性能效率支柱

AWS 良好架构框架

2016 年 11 月

通告

本文档所提供的信息仅供参考，且仅代表截至本文件发布之日时 AWS 的当前产品与实践情况，若有变更恕不另行通知。客户有责任利用自身信息独立评估本文档中的内容以及任何对 AWS 产品或服务的使用方式，任何“原文”内容不作为任何形式的担保、声明、合同承诺、条件或者来自 AWS 及其附属公司或供应商的授权保证。AWS 面向客户所履行之责任或者保障遵循 AWS 协议内容，本文件与此类责任或保障无关，亦不影响 AWS 与客户之间签订的任何协议内容。

目录

摘要

本篇白皮书主要着眼于 Amazon Web Services（简称 AWS）良好架构框架当中的性能效率支柱。其中提供的指导意见旨在帮助客户在 AWS 环境的设计、交付与维护工作当中遵循相关最佳实践。

内容简介

在 Amazon Web Serivces，我们意识到为客户提供架构最佳实践指导并借此帮助其立足云环境设计并运营起可靠、安全、高效且具备成本效益的系统方案的重要意义。作为这项工作中的重要组成部分，我们开发 AWS 良好架构框架，希望帮助大家了解您在 AWS 之上进行系统构建时所作出各项决策的优势与弊端。我们相信，良好架构系统能够显著提升您实现业务成功的机率。

这套框架共基于以下五大支柱：

- 安全性
- 可靠性
- 性能效率
- 成本优化
- 卓越操作

本份白皮书将重点介绍性能效率支柱以及如何将其纳入您的解决方案。从传统的内部环境角度来看，实现高水平且可持续性能水平一直极具挑战。通过采用本份白皮书中提到的指导方针，您将能够建立起拥有高效性能表现且可随时间推移维持性能水平的理想架构体系。

本份白皮书主要适用于各类技术性职能角色，包括首席技术官（简称 CTO）、架构师、开发者以及运营团队成员。在阅读本份白皮书之后，您将了解到 AWS 各最佳实践及策略，并可将其实际应用于云架构设计工作当中。本份白皮书并不提供关于架构模式或者实现途径的具体细节，但我们将在文末列出与此类议题相关的参考资源。

性能效率

性能效率支柱专注于高效利用计算资源以满足各类需求，同时随着需求变更及技术演进而不断保持这种效率水平。本份白皮书为 AWS 环境下的性能效率架构设计工作提供深入的最佳实践类指导。

设计原则

在云环境中，以下设计原则能够帮助您确切实现性能效率。

- **普及先进技术:** 原本难于实现的技术方案如今可凭借着云供应商所拥有的深厚知识积累与复杂性解决能力得以轻松运用。相较于由内部 IT 团队学习如何托管并运营新型技术成果，云服务供应商能够轻松将其转化为“即服务”形式。举例来说，NoSQL 数据库、媒体转码以及机器学习皆属于需要专业技术能力，且在人才市场上较为稀缺的应用方向。在云环境中，这些技术能够以“即服务”形式供您的团队直接消费，确保其专注于产品开发而非系统资源的配置与管理。
- **数分钟内实现全球化:** 您只需要数次点击，即可在全球范围内多个区域内轻松实现系统部署。如此一来，您将能够为客户提供更低延迟以及更理想的使用体验，且继续保持最低运营成本。
- **使用无服务器架构:** 在云环境当中，无服务器架构能够帮助您在承载传统计算活动的同时，不再需要运行或者维护任何服务器系统。举例来说，存储设备能够作为静态网站存在，您将不再需要 Web 服务器，而事件服务则可帮助您托管业务代码。这种对运营负担的有力精简让您告别服务器管理工作，并通过以云规模运营的托管服务降低交易成本。
- **提升实验频率:** 利用虚拟与自动化资源，您能够利用各种不同实例、存储或者配置类型快速完成比较测试。
- **机器同理心:** 切实使用能够与您欲达成之目标相匹配的技术方案。例如在选择数据库或者存储方案时，考虑具体数据访问模式。

定义

云环境下的**性能效率**分为四项最佳实践:

- 选择
- 审查
- 监控
- 权衡

采取一套数据驱动型方案以选择高性能架构选项。面向架构中的各个方面收集数据，从概要设计到服务资源类型的选择及配置皆在其中。通过周期性对您的选择进行复查，您将能够确保现有架构充分发挥 AWS 平台持续演进所带来的显著优势。监控将确保您可随时发现意料之外的性能问题并采取针对性措施。最后，您

的架构亦应作出权衡以进一步改善性能水平，例如采用压缩、缓存或者放松一致性要求等举措。

选择

特定系统当中的最佳解决方案将根据您工作负载类型的不同而有所区别，且一般需要将多种方案加以结合。良好架构系统采用多种解决方案，并且实现多种功能以改善性能水平。

在 AWS 当中，资源以虚拟化形式存在，且通过多种不同类型及配置方案进行交付。这意味着客户能够更轻松地找到契合自身需求的方案，亦可建立起更多在内部环境下往往很难实现的选项。举例来说，Amazon DynamoDB 等托管服务能够提供一套全面托管的 NoSQL 数据库，且保证其在任意运行规模之下皆仅带来个位数毫秒延迟。

当您为自己的架构选定模式及实现方式，并采用数据驱动型方案以达成最优效果时，AWS 解决方案架构师、AWS 参考架构以及 AWS 合作伙伴皆可根据我们多年来积累起的丰富经验帮助您作出明智的架构选择，同时通过基准测试或者压力测试提取数据信息以优化这套架构。在您确定自己的架构方案之后，则应用采取一套数据驱动的流程以持续调整您的资源类型与配置选项选择。您可利用基准测试以及压力测试获取此类指导性数据。欲了解更多信息，请参阅本份白皮书中的“审查”章节。

您的架构将可能引入其它多种不同架构方案（例如事件驱动型、ETL 或者管道）。您的架构实现方案将使用各项专门针对架构性能作出优化的 AWS 服务。在以下章节当中，我们将着眼于您应当认真考量的四大主要资源类型（即计算、存储、数据库以及网络）。

计算

针对特定系统的优化计算解决方案往往取决于应用程序设计、使用模式以及配置设置等几大因素。架构可能利用多种不同计算解决方案以支持其中的不同组件，同时启用多种功能以实现性能提升。如果在架构当中选择错误的计算组合方案，则有可能导致性能效率表现低下。

在为计算资源进行架构设计时，大家应充分发挥其中的弹性机制优势，从而确保能够在需求变化的同时始终保有与之相适应的系统资源。在 AWS 当中，计算资源以三种形式进行交付：实例、容器与函数。您应在架构当中为各类组件主动选择与之匹配的计算资源形式。其中实例属于默认选项；利用容器机制能够提升实例利用率；而函数则更适合事件驱动型或者高并发类任务。

