# 云端的小飞象—Hadoop

Hadoop是一个可以更容易开发和并行处理大规模数据的分布式计算平台，未来几年内它可能具有与Google系统架构技术相同的竞争力。

■ 文 / 孙牧

## Hadoop简史

在搜索技术界，也许有人不熟悉Doug Cutting，但很少有人不知道Lucene这个著名的全文检索引擎。事实上，Lucene应该是Doug Cutting的成名作，它被广泛地应用在各种规模的网站和系统中，甚至Eclipse中的搜索功能也是Lucene来实现的。

但Doug Cutting并没有满足Lucene取得的成绩。2002年，他发起了一个基于Lucene的开源项目Nutch，其目标是构建出一个包括网络蜘蛛、文件存储等模块的网页搜索系统。经过2年的努力，Nutch虽然可以用4台机器支持1亿网页的抓取和检索，但系统的扩展性开始遇到瓶颈。恰在此时，Google发表了GFS、MapReduce的论文，这两个创新性的思路点燃了Nutch 2名开发人员的斗志，他们又花了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，开源社区如火如荼，当美国雅虎在思索构建一个高度利用硬件资源、维护和开发都非常简易的软件架构时，Doug Cutting和他的Nutch进入了他们的视野。一方具有超强的技术前瞻性和实战经验，另一方能提供世界上数一数二的数据、硬件和人力资源，双方一拍即合，同年1月Doug Cutting正式加入雅虎，2月Hadoop从Nutch中分离出来，正式成为Apache组织中一个专注于DFS和MapReduce的开源项目。

2008年2月，又是两年，雅虎宣布搭建出一个世界上最大的基于Hadoop的生产集群系统—Yahoo! Search Webmap（简单地讲，就是雅虎网页搜索抓取的所有站点和网页及其关系的数据库），下面一组数据可以让我们对该系统的规模有个初步的认识：

- 页面之间的链接数超过1000亿；
- Webmap输出的压缩数据超过300TB（Terabyte）；
- 有单一的MapReduce任务同时在1万多个CPU的核（core）上运行；
- 生产集群硬盘空间占用超过5PB（Petabyte）；
- 与原来没用Hadoop的方案相比节约了30%的时间。

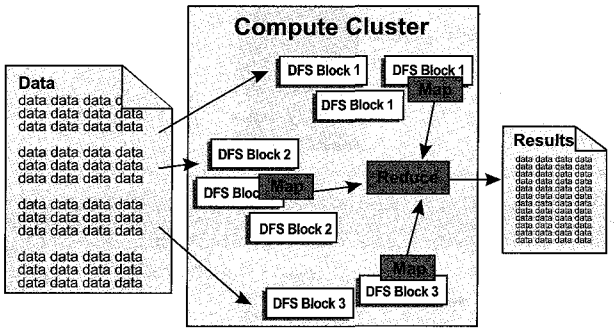
这时候，可以说Doug Cutting想构建一个Web-scale级别系统的心愿也终于实现了！

## Hadoop的系统架构

简单地讲，Hadoop是一个可以更容易开发和并行处理大规模数据的分布式计算平台。它的主要特点是：扩容能力（Scalable）、成本低（Economical）、高效率（Efficient）、可靠性（Reliable）。另外，Hadoop是一款完全用Java开发的开源软件，因此它可以运行在多种操作系统和商用硬件上。

Hadoop主要由两部分构成：Hadoop分布式文件系统（HDFS）和MapReduce的实现。

HDFS和MapReduce的关系如下图所示。




MapReduce是依赖于HDFS实现的。通常MapReduce会将被计算的数据分为很多小块，HDFS会将每个块复制若干份以确保系统的可靠性，同时它按照一定的规则将数据块放置在集群中的不同机器上，以便MapReduce在数据宿主机器上进行最便捷的计算。

下面我们再深入一些看看HDFS和MapReduce的实现细节：

## HDFS

HDFS设计时基于如下的前提和目标：

1 **硬件错误是常态而不是异常**：HDFS可能由成百上千的服务器所构成，每个服务器上存储着文件系统的部分数据。任一组件都有可能失效，这意味着总是有一部分HDFS的组件是不工作的。因此错误检测和快速、自动的恢复是HDFS最核心的架构目标。

