面向对象程序设计大作业

Apache Flink HDFS 源码分析报告

吕恒磊

2018K8009926007

一、概述

我负责阅读分析 Flink 与 Hadoop 接触的文件系统的代码（以后称之为 Flink HDFS），包括 IO，数据恢复等。此报告将分成两部分部分展开：背景介绍和源码分析。背景介绍部分对项目基本信息进行介绍，源码分析部分将围绕 Flink HDFS 重点体现出的设计模式，结合代码与 UML 图进行分析。

二、背景介绍

Apache Flink 是一款开源的流处理和批处理框架，其核心是用 Java 和 Scala 编写的分布式流数据流引擎。Flink 以数据并行和流水线的方式执行任意数据流程序。Flink 提供了高吞吐量，低延迟的流引擎，并支持事件时间处理和状态管理。Flink 没有提供自己的数据存储系统，但为 Amazon Kinesis，Apache Kafka，Alluxio，HDFS，Apache Cassandra 和 ElasticSearch 等系统提供了适配器。[15]

HDFS（Hadoop Distributed File System）是一个用 Java 为 Hadoop 框架编写的分布式、可扩展和可移植的文件系统。HDFS 并不完全兼容 POSIX 标准，目的是提高数据吞吐量的性能以及对非 POSIX 操作（如 Append）的支持。

三、功能分析与建模

1. 功能分析

Flink HDFS 的主要功能是给 Flink 系统提供 HDFS 支持，提供 HDFS 各种常用类的适配类型，包括文件系统、输入输出流、可恢复输出流、可恢复文件等。

2. 需求建模

```
需求建模
[用例名称]
    给 Flink 系统提供 HDFS 各类的适配类型
[场景]
    Who：调用者、HDFS 类、Flink 系统支持类
    Where：内存
    When：运行时
```

以上是笼统的用例需求，细化需要实现的类如下：

```
需求建模
[类]：文件系统(HadoopFileSystem)
[方法]：显示状态、返回当前路径、打开文件、删除文件、创建目录等等
[属性]：HDFS 文件系统实例、文件系统种类

[类]：输入流(HadoopDataInputStream)
[方法]：获取位置、寻找位置、跳过位置、读文件、关闭文件等等
[属性]：缓存输入流实例 FsDataInputStream

[类]：输出流(Storage)
[方法]：获取流位置、写文件流、关闭文件流、刷新文件流、同步数据等等
[属性]：缓存输出流实例 FsDataOutputStream

[类]：可恢复写(HadoopRecoverableWriter)
[方法]：打开文件流、恢复文件流等
[属性]：HDFS 文件系统实例
```

由于 Flink HDFS 只是 Flink 系统给 HDFS 提供的一个连接器，没有什么可以分析流程的，流程设计分析略过。

四、类间关系分析

如图 1 所示，Flink HDFS 的核心类是 HadoopFileSystem 类，其余诸多类以聚合方式与其关联，如 HadoopFileStatus、HadoopBlockLocation 等等。HadoopFileSystem 类的功能依赖于其他类完成，如输入输出依赖输入输出流类 HadoopDataInputStream 和 HadoopDataOutputStream、恢复文件依赖 HadoopRecoverableWrite 类。

Flink HDFS 各组件类均由 Flink 系统支持的对应接口或抽象类继承而来，体现继承思想；各组件聚合成为文件系统类，体现低耦合思想；SimpleVersionedSerializer 类提供泛型接口，将泛型类实例化为 HadoopFsRecoverable 类就实现了 HadoopRecoverableSerializer 类，体现泛型简便的优势。整个系统类间互相关联，高内聚而低耦合，符合单一职责原则和接口隔离原则。

