## 1) hdfs的基本概念:

HDFS是 分布式文件存储系统

HDFS就是一个hadoop的分布式文件存储系统, 是大数据中专门用来存储海量数据的容器,支持存储上TB/PB级别以上的数据

## 2) HDFS的应用场景:

适用场景:

1) 存储大文件 (单个文件比较大)

2) 一次写入、多次读取 不支持随机修改一个文件内容

3) 构建HDFS, 有廉价的服务器即可满足数据存储

4) 对数据提供扩展能力,保证数据不丢失

5) 对实时性要求不高的场景

不适用场景:

1) 低延时的数据访问 对延时要求在毫秒级别的应用

2) 大量小文件, 也不适合使用HDFS

3) 需要对文件内容进行随机修改的场景

## 3) HDFS的架构:

HDFS的client: hdfs的客户端

作用:

发起请求

负责文件块的切分操作 和 合并过程

hdfs的主节点: namenode 可以有多个的, 最多只有2个

作用:

存储元数据信息(内存的)

管理众多的从节点

接收客户端发送过来请求, 并下datanode下达命令

hdfs的从节点: datanode 可以有多个的

作用:

负责数据的存储

负责数据的最终读写操作

datanode要主动和namenode保持心跳连接

hdfs的辅助接点: secondnamenode

作用:

赋值namenode进行元数据的管理

副本机制:

hdfs将文件会拆分多个block块的, 默认情况下, 一个block大小为128M

每个block可以构建多个副本, 副本数量最多和datanode的节点数量是相等的

默认情况下 为 3个

注意: block本质上是一个逻辑架构, 并不是真实占用空间, 一个块具体占用多大的磁盘空间

取决于块中文件数据的大小,最多为128M

负载均衡机制:

namenode在分配块放置在那些节点的时候, 会尽可能保证让每个datanode块是均衡得

心跳机制:

datanode每隔3秒 会给namenode发送一次心跳包, 报告自己的状态, 如果在一定时间(10分钟+30s)范围内

没有发送心跳包, 那么namenode会认为当前这个datanode是否已经宕机了

4) HDFS的机架感知原理: 描述当有多个机架的时候, 副本如何放置的问题

第一副本：优先放置到离写入客户端最近的DataNode节点，如果上传节点就是DataNode，则直接上传到该节点，如果是集群外提交，则随机挑选一台磁盘不太慢，CPU不太忙的节点。

第二个副本：放置在另一个机架中, 某一个服务器中

第三个副本：放置在与第二个同机架的不同机器中

## 5) HDFS的shell命令的操作:

hadoop中hdfs的提供的命令客户端格式:

hadoop fs <args> 是在hadoop1.0版本就已经在开始使用了

hdfs dfs <args> 是hadoop2.0出现新的操作hdfs的命令

如果操作的时候 hdfs的, 建议采用 hdfs dfs 操作

选项名称 使用格式 含义

-ls -ls <路径> 查看指定路径的当前目录结构

-lsr -lsr <路径> 递归查看指定路径的目录结构

-du -du <路径> 统计目录下个文件大小

-dus -dus <路径> 汇总统计目录下文件(夹)大小

-count -count [-q] <路径> 统计文件(夹)数量

-mv -mv <源路径> <目的路径> 移动

-cp -cp <源路径> <目的路径> 复制

-rm -rm [-skipTrash] <路径> 删除文件/空白文件夹

-rmr -rmr [-skipTrash] <路径> 递归删除

-put -put <多个linux上的文件> <hdfs路径> 上传文件

-copyFromLocal -copyFromLocal <多个linux上的文件> <hdfs路径> 从本地复制

-moveFromLocal -moveFromLocal <多个linux上的文件> <hdfs路径> 从本地移动

-getmerge -getmerge <源路径> <linux路径> 合并到本地

-cat -cat <hdfs路径> 查看文件内容

-text -text <hdfs路径> 查看文件内容

-copyToLocal -copyToLocal [-ignoreCrc] [-crc] [hdfs源路径] [linux目的路径] 从本地复制

-moveToLocal -moveToLocal [-crc] <hdfs源路径> <linux目的路径> 从本地移动

-mkdir -mkdir <hdfs路径> 创建空白文件夹

-setrep -setrep [-R] [-w] <副本数> <路径> 修改副本数量

-touchz -touchz <文件路径> 创建空白文件

-stat -stat [format] <路径> 显示文件统计信息

-tail -tail [-f] <文件> 查看文件尾部信息

-chmod -chmod [-R] <权限模式> [路径] 修改权限

-chown -chown [-R] [属主][:[属组]] 路径 修改属主

-chgrp -chgrp [-R] 属组名称 路径 修改属组

-help -help [命令选项] 帮助

hdfs的常见命令:

hdfs dfs -ls [-R] path 查看hdfs的目录结构

hadoop fs -mkdir -p path 在hdfs中创建目录

hdfs dfs -put localFile hdfsPath 将linux本地文件上传到HDFS中 本地依然是存在的

hdfs dfs -moveFromLocal localFile hdfsPath 将linux本地文件移动hdfs的指定目录下

hdfs dfs -get hdfsFile localPath 将 hdfs的文件下载到本地

hdfs dfs -getmerge hdfsFile.... localPath 合并下载

hdfs dfs -mv hdfsPath hdfsPath 将文件从hdfs的某一个路径 移动到hdfs的另一个路径下

hdfs dfs -rm -r [-skipTrash] hdfsPath 删除hdfs中文件或者目录的操作

hdfs dfs -cp -r hdfsPath hdfsPath 将文件从hdfs的某一个路径 复制到hdfs的另一个路径下

hdfs dfs -cat hdfsPath 查看文件的内容

hdfs dfs -du hdfsPath 查看文件的大小

hdfs dfs -chmod -R 权限信息 hdfsPath 设置hdfs的目录或者文件的权限

hdfs的权限管理原则: 只能防止好人做错事, 不能防止坏人做坏事

hdfs dfs -appendToFile localFile hdfsFile 将本地的某一个文件内容, 追加到hdfs的某一个文件尾部

如果想要将按照目录从 /export/servers 移动到 /export/server下 : 可选项 仅仅是建议

第一步: 创建一个 /export/server

mkdir -p /export/server

第二步: 进入 servers目录下, 执行移动操作:

cd /export/servers

mv \* ../server

rm -rf /export/servers

前二步 是三台机子都要干的

第三步: 修改 hadoop的配置文件: hadoop-env.sh core-site.xml hdfs-site.xml

修改点:

将所有的servers的目录更换为server

第四步: 修改后, 将这个三个文件发送到第二台和第三台

cd /export/server/hadoop-2.7.5/etc/hadoop

scp -r hadoop-env.sh core-site.xml hdfs-site.xml node2:$PWD

scp -r hadoop-env.sh core-site.xml hdfs-site.xml node3:$PWD

第五步: 修改zookeeper的conf目录下的 zoo.cfg

修改一个目录结构

改后后, 通过scp命令发送给 node2 和 node3

第六步:修改 /etc/profile的环境变量: 三台都要改

改 jdk的目录 和 hadoop的目录

改了之后记得 source /etc/profile

## 6) hdfs的安全模式:

hdfs的安全模式是hdfs的一种保护机制 主要目的是为了保证hdfs中文件块的有效率

1) 在刚刚启动hadoop的时候, hdfs会首先进入安全模式下, 在这个模式下, 要求datanode向namenode报告块信息, 有namenode来检测

块的副本数量是否完整(99.9%)以上, 认为是块是完整的, 在这个过程中, 如果发现某一个块不足, 会自动通知datanode添加副本块

如果副本块多个, 就会通知datannode删除指定块副本操作

如果都ok, 安全模式将在30s 自动退出, 如果发现有问题 直到所有所有副本解决了, 才会自动退出

2) 在运行过程中, 也在定时检测, 如果发现有问题, 自动进行安全模式

安全模式的相关的命令:

hdfs dfsadmin -safemode get #查看安全模式状态

hdfs dfsadmin -safemode enter #进入安全模式

hdfs dfsadmin -safemode leave #离开安全模式

注意:

当后期发现迟迟无法退出安全模式, 可以尝试使用 hdfs dfsadmin  -safemode leave强制离开安全模式

离开之后, hdfs主页面就会报错, 报错中会显示那个数据的块存在问题

解决方案: 常规解决方案

1) 可以尝试先将这个数据下载下来

2) 将hdfs的对应的出现问题的数据进行删除

3) 重新上传上去即可

如果数据不重要:

直接删除即可

## 7) hdfs的压力检测(吞吐量检测)

第一步: 写入测试

hadoop jar /export/server/hadoop-2.7.5/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.5.jar TestDFSIO -write -nrFiles 10 -fileSize 10MB

