AWS 白皮书

最佳实践设计模式：优化 Amazon S3 性能

最佳实践设计模式：优化 Amazon S3 性能: AWS 白皮书

Table of Contents

最佳实践设计模式：优化 Amazon S3 性能

初次发布日期：2019 年 6 月（文档修订）

摘要

当构建用于从 Amazon S3 上载和检索对象的应用程序时，请遵循 AWS 最佳实践准则以优化性能。AWS 还提供了更详细的性能设计模式。

简介

当从 Amazon S3 上载和检索存储时，您的应用程序可以轻松地实现每秒数千个事务的请求性能。Amazon S3 自动扩展到高请求速率。例如，您的应用程序可以在存储桶中实现至少每秒每个前缀 3,500 个 PUT/COPY/POST/DELETE 请求和 5,500 个 GET/HEAD 请求。对存储桶中的前缀数量没有限制。您可以通过并行读取来增加读取或写入性能。例如，如果您在 Amazon S3 存储桶中创建 10 个前缀以并行处理读取，则可以将读取性能扩展到每秒 55,000 个读取请求。

Amazon S3 上的某些数据湖应用程序对于运行超过 PB 级数据的查询扫描数百万或数十亿个对象。这些数据湖应用程序实现的单一实例传输速率可最大限度地提高 Amazon EC2 实例的网络接口利用率，这在单一实例上可高达 100 Gb/s。然后，这些应用程序跨多个实例聚合吞吐量，以获得每秒多个 Tb 的级别。

另外一些应用程序对延迟很敏感，例如社交媒体消息传递应用程序。这些应用程序可实现一致的小对象延迟（对于较大的对象，为第一个字节输出延迟），延迟时间大约为 100 - 200 毫秒。

其他 AWS 服务也可帮助加快不同应用程序架构的性能。例如，如果您希望通过单一 HTTP 连接实现较高的传输速率，或需要单一位毫秒延迟，请使用 Amazon CloudFront 或 Amazon ElastiCache 以通过 Amazon S3 进行缓存。

此外，如果您希望在客户端与 S3 存储桶之间获得较快的长距离数据传输速度，请使用 Amazon S3 Transfer Acceleration。Transfer Acceleration 使用 CloudFront 中的全球分布式边缘站点来加快跨地理距离的数据传输。

如果 Amazon S3 工作负载将服务器端加密与 AWS Key Management Service (SSE-KMS) 结合使用，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS 限制，以获取有关使用案例支持的请求速率的信息。

下面的主题介绍的最佳实践准则和设计模式用于优化使用 Amazon S3 的应用程序的性能。

本指南的优先级高于之前有关优化 Amazon S3 的性能的任何指南。例如，以前的 Amazon S3 性能指南建议用哈希字符来随机化前缀命名，以便优化频繁数据检索的性能。现在，您不再需要为了提高性能随机化前缀命名，而是可以对前缀使用基于顺序日期的命名方式。有关 Amazon S3 性能优化的最新信息，请参阅性能指南和性能设计模式。

Amazon S3 的性能准则

要在 Amazon S3 上获得应用程序的最佳性能，AWS 建议遵循以下准则。

主题

衡量性能

当优化性能时，请查看网络吞吐量、CPU 和动态随机存取内存 (DRAM) 要求。根据针对这些不同资源的组合需求，可能需要评估不同的 Amazon EC2 实例类型。有关实例类型的信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的实例类型。

它还有助于在衡量性能时使用 HTTP 分析工具查看 DNS 查找时间、延迟和数据传输速度。

横向扩展存储连接

跨许多连接分布请求是一种常用的横向扩展性能的设计模式。当构建高性能应用程序时，将 Amazon S3 视为非常大的分布式系统，而不是类似于传统存储服务器的单个网络终端节点。您可以通过将多个并行请求发到 Amazon S3 来实现最佳性能。将这些请求分布在不同的连接，以最大限度地利用 Amazon S3 的可访问带宽。Amazon S3 对与存储桶建立的连接数没有任何限制。

使用字节范围提取

通过在 GET 对象请求中使用范围 HTTP 标头，您可以从对象中提取字节范围，只传输指定的部分。您可以使用到 Amazon S3 的并行连接，从相同对象中提取不同的字节范围。这有助于您通过单一整个对象请求实现更高的聚合吞吐量。通过提取较小范围的大型对象，您的应用程序还可以在请求中断时改善重试次数。有关更多信息，请参阅获取对象。

