HDFS权限管理用户指南

目录

1. 概述

Hadoop分布式文件系统实现了一个和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（owner）和一个组（group）。文件或目录对其所有者、同组的其他用户以及所有其他用户分别有着不同的权限。对文件而言，当读取这个文件时需要有r权限，当写入或者追加到文件时需要有w权限。对目录而言，当列出目录内容时需要具有r权限，当新建或删除子文件或子目录时需要有w权限，当访问目录的子节点时需要有x权限。不同于POSIX模型，HDFS权限模型中的文件没有sticky，setuid或setgid位，因为这里没有可执行文件的概念。为了简单起见，这里也没有目录的sticky，setuid或setgid位。总的来说，文件或目录的权限就是它的模式（mode）。HDFS采用了Unix表示和显示模式的习惯，包括使用八进制数来表示权限。当新建一个文件或目录，它的所有者即客户进程的用户，它的所属组是父目录的组（BSD的规定）。

每个访问HDFS的用户进程的标识分为两个部分，分别是用户名和组名列表。每次用户进程访问一个文件或目录foo，HDFS都要对其进行权限检查，

- 如果用户即foo的所有者，则检查所有者的访问权限；
- 如果foo关联的组在组名列表中出现，则检查组用户的访问权限；
- 否则检查foo其他用户的访问权限。

如果权限检查失败，则客户的操作会失败。

2. 用户身份

在这个版本的Hadoop中，客户端用户身份是通过宿主操作系统给出。对类Unix系统来说，

- 用户名等于`whoami`；
- 组列表等于`bash -c groups`。

将来会增加其他的方式来确定用户身份（比如Kerberos、LDAP等）。期待用上文中提到的第一种方式来防止一个用户假冒另一个用户是不现实的。这种用户身份识别机制结合权限模型允许一个协作团体以一种有组织的形式共享文件系统中的资源。

不管怎样，用户身份机制对HDFS本身来说只是外部特性。HDFS并不提供创建用户身份、创建组或处理用户凭证等功能。

3．理解系统的实现

每次文件或目录操作都传递完整的路径名给name node，每一个操作都会对此路径做权限检查。客户框架会隐式地将用户身份和与name node的连接关联起来，从而减少改变现有客户端API的需求。经常会有这种情况，当对一个文件的某一操作成功后，之后同样的操作却会失败，这是因为文件或路径上的某些目录已经不复存在了。比如，客户端首先开始读一个文件，它向name node发出一个请求以获取文件第一个数据块的位置。但接下去的获取其他数据块的第二个请求可能会失败。另一方面，删除一个文件并不会撤销客户端已经获得的对文件数据块的访问权限。而权限管理能使得客户端对一个文件的访问许可在两次请求之间被收回。重复一下，权限的改变并不会撤销当前客户端对文件数据块的访问许可。

map-reduce框架通过传递字符串来指派用户身份，没有做其他特别的安全方面的考虑。文件或目录的所有者和组属性是以字符串的形式保存，而不是像传统的Unix方式转换为用户和组的数字ID。

这个发行版本的权限管理特性并不需要改变data node的任何行为。Data node上的数据块上并没有任何Hadoop所有者或权限等关联属性。

4．文件系统API变更

如果权限检查失败，所有使用一个路径参数的方法都可能抛出AccessControlException异常。

新增方法:

- public FSDataOutputStream create(Path f, FsPermission permission, boolean overwrite, int bufferSize, short replication, long blockSize, Progressable progress) throws IOException;
- public boolean mkdirs(Path f, FsPermission permission) throws IOException;
- public void setPermission(Path p, FsPermission permission) throws IOException;
- public void setOwner(Path p, String username, String groupname) throws IOException;
- public FileStatus getFileStatus(Path f) throws IOException; 也会返回路径关联的所有者、组和模式属性。

新建文件或目录的模式受配置参数umask的约束。当使用之前的 create(path, …) 方法（没有指定权限参数）时，新文件的模式是666&^umask。当使用新的 create(path, permission, …) 方法（指定了权限参数P）时，新文件的模式是P&^umask&666。当使用先前的 mkdirs(path) 方法（没有指定 权限参数）新建一个目录时，新目录的模式是777&^umask。当使用新的 mkdirs(path, permission) 方法（指定了权限参数P）新建一个目录时，新目录的模式是P&^umask&777。

5．She11命令变更

新增操作:

chmod [-R] ***mode file …***
只有文件的所有者或者超级用户才有权限改变文件模式。
chgrp [-R] ***group file …***
使用chgrp命令的用户必须属于特定的组且是文件的所有者，或者用户是超级用户。
chown [-R] ***[owner][:[group]] file …***
文件的所有者的只能被超级用户更改。
1s ***file …***
1sr ***file …***
输出格式做了调整以显示所有者、组和模式。

6．超级用户

超级用户即运行name node进程的用户。宽泛的讲，如果你启动了name node，你就是超级用户。超级用户干任何事情，因为超级用户能够通过所有的权限检查。没有永久记号保留谁过去是超级用户；当name node开始运行时，进程自动判断谁现在是超级用户。HDFS的超级用户不一定非得是name node主机上的超级用户，也不需要所有的集群的超级用户都是一个。同样的，在个人工作站上运行HDFS的实验者，不需任何配置就已方便的成为了他的部署实例的超级用户。

另外，管理员可以用配置参数指定一组特定的用户，如果做了设定，这个组的成员也会是超级用户。

7．Web服务器

Web服务器的身份是一个可配置参数。Name node并没有真实用户的概念，但是Web服务器表现地就像它具有管理员选定的用户的身份（用户名和组）一样。除非这个选定的

身份是超级用户，否则会有名字空间中的一部分对Web服务器来说不可见。

8．在线升级

如果集群在0.15版本的数据集（fsimage）上启动，所有的文件和目录都有所有者O，组G，和模式M，这里 O 和 G 分别是超级用户的用户标识和组名，M是一个配置参数。

9．配置参数

dfs.permissions = true

如果是 true，则打开前文所述的权限系统。如果是 false，权限检查 就是关闭的，但是其他的行为没有改变。这个配置参数的改变并不改变文件或目录的模式、所有者和组等信息。

不管权限模式是开还是关，chmod，chgrp 和 chown 总是 会检查权限。这些命令只有在权限检查背景下才有用，所以不会有兼容性问题。这样，这就能让管理员在打开常规的权限检查之前可以可靠地设置文件的所有者和权限。

dfs.web.ugi = webuser,webgroup

Web服务器使用的用户名。如果将这个参数设置为超级用户的名称，则所有Web客户就可以看到所有的信息。如果将这个参数设置为一个不使用的用户，则Web客户就只能访问到“other”权限可访问的资源了。额外的组可以加在后面，形成一个用逗号分隔的列表。

dfs.permissions.supergroup = supergroup

超级用户的组名。

dfs.upgrade.permission = 777

升级时的初始模式。文件永不会被设置x权限。在配置文件中，可以使用十进制数51110。

dfs.umask = 022

umask参数在创建文件和目录时使用。在配置文件中，可以使用十进制数1810。