AWS Well-Architected Framework

High Performance Computing Lens

High Performance Computing Lens: AWS Well-Architected Framework

Table of Contents

High Performance Computing Lens - AWS Well-Architected Framework

Publication date: **June 23, 2025** (*Document revisions*)

The AWS Well-Architected High Performance Computing (HPC) Lens describes user scenarios, questions related to the AWS Well-Architected pillars, best practices, and prescriptive guidance for customers building their workloads. This lens is defined within the AWS Well-Architected Framework, covers common HPC customer scenarios, and identifies key architectural elements that help you build your workloads according to AWS best practices.

Introduction

This document describes the AWS Well-Architected High-Performance Computing (HPC) Lens, a collection of customer-proven scenarios and best practices for designing well-architected HPC workloads. The HPC Lens contains insights that AWS has gathered from real-world case studies and helps you learn the key design elements of well-architected compute workloads along with recommendations for improvement. The document is intended for IT architects, developers, and team members who build and operate HPC systems.

You can use this lens to learn and implement architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. The Well-Architected Framework provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. We believe that having well-architected systems greatly increases the likelihood of business success.

In this lens, we focus in on how to design, deploy, and architect your HPC workloads in the AWS Cloud. HPC workloads run exceptionally well in the cloud. The spiky nature of HPC workloads make them well suited for pay-as-you-go and elastic cloud infrastructure. In addition, the ability to fine-tune cloud resources and create cloud architectures provide greater flexibility than static, on-premises environments.

For brevity, we only cover details from the AWS Well-Architected Framework that are specific to HPC workloads. When designing your architecture, we recommend that you consider best practices and strategies from the AWS Well- Architected Framework.

This paper is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, and operations team members. After reading this paper, you will

understand AWS best practices and strategies to use when designing and operating HPC in a cloud environment.

Lens availability

The HPC Lens is available as an AWS-official lens in the Lens Catalog of the AWS Well-Architected Tool.

To get started, follow the steps in Adding a lens to a workload and select the **HPC Lens**.

Design principles

Consider the following design principles when building an HPC workload in the cloud:

- **Use dynamic architectures**: Avoid defaulting to static architectures and cost estimates that use a steady-state model. Your architecture can be dynamic, growing and shrinking to match your HPC demands over time. Match your architecture design and cost management explicitly to the natural cycles of HPC activity.
- **Align the procurement model to the workload**: AWS makes a range of compute procurement models, such as on-demand and savings plans, available for different HPC demand patterns. By selecting the correct model, you only pay for what you need. You can potentially combine models for a committed rate while having burst capacity with on-demand compute.
- **Consider your data**: Understand your data before you begin designing your architecture. Consider your data's location, size, update, and regulatory requirements. A holistic optimization of performance and cost focuses on compute and includes data considerations. Your data requirements may shape optimal performance based on available resources in your selected Regions.
- **Automate with infrastructure as code:** Consider products, such as AWS ParallelCluster and AWS Research and Engineering Studio, to ease your effort in automating your infrastructure. Alternatively, you can customize your infrastructure with services, such as AWS CloudFormation, if needed but plan for a higher development effort.
- **Collaborate securely**: HPC work often occurs in a collaborative context and is usually part of a larger workflow. Take full advantage of the security and collaboration features that make AWS an excellent environment for you and your collaborators to solve your HPC problems. For example, you can grant cross-account access to share a full or partial data set with Amazon S3 to another AWS account. This helps your computing solutions and datasets achieve a greater impact by securely sharing within a selective group or publicly sharing with the broader community. When collaborating, consider data locality and workflow architecture. For example, use remote visualization with Amazon DCV rather than transferring data to users.
- **Use designs built for the cloud**: Directly replicating your on-premises architecture is usually unnecessary and suboptimal when migrating workloads to AWS. Consider the breadth and depth of AWS services to create new design patterns with solutions that are built for cloud architectures. For example, in the cloud, each user or group can use a separate cluster, which can independently scale depending on the load.

- **Test real-world workloads**: You only pay for what you actually use with AWS, which makes it possible to create a realistic proof-of-concept with your own representative models. Most HPC applications are complex, and their memory, CPU, and network patterns often can't be reduced to simplified microbenchmarks or theoretical performance benchmarks. With AWS, you can quickly get started, iterate, and optimize your design with only paying for consumed resources as you finalize your architecture.
- **Balance your desired price and time-to-results**: Use time and cost to analyze performance. Sometimes you can prioritize reducing cost when you do not immediately require results. For time-critical workloads, consider trading cost optimization for faster time-to-results. Determine your desired price-for-performance approach based on your results timeline and available budget.

Definitions

The AWS Well-Architected Framework is based on six pillars: operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability. When architecting solutions, you make tradeoffs between pillars based upon your business context. These business decisions can drive your engineering priorities. You might reduce cost at the expense of reliability in development environments, or for mission-critical solutions, you might improve reliability with increased costs. Security and operational excellence are generally not traded off against other pillars.

- **HPC workloads**: HPC users commonly speak of jobs. AWS refers to these as workloads.
- **Loosely coupled**: Loosely coupled workloads typically see a large number of small jobs working independently of one another. This is sometimes referred to as high-throughput computing (HTC).
- **Tightly coupled**: Tightly coupled workloads, such as Computational Fluid Dynamics on AWS and weather forecasting, have multi-threaded or multi-core processes working together. For additional information on these use cases, see the Tightly Coupled Scenarios.
- **Workload scheduling:** Job scheduling is a means of queuing multiple jobs up to be processed (either sequentially or concurrently depending on compute resources available). Several industry-standard HPC job schedulers are available with AWS services, such as Slurm with AWS ParallelCluster, as well as AWS Partner Independent Software Vendors (ISVs) offerings. Job scheduling can also be defined using serverless algorithm design.
- **Compute vCPUs:** HPC users may refer to a server as a node while AWS refers to a virtual server as an Amazon EC2 instance. AWS names and sizes instances based on the available resources allocated to the instances, such as the number of vCPUs. Depending on the instance type, a vCPU could be a thread through simultaneous multithreading (SMT) or a physical CPU core. For instance specifics, see Amazon EC2 instance type specifications.
- **Cluster placement group:** A cluster placement group is a logical grouping of your compute instances within a single Availability Zone (AZ) within the same high-bisection bandwidth segment of the network. Cluster placement groups provide low latency and high bandwidth between your compute instances.
- **Elastic Fabric Adapter:** Tightly coupled workloads on AWS often benefit from the use of the Elastic Fabric Adapter (EFA) and cluster placement groups. EFA is a networking technology that provides lower and more consistent latency and higher throughput than the TCP transport

traditionally used in cloud-based HPC systems. EFA is described in more detail in the Networking section.

- **Amazon Machine Image (AMI):** An Amazon Machine Images in Amazon EC2 is an image that contains the software for an EC2 instance to launch. These are images supported and maintained by AWS, as well as Marketplace offerings and the ability for users to create their own.
- **Blue/green instances:** A blue/green deployment is a strategy whereby two separate, but identical, deployments are created. The blue environment runs the current application version while the green environment runs the new application version.
- **CI/CD:** Continuous Integration and Continuous Development. CI is a set of SW coding practices that drive development teams to frequently implement small code changes and check them in to a version control repository. It establishes an automated way to build, package, and test their applications. This encourages developers to commit code changes more frequently, leading to better collaboration and code quality. CD automates application delivery to production, development, and testing environments and is an automated way to push code changes to these environments.
- **Computational Fluid Dynamics (CFD):** This is a form of Computational Aided Engineering (CAE) to simulate fluid flows around or within objects. It uses a Finite Volume approach to solve numerical equations across domains split into many smaller cells in an iterative approach. A commonly used example is aerodynamics of cars.
- **Data retention requirement:** Data management – where and how much data is retained – is a significant consideration for HPC environments. User and regulatory requirements for data retention should be considered, with different strategies possible.
- **EasyBuild:** Third party package management software that provides a software build and install framework to manage (chiefly scientific) software on High Performance Computing (HPC) systems efficiently.
- **Fail fast:** This is a philosophy for frequent, incremental, development and testing to determine whether a development path is the best one.
- **Finite Element Analysis (FEA):** A form of Computational Aided Engineering (CAE) that simulates structural analysis. This is used for many engineered products such as crash simulation in cars or building reaction to earthquakes. This simulation type is heavily used in Mechanical Engineering.
- **Genomics:** A branch of study that refers to mapping the genetic code of living tissues to identify the genetic material and the result of interactions of genes with each other and their environment.

- **Hypervisor:** Software that allows for the running of multiple virtual machines on a single physical machine. AWS use the AWS Nitro System.
- **IOPS:** Input/output operations per second. This is a measure of data throughput on a bus, in a CPU or GPU, or when using a storage device.
- **Job scheduling software:** This is software that knows what resources are available and allows users to submit work to run on the compute machines. If all resources are busy, jobs are queued up until the required resources are available.
- **Lazy load:** When data in an S3 bucket linked to a Data Repository Association (for FSx for Lustre) is accessed, Amazon File Cache automatically loads the metadata and file contents if they are not already present in the cache. Lazy load allows reading or writing data or metadata to files in a DRA directory.
- **Luigi:** A lightweight, open-sourced, Python-based workflow scheduler that is typically used for orchestrating data assets in the Hadoop ecosystem.
- **Memory to Core Ratio:** A compute machine has a physical amount of Random Access Memory (RAM) available as well as number of cores. By dividing RAM by core count it is possible to determine the amount of RAM per core available. If the machine has a job running on it that divides the compute resources equally, the workload that is going to run will be limited on a per-core basis by the amount of memory available for that core. This can place an upper limit on workload sizes on a particular machine configuration.
- **Monte Carlo simulations:** The Monte Carlo simulation method is a means of modelling approximate solutions using repeated random sampling.
- **Message Passing Interface (MPI):**Message Passing Interface is a standard that defines communication between machines that are working on workloads together, allowing those workloads to scale out to more than one instance while solving the same problem.
- **Network access control lists (NACLs):** A Control subnet traffic with network access control lists allows administrators to control who has access into a network. This controls traffic at the subnet level rather than at an instance.
- **Parallel file system:** This is a file system that allows multiple machines to access the same file system. It is also possible for multiple reads/writes to be made synchronously by a single instance to speed up file input/output.
- **Personally identifiable information (PII):** Information that identifies one or more details about a person including, but not limited to, name, address, gender, birth-date, SSN, and Government identifier and has strict compliance restrictions on organizations that use it.

- **Placement group:** In a physical, on-premises, High Performance Compute (HPC) cluster, machines would generally be physically in very close proximity to each other. This reduces network latency as the transmission distance between machines is low while also needing few numbers of 'hops' across different hardware (e.g. network switches). The equivalent on AWS is requesting instances in a Placement Group whereby requested instances will be allocated as close to each other as possible.
- **Recovery Point Objective (RPO):** According to the US National Institute of Standards and Technology, Recovery Point Objective is defined as The point in time to which data must be recovered after an outage.
- **Recovery Time Objective (RTO):** According to the US National Institute of Standards and Technology, Recovery Time Objective is defined as The overall length of time an information system's components can be in the recovery phase before negatively impacting the organization's mission or mission/business processes.
- **REST-based communication:** REST is defined by the US NIST as a software architectural style that defines a common method for defining APIs for Web services.
- **Role-based access control (RBAC):** the US NIST provides a definition as 'Access control based on user roles (that is a collection of access authorizations that a user receives based on an explicit or implicit assumption of a given role). Role permissions may be inherited through a role hierarchy and typically reflect the permissions needed to perform defined functions within an organization. A given role may apply to a single individual or to several individuals.'
- **SBGrid:** Third party package management software that provides a software build and install framework to manage software on High Performance Computing (HPC) systems efficiently.
- **Slurm:** This is an open-source workload manager (queuer) to schedule jobs for Linux or Unix based systems. Now also known as the Slurm Workload Manager, previously Simple Linux Utility for Resource Management (Slurm). This is the default choice for scheduler for AWS ParallelCluster.
- **Slurm partition:** Computational resources on a cluster with a queue hosted by a Slurm system can be split across different partitions, allowing users to choose what hardware configuration to queue their jobs on to.
- **Spack:** Third party package management software that provides a software build and install framework to manage (chiefly scientific) software on High Performance Computing (HPC) systems efficiently.

Scenarios

HPC cases are typically complex computational problems that require parallel-processing techniques. To support the calculations, a well-architected HPC infrastructure is capable of sustained performance for the duration of the calculations. HPC workloads span traditional applications, like computational chemistry, financial risk modeling, computer aided engineering, weather prediction, and seismic imaging, as well as emerging applications, like artificial intelligence, autonomous driving, and bioinformatics.

The traditional grids or HPC clusters that support these calculations are similar in architecture with particular cluster attributes optimized for the specific workload. In a traditional on-premises HPC cluster, every workload has to be optimized for the given infrastructure. In AWS, the network, storage type, compute (instance) type, and deployment method can be chosen to optimize performance, cost, robustness and usability for a particular workload.

In this section we will introduce the following scenarios commonly seen with HPC workloads:

- Loosely coupled cases
- Tightly coupled cases
- Hybrid HPC cases

We also provide a reference architecture for each workload type. Architecture may differ based on a workload's data access pattern, so for each of the scenarios, there are considerations for data-light and data-intensive workloads. The reference architectures provided here are representative examples and do not exclude the possible selection of other AWS services or third-party solutions.

Loosely coupled cases are those where the multiple or parallel processes do not strongly interact with each other in the course of the entire simulation (often it is based on data parallelism). With loosely coupled workloads, the completion of an entire calculation or simulation often requires hundreds to millions of parallel processes. These processes occur in any order and at any speed through the course of the simulation. This offers flexibility on the computing infrastructure required for loosely coupled simulations.

Tightly coupled cases are those where the parallel processes are simultaneously running and regularly exchanging information between each other at each iteration or step of the simulation. Typically, these tightly coupled simulations run on a homogenous cluster using MPI. The total core

or processor count can range from tens to thousands and occasionally to hundreds of thousands if the infrastructure allows. The interactions of the processes during the simulation place extra demands on the infrastructure, such as the compute nodes and network infrastructure.

In a hybrid HPC case, an on-premises HPC infrastructure interacts with HPC resources on AWS to extend on-premises resources and functionalities. Hybrid scenarios vary from minimal coordination, like compute separation, to tightly integrated approaches, like scheduler driven job placement. On-premises infrastructure is normally connected with AWS through a secure VPN tunnel or a dedicated network. Typically, some or all of the data is synced in between the on-premises environment and AWS through this network.

