HDFS基本概念

1.HDFS介绍

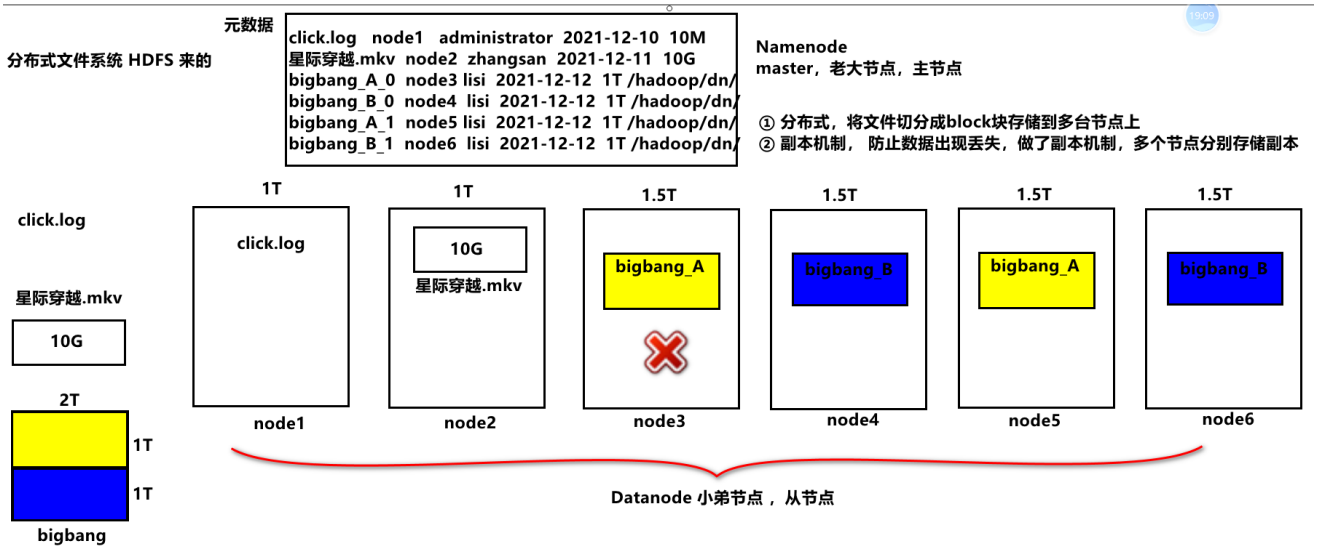
- HDFS是Hadoop DIstribute File System 的简称 ,意为:Hadoop分布式文件系统. 是Hadoop核心组件之一,作为最底层的分布式存储服务而存在
- 分布式文件系统解决的问题就是大数据存储

1.2HDFS设计目标

- 硬件故障是常态, HDFS将有成百上千的服务器组成,每一个组成部分都有可能出现故障.因此故障的检测和自动快速恢复是HDFS的核心架构目标.
- HDFS上的应用与一般的应用不同,它们主要是以流式读取数据.注重数据访问的高吞吐量
- 典型的HDFS文件大小是GB到TB的级别.所以,HDFS被调整成支持大文件.
- 大部分HDFS应用对文件要求的是write-one-read-many访问模型.一个文件一旦创建、写入、关闭之后就不需要修改了
- 移动计算的代价比之移动数据代价低.
- 在异构的硬件和软件平台上的可移植性.

2.HDFS重要特性

- 首先,它是一个文件系统,用于存储文件、存储数据,通过统一的命名空间目录数来定位文件;
- 其次,它是分布式的文件系统,分布式意味着多台机器存储.
-



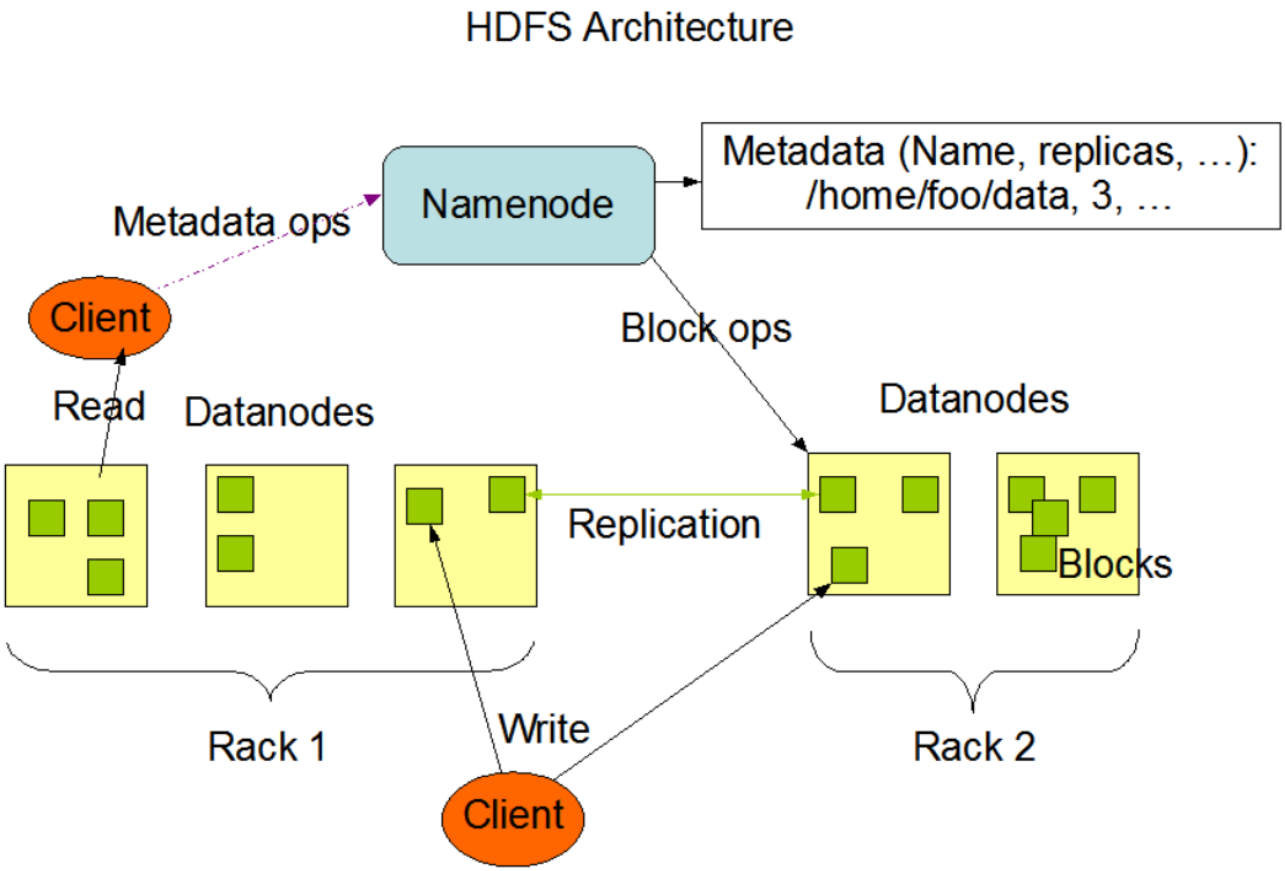
- 分布式多台机器存储
  - 问题:数据量大,单机存储遇到瓶颈
  - 解决:
    - 单机纵向扩展:磁盘不够加磁盘,又上线瓶颈限制
    - 多及横向扩展:机器不够加机器,理论上无线扩展
- 记录元数据

