ArchSummit
全球架构师峰会 2017

主办方 Geekbang 极客邦科技 InfoQ

腾讯企业级消息中间件DevOps之路

闫二辉
腾讯中间件专家工程师

SPEAKER INTRODUCE

闫二辉(zizayan)

腾讯中间件专家工程师，2012年加入腾讯基础架构部

主要从事：

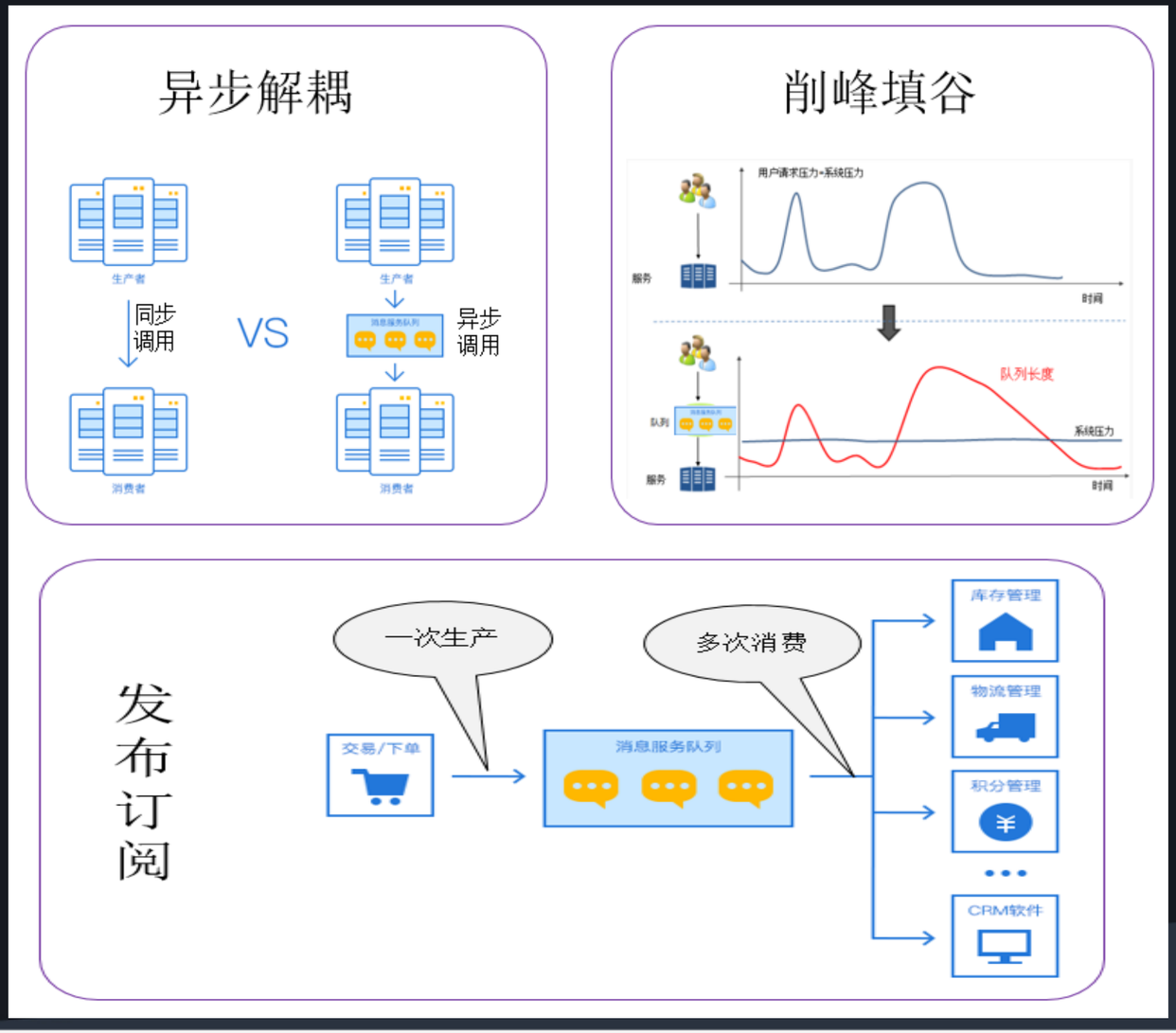
- 腾讯分布式服务开发框架TSF：微服务开发和生命周期管理PaaS平台
- 消息中间件CMQ、CKafka: 高可靠、强一致、高弹性分布式消息服务
- IoT Hub ：安全、高并发、多协议设备接入与规则引擎解析

