
获取教材和讲义 PPT 等各种课程资料请访问 http://dblab.xmu.edu.cn/node/422

=课程教材由林子雨老师根据网络资料编著=

**厦门大学计算机科学系教师 林子雨 编著**

http://www.cs.xmu.edu.cn/linziyu

2013 年 9 月

# 前言

本教程由厦门大学计算机科学系教师林子雨编著，可以作为计算机专业研究生课程《大数据技术基础》的辅助教材。

本教程的主要内容包括：大数据概述、大数据处理模型、大数据关键技术、大数据时代面临的新挑战、NoSQL 数据库、云数据库、Google Spanner、Hadoop、HDFS、HBase、MapReduce、Zookeeper、流计算、图计算和 Google Dremel 等。

本教程是林子雨通过大量阅读、收集、整理各种资料后精心制作的学习材料，与广大数据库爱好者共享。教程中的内容大部分来自网络资料和书籍，一部分是自己撰写。对于自写内容，林子雨老师拥有著作权。

本教程 PDF 文档及其全套教学 PPT 可以通过网络免费下载和使用（下载地址：http://dblab.xmu.edu.cn/node/422）。教程中可能存在一些问题，欢迎读者提出宝贵意见和建议！

本教程已经应用于厦门大学计算机科学系研究生课程《大数据技术基础》，欢迎访问 2013 班级网站 http://dblab.xmu.edu.cn/node/423。

林子雨的 E-mail 是：ziyulin@xmu.edu.cn。

林子雨的个人主页是：http://www.cs.xmu.edu.cn/linziyu。

林子雨于厦门大学海韵园

2013 年 9 月

# 第 3 章 Hadoop

**厦门大学计算机科学系教师 林子雨 编著**

个人主页：http://www.cs.xmu.edu.cn/linziyu

课程网址：http://dblab.xmu.edu.cn/node/422

2013 年 9 月

# 第 3 章 Hadoop

Hadoop 是一个开源的、可运行于大规模集群上的分布式并行编程框架，它实现了 Map/Reduce 计算模型。借助于 Hadoop，程序员可以轻松地编写分布式并行程序，将其运行于计算机集群上，完成海量数据的计算。

本章介绍 Hadoop 相关知识，内容要点如下：

- Hadoop 概述
- Hadoop 发展简史
- Hadoop 的功能与作用
- 为什么不用关系型数据库管理系统
- Hadoop 的优点
- Hadoop 的应用现状和发展趋势
- Hadoop 项目及其结构
- Hadoop 的体系结构
- Hadoop 与分布式开发
- Hadoop 应用案例
- Hadoop 平台上的海量数据排序

## 3.1 Hadoop 概述

Hadoop 是 Apache 软件基金会旗下的一个开源分布式计算平台。以 Hadoop 分布式文件系统（HDFS，Hadoop Distributed File System）和 MapReduce（Google MapReduce 的开源实现）为核心的 Hadoop，为用户提供了系统底层细节透明的分布式基础架构。HDFS 的高容错性、高伸缩性等优点允许用户将 Hadoop 部署在低廉的硬件上，形成分布式系统；MapReduce 分布式编程模型允许用户在不了解分布式系统底层细节的情况下开发并行应用程序。所以，用户可以利用 Hadoop 轻松地组织计算机资源，从而搭建自己的分布式计算平台，并且可以充分利用集群的计算和存储能力，完成海量数据的处理（如图 3-1 所示）。

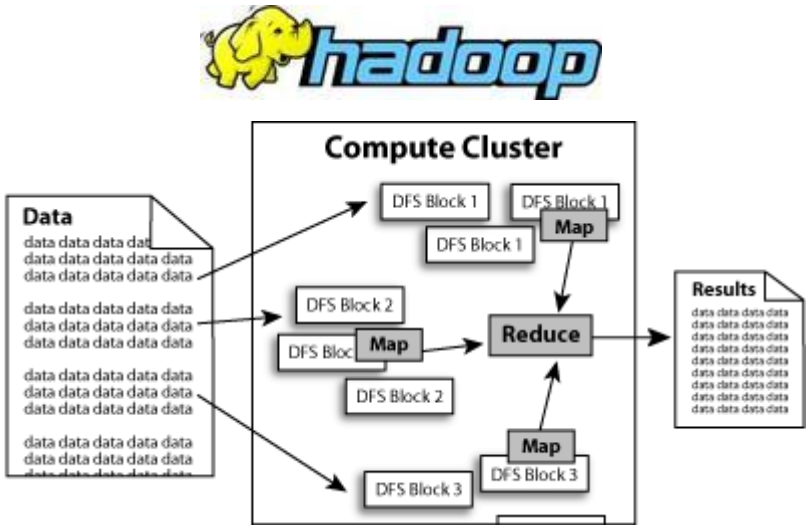


图 3-1 Hadoop 计算过程示意图

Hadoop 被公认是一套行业大数据标准开源软件，在分布式环境下提供了海量数据的处理能力。几乎所有主流厂商都围绕 Hadoop 开发工具、开源软件、商业化工具和技术服务。2013 年，大型 IT 公司，如 EMC、Microsoft、Intel、Teradata、Cisco 都明显增加了 Hadoop 方面的投入，Teradata 还公开展示了一个一体机；另一方面，创业型 Hadoop 公司层出不穷，如 Sqrrl、Wandisco、GridGain、InMobi 等等，都推出了开源的或者商用的软件。

## 3.2Hadoop 发展简史

Hadoop 是 Doug Cutting——Apache Lucene 创始人——开发的使用广泛的文本搜索库。Hadoop 起源于 Apache Nutch，后者是一个开源的网络搜索引擎，本身也是 Lucene 项目的一部分。

Hadoop 这个名字不是一个缩写，它是一个虚构的名字。该项目的创建者，Doug Cutting 如此解释 Hadoop 的得名：“这个名字是我孩子给一头吃饱了的棕黄色大象命名的。我的命名标准就是简短，容易发音和拼写，没有太多的意义，并且不会被用于别处。小孩子是这方面的高手。Google 就是由小孩命名的”。

Hadoop 及其子项目和后继模块所使用的名字往往也与其功能不相关，经常用一头大象和其他动物主题（例如：Pig）。较小的各个组成部分给与更多描述性（因此也更俗）的名称。这是一个很好的原则，因为它意味着可以大致从其名字猜测其功能，例如，jobtracker 的任务就是跟踪 MapReduce 作业。

从头开始构建一个网络搜索引擎是一个雄心勃勃的目标，不只是要编写一个复杂的、能

够抓取和索引网站的软件，还需要面临着没有专业运行团队支持运行它的挑战，因为它有那么多独立部件。同样昂贵的还有：据 Mike Cafarella 和 Doug Cutting 估计，一个支持此 10 亿页的索引需要价值约 50 万美元的硬件投入，每月运行费用还需要 3 万美元。不过，他们相信这是一个有价值的目标，因为这会开放并最终使搜索引擎算法普及化。

Nutch 项目开始于 2002 年，一个可工作的抓取工具和搜索系统很快浮出水面。但他们意识到，他们的架构将无法扩展到拥有数十亿网页的网络。在 2003 年发表的一篇描述 Google 分布式文件系统（Google File System，简称 GFS）的论文为他们提供了及时的帮助，文中称 Google 正在使用此文件系统。GFS 或类似的东西，可以解决他们在网络抓取和索引过程中产生的大量的文件存储需求。具体而言，GFS 会省掉管理所花的时间，如管理存储节点。在 2004 年，他们开始写一个开放源码的应用，即 Nutch 的分布式文件系统（NDFS）。

2004 年，Google 发表了论文，向全世界介绍了 MapReduce。2005 年初，Nutch 的开发者在 Nutch 上有了一个可工作的 MapReduce 应用，到当年年中，所有主要的 Nutch 算法被移植到使用 MapReduce 和 NDFS 来运行。

Nutch 中的 NDFS 和 MapReduce 实现的应用远不只是搜索领域，在 2006 年 2 月，他们从 Nutch 转移出来成为一个独立的 Lucene 子项目，成为 Hadoop。大约在同一时间，Doug Cutting 加入雅虎，Yahoo 提供一个专门的团队和资源将 Hadoop 发展成一个可在网络上运行的系统。在 2008 年 2 月，雅虎宣布其搜索引擎产品部署在一个拥有 1 万个内核的 Hadoop 集群上。

2008 年 1 月，Hadoop 已成为 Apache 顶级项目，证明它是成功的，是一个多样化、活跃的社区。通过这次机会，Hadoop 成功地被雅虎之外的很多公司应用，如 Last.fm、Facebook 和《纽约时报》。

有一个良好的宣传范例，《纽约时报》使用亚马逊的 EC2 云计算将 4TB 的报纸扫描文档压缩，转换为用于 Web 的 PDF 文件。这个过程历时不到 24 小时，使用 100 台机器运行，如果不结合亚马逊的按小时付费的模式（即允许《纽约时报》在很短的一段时间内访问大量机器）和 Hadoop 易于使用的并行程序设计模型，该项目很可能不会这么快开始启动。

2008 年 4 月，Hadoop 打破世界纪录，成为最快排序 1TB 数据的系统。运行在一个 910 节点的群集，Hadoop 在 209 秒内排序了 1TB 的数据（还不到三分半钟），击败了前一年的 297 秒冠军。同年 11 月，谷歌在报告中声称，它的 MapReduce 实现执行 1TB 数据的排序只用了 68 秒。在 2009 年 5 月，有报道宣称 Yahoo 的团队使用 Hadoop 对 1TB 的数据进行排序只花了 62 秒时间。

**Hadoop 大事记**

- 2004 年——最初的版本（现在称为 HDFS 和 MapReduce）由 Doug Cutting 和 Mike Cafarella 开始实施。
- 2005 年 12 月——Nutch 移植到新的框架，Hadoop 在 20 个节点上稳定运行。
- 2006 年 1 月——Doug Cutting 加入雅虎。
- 2006 年 2 月——Apache Hadoop 项目正式启动以支持 MapReduce 和 HDFS 的独立发展。
- 2006 年 2 月——雅虎的网络计算团队采用 Hadoop。
- 2006 年 4 月——标准排序（10GB 每个节点）在 188 个节点上运行 47.9 个小时。
- 2006 年 5 月——雅虎建立了一个 300 个节点的 Hadoop 研究集群。
- 2006 年 5 月——标准排序在 500 个节点上运行 42 个小时（硬件配置比 4 月的更好）。
- 2006 年 11 月——研究集群增加到 600 个节点。
- 2006 年 12 月——标准排序在 20 个节点上运行 1.8 个小时，100 个节点 3.3 小时，500 个节点 5.2 小时，900 个节点 7.8 个小时。
- 2007 年 1 月——研究集群到达 900 个节点。
- 2007 年 4 月——研究集群达到两个 1000 个节点的集群。
- 2008 年 4 月——赢得世界最快 1TB 数据排序在 900 个节点上用时 209 秒。
- 2008 年 10 月——研究集群每天装载 10TB 的数据。
- 2009 年 3 月——17 个集群总共 24000 台机器。
- 2009 年 4 月——赢得每分钟排序，59 秒内排序 500GB（在 1400 个节点上）和 173 分钟内排序 100TB 数据（在 3400 个节点上）。

## 3.3 Hadoop 的功能与作用

我们为什么需要 Hadoop 呢？众所周知，现代社会的信息量增长速度极快，这些信息里又积累着大量的数据，其中包括个人数据和工业数据。预计到 2020 年，每年产生的数字信息将会有超过 1/3 的内容驻留在云平台中或借助云平台来处理。我们需要对这些数据进行分析和处理，以获取更多有价值的信息。那么我们如何高效地存储和管理这些数据，如何分析

这些数据呢？这时可以选用 Hadoop 系统，它在处理这类问题时，采用了分布式存储方式，提高了读写速度，并扩大了存储容量。采用 MapReduce 来整合分布式文件系统上的数据，可保证分析和处理数据的高效。与此同时，Hadoop 还采用存储冗余数据的方式保证了数据的安全性。

Hadoop 中 HDFS 的高容错特性以及它是基于 Java 语言开发的，这使得 Hadoop 可以部署在低廉的计算机集群中，同时不限于某个操作系统。Hadoop 中 HDFS 的数据管理能力，MapReduce 处理任务时的高效率，以及它的开源特性，使其在同类的分布式系统中大放异彩，并在众多行业和科研领域中被广泛采用。

