第3章

云计算原理与技术

3.1 云计算概述

本章首先介绍云计算起源、定义和分类等基本概念，接着重点阐述了云计算的关键技术，然后分别讨论了谷歌云、亚马逊云和阿里云的技术原理，并给出了一个基于亚马逊云的大数据分析案例，让读者更深刻地理解如何利用公有云来实现大数据分析应用。

3.1.1 云计算起源

随着信息和网络通信技术的快速发展，计算模式从最初的把任务交给大型处理机集中计算，逐渐发展为更有效率的基于网络的分布式任务处理模式，自 20 世纪 80 年代起，互联网得到快速发展，基于互联网的相关服务的增加，以及使用和交付模式的变化，云计算模式应运而生。如图 3-1 所示，云计算是从网络即计算机、网格计算池发展而来的概念。

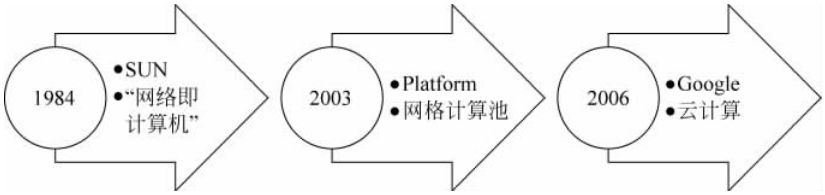

图 3-1　云计算的起源

早期的单处理机模式计算能力有限，网络请求通常不能被及时响应，效率低下。随着网络技术不断发展，用户通过配置具有高负载通信能力的服务器集群提供急速增长的互联网服务，但在遇到负载低峰的时候，通常会有资源浪费和闲置，导致用户的运行维护成本提高。而云计算把网络上的服务资源虚拟化并提供给其他用户使用，整个服务资源的调度、管理、

维护等工作都由云端负责，用户不必关心“云”内部的实现就可以直接使用其提供的各种服务，因此，如图 3-2 所示，云计算实质上是给用户提供像传统的电力、水、煤气一样的按需计算服务，它是一种新的有效的计算使用范式。

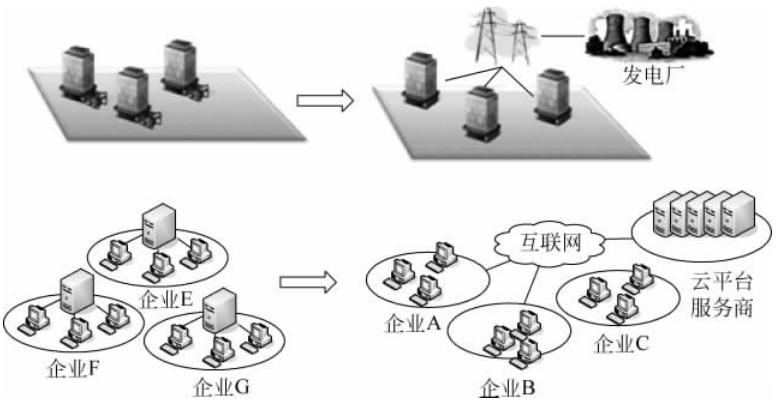

图 3-2　云计算的目标

云计算是分布式计算、效用计算、虚拟化技术、Web 服务、网格计算等技术的融合和发展，其目标是用户通过网络能够在任何时间、任何地点最大限度地使用虚拟资源池，处理大规模计算问题。目前，在学术界和工业界共同推动之下，云计算及其应用呈现迅速增长的趋势，各大云计算厂商如 Amazon、IBM、Microsoft、Sun 等公司都推出了自己研发的云计算服务平台。而学术界也源于云计算的现实背景纷纷对模型、应用、成本、仿真、性能优化、测试等诸多问题进行了深入研究，提出了各自的理论方法和技术成果，极大地推动了云计算继续向前发展。

3.1.2　云计算的概念与定义

2006 年，Google 高级工程师克里斯托夫·比希利亚第一次向 Google 董事长兼 CEO 施密特提出“云计算”的想法，在施密特的支持下，Google 推出了“Google 101 计划”（该计划目的是让高校的学生参与到云的开发），并正式提出“云”的概念。由此，拉开了一个时代计算技术以及商业模式的变革。

如图 3-3 所示，对一般用户而言：云计算是指通过网络以按需、易扩展的方式获得所需的服务。即随时随地只要能上网就能使用各种各样的服务，如同钱庄、银行、发电厂等。这种服务可以是 IT 和软件、互联网相关的，也可以是任意其他的服务。

图 3-3　一般用户的云计算概念

如图 3-4 所示，对专业人员而言：云计算是分布式处理、并行处理和网格计算的发展，或者说是这些计算机科学概念的商业实现。是指基于互联网的超级计算模式——即把原本存储于个人计算机、移动设备等个人设备上的大量信息集中在一起，在强大的服务器端协同工作。它是一种新兴的共享计算资源的方法，能够将巨大的系统连接在一起，以提供各种计算服务。

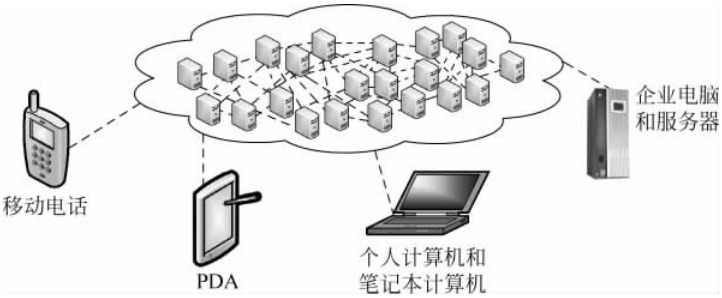

图 3-4　专业人员的云计算概念

目前，比较权威的云计算定义是美国国家标准技术研究院 NIST 提出的，包括以下 4 点：

（1）云计算是一种利用互联网实现随时随地、按需、便捷地访问共享资源池（如计算设施、存储设备、应用程序等）的计算模式。

（2）云计算模式具有 5 个基本特征：按需自助服务、广泛的网络访问、共享的资源池、快速弹性能力、可度量的服务。

（3）3 种服务模式：软件即服务（SaaS）、平台即服务（PaaS）、基础设施即服务（IaaS）。

（4）4 种部署方式：私有云、社区云、公有云、混合云。

在我们看来，理解云计算概念，应该区分云计算的两种不同技术模式：

（1）以大分小（Amazon 模式），特征有：硬件虚拟化技术，统一的资源池管理动态分配资源，提高资源利用率，降低硬件投资成本，适合于公共云平台提供商和面向中小型租赁用户。

（2）以小聚大（Google 模式），特征有：分布式存储（适合海量数据存储），并行计算（适合海量数据处理），线性的水平扩展能力，适合海量数据存储、检索、统计、挖掘，在互联网企业应用成熟。

3.1.3　云计算与分布式计算

1. 云计算与分布式计算

按照狭义的概念来讲，分布式计算是将待解决问题分成多个小问题，再分配给许多计算系统处理，最后将处理结果加以综合。分布式计算的特点是把计算任务分派给网络中的多台独立的机器并行计算，与传统的单机计算形式相比。分布式计算的优点主要有：

（1）稀有资源可以共享。

（2）通过分布式计算可以在多台计算机上平衡计算负载。

（3）可以把程序放在最适合运行它的计算机上。

分布式计算已经在很多领域加以应用，目前比较流行的分布式项目主要有：

① SETI@Home：寻找外星文明。

② RC-72：密码分析破解，研究和寻找最为安全的密码系统。

③ Folding@home：研究蛋白质折叠、聚合问题。

④ United Devices：寻找对抗癌症的有效的药物。

⑤ GIMPS：寻找最大的梅森素数(解决较为复杂的数学问题)。

云计算是分布式计算的一种新形式，但云计算提供的服务包含了更复杂的商业模式，云计算包含的分布式计算特征主要有：

(1) 通过资源调度和组合满足用户的资源请求。

(2) 对外提供统一的单一的接口。

2. 云计算与网格计算

网格计算有了十几年的历史，提出时主要用于科学计算。网格计算的目的是整合大量异构计算机的闲置资源(如计算资源和磁盘存储等)，组成虚拟组织，以解决大规模计算问题。

云计算是从网格计算演化来的，发展并包含了网格计算的内容。网格计算与云计算主要区别有：第一，网格主要是通过聚合式分布的资源，通过虚拟组织提供高层次的服务，而云计算资源相对集中，通常以数据中心的形式提供对底层资源的共享使用，而不强调虚拟组织的观念；第二，网格聚合资源的主要目的是支持挑战性的应用，主要面向教育和科学计算，而云计算一开始就是用来支持广泛的企业计算、Web 应用等；第三，网格用中间件屏蔽异构性，而云计算承认异构，用提供服务的机制来解决异构性的问题。

3. 云计算与对等计算

对等计算(P2P)是一种高效的计算模式。如图 3-5 所示，在对等计算系统中，每个节点都拥有对等的功能与责任，既可以充当服务器向其他节点提供数据或服务，又可以作为客户机享用其他节点提供的数据或服务；节点之间的交互可以是直接对等的，任何节点可以随时自由地加入或离开系统。云计算对超大规模、多类型资源的统一管理是困难的，而对等计算具有在鲁棒性、可扩展性、成本、搜索等方面的优点，在云计算体系结构和平台设计方面多有应用。

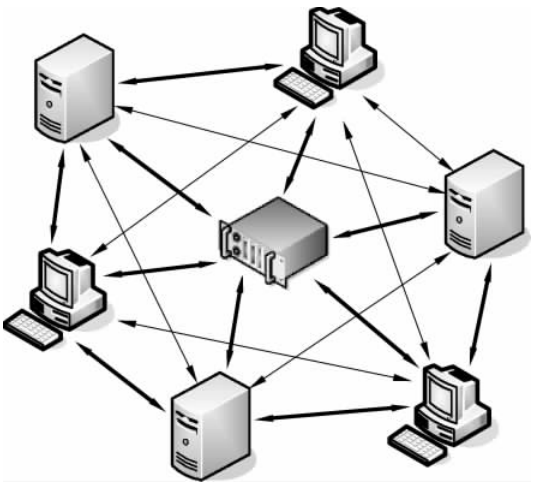

图 3-5　对等网络

4. 云计算与并行计算

早期的并行计算就是在并行计算机上所做的计算，它与常说的高性能计算、超级计算是同义词，因为任何高性能计算和超级计算都离不开并行计算。目前比较正式的定义是，并行计算是相对于串行计算来说的，是指同时使用多种计算资源解决计算问题的过程。并行计算可以划分成时间并行和空间并行。时间并行即流水线技术，空间并行使用多个处理器执行并发计算，并行计算科学中主要研究的是空间上的并行问题。从程序和算法设计人员的角度看，并行计算又可分为数据并行和任务并行。一般来说，因为数据并行主要是将一个大任务化解成相同的各个子任务，比任务并行要容易处理。

云计算是在并行计算之后产生的概念，是由并行计算和分布式计算发展而来，两者在很多方面有着共性。但并行计算不等于云计算，云计算也不等同并行计算。两者区别如下：

（1）云计算萌芽于并行计算；

（2）并行计算、高性能计算、网格计算等只用于特定的科学领域、专业的用户；

（3）并行计算追求的高性能；

（4）云计算对于单节点的计算能力要求低，主要目的是资源共享。

随着云计算的出现，云计算也可以作为并行计算的一种形式，即通过云计算实现并行计算。反之，云计算也包含了用户资源请求的并行处理等并行计算特征。

3.1.4　云计算分类

云计算按照提供服务的类型可以分为：基础设施即服务（IaaS），平台即服务（PaaS）和软件即服务（SaaS）。如图 3-6 所示，3 种类型云服务对应不同的抽象层次。

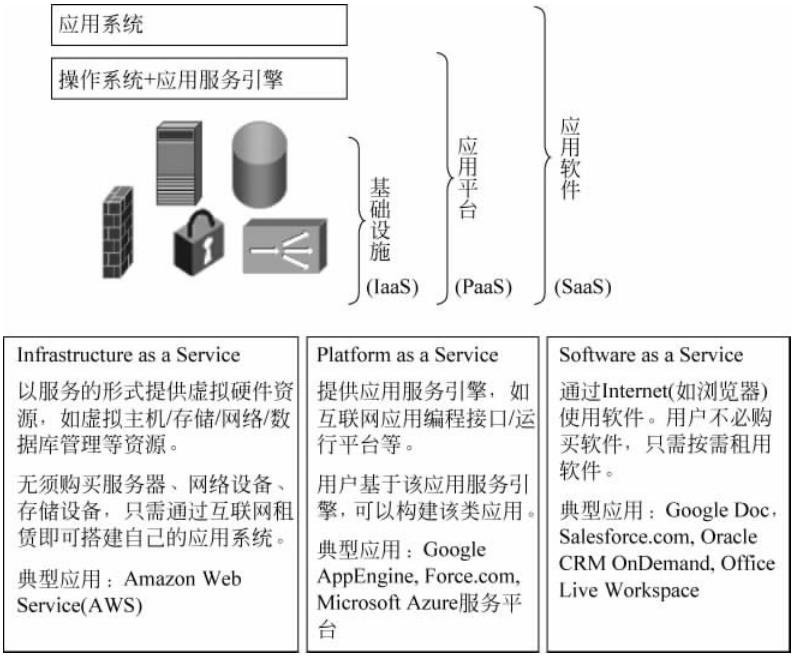

图 3-6　云计算分类

1. IaaS：基础设施即服务

IaaS(Infrastructure as a Service)：基础设施即服务。IaaS 是云计算的基础，为上层云计算服务提供必要的硬件资源，同时在虚拟化技术的支持下，IaaS 层可以实现硬件资源的按需配置，创建虚拟的计算、存储中心，使得其能够把计算单元、存储器、I/O 设备、带宽等计算机基础设施，集中起来成为一个虚拟的资源池对外提供服务(如硬件服务器租用)。如图 3-7 所示，虚拟化技术是 IaaS 的关键技术。

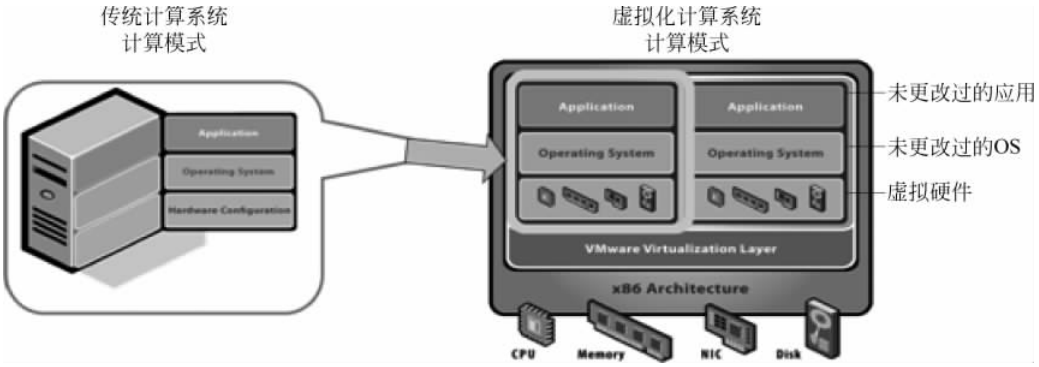

图 3-7　虚拟化技术

许多大型的电子商务企业，积累了大规模 IT 系统设计和维护的技术与经验，同时面临业务淡季时 IT 设备的闲置问题，于是可以将设备、技术和经验作为一种打包产品去为其他企业提供服务，利用闲置的 IT 设备创造价值。Amazon 是第一家将基础设施作为服务出售的公司，如图 3-8 所示，Amazon 的云计算平台弹性计算云 EC2(elastic compute cloud)可以为用户或开发人员提供一个虚拟的集群环境，既满足了小规模软件开发人员对集群系统的需求，减小了维护的负担，又有效解决了设备闲置问题。

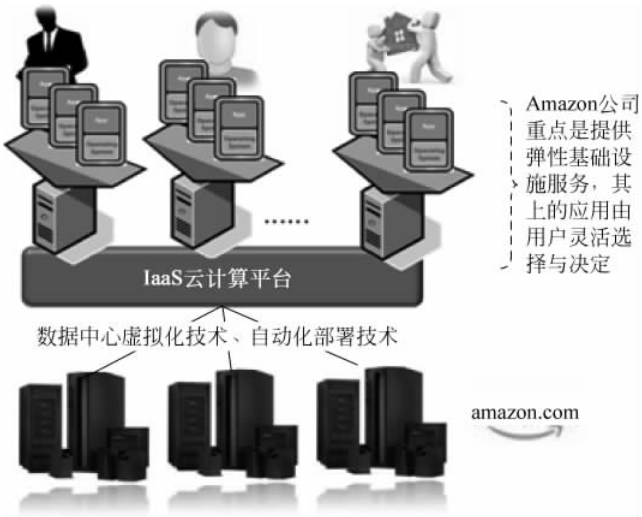

图 3-8　IaaS 云计算平台

2. PaaS：平台即服务

PaaS(Platform as a Service)：平台即服务。一些大型电子商务企业，为支持搜索引擎和邮件服务等需要海量数据处理能力的应用，开发了分布式并行技术的平台，在技术和经验有一定积累后，逐步将平台能力作为软件开发和交付的环境进行开放。如图 3-9 所示，Google 以自己的文件系统(GFS)为基础打造出的开放式分布式计算平台 Google App Engine，App Engine 是基于 Google 数据中心的开发、托管 Web 应用程序的平台。通过该平台，程序开发者可以构建规模可扩展的 Web 应用程序，而不用考虑硬件基础设施的管理。App Engine 由 GFS 管理数据、MapReduce 处理数据，并用 Sawzall 为编程语言提供接口，为用户提供可靠并且有效的平台服务。

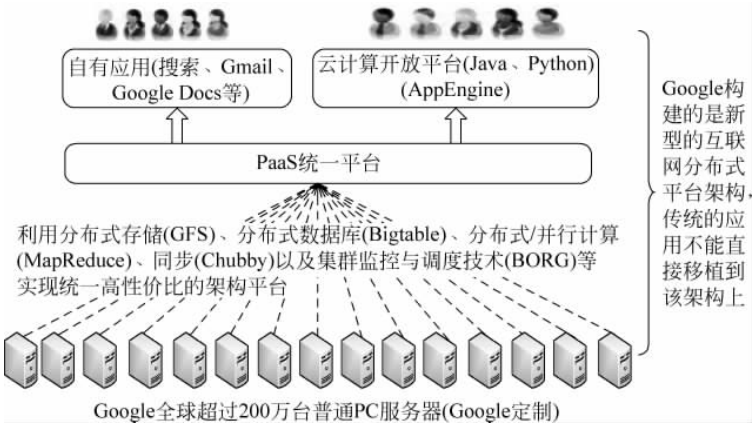

图 3-9　Google 分布式计算平台

PaaS 既要为 SaaS 层提供可靠的分布式编程框架，又要为 IaaS 层提供资源调度，数据管理，屏蔽底层系统的复杂性等，同时 PaaS 又将自己的软件研发平台作为一种服务开放给用户。例如，软件的个性化定制开发。PaaS 层需要具备存储与处理海量数据的能力，用于支撑 SaaS 层提供的各种应用。因此，PaaS 的关键技术包括并行编程模型、海量数据库、资源调度与监控、超大型分布式文件系统等分布式并行计算平台技术(如图 3-10 所示)。基于这些关键技术，通过将众多性能一般的服务器的计算能力和存储能力充分发挥和聚合起来，形成一个高效的软件应用开发和运行平台，能够为特定的应用提供海量数据处理。