TABLE OF

CONTENTS 大纲

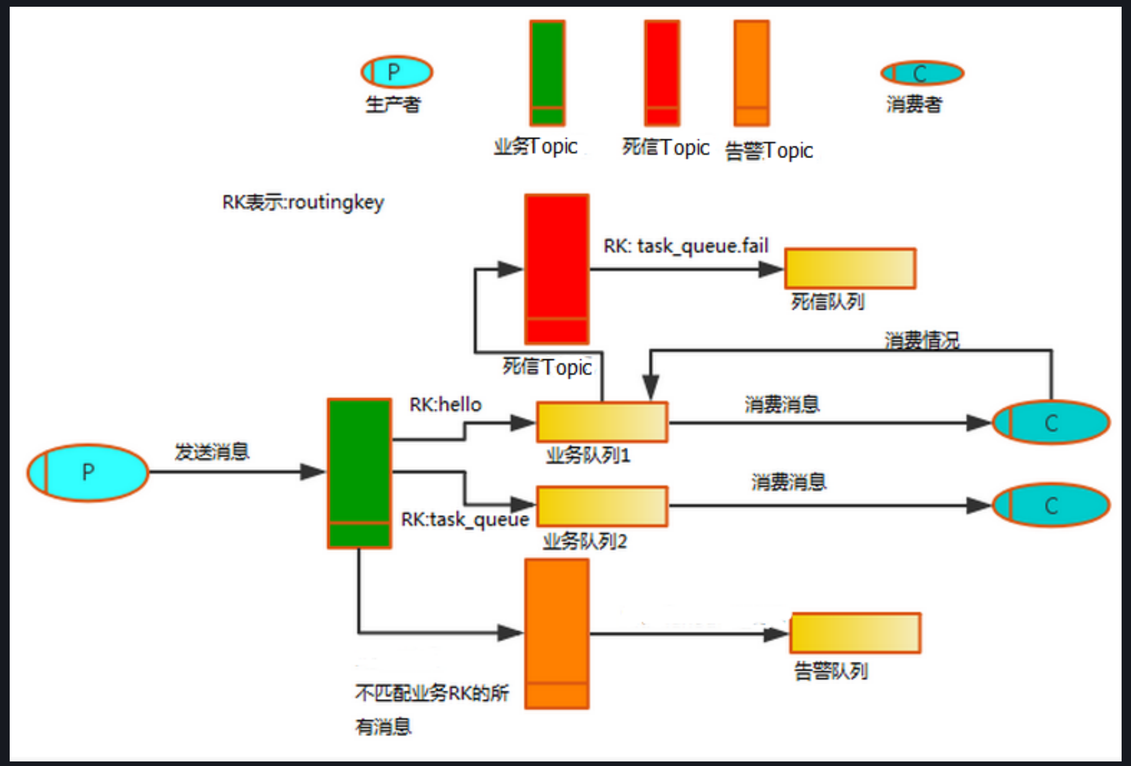
- 背景介绍
- 从开发、运维维度解析核心原理
- 分布式消息系统对测试的挑战
- 微信红包中如何使用消息中间件

