Datanode源码分析

Datanode以数据块的形式在本地文件系统上 保存了HDFS文件的内容，并提供文件数据访问功能。客户端读写HDFS文件时，首先从Namenode获取文件及datanode信息，然后进一步与Datanode交互。同时Datanode受namenode的管理，执行Namenode指令，并通过心跳、注册等将Datanode信息上报给Namenode。

# 1、Datanode的磁盘目录文件结构

在HDFS中，namenode/secondarynamenode和datanode都需要在磁盘上组织、存储持久化数据，而且在磁盘上会维护一定的文件结构。在第一次启动HDFS集群时，会对namenode节点进行格式化(bin/hadoop namenode -format)，但是datanode不需要格式化。另外，数据节点可以管理多个数据目录，由${dfs.data.dir}指定，数据目录保存数据块。配置如下：

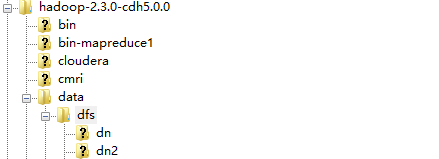
*<property>*

*<name>dfs.data.dir</name>*

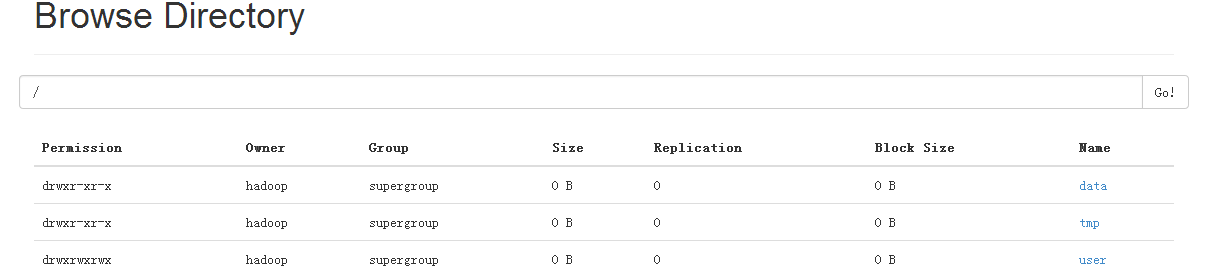
*<value>/data/dfs/dn,/data1/dfs/dn</value>*

*</property>*

启动dfs后，在../dfs下，存在两个数据目录，分别是/dn,/dn2，如下图所示：



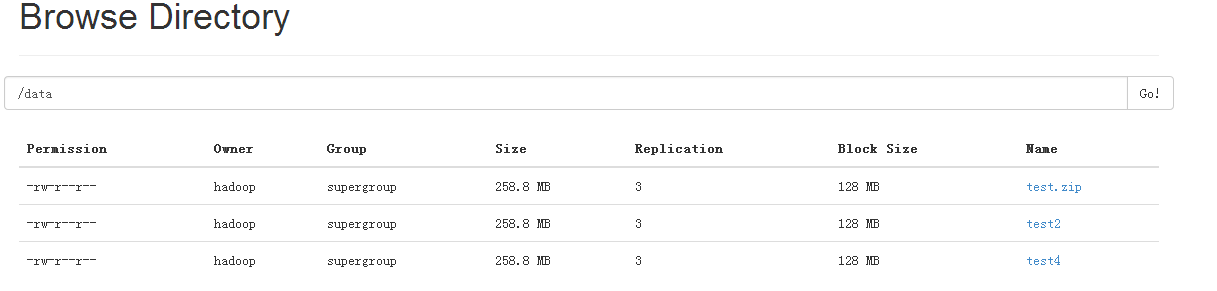
如果初始时，../dn,../dn2不存在时，系统会自动创建这两个目录。之所以可以配置多个目录，是为了保证数据读写的并发性，多个目录可以同时进行数据的读写。为了最有效进行数据操作，可以将多个目录创建在不同的盘上。虽然存在多个目录，但是用户可见的目录结构不存在目录的区分，例如上面的例子中，通过Web UI上，显示数据，如下图所示。每个datanode节点可以配置的dfs.data.dir可以不同，每个目录都对应本地Linux操作系统的一个目录。但是在../dfs下可以看到所有节点配置的{dfs.data.dir}及目录下的blk数据，但是datanode只能写入本节点配置的{dfs.data.dir}中（通过in\_use.lock来控制）。