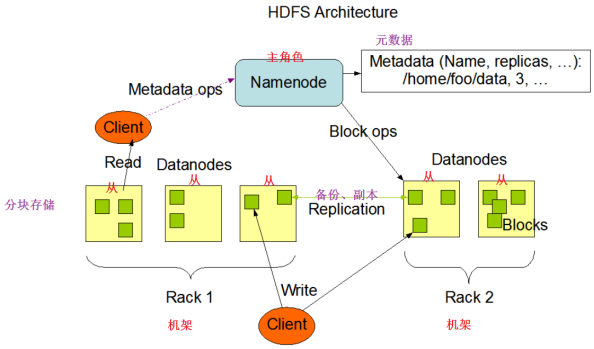
    - 问题:文件分布在不同机器上不利于寻找
    - 解决:
        - 元数据记录下文件及其存储位置信息,快速定位文件位置
- 分块存储
    - 问题:文件过大导致但存不下、上传下载效率低
    - 解决:
        - 文件分块存储在不同机器,针对块并行操作提高效率
- 副本机制(默认3备份)
    - 问题:硬件故障难以避免,数据易丢失
    - 解决: 为了容错,所有block都有副本,可以通过 dfs.replication 修改,默认是
        - 不同机器设置备份,冗余存储,

Hadoop团队针对HDFS设计目标

- 具备故障检测和快速恢复的能力
- 面对海量数据的存储,注重吞吐能力,而不是交互式(延迟高)
- 支持大文件存储(越大越开心)
- 一次写入多次读取模型(不支持修改操作)
- 异构存储、可移植性

HDFS重要特性





- master|slaves 主从架构.
  - 主角色:namenode,管理维护着元数据:目录树结构,文件大小,副本,备份,位置信息
  - 从角色:datanode,存储着最终的数据块
  - 秘书: secoudarynamenode ,帮助namenode合并元数据
- 分块存储.
  - 默认128M
  - 物理上把文件分开了,block size =128M 134217728 hadoop2.x (hadoop1.x 64M)
- 副本机制.
  - 冗余存储,提高安全
  - 默认是3副本,1+2=3,本身一份,额外两份,最终3副本
- namespace 名字空间,命名空间
  - namespace即"命名空间",也称'名称空间'
  - 层次感结构,兼顾传统对应文件系统的认知,目录树结构用户可以针对目录树进行文件夹、文件的增删改查.统一的抽象目录树
  - 从根开始,目录树
- matadata元数据
  - 元数据:记录数据的数据,描述性数据、解释性数据
  - 对于HDFS来说,目录结构及文件分块位置信息叫做元数据.
  - 元数据是由namenode维护的
- write noe read many (一次写入多次读取)
  - hdfs 的模式是一次写入多次读取
  - hdfs没有随机修改编辑的操作,只能对已有的数据进行追加.设计目标是这么决定的
  - 侧重于数据吞吐量,不注重实时交互性,意味着hdfs操作延迟很高

3.HDFS shell操作

- RPC访问地址 时使用hdfs的协议 hdfs://nn_host:8020

- webUI页面协议端口 http://nn_host:9870
  - shell命令解释

```
hadoop  fs <args>  文件系统的路径
hdfs dfs <args> 文件系统的路径

#hadoop  fs可以操作的文件系统不仅仅有HDFS,还包括本地文件系统、GFS、TFS。
#如何区分操作访问的是什么文件系统呢？  根据文件系统协议

#协议
hadoop fs -ls hdfs://node1:8020/ hdfs  #文件系统

hadoop fs -ls file:///  #本地文件系统
hadoop fs -ls gfs://  #谷歌文件系统

#如果不写协议 直接/目录 操作访问的是谁？
[root@node1 ~]# hadoop fs -ls /
Found 4 items
drwxr-xr-x   - root supergroup          0 2021-05-23 16:49 /itcast
drwx------   - root supergroup          0 2021-05-23 16:12 /tmp
drwxr-xr-x   - root supergroup          0 2021-05-23 16:12 /user
drwxr-xr-x   - root supergroup          0 2021-05-23 16:16 /wc

#默认是谁，取决于参数fs.defaultFS
<property>
        <name>fs.defaultFS</name>
        <value>hdfs://node1:8020</value>
</property>

#如果fs.defaultFS没有配置  默认的是file:///

#新旧命令  推荐使用hadoop fs
hadoop fs <args> = hdfs dfs <args>
```