字节范围请求的典型大小为 8 MB 或 16 MB。如果使用分段上传来 PUT 对象，则最佳实践是以相同的分段大小来 GET 它们（或者至少与分段边界相符）以获得最佳性能。GET 请求可以直接面向单独的分段；例如 `GET ?partNumber=N`。

延迟敏感型应用程序的重试请求

大量的超时和重试会导致持续不断的延迟。考虑到 Amazon S3 的大规模，如果第一个请求较慢，则退出的请求可能采取不同的路径并快速成功。AWS 开发工具包具有可配置的超时和重试值，您可以进行调整以符合特定应用程序的容限。

在同一 AWS 区域中结合 Amazon S3（存储）和 Amazon EC2（计算）

尽管 S3 存储桶名称全局唯一，但每个存储桶存储在您创建它时所选择的区域中。为了优化性能，我们建议您尽可能从位于同一个 AWS 区域的 Amazon EC2 实例中访问此存储桶。这可以减少网络延迟和数据传输成本。

有关数据传输成本的更多信息，请参阅 Amazon S3 定价。

使用 Amazon S3 Transfer Acceleration 最大限度地减少因距离导致的延迟

Amazon S3 Transfer Acceleration 可在客户端与 S3 存储桶之间管理快速、轻松且安全的长地理距离文件传输。Transfer Acceleration 利用 Amazon CloudFront 中的全球分布式边缘站点。当数据到达某个边缘站点时，会通过经过优化的网络路径路由至 Amazon S3。Transfer Acceleration 适合定期跨大洲传输 GB 到 TB 的数据。它也适用于全球各地需要上载到集中式存储桶的客户。

您可以使用 Amazon S3 Transfer Acceleration 速度比较工具来比较各个 Amazon S3 区域内加快的上载速度和未加快的上载速度。此速度比较工具使用分段上传来将文件从浏览器传输到各种使用和未使用 Amazon S3 Transfer Acceleration 的 Amazon S3 区域。

使用最新版本的 AWS 开发工具包

AWS 开发工具包为许多用于优化 Amazon S3 性能的建议准则提供内在的支持。这些开发工具包提供了更简单的 API 以便从应用程序内部利用 Amazon S3，并定期更新以遵循最新的最佳实践。例如，这些开发工具包包含自动对 HTTP 503 错误重试请求的逻辑，并投资编写代码来应对和适应慢速连接。

这些开发工具包还提供传输管理器，以便在适当的时机使用字节范围请求来自动横向扩展连接，从而实现每秒数千个请求。务必使用最新版本的 AWS 开发工具包，以获取最新的性能优化功能。

您还可以在使用 HTTP REST API 请求时优化性能。当使用 REST API 时，您应遵循属于开发工具包一部分的相同最佳实践。对于慢速连接允许超时和重试，并使用多个连接以允许并行提取对象数据。有关使用 REST API 的信息，请参阅 Amazon Simple Storage Service API 参考。

Amazon S3 的性能设计模式

当设计应用程序以从 Amazon S3 上载和检索存储时，请使用我们的最佳实践设计模式，以实现应用程序的最佳性能。我们还提供性能指南，供您在规划应用程序架构时考虑。

要优化性能，您可以使用以下设计模式。

主题

对频繁访问的内容使用缓存

许多在 Amazon S3 中存储数据的应用程序会提供用户反复请求的数据的一个“工作集”。如果工作负载针对一组常用对象发送重复的 GET 请求，则可以使用 Amazon CloudFront、Amazon ElastiCache 或 AWS Elemental MediaStore 等缓存来优化性能。成功地采用缓存可以实现低延迟和较高的数据传输速率。使用缓存的应用程序还会向 Amazon S3 发送少量的直接请求，这可帮助降低请求成本。

Amazon CloudFront 是一个快速内容交付网络 (CDN)，它透明地在一大组地理位置分散的节点 (PoP) 中缓存 Amazon S3 中的数据。当可能从多个区域或通过 Internet 访问对象时，CloudFront 允许在访问对象的用户附近缓存数据。这样就可以实现常见 Amazon S3 内容的高性能交付。有关 CloudFront 的更多信息，请参阅 Amazon CloudFront 开发人员指南。

