初步掌握HDFS的架构及原理

转载自：https://www.cnblogs.com/codeOfLife/p/5375120.html

1、HDFS 是做什么的

HDFS（Hadoop Distributed File System）是Hadoop项目的核心子项目，是分布式计算中数据存储管理的基础，是基于流数据模式访问和处理超大文件的需求而开发的，可以运行于廉价的商用服务器上。它所具有的高容错、高可靠性、高可扩展性、高获得性、高吞吐率等特征为海量数据提供了不怕故障的存储，为超大数据集（Large Data Set）的应用处理带来了很多便利。

2、HDFS 从何而来

HDFS 源于 Google 在2003年10月份发表的GFS（Google File System） 论文。 它其实就是 GFS 的一个克隆版本

3、为什么选择 HDFS 存储数据

之所以选择 HDFS 存储数据，因为 HDFS 具有以下优点：

1、高容错性

- 数据自动保存多个副本。它通过增加副本的形式，提高容错性。
- 某一个副本丢失以后，它可以自动恢复，这是由 HDFS 内部机制实现的，我们不必关心。

2、适合批处理

- 它是通过移动计算而不是移动数据。
- 它会把数据位置暴露给计算框架。

3、适合大数据处理

- 处理数据达到 GB、TB、甚至PB级别的数据。
- 能够处理百万规模以上的文件数量，数量相当之大。
- 能够处理10K节点的规模。

4、流式文件访问

- 一次写入，多次读取。文件一旦写入不能修改，只能追加。
- 它能保证数据的一致性。

5、可构建在廉价机器上

- 它通过多副本机制，提高可靠性。
- 它提供了容错和恢复机制。比如某一个副本丢失，可以通过其它副本来恢复。

当然 HDFS 也有它的劣势，并不适合所有的场合：

1、低延时数据访问

- 比如毫秒级的来存储数据，这是不行的，它做不到。
- 它适合高吞吐率的场景，就是在某一时间内写入大量的数据。但是它在低延时的情况下是不行的，比如毫秒级以内读取数据，这样它是很难做到的。

