这是一个信息爆炸的时代，互联网上的信息正在以几何级数的速度增长。在这个大背景下，消耗 CPU 最多的计算逐渐从“提升软件本身性能”方面转移到信息处理方面。与此同时，摩尔定律似乎也不再像以前那么准确地发挥作用了。在这样的严峻形势下，各大厂商面临着极大的挑战——他们需要从 TB 乃至 PB 级的数据中挖掘出有用的信息，并对这些海量的数据进行快捷、高效的处理。在这段特殊时期，Google 公司以 MapReduce 为基石，结合 GFS、Bigtable 逐步发展成为全球互联网企业的领头羊。然而，出于技术保密的原因，Google 公司并没有开源其 MapReduce 的实现细节，这使得人们无法深入了解和认识它。就在这时，一头神秘的大象——Hadoop 从天而降，它的开源给人们带来了新的希望。

1.1 初识神象

要认识 Hadoop 这头神象，便不得不提它的“父亲”Doug Cutting，以及它的“近亲”Nutch 和 Lucene。“父亲”Doug Cutting 本人不仅是 Nutch 的创始人，还是 Lucene 项目的创始人，他就像希腊神话中的“盗火人”普罗米修斯，把高深莫测的搜索技术形成产品贡献给普罗大众。“近亲”Lucene 是一个 Java 高性能全文索引引擎工具包，它可以方便地嵌入到各种实际应用中实现全文索引搜索功能。而 Nutch 则是一个应用程序，它是一个以 Lucene 为基础实现的搜索引擎应用，Lucene 为 Nutch 提供了文本搜索和索引的 API。与此同时，Nutch 不仅提供了搜索功能，而且还提供了数据抓取的功能。此外，Nutch 中还包含一个分布式文件系统用于存储数据。

2003 和 2004 年，Google 先后发表了《The Google File System》、《MapReduce: Simplified Data Processing on Large Clusters》两篇论文。这两个创新性的思路点燃了 Doug Cutting 和 Mike Cafarella 的斗志，他们花了 2 年的业余时间实现了 DFS（分布式文件系统）和 MapReduce 机制。这次改

造使 Nutch 可以在 20 台机器上支持几亿的数据规模，其编程和运维的简易性也得到了大幅提升，但系统的吞吐能力与一个真正的网页搜索系统仍有不小的差距。2006 年是开源社区如火如荼的一年，当 Yahoo！在考虑构建一个高度利用硬件资源、并且维护和开发都非常简易的软件架构时，Doug Cutting 和他的“孩子”Nutch 进入了 Yahoo！的视野。Doug Cutting 和 Yahoo!各有千秋，一方具有超强的技术前瞻性和实战经验，而另一方能提供世界上数一数二的数据、硬件和人力资源。就这样，双方一拍即合，在同年 1 月 Doug Cutting 正式加入了 Yahoo！，2 月 Hadoop 这头小象从 Nutch 中分离出来，并且正式成为 Apache 组织中一个专注于 DFS 和 MapReduce 的开源项目。作为一个分布式系统基础架构，Hadoop 使用户可以在不了解分布式底层细节的情况下，充分利用集群的威力，开发分布式程序，实现高速运算和存储。

Hadoop 这个名字[5]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个棕黄色的填充大象玩具命名的。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。”

五年的时间里，昔日的小象历练成了一头日渐成熟和强大的神象。2007 年 1 月 Hadoop 单 cluster 集群已达 900 个节点；2008 年 4 月[1]，依托拥有 910 个节点的群集，Hadoop 在 209 秒内完成了对 1TB 数据的排序，击败了前一年的 297 秒冠军；2009 年 4 月，取得了在 1400 个节点的集群上对 500G 数据排序仅用时 59 秒的成绩。如今的它不仅仅致力于应对网络流量和科学研究，而且还涉猎搜索引擎、广告优化、机器学习等领域，并成为 IT 产业里优秀的大数据平台。

1.2 Hadoop 初体验

1.2.1 了解 Hadoop 的构架

Hadoop 是一个能够对大量数据进行分布式处理的软件框架，实现了 Google 的 MapReduce 编程模型和框架，能够把应用程序分割成许多小的工作单元，并把这些单元放到任何集群节点上执行。在 MapReduce 中，一个准备提交执行的应用程序称为“作业（job）”，而从一个作业划分出的、运行于各计算节点的工作单元称为“任务（task）”。此外，Hadoop 提供的分布式文件系统（HDFS）主要负责各个节点上的数据存储，并实现了高吞吐率的数据读写。

