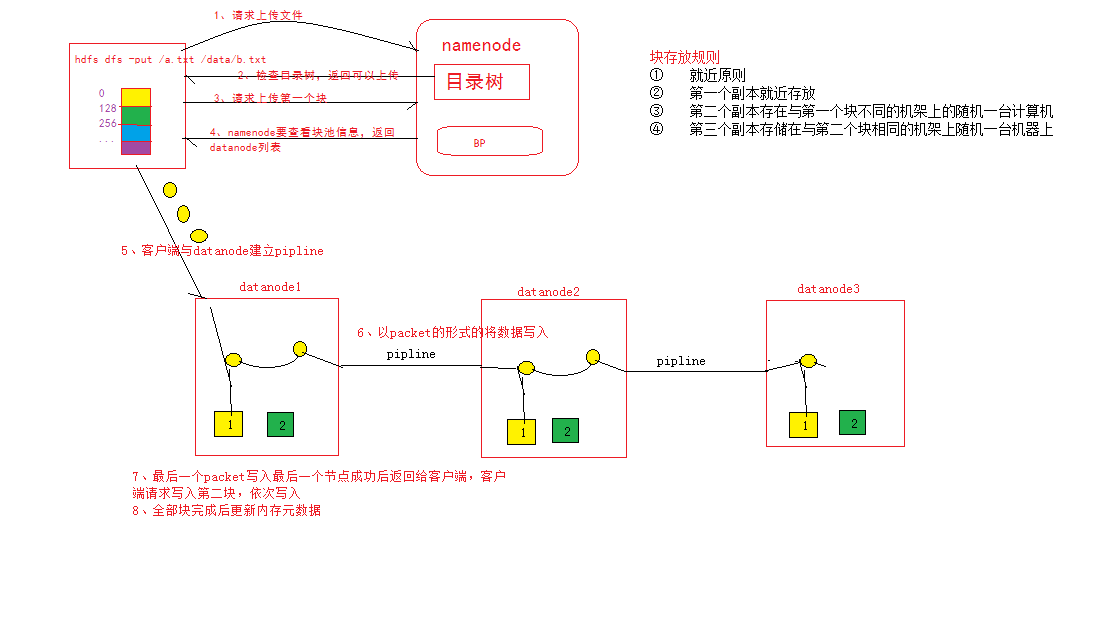
### HDFS的工作机制

1. 概述
   1. HDFS集群分为两大角色：namenode和datanode
   2. Namenode负责管理文件系统的元数据
   3. Datanode负责数据的存储，管理用户存储的文件数据块
   4. 文件会按照固定的大小切成块，分布式存储在若干台机器上
   5. 每一个块可以有多个副本，副本存放在不同datanode上
   6. Datanode定期的向namendoe汇报自身所维护的块的信息，namenode负责保持块的副本数量
   7. 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
2. 写数据流程
   1. 客户端先向namenode申请写入一个文件

hdfs dfs -put /etc/profile /a/b/c/a.txt 首先要确认目录是否存在

* 1. Namenode会查看自己维护的目录树，确认目录是否存在，若存在，通知客户端可以进行上传
  2. 客户端和namenode说我要上传第一个块，询问namendoe,第一个块存3份，存在哪儿
  3. Namenode查看自己块池的信息（均衡的分配），返回给客户端一个host列表，列表包含3个datanode
     1. 就近原则
     2. 第一个副本就近存放
     3. 第二个副本存在与第一个块不同的机架上的随机一台计算机
     4. 第三个副本存储在与第二个块相同的机架上随机一台机器上
     5. 注：机架感知策略（了解）
  4. 客户端就要和其中的一个datanode建立连接（就近原则）
  5. 多个datanode节点组成一个pipline
  6. 数据以packet的形式写入管道，并写入每个节点，在写当前节点的同时写给下一个节点
  7. 直至这个块的最后一个packet写完，整个块完成，继续存储下一个块
  8. 重复（3）-（8）的过程
  9. 最后一个块完成后，更新元数据（内存元数据）