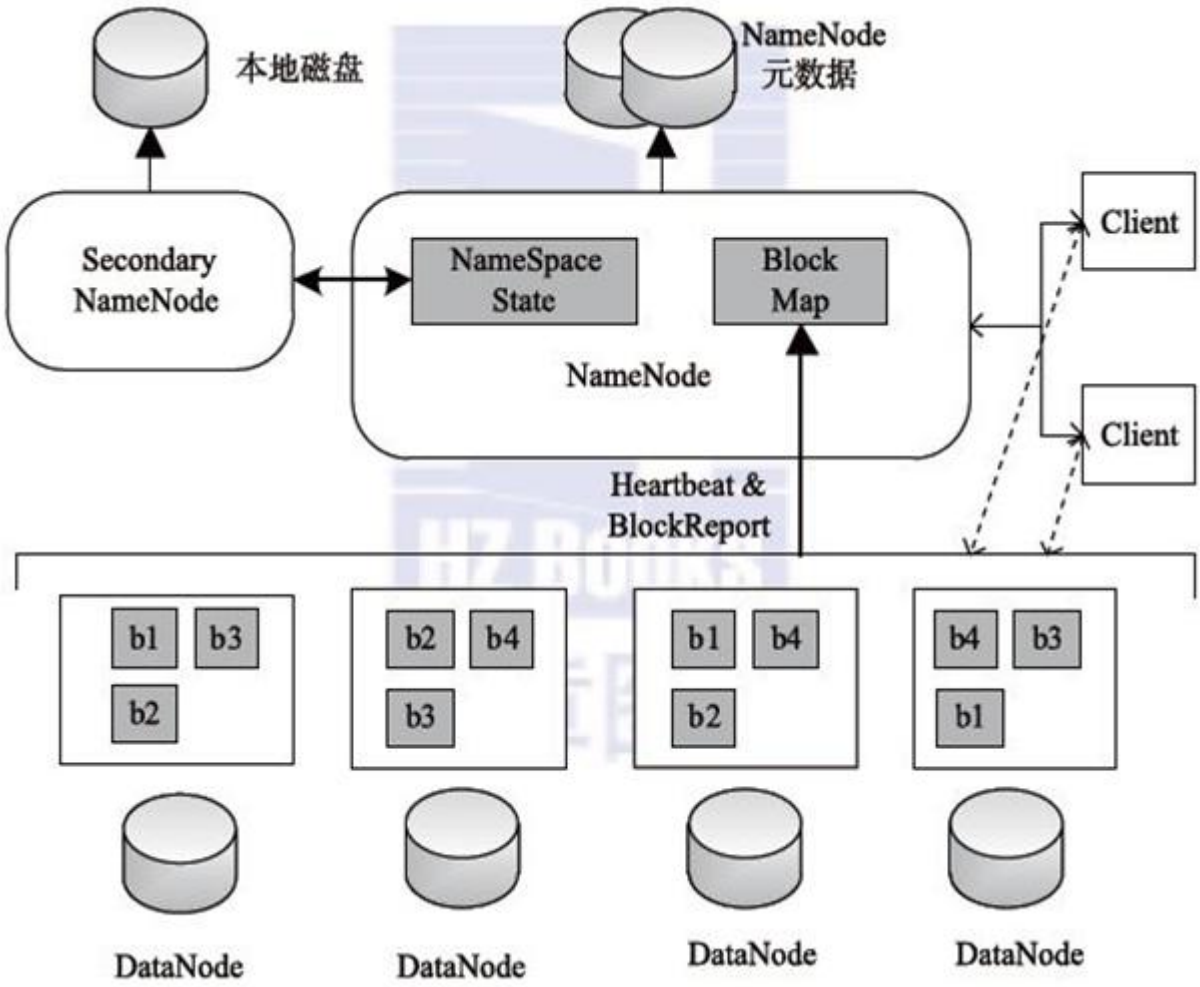
2、小文件存储

- 存储大量小文件(这里的小文件是指小于HDFS系统的Block大小的文件（默认64M）)的话，它会占用NameNode大量的内存来存储文件、目录和块信息。这样是不可取的，因为NameNode的内存总是有限的。
- 小文件存储的寻道时间会超过读取时间，它违反了HDFS的设计目标。

3、并发写入、文件随机修改

- 一个文件只能有一个写，不允许多个线程同时写。
- 仅支持数据 append（追加），不支持文件的随机修改。

4、HDFS 如何存储数据



HDFS的架构图

HDFS 采用Master/Slave的架构来存储数据，这种架构主要由四个部分组成，分别为HDFS Client、NameNode、DataNode和Secondary NameNode。下面我们分别介绍这四个组成部分

1、Client：就是客户端。

- 文件切分。文件上传 HDFS 的时候，Client 将文件切分成 一个一个的Block，然后进行存储。
- 与 NameNode 交互，获取文件的位置信息。
- 与 DataNode 交互，读取或者写入数据。
- Client 提供一些命令来管理 HDFS，比如启动或者关闭HDFS。
- Client 可以通过一些命令来访问 HDFS。

