HDFS symlink的原理和使用

Hadoop允许创建软连接并且允许MapReduce程序直接处理这些Symlink，使用FileSystem.createSymlink直接处理这些symlink。

1. 调用顺序

DistributedFileSystem.createSymlink(Path target,Path link,boolean

createParent)

DFSClient.createSystemlink(target,link,craeatParent)

通过RPC调用Namenode.createSystemlink(target,link,createParent)

nameSystem.createSymlink(target,link,permissionStatus,createParent)->createSymlinkInt()

FSDirectory.addSymlink(link,target,dirPerms,createParent,logRegryCache):将给定的symbolick link 添加到文件系统中，并写入到edits log中

1) 通过unprotectedAddSystemlin(id,path,target,modTime,modTime,pression)方法创建INodeSymlink对象

2) symlink = new InodeSymlink(id,null,perm,mtime,atime,target)

3) 调用addINode(path,symlink)，将symlink加入到FSdirectory 文件目录树中，首先获取Path的components，Inode.getPathComponets(src)

4) addLastInode(getExistingPathINodes(components),child,true)，将symlink加入到该文件INodeFile中的Inode列表的最后，父目录的List<INode>中。

2 INodeSymlink

INodeSymlink继承类INodeWithAdditonalFields，表示symbolic link。其成员变量包括：

byte[] symlink : target URI

其他的成员变量继承其父类，包括INode name，权限、访问时间等参数

3、Symlink的使用

Hadoop的分布式缓存机制（DistributedCache）使得一个Job的所有map或reduce可以访问同一份文件。在任务提交后，hadoop将由-file和-archive选项指定的文件复制到HDFS上，任务运行前，TaskTracker从JobTracker文件系统复制文件到本地磁盘作为缓存，这样任务就可以访问这些文件。

在使用DistributedCache时，对于本地化文件的访问，通常使用Symblolic link来访问。通过URI hdfs://namenode/test/input/input/files #myfile指定文件在当前目录中被符号链接为myfile。这样job里面可直接通过myfile来访问文件，而不用关心该文件在本地的具体路径。

例如，String uriWithLink = "/tet/file/file1"+"# god.txt"，指定的文件在当前目录中被符号链接为god.txt。

通过DistributedCache.addCacheFile(new URI(uriWithlink))，分布式环境将该目录的文件加载到内存中后，通过god.txt即可访问文件。当程序中有很多小文件时，使用Symbolic link时很方便。

用户可以通过DistributedCache.createSymlink(conf)让DistributedCache在当前目录中创建到缓存文件的符号链接，或者通过设置文件属性mapred.create.symlink=yes。分布式缓存会截取片段作为链接的名字。

（在分布式缓存中，会有一个名为god.txt的链接指向源文件）

4、实现

DistributedCache.createSymlink(conf)：其中Configuration实例中的mapred.cache.file保存了在分布式缓存中的文件名，格式为fileName#symlinkName。createSymlink(conf)根据cache.file的内容创建文件对应的symlink。