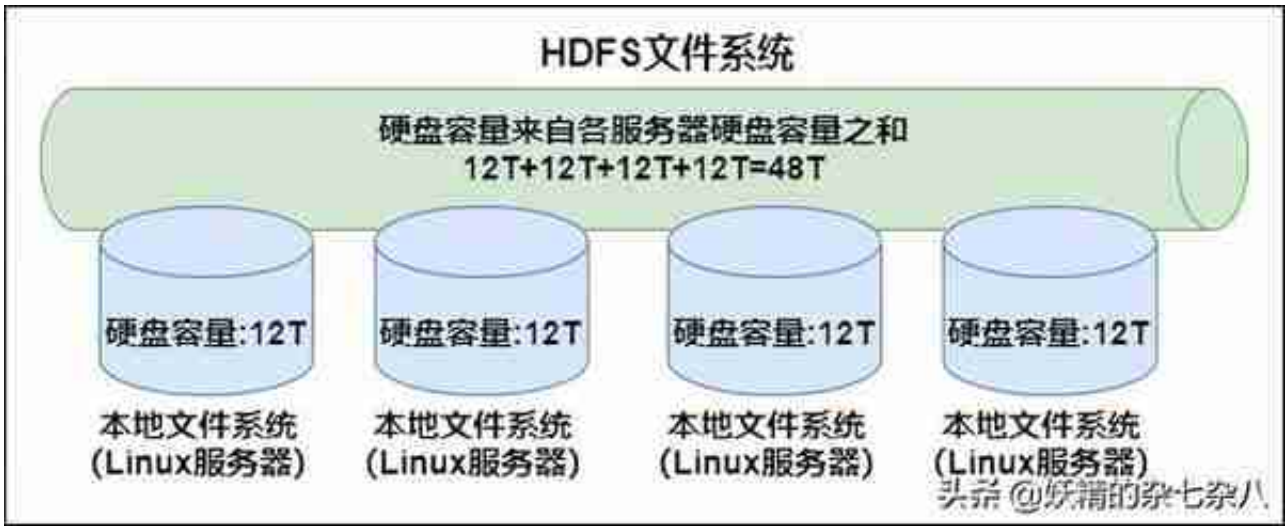
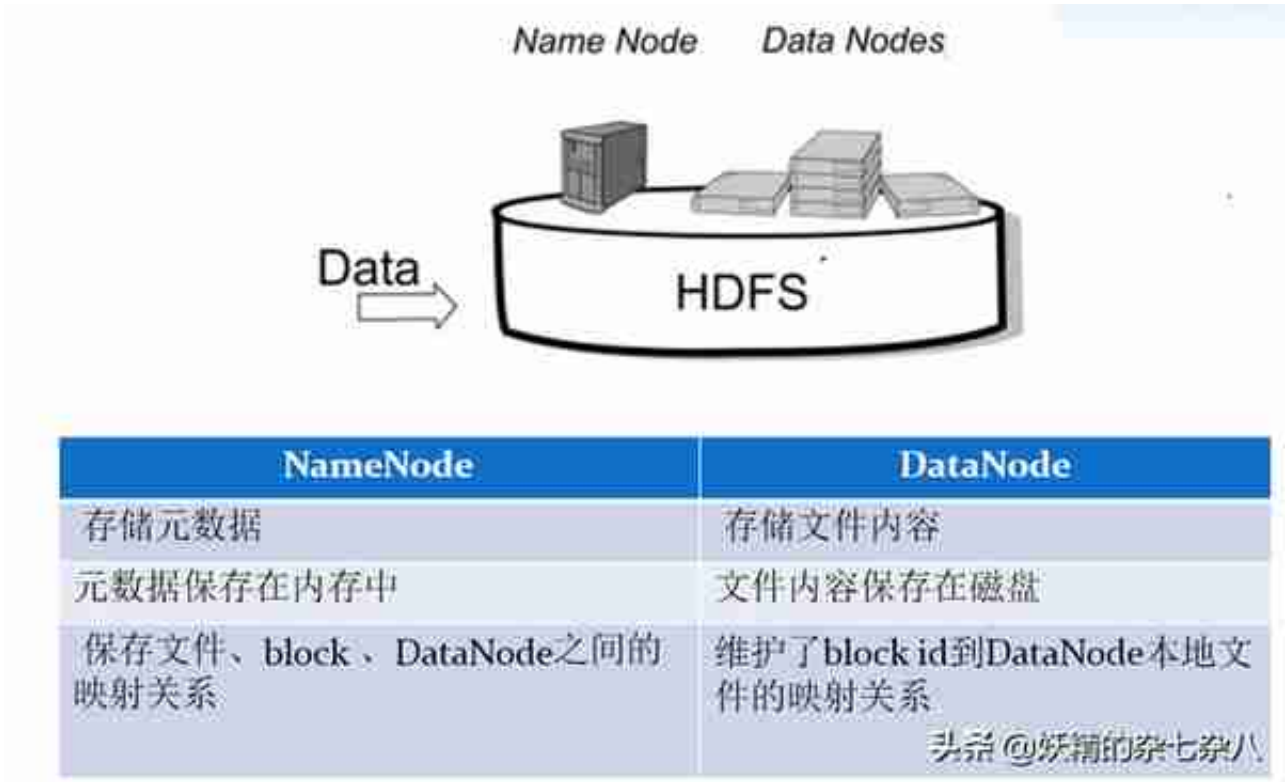


2. HDFS架构



NameNode	DataNode
存储元数据	存储文件内容
元数据保存在内存中	文件内容保存在磁盘
保存文件、block 、DataNode之间的映射关系	维护了block id到DataNode本地文件的映射关系

3. HDFS的特性

首先，它是一个文件系统，用于存储文件，通过统一的命名空间目录树来定位文件

；

其次，它是分布式的，由很多服务器联合起来实现其功能，集群中的服务器有各自的角色。

1. master/slave 架构（主从架构）

HDFS 采用 master/slave 架构。一般一个 HDFS 集群是有一个 Namenode 和一定数目的 Datanode 组成。Namenode 是 HDFS 集群主节点，Datanode 是 HDFS
集群从节点，两种角色各司其职，共同协调完成分布式的文件存储服务。

