初识Hadoop

> 在古时候，人们用牛来拉重物。当一头牛拉不动一根圆木时，人们从来没有考虑过要培育更强壮的牛。同理，我们也不该想方设法打造超级计算机，而应该千方百计综合利用更多计算机来解决问题。^
>
> ——格蕾斯·霍珀(Grace Hopper)

1.1 数据！数据！^

我们生活在这个数据大爆炸的时代，很难估算全球电子设备中存储的数据总共有多少。国际数据公司(IDC)曾经发布报告称，2006 年数字世界(digital universe)项目统计得出全球数据总量为 0.18 ZB 并预测在 2011 年将达到 1.8 ZB。[1] 1 ZB 等于 10^{21} 字节，等于 1000 EB(exabytes)，1 000 000 PB (petabytes)，等于大家更熟悉的 10 亿TB(terrabytes)！这相当于全世界每人一个硬盘中保存的数据总量！^

数据“洪流”有很多来源。以下面列出的为例：[2]^

- 纽约证交所每天产生的交易数据多达 1 TB^
- 脸谱网(Facebook)存储的照片约 100 亿张，存储容量约为 1 PB^
- 家谱网站 Ancestry.com 存储的数据约为 2.5 PB^
- 互联网档案馆(The Internet Archive)存储的数据约为 2 PB，并以每月至少 20 TB 的速度持续增长^
- 瑞士日内瓦附近的大型强子对撞机每年产生的数据约为 15 PB^

还有其他大量的数据。但是你可能会想它对自己又有哪些影响呢？地球人都知道，大部分数据都严密锁存在一些大型互联网公司(如搜索引擎公司)或科学机构与金融机构中。难道所谓的“大数据”只影响小机构和个人？^

我个人是这样认为的。以照片为例，我妻子的爷爷是一个骨灰级的摄影爱好者。在成年之后，他一直都在拍照。他的整个相册，包括普通胶片、幻灯片、35mm 胶片，在扫描成高分辨率的图片之后，大约有 10 GB。相比之下，在 2008 年，我家用数码相机拍摄的照片总共有 5 GB。对照爷爷的照片生成速度，我家是他老人家的 35 倍！并且，而且这个速度还在不断增长中，因为现在拍照片真的是越来越容易了。^

有一种情况更普遍，个人产生的数据正在快速增长。微软研究院的MyLifeBits 项目[3](*http://research.microsoft.com/enus/projects/mylifebits/ default.aspx*)显示，在不久的将来，个人

---

[1] Gantz 等人 2008 年 3 月发表的文章“The Diverse and Exploding Digital Universe”(纷繁多样并不断膨胀的数字世界)，网址为 *http://china.emc.com/collateral/ analyst-reports/expanding-digital-idc-white-paper.pdf*。^

[2] *http://www.intelligententerprise.com/showArticle.jhtml?articleID=207800705 ; http://mashable.com/2008/10/15/facebook-10-billion-photos/ ; http://blog.familytreemagazine.com/insider/Inside+Ancestrycoms+TopSecret+Data+Center.aspx ; http://www.archive.org/about/faqs.php ; http://www.interactions.org/cms/?pid=1027032*。^

[3] 编注：更多详细介绍可以参见阮一峰的博客文章“微软的 MyLifeBits 项目”，网址为

信息档案将日益普及。MyLifeBits的一个实验是获取和保存个人的对外联系情况(包括电话、邮件和文件)，供日后存取。收集的数据中包括每分钟拍摄的照片等，数据量每月约为 1 GB。当存储成本急剧下降以至于可以存储音频和视频时，MyLifeBits项目在未来的存储的数据量将是现在的很多倍。^

保存个人成长过程中产生的所有数据似乎逐渐成为主流，但更重要的是，计算机产生的数据可能远远超过我们个人所产生的。机器日志、RFID 检测仪、传感器网络、车载 GPS 和零售交易数据等——所有这些都将产生巨量的数据。^