第二步: 读取测试

hadoop jar /export/server/hadoop-2.7.5/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.5.jar TestDFSIO -read -nrFiles 10 -fileSize 10MB

第三步: 清理测试数据

hadoop jar /export/server/hadoop-2.7.5/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.5.jar TestDFSIO -clean

## 8) hdfs的数据写入流程:

1. 客户端发送写入请求给namenode

2. namenode接收到请求, 然后首先判断当前操作的用户是否具有写入的权限, 如果没有,直接报错 如果有权限, 接着判断要写入的数据目录下是否

存在这个文件, 如果存在, 直接报错, 如果不存在, 此时就会给客户端返回可以写入的状态

3. 当客户端获取可以写入的状态后, 开始对文件执行切割操作, 默认情况下, 会按照128M方式来切割文件

4. 客户端拿着第一个block再次请求namenode, 让namenode为这个blcok分配datanode地址列表

5. namenode会根据datanode的块信息,以及根据 机架感知原理, 网络拓扑关系 和 副本信息 来挑选出最合适的datanode地址, 将这些地址按照

由近到远的顺序放置到列表中返回客户端

6. 客户端根据列表中第一个地址连接指定的datanode, 然后由这个datanode连接下一次, 然后由下一个连接下一个依次类推,由此

形成一个pipeline的管道 , 同时反向还会形成一条ack的校验管道

7. 客户端开始写入. 数据以数据包的形成进行传输, 一个数据包为64kb, 当第一个datanode接收到, 接着传输给下一个, 依次类推

同时,每一个接收到数据后, 都要在ack校验通道中进行记录

8. 第一个datanode, 将ack中校验信息收集起来, 统计发送给客户端,由客户端校验此数据包是否全部都接收到了

9. 客户端一次次的开始传输, 一次次进行校验, 直到将第一个block传输完成, 接下来拿的第二个block再次请求namenode,

获取block要存储在那些datanode上, 接下来执行第6~8 , 直到将所有的block传输完成, 到此写入流程结束了

## 9) hdfs的数据读取流程

1) 客户端发起数据读取的请求

2) namenode接收到数据读取的请求, 首先判断要读取的文件是否存在, 如果不存在, 直接报错, 如果存在, 接着会判断当前操作的用户是否具备

读取数据的权限, 如果没有 直接报错, 如果有权限, namenode 会根据 block信息, 机架感知原理, 网络拓扑关系, 副本信息 ,本地原则,

返回这个文件部分或者全部的block的地址

3) 客户端拿到block部分或者全部地址, 接下来, 采用并行读取策略, 将block数据全部读取到客户端

4) 如果namenode之前返回的部分的block地址, 此时客户端接着再次请求namenode, 获取下一批的block地址,执行第四步

以此类推, 直到将所有的block全部的读取到客户端

5) 客户端将读取到block数据按照顺序, 依次拼接 形成最终的文件 返回给用户

## 10) SNN的数据写入的流程

第一步：通知namenode将hdfs更新记录写入一个新的文件——edits.new。

第二步：将fsimage和editlog通过http协议发送至secondary namenode。

第三步：将fsimage与editlog合并，生成一个新的文件——fsimage.ckpt。这步之所以要在secondary namenode中进行，是因为比较耗时，如果在namenode中进行，或导致整个系统卡顿。

