FROM [IMAGE_URI_HERE]`

Quindi, compila il resto Dockerfile per installare e configurare il software che desideri aggiungere al contenitore. Le immagini Braket predefinite conterranno già lo script del punto di ingresso del contenitore appropriato, quindi non devi preoccuparti di includerlo.

Se vuoi usare un linguaggio non Python, come C++, Rust o Julia, o se vuoi creare un'immagine per un'architettura CPU non x86, come ARM, potresti dover creare su un'immagine pubblica barebone. Puoi trovare molte di queste immagini nella galleria pubblica di Amazon Elastic Container Registry. Assicurati di sceglierne una adatta all'architettura della CPU e, se necessario, alla GPU che desideri utilizzare.

(Facoltativo) Uno script modificato per il punto di ingresso del contenitore

> **Note**
>
> Se stai solo aggiungendo software aggiuntivo a un'immagine Braket predefinita, puoi saltare questa sezione.

Per eseguire codice non Python come parte del tuo lavoro ibrido, modifica lo script Python che definisce il punto di ingresso del contenitore. Ad esempio, lo script `braket_container.py` python su Amazon Braket Github. Questo è lo script che le immagini precreate da Braket utilizzano per avviare lo script dell'algoritmo e impostare le variabili di ambiente appropriate. Lo script del punto di ingresso del contenitore stesso deve essere in Python, ma può avviare script non Python. Nell'esempio predefinito, puoi vedere che gli script degli algoritmi Python vengono avviati come sottoprocesso Python o come processo completamente nuovo. Modificando questa logica, è possibile abilitare lo script del punto di ingresso per avviare script di algoritmi non Python. Ad esempio, è

possibile modificare la `thekick_off_customer_script()`funzione per avviare i processi Rust in base alla fine dell'estensione del file.

Puoi anche scegliere di scriverne uno completamente nuovo`braket_container.py`. Dovrebbe copiare i dati di input, gli archivi di origine e altri file necessari da Amazon S3 nel contenitore e definire le variabili di ambiente appropriate.

Installa il software e lo script del contenitore necessari con **`Dockerfile`**

> **Note**
>
> Se utilizzi un'immagine Braket predefinita come immagine di Docker base, lo script del contenitore è già presente.

Se hai creato uno script contenitore modificato nel passaggio precedente, dovrai copiarlo nel contenitore e definire la variabile di ambiente o il nome che hai dato `SAGEMAKER_PROGRAM` al `braket_container.py` nuovo script del punto di ingresso del contenitore.

Di seguito è riportato un esempio `Dockerfile` che consente di utilizzare Julia su istanze Jobs accelerate da GPU:

```
FROM nvidia/cuda:12.2.0-devel-ubuntu22.04


 ARG DEBIAN_FRONTEND=noninteractive
 ARG JULIA_RELEASE=1.8
 ARG JULIA_VERSION=1.8.3


 ARG PYTHON=python3.11
 ARG PYTHON_PIP=python3-pip
 ARG PIP=pip


 ARG JULIA_URL = https://julialang-s3.julialang.org/bin/linux/x64/${JULIA_RELEASE}/
 ARG TAR_NAME = julia-${JULIA_VERSION}-linux-x86_64.tar.gz


 ARG PYTHON_PKGS = # list your Python packages and versions here
```

```
RUN curl -s -L ${JULIA_URL}/${TAR_NAME} | tar -C /usr/local -x -z --strip-components=1
-f -


RUN apt-get update \

   && apt-get install -y --no-install-recommends \

   build-essential \

   tzdata \

   openssh-client \

   openssh-server \

   ca-certificates \

   curl \

   git \

   libtemplate-perl \

   libssl1.1 \

   openssl \

   unzip \

   wget \

   zlib1g-dev \

   ${PYTHON_PIP} \

   ${PYTHON}-dev \




RUN ${PIP} install --no-cache --upgrade ${PYTHON_PKGS}
```

```
 RUN ${PIP} install --no-cache --upgrade sagemaker-training==4.1.3


 # Add EFA and SMDDP to LD library path
 ENV LD_LIBRARY_PATH="/opt/conda/lib/python${PYTHON_SHORT_VERSION}/site-packages/
smdistributed/dataparallel/lib:$LD_LIBRARY_PATH"
 ENV LD_LIBRARY_PATH=/opt/amazon/efa/lib/:$LD_LIBRARY_PATH


 # Julia specific installation instructions
 COPY Project.toml /usr/local/share/julia/environments/v${JULIA_RELEASE}/
 RUN JULIA_DEPOT_PATH=/usr/local/share/julia \

    julia -e 'using Pkg; Pkg.instantiate(); Pkg.API.precompile()'
 # generate the device runtime library for all known and supported devices
 RUN JULIA_DEPOT_PATH=/usr/local/share/julia \

    julia -e 'using CUDA; CUDA.precompile_runtime()'


 # Open source compliance scripts
 RUN HOME_DIR=/root \

 && curl -o ${HOME_DIR}/oss_compliance.zip https://aws-dlinfra-
utilities.s3.amazonaws.com/oss_compliance.zip \

 && unzip ${HOME_DIR}/oss_compliance.zip -d ${HOME_DIR}/ \

 && cp ${HOME_DIR}/oss_compliance/test/testOSSCompliance /usr/local/bin/
testOSSCompliance \

 && chmod +x /usr/local/bin/testOSSCompliance \

 && chmod +x ${HOME_DIR}/oss_compliance/generate_oss_compliance.sh \

 && ${HOME_DIR}/oss_compliance/generate_oss_compliance.sh ${HOME_DIR} ${PYTHON} \

 && rm -rf ${HOME_DIR}/oss_compliance*


 # Copying the container entry point script
 COPY braket_container.py /opt/ml/code/braket_container.py
 ENV SAGEMAKER_PROGRAM braket_container.py
```

Questo esempio scarica ed esegue gli script forniti da per garantire la conformità con tutte le licenze AWS Open-Source pertinenti. Ad esempio, attribuendo correttamente qualsiasi codice installato governato da un. MIT license

Se devi includere codice non pubblico, ad esempio codice ospitato in un GitLab archivio GitHub o in un archivio privato, non incorporare le chiavi SSH nell'immagine per accedervi. Docker Utilizza invece Docker Compose when you build per consentire l'accesso Docker a SSH sulla macchina host su cui è costruito. Per maggiori informazioni, consulta la guida Uso sicuro delle chiavi SSH in Docker per accedere ai repository privati di Github.

DockerCreazione e caricamento della tua immagine

Una volta definito correttamente`Dockerfile`, sei pronto a seguire i passaggi per creare un repository Amazon ECR privato, se non ne esiste già uno. Puoi anche creare, etichettare e caricare l'immagine del contenitore nel repository.

Sei pronto per creare, etichettare e inserire l'immagine. Consulta la documentazione della build di Docker per una spiegazione completa delle opzioni `docker build` e alcuni esempi.

Per il file di esempio sopra definito, puoi eseguire:

```
aws ecr get-login-password --region ${your_region} | docker login --username AWS --
password-stdin ${aws_account_id}.dkr.ecr.${your_region}.amazonaws.com
 docker build -t braket-julia .
 docker tag braket-julia:latest ${aws_account_id}.dkr.ecr.${your_region}.amazonaws.com/
braket-julia:latest
 docker push ${aws_account_id}.dkr.ecr.${your_region}.amazonaws.com/braket-julia:latest
```

Assegnazione delle autorizzazioni Amazon ECR appropriate

Braket Hybrid Jobs Dockerle immagini devono essere ospitate in repository privati di Amazon ECR. Per impostazione predefinita, un repository Amazon ECR privato non fornisce l'accesso in lettura a Braket Hybrid Jobs IAM role o ad altri utenti che desiderano utilizzare la tua immagine, ad esempio un collaboratore o uno studente. È necessario impostare una politica di repository per concedere le autorizzazioni appropriate. In generale, concedi l'autorizzazione solo agli utenti e ai IAM ruoli specifici a cui desideri accedere alle tue immagini, anziché consentire a chiunque li possieda di image URI recuperarle.

Esecuzione dei job ibridi di Braket nel tuo contenitore

Per creare un lavoro ibrido con il tuo contenitore, chiama `AwsQuantumJob.create()` con l'argomento `image_uri` specificato. È possibile utilizzare una QPU, un simulatore on-demand o eseguire il codice localmente sul processore classico disponibile con Braket Hybrid Jobs. Ti consigliamo di testare il codice su un simulatore come SV1 DM1, o TN1 prima di eseguirlo su una vera QPU.

Per eseguire il codice sul processore classico, specifica `instanceType` and the `instanceCount` che usi aggiornando il. `InstanceConfig` Tieni presente che se specifichi un valore `instance_count` > 1, devi assicurarti che il codice possa essere eseguito su più host. Il limite massimo per il numero di istanze che puoi scegliere è 5. Esempio:

```
job = AwsQuantumJob.create(
    source_module="source_dir",
    entry_point="source_dir.algorithm_script:start_here",
    image_uri="111122223333.dkr.ecr.us-west-2.amazonaws.com/my-byoc-container:latest",
    instance_config=InstanceConfig(instanceType="ml.g4dn.xlarge", instanceCount=3),
    device="local:braket/braket.local.qubit",
    # ...)
```

> **Note**
>
> Usa l'ARN del dispositivo per tracciare il simulatore che hai usato come metadati di lavoro ibridi. I valori accettabili devono seguire il formato. `device = "local:<provider>/<simulator_name>"` Ricordatelo `<provider>` e `<simulator_name>` deve essere composto solo da lettere, numeri`_`,`-`, e`.`. La stringa è limitata a 256 caratteri.
> Se intendi utilizzare BYOC e non utilizzi l'SDK Braket per creare attività quantistiche, dovresti passare il valore della variabile ambientale `AMZN_BRAKET_JOB_TOKEN` al parametro nella richiesta. `jobToken` `CreateQuantumTask` In caso contrario, le attività quantistiche non hanno la priorità e vengono fatturate come normali attività quantistiche autonome.

Utilizzo degli iperparametri

È possibile definire gli iperparametri necessari all'algoritmo, come il tasso di apprendimento o la dimensione del passo, quando si crea un lavoro ibrido. I valori degli iperparametri vengono in genere utilizzati per controllare vari aspetti dell'algoritmo e spesso possono essere regolati per

ottimizzare le prestazioni dell'algoritmo. Per utilizzare gli iperparametri in un processo ibrido Braket, è necessario specificarne i nomi e i valori in modo esplicito come dizionario. Specificate i valori degli iperparametri da testare durante la ricerca del set di valori ottimale. Il primo passaggio per utilizzare gli iperparametri consiste nell'impostare e definire gli iperparametri come dizionario, come illustrato nel codice seguente.

```
from braket.devices import Devices

device_arn = Devices.Amazon.SV1

hyperparameters = {"shots": 1_000}
```

Quindi passate gli iperparametri definiti nel frammento di codice sopra riportato da utilizzare nell'algoritmo di vostra scelta. Per eseguire il seguente esempio di codice, create una directory denominata «src» nello stesso percorso del file di iperparametri. All'interno della directory «src», aggiungi i file di codice 0_getting_started_papermill.ipynb, notebook_runner.py e requirements.txt.

```
import time
from braket.aws import AwsQuantumJob

job = AwsQuantumJob.create(
    device=device_arn,
    source_module="src",
    entry_point="src.notebook_runner:run_notebook",
    input_data="src/0_Getting_started_papermill.ipynb",
    hyperparameters=hyperparameters,
    job_name=f"papermill-job-demo-{int(time.time())}",
)

# Print job to record the ARN
print(job)
```

Per accedere ai tuoi iperparametri dall'interno dello script di lavoro ibrido, consulta la funzione nel file notebook_runner.py python. *`load_jobs_hyperparams()`* Per accedere ai tuoi iperparametri al di fuori dello script di lavoro ibrido, esegui il codice seguente.

```
from braket.aws import AwsQuantumJob

# Get the job using the ARN
job_arn = "arn:aws:braket:us-east-1:111122223333:job/5eabb790-d3ff-47cc-98ed-
b4025e9e296f"  # Replace with your job ARN
```

```
job = AwsQuantumJob(arn=job_arn)

# Access the hyperparameters
job_metadata = job.metadata()
hyperparameters = job_metadata.get("hyperParameters", {})
print(hyperparameters)
```

Per ulteriori informazioni su come imparare a usare gli iperparametri, consulta i tutorial QAOA con Amazon Braket Hybrid Jobs PennyLane e Quantum machine learning nei tutorial Amazon Braket Hybrid Jobs.

Configura la tua istanza di lavoro ibrida

A seconda dell'algoritmo, potresti avere requisiti diversi. Per impostazione predefinita, Amazon Braket esegue lo script dell'algoritmo su un'`ml.m5.large`istanza. Tuttavia, puoi personalizzare questo tipo di istanza quando crei un lavoro ibrido utilizzando il seguente argomento di importazione e configurazione.

```
from braket.jobs.config import InstanceConfig

job = AwsQuantumJob.create(
    ...
    instance_config=InstanceConfig(instanceType="ml.g4dn.xlarge"), # Use NVIDIA T4
 instance with 4 GPUs.
    ...
    ),
```

Se state eseguendo una simulazione incorporata e avete specificato un dispositivo locale nella configurazione del dispositivo, potete inoltre richiedere più di un'istanza `InstanceConfig` specificando `instanceCount` e impostando che sia maggiore di una. Il limite massimo è 5. Ad esempio, puoi scegliere 3 istanze come segue.

```
from braket.jobs.config import InstanceConfig
job = AwsQuantumJob.create(
    ...
    instance_config=InstanceConfig(instanceType="ml.g4dn.xlarge", instanceCount=3), #
 Use 3 NVIDIA T4 instances
    ...
    ),
```

Quando utilizzi più istanze, valuta la possibilità di distribuire il job ibrido utilizzando la funzionalità data parallel. Consulta il seguente taccuino di esempio per maggiori dettagli su come vedere questo esempio di formazione su Parallelize per QML.

Le tre tabelle seguenti elencano i tipi di istanze e le specifiche disponibili per le istanze standard, ad alte prestazioni e con accelerazione GPU.

> **Note**
>
> Per visualizzare le quote predefinite delle istanze di calcolo classiche per Hybrid Jobs, consulta la pagina Amazon Braket Quotas.

Istanza	VPCU	Memoria (GiB)
ml.m5.large (impostazione predefinita)	4	16
ml.m5.xlarge	4	16
ml.m5.2xlarge	8	32
ml.m5.4xlarge	16	64
ml.m5.12xlarge	48	192
ml.m5.24xlarge	96	384

Istanze ad alte prestazioni	VPCU	Memoria (GiB)
ml.c5.xlarge	4	8
ml.c5.2xlarge	8	16
ml.c5.4xlarge	16	32
ml.c5.9xlarge	36	72
ml.c5.18xlarge	72	144

Istanze ad alte prestazioni	VPCU	Memoria (GiB)
ml.c5n.xlarge	4	10,5
ml.c5n.2xlarge	8	21
ml.c5n.4xlarge	16	32
ml.c5n.9xlarge	36	72
ml.c5n.18xlarge	72	192

Istanze con accelerazione GPU	GPUs	VPCU	Memoria (GiB)	Memoria GPU (GiB)
ml.p4d.24xlarge	8	96	1152	320
ml.g4dn.xlarge	1	4	16	16
ml.g4dn.2xlarge	1	8	32	16
ml.g4dn.4xlarge	1	16	64	16
ml.g4dn.8xlarge	1	32	128	16
ml.g4dn.12xlarge	4	48	192	64
ml.g4dn.16xlarge	1	64	256	16

Ogni istanza utilizza una configurazione predefinita di archiviazione dati (SSD) di 30 GB. È tuttavia possibile regolare lo spazio di archiviazione nello stesso modo in cui si configura il`instanceType`. L'esempio seguente mostra come aumentare lo spazio di archiviazione totale a 50 GB.

```
from braket.jobs.config import InstanceConfig

job = AwsQuantumJob.create(
    ...
    instance_config=InstanceConfig(
```

```
        instanceType="ml.g4dn.xlarge",
        volumeSizeInGb=50,
    ),
    ...
    ),
```

Configura il bucket predefinito in **`AwsSession`**

L'utilizzo della tua `AwsSession` istanza ti offre una maggiore flessibilità, come la possibilità di specificare una posizione personalizzata per il tuo bucket Amazon S3 predefinito. Per impostazione predefinita, an `AwsSession` ha una posizione del bucket Amazon S3 preconfigurata di. `"amazon-braket-{id}-{region}"` Tuttavia, hai la possibilità di sovrascrivere la posizione predefinita del bucket Amazon S3 durante la creazione di un. `AwsSession` Gli utenti possono facoltativamente passare un `AwsSession` oggetto al `AwsQuantumJob.create()` metodo, fornendo il `aws_session` parametro come illustrato nel seguente esempio di codice.

```
aws_session = AwsSession(default_bucket="amazon-braket-s3-demo-bucket")

# Then you can use that AwsSession when creating a hybrid job
job = AwsQuantumJob.create(
    ...
    aws_session=aws_session
)
```

Utilizzo della compilazione parametrica per velocizzare i lavori ibridi

Amazon Braket supporta la compilazione parametrica su alcuni. QPUs Ciò consente di ridurre il sovraccarico associato alla fase di compilazione, dal punto di vista computazionalmente costoso, compilando un circuito una sola volta e non per ogni iterazione del tuo algoritmo ibrido. Ciò può migliorare notevolmente i tempi di esecuzione per Hybrid Jobs, poiché si evita la necessità di ricompilare il circuito in ogni fase. Basta inviare circuiti parametrizzati a uno dei nostri Braket Hybrid QPUs Job supportati. Per lavori ibridi di lunga durata, Braket utilizza automaticamente i dati di calibrazione aggiornati del fornitore di hardware durante la compilazione del circuito per garantire risultati della massima qualità.

Per creare un circuito parametrico, devi prima fornire i parametri come input nello script dell'algoritmo. In questo esempio, utilizziamo un piccolo circuito parametrico e ignoriamo qualsiasi elaborazione classica tra ogni iterazione. Per i carichi di lavoro tipici, è necessario inviare molti circuiti in batch ed eseguire l'elaborazione classica, ad esempio l'aggiornamento dei parametri in ogni iterazione.

```
import os

from braket.aws import AwsDevice
from braket.circuits import Circuit, FreeParameter

def start_here():

    print("Test job started.")

    # Use the device declared in the job script
    device = AwsDevice(os.environ["AMZN_BRAKET_DEVICE_ARN"])

    circuit = Circuit().rx(0, FreeParameter("theta"))
    parameter_list = [0.1, 0.2, 0.3]

    for parameter in parameter_list:
        result = device.run(circuit, shots=1000, inputs={"theta": parameter})

    print("Test job completed.")
```

È possibile inviare lo script dell'algoritmo per l'esecuzione come Hybrid Job con il seguente script di processo. Quando si esegue Hybrid Job su una QPU che supporta la compilazione parametrica, il circuito viene compilato solo alla prima esecuzione. Nelle esecuzioni successive, il circuito compilato viene riutilizzato, aumentando le prestazioni di runtime di Hybrid Job senza righe di codice aggiuntive.

```
from braket.aws import AwsQuantumJob

job = AwsQuantumJob.create(
    device=device_arn,
    source_module="algorithm_script.py",
)
```

> **Note**
>
> La compilazione parametrica è supportata su tutti i moduli superconduttori basati su gate, ad eccezione dei programmi a livello di QPUs Rigetti Computing impulsi.

Utilizzo PennyLane con Amazon Braket

Gli algoritmi ibridi sono algoritmi che contengono istruzioni sia classiche che quantistiche. Le istruzioni classiche vengono eseguite su hardware classico (un'istanza EC2 o un laptop) e le istruzioni quantistiche vengono eseguite su un simulatore o su un computer quantistico. Ti consigliamo di eseguire algoritmi ibridi utilizzando la funzionalità Hybrid Jobs. Per ulteriori informazioni, consulta Quando usare Amazon Braket Jobs.

Amazon Braket ti consente di configurare ed eseguire algoritmi quantistici ibridi con l'assistenza del PennyLane plug-in Amazon Braket o con l'SDK Amazon Braket Python e repository di notebook di esempio. I notebook di esempio Amazon Braket, basati sull'SDK, consentono di configurare ed eseguire determinati algoritmi ibridi senza il plug-in. PennyLane Tuttavia, lo consigliamo PennyLane perché offre un'esperienza più ricca.

