ChooseTarget

HDFS在文件写入的时候，要选择写入的datanode。一个Datanode会有多个DatanodeStorageInfo，因此系统在选择Datanode的时候也会确定具体存入哪个DatanodeStorageInfo。ChooseTarget()，根据参数numOfReplicas及副本大小选择datanode

参数如下：

*1) srcPath，要处理的文件，文件类型包括no\_raid,source,parity,处理方式不同*

*2)numOfReplicas，需要的副本数目*

*3)Writer，用于写数目的节点，可以是选择数据节点中的一个或者节点以外的datanode*

*4)chosenNodes，已经选择用于targets的datanode*

*5)returnChosenNodes，如果为true,则返回chosenNodes*

*6)excludeNodes，不应该作为target的datanode*

*7)blockSize，块大小*

*返回值：*

*DatanodeStorageInfo[]，作为目标节点的datanodeDescriptor数组*

# 1.集群机架及主机

Hadoop系统，在系统内部会建立一套服务器和机架的位置拓扑图，并且能识别系统节点的拓扑位置。使用位置拓扑，来控制副本放置和作业本地化。在通常情况下， hadoop系统是不能自动感知集群的架构，通过配置core-site.xml中的net.topology.script.file.

name的参数来初始化的系统的集群状况，这个配置项value是一个可执行脚本。hadoop系统是不能自动感知集群的架构。

例如，配置如下：

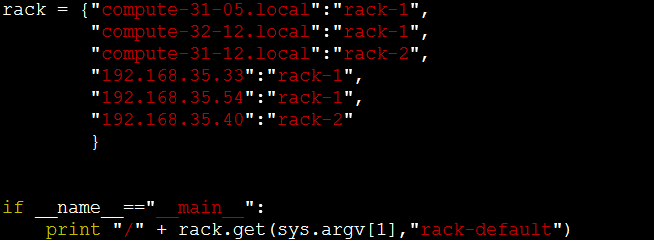
*<property>*

*<name>net.topology.script.file.name</name>*

*<value>/home/raid/hadoop/etc/hadoop/rack.py</value>*

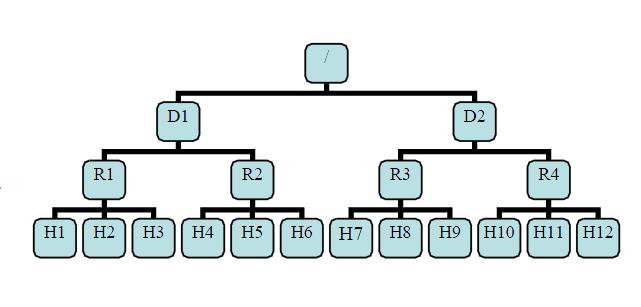
*</property>*

rack.py，内容如下图：



namenode启动后，会根据这个文件配置NetworkTopology。

下图是网络拓扑案例：



Dx和Rx是交换机，最底层是datanode。则H1的rackId是/D1/R1/H1，H1的parent是R1，R1的parent是D1。有些这些rackId的信息，就可以计算任意两台datanode之间的距离:

*distance(/D1/R1/H1,/D1/R1/H1)=0 相同的datanode*

*distance(/D1/R1/H1,/D1/R1/H2)=2 同一rack下的不同datanode*

*distance(/D1/R1/H1,/D1/R1/H4)=4 同一IDC下的不同datanode*

*distance(/D1/R1/H1,/D2/R3/H7)=6 不同IDC下的datanode*

副本放置策略：

HDFS副本放置的出发点是最小化写代价，最大化数据可靠性及可用性，提高数据的读带宽。放置策略：

1) 创建一个新的block,第一个副本放置在本地节点

2) 第二个副本放置在不同的机架上

3) 第三个副本放置在本地机架的不同节点中

4) 其他副本放置在随机节点上，但是要保证同一个节点超过1个副本，同一机架不能超过两个副本（副本号要小于机架号的两倍）

当重新创建副本，假设系统中只有一个副本，将第二个副本放在不同的机架上。当系统有两个副本，副本在同一机架上，将第一个副本放在不同的机架上。否则，将副本放在和第一个副本不同的节点上。当系统中可用的副本超过两个，其他的副本随机放置。

当所有的target node选择好，则将node组织成一个pipeline，以与第一个副本的距离排序，数据按这个属性进行传输。

读数据时，namenode首先判断这个client是否在这个集群中，如果是则与client reader的距离来返回最近的block locations。

# 2、Networktopology

该类定义了HDFS的网络拓扑，类图如下：



Hadoop实现了一个树状的拓扑结构抽象，其中Node接口，定义了网络节点的一些方法，NodeBase是Node的一个实现，提供叶子节点的一些方法，而InnerNode实现了树的内部节点，如果考虑一个网络部署的话，叶子节点是服务器，而InnodeNode是所在的机架或者交换机、路由器。Node提供网络位置信息（采用类似文件树的方式），节点所在树的深度方法。InnerNode的children保存了所有的子节点，这样就可以构造一个拓扑树，叶子是服务器或者机架，内部是机架或者路由，InnerNode提供区分的方法。

getDistance(node,node)：获取两个节点的距离，isOnSameRack判断两个节点是否处于同一个机架。

# 3、ChooseTarget流程

函数流程如下：

1) 判断numOfReplicas及clusterMap.getNumOfLeaves()，如果为0则，没有datanode可供选择，返回空

2) getMaxNodesPerRack(numOfChosen,numOfReplicas):从每个rack上获取datanode的最大数目：综合考虑racks数目、节点数目及已选择的副本，根据test类的机架:

*String[] racks = new String[]{"/rack0", "/rack0", "/rack1", "/rack1", "/rack2", "/rack2"};*

*String[] hosts = new String[]{"host0", "host1", "host2", "host3", "host4", "host5"};*

副本数目为3

则返回的结果是：副本3，每个rack的最多副本为2

1. 将chosenStorage的datanode加入到excludeNodes中，选择的时候排除这些datanode
2. 调用chooseTarget(numOfReplicas,writer,excludeNodes,....)返回结果

# chooseLocalStorage

(writer,excludeNodes,blocksize,maxNodesPerRack,results,avoidStaleNodes,StorageType)

选择本地storage的所在节点（本地rack），作为writer节点

1)如果writer为空，则调用chooseRandom，选择一个节点

chooseRandom(scope):从clusterMap中选择可用的datanode，排除excludeNodes，scope是区域，选择NodeBase.ROOT

clusterMap.chooseRandom(scope,excludeScope)，随机选择一个节点

2)不为空，则选择与writer同一机架（邻近的）数据节点

3)results.add(Writer)，将numOfReplica--

4)如果，numOfResults<=1，则调用chooseRemoteRack(1,dn0,excludedNodes,,,,)，从另外一个机架选择datanode

5)如果，numOfResults<=2,dn0与dn1同机架，从另外的机架选择，否则从本地机架选择

6)多余3个结果，则随机选择节点,通过excludeNodes，来实现选过的节点不再被选到