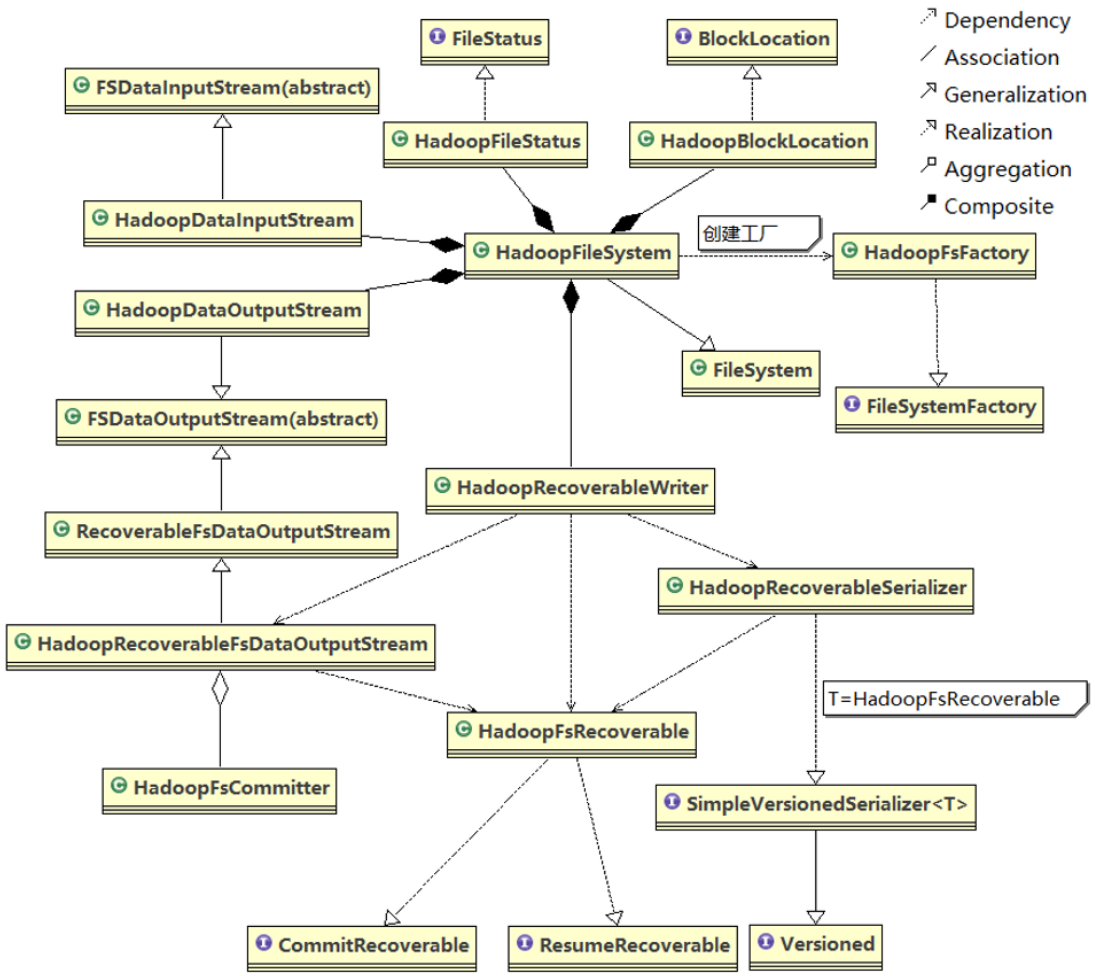

图 1 类间关系图

五、高级设计意图分析

1. 设计模式之适配器

意图：将一个类的接口转换成客户希望的另外一个接口。

主要解决：使由于接口不兼容而不能一起工作的那些类可以一起工作。

应用场景：系统需要使用现有的类，而此类的接口不符合系统的需要。

如何解决：继承或依赖。

Flink HDFS 大量使用适配器设计模式。Flink HDFS 的首要功能之一就是将 Hadoop 的 FileSystem 对象包装成一个 Flink 系统支持的文件系统 FileSystem 对象。UML 类图如图 2 所示。HadoopFileSystem 类继承自抽象类 FileSystem，后者是 Flink 系统支持的文件系统抽象类。由于 FileSystem 抽象类变量与方法较多，没有画出。

HadoopFileSystem 类缓存一个 org.apache.hadoop.fs.FileSystem 对象 fs，并在许多方法的实现中直接调用 fs 的方法。如代码 1 所示，HadoopFileSystem 类的 getFileStatus 方法直接调用缓存的 fs 的 getFilsStatus 方法，只是将路径和状态进行相应的 Hadoop 式转换。

```
@Override
public FileStatus getFileStatus(final Path f) throws IOException {
    org.apache.hadoop.fs.FileStatus status = this.fs.getFileStatus(toHa
doopPath(f));
    return new HadoopFileStatus(status);
}
```

