百万司机在线打车平台架构演进

沈剑 - 快狗打车CTO

2021 InfoQ 技术大会近期会议推荐

—— 盘点一线大厂创新技术实践

北京站

GMTC 全球大前端技术大会

时间：2021年07月04-05日

地点：北京 · 国际会议中心

扫码查看完整日程

深圳站

ArchSummit 全球架构师峰会 th

时间：2021年07月23-24日

地点：深圳 · 大中华喜来登酒店

扫码查看大会专题

关于-我

- “架构师之路”作者，深夜写写技术文章
- **到家集团 - 技术委员会主席**
- **快狗打车(原58速运) - CTO，负责产研**
- 前58同城 - 技术委员会主席，高级架构师，技术学院优秀讲师
- 前百度 - 高级工程师

【短途】【即时】【货运】

快狗打车

创业初期，对技术团队与系统的要求?

快速迭代

快速发展期，对技术团队与系统的要求?

稳定性，扩展性，**合理性**

矛盾来了，系统不能突变

如何搞定“快”与“稳定性，扩展性，合理性”的过渡演进？

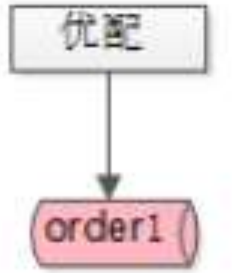
目录

- 多业务，订单中心，**系统统一**过渡演进
- 高并发，经纬度检索，**系统降压**过渡演进
- 多场景，消息中心，**系统抽象**过渡演进
- 多策略，策略中心，**系统解耦**过渡演进
- 总结

一、订单中心，从“快”到“合理”

