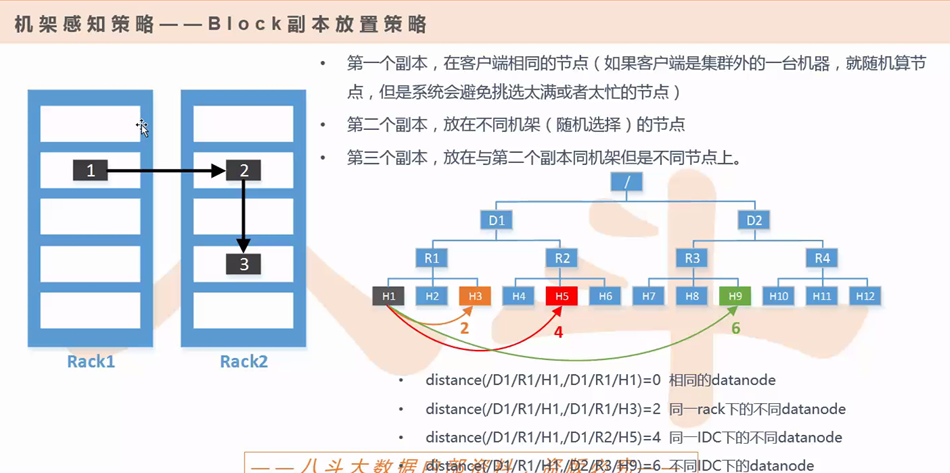
HDFS初步笔记2

图1：



机架感知策略它是直接和你的副本数有着直接的关系，比如说这个机器里面默认就是三个副本，那这个三个副本如果全部都存在同一个机器上面当然是不支持的，那不放在同一个机器上，放在在同一个机柜上那这个也是有风险的，比如这个机柜断电了，所以考虑到这么一种策略把，尽量的把这些风险分散开，然后按照上图1的左边那两个长方形的图，比如三个副本，一个机柜放一个副本，然后另外一个机柜放第二个副本，然后同时在放机柜第二个副本的同一机柜，然后随机找一个机器放第三个副本，但是这个时候又考虑到我们提交这个数据是什么方式提供的？就是说在集群的某一个节点上传的数据还是说跟这个集群完全没有任何关系的另外一台机器上传数据？那就是在客户端相同的节点上，但是如果说提交的任务是一个从节点上那么就是相当与在客户端所在的那个从节点的那台机器上找一个位置然后把数据写下来，那如果要是说客户端不属于集群里面的一个节点，那就是随机找一个节点放数据即可。

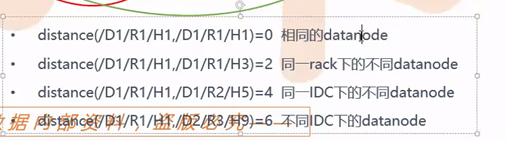
当然了选的时候不能选这个太满太忙的节点，必须要找一些空闲的节点，相当于这个时候不管是通过用哪种角度的话呢，已经是把第一个副本放好了。

那第二个副本在不同的机架上随机找一个节点放。

那第三个副本是在和第二个副本同一个机架上但是不同的节点上。

那对于内部策咯的时候就应该有一个距离计算，比如说在同一个机架同一个机器上，那么算这个距离的话相当于就没有绕出这个圈子，相当于在本地就可以找到这个数据，那这个的距离是0，因为在同一个机房同一个机架同一个机器上它的数据它的距离可能按照下图2的公式就可以看出是相同的datanode

