Analytics Lens

AWS 架构完善的框架

2020 年 5 月

版权声明

目录

摘要

本文档介绍了 AWS 分析类相关的架构完善的框架。文档涵盖多种常见的分析应用场景，同时确定各项关键元素，帮助您根据最佳实践设计自有工作架构。

简介

AWS 架构完善的框架（AWS Well-Architected Framework）能够帮助您认识到您在 AWS 上构建系统时所做决策的优缺点。通过使用该框架，您将了解如何在云环境中设计和运行可靠、安全、高效且经济实惠的系统架构的最佳实践。它为您提供了一种方法，使您能够根据最佳实践持续衡量架构，并确定需要改进的方面。我们相信，拥有架构良好的系统能够大大提高实现业务成功的可能性。

在本“Lens”当中，我们将重点介绍如何在 AWS 云中设计、部署以及构建您的分析应用程序工作负载。为了简洁起见，我们仅探讨架构完善的框架当中与分析工作负载相关的详细内容。在架构的实际设计当中，您还须考虑本文档未加讨论的其他最佳实践与实际问题。关于更多细节，请参阅我们的 AWS 架构完善的白皮书。

本文档主要面向技术类人群，包括首席技术官（CTO）、架构师、开发人员以及运营团队成员。在阅读本文之后，您将了解在设计分析类应用程序与环境架构时应当遵循的 AWS 最佳实践与策略。

定义

AWS 架构完善的框架基于五大核心支柱，分别为安全性、可靠性、性能效率、成本优化与卓越运营。对于分析类工作负载与环境，AWS 提供多种核心组件，可帮助您设计出更为健壮的架构方案。在本章节中，我们将概述此白皮书中所涉及的 AWS 服务。

将数据架构组织为“层”概念的形式，可帮助大家根据实际需求选择最适用的访问控制，管道，提取、转换与加载（简称 ETL）乃至特定用例集成。在构建分析类工作负载时，我们主要考虑以下六方面因素：

- 数据摄取层
- 数据访问与安全层
- 目录与搜索层
- 集中存储层
- 处理与分析层
- 用户访问与界面层

数据摄取层

数据摄取层主要负责将数据摄取至集中存储层（例如数据湖）以进行分析。摄取层包含多项服务，这些服务用来处理来自

外部来源数据集的批量与实时流数据，具体涵盖网站点击流、数据库事件流、财务交易、社交媒体馈送、IT 日志、位置跟踪事件、物联网遥测数据、本地数据源以及云原生数据存储等等。

Amazon Kinesis 是包含一系列用于实时数据摄取的服务，提供对流数据进行安全加载与分析功能，同时可将流数据传输至 Amazon Simple Storage Service（简称 Amazon S3）以实现长期存储。我们还提供 Amazon Managed Streaming for Kafka（简称 MSK），这是一项全托管服务，可帮助用户运行高可用性和安全性的 Apache Kafka 集群来处理流数据，且无需修改现有代码库。

使用 AWS Database Migration Services（简称 DMS），您可以在保证源数据库正常运行的同时复制并摄取该数据库。此项服务支持多个数据库源与目标，包括直接将数据写入至 Amazon S3。为了加快 DMS 迁移，您还可以选择 AWS Snowball——这项服务通过使用安全的物理设备实现大规模数据传入到 AWS 云或从 AWS 云中传出。最后，您也可以使用 AWS Direct Connect，这项服务会在本地数据中心与 AWS 云环境之间建立一个专用网络连接。

您还可以使用其他数据摄取方法，例如 AWS IoT Core，这是一套托管平台，能够可靠且安全地大规模处理消息并将消息内容路由至 AWS 数据存储当中。AWS DataSync 是一项数据传输服务，能够自动简化并加快数据在本地存储系统（例如 NFS）与 AWS 存储服务（例如 Amazon EFS 及 Amazon S3）之间的移动与复制流程，并将数据内容摄取至分析类工作负载当中。

数据访问与安全层

数据访问与安全层提供一套完整机制，用于实现数据资产保护及访问，以确保数据被安全地存储，并且只向授权的人提供访问案。该层的具体功能包括：

- 保护指向集中数据存储库（即数据湖）的安全访问
- 保护指向集中数据目录的访问
- 对数据库目录中的数据库、表及列进行细粒度访问控制
- 对传输及静态数据进行加密

要安全管理指向 AWS 服务与资源的访问操作，您需要使用 AWS 身份与访问管理（AWS Identity and Access Management，简称 IAM）。使用 IAM，您可以创建并管理 AWS 用户与分组，并通过权限机制许可及拒绝指向 AWS 资源的访问操作。AWS CloudTrail 则帮助用户记录、持续监控并保留 AWS 基础架构中各用户及角色数据访问相关的账户活动。此外，您还可以使用 Amazon CloudWatch 收集监控与运营数据，以日志、指标以及事件的形式把握分析类工作负载的运行状态。

要实现静态数据加密，请使用 AWS 密钥管理服务（AWS Key Management Service，简称 KMS）。这是一种安全的弹性服务，可帮助您轻松创建并控制用于加密数据的密钥。部分法规还要求您将 KMS 与 AWS CloudHSM 配合使用——AWS CloudHSM 为基于云的硬件安全模块（HSM），用户可以借此轻松生成并使用自己的加密密钥。AWS CloudHSM 可帮助用户证明安全性、隐私与防篡改等法规层面的合规性，遵循 HIPAA、FedRAMP 与 PCI 等法规的具体要求。您还可以在 KMS 配置中将 CloudHSM 集群设定为自定义密钥存储，借此替代默认的 KMS 密钥存储。

AWS Lake Formation 是一项集成化数据湖服务，可帮助用户轻松摄取、清洗、分类、转换并保护数据，并将处理后的数据用于分析及机器学习（ML）。Lake Formation 还提供自己的权限模型，用于进一步扩展 AWS IAM 提供的权限模型——包括配置数据湖的数据访问与安全策略，同时审计并控制来自 AWS 分析与 ML 服务的访问操作。通过这套集中定义的权限模型，我们可以使用简单的授权 / 撤销机制对存储在数据湖内的数据进行细粒度访问。

目录与搜索层

分析类工作负载中的目录与搜索层，用于管理跟数据资产相关的元数据的发现与分类。随着数据资产数量与规模的增长，该层还提供搜索功能。在这类使用场景下，您需要根据事先定义的条件查找表摄取数据子集，这也是一种在分析类应用当中相当常见的操作。

AWS Glue 是一项全托管型提取、转换与加载（ETL）服务，可帮助客户轻松准备并加载等待分析的数据。您可以直接将 AWS Glue 指向保存在 AWS 上的数据，该服务会自动识别数据，并将与之关联的元数据（例如表定义与 schema）存储在 AWS Glue 数据目录当中。在分类完成之后，您即可直接对数据进行搜索、查询以及 ETL。

使用 Amazon Elasticsearch Service，您可以在 AWS 云中部署一套或多套全托管 Elasticsearch 集群，进而搜索您的数据集。您可以直接访问 Elasticsearch API——该服务能够将现有代码与应用程序无缝匹配起来，提供对 Kibana、Logstash 以及其他 AWS 服务的集成支持，而且内置有警报与 SQL 查询功能。

Amazon 关系数据库服务（Amazon Relational Database Service，简称 Amazon RDS）能够极大简化云环境中关系数据库的设置、操作与扩展等流程。除了 AWS Glue 之外，您也可以使用 Amazon RDS 为 EMR 创建外部 Hive 元存储。元存储中包含对表以及其基础数据相关的描述，例如分区名称、数据类型等等。

Amazon DynamoDB 是一项 NoSQL 数据存储服务，可用于创建高性能、低成本的外部索引，并借此将可查询的属性映射至 Amazon S3 对象键。Amazon DynamoDB 具备自动扩展与高可用性等优势，帮助用户摆脱传统服务器带来的维护负担。

集中存储层

集中存储层负责管理摄取自各生产程序处的数据，并将其进一步提供给下游应用程序。集中存储层是数据湖的核心，应该能够支持对一切数据类型的存储，包括非结构化数据、半结构化数据与结构化数据。随着时间推移，您的数据量可能不断增长，因此集中存储层还应以安全且经济高效的方式实现灵活扩展。

在数据处理管道当中，我们也可以将数据存储在流程的中间阶段处，这既可以避免不必要的重复操作，也可以让中间数据同时供多个下游消费程序使用。根据实际场景的需求，这些中间数据可能需要经常更新、临时存储或者长期存储。

Amazon S3 具备近乎无限的可扩展性，99.999999999%（11 个 9）持久性以及原生加密与访问控制功能，因此非常适合作为集中存储方案的底层基础。随着数据存储需求的增长，您可以将数据迁移至低成本层，例如 S3 Infrequent Access 或者 Amazon S3 Glacier，通过多种生命周期管理策略节约存储成本，同时保证原始数据的安全性与完整性。另外，您也可以使用 S3 Intelligent-Tiering，借此在数据访问模式发生改变时自动优化存储成本，且不会对性能或运营开销造成任何影响。

Amazon S3 让多租户环境的构建变得非常轻松，不同用户可以将自己的独特数据分析工具与一组通用数据对接起来。与需要建立多套分布式数据副本的传统解决方案相比，多租户环境能够显著改善成本与数据治理能力。为了进一步降低访问门槛，Amazon S3 还提供简单的 RESTful API，目前 Apache Hadoop 以及大多数第三方独立软件供应商（ISV）及分析工具供应商已经在支持这些 RESTful API。

在 Amazon S3 的支持下，您的数据湖能够将存储同计算 / 数据处理机制拆分开来。在传统 Hadoop 及数据仓库解决方案当中，存储与计算总是紧密结合在一起，因此对成本与数据处理工作流程的优化总是困难无比。Amazon S3 允许您以原生格式存储一切数据类型，并指定任意数量的虚拟服务器执行数据处理。此外，您还可以与多种无服务器解决方案进行集成，例如 AWS Lambda、Amazon Athena、Amazon Redshift Spectrum、Amazon Rekognition 以及 AWS Glue 等等，您无需置备或管理任何服务器，即可轻松获得数据处理能力。

Amazon Elastic Block Store（简称 EBS）提供永久性的块存储卷，可用于 AWS 云中的 Amazon EC2 实例。每一个 Amazon EBS 存储卷都会自动在其所在可用区内复制，保证您的系统免受组件故障的影响，借此实现高可用性与持久性。对于分析类工作负载，您可以将 EBS 与运行在 EC2 实例上的各类大数据分析引擎（例如 Hadoop/HDFS 生态系统或者 Amazon EMR 集群）、关系数据库与 NoSQL 数据库（例如微软 SQL Server 与 MySQL，或 Cassandra 与 MongoDB）、流数据与日志处理应用程序（例如 Kafka 与 Splunk）以及数据仓库应用程序（例如 Vertica 与 Teradata）配合使用。

处理与分析层

处理与分析层负责提供对数据集查询及处理（包括清洗、验证、转换、丰富以及规范化）相关的工具及服务，并以批量及实时流数据模式提取业务见解。处理与分析层当中包含多项具体服务。

Amazon EMR 是一项托管服务，能够跨越多个 Amazon EC2 实例轻松运行并动态扩展 Apache Spark、Hadoop、HBase、Presto、Hive 以及其他多种大数据框架，并与 Amazon S3 及 Amazon DynamoDB 等其他 AWS 数据存储实现交互。

Amazon Redshift 是一套全托管数据仓库，它使得使用标准 SQL 和现有的商业智能（BI）工具分析所有数据变得简单而经济高效。Redshift Spectrum 属于 Amazon Redshift 中的一项功能，可帮助用户查询保存在 Amazon S3 中的 EB 级超大规模非结构化数据，且无需涉及任何加载或 ETL 操作。Redshift Spectrum 能够在短短几分钟内对 EB 甚至更高数量级的数据执行极为复杂的查询。

Amazon Athena 是一项交互式查询服务，可使用标准 SQL 轻松分析保存在 Amazon S3 中的数据。Athena 具备无服务器特性，用户无需管理任何基础设施，仅按使用量付费即可。Athena 还能够与 AWS Glue 数据目录相集成，帮助您跨多项服务创建起统一的元数据存储、爬取数据源以识别重要 schema、使用新的 / 经过修改的表及分区定义执行目录填充，同时实现对 schema 的版本控制能力。

使用 Amazon Neptune，您可以创建起一套快速、可靠的全托管图数据库，从而轻松构建并运行基于高连接度数据集的应用程序。Amazon Neptune 支持多种流行图模型，包括 Property Graph 与 W3C 的 RDF，且支持二者对应的查询语言 Apache TinkerPo Gremlin 与 SPARQL。Amazon Neptune 还能够增强图关系用例的功能，实现包括推荐引擎、欺诈检测、知识图谱、药物发现以及网络安全性等多种应用。

Amazon SageMaker 是一套全托管机器学习平台，帮助开发人员与数据科学家快速灵活地构建、训练并部署任意规模的机器学习模型。数据科学家可以使用它针对不同数据湖元素轻松完成 ML 模型的创建、训练与部署。

除此之外，还有更多现有服务也可以用于处理与分析，包括 Amazon Kinesis、Amazon RDS、Apache Kafka 以及 AWS Glue ETL 作业。

用户访问与接口层

用户访问与接口层提供了用于用户访问的安全方式和用于管理用户的管理接口。

AWS Lambda 帮助用户在托管平台上运行各类无状态无服务器应用程序，能够在函数层面支持微服务架构、部署以及执行管理等功能。

使用 Amazon API Gateway，您可以运行与 Lambda 集成的全托管 REST API 以执行业务逻辑，同时实现流量管理、授权与访问控制、监控以及 API 版本控制等多种功能。例如，您可以使用 API 网关（通过 HTTPS 接收请求）创建数据湖 API。在发出 API 请求时，Amazon API Gateway 会使用自定义授权器（Lambda 函数）来确保在数据被服务之前所有请求都被授权。

使用 Amazon Cognito，您可以轻松在无服务器应用程序当中添加用户注册、登录与数据同步等功能。Amazon Cognito 用户池提供内置登录屏幕，并通过安全性声明标记语言（SAML）实现与 Facebook、Google 及 Amazon 的身份联合验证。Amazon Cognito Federated Identities 帮助您更安全地对无服务器架构内的各类 AWS 资源进行受限、受控访问。

总体设计原则

架构完善的框架确定了一组总体设计原则，帮助您在云上对分析类应用程序进行良好的设计：

- **自动数据摄取：**应用使用触发器、时间表与变更检测等机制实现数据摄取自动化。自动化摄取流程能够消除手动操作中常见的错误，保证数据到达时即进行处理，同时以更低成本实现系统的创建与复制。您可以使用编排工具中的调度程序选项实现定期调度并触发摄取，或者为流传输应用提供持续运行的摄取脚本。
- **针对故障与重复项设计的摄取机制：**在发生故障时，特定消息可能会传递多次（包括对下游呼叫时的正常重试），因此来自请求与事件所触发的摄取操作必须相互幂等。
- **保留原始源数据：**应按原样保留已摄取的原始数据，这是因为原始格式下的原始数据允许您在发生故障时重复此前的 ETL 过程。在管道执行期间，不应该对原始数据文件进行任何转换。
- **用元数据描述数据：**随着数据集的多样性和数量增长，任何进入数据存储环境的数据集都必须被发现和分类。而通过捕捉与数据存储相关的元数据，我们可以确保下游应用程序能够正确使用摄取到的数据集。此外，应保证整个过程都有完整的文档记录且自动化。
- **建立数据沿袭：**数据沿袭是指跟踪数据来源，以及它在不同数据系统之间的流动情况。只有具备这种从源到目的地查看、跟踪并管理数据流的能力，才能极大简化对分析管道中一切错误的跟踪过程，同时深入理解数据的演变过程。
- **使用正确的 ETL 工具处理相关作业：**对于提取、转换与加载（ETL）工具，我们往往面对着多种选择：为解决特定问题而构建的定制化方案、由开源组件拼凑而成的方案、以及获得商业许可的 ETL 平台。请根据您的需求选择最合适的 ETL 工具，借此简化源与目标之间的工作流程。要考察的因素包括对复杂工作流、API 和特定语言的支持、到各种数据

存储的连接器、性能、预算和企业规模。

- **编排 ETL 工作流：**自动化整个 ETL 工作流。在分析环境中，单一过程或者作业的输出通常会被用作另一过程或作业的输入。这种链式 ETL 作业能够保证整个 ETL 工作流以无缝方式执行，同时帮助大家轻松跟踪并调试一切故障。
- **适当进行存储分层：**将数据存储在最佳层中，以保证您的分析应用程序能够充分发挥该存储服务层的特性优势。在进行存储服务选择时，最重要的两大基本指标分别为：存储格式与访问频率。首先将数据集分配给不同的服务，而后对不同服务的实际表现进行评估，即可逐步为您的分析类应用程序构建起健壮的后端存储基础架构。
- **保护与管理整个分析管道：**我们当然有必要保护数据资产本身，以及用于存储、处理及分析这些数据的基础设施。要保护数据资产，首先需要实施细粒度控制手段，仅允许授权用户查看、访问、处理及修改特定资产，同时确保阻止一切未授权的用户采取任何可能有损数据机密性及安全性的操作。同样重要的是，在分析流程中的各个阶段，执行访问的具体角色可能发生变化；这就要求我们确保在各阶段只允许授权用户进行访问。
- **设计出可扩展且可靠的分析管道：**保证执行分析的计算环境具备可靠性与可扩展性，以避免数据量或者执行速度的增长对生产管道造成负面影响。另外，提高数据可靠性并优化查询性能以支持不同的分析类应用程序也同样是分析工作流设计中的重点，我们需要保证管道有能力支持批处理、流数据摄取、快速即席查询以及数据科学等多种消费程序式。

场景

在本章节中，我们将介绍分析类应用程序当中常见的五大核心场景，以及各场景如何影响 AWS 上分析环境的具体设计与架构。我们将介绍五大场景所对应的基本假设、设计决策的一般驱动因素，以及用于实现这些场景的参考架构。

数据湖

数据湖是一套集中式存储库，用户可以在这里存储任意规模的结构化与非结构化数据。您可以直接存储原始数据（无需预先构建数据结构），而后运行各种分析类型（仪表板、可视化、大数据处理、实时分析以及机器学习）以更好地指导决策。