在网上公开发布的数据也在逐年增加。组织或企业，要想在未来取得成功，不仅需要管理好自己的数据，更需要从其他组织或企业的数据中获取有价值的信息。^

这方面的先锋有 Amazon Web Services(*http://aws.amazon.com/publicdatasets*)、Infochimps.org(*http://infochimps.org/*)和theinfo.org(*http://theinfo.org*)，它们所发布的共享数据集，正在促进信息共享(information commons)，供所有人自由下载和分析 (或者只需要支付合理的价格通过AWS 平台来共享)。不同来源的信息在经过混搭和处理之后，会带来意外的效果和我们今天难以想象的应用。^

以 Astrometry.net(*http://astrometry.net*)为例，主要查看和分析 Flickr 网站上星空机器人小组所拍摄的星空照片。它对每一张照片进行分析并能辨别出它来自星空或其他天体(例如恒星和银河系等)的哪一部分。虽然这项研究尚处于试验阶段，但也表明如果可用的数据足够多(在本例中，为加有标签的图片数据)，通过它们而产生的后续应用也许会超乎这些拍照片的人最初的想象 (图片分析)。^

有句话说得好：“大数据胜于好算法。” 意思是说对于某些应用 (譬如根据以往的偏好来推荐电影和音乐)，不论算法有多牛，基于小数据的推荐效果往往都不如基于大量可用数据的一般算法的推荐效果。[4]^

现在，我们已经有了大量数据，这是个好消息。但不幸的是，我们必须想方设法好好地存储和分析这些数据。^

1.2 数据的存储与分析^

我们遇到的问题很简单：在硬盘存储容量多年来不断提升的同时，访问速度(硬盘数据读取速度)却没有与时俱进。1990 年，一个普通硬盘可以存储 1370 MB数据，传输速度为 4.4 MB/s[5]，因此只需要 5 分钟就可以读完整个硬盘中的数据。20 年过去了，1 TB的硬盘已然成为主流，但其数据传输速度约为 100 MB/s，读完整个硬盘中的数据至少得花 2.5 个小时。^

读完整个硬盘中的数据需要更长时间，写入数据就别提了。一个很简单的减少读取时间的办法是同时从多个硬盘上读数据。试想，如果我们有 100 个硬盘，每个硬盘存储 1%的数据，并行读取，那么不到两分钟就可以读完所有数据。^

仅使用硬盘容量的 1%似乎很浪费。但是我们可以存储 100 个数据集，每个数据集 1 TB，并

---

http://www.ruanyifeng.com/blog/2007/12/mylifebits.html。^

4 引自 Anand Rajaraman 发表的文章“Netflix Challenge”(Negfix 挑战大赛)，网址为 *http://anand.typepad.com/datawocky/2008/03more-data-usual.html*。在这个挑战大赛中，Netflix 公司公开自己的用户评分数据，让研究者根据这些数据对用户没有看过的电影预测评分，谁最先比现有系统好 10%，谁就能赢得 100 万美元的奖金。Alon Halevy，Peter Norvig(谷歌研究主管)和 Fernando Pereira 在他们的一篇文章中也提出了类似的观点，题为“The Unreasonable Effectiveness of Data”(数据的非理性效果)，发表于 *IEEE Intelligent Systems* 2009 年 3/4 月合刊。

5 这些规格对应的是希捷的 ST-41600n 硬盘。

实现共享硬盘的读取。可以想象，用户肯定很乐于通过硬盘共享来缩短数据分析时间；并且，从统计角度来看，用户的分析工作都是在不同时间点进行的，所以彼此之间的干扰并不太大。^

虽然如此，但要对多个硬盘中的数据并行进行读写数据，还有更多问题要解决。第一个需要解决的是硬件故障问题。一旦开始使用多个硬件，其中个别硬件就很有可能发生故障。为了避免数据丢失，最常见的做法是复制(replication)：系统保存数据的复本(replica)，一旦有系统发生故障，就可以使用另外保存的复本。例如，冗余硬盘阵列(RAID)就是按这个原理实现的，另外，Hadoop 的文件系统(HDFS，Hadoop Distributed FileSystem)也是一类，不过它采取的方法稍有不同，详见后文的描述。^

