Hadoop分布式文件系统学习总结

安思宇

Hadoop文件系统具体实现

文件系统	URI方案	Java实现（org.apache.hadoop）	定义
Local	file	fs.LocalFileSystem	支持有客户端校验和本地文件系统。带有校验和的本地系统文件在fs.RawLocalFileSystem中实现。
HDFS	hdfs	hdfs.DistributionFileSystem	Hadoop的分布式文件系统。
HFTP	hftp	hdfs.HftpFileSystem	支持通过**HTTP**方式以**只读**的方式访问HDFS，distcp经常用在**不同**的HDFS**集群间**复制数据。
HSFTP	hsftp	hdfs.HsftpFileSystem	支持通过**HTTPS**方式以**只读**的方式访问HDFS。
HAR	har	fs.HarFileSystem	构建在Hadoop文件系统之上，对文件进行归档。Hadoop归档文件主要用来**减少**NameNode的**内存使用**。
KFS	kfs	fs.kfs.KosmosFileSystem	Cloudstore（其前身是Kosmos文件系统）文件系统是**类似于**HDFS和Google的GFS文件系统，使用C++编写。

HDFS设计及特点

- 超大文件存储
- 流式数据访问（一次写入，多次读取最高效）
- 对硬件要求低,可运行在廉价的商用服务器上
- 数据访问延迟高（HDFS是为高数据吞吐量应用优化的，可能会以提高时间延迟为代价，HBase更适用低延迟的访问需求）
- Namenode将文件系统的元数据存在内存中，因此该文件系统的存储量受限于Namenode的内存容量，小文件多的情况。
- HDFS中的文件可能只有一个writer，而且写操作总是将数据添加在文件的末尾。不支持多个写入者操作，也不支持在文件的任意位置进行修改。

HDFS数据块

- 磁盘块：一般为512B，文件系统块的大小是磁盘块大小的整数倍。
- HDFS数据块：默认为64MB，时常设为128MB，且小于一个块的文件不会占据整个块的空间。每一个数据块会在多个DataNode上存储多份副本，默认是三份。

- HDFS数据块很大的原因：最小化寻址开销。使磁盘传输数据的时间明显大于定位到这个块开始位置所需的时间。
- 不能过大：由于MapReduce中的map任务通常一次只处理一个块中的数据，因此如果任务数太少（少于集群中的节点数量），作业的运行速度就会变慢。

HDFS块抽象的优点

- 使一个文件的大小可以大于网络中任意一个磁盘的容量。文件的所有块并不需要存储在同一个磁盘上。
- 可以将元数据放在块外，使用抽象块而非整个文件作为存储单元，简化了存储子系统的设计。
- 块适合用于数据备份进而提供数据容错能力和提高可用性。将每个块复制到少数几个独立的（默认3个）。可以确保在磁盘或机器上发生故障后数据不会丢失。

- HDFS中fsck指令可以显示块信息。

NameNode & DataNode

- NameNode： 元数据节点

➢管理文件系统的命名空间，维护文件系统树及整棵树内所有的文件和目录。这些信息以命名空间镜像文件（namespace image）和编辑日志文件（edit log）两个文件形式保存在本地磁盘上

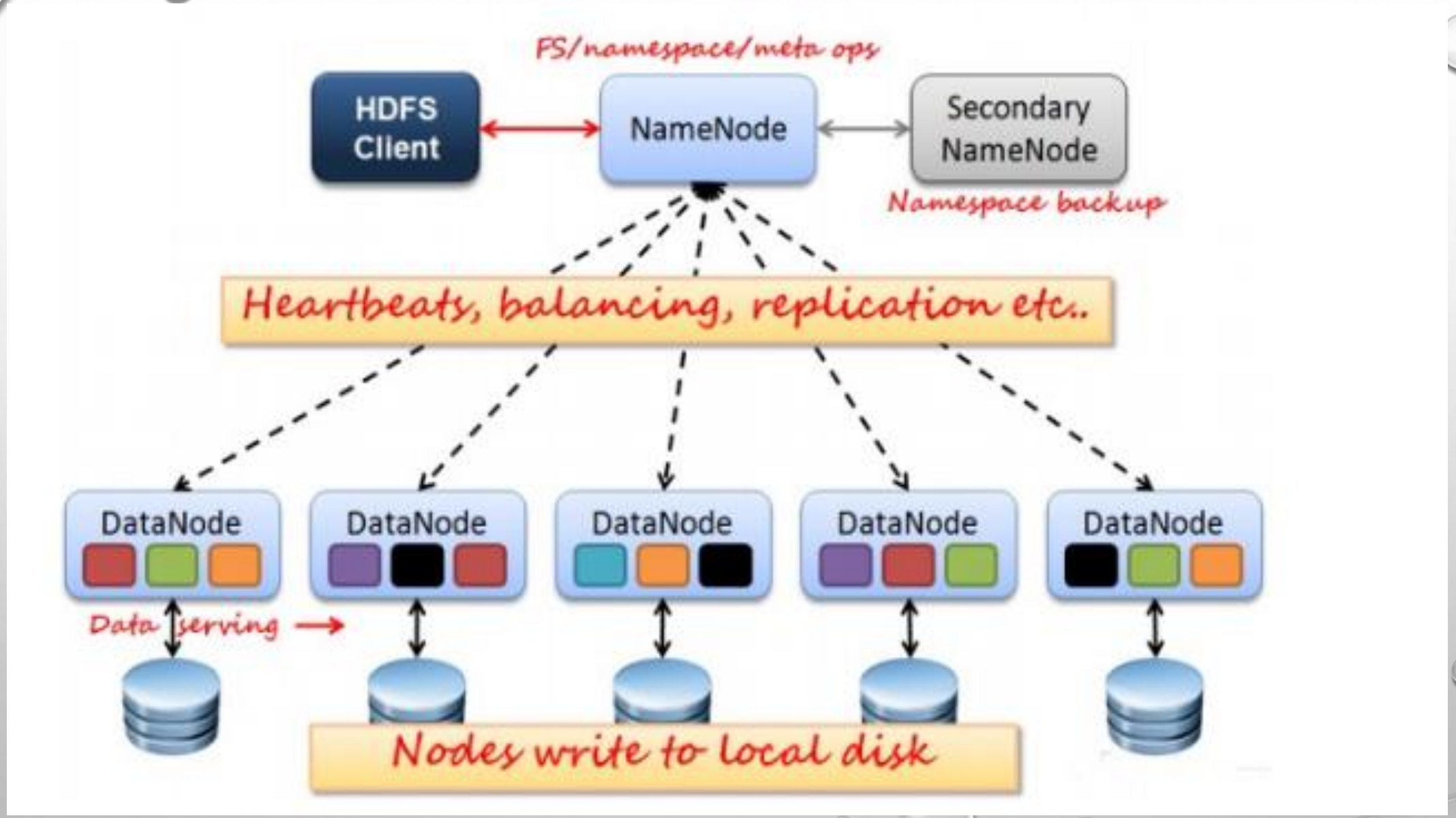
➢记录每个文件中各个块所在的数据节点信息，但并不将其存储在磁盘上，因为这些信息会在系统启动时由数据节点重建

