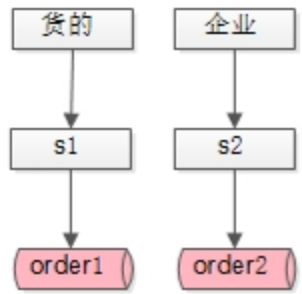
海量异构数据，在线业务存储架构演进与实践

KG沈剑

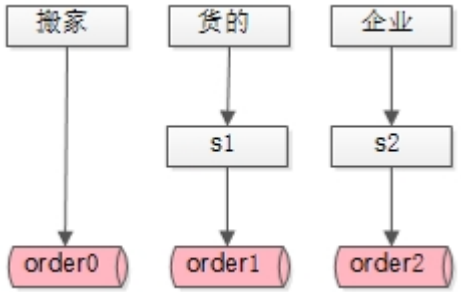
关于-我

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


搬家
order0


货的
企业
s1
s2
order1
order2

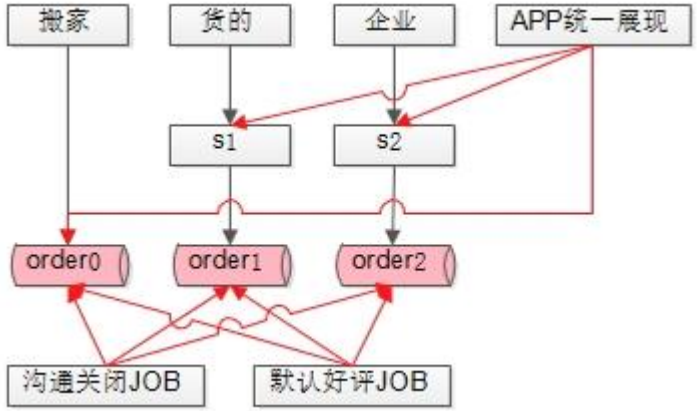
订单各自为战，存在什么问题?



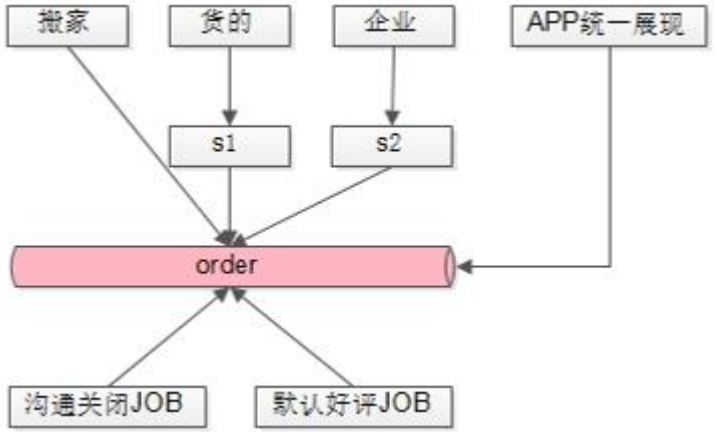
- 订单数据仓库如何统一建设?
- 订单业务对账，风控如何统一处理?
- 大数据量、高并发量、高可用、订单技术体系如何统一建设?
- 举例:
  - 用户订单列表如何实现?
  - 沟通关闭JOB如何实现?
  - 默认好评JOB如何实现?

方案一：继续折腾

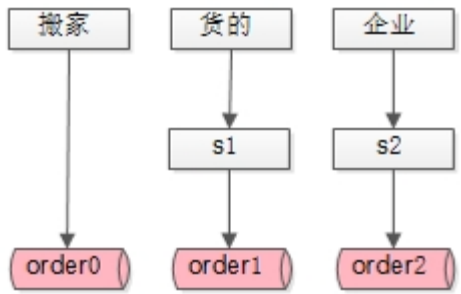
- 订单列表?
- 沟通关闭JOB?
- 默认好评JOB?