第二个问题是大多数分析任务需要以某种方式结合大部分数据来共同完成分析，即从一个硬盘读取的数据可能需要与从另外 99 个硬盘中读取的数据结合使用。各种分布式系统允许结合不同来源的数据进行分析，但保证其正确性是一个非常大的挑战。MapReduce 提出一个编程模型，该模型抽象出这些硬盘读写问题并将其转换为对一个数据集(由键值对组成)的计算。后文将详细讨论这个模型，这样的计算由 map 和 reduce 两部分组成，而且只有这两部分提供对外的接口。与 HDFS 类似，MapReduce 自身也有很高的可靠性。^

简而言之，Hadoop 为我们提供了一个可靠的共享存储和分析系统。HDFS 实现数据的存储，MapReduce 实现数据的分析和处理。虽然 Hadoop 还有其他功能，但 HDFS 和 MapReduce 是它的核心价值。^

1.3 相较于其他系统的优势^

MapReduce 看似采用了一种蛮力方法。每个查询需要处理整个数据集或至少一个数据集的绝大部分。但反过来想，这也正是它的能力。MapReduce 是一个批量查询处理器，能够在合理的时间范围内处理针对整个数据集的动态查询。它改变了我们对数据的传统看法，解放了以前只是保存在磁带和硬盘上的数据。它让我们有机会对数据进行创新。以前需要很长时间处理才能获得结果的问题，到现在变得顷刻之间就迎刃而解，同时还可以引发新的问题和新的见解。^

例如，Rackspace 公司的邮件部门 Mailtrust 就用 Hadoop 来处理邮件日志。他们写动态查询，想借此找出用户的地理分布。他们是这么描述的：“这些数据非常有用，我们每月运行一次 MapReduce 任务来帮助我们决定哪些 Rackspace 数据中心需要添加新的邮件服务器。” ^

通过整合好几百 GB 的数据，用 MapReduce 来分析这些数据，Rackspace 的工程师从中发现了以前从来没有注意到的数据，甚至还运用这些信息来改善了现有的服务。第 16 章将详细介绍 Rackspace 公司内部是如何使用 Hadoop 的。^

1.3.1 关系型数据库管理系统^

为什么不能用数据库来对大量硬盘上的大规模数据进行批量分析呢？我们为什么需要 MapReduce？^

这两个问题的答案来自于计算机硬盘的另一个发展趋势：寻址时间的提升远远不敌于传输速率的提升。寻址是将磁头移动到特定硬盘位置进行读写操作的过程。它是导致硬盘操作延迟的主要原因，而传输速率取决于硬盘的带宽。

如果数据访问模式中包含大量的硬盘寻址，那么读取大量数据集就必然会花更长的时间(相较于流数据读取模式，流读取主要取决于传输速率)。另一方面，如果数据库系统只更新一小部分记录，那么传统的 B 树就更有优势(关系型数据库中使用的一种数据结构，受限于寻

址的比例)。但数据库系统如果有大量数据更新时，B 树的效率就明显落后于 MapReduce，因为需要使用“排序/合并“(sort/merge)来重建数据库。^

在许多情况下，可以将 MapReduce 视为关系型数据库管理系统的补充。两个系统之间的差异如表 1-1 所示。^

MapReduce 比较适合以批处理方式处理需要分析整个数据集的问题，尤其是动态分析。RDBMS 适用于点查询 (point query)和更新，数据集被索引之后，数据库系统能够提供低延迟的数据检索和快速的少量数据更新。MapReduce 适合一次写入、多次读取数据的应用，关系型数据库则更适合持续更新的数据集。^

表 1-1. 关系型数据库和 MapReduce 的比较^

	传统的关系型数据库	MapReduce
数据大小	GB	PB
数据存取	交互式和批处理	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
完整性	高	低
横向扩展	非线性的	线性的