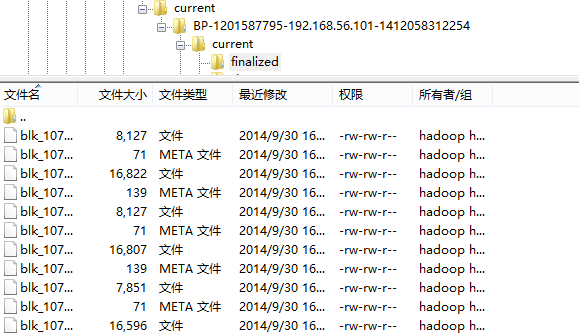
例如，在host1及host2上同时执行:

*bin/hadoop fs -copyFromLocal test.zip /data/testxx*

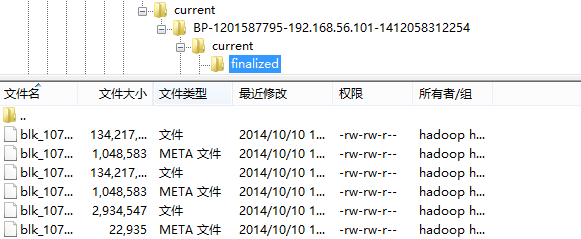
执行后的结果如下图所示：



数据在../dn及../dn2均放置数据：

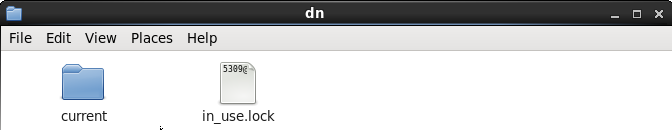


图：../dn/../finalized目录下的blk数据



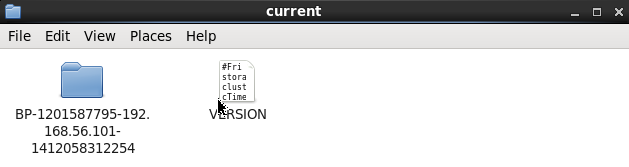
图：../dn2/../finalized目录下的blk数据

下面介绍每个目录的结构，以../dn为例，目录如下：

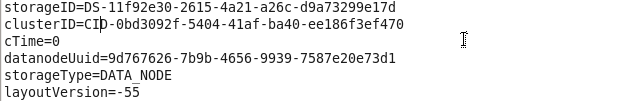


1. current：数据节点管理的最重要目录，保存着已经写入HDFS文件系统的数据块，就是写操作已经结束的已提交的数据块。该目录包含一些系统工作时需要的文件。

current下的目录，如下图所示：

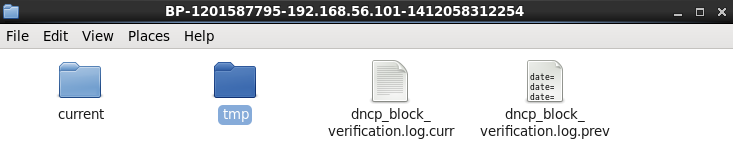


* VERSION文件，保存了以下信息：

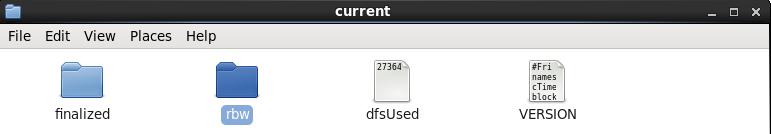


每个../dn都对应一个StorageDirectory(下面会详细介绍)，版本文件中保存了StorageID,clusterID,datanodeUuid及storageType等信息。