图2：

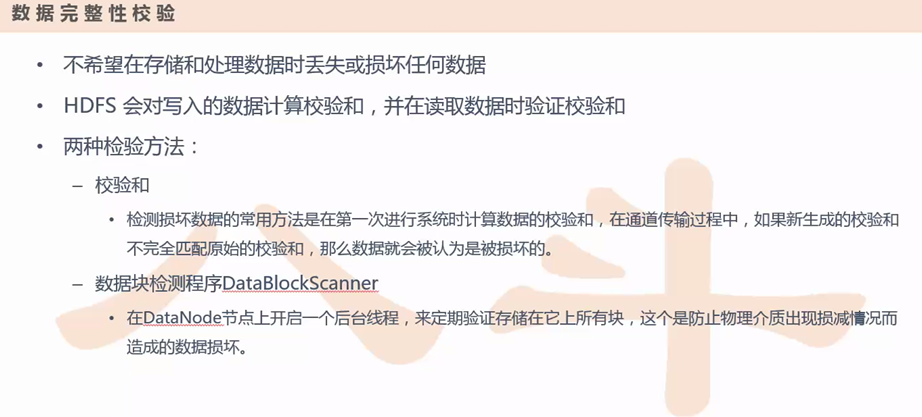


然后另外一个。就是在相同的机房相同的机柜但是不同的机器它的距离是2

如果是相同的机房不同的机柜但是这两个机柜属于同一个机房那么他们之间的距离就是4

那如果是连机房都不一样的话那就是6，这个在这里面就是相当于算是一个比较固定死的这么一个策略把，主要是判断两个datanode之间的一个远近程序来描述一个这么一个情况。

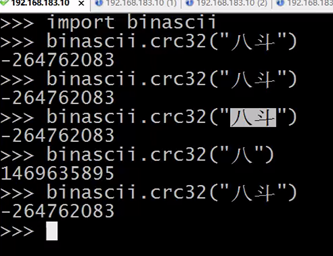
图3：



比如说为了保证你的数据的完整性，那如果来保证呢？通常来保证完整性的策略就是来通过一个校验，这个校验就相当是一个比如说张三跟李四寄了一封信，然后在这个信封上打了一个标记，那这个标记只有张三跟李四这两个人才能够认识的，其他人是不认识的，那张三就是在这信封上打了一个标记，然后李四看到了这个标记以后如果你是不看这个标记的话你也不确定这信封的内容到底有没有被创改过，如果你是没有这个标记的话，那创改过那这个李四也是不知道，所以为了效验这个信息这封信的有效性是不是真实的，是不是由张三寄出来的，那就看这信封上是否有标记了，相当于一个签名验证这么一个过程。

相当于在这个数据完整性效验这块就借助了这样的一个思想去做了一个效验来判断你这个数据，比如这个数据从客户端，数据肯定是从客户端写，写出来的时候和真正存到这个datanode节点上去的的这个数据，客户想存的数据和datanode存的数据是不是一个相同的数据，所以这需要做一些签名，通常在这个hdfs上做签名通过叫一个CRC32这么一个校验码。Python有一个校验模块，可以去试一试（如下图4所示）

图4：



怎样去用这样的签名呢？

签名在这个HDFS里面叫做一个校验和，其实校验和就是CRC32返回的结果。

两个校验方法：

第一个：client ->datanode上去写

这个时候client去写这个数据，比如说写了一些数据，然后它会对这些数据做一个CRC32校验码，但是它不会对整个的数据做校验码，如果对整个的数据做校验码数据量就比较大，那么就不会非常合适了。如果提前发现这个数据有问题的话可以提前去解决这样的问题。

那就是默认512字节创建一个校验码，那假如说每三个数据做一个校验码，那这个时候这个数据是从头往后传输的，那校验码就对三个数据做一个加密，那也就是说datanode要接收client传的数据时候肯定是接收已经加密的那三个数据，如果说datanode得到的这个校验码跟client传过来的校验码相同的话这三个加密的数据信息是准确无误的，那一旦判断这加密后的数据是准确无误的话，datanode就按步就班的写到了block里面。