- shell常见命令操作
  - HDFS hsell操作大多数和Linux shell命令类似

```
# 查看指定目录下信息
hadoop fs -ls [-h] [-R] <args>
    -h 人性化显示
    -R 递归显示
```

```
5
6  #创建文件夹
7  hadoop fs -mkdir [-p] <paths>
8      -p 创建父目录
9
10 #上传文件
11 hadoop fs -put src  dst
12 将单个 src 或多个 srcs 从本地文件系统复制到目标文件系统
13     #src代表的是本地目录 所谓的本地指的是客户端所在的机器
14     #dst代表的是HDFS
15     -p：保留访问和修改时间，所有权和权限。
16     -f：覆盖目的地（如果已经存在）
17
18 hadoop fs -put file:///root/itcast.txt hdfs://node1:8020/itcast
19
20 hadoop fs -put itcast.txt /itcast
21
22 #下载文件
23 hadoop fs -get  src  localdst
24     #将文件复制到本地文件系统。
25 hadoop fs -get hdfs://node1:8020/itcast/itcast.txt file:///root/
26 hadoop fs -get /itcast/itcast.txt ./
27
28 #追加内容到文件尾部 appendToFile
29 [root@node3 ~]# echo 1 >> 1.txt
30 [root@node3 ~]# echo 2 >> 2.txt
31 [root@node3 ~]# echo 3 >> 3.txt
32 [root@node3 ~]# hadoop fs -put 1.txt /
33 [root@node3 ~]# hadoop fs -cat /1.txt
34 1
35 [root@node3 ~]# hadoop fs -appendToFile 2.txt 3.txt /1.txt
36 [root@node3 ~]# hadoop fs -cat /1.txt
37 1
38 2
39 3
40 [root@node3 ~]#
41
42 #追加的用途：把本地的小文件上传中合并成为大文件 解决小文件场景的。
43
44 #文件内容的查看
```

```
cat 适合小文件
tail 将文件的最后一千字节内容显示到stdout  -f参数支持实时追踪查看

#权限 拥有者 所属组修改

    hdfs在设计的时候 借鉴模仿着linux权限管理模式
    也有所谓的读写执行 user group others  777
chgrp  修改所属组
chmod  修改权限
cgown  修改拥有者

hadoop fs -chmod 755 /1.txt

#文件移动 复制 删除
mv cp
rm -r递归删除


#合并下载 getmerge
合并下载多个文件  其功能和appendToFile相反的动作
[root@node3 ~]# hadoop fs -mkdir /small
[root@node3 ~]# hadoop fs -put *.txt /small
[root@node3 ~]# hadoop fs -getmerge /small/* ./merge.txt
[root@node3 ~]# cat merge.txt

#统计HDFS可用空间  指定目录大小
[root@node3 ~]# hadoop fs -df -h /
Filesystem            Size   Used  Available  Use%
hdfs://node1:8020  111.1 G  5.0 M     98.3 G    0%

#修改文件的副本数
hadoop fs -setrep -w N -R   N就是修改之后的副本数
-w wait等待 修改副本客户端是否等待修改完毕再推出

[root@node3 ~]# hadoop fs -setrep 2 /small/1.txt
Replication 2 set: /small/1.txt

[root@node3 ~]# hadoop fs -setrep -w 2 /small/2.txt
Replication 2 set: /small/2.txt
Waiting for /small/2.txt ...
```

```
WARNING: the waiting time may be long for DECREASING the number of replications.
. done

#企业中避免使用setrep修改文件的副本数。
副本的修改操作可能会影响hdfs正常的读写服务请求。
因此在实际工作中 事先根据数据的重要性在上传之前就决定该文件的备份数是多少 避免线上修改。
```

4.HDFS工作机制

- NameNode负责管理整个文件系统元数据;DataNode负责管理具体文件数据块存储;SecondaryNameNone协助NameNode进行元数据的备份
- HDFS的内部工作机制对客户端保持透明,客户端请求访问HDFS都是通过向NameNode申请来进行的

4.1HDFS基本原理

4.1.1NameNode概述

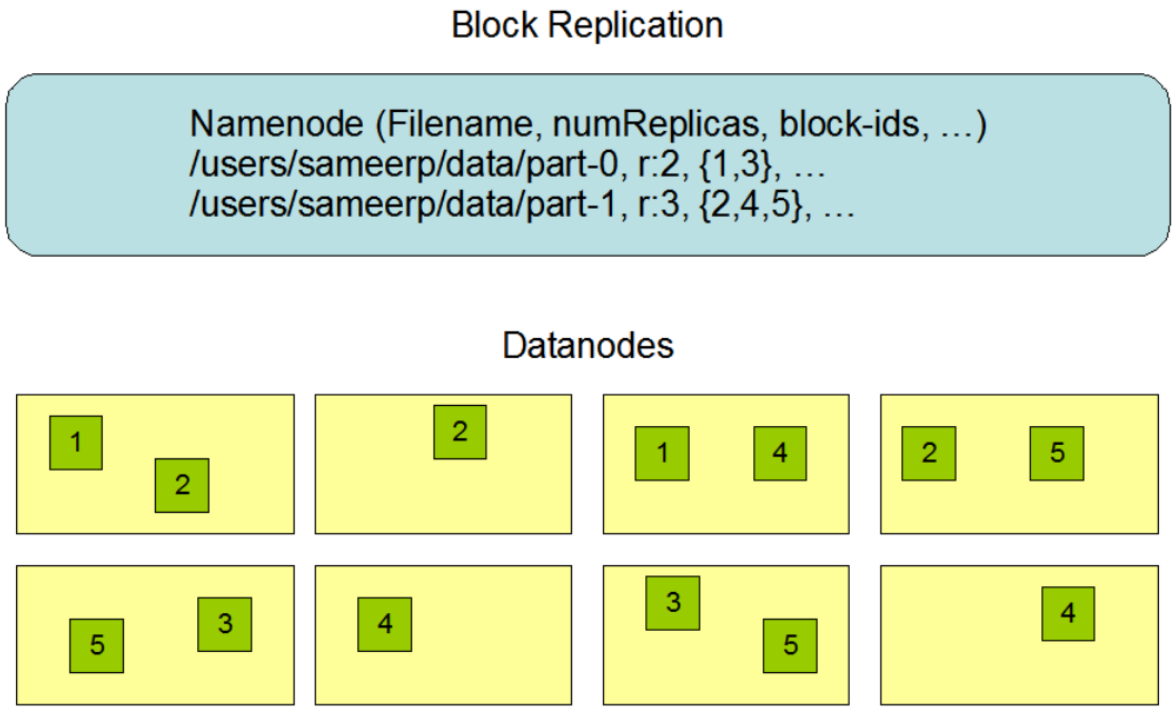
- NameNode是HDFS的核心
- NameNode也称为Master
- NameNode仅存储HDFS的元数据:文件系统中所有文件的目录树,并跟踪整合集群中的文件
- NameNode不存储实际数据或数据集.数据本身实际存储在DataNode中
- NameNode知道HDFS中任何给定文件的快列表及其位置.使用此信息NameNode知道如何从快中构建文件
- NameNode并不持久化存储每个文件中各个块所在的DataNode的位置信息,这些信息会在系统启动时从数据节点重建
- NameNode对于HDFS至关重要,当NameNode关闭时,HDFS/Hadoop集群无法访问
- NameNode是Hadoop集群中的单点故障
- NameNode所在机器通常会配置有大量内存