2. 分块存储

HDFS 中的文件在物理上是分块存储（block）的，块的大小可以通过配置参数来规定，默认大小在 hadoop2.x 版本中是 128M。

3. 名字空间（NameSpace）

HDFS 支持传统的层次型文件组织结构。用户或者应用程序可以创建目录，然后将文件保存在这些目录里。文件系统名字空间的层次结构和大多数现有的文件系统类似：用户可以创建、删除、移动或重命名文件。
Namenode 负责维护文件系统的名字空间，任何对文件系统名字空间或属性的修改都将被 Namenode 记录下来。
HDFS 会给客户端提供一个统一的抽象目录树，客户端通过路径来访问文件，形如：hdfs://namenode:port/dir-a/dir-b/dir-c/file.data。

4. NameNode 元数据管理

我们把目录结构及文件分块位置信息叫做元数据。NameNode 负责维护整个 HDFS 文件系统的目录树结构，以及每一个文件所对应的 block 块信息（block 的 id，及所在的 DataNode 服务器）。

5. DataNode 数据存储

文件的各个 block 的具体存储管理由 DataNode 节点承担。每一个 block 都可以在多个 DataNode 上。DataNode 需要定时向 NameNode 汇报自己持有的 block 信息。
存储多个副本（副本数量也可以通过参数设置 dfs.replication，默认是 3）

6. 副本机制

为了容错，文件的所有 block 都会有副本。每个文件的 block 大小和副本系数都是可配置的。应用程序可以指定某个文件的副本数目。副本系数可以在文件创建的时候指定，也可以在之后改变。

7. 一次写入，多次读出

HDFS 是设计成适应一次写入，多次读出的场景，且不支持文件的修改。正因为如此，HDFS 适合用来做大数据分析的底层存储服务，并不适合用来做网盘等应用，因为修改不方便，延迟大，网络开销大，成本太高。

4. HDFS 的命令行使用

如果没有配置 hadoop 的环境变量，则在 hadoop 的安装目录下的bin目录中执行以下命令，如已配置 hadoop 环境变量，则可在任意目录下执行

help

```
??:  hdfs dfs -help ??????: ?????????????
```

ls

```
???hdfs dfs -ls  URI??????Linux?ls?????????
```

lsr

```
??  :   hdfs  dfs -lsr URI??  : ??????????ls, ?UNIX??ls-R??
```

mkdir

?? ? hdfs dfs -mkdir [-p] <paths>?? : ?<paths>??URI????? ???????-p??????????

put

?? ? hdfs dfs -put <localsrc > ... <dst>?? ? ???????src? ??????srcs??????????????????<dst>?????????????????????????? ???

hdfs dfs -put /rooot/bigdata.txt /dir1

moveFromLocal

??? hdfs dfs -moveFromLocal <localsrc> <dst>??: ?put??? ???????localsrc?????????

hdfs dfs -moveFromLocal /root/bigdata.txt /

copyFromLocal

??: hdfs dfs -copyFromLocal <localsrc> ... <dst>??: ??????? ??????hdfs???

appendToFile

??: hdfs dfs -appendToFile <localsrc> ... <dst>??: ????????? ??hdfs?????.???????????.

hdfs dfs -appendToFile a.xml b.xml /big.xml

moveToLocal

? hadoop 2.6.4 ??????????????hadoop dfs -moveToLocal [-cr c] <src> <dst>??????????? HDFS

get

?? hdfs dfs -get [-ignorecrc] [-crc] <src> <localdst>?
????????????? CRC ?????????-ignorecrc????? ???CRC??????-
CRC????

hdfs dfs -get /bigdata.txt /export/servers

getmerge

??: hdfs dfs -getmerge <src> <localdst>??: ???????????hdfs??
? /aaa/??????:log.1, log.2,log.3,...

copyToLocal

??: hdfs dfs -copyToLocal <src> ... <localdst>??: ?hdfs???
??

mv

?? ? hdfs dfs -mv URI <dest>??? ?hdfs??????????????????
???????????????

hdfs dfs -mv /dir1/bigdata.txt /dir2

rm

??? hdfs dfs -rm [-r] ?-skipTrash? URI ?URI ??????? ????????
?????????? ??????????????????-skipTrash??????????????????
??????????????????????????HDFS Shell ??????????????????

hdfs dfs -rm -r /dir1

cp

??: hdfs dfs -cp URI [URI ...] <dest>??? ?????????????<de
st> ??????????????????-f??????????????-p?????????????
???????ACL?XAttr??

hdfs dfs -cp /dir1/a.txt /dir2/bigdata.txt

cat

```
hdfs dfs  -cat  URI [uri  ...]???????????????stdout

hdfs dfs  -cat /bigdata.txt
```

tail

```
??: hdfs dfs -tail path??: ????????
```

text

```
??:hdfs dfs -text path??: ?????????????
```

chmod

```
??:hdfs  dfs  -chmod  [-R]  URI[URI  ...]?????????????  -R
????????????????????????????????????????
```

```
hdfs dfs -chmod -R 777 /bigdata.txt
```

chown

```
??:  hdfs  dfs  -chmod  [-R]  URI[URI ...]???  ????????????
?????  -R ????????????????????????????????????????
```

```
hdfs  dfs  -chown  -R hadoop:hadoop  /bigdata.txt
```

df

```
??: hdfs dfs  -df  -h  path??: ????????????
```

du

```
??: hdfs dfs -du -s -h path??: ?????????
```

count

```
??: hdfs dfs -count path??: ???????????????
```

setrep

```
??:  hdfs dfs -setrep num filePath??: ??hdfs??????????: ????
????datanode???,???????????datanode???????
```