所以中间数据出现问题了，就会把这数据丢弃了

如果要是说新生成的校验和你原始校验不匹配的话，那么数据就会被认为损坏

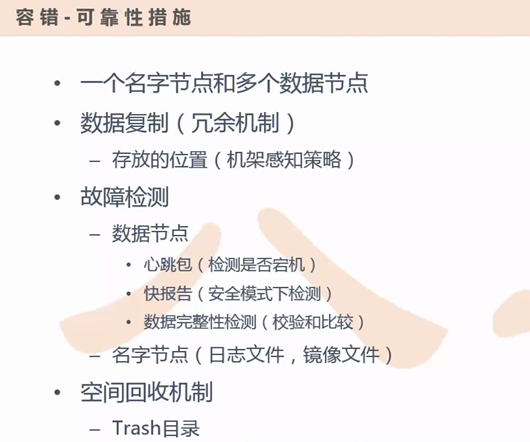
那还有一个检验方法那就是DataBlockScanner这样的一个后台的一个线程，那这个是做来定期检查的一个作用，它的定期的验证的存储在整个DataNode上的一个数据块，虽然这一开始这个数据块存储在本地是OK的，但是经过长时间存储，那这数据块有可能会发生一些个数据的变化，所以它要保证数据的可用性，那就是它会定期去检查你的数据块有没有出现一些变漏，一旦DataNode数据块检测出有问题那这时候就会在心跳的阶段DataNode会收到一个NameNode传过来的一个block修复命令，这个修复命令刚才说过了是个DataNode接收NameNode发送的一个block，那一旦DataNode收到这样的指令的话，那就是从其他的数据块中进行拷贝，拷贝一个新的数据块，然后把之前那个坏的数据块直接扔掉，然后重新达到了一个副本的平衡，这个是说的是client ->datanode单方向是吧？

那么反过来，如果是:

Datanode -》client去写数据去发送数据，相当于是client从Datanode 读数据，

其实读的时候，client也会有一个校验和的工作，它是怎么校验的呢？就是一堆数据假设从Datanode得到的，那你这数据从Datanode得到的时候其实它也会带一个校验和，只不过这个校验和它不会像上面之前所说的那样client会实时去写的，这个校验和它是仅仅是记住上一次校验的那个校验码，所以相对于它这个校验码不是那么的特别的实时。大家记住这么一个差别啊，那不管它是实时的还是不实时的，那client既然是收到这样的数据，那么就会对它做一个编码，然后看看编码看看是不是跟Datanode的一样的，然后去做一些验证，如果说是验证通过，那么这个时候要做一些操作，那就是更新时间：就是说那个验证码它是有时间的，如果要是client校验完之后，当前这个时间，相当于是对已经编码好的数据重新做一个确认，那确认这个时间也会告诉Datanode去更新它的一些相应的记录和日志，所以两个方向：client ->datanode和Datanode -》client都需要做校验的，除了这两个还有一个后端的服务就是DataBlockScanner，保证数据的完整性

图5：



还有一个并行的就是可靠性（如上图5所示）

这个可靠性措施也比较容易，可靠性怎么个可靠呢？其实这个就是之前我们讲的那三个副本就是干可靠性的，那如果副本不可靠的话那就说明，如果没有这样一个机制保证的话就不能够保证你的数据的高可用性，除了这一点还有哪些点能保证可靠性呢？

相当于这块就是对之前的内容做一个总结，那第一个：很明显这个NameNode有问题，是个单点，刚刚说过了，这个hadoop1.0里面有这样的缺陷，但是后面的版本里面系统就做进一步的复杂化的优化，使得NameNode就不再是系统的并进，这个后面再说。

那多个数据节点相当于是由多个DataNode对不对？多个DataNode的话呢还好，不像是这种单点故障，如果一个DataNode挂了的话，它会从别的地方进行恢复，不会影响你的存储然后不会影响你的数据的丢失，然后数据的复制就是刚刚我们说过的那个三个副本的问题，然后具体副本怎么存呢？就是有个机架感知策略。

另外一个就是说故障检测很重要，那故障检测怎么样来保证呢？那就是心跳，当然安全模式还有一个快的一个报告，用每一个block它会有一个report。

然后我们刚才讲过完整性的校验也算是一个，然后最后一个有一个名字节点（日志文件，镜像文件），名字节点那这个有点像最近开始说的那个fsimage和那个edits logs是不是？名字节点嘛就是在那个NameNode上的节点，这也是一个保障。