实例

实例属于虚拟化服务器，因此您可以通过一键式方式或者 API 调用对其容量作出调整。由于云资源决策不再长期固定，因此您可以以实验性方式测试不同服务器类型。

在 AWS 计算服务 Amazon 弹性计算云（Amazon Elastic Compute Cloud ，简称 EC2）

当中，这些虚拟服务器**实例**拥有不同的家族与大小，且提供多种多样的特性选项

——包括固态硬盘（SSD）以及图形处理单元（GPU）。在启动 EC2 实例时，您所指定的实例类型决定了您可以使用的主机计算设备硬件配置。各种实例类型提供不同的计算、内存与存储容量。而各种实例类型也正是根据这些容量水平被归类为实例家族。

在选择实例家族与类型时，亦应当考虑到您工作负载所需要的配置选项：

- **图形处理单元**(简称 GPU)。利用图形处理单元上的通用计算（简称 GPGPU）计算能力，您能够通过在开发流程中纳入 CUDA 等平台确保自己的应用程序充分发挥 GPU 所提供的高并发处理能力。另外，如果您的应用程序需要 3D 渲染或者视频解压，GPU 亦可实现硬件加速计算与编码能力，帮助您的应用达到更为高效的运行效果。

- **突发型实例家族**。**突发型实例**的设计目标在于提供适度的基线性能与爆发式性能，确保您的工作负载在需要时能够获得理想的性能提升。此类实例适用于多数情况下不需要完整 CPU 工作负载，但偶尔需要爆发性性能的场景。其非常适用各类常规工作负载，例如 Web 服务器、开发者环境以及小型数据库。此类实例可在实例需要时以积分方式兑换额外性能。而在实例不再需要这些性能时，积分则持续进行积累。

- Amazon EC2 允许您访问**高级计算功能**，具体包括管理 C-state 与 P-state 寄存器并控制处理器的智能超额能力。您亦可访问协处理器以处理 AES-NI 加密运算，或者经由 AVX 扩展完成其它高级计算操作。

您应当利用需求数据为自己的工作负载选择最优 EC2 实例类型，从而确保匹配正确的网络与存储选项，同时考虑利用操作系统设置为工作负载带来更佳性能水平。

容器

容器属于一种操作系统虚拟化方法，允许大家立足资源隔离型进程运行应用程序及其全部依赖程序。

Amazon EC2 容器服务（Amazon EC2 Container Service ，简称 ECS）允许您以自动化方式基于 Amazon EC2 实例集群执行并管理大量容器。您可以在 Amazon ECS 当中创建由容器负责支持的应用服务。通过使用应用程序 Auto Scaling，您能够为服务定义基于指标的自动化规模伸缩能力，从而在服务需求增长时提升容器数量以满足具体需要。

Amazon ECS 还可采用 Auto Scaling 组，其允许大家通过添加 EC2 实例数量的方式实现 Amazon EC2 集群扩展。如此一来，您将能够确保底层服务器容量始终与所托管容器的资源需要相契合。

Amazon ECS 可与弹性负载均衡（Elastic Load Balancing ，简称 ELB）相集成，从而立足容器群组对您的服务进行负载均衡。通过创建应用程序负载均衡器实例，您的容器将能够自动在规模伸缩时自动面向该负载均衡器进行注册。凭借这些功能，大家可以在将多个相同服务容器服务共同托管在同一 Amazon EC2 实例，从而提升应用程序的可扩展能力。

在使用容器时，大家应当利用需求数据选择最适合自身工作负载的选项类型——同理于利用此类数据选择最优 EC2 实例类型。您亦应考虑容器各配置选项的具体设置，包括内存、CPU 以及租户配置等。

函数

函数可从希望执行的代码当中将执行环境抽象出来。利用函数，您将能够在无需运行或者管理任何实例的前提下执行代码或者提供服务。

AWS Lambda 允许您无需配置或者管理服务器及容器即可完成代码运行。您只需要进行代码上传，AWS Lambda 就能够帮助您完成一切与代码运行及扩展相关的任务。您可以设置代码以自动经由其它 AWS 服务进行触发；您可直接进行调用；或者配合 Amazon API 网关共同使用。

Amazon API 网关是一项全面托管服务，能够帮助开发人员更轻松地以任意规模进行 API 创建、发布、维护、监控以及安全保护。您可以创建 API 并将其作为您 AWS Lambda 函数的“前门”。API 网关负责处理各类涉及接收请求并处理成千上万条并发 API 调用相关的任务，其中包括流量管理、身份与访问控制、监控以及 API 版本管理等等。

为了配合 AWS Lambda 实现最优性能表现，您应选择符合函数需求量的内存配额，而后为其分配合适的 CPU 容量及其它资源。举例来说，如果您的 Lambda 函数需要 128 MB 内存，则应为其分配 256 MB 内存以及两倍左右的 CPU 计算资源。您还应当控制每项函数可运行的时间长度（最大为 300 秒）。

弹性

弹性允许您将资源供应量与实际需求量匹配起来。实例、容器与函数皆可提供弹性机制，具体包括配合 Auto Scaling 或者作为服务自身的功能之一。

最优供需匹配应能够为特定系统提供最低成本，同时亦需要具备充足的额外供应以控制供应时间并应对个别资源故障。需求可为固定或是可变形式，且应利用指标及自动化机制确保管理工作不致产生过高成本。

在 AWS 当中，您可以利用多种不同方法实现供需匹配。AWS **良好架构框架成本优化支柱**白皮书描述了如何使用以下各项方案：：

- 基于需求的方案
- 基于缓冲的方案
- 基于时间的方案

关键性 AWS 服务

弹性计算解决方案当中的关键 AWS 服务为 Auto Scaling，因为您可以利用其将资源供应与您的实际需求加以匹配。实例、容器与函数皆可提供弹性机制，具体包括配合 Auto Scaling 或者作为服务自身的功能之一。

资源

请参阅以下资源以了解与计算相关之 AWS 最佳实践的细节信息。

说明文档

- 实例: 实例类型
- 容器: Amazon ECS 容器实例
- 函数: 计算要求 – Lambda 函数配置

存储

特定系统的最优存储解决方案取决于访问方式（块、文件或者对象）、访问模式（随机或者连续）、数据吞吐量要求、访问频率（在线、离线及归档）、更新频率（WORM 与动态）以及可用性与持久性限制等因素。

在 AWS 当中，存储资源经过虚拟化且可通过多种不同形式加以交付。这意味着您将能够更为轻松地将存储方法与实际需求结合起来，同时实现本地基础设施当中往往难以实现的存储选项。举例来说，Amazon S3 在设计当中采用 11 个 9 持久性。您亦可选择将磁盘驱动器（简称 HDD）替换为固态驱动器（简称 SSD），或者在数秒钟之内轻松完成不同实例间的虚拟驱动器迁移。

特性

在选择存储解决方案时，您应当考虑您所需要的不同特性，具体包括可共享性、文件大小、缓存大小、延迟、数据吞吐能力以及数据持久性。此外，确保 AWS 服务所提供的能力确切满足您的实际需求：Amazon S3、Amazon 弹性块存储（简称 EBS）、Amazon 弹性文件系统（Amazon Elastic File System ，简称 EFS）或者 Amazon EC2 实例存储等。