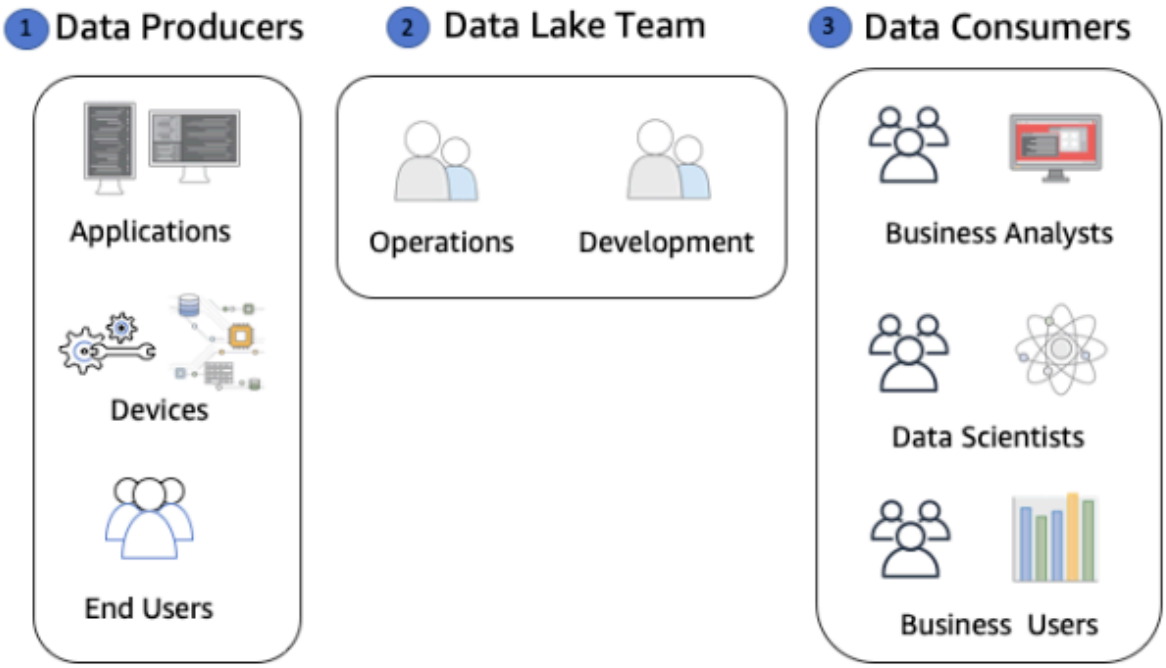
要从数据当中成功产生业务价值，组织必须建立起以数据湖为基础的新型分析能力，例如对来自日志文件、点击流、社交媒体以及联网设备的数据进行机器学习处理的能力。只有这样，您才能吸引并留住客户、提高生产力、主动维护设备并制定出明智的决策，进而确定业务机遇并采取行动，最终实现业务增长。

数据湖架构的主要挑战是存储原始数据时无法实现内容监督。为了实现数据可用性，您需要明确定义一套数据分类与保护机制。没有这套机制，这些数据将无法被找到或信任，从而导致“数据沼泽”。另外，为了满足不同利益相关方的需求，数据湖还必须具备治理功能、语义一致性与访问控制机制。

特点

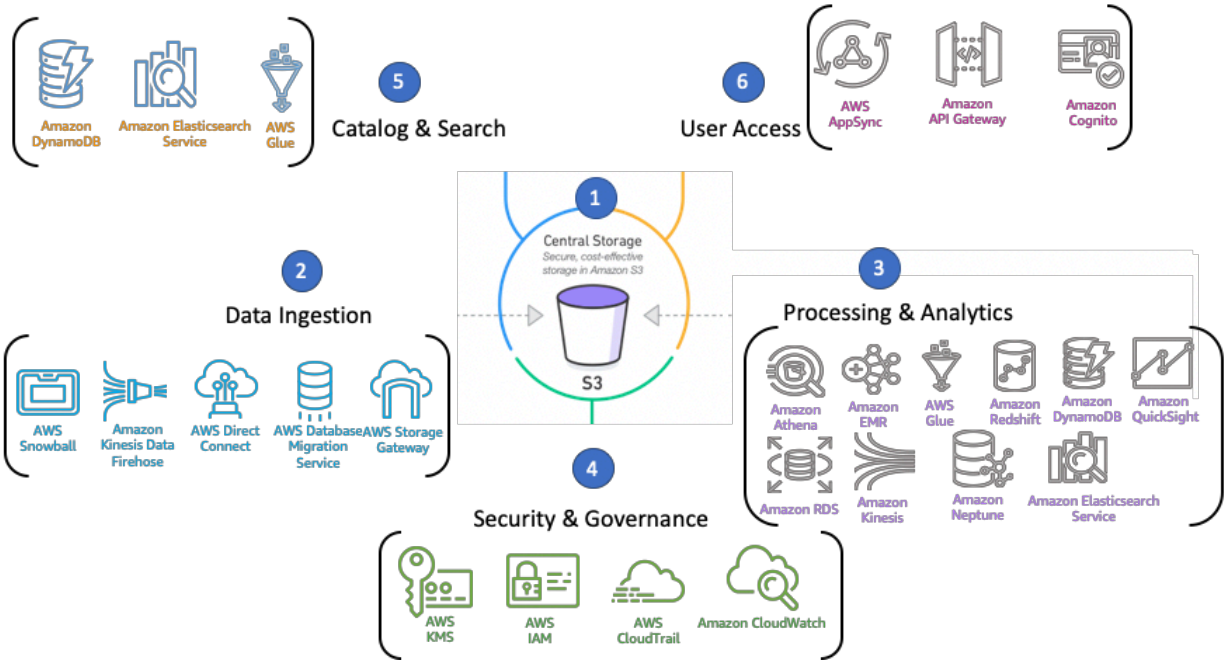
- 无论源类型、数据结构以及数据数量如何，所有原始源数据都应存储在同一位置，这就是所谓“唯一真相源”。
- 数据湖应始终保证存储与计算资源彼此解耦。
- 数据湖不仅需要支持良好的摄取与使用速度，同时还要充分考虑到数据设计层面的灵活性需求。换句话说，数据湖只需告知数据提供方将数据旋转在何处（哪个 S3 存储桶）；其他关于存储结构、架构、摄取频率以及数据质量的选择都应由数据生产者自行决定。
- 数据湖支持读时模式（schema on read）。在将数据从数据湖导入至计算环境时，我们可以使用多种 schema，这一点与采用固定结构记录数据的数据仓库有所不同。
- 数据湖在设计层面应作为低成本存储方案存在。由于历史数据的整体商业价值会随着时间推移而不断下降，这种定位将保证历史数据以较低成本保存更长时间。
- 数据湖支持保护与安全规则。这些规则提供了只允许授权访问数据的机制，同时跟踪数据在流经湖中各层时的具体使用情况。

参考架构



图一：数据湖的组织动态

- 数据生产者属于组织收入的产生者。这些实体，无论是逻辑（软件）实体还是物理（应用程序用户）实体，只有在将产生的数据存储在数据湖中后，才需要强制遵循与数据相关的对应合约（schema、频率、结构等）。
- 数据湖团队通常包含运营团队与开发团队两重属性：运营属性负责定义数据湖内须强制执行的安全与策略机制，开发属性负责支持当前 schema 与 ETL 管理。
- 数据消费者通过数据湖团队的授权从数据湖中检索数据，并进一步迭代该数据以满足业务需求。



图二：高级数据湖技术参考架构

1. Amazon S3 是 AWS 上的数据湖核心。Amazon S3 能够支持 ETL 处理与分析环境所创建并使用的一切原始及迭代数据集的对象存储方案。根据数据湖的实际业务需求，我们可以将 Amazon S3 中容纳的数据划分为两层或者三层结构：
   - T1 存储（原始数据）——此存储层为一个或者多个 S3 存储桶，用于承载来自摄取服务的原始数据。最重要的是，此层中的数据必须以原始格式进行维护与存储，不可进行任何形式的数据转换。
   - T2 存储分析优化——此存储层负责托管在 T1 层中通过 ETL 作业处理（例如 Spark on EMR，或 AWS Glue）将原始数据转换成的常见列格式（例如 Parquet、ORC 或 Avro）而产生的迭代数据集。这些数据将按分区形式组织并以列格式存储，这使得从 T2 存储层摄取数据的计算环境可以以较低的成本获得更好的性能。
   - T3 特定于用例的数据集（可选）——此层中托管的数据属于 T2 层数据的子集，针对特定用例数据集组织而成。T3 数据通常拥有更高的访问权限与安全性约束。根据实际使用情况，数据提供方将由更合适的计算与分析环境充当（例如 Amazon EMR、Amazon Redshift、Amazon Neptune 以及 Amazon Aurora）。

2. 数据湖架构的数据摄取模块，代表着将数据持久存储在 Amazon S3 中的具体过程。来自数据湖这一部分的数据旨在成为第 1 层原始数据。除了需要预先指定存储数据所使用的 S3 存储桶之外，数据生成服务不需要考虑其他任何约束条件、结构一致性或者数据合约要求。

3. 处理与分析（Processing & Analytics）部分囊括多种 AWS 服务，用于处理或存储来自 T1 层的 ETL 数据，并将处理后的数据传递至 T2 层。接下来，大家可以通过交互式查询或者机器学习 / 建模等分析或机器学习环境使用 T2 层中的数据。

4. 保护与安全（Protect & Secure）是一种总体抽象，被全面集成至数据湖内的多项服务当中。这一部分主要负责强调处理静态 / 传输数据时，身份验证、授权、审计、合规性与加密等机制的重要性，这些机制需要全面对接数据湖内的摄取、存储与处理 / 分析等各个环节。这些安全概念大多以功能集的形式被集成至每一项数据处理服务当中，用以保护数据管理服务的访问能力与完整性。关于如何将不同的安全针对性服务（例如 KMS、IAM 以及 CloudTrail）集成至摄取与分析服务（例如 Kinesis 与 Amazon Redshift）的更多详细信息，请参阅后文中的“安全支柱”部分。

5. 目录与搜索模块负责捕捉、管理、维护、索引以及搜索托管在数据湖存储（Amazon S3）内数据集中的元数据。为此，组织通常需要对数据湖目录所管理的各个数据集所需要的最小元数据集（包括描述、标签及使用说明等）进行授权。也正是通过这种方式，数据湖数据集内的各个单一对象都会被映射至整体数据集的元数据记录中。您可以使用 DynamoDB、Elasticsearch Service 以及 AWS Glue Catalog 等 AWS 服务来跟踪并索引数据湖内所托管数据集的元数据。使用 Amazon S3 新的或者更新的对象事件可以触发 AWS Lambda 函数，帮助您轻松保持目录的持续更新。

6. 访问与用户接口

   在 AWS 中，包括 Amazon API Gateway、AWS Lambda 以及 AWS Directory Service 在内的多种服务，都可帮助您安全跟踪执行进程：

   - 创建、修改并删除新的数据湖数据集及其对应元数据。
   - 负责创建、更新及合并迭代数据集的 ETL 作业（用于将 T1 数据转换为列式、压缩和高性能的 T2 数据）的创建与管理。
   - 对摄取 T2 数据的分析计算环境进行创建、删除或修改，从而实现后续查询、可视化开发及见解分析。

配置说明

- 确定数据湖摄取的具体位置（在本文示例中为 S3 存储桶）。根据业务需求选择对应的频率与隔离机制。
- 对于 T2 层数据，使用与常见查询过滤器相匹配的键进行数据分区。如此一来，我们可以使用通用型分析工具对原始数据文件进行修剪以提高性能表现。

- 选择最佳文件大小，以减少计算环境摄取期间发生的 Amazon S3 往来数据量：
  - 推荐：512 MB-1 GB/ 每分区，列格式（ORC/Parquet）。
- 根据之前提到的最佳文件大小，频繁执行计划内压缩操作。
  - 例如，如果每小时文件太小，则将其压缩为每日分区。
- 对于频繁更新或删除的数据（即可变数据）：
  - 将副本数据临时存储在 Amazon Redshift、Apache Hive 或者 Amazon RDS 等数据库当中，直到数据转为静态，而后转移至 Amazon S3；或者
  - 将数据附加至各分区的增量文件当中，并使用 AWS Glue 或 Apache Spark On EMR 按计划对数据进行压缩。
- 当 T2 与 T3 数据保存在 Amazon S3 中时：
  - 使用高基数键进行数据分区。Presto、Apache Hive 与 Apache Spark 都提供这项功能，可显著提高该键上的查询过滤器性能。
  - 利用与各类常见过滤器查询相匹配的辅助键对各分区内的数据进行排序。如此一来，查询引擎可以跳过无关文件并更快获取到所请求的数据。

批量数据处理

大多数分析类应用程序都需要频繁的批处理过程，例如使用预先汇总的结果更新数据存储，以使最终用户的表查询操作变得更快、更简单。批处理系统必须能够根据数据规模进行灵活伸缩，并与待处理数据集的大小等比例增长。

数据源和数据量的快速扩展要求批量数据处理系统既灵活又不受影响。从业务需求的角度来看，批处理数据作业可能受到 SLA（服务等级协议）的约束，或者需要考虑一定的预算阈值。这些要求的具体情况，将直接决定整个批处理架构的实际特征。在 AWS，包括 Amazon EMR、AWS Glue 及 AWS Batch 在内的多项服务，都能帮助用户以可动态伸缩的方式通过 EC2 实例或全托管环境运行分布式计算框架。在批处理环境中，Amazon EMR 能够面向 Amazon S3、NoSQL on DynamoDB、SQL databases on Amazon RDS、Amazon Redshift、Hadoop 分布式文件系统（HDFS）等进行数据读写操作。此外，您还可以选择 Spark、MapReduce 及 Flink 等流行框架将工作负载分发至动态扩展集群当中。AWS Batch 可利用部署在动态可扩展容器计算基础设施上的 Docker 容器运行独立作业或作业阵列。使用 AWS Glue，您可以直接运行 Spark 作业而无需管理任何 EC2 实例。

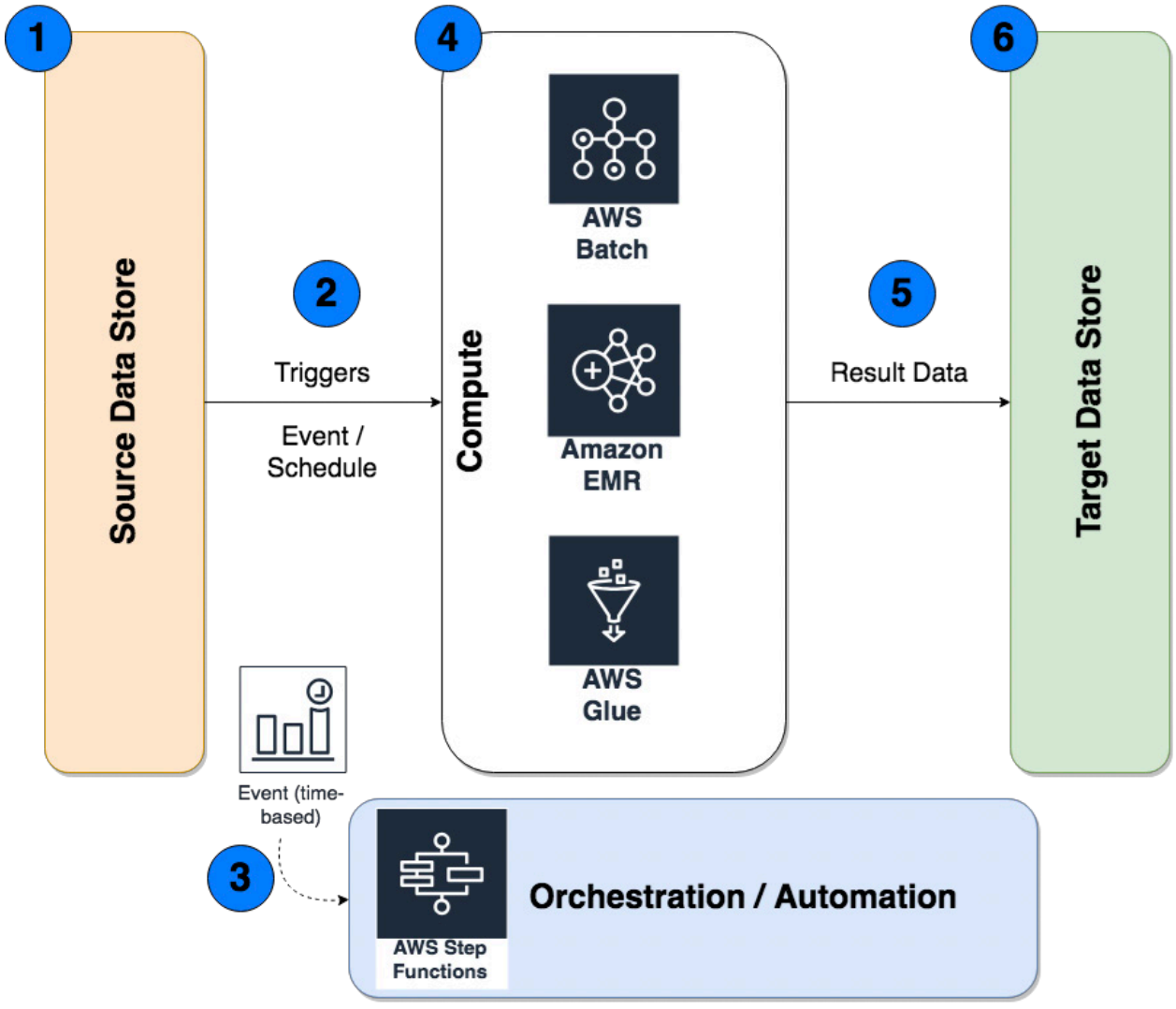
通过对批处理计算环境之外的外部来源进行数据读取与写入，得以将存储资源与计算资源解耦。以此为基础，您只需要在

当前作业被调度是检查正在运行的 Amazon EMR、AWS Batch 以及 AWS Glue 资源。这种特征，也标志着批处理系统的传统范式开始发生转变。现在，只有在实际处理数据时，系统才临时运行计算资源。Amazon EMR 与 AWS Batch 等服务也与 EC2 竞价实例高度集成，您可以按需求选择 EC2 竞价实例，借此获得远低于 EC2 按需实例的资源使用成本。

特点

- 您希望创建一套批量数据处理系统，且将集群与计算资源的管理开销控制在最低水平。
- 您希望缩短业务或最终用户在运行见解查询时所花费的时间，同时帮助他们更好地理解数据内容。
- 您希望仅在需要时才运行批量数据处理计算资源，而在不需要时立即将其关闭。

参考架构



图三：高级批量数据处理架构

1. 批量数据处理系统往往需要使用持久性存储以存储源数据，这一点对于提升系统可靠性非常重要。将源数据集保存在持久存储内时，您可以在发生故障后重试处理作业，并在后续随时查询数据集以进一步挖掘数据价值。在 AWS 上，我们为您提供丰富的源数据存储选项。作为托管数据库与存储服务，Amazon S3、Amazon RDS、Amazon DynamoDB、Amazon EFS、Amazon Elasticsearch Service、Amazon Redshift 以及 Amazon Neptune 都可以作为源数据存储方案使用。此外，您还可以选择在 Amazon EC2 或 EBS 上运行自己的数据库或存储解决方案。关于这些选项的更多详细信息，请参阅“数据湖”与“为分析类负载构建高效存储层”部分的内容。

2. 批量数据处理系统应该实现自动化，并通过调度保证可靠性、性能效率与成本优化。您可以使用 Amazon CloudWatch Events 根据计划（例如每天一次）或者事件（例如当有新文件上传）来触发对应的下游作业。

3. 批量数据处理作业通常涉及多个步骤——其中某些步骤可能按序进行，也可能并行执行。通过编排服务（例如 AWS Step Functions），您可以轻松针对或简单、或复杂的处理作业建立起自动化工作流。AWS Step Functions 可通过可视化工作流构建起分布式数据处理应用。在 AWS Step Functions 工作流中，您可以使用 Lambda 函数与原生服务集成在 Amazon EC2 或本地设施上触发 Amazon EMR 步骤、AWS Glue ETL 作业、AWS Batch 作业、Amazon SageMaker 作业以及其他自定义作业。