在分布式存储和分布式计算方面，Hadoop 都使用主/从（Master/Slave）架构。在一个配置完整的集群上想让 Hadoop 这头大象奔跑起来，需要在集群中运行一系列后台（deamon）程序。不同的后台程序扮演不同的角色，这些角色由 NameNode、DataNode、Secondary NameNode、JobTracker、TaskTracker 组成。其中 NameNode、Secondary NameNode、JobTracker 运行在 Master 节点上（如图 1-1 所示），而在每个 Slave 节点上，部署一个 DataNode 和 TaskTracker，以便这个 Slave 服务器上运行的数据处理程序能尽可能直接处理本机的数据。对 Master 节点需要特别说明的是，在小集群中，Secondary NameNode 可以属于某个从节点；在大型集群中，NameNode 和 JobTracker 被分别部署在两台服务器上。

下面介绍各个角色在 Hadoop 中的作用[3]。

1．NameNode

NameNode 是 HDFS 的守护程序，负责记录文件是如何分割成数据块

的，以及这些数据块分别被存储到哪些数据节点上。它的主要功能是对内存及 I/O 进行集中管理。

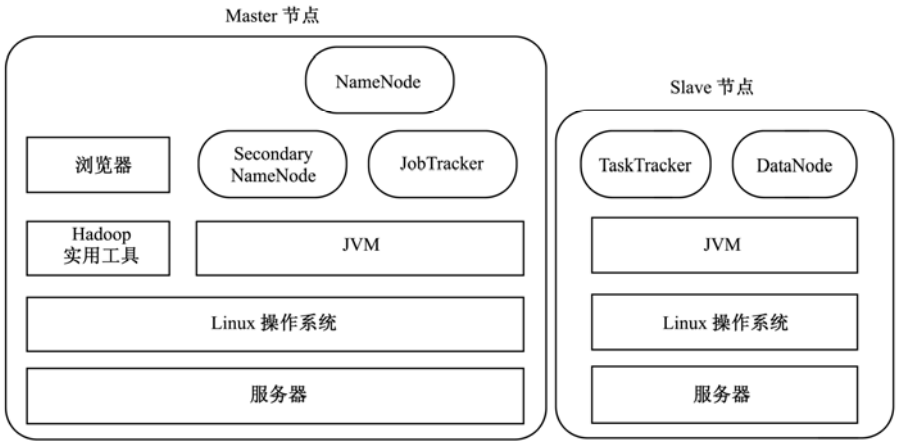

图 1-1　Hadoop 主从节点结构分解

一般来说，NameNode 所在的服务器不存储任何用户信息或执行计算任务，以避免这些程序降低服务器的性能。如果其它从服务器因出现软硬件的问题而宕机，Hadoop 集群仍旧可以继续运转，或者快速重启。但是，由于 NameNode 是 Hadoop 集群中的一个单点，一旦 NameNode 服务器宕机，整个系统将无法运行。

2．DataNode

集群中的每个从服务器都运行一个 DataNode 后台程序，这个后台程序负责把 HDFS 数据块读写到本地的文件系统。当需要通过客户端读/写某个数据时，先由 NameNode 告诉客户端去哪个 DataNode 进行具体的读/写操作，然后，客户端直接与这个 DataNode 服务器上的后台程序进行通信，并且对相关的数据块进行读/写操作。

3．Secondary NameNode

Secondary NameNode 是一个用来监控 HDFS 状态的辅助后台程序。就像 NameNode 一样，每个集群都有一个 Secondary NameNode，并且部署在一台单独的服务器上。Secondary NameNode 不同于 NameNode，它不接收或记录任何实时的数据变化，但是，它会与 NameNode 进行通信，以便定期地保存 HDFS 元数据的快照。由于 NameNode 是单点的，通过 Secondary NameNode 的快照功能，可以将 NameNode 的宕机时间和数据损失降低到最小。同时，如果 NameNode 发生问题，Secondary NameNode 可以及时地作为备用 NameNode 使用。

4．JobTracker

JobTracker 后台程序用来连接应用程序与 Hadoop。用户代码提交到集群以后，由 JobTracker 决定哪个文件将被处理，并且为不同的 task 分配节点。同时，它还监控所有运行的 task，一旦某个 task 失败了，JobTracker 就会自动重新开启这个 task，在大多数情况下这个 task 会被放在不同的节点上。当然，具体运行情况取决于重启的预设值。每个 Hadoop 集群只有一个 JobTracker，一般运行在集群的 Master 节点上。

