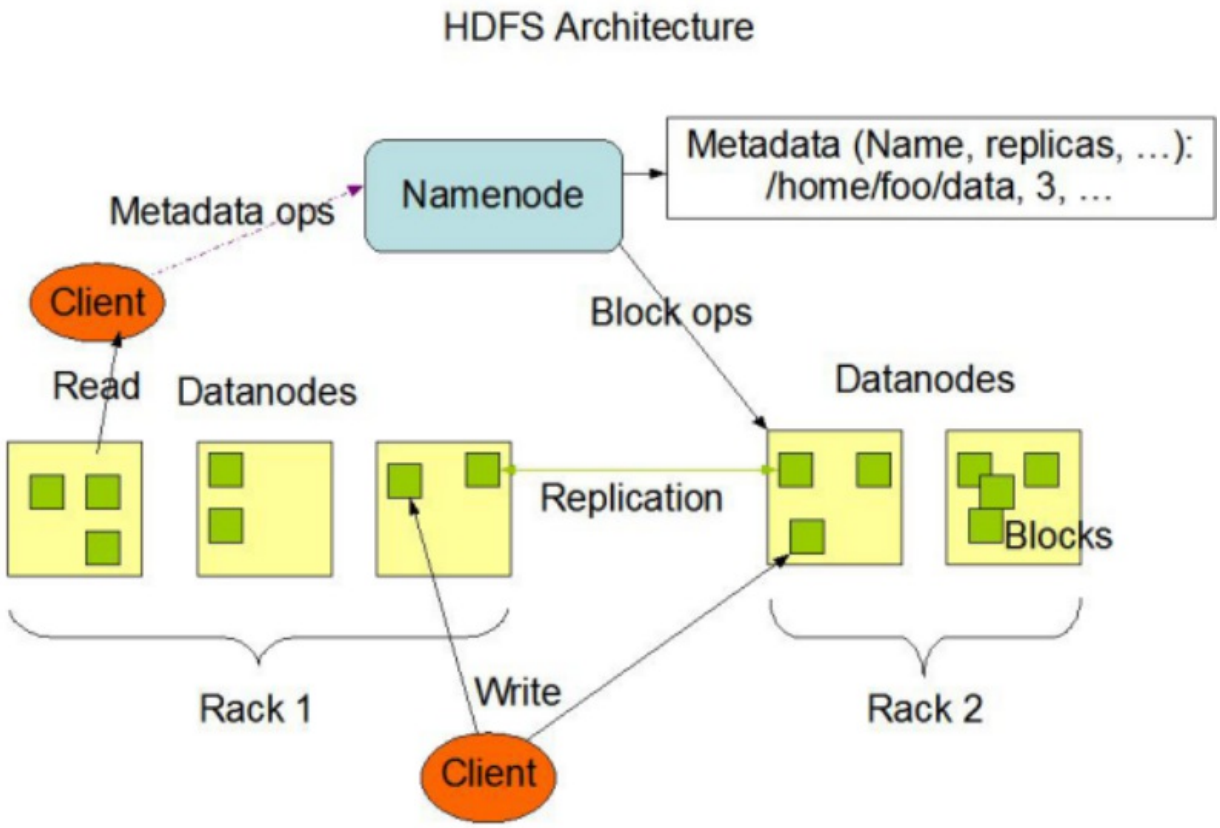
Hadoop HDFS运行原理

一、引言

Hadoop分布式文件系统(HDFS Hadoop Distributed File System)被设计成适合运行在通用硬件(commodity hardware)上的分布式文件系统。它和现有的分布式文件系统有很多共同点。但同时，它和其他的分布式文件系统的区别也是很明显的。HDFS是一个高度容错性的系统，适合部署在廉价的机器上。HDFS能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。HDFS放宽了一部分POSIX约束，来实现流式读取文件系统数据的目的。HDFS在最开始是作为Apache Nutch搜索引擎项目的基础架构而开发的。

二、HDFS架构



三、HDFS基础概念

NameNode：元数据节点,负责管理文件系统命名空间（Namespace）和处理客户端对文件的访问。

DataNode：数据节点，是HDFS真正存储数据的地方。客户端（client）和元数据节点（NameNode）可以向数据节点请求写入或者读数据块。此外，DataNode需要周期性的向元数据节点回报其存储的数据块信息。

Secondary NameNode：NameNode将对文件系统的改动追加保存到本地文件系统上的一个日志文件（edits）。当一个NameNode启动时，它首先从一个映像文件（fsimage）中读取HDFS的状态，接着应用日志文件中的edits操作。然后它将新的HDFS状态写入（fsimage）中，并使用一个空的edits文件开始正常操作。因为NameNode只有在启动阶段才合并fsimage和edits，所以久而久之日志文件可能会变得非常庞大，特别是对大型的集群。日志文件太大的另一个副作用是下一次NameNode启动会花很长时间。Secondary NameNode定期合并fsimage和edits日志，将edits日志文件大小控制在一个限度下。因为内存需求和NameNode在一个数量级上，所以通常secondary NameNode和NameNode运行在不同的机器上。Secondary NameNode通过bin/start-dfs.sh在conf/masters中指定的节点上启动。

