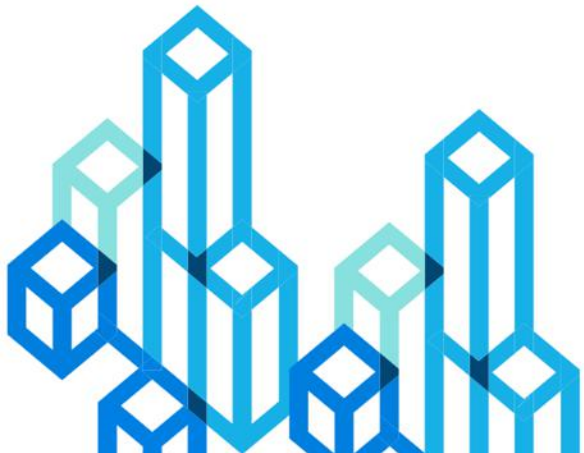
顺丰快运-基于线上压测的性能优化之旅



个人简介

10年以上IT团队管理经验

顺丰科技架构委员会成员

顺丰科技质量委员会成员

顺丰金融：增值业务线负责人

信贷、保险、结算

顺丰科技：综合物资研发部负责人

采购供应链、行政办公、综合后勤

顺丰快运科技：研发工程部负责人

零担物流系统、跨境、城配、整车、家装、3PL等

- 18年9月从0到1搭建快运科技研发团队，目前团队规模**150人**
- 18年9月-19年底，打造快运双网融通的零担物流系统底盘
- 20年-21年，顺心IT团队融合及系统升级重构，打造乐高式架构，支持新业务建设

演讲目录

- **顺丰快运介绍**
- 快运科技成长历程
- 全链路压测实施过程
- 核心系统优化案例

顺丰集团

顺丰快运，**顺丰集团旗下的综合物流业务品牌**，作为顺丰集团第二大业务板块，规模体量位居行业第一，并保持高速发展，助力客户商业前行，引领快运行业蓬勃发展。

快递业务

快运业务

冷运业务

医药业务

同城业务

国际业务

仓储服务

金融服务

产业园业务

其他

顺丰快运简介

企业概况

顺丰快运是顺丰集团旗下的**综合物流业务品牌**，包含**快运直营网、顺心加盟网**为一体的全面覆盖所有价值客户的双网模式，双网与速运大网在转运、收派上深度协同，实现成本最优。

- **快运直营网，定位中高端零担市场**，为客户提供20KG+大件包裹托运、传统零担、整车直达、同城货运搬家、仓储以及高效解决客户供应链在厂仓、仓仓、电商等具体物流解决方案。
- **顺心加盟网，专注全网中端快运市场，**通过网点加盟、分拨直营、运力众包，整合资源搭建一张覆盖全国的零担快运网络，为客户提供高品质、高客户体验、高性价比的综合物流服务。
- 20年双11直营货量突破**4.5万吨/天**（直营行业第一），加盟货量突破**2.4万吨/天**（加盟行业前五）

顺丰快运简介

业务规模

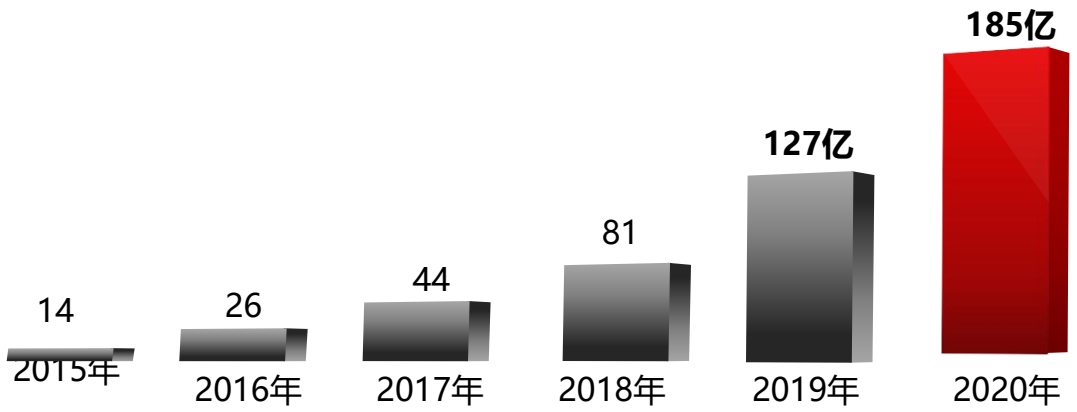
快运行业开拓者

2020年，顺丰快运业务整体实现不含税营业收入185.17亿元，同比增长46.27%。全年整体零担货量同比增长超过70%，营收规模及业务增速在零担快运主流玩家中均排名第一。

同时在顺丰控股总业务收入中的占比持续提升，2020年上升至12.06%（环比19年+0.72%）。

营收规模

- **营收规模及业务增速在零担快运主流玩家中均排名第一**



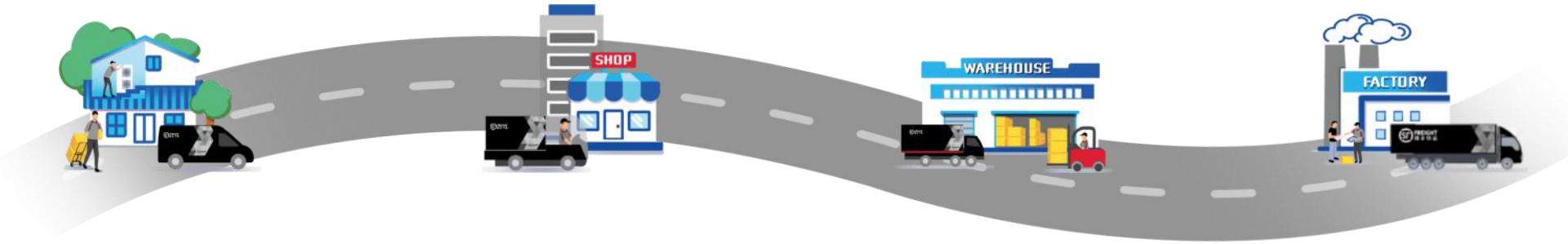
- **快运业务在顺丰控股总业务中占比持续增加**

（数据取值顺丰控股发布年度/季度报告，营收包含快运直营及顺心加盟业务，单位：亿元人民币）

顺丰快运简介

产品概况

覆盖供应链全环节物流服务，涵盖“家、店、仓、厂”全服务场景



重货包裹	标准零担	丰城专运	大票直送	整车直达
电商/店配等大包裹	**零担批量件/托盘货**	**个人/企业搬迁、定制城配**	**零担大批量业务**	**专业整车承运平台**
20-100KG	**100KG以上**	**20KG+**	**500-3000KG**	**3000KG以上**
2C电商大包裹	工业类批量件	精品搬家-日式定制	工业园区-大批量/超大件	高价值-百万货值整车
2B仓店调拨	托盘货	社区生鲜团购	专业市场-大批量	农特拼/整车-绿色助农
打造极致的入户体验	一站式解决方案	医药医疗、汽配五金	转寄大量优质专线	新能源运输、特种运输

科技赋能

全链路智慧物流解决方案

1/网络规划

场地规划、线路优化、货量预测

大数据、机器学习、运筹优化

2/智能客服

快捷下单、智能客服、智能机器人

语音识别、深度学习

3/数字收派

地址识别、AR量方、电子面单

图像识别、计算机视觉、GIS

5/天眼监控

司机驾驶行为、车辆位置轨迹、暴力分拣识别

人脸识别、地址库、GPS、AIoT

4/智慧中转

自动分拣机、无人叉车/AGV、DWS

AIoT、计算机视觉、边缘计算、大数据

演讲目录

- 顺丰快运介绍
- **快运科技技术发展**
- 全链路压测实施过程
- 核心系统优化案例

快运科技发展历程 在蜕变中成长

2018
- 8月： 顺丰科技成立快运业务科技部
- 9月：承接Dubbo+Oracle外包系统

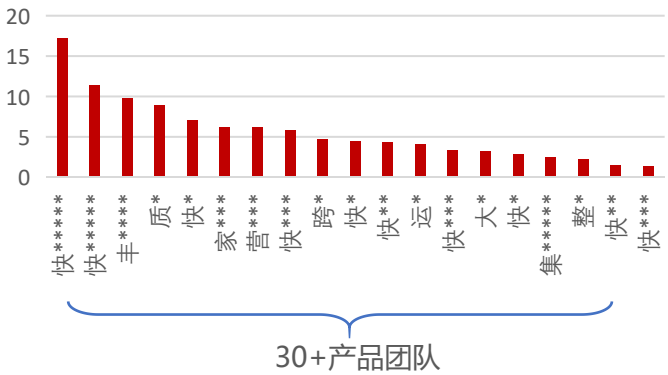
2019
- 3月：Mycat、ES
- 5月：cat、swagger+yapi
- 6月：Docker容器化非云自动化部署
- 9月：TiDB
- 12月：去Oracle

2020
- 2月：流水线自动化部署
- 4月：快运核心业务系统上云
- 7月：顺心系统全部上顺丰云，成功下线IDC机房
- 8月：快运科技发版次数突破1000次/月
- 9月：混合云（腾讯云、华为云）
- 11月：上线全链路线上压测平台，平稳保障双十一

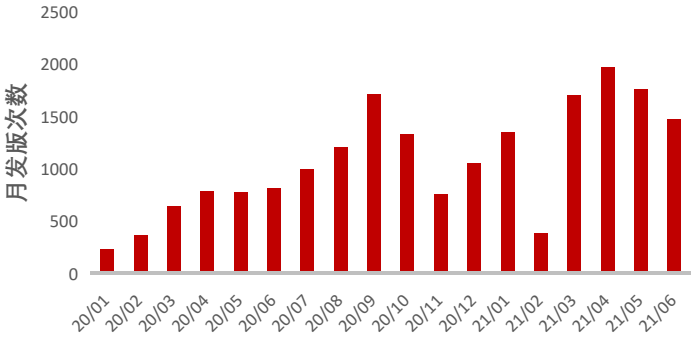
2021
- 2月：天玑网关（kong）
- 4月：领域拆分
- 6月：全链路监控
- 7月：双活
- 9月：顺心系统重构去Oracle