expunge (慎用)

```
??:  hdfs dfs  -expunge??: ??hdfs???
```

5. hdfs的高级使用命令

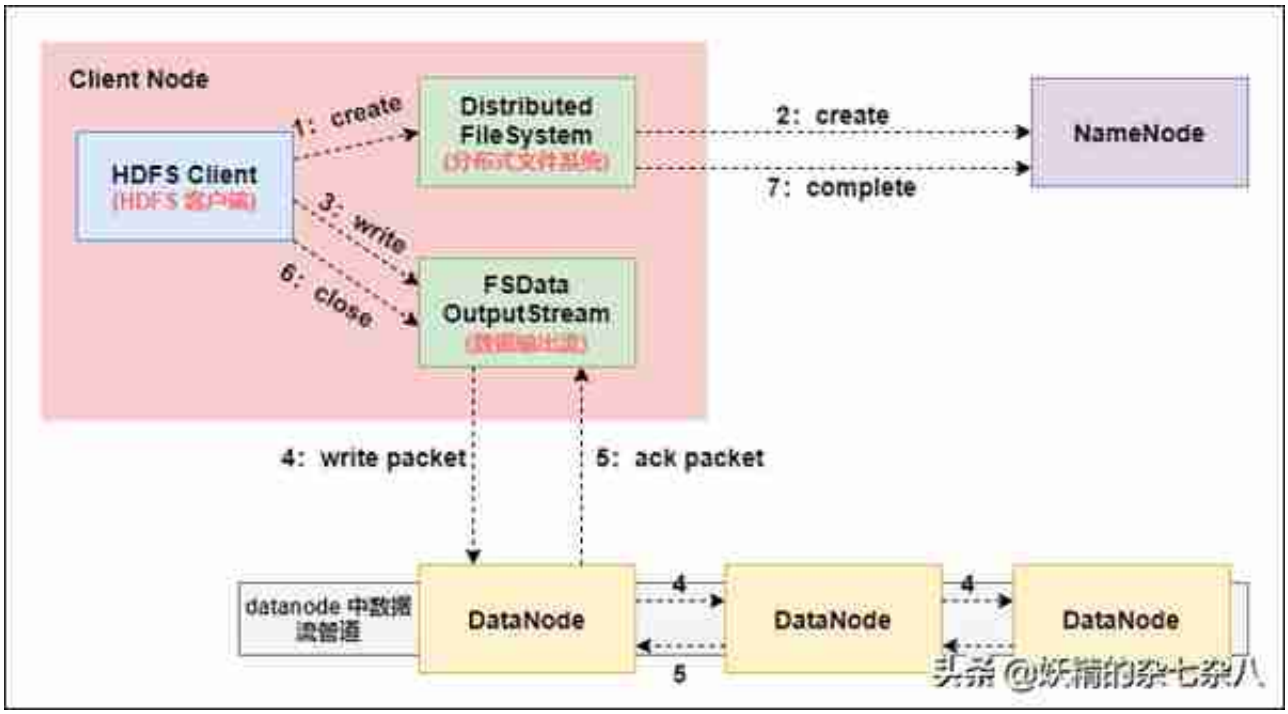
5.1 HDFS文件限额配置

在多人共用HDFS的环境下，配置设置非常重要。特别是在 Hadoop 处理大量资料的环境，如果没有配额管理，很容易把所有的空间用完造成别人无法存取。HDFS的配额设定是针对目录而不是针对账号，可以让每个账号仅操作某一个目录，然后对目录设置配置。

HDFS 文件的限额配置允许我们以文件个数，或者文件大小来限制我们在某个目录下上传的文件数量或者文件内容总量，以便达到我们类似百度网盘网盘等限制每个用户允许上传的最大的文件的量。

```
 hdfs dfs -count -q -h /user/root/dir1  #??????
```

结果：



1. Client 发起文件上传请求，通过 RPC 与 NameNode 建立通讯, NameNode
   检查目标文件是否已存在，父目录是否存在，返回是否可以上传；
2. Client 请求第一个 block 该传输到哪些 DataNode 服务器上；
3. NameNode
   根据配置文件中指定的备份数量及机架感知原理进行文件分配, 返回可用的 DataNode 的地址如：A, B, C；

   Hadoop 在设计时考虑到数据的安全与高效， 数据文件默认在 HDFS 上存放三份， 存储策略为本地一份，同机架内其它某一节点上一份，不同机架的某一节点上一份。

4. Client 请求 3 台 DataNode 中的一台 A 上传数据（本质上是一个 RPC 调用，建立 pipeline ），A 收到请求会继续调用 B，然后 B 调用 C，将整个 pipeline 建立完成， 后逐级返回 client；
5. Client 开始往 A 上传第一个 block（先从磁盘读取数据放到一个本地内存缓存），以 packet 为单位（默认64K），A 收到一个 packet 就会传给 B，B 传给 C。A 每传一个 packet 会放入一个应答队列等待应答；
6. 数据被分割成一个个 packet 数据包在 pipeline 上依次传输，在 pipeline 反方向上， 逐个发送 ack（命令正确应答），最终由 pipeline 中第一个 DataNode 节点 A 将 pipelineack 发送给 Client；

7. 当一个 block 传输完成之后，Client 再次请求 NameNode 上传第二个 block，重复步骤 2；

7.1 网络拓扑概念

在本地网络中，两个节点被称为“彼此近邻”是什么意思？在海量数据处理中，其主要限制因素是节点之间数据的传输速率——带宽很稀缺。这里的想法是将两个节点间的带宽作为距离的衡量标准。