2、NameNode：就是 master，它是一个主管、管理者。

- 管理 HDFS 的名称空间
- 管理数据块（Block）映射信息
- 配置副本策略
- 处理客户端读写请求。

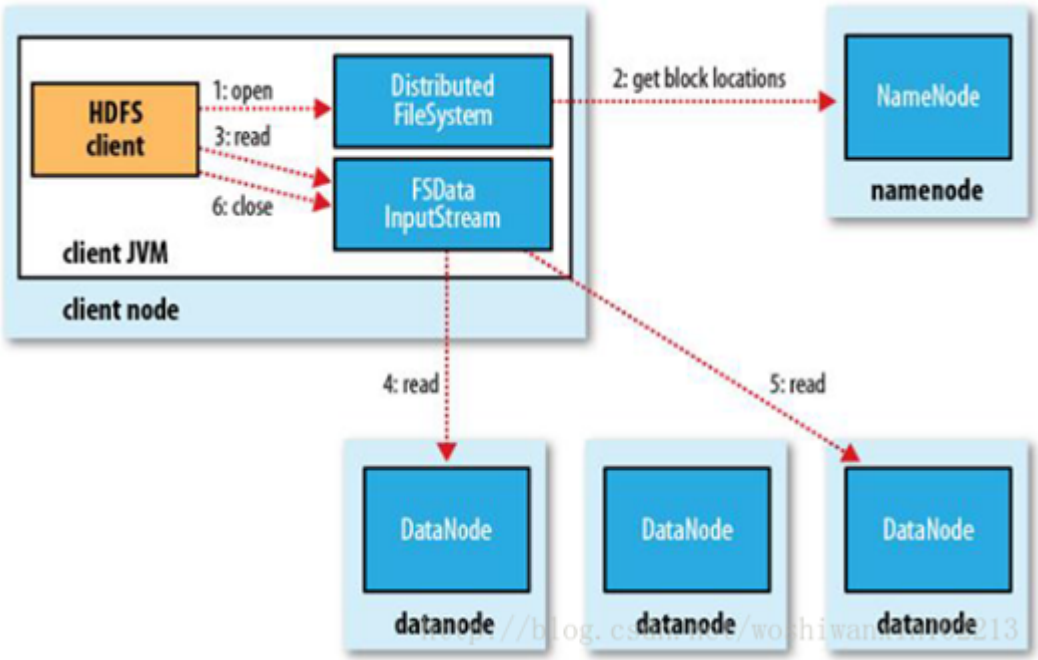
3、DataNode：就是Slave。NameNode 下达命令，DataNode 执行实际的操作。

- 存储实际的数据块。
- 执行数据块的读/写操作。

4、Secondary NameNode：并非 NameNode 的热备。当NameNode 挂掉的时候，它并不能马上替换 NameNode 并提供服务。

- 辅助 NameNode，分担其工作量。
- 定期合并 fsimage和fsedits，并推送给NameNode。
- 在紧急情况下，可辅助恢复 NameNode。

