# hadoop

## hdfs

### 体系结构

#### （★★）hdfs的优点与缺点

HDFS 具有以下优点：

(1) 高容错性

1) 数据自动保存多个副本。它通过增加副本的形式，提高容错性。

2) 某一个副本丢失以后，它可以自动恢复，这是由 HDFS 内部机制实现的，我们不必关心。

(2) 适合批处理

1) 它是通过移动计算而不是移动数据。

2) 它会把数据位置暴露给计算框架。

(3) 适合大数据处理

1) 数据规模：能够处理数据规模达到 GB、TB、甚至PB级别的数据。

2) 文件规模：能够处理百万规模以上的文件数量，数量相当之大。

3) 节点规模：能够处理10K节点的规模。

(4) 流式数据访问

1) 一次写入，多次读取，不能修改，只能追加。

2) 它能保证数据的一致性。

(5) 可构建在廉价机器上

1) 它通过多副本机制，提高可靠性。

2) 它提供了容错和恢复机制。比如某一个副本丢失，可以通过其它副本来恢复。

2、 HDFS 缺点：

(1) 不适合低延时数据访问；

1) 比如毫秒级的来存储数据，这是不行的，它做不到。

2) 它适合高吞吐率的场景，就是在某一时间内写入大量的数据。但是它在低延时的情况 下是不行的，比如毫秒级以内读取数据，这样它是很难做到的。

(2) 无法高效的对大量小文件进行存储

1) 存储大量小文件的话，它会占用 NameNode大量的内存来存储文件、目录和块信息。这样是不可取的，因为NameNode的内存总是有限的。

2) 小文件存储的寻道时间会超过读取时间，它违反了HDFS的设计目标。 改进策略

(3) 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) 仅支持数据 append（追加），不支持文件的随机修改。

#### （★★）HDFS的体系结构

Client：切分文件；访问或通过命令行管理HDFS；与NameNode交互，获取文件位置信息；与DataNode交互，读取和写入数据。

NameNode：Master节点，只有一个，管理HDFS的名称空间和数据块映射信息；配置副本策略；处理客户端请求。

DataNode：Slave节点，存储实际的数据；执行数据块的读写；汇报存储信息给NameNode。

Secondary NameNode：辅助NameNode，分担其工作量；定期合并fsimage和fsedits，推送给NameNode；紧急情况下，可辅助恢复NameNode，但Secondary NameNode并非NameNode的热备。

#### （★）相比于 HDFS1.0, HDFS 2.0 最主要的改进在哪几方面？

NameNode支持HA

命名空间支持分区(Federation)联邦

支持目录快照

https://blog.csdn.net/qw311113qin/article/details/81707258

### 工作原理

#### （★★★）hdfs底层存储设计

对于大数据来说，数据量是巨大的，多样性的，而且后期是要进行本地计算的，这样的话数据的存储要求是可靠的来保证数据的不丢失，HDFS就实现了这样的一个功能

HDFS存储数据的可靠性主要表现在其的副本机制，默认是3个，而且有其自己的一套存储策略。一个分布式的集群包含有很多的机架，每个机架又包含很多节点，机架内的节点之间的传输效率高于跨机架的节点之间的传输效率，并且机架之间机器的网络通讯通常要收到上层交换机的间网络带宽的限制。所以为了数据的安全和高效，Hadoop默认的存储策略是：

1、第一个副本存储在client所在的node中。(如果client不在集群中，则随机存储一个node)

2、第二个副本存储在与第一个副本不同机架的其他任意机架的任意node中。

3、第三个副本存储在与第一个副本同一个机架但是不同的node中。

4、如果还有其他副本设置，则随机选择存储。

这样的存储策略可以保证如果第一个节点失效时，能够优先在本机架找到另一个副本，另外如果整个机架发生了故障，能够保证在其他的机架中同样能够找到副本。这样既保证了数据的可靠，又兼容了高效。

#### （★★★）HDFS 的文件上传与下载底层工作原理（或 HDFS 部分源码分析）：FileSystem 的create()和 open()方法源码分析？

<https://www.cnblogs.com/qq503665965/p/6696675.html>

<https://www.cnblogs.com/qq503665965/p/6740992.html>

#### （★★★）HDFS 的 namenode 与 secondarynamenode 的工作原理（重点是日志拉取和合并过程）？

在NameNode端的工作流程如下：

1.Namenode始终在内存中保存metedata，用于处理“读请求” 到有“写请求”到来时，namenode会首先写editlog到磁盘，即向edits文件中写日志，成功返回后，才会修改内存，并且向客户端返回.