NameNode & DataNode

- DataNode：数据节点

➢ 根据需要存储并检索数据块（受客户端或NameNode调度）

➢ 并且定期向NameNode发送它们所存储的数据块信息。


FS/namespace/meta ops
HDFS Client
NameNode
Secondary NameNode
Namespace backup
Heartbeats, balancing, replication etc..
DataNode
DataNode
DataNode
DataNode
DataNode
Data serving
Nodes write to local disk

NameNode容错机制

- 备份组成文件系统元数据持久状态的文件：通过配置使NameNode在多个文件系统上实时保存元数据的状态。一般的配置是，将持久状态写入本地磁盘的同时，写入一个远程挂载的网络文件系统(NFS)

- 运行一个辅助NameNode（secondary NameNode）：在另外一台单独的物理机上运行，主要负责周期性地将元数据节点的命名空间镜像文件和编辑日志合并，以防日志文件过大。同时实时保存合并后的命名空间镜像的副本，并在NameNode发生故障时启用恢复。（不是元数据节点的备用节点）

从元数据节点不是元数据节点的备用节点

元数据节点的目录结构

```
${dfs.name.dir}/current/VERSION
                       /edits
                       /fsimage
                       /fstime
```

从元数据节点的目录结构

```
${fs.checkpoint.dir}/current/VERSION
                            /edits
                            /feimage
                            /fstime
                  /previous.checkpoint/VERSION
                                      /edits
                                      /fsimage
                                      /fstime
```

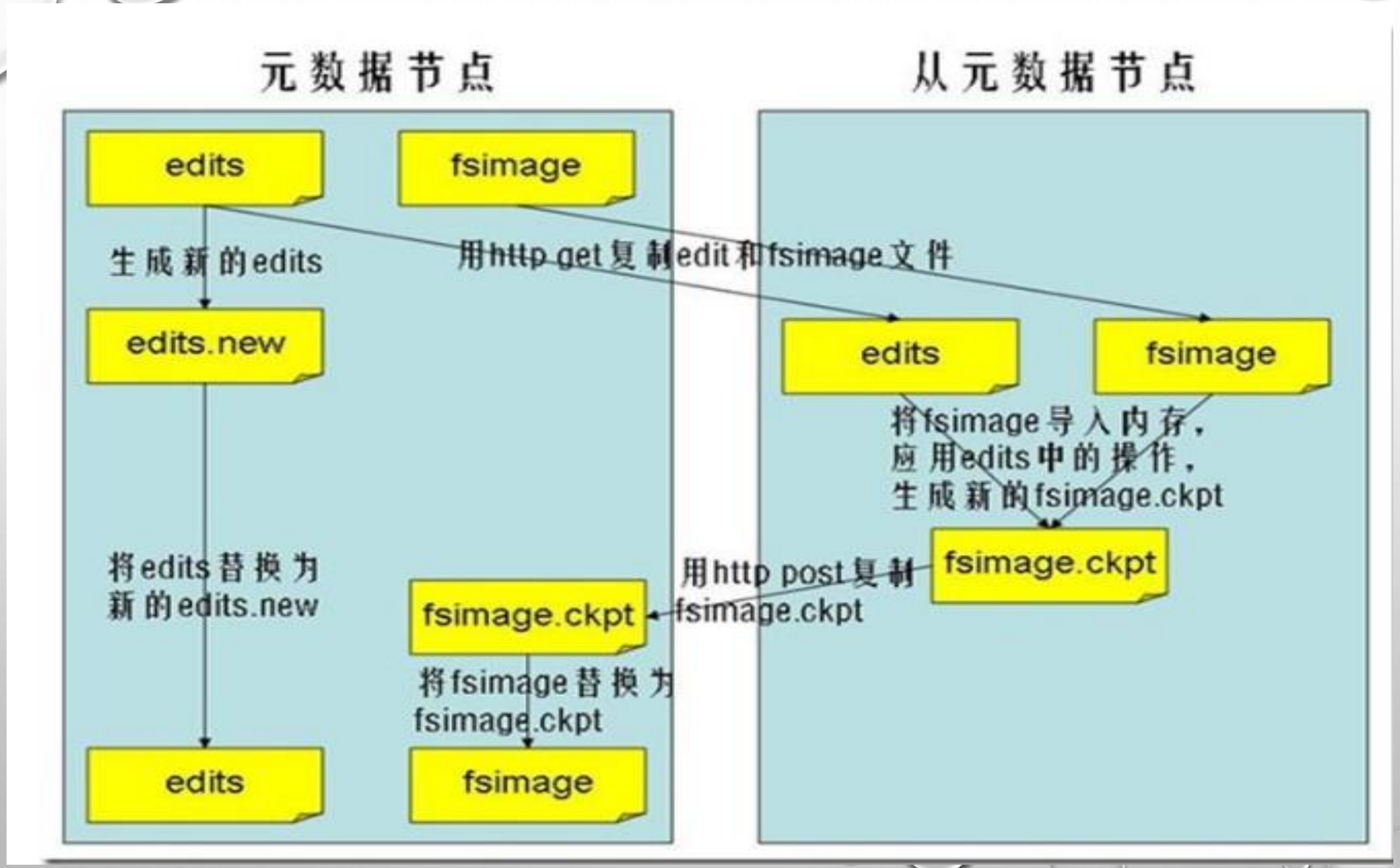
数据节点的目录结构

```
${dfs.data.dir}/current/VERSION
                       /blk_<id_1>
                       /blk_<id_1>.meta
                       /blk_<id_2>
                       /blk_<id_2>.meta
                       /...
                       /blk_<id_64>
                      /blk_<id_64>.meta
                      /subdir0/
                      /subdir1/
                      /...
                      /subdir63/
```