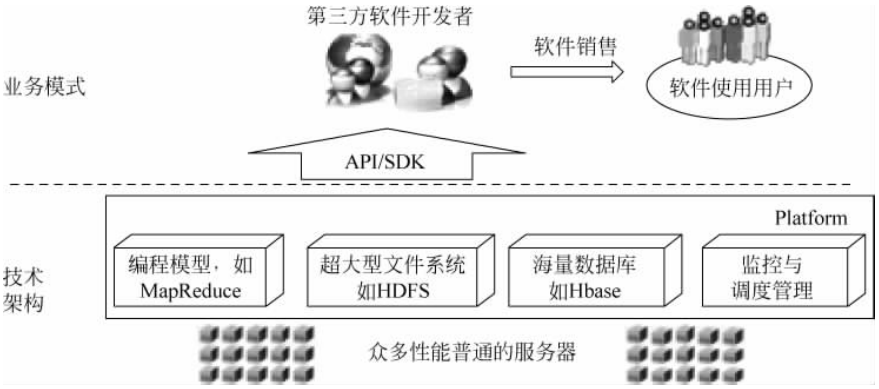

图 3-10　PaaS 的关键技术

3. SaaS：软件即服务

SaaS(Software as a Service)：软件即服务。云计算要求硬件资源和软件资源能够更好地被共享，具有良好的伸缩性，任何一个用户都能够按照自己的需求进行客户化配置而不影响其他用户的使用。多租户技术就是云计算环境中能够满足上述需求的关键技术，而软件资源共享则是 SaaS 的服务目的，用户可以使用按需定制的软件服务，通过浏览器访问所需的服务，如文字处理、照片管理等，而且不需要安装此类软件。

SaaS 层部署在 PaaS 和 IaaS 平台之上，同时用户可以在 PaaS 平台上开发并部署 SaaS 服务，SaaS 面向的是云计算终端用户，提供基于互联网的软件应用服务。随着网络技术的成熟与标准化，SaaS 应用近年来发展迅速。典型的 SaaS 应用包括 Google Apps、Salesforce 等。

Google Apps 包括 Google Docs、Gmail 等大量 SaaS 应用，Google Apps 将常用的一些传统的桌面应用程序(如文字处理软件、电子邮件服务、照片管理、通信录、日程表等)迁移到互联网，并托管这些应用程序。用户通过网络浏览器，便可随时随地使用 Google Apps 提供的应用服务，而不需要下载、安装或维护任何硬件或软件。

3.2 云计算关键技术

3.2.1 体系结构

云计算可以按需提供弹性的服务，如图 3-11 所示，它的体系架构可以大致分为三个层次：核心服务、服务管理和用户访问接口[13]。核心服务层将硬件基础设施、软件运行环境、应用程序抽象成服务，这些服务具有可靠性强、可用性高、规模可伸缩等特点，满足多样化应用需求。服务管理层为核心服务提供支持，进一步确保核心服务的可靠性、可用性与安全性。用户访问接口层实现端到云的访问。

1. 核心服务层

云计算核心服务通常可以分为 3 个子层：基础设施即服务层(Infrastructure as a Service，IaaS)、平台即服务层(Platform as a Service，PaaS)、软件即服务层(Software as a Service，SaaS)。

IaaS 提供硬件基础设施部署服务，为用户按需提供实体或虚拟的计算、存储和网络等资源。在使用 IaaS 层服务的过程中，用户需要向 IaaS 层服务提供商提供基础设施的配置信息，运行于基础设施的程序代码以及相关的用户数据。为了优化硬件资源的分配，IaaS 层引入了虚拟化技术。借助于 Xen、KVM、VMware 等虚拟化工具，可以提供可靠性高、可定制性强、规模可扩展的 IaaS 层服务。

PaaS 是云计算应用程序运行环境，提供应用程序部署与管理服务。通过 PaaS 层的软件工具和开发语言，应用程序开发者只需上传程序代码和数据即可使用服务，而不必关注底层的网络、存储、操作系统的管理问题。由于目前互联网应用平台(如 Facebook、Google、淘宝等)的数据量日趋庞大，PaaS 层应当充分考虑对海量数据的存储与处理能力，并利用有效的资源管理与调度策略提高处理效率。

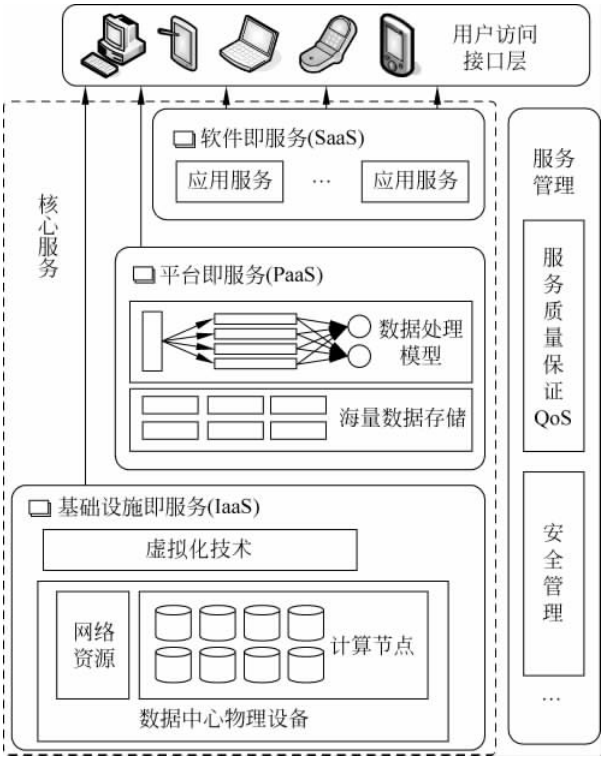

图 3-11　云计算体系结构

SaaS 是基于云计算基础平台所开发的应用程序。企业可以通过租用 SaaS 层服务解决企业信息化问题，如企业通过 GMail 建立属于该企业的电子邮件服务。该服务托管于 Google 的数据中心，企业不必考虑服务器的管理、维护问题。对于普通用户，SaaS 层服务将桌面应用程序迁移到互联网，可实现应用程序的泛在访问。

2. 服务管理层

服务管理层对核心服务层的可用性、可靠性和安全性提供保障。服务管理包括服务质量(Quality of Service，QoS)保证和安全管理等。此外，数据的安全性一直是用户较为关心的问题。云计算数据中心采用的资源集中式管理方式使得云计算平台存在单点失效问题。保存在数据中心的关键数据会因为突发事件(如地震、断电)、病毒入侵、黑客攻击而丢失或泄露。根据云计算服务特点，研究云计算环境下的安全与隐私保护技术(如数据隔离、隐私保护、访问控制等)是保证云计算得以广泛应用的关键。除了 QoS 保证、安全管理外，服务管理层还包括计费管理、资源监控等管理内容，这些管理措施对云计算的稳定运行同样起到重要作用。

3. 用户访问接口层

用户访问接口实现了云计算服务的泛在访问，通常包括命令行、Web 服务、Web 门户等形式。命令行和 Web 服务的访问模式既可为终端设备提供应用程序开发接口，又便于多种服务的组合。Web 门户是访问接口的另一种模式。通过 Web 门户，云计算将用户的桌面应用迁移到互联网，从而使用户随时随地通过浏览器就可以访问数据和程序，提高工作

效率。

3.2.2 数据存储

云计算环境下的数据存储，通常称之为海量数据存储，或大数据存储。大数据存储与传统的数据库服务在本质上有着较大的区别，传统的关系数据库中强调事务的 ACID 特性，即原子性(atomicity)、一致性(consistency)、隔离性(isolation)和持久性(durability)，对于数据的一致性的严格要求使其在很多分布式场景中无法应用。在这种情况下，出现了基于 BASE 特性的新型数据库，即只要求满足 basically available(基本可用)、soft state(柔性状态)和 eventually consistent(最终一致性)。从分布式领域著名的 CAP 理论角度看，ACID 追求一致性，而 BASE 更加关注可用性，正是在事务处理过程中对一致性的严格要求，使得关系数据库的可扩展性极其有限。

面对这些挑战，以 Google 为代表的一批技术公司纷纷推出自己的解决方案。BigTable 是 Google 早期开发的数据库系统，它是一个多维稀疏排序表，由行和列组成，每个存储单元都有一个时间戳，形成三维结构。不同的时间对同一个数据单元的多个操作形成的数据的多个版本由时间戳区分。除了 BigTable 外，Amazon 公司的 Dynamo 和 Yahoo 公司的 PNUTS 也都是非常具有代表性的系统。Dynamo 综合使用了键值存储、改进的分布式哈希表(DHT)、向量时钟(vector clock)等技术实现了一个完全的分布式、去中心化的高可用系统。PNUTS 是一个分布式数据库，在设计上使用弱一致性达到高可用性的目标，主要的服务对象是相对较小的记录，例如在线的大量单个记录或者小范围记录集合的读和写访问，不适合存储大文件、流媒体等。BigTable、Dynamo、PNUTS 等的成功促使人们开始对关系数据库进行反思，由此产生了一批未采用关系模型数据库，这些方案现在被统一称为 NoSQL (Not only SQL)。NoSQL 并没有一个准确的定义，但一般认为 NoSQL 数据库应当具有以下的特征：模式自由(schema-free)、支持简易备份(easy replication support)、简单的应用程序接口(simple API)、最终一致性(或者说支持 BASE 特性，不支持 ACID)、支持海量数据(huge amount of data)。

NoSQL 仅仅是一个概念，NoSQL 数据库根据数据的存储模型和特点分为很多种类。表 3-1 是 NoSQL 数据库的一个基本分类，当然，表中的 NoSQL 数据库类型的划分并不是绝对的，只是从存储模型上进行的大体划分。而且，它们之间没有绝对的分界，也有交差的情况，例如 Tokyo Cabinet/Tyrant 的 Table 类型存储，就可以理解为是文档型存储，Berkeley DB XML 数据库是基于 Berkeley DB 之上开发的。

表 3-1 NoSQL 数据库分类

类　别	产　品	特　性
列存储	HBase Cassandra Hypertable	顾名思义，是按列存储数据的。最大的特点是方便存储结构化和半结构化数据，方便做数据压缩，对某一列或者某几列的查询有非常大的 IO 优势
文档存储	MongoDB CouchDB	文档存储一般用类似 json 的格式存储，存储的内容是文档型的。这样也就有机会对某些字段建立索引，实现关系数据库的某些功能

续表

类　别	产　品	特　性
key-value 存储	Tokyo Cabinet/Tyrant Berkeley DB MemcacheDB Redis	可以通过 key 快速查询到其 value。一般来说，存储不管 value 的格式，照单全收。（Redis 包含了其他功能）
图存储	Neo4J FlockDB	图形关系的最佳存储。使用传统关系数据库解决的话性能低下，而且设计使用不方便
对象存储	db4o Versant	通过类似面向对象语言的语法操作数据库，通过对象的方式存取数据
XML 数据库	Berkeley DB XML BaseX	高效地存储 XML 数据，并支持 XML 的内部查询语法，比如 XQuery，Xpath

1. 数据中心

实现云计算环境下数据存储的基础是由数以万计的廉价存储设备所构成的庞大的存储中心，这些异构的存储设备通过各自的分布式文件系统将分散的、低可靠的资源聚合为一个具有高可靠性、高可扩展性的整体，在此基础上构建面向用户的云存储服务。如图 3-12 所示，数据中心是实现云计算海量数据存储的基础，主要包括各种存储设备，以及对各种异构的存储设备进行管理的分布式文件系统。

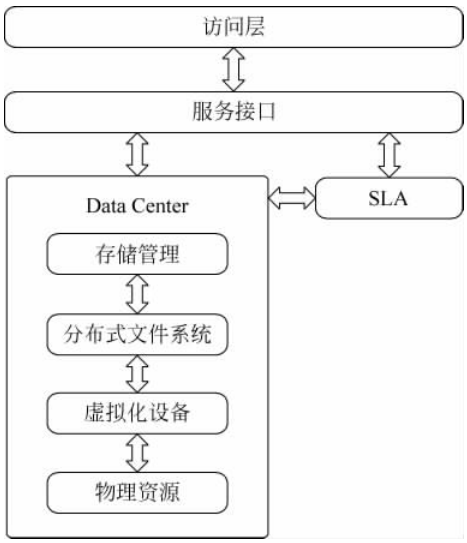

图 3-12　云计算平台存储结构

2. 分布式文件系统

分布式文件系统（Distributed File System，DFS）是云存储的核心，一般作为云计算的数据存储系统，对 DFS 的设计既要考虑系统的 IO 性能，又要保证文件系统的可靠性与可用性。文件系统是支撑上层应用的基础，Google 自行研发的 GFS（Google File System）是一种构建在大量服务器之上的可扩展的分布式文件系统，采用主从架构，通过数据分块，追加更新等方式实现海量数据的高效存储。

Google 以论文的形式公开其在云计算领域研发的各种技术，使得以 GFS 和 Bigtable 为代表的一系列大数据处理技术被广泛了解并得到应用，并催生出以 Hadoop 为代表的一系列云计算开源工具。而 GFS 类的文件系统主要针对较大的文件设计的，而在一些场景系统需要频繁地读写海量小文件，此时 GFS 类文件系统因为频繁读取元数据等原因，显得效率很低，Facebook 推出的专门针对海量小文件的文件系统 Haystack，通过多个逻辑文件共享同一个物理文件，增加缓存层，部分元数据加载到内存等方式有效解决了 Facebook 海量图片存储问题。淘宝推出的类似的文件系统 TFS(Tao File System)，通过将小文件合并成大文件，文件名隐含部分元数据等方式实现了海量小文件的高效存储。此外被广泛使用的还有 Lustre、FastDFS、HDFS 和 NFS 等，分别适用于不同应用环境下的分布式文件系统。

3.2.3 计算模型

云计算的计算模型是一种可编程的并行计算框架，需要高扩展性和容错性支持。PaaS 平台不仅要实现海量数据的存储，而且要提供面向海量数据的分析处理功能。由于 PaaS 平台部署于大规模硬件资源上，所以海量数据的分析处理需要抽象处理过程，并要求其编程模型支持规模扩展，屏蔽底层细节并且简单有效。目前比较成熟的技术有 MapReduce，Dryad 等。

MapReduce 是 Google 提出的并行程序编程模型，运行于 GFS 之上。MapReduce 的设计思想在于将问题分而治之，首先将用户的原始数据源进行分块，然后分别交给不同的 Map 任务去处理。Map 任务从输入中解析出键/值对(key/value)集合，然后对这些集合执行用户自行定义的 Map 函数得到中间结果，并将该结果写入本地硬盘。Reduce 任务从硬盘上读取数据之后会根据 key 值进行排序，将具有相同 key 值的数据组织在一起。最后用户自定义的 Reduce 函数会作用于这些排好序的结果并输出最终结果。图 3-13 给出 MapReduce 任务调度过程。

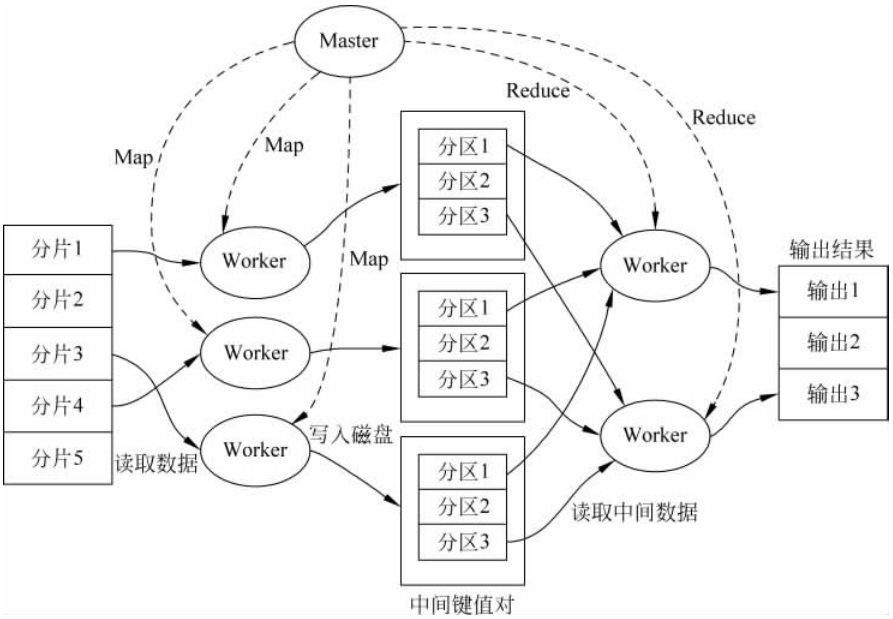

图 3-13 MapReduce 的任务调度

第一步：用户程序首先调用的 MapReduce 库将输入文件分成 M 个数据片段，然后用户程序在集群中创建大量的程序副本。

第二步：程序副本 master 将 Map 任务和 Reduce 任务分配给 worker 程序。

第三步：被分配 Map 任务的 worker 程序读取相关的输入数据片段。

第四步：Map 任务的执行结果写入到本地磁盘上。

第五步：Reduce worker 程序使用 RPC 从 Map worker 所在主机磁盘上读取这些缓存数据。

第六步：Reduce worker 程序遍历排序后的中间数据，Reduce 函数的输出被追加到所属分区的输出文件。

第七步：当所有的 Map 和 Reduce 任务都完成之后，master 唤醒用户程序。在这个时候，在用户程序里的对 MapReduce 调用才返回。

与 Google 的 MapReduce 相似，2010 年 12 月 21 日微软公司推出了 Dryad 的公测版，Dryad 也通过分布式计算机网络计算海量数据，成为谷歌 MapReduce 分布式数据计算平台的竞争对手。由于许多问题难以抽象成 MapReduce 模型，Dryad 采用基于有向无环图 DAG 的并行模型，在 Dryad 中，每一个数据处理作业都由 DAG 表示，图中的每一个节点表示需要执行的子任务，节点之间的边表示 2 个子任务之间的通信，Dryad 任务结构如图 3-14 所示。Dryad 可以直观地表示出作业内的数据流。基于 DAG 优化技术，Dryad 可以更加简单高效地处理复杂流程。同 MapReduce 相似，Dryad 为程序开发者屏蔽了底层的复杂性，并可在计算节点规模扩展时提高处理性能。

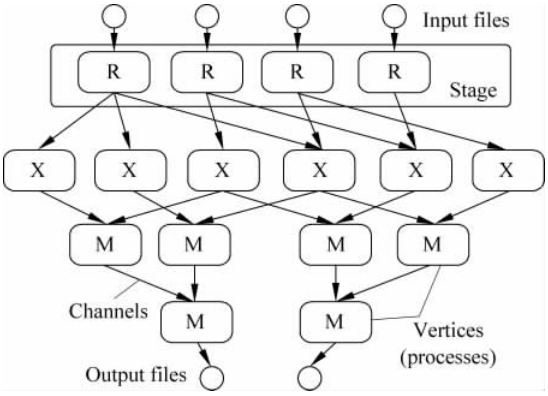

图 3-14　Dryad 任务结构

3.2.4　资源调度

海量数据处理平台的大规模性给资源管理与调度带来挑战。云计算平台的资源调度包括异构资源管理、资源合理调度与分配等。

云计算平台包含大量文件副本，对这些副本的有效管理是 PaaS 层保证数据可靠性的基础，因此一个有效的副本策略不但可以降低数据丢失的风险，还能优化作业完成时间。

PaaS 层的海量数据处理以数据密集型作业为主，其执行能力受到 IO 带宽的影响。网

络带宽是计算集群(计算集群既包括数据中心中物理计算节点集群,也包括虚拟机构建的集群)中的急缺的资源:

1) 云计算数据中心考虑成本因素,很少采用高带宽的网络设备。

2) IaaS 层部署的虚拟机集群共享有限的网络带宽。

3) 海量数据的读写操作占用了大量带宽资源。因此 PaaS 层海量数据处理平台的任务调度需要考虑网络带宽因素。

目前对于云计算资源管理方面进行的研究主要在降低数据中心能耗,提高系统资源利用率等方面,例如通过动态调整服务器 CPU 的电压或频率来节省电能,关闭不需要的服务器资源实现节能等;也有对虚拟机放置策略的算法,实现负载低峰或高峰时,通过有效放置虚拟机达到系统资源的有效利用。研究有效的资源管理与调度技术可以提高 MapReduce 等 PaaS 层海量数据处理平台的性能。

