HDFS读写过程

写数据流程

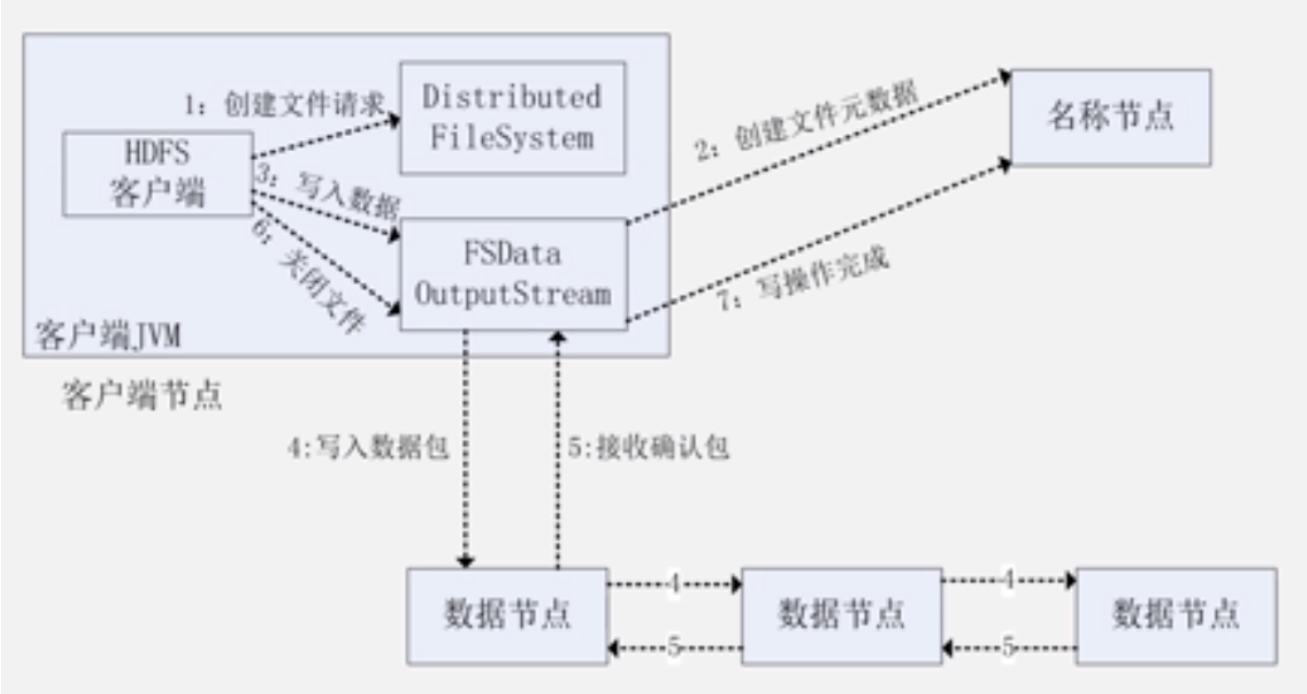
- 使用HDFS客户端向远程Namenode发送RPC请求
- Namenode会检查要创建的文件是否已经存在，创建者是否有权限进行操作，成功则会文件文件创建一个记录，否则会让客户端抛出异常
- 当客户端开始写入文件的时候，客户端会将文件分为多个packets，并在内部以数据队列的形式管理这些packets，并向Namenode申请blocks，获取用来存储复本的合适的Datanode列表
- 开始用管道的形式将packet写入所有的复本中，开发库将packet以流的方式写入第一个Datanode，该Datanode将该packet储存之后，再将其传递给在此管道中的下一个Datanode，知道最后一个Datanode，这种写数据方式呈流水线形式
- 最后一个Datanode成功储存之后会返回一个ack packet确认序列，在管道中传递到客户端，在客户端的开发库中维护这个ack queue，成功收到Datanode返回的ack packet后会从ack queue移除相应的packet
- 如果传输过程中，有某个Datanode出现了故障，那么当前管道就会被关闭，出现故障的Datanode会从当前的管道中移除，剩余的块会继续在剩下的Datanode中以管道的方式进行传输，同时Namenode会重新分配一个新的Datanode，保持复本的数量
- 客户端完成数据的写入之后，会对数据流调用close（）方法，关闭数据流