节点距离：两个节点到达最近的共同祖先的距离总和。

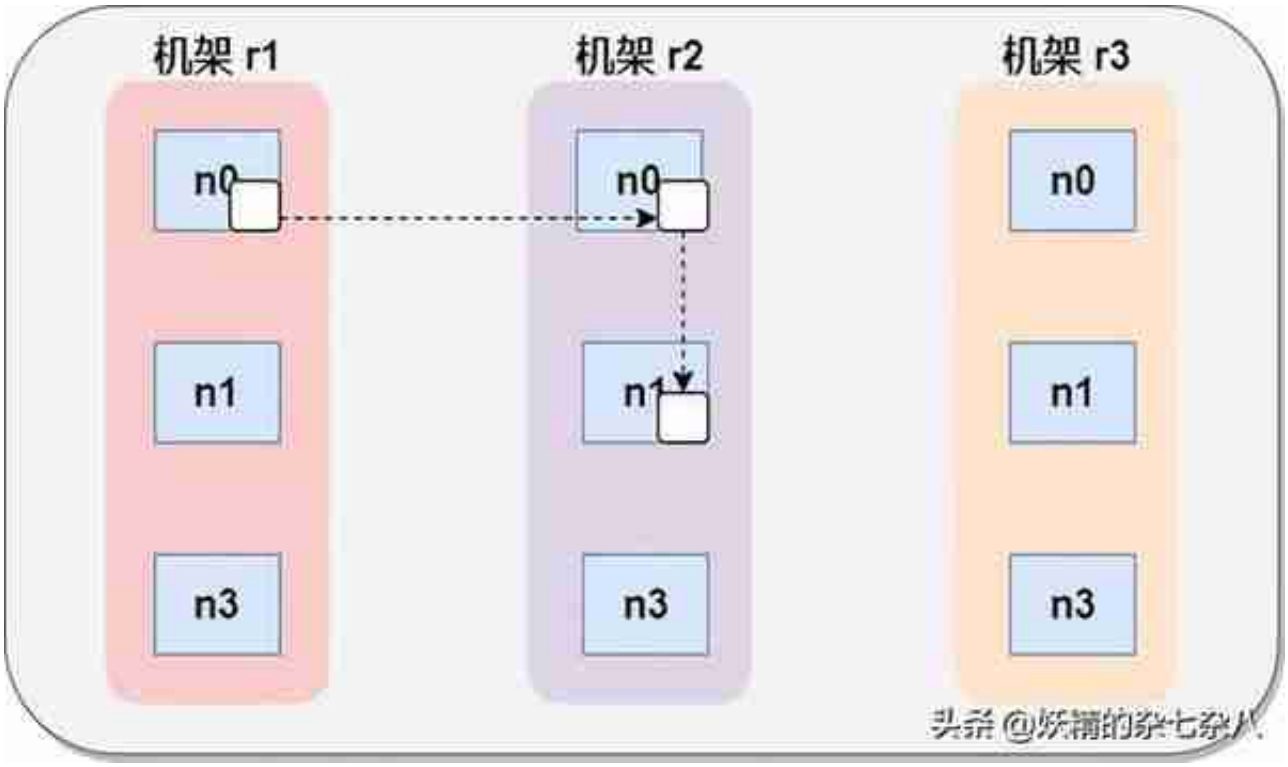
例如，假设有数据中心d1机架r1中的节点n1。该节点可以表示为/d1/r1/n1。利用这种标记，这里给出四种距离描述。