MapReduce 和关系型数据库之间的另一个区别在于它们所操作的数据集的结构化程度。**结构化数据**(structured data)是具有既定格式的实体化数据，如 XML 文档或满足特定预定义格式的数据库表。这是 RDBMS 包括的内容。另一方面，**半结构化数据**(semi-structured data)比较松散，虽然可能有格式，但经常被忽略，所以它只能作为对数据结构的一般性指导。例如电子表格，它在结构上是由单元格组成的网格，但是每个单元格内可以保存任何形式的数据。**非结构化数据**(unstructured data)没有什么特别的内部结构，例如纯文本或图像数据。MapReduce 对非结构化或半结构化数据非常有效，因为它是在处理数据时才对数据进行解释。换句话说，MapReduce 输入的键和值并不是数据固有的属性，而是由分析数据的人来选的。^

关系型数据往往是**规范的**(normalized)，以保持其数据的完整性且不含冗余。规范给 MapReduce 带来了问题，因为它使记录读取成为非本地操作，而 MapReduce 的核心假设之一偏偏就是可以进行(高速的)流读写操作。^

Web 服务器日志是典型的非规范化数据记录(例如，每次都需要记录客户端主机全名，这会导致同一客户端的全名可能多次出现)，这也是 MapReduce 非常适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编程模型。程序员要写两个函数，分别为 map 函数和 reduce 函数，每个函数定义从一个键值对集合到另一个键值对集合的映射。这些函数不必关注数据集及其所用集群的大小，可以原封不动地应用于小规模数据集或大规模的数据集。更重要的是，如果输入的数据量是原来的两倍，那么运行时间也需要两倍。但如果集群是原来的两倍，作业的运行速度却仍然与原来一样快。SQL 查询一般不具备该特性。^

但是，在不久的将来，关系型数据库系统和MapReduce系统之间的差异很可能变得模糊。关系型数据库都开始吸收MapReduce的一些思路(如Aster Data的数据库和GreenPlum的数据库)，另一方面，基于MapReduce的高级查询语言(如Pig和Hive)使传统数据库的程序员更容易接受MapReduce系统。[6]^

[6] 2007 年 1 月，数据库理论专家 David J. DeWitt 和 Michael Stonebraker 发表的论文引发一场激烈的口水大

1.3.2 网格计算^

高性能计算(High Performance Computing，HPC)和网格计算(Grid Computing)组织多年以来一直在研究大规模数据处理，主要使用类似于消息传递接口(Message Passing Interface，MPI)的 API。从广义上讲，高性能计算采用的方法是将作业分散到集群的各台机器上，这些机器访问存储区域网络(SAN)所组成的共享文件系统。这比较适用于计算密集型的作业，但如果节点需要访问的数据量更庞大 (高达几百 GB，MapReduce 开始施展它的魔法)，很多计算节点就会因为网络带宽的瓶颈问题不得不闲下来等数据。^

MapReduc尽量在计算节点上存储数据，以实现数据的本地快速访问。[7]数据本地化(data locality)特性是MapReduce的核心特征，并因此而获得良好的性能。意识到网络带宽是数据中心环境最珍贵的资源(到处复制数据很容易耗尽网络带宽)之后，MapReduce通过显式网络拓扑结构来保留网络带宽。注意，这种排列方式并没有降低MapReduce对计算密集型数据进行分析的能力。^

虽然 MPI 赋予程序员很大的控制权，但需要程序员显式控制数据流机制，包括用 C 语言构造底层的功能模块(例如套接字)和高层的数据分析算法。而 MapReduce 则在更高层次上执行任务，即程序员仅从键值对函数的角度考虑任务的执行，而且数据流是隐含的。^

在大规模分布式计算环境下，协调各个进程的执行是一个很大的挑战。最困难的是合理处理系统的部分失效问题——在不知道一个远程进程是否挂了的情况下——同时还需要继续完成整个计算。有了MapReduce，程序员不必操心系统部分失效的问题，因为它自己的系统实现能够检测到并重新执行那些失败的map或reduce任务。正因为采用的是无共享(shared-nothing)框架，MapReduce才能够实现失败检测，这意味着各个任务之间是彼此独立的。[8]因此，从程序员的角度来看，任务的执行顺序无关紧要。相比之下，MPI程序必须显式管理自己的检查点和恢复机制，虽然赋予程序员的控制权加大了，但编程的难度也增加了。^