Informazioni sugli algoritmi quantistici ibridi

Gli algoritmi quantistici ibridi sono importanti per il settore odierno perché i dispositivi informatici quantistici contemporanei generalmente producono rumore e quindi errori. Ogni porta quantistica aggiunta a un calcolo aumenta la possibilità di aggiungere rumore; pertanto, gli algoritmi di lunga durata possono essere sopraffatti dal rumore, con conseguenti errori di calcolo.

Algoritmi quantistici puri come quelli di Shor (esempio Quantum Phase Estimation) o Grover (esempio Grover) richiedono migliaia o milioni di operazioni. Per questo motivo, possono essere poco pratici per i dispositivi quantistici esistenti, che vengono generalmente definiti dispositivi quantistici rumorosi su scala intermedia (NISQ).

Negli algoritmi quantistici ibridi, le unità di elaborazione quantistica (QPUs) funzionano come coprocessori per la versione classica, in particolare per velocizzare determinati calcoli in un algoritmo classico. CPUs Le esecuzioni dei circuiti diventano molto più brevi, alla portata delle funzionalità dei dispositivi odierni.

In questa sezione:

- Amazon Braket con PennyLane
- Algoritmi ibridi nei notebook di esempio Amazon Braket
- Algoritmi ibridi con simulatori integrati PennyLane
- Aggiungi il gradiente con i simulatori Amazon PennyLane Braket
- PennyLane Utilizzo di Hybrid Jobs ed esecuzione di un algoritmo QAOA

- Esegui carichi di lavoro ibridi con simulatori integrati PennyLane

Amazon Braket con PennyLane

Amazon Braket fornisce supporto per PennyLaneun framework software open source basato sul concetto di programmazione quantistica differenziabile. Puoi usare questo framework per addestrare i circuiti quantistici nello stesso modo in cui addestreresti una rete neurale per trovare soluzioni a problemi computazionali di chimica quantistica, apprendimento automatico quantistico e ottimizzazione.

La PennyLane libreria fornisce interfacce a strumenti di apprendimento automatico familiari, tra cui PyTorch e, per rendere l'addestramento dei circuiti quantistici TensorFlow rapido e intuitivo.

- La PennyLane libreria -— PennyLane è preinstallata nei notebook Braket. Amazon Per accedere ai dispositivi Amazon Braket da PennyLane, apri un notebook e importa la libreria con il PennyLane seguente comando.

```
import pennylane as qml
```

I taccuini tutorial ti aiutano a iniziare rapidamente. In alternativa, puoi utilizzarlo PennyLane su Amazon Braket da un IDE a tua scelta.

- Il PennyLane plug-in Amazon Braket: per utilizzare il tuo IDE, puoi installare il plug-in Amazon Braket PennyLane manualmente. Il plug-in si connette PennyLane all'SDK Amazon Braket Python, in modo da poter eseguire circuiti sui dispositivi Braket. PennyLane Amazon Per installare il PennyLane plugin, usa il seguente comando.

```
pip install amazon-braket-pennylane-plugin
```

L'esempio seguente mostra come configurare l'accesso ai dispositivi Amazon Braket in: PennyLane

```
# to use SV1
import pennylane as qml
sv1 = qml.device("braket.aws.qubit", device_arn="arn:aws:braket:::device/quantum-
simulator/amazon/sv1", wires=2)

# to run a circuit:
```

```
@qml.qnode(sv1)
def circuit(x):
    qml.RZ(x, wires=0)
    qml.CNOT(wires=[0,1])
    qml.RY(x, wires=1)
    return qml.expval(qml.PauliZ(1))

result = circuit(0.543)


#To use the local sim:
local = qml.device("braket.local.qubit", wires=2)
```

Per esempi di tutorial e ulteriori informazioni a riguardo PennyLane, consulta l'archivio degli esempi di Amazon Braket.

Il PennyLane plug-in Amazon Braket ti consente di passare da Amazon Braket QPU a dispositivi di simulazione integrati PennyLane con una sola riga di codice. Offre due dispositivi quantistici Amazon Braket con cui lavorare: PennyLane

- `braket.aws.qubit`per funzionare con i dispositivi quantistici del servizio Amazon Braket, inclusi i simulatori QPUs
- `braket.local.qubit`per funzionare con il simulatore locale di Amazon Braket SDK

Il PennyLane plugin Amazon Braket è open source. Puoi installarlo dal GitHub repository dei PennyLane plugin.

Per ulteriori informazioni in merito PennyLane, consulta la documentazione sul PennyLane sito Web.

Algoritmi ibridi nei notebook di esempio Amazon Braket

Amazon Braket fornisce una serie di notebook di esempio che non si basano sul PennyLane plug-in per l'esecuzione di algoritmi ibridi. Puoi iniziare con uno qualsiasi di questi notebook di esempio ibridi Amazon Braket che illustrano metodi variazionali, come Quantum Approximate Optimization Algorithm (QAOA) o Variational Quantum Eigensolver (VQE).

I notebook di esempio Amazon Braket si basano sull'SDK Amazon Braket Python. L'SDK fornisce un framework per interagire con i dispositivi hardware di calcolo quantistico tramite Braket. Amazon È una libreria open source progettata per assistervi nella parte quantistica del vostro flusso di lavoro ibrido.

Puoi esplorare ulteriormente Amazon Braket con i nostri taccuini di esempio.

Algoritmi ibridi con simulatori integrati PennyLane

Amazon Braket Hybrid Jobs ora include simulatori integrati ad alte prestazioni basati su CPU e GPU di. PennyLane Questa famiglia di simulatori integrati può essere incorporata direttamente nel tuo contenitore di lavori ibridi e include il veloce simulatore state-vector, il `lightning.qubit` `lightning.gpu` simulatore accelerato utilizzando la libreria cuQuantum di NVIDIA e altri. Questi simulatori integrati sono ideali per algoritmi variazionali come l'apprendimento automatico quantistico, che possono trarre vantaggio da metodi avanzati come il metodo di differenziazione adjoint. È possibile eseguire questi simulatori incorporati su una o più istanze di CPU o GPU.

Con Hybrid Jobs, ora puoi eseguire il codice dell'algoritmo variazionale utilizzando una combinazione di un coprocessore classico e una QPU, un simulatore on-demand Amazon Braket comeSV1, o utilizzando direttamente il simulatore incorporato di. PennyLane

Il simulatore incorporato è già disponibile con il contenitore Hybrid Jobs, devi decorare la tua funzione Python principale con `@hybrid_job` il decoratore. Per utilizzare il PennyLane `lightning.gpu` simulatore, è inoltre necessario specificare un'istanza GPU `InstanceConfig` come mostrato nel seguente frammento di codice:

```
import pennylane as qml
from braket.jobs import hybrid_job
from braket.jobs.config import InstanceConfig


@hybrid_job(device="local:pennylane/lightning.gpu",
 instance_config=InstanceConfig(instanceType="ml.g4dn.xlarge"))
def function(wires):
    dev = qml.device("lightning.gpu", wires=wires)
    ...
```

Fate riferimento al notebook di esempio per iniziare a utilizzare un simulatore PennyLane incorporato con Hybrid Jobs.

Aggiungi il gradiente con i simulatori Amazon PennyLane Braket

Con il PennyLane plug-in per Amazon Braket, puoi calcolare i gradienti utilizzando il metodo di differenziazione adjoint quando esegui sul simulatore vettoriale statale locale o. SV1

Nota: per utilizzare il metodo di differenziazione aggiuntiva, devi specificare nel tuo, e non. **`diff_method='device'` `qnode`** `diff_method='adjoint'` Guarda l'esempio seguente.

```
device_arn = "arn:aws:braket:::device/quantum-simulator/amazon/sv1"
dev = qml.device("braket.aws.qubit", wires=wires, shots=0, device_arn=device_arn)

@qml.qnode(dev, diff_method="device")
def cost_function(params):
    circuit(params)
    return qml.expval(cost_h)

gradient = qml.grad(circuit)
initial_gradient = gradient(params0)
```

> **Note**
>
> Attualmente, PennyLane calcolerà gli indici di raggruppamento per gli hamiltoniani di QAOA e li utilizzerà per dividere l'hamiltoniano in più valori di aspettativa. Se desideri utilizzare la funzionalità di differenziazione aggiuntiva SV1 di QAOA da cui esegui QAOA, dovrai ricostruire il costo hamiltoniano rimuovendo gli indici di raggruppamentoPennyLane, in questo modo: `cost_h, mixer_h = qml.qaoa.max_clique(g, constrained=False)` `cost_h = qml.Hamiltonian(cost_h.coeffs, cost_h.ops)`

PennyLane Utilizzo di Hybrid Jobs ed esecuzione di un algoritmo QAOA

In questa sezione, userai ciò che hai imparato per scrivere un vero programma ibrido utilizzando PennyLane la compilazione parametrica. Lo script dell'algoritmo viene utilizzato per risolvere un problema relativo al Quantum Approximate Optimization Algorithm (QAOA). Il programma crea una funzione di costo corrispondente a un classico problema di ottimizzazione Max Cut, specifica un circuito quantistico parametrizzato e utilizza un metodo di discesa del gradiente per ottimizzare i parametri in modo da ridurre al minimo la funzione di costo. In questo esempio, generiamo il grafico del problema nello script dell'algoritmo per semplicità, ma per i casi d'uso più tipici la migliore pratica consiste nel fornire le specifiche del problema attraverso un canale dedicato nella configurazione dei dati di input. L'`parametrize_differentiable`impostazione predefinita del flag consente di `True` ottenere automaticamente i vantaggi di prestazioni di runtime migliorate grazie alla compilazione parametrica su Support. QPUs

```
import os
```

```
import json
import time

from braket.jobs import save_job_result
from braket.jobs.metrics import log_metric

import networkx as nx
import pennylane as qml
from pennylane import numpy as np
from matplotlib import pyplot as plt

def init_pl_device(device_arn, num_nodes, shots, max_parallel):
    return qml.device(
        "braket.aws.qubit",
        device_arn=device_arn,
        wires=num_nodes,
        shots=shots,
        # Set s3_destination_folder=None to output task results to a default folder
        s3_destination_folder=None,
        parallel=True,
        max_parallel=max_parallel,
        parametrize_differentiable=True, # This flag is True by default.
    )

def start_here():
    input_dir = os.environ["AMZN_BRAKET_INPUT_DIR"]
    output_dir = os.environ["AMZN_BRAKET_JOB_RESULTS_DIR"]
    job_name = os.environ["AMZN_BRAKET_JOB_NAME"]
    checkpoint_dir = os.environ["AMZN_BRAKET_CHECKPOINT_DIR"]
    hp_file = os.environ["AMZN_BRAKET_HP_FILE"]
    device_arn = os.environ["AMZN_BRAKET_DEVICE_ARN"]

    # Read the hyperparameters
    with open(hp_file, "r") as f:
        hyperparams = json.load(f)

    p = int(hyperparams["p"])
    seed = int(hyperparams["seed"])
    max_parallel = int(hyperparams["max_parallel"])
    num_iterations = int(hyperparams["num_iterations"])
    stepsize = float(hyperparams["stepsize"])
    shots = int(hyperparams["shots"])

    # Generate random graph
```

```
    num_nodes = 6
    num_edges = 8
    graph_seed = 1967
    g = nx.gnm_random_graph(num_nodes, num_edges, seed=graph_seed)

    # Output figure to file
    positions = nx.spring_layout(g, seed=seed)
    nx.draw(g, with_labels=True, pos=positions, node_size=600)
    plt.savefig(f"{output_dir}/graph.png")

    # Set up the QAOA problem
    cost_h, mixer_h = qml.qaoa.maxcut(g)

    def qaoa_layer(gamma, alpha):
        qml.qaoa.cost_layer(gamma, cost_h)
        qml.qaoa.mixer_layer(alpha, mixer_h)

    def circuit(params, **kwargs):
        for i in range(num_nodes):
            qml.Hadamard(wires=i)
        qml.layer(qaoa_layer, p, params[0], params[1])

    dev = init_pl_device(device_arn, num_nodes, shots, max_parallel)

    np.random.seed(seed)
    cost_function = qml.ExpvalCost(circuit, cost_h, dev, optimize=True)
    params = 0.01 * np.random.uniform(size=[2, p])

    optimizer = qml.GradientDescentOptimizer(stepsize=stepsize)
    print("Optimization start")

    for iteration in range(num_iterations):
        t0 = time.time()

        # Evaluates the cost, then does a gradient step to new params
        params, cost_before = optimizer.step_and_cost(cost_function, params)
        # Convert cost_before to a float so it's easier to handle
        cost_before = float(cost_before)

        t1 = time.time()

        if iteration == 0:
            print("Initial cost:", cost_before)
        else:
```

```
            print(f"Cost at step {iteration}:", cost_before)

        # Log the current loss as a metric
        log_metric(
            metric_name="Cost",
            value=cost_before,
            iteration_number=iteration,
        )

        print(f"Completed iteration {iteration + 1}")
        print(f"Time to complete iteration: {t1 - t0} seconds")

    final_cost = float(cost_function(params))
    log_metric(
        metric_name="Cost",
        value=final_cost,
        iteration_number=num_iterations,
    )

    # We're done with the hybrid job, so save the result.
    # This will be returned in job.result()
    save_job_result({"params": params.numpy().tolist(), "cost": final_cost})
```

> **Note**
>
> La compilazione parametrica è supportata su tutti i formati superconduttori basati su gate QPUs , ad eccezione dei programmi a livello di impulsi. Rigetti Computing

Esegui carichi di lavoro ibridi con simulatori integrati PennyLane

Vediamo come utilizzare i simulatori integrati di Amazon Braket Hybrid Jobs per eseguire carichi di lavoro ibridi. PennyLane Il simulatore integrato basato su GPU di Pennylane utilizza la libreria Nvidia CuQuantum per accelerare `lightning.gpu` le simulazioni di circuiti. Il simulatore GPU integrato è preconfigurato in tutti i contenitori di lavoro Braket che gli utenti possono utilizzare immediatamente. In questa pagina, ti mostriamo come utilizzarlo per velocizzare i carichi `lightning.gpu` di lavoro ibridi.

Utilizzo **`lightning.gpu`** per carichi di lavoro QAOA

Considerate gli esempi di Quantum Approximate Optimization Algorithm (QAOA) tratti da questo taccuino. Per selezionare un simulatore incorporato, si specifica che l'`device`argomento sia una stringa del formato:. `"local:<provider>/<simulator_name>"` Ad esempio, imposteresti `"local:pennylane/lightning.gpu"` per`lightning.gpu`. La stringa del dispositivo fornita a Hybrid Job all'avvio viene passata al lavoro come variabile di ambiente`"AMZN_BRAKET_DEVICE_ARN"`.

```
device_string = os.environ["AMZN_BRAKET_DEVICE_ARN"]
prefix, device_name = device_string.split("/")
device = qml.device(simulator_name, wires=n_wires)
```

In questa pagina, confronta i due simulatori vettoriali di PennyLane stato incorporati `lightning.qubit` (basati su CPU) e `lightning.gpu` (basati su GPU). Fornisci ai simulatori scomposizioni di gate personalizzate per calcolare vari gradienti.

Ora sei pronto per preparare lo script ibrido di avvio del lavoro. Esegui l'algoritmo QAOA utilizzando due tipi di istanza: e. `ml.m5.2xlarge` `ml.g4dn.xlarge` Il tipo di `ml.m5.2xlarge` istanza è paragonabile a un laptop standard per sviluppatori. `ml.g4dn.xlarge`Si tratta di un'istanza di elaborazione accelerata con una singola GPU NVIDIA T4 con 16 GB di memoria.

Per eseguire la GPU, dobbiamo prima specificare un'immagine compatibile e l'istanza corretta (che per impostazione predefinita è un'istanza). `ml.m5.2xlarge`

```
from braket.aws import AwsSession
from braket.jobs.image_uris import Framework, retrieve_image

image_uri = retrieve_image(Framework.PL_PYTORCH, AwsSession().region)
instance_config = InstanceConfig(instanceType="ml.g4dn.xlarge")
```

Dobbiamo quindi inserirli nell'hybrid job decorator, insieme ai parametri del dispositivo aggiornati sia nel sistema che negli argomenti del lavoro ibrido.

```
@hybrid_job(
        device="local:pennylane/lightning.gpu",
        input_data=input_file_path,
        image_uri=image_uri,
        instance_config=instance_config)
```

```
def run_qaoa_hybrid_job_gpu(p=1, steps=10):
    params = np.random.rand(2, p)

    braket_task_tracker = Tracker()

    graph = nx.read_adjlist(input_file_path, nodetype=int)
    wires = list(graph.nodes)
    cost_h, _mixer_h = qaoa.maxcut(graph)

    device_string = os.environ["AMZN_BRAKET_DEVICE_ARN"]
    prefix, device_name = device_string.split("/")
    dev= qml.device(simulator_name, wires=len(wires))
    ...
```

> **Note**
>
> Se si specifica l'`instance_config`as utilizzando un'istanza basata su GPU, ma si sceglie come simulatore basato su CPU incorporato (`lightning.qubit`), la GPU non verrà utilizzata. `device` Assicurati di utilizzare il simulatore integrato basato su GPU se desideri utilizzare come target la GPU!

Il tempo medio di iterazione per l'`m5.2xlarge`istanza è di circa 73 secondi, mentre per l'`ml.g4dn.xlarge`istanza è di circa 0,6 secondi. Per questo flusso di lavoro da 21 qubit, l'istanza GPU ci offre una velocità di 100 volte superiore. Se guardi la pagina dei prezzi di Amazon Braket Hybrid Jobs, puoi vedere che il costo al minuto per un'`m5.2xlarge`istanza è di 0,00768 USD, mentre per l'`ml.g4dn.xlarge`istanza è di 0,01227 USD. In questo caso è più veloce ed economico eseguire l'istanza GPU.

Apprendimento automatico quantistico e parallelismo dei dati

Se il tuo tipo di carico di lavoro è l'apprendimento automatico quantistico (QML) che si addestra su set di dati, puoi accelerare ulteriormente il carico di lavoro utilizzando il parallelismo dei dati. In QML, il modello contiene uno o più circuiti quantistici. Il modello può contenere o meno anche reti neurali classiche. Quando si addestra il modello con il set di dati, i parametri del modello vengono aggiornati per ridurre al minimo la funzione di perdita. Di solito viene definita una funzione di perdita per un singolo punto dati e la perdita totale per la perdita media sull'intero set di dati. In QML, le perdite vengono generalmente calcolate in serie prima di calcolare la media della perdita totale per i calcoli a gradiente. Questa procedura richiede molto tempo, soprattutto quando ci sono centinaia di punti dati.

Poiché la perdita da un punto dati non dipende da altri punti dati, le perdite possono essere valutate in parallelo! Le perdite e i gradienti associati a diversi punti dati possono essere valutati contemporaneamente. Questo è noto come parallelismo dei dati. Con SageMaker la libreria parallela di dati distribuiti, Amazon Braket Hybrid Jobs semplifica l'utilizzo del parallelismo dei dati per accelerare la formazione.