Distance(/d1/r1/n1, /d1/r1/n1)=0（同一节点上的进程）

Distance(/d1/r1/n1, /d1/r1/n2)=2（同一机架上的不同节点）

Distance(/d1/r1/n1, /d1/r3/n2)=4（同一数据中心不同机架上的节点）

Distance(/d1/r1/n1, /d2/r4/n2)=6（不同数据中心的节点）

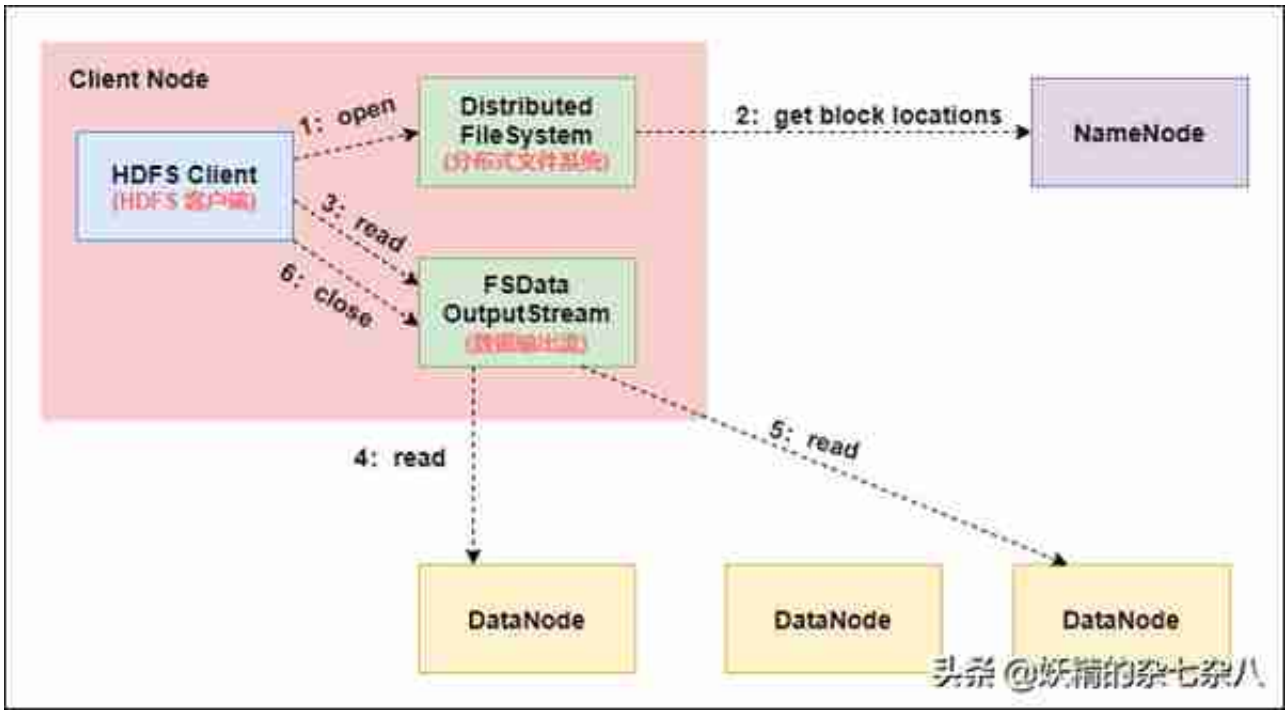


2. Hadoop2.7.2 副本节点选择

第一个副本在client所处的节点上。如果客户端在集群外，随机选一个。

第二个副本和第一个副本位于相同机架，随机节点。

第三个副本位于不同机架，随机节点。

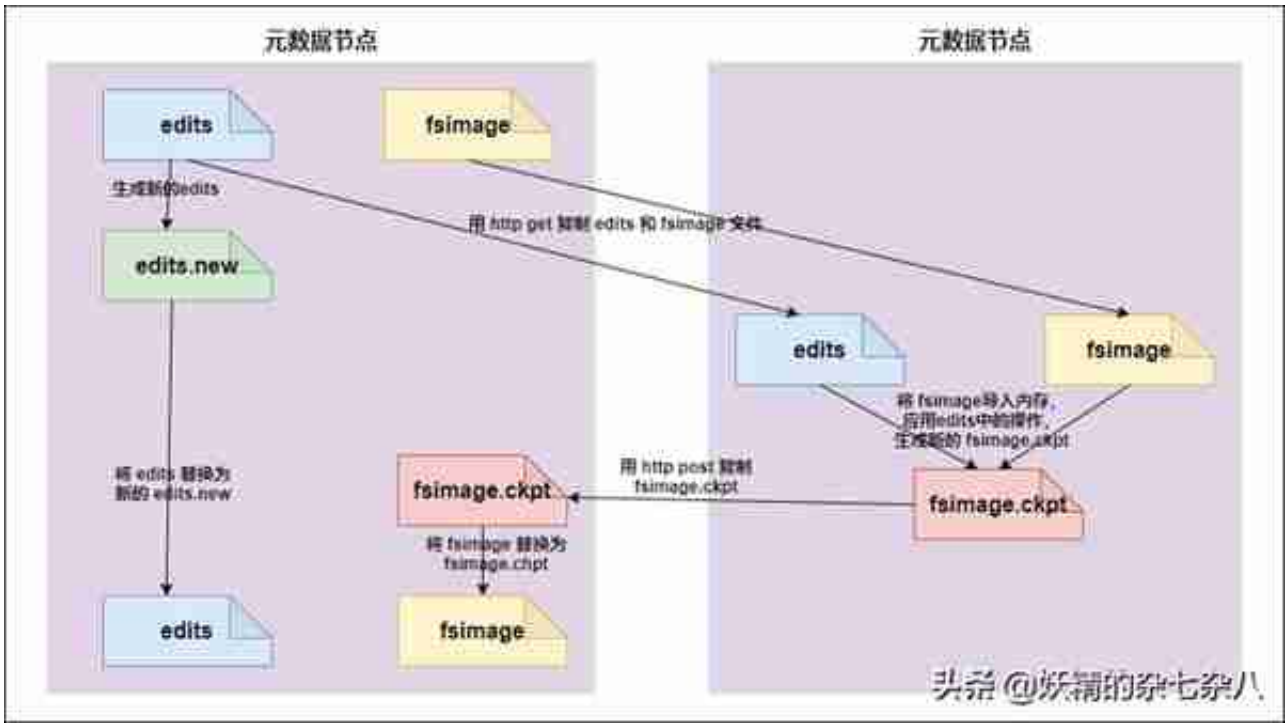


1. Client向NameNode发起RPC请求，来确定请求文件block所在的位置；
2. NameNode会视情况返回文件的部分或者全部block列表，对于每个block，NameNode 都会返回含有该 block 副本的 DataNode 地址；
   这些返回的 DN 地址，会按照集群拓扑结构得出 DataNode 与客户端的距离，然后进行排序，排序两个规则：网络拓扑结构中距离 Client 近的排靠前；心跳机制中超时汇报的 DN 状态为 STALE，这样的排靠后；
3. Client 选取排序靠前的 DataNode 来读取 block，如果客户端本身就是DataNode，那么将从本地直接获取数据(短路读取特性)；
4. 底层上本质是建立 Socket Stream（FSDataInputStream），重复的调用父类 DataInputStream 的 read 方法，直到这个块上的数据读取完毕；
5. 当读完列表的 block 后，若文件读取还没有结束，客户端会继续向NameNode 获取下一批的 block 列表；

6. 读取完一个 block 都会进行 checksum 验证，如果读取 DataNode 时出现错误，客户端会通知 NameNode，然后再从下一个拥有该 block 副本的DataNode 继续读。
7. read 方法是并行的读取 block 信息，不是一块一块的读取；NameNode 只是返回Client请求包含块的DataNode地址，并不是返回请求块的数据；
8. 最终读取来所有的 block 会合并成一个完整的最终文件。

从 HDFS 文件读写过程中，可以看出，HDFS 文件写入时是串行写入的，数据包先发送给节点A，然后节点A发送给B，B在给C；而HDFS文件读取是并行的， 客户端 Client 直接并行读取block所在的节点。

9. NameNode 工作机制以及元数据管理（重要）



完成合并的是 secondarynamenode，会请求namenode停止使用edits，暂时将新写操作放入一个新的文件中（edits.new)。

secondarynamenode从namenode中通过http get获得edits，因为要和fsimage合并，所以也是通过http get 的方式把fsimage加载到内存，然后逐一执行具体对文件系统的操作，与fsimage合并，生成新的fsimage，然后把fsimage发送给namenode，通过http post的方式。

namenode从secondarynamenode获得了fsimage后会把原有的fsimage替换为新的fsimage，把edits.new变成edits。同时会更新fsimage。

hadoop进入安全模式时需要管理员使用dfsadmin的save namespace来创建新的检查点。

secondarynamenode在合并edits和fsimage时需要消耗的内存和namenode差不多，所以一般把namenode和secondarynamenode放在不同的机器上。

fsimage与edits的合并时机取决于两个参数，第一个参数是默认1小时fsimage与edits合并一次。

- 第一个参数：时间达到一个小时fsimage与edits就会进行合并

```
dfs.namenode.checkpoint.period      3600
```

- 第二个参数：hdfs操作达到1000000次也会进行合并

```
dfs.namenode.checkpoint.txns        1000000
```

- 第三个参数：每隔多长时间检查一次hdfs的操作次数

```
dfs.namenode.checkpoint.check.period    60
```

9.6 namenode元数据信息多目录配置

为了保证元数据的安全性，我们一般都是先确定好我们的磁盘挂载目录，将元数据的磁盘做RAID1

namenode的本地目录可以配置成多个，且每个目录存放内容相同，增加了可靠性。

- 具体配置方案:hdfs-site.xml

```
 <property>          <name>dfs.namenode.name.dir</name>
   <value>file:///export/servers/hadoop-2.6.0-cdh5.14.0/hado
opDatas/namenodeDatas</value>    </property>
```

9.7 namenode故障恢复

在我们的secondaryNamenode对namenode当中的fsimage和edits进行合并的时候，每次都会先将namenode的fsimage与edits文件拷贝一份过来，所以fsimage与edits文件在secondarNamendoe当中也会保存有一份，如果namenode的fsimage与edits文件损坏，那么我们可以将secondaryNamenode当中的fsimage与edits拷贝过去给namenode继续使用，只不过有可能会丢失一部分数据。这里涉及到几个配置选项