4. AWS Batch、AWS Glue 以及 Amazon EMR 为您的特定用例提供适合的批处理作业托管服务与框架。根据实际作业类型，您可以选择不同选项。对于能够在 Docker 容器中运行的简单作业（例如视频媒体处理、机器学习训练以及文件压缩），AWS Batch 可帮助用户轻松快捷地将作业以 Docker 容器的形式提交至 Amazon EC2 上的容器计算基础设施。对于 PySPark 或者 Scala 中的 Apache Spark 作业，您可以使用 AWS Glue，此服务可在全托管 Spark 环境中运行各类 Spark 作业。对于其他规模较大的并行处理作业，您可以选择 Amazon EMR，它能够在 VPC 内的 Amazon EC2 实例之上运行 Spark、MapReduce、Hive、Presto、Flink 以及 Tez 等多种流行框架。

5. 与源数据存储类似，批处理作业也需要可靠的存储以保存作业输出及结果。您可以使用 AWS SDK 与 Amazon S3 及 DynamoDB 进行交互，并使用通用文件协议及 JDBC 连接将结果存储在文件系统或者数据库当中。

6. 结果数据集往往也需要持久保存，以备后续通过可视化工具（例如 Amazon QuickSight、API 以及基于搜索的查询）进行访问。根据访问模式的不同，您可以选择最适合当前用例的数据存储解决方案。关于这些选项的更多详细信息，请参阅“数据湖”与“为分析类负载构建高效存储层”部分的内容。

配置说明

1. 通过批处理作业准备出大规模批量数据集，以供下游分析使用。对于体量极大且高度复杂的数据集，您可能需要简化实际查询方式，帮助最终用户与分析师更轻松地获取数据内容。如若不然，用户可能会在查找简单聚合的操作中，发现原始数据的查询难度过大。例如，您可能需要提前将前一天的销售数据整理为每日销售摘要视图。这类经过整

理的表往往包含相对较少的行与列，可帮助业务用户轻松便捷地完成数据查询。

2. 避免将批处理直接迁移至 AWS。如果直接将传统批处理系统迁移至 AWS，可能导致 Amazon EC2 实例的资源出现过度配置问题。例如，本地部署中的传统 Hadoop 集群往往被过度配置且长期处于空闲状态。您应使用 AWS 托管服务（例如 AWS Glue、Amazon EMR 以及 AWS Batch）来简化您的架构，借此消除集群及分布式环境管理等繁重的日常工作。利用现代批处理架构，您可以利用 AWS 提供的各项服务将存储与计算资源剥离开来，借此消除闲置计算资源或未得到充分利用的磁盘存储空间，最终显著降低运营成本。此外，使用针对特定批处理任务进行优化的 EC2 实例类型（而非通用型持久集群），您也可以进一步提升性能表现。

3. 尽可能使用自动化与编排工具。在传统批量数据处理环境中，系统最佳实践往往要求自动实现作业的执行与调度。在 AWS 中，您同样需要实现批量数据处理作业的自动化与编排，结合 AWS API 以启动及关闭整个计算环境，确保仅为实际使用的计算服务资源付费。例如，当作业调度开始时，工作流服务（例如 AWS Step Functions）将使用 AWS SDK 来配置新的 EMR 集群、提交作业并在任务完成后关闭该集群。

4. 使用竞价实例保存较为灵活的批处理作业。对于相对较为灵活（可以随时重试）的批处理作业，您可以选择竞价实例并将数据与计算资源拆分开来。使用 EMR 与 AWS Batch 中的 Spot Fleet、EC2 Fleet 以及竞价实例等功能管理竞价型实例。

5. 持续监控并改进批处理系统。随着数据源数量的增加，批处理系统需要快速发展、创造新的批处理作业，并启动新的批处理框架。通过指标、超时与警报，您将更从容地对作业进行检测、获取运行状态与分析见解，并最终为批量数据处理系统的进一步改进做出明智决策。

流数据摄取与处理

在对实时流数据进行摄取与处理时，必须在可扩展性、可靠性及低延迟等方面建立严格保障，才能实际支持各类应用。常见的应用场景包括活动跟踪、交易订单处理、点击流分析、数据清洗、指标生成、日志过滤、索引编制、社交媒体分析、物联网设备遥测与计量等等。这些应用场景随时可能出现资源需求峰值，每秒事件处理量快速提升至数千之巨。

借助 AWS，您可以直接使用 Amazon Kinesis 提供的托管流数据服务，也可以在 Amazon EC2 实例当中部署并管理自己的流数据解决方案。关于在 AWS 上部署流数据服务的更多详细信息，请参阅本文中的“定义”部分内容。

对于实时摄取并连续处理流数据的管道，我们需要在设计过程中考虑如下特点。

特点

- **可扩展：**对于实时分析，您需要规划一套具备弹性的基础设施，能够适应流数据的速率变化。规模伸缩通常由负责监控分片及分区数据处理指标的管理应用负责执行。我们希望管理应用能够自动发现新添加的分片或分区，并将其公平分配给所有可用工作节点进行处理。

- **持久性：**实时流数据系统应提供高可用性与数据持久性。例如，Amazon Kinesis Data Streams 可跨三个可用区进行数据复制，从而提供流数据应用所需要的高持久性。

- **可重播读取：**流处理系统应提供按序记录，并能够以相同的顺序支持多个消费程序实现记录读取或重播。

- **容错性、检查点与重播：**检查点是指记录着流数据中已使用及已处理数据记录的最近节点。如果应用程序崩溃，系统应该能够通过检查点恢复流数据读取（而非从头开始）。

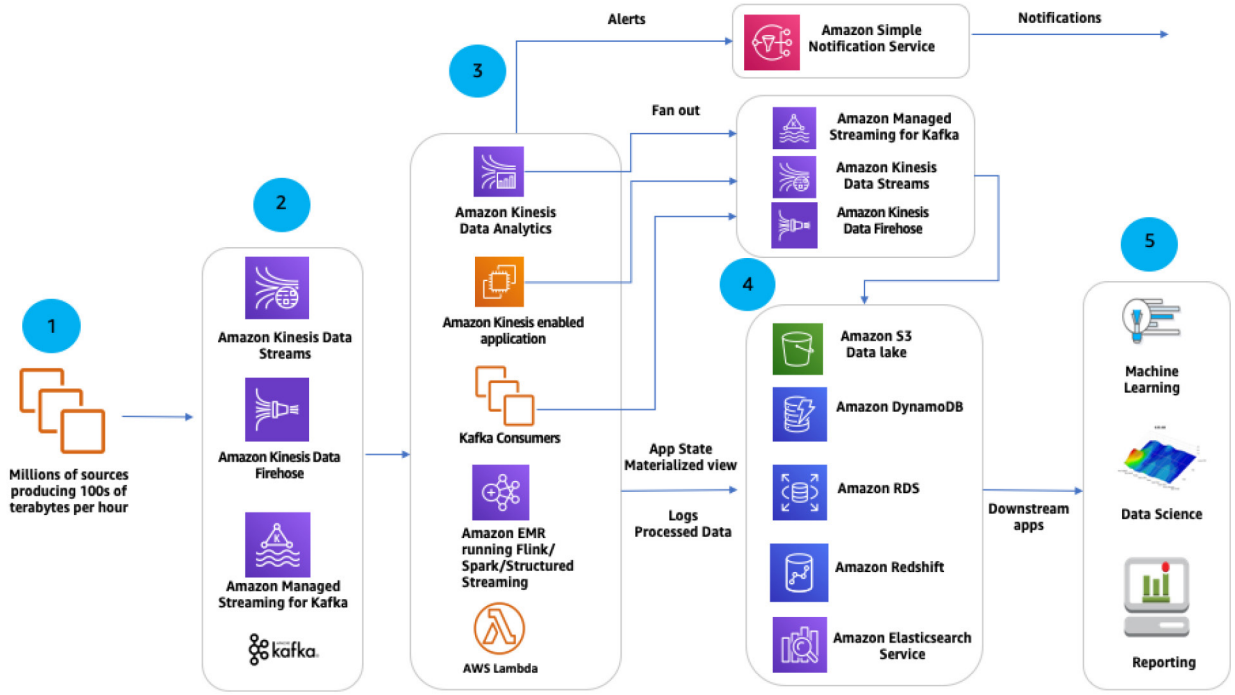
- **并行启用多个处理应用：**作为流处理系统的基本特征，其必须支持多个应用程序同时使用同一流数据。例如，您的某一应用负责更新实时仪表板，另一应用则将数据归档至 Amazon Redshift。在这类场景下，我们需要两款应用能够同时并各自独立地使用同一流数据中的数据。

- **消息语义：**在分布式消息系统当中，各个组件有可能发生独立故障。一旦出现此类故障，不同消息收发系统会在生产程序与消费程序之间实现不同的语义保证。下面来看几种最常见的消息交付保证实现：
  - 最多一次：消息无法重新发送，或者因丢失而永远无法重新发送。
  - 至少一次：消息可能向消费程序发送多次。
  - 发送一次：消息只被发送一次。

  根据实际应用需求，您还要选择一套支持必要语义的消息收发系统。

- **安全性：**在默认情况下，流数据的摄取与处理系统必须具备安全保障。您需要限制对流 API 及基础设施的访问操作，并对静态及传输中的数据进行加密。使用 Kinesis Data Streams，我们可以确保只有账户及流数据的所有者能够访问他们创建的 Kinesis 资源。Kinesis 还支持用户身份验证机制，借此控制对数据的访问活动。您可以通过 AWS IAM 策略有选择地向用户及用户组授予权限，也可以使用 HTPPS 协议通过 SSL 端点面向 Kinesis 安全地存放 / 获取数据。在运行 Apache Kafka 的情况下，为了保证 Kafka 集群的安全性，您还须部署 HTTPS、维护证书颁发机构并配置 Kafka 实例以使用 SSL 对传输数据进行加密。

参考架构



图四：流数据分析参考架构

1. **数据生产者：**多个生产者连续生成的数据每天可能达到 TB 级别的数据总量。生产者可以使用 Kinesis Agent（这是一个独立的 Java 软件应用程序）来收集数据并将其发送到 Amazon Kinesis Data Streams 或 Amazon Kinesis Data Firehose。该代理会持续监控一组文件，并将新数据发送至交付流当中。代理还负责处理文件轮替、检查点机制并在发生故障时执行重试与再次发送，保证以可靠、及时且简单的方式处理您的数据。如果您使用的操作系统与 Kinesis 代理不兼容，也可选择 Kinesis Producer Library（简称 KPL）以实现对 Kinesis 流数据的高写入吞吐量。KPL 是一套易于使用且配置灵活度极高的库，可帮助您向 Kinesis 流数据写入数据。它能够在生产程序应用代码与 Kinesis Data Streams API 活动之间充当中介。此外，生产者还可以使用 Kafka Producers 将消息发送至 Kafka 集群。

2. **流摄取：**Kinesis Data Streams 与 Kinesis Data Firehose 能够摄取并处理大规模数据记录流。如果您已经在使用 Apach Kafka，请选择 Kinesis Data Streams 进行快速、连续的数据获取和聚合，数据获取与处理将拥有实时级别的响应速度。使用 Kinesis Data Firehose（一项全托管流服务），您可以将实时流数据转换并将其发送到目的地，如 Amazon S3、Amazon Redshift、Amazon Elasticsearch Service（简称 Amazon ES）和 Splunk。

如果您使用 Apache Kafka， 您可以将 Kafka 集群部署在 Amazon EC2 实例中以获得高性能、可扩展的流数据摄取解决方案。AWS 提供多种与 Kafka 部署良好匹配的不同实例类型与存储方案供您选择。此外，您也可以使用 Amazon Managed Streaming for Kafka（简称 Amazon MSK）在 Apache Kafka 上构建并运行生产级应用程序，整个流程无需任何专业的 Apache Kafka 基础设施管理知识。

3. **流处理：**您可以使用各种服务，在滑动时间窗口上按记录顺序递增地处理实时流数。
   例如，在使用 Kinesis Data Analytics 时，您可以使用标准 SQL 以无服务器方式处理并分析流数据。此项服务将帮助您快速编写出面向流数据源运行的 SQL 代码，借此执行时序分析、建立实时仪表板以及其他实时指标。以此为基础，您可以通过配置将 SQL 查询结果扇出至外部目标，例如 Kinesis Data Firehose 或 Kinesis Data Streams。

   在使用 Kinesis Data Streams 进行数据摄取时，您可以使用 Kinesis Client Library（简称 KCL）开发出消费应用程序。您当然也可以选择 Kinesis Data Streams API 从 Kinesis 数据流中获取数据，但我们建议您使用 KCL 提供的设计模式与代码。与之对应，您可以选择 Kinesis Data Streams 作为 Lambda 函数的全托管事件源。

   目前另一种主流的流数据处理方法，则是使用 Kinesis Data Firehose。Kinesis Data Firhose 可以调用 Lambda 函数对传入的源数据进行转换，并将转换完成后的数据发送至目标处。在创建交付流时，您可以直接启用 Kinesis Data Firehose 进行数据转换。

   如果您在 Hadoop 环境中工作，则可以使用多种选项（Spark Streaming、Apach Flink 或者 Structured Streaming 等）处理流数据。消费程序可以使用多种流数据摄取解决方案（例如 Kinesis Data Streams、Amazon MSK、或者 Amazon EC2 上启用了 Apache Spark Streaming 的 Apache Kafka）对实时数据流进行可容错流处理，并配合 Spark SQL（支持通过 Spark 代码执行关系查询）构建起同时兼容实时与批量数据处理的单一架构。

   Apache Flink 是一款流式数据流引擎，可用于对高吞吐量数据源进行实时流处理。Flink 还支持乱序事件的事件时间语义、发送一次语义、反压控制以及针对流 / 批处理应用程序进行优化的 API。此外，Flink 还提供面向第三方数据源的连接器，可支持 Amazon Kinesis、Apache Kafka、Amazon ES、Twitter Streaming 以及 Cassandra 等。Amazon EMR 目前以 YARN 应用的形式支持 Flink，您可以对 Flink 资源与集群内的其他应用资源进行统一管理。Flink-on-YARN 还支持用户提交即席 Flink 作业；或创建一套长时间运行的集群，用于接收多项作业并根据 YARN 的总体容量配额进行资源分配。

   如果将 Apache Kafka 作为流数据源，您还可以在 Amazon EMR 上使用 Spark Structured Streaming。Structured Streaming 是基于 Spark SQL 构建而成的一套具备良好容错性的流处理引擎。

4. **警报、消息扇出与存储：**可以将处理后的流数据馈送至实时预测分析系统当中，据此得出推理结论，这些结论又可进一步被 Amazon SNS 用于触发警报。当然，我们也可以将处理后的消息分发至 Kinesis Data Streams、Kinesis Data Firehose、Kinesis Data Analytics 或者 AWS Lambda，借此生成新的流并完成后续处理。在管道下游，应用程序能

够以流数据为基础执行简单的聚合，并将处理后的数据发送至 Amazon S3。其他常见使用模式还包括将实时数据存储在 Amazon Redshift 中以供复杂分析，存储在 DynamoDB 中以查询事件，或者存储在 Amazon ES 中以实现全文搜索。

5. **下游分析：**使用流技术完成即时处理的数据可以持久保存，并成为实时分析、机器学习、警报及其他自定义操作的必要素材。

配置说明

1. 首先聚合记录，而后将结果发送至 Kinesis Data Streams。在使用 Kafka 时，请确保在生产程序一侧进行消息收集。
2. 在配合 Kinesis Data Streams 时，请使用 KCL 以进行记录反聚合。KCL 负责处理与分布式计算相关的多种复杂任务——例如跨多实例间负载均衡、响应实例故障、为处理后的记录保存检查点、以及对重新分片操作做出反应等。
3. 使用 AWS 竞价实例，保证以经济高效的方式处理流数据。您也可以将 AWS Lambda 与 Kinesis 配合使用，或者在 AWS Lambda 上使用 Kinesis 记录聚合与反聚合模块。
4. 使用 Amazon CloudWatch 监控 Kinesis Data Stream 指标。如此一来，您可以获得基础的流层级指标以及分片层级指标。

Lambda 架构

分析类 Lambda 架构模式的作用，是将庞大数据流与批处理系统整合到聚合视图当中。传统的批处理与流处理方法为了保证工具的通用性，往往在性能方面做出牺牲；如果使用其他特定于系统的传统方案，虽然不致降低性能、但却会提高管理难度，且要求使用者具备特定的软件专业知识。

包括 Amazon EMR、Amazon Kinesis Data Streams、Amazon Kinesis Data Firehose、Amazon Kinesis Data Analytics、AWS Glue 以及 Amazon Athena 在内的多项 AWS 服务，可帮助您轻松将流处理与批处理结果汇总至单一服务层内供最终用户使用，且无需管理任何复杂的基础设施或集群环境。

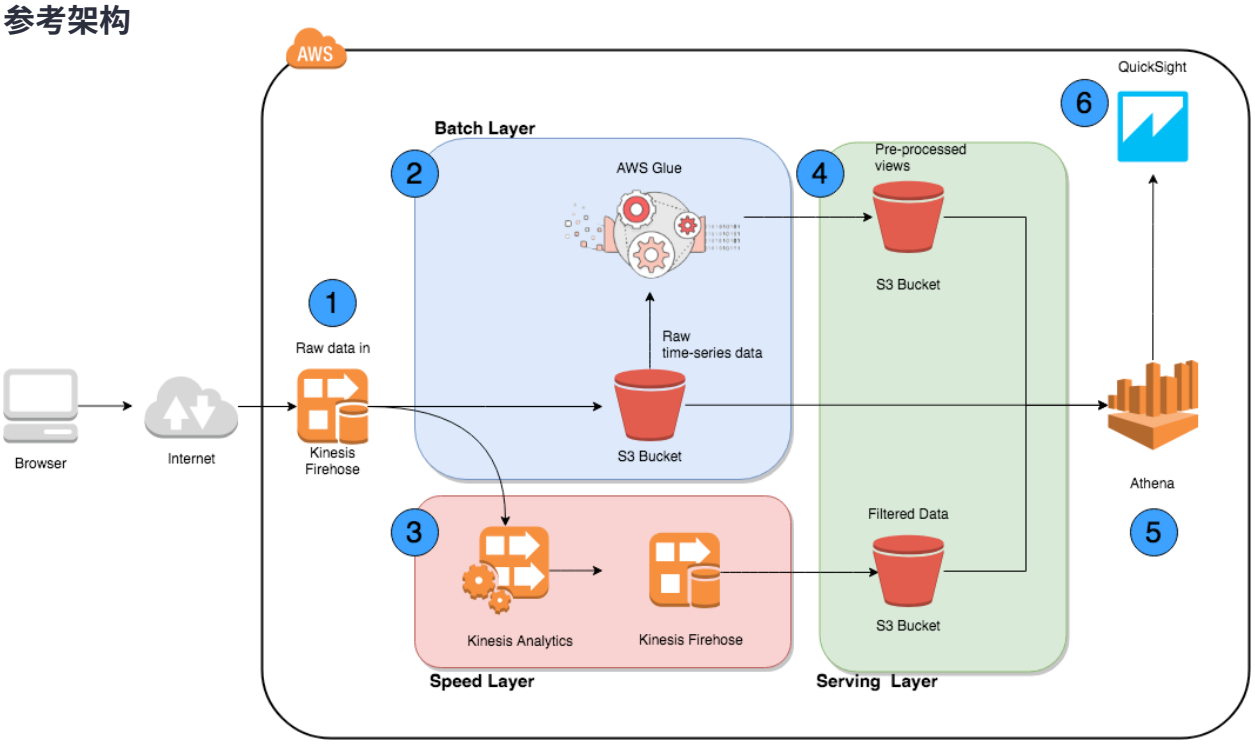
特点

1. 对于相互关联的批处理数据与流数据，您需要将二者集成起来，借此为业务带来速度更快的见解获取能力。

2. 您可能已经拥有批处理数据与流数据合并方案，但整个过程非常缓慢、极度复杂或者难以管理（需要同时管理运行在 Amazon EC2 实例上的多个开源堆栈，例如 Apache Kafka、Apache Hadoop 等）。