Considera il seguente carico di lavoro QML per il parallelismo dei dati, che utilizza il set di dati Sonar del noto repository UCI come esempio di classificazione binaria. Il set di dati Sonar ha 208 punti dati ciascuno con 60 caratteristiche raccolte dai segnali sonar che rimbalzano sui materiali. Ogni punto dati è etichettato come «M» per le miniere o «R» per le rocce. Il nostro modello QML è costituito da un livello di input, un circuito quantistico come livello nascosto e un livello di output. I livelli di input e output sono reti neurali classiche implementate in. PyTorch Il circuito quantistico è integrato con le reti PyTorch neurali utilizzando il modulo qml.qnn. PennyLane Consulta i nostri taccuini di esempio per maggiori dettagli sul carico di lavoro. Come nell'esempio QAOA riportato sopra, puoi sfruttare la potenza della GPU utilizzando simulatori integrati basati su GPU come quelli per migliorare le prestazioni rispetto ai simulatori integrati basati su CPU. PennyLane `lightning.gpu`

Per creare un lavoro ibrido, puoi chiamare `AwsQuantumJob.create` e specificare lo script dell'algoritmo, il dispositivo e altre configurazioni tramite gli argomenti delle parole chiave.

```
instance_config = InstanceConfig(instanceType='ml.g4dn.xlarge')

hyperparameters={"nwires": "10",
                 "ndata": "32",
                 ...
}

job = AwsQuantumJob.create(
    device="local:pennylane/lightning.gpu",
    source_module="qml_source",
    entry_point="qml_source.train_single",
    hyperparameters=hyperparameters,
    instance_config=instance_config,
    ...
)
```

Per utilizzare il parallelismo dei dati, è necessario modificare alcune righe di codice nello script dell'algoritmo per la libreria SageMaker distribuita per parallelizzare correttamente l'addestramento. Innanzitutto, importate il `smdistributed` pacchetto che svolge la maggior parte del lavoro necessario per distribuire i carichi di lavoro su più e più istanze. GPUs Questo pacchetto è

preconfigurato nel Braket e nei contenitori. PyTorch TensorFlow Il `dist` modulo indica al nostro script di algoritmo qual è il numero totale di elementi GPUs per il training (`world_size`) `rank` e la fine `local_rank` di un core della GPU. `rank`è l'indice assoluto di una GPU in tutte le istanze, mentre `local_rank` è l'indice di una GPU all'interno di un'istanza. Ad esempio, se ci sono quattro istanze ciascuna di cui otto GPUs allocate per l'addestramento, i `rank` valori vanno da 0 a 31 e quelli da 0 a 7`local_rank`.

```
import smdistributed.dataparallel.torch.distributed as dist

dp_info = {
    "world_size": dist.get_world_size(),
    "rank": dist.get_rank(),
    "local_rank": dist.get_local_rank(),
}
batch_size //= dp_info["world_size"] // 8
batch_size = max(batch_size, 1)
```

Successivamente, si definisce un `DistributedSampler` in base a `world_size` `rank` e quindi lo si passa nel caricatore di dati. Questo campionatore evita di GPUs accedere alla stessa porzione di un set di dati.

```
train_sampler = torch.utils.data.distributed.DistributedSampler(
    train_dataset,
    num_replicas=dp_info["world_size"],
    rank=dp_info["rank"]
)
train_loader = torch.utils.data.DataLoader(
    train_dataset,
    batch_size=batch_size,
    shuffle=False,
    num_workers=0,
    pin_memory=True,
    sampler=train_sampler,
)
```

Successivamente, si utilizza la `DistributedDataParallel` classe per abilitare il parallelismo dei dati.

```
from smdistributed.dataparallel.torch.parallel.distributed import
 DistributedDataParallel as DDP
```

```
model = DressedQNN(qc_dev).to(device)
model = DDP(model)
torch.cuda.set_device(dp_info["local_rank"])
model.cuda(dp_info["local_rank"])
```

Quanto sopra sono le modifiche necessarie per utilizzare il parallelismo dei dati. In QML, spesso si desidera salvare i risultati e stampare i progressi della formazione. Se ogni GPU esegue il comando di salvataggio e stampa, il registro verrà invaso dalle informazioni ripetute e i risultati si sovrascriveranno a vicenda. Per evitare ciò, è possibile salvare e stampare solo dalla GPU con 0. `rank`

```
if dp_info["rank"]==0:
    print('elapsed time: ', elapsed)
    torch.save(model.state_dict(), f"{output_dir}/test_local.pt")
    save_job_result({"last loss": loss_before})
```

Amazon Braket Hybrid Jobs supporta i tipi di `ml.g4dn.12xlarge` istanza per la libreria parallela di SageMaker dati distribuiti. Puoi configurare il tipo di istanza tramite l'`InstanceConfig`argomento in Hybrid Jobs. Affinché la libreria parallela di dati SageMaker distribuiti sappia che il parallelismo dei dati è abilitato, devi aggiungere due iperparametri aggiuntivi, `"sagemaker_distributed_dataparallel_enabled"` impostandoli `"true"` e `"sagemaker_instance_type"` impostandoli sul tipo di istanza che stai utilizzando. Questi due iperparametri vengono utilizzati per pacchetto. `smdistributed` Lo script dell'algoritmo non deve utilizzarli in modo esplicito. In Amazon Braket SDK, fornisce un comodo argomento per le parole chiave. `distribution` Per quanto `distribution="data_parallel"` riguarda la creazione di posti di lavoro ibridi, l'SDK Amazon Braket inserisce automaticamente i due iperparametri per te. Se utilizzi l'API Amazon Braket, devi includere questi due iperparametri.

Con il parallelismo di istanze e dati configurato, ora puoi inviare il tuo lavoro ibrido. Ce ne sono 4 GPUs in un'`ml.g4dn.12xlarge`istanza. Quando si imposta`instanceCount=1`, il carico di lavoro viene distribuito tra gli 8 membri GPUs dell'istanza. Se ne imposti `instanceCount` più di uno, il carico di lavoro viene distribuito tra quelli GPUs disponibili in tutte le istanze. Quando si utilizzano più istanze, ogni istanza comporta un addebito in base al tempo di utilizzo. Ad esempio, quando utilizzi quattro istanze, il tempo fatturabile è quattro volte il tempo di esecuzione per istanza perché ci sono quattro istanze che eseguono i tuoi carichi di lavoro contemporaneamente.

```
instance_config = InstanceConfig(instanceType='ml.g4dn.12xlarge',
                                 instanceCount=1,
```

```
)

hyperparameters={"nwires": "10",
                 "ndata": "32",
                 ...,
}

job = AwsQuantumJob.create(
    device="local:pennylane/lightning.gpu",
    source_module="qml_source",
    entry_point="qml_source.train_dp",
    hyperparameters=hyperparameters,
    instance_config=instance_config,
    distribution="data_parallel",
    ...
)
```

> **Note**
>
> Nella creazione di posti di lavoro ibridi di cui sopra, `train_dp.py` si tratta dello script algoritmico modificato per l'utilizzo del parallelismo dei dati. Tieni presente che il parallelismo dei dati funziona correttamente solo quando modifichi lo script dell'algoritmo in base alla sezione precedente. Se l'opzione di parallelismo dei dati è abilitata senza uno script di algoritmo modificato correttamente, il processo ibrido può generare errori o ogni GPU può elaborare ripetutamente la stessa porzione di dati, il che è inefficiente.

Se usato correttamente, l'utilizzo di più istanze può portare a una riduzione di ordini di grandezza in termini di tempi e costi. Vedi il taccuino di esempio per maggiori dettagli.

Utilizzo di CUDA-Q con Amazon Braket

NVIDIA's CUDA-Qè una libreria software progettata per la programmazione di algoritmi quantistici ibridi che combinano CPUs unità di elaborazione quantistiche e quantistiche (). GPUs QPUs Fornisce un modello di programmazione unificato, che consente agli sviluppatori di esprimere istruzioni classiche e quantistiche all'interno di un unico programma, semplificando i flussi di lavoro. CUDA-Qaccelera la simulazione e il runtime quantistici dei programmi grazie ai simulatori di CPU e GPU integrati. CUDA-Qè disponibile con le istanze native per notebook Braket (NBIs) e Amazon Braket Hybrid Jobs.

In questa sezione:

CUDA-Q in NBIs

CUDA-Qè installato per impostazione predefinita nell'ambiente Braket NBI. È possibile aprire un notebook di CUDA-Q esempio accedendo alla pagina di avvio di Jupyter e selezionando il riquadro and Braket. CUDA-Q Questo apre il taccuino `0_Getting_started_with_CUDA-Q.ipynb` di esempio nella finestra principale. Per altri CUDA-Q esempi, consultate il pannello a sinistra nella `nvidia_cuda_q/` directory.

Puoi anche verificare la versione CUDA-Q o qualsiasi altro pacchetto di terze parti installato nel tuo NBI. Ad esempio, è possibile eseguire il comando seguente in una cella di codice di un notebook per verificare le versioni dei CUDA-Q pacchetti Qiskit e Braket installate nell'ambiente. PennyLane

```
%pip freeze | grep -i -e cudaq -e qiskit -e pennylane -e braket
```

CUDA-Q nei lavori ibridi

L'utilizzo CUDA-Q su Amazon Braket Hybrid Jobs offre un ambiente di elaborazione flessibile e on-demand. Le istanze computazionali vengono eseguite solo per la durata del carico di lavoro, assicurandoti di pagare solo per ciò che usi. Amazon Braket Hybrid Jobs offre anche un'esperienza scalabile. Gli utenti possono iniziare con istanze più piccole per la prototipazione e il test, quindi passare a istanze più grandi in grado di gestire carichi di lavoro maggiori per esperimenti completi.

Il supporto di Amazon Braket Hybrid Jobs GPUs è essenziale per massimizzare CUDA-Q il potenziale di Amazon Braket. GPUs velocizza notevolmente le simulazioni di programmi quantistici rispetto ai simulatori basati su CPU, specialmente quando si lavora con circuiti ad alto numero di qubit. La parallelizzazione diventa semplice quando viene utilizzata su Amazon CUDA-Q Braket Hybrid Jobs. Hybrid Jobs semplifica la distribuzione del campionamento dei circuiti e delle valutazioni osservabili su più nodi computazionali. Questa perfetta parallelizzazione dei carichi di lavoro consente agli utenti di concentrarsi maggiormente sullo sviluppo dei propri CUDA-Q carichi di lavoro piuttosto che sulla creazione di infrastrutture per esperimenti su larga scala.

Per iniziare, consulta l'esempio CUDA-Q iniziale su Amazon Braket examples Github per utilizzare CUDA-Q un contenitore di lavori ibrido fornito da Braket.

Il seguente frammento di codice è un `hello-world` esempio di esecuzione di un CUDA-Q programma con Amazon Braket Hybrid Jobs.

```
image_uri = retrieve_image(Framework.CUDAQ, AwsSession().region)

@hybrid_job(device='local:nvidia/qpp-cpu', image_uri=image_uri)
def hello_quantum():
    import cudaq

    # define the backend
    device=get_job_device_arn()
    cudaq.set_target(device.split('/')[-1])

    # define the Bell circuit
    kernel = cudaq.make_kernel()
    qubits = kernel.qalloc(2)
    kernel.h(qubits[0])
    kernel.cx(qubits[0], qubits[1])

    # sample the Bell circuit
    result = cudaq.sample(kernel, shots_count=1000)
    measurement_probabilities = dict(result.items())

    return measurement_probabilities
```

L'esempio precedente simula un circuito Bell su un simulatore di CPU. Questo esempio viene eseguito localmente sul laptop o sul notebook Braket Jupyter. A causa dell'`local=True`impostazione, quando si esegue questo script, nell'ambiente locale verrà avviato un contenitore per eseguire il CUDA-Q programma per il test e il debug. Dopo aver terminato il test, puoi rimuovere il `local=True` flag ed eseguire il job su. AWS Per ulteriori informazioni, consulta Lavorare con Amazon Braket Hybrid Jobs.

Se i tuoi carichi di lavoro hanno un numero elevato di qubit, un gran numero di circuiti o un gran numero di iterazioni, puoi utilizzare risorse di elaborazione della CPU più potenti specificando l'impostazione. `instance_config` Il seguente frammento di codice mostra come configurare l'impostazione nel decoratore. `instance_config` `hybrid_job` Per ulteriori informazioni sui tipi di istanze supportati, consulta Configurare l'istanza di job ibrida. Per un elenco dei tipi di istanze, consulta Tipi di istanze Amazon EC2.

```
@hybrid_job(
    device="local:nvidia/qpp-cpu",
```

```
    image_uri=image_uri,
    instance_config=InstanceConfig(instanceType="ml.c5.2xlarge"),
)
def my_job_script():
    ...
```

Per carichi di lavoro più impegnativi, puoi eseguire i carichi di lavoro su un CUDA-Q simulatore GPU. Per abilitare un simulatore GPU, usa il nome del backend. `nvidia` Il `nvidia` backend funziona come un simulatore di GPU. CUDA-Q Quindi, seleziona un tipo di istanza Amazon EC2 che supporti una NVIDIA GPU. Il seguente frammento di codice mostra il decoratore configurato dalla GPU. `hybrid_job`

```
@hybrid_job(
    device="local:nvidia/nvidia",
    image_uri=image_uri,
    instance_config=InstanceConfig(instanceType="ml.g4dn.xlarge"),
)
def my_job_script():
    ...
```

Amazon Braket Hybrid Jobs e supporta simulazioni GPU NBIs parallele con. CUDA-Q Puoi parallelizzare la valutazione di più osservabili o più circuiti per aumentare le prestazioni del tuo carico di lavoro. Per parallelizzare più osservabili, apporta le seguenti modifiche allo script dell'algoritmo.

Imposta l'`mgpuopzione` del backend. `nvidia` Ciò è necessario per parallelizzare gli osservabili. La parallelizzazione utilizza MPI per la comunicazione tra di loro GPUs, quindi MPI deve essere inizializzato prima dell'esecuzione e finalizzato dopo.

Quindi, specifica la modalità di esecuzione impostando. `execution=cudaq.parallel.mpi` Il seguente frammento di codice mostra queste modifiche.

```
cudaq.set_target("nvidia", option="mqpu")
cudaq.mpi.initialize()
result = cudaq.observe(
    kernel, hamiltonian, shots_count=n_shots, execution=cudaq.parallel.mpi
)
cudaq.mpi.finalize()
```

Nel `hybrid_job` decoratore, specifica un tipo di istanza che ne ospita più di uno, GPUs come mostrato nel seguente frammento di codice.

```
@hybrid_job(
    device="local:nvidia/nvidia-mqpu",
    instance_config=InstanceConfig(instanceType="ml.g4dn.12xlarge", instanceCount=1),
    image_uri=image_uri,
)
def parallel_observables_gpu_job(sagemaker_mpi_enabled=True):
    ...
```

Il notebook di simulazioni parallele negli esempi di Amazon Braket Github end-to-end fornisce esempi che dimostrano come eseguire simulazioni di programmi quantistici su backend di GPU ed eseguire simulazioni parallele di osservabili e batch di circuiti.

Esecuzione dei carichi di lavoro su computer quantistici

Dopo aver completato i test con il simulatore, puoi passare all'esecuzione di esperimenti su. QPUs Basta passare il target a una QPU Amazon Braket, ad esempio IQMIonQ, o dispositivi. Rigetti Il seguente frammento di codice illustra come impostare la destinazione sul dispositivo. IQM Garnet Per un elenco di quelli disponibili QPUs, consulta la console Amazon Braket.

```
device_arn = "arn:aws:braket:eu-north-1::device/qpu/iqm/Garnet"
cudaq.set_target("braket", machine=device_arn)
```

Per ulteriori informazioni su Hybrid Jobs, consulta Working with Amazon Braket Hybrid Jobs nella guida per sviluppatori. Per ulteriori informazioni su CUDA-Q, consulta la documentazione su NVIDIA CUDA-Q.

Risoluzione dei problemi con Amazon Braket

Utilizza le informazioni e le soluzioni per la risoluzione dei problemi in questa sezione per risolvere i problemi con Amazon Braket.

In questa sezione:

AccessDeniedException

Se ricevi un messaggio AccessDeniedExceptionquando attivi o utilizzi Braket, è probabile che tu stia tentando di abilitare o utilizzare Braket in una regione in cui il tuo ruolo limitato non ha accesso.

In questi casi, contatta il tuo AWS amministratore interno per capire quale delle seguenti condizioni si applica:

- Se esistono restrizioni relative ai ruoli che impediscono l'accesso a una regione.
- Se il ruolo che stai tentando di utilizzare è autorizzato, usa Braket.

Se il tuo ruolo non ha accesso a una determinata regione quando usi Braket, non potrai utilizzare i dispositivi in quella particolare regione.

Si è verificato un errore (ValidationException) durante la chiamata dell'operazione CreateQuantumTask

Se ricevi un errore simile a: `An error occurred (ValidationException) when calling the CreateQuantumTask operation: Caller doesn't have access to amazon-`

`braket-…` Verifica di fare riferimento a una s3_folder esistente. Braket non crea automaticamente nuovi bucket e prefissi Amazon S3 per te.

Se accedi API direttamente a e ricevi un errore simile a: `Failed to create quantum task: Caller doesn't have access to s3://MY_BUCKET` Verifica di non includere `s3://` nel percorso del bucket Amazon S3.

Una funzionalità SDK non funziona

La tua versione di Python deve essere 3.10 o superiore. Per Amazon Braket Hybrid Jobs, consigliamo Python 3.12.

Verifica che il tuo SDK e gli schemi lo siano. up-to-date Per aggiornare l'SDK dal notebook o dall'editor Python, esegui il seguente comando:

```
pip install amazon-braket-sdk --upgrade --upgrade-strategy eager
```

Per aggiornare gli schemi, esegui il seguente comando:

```
pip install amazon-braket-schemas --upgrade
```

Se accedi ad Amazon Braket dal tuo cliente, verifica che la tua AWS regione sia impostata su una regione supportata da Amazon Braket.

Il processo ibrido fallisce a causa di ServiceQuotaExceededException

Un job ibrido che esegue attività quantistiche con i simulatori Amazon Braket può non essere creato se si supera il limite di attività quantistiche simultanee per il dispositivo di simulazione a cui si sta puntando. Per ulteriori informazioni sui limiti del servizio, consulta l'argomento Quotas.

Se esegui attività simultanee su un dispositivo di simulazione in più lavori ibridi dal tuo account, potresti riscontrare questo errore.

Per visualizzare il numero di attività quantistiche simultanee su uno specifico dispositivo di simulazione, utilizzate il `search-quantum-tasks`API, come illustrato nel seguente esempio di codice.

```
DEVICE_ARN=arn:aws:braket:::device/quantum-simulator/amazon/sv1
```

```
task_list=""
for status_value in "CREATED" "QUEUED" "RUNNING" "CANCELLING"; do
    tasks=$(aws braket search-quantum-tasks --filters
 name=status,operator=EQUAL,values=${status_value}
 name=deviceArn,operator=EQUAL,values=$DEVICE_ARN --max-results 100 --query
 'quantumTasks[*].quantumTaskArn' --output text)
    task_list="$task_list $tasks"
done;
echo "$task_list" | tr -s ' \t' '[\n*]' | sort | uniq
```

Puoi anche visualizzare le attività quantistiche create su un dispositivo utilizzando i CloudWatch parametri di Amazon: Braket > By Device.

Per evitare di incorrere in questi errori:

1. Richiedete un aumento della quota di servizio per il numero di attività quantistiche simultanee per il dispositivo di simulazione. Questo è applicabile solo al dispositivo. SV1
2. Gestisci `ServiceQuotaExceeded` le eccezioni nel codice e riprova.

I componenti hanno smesso di funzionare nell'istanza del notebook

Se alcuni componenti del notebook smettono di funzionare, prova quanto segue:

1. Scarica tutti i notebook che hai creato o modificato su un'unità locale.
2. Interrompi l'istanza del tuo notebook.
3. Elimina l'istanza del tuo notebook.
4. Crea una nuova istanza del notebook con un nome diverso.
5. Carica i taccuini nella nuova istanza.

Risoluzione dei problemi di aggiornamento a Python 3.12

Data di decorrenza: 21 gennaio 2026

Panoramica di

A partire dal 21 gennaio 2026, Amazon Braket aggiorna il runtime Python dalla versione 3.10 alla 3.12 per tutte le istanze di notebook e le immagini dei container gestiti (Base, CUDA-Q e). TensorFlow PyTorch Questa guida fornisce soluzioni per problemi di compatibilità comuni.

In questa sezione:

Messaggi di errore comuni

Errore di mancata corrispondenza della versione di Python dell'SDK

Errore:

```
RuntimeError: Python version must match between local environment and container. Client
 is running Python 3.10 locally, but container uses Python 3.12.
```

Causa: l'SDK Braket ha rilevato che sul notebook è in esecuzione Python 3.10 ma nel contenitore Hybrid Job è in esecuzione Python 3.12.

Soluzione: aggiorna il notebook a Python 3.12 o esegui il pin ai contenitori Python 3.10.

Errore di serializzazione di Cloudpickle

Errore:

```
TypeError: code() argument 13 must be str, not int
```

Causa: se la convalida dell'SDK viene ignorata, cloudpickle non riesce a serializzare il codice tra Python 3.10 e 3.12 a causa di una modifica del costruttore in Python 3.12. CodeType

Soluzione: assicurati che il notebook e il contenitore utilizzino la stessa versione di Python.

Notebook gestiti da Braket

Se stai eseguendo un'istanza di Braket Notebook su Python 3.10 e invii lavori ibridi, riscontrerai errori di mancata corrispondenza delle versioni perché i contenitori di lavoro ora utilizzano Python 3.12 per impostazione predefinita.

Sono disponibili due opzioni:

1. [Consigliato] Crea una nuova istanza Notebook con Python 3.12 - vedi Aggiornamento dell'istanza di Braket Notebook
2. Pin ai contenitori Python 3.10 - vedi Hybrid Job Decorator

Job Decorator ibrido

Per usare il `@hybrid_job` decoratore, la versione Python del tuo ambiente deve corrispondere alla versione Python del contenitore.

Opzione 1: usa i contenitori Python 3.12 (consigliato)

Se hai aggiornato il tuo ambiente a Python 3.12, utilizza il tag più recente (comportamento predefinito).