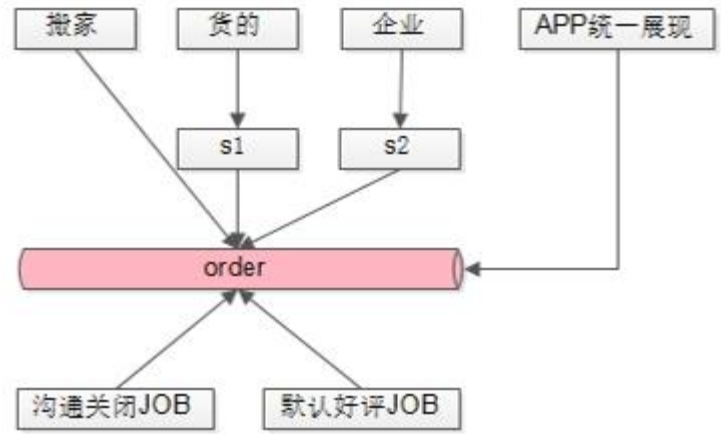


方案二：收归一统



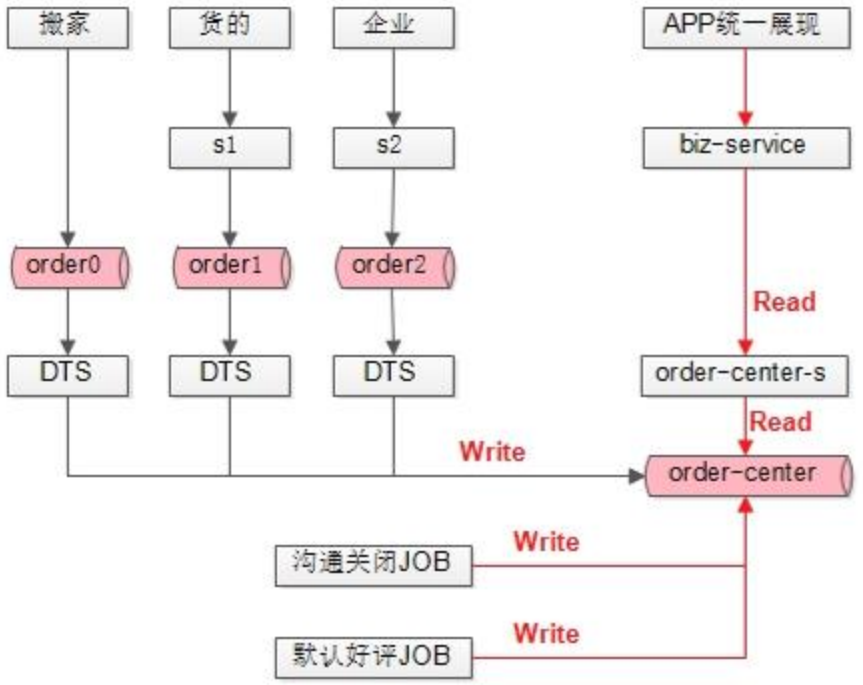
- 订单列表 -> 统一拉取
- 沟通关闭JOB -> 统一实施
- 默认好评JOB -> 统一实施
- 数据仓库统一
- 风控对账统一
- 大数据量、高并发量、高可用、技术体系统一