- 当文件系统客户端(client)进行写操作时，首先把它记录在编辑日志中(edit log)
- 元数据节点在内存中保存了文件系统的元数据信息。在记录了编辑日志后，元数据节点则修改内存中的数据结构。
- 每次的写操作成功之前，编辑日志都会同步(sync)到文件系统。
- fsimage文件，也即命名空间映像文件，是内存中的元数据在硬盘上的checkpoint，它是一种序列化的格式，并不能够在硬盘上直接修改。
- 当元数据节点失败时，则最新checkpoint的元数据信息从fsimage加载到内存中，然后逐一重新执行修改日志中的操作。
- 从元数据节点就是用来帮助元数据节点将内存中的元数据信息checkpoint到硬盘上的

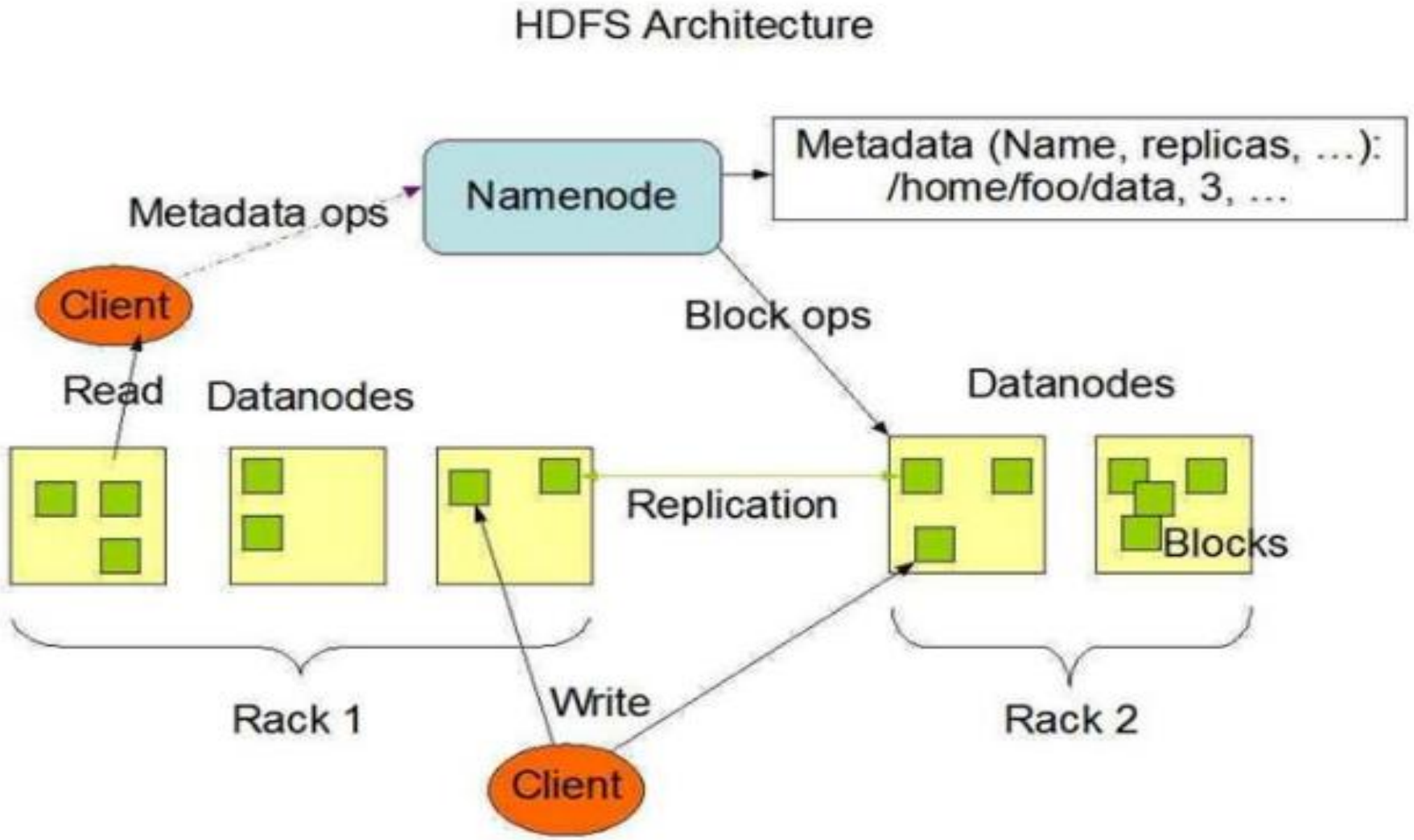
checkpoint的过程如下：

I. 从元数据节点通知元数据节点生成新的日志文件，以后的日志都写到新的日志文件中。
II. 从元数据节点用http get从元数据节点获得fsimage文件及旧的日志文件。
III. 从元数据节点将fsimage文件加载到内存中，并执行日志文件中的操作，然后生成新的fsimage文件。
IV. 从元数据节点将新的fsimage文件用http post传回元数据节点
V. 元数据节点可以将旧的fsimage文件及旧的日志文件，换为新的fsimage文件和新的日志文件(第一步生成的)，然后更新fstime文件，写入此次checkpoint的时间。
VI. 这样元数据节点中的fsimage文件保存了最新的checkpoint的元数据信息，日志文件也重新开始，不会变的很大了。


元数据节点
从元数据节点
edits
fsimage
生成新的edits
用http get复制edit和fsimage文件
edits.new
edits
fsimage
将fsimage导入内存，
应用edits中的操作，
生成新的fsimage.ckpt
fsimage.ckpt
将edits替换为
新的edits.new
用http post复制
fsimage.ckpt
fsimage.ckpt
将fsimage替换为
fsimage.ckpt
edits
fsimage

HDFS体系结构

- HDFS是一个主/从（master/slave）体系结构，由于分布式存储的性质，HDFS集群拥有一个NameNode和一些DataNode。NameNode管理文件系统的元数据，DataNode存储实际的数据。客户端通过NameNode和DataNode的交互访问文件系统。客户端联系NameNode以获取文件的元数据，而真正的文件I/O操作是直接和DataNode进行交互的。（多个机柜）



