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，设定相关流程，方便业务方直接使用，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结构化，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。

最后输出翻译引擎工具使用说明文档、操作码逻辑文档、配置功能使用教程文档。

提供可视化规则定制、审批、查询及管理。引擎动态翻译工具上线后，节约50%研发工作两，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。

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运行至今已有10年多，当前8套子系统均使用公司内部陈旧框架novartar，已不再维护，导致业务提出的部分需求没法快速实现、新技术无法接入等，影响业务的快速发展及系统的运营维护。我的任务是参与SISP服务升级，升级过程中，基础组件开发，新功能实现；新服务上线后，根据监控指标对系统进行优化，提高系统性能和稳定性。

**首先disconf扩展组件研发**

SISP项目中有部分应用配置是用spring value注解注入的，每次更新配置时需要重启才能生效，影响系统稳定性。因此需要实现value支持动态配置。代码实现spring BeanPostProcessor接口，当容器启动时，收集使用value注解的属性，实现 disconf IDisconfUpdatePipeline接口，接收disconf变更文件，解析文件，替换value注解的属性值。提供disconf-extend jar包，项目引入jar包时，spring value注解就会动态生效。

**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。

对此我优化扩张缓存框架，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，可配置实时更新缓存功能。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。优化后，数据CPU使用率从82%降低至16%，下降66%。

**架构能力举证**

SISP提供KAFKA路由翻译推送服务、（运单、路由、图片、备注等）接口服务（30000tps），并有内网40万用户的核心查询WEB服务。

每次版本发布只能在深夜；业务功能无法提供试点验证；功能验证、异常版本回滚非常繁琐、复杂和耗时。

我的任务是让快件信息化服务平台KAFKA、REST服务和WEB界面支持灰度，并支持不同应用不同粒度的进行试点验证。

为此我做了以下几点？？？？

参考了业界的蓝绿发布和金丝雀发布，并结合系统自身的特点和业务场景，设计了满足KAFKA、REST、WEB三种场景的灰度架构。

加入**分流层**：接口使用API网关分流，web 服务使用SISP-WEB-SHUNT分流，kafka推送服务使用SISP-PUSH-SHUNT分流。

基于业务场景**定制灰度规则**：接口基于请求头系统编码；WEB基于用户（网点、区域等）；kafka基于消息数据本身（主题、网点、区域等）

**接口隔离原则**，让A/B 环境的KAFKA/WEB对应A/B环境应用，达到环境级别的隔离；

**模板模式**来控制整个KAFKA灰度的处理过程，整个处理过程拆分为：报文解码、获取业务数据、获取灰度配置、寻找A/B环境和数据分流五个部分；

**策略模式：**来对每个部分进行扩展，达到同一个功能有不同的实现方式；

**约定大于配置原则：**支持KAFKA生产者/消费者的动态管理，减少大量重复的配置，提升服务能力

上线发布时，对整个上线发布过程进行**上线推演**，整理出灰度上线部署方案、发布前资源检查清单、发布步骤执行清单、发布后验证检查清单和灰度发布回归方案等文档。

灰度架构成功上线后。通过控制流量，可以在白天进行版本发布，解放运维同事和研发同事

可以支持不同应用不同粒度的灰度（系统编码、网点、城市、大区、主题等）试点，方便业务验证和推广。

抽象出灰度框架和相关的可复用组件，和业务解耦，能够在多个系统复用。

通过约定和配置可以对新的业务接入，已接入20+个上游KAFKA的数据，无须重复开发，加快新业务的上线。

下一P架构图介绍

这是我们主要服务架构图，红色标注的是分流服务

**数据网关接入**

背景是SISP ADMIN服务日常有30多万文件下载，安全组要求所有文件下载都需要经过安全检测是否有敏感数据，如果检测到敏感数据，则跳转到安全组指定界面查看或申请下载，非敏感信息正常下载。

基于以上背景，设计出三种候选方案

第一种，修改代码，所有文件下载时，将文件上传给数据网关接口检测，是否存在敏感信息，这样做的风险就是改动的地方多，测试工作量大，出错的可能性大。

第二种，将SISP web服务域名cname到数据网关，所有流量都经过安全检测，这样的坏处就是给数据网关接受所有流量，服务压力很大。

第三种，通过天机网关，根据url分流，将url以download开发的请求，转发到数据网关进行检测。将host 透传，这样admin-ingress转发配置不用修改，如果天机网关或数据网关异常，只用将域名直接解析admin-ingress。这个方案不用修改代码，不用修改原有配置，且有合适的逃生方案。

接下来介绍我的短期工作目标，

第一，路由翻译引擎支持更多场景,目前已经支持FVP和OMS操作运单作为条件和变量的场景，接下准备接入pis。

第二，OISP系统准时上线，OISP是我们WEB前端重构项目，现阶段已经灰度区域了，9月中旬前完成用户反馈的问题及推向全网。

第三，提高系统的吞吐量，根据双十一压测反馈问题，提高int服务http线程数。

我需要重点提升的能力

1. 需求分析
2. 架构能力
3. 系统性能优化

我的关键举措

1. 日常的业务开发和对接中以及生产问题咨询过程中，多了解业务的使用场景，挖掘新的业务价值和需求
2. 对应用的功能进行拆分和归纳，在架构上对系统进行优化
3. 整理分析业务处理流程，观察应用监控，找出优化点

好了，我的陈述已经结束，请各位评委老提问。