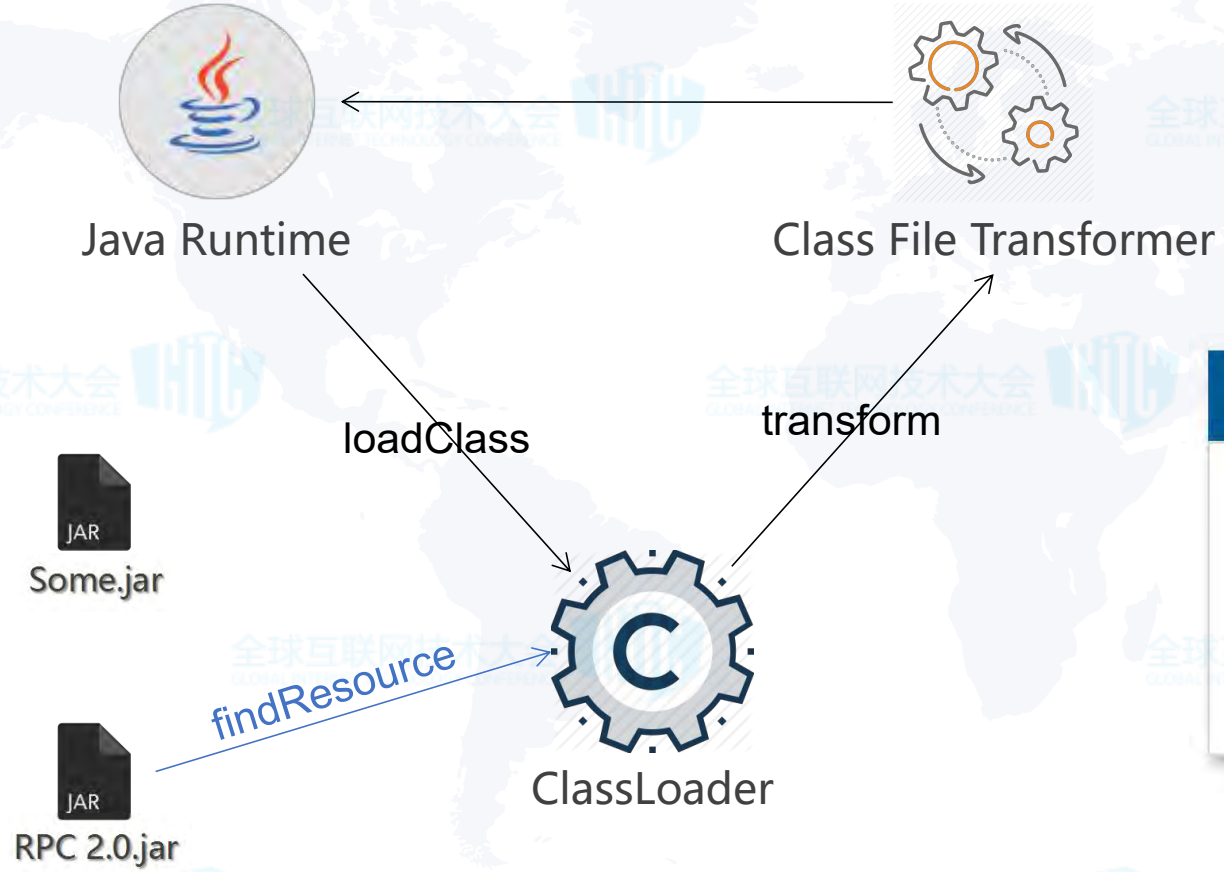
2.2 后庭别院菊花香

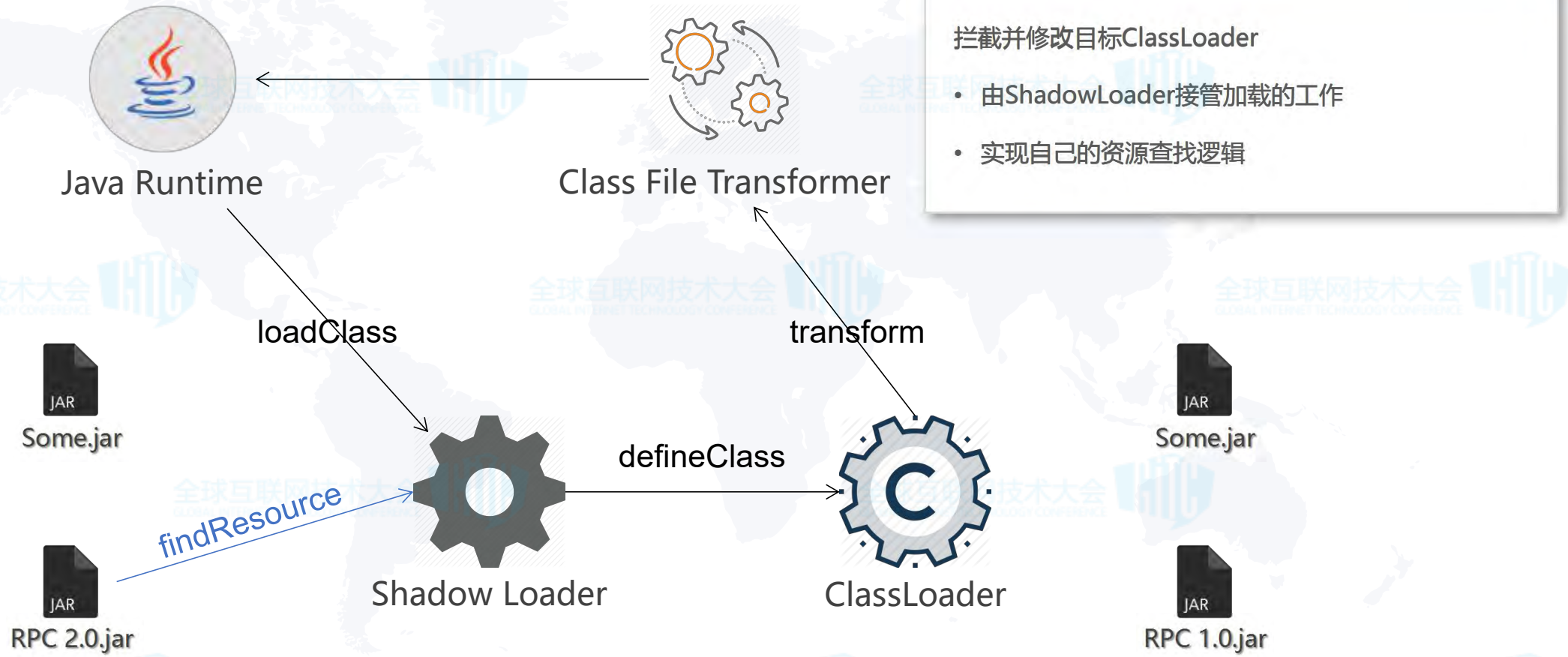


AOP

通过ClassFileTransformer进行基于字节码的方法切面

- 几乎没有性能损失
- 对任意非原生类任意方法操作

2.3 偷梁不换柱，移花能接木



HOOK

拦截并修改目标ClassLoader

- 由ShadowLoader接管加载的工作
- 实现自己的资源查找逻辑

2.4 遵守基本的设计原则

I'm Interface !

实现的还可以是不是

2.5 运行时为实现独立容器加载

```
public class LogFactory {
    private static final boolean isLogClientPresent;

    static {
        isLogClientPresent = ClassUtils.isPresent(contair
    }

    public static Logger create(String name) {
        if (isLogClientPresent) {
            Logger instance = new GangLogAdapter(name);
            instance.init();
            return instance;
        }
        return new Log4jAdapter(name);
    }
}
```

```
public static Logger create(String name) {

    Isolation container = Isolation.newContainer();
    if (isLogClientPresent) {
        Logger instance = container.single("cloud.appgov.pavepaws.impl.logger.GangLogAdapter", name);
        instance.init();
        return instance;
    }
    return container.single("cloud.appgov.pavepaws.impl.logger.Log4jAdapter", name);
}
```