然后还有一个是空间回收机制，这个机制还是很重要的，就是经常会有人误操作，就是一不小心就把这个数据就丢失了，比如你在你本地机器上，你在你操作系统上你不小心删了个文件，你删了个文件你可能会觉得很沮丧是吧？但是你毕竟不会影响其他人，那你如果是在集群上删了一个文件，那不仅仅是影响你自己那所有的人的工作都会受到影响，有可能这样的风险，而且这个风险是非常的大，那为了进一步能够保证这样的一个情况，它是有一个回收站机制，叫做Trash目录，这个跟我们的windows不是有一个回收站吗？比如你一删除一个文件，那么其实这个文件没有删，它只是在回收站里暂时存一下，如果你突然发现这个失误操作，那你就可以在回收站把删除的文件再还原回来。

其实在hadoop里面在这个hdfs里面它是存在一个这样一个回收机制的，只不过是我们没有把它开放出来而已，那我们一会验证一下这个Trash目录，这Trash目录怎么去让它支持这样的一个功能呢？是需要我们去改一个配置文件叫做Core-site.xml这样的一个文件，想必大家应该知道把，这个在搭建集群的时候其中有一个文件就叫Core-site.xml，那具体一会怎么配置呢？我会跟大家说。

图6：



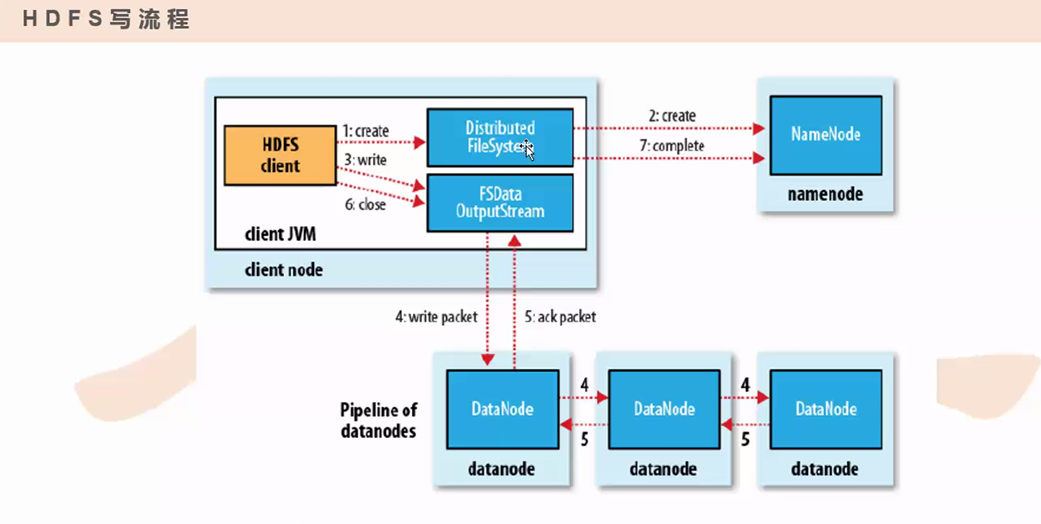
（如上图6所示）那HDFS特点是其实这里面就是把一些它的特点重新做一个整理把，那存储大数据这个就不说了，处理非结构化数据那刚才说了处理非结构化数据其实在任何地方都可以得到，比如说一些邮件啊，聊天记录啊，评论，照片，视频，pdf一些文档等等这都是一些非结构化的数据，当然非结构化数据从另外一个角度来看，这个怎么看呢？很多人都有这个数据库的开发经验，那数据库一般来说都是理解成一张表，那对于一张表的话它是有不同的字段有不同的记录是把？然后每一行是一条记录而且每一条字段约束着每一个字段它具体是什么样子的信息，这样的话会保证这个数据是非常的结构化非常的有条理非常整齐把！这个大家都应该好理解，所以在这个hdfs上面它就是一个你认为就是一个磁盘，那它存在什么样的数据它是没有办法约束的，所以在上面你可以存结构化数据和非结构化数据都可以，那具体怎么把这个非结构化的数据变成结构化的数据呢？那么这个我们后面会讲，比如说通过hive和Hbase都可以，特别是hive，怎么把一个非结构化的数据变成一个结构化数据。