存储性能可由数据吞吐量、每秒输入/输出操作（简称 IOPS）以及延迟衡量性能表现。了解这些指标之间的关系能够帮助您选择最合适的存储解决方案。

存储	服务	延迟	数据吞吐能力	可共享性
块	EBS、实例存储	最低，一致	单	挂载于单一实例之上，通过快照进行复制
文件系统	EFS	低，一致	多	多客户
对象	S3	低延迟	Web 规模	多客户

从访问延迟的角度来看，如果您的数据仅由单一实例进行访问，则您应使用块存储——例如 Amazon EBS 配合预配置 IOPS。Amazon EFS 等分布式文件系统通常能够在单一文件操作层面带来较低延迟，因此更适合应用于多实例访问需求。

除了降低延迟与提供数据吞吐能力之外，Amazon S3 还拥有其它功能。大家可以利用跨服务区复制（简称 CRR）为不同地理位置提供更低数据访问延迟。

从数据吞吐量的角度来看，Amazon EFS 能够支持各类高并发工作负载（例如多线程与多 Amazon EC2 实例中的并发操作），其能够实现较高吞吐量聚合与每秒操作。对于 Amazon EFS，您可以利用基准测试或者压力测试以选定合适的性能模式。

关键性 AWS 服务

存储层面的关键 AWS 服务为 Amazon S3，其可提供安全、持久且高度可扩展的云存储资源。以下服务与功能亦在其中发挥有重要作用：

- **Amazon EBS** 提供持久块存储分卷以供 EC2 实例使用。
- **Amazon EFS** 提供简单且可扩展的文件存储机制以供 EC2 实例使用。
- **Amazon EC2 实例存储**提供临时块级存储以供 EC2 实例使用。

资源

请参考以下资源以了解更多与存储 AWS 最佳实践相关的细节信息。

说明文档

- Amazon S3: 请求率与性能考量
- Amazon EFS: Amazon EFS 性能
- Amazon EBS: I/O 特性

视频

- Amazon EBS 高性能设计

配置选项

存储解决方案通常提供多种配置选项，允许您针对工作负载类型作出最优选择。

Amazon EBS 提供一系列选项以帮助您根据工作负载需求进行存储性能及成本优化。这些选项主要被划分为两大类：基于 SSD 的事务工作负载存储——例如数据库与引导分卷（性能主要取决于 IOPS）；以及基于 HDD 的吞吐量敏感型工作负载存储——例如 MapReduce 以及日志处理（性能主要取决于每秒 MB 传输能力）。

基于 SSD 的分卷中包括：面向延迟敏感型事务工作负载的，性能水平最高的 Provisioned IOPS SSD；以及适合处理多种事务数据的性价比均衡型通用 SSD。

Amazon S3 传输加速能够在客户端与 Amazon S3 存储桶之间距离较远时提供理想的文件传输速度。传输加速服务利用 Amazon CloudFront 的全球分布式边缘站点经由优化网络路径实现数据路由。对于包含大量 GET 请求的 Amazon S3 存储桶内工作负载，您应当将 Amazon S3 与 Amazon CloudFront 配合使用。在上传大型

文件时，您应当使用分片上传——即一次性上传多个部分以最大化利用网络吞吐能力。

访问模式

您对数据的具体访问方式会影响到存储解决方案的实际性能表现。选择最适合您访问模式的存储解决方案，或者请调整您的访问模式以适应存储解决方案自身特点，从而最大程度发挥其性能水平。

创建一套 RAID 0 阵列能够帮助您在文件系统当中获得超越单一分卷配置方案的性能表现。您可以考虑在 I/O 性能重要性高于容错性时使用 RAID 0。举例来说，您可以利用其配合拥有独立数据复制机制的高强度数据库场景。

Amazon EBS 基于 HDD 分卷当中包含面向频繁访问、吞吐量敏感型工作负载的数据吞吐量优化型 HDD，以及面向低访问频度数据且成本更低的冷门 HDD。

您可以利用本份白皮书当中“权衡”章节中的“**分区与分段**”部分技术方案对您的存储系统进行进一步优化。

关键性 AWS 服务

存储解决方案层面的关键 AWS 服务为 Amazon EBS、Amazon S3 以及 Amazon EFS——三者分别负责提供块、对象与文件系统存储解决方案。这些服务皆拥有多种配置选项，可帮助您对存储解决方案加以进一步优化。您各组件的访问模式亦应被纳入到存储解决方案的具体选择考量当中。

资源

请参阅以下资源以了解存储 AWS 最佳实践的相关细节信息。

说明文档

- 存储

数据库

针对特定系统的最优数据库解决方案取决于您的具体需求，包括可用性、一致性、分区容限、延迟、持久性、可扩展性以及查询能力等等。多数系统采用多种不同数据库解决方案以适应其中各子系统的实际需要，同时启用各类不同功能以进一步提升性能水平。为系统选择错误的数据库解决方案及功能可能导致性能效率低下。

尽管某一工作负载的配套数据库方案（例如 RDBMS、NoSQL 等等）会对具体性能效率产生巨大影响，但企业往往倾向于采用默认方案选项——而非使用数据驱动型方案。在存储方面，大家务必根据自身工作负载的访问模式进行考量，同时思考其它非数据库类解决方案是否能够更为高效地解决问题（例如使用对象存储 Amazon S3，搜索引擎或者数据仓库）。

特性

考虑您所需要的不同具体特性（例如可用性、一致性、分区容错性、延迟、持久性、可扩展性以及查询能力等），确保您能够选择最具性能优势的数据库方案选项（包括关键、NoSQL、数据仓库以及内存数据库）。

在 AWS 当中，我们针对各类需求提供多种服务方案，您应根据不同数据类型选择与之匹配的服务选项。

- **Amazon 关系数据库服务（Relational Database Service ，简称 RDS）** 提供一套全面托管关系数据库方案。适用于同 Amazon RDS 高度相关的信息处理场景。
- **Amazon DynamoDB** 提供一套全面托管 NoSQL 数据库方案，能够立足任意规模提供个位数毫秒延迟水平。Amazon DynamoDB 适用于用户个人资料等互联网规模信息的处理。
- **Amazon Redshift** 提供一套托管型 PB 级别数据仓库方案，允许您根据性能或者容量需求变更随时变更节点数量或者类型。如果您需要处理未来可能进行规模伸缩的 SQL 操作，则推荐使用 Amazon Redshift。
- **Amazon ElastiCache** 是一项 Web 服务，能够帮助您轻松实现内存内数据存储或者云端缓存体系的部署、操作与规模伸缩。如果您希望通过立足高速、托管的内存内数据存储体系提升 Web 应用程序性能水平，则推荐使用 Amazon ElastiCache。
- **Amazon CloudSearch** 是一项 AWS 云内的托管服务，能够为您的网站或者应用程序以简单易行且极具成本效益的方式实现搜索解决方案的设置、管理与规模伸缩。如果您希望以低延迟与高通量方式实现搜索或者报告，则推荐使用 Amazon CloudSearch。

