AWS 上的各类大数据分析选项

2016 年 1 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

本份白皮书对各项服务加以概括，旨在帮助架构师、数据科学家以及开发人员了解 AWS 云之上的各类大数据分析选项，具体信息包括：

- 理想的使用模式
- 成本模型
- 性能
- 技术性与可用性
- 可扩展性与弹性
- 接口
- 反模式

本份白皮书还将展示部分实际分析选项场景，同时探讨用于在 AWS 之上实现大数据分析的各类资源。

简介

随着数字化社交水平的逐渐提高，数据生成与收集总量亦迎来了显著增长。这种日益增长的待分析数据使得传统分析工具面临着极为严峻的挑战。我们需要利用创新手段以缩小所产生数据与可分析数据之间的巨大鸿沟。

大数据工具与技术提供众多机遇与挑战，其能够有效地分析数据以了解客户偏好，从而获得市场竞争优势并实现业务拓展。数据管理架构已经由传统的简单数据仓库演变为复杂的结构模型，且能够解决更多要求，例如执行实时与批量处理、应对结构化与非结构化数据以及高速事务处理等等。

Amazon Web Services （简称 AWS）提供广泛的托管服务平台，能够帮助大家快速且轻松地构建、保护并以无缝化方式扩展端到端大数据应用。无论大家的应用需要处理实时数据流或者进行批量数据处理，AWS 都能够提供相关基础设施与工具，从而解决大数据项目中的各类实际挑战。无需购置硬件，亦不需要基础设施维护与规模伸缩。大家只需要收集、存储、处理与分析大数据内容即可。AWS 拥有一套完整的分析解决方案生态系统，其专门针对处理此种规模数据并为业务提供洞察能力等需求所设计。

AWS 在大数据分析领域的优势

在分析大规模数据集时，我们需要能够甚至输入数量的总量与分析方式随时调整计算能力。此类大数据工作负载的处理模式非常适合利用云计算模型加以解决，其能够根据实际需求轻松对应用程序对应资源进行规模伸缩。而随着需求的实际变动，大家亦可对 AWS 环境进行横向或者纵向调整，无需等待额外的硬件配送或者进行前期投资，即可为需求提供充足的资源容量。

更为传统的基础设施之上的关键性任务应用程序，系统设计师只能选择过度配置以满足各类需求，从而确保业务需求提升、数据量激增时具备系统资源进行处理。相比之下，AWS 能够在数分钟之内轻松配置更多容量与计算资源，意味着大家的大数据应用程序随意伸缩，且时刻保持系统尽可能接近最佳效率。

另外，大家也可以随意访问 AWS 提供的多个不同地理服务区 [1]，从而充分发挥计算灵活性并使用其它可扩展服务以构建复杂的大数据应用。其它服务具体包括负责存储数据的 Amazon Simple Storage Service（即 Amazon 简单存储服务，简称 Amazon S3）[2] 以及能够轻松编排任务以移动及传输数据的 AWS Data Pipeline[3]。另外，AWS IoT（物联网）[4] 允许联网设备与云应用及其它联网设备相交互。

再有，AWS 拥有众多选项以将数据引入云环境，其中包括用于加快 PB 级别数据进行传输的安全设备 AWS Import/Export Snowball[5]，负责加载数据流的 Amazon Kinesis

Firehose[6] 以及通过 AWS Direct Connect[7] 实现的可扩展专有连接。随着移动设备的使用量快速增长，大家可以在 AWS Mobile Hub[8] 当中可以将各个服务加以结合，从而收集并量化应用使用量及数据，或者将数据导出至其它服务以进行后续定制化分析。

AWS 平台的这些特性使其非常 适合处理各类大数据难题，而且相当一部分客户已经成功在 AWS 之上成功实现大数据分析工作负载。欲了解更多与此类实例相关的细节信息，请参阅由 AWS 云支持的大数据与高性能计算。[9]

以下各项服务分别可用于大数据的收集、处理、存储与分析：

- Amazon Kinesis Streams
- AWS Lambda
- Amazon Elastic MapReduce
- Amazon Machine Learning
- Amazon DynamoDB
- Amazon Redshift
- Amazon Elasticsearch Service
- Amazon QuickSight

另外，Amazon EC2 实例亦可用于自管理型大数据应用。

Amazon Kinesis Streams

Amazon Kinesis Streams[10] 允许大家构建自定义应用程序，用于实时处理或者分析数据流。Amazon Kinesis Streams 能够持续捕捉并存储每小时 TB 级别数据，这些数据可来自数十万个源，例如网站点击流、财务交易、社交媒体推送、IT 日志以及位置追踪事件等等。

凭借着 Amazon Kinesis 客户端库（简称 KCL），大家可以构建 Amazon Kinesis 应用并利用数据流以支持实时仪表板、生成警报并实时动态计费与广告方案。大家也可以将数据由 Amazon Kinesis Streams 发送至其它 AWS 服务，例如 Amazon Simpe Storage Service（简称 Amazon S3）、Amazon Redshift、Amazon Elastic MapReduce（简称 Amazon EMR）以及 AWS Lambda。

要为我们的数据流配置输入与输出级别，大家需要以每秒 1 MB 作为基本单位（MB 每秒），使用 AWS 管理控制台、API[11] 或者 SDK[12]。大家的数据流规模亦可随时进行上调与下调，且无需重启数据流或者对推送数据的数据源造成任何影响。在数秒之内，引入流内的数据即可用于分析。

数据流可立足于单一服务区跨越多个可用区，且全天 24 小时起效。在这一时间窗口之内，数据可用于读取、重复读取、回填及分析，或者移动至长期存储资源当中（例如 Amazon S3 或者 Amazon Redshift）。该 KCL 还能够帮助开发人员专注于创建自己的业务应用，同时摆脱数据流负载均衡、分布式服务协调以及数据处理容错等繁重的日常任务。

理想的使用模式

Amazon Kinesis Streams 适用于各类需要自产生方（数据源）快速移动数据并持续进行处理。这一处理流程可在将数据交付至其它数据存储资源之前对其进行转换，驱动实时指标与分析，或者导出及汇聚多个数据流以引入更多复杂数据流或者进行下游处理。以下几项为使用 Amazon Kinesis Streams 进行分析 典型场景。

- **实时数据分析**– Amazon Kinesis Streams 能够对各类数据流进行实时数据分析，具体包括分析网站点击流数据及客户互动分析。
- **日志与数据供应的纳入与处理**– 利用 Amazon Kinesis Streams，大家可以将来自生成方的数据直接推送至 Amazon Kinesis Streams。举例来说，大家可以将系统与应用程

序日志提交至 Amazon Kinesis Streams，并访问该数据流以在数秒内完成处理。这将避免日志数据在前端或者应用程序服务器发生故障时遭遇丢失，亦可降低对数据源处本地日志存储容量的需求。Amazon Kinesis Streams 提供加速型数据纳入机制，这意味着大家不需要在提交前进行数据批量处理即可完成纳入。

- **实时指标与报告**– 大家可以在 Amazon Kinesis Streams 当中使用提取到的数据，利用各项指标并生成 KPI 以实时生成报告与仪表板。这意味着应用程序逻辑能够在数据流持续交付的同时进行数据处理，而不再需要等待数据批量抵达。

成本模型

Amazon Kinesis Streams 采取简单的按使用量付费形式，其不存在任何前期成本或者最低消费门槛，且大家只需要承担所使用资源量带来的成本。Amazon Kinesis 流由一到多个片段构成，每个片段能够为用户提供每秒 5 次读取事务，且每秒数据读取量最高为 2 MB。每个片段能够支持每秒最高 1000 次写入事务，且每秒最高数据写入量可达 1 MB。

数据流的数据容量为大家为该流指定的片段数量。该数据流总体容量为各片段容量的加和。其中只包含两种计费组成部分，其一为每片段每小时费用，其二为每 100 万 PUT 事务费用。欲了解更多信息，请参阅 Amazon Kinesis Streams 计费页面。[13] 运行在 Amazon EC2 之上且负责处理各 Amazon Kinesis 数据流亦会带来标准的 Amazon EC2 成本。

性能

Amazon Kinesis Streams 允许大家以片段的形式选择必要的数据吞吐容量。利用 Amazon Kinesis 数据流内的各个片段，大家可以以每秒 1000 次写入事务实现每秒最高 1 MB 数据捕捉。大家的 Amazon Kinesis 应用还能够从每个片段处读取数据，具体速率为每秒 2 MB。大家亦随意根据需要配置对应的片段量以实现吞吐容量; 举例来说，每秒 1 GB 数据流要求 1024 个片段。

持久性与可用性

Amazon Kinesis Streams 能够在单一 AWS 服务区内的三个可用区之间以同步方式进行数据

复制，从而实现高可用性与数据持久性。另外，大家也可以在 DynamoDB 当中存储一套探针，从而利用其持续追踪 Amazon Kinesis 数据流内的信息。如果大家的应用程序在自该数据流处读取数据时发生故障，则可重启自己的应用程序并利用该探针了解应用当中引发故障的具体来源。

可扩展性与弹性

大家可以随意根据业务或者运营需求增加或者降低数据流容量，且无需对当前数据流处理造成任何影响。通过使用 API 调用或者开发工具，大家能够自动实现 Amazon Kinesis Streams 环境进行规模伸缩，从而满足具体需求并确保只为实际使用的资源付费。

接口

Amazon Kinesis Streams 提供两种接口：其一由数据生成方使用，负责将提取数据并交付至 Amazon Kinesis Streams 当中; 其二负责处理并分析导入的数据。供应方可利用 Amazon Kinesis PUT API 进行数据写入，利用 AWS 软件开发套件（简称 SDK）或者工具包实现 [14] 抽象，同时使用 Amazon Kinesis Producer Library（简称 KPL）[15] 或者 Amazon Kinesis Agent。[16]

要对已经存在于 Amazon Kinesis 数据流中的数据进行处理，大家可以使用内置客户端库以构建并运营实时数据流应用程序。其中 KCL[17] 负责作为 Amazon Kinesis Streams 与业务应用程序之间的中间机制，其中包含特定处理逻辑。另外，大家亦可将读取自 Amazon Kinesis 数据流的数据通过 Amazon Kinesis Storm Spout[18] 导入至 Apache Storm。

反模式

Amazon Kinesis Streams 拥有以下几种反模式：

- **小规模一致性性吞吐量** – 虽然专门面向每秒速率低于 200 KB 的小型 Amazon Kinesis Streams，其同时亦针对大规模数据吞吐容量进行了设计与优化。
- **长期数据存储与分析** – Amazon Kinesis Streams 并不适合用于长期数据存储。在默