Amazon ElastiCache 是一个托管的内存中缓存。通过 ElastiCache，您可以预配置在内存中缓存对象的 Amazon EC2 实例。这种缓存会令 GET 延迟下降若干个数量级，并显著增加下载吞吐量。要使用 ElastiCache，请修改应用程序逻辑，以便用热门对象填充缓存和检查缓存中的热门对象，然后从 Amazon S3 中请求它们。有关使用 ElastiCache 提高 Amazon S3 GET 性能的示例，请参阅博客文章使用 Amazon ElastiCache for Redis 增强 Amazon S3。

AWS Elemental MediaStore 是一个专为 Amazon S3 中的视频工作流和媒体交付构建的缓存和内容分发系统。MediaStore 提供专用于视频的端到端存储 API，推荐用于性能敏感型视频工作负载。有关 MediaStore 的信息，请参阅 AWS Elemental MediaStore 用户指南。

延迟敏感型应用程序的超时和重试

在某些情况下，应用程序会收到来自 Amazon S3 的响应，指示需要重试。Amazon S3 将存储桶和对象名称映射到与其关联的对象数据。如果应用程序生成高的请求率（通常针对少量对象持续超过每秒 5,000 个请求的速率），则它可能会收到 HTTP 503 速度下降 响应。如果出现这些错误，每个 AWS 开发工具包都会使用指数退避来实现自动重试逻辑。如果您不使用 AWS 开发工具包，则应在收到 HTTP 503 错误时实施重试逻辑。有关退避技术的信息，请参阅《Amazon Web Services 一般参考》中的 AWS 中的错误重试和指数退避。

Amazon S3 自动扩展以应对持续的新请求速率，同时动态地优化性能。尽管 Amazon S3 在内部针对新的请求速率进行优化，但您将临时收到 HTTP 503 请求响应，直至优化完成。当 Amazon S3 在内部针对新的请求速率优化性能后，通常无需重试就能正常应对所有请求。

对于延迟密集型应用程序，Amazon S3 建议跟踪和主动重试较慢的操作。当重试请求时，我们建议您使用到 Amazon S3 的新连接并执行全新 DNS 查找。

当您发出大型可变大小的请求（例如，超过 128 MB）时，我们建议跟踪所实现的吞吐量并重试最慢 5% 的请求。当您发出较小的请求（例如，小于 512 KB）时，其中，延迟时间中值通常处于数十毫秒的范围内，好的指导原则是在 2 秒后重试 GET 或 PUT 操作。如果需要更多重试，则最佳实践是退避。例如，我们建议您在 2 秒后发出一次重试，然后 4 秒后再发出一次重试。

如果应用程序对 Amazon S3 发出固定大小的请求，则应预期对于其中每个请求获得更一致的响应时间。在这种情况下，一个简单的策略是确定最慢的 1% 的请求并重试这些请求。甚至一次重试也会频繁、有效地减少延迟。

如果您使用 AWS Key Management Service (AWS KMS) 进行服务器端加密，请参阅《AWS Key Management Service 开发人员指南》中的配额，以获取有关使用案例支持的请求速率的信息。

横向扩展和请求并行化以实现高吞吐量

Amazon S3 是一个非常大的分布式系统。为了帮助您利用其规模，我们建议您将并行请求横向扩展到 Amazon S3 服务终端节点。除了在 Amazon S3 中分布请求之外，这种类型的扩展方法还有助于将负载分布到整个网络中的多个路径。

对于高吞吐量传输，Amazon S3 建议使用并行利用多个连接来 GET 或 PUT 数据的应用程序。例如，AWS Java 开发工具包中的 Amazon S3 Transfer Manager 支持上述功能，大多数其他 AWS 开发工具包提供类似的构造。对于某些应用程序，您可以实现并行连接，具体方法为：在不同的应用程序线程或不同的应用程序实例中并行启动多个连接。要采取的最佳方法取决于应用程序以及您访问的对象的结构。

您可以使用 AWS 开发工具包直接发出 GET 和 PUT 请求，而不是利用 AWS 开发工具包中的传输管理。这种方法让您可以更直接地调整工作负载，同时仍受益于软件工具包支持重试和处理任何可能发生的 HTTP 503 响应。通常来说，当您将某个区域中的大型对象从 Amazon S3 下载到 Amazon EC2 时，我们建议您以 8-16 MB 的粒度向对象的字节范围发出并行请求。对于每个 85-90 MB/s 的所需网络吞吐量发出一个并行请求。要使 10 Gb/s 网络接口卡 (NIC) 达到饱和，您可以通过单独的连接使用大约 15 个并行请求。您可以通过更多连接扩展并行请求以使速度更快的 NIC 达到饱和，如 25 Gb/s 或 100 Gb/s NIC。