搬家
货的
企业
s1
s2
order0
order1
order2


搬家
货的
企业
APP统一展现
s1
s2
order
沟通关闭JOB
默认好评JOB

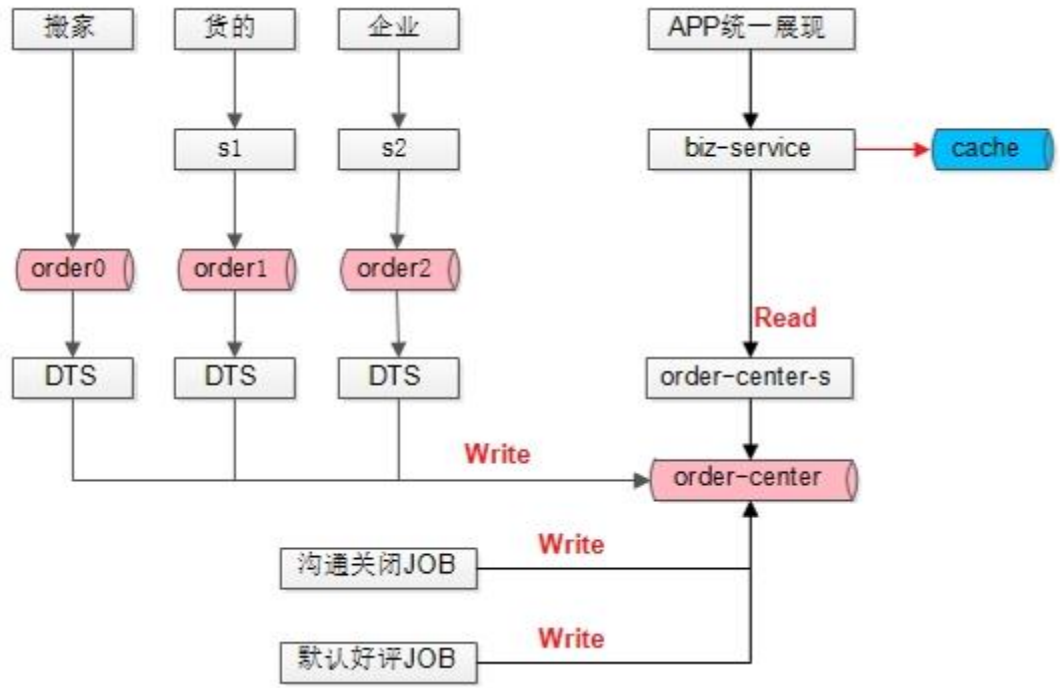
数据收口

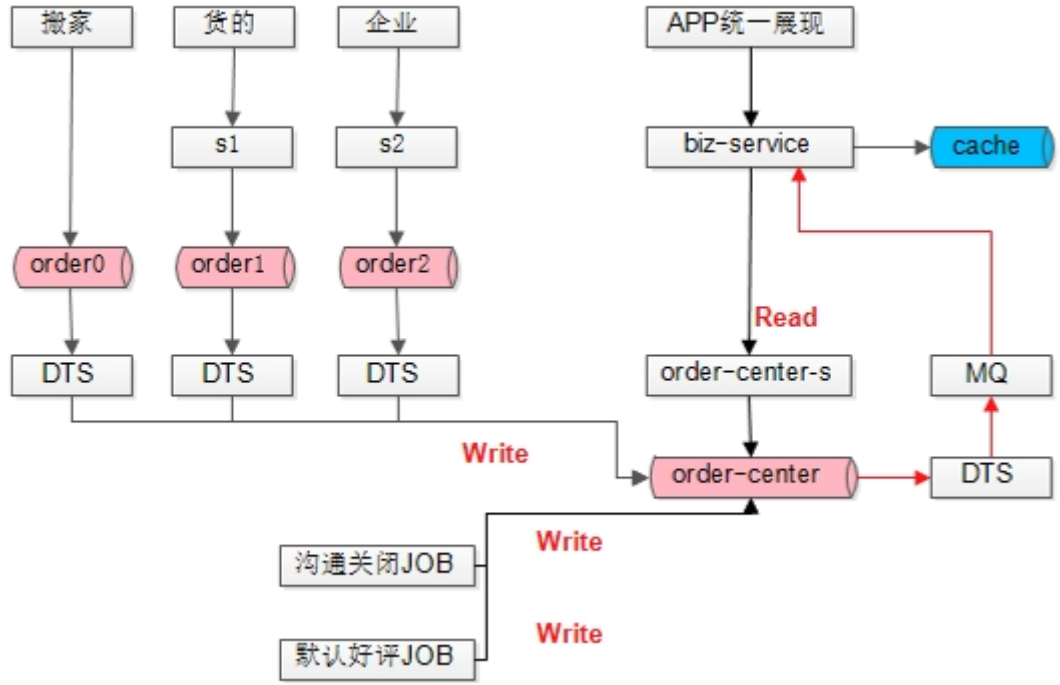
- 快速实现业务需求
  - 订单列表
  - 沟通关闭JOB
  - 默认好评JOB
- 数据收口
  - 数据仓库统一
  - 风控对账统一
- 原系统不变