认情况下，数据的保留时长为 24 小时，而且大家可以将保留周期延长至 7 天。大家亦可以将任何必要存储周期长于 7 天的数据迁移至其它持久性存储服务当中，具体包括 Amazon S3、Amazon Glacier、Amazon Redshift 或者 DynamoDB。

AWS Lambda

AWS Lambda[19] 允许大家随时运行代码，且无需配置或者管理服务器设备。大家只需要为实际使用的计算资源付费——代码在未运行时段不会带来任何成本。利用 Lambda，大家几乎能够运行任何应用程序或者后端服务类型的相关代码——且无需任何管理工作。只需要将代码上传至云端，Lambda 即可处理一切以高可用性方式运行并扩展代码的后续任务。大家亦可以设置代码以由其它 AWS 服务进行自动化触发，或者直接立足其它 Web 或者移动应用对其加以调用。

理想的使用模式

Lambda 允许大家执行代码以响应各类触发机制，例如变更数据内容、切换系统状态或者执行用户操作。Lambda 能够由各类 AWS 服务直接触发，具体包括 Amazon S3、DynamoDB、Amazon Kinesis Streams、Amazon Simple Notification Service（即 Amazon 简单通知服务，简称 Amazon SNS）以及 Amazon CloudWatch 等等，允许大家构建多种实时数据处理系统。

- **实时文件处理** – 大家可以触发 Lambda 以调用相关流程，以处理已上传至 Amazon S3 或者经过修改的文件。举例来说，大家可以将已经上传至 Amazon S3 中的图像由彩色变更为黑白形式。
- **实时数据流处理** – 大家可以利用 Amazon Kinesis Streams 与 Lambda 以处理各类实时流数据，具体包括点击流分析、日志过滤以及社交媒体分析。
- **提取、转换与加载**– 大家可以利用 Lambda 以运行数据转换任务，并将其由一套数据

存储库加载至另一存储库。

· **替代 cron**– 利用规划表达式定期运行一项 Lambda 函数，这一解决方案在使用成本及可用性方面要优于在 EC2 实例中运行 cron。

· **处理 AWS 事件** – 众多其它服务，例如 AWS CloudTrail，能够作为事件源发挥作用，即在 Amazon S3 当中进行日志记录并利用 S3 存储桶通知机制以触发各类 Lambda 函数。

成本模型

利用 Lambda，大家只需要为自己的实际使用资源付费。另外，大家需要根据各项函数的请求数量以及代码执行时间计算成本支出。Lambda 免费层当中包含每月 100 万条免费请求以及每月 40 万 GB-秒计算次数。在超出的部分，每 100 万条请求使用成本为 0.20 美元（折合每条请求 0.0000002 美元）。另外，代码执行阶段还需要根据对应分配内存计算使用成本。再有，大家所使用的每 GB-秒成本为 0.00001667 美元。欲了解更多信息，请参阅 AWS Lambda 计费页面。

性能

在首次将代码部署至 Lmbda 当中时，大家的函数通常可在数秒内进行上传调用。Lambda 的设计作用在于在毫秒级别处理各类事件。另外，延迟可能会在 Lambda 函数创建、更新或者最近发生不可用问题时出现暂时的升高。

持久性与可用性

Lambda 的设计目标在于利用复制与冗余机制为服务本身以及其操作的 Lambda 函数提供高可用性。二者不存在任何维护窗口或者计划停机时间。当出现故障时，Lambda 函数将接受同步调用以响应意外状况。Lambda 函数的同步调用至少会尝试 3 次，在此之后事件可能会被直接拒绝。

可扩展性与弹性

大家能够运行的 Lambda 函数数量并无限制。然而，Lambda 自身拥有一项默认安全阈值，

设定为每服务区每账户 100 项并发执行。AWS 支持团队的成员能够根据客户需求提升这一限制。

Lambda 在设计中能够自动根据操作活动进行规模伸缩。函数的规模伸缩不会造成任何费用。Lambda 会动态分配容量，从而匹配输入事件的实际速率。

接口

Lambda 函数可通过多种方式接受管理。大家可以通过 Lambda 控制台中的仪表板轻松罗列、删除、更新以及监控自己的各项 Lambda 函数。大家也可以使用 AWS CLI 以及 AWS SDK 以管理自己的 Lambda 函数。

大家可以通过 AWS 事件触发一项 Lambda 函数，例如 Amazon S3 存储桶通知、DynamoDB Streams、CloudWatch 日志、Amazon SES、Amazon Kinesis Streams、Amazon SNS、Amazon Cognito 等等。任意服务中的任何支持 CloudTrail 的 API 调用都可作为 Lambda 中的事件进行处理，即响应 CloudTrail 审计日志。欲了解更多与事件源相关的细节信息，请参阅核心组件：AWS Lambda 函数与事件源。[20]
Lambda 支持多种编程语言，具体包括 Java、Node.js 以及 Python。大家的代码能够包含现有库甚至是原生库。Lambda 函数可利用 Amazon Linux AMI[21] 支持的语言轻松启动相关流程，包括 Bash、Go 与 Ruby。欲了解更多信息，请参阅 Node.js[22]、Python[23] 以及 Java[24] 说明文档。

反模式

Lambda 拥有以下几种反模式：

- **长期运行应用程序**– 每项 Lambda 函数都必须在 300 秒之内完成。对于要求相关任务运行时长超过 5 分钟的长期运行应用程序，Amazon EC2 将成为更为合适的载体。另外，大家也可以创建 Lambda 函数链，其中函数 1 调用函数 2，后者再调用函数 3，从而实现流程延长效果。
- **动态网站**– 尽管我们可以利用 Lambda 运行静态网站，但运行高度动态且规模较大的网站可能会遭遇性能问题。我们建议大家使用 Amazon EC2 以及 Amazon CLoudFront 以支撑此类用例。
- **有状态应用程序**–Lambda 代码必须以“无状态”方式编写，也就是说其应当假定不依赖于底层计算基础设施。本地文件系统访问、子进程以及同样的机制无法超出请求本身的生命周期，且任何持久性状态皆应被存储于 Amazon S3、DynamoDB 或者其它可用于互联网的存储服务当中。

Amazon EMR

Amazon EMR[25] 是一项高度分布式的计算框架，可以极具成本效益的方式轻松处理及存储数据。Amazon EMR 利用 Apache Hadoop，一套开源框架，将大家的数据进行分布并跨越一整套可随意调整大小的 Amazon EC2 实例集群进行处理，同时允许用户利用 Hive、Pig 以及 Spark 等常见的 Hadoop 工具。Hadoop 提供的框架能够运行大数据处理与分析任务，而 Amazon EMR 则负责处理余下的各类基础设施与 Hadoop 集群软件的配置、管理以及维护工作。

理想的使用模式

Amazon EMR 的灵活框架能够将大规模处理任务与数据集拆分为小型任务，并将其分发至同一 Hadoop 集群当中的多个计算节点之上。这种能力使其能够为大数据分析提供多种使用模式。以下为其中的部分示例：

- 日志处理与分析
- 大规模数据的提取、转换与加载（简称 ETL）
- 风险建模与威胁分析
- 广告定向与点击流分析
- 基因组分析
- 可预测分析
- 特定数据挖掘与分析

欲了解更多相关信息，请参阅 Amazon EMR 最佳实践[26] 白皮书。

成本模型

利用 Amazon EMR，大家可以启动一套持久性集群以处理长期任务，或者建立临时性集群并确保其在分析任务结束后自动关闭。无论选择哪种使用方式，大家都只需要在集群处于运行时根据小时数进行付费。

Amazon EMR 支持多种 Amazon EC2 实例类型（包括标准、高 CPU、高内存、高 I/O 等等）以及全部 Amazon EC2 计费选项（按需、保留与焦点等）。在启动一套 Amazon EMR 集群时（亦被称为‘任务流’），大家可以在配置中选择 Amazon EC2 实例的类型与具体数量。Amazon EMR 使用成本与 Amazon EC2 实例成本相互叠加。欲了解更多相关信息，请参阅 Amazon EMR 计费页面。[27]

性能

Amazon EMR 性能主要受到您选定运行自有集群的 EC2 实例的具体类型以及运行分析任务的实例数量决定。大家应当选择适合自己处理要求的实例类型，同时配合充足的内存、存储与处理资源。欲了解更多与 EC2 实例规格相关的信息，请参阅 Amazon EC2 实例类型。[28]

持久性与可用性

在默认情况下，Amazon EMR 具备核心节点故障容错能力，并能够在从节点下线后继续保持任务执行。目前，Amazon EMR 还无法自动配置其它节点以实现从节点的故障转移，但客户能够监控各节点的运行状态并利用 CloudWatch 替换发生故障的节点。

为了协助处理主节点故障等意外状况，我们建议大家将数据备份至其它持久性存储服务当中，例如 Amazon S3。另外，大家也可以将 Amazon EMR 与 MapR 发行版配合运行 [29]，后者提供的非 NameNode 架构能够利用自动化故障转移与故障回滚机制实现多种故障同时出现时的容错性。另外，元数据亦经过分发与复制，与普通数据处理方式相同。利用这种非 NameNode 架构，文件存储量将不再存在特定限制，且不会对外部网络附加存储机制产生依赖性。

可扩展性与弹性

利用 Amazon EMR，大家能够轻松对运行中的集群进行规模伸缩。[30] 大家也可以随时添加包含有 Hadoop 分布式文件系统（简称 HDFS）的核心节点，从而提升处理性能并增加 HDFS 存储容量（及数据吞吐能力）。另外，大家也可以以原生方式使用 Amazon S3，或者配合 EMFS 或者本地 HDFS，从而将内存与计算资源从存储体系中解耦出来，最终实现更为出色的灵活性与成本效益。

大家也可以随时添加及移除负责处理 Hadoop 任务的任务节点，但不再维持 HDFS。部分客户会在需要进行批量处理时向集群当中加入数百个实例，并在任务完成后将多余实例移除。举例来说，大家可能不太确定自己在未来 6 个月中需要处理多少数据，或者是峰会处理资源需求。利用 Amazon EMR，大家不再需要猜测此类要求或者配置峰值资源——因为我们能够随时轻松添加或者移除各类资源容量。

另外，大家也可以添加多套规模各异的新集群，并随时在控制台中进行数次点击或者使用编程 API 调用 [31] 将其轻松移除。

接口

Amazon EMR 支持 Hadoop 基础之上的多种工具，其可用于大数据分析且各自拥有自己的接口。下面来看其中最具人气的几种选项：

Hive

Hive 是一套开源数据仓库及分析软件包，其运行在 Hadoop 基础之上。Hive 由 Hive QL 负责操作，这是一种基于 SQL 的语言，允许用户对数据进行架构设计、整理与查询。Hive QL 在标准 SQL 之外还添加了一流的映射/规约功能支持，并能处理多种复杂的用户定义数据类型，例如 JSON 与 Thrift。这种能力允许大家处理各类复杂的非结构化数据源，例如文本文档及日志文件。

