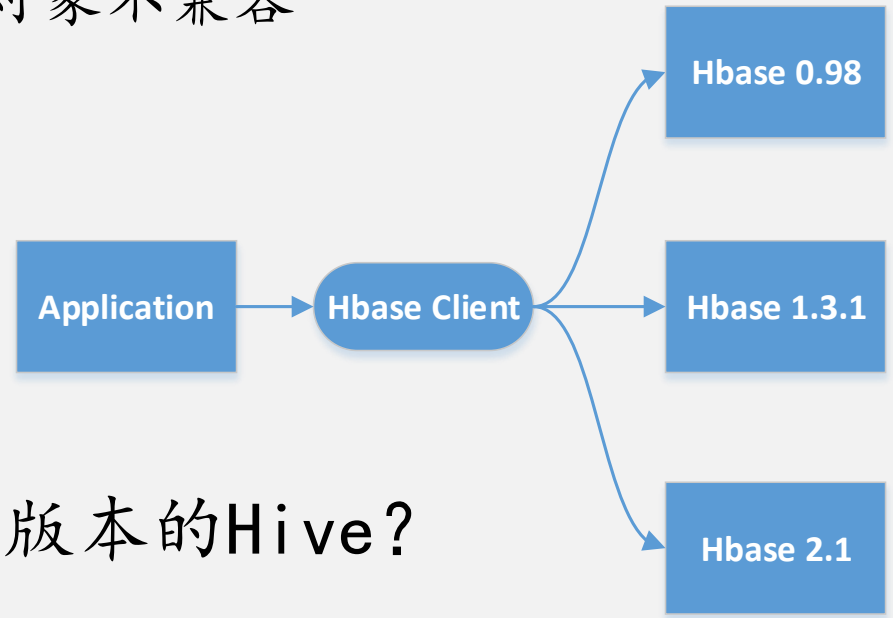
Java ClassLoader

tianjiqx

一、问题的起因

- HBase联邦计算场景
- 一个应用如何同时访问不同版本的Hbase服务？
  - Hbase client接口大版本间的不兼容
  - Hbase client接口的参数和返回值对象不兼容