配置选项

数据库解决方案通常提供多种配置选项，允许您针对具体工作负载类型实现优化。在选择具体数据库方案时，您应考量以下配置选项：存储优化、数据库级设置、内存以及缓存。

- **Amazon 关系数据库服务（ Relational Database Service，简称 RDS）** 提供负责实现读取性能扩展的读取副本（不适用于 Oracle 或者 SQL Server）、SSD 型存储选项、配置 IOPS 以及数据库级设置。
- **Amazon DynamoDB** 提供数据吞吐量与存储规模伸缩，亦提供辅助索引以帮助您更为高效地查询任何属性（列）。您也可以通过属性映射（将属性由表内复制至索引中）以进一步提升性能水平。
- **Amazon Redshift** 允许您变更数据仓库当中的节点数量或者类型，并可将压缩用户数据向上扩展至 PB 甚至更高级别。密集计算（简称 DC）节点允许您利用高速 CPU、大容量内存以及固态硬盘（SSD）建立起拥有超高性能的数据仓库体系。
- **Amazon ElastiCache** 配合 Memcached 可支持分段机制，从而利用多节点实现内存内缓存扩展。Amazon ElastiCache for Redis 当中包含集群部署、多分段构建单一内存内 TB 级别键-值存储以及每分段读取副本添加等功能，旨在提升数据访问性能。
- **Amazon CloudSearch** 能够面向各类搜索范畴提供强大的自动规模伸缩能力。随着您的数据或者查询总量的变化，Amazon CloudSearch 能够根据需要对您的搜索域资源进行扩展或者收缩。如果明确意识到您需要更多容量以处理批量上传或者即将迎来搜索流量激增，则可随时控制规模变更。

访问模式

数据的具体访问方式将对数据库解决方案的性能表现产生重大影响。选择最适合您访问模式的数据库解决方案，或者请调整您的访问模式以适应数据库解决方案自身特点，从而最大程度发挥其性能水平。

根据您的访问模式（例如索引、键值分布、分区或者横向扩展）对数据库系统的使用方式进行优化。

- **Amazon 关系数据库服务（ Relational Database Service，简称 RDS）**能够确保您始终拥有面向各关键性搜索任务的索引，且支持物化视图。您亦可添加副本读取索引以进一步提升查询性能。
- **Amazon DynamoDB** 以自动化方式为您管理表分区，可在必要时添加新的分区并在各分区间分布配置数据吞吐容量。为了确保您的负载可跨各分区进行平均分布，您应在表当中跨越多个条目进行数据访问统一。
- **Amazon Redshift** 能够在配合指定排序键、分布键以及列编码的情况下提供最佳性能，这一切都能够显著提升实际存储、IO 与查询性能水平。

- **Amazon ElastiCache** 配合 Memcache 引擎将在您跨越多个节点进行缓存键分布时，高效利用多个 ElastiCache 节点。您应配置各客户端以使用一致散列机制，从而确保在进行节点添加或者移除时不致因键移动引发大量缓存未命中问题。

- **Amazon CloudSearch** 请求往往体现为资源密集型进程，这可能对您的搜索域造成性能以及运行成本影响。您应认真调整搜索请求以帮助降低此类进程的日常开支。

为了进一步优化您对数据库系统的使用方式，请参阅本份白皮书当中“权衡”章节当中的“**分区或分段**”部分内容。

关键性 AWS 服务

数据库解决方案层面的关键 AWS 服务为 Amazon RDS、Amazon DynamoDB 以及 Amazon RedShift——其分别负责提供关系、NoSQL 与数据库仓库解决方案。这些服务皆拥有大量配置选项，允许您更进一步对存储解决方案进行优化。您各组件的访问模式亦应被纳入对存储解决方案的实际选择当中。

资源

主参阅以下资源以了解更多与数据库 AWS 最佳实践相关的细节信息。

说明文档

- 云数据库与 AWS

网络

针对特定系统的最优网络解决方案往往取决于延迟、数据吞吐量要求等因素。用户或者内部资源等物理限制条件往往决定着位置的选项，而您可利用边缘节点技术或者资源安置方案解决此类问题。

在 AWS 当中，网络资源以虚拟化形式存在，且可采用多种不同类型及配置选项。这意味着您能够更轻松地确保网络方案与自身需求相匹配。AWS 提供多种产品功能（例如大量网络实例类型，Amazon EBS 优化实例、Amazon S3 传输加速以及动态 Amazon CloudFront 等）以优化网络流量。AWS 亦提供多种网络功能（例如 Amazon Route53 基于延迟路由、Amazon VPC 端点以及 AWS Direct Connect）以减少网络距离或者卡顿现象。

位置

AWS 云基础设施围绕服务区与可用区两大基本概念构建而成。服务区为世界范围内的一个物理位置，其中包含多个可用区。可用区则由一座或者多座离散数据中心所组成，其中各数据库皆拥有冗余电源、网络与连接并部署在独立的设施当中。这些可用区将为您提供远超单一数据中心的生产应用程序运营能力以及更出色的数据库可用性、容错性与可扩展性。

根据以下关键性要素，您可为部署方案选择最适合的服务区：

- **用户所在位置**：选择尽可能接近应用程序用户所在位置的服务区，从而确保带来更低应用程序使用延迟。
- **数据所在位置**：对于数据量较大的应用程序，主要延迟瓶颈在于数据被传输至应用程序内计算部分的过程当中。应用程序代码应尽可能在接近数据的位置执行。
- **其它约束因素**：请将安全性与合规性等其它约束因素纳入考量。

安置组

Amazon EC2 在网络层面提供安置组机制。一个安置组相当于单一可用区之内的一个实例逻辑分组。利用安置组并配合受支持实例类型，即可确保应用程序始终采用低延迟水平的 10 Gb 每秒（Gbps）网络资源。我们建议您为需要配合低网络延迟及/或高网络通量的应用程序选择安置组机制。采用安置组能够有效缓解网络通信当中的卡顿问题。

边缘位置

我们立足于全球**边缘位置**网络交付延迟敏感型服务。这些边缘位置通常提供内容分发网络（简称 CDN）以及域名解析（简称 DNS）等服务。通过立足边缘位置使用这些服务，其可实现低延迟请求响应以进行内容分发或者 DNS 解析。这些服务亦可提供内容地理定位（即根据最终用户的实际位置交付不同内容）等地理服务，或者基于延迟将最终用户定向至最近区域（以实现最低延迟水平）。

Amazon CloudFront 属于一套全球性 CDN，其可用于对图像、脚本以及视频等静态内容以及 API 或 Web 应用程序等动态内容进行加速。其起效需要依赖于全球边缘位置网络——后者负责执行内容缓存并为用户提供高性能网络连接能力。Amzon CloudFront 亦可对其它多种功能进行加速——具体包括内容上传与动态应用程序，这使得一切立足互联网进行交流供应的应用程序皆获得性能提升。

Amazon Route 53 是一项具备高可用性与可扩展性的云 DNS Web 服务。其设计目标在于为开发人员及企业提供极为可靠且具有成本效益的路由方案，可将名称（例如 www.example.com）翻译为数字 IP 地址（例如 192.0.2.1）以实现计算机间的彼此连接，最终引导最终用户使用互联网应用。Amazon Route 53 亦全面兼容 IPv6。