## 3.4为什么不用关系型数据库管理系统

为什么我们不能使用数据库加上更多磁盘来做大规模的批量分析呢？为什么需要 MapReduce？

这个问题的答案来自于磁盘驱动器的另一个发展趋势：寻址时间的提高速度远远慢于传输速率的提高速度。寻址就是将磁头移动到特定位置进行读写操作的工序，它的特点是磁盘操作有延迟，而传输速率对应于磁盘的带宽。

如果数据的访问模式受限于磁盘的寻址，势必会导致它花更长时间(相较于流)来读或写大部分数据。另一方面，在更新一小部分数据库记录的时候，传统的 B 树(关系型数据库中使用的一种数据结构，受限于执行查找的速度)效果很好。但在更新大部分数据库数据的时候，B 树的效率就没有 MapReduce 的效率高，因为它需要使用排序/合并来重建数据库。

在许多情况下，MapReduce 能够被视为一种 RDBMS(关系型数据库管理系统)的补充(两个系统之间的差异见表 3-1)。MapReduce 很适合处理那些需要分析整个数据集的问题（以批处理的方式），尤其是 Ad Hoc(自主或即时)分析，而 RDBMS 则适用于点查询和更新（其中，数据集已经被索引以提供低延迟的检索和短时间的少量数据更新）。MapReduce 适合数据被一次写入和多次读取的应用，而关系型数据库更适合持续更新的数据集。

表 3-1 关系型数据库和 MapReduce 的比较

| | 传统关系型数据库 | MapReduce |
|---|---|---|
| 数据大小 | GB | PB |
| 访问 | 交互型和批处理 | 批处理 |
| 更新 | 多次读写 | 一次写入多次读取 |

| 结构 | 静态模式 | 动态模式 |
| --- | --- | --- |
| 集成度 | 高 | 低 |
| 伸缩性 | 非线性 | 线性 |

MapReduce 和关系型数据库之间的另一个区别是，它们操作的数据集中的结构化数据的数量。结构化数据是拥有准确定义的实体化数据，具有诸如数据库表定义的格式，符合特定的预定义模式，这就是 RDBMS 包括的内容。另一方面，半结构化数据比较宽松（比如 XML 文档），虽然可能有模式，但经常被忽略，所以它只能用作数据结构指南。非结构化数据没有什么特别的内部结构，例如纯文本或图像数据。MapReduce 对于非结构化或半结构化数据非常有效，因为它被设计为在处理时间内解释数据。换句话说：MapReduce 输入的键和值并不是数据固有的属性，它们是由分析数据的人来选择的。

关系型数据往往是规范的，以保持其完整性和删除冗余。但是，规范化会给 MapReduce 带来问题，因为它使读取记录成为一个非本地操作，并且 MapReduce 的核心假设之一就是，它可以进行(高速)流的读写。

Web 服务器日志是记录集的一个很好的非规范化例子(例如，客户端主机名每次都以全名来指定，即使同一客户端可能会出现很多次)，这也是 MapReduce 非常适合用于分析各种日志文件的原因之一。

MapReduce 是一种线性的可伸缩的编程模型。程序员编写两个函数——map 函数和 Reduce 函数——每一个都定义一个键/值对集映射到另一个。这些函数无视数据的大小或者它们正在使用的集群的特性，这样它们就可以原封不动地应用到小规模数据集或者大的数据集上。更重要的是，如果放入两倍的数据量，运行的时间会少于两倍。但是如果是两倍大小的集群，一个任务仍然只是和原来的一样快。这不是一般的 SQL 查询的效果。

随着时间的推移，关系型数据库和 MapReduce 之间的差异很可能变得模糊。一方面，关系型数据库都开始吸收 MapReduce 的一些思路(如 ASTER DATA 和 GreenPlum 的数据库)；另一方面，基于 MapReduce 的高级查询语言(如 Pig 和 Hive)使 MapReduce 的系统更接近传统的数据库编程人员。

## 3.5 Hadoop 的优点

Hadoop 是一个能够对大量数据进行分布式处理的软件框架，并且是以一种可靠、高效、

可伸缩的方式进行处理的，它具有以下几个方面的优点：

- 高可靠性：因为它假设计算元素和存储会失败，因此它维护多个工作数据副本，确保能够针对失败的节点重新分布处理。
- 高效性：因为它以并行的方式工作，通过并行处理加快处理速度。Hadoop 还是可伸缩的，能够处理 PB 级数据。Hadoop 能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此其处理速度非常快。
- 高可扩展性：Hadoop 是在可用的计算机集簇间分配数据并完成计算任务的，这些集簇可以方便地扩展到数以千计的节点中。
- 高容错性：Hadoop 能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。
- Hadoop 成本低：依赖于廉价服务器，因此它的成本比较低，任何人都可以使用。
- 运行在 Linux 平台上：Hadoop 带有用 Java 语言编写的框架，因此运行在 Linux 生产平台上是非常理想的。
- 支持多种编程语言：Hadoop 上的应用程序也可以使用其他语言编写，比如 C++。

## 3.6 Hadoop 的应用现状和发展趋势

由于Hadoop优势突出，基于Hadoop的应用已经遍地开花，尤其是在互联网领域。Yahoo!通过集群运行Hadoop，以支持广告系统和Web搜索的研究；Facebook借助集群运行Hadoop，以支持其数据分析和机器学习；百度则使用Hadoop进行搜索日志的分析和网页数据的挖掘工作；淘宝的Hadoop系统用于存储并处理电子商务交易的相关数据；中国移动研究院基于Hadoop的“大云”（BigCloud）系统，用于对数据进行分析和对外提供服务。

2008年2月，Hadoop最大贡献者的Yahoo！构建了当时规模最大的Hadoop应用，它们在2000个节点上面执行了超过1万个Hadoop虚拟机器，来处理超过5PB的网页内容，分析大约 1 兆个网络连接之间的网页索引资料。这些网页索引资料压缩后超过 300TB。Yahoo！正是基于这些为用户提供了高质量的搜索服务。

Hadoop目前已经取得了非常突出的成绩。随着互联网的发展，新的业务模式还将不断涌现，Hadoop 的应用也会从互联网领域向电信、电子商务、银行、生物制药等领域拓展。相信在未来，Hadoop将会在更多的领域中扮演幕后英雄，为我们提供更加快捷优质的服务。

# 3.7Hadoop 项目及其结构

Hadoop 有许多元素构成。最底部是 Hadoop Distributed File System（HDFS），它存储 Hadoop 集群中所有存储节点上的文件。HDFS 的上一层是 MapReduce 引擎，该引擎由 JobTrackers 和 TaskTrackers 组成。

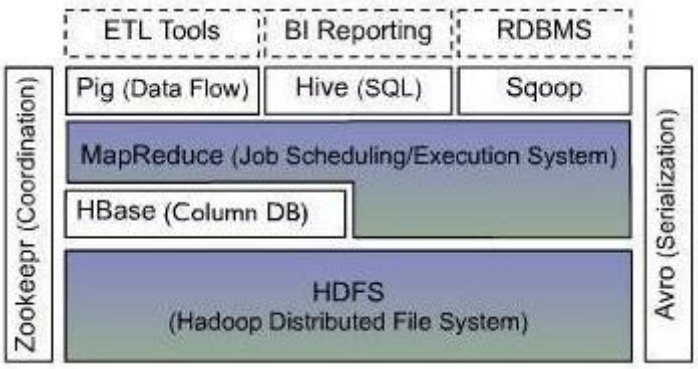


图 3-2 Hadoop 生态系统图

图 3-2 描述了 Hadoop 生态系统中的各层子系统，具体如下：

- **Avro**

Avro 是用于数据序列化的系统。它提供了丰富的数据结构类型、快速可压缩的二进制数据格式、存储持久性数据的文件集、远程调用 RPC 的功能和简单的动态语言集成功能。其中，代码生成器既不需要读写文件数据，也不需要使用或实现 RPC 协议，它只是一个可选的对静态类型语言的实现。Avro 系统依赖于模式（Schema），Avro 数据的读和写是在模式之下完成的。这样就可以减少写入数据的开销，提高序列化的速度并缩减其大小。同时，也可以方便动态脚本语言的使用，因为数据连同其模式都是自描述的。在 RPC 中，Avro 系统的客户端和服务端通过握手协议进行模式的交换。因此当客户端和服务端拥有彼此全部的模式时，不同模式下的相同命名字段、丢失字段和附加字段等信息的一致性问题就得到了很好的解决。

- **HDFS**

HDFS 是一种分布式文件系统，运行于大型商用机集群，HDFS 为 HBase 提供了高可靠性的底层存储支持；由于 HDFS 具有高容错性（fault-tolerant）的特点，所以可以设计部署在低廉（low-cost）的硬件上。它可以以很高的高吞吐率（high throughput）来访问应用程序的数据，适合那些有着超大数据集的应用程序。HDFS 放宽了可移植操作系统接口（POSIX，Portable Operating System Interface）的要求，这样就可以实现以流的形式访问文件系统中的

数据。HDFS 原本是开源的 Apache 项目 Nutch 的基础结构，最后它成为了 Hadoop 的基础架构之一。

以下是 HDFS 的设计目标：

（1）检测和快速恢复硬件故障。硬件故障是常见的问题，整个 HDFS 系统由数百台或数千台存储着数据文件的服务器组成，而如此多的服务器意味着高故障率，因此，故障的检测和自动快速恢复是 HDFS 的一个核心目标。

（2）流式的数据访问。HDFS 使应用程序能流式地访问它们的数据集。HDFS 被设计成适合进行批量处理，而不是用户交互式的处理。所以它重视数据吞吐量，而不是数据访问的反应速度。

（3）简化一致性模型。大部分的 HDFS 程序操作文件时需要一次写入，多次读取。一个文件一旦经过创建、写入、关闭之后就不需要修改了，从而简化了数据一致性问题和高吞吐量的数据访问问题。

（4）通信协议。所有的通信协议都在 TCP/IP 协议之上。一个客户端和明确配置了端口的目录节点（NameNode）建立连接之后，它和目录节点（NameNode）的协议便是客户端协议（Client Protocal）。数据节点（DataNode）和目录节点（NameNode）之间则用数据节点协议（DataNode Protocal）。

- **HBase**

HBase 位于结构化存储层，是一个分布式的列存储数据库；该技术来源于 Google 的论文“Bigtable：一个结构化数据的分布式存储系统”。如同 Bigtable 利用了 Google 文件系统（Google File System）提供的分布式数据存储方式一样，HBase 在 Hadoop 之上提供了类似于 Bigtable 的能力。HBase 是 Hadoop 项目的子项目。HBase 不同于一般的关系数据库，其一，HBase 是一个适合于存储非结构化数据的数据库；其二，HBase 是基于列而不是基于行的模式。HBase 和 Bigtable 使用相同的数据模型。用户将数据存储在一个表里，一个数据行拥有一个可选择的键和任意数量的列。由于 HBase 表示疏松的，用户可以给行定义各种不同的列。HBase 主要用于需要随机访问、实时读写的大数据（BigData）。

- **MapReduce**

Mapreduce 是一种编程模型，用于大规模数据集（大于 1TB）的并行运算。“映射”（map）、“化简”（reduce）等概念和它们的主要思想都是从函数式编程语言中借来的。它使得编程人员在不了解分布式并行编程的情况下也能方便地将自己的程序运行在分布式系统上。MapReduce 在执行时先指定一个 map（映射）函数，把输入键值对映射成一组新的键值对，

经过一定的处理后交给 reduce，reduce 对相同 key 下的所有 value 进行处理后再输出键值对作为最终的结果。

- **Zookeeper**

Zookeeper 是一个分布式的、高可用性的协调服务，提供分布式锁之类的基本服务，用于构建分布式应用，为 HBase 提供了稳定服务和失败恢复机制。

- **Hive**

Hive 最早是由 Facebook 设计的，是一个建立在 Hadoop 基础之上的数据仓库，它提供了一些对存储在 Hadoop 文件中的数据集进行数据整理、特殊查询和分析的工具。Hive 提供的是一种结构化数据的机制，它支持类似于传统 RDBMS 中的 SQL 语言来帮助那些熟悉 SQL 的用户查询 Hadoop 中的数据，该查询语言称为 HiveQL。与此同时，那些传统的 MapReduce 编程人员也可以在 Mapper 或 Reducer 中通过 HiveQL 查询数据。Hive 编译器会把 HiveQL 编译成一组 MapReduce 任务，从而方便 MapReduce 编程人员进行 Hadoop 应用的开发。