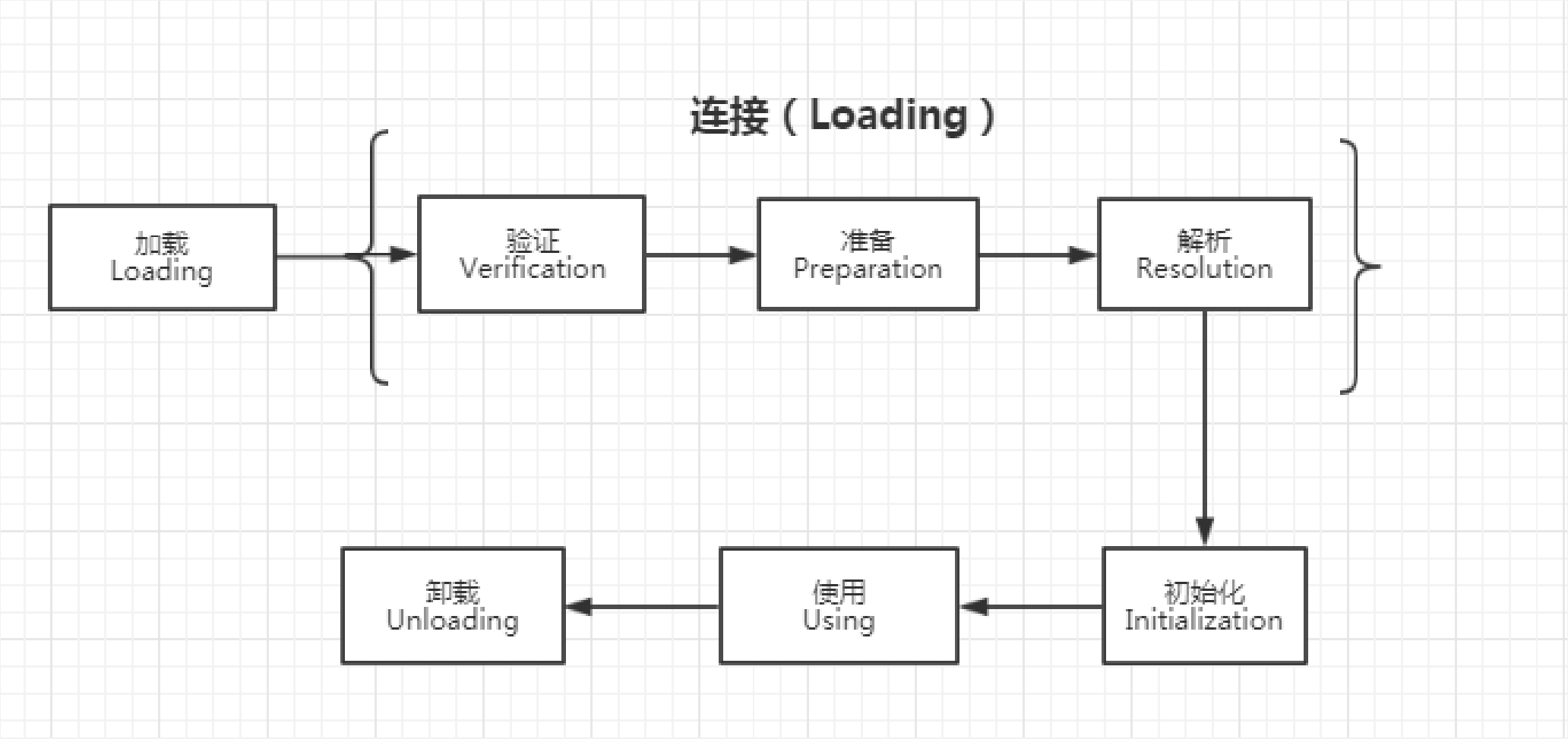
- Spark是如何支持兼容同时访问多版本的Hive？

一、问题的起因：Spark的解决方案

- Spark定义了**Hive Client接口**和**Hive相关的对象**（CatalogDatabase, CatalogTablePartition等）
- 一个接口的实现类HiveClientImpl的对象对应一个Hive版本
- HiveClientImpl类包含一个**classLoader**成员，用以加载特定Hive版本的jar包，创建获取Hive定义的client相关类对象，访问Hive的Metastore服务，完成实现。
- HiveClientImpl的各个接口的实现，直接调用Hive的client对象访问或者使用shims屏蔽接口的变动，对然后返回结果转换为自己的定义的数据类型。

- 产生了问题：Java的ClassLoader是怎样的加载类的？不同版本hive对象为何可以在一个jvm里共存?