The infrastructure used to run the huge variety of loosely coupled and tightly coupled workloads is differentiated by its ability for process interactions across nodes and storage infrastructure. There are fundamental aspects that apply to loosely and tightly coupled scenarios, as well as specific design considerations. There are additional aspects for hybrid scenarios to be considered. Consider the following fundamentals for all scenarios when selecting an HPC infrastructure on AWS:

- **Network:** Network requirements can range from cases with low demands, such as loosely coupled applications with minimal communication traffic, to tightly coupled and massively parallel applications that require a performant network with high bandwidth and low latency. For hybrid scenarios, a secure link between the on-premises data center and AWS may be required, such as AWS Direct Connect or Site-to-Site VPN.
- **Storage:** HPC calculations use, create, and move data in unique ways. Storage infrastructure must support these requirements during each step of the calculation. Factors to be considered include data size, media type, transfer speeds, shared access, and storage properties (for example, durability and availability). AWS offers a wide range of storage options to meet the performance requirements of both data-light and data-intensive workloads.
- **Compute:** The Amazon EC2 instance type defines the hardware capabilities available for your HPC workload. Hardware capabilities include the processor type, core frequency, processor features (for example, vector extensions), memory-to-core ratio, and network performance. In the cloud, you can also use HPC resources interactively through AWS-managed tools such as Amazon DCV desktops, open source tools such as Jupyter, or other third-party options for domain specific applications.
- **Deployment:** AWS provides many options for deploying HPC workloads. For an automated deployment, a variety of software development kits (SDKs) are available for coding end-to-end solutions in different programming languages. A popular HPC deployment option combines bash shell scripting with the AWS Command Line Interface (AWS CLI). There are also Infrastructure as

code (IaC) options such as AWS CloudFormation and AWS ParallelCluster, as well as managed deployment services for container-based workloads such as Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), AWS Fargate, and AWS Batch.

In the following sections, we review a few example scenarios and architectures to demonstrate how AWS can address requirements for the wide range of HPC use cases.

Scenarios

- Loosely coupled scenarios
- Tightly coupled scenarios
- Hybrid scenarios

Loosely coupled scenarios

A loosely coupled workload entails the processing of a large number of smaller tasks. Generally, the smaller task runs on one node, either consuming one process or multiple processes with shared memory parallelization (SMP) for parallelization within that node. The parallel processes, or the iterations in the simulation, are post-processed to create one solution or discovery from the simulation. The loss of one node or job in a loosely coupled workload usually doesn't delay the entire calculation. The lost work can be picked up later or omitted altogether. The nodes involved in the calculation can vary in specification and power. Loosely coupled applications are found in many disciplines and range from data-light workloads, such as Monte Carlo simulations, to data-intensive workloads, such as image processing and genomics analysis. Data-light workloads are generally more flexible in architectures, while data-intensive workloads are generally more impacted by storage performance, data locality with compute resources, and data transfer costs. A suitable architecture for a loosely coupled workload has the following considerations:

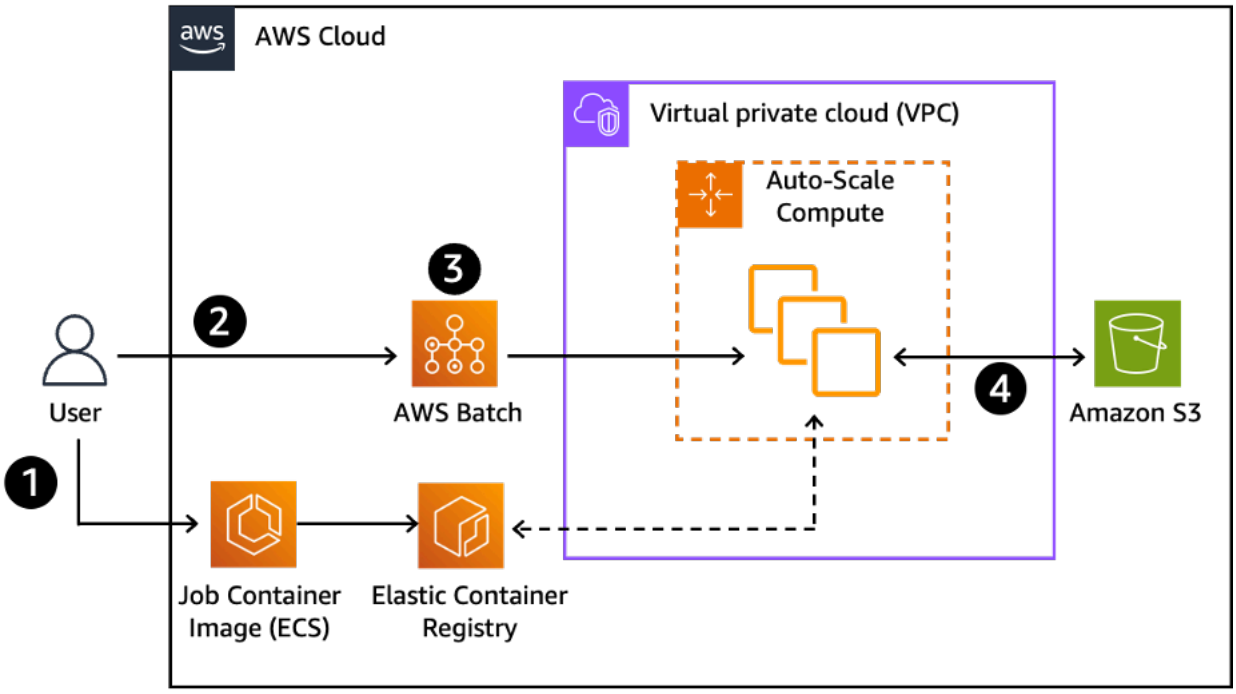
- **Network:** Because parallel processes do not typically interact with each other, the feasibility or performance of the workloads is typically not sensitive to the bandwidth and latency capabilities of the network between instances. Therefore, cluster placement groups are not necessary for this scenario because they weaken resiliency without providing a performance gain. However, data-intensive workloads are potentially more sensitive to network bandwidth to your storage solution when compared to data-light workloads.
- **Storage:** Loosely coupled workloads vary in storage requirements and are driven by the dataset size and desired performance for transferring, reading, and writing the data. Some workloads may require a local disk for low latency access to data, in which case customers may choose an

EC2 instance with instance storage. Some workloads may require a shared file system that can be mounted on different EC2 instances, which would be a better fit with an NFS file share, such as Amazon Elastic File System (EFS), or a parallel file system, such as Amazon FSx for Lustre.

- **Compute:** Each application is different, but in general, the application's memory-to-compute ratio drives the underlying EC2 instance type. Some applications are optimized to take advantage of AI accelerators such as AWS Trainium and AWS Inferentia, graphics processing units (GPUs), or field-programmable gate array (FPGA) on EC2 instances.
- **Deployment:** Loosely coupled simulations can be run across many (sometimes millions) of compute cores that can be spread across Availability Zones without sacrificing performance. They consist of many individual tasks, so workflow management is key. Deployment can be performed through end-to-end services and solutions such as AWS Batch and AWS ParallelCluster, or through a combination of AWS services, such as Amazon Simple Queue Service (Amazon SQS), AWS Auto Scaling, Serverless Computing - AWS Lambda, and AWS Step Functions. Loosely coupled jobs can also be orchestrated by commercial and open-source workflow engines.

Reference architecture: AWS Batch

AWS Batch is a fully managed service that helps you run large-scale compute workloads in the cloud without provisioning resources or managing schedulers. AWS Batch enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS. AWS Batch dynamically provisions the optimal quantity and type of compute resources (for example, CPU or memory-optimized instances) based on the volume and specified resource requirements of the batch jobs submitted. It plans, schedules, and runs containerized batch computing workloads across the full range of AWS compute services and features, such as Amazon EC2 and AWS Fargate. Without the need to install and manage the batch computing software or server clusters necessary for running your jobs, you can focus on analyzing results and gaining new insights. With AWS Batch, you package your application in a container, specify your job's dependencies, and submit your batch jobs using the AWS Management Console, the CLI, or an SDK. You can specify runtime parameters and job dependencies and integrate with a broad range of popular batch computing workflow engines and languages (for example, Pegasus WMS, Luigi, and AWS Step Functions). AWS Batch provides default job queues and compute environment definitions that enable you to get started quickly.



AWS Batch reference architecture

Workflow steps

1. User creates a container containing applications and their dependencies, uploads the container to the Amazon Elastic Container Registry or another container registry (for example, DockerHub), and creates a job definition to AWS Batch.
2. User submits jobs to a job queue in AWS Batch.
3. AWS Batch pulls the image from the container registry and processes the jobs in the queue
4. Input and output data from each job is stored in an S3 bucket.

AWS Batch can be used for data-light and data-intensive workloads. It also can be deployed in a single Availability Zone or across multiple Availability Zones for additional compute capacity or architecture resiliency. When using any multi-AZ architecture, consider the service and location for data storage to manage performance and data-transfer costs, especially for data-intensive workloads.

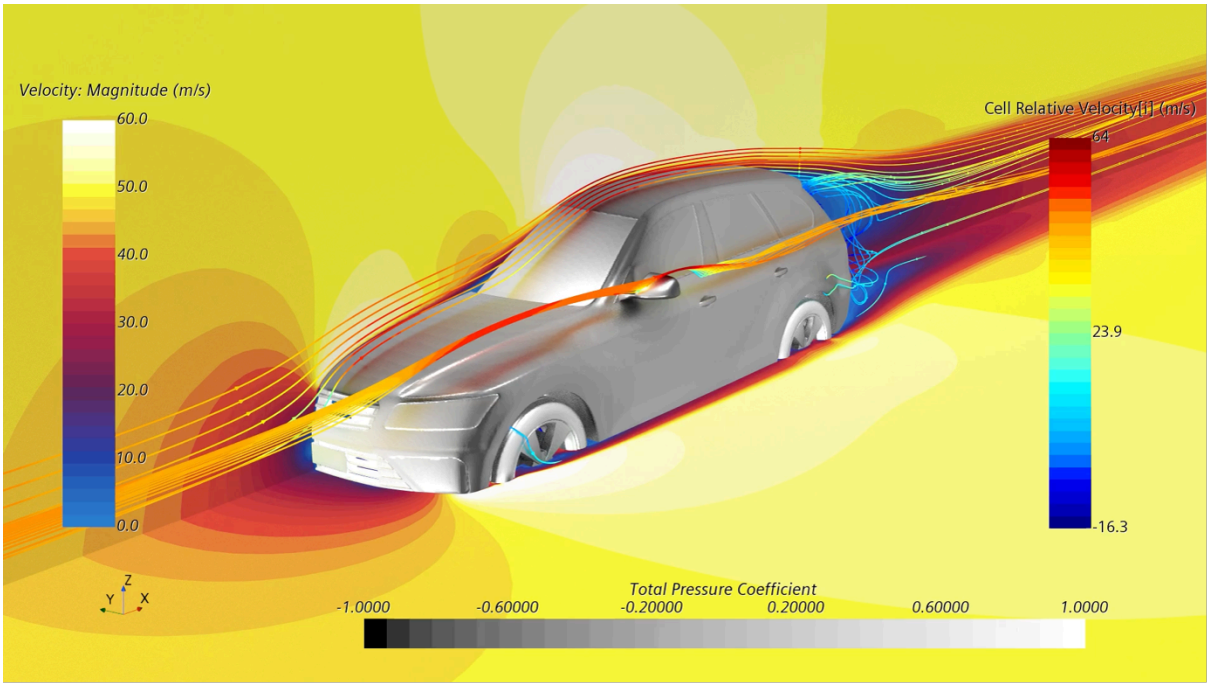
Tightly coupled scenarios

Tightly coupled applications consist of parallel processes that are dependent on each other to carry out the calculation. These applications can vary in size and total run time, but the main

common theme is the requirement for all processes to complete their tasks. Unlike a loosely coupled computation, all processes of a tightly coupled simulation iterate together and require communication with one another.

An *iteration* is defined as one step of the overall simulation. Tightly coupled calculations rely on tens to thousands of processes or cores over one to many iterations. The failure of one node usually leads to the failure of the entire calculation. To mitigate the risk of complete failure, application-level checkpointing can be used. This is a feature of some software that allows checkpointing (saving) regularly during computation to allow for the restarting of a simulation from a known state.

Tightly coupled simulations typically rely on a Message Passing Interface (MPI) for inter-process communication. Multi-threading and shared memory parallelism through OpenMP can be used with MPI. Examples of tightly coupled HPC workloads include computational fluid dynamics (CFD), finite element analysis (FEA), weather prediction, and reservoir simulation.



An example of a tightly coupled workload; a high cell count Computational Fluid Dynamics simulation

A suitable architecture attempts to minimize simulation runtimes. A tightly coupled HPC workload has the following key considerations:

- **Network**: The network requirements for tightly coupled calculations are demanding. Slow communication between nodes results in the slowdown of the entire calculation. Cluster

placement groups and high-speed networking cards, such as the Elastic Fabric Adapter (EFA), help to achieve this in the cloud. Larger workloads run on HPC systems are more reliant on core or memory speed and can scale well across multiple instances. Smaller workloads with a lower total computational requirement find networking can be the bottleneck (due to the amount of communication between instances required) and can have the greatest demand on the network infrastructure. EFA enables running applications that require high levels of internode communications at scale on AWS.

- **Storage**: Tightly coupled workloads vary in storage requirements and are driven by the dataset size and desired performance for transferring, reading, and writing the data. Some applications make use of frequent I/O calls to disk, while others only read and write on initial load and final save of files. Particular instances on Amazon EC2 have local storage (often NVMe) which suit being used as fast scratch storage well. In other cases, a shared file system, such as Amazon FSx for Lustre, would provide the throughput required for HPC applications.
- **Compute**: EC2 instances are offered in a variety of configurations with varying core to memory ratios. For parallel applications, it is helpful to spread memory-intensive parallel simulations across more compute nodes to lessen the memory-per-core requirements and to target the best performing instance type. Tightly coupled applications require queues with homogenous compute nodes, though it is possible to assign different instance types to different queues. Targeting the largest instance size minimizes internode network latency while providing the maximum network performance when communicating between nodes. Some software benefits from particular compute features and defining a queue with instances that benefit one code over another can provide more flexibility and optimization for the user.
- **Deployment**: A variety of deployment options are available. End-to-end automation is achievable, as is launching simulations in a traditional cluster environment. Cloud scalability means that you can launch hundreds of large multi-process cases at once, so there is no need to wait in a queue. Tightly coupled simulations can be deployed with end-to-end solutions such as AWS ParallelCluster and AWS Batch, or through solutions based on AWS services such as AWS CloudFormation or EC2 Fleet.

Reference architecture: AWS ParallelCluster

A data light workload for tightly coupled compute scenarios may be one where a lot of the file-based input and output happens at the start and end of a compute job, with relatively small data sets used. An example workload of this is computational fluid dynamics, where a simulation could be created from a light-weight geometry model. Computational fluid dynamics involves the computation of numerical methods to solve problems such as aerodynamics. One method

of solving these problems requires the computational domain to be broken into small cells with equations, then solved iteratively for each cell. Even the simplest geometric model can require millions of mathematical equations to be solved. Due to the splitting of the domain into smaller cells (which in turn can be grouped across different processors), these simulations can scale well across many compute cores.

Data intensive workloads for tightly coupled workloads have large data sets in frequent I/O operations. Two example data intensive workloads in a tightly coupled scenario are finite element analysis (FEA) and computational fluid dynamics (CFD). Both workload types require an initial read and final write of data which can be large in size. However, FEA also requires constant I/O during compute time. For these data intensive workloads, it is important the path between the data and the processor is as fast as possible while also acquiring the best throughput possible from the storage device. Generally, if the computational code supports parallel input and output, the HPC cluster benefits from a very high throughput shared file system, such as one based on Amazon FSx for Lustre.