例如：流数据来自物联网传感器设备，批处理数据来自设备设置数据库，我们需要将二者结合起来。

参考架构



图五：Lambda 架构

请注意，这套架构并未使用任何 Amazon EC2 实例来运行集群 / 分布式流 / 处理框架。在批处理层中，我们选择 AWS Glue 作为无服务器 ETL 方案，并使用 Spark Jobs 执行批处理。Glue 能够从实时数据库中提取数据，将这些数据交由批处理层处理，从而将实时数据库数据与流数据结合起来以创建复杂的预处理视图。在速度层（Speed Layer）中，Amazon Kinesis Data Analytics 从 Amazon Kinesis Data Firehose 中实时读取流数据，您可以在这里将流数据和 Amazon S3（上一步提取自实时数据库的数据）上的参考数据做连接。这里，我们使用 Amazon Kinesis Data Analytics 实现实时的数据连接、数据过滤或运行机器学习异常检测算法，并将结果保存在 Amazon S3 中。Amazon S3 充当服务层，Amazon Athena 与 Amazon QuickSight 则负责查询经过处理后的数据。

1. **流数据生产者：**连续生成的数据每天可能达到 TB 级别，这些数据将被收集并发送至 Kinesis Data Streams 或 Kinesis Data Firehose。

2. **批处理层：**使用 AWS Glue，您能够以批处理方式分析来自 Amazon S3 的原始数据，对照历史数据查看特定时间段内的数据集，并将结果再将保存至 Amazon S3 内。
3. **速度层：**使用 Amazon Kinesis Data Analytics，您可以分析并过滤数据，借此实时检测各类异常情况。
4. **服务层:** 对于 Amazon S3 中提供的原始数据、经过预处理的视图以及实时过滤数据，您可以在服务层内进行直接查询。
5. Amazon Athena 在 Amazon S3 之上提供无服务器 SQL 查询，借此支持可视化、仪表板与即席查询。
6. 最后，将 Amazon Athena 与 Amazon QuickSight 配合使用以查询并可视化数据，进而构建起可与组织内其他 Amazon QuickSight 用户共享的仪表板。

配置说明：

1. **尽可能使用无服务器计算与分析服务，借此消除分布式与集群环境带来的无差别繁重管理负担。**在 AWS 中，您可以对 Kinesis Data Firehose、Amazon Kinesis Data Analytics、AWS Glue、Amazon S3、Athena 以及 QuickSight 等各类服务进行全面管理，且无需分神于补丁修复、软件安装及备份等运营事务。
2. **在无服务器范围之外，充分使用 Amazon EMR 中的各类框架。**结合实践，您可能会发现 Apache Flink 或 Spark Streaming 等应用往往更适合速度层计算的使用场景。您可以利用 Amazon EMR 管理这类应用的集群环境。
3. **在正确的时间合并正确的数据。**Lambda 架构不可能满足所有用例的实际需求。我们应该从实际需求出发，通过不同方案满足服务层视图对于速度与数据更新能力提出的要求。
4. **为最终用户创建预处理视图。**对于部分用户，直接访问原始流数据与主数据可能太过复杂，这会造成用户被大量数据困扰、严重时会极大拖慢业务的正常执行效率（例如未经过滤、面向所有列进行 SQL 查询）。因此，您可以利用预处理视图（在其他域中也被称为物化视图）简化查询流程并提高性能表现。同时根据业务预期调整视图的更新频率。

关于 Lambda 架构的更多详细信息以及上手教程，请参阅博文：《无服务器架构：使用大数据 Lambda 架构实现统一的实时与批量分析》。

数据科学

在典型的精简数据科学工作流程中，数据科学家们主要负责准备机器学习所需数据、使用训练数据集进行算法训练、针对独立的验证数据集评估算法性能、完善数据准备与算法以重新训练，最后将算法打包进一套生产级部署框架当中。

AWS 为机器学习场景下的服务提供多种抽象级别。在抽象级最高的层级上，AWS AI 服务（例如 Amazon Polly、Amazon Lex、Comprehend 以及 Amazon Rekognition）负责提供经过预训练的 API 终端节点，用于根据输入数据进行推理。而在抽象服务的中间层的平台服务（例如 Amazon Machine Learning、Amazon SageMaker、EMR 内运行的 SparkML 以及用于数据标记的 Mechanical Turk）负责在加快算法开发的同时，帮助用户摆脱由基础资源管理带来的日常负担。最后，抽象度最低的层级涵盖各类框架与基础设施，提供 CPU 优化与 GPU 优化的 AWS Deep Learning AMI 并内置了各类框架（包括 TensorFlow、PyTorch 与 Apache MXNet 等经过预配置的流行框架）。具体选择哪一种机器学习堆栈，主要取决于您当前面对的业务问题、以及开发团队的知识与经验水平。

受篇幅所限，本文无法具体介绍数据科学算法的开发细节，下面仅简单解释目前机器学习领域几种常见的大数据准备、捕捉与使用方法。

在大多数业务环境中，数据被存储在集中存储库或者分布在多个存储库内。此存储库可以是 Amazon S3、EMR 内的 Hadoop 框架、RDS 等关系型数据库、DynamoDB 等 NoSQL 数据库、数据仓库或者多种存储方案的组合（例如与 Hive 相兼容的元存储）。但在业务流程中直接收集到的原始数据往往太过粗糙，必须经过清洗才能供机器学习应用使用。大多数机器学习算法都会对数据素材提出相当严格的要求，例如避免出现空值。对数据科学家们来说，日常工作中的大部分内容是为机器学习模型准备数据。相关最佳实践建议我们开发出健壮且自动化的数据管道，可将企业收集到的原始数据聚合起来，并转换为可支持机器学习模型的数据集。数据集的不断扩展往往推动模型性能达到新的高度，具体扩展方法则包括数据列规范化或组合、文本语料库分词、或者将视频预处理为单一帧以进行特征工程等等。

在 AWS 上，Glue 作业、EMR 以及 Data Pipeline 等托管 ETL 服务显著减少了数据转换负担，同时为用户提供多步骤数据转换流程编排管理服务。通过将 AWS 基础设施视为可直接调用的代码，同时配合 AWS API 终端节点进行服务交付，数据科学家们得以轻松开发出清洗管道并实现版本控制，也能在此期间总结出有助于其他数据科学家的最佳实践。

通过管道依次进行分段、清洗与特征工程，将帮助我们极大提升模型的开发速度。与仅面向较小型数据集的特征工程任务相比，从大规模原始数据集中清理和采样到一套更小的干净数据集通常需要更多的计算量和内存。根据架构完善原则中提出的性能效率支柱，我们应针对管道中的不同阶段采用适当的实例类型。对于数据科学家们来说，在处理 ETL 清洗管道的末端，最佳实践是将清洗后的数据存储在独立的数据存储当中，例如 Amazon S3。这样面向机器学习模型开发场景的独立数据存储方案能够保证模型训练与模型运行之间的数据一致性，进而为评估模型改进提供稳定的底层基准。而另一项架构完善支柱——成本优化支柱——建议我们将计算与存储资源区分开来，借此实现成本优化。

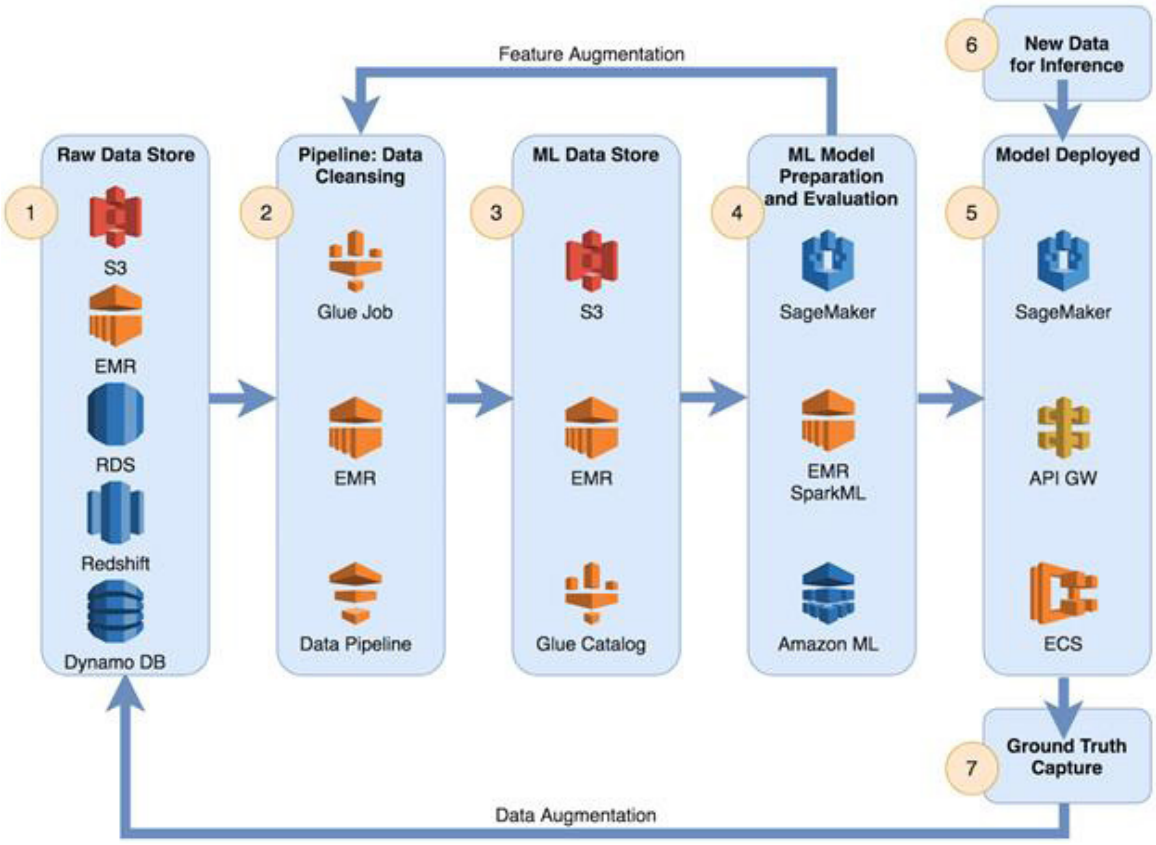
为了解决实际业务问题，我们需要部署机器学习模型以支持新数据的推理工作。另一项最佳实践建议我们通过使用特定于机器学习上下文的额外附加标记的数据来扩展原始数据，以便跟踪和改善模型的性能，这项操作被称为“数据增强（data

augmentation）”。数据增强的具体操作，包括记录新数据、推理以及实际数据点中的“基准真相（ground truth）”。例如，这可以帮助我们判断某些推理出的“最佳”广告是否针对特定客户群体得到了销售转换率的提高。在通过重复训练衡量并改善模型性能方面，基准真相数据具有无可比拟的价值。但在某些情况下，我们往往无法在特定时间段内获取基准真相，因此需要采取将推理与结果相关联的跟踪方法。

特点

- 您希望使用机器学习来解决业务问题。
- 您拥有大量数据，而这些数据间的关系很难通过传统仓库工具实现可视化。
- 您希望确定稀疏数据集中的关系与趋势，例如实现产品推荐功能。
- 您希望根据历史数据做出预测，例如预测用户流失情况。
- 您希望实时改善业务功能，例如投放最有利于销售转化的广告内容。
- 您希望实时改善业务功能，例如根据传感器数据预测零件故障。

参考架构



图六：数据科学管道参考架构

1. 原始数据存储。原始数据存储通常是指数据湖中用于摄取与存放数据的存储位置。
2. 数据清洗管道负责将原始数据转换为可供机器学习直接使用的数据集。
3. 机器学习数据存储将用于存放训练数据于另一存储库内，例如写入至 Amazon S3、HDFS 存储或者元存储中。
4. 机器学习模型的开发涉及多种框架，包括 Amazon SageMaker、SparkML on EMR、运行 Deep Learning AMI 的 GPU 实例，以及 R、Python 或 Scala 等语言。
5. 将机器学习模型打包至生产端点当中，其通常位于推理 API 之后。
6. 在生产环境中，向机器学习部署方案提供新数据，并使用其推理结果解决业务需求。
7. 在生产部署期间捕捉到的数据将配合适当标记再统一进行存储，管道将定期使用这些数据重新训练模型。

配置说明

- 我们需要经常清除机器学习数据中的缺失字段、标记数据内容、对特征进行“工程设计”，借此减少或合并无关的列 / 特征。此外，机器学习训练通常只需要全部可用数据中的一个子集，因此业界通常会选择下采样方法以加快模型训练速度，进而缩短学习 / 改进周期。
- 在模型准备期间，数据一致性同样至关重要。只有保证一致性，我们才能将更新后的模型与之前的版本进行准确比较。训练数据集与“保留（hold-out）”或验证数据集可以分别存储，供后续用于模型训练与评估。
- 在计算资源的成本优化方面，请选择正确的实例类型进行训练与推理。例如，GPU 优化型实例可以加快神经网络模型的训练速度，而通用型实例则更适合运行 NN 模型中的 API 推理。
- 通过扇出资源（例如 Amazon SageMaker 或 SPOT 市场资源）使用不同的调整参数（即超参数优化）同时为单一模型训练出多个版本，这能极大加快模型的开发周期。
- 数据科学家们往往拥有自己的首选模型开发方法。Jupyter Notebook、Zeppelin 或者 R 等笔记型 IDE 可在用户的本地浏览器中提供 IDE 开发环境，其中的实际命令则由 AWS 云内的远程资源负责执行。
- 尽量避免在云端与数据科学家的本地工作站之间频繁移动数据集。目前的首选模式之一，是将所有数据保存在 Amazon S3 当中，同时保留 ETL 管道以根据开发需求加载数据。
- 使用 API Gateway、Amazon SageMaker 或者 ECS 部署您的模型，即可将传入流量路由至机器学习的不同版本，保证在全部流量指向新版本之前验证模型的生产性能。

多租户分析

多租户架构允许组织在多位用户 / 多方之间共享资源，同时提供控制机制以保证各租户间的隔离性与安全性。在分析场景中，多租户设计能够实现一系列现实用例，包括通过工作负载支持数据湖上的多个部门、测试同一应用程序的多个版本及 / 或使用不同数据集测试同一应用程序，借此充分利用昂贵的集群资源。另外，由于资源与相关维护成本将由多个租户共同分担，因此多租户设计往往具有更好的成本效益。

特点：

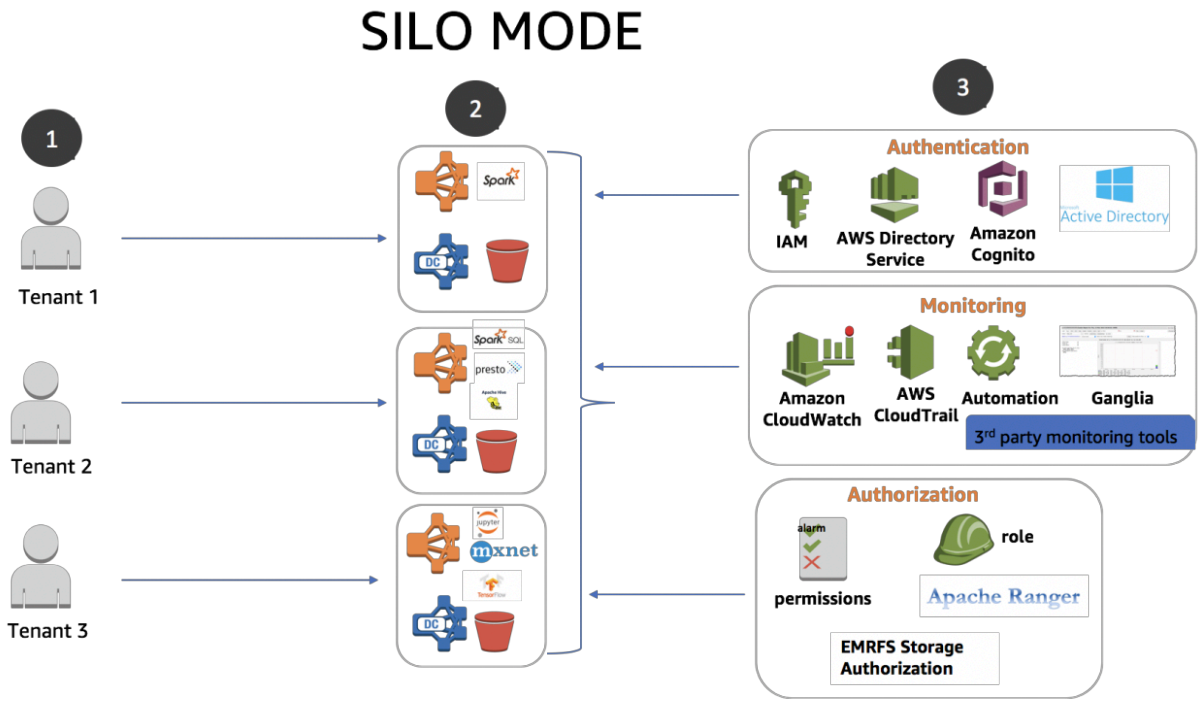
资源隔离：在多租户环境中，我们应对各租户消耗的资源进行逻辑隔离。这样可以确保单一租户的分析操作不会干扰或阻碍到其他租户的任务。

数据安全：租户的数据资产可以存放在多种数据存储内。通过以下两种机制，我们可以将租户的数据彼此隔离：1）每租户独立 schema 及 / 或 数据库；2）在数据库设计中考虑多租户共享需求，并通过代表唯一租户的列对数据进行分区标记。无论选择哪种方法，我们都应采取身份验证、授权与审计等方法保障数据安全。