另外一个这个数据的吞吐量，这个数据的吞吐量，就是HDFS它存的这个数据，它数据是非常大的，那它这个数据一个block比较大，那它通常普通的文件系统单位要大很多，那这个时候呢再去一个对数据进行一个寻址的话呢那它就会一直在这个数据从头到尾开始去读对不对？那这个时候就跟你的操作系统有一个优化有关，有一个叫预写不知道大家知道不知道？就是操作系统级别的有一个叫预写，相当于你读的上一条记录它自然就把你下一条记录提前给读出来，然后这样的话就可以提高数据的读取的一个速度，这就是一个非常大的一个优化，如果你要是对这个文件如果不是这个你的单位设的太小的话呢其实这个优势就完全就体现不出来了，所以这个时候有这样的优势呢，它就可以大大的提高数据的一个吞吐量。

然后还有一个就是说HDFS上的数据一旦写了那就不能修改了，这个请大家一定要记住，如果要修改怎办呢？其实是也可以修改的，但是这个不建议大家做，我们在本地读文件的时候你是可以从文件头然后通过一个指针偏移，你可以直接定位到这个文件的某一个位置进行一些个数据的插入对不对？但是严格来说你的HDFS数据其实也可以这么做，但是不建议这么做，这样的话会非常影响效率，所以通常来说这个文件你要是错了你就直接删了重搞一遍，另外一个呢你这个数据想更新怎么办呢？用open的追加到这个结尾，因为通过追加的方式会提高整个数据的一个吞吐效率，所以直接往文件结尾去追加就可以了，所以这个性能是基本没什么特别耗的，所以刚才有一些内容已经讲过了就是不适合做哪些，比如：存储小文件这个不建议，这个之前也说过了，大量随机读不建议，但是不建议不代表不能做，然后需要对文件的修改也是不支持，那最后一个多用户写入也是不支持的，这个上面说了，就是图6里面的write-once-read-many，就是一个人去写，然后多个人去读，就是不要多个人同时去写，因为你这个HDFS的数据它通常定位是这样的就是一旦这个数据库发了之后，这个数据就不会被变化了，但是对外更多的是以读的方法提供的。

好了我们看一下整个的HDFS一个操作流程把（如下图7所示）

图7：

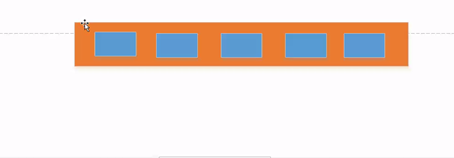


那如上图7所示，谁发送写呢？这个大家都应该知道把？那就是client客户端发送写，那么以谁提交任务？谁发出请求谁写嘛对吧？那就是client客户端去写，那client客户端你想写的话呢其实你得有个笔把对吧？起码有个工具是不是？那这个工具是什么呢？那就叫DistributedFileSystem，这是个什么东西呢？因为这个HDFS或者是hadoop它是由java开发的，那通常这样的一些接口和方法都是一些java的类，那DistributedFileSystem这个就是一个java的类，只不过是create是这个DistributedFileSystem类的方法，然后你可以通过这个方法去创建你的文件的行为的发起，就通过create这个方法去通过这样的一个方式去发起一个创建的一个行为，然后这种不是你想创建就一定能创建，你必须得得到某一个人的同意你才能去创建，那么这个人是谁呢？就是你的NameNode对不对？必须要让这个NameNode得到同意你才能去写才能去创建，否则的话没人告诉你应该往哪写，因为你这个client客户端想去写数据的话那肯定往一个DataNode上去写，但是你往哪写你都不知道，所以必须由NameNode告诉你这个client客户端往哪写，如果NameNode不让你写，那你也没办法写。