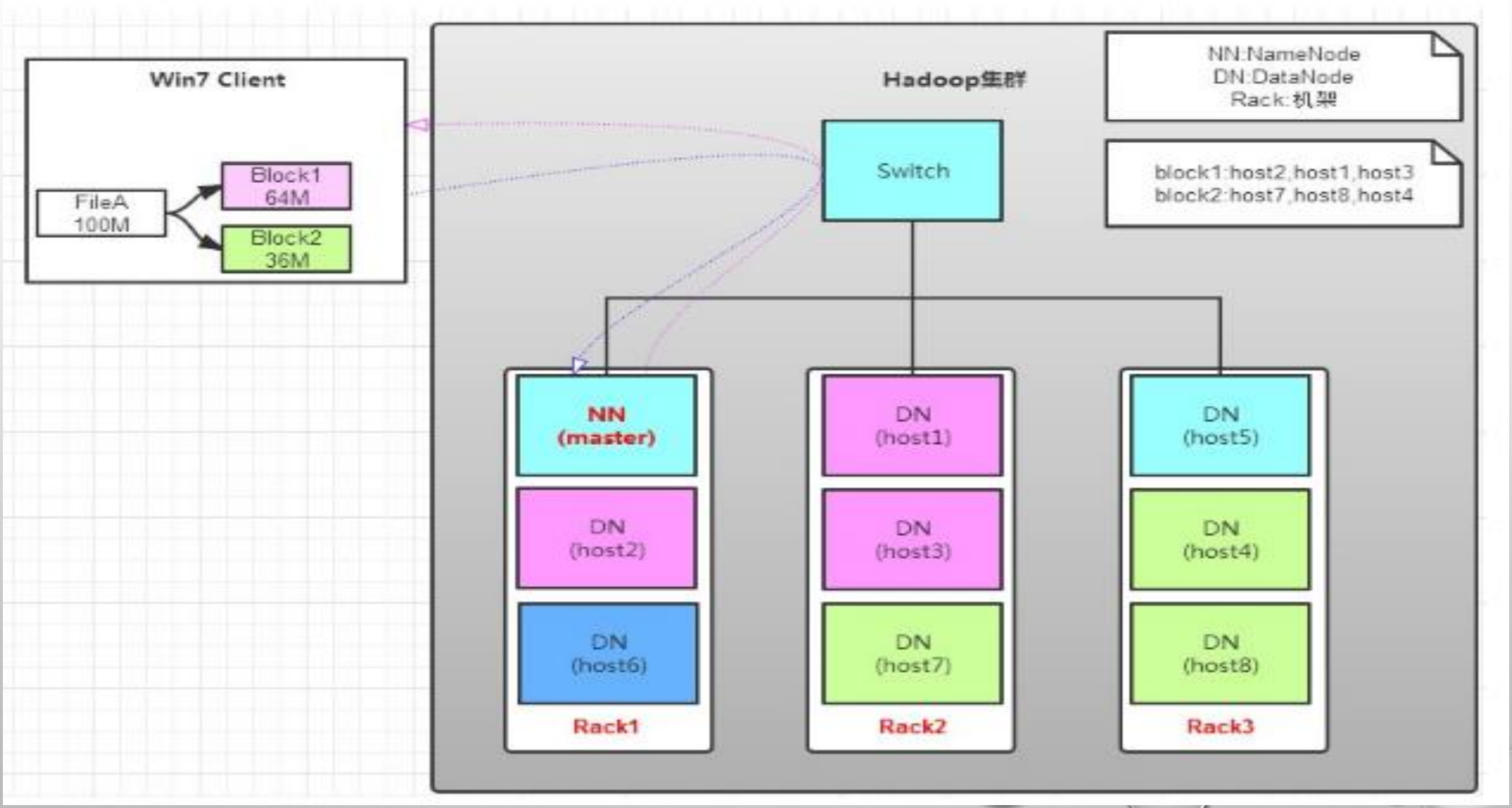
通信协议

- 所有HDFS通讯协议都是构建在TCP/IP协议上
- 客户端通过一个可配置的端口连接到NameNode，通过ClientProtocol与NameNode交互
- DataNode使用DataNodeProtocol与NameNode交互
- 从ClientProtocol和DataNodeProtocol抽象出一个远程调用（RPC），NameNode不会主动发起RPC，而是相应来此客户端和DataNode的RPC请求

读写过程

- 写：
  - Client向NameNode发起文件写入请求
  - NameNode根据文件大小和文件块配置情况，返回给Client它所管理部分DataNode的信息
  - Client将文件划分为多个Block，根据DataNode的地址信息，按顺序写入到每一个DataNode块中。
- 读：
