Hadoop 云计算技术手册

序　言

Hadoop是一个开源的分布式并行计算平台，它主要由MapReduce的算法执行和一个分布式的文件系统等两部分组成。

Hadoop起源于Doug Cutting大牛领导开发的Nutch搜索引擎项目的子项目。现在是Apache软件基金会管理的开源项目。

本文主要介绍Hadoop及相关技术，从Hadoop的起源开始讲述，主要涵盖了MapReduce算法思想，基本框架，运行流程和编程粒度等内容，以期给入门者提供一个关于Hadoop的技术简介和研究参考。关于Hadoop的安装指南和编程范例并不在本文叙述范围内，有需要者请参考其它资料。

因笔者水平实在太有限了，文中如有疏漏错误请不吝指出，万分感谢。

本人资料多数来源于互联网的技术文档，附录列出引文列表，特此致谢原文作者。

最后，发自内心、无与伦比地感谢Google、Apache软件基金会和Doug Cutting带给我们如此简约、优雅的技术。

OK，让我们开始吧！去寻找那神奇的小飞象。

目 录

1. 引言——Hadoop 从何而来

自从Google工程师Jeffrey Dean提出MapReduce编程思想，MapReduce便在Google的各种Web应用中释放着魔力。然而，也许出于技术保密的目的，Google公司并没有透露其MapReduce的实现细节。

幸运的是，Doug Cutting开发的Hadoop作为MapReduce开源实现，让MapReduce这么平易近人地走到了我们面前。2006年1月，Doug Cutting因其在开源项目Nutch和Lucene的卓越表现受邀加入Yahoo公司，专职在Hadoop项目上进行开发。现在，Doug Cutting大牛已经加盟Cloudera（一家从事Hadoop产品商业化及技术支持的公司）。

注：Hadoop 名称的来历——Hadoop 原本是小 Doug Cutting 的大象玩具。

作为Google MapReduce技术的开源实现，Hadoop理所当然地借鉴了Google的Google File System文件系统、MapReduce并行算法以及BigTable。因此，Hadoop也是一个能够分布式处理大规模海量数据的软件框架，这一点不足为奇。当然，这一切都是在可靠、高效、可扩展的基础上。Hadoop的可靠性——因为Hadoop假设计算元素和存储会出现故障，因为它维护多个工作数据副本，在出现故障时可以对失败的节点重新分布处理。Hadoop的高效性——在MapReduce

的思想下，Hadoop是并行工作的，以加快任务处理速度。Hadoop的可扩展

依赖于部署Hadoop软件框架计算集群的规模，Hadoop的运算是可扩展的，具有处理PB级数据的能力。

虽然Hadoop自身由Java语言开发，但它除了使用Java语言进行编程外，同样支持多种编程语言，如C++。

Hadoop的长期目标是提供世界级的分布式计算工具，也是对下一代业务（如搜索结果分析等）提供支持的Web扩展（web-scale）服务。

2. 算法思想——Hadoop 是怎么思考的

MapReduce 主要反映了映射和规约两个概念，分别完成映射操作和规约操作。映射操作按照需求操作独立元素组里面的每个元素，这个操作是独立的，然后新建一个元素组保存刚生成的中间结果。因为元素组之间是独立的，所以映射操作基本上是高度并行的。规约操作对一个元素组的元素进行合适的归并。虽然有可能规约操作不如映射操作并行度那么高，但是求得一个简单答案，大规模的运行仍然可能相对独立，所以规约操作也有高度并行的可能。

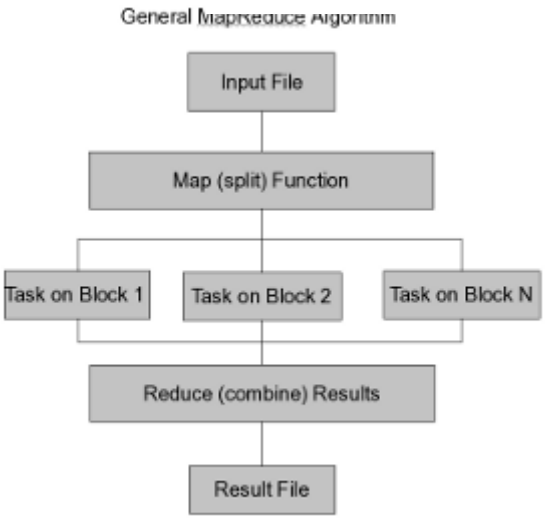


图 1

MapReduce 把数据集的大规模操作分配到网络互联的若干节点上进行 ,以实现其可靠性；每个节点都会向主节点发送心跳信息，周期性地把执行进度和状态报告回来。假如某个节点的心跳信息停止发送，或者超过预定时隙，主节点标记该节点为死亡状态，并把先前分配到它的数据发送到其它节点。其中，每个操作使用命名文件的原子操作，避免并行线程之间冲突；当文件被改名时，系统可能会把它复制到任务名以外的其它名字节点上。