Opzione 2: usa i contenitori Python 3.10

Se devi rimanere su Python 3.10, specifica esplicitamente il `image_uri` parametro in decorator. `@hybrid_job`

Tag contenitore Python 3.10:

Nome immagine	Tag
Base	1.0-cpu-py310-ubuntu22.04
CUDA-Q	0.12.0-cpu-py310-0.12.0
PyTorch	2.2.0-gpu-py310-cu121-ubuntu20.04
TensorFlow	2.14.1-gpu-py310-cu118-ubuntu20.04

L'esempio seguente si riferisce alla regione us-west-2.

Immagine completa: URIs

```
Base:       292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-base-jobs:1.0-cpu-py310-ubuntu22.04
```

```
CUDA-Q:     292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-cudaq-
jobs:0.12.0-cpu-py310-0.12.0
PyTorch:    292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-pytorch-
jobs:2.2.0-gpu-py310-cu121-ubuntu20.04
TensorFlow: 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-tensorflow-
jobs:2.14.1-gpu-py310-cu118-ubuntu20.04
```

Esempio:

```
from braket.jobs.hybrid_job import hybrid_job
from braket.devices import Devices

device_arn = Devices.Amazon.SV1

@hybrid_job(
    device=device_arn,
    image_uri="292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-base-
jobs:1.0-cpu-py310-ubuntu22.04"
)
def my_job():
    pass
```

Note

- I contenitori Python 3.10 rimarranno disponibili ma non riceveranno aggiornamenti.
- Vedi Definire l'ambiente per lo script dell'algoritmo.

Bring-Your-Own-Container (BYOC)

Se il tuo Dockerfile utilizza un'immagine gestita da Braket con il tag più recente, la ricostruzione dopo il 21 gennaio 2026 estrarrà le immagini supportate da Python 3.12.

Per rimanere sulle immagini gestite da Braket supportate da Python 3.10, aggiorna il tuo Dockerfile:

Before (ottiene Python 3.12 dopo l'aggiornamento):

```
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-base-jobs:latest
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-cudaq-jobs:latest
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-tensorflow-jobs:latest
```

```
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-pytorch-jobs:latest
```

Dopo (rimane su Python 3.10):

```
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-base-jobs:1.0-cpu-py310-ubuntu22.04
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-cudaq-jobs:0.12.0-cpu-py310-0.12.0
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-pytorch-jobs:2.2.0-gpu-py310-cu121-ubuntu20.04
FROM 292282985366.dkr.ecr.us-west-2.amazonaws.com/amazon-braket-tensorflow-jobs:2.14.1-gpu-py310-cu118-ubuntu20.04
```

Aggiornamento dell'istanza di Braket Notebook

Segui questi passaggi per eseguire l'aggiornamento a Python 3.12:

> **Important**
>
> Prima di eliminare l'istanza del notebook, assicurati di aver scaricato tutti i taccuini e i file che desideri conservare. Questi dati non possono essere recuperati dopo l'eliminazione.

1. Scarica tutti i taccuini creati o modificati su un'unità locale.
2. Arresta l'istanza del notebook.
3. Elimina l'istanza del tuo notebook.
4. Crea una nuova istanza del notebook con un nome diverso.
5. Carica i tuoi taccuini nella nuova istanza.

Risoluzione dei problemi relativi a OpenQASM

Questa sezione fornisce suggerimenti per la risoluzione dei problemi che potrebbero essere utili in caso di errori utilizzando OpenQASM 3.0.

In questa sezione:

- Combinazione di errori fisici qubits con qubits errori virtuali
- Richiesta dei tipi di risultati e misurazione qubits nello stesso errore di programma
- I limiti classici e di qubit registro hanno superato l'errore
- Casella non preceduta da un errore pragmatico letterale
- Errore con porte native mancanti nelle caselle Verbatim
- Errore fisico mancante nelle caselle Verbatim qubits
- Nel pragma letterale manca l'errore «braket»
- L'errore Single qubits non può essere indicizzato
- Errore fisico qubits in una connessione a due qubit porte
- Avviso di supporto al simulatore locale

Errore di inclusione dell'istruzione

Braket attualmente non dispone di un file di libreria gate standard da includere nei programmi OpenQASM. Ad esempio, l'esempio seguente genera un errore del parser.

```
OPENQASM 3;
include "standardlib.inc";
```

Questo codice genera il messaggio di errore: `No terminal matches '"' in the current parser context, at line 2 col 17.`

Errore non contiguo qubits

L'utilizzo della funzionalità non contigua qubits su dispositivi `requiresContiguousQubitIndices` impostati sulla funzionalità del dispositivo `true` genera un errore.

Quando si eseguono attività quantistiche su simulatori eIonQ, il seguente programma attiva l'errore.

```
OPENQASM 3;

qubit[4] q;

h q[0];
cnot q[0], q[2];
cnot q[0], q[3];
```

Questo codice genera il messaggio di errore: `Device requires contiguous qubits. Qubit register q has unused qubits q[1], q[4].`

Combinazione di errori fisici qubits con qubits errori virtuali

La combinazione qubits di elementi fisici e virtuali qubits nello stesso programma non è consentita e genera un errore. Il codice seguente genera l'errore.

```
OPENQASM 3;

qubit[2] q;
cnot q[0], $1;
```

Questo codice genera il messaggio di errore: `[line 4] mixes physical qubits and qubits registers.`

Richiesta dei tipi di risultati e misurazione qubits nello stesso errore di programma

La richiesta di tipi di risultati qubits misurati esplicitamente nello stesso programma genera un errore. Il codice seguente genera l'errore.

```
OPENQASM 3;

qubit[2] q;

h q[0];
cnot q[0], q[1];
measure q;

#pragma braket result expectation x(q[0]) @ z(q[1])
```

Questo codice genera il messaggio di errore: `Qubits should not be explicitly measured when result types are requested.`

I limiti classici e di qubit registro hanno superato l'errore

Sono ammessi solo un qubit registro classico e un registro. Il codice seguente genera l'errore.

```
OPENQASM 3;
```

```
qubit[2] q0;
qubit[2] q1;
```

Questo codice genera il messaggio di errore: `[line 4] cannot declare a qubit register. Only 1 qubit register is supported.`

Casella non preceduta da un errore pragmatico letterale

Tutte le caselle devono essere precedute da un pragma letterale. Il codice seguente genera l'errore.

```
box{
rx(0.5) $0;
}
```

Questo codice genera il messaggio di errore: `In verbatim boxes, native gates are required. x is not a device native gate.`

Errore con porte native mancanti nelle caselle Verbatim

Le scatole Verbatim devono avere porte native e fisiche. qubits Il codice seguente genera l'errore native gates.

```
#pragma braket verbatim
box{
x $0;
}
```

Questo codice genera il messaggio di errore: `In verbatim boxes, native gates are required. x is not a device native gate.`

Errore fisico mancante nelle caselle Verbatim qubits

Le scatole Verbatim devono essere fisiche. qubits Il codice seguente genera l'errore fisico qubits mancante.

```
qubit[2] q;

#pragma braket verbatim
box{
```

```
rx(0.1) q[0];
}
```

Questo codice genera il messaggio di errore: `Physical qubits are required in verbatim box.`

Nel pragma letterale manca l'errore «braket»

Devi includere «parentesi» nel pragma letterale. Il codice seguente genera l'errore.

```
#pragma braket verbatim          // Correct
#pragma verbatim                 // wrong
```

Questo codice genera il messaggio di errore: `You must include “braket” in the verbatim pragma`

L'errore Single qubits non può essere indicizzato

Il singolo qubits non può essere indicizzato. Il codice seguente genera l'errore.

```
OPENQASM 3;

qubit q;
h q[0];
```

Questo codice genera l'errore: `[line 4] single qubit cannot be indexed.`

Tuttavia, i singoli qubit array possono essere indicizzati come segue:

```
OPENQASM 3;

qubit[1] q;
h q[0];   // This is valid
```

Errore fisico qubits in una connessione a due qubit porte

Per utilizzare dispositivi fisiciqubits, verifica innanzitutto che il dispositivo utilizzi dispositivi fisici qubits controllando `device.properties.action[DeviceActionType.OPENQASM].supportPhysicalQubits` e

quindi verifica il grafico di connettività selezionando `device.properties.paradigm.connectivity.connectivityGraph` o`device.properties.paradigm.connectivity.fullyConnected`.

```
OPENQASM 3;

cnot $0, $14;
```

Questo codice genera il messaggio di errore: `[line 3] has disconnected qubits 0 and 14`

Avviso di supporto al simulatore locale

`LocalSimulator`Supporta funzionalità avanzate in OpenQASM che potrebbero non essere disponibili nei simulatori QPUs o su richiesta. Se il programma contiene funzionalità linguistiche specifiche solo per`LocalSimulator`, come illustrato nell'esempio seguente, riceverai un avviso.

```
qasm_string = """
qubit[2] q;

h q[0];
ctrl @ x q[0], q[1];
"""
qasm_program = Program(source=qasm_string)
```

Questo codice genera l'avviso: `Questo programma utilizza le funzionalità del linguaggio OpenQASM supportate solo in. LocalSimulator Alcune di queste funzionalità potrebbero non essere supportate nei simulatori QPUs o su richiesta.

Per ulteriori informazioni sulle funzionalità OpenQASM supportate, esplora la pagina Supporto delle funzionalità avanzate per OpenQASM su Local Simulator.

Sicurezza in Amazon Braket

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di data center e architetture di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo aspetto come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi in Cloud AWS. AWS fornisce inoltre servizi che è possibile utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon Braket, consulta AWS Services in Scope by Compliance Program Program.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Inoltre, sei responsabile anche di altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e le leggi e le normative applicabili.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa quando usi Braket. I seguenti argomenti mostrano come configurare Braket per soddisfare i tuoi obiettivi di sicurezza e conformità. Imparerai anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Braket.

In questa sezione:

Responsabilità condivisa per la sicurezza

La sicurezza è una responsabilità condivisa tra voi AWS e voi. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gira Servizi AWS su Cloud AWS. AWS fornisce inoltre servizi che è possibile utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon Braket, consulta AWS Services in Scope by Compliance Program.
- Sicurezza nel cloud: hai la responsabilità di mantenere il controllo sui contenuti ospitati su questa AWS infrastruttura. Questo contenuto include le attività di configurazione e gestione della sicurezza per Servizi AWS ciò che utilizzi.

Protezione dei dati

Il modello di responsabilità AWS condivisa si applica alla protezione dei dati in Amazon Braket. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per maggiori informazioni sulla privacy dei dati, consulta le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al AWS Modello di responsabilità condivisa e GDPR nel AWS Blog sulla sicurezza.

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.

- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Amazon Braket o altro Servizi AWS utilizzando la console, l'API o. AWS CLI AWS SDKs I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la richiesta al server.

Conservazione dei dati

Dopo 90 giorni, Amazon Braket rimuove automaticamente tutte le attività quantistiche IDs e gli altri metadati associati alle tue attività quantistiche. Come risultato di questa politica di conservazione dei dati, queste attività e i risultati non sono più recuperabili tramite ricerca dalla console Amazon Braket, sebbene rimangano archiviati nel bucket S3.

Se hai bisogno di accedere alle attività e ai risultati quantistici storici archiviati nel tuo bucket S3 per più di 90 giorni, devi tenere un registro separato dell'ID dell'attività e degli altri metadati associati a tali dati. Assicurati di salvare le informazioni prima di 90 giorni. È possibile utilizzare le informazioni salvate per recuperare i dati storici.

Gestione dell'accesso ad Amazon Braket

Questo capitolo descrive le autorizzazioni necessarie per eseguire Amazon Braket o per limitare l'accesso di utenti e ruoli specifici. Puoi concedere (o negare) le autorizzazioni richieste a qualsiasi utente o ruolo del tuo account. A tale scopo, allega la policy Amazon Braket appropriata a quell'utente o ruolo nel tuo account, come descritto nelle sezioni seguenti.

Come prerequisito, devi abilitare Amazon Braket. Per abilitare Braket, assicurati di accedere come utente o ruolo con (1) autorizzazioni di amministratore o (2) che disponga della

AmazonBraketFullAccesspolicy e delle autorizzazioni per creare bucket Amazon Simple Storage Service (Amazon S3).

In questa sezione:

Risorse Amazon Braket

Braket crea un tipo di risorsa: la risorsa quantum-task. Il nome della AWS risorsa (ARN) per questo tipo di risorsa è il seguente:

- Nome risorsa:: :Service AWS: :Braket
- ARN Regex: arn: $ {Partition} :braket: $ {Region} :$ {Account} :quantum-task/$ {} RandomId

Notebook e ruoli

Puoi usare il tipo di risorsa notebook in Braket. Un notebook è una risorsa SageMaker AI di Amazon che Braket è in grado di condividere. Per utilizzare un notebook con Braket, devi specificare un ruolo IAM con un nome che inizia con. `AmazonBraketServiceSageMakerNotebook`

Per creare un notebook, è necessario utilizzare un ruolo con autorizzazioni di amministratore o a cui sia associata la seguente politica in linea.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CreateTheRole",
```

```
      "Effect": "Allow",
      "Action": "iam:CreateRole",
      "Resource": "arn:aws:iam::*:role/service-role/
AmazonBraketServiceSageMakerNotebookRole*"
    },
    {
      "Sid": "CreateThePolicy",
      "Effect": "Allow",
      "Action": "iam:CreatePolicy",
      "Resource": [
        "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookAccess*",
        "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookRole*"
      ]
    },
    {
      "Sid": "AttachTheRolePolicy",
      "Effect": "Allow",
      "Action": "iam:AttachRolePolicy",
      "Resource": "arn:aws:iam::*:role/service-role/
AmazonBraketServiceSageMakerNotebookRole*",
      "Condition": {
        "ArnLike": {
          "iam:PolicyARN": [
            "arn:aws:iam::aws:policy/AmazonBraketFullAccess",
            "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookAccess*",
            "arn:aws:iam::*:policy/service-role/
AmazonBraketServiceSageMakerNotebookRole*"
          ]
        }
      }
    }
  ]
}
```

Per creare il ruolo, segui i passaggi indicati nella pagina Crea un taccuino o chiedi all'amministratore di crearlo per te. Assicurati che la AmazonBraketFullAccesspolicy sia allegata.

Dopo aver creato il ruolo, puoi riutilizzarlo per tutti i notebook che lancerai in futuro.

AWS politiche gestite per Amazon Braket

Una politica AWS gestita è una politica autonoma creata e amministrata da AWS. AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Si consiglia pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i propri casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

Argomenti

AWS politica gestita: AmazonBraketFullAccess

La AmazonBraketFullAccesspolicy concede le autorizzazioni per le operazioni di Amazon Braket, incluse le autorizzazioni per le seguenti attività:

- Scarica contenitori da Amazon Elastic Container Registry: per leggere e scaricare immagini di container utilizzate per la funzionalità Amazon Braket Hybrid Jobs. I contenitori devono essere conformi al formato «arn:aws:ecr: :repository/amazon-braket».
- Conserva AWS CloudTrail i log: per tutte le azioni di descrizione, recupero ed elenco, oltre all'avvio e all'interruzione delle query, al test dei filtri delle metriche e al filtraggio degli eventi di registro. Il file di AWS CloudTrail registro contiene un record di tutte le API attività di Amazon Braket che si verificano nel tuo account.

- Utilizza i ruoli per controllare le risorse: per creare un ruolo collegato al servizio nel tuo account. Il ruolo collegato al servizio ha accesso alle risorse per AWS tuo conto. Può essere utilizzato solo dal servizio Amazon Braket. Inoltre, per trasferire i ruoli IAM ad Amazon Braket `CreateJob` API e creare un ruolo e allegare una policy mirata AmazonBraketFullAccess al ruolo.
- Crea gruppi di log, eventi di log e gruppi di log di query per gestire i file di log di utilizzo per il tuo account: per creare, archiviare e visualizzare informazioni di registrazione sull'utilizzo di Amazon Braket nel tuo account. Interroga le metriche sui gruppi di log dei lavori ibridi. Comprende il percorso Braket corretto e consenti l'inserimento dei dati di registro. Inserisci i dati metrici. CloudWatch
- Crea e archivia dati nei bucket Amazon S3 ed elenca tutti i bucket — Per creare bucket S3, elenca i bucket S3 nel tuo account e inserisci oggetti e ottieni oggetti da qualsiasi bucket del tuo account il cui nome inizia con amazon-braket-. Queste autorizzazioni sono necessarie per consentire a Braket di inserire i file contenenti i risultati di attività quantistiche elaborate nel bucket e di recuperarli dal bucket.
- Passa i ruoli IAM: per passare i ruoli IAM a. `CreateJob` API
- Amazon SageMaker AI Notebook: per creare e gestire istanze di SageMaker notebook limitate alla risorsa da «arn:aws:sagemaker: ::notebook-instance/amazon-braket-».
- Convalida delle quote di servizio: per creare notebook SageMaker AI e lavori Amazon Braket Hybrid, il numero di risorse non può superare le quote del tuo account.
- Visualizza i prezzi dei prodotti: esamina e pianifica i costi dell'hardware quantistico prima di inviare i carichi di lavoro.

Per visualizzare le autorizzazioni per questa policy, consulta AmazonBraketFullAccessAWS Managed Policy Reference.

AWS politica gestita: AmazonBraketJobsExecutionPolicy

La AmazonBraketJobsExecutionPolicypolicy concede le autorizzazioni per i ruoli di esecuzione utilizzati in Amazon Braket Hybrid Jobs come segue:

- Scarica contenitori da Amazon Elastic Container Registry - Autorizzazioni per leggere e scaricare immagini di container utilizzate per la funzionalità Amazon Braket Hybrid Jobs. I contenitori devono essere conformi al formato «arn:aws:ecr: *:*:repository/amazon-braket*».
- Crea gruppi di log ed eventi di log e interroga gruppi di log per gestire i file di log di utilizzo per il tuo account: crea, archivia e visualizza le informazioni di registrazione sull'utilizzo di Amazon Braket

nel tuo account. Esegui query sui gruppi di log dei lavori ibridi. Comprende il percorso Braket corretto e consenti l'inserimento dei dati di registro. Inserisci i dati metrici. CloudWatch

- Archivia i dati in bucket Amazon S3: elenca i bucket S3 nel tuo account, inserisci oggetti e ottieni oggetti da qualsiasi bucket del tuo account che inizia con amazon-braket- nel nome. Queste autorizzazioni sono necessarie per consentire a Braket di inserire i file contenenti i risultati di attività quantistiche elaborate nel bucket e di recuperarli dal bucket.
- Passa i ruoli IAM: trasferimento dei ruoli IAM a. CreateJob API I ruoli devono essere conformi al formato arn:aws:iam: :* *. :role/service-role/AmazonBraketJobsExecutionRole

Per visualizzare le autorizzazioni per questa policy, consulta AmazonBraketJobsExecutionPolicyAWS Managed Policy Reference.

AWS politica gestita: AmazonBraketServiceRolePolicy

La AmazonBraketServiceRolePolicypolicy concede le autorizzazioni per le operazioni di Amazon Braket, incluse le autorizzazioni per le seguenti attività:

- Amazon S3: autorizzazioni per elencare i bucket nel tuo account e inserire oggetti e ottenere oggetti da qualsiasi bucket del tuo account con un nome che inizia con. `amazon-braket-`
- Amazon CloudWatch Logs: autorizzazioni per elencare e creare gruppi di log, creare i flussi di log associati e inserire eventi nel gruppo di log creato per Amazon Braket.

Per ulteriori informazioni sui ruoli collegati ai servizi, consulta il ruolo collegato ai servizi di Amazon Braket.

Per visualizzare le autorizzazioni per questa policy, consulta AmazonBraketServiceRolePolicyAWS Managed Policy Reference.

Amazon Braket si aggiorna alle AWS politiche gestite

La tabella seguente fornisce dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon Braket dal momento in cui questo servizio ha iniziato a tracciare queste modifiche.

Modifica	Descrizione	Data
AmazonBraketServiceRolePolicy- Politica di gestione delle risorse	Aggiunto l'ambito della condizion e «aws:ResourceAccount":	11 luglio 2025

Modifica	Descrizione	Data
	«$ {aws:PrincipalAccount}» ad Amazon S3 e CloudWatch registra le azioni.	
AmazonBraketFullAccess- Politica di accesso completo per Braket	Aggiunta l'azione «pricing: GetProducts».	14 aprile 2025
AmazonBraketFullAccess- Politica di accesso completo per Braket	Aggiunto l'ambito della condizion e ResourceAccount«aws: «: «$ {aws:PrincipalAccount}» alle azioni S3.	7 marzo 2025
AmazonBraketFullAccess- Politica di accesso completa per Braket	Aggiunte le azioni servicequotas: GetServiceQuota e cloudwatch:. GetMetricData	24 marzo 2023
AmazonBraketFullAccess- Politica di accesso completo per Braket	Aggiunto s3: ListAllMyBuckets autorizzazioni per visualizzare e ispezionare i bucket Amazon S3 usati.	31 marzo 2022
AmazonBraketFullAccess- Politica di accesso completo per Braket	Braket adjusted iam: PassRole autorizzazioni AmazonBra ketFullAccess per includere il percorso. `service-role/`	29 novembre 2021
AmazonBraketJobsExecutionPolicy- Politica di esecuzione dei lavori ibridi per Amazon Braket Hybrid Jobs	Braket ha aggiornato l'ARN del ruolo di esecuzione dei lavori ibridi per includere `service-r ole/` il percorso.	29 novembre 2021
Braket ha iniziato a tracciare le modifiche	Braket ha iniziato a tenere traccia delle modifiche per le sue politiche AWS gestite.	29 novembre 2021

Limita l'accesso degli utenti a determinati dispositivi

Per limitare l'accesso degli utenti a determinati dispositivi Braket, puoi aggiungere una politica di negazione delle autorizzazioni a un ruolo specifico. IAM

Le seguenti azioni possono essere limitate:

- `CreateQuantumTask`- per negare la creazione di attività quantistiche su dispositivi specifici.
- `CreateJob`- negare la creazione di posti di lavoro ibridi su dispositivi specifici.
- `GetDevice`- negare di ottenere dettagli su dispositivi specifici.

L'esempio seguente limita l'accesso a tutti QPUs per. Account AWS `123456789012`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
    "Effect": "Deny",
    "Action": [
      "braket:CreateQuantumTask",
      "braket:CreateJob",
      "braket:GetDevice"
    ],
    "Resource": [
    "arn:aws:braket:*:*:device/qpu/*"
    ],
    "Condition": {
             "StringEquals": {
                 "aws:PrincipalAccount": "123456789012"
             }
      }
    }
  ]
}
```

> **Note**
>
> Escludi l'`braket:GetDeviceAction` dalla policy per consentire l'accesso in lettura di un utente alle proprietà del dispositivo, come la disponibilità del dispositivo, i dati di calibrazione e i prezzi, tramite la console Braket.