相比于Hadoop1.0，Hadoop 2.0中的HDFS增加了两个重大特性，HA和Federaion。HA即为High Availability，用于解决NameNode单点故障问题，该特性通过热备的方式为主NameNode提供一个备用者，一旦主NameNode出现故障，可以迅速切换

Secondary NameNode至备NameNode，从而实现不间断对外提供服务。Federation即为“联邦”，该特性允许一个HDFS集群中存在多个NameNode同时对外提供服务，这些NameNode分管一部分目录（水平切分），彼此之间相互隔离，但共享底层的DataNode存储资源。

Secondary NameNode的检查点进程启动，是由两个配置参数控制的：

fs.checkpoint.period，指定连续两次检查点的最大时间间隔， 默认值是1小时。

fs.checkpoint.size定义了edits日志文件的最大值，一旦超过这个值会导致强制执行检查点（即使没到检查点的最大时间间隔）。默认值是64MB。

日志与镜像的定期合并总共分五步：

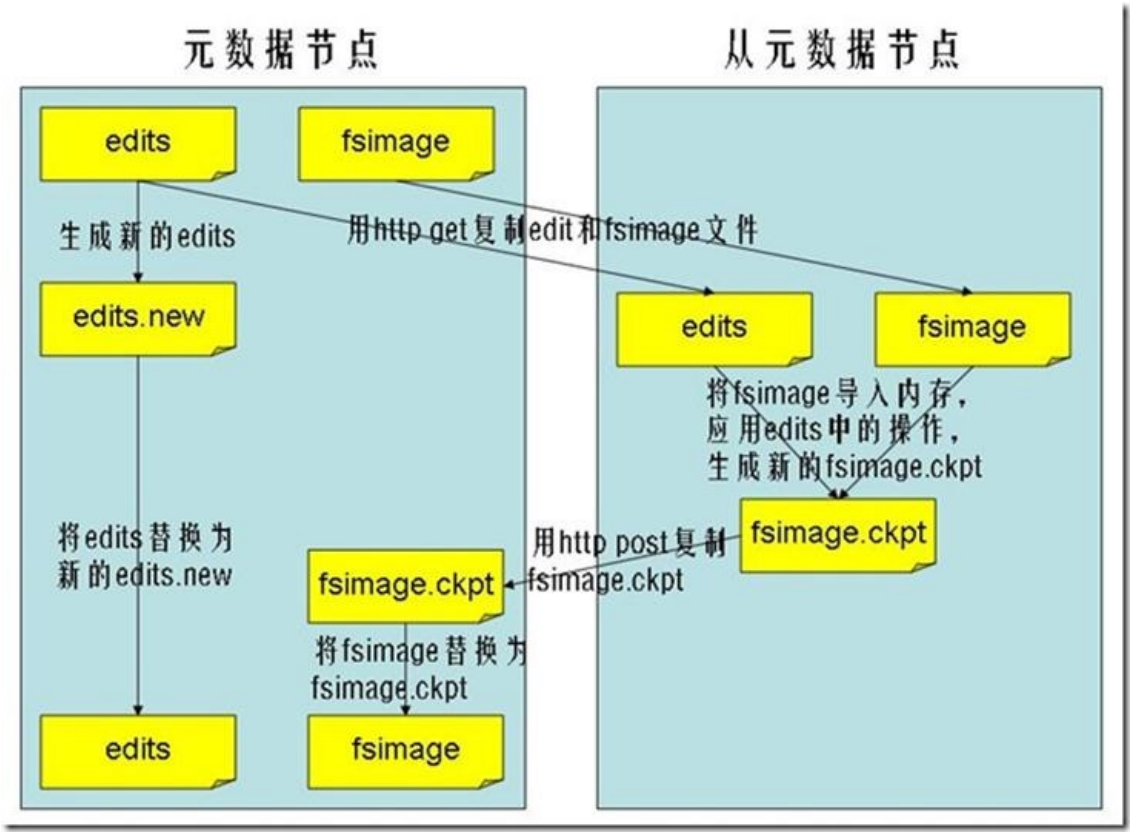
1、SecondaryNameNode通知NameNode准备提交edits文件，此时主节点产生edits.new

2、SecondaryNameNode通过http get方式获取NameNode的fsimage与edits文件（在SecondaryNameNode的current同级目录下可见到 temp.check-point或者previous-checkpoint目录，这些目录中存储着从namenode拷贝来的镜像文件）

3、SecondaryNameNode开始合并获取的上述两个文件，产生一个新的fsimage文件fsimage.ckpt

4、SecondaryNameNode用http post方式发送fsimage.ckpt至NameNode

5、NameNode将fsimage.ckpt与edits.new文件分别重命名为fsimage与edits，然后更新fstime，整个checkpoint过程到此结束。