由于规约操作的并行能力较弱，主节点尽可能把规约操作调度在同一个节点上，或者距离操作数据最近（或次近，最近节点出现故障时）的节点上。

MapReduce 技术的优势在于对映射和规约操作的合理抽象 ,使得程序员在编写大规模分布式并行应用程序时，几乎不用考虑计算节点群的可靠性和扩展性等问题。

应用程序开发人员把精力集中在应用程序本身，关于集群的处理问题等交由 MapReduce 完成。

3. 基本架构——Hadoop 是如何构成的

Hadoop 主要由 HDFS（Hadoop Distributed File System）和 MapReduce 引擎两部分组成。最底部是 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。

3.1 HDFS

HDFS 可以执行的操作有创建、删除、移动或重命名文件等，架构类似于传统的分级文件系统。需要注意的是，HDFS 的架构基于一组特定的节点而构建（参见图 2），这是它自身的特点。HDFS 包括 唯一的 NameNode，它在 HDFS 内部提供元数据服务；DataNode 为 HDFS 提供存储块。由于 NameNode 是唯一的，这也是 HDFS 的一个弱点（单点失败）。一旦 NameNode 故障，后果可想而知。正所谓皮之不存，毛将焉附？

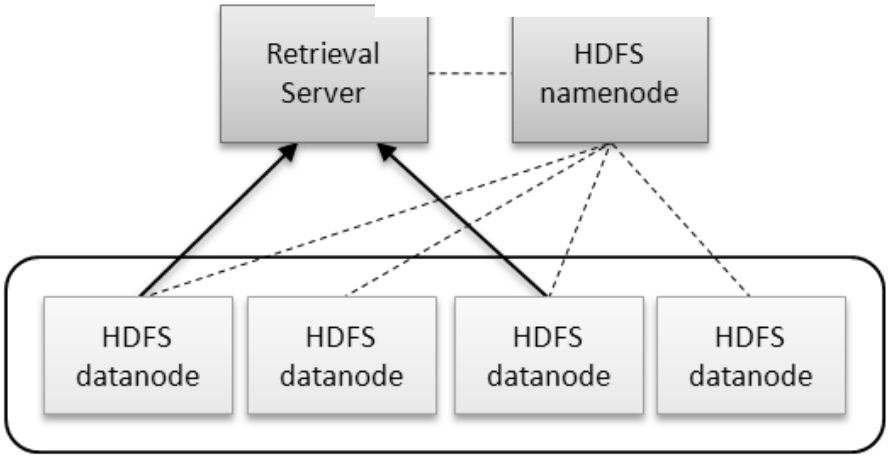


图 2

NameNode 可以控制所有文件操作。HDFS 中存储文件被分割成块，这些块被复制到多个节点中（DataNode）。块的大小（默认为 64MB）和复制的块数量在创建文件时由客户机决定。

HDFS 内部的所有通信都基于标准的 TCP/IP 协议。

NameNode

NameNode 负责管理文件系统名称空间和控制外部客户机的访问。NameNode 决定是否将文件映射到 DataNode 上的复制块上。通常的策略是，对于最常见的 3 个复制块，第一个复制块存储在同一机架的不同节点上，最后一个复制块存储在不同机架的某个节点上。

请注意，只有表示 DataNode 和块的文件映射的元数据经过 NameNode。当外部客户机发送请求要求创建文件时，NameNode 会以块标识和该块的第一个副