5．TaskTracker

TaskTracker 与负责存储数据的 DataNode 相结合，其处理结构上也遵循主/从架构。JobTracker 位于主节点，统领 MapReduce 工作；而 TaskTrackers 位于从节点，独立管理各自的 task。每个 TaskTracker 负责独立执行具体的 task，而 JobTracker 负责分配 task。虽然每个从节点上仅有唯一的一个 TaskTracker，但是每个 TaskTracker 可以产生多个 Java 虚拟机（JVM），用于并行处理多个 map 以及 reduce 任务。TaskTracker 的一个重要职责就是与 JobTracker 交互。如果 JobTracker 无法准时地获取 TaskTracker 提交的信息，JobTracker 就判定 TaskTracker 已经崩溃，并将任务分配给其他节点处理。

1.2.2　查看 Hadoop 活动

通过“http://（Master node 的主机名/IP）:50030”命令链接到 JobTracker，可以直观地查看 Hadoop 执行任务时的情况。

1．JobTracker 页面

图 1-2 给出通过 http://UbuntuMaster:50030 访问 JobTracker 主页的截图。

页面最上方给出了 Hadoop 的安装细节，读者可以了解 Hadoop 的版本号、编译时间以及 JobTracker 的当前状态（在本例中，状态是 running）和启动时间。

集群信息（Cluster Summary）部分提供的是关于集群的概要信息。包括：当前正在集群上运行的 map 和 reduce 的数量、作业提交的数量、集群的负载状况、集群中可用的 map 和 reduce 任务数（“Map Task Capacity”和“Reduce Task Capacity”），每个节点平均可用的任务数、列入黑名单的 TaskTrackers 数等信息。调度信息（Scheduling Information）显示正在运行的作业调度器的相关信息（此处是“默认值”），单击“default”可以查看作业队列（显示的是正在运行、完成和失败的作业）。如果任务还没有完成，那么每部分都会有一个作业表，表里每行显示出作业的 ID、作业拥有者、作业名（使用 JobConf 的 setJobName()方法设置的 mapred.job.name 属性）以及进度信息。接下来的三部分依次给出了有关作业运行（Running Jobs）、结束（Completed Jobs）以及失败（Failed Jobs）的信息，读者可通过这三部分了解作业运行状态。

页面的最后给出了一些链接信息，它们指向了 JobTracker 日志以及 JobTracker 历史信息，这些信息是永久存储的。

2．作业页面

单击图 1-2 中的作业 ID，可以进入如图 1-3 所示的作业页面。页面最上

方是作业的摘要，包括：作业的拥有者、作业名和作业运行时间。作业文件是整理过的作业配置文件，包括作业运行中有效的所有属性和值。如果不确定某个属性的值，可以单击该属性查看文件。

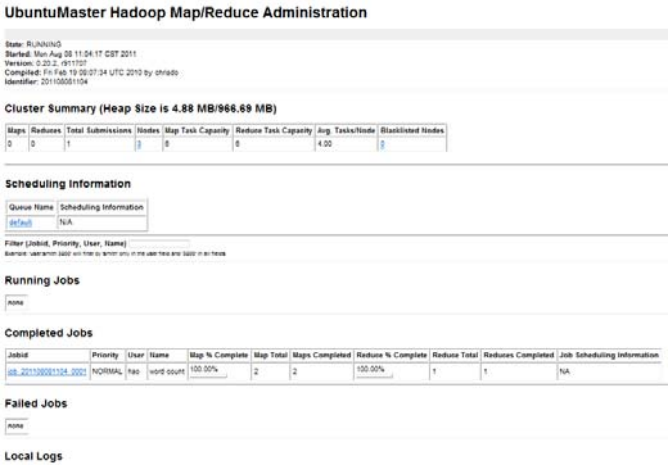

图 1-2　JobTracker 页面

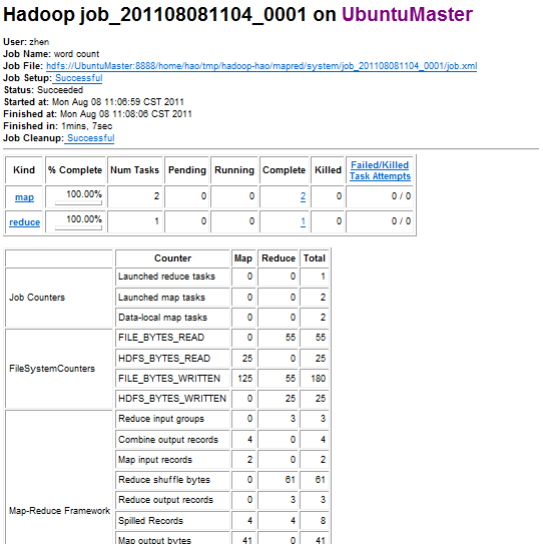

图 1-3　作业页面

当作业在运行时，可通过作业页面监视作业进度，页面信息会定期自动更新。摘要信息下方的表格展示 map 和 reduce 进度。Num Tasks 显示该作业 map 和 reduce 的总数（图中的作业示例执行了两个 map 任务和一个 reduce 任务）。其他列显示的是这些任务的状态：Pending（挂起）、Running（运行）、Complete（完成）和 Killed（失败）。最后一列显示的是一个作业所有 map 和 reduce 任务中失败和中止的任务尝试（Task Attempt）总数（Task Attempt 标记为 Killed 的原因可能是：它们是推测执行的副本，Task Attempt 运行的 Task Tracker 已结束，或这些 Task Attempt 已被用户中止）。