代码 1 getFileStatus 方法

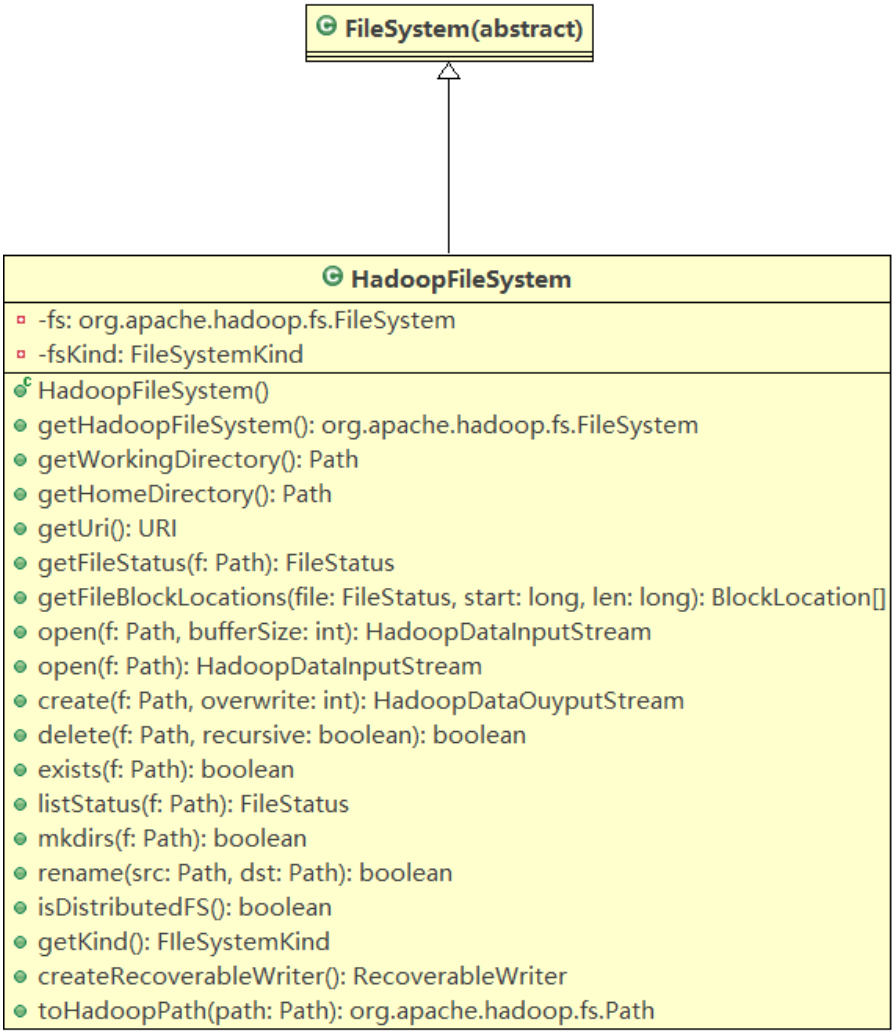

图 2 HadoopFileSystem 类图

可以看出，HadoopFileSystem 就是一个 org.apache.hadoop.fs.FileSystem 对象的包装，使其能兼容 Flink 系统自身的文件系统抽象类 FileSystem。可以将 HadoopFileSystem 类看作 Flink 对 hadoop 文件系统的适配器。

2. 设计模式之装饰器

意图：动态地给一个对象添加一些额外的职责。

主要解决：继承为类引入静态特征，随着扩展功能的增多，子类会变得膨胀。

应用场景：不想增加很多子类的情况下扩展类。

如何解决：将具体功能职责划分，同时继承装饰者模式。

Flink HDFS 的 HadoopDataInputStream 类使用了装饰器设计模式。如图 3 所示，HadoopDatInputStream 类继承自 FSDataInputStream 抽象类，后者继承自 java.io.InputStream 类。FSDataInputStream 是 Flink 系统下的输入流类，而 HadoopDataInputStream 类是一个 org.apache. hadoop.fs.FSDataInputStream 类的包装，使其后者兼容 Flink 系统。

装饰器设计思想体现在于，FSDataInputStream 抽象类在继承 InputStream 类的方法基础之上新增了 seek 和 getPos 的接口，而 HadoopFileSystem 类不仅实现了 seek 和 getPos 方法，又新增了 forceSeek，skipfully 等方法。这种模式创建了一个装饰类，用来包装原有的类，并在保持类方法签名完整性的前提下，提供了额外的功能。

