综述：

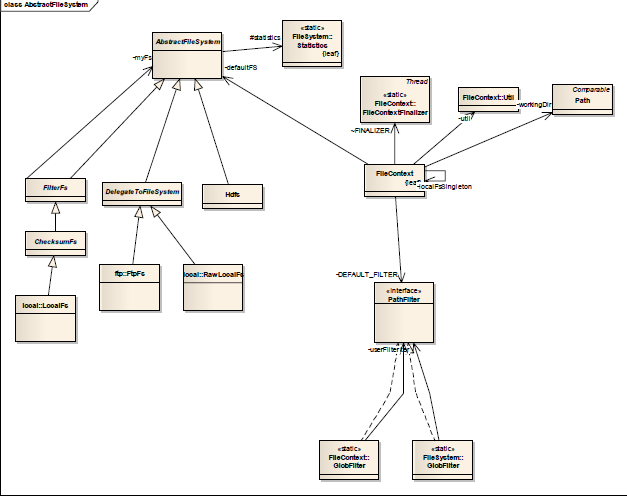
AbstractFileSystem与FileSystem对应

FilerFs与FilerFileSystem对应

Checksums与ChecksumFileSystem对应

LocalFs与LocalFileSystem对应

RawLocalFs与RawLocalFileSystem对应



PDF Page6

AbstractFileSystem提供了一个Hadoop文件系统的一个模版，一个虚拟文件系统的框架，Hadoop应用程序是不需要接触到这个类的。AbstractFileSystem 是0.21 版本新出现的API，应该是用来替代FileSystem 的。该类为 Hadoop文件系统的实现提供了一个接口。AbstractFileSystem 是一个抽象类。它与 FileSystem 有很多同名的方法。这些同名的方法完成相同的功能。比如get，create 等等。其它的方法大都是抽象方法。

FilterFs 也是一个抽象类，它同FilterFileSystem 完全一样，只是拥有一个AbstractFileSystem

类型的属性myFs。FilterFs 将其作为基本的文件系统。FilterFs 类几乎将所有重写的方法交给了其内部保存的myFs 来处理。但在交给myFs 处理之前，自己可以做一些处理，以此来实现过滤。

ChecksumFs 抽象类的实现与ChecksumFileSystem 的实现完全一样。只是ChecksumFs 使用的是ChecksumFs.ChecksumFSInputStream ， 而ChecksumFileSystem 使用的是ChecksumFileSystem.ChecksumFSInputStream。

顾名思义，DelegateToFileSystm 是一个代理类。它简单的将所有的操作交给FileSystm 类型的

属性fsImpl 来处理。目前 Hadoop 源码中只有FtpFs 和RawLocalFs 是从其派生的。

FileContext 类是0.21 版中提供的新的API，它为应用程序编写者提供了访问Hadoop 文件系统的接口，例如create open list 等方法。

1. AbstracFileSystem.java

**抽象类**

Statistics:记录文件信息

CreateFileSystem方法：根据uri和conf创建一个文件系统实例，conf可作为文件系统的configuration

Get方法：Hadoop是工厂模式的架构，get方法是最重要的方法，通过获取uri和conf可以得到需要创建AbstractFileSystem的具体类

Uri：需要创建的文件系统

Conf：传递给文件系统的实现

返回值：根据给定的uri返回相应的文件系统

其中包含有各个错误检查：

CheckPath();

checkScheme();

getUriDefaultPort();

getInitialWorkingDirectory(); 本意是得到一些文件系统需要的初始工作目录

getServerDefaults();初始化默认configuration参数，是虚拟方法，被继承时需要实现，因此符合工厂模式的开发，根据需要创建的不同的文件系统得到不同的默认参数

resolvePath();新增功能，根据内部链接符号和linux挂载点返回路径

public Path resolvePath(final Path p) throws FileNotFoundException,

UnresolvedLinkException, AccessControlException, IOException {

checkPath(p);

return getFileStatus(p).getPath(); // default impl is to return the path

}

create()方法（实现同FileContext.creat方法，但是要求参数中的Path f必须是恰当的），根据传入的参数（路径，标识等）创建或者复写某个文件，并且返回一个向该文件的一个输出流

参数：f：需要打开的文件名

Opt：创建时的一些选项

createInternal();实现同create()但是opt必须是显式声明的，是抽象方法

mkdir()抽象方法：实现时按照FileContext.mkdir()

delete()抽象方法：FileContext.delete()

open();一个抽象方法，还有一个具体实现，在具体实现中传入的bufferSize是默认参数，FileContext.open()

setReplication();抽象方法，同FileContext.setReplication();

rename():FileContext.rename

renameInternal():抽象方法，同FileContext.rename()，不能实现Overwrite。没有覆写该方法的文件系统需要实现这个方法，并且只有这个方法可以使用其他renameInternal的默认impl

supportsSymlinks();根据要实现的文件系统是否支持符号链接来返回true或者false

createSymlink();抽象方法，FileContext.createSymlink()

getLinkTarget();部分分解路径，在FSLinkResolver分解symlink时使用，**不同于**FileContext.getLinkTarget();

setPermission(); FileContext.setPermission();

setOwner():FileContext.setOwner();

setTimes():FileContext.setTimes();

getFileChecksum();FileContext.setFileChecksum();

getFileStatus();FileContext.getFileStatus();抛出UnresolvedLinkException

getFileLinkStatus();getFileBlockLocations();getFsStatus();

listStatusIterator():实现类似FileContext.listStatus();

listLocatedStatus();listStatus();

listCorruptFileBlocks():列出在给定路径下的错误文件

setVerifyChecksum();FileCOntext.setVerifyChecksum();

getCanonicalServiceName(); 返回值：一个唯一标识该文件系统的URI字符串