3.2.5 虚拟化

云计算的发展离不开虚拟化技术。虚拟化技术可以使物理上的单台服务器,被虚拟成逻辑上的多台服务器环境,可以修改单台虚拟机的分配 CPU,内存空间,硬盘等,每台虚拟机逻辑上可以被单独作为服务器使用。通过这种分割行为,将闲置或处于低峰的服务器紧凑地使用起来,数据中心为云计算提供了大规模资源,通过虚拟化技术实现基础设施服务的按需分配,虚拟化是 IaaS 层的重要组成部分,也是云计算的最重要特点。虚拟化技术可以提供以下特点。

1) 资源共享。通过虚拟机封装用户各自的运行环境,有效实现多用户分享数据中心资源。

2) 资源定制。用户利用虚拟化技术,配置私有的服务器,指定所需的 CPU 数目、内存容量、磁盘空间,实现资源的按需分配。

3) 细粒度资源管理。将物理服务器拆分成若干虚拟机,可以提高服务器的资源利用率,减少浪费,而且有助于服务器的负载均衡和节能。

基于以上特点,虚拟化技术成为实现云计算资源池化和按需服务的基础。为了进一步满足云计算弹性服务和数据中心自治性的需求,需要虚拟机快速部署和在线迁移技术的支持。

传统的虚拟机部署需要经过创建虚拟机、安装操作系统与应用程序、配置虚拟机属性以及应用程序运行环境、启动虚拟机四个阶段,通过修改虚拟机配置(如增减 CPU 数目、磁盘空间、内存容量等)可以改变单台虚拟机性能,但这个过程通常部署时间较长,不能满足云计算弹性服务的要求,为此,有的学者提出基于进程原理的虚拟机部署方式,利用父虚拟机迅速克隆出大量子虚拟机,就像启动很多子进程或线程那样快速部署虚拟机。利用分布式环境下的并行虚拟机 fork 技术,甚至可以在 1 秒内完成 32 台虚拟机的部署。

虚拟机在线迁移是指虚拟机在运行状态下从一台物理机移动到另一台物理机。利用虚拟机在线迁移技术,可以在不影响服务质量的情况下优化和管理数据中心,当原始虚拟机发生错误时,系统可以立即切换到备份虚拟机,而不会影响到关键任务的执行,保证了系统的

可靠性；在服务器负载高峰时期，可以将虚拟机切换至其他低峰服务器从而达到负载均衡；还可以在服务器集群处于低峰期时，将虚拟机集中放置，达到节能目的。因此虚拟机在线迁移技术对云计算平台有效管理具有重要意义。

3.3　Google 云计算原理

Google 公司有一套专属的云计算平台，这个平台先是为 Google 最重要的搜索应用提供服务，现在已经扩展到其他应用程序。Google 的云计算基础架构模式包括 4 个相互独立又紧密结合在一起的系统：Google File System 分布式文件系统[14]，针对 Google 应用程序的特点提出的 MapReduce 编程模式，分布式的锁机制 Chubby 以及 Google 开发的模型简化的大规模分布式数据库 BigTable。

3.3.1　GFS

网页搜索业务需要海量的数据存储，同时还需要满足高可用性、高可靠性和经济性等要求。为此，Google 基于以下几个假设开发了分布式文件系统——Google File System：

1）硬件故障是常态，充分考虑到大量节点的失效问题，需要通过软件将容错以及自动恢复功能集成在系统中。

2）支持大数据集，系统平台需要支持海量大文件的存储，文件通常大小以 GB 计，并包含大量小文件。

3）一次写入、多次读取的处理模式，充分考虑应用的特性，增加文件追加操作，优化顺序读写速度。

4）高并发性，系统平台需要支持多个客户端同时对某一个文件的追加写入操作，这些客户端可能分布在几百个不同的节点上，同时需要以最小的开销保证写入操作的原子性。

图 3-15 给出了 Google File System 的系统架构。如图所示，一个 GFS 集群包含一个主服务器和多个块服务器，被多个客户端访问。大文件被分割成固定尺寸的块，块服务器把块作为 Linux 文件保存在本地硬盘上，并根据指定的块句柄和字节范围读写块数据。为了保证可靠性，每个块被缺省保存 3 个备份。主服务器管理文件系统所有的元数据，包括名字空间、访问控制、文件到块的映射、块物理位置等相关信息。通过服务器端和客户端的联合设计，GFS 对应用支持达到性能与可用性最优。GFS 是为 Google 应用程序本身而设计的，在内部部署了许多 GFS 集群。有的集群拥有超过 1000 个存储节点，超过 300TB 的硬盘空间，被不同机器上的数百个客户端连续不断地频繁访问着。

3.3.2　MapReduce

Google 构造 MapReduce 编程规范简化分布式系统的编程。应用程序编写人员只需将精力放在应用程序本身，而关于集群的处理问题，包括可靠性和可扩展性，则交由平台来处理。MapReduce 通过“Map(映射)”和“Reduce(化简)”两个简单的概念构成运算基本单元，用户只需提供自己的 Map 函数以及 Reduce 函数即可并行处理海量数据。为了进一步理解

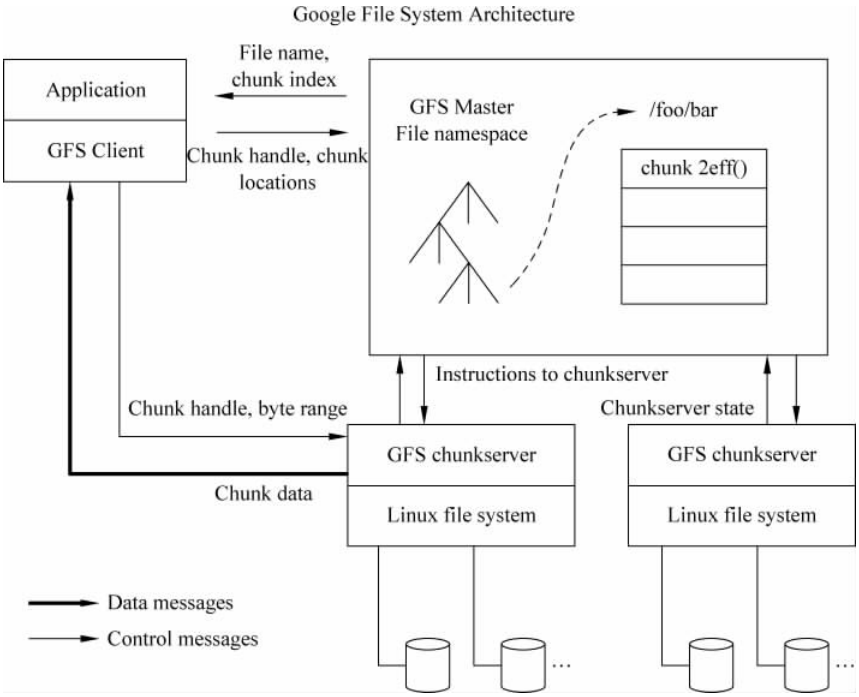

图 3-15 Google File System 的系统架构

MapReduce 的编程方式，下面给出一个基于 MapReduce 编程方式的程序伪代码。程序功能是统计文本中所有单词出现的次数。

```
map(String input_key, String input_value):
 // input_key: document name
 // input_value: document contents
 for each word w in input_value:
EmitIntermediate(w,"1");
reduce(String output_key, Interator intermediate_values):
 // output_key: a word
 // output_values: a list of counts
 int result = 0;
 for each v in intermediate_values:
result += ParseInt(v);
 Emit(AsString(result));
```

在 Map 函数中，用户的程序将文本中所有出现的单词都按照出现计数 1（以 Key-Value 对的形式）发射到 MapReduce 给出的一个中间临时空间中。通过 MapReduce 中间处理过程，将所有相同的单词产生的中间结果分配到同样一个 Reduce 函数中。而每一个 Reduce 函数则只需把计数累加在一起即可获得最后结果。

图 3-16 给出了 MapReduce 执行过程，分为 Map 阶段及 Reduce 两个阶段，都使用了集群中的所有节点。在两个阶段之间还有一个中间的分类阶段，即将包含相同的 key 的中间结果交给同一个 Reduce 函数执行。

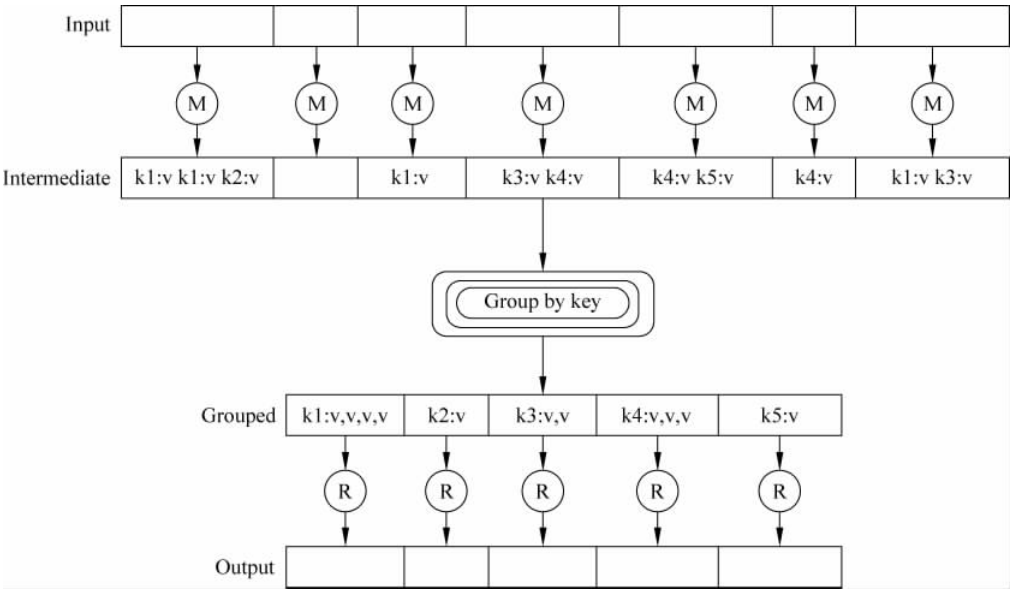

图 3-16　MapReduce 处理程序的执行过程(M 代表 Map 函数,R 代表 Reduce 函数)

3.3.3　BigTable

由于 Google 的许多应用(包括 Search History、Maps、Orkut 和 RSS 阅读器等)需要管理大量的格式化及半格式化数据,上述应用的共同特点是需要支持海量的数据存储,读取后进行大量的分析,数据的读操作频率远大于数据的更新频率等,为此 Google 开发了弱一致性要求的大规模数据库系统——BigTable。

BigTable 针对数据读操作进行了优化,采用基于列存储的分布式数据管理模式以提高数据读取效率。BigTable 的基本元素是行、列、记录板和时间戳,行键和列键都是字节串,时间戳是 64 位整型,可以用(row:string, column:string, time:int64)→string 表示一条键值对记录。其中,记录板 Table 就是一段行的集合体。

图 3-17 是 BigTable 的一个例子 Webtable,表 Webtable 存储了大量的网页和相关信息,在 Webtable,每一行存储一个网页,其反转的 url 作为行键,比如"com. google. maps",反转的原因是为了让同一个域名下的子域名网页能聚集在一起。

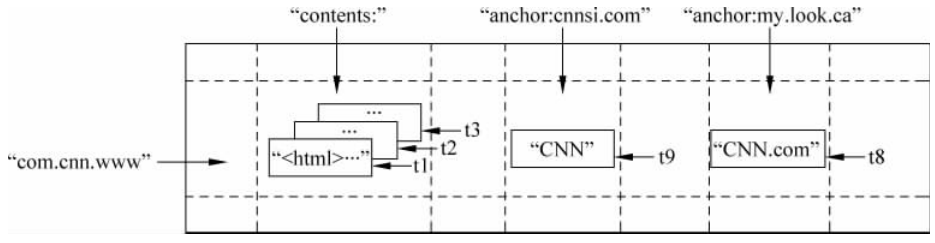

图 3-17　BigTable 的一个例子 Webtable

BigTable 中的数据项按照行关键字的字典序排列,行键可以是任意字节串,通常有 10～100B。BigTable 按照行键的字典序存储数据。BigTable 的表会根据行键自动划分为

片(tablet),片是负载均衡的单元。最初表都只有一个片,但随着表不断增大,片会自动分裂,片的大小控制在100～200MB。行是表的第一级索引,可以把该行的列、时间和值看成一个整体,简化为一维键值映射,类似于:

```
table{
  "com.cnn.www" : {sth.},          //一行,行键是 com.cnn.www
  "com.bbc.www" : {sth.},
  "com.google.www" : {sth.},
  "com.baidu.www" : {sth.}
}
```

列是第二级索引,每行拥有的列是不受限制的,可以随时增加减少。为了方便管理,列被分为多个列族(column family,是访问控制的单元),一个列族里的列一般存储相同类型的数据。一行的列族很少变化,但是列族里的列可以随意添加删除。列键按照 family:qualifier 格式命名的,如果将列的值和时间看作一个整体,那么 table 可以表示为二维键值映射,类似于:

```
table{
  "com.cnn.www" : {                  //一行
    "contents:":{sth.},              //一列,family 为 contents,qualifier 为空
    "anchor:cnnsi.com":{sth.},       //一列,family 为 anchor,qualifier 为 cnnsi.com
    "anchor:my.look.ca":{sth.}
  },
  "com.bbc.www" : {                  //一行
    "contents:":{sth.}
  },
  "hk.com.google.www" : {
    "contents:":{sth.},
    "anchor:youtube.com":{sth.}
  },
  "com.bing.cn" : {sth.}
}
```

也可以将 family 当作一层新的索引,类似于:

```
table{
  "com.cnn.www" : {                  //一行
    "contents":{sth.},               //family 为 contents
    "anchor":{
        "cnnsi.com":{sth.}
        "my.look.ca":{sth.}
    },                               //一列,family 为 anchor
  },
  "com.bbc.www" : {                  //一行
    "contents":{sth.}
  },
  "hk.com.google.www" : {
```

```
        "contents:":{sth.},
      "anchor":{
          "youtube.com":{sth.}
      }
    },
    "com.bing.cn" : {sth.}
  }
```

时间戳是第三级索引。BigTable 允许保存数据的多个版本，版本区分的依据就是时间戳。时间戳可以由 BigTable 赋值，代表数据进入 BigTable 的准确时间，也可以由客户端赋值。数据的不同版本按照时间戳降序存储，因此先读到的是最新版本的数据。我们加入时间戳后，就得到了 BigTable 的完整数据模型，类似于：

```
table{
  "com.cnn.www" : {                  //一行
    "contents:":{
        t1:"<html>…",                //t1 时刻的网页内容
        t2:"<html>…",                //t2 时刻的网页内容
        t3:"<html>…"                 //t3 时刻的网页内容
    },                               //一列,family 为 contents,qualifier 为空
    "anchor:cnnsi.com":{sth.},       //一列,family 为 anchor,qualifier 为 cnnsi.com
    "anchor:my.look.ca":{sth.}
  },
  "com.bbc.www" : {                  //一行
    "contents:":{sth.}
  },
  "hk.com.google.www" : {
    "contents:":{sth.},
    "anchor:youtube.com":{sth.}
  },
  "com.bing.cn" : {sth.}
}
```

图 3-17 中的列族“anchor”保存了该网页的引用站点（比如引用了 CNN 主页的站点），qualifier 是引用站点的名称，而数据是链接文本；列族“contents”保存的是网页的内容，这个列族只有一个空列“contents:”。“contents:”列下保存了网页的三个版本，可以用("com.cnn.www", "contents:", t5)找到 CNN 主页在 t1 时刻的内容。

BigTable 系统依赖于集群系统的底层结构，一个是分布式集群任务调度器，一个是前述的 GFS 文件系统，还有一个分布式锁服务 Chubby。如图 3-18 所示，Chubby 是一个非常健壮的粗粒度锁，BigTable 使用 Chubby 保存 Root Tablet 的指针，并使用一台服务器作为主服务器，用来保存和操作元数据。当客户端读取数据时，用户首先从 Chubby Server 中获得 Root Tablet 的位置信息，并从中读取相应的元数据表 Metadata Tablet 的位置信息，接着从 Metadata Tablet 中读取包含目标数据位置信息的 User Table 的位置信息，然后从该 User Table 中读取目标数据的位置信息项。

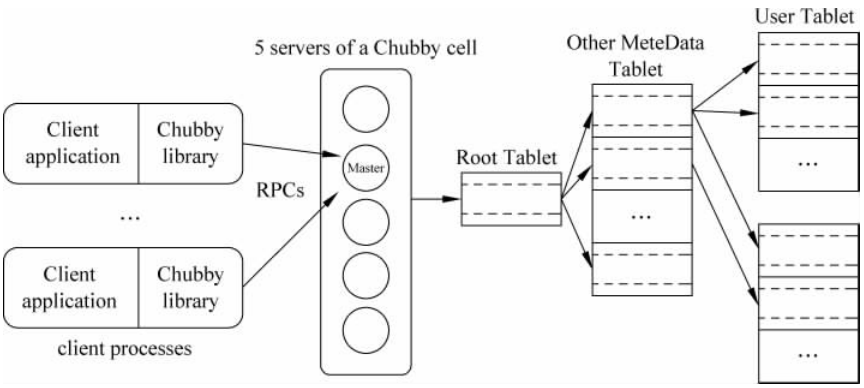

图 3-18 Chubby 的结构

3.3.4 Dremel

Dremel[17]是 Google 的“交互式”数据分析系统。可以组建成规模上千的集群，处理 PB 级别的数据。MapReduce 处理一个数据，需要分钟级的时间。作为 MapReduce 的发起人，Google 开发了 Dremel，将处理时间缩短到秒级，作为 MapReduce 的交互式查询能力不足的有力补充。

Dremel 的数据模型是嵌套的，用列式存储，并结合了 Web 搜索和并行 DBMS 的技术，建立查询树，将一个巨大的复杂的查询，分割成较小较简单的查询，大事化小，小事化了，能并发地在大量节点上跑，如图 3-19 所示，在这种按记录存储的模式中，一个记录的多列是连续写在一起的，按列存储可以将数据按列展开成查询树，扫描时可以仅仅扫描 A. B. C. 分支而不用扫描 A. E. 或 A. B. D. 分支，其次，Dremel 提供 SQL-like 接口，提供简单的 SQL 查询功能，可以将 SQL 语句转换成 MapReduce 任务执行。

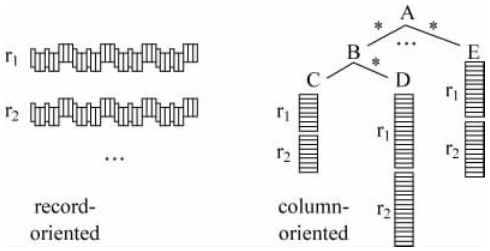

图 3-19 Google Dremel 数据模型

图 3-20 定义了一个组合类型 Document。有一个必选列 DocId，可选列 Links，还有一个数组列 Name。可以用 Name. Language. Code 表示 Code 列。

这种数据格式是语言无关、平台无关的。可以使用 Java 写 MapReduce 程序以生成这个格式，然后用 C++读取。在这种列式存储中，能够快速通用处理也是非常重要的。图 3-21 是数据在 Dremel 中的实际存储格式。

如果是关系型数据，而不是嵌套的结构，存储的时候，可以将每一列的值直接排列下来，不用引入其他概念，也不会丢失数据。对于嵌套的结构，还需要两个变量 R（Repetition