Hive 允许用户通过以 Java 编写的用户定义函数实现扩展。Amazon EMR 则对 Hive 做出了一系列改进，其中包括与 DynamoDB 与 Amazon S3 的直接集成。举例来说，大家可以利用 Amazon EMR 自动从 Amazon S3 中加载表内容，无需利用临时文件向 Amazon S3 中的表内写入数据，同时访问 Amazon S3 中的各类资源——包括自定义映射及/或规约操作脚本及其它库。欲了解更多信息，请参阅 Amazon EMR 发布指南中的 Apache Hive 说明 [32]。

Pig

Pig 是一套开源分析软件包，其运行在 Hadoop 基础之上。Pig 由 Pig Latin 负责操作，这是一种 SQL 类语言，允许用户对数据进行架构设计、整理与查询。除了 SQL 类操作之外，Pig Latin 还添加了一流的映射/规约功能支持，并能处理多种复杂的用户定义数据类型。这种能力允许大家处理各类复杂的非结构化数据源，例如文本文档及日志文件。

Pig 允许用户通过以 Java 编写的用户定义函数实现扩展。Amazon EMR 则对 Pig 做出了一系列改进，其中包括使用多种文件系统的能力（通常情况下，Pig 只能访问单一远程文件系统）、从 Amazon S3 处加载客户 JAR 与脚本的能力（例如‘REGISTER s3://my-bucket/piggybank.jar’）以及额外的 String 与 DateTime 处理能力。欲了解更多相关信息，请参阅 Amazon EMR 发布指南中的 Apache Pig 说明。[33]

Spark

Spark 是一套开源数据分析引擎，以 Hadoop 为基础构建并具备内存内 MapReduce 功能。Spark 能够为特定分析任务提供额外的速度提升，且已经成为 Shark（SQL 驱动型数据仓库）、Spark Streaming（流应用）、GraphX（图形系统）以及 MLlib（机器学习）等强大工具的实现基础。欲了解更多信息，请参阅在 Amazon EMR 集群之上安装 Apache Spark 博文。[34]

HBase

HBase 是一套开源、非关系型分布式数据库，其模型脱胎于谷歌的 BigTable。HBase 最初作为 Apache 软件基金会的 Hadoop 项目组成部分，同时运行在 Hadoop 分布式文件系统（简称 HDFS）之上以向 Hadoop 提供 BigTable 类功能。HBase 具备出色的容错性，

且能够利用列式压缩与存储机制高效存储大规模离散数据。另外，HBase 还能够实现数据快速查找，这是因为其数据被存储在内存当中而非磁盘之上。

HBase 针对连续写入操作进行了优化，因此能够高效处理批量插入、更新与删除等任务。HBase 还能够与 Hadoop 无缝对接，共享文件系统并作为 Hadoop 任务的直接输入与输出内容。HBase 亦可与 Apache Hive 相集成，用于通过 HBase 表实现 SQL 类查询、加入 Hive 表以及支持 Java 数据库连接（简称 JDBC）。在配合 Amazon EMR 的情况下，大家可以将 HBase 备份至 Amazon S3（完全或者增量，手动或者自动），并利用原有备份进行恢复。欲了解更多细节信息，请参阅 Amazon EMR 开发者指南中的 HBase 与 EMR 说明。[35]

Impala

Impala 是一款来自 Hadoop 生态系统的开源工具，可利用 SQL 语法实现交互与临时查询。相较于使用 MapReduce，Impala 利用一套大规模并发处理（简称 MPP）引擎实现了与传统关系型数据库管理系统（简称 RDBMS）类似的效果。在这套架构当中，大家可以在 HDFS 或者 HBase 表中快速查询数据，同时利用 Hadoop 的强大功能处理离散数据类型并在运行时中实现模式。这使得 Impala 成为一款执行交互式低延迟分析任务的出色工具。

Impala 还允许用户利用 Java 及 C++语言进行函数定义，同时可通过 ODBC 接入商务智能工具以及 JDBC 驱动程序。Impala 利用 Hive 元存储以保存与输入数据相关的信息，其中包括分区名称以及数据类型。欲了解更多信息，请参阅 Amazon EMR 开发者指南中的 Impala 与 EMR 说明。[36]

Hunk

Hunk 由 Splunk 开发，其使得机器数据可访问性、可用性及价值能够为每位用户服务。在 Hunk 的帮助下，大家可以对存储于 Amazon EMR 以及 Amazon S3 中的数据进行探索、分析与可视化处理，从而在 Hadoop 之上实现 Spunk 分析。欲了解更多信息，请参阅 Amazon EMR 与 Hunk：面向 Hadoop 与 NoSQL 的 Spunk 分析。[37]

Presto

Presto 属于一套开源分布式 SQL 查询引擎，其专门针对数据的低延迟临时性分析做出了优化。其支持 ANSI SQL 标准，具体包括复杂的查询、聚合、加入与窗口功能。Presto 能够对来自多个来源的数据进行处理，具体包括 Hadoop 分布式文件系统（简称 HDFS）与 Amazon S3。

其它第三方工具

Amazon EMR 还支持多种其它来自 Hadoop 生态系统的高人气应用与工具，其中包括 R（统计）、Mahout（机器学习）、Ganglia（监控）、Accumulo（安全 NoSQL 数据库）Hue（用于分析 Hadoop 数据的用户界面）、Sqoop（关系型数据库连接机制）以及 HCatalog（表与存储管理）等等。

另外，大家也可以在 Amazon EMR 之上安装自有软件以帮助解决实际业务需求。AWS 能够快速将大量数据由 Amazon S3 迁移至 HDFS、由 HDFS 迁移至 Amazon S3 并利用 Amazon EMR 的 S3DistCp[38] 在不同 Amazon S3 存储桶间进行数据转移。作为开源工具 DistCp 的扩展方案，S3DistCp 能够利用 MapReduce 高效实现大规模数据移动。

大家也可以选择使用 EMR 文件系统（简称 EMRFS），这套 HDFS 实现方案允许 Amazon EMR 集群将数据存储在 Amazon S3 之上。另外，我们也能够启用 Amazon S3 的服务器端与客户端加密，并继续保持对 EMRFS 的快速查看。在使用 EMRFS 时，元数据会临时存储于 DynamoDB 当中以帮助管理其同 Amazon S3 之间的交互，允许大家轻松利用同一套 EMRFS 元数据与 Amazon S3 上的存储支撑多套 EMR 集群。

反模式

Amazon EMR 拥有以下几种反模式：

- **小型数据集**– Amazon EMR 专门面向大规模并行处理所构建; 如果大家的数据集较小且能够快速运行在单一设备之上、单一线程当中，那么应当将其容纳于单一系统内，从

而避免 EMR 带来的任务映射与规约资源占用。

· **ACID 事务要求**– 尽管提供多种 ACID（即原子性、一致性、隔离性与持久性）实现方式或者在 Hadoop 之内有限支持 ACID，但其它数据库——例如 Amazon RDS 或者运行在 Amazon EC2 之上的关系型数据库——可能更适合处理存在严格要求的工作负载。

Amazon Machine Learning

Amazon ML[39] 是一项用于简化预测性分析与机器学习技术的服务。Amazon ML 可视化工具及向导能够帮助大家通过一系列流程完成机器学习（简称 ML）模型的创建，且无需掌握复杂的机器学习算法与技术。在模型准备就绪之后，Amazln ML 能够轻松利用 API 操作为应用程序提供预测支持，且无需使用自定义预测生成代码或者管理任何基础设施。

Amazon ML 能够根据存储在 Amazon S3、Amazon Redshift 或者 Amazon RDS 中的数据创建机器学习模型。内置向导则可帮助大家经由一系列步骤以交互方式探索自己的数据、训练机器学习模型或者评估当前模型质量并根据业务目标调整输出结果。在模型完成之后，大家可以通过批量方式或者使用低延迟实时 API 请求预测结论。

理想的使用模式

Amazon ML 非常适合立足于数据进行模式发现，同时利用这些模式创建新的机器学习模型以生成与潜在数据点相关的预测结果。举例来说，大家可以：

· **利用应用程序标记可疑事务** – 构建一套机器学习模型，用于预测某一新事务是否合法。

· **预测产品需求**– 将历史订单信息作为输入信息，用于预测未来订单数量。

· **个性化应用程序内容** – 预测用户可能与哪些条目交互，并以实时方式从应用中检索这些预测结论。

- **预测用户行为**– 分析用户行为以定制网站并提供更佳用户体验。
- **听取社交媒体意见**– 采集并分析社交媒体信息，并据此做出业务决策。

成本模型

利用 Amazon ML，大家只需要为自己实际使用的资源付费。此项服务不设最低消费门槛，亦无需承担前期成本。Amazon ML 按照构建预测模型的计算实际时长（以小时为单位）进行计费。要实现实时预测，大家还需要以小时为单位基于模型运行所需要的内存总量付费。

数据分析、模型训练与基于计算-小时数量的计费评估需要根据输入数量大小、其中属性数量以及所使用的转换类型确定。数据分析与建模构建成本为每小时 0.42 美元。预测费用可按照批量与实时两种方式进行分类。批量预测为每 1000 项预测 0.10 美元，以每 1000 项为单位；而实时预测为每条预测 0.0001 美元，以四舍五入方式计算（最低计费单位为美分）。在实时预测当中，大家也可以 10 MB 为内存单位进行模型配置，其中每 10 MB 内存计费为每小时 0.001 美元。

在模型创建过程中，大家需要为每套模型指定最大内存分配量，从而管理成本并保证性能的可预测性。另外，大家只需要在整套模型执行实时预测的过程中支付费用。另外，存储于 Amazon S3、Amazon RDS 以及 Amazon Redshift 当中的数据须独立计算成本。欲了解更多细节信息，请参阅 Amazon Machine Learning 计费页面。[4]

性能

创建模型或者立足于这些模型请求批量预测所耗费的具体时间，取决于您的输入数据记录数量、类型以及这些记录中的属性分布，同时亦涉及您所指定使用的数据处理手段的复杂程度。

大部分实时预测请求会在 100 ms 之内返回响应结果，这样的速度水平已经足以应对交互

式 Web、移动或者桌面应用程序。而负责生成预测结论的实时 API 产生的额外时间消耗则取决于输入数据记录的大小，以及生成预测结果的机器学习模型所使用的数据处理“模板”[41] 的实际复杂程度。每套能够支持实时预测功能的机器学习模型在默认情况下每秒最多请求 200 项事务，且这一数字可通过客户支持合同进行增加。大家可以利用 CloudWatch 指标对机器学习模型的预测请求数量进行监控。