二、CLassLoder：Class的生命周期



二、CLassLoder：Class的生命周期

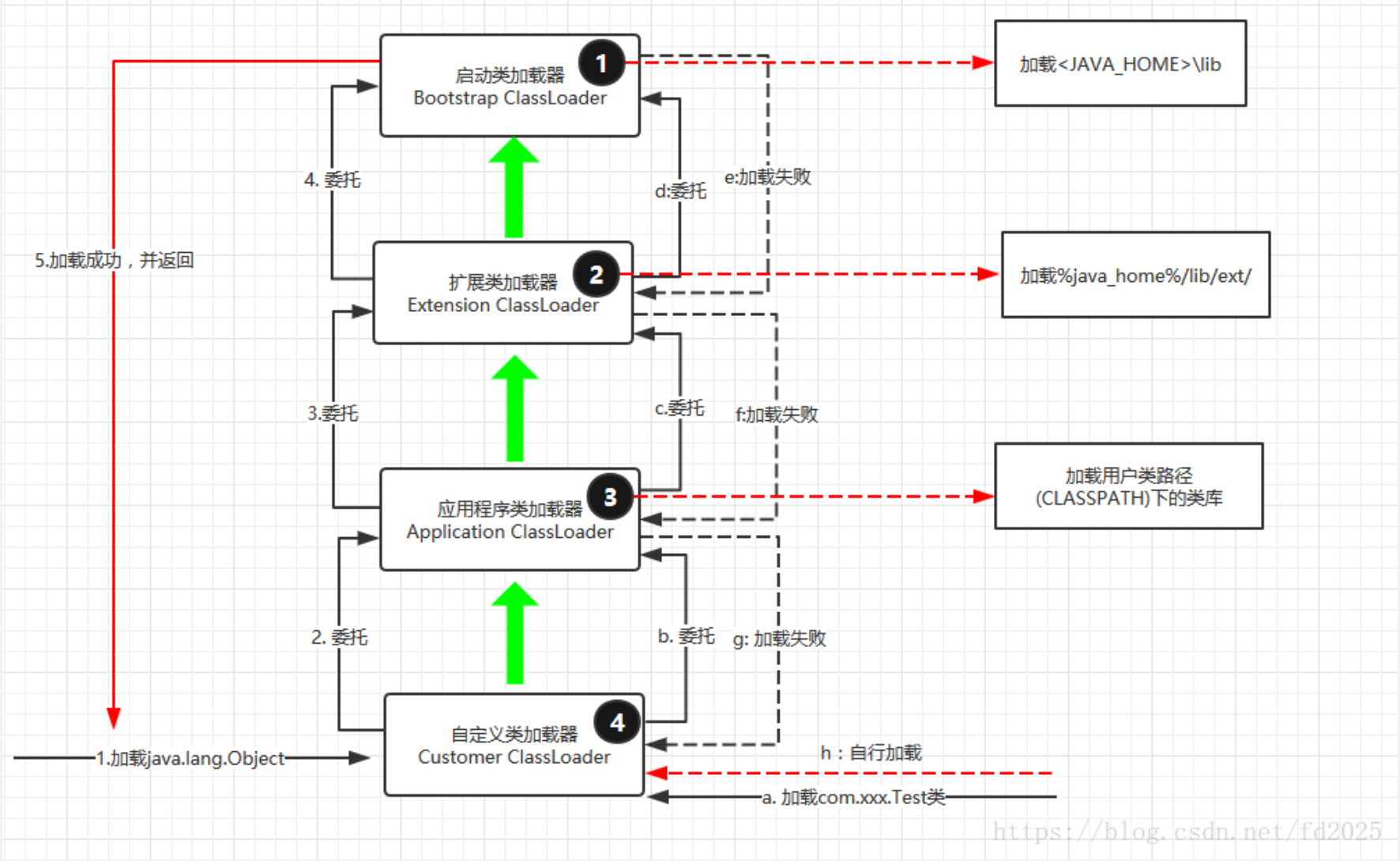
- **加载时机：**
  - 预加载（随着jvm启动加载的，如JAVA_HOME/lib/rt.jar）
  - 运行时加载(通过类的全限定名在内存中未查找到时加载)
- **加载器：**
  - 类由此组件加载。
  - 启动类加载器（BootStrap class Loader)、扩展类加载器(Extension class Loader)和应用程序类加载器(Application class Loader）默认的三类。
- **加载过程：**
  - 通过类的全限定名来获取定义此类的二进制字节流
  - 将这个类字节流代表的静态存储结构转为方法区的运行时数据结构
  - 在堆中生成一个代表此类的java.lang.Class对象，作为访问方法区这些数据结构的入口。

二、CLassLoder：Class的生命周期

- **链接**
  - **校验：** 字节码校验器会校验生成的字节码是否正确，如果校验失败，我们会得到校验错误。
  - **准备：** 分配内存并初始化默认值给所有的静态变量。
  - **解析：** 所有**符号引用**（类的全限定名）被方法区(Method Area)的**直接引用**（真实内存地址）所替代。（可能触发被引用类的加载）

- **初始化**
  - 类加载的最后阶段，所有的静态变量会被赋初始值，并且静态块将被执行。
  - **触发初始化：**
    - new关键字实例化对象；反射调用；
    - 初始化时，父类未初始化；等