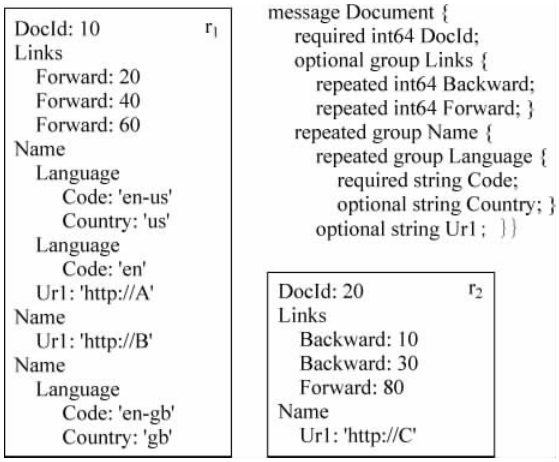

图 3-20　r_1，r_2 数据结构

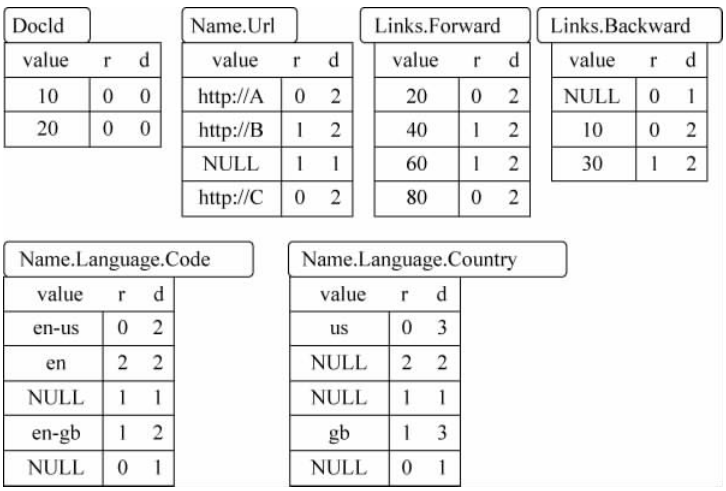

图 3-21　Document 类型的实际存储格式

Level)，D(Definition Level)才能存储其完整信息。Repetition Level 是记录该列的值是在哪一个级别上重复的。举例子说明，对于 Name. Language. Code 一共有三条非 Null 的记录：

第一个是“en-us”，出现在第一个 Name 的第一个 Language 的第一个 Code 里面。在此之前，这三个元素是没有重复过的，都是第一个。所以其 R 为 0。

第二个是“en”，出现在第一个 Name 的第二个 Language 里面。也就是说 Language 是重复的元素。Name. Language. Code 中 Language 嵌套位置是第二层，所以其 R 为 2。

第三个是“en-gb”，出现在第二个 Name 中的第一个 Language，Name 是重复元素，嵌套位置为第一层，所以其 R 为 1。

Definition Level 是定义的深度，用来记录该记录的实际层次。所以对于非 NULL 的记录，是没有意义的，其值必然为相同。同样举个例子，例如 Name. Language. Country：

第一个“us”是在 R1 里面，其中 Name，Language，Country 是有定义的。所以 D 为 3。

第二个“NULL”也是在 R1 的里面，其中 Name，Language 是有定义的，其他都是没有定义的。所以 D 为 2。

第三个“NULL”还是在 R1 的里面，其中 Name 是有定义的，其他是想象的。所以 D 为 1。

第四个“gb”是在 R1 里面，其中 Name，Language，Country 是有定义的。所以 D 为 3。

在这种存储格式下，读的时候，可以只读其中部分字段，构建部分的数据模型。例如，只读取 DocId 和 Name. Language. Country。可以同时扫描两个字段，先扫描 DocId 记录下第一个，然后发现下一个 DocId 的 R 是 0；于是该读 Name. Language. Country，如果下一个 R 是 1 或者 2 就继续读，如果是 0 就开始读下一个 DocId。图 3-22 展示了只读 DocId 和 Name. Language. Country 构建部分数据模型。

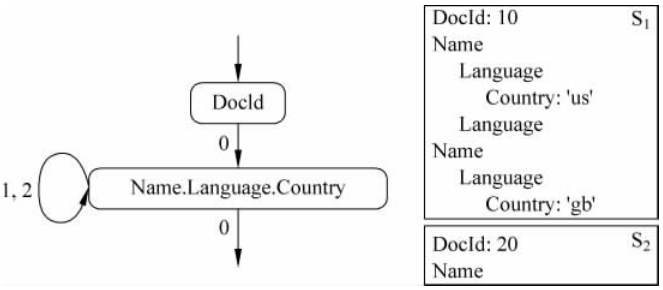

图 3-22　只读 DocId 和 Name. Language. Country 构建部分数据模型

Dremel 的扫描方式是全表扫描，而这种列存储设计可以有效回避大部分 join 需求，做到扫描最少的列，Dremel 可以使用 Sql-like 的语法查询，建立查询树如图 3-23 所示，当 client 发出一个请求，根节点收到请求，根据 metedata 将其分解到叶子节点，叶子节点直接扫描数据，不断汇总到根节点。这样就把对大数据集的查询分解为对很多小数据集的并行查询，因此，Dremel 的分析处理速度非常快。

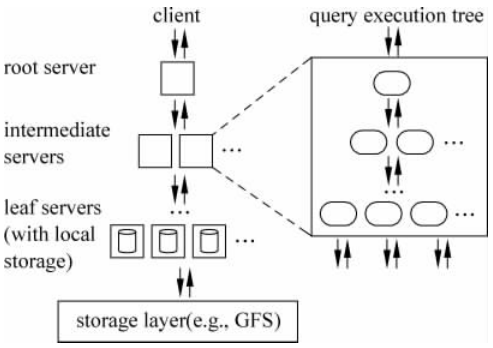

图 3-23　Dremel 的查询方式

Dremel 是一个大规模系统。在一个 PB 级别的数据集上面，将任务缩短到秒级，无疑需要大量的并发。磁盘的顺序读速度在 100MB/s 上下，那么在 1s 内处理 1TB 数据，意味着至少需要有 1 万个磁盘的并发读！Google 一向是用廉价机器办大事的好手。但是机器越多，出问题的概率越大，如此大的集群规模，需要有足够的容错考虑，保证整个分析的速度不

受集群中个别慢(坏)节点的影响。

3.4　亚马逊云服务

作为全球最大的电子商务网站,Amazon(亚马逊)为了处理数量庞大的并发访问和交易购置了大量服务器。2001 年互联网泡沫使业务量锐减,系统资源大量闲置。在这种背景下,Amazon 给出一个创新的想法是,将硬件设施等基础资源封装成服务供用户使用,即通过虚拟化技术提供可动态调度的弹性服务(IaaS)。之后经过不断的完善,现在的亚马逊云服务(Amazon Web Services,AWS)提供一组广泛的全球计算、存储、数据库、分析、应用程序和部署服务,可帮助组织更快地迁移、降低 IT 成本和扩展应用程序。很多大型企业和热门的初创公司都信任这些服务,并通过这些服务为各种工作负载提供技术支持,包括:Web 和移动应用程序、数据处理和仓库、存储、归档和很多其他工作负载。目前,亚马逊云服务主要包括[6]:弹性计算云 EC2、简单存储服务 S3、简单数据库服务 Simple DB、简单队列服务 SQS、弹性 MapReduce 服务、内容推送服务 CloudFront、数据导入/导出服务 AWS Import/Export、关系数据库服务 RDS 等。

3.4.1　亚马逊云平台存储架构

AWS 提供一系列云计算服务,无疑要建立在一个强大的基础存储架构之上,Dynamo 是 Amazon 提供的一款高可用的分布式 Key-Value 存储系统,具备去中心化、高可用性、高扩展性的特点,但是为了达到这个目标在很多场景中牺牲了一致性(CAP),能够跨数据中心部署于上万个节点上提供服务,Dynamo 组合使用了多种 P2P 技术,在集群中它的每一台机器都是对等的。

为了达到增量可伸缩性的目的,Dynamo 采用一致性哈希完成数据分区。在一致性哈希中,哈希函数的输出范围为一个圆环,系统中每个节点映射到环中某个位置,而 Key 也被 Hash 到环中某个位置,Key 从其被映射的位置开始沿顺时针方向找到第一个位置比其大的节点作为其存储节点。换个角度说,就是每个系统节点负责从其映射的位置起到逆时针方向的第一个系统节点间的区域。一致性哈希最大的优点在于节点的扩容与缩容,只影响其直接的邻居节点,而对其他节点没有影响。

在分布式环境中,为了达到高可用性需要有数据副本,而 Dynamo 将每个数据复制到 N 台机器上,其中 N 是每个实例的可配置参数,每个 Key 被分配到一个协调器(coordinator)节点,协调器节点管理其负责范围内的复制数据项,其除了在本地存储责任范围内的每个 Key 外,还复制这些 Key 到环上顺时针方向的 $N-1$ 个后继节点。这样,系统中每个节点负责环上从自己位置开始到第 N 个前驱节点间的一段区域。具体逻辑见图 3-24,图中节点 B 除了在本地存储键 Key K 外,还在节点 C 和 D 处复制键 K,这样节点 D 将存储落在范围(A, B]、(B, C]和(C, D]上的所有键。

Dynamo 并不提供强一致性,在数据并没有被复制到所有副本前,如果有 get 操作,会取到不一致的数据,但是 Dynamo 用向量时钟(vector clock)保证数据的最终一致性。在 Amazon 平台中,购物车就是这种情况的典型应用。购物车应用程序要求一个“添加到购物

车”动作从来不会被忘记或拒绝，当用户向当前购物车添加或删除一件物品时，如果当前购物车的状态是不可用，该物品会被添加到旧版本购物车中，并且不同版本的购物车会在后来协调，Dynamo 把版本合并的责任推给应用程序。也就是说，购物车应用程序会收到不同版本的数据，并负责合并，这种机制使得“添加到购物车”操作永远不会丢失，但是已删除的条目可能会“重新浮出水面”。图 3-25 是一个 Dynamo 提供最终一致性的具体例子：

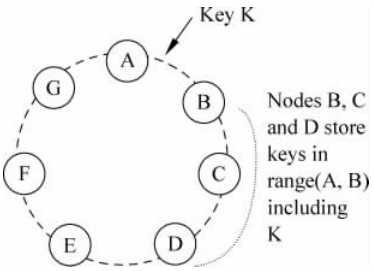

图 3-24　在 Dynamo 环上的分区与 Key 复制

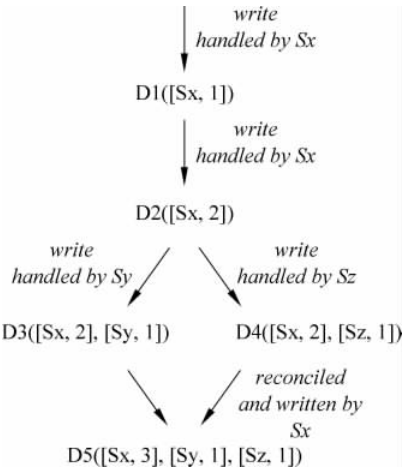

图 3-25　Dynamo 的最终一致性保证

（1）在某个时刻，某个节点 Sx 向系统写入了一个新对象，系统中有了该对象的一个版本 D1 和其相关的向量时钟[Sx,1]。

（2）随后节点 Sx 修改了 D1，系统中便有了不同的版本 D2 和其相关的时钟[Sx,2]，D2 继承自 D1，所以 D2 复写 D1，但系统中或许还存在还没有看到 D2 的 D1 副本。

（3）接下来不同的节点读取 D2，并尝试修改它，于是系统中有了版本 D3 和 D4 以及和他们相关的向量，现在系统中可能有了该对象的 4 个版本：D1，D2，D3，D4。

（4）接下来假设不同的客户端读取该对象，版本 D2 会覆盖版本 D1，而 D3 和 D4 会覆盖版本 D2，但如果客户端同时读到 D3 和 D4，就会由客户端进行语义协调（syntactically reconciled），如果交由 Sx 节点协调，Sx 将更新其时钟序号，将版本更新为（[Sx,3]，[Sy,1]，[Sz,1]）。

由于采用 P2P 对等模型和一致性哈希环，每个节点通过 Gossip 协议传播节点的映射信息得到自己所处理的范围，并互相检测节点状态，如果有新加入节点或故障节点，只需要调整处理范围内的节点。Dynamo 的高度伸缩性和高可用性的特点，为 Amazon 提供的各种上层服务提供可靠保证。

3.4.2　EC2、S3、SimpleDB 等组件

1. EC2

亚马逊弹性计算云（Elastic Compute Cloud，EC2）[19]是一个让使用者可以租用云端计算机运行所需应用的系统，提供基础设施层次的服务（IaaS）。EC2 提供了可定制化的云计

算能力，这是专为简化开发者开发 Web 伸缩性计算而打造的，EC2 借由提供 Web 服务的方式让使用者可以弹性地运行自己的 Amazon 虚拟机，使用者将可以在这个虚拟机器上运行任何需要的软件或应用程式。Amazon 为 EC2 提供简单的 Web 服务界面，让用户轻松获取和配置资源。用户以虚拟机为单位租用 Amazon 的服务器资源。用户可以全面掌控自身的计算资源，同时 Amazon 运作是基于“即买即用”模式的，只需花费几分钟时间就可获得并启动服务器实例，所以它可以快速定制响应计算需求的变化。

Amazon EC2 的优势有：在 AWS 云中提供可扩展的计算容量；使用 Amazon EC2 可避免前期的硬件投入，因此用户能够快速开发和部署应用程序；通过使用 Amazon EC2，用户可以根据自身需要启动任意数量的虚拟服务器、配置安全和网络以及管理存储；Amazon EC2 允许用户根据需要进行缩放以应对需求变化或流行高峰，降低流量预测需求。Amazon EC2 提供以下具体功能：

(1) 虚拟计算环境，也称为实例。

(2) 实例的预配置模板，也称为亚马逊系统映像(AMI)，其中包含服务器需要的程序包(包括操作系统和其他软件)。

(3) 实例 CPU、内存、存储和网络容量的多种配置，也称为实例类型。

(4) 使用密钥对的实例的安全登录信息(AWS 存储公有密钥，您在安全位置存储私有密钥)。

(5) 临时数据(停止或终止实例时会删除这些数据)的存储卷，也称为实例存储卷。

(6) 使用 Amazon Elastic Block Store (Amazon EBS)的数据的持久性存储卷，也称为 Amazon EBS 卷。

(7) 用于存储资源的多个物理位置，例如实例和 Amazon EBS 卷，也称为区域和可用区。

(8) 防火墙，让用户可以指定协议、端口，以及能够使用安全组到达实例的源 IP 范围。

(9) 用于动态云计算的静态 IP 地址，也称为弹性 IP 地址。

(10) 元数据，也称为标签，用户可以创建元数据并分配给 Amazon EC2 资源。

(11) 用户可以创建虚拟网络，这些网络与其余 AWS 云在逻辑上隔离，并且用户可以选择连接到自己的网络，也称为 Virtual Private Cloud (VPC)。

2. S3

Amazon S3(Simple Storage Service)[20]是一款在线存储服务，在云计算环境下提供了不受限制的数据存储空间。用户可通过授权访问一个简单的 Web 服务界面存储和获取 Web 上任何地点的数据。Amazon S3 提供了完全冗余的数据存储基础设施，用户可以将存储内容发送到 Amazon EC2 进行计算，调整大小或其他分析，Amazon S3 负责数据的持久、备份、存档与恢复等可靠服务。

S3 的基本结构如图 3-26 所示，S3 存储系统中涉及如下三个基本概念。

1) 对象：S3 的基本存储单元，由数据和元数据组成；数据可以是任意类型。

2) 键：对象的唯一标识符。

3) 桶：存储对象的容器；不能嵌套、在 S3 中名称唯一、每个用户最多创建 100 个桶。

S3 的操作流程如图 3-27 所示，用户登录 S3 后，首先创建一个桶(Bucket)，然后可以增

加一个数据对象(Object)到桶中,接着用户可以查看对象或移动对象,当用户不再需要存储数据时,则可以删除对象和桶。

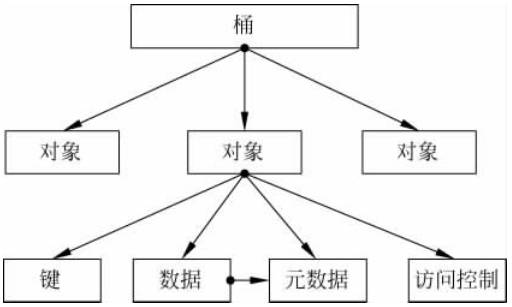

图 3-26 S3 的基本结构

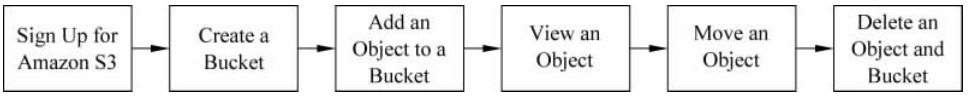

图 3-27 S3 的操作流程

3. SimpleDB

Amazon SimpleDB[21]是一种可用性高、灵活性大的非关系数据存储服务。与 S3 不同(主要用于非结构化数据存储),它主要用于存储结构化数据。开发人员只需通过 Web 服务请求执行数据项的存储和查询,Amazon SimpleDB 将负责余下的工作。

Amazon SimpleDB 不会受制于关系数据库的严格要求,而且已经过优化,能提供更高的可用性和灵活性,让管理负担大幅减少甚至是零负担。而在后台工作时,Amazon SimpleDB 将自动创建和管理分布在多个地理位置的数据副本,以此提高可用性和数据持久性。

SimpleDB 的操作流程如图 3-28 所示,用户注册登录后,然后可以创建一个域(Domain,它是存放数据的容器),然后可以向域中添加数据条目(Item,它是一个实际的数据对象,由属性和指组成),接着用户可以查看或修改域中的数据条目,当用户不再需要存储的数据时,则可以删除域。

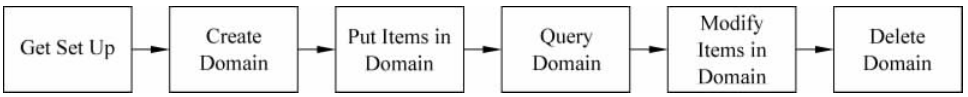

图 3-28 SimpleDB 的操作流程

4. SQS

Amazon SQS(Simple Queue Service)是面向消息的中间件(MOM)的云计算解决方案,不局限于某一种语言。Amazon SQS 提供了可靠且可扩展的托管队列,用于存储计算机之间传输的消息。使用 Amazon SQS,可以在执行不同任务的应用程序的分布式组件之间移动数据,既不会丢失消息,也不要求各个组件始终处于可用状态。Amazon SQS 是分布式队

列系统，可以让 Web 服务应用程序快速可靠地对应用程序中的一个组件生成给另一组件使用的消息进行排队。队列是等待处理的消息的临时储存库。

Amazon SQS 提供以下主要功能：

1）冗余基础设施：确保将消息至少传输一次、对消息的高度并发访问以及发送和检索消息的高度可用性；

2）多个写入器和读取器：系统的多个部分可以同时发送或接收消息；

3）每个队列的设置均可配置：并非所有队列都要完全相同；

4）可变消息大小：消息大小可高达 262 144 字节(256 KB)；

5）访问控制：用户可以控制谁可以从队列发送和收取消息；

6）延迟队列：延迟队列即用户对其设置默认延迟的队列，从而所有排队消息的传送会推迟那一段时间。

5. Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR)是一个能够高性能地处理大规模数据的 Web service。Amazon EMR 使用 Hadoop 处理方法，并结合多种 AWS 产品，可完成以下各项任务：Web 索引、数据挖掘、日志文件分析、机器学习、科学模拟以及数据仓库。

Amazon EMR 已增强了 Hadoop 和其他开源应用程序，以便与 AWS 无缝协作，如图 3-29 所示。例如，在 Amazon EMR 上运行的 Hadoop 集群使用 EC2 实例作为虚拟 Linux 服务器用于主节点和从属节点，将 Amazon S3 用于输入和输出数据的批量存储，并将 Amazon CloudWatch 用于监控集群性能和发出警报。用户还可以使用 Amazon EMR 和 Hive 将数据迁移到 Amazon DynamoDB 以及从中迁出。所有这些操作都由启动和管理 Hadoop 集群的 Amazon EMR 控制软件进行编排。这个流程名为 Amazon EMR 集群。