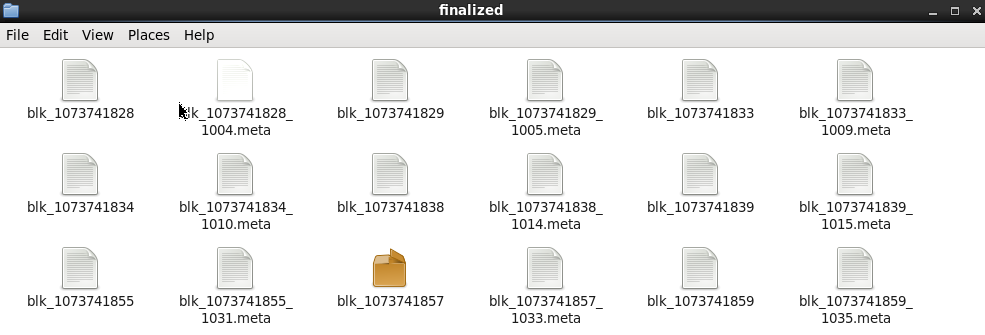
* BP-120xxx-xxx,该目录对应一个BlockPool目录，由于集群中没有采用federation，因此该类型的目录只有一个，目录结构如下图:



current下的目录结构：

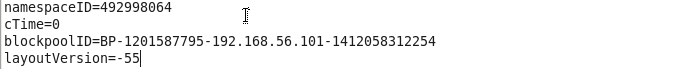


finalized目录放置了具体的block文件，如图所示：



dfsUsed：硬盘空间的使用

VERSION中的信息如下图所示：



保存了blockpoolID，namespaceID等信息。

* in\_use.lock:表明目录已经被hostx使用，实现了一种锁机制，如果停止数据节点，该文件会消失，in\_use.lock内文件内容为: id@hostx。进行数据写的操作，本地数据优先选择本地节点进行写入。但是从网络过来的数据namenode会根据各节点的使用状况选择最佳节点（将数据均衡写到各节点），进行数据的写入。
* 其他文件 在数据写入的过程中，会形成多个临时文件目录，但是不在目录中显示：
* detach:用于配合数据节点升级，供数据块分离操作保存临时工作文件
* tmp:该文件夹也保存着当前正在写的数据块，这里的写操作是由数据块复制引发的，另一个数据节点正在发送数据到数据块中

# 2、类关系图及目录对应关系

对应上面的文件目录结构分析，Datanode数据存储的核心类关系及目录对应关系如下图所示：



如上图所示：

每个DataNode节点对应一个DataStorage对象，每个节点可以配置多个本地存储路径，每个存储路径抽象成StorageDirectory。

上面的目录分析中，StorageDiretory->../dn,../dn2，在每个StorageDirectory中根据集群中Namenode federation个数组织成多个存放Block文件的目录，即对应多个BP-20xxx,BP-21068xxx目录，每个目录维护着对应的BlockPoolID的block文件（每个Namenode对应一个目录，VERSION文件保存BlockPoolID的值）。

BlockpoolSliceStorage维护着本Namenode的StorageDirectory信息，在每个StorageDirectory中构建对应的Namenode的子目录（BP-xxx）。

# 3、DataNode类图

DataNode相关的代码分成四个部分：目录管理，数据操作，DataNode进程及Datanode与Namenode通信四个模块。下面分别对这四个模块进行阐述。

## 3.1 目录管理

类图如下所示：



数据节点存储DataStorage是抽象类Storage的子类，而Storage又继承自StorageInfo。

1)StorageInfo

类中保存了Datanode的集群信息，包括namespaceId,ClusterID及layoutVersion。在../dn目录中的Version文件内的信息与StorageInfo类相关。