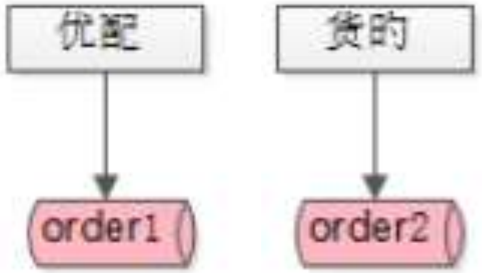
第一个业务

订单系统，如何快速实现？



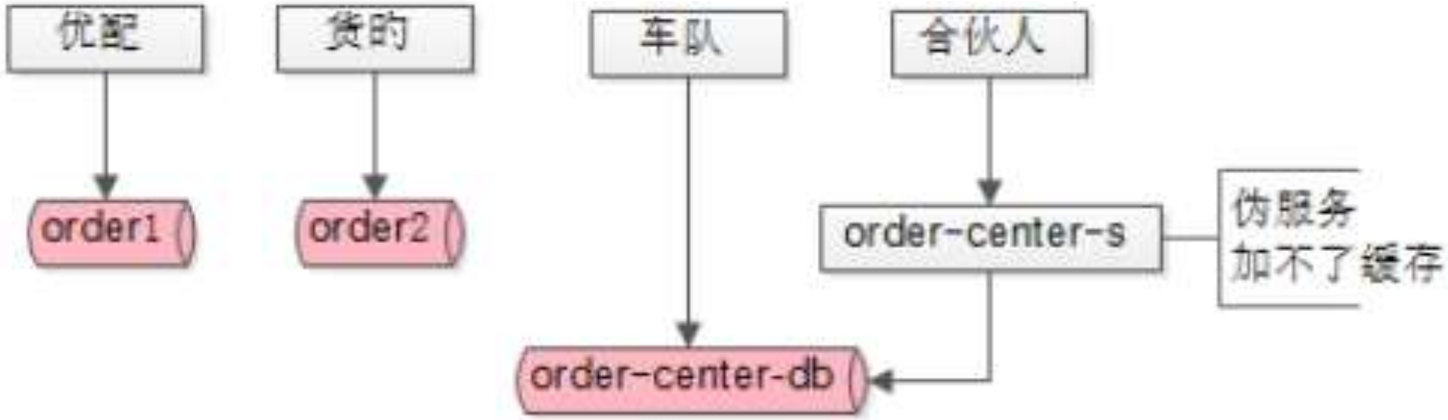
第二个业务，如何快速实现？

闭环



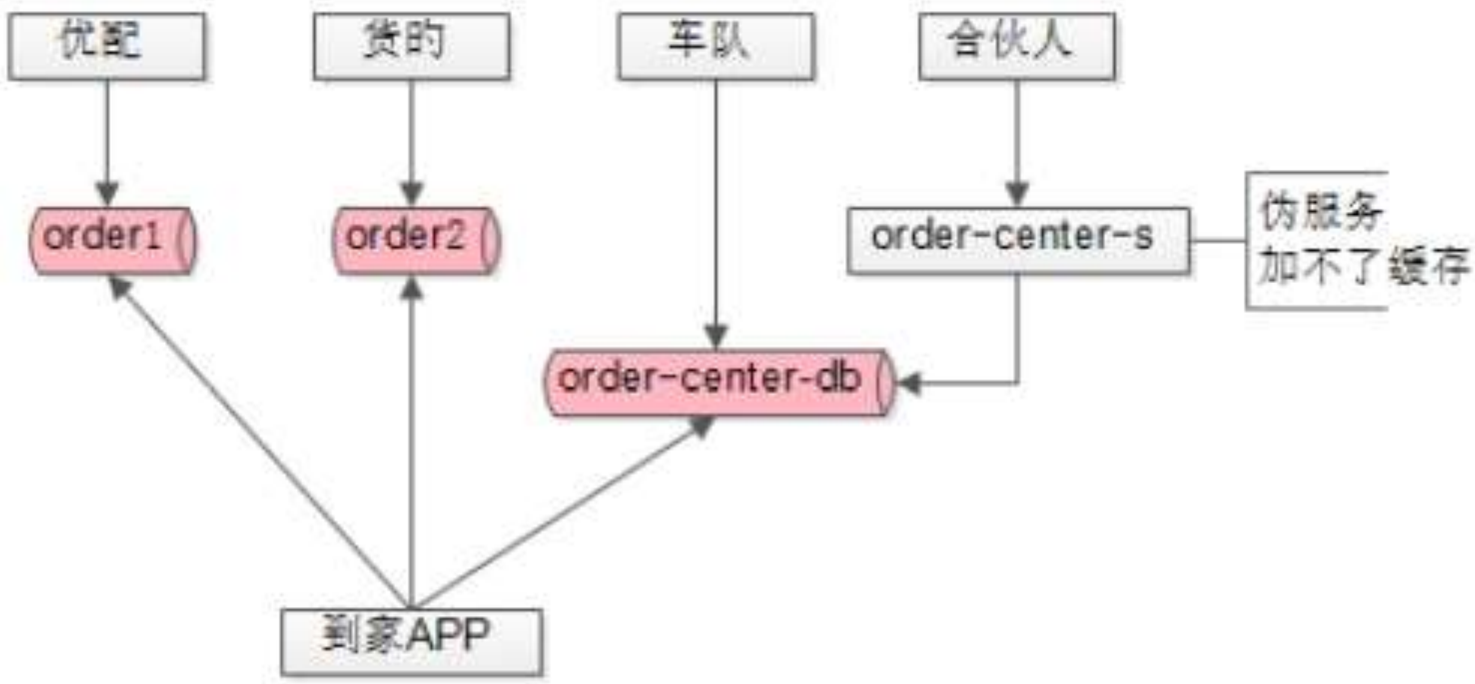
第三个业务，第四个业务，对点不对劲了，得统一

业务三有了统一意识，业务四有了微服务意识



有统一展现的需求

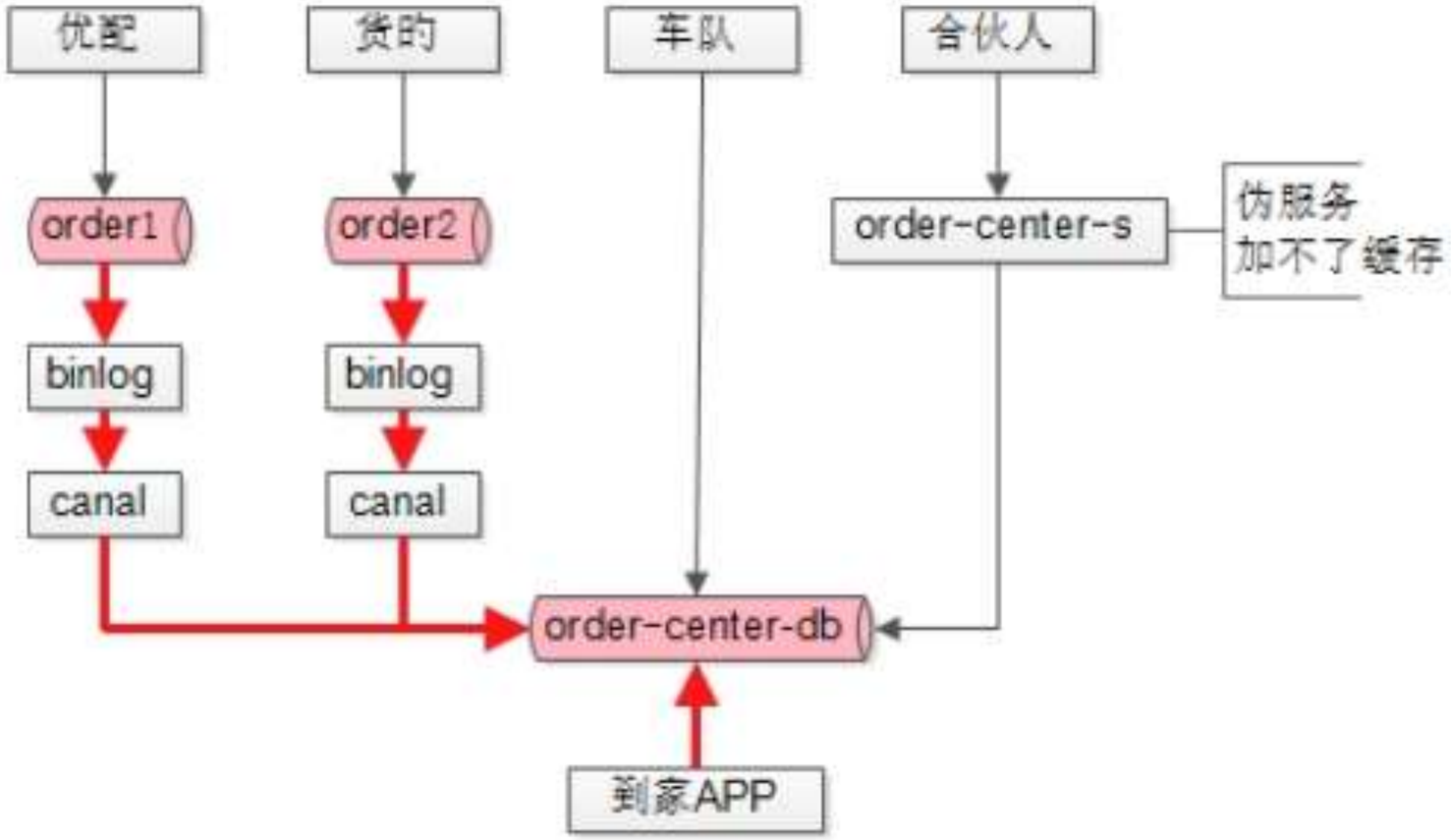
麻烦了，要访问多库，抱怨了，为何不一早就统一?