该页的中间部分是用来给作业计数的表。在作业运行时，表中的这些信息都会定期地动态更新。

该页面的后半部分显示每个任务进度的完成图（如图 1-4 所示）。其中，Reduce 完成图可以分为三个阶段：copy（map 输出的数据传输到 reduce 的 TaskTracker 需要通过 copy 来实现）、sort（在 reduce 输入时，对数据进行 sort）以及 reduce（reduce 函数运行产生最后输出结果）。

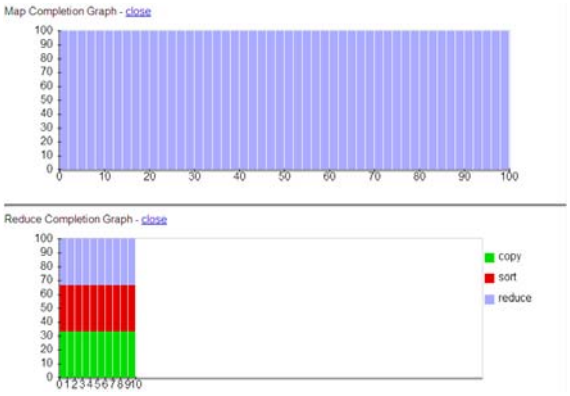

图 1-4　任务进度图

此外，通过“http://（Master node 的主机名/IP）:50070”可以链接到 NameNode，对 HDFS 进行检测。另外，在 NameNode 和 JobTracker Web 页

面内，还含有大量链接，可以获取更多有关 Hadoop 配置和操作的其它信息（包括运行日志），读者需要时可以自行查看更多的细节。

1.3 Hadoop 族群

Hadoop 之所以在本书中称为神象并不仅仅因为它自身很强大，更重要的是以它为代表的整个 Hadoop 族群都不可小觑！下面简单介绍一下 Hadoop 的家族成员。

1．Hadoop 子项目

Hadoop Common：就是原来的 Hadoop Core，是整个 Hadoop 项目的核心部分，为 Hadoop 各子项目提供各种工具，比如配置文件和日志操作等。其他的 Hadoop 子项目都是在 Hadoop Common 的基础上发展的。

HDFS：向应用程序提供高吞吐量访问的分布式文件系统，是 GFS 的开源实现（详见第 2 章介绍）。

MapReduce：大型数据的分布式并行编程模型和程序执行框架，Google 的 MapReduce 的开源实现（详见第 3 章介绍）。

2．Hadoop 相关项目

Avro：Doug Cutting 主持的 RPC 项目，有点类似于 Google 的 protobuf 和 Facebook 的 thrift。它作为 Hadoop 的 RPC(远程过程调用模块)，使 Hadoop 的 RPC 模块通信速度更快，数据结构更紧凑。

Cassandra：是一套开源分布式 NoSQL 数据库系统。它最初由 Facebook 开发，用于储存收件箱等简单格式数据，集 Google BigTable 的数据模型与

Amazon Dynamo 的完全分布式的架构于一身（详见第 8 章介绍）。

Chukwa：一个基于 Hadoop 用来管理大型分布式系统的数据采集系统（详见第 9 章介绍）。

Hama：为科学计算提供的一个基于整体同步并行计算技术的分布式计算框架。

HBase：是 Apache Hadoop 项目的一个重要部分，是一个开源的、基于列存储模型的分布式数据库（详见第 4 章介绍）。

Hive：提供数据摘要和查询功能的数据仓库（详见第 6 章介绍）。

Pig：是在 MapReduce 上构建的一种高级的数据流语言，是 Sawzall 的开源实现。Sawzall 是一种建立在 MapReduce 基础上的领域语言，它的程序控制结构（如 if、while 等）与 C 语言无异，但它的领域语言语义使它完成相同功能的代码比 MapReduce 的 C++代码简洁得多（详见第 7 章介绍）。

ZooKeeper：用于解决分布式系统中一致性问题，是 Chubby 的开源实现（详见第 10 章介绍）。