- Client向NameNode发起文件读取的请求
- NameNode返回文件存储的DataNode信息
- Client读取文件信息

文件读取数据流


Win7 Client
FileA
100M
Block1
64M
Block2
36M
Hadoop集群
NN:NameNode
DN:DataNode
Rack:机架
block1:host2,host1,host3
block2:host7,host8,host4
Switch
NN
(master)
DN
(host2)
DN
(host6)
Rack1
DN
(host1)
DN
(host3)
DN
(host7)
Rack2
DN
(host5)
DN
(host4)
DN
(host8)
Rack3

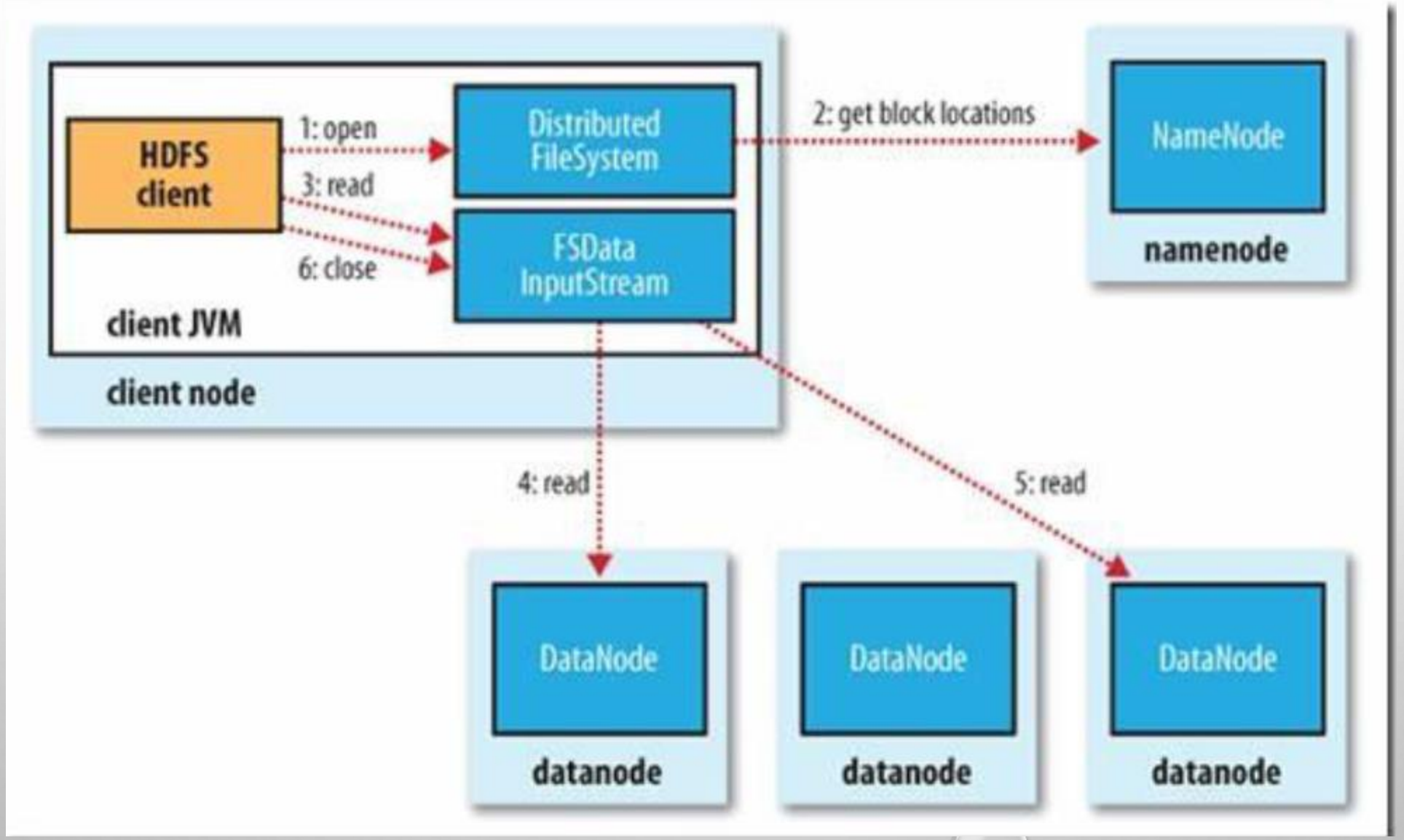
- client要从DataNode上读取FileA，FileA由block1和block2组成

