1.问题

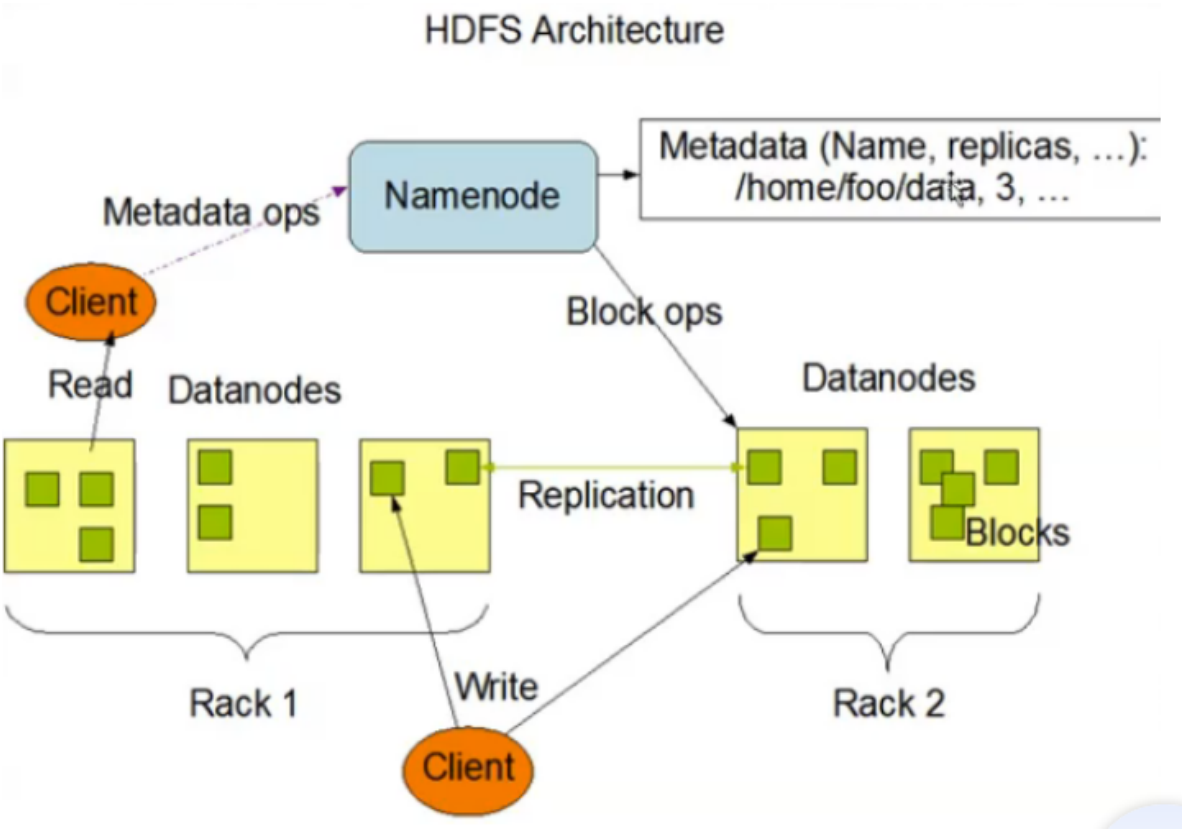
分布式文件系统那么多，为什么hadoop项目中还要开发一个分布式文件系统呢?

2.存储模型

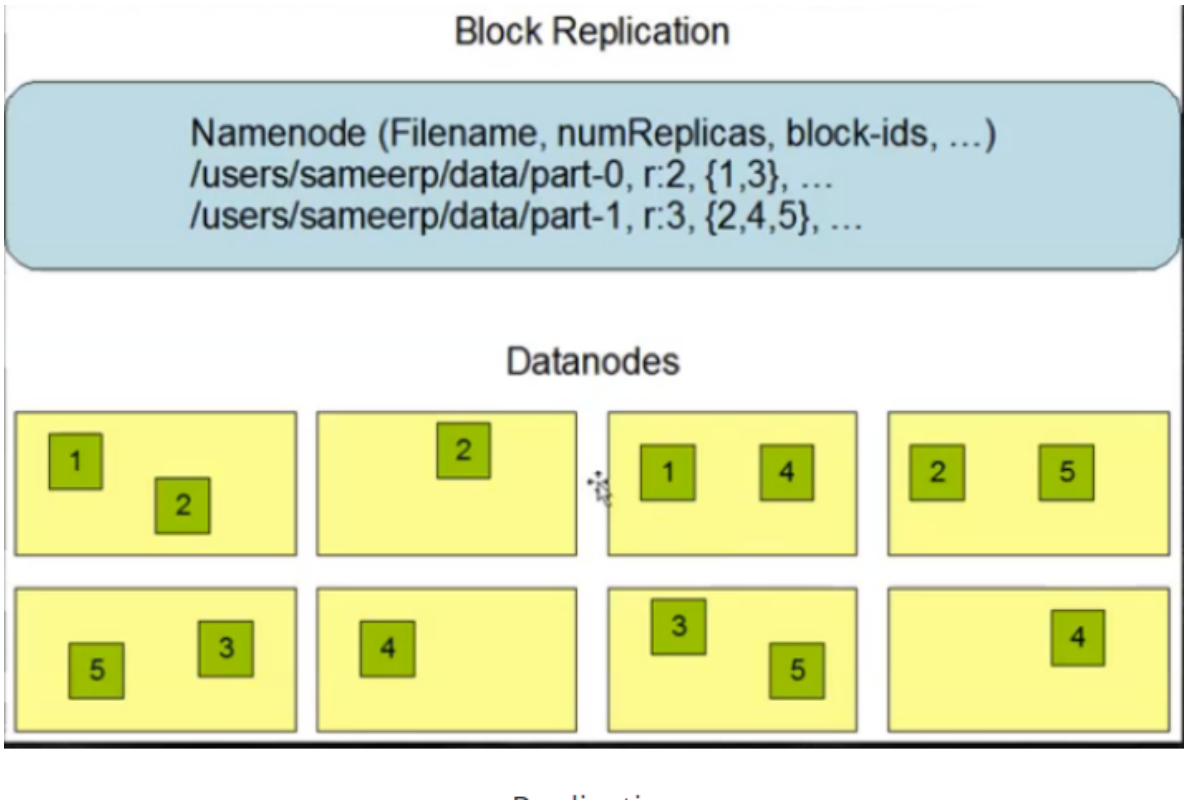
- 文件线性字节切割成块（Block），具有offset，id
- 文件与文件的block大小可以不一样
- 一个文件除了最后一个block，其他block大小一致
- block大小依据硬件I/O进行调整
- block被分散存放在集群节点中，具有location
- block具有副本(replication)，没有主从概念，副本不能出现在同一个节点
- 副本是满足可靠性和性能的关键
- 文件上传可以指定block大小和副本数，上传后只能修改副本数
- 一次写入多次读取，不支持修改
- 支持追加数据