FileSystem API写入调用流程

入下图所示，下面对该图进行解析，可以参考上部分的写入流程

- HDFS客户端通过在**DistributedFileSystem** 上调用create()方法来创建一个文件夹，DistributedFileSystem使用RPC呼叫Namenode，在文件系统的命名空间上创建一个没有任何块关联的新文件，Namenode会执行各种各样的检查以确认文件是否存在，并检查客户端是否有创建文件的权限，如果检查通过，Name就会为新文件生成一条记录，否则，抛出IO异常。
- 成功之后，DistributedFileSystem会返回一个**FSDataOutputStream** 给客户端然他开始写数据，和读流程一样，**FSDataOutputStream包装 了一个DFSOutputStream** ，让他掌握了与Datanode和Namenode的联系
- 客户端开始写数据，DFSOutputStream将文件分成很多很小的数据，然后将小数据放进一个数据包，包中处理数据还有描述数据用的标识，包会写入**数据队列** ，数据队列被DataStreamr消费，他负责要求Namenode去挑选合适的存储块备份Datanode的一个列表，这个列表会构成一个**pipeline** （管线，管道）
- 在pipeline中有三个Datanode（默认三个复本），DataStreamer将能够组成块的包先流入pipeline中的第一个Datanode，第一个Datanode会先窜出来到的包，然后继续讲包转交到pipeline的第二个Datanode中，直到最后一个Datanode

- DFSOutputStream会维护一个包的内部队列，其中会有所有的数据包，该队列等待Datanode的写入确认，所以叫做ack quence（确认队列），当一个包已经被pipeline中的所有Datanode写入成功时，才会给客户端返回一个确认序列，然后将确认队列中相应的数据包移除；如果失败，那么就关闭pipeline，在确认队列中的剩下的包会被添加进数据队列的起始位置，以至于在在失败的节点下游的任何检点都不会丢失任何的包。



读取数据流程

- HDFS客户端向远程Namenode发起RPC请求
- Namenode会视情况返回文件的部分或者全部block列表，对于每个block，Namenode都会返回有该block拷贝的DataNode地址
- 客户端会选取离客户端最近的DataNode来读取数据block，如果客户端本身就是DataNode，那么将从本地直接读取
- 读取完当前block的数据之后，关闭当前的Datanode链接，并为读取下一个block寻找最佳的DataNode
- 当读完列表的block之后，且文件读取还没结束，客户端会继续向Namenode读取下一批的block列表
- 读取完一个block之后，都会进行checksum验证，如果读取datanode时出现错误，客户端会通知Namenode，然后会从下一个复本读取

FileSystem API读取调用流程

- 客户端通过FileSystem上调用open()方法打开它想要打开的文件，对于HDFS来说，就是在 **DistributedFileSystem** 的实例上调用
- DistributedFileSystem使用RPC去呼叫Namenode，来获取组成文件的前几个块的位置
- 对于每一个块，Namenode返回拥有块复本的Datanode的地址（这些Datanode会按照与客户端的接近度来排序），如果客户端节点运行在Datanode上，那么就直接从本地读取数据块
- DistributedFileSystem返回一个**FSDataInputStream** 给客户端，客户端就能从流中读取数据，FSDataInputStream中封装了一个管理**Datanode和Namenode I/O的DFSInputStream**

- 然后客户端调用read()方法，存储了文件的前几个块的地址的DFSInputStream，就会链接存储了第一个块的Datanode，然后DFSInputStream就通过重复调用read()方法，数据就从Datanode流向了客户端
- 当Datanode中的租后一个快读取完成，DFSInputStream会关闭与Datanode的连接，然后为下一个快寻找最佳节点，这个过程对用户透明
- 客户端完成读取，就在FSDataInputStream上调用close()方法结束整个流程
- 在读取过程中，如果FSDataInputStream在和一个Datanode进行交流时出现了一个错误，他就去连接下一个离自己最近的块，并记住刚刚发生错误的Datanode以至于以后不会访问这个Datanode，DFSInputStream会在Datanode上传输的数据上检查**checknums**，如果发现损坏，DFSInputStream就试图从另一个拥有备份的Datanode中读取备份数据。