读操作流程为：

- A.client向NameNode发送读请求
- B.NameNode查看MetaData信息，返回fileA的block位置

  block1： host2,host1,host3

  block2： host7,host8,host4
- C.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；block2去host7读取；
- 当client位于机架内的某个DataNode上时，读取遵循的规律是优先读取本机架上的数据。


HDFS client
1: open
3: read
6: close
Distributed FileSystem
FSData InputStream
client JVM
client node
2: get block locations
NameNode
namenode
4: read
5: read
DataNode
datanode
DataNode
datanode
DataNode
datanode

I. 客户端（client）调用FileSystem的open()函数打开文件

II. DistributedFileSystem通过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的地址

III. DistributedFileSystem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

IV. 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）

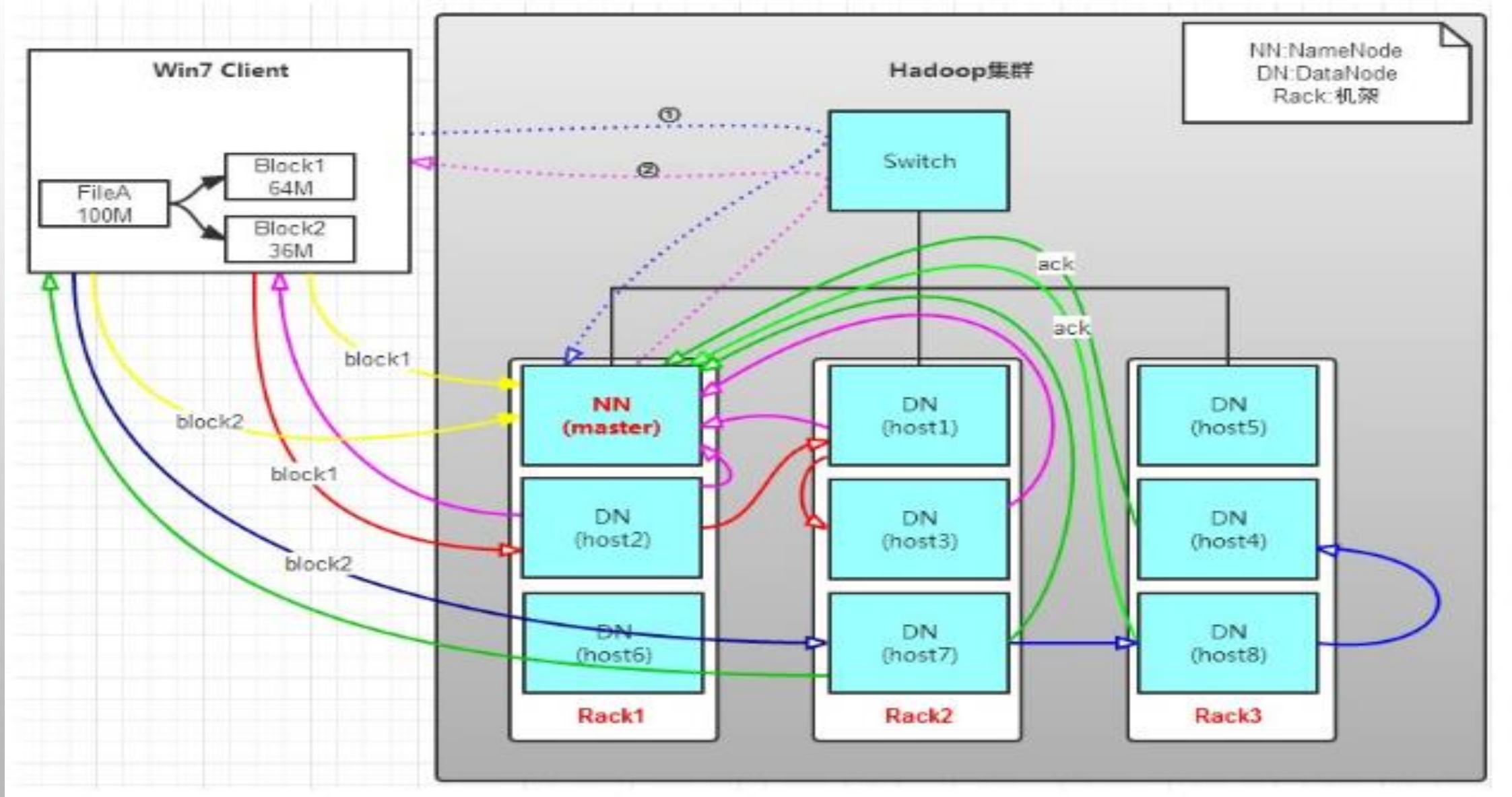
V. 当一个数据块读取完毕时，DFSInputStream关闭和此数据节点的连接，然后连接此文件下一个数据块的最近的数据节点

VI. 当客户端读取数据完毕，调用FSDataInputStream的close函数

◆读取中如果客户端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节点，失败的数据节点将被记录，以后不再连接。

◆NameNode会告知客户端每个块中最佳的DataNode，并让客户端直接连接到该DataNode检索数据。由于数据流分散在集群中的所有DataNode，所以这种设计能使HDFS可扩展到大量的并发客户端。而且NameNode只需要响应块位置请求，无需相应数据请求，因此更加高效。