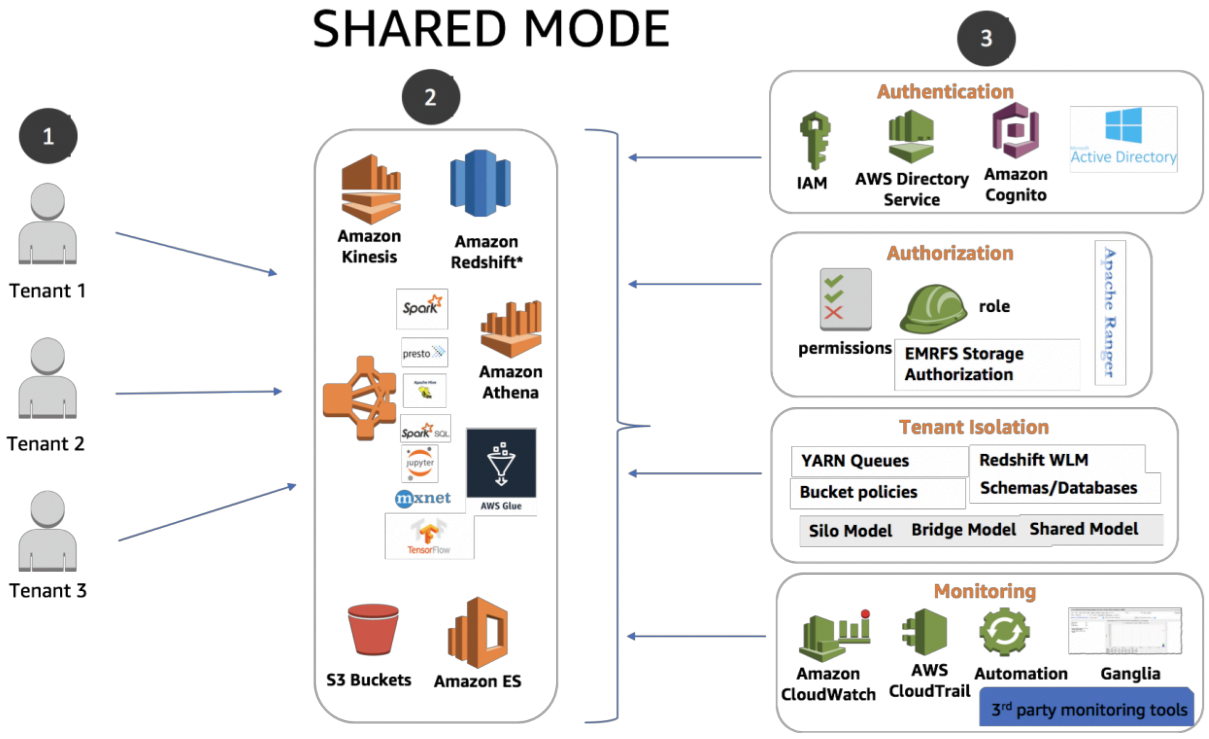
计量使用：租户应仅为自己在多租户分析环境中实际使用的存储与处理资源付费。这就要求我们认真监控并测量各租户所使用的组件与子系统。

可扩展性：随着新租户的加入，多租户分析环境应具备强大的可扩展能力以满足不断提升的处理与存储需求。

参考架构：



图七：AWS 上的多租户分析参考架构（专属模式）



图八：AWS 上的多租户分析参考架构（共享模式）

在多租户分析架构中划分租户的使用情况与数据时，我们通常使用两种基本模式。其中每一种模式都将带来非常独特的租户数据管理、访问及拆分方法。

孤岛 / 专属模式：每个租户都拥有自己的专用资源，且数据与其他租户数据完全隔离。在逻辑角度上，所有用于表示租户数据的结构都将“唯一”于该客户，意味着每位租户通常使用不同的表示、监控、管理与安全保护方法。

1. 租户——需要资源运行各自操作的数据工程师、分析师以及数据科学家等。
2. 每个租户将启动自己的集群。数据工程师可能会在其中安装 Spark 及 Hive 等工具，并将处理后的结果保存在 Amazon S3 当中。分析师可能会运行 Spark SQL 及 Presto 等工具以探索数据集，并将查询结果发送至自己的 S3 存储桶内。数据科学家们则往往使用 EMR 集群运行自己的机器学习或深度学习框架。
3. 根据所选分析工具的不同，采用不同的身份验证与隔离模式

共享 / 池模式：这是一种真正的多租户模式，其中各租户共享分析系统内的全部基础设施。租户数据被存放在统一的公共数据库内，所有租户共享共同的数据表示形式（schema）。这就要求引入分区键，用于确定并控制指向租户数据的访问操作。租户之间通过队列与工作负载管理模块等设计实现集群分析资源的共享。这种模式旨在简化 SaaS 解决方案的交付、管理与更新方式，同时又继承了 SaaS 供应商引以为傲的持续交付与敏捷性优势。

1. 租户——需要资源运行各自操作的数据工程师、分析师以及数据科学家等。
2. 一套安装有各类工具与框架的大型多节点集群，可为众多用户提供支持。此外，最终用户还可以使用这套基础设施启动边缘节点以运行自己的数据科学平台。虽然成本效益突出，但共享集群也面对一个令人担忧的难题——某些租户可能大量垄断资源，导致其他租户无法获得 SLA 规定的使用体验。例如，分析师可能针对 Presto 或 Hive 发出需要长时间占用大量集群资源的查询；数据科学家也有可能在这里训练一套涉及庞大数据量的模型。
3. 根据所选分析工具的不同，采用不同的身份验证与隔离模式。

配置说明：

在设计安全的多租户分析环境时，我们需要立足多个方面进行考量，具体包括：

- **租户身份验证：**多租户架构需要管理不同租户的对应身份。用户应使用 AWS IAM 或托管凭证对分析类应用程序进行身份验证。
- **数据集授权：**在通过身份验证之后，还应确保用户只能访问自己的数据集。在多租户分析场景中，这要求我们仅向用户提供访问自有数据所必需的权限，而绝对无法触及其他租户的数据。

- **租户资源隔离：**在多租户应用场景当中，每个租户都通过明确定义的分区策略与其他租户相互隔离。租户之间的隔离度取决于您选定的分区形式——专属或共享。
- **监控与指标、租户集中管理与计费机制:** 在设计多租户分析环境时，必须建立管理层以监控运营、指标、性能及计费机制，借此有效管理各租户。

架构完善的框架支柱

本章节将结合分析类应用设计场景对五大支柱进行逐一介绍，具体包括其基本定义、最佳实践、问题、注意事项以及相对应的核心 AWS 服务等。

为了简单起见，我们这里仅讨论与分析类工作负载相关的问题。在实际架构设计当中，还存在诸多本文未加提及的具体问题。关于更多详细信息，我们建议您参阅《AWS 架构完善的框架》白皮书。

卓越运营支柱

卓越运营支柱涵盖多种系统运行与监控因素，旨在不断改进流程与规程支持能力以实现业务价值。

设计原则

在云环境当中，以下几项指导性原则可用于支持卓越运营。我们建议您参考《AWS 架构完善的框架》白皮书中列出的设计原则，这些基本原则具有普适性，因此也适合本文设定的分析类工作负载场景。

定义

云环境下的卓越运营主要由以下三个方面组成：

- 筹备
- 运营
- 演进

除了架构完善的框架中所提及的相关流程、操作手册与活动日（Gameday）之外，您还应结合自身实际在特定领域提升分析类应用场景中的卓越运营能力。

最佳实践

筹备

为了在分析类工作负载中实现卓越运营，您需要深入了解管道及其预期行为。以此为基础，您可以在设计当中充分体现自己对其运作状态以及支持需求的确切理解。

在卓越运营的筹备环节中，您需要考虑以下问题：

- **运营优先级：**分析团队中通常包含数据工程师、数据分析师以及数据科学家等职能角色。各团队成员需要对分析管道、自身在团队中的角色以及团队的整体业务目标拥有统一的理解，并据此为业务事务设定优先级。此外，您还需要考虑外部法规与合规性对于数据资产及处理引擎提出的监管要求，这同样会影响到事务优先级以及管道中的具体工具选择。以运营优先级为基础，您能够将改进举措集中在成效最为显著的层面（例如增强开发团队技能、改进分析类应用的性能衡量技术、对批处理与实时管道进行自动化及编排、增强监控能力等等）。最后，请随着需求的变化而不断更新优先级。
- **运营设计：**分析管道的设计应涵盖具体的部署、更新与运营方式。良好的 CI/CD 管道能够帮助您的组织在 bug 被引入生产环境前将其发现，同时更频繁地发布功能更新。CI/CD 还能帮助开发人员编写出高质量代码、实现 ETL 作业管理流程自动化、降低风险并提高运营体系的可预测性。在 AWS 中，您可以将全部工作负载（包括应用程序、基础设施、策略、治理与运营）视为代码，并通过代码进行定义与更新。换句话说，您可以将管理应用程序代码的统一工程原则应用于堆栈内的各个元素。
- **运营就绪：**分析管道的成功实施需要涉及诸多基础步骤，我们应在每一步中充分考虑风险与收益影响。对于一套全面的运营改进计划，其中必须包含如何保存原始数据资产、如何创建管道活动记录、如何监控并调试各项功能等等。将这些问题汇总起来并整理成运营手册，可帮助我们显著提升运营质量与执行效率。此外，您还需要测试程序、潜在故障以及响应效果（例如组织活动日测试之前完成的工作，在确认无误后再进一步推进），保证一步一个脚印地踏实前行。

分析 - 卓越运营 01： 您如何选择最合适的数据分析服务和解决方案？

对您的工作负载进行分类并选择最适合您的分析服务非常重要。无论您的分析用例属于商务智能、机器学习、NoSQL 还是其他类别，AWS 都为您提供多种服务选项，可显著简化云环境下的分析工作负载。

您应根据速度 / 性能、批处理 / 实时、交互式 / 自动化以及自托管 / 无服务器 /AWS 托管等具体运营标准，为自己的用

例选择分析服务。例如，对于主要面向核心 ETL 作业的 Apache Spark 作业，您应选择 AWS Glue 以减轻 Apache Spark 集群管理负担，进而简化整体架构。而对于高度定制化工作流（例如使用 Kinesis Data Streams 进行实时流传输）或者 Amazon EC2 Spot 实例，Amazon EMR 则更为合适。此外，Amazon Redshift 与 Redshift Spectrum 是一对完美互补的服务，能够将商务智能与业务报告等查询操作延伸至 Amazon S3 数据湖当中。

分析 - 卓越运营 02： 您如何创建和部署您的数据分析管道?

要建立一整套分析管道，您需要分别为数据集获取、用于数据处理的高性能集群以及数据查询 / 可视化指定对应的服务。在筹备过程当中，您应建立起自动化的服务创建、配置和部署体系，借此实现无缝化管道运营流程。您还可以使用 AWS CloudFormation 预配置工具构建一个干净的环境。以此为基础，您不仅可以使用模板来配置基础架构，还可以在安全、隔离的环境中进行工作。通过将原有 ETL 应用转换为基于云的无服务器 ETL 架构，组织可以真正建立起一套无缝化、端到端持续集成与持续交付（CI/CD）管道，并全面涵盖从源代码到构建、到部署、再到产品交付的整个业务流程。在 CI/CD 管道的帮助下，组织能够在 bug 引发实际影响前将其发现，同时更频繁地发布更新。该管道还可帮助开发人员编写出高质量代码、自动执行 ETL 作业发布管理流程，并显著降低安全风险。

运营

所谓运营成功，是指所实现的成果切实符合您预先定义的衡量指标。通过了解分析类应用的运行状况，您可以确定该应用何时会受到运行事件的影响，并及时做出适当响应。

要实现成功运营，您需要考虑以下因素：

- 了解整个分析管道的运营状况
- 针对事件做出响应

分析 - 卓越运营 03： 您如何监控分析管道的运行状况？

您的团队必须能够轻松了解当前分析工作负载的运行状况。您需要以关键性能指标（KPI）与运营成果指标为基础，获取有指导性的见解结论。通过这些指标建立起包含业务与技术视角的仪表板，将帮助团队成员做出更明智的决策。在 AWS 上，您可以使用多种功能实现对工作负载及运营日志的汇总与分析，借此了解当前运营状态并随时间推移从运营过程中获取见解。

还有多种 AWS 托管服务（例如 Amazon RDS 与 Amazon Redshift）提供专用的服务指标，您可以将这些指标整合到 CloudWatch 仪表板当中。例如，通过监控数据仓库的每秒平均读写操作数量，您可以确定当前使用模式并相应扩展容量。其中的指标（包括队列长度 / 深度）也非常重要，能帮助我们了解在给定时间段内，等待数据仓库处理的具体查询或请求数量。我们可以将集群中的日志数据发送至 Amazon CloudWatch Logs，并建立基准指标以定义常规运营模式。此外，可以创建 Amazon CloudWatch 仪表板，通过系统级与业务级视图显示各项相关指标。对于 Amazon EMR 等处理框架，您还应安装与之配套的分布式可扩展 Ganglia 监控系统，专门负责跟踪高性能计算系统（包括集群与网格）的运行状态。

您还可以将 CloudWatch Logs 日志数据摄取至 Amazon Elasticsearch Service（简称 Amazon ES）当中，而后通过内置的 Kibana 支持功能创建运行状态仪表板与可视化结果（例如订单价格、已接入用户以及交易时间）。如 AWS 责任共担模型的定义，您还可以通过 AWS Service Health Dashboard（简称 SHD）与 Personal Health Dashboard（简称 PHD）直接获取一部分监控结果。当 AWS 发现某些可能影响业务正常运行的事件时，将通过这些仪表板发出警报以及对应的修复指导。订阅了 Business 和 Enterprise Support 支持服务的客户还可以访问 Amazon PHD API，将其集成至自己的事件管理系统当中。AWS 也通过 AWS 服务 API 与 SDK 支持多种第三方日志分析系统与商务智能工具（例如 Grafana、Kibana 以及 Logstash）。总而言之，为分析管道规划正确的资源容量绝不能纯靠猜测，而应立足于扎实可信的运营指标。

分析 - 卓越运营 04： 您如何应对数据分析应用程序的运营事件？

分析管道中涉及诸多活动部分，具体涵盖数据的摄取、存储、分析以及可视化等环节。作为数据与分析管道的组成部分，任何一环发生故障都可能对依赖于分析层的下游关键业务应用产生直接影响。为此，我们应确保针对此类事件预先建立起应急措施，保障业务连续性并满足可用性 SLA。总之，一切计划内（例如促销活动、部署与故障测试）以及计划外（例如资源利用率激增与组件故障等）的运营事件都应被纳入预期范畴。

我们还应创建并使用运营手册（runbook）及事件响应手册（playbook）以避免混乱，并按预定顺序对警报做出响应。预先定义的警报机制必须由负责响应及升级的职能角色或团队直接管理。此外，务必理解各系统组件对业务的影响，并在必要时使用这类知识确定工作的优先级。在事件发生后，执行根本原因分析（RCA），制定防止故障再次发生的应对策略并将结论整理为指导文件。在采取行动之前，我们还应明确知晓在哪些情况下需要人为做出决定，保证先权衡决定、再加以实施。当自动化流程发生故障时，还必须有可行的手动处理流程作为补充。

AWS 以编程方式提供能够全面支持工作负载与运营事务的多种工具选项，并借此简化了事件响应流程。在这些工具的帮助下，您可以编写面向运营事件的响应脚本，并根据监控数据自动触发脚本执行。在 AWS 中，您可以将故障组件替换为已知的健康版本以缩短恢复时间，随后慢慢尝试故障修复。您可以在 AWS 上通过多种方式自动执行运营手册与事件响应

手册中指定的操作。

要对 AWS 资源状态变更事件或者其他自定义事件做出自动响应，您需要通过 Amazon CloudWatch 目标（例如 Lambda 函数、Amazon Simple Notification Service（简称 Amazon SNS）主题、Amazon Elastic Container Service（简称 Amazon ECS）任务、Step Functions 或者 AWS Systems Manager Automation 等）创建 CloudWatch 规则。AWS 还通过 AWS 服务 API 及 SDK 支持各类第三方系统。目前，合作伙伴与第三方提供多种工具，可用于实现监控、通知与响应等运营功能。其中的典型解决方案包括 New Relic、Splunk、Loggly、SumoLogic 以及 Datadog 等。

分析 - 卓越运营 05：您如何在尽可能降低变更影响的同时，改进您的数据分析工作负载？

随着新技术的引入，组织通常会将数据与分析堆栈升级为更新版本，或者选择托管或无服务器解决方案替换掉原本用于数据接收、处理或可视化的服务。使用外部元数据存储可将存储与计算剥离开来；而 Amazon S3 存储桶提供的组件版本配置则可帮助用户实现集群的升级 / 重新启动与进程恢复。

如果您需要跨多项应用的共享式分析进行细粒度控制配置，可以考虑使用 AWS Systems Manager Parameter Store 提供的参数存储功能。AWS Secrets Manager 则可用于存储数据库凭证及其他“密钥”，帮助您轻松轮换、管理并检索分析类应用代码中的密钥与凭证。

演进

关于分析类应用程序中的卓越运营演进最佳实践，请参阅《AWS 架构完善的框架》白皮书。

资源

请参考以下资源，以了解我们关于卓越运营最佳实践的更多详细信息。

说明文档与博客：

- 使用 AWS Step Functions 与 AWS Lambda 编排多项 ETL 作业
- 使用 AWS Developer Tools 实现无服务器的 AWS Glue ETL 应用程序的持续集成和交付
- 使用 AWS Step Functions 实现动态 ETL 管道
- 使用 Parameter Store 正确存储密钥信息
- 教程——密钥的存储与检索

白皮书

- 卓越运营支柱

视频

- 走钢丝：在创新、可靠性与安全性之间求取平衡
- Netflix 工程师的一天 III

安全性支柱

安全性支柱涵盖对信息、系统及资产的保护能力，同时要求通过风险评估与缓解策略保障业务价值。

定义

云环境中的安全性最佳实践分为以下五种类别：

- 身份与访问管理
- 检测控制
- 基础设施保护
- 数据保护
- 事件响应

托管服务解决了目前分析类应用环境所面临的最大安全隐患，即尽可能消除了操作系统补丁安装、二进制补丁修复等不同形式的基础设施管理任务。但需要强调的是，与非托管分析类应用架构相比，托管架构的攻击面虽然有所缩小，但仍然建议大家遵循开源 Web 应用安全计划（简称 OWASP）的建议以及应用程序安全最佳实践。

本章节中提出的问题，旨在帮助大家在实际配置中发现可能被攻击者利用引发的滥用的各类错误权限配置。本章节中描述的错误实践会严重影响您云平台的整体安全性，因此您不仅需要认真验证配置方法，同时也应定期对现有配置进行检查。

本文不再另行叙述基础设施保护方面的最佳实践，相关详细信息请参阅《AWS 架构完善的框架》白皮书。

设计原则：

在云环境当中，大家可以利用多种指导原则强化您的系统安全性水平。针对分析类工作负载，以下内容尤其值得强调。关于更多详细信息，请参阅《AWS 架构完善的框架》白皮书中的设计原则部分。

- **实现数据可追溯性：**实时监控、告警并审计各项操作，以及环境中与数据沿袭相关的变更。将指标与系统集成起来，建立自动化响应与应对机制。

最佳实践

身份与访问管理

身份与访问管理属于信息安全规划当中的核心组成部分，负责确保仅通过验证及授权的用户可以以符合预期的方式访问您的资源。举例来说，您可定义各相关主体（包括用户、组、服务与角色对您帐户采取的具体操作）、制定与这些主体相统一的政策，同时实施强大的凭证管理机制。这些权限管理因素将共同构成业务系统内验证与授权的核心概念。分析类应用通常需要涉及多项采用不同访问方式与方法、且相互依赖的服务。因此，正确判定各项服务在整体环境中的服务级访问范围继承情况，也同样至关重要。

分析 - 安全 1： 您如何验证组织内部对分析类应用的访问操作？

身份验证，是指各实体（用户或应用程序）向目标应用或系统证明其身份真实性的过程。AWS 内的各项大数据服务提供多种身份验证机制，您可以根据需求选择最适合的方案选项。

AWS 身份与访问管理（IAM）支持与 Active Directory 或其他使用 LDAP 的目录服务实现联合验证。目前，众多大型组织已经在实施联合身份验证机制。联合用户使用 IAM 角色以实现面向下游服务的身份验证能力。例如，在 Amazon Redshift 中，IAM 角色可被映射至 Amazon Redshift 数据库组，Amazon EMR IAM 角色可被映射至 EMR 安全配置或者基于 Apache Range AD 组的策略，而 AWS Glue 中的 IAM 角色则可映射至 Glue 目录资源策略。

下面是各项服务所对应的部分身份验证选项。关于更多详细信息，请参阅每项服务对应的说明文档：

服务	身份验证选项
AWS Lake Formation	IAM 验证、用于数据目录访问的 Lake Formation 权限机制。
Amazon Athena	IAM 验证、用于 JDBC 连接的 SSH 密钥、联合身份、跨账户、EC2 实例配置。
Amazon Redshift	IAM 验证、原生数据库身份验证功能。
Amazon EMR	IAM 验证、LDAP 验证（面向 HiveServer2、Hue、Presto 与 Zeppelin）、Kerberos 验证、SSH 密钥、Apache Knox。

分析 - 安全 2： 在组织内部，您如何授权指向分析服务的访问操作?