Per adattare questo codice, sostituite il Amazon Resource Number (ARN) del dispositivo con restrizioni alla stringa mostrata nell'esempio precedente. Questa stringa fornisce il valore Resource. In Braket, un dispositivo rappresenta una QPU o un simulatore che è possibile chiamare per eseguire attività quantistiche. I dispositivi disponibili sono elencati nella pagina Dispositivi. Esistono due schemi utilizzati per specificare l'accesso a questi dispositivi:

- `arn:aws:braket:<region>:*:device/qpu/<provider>/<device_id>`
- `arn:aws:braket:<region>:*:device/quantum-simulator/<provider>/<device_id>`

Di seguito sono riportati alcuni esempi di vari tipi di accesso ai dispositivi

- Per selezionare QPUs tutte le aree geografiche: `arn:aws:braket:*:*:device/qpu/*`
- Per selezionare tutto QPUs SOLO nella regione us-west-2: `arn:aws:braket:us-west-2:*:device/qpu/*`
- Equivalentemente, per selezionare tutto SOLO QPUs nella regione us-west-2 (poiché i dispositivi sono una risorsa di servizio, non una risorsa per i clienti): `arn:aws:braket:us-west-2:*:device/qpu/*`
- Per limitare l'accesso a tutti i dispositivi di simulazione su richiesta: `arn:aws:braket:*:*:device/quantum-simulator/*`
- Per limitare l'accesso ai dispositivi di un determinato provider (ad esempio, ai Rigetti QPU dispositivi): `arn:aws:braket:*:*:device/qpu/rigetti/*`
- Per limitare l'accesso al TN1 dispositivo: `arn:aws:braket:*:*:device/quantum-simulator/amazon/tn1`
- Per limitare l'accesso a tutte le `Create` azioni: `braket:Create*`

Limita l'accesso degli utenti a determinate istanze del notebook

Per limitare l'accesso di determinati utenti a istanze specifiche di Braket Notebook, puoi aggiungere una politica di negazione delle autorizzazioni a un ruolo, utente o gruppo specifico.

L'esempio seguente utilizza variabili di policy per limitare in modo efficiente le autorizzazioni di avvio, arresto e accesso a istanze di notebook specifiche in Account AWS `123456789012`, denominate in base all'utente che dovrebbe avere accesso (ad esempio, l'utente `Alice` avrebbe accesso a un'istanza del notebook denominata). `amazon-braket-Alice`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyCreateDeleteUpdateNotebookInstances",
      "Effect": "Deny",
      "Action": [
        "sagemaker:CreateNotebookInstance",
        "sagemaker:DeleteNotebookInstance",
        "sagemaker:UpdateNotebookInstance",
        "sagemaker:CreateNotebookInstanceLifecycleConfig",
        "sagemaker:DeleteNotebookInstanceLifecycleConfig",
        "sagemaker:UpdateNotebookInstanceLifecycleConfig"
      ],
      "Resource": "*"
    },
    {
      "Sid": "DenyDescribeStartStopNotebookInstances",
      "Effect": "Deny",
      "Action": [
        "sagemaker:DescribeNotebookInstance",
        "sagemaker:StartNotebookInstance",
        "sagemaker:StopNotebookInstance"
      ],
      "NotResource": [
        "arn:aws:sagemaker:*:123456789012:notebook-instance/amazon-braket-
${aws:username}"
      ]
    },
    {
```

```
      "Sid": "DenyNotebookInstanceUrl",
      "Effect": "Deny",
      "Action": [
        "sagemaker:CreatePresignedNotebookInstanceUrl"
      ],
      "NotResource": [
        "arn:aws:sagemaker:*:123456789012:notebook-instance/amazon-braket-
${aws:username}*"
      ]
    }
  ]
}
```

Limita l'accesso degli utenti a determinati bucket S3

Per limitare l'accesso di determinati utenti a specifici bucket Amazon S3, puoi aggiungere una politica di rifiuto a un ruolo, utente o gruppo specifico.

L'esempio seguente limita le autorizzazioni per recuperare e posizionare oggetti in un S3 bucket specifico (`arn:aws:s3:::amazon-braket-us-east-1-123456789012-Alice`) e limita anche l'elenco di tali oggetti.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "s3:ListBucket"
      ],
      "NotResource": [
        "arn:aws:s3:::amazon-braket-us-east-1-123456789012-Alice"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "s3:GetObject"
      ],
```

```
      "NotResource": [
        "arn:aws:s3:::amazon-braket-us-east-1-123456789012-Alice/*"
      ]
    }
  ]
}
```

Per limitare l'accesso al bucket per una determinata istanza di notebook, è possibile aggiungere la politica precedente al ruolo di esecuzione del notebook.

Ruolo collegato al servizio Amazon Braket

Quando abiliti Amazon Braket, nel tuo account viene creato un ruolo collegato al servizio.

Un ruolo collegato ai servizi è un tipo unico di ruolo IAM che, in questo caso, è collegato direttamente ad Amazon Braket. Il ruolo collegato al servizio Amazon Braket è predefinito per includere tutte le autorizzazioni richieste da Braket quando chiama altre persone per tuo conto. Servizi AWS

Un ruolo collegato al servizio semplifica la configurazione di Amazon Braket perché non è necessario aggiungere manualmente le autorizzazioni necessarie. Amazon Braket definisce le autorizzazioni dei suoi ruoli collegati ai servizi. A meno che non modifichi queste definizioni, solo Amazon Braket può assumerne i ruoli. Le autorizzazioni definite includono la politica di fiducia e la politica delle autorizzazioni. Le policy di autorizzazioni non possono essere collegate a nessun'altra entità IAM.

Il ruolo collegato ai servizi impostato da Amazon Braket fa parte della funzionalità dei ruoli collegati AWS Identity and Access Management ai servizi (IAM). Per informazioni su altri ruoli Servizi AWS che supportano ruoli collegati ai servizi, consulta AWS Servizi che funzionano con IAM e cerca i servizi con Sì nella colonna Service-Linked Role. Scegli un Sì con un link per visualizzare la documentazione del ruolo collegato al servizio per quel servizio.

Per ulteriori informazioni sulla politica AWS gestita per i ruoli collegati al servizio, consulta. AmazonBraketServiceRolePolicy

Convalida della conformità per Amazon Braket

> **Note**
>
> AWS i report di conformità non riguardano QPUs i fornitori di hardware di terze parti che possono scegliere di sottoporsi a controlli indipendenti.

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Ambito per programma di conformitàServizi AWS di conformità e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta AWS la documentazione sulla sicurezza.

Sicurezza dell'infrastruttura in Amazon Braket

In quanto servizio gestito, Amazon Braket è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere ad Amazon Braket attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Puoi chiamare queste operazioni API da qualsiasi posizione di rete, ma Braket supporta politiche di accesso basate sulle risorse, che possono includere restrizioni basate sull'indirizzo IP di origine. Puoi

anche utilizzare le policy di Braket per controllare l'accesso da endpoint Amazon Virtual Private Cloud (Amazon VPC) specifici o specifici. VPCs In effetti, questo isola l'accesso alla rete a una determinata risorsa Braket solo dal VPC specifico all'interno della rete. AWS

Sicurezza dei fornitori di hardware Amazon Braket

QPUs su Amazon Braket sono ospitati da fornitori di hardware di terze parti. Quando esegui un'attività quantistica su una QPU, Amazon Braket utilizza DeviceEarn come identificatore per inviare il circuito alla QPU specificata per l'elaborazione.

Se utilizzi Amazon Braket per accedere all'hardware di calcolo quantistico gestito da uno dei fornitori di hardware di terze parti, il tuo circuito e i dati associati vengono elaborati da fornitori di hardware esterni alle strutture gestite da. AWS Le informazioni sulla posizione fisica e sulla AWS regione in cui ogni QPU è disponibile sono disponibili nella sezione Dettagli del dispositivo della console Amazon Braket.

I tuoi contenuti sono resi anonimi. Solo il contenuto necessario per elaborare il circuito viene inviato a terzi. Account AWS le informazioni non vengono trasmesse a terzi.

Tutti i dati sono crittografati sia a riposo che in transito. I dati vengono decrittografati solo per l'elaborazione. I fornitori terzi di Amazon Braket non sono autorizzati a archiviare o utilizzare i tuoi contenuti per scopi diversi dall'elaborazione del tuo circuito. Una volta completato il circuito, i risultati vengono restituiti ad Amazon Braket e archiviati nel bucket S3.

La sicurezza dei fornitori di hardware quantistico di terze parti di Amazon Braket viene verificata periodicamente per garantire il rispetto degli standard di sicurezza della rete, controllo degli accessi, protezione dei dati e sicurezza fisica.

Endpoint Amazon VPC per Amazon Braket

Puoi stabilire una connessione privata tra il tuo VPC e Amazon Braket creando un endpoint VPC di interfaccia. Gli endpoint di interfaccia sono alimentati da AWS PrivateLink, una tecnologia che consente l'accesso a Braket APIs senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. Direct Connect Le istanze nel tuo VPC non necessitano di indirizzi IP pubblici per comunicare con Braket. APIs

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle sottoreti.

Inoltre AWS PrivateLink, il traffico tra il tuo VPC e Braket non esce dalla Amazon rete, il che aumenta la sicurezza dei dati condivisi con le applicazioni basate sul cloud, poiché riduce l'esposizione dei dati

alla rete Internet pubblica. Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un endpoint VPC di interfaccia nella Amazon VPC User Guide.

In questa sezione:

Considerazioni sugli endpoint VPC Amazon Braket

Prima di configurare un endpoint VPC di interfaccia per Braket, assicurati di esaminare i prerequisiti dell'endpoint dell'interfaccia nella Amazon VPC User Guide.

Braket supporta l'esecuzione di chiamate a tutte le sue azioni API dal tuo VPC.

Per impostazione predefinita, l'accesso completo a Braket è consentito tramite l'endpoint VPC. Puoi controllare l'accesso se specifichi le policy degli endpoint VPC. Per ulteriori informazioni, consulta Controllo degli accessi agli endpoint VPC tramite le policy degli endpoint nella Guida per l'utente di Amazon VPC.

Configura Braket e PrivateLink

Per utilizzarlo AWS PrivateLink con Amazon Braket, devi creare un endpoint Amazon Virtual Private Cloud (Amazon VPC) come interfaccia e quindi connetterti all'endpoint tramite il servizio Amazon Braket. API

Ecco i passaggi generali di questo processo, spiegati in dettaglio nelle sezioni successive.

- Configura e avvia un Amazon VPC per ospitare le tue AWS risorse. Se hai già un VPC, questa fase può essere ignorata.
- Crea un endpoint Amazon VPC per Braket
- Connect ed esegui le attività quantistiche di Braket tramite il tuo endpoint

Fase 1: Avvia un Amazon VPC, se necessario

Ricorda che puoi saltare questo passaggio se il tuo account ha già un VPC in funzione.

Un VPC controlla le impostazioni di rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. In sostanza, stai lanciando le tue AWS risorse in una rete virtuale personalizzata. Per ulteriori informazioni VPCs, consulta la Amazon VPC User Guide.

Apri la console Amazon VPC e crea un nuovo VPC con sottoreti, gruppi di sicurezza e gateway di rete.

Passaggio 2: creare un endpoint VPC di interfaccia per Braket

Puoi creare un endpoint VPC per il servizio Braket utilizzando la console Amazon VPC o (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta Creare un endpoint VPC nella Amazon VPC User Guide.

Per creare un endpoint VPC nella console, apri la console Amazon VPC, apri la pagina Endpoints e procedi con la creazione del nuovo endpoint. Prendi nota dell'ID dell'endpoint per riferimento successivo. È obbligatorio come parte del `–endpoint-url` flag quando si effettuano determinate chiamate al API Braket.

Crea l'endpoint VPC per Braket utilizzando il seguente nome di servizio:

- `com.amazonaws.substitute_your_region.braket`

Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un endpoint VPC di interfaccia nella Amazon VPC User Guide.

Passaggio 3: Connect ed esegui le attività quantistiche di Braket tramite il tuo endpoint

Dopo aver creato un endpoint VPC, puoi eseguire comandi CLI che includono il `endpoint-url` parametro per specificare gli endpoint dell'interfaccia al runtime API or, come l'esempio seguente:

```
aws braket search-quantum-tasks --endpoint-url
 VPC_Endpoint_ID.braket.substituteYourRegionHere.vpce.amazonaws.com
```

Se abiliti i nomi host DNS privati per il tuo endpoint VPC, non è necessario specificare l'endpoint come URL nei comandi CLI. Invece, il nome host API DNS di Amazon Braket, utilizzato di default da CLI e Braket SDK, si risolve nell'endpoint VPC. Ha la forma mostrata nell'esempio seguente:

```
https://braket.substituteYourRegionHere.amazonaws.com
```

Il post sul blog intitolato Accesso diretto SageMaker ai notebook Amazon AI da Amazon VPC utilizzando un AWS PrivateLink endpoint fornisce un esempio di come configurare un endpoint per stabilire connessioni sicure ai notebook, simili ai SageMaker notebook Braket. Amazon

Se stai seguendo i passaggi descritti nel post del blog, ricordati di sostituire Amazon SageMaker AI con il nome AmazonBraket. Per Service Name inserisci `com.amazonaws.us-east-1.braket` o sostituisci il tuo Regione AWS nome corretto in quella stringa, se la tua regione non è us-east-1.

Informazioni aggiuntive sulla creazione di un endpoint

- Per informazioni su come creare un VPC con sottoreti private, consulta Creare un VPC con sottoreti private.
- Per informazioni sulla creazione e configurazione di un endpoint utilizzando la console Amazon VPC o la AWS CLI, consulta Create a VPC endpoint nella Amazon VPC User Guide.
- Per informazioni sulla creazione e configurazione di un endpoint utilizzando CloudFormation, consulta la risorsa AWS: :EC2:: nella Guida per l'utente. VPCEndpoint CloudFormation

Controlla l'accesso con le policy degli endpoint di Amazon VPC

Per controllare l'accesso alla connettività ad Amazon Braket, puoi allegare una policy di endpoint AWS Identity and Access Management (IAM) al tuo endpoint Amazon VPC. Questa policy specifica le informazioni riportate di seguito:

- Il principale (utente o ruolo) che può eseguire azioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire azioni.

Per ulteriori informazioni, consulta Controllo degli accessi agli endpoint VPC tramite le policy degli endpoint nella Guida per l'utente di Amazon VPC.

Esempio: policy degli endpoint VPC per le azioni Braket

L'esempio seguente mostra una policy sugli endpoint per Braket. Se collegata a un endpoint, questa politica consente l'accesso alle azioni Braket elencate per tutti i principali su tutte le risorse.

```
{
 "Statement":[
```

```
 {
   "Principal":"*",
   "Effect":"Allow",
   "Action":[
     “braket:action-1",
     “braket:action-2",
     “braket:action-3”
     ],
   "Resource":"*"
   }
  ]
}
```

Puoi creare regole IAM complesse allegando più policy per gli endpoint. Per ulteriori informazioni ed esempi, consulta:

- Policy degli endpoint di Amazon Virtual Private Cloud per Step Functions
- Creazione di autorizzazioni IAM granulari per utenti non amministratori
- Controlla l'accesso agli endpoint VPC utilizzando le policy degli endpoint

Registrazione di log e monitoraggio

Dopo aver inviato un'attività quantistica tramite il servizio Amazon Braket, puoi monitorare attentamente lo stato e la progressione di tale attività tramite l'SDK e la console Amazon Braket. Questo ti fornisce un'interfaccia centralizzata per tracciare l'implementazione dei tuoi carichi di lavoro, identificare eventuali strozzature o problemi e intraprendere le azioni appropriate per ottimizzare le prestazioni e l'affidabilità delle tue applicazioni quantistiche. Una volta completata l'attività quantistica, Braket salva i risultati nella posizione Amazon S3 specificata. Il tempo di completamento delle attività quantistiche può variare, in particolare per quelle eseguite su dispositivi QPU (Quantum Processing Unit). Ciò è dovuto in gran parte alla lunghezza della coda di esecuzione, poiché le risorse hardware quantistiche sono condivise tra più utenti.

Elenco dei tipi di stato:

- `CREATED`— Amazon Braket ha ricevuto il tuo compito quantistico.
- `QUEUED`— Amazon Braket ha elaborato il tuo task quantistico ed è ora in attesa di essere eseguito sul dispositivo.
- `RUNNING`— La tua attività quantistica è in esecuzione su una QPU o un simulatore on-demand.
- `COMPLETED`— L'esecuzione del task quantistico è terminata sulla QPU o sul simulatore on-demand.
- `FAILED`— Il tuo task quantistico ha tentato di essere eseguito e non è riuscito. A seconda del motivo per cui il task quantistico non è riuscito, provate a inviarlo nuovamente.
- `CANCELLED`— Hai annullato l'operazione quantistica. Il task quantistico non è stato eseguito.

In questa sezione:

Monitoraggio delle attività quantistiche dall'SDK Amazon Braket

Il comando `device.run(…)` definisce un'attività quantistica con un ID di attività quantistica univoco. È possibile interrogare e tenere traccia dello stato con `task.state()` come mostrato nell'esempio seguente.

Nota: `task = device.run()` è un'operazione asincrona, il che significa che potete continuare a lavorare mentre il sistema elabora il vostro compito quantistico in background.