文件写入数据流


Win7 Client
FileA
100M
Block1
64M
Block2
36M
Hadoop集群
NN:NameNode
DN:DataNode
Rack:机架
①
②
Switch
ack
ack
block1
block2
block1
block2
NN
(master)
DN
(host2)
DN
(host6)
DN
(host1)
DN
(host3)
DN
(host7)
DN
(host5)
DN
(host4)
DN
(host8)
Rack1
Rack2
Rack3

- 将一个文件FILEA，100M大小。Client将FileA写入到HDFS上
- HDFS分布在三个机架上Rack1，Rack2，Rack3

a) Client将FileA按64M分块。分成两块，block1和block2；

b) Client向NameNode发送写数据请求

c) NameNode节点，记录block信息。并返回可用的DataNode，两个管线

Block1：host2，host1，host3

Block2：host7，host8，host4

若client为DataNode节点，那么存储block时，规则为：副本1在同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；其他副本任意

若client不为DataNode节点，那么存储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的机架上；副本3选择同副本2相同机架的另一个节点上；其他副本随机挑选。

复本管线要求能够提供良好的稳定性（不同机架）并实现很好的负载均衡，包括写入带宽（写入操作只需要遍历一个交换机），读取性能（可以从两个机架中选择读取）和集群中块的均匀分布（客户端只在本地机架上写入一个块）

d）client向DataNode发送block1；发送过程是以流式写入。

- 流式写入过程，

1.将64M的block1按64k的package划分

2.将第一个package发送给host2

3.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

4.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

5.以此类推，直至将block1发送完毕

6.host2，host1，host3向NameNode，host2向Client发送通知消息发送结束

7.client收到host2发来的消息后，向NameNode发送消息写入完成

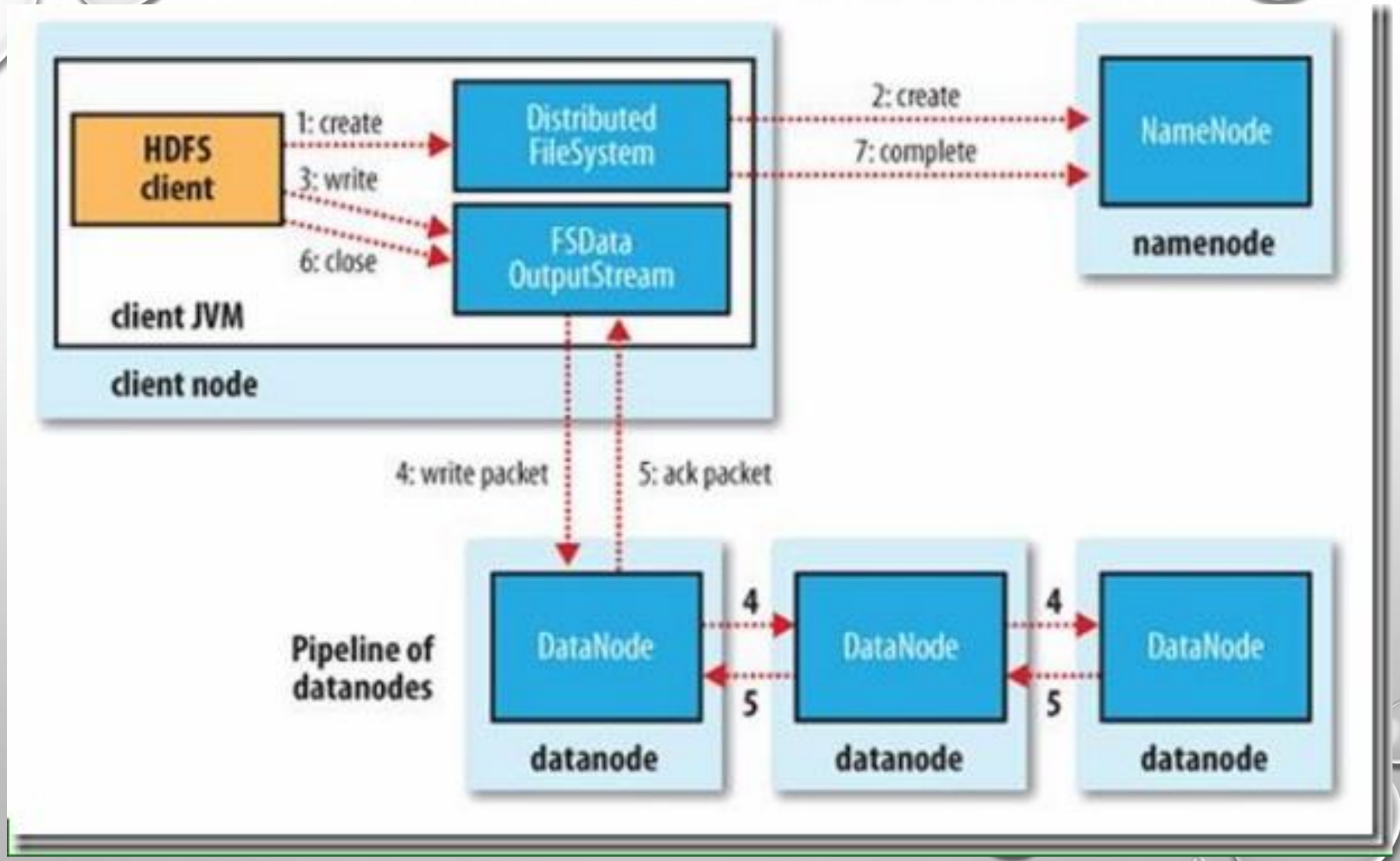
8.发送完block1后，再向host7，host8，host4发送block2

