

#1 HDFS:

hadoop 主要解决海量数据存储和海量数据分析计算问题

#2 DFS思路:

#3 架构:

1. 每台机器可以透明地访问其他机器上的文件(通过RPC协议访问,对用户而言透明)
2. 将所有机器的文件系统都联合起来,用户以为都保存在自己的本地
3. 所有的文件资源通过Hash 函数分配到不同主机上, 然后访问文件可以通过Hash函数定位
4. client 与主节点master进行交互,主节点master进行管理, 从节点进行文件存储

#3 文件访问:


回忆操作系统的中公共资源的读写锁

#3 备份与一致性:

???

#2 HDFS思路:

#3 架构:

1. NameNode:存储文件的元数据,如文件名,文件目录结构,文件属性(生成时间、副本数,文件权限)
 1. 管理副本数(几个副本)
 2. 管理数据块(Block) 映射信息 (Block是最基础的存储单位, 是一个linux文件,HDFS默认大小是128MB)
 3. 处理客户端读写请求
2. DataNode: 在本地文件系统存储文件块数据,以及块数据的校验和
 1. 是一个slave,存储实际数据块
 2. 执行数据块的读/写操作
3. Secondary NameNode: 用来监控HDFS状态的辅助后台程序, 每隔一段时间获取HDFS元数据的快照,作为NameNode的备份
 1. 并非NameNode的热备.
 2. 辅助NameNode,分担其工作量,比如定期合并Fsimage和Edits,并推送给NameNode.
 1. FsImage: 内存在文件目录结构及其元信息在磁盘上的快照
 2. Editlog:两次快照之间,针对目录以及文件的修改部分image-20191227113553355
4. client :就是客户端
 1. 文件切分. 文件上传HDFS时,client将文件切分成一个个Block,然后上传
 2. 与NameNode交互,获取文件位置

3. 与DataNode交互,写入或读取信息
4. Client 提供一些命令 **管理** HDFS,比如NameNode格式化
5. Client 可以通过一些命令来 **访问** HDFS,比如对HDFS进行增删查改

#3 HDFS文件块:

块的大小:

HDFS中的文件在物理上时分块存储的(默认128M,老版本时64M)

寻址时间为传输时间的1%则为最佳状态(例如寻址时间时普遍为10ms,则 $10/0.01=1000\text{ms}=1\text{s}$)

当前磁盘的传输速率普遍时100MB/s

所以块的大小设置与磁盘传输速率有很大关系.

为什么块的大小不能太小也不能太大?

1. HDFS块的大小太小,则会导致寻址时间过大
2. 如果块设置太大,从磁盘传输数据的时间会明显大于定位这个块开始位置所需的时间,导致程序在处理这个数据块时会非常慢.其次,mapreduce中的job一次处理一个块,块太大会导致运行很慢

块的存储策略:

第一个副本:放置在上传文件的数据节点;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 (快速写入)

第二个副本:放置在与第一个副本不同的机架rack的节点上 (如果这个不同rack内节点发起读请求,那么可以减少跨rack的网络流量)

第三个副本:与第一个副本相同机架的其他节点上(如果第一个副本 所在节点和交换机故障,而该机架其它节点发起读请求,那么依然 可以保证数据访问)

更多副本:随机节点

#3 HDFS执行流程:

1. Client 向NameNode发起请求
2. NameNode反馈:
 1. 如果是读写操作,则告知Client文件存储的位置
 2. 如果是创建,删除,重命名,目录或文件等操作,NameNode修改文件目录结构成功后结束
 3. 对于删除操作,HDFS不会立即删去DataNode上的Block,而是等到特定时间才会删除
3. 对于读写文件操作,客户端获知具体信息后再与DataNode进行读写交互

#3 HDFS读数据:

1. 向NameNode发送请求
2. NameNode返回目标文件块的元数据(包含了同一数据块在不同节点的位置)
3. Client 向 距离最近的DataNode发送请求,DataNode 传输数据(△: 读不同的文件块不是并行的,读完一个后才可以读第二个)

#3 HDFS文件特性:

1. 一个文件一旦创建,写入和关闭后就不得修改,但支持append操作
2. 在文件被追加写时,会被上锁,其他节点的读写操作都将失效
3. 但一个文件被读时,其他节点的读操作是允许的

#3 HDFS容错:

1. NameNode故障: 根据SecondaryNameNode中的FsImage 和 Editlog进行恢复(所以说SNN并不是一个热备份)
2. DataNode故障: 节点上面所有的数据都会被标注为不可读写,同时自动备份到另一个available的节点上去.同时定期检查备份是否完好

#3 优点:

1. 高容错性: 数据自动保存多个副本(默认是三个). 它通过增加副本的形式,提高容错性
 1. 某个副本丢失之后,在另外的一个节点再创建一个副本(始终是三个)
2. 适合处理大数据
3. 可以构建在廉价的机器上

#3 缺点:

1. 不适合低延时的数据访问,比如毫秒级别的存储数据,是做不到的
2. 无法高效对大量小文件进行存储
 1. 存储大量小文件的话, NameNode的大量内存将会被用来存储这些文件的元信息,这样很可能会是NameNode的内存被耗尽
 2. 小文件存储的寻址时间会超过读取时间,它违反了HDFS的设计目标不支持并发写入
3. 不支持并发写入,文件随机修改
 1. 一个文件只能有一个写,不允许多个线程同时写
 2. 只支持数据的追加(append),不允许文件的随机修改

#3 Yarn的架构:

1. ResourceManager (集群资源的宗主,管理着集群中所有的CPU,内存等)主要作用如下
 1. 处理客户端请求
 2. 监控 NodeManager
 3. 启动或监控ApplicationMaster
 4. 资源的分配和调度
2. NodeManager 主要作用如下
 1. 管理单个节点上的资源
 2. 处理来自于ResourceManager的命令
 3. 处理来自于ApplicationMaster的命令
3. ApplicationMaster (负责集群中某一个任务的调度)作用如下:
 1. 负责数据的切分

2. 为应用程序申请资源并分配给内部的任务
3. 任务的监控和容错

4. Container的作用如下:

1. Container是Yarn中资源的抽象,它封装了某个节点上的多维度资源,如内存,CPU,磁盘,网络等.
2. 为application master服务, 因为每一个job运行都需要计算机资源

#2 Hadoop四大优势:

1. 高可靠性:Hadoop 底层维护了多个数据副本,所以即使Hadoop 某个计算元素或存储出现故障,也不会导致数据的丢失.
2. 高扩展性: 在集群间分配任务数据,方便扩展节点
3. 高效性: 在Mapreduce 的思想下, Hadoop 是并行工作的,以加快任务处理速度
4. 高容错性: 可以将失败的任务重新分配

#2 Hadoop1.x 和 Hadoop 2.x的区别:

1.x中mapreduce 的资源调度在2.x中由 Yarn来执行, Mapreduce只负责计算. 增强系统的模块化,降低耦合性.