您应利用边缘服务以降低延迟水平并启用内容缓存。您需要确保为 DNS 及 HTTP/HTTPS 配置有正确的缓存控制方案，从而最大程度发挥其自身助益。

产品功能

在云计算体系当中，网络的作用至关重要，因此我们通常需要在交付服务时对网络性能加以优化。您应考虑采用 EC2 实例网络容量、强化网络实例类型、Amazon EBS 优化实例、Amazon S3 传输加速以及动态 Amazon CloudFront 等产品功能进行网络流量优化。

Amazon S3 内容加速允许外部用户充分享受 Amazon CloudFront 将数据上传至 Amazon S3 而实现的网络优化效果。这意味着用户能够在无需任何专用 AWS 云传输连接的前提下轻松立足远程位置完成大规模数据的传输。

Amazon EC2 实例亦能够利用强化网络功能。强化网络利用单 root IO 虚拟化（简称 SR-IOV）机制为各受支持实例类型提供高性能网络容量。SR-IOV 是一种设备虚拟化方法，可提供远优于传统虚拟网络接口的更高 IO 性能与更低 CPU 占用率。强化网络可实现更高传输带宽、更高每秒数据包（简称 PPS）传输性能以及更加一致的实例间延迟水平。

Amazon 弹性网络适配器（Amazon Elastic Network Adapters ，简称 ENA）立足单一安置组为您的各实例提供 20 Gbps 网络容量，从而进一步实现性能优化。

Amazon EBS 优化型实例采用一套优化型配置堆栈以为 Amazon EBS IO 提供额外的专用容量。这套优化方案可最大程度降低 Amazon EBS IO 与实例内其它实例间的争用情况，从而为各 EBS 分卷带来最佳性能水平。

网络功能

在云端进行解决方案架构设计时，您应考虑利用各类网络功能以减少网络距离或者卡顿现象。

Amazon Route 53 提供的基于延迟路由（简称 LBR）功能可帮助大家面向全球受众提升应用程序的性能水平。LBR 可根据应用程序运行所在的不同 AWS 服务区的具体性能指标将您的客户路由至最佳 AWS 端点（包括 EC2 实例、弹性 IP 地址或者 ELB 负载均衡器），从而提供最佳使用体验。

AWS Direct Connect 负责为 AWS 环境提供专用连接，速率从 50 Mbps 到 10 Gbps 不等。这意味着您将能够拥有全面受控的延迟水平与配置带宽，确保您的应用程序能够轻松接入其它环境并拥有极佳性能水平。在配合 AWS Direct Connect 合作伙伴的情况下，您能够面向多种环境实现端到端连接，并借此建立起一套拥有一致性性能的扩展网络体系。

Amazon VPC 端点可在无需配合互联网网关或者网络地址转换（简称 NAT）实例的前提下提供可靠的 AWS 服务连接（例如 Amazon S3）。利用 VPC 端点，VPC 与其它 AWS 服务间的数据可在 Amazon 网络之内进行传输，从而保护您的实例免受互联网流量的影响。

关键性 AWS 服务

网络解决方案层面的关键 AWS 服务为 Amazon Route 53，负责提供基于延迟的路由功能。另外，采用 Amazon VPC 端点以及 AWS Direct Connect 亦可减少网络距离或者卡顿现象。

资源

请参阅以下资源以了解更多与网络 AWS 最佳实践相关的细节信息。

说明文档

- AWS 与网络产品

审查

当您首次构建自己解决方案时，您可以您可以立足于一组有限选项作为起步模板。然而随着时间推移，新型技术与方案将持续出现，您可能需要将其引入系统以进一步提升架构的性能水平。在 AWS 云当中，由于您的基础设施以“即代码”形式存在，因此可以更加轻松地完成新功能与服务实验性尝试。

利用 AWS，您能够充分发挥我们持续创新工作带来的优势，且我们的创新努力始终以客户需求为核心蓋和。我们会定期推出新的服务区，边缘站点、服务以及功能。这一切都将给您架构的性能效率带来积极改进。

在确定了架构方案之后，您应当利用数据以指导对资源类型及配置选项的具体选择。您可以利用基准测试与压力测试获取此类数据。

要采用数据驱动型架构设计方案，您需要建立起一套性能审查流程，具体内容包括：

- **基础设施即代码：** 利用 AWS CloudFormation 模板等方法定义您的基础设施。利用模板将帮助您将基础设施与应用程序代码以及配置机制一道纳

入源控制体系。如此一来，您将能够将同样的软件开发实践引入基础设施，从而实现快速迭代。

- **部署管道:** 利用一套持续集成/持续部署（简称 CI/CD）管道（例如源代码库、编译构建系统、部署以及测试自动化）进行基础设施部署。如此一来，您将能够以等同于迭代方式的可重复性、一致性以及低成本方式实现部署。

- **明确定义的指标:** 制定指标与监控机制，借以收集关键性能指标（简称 KPI）。我们建议您同时结合技术与业务等其它指标。对于网站或者移动应用而言，其中最为重要的当数首字节时间或者渲染时间。其它一般性适用性指标还包括线程数量、垃圾回收率以及等待状态等。包括每请求总体成本在内的各类业务指标则可为您提供与成本降低相关的重要启示，因此请认真考虑如何对指标作出解释。举例来说，您可以选择占比最大或者第 99 百分位数字，而非简单取平均值。

- **自动化性能测试:** 作为部署流程中的组成部分，您应以自动化方式在运行测试成功完成后触发性能测试。这种自动化机制应能够创建新环境、设置初始条件（例如测试数据），而后执行一系列基准测试与压力测试。此类测试的结果应该被绑定回特定编译版本之内，确保您能够追踪性能随时间推移发生的变化。对于长期运行测试，您可以这部分管道与其它编译版本进行异步处理。或者，您也可以利用竞价实例在夜间或者其它非工作时段执行性能测试。

- **负载生成:** 您应当创建一系列测试脚本，用以进行复制合成或者预录用户使用流程。这些脚本且以幂等且非耦合形式存在，您可能需要纳入“预热”脚本等以产生有效的结果。尽可能在您的测试脚本当中复制生产环境下的使用行为。您可以利用软件或者软件即服务（简称 SaaS）解决方案以生成负载。考虑利用 AWS Marketplace 提供的解决方案与竞价实例; 其能够以更具成本效益的方式完成负载生成任务。

- **性能可见性:** 各项关键性指标应以可见性方式供团队查看，特别是那些与各编译版本相关的指标。如此一来，您将能够了解到随时间推移所产生的重大积极或者消极趋势。您亦应当显示错误数量或者意外数量等指标，借以确保您所测试的目标系统拥有正常工作能力。

- **可视化:** 利用可视化技术明确性能问题、热点、等待状态以及低利用率的发生位置。将性能指标覆盖对架构图表之上，调用图形或者代码以更快进行问题识别。

我们可以将性能审查流程作为现有部署管道当中的一种简单扩展加以实现，而后随时间推移根据测试要求的复杂度变化进行调整。考虑到未来的架构变化，大家应当有能力建立自己的方案并复用其中的部分流程与成果。

当架构表现不佳时，其根本理由通常源自审查规程尚未落实或者无法起效。如果您的架构确实无法满足预期，则性能审查工作将帮助您利用戴明质量管理方法的计划检查（简称 PDCA）循环取代原本的迭代式改进。