统一是合理的，问题转化为，怎么统一？

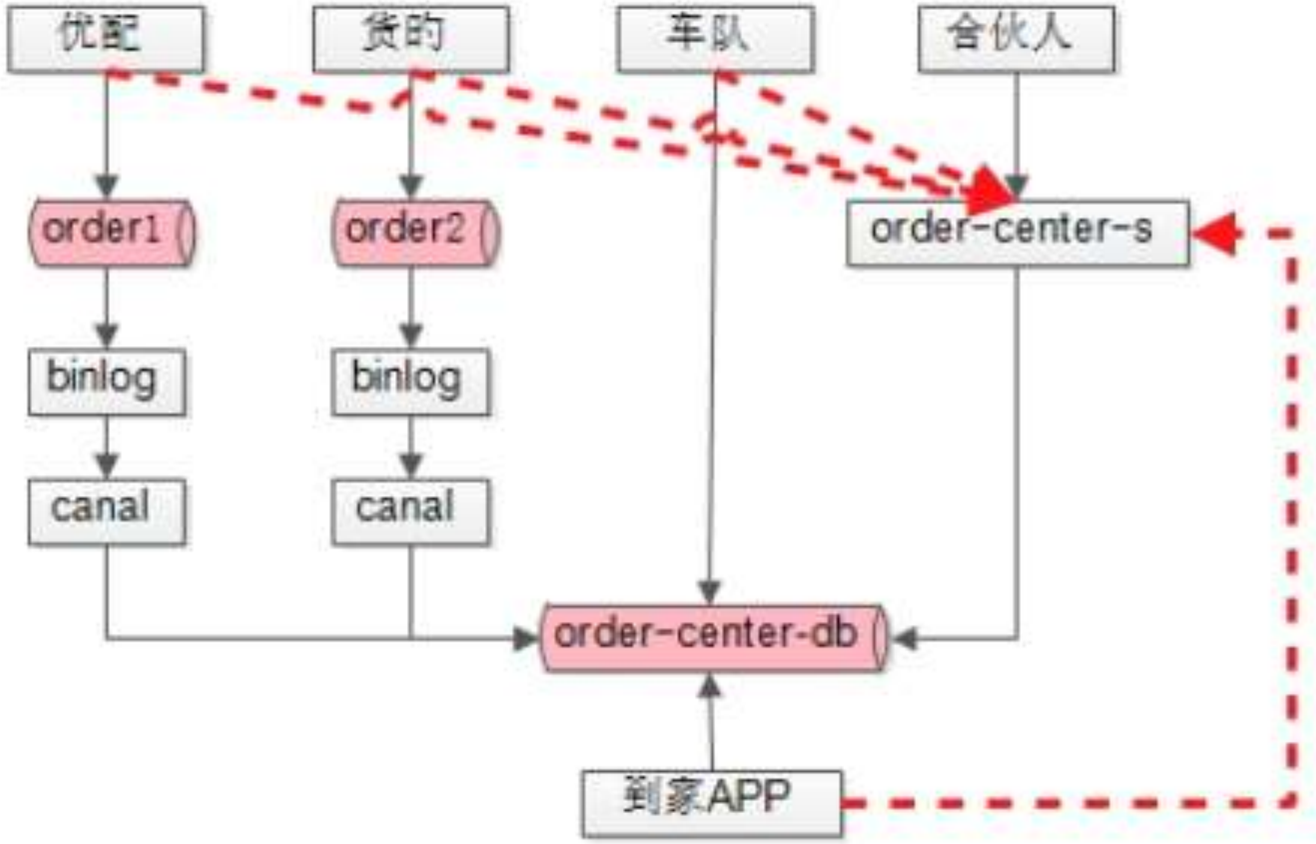
第一步：

数据收口（尽量减少对原业务的影响）



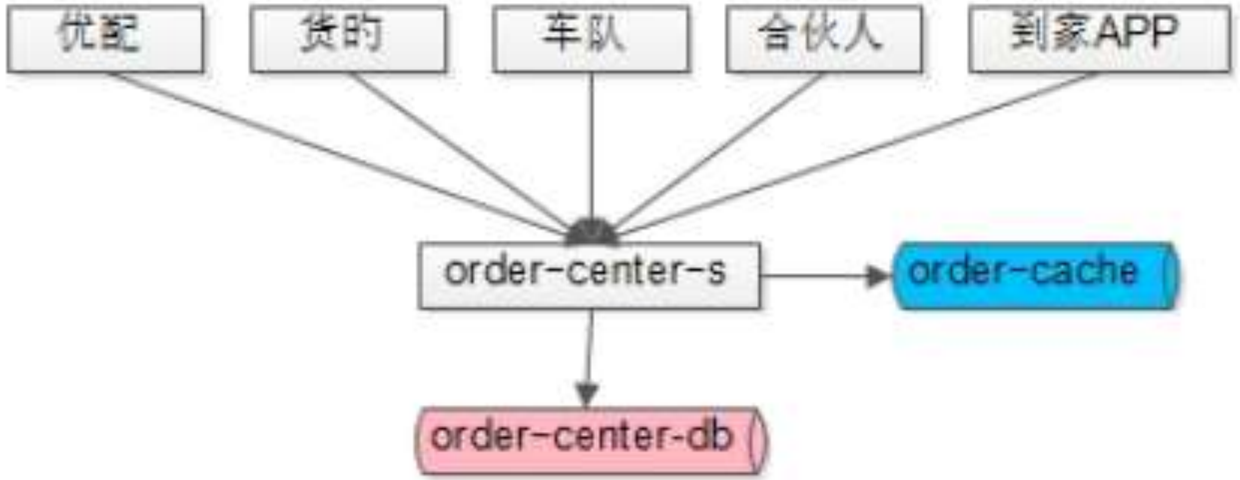
第二步：

服务收口（尽量支持分业务平滑过渡）



第三步：

旧系统下线，完成统一



二、经纬度检索，从“快”到“合理”

为什么司机/用户要上报经纬度，检索经纬度?

初期，如何快速实现?

- 如何简单实现位置上报 + 位置搜索

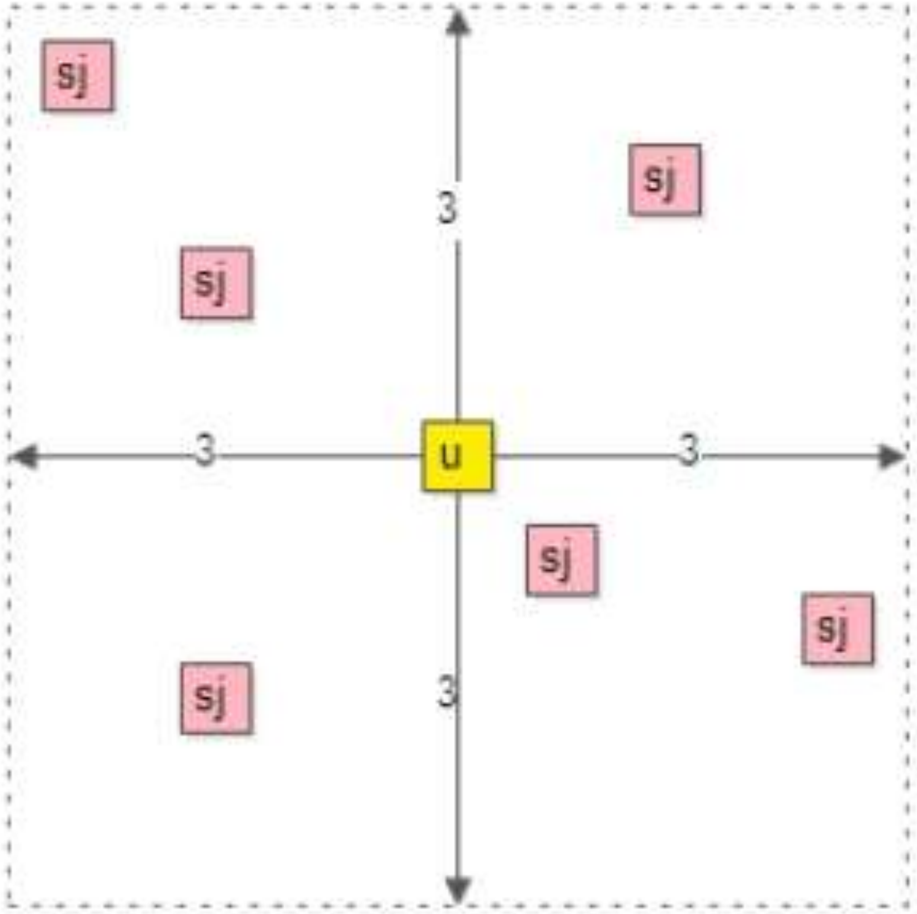
  - 数据库存储