3.架构设计

- HDFS是一个主从(Master/Slaves)架构
- 由一个NameNode和一些DataNode组成
- 面向文件包含：文件数据（data）和文件元数据（metadata）
- NameNode负责存储和管理文件元数据，并且维护了一个层次性文件目录树
- DataNode负责存储文件数据（block块），并提供block读写
- DataNode与NameNode维持心跳，并汇报自己持有的block信息
- Client和NameNode交互文件元数据和DataNode交互文件block数据



HDFS Architecture



Replication

4.角色功能

NameNode

- 完全基于内存存储元数据、目录结构、文件block映射

- 需要持久化方案保证数据可靠性
- 提供副本放置策略

DataNode

- 基于本地磁盘存储block（文件形式）
- 并保存block校验和数据保证block的可靠性
- 与NameNode保持心跳，汇报block列表状态

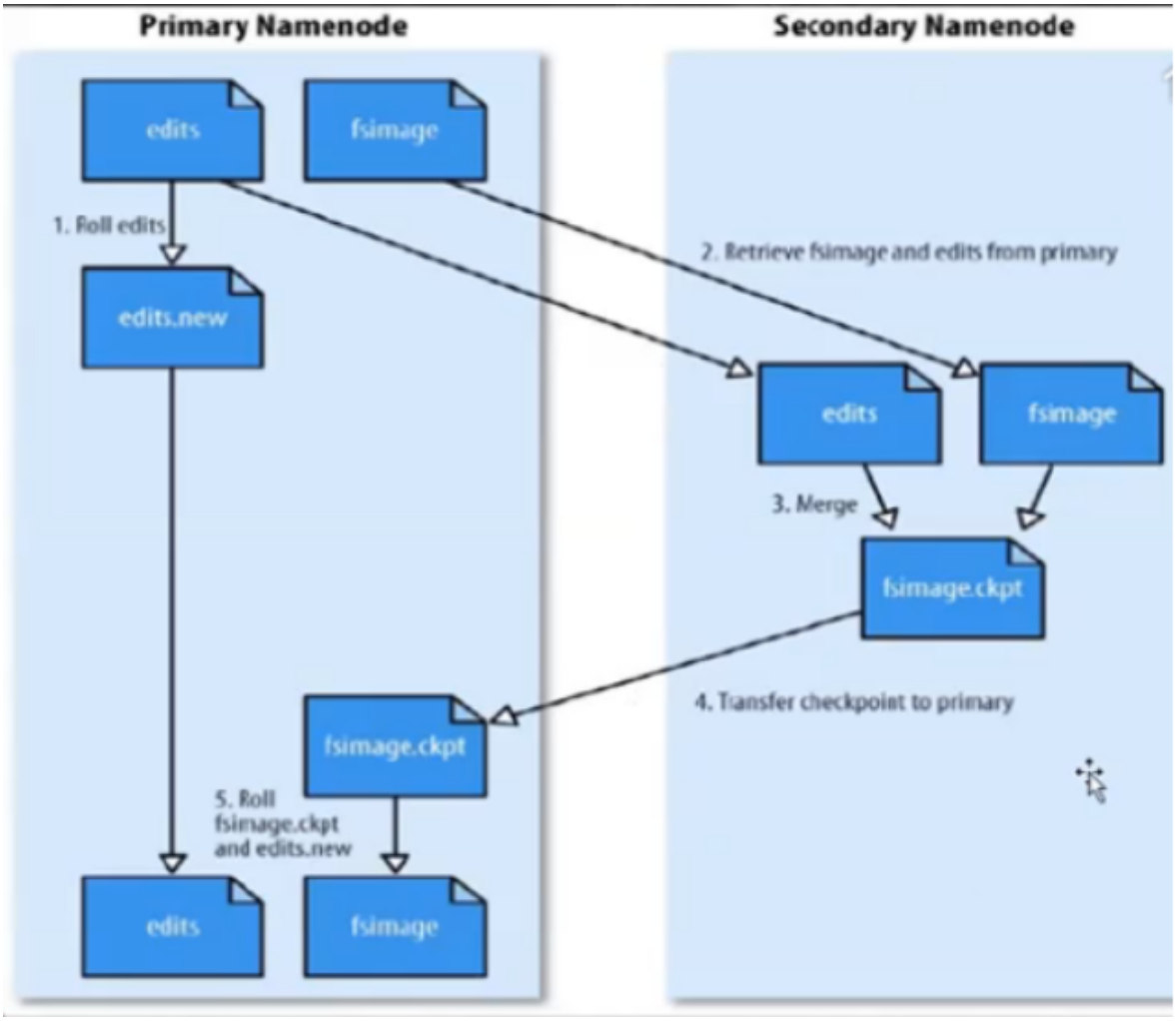
5.安全模式

- HDFS搭建时会格式化，格式化会产生一个空的FsImage
- 当NameNode启动，会加载Editlog和FsImage
- 将所有Editlog中事务作用在内存中的FsImage上
- 这个新版本的FsImage从内存保存在磁盘上
- 删除旧的Editlog，旧的Editlog事务已经作用在FsImage上了
- NameNode启动时候会进入一个安全模式的特殊状态
- 处于安全模式的NameNode是不会进行数据块复制的
- NameNode从所有的DataNode接收心跳信号和块状态报告
- 每当NameNode检测确认某个数据块副本数目达到这个最小值，那么该数据库就会被认为是副本安全（safely replicated）的
- 在一定百分比数据块被NameNode确认是安全后，加上30s的等待时间，NameNode会退出安全模式
- 接下来它会确定还有哪些数据库副本没有达到指定数目，并将这些数据库复制到其他DataNode上

6.HDFS中的SNN

Log System

- EditsLog FsImage
- 可以由某一点开始溢写全量，其余增量
- 思考：全量和增量的好处
- 在非Ha模式下，存在SNN，SNN一般是独立节点，周期完成对NN的EditLog想FsImage合并，减少EditLog大小，较少NN启动时间
- 根据配置文件设置时间间隔fs.checkpoint.period 默认3600秒
- 根据配置文件设置edits log大小 fs.checkpoint.size 规定edits文件的最大值默认64MB



