

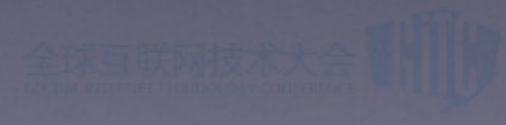
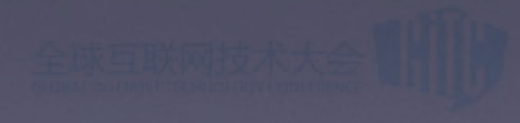
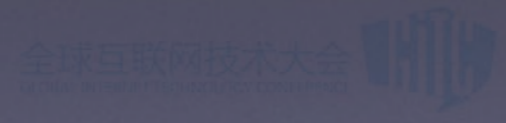
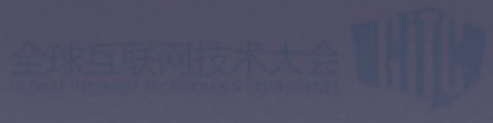
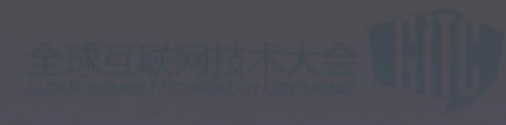
饿了么全链路压测的探索与实践

饿了么

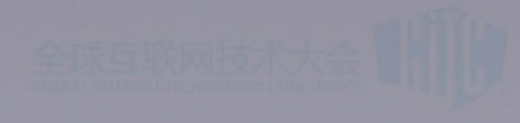
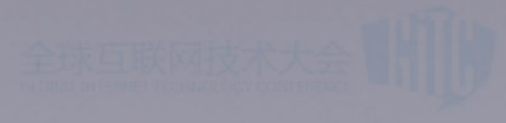
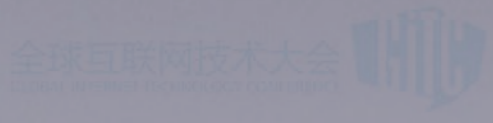
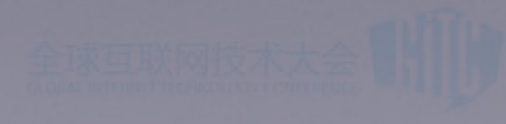
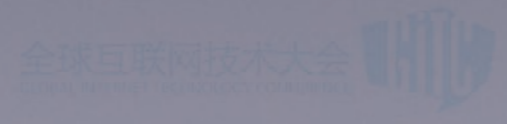
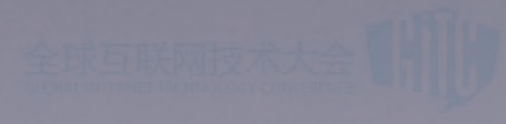
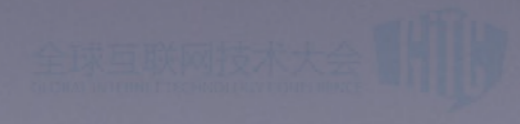
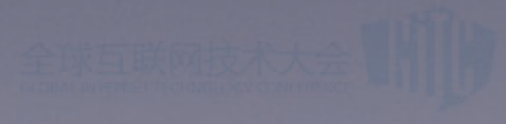
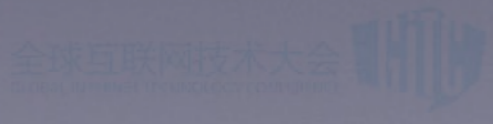
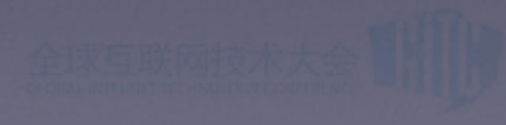
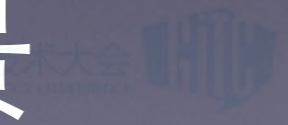
— 费翔