打造敏捷型产品团队

Pizza team



碎片化发版，快速响应需求



多网融合的系统架构

收派员:	30万	3万	3万
网　点:	30000	1500	10000
转运枢纽:	1000	90	50
线　路:	16000	3000	1700

快运控股

开放平台 → 订单 → 开单 → 接驳 → 中转 → 运输 → 中转 → 接驳 → 派送 → 结算

运单中台　　**路由中台**

30个业务域，300+微服务

技术平台架构 综合应用云原生、移动互联网、IoT、大数据和AI技术

资源配置智能化

- 货量预测覆盖全环节，偏离率<5%
- 网络规划工作全面算法化
- 场地仿真，优化布局，合理分配资源

数据业务化

- 三网数据聚合
- 面向数据用户，千人千面
- 数据地图，即席查询，自助分析

业务数据化

- 端到端全链路数字化
- 三网信息流融通（速运、快运、顺心）
- 数据埋点采集

云原生基础架构

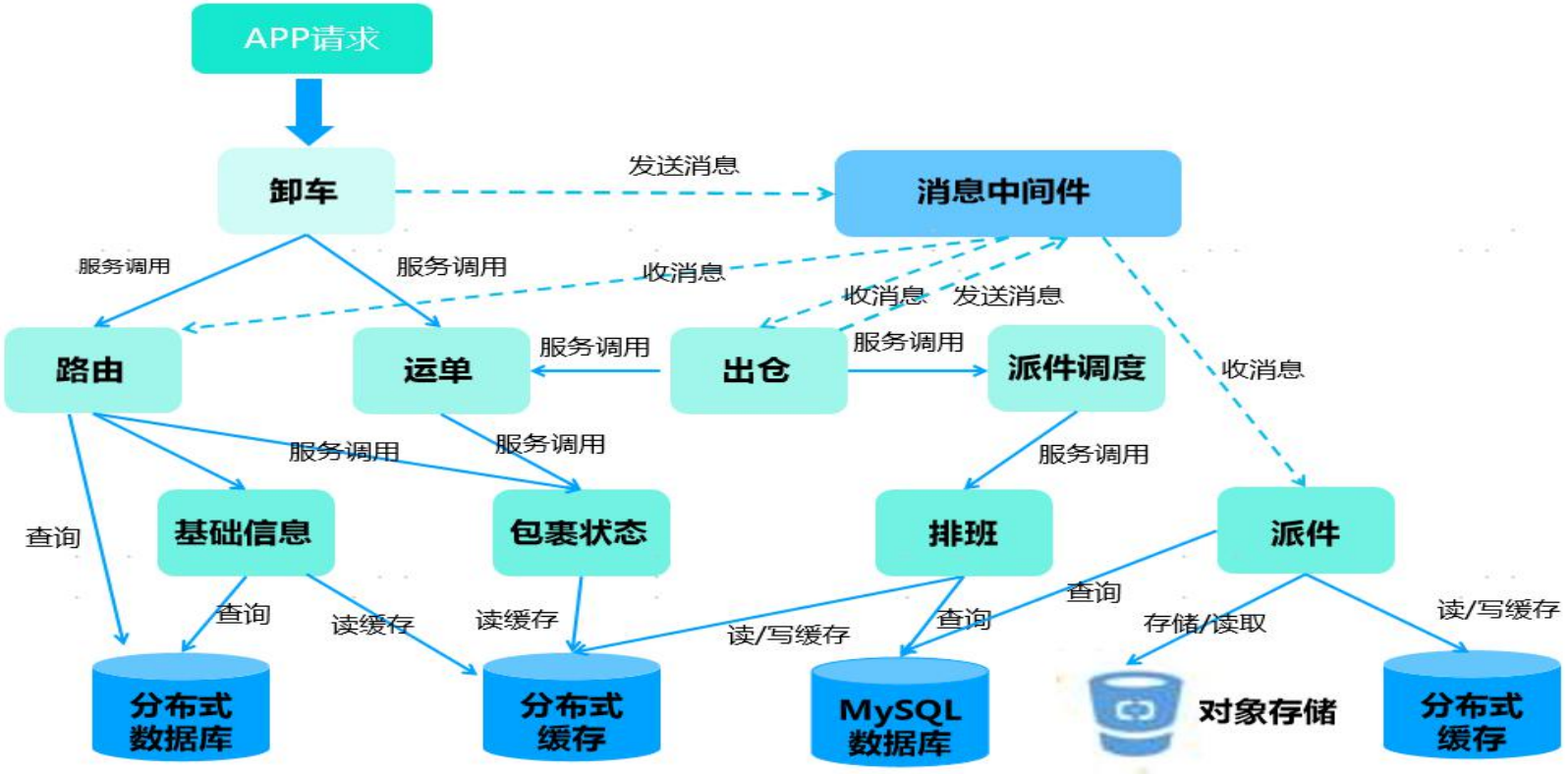
- 微服务设计，架构灵活，故障隔离和恢复
- 容器化，资源弹性伸缩
- 自助发布，碎片化交付功能

演讲目录

- 顺丰快运介绍
- 快运科技技术发展
- **全链路压测实施过程**
- 核心系统优化案例

传统性能测试面对大规模分布式系统的挑战

背景：双11系统流量大，系统之间相互影响，链路长、所使用的中间件、数据库等种类繁多，技术复杂。



- **成本高**

多个系统性能测试环境占用的硬件资源较大。性能环境的搭建链路复杂耗时长，人力成本投入巨大

- **问题定位、链路验证难**

复杂的应用拓扑，节点之间的调用关系变化比较频繁，系统问题定位越来难，部分问题只有在真正大流量下才会暴露（比如网络带宽等等）。

- **容量评估难**

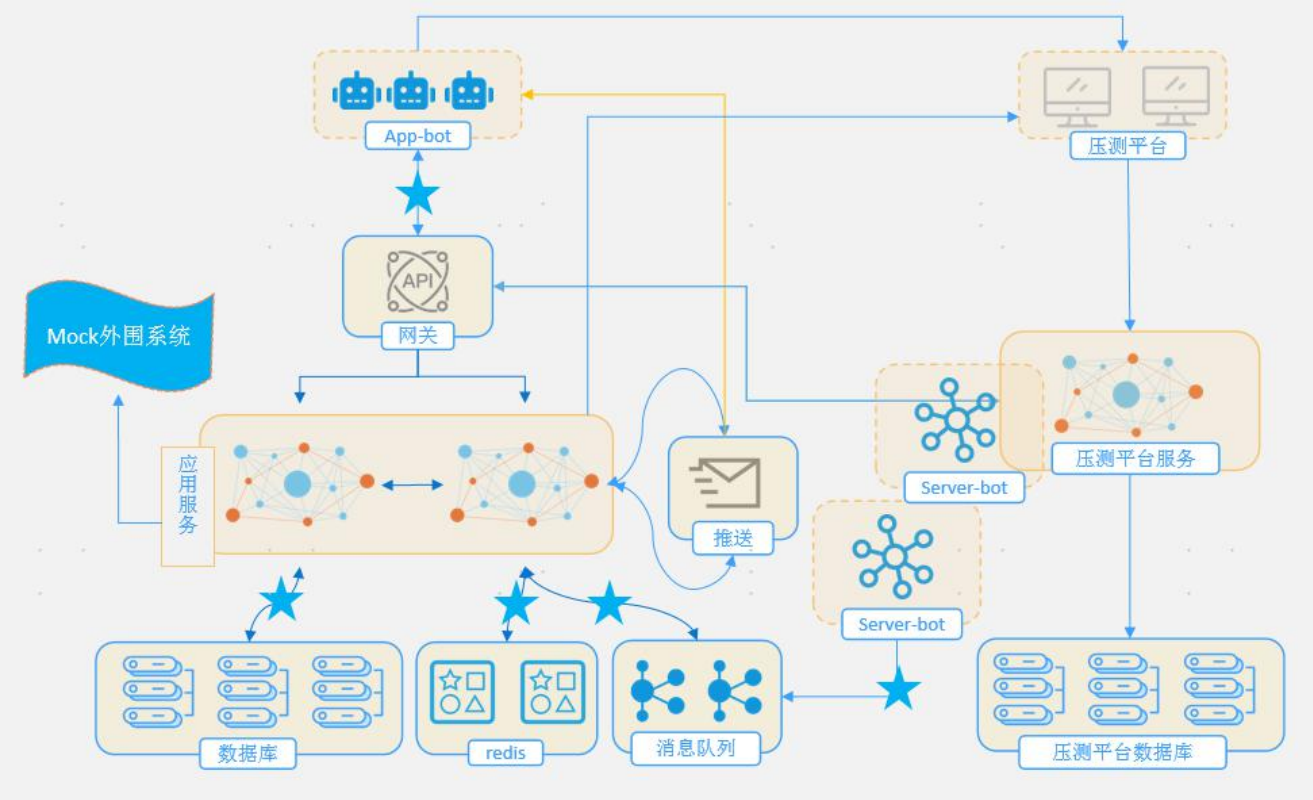
容量评估没有统一的执行标准，评估的机器资源到高峰时经常发现不准（设备配置差异、设备规模差异），易导致容量瓶颈