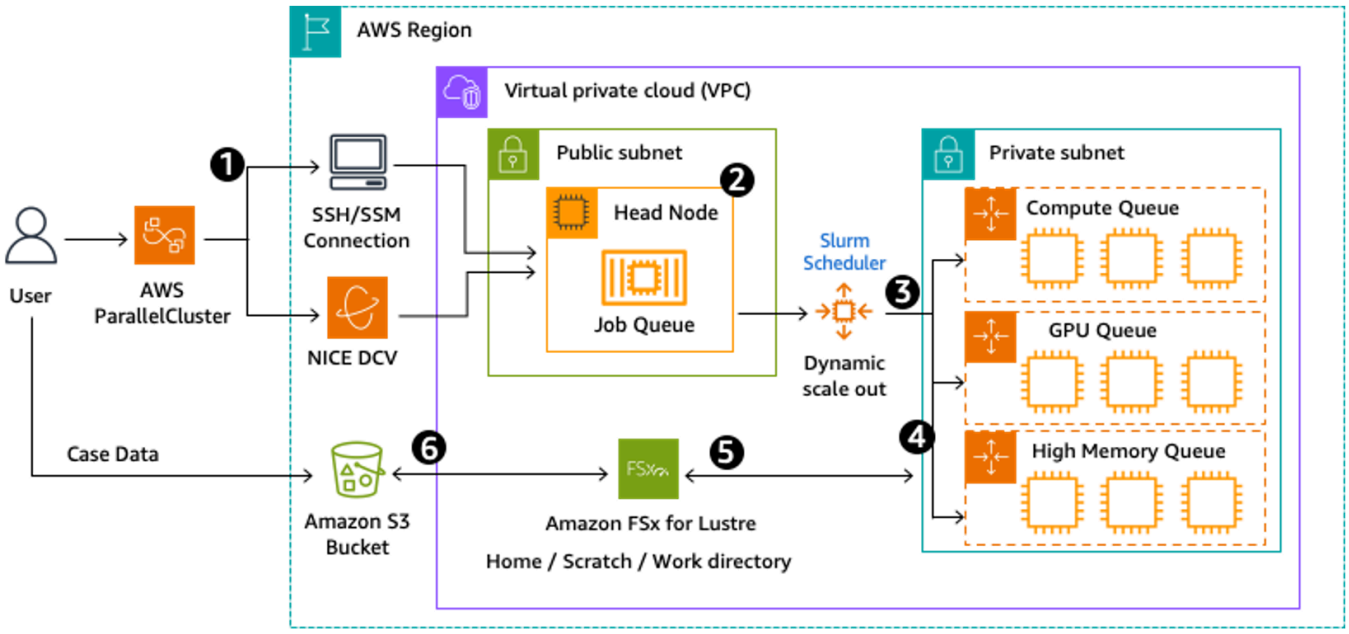
For workloads that require many read and write operations during solve time, such as FEA, a common approach is to allow the user to specify a scratch drive to use for data transfer during the simulation run time. Instance store volumes based on NVMe help here because temporary read and write operations can be kept local to the compute processors reducing any network or file system bottlenecks during simulation solve time and removing the I/O bottleneck. In this case, running these solvers on an AWS Instance with Instance Store Volumes, such as hpc6id.32xlarge, provides the best possible throughput for in-job I/O required.

A reference architecture suitable for this workload is shown in the following AWS ParallelCluster Architecture diagram. This diagram shows Amazon FSx for Lustre in use for the shared file storage system. In many cases, Amazon EFS would be suitable for a data light workload, though when running a large number of simultaneous jobs Amazon FSx for Lustre may be more suited to avoid potential race conditions occurring during software load.

AWS ParallelCluster is an AWS supported open-source cluster management tool that makes it easy to deploy and manage an HPC cluster with a scheduler such as Slurm. It deploys cluster through configuration files so uses infrastructure as code (IaC) to automate and version cluster provisioning. There are multiple ways ParallelCluster can be used to create a cluster, like through an API, the CLI, a GUI, or an AWS CloudFormation template.

AWS ParallelCluster can be deployed with a shared file system (for example, Amazon Elastic File System or Amazon FSx for Lustre) that a head node, as well as any compute nodes, all have access to. With no running jobs on the cluster, Amazon EC2 compute instances remain in terminated state,

and users are not billed for compute instance resource consumption. When a job is submitted to the cluster, EC2 instances are started using a specified Amazon Machine Image (AMI), and once provisioned and fully running, become available for the compute queue the job was submitted to. These machines remain available and used by the scheduler until they are idle for a cooldown period of time (10 minutes by default), at which point the EC2 instances will then be shut down and released back to AWS.



Reference architecture: AWS ParallelCluster

Workflow steps

1. User connects to cluster through a terminal session using either SSH connections or SSM through the AWS console. Alternatively, an Amazon DCV server may be running on an instance that allows connection to the head node through a graphical user interface.
2. User prepares jobs using the head node and submits jobs to a job queue using the scheduler, such as Slurm.
3. AWS ParallelCluster starts compute instances based on the requested resources. These instances are started and assigned to the AWS account before being available within the cluster's compute queue.
4. Jobs are processed through the scheduler queue with compute instance types and number determined by the user.
5. Input and output data from each job is stored on a shared storage device, such as FSx for Lustre.

6. Data can be exported to Amazon S3 (and archived in Amazon Glacier), with S3 also being used as a means of loading data onto the cluster from on-premises machines.

AWS ParallelCluster can be used for data-light and data-intensive workloads. It also can be deployed in a single Availability Zone or across multiple Availability Zones within an AWS Region for scalability with additional compute capacity. When using any multi-AZ architecture, consider the service and location of the storage for lower latency and data-transfer costs, especially for data-intensive workloads.

Hybrid scenarios

Hybrid deployments are primarily considered by organizations that are invested in their on-premises infrastructure and also want to use AWS. This approach allows organizations to augment on-premises resources and creates an alternative path to AWS rather than an immediate full migration. Hybrid-deployment architectures can be used for loosely and tightly coupled workloads.

Hybrid scenarios vary from minimal coordination, like workload separation, to tightly integrated approaches, like scheduler-driven job placement. Motivations to drive the hybrid approach could be one of the following:

- **To meet specific workload requirements:** An organization may separate their workloads and run all workloads of a certain type on AWS infrastructure. For example, an organization may choose to run research and development workloads in their on-premises environment, but may choose to run their production workloads on AWS for higher resiliency and elasticity.
- **To extend HPC resources:** Organizations with a large investment in their on-premises infrastructure typically have a large number of users that compete for resources during peak hours. During these times, they require more resources to run their workloads.
- **To extend technical functionality:** In many cases, on-premises HPC infrastructure runs on a unified CPU architecture. Some may have accelerators such as GPUs for specialized use cases. Customers may avoid lengthy hardware procurement processes and run some workloads on AWS to experiment with hardware that is not already available in their on-premises environment. AWS provides a variety of choices in architecture on Amazon EC2 in CPU and accelerators. There are also other services such as Amazon Braket (a fully managed service for quantum computing), Machine Learning Service - Amazon SageMaker AI AI (a fully-managed service to build, train, and deploy machine learning models), and many more that can be used in conjunction with on-premises resources.

- **To enhance commercial capability:** Some organizations have policy restrictions that dictate that their users cannot publish or commercialize their in-house developed HPC solutions to the general public in their on-premises environment, and may choose to do so on AWS. For example, users of a government-owned HPC facility may develop applications and solutions through their scientific research. To share their work within the larger community, they can host their solution on AWS and provide it as a solution on AWS Marketplace.

Data locality and data movement are critical factors in successfully operating a hybrid scenario. A suitable architecture for a hybrid workload has the following considerations:

- **Network:** In a hybrid scenario, the network architecture in between the on-premises environment and AWS should be considered carefully. To establish a secure connection, an organization may use AWS Site-to-Site VPN to move their data light workloads between their own data center and AWS. For data intensive workloads, a dedicated network connection in between an on-premises environment and AWS with AWS Direct Connect may be a better choice for sustainable network performance. For further low latency requirements or to meet stringent data residency requirements, AWS Local Zones or AWS Dedicated Local Zones may be an option.
- **Storage:** Techniques to address data management vary depending on organization. For example, one organization may have their users manage the data transfer in their job submission scripts, while others might only run certain jobs in the location where a dataset resides, and another organization might choose to use a combination of several options. Depending on the data management approach, AWS provides several services to aid in a hybrid deployment.

  For example, Amazon File Cache can link an on-premises file system to a cache on AWS, AWS Storage Gateway File Gateway can expose an Amazon S3 bucket to on-premises resources over NFS or SMB, and AWS DataSync automatically moves data from on-premises storage to Amazon S3 or Amazon Elastic File System. Additional software options are available from third-party companies in the AWS Marketplace and the AWS Partner Network (APN).
- **Compute:** A single loosely coupled workload may span across an on-premises and AWS environment without sacrificing performance, depending on the data locality and amount of data being processed. Tightly coupled workloads are sensitive to network latency, so a single tightly coupled workload should reside either on premises or in AWS for best performance.
- **Deployment:** Many job schedulers support bursting capabilities onto AWS, so an organization may choose to use their existing on-premises job scheduler to handle deployment of compute resources onto the cloud. There are also third-party HPC portals or meta schedulers that can be integrated with clusters either on premises or in the cloud. Depending on its hybrid strategy,

an organization may choose to separate their workloads by different teams, in which case they may consider using AWS tools and services such as AWS ParallelCluster and AWS Batch with file sharing capabilities through hybrid storage offerings such as Amazon File Cache, AWS DataSync or AWS Storage Gateway.

Reference architecture: Hybrid deployment

Data-light workloads in a hybrid HPC architecture are typically workloads with a small dataset that does not require many I/O operations during runtime. For data-light workloads where data movement between on-premises infrastructure and the AWS Cloud is not a significant factor, consider setting up a cache functionality on AWS. Amazon File Cache can be used to create a cache for workloads on AWS and can be mounted directly on Amazon EC2 instances. Amazon File Cache is an AWS service that can be associated with Amazon S3 or NFS as a data repository.

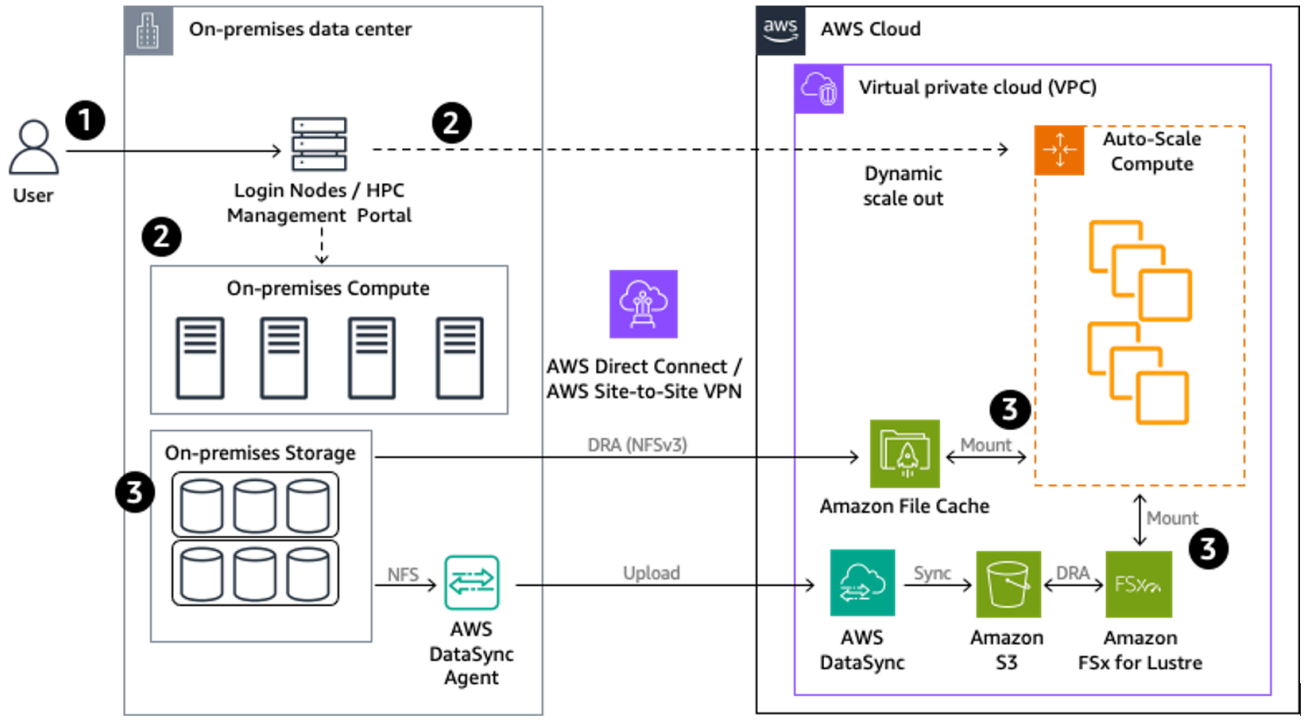
Many HPC facilities have a parallel file system in their on-premises infrastructure, which can be exported over NFS protocol. When data is accessed on a linked NFS data repository using the cache, Amazon File Cache automatically loads the metadata (the name, ownership, timestamps, and permissions) and file contents if they are not already present in the cache. The data in the data repositories appears as files and directories in the cache. If data is not in the cache on first read, Amazon File Cache initiates lazy load to make it available on the cache.

Data-intensive workloads generate a large amount of I/O to files while the workload is running. When processing data intensive workloads with HPC infrastructure, it is important that the data is placed as close to the compute resources as possible for lower latency. Due to latency issues, it is not realistic to move large data sets whenever necessary or access files multiple times that are located at a distant place. Therefore, for data-intensive workloads, it is critical to place data in the location where the workload will run. If the primary storage is in the on-premises data center, there should be a copy of the data on AWS before a workload is run.

AWS DataSync is a service that allows movement of data in and out of AWS for timely in-cloud processing, which can be used for this type of scenario. Depending on the usage pattern or operational policy, you can choose to sync your entire storage data (or a subset of the data) to optimize cost. When a secure network connection such as AWS Direct Connect or AWS Site-to-site VPN is already configured, data can be synced securely in between the on-premises storage system and AWS.

In general, Amazon S3 is a good choice as a target storage system for data transfers due to its durability, low cost, and capability to integrate with other AWS services. Associating an Amazon

FSx for Lustre file system with an S3 bucket makes it possible to seamlessly access the objects stored in your S3 bucket from Amazon EC2 instances that mount the Amazon FSx for Lustre file system. Once the job is complete, results can be exported back onto the data repository.