基准测试

基准测试利用复合测试为您提供与组件执行效果相关的数据。在本章节当中，我们将探讨如何利用基准测试改进您的工作负载运行效果。在这里，我们暂不讨论如何利用基准测试比较不同供应商之具体产品或实现方案。

基准测试在速度方面通常较压力测试更快，而且专门用于针对您希望评估的特定组件。基准测试通常会在新项目起步时使用，意味着其无需像使用压力测试那样首先建立起整体解决方案。

对于基准测试工作，您应当遵循性能审查流程。然而，您的部署管道将只考虑到基准测试本身。您可以自行构建定制化基准测试，或者使用 TPC-DS（用于测试您的数据仓库工作负载）等行业标准测试。如果您需要对不同环境作出比较，那么行业标准基准测试往往效果更好。定制化基准测试则主要面向您希望在架构当中使用的特定运营方案类型。

在使用基准测试时，最重要的一点是对测试环境进行预热以确保获得有效结果。您应多次运行同一基准测试，借以确保能够收集到随时间推移而出现的各类差异性结论。

由于基准测试在速度上一般快于压力测试，因此其可在部署管道早期即加以运用，旨在为您的团队提供更多性能偏离反馈。当您对组件或者服务当中的重大变化进行评估时，基准测试的结果亦可帮助大家快速了解是否有必要执行这一调整。基准测试应与压力测试结合使用，因为压力测试能够告知我们自身整体工作负载在生产环境中将拥有怎样的运行表现。

关键性 AWS 服务

基准测试层面的关键 AWS 服务包括 AWS CodeDeploy 与 AWS CloudFormation。利用这些服务，您可以可重复方式对您的基础设施进行自动化测试。

资源

您可参阅以下资源以了解更多与基准测试 AWS 最佳实践相关的细节信息。

视频

- 性能频道

- AWS 上的性能基准测试

说明文档

- Amazon S3 性能优化 Optimization
- Amazon EBS 分卷性能 Performance
- AWS CodeDeploy
- AWS CloudFormation

压力测试

压力测试使用您的**实际**工作负载，因此您可以了解到自身解决方案在生产环境下的整体表现。压力测试应利用生产数据的复合或者精简版本（即删除其中的敏感或者身份信息）加以执行。对用户在您应用程序当中的使用流程进行回放或者预编程用户旅程，借此演练整体架构表现。作为其中的重要组成部分，大家还应确保自己的交付管道能够自动执行压力测试并针对预定义 KPI 及阈值进行比较，从而确保您持续实现与预期相符的运行表现。

Amazon Cloud Watch 能够跨越架构当中的多种资源进行指标收集。您亦可收集并发布定制化指标，用以涵盖业务或者派生状况。利用 CloudWatch 设置警报，用以指示阈值超出情况——这将帮助大家快速了解到测试当中出现的性能问题。

利用 AWS 服务，您可以运行生产规模环境以主动测试您的架构。由于您只需在必要时支付测试环境运行成本，因此完全能够借助更为灵活的云资源实现内部环境所根本无法提供的全规模测试工作。您亦可充分发挥 AWS 云在工作压力测试领域的固有优势，包括以非线性方式进行规模伸缩并借此了解潜在问题。您可以利用竞价实例以极低成本生成负载，并在其对生产环境造成实际影响之前发现其中的潜在瓶颈。

在为架构编写关键性用户案例时，大家应在其中纳入具体性能要求，具体包括各具体案例的性能需求。对于此类重要案例，我们应配合额外脚本化方案以确保有能力全程追踪其实际运行效果与预期要求是否相符。

压力测试往往需要耗费相当长的执行时间，因此您应当以并行方式利用其对您环境下的多套副本进行测试。这种作法的成本水平基本不变，但测试时间将大幅降低。（运行单一 EC2 实例 100 小时的成本等同于运行 100 个 EC2 实例 1 小时。）你料可选择其它服务价格更低的服务区以进一步压缩测试工作的实际成本。

您压力测试客户端的执行位置应该能够反映最终用户的实际地理位置。此外，您亦需要了解各类缓存机制（包括 DNS、Amazon CloudFront 等）以确保准确体现实际性能水平。

关键性 AWS 服务

压力测试层面的关键 AWS 服务为 Amazon CloudWatch，其允许大家在压力测试过程当中收集各项能够说明架构执行效果的重要指标。利用 CloudWatch，大家还能够建立起定制化及业务性指标。

资源

请参阅以下资源以了解更多与压力测试 AWS 最佳实践相关的细节信息。

说明文档

压力测试 CloudFront

监控

在架构构建完成之后，您需要对其性能加以监控，借以在影响到客户之前发现一切潜在问题。监控指标可用于在触及阈值时发布警报，并利用这些警报自动触发指向性能表现低下组件的后续处理操作。

在使用 AWS 时，Amazon CloudWatch 允许您监控并发送警报通知。您可以利用自动化方式通过 Amazon Kinesis、Amazon 简单队列服务（简称 SQS）以及 Amazon Lambda 触发对应操作以解决此类性能问题。

主动与被动

监控解决方案主要分为两大类：主动监控（简称 AM）与被动监控（简称 PM）。AM 与 PM 相互补充，旨在帮助大家对于自身工作负载的实际运行状态拥有更为全面的了解。

主动监控负责以脚本化方式模拟产品内各关键性路径中的用户活动。AM 应持续执行以时刻测试工作负载的执行与可用性效果。AM 能够在持续性、轻量化与可预测性等层面为 PM 提供补充。其可立足一切环境运行（特别是预生产环境），从而在客户受到实际影响之前发现各类问题或性能浮动。

被动监控通常面向基于 Web 的工作负载。PM 能够从浏览器当中收集性能指标（非 Web 类工作负载亦可采取类似的方案）。您可以跨越全部用户（或者用户中的特定子集）、地理位置、浏览器以及设备类型进行指标收集。您应利用 PM 解决以下问题：

- **用户体验性能:** PM 为您提供用户所实际感受到的性能表现，这意味着大家将能够持续观察生产环境的运行情况，同时了解随时间推移出现的各项变更对性能造成的具体影响。
- **地理性能变化:** 如果某项工作负载立足全球范围且该应用的用户来自世界各地，则利用 PM 能够帮助大家着眼于特定地理位置发现可能对用户造成影响的性能问题。
- **API 相关影响:** 现代工作负载往往大量使用内部 API 与第三方 API。PM 可提供 API 使用情况可见性，意味着您能够发现一切由内部 API 乃至第三方 API 供应方造成的性能瓶颈。

阶段性

AWS 监控工作主要分为五大阶段，具体信息请参阅《AWS **良好架构框架可靠性支柱**》白皮书当中的相关阐述：

1. 生成——监控、指标与阈值范围
2. 聚合——创建一套面向多个来源的完整视图
3. 实时处理与警报——发现与响应
4. 存储——数据管理与保留策略
5. 分析——仪表板、报告与结论