伪代码

IsolationTransformer.class

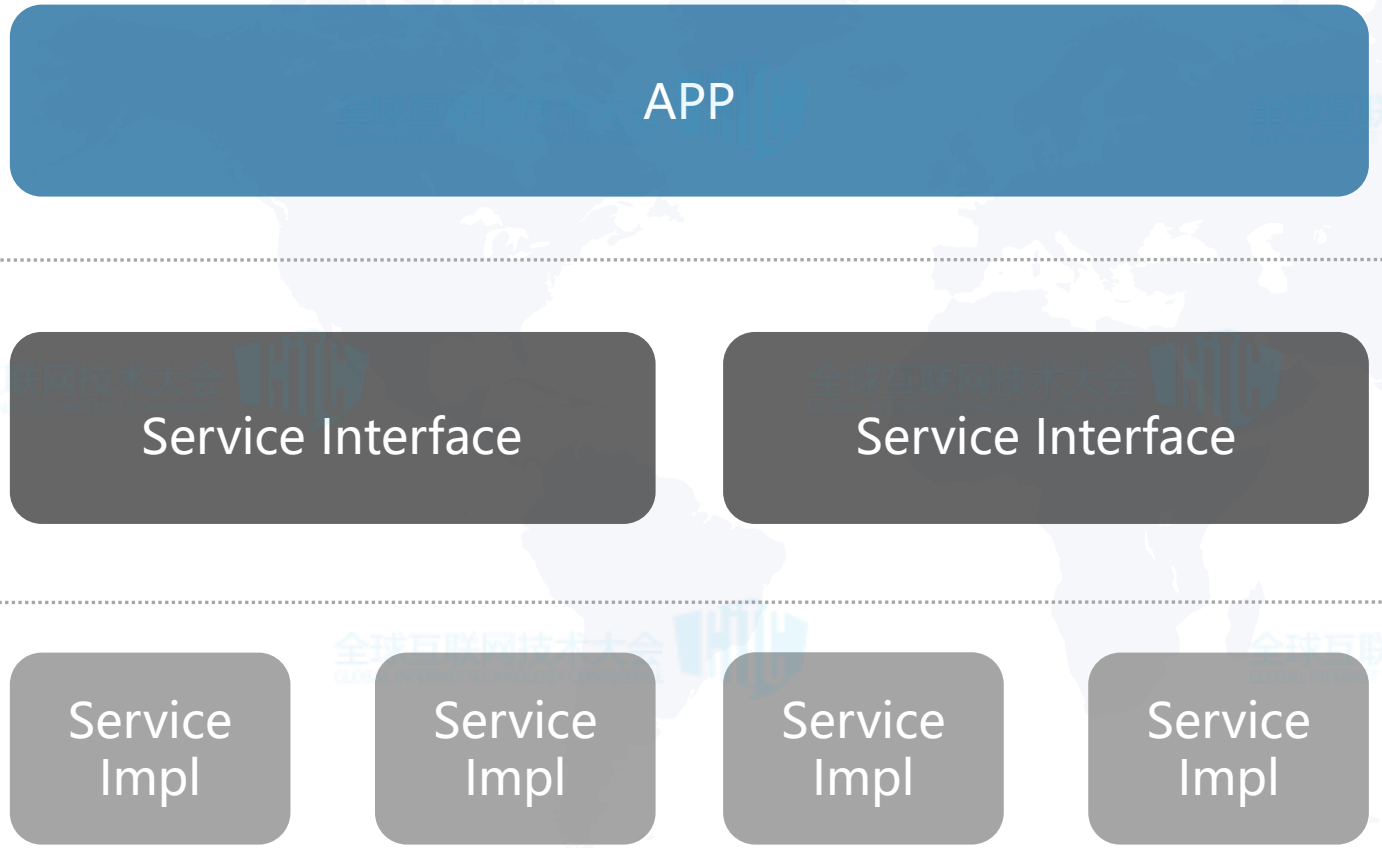
```
byte[] transform(String classname,...,byte[] bytecodes){
    ClassEditor editor = new AsmClassEditor(bytecodes);
    for(method in editor.Methods){
        scanAndReplaceInit(method);
    }
}
```