Reference architecture: Hybrid deployment

Workflow steps

1. User logs on to the Login Node or HPC Management Portal and submits a job through the mechanism available (for example, scheduler, meta-scheduler, or commercial job submission portals).
2. The job is run on either on-premises compute or AWS infrastructure based on configuration.
3. The job access shared storage based on their run location. Depending on whether the workload is data light or data heavy, files should be placed in a strategically chosen place. For data-light workloads, if user wants to run on AWS and data is not available on the cache, Amazon File Cache initiates lazy load before data is processed by the compute fleet. For data-intensive workloads, files should have already copied over to Amazon S3 using AWS DataSync before the job is run on AWS. Once the job starts running, Amazon FSx for Lustre initiates lazy load from Amazon S3 to Lustre.

Operational excellence

The operational excellence pillar contains best practices for organizing your team, designing your workload, operating it at scale, and evolving it over time to deliver business value and continuously improve supporting processes and procedures.

Topics

Best practices

There are four best practice areas for operational excellence in the cloud.

For more information on these areas, see the AWS Well-Architected Framework whitepaper.

Organization

- HPCOPS01-BP01 Start from business objectives and organizational priorities, and drive operational decisions about your HPC environment from them

Prepare

- HPCOPS02-BP01 Evaluate options for scheduling jobs in your cloud environment
- HPCOPS03-BP01 Understand your data movement requirements
- HPCOPS04-BP01 Minimize impact when migrating users and their jobs between HPC environments
- HPCOPS04-BP02 Implement your environments with infrastructure as code and version control your deployments
- HPCOPS05-BP01 Predict how your system will respond to failures and design your operational management to mitigate these.

Operate

- HPCOPS06-BP01 Test and observe job-level performance for every change

Evolve

- HPCOPS07-BP01 Review and test the latest service, product and software updates for business fit and reduction in operational overhead.

Focus areas

- Organization
- Prepare
- Operate
- Evolve

Organization

HPCOPS01: What factors drive operational design decisions for your HPC environment?

Understand how the benefits of the cloud such as scalability and elasticity are used in your HPC environment so that you can make informed decisions about your operational architecture. This understanding helps you avoid bias from a fixed cluster mindset influencing any assumptions made when designing your architecture and help your organization use the broadest range of cloud benefits applicable to your use case.

There are also a number of factors related to your organization to consider when designing your HPC system. Will the engineers value having the flexibility to manage the underlying infrastructure systems to tune the environments for their needs, or would they value ease of use above this? What kind of administration and protection do you need to implement centrally in these environments, and what capabilities do you have to manage the maintenance of such a system.

HPCOPS01-BP01 Start from business objectives and organizational priorities, and drive operational decisions about your HPC environment from them

Collect business objectives and priorities from your users and organizational stakeholders to guide your operational decisions. Use these in combination with price objectives to design the HPC system, as opposed to starting with hardware requirements, in order to take the most advantage of the cloud. Examples of business requirements may include number of completed jobs per month,

wait times for jobs to begin, requirements for runtimes, ability to specify priorities for jobs which affect the scheduling, and managing cluster access for fair usage from all users over time.

Traditional HPC environments are fixed and shared across users, which brings the benefit of a familiar system for users. However, in the cloud, there are flexible ways to build and operate your HPC environment. One approach is to allow your users, or individual departments the flexibility of creating and evolving their own environments, while applying guardrails at the account or organizational level to provide a safe environment for experimentation. Examples of such guardrails could be limiting access to particular instance types and AWS services, and enforcing tagging on all deployed instances so that you can use tag-based observability mechanisms to attribute cost to business departments. Alternatively, you could provide a managed cluster to all end-users with centralized administration. This HPC environment would be similar to a traditional HPC environment. In this case, end-users would not require permissions to modify infrastructure in your cloud environment but would need a unified interface to handle jobs and data management.

Your organizational objectives will help you to choose the right approach. For example, if your primary motivation is to increase the pace of research in your organization by leveraging the latest domain specific AWS products and features, and reducing the dependency on central IT systems, then the first approach of distributed infrastructure management would be a strong fit. If it is more important to maintain ease of use for end-users and get them up and running in a familiar environment, while introducing background improvements such as performance uplift through new instances, and reduced operational overhead through managed service options, then the second approach is a more natural starting point for your cloud deployment.

Additionally, the second approach also allows your central IT team to enforce specific rules that you may require for security and compliance reasons in a straightforward and familiar way. Conversely, if your organization is split into departments with very different IT requirements, or you need to clearly attribute costs back to each department, a distributed infrastructure across separated accounts in an AWS Organization will allow you to simplify the management of separating environments and handling chargeback.

Implementation guidance

- Collect, organize and discuss your stakeholder requirements.

After collecting requirements from your business stakeholders, you are likely to end up with a combination of functional and design implementation requirements that you should separate. By starting your environment design from the functional requirements, you may design architectures

for example that do not tie you into specific hardware configurations, specific cluster sizes, and may highlight opportunities to simplify your operational burden given that some of the restrictions in fixed-cluster environments no longer apply.

Also aim to create a budgeting model that minimizes the need for users to tie into particular hardware configurations, to make it easier to leverage the flexible hardware choices and take advantage of new hardware as it becomes available. Then take the benefits of this cloud native design, and compare it to the design implementation requirements. Where a discrepancy exists, discuss with the original stakeholder to make an informed tradeoff.

- If you grant users the ability to manage infrastructure, implement guardrails that enforce your organizational requirements.

We inherit broader cloud best-practices and learnings for setting up cloud environments with distributed infrastructure administration, which are well documented in the Operational Excellence pillar of the AWS Well Architected whitepaper, many of which can be carried over into HPC contexts. Specifically for HPC environments, see The plumbing: best-practice infrastructure to facilitate HPC on AWS. The Landing Zone Accelerator on AWS Solution is discussed, which proposes a solution with multiple AWS accounts. Such a design logically separates different HPC clusters and customizes them to specific groups or departments while providing a boundary for administration and reducing impact on other environments. While it is possible to self-build and manage a landing zone, best-practices (including for Landing Zone Accelerator) leverage AWS Control Tower, which is a managed service purpose-built for this task.

- Aim to map any architecture decisions to managed products and services, or supported solutions to reduce your operational burden.

Once the stakeholder requirements have been organized, aim to map their infrastructure implementation to managed products and services which have defined support models. Understand which parts of your environment will be supported and at what level, and identify early where any operational risks remain. Reference or illustrative solutions can help to stitch together complex infrastructures quickly for demonstrative purposes, but for your production environments lean towards using supported building blocks that can be assembled together in a well-understood fashion. This will reduce your operational burden for proactive maintenance, and give you clear points of contact for reactive support of your environments.

For more information on the latest AWS supported HPC components, see High Performance Computing. Each of these components offer functionality for common sets of requirements in HPC environments. For example, AWS Parallel Computing Service offers features to scale compute capacity to run submitted jobs and manage cluster operations. Research and Engineering Studio on AWS offers features that allow administrators to manage projects and map user identities to their cloud AWS HPC environment, and allows users to manage sessions and run interactive applications with remote visualization.

AWS HPC Partner organizations can provide solutions with additional functionality, a range of implementation and maintenance support options, and industry focused guidance. The AWS HPC blogs highlight common architectural patterns that combine various solutions and services, and also offer guidance on industry specific patterns. These patterns and partner offerings can combine to help you build up your environment with clear lines of support.

Prepare

HPCOPS02: How do you plan to schedule and run your batch jobs in the cloud?

In a traditional HPC environment, there is typically a single scheduling system that is used to handle batch jobs with a variety of characteristics. In the cloud there are a number of options for job scheduling and orchestration, which address different requirements you may have.

- Are your users comfortable with a particular scheduler already, and would they like to continue using this consolidated scheduling approach?
- Do you need to integrate an on-premise environment with your cloud solution?
- Do the run characteristics of your workload impact the way they are scheduled?

HPCOPS02-BP01 Evaluate options for scheduling jobs in your cloud environment

If you have an existing system, consider how you currently schedule and manage jobs and if it meets your current requirements. Considering whether you want to complement, augment or replace your current system with your cloud system, determine the level of integration needed between your hybrid environments. Determine whether you want to integrate a traditional scheduler with flexible cloud provisioning, use a cloud native scheduling mechanism, such as

AWS Batch, or develop job orchestration to create an ephemeral cluster for each job. Cloud offers various flexible and efficient ways to manage jobs and orchestrate infrastructure.

Implementation guidance

If you have a simple workflow where you need to run a single job or a small set of jobs without the overhead of a scheduler, implement an event-driven pattern that can run your job directly and tear down the resources automatically.

In a case where batch jobs are not running continuously and there is significant time where your cloud cluster is unused, it may be worth considering additional operations of tearing down your cluster and recreating it either on a fixed schedule or on-demand. This may increase the latency with which the first job begins running and forces any environment customizations to be scripted for repeatability, but can optimize costs. It is important to separate the compute and storage requirements in such a scenario, so that the compute cluster can be deleted and recreated without affecting files on shared file systems. To carry this process even further, you may consider mounting multiple file systems into your cluster and persisting some of them but deleting others.

The infrastructure as code example on GitHub: Event-driven weather forecasts shows you how this case can be implemented using an event driven pattern to create an ephemeral cluster to run a single job and store the results without manual intervention. In this case, it is for weather simulations that need to occur periodically, and when not in use, as much of the infrastructure as possible is removed to optimize costs.

- If using a scheduler, evaluate cloud-native schedulers and traditional HPC schedulers with cloud integrations with the level of operations management.

A traditional HPC scheduler can offer benefits such as familiarity for your system end-users, and minimal or no changes to your existing job scripts. Implementations such as AWS ParallelCluster enable you to leverage these traditional schedulers while still taking advantage of cloud benefits by scaling compute capacity up when jobs are submitted to the scheduler, and down when there are no more remaining jobs to optimize cost. Managed implementations such as AWS Parallel Computing Service take this one level further, and handle the operational management of the head-node for you, including aspects such as failover and system upgrades.

Meanwhile, cloud-native schedulers can offer reduced operational overhead, and workflow level integrations to abstract away concepts such as head nodes and compute pools from end-users. They can be a great choice when running standardized workflows and pipelines of tasks. For

example AWS Batch is a cloud native scheduler that can integrate with services such as AWS Step Functions for complex workflow logic, as well as domain-specific workflow languages such as NextFlow.

You may choose to implement multiple types of scheduling solutions to suit differing applications, user needs, and job profiles. Alternatively, you might choose a traditional scheduler to meet current user expectations and modernize their workflow in phases. This is often an attractive choice in large organizations with multiple research departments with well-established workflows.

HPCOPS03: Does your use case require data movement between separated environments, and how is this handled?

Will your users be moving data between separated environments, such as an on-premises cluster and the cloud, and if so, do you know the predicted amount and movement patterns? Do you want to enable a seamless data management workflow for movement and archiving for ease of use, or do you want users to make an intentional choice of where they run their workloads at submission time? Have you considered alternative options to minimize the required data movement, such as remote visualization solutions? See Scenarios for additional considerations.

HPCOPS03-BP01 Understand your data movement requirements

In general, moving data back and forth regularly should be avoided unless absolutely necessary. Identify early however if data needs to be loaded to a location before a job starts and if results need to be copied out on job completion, and whether this can occur asynchronously or not. Identify how users are interacting with each of your environments, and whether they are taking advantage of the benefits of each one as far as reasonable.

For example, some environments may offer latency benefits to large datasets and some may offer more flexible hardware choices, and jobs should be distributed accordingly. Being intentional about these trade-offs at job submission time helps you verify that you are only moving data when it is beneficial to do so.

Consider extending the HPC system to include virtual desktop infrastructure (VDI) solutions and/or automated data processing steps to avoid the need for data movement for pre-processing and post-processing, centralize access control, and reduce the security exposure of your files, whilst reducing the need to manage the operations to handle regular data movement.

Implementation guidance

- Implement remote visualization solutions to minimize data movement, saving movement time and cost, and centralizing file access controls.

Start by implementing AWS Research and Engineering Studio to streamline your VDI requirements while addressing other HPC management needs like project budgeting. Another option if you are using a traditional HPC scheduler is to implement visualization queues, as demonstrated in the blog post for AWS ParallelCluster: Elastic visualization queues with Amazon DCV in AWS ParallelCluster. VDI solutions offer the additional advantage of accessing graphics-accelerated instances when required.

- Schedule jobs to run near existing data

Configure automatic job scheduling to run computations close to data sources, eliminating separate data transfer operations. Alternatively, if user-controlled job placement is preferred, ensure data location visibility to help users select optimal computing environments. For data movement and hybrid storage considerations, see the Performance Efficiency pillar.

HPCOPS04: How will you handle future environment updates with minimal user impact?

With constantly improving hardware, service, and product offerings and patches, it is worthwhile considering how you can design your system upfront to allow for easy replacement of modules and phased migrations between environments with minimal effort and automated testing.

HPCOPS04-BP01 Minimize impact when migrating users and their jobs between HPC environments

Standardize access across HPC environments, and retain and migrate data in the cases of environment upgrades or migrations. If you are using a scheduling mechanism, understand how these can be migrated to different environments, and the impact on running jobs. In some HPC cluster environments there are long running jobs that run over time periods that may otherwise be used as maintenance windows such as weekends. In such cases you may consider having a longer period for blue or green migrations, where new jobs are submitted to the new cluster and the old cluster is given multiple days before deleting to complete all jobs.

Separate your file system from the lifecycle of your HPC environment, and implement regular backups. For example, while a tool like AWS ParallelCluster is able to create an FSx for Lustre file system for you, choose to create a file system separately and reference that in your cluster deployment. This will allow you to implement strategies such as ephemeral compute clusters and upgrade your clusters independently of non-scratch data. These file systems can also be simultaneously mounted to your new cluster and old cluster, helping provide a seamless transition between environments for end-users.

Consider decoupling the cluster access from the user access, for example with an Application Load Balancer (ALB), Elastic IP addresses, or abstracting the submission to the scheduler with a user facing submission form that can be connected to different schedulers, such as Open OnDemand. This will allow you to replace the cluster transparently to the user and allow you to implement migration mechanisms such as gradual weighted blue or green deployments.

Implementation guidance

Manage your data operations separately to the lifecycle of your compute environment.

When migrating between compute environments, users' data should be preserved as far as possible. Create file systems separately to the infrastructure as code stacks that define the compute environments, and reference the file systems to import them into your cluster where possible. If using AWS ParallelCluster for example, you should mount existing file systems in the SharedStorage section of the cluster configuration file. You can then handle the operations of different compute environments flexibly, and for example integrate different compute orchestration services, while providing a single location for end-users to store their data.

Educate users if you intend to treat particular file systems as ephemeral or scratch, so that they know any data stored on these file systems will be lost between environment changes. This may be desirable for some use cases where temporary data is created during job runs and not automatically deleted, and in such cases an intentional choice can be made to not carry this data between clusters. This also allows you to handle the operations of your data in a more tailored way. For example, performing automated backups of your persistent file systems, whilst optimizing costs on your scratch file systems.