- **效果验证难**

分布式系统的性能测试环境和生产环境存在差异较大，性能指标参考意义有限

基于线上全链路压测的解决方案-自研

基于上面的挑战，顺丰从19年开始，通过改造中间件，实现压测数据的识别和转发到影子区域的方案，来进行链路压测。



需要对以下组件进行改造

- 网关鉴权登录
- 数据库client
- redis client
- 消息队列（MQ，kafka）client

存在不足：

1. 业务系统需要修改为适配压测所需的中间件版本

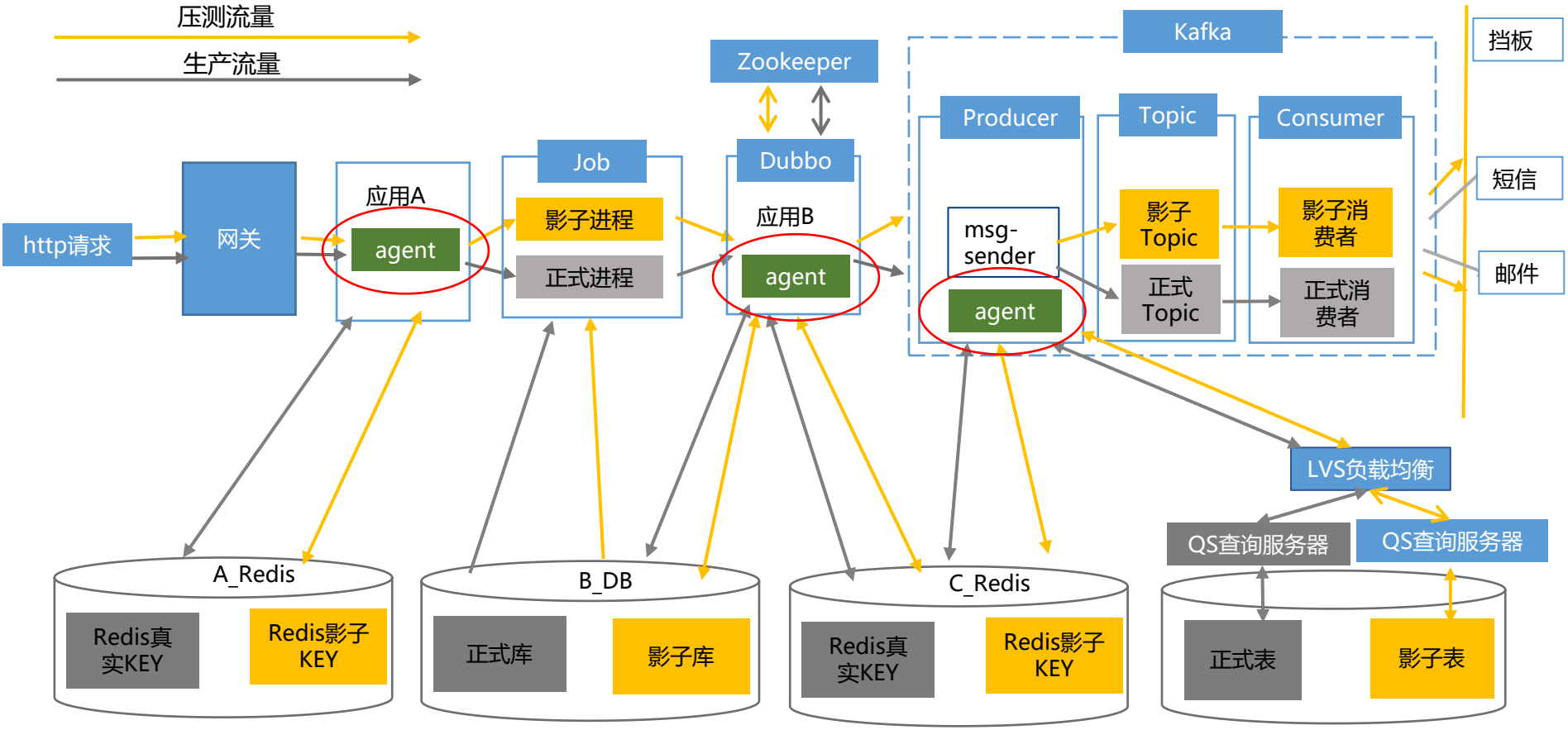
- 存在遗留及外购系统，中间件组件类型繁多
- 中间件版本多，版本不一致，老系统组件很难升级
- 架构多样，有微服务架构的系统(Dubbo,Jetty,SpringBoot)，也有传统框架系统

2.各组织推广成本高

- 改造时间成本高，推动所有系统中间件升级改造时间周期长、风险高，短期很难落地
- 改造人力成本高，缺少足够技术专家来做中间件的适配修改

基于线上全链路压测的解决方案-数列链路压测架构

基于JavaAgent字节码增强技术，业务系统无需改造，JVM层实现压测数据的识别和转发。



全链路压测两种方案对比

通过对比分析,数列相比自研具有明显优势：

- 采用动态的agent植入方式，业务接入无需额外的中间件、代码改造成本，节省大量人力
- 流量/数据隔离，支持针对不同的协议
- 支持自动梳理链路，告别人工梳理链路时代，降低链路梳理成本，提高链路梳理准确率
- 支持比较完善的监控

压测阶段	全链路压测关键能力	方案1：改造中间件（自研）	方案2：JavaAgent（数列）
压测准备	链路接入改造	需要升级到SGS定制中间件版本,业务代码可能需要改造	★需要接入数列Agent探针，业务代码无需改造
	链路梳理	人工梳理	★自动梳理，基于agent上报链路信息辅助链路梳理
	中间件支持范围	支持dubbo/kafka/redis/mysql/hbase定制版本	★支持目前主流中间件，无需改造中间件
	数据隔离	影子表/影子库	影子表/影子库
	流量隔离	流量挡板(http)	★流量挡板(应用内挡板，无协议限制) 白名单(http,dubbo)
	压测配置管理	支持	支持
压测执行	分布式压测	通过自己写的施压工具（Groovy脚本）实现	★支持开源工具，Jmeter、Gatling
	压测链路监控	/	★支持，基于agent上报链路信息，原理类似apm-skywalking
	自动识别熔断	/	★支持
压测结果	实时性能分析报告	压测性能分析需结合第三方监控工具（nmon，balant等）	★压测平台集成链路性能分析指标

全链路压测实施过程

顺丰快运2020年实施全链路压测的过程，2个月时间完成核心链路的压测实施。

项目启动

2020.7.16

01

02

2020.7.24

第一批链路接入

- 顺丰收派
- 顺心开单

第一批链路

- 顺丰收派生产压测
- 顺心开单生产压测
- 产出压测报告

2020.7.30

03

04

2020.8.31

第二批链路接入

- 顺丰和顺心中转
- 顺丰网规
- 顺心收派

第二批链路

- 顺丰中转生产压测
- 顺心收派生产压测
- 产出压测报告

2020.9.9

05

06

2020.9.15

第二批链路

- 顺丰网规生产压测
- 顺心中转生产压测
- 产出压测报告

全链路压测实施过程-关键工作

1.基础工作

- 核心压测链路确认（顺丰派件，顺心开单，中转，顺心收派，顺丰网规）
- 环境部署（压测控制台，K8S压测引擎）

2.链路梳理

- 整理业务流程图，产出链路梳理详情表
- Agent部署，对中间件进行改造、优化，使性能测试线程与生产线程进行隔离，提升安全性。

3.链路接入方案

- 压测活动目标制定
- 数据准备方案
- 链路接入清单
- 压测整体方案

4.链路接入

- 压测脚本编写，包括压测库建立脚本，清洗脚本，数据导入脚本，清理脚本，漏数检测脚本
- 压测数据准备
- 压测脚本试跑
- 压测脚本联调

5.压测试跑

- 检查准备事项
- 从1条开始逐步施压试跑并检查试跑问题
- 试跑通过后正式生产压测

6.生产压测

- 实现整体控制与调度
- 解决核心链路的大并发流量发起，任务调度，数据调度，引擎调度
- 输出压测报告等问题。

全链路压测实施过程-角色配合

实施过程

角色	压测前	压测中	压测后
研发&测试	确定压测范围 方案评审 协调资源 项目计划		压测复盘
测试	确定压测范围 压测方案编写 梳理压测链路 压测数据构造 紧急预案 监控指标 压测配置	压测预热 问题记录 压测施压 预案演练 优化复测	压测报告
研发	确定压测范围 链路改造 监控指标	观察监控指标	问题定位 链路优化
运维	接入探针 建立影子库表 监控指标	观察监控指标 紧急预案	压测数据清理 压测复盘

全链路压测成果

压测成果：

- 链路压测完成接入应用**200+**，白名单**600+**，压测场景**60+**
- 发现性能问题**30+**，并完成优化和修复
- 全链路压测相比传统高峰压测
  - 硬件资源投入减少63%
  - 人力投入减少65%
- 双十一高峰期间，系统**0**故障
- 基于链路压测的场景，实现在生产环境快速完成功能回归验证测试

工具落地过程中主要问题：

1、agent兼容性问题，包括：

- 部分场景存在漏数的情况
- 顺丰定制化的kafka不兼容
- 异步的定时任务不兼容

解决方案：在测试环境做好兼容性验证，通过升级agent修复

2、链路压测平台，配置（白名单、数据库影子库/影子表、挡板等）从测试环境迁移到生产环境，**手工迁移人力成本大**

解决方案：对平台进行优化，支持配置的导入和导出

未来规划

一、提升性能测试脚本编写和执行的效率

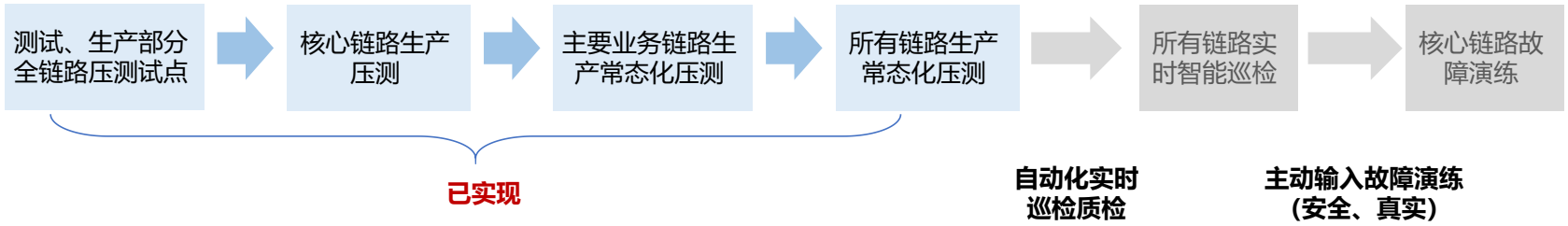
- **测试用例录制**，通过agent直接获取入口的input数据和格式，减少测试用例编写的耗时。
- **压测流量录制**，录制现网流量，并且在压测的时候回放，模拟真实用户的流量比例和访问行为。
- **非人工自动化测试**，建设不基于人编写测试用例的自动化回归和拨测能力。