2)Storage/DataStorage

Storage是Hadoop中的最大结构，继承了StorageInfo。Storage可以包含多个根（参考hdfs-site.xml中的配置项dfs.data.dir），这些根通过StorageDirectory（storage内部类）来表示。

DataStorage是Storage的子类，Datanode的升级、回滚和提交过程，就是对DataStorage的doUpgrade/doRollBack/doFinzalize分析得到的。

3) BlockPoolSliceStorage

在Hadoop 2.0中的Federation模式下，集群中可以有多个Namenode，每个Namenode都对应一个BlockPoolSliceStorage。在上面的目录结构文件中，每个BlockPoolSliceStorage都对应../dn/BPxxx目录。目录中的VERSION,保存了其BlockPoolID信息。

4)执行流程(初始化)



DataStorage只是对Datanode的存储路径进行管理，并没有对存储路径下的具体数据文件进行管理。

1)在启动Datanode节点的时候，会指定若干本地存储路径，首先构造DataStorage

2)根据BlockPool信息，构造BlockPoolStorage，添加至bpStorageMap

3)分析每个存储路径当前状态，对每个存储路径构造StorageDirecotry，添加至sdDir（每个bpStorage都有一个列表）

4)对于非正常状态的存储路径进行相应的操作(StorageDirectory中有详细分析)

5)正常状态的路径，进行用于启动节点时的指定操作（备份，升级，回滚及提交等）

6)操作成功，更新每一个存储路径下的版本信息，及对应的Version文件

## 3.2 Block管理相关类

DataStorage只是对目录进行维护，并不涉及到数据块的操作，数据块的操作通过类FSDataSet来完成，下图是类图：



在目录中每个Block都对应一个BlockFile，在Datanode中对Block的操作都转化为对文件的操作。在Datanode中，DataStorage->FsDatasetImpl，StoragedDirecoty->FsVo

lumeImpl，../dn/../BPxxx->BlockPoolSlice。

1. BlockPoolSlice

该类用于表示存在的Volumne中的block pool portion，对应storageDirectory/current

/Bpid/current。在集群中的所有datanode中相同的blockID的blockPool构成了Namenode的存储空间。

核心的方式是addBlock(Block b,File f):根据block创建blockFile及metaFile。其他的方式如createRBWfile(block)，createTmpFile(block)等，检查文件状态及空间使用情况。

1. FsVolumeImpl

这个类对应一个StorageDirectory，File currentDir:/storageDirectory/current。其中成员变量bpSlices:Map<String,blockPoolSlice>。其中除了一些set/get方法外，核心的方法如addBlock(bpid,block,File)：调用方法，getBlockpoolSlice(bpid).addBlock(block,file)。

在类中会有两个线程，ThreadPoolExecutor及Executorservice，接收命令及执行维护操作。

3) FsDatasetImpl

一个FsDataset对应一个DataStorage，由于在Datanode中可以对应定义多个data.dir，因此维护了成员变量FsVolumeList。核心的方法如：append(ExtendedBlock,... locklen)，getBlockFile(block)，blockPool的对应操作。以append(ExtendedBlock,.....)为例：

* 将block至于缓存中
* 创建blkfile/metaFile
* 获取该block对应的FsVolumImpl

其他操作不再介绍，可以根据分析可知，当加入一个block时首选根据ExtendedBlock的pooIId创建对应的blkfile及metafile，然后获取其所属的FsVolumImpl(../dn)，然后写入对应的blockPoolslice中。

## 3.3 与Namenode通信管理

下图是Datanode中与Namenode通信相关的类，完成datanode的注册，更新及发送心跳等操作。



在hadoop 2.0中，提供Federation模式，因此与hadoop 1.0相比类有很大的变化。其中datanode中核心的类是BlockPoolManager，其成员变量：List<BPofferservice> offerService。

BPOfferService中成员变量：DatanodeRegistration包含了与Namenode通信有关的标识信息。BPServiceActor，每个namenode都会对应一个BPServiceActor线程，用于和namenode握手、注册、发送心跳及处理从namenode接收到的命令。