Recupera un risultato

Quando chiami`task.result()`, l'SDK inizia a interrogare Amazon Braket per vedere se l'attività quantistica è completa. L'SDK utilizza i parametri di polling definiti in. `.run()` Una volta completata l'attività quantistica, l'SDK recupera il risultato dal bucket S3 e lo restituisce come oggetto. `QuantumTaskResult`

```
# create a circuit, specify the device and run the circuit
circ = Circuit().rx(0, 0.15).ry(1, 0.2).cnot(0,2)
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
task = device.run(circ, s3_location, shots=1000)

# get ID and status of submitted task
task_id = task.id
status = task.state()
print('ID of task:', task_id)
print('Status of task:', status)
# wait for job to complete
while status != 'COMPLETED':
    status = task.state()
    print('Status:', status)
```

```
ID of task:
arn:aws:braket:us-west-2:123412341234:quantum-task/b68ae94b-1547-4d1d-aa92-1500b82c300d
Status of task: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: QUEUED
Status: RUNNING
```

```
Status: RUNNING
Status: COMPLETED
```

Annulla un'attività quantistica

Per annullare un'operazione quantistica, chiamate il `cancel()` metodo, come illustrato nell'esempio seguente.

```
# cancel quantum task
task.cancel()
status = task.state()
print('Status of task:', status)
```

```
Status of task: CANCELLING
```

Controllate i metadati

È possibile controllare i metadati dell'operazione quantistica completata, come mostrato nell'esempio seguente.

```
# get the metadata of the quantum task
metadata = task.metadata()
# example of metadata
shots = metadata['shots']
date = metadata['ResponseMetadata']['HTTPHeaders']['date']
# print example metadata
print("{} shots taken on {}.".format(shots, date))

# print name of the s3 bucket where the result is saved
results_bucket = metadata['outputS3Bucket']
print('Bucket where results are stored:', results_bucket)
# print the s3 object key (folder name)
results_object_key = metadata['outputS3Directory']
print('S3 object key:', results_object_key)

# the entire look-up string of the saved result data
look_up = 's3://'+results_bucket+'/'+results_object_key
print('S3 URI:', look_up)
```

```
1000 shots taken on Wed, 05 Aug 2020 14:44:22 GMT.
Bucket where results are stored: amazon-braket-123412341234
```

```
S3 object key: simulation-output/b68ae94b-1547-4d1d-aa92-1500b82c300d
S3 URI: s3://amazon-braket-123412341234/simulation-output/b68ae94b-1547-4d1d-
aa92-1500b82c300d
```

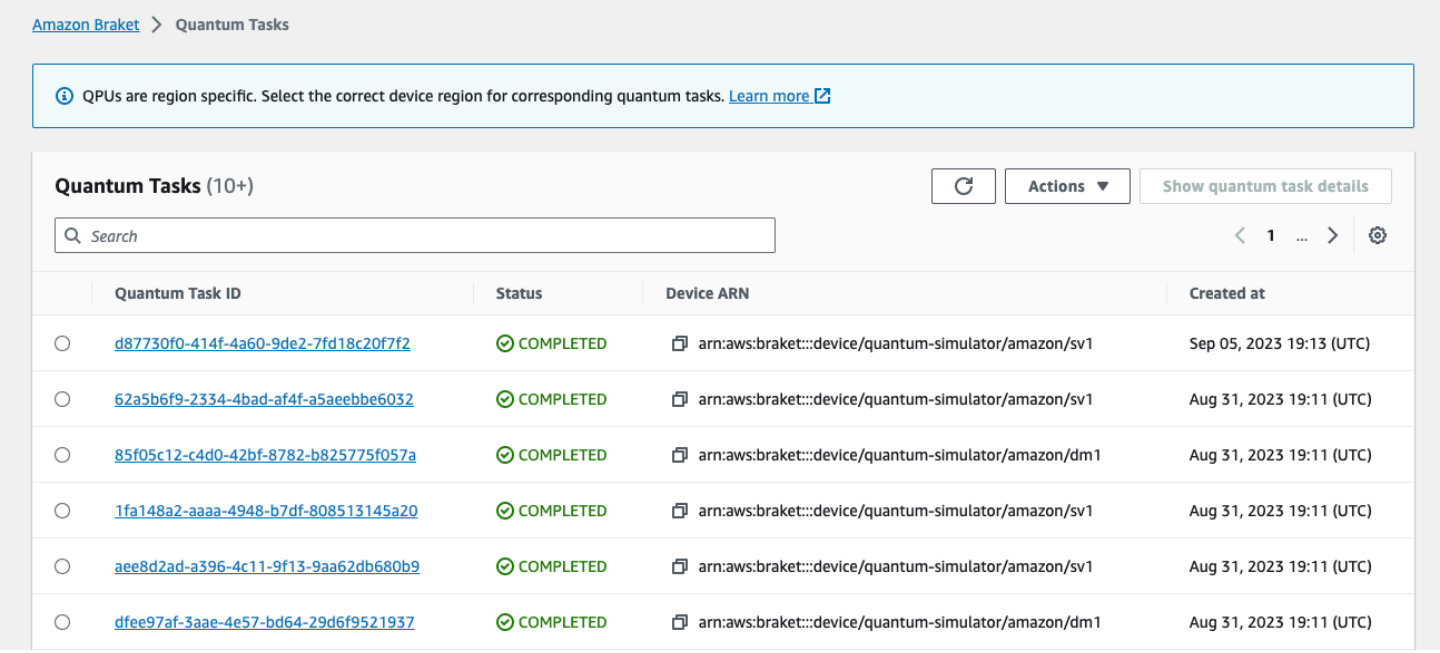
Recuperate un'attività o un risultato quantistico

Se il kernel muore dopo aver inviato il task quantistico o se chiudi il notebook o il computer, puoi ricostruire l'`task`oggetto con il suo ARN (quantum task ID) univoco. Quindi puoi chiamare `task.result()` per ottenere il risultato dal bucket S3 in cui è memorizzato.

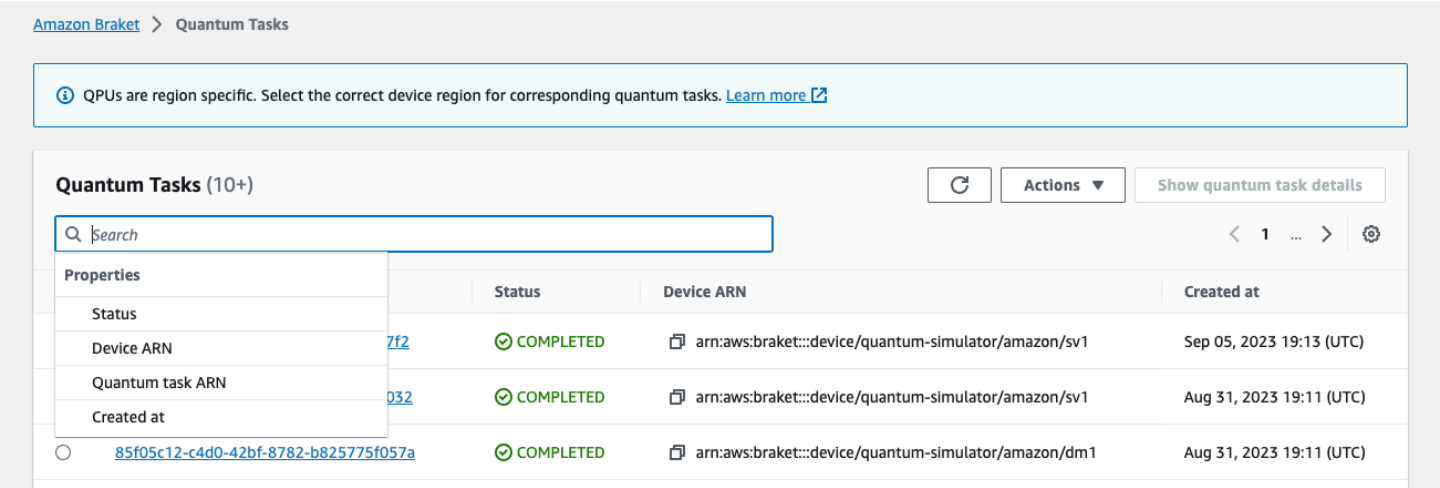
```
from braket.aws import AwsSession, AwsQuantumTask

# restore task with unique arn
task_load = AwsQuantumTask(arn=task_id)
# retrieve the result of the task
result = task_load.result()
```

Monitoraggio delle attività quantistiche tramite la console Amazon Braket

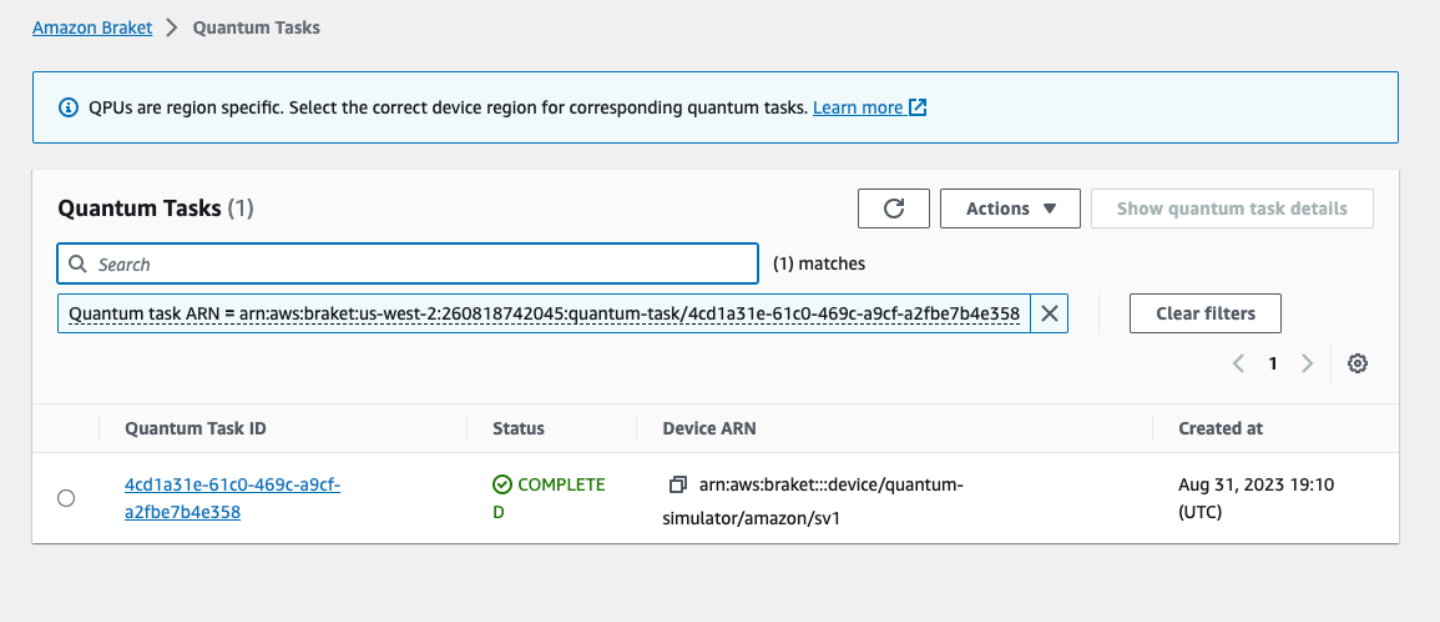
Amazon Braket offre un modo pratico per monitorare l'attività quantistica tramite la console Amazon Braket. Tutte le attività quantistiche inviate sono elencate nel campo Attività quantistiche, come mostrato nella figura seguente. Questo servizio è specifico della regione, il che significa che è possibile visualizzare solo le attività quantistiche create nello specifico. Regione AWS



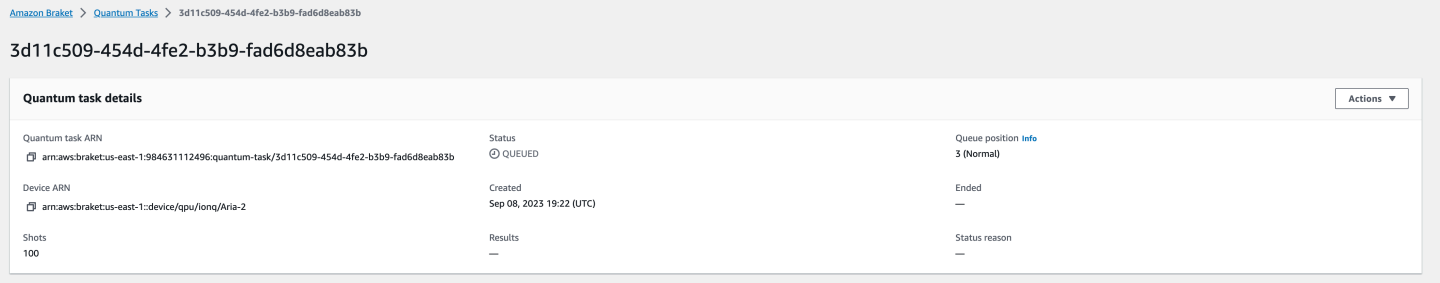
Puoi cercare attività quantistiche particolari tramite la barra di navigazione. La ricerca può essere basata su Quantum Task ARN (ID), stato, dispositivo e ora di creazione. Le opzioni vengono visualizzate automaticamente quando si seleziona la barra di navigazione, come illustrato nell'esempio seguente.



L'immagine seguente mostra un esempio di ricerca di un'attività quantistica in base al relativo ID univoco di attività quantistica, che può essere ottenuto chiamando. `task.id`



Inoltre, come mostrato nella figura seguente, lo stato di un'attività quantistica può essere monitorato mentre si trova in uno stato. QUEUED Facendo clic sull'ID dell'attività quantistica viene visualizzata la pagina dei dettagli. Questa pagina mostra la posizione dinamica della coda per il task quantistico rispetto al dispositivo su cui verrà elaborato.



Le attività quantistiche inviate come parte di un processo ibrido avranno la priorità quando sono in coda. Le attività quantistiche inviate al di fuori di un lavoro ibrido avranno la normale priorità di messa in coda.

I clienti che desiderano interrogare Braket SDK possono ottenere in modo programmatico le posizioni relative alle attività quantistiche e alle posizioni ibride dei lavori in coda. Per ulteriori informazioni, consulta la pagina Quando verrà eseguita la mia attività.

Etichettare le risorse di Amazon Braket

Un tag è un'etichetta di attributo personalizzata che si assegna o che si assegna a AWS una risorsa. AWS Un tag è un metadato che fornisce ulteriori informazioni sulla risorsa. Ciascun tag è formato da una chiave e da un valore, Tutti questi sono noti come coppie chiave-valore. Per i tag assegnati da te, puoi definire la chiave e il valore.

Nella console Amazon Braket, puoi accedere a un'attività quantistica o a un notebook e visualizzare l'elenco dei tag ad esso associati. Puoi aggiungere un tag, rimuovere un tag o modificare un tag. È possibile etichettare un'attività quantistica o un taccuino al momento della creazione e quindi gestire i tag associati tramite la console AWS CLI, oppureAPI.

Ulteriori informazioni AWS e tag

- Per informazioni generali sull'etichettatura, incluse le convenzioni di denominazione e utilizzo, consulta Cos'è Tag Editor? nella Guida per l'utente di Tagging AWS Resources and Tag Editor.
- Per informazioni sulle restrizioni relative all'etichettatura, consulta Limiti e requisiti di denominazione dei tag nella Guida per l'utente di Tagging AWS Resources and Tag Editor.
- Per le best practice e le strategie di tagging, consulta Best Practices for Tagging AWS Resources.
- Per un elenco dei servizi che supportano l'utilizzo dei tag, consulta l'argomento della documentazione di riferimento delle API per l'applicazione di tag a Resource Groups.

Le seguenti sezioni forniscono informazioni più specifiche sui tag per Amazon Braket.

In questa sezione:

Utilizzo dei tag

I tag possono organizzare le tue risorse in categorie che ti sono utili. Ad esempio, puoi assegnare un tag «Dipartimento» per specificare il dipartimento che possiede questa risorsa.

Ogni tag è costituito da due parti:

- Una chiave di tag (ad esempio CostCenter, Ambiente o Progetto). Le chiavi dei tag distinguono tra maiuscole e minuscole
- Un campo opzionale noto come valore di tag (ad esempio, 111122223333 o Production). Non specificare il valore del tag equivale a utilizzare una stringa vuota. Come le chiavi dei tag, i valori dei tag distinguono tra maiuscole e minuscole.

I tag ti aiutano a fare le seguenti cose:

- Identifica e organizza AWS le tue risorse. Molti Servizi AWS supportano l'etichettatura, quindi puoi assegnare lo stesso tag a risorse di servizi diversi per indicare che le risorse sono correlate.
- Tieni traccia dei costi. AWS Attivi questi tag sulla Gestione dei costi e fatturazione AWS dashboard. AWS utilizza i tag per classificare i costi e fornirti un rapporto mensile sull'allocazione dei costi. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella Gestione dei costi e fatturazione AWS Guida per l'utente.
- Controlla l'accesso alle tue risorse. AWS Per ulteriori informazioni, consulta Controllo dell'accesso tramite tag.

Risorse supportate per l'etichettatura in Amazon Braket

Il seguente tipo di risorsa in Amazon Braket supporta l'etichettatura:

- Risorsa `quantum-task`
- Nome della risorsa: `AWS::Service::Braket`
- Regex ARN: `arn:${Partition}:braket:${Region}:${Account}:quantum-task/${RandomId}`

Nota: puoi applicare e gestire i tag per i tuoi notebook Amazon Braket nella console Amazon Braket, utilizzando la console per accedere alla risorsa notebook, sebbene i notebook siano in realtà risorse Amazon AI. SageMaker Per ulteriori informazioni, consulta Notebook Instance Metadata nella documentazione. SageMaker

Tagging con la Amazon Braket API

- Se utilizzi Amazon Braket API per configurare i tag su una risorsa, chiama il. `TagResource`API

```
aws braket tag-resource --resource-arn $YOUR_TASK_ARN --tags {\"city\":
\"Seattle\"}
```

- Per rimuovere i tag da una risorsa, chiama il `UntagResource`API.

```
aws braket list-tags-for-resource --resource-arn $YOUR_TASK_ARN
```

- Per elencare tutti i tag associati a una particolare risorsa, chiama il `ListTagsForResource`API.

```
aws braket tag-resource --resource-arn $YOUR_TASK_ARN --tag-keys "[\"city
\",\"state\"]"
```

Restrizioni di tagging

Le seguenti restrizioni di base si applicano ai tag nelle risorse Amazon Braket:

- Numero massimo di tag che puoi assegnare a una risorsa: 50
- lunghezza massima della chiave: 128 caratteri Unicode;
- lunghezza massima del valore: 256 caratteri Unicode;

- Caratteri validi per chiave e valore: `a-z, A-Z, 0-9, space` e questi caratteri: `_ . : / = + -` e `@`
- Le chiavi e i valori fanno distinzione tra maiuscole e minuscole.
- Non utilizzare `aws` come prefisso per le chiavi; è riservato all' AWS uso.

Gestione dei tag in Amazon Braket

Si impostano i tag come proprietà su una risorsa. Puoi visualizzare, aggiungere, modificare, elencare ed eliminare i tag tramite la console Amazon Braket, Amazon API Braket o. AWS CLI Per ulteriori informazioni, consulta il riferimento all'API Amazon Braket.

In questa sezione:

- Aggiunta di tag
- Visualizzazione dei tag
- Modifica dei tag
- Rimozione dei tag

Aggiunta di tag

Puoi aggiungere tag alle risorse taggabili nei seguenti orari:

- Quando crei la risorsa: utilizza la console o includi il `Tags` parametro con l'`Create`operazione nell'AWS API.
- Dopo aver creato la risorsa: utilizza la console per accedere alla risorsa Quantum Task o Notebook oppure richiama l'`TagResource`operazione nell'AWS API.

Per aggiungere tag a una risorsa al momento della creazione, è inoltre necessaria l'autorizzazione per creare una risorsa del tipo specificato.

Visualizzazione dei tag

Puoi visualizzare i tag su qualsiasi risorsa etichettabile in Amazon Braket utilizzando la console per accedere all'attività o alla risorsa notebook oppure chiamando l'operazione. AWS `ListTagsForResource` API

Puoi usare il seguente AWS API comando per visualizzare i tag su una risorsa:

- AWS API: `ListTagsForResource`

Modifica dei tag

È possibile modificare i tag utilizzando la console per accedere all'attività quantistica o alla risorsa notebook oppure è possibile utilizzare il comando seguente per modificare il valore di un tag allegato a una risorsa etichettabile. Quando specificate una chiave di tag già esistente, il valore di quella chiave viene sovrascritto:

- AWS API: `TagResource`

Rimozione dei tag

È possibile rimuovere i tag da una risorsa specificando le chiavi da rimuovere, utilizzando la console per accedere alla risorsa Quantum Task o Notebook o quando si richiama l'operazione. `UntagResource`

- AWS API: `UntagResource`

Esempio di AWS CLI etichettatura in Amazon Braket

Quando lavori con AWS Command Line Interface (AWS CLI) per interagire con Amazon Braket, il codice seguente è un comando di esempio che dimostra come creare un tag che si applica a un'attività quantistica che crei. In questo esempio, l'attività viene eseguita sul simulatore SV1 quantistico con le impostazioni dei parametri specificate per l'unità di elaborazione Rigetti quantistica (QPU). È importante che all'interno del comando di esempio il tag sia specificato alla fine, dopo tutti gli altri parametri richiesti. In questo caso, il tag ha una chiave di `state` e un valore di. `Washington` Questi tag potrebbero essere usati per aiutare a classificare o identificare questo particolare compito quantistico.

```
aws braket create-quantum-task --action /
"{\"braketSchemaHeader\": {\"name\": \"braket.ir.jaqcd.program\", /
    \"version\": \"1\"}, /
    \"instructions\": [{\"angle\": 0.15, \"target\": 0, \"type\": \"rz\"}], /
    \"results\": null, /
    \"basis_rotation_instructions\": null}" /
  --device-arn "arn:aws:braket:::device/quantum-simulator/amazon/sv1" /
  --output-s3-bucket  "my-example-braket-bucket-name" /
```

```
--output-s3-key-prefix "my-example-username"  /
--shots 100   /
--device-parameters /
"{\"braketSchemaHeader\": /
   {\"name\": \"braket.device_schema.rigetti.rigetti_device_parameters\", /
    \"version\": \"1\"}, \"paradigmParameters\": /
     {\"braketSchemaHeader\": /
       {\"name\": \"braket.device_schema.gate_model_parameters\", /
        \"version\": \"1\"}, /
        \"qubitCount\": 2}}" /
        --tags {\"state\":\”Washington\"}
```

Questo esempio dimostra come è possibile applicare tag alle attività quantistiche quando le si esegue tramite il AWS CLI, il che è utile per organizzare e tracciare le risorse Braket.