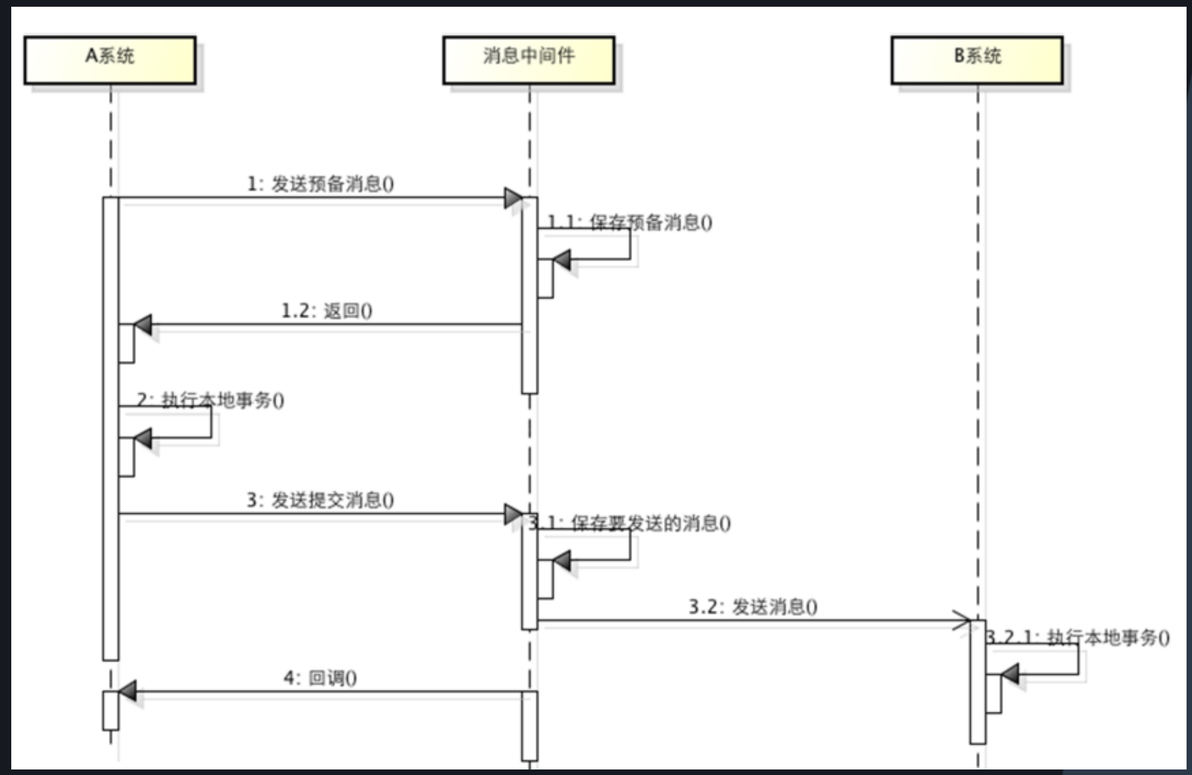
消息中间件应用场景



- 业务解耦:
  - 同步变异步
  - 最终一致
- 削峰限流：
  - 防止雪崩
  - 按需消费
- 广播：
  - 透明生产
  - 谁需要谁订阅
- 延时消费：
  - 定时触发-简化业务逻辑
- 回档消费：
  - 离线消费-从指定点消费

消息中间件应用场景





- 流数据处理：
  按需对消息过滤分类
  简化业务程序逻辑

- 分布式事务
  多个本地事务
  简单。高效，最终一致

背景介绍

CRMQ-Rabbitmq
2012内部大规模使用

CRMQ-Raft
2014年自研2.0

CMQ
2016年上线腾讯云

Ckafka
2014年开始内部使用

Ckafka
2017年上线腾讯云

- CMQ: 金融级别，多副本，高可靠，强一致，多级容灾
- Ckafka： 大数据领域，高吞吐，低延时
- MQ for IoT: MQTT接入，支持千万并发，安全性高