1. 读数据流程
   1. 客户端跟namenode通信查询元数据，找到块文件所在的datanode服务器
   2. 挑选一台datanode（就近原则，然后随机）服务器，请求socket流
   3. Datanode开始发送数据给客户端（从磁盘里面读取数据写入流，以packet为单位的）
   4. 客户端已packet为单位去接收数据，先在本地缓存，然后写入目标文件

注：读数据时，会进行数据的校验，如果块损坏，读其他节点上的块

### namenode的工作机制

1. 工作职责：管理元数据、维护目录结构、响应客户端请求
   1. 元数据：内存元数据，磁盘元数据（dfs.namenode.name.dir目录下的fsimage和edits）
2. 元数据管理
   1. 内存元数据
   2. Fsimage：内存元数据的镜像
   3. Edits:日志

HDFS不适合存储小文件：每一个块的元数据信息大小在150B左右，如果存储大量的小文件，那么每一个小文件都有有一个内存元数据，namendoe的内存将很快被占满

1. 元数据手动查看
   1. hdfs oev -i edits\_0000000000000000917-0000000000000000918 -o edits.xml
   2. hdfs oiv -i fsimage\_0000000000000000000 -p XML -o ./fs.xml
2. 元数据的checkpoint
3. 每隔一段时间（默认1小时）secondarynamenode将 namendoe上的镜像文件和操作日志下载到由secondarynamendoe管理的目录下，将fsimage和edits加载到内存，将数据进行还原形成新的fsimage文件，将新的fsimage文件回传给namenode
   1. Checkpoint的出发条件
      1. 时间一小时
      2. 事物达到1000000次
   2. Checkpoint的检查目录

<!--镜像文件fsimage的检测目录-->

<property>

<name>dfs.namenode.checkpoint.dir</name>

<value>/hadoop/data/name</value>

</property>

<!--日志文件edits的检测目录-->

<property>

<name>dfs.namenode.checkpoint.edits.dir</name>

<value>/hadoop/data/edits</value>

</property>

<!--时间一小时-->

<property>

<name>dfs.namenode.checkpoint.period</name>

<value>3600</value>

</property>

<!--事物达到1000000-->

<property>

<name>dfs.namenode.checkpoint.txns</name>

<value>1000000</value>

</property>

1. namenode的启动过程（元数据恢复过程）
   1. 加载镜像文件
   2. 加载日志
   3. Blockreport（块的报告）
2. 安全模式

HDFS的一种工作状态，安全模式下只向客户端提供读的操作，不接受对命名空间的修改，同时namenode不会对数据块进行复制和删除

什么情况会进入安全模式？

块的数量小于阀值0.999f（默认）

存活的datanode的数量（默认是0）

如何离开安全模式？

使用命令离开hdfs dfsadmin -safemode enter|leave

修改配置文件调低阀值

重新格式化集群

### Datanode的工作职责

1. 存储管理用户的文件数据块，定期向namendoe报告自身所持有的块的信息（通过心跳机制）（心跳机制很重要，因为通过这种心跳机制，当集群中机器宕机，副本失效时namenode会感知到）datanode一旦宕机，不会立即下线，heartbeat.recheck.interval\*2+dfs.heartbeat.interval\*10

<property>

<name>heartbeat.recheck.interval</name>

<value>2000</value> ## 单位毫秒

</property>

<property>

<name>dfs.heartbeat.interval</name>

<value>3</value>

</property>

1. 心跳机制
   1. Hdfs是这种主从结构，namenode是master,datanode是slave.master会开启一个IPC服务，slave启动后，会主动连接IPC服务，并且至少每3面秒钟连接一次，这个时间是可以设置的。这种每隔一段时间连接一次的机制我们称为心跳机制
   2. Namendoe通过心跳机制得知datanode状态
2. 集群的动态上下线
   1. 动态的上线
      1. 配置好JDK，hadoop,
      2. 配置好namenode到新增加节点的免密码登录
      3. 配置slaves（以便下次namenode重新启动时会自动把新加入的节点启动）
      4. 每天机器上的主机映射文件
      5. 时间同步
      6. 关闭防火墙
      7. 负载均衡的问题
         1. start-balancer.sh自动启动负载均衡，默认阀值10%
         2. start-balancer.sh --threshold 5 启动负载均衡，阀值5%
   2. 动态的下线
      1. 配置配置文件