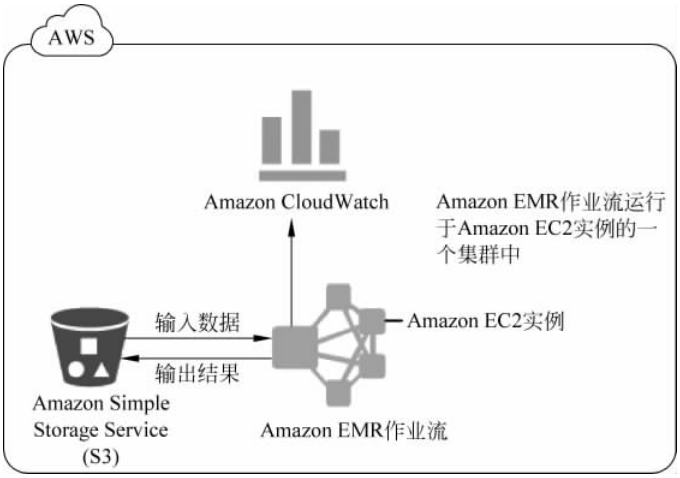

图 3-29　Elastic MapReduce

在 Hadoop 架构顶层运行的开源项目也可以在 Amazon EMR 上运行。最流行的应用程序，例如 Hive、Pig、HBase、DistCp 和 Ganglia，都已与 Amazon EMR 集成。

通过在 Amazon EMR 上运行 Hadoop，可以从云计算获得以下好处：

(1) 能够在几分钟内调配虚拟服务器集群。

(2) 可以扩展集群中虚拟服务器的数量满足计算需求,而且仅需按实际使用量付费。

(3) 与其他 AWS 服务集成。

6. CloudFront

CloudFront 是一个内容分发网络服务(Web service),该服务可以很容易地将内容投送到终端用户,具有低延迟、高数据传输速率等特点。简单来说,就是使用 CDN 进行网络加速和向最终用户分发静态和动态 Web 内容(例如,. html、. css、. php 和图像文件)。CloudFront 通过一个由遍布全球的数据中心(称作节点)组成的网络传输内容。当用户请求用 CloudFront 提供的内容时,用户的请求将被传送到延迟(时延)最短的节点,以便以可以达到的最佳性能来传输内容。如果该内容已经在延迟最短的节点上,CloudFront 将直接提供它。如果该内容目前不在这样的节点上,CloudFront 将从已指定为该内容最终版本来源的 Amazon S3 存储桶或 HTTP 服务器(例如,Web 服务器)检索该内容。

内容推送服务 CloudFront 集合了其他的 Amazon 云服务,为企业和开发者提供了一种简单方式,以实现高速传输分发数据。同 EC2 和 S3 最优化地协同工作,CloudFront 使用涵盖了边缘的全球网络交付静态和动态内容。配置 CloudFront 传输用户的内容信息的步骤如图 3-30 所示:①配置原始服务器,CloudFront 将从这些服务器中获取文件,以便从遍布全球的 CloudFront 节点进行分发;②将用户的文件(也称作对象,通常包括网页、图像和媒体文件)上传至原始服务器;③创建一项 CloudFront 分配,此项分配将在其他用户请求文件时,告诉 CloudFront 从哪些原始服务器获取用户分发的文件;④在开发网站或应用程序时,可以使用 CloudFront 为用户的 URL 提供的域名;⑤CloudFront 将此项分配的配置(而不是用户的内容)发送到其所有节点,这些节点即服务器的集合,位于分散在不同地理位置的数据中心内。

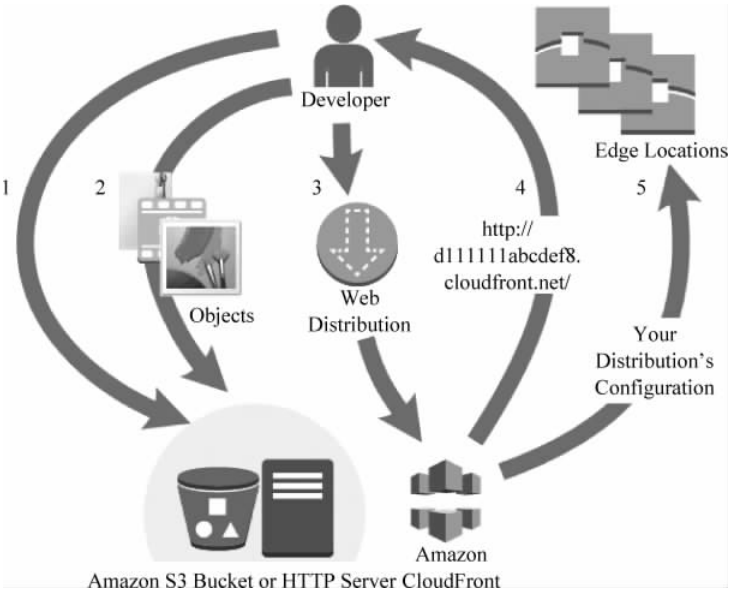

图 3-30 CouldFront

7. AWS Import/Export

AWS Import/Export 工具采用 Amazon 公司内部的高速网络和便携存储设备，绕过互联网对 Amazon 云上的数据导入导出，所以 Import/Export 通常快于互联网的数据传输。

AWS Import/Export 支持从 S3 的桶中上传和下载数据、数据上传到亚马逊弹性块存储(Amazon EBS)中。AWS Import/Export 的操作流程如图 3-31 所示，在使用 AWS Import/Export 上传和下载数据前，用户需要使用 S3 的账号登录，然后下载 Import/Export 工具，再保存用户证书文件，接着可以创建一个导入或导出数据的任务。

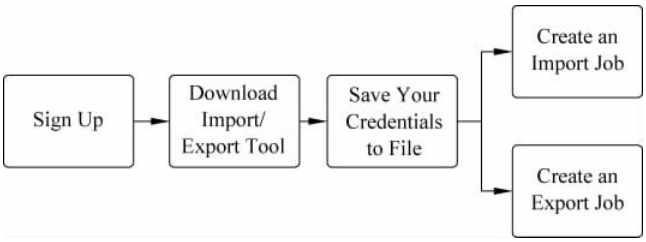

图 3-31　AWS Import/Export 的操作流程

8. RDS

Amazon Relational Database Service (Amazon RDS)[22]是一种 Web 服务，可让用户更轻松地在云中设置、操作和扩展关系数据库。它可以为行业标准关系数据库提供经济高效且可以调节大小的容量，并管理常见数据库管理任务。

Relational Database Service(RDS，关系数据库服务)在云计算环境下通过 Web 服务提供了弹性化的关系数据库。接管数据库的管理员任务，以前使用 MySQL 数据库的所有代码、应用和工具都可兼容 Amazon RDS。它可以自动地为数据库软件打补丁并完成定期的按计划备份。

Amazon RDS 会接管关系数据库的许多困难或烦琐的管理任务。

(1) 购买服务器时，您会一并获得 CPU、内存、存储空间和 IOPS。利用 Amazon RDS，您可以将这些部分进行拆分，以便单独对其进行扩展。因此，如果您需要更多 CPU、更少 IOPS 或更多存储空间，就可以轻松地对它们进行分配。

(2) Amazon RDS 可以管理备份、软件修补、自动故障检测和恢复。

(3) 为了让用户获得托管式服务体验，Amazon RDS 未提供对数据库实例的 Shell 访问权限，并且限制对需要高级特权的某些系统程序和表的访问权限。

(4) 可以在需要时执行自动备份，或者创建自己的备份快照。这些备份可用于还原数据库，并且 Amazon RDS 的还原过程可靠且高效。

(5) 可以通过主实例和在发生问题时向其执行故障转移操作的同步辅助实例实现高可用性。还可以使用 MySQL 只读副本增加只读扩展。

(6) 可以使用您已熟悉的数据库产品：MySQL、PostgreSQL、Oracle 和 Microsoft SQL Server。

(7) 除了数据库包的安全外，使用 AWS IAM 定义用户和权限，还有助于控制可以访问 RDS 数据库的人员。此外，将数据库放置在虚拟私有云中，也有助于保护数据库。

3.5 基于亚马逊云的大数据分析案例

3.5.1 亚马逊云平台存储架构

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。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。

AWS 提供了一系列广泛的服务，以帮助用户快速轻松地构建和部署大数据分析应用程序。借助 AWS，可以快速访问灵活的低成本 IT 资源，可以迅速扩展几乎任何大数据应用程序，其中包括数据仓库、单击流分析、欺诈侦测、推荐引擎、事件驱动 ETL、无服务器计算和物联网等应用程序。借助 AWS，无须在前期投入大量的时间和费用来构建和维护基础设施。相反地，可以精确地预置支持大数据应用程序所需的资源类型和大小。AWS 提供的服务覆盖的领域非常广泛，帮助用户在云中对大数据进行收集、存储、处理、分析和可视化。在大数据分析框架方面，提供了 Amazon EMR、Amazon Elasticsearch Service 和 Amazon Athena(交互式查新服务)，这些服务可用于大数据的托管型分布式计算；在实时大数据分析方面，提供了 Amazon Kinesis Analytics、Amazon Kinesis Streams 和 Amazon Kinesis Firehose，这些服务可用于加载和分析流数据的强大服务；在大数据存储与数据库方面，提供了对象存储 Amazon S3、Amazon NoSQL 存储 DynamoDB 和 HBase、关系数据库 Amazon RDS 等，可实现安全、持久、高度可扩展的大数据存储；在大数据计算方面，提供了 Amazon EC2、Amazon EC2 Container Registry 和 Container Service；用于商业智能分析的 Amazon QuickSight；在大数据迁移方面，AWS Database Migration Service 和 AWS Server Migration Service；数据仓库服务 Amazon Redshift，可以使用高性能本地磁盘上的列式存储通过复杂的查询优化对 PB 级结构化数据运行复杂的分析查询，并能大规模执行并行查询；机器学习服务 Amazon Machine Learning，提供可视化工具和向导来指导用户完成机器学习(ML)模型的创建过程。可帮助用户轻松并安全地将数据库迁移至 AWS。下面针对其中部分主要大数据分析服务进行简介，更详细的技术说明请参考 AWS 官网主页的大数据服务介绍[25]。

Amazon EMR 提供的托管 Hadoop 框架可以让用户快速轻松、经济高效地在多个动态可扩展的 Amazon EC2 实例之间处理大量数据。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，同时允许用户使用 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。用户还可以运行其他常用的分布式框架(例如 Amazon EMR 中的 Apache Spark、HBase、Presto 和 Flink)，以及与其他 AWS 数据存储服务(例如 Amazon S3 和 Amazon DynamoDB)中的数据进行交互。Amazon EMR 能够安全可靠地处理广泛的大数据使用案例，包括日志分析、Web 索引、数据转换(ETL)、机

器学习、财务分析、科学模拟和生物信息。

Amazon Athena是一种交互式查询服务，能够使用SQL分析Amazon S3中的数据。Athena没有服务器，因此无须管理任何基础设施。用户只需指向存储在Amazon S3中的数据，定义架构并使用标准SQL开始查询。在数秒内即可获得结果。借助Athena，用户无须为了进行分析而执行复杂的ETL任务准备数据。因此，具备SQL技能的任何人都可以轻松快速地分析大规模数据集。

Amazon Redshift是一种快速且完全托管的数据仓库，允许用户使用标准SQL和现有的商业智能(BI)工具分析用户的所有数据。利用Amazon Redshift，用户可以使用本地高性能磁盘上的列式存储通过复杂的查询优化对PB级结构化数据运行复杂的分析查询，并能大规模执行并行查询。大多数结果在几秒内即可返回。Amazon Redshift还包含Redshift Spectrum，让用户可以对Amazon S3中的EB级非结构化数据直接运行SQL查询。不需要加载或转换，并允许用户使用Avro、CSV、Grok、ORC、Parquet、RCFile、RegexSerDe、SequenceFile、TextFile和TSV等开源数据格式。Redshift Spectrum可以根据检索的数据自动扩展查询计算容量，因此对Amazon S3的查询速度非常快，不受数据集大小的影响。

AWS Glue是一项完全托管的提取、转换和加载(ETL)服务，让用户能够轻松准备和加载数据进行分析。用户只需将AWS Glue指向存储在AWS上的数据，AWS Glue便会发现用户的数据，并将关联的元数据(例如表定义和架构)存储在AWS Glue数据目录中。存入目录后，这些数据可立即供ETL搜索、查询和使用。AWS Glue可生成代码执行数据转换和数据加载流程。AWS Glue可生成可自定义、可重复使用且可移植的Python代码。ETL作业准备就绪后，用户就可以安排它在AWS Glue完全托管的横向扩展Apache Spark环境中运行。AWS Glue可提供一个具有依赖关系解析、作业监控和警报功能的灵活计划程序。如图3-32所示，使用AWS Glue数据目录跨多个AWS数据集快速发现和搜索数据，无须移动数据。数据存入目录后，可以使用Amazon Athena、Amazon EMR和Amazon Redshift Spectrum对其进行搜索和查询。

Amazon Elasticsearch Service允许用户部署、保护、操作和扩展Elasticsearch，以便进行日志分析、全文检索和应用程序监控等工作。Amazon Elasticsearch Service是一项完全托管的服务，可以提供各种易于使用的Elasticsearch API和实时分析功能，还可以实现生产工作负载需要的可用性、可扩展性和安全性。本服务在内部集成了Kibana、Logstash以及Amazon Virtual Private Cloud (VPC)、Amazon Kinesis Firehose、AWS Lambda和Amazon CloudWatch等AWS服务，因此用户可以将原始数据安全快速地转变为可付诸实施的分析结果。AWS管理控制台使得设置和配置Amazon Elasticsearch Service域变得很方便。Amazon Elasticsearch Service可以为用户的域预置所有资源并启动域。用户可以从自己的VPC或公共终端节点访问域。本服务可以自动检测并替换出现故障的Elasticsearch节点，减少与自管理的基础设施和Elasticsearch软件相关的开销。Amazon Elasticsearch Service让用户只需要通过单个API调用或在控制台中单击几次就可以轻松扩展群集。利用Amazon Elasticsearch Service，用户可以直接访问Elasticsearch开源API，因此已经用于现有Elasticsearch环境的代码和应用程序都可以流畅工作。

Amazon Kinesis能够帮助用户收集、处理和分析实时流数据，从而及时地了解新信息

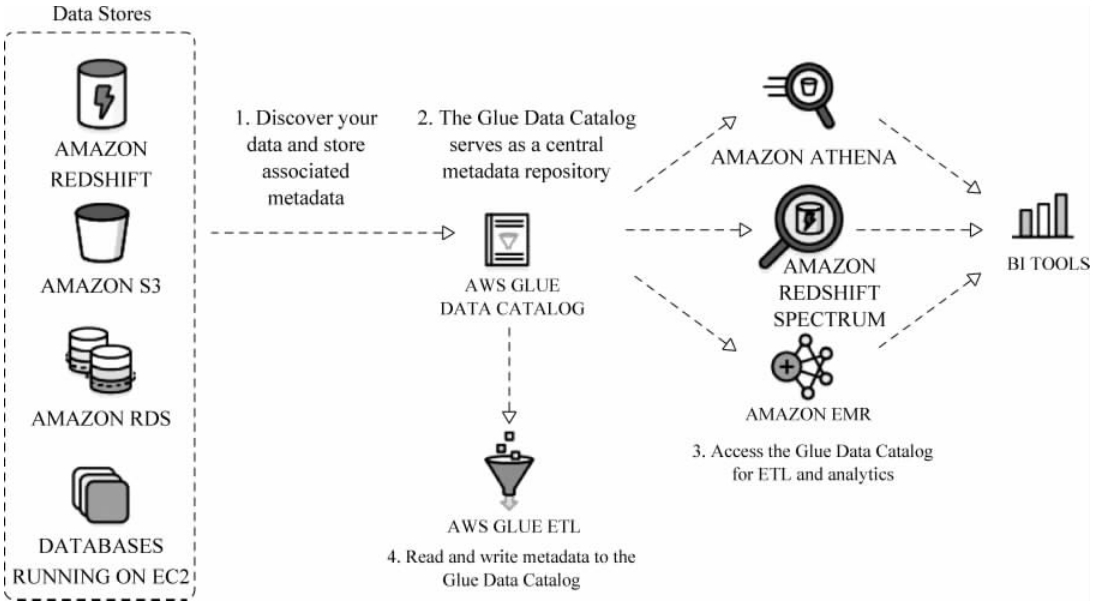

图 3-32 跨多个数据存储的统一数据视图用例

并快速做出反应。Amazon Kinesis 提供多种核心功能，可以处理任意规模的流数据，同时具有很高的灵活性，让用户可以选择最符合应用程序需求的工具。Amazon Kinesis 的功能具体包括将流数据轻松加载到 AWS 的 Amazon Kinesis Firehose 服务、使用标准 SQL 处理和分析流数据 Amazon Kinesis Analytics 服务、构建用于处理和分析流数据的自定义应用程序 Amazon Kinesis Streams 服务。借助 Amazon Kinesis，用户可以在数据库、数据湖和数据仓库中接收应用程序日志、网站单击流、IoT 遥测数据等实时数据，也可以使用这些数据构建自己的应用程序。Amazon Kinesis 允许用户对收到的数据进行实时处理和分析并做出响应，无须等到收集完全部数据后才开始分析。如图 3-33 所示，给出 Amazon Kinesis 用例，实现使用 SQL 实时处理和分析流数据。

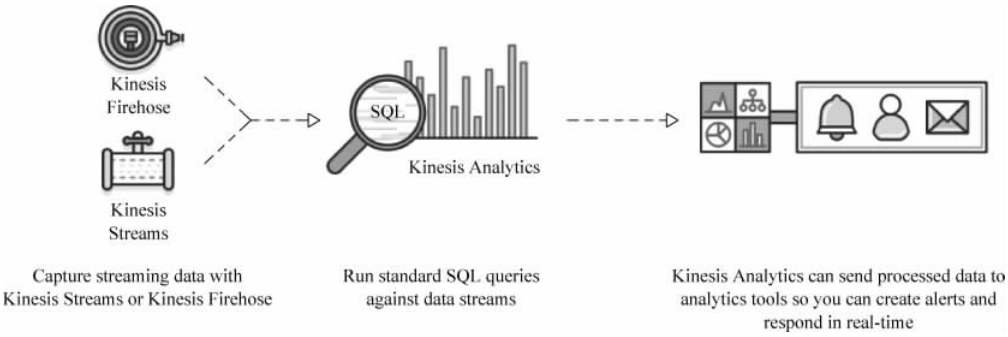

图 3-33 Amazon Kinesis 用例：使用 SQL 实时处理和分析流数据

AWS Data Pipeline 是一种 Web 服务，可帮助用户可靠地处理数据并以指定的间隔在不同 AWS 计算与存储服务以及内部数据源之间移动数据。利用 AWS Data Pipeline，用户可以定期在存储数据的位置访问数据，大规模转换和处理数据，高效地将结果传输到各种

AWS服务中，例如Amazon S3、Amazon RDS、Amazon DynamoDB和Amazon EMR。该服务帮助用户轻松创建具有容错、可重复和高可用性特征的复杂数据处理工作负载。用户无须确保资源可用性，管理作业间的从属性，或担心重试瞬时失效或超时的单个任务，以及创建故障通知系统等问题。

Amazon QuickSight是基于云端的业务分析服务，允许用户进行数据可视化，执行特设分析(ad-hoc analysis)，通过方便的分析数据方法洞察业务的走向。该服务允许用户轻松连接到用户的数据，执行高级分析，创建可从任何浏览器或移动设备访问到的丰富的由数据可视化生成的图表。