小插曲，APP侧并发量增大

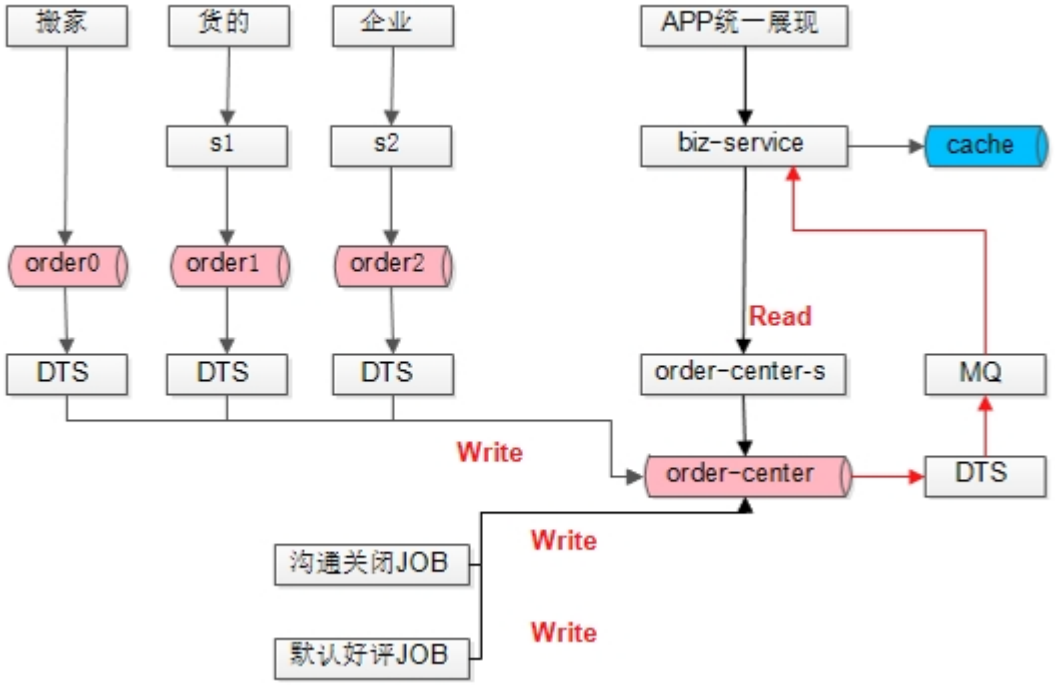
为了提升读性能，单独加了缓存

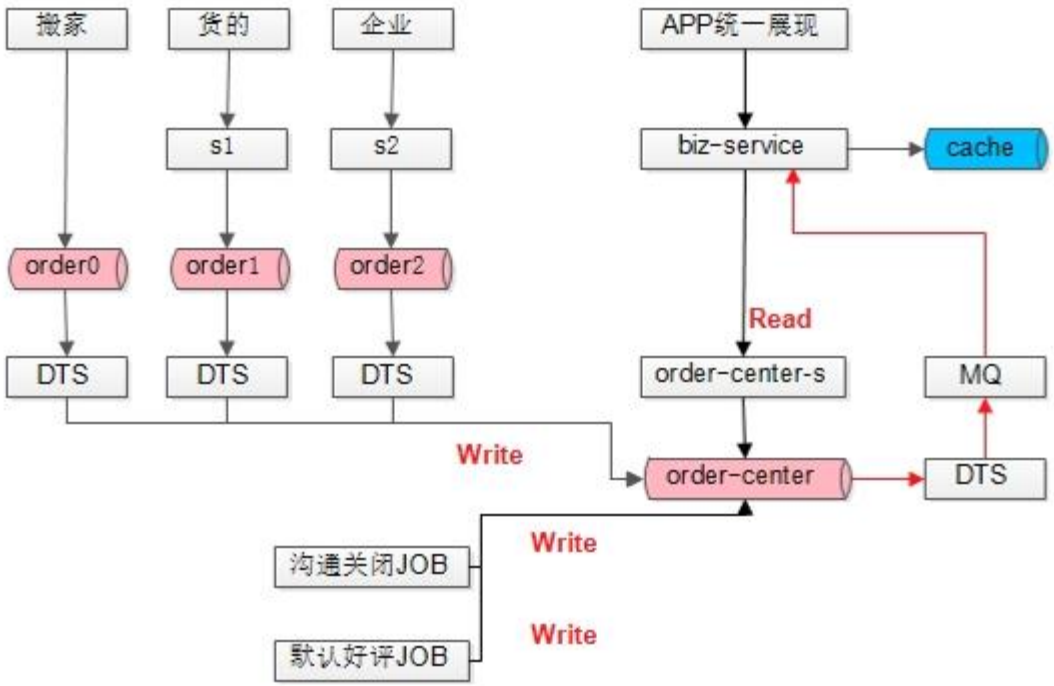

搬家
货的
企业
APP统一展现
s1
s2
biz-service
cache
order0
order1
order2
Read
DTS
DTS
DTS
order-center-s
Write
order-center
Write
沟通关闭JOB
Write
默认好评JOB