  - 司机更新：每隔2s上报更新

  - 用户查询：上报自己，去数据库查询

driver(sj_uid, jingdu, weidu)

```
SELECT sj_uid FROM driver WHERE
      (jingdu > $jd - 3) AND (jingdu < $jd + 3) AND
      (weidu > $wd - 3) AND (weidu < $wd + 3)
```



当有100w在线司机之后呢?

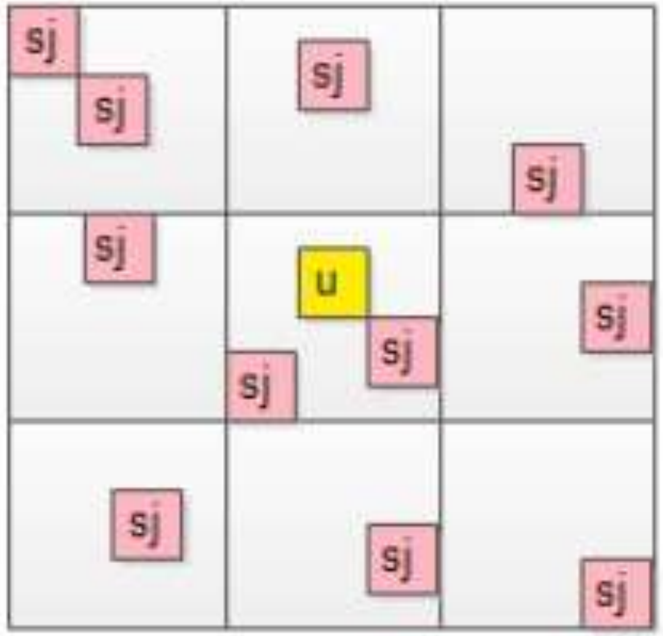
降压提效是合理的，问题转化为，怎么降压提效?