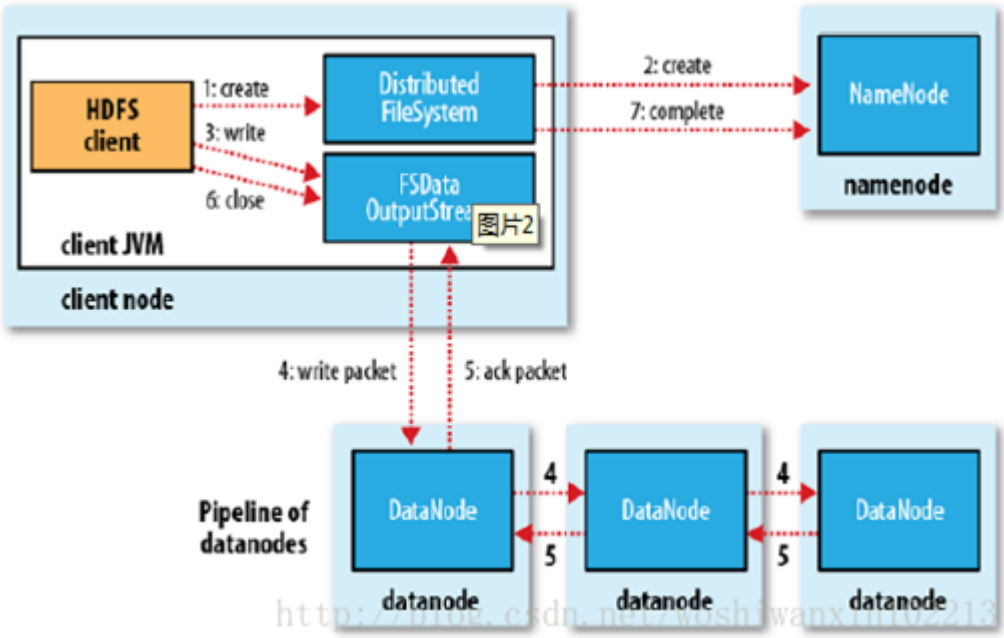
5、HDFS 如何读取文件



HDFS的文件读取原理，主要包括以下几个步骤：

- 首先调用FileSystem对象的open方法，其实获取的是一个DistributedFileSystem的实例。
- DistributedFileSystem通过RPC(远程过程调用)获得文件的第一批block的locations，同一block按照重复数会返回多个locations，这些locations按照hadoop拓扑结构排序，距离客户端近的排在前面。
- 前两步会返回一个FSDataInputStream对象，该对象会被封装成 DFSInputStream对象，DFSInputStream可以方便的管理datanode和namenode数据流。客户端调用read方法，DFSInputStream就会找出离客户端最近的datanode并连接datanode。
- 数据从datanode源源不断的流向客户端。
- 如果第一个block块的数据读完了，就会关闭指向第一个block块的datanode连接，接着读取下一个block块。这些操作对客户端来说是透明的，从客户端的角度来看只是读一个持续不断的流。
- 如果第一批block都读完了，DFSInputStream就会去namenode拿下一批blocks的location，然后继续读，如果所有的block块都读完，这时就会关闭掉所有的流。