2 **流式数据访问**：HDFS的设计中更多地考虑到了数据批处理，而不是用户交互处理。比之数据访问的低延迟问题，更关键的在于数据访问的高吞吐量。

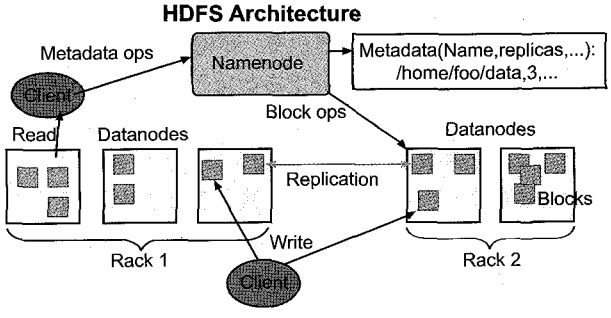
3 **大规模数据集**：HDFS上的一个典型文件大小一般都在G字节至T字节。因此，HDFS被调节以支持大文件存储，并能提供整体上高的数据传输带宽，能在一个集群里扩展到数百个节点。一个单一的HDFS实例应该能支撑数以千万计的文件。

4 **简单的一致性模型**：HDFS应用需要一个“一次写入多次读取”的文件访问模型。文件经过创建、写入和关闭之后就不需要改变。这一假设简化了数据一致性问题，并且使高吞吐量的数据访问成为可能。MapReduce应用或者网络爬虫应用都非常适合这个模型。

5 **移动计算比移动数据更划算**：一个应用请求的计算，离它操作的数据越近就越高效，在数据达到海量级别的时候更是如此。因为这样就能降低网络阻塞的影响，提高系统数据的吞吐量。HDFS为应用提供了将计算移动到数据附近的接口。

6 **异构软硬件平台间的可移植性**：这种特性方便了HDFS作为大规模数据应用平台的推广。

HDFS的系统架构如下图所示。




HDFS采用Master/Slave架构，一个HDFS集群是由一个Namenode和一定数目的Datanodes组成。Namenode是一个中心服务器，负责管理文件系统的名字空间（Namespace）以及客户端对文件的访问。集群中的Datanode一般是一个节点一个，负责管理它所在节点上的存储。HDFS暴露了文件系统的名字空间，用户能够以文件的形式在上面存储数据。

从内部看，一个文件其实被分成一个或多个数据块（Block），这些块存储在一组Datanode上。Namenode执行文件系统的名字空间操作，比如打开、关闭、重命名文件或目录，它也负责确定数据块到具体Datanode节点的映射。Datanode负责处理文件系统客户端的读写请求，在Namenode的统一调度下进行数据块的创建、删除和复制。

单一节点的Namenode大大简化了系统的架构。Namenode负责保管和管理所有的HDFS元数据（Metadata），因而用户数据就不需要通过Namenode（也就是说文件数据的读写是直接在Datanode上）。

## MapReduce

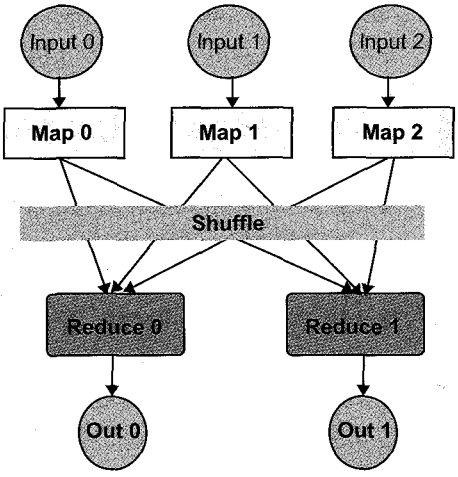
MapReduce是一种高效的分布式编程模型，同时是一种用于处理和生成大规模数据集的实现方式。其实，现实世界中很多计算任务都可以用这个模型来表达，熟悉Unix Shell的同学一定写过类似这样的命令行：

```
-> cat input | grep xxx | sort | uniq -c | cat > output
```

上面每个管道符中间正好对应了一个典型MapReduce的几个阶段：

```
Input | Map | Shuffle & Sort | Reduce | Output
```

下图表明了这几个阶段的工作流及结构关系。




1 Input：一个Hadoop MapReduce应用通常需要提供一对通过实现合适的接口或抽象类提供的Map和Reduce函数，还应该指明输入/输出的位置（路径）和其他一些运行参数。此外，此阶段还会把输入目录下的大数据文件切分为若干独立的数据块。

2 Map：MapReduce框架把应用作业的输入看为是一组<key, value>键值对，在Map这个阶段，框架会调用用户自定义的Map函数处理每一个<key, value>键值对，生成一批新的中间<key, value>键值对，这两组键值对的类型可能不同。

3 Shuffle & Sort：为了保证Reduce的输入是Map排好序的输出。在Shuffle阶段，框架通过HTTP为每个Reduce获得所有Map输出中与之相关的<key, value>键值对；而在Sort阶段，框架将按照key的值对Reduce的输入进行分组（因为不同map的输出中可能会有相同的key）。通常Shuffle和Sort两个阶段是同时进行的，Reduce的输入也是一边被取回，一边被合并的。

4 Reduce：此阶段会遍历中间数据，对每一个唯一key，执行用户自定义的Reduce函数（输入参数是<key, (list