- **Pig**

Pig 是一种数据流语言和运行环境，用以检索非常大的数据集，大大简化了 Hadoop 常见的工作任务。Pig 可以加载数据、表达转换数据以及存储最终结果。Pig 内置的操作使得半结构化数据变得有意义（如日志文件）。Pig 和 Hive 都为 HBase 提供了高层语言支持，使得在 HBase 上进行数据统计处理变的非常简单；但是，二者还是有所区别的。Hive 在 Hadoop 中扮演数据仓库的角色，允许使用类似于 SQL 语法进行数据查询。Hive 更适合于数据仓库的任务，主要用于静态的结构以及需要经常分析的工作。Hive 与 SQL 相似，这一点使其成为 Hadoop 与其他 BI 工具结合的理想交集。Pig 赋予开发人员在大数据集领域更多的灵活性，并允许开发简洁的脚本用于转换数据流以便嵌入到较大的应用程序。与 Hive 相比较，Pig 属于较轻量级，它主要的优势是，相比于直接使用 Hadoop Java APIs 而言，使用 Pig 可以大幅削减代码量。

- **Sqoop**

Sqoop 为 HBase 提供了方便的 RDBMS 数据导入功能，使得传统数据库数据向 HBase 中迁移变得非常方便。

## 3.8 Hadoop 的体系结构

HDFS 和 MapReduce 是 Hadoop 的两大核心。而整个 Hadoop 的体系结构主要是通过

HDFS 来实现对分布式存储的底层支持的，并且它会通过 MapReduce 来实现对分布式并行任务处理的程序支持。

## 3.8.1 HDFS 的体系结构

我们首先介绍 HDFS 的体系结构。HDFS 采用了主从（Master/Slave）结构模型，一个 HDFS 集群是由一个 NameNode 和若干个 DataNode 组成的。其中 NameNode 作为主服务器，管理文件系统的命名空间和客户端对文件的访问操作；集群中的 DataNode 管理存储的数据。HDFS 允许用户以文件的形式存储数据。从内部来看，文件被分成若干个数据块，而且这若干个数据块存放在一组 DataNode 上。NameNode 执行文件系统的命名空间操作，比如打开、关闭、重命名文件或目录等，它也负责数据块到具体 DataNode 的映射。DataNode 负责处理文件系统客户端的文件读写请求，并在 NameNode 的统一调度下进行数据块的创建、删除和复制工作。图 3-3 给出了 HDFS 的体系结构。

NameNode 和 DataNode 都被设计成可以在普通商用计算机上运行。这些计算机通常运行的是 GNU/Linux 操作系统。HDFS 采用 Java 语言开发，因此，任何支持 Java 的机器都可以部署 NameNode 和 DataNode。一个典型的部署场景是集群中的一台机器运行一个 NameNode 实例，其他机器分别运行一个 DataNode 实例。当然，并不排除一台机器运行多个 DataNode 实例的情况。集群中单一的 NameNode 的设计则大大简化了系统的架构。NameNode 是所有 HDFS 元数据的管理者，用户数据永远不会经过 NameNode。

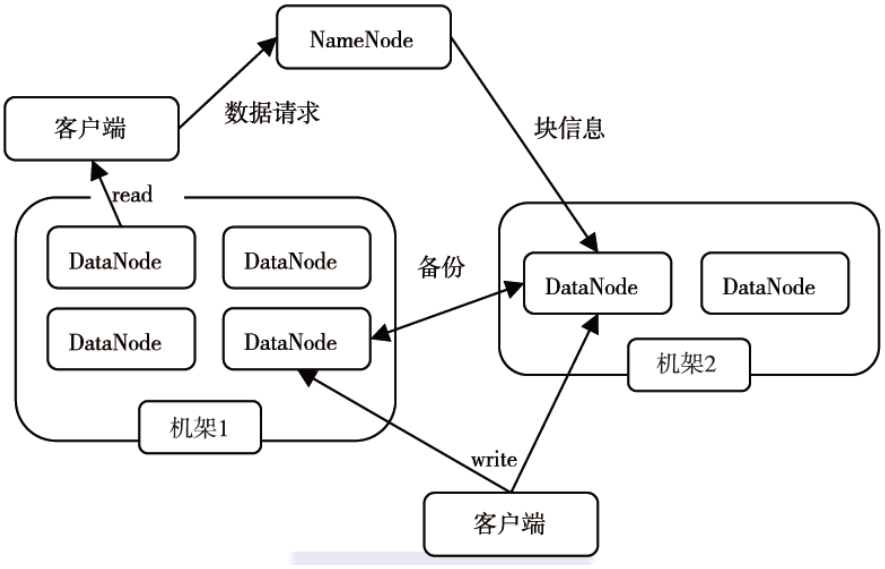


图 3-3 HDFS 的体系结构图

## 3.8.2 MapReduce 的体系结构

接下来介绍 MapReduce 的体系结构。MapReduce 是一种并行编程模式，这种模式使得软件开发者可以轻松地编写出分布式并行程序。在 Hadoop 的体系结构中，MapReduce 是一个简单易用的软件框架，基于它可以将任务分发到由上千台商用机器组成的集群上，并以一种高容错的方式并行处理大量的数据集，实现 Hadoop 的并行任务处理功能。MapReduce 框架是由一个单独运行在“主节点”上的 JobTracker 和运行在每个集群“从节点”上的 TaskTracker 共同组成的。主节点负责调度构成一个作业的所有任务，这些任务分布在不同的从节点上。主节点监控它们的执行情况，并且重新执行之前失败的任务；从节点仅负责由主节点指派的任务。当一个 Job 被提交时，JobTracker 接收到提交作业和配置信息之后，就会将配置信息等分发给从节点，同时调度任务并监控 TaskTracker 的执行。

从上面的介绍可以看出，HDFS 和 MapReduce 共同组成了 Hadoop 分布式系统体系结构的核心。HDFS 在集群上实现了分布式文件系统，MapReduce 在集群上实现了分布式计算和任务处理。HDFS 在 MapReduce 任务处理过程中提供了文件操作和存储等支持，MapReduce 在 HDFS 的基础上实现了任务的分发、跟踪、执行等工作，并收集结果，二者相互作用，完成了 Hadoop 分布式集群的主要任务。

## 3.9 Hadoop 与分布式开发

我们通常说的分布式系统其实是分布式软件系统，即支持分布式处理的软件系统，它是在通信网络互联的多处理机体系结构上执行任务的，包括分布式操作系统、分布式程序设计语言及其编译（解释）系统、分布式文件系统和分布式数据库系统等。Hadoop 是分布式软件系统中文件系统这一层的软件，它实现了分布式文件系统和部分分布式数据库的功能。

Hadoop 中的分布式文件系统 HDFS，能够实现数据在计算机集群组成的云上高效的存储和管理。Hadoop 中的并行编程框架 MapReduce，能够让用户编写的 Hadoop 并行应用程序运行更加简化。

Hadoop 上的并行应用程序开发是基于 MapReduce 编程框架的。MapReduce 编程模型的原理是：利用一个输入的 key/value 对集合来产生一个输出的 key/value 对集合。MapReduce 库的用户用两个函数来表达这个计算：Map 和 Reduce。

用户自定义的 map 函数接收一个输入的 key/value 对，然后产生一个中间 key/value 对的

集合。MapReduce 把所有具有相同 key 值的 value 集合在一起，然后传递给 reduce 函数。用户自定义的 reduce 函数接收 key 和相关的 value 集合。reduce 函数合并这些 value 值，形成一个较小的 value 集合。一般来说，每次 reduce 函数调用只产生 0 或 1 个输出的 value 值。通常我们通过一个迭代器把中间的 value 值提供给 reduce 函数，这样就可以处理无法全部放入内存中的大量的 value 值集合了。

图 3-4 是 MapReduce 的数据流图，这个过程简而言之就是将大数据集分解为成百上千个小数据集，每个（或若干个）数据集分别由集群中的一个节点（一般就是一台普通的计算机）进行处理并生成中间结果，然后这些中间结果又由大量的节点合并，形成最终结果。图 3-4 也指出了 MapReduce 框架下并行程序中的两个主要函数：map 和 reduce。在这个结构中，需要用户完成的工作仅仅是根据任务编写 map 和 reduce 两个函数。

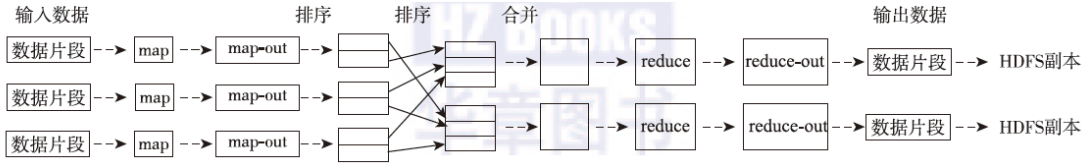


图 3-4MapReduce 的数据流图

MapReduce 计算模型非常适合在大量计算机组成的大规模集群上并行运行。图 3-4 中的每一个 map 任务和每一个 reduce 任务均可以同时运行于一个单独的计算节点上，可想而知，其运算效率是很高的，那么这样的并行计算是如何做到的呢？下面将简单介绍一下其原理。

**1. 数据分布存储**

Hadoop 分布式文件系统（HDFS）由一个目录节点（NameNode）和 N 个数据节点（DataNode）组成，每个节点均是一台普通的计算机。在使用方式上 HDFS 与我们熟悉的单机文件系统非常类似，它可以创建目录，创建、复制和删除文件，以及查看文件的内容等。但 HDFS 底层把文件切割成了 Block，然后这些 Block 分散地存储于不同的 DataNode 上，每个 Block 还可以复制数份数据存储于不同的 DataNode 上，达到容错容灾的目的。NameNode 则是整个 HDFS 的核心，它通过维护一些数据结构来记录每一个文件被切割成了多少个 Block、这些 Block 可以从哪些 DataNode 中获得，以及各个 DataNode 的状态等重要信息。

**2. 分布式并行计算**

Hadoop 中有一个作为主控的 JobTracker，用于调度和管理其他的 TaskTracker，JobTracker 可以运行于集群中的任意一台计算机上。TaskTracker 则负责执行任务，它必须运行于 DataNode 上，也就是说 DataNode 既是数据存储节点，也是计算节点。JobTracker 将 map 任务和 reduce 任务分发给空闲的 TaskTracker，让这些任务并行运行，并负责监控任务的运行

情况。如果某一个 TaskTracker 出了故障，JobTracker 会将其负责的任务转交给另一个空闲的 TaskTracker 重新运行。

**3. 本地计算**

数据存储在哪一台计算机上，就由哪台计算机进行这部分数据的计算，这样可以减少数据在网络上的传输，降低对网络带宽的需求。在 Hadoop 这类基于集群的分布式并行系统中，计算节点可以很方便地扩充，它所能够提供的计算能力近乎无限，但是由于数据需要在不同的计算机之间流动，故网络带宽变成了瓶颈，“本地计算”是一种最有效的节约网络带宽的手段，业界把这形容为“移动计算比移动数据更经济”。

**4. 任务粒度**

把原始大数据集切割成小数据集时，通常让小数据集小于或等于 HDFS 中一个 Block 的大小（默认是 64MB），这样能够保证一个小数据集是位于一台计算机上的，便于本地计算。有 M 个小数据集待处理，就启动 M 个 map 任务，注意这 M 个 map 任务分布于 N 台计算机上，它们会并行运行，reduce 任务的数量 R 则可由用户指定。

**5. 数据分割（Partition）**

把 map 任务输出的中间结果按 key 的范围划分成 R 份（R 是预先定义的 reduce 任务的个数），划分时通常使用 hash 函数（如：hash(key) mod R），这样可以保证某一范围内的 key 一定是由一个 reduce 任务来处理的，可以简化 Reduce 的过程。

**6. 数据合并（Combine）**

在数据分割之前，还可以先对中间结果进行数据合并（Combine），即将中间结果中有相同 key 的 <key, value> 对合并成一对。Combine 的过程与 reduce 的过程类似，很多情况下可以直接使用 reduce 函数，但 Combine 是作为 map 任务的一部分，在执行完 map 函数后紧接着执行的。Combine 能够减少中间结果中<key, value>对的数目，从而降低网络流量。