本的 DataNode IP 地址作为响应。此外，NameNode 还会通知其他将要接收该块副本的 DataNode。

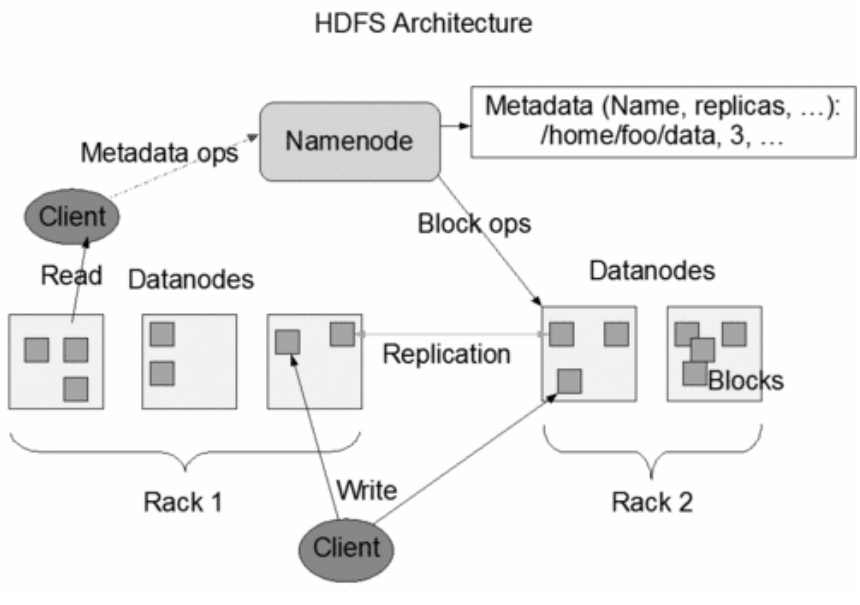


图 3

NameNode 在 FsImage 文件中存储所有关于文件系统名称空间的信息，该文件和一个包含所有事务的记录文件将存储在 NameNode 的本地文件系统上。FsImage 和 EditLog 文件也有副本，以防止文件损坏或 NameNode 系统故障。

DataNode

Hadoop 集群包含一个 NameNode 和 N（N>=1）个 DataNode。DataNode 通常以机架的形式组织，机架通过一个交换机将所有系统连接起来。通常，机架内部节点之间的传输速度快于机架间节点的传输速度。

DataNode 响应来自 HDFS 客户机的读与请求，响应 NameNode 发送的创建、删除和复制块的命令；NameNode 获取每个 DataNode 的心跳消息，每条消息包含一个块报告，NameNode 据此验证块映射和其他文件系统的元数据。如果 DataNode 无法发送心跳消息，NameNode 将采取修复措施，重新复制在该节点上丢失的块。