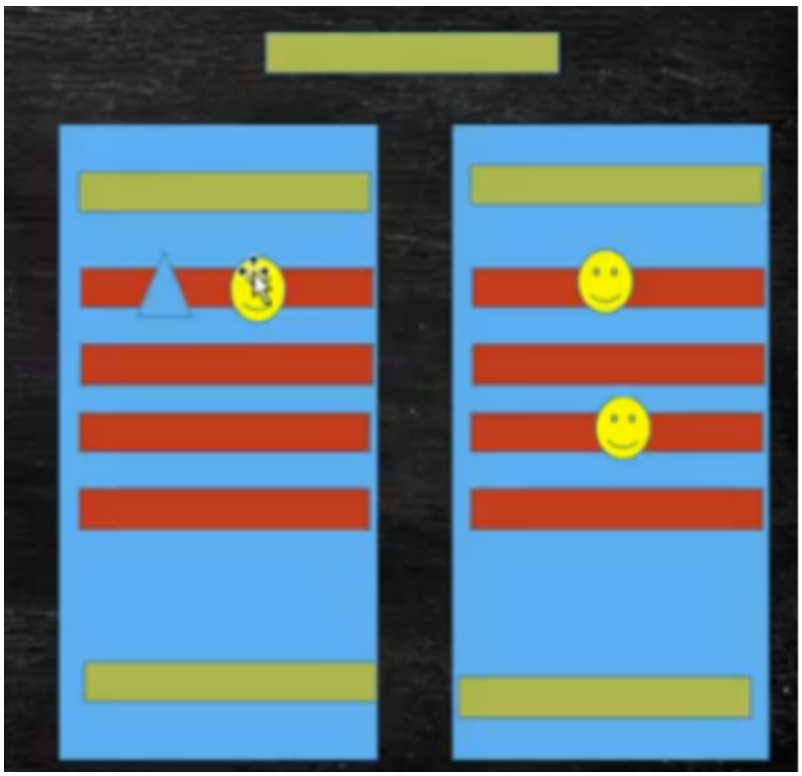
日志流程

7.Block副本放置策略

- 第一个副本：放置在上传文件的DataNode

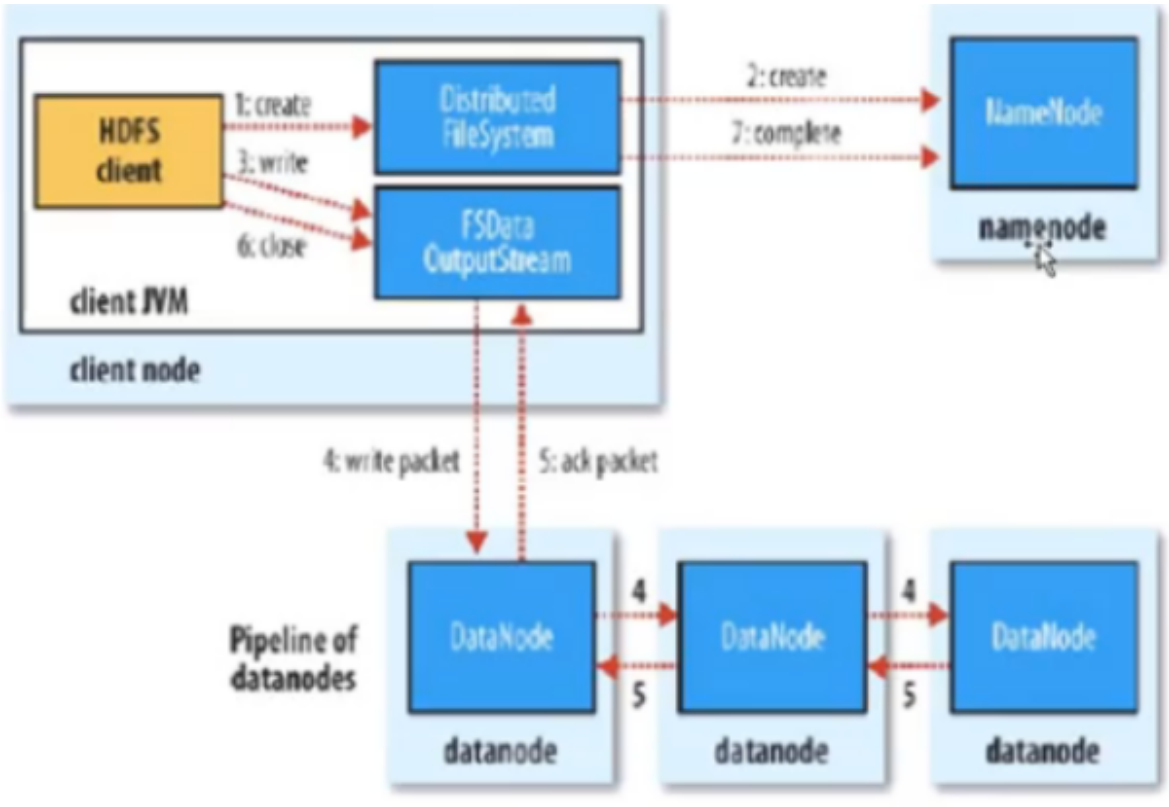
如果是集群外提交，则随机挑选一台磁盘不太满，CPU不太忙的节点

- 第二个副本：放置在与第一个副本不同的机架节点上
- 第三个副本：与第二个副本相同机架的节点
- 更多副本：随机节点



副本放置策略

8.读写流程



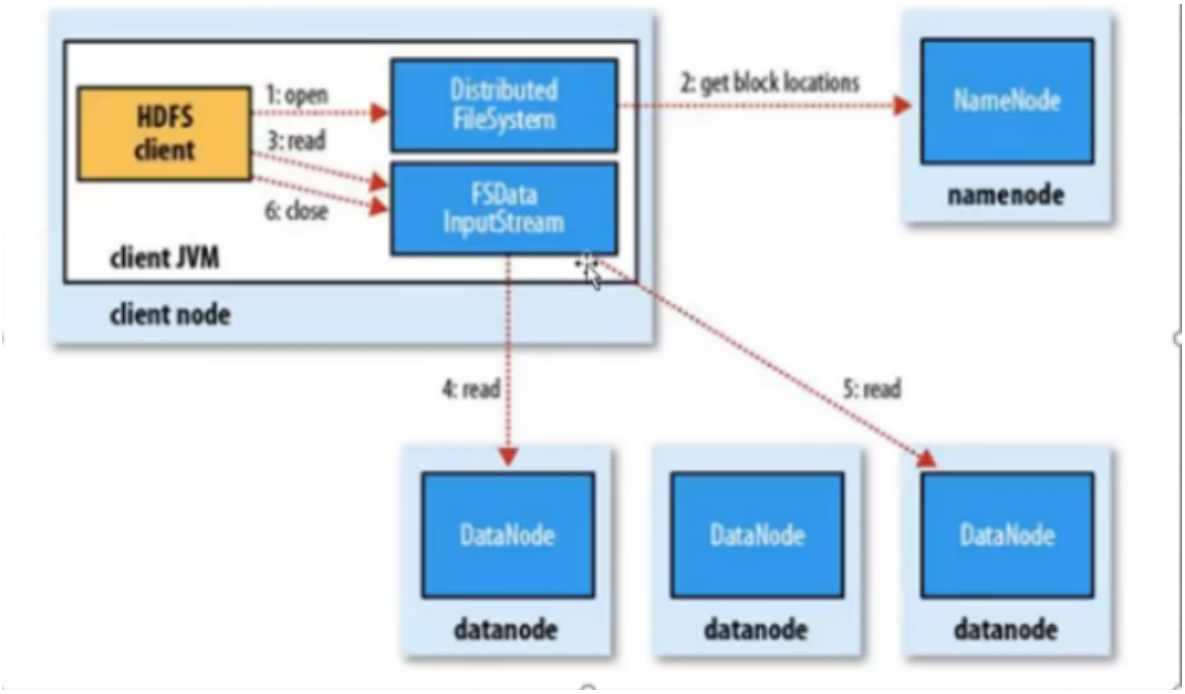
write flow

- Client和NN连接创建文件元数据
- NN判定元数据是否有效
- NN触发副本放置策略，返回一个有序的DN列表

- Client和DN简历Pipeline连接
- Client将块切分成packet(64KB)，并使用chunk(512B) + chunksum(4B)填充
- Client将packet放入dataqueue中，并向第一个DN发送
- 第一个DN收到packet后本地保存并发送给第二个DN
- 第二个DN收到packet后本地保存并发送给第三个DN
- 这个过程，上游节点同时发送下一个packet

生活中类比流水线，进行，结论：流式也是变种并行计算

- HDFS使用这种传输方式、副本数对于client是透明的
- 当block传输完成后，DN各自向NN汇报，同时client继续传输下一个block
- 所以，client传输和block汇报也是并行的



读流程

- 为了整体带宽和读演示，HDFS会尽量读取程序离最近副本
- 如果读取程序的同一个机架上有一个副本，直接读取该副本
- 如果一个HDFS集群跨越多个数据中心，那么客户端也将首先本地数据中心的副本

download：

client与NN交互，获取fileBlockLocation -> NN会按照距离策略排序返回 -> Client尝试下载block并且返回校验数据的完整性

- HDFS支持client给出文件的offset自定义连接block的DN，自定义获取数据
- 这个是支持计算层分治、并行计算的核心

9.伪分布式模式的搭建

关于伪分布式的配置全程

host	NN	SNN	DN
node01	*	*	*

伪分布式节点分布

1.安装VMWare WorkStation，直接下一步，输入激活码即可安装

2.安装Linux(需要100GB)

引导分区Boot200MB

交换分区Swap2048MB

其余分配到/

3.配置网络服务

```
vim /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=static
IPADDR=192.168.118.11
NETMASK=255.255.255.0
GATEWAY=192.168.118.2
DNS1=114.114.114.114
DNS2=233.5.5.5
```