Amazon CloudWatch 为一项面向各类 AWS 云资源以及运行在 AWS 之上各工作负载的监控服务。大家可以利用 Amazon CloudWatch 收集并追踪各类指标、收集及监控日志文件以及设置警报。Amazon CloudWatch 可用于监控多种 AWS 资源，包括 Amazon EC2 实例以及 Amazon RDS 实例等，亦可监控由您应用程序及服务生成的定制化指标以及来自应用程序的一切日志文件。您可以利用 Amazon CloudWatch 实现面向资源利用率、应用程序性能以及运营状态的全系统可见性。您可利用这些结论快速执行响应操作，从而保证应用程序始终拥有顺畅的运行效果。Amazon CloudWatch 仪表板允许您创建可重复使用的 AWS 资源与定制化指标图形，借以监控运营状态并快速发现其中的问题。

需要强调的是，您需要确保自身不致面对过多误报或者无法处理的海量数据，这一点对于监控解决方案的有效性至关重要。自动触发机制能够避免人为失误并降低问题修复的时间投入。另外，您可规划**竞赛日**活动，借此以模拟方式建立生产环境以测试您的警报解决方案，进而确保其能够正确识别各类问题。

关键性 AWS 服务

监控支持层面的关键 AWS 服务为 **Amazon CloudWatch**，其能够帮助大家轻松创建警报并借此触发规模伸缩操作。以下服务与功能亦在这一层面发挥着重要作用：

- **Amazon S3:** 作为存储层存在，可配合生命周期政策与数据管理共同起效。
- **Amazon EMR:** 能够分析各项指标，帮助您监控性能表现。

资源

请参阅以下资源以了解更多与利用监控机制提升性能效率 AWS 最佳实践相关的细节信息。

视频

- AWS re:Invent 2015 | (DVO315) 利用 Amazon CloudWatch 对您的 IT 体系进行记录、监控与分析 Amazon CloudWatch
- AWS re:Invent 2015 | (BDT312) 应用程序监控：数据索引为何如此重要

说明文档

- CloudWatch 说明文档

权衡

当您设计架构解决方案时，请认真考量权衡方面的工作，确保您能够从中选出最优方案。根据实际需求，您可以在一致性、持久性、空间、时间或者延迟等因素之间作出取舍，最终得到更理想的性能表现。

利用 AWS，您可以在数分钟之内完成全球化拓展，包括在世界范围内选定距您用户最近的多个服务区以进行资源部署。您亦能够向信息存储体系（例如数据库）内动态添加只读副本，借以降低主数据库所承受的负载压力。AWS 还提供 Amazon ElastiCache 等缓存解决方案，其可提供一套内存内数据存储或缓存体系；Amazon CloudFront 则可将您的静态内容副本缓存在与最终用户更近的位置。

以下部分将详细说明您能够作出的部分权衡以及如何进行具体实现。

技术方案	适用于	使用	收益
缓存	高强度读取	空间（内存）	时间
分区或分段	高强度写入	大小与复杂性	时间
压缩	大规模数据	时间	空间
缓冲	大量请求	空间与时间	效率

缓存

大多数工作负载依赖于特定服务或数据库等组件作为惟一事实来源或统一数据视图。一般情况下，此类架构当中的组件难于扩展，且在工作负载成本当中占据重要比例。大家可以利用缓存机制在新内容与原有存储内容之间寻求平衡点，借以提升性能效率。此类技术通常以异步或者周期性方式进行更新。需要权衡的是，由于您的数据并非总能体现最新内容，因此事实来源的一致性可能受到影响。

应用级

您可以利用应用级缓存或者内存化方式在代码层面实现权衡。立足架构层级，您应选用 Amazon ElastiCache——这是一套内存内数据存储或缓存机制，目前支持着 Redis 以及 Memcached 等多种引擎。当请求进行缓存处理后，执行时间将大幅缩短。如此一来，您将能够立足缓冲层进行横向扩展并帮助高强度使用组件降低负载水平。

数据库级

数据库副本可通过将全部变更复制至主数据库以进行副本读取的方式增强性能表现（不适用于 Oracle 或者 SQL Server）。这种复制操作使其能够超出单一数据库的容量限制，从而更好地满足高强度读取型数据库工作负载的实际需求。

Amazon RDS 以全面托管服务方式提供读只读副本。您可以为特定源数据库创建一套或者多套副本支持大规模应用程序读取流量，从而显著提升总体读取数据通量。如果数据库引擎支持，您亦应为读取副本提供额外的索引。举例来说，您可以在 MySQL 读取副本当中添加更多索引。对于延迟敏感型工作负载，您应利用多可用区功能指定各读取副本的具体所在可用区，从而降低跨可用区流量水平。

地理级

另一项缓存用例在于使用内容分发网络（简称 CDN），这是降低客户延迟体验的理想方式。您应利用 CDN 存储静态内容并加速动态内容交付。您应考虑利用 CDN 作为 API；另外，动态内容亦可通过各类网络优化方法的运用而获得助益。

Amazon CloudFront 可用于交付您的整体网站，具体包括经由一套全球边缘位置网络交付动态、静态、流式以及交互式内容。指向您内容的请求将被自动路由至最近的边缘位置处，从而确保以最佳性能实现内容交付。Amazon CloudFront 服务专门针对其他 AWS 服务进行优化，可顺畅匹配多种其它 Amazon Web Services——包括 Amazon S3、Amazon EC2、弹性负载均衡以及 Amazon Route 53。Amazon CloudFront 亦可以无缝化方式匹配其它负责存储原始及最终版本文件的非 AWS 源服务器。

关键性 AWS 服务

缓存解决方案层面的关键 AWS 服务为 Amazon ElastiCache，其可提供通用型应用程序缓存；以及 Amazon CloudFront，其允许大家以尽可能邻近用户的方式进行信息缓存。

资源

请参阅以下资源以了解更多与缓存 AWS 最佳实践相关的细节信息。

说明文档

- Amazon ElastiCache 实现最佳实践
- AWS CloudFormation 最佳实践

视频

- 利用 Amazon ElatiCache 为应用带来“动力增压”

分区或分段

在使用关系数据库等技术方案时，其出于一致性限制而要求使用单一实例，这意味着您仅能够实现垂直规模伸缩（利用更高规格实例及存储功能）。而一旦达到

垂直规模伸缩上限，您亦可选择数据分区或分段方式解决此类需求。通过这种模式，数据将跨越多种数据库模式进行拆分，且各个分区皆运行有自己的主数据库实例。

Amazon RDS 能够帮助您承担运行多个实例时的运营难题，但具体分段工作对于应用程序而言仍然相当复杂。大家需要对应用程序的数据访问层进行修改以确保其有能力感知数据的拆分方式，最终将请求定向至正确的实例处。（您可以利用代理或者路由机制移除应用程序当中的缓存代码，或将缓存机制纳入数据访问层。）另外，任何模式变更皆需要跨越多数据库模式进行执行，因此大家有必要以自动化方式处理此类操作。

NoSQL 数据库引擎通常可执行数据分区及复制，从而以横向方式对读取及写入操作进行规模伸缩。此类操作以透明化方式存在，无需在应用程序内的数据访问层执行任何数据分区逻辑操作。Amazon DynamoDB 则特别以自动化方式管理您的表分区工作，并可根据表大小或者读取/写入配置容量变更添加新的分区。