持久性与可用性

Amazon ML 在设计中充分考虑到高可用性要求。其中不存在任何维护容器或者计划内停机时间。该服务运行在经过严格规划的 Amazon 高可用性数据中心之内，且服务堆栈副本被配置至同一 AWS 服务区内的三座设施当中，从而在发生服务器故障或者可用区宕机时提供容错性。

可扩展性与弹性

大家可以利用机器学习模型处理最高 100 GB 数据集，或者以此为基础请求批量预测。对于大规模批量预测任务，大家可以将自己的输入数据记录拆分为多个分段，从而实现超越默认上限的数据量处理。

在默认情况下，大家可以同时运行最高 5 项任务，并通过客户服务合同提升这一数量上限。由于 Amazon ML 是一项托管服务，因此大家不需要进行服务器配置，意味着用户无需过度配置资源或者为额外资源付费，即可轻松扩展实际应用规模。

接口

数据源创建可通过多种方式实现，最简单的办法为将数据添加至 Amazon S3 当中，或者直接从 Amazon Redshift 或者由 Amazon RDS 负责托管的 MySQL 数据库中提取数据。在数据源定义完成后，大家可以利用控制台同 Amazon ML 进行交互。在这里，我们能够使用 AWS SDK 以及 Amazon ML API[42] 以编程化方式访问 Amazon ML。另外，大家也可以利用面向 Windows、Mac 以及 Linux/UNIX 系统平台的 AWS CLI 实现 Amazon ML

实体的创建与管理。

反模式

Amazon ML 拥有以下几种反模式：

- **过于庞大的数据集**– 尽管 Amazon ML 能够支持最高 100 GB 数据，但目前尚无法支持 TB 级别数据的提取。在此类用例当中，客户通常会选择使用 Amazon EMR 以运行 Spark 的机器学习库（简称 MLlib）。
- **不受支持的学习任务**– Amazon ML 可用于创建多种机器学习模型，包括负责二元分类（从两个选项中选择一个，并提供良好的结果准确性）、多类分类（即从两个以上选项中进行选择）或者数字递归（直接预测一个数字）等。但机器学习可能无法支持某些任务类型，例如序列预测或者无监督集群，此类任务可利用 Amazon EMR 运行 Spark 及 MLlib 完成。

Amazon DynamoDB

Amazon DynamoDB[43] 是一项速度极快且完全托管的 NoSQL 数据库服务，能够帮助用户轻松高效地存储并检索任意规模的数据，同时处理任意级别的请求流量。DynamoDB 有助于减轻高可用性分布式数据库集群的日常运营与规模调整负担。这套存储方案提供个位数毫秒延迟且立足于无缝化吞吐及存储扩展性提供可预测性能水平，因此能够满足敏感型应用对延迟及吞吐能力的实际要求。

DynamoDB 将结构化数据存储于表当中，利用主键进行索引，并允许用户对大小在 1 字节到 400 KB 之间的条目进行低延迟读取与写入访问。DynamoDB 支持三种数据类型（数字、字符串与二进制），且基于标量及多值两种集合。其支持在这些数据类型中存储 JSON、XML 以及 HTML 等文件格式。此类表不存在固定模式，因此各个数据条目皆可拥有不同的属性数量。其主键则可为单一属性哈希键，或者复合型哈希范围键。

DynamoDB 能够提供全局与本地次级索引，用于针对主键之外的属性实现额外的查询灵活性。DynamoDB 提供最终一致性读取（默认设置）以及强一致性读取（可选），同时亦能够在条目层级实现条目放置、更新、删除、条件操作以及递增/递减等操作。

DynamoDB 可与其它服务配合使用，具体包括 Amazon EMR、Amazon Redshift、AWS Data Pipeline 以及 Amazon S3 等，从而实现分析、数据仓库、数据导入/导出、备份以及归档等功能。

理想的使用模式

DynamoDB 适合处理各类需要低读取与写入延迟及高灵活性 NoSQL 数据库的现有或者新型应用程序，且其能够随时根据需要实现存储与吞吐能力资源的规模伸缩——无需任何代码变更或者停机时间。

常见用例包括：

- 移动应用
- 游戏
- 数字化广告投放
- 实时投票
- 现场活动观众互动
- 传感器网络
- 日志提取
- 对 Web 内容进行访问控制
- 为 Amazon S3 对象提供元数据存储
- 电子商务购物车

- Web 会话管理

大部分此类用例要求高可用性及可扩展数据库，因为任何停机或者性能下降都可能对企业业务造成即时的负面影响。

成本模型

利用 DynamoDB，大家只需要为自己实际使用的资源量付费，且不存在最低消费门槛。DynamoDB 拥有三种计费组成部分：配置数据吞吐容量（每小时）、索引数据存储（每月每 GB）以及数据传入或者传出（每月每 GB）。新客户能够将 DynamoDB 作为 AWS 免费使用层 [44] 中的组成部分进行免费体验。[45]

性能

SSD 与对属性索引的限制实现高吞吐量以及低延迟水平 [46]，且能够大幅降低读取与写入操作成本。随着数据集合的增长，可预测性能将成为一大必要前提，从而保证工作负载具备低延迟。这种可预测性能可通过面向给定表的必要吞吐容量配置进行定义。

立足内部机制，该服务能够处理资源配置任务以提供所需要的数据吞吐速率；大家不需要考虑实例、硬件、内存乃至其它可能影响到应用程序的吞吐容量。所配置的吞吐容量具备弹性，且能够随时根据需求提升或者下调。

持久性与可用性

DynamoDB 具备内置容错机制，能够以自动化方式在同一服务区内的三座数据中心之间进行数据同步复制，从而实现高可用性并有助于保护数据免受单一乃至整体设施故障的影响。DynamoDB Streams[47] 能够捕捉发生在表中的全部数据活动，并允许大家立足各个地理区域之间设置区域副本，旨在提供更为理想的可用性水平。

可扩展性与弹性

DynamoDB 同时具备高可扩展性与弹性。大家能够在一套 DynamoDB 表内存储的数据

量不设上限，且该服务能够自动利用 DynamoDB 写入 API 操作为数据分配更多存储资源。另外，数据资源能够根据需要自动或者重新分配，SSD 的引入则能够在任意规模下提供可预测的低延迟响应时间。该项服务亦具备弹性，大家能够借此轻松“拨上”[48] 或者“拨下”[49] 需要变更的表的读取与写入容量。

接口

DynamoDB 提供一项低级 REST API，同时亦提供面向 Java、.NET 以及 PHP 等打包有该低级 REST API 的高级 SDK，并借此实现部分对象关系映射（简称 ORM）功能。这些 API 可为 DynamoDB 提供一硕管理与数据接口。该 API 目前可实现表管理（包括元数据的创建、罗列、删除以及获取）以及属性处理（包括属性的获取、写入与删除; 利用索引实现的查询以及全面扫描）。

虽然不支持标准 SQL，但大家可以利用 DynamoDB 的 select 操作以创建 SQL 类查询，从而基于所提供的标准对属性组进行检索。另外，大家亦能够利用控制台与 DynamoDB 进行协作。

反模式

DynamoDB 拥有以下反模式：

- **需要依赖于传统关系型数据库的预编写应用程序** – 如果大家尝试将现有应用程序移植到 AWS 云，同时需要继续使用关系型数据库，则可选择使用 Amazon RDS（包括 Amazon Aurora、MySQL、PostgreSQL、甲骨文或者 SQL Server）或者多种预配置 Amazon EC2 数据库 AMI 之一。大家也能够将自己选定的数据库软件安装至 EC2 实例当中。
- **加入或者复杂事务**– 虽然多数年解决方案都能够利用 DynamoDB 支持其正常使用，但大家的实际应用可能需要配合由传统数据库平台提供的加入、复杂事务及其它关系型基础设施。在这种情况下，大家可以利用 Amazon Redshift、Amazon RDS 或者 Amazon EC2 配合一套自管理数据库方案。

- **大型二进制对象（简称 BLOB）数据** – 如果大家打算存储大规模（超过 400 KB）的 BLOB 数据，例如数字视频、图像或者音乐，则可以考虑使用 Amazon S3。然而，DynamoDB 仍然能够在此类场景中发挥一定作用，即对与二进制对象相关的元数据进行追踪（例如追踪其条目名称、大小、创建数据、拥有者及位置等等）。
- **低 I/O 速率大规模数据** –DynamoDB 利用 SSD 驱动器且针对高 I/O 速率进行工作负载优化。如果大家希望存储大量不需要频繁访问的数据，则应优先考虑使用其它选项，例如 Amazon S3。

Amazon Redshift

Amazon Redshift[50] 是一项快速、全面托管的 PB 级别数据仓库服务，其能够以轻松且极具成本效益的方式利用现有商务智能工具高效分析全部数据。其针对多种数据集规模进行优化，范围由数百 GB 到 PB 级别，且能够提供仅相当于传统数据仓库解决方案十分之一的成本。

Amazon Redshift 能够利用列式存储技术为几乎规模的任意数据集提供高速查询与 I/O 性能，同时以并行及分布方式跨越多个节点进行查询。其能够自动处理大部分常见管理任务，包括设置、配置、监控、备份以及数据仓库的安全保护等，从而有效降低相关体系的管理与维护难度及成本。这种自动化机制允许大家在数分钟之内构建 PB 级别数据仓库，而无需像传统内部实现方案那样耗时数周甚至数月。

理想的使用模式

Amazon Redshift 适合处理在线分析处理（简称 OLAP）并可继续使用大家的现有商务智能工具。企业能够利用 Amazon Redshift 实现以下任务：

- 面向多款产品分析全局销售数据。
- 存储历史股票交易数据。

- 分析广告展示与点击情况。
- 聚合游戏数据。
- 分析社交趋势。
- 衡量医疗质量、运营效率以及医疗卫生工作中的财务业绩。

成本模型

Amazon Redshift 数据仓库集群不需要客户承诺长期使用或者承担前期成本。这意味着大家能够避免资本投入以及超前规划数据仓库容量与采购任务的复杂性。其收费机制基于集群的节点大小与具体数量。

在使用过程中，您只需要为实际使用的备份存储容量付费，且付费额度不会超过您的实际配置容量。举例来说，如果您构建起包含 2 个 XL 节点的活动集群，总存储容量为 4 TB，那么 AWS 则会在 Amazon S3 之上为您提供 4 TB 备份存储资源，除此之外再无任何支出。超出配置存储容量之外的备份存储资源以及集群关闭后继续保留的备份数据，会按照标准 Amazon S3 费率进行计费 [51]。Amazon S3 与 Amazon Redshift 之间的数据通信不产生任何费用。欲了解更多细节信息，请参阅 Amazon Redshift 计费页面。[52]