- namenode保存fsimage的配置路径

```
<!--  namenode??????????????????SSD????????????????????? --> <property>  <name>dfs.namenode.name.dir</name>  <value>file:///export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/namenodeDatas</value> </property>
```

- namenode保存edits文件的配置路径

```
<property>  <name>dfs.namenode.edits.dir</name>  <value>file:///export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/nn/edits</value></property>
```

- secondaryNamenode保存fsimage文件的配置路径

```
<property>  <name>dfs.namenode.checkpoint.dir</name>  <value>file:///export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/snn/name</value></property>
```

- secondaryNamenode保存edits文件的配置路径

```
<property>  <name>dfs.namenode.checkpoint.edits.dir</name>  <value>file:///export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/nn/snn/edits</value></property>
```

接下来我们来模拟namenode的故障恢复功能

1. 杀死namenode进程: 使用jps查看namenode的进程号，kill -9 直接杀死。
2. 删除namenode的fsimage文件和edits文件。

根据上述配置, 找到namenode放置fsimage和edits路径. 直接全部rm -rf 删除。

3. 拷贝secondaryNamenode的fsimage与edits文件到namenode的fsimage与edits文件夹下面去。

   根据上述配置, 找到secondaryNamenode的fsimage和edits路径, 将内容使用cp -r 全部复制到namenode对应的目录下即可。

4. 重新启动namenode, 观察数据是否存在。

10. datanode工作机制以及数据存储

- datanode工作机制

1. 一个数据块在datanode上以文件形式存储在磁盘上，包括两个文件，一个是数据本身，一个是元数据包括数据块的长度，块数据的校验和，以及时间戳。
2. DataNode启动后向namenode注册，通过后，周期性（1小时）的向namenode上报所有的块信息。(dfs.blockreport.intervalMsec)。
3. 心跳是每3秒一次，心跳返回结果带有namenode给该datanode的命令如复制块数据到另一台机器，或删除某个数据块。如果超过10分钟没有收到某个datanode的心跳，则认为该节点不可用。
4. 集群运行中可以安全加入和退出一些机器。

- 数据完整性

1. 当DataNode读取block的时候，它会计算checksum。
2. 如果计算后的checksum，与block创建时值不一样，说明block已经损坏。
3. client读取其他DataNode上的block。
4. datanode在其文件创建后周期验证checksum。

- 掉线时限参数设置

datanode进程死亡或者网络故障造成datanode无法与namenode通信，namenode不会立即把该节点判定为死亡，要经过一段时间，这段时间暂称作超时时长。HDFS默认的超时时长为10分钟+30秒。如果定义超时时间为timeout，则超时时长

的计算公式为：

timeout = 2 * dfs.namenode.heartbeat.recheck-interval + 10 * dfs.heartbeat.interval。

而默认的dfs.namenode.heartbeat.recheck-interval
大小为5分钟，dfs.heartbeat.interval默认为3秒。

需要注意的是hdfs-site.xml
配置文件中的heartbeat.re
check.interval的单位为毫秒，dfs.heartbeat.interval的单位为秒。

```
<property>    <name>dfs.namenode.heartbeat.recheck-interval</name>    <value>300000</value></property><property>    <name>dfs.heartbeat.interval </name>    <value>3</value></property>
```

- DataNode的目录结构
  和namenode不同的是，datanode的存储目录是初始阶段自动创建的，不需要额外格式化。

在/opt/hadoop-2.6.0-cdh5.14.0/hadoopDatas/datanodeDatas/current这个目录下查看版本号

```
    cat VERSION          #Thu Mar 14 07:58:46 CST 2019    storageID=DS-47bcc6d5-c9b7-4c88-9cc8-6154b8a2bf39    clusterID=CID-dac2e9fa-65d2-4963-a7b5-bb4d0280d3f4    cTime=0    datanodeUuid=c44514a0-9ed6-4642-b3a8-5af79f03d7a4    storageType=DATA_NODE    layoutVersion=-56
```

具体解释:

storageID：存储id号。

clusterID集群id，全局唯一。

cTime属性标记了datanode存储系统的创建时间，对于刚刚格式化的存储系统，这个属性为0；但是在文件系统升级之后，该值会更新到新的时间戳。

datanodeUuid：datanode的唯一识别码。

storageType：存储类型。

layoutVersion是一个负整数。通常只有HDFS增加新特性时才会更新这个版本号。

- datanode多目录配置

datanode也可以配置成多个目录，每个目录存储的数据不一样。即：数据不是副本。具体配置如下：
- 只需要在value中使用逗号分隔出多个存储目录即可

```
  cd /opt/hadoop-2.6.0-cdh5.14.0/etc/hadoop  <!--  ??dataNod
e??????????????????????????????????????????  -->          <pr
operty>                    <name>dfs.datanode.data.dir</name>
                  <value>file:///opt/hadoop-2.6.0-cdh5.14.0/h
adoopDatas/datanodeDatas</value>          </property>
```

10.1 服役新数据节点

需求说明:

随着公司业务的增长，数据量越来越大，原有的数据节点的容量已经不能满足存储数据的需求，需要在原有集群基础上动态添加新的数据节点。

10.1.1 环境准备

1. 复制一台新的虚拟机出来

   将我们纯净的虚拟机复制一台出来，作为我们新的节点

2. 修改mac地址以及IP地址

```
??mac???? vim /etc/udev/rules.d/70-persistent-net.rules??ip?
??? vim /etc/sysconfig/network-scripts/ifcfg-eth0
```

3. 关闭防火墙，关闭selinux

????? service iptables stop??selinux vim /etc/selinux/config

4. 更改主机名

????????node04??????node04.hadoop.comvim /etc/sysconfig/network

5. 四台机器更改主机名与IP地址映射

????????hosts??vim /etc/hosts192.168.52.100 node01.hadoop.com node01192.168.52.110 node02.hadoop.com node02192.168.52.120 node03.hadoop.com node03192.168.52.130 node04.hadoop.com node04

6. node04服务器关机重启

node04?????????? reboot -h now

7. node04安装jdk

node04?????? mkdir -p /export/softwares/ mkdir -p /export/servers/

然后解压jdk安装包，配置环境变量

8. 解压hadoop安装包

?node04???????hadoop????/export/servers , node01???????hadoop??????node04??? cd /export/softwares/ scp hadoop-2.6.0-cdh5.14.0-????????.tar.gz node04:$PWDnode04????? tar -zxf hadoop-2.6.0-cdh5.14.0-????????.tar.gz -C /export/servers/