分区或分段机制能够对高强度写入工作负载进行规模伸缩，但同时亦要求整体均衡的分区或分段进行数据分布及访问。这种作法会在关系数据库解决方案当中引入复杂性，同时 NoSQL 解决方案往往需要牺牲一致性以实现交付。

关键性 AWS 服务

分区或分段层面的关键 AWS 服务为 Amazon DyanmoDB，其负责帮助您以自动化方式管理表分区。

资源

请参阅以下资源以了解更多分区与分段 AWS 最佳实践相关的细节信息。

说明文档

- DynamoDB 最佳实践

视频

- AWS re:Invent 2015 | (DAT401) Amazon DynamoDB 深度解析

压缩

数据压缩意味着在计算时间与空间之间作出取舍，其可显著降低存储与网络要求。压缩机制可应用于文件系统、数据文件以及电子表格与图像等 Web 资源，同时亦适用于 API 等动态响应。

Amazon CloudFront 支持边缘位置压缩。源系统可以标准方式交付资源，并在 Web 客户端能够支持时由 CDN 自动进行资源压缩。

在将大量信息传入或移出云端时，大家应考虑非网络解决方案。AWS Snowball 属于一套 PB 级别数据传输解决方案，旨在利用安装设备将大规模数据传入及传出 AWS 云。利用 AWS Snowball 能够解决各类大规模数据传输挑战，具体包括高网络成本、长传输时间以及其它安全要求。

Amazon Redshift 利用列式数据存储并配合压缩功能。相较于将数据存储为多行，Amazon Redshift 以列式方式进行数据组合。列式数据存储机制与磁盘上的有序数据存储类似，因此相较于列式数据存储可实现更佳压缩效果。Amazon Redshift 采用多种压缩技术，通常能够实现远优于传统关系数据存储方案的压缩成效。数据会被加载至一套空表内，Amazon Redshift 会自动对数据进行采样并选择最适合的压缩方案。

关键性 AWS 服务

压缩层面的关键 AWS 服务为 Amazon CloudFront，其可支持边缘位置的压缩功能。

资源

请参阅以下资源以了解更多与压缩 AWS 最佳实践的细节信息。

说明文档

- Amazon CloudFront：交付压缩文件
- Amazon RedShift：列式存储
- AWS Snowball: AWS Snowball 是什么？

缓冲

缓冲机制利用队列接收来自生产方的消息（工作单元）。出于弹性考量，队列应采用持久性存储机制。**缓冲机制**负责确保各应用程序能够在以不同运行速率的情况下实现彼此通信。消息随后由消费方进行读取，借此允许各消息确切匹配由生

产方到消费方的数据通量速率。利用缓冲区，您能够将生产方通量速率从消费方中解耦出来。大家不再需要分神于生产方处理数据持久性与“**背压**”（消费方运行缓慢会影响到生产方速度）。

在您的工作负载生成大量写入负载且不需要立即加以处理时，您可以利用缓冲区以平滑消费方的实际需求。

在 AWS 上，大家可以从多种服务当中作出选择，从而实现缓冲效果。Amazon 简单队列服务（简称 SQS）提供一套队列，允许单一消费方阅读个别消息。Amazon Kinesis 提供一套流体系，允许多个消费方读取同样的消息内容。

Amazon SQS **队列**属于一套可靠、可扩展且全面托管库，面向等待处理的消息。Amazon SQS 能够快速且可靠地为应用程序提供队列消息，且应用程序中某一组件生成的消息可供其它组件消费。在这套方案当中，队列作为生产方的消息发送目的地，且该目的地拥有公开地址。

为了降低成本，生产方可利用 Amazon SQS 批量 API 操作进行消息发布。消费方利用固定大小的 Auto Scaling EC2 实例组从公开队列内读取消息，从而应对实例故障。**长轮询**机制则可使您的 Amazon SQS 队列在上线后立即供消息检索。利用长轮询方式可降低消息检索成本。Amazon SQS 利用**消息可见性**特性以隐藏各已经读取完成的消息。消息可见性能够降低您重复处理同一消息的风险，不过需要强调的是，默认 Amazon SQS **至少进行一次**交付，意味着您可能多次收取到同一条消息。

如果需要保证仅进行一次处理，大家则可使用 Amazon SQS 先进/先出（简称 FIFO）队列。消息可见性功能允许各尚未经过处理的消息重新出现（例如在发生实例故障的情况下）。对于长期运行的任务，您可以扩展可见性超时，这时您的应用程序需要在消息被处理完成后对其进行主动删除

另一种方案在于利用 Amazon Kinesis 提供缓冲区。与 Amazon SQS 的区别在于，Amazon Kinesis 允许多个**消费方**随时读取同一消息。然而，单一将利用 Kinesis 客户端库或者 AWS Lambda 进行读取，而后被交付至一个且惟一一个消费方——即“**应用程序**”，此名称在消费方接入处理流时提供。不同的“**应用程序**”可同时消费同一批消息，相当于建立起一套“发布与订阅”模式。

在利用基于缓冲的方案设计架构时，请注意两大关键性因素：其一，工作生产与工作消费之间的延迟水平接受范围如何？第二，您计划如何处理重复的工作请求？

为了在由多个消费方消费工作条目的场景下优化速度表现，大家可以使用 Amazon 竞价实例。利用竞价实例，您可以竞标备用 Amazon EC2 计算容量。要处理 Amazon SQS 中的重复消息，我们建议您使用 Amazon SQS（FIFO）队列，其能够实现仅一次处理效果。

对于 AWS Kinesis，大家可以考虑在 Amazon DynamoDB 当中配置一套表，用于追踪已经被成功处理的工作条目。另一种重复消息处理方式则为幂等处理，其更适用于对数据吞吐量要求更高的场景。**幂等**机制允许消费方利用应用程序的逻辑模式对消息进行多次处理，且消息在按序处理时不会影响到下游系统或者存储。

关键性 AWS 服务

缓冲层面的关键 AWS 服务为 Amazon SQS，其允许大家利用队列实现生产方与消费方解耦。

资源

请参阅以下资源以了解更多与缓冲 AWS 最佳实践相关的细节信息。

说明文档

- Amazon SQS 最佳实践

结论

实现并保持性能效率需要采取数据驱动型实践。您应立足访问模式作出权衡，旨在进一步优化性能表现。利用基于审查的流程配合基准测试及压力测试允许您根据实际需求选择适当的资源类型与配置。采取基础设施即代码机制能够帮助您快速、安全地进行架构演进，您同时亦可利用数据作出关于架构的基于事实决策。将主动与被动监控手段相结合，将确保您的架构性能不会随时间推移而发生退化。

AWS 努力帮助您构建具备性能效率的架构，同时借此实现业务价值。为了使您的架构真正具备可行性，您应采用本份白皮书中提及之工具与技术。

贡献者

以下个人与组织为本份白皮书的编撰作出贡献：

- Philip Fitzsimons， Amazon Web Services 高级良好架构经理
- Julien Lépine，Amazon Web Services 解决方案架构师
- Ronnen Slasky，Amazon Web Services 解决方案架构师

扩展阅读

欲获得更多帮助，请参阅以下资源：

- AWS 良好架构框架