4.1.2DataNode概述

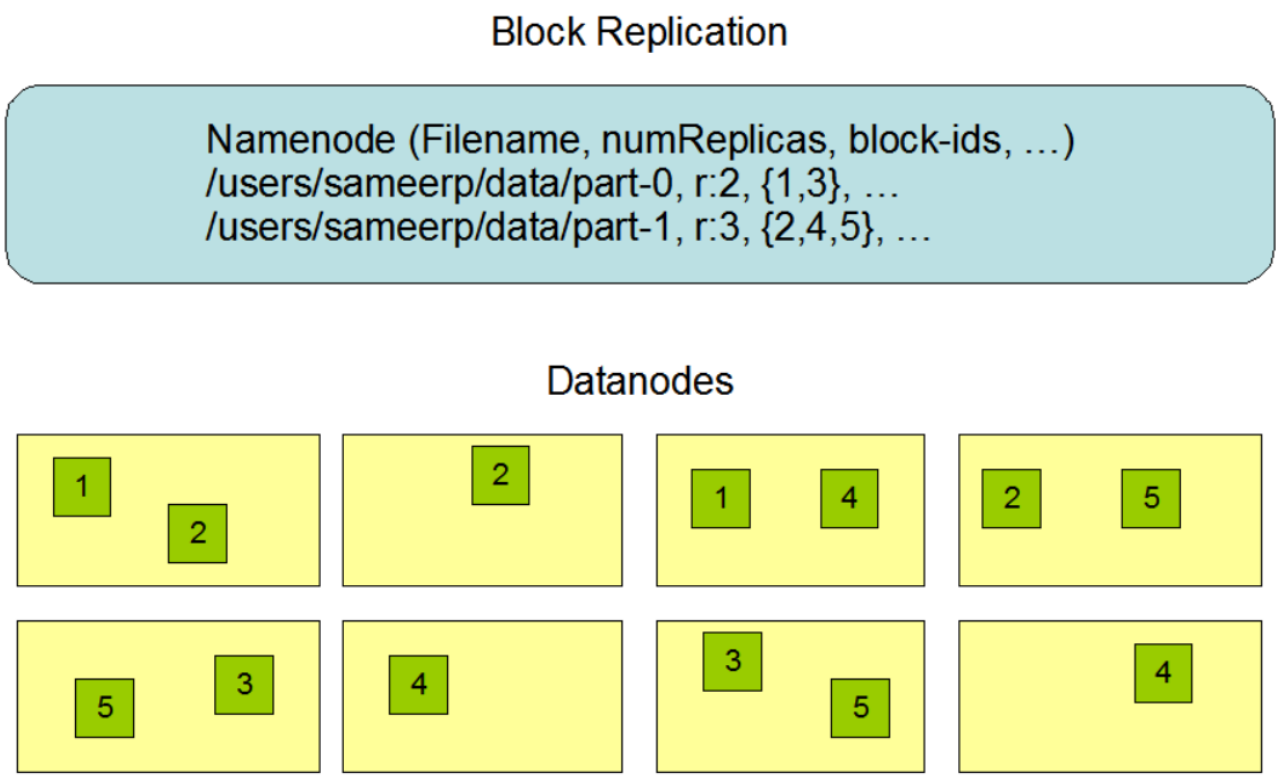
- DataNode负责将实际数据存储在HDFS中
- DataNode也称为slave
- NameNode和DataNode会保持不断通信
- DataNode启动时,它将自己发布到NameNode并汇报自己负责持有的块列表
- 当某个DataNode关闭时,他不会影响数据和群集的可用性.NameNode将安排由其他DataNode管理的块进行副本复制
- DataNode所在机器通常配置有大量的硬盘空间,因为实际数据存储在DataNode中
- DataNode会定期(dfs.heartbeat.interval 配置项配置,默认是3秒)向NameNode发送心跳,如果NameNode长时间没有接受到DataNode发送的心跳,NameNode就会认为该DataNode失效
- block汇报时间间隔参数dfs.blockreport,intervalMsec,参数为配置的话默认为6小时