MapReduce 听起来似乎是一个相当严格的编程模型，而且在某种意义上看的确如此：限定用户使用有特定关联的键值对，mapper 和 reducer 彼此间的协调非常有限(每个 mapper 将键值对传给 reducer)。由此，我们自然联想到一个问题：能用这个编程模型做一些有用或实际的事情吗？^

答案是肯定的。MapReduce由谷歌的工程师开发，用于构建搜索引擎的索引，而且，事实已

战，论文标题为“MapReduce: A major step backwards”(MapReduce：一个巨大的倒退)，原文可参见 *http://databasecolumn.vertica.com/ database-innovation/mapreduce a-major-step-backwards*，中文版可参考 *http://wap. oschina.net/question/17793_31108*)。在文中，他们认为 MapReduce 不宜取代关系型数据库。许多评论认为这是一种错误的比较，详情可参见 Mark C. Chu-Carroll 的文章“Databasesare hammers; MapReduce is a screwdriver”(如果说数据库是锤子，MapReduce 则是螺丝刀)，英文版网址为 *http://scienceblogs.com/ goodmath/2008/01 databases_are_hammers_mapreduc.php*，*中文版可以参考 http://blog.csdn.net/ wanghai__/article/details/5954108*。DeWitt 与 Stonebraker 以“再说 MapReduce”一文对其他人的观点进行了阐述，原文可参见 *http://databasecolumn. vertica.com/database-innovation/mapreduce-ii*，他们对其他人的主要观点进行了阐述。

[7] 1998 年图灵奖得主 Jim Gray 在 2003 年 3 月发表的“Distributed Computing Economics”(分布式计算经济学)一文中，率先提出这个结论：数据处理应该在离数据本身比较近的地方进行，因为这样有利于降低成本，尤其是网络带宽消费所造成的成本。原文网址为 *http://research.microsoft.com/apps/pubs/default.aspx?id=70001*。

[8] 这里讲得太简单了一点，因为 MapReduce 系统本身控制着 mapper 输出结果传给 reducer 的过程，所以在这种情况下，重新运行 reducer 比重新运行 mapper 更要小心，因为 reducer 需要获取必要的 mapper 输出结果，如果没有，必须再次运行对应的 mapper，重新生成输出结果。

经证明它能够一次又一次地解决这个问题(MapReduce 的灵感来自于传统的函数式编程、分布式计算和数据库社区)，但此后，该模型在其他行业还有着很多其他的应用。我们欣喜地发现，有很多算法都可以用 MapReduce来表达，从图像图形分析到各种各样基于图像分析的问题，再到机器学习算法。[9]当然，它也不是包治百病的灵丹妙药，不能解决所有问题，但它真的是一个很通用的数据处理工具。^

我们将在第 16 章介绍 Hadoop 的一些典型应用。^

[9] Apache Mahout(*http://mahout.apache.org/*)是一个在 Hadoop 上运行的机器学习类库(例如分类和聚类算法)。

1.3.3 志愿计算^

人们第一次听说Hadoop和MapReduce的时候，经常会问这个问题：“它们和SETI@home有什么不同？”SETI全称为Search for Extra-Terrestrial Intelligence(搜索外星智能)，项目名称为SETI@home(*http://setiathome.berkeley.edu*)。在该项目中，志愿者把自己计算机CPU的空闲时间贡献出来分析无线天文望远镜的数据，借此寻找外星智慧生命信号。SETI@home因为拥有庞大的志愿者队伍而非常出名，其他还有“搜索大素数”(Great Internet Mersenne Prime Search)项目与Folding@home项目(了解蛋白质构成及其与疾病之间的关系)。^

志愿计算项目将问题分成很多块，每一块称为一个**工作单元**(work unit)，发到世界各地的计算机上进行分析。例如，SETI@home的工作单元是0.35 MB无线电望远镜数据，要对这等大小的数据量进行分析，一台普通计算机需要几个小时或几天时间才能完成。完成分析后，结果发送回服务器，客户端随后再获得另一个工作单元。为防止欺骗，每个工作单元要发送到3台不同的机器上执行，而且收到的结果中至少有两个相同才会被接受。^