2.Hadoop会维护一个fsimage文件，也就是namenode中metedata的镜像，但是fsimage不会随时与namenode内存中的metedata保持一致，而是每隔一段时间通过合并edits文件来更新内容。

secondary namenode的工作流程: 1.secondary通知namenode切换edits文件

2.secondary从namenode获得fsimage和edits(通过http)

3.secondary将fsimage载入内存，然后开始合并edits

4.secondary将新的fsimage发回给namenode

5.namenode用新的fsimage替换旧的fsimage

#### （★★★）SecondaryNameNode的作用是什么?

SecondaryNameNode它的职责是合并NameNode的edit logs到fsimage文件中

首先，SecondaryNameNode定时到NameNode去获取edit logs，并更新到fsimage上。一旦它有了新的fsimage文件，SecondaryNameNode将其拷贝回NameNode中。NameNode在下次重启时会使用这个新的fsimage文件，从而减少重启的时间。

Secondary NameNode的整个目的是在HDFS中提供一个检查点。它只是NameNode的一个助手节点。这也是它在社区内被认为是检查点节点的原因。

#### （★★★）hdfs读写过程

（一）写数据流程

1).客户端发出请求 hdfs dfs -put /etc/profile /qf/data

2).namenode查看维护的目录结构，检查/qf/data是否存在，如不存在直接报错”no such file or directory“，

如存在返回给客户端同意上传文件请求，将操作写入日志文件

3).客户端请求上传第一个块，询问namenode块的存储位置

4).namenode查看自己的datanode池，返回给客户端一个datanode列表

5).客户端发出请求建立pipeline

6).客户端先把文件写入缓存，达到一个块的大小时,会在客户端和第一个datanode建立连接开始流式的传输数据，

这个datanode会一小部分一小部分（4K）的接收数据然后写入本地仓库，同时会把这些数据传输到第二个datanode，

第二个datanode也同样一小部分一小部分的接收数据并写入本地仓库，同时传输给第三个datanode

(在流式复制时，逐级传输和响应采用响应队列来等待传输结果。队列响应完成后返回给客户端)

7).第一个数据块传输完成后会使用同样的方式传输下面的数据块直到整个文件上传完成。

8).整个文件完成，namenode更新内存元数据

（二）读数据流程

1)客户端向namenode发起RPC调用，请求读取文件数据。

2)namenode检查文件是否存在，如果存在则获取文件的元信息（blockid以及对应的datanode列表）。

3)客户端收到元信息后选取一个网络距离最近的datanode，依次请求读取每个数据块。客户端首先要校检文件是否损坏，如果损坏，客户端会选取另外的datanode请求。

4)datanode与客户端建立socket连接，传输对应的数据块，客户端收到数据缓存到本地，之后写入文件。

5)依次传输剩下的数据块，直到整个文件合并完成。

#### （★★）hdfs的client 端，复制到第三个副本时宕机，hdfs怎么恢复保证下次写第三副本?block块信息是先写 dataNode 还是先写 nameNode?

Datanode会定时上报block块的信息给namenode ，namenode就会得知副本缺失，然后namenode就会启动副本复制流程以保证数据块的备份

Client向NameNode发起文件写入的请求。NameNode根据文件大小和文件块配置情况，返回给Client它所管理部分DataNode的信息。Client将文件划分为多个Block，根据DataNode的地址信息，按顺序写入到每一个DataNode块中。