Amazon CloudSearch是一种在AWS云中托管的服务，允许用户为网站或应用程序设置、管理或扩展搜索解决方案。Amazon CloudSearch支持34种语言和常用搜索功能(如突出显示、自动完成和地理空间搜索)。借助Amazon CloudSearch，用户可以迅速为自己的网站或应用程序添加丰富的搜索功能。用户不需要担心硬件的预配置、设置和维护，在AWS管理控制台中可以创建一个搜索域，又或者上传希望能搜索的数据，Amazon CloudSearch会自动预配置所需的资源，并部署一个高度优化的搜索索引。用户可以随时更改搜索参数、优化搜索相关性和应用新设置。随着数据量和流量变化，Amazon CloudSearch会进行无缝调整，满足不同的需求。

3.5.2　亚马逊云的Web服务器日志大数据分析案例

1. Web服务器日志大数据分析的需求与分析

随着Web服务的发展，几乎各个政府部门、企业公司、科研院校等都拥有了自己的网站。而与此同时，在构建和管理网站中，各个单位都会遇到各种各样的问题。为了了解网站的运行情况，发现网站存在的不足，促进网站更好地发展，对于Web服务器的运行和访问情况进行详细周全的分析是非常重要的。管理网站需要监视Web的速度、内容传输、吞吐、访问等数据，通过这些数据可以更好地了解网站的情况，得到需要进行改善的地方。而处理这些需求，可以通过对Web服务器的日志文件进行分析来做到。

假设我们托管一个受欢迎的电子商务网站，想要分析Apache Web日志，以了解人们发现我们的网站的方式，从而希望确定网站的哪一个在线广告活动推动了最多的在线商店流量。通常，对于数据的处理，我们会想到利用如MySQL等关系型数据库进行处理。但是，Web服务器日志因过于庞大而无法导入MySQL数据库，并且它们未采用关系格式。因此我们需要使用其他方法分析这些日志。Amazon EMR将Hadoop和Hive等开源应用程序与Amazon Web Services集成，可为分析Apache Web日志等大规模数据提供可扩展的高效架构。

本案例的目标在于使用AWS创建Amazon EMR集群，导入需要分析的日志文件，利用集群中Hive应用程序，对日志文件的数据进行处理并分析，从而得出我们需要的信息。

2. Web服务器日志大数据分析的设计

WordCount在本案例中，日志文件数据来自Apache服务器网站生成的日志文件。日志文件的生成由Apache Commons Logging完成，生成的文件存储在服务器主机上。Apache Commons Logging，又叫作JakartaCommons Logging(JCL)，它提供的是一个日志

(Log)接口(interface),同时兼顾轻量级和不依赖于具体的日志实现工具。它提供给中间件/日志工具开发者一个简单的日志操作抽象,允许程序开发人员使用不同的具体日志实现工具。用户被假定已熟悉某种日志实现工具的更高级别的细节。JCL 提供的接口,对其他一些日志工具,包括 Log4J、Avalon LogKit 和 JDK 等,进行了简单的包装,此接口更接近于 Log4J 和 LogKit 的实现。

日志文件的转存主要使用 Amazon S3 服务。我们会将即将进行分析的文件从网站服务器上机上传到 Amazon S3 服务,以进行后续进一步的处理。Amazon S3 是专为从任意位置存储和检索任意数量的数据而构建的对象存储,这些数据包括来自网站和移动应用程序、公司应用程序的数据以及来自 IoT 传感器或设备的数据。

日志文件的数据主要使用 Hive 应用程序进行处理和分析。Hive 是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具,可以用来进行数据提取转化加载(ETL),这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言,称为 HQL,它允许熟悉 SQL 的用户查询数据。同时,这个语言也允许熟悉 MapReduce 的开发者开发自定义的 mapper 和 reducer 处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。Hive 没有专门的数据格式。Hive 可以很好地工作在 Thrift 之上,控制分隔符,也允许用户指定数据格式。AWS 的 Amazon EMR 集群已将 Hadoop 和 Hive 等应用程序集成,因此可以利用 AWS 创建相应的 Amazon EMR 集群。

因此,在本案例中我们的目的是:从 Amazon S3 导入数据(日志文件)并创建 Amazon EMR 集群。随后,连接到集群的主节点,并在其中运行 Hive 以便使用简化的 SQL 语法查询 Apache 日志。

本案例的数据日志来源于国内某技术学习论坛,该论坛由某培训机构主办,汇聚了众多技术学习者,每天都有人发帖、回帖,如图 3-34 所示。日志数据的具体下载地址为:http://blog.csdn.net/mergerly/article/details/52759903。

图 3-34 日志文件来源论坛网站

本案例中所用到的 Web 日志文件格式为.log,数据格式如图 3-35 所示。

以第一行日志为例,变量的解释如下:

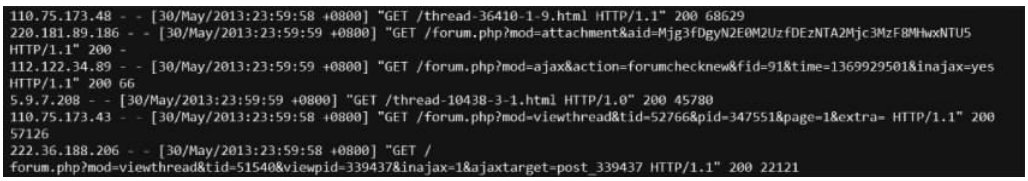

图 3-35　日志文件示例

remote_addr：记录客户端的 IP 地址。110.75.173.48。

remote_user：记录客户端用户名称。-。

time_local：记录访问时间与时区。[30/May/2013:23:59:58 +0800]。

request：记录请求的 URL 与 HTTP 协议。“GET /thread-36410-1-9.html HTTP/1.1”。

status：记录请求状态，成功是 200。200。

body_bytes_sent：记录发送给客户端文件主体内容大小。19939。

http_referer：用来记录从哪个页面链接访问过来的。NULL。

http_user_agent：记录客户浏览器的相关信息。NULL。

本案例使用两个 Web 日志文件分别为 58.3MB 的 access_2013_05_30.log、149.8MB 的 access_2013_05_31.log 作为分析对象，并将其上传至 Amazon S3 服务，以方便 Amazon EMR 集群的导入及分析。

3. Web 服务器日志大数据分析的实现过程

基于亚马逊云的 Web 服务器日志大数据分析的具体实现过程分为以下 8 个步骤。

步骤 1：创建密钥对

在首次使用 AWS 的服务之前，我们首先要申请一个账号。AWS 使用公钥加密保护我们的实例的登录信息。一个 Linux 实例没有密码，但是我们需要使用密钥对安全地登录到实例。在启动实例时指定密钥对的名称，然后在使用 SSH 登录时提供私钥。对于 Windows 实例，需要获取实例的管理员密码，以便使用 RDP 登录。

由于需要用到的 Hadoop 框架是在 Linux 系统上的，所以为了能让计算机连接上 AWS 的服务，需要在 AWS 上创建一个密钥对(key pair)。登录账号，然后会进入到控制台页面(见图 3-36)。单击“计算”模块中的“EC2”，进入 EC2 控制面板(见图 3-37)。

从左边的菜单栏中选择“网络与安全”→“密钥对”，在出现的密钥对管理页面(见图 3-38)中选择“创建密钥对”。

输入密钥对名称(见图 3-39)。此名称用于 SSH 登录时选择对应的密钥对，以便能成功登录到实例。

单击创建之后网站，会自动开始该密钥对的私钥文件(格式为 pem)的下载任务(见图 3-40)，我们需要将私钥保存到安全的地方。这里将密钥对命名为 doc_exp。

然后，就可以看到所创建的密钥对在密钥管理页面中的显示(见图 3-41)。

对于 Mac OS X 和 Linux 系统，我们可以用以下命令使得私钥文件只有我们可读。把其中的红色字体部分(doc_exp)换成我们创建时的密钥对名称。

```
$ chmod 400 doc_exp.pem
```

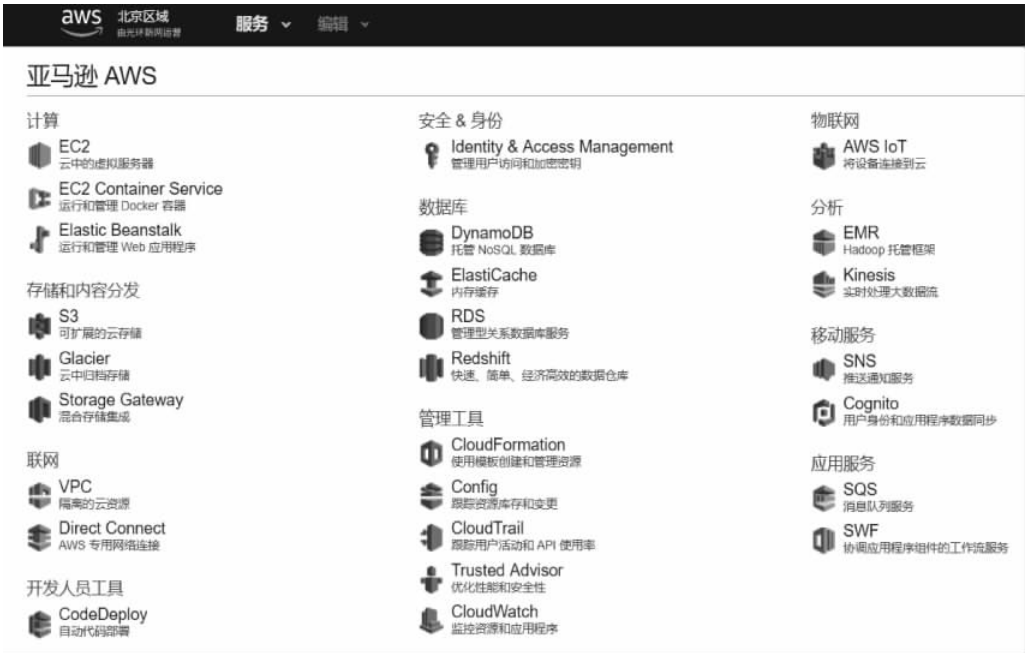

图 3-36　AWS 控制台

图 3-37　EC2 控制面板

图 3-38　密钥对管理页面

图 3-39　输入密钥对名称

图 3-40　下载私钥文件

图 3-41　管理密钥对

对于 Windows 系统，需要使用 PuTTY 把. pem 文件转换为. ppk 文件使用。具体步骤如下。

- 从 http://www. chiark. greenend. org. uk/~sgtatham/putty/下载 PuTTY 并安装。
- 打开 PuTTYgen（例如，打开“开始”菜单，单击“所有程序”→“PuTTY”→“PuTTYgen”）。
- 在 Type of key to generate 下，选择“SSH-2 RSA”。
- 单击“load”，在默认情况下，PuTTYgen 只显示. ppk 文件，为了定位到我们的. pem 文件，我们需要选择显示所有格式的文件。
- 选择我们的私钥文件并单击“Open”，单击“OK”按钮关闭确认对话框。
- 单击 Save private key，PuTTYgen 会弹出一个有关“Saving the key without a passphrase”的警告提示，单击“Yes”。
- 指定与我们的密钥对相同的名称并单击“Save”，PuTTYgen 将会自动生成. ppk 的文件。

步骤 2：创建 Amazon EMR 集群

接下来，开始进行实验环境的搭建。首先返回 AWS 控制台页面（见图 3-36），单击“分析”下的“EMR”，进入 Amazon EMR 管理页面（见图 3-42）。

图 3-42　Amazon EMR 管理页面

单击“创建集群”，开始集群的创建。然后单击“创建集群-快速选项”右边的“转到高级选项”按钮，进行更详细的配置（见图 3-43）。

图 3-43　开始集群的创建

在“软件配置”下，选择最新的 AMI Version（AMI 版本）（见图 3-44）。由于我们只需要用到 Hadoop 和 Hive，所以保留 Hadoop 和 Hive 的选择，将其他工具的选择去掉（见图 3-45）。然后单击“下一步”按钮。

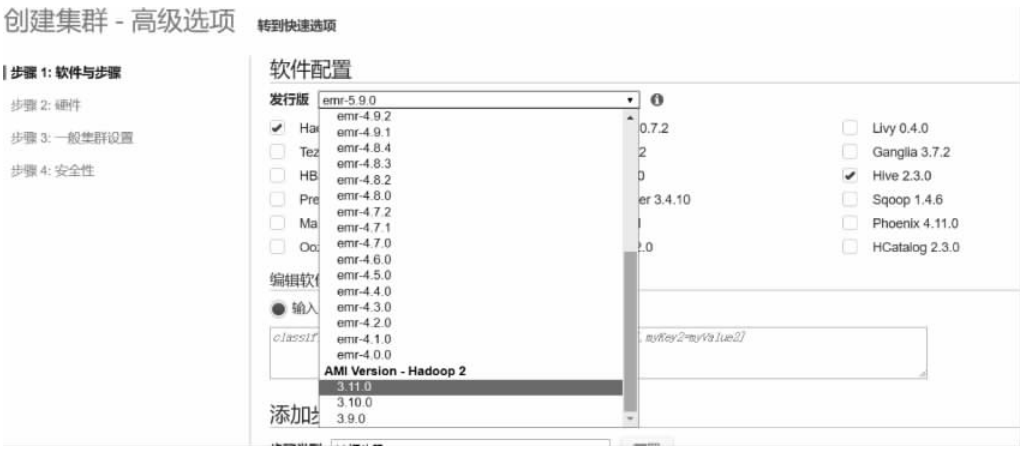

图 3-44　选择 AMI 版本

图 3-45　保留需要的工具

在“硬件配置”下，保留默认设置（见图 3-46）。默认的硬件配置包括一个主实例（主节点：8 个 CPU，15GB 内存，80GB 硬盘容量），两个核心实例（计算节点：8 个 CPU，15GB 内存，80GB 硬盘容量）。

图 3-46　保留默认硬件配置

在“一般选项”下，输入我们的集群名称，去除“日志记录”及“终止保护”的选择（见图 3-47）。其他设置保持默认。

图 3-47　一般选项

在“安全选项”下，在“EC2 键对”选择之前生成的密钥对，其他设置保持默认（见图 3-48）。然后单击“创建集群”结束集群的配置，并创建集群。

集群的创建需要一点时间，这时候需要耐心等候一段时间，直到页面上方的提示“正在启动”（见图 3-49）变为“正在等待”（见图 3-51）。第一次创建集群的时候，“网络和硬件”部分可能需要对用户的身份进行验证（见图 3-50），耗费的时间会比平常多。

图 3-48 安全选项

图 3-49 正在进行集群的配置

图 3-50 对账户进行验证

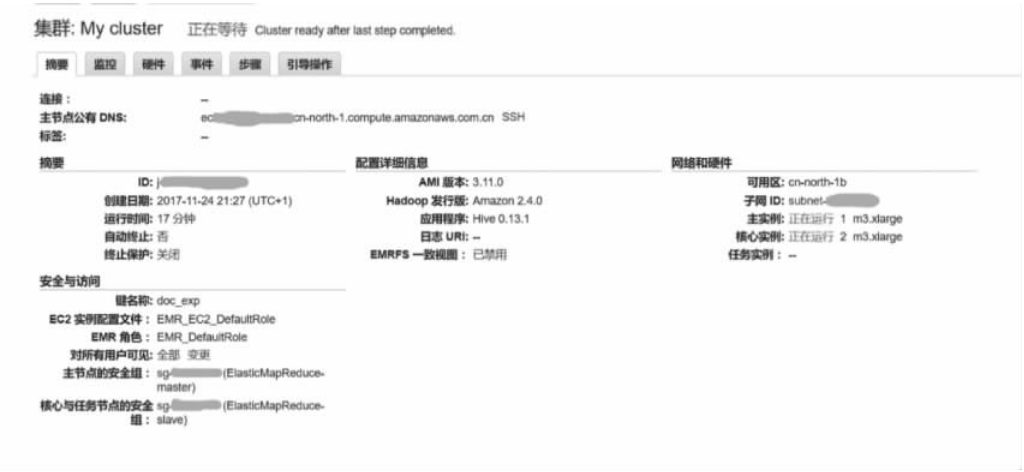

图 3-51　集群完成配置

步骤 3：连接到主节点

我们创建的集群默认是关闭 ssh 端口的，为了让我们的机器能够连上集群，需要将集群的 ssh 的 22 端口打开。在集群状态页面（见图 3-51）中，单击“安全与访问”下的“主节点的安全组”右边的节点名字，进入安全组的配置页面（见图 3-52）。

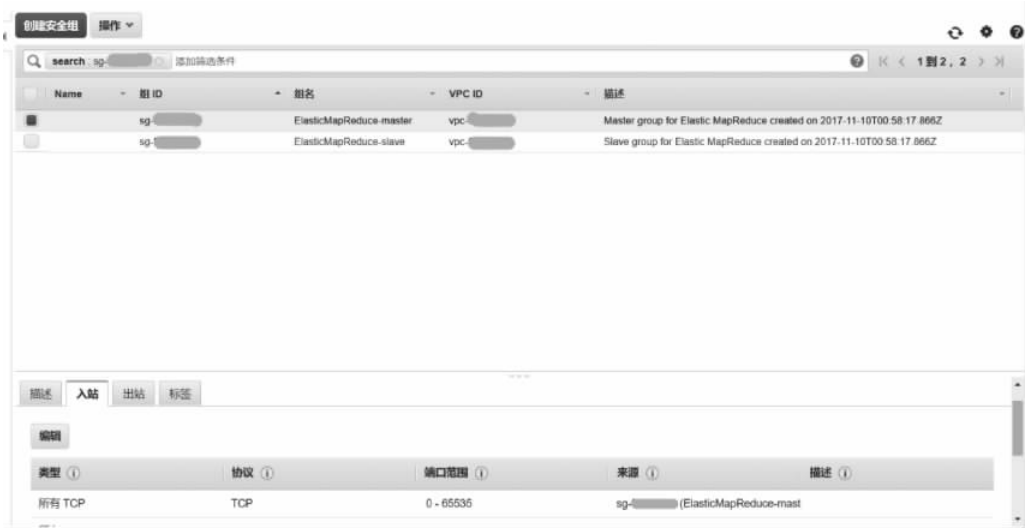

图 3-52　集群安全组配置

选择“描述”为“Master group for”的组（即主节点组），下方选择“入站”标签页，单击“编辑”，进入入站规则的配置页面（见图 3-53）。单击“添加规则”，在新添加的一行规则的“类型”选择“ssh”，“来源”选择“任何位置”，单击“保存”按钮以保存新的安全组规则。

完成入站规则的修改后，回到集群的状态页（见图 3-51），单击“主节点公有 DNS”后的地址右边的“SSH”，将会看到使用不同机器连接到集群的提示。Windows 可以按照图 3-54 所示进行连接。Mac 和 Linux 系统可以按图 3-55 所示进行连接。

以下我们以 Linux 系统为例。打开终端，输入以下命令，将～/key_dirt/doc_exp. pem 替换

图 3-53　编辑入站规则

图 3-54　Windows 连接提示

图 3-55　Mac/Linux 连接提示

为自己的私钥文件的位置和文件名，将 ec0-00-00-000-00. cn-north-1. compute. amazonaws. com. cn 替换为集群状态页面中“主节点公有 DNS”后的地址。

```
$ ssh -i ~/key_dirt/doc_exp.pem hadoop@ec0-00-00-000-00.cn-north-1.compute.amazonaws.com.cn
```