4.1namenode、dataname职责

- namenode管理元数据,维护namespace

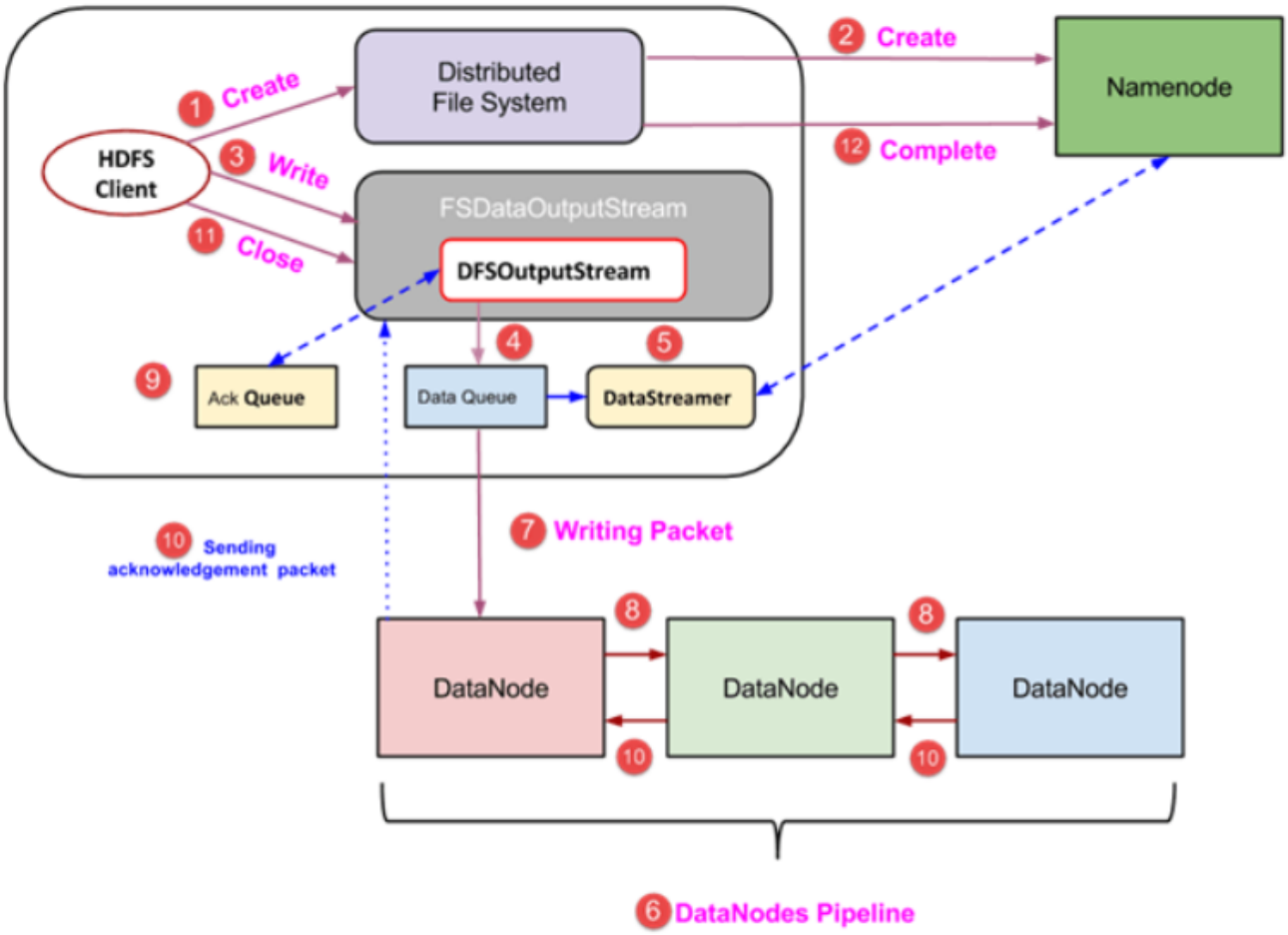


- datanode管理数据
-



4.2读写流程图

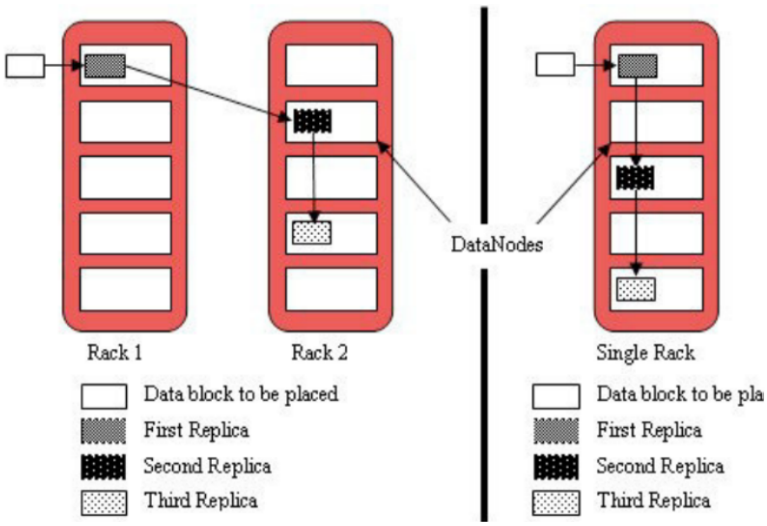
- 上传文件写数据流程
-



机架感知原理

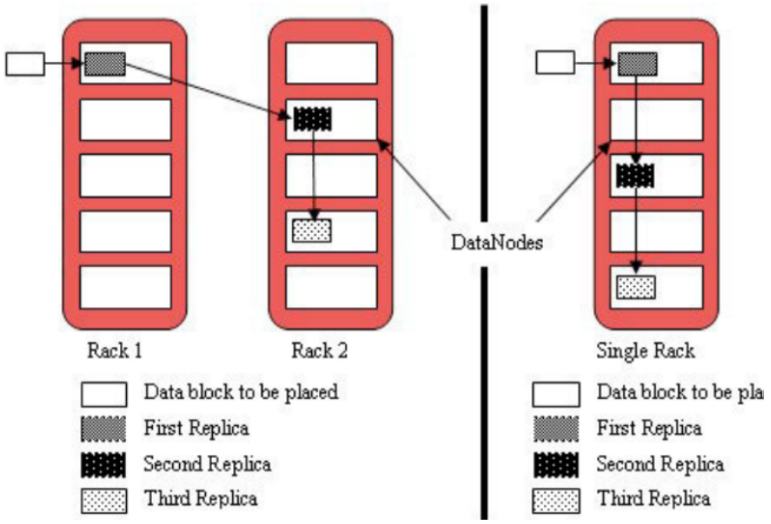
1. 找到 客户端Client最近的一个服务器存一份
2. 在随机选择一个机架上的服务器存储一份
3. 在这个机架的随机再找一台存储第三份

