# 概述

以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。**利用HDFS的高容错性、高伸缩性等的点，用户可把hadoop部署在廉价的机器上，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轻松的组织计算机资源，搭建自己的分布式计算平台，并且充分利用集群的计算和存储能力。

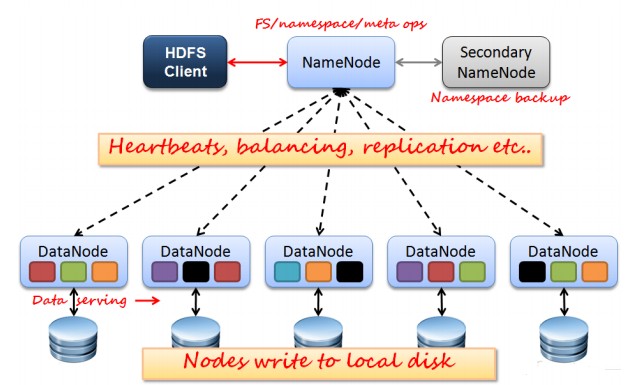
HDFS：实现分布式存储的底层的支持；

MapReduce：实现分布式并行任务处理的程序支持。

# HDFS

<https://www.cnblogs.com/laov/p/3434917.html>

HDFS使应用程序流式的访问集群中的数据集，HDFS被设计成适合进行批处理，而不是用户交互式处理。它重视的是数据的吞吐量，而不是数据访问的反应速度。



HDFS也是采用Master和Slave结构，在系统中分为NameNode、SecondaryNameNode和DataNode这三个角色。

**NameNode**

也称为Master节点。在集群中的主要任务有：管理数据块映射；处理客户端的读写请求；配置副本策略；管理HDFS的名称空间。

**SecondaryNameNode**

从命名就可以看出在系统中的作用，主要重要是分担NameNode的工作量，是作为NameNode的冷备份，合并fsimage（元数据镜像文件，是文件系统的目录树）和fsedits（元数据的操作日志）然后再发给namenode。

**DataNode**

称为Slave节点，负责存储clinet发来的数据块block；执行数据块的读写操作。

Hadoop中有一个作为主控的JobTracker，功能是调度和管理其他的TaskTracker，JobTracker可以运行在集群的任意一台机器上；TaskTracker负责执行任务，必须运行在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，这些任务并行进行，并负责监控任务的运行情况，如果一个TaskTracker出了故障，JobTracker会将其负责的任务转交给另一个空闲的TaskTracker。

NameNode内存中存储的是fsimage+fsedits，SecondaryNameNode负责定时（默认1小时）从NameNode上获取fsimage和fsedits来进行合并，然后再发送给namenode，减少namenode的工作量。

## 任务粒度

把原始大数据集切割从小数据集时，通常让小数据集小于等于HDFS中一个Block的大小，这样保证一个小数据集是位于一台计算机上的，便于**本地计算**（数据存储在哪个几点上，就让哪个节点负责这部分数据的计算任务）。假设有M个小数据，那么就会有M个map任务，reduce任务的数量R是由用户指定的。

## 数据分割 partition

map任务输出的中间结果按照key的范围划分成R份，划分通常使用Hash函数（hash(key) mod R），这样可以保证某一段范围的key由一个任务来处理，可以简化reduce的过程。

## 数据合并 combine

在数据分割之前，可以对中间结果进行数据合并，把中间结果中具有相同key的键值对合并起来。合并的过程和reduce过程类似，通常可以采用reduce函数进行，但是合并过程是作为map的一部分。合并操作可以减少中间结果中键值对的数量，从而降低网络流量。

## Reduce

map任务的中间结果经过combine和partition之后，以文件形式存储在本地磁盘上。中间结果文件的位置会通知JobTracker，然后JobTracker通知reduce任务到哪一个**TaskTracker上去取中间结果**。所有map任务产生的中间结果按key值通过Hash函数划分成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区间的中间结果，然后执行reduce函数，形成一个最终的结果文件。

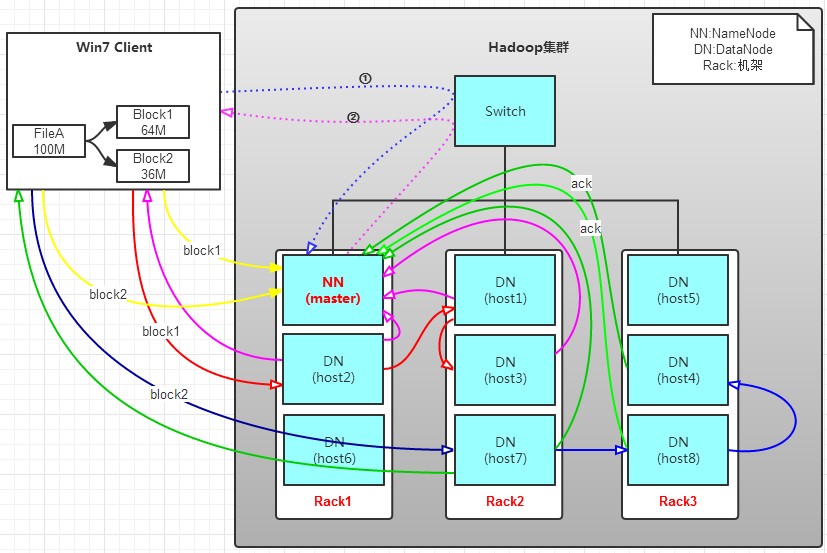
有R个reduce任务就会有R个最终结果，很多情况这R个最终结果不需要合并成一个最终结果，因为这R个最终结果可以作为下一个计算任务的输入。