Monitoraggio delle attività quantistiche con EventBridge

Amazon EventBridge monitora gli eventi di modifica dello stato nelle attività quantistiche di Amazon Braket. Gli eventi di Amazon Braket vengono consegnati EventBridge quasi in tempo reale. Puoi scrivere regole che indicano quali eventi ti interessano, incluse azioni automatiche da intraprendere quando un evento corrisponde a una regola. Le azioni automatiche che possono essere attivate includono:

- Invocare una funzione AWS Lambda
- Attivazione di una macchina a stati AWS Step Functions
- Notifica di un argomento Amazon SNS

EventBridge monitora questi eventi di modifica dello stato di Amazon Braket:

- Lo stato delle attività quantistiche cambia

Amazon Braket garantisce la fornitura di eventi di modifica dello stato delle attività quantistiche. Questi eventi vengono forniti almeno una volta, ma potrebbero non funzionare correttamente.

Per ulteriori informazioni, consulta la sezione Eventi in Amazon EventBridge.

In questa sezione:

- Monitora lo stato delle attività quantistiche con EventBridge

- Esempio di evento Amazon Braket EventBridge

Monitora lo stato delle attività quantistiche con EventBridge

Con EventBridge, puoi creare regole che definiscono le azioni da intraprendere quando Amazon Braket invia una notifica di una modifica dello stato relativa a un'attività quantistica di Braket. Ad esempio, puoi creare una regola che ti invii un messaggio e-mail ogni volta che lo stato di un'attività quantistica cambia.

1. Accedi AWS utilizzando un account con autorizzazioni di utilizzo EventBridge e Amazon Braket.
2. Apri la EventBridge console Amazon.
3. Utilizzando i seguenti valori, crea una EventBridge regola:
   - Per Tipo di regola, scegli Regola con un modello di eventi.
   - In Event source (Origine eventi), scegli Other (Altro).
   - Nella sezione Schema di eventi, scegliete Modelli personalizzati (editor JSON), quindi incollate il seguente modello di evento nell'area di testo:

     ```
     {
       "source": [
         "aws.braket"
       ],
       "detail-type": [
         "Braket Task State Change"
       ]
     }
     ```

     Per acquisire tutti gli eventi da Amazon Braket, escludi la `detail-type` sezione come mostrato nel codice seguente:

     ```
     {
       "source": [
         "aws.braket"
       ]
     }
     ```

   - Per i tipi di Target Servizio AWS, scegli, e per Seleziona un target, scegli un target come un argomento o AWS Lambda una funzione di Amazon SNS. Il target viene attivato quando Braket riceve un evento di modifica dello stato di un'attività quantistica. Amazon

Ad esempio, utilizza un argomento Amazon Simple Notification Service (SNS) per inviare un'e-mail o un messaggio di testo quando si verifica un evento. A tale scopo, crea innanzitutto un argomento Amazon SNS utilizzando la console Amazon SNS. Per ulteriori informazioni, consulta Utilizzo di Amazon SNS per notifiche all'utente.

Per informazioni dettagliate sulla creazione di regole, consulta Creazione di EventBridge regole Amazon che reagiscono agli eventi.

Esempio di evento Amazon Braket EventBridge

Per informazioni sui campi per un evento Amazon Braket Quantum Task Status Change, consulta Eventi in Amazon. EventBridge

I seguenti attributi vengono visualizzati nel campo «dettaglio» JSON.

- **`quantumTaskArn`**(str): L'attività quantistica per la quale è stato generato questo evento.
- **`status`**(Opzionale [str]): Lo stato a cui è passata l'attività quantistica.
- **`deviceArn`**(str): Il dispositivo specificato dall'utente per il quale è stata creata questa operazione quantistica.
- shots(int): Il numero di shots richieste dall'utente.
- **`outputS3Bucket`**(str): Il bucket di output specificato dall'utente.
- **`outputS3Directory`**(str): Il prefisso della chiave di output specificato dall'utente.
- **`createdAt`**(str): L'ora di creazione dell'attività quantistica come stringa ISO-8601.
- **`endedAt`**(Opzionale [str]): L'ora in cui l'attività quantistica ha raggiunto uno stato terminale. Questo campo è presente solo quando l'attività quantistica è passata a uno stato terminale.

Il codice JSON seguente mostra un esempio di evento Amazon Braket Quantum Task Status Change.

```
{
    "version":"0",
    "id":"6101452d-8caf-062b-6dbc-ceb5421334c5",
    "detail-type":"Braket Task State Change",
    "source":"aws.braket",
    "account":"012345678901",
```

```
    "time":"2021-10-28T01:17:45Z",
    "region":"us-east-1",
    "resources":[
        "arn:aws:braket:us-east-1:012345678901:quantum-task/834b21ed-77a7-4b36-a90c-
c776afc9a71e"
    ],
    "detail":{
        "quantumTaskArn":"arn:aws:braket:us-east-1:012345678901:quantum-
task/834b21ed-77a7-4b36-a90c-c776afc9a71e",
        "status":"COMPLETED",
        "deviceArn":"arn:aws:braket:::device/quantum-simulator/amazon/sv1",
        "shots":"100",
        "outputS3Bucket":"amazon-braket-0260a8bc871e",
        "outputS3Directory":"sns-testing/834b21ed-77a7-4b36-a90c-c776afc9a71e",
        "createdAt":"2021-10-28T01:17:42.898Z",
        "eventName":"MODIFY",
        "endedAt":"2021-10-28T01:17:44.735Z"
      }
}
```

Monitoraggio delle metriche con CloudWatch

Puoi monitorare Amazon Braket utilizzando Amazon CloudWatch, che raccoglie dati grezzi e li elabora in metriche leggibili quasi in tempo reale. Visualizza le informazioni storiche generate fino a 15 mesi fa o le metriche di ricerca aggiornate nelle ultime 2 settimane nella CloudWatch console Amazon per avere una prospettiva migliore sulle prestazioni di Amazon Braket. Per ulteriori informazioni, consulta Utilizzo CloudWatch delle metriche.

> **Note**
>
> Puoi visualizzare i flussi di CloudWatch log per i notebook Amazon Braket accedendo alla pagina dei dettagli di Notebook sulla console Amazon AI. SageMaker Ulteriori impostazioni del notebook Amazon Braket sono disponibili tramite la SageMaker console.

In questa sezione:

- Metriche e dimensioni di Amazon Braket

Metriche e dimensioni di Amazon Braket

Le metriche sono il concetto fondamentale in. CloudWatch Una metrica rappresenta un insieme di punti dati ordinati nel tempo su cui vengono pubblicati. CloudWatch Ogni metrica è caratterizzata da un insieme di dimensioni. Per ulteriori informazioni sulle dimensioni delle metriche in CloudWatch, consulta CloudWatch Dimensioni.

Amazon Braket invia i seguenti dati metrici, specifici di Amazon Braket, ai parametri Amazon: CloudWatch

Quantum Task Metrics

Le metriche sono disponibili se esistono attività quantistiche. Vengono visualizzate in AWS/Braket/By Device nella console. CloudWatch

Metrica	Description
Conteggio	Numero di attività quantistiche.
Latenza	Questa metrica viene emessa quando viene completata un'attività quantistica. Rappresenta il tempo totale dall'inizializzazione al completamento dell'attività quantistica.

Dimensioni per Quantum Task Metrics

Le metriche delle attività quantistiche sono pubblicate con una dimensione basata sul **`deviceArn`** parametro, che ha la forma arn:aws:braket: ::device/xxx.

Registrazione delle attività quantistiche con CloudTrail

Amazon Braket è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, da un ruolo o da un utente Servizio AWS in Amazon Braket. CloudTrail acquisisce tutte le API chiamate per Amazon Braket come eventi. Le chiamate acquisite includono chiamate dalla console Amazon Braket e chiamate in codice verso le operazioni Amazon Braket. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon Braket. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le

informazioni raccolte da CloudTrail, puoi determinare la richiesta che è stata effettuata ad Amazon Braket, l'indirizzo IP da cui è stata effettuata, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la Guida per l'AWS CloudTrail utente.

In questa sezione:

- Informazioni su Amazon Braket in CloudTrail
- Comprendere le voci dei file di log di Amazon Braket

Informazioni su Amazon Braket in CloudTrail

CloudTrail è abilitato sul tuo Account AWS quando crei l'account. Quando si verifica un'attività in Amazon Braket, tale attività viene registrata in un CloudTrail evento insieme ad altri Servizio AWS eventi nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti nel tuo Account AWS. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi del tuo sito Account AWS, compresi gli eventi per Amazon Braket, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un percorso nella console, questo sarà valido in tutte le Regioni AWS. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurarne altri Servizi AWS per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte le azioni di Amazon Braket vengono registrate da. CloudTrail Ad esempio, le chiamate alle `GetDevice` azioni `GetQuantumTask` o generano voci nei file di CloudTrail registro.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro Servizio AWS.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Comprendere le voci dei file di log di Amazon Braket

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia stack ordinata delle API chiamate pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente è una voce di registro per l'`GetQuantumTask`azione, che ottiene i dettagli di un'attività quantistica.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "foobar",
    "arn": "foobar",
    "accountId": "foobar",
    "accessKeyId": "foobar",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "foobar",
        "arn": "foobar",
        "accountId": "foobar",
        "userName": "foobar"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-08-07T00:56:57Z"
      }
    }
  },
```

```
  "eventTime": "2020-08-07T01:00:08Z",
  "eventSource": "braket.amazonaws.com",
  "eventName": "GetQuantumTask",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "foobar",
  "userAgent": "aws-cli/1.18.110 Python/3.6.10
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.17.33",
  "requestParameters": {
    "quantumTaskArn": "foobar"
  },
  "responseElements": null,
  "requestID": "20e8000c-29b8-4137-9cbc-af77d1dd12f7",
  "eventID": "4a2fdb22-a73d-414a-b30f-c0797c088f7c",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "foobar"
}
```

Di seguito viene mostrata una voce di registro relativa all'`GetDevice`azione, che restituisce i dettagli di un evento del dispositivo.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "foobar",
    "arn": "foobar",
    "accountId": "foobar",
    "accessKeyId": "foobar",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "foobar",
        "arn": "foobar",
        "accountId": "foobar",
        "userName": "foobar"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-08-07T00:46:29Z"
      }
    }
```

```
  },
  "eventTime": "2020-08-07T00:46:32Z",
  "eventSource": "braket.amazonaws.com",
  "eventName": "GetDevice",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "foobar",
  "userAgent": "Boto3/1.14.33 Python/3.7.6 Linux/4.14.158-129.185.amzn2.x86_64 exec-
env/AWS_ECS_FARGATE Botocore/1.17.33",
  "errorCode": "404",
  "requestParameters": {
    "deviceArn": "foobar"
  },
  "responseElements": null,
  "requestID": "c614858b-4dcf-43bd-83c9-bcf9f17f522e",
  "eventID": "9642512a-478b-4e7b-9f34-75ba5a3408eb",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "foobar"
}
```

Registrazione avanzata con Amazon Braket

È possibile registrare l'intero processo di elaborazione delle attività utilizzando un logger. Queste tecniche di registrazione avanzate consentono di visualizzare il polling in background e di creare un record per il debug successivo.

Per utilizzare il logger, consigliamo di modificare i `poll_interval_seconds` parametri `poll_timeout_seconds` and, in modo che un'attività quantistica possa durare a lungo e lo stato dell'attività quantistica venga registrato continuamente, con i risultati salvati in un file. È possibile trasferire questo codice su uno script Python anziché su un notebook Jupyter, in modo che lo script possa essere eseguito come processo in background.

Configura il logger

Innanzitutto, configura il logger in modo che tutti i registri vengano scritti automaticamente in un file di testo, come mostrato nelle seguenti righe di esempio.

```
# import the module
import logging
from datetime import datetime

# set filename for logs
```

```
log_file = 'device_logs-'+datetime.strftime(datetime.now(), '%Y%m%d%H%M%S')+'.txt'
print('Task info will be logged in:', log_file)

# create new logger object
logger = logging.getLogger("newLogger")

# configure to log to file device_logs.txt in the appending mode
logger.addHandler(logging.FileHandler(filename=log_file, mode='a'))

# add to file all log messages with level DEBUG or above
logger.setLevel(logging.DEBUG)
```

```
Task info will be logged in: device_logs-20200803203309.txt
```

Crea ed esegui il circuito

Ora puoi creare un circuito, inviarlo a un dispositivo per farlo funzionare e vedere cosa succede, come mostrato in questo esempio.

```
# define circuit
circ_log = Circuit().rx(0, 0.15).ry(1, 0.2).rz(2, 0.25).h(3).cnot(control=0,
 target=2).zz(1, 3, 0.15).x(4)
print(circ_log)
# define backend
device = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
# define what info to log
logger.info(
    device.run(circ_log, s3_location,
               poll_timeout_seconds=1200, poll_interval_seconds=0.25, logger=logger,
 shots=1000)
    .result().measurement_counts
)
```

Controllate il file di registro

Puoi controllare cosa è scritto nel file inserendo il seguente comando.

```
# print logs
! cat {log_file}
```

```
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: start polling for completion
```

```
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status CREATED
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status CREATED
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status QUEUED
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status RUNNING
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status RUNNING
Task arn:aws:braket:us-west-2:123412341234:quantum-
task/5088ec6c-89cf-4338-9750-9f5bb12a0dc4: task status COMPLETED
Counter({'00001': 493, '00011': 493, '01001': 5, '10111': 4, '01011': 3, '10101': 2})
```

Ottieni l'ARN dal file di registro

Dall'output del file di registro restituito, come mostrato nell'esempio precedente, è possibile ottenere le informazioni ARN. Con l'ID ARN, è possibile recuperare il risultato dell'attività quantistica completata.

```
# parse log file for arn
with open(log_file) as openfile:
    for line in openfile:
        for part in line.split():
            if "arn:" in part:
                arn = part
                break
# remove final semicolon in logs
arn = arn[:-1]

# with this arn you can restore again task from unique arn
task_load = AwsQuantumTask(arn=arn, aws_session=AwsSession())

# get results of task
result = task_load.result()
```

Quote Amazon Braket

La tabella seguente elenca le quote di servizio per Amazon Braket. Le quote di servizio, a cui si fa riferimento anche come limiti, rappresentano il numero massimo di risorse di servizio o operazioni per l'Account AWS.

Alcune quote possono essere aumentate. Per ulteriori informazioni, consulta la pagina relativa alle quote di Servizio AWS.

- Le quote burst rate non possono essere aumentate.
- L'aumento massimo del tasso per le quote regolabili (ad eccezione del burst rate, che non può essere modificato) è 2 volte il limite di tasso predefinito specificato. Ad esempio, una quota predefinita di 60 può essere regolata fino a un massimo di 120.
- La quota regolabile per le attività quantistiche simultanee SV1 (DM1) consente un massimo di 60 per.Regione AWS
- Il numero massimo consentito di istanze di calcolo per un processo ibrido è 1 e le quote sono regolabili.

Risorsa	Description	Limits	Regolabile
Frequenza delle richieste API	Il numero massimo di richieste al secondo che puoi inviare in questo account nella regione corrente.	140	Sì
Frequenza di raffica delle richieste API	Il numero massimo di richieste aggiuntive al secondo (RPS) che puoi inviare in una espansione in questo account nella regione corrente.	600	No

Risorsa	Description	Limits	Regolabile
Frequenza delle richieste `CreateQuantumTask`	Il numero massimo di `CreateQuantumTask` richieste che puoi inviare al secondo in questo account per regione.	20 al secondo	Sì
Frequenza di burst delle richieste `CreateQuantumTask`	Il numero massimo di `CreateQuantumTask` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	40	No
Frequenza delle richieste `SearchQuantumTasks`	Il numero massimo di `SearchQuantumTasks` richieste che puoi inviare al secondo in questo account per regione.	5 al secondo	Sì
Frequenza di burst delle richieste `SearchQuantumTasks`	Il numero massimo di `SearchQuantumTasks` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No

Risorsa	Description	Limits	Regolabile
Frequenza delle richieste `GetQuantumTask`	Il numero massimo di `GetQuantumTask` richieste che puoi inviare al secondo in questo account per regione.	100 al secondo	Sì
Frequenza di burst delle richieste `GetQuantumTask`	Il numero massimo di `GetQuantumTask` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	500	No
Frequenza delle richieste `CancelQuantumTask`	Il numero massimo di `CancelQuantumTask` richieste che puoi inviare al secondo in questo account per regione.	2 al secondo	Sì
Frequenza di burst delle richieste `CancelQuantumTask`	Il numero massimo di `CancelQuantumTask` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	20	No

Risorsa	Description	Limits	Regolabile
Frequenza delle richieste `GetDevice`	Il numero massimo di `GetDevice` richieste che puoi inviare al secondo in questo account per regione.	5 al secondo	Sì
Frequenza di burst delle richieste `GetDevice`	Il numero massimo di `GetDevice` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No
Frequenza delle richieste `SearchDevices`	Il numero massimo di `SearchDevices` richieste che puoi inviare al secondo in questo account per regione.	5 al secondo	Sì
Frequenza di burst delle richieste `SearchDevices`	Il numero massimo di `SearchDevices` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No

Risorsa	Description	Limits	Regolabile
Frequenza delle richieste `CreateJob`	Il numero massimo di `CreateJob` richieste che puoi inviare al secondo in questo account per regione.	1 al secondo	Sì
Frequenza di burst delle richieste `CreateJob`	Il numero massimo di `CreateJob` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	5	No
Frequenza delle richieste `SearchJobs`	Il numero massimo di `SearchJob` richieste che puoi inviare al secondo in questo account per regione.	5 al secondo	Sì
Frequenza di burst delle richieste `SearchJobs`	Il numero massimo di `SearchJob` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	50	No

Risorsa	Description	Limits	Regolabile
Frequenza delle richieste `GetJob`	Il numero massimo di `GetJob` richieste che puoi inviare al secondo in questo account per regione.	5 al secondo	Sì
Frequenza di burst delle richieste `GetJob`	Il numero massimo di `GetJob` richieste aggiuntive al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	25	No
Frequenza delle richieste `CancelJob`	Il numero massimo di `CancelJob` richieste che puoi inviare al secondo in questo account per regione.	2 al secondo	Sì
Frequenza di burst delle richieste `CancelJob`	Il numero massimo di `CancelJob` richieste aggiuntiv e al secondo (RPS) che puoi inviare in un'unica sequenza a questo account nella regione corrente.	5	No

Risorsa	Description	Limits	Regolabile
Numero di attività quantistiche simultanee SV1	Il numero massimo di attività quantisti che simultanee in esecuzione sul simulatore vettoriale di stato () SV1 nella regione corrente.	100 us-est-1, 50 Stati Uniti - Ovest-1, 100 Stati Uniti-West-2, 50 eu-west-2	No
Numero di attività quantistiche simultanee DM1	Il numero massimo di attività quantisti che simultanee in esecuzione sul simulatore di matrice di densità (DM1) nella regione corrente.	100 us-est-1, 50 Stati Uniti - Ovest-1, 100 Stati Uniti-West-2, 50 eu-west-2	No
Numero di attività quantistiche simultanee TN1	Il numero massimo di attività quantisti che simultanee in esecuzione sul simulatore di rete tensoriale () nella regione corrente. TN1	10 us-est-1, 10 Stati Uniti - Ovest - 2, 5 eu-west-2,	Sì
Numero di lavori ibridi simultanei	Il numero massimo di lavori ibridi simultanei nella regione corrente.	3	Sì
Limite di runtime dei lavori ibridi	La quantità massima di tempo in giorni in cui un processo ibrido può essere eseguito.	5	No

Di seguito sono riportate le quote predefinite delle istanze di calcolo classiche per Hybrid Jobs. Per aumentare queste quote, contatta. Supporto Inoltre, le regioni disponibili sono specificate per ogni istanza.