在弹出安全警告提示后输入“yes”，取消安全警告。连接成功后将会看到图 3-56 所示输出。

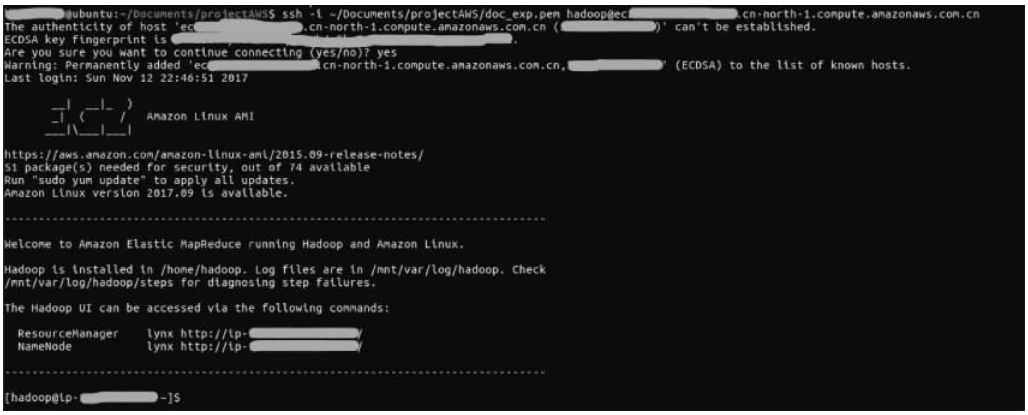

图 3-56　成功连接集群

步骤 4：启动和配置 Hive

Apache Hive 是一种数据仓库应用程序，使用类似 SQL 的语言来查询 Amazon EMR 集群数据。由于我们在配置集群时选择了 Hive，它已在主节点上准备就绪可供使用。

要以交互方式使用 Hive 查询 Web 服务器日志数据，需要加载一些额外的库。这些额外的库包含在主节点上名为 hive_contrib. jar 的 Java 存档文件中。当加载这些库后，Hive 会将这些库与它为处理之后的查询而启动的 map-reduce 作业绑定。

输入“hive”命令以启动 Hive。如果找不到 hive 命令，请确保在连接到主节点时指定了 hadoop 而不是 ec2-user 作为用户名。否则，关闭此连接并再次连接到主节点。

然后在 hive >命令提示符下，运行以下命令对 Hive 进行配置（见图 3-57）：

```
hive> add jar /home/hadoop/hive/lib/hive_contrib.jar;
```

如果找不到/home/hadoop/hive/lib/hive_contrib. jar，可能因为创建集群时所选择的 AMI 存在问题。可以根据第 2. 3. 7 节中的指示执行操作，然后使用不同的 AMI 版本重新开始实验。

图 3-57　打开与配置 Hive

步骤 5：上传数据文件

现在已经可以使用 Hive 导入需要分析的数据文件，也就是 Web 日志文件到 HDFS 上进行分析了。我们准备从 Amazon S3 中导入数据，但是现在还没有数据在 Amazon S3 中，因此需要先将数据上传到 Amazon S3 服务。

首先回到图 3.36 所示 AWS 控制台页面，单击“存储与内容分发”下的“S3”，进入 Amazon S3 管理页面（见图 3-58）。

图 3-58 Amazon S3 管理页面

单击“创建存储桶”，输入适合存储桶的名称，保持所有默认设置，然后单击“创建”，生成存储桶。然后单击进入存储桶，单击“上传”选择需要上传的文件进行上传。上传文件时对文件的设置保持默认即可。然后，就可以看见我们成功上传的文件在页面中显示（见图 3-59）。页面上方“Amazon S3 >”后的路径即为文件路径。

图 3-59 Amazon S3 存储桶内容

步骤 6：创建 Hive 表并向 HDFS 加载数据

要使 Hive 能够与数据进行交互，必须将数据从其现有格式（就 Apache Web 日志来说，数据为文本文件）转换为可表示为数据库表的格式。Hive 使用串行器/解串器（SerDe）执行此转换，存在适用于各种数据格式的 SerDe。有关如何编写自定义 SerDe 的信息，可参阅 Apache Hive Developer Guide。

我们在此实验中使用的 SerDe 采用正则表达式分析日志文件数据。SerDe 来自 Hive

开源社区。使用此 SerDe,可以将日志文件定义为表,在此教程的后面部分中,我们将使用类似 SQL 的语句查询该表。Hive 加载数据后,只要 Amazon EMR 集群处于运行状态,数据就会保留在 HDFS 存储中,即使关闭 Hive 会话和 SSH 连接也是如此。

复制下面的多行命令:

```
CREATE TABLE serde_regex(
  host STRING,
  identity STRING,
  user STRING,
  time STRING,
  request STRING,
  status STRING,
  size STRING,
  referer STRING,
  agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  "input.regex" = "([^]*) ([^]*) ([^]*) (-|\\[[^\\]]*\\]) ([^\"]*|\"[^\"]*\") (-
|[0-9]*) (-|[0-9]*)(?: ([^\"]*|\"[^\"]*\") ([^\"]*|\"[^\"]*\"))?",
  "output.format.string" = "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s"
)
  LOCATION 's3://elasticmapreduce/samples/pig-apache/input/';
```

其中,LOCATION 参数指定 Amazon S3 中一组示例 Apache 日志文件的位置。为了分析我们自己的 Apache Web 服务器日志文件,需要将此命令中的 URL 替换为 Amazon S3 中我们自己的日志文件的位置,可以选择我们在创建集群时自动生成的日志文件路径,或是我们上传的有关 Apache Web 的日志文件。在 AWS 控制台中进入“存储和内容分发”下的“S3”中找到对应的文件路径。例如,如图 3-59 所示,本案例的文件路径为

```
aws-logs-**********-cn-north-1/elasticmapreduce/webLogsSample
```

因此本案例输入:

```
LOCATION 's3:// aws-logs-**********-cn-north-1/elasticmapreduce/webLogsSample /'
```

在 hive 命令提示符下,粘贴该命令(在终端窗口中使用 Ctrl+Shift+V 组合键或在 PuTTY 窗口中右击),然后按 Enter 键。当命令完成时,我们将看到如图 3-60 所示提示。

图 3-60　创建 Hive 表

步骤 7：查询 Hive 和分析数据

可以使用 Hive 查询 Apache 日志文件数据。Hive 会将查询转换为 Hadoop MapReduce 作业并在 Amazon EMR 集群上运行该作业。当 Hadoop 作业运行时，将显示状态消息。Hive SQL 是 SQL 的一个子集，了解 SQL，就可以轻松创建 Hive 查询。以下是一些查询示例。

1）统计日志文件中的行数（见图 3-61）

```
select count(1) from serde_regex;
```

```
hive> select count(1) from serde_regex;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1511555355696_0001, Tracking URL = http://172.31.5.207:9046/proxy/application_1511555355696_0001/
Kill Command = /home/hadoop/bin/hadoop job  -kill job_1511555355696_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-11-24 20:35:49,774 Stage-1 map = 0%,  reduce = 0%
2017-11-24 20:36:01,205 Stage-1 map = 19%,  reduce = 0%, Cumulative CPU 7.49 sec
2017-11-24 20:36:04,306 Stage-1 map = 39%,  reduce = 0%, Cumulative CPU 10.84 sec
2017-11-24 20:36:07,410 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 14.26 sec
2017-11-24 20:36:15,704 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 15.88 sec
MapReduce Total cumulative CPU time: 15 seconds 880 msec
Ended Job = job_1511555355696_0001
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 15.88 sec   HDFS Read: 458 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 880 msec
OK
1948789
Time taken: 37.93 seconds, Fetched: 1 row(s)
```

图 3-61　统计日志文件中的行数

如上图所示，倒数第二行输出的是我们查询的结果，日志文件中的行数为 1 948 789 行。

2）返回五行日志文件数据中的所有字段（见图 3-62）

```
select * from serde_regex limit 5;
```

```
hive> select * from serde_regex limit 5;
OK
27.19.74.143    -       -       [30/May/2013:17:38:20 +0800]    "GET /static/image/common/faq.gif HTTP/1.1"     200     1127    NULL    NULL
110.52.250.126  -       -       [30/May/2013:17:38:20 +0800]    "GET /data/cache/style_1_widthauto.css?y7a HTTP/1.1"    200     1292    NULL    NULL
27.19.74.143    -       -       [30/May/2013:17:38:20 +0800]    "GET /static/image/common/hot_1.gif HTTP/1.1"   200     680     NULL    NULL
27.19.74.143    -       -       [30/May/2013:17:38:20 +0800]    "GET /static/image/common/hot_2.gif HTTP/1.1"   200     682     NULL    NULL
27.19.74.143    -       -       [30/May/2013:17:38:20 +0800]    "GET /static/image/filetype/common.gif HTTP/1.1"        200     90      NULL    NULL
Time taken: 0.037 seconds, Fetched: 5 row(s)
```

图 3-62　返回五行日志文件数据中的所有字段

3）统计来自 IP 地址为 59.46.212.74 的主机的请求数（见图 3-63）

```
select count(1) from serde_regex where host = "59.46.212.74";
```

如图 3-63 所示，来自该 IP 地址的主机的请求数为 5569。

步骤 8：清除集群

为了防止账户产生额外费用，可以按照以下步骤清除此实验创建的 AWS 资源。

1）断开与主节点的连接

（1）在您的终端窗口或 PuTTY 窗口中，请按 Ctrl＋C 键以退出 Hive。

```
hive> select count(1) from serde_regex where host="59.46.212.74";
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1511555355696_0003, Tracking URL = http://172.31.5.207:9046/proxy/application_1511555355696_0003/
Kill Command = /home/hadoop/bin/hadoop job  -kill job_1511555355696_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-11-24 20:42:09,676 Stage-1 map = 0%,  reduce = 0%
2017-11-24 20:42:24,092 Stage-1 map = 19%,  reduce = 0%, Cumulative CPU 11.01 sec
2017-11-24 20:42:27,205 Stage-1 map = 39%,  reduce = 0%, Cumulative CPU 14.07 sec
2017-11-24 20:42:28,235 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 15.21 sec
2017-11-24 20:42:34,420 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.92 sec
MapReduce Total cumulative CPU time: 16 seconds 920 msec
Ended Job = job_1511555355696_0003
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 16.92 sec   HDFS Read: 458 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 920 msec
OK
5569
Time taken: 33.818 seconds, Fetched: 1 row(s)
```

图 3-63　统计来自 IP 地址为 59.46.212.74 的主机的请求数

(2) 在 SSH 命令提示符下,运行 exit。

(3) 关闭终端窗口或 PuTTY 窗口。

2) 终止集群

(1) 打开 Amazon EMR 控制台。

(2) 单击 Cluster List (集群列表)。

(3) 选择集群名称,然后单击 Terminate (终止)按钮。当系统提示您确认时,单击“确认”按钮终止。

3.6　阿里云

于 2009 年成立,总部位于中国杭州的阿里云,已经成长为中国最大的云服务提供商。阿里云独立开发出一套完整的云计算平台——飞天平台,并初步形成了一套比较完整的开放服务,如弹性计算服务(Elastic Compute Service,ECS)、开放存储服务(Open Storage Service,OSS)、开放结构化数据服务(Open Table Service,OTS)、开放数据处理服务(Open Data Processing Service,ODPS)、关系型数据库服务(ApsaraDB for RDS,RDS)等。

3.6.1　飞天开放平台架构

阿里云计算有限公司(简称“阿里云”)成立于 2009 年 9 月 10 日,致力于打造云计算的基础服务平台,注重为中小企业提供大规模、低成本、高可靠的云计算应用及服务。飞天是由阿里云开发的一个大规模分布式计算系统,其中包括飞天内核和飞天开放服务。飞天内核负责管理数据中心 Linux 集群的物理资源,控制分布式程序运行,隐藏下层故障恢复和数据冗余等细节,有效提供弹性计算和负载均衡。如图 3-64 所示,飞天体系架构主要包含四大部分:1)资源管理、安全、远程过程调用等构建分布式系统常用的底层服务;2)分布式文件系统;3)任务调度;4)集群部署和监控。飞天开放服务为用户应用程序提供了计算和存储两方面的接口和服务,包括弹性计算服务(Elastic Compute Service,ECS)、开放存储服务(Open Storage Service,OSS)、开放结构化数据服务(Open Table Service,OTS)、关系型数

据库服务(Relational Database Service,RDS)和开放数据处理服务(Open Data Processing Service,ODPS),并基于弹性计算服务提供了云服务引擎(AliCloud Engine,ACE)作为第三方应用开发和 Web 应用运行和托管的平台。

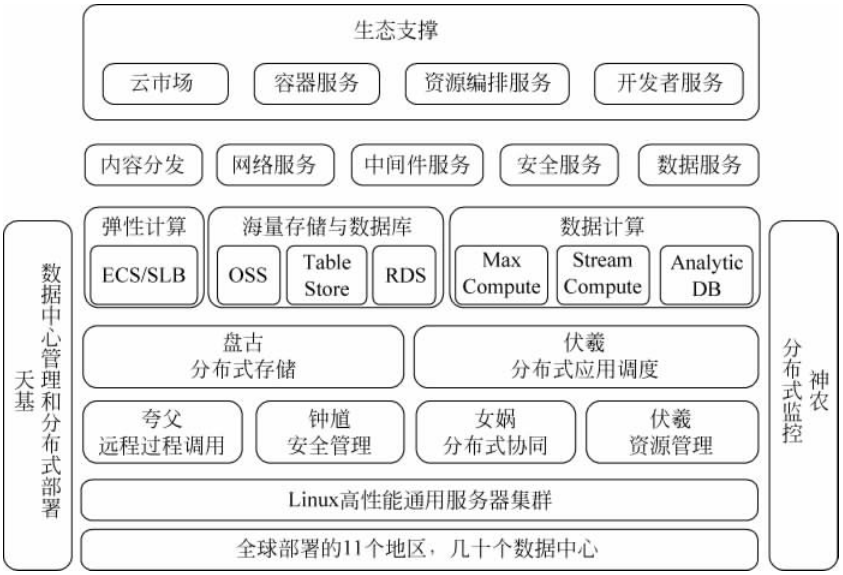

图 3-64 飞天开放平台架构图

按照四层体系结构来看,飞天平台的最底层是全球部署的 11 个地区和几十个数据中心,这些数据中心里是安装了 Linux 操作系统的通用高端服务器。如飞天开放平台架构图所示,橙色组件组成了大规模通用计算平台,最底下四个橙色块(夸父远程过程调用、安全管理、女娲分布式协同和伏羲资源管理)代表构建分布式系统最基本的组件。盘古分布式存储,简单来说,就是把所有集群中的硬盘组织成一个单个的文件系统。同时,两侧分别是天基的数据中心管理、分布式部署,以及神农分布式监控。整个系统架构里面部署和监控也是核心系统的一部分,能实现 7×24h 不间断的部署和监控,秒级监控所有指标判断是否有问题并且实时修复。

中间蓝色一层是核心的资源型服务组件,大概分为三类。一是弹性计算,简单理解就是将物理机切分成虚拟服务器的概念。二是海量存储的数据库,其中 OSS 是存储无结构的数据如视频、照片、音乐之类的,Table Store 可以认为是半结构化存储,RDS 则是关系型数据库服务。三是数据计算,它则分为多维度准实时数据的查询服务、实时流计算处理服务和大规模批量计算服务。

在核心的资源型服务组件上面还有一些端到端、基于云的应用所需要的核心服务,例如内容分发 CDN、网络服务、安全服务、数据服务等。网络服务,包括 VPC、域名服务和 VPN。中间件服务,包括消息队列、工作流等。数据服务,则包括如人工智能、语音识别、翻译、图像识别之类。

最上层则是生态支撑,容器服务可以支持那些基于容器的微服务架构,或者是编排服务帮助开发者在云上开展资源的编排。还有云市场,可以认为是云上的 AppStore,开发者可以把他们的应用注册在云市场里,使用者可直接注册使用。还有开发者服务,开发者很容易

监控诊断他们的应用并且发现问题和调试。

在飞天体系结构中，最关键技术是：分布式系统底层服务、分布式文件系统、任务调度和集群监控和部署。其中，分布式系统底层服务：主要提供分布式环境下所需要的协调服务（女娲）、远程过程调用（夸父），以及提供系统安全的钟馗模块。分布式文件系统：主要提供一个海量的、可靠的、可扩展的数据存储服务，将集群中各个节点的存储能力聚集起来，并能够自动屏蔽软硬件故障，为用户提供不间断的数据访问服务。任务调度：为集群系统中的任务提供调度服务，同时支持强调响应速度的在线服务和强调处理数据吞吐量的离线任务。集群监控和部署：对集群的状态和事件进行监控，对异常事件产生警报和记录；为运维人员提供整个飞天系统以及上层应用的部署和配置管理，支持在线集群扩容和应用服务的在线升级。接下来详细介绍这5个技术。

1. 分布式基础架构

命名服务——女娲。女娲系统为飞天平台提供高可用的协调服务，是整个飞天系统的核心服务，它的作用类似于文件系统的树形命名空间使分布式进程互相协同工作。女娲系统与Google的chubby和Hadoop的zookeeper系统的功能与实现相似。

远程过程调用——夸父。夸父是飞天平台中负责网络通信的组件，它提供了一个远程过程调用的接口，简化编写基于网络的分布式应用。其中，异步调用时，不等接收结果便会立即返回，用户必须通过显式调用接收函数取得请求结果；同步调用时则会等待，直到接收到结果才返回。在实现中，同步调用是通过封装异步调用来实现的。

安全管理——钟馗。飞天操作系统中安全管理的机制提供了以用户为单位的身份认证和授权，以及对集群数据资源和服务进行的访问控制。

2. 分布式文件系统

飞天操作系统中数据存储是由分布式文件系统完成的。盘古与Google文件系统和Hadoop的HDFS的设计目标有一致的部分，都是将大量廉价机器的存储资源聚合在一起，为用户提供大规模、高可靠、高吞吐量、高可用和可扩展的存储服务，是集群操作系统中的重要组成部分。盘古还能很好地支持在线应用的低延时需求，这是Google文件系统和Hadoop的HDFS所不具备的。

3. 任务调度

如图3-65所示，伏羲是飞天平台的调度系统，同时也为应用开发提供了一套编程基础框架。与盘古一样，伏羲也必须在一个系统架构下才能同时支持强调响应速度的在线服务和强调处理数据吞吐量的离线任务。关于在线服务调度，在飞天平台上，每个具体的service都有一个service master和多个不同角色的service worker，它们一起协同工作完成整个服务功能。关于离线任务调度，在飞天平台上，一个离线任务的执行过程被抽象为一个有向无环图：图上每个顶点对应一个task，每条边对应一个pipeline。一个连接的两个task的pipeline表示前一个task的输出是后一个task的输入。

4. 集群监控——神农

神农是飞天平台上负责信息收集、监控和诊断的系统。它通过在每台物理机器上部署轻量级的信息采集模块，获取各个机器的操作系统与应用软件运行状态，监控集群中的故障，并通过分析引擎对整个飞天系统的运行状态进行评估。神农系统包括Master、

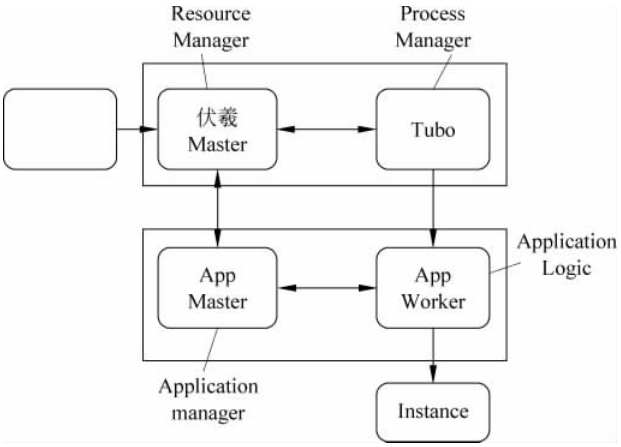

图 3-65 伏羲体系结构图

Inspector 和 Agent 三部分。

Master：负责管理所有神农 Agent，并对外提供统一的接口处理神农用户的订阅(Subscription)请求，在集群中只要一个 Master。

Inspector：部署在每一冷机器上的进程，负责采集当前机器和进程的通用信息，并在 If 时发送给该机器上的神农 Agent。

Agent：部署在每台物理机器的后台(Daemon)程序，负责接收来自应用的 Inspector 写入的信息。

5. 飞天的技术特色

同一个平台同时支持离线、在线服务，如阿里巴巴集团子公司神马搜索就是建在飞天上，他们会进行千亿级别网页的离线处理，索引所有网页，大概每一两个月把整个索引翻一遍。此外，拥有多网页的同时同样拥有整个网页之间关联的连接图，也是千亿级别的节点，并且有百亿级别的索引可以在线查询。在线方面，基于飞天平台的邮箱服务每天处理亿量级的邮件，日发送邮件达到千万量级，所有发送和接收在 10ms 级完成。