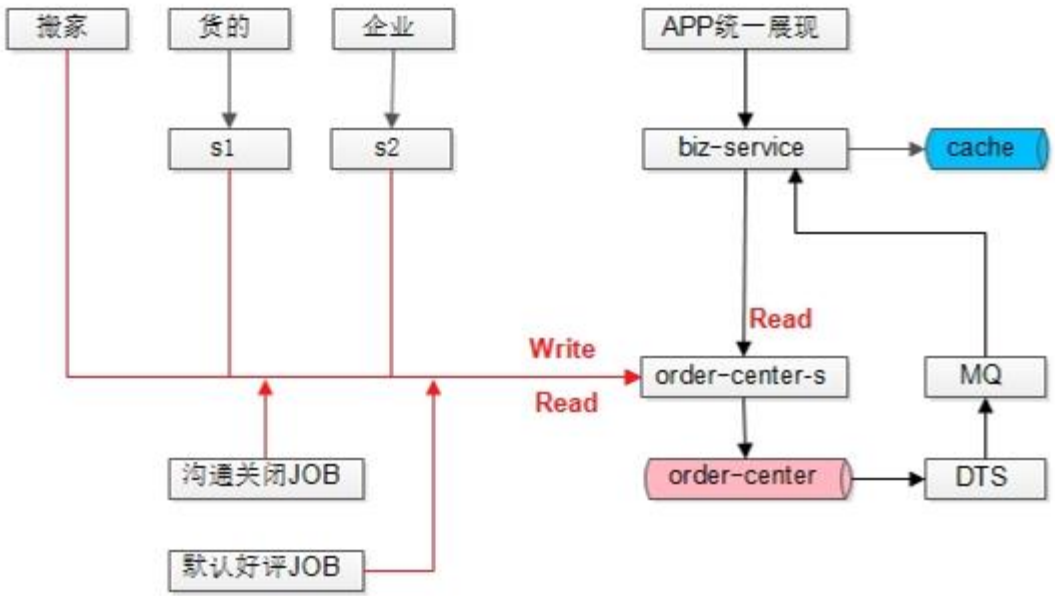

搬家
货的
企业
APP统一展现
s1
s2
biz-service
cache
order0
order1
order2
Read
DTS
DTS
DTS
order-center-s
MQ
Write
order-center
DTS
Write
沟通关闭JOB
Write
默认好评JOB

微服务大忌

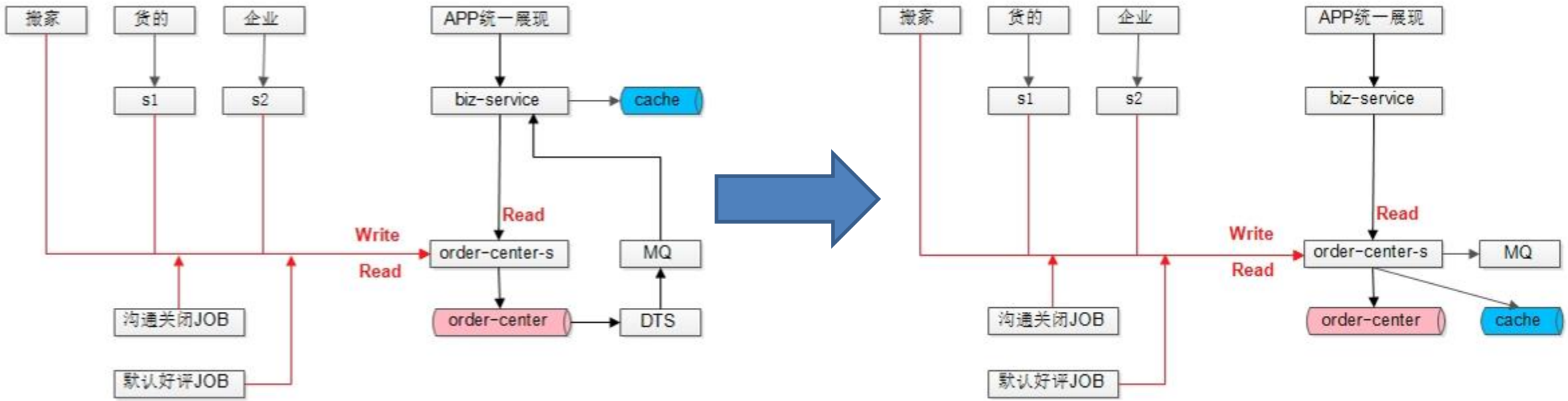
- 绕过服务直接读写数据库
  - 业务DTS直接写服务后端数据库
  - 业务JOB直接写服务后端数据库
- 无法添加缓存，屏蔽底层复杂性
  - 业务替服务加缓存，复杂性透传
  - 服务本身无法加缓存，读写不收口