3.2 Hadoop Map/Reduce

Hadoop Map/Reduce 引擎由 JobTracker（作业服务器）和 Task Tracker（任务服务器）组成。

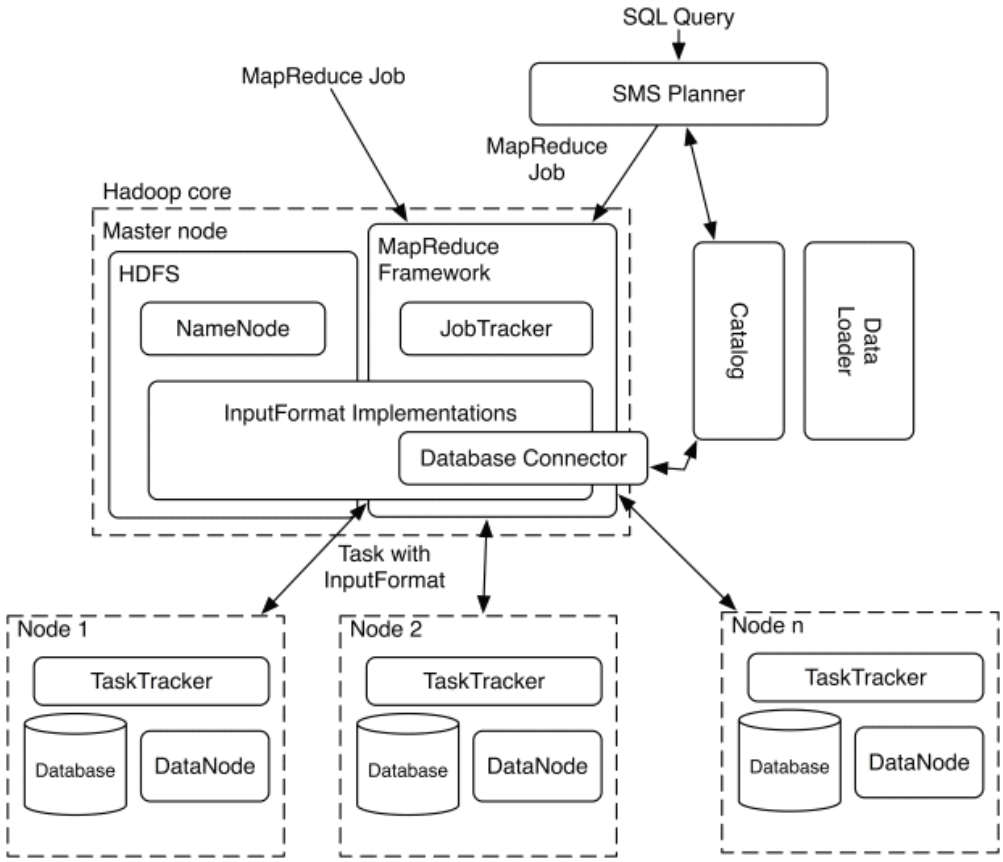


图 4

Job Tracker

Job Tracker（Google 称为 Master）是负责管理调度所有作业，它是整个系统分配任务的核心。它也是唯一的，这与 HDFS 类似。因此，简化了同步流程问题。

Task Tracker

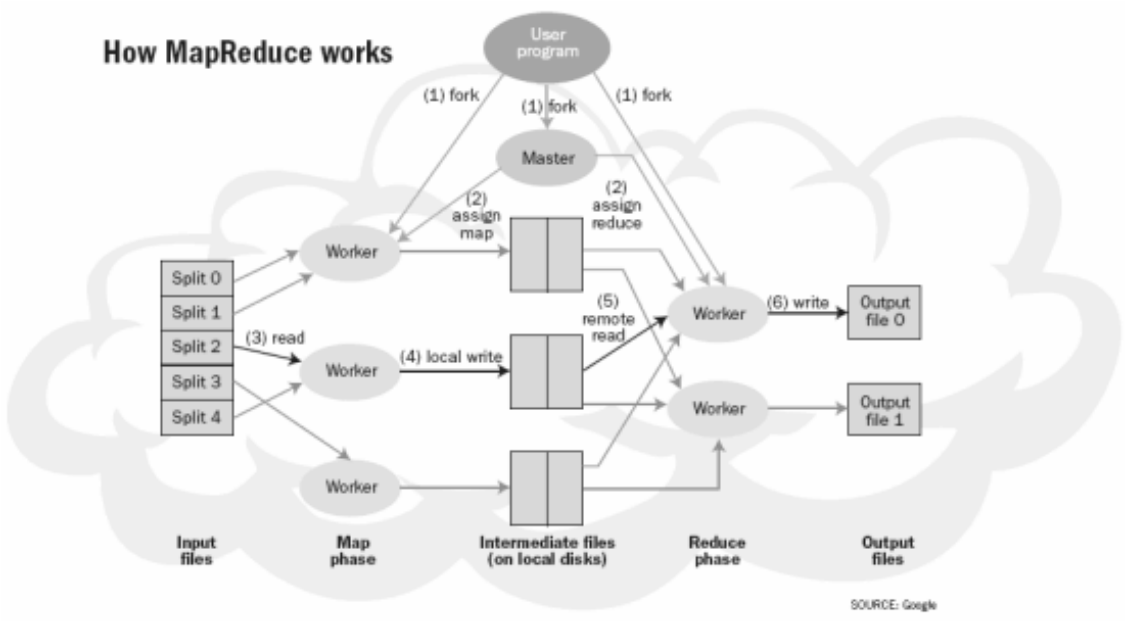
Task Tracker 具体负责执行用户定义操作，每个作业被分割为任务集，包括 Map 任务和 Reduce 任务。任务是具体执行的基本单元， Task Tracker 执行过程中需要向 Job Tracker 发送心跳信息，汇报每个任务的执行状态，帮助 Job Tracker 收集作业执行的整体情况，为下次任务分配提供依据。

在 Hadoop 中，客户端（任务的提交者）是一组 API，用户需要自定义自己需要的内容，由客户端将作业及其配置提交到 Job Tracker，并监控执行状况。

与 HDFS 的通信机制相同，Hadoop Map/Reduce 也使用协议接口来实现服务器间的通信。实现者作为 RPC 服务器，调用者经由 RPC 的代理进行调用。客户端与 Task Tracker 以及 Task Tracker 之间，都不再有直接通信。难道客户端就不需要了解具体任务的执行状况吗？不是。难道 Task Tracker 相互无需了解任务执行情况吗？也不是。由于整个集群各机器的通信比 HDFS 复杂的多，点对点直接通信难以维持状态信息，所以统一由 Job Tracker 收集整理转发。