四、HDFS中文件读写操作流程

在HDFS中，文件的读写过程就是client和NameNode以及DataNode一起交互的过程。我们已经知道NameNode管理着文件系统的元数据，DataNode存储的是实际的数据，那么client就会联系NameNode以获取文件的元数据，而真正的文件读取操作是直接和DataNode进行交互的。

写文件的过程：

1.初始化FileSystem，客户端调用create()来创建文件

2.FileSystem用RPC调用元数据节点，在文件系统的命名空间中创建一个新的文件，元数据节点首先确定文件原来不存在，并且客户端有创建文件的权限，然后创建新文件。

3.FileSystem返回DFSOutputStream，客户端用于写数据，客户端开始写入数据。

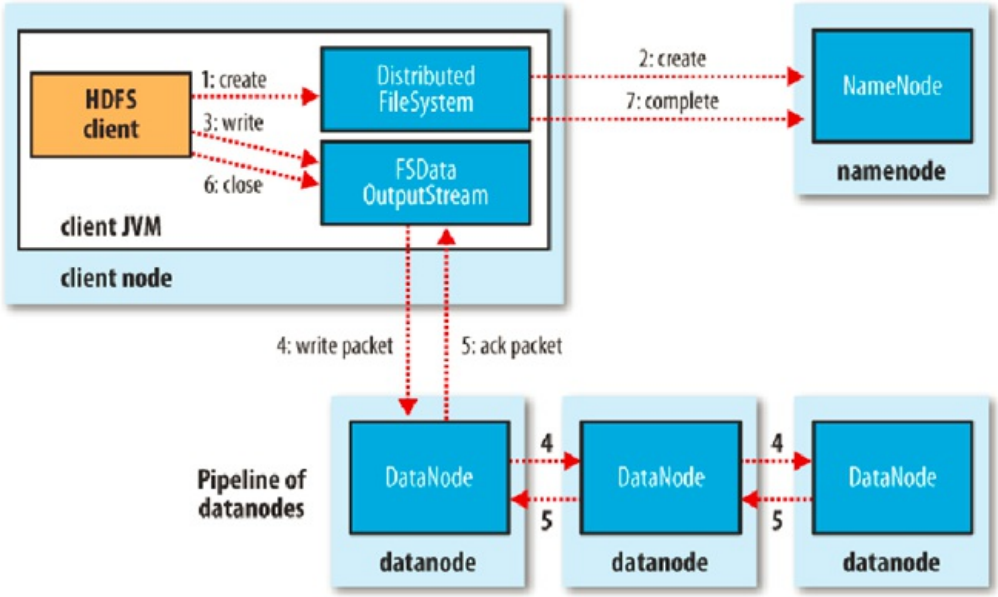
4.DFSOutputStream将数据分成块，写入data queue。data queue由Data Streamer读取，并通知元数据节点分配数据节点，用来

存储数据块(每块默认复制3块)。分配的数据节点放在一个pipeline里。Data Streamer将数据块写入pipeline中的第一个数据节点。第一个数据节点将数据块发送给第二个数据节点。第二个数据节点将数据发送给第三个数据节点。

5.DFSOutputStream为发出去的数据块保存了ack queue，等待pipeline中的数据节点告知数据已经写入成功。

6.当客户端结束写入数据，则调用stream的close函数。此操作将所有的数据块写入pipeline中的数据节点，并等待ack queue返回成功。最后通知元数据节点写入完毕。

7.如果数据节点在写入的过程中失败，关闭pipeline，将ack queue中的数据块放入data queue的开始，当前的数据块在已经写入的数据节点中被元数据节点赋予新的标示，则错误节点重启后能够察觉其数据块是过时的，会被删除。失败的数据节点从pipeline中移除，另外的数据块则写入pipeline中的另外两个数据节点。元数据节点则被通知此数据块是复制块数不足，将来会再创建第三份备份。



读文件的过程：

1.初始化FileSystem，然后客户端(client)用FileSystem的open()函数打开文件

2.FileSystem用RPC调用元数据节点，得到文件的数据块信息，对于每一个数据块，元数据节点返回保存数据块的数据节点的地址。

3.FileSystem返回FSDataInputStream给客户端，用来读取数据，客户端调用stream的read()函数开始读取数据。

4.DFSInputStream连接保存此文件第一个数据块的最近的数据节点，data从数据节点读到客户端(client)

5.当此数据块读取完毕时，DFSInputStream关闭和此数据节点的连接，然后连接此文件下一个数据块的最近的数据节点。

6.当客户端读取完毕数据的时候，调用FSDataInputStream的close函数。

7.在读取数据的过程中，如果客户端在与数据节点通信出现错误，则尝试连接包含此数据块的下一个数据节点。

8.失败的数据节点将被记录，以后不再连接。