搬家
货的
企业
APP统一展现
s1
s2
biz-service
cache
order0
order1
order2
Read
DTS
DTS
DTS
order-center-s
MQ
Write
order-center
DTS
Write
沟通关闭JOB
Write
默认好评JOB

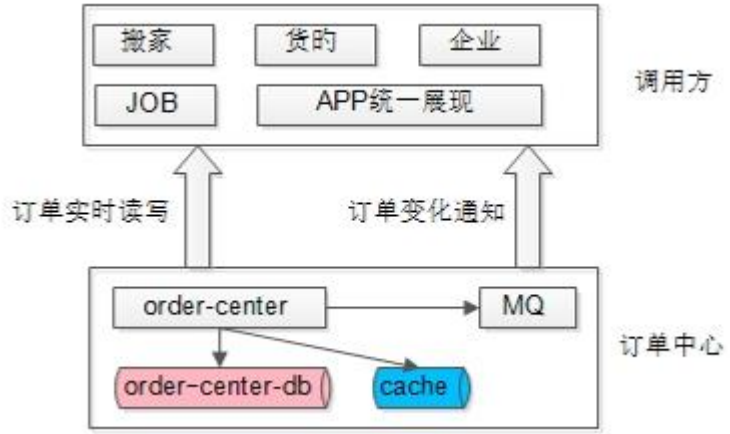

搬家
货的
企业
APP统一展现
s1
s2
biz-service
cache
Read
Write
Read
order-center-s
MQ
沟通关闭JOB
order-center
DTS
默认好评JOB

当读写收口到订单中心服务后...


搬家
货的
企业
APP统一展现
s1
s2
biz-service
cache
Read
Write
Read
order-center-s
MQ
沟通关闭JOB
order-center
DTS
默认好评JOB
搬家
货的
企业
APP统一展现
s1
s2
biz-service
Read
Write
Read
order-center-s
MQ
沟通关闭JOB
order-center
cache
默认好评JOB

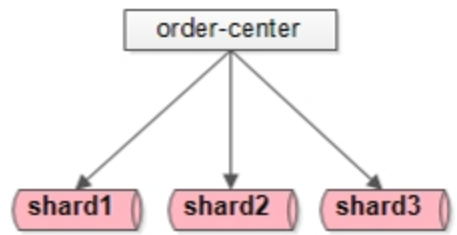
订单中心收口完成

- 职责集中
  - 提供统一读写RPC接口
  - 提供统一状态变化MQ通知
- 复杂性屏蔽
  - 屏蔽存储引擎细节
  - 屏蔽cache



接下来，数据量膨胀了...

