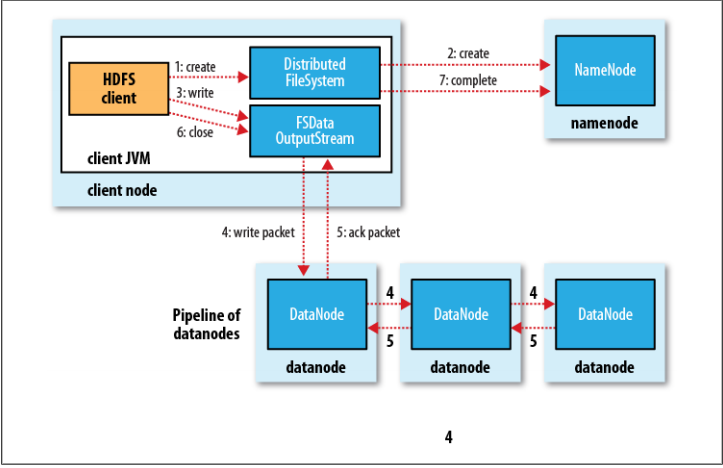
HADOOP整理

段卫东

# hadoop put 命令

  下面我们来先看看下面的“写”流程图：   
  
  
        假如我们有一个文件test.txt，想要把它放到Hadoop上，执行如下命令：

**引用**

        # hadoop fs  -put  /usr/bigdata/dataset/input/20130706/test.txt   /opt/bigdata/hadoop/dataset/input/20130706  //或执行下面的命令   
        # hadoop fs -copyFromLocal /usr/bigdata/dataset/input/20130706/test.txt  /opt/bigdata/hadoop/dataset/input/20130706        

        整个写流程如下：   
        第一步，客户端调用DistributedFileSystem的create()方法，开始创建新文件：DistributedFileSystem创建DFSOutputStream，产生一个RPC调用，让NameNode在文件系统的命名空间中创建这一新文件；   
        第二步，NameNode接收到用户的写文件的RPC请求后，谁偶先要执行各种检查，如客户是否有相关的创佳权限和该文件是否已存在等，检查都通过后才会创建一个新文件，并将操作记录到编辑日志，然后DistributedFileSystem会将DFSOutputStream对象包装在FSDataOutStream实例中，返回客户端；否则文件创建失败并且给客户端抛IOException。   
        第三步，客户端开始写文件：DFSOutputStream会将文件分割成packets数据包，然后将这些packets写到其内部的一个叫做data queue(数据队列)。data queue会向NameNode节点请求适合存储数据副本的DataNode节点的列表，然后这些DataNode之前生成一个Pipeline数据流管道，我们假设副本集参数被设置为3，那么这个数据流管道中就有三个DataNode节点。   
        第四步，首先DFSOutputStream会将packets向Pipeline数据流管道中的第一个DataNode节点写数据，第一个DataNode接收packets然后把packets写向Pipeline中的第二个节点，同理，第二个节点保存接收到的数据然后将数据写向Pipeline中的第三个DataNode节点。   
        第五步，DFSOutputStream内部同样维护另外一个内部的写数据确认队列——ack queue。当Pipeline中的第三个DataNode节点将packets成功保存后，该节点回向第二个DataNode返回一个确认数据写成功的信息，第二个DataNode接收到该确认信息后在当前节点数据写成功后也会向Pipeline中第一个DataNode节点发送一个确认数据写成功的信息，然后第一个节点在收到该信息后如果该节点的数据也写成功后，会将packets从ack queue中将数据删除。   
        在写数据的过程中，如果Pipeline数据流管道中的一个DataNode节点写失败了会发生什问题、需要做哪些内部处理呢？如果这种情况发生，那么就会执行一些操作：   
        首先，Pipeline数据流管道会被关闭，ack queue中的packets会被添加到data queue的前面以确保不会发生packets数据包的丢失；   
        接着，在正常的DataNode节点上的以保存好的block的ID版本会升级——这样发生故障的DataNode节点上的block数据会在节点恢复正常后被删除，失效节点也会被从Pipeline中删除；   
        最后，剩下的数据会被写入到Pipeline数据流管道中的其他两个节点中。   
        如果Pipeline中的多个节点在写数据是发生失败，那么只要写成功的block的数量达到**dfs.replication.min(默认为1)**，那么就任务是写成功的，然后NameNode后通过一步的方式将block复制到其他节点，最后事数据副本达到**dfs.replication参数**配置的个数。   
        第六步，，完成写操作后，客户端调用close()关闭写操作，刷新数据；   
        第七步，，在数据刷新完后NameNode后关闭写操作流。到此，整个写操作完成。