提纲

- 背景
- 辛酸历程
- 实施
- 后续规划



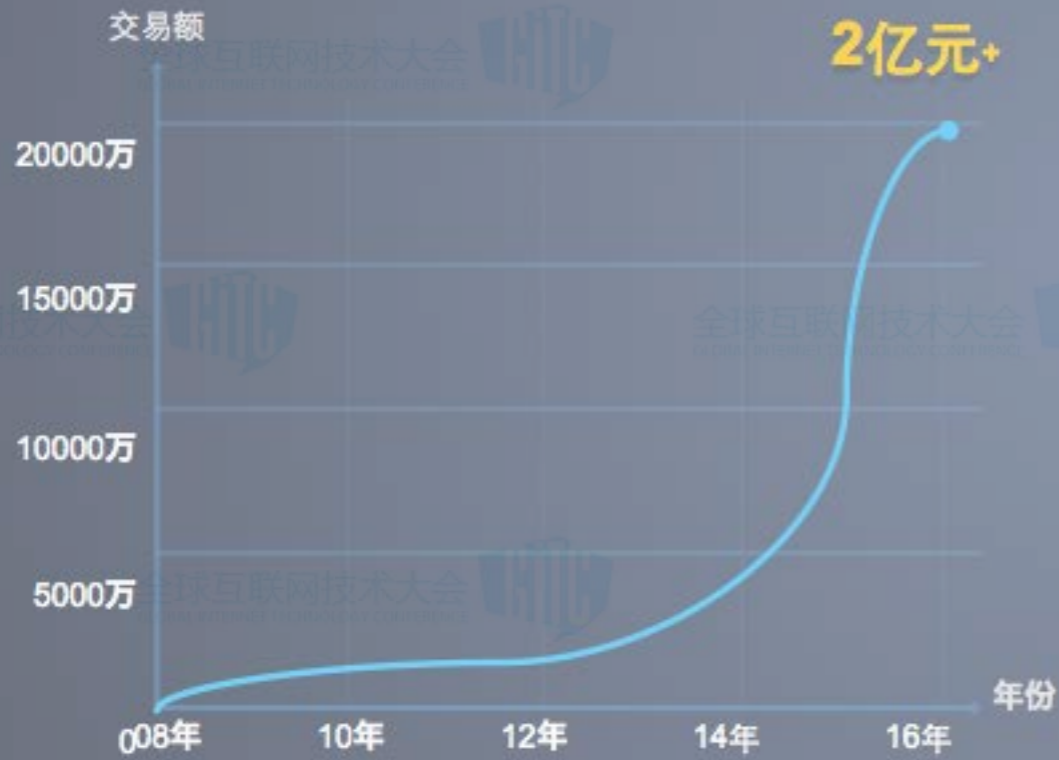
背景



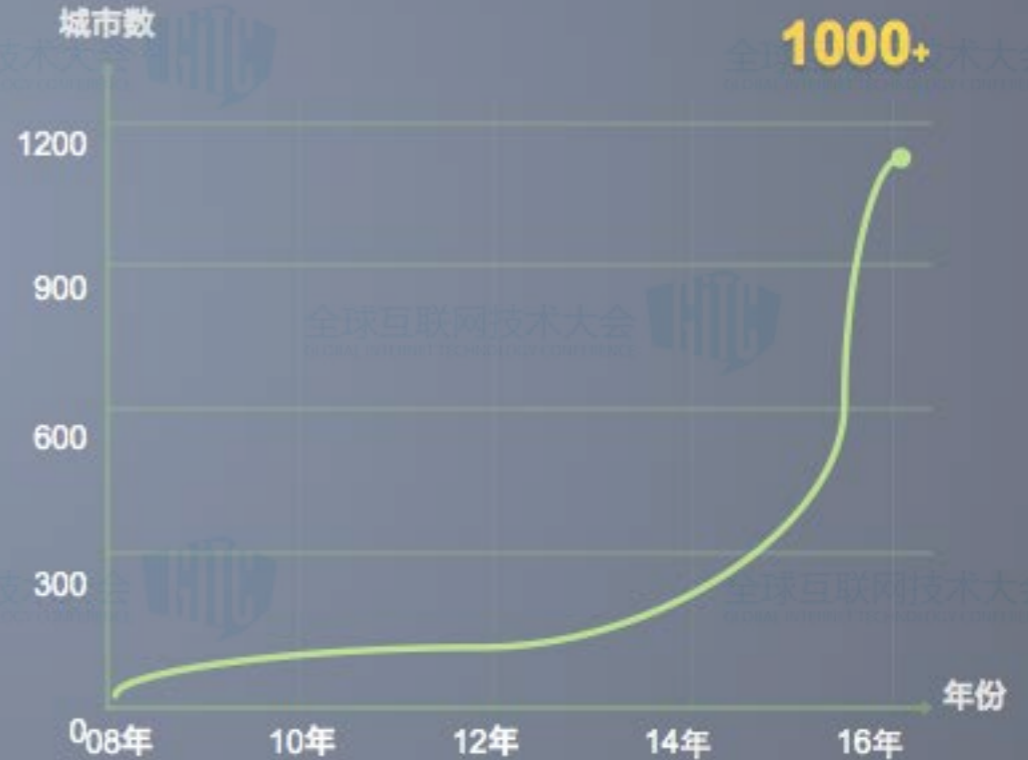
- 爆炸式增长



日均交易额

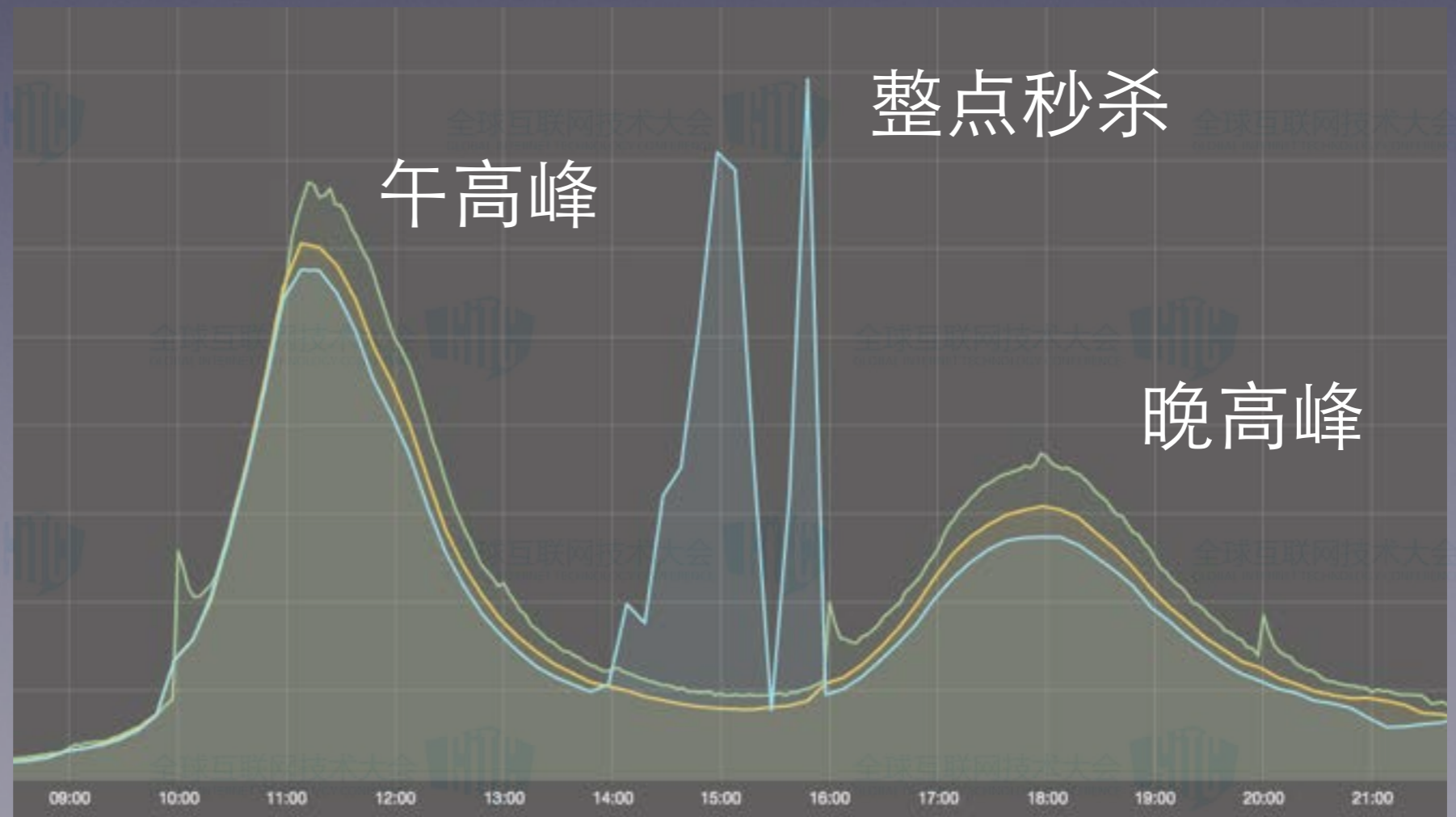


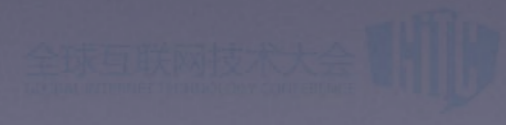
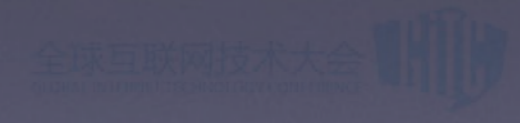
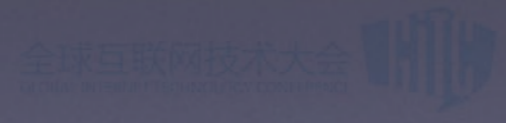
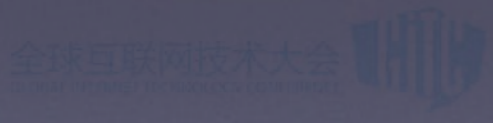
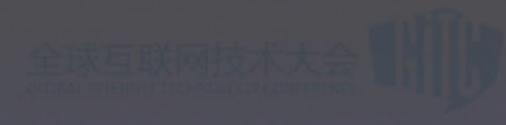
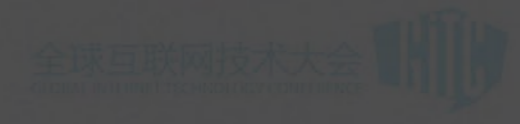