另外一个就是比如说你是银行甲方，然后有很多的外包工作来帮你去开发一个系统之内的，不是你这个甲方提任何需求，乙方都满足的，比如说就一种情况不满足，什么情况呢？

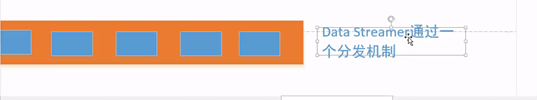
如果你写的这个路径，你压根就没有权限，那你肯定是写不了的，所以这种情况也是不能写的，另外一个呢你写的这个文件已经存在了，你也是不能写的，这个大家应该有经历是吧？像我们之前跑MapReduce任务的时候，我们那个OUTPUT路径，如果你要是没有删除的话你最后跑玩MapReduce之后肯定是报错的，它就会说你这个路径已存在，它就不会写了它就报错了，所以经常很多人去开发MapReduce的时候好不容易你这个任务已经跑起来了，然后跑了很长时间最后就报了个错，这个就需要去注意一下。所以这种情况你作为client客户端来说呢也是不能够去写，好了如果要是说这些问题都没有问题的话呢，那这个时候NameNode就让你去写，那让你去写的时候你得到了一些批准，然后你拿到这个批准之后就开始真正的去写，这个时候才涉及到了你真正应该用什么工具什么笔去写，那什么工具什么笔呢？那就是FSDataOutputStream这个东西，那FSDataOutputStream这个工具是谁给你的呢？这个FSDataOutputStream工具是由NameNode借助DistributedFileSystem来返回给你的一个对象类，也是一个java的类，然后紧接着你就可以用FSDataOutputStream这么一个类去开始做你真正的写的操作了，接下来客户端就开始写了，那怎么写呢？那客户端是不断的去生成数据，比如说我们要上传一个文件往HDFS上，那这个文件是在你本地，那你这个文件会被OutputStream分成一个一个的一个数据块，就会把你这个原始的数据会被这个Stream分成一个个小的数据块，然后把这个数据块会写到一个队列里面去（如下图8所示）

图8：



这个黄色就是队列，这个蓝色呢就是你想写的这个文件然后被Stream这个东西给你分成一个个数据块放入在队列里面。然后这个时候，接下来有一个东西，有一个叫做Data Streamer(如下图9所示)进行管理

图9：



然后相当于这个Data Streamer会读取你的队列，然后读取的时候就会有一个分发机制，数据被Stream读，读完之后会通过一个分发机制，那接下来好理解了，什么叫分发？那就是多副本嘛对吧？把一个文件形成多个副本然后存在dataNode上。

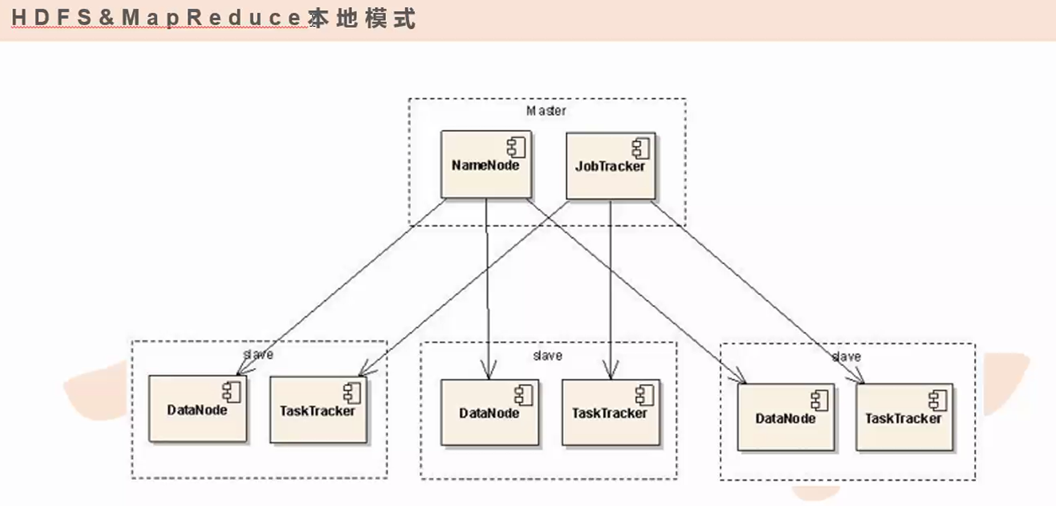
比如讲一个例子如图7中不是有三个dataNode对吧？这个时候就是dataNode去写的时候也是不是说按照并行去写，是按照顺序去写，相同的数据有三个副本对不对？我第一个副本先写第一个，第二个副本写第二个，第三个副本写第三个，但是这里面有一个细节大家要注意一下不是说你三个副本全部写完之后才返回给FSDataOutputStream写成功，比如说第一个副本写成功了，我就返回一个ack packet一个确认包，那这时候就有疑问了，那其实你这个写完然后这个并且给你返回成功了，其实按道理说你整个的环境里面其实都没有达到三个副本怎办呢？如果说这种情况的话那就是谁有这个数据就对外提供服务了，然后这剩下的两个机器就是没有这个数据那就让他们自己内部去做复制就可以了，然后一旦把这个数据块都在各自的机器上复制完有这样的数据的话呢，那其实这个时候也会进行提供服务。这个就很像我们之前说的弱一致性，所以这个流程大家没什么问题了把？