## Hadoop集群配置

Hadoop的配置文件分为两类：

1. 只读类型的默认文件：core-default.xml、hdfs-default.xml、mapred-default.xml/mapred-queue.xml
2. 定位（site-specific）设置：core-site.xml、hdfs-site.xml、mapred-site.xml等。

## 写操作



假设客户端有一个100M的文件FileA，client要把FileA写入到HDFS中。HDFS按照上图的配置，block大小为64MB，HDFS分布在Rock1、Rock2和Rock3上。

1. Client将FileA按64M分块。分成两块，block1和Block2；
2. Client向nameNode发送写数据请求，如图蓝色虚线；
3. NameNode节点，记录block信息。并返回可用的DataNode，如粉色虚线；

Block1: host2,host1,host3

Block2: host7,host8,host4

master选择DataNode的原理是：

NameNode具有RackAware机架感知功能，这个可以配置。若client为DataNode节点，那存储block时，规则为：副本1，同client的节点上；副本2，不同机架节点上；副本3，同第二个副本机架的另一个节点上；其他副本随机挑选。若client不为DataNode节点，那存储block时，规则为：副本1，随机选择一个节点上；副本2，不同副本1，机架上；副本3，同副本2相同的另一个节点上；其他副本随机挑选。

1. client向DataNode发送block1；发送过程是以流式写入。

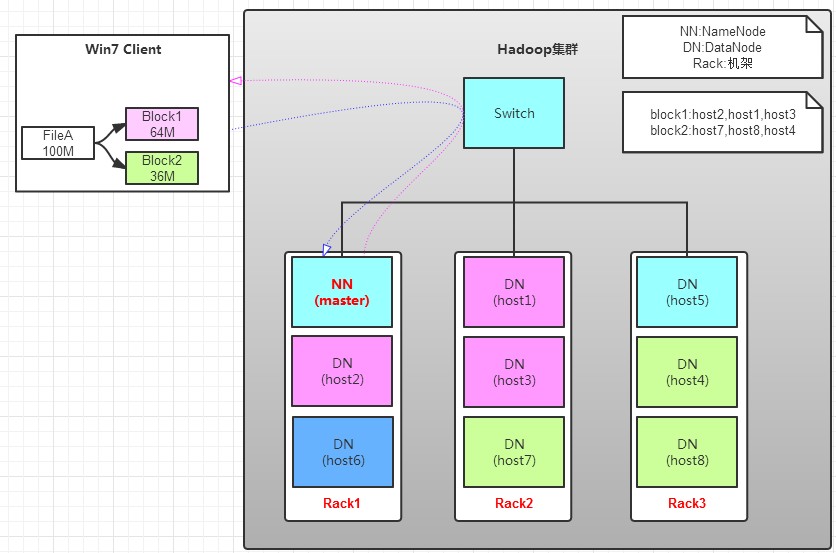
具体的写入流程为：

* 将64M的block1按64k的package划分;
* 然后将第一个package发送给host2;
* host2接收完后，将第一个package发送给host1，同时client向host2发送第二个package；
* host1接收完第一个package后，发送给host3，同时接收host2发来的第二个package。
* 以此类推，如图红线实线所示，直到将block1发送完毕。
* host2,host1,host3向NameNode，host2向Client发送通知，说“消息发送完了”。如图粉红颜色实线所示。
* client收到host2发来的消息后，向namenode发送消息，说我写完了。这样就真完成了。如图黄色粗实线
* 发送完block1后，再向host7，host8，host4发送block2，如图蓝色实线所示。
* 发送完block2后，host7,host8,host4向NameNode，host7向Client发送通知，如图浅绿色实线所示。
* client向NameNode发送消息，说我写完了，如图黄色粗实线。。。这样就完毕了。

通过分析写操作的过程，可以发现：

1. 写1T文件，我们需要3T的存储，3T的网络流量贷款。
2. 在执行读或写的过程中，NameNode和DataNode通过HeartBeat进行保存通信，确定DataNode活着。如果发现DataNode死掉了，就将死掉的DataNode上的数据，放到其他节点去。读取时，要读其他节点去。
3. 挂掉一个节点，没关系，还有其他节点可以备份；甚至，挂掉某一个机架，也没关系；其他机架上，也有备份。

## 读操作



读操作就简单一些了，如图所示，client要从datanode上，读取FileA。而FileA由block1和block2组成。读取的流程为：

1. client向namenode发送读请求。
2. namenode查看Metadata信息，返回fileA的block的位置。block1:host2,host1,host3

block2:host7,host8,host4

1. block的位置是有先后顺序的，先读block1，再读block2。而且block1去host2上读取；然后block2，去host7上读取。

上面例子中，client位于机架外，那么如果client位于机架内某个DataNode上，例如,client是host6。那么读取的时候，遵循的规律是：优选读取本机架上的数据。

## 漫画图解HDFS