二、以全链路压测为核心，建设系统稳定性保障

- 结合降级限流、监控体系的完善、灾备双活、快速扩缩容等

三、建立基于全链路的生产常态化压测和智能巡检质控体系

- 从被动式测试、监控，跨越到主动式压测、巡检，频次越来越高、越来越自动化
- 主动制造故障演练以不断验证和提升质量



演讲目录

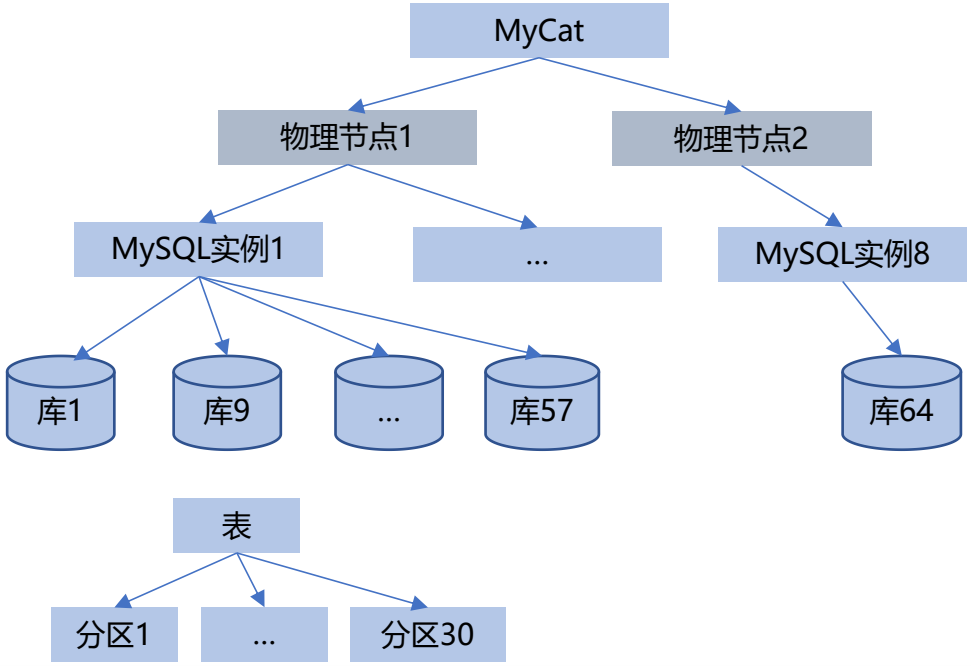
- 顺丰快运介绍
- 快运科技技术发展
- 全链路压测实施过程
- **核心系统优化案例**

运单中台-性能优化

- **背景**：快运运单服务中台提供统一的运单信息，运单包裹状态，运单标签等信息管理及查询服务，支持收派，中转，质控，结算等业务场景，支撑四网融通相关运单服务。

- **高峰目标**：
  - 运单高峰数据处理计算量：300万/5min、包裹状态： 600万/5min
  - 运单查询接口：支撑日均5000万次的请求调用，高峰QPS：3000，接口平均响应时间30ms以内
- **问题描述**：
  - 查询QPS高时**MySQL物理机CPU飙升到70%**
  - 多场景混合压测：数据处理出现瓶颈，数据库**CPU高峰达到80%**

- **架构分析**：
  - 快运运单服务的MyCat下挂了64个分片库：2台物理机*4个MySQL实例*8个库
  - **表分片字段**：运单号
  - **表分区字段：**创建时间



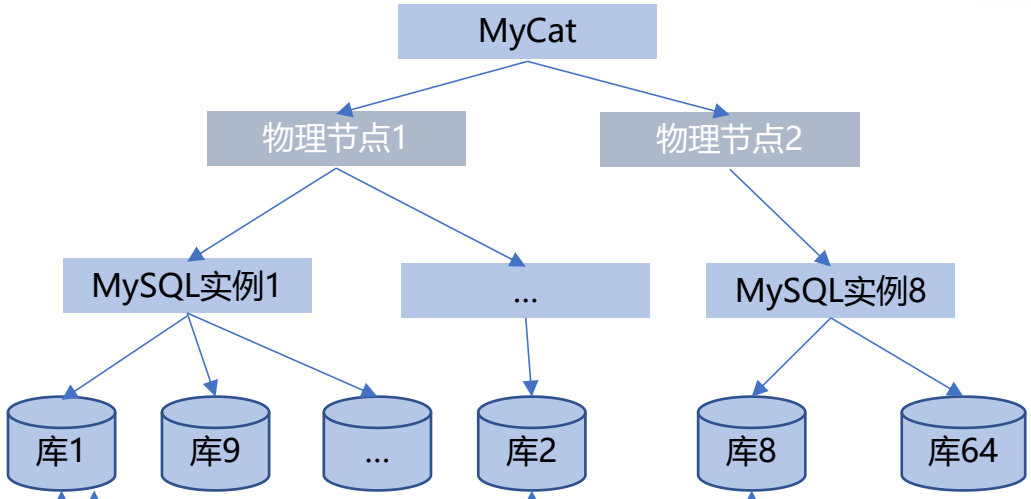
运单中台-MyCat-MySQL性能优化-分片

跨分片查询物理机**CPU**：

单物理机：
2%*8*4=64%

单实例：
2%*8=16%

单库：2%



按件查询 √
按票查询 ×

件与票关系表：SF1234567891001　SF1234567891002 SF1234567891003
（**分片字段：件**）

优化后
按票查询 √

票与件关系表：SF1234567891001
（SF1234567891001,SF1234567891002,SF1234567891003）
（**分片字段：票**）

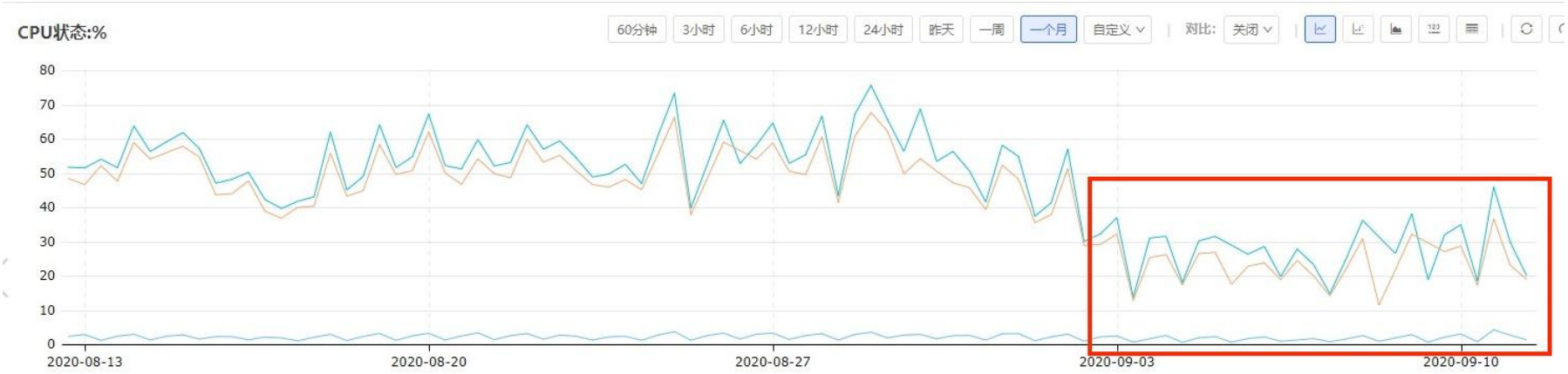
场景一：跨分片字段查询：一票多件按**票运单号查询**件列表，查询件与票关系表时并发到**64个库**查询，导致物理机CPU飙升到70%以上。

分析：单库CPU2%不高，但是由于MyCat对应数据库物理机集群共用导致会放大**32倍。**

优化方案：不扩容，调整存储架构，避免跨分片查询，添加一张冗余票与件关系表，按**票运单号做分片字段**

运单中台-MyCat-MySQL性能优化-分片

优化后：CPU使用率峰值从70%下降到40%，效果显著



曲线名称	最大值	最小值	平均值
CPU used:-	75.61%	13.69%	46.57%
user used:-	67.66%	11.47%	41.89%
sys used:-	4.22%	0.52%	2.09%