#### （★）Client 端上传文件的时候下列哪项正确

A、数据经过 NameNode 传递给 DataNode

B、Client 端将文件切分为 Block，依次上传

C、Client 只上传数据到一台 DataNode，然后由 NameNode 负责 Block 复制工作

答案 B

分析：Client 向 NameNode 发起文件写入的请求。NameNode 根据文件大小和文件块配置情况，返回给 Client 它所管理部分 DataNode 的信息。Client 将文件划分为多个 Block，根据 DataNode 的地址信息，按顺序写入到每一个DataNode 块中。具体查看HDFS 写数据流程。

### HA高可用

#### （★★）HDFS HA 需要secondary namenode吗，灾难恢复流程

不需要 当启用HDFS HA时会启动多台NameNode但是只有一台处于活跃状态，尽管其他NameNode处于休眠但是依旧通过ZKFC与活跃NameNode进行元数据同步，当活跃NameNode挂掉时通过zookeeper重新选举一个NameNode接替

#### （★）HadoopHA 集群中，各个服务的启动和关闭的顺序

启动流程：先启动zookeeper，再启动hdfs，然后启动yarn最后单独启动resourcemanager

关闭流程：先关闭hdfs，再关闭yarn，然后单独关闭resourcemanager，最后关闭zookeeper

### 常见问题

#### （★）hdfs运维是常见问题(zabix，ganglia)

磁盘损坏：在规模集群中由于IO读写频繁，造成磁盘经常损坏，出现该问题可将坏的磁盘umount下即可，由于hdfs检查副本丢失一份，会自动复制一份。坏掉的磁盘如果修复好后，格式化后可在插入到磁盘槽中

添加新的服务器：由于添加新的服务器，原有的服务器中的数据过多，但新的服务器中的数据可能数据却很少，会造成负载不均匀，这个时候可以使用hdfs的reblance方法对数据进行重新负载

#### （★）HDFS 的常用 shell 命令有哪些？分别对应哪些 Client Java API？：显示文件列表、创建目录、文件上传与下载、文件内容查看、删除文件

答：显示文件列表: hdfs dfs –ls <path> Java:listStatus

创建目录:hdfs dfs -mkdir <path>

Java:mkdirs

文件上传与下载:hdfs dfs –put [-r] <path> java: copyToLocalFile

hdfs dfs –get <path> Java: copyFromLocalFile

文件内容查看:hdfs dfs –cat[-AbeEnstTuv] [--help] [--version] <path>

Java：listStatus

删除文件:hdfs dfs –rm [–r] [–f] <path> Java:delete

#### （★）因为 HDFS 有多个副本，所以 NameNode 是不存在单点问题的。（错误）

[https://blog.csdn.net/nankangren123/article/details/20924681](https://www.jianshu.com/p/9b1304a1db4d)

#### （★）Hadoop1.0 和 2.0 都具备完善的 HDFS HA 策略。（错误）.

2.0有完善的HA

#### （★）HDFS 中的 block 块默认保存几份？默认大小多少？

3份，128M

#### （★★）HDFS，replica 如何定位

您可以使用HDFS命令行来ls文件。

输出的第二列将显示文件的复制因子。

例如，

$ hdfs dfs -ls /usr/GroupStorage/data1/out.txt

-rw-r--r-- 3 hadoop zma 11906625598 2014-10-22 18:35 /usr/GroupStorage/data1/out.txt

out.txt的复制因子是3。

方法2：使用stathdfs命令工具获取复制因子。

以上述文件为例：

$ hdfs dfs -stat %r /usr/GroupStorage/data1/out.txt

它会打印出来 3

#### （★）hadoop dfsadmin –report 命令用于检测 HDFS 损坏块。（错误）

此命令用于查看hadoop的状态信息

Configured Capacity: 0 (0 B)

Present Capacity: 0 (0 B)

DFS Remaining: 0 (0 B)

DFS Used: 0 (0 B)

DFS Used%: NaN%

Under replicated blocks: 0

Blocks with corrupt replicas: 0

Missing blocks: 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codec | File Extension | Splittable? | Degree of Compression | Compression Speed |
| Gzip | .gz | No | Medium | Medium |
| Bzip2 | .bz2 | Yes | High | Slow |
| Snappy | .snappy | No | Medium | Fast |
| LZO | .lzo | No, unless indexed | Medium | Fast |

#### （★★★）hdfs 的数据压缩算法