二、CLassLoder：双亲委派模型



1.避免重复加载
2.保留优先级的层次关系

二、CLassLoder：ClassLoader类结构

- 主要的方法
  - defineClass:将类文件的字节数组转换成JVM内部的java.lang.Class对象
  - findClass:通过类名去加载对应的Class对象
    - 被loadClass方法调用
  - loadClass:装载指定的class
  - resolveClass：手动解析类

- 自定义ClassLoder的实现
  - 继承URLClassLoader并重写findClass方法，调用defineClass，遵从双亲委派模型
  - 继承URLClassLoader并重写loadClass方法，根据需要破坏双亲委派模型，达成加载特定版本的class。（spark）

二、CLassLoder

- Java中Class是如何标识的？
  - **二进制字节流的class定义**（来源与jar包或其他方式）和**对应的classLoader**唯一标识该类
  - 两个class比较的时候， 如isInstance，asInstance，任意一个不匹配将认为不等
  - 因此，可以在一个JVM中加载不同的版本的class，由于classLoader不同。

二、CLassLoder：三种加载类方式

- 1.New 关键字
- 2.反射Class.forName
- 3.调用classLoder的loadClass方法

- classLoader比较：
  - 1使用当前类的classLoader，this.getClass.getClassLoader
  - 2使用当前线程上下文的classLoader
    - 线程上下文的classLoader可以被设置
  - 3使用调用的classLoader
- 1静态加载，2,3动态加载
- 加载的状态：
  - Class.forName默认初始化
  - classLoader默认不进行链接，则也不做后面的初始化

二、CLassLoder

- 如何使New创建对象使用自定义的ClassLoader？
  - 通过new创建的对象，继承父对象的Classloader，在一个使用AppClassLoader加载class的对象，通过setContextClassLoader修改classloader不会生效，其方法中new的对象仍然是默认的AppClassLoader
  - Thread.currentThread().setContextClassLoader(classLoader)可修改线程的ClassLoader
  - 在修改之后，通过反射创建的对象，这个对象在方法中new的对象会是修改后的classloader

