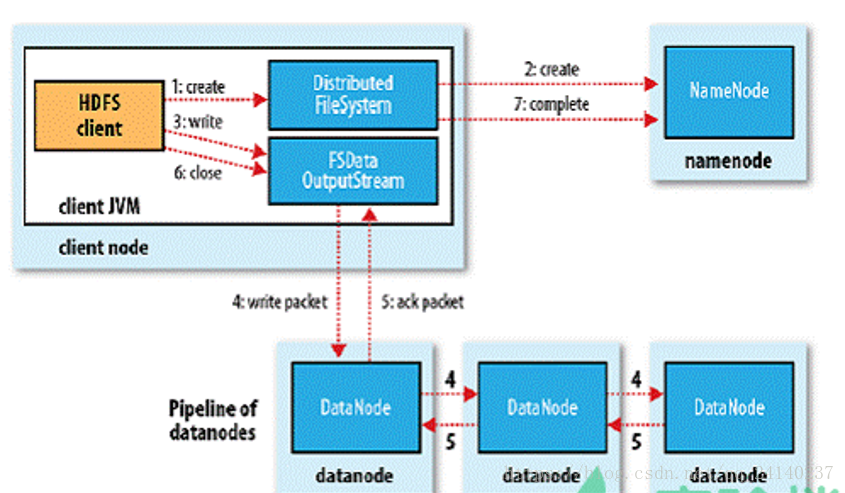
## HDFS

### 写



HDFS的文件写入原理，主要包括以下几个步骤：

1，客户端通过调用DistributedFileSystem的create()方法创建新文件；

2，DistributedFileSystem通过RPC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会为创建新文件记录一条记录，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，

3，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的数据包，并写入内部队列称为“数据队列”(Data Queue)；

4，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，把他们排成一个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；

5，DFSOutputStream还有一个对列叫Ack Quene，也是有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 Quene才会把对应的Packet包移除掉；

6，客户端完成写数据后调用close()方法关闭写入流；

7，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NameNode把文件标示为已完成。

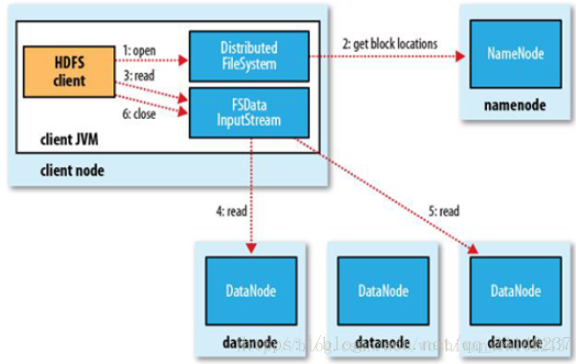
描述：

一个客户端创建一个文件的请求是不会立即发送到NameNode的，事实上在开始时，HDFS客户端会先把数据缓存到本地的一个缓冲区。应用程序会把数据库重定向写入到此缓冲区，当缓冲区累积的数据（当客户端向HDFS中写入数据的时候，首先会读取Hadoop的配置项，获取数据块的大小（大文件会被分割成多个Block进行存储，一般为64或128MB）以及备份数（每个Block会在多个Datanode上存储多份副本，一般为3份），之后将数据写到本地临时文件（缓存）中）超过配置的一个数据块大小时，客户端才会连接NameNode。

NameNode将文件名插入到文件系统层次结构中并为其分配存储空间，接着向客户端返回DataNode的标识和数据块的路径。然后客户端将数据块从本地缓冲区传输到指定的DataNode。

客户端会从Namenode获取一个Datanode列表用于存放数据块（Datanaode列表列出了存储数据块的地址，并根据距离对他们进行了排序）。然后客户端开始向第一个Datanode传输数据，第一个 Datanode 一小部分一小部分地接收数据，将每一部分写入本地仓库，并同时传输该部分到列表中 第二个 Datanode 节点。第二个 Datanode 也是这样，一小部分一小部分地接收数据，写入本地 仓库，并同时传给第三个 Datanode 。最后，第三个 Datanode 接收数据并存储在本地。当完成该数据块的存储后，Datanode会向Namenode报告数据传输完成，Namenode通知客户端该数据块已成功存储并复制在HDFS中，客户端继续重复发送下个数据块，直至所有数据块传送完成。

### 读



HDFS的文件读取原理，主要包括以下几个步骤：

1，客户端通过调用FileSystem对象的open()方法来打开希望读取的文件，对于HDFS来说，这个对象是分布文件系统的一个实例；

2，DistributedFileSystem通过使用RPC（远程过程调用）来调用NameNode以确定文件起始块的位置，同一Block按照重复数会返回多个位置，这些位置按照Hadoop集群拓扑结构排序，距离客户端近的排在前面；

3，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，客户端对这个输入流调用read()方法；

4， 存储着文件起始块的DataNode地址的DFSInputStream随即连接距离最近的DataNode，通过对数据流反复调用read()方法，可以将数据从DataNode传输到客户端；

5，到达块的末端时，DFSInputStream会关闭与该DataNode的连接，然后寻找下一个块的最佳DataNode，这些操作对客户端来说是透明的，客户端的角度看来只是读一个持续不断的流；

一旦客户端完成读取，就对FSDataInputStream调用close()方法关闭文件读取。

描述：

客户端从HDFS中读取文件，首先向Namenode请求要读取文件的信息，Namenode递给客户端数据块列表，客户端知道了有多少个数据块需要下载，也清楚了储存每个数据块的Datanode位置，就会逐个下载所有数据块（在写数据过程中，数据存储已经按照客户端与DataNode节点之间的距离进行了排序，距客户端越近的DataNode节点被放在最前面，客户端会优先从本地读取该数据块）

### Secondary Namenode

作用：

Hadoop会维护一个fsimage文件，也就是NameNode中metedata的镜像，但是fsimage不会随时与NameNode内存中的metedata保持一致，而是每隔一段时间通过合并edits文件来更新内容。Secondary NameNode就是用来合并fsimage和edits文件来更新NameNode的metedata的，它会从NameNode上下载元数据信息（fsimage,edits），然后把二者合并，生成新的fsimage，在本地保存，并将其推送到NameNode，替换旧的fsimage。但这**不是热备份**。

（1）secondary通知namenode切换edits文件；

（2）secondary从namenode获得fsimage和edits(通过http)；

（3）secondary将fsimage载入内存，然后开始合并edits；

（4）secondary将新的fsimage发回给namenode；

（5）namenode用新的fsimage替换旧的fsimage；

checkpoint：

**fs.checkpoint.period** 指定两次checkpoint的最大时间间隔，默认3600秒。   
 **fs.checkpoint.size**    规定edits文件的最大值，一旦超过这个值则强制checkpoint，不管是否到达最大时间间隔。默认大小是64M。