注意点：

1.关于IPADDR的前三个网关，要与虚拟网络编辑器的VMnet8的子网IP的前三个网关一样

2.关于GATEWAY要与NAT下的GATEWAY一样，详情如下

虚拟网络编辑器：在VMWare编辑下打开

点击NAT设置，查看GATEWAY

4.修改主机名称

```
vi /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=node01
```

5.设置Host（关于Host，是指IP和主机名的映射关系）

```
vi /etc/hosts
192.168.150.11 node01
192.168.150.12 node02
```

6.关闭防火墙，开机不启动防火墙

```
service iptables stop
chkconfig iptables off
```

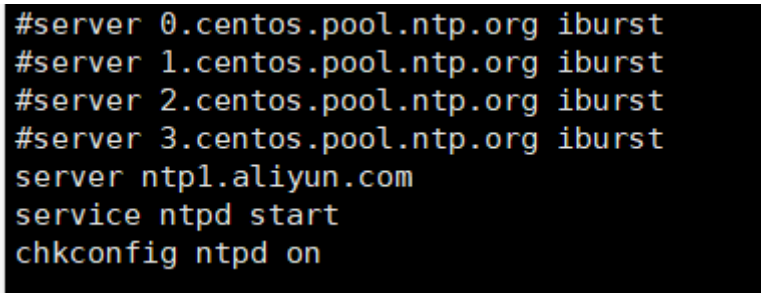
7.关闭selinux（selinux是Linux下一种安全模式，打开可能会连不上XShell）

```
vi /etc/selinux/config
SELINUX=disabled
```

8.时间同步

使用yum安装ntp，并把原有的server注释，替换成

```
server ntp1.aliyun.com
service ntpd start
chkconfig ntpd on
```



9.安装jdk，使用xftp上传rpm文件

```
jdk-8u181-linux-x64.rpm
```

修改JAVA_HOME

```
vi /etc/profile
export JAVA_HOME=/usr/java/default
export PATH=$PATH:$JAVA_HOME/bin
source /etc/profile
```

10.安装ssh免密

（1）检验ssh是否可以登录

```
ssh localhost
```

需要输入密码，则不免密

（2）设置免密

ssh-keygen -t dsa表示使用dsa算法加密
-p ''表示密码为空
-f ~/ .ssh/id_dsa 将公钥放在/home/.ssh/id_dsa下

```
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

不能自己私自创建目录，关于ssh的目录权限，必须为755或者700，不能是777，否则不能使用免密

11.安装Hadoop

```
mkdir /opt/bigdata
tar xf hadoop-2.6.5.tar.gz
mv hadoop-2.6.5 /opt/bigdata/
pwd
/opt/bigdata/hadoop-2.6.5
```

设置Hadoop的环境变量

```
vi /etc/profile
export JAVA_HOME=/usr/java/default
export HADOOP_HOME=/opt/bigdata/hadoop-2.6.5
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
source /etc/profile
```

12.修改hadoop-env.sh，此文件为hadoop启动脚本，将JAVA_HOME改为具体的环境变量

```
cd $HADOOP_HOME/etc/hadoop
vi hadoop-env.sh
export JAVA_HOME=/usr/java/default
```

13.给出NN角色在哪里启动vi core-site.xml

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://node01:9000</value>
</property>
```

14.配置一个hdfs副本

```
<!-- 副本数量为1 -->
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property><!-- NameNode的路径-->
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/var/bigdata/hadoop/local/dfs/name</value>
</property><!-- DataNode的路径-->
<property>
<name>dfs.datanode.data.dir</name>
    <value>/var/bigdata/hadoop/local/dfs/data</value>
    </property><!-- SecondaryNameNode在哪个端口启动-->
<property>
<name>dfs.namenode.secondary.http-address</name>
    <value>node01:50090</value>
    </property><!-- SecondaryNameNode的路径-->
<property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>/var/bigdata/hadoop/local/dfs/secondary</value>
</property>
```

15.配置Slave

```
vi slaves
node01
```

16.初始化&启动

```
hdfs namenode -format
```

创建目录，并且初始化一个空的fsimage

VERSION CID

```
start-dfs.sh
```

17.修改windows： C:\Windows\System32\drivers\etc\hosts（注意这边IP要与端口一样）

```
192.168.150.11 node01
192.168.150.12 node02
192.168.150.13 node03
192.168.150.14 node04
```