覆盖城市



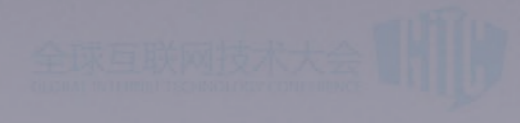
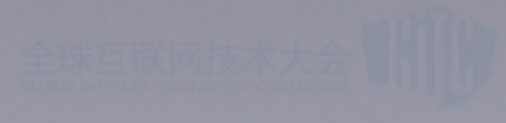
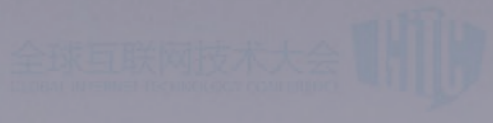
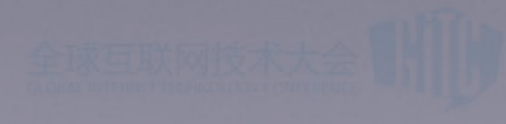
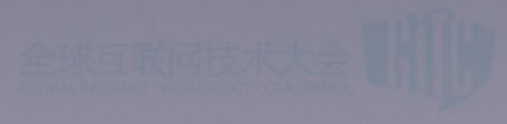
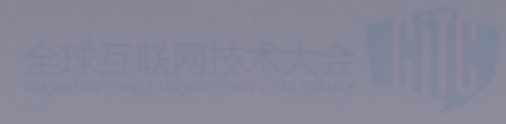
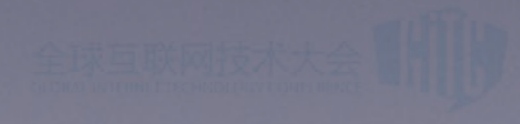
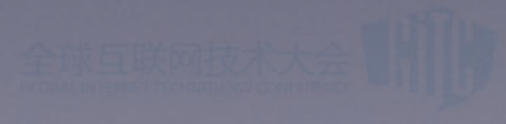
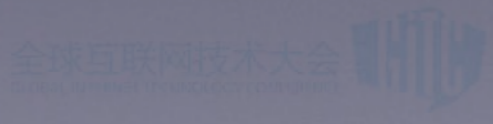
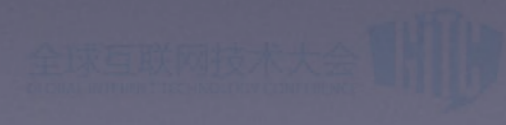
• 业务的特点