当您调整并行发出的请求数量时，衡量性能至关重要。我们建议一次只用一个请求为起点。衡量所实现的网络带宽以及应用程序用于处理数据的其他资源的使用情况。然后，您可以确定瓶颈资源（也即使用率最高的资源），进而确定可能有用的请求数。例如，如果一次处理一个请求会导致 CPU 利用率达到 25%，则建议可以容纳多达四个并行请求。

衡量是必不可少的，随着请求速率增加，务必确认资源使用情况。

如果应用程序使用 REST API 直接向 Amazon S3 发出请求，我们建议您使用 HTTP 连接池，并对一系列请求重用每个连接。通过避免逐个请求连接设置，可消除对每个请求执行 TCP 慢速启动和安全套接字层 (SSL) 握手的需要。有关使用 REST API 的信息，请参阅 Amazon S3 REST API 简介。

最后，务必注意 DNS 并仔细检查请求正在分散到广泛的 Amazon S3 IP 地址池。对于 Amazon S3 的 DNS 查询会循环经过一个很大的 IP 终端节点列表。但是，缓存解析程序或重用单一 IP 地址的应用程序代码不会受益于地址多样性和由此产生的负载均衡。网络实用工具（如 `netstat` 命令行工具）可以显示用来与 Amazon S3 通信的 IP 地址，并且我们提供要使用的 DNS 配置的准则。有关这些准则的更多信息，请参阅请求路由。

使用 Amazon S3 Transfer Acceleration 加快地理位置分散的数据传输

Amazon S3 Transfer Acceleration 通过使用 Amazon S3，有效地将由于全球分散的客户端与区域性应用程序之间的地理距离所导致的延迟降至最小甚至消除。Transfer Acceleration 使用 CloudFront 中的全球分布式边缘站点进行数据传输。AWS 边缘网络在超过 50 个站点具有接入点。目前，它用于通过 CloudFront 分布内容，并向针对 Amazon Route 53 发出的 DNS 查询提供快速响应。

边缘网络还有助于加速进出 Amazon S3 的数据传输。它对具有以下特点的应用程序是理想之选：跨大洲或在大洲之间传输数据、具有快速 Internet 连接、使用大型对象，或具有大量要上载的内容。当数据到达某个边缘站点时，数据会被经过优化的网络路径路由至 Amazon S3。通常，您与 Amazon S3 区域之间的距离越远，使用 Transfer Acceleration 可预期获得的速度改进就越大。

您可以对新的或现有的存储桶设置 Transfer Acceleration。您可以通过单独的 Amazon S3 Transfer Acceleration 终端节点来使用 AWS 边缘站点。测试 Transfer Acceleration 是否有助于提高客户端请求性能的最佳方法是使用 Amazon S3 Transfer Acceleration 速度比较工具。网络配置和条件随时间或站点不同而可能发生变化。因此，我们只对 Amazon S3 Transfer Acceleration 可能会改进上载性能的传输向您收费。有关将 Transfer Acceleration 与不同的 AWS 开发工具包结合使用的信息，请参阅 Amazon S3 Transfer Acceleration 示例。

贡献者

本文档的贡献者包括：

- Mai-Lan Tomsen Bukovec，Amazon S3 副总裁
- Andy Warfield，Amazon S3 高级首席工程师
- Tim Harris，Amazon S3 首席工程师

文档修订

要获得有关此白皮书的更新通知，请订阅 RSS 馈送。

更新-历史记录-更改	更新-历史记录-描述	更新-历史记录-日期
已更新	已进行技术准确性审核	2021 年 3 月 10 日
初次发布	初次发布	2019 年 6 月 1 日

声明

客户负责对本文档中的信息进行独立评估判断。本文档：(a) 仅供参考；(b) 代表当前提供的 AWS 产品和实践，如有更改，恕不另行通知；并且 (c) AWS 及其附属机构、供应商或许可方不做任何承诺或保证。AWS 产品或服务“按原样”提供，不提供任何形式的保证、陈述或条件，无论是明示还是暗示。AWS 对其客户的责任和义务由 AWS 协议决定，本文档与 AWS 和客户之间签订的任何协议无关，亦不影响任何此类协议。