授权，是指允许或拒绝某一身份执行某项操作的行为。授权过程需要首先进行身份验证，通过经过验证的真实身份确定当前申请者有权执行哪些操作。数据授权可进一步分为存储授权、元数据授权与应用程序授权。

在组织之内，各团队往往共享同一批分析类服务与应用程序。服务授权机制的选择，取决于您在用户管理方面采取“细粒度”抑或是“粗粒度”划分方法。在使用细粒度方法时，各团队可以获得公开共享资源 AWS 账户的访问权限，例如组织内共有共用的大型 Amazon Redshift 集群；而对单一数据库的访问则通过 IAM 策略级别面向各团队进行授权。相比之下，粗粒度方法直接按团队划分授权范围，且各团队负责控制各内部成员的个人账户以及账户内涵盖的工作职能 / 相关资源。至于粗粒度协作访问，需要通过其他团队提供的跨账户角色实现。例如，在粗粒度方法中，“数据湖团队”通过某一账户保持对 S3 存储桶的控制权限，而“数据工程团队”则在另一账户中使用 AWS Glue 服务。该 AWS Glue 服务通过跨账户角色授权获得指向 S3 数据湖的只读访问权限。Amazon 更喜欢采用粗粒度控制方法，借此帮助拥有大量用户的组织简化内部安全管理流程。

以下是各项服务所对应的部分存储授权选项。关于更多详细信息，请参阅每项服务对应的说明文档：

服务	存储授权选项
Amazon S3	S3 存储桶策略及 ACL 允许用户为 S3 对象定义细粒度管理权限
Amazon EMR	通过 Amazon EMR 服务中的“安全配置（Security Configuration）”为 Amazon S3 中的数据集提供 EMRFS 授权。 在 Hadoop 中启用“安全模式（Secure Mode）”。使用 Hadoop ACL。

各项服务提供的部分元数据授权选项。关于更多详细信息，请参阅每项服务对应的说明文档：

服务	元数据授权选项
Apache Hive on Amazon EMR	基于 SQL 标准的 Hive 授权。
Amazon Redshift	可使用 Apache Ranger on EC2 实现细粒度访问控制（权限于 Apache Hive）。
AWS Glue	使用用户与分组实现基于 SQL 标准的授权机制。 允许向目录项中添加细粒度访问控制策略。

应用程序授权规则，是指负责定义每个用户可对各项服务资源执行哪些具体操作的规则，具体包括运行 Glue 爬网程序、运行 Glue ETL 作业、启动或终止 EMR 集群等。应用程序授权规则需要在跨服务 IAM 中定义。以下为各项服务提供的部分应用程序授权选项。关于更多详细信息，请参阅每项服务对应的说明文档：

服务	应用程序授权选项
Amazon Redshift	用于定义哪些用户可以列出、读取、创建及管理 Amazon Redshift 资源的 IAM 策略。
AWS Glue	用于定义哪些用户可以读取并修改各类 Glue 资源（例如 Crawler、作业等）的 IAM 策略。
Amazon EMR	用于定义哪些用户可以列出并修改 Amazon EMR 集群的 IAM 策略。
Amazon Athena	被分配给用户，用于定义哪些用户可以在 Amazon Athena 中执行读取及查询操作的 IAM 策略。
Amazon Kinesis	被分配给用户，用于定义哪些用户可以创建及修改 Amazon Kinesis Streams、Amazon Kinesis Data Firhose 交付流等的 IAM 策略。
Amazon QuickSight	被分配给用户，用于定义哪些用户身为 Amazon QuickSight 管理员、作者或读取方的 IAM 策略。

检测控制

分析 - 安全 3： 您如何存储、监控并分析自己的访问与查询日志?

对数据访问及查询日志进行监控与分析，是一项非常重要的工作。在大型组织中，比较常见的方案是使用集中日志管理服务，例如选择 Elasticsearch 和 Kibana 以保存并分析日志文件。Amazon S3 是长期保留及归档日志数据的最佳选择，您还可以使用 Amazon Athena 轻松搜索日志信息并发现值得关注的内容。通过 VPC Flow Logs，您可以监控并分析应用程序中所使用的各项服务间的连接。

以下列出的，是分析类应用程序中常用的日志存储、访问与查询类 AWS 服务。若需了解更多详细信息，请参阅每项服务对应的说明文档：

服务	日志存储选项
Amazon Athena	将查询请求存储在 CloudTrail 中，将查询结果存储在 Amazon S3 中。
Amazon S3	可启用存储桶来存储及访问日志
Amazon Redshift	可启用存储桶中的 Audit Logging 选项以存储访问日志。
Amazon EMR	将日志存储在 Amazon S3 中。您还可使用 log4j appenders 将面向 Presto 及 Hive 的自定义查询日志保存在 Elasticsearch 与 Amazon S3 等当中。
AWS Glue	将日志存储在 Amazon CloudWatch 中。

基础设施保护

关于分析类应用场景下的基础设施保护详细信息，请参阅《AWS 架构完善的框架》白皮书中提出的相关最佳实践。

数据保护

在着手设计分析系统之前，应首先明确与安全性相关的基础实践。例如，您可以在数据分类方法中根据组织数据的敏感度进行划分，同时执行数据加密以提高成功执行未授权访问的难度。这些方法非常重要，能够帮助您有效预防财务损失、严格遵循监管要求等合规性目标。

分析 - 安全 4： 您如何保护静态数据?

要实现并维持良好的安全状态，我们必须保证静态敏感数据得到及时加密。请分析您应用程序需要遵循的安全性与合规性要求，以确定如何执行静态数据加密。在进行数据加密时，请设计出适当的密钥轮换策略。更重要的是，您应使用密钥管理系统以保证以正确方式加密及解密数据。在密钥管理方面，建议您使用外部密钥管理系统或硬件安全模块（HSM）以保证不给未授权访问留下任何可乘之机。最后，建立适当的密钥配额策略，保证密钥不致因数量过多而意外泄露。

要在 AWS 内部进行数据加密，第一步需要定义加密密钥管理策略。我们首先需要回答这样几个问题：我有没有必要自行管理加密密钥？我需要使用专用的密钥管理硬件吗？我需要在本地管理自己的密钥吗？要找到问题的答案，请参照下表了解如何为静态数据选择最佳密钥管理解决方案。

我有没有必要自行管理加密密钥?	我需要使用专用的密钥管理硬件吗?	我需要在本地管理自己的密钥吗?	策略
否	否	否	使用托管 AWS 服务
是	否	否	使用 AWS KMS
是	是	否	使用 AWS CloudHSM
是	否	是	使用您自己的 KMS
是	是	是	使用您自己的 HSM

Amazon S3 允许用户通过服务器端加密（SSE）与客户端加密（CSE）保护静态数据。在 SSE 场景下，您可以从三种互斥选项中任选其一：使用 Amazon S3 托管密钥的服务器端加密（SSE-S3）；使用 AWS KMS 托管密钥的服务器端加密（SSE-KMS）、以及使用客户提供密钥的服务器端加密（SSE-C）。在 CSE 场景下，您又可以使用以下两个具体选项：使用 AWS KMS 托管的客户端主密钥，或者使用客户端主密钥。

Amazon EMR 静态数据加密通过托管“安全配置（Security Configuration）”对象进行配置。此配置文件负责配置 EMRFS 及本地存储卷静态加密。在必要时，您也可以配置 HDFS 透明加密。

Amazon Redshift 可以通过“加密数据库（Ecrypt database）”选项进行加密配置。选择此选项后，服务将对静态数据及备份数据进行加密。相关密钥管理存储选项则为 AWS KMS 与 HSM 服务。

AWS Glue 支持对 ETL 作业及开发端点数据进行静态加密。您可以在配置中将 ETL 作业与开发端点设定为使用 AWS 密钥管理服务（AWS Key Management Service，简称 KMS）密钥以静态写入加密数据。您也可以通过 AWS KMS 管理的密钥，对存储在 Glue 数据目录中的元数据进行加密。此外，您还可以使用 AWS KMS 密钥加密作业书签，以及由爬网程序及 ETL 作业生成的日志。您可以使用 Glue 中的安全配置设定爬网程序、ETL 作业以及开发端点。最后，您可以通过 Glue 数据目录中的设置选项对 Glue 数据目录进行加密。

分析 - 安全 5： 您如何保护传输中的数据?

传输中的数据同样需要安全保障。以下列出的，为分析类应用中经常使用的几种 AWS 服务传输数据加密选项。关于更多详细信息，请参阅各服务所对应的说明文档：

A 节点	B 节点	数据流保护
企业数据	Amazon S3	使用 SSL / TLS 加密；用 AWS Sigv4 签名的 S3 请求
Amazon S3	Amazon EMR Amazon Redshift Amazon Athena	使用 SSL / TLS 加密
Amazon Athena	客户	JDBC 默认使用 SSL / TLS 连接
Amazon EMR	客户	使用 SSL / TLS 加密；随 Hadoop 应用程序客户端而变化
Amazon Redshift	客户	支持 SSL/TLS；需要配置
Amazon EMR 上的 Apache Hadoop		• Hadoop RPC 加密 • HDFS 数据块传输加密 • Hadoop KMS 默认不启用基于 HTTPS 的 KMS

分析 - 安全 6： 您如何保护组织内部的敏感数据?

敏感数据，是指绝不可直接暴露的数据，甚至在组织内部也不可公开。敏感数据的常见示例包括极可能遭到滥用或用于欺诈行为的数据，包括财务信息（例如信用卡号）、个人身份信息（例如社保号码）以及法律要求更严格访问控制的记录。在设计分析类工作流时，请务必了解敏感信息的处理方式与存放位置，并采取措施加以保护。

在着手保护敏感数据之前，我们首先需要明确标记哪些属于敏感数据。Amazon Macie 是一项安全服务，使用机器学习技术以自动发现、分类并保护 AWS 中存放的敏感数据。Amazon Macie 能够快速识别出个人身份信息（PII）或知识产权等敏感数据，并通过仪表板与警报机制帮助查看这些数据是如何被访问或者移动的。

我们可以通过标签指示存储在 Amazon S3 中的数据的对应敏感度以及当前敏感数据的类型。我们可以使用包含 Condition 子句的 IAM 策略限制用户对这类敏感数据的访问操作。Amazon Athena 使用来自存储层授予的权限，因此通过控制对 Amazon S3 中数据的访问，可以限制用户使用 Athena 执行的数据查询。同样，您也可以通过标签与 IAM 策略限制用户对 EMR 及 Athena 中敏感数据的访问。要保护 Amazon Redshift 中的敏感数据，我们可以创建不包含任何敏感数据列的视图，并确保只允许这些非敏感视图的用户访问。

明确拒绝用户执行某些标签操作是一项重要的考虑因素，这样可以防止用户将敏感数据标记为可访问形式。

事件响应

即使采用非常成熟的预防与检测控制措施，我们仍有必要在组织当中制定流程以响应并减轻安全事件带来的潜在影响。工作负载的架构在很大程度上将决定在安全事件期间团队能否正常运行，是否可以隔离或控制目标系统，并将运营恢复至已知良好状态的具体能力。大家应在发生安全事件之前部署工具并设计访问权限，通过活动日（GameDay）举行事件响应演习，这将帮助您的架构有能力及时满足安全调查与恢复工作的需求。关于事件响应的更多具体建议，请参阅《AWS 架构完善的框架》白皮书中的安全支柱部分。

分析 - 安全 7： 在安全警报与事件响应方面，您如何定义及实施相关策略？

在预先定义并达成共识的 SLA 时间内响应与分析应用程序相关的安全警报和事件是非常重要的。数据是您最宝贵的资产，如果分析的基础设施安全出现漏洞，那么一旦无法及时控制，整个数据资产都有可能受到损害。因此，一个重要的目标就是主动发现这些事件并尽快加以补救。例如，您需要通过扫描确定敏感信息（例如数据库的身份验证密钥与密码）是否被嵌入其他公开或私有代码库，防止此类信息外泄给数据库访问造成的影响。此外，如果发现数据存储在特定时段内出现异常的访问模式，您应该立即对其开展调查。不同的组织有不同的实际情况，您应与安全及合规团队紧密合作，为每一项关乎安全的事件制定适当的 SLA。

AWS 提供多种服务以定义、实施及响应安全警报与事件。AWS CloudTrail 是一项支持对您的 AWS 账户进行监管、合规性检查、操作审核以及风险审核的服务。借助 CloudTrail，您可以记录日志、持续监控并保留与整个 AWS 基础设施中的操作相关的账户活动。Amazon Macie 作为一项安全服务，可帮助您轻松发现、分类并保护存储于 Amazon S3 中的敏感数据。Macie 会持续收集 AWS CloudTrail 事件与 Amazon S3 元数据，包括权限与内容分类等。Amazon GuardDuty 是一项托管的威胁检测服务，可持续监控恶意或未经授权的行为，帮助您保护 AWS 账户与工作负载。Amazon GuardDuty 会收集、分析并关联来自 AWS CloudTrail、Amazon VPC Flow Logs 以及 DNS 日志中与 AWS 账户相关的数十亿个事件，为您带来强大的智能威胁检测支持。

资源

请参考以下资源，以了解我们为您整理的卓越运营最佳实践。

文档与博客

- 利用临时凭证通过联合身份连接 Amazon Athena

- Amazon EMR 安全保护最佳实践
- IAM 策略、存储桶策略与 ACL（S3 资源访问控制）
- 数据安全、保护与管理
- 视频 : 在 AWS 上保护企业大数据工作负载
- 视频 : AWS 上的数据湖安全最佳实践

白皮书

- Amazon Web Services: 安全流程概述
- 大数据分析选项
- 安全支柱：AWS 架构完善的框架

合作伙伴解决方案

- AWS 大数据合作伙伴解决方案

第三方工具

- Lumada 数据目录
- Collibra
- Okera
- Apache Atlas
- Apache Knox
- Apache Ranger

可靠性支柱

可靠性支柱包括系统从基础设施或服务中断中恢复、动态获取计算资源以满足需求以及减少诸如配置错误或暂时性网络问题等中断的能力。

定义

云中的可靠性有三个最佳实践领域：

- 基础
- 变更管理
- 故障管理

要实现可靠性，系统必须具有经过周密规划的基础和适当的监控功能，以及处理需求变更、减轻数据存储故障风险的机制。系统的设计应充分考虑故障检测需求，在理想情况下最好具备自动的自我修复功能。

设计原则

在云环境中，我们可以通过多种原则实现可靠性提升。在分析类工作负载中，尤其需要关注以下几点。关于更多详细信息，请参阅《AWS 架构完善的框架》白皮书中提出的设计原则。

- **管理数据资产生命周期、转换与过期机制：**将治理流程引入数据资产的维护体系当中。建立关注数据集相关性与新鲜度的审查周期，保证对托管数据集进行周期性维护。这样一套治理思路可以涵盖数据的更新或刷新方式、对数据价值与使用方输入信息的跟踪、数据过期的标准与流程（包括使用分层存储实现数据的成本管理）。
- **执行数据清理制度：**当管理供机构使用的数据集时，请应用相关机制以保障数据模型标准的定义与执行。各机构工作负载中使用的数据集都应拥有治理模型，以及一个负责记录其控制、访问、清理标准、使用来源以及整体管理方式的存储库。
- **保留数据血缘：**衍生数据集具有可变性，原始数据则不可变。在将数据摄取至数据湖之前，首先应保证下游各流程与操作皆以未经修改的“原始”数据为起点。该原始数据集在经由下游分析应用之前可能会经历多次转换。当数据通过数据血缘捕捉流经分析系统中的每一层时，我们必须始终保持对数据属性的可追溯性。我们可以使用元数据存储对架构变更进行映射与跟踪，从而捕捉这类血缘信息。

最佳实践

基础

关于分析类应用中可靠性的基础最佳实践，请参阅《AWS 架构完善的框架》白皮书。

变更管理

> 分析 - 可靠性 1： 您如何跟踪数据仓库中的元数据变更?

您应建立起一套集中的元数据存储，用以跟踪源与目标中的各项变更。您应建立一条便捷的路径，将这些变更传递至依赖源系统与目标系统的其他下游系统。例如，如果源表中的列发生变更，则应修改读取该列的 ETL 作业，保证其读取经过修改后的列；此外，我们还需要修改指向该列的报告系统中的相关指标计算。

当数据经过数据仓库中的多个层时，能够跟踪目标指标是如何从源开始逐步计算的能力将变得非常重要。只有这样，我们才能对指标中的数值性错误做出调试。系统应该能够跟踪并审计特定时段内元数据发生的一切变更。

在 AWS 上，我们使用 Glue 数据目录维护源系统与目标系统中的元数据存储。Glue 数据目录还能够与 Athena 及 EMR 等其他 AWS 服务相集成，确保查询环境与 ETL 环境之间的一致性。例如，通过 Glue ETL 作业加载至 Amazon S3 中的文件可以作为 Glue 数据目录中的一部分，并通过与 Athena 共享的形式实现无缝查询——整个流程无需将该文件移动至任何其他服务。Glue 数据目录还可以作为运行在 Amazon EMR 上的 Hive metastore 方案。

分析 - 可靠性 2： 您如何保证分析环境中的各配置设定版本保持一致?