- 时效性（用户、商户、物流等）
- 高并发
- 瞬时冲击
- 秒杀活动
- 常规化





辛酸历程



- 三阶段（线上环境实施）

- ◆ 缩减服务器

- ◆ 单独业务压测

- ◆ 全链路压测

• 缩减服务器

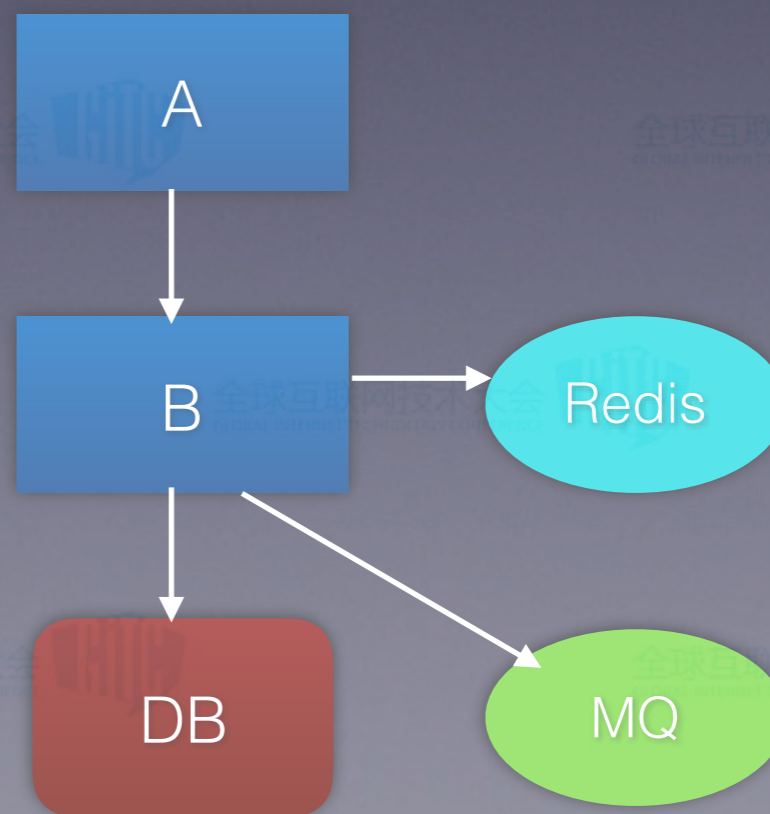
在低峰时间段，用减少服务器数量的方式来评测线上服务器的极限容量。通过对比低峰时间段和高峰时间段的请求量，预估将来订单量增长的情况下，大致需要的服务器数量。