4. 运行流程——Hadoop 是如何工作的

最简单的 MapReduce 应用程序至少包含 3 个部分：一个 Map 函数、一个 Reduce 函数和一个 main 函数。main 函数将作业控制和文件输入/输出结合起来。在这点上，Hadoop 提供了大量的接口和抽象类，从而为 Hadoop 应用程序开发人员提供许多工具，可用于调试和性能度量等。



MapReduce 本身就是用于并行处理大数据集的软件框架。MapReduce 的根源是函数性编程中的 map 和 reduce 函数。它由两个可能包含有许多实例（许多 Map 和 Reduce）的操作组成。Map 函数接受一组数据并将其转换为一个键/值对列表，输入域中的每个元素对应一个键/值对。Reduce 函数接受 Map 函数生成的列表，然后根据它们的键（为每个键生成一个键/值对）缩小键/值对列表。

这里提供一个示例，帮助您理解它。假设输入域是：我爱中国云计算网，我爱云计算技术论坛。在这个域上运行 Map 函数将得出以下的键/值对列表：

```
(我, 1)  (爱, 1)  (中国, 1)  (云计算, 1)  (网, 1)

(我, 1)  (爱, 1)  (云计算, 1)  (技术, 1)  (论坛, 1)
```

如果对这个键/值对列表应用 Reduce 函数，将得到以下一组键/值对：

```
(我, 2)  (爱, 2)  (中国, 1)  (云计算, 2)  (技术, 1)

(网, 1)  (论坛, 1)
```

结果是对输入域中的单词进行计数，这无疑对处理索引十分有用。但是，现在假设有两个输入域，第一个是“我爱中国云计算网”，第二个是“我爱云计算技术论坛”。您可以在每个域上执行 Map 函数和 Reduce 函数，然后将这两个键/值对列表应用到另一个 Reduce 函数，这时得到与前面一样的结果。换句话说，可以在输入域并行使用相同的操作，得到的结果是一样的，但速度更快。这便是 MapReduce 的威力；它的并行功能可在任意数量的系统上使用。图 5 演示了 MapReduce 的思想。

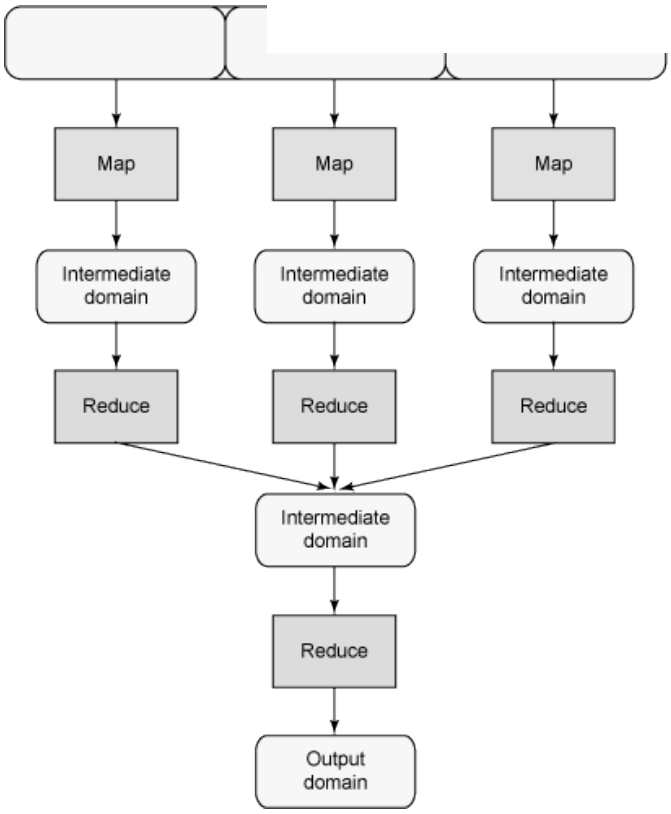


图 5

一个代表客户机在单个主系统上启动的 MapReduce 应用程序称为 JobTracker。类似于 NameNode，它是 Hadoop 集群中惟一负责控制 MapReduce 应用程序的系统。在应用程序提交之后，将提供包含在 HDFS 中的输入和输出目录。JobTracker 使用文件块信息(物理量和位置)确定如何创建其他 TaskTracker 从属任务。MapReduce 应用程序被复制到每个出现输入文件块的节点。将为特定节点上的每个文件块创建一个惟一的从属任务。每个 TaskTracker 将状态和完成信息报告给 JobTracker。