机架 Rack ， 存储服务器的机柜



机架感知原理
1. 找到 客户端Client最近的一个服务器存一份
2. 在随机选择一个机架上的服务器存储一份
3. 在这个机架的随机再找一台存储第三份

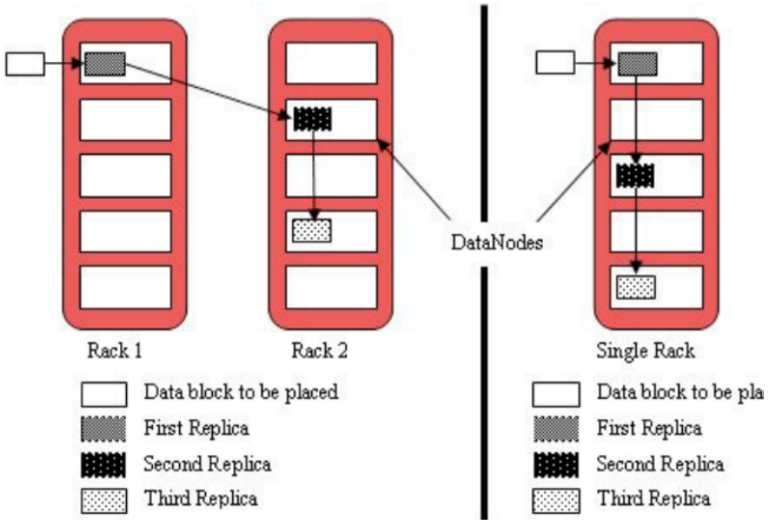
机架 Rack ， 存储服务器的机柜



- 上传文件写数据流程详细步骤分析
  - 1.client发起文件上传请求,通过RPC与NameNode建立通讯,NameNode检查是否有权限和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据及副本放置策略进行文件分配,返回可用的DataNode的地址,如:A,B,C;
    - 注:默认存储策略由BlockPlacementPolicyDefault类支持,也就是日常生活中提到最经典的3副本策略
    - 1st replica 如果写请求方所在机器是其中一个datanode,则直接存放在本地,否则随机在集群中选择一个datanode
    - 2nd replica 第二个副本存放与不同第一个副本的所在机架
    - 3rd replica 第三个副本存放与第二个副本所在的机架,但是属于不同的节点
  - 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pipeline ack发送给client
  - 7.当一个block传输完成后,client再次请求NameNode上传第二个block到服务器
- HDFS元数据3副本机制
-

机架感知原理
1. 找到 客户端Client最近的一个服务器存一份
2. 在随机选择一个机架上的服务器存储一份
3. 在这个机架的随机再找一台存储第三份

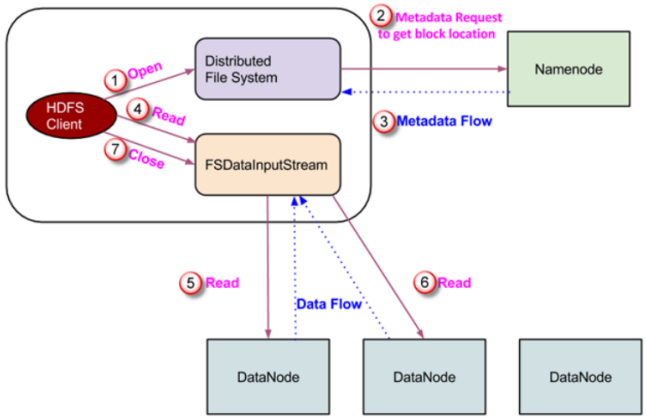
机架 Rack ， 存储服务器的机柜



- 下载文件读数据流程

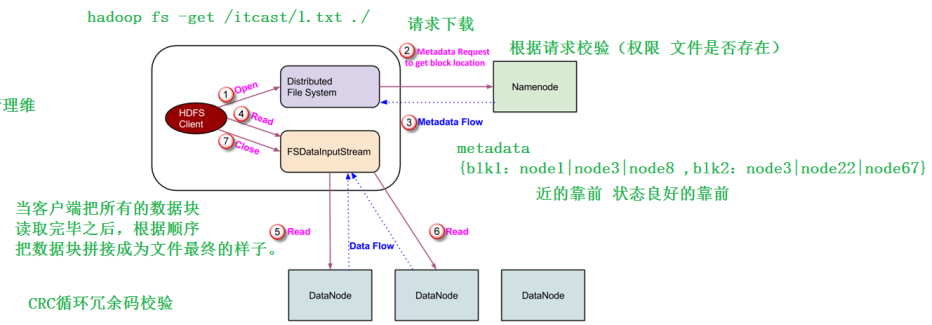
HDFS读取数据的流程

1.客户端Client请求数据向Namenode
hdfs dfs -get /readme.txt file:///root/data
2.向Namenode请求元数据，Namenode会
根据当前用户是否有权限，是否存在此文件
3.如果存在此文件并有权限读取数据会返回
数据块的资源列表
4.Client会向资源列表【node1 node3 node5】
请求 read 资源；
5.拿到每个数据块三备份并行读取数据，读取
完毕之后
6.依次读取下一块数据
7.Client将这些数据块合并生成一个新的数据文件



背景：
因为文件在HDFS上存储的时候是分块存储的，并且是分布式存储，那么
文件被分了几块？块在哪些机器上？这些信息决定了最终能否完整下载文件。

而这些信息我们称之为叫做元数据，被namenode管理维护。因此，读数据（下载文件）的第一步也是连接namenode



- **详细步骤分析:**

1.client向NameNode发起RPC请求,来确定是否有权限和请求文件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