**7. Reduce**

Map 任务的中间结果在做完 Combine 和 Partition 之后，以文件形式存于本地磁盘上。中间结果文件的位置会通知主控 JobTracker，JobTracker 再通知 reduce 任务到哪一个 DataNode 上去取中间结果。注意，所有的 map 任务产生的中间结果均按其 key 值用同一个 hash 函数划分成了 R 份，R 个 reduce 任务各自负责一段 key 区间。每个 reduce 需要向许多个 map 任务节点取得落在其负责的 key 区间内的中间结果，然后执行 reduce 函数，形成一个最终的结果文件。

**8. 任务管道**

有 R 个 reduce 任务，就会有 R 个最终结果，很多情况下这 R 个最终结果并不需要合并成一个最终结果，因为这 R 个最终结果又可以作为另一个计算任务的输入，开始另一个并行计算任务，这也就形成了任务管道。

# 3.10Hadoop 应用案例

随着企业的数据量的迅速增长，存储和处理大规模数据已成为企业的迫切需求。Hadoop 作为开源的云计算平台，已引起了学术界和企业的普遍兴趣。

在学术方面，Hadoop 得到了各科研院所的广泛关注，多所著名大学加入到 Hadoop 集群的研究中来，其中包括斯坦福大学、加州大学伯克利分校、康奈尔大学、卡耐基.梅隆大学、普渡大学等。一些国内高校和科研院所如中科院计算所、清华大学、中国人民大学等也开始对 Hadoop 展开相关研究，研究内容涉及 Hadoop 的数据存储、资源管理、作业调度、性能优化、系统可用性和安全性等多个方面。

在商业方面，Hadoop 技术已经在互联网领域得到了广泛的应用。互联网公司往往需要存储海量的数据并对其进行处理，而这正是 Hadoop 的强项。如 Facebook 使用 Hadoop 存储内部的日志拷贝以及数据挖掘和日志统计；Yahoo！利用 Hadoop 支持广告系统并处理网页搜索；Twitter 则使用 Hadoop 存储微博数据、日志文件和其他中间数据等。在国内，Hadoop 同样也得到了许多公司的青睐，如百度主要将 Hadoop 应用于日志分析和网页数据库的数据挖掘；阿里巴巴则将 Hadoop 用于商业数据的排序和搜索引擎的优化等。

下面我们将选取具有代表性的 Hadoop 应用案例进行分析，让读者了解 Hadoop 在企业界的应用情况。

## 3.10.1Hadoop 在 Yahoo！的应用

关于 Hadoop 技术的研究和应用，Yahoo！始终处于领先地位，它将 Hadoop 应用于自己的各种产品中，包括数据分析、内容优化、反垃圾邮件系统、广告的优化选择、大数据处理和 ETL 等；同样，在用户兴趣预测、搜索排名、广告定位等方面得到了充分的应用。

在 Yahoo！主页个性化方面，实时服务系统通过 Apache 从数据库中读取 user 到 interest 的映射，并且每隔 5 分钟生产环境中的 Hadoop 集群就会基于最新数据重新排列内容，每隔 7 分钟则在页面上更新内容。

在邮箱方面，Yahoo！利用 Hadoop 集群根据垃圾邮件模式为邮件计分，并且每隔几个小时就在集群上改进反垃圾邮件模型，集群系统每天还可以推动 50 亿次的邮件投递。

目前 Hadoop 最大的生产应用是 Yahoo！的 Search Webmap 应用，它运行在超过 10000 台机器的 Linux 系统集群里，Yahoo！的网页搜索查询使用的就是它产生的数据。Webmap 的构建步骤如下：首先进行网页的爬取，同时产生包含所有已知网页和互联网站点的数据库，以及一个关于所有页面及站点的海量数据组；然后将这些数据传输给 Yahoo！搜索中心执行排序算法。在整个过程中，索引中页面间的链接数量将会达到 1TB，经过压缩的数据产出量会达到 300TB，运行一个 MapReduce 任务就需使用超过 10000 的内核，而在生产环境中使用数据的存储量超过 5PB。

Yahoo！在 Hadoop 中同时使用了 Hive 和 Pig，在许多人看来，Hive 和 Pig 大体上相似，而且 Pig Latin 与 SQL 也十分相似。那么 Yahoo！为什么要同时使用这些技术呢？主要是因为 Yahoo！的研究人员在查看了它们的工作负载并分析了应用案例后认为，不同的情况下需要使用不同的工具。

先了解一下大规模数据的使用和处理背景。大规模的数据处理经常分为三个不同的任务：数据收集、数据准备和数据表示，这里并不打算介绍数据收集阶段，因为 Pig 和 Hive 主要用于数据准备和数据表示阶段。

数据准备阶段通常被认为是提取、转换和加载（Extract Transform Load，ETL）数据的阶段，或者认为这个阶段是数据工厂。这里的数据工厂只是一个类比，在现实生活中的工厂接收原材料后会生产出客户所需的产品，而数据工厂与之相似，它在接收原始数据后，可以输出供客户使用的数据集。这个阶段需要装载和清洗原始数据，并让它遵守特定的数据模型，还要尽可能地让它与其他数据源结合等。这一阶段的客户一般都是程序员、数据专家或研究者。

数据表示阶段一般指的都是数据仓库，数据仓库存储了客户所需要的产品，客户会根据需要选取合适的产品。这一阶段的客户可能是系统的数据工程师、分析师或决策者。根据每个阶段负载和用户情况的不同，Yahoo！在不同的阶段使用不同的工具。结合了诸如 Oozie 等工作流系统的 Pig 特别适合于数据工厂，而 Hive 则适合于数据仓库。下面将分别介绍数据工厂和数据仓库。

Yahoo！的数据工厂存在三种不同的工作用途：流水线、迭代处理和科学研究。

经典的数据流水线包括数据反馈、清洗和转换。一个常见例子是 Yahoo！的网络服务器日志，这些日志需要进行清洗以去除不必要的信息，数据转换则是要找到点击之后所转到的

页面。Pig 是分析大规模数据集的平台，它建立在 Hadoop 之上并提供了良好的编程环境、优化条件和可扩展的性能。Pig Latin 是关系型数据流语言，并且是 Pig 核心的一部分，基于以下的原因，Pig Latin 相比于 SQL 而言，更适合构建数据流。首先，Pig Latin 是面向过程的，并且 Pig Latin 允许流水线开发者自定义流水线中检查点的位置；其次，Pig Latin 允许开发者直接选择特定的操作实现方式而不是依赖于优化器；最后，Pig Latin 支持流水线的分支，并且 Pig Latin 允许流水线开发者在数据流水线的任何地方插入自己的代码。Pig 和诸如 Oozie 等的工作流工具一起使用来创建流水线，一天可以运行数以万计的 Pig 作业。

迭代处理也是需要 Pig 的，在这种情况下通常需要维护一个大规模的数据集。数据集上的典型处理包括加入一小片数据后就会改变大规模数据集的状态。如考虑这样一个数据集，它存储了 Yahoo！新闻中现有的所有新闻。我们可以把它想象成一幅巨大的图，每个新闻就是一个节点，新闻节点若有边相连则说明这些新闻指的是同一个事件。每隔几分钟就会有新的新闻加入进来，这些工具需要将这些新闻节点加到图中，并找到相似的新闻节点用边连接起来，还要删除被新节点覆盖的旧节点。这和标准流水线不同的是，它不断有小变化，这就需要使用增长处理模型在合理的时间范围内处理这些数据了。例如，所有的新节点加入图中后，又有一批新的新闻节点到达，在整个图上重新执行连接操作是不现实的，这也许会花费数个小时。相反，在新增加的节点上执行连接操作并使用全连接（full join）的结果是可行的，而且这个过程只需要花费几分钟时间。标准的数据库操作可以使用 Pig Latin 通过上述方式实现，这时 Pig 就会得到很好的应用。

Yahoo！有许多的科研人员，他们需要用网格工具处理千万亿大小的数据，还有许多研究人员希望快速地写出脚本来测试自己的理论或获得更深的理解。但是在数据工厂中，数据不是以一种友好的、标准的方式呈现的，这时 Pig 就可以大显身手了，因为它可以处理未知模式的数据，还有半结构化和非结构化的数据。Pig 与 streaming 相结合使得研究者在小规模数据集上测试的 Perl 和 Python 脚本可以很方便地在大规模数据集上运行。

在数据仓库处理阶段，有两个主要的应用：商业智能分析和特定查询（Ad-hoc query）。在第一种情况下，用户将数据连接到商业智能（BI）工具（如 MicroStrategy）上来产生报告或深入的分析。在第二种情况下，用户执行数据分析师或决策者的特定查询。这两种情况下，关系模型和 SQL 都很好用。事实上，数据仓库已经成为 SQL 使用的核心，它支持多种查询并具有分析师所需的工具，Hive 作为 Hadoop 的子项目为其提供了 SQL 接口和关系模型，现在 Hive 团队正开始将 Hive 与 BI 工具通过接口（如 ODBC）结合起来使用。

Pig 在 Yahoo！得到了广泛应用，这使得数据工厂的数据被移植到 Hadoop 上运行成为

可能。随着 Hive 的深入使用，Yahoo！打算将数据仓库移植到 Hadoop 上。在同一系统上部署数据工厂和数据仓库将会降低数据加载到仓库的时间，这也使得共享工厂和仓库之间的数据、管理工具、硬件等成为可能。Yahoo！在 Hadoop 上同时使用多种工具使 Hadoop 能够执行更多的数据处理。

## 3.10.2 Hadoop 在 eBay 的应用

在 eBay 上存储着上亿种商品的信息，而且每天有数百万种的新商品增加，因此需要用云系统来存储和处理 PB 级别的数据，而 Hadoop 则是个很好的选择。

Hadoop 是建立在商业硬件上的容错、可扩展、分布式的云计算框架，eBay 利用 Hadoop 建立了一个大规模的集群系统—Athena，它被分为五层（如图 3-5 所示），下面从最底层向上开始介绍：

1) Hadoop 核心层：包括 Hadoop 运行时环境、一些通用设施和 HDFS，其中文件系统为读写大块数据而做了一些优化，如将块的大小由 128MB 改为 256MB。
2) MapReduce 层：为开发和执行任务提供 API 和控件。
3) 数据获取层：现在数据获取层的主要框架是 HBase、Pig 和 Hive：
   - HBase 是根据 Google BigTable 开发的按列存储的多维空间数据库，通过维护数据的划分和范围提供有序的数据，其数据储存在 HDFS 上。
   - Pig（Latin）是提供加载、筛选、转换、提取、聚集、连接、分组等操作的面向过程的语言，开发者使用 Pig 建立数据管道和数据工厂。
   - Hive 是用于建立数据仓库的使用 SQL 语法的声明性语言。对于开发者、产品经理和分析师来说，SQL 接口使得 Hive 成为很好的选择。
4) 工具和加载库层：UC4 是 eBay 从多个数据源自动加载数据的企业级调度程序。加载库有：统计库（R）、机器学习库（Mahout）、数学相关库（Hama）和 eBay 自己开发的用于解析网络日志的库（Mobius）。
5) 监视和警告层：Ganglia 是分布式集群的监视系统，Nagios 则用来警告一些关键事件如服务器不可达、硬盘已满等。

eBay 的企业服务器运行着 64 位的 RedHat Linux：

- NameNode 负责管理 HDFS 的主服务器；

- JobTracker 负责任务的协调；
- HBaseMaster 负责存储 HBase 存储的根信息，并且方便与数据块或存取区域进行协调；
- ZooKeeper 是保证 HBase 一致性的分布式锁协调器。


图 3-5 Athena 的层次

用于存储和计算的节点是 1U 大小的运行 CentOS 的机器，每台机器拥有 2 个四核处理器和 2TB 大小的存储空间，每 38~42 个节点单元为一个 rack，这组建成了高密度网格。有关网络方面，顶层 rack 交换机到节点的带宽为 1Gbps，rack 交换机到核心交换机的带宽为 40Gpbs。

这个集群是 eBay 内多个团队共同使用的，包括产品和一次性任务。这里使用 Hadoop 公平调度器（Fair Scheduler）来管理分配、定义团队的任务池、分配权限、限制每个用户和组的并行任务、设置优先权期限和延迟调度。