（读量增加用cache，数据量增加怎么办？）


order-center
shard1
shard2
shard3

订单中心典型业务场景

- 点查询（90%流量）

- order_id(oid)查实体

- 列表查询（9%+流量）

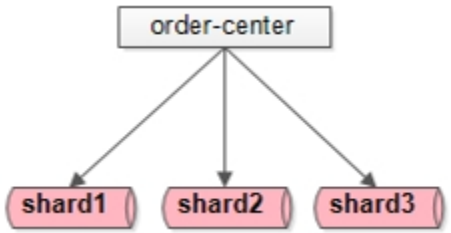
- user_id(uid)查oid_list，8%+流量

- driver_id(did)查oid_list，1%+流量

- 其他查询（1%-流量）

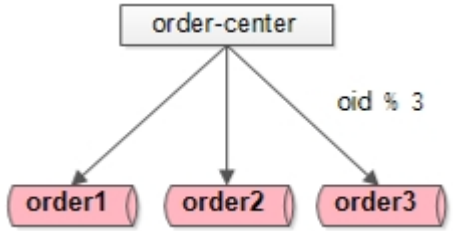
- 后台状态，金额，城市，备注，时间等各种属性

- group by, offset… limit, sum/avg等各种需求



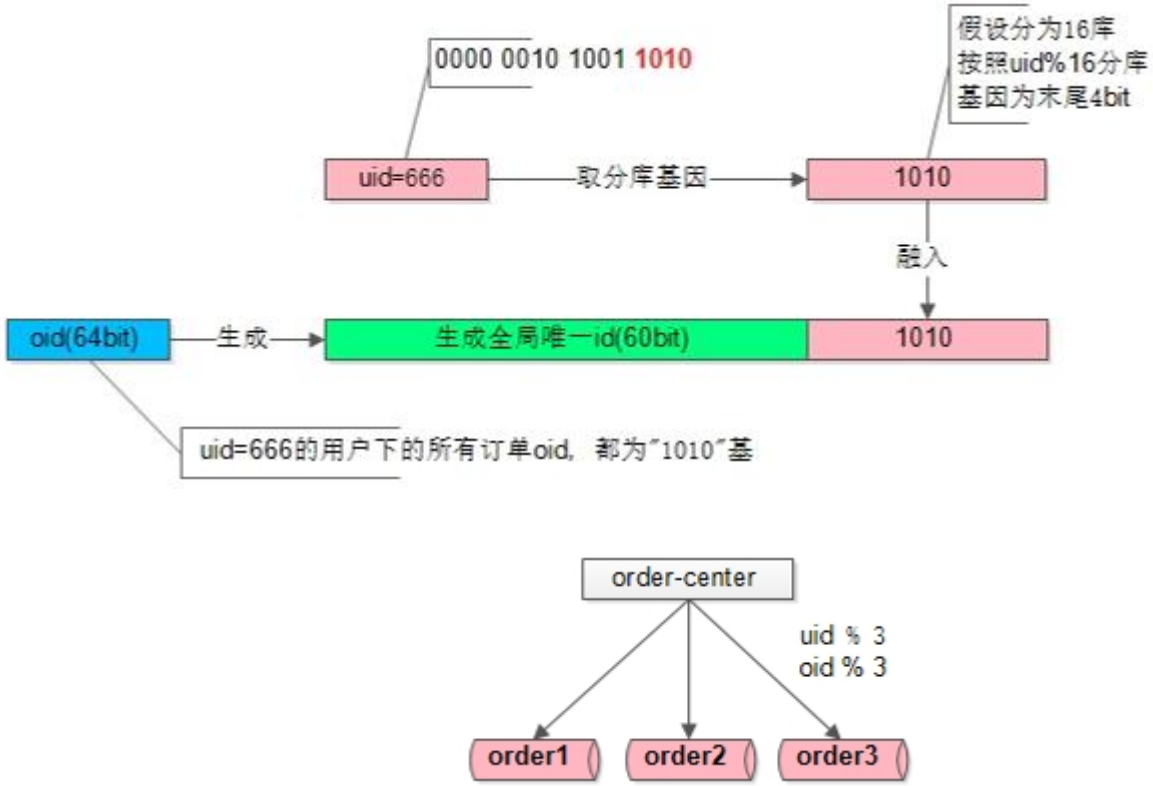
常见玩法，订单中心用订单ID水平切分

- 点查询（90%流量）
  - order_id(oid)查实体
- 列表查询（9%+流量）
  - user_id(uid)查oid_list，8%+流量
  - driver_id(did)查oid_list，1%+流量
- 其他查询（1%-流量）
  - 后台状态，金额，城市，备注，时间等各种属性
  - group by, offset… limit, sum/avg等各种需求