18.简单创建目录

```
hdfs dfs -mkdir /bigdata
hdfs dfs -mkdir -p /user/root
```

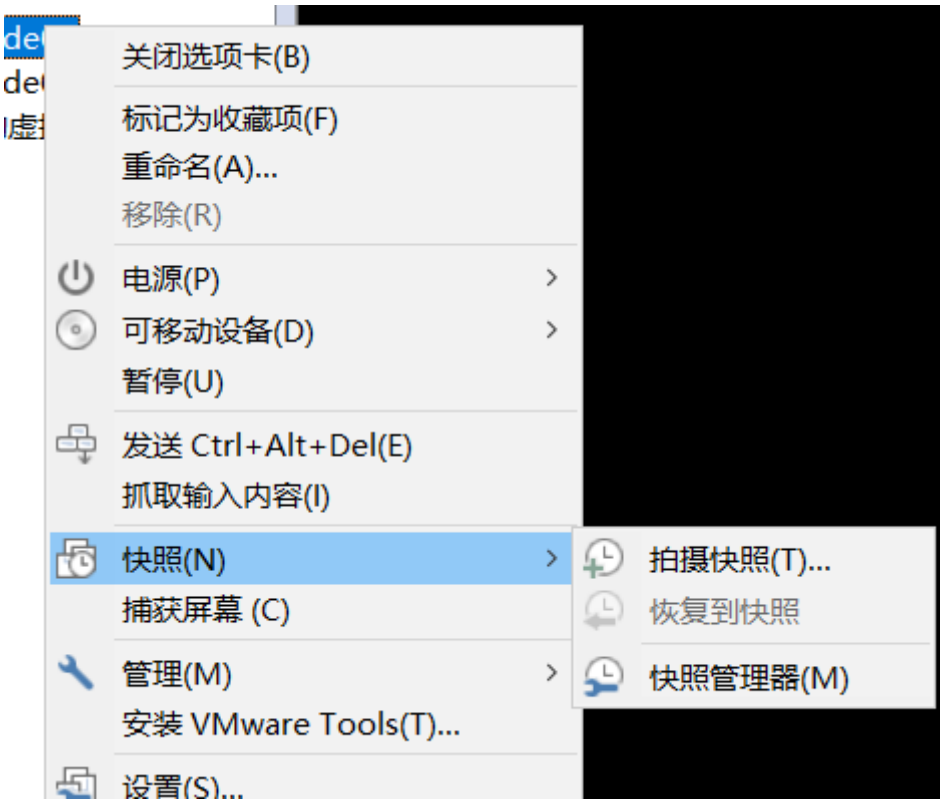
19.HDFS的常见命令

```
hadoop fs == hdfs dfs

命令的执行要在bin目录下
例：./hadoop fs -ls /
hadoop fs -ls / 查看
hadoop fs -lsr
hadoop fs -mkdir /user/haodop 创建文件夹
hadoop fs -put a.txt /user/hadoop 上传到hdfs
hadoop fs -get /user/hadoop/a.txt 从hdfs下载
hadoop fs -cp src dst 复制
hadoop fs -mv src dst 移动
hadoop fs -cat /user/hadoop/a.txt 查看文件内容
hadoop fs -rm /user/hadoop/a.txt 删除文件
hadoop fs -rmr /user/hadoop 删除文件夹
hadoop fs -text /user/hadoop/a.txt 查看文件内容
hadoop fs -copyFromLocal localsrc dst  与hadoop fs -put功能类似
hadoop fs -moveFromLocal localsrc dst 将本地文件上传到hdfs，同时删除本地文件
    2、帮助命令查看

hadoop帮助命令查看，不需要输入help，只需要在bin目录下输入即可。
例：./hadoop
    ./hadoop fs
```

10.完全分布式

host	NN	SNN	DN
node01	*		
node02		*	*
node03			*
node04			*

1.建立4台Linux主机



2.修改自己的主机名和网关

```
vim /etc/sysconfig/network-scripts/ifcfg-eth0
```

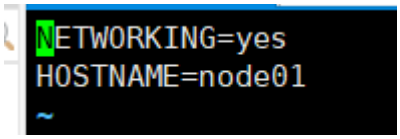
```
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=static
IPADDR=192.168.118.11
NETMASK=255.255.255.0
GATEWAY=192.168.118.2
DNS1=114.114.114.114
DNS2=192.168.118.11
```

IPADDR分配4个不一样的IP

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.118.11 node01
192.168.118.12 node02
192.168.118.13 node03
192.168.118.14 node04
```

分配4个主机名