9. 将node01关于hadoop的配置文件全部拷贝到node04

node01????????hadoop??????????node04????? cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop/ scp ./* node04:$PWD

10.1.2 服役新节点具体步骤

1. 创建dfs.hosts文件

```
?node01???namenode??????/export/servers/hadoop-2.6.0-cdh5.14
.0/etc/hadoop?????dfs.hosts??[root@node01 hadoop]# cd /expor
t/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop[root@node01 hado
op]# touch dfs.hosts[root@node01 hadoop]# vim dfs.hosts?????
???????????????node01node02node03node04
```

2. node01编辑hdfs-site.xml添加以下配置

在namenode的hdfs-site.xml配置文件中增加dfs.hosts属性

```
node01?????? :cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/
hadoopvim hdfs-site.xml# ?????? <property>         <name>dfs
.hosts</name>         <value>/export/servers/hadoop-2.6.0-cd
h5.14.0/etc/hadoop/dfs.hosts</value>    </property>    <!--?
??????: ????????, ??????-->    <property>  <name>dfs.hosts</
name>  <value>/export/servers/hadoop-2.6.0-cdh5.14.0/etc/had
oop/accept_host</value> </property>  <property>  <name>dfs.h
osts.exclude</name>  <value>/export/servers/hadoop-2.6.0-cdh
5.14.0/etc/hadoop/deny_host</value> </property>
```

3. 刷新namenode

- node01执行以下命令刷新namenode

```
[root@node01 hadoop]# hdfs dfsadmin -refreshNodesRefresh nod
es successful
```

4. 更新resourceManager节点

- node01执行以下命令刷新resourceManager

```
[root@node01 hadoop]# yarn rmadmin -refreshNodes19/03/16 11:
19:47 INFO client.RMProxy: Connecting to ResourceManager at
node01/192.168.52.100:8033
```

5. namenode的slaves文件增加新服务节点主机名称

node01编辑slaves文件，并添加新增节点的主机，更改完后，slaves文件不需要分发到其他机器上面去

```
node01????????slaves?? : cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop vim slaves ??????:  node01node02node03node04
```

6. 单独启动新增节点

```
node04???????????datanode?nodemanager :  cd /export/servers/hadoop-2.6.0-cdh5.14.0/ sbin/hadoop-daemon.sh start datanode sbin/yarn-daemon.sh start nodemanager
```

7. 使用负载均衡命令，让数据均匀负载所有机器

```
node01?????? :  cd /export/servers/hadoop-2.6.0-cdh5.14.0/ sbin/start-balancer.sh
```

10.2 退役旧数据

1. 创建dfs.hosts.exclude配置文件

在namenod所在服务器的/export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop目录下创建dfs.hosts.exclude文件，并添加需要退役的主机名称

```
node01?????? :  cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop touch dfs.hosts.exclude vim dfs.hosts.exclude?????? : node04.hadoop.com??????????????????????ip????????node04
```

2. 编辑namenode所在机器的hdfs-site.xml

编辑namenode所在的机器的hdfs-site.xml配置文件，添加以下配置

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoopvim hdfs-site.xml#??????: <property>         <name>dfs.hosts.exclude</name>         <value>/export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop/dfs.hosts.exclude</value>   </property>
```

3. 刷新namenode，刷新resourceManager

```
?namenode?????????????namenode???resourceManager : hdfs dfs admin -refreshNodesyarn rmadmin -refreshNodes
```

4. 节点退役完成，停止该节点进程

等待退役节点状态为decommissioned（所有块已经复制完成），停止该节点及节点资源管理器。注意：如果副本数是3，服役的节点小于等于3，是不能退役成功的，需要修改副本数后才能退役。

```
node04????????????? :  cd /export/servers/hadoop-2.6.0-cdh5.14.0 sbin/hadoop-daemon.sh stop datanode sbin/yarn-daemon.sh stop nodemanager
```

5. 从include文件中删除退役节点

```
namenode???????node01??????????? : cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop vim dfs.hosts ??????: ???node04 node01node02node03
```

6. node01执行一下命令刷新namenode，刷新resourceManager

```
hdfs dfsadmin -refreshNodesyarn rmadmin -refreshNodes
```

7. 从namenode的slave文件中删除退役节点

```
namenode???????node01???????slaves????????? :  cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop vim slaves??????: ??? node04 node01node02node03
```

8. 如果数据负载不均衡，执行以下命令进行均衡负载

```
node01??????????? cd /export/servers/hadoop-2.6.0-cdh5.14.0/ sbin/start-balancer.sh
```

11. block块手动拼接成为完整数据

所有的数据都是以一个个的block块存储的，只要我们能够将文件的所有block块全

部找出来，拼接到一起，又会成为一个完整的文件，接下来我们就来通过命令将文件进行拼接:

1. 上传一个大于128M的文件到hdfs上面去

我们选择一个大于128M的文件上传到hdfs上面去，只有一个大于128M的文件才会有多个block块。

这里我们选择将我们的jdk安装包上传到hdfs上面去。

node01执行以下命令上传jdk安装包

```
cd /export/softwares/hdfs dfs -put jdk-8u141-linux-
x64.tar.gz  /
```

2. web浏览器界面查看jdk的两个block块id

这里我们看到两个block块id分别为

1073742699和1073742700

那么我们就可以通过blockid将我们两个block块进行手动拼接了。

3. 根据我们的配置文件找到block块所在的路径

```
????hdfs-site.xml?????datanode?????<!--  ??dataNode????????
??????????????????????????????????  -->        <property>
            <name>dfs.datanode.data.dir</name>
  <value>file:///export/servers/hadoop-2.6.0-cdh5.14.0/hadoo
pDatas/datanodeDatas</value>        </property>
   ??????? : ?????? ?????value???cd /export/servers/hadoop-2
.6.0-cdh5.14.0/hadoopDatas/datanodeDatas/current/BP-55746692
6-192.168.52.100-1549868683602/current/finalized/subdir0/sub
dir3
```

4. 执行block块的拼接

```
??????block??????????????????????cat blk_1073742699 >> jdk8u1
41.tar.gzcat blk_1073742700 >> jdk8u141.tar.gz?????jdk?/expo
```