性能

Amazon Redshift 利用一系列创新成果提供出色的性能水平，且数据集规模可由数百 GB 到 PB 级别浮动。其利用列式存储、数据压缩以及区域映射等机制以降低执行查询所需要的 I/O 数量。

Amazon Redshift 拥有一套大规模并发处理（简称 MPP）架构，其能够实现 SQL 操作的并发与分布，从而充分发挥全部可用资源所蕴含的优势。其底层硬件专门针对高性能数据处理所设备，能够利用本地附加存储资源以最大程度提升 CPU 与驱动器间的数据吞吐能力，同时利用 10 GigE 大型网络保证各节点间的顺畅通信。大家可以根据数据仓库的实际需求调整性能：AWS 提供配合 SSD 驱动器的密集计算（简称 DC）与密集存储（简称 DS）

两种选项。

持久性与可用性

Amazon Redshift 能够自动检测并替换大家数据仓库集群当中的故障节点。该数据仓库集群将始终保持只读状态，直到替换节点配置完成并被添加至数据库当中，整个过程一般需要数分钟时间。Amazon Redshift 能够使替换节点即时可用，且首先导流 Amazon S3 当中访问频率最高的数据，从而尽可能快地恢复数据查询能力。

另外，大家的数据仓库集群在驱动器发生故障时也依然可用；这是因为 Amazon Redshift 能够跨越整套集群进行数据镜像，因此系统能够利用来自其它节点的数据完成故障驱动器重构。Amazon Redshift 各集群存在于同一可用区 [53] 当中，但如果大家希望设置一套多可用区 Amazon Redshift 体系，则可以首先建立镜像而后以自动管理方式完成复制与故障转移。

可扩展性与弹性

只需在控制台中进行数次点击或者执行 API 调用 [54]，大家即可在性能或者容量需求发生变化时调整数据仓库内的节点数量或者类型。Amazon Redshift 允许大家首先建立一套最小仅为 160 GB 的节点，而后逐步按需求进行向上扩展，最终甚至可达 PB 级别并包含多个节点。欲了解更多信息，请参阅 Amazon Redshift 管理指南中 Amazon Redshift 集群议题下的“关于集群与节点章节”[55]。

在调整资源规模时，Amazon Redshift 会将现有集群切换为只读模式，而后按照选定的大小配置新集群，并将数据由旧有集群转移至新集群；在这一过程中，大家只需要为处于活动状态的 Amazon Redshift 集群付费。在此之后，大家可以对旧有集群加以查询，而新集群则同步进行配置。数据全部复制至新集群后，Amazon Redshift 会自动将查询重新指向至新集群并移除旧有集群。

接口

Amazon Redshift 拥有 JDBC 与 ODBC 驱动程序，大家可以从控制台的 Connect Client 标签处进行下载，从而使用多种您所熟悉的 SQL 客户端。大家亦能够使用标准 PostgreSQL JDBC 与 ODBC 驱动程序。欲了解更多与 Amazon Redshift 驱动程序相关的细节信息，请参阅 Amazon Redshift 与 PostgreSQL 说明文档。[56]

各类主流商务智能与 ETL 方案供应商 [57] 都能够将自身产品与 Amazon Redshift 进行成功集成。当大家从数据仓库、Amazon S3 以及 DynamoDB 中获取数据时，加载与卸载操作会以并发形式执行从而最大程度提升性能水平。大家亦能够轻松利用 Amazon Kinesis FIrehose 将数据流加载至 Amazon Redshift 当中，从而实现与现有商务智能工具与仪表板相同的近实时分析机制。用于追踪 Amazon Redshift 数据仓库集群性能的 CPU 使用率、内存使用率、存储资源使用率以及读取/写入流量等指标皆可免费通过控制台或者 CloudWatch API 操作进行使用。

反模式

Amazon Redshift 拥有以下几种反模式：

- **小型数据集**– Amazon Redshift 专门针对单一集群内的并发处理任务所设计，因此如果大家的数据集小型 100 GB，将无法充分发挥 Amazon Redshift 所提供的各类优势。在这种情况下，Amazon RDS 可能会是更理想的解决方案。
- **在线事务处理(简称 OLTP)** – Amazon Redshift 的设计目标在于为数据仓库工作负载提供极高速度与低成本分析功能。如果大家需要一套高速事务处理系统，则可以选择运行于 Amazon RDS 之上的传统关系型数据库系统，或者 DynamoDB 等等 NoSQL 数据库选项。
- **非结构化数据** – Amazon Redshift 中的数据必须由经过定义的模式进行结构指定，其无法在各行中支持任意模式结构。如果大家需要处理非结构化数据，则可在 Amazon EMR 之上执行提取、转换与加载（简称 ETL），从而确保向 Amazon Redshift 中导入其能够应

对的数据类型。

· **BLOB 数据**– 如果大家计划存储大型二进制文件（例如数字化视频、图片或者音乐），则可能需要考虑将数据存储在 Amazon S3 当中，并在 Amazon Redshift 内直接引用其位置。在这种情况下，Amazon Redshift 能够持续追踪您二进制对象的元数据（例如条目名称、大小、已创建数据、拥有者以及位置等等）。但再次强调，大型对象本身应当被存储于 Amazon S3 内。

Amazon Elasticsearch 服务

Amazon ES[58] 是一项托管服务，能够帮助用户在 AWS 云中轻松完成 Elasticsearch 的部署、操作与规模调整工作。Elasticsearch 属于一套实时分布式搜索与分析引擎，它允许大家以前所未有的速度与规模探索数据内容。它可用于全文本搜索、结构化搜索、分析并能够将三种功能加以结合。

大家可以利用控制台在数分钟内完成 Amazon ES 集群的设置与配置。Amazon ES 能够管理的任务包括设置一套域，包括配置必要的基础设施资源以及安装 Elasticsearch 软件等等。

在域开始运行之后，Amazon ES 能够自动执行各类常规管理任务，其中包括执行备份、监控实例并为 Amazon ES 实例中的各项软件安装补丁。它还能够自动检测并替换发生故障的 Elasticsearch 节点，利用自管理基础设施与 Elasticsearch 软件降低日常管理开销。这项服务允许大家通过单一 API 调用或者控制台内的数次点击轻松完成集群规模伸缩。

利用 Amazon ES，大家能够直接访问 Elasticsearch 开源 API，从而确保各代码与应用能够同现有 Elasticsearch 环境实现无缝化协同。其还支持与 Logstash 相集成，这是一套开源数据管道，可帮助用户处理日志及其它事件数据。另外，Amazon ES 亦内置 Kibana 支持能力，这套开源的分析与可视化平台能够帮助我们更好地了解自己的数据内容。

理想的使用模式

Amazon ES 适合查询并搜索大规模数据集。企业可以利用 Amazon ES 实现以下任务：

- 分析活动日志，例如面向客户的应用程序或者网站日志。
- 利用 Elasticsearch 对 CloudSatch 日志进行分析。
- 对来自多种服务及系统的产品使用情况数据进行分析。
- 分析社交媒体情感及 CRM 数据，同时发现品牌及产品趋势。
- 对来自其它 AWS 服务的数据流更新进行分析，诸如 Amazon Kinesis Streams 以及 DynamoDB。
- 为客户提供更为丰富的搜索与导航体验。
- 监控移动应用程序的使用情况。

成本模型

利用 Amazon ES，大家只需要为自己实际使用的计算与存储资源付费。其不设最低消费门槛或者前期投入要求。大家按小时、Amazon EBS 存储（如果选择使用）以及共享数据传输费用 [59] 计算 Amazon ES 实例使用成本。

如果大家利用 EBS 分卷进行存储，Amazon ES 允许大家选择具体分卷类型。如果大家选择配置 IOPS（SSD）存储 [60]，则需要承担存储资源本身以及所配置数据吞吐容量两种成本。多大同，大家不需要为所消费的 I/O 付费。另外，您也可以根据域内接入各数据节点的 EBS 分卷的累计大小支付存储资源费用。

Amazon ES 为每个 Amazon ES 域提供具备免费自动快照保存功能的存储空间。手动快照需要根据 Amazon S3 存储费率进行计算。欲了解更多细节信息，请参阅 Amazon Elasticsearch 服务计费页面。[61]

性能

Amazon ES 的实际性能取决于多项因素，具体包括实例类型、工作负载、索引、所用片段数量、读取副本以及存储配置（实例存储或者 EBS 存储，例如通用型 SSD）。索引由多个数据片段构成，且可被分发至多个可用区内的多个实例当中。

如果选择使用区域识别功能，则片段的副本读取操作会跨越不同可用区内的各个 Amazon ES 实例进行。Amazon ES 可利用高速 SSD 实例存储索引，亦可使用多套 EBS 分卷完成这项任务。其提供一套搜索引擎，旨在为高强度存储设备及磁盘提供更快的查询与搜索性能。

持久性与可用性

大家可以在创建域时设置或者随时对实时域进行修改，从而启用区域识别选项以确保 Amazon ES 域拥有高可用性。当区域识别被启用时，Amazon ES 会将用于支持该域的各项实例分发至两套不同可用区当中。在此之后，如果大家在 Elasticsearch 中启用副本，则各实例会自动以此方式分发以作为跨可用区副本。

大家可以通过自动及手动快照保存机制为 Amazon ES 域建立数据持久性。大家能够利用这些快照经由预加载数据恢复原有域，亦可利用预加载数据创建新的域。这些快照被保存在 Amazon S3 当中，这是一项安全、持久且具备高度可扩展性的对象存储服务。在默认情况下，Amazon ES 会每天自动为各个域创建快照。另外，大家也可以使用 Amazon ES 快照 API 以创建额外的手动快照。这些手动快照亦被存储于 Amazon S3 之内。手动快照可用于跨区域灾难恢复并提供额外的持久性保障。

可扩展性与弹性

大家可以添加或者移除实例，同时轻松修改 Amazon EBS 分卷以应对数据量的增长。大家可以编写数行代码以通过 CLoudWatch 指标监控域当前运行状态，同时调用 Amazon ES API 以根据事先设定的阈值进行域规模伸缩。此项服务在执行规模调整时，不会带来任何停机时间。

Amazon ES 支持集群内每实例一套 EBS 分卷（最大容量为 512 GB）。每套 Amazon ES 集群最高可容纳 10 个实例，因此客户可以为单一 Amazon ES 域分配约 5 TB 存储资源。

接口

Amazon ES 支持 Elasticsearch API[62]，因此大家目前已经使用的任何代码、应用程序以及主流工具皆可与当前 Elasticsearch 环境进行无缝化对接。AWS SDK 支持全部 Amazon ES API 操作，这意味着大家可以利用自己熟悉的技术轻松实现与域之间的管理与交互。AWS CLI 或者控制台亦可用于创建及管理域。