```
vim /etc/sysconfig/network
```



3.重启网卡

```
service network restart
```

4.重启网卡要记住删除文件

```
rm -f /etc/udev/rules.d/70-persistent-net.rules
```

5.文件命令

```
scp xxx node:0x/xx scp是一种远程拷贝
```

6. `pwd` 可以在另一台主机同样位置进行定位

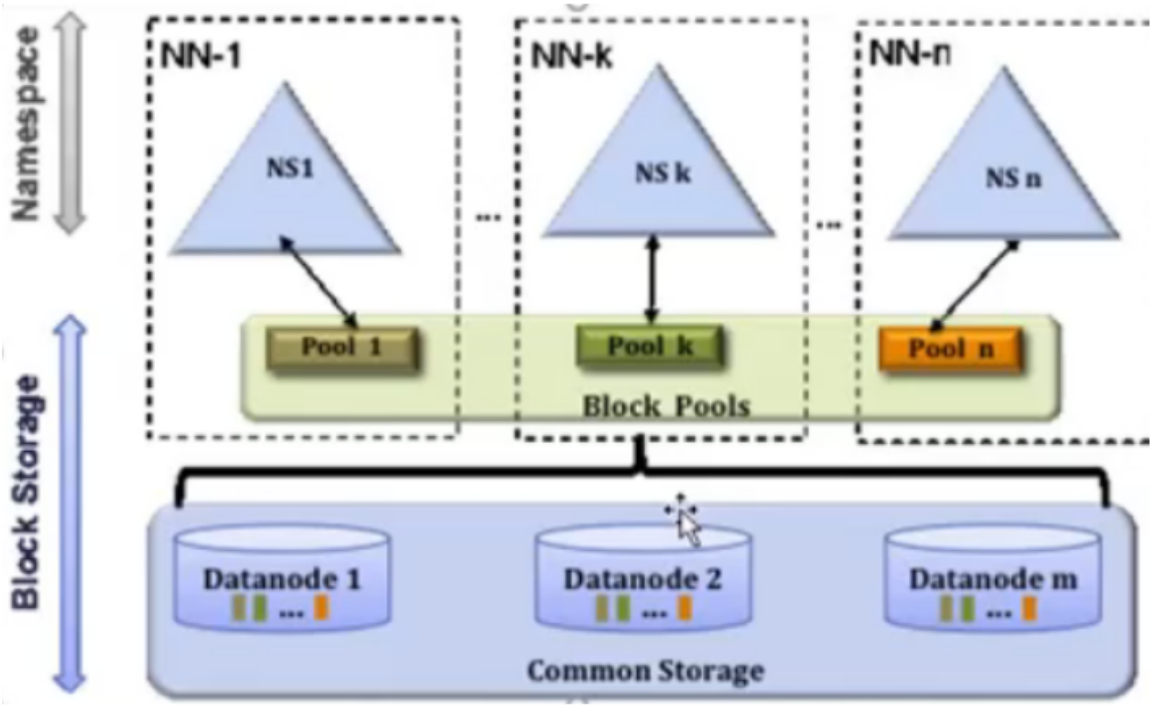
接下来就是从单节点变为多节点的配置，仅需要把各个node的节点修改为各个节点

11.HDFS单点故障解决方案

- 高可用方案：HA(High Available)
- 多个NN，主备切换：面临问题，压力过大，内存受限
- 联邦机制：Federation(元数据分片)
- 多个NN，管理不同元数据
- HADOOP 2.x 只支持HA一主一备
- HADOOP 3.x 支持最多5个主，官方推荐3个

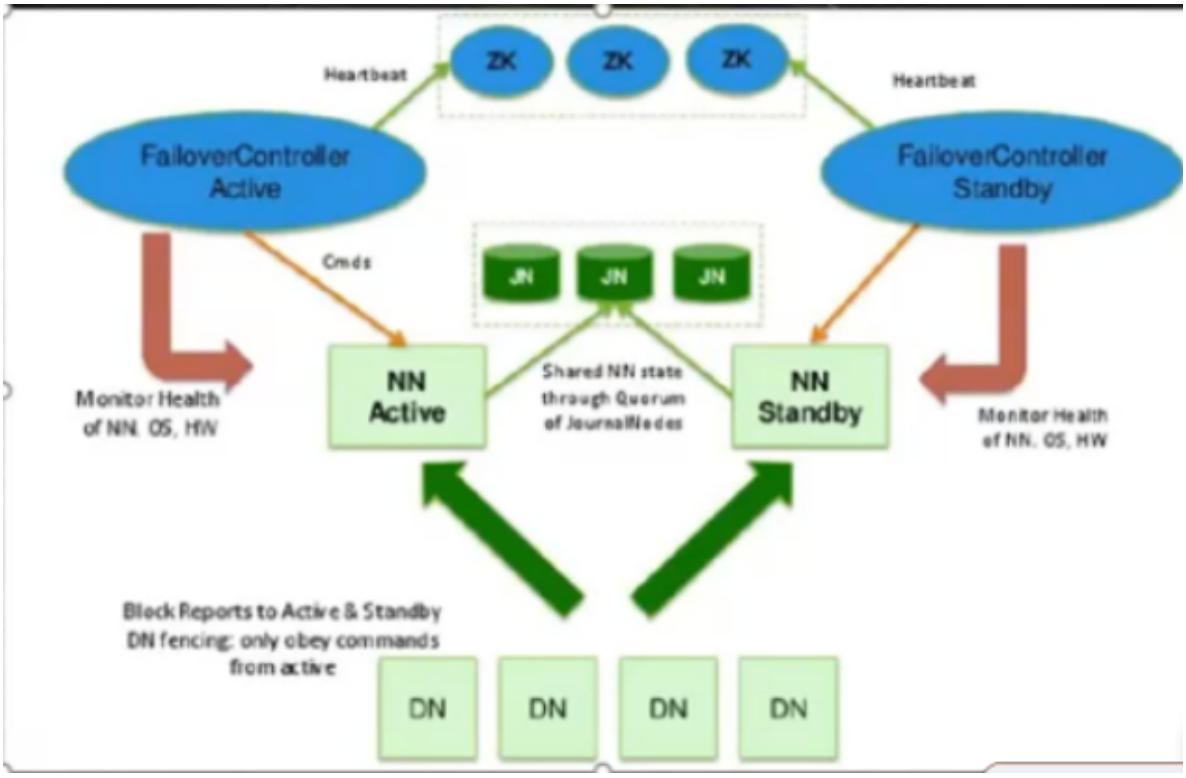
12.HDFS-Federation解决方案

- NameNode压力过大，内存受限问题
- 元数据分治，复用DN存储
- 元数据访问隔离性
- DN隔离block



联邦机制

13.HA模式



HA模式

HOST	NN	NN	SNN	DN	ZKFC	ZK
node01	*				*	
node02		*	*	*	*	*
node03				*		*
node04				*		*

HA节点分布

JoinNode 分布在node01,node02,node03

1.停止之前的集群

2.免密：node01,node02

```
node02:
cd ~/.ssh
ssh-keygen -t dsa -P '' -f ./id_dsa
cat id_dsa.pub >> authorized_keys
scp ./id_dsa.pub node01:`pwd`/node02.pub
node01:
cd ~/.ssh
cat node02.pub >> authorized_keys
```

3.zookeeper 集群搭建 java语言开发（需要jdk）

```
node02:
            tar xf zook....tar.gz
            mv zoo...    /opt/bigdata
            cd /opt/bigdata/zoo....
            cd conf
            cp zoo_sample.cfg  zoo.cfg
            vi zoo.cfg
                dataDir=/var/bigdata/hadoop/zk
                server.1=node02:2888:3888
                server.2=node03:2888:3888
                server.3=node04:2888:3888
            mkdir /var/bigdata/hadoop/zk
            echo 1 >  /var/bigdata/hadoop/zk/myid
            vi /etc/profile
                export ZOOKEEPER_HOME=/opt/bigdata/zookeeper-3.4.6
                export
PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$ZOOKEEPER_HOME/bin
            . /etc/profile
            cd /opt/bigdata
            scp -r ./zookeeper-3.4.6  node03:`pwd`
            scp -r ./zookeeper-3.4.6  node04:`pwd`
        node03:
            mkdir /var/bigdata/hadoop/zk
            echo 2 >  /var/bigdata/hadoop/zk/myid
            *环境变量
            . /etc/profile
        node04:
            mkdir /var/bigdata/hadoop/zk
            echo 3 >  /var/bigdata/hadoop/zk/myid
            *环境变量
```