3.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;心跳机制中超时回报的DN状态为STALE,这样的排靠后;
4.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
5.底层上本质是建立FSDataInputStream,重复地调用父类DataInputStream的read方法,直到这个块上的数据读取完毕
6.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;一旦客户端完成读取,它就会调用close()方法
7.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
8.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
9.最终读取来所有的block会合并成一个完整的最终文件

4.3NameNode与DataNode通信机制

- dn启动时
  - datanode向nameNode进行注册,并行汇报自己持有数据块信息
  - 注册表是自己启动成功,汇报是高速namenode自己保存了哪些数据块
- dn后续工作时
  - 心跳机制
    - datanode每隔3s向namenode进行心跳 目的:报活
    - dfs.heartbeat.interval
  - 数据块汇报机制blockreport
    - datanode每隔6小时向NameNode进行数据块汇报自己数据块信息,
    - dfs.blockport.intervalMsec

5.HDFS辅助工具

5.1集群内部的数据拷贝 scp

```
scp -r [文件夹/文件] 目的地服务器:地址
```

5.2跨集群复制数据

- distcp(distriuted copy)
- 功能:实现在不同的hadoop集群之间进行数据复制同步
- 用法:

```
#同一个集群内 复制操作
hadoop fs -cp /zookeeper.out /itcast
```

```
3
4 #跨集群复制操作
5 hadoop distcp hdfs://node1:8020/1.txt  hdfs://node5:8020/itcast
```

5.3文件归档工具 archive

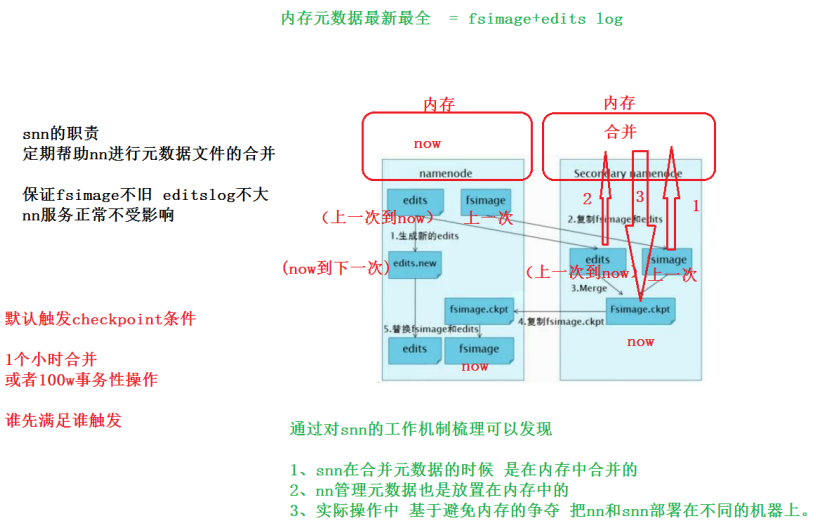
- 背景
  - hdfs的架构设计不适合小文件存储的
  - 因为小文件不管多小,都需要一定的元数据记录它,元数据存储在内存中
  - 如果集群小文件过多,就好造成内存被撑爆.俗称,小文件吃内存
- archive功能
  - 将一批小文件归档一个档案文件
  - 底层是通过MapReduce程序将小文件进行合并的.启动yarn集群执行mr程序.
  - 企业中可以根据时间,定时进行归档,比如一周创建一个档案
- 使用

```
1 #创建档案
2 hadoop archive -archiveName test.har -p /small /outputdir
3
4 基于自己的需求 删除小文件 减少对内存的消耗
5 hadoop fs -rm /small/*
6
7 #查看档案文件 --归档之后的样子
8 [root@node1 ~]# hadoop fs -ls hdfs://node1:8020/outputdir/test.har
9 Found 4 items
10 hdfs://node1:8020/outputdir/test.har/_SUCCESS
11 hdfs://node1:8020/outputdir/test.har/_index
12 hdfs://node1:8020/outputdir/test.har/_masterindex
13 hdfs://node1:8020/outputdir/test.har/part-0
14
15 #查看档案文件 --归档之前的样子
16 [root@node1 ~]# hadoop fs -ls har://hdfs-node1:8020/outputdir/test.har
17 Found 3 items
18  har://hdfs-node1:8020/outputdir/test.har/1.txt
19  har://hdfs-node1:8020/outputdir/test.har/2.txt
20  har://hdfs-node1:8020/outputdir/test.har/3.txt
21
22 #从档案文件中提取文件
23 [root@node1 ~]# hadoop fs -cp har://hdfs-node1:8020/outputdir/test.har/* /small/
24 [root@node1 ~]# hadoop fs -ls /small
```

```
Found 3 items
-rw-r--r--   3 root supergroup          2 2021-05-24 17:58 /small/1.txt
-rw-r--r--   3 root supergroup          2 2021-05-24 17:58 /small/2.txt
-rw-r--r--   3 root supergroup          2 2021-05-24 17:58 /small/3.txt
```

- 注意
  - archive没有压缩的功能,就是简单的合二为一的操作减少小文件个数.
  - 需要消耗源文件一样的空间;不支持压缩;创建achive时,源文件不会被更改或删除