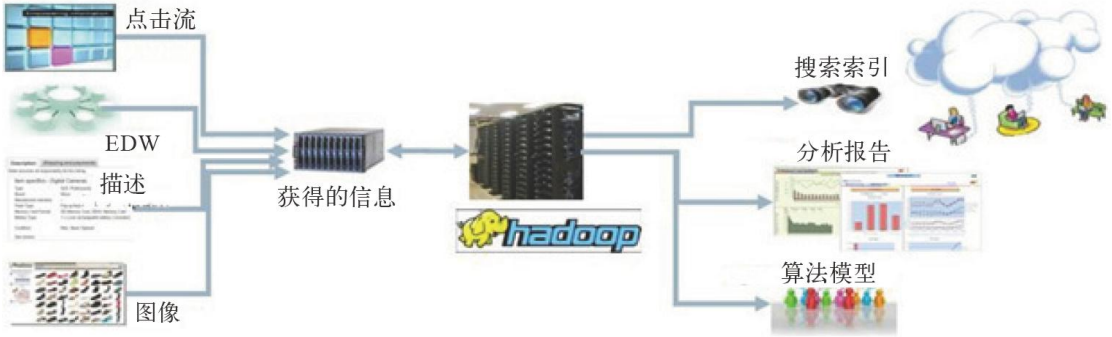


图 3-6 数据流

数据流的具体处理过程如图 3-6 所示，系统每天需要处理 8TB 至 10TB 的新数据，而 Hadoop 主要用于：

- **基于机器学习的排序**，使用 Hadoop 计算需要考虑多个因素（如价格、列表格式、卖家记录、相关性）的排序函数，并需要添加新因素来验证假设的扩展功能，以增强 eBay 物品搜索的相关性。
- **对物品描述数据的挖掘**，在完全无人监管的方式下使用数据挖掘和机器学习技术将物品描述清单转化为与物品相关的键/值对，以扩大分类的覆盖范围。

eBay 研究人员在系统构建和使用过程中遇到的挑战及一些初步计划有以下几个方面：

- **可扩展性**：当前主系统的 NameNode 拥有扩展的功能，随着集群的文件系统不断增长，需要存储大量的元数据，所以内存占有量也在不断增长。若是 1PB 的存储量则需要将近 1GB 的内存量，可能的解决方案是使用等级结构的命名空间划分，或者使用 HBase 和 ZooKeeper 联合对元数据进行管理。
- **有效性**：NameNode 的有效性对产品的工作负载很重要，开源社区提出了一些备用选择，如使用检查点和备份节点、从 Secondary NameNode 中转移到 Avatar 节点、日志元数据复制技术等。eBay 研究人员根据这些方法建立了自己的产品集群。
- **数据挖掘**：在存储非结构化数据的系统上建立支持数据管理、数据挖掘和模式管理的系统。新的计划提议将 Hive 的元数据和 Owl 添加到新系统中，并称为 Howl。eBay 研究人员努力将这个系统联系到分析平台上去，这样用户可以很容易地在不同的数据系统中挖掘数据。
- **数据移动**：eBay 研究人员考虑发布数据转移工具，这个工具可以支持在不同的子系统如数据仓库和 HDFS 之间进行数据的复制。
- **策略**：通过配额实现较好的归档、备份等策略（Hadoop 现有版本的配额需要改进）。eBay 的研究人员基于工作负载和集群的特点对不同的集群确定配额。
- **标准**：eBay 研究人员开发健壮的工具来为数据来源、消耗情况、预算情况、使用情况等进行度量。

同时 eBay 正在改变收集、转换、使用数据的方式，以提供更好的商业智能服务。

## 3.10.3 Hadoop 在百度的应用

百度作为全球最大的中文搜索引擎公司，提供基于搜索引擎的各种产品，包括以网络搜索为主的功能性搜索；以贴吧为主的社区搜索；针对区域、行业的垂直搜索、MP3 音乐搜索，以及百科等，几乎覆盖了中文网络世界中所有的搜索需求。

百度对海量数据处理的要求是比较高的，要在线下对数据进行分析，还要在规定的时间内处理完并反馈到平台上。百度在互联网领域的平台需求如图 3-7 所示，这里就需要通过性能较好的云平台进行处理了，Hadoop 就是很好的选择。在百度，Hadoop 主要应用于以下几个方面：

- 日志的存储和统计；
- 网页数据的分析和挖掘；
- 商业分析，如用户的行为和广告关注度等；
- 在线数据的反馈，及时得到在线广告的点击情况；
- 用户网页的聚类，分析用户的推荐度及用户之间的关联度。

MapReduce 主要是一种思想，不能解决所有领域内与计算有关的问题，百度的研究人员认为比较好的模型应该如图 3-8 所示，HDFS 实现共享存储，一些计算使用 MapReduce 解决，一些计算使用 MPI 解决，而还有一些计算需要通过两者来共同处理。因为 MapReduce 适合处理数据很大且适合划分的数据，所以在处理这类数据时就可以用 MapReduce 做一些过滤，得到基本的向量矩阵，然后通过 MPI 进一步处理后返回结果，只有整合技术才能更好地解决问题。

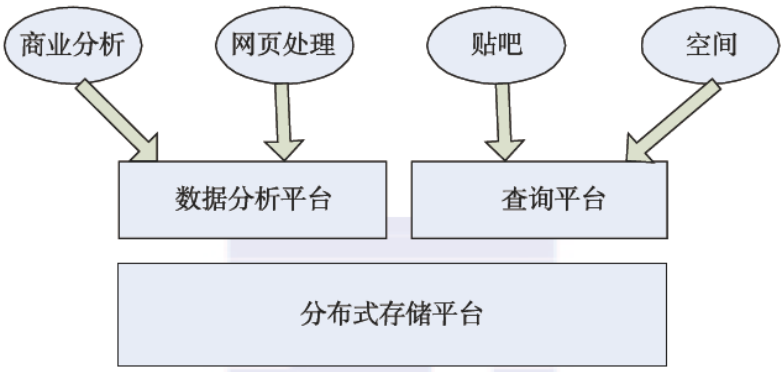


图 3-7 互联网领域的平台需求

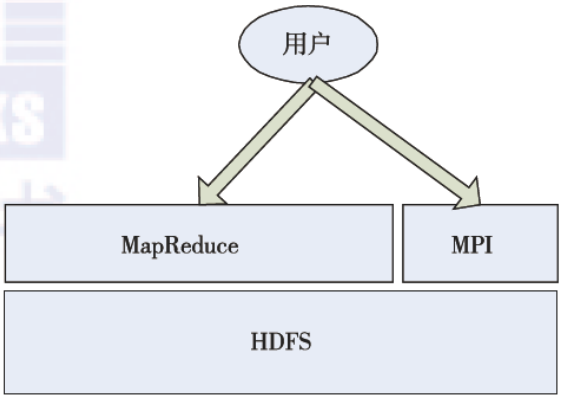


图 3-8 计算模型

百度现在拥有 3 个 Hadoop 集群，总规模在 700 台机器左右，其中有 100 多台新机器和 600 多台要淘汰的机器（它们的计算能力相当于 200 多台新机器），不过其规模还在不断的增加中。现在每天运行的 MapReduce 任务在 3000 个左右，处理数据约 120TB/天。

百度为了更好地用 Hadoop 进行数据处理，在以下几个方面做了改进和调整：

**1) 调整 MapReduce 策略**

- 限制作业处于运行状态的任务数；
- 调整预测执行策略，控制预测执行量，一些任务不需要预测执行；
- 根据节点内存状况进行调度；
- 平衡中间结果输出，通过压缩处理减少 I/O 负担。

**2) 改进 HDFS 的效率和功能**

- 权限控制，在 PB 级数据量的集群上数据应该是共享的，这样分析起来比较容易，但是需要对权限进行限制；
- 让分区与节点独立，这样，一个分区坏掉后节点上的其他分区还可以正常使用；
- 修改 DFSClient 选取块副本位置的策略，增加功能使 DFSClient 选取块时跳过出错的 DataNode；
- 解决 VFS（Virtual File System）的 POSIX（Portable Operating System Interface of Unix）兼容性问题。

**3) 修改 Speculative 的执行策略**

- 采用速率倒数替代速率，防止数据分布不均时经常不能启动预测执行情况的发生；
- 增加任务时必须达到某个百分比后才能启动预测执行的限制，解决 reduce 运行等待 map 数据的时间问题；
- 只有一个 map 或 reduce 时，可以直接启动预测执行。

**4) 对资源使用进行控制**

- 对应用物理内存进行控制。如果内存使用过多会导致操作系统跳过一些任务，百度通过修改 Linux 内核对进程使用的物理内存进行独立的限制，超过阈值可以终止进程。
- 分组调度计算资源，实现存储共享、计算独立，在 Hadoop 中运行的进程是不可抢占的。
- 在大块文件系统中，X86 平台下一个页的大小是 4KB。如果页较小，管理的

数据就会很多，会增加数据操作的代价并影响计算效率，因此需要增加页的大小。

百度在使用 Hadoop 时也遇到了一些问题，主要有：

- **MapReduce 的效率问题**：比如，如何在 shuffle 效率方面减少 I/O 次数以提高并行效率；如何在排序效率方面设置排序为可配置的，因为排序过程会浪费很多的计算资源，而一些情况下是不需要排序的。
- **HDFS 的效率和可靠性问题**：如何提高随机访问效率，以及数据写入的实时性问题，如果 Hadoop 每写一条日志就在 HDFS 上存储一次，效率会很低。
- **内存使用的问题**：reducer 端的 shuffle 会频繁地使用内存，这里采用类似 Linux 的 buddy system 来解决，保证 Hadoop 用最小的开销达到最高的利用率；当 Java 进程内容使用内存较多时，可以调整垃圾回收（GC）策略；有时存在大量的内存复制现象，这会消耗大量 CPU 资源，同时还会导致内存使用峰值极高，这时需要减少内存的复制。
- **作业调度的问题**：如何限制任务的 map 和 reduce 计算单元的数量，以确保重要计算可以有足够的计算单元；如何对 TaskTracker 进行分组控制，以限制作业执行的机器，同时还可以在用户提交任务时确定执行的分组并对分组进行认证。
- **性能提升的问题**：UserLogs cleanup 在每次 task 结束的时候都要查看一下日志，以决定是否清除，这会占用一定的任务资源，可以通过将清理线程从子 Java 进程移到 TaskTracker 来解决；子 Java 进程会对文本行进行切割而 map 和 reduce 进程则会重新切割，这将造成重复处理，这时需要关掉 Java 进程的切割功能；在排序的时候也可以实现并行排序来提升性能；实现对数据的异步读写也可以提升性能。
- **健壮性的问题**：需要对 mapper 和 reducer 程序的内存消耗进行限制，这就要修改 Linux 内核，增加其限制进程的物理内存的功能；也可以通过多个 map 程序共享一块内存，以一定的代价减少对物理内存的使用；还可以将 DataNode 和 TaskTracker 的 UGI 配置为普通用户并设置账号密码；或者让 DataNode 和 TaskTracker 分账号启动，确保 HDFS 数据的安全性，防止 Tracker 操作 DataNode 中的内容；在不能保证用户的每个程序都很健壮的情况下，有时需要将进程终止掉，但要保证父进程终止后子进程也被终止。

- **Streaming 局限性的问题**：比如，只能处理文本数据，mapper 和 reducer 按照文本行的协议通信，无法对二进制的数据进行简单处理。为了解决这个问题，百度人员新写了一个类 Bistreaming（Binary Streaming），这里的子 Java 进程 mapper 和 reducer 按照（KeyLen，Key，ValLen，Value）的方式通信，用户可以按照这个协议编写程序。
- **用户认证的问题**：这个问题的解决办法是让用户名、密码、所属组都在 NameNode 和 Job Tracker 上集中维护，用户连接时需要提供用户名和密码，从而保证数据的安全性。

百度下一步的工作重点可能主要会涉及以下内容：

- 内存方面：降低 NameNode 的内存使用并研究 JVM 的内存管理；
- 调度方面：改进任务可以被抢占的情况，同时开发出自己的基于 Capacity 的作业调度器，让等待作业队列具有优先级且队列中的作业可以设置 Capacity，并可以支持 TaskTracker 分组；
- 压缩算法：选择较好的方法提高压缩比、减少存储容量，同时选取高效率的算法以进行 shuffle 数据的压缩和解压；
- 对 mapper 程序和 reducer 程序使用的资源进行控制，防止过度消耗资源导致机器死机。以前是通过修改 Linux 内核来进行控制的，现在考虑通过在 Linux 中引入 cgroup 来对 mapper 和 reducer 使用的资源进行控制；
- 将 DataNode 的并发数据读写方式由多线程改为 select 方式，以支持大规模并发读写和 Hypertable 的应用。