```
            . /etc/profile

        node02~node04:
            zkServer.sh start
```

4.配置hadoop的core和hdfs

```
core-site.xml
        <property>
          <name>fs.defaultFS</name>
          <value>hdfs://mycluster</value>
        </property>

         <property>
           <name>ha.zookeeper.quorum</name>
           <value>node02:2181,node03:2181,node04:2181</value>
         </property>

 hdfs-site.xml
        #下面是重命名
        <property>
            <name>dfs.replication</name>
            <value>1</value>
        </property>
        <property>
            <name>dfs.namenode.name.dir</name>
            <value>/var/bigdata/hadoop/ha/dfs/name</value>
        </property>
        <property>
            <name>dfs.datanode.data.dir</name>
            <value>/var/bigdata/hadoop/ha/dfs/data</value>
        </property>
        #以下是  一对多，逻辑到物理节点的映射
        <property>
            <name>dfs.nameservices</name>
            <value>mycluster</value>
        </property>
        <property>
            <name>dfs.ha.namenodes.mycluster</name>
            <value>nn1,nn2</value>
        </property>
        <property>
            <name>dfs.namenode.rpc-address.mycluster.nn1</name>
            <value>node01:8020</value>
        </property>
        <property>
            <name>dfs.namenode.rpc-address.mycluster.nn2</name>
            <value>node02:8020</value>
        </property>
        <property>
            <name>dfs.namenode.http-address.mycluster.nn1</name>
            <value>node01:50070</value>
        </property>
        <property>
            <name>dfs.namenode.http-address.mycluster.nn2</name>
            <value>node02:50070</value>
        </property>
```

```
        #以下是JN在哪里启动，数据存那个磁盘
        <property>
            <name>dfs.namenode.shared.edits.dir</name>

<value>qjournal://node01:8485;node02:8485;node03:8485/mycluster</value>
        </property>
        <property>
          <name>dfs.journalnode.edits.dir</name>
          <value>/var/bigdata/hadoop/ha/dfs/jn</value>
        </property>

        #HA角色切换的代理类和实现方法，我们用的ssh免密
        <property>
          <name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider
</value>
        </property>
        <property>
            <name>dfs.ha.fencing.methods</name>
            <value>sshfence</value>
        </property>
        <property>
            <name>dfs.ha.fencing.ssh.private-key-files</name>
            <value>/root/.ssh/id_dsa</value>
        </property>

        #开启自动化：  启动zkfc
         <property>
           <name>dfs.ha.automatic-failover.enabled</name>
           <value>true</value>
         </property>
```

5.分发两个配置文件

scp命令

6.开启1,2,3台的journalnode

```
hadoop-daemon.sh start journalnode
```

7.选择一个NN 做格式化

```
hdfs namenode -format
```

8.启动该NN的namenode

```
hadoop-daemon.sh start namenode
```

9.在另一台NN进行同步

```
hdfs namenode -bootstrapStandby
```

10.在node01下格式化zk

```
hdfs zkfc  -formatZK
```

11.启动

```
start-dfs.sh
```

12.验证

```
kill -9  xxx
        a)杀死active NN
        b)杀死active NN身边的zkfc
        c)shutdown activeNN 主机的网卡 :  ifconfig eth0 down
            2节点一直阻塞降级
            如果恢复1上的网卡    ifconfig eth0 up
            最终 2 变成active
```

14.HDFS权限、企业级搭建、IDEA+Maven开发HDFS

hdfs权限介绍

HDFS是一个文件系统，所以权限也有

```
node01~node04:
    1)添加用户:root
        useradd god
        passwd god
        root
    2)资源与用户绑定（a.安装部署程序 b.数据存放的目录）
        chown -R god hadoop-2.6.5
        chown -R god /var/bigdata/hadoop/
    3)给god做免密
        ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
        cd ~/.ssh
        cat id_dsa.pub  >> authorized_keys
    4)切换到god去启动
        start-dfs.sh
```

用户权限实操

```
node01:
    su god
    hdfs dfs -mkdir /temp
    hdfs dfs -chown god:ooxx /temp
    hdfs dfs -chmod 770 /temp
node04:
    root:
```

```
        useradd good
        groupadd ooxx
        usermod -a -G ooxx good
        id good
    su good
        hdfs dfs -mkdir /temp/abc  <失败
        hdfs groups
            good:     <因为hdfs已经启动了，不知道你操作系统
                        偷偷创建了用户和组
node01:
    root:
        useradd good
        groupadd ooxx
        usermod -a -G ooxx good
    su god:
        hdfs dfs -refreshUserToGroupsMappings
    node04:
        good:
            hdfs groups
                good: good ooxx
```

结论：默认hdfs依赖操作系统上的用户和组

hdfs IDEA开发

```
hdfs的pom:
    hadoop:(common,hdfs,yarn,mapreduce)
        我们导入common 2.6.5  hdfs 2.6.5

public Configuration conf = new Configuration(true);
//这边为true会自动加载core-site.xml 默认在resource文件夹里面
fs = FileSystem.get(conf)
//去取环境变量 HADOOP_USER_NAME 的值

//创建文件夹
Path dir = new Path("/xxx00");
if(fs.exists(dir)){
    fs.delete(dir,true);
}
fs.mkdirs(dir);

//上传文件
BufferedInputStream bis = new ..
Path  out = new Path("/xxx.txt");
FSDataOutputStream output = fs.create(out)
IOUtils.copyBytes(input,output,conf,true)

//查看块信息
Path file = new Path("xxx");
FileStatus fss = fs.getFileStatus(file);
BlockLocation[] blks = fs.getFileBlockLocations(
                                fss,0,fss.getLen();
for(BlockLocation b:blks){
    System.out.println(b)
}
这边记录了偏移量信息，所以这边是大数据中计算向数据移动的体现
```