通过对数据库、Hadoop 集群以及查询工具等资源进行模板化，我们能够以代码形式维护当前配置参数与环境，以更轻松、更一致的方式部署大数据环境。

通过代码存储库跟踪配置层面发生的变更，我们可以随时了解环境中发生的变更，并在必要时回滚至环境的早期特定版本，以防止配置变更引发的负面影响。

AWS Config 可帮助用户跟踪 AWS 服务配置中的变更。此外，您还可以使用 CloudFormation 维护不同大数据环境间的一致性，借此轻松管理基础设施中的 AWS 资源配置与更新。您还可以使用 AWS OpsWorks，这是一项基于 Chef 的配置与自动化服务。

故障管理

分析 - 可靠性 3： 您如何实现主数据仓库的故障恢复?

主数据仓库遭遇宕机，对于任何企业都不啻为一场灾难。数据仓库负责为下游分析应用（包括各类关键性任务）提供支持，因此需要配合适当的恢复策略，保证能够根据组织划定的恢复时间目标（RTO）与恢复点目标（RPO）及时恢复正常运转。在数据库方面，AWS 为您提供了多种高容错的架构设计选项。例如，您可以在 Amazon Redshift 多节点集群中持续监控各节点是否发生故障；一旦出现故障，系统会自动创建数据副本并将其复制到运行状况良好的节点上，保证查询操作能够获得正常响应。Amazon RDS 还提供跨区域副本与多可用区部署选项，可帮助您立足另一可用区维护主数据库的同步副本，同时保障良好的读性能。如果需要从灾难性故障中恢复，RDS 与 Amazon Redshift 还支持从自动快照到持久对象存储的备份 / 恢复方案。

分析 - 可靠性 4： 您如何实现 ETL 作业的故障恢复？

ETL 作业能够从多个不同源系统处摄取数据，进行转换以供下游使用，而后加载至数据集市内以生成报告。根据数据摄取速度的不同，ETL 作业可以分为批处理或流处理两种方式。此外，ETL 作业还可以选择按计划或按事件进行触发。一旦 ETL 步骤发生故障，可能导致分析应用中的指标过期或错误。因此，我们必须保证 ETL 作业能够持续运行，并在出现故障时具备良好的可恢复能力。

AWS Glue 是一项无服务器服务，帮助您灵活运行批处理 ETL 作业且无需维护任何服务器基础设施。这将大大降低批处理 ETL 作业遭遇故障的几率。同样的，Amazon Kinesis Data Streams 与 Amazon Kinesis Data Firehose 为无服务器流处理服务，可降低流处理 ETL 作业发生故障的风险。AWS Lambda 能够根据事件触发或计划内运行代码完成小型 ETL 作业，也可以在长时间运行的批处理 ETL 作业中充当触发器——无论哪种场景，您都无需维护任何服务器基础设施。

资源

请参考以下资源，了解关于卓越运营最佳实践的更多详细信息。

文档与博客

- AWS Glue - 运行及监控 AWS Glue
- Amazon EMR – 查看及监控集群
- 使用 Amazon CloudWatch 监控 Amazon Kinesis Data Streams 服务

白皮书

- 可靠性支柱

性能效率支柱

性能效率支柱，侧重于如何有效利用计算资源满足业务需求，并强调随着需求变化与技术发展始终保持理想效率。

定义

云中的性能效率有四个最佳实践领域：

- 选择

- 审核
- 监控
- 权衡

建议以数据驱动的方式选择高性能架构。从顶层设计到资源类型的选择与配置，我们应收集来自架构内各个层面的对应数据。此外，重复审核您的选择，确保充分利用 AWS 云在持续发展当中推出的各类最新成果。做好监控工作，则能帮助我们了解实际性能与预期性能之间是否存在差异，并采取针对性的处理措施。最后，大家可以在架构决策中进行权衡以进一步提升性能，例如使用压缩、缓存或者放宽一致性要求等。

本文不再具体阐述审核与权衡方面的最佳实践内容，大家可以直接参考《AWS 架构完善的框架》中的相关实践。

设计原则

在立足云端设计分析类工作负载时，我们可以采取以下几项原则以改进性能效率：

- **使用数据剖析以提升性能。**根据数据的访问与查询检索模式，保证将数据存储在最合适的环境当中。根据业务与应用需求准确定义性能与成本优化目标。发布的数据应包含服务设计目标（例如数据刷新率）。数据剖析具体涵盖数据统计信息（总和、平均值等）、数据倾斜（数值比例失调）以及数据丢失等。一旦出现这些情况，可能会对查询性能与分区策略造成影响，因此应对其开展持续监控。
- **持续优化数据存储。**数据存储优化方法——特别是压缩、分区列、分配键以及排序键——需要随查询模式的变化而不断接受评估与改进。

最佳实践

选择

分析 - 性能效率 01： 您如何为存储数据选择文件格式与压缩选项?

在大多数分析类应用程序当中，数据加载性能往往成为制约数据处理能力的瓶颈。通过压缩静态数据并使用可直接读取压缩数据格式的 AWS 服务，您可以显著降低存储成本、并通过减少查询操作中的数据扫描量以提高性能。您还应使用可进行数据分区的存储格式以进一步减少数据扫描量，从而提高性能效率。

Amazon S3 中提供多种与大数据处理相关的存储选项。您可以选择使用开放格式存储数据，包括带有分隔符的文本（例

如 CSV）、ORC、Parquet、Avro 以及 SequenceFile 等，也可以将数据压缩为 GZIP、BZIP、LZO 以及 Snappy 等格式。这些格式与压缩算法能够优化您的工作负载，进而提高性能并优化存储与查询成本。

对于运行在 Amazon EMR、Athena、AWS Glue 以及 Redshift Spectrum 上的分析类应用程序及对应框架，我们还可以在 Amazon S3 中为其使用的数据选择列格式与压缩。Amazon Redshift 存储还直接提供压缩存储选项。Amazon Redshift 列压缩与编码能够提升查询性能，其中压缩选项可以减少您的存储空间占用量，从而降低 Amazon Redshift 使用成本。另外，请尽可能使用可拆分压缩，保证多个并行运行的工作程序能够从单一对象中读取并处理不同的数据块。

尽可能使用列格式。列格式能够减少数据扫描量并提高性能水平。在查询数据时，我们通常只需要检索其中几列内容。列格式可以确保大数据处理引擎仅从请求的列中检索数据。

分析 - 性能效率 02： 您如何确定分析类应用程序的计算选项与实际大小?

针对每项工作负载运行性能测试，确定架构大小与其中存在的性能瓶颈，据此考虑如何满足目标 SLA 对数据加载、处理与更新提出的性能要求。

AWS 上的分析类服务提供一系列针对 CPU、GPU、内存以及存储资源的计算优化选项。为大数据应用选择合适的实例数量与实例类型，有望提高性能表现并节约成本。例如，大多数分析类工作负载对时间非常敏感，这是为了及时整理出业务运营报告。我们应该立足分析类工作负载的整体端到端完成时间这一背景进行性能指标检查，同时监控分析管道中的各个步骤。

对应用程序资源进行监控，例如 EC2 实例、EMR 集群、RDS 实例、Amazon Redshift 集群、Amazon ElastiCache 以及 AWS Glue 等，以保证既充分满足性能目标、又不致发生资源过度配置问题。

分析 - 性能效率 03： 您如何在数据湖中组织或分区数据?

通过数据分区，可以减少单项查询中的数据扫描量，借此提高性能并降低成本。

监控用户的查询行为，以查找过滤器及“group by”语句中经常使用的列。我们可以对这些列进行分区以获得最佳性能。按数据列进行分区的具体方法多种多样，比较常见的做法是根据时间进行分区，其中多级分区方案已经得到行业的普遍认可。例如，如果每小时都有数据传入，则我们可以按照年、月、日及小时进行分区。而如果您的数据来自多种不同来源，

但每天仅需加载一次，则可按照数据源标识符与日期进行分区。如果同时存在多个业务组或客户执行数据查询，您还可以考虑按数据中的业务组 ID 或客户 ID 列进行分区。

避免过度分区，因为这可能带来额外开销，并导致 S3 存储桶中的文件过小而引发性能下降。例如，若大多数查询会按年份进行过滤，那么按照年、月、日及小时进行分区就属于过度分区。具体分区方式视用例而定，而且组织内的用例情况也有可能持续变化，我们需要随时跟进以选择最合适的分区方法。

审核

关于在分析类应用中提升性能效率的审核最佳实践，请参阅《AWS 架构完善的框架》白皮书。

监控

分析 - 性能效率 04： 您如何监控数据摄取、批处理作业以及 ETL 作业的性能表现？

对分析类应用进行监控，可以保证您随时使用经验数据确定横向扩展策略、计算选择，同时主动管理一切可能对性能造成负面影响的服务限制。

摄取监控

AWS 提供 Kinesis Data Streams、Kinesis Data Firehose、AWS Database Migration Services（数据库迁移服务，简称 AWS DMS）以及 AWS Glue 等服务以简化超大规模数据的摄取流程。您还可以使用 Amazon EMR 或 Amazon EC2 上的框架开发自己的摄取应用。与其他应用程序及基础设施一样，您同样需要监控这些服务以根据需求对其进行适当伸缩。

例如，Kinesis Data Streams 中的数据流应以流与分片的层级进行监控，以确保随时可进行重新分片，并灵活扩展 Kinesis 生产者与消费者以适应实际需求。Glue ETL 作业还应监控数据处理单元（DPU）中的内存与 CPU 使用情况，保证您有充足的 DPU 在正常时间范围内完成当前作业。AWS DMS 能够捕捉连续变更的数据并生成副本，您在实际使用时应选择最适合当前吞吐量需求的 DMS 复制实例类型。

批处理与 ETL 作业监控

多数分析类应用中都涉及到批处理与 ETL 作业，用以运行复杂的计算、聚合、模拟以及预测；或者需要在离线场景中利用海量的数据集进行机器学习模型训练。这类工作负载往往需要持续运行数小时甚至数天。

对于这类作业，我们通常会选择以自动化方式进行编排与调度。因此，除了与批处理及 ETL 作业直接相关的计算与存储资源利用率指标外，我们还需要检测作业的编排与调度系统，收集端到端指标与作业警报。作为基本参考，大家至少应该关注作业执行时间、作业 SLA 以及作业的整体资源利用率。

CloudWatch 指标与警报

在使用更高级别的 AWS 分析服务时，您应使用其中内置的各项 CloudWatch 指标并创建警报。这些警报既可用于通知，也可以在应用程序之内自动执行扩展策略。

对于运行在 Amazon EC2 实例上的大数据应用，您可能还需要检测其他监控指标与日志记录。您可以使用 CloudWatch 日志与自定义指标将这些应用统计信息发送至 Amazon CloudWatch，也可以与我们的 APN 合作伙伴联手，由他们帮助您简化并增强应用场景中对监控与日志记录的性能分析流程。

分析 - 性能效率 05： 您如何监控查询操作的性能表现?

随着时间推移，我们也需要对查询与检索性能进行监控，据此不断重新评估当前用例中的数据存储、数据库选项与文件格式是否合适。大部分分析类应用最终会将摄取到的数据与处理后的结果保存在持久存储内，以支持各类后续查询或检索操作——包括应用程序查询、数据湖上的即席 SQL 查询、机器学习训练以及给予数据仓库的商务智能查询等等。

您应该对 Amazon Redshift、Amazon Athena、Amazon EMR、Amazon DynamoDB 等服务的查询性能进行监控，以便在性能优化方面做出明智的决策。具体判断结果，取决于您当前使用的 AWS 功能、服务以及整体架构。

权衡

关于分析类应用场景下关于性能效率支柱中权衡因素的详细信息，请参阅《AWS 架构完善的框架》白皮书中列出的最佳实践。

资源

请参考以下资源，了解关于性能提升的更多最佳实践相关信息。

文档与博客

- AWS Glue – 运行及监控 AWS Glue
- Amazon EMR – 查看并监控集群
- Realtor.com 网站如何利用 AWS CloudTrail 与 Amazon QuickSight 监控 Amazon Athena 资源使用情况

- Amazon Athena 十大性能调优技巧
- 使用 Amazon CloudWatch 监控 Amazon Kinesis Data Streams 服务
- 数据单元测试

白皮书

- 性能效率支柱

成本优化支柱

成本优化支柱当中，包含系统在整个生命周期之内不断完善及改进的过程。从最初的概念验证、到初始设计、再到生产级工作负载的持续运行，本章节中提出的最佳实践将帮助大家构建并运行起具有明确成本的系统，借此实现业务成果、将成本控制在最低水平，最终帮助您的企业获得最理想的投资回报。

定义

云中的成本优化有四个最佳实践领域：

- 具有成本效益的资源
- 供需匹配
- 对支出的认识
- 持续优化

与其他支柱一样，我们在成本优化方面同样需要做出权衡取舍。例如，您到底是希望优化上市速度，还是运营成本？在某些情况下，与前期成本优化相比，对速度进行优化往往更具现实意义——包括快速推动产品上市、交付新功能或者只是纯粹为了按时完成任务。需要强调的是，设计决策中总会有因过于仓促而缺少经验数据指导的情况，这是因为我们总会过度补偿“以防万一”的情况，而不愿意投入时间通过基准测试找到最具成本优势的部署方案。这通常导致过度配置与优化缺失。在以下章节中，我们将探讨与初始部署及持续成本优化相关的技术性与战略性指导。

在规模层面，分析类工作负载与传统工作负载的主要区别在于对数据存储及计算资源的需求。以分析为基础实现业务价值的应用程序通常对时间比较敏感，为了满足这种响应要求，我们的架构设计很可能出现过度配置问题。为此，我们可以考虑以下几点，以优化分析平台的成本水平。

设计原则

在云环境中，我们建议大家参考《AWS 架构完善的框架》白皮书中列出的成本优化设计原则，这些原则在分析类工作负载下仍然切实有效。

最佳实践

具有成本效益的资源

分析 - 成本优化 01： 如何为分析类应用程序选择正确的计算解决方案?

在分析类工作负载的初始规划与测试阶段，请选择按需资源以发挥云服务的灵活性优势，并对多种备选架构进行持续迭代，直到根据分析项目的各类需求对架构完成全方位优化——例如为特定作业选择合适的工具，以满足分析管道运行所需的时间要求。按需资源的计费模式可帮助大家结合多种方案与不同规模实现广泛的低承诺测试。在测试阶段结束时，您的项目应该已经拥有一套清晰明确的资源需求，具体包括资源数量、类型与大小，以及工作负载的稳定状态或峰值水平。

您也可以选择预留实例，在资源利用率较为稳定的工作负载中节约大量成本。可以为 Amazon DynamoDB、Amazon RDS、Amazon Redshift、Amazon EMR 以及 Amazon ElastiCache 等服务选择预留实例。

对于瞬时工作负载或者无状态逻辑的工作负载，您也可以考虑使用 Amazon EC2 竞价实例以进一步节约成本。您可以将竞价实例与其他 AWS 服务紧密结合起来，包括 Amazon EMR、Amazon EC2 Auto Scaling、AWS Auto Scaling、CloudFormation、AWS Batch、Data Pipeline 以及 Amazon Elastic Container Service（简称 Amazon ECS）等。您也可以将其与各类非 AWS 服务集成起来，包括 Jenkins、Bamboo、Alces Flight 以及 Terraform 等。如果您希望尽可能降低开发或质量保证（Dev/QA）类应用程序的成本，那么竞价实例无疑将是您的最佳选择。

在 Amazon EMR 上，您可以通过 instance fleet 获得更多与实例配置相关的选项与自动化的智能方案。例如，您现在可以在单一列表中添加最多五种实例类型及其对应容量级别。在集群创建过程中，Amazon EMR 可以根据实例类型自动配置按需实例与竞价实例的容量。如此一来，您的集群容量既能够满足需求，又可获得较低的运营门槛与更高的成本效益。

分析 - 成本优化 02： 您如何对数据湖内的存储数据进行成本优化?

在实际场景中，数据湖内数据资产呈指数级增长的情况已经非常普遍。面对这种增长趋势，我们当然有必要认真考虑数据湖的成本优化问题。

Amazon S3 提供多种存储类型，帮助您随时间推移持续优化运营成本。您可以考虑将数据从 S3 Standard 迁移至 S3-Infrequent Access（在某些情况下，也可以选择 S3-Infrequent Access 单可用区）、再到 Amazon S3 Glacier 以不断降低存储成本。更重要的是，Amazon Redshift Spectrum、Amazon Athena、Amazon S3 Select 以及 S3 Glacier Select 等新推出的服务已经能够直接查询存储在 Amazon S3 中的数据，这意味着我们可以更灵活地变更数据存储方式，且不会对正常分析能力造成影响。在大多数情况下，向 Amazon S3 前缀命名约定中添加 hive 风格的分区结构、压缩数据、以及将数据存储为列格式，都能提高查询速度并降低查询操作的成本。

Amazon Athena 是一项无服务器查询服务，可帮助用户轻松使用标准 SQL 在 Amazon S3 中直接实现数据分析。使用 Amazon Athena，您只需为实际执行的查询操作付费，具体计费单位为每项查询所扫描的对应数据量。通过数据压缩、数据分区以及列格式转换等方法，您可以极大减少每项查询操作中 Athena 所执行的数据扫描量，借此显著节约成本并提高查询性能。Amazon Athena 支持多种数据格式，包括 CSV、TSV、JSON 以及文本文件，同时亦支持多种开源列格式，例如 Apache ORC 与 Apache Parquet。Athena 还支持 Snappy、Zlib、LZO 以及 GZIP 等数据压缩格式。通过数据压缩、数据分区以及列格式转换等方法，您可以显著节约成本并提高查询性能。

S3 Select 与 S3 Glacier Select 可以帮户应用程序使用简单的 SQL 表达式从对象中检索特定的数据子集。您可以使用 Amazon S3 Select 查询 Apache Parquet 格式、JSON 数组以及 GZIP 或 BZIP2 压缩格式的 CSV 与 JSON 对象。GZIP 与 BZIP2 也是目前 Amazon S3 Select 所支持的两种压缩格式。

Amazon Redshift 还包含 Redshift Spectrum，可帮助用户直接针对 Amazon S3 中的 EB 级非结构化数据运行 SQL 查询。Amazon Redshift Spectrum 根据您针对 Amazon S3 所执行查询的具体数据扫描量计算费用。Parquet 将数据保存为列格式，可帮助 Amazon Redshift Spectrum 尽可能减少查询所涉及的列数量。各类测试表明，经过分区的 Parquet 文件不仅执行速度更快，而且要比未经分区的、基于行的 CSV 文件更具成本效益。Redshift Spectrum 支持多种结构化与半结构化数据格式。关于受支持格式的更多详细信息，请参阅 Redshift Spectrum 说明文档。