百度同时也在使用 Hypertable，它是以 Google 发布的 BigTable 为基础的开源分布式数据存储系统，百度将它作为分析用户行为的平台，同时在元数据集中化、内存占用优化、集群安全停机、故障自动恢复等方面做了一些改进。

## 3.10.4 Hadoop 在 Facebook 的应用

Facebook 作为全球知名的社交网站，拥有超过 3 亿的活跃用户，其中约有 3 千万用户至少每天更新一次自己的状态；用户每月总共上传 10 亿余张照片、1 千万个视频；以及每周共享 10 亿条内容，包括日志、链接、新闻、微博等。因此 Facebook 需要存储和处理的数据量是非常巨大的，每天新增加 4TB 压缩后的数据，扫描 135TB 大小的数据，在集群上执

行 Hive 任务超过 7500 次，每小时需要进行 8 万次计算，所以高性能的云平台对 Facebook 来说是非常重要的，而 Facebook 主要将 Hadoop 平台用于日志处理、推荐系统和数据仓库等方面。

Facebook 将数据存储在利用 Hadoop/Hive 搭建的数据仓库上，这个数据仓库拥有 4800 个内核，具有 5.5PB 的存储量，每个节点可存储 12TB 大小的数据，同时，它还具有两层网络拓扑，如图 3-9 所示。Facebook 中的 MapReduce 集群是动态变化的，它基于负载情况和集群节点之间的配置信息可动态移动。

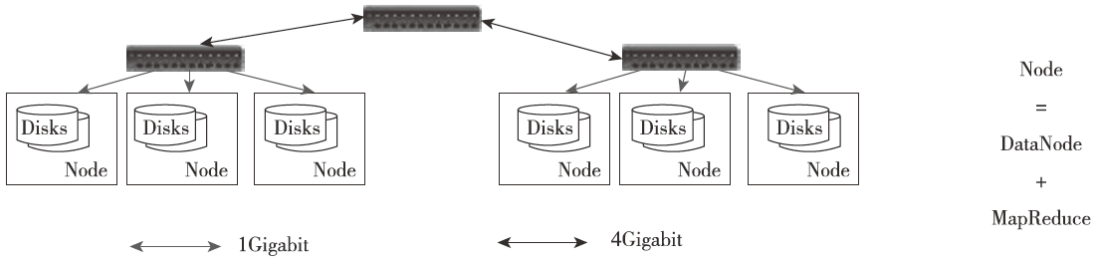


图 3-9 集群的网络拓扑

图 3-10 为 Facebook 的数据仓库架构，在这个架构中，网络服务器和内部服务生成日志数据，这里 Facebook 使用开源日志收集系统，它可以将数以百计的日志数据集存储在 NFS 服务器上，但大部分日志数据会复制到同一个中心的 HDFS 实例中，而 HDFS 存储的数据都会放到利用 Hive 构建的数据仓库中。Hive 提供了类 SQL 的语言来与 MapReduce 结合，创建并发布多种摘要和报告，以及在它们的基础上进行历史分析。Hive 上基于浏览器的接口允许用户执行 Hive 查询。Oracle 和 MySQL 数据库用来发布这些摘要，这些数据容量相对较小，但查询频率较高并需要实时响应。一些旧的数据需要及时归档，并存储在较便宜的存储器上，如图 3-11 所示。

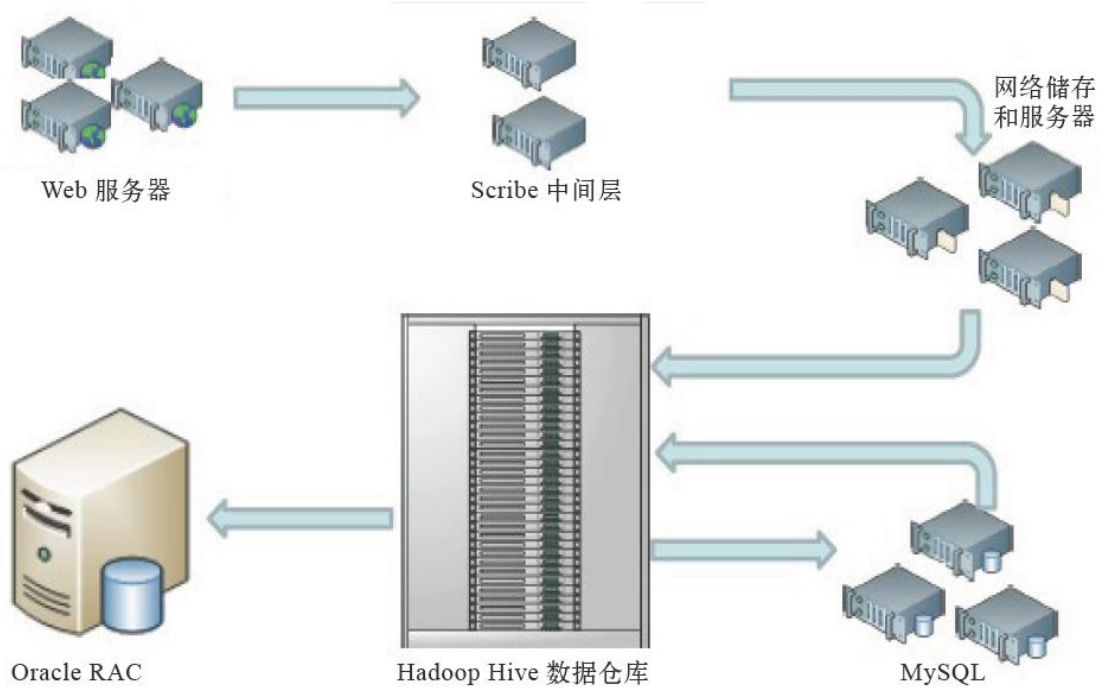


图 3-10 Facebook 数据仓库架构

下面介绍 Facebook 在 AvatarNode 和调度策略方面所做的一些工作。AvatarNode 主要用于 HDFS 的恢复和启动，若 HDFS 崩溃，原有技术恢复首先需要花 10～15 分钟来读取 12GB 的文件镜像并写回，还要用 20～30 分钟处理来自 2000 个 DataNode 的数据块报告，最后用 40～60 分钟来恢复崩溃的 NameNode 和部署软件。表 3-2 说明了 BackupNode 和 AvatarNode 的区别，AvatarNode 作为普通的 NameNode 启动，处理所有来自 DataNode 的消息。AvatarDataNode 与 DataNode 相似，支持多线程和针对多个主节点的多队列，但无法区分原始和备份。人工恢复使用 AvatarShell 命令行工具，AvatarShell 执行恢复操作并更新 ZooKeeper 的 zNode，恢复过程对用户来说是透明的。分布式 Avatar 文件系统实现在现有文件系统的上层。

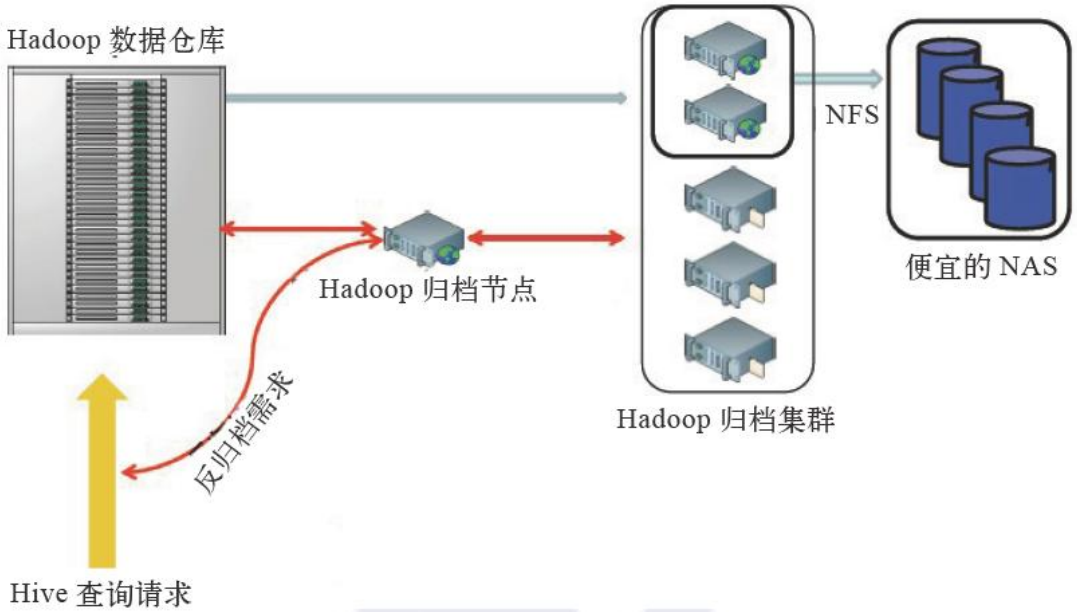


图 3-11 数据归档

表 3-2 BackupNode 和 AvatarNode 的区别

| BackupNode(冷备份) | AvatarNode(热备份) |
|---|---|
| Namespace 状态与原始的同步 | Namespace 状态与原始相比有几个事务的延迟 |
| 没有数据块和 DataNode 信息 | 拥有全部的数据块和 DataNode 信息 |
| 在可用之前仍需 20~30 分钟 | 6500 万个文件恢复不超过一分钟 |

基于位置的调度策略在实际应用中存在着一些问题：如需要高内存的任务可能会被分配给拥有低内存的 TaskTracker；CPU 资源有时未被充分利用；为不同硬件的 TaskTracker 进行配置也比较困难等。Facebook 采用基于资源的调度策略，即公平享有调度方法，实时监测系统并收集 CPU 和内存的使用情况，调度器会分析实时的内存消耗情况，然后在任务之间公平分配任务的内存使用量。它通过读取/proc/目录解析进程树，并收集进程树上所有的 CPU 和内存的使用信息，然后通过 TaskCounters 在心跳（heartbeat）时发送信息。

Facebook 的数据仓库使用 Hive，其构架如图 3-12 所示。这里 HDFS 支持三种文件格式：文本文件（TextFile），方便其他应用程序读写；顺序文件（SequenceFile），只有 Hadoop 能够读取并支持分块压缩；RCFile，使用顺序文件基于块的存储方式，每个块按列存储，这样有较好的压缩率和查询性能。Facebook 未来会在 Hive 上进行改进，以支持索引、视图、子查询等新功能。

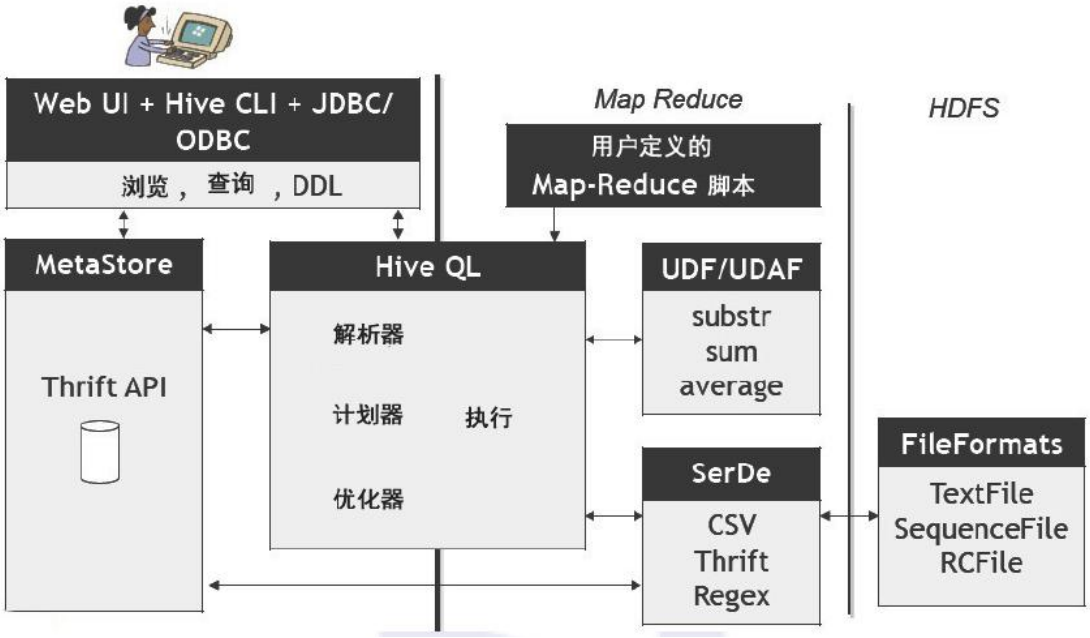


图 3-12 Hive 的体系结构

现在 Facebook 使用 Hadoop 遇到的挑战有：

- **服务质量和隔离性方面**：较大的任务会影响集群性能；
- **安全性方面**：如果软件漏洞导致 NameNode 事务日志崩溃该如何处理；
- **数据归档方面**：如何选择归档数据以及数据如何归档；
- **性能提升方面**：如何有效地解决瓶颈等。