HPCOPS04-BP02 Implement your environments with infrastructure as code and version control your deployments

Implementing your environment with infrastructure as code (IaC) as much as feasible, and complementing it with clear documentation steps for components that cannot be automated

allows you to automate your deployments. These templates can then also be put under version control, which allows you to track changes between deployment versions, and provides a centralized location for different stakeholders in your organization to observe and approve operational changes. This also gives you the ability to reproduce environments for use cases such as results verification and reproducibility, and the ability to fail back to old versions in the case of regressions.

See Operational excellence pillar of the AWS Well Architected whitepaper for guidance on using infrastructure as code for your deployments. There are a few common customizations to this for HPC environments. One aspect is that HPC codes are often compiled such that a shared POSIX compatible file system is required, and these compilations can also be lengthy. Therefore, it often makes sense to leave the shared file system which stores the applications running even if the rest of the environment scales up and down elastically.

Another aspect is that HPC instance capacity may be deployed in specific availability zones (AZ). If you use this capacity, parametrize or create a mapping of desired availability zones in your infrastructure as code (IaC) templates to keep them flexible across regions. Similarly, if you have deployed file systems in a particular availability zone that need to be mounted to your environment, your cluster should also be deployed in that availability zone to prevent data transfer between availability zones.

Implementation guidance

- Utilize tools such as AWS CloudFormation and bootstrap scripts to define your HPC environments with code.

Tools such as AWS ParallelCluster themselves are forms of IaC, but can also sit within broader AWS CloudFormation IaC scripts, as detailed in the tutorial: Creating a cluster with AWS CloudFormation. This allows you to provision full stack deployments from these scripts, and examples of such environments and helper scripts for further customization are documented in the HPC Recipes for AWS repository.

HPCOPS05: How will your system respond to failures and anomalies?

- Have you designed your architecture to mitigate predictable failure modes of your system and user jobs?

- How easy will it be to diagnose and correct various error sources, and are there opportunities to automate responses?

While we will test the system and implement recovery strategies in the Reliability pillar of the lens, planning for predictable failure modes and working backwards to architect solutions will have implications for your operational decisions.

HPCOPS05-BP01 Predict how your system will respond to failures and design your operational management to mitigate these

For some of the potential failure modes you identify, you may be able to mitigate them entirely by considering alternative services and products early that reduce your operational burden. For others you may have to implement automated responses or documented runbooks as part of your reliability planning.

Specifically, for HPC environments, there are a number of operational procedures you can modify when considering failure modes. Determine and configure the behavior of your scheduler in the case of compute node failures, for example if it resubmits jobs and/or notifies users. If the head node is self-hosted, consider designing procedures to handle its failure. For example, you may choose to implement an alerting operation so you can manually intervene or opt to add an active failover head node to avoid interruption in cluster operations.

For tightly coupled HPC jobs, architecting for job-level resiliency at runtime may come at the expense of job performance or not be possible at all (i.e. any compute node failure will result in total job failure), and so alternatives such as checkpointing your state for long running jobs and resubmitting jobs automatically in the case of infrastructure failure should be implemented where possible.

Implementation guidance

- Minimize your operational burden by choosing managed services where possible and configuring your environment to automate recovery.

Choosing managed services and features for your storage and scheduling systems reduces your operational burden, for example AWS Parallel Computing Service, and using the persistent file system mode in Amazon FSx for Lustre. At the scheduler level, implement job retries on failure.

For example, if using AWS Batch you can implement Automated job retries strategies to take action based on the reason for failure.

Operate

HPCOPS06: How do you monitor your workloads to verify they are operating as expected?

In HPC environments, job-level performance is often one of the most important characteristics not just for end-user experience but also for cost optimization. This performance can be affected by a number of different factors and updates or changes can cause unexpected performance impacts. Your environment is also likely to change frequently with small updates such as for packages and drivers, and any impact of these may go unaccounted for if we only test performance for major upgrades. In such cases, it may be worthwhile creating a procedure that all changes have to go through before being implemented, and adding a step to this procedure which tests if the change impacts your runtime performance.

HPCOPS06-BP01 Test and observe job-level performance for every change

Before moving users to a new cluster environment, you should run a set of representative HPC job benchmarks to confirm that your system is performing as expected. To verify that this performance is maintained, consider periodically rerunning these benchmarks or a subset so that any unexpected changes can be localized and investigated early.

As the projects of your users evolve, the requirements and usage of your HPC environment will also change from the initial set of jobs on which your representative set of testcases were built. To verify that the performance tests are relevant, you could periodically update your testcases, or you can consider alternative methods such as monitoring the jobs that your users are running. For example, by monitoring job logs or percent usage by department or user, you may proactively detect anomalies. You can then investigate whether these anomalies were caused by a known change in usage patterns, or an unexplained performance regression. Set alerts and automated responses where appropriate.

Performance regressions can go undetected as they may not throw any errors, but can result in longer running jobs and increased cost per job. Consider adding operational mechanisms to track metrics of your jobs and building them into a cohesive dashboard. You can use these collected metrics to tune your environment based on real usage, such as rightsizing the tier of throughput and capacity of your file systems, or adding new compute options similar to hardware configurations that are currently oversubscribed.

Implementation guidance

Log job-level statistics, track anomalies and integrate your environment logging into a dashboard.

There are a number of options for tracking the operational performance of your HPC environment which vary in the level of granularity the offer and operational overhead required to run them. Most HPC schedulers have their own tools to track job level metrics, and these can be the easiest place to start as they natively integrate with the scheduler.

If using AWS ParallelCluster with Slurm, leverage Slurm accounting with AWS ParallelCluster to log job-level statistics in an external database. You can then add a method to visualize these metrics so you can easily gain a view across your environment. The ParallelCluster Monitoring dashboard repository is an example of how you can construct a dashboard to track job data. If using AWS Batch, a similar tool is the AWS Batch Runtime Monitoring Dashboards Solution.

Higher level alternative or complementary tracking methods such as tagging cloud resources by project and using them to drill down into cost reports using AWS cost allocation tags to detect anomalies can offer a similar effect with lower operational overhead but reduced granularity. For more information, see Organizing and tracking costs using AWS cost allocation tags. Many tools such as AWS ParallelCluster resources and tagging and AWS Batch resource tagging: Tag your AWS Batch resources integrate with this mechanism natively to simplify automated tagging.

Evolve

HPCOPS07: What mechanisms are you using to keep your environment updated with relevant service, product, and software improvements?

Having designed your system to allow for upgrades and modular additions of new components, consider how you can keep up to date with and test the latest releases of services, products and software.

- Consider evolving the way your users interact with the environment and carry out their operations.
- Look at offering additional functionality to encourage the behavior of end users. For example, offering additional compute instances for different stages of users' workflows.

- Educate users on the differences between environments to help them make choices, such as by providing demonstrations of new workload specific solutions that reduce their operational overhead.
- Collect end-user feedback on the systems already in place, and implement mechanisms to respond to requests and explore options to evolve your environment or highlight areas to educate end-users.

HPCOPS07-BP01 Review and test the latest service, product and software updates for business fit and reduction in operational overhead

Consider creating a test environment where you can review your representative workload benchmarks on new instances and software versions. If using AWS Organizations, create your test environment within a designated Sandbox organizational unit (OU). For more information, see Organizing Your AWS Environment Using Multiple Accounts. Consider also new storage options and configurations which may offer additional functionality such as increased resiliency. Software updates such as communication library upgrades can provide performance improvements on existing hardware, and so should also be tested periodically with your representative benchmarks, as well as new functionality that can improve the user experience.

In addition to performance improvements, new services and products may offer reduced operational overhead. This may be by managing more of the operations of the cluster for you in a prescriptive approach, or may be by offering domain or workload specific interfaces that make it easier for end-users to interact with the system and reduce the customization you have to maintain from a single generic environment. By being able to integrate these new modules into your existing environment, for example by mounting the same file systems, you can enable experimentation and provide a mechanism to asses business fit.

Implementation guidance

Test the latest versions of your chosen products, and review the latest HPC on AWS news.

If using AWS ParallelCluster, periodically deploy test environments with the latest release, as these often have software upgrades included. New major releases also often add new functionality that you can explore to see if it helps your use case. AWS Parallel Computing Service is a good example of a service that offers a way for you to reduce your operational overhead in the management of your cluster, and introduces prescriptive implementations of patterns and features that are commonly required in HPC environments.

To keep updated on new releases relevant to the HPC space more broadly and learn from continuously evolving best practices, periodically review the AWS HPC community site, Day1HPC. Updates such as new instance launches can offer price and performance improvements, and new features can improve the end-user experience or reduce your operations management overhead.

Key AWS services

- AWS ParallelCluster
- AWS Batch
- Amazon S3
- Amazon FSx for Lustre
- AWS Research and Engineering Studio
- AWS Parallel Computing Service

Resources

Documents and blogs:

- The plumbing: best-practice infrastructure to facilitate HPC on AWS
- Managing your costs with AWS Budgets
- Choosing between AWS Batch or AWS ParallelCluster for your HPC Workloads
- Elastic Visualization Queues with Amazon DCV in AWS ParallelCluster
- HPC Recipes for AWS
- Slurm Accounting with AWS ParallelCluster
- Event-driven weather forecasts

Whitepapers:

- Operational excellence
- Operational Readiness Reviews (ORR)

AWS and Partner solutions:

- Landing Zone Accelerator on AWS

- EF Portal
- Open OnDemand
- AWS Batch Runtime Monitoring Dashboards Solution
- AWS ParallelCluster monitoring dashboard

Videos:

- HPC Tech Shorts on YouTube

Training resources:

- Day1HPC Learning tracks

Security

The security pillar encompasses the ability to protect data, systems, and assets by taking advantage of cloud technologies. The AWS Well Architected Security Pillar provides guidance to help you apply best practices, current recommendations in the design, delivery, and maintenance of secure AWS workloads. All of the Security Pillar best practices apply to HPC workloads with the following extensions.

Topics

Best practices

Infrastructure protection

- HPCSEC01-BP01 Separate HPC cluster components in different network layers
- HPCSEC01-BP02 Control traffic flow within your HPC cluster

Data protection

- HPCSEC02-BP01 Enforce encryption at rest in your HPC environment
- HPCSEC02-BP02 Enforce encryption in transit in your HPC environment

Focus areas

Infrastructure protection

HPCSEC01: How do you implement network-layer separation in your HPC cluster?

HPC clusters can be separated by workload components into different network layers based on their data sensitivity, access requirements, and functional purpose. When separated by layers, traffic can be controlled between layers and additional controls can be applied based on layer requirements as part of a defense-in-depth security approach.

Best practices

- HPCSEC01-BP01 Separate HPC cluster components in different network layers
- HPCSEC01-BP02 Control traffic flow within your HPC cluster

HPCSEC01-BP01 Separate HPC cluster components in different network layers

When architecting for SEC05-BP01, you create network layers and separate components into different layers. In HPC clusters, you can separate different cluster components, such as head node, login nodes, and compute resources. For example, with AWS ParallelCluster, the cluster head node can be separated from the compute resources. The head node could be running in a public subnet with the compute fleet running in a private subnet. Additionally, you could further isolate your cluster by running in private subnets with private connectivity to the cluster.

HPCSEC01-BP02 Control traffic flow within your HPC cluster

According to SEC05-BP02, you control traffic flow within your network layers. You permit only the network flows necessary for the components of your workloads to communicate. When running tightly coupled HPC workloads with Elastic Fabric Adapter (EFA), EFA requires being a member of a security group allowing all inbound and outbound traffic to and from itself. Each cluster member will allow all traffic between members when processing the same EFA-based job.

Clusters are commonly used with multiple running jobs and a single security group would be used for all EFA traffic without separation by job. If your environment requires further security separation by job, consider an alternative design, such as multiple clusters or a more advanced security group mapping, rather than having one security group for all traffic between members.

Data protection

HPCSEC02: How do you manage data encryption in your HPC cluster?

As described in the Data Protection section, classifying and protecting data are foundational practices. The same foundational practices apply to the data related to your HPC workloads, and once the data is classified, there are a few additional HPC consideration for protecting it.

Best practices

- HPCSEC02-BP01 Enforce encryption at rest in your HPC environment
- HPCSEC02-BP02 Enforce encryption in transit in your HPC environment

HPCSEC02-BP01 Enforce encryption at rest in your HPC environment

With SEC08-BP01, you securely manage encryption keys to protect data at rest. You also tightly control access through the use of key policies and IAM policies. AWS HPC products, such as AWS ParallelCluster, configure IAM permissions on different components, such as cluster, queues, and login nodes. Therefore, cluster users can unencrypt data at rest by IAM permissions associated with the cluster or component. If your HPC environment needs further isolation, such as by project or group separation, consider an architecture with multiple clusters for data separation by encryption key.

HPCSEC02-BP02 Enforce encryption in transit in your HPC environment

When implementing SEC09-BP02, you enforce encryption in transit. Tightly coupled HPC applications using EFA bypass the operating system kernel and directly communicate with the EFA device rather than traditional TCP/IP networking. This provides low-latency, reliable communication between cluster instances but introduces additional considerations when enforcing encryption in transit compared to traditional TCP/IP approaches.

AWS provides secure and private connectivity between EC2 instances of all types. In addition, some instance types use the offload capabilities of the underlying Nitro System hardware to automatically encrypt in-transit traffic between instances. This encryption uses Authenticated Encryption with Associated Data (AEAD) algorithms with 256-bit encryption, which also helps implement SEC09-BP03. There is no impact on network performance.

Therefore, your EFA traffic is automatically encrypted between cluster members with no impact to performance. This automatic encryption is also used with FSx for Lustre and enforcing encryption in transit between cluster members and an FSx for Lustre filesystem. See Encryption in transit for additional details.

Resources

- Security Pillar - AWS Well-Architected Framework
- Amazon Elastic Compute Cloud: Encryption in transit
- FSx for Lustre: Encrypting data in transit
- Securing HPC on AWS – isolated clusters

Reliability

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it is expected to. This includes the ability to operate and test the workload through its total lifecycle. This paper provides in-depth, best practice guidance for implementing reliable HPC workloads on AWS, and is complementary to the broader reliability pillar.

The reliability pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Reliability Pillar - AWS Well-Architected Framework.

Topics

Design principles

In addition to the design principles outlined in the Reliability Pillar whitepaper, the following pertains specifically to HPC architectures:

- **Assess performance tradeoffs with cost and availability requirements:** In high performance computing applications, performance is key. Whether high performance means executing at sub-microsecond latencies or processing at high throughput, when designing your architecture these requirements will need to be balanced with cost, availability, and disaster recovery considerations.

Best practices

HPC workloads can run for hours to days in duration, and often scale out to thousands and even millions of vCPUs. Traditionally, multiple users often access the same large system and require access to domain-specific applications. The best practices below provide guidance on how to best implement these reliability requirements on the cloud.

Workload architecture

- HPCREL01-BP01 Consider instance type flexibility for your workload
- HPCREL01-BP02 Deploy loosely coupled workloads across multiple Availability Zones

Change management

- HPCREL02-BP01 Install software packages in a shared location to simplify multi-user, multi-resource software requirements management
- HPCREL02-BP02 Use package managers to simplify software dependency management when possible

Failure management

- HPCREL03-BP01 Implement checkpointing
- HPCREL03-BP02 Use durable storage to store and back up datasets

Focus areas

- Foundations
- Workload architecture
- Change management
- Failure management

Foundations

There are no reliability best practices for Foundations specific to this lens. For more information, see Foundations.