从表面上看，SETI@home与MapReduce好像差不多(将问题分解为独立的小块，然后并行进行计算)，但事实上还是有很多明显的差异。SETI@home问题是CPU高度密集的，比较适合在全球成千上万台计算机上运行，[10]因为计算所花的时间远远超过工作单元数据的传输时间。也就是说，志愿者贡献的是CPU周期，而不是网络带宽。^

MapReduce有三大设计目标：(1)为只需要短短几分钟或几个小时就可以完成的作业提供服务；(2)运行于同一个内部有高速网络连接的数据中心内；(3)数据中心内的计算机都是可靠的、定制的硬件。相比之下，SETI@home则是在接入互联网的不可信的计算机上长时间运行，这些计算机的网络带宽不同，对数据本地化也没有要求。^

---

[10] 2008年1月，SETI@home发表评论说每天使用320 000台计算机处理300 GB数据，同时他们也在做其他的一些数据计算，原文参见*http://www.planetary.org/programs/ projects/setiathome/setiathome_20080115*。

1.4 Hadoop发展简史^

Hadoop 是 Apache Lucene 创始人 Doug Cutting 创建的，Lucene 是一个应用广泛的文本搜索系统库。Hadoop 起源于开源的网络搜索引擎 Apache Nutch，它本身也是 Lucene 项目的一部分。^

Hadoop 的得名

Hadoop 不是缩写，它是一个生造出来的词。Hadoop 之父 Doug Cutting 这样解释 Hadoop 的来历：

> “这个名字是我的小孩给他的毛绒象玩具取的。我的命名标准是好拼读，含义宽泛，不会被用于其他地方。小孩子是这方面的高手。Googol 就是小孩子起的名字。”

Hadoop的子项目及后续模块所使用的名称也往往与其功能不相关，通常也以大象或其他动物为主题取名(例如Pig)。较小一些的组件，名称通常都有较好的描述性(因此也更流俗)。这个原则很好，意味着我们可以望文知义，例如jobtracker[11]，一看就知道它是用来跟踪MapReduce作业的。

从头打造一个网络搜索引擎是一个雄心勃勃的计划，不只是因为写爬虫程序很复杂，更因为必须有一个专职团队来实现——项目中包含许许多多需要随时修改的活动部件。同时，构建这样的系统代价非常高——据Mike Cafarella和Doug Cutting估计，一个支持 10 亿网页的索引系统，单是硬件上的投入就高达 50 万美元，另外还有每月高达 3 万美元的运维费用。[12]不过，他们认为这个工作仍然值得投入，因为它开创的是一个优化搜索引擎算法的平台。^

Nutch项目开始于 2002 年，一个可以运行的网页爬取工具和搜索引擎系统很快面试。但后来，开发者认为这一架构的灵活性不够，不足以解决数十亿网页的搜索问题。一篇发表于 2003 年的论文为此提供了帮助，文中描述的是谷歌产品架构，该架构称为“谷歌分布式文件系统”，简称GFS。[13]GFS或类似的架构，可以解决他们在网页爬取和索引过程中产生的超大文件的存储需求。特别关键的是，GFS能够节省系统管理(如管理存储节点)所花的大量时间。在 2004 年，他们开始着手做开源版本的实现，即Nutch分布式文件系统(NDFS)。^

2004 年，谷歌发表论文向全世界介绍他们的MapReduce系统。[14]2005 年初，Nutch的开发人员在Nutch上实现了一个MapReduce系统，到年中，Nutch的所有主要算法均完成移植，用MapReduce和NDFS来运行。^

Nutch的NDFS和MapReduce实现不只适用于搜索领域。在 2006 年 2 月，开发人员将NDFS和MapReduce移出Nutch形成Lucene的一个子项目，命名为Hadoop。大约在同一时间，Doug Cutting加入雅虎，雅虎为此组织了专门的团队和资源，将Hadoop发展成能够处理Web数据的