<property>

<name>dfs.hosts.exclude</name>

<value></value>

</property>

Value指向一个文件，文件中写入要下线的节点的hosts,每行一个

* + 1. 使用命令hdfs dfsadmin -refreshNodes
    2. 注:写在dfs.hosts.exclude文件中的节点，集群重新启动的时候不会再启动该节点



### HDFS配置的总结

core-site.xml

1. fs.defaultFS
2. hadoop.tmp.dir
3. Io.file.buffer.size
4. fs.trash.interval
5. fs.trash.checkpoint.interval

Hdfs-site.xml

1. dfs.namenode.name.dir

可以配置多个目录，将元数据相关信息存储多份，多个目录分配在多个磁盘上才有意义，一旦一个磁盘坏掉，其他磁盘上有备份的信息

1. dfs.datenode.data.dir

可以配置多个目录，多个目录也是在多个磁盘上，增大存储的空间

1. dfs.namenode.http-address
2. dfs.namenode.heartbeat.recheck-interval

计算节点退役的步长

1. dfs.heartbeat.interval
2. dfs.blockreport.intervalMsec

块报告时间间隔

1. dfs.blocksize#块的大小
2. Dfs.replacition 副本数量
3. dfs.permissions.enabled 文件系统权限验证
4. dfs.namenode.edits.dir namendoe的日志存储目录
5. dfs.namenode.safemode.extension 集群启动30秒安全模式
6. dfs.hosts
7. dfs.hosts.exclude
8. dfs.namenode.checkpoint.dir
9. dfs.namenode.checkpoint.edits.dir
10. dfs.namenode.checkpoint.period
11. dfs.namenode.checkpoint.txns
12. dfs.namenode.checkpoint.check.period
13. dfs.namenode.safemode.threshold-pct
14. dfs.namenode.safemode.min.datanodes

### HDFS的API

1. 配置环境
   1. 解压hadoop安装包，配置环境变量
   2. 将相应的windows的动态库文件和模拟器拷贝到hadoop的bin目录下
   3. 注：jdk的版本，JDK的安装路径不能有中文或者空格
   4. Hadoop version
2. 配置maven
   1. 安装maven和安装JDK一样
   2. 配置maven,配置maven/conf下的settings.xml
      1. 本地仓库路径
      2. 阿里云的镜像地址
      3. Jdk的默认版本
3. 解决权限问题
   1. 配置JVM环境-DHADOOP\_USER\_NAME=root
   2. 关闭权限验证

<property>

<name>dfs.permissions.enabled</name>

<value>true</value>

</property>

改成false

* 1. 设置用户名

System.*setProperty*("HADOOP\_USER\_NAME","root");

* 1. 获取文件系统时直接指定用户

fs = FileSystem.get(new URI("hdfs://192.168.91.3:9000"),conf,"root");

HDFS的设计目标:

1、大数据文件，非常适合上T级别的大文件或者一堆大数据文件的存储，如果文件只有几个G甚至更小就没有必要使用hdfs。

2、文件分块存储，HDFS会将一个完整的大文件平均分块存储到不同计算机上，它的意义在于读取文件时可以同时从多个主机取不同区块的文件，多主机读取比单主机读取效率要高得多。

3、流式数据访问，即数据批量读取而非随机读写，Hadoop擅长做的是数据分析而不是事务处理。一次写入多次读写，这种模式跟传统文件不同，它不支持动态改变文件内容，而是要求让文件一次写入就不做变化，要变化也只能在文件末添加内容。

4、廉价硬件，硬件错误是常态。因此需要冗余。HDFS可以应用在普通PC机上，这种机制能够让给一些公司用几十台廉价的计算机就可以撑起一个大数据集群。

5、硬件故障，HDFS认为所有计算机都可能会出问题，为了防止某个主机失效读取不到该主机的块文件，它将同一个文件块副本分配到其它某几个主机上，如果其中一台主机失效，可以迅速找另一块副本取文件。

6、程序采用“数据就近”原则分配节点执行