瓶颈一，数据库写压力大

- 数据库**写压力暴增**，50WQPS
- **大量无效写入**，很多写入的数据没有机会被读取，写入数据利用率低
- **如何快速优化?**
  - 数据库降压，用缓存抗写压力
  - 压力控制，异步线程写入

瓶颈二，数据库查询性能低

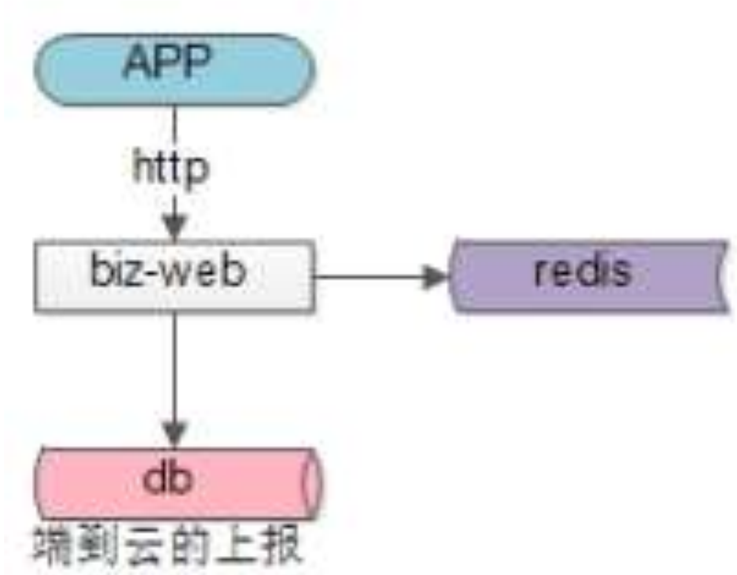
- 查询效率低，O(lg(n))
- **短期，如何快速优化?**
  - 正排 sj_uid => area_id
  - 倒排 area_id => set<sj_uid>
  - 时间复杂度： O(1)
- **中长期，让专业的软件干专业的事情：**
  - DB：元数据存储与索引
  - Elastic Search：经纬度的更新与检索



三、消息中心，从“快”到“合理”