好处：

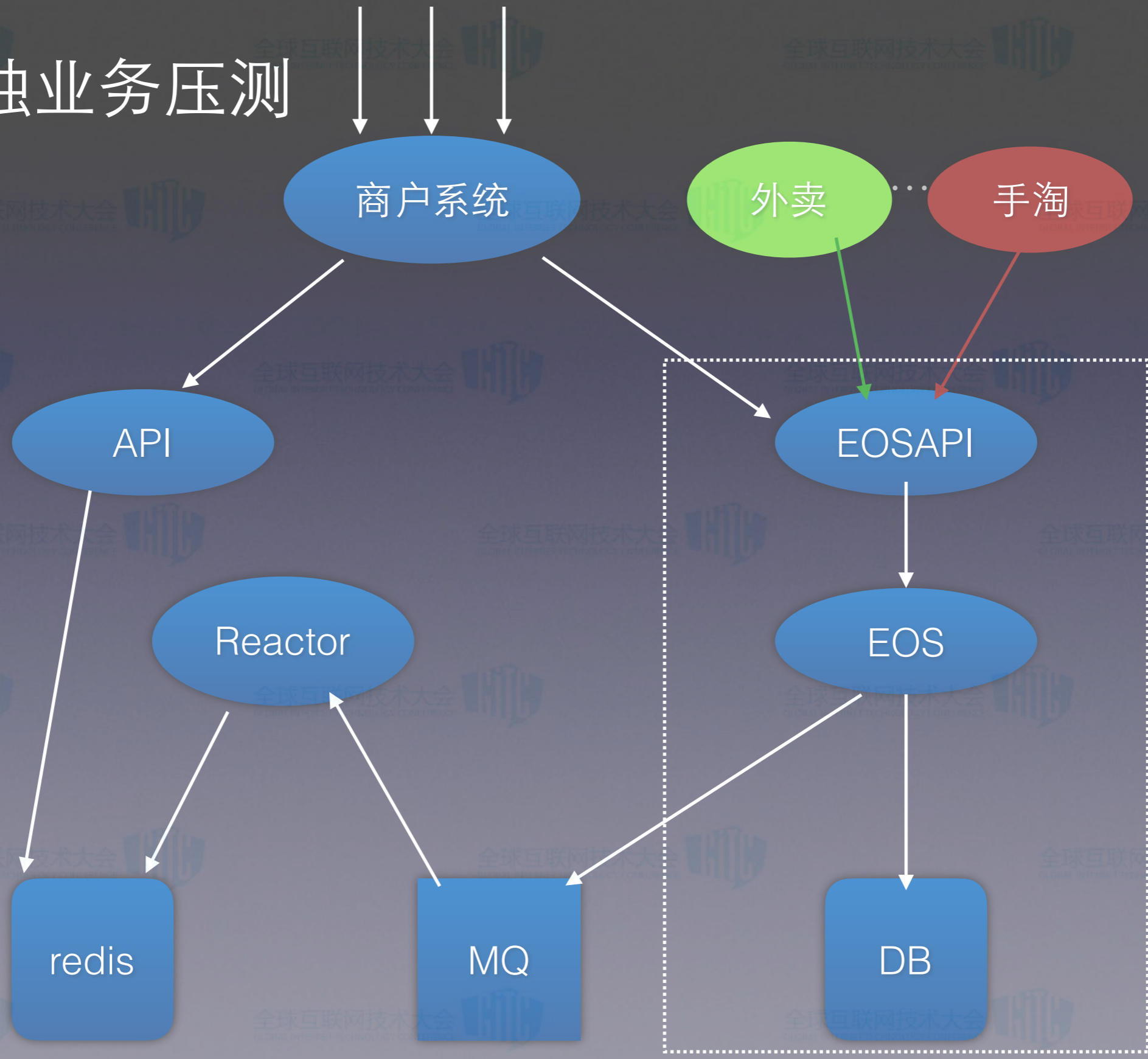
- 无脏数据
- 真实流量

问题：

- 底层容量不易评估
- 容量评估不精准



- 单独业务压测



• 全链路线上压测

模拟外卖平台下单，开放平台下单(如手淘)，商户接单，物流派送；

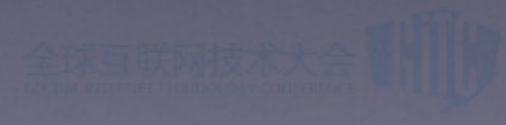
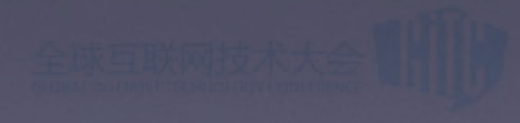
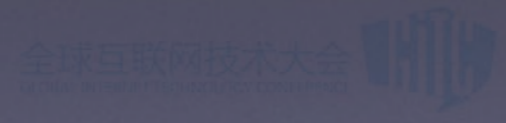
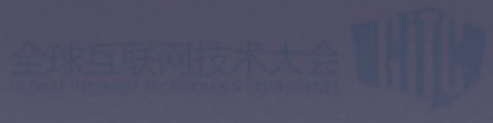
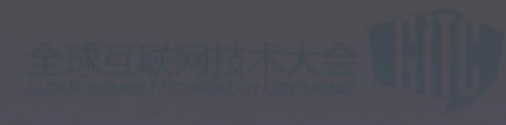
好处：

1. 容量评估相对精准，TPS为导向
2. 覆盖所有关键路径接口
3. 底层服务性能及容量易评估

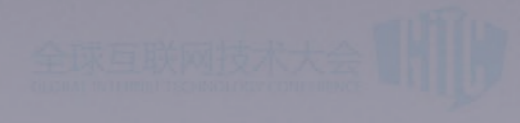
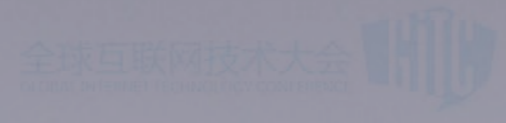
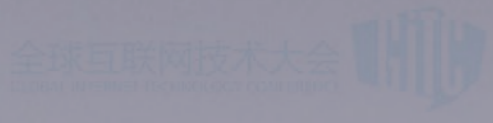
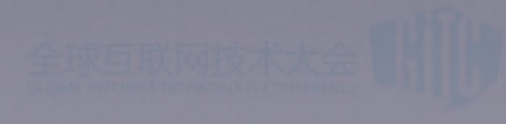
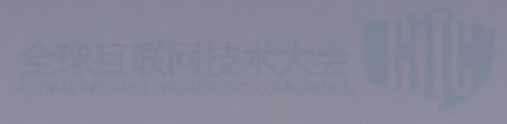
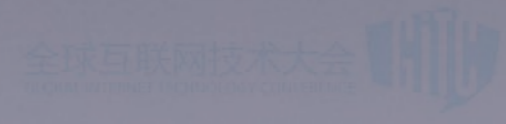
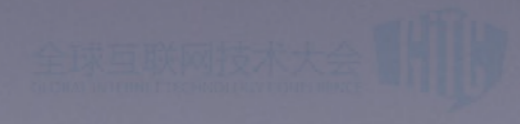
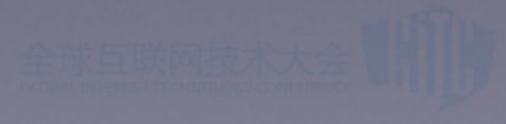
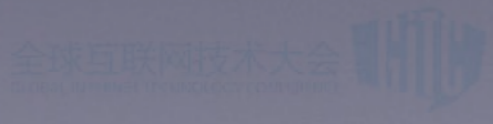
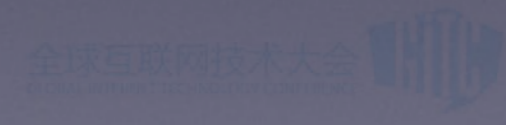
问题：