```
rt?????????mv  jdk8u141.tar.gz /export/cd /export/tar -zxf j
dk8u141.tar.gz???????????????????block???????
```

12. HDFS其他重要功能

1. 多个集群之间的数据拷贝

在我们实际工作当中，极有可能会遇到将测试集群的数据拷贝到生产环境集群，或者将生产环境集群的数据拷贝到测试集群，那么就需要我们在多个集群之间进行数据的远程拷贝，hadoop自带也有命令可以帮我们实现这个功能

1. 本地文件拷贝scp

```
cd /export/softwares/scp -r jdk-8u141-linux-
x64.tar.gz root@node02:/export/
```

2. 集群之间的数据拷贝distcp

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/bin/hadoop distcp
hdfs://node01:8020/jdk-8u141-linux-
x64.tar.gz  hdfs://cluster2:8020/
```

2. hadoop归档文件archive

每个文件均按块存储，每个块的元数据存储在namenode的内存中，因此hadoop存储小文件会非常低效。因为大量的小文件会耗尽namenode中的大部分内存。但注意，存储小文件所需要的磁盘容量和存储这些文件原始内容所需要的磁盘空间相比也不会增多。例如，一个1MB的文件以大小为128MB的块存储，使用的是1MB的磁盘空间，而不是128MB。

Hadoop存档文件或HAR文件，是一个更高效的文件存档工具，它将文件存入HDFS块，在减少namenode内存使用的同时，允许对文件进行透明的访问。具体说来，Hadoop存档文件可以用作MapReduce的输入。

创建归档文件

- 第一步：创建归档文件注意：归档文件一定要保证yarn集群启动

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0bin/hadoop archive
-archiveName myhar.har -p /user/root /user
```

- 第二步：查看归档文件内容

```
hdfs dfs -lsr /user/myhar.harhdfs dfs -lsr har:///user/myhar
.har
```

- 第三步：解压归档文件

```
hdfs dfs -mkdir -p /user/harhdfs dfs -cp har:///user/myhar.h
ar/* /user/har/
```

3. hdfs快照snapShot管理

快照顾名思义，就是相当于对我们的hdfs文件系统做一个备份，我们可以通过快照对我们指定的文件夹设置备份，但是添加快照之后，并不会立即复制所有文件，而是指向同一个文件。当写入发生时，才会产生新文件

1. 快照使用基本语法

```
1? ??????????? hdfs dfsadmin  -allowSnapshot  ?? 2??????????
???????????? hdfs dfsadmin  -disallowSnapshot  ??3???????????
snapshot hdfs dfs -createSnapshot  ??4?????????????snapshot
hdfs dfs  -createSanpshot ?? ??    5???????? hdfs dfs  -rena
meSnapshot  ?? ???  ???6????????????? hdfs lsSnapshottableD
ir  7????????????? hdfs snapshotDiff  ??1  ???28????snapsho
t hdfs dfs -deleteSnapshot <path> <snapshotName>
```

2. 快照操作实际案例

```
1?????????????    [root@node01 hadoop-2.6.0-cdh5.14.0]# hdfs
 dfsadmin -allowSnapshot /user    Allowing snaphot on /user
succeeded    [root@node01 hadoop-2.6.0-cdh5.14.0]# hdfs dfsa
dmin -disallowSnapshot /user    Disallowing snaphot on /user
 succeeded2?????????? ????????????????????    [root@node01
hadoop-2.6.0-cdh5.14.0]# hdfs dfsadmin -allowSnapshot /user
   Allowing snaphot on /user succeeded    [root@node01 hadoo
p-2.6.0-cdh5.14.0]# hdfs dfs -createSnapshot /user        Cr
```

```
eated snapshot /user/.snapshot/s20190317-210906.549 ??web???
???? http://node01:50070/explorer.html#/user/.snapshot/s2019
0317-210906.5493????????    [root@node01 hadoop-2.6.0-cdh5.
14.0]# hdfs dfs -createSnapshot /user mysnap1    Created sna
pshot /user/.snapshot/mysnap14?????? hdfs dfs -renameSnapsho
t /user mysnap1 mysnap2 5????????????????? hdfs lsSnapshottab
leDir6???????????    hdfs dfs -createSnapshot /user snap1
 hdfs dfs -createSnapshot /user snap2    hdfs snapshotDiff
snap1 snap27????? hdfs dfs -deleteSnapshot /user snap1
```

4. hdfs回收站

任何一个文件系统，基本上都会有垃圾桶机制，也就是删除的文件，不会直接彻底清掉，我们一把都是将文件放置到垃圾桶当中去，过一段时间之后，自动清空垃圾桶当中的文件，这样对于文件的安全删除比较有保证，避免我们一些误操作，导致误删除文件或者数据

1. 回收站配置两个参数

默认值fs.trash.interval=0，0表示禁用回收站，可以设置删除文件的存活时间。

默认值fs.trash.checkpoint.interval=0，检查回收站的间隔时间。

要求fs.trash.checkpoint.interval<=fs.trash.interval。

2. 启用回收站

修改所有服务器的core-site.xml配置文件

```
<!--  ??hdfs??????????????????????????? -->        <property
>                <name>fs.trash.interval</name>
   <value>10080</value>        </property>
```

3. 查看回收站

回收站在集群的 /user/root/.Trash/ 这个路径下

4. 通过javaAPI删除的数据，不会进入回收站，需要调用moveToTrash()才会进入回收站

```
//????????: ????    @Test    public void  deleteFile() throw
s Exception{        //1. ??FileSystem??        Configuration
 configuration = new Configuration();        FileSystem file
System = FileSystem.get(new URI("hdfs://node01:8020"), confi
guration, "root");        //2. ??????        // fileSystem.d
elete();  ???????????, ???????        Trash trash = new Tra
sh(fileSystem,configuration);        boolean flag = trash.is
Enabled(); // ???????????        System.out.println(flag);
       trash.moveToTrash(new Path("/quota"));        //3. ??
??        fileSystem.close();    }
```

5. 恢复回收站数据

hdfs dfs -mv trashFileDir hdfsdir

trashFileDir ：回收站的文件路径

hdfsdir ：将文件移动到hdfs的哪个路径下

6. 清空回收站

hdfs dfs -expunge