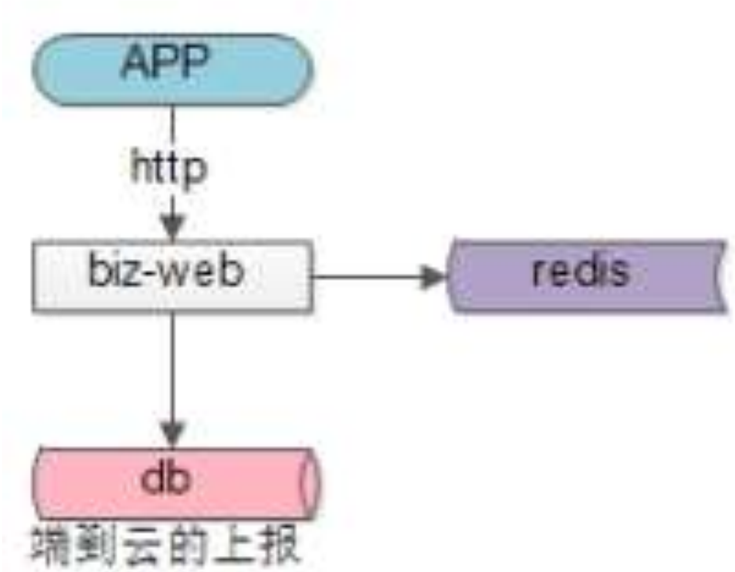
第一个场景：**端到云**

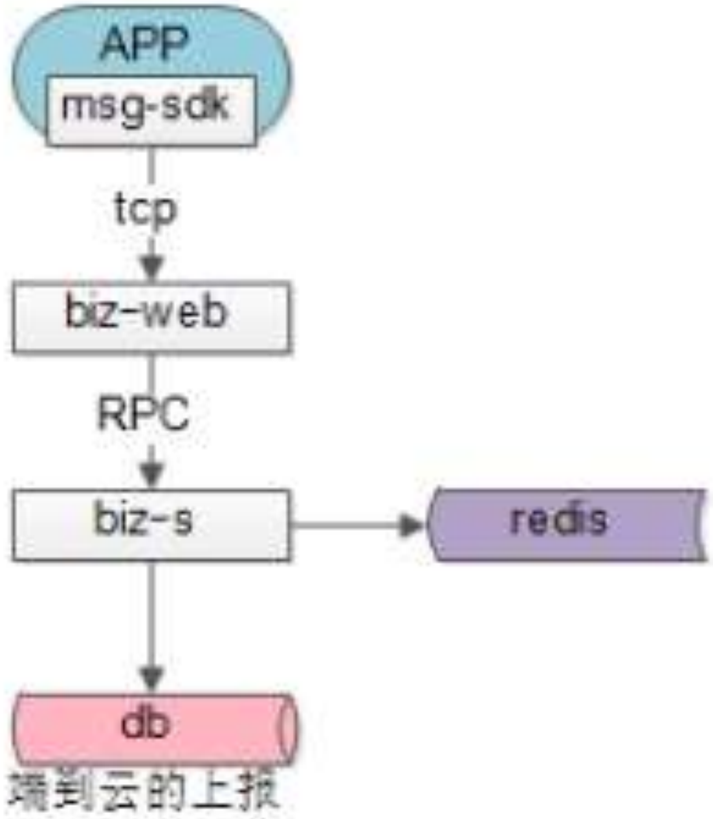
GPS上报，快速实现



第一个场景：**端到云**

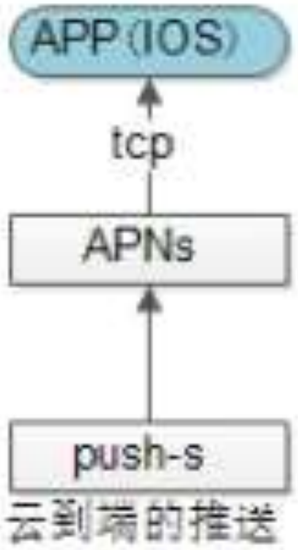
GPS上报，优化实现



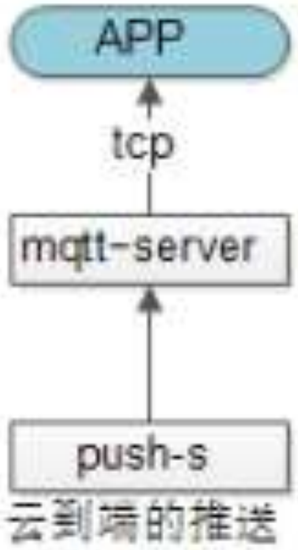


第二个场景：**云到端**

订单推送，快速实现



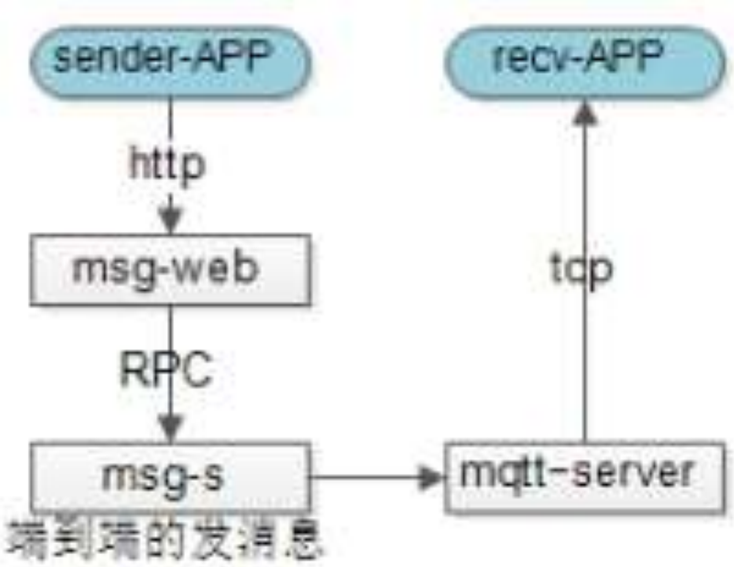
云到端的推送



云到端的推送

第三个场景：**端到端**

司乘聊天，快速实现



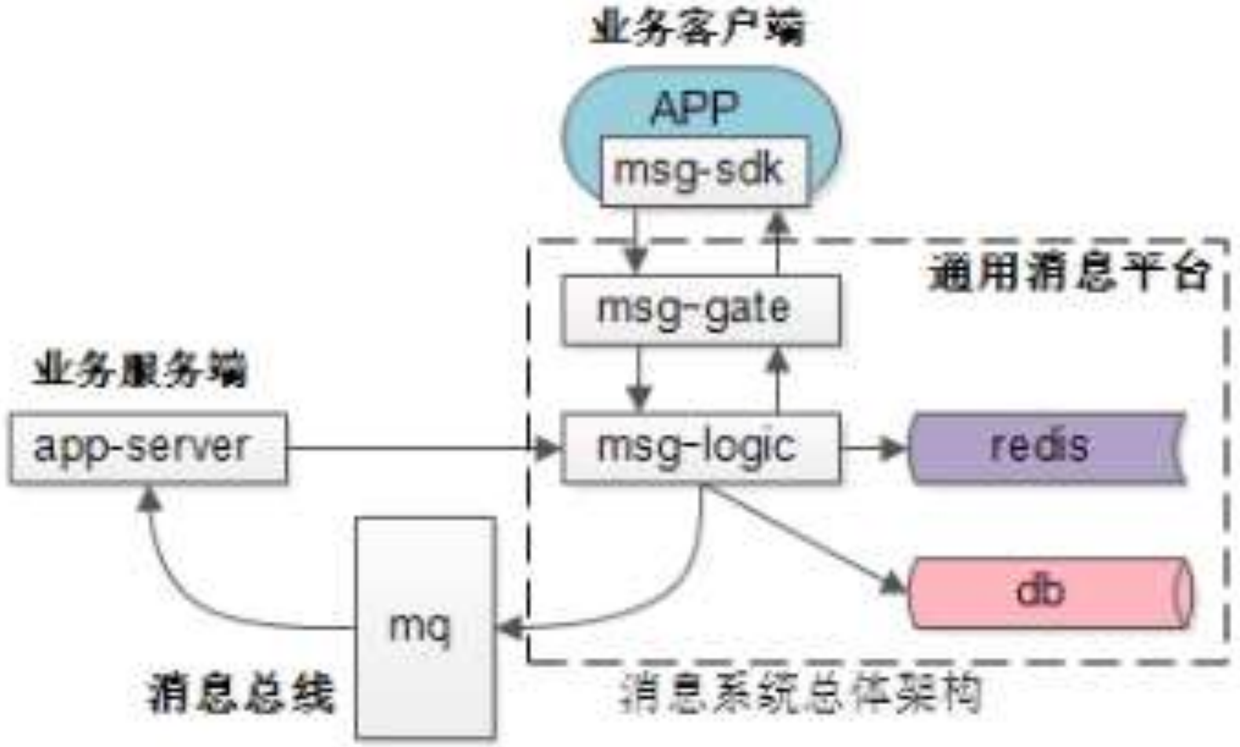
每个场景都要建立业务各自的消息通道?