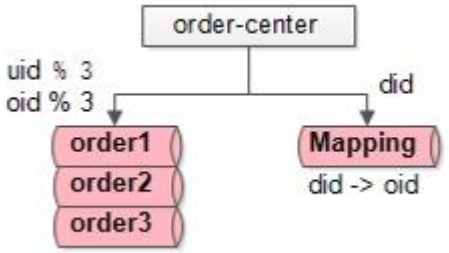
实践一，基因法，用户ID切分

- 点查询（90%流量）
  - order_id(oid)查实体
- 列表查询（9%+流量）
  - user_id(uid)查oid_list，8%+流量
  - driver_id(did)查oid_list，1%+流量
- 其他查询（1%-流量）
  - 后台状态，金额，城市，备注，时间等各种属性
  - group by, offset… limit, sum/avg等各种需求



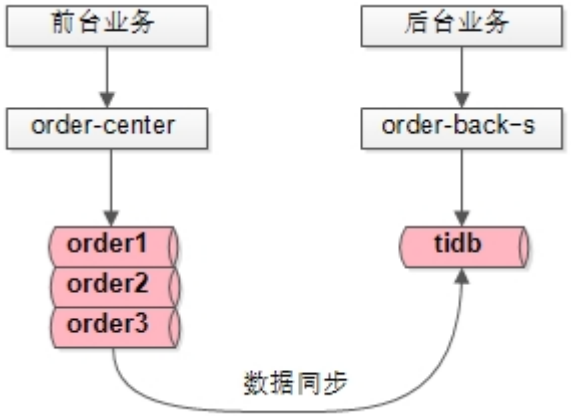
实践二，数据冗余

- 点查询（90%流量）
  - order_id(oid)查实体
- 列表查询（9%+流量）
  - user_id(uid)查oid_list，8%+流量
  - driver_id(did)查oid_list，1%+流量
- 其他查询（1%-流量）
  - 后台状态，金额，城市，备注，时间等各种属性
  - group by, offset… limit, sum/avg等各种需求



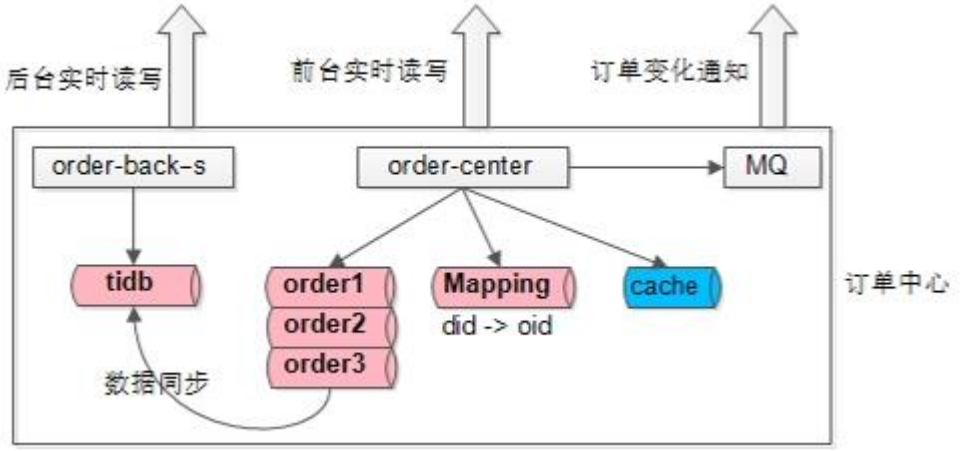
实践三，前台与后台分离

- 后台查询（1%-流量）
- 前台访问，后台访问场景差异
  - 访问模式
  - 访问量
  - 时延敏感性
  - 可用性要求
  - 一致性要求
- 资源解耦



总结

- 早期业务快速迭代，订单库，分
- 中长期看，订单中心，合
  - 数据建设，数仓统一
  - 风控对账，业务统一
  - 大数据量、高并发量、高可用、技术体系统一
- 从分到合，重在**平滑**
  - 数据收口，原系统不变
  - 服务收口，原系统蚂蚁搬家迁移



- 微服务分层，两点原则
  - 任何上游不得绕过服务读写数据库
  - 微服务对上游屏蔽存储引擎，分库分表，缓存等复杂性
- 三点实践
  - 水平切分，基因法，点查+列表查询均一步到位
  - 数据冗余，Mapping表解决多维度列表查询
  - 前台与后台分离，解耦，不同技术方案解决不同业务场景

Q&A

谢谢！