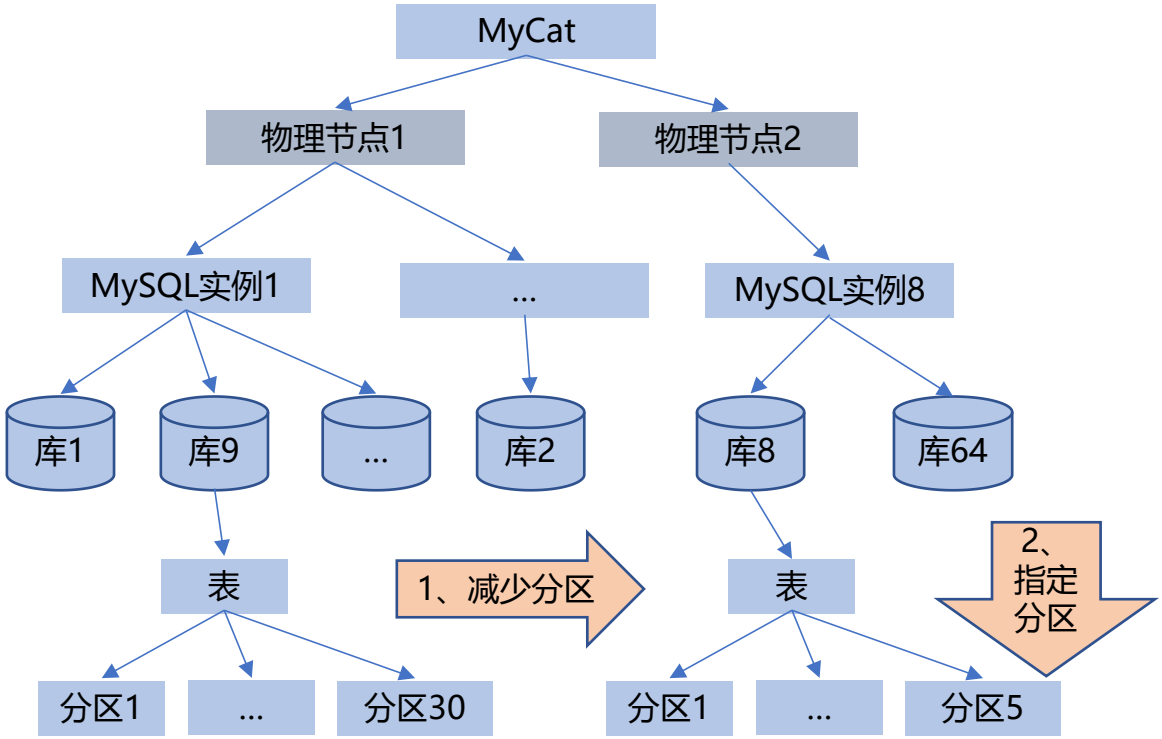
运单中台-MyCat-MySQL性能优化-分区

跨分区查询物理机
CPU：
单物理机：
1%*N*8*4=1%*3
2*N倍

单实例：1%*N*8

单库：1%*N

单分区：1%



场景二：分区数过多：运单包裹信息表按天分区保留30天，查询会扫描30个分区，QPS上千时CPU飙升80%

未指定分区：查询或更新时导致全分区扫描，CPU飙升

优化方案：

1、减少分区：按天保留30天调整为按周分区保留**5周**，数据量有增加但逻辑分区由30减少到5，CPU资源下降到30%

2、指定分区：查询或更新时指定分区时间

路由中台-处理能力性能优化

■ **背景**：路由中台服务支持一票货物从开单收件到客户签收，基于实时高效的数据处理技术，提供全链路生命周期的货物最新状态追踪查询，支撑四网融通相关路由服务。

■ **高峰目标**：

- 快运路由每天**亿级**数据， **4000万/5min**的kafka数据处理量。
- 完成**每秒15万**的路由计算能力。

■ **问题及优化迭代方案**：

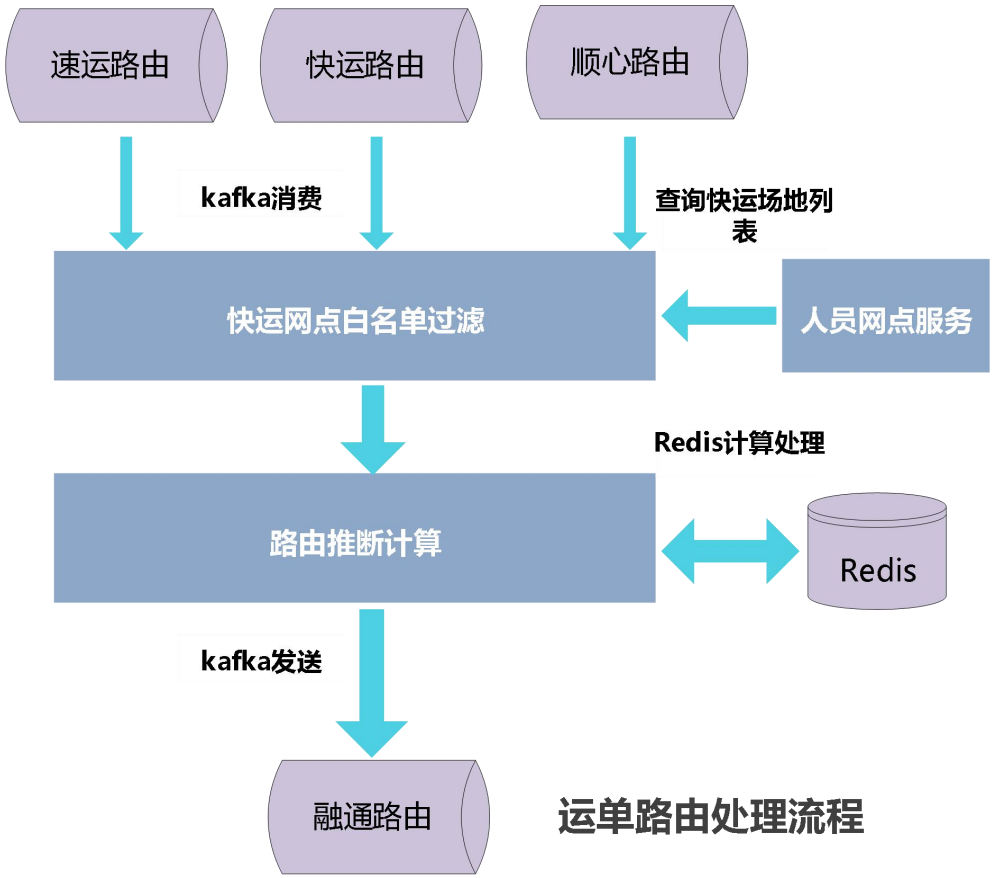
➢ kafka消费速率达到**1000万/5min**后出现瓶颈

优化方案：

1. 消费者调整：线程数由32调整为64
2. 生产者调整：主题分区数由8调整到16，生产者线程数由32调整为64。

➢ kafka消费速率达到**2000万/5min**后出现瓶颈，Redis执行次数只能达到**1.5万/秒**

优化方案：Redis线程数由32改为64，Redis集群从4主4备扩容到8主8备。



运单路由处理流程

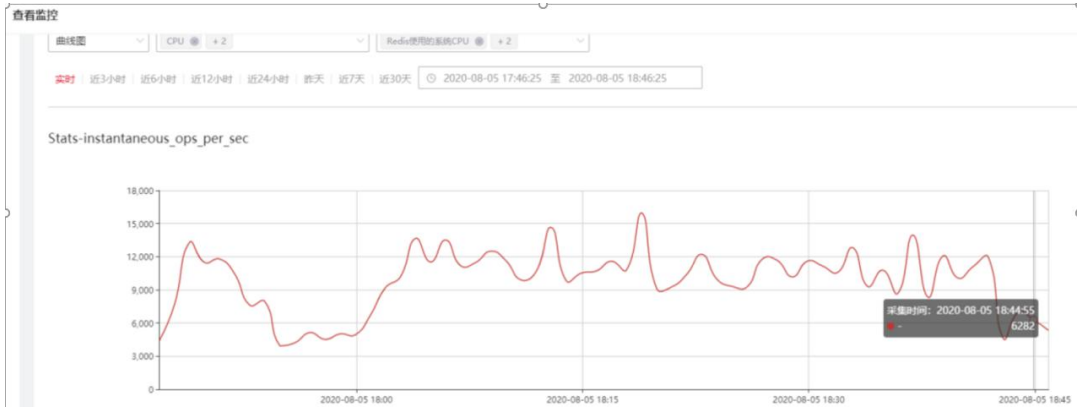
路由中台-处理能力性能优化

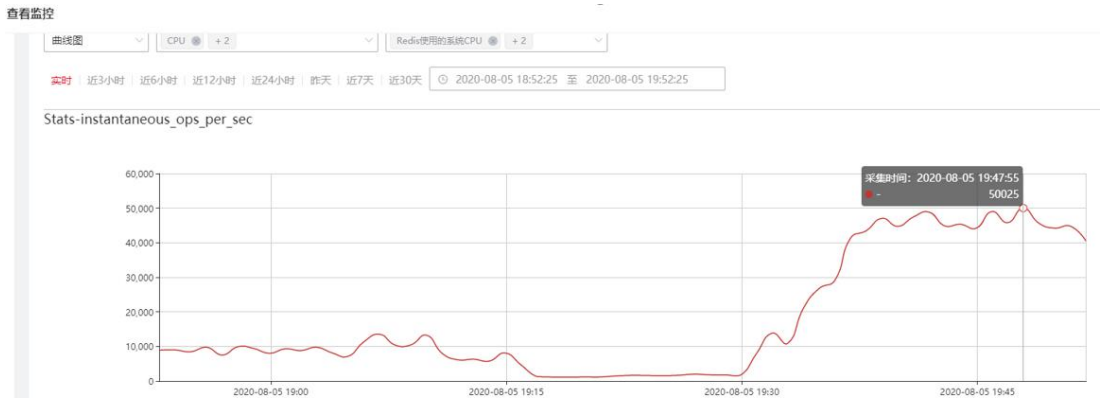
- **问题及优化迭代方案**：

kafka消费速率达到**2500万/5min**后出现瓶颈，Redis执行次数只能达到**2万/秒**。

优化方案：

通过分析应用CPU高，分析线程DUMP发现RedisCacheManager.addCache方法占用很多RUNNABLE线程，存在Spring事务，是由于框架拦截所有add*方法添加了事务，排除后Redis执行次数达到**5万/秒**。





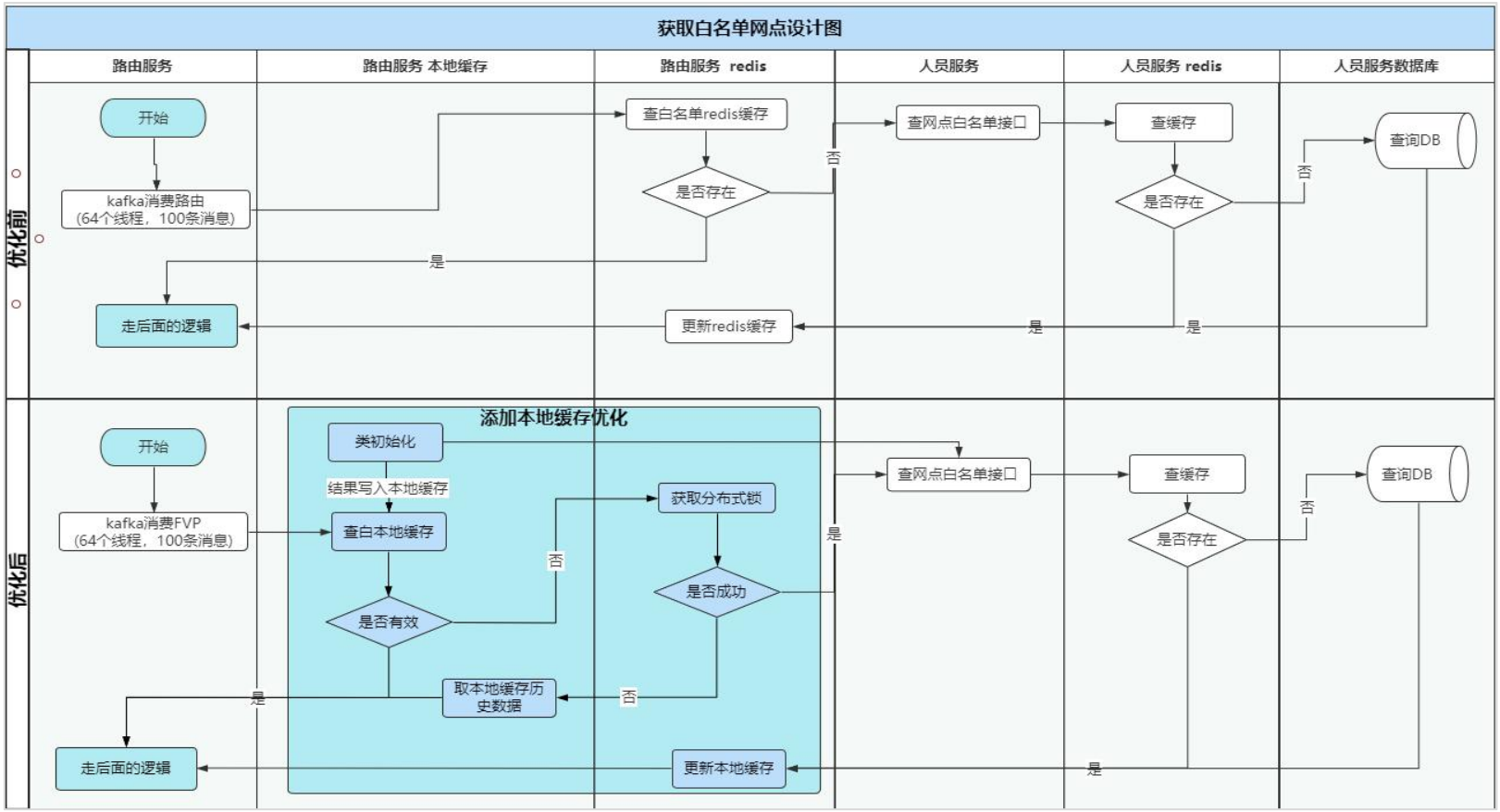

```
- locked <0x00000006844a1500> (a com.mysql.jdbc.JDBC4Connection)
at com.mysql.jdbc.ConnectionImpl.commit(ConnectionImpl.java:1555)
- locked <0x00000006844a1500> (a com.mysql.jdbc.JDBC4Connection)
at com.alibaba.druid.filter.FilterChainImpl.connection_commit(FilterChainImpl.java:199)
at com.alibaba.druid.filter.FilterAdapter.connection_commit(FilterAdapter.java:782)
at com.alibaba.druid.filter.FilterChainImpl.connection_commit(FilterChainImpl.java:194)
at com.alibaba.druid.proxy.jdbc.ConnectionProxyImpl.commit(ConnectionProxyImpl.java:122)
at com.alibaba.druid.pool.DruidPooledConnection.commit(DruidPooledConnection.java:755)
at org.springframework.jdbc.datasource.DataSourceTransactionManager.doCommit(DataSourceTransactionManager.java:313)
at org.springframework.transaction.support.AbstractPlatformTransactionManager.processCommit(AbstractPlatformTransactionManager.java:765)
at org.springframework.transaction.support.AbstractPlatformTransactionManager.commit(AbstractPlatformTransactionManager.java:734)
at org.springframework.transaction.interceptor.TransactionAspectSupport.commitTransactionAfterReturning(TransactionAspectSupport.java:518)
at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.java:292)
at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:96)
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)
at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:92)
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)
at org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(CglibAopProxy.java:673)
at com.sf.                    .impl.RedisCacheManager$$EnhancerBySpringCGLIB$$7ed062b3.addCache(<generated>)
```