能不能抽象?

抽象是合理的，问题转化为，怎么抽象？

消息中心抽象与统一

- **模块简述**：gate + logic + redis + db
- **能力简述**：c2s + s2c + c2c
- **流程简述**：c2s + s2c + c2c
- **效果**：
  - 让业务关注业务
  - 让技术关注技术，高可用，扩展，可达性



四、策略中心，从“快”到“合理”

有些什么策略?

捞取策略，推送策略，指派策略，中单策略

初期，如何快速实现？

- 串行，按照逻辑往下走即可

下单 => 写入订单表

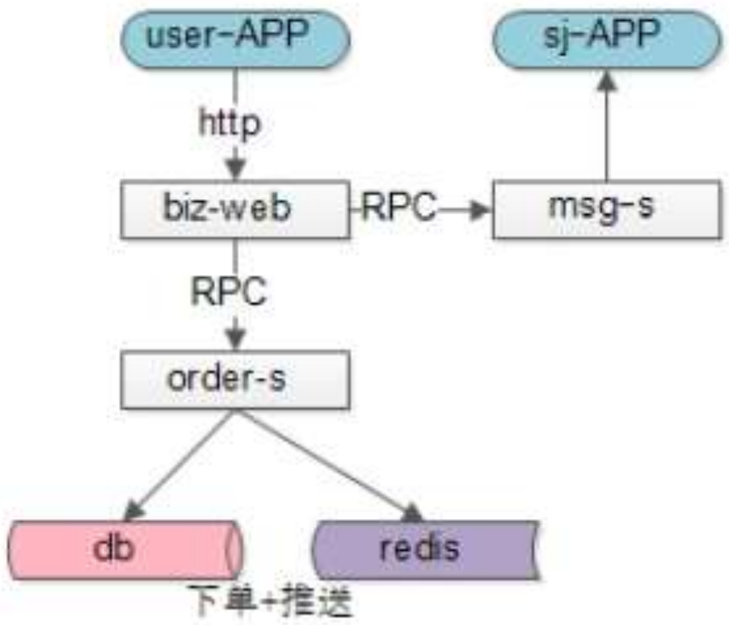
捞取策略 => **指派策略**(if 指派)