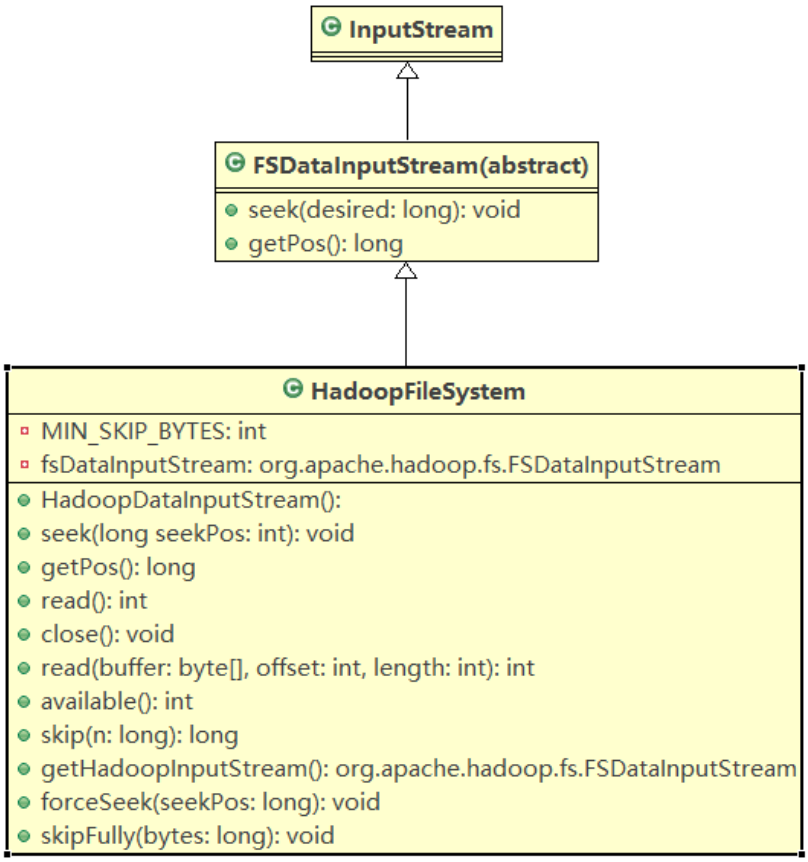

图 3 HadoopFileSystem 类图

3. 设计模式之工厂

意图：定义一个创建对象的接口，让其子类自己决定实例化哪一个工厂类。

主要解决：主要解决接口选择的问题。

应用场景：我们明确地计划不同条件下创建不同实例时。

如何解决：让其子类实现工厂接口，返回的也是一个抽象的产品。

Flink HDFS 的 HadoopFsFactory 类使用工厂设计模式。之前提到过，Flink 系统支持的文件系统抽象类 FileSystem 可以有许多实现，而 HadoopFileSystem 是其中一种。FileSystem 的子对象有许多可能，使用工厂创建它们会方便许多。如图 4 所示，HadoopFsFactory 继承了接口 FileSystemFactory，通过 configure 方法设置配置，再通过 create 方法根据文件系统 URI 创建相应的 FileSystem 对象。这样一来多种 FileSystem 子类不需要各自设计一个创建方法，统一使用工厂方法就足够了。

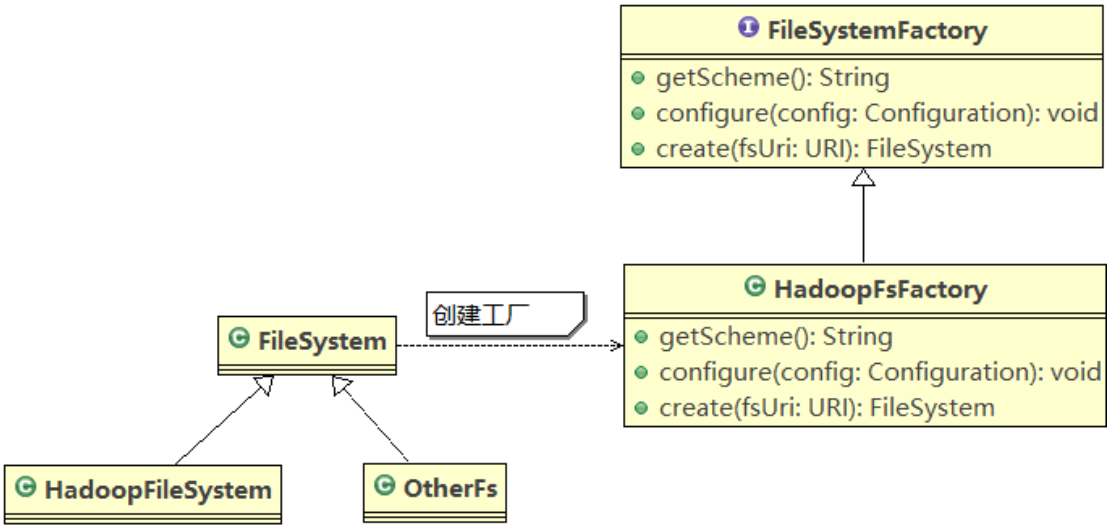

图 4 HadoopFsFactory 类图