路由中台-稳定性性能优化

■ **问题及优化迭代方案**:

消费速率达标后，性能不稳定，通过链路追踪发现存在**Redis缓存击穿**情况：消费处理过滤场地白名单时，查询场地列表接口缓存失效后QPS高达15万。

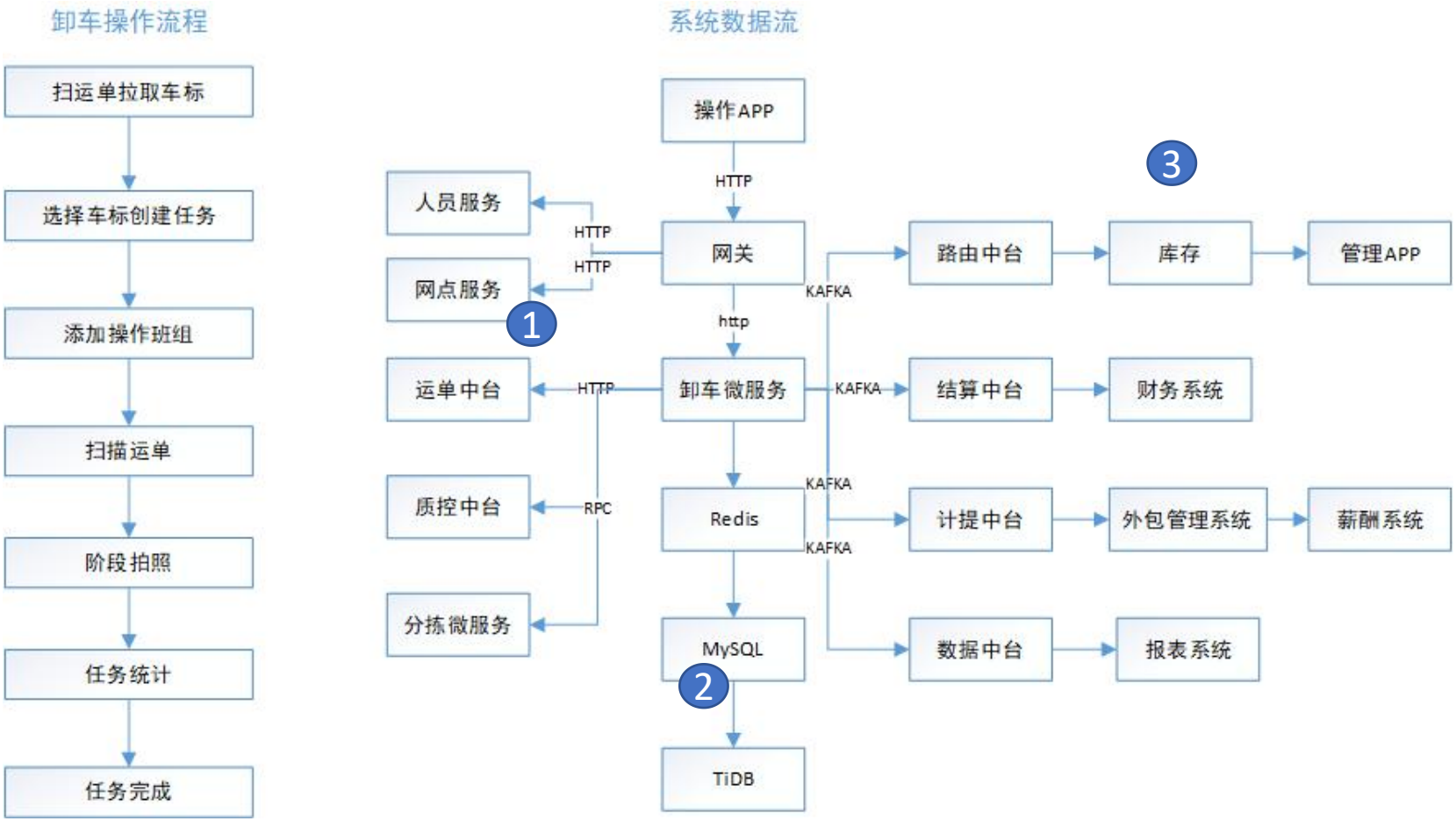
优化方案：添加**多级本地缓存**，Redis高频使用的场景通过提前加载+分布式锁，减少请求次数，缓解服务端压力，避免缓存雪崩，提升稳定性。



中转核心链路 - 卸车

背景：卸车服务在派件出仓早高峰时段的并发调用量高1000qps，且要求低延迟的响应，基于2020年顺丰双十一的高峰预测量，制定平日三倍QPS请求量的压测方案3000qps

架构模型：



中转-卸车流程优化

1、系统间调用性能瓶颈改进

- **背景**

卸车服务构建应卸明细，高并发，低时延，聚合复杂度高，需要将整车货物的运单、质控、分拣等信息合并，在用户扫描运单之前返回app，链路压测发现批量查询运单存在性能瓶颈。

- **问题描述**

并发请求超过100，批量查运单接口大量超时。

- **优化方案**

- 卸车服务开启**熔断限流**，超时时间缩短为3s，避免长期占用应用线程。
- 运单服务调用优化：运单查询按需最小化查询相关信息，只查询卸车需要的运单信息，运单标签、费用、信任标识等信息不查询。
- 运单批量查询接口优化，控制每次批量请求100条
- 为了避免一趟车几万件货，查询并发数瞬间飙升，开启多线程，控制线程数在100以内。

2、TiDB性能调优

- **背景**

卸车扫描上报，是系统并发最高的接口(3000qps)，操作数据写TiDB明细表，单表数据量超过3亿，用作人员计提。

- **问题描述**

链路压测，发现大量写冲突及偶发执行计划失效。

- **优化方案**

- 打开重试开关，设置tidb_disable_txn_auto_retry=true。
- SQL语句固化索引（ force index）。
- tidb集群从3.0升级到4.0.10。

中转-库存聚合查询，ES性能优化

- **背景**：

需要根据复杂多变的线路配配载，提前拉取满足配载的货物。提高装车速率。同时实时监控统计场地库存，指导主管进行人车资源协调。

- **问题描述**：

- 根据线路配载拉库存QPS上不去。不能满足压测要求（300qps）
- 统计场地库存数据的聚合查询QPS(200qps)上不去，聚合场景下es集群内存old gc很严重，长达10秒+ 。拖垮整个集群的查询速率。

