HDFS架构角色：

namenode：主角色，管理hdfs集群和datanode角色（领导）

datanode：从角色,负责数据的存储（员工）

secondarynamenode：辅助角色，协助namenode整理元数据（秘书）

如何区分本地和hdfs文件系统的路径

加上协议头：file://

hdfs：//node1:8020/

关于HDFS文件系统的操作命令，hadoop提供了两套命令

1。创建文件夹

hadoop fs -mkdir -p path

hdfs dfs -mkdir -p path

-p 沿着父文件夹开始创建

2.查看指定目录下的内容

hadoop fs -ls[-h][-R] path

-h 人性化显示size

-R递归查看指定目录及其子目录

3.上传文件到hdfs

-put [-f][-p] <localsrc><dst>

-f 覆盖目标文件

-p 保留访问和修改时间，所有权和权限

<localsrc>本地文件系统

<dst>目标文件系统

4.查看hdfs内容

-cat <src>

5.下载

-get [-f][-p] <src><localsrc>

<目标文件系统>

-f 覆盖文件

6.复制文件

-cp [-f] <src><dst>

-f 覆盖目标文件

7.追加数据到hdfs文件中

只能删除和追加，不能修改某一行

-appendToFile <localsrc><dst>

8.文件移动

-mv

9.数据删除

-rm –r [-skipTrash] path

-r 删除文件夹

HDFS超级用户，是启动了namenode的用户，不是root用户

hdfs分布式文件存储：把文件进行切分，分布在每一台服务器上，

解决取文件的问题：文件大小不一，不利于管理

设定统一的管理模块，block块，HDFS最小存储单位，每个默认256MB（可以修改）

如果丢失或损坏某个block块呢：通过多个副本（备份）解决

为什么不把备份放到同一个服务器：但凡一个服务器出问题，这份文件就损坏了

每个块有两个备份，分发到其他服务器中，安全性大大提高。

如何配置副本块的数量：

可以在hdfs-site.xml配置如下属性

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

方式2：上传文件时设置hadoop fs –D dfs.replication=2 –put test.txt /tmp/

对于已经存在的文件：hdfs dfs –setrep [-R] 2 path

-R表示子目录也生效

fsck检查文件副本数

hdfs fsck path [-files [-blocks [-locations]]]

-files 列出路径内的文件状态



-blocks列出每个文件块

-locations 列出备份位置

块大小可以通过参数:修改hdfs-site.xml

<property>

<name>dfs.blocksize</name>

<value>268435456</value>

<description>设置HDFS块大小，单位是b</description>

</property>

namenode是如何管理模块的：

namenode通过edits文件，是一个流水账文件，记录了hdfs的每一次操作，以及本次操作影响的文件其对应的block

当用户查看某文件内容，就需要在全部的edits中搜索，避免后期改名或删除，效率低。

需要合并edits文件，得到最后的结果，即可得到FSImage文件

总结：

1. 每次对hdfs操作，均被edits文件记录
2. edits达到上限大小，开启新的edits记录
3. 定期进行edits合并操作

如果没有fsimage文件，将全部edits合并为第一个fsimage

如果已存在，将以存在的fsimage和新的edits合并。

对于元数据的合并，是一个定时过程：

dfs.namenode.checkpoint.period,默认3600秒，即1小时

dfs.namenode.checkpoint.txns 默认100w次事务

检查是否达到条件，默认60秒检查一次

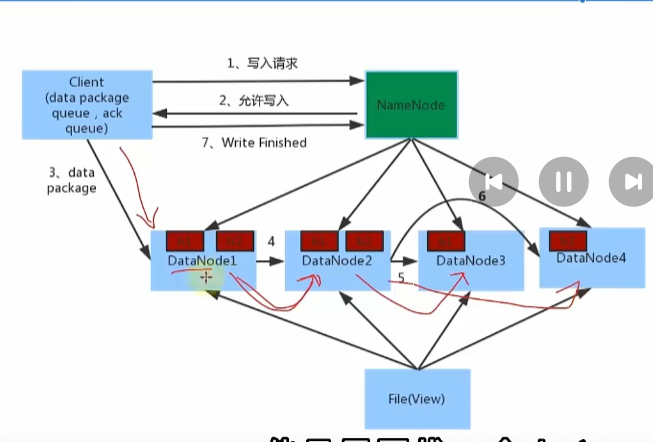
dfs.namenode.checkpoint.check.period 默认60秒

合并元数据的事情就是secondarynamenode干的，

secondarynamenode会通过http从namenode拉数据（edits和fsimage）再发回去namenode使用

数据写入流程：

1. 客户端向namenode发送请求
2. namenode审核权限，剩余空间后，满足条件写入，并告知客户端写入的datanode地址
3. 客户端向指定的datanode发送数据包
4. 被写入的数据datanode同时完成数据副本的复制工作，将其接受的数据分发给其他datanode，
5. datanode1复制给datanode2，然后基于datanode2复制给datanode3.。。。



1. 写入完成客户端通知namenode，namenode做元数据记录

注意：

namenode不负责数据写入，只负责审批和元数据记录

客户端直接向1台datanode写数据，这个datanode一般是客户端最近的那一个

数据块副本的复制工作，有datanode之间自行完成（构建一个pipline，按顺序复制分发）

数据读取文件：

客户端向namenode申请读取

namenode判断权限，允许读取，返回此文件的block列表

客户端拿到block列表后自行寻找datanode读取

注意：

数据同样不通过namenode提供

namenode提供的block列表，会基于网络距离计算尽量离客户端最近的datanode上的block让其读取