scanAndReplaceInit

在类加载时检查类是否有依赖的服务,将原具体实现类的构建转给负责“隔离”的容器。容器会在单独的ClassLoader中加载，之后创建对应的实例返回。

2.6 一山容二虎，一公和一母

2.7 进程内的微服务



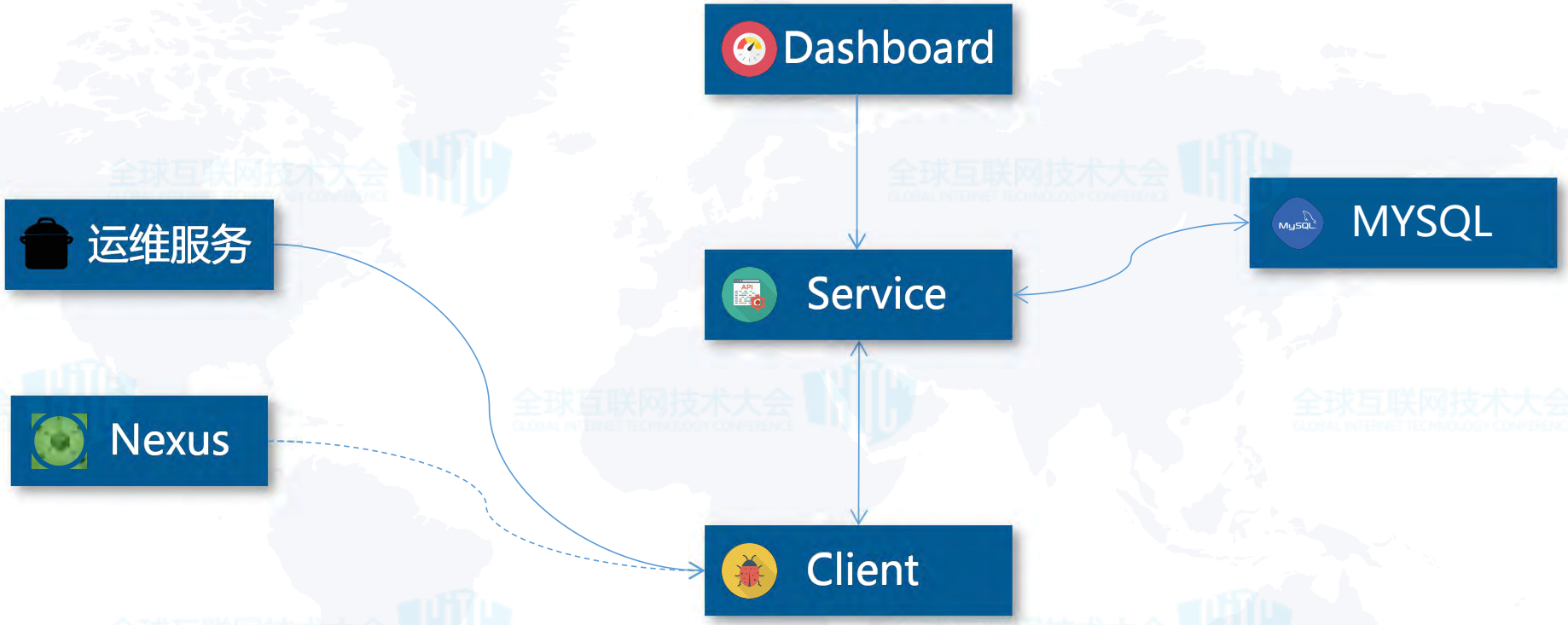
应用自身

服务接口

应用只关心和使用接口，不关心背后实现

服务实现

实现可以有多个，单例，在需要时可以切换实现



整体结构

非常简单的结构，Service定下规则，通过运维服务推送文件到目标机器。再由Client按照规则执行。

2.9 配套实施流程，形成闭合

QA

与QA制定中间件升级流程，将中间件升级发布与业务应用升级发布隔离开

研发

有了QA流程后，研发这一块主要在升级前做到沟通，协调时间避免与大项目冲突

运维

做好使用培训，用于在突发风险时，即时关停或组件降级

流控系统

做好流控为了在风险发生时，能够即时通过调整策略进行止损

3.1 大路朝天，各走一边


业务系统
SIT
UAT
独立发布
中间件升级更新不再受制于业务系统的发布周期
快速部署
任何环境都可以批量进行组件升级更新操作，而研发是无感知的
容易验证
众多的应用系统能够提供更全更多的运行测试，发现未知问题
中间件
SIT
UAT
生产