Workload architecture

HPCREL01: How does your architecture optimize for capacity pool flexibility?

HPC workloads can potentially scale to hundreds, thousands, and even millions of cores. In order for an HPC job to run reliably, it needs access to the amount of cores or nodes requested. Without the required capacity, a job could fail at runtime, or queue with an insufficient number of nodes provisioned in the AWS account.

When designing your architecture for your required scalability, consider building flexibility in instance choice and deploying across multiple availability zones if your workload supports it, such as with some loosely coupled workloads.

HPCREL01-BP01 Consider instance type flexibility for your workload

Consider using multiple instance types for your workload. HPC orchestration services including Amazon Parallel Computing Service and AWS Batch allow you to select multiple instance types for a given workload. Additionally, in AWS ParallelCluster, multiple instance types can be selected for a single Slurm partition. In AWS Batch, multiple instance types can be specified for a single Compute Environment.

Implementation guidance

- Allow for multiple instance types in your compute environment, and cycle through different instance types if need be.

With AWS Batch, users can list multiple instance types or families for the Compute Environment. For more information, see JobQueueDetail. Multiple instance types can also be configured with AWS ParallelCluster and with Amazon Parallel Computing Service. For more information, see Multiple instance type allocation with Slurm.

Consider starting with AWS HPC instances for best price-performance, and if unavailable, consider cycling to higher cost compute-optimized instances.

- Cycle through capacity pools in different Availability Zones for specific instance types for tightly coupled workloads.

Cycling through capacity pools allows you to identify which Availability Zone within a region meets the capacity requirements for your workload, and to run your job within that Availability Zone. Cycling can be implemented with user scripts, or alternatively within specific orchestrators on AWS. With AWS ParallelCluster, for example, you can create set a multi-availability zone queue (see Multiple Availability Zones now supported in AWS ParallelCluster 3.4), and all or nothing

scheduling (see Minimize HPC compute costs with all-or-nothing instance launching), to help take full advantage of any available capacity pools.

Your shared storage location should match your selected Availability Zone for compute resources. Specifically, if you have an FSx for Lustre filesystem in one Availability Zone, and compute in another, you will incur inter-AZ traffic costs.

HPCREL01-BP02 Deploy loosely coupled workloads across multiple Availability Zones

Deploying across multiple Availability Zones can help improve capacity availability for scale-out workloads, and can also help with disaster recovery. AWS ParallelCluster and AWS Batch both allow for deploying architectures across multiple Availability Zones. Loosely coupled jobs that do not have interdependency on each other should be deployed across multiple Availability Zones. When deploying across multiple Availability Zones, consider cost implications for transferring data across Availability Zones.

Tightly coupled workloads should be deployed within a single Availability Zone, and within a placement group. Using a central data repository, such as Amazon FSx for Lustre or Amazon S3, and utilizing infrastructure as code with CloudFormation templates, allows for recovering to an alternative Availability Zone, even for tightly coupled, single-AZ workloads.

Implementation guidance

Depending on services utilized in your architecture, evaluate and verify that each service utilizes multiple availability zones.

Change management

HPCREL02: How do you reliably manage software packages and dependencies across users?

HPC workloads often consist of software packages with complex dependencies. Managing software and dependencies, with specific versioning requirements, across many users, can be complex. Consider the following best practices when building out your architecture.

HPCREL02-BP01 Install software packages in a shared location to simplify multi-user, multi-resource software requirements management

Installing software packages on shared storage, such as Amazon Elastic File System (EFS), allows users across compute environments and clusters access the same versioned set of packages. Dependencies can also be installed using a custom AMI, or using a post-install script, such as configured with AWS ParallelCluster.

Implementation guidance

Install software in shared location for multi-user or multi-resource environments. Single resource environments can use a custom AMI or separate Amazon Elastic Block Store (Amazon EBS) volume, and multi-resource environments can use shared storage, such as Amazon Elastic File System (Amazon EFS).

HPCREL02-BP02 Use package managers to simplify software dependency management when possible

Managing HPC applications can potentially be simplified with package managers, such as Spack, SBGrid, and EasyBuild. AWS hosts a Spack binary cache for fast installation of commonly used HPC packages and applications. Leveraging a package manager simplifies dependency management for system admins while providing exact versioning per user requirements.

Implementation guidance

Consider a package manager to simplify software dependencies.

Failure management

HPCREL03: How does your application recover from failures?

Failure tolerance can be improved in multiple ways, including checkpointing your applications and backing up data to Amazon S3. Some of these best practices are covered in the primary reliability pillar. See the following for some HPC-specific considerations.

HPCREL03-BP01 Implement checkpointing

For long-running cases, incorporating regular checkpoints in your code allows you to continue from a partial state in the event of a failure. Checkpointing is a common feature of application-level failure management already built into many HPC applications.

Checkpointing is recommended for long running jobs on both on demand and Spot Instances. When using Spot Instances, some applications may benefit from changing the default Spot interruption behavior (for example, stopping or hibernating the instance rather than terminating it). Consider the durability of the storage option when relying on checkpointing for failure management.

Implementation guidance

Implement checkpointing for long-running jobs

The most common approach is for applications to periodically write out intermediate results. The intermediate results offer potential insight into application errors and the ability to restart the case as needed while only partially losing the work. For example, you can implement checkpointing at the application level (see Running cost-effective GROMACS simulations using Amazon EC2 Spot Instances with AWS ParallelCluster) and use the Spot Interruption notices.

HPCREL03-BP02 Use durable storage to store and back up datasets

HPC workloads often require parallel file systems for high throughput I/O. In many cases, data also needs to be stored securely and durably for several years. To achieve both these requirements, store and back up data in durable storage, and use high performant file systems primarily during data processing.

Implementation guidance

Create a Data Repository Association (DRA) between Amazon FSx for Lustre and Amazon S3.

When using Amazon FSx for Lustre, create a data repository association (DRA) with an Amazon S3 bucket. Amazon Simple Storage Service (S3) allows users to store data in highly available, cost-effective object storage, with tiering to manage hot through cold data. Amazon S3 improves reliability by serving as a central repository for workload datasets. With a DRA between S3 and FSx for Lustre, by default, data is automatically transferred from Amazon S3 to Amazon FSx for Lustre when it is first accessed. Amazon FSx for Lustre also supports additional ways of Importing changes from your data repository and Exporting changes to the data repository data.

Key AWS services

- AWS Batch
- Amazon Parallel Computing Service
- AWS ParallelCluster
- AWS Pricing Calculator
- Amazon FSx for Lustre
- Amazon Simple Storage Service (S3)
- Amazon Elastic Block Store (EBS)
- Amazon Elastic File System (EFS)
- AWS CloudFormation

Resources

- AWS Batch Dos and Don'ts: Best Practices in a Nutshell
- Multiple Availability Zones Now Supported in AWS ParallelCluster 3.4
- Deployment and storage options for FSx for Lustre file systems
- Cost Optimization on Spot Instances Using Checkpoints for Ansys LS DYNA
- Running cost-effective GROMACS simulations using Amazon EC2 Spot Instances with AWS ParallelCluster

Performance efficiency

The performance efficiency pillar includes the ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.

The performance efficiency pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Performance Efficiency Pillar whitepaper.

Topics

Design principles

When designing for HPC in the cloud, a number of principles help you achieve performance efficiency:

- **Design the cluster for the application**: Traditional clusters are static and require that the application is designed for the cluster. AWS offers the capability to design the cluster for the application. A one- size-fits-all model is no longer necessary with individual clusters for each application. When running a variety of applications on AWS, a variety of architectures can be used to meet each application's demands. This allows for the best performance while minimizing cost.
- **Test performance with a meaningful use case**: The best method to gauge an HPC application's performance on a particular architecture is to run a meaningful demonstration of the application itself. An inadvertently small or large demonstration case one without the expected compute, memory, data transfer, or network traffic will not provide a meaningful test of application performance on AWS. Although system-specific benchmarks offer an understanding of the underlying compute infrastructure performance, they do not reflect how an application will perform in the aggregate. The AWS pay-as-you-go model makes a proof-of-concept quick and cost-effective.
- **Use cloud-native architectures where applicable**: In the cloud, managed, serverless, and cloud-native architectures remove the need for you to run and maintain servers to carry out traditional

compute activities. Cloud-native components for HPC target compute, storage, job orchestration and organization of the data and metadata. The variety of AWS services allows each step in the workload process to be decoupled and optimized for a more performant capability.

- **Experiment often**: Virtual and automatable resources help you quickly carry out comparative testing using different types of instances, storage, and configurations.

Best practices

The following are the performance efficiency questions, best practices, and prescriptive guidance for high performance computing workloads.

Compute architecture

- HPCPERF01-BP01 Evaluate containers or serverless functions
- HPCPERF02-BP01 Select the best computing instance type for your workload by measuring application performance
- HPCPERF02-BP02 Default to virtualized over bare-metal instances
- HPCPERF03-BP01 Optimize your compute environment for your workload

Storage architecture

- HPCPERF04-BP01 Select the optimal HPC storage solution based on your targeted individual application

Network architecture

- HPCPERF05-BP01 Consider latency, bandwidth, and throughput requirements for HPC workloads

Focus areas

- Compute architecture
- Storage architecture
- Network architecture

Compute architecture

HPCPERF01: How do you select the compute environment for your application?

The optimal compute solution for a particular HPC architecture depends on the workload deployment method, degree of automation, usage patterns, and configuration. Different compute solutions may be chosen for each step of a process. Selecting the wrong compute solutions for an architecture can cause lower performance efficiency.

HPCPERF01-BP01 Evaluate containers or serverless functions

When evaluating compute options, consider containers or serverless functions for your HPC workload or for parts of your surrounding workflow.

Implementation guidance

- Containers are a method of operating system virtualization that is attractive for many HPC workloads, particularly if the applications have already been containerized. AWS services such as AWS Batch, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS) help deploy containerized applications.
- Serverless functions abstract the execution environment. You can use AWS Lambda to run code without deploying, running, or maintaining, an instance. Many AWS services emit events based on activity inside the service, and often a Lambda function can be initiated off of service events. For example, a Lambda function can be run after an object is uploaded to Amazon S3. Many HPC users use Lambda to automatically run code as part of their workflow.

Key AWS services

- Amazon EC2
- AWS Nitro System
- AWS ParallelCluster
- AWS Batch
- Amazon Elastic Container Service (ECS)
- Amazon Elastic Kubernetes Service (EKS)

- AWS Lambda

Resources

- AWS HPC Blog: How to manage HPC jobs using a serverless API
- Bare metal performance with the AWS Nitro System
- Deploying and running HPC applications on AWS Batch
- How to manage HPC jobs using a serverless API
- What is AWS ParallelCluster?
- Performance Efficiency Pillar: AWS Well-Architected Framework

HPCPERF02: How do you select your computing instances?

EC2 instances are virtualized servers and come in different families and sizes to offer a wide variety of capabilities. Some instance families target specific workloads, for example, compute, memory, or GPU intensive workloads. Other instances are general purpose.

Both the targeted-workload and general-purpose instance families are useful for HPC applications. Instances of particular interest to HPC include the HPC Optimized family, the Compute Optimized family and Accelerated Computing instance types that are powered by GPUs and FPGAs.

HPCPERF02-BP01 Select the best computing instance type for your workload by measuring application performance

Select the optimal Amazon EC2 instance type for your workload and consider factors, such as family and generation, to optimize for your desired price-for-performance. With access to on-demand instances, testing different configurations is the best way to determine your desired configuration for each of your workloads.

Implementation guidance

EC2 Instances are available in different generations. Previous generation instances are still fully supported, but we recommend you to use the Amazon EC2 instance types to get the best performance.

Some instance families provide variants within the family for additional capabilities. For example, an instance family may have a variant with local storage, greater networking capabilities, or a different processor. These variants can be viewed in the Amazon EC2 Instance types and may improve the performance of some HPC workloads.

Within each instance family, one or more instance sizes allow vertical scaling of resources. Some applications require a larger instance type (for example, 48xlarge) while others run on smaller types (for example, 2xlarge) depending on the number or processes supported by the application.

For tightly coupled workloads, optimum performance is obtained when the memory of the computing node is not shared between different virtual machines within the same physical host. Therefore, it is recommended to use only EC2 instances whose size is big enough to occupy the entire physical node. This is usually obtained with the largest instance type even if there are some noticeable exceptions. For example, in the 7th generation of HPC instances, each size in the instance family has the same engineering specs, memory access and price, and differs only by the number of cores offered. That means that all the cores in the instances will be able to access the entire host memory regardless of the instance size. So, you can select also a smaller size without warring about the performance impact of sharing the host memory with other virtual machines.

The T-series instance family is designed for applications with moderate CPU usage that can benefit from bursting beyond a baseline level of CPU performance. Most HPC applications are compute-intensive and suffer a performance decline with the T-series instance family.

Applications vary in their requirements (for example, desired cores, processor speed, memory requirements, storage needs, and networking specifications). When selecting an instance family and type, begin with the specific needs of the application. You can also split a specific workflow in its individual steps (for example, mesher and solver in a CFD simulation) and run each step on a different instance type. Instance types can be mixed and matched for applications requiring targeted instances for specific application components. You can use the AWS Management Console or the AWC CLI to search for instances that satisfy your needs.

As an example, you can use the following command to display only current generation instance types with 64 GiB (65536 MiB) of memory:

```
AWS ec2 describe-instance-types --filters
          "Name=current-generation,Values=true"
          "Name=memory-info.size-in-mib,Values=65536" --query
          "InstanceTypes[*].[InstanceType]" --output text |
          sort
```

Testing different instance types is affordable since you only pay for active usage. Even after your initial choice, you can switch instance types whenever your requirements shift.

HPCPERF02-BP02 Default to virtualized over bare-metal instances

Virtualized instances have a faster initialization time and offer indistinguishable performance when compared to bare-metal instances. Unless you specifically require a bare-metal instance, we recommend virtualized instances, especially in dynamic HPC environments.

Implementation guidance

New-generation EC2 instances run on the **AWS Nitro System.** The Nitro System delivers practically all of the compute and memory resources of the host hardware to your instances resulting in better overall performance. Dedicated Nitro Cards enable high-speed networking, high-speed EBS, and I/O acceleration without having to hold back host resources for management software.

The Nitro Hypervisor is a lightweight hypervisor that manages memory and CPU allocation and delivers performance that is indistinguishable from bare metal. The Nitro System also makes bare metal instances available to run without the Nitro Hypervisor. Launching a bare metal instance boots the underlying server, which includes verifying all hardware and firmware components. This means it can take longer before the bare metal instance becomes available to start your workload, as compared to a virtualized instance. The additional initialization time must be considered when operating in a dynamic HPC environment where resources launch and terminate based on demand.

Unless your application specifically requires a bare metal node, we recommend using virtualized instances to avoid the longer boot time with metal instances without a gain in performance.