6、HDFS 如何写入文件

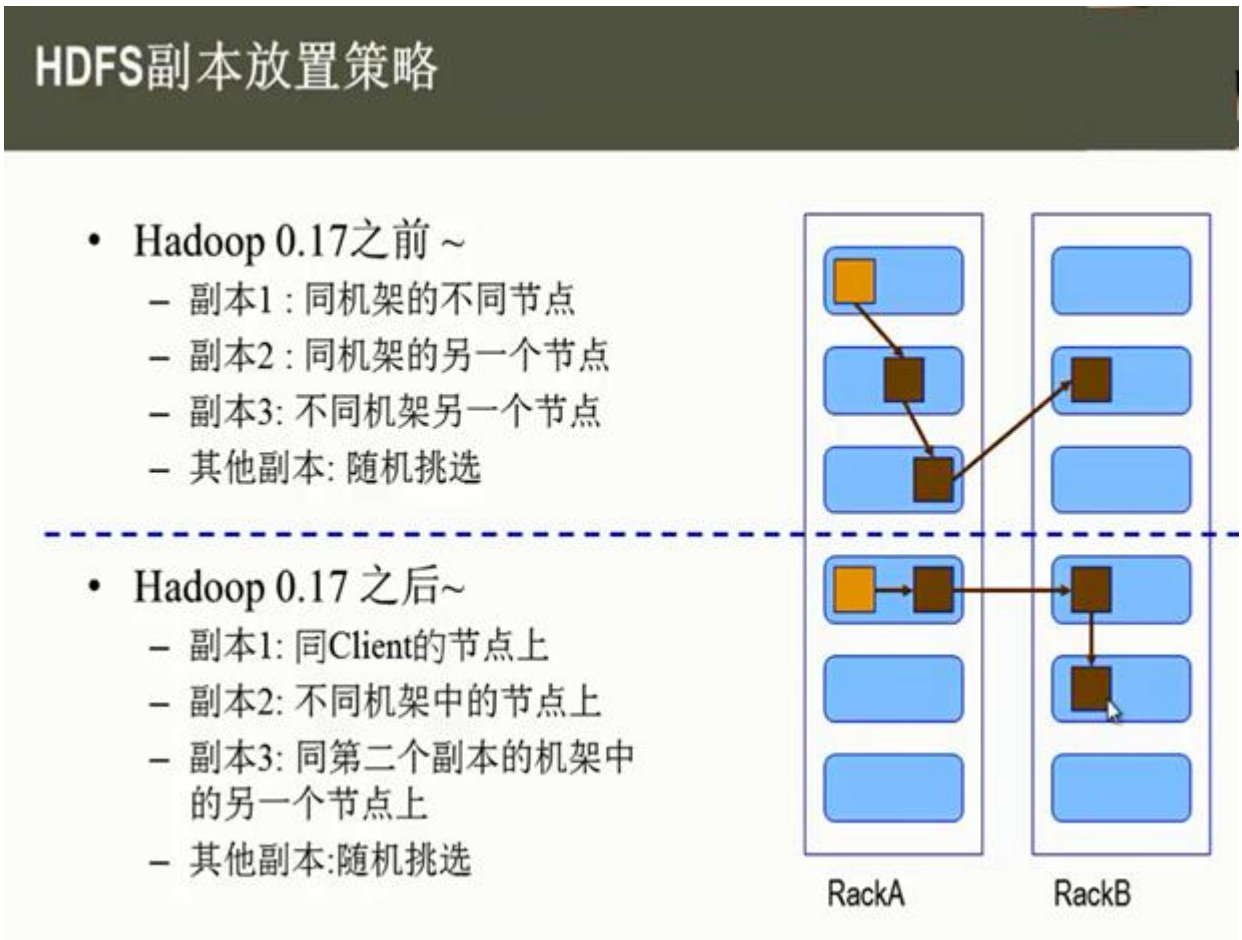


HDFS的文件写入原理，主要包括以下几个步骤：

- 客户端通过调用 DistributedFileSystem 的create方法，创建一个新的文件。
- DistributedFileSystem 通过 RPC（远程过程调用）调用 NameNode，去创建一个没有blocks关联的新文件。创建前，NameNode 会做各种校验，比如文件是否存在，客户端有无权限去创建等。如果校验通过，NameNode 就会记录下新文件，否则就会抛出IO异常。
- 前两步结束后会返回 FSDataOutputStream 的对象，和读文件的时候相似，FSDataOutputStream 被封装成 DFSOutputStream，DFSOutputStream 可以协调 NameNode和 DataNode。客户端开始写数据到 DFSOutputStream,DFSOutputStream会把数据切成一个个小packet，然后排成队列 data queue。
- DataStreamer 会去处理接受 data queue，它先问询 NameNode 这个新的 block 最适合存储的在哪几个 DataNode里，比如重复数是3，那么就找到3个最适合的 DataNode，把它们排成一个 pipeline。DataStreamer 把 packet 按队列输出到管道的第一个 DataNode 中，第一个 DataNode又把 packet 输出到第二个 DataNode 中，以此类推。
- DFSOutputStream 还有一个队列叫 ack queue，也是由 packet 组成，等待DataNode的收到响应，当pipeline中的所有DataNode都表示已经收到的时候，这时akc queue才会把对应的packet包移除掉。
- 客户端完成写数据后，调用close方法关闭写入流。
- DataStreamer 把剩余的包都刷到 pipeline 里，然后等待 ack 信息，收到最后一个 ack 后，通知 DataNode 把文件标示为已完成。

7、HDFS 副本存放策略

namenode如何选择在哪个datanode 存储副本（replication）？这里需要对可靠性、写入带宽和读取带宽进行权衡。Hadoop对datanode存储副本有自己的副本策略，在其发展过程中一共有两个版本的副本策略，分别如下所示