第四步：将生成的fsimage.ckpt通过http协议发送至namenode。

第五步：重命名fsimage.ckpt为fsimage，edits.new为edits。

第六步：等待下一次checkpoint触发SecondaryNameNode进行工作，一直这样循环操作。

## 11) HDFS不同的集群之间数据拷贝:

11.1: 服务器之间数据拷贝操作: scp命令

从本地服务器将数据发送给远端服务器

方式1：指定用户名，命令执行后需要再输入密码；

scp -r local\_folder remote\_username@remote\_ip:remote\_folder

方式2:没有指定用户名，命令执行后需要输入用户名和密码；

scp -r local\_folder remote\_ip:remote\_folder

注意，如果实现了ssh免密登录之后，则不需要输入密码即可拷贝。

从远端服务器中将数据发送给本地服务器: 用的不是特别多

方式1：指定用户名，命令执行后需要再输入密码；

scp -r remote\_username@remote\_ip:remote\_folder local\_folder

方式2:没有指定用户名，命令执行后需要输入用户名和密码；

scp -r remote\_ip:remote\_folder local\_folder

注意，如果实现了ssh免密登录之后，则不需要输入密码即可拷贝。

11.2: 集群之间的数据拷贝的工作: 数据迁移

hadoop distcp hdfs://node1:8020/jdk-8u241-linux-x64.tar.gz hdfs://cluster2:8020/

底层执行MR 完成此处操作

## 12): HDFS的归档文件操作:

作用: 用来进行小文件合并的操作

方式一: 通过 javaAPI操作 实现小文件合并

使用场景: 同类型的文本数据合并工作

弊端: 多个文件数据合并在一个文件中, 导致后续进行拆分出现问题

方式二: 归档文件

一种类似于window中 zip方案, 将多个文件合并在一个文件中, 后续也可以将文件在解压还原出来

12.1. 如何创建归档文件:

hadoop archive -archiveName name -p <parent> <src>\* <dest>

例如:

hadoop archive -archiveName test.har -p /config /outputdir

描述:

创建一个归档文件为 test.har 将这个文件放置到 outputdir目录下, 归档文件中内容为 config目录下内容

12.2. 如何查看归档文件中内容:

看一下 归档中主要构成:

hadoop fs -ls /outputdir/test.har

查看归档文件中, 原来文件内容:

hadoop fs -ls har://hdfs-node1:8020/outputdir/test.har

上述可以简写为:

hadoop fs -ls har:///outputdir/test.har

12.3. 如何将归档文件原有数据解压出来:

hadoop fs -mkdir /config2

hadoop fs -cp har:///outputdir/test.har/\* /config2

相关的注意事项:

1.Hadoop archives是特殊的档案格式。一个Hadoop archive对应一个文件系统目录。Hadoop archive的扩展名是\*.har；

2.创建archives本质是运行一个Map/Reduce任务，所以应该在Hadoop集群上运行创建档案的命令，要提前启动Yarn集群；

3.创建archive文件要消耗和原文件一样多的硬盘空间；

4.archive文件不支持压缩，尽管archive文件看起来像已经被压缩过；

5.archive文件一旦创建就无法改变，要修改的话，需要创建新的archive文件。事实上，一般不会再对存档后的文件进行修改，因为它们是定期存档的，比如每周或每日；

6.当创建archive时，源文件不会被更改或删除；

13). HDFS的数据快照功能: 实际生产中使用较少

主要是对HDFS的文件夹 进行拍摄快照, 提供一种备份的操作

HDFS的数据快照功能 主要是采用的差异化快照:

首先对某一个文件夹拍摄快照, 此时在快照文件中不会进行任何记录操作, 只有当这个拍摄快照文件夹中

进行相关的操作, 此时快照中才会记录修改的内容即可

默认情况: 文件夹不支持拍摄快照

注意: 快照只能给文件夹拍摄, 不允许给文件拍摄快照

HDFS的快照的相关的操作:

1、开启指定目录的快照功能

hdfs dfsadmin -allowSnapshot 路径

2、禁用指定目录的快照功能（默认就是禁用状态）

hdfs dfsadmin -disallowSnapshot 路径

3、给某个路径创建快照snapshot

hdfs dfs -createSnapshot 路径

4、指定快照名称进行创建快照snapshot

hdfs dfs -createSanpshot 路径 名称

5、给快照重新命名

hdfs dfs -renameSnapshot 路径 旧名称 新名称

6、列出当前用户所有可快照目录

hdfs lsSnapshottableDir

7、比较两个快照的目录不同之处

hdfs snapshotDiff 路径1 路径2

8、删除快照snapshot

hdfs dfs -deleteSnapshot <path> <snapshotName>

## 14) HDFS的垃圾桶的机制:

为什么文件系统需要有垃圾桶呢? 给一个反悔机会

hdfs的回收站, 默认情况下 7天之后, 自动删除掉 此时间也可以在 core-site.xml中进行修改配置的

<property>

<name>fs.trash.interval</name>

<value>10080</value>

</property>

/\*单位为分钟\*/

如果想对回收站进行还原, 只需要进入回收站, 将对应数据移动到原有目录下即可还原操作

如果想要手动清空回收站: 此操作慎用 一般不做此操作

hadoop fs -expunge

注意:

如果使用 java API的中 delete操作, 直接跳过回收站, 直接删除数据