Key AWS services

- Amazon EC2
- AWS Nitro System

Resources

- Amazon Elastic Compute Cloud: Amazon EC2 instance types
- Amazon EC2 Instance types
- Find an Amazon EC2 instance type
- Specify CPU options for an Amazon EC2 instance

- Getting the best OpenFOAM Performance on AWS
- Optimizing HPC workloads with Amazon EC2 instances

HPCPERF03: How do you optimize the compute environment?

You can optimize your compute environment through multiple components, including the operating system and hardware features. Since running in the cloud provides flexibility, we recommend testing different configurations before determining your final implementation.

HPCPERF03-BP01 Optimize your compute environment for your workload

We recommend optimizing your machine image, application-compile options, instance configuration, and runtime environment when running your HPC applications.

Implementation guidance

- A current operating system running a modern kernel is critical to achieve the best performance and ensuring access to the most up-to-date libraries. An Amazon Machine Image (AMI) is a template that contains the software configuration (operating system, libraries, and applications) required to launch your instance. You can select an AMI with the latest version of the operating system supported by your application. For MPI workloads, it is also important to use a modern MPI version.
- In addition to choosing an AMI, you can further optimize your environment by taking advantage of the hardware features of the underlying processors. There are three primary methods to consider when optimizing the underlying hardware:

1. Advanced processor features
2. Simultaneous multithreading (SMT)
3. Processor affinity

HPC applications can benefit from these advanced processor features (for example, Advanced Vector Extensions) and can increase their calculation speeds by compiling the software for the target CPU architecture. The compiler options for architecture-specific instructions vary by compiler (check the usage guide for your compiler).

AWS enables Simultaneous multithreading (SMT), commonly referred to as Hyper-Threading, by default on most of the EC2 instances. Multithreading improves performance for some applications by allowing two threads to run on the same physical core. This command will give you the list of the EC2 instances that are offered in a location (or Region) with two threads per core:

```
AWS ec2 describe-instance-types --filters
          "Name=current-generation,Values=true"
          "Name=vcpu-info.default-threads-per-core,Values=2"
          --query "InstanceTypes[*].[InstanceType]" --output
          text --region us-east-2 | sort
```

Most HPC applications benefit from disabling multithreading, and therefore, it tends to be the preferred environment for HPC applications. Multithreading is easily disabled in Amazon EC2 by Specify CPU options for an Amazon EC2 instance for your instance. Unless an application has been tested with multithreading enabled, it is recommended that multithreading be disabled and that processes are launched and individually pinned to cores when running HPC applications. CPU or processor affinity allows process pinning to easily happen.

Processor affinity can be controlled in a variety of ways. For example, it can be configured at the operating system level (available in both Windows and Linux), set as a compiler flag within the threading library, or specified as an MPI flag during execution. The chosen method of controlling processor affinity depends on your workload and application.

There are many compute options available to optimize a compute environment. Cloud deployment allows experimentation on every level from operating system to instance type, to bare-metal deployments. Because clusters are tuned before deployment, time spent experimenting with cloud-based clusters is vital to achieving the desired performance.

Key AWS services

- Amazon EC2
- AWS Nitro System

Resources

- Specify CPU options for an Amazon EC2 instance
- Processor state control for Amazon EC2 Linux instances
- Instance sizes in the Amazon EC2 Hpc7 family – a different experience

- Application deep-dive into the AWS Graviton3E-based Amazon EC2 Hpc7g instance

Storage architecture

HPCPERF04: How do you select your storage solution?

AWS offers a wide range of storage services. The right storage choice for optimal performance is specific to your HPC application rather than a one-size-fits-all approach. We recommend testing different configurations to find the most cost-effective solution that meets your performance needs.

HPCPERF04-BP01 Select the optimal HPC storage solution based on your targeted individual application

The optimal storage solution for a particular HPC architecture depends largely on the individual applications targeted for that architecture. Workload deployment method, degree of automation, and desired data lifecycle patterns are also factors. AWS offers a wide range of storage options.

As with compute, the best performance is obtained when targeting the specific storage needs of an application. AWS does not require you to over-provision your storage for a one-size-fits-all approach, and large, high-speed, shared file systems are not always required. Optimizing the compute choice is important for optimizing HPC performance and many HPC applications will not benefit from the fastest storage solution possible.

HPC deployments often require a shared or high-performance file system that is accessed by the cluster compute nodes. There are several architecture patterns you can use to implement these storage solutions from AWS Managed Services, AWS Marketplace offerings, APN Partner solutions, and open-source configurations deployed on EC2 instances.

Implementation guidance

- Amazon FSx is a suite of AWS managed services designed to help customers to deploy and manage file systems in the cloud. It supports a wide range of workloads with its reliability, security, scalability, and broad set of capabilities. In particular, Amazon FSx for Lustre is a managed service that provides a cost effective and performant solution for HPC architectures requiring a high-performance parallel file system. Similarly, Amazon FSx for OpenZFS is a fully

managed file storage service that provides a ZFS file system. Based on your application needs, you can explore additional file systems managed by Amazon FSx such as NetApp ONTAP and Windows File Server (SMB).

- Shared file systems can also be created from Amazon Elastic File System (EFS) or EC2 instances with Amazon Elastic Block Store (EBS) volumes or instance store volumes. Frequently, a simple NFS mount is used to create a shared directory.

When selecting your storage solution, you can use EBS for either or both of your local and shared storages disks. EBS volumes are often the basis for an HPC storage solution. Various types of EBS volumes are available including magnetic hard disk drives (HDDs), general-purpose solid-state drives (SSDs), and provisioned IOPS SSDs for high IOPS solutions. They differ in throughput, IOPS performance, and cost.

You can gain further performance enhancements by selecting an Amazon EBS-optimized instance.

An Amazon EBS-optimized instance types uses an optimized configuration stack and provides additional, dedicated capacity for Amazon EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between Amazon EBS I/O and other network traffic to and from your instance. Choose an EBS-optimized instance for more consistent performance and for HPC applications that rely on a low-latency network or have intensive I/O data needs to EBS volumes.

To launch an EBS-optimized instance, choose an instance type that enables EBS optimization by default, or choose an instance type that allows enabling EBS optimization at launch.

Instance store volumes, including nonvolatile memory express (NVMe) SSD volumes (only available on certain instance families), can be used for temporary block-level storage. For EBS optimization and instance-store volume support, see Amazon EC2 Instance types.

When you select a storage solution, align the solution with your access patterns to achieve the desired performance. It is easy to experiment with different storage types and configurations. With HPC workloads, the most expensive option is not always the best performing solution.

Key AWS services

- Amazon FSx
- Amazon FSx for Lustre
- Amazon FSx for OpenZFS

- Amazon Elastic File System
- Amazon Elastic Block Store
- Instance store temporary block storage for EC2 instances

Resources

- Amazon EC2 Instance types
- Deep dive on accelerating HPC and ML with Amazon FSx
- Amazon FSx for Lustre performance
- Using Lustre to build very fast file systems with Amazon FSx for Lustre
- How Amazon File Cache works

Network architecture

HPCPERF05: How do you select your network solution?

Loosely and tightly coupled workloads have different network communication patterns and needs. When selecting your network solution, consider your workload's network latency, bandwidth, and throughput requirements.

HPCPERF05-BP01 Consider latency, bandwidth, and throughput requirements for HPC workloads

The optimal network solution for an HPC workload varies based on latency, bandwidth, and throughput requirements. Tightly coupled HPC applications often require the lowest latency possible for network connections between compute nodes. For moderately sized, tightly coupled workloads, it is possible to select a large instance type with a large number of cores so that the application fits entirely within the instance without crossing the network at all.

Alternatively, some applications are network bound and require high network performance. Instances with higher network performance can be selected for these applications. The highest network performance is usually obtained with the largest instance type in a family. Refer to the Amazon EC2 Instance types for more details.

Implementation guidance

Use cluster placement groups and Elastic Fabric Adapter for tightly coupled applications.

Multiple instances with low latency between the instances are required for large tightly coupled applications. On AWS, this is achieved by launching compute nodes into a cluster placement group, which is a logical grouping of instances within an Availability Zone. A cluster placement group provides non-blocking and non-oversubscribed connectivity, including full bisection bandwidth between instances. Use cluster placement groups for latency sensitive tightly coupled applications spanning multiple instances.

In addition to cluster placement groups, tightly coupled applications benefit from Elastic Fabric Adapter (EFA), a network device that can attach to your Amazon EC2 instance. EFA provides lower and more consistent latency and higher throughput than the TCP transport traditionally used in cloud-based HPC systems. It enables an OS-bypass access model through the *Libfabric* API that allows HPC applications to communicate directly with the network interface hardware. EFA enhances the performance of MPI and NCCL inter-instance communication, is optimized to work on the existing AWS network infrastructure, and is critical for scaling tightly coupled applications.

If an application cannot take advantage of EFA's OS-bypass functionality, or an instance type does not support EFA, optimal network performance can be obtained by selecting an instance type that supports Elastic Network Adapter (ENA) or ENA Express. ENA provides EC2 instances with higher networking performance and lower CPU utilization through the use of pass-through rather than hardware-emulated devices. This method allows EC2 instances to achieve higher bandwidth, higher packet-per-second processing, and lower inter-instance latency compared to traditional device virtualization.

ENA is available on all current-generation instance types and requires an AMI with supported drivers. Although most current AMIs contain supported drivers, custom AMIs may require updated drivers. For more information on enabling enhanced networking and instance support, refer to the Enhanced networking on Amazon EC2 instances.

ENA Express is an ENA feature that can be enabled on certain EC2 instances and it is designed to increase the single flow bandwidth and lower the tail latency of network traffic between EC2 instances. Workloads such as distributed storage systems and live media encoding that require large flows and are sensitive to variance in latency can benefit from ENA Express.

- Distribute your jobs across multiple Availability Zones or Regions for loosely coupled workloads.

Loosely coupled workloads are generally not sensitive to very low-latency networking and do not require the use of a cluster placement group or the need to keep instances in the same Availability Zone or Region.

- Use the Instance Type Matrix to determine the right instance type for your network bandwidth needs.

In some cases, the available network bandwidth depends on the type of network traffic. For example, HPC optimized instances, can access a higher network bandwidth when used together with EFA. The Amazon EC2 Instance types page contains all the networking details that will help you to select the right instance type for your network bandwidth needs.

Key AWS services

- Placement groups for your Amazon EC2 instances
- Elastic Fabric Adapter for AI/ML and HPC workloads on Amazon EC2
- Enable enhanced networking with ENA on your EC2 instances
- Improve network performance between EC2 instances with ENA Express

Resources

- Amazon EC2 Instance types
- How EFA works and why we don't use InfiniBand in the cloud
- NCCL on EFA

Cost optimization

The cost optimization pillar includes the continual process of refinement and improvement of an HPC system over its entire lifecycle. From the initial design of your first proof of concept, to the ongoing operation of production workloads, adopting the practices in this paper enables you to build and operate cost-aware systems that achieve business outcomes and minimize costs. This allows your organization to maximize its return on investment. Review the corresponding section in the AWS Well-Architected Framework whitepaper.

Topics

Design principles

Several strategies help reduce costs when running high performance computing (HPC) workloads in the cloud such as:

- **Find your desired price-for-performance configuration:** Many HPC workloads are part of a data processing pipeline that includes multiple steps. These will vary depending on the workload and often include; data transfer in, pre-processing, computational calculations, post-processing and data transfer out. In the cloud, rather than on a large and expensive server, the computing platform can be optimized at each step. For example, if a single step in an HPC workflow pipeline requires a large amount of memory such as mesh or grid generation, you only need to pay for a more expensive large-memory server for the memory-intensive application. Other parts of the workflow should be similarly deployed on EC2 or storage configurations best suited for that job. When comparing different configurations, think about the cost as well as the performance.
- **Burst workloads in the most efficient way:** Savings are obtained for HPC workloads through horizontal scaling in the cloud. When scaling horizontally, many jobs or iterations of an entire workload are run simultaneously for less total elapsed wall time. Depending on the application, horizontal scaling can be cost neutral while offering indirect cost savings by delivering results in a fraction of the time.

- **Evaluate spot pricing:** Amazon EC2 Spot Instances offer spare compute capacity in AWS at steep discounts compared to On-Demand instances. However, Spot Instances can be interrupted when EC2 needs to reclaim the capacity. Spot Instances are frequently the most cost-effective resource for flexible or fault-tolerant workloads. The intermittent nature of HPC workloads makes them well suited to Spot Instances. The risk of Spot Instance interruption can be minimized by working with the Spot Advisor, and the interruption impact can be mitigated by changing the default interruption behavior and using Spot Fleet to manage your Spot Instances. The need to occasionally restart a workload is easily offset by the cost savings of Spot Instances. Some HPC workloads can be checkpointed and resumed, this can make spot a very good fit.
- **Assess the trade-off of cost versus time**: Tightly coupled, massively parallel workloads are able to run on a wide range of core counts. For these workloads, the scaling or parallel efficiency of a case falls off at higher core counts. Curves are specific to the workload (model size, application, solver algorithm, IO, user defined functions) as scaling depends significantly on the ratio of computational load to non-parallelizable components of the solve time (network latency, serial operations, and other blocking functions). Larger models are generally capable of scaling further than smaller models. Running on less cores is generally more cost efficient than running on more cores, at the cost of increasing turnaround-time. With an understanding of the cost versus turnaround-time tradeoff, the most appropriate scaling can be used.

For example, if a job is being submitted over the weekend and the results needed on Monday, there is no value in running on more cores with a lower parallel efficiency and incurring extra costs to get the results ready by Saturday morning. Alternatively, if the results of a calculation are in the critical path of releasing a new product, reducing the time-to-solution to allow for more design iteration or result analysis may be very worthwhile. In addition, consider software licenses when thinking about overall cost. Some licenses can be expensive, and it may be overall more cost effective to run your workload less efficiently to reduce turnaround time and licensing cost.

- **Choose Region based on workloads requirements and cost goals:** AWS has many regions around the world. Capabilities vary between different regions. For example, some instance types may not be available in all regions, moving your calculations to a different region might allow the use of a more efficient instance type or storage configuration. Costs can also vary between different regions. In some cases, you may have regulatory or legal requirements that restrict choice of the regions you can use. In other cases, you may be free to use resources anywhere in the world. Balance the complexity of running in multiple regions with cost savings.

Best practices

Practice Cloud Financial Management

- HPCCOST01-BP01 Use the right tools to collect and analyze the data you need.

Cost effective resources

- HPCCOST02-BP01 Use the most appropriate instances and resources

Focus areas

- Practice Cloud Financial Management
- Expenditure and usage awareness
- Cost effective resources
- Manage demand and supplying resources
- Optimize over time

Practice Cloud Financial Management

HPCCOST01: How do you keep track of expenditure used for HPC?

HPC workloads can use a large number of resources in a short space of time. Use of cloud compute changes the way users decide what to run, where decisions on what jobs will be run are budget based. Using the following best practices, on-going expenditure can be tracked.

HPCCOST01-BP01 Use the right tools to collect and analyze the data you need.

