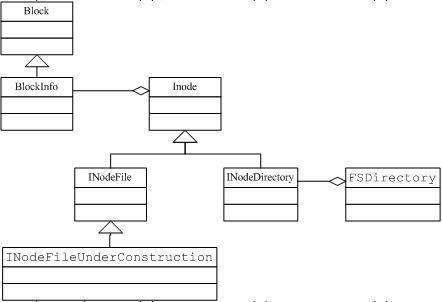
Hadoop FsDirectory

HDFS中存储数据的最小单位是Block，一个文件对应的所有Block全部按照一定的部署策略存在于DataNode上。Namenode主要存储文件元数据信息（文件目录，具体文件由Block组成），该部分主要涉及的关键对象为：INode,INodeFile,INodeDirectory,Block,

BlockInof,FSDirectory等。

Namenode主要负责存储文件目录结构以及具体文件由哪些Block组成等元数据基本信息，以上对象存储在于namenode上。主要关系如下：



1.1 Inode是抽象基类，表示一个文件属性目录结构的一个节点，包含属性为：

userName : 文件\目录所属用户名

groupName : 文件\目录所属组名

modificationTime : 上次修改时间

accessTime : 上次访问时间

fsPermission : 访问权限

1.2 InodeFile，继承INode，表示文件节点，包含属性如下：

1) BlockInfo blocks[]

该对象并不包含实际的block内容，主要属性为blocks，后续通过BlockID和时间戳信息可以找到这些Block存在哪台datanode机器上。

在FB RAID中，该成员变量为INodeStorage（抽象类），INodeStorage包含的

StorageType:{

REGULA\_STORAGE,

RAID\_STORAGE

}

BlockInfo blocks[]

该类有两个子类INodeRegularStorage和INodeRaidStorage,INodeRegularStorage可以转换成INodeRaidStorage。

2) protected short blockReplication;

//block的复制个数

 3)protected long preferredBlockSize;

//缺省block大小

InodeDirectory

继承自INode，表示文件目录节点，主要包含的属性如下：

private List<INode> children;

1.3 Block:

HDFS最小的存储单元，包含的属性如下：

 //BlockID标识

 private long blockId;

  //这个Block包含多少字节数据

  private long numBytes;

  //一个时间戳,表示Block的版本

 private long generationStamp;

 Block信息的读取就是client通过网络传递一个序列化的block对象到Datanode，DataNode在本机中读取这个Block对应的存储文本，返回给客户端。

1.4 BlockInfo:

blockInfo继承自Block，包含的主要属性如如下：

private INodeFile   inode;

  private Object[] triplets

1.5 FSDirectory

主要作用从镜像文件中读取目录结构信息，同时执行EditLog中记录的操作，用以恢复最新的内存对象。

另外的作用是操作INodeDirectory,INodeFile对象，对文件系统的目录，文件以及文件包含的Block进行操作。FSDirectory的关键属性rootDIR，整个文件系统的根目录。

二、HDFS Block

当Client用户请求某个文件时，Client首先通过RPC的方式调用Namenode上的服务模块，得到一个文件包含的所有Block列表，以及这些Block所在的DataNode信息。随后Client与相应的Datanode建立连接，发送Block请求。

下面介绍Namenode获取block部署在哪台Datanode上，以及管理这些信息，涉及的关键类为：Block,BlockInfo,DataNodeID,DataNodeInfo,DataNodeDescriptor,BlockMap。

其中BlockInfo中的triplets属性是关键，负责两个功能：

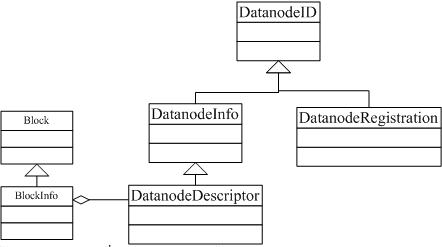
1)双向链表:

通过triplets[3\*i+1]和triplets[3\*i+2]可以得到某台datanode机器上所有的block列表。

2)

通过triplets[3\*i]可以得到这个Block其他副本的所有datanode位置，为DatanodeDescriptor类型对象。这些字段主要存储block存储在哪些Datanode上。