6.HDFS namenode元数据管理机制

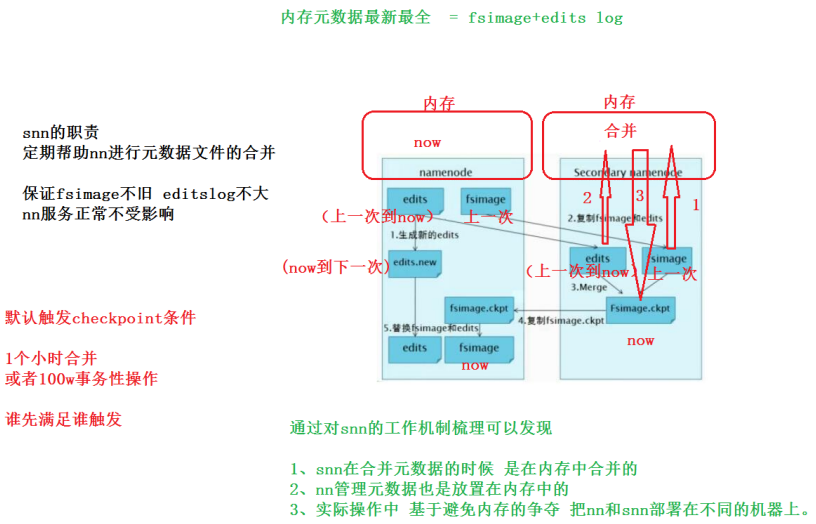


6.1namenode元数据

- 元数据是什么
  - 元数据(Metedata),又称中介数据、中继数据,为描述数据的数据(data about data),主要是描述数据属性(property)的信息,用来支持如指示存储位置、历史数据、资源查找、文件记录等功能
  - 记录数据的数据,描述数据的数据
- hdfs中元数据指的是什么
  - 文件系统的元数据(namespace、块的位置)
  - datanodes状态信息(健康、磁盘使用率)
  - 文件、目录自身的属性信息,例如文件名,目录名,修改信息等
  - 文件记录的信息存储相关的信息,例如存储块信息,分块情况,副本个数等
  - 记录HDFS的DataNode的信息,用于DataNode的管理
- HDFS磁盘上元数据文件分为两类
  - 内存元数据和元数据文件两种
  - fsimage镜像文件:

        - 是元数据的一个持久化的检查点,包含Hadoop文件系统中的所有目录和文件元数据信息,但不包含文件块位置的信息
    - Edits编辑日志:
        - 上一次到现在的操作
        - 存在的是Hadoop文件系统的所有更改操作(文件创建,删除或修改)的日志,文件系统客户端执行的更改操作首先会被记录到edits文件中
        - fsimage和edits文件都是经过序列化的,在NameNode启动的时候,将fsimage文件中的内容加载到内存中,之后再执行edits文件中的各项操作,使得内存中的元数据和实际的同步,存在内存中的原数据支持客户端的读操作,也是最完整的元数据
    - NameNode维护整个文件系统元数据
- 回想首次启动HDFS几群的时候进行format操作
    - 本质就是初始化操作,初始化namenode工作目录和元数据文件
    - 元数据存储的目录由参数dfs.namenode.name.dir决定在NN部署机器的本地Linux文件系统中
        - 针对课程环境,最终目录 : /export/data/hadoopdata/dfs/name

6.2secondarynamenode功能职责



- 要想成为namenode的备份,需要具备两个东西
    - 数据状态要和namenode保持一致
    - 承担和namenode一样的职责
- secondarynamenode根本不是namenode的备份,其主要职责帮助nameNode进行元数据的合并.
- replay重演机制
- 条件:
    - 日志超过64M
    - 每1小时

- 最大执行100万次

7.HDFS安全模式

- 安全模式(safe mode)是HDFS集群处于一种保护状态,文件系统只可以读,不可以写.
- 安全模式如何进入离开的?
  - 自动进入离开

```
#在HDFS集群刚启动时候 会自动进入 为了演示方便  使用单个进程逐个启动方式

#step1: 启动namenode
hadoop-daemon.sh start namenode

#step2: 执行事务性操作 报错
[root@node1 ~]# hadoop fs -mkdir /aaaa
mkdir: Cannot create directory /aaaa. Name node is in safe mode.

Safe mode is ON. The reported blocks 0 needs additional 52 blocks to reach the threshold 0.9990 of total blocks 52. The number of live datanodes 0 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.

#1、条件1:已经汇报的block达到总数据块的 0.999
#2、条件2:存活的dn数量大于等于0  说明这个条件不严格


#step3:依次手动启动datanode
hadoop-daemon.sh start datanode

Safe mode is ON. The reported blocks 52 has reached the threshold 0.9990 of total blocks 52. The number of live datanodes 2 has reached the minimum number 0. In safe mode extension. Safe mode will be turned off automatically in 25 seconds.
#3、条件3:满足12条件的情况下 持续30s 结束自动离开安全模式

Safemode is off.


#为什么集群刚启动的时候 要进入安全模式
文件系统元数据不完整 无法对外提供可高的文件服务  属于内部的元数据汇报、校验、构建的过程。
```

  - 手动进入离开

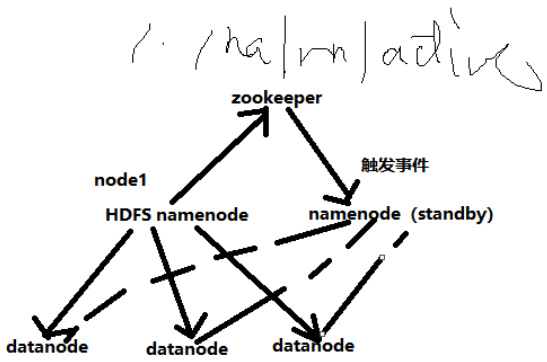
```
hdfs dfsadmin -safemode enter
hdfs dfsadmin -safemode leave

Safe mode is ON. It was turned on manually. Use "hdfs dfsadmin -safemode leave" to turn safe mode off.

#运维人员可以手动进入安全模式 进行集群的维护升级等动作 避免了群起群停浪费时间。
```

- 安全模式的注意事项
  - 刚启动完hdfs集群之后,等安全模式介绍才可以正常使用文件系统,文件系统服务才是正常可用
  - 后续如果某些软件依赖HDFS工作,必须先启动HDFS且等安全模式结束才可以使用你的软件
  - 启动=> 启动成功=>可用 (安全模式结束)

zookeeper和Hadoop Namenode之间的关系



zookeeper 和 hadoop 两个组件本身没有关系， zookeeper 帮助管理集群
HA高可用的时候管理集群 hadoop（hdfs，yarn）
怎么管理？ HA高可用，master（namenode）高可用