Amazon ES 支持与多项 AWS 服务进行集成，其中包括来自 Amazon S3 的数据流、Amazon Kinesis Streams 以及 DynamoDB Streams。这一集成工作利用一项 Lambda 函数作为云环境下的事件处理程序，其负责处理数据并将其引导至 Amazon ES 域。Amazon ES 还能够与 CloudWatch 相整合以监控各项 Amazon ES 域指标，亦可协同 CloudTrail 以审计指向 Amazon ES 域的各配置 API 调用。

Amazon ES 当中内置有 Kibana 集成（这是一套开源分析与可视化平台），同时亦支持同 Logstash 对接——这是一套开源数据管道，可帮助大家处理日志及其它事件数据。您也可以将自己的 Amazon ES 域设置为后端存储机制，从而通过 Logstash 容纳全部输入日志记录，旨在简化来自多个来源的结构化与非结构化数据的收集工作。

反模式

Amazon ES 拥有以下几种反模式：

- **在线事务处理(简称 OLTP)** – Amazon ES 是一套实时分布式搜索与分析引擎。其不支持数据操作类事务或者处理机制。如果大家需要一套高速事务处理系统，那么建立在

Amazon RDS 之上的传统关系型数据库系统或者 DynamoDB 等 NoSQL 数据库可能会是更好的选择。

- **PB 级别存储**– 受到每 Amazon ES 集群只能容纳 10 个实例的最大容量限制，大家只能为单一 Amazon ES 域分配约 5 TB 存储资源。如果工作负载的规模超过这一水平，请考虑使用运行在 Amazon EC2 上的 Elasticsearch。

Amazon QuickSight

2015 年 10 月，AWS 方面推出了 Amazon QuickSight 的预览版本，这项高速、云支持型商务智能（简称 BI）服务能够帮助大家轻松构建起可视化方案、执行即时分析并快速通过数据获取业务洞察结论。

QuickSight 采用一套新型、极快、并发、内存内、计算引擎（简称 SPICE），旨在快速执行高级计算与可视化渲染。QuickSight 能够自动与各类 AWS 数据服务相集成，帮助企业将分析规模提升至数十万用户，同时通过 SPICE 的查询引擎提供高速响应查询性能。由于使用成本仅为传统解决方案的十分之一，QuickSight 允许大家将成本低廉的商务智能功能交付至企业中的每位员工。欲了解更多细节信息并登录至预览版，请参阅 QuickSight 页面。[63]

Amazon EC2

Amazon EC2,[64] 提供的各项实例可看作是 AWS 版本的虚拟机，其提供的理想平台能够有效利用 AWS 基础设施支持大家的自管理大数据分析应用。能够安装在 Linux 或者 Windows 虚拟环境下的几乎全部软件都能够运行在 Amaozn EC2 当中，大家也可以使用充分享受其按实际使用量计费的便利。不过在 EC2 中，大家无法获得本白皮书内提到的应用级托管服务。下面来看部分可供选择的自管理大数据分析解决方案：

- MongoDB 等 NoSQL 方案。

- Vertica 等数据仓库或者列式存储方案。
- Hadoop 集群。
- Apache Storm 集群。
- Apache Kafka 环境。

理想的使用模式

- **专业环境**– 在运行定制化应用程序时，单一方案往往无法涵盖其中的全部标准 Hadoop 组或者应用。Amazon EC2 能够提供出色的灵活性与可扩展性，用于满足大家的具体计算需求。
- **合规性要求**– 特定合规性要求可能要求大家将应用运行在 Amazon EC2 之上，而非作为托管服务进行交付。

成本模型

Amazon EC2 拥有面向多种实例家族的一系列实例类型（包括标准、高 CPU、高内存以及高 I/O 等等），同时提供不同计费选项（按需、保留及焦点）。根据应用程序的具体要求，大家可以使用其它服务配合 Amazon EC2，具体包括 Amazon Elastic Block Store（即 Amazon 弹性块存储，简称 Amazon EBS）以作为直连持久性存储，或者利用 Amazon S3 作为持久性对象存储; 每种选项都拥有自己的计费模式。如果大家在 Amazon EC2 上运行大数据应用程序，则需要像使用内部数据中心一样自行提供授权许可。AWS Marketplace[65] 提供多种第三方大数据软件包，且各项方案全部经过预配置，只需一键点击即可起效。

性能

Amazon EC2 中的性能主要由大家为自身大数据平台选定的具体实例类型决定。每种实例类型拥有不同的 CPU、内在、存储、IOPS 以及网络容量配置，因此大家可以按照实际要求选择正确的性能水平。

持久性与可用性

关键性应用程序应当运行在单一 AWS 服务区中的跨可用区集群之内，从而确保任何实例或者数据中心故障都不会对应用用户造成影响。对于正常运行时间要求较低的应用，大家可以将自己的应用备份至 Amazon S3，并在实例或者可用区发生故障时将其恢复至同一服务区内的任意其它可用区中。另外，大家也可以根据所运行的应用要求选择其它选项，例如将应用以镜像形式同步至其它集群。

可扩展性与弹性

Auto Scaling[66] 服务允许大家以自动化方式根据事前定义的条件进行 Amazon EC2 容量伸缩。在 Auto Scaling 的帮助下，大家可以确保当前使用的 EC2 实例在需求提升时无缝完成规模向上扩展，从而保持既定性能; 同时在需求降低时进行规模收缩以降低使用成本。Auto Scaling 特别适合那些每周、每日甚至每小时发生明显使用量波动的应用程序。Auto Scaling 由 CloudWatch 负责实现，且大家只需支付 CloudWatch 正常使用成本即可享受其便利。

接口

Amazon EC2 可通过 API、SDK 或者控制台实现编程式接入。计算使用量、内存使用量、存储使用量、网络消耗量以及读取/写入流量等与实例相关的指标可通过控制台或者 CloudWatch API 操作免费获取。

您运行在 Amazon EC2 实例上的大数据分析软件可使用多种接口，具体取决于您所选择软件的实际情况。

反模式

Amazon EC2 拥有以下几种反模式：

· **托管服务**– 如果大家需要利用利用托管服务对基础设施层进行抽象，同时实现大数据分析管理，那么在 Amazon EC2 上以“自己动手”方式管理分析软件也许并不是个好主意。

· 缺少专业知识或者相关资源– 如果大家的企业不具备或者不希望扩展资源或者专业知识以安装及管理高可用性系统，那么大家应该考虑使用其它 AWS 服务选项，包括 Amazon EMR、DynamoDB、Amazon Kinesis Streams 或者 Amazon Redshift。

在 AWS 上解决大数据难题

在本份白皮书中，我们已经探讨了多款可在 AWS 之上实现大数据分析的工具选项。这些将成为大家设计自有大数据应用时的理想参考点。不过，在为自己的具体用例选择正确工具时，大家还应该考虑其它一些问题。总体而言，每种分析工作负载都拥有自己的特性与要求，这意味着其应当利用不同的工具加以打理：

· 您需要以怎样的速度获取分析结果——实时、数秒或者数小时？

· 这些分析结果能够为您的企业带来怎样的价值，您又打算为其分配多少预算？

· 数据集规模有多大？其增长速度又是怎样的？

· 数据结构是怎样的？

· 生产者与消费者各自拥有怎样的集成能力？

· 生产者与消费者能够接受怎样的延迟水平？

· 停机时间会带来何等水平的损失？解决方案需要如何提供可用性与持久性保障？

- 分析工作负载要求一致性还是弹性？

这些特性与要求能够帮助大家更为科学地选择正确工具。在某些情况下，大家可以直接将大数据分析工作负载映射至某项服务，且全部要求皆可得到满足。然而，在大多数情况下，大数据分析工作负载彼此各异甚至 相互冲突，这意味着同一数据集内亦存在着多种多样的特性与要求。

举例来说，部分分析结果集可能要求用户以实时方式与系统交互，但也有一些分析结果可以批量形式完成且需要每日运行。这些不同的要求往往出现在同一数据集内，且应进行解耦并利用多种工具分别处理。如果大家尝试利用同一工具集同时解决两项问题，那么最终结果要么是资源过度配置从而带来不必要的资金浪费，要么是解决方案的速度不足以为用户提供实时分析功能。通过将最适合的工具与独立分析问题一一对应，我们将能够保证以最为高效的方式使用计算与存储资源。

大数据并不意味着一定带来“大成本”。因此，在设计自己的应用程序时，大家必须要确保整体方案具备成本效益。如果无法实现，则考虑使用其它替代选项，直到找到正确答案。另一种常见误区在于，利用多套工具集解决同一大数据问题会带来过高成本或者产生高于单一大型工具的使用难度。以同一数据集中的两种不同要求为例，其中实时请求对 CPU 资源要求不高但却需要强大的 I/O 支持，而慢速处理请求则往往具备计算资源密集型属性。将二者解耦能够显著降低成本，同时确保大家更为轻松地完成管理工作——这是因为我们能够实现工具与任务间的对应，且不致进行过度配置。由于 AWS 采用按使用量计费的方式，且基础设施以服务形式进行交付，大家可在一小时内完成批量分析且只需要为此期间使用的计算资源付费。另外，这种方式也能够简化管理工作，意味着大家不需要利用单一系统满足全部需求。事实上，利用单一工具解决各类不同需求就像是把方形积木（实时请求）放进圆孔（大型数据仓库）当中。

AWS 平台提供多种不同工具以分析同一数据集，从而显著简化架构的解耦工作。其中还内置有各类AWS服务，意味着大家能够轻松将数据的某一子集在不同工具之间往来移动，且与其它任务并行运作。下面我们将设置几项实际案例，了解大数据分析方案带来的具体问题以及 AWS 提供的解决思路。

示例一：企业数据仓库

一家跨国服装厂商拥有 1000 多家零售门店，向百货及加盟商供货，同时亦提供在线商店。目前，这三种销售渠道各自拥有独立的技术体系。他们采取不同的管理制度及互不相关的财会部门。很明显，必须建立一套新系统将这些数据合并起来，从而确保 CEO 拥有更为准确的业务洞察能力。CEO 希望了解各销售渠道的宏观状况，并在必要时进行即时分析。在本示例中，这家公司要求达成的目标包括：

- 各销售渠道的现有趋势如何？
- 哪些地理区域在跨渠道销售方面表现更好？
- 他们的广告宣传与优惠活动的实际效果如何？
- 各服务产品线的现有趋势如何？
- 哪些外部因素会给销售额造成影响，例如失业率或者天气情况？
- 门店的哪些属性会影响销售情况，例如员工/管理人员的任期、所处地点、店内的商品摆放位置、促销活动、销售通告以及店内展示？

