各位评委老师，大家好，我叫徐保华，是客渠OMS快件服务研发组的后端开发工程师，当前职级8级。我在2013年大学本科毕业，2022年11月加入顺丰，一直在快件服务研发组做后端开发工程师。下面开始给大家讲下我的能力举证：

第一个要讲的是开发工具能力。这个事情的背景是，SISP提供全量路由翻译的接口服务和订阅操作节点路由翻译推送服务；其中接口服务有100多套系统使用，日均调用量1亿多，路由推送服务有10多套系统接入，日均推送量10亿多；使用场景丰富，调用量大。每次需要调整路由翻译内容时，都需要修改代码，发版解决，业务响应慢。我的任务是兼容旧路由翻译规则，实现路由翻译规则可配置，快速响应业务需求。针对这个目标，我做了以下努力：

首先是梳理历史路由逻辑：涉及100多个操作码，900多个逻辑分支；重构历史代码，巧用策略模式，模板方法模式，责任链模式，解耦代码，方便扩展。

其次是路由翻译引擎设计与研发：

包括以下几个步骤，

第一步是确定方案，根据配置，采用字节码工具动态生成规则处理器，添加到处理链。

第二部是表结构设计，包括路由内容表，处理器条件表，变量翻译表，属性关系表（通过BDP同步FVP，操作运单信息）等，

第三部是字节码生成工具的选型，对比asm，bytebutty，javassist字节码生成工具，选用了生成字节码方便，直接拼接源码的javassit工具。

第四部是将动态规则生成流程化，当接收路由规则变更事件时，会经历解析配置，生成源码，编译生成class，实例化对象，合并新旧处理器，切换处理链等6个核心步骤。

第五步是添加使用率统计，异步任务统计各处理器的使用情况，后期会根据使用情况下线未使用处理器。

第三部：输出文档：代码方面：输出翻译引擎工具使用说明文档。业务方面：输出操作码逻辑文档，配置功能使用教程文档。

产生了以下结果和影响：提供可视化规则定制、审批、查询及管理。

动态代码生成工具，保证代码规范化，节约25%研发工作量。

业务响应快，提升业务体验，实现业务响应14天到分钟级的转变。

可能的提问：

规则引擎适用其他场景？ 如果当有很多规则时，抽象出策略接口，按照规则引擎的规则表配置，实现策略接口，即可动态生成生成。

第二个要讲的是开发框架，数据库与中间能力。这个事情的背景是，SISP需要降低系统资源开销成本，提升单机处理请求吞吐量和响应速度。我得任务是从系统架构、业务数据量、系统监控找出系统优化点。

首先对多线程使用场景梳理和优化：sisp项目中有部分拆分大任务进行多任务并发处理场景，使用CountDownLatch等待多个线程完成工作，但是对于countdownlatch使用不规范，导致countdownlatch一直等待，且当子任务失败，没有中断其他未执行的任务，导致系统下降。

对此封装线程组件，使用线程池的方式，控制线程数量，统一大任务拆分，支持线程interrupt方法及共享变量的组合方式中断线程，调用方仅需要传入任务列表。

其次优化数据库CPU：SISP数据库CPU周期性飚高；是由于我们应用服务定时会加载本地缓存，且大部分缓存设置了相同的缓存时间；当缓存同时过期时，数据请求量剧增，进而导致CPU飚高。

对此我重写缓存框架，提供缓存注解，方便使用。支持单线程按顺序全量加载缓存功能、懒加载功能，及时更新缓存功能。使用缓存占位符，短缓存，比如1分钟，解决缓存穿透。使用锁，或异步定时更新热点数据，解决缓存击穿。使用缓存过期时间加上随机偏差，解决缓存雪崩。

再次优化应用服务CPU和内存：SISP WEB服务CPU飚高，内存上升；结合线程dump和请求日志，定位发现是批量下载图片功能导致。批量图片下载处理流程为，分批下载图片存入内存，然后在内存中压缩图片，800M是否需要提一下？？？，导致服务器CPU和内存暴涨。

对此我将批量图片下载同步改为异步，通过线程池参数配置，控制文件下载并发数。下载时，文件先持久化到硬盘，然后通过文件流的方式压缩，最后上传至OSS供用户下载。

经过上面一些优化，

数据库CPU使用率从82%降低至16%，下降76%。

WEB服务CPU使用率从95%降低至32%，下降65%。

WEB服务堆内存从8G降至6G，下降25%。

图片介绍。。。。。