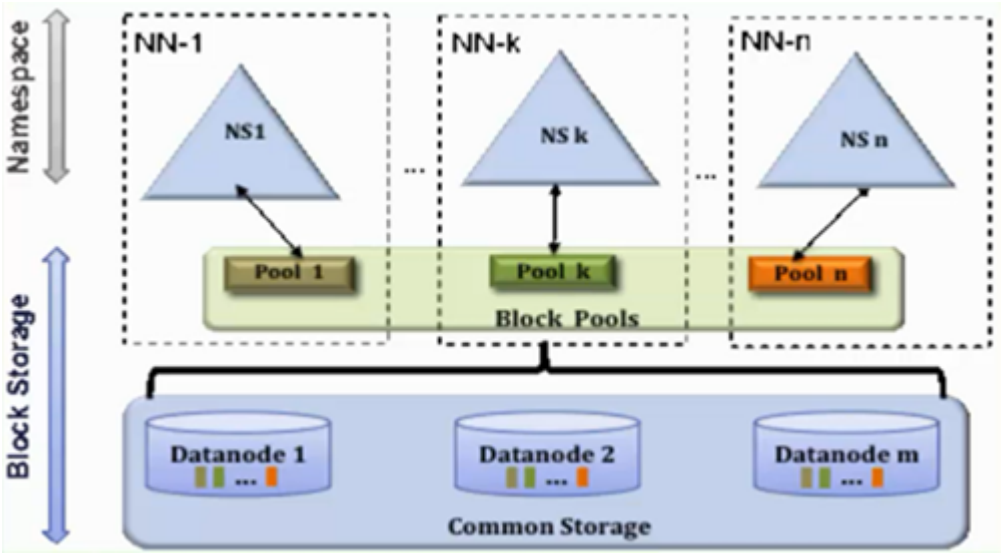


8、hadoop2.x新特性

- 引入了NameNode Federation，解决了横向内存扩展
- 引入了Namenode HA，解决了namenode单点故障
- 引入了YARN，负责资源管理和调度
- 增加了ResourceManager HA解决了ResourceManager单点故障

1、NameNode Federation

架构如下图



- 存在多个NameNode，每个NameNode分管一部分目录

- NameNode共用DataNode

这样做的好处就是当NN内存受限时，能扩展内存，解决内存扩展问题，而且每个NN独立工作相互不受影响，比如其中一个NN挂掉啦，它不会影响其他NN提供服务，但我们需要注意的是，虽然有多个NN，分管不同的目录，但是对于特定的NN，依然存在单点故障，因为没有它没有热备，解决单点故障使用NameNode HA

2、NameNode HA

解决方案：

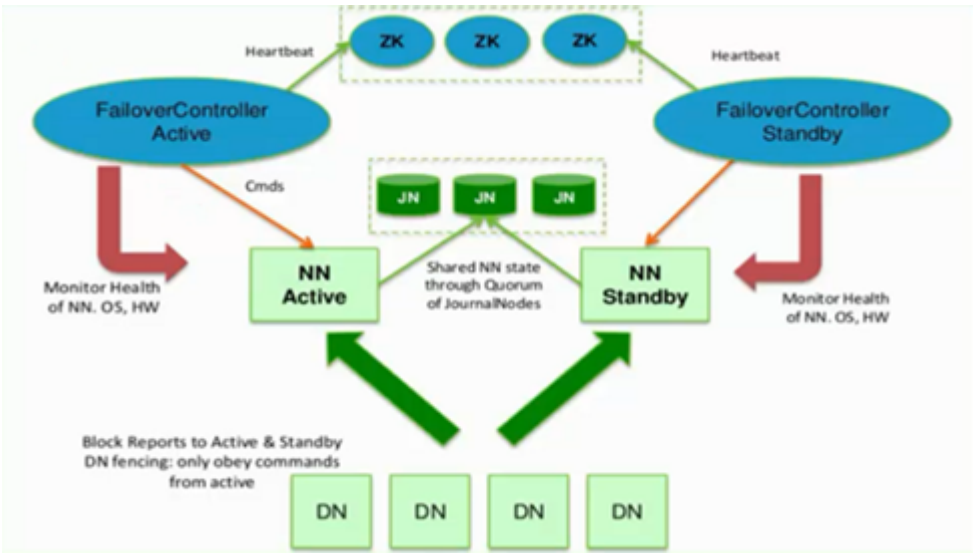
- 基于NFS共享存储解决方案
- 基于Qurom Journal Manager(QJM)解决方案