飞天单集群达到了万台规模、百 PB 级别存储、10 万级别的 CPU 合数；整个架构设计里面没有单点，确保了整个系统可用性达到 99.95%；飞天应用设有默认等级，通过多副本冗余算法，数据可靠性极强；完全分布式部署、监控和诊断。

3.6.2 开放云计算服务 ECS

弹性计算服务(ECS)基于飞天大规模分布式计算系统，以虚拟化方式将一台物理机分成多台云服务器，向广大互联网站长和开发者提供可伸缩的计算资源，其体系结构如图 3-66 所示。

在数据中心中，大量的计算节点和存储节点通过飞天分布式计算系统将物理资源整合为一个整体，上层通过 XEN 虚拟化技术，对外提供弹性计算服务。ECS 包含两个重要模块：计算资源模块和存储资源模块。ECS 的计算资源指 CPU、内存、带宽等资源，主要通过将物理服务器(宿主机)上的计算资源虚拟化，然后再分配给云服务器使用。云服务器的存

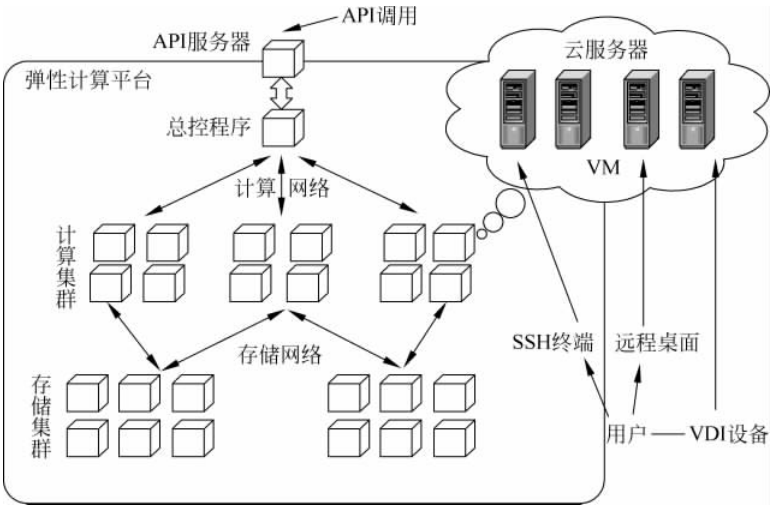

图 3-66　ECS 的体系结构图

储资源采用了飞天的大规模分布式文件系统，将整个集群中的存储资源虚拟化后对外提供服务。用户数据在集群中存储多个副本，任意一份副本损坏后系统都可以自动恢复到多个副本，使用户数据达到 99.999%的可靠性。用户可以使用 SSH(ECS 为 Linux 系统)或者远程桌面(ECS 为 Windows 系统)直接远程登录并管理云服务器。

弹性服务可伸缩架构的优点：

(1) 自助管理：云服务器的配置、管理、升级、监控工作通过 API 和网站页面实现自助操作。大大降低了维护成本，提高运维响应速度。

(2) 数据安全性：云服务器的磁盘数据存放在云存储空间中，数据自动实现分布式存储，“永不丢失”。

(3) 故障恢复：当宿主物理机发生故障时，平台能够自动迁移云服务器，并且将其数据恢复到最后一刻的状态。

(4) 便捷的快照与回滚：云服务器的磁盘支持快照功能。可以根据需要，将磁盘数据快速回滚到之前的任一快照版本。

(5) 安全域：将一个数据中心分成若干个安全域，每个安全域可独立设置防火墙策略。

(6) 防 DDOS 攻击：系统自动检测 DDOS 攻击特征，根据攻击规模和策略设置进行流量清洗或者黑洞处理。

(7) 在多台服务器间分配请求流量，负载均衡器自身实现了自适应扩展。

3.6.3　开放存储服务 OSS 和 CDN

开放存储服务(Open Storage Service，OSS)是阿里云对外提供的海量、安全、低成本和高可靠的云存储服务，如图 3-67 所示，OSS 概念图如图 3-68 所示。开放存储服务为广大站长、开发者，及大容量存储需求的企业或个人，提供海量、安全、低成本、高可靠性的云存储服务。通过简单的 REST 接口，存放网站或应用中的图片、音频、视频、附件等较大文件。当用户面对大量静态文件(如图片、视频等)访问请求和数据存储时，使用 OSS 可以彻底解决

存储的问题，并且极大地减轻原服务器的带宽负载。使用 CDN 可以进一步加快网络应用内容传递到用户端的速度。

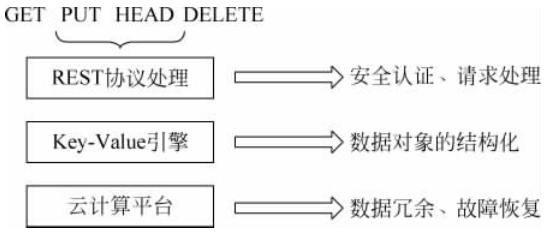

图 3-67 云存储服务的架构图

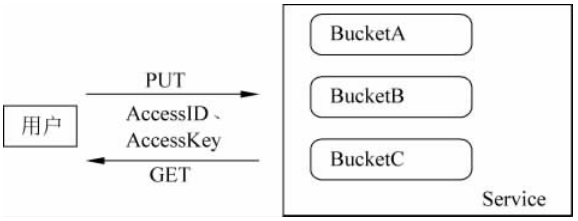

图 3-68 OSS 概念图

Service：对于某用户来说，就是 OSS 提供给该用户的虚拟存储空间。用户可以在这个存储空间中拥有一个或者多个 Bucket。

Access ID & Access Key（API 密钥）：用户注册 OSS 时，系统会给用户分配一对 Access ID 和 Access Key，称为 ID 对，用于标识用户，为访问 OSS 做签名验证。当用户通过 oss.aliyun.com 成功创建 OSS 存储服务后，可以进入“管理中心”→“获取 API 密钥”得到 Access ID & Access Key。为了更深入理解 OSS，下面给出一个授权访问控制的应用场景，如图 3-69 所示。

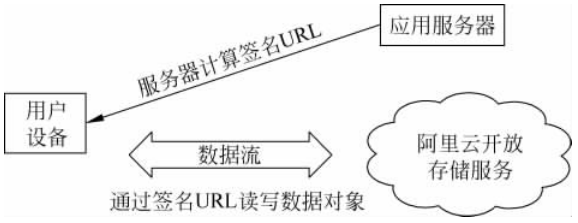

图 3-69 授权访问控制图

Step1：服务器计算签名 URL

```
url = oss.sign_url("GET","bucket1","oss.jpg",60s);
```

Step2：用户通过签名 URL 读数据对象

```
< img
src = "http://storage.aliyun.com/bucket1/oss.jpg?OSSAc
cessKevId = v6h7nbcothehvcp7jlnwmrw9&Expires = 1334
```

```
571637&Signature = MwECjsQNnTB5RdNmKY5HdU37H
mM % 3D"
/>
```

用户在 URL 中加入签名信息，在 URL 中实现签名，至少包含 Signature、Expires、OSSAccessKeyId 三个参数。Expires 这个参数的值是一个 UNIX 时间，用于标识该 URL 的超过时间。如果 OSS 接收到这个 URL 请求的时候晚于签名中包含的 Expires 参数时，则返回请求超时的错误码。

3.6.4　开放结构化数据服务 OTS

开放结构化数据服务(Open Table Service，OTS)适合存储海量的结构化数据，并且提供了高性能的访问速度。当数据量猛增时，传统的关系型数据需要资深的 DBA 才能搞定；而使用 OTS，数据再怎么增长，它都自动默默搞定所有事情。高效可扩展的 NoSQL 服务面向海量结构化和半结构化数据的存储；实时读写操作满足强一致性；支持强数据类型；通过 REST API 提供服务，提供多种语言 SDK；提供用户级别的数据隔离、访问控制、权限管理。

OTS 服务的系统架构分为四层，最上层是应用程序，应用通过调用各种语言的 SDK 与 OTS 服务进行交互；第二层是用户服务层，这一层完成的功能是对应用发送的请求进行协议处理、身份权限的校验、资源计量和请求到后端存储引擎节点的路由；第三层是存储引擎层，负责表分区的扩展和管理、负载均衡、存储数据和索引的管理、故障的处理以及高可用容灾等方面；最下面一层是飞天操作系统，负责管理底层的硬件资源，向上提供统一的分布式存储(盘古)和计算(伏羲)，OTS 架构图如图 3-70 所示。其中下面三层运行在阿里云数据中心的物理集群上，对应用程序透明，最上面一层是用户的程序，通常运行在阿里云的 ECS 服务器以获得更好的访问 OTS 的性能，当然也可以运行在用户自己的物理服务器或者移动设备上(我们目前正在开发移动端的 OTS SDK，包括 Android 和 iOS)。

图 3-70　OTS 架构图

下面提供一个 OTS 的应用实例：邮箱应用的元数据存储，如图 3-71 所示。

相应代码如下：

CreateTable：

User Table

用户ID	用户名	已用容量	总容量
U0001	tom@aliyun.com	45000	100000
U0002	eric@aliyun.com	100000	1000000

用户ID	接收时间	发件人	邮件尺寸	主题	是否已读
U0001	1998-11-1	eric@aliyun.com	10000	Hello	Y
U0001	2011-10-21	alice@aliyun.com	20000	Lunch together	N
U0001	2011-10-24	eric@aliyun.com	15000	Re: Hello	Y
U0002	1999-12-25	tom@aliyun.com	21000	Meeting	Y

⇩ View: 调整PK列顺序，改变排序规则

用户ID	发件人	接收时间	邮件尺寸
U0001	alice@aliyun.com	2011-10-21	20000
U0001	eric@aliyun.com	1998-11-1	10000
U0001	eric@aliyun.com	2011-10-24	15000
U0002	tom@aliyun.com	1999-12-25	21000

图 3-71　OTS 实例图

```
from ots import *
ots_conn = OTSConnection('http://service.ots.aliyun.com:80',      #服务器域名
                         '8vy14gd2blhb87o88bslslfa',              #AccessID
                         'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX=')     #AccessKey
table_info = {}
table_info['TableName'] = 'OTSTestTable'                          #设置表名
table_info['PrimaryKey'] = [{'Name':'ID','Type':'STRING'}]        #设置表的主键
table_info['PagingKeyLen'] = 0                                    #设置分页键

table_meta = TableMeta(table_info,[])                             #构造 TableMeta 对象

ots_conn.create_table(table_meta)                                 #OTS 创建表
```

PutData：

```
insert_pk = [{'Name':'ID','Value':'0001'}]              #设置要插入行的主键
columns = [{'Name':'UserName','Value':'Alice'},
        {'Name':'Gender','Value':'Famale'}]             #设置要插入的行和列

ots_conn.put_data('OSTTestTable',insert_pk,columns)   #使 OTS 在 OTSTestTable 表中写入一行
```

GetRowsByRange：

```
query_range = {'Name':'ID','Range':{'Begin':'0','End':'z'}}          #设置要查询的范围

rows = ots_conn.get_rows_by_range('OTSTestTable',[],                 #让 OTS 读取 OTSTable 表
    query_range,[],100)              #读取指定的范围,并读取所有的列,结果集中返回前 100 条
```

OTS 具有的优势：①基于飞天大规模分布式计算系统，数据有多个备份；②单表的数据规模达 TB 级，每秒万次并发访问；③读写访问的平均时延在 ms 级；④表结构无须固

定；⑤全托管的服务，零运维，可随时随地访问。

3.6.5　关系型数据库(RDS)

关系型数据库是一个基于高稳定、大规模平台的商用关系型数据库服务。其帮助个人与企业用户解决费时、费力的数据库管理，节约硬件成本和维护成本。与现有商用 MySQL 和 MS SQL Server 完全兼容。

基本功能主要包括日常操作：创建、报警、监控；备份恢复：根据用户自定义备份策略进行自动备份，同时对备份数据可进行查看、还原、下载等操作；弹性升级：例如与聚石塔合作，聚石塔是支撑天猫的一个重要服务，在双十一时，可以根据服务的压力，动态调整配置和资源；安全设置：RDS 设有白名单，同时还具有防 DDOS 攻击、数据清洗、密码暴力攻击检测的功能；慢 SQL 查询：系统会记录下执行时间超过 1s SQL 语句；故障恢复：采用双机热备方式，在遇到故障时 30s 内完成切换，用户只需要在程序设置好自动重联即可；在线迁移：主要只是将某个实例在线迁移到物理机上，迁移过程中对业务不会造成影响。

RDS 的几点不同：除价格外，RDS 采用多实例，故障恢复在 30s 内自动完成；在性能上，可以最高达到 12000IOPS，安全，水平拆分提供分布式 RDS，便于用户做水平拆分和水平扩展。

3.6.6　开放数据处理服务(ODPS)

开放数据处理服务(Open Data Process Service，ODPS)的诞生是为了深度挖掘出海量数据(如 HTTP Log)中蕴藏的价值。2014 年 7 月 1 日 MaxCompute(原 ODPS 服务)正式对外开放，这也标志着阿里巴巴成为世界上第一家对外公开提供 5K 处理能力的公司。大数据计算服务(MaxCompute)是一种快速、完全托管的 PB/EB 级数据仓库解决方案，MaxCompute 具备万台服务器扩展能力和跨地域容灾能力，是阿里巴巴内部核心大数据平台，承担了集团内部绝大多数的计算任务，支撑每日百万级作业规模。MaxCompute 基本的体系结构如图 3-72 所示，最底层就是在物理机器之上打造的提供统一存储的盘古分布式文件存储系统；在盘古之上一层就是伏羲分布式调度系统，这一层将包括 CPU、内存、网络以及磁盘等在内的所有计算资源管理起来；再上一层就是统一的执行引擎也就是 MaxCompute 执行引擎；而在执行引擎之上会打造各种各样的运算模式，比如流计算、图计算、离线处理、内存计算以及机器学习等；在这之上还会有一层相关的编程语言，也就是 MaxCompute 语言；在语言上面希望为各应用方能够提供一个很好的平台，让数据工程师能够通过平台开发相关的应用，并使得应用能够快速地在分布式场景里面得到部署运行。

ODPS 以 REST API 的形式，支持用户提交类 SQL 的查询语言，对海量数据进行处理。在 API 之上，还提供 SDK 开发包和命令行工具。与传统的数据仓库工具相比，ODPS 的处理能力强大、成本低廉、伸缩灵活。MaxCompute 包括如下功能：

(1) 批量、历史数据通道

Tunnel 是 MaxCompute 向用户提供的数据传输服务。该服务水平可扩展，支持每天 TB/PB 级别的数据导入导出。特别适合于全量数据或历史数据的批量导入。Tunnel 提供了 Java SDK，且在 MaxCompute 的客户端工具中，有对应的命令实现本地文件与服务数据

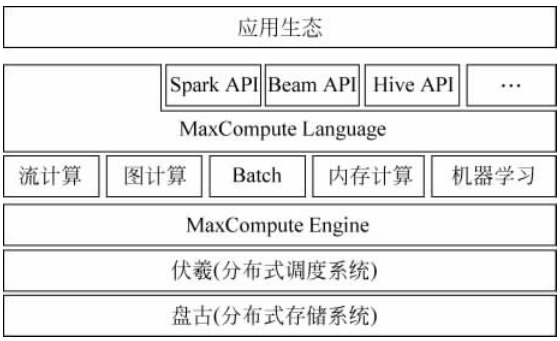

图 3-72 MaxCompute 架构图

的互通。

(2) 实时、增量数据通道

针对实时数据上传的场景,MaxCompute 提供了另一套名为 DataHub 的服务。该服务具有延迟低、使用方便的特点,特别适用于增量数据的导入。Datahub 还支持多种数据传输插件,例如 Logstash、Flume、Fluentd、Sqoop 等。同时支持日志服务 Log Service 中的日志数据的一键投递至 MaxCompute,进而利用大数据开发套件进行日志分析和挖掘。

(3) SQL

MaxCompute SQL 采用标准的 SQL 语法,兼容部分 Hive 语法。在语法上和 HQL 非常接近,熟悉 SQL 或 HQL 的编程人员都容易上手。另外,MaxCompute 提供更高效的计算框架支持 SQL 计算模型,执行效率比普通的 MapReduce 模型更高。需要注意的是,MaxCompute SQL 不支持事务、索引及 Update/Delete 等操作。

(4) MapReduce

MaxCompute 提供 Java MapReduce 编程模型。值得注意的是,由于 MaxCompute 并没有开放文件接口,用户只能通过它所提供的 Table 读写数据,因此 MaxCompute 的 MapReduce 模型与开源社区中通用的 MapReduce 模型在使用上有一定的区别。我们相信,这样的改动虽然失去一定的灵活性,例如不能够自定义排序及哈希算法,但却能够简化开发流程,免除很多琐碎的工作。更为重要的是,MaxCompute 还提供了基于 MapReduce 的扩展计算模型,即 MR2。在该模型下,一个 Map 函数后,可以接入连续多个 Reduce 函数。

(5) Graph

对于某些复杂的迭代计算场景,如 K-Means、PageRank 等,如果仍然使用 MapReduce 完成这些计算任务将是非常耗时的。MaxCompute 提供的 Graph 模型能够非常好地完成这一类计算任务。

(6) 安全

MaxCompute 是一个多租户的计算平台。默认情况下,各租户间数据不共享,彼此隔离,但用户可以通过 MaxCompute 提供的授权机制将数据共享给项目组其他人。MaxCompute 以二维表格式存储数据,所有数据均以表格式存储,不暴露文件系统。并采用列压缩存储格式,极高的数据压缩比可极大节省用户成本。通常情况下,MaxCompute 存储具备 5 倍压缩的能力。

ODPS提供了SQL与MapReduce两种API供用户开发调用。ODPS SQL采用类似SQL的语法处理大规模(PB级别)数据,适合于处理强调数据吞吐量的离线任务。ODPS SQL提供了大量操作海量数据的SQL语法支持(API),例如,创建、删除表和视图的DDL语法,更新表的DML语法等。为了方便用户完成数据处理的各类任务,ODPS SQL还提供了很多高级功能,例如,窗口函数、用户自定义函数、存储过程等。与数据库相比,ODPS SQL并不具备数据库的一些特征,包括事务和主键约束。ODPS SQL的优势在于能够快速处理海量数据,它能够将多个SQL语句以它们之间的数据依赖关系组成一个工作流,然后以执行工作流的方式完成复杂的数据分析功能。

阿里云的海量数据存储的数据库中,OTS、ODPS、RDS三者的区别如下:

1) OTS服务的特点是大规模、低延时、强一致,其适用场景是对数据规模和实时性要求高的应用。

2) ODPS重点面向数据量大(TB级别)且实时性要求不高的OLAP(Online Analytical Processing),适用于构建数据仓库、海量数据统计、数据挖掘、数据商业智能等应用。

3) OTS和ODPS可以配合使用,前者支持大规模并发的日常访问(例如铁路售票前台系统),然后每隔24小时就把交易数据推入ODPS支撑的数据仓库,利用后者进行进一步的业务分析。

4) 最后,RDS适合较小数据规模的常规OLTP(online transactional processing)应用。如果用户的需求是把所有关系数据库服务(例如MySQL和SQL Server)迁移到云平台上,主要重视兼容性,可以选择RDS。

习题

1. 简述云计算的定义。
2. 简述云计算的体系结构。
3. 简述ACID理论、BASE理论与CAP理论。
4. 简述云计算平台的存储结构。
5. 简述何为分布式文件系统。
6. 简述MapReduce计算模型的原理。
7. 简述一致性哈希算法与Dynamo环的原理。
8. 畅谈云计算在未来的应用。