## 3.11 Hadoop 平台上的海量数据排序

Yahoo!研究人员使用 Hadoop 完成了 Jim Gray 基准排序，此排序包含许多相关的基准，每个基准都有自己的规则。所有的排序基准都是通过测量不同记录的排序时间来制定的，每个记录为 100 字节，其中前面的 10 字节是键，剩余的部分是数值。MinuteSort 是比较在一分钟内所排序的数据量大小，GraySort 是比较在对大规模数据（至少 100TB）进行排序时的排序速率（TBs/minute）。基准规则具体如下：

- 输入数据必须与数据生成器生成的数据完全匹配；
- 任务开始的时候，输入数据不能在操作系统的文件缓存中。在 Linux 环境下，排序程序之间需要使用内存来交换其他内容；
- 输入和输出数据都是没有经过压缩的；
- 输出不能对输入进行重写；

- 输出文件必须存放到磁盘上；
- 必须计算输入和输出数据的每个键/值对的 CRC32，共 128 位校验和，当然，输入和输出必须对应相等；
- 输出如果分成多个输出文件，那么必须是完全有序的，也就是将这些输出文件连接以后必须是完全有序的输出；
- 开始和分布程序到集群上也要记入计算时间内；
- 任何抽样也要记入计算时间内。

Yahoo!的研究人员使用 Hadoop 排列 1TB 数据用时 62 秒，排列 1PB 数据用时 16.25 个小时，具体如表 3-3 所示，它获得了 Daytona 类 GraySort 和 MinuteSort 级别的冠军。

表 3-3 数据规模与排序时间

| 数据大小（Bytes） | 节点数 | 副本数 | 时间 |
|---|---|---|---|
| 500,000,000,000 | 1406 | 1 | 59 秒 |
| 1,000,000,000,000 | 1460 | 1 | 62 秒 |
| 100,000,000,000,000 | 3452 | 2 | 173 分钟 |
| 1,000,000,000,000,000 | 3658 | 2 | 957 分钟 |

下面的内容是根据基准排序的官方网站（http://sortbenchmark.org/）上有关使用 Hadoop 排序的相关内容整理而成。

Yahoo!的研究人员编写了三个 Hadoop 应用程序来进行 TB 级数据的排序：

- TeraGen 是产生数据的 map/reduce 程序；
- TeraSort 进行数据取样，并使用 map/reduce 对数据进行排序；
- TeraValidate 是用来验证输出数据是否有序的 map/reduce 程序。

TeraGen 用来产生数据，它将数据按行排列并且根据执行任务的数目为每个 map 分配任务，每个 map 任务产生所分配行数范围内的数据。最后，TeraGen 使用 1800 个任务产生总共 100 亿行的数据存储在 HDFS 上，每个存储块的大小为 512MB。

TeraSort 是标准的 map/reduce 排序程序，但这里使用的是不同的分配方法。程序中使用 N-1 个已排好序的抽样键值来为 reduce 任务分配排序数据的行数范围。例如，键值 key 在范围 sample[i-1]<=key<sample[i]的数据会分配给第 i 个 reduce 任务。这样就保证了第 i 个 reduce 任务输出的数据都比第 i+1 个 reduce 任务输出的数据小。为了加速分配过程，分配器在抽样键值上建立两层的索引结构树。TeraSort 在任务提交之前在输入数据中进行抽样，并将产生

的抽样数据写入 HDFS 中。这里规定了输入输出格式，使得三个应用程序可以正确地读取并写入数据。reduce 任务的副本数默认是 3，这里设置为 1，因为输出数据不需要复制到多个节点上。这里配置的任务为 1800 个 map 任务和 1800 个 reduce 任务，并为任务的栈设置了充足的空间，防止产生的中间数据溢出到磁盘上。抽样器使用 100000 个键值来决定 reduce 任务的边界，如图 3-13 所示，分布并不是很完美。

TeraValidate 保证输出数据是全部排好序的，它为输出目录的每个文件分配一个 map 任务（如图 3-10 所示），map 任务检查每个值是否大于等于前一个值，同时输出最大值和最小值给 reduce 任务，reduce 任务检查第 i 个文件的最小值是否大于第 i-1 文件的最大值，如果不是则产生错误报告。

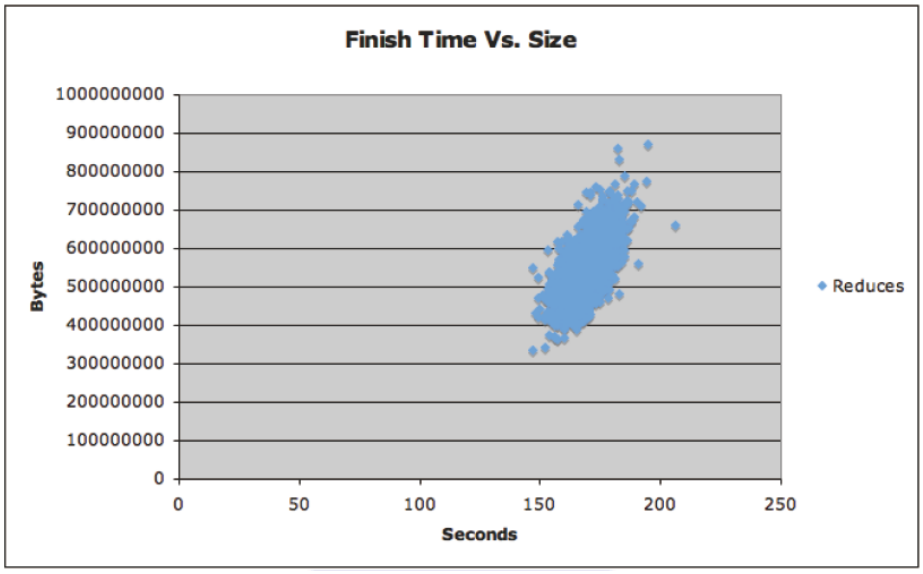


图 3-13 reduce 任务的输出大小和完成时间分布图

以上应用程序运行在雅虎搭建的集群上，其集群配置为：

- 910 个节点；
- 每个节点拥有 4 个英特尔双核 2.0GHz 至强处理器；
- 每个节点拥有 4 个 SATA 硬盘；
- 每个节点有 8GB 的内存；
- 每个节点有 1GB 的以太网带宽；
- 40 个节点一个 rack；
- 每个 rack 到核心有 8GB 的以太网带宽；
- 操作系统为 Red Hat Enterprise Linux Server Release 5.1(kernel 2.6.18) ；
- JDK 为 Sun Java JDK 1.6.0_05-b13。

整个排序过程在 209 秒（3.48 分钟）内完成，尽管拥有 910 个节点，但是网络核心是与其他 2000 个节点的集群共享的，所以运行时间会因为其他集群的活动而有所变化。

使用 Hadoop 进行 GraySort 基准排序时，Yahoo!的研究人员将上面的 map/reduce 应用程序稍加修改以适应新的规则，整个程序分为 4 个部分，分别为：

- TeraGen 是产生数据的 map/reduce 程序；
- TeraSort 进行数据取样，并使用 map/reduce 对数据进行排序；
- TeraSum 是 map/reduce 程序，用来计算每个键/值对的 CRC32，共 128 位校验和；
- TeraValidate 是用来验证输出数据是否有序的 map/reduce 程序，并且计算校验和的总和。

TeraGen 和 TeraSort 与上面介绍的一样，TeraValidate 除了增加了计算输出目录校验和总和的任务以外，其他都一样，这里不再赘述。

TeraSum 计算每个键/值对的 CRC32 的校验和，每个 map 任务计算输入的校验和并输出，然后一个 reduce 任务将每个 map 生成的校验和相加。这个程序用来计算输入目录下每个键/值对校验和的和，还用来检查排序输出后的正确性。

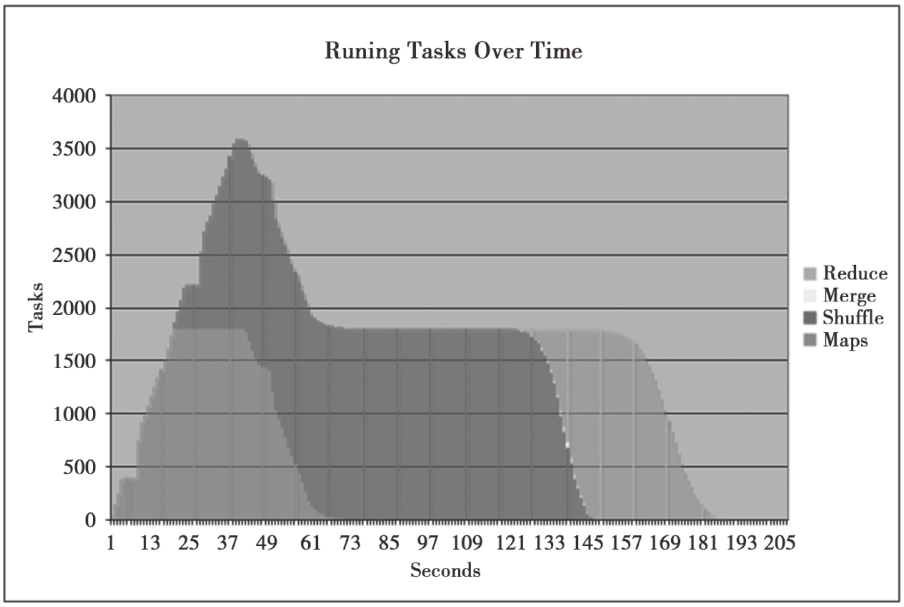


图 3-14 每个阶段的任务数

这次基准测试运行在 Yahoo!的 Hammer 集群上，集群的具体细节如下：

- 将近 3800 个节点（在这样大规模的集群中，一些节点会坏掉）；
- 每个节点两个双核 2.5GHz 的 Xeons 处理器；
- 每个节点 4 个 SATA 硬盘；
- 每个节点 8GB 内存（在 PB 级排序前会升级到 16GB）；

- 每个节点 1GB 的以太网带宽；
- 每个 rack 拥有 40 个节点；
- 每个节点到核心有 8GB 的以太网带宽；
- 操作系统为 Red Hat Enterprise Linux Server Realease 5.1（kernel 2.6.18）；
- JDK 为 Sun Java JDK（1.6.0 05-b13 and 1.6.0 13-b03）（32 and 64 bit）。

对于较大规模的排序，这里 NameNode 和 JobTracker 使用的是 64 位的 JVM。排序测试所用的 Hadoop 平台也做了一些变化，主要有：

- 重新实现了 Hadoop shuffle 阶段的 reducer 部分，在重新设计后提高了 shuffle 的性能，解除了瓶颈，而且代码也更容易维护和理解了；
- 新的 shuffle 过程从一个节点获取多个 map 的结果，而不是之前的一次只取一个结果。这样防止了多余的连接和传输开销；
- 允许配置 shuffle 连接的超时时间，在小规模排序时则可以将其减小，因为一些情况下 shuffle 会在超时时间到期后停止，这会增加任务的延迟时间；
- 设置 TCP 为无延迟并增加 TaskTracker 和 TaskTracker 之间 ping 的频率，以减少发现问题的延迟时间；
- 增加一些代码，用来检测从 shuffle 传输数据的正确性，防止引起 reduce 任务的失败。
- 在 map 输出的时候使用 LZO 压缩，LZO 能压缩 45%的数据量；
- 在 shuffle 阶段，在内存中将 map 的结果聚集输出的时候实现了 reduce 需要的内存到内存的聚集，这样减少了 reduce 运行时的工作量；
- 使用多线程实现抽样过程，并编写一个基于键值平均分布的较为简单的分配器；
- 在较小规模的集群上，配置系统在 TaskTracker 和 JobTracker 之间拥有较快的心跳频率，以减少延迟（默认为 10 秒/1000 节点，配置为 2 秒/1000 节点）；
- 默认的 JobTracker 按照先来先服务策略为 TaskTracker 分配任务，这种贪心的任务分配方法并不能很好地分布数据。从全局的角度来看，如果一次性为 map 分配好任务，系统会拥有较好的分布，但是为所有的 Hadoop 程序实现全局调度策略是非常困难的，这里只是实现了 TeraSort 的全局调度策略；
- Hadoop0.20 增加了安装和清除任务的功能，但是在排序基准测试里这并不需要，可以设置为不启动来减少开始和结束任务的延迟；
- 删除了框架中与较大任务无关的一些硬编码等待循环，因为它会增加任务延迟时