---

[11] 在本书中我们使用小写的 jobtracker 来代表实体(泛称)，用驼峰体 JobTracker 来表示对 Java 类的实现。

[12] Mike Cafarella 和 Doug Cutting 在 2004 年 4 月发表在 ACM Queue 上的文章“Building Nutch: Open Source Search”，网址为 *http://queue.acm.org/detail.cfm?id=988408*。

[13] Sanjay Ghemawat ,Howard Gobioff 和 Shun-Tak Leung 在 2003 年 10 月发表的文章“The Google File System”，网址为 *http://labs.google.com/papers/gfs.html*。

[14] 参见 Jeffrey Dean 和 Sanjay Ghemawat 2004 年 12 月发表的文章“MapReduce: Simplified Data Processing on Large Clusters”(MapReduce: 大型集群的数据简化处理)，网址为 *http://labs.google.com/papers mapreduce.html*。

系统(参见后面的补充材料“Hadoop在雅虎“)。在 2008 年 2 月，雅虎宣布，雅虎搜索引擎使用的索引是在一个拥有 1 万个内核的Hadoop集群上构建的。[15]^

2008 年 1 月，Hadoop 已成为 Apache 的顶级项目，证明了它的成功、多样化和生命力。到目前为止，除雅虎之外，还有很多公司在用 Hadoop，例如 Last.fm、Facebook 和《纽约时报》等。第 16 章和 Hadoop 维基页面(英文)介绍了一些案例(*http://wiki.apache.org/hadoop/PoweredBy*)。^

《纽约时报》的案例广为流传，他们把 1851 年到 1980 年的存档扫描之后得到 4 TB的文件并用亚马逊的EC2 云服务将文件存为PDF格式放到网上共享。[16]整个过程一共使用了 100 台计算机，所花的时间不到 24 小时。如果没有亚马逊的按小时付费模式(即允许《纽约时报》短期内访问大量机器)和Hadoop好用的并发编程模型珠联璧合，这个项目不太可能这么快就启动和完成。^

2008 年 4 月，Hadoop打破世界纪录，成为最快的TB级数据排序系统。在一个 910 节点的群集，Hadoop在 209 秒内(不到 3.5 分钟)完成了对 1TB数据的排序，击败了前一年的 297 秒冠军(详情参见 15.5 节的补充材料“Apache Hadoop的TB级数据处理”)。同年 11 月，谷歌在报告中声称，它的MapReduce对 1 TB数据排序只用了 68 秒。[17]2009 年 5 月本书第 1 版出版的时候，有报道称雅虎有一个的团队使用 Hadoop对 1 TB数据进行排序只花了 62 秒。^

从那以后，Hadoop跃升为企业主流的部署系统。在工业界，Hadoop已经是公认的大数据通用存储和分析平台，这一事实主要体现在大量直接使用或者间接辅助Hadoop系统的产品如雨后春笋般大量涌现。一些大公司也发布Hadoop发行版本，包括EMC，IBM，Microsft和Oracle以及一些专注于Hadoop的公司，如Cloudera，Hortonworks[18]和MapR。^

Hadoop 在雅虎^

作者：Owen O’Melly

^

构建互联网规模的搜索引擎离不开大量的数据，因此也离不开大量的机器来处理巨量的数据。雅虎搜索引擎(Yahoo！Search)有 4 个主要组成部分：Crawler，从网页服务器爬取网页；WebMap，构建一个已知网页的链接图；Indexer，为最佳页面构建一个反向索引；Runtime，处理用户的查询。WebMap 生成的链接图非常大，大约包括一万亿(10^{12})条边(每条边代表一个网页链接)和一千亿(10^{11})个节点(每个节点代表不同的网址)。创建并分析如此大的图需要大批计算机很多天长时间运行。到 2005 年初，WebMap 用的底层架构 Dreadnaught 需要重新设计以便日后扩展到更多的节点。^

Dreadnaught 从 20 个节点成功扩展到 600 个，但需要完全重新设计才能进一步扩大。Dreadnaught 与 MapReduce 在很多方面都很相似，但灵活性更强，结构也更松散。说具体点，一个 Dreadnaught 作业的每一个片断(fragment，也称“分