下面是几个重要的类：



2.1 DatanodeID

这个类完全标识一个DataNode，下面是几个关键字段：

**public** String name;      /// hostname:portNumber

**public** String storageID; //unique per cluster storageID

**protected** **int** infoPort;//the port where the infoserver is running

**public** **int** ipcPort;// the port where the ipc server is running

2.2 DatanodeInfo extends DatanodeID

该类的关键字段：

**protected** **long** capacity;

**protected** **long** dfsUsed;

**protected** **long** remaining;

**protected** **long** lastUpdate;

**protected** **int** xceiverCount;

**protected** String location = NetworkTopology.*DEFAULT\_RACK*;

另外该对象还包行两个关键字段

**private** **int** level; //which level of the tree the node resides

**private** Node parent; //its parent

这两个字段表示该datanode节点在网络拓扑架构中的相对位置信息。

2.3 DatanodeDescriptor extends DatanodeInfo

该类的实例存在于Namenode Server上，包括三个重要的内部类：

BlockIterator,BlockTargetpair,BlockQueue

BlockIterator：这个类从命名中就可以了解到，他主要的作用是对这个DataNode上所有的Block信息进行遍历，可想而知整个遍历过程借助的就是DatanodeDescriptor的private volatile BlockInfo blockList = null 这个字段, BlockInfo中的triplets[3\*i+1] 和 triplets[3\*i+2]。

遍历的流程如下：首先设置迭代器中current对象为DatanodeDescriptor 中的 blockList。利用current.findDatanode(this)方法得到DatanodeDescriptor在blockInfo中 triplets数组的索引位置index，最后利用triplets[index\*3+2]得到这个datanode的下一个BlockInfo,并将这个BlockInfo保存为current。

BlockTargetPair：这个类主要保存一个Block和一组DatanodeDescriptor信息。主要为了方便其他模块儿在这组datanode上进行对该Block的操作。

BlockQueue：这个类实质就是一个BlockTargetPair对象的队列。

关于这个类的详细应用我们后续进行讨论，本次主要目的就是了解block如何与datanode节点建立对应关系。

2.4 BlocksMap

这个类简单但是贯穿整个HDFS架构，这个类主要是对BlockInfo操作的封装。BlockMap有两个类，BlockInfo和NodeInterator。这两个类的主要作用是对某个具体block所有的Datanode节点信息进行遍历，主要实现就是通过BlockInfo中的triplets[3\*i]。

1) BlockInfo addINode(Block b,INodeFile inode)

用户对新建的文件进行写操作，Namenode新创建一个block来标识用户写入的数据（，对应于Datanode上的一个具体block文件），一个block写满时(64M)，会新建一个block标识后续用户写入的数据，这些新建的block需要添加到INodeFile中的BlockInfo数组。

loadFSImage,同样进行这个函数的调用，将文件系统的元数据信息从文件镜像恢复到内存镜像。

2)BlockInfo checkBlockInfo

建立block到blockinfo的映射关系，通过网络进行RPC调用时一些参数是Block类型。datanode启动时上报所有的block给namenode。因为blockinfo这个对象只是存在于Namenode server上。

3) getstoredBlock

返回一个block对象，返回namenode上与之对应的blockinfo对象

4) Iterator<DatanodeDescriptor> nodeIterator(Block b)

返回上述提到的一个DatanodeDescriptor迭代器，遍历存储同一个block的所有datanode信息。这个函数主要应用于用户通过client对HDFS的文件进行读取的过程，Namenode会返回一些列包含block以及datanodeinfo[]对象。其中datanodeinfo数组就是通过迭代器获取所有的datanode列表。

5)removeBlock

应用于namesystm.commitBlocksSynchronization函数中，调用这个函数的主要原因是系统主动调用recoverBlock方法

6)removeINode(Block b)

用户创建文件后，准备队文件进行写操作，首选需要在Namenode上注册新的block，并添加到INodeFile对象的block数组中，然后与多个datanode尝试创建写block的管道，如果创建失败，调用该方法，删除第一步注册的block。同时该方法，也应用于用户删除文件操作。

7) removeNode(block b ,DatanodeDescriptor node)

调用的主要原因是某一个Datanode节点失效，或者某个Datanode节点的某个具体block失效。