for(N个司机){

　　补贴策略 => **推送策略** => 消息推送

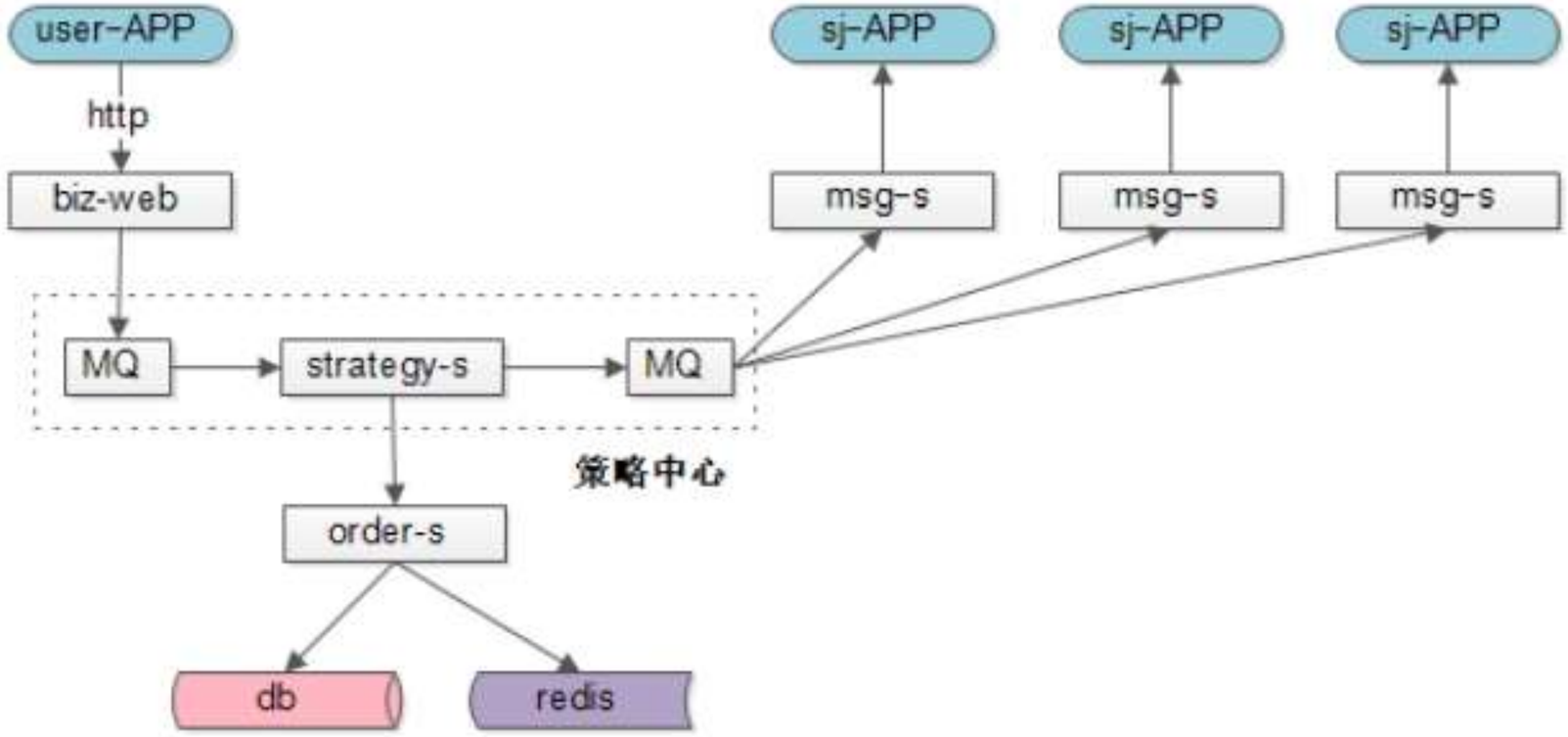
}

抢单策略 => **中单策略** => 消息推送



初期，如何快速优化?

- 串行，存在什么问题?
  - 用户等待时间长
  - 司机等待时间长
  - 串行，效率低
- 如何解耦?
  - MQ，下单与策略执行解耦
  - MQ，推送异步，并行



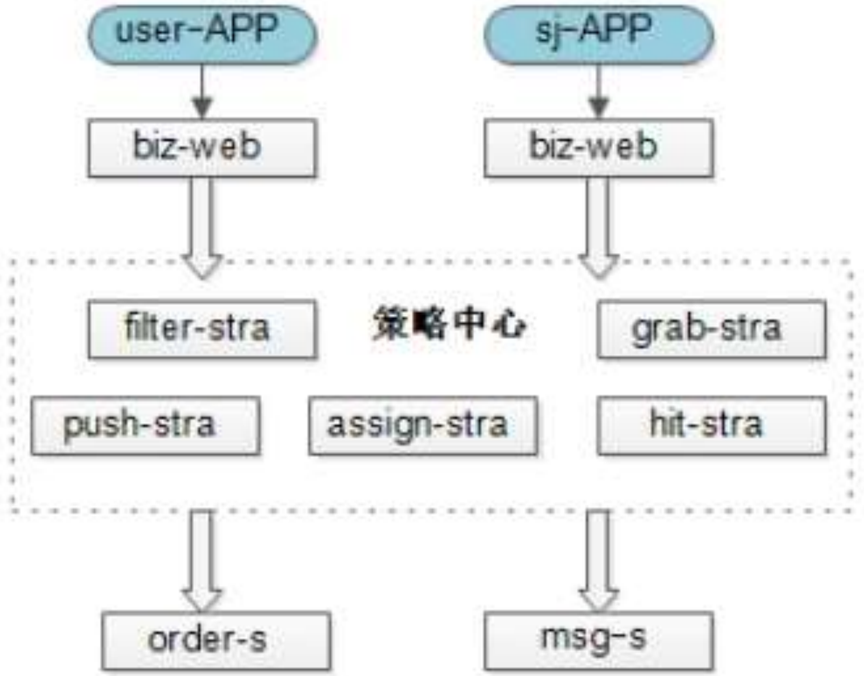
可当每个策略有N种，且迭代频繁呢?

都耦合在一个服务里?

解耦是合理的，问题转化为，怎么解耦？

策略中心，拆分解耦聚焦

- 拆分策略服务，专注于特定的策略
- 状态机，决定策略服务路由
- 标签，组合策略效果测试



总结 - 架构启示

- **指导方向**

（1）先实现，再优化

（2）先快速优化，再从长计议

一、订单中心，分分合合启示

（1）早期各自为战，闭环，快

（2）长期订单中心要统一

（3）由分到合怎么过渡?

- 先数据收口
- 再服务收口
- 最后旧系统下线

二、经纬度检索，降压启示

（1）早期快速实现

（2）数据库降压，缓存是利器

（3）异步批量写入，控制数据库压力

（4）倒排，检索复杂度O(1)

（5）让专业的软件干专业的事情

三、消息中心，抽象启示

（1）早期多场景各自为战，高效

（2）后期分离抽象，业务系统与技术系统分离

四、策略中心，解耦启示

（1）早期串行实现，快

（2）MQ，解耦利器

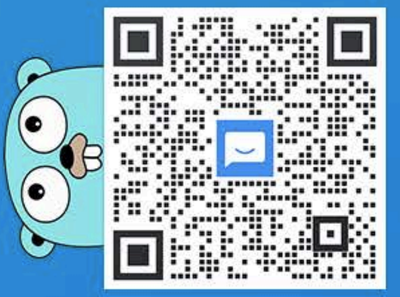
（3）服务拆分，职责专注，策略解耦

任何脱离业务，脱离业务阶段的架构设计，都是耍流氓

百万司机在线打车平台架构演进

扫描二维码 提交议题反馈

抢占先机，成为未来 3 年抢手的后端开发人才



扫码获取详细大纲
并咨询课程详情

【Go 进阶训练营】带你成为字节跳动 2-2 级别 Go 工程师

- 12 大模块教学
- 3 大企业级实战案例
- 13 周视频课程教学
- 大厂助教 1v1 答疑
- 班主任全程贴心督学
- 简历直推一线互联网公司