HDFS Permission

HDFS分布式分析系统实现了一个和POSIX系统类似的文件和目录的权限模型，每个文件和目录有一个所有者和用户组，文件或目录对其所有者、同组的其他用户以及其他用户分别有着不同的权限。但是HDFS并不提供创建用户身份、创建组或处理用户凭证等功能。HDFS分布式文件系统中使用的用户及用户组，是Linux系统的用户身份等信息。在HDFS中超级用户即启动Namenode进程的用户，可以通过所有的权限检查，namenode开始运行时，进程自动判断谁是超级用户。在hdfs-site.xml中通过dfs.permissions = true，设置打开权限系统。

下面是进行文件权限控制的几个命令，通过Unix command来设置权限：

chmod -R mod file

chgrp [-R] group file

chown [-R] [owner]:[group] file

一、相关类

1.1、FsActions

这个类是文件系统的actions，例如读、写等，文件执行的模式和Linux操作系统的表示方式相同，例如：WRITE\_EXECUTE("-wx")。

在HDFS文件系统中，使用这个模式对文件进行可读写的控制

1.2、FsPermissions

这个类表示文件或目录的权限，成员变量如下：

FsAction useraction:用户的可执行操作

FsAction groupaction:用户组的可执行操作

FsAction otherAction:其他用户或者用户组的可执行操作

1.3 PermissionStatus

该类保存权限相关信息，成员变量：

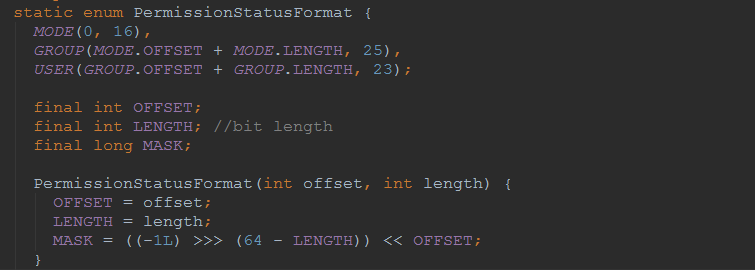
String username; //文

String groupname;

FsPermission permission;

1.4.INodeWithAdditionalFields

在类中定义了权限相关的内部枚举变量类，PermissionStatusFormat：



这个枚举类定义了文件的操作权限，通过long MASK来进行记录文件的操作模式，对文件权限的操作都是通过MASK来实现。

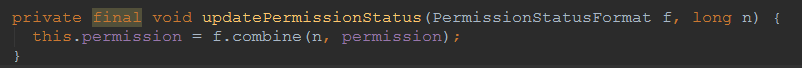
二、文件权限相关操作

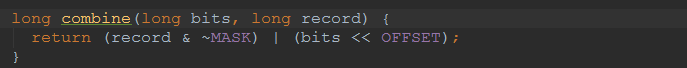
在DistributedFileSystem中与文件权限相关的核心方法是：

setPermission(Path p,FsPermission permission)

通过源码trace，DFS->DFSClient->Namenode->FSNameSystem->....->INode.setP

ermission(FsPermission)来完成操作，操作源码为：





2.1 DFS.mkdir(Path f,FsPermission permission)

在创建目录的过程中涉及文件权限的管理，创建INodeDirectory的同时赋予权限，代码trace:

DsitributedFileSystem.mkdir(fs,permission)调用Namenode的mkdir方法，NamenodeRPCServer.mkdir(src,fspermission,createparent):

在方法中，根据需要生成PermissionStatus，username<-getRemoteUser()->Nameno

de.getRemoteUser()，获取namenode的当前用户，用户组为空

最终调用FSDirectory.mkdir(src,permissions,....)，根据src创建INodeDirectory



然后通过addChild()加入文件目录树

2.2 FsShell.copyFromLocal(src,dst)

将本地文件复制到HDFS中，实质上是在HDFS中创建文件

1) dstFs.mkdir()在目标文件系统（HDFS）中创建目录

mkdirs(path,FsPermission.getDirDefault()):默认的文件目录权限是00777,

2) 文件输入流，dstFs.create(dst)

FSDataOutputStream create(.....):调用FsPermission.getFileDefault()，默认文件权限是00666

3) 调用IOUtils.copyBytes(in,out,conf)完成文件的复制

2.3 文件权限的改变

下面分析权限控制的源码，通过chmod操作过程来分析命令执行相关的代码：

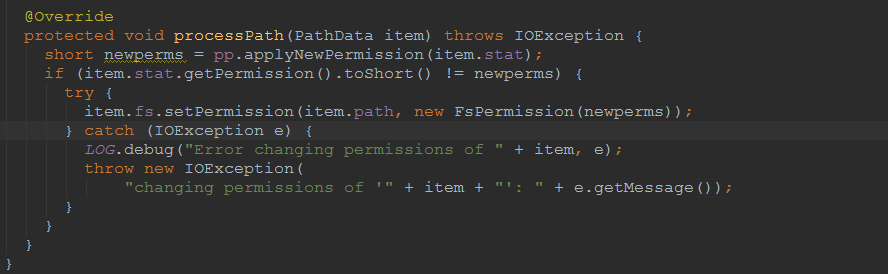
FileUtil.chmod(String filename,String perm,boolean recursive):

1)首先通过Shell.getSetPermissionCommand(perm,recursive)获取permission执行命令:String[] {chod,-R,perm}

2)通过ShellCommandExecutor shExec来完成操作

3)shExec.execute()来执行操作

其chmod命令的执行，通过类Chmod.processPath(PathData item)来完成，代码如下图所示：



命令的执行最终的实现是FileSystem.setPermission(item.path,newPermission)来实现。

类Chmod继承了FsShellPermission，FsCommands，细节不再分析。