- **优化方案**：

- 1、建索引优化：

  1）不用检索的字段不建索引； 2）尽可能只用keyword类型

- 2、查询优化：

  1）filter代替must,不会计算分值,且使用缓存； 2）精简返回字段，减少网络带宽

- 3、集群版本优化：

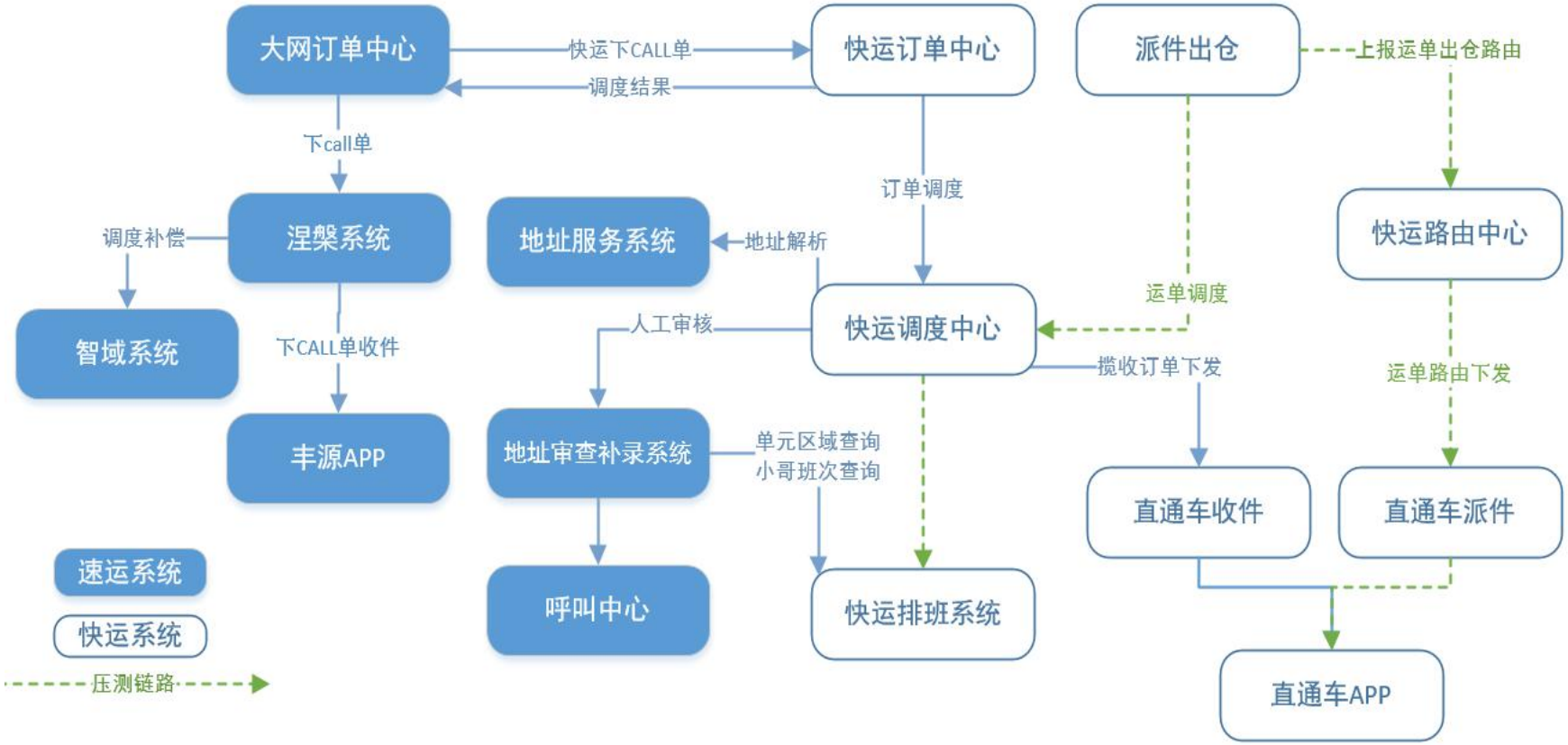
在线链路聚合查询压测结果对比：针对es的5.x版本聚合查询集群内存gc问题，升级到7.x版本

序号	es版本	场景	并发用户数	QPS	平均响应时间（毫秒）	es集群性能瓶颈	说明
1	V5.x	调优前	100	30	3310ms	gc最高时长：4299ms	
2	V5.x	优化后	100	38	2091ms	gc最高时长：2106ms	有提升，效果不明显
3	V7.x	调优前	100	1295	76ms	cpu最高：83%	对比5.x版本QPS有很大提升
4	V7.x	优化后	100	**1647**	59ms	cpu最高：76%	开启缓存等优化后QPS有小的提升

收派-线程性能优化

背景：调度中心服务在派件出仓早高峰时段的并发调用量高，且要求低延迟的响应，因此基于2020年顺丰双十一的高峰预测量，制定平日三倍QPS(3459次/秒)的性能压测指标。

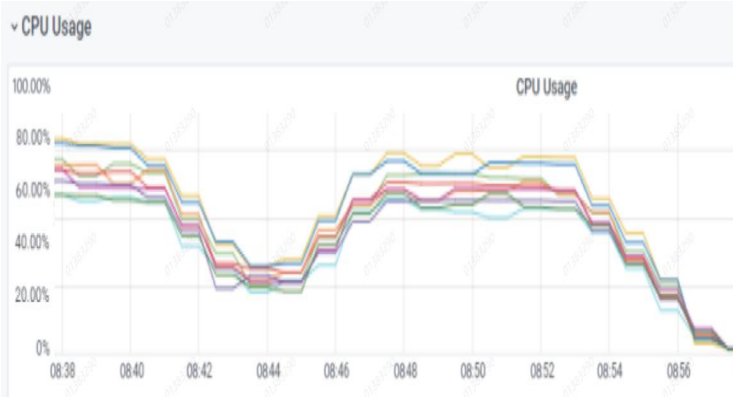
架构模型：



收派-线程性能优化

■ 压测问题：

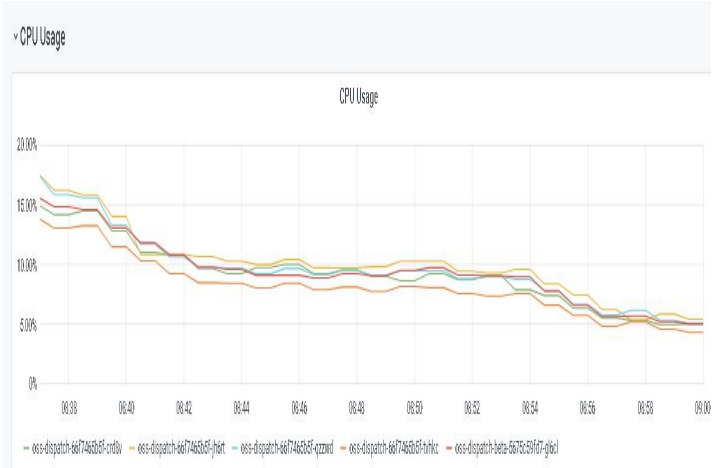
- YGC响应耗时50-100ms，且JVM内存分析使用率低
- 多场景混合压测下dubbo连接池队列满，拒绝服务请求
- 多线程异步调用，线程池submit Callable任务总执行时间没有缩短



优化前CPU性能图

■ 优化方案：

- 分析发现容器的JVM ParNew线程数为64，与容器物理机核心数一致，容器配置为4核6G，过多的GC线程导致上下文切换频繁，指定JVM参数ParallelGCThreads设置成和CPU核数相同
- 分析业务场景，基础服务数据可利用JVM内存作为一级缓存，减少服务间的调用次数，释放更多的网络调用资源
- 多线程异步使用长事务与短事务线程池隔离，相互不影响



优化后CPU性能图

收派-线程性能优化

- 压测结果性能指标：

性能指标		生产环境目前性能	高峰压测
		日常高峰量	日常高峰量的3倍
派件调度-K8S	CPU使用率（%）	13.90%	39.30%
	jdbc连接数使用率（%）	1%	6.67%
	http线程使用率（%）	1%	6%
	Redis应用连接数使用率（%）	1%	6%
	最大处理能力（次/秒）	1,161	4,244
派件调度-REDIS	CPU（%）	0.06%	0.64%
	每秒执行KEYS	996.00	3858.00
	连接数Clients	47.00	398.00
	Redis内存使用率（%）	0.17%	0.26%
派件调度-MYSQL	MYSQL(CPU使用率)（%）	17.00%	40.00%
	MYSQL(QPS)	6.7	7102
	MYSQL(TPS)	0	3539
	MYSQL(活动会话)	5	21

网规系统性能优化-背景说明

- **案例背景**：网络规划领域包含规划快件路由轨迹、计算快件时效、实现异常预警等功能。
- **案例说明：**业务需要计算快递运单轨迹的服务，计算出全网任意两个网点快件时效最快的到达方案。
- **挑战**