间；

- 允许为任务设置日志的级别，这样通过配置日志级别可以从 INFO 到 WARN 减少日志的内容，减少日志的内容对系统的性能有较大的提高，但是增加了调试和分析的困难；
- 优化任务分配代码，但还未完成。目前，对输入文件使用 RPC 请求到 NameNode 上会花费大量的时间。

Hadoop 与上面的测试相比有了很大的改进，可以在更短的时间内执行更多的任务。值得注意的是，在大集群和分布式应用程序中需要转移大量数据，这会导致执行时间有很大的变化。但是随着 Hadoop 的改进，它能够更好地处理硬件故障，这种时间变化也就微不足道了。不同规模的数据排序所需的时间如表 3-3 所示。

因为较小规模的数据需要更短的延迟和更快的网络，所以使用集群中的部分节点来进行计算。将较小规模计算的输出副本数设置为 1，因为整个过程较短且运行在较小的集群上，节点坏掉的可能性相对较小。而在较大规模的计算上，节点坏掉是难免的，于是将节点副本数设置为 2。HDFS 保证节点换掉后数据不会丢失，因为不同的副本放在不同的节点上。

Yahoo!的研究人员统计了 JobTracker 上从任务提交状况获得的任务数随时间的变化，图 3-15、图 3-16、图 3-17、图 3-18 显示了每个时间点下的任务数。maps 只有一个阶段，而 reduces 拥有三个阶段：shuffle、merge 和 reduce。shuffle 是从 maps 中转移数据的，merge 在测试中并不需要；reduce 阶段进行最后的聚集并写到 HDFS 上。如果将这些图与图 3-10 进行比较，你会发现建立任务的速度变快了。图 3-10 每次心跳建立一个任务，那么所有任务建立起来需要 40 秒，现在 Hadoop 每次心跳可以设置好一个 TaskTracker，可见减少任务建立的开销是非常重要的。

运行大规模数据时，数据传输的次数对任务性能的影响也是非常大的。在 PB 级数据排序中，每个 map 处理 15GB 的数据而不是默认的 128MB，每个 reduce 处理 50GB 的数据。如果按照 1.5GB/map 进行处理，需要 40 个小时才能完成。因此，为了增加吞吐量，增加每个块的大小是非常重要的。

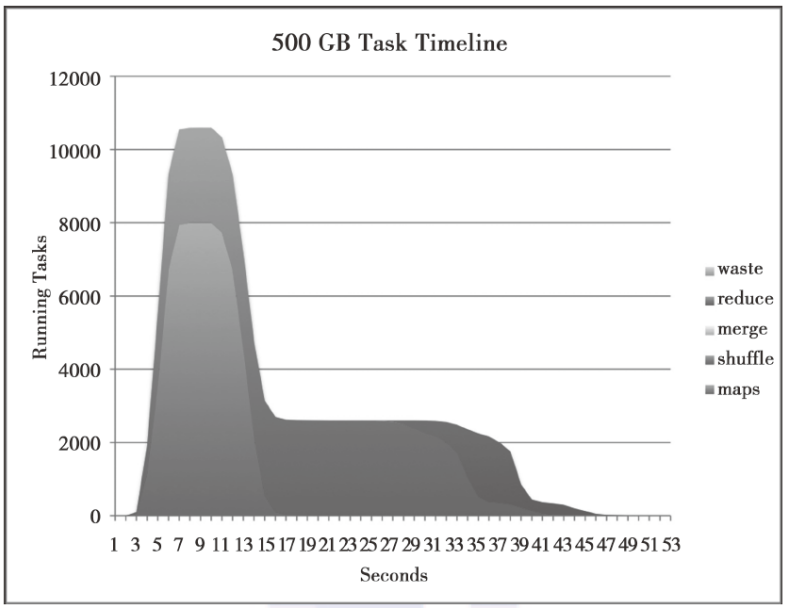


图 3-15 数据量为 500GB 时任务数随时间的变化

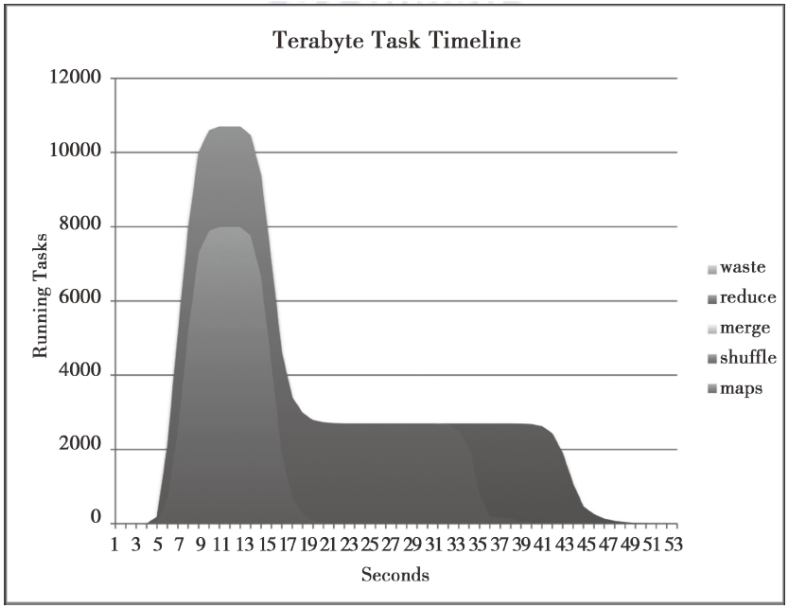


图 3-16 数据量为 1TB 时任务数随时间的变化

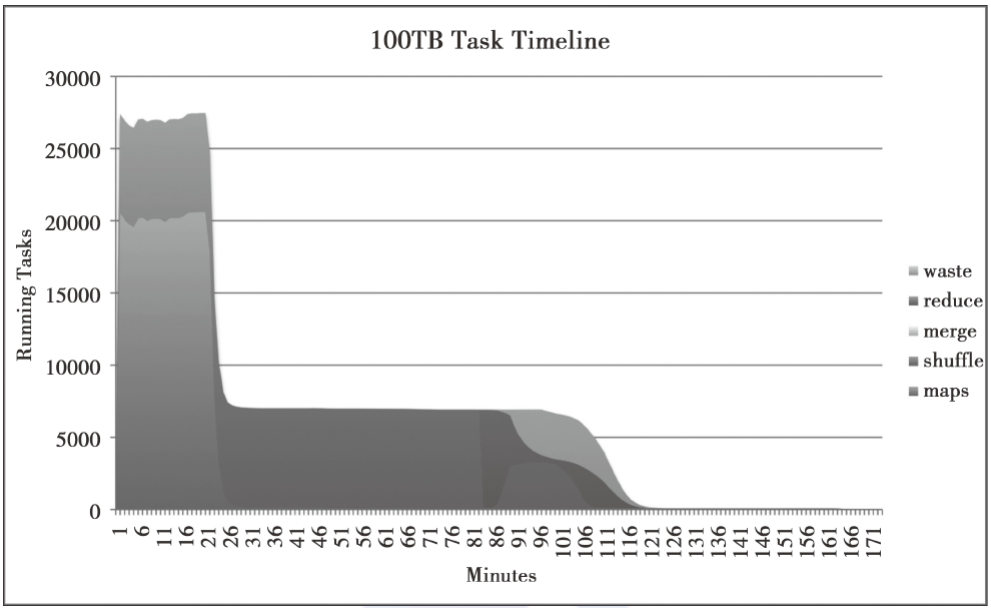


图 3-17 数据量为 100TB 时任务数随时间的变化

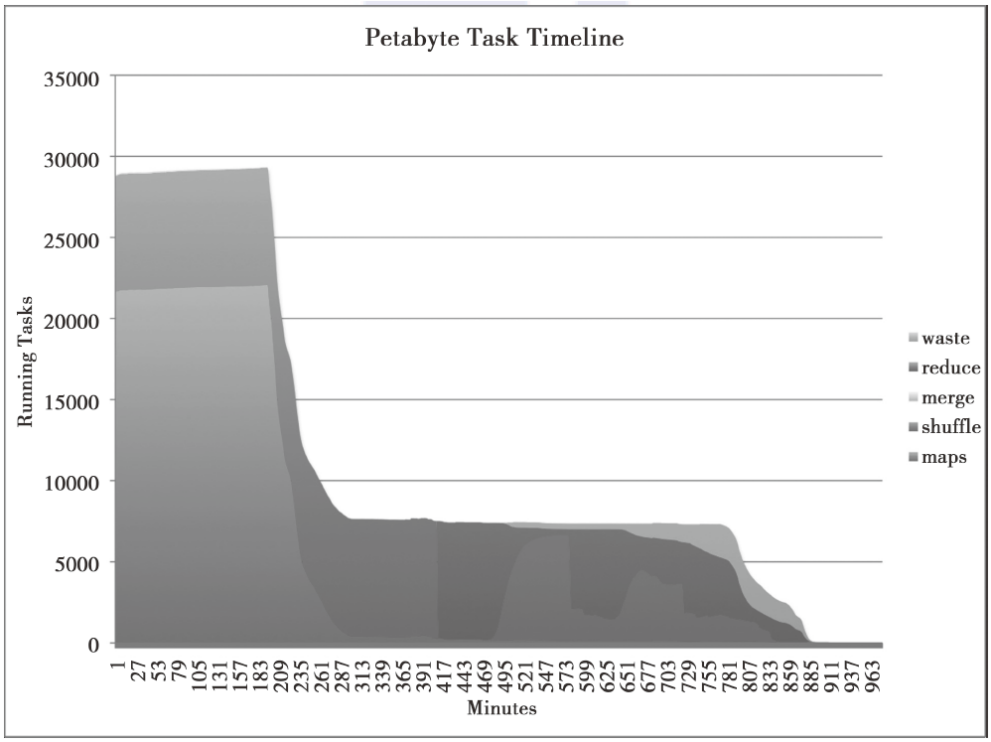


图 3-18 数据量为 1PB 时任务数随时间的变化

# 本章小结

本章首先介绍了 Hadoop 分布式计算平台：它是由 Apache 软件基金会开发的一个开源分布式计算平台。以 Hadoop 分布式文件系统（HDFS）和 MapReduce 为核心的 Hadoop 为用户提供了系统底层细节透明的分布式基础架构。由于 Hadoop 拥有可计量、成本低、高效、可信等突出特点，基于 Hadoop 的应用已经遍地开花，尤其是在互联网领域。

本章接下来介绍了 Hadoop 项目及其结构，现在 Hadoop 已经发展成为一个包含多个子项目的集合，被用于分布式计算，虽然 Hadoop 的核心是 Hadoop 分布式文件系统和 MapReduce，但 Hadoop 下的 Avro、Hive、HBase 等子项目提供了互补性服务或在核心层之上提供了更高层的服务；接下来简要介绍了以 HDFS 和 MapReduce 为核心的 Hadoop 体系结构。

最后，介绍了 Hadoop 的典型应用案例以及在 Hadoop 平台上的海量数据排序。

# 参考文献

[1] 陆嘉恒. Hadoop 实战. 机械工业出版社. 2011 年.

[2] 曾大聃, 周傲英(译). Hadoop 权威指南中文版. 清华大学出版社. 2010 年.

# 附录 1:任课教师介绍

林子雨(1978－),男,博士,厦门大学计算机科学系助理教授,主要研究领域为数据库,数据仓库,数据挖掘.

主讲课程：《大数据技术基础》

办公地点：厦门大学海韵园科研 2 号楼

E-mail: ziyulin@xmu.edu.cn

个人网页：http://www.cs.xmu.edu.cn/linziyu