5. 任务粒度——Hadoop 是如何并行的

Map 调用把输入数据自动分割成 M 片，并且分发到多个节点上，使得输入数据能够在多个节点上并行处理。Reduce 调用利用分割函数分割中间 key，从而形成 R 片(例如，hash(key) mod R)，它们也会被分发到多个节点上。分割数量 R 和分割函数由用户来决定。

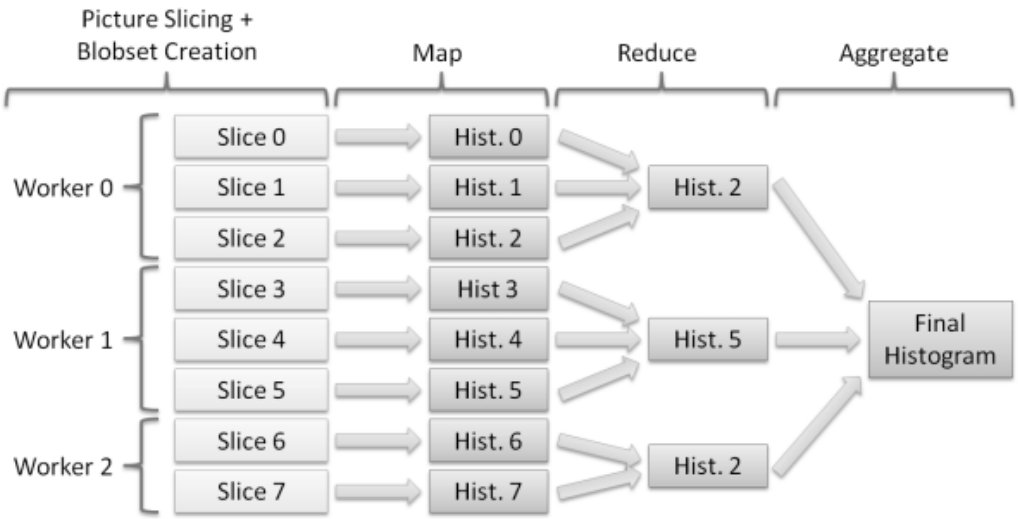


图 6

由上的分析可知，我们细分 map 阶段成 M 片，reduce 阶段成 R 片。在理想状态下，M 和 R 应当比 worker 机器数量要多得多。每个 worker 执行许多不同的工作来提高动态负载均衡，并且能够加快故障恢复的速度，这个失效机器上执行的大量 map 任务都可以分布到所有其他 worker 机器上执行。

但是在具体编程中，实际上对于 M 和 R 的取值有一定的限制，因为 master 必须执行 O(M+R)次调度，并且在内存中保存 O(M*R)个状态。（对影响内存使用的因素还是比较小的：O(M*R)块状态，大概每对 map 任务/reduce 任务 1 个字节就可以了）。

进一步来说，用户通常会指定 R 的值，因为每一个 reduce 任务最终都是一个独立的输出文件。在实际中，我们倾向于调整 M 的值，使得每一个独立任务都是处理大约 16M 到 64M 的输入数据（这样，上面描写的本地优化策略会最有效），另外，我们使 R 比较小，这样使得 R 占用不多的 worker 机器。我们通常会用这样的比例来执行 MapReduce: M=200，000，R=5，000，使用 2，000 台 worker 机器。

本节内容参考 MapReduce: Simplified Data Processing on Large Clusters 原文。

6. 参考文献

1. Hadoop Site，http://hadoop.apache.org/
2. Hadoop Wiki，http://en.wikipedia.org/wiki/Hadoop
3. MapReduce: Simplified Data Processing on Large Clusters. Google Inc.
4. http://www.cnblogs.com/duguguiyu/archive/2009/02/28/1400278.htm

l

5. Hadoop 分布式文件系统：架构和设计

   http://hadoop.apache.org/common/docs/r0.18.2/cn/hdfs_design.html

6. Hadoop Map/Reduce 教程

   http://hadoop.apache.org/common/docs/r0.18.2/cn/mapred_tutorial.html

7. 使用 Linux 和 Hadoop 进行分布式计算，

   http://www.ibm.com/developerworks/cn/linux/l-hadoop/