There are many ways to keep track of costs using the standard AWS tools, which will vary depending on the chosen architecture. Please refer to the Cost Optimization Pillar - AWS Well-Architected Framework.

Implementation guidance

- For HPC applications, it is very common to use tagging. Resources used for jobs can be tagged with project names, user ids or any other attributes you choose. Once resources are tagged

with tags activated for cost allocation, you can generate reports based on the attributes chosen earlier.

- If a queuing system, such as Slurm or IBM Spectrum LSF Suites is in use, they typically have ways to log the usage of resources, such as Slurm Accounting or LSF Analytics. Details vary depending on the system in use.

Key AWS services

- AWS Cost Explorer
- Slurm accounting with AWS ParallelCluster
- Metrics in Amazon CloudWatch

Expenditure and usage awareness

There are no best practices unique to HPC for the Expenditure and usage awareness best practice area. Please review the corresponding section in the AWS Well-Architected Cost Optimization Pillar whitepaper.

Cost effective resources

HPCCOST02: How have you evaluated the trade-offs between job completion time and cost?

With greater resource availability there is greater capability to run jobs on faster – or more – compute resources. Even though this may result in a faster turnaround of a job (based on wall clock time), it could also result in a much greater cost of running that job if speed up is non-linear. For every workload there is a sweet-spot for run time and cost.

HPCCOST02-BP01 Use the most appropriate instances and resources

Using the appropriate instances and resources for your system is key to cost management. The technology choice may increase or decrease the overall cost of running an HPC workload.

Implementation guidance

- For example, a tightly coupled HPC workload might take ten hours to run on one instance (X CPU cores), if the same job is run on 10 EC2 instances (10X CPU cores), it may take 2 hours

(performance scaling can be but is typically not linear). The cost for EC2 will be higher, however the results of the calculation will be available much quicker. This could reduce the research and development time, and for example reduce time to market.

- Verify that instances have sufficient physical memory to complete jobs but not more, as unused memory will not improve compute performance. Depending on the methodology, increasing the number of nodes per job may distribute the computational problem and reduce the required memory per node.
- Choose the pricing model best suited for workload duration and criticality, such as using On Demand for high priority workloads, spot for flexible HPC workloads, and RI for consistent HPC workloads to help optimize cost.
- Reducing the runtime can also reduce costs for surrounding services, such as storage, since these resources will not be needed for as long.
- The choice of storage can also impact cost. Many HPC applications read and write significant amounts of data. If the time to read and write data can be reduced, then the compute will be needed for less time. There are many different types and performance settings for storage. Picking the optimum version for your application can improve efficiency and reduce cost overall.
- For some applications, the cost of licenses exceeds the cost of AWS resources. It may be worth spending a little more on AWS resources to achieve better performance and save money overall.

Manage demand and supplying resources

There are no best practices unique to HPC for the Manage demand and supplying resources best practice area. Please review the corresponding section in the AWS Well-Architected Cost Optimization Pillar whitepaper.

Optimize over time

There are no best practices unique to HPC for the optimize over time best practice area. For more information, see the corresponding section in the AWS Well-Architected Cost Optimization Pillar whitepaper.

Key AWS services

- AWS ParallelCluster
- AWS Batch

- Amazon FSx for Lustre
- AWS Cost Explorer

Resources

- Benchmarking AWS and HPC Services Whitepaper
- Gromacs Price Performance Optimizations on AWS
- Metrics in Amazon CloudWatch

Sustainability

The sustainability pillar includes the ability to continually improve sustainability impacts by reducing energy consumption and increasing efficiency across all components of a workload by maximizing the benefits from the provisioned resources and minimizing the total resources required. This Pillar also provides design principles, questions, best practices, and prescriptive guidance to meet sustainability targets for your AWS workloads.

You can find general prescriptive guidance on sustainable implementation in the Sustainability Pillar - AWS Well-Architected Framework whitepaper.

Topics

Design principles

The following are sustainability design principles to frame your AWS workload:

- **Maximize utilization and reduce the downstream impact of your workloads:** HPC computing nodes are ephemeral in the cloud, you do not need to keep your computing nodes always up and running. Start the computing nodes only when you need to run a job and terminate the instances as soon as the job is completed to minimize the carbon emissions. Do not duplicate your data by copying the results back to your on-premises data center, unless there is a business need to store a local copy. Intermediate data should be removed when no longer needed

- **Anticipate and adopt new and more efficient hardware and software and use managed services:** HPC workloads aggregate computing power and storage to allow for fast processing and calculation to solve scientific, mathematical, and engineering challenges. Due to its scale, HPC is more energy-intensive than general purpose computing. Therefore, better use of resources, both hardware and software, coupled with shorter runtimes, means improving utilization and environmental sustainability.

Since the impact of HPC is so relevant to help you achieve your sustainability goals, it is important to promote a culture of constant monitoring and improvement for this workload. HPC clusters

on-premises usually have a life cycle that can spans years during which they are rarely updated. When running HPC in the cloud, leaders need to promote a culture of continuous innovation. This will help to improve performance, reduce cost and improve sustainability. To continually improve and streamline your HPC workloads, you can automate your cluster deployment using CI/CD pipelines to easily test and deploy potential performance improvements and limit errors caused by manual processes. Prefer AWS Managed Services such as AWS ParallelCluster or AWS Batch to automate the provisioning and de-provisioning of the computing nodes. Continually monitor the release of new instance types and take advantage of energy efficiency improvements, including those instance types designed to support specific workloads such as machine learning training and inference.

Best practices

The best practices described in the Sustainability Pillar - AWS Well-Architected Framework are valid also for this HPC Lens.

Region selection

- HPCSUS01-BP01 Select target AWS Regions that balance performance and resource availability with your sustainability goals
- HPCSUS01-BP02 Select Regions based on where your users are located

Software and architecture

- HPCSUS02-BP01 Use a VDI solution to reduce data movement

Hardware and services

- HPCSUS03-BP01 Continually monitor the release of new EC2 instance types

Process and culture

- HPCSUS04-BP01 Promote a culture of constant monitoring and performance improvement

Focus areas

- Region selection

- Alignment to demand
- Software and architecture
- Data management
- Hardware and services
- Process and culture

Region selection

HPCSUS01: Have you chosen an AWS Region that takes into consideration your sustainability KPI?

In SUS01-BP01, you choose a Region based on both business requirements and sustainability goals. For HPC workloads, you need to consider additional operational requirements, such as instance availability and performance needs.

HPCSUS01-BP01 Select target AWS Regions that balance performance and resource availability with your sustainability goals

The AWS Cloud is a constantly expanding network of Regions and points of presence (PoP), with a global network infrastructure linking them together. The choice of Region for your workload significantly affects its KPIs, including end user latency, cost, and carbon footprint. To effectively improve these KPIs, you should choose Regions for your workload based on both your business requirements and sustainability goals.

Following performance efficiency best practices, you have identified the right Amazon EC2 instances for your HPC workloads. Amazon EC2 provides the ability to deploy instances in multiple locations, so you need to find which Regions have your preferred instance type. Then, select the best Region following the other practices in this pillar.

Implementation guidance

To achieve your sustainability goals, choose Regions that are near Amazon renewable energy projects and where the grid has a published carbon intensity that is lower than other locations (or Regions). For more detail on choosing a Region based on your sustainability guidelines, see How to select a Region for your workload based on sustainability goals.

HPCSUS01-BP02 Select Regions based on where your users are located

Placing a workload closer to its users provides the lowest latency while decreasing data movement across the network and reducing environmental impact.

Implementation guidance

It may happen that the two best practices described above cannot be implemented simultaneously (for example, if the preferred instances are not available in the Region closest to the end-users). In this case, the HPC cluster administrators must find the right tradeoff between the business objectives and the sustainability objectives.

Key AWS services

- AWS Global Infrastructure

Resources

- How to select a Region for your workload based on sustainability goals.

Alignment to demand

There are no alignment to demand best practices specific to this lens. For more detail, see Alignment to demand.

Software and architecture

HPCSUS02: Have you complemented your architecture with a remote desktop solution?

By enabling remote visualization of simulation results, rather than transferring and storing large datasets locally, organizations can minimize network traffic, optimize resource usage, and improve overall sustainability of their HPC operations.

HPCSUS02-BP01 Use a VDI solution to reduce data movement

In SUS03-BP05 Use software patterns and architectures that best support data access and storage patterns, you understand how data is used within your workload, consumed by your users,

transferred, and stored. For HPC workloads, Virtual Desktop Infrastructure (VDI) technologies help you reduce network traffic between the end-users' clients rather than transferring the entire data set and duplicating storage on-premises. In addition, optimizing data movement across the network reduces the total networking resources required for the workload and lowers its environmental impact.

Implementation guidance

Use a remote visualization technology, such as Amazon DCV or Amazon AppStream 2.0, to visualize the results of your simulations without the need of copying back the results.

Key AWS services

- Amazon DCV
- Amazon AppStream 2.0
- Research and Engineering Studio on AWS

Resources

- Empowering Researchers to Run HPC Workloads on AWS with Research Gateway

Data management

There are no data management best practices specific to this lens. For more detail, see Data management.

Hardware and services

HPCSUS03: Have you selected the most efficient EC2 instance type for your HPC workload?

Evaluate and stay current with new Amazon EC2 instance types for HPC workloads to maximize efficiency and sustainability. As new instances are introduced, you can adopt them for your workload and become more efficient.

HPCSUS03-BP01 Continually monitor the release of new EC2 instance types

As stated in the performance efficiency Pillar, we recommend benchmarking your applications with different EC2 instances to identify which instance type can deliver the best performance for your HPC workloads. Using efficient Amazon EC2 instances in HPC workloads is crucial for lower resource usage and cost-effectiveness.

Implementation guidance

Continually monitor the release of new instance types and take advantage of energy efficiency improvements, including those instance types designed to support specific workloads such as machine learning training and inference. Independent software vendors (ISVs) continue to update their supported architectures. Keep an eye for ISVs that can support the ARM architecture or accelerators, such as GPUs. Updating your applications helps you improve the efficiency of your workload.

Key AWS services

- Amazon EC2 Instance types

Resources

- Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances
- Deep dive into AWS Graviton3 and Amazon EC2 C7g instances
- AWS Graviton4-based Amazon EC2 R8g instances: best price performance in Amazon EC2

Process and culture

HPCSUS04: What are your methods to rapidly introduce sustainability improvements?

With the energy-intensive nature of HPC workloads, it is important to constantly monitor and continually adopt new technologies to reduce your environmental impact. You can test and adopt new technologies quicker in the cloud when compared to a traditional on-premises environment.

HPCSUS04-BP01 Promote a culture of constant monitoring and performance improvement

HPC workloads aggregate computing power and storage to allow for fast processing and calculation to solve scientific, mathematical, and engineering challenges. As a result of scale, HPC workloads are often more energy-intensive than general purpose computing. Therefore, better use of resources (both hardware and software), coupled with shorter runtimes, can lead to improved utilization and environmental sustainability.

Implementation guidance

- Since the impact of HPC is relevant in achieving your sustainability goals, it's important to promote a culture of constant monitoring and improvement for this workload. On-premises HPC clusters usually have a life cycle that can spans years during which they are rarely updated. When running HPC in the cloud, leaders should promote a culture of continuous innovation. This helps improve performance, reduce cost, and improve sustainability.
- To continually improve and streamline your HPC workloads, you can automate your cluster deployment using CI/CD pipelines to easily test and deploy potential performance improvements and limit errors caused by manual processes. For more information, see Tutorial: Create a simple pipeline (S3 bucket).

Resources

- Delivering sustainable, high-performing architectures
- Baker Hughes Reduces Time to Results, Carbon Footprint, and Cost Using AWS HPC: Customer Success Stories

Conclusion

The AWS Well-Architected Framework provides architectural best practices across the six pillars for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. The Framework provides a set of questions that allows you to review an existing or proposed architecture. It also provides a set of AWS best practices for each pillar. Using the Framework in your architecture will help you produce stable and efficient systems, which allow you to focus on your functional requirements. This High Performance Computing Lens whitepaper provides architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems for High Performance Computing workloads in the cloud. We've provided and overview of prototypical HPC architectures and overarching HPC design principles. We've revisited the Well-Architected pillars through the lens of HPC, providing you with a set of questions to help you review an existing or proposed HPC architecture. Applying the AWS Well-Architected Framework to your architecture helps you build stable and efficient systems, allowing you to focus on running HPC applications and pushing the boundaries of your field.

Contributors

The following individuals and organizations contributed to this version of this whitepaper:

- Aaron Bucher, Principal HPC Solutions Architect, Amazon Web Services
- Art Sedighi, Senior Partner Solutions Architect, Amazon Web Services
- Brendan Sisson, Senior Manager Solutions Architecture, Amazon Web Services
- Francesco Ruffino, Principal HPC Solutions Architect, Amazon Web Services
- Marissa Powers, Senior HPC Solutions Architect, Amazon Web Services
- Max Starr, Senior HPC Solutions Architect, Amazon Web Services
- Nina Vogl, Principal HPC Solutions Architect, Amazon Web Services
- Ramin Torabi, Senior HPC Solutions Architect, Amazon Web Services
- Shun Utsui, Senior Industry Solutions Architect - Research, Amazon Web Services
- Soham Garg, HPC Specialist Solutions Architect, Amazon Web Services
- Bruce Ross, Well-Architected Lens Leader, Amazon Web Services
- Raymond Chow, Sr. WW PSA, HPC, Amazon Web Services
- Arvind Raghunathan, Principal Operations Lead SA, Amazon Web Services
- Mahmoud Matouk, Principal Security Lead SA, Amazon Web Services
- Ryan Dsouza, Principal Guidance Lead SA, Amazon Web Services
- Nataliya Godunok, Cloud Optimization Success SA, Amazon Web Services
- Haleh Najafzadeh, Sr. Manager, Cloud Optimization Success Guidance, Amazon Web Services
- Madhuri Srinivasan, Sr. Technical Writer, Amazon Web Services
- Stewart Matzek, Sr. Technical Writer, Amazon Web Services
- Matthew Wygant, Sr. TPM Guidance, Amazon Web Services

The following individuals and organizations contributed to earlier versions of this whitepaper:

- Aaron Bucher, Principal HPC Solutions Architect, Amazon Web Services
- Omar Shorbaji, Global Solutions Architect, Amazon Web Services
- Linda Hedges, HPC Application Engineer, Amazon Web Services
- Nina Vogl, Principal HPC Solutions Architect, Amazon Web Services

- Sean Smith, HPC Software Development Engineer, Amazon Web Services
- Kevin Jorissen, Solutions Architect Climate and Weather, Amazon Web Services
- Philip Fitzsimons, Sr. Manager Well-Architected, Amazon Web Services

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Major update	Major content updates across all pillars.	June 23, 2025
Minor update	Fixed broken links.	April 6, 2023
Minor update	Updated text to reflect the Sustainability Pillar.	October 3, 2022
Minor update	Updated links.	March 10, 2021
Whitepaper updated	Minor updates	December 19, 2019
Whitepaper updated	Minor updates	November 1, 2018
Initial publication	High Performance Computing Lens first published.	November 1, 2017

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.