1、基于NFS方案

Active NN与Standby NN通过NFS实现共享数据，但如果Active NN与NFS之间或Standby NN与NFS之间，其中一处有网络故障的话，那就会造成数据同步问题

2、基于QJM方案

架构如下图



Active NN、Standby NN有主备之分，NN Active是主的，NN Standby备用的

集群启动之后，一个namenode是active状态，来处理client与datanode之间的请求，并把相应的日志文件写到本地中或JN中；

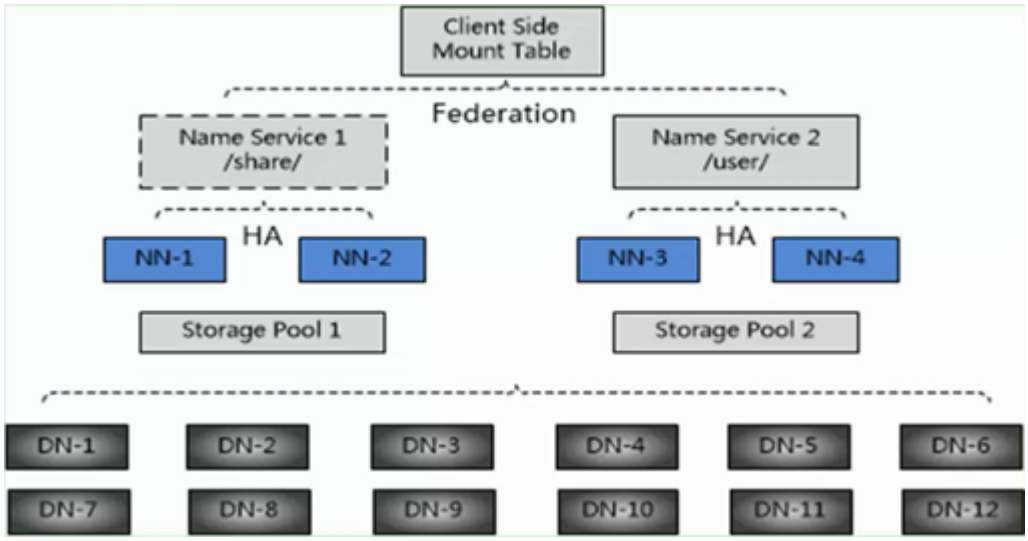
Active NN与Standby NN之间是通过一组JN共享数据（JN一般为奇数个，ZK一般也为奇数个），Active NN会把日志文件、镜像文件写到JN中去，只要JN中有一半写成功，那就表明Active NN向JN中写成功啦，Standby NN就开始从JN中读取数据，来实现与Active NN数据同步，这种方式支持容错，因为Standby NN在启动的时候，会加载镜像文件（fsimage）并周期性的从JN中获取日志文件来保持与Active NN同步

为了实现Standby NN在Active NN挂掉之后，能迅速的再提供服务，需要DN不仅需要向Active NN汇报，同时还要向Standby NN汇报，这样就使得Standby NN能保存数据块在DN上的位置信息，因为在NameNode在启动过程中最费时工作，就是处理所有DN上的数据块的信息

为了实现Active NN高热备，增加了FailoverController和ZK，FailoverController通过Heartbeat的方式与ZK通信，通过ZK来选举，一旦Active NN挂掉，就选取另一个FailoverController作为active状态，然后FailoverController通过rpc，让standby NN转变为Active NN

FailoverController一方面监控NN的状态信息，一方面还向ZK定时发送心跳，使自己被选举。当自己被选为主（Active）的时候，就会通过rpc使相应NN转变Active状态

3、结合HDFS2的新特性，在实际生成环境中部署图



这里有12个DN,有4个NN，NN-1与NN-2是主备关系，它们管理/share目录；NN-3与NN-4是主备关系，它们管理/user目录