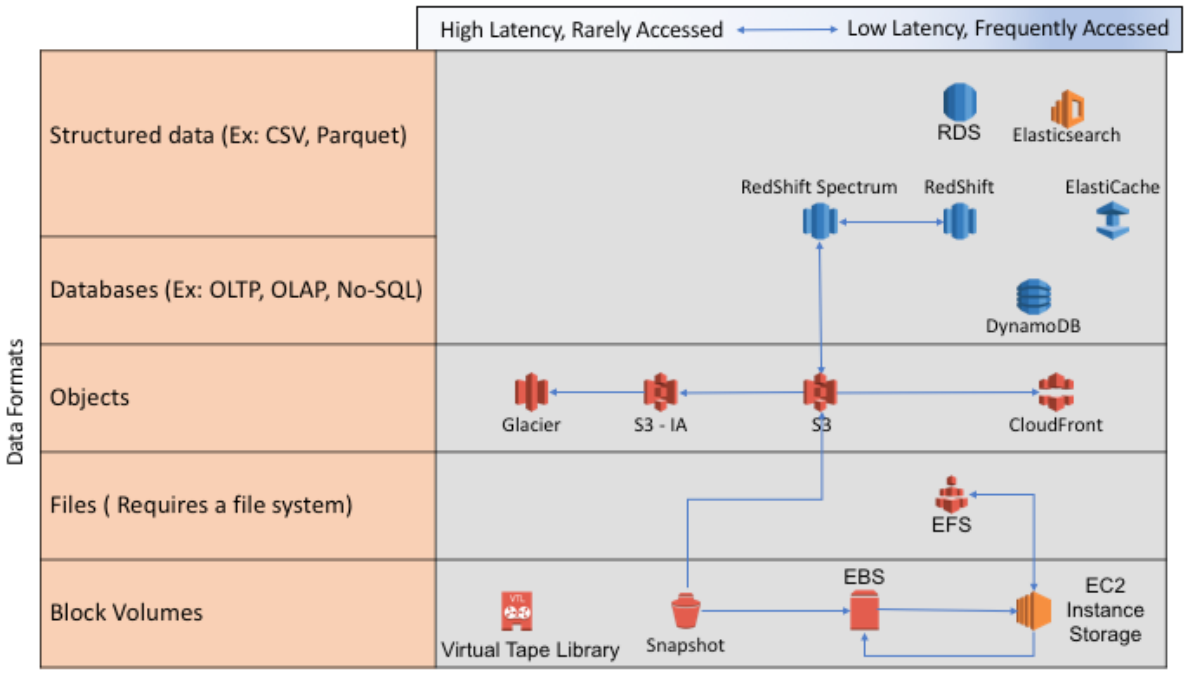
对于 MapReduce 工作负载，建议大家考虑将数据存储在 Amazon S3 中并配合使用 Amazon EMR 文件系统（EMRFS）。EMRFS 是 HDFS 的一种具体实现，所有 Amazon EMR 集群都使用它直接从 Amazon EMR 读取常规文件并写入至 Amazon S3。EMRFS 能够在 Amazon S3 当中存储持久数据以轻松供 Hadoop 使用，同时提供具备一致性保证的视图与数据加密等功能。在 EMRFS 的支持下，您可以轻松启动暂时性 Amazon EMR 集群，且仅按集群使用时长付费以节约宝贵资金。

分析 - 成本优化 03： 如何对存储进行分层以提高成本效益?

分层存储则根据使用模式将数据集转移至不同的存储层，借此优化存储成本。以“热数据”（经常访问的数据）为例，我们通常将其保存在内存内（高速缓存）存储中，“温数据”（偶尔访问的数据）保存在对象存储与数据库当中，而“冷数据”则保存在磁盘驱动器或磁带当中用于归档。我们强烈建议您根据分析类工作负载的具体需求进行存储分层。

在分析类应用的存储层选择方面，AWS 为您提供多种解决方案选项。正确的存储层将成为您分析基础设施的根基，也将帮助您通过正确服务建立必要的业务规模。关于 AWS 存储产品的更多详细信息，请参阅此链接。

下表展示了如何在不同的 AWS 存储服务之间进行数据分层与移动。



图九：AWS 中的分层存储

最左边的列表示构建分析类应用时所常用的数据格式。右侧则列出与之对应的不同 AWS 服务。越靠右，代表该服务越适用于低延迟、高访问频率的数据；越靠左，则代表该服务越适用于高延迟、低访问频率的数据。箭头的指向，表示数据如何从一项服务转移至另一项服务。

- **结构化数据与数据库：**该层表示以 CSV、Parquet 以及 ORC 等结构化格式存在的数据集，或者存储在 OLAP 或 OLTP 数据库中的数据集。这些都是实现高查询性能的理想选择，同时也是低延迟类应用程序所必需的格式。如上图所示，存储在 S3 上的结构化数据可以与 Redshift Spectrum 无缝对接以创建外部表。您可以通过 SQL 语句将外部表中的数据与 Amazon Redshift 物理层表进行合并。AWS 在这一层中还提供 ElastiCache 及 RDS 等其他多种服务，其中 ElastiCache 可在内存内处理数据集以加快响应速度；Elasticsearch 允许您在数据集上构建搜索功能；RDS 用于以 OLTP 格式存储数据；Amazon DynamoDB 用于存储需要 NoSQL 技术的数据。您可以从这些服务中任意选择，存储需要经常访问且要求较低检索延迟的数据。

- **对象：**Amazon S3 是 AWS 推出的对象存储服务，具有良好的持久性。它能够与其他 AWS 计算服务（例如 EC2、EMR、Lambda、DynamoDB 以及 Amazon Redshift）通过接口连通，以支持可扩展计算架构。S3 在存储层内提供多种不同选项，可供您任意选择。其中 S3 Standard 适用于访问频率较高的对象；S3 Infrequent Access（S3 IA）的存储成本较低，更适合访问频率不高的对象。需要注意的是，S3 IA 会带来额外的检索费用，因此请确保只在其中保存访问频率较低的对象。至于几乎从不需要访问的对象，建议您将其保存在 AWS 的归档服务 Glacier 当中。

- **文件：**对于希望在 AWS 上托管共享文件系统的客户而言，Amazon Elastic File System（简称 EFS）是个不错的选择。EFS 为能够并行安装在多个实例上的文件系统提供 POSIX 接口与 NFS 挂载点，因此特别适合承载那些可以通过多个计算实例并行处理的大数据工作负载。此外，EFS 存储容量也将随着客户资源用量的提升而增长，您无需预配置任何存储卷，自然不必考虑配置过度或配置不足等问题。对于某些特定分析类工作负载，您还可以考虑使用 Amazon FSx for Lustre——这是一套原生匹配 Amazon S3 的高性能文件系统。

- 块存储卷：Amazon EBS 允许客户将块存储添加至 EC2 计算实例中。这些块存储卷能够以 EBS 快照的形式脱离于实例生命周期之外。快照将被保存在 S3 上，可挂载至任意 EC2 实例中，供用户对卷内存储的文件执行计算。您可以使用 Amazon 数据生命周期管理器（Amazon DLM）自动化快照的创建、保留和删除以备份 Amazon EBS 卷。

您可使用生命周期策略在不同存储层间转换，并通过生命周期策略实现对象在不同存储层间的自动转移。您可以使用 S3 存储管理功能构建生命周期策略，并借此了解特定时段之内某一对象的访问频率。S3 还提供 S3 Intelligent-Tiering 存储类，负责根据对象的访问模式自动将对象转移至两种访问层之一：频繁访问与非频繁访问。整个分层过程将自动执行，您无需采取任何额外操作。任何连续 30 天未被访问的对象都将被转移至非频繁访问层；当该对象被再次访问后，则将自动被移回频繁访问层。

对于需要在全球范围内分发及 / 或需要在请求源附近保留缓存以降低延迟的对象，客户可以使用 Amazon CloudFront，即 AWS 提供的内容交付网络（CDN）服务。它可以在多个 AWS 边缘位置使用，负责缓存 S3 中的数据对象。

在 AWS 上构建数据湖时，重要的是关注其中上面重点讨论的各类 AWS 存储选项，保证为您的数据集选择正确的服务。

供需匹配

分析 - 成本优化 04： 您对数据生命周期有何规划?

在设计数据湖或者数据分析项目时，访问模式、事务并发量以及可接受的事务等待时间等具体指标，将直接决定您为数据选定的存储位置。同样重要的是，我们还应充分考量旧数据的访问频率，据此制定数据生命周期规划，保证按不同分层将数据按访问频率逐步迁移至成本更低的存储方案中，且不对业务运营目标造成任何影响。

例如，在分析日志数据时，我们可以在前 90 天中将日志保存在 S3 Standard 中以保障查询性能；从第 91 天到第 365 天，这些陈旧数据可被迁移至 S3 Infrequently Accessed 类。从第 365 天开始，我们可以将数据进一步迁移至 S3 Glacier 当中。准确把握数据的访问时间与访问频率，将帮助大家确切理解何时适合在不同存储层间转移数据，以及何时可以编入数据生命周期规划。

您可以使用多项 AWS 功能对当前数据存储需求以及与之匹配的数据生命周期规划进行评估。S3 Inventory 与 S3 Analytics 就能帮助用户对 S3 中的存储对象进行识别，进而分析对这部分存储数据的访问模式，帮助您决定何时将正确的数据转换为正确的存储类别。例如，使用 S3 Analytics，您可以回答以下问题：我检索了多大的存储空间？我检索了全部存储空间中的百分之多少？我不常访问的存储空间有多大？存储设备中的对象已经存在多久了？

此外，对于数据仓库类工作负载，我们还可以将其中访问频率较低的数据从本地存储迁移至 S3。在需要对冷数据进行分析时，建议大家使用 Athena 或 Redshift Spectrum 等服务，将关系型或数据仓库存储中的数据与 S3 内可直接查询的数据结合起来。通过这种方式，数据库与数据仓库中的存储配置总量往往将大大低于在本地存储全部数据所需要的存储量。

分析 - 成本优化 05： 您如何计算单一数据处理步骤的成本?

重要的是，我们需要在每个单独的数据处理步骤或单独的管道分支中考虑分析工作流成本。只有了解了这种细粒度分析工作流成本，我们才能确定需要将开发资源集中在哪里，并对整体分析组合进行更准确的投资回报计算。例如，如果我们需要每小时执行一次高成本 ETL 作业，且需要每晚进行一次下游分析，那么在确定管道中相应步骤的有效成本之后，我们可以更自信、更有理有据地降低 ETL 作业频率。

在以上场景下，“数据处理步骤”是指产生响应数据集的事件，例如 SQL 查询、MapReduce 函数、数据转换、机器学习模型推理、甚至是导入 / 导出作业等。AWS 为您的分析类应用提供多种选项，因此一旦只考虑性能而未关注成本，可能导致您为业务需求保留过高的配置容量。

影响数据处理成本的因素包括：数据存储位置、数据处理频率、ETL 或作业并发性、每个处理周期所对应的数据大小、处理持续时间及 / 或可接受的延迟、数据格式、网络流量成本以及上下游流程等。

在对大数据工作流程进行成本优化时，一项最佳实践在于创建数据管道模型。对于每个步骤，请考虑上述因素以准确计算单一步骤所对应的成本。例如，将数据存储在 S3 Standard Infrequently Accessed 类（S3-IA）中的成本，要低于直接将静态数据保存在 S3 Standard 中。但是，我们还要考虑到从 S3-IA 中反复检索数据所带来的处理成本，因为存储 + 检索的总成本将高于 S3 Standard，意味着 S3 Standard 反而更具经济效益。大家还需要以类似的方式考虑如何存储及处理 Amazon Redshift/Redshift Spectrum、Athena/RDS 中的数据。每项 AWS 服务都是为了满足特定需求而设计的，通过实际使用成本，用户完全可以摸索出它们最理想的使用场景。

AWS 提供详尽的账单报告，再配合 APN 合作伙伴提供的第三方解决方案，可按资源类型为用户提供细粒度成本明细、帮助您快速计算单一数据处理步骤所对应的成本。

对支出的认识

分析 - 成本优化 06： 您如何跟踪分析类应用所使用数据的“新鲜度”？

在制定数据生命周期规划的过程中，跟踪数据“新鲜度”是一项非常重要的工作。在大多数情况下，我们需要建立元数据存储库以跟踪数据移动——这不仅能够让我们对当前数据质量保持信心，更能够及时发现那些更新频率较低的数据。这类更新频率较低的数据一般应该从相对昂贵的低延迟存储，被转移于成本较低的高延迟存储当中。您也可以考虑利用自动化系统执行不同存储层间的数据移动操作。

例如，假定一家企业需要以每天、每月及每年为周期更新销售报告。在将交易情况汇总为每日报告之后，他们可以将数据从 RDBMS 迁移至 Amazon S3，并在 S3 中将数据摄取至批处理系统（例如 Amazon EMR 或者 WS Glue 作业）当中。

Herd 是一款由美国金融行业监管局（FINRA）开发的开源元数据目录工具。此外，也有不少其他商业供应商提供元数据管理与治理解决方案。

分析 - 成本优化 07： 您如何在分析类应用中管理数据传输成本?

在 AWS 中，数据传输成本是指将数据由 AWS 通过互联网或 Direct Connect 向外部传输、在不同区域间转移数据、以及在不同可用区间传输数据所带来的成本。面向 AWS 的数据传入及可用区之间的数据传输完全免费。

一项最佳实践是对应用程序进行横向扩展（即添加更多主机），而非纵向扩展（增加单一主机的内存或 CPU 容量），旨在降低因主机宕机所引发的应用程序崩溃的风险。类似地，另一项最佳实践建议我们将横向扩展主机部署在多个可用区当中，借此减少单一可用区宕机所引发的应用程序崩溃的风险。我们需要监控那些通信频繁、或者在不同主机之间产生大量网络流量的应用程序，借此明确网络流量成本，并在必要时重构应用程序以减少网络流量，同时保持良好的应用可用性与弹性。对于临时工作负载，我们可以在单一可用区内进行横向扩展，并在必要时随时关闭 / 重启以降低成本。

另外，大家还应该特别关注那些将全部数据托管在同一区域或者本地数据存储中的应用程序。在大多数情况下，缩小数据存储位置与云端分析工具间的距离能够降低数据传输成本、提高运营成本效益。再有，对于需要重复读取相同数据的应用程序，我们也可以在应用程序与本地数据存储之间部署 Amazon ELastiCache 等数据缓存服务。

VPC Flow Logs 是一款用于在 AWS 环境中监控数据流量的工具。VPC Flow Logs 可帮助我们捕捉到往返于各 VPC、子网或特定网络接口的 IP 流量信息。这些被监控网络接口产生的日志流数据将被保存为流日志记录，其中包含流量源、流量目的地以及流量字节数等记录字段。定期收集并分析 VPC Flow Logs，将为分析类应用的网络传输成本（特别是在多租户环境当中）的特征提供重要的见解。对于已经了解起特征的应用程序，这里不建议大家连续收集并存储 VPC Flow Logs 记录，这是因为此类记录往往体积庞大、会带来可观的存储成本。

分析 - 成本优化 08： 您为分析类应用需要消耗的资源制定了怎样的成本分配策略?

在成本优化方面，很重要的一点就是为用户及财务利益相关方提供分析类工作负载成本的可见性。对于往往需要大量计算资源才能在短时间内得出高价值结果的分析类工作负载而言，这种可见性更是至关重要。与之对应的，在使用独立或共享租户架构时，您选择的成本分配策略可能会有所不同。

在独立的分析方法中，各团队或业务部门结合业务实际情况保留一部分 AWS 资源。这种成本分配可以通过多种方式实现，目前两个主流方法分别为成本分配标签与多账号策略。在成本分配标签中，我们使用一个或者多个元数据键值对作为 AWS 资源的标签，其中的键负责指定成本指标，而值则指定业务线或账单组。例如，一个键值对可以将键“成本中心”

与值“商务智能工具”关联起来；另一个键“部署阶段”则可与值“测试”关联起来。使用多个键值对，即可建立起细粒度的成本报告。与孤立分析工作负载对应的另一种方法则是多账户策略，主要用于更高层级的安全与计费分离。

AWS Organizations 服务允许用户建立起一个嵌套式的 AWS 账户组体系，其中所有账户账单都将被合并在最顶层的主付款人账户之下。各账户中的一切结算性活动，都将在月度账单上以独立的订单项形式显示。对比这两种方法，多账户策略能够大大降低计费分离的门槛，用户无需在全部资源上强制使用标签；但其弊端就是在面对多个账户时，用户与资源管理可能带来不必要的复杂性。

在共享租户分析方法中，我们可以部署一套分析平台以供多个业务部门共同使用。同样的，共享租户平台也将为客户提供 SaaS 分析服务。共享租户模型中的成本分配机制，要求我们将日志记录与成本模型结合起来，借此确定每一笔交易或者工作负载的正确价格。除了记录发出请求的实体之外，我们还建议大家捕捉运行时间、读取 / 写入量以及数据扫描量等指标。AWS CloudTrail 与 CloudWatch Logs 都提供面向 API 调用与事件日志的捕捉功能，您可以借此配合成本分配模型以会话为单位严密控制成本。

持续优化

《AWS 架构完善的框架》白皮书中提出的持续优化最佳实践，也同样适用于分析类应用场景中的成本优化目标。

资源

请参考以下资源，了解卓越运营最佳实践方面的更多详细信息。

文档与博客

- 使用 Amazon 预留实例进行成本节约
- Goodreads 如何将 Amazon DynamoDB 表转移至 Amazon S3，并使用 Amazon Athena 执行查询
- Amazon EMR 中的集群规模调整与自动伸缩最佳实践
- 在 AWS Glue 中处理分区数据
- 在生产环境下将 Amazon Redshift Spectrum、Amazon Athena 与 AWS Glue 同 Node.js 配合使用
- 文档 : 使用成本分配标记

白皮书

- AWS 架构完善的框架中的成本优化支柱
- 奠定基础：为成本优化设置必要环境
- Amazon EC2 预留实例及其他预留模型

- 规模化使用 Amazon EC2 竞价实例
- 建立成本透明与责任分摊文化
- 正确配置实例尺寸：如何根据工作负载配置实例资源
- AWS 存储优化

总结

此文档旨在帮助大家了解如何在云环境中设计并构建起可靠、安全、高效且经济的分析类工作负载。我们探讨并分析了一系列通行的架构与设计原则。本文还立足分析类场景对架构完善的支柱做出解读，提出多项在新的或现有分析架构中需要关注并考量的问题。将架构完善的框架引入架构，将帮助您构建起更健壮、更稳定且更高效的系统，保证您专注于运行分析管道以突破固有的业务极限。随着工具与流程生态系统的发展成熟，分析领域也在不断汲取营养、持续壮大。以此为基础，我们将不断更新本文内容，帮助大家始终享受最新成果、保证分析应用架构的合理性。

贡献者

感谢各位参与者为本文编撰做出的卓越贡献：

- Radhika Ravirala, Amazon Web Services 高级专业解决方案架构师
- Laith Al-Saadoon, Amazon Web Services 高级解决方案架构师
- Wallace Printz, Amazon Web Services 高级解决方案架构师
- Ujjwal Ratan, Amazon Web Services ML/AI 生命科学解决方案首席架构师
- Neil Mukerje, Amazon Web Services EMR 解决方案架构师

扩展阅读

关于更多详细信息，请参阅以下资源：

- AWS 架构完善的框架
- AWS 上的数据湖与分析
- AWS 上的大数据
- AWS 大数据博客

• 数据与分析解决方案主题

文档修订

日期	说明
2020 年 5 月	首次发布