9.发送完block2后，host7，host8，host4向NameNode，host7向client发送通知

10.client向NameNode发送消息写入完成。

通过写入过程，可知：

- 写1T文件，需要3T存储，3T的网络流量
- 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在活动。如果发现DataNode故障，就将故障的DataNode上的数据放到其他节点上去。读取时去读其他节点
- 挂掉一个节点或者机架不会影响到整个系统的正常运行，因为有备份


HDFS client
1: create
3: write
6: close
Distributed FileSystem
FSData OutputStream
client JVM
client node
2: create
7: complete
NameNode
namenode
4: write packet
5: ack packet
Pipeline of datanodes
DataNode
4
5
DataNode
4
5
DataNode
datanode
datanode
datanode

1. 客户端调用create()来创建文件
2. DistributedFileSystem对NameNode创建一个RPC调用，在文件系统的命名空间中新建一个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. DistributedFileSystem向客户端返回一个FSDataOutputStream对象，由此客户端可以开始写入数据。写入数据时，DFSOutputStream将数据分成块，写入数据队列。
4. 数据队列由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管线（pipeline）里。Data Streamer将数据块写入管线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确认队列（ack queue），等待管线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。
7. 最后通知元数据节点写入完毕。

如果数据节点在写入DataNode的期间发生故障，则执行以下操作：

1. 关闭管线，确认把队列中的所有数据包都添加回数据队列的最前端，以确保故障节点下游的DataNode不会漏掉任何一个数据包。
2. 为存储在另一个正常DataNode的当前数据块制定一个新的标识，并将该标识传送给NameNode。以便故障DataNode在恢复后可以删除存储的部分数据块。
3. 将故障节点从管线中移除，余下的数据块则写入管线中的另外两个数据节点
4. NameNode被通知到复制块数不足时，会在另一个节点上创建一个新的副本，后续的数据块继续正常接收处理

数据复制

- NameNode全权管理block的复制，它周期性地从集群中的每个DataNode接收heartbeat和一个Blockreport。
- Heartbeat包的接收表示该DataNode节点正常工作，而Blockreport包括了该DataNode上所有的blok组成的列表。
- HDFS采用一种成为rack-aware的策略（不同机架）来改进数据的可靠性、有效性和网络带宽的利用，完成对副本的存放

容错机制总结

- 在NameNode和DataNode之间维持心跳检测，当由于网络故障之类的原因，导致DataNode发出的心跳包没有被NameNode正常收到的时候，NameNode就不会将任何新的IO操作派发给那个DataNode，该DataNode上的数据被认为是无效的，因此NameNode会检测是否有文件block的副本数目小于设置值，如果小于就自动开始复制新的副本并分发到其他DataNode节点。
- 检测文件block的完整性，HDFS会记录每个新创建的文件的所有block的校验和。当以后检索这些文件的时候，从某个节点获取blcok，会先确认校验和是否一致，如果不一致，会从其他DataNode节点上获取该block的副本。

- 集群的负载均衡，由于节点的失效或者增加，可能导致数据分布的不均匀，当某个DataNode节点的空闲空间大于一个临界值的时候，HDFS会自动从其他DataNode迁移数据过来
- NameNode上的fsimage和edits日志文件是HDFS的核心数据结构，如果这些文件损坏了，HDFS将失效。因此，NameNode可以配置成支持维护多个fsimage和edig log的拷贝。任何对fsimage或edit log的修改都将同步到它们的副本上
- 文件的删除，删除并不是马上从NameNode移出Namespace，而是放在/trash目录随时可以恢复的状态，知道超过设置时间才被正式删除

HDFS常用文件操作命令

- hadoop fs –ls：在HDFS中，没有当前目录的概念，没有cd命令？未带参数的-ls默认返回HDFS ”home” 目录下的内容
- hadoop fs –ls input：浏览HDFS下名为”input”的文档中文件
- hadoop fs –put ~/file test：通过put命令将本地目录下的file文件上传都HDfS上并重命名为test
- hadoop fs –get /user/hadoop/a.txt /：从HDFS传输文件到本地
- hadoop fs –rm /user/hadoop/a.txt：删除文件
- hadoop fs –rmr /user/hadoop/a.txt：迭代删除

- hadoop fs –get output getout：将HDFS中的”output”文件复制到本地系统并命名为”getout”
- hadoop fs –rmr newoutput：将HDFS下名为”newoutput”的文档删除
- hadoop fs –cat input/*：通过”-cat 文件”命令查看HDFS下input文件中的内容
- hadoop fs –copyFromLocal localsrc dst： 类似hadoop fs –put命令
- hadoop fs –moveFromLocal localsrc dst ：将本地文件上传到hdfs，同时删除本地文件
- hadoop fs –mkdir /user/hadoop：创建目录
- hadoop fs –text：将源文件输出为文本格式
- hadoop fs –help commandName：列出hdfs命令清单

HDFS管理命令

- hadoop dfsadmin –report：报告HDFS基本统计情况和状态
- hadoop dfsadmin –safemode enter：进入安全模式
- hadoop dfsadmin –safemode leave：退出安全模式
- start-balancer.sh：负载均衡（当新增DataNode或者DataNode出现故障时，HDFS的数据可能在DataNode中分布不均匀，使用该命令重新平衡DataNode上的数据块的分布）

添加新的DataNode节点

- 在新节点上安装好hadoop，和NameNode使用相同的配置（可复制）
- 修改”/usr/hadoop/conf/master”文件，加入NameNode主机名
- 在NameNode节点上修改”/usr/hadoop/conf/slaves”文件，加入新节点主机名
- 建立到新加点无密码的SSH连接
- 运行启动命令 start-all.sh

结束

谢谢！