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c4.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c4.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No
Numero massimo di istanze di ml.c4.2xlarge per	Il numero massimo di istanze di tipo ml.c4.2xlarge consentit	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Descrizione	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
lavori ibridi	o per tutti i lavori ibridi Amazon Braket in questo account e regione.							
Numero massimo di istanze di ml.c4.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c4.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c4.8xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c4.8xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	No	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descripti on	Limits	Regolabil e	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.4xl arge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.4xl arge consentit o per tutti i lavori ibridi Amazon Braket in questo account e regione.	1	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descripti on	Limits	Regolabil e	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.9xl arge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.9xl arge consentit o per tutti i lavori ibridi Amazon Braket in questo account e regione.	1	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5.18xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5.18xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.9xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.9xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.c5n.18xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.c5n.18xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	No	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.8xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.8xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.12xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.12xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.g4dn.16xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.g4dn.16xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Descrizione	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	2	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.10› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.10› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m4.16› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m4.16› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.large per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.2xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.2xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.4xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.4xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	5	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.12› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.12› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.m5.24› large per lavori ibridi	Il numero massimo di istanze di tipo ml.m5.24› large consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	Sì	Sì	Sì	Sì

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p2.xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p2.xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descripti on	Limits	Regolabil e	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p2.8xl arge per lavori ibridi	Il numero massimo di istanze di tipo ml.p2.8xl arge consentit o per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Description	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p2.16xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p2.16xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Risorsa	Descrizione	Limits	Regolabile	us-east-1	us-west-1	us-west-2	eu-west-2	eu-north-1
Numero massimo di istanze di ml.p4d.24xlarge per lavori ibridi	Il numero massimo di istanze di tipo ml.p4d.24xlarge consentito per tutti i lavori ibridi Amazon Braket in questo account e regione.	0	Sì	Sì	No	Sì	No	No

Richiesta di aggiornamenti dei limiti

Se ricevi un' ServiceQuotaExceeded eccezione per un tipo di istanza e non disponi di un numero sufficiente di istanze disponibili, puoi richiedere un aumento del limite dalla pagina Service Quotas AWS della console e cercare Amazon Braket nella sezione Servizi.AWS

> **Note**
>
> Se il tuo processo ibrido non è in grado di fornire la capacità di calcolo ML richiesta, utilizza un'altra regione. Inoltre, se non vedi un'istanza nella tabella, non è disponibile per Hybrid Jobs.

Quote e limiti aggiuntivi

- L'azione dell'attività quantistica di Amazon Braket ha una dimensione limitata a 3 MB.
- Infatti SV1, la durata massima di funzionamento è di 3 ore per i circuiti fino a 31 qubit e di 11 ore per i circuiti superiori a 31 qubit.
- Il numero massimo di scatti consentiti per SV1 attività e dispositivi è 50.000. DM1 Rigetti
- Il numero massimo di scatti per attività TN1 è 1000.
- Per AQT il IBEX-Q1 dispositivo, il massimo è 2000 scatti per operazione.
- Per tutti IonQ i dispositivi: il numero minimo di scatti per operazione è 100. Quando si utilizza un modello on-demand, è previsto un limite di 1 milione di gateshot e un minimo di 2500 scatti per le attività di mitigazione degli errori. Per una prenotazione diretta, non è previsto alcun limite di gateshot e un minimo di 500 shot per le attività di mitigazione degli errori.
- Per QuEra il dispositivo Aquila, il massimo è 1.000 colpi per operazione.
- Per IQM computer Garnet e Emerald dispositivi, il massimo è 20.000 scatti per attività.
- Per TN1 i dispositivi QPU, gli scatti per attività devono essere > 0.

Cronologia dei documenti per la Amazon Braket Developer Guide

La tabella seguente descrive le versioni della documentazione per Amazon Braket.

- Ultimo aggiornamento API di riferimento: 20 novembre 2025
- Ultimo aggiornamento della documentazione: 2 marzo 2026

Modifica	Descrizione	Data
Ritiro del dispositivo IonQ Aria-1	Supporto rimosso per il dispositivo IonQ Aria-1.	2 marzo 2026
Aggiorna le pagine «Lavorare con le prenotazioni»	Maggiore chiarezza delle pagine «Lavorare con le prenotazioni»	3 febbraio 2026
Supporta la versione 3.12 di Python per notebook Braket e contenitori gestiti	È stato aggiunto il supporto Python versione 3.12 per notebook e contenitori gestiti Amazon Braket (Base, CUDA-Q e Tensorflow). PennyLane Include una guida alla risoluzione dei problemi per l'aggiornamento a Python 3.12.	21 gennaio 2026
Rimuovi gli esempi P3	SageMaker sta ritirando la loro famiglia di `ml.p3` istanze. Esempi sostituiti con la famiglia di `ml.g4dn` istanze consigliata.	19 dicembre 2025
Nuova funzionalità relativa al limite di spesa	È stato aggiunto il supporto per la funzionalità di limite di spesa di Amazon Braket, che consente di impostare	20 novembre 2025

	limiti di budget opzionali per singoli individui QPUs che convalidano e rifiutano automaticamente le attività che superano la soglia di spesa configurata.	
Nuovo dispositivo Braket AQT IBEX-Q1	È stato aggiunto il supporto per un AQT IBEX-Q1dispositivo. Questo dispositivo si basa su un cristallo di ^{40} ioni $^{Ca+}$ in una trappola a radiofrequenza macroscopica situata in una camera ad altissimo vuoto.	18 novembre 2025
Nuovo supporto nativo per CUDA-Q Amazon Braket NBIs	È stato aggiunto il supporto nativo per CUDA-Q le istanze di notebook Amazon Braket. Per ulteriori informazioni, consulta CUDA-Q in. NBIs	10 novembre 2025
IonQ Aria-2ritiro del dispositivo	Supporto rimosso per il IonQ Aria-2 dispositivo.	27 ottobre 2025
Documentazione consolidata Hybrid Jobs	Le sezioni relative ai lavori ibridi sono state consolidate in modo che vengano visualizzate in Lavorare con Amazon Braket Hybrid Jobs.	21 ottobre 2025
Contenitore fornito da New Braket CUDA-Q	È stato aggiunto il supporto per un contenitore di lavori CUDA-Q ibrido fornito. Per ulteriori informazioni, consulta Definire l'ambiente per lo script dell'algoritmo.	2 settembre 2025

Nuova funzionalità di emulazione di dispositivi locali	È stato aggiunto il supporto per uno strumento di emulazione di dispositivi quantistici locale per emulare i programmi letterali prima di inviarli ai dispositivi quantistici.	25 agosto 2025
Le pagine Pennylane e CUDA-Q sono state spostate nella sezione Build	Le pagine Pennylane e CUDA-Q sono state spostate in modo che vengano visualizzate nella sezione Build del sommario.	15 agosto 2025
Nuova funzionalità `ProgramSet`	È stato aggiunto il supporto per i set di programmi, un'operazione per eseguire più circuiti quantistici in un'unica attività quantistica.	14 agosto 2025
Nuovo dispositivo IQM Emerald	È stato aggiunto il supporto per il IQM Emerald dispositivo. Un dispositivo a 54 quibit con una topologia reticolare quadrata (Crystal).	21 luglio 2025
È AmazonBraketServiceRolePolicystata aggiornata la politica	AmazonBraketServiceRolePolicyora fornisce solo le azioni s3: * e logs: * a aws:. Principal Account Ciò limita l'accesso solo ai bucket e ai gruppi di log del richiedente.	11 luglio 2025
Nuova funzionalità Experimental Capability: circuiti dinamici	La misurazione dei circuiti centrali e le operazioni di feed-forward sono disponibili come funzionalità sperimentali, vedi Accesso ai circuiti dinamici sui dispositivi IQM.	26 giugno 2025

È stata aggiornata la politica AmazonBraketFullAccess	AmazonBraketFullAccessora include i prezzi: GetProduc ts per visualizzare i costi dell'hardware sulla console.	14 aprile 2025
Nuovo dispositivo IonQ Forte-Enterprise-1	È stato aggiunto il supporto per il dispositivo IonQ Forte-Enterprise-1. Un dispositi vo a 36 quibit che utilizza la tecnologia degli ioni intrappol ati.	17 marzo 2025
S3, condizioni e autorizzazioni migliorate	Per migliorare la sicurezza , `AmazonBraketFullAc cess` ora fornisce solo `s3:*` azioni a. `aws:Princ ipalAccount` Ciò limita l'accesso solo ai bucket del richiedente.	7 marzo 2025
Nuovo dispositivo Rigetti Ankaa-3	È stato aggiunto il supporto per il Rigetti Ankaa-3 dispositi vo. Un dispositivo da 84 quibit che utilizza una tecnologia multi-chip scalabile.	14 gennaio 2025
Rigetti Ankaa-2ritiro del dispositivo	Supporto rimosso per il Rigetti Ankaa-2 dispositivo.	14 gennaio 2025
Support per il IPv6 traffico	Amazon Braket ora supporta il IPv6 traffico che utilizza l'endpoint dualstack. `braket. {region}.api.aws`	12 dicembre 2024

Support per NVIDIA's CUDA-QAmazon Braket	I clienti possono ora eseguire programmi quantistici utilizzando il framework per NVIDIA's CUDA-Q sviluppatori su Amazon Braket.	6 dicembre 2024
IonQ Forte-1il dispositivo è immediatamente disponibile	IonQ Forte-1il dispositivo non è più disponibile solo su prenotazione e ora è immediatamente disponibile per i nostri clienti.	22 novembre 2024
Rigetti Aspen-M-3ritiro del dispositivo	Supporto rimosso per il Rigetti Aspen-M-3 dispositivo.	27 settembre 2024
IonQ Harmonyritiro del dispositivo	Supporto rimosso per il IonQ Harmony dispositivo.	29 agosto 2024
Nuovo dispositivo Rigetti Ankaa-2	È stato aggiunto il supporto per il Rigetti Ankaa-2 dispositivo. Un dispositivo da 84 quibit che utilizza una tecnologia multi-chip scalabile.	26 agosto 2024
Riorganizzazione della guida per gli sviluppatori	La nuova guida per sviluppatori riprende il percorso esistente del cliente Build, Test, Run e guida gli utenti lungo questo percorso con Amazon Braket.	23 agosto 2024
OQC Lucyritiro del dispositivo	Supporto rimosso per il OQC Lucy dispositivo.	28 giugno 2024

Nuovo dispositivo IQM Garnet e nuova regione Europe North 1	È stato aggiunto il supporto per il dispositivo IQM Garnet. Un dispositivo da 20 qubit con topologia a reticolo quadrato. Expanded Braket ha supportato le regioni dell'Europa settentrionale 1 (Stoccolma).	22 maggio 2024
Rilasciata la desintonizzazione locale	Le funzionalità sperimentali ora includono la funzionalità di detoning locale della QuEra QPU Aquila.	11 aprile 2024
Rilasciato Notebook Inactivity Manager	Quando crei un'istanza del notebook, abilita il gestore dell'inattività e imposta un periodo di inattività per ripristinare automaticamente l'istanza del notebook Braket.	27 marzo 2024
Rielaborazione del sommario	Ha riorganizzato il sommario di Amazon Braket per rispettare i requisiti della guida di stile e migliorare AWS il flusso di contenuti per l'esperienza del cliente.	12 dicembre 2023
Braket rilasciato direttamente	È stato aggiunto il supporto per le funzionalità di Braket Direct, tra cui: • Lavorare con le prenotazioni • Ottenere la consulenza di un esperto • Esplora le funzionalità sperimentali	27 novembre 2023

Aggiornato Crea un'istanza di notebook Amazon Braket	È stata aggiornata la documentazione per aggiungere informazioni per creare un'istanza notebook per clienti Amazon Braket nuovi ed esistenti.	27 novembre 2023
Aggiornato Importare un container personalizzato (Bring Your Own Container, BYOC)	È stata aggiornata la documentazione per aggiungere informazioni su quando passare al BYOC, sulla ricetta per passare al BYOC e sull'esecuzione di Braket Hybrid Jobs sul contenitore.	18 ottobre 2023
Rilasciato Hybrid Job Decorator	Esegui il codice locale come lavoro ibridoPagina aggiunta. Contiene esempi: • Crea un lavoro ibrido dal codice Python locale • Installa pacchetti Python e codice sorgente aggiuntivi • Salva e carica i dati in un'istanza di lavoro ibrida • Le migliori pratiche per i decoratori di lavori ibridi	16 ottobre 2023

Aggiunta la visibilità della coda	È stata aggiornata la documentazione della Developer's Guide per includere queue depth equeue position. È stata aggiornata la documentazione dell'API per riflettere le nuove modifiche all'API per la visibilità delle code.	25 settembre 2023
Standardizza la denominazione nella documentazione	È stata aggiornata la documentazione per modificare qualsiasi istanza di «job» in «hybrid job» e «task» in «quantum task»	11 settembre 2023
Nuovo dispositivo IonQ Aria 2	È stato aggiunto il supporto per il IonQ Aria 2 dispositivo	8 settembre 2023
Native Gates aggiornati	È stata aggiornata la documentazione per aggiungere informazioni sull'accesso programmatico ai gate nativi di Rigetti.	16 agosto 2023
Xanadupartenza	È stata aggiornata la documentazione per rimuovere tutti i Xanadu dispositivi	2 giugno 2023
Nuovo dispositivo IonQ Aria	È stato aggiunto il supporto per il IonQ Aria dispositivo	16 maggio 2023
Dispositivo ritirato Rigetti	Supporto interrotto per Rigetti Aspen-M-2	2 maggio 2023

Informazioni aggiornate AmazonBraketFullAccesssulla politica	È stato aggiornato lo script che definisce il contenuto della AmazonBraketFullAccesspolicy per includere le GetMetricData azioni servicequotas: GetServiceQuota e cloudwatch:, nonché informazioni sulle limitazioni rispetto alle quote.	19 aprile 2023
Lancio di Guided Journeys	È stata modificata la documentazione per riflettere il metodo più aggiornato e semplificato per l'onboarding di Braket.	5 aprile 2023
Nuovo dispositivo Rigetti Aspen-M-3	È stato aggiunto il supporto per il Rigetti Aspen-M-3 dispositivo	17 gennaio 2023
Nuova funzione di gradiente aggiuntivo	Sono state aggiunte informazioni sulla funzione di sfumatura aggiuntiva offerta da SV1	7 dicembre 2022
Nuova funzionalità di libreria di algoritmi	Sono state aggiunte informazioni sulla libreria di algoritmi Braket, che fornisce un catalogo di algoritmi quantistici predefiniti	28 novembre 2022
D-Wavepartenza	È stata aggiornata la documentazione per consentire la rimozione di tutti i D-Wave dispositivi	17 novembre 2022
Nuovo dispositivo QuEra Aquila	È stato aggiunto il supporto per il QuEra Aquila dispositivo	31 ottobre 2022

Supporto per Braket Pulse	È stato aggiunto il supporto per Braket Pulse, che consente di utilizzare il controllo degli impulsi su dispositivi Rigetti OQC	20 ottobre 2022
Support per porte native IonQ	È stato aggiunto il supporto per il set di porte nativo offerto dal dispositivo IonQ	13 settembre 2022
Nuove quote di istanze	Sono state aggiornate le quote predefinite delle istanze di calcolo classiche associate a Hybrid Jobs	22 agosto 2022
Nuova dashboard di servizio	Schermate della console aggiornate per includere la dashboard del servizio	17 agosto 2022
Nuovo dispositivo Rigetti Aspen-M-2	È stato aggiunto il supporto per il Rigetti Aspen-M-2 dispositivo	12 agosto 2022
Nuove funzionalità di OpenQASM	Aggiunte le funzionalità di OpenQASM, il supporto per i simulatori locali (braket_sv e braket_dm)	4 agosto 2022
Nuove procedure di tracciamento dei costi	È stato aggiunto come ottenere stime dei costi massimi quasi in tempo reale per simulatori e carichi di lavoro hardware	18 luglio 2022
Nuovo dispositivo Xanadu Borealis	È stato aggiunto il supporto per il Xanadu Borealis dispositivo	2 giugno 2022

Nuove procedure di semplificazione dell'onboarding	Sono state aggiunte informazioni su come funzionano le nuove e semplificate procedure di onboarding	16 maggio 2022
Nuovo dispositivo D-Wave Advantage_system6.1	È stato aggiunto il supporto per il D-Wave Advantage _system6.1 dispositivo	12 maggio 2022
Support per simulatori integrati	È stato aggiunto come eseguire simulazioni integrate con lavori ibridi e come utilizzare il simulatore di fulmini PennyLane	4 maggio 2022
AmazonBraketFullAccess - Politica di accesso completo per Amazon Braket	Aggiunto s3: ListAllMyBuckets autorizzazioni per consentire e agli utenti di visualizzare e ispezionare i bucket creati e utilizzati per Amazon Braket	31 marzo 2022
Support per OpenQASM	Aggiunto il supporto OpenQASM 3.0 per dispositivi e simulatori quantistici basati su gate	7 marzo 2022
Nuovo fornitore di hardware quantistico Oxford Quantum Circuits e nuova regione, eu-west-2	Aggiunto il supporto per OQC e eu-west-2	28 febbraio 2022
Nuovo dispositivo Rigetti	Aggiunto supporto per Rigetti Aspen M-1	15 febbraio 2022
Nuovi limiti di risorse	Aumentato il numero massimo di SV1 attività DM1 e attività simultanee da 55 a 100	5 gennaio 2022

Nuovo dispositivo Rigetti	Aggiunto supporto per Rigetti Aspen-11	20 dicembre 2021
Dispositivo ritirato Rigetti	Supporto interrotto per il dispositivo Rigetti Aspen-10	20 dicembre 2021
Nuovo tipo di risultato	Tipo di risultato a matrice a densità ridotta supportato dal simulatore e DM1 dai dispositivi di matrice a densità locale	20 dicembre 2021
Descrizione aggiornata della politica	Amazon Braket ha aggiornato l'ARN del ruolo per includere il percorso. `servicerole/` Per informazioni sugli aggiornamenti delle policy, consulta la tabella degli aggiornamenti di Amazon Braket alle politiche AWS gestite.	29 novembre 2021
Offerte di lavoro Amazon Braket	Guida utente per Amazon Braket Hybrid Jobs e aggiunte API	29 novembre 2021
Nuovo dispositivo Rigetti	Aggiunto supporto per Rigetti Aspen-10	20 novembre 2021
Dispositivo ritirato D-Wave	Supporto interrotto per D-Wave QPU, Advantage _system1	4 novembre 2021
Nuovo dispositivo D-Wave	È stato aggiunto il supporto per una D-Wave QPU aggiuntiva, Advantage _system4	5 ottobre 2021

Nuovi simulatori di rumore	È stato aggiunto il supporto per un simulatore di matrice di densità (DM1), che può simulare circuiti fino a 17 qubits e un simulatore di rumore locale braket_dm	25 maggio 2021
PennyLane supporto	È stato aggiunto il supporto per PennyLane Amazon Braket	8 dicembre 2020
Nuovo simulatore	Aggiunto il supporto per un Tensor Network Simulator (TN1), che consente circuiti più grandi	8 dicembre 2020
Raggruppamento delle attività	Braket supporta il raggruppamento delle attività dei clienti	24 novembre 2020
Allocazione manuale qubit	Braket supporta l'qubital locazione manuale sul dispositivo Rigetti	24 novembre 2020
Quote modificabili	Braket supporta quote regolabili in modalità self-service per le risorse operative	30 ottobre 2020
Support per PrivateLink	Puoi configurare endpoint VPC privati per i tuoi lavori Braket	30 ottobre 2020
Supporto per i tag	Braket supporta tag API basati per la risorsa quantum-task	30 ottobre 2020
Nuovo dispositivo D-Wave	È stato aggiunto il supporto per una D-Wave QPU aggiuntiva, Advantage _system1	29 settembre 2020

Rilascio iniziale	Versione iniziale della documentazione di Amazon Braket	12 agosto 2020

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.