企业数据仓库方案是解决这些难题的好办法。数据仓库需要从三种销售渠道的不同系统处收集数据，同时将天气预报及经济情况等公共数据记录纳入进来。由于各数据源的结构可能有所区别，因此其需要利用提取、转换与加载（简称 ETL）流程将数据重新格式化为通用结构。在此之后，同时面向全部来源进行数据分析。要实际这一目标，我们需要使用以下数据流架构：

企业数据仓库

1. 此流程中的第一步在于从多个不同源处获取数据并交付 Amazon S3。之所以选择 Amazon S3，是因为其拥有出色的持久性、成本低廉且极具可扩展能力，可以极低成本实现多个数据源的并行写入。

2. Amazon EMR 用于清理并将数据源格式转换为特定的目标格式。Amazon EMR 能够直接与 Amazon S3 相集成，意味着我们能够在集群当中实现指向 Amazon S3 的各节点并发吞吐线程。一般来讲，数据仓库会在夜间从不同来源处获取新数据。由于午夜阶段不需要执行分析，因此这时系统的惟一任务就是完成其转换流程，这样 CEO 及其他业务用户就能够在早晨上班时加以使用了。这样的要求意味着，大家可以使用 Amazon EC2 Spot 市场 [67] 以进一步降低转换任务的实现成本。良好的 Spot（焦点）策略往往会立足于夜间时段以实现极低资源使用成本，同时随时段增加容量水平以适应实际需要。如果 Spot 无法实现预期效果，大家可以重新回归按需计费机制，确保自己能够满足实际性能需求。每一数据源可能在 Amazon EMR 上拥有着不同的转换流程，但立足于 AWS 的按使用量计费模式，大家完全可以专门为转换任务设置一套独立 Amazon EMR 集群，并为其分配充足的资源以完成一切数据转换任务，且无需同其它任务争夺任何资源。

3. 每项转换任务随后都会提取经过格式化及重新整理的数据，并将交付至 Amazon S3。这里之所以再次使用 Amazon S3，是因为 Amazon Redshift 能够借此立足各个节点进行多线程并发数据处理。Amazon S3 的这一定位还使其适合作为历史记录存储载体，并可在不同系统之间作为格式化数据源。Amazon S3 上的数据可由其它分析工具加以使用，从而随时满足各类新增处理需求。

4. Amazon Redshift 能够将数据加载、排序、分发及压缩为表形式，从而实现分析查询的高效与并发执行。Amazon Redshift 则专门面向数据仓库工作负载所构建，能够轻松通过添加更多节点进行规模扩展，从而应对随时间推移及业务发展而不断增长的数据总量。

5. 为了对分析结果进行可视化处理，大家可以选择使用 Amazon QuickSight 或者通过 OBDC/JDBC 接入 Amazon Redshift 的其它合作方可视化平台。通过这种方式，CEO 及其管理团队成员可以报告及图表形式查看数据分析结论，从而就企业资源做出更为明智的业务决策，最终提升运营收益并实现股东价值。

这套架构非常灵活且能够在业务发展、输入数量量增加、开设新型销售渠道或者发布涉及客户特定数据的移动应用时轻松实现规模扩张。无论何时，大家都能够将更多工具集成进来，并通过数次点击快速增加 Amazon Redshift 集群内的节点数量以扩大仓库容量。

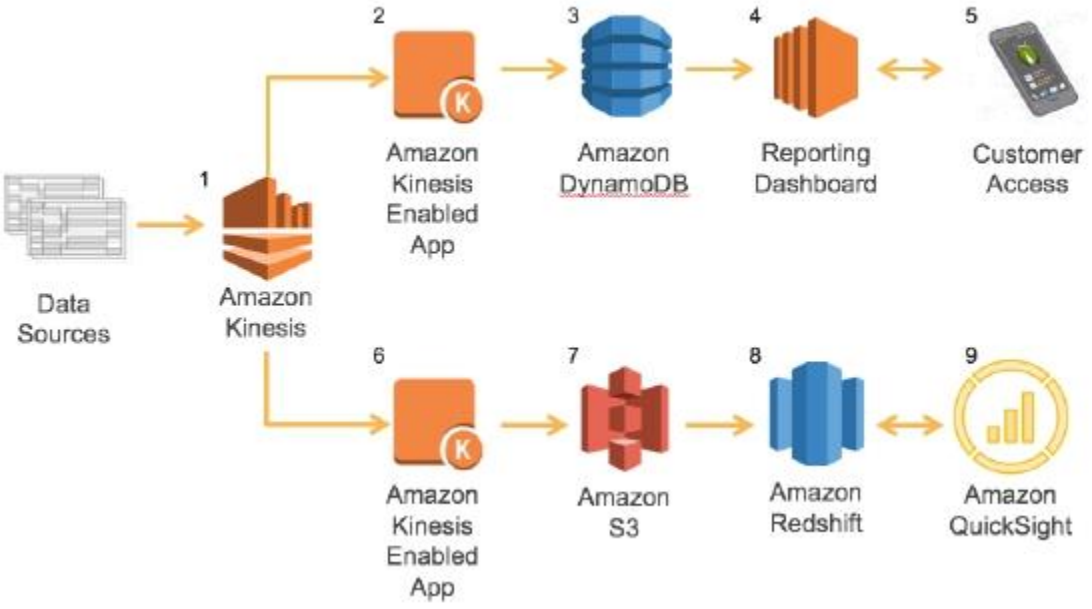
示例二：捕捉及分析传感器数据

某家国际空调制造商一直向多家商业及工业企业出售大型空调设备。除了销售空调机组之外，他们还需要提供附加服务以保持市场竞争优势——其中包括实时仪表板，供用户通过移动应用或者网络浏览器随时查看设备运行情况。每套机组都拥有独立的，需要处理及分析的传感器信息。制造商与客户可对这些数据加以使用。凭借着这一能力，该制造商得以对数据集及焦点趋势进行可视化处理。

目前，他们在数千套预售设备中纳入了这项功能。他们希望在未来几个月内将产品交付给客户，并希望这些设备能够及时上线。如果成功，他们将能够大幅扩张自己的客户基础，从而占得更为可观的市场份额。这套解决方案需要能够处理大规模数据，并以无中断方式随业务需求增长进行规模扩展。在这种情况下，我们该如何设计这套系统？首先，我们将其拆分为两项工作流程，二者皆源自同一数据集：

- 空调机组的当前信息与近实时要求，将有大量客户使用这部分信息。
- 与空调机组相关的全部历史信息，由该公司用于进行趋势预测及分析。

我们可以利用以下数据流架构解决上述大数据难题：



捕捉及分析传感器数据

1. 这套流程首先将各空调机组的数据流以一致性方式导入 Amazon Kinesis Streams。其提供一套弹性且持久的界面，可供机组与之通信从而在售出设备数量增加以无缝实现规模扩展。
2. 使用 Kinesis 客户端库或者 SDK 等 Amazon Kinesis Streams 提供的工具，立足于

Amazon EC2 读取一切写入至 Amazon Kinesis Streams 的数据、进行分析并检测其是否有必要被更新至实时仪表板。这部分信息面向系统运行变化、温度波动以及对应机组中出现的任何故障。

3. 该数据流程还需要以近实时方式进行，从而帮助客户及维护团队在机组出现问题时尽快得到警告。仪表板中的数据还包含有部分聚合趋势信息，但这里主要关注的是当前运行状态以及系统错误情况。因此，需要填充至仪表板中的数据规模并不大。另外，以下几部分受众可能也需要对此类数据进行访问：

- 客户通过移动设备或者浏览器检查其系统状况。
- 维护团队检查机组的运行状态。
- 报告平台中的数据与智能算法以及分析结论随后会将焦点趋势以提醒方式进行发送，例如空调机组已经运行很长时间、但建筑内温度仍未下降。

这里我们选择 DynamoDB 以存储这部分近实时数据，因为其同时具备高可用性与可扩展性; 通过这种方式，我们能够根据客户的需求对平台进行规模伸缩。

4. 报告仪表板属于一套定制化 Web 应用，其立足于这部分数据集并运行在 Amazon EC2 之上。其提供基于系统状态及趋势的内容，同时会在空调机组发生任何意外情况时向客户及维护人员发出警告。

5. 来自移动设备或者网络浏览器的客户访问能够获取系统的当前状态，同时亦可查看历史趋势的可视化结果。

这套数据流程（第 2 步到第 5 步）专门面向近实时信息报告构建，旨在供人员直接使用。其设计充分考虑到低延迟要求，且可根据需要快速完成规模伸缩。而下一数据流程（第 6 步到第 9 步）则不再对速度及延迟提出严格要求。这意味着架构师能够设计一套不同的解决方案堆栈，旨在以极低的每字节成本保存大量数据，同时选择价格更低廉的计算与存

储资源。

6. 要从 Amazon Kinesis 流内进行数据读取，我们需要使用一款独立的 Amazon Kinesis 应用，其可以运行在小型 EC2 实例之上并随需求缓慢进行规模扩展。尽管此应用将负责对同一数据集进行分析，但其最终目标在于将这部分数据进行长期保存并在数据仓库内加以托管。此数据集最终将容纳有来自该系统的全部数据，且允许多种分析工具执行处理而无需满足近实时要求。

7. 此数据由 Amazon Kinesis 应用进行格式转换，确保其适合进行长期存储，而后将其加载至数据仓库并存储在 Amazon S3 当中。Amazon S3 上的数据不仅可作为 Amazon Redshift 的并发输入源，同时亦可作为系统内全部数据的持久存储手段。事实上，Amazon S3 可以作为一种万能式存储载体，用于在必要时加载其它分析工具，或者将低访问量数据迁移至 Amazon Glacier 以进行长期、极低成本存储。

8. Amazon Redshift 在这里再次被用作大规模数据集的数据仓库。它能够在数据集体积扩张时轻松实现规模扩展，即向集群内添加更多节点。

9. 要对分析结果进行可视化处理，我们可以通过 OBDC/JDBC 将选定的合作伙伴可视化平台接入 Amazon Redshift。在这里，相关工具可对数据集进行报告、图形化以及即时分析，从而发现各类可能导致空调机组停止运转或者损坏的特定变量及趋势。

这套架构在起步阶段可选择小规模构建，并随需求增长而进行扩展。另外，通过将两种不同工作流进行解耦，二者能够彼此独立地发展且不致带来任何前期投入，这意味着制造商可以节约投资且快速判断其实际运行状况。在后续使用中，大家也能够轻松向其中引入 Amazon ML 等其它服务，从而预测特定空调机组的使用寿命并根据预测算法调配维护团队，从而为客户提供最佳服务与体验。如此级别的服务将在未来的销售工作中成为一种极具吸引力的差异化竞争优势。