---

[15] 参见 2008 年 2 月 19 日发表的文章“雅虎发布全球最大的 Hadoop 产品应用”(Yahoo! Lauches World’s Largest Hadoop ProductionApplications)，网址为 *http://developer. yahoo.com/blogs/hadoop/posts/2008/02/yahoo-worlds-largest-production-hadoop/*。

[16] 参见 Derek Gottfrid 在 2007 年 11 月 1 日发表的文章“Self-service, Prorated Super Computing Fun!”(自助式比例分配超级计算的乐趣！)，网址为 *http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/*。

[17] 全文参见 2008 年 11 月 21 日的文章“Sorting 1PB with MapReduce”(MapReduce 处理 1 PB 数据)，网址为 *http://googleblog.blogspot.com/2008/11/sorting-1pb-with-mapreduce.html*。

[18] 编者注：该公司是雅虎的几个核心开发人员创办的，主要提供 Hadoop 支持和咨询服务，他们已经与微软在 2011 年建立战略合作关系，帮助微软将 Hadoop 移植到 Wiondows Server 和 Azure。

块”)都可以输送到下一阶段的各个片段继续执行，排序则是通过库函数来完成的。但实际情形是，大多数 WebMap 阶段是两两一对，对应于 MapReduce。因此，WebMap 应用不需要做大量重构操作就可以适应 MapReduce。

Eric Baldeschwieler(Eric14)组建了一个小团队，于是我们开始设计并在 GFS 和 MapReduce 上用 C++来建立一个新框架的原型，并打算用它来取代 Dreadnaught。尽管我们的当务之急是需要一个新的 WebMap 框架，但更清楚的是，建立雅虎搜索引擎批处理平台的标准对我们更重要。使平台更通用以便支持其他用户，才能够更好地实现新平台的均衡性投资。

与此同时，我们也关注在 Hadoop(当时也是 Nutch 的一部分)及其进展情况。2006 年 1 月，雅虎聘请了 Doug Cutting。一个月后，我们决定放弃原型，转而采用 Hadoop。与我们的原型和设计相比，Hadoop 的优势在于它已经在 20 个节点上实际应用过(Nutch)。这样一来，我们便能在两个月内搭建一个研究集群并能够以很快的速度帮助我们的客户使用这个新的框架。另一个显著的优点是 Hadoop 已经开源，比较容易(尽管也不是想象的那么容易！)从雅虎法务部门获得许可对该开源系统进行进一步研究。因此，我们在 2006 年初建立了一个 200 节点的研究集群并暂时搁置 WebMap 计划，转而为研究用户提供 Hadoop 支持和优化服务。

Hadoop 大事记

2004 年	Doug Cutting 和 Mike Cafarella 实现了 HDFS 和 MapReduce 的初版
2005 年 12 月	Nutch 移植到新框架，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，支持 MapReduce 和 HDFS 独立发展
2006 年 2 月	雅虎的网格计算团队采用 Hadoop
2006 年 4 月	在 188 个节点上(每节点 10 GB)运行排序测试集需要 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	在 500 个节点上运行排序测试集需要 42 个小时(硬件配置比 4 月份的更好)
2006 年 11 月	研究集群增加到 600 个节点

2006 年 12 月	排序测试集在 20 个节点上运行 1.8 个小时，100 个节点上运行 3.3 小时，500 个节点上运行 5.2 小时，900 个节点上运行 7.8 个小时
2007 年 1 月	研究集群增加到 900 个节点
2007 年 4 月	研究集群增加到两个集群 1000 个节点
2008 年 4 月	在 900 个节点上运行 1 TB 排序测试集仅需 209 秒，成为全球最快
2008 年 10 月	研究集群每天装载 10 TB 的数据
2009 年 3 月	17 个集群共 24 000 个节点
2009 年 4 月	在每分钟排序中胜出，59 秒内排序 500 GB(在 1400 个节点上)和 173 分钟内排序 100 TB 数据(在 3400 个节点上)

书摘与插图（插图或海报）：