这个切分是客户端去切分的，还有一个需要注意一下，就是HDFS如果认为一个节点成功了，那就算成功了，然后后续呢本身自己会意识到如果你的副本数不足的话呢它就会进行一些复制，把那些冗余的数据复制过来，意就是把它的本尊复制过来作为冗余，那么我们这是写流程啊，我们再看一下读流程。

写流程搞的明白的话相对于读流程就更简单了，读流程怎么读呢？其实它也是要求和NameNode去做一个申请，这个怎么申请呢？其实这个流程跟刚才写的流程一样，那么一样的话呢那接下来就真正要去读的时候也是要借助一个intputStream，就是我去写数据的时候这里面是OutputStream，那么读的时候是intputStream，那读的时候就比较容易了，是因为你的NameNode返回给你客户端就已经告诉了你这个数据是存在于具体哪台机器上，所以读的时候直接定位到那个机器去读你的相应的block就可以了，所以这个相对于写来说很简单。

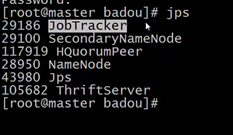
然后还有一个很重要的知识点（如下图10所示）

图10：



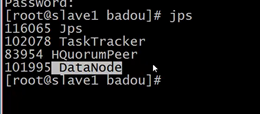
本地模式很重要，那你看这个上图10可以发现出什么样的情况？能发现NameNode，JobTracker，你会发现这个虚线是什么意思？这个意思就是一个机器，发现NameNode和JobTracker布置到一块了，其实我们最近一直用的这个集群就是这个样子，大家应该没问题把？我们来看一下（如下图11所示）

图11：



是吧？看上图11就知道JobTracker和NameNode布置到一块了

图12：



然后再看上图12所示，TaskTracker和DataNode布置到一块了，这有什么好处呢？这个好处就是你先看图11中的salve这个节点，salve节点上如果说因为你这个TaskTracker是负责计算的，因为它是mapreduce集成框架的一个节点，DataNode是你的存储数据的存储的节点，那通常如果要计算的这个节点它所要读的这个数据刚刚在你的这个DataNode本机，相当于就不需要在别的地方进行一个拷贝了，那这样的话就大大的提高了你的这个整个集群整个系统的性能，所以呢有什么事尽可能的在你的本地解决，这相当于是一个本地模式。

然后另外一个通常不管你是大集群还是小集群通常这个DataNode和TaskTracker都布置到一块了，还有一个大家可能会有一个质疑 那就是NameNode和JobTracker一定要布置到一块吗？就是我们在小集群的时候，你是可以布置到一块的，如果你要是真正的大型集群，正式的大型工业集群其实NameNode和JobTracker要分开布置，要分到两个不同的机器上去，这个大家注意一下。

好了，接下来关于整个的HDFS的初步就到这些，接下来我们做一个实践部分！！！至于实践部分我就在这不写了。