1. 脏数据的处理



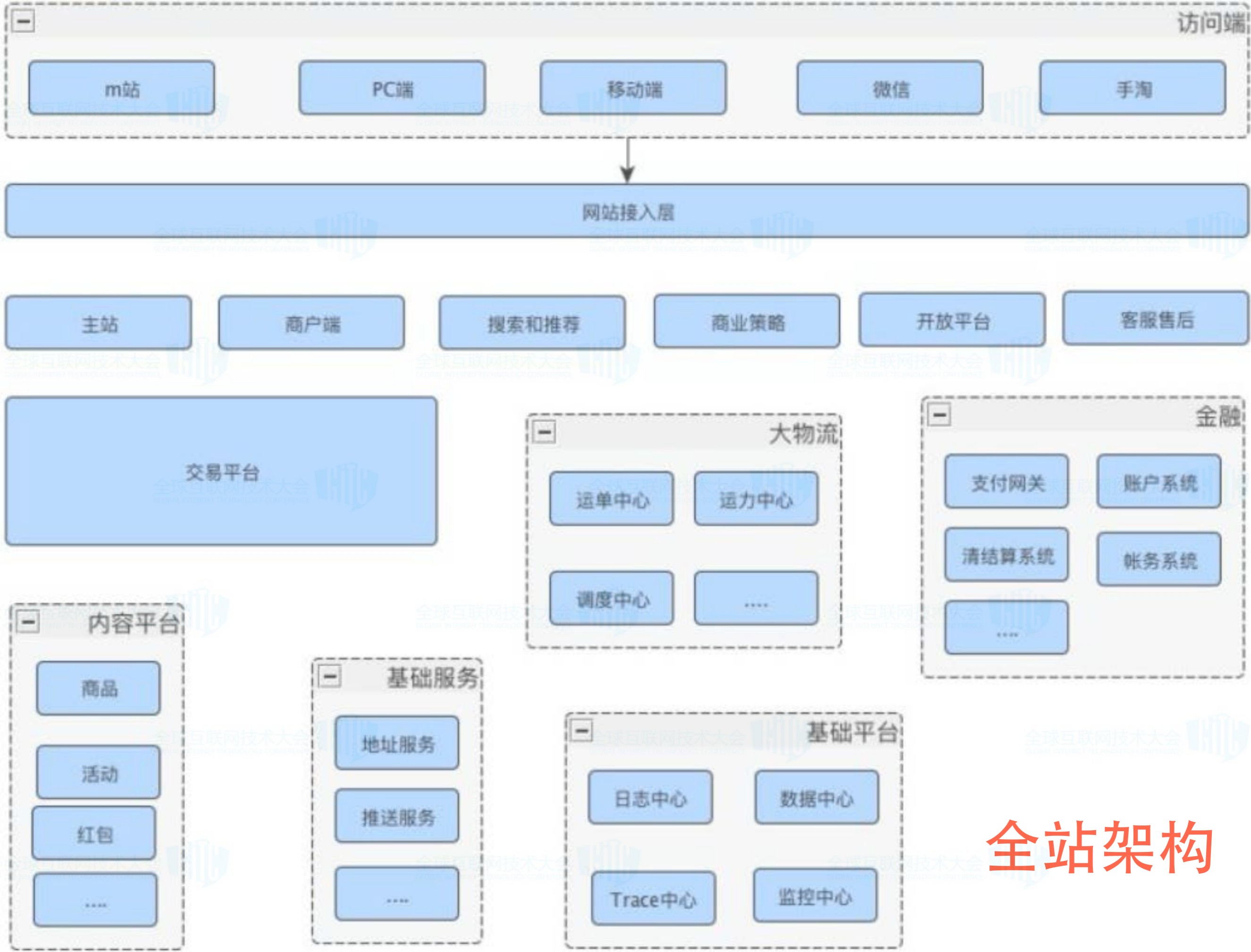


实施



- 业务模型的梳理
- 数据模型的准备
- 工具的选型
- 测试脚本
- 性能指标的监控和收集
- 测试报告

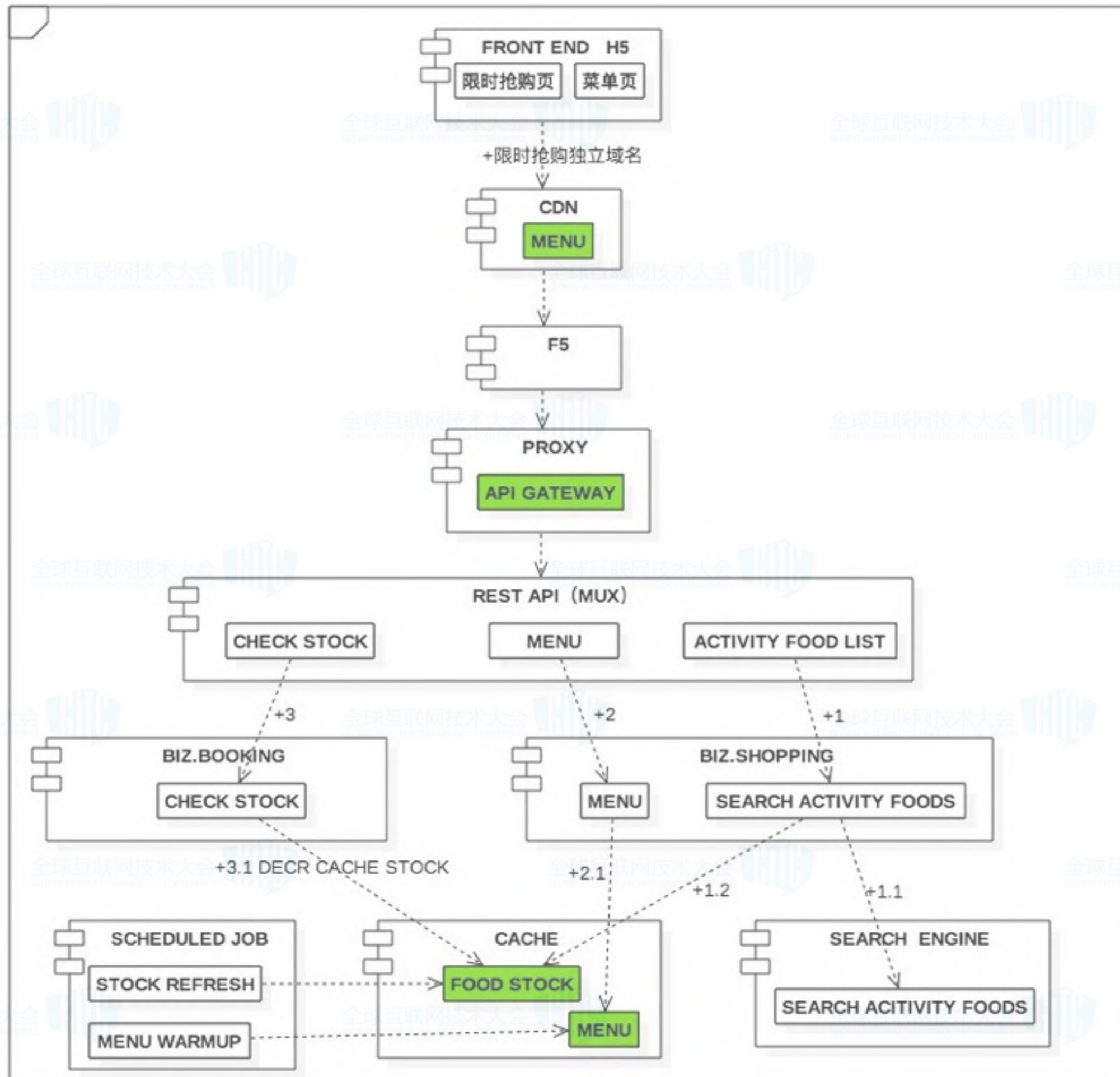
- 业务模型的梳理
 - ◆ 结合业务和架构



全站架构

秒杀场景





• 业务模型的梳理

- ◆ 关键路径？ 非关键路径？
- ◆ 业务的调用关系
- ◆ 业务的接口列表
- ◆ 接口类型(http、thrift、soa等)
- ◆ 读接口？ 写接口？
- ◆ 接口之间的调用比例