示例三：社交媒体情感分析

一家大型玩具制造商一直在快速发展并扩张自身产品线。在发布每款新玩具时，该公司需要了解消费者对该产品的具体态度。另外，公司还需要确保消费者拥有良好的产品使用体验。随着玩具业务生态系统的不断发展，该公司希望确保自家产品仍然能够满足客户要求，并根据客户反馈规划未来的发展路线图。该公司希望通过社交媒体实现以下目标：

- 了解消费者者如何使用他们的产品。
- 确保客户满意度。
- 规划未来发展路线图。

虽然从社交媒体处捕捉数据相对简单，但真正的挑战在于以编程化方式建立情报体系。在数据获取完成后，设计公司公司希望能够以具备成本效益的可编程方式分析并归类这部分数据。为了实现这项目标，他们采用了以下架构：

社交媒体情感分析

1. 第一项任务是决定监听哪些社交媒体。在此之后，创建一款应用通过对应的 API 查询这些站点，且此应用运行在 Amazon EC2 之上。
2. 接下来，创建 Amazon Kinesis 数据流，这是因为我们需要使用多个数据源：Twitter、

Tumblr 等等。如此一来，每当有新的数据源添加进来，Amazon Kinesis Sterams 即可创建新的数据流，且继续沿用现有应用代码及架构。另外，在本示例中，我们创建新的 Amazon Kinesis 流以将原始数据复制到 Amazon S3 当中。

3. 为了实现归档、长期分析以及历史引用，原始数据应被存储在 Amazon S3 当中。另外，我们也可利用 Amazon ML 批量模型对 Amazon S3 中的数据进行预测性分析，同时追踪消费者的购买趋势。

4. 正如以上架构示意图所示，我们利用 Lambda 实现数据处理与规范化，同时从 Amazon ML 处请求预测结论。在 Amazon ML 预测结论成功返回后，Lambda 能够根据预测采取行动——例如将社交媒体内容交付至客户服务团队以进行进一步讨论。

5. Amazon ML 用于对输入数据进行预测分析。举例来说，大家可以构建一套机器学习模型以分析社交评论，从而了解客户对于某款产品的评论为积极还是消极。为了利用 Amazon ML 获得准确的预测结论，我们首先需要收集训练数据并确保机器学习模型运作正常。如果大家是第一次创建机器学习模型，则请参阅《教程：利用 Amazon ML 为市场营销策略提供预测响应》[68]。正如之前所提到，如果需要使用多个社交网络数据源，则请为其分别指定不同的机器学习模型以保证预测准确性。

6. 最后，利用 Lambda 将可操作数据发送至 Amazon SNS，同时通过短信或者邮件进一步调查信息来源。

7. 作为情感分析系统的组成部分，创建的 Amazon ML 模型需要定期更新以提供准确的分析结果。特定模型中的其它指标亦可通过控制台进行图形化显示，具体包括准确性、假阳性率、精度与回顾。欲了解更多信息，请参阅《第四步：审查机器学习型预测效果并设置截止点》。[69]

通过将 Amazon Kinesis Streams、Lambda、Amazon ML 以及 Amazon SES 配合使用，我们已经建立起了一套具备可扩展性且易于定制的社交监听平台。需要强调的是，这里并没有提到机器学习模型的具体创建方式。这项工作至少需要者一次，而且我们建议大家定期重构以保持模型更新。创建新模型的具体工作量不尽相同，且无疑属于提升模型分析准确率的重要手段。

总结

随着数据产生量及收集量的不断提升，数据分析要求我们利用可扩展、灵活且性能出色的工具及时提供洞察结论。然而，企业面对着不断变化的大数据生态系统，众多新兴工具快速出现但又立即消亡。因此，客户往往很难紧跟潮流并选择到正确的工具。

本份白皮书提供了多种解决方案以应对大数据分析需求。利用多种托管服务集合收集、处理及分析大数据，AWS 平台能够帮助大家显著简化大数据应用的构建、部署与规模伸缩工作，从而将主要精力集中在业务问题而非工具更新及管理身上。

AWS 提供多种解决方案以满足大数据分析要求。大部分大数据架构解决方案都会使用多种 AWS 工具以建立完整的解决方案，旨在满足企业对成本优化、性能以及弹性提出的严苛要求。这样一套灵活的大数据架构能够根据业务需要随时进行规模伸缩。

贡献者

以下人员为本份白皮书的编撰做出贡献：

- Erik Swensson, Amazon Web Services 解决方案架构经理
- Erick Dame, Amazon Web Services 解决方案架构师
- Shree Kenghe, Amazon Web Services 解决方案构架师

扩展阅读

以下资源可帮助大家在 AWS 之上着手运行大数据分析负载：

- 访问 aws.amazon.com/big-data[70]

查看完整的大数据服务组合以及其它 AWS 大数据合作伙伴、教程、文章等资源的链接。AWS Marketplace 亦提供多种大数据解决方案。如果您需要进一步帮助，请联系我们。

- 参阅 AWS 大数据博客。[71]

这份博客列举了大量实际示例及思路且定期更新，能够帮助大家对大数据进行收集、存储、清理、处理与可视化。

- 请从大数据测试驱动方案 [72] 中选择一种进行尝试。

利用 AWS 尝试利用丰富的产品生态系统解决各类大数据挑战。测试驱动方案由 AWS 合作伙伴网络（简称 APN）咨询与技术合作方开发，其中包含大量免费的教育、展示与评估资源。

- 选取一项 AWS 大数据培训课程。[73]

AWS 大数据课程能够为大家提供基于云的大数据解决方案与 Amazon EMR 使用指导。我们将讲解如何利用 Amazon EMR 借助 Pig 及 Hive 等 Hadoop 工具生态系统的力量实现数据处理。我们还将帮助大家了解如何创建大数据环境，使用 DynamoDB 与 Amazon Redshift，掌握 Amazon Kinesis Streams 的优势并通过各项最佳实践设计出安全且具备成本效益的大数据环境。

- 查看大数据客户案例研究。[74]

了解更多与其他客户相关的大数据平台成功经验。

文档修订

备注

[1] http://aws.amazon.com/about-aws/globalinfrastructure/

[2] http://aws.amazon.com/s3/

[3] http://aws.amazon.com/datapipeline/

[4] https://aws.amazon.com/iot/

[5] https://aws.amazon.com/importexport/

[6] http://aws.amazon.com/kinesis/firehose

[7] https://aws.amazon.com/directconnect/

[8] https://aws.amazon.com/mobile/

[9] http://aws.amazon.com/solutions/case-studies/big-data/

[10] https://aws.amazon.com/kinesis/streams

[11] http://docs.aws.amazon.com/kinesis/latest/APIReference/Welcome.html

[12] http://docs.aws.amazon.com/aws-sdk-php/v2/guide/service-kinesis.html

[13] http://aws.amazon.com/kinesis/pricing/

[14] http://aws.amazon.com/tools/

[15] http://docs.aws.amazon.com/kinesis/latest/dev/developing-producers-with-kpl.html

[16] http://docs.aws.amazon.com/kinesis/latest/dev/writing-with-agents.html

[17] https://github.com/awslabs/amazon-kinesis-client

[18] https://github.com/awslabs/kinesis-storm-spout

[19] https://aws.amazon.com/lambda/

[20] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html

[21] https://aws.amazon.com/amazon-linux-ami/

[22] http://docs.aws.amazon.com/lambda/latest/dg/nodejs-create-deployment-pkg.html

[23] http://docs.aws.amazon.com/lambda/latest/dg/lambda-python-how-to-create-deployment-package.html

[24] http://docs.aws.amazon.com/lambda/latest/dg/lambda-java-how-to-create-deployment-package.html

[25] http://aws.amazon.com/elasticmapreduce/

[26] https://media.amazonwebservices.com/AWS_Amazon_EMR_Best_Practices.pdf

[27] http://aws.amazon.com/elasticmapreduce/pricing/

[28] http://aws.amazon.com/ec2/instance-types/

[29] http://aws.amazon.com/elasticmapreduce/mapr/

[30] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-manage-resize.html

[31] http://docs.aws.amazon.com/ElasticMapReduce/latest/API/Welcome.html

[32] http://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/emr-hive.html

[33] http://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/emr-pig.html

[34] http://blogs.aws.amazon.com/bigdata/post/Tx15AY5C50K70RV/Installing-Apache-Spark-on-an-Amazon-EMR-Cluster

35 http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-hbase.html

36 http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-impala.html

37 http://aws.amazon.com/elasticmapreduce/hunk/

38 http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/UsingEMR_s3distcp.html

39 https://aws.amazon.com/machine-learning/

40 https://aws.amazon.com/machine-learning/pricing/

41 http://docs.aws.amazon.com/machine-learning/latest/dg/suggested-recipes.html

42 http://docs.aws.amazon.com/machine-learning/latest/APIReference/Welcome.html

43 https://aws.amazon.com/dynamodb

44 http://aws.amazon.com/free/

45 http://aws.amazon.com/dynamodb/pricing/

46 平均服务器端响应时间为个位数毫秒

47 http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html

48 DynamoDB 允许大家单纯利用一项 UpdateTable API 操作调用实现最高 100%吞吐容量配置变更。要将吞吐容量再次提升 100%，可重复调用 UpdateTable。

49 大家可以随时提升自己的配置吞吐容量; 不过，每天下调操作只能进行 2 次。

50 https://aws.amazon.com/redshift/

51 http://aws.amazon.com/s3/pricing/

52 http://aws.amazon.com/redshift/pricing/

53 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html

54 http://docs.aws.amazon.com/redshift/latest/APIReference/Welcome.html

55 http://docs.aws.amazon.com/redshift/latest/mgmt/working-with-clusters.html#rs-about-clusters-and-nodes

56 http://docs.aws.amazon.com/redshift/latest/dg/c_redshift-and-postgres-sql.html

57 http://aws.amazon.com/redshift/partners/

58 https://aws.amazon.com/elasticsearch-service/

59 https://aws.amazon.com/ec2/pricing/

60 https://aws.amazon.com/ebs/details/

61 https://aws.amazon.com/elasticsearch-service/pricing/

62 https://aws.amazon.com/elasticsearch-service/faqs/

63 https://aws.amazon.com/quicksight

64 https://aws.amazon.com/ec2/

65 https://aws.amazon.com/marketplace

66 http://aws.amazon.com/autoscaling/

67 http://aws.amazon.com/ec2/spot/

68 http://docs.aws.amazon.com/machine-learning/latest/dg/tutorial.html

69 http://docs.aws.amazon.com/machine-learning/latest/dg/step-4-review-the-ml-model-predictive-performance-and-set-a-cut-off.html

70 http://aws.amazon.com/big-data

71 http://blogs.aws.amazon.com/bigdata/

72 https://aws.amazon.com/testdrive/bigdata/

73 http://aws.amazon.com/training/course-descriptions/bigdata/

74 http://aws.amazon.com/solutions/case-studies/big-data/