  计算量大

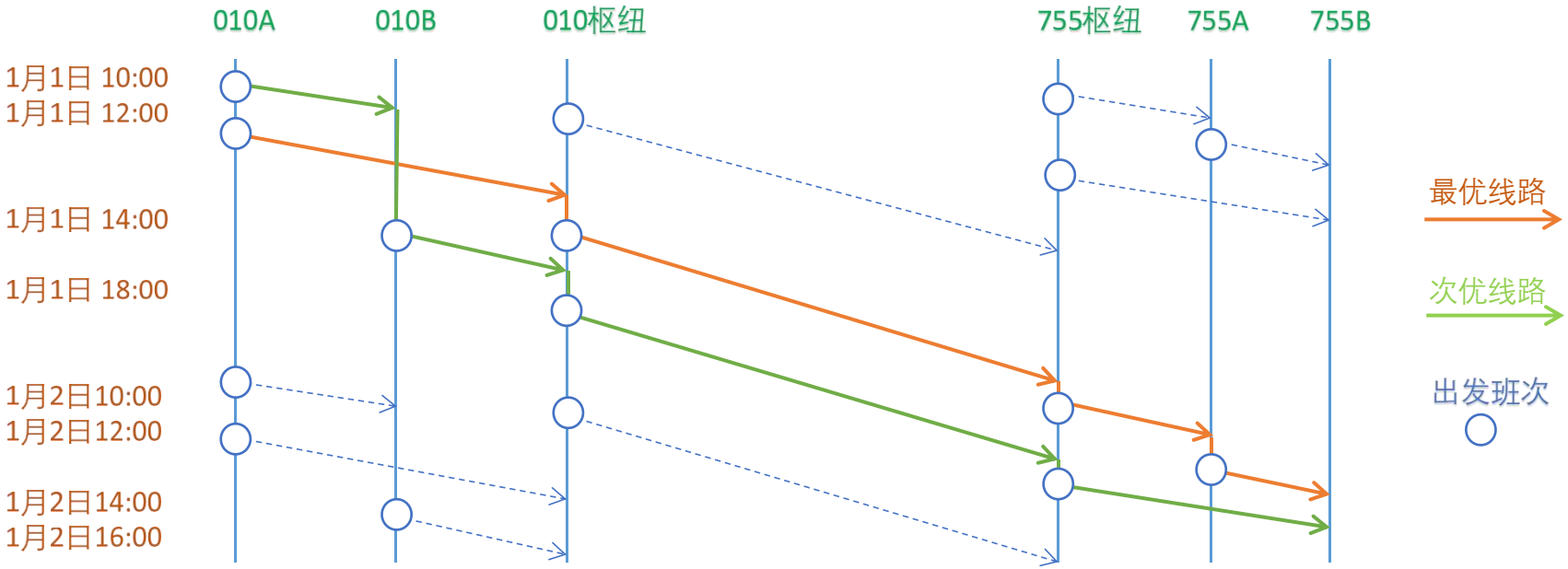
  　　百万级快件都需要独立计算最优运行线路。

  数据结构庞大

  　　万级节点和百万级边有向图。

  算法复杂度高

  　　时空双维度的最优路径查找算法。



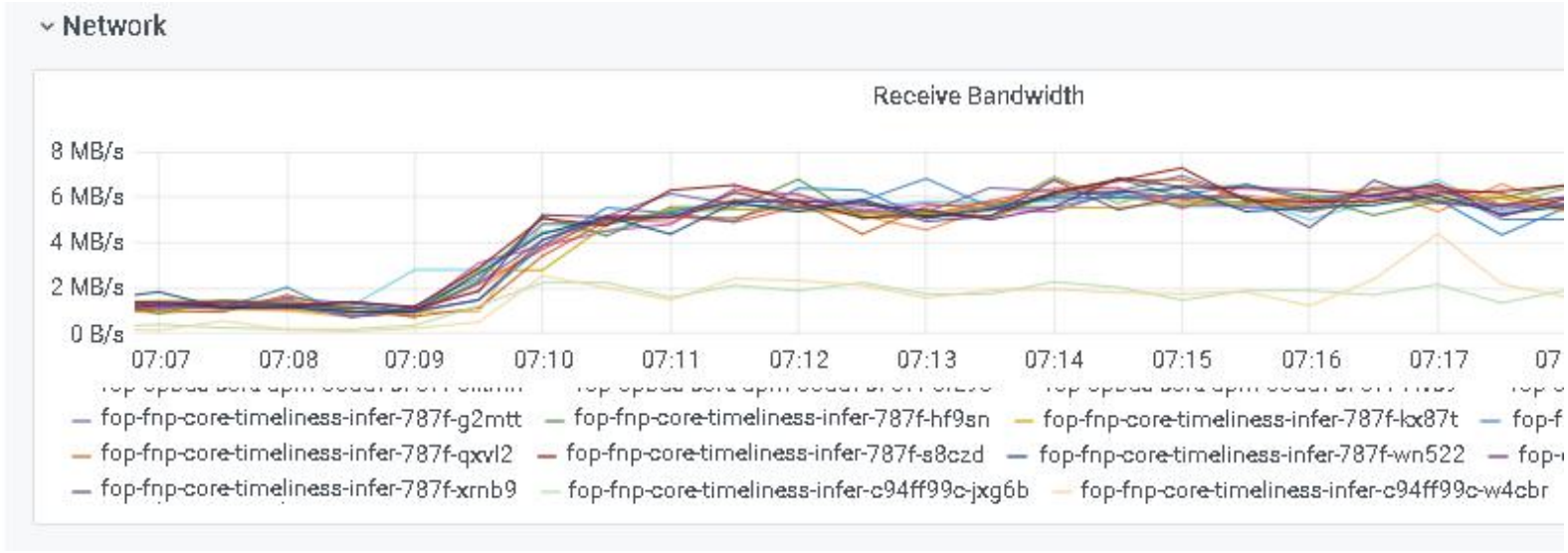
网规性能优化-redis访问网络瓶颈优化

- **案例背景：**
  - 系统中大量使用Redis缓存相似路由的中间计算结果，减少重复计算。
  - 单案例压测时无问题，多案例共同压测时出现redis响应缓慢问题。

- **问题分析过程：**
  - Redis集群性能分析：无热Key、无慢查询、性能表现良好。
  - 应用集群性能分析： CPU、内存峰值均未到达极限、也排除线程池问题。
  - 增加应用节点压测仍然无提升，怀疑是网络瓶颈，通过云监控最终确认应用服务器与redis集群存在网络流量瓶颈。
  - 根本原因：Redis缓存的对象体积过大 30000qps*30Kb ≈ 900Mbps。

优化前网络带宽

6MB*19 ≈ 912Mbps

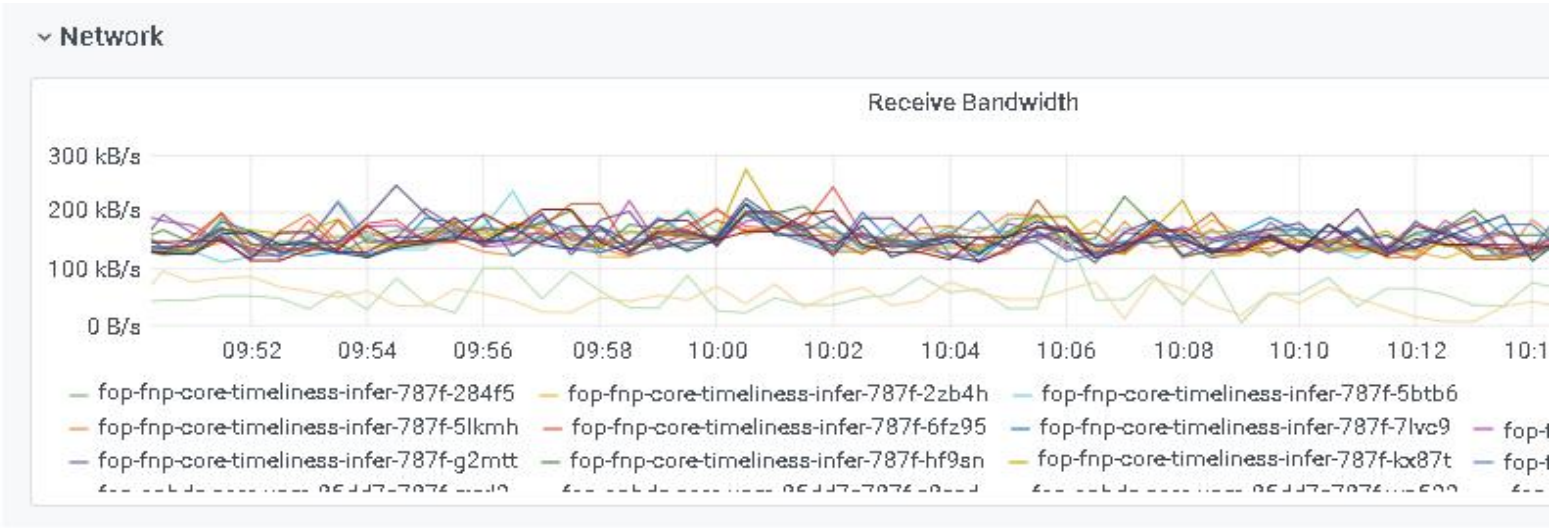


网规性能优化-redis访问网络瓶颈优化

- **优化方案：**
  - 多级缓存
    使用本地缓存代替部分redis缓存，降低网络压力。
  - 数据结构优化体积
    放弃方案：
    1.采用gzip算法压缩，CPU消耗大幅度增加。
    2.采取不同的对象序列化方式效果不明显。
    采用方案：
    1.精简数据结构,去掉不必要的字段。
    1.对字符串常量数组使用数字Integer对其进行编码，体积降低1/2。
    2.使用short代替integer进行编码 体积再次降低1/3。
  - 算法优化:深度遍历方式改完广度遍历方式，降低请求数。

优化后网络带宽

200KB*19 ≈ 30.4Mbps



网规性能优化-图形计算优化

- **背景**：顺心路由串联是计算快递运单轨迹的服务，在顺丰的货物运输的网络下，计算全网任意两个网点的是时空维度的运行轨迹。
- **挑战** 计算可行线路复杂，每天计算500W+条可行线路
- **架构演进**

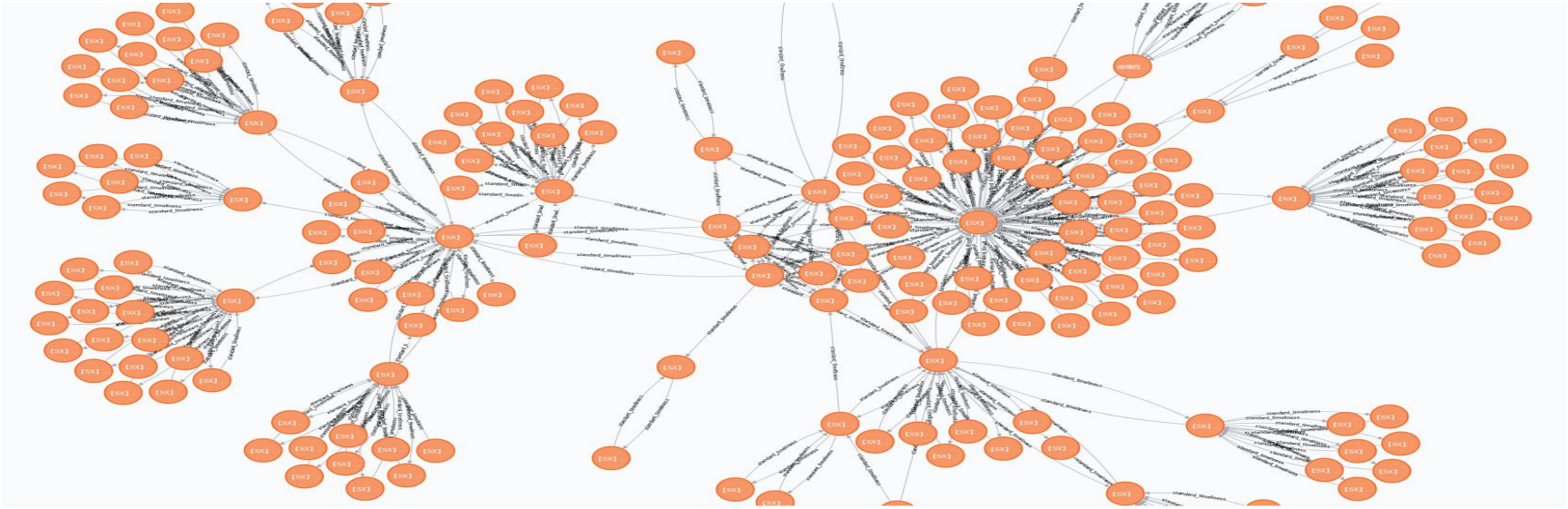
优化前：非图形计算

- 数据全部加载到内存，单实例内存占用20G。
- 手写算法实现复杂逻辑，算法优化困难，占用CPU很高，常态下占用6台8核实例。

优化后：图计算

- 将**空间网络映射成图模型**，即网点是顶点，线路即是有向边。
- 将**时间属性映射成边属性**，降低了整个模型边数量。
- 使用图形数据库自带算法查找合适的线路。
- TPS从100提升至3000

Neo4j图模型局部示例



结束语

技术是星辰大海

落地靠水滴石穿

麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手2000余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过3000余家企业续约学习，是科技领域占有率第1的客座导师品牌，msup以整合全球领先经验实践为己任，为中国产业快速发展提供智库。

高可用架构公众号主要关注互联网架构及高可用、可扩展及高性能领域的知识传播。订阅用户覆盖主流互联网及软件领域系统架构技术从业人员。

高可用架构系列社群是一个社区组织，其精神是“分享+交流”，提倡社区的人人参与，同时从社区获得高质量的内容。