• 数据模型构建（持续调整优化）

用户、商户、配送方、菜品等在数量上与线上等比例缩放

- ◆ 数据是否sharding分布均匀
- ◆ 数据的时效性，比如查询操作

原则：数据模型紧贴业务场景

• 数据准备问题导致的坑

- ◆ 用户数据未考虑sharding分布问题->db单点过热
- ◆ 用户数量过少->单个测试用户订单量过多
- ◆ 商家数量过少 ->导致菜品减库存锁争抢激烈

- 工欲善其事，必先利其器(Jmeter)

- ◆ 开源轻量级

- ◆ 方便开发插件

- ◆ 支持丰富（比如RemoteServer，设定集合点）

• 测试的节奏

- ◆ 参与方：压测测试人员、相关业务开发、架构师、运维等
- ◆ TPS（下单量）为导向，来进行测试的迭代

• 性能指标

■ 应用层面

- ① 错误率
- ② 吞吐量
- ③ 响应时间，
median, 90,
95, 99线
- ④ 内存曲线
(JVM、GC)

■ 基础资源

- ① CPU使用率和负载
- ② 磁盘I/O
- ③ 网络I/O

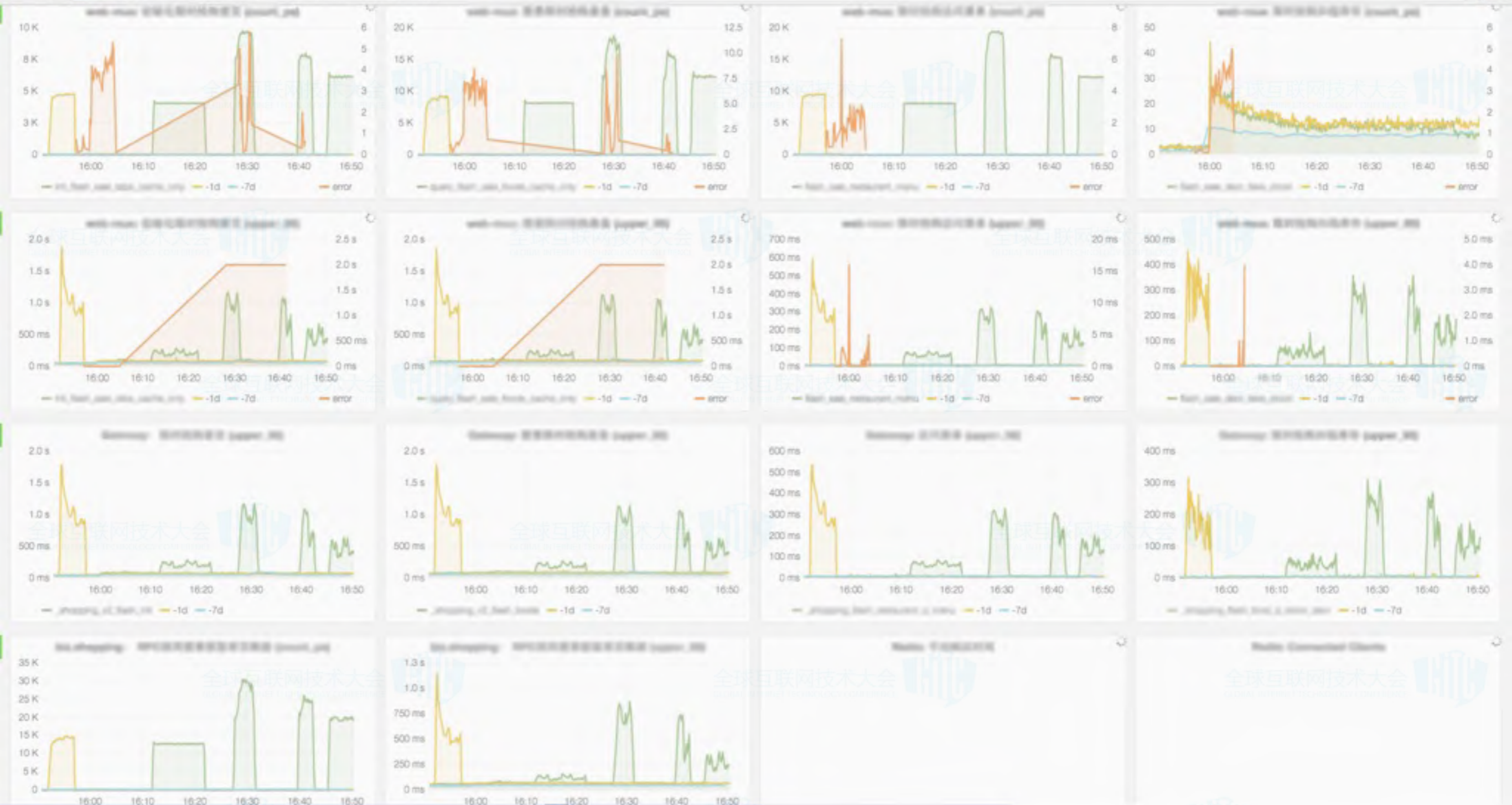
■ 基础服务

- ① MQ
- ② Redis
- ③ DB

- 响应时间不要用平均响应时间，关注95线；
- 吞吐量考虑响应时间变化；
- 吞吐量需要和成功率挂钩；

- 性能指标的分析

- ◆ 线上监控工具，grafana、ettrace等



Details

Sampling: **4.34sec** 4.29sec 3.26sec 1.16sec 836.00ms 501.00ms 439.00ms 417.00ms 399.00ms 395.00ms 385

Topology

AppId:
 HostIP:
 HostName:

Duration	Duration(%)	Description(Type:Name)
4.34sec	100.00%	URL: /getShopDeliveryProfile ⚠
2.00ms	0.05%	SOACall: KeeperService.getKeeperBySessionId 📄
1.00ms	0.02%	SOACall: KeeperService.getManagedRestaurantIds 📄
9.00ms	0.21%	SOACall: RestaurantService.getRestaurantByOid 📄
4.00ms	0.09%	SOACall: MerchantService.getShopDistProfile 📄
4.00ms	0.09%	SOACall: DeliveryService.getPShopProfile 📄
2.00ms	0.05%	SOACall: MerchantService.isCrowdProduct 📄
6.00ms	0.14%	SOACall: ShopProfileService.getShopMealTakenAddress 📄
7.00ms	0.16%	SOACall: ShopProfileService.getControlledShopProductStatus 📄
8.00ms	0.18%	SOACall: ShopProfileService.getCrowdShopProductStatus 📄
78.00ms	1.80%	SOACall: DeliveryService.getIsCrowdShopProductVisible 📄
6.00ms	0.14%	SOACall: DeliveryService.getIsOnlinePayStatus 📄
9.00ms	0.21%	SOACall: SettlementService.isOnlineSettlementSupport 📄
4.20sec	96.82%	SOACall: DeliveryService.getDeliveryTimePairs ⚠

测试报告

一、测试环境

1. 订单接口直接下单，producer直接把测试餐厅的订单挡回去，这样测试数据都走补偿线路，验证补偿线路性能，不影响线上正常下单流程
2. 测试餐厅类型为全推
3. 测试时间：[REDACTED]
4. 下单走的是老支付

二、测试结果

线程数	下单量(/sec)	平均响应时间(ms)	90%响应时间(ms)	95%响应时间(ms)	99%响应时间(ms)	最大响应时间(ms)
10	[REDACTED]	20	26	30	44	169
100	[REDACTED]	29	47	59	92	2020
120	[REDACTED]	52	96	149	350	3509

从压测数据来看，目前订单的create_order接口，吞吐量最多也就是[REDACTED]，ToD的其他指标参照DashBoard

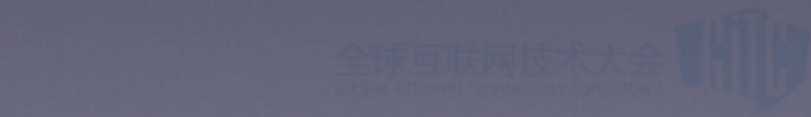
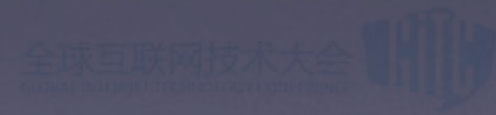
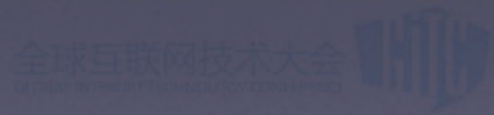
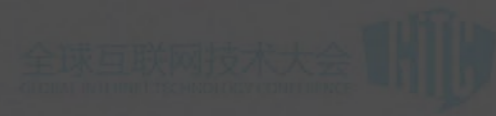
压测时保存orderid用于后续核对实际落库的orderid是否一致

线上压测时订单接口的orderid列表：

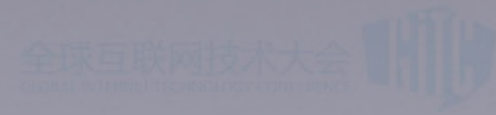
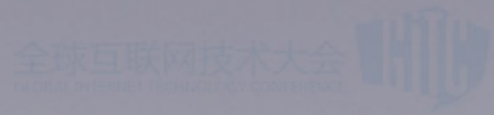
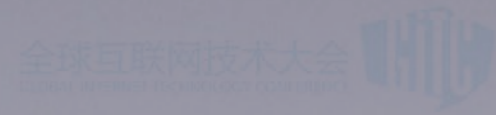
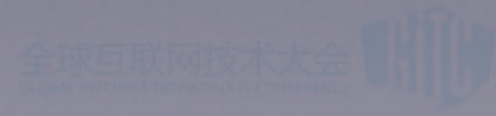
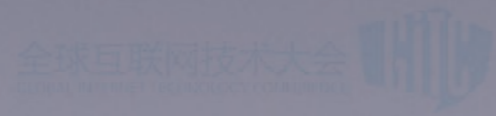
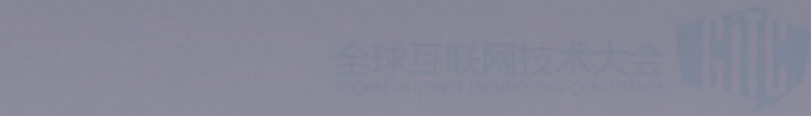
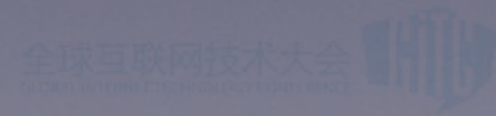
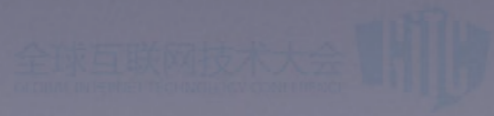
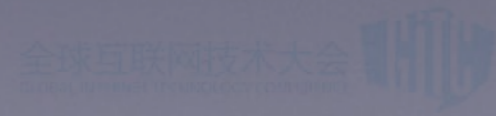
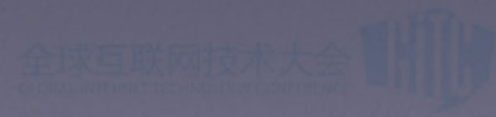
result_orderID_1_5000.txt

压测问题：

1. 备线task扩容至13台机器，压测能达到[REDACTED]，下单在[REDACTED]，推单延时10分钟
2. 备线设计中使用的redis Q压力仅在两个redis node上，此为怀疑有瓶颈，cpu load达到35左右，cpu usage最高80%
3. 主线迁移至新的ToD集群后，按[REDACTED]压测，RMQ的publish数量能达到[REDACTED]，老集群同量压测时，在publish数量在[REDACTED]左右
4. apollo备线处理能力不足，需调优解决



后续规划



- 压测平台开发



Will Fei

上海 长宁



扫一扫上面的二维码图案，加我微信



GITC性能测试讨论群



该二维码7天内(11月25日前)有效，重新进入将更新