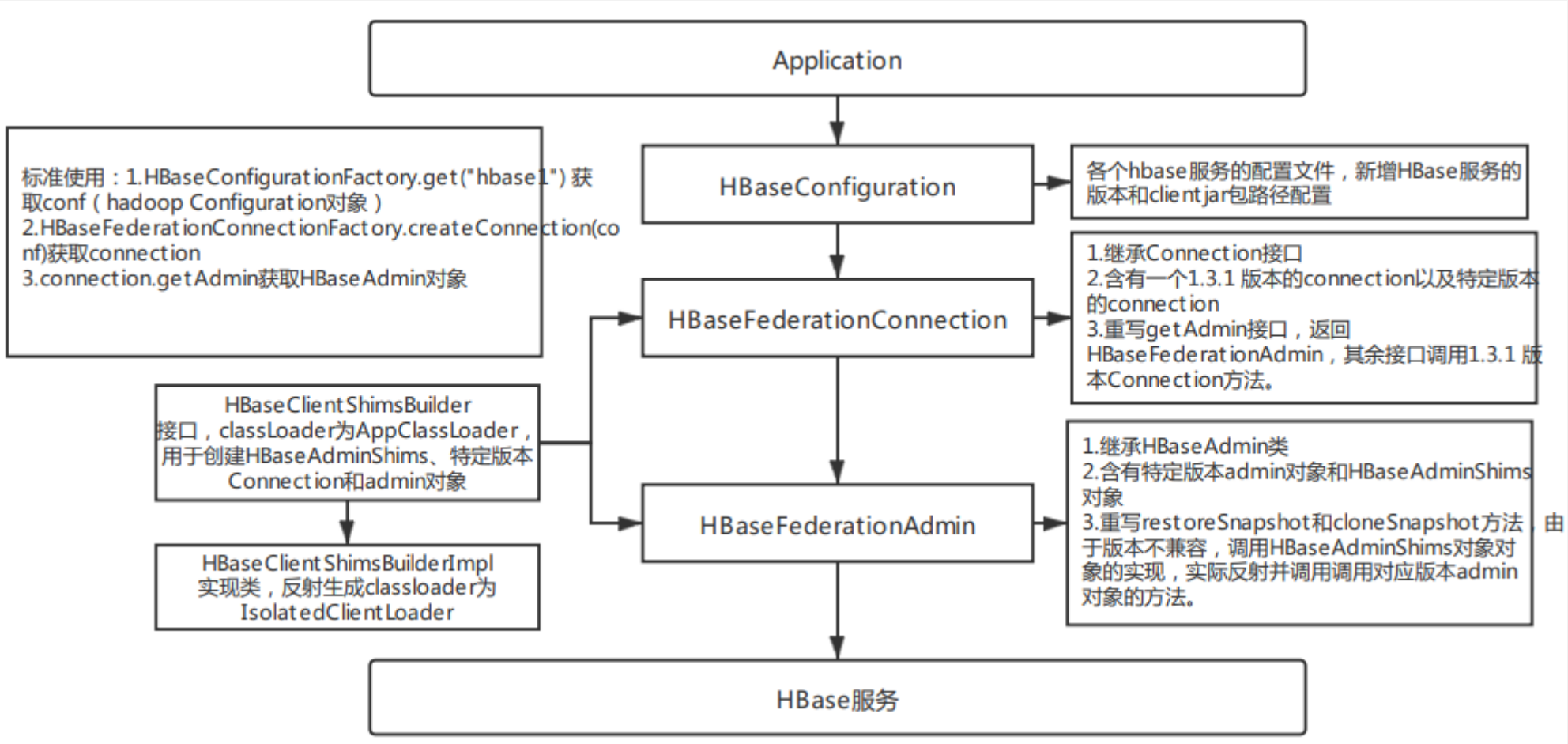
二、CLassLoder

- 接口和实现类
  - 父类的classLoader可以和子类的classLoader不同
  - 但是接口参数和返回值中引用的Class，在子类和父类必须相同。

二、CLassLoder：Spark classLoder的原理

- 接口Class的classLoder为AppClassLoader，被接口的参数和返回值的class的对象也都是AppClassLoader，可被spark其他对象所调用；
- 而创建实现类如HiveClientImpl时，修改线程上下文的classLoader，使用自定义的ClassLoader，该classLoader将加载指定版本的hive jar，在HiveClientImpl中方法new出来的对象使用的都将是指定版本的hive jar。
- 在HiveClientImpl中的接口的实现中，包装一层修改线程上下文ClassLoader的逻辑，用于处理实现的方法中包含是反射调用情况。

三、Hbase联邦计算方案



三、Hbase联邦计算方案：ClassLoder分类

- AppClassLoader
  - HBaesFederationAdmin 继承HBaseAdmin接口，重写不兼容接口
  - HBaesFederationConnectition 继承Connect，重写
  - HBaseAdminShims admin不兼容的接口
  - HBaseClientShimsBuilder 创建Shims，connection，admin的接口
  - IsoloatedClientLoader 自定义的类加载器
- IsoloatedClientLoader
  - HBaseClientShimsBuilderImpl 创建Shims，connection，admin对象的实现，对接不同版本版本
  - HBaseAdminShimesXX 不同Hbsae服务client接口的shims版本

Hbase联邦计算方案

- 同样使用ClassLoder，加载不同版本的HbaseClient的对象，接口的ClassLoader为用AppClassLoader，接口实现的使用自定义的ClassLoader。
- 但是与Spark原始方案的区别是：
  - 应用仍然感知的是一个版本Hbase client（1.3.1版本，兼容现有的应用）
  - 继承1.3.1类，通过override方式，支持其他版本不兼容的接口：不兼容参数，1.3.1的参数转为特定版本的参数对象，而不兼容的返回对象将转为1.3.1版本的结果。

参考

- Java 虚拟机 3 ： Java的类加载机制

https://crazyfzw.github.io/2018/07/05/classloader/

- JVM和ClassLoader
  https://www.cnblogs.com/Ming8006/p/11818218.html
- Spark对HiveMetastore客户端的多版本管理、兼容性探究以及栅栏实现 https://blog.csdn.net/zhanyuanlin/article/details/95898018
-