分布式消息系统

- 内部日消息量：超万亿条
- 接入业务个数：超万个

分布式消息系统特点

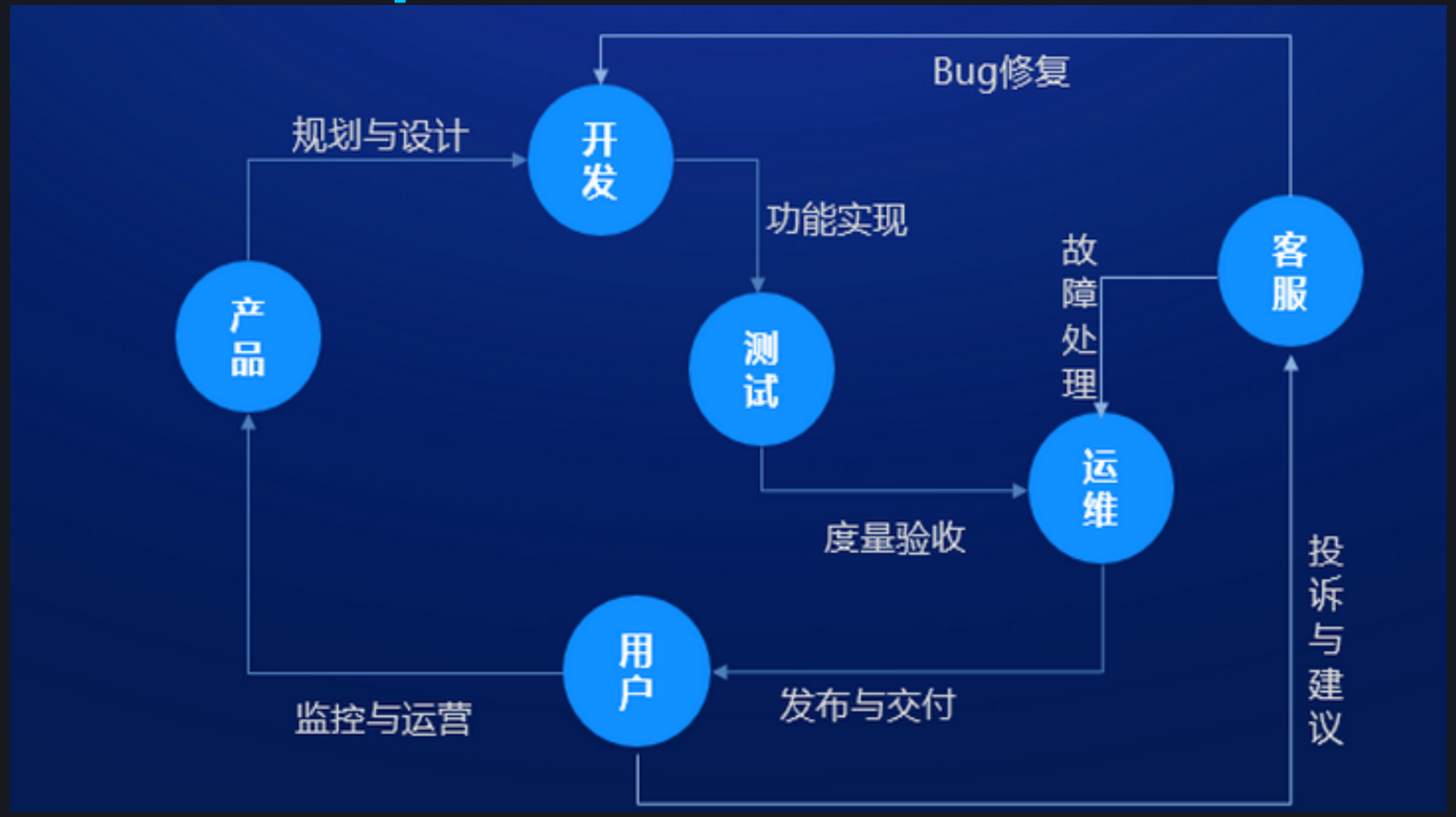
最大特点：可扩展性(Scale Out)

核心理念：多个节点协同工作，完成单个节点无法处理的任务
对硬件要求低
强调横向可扩展性
不允许出现单点故障服务不可用（RTO）
不允许单点故障导致数据丢失(RPO)

核心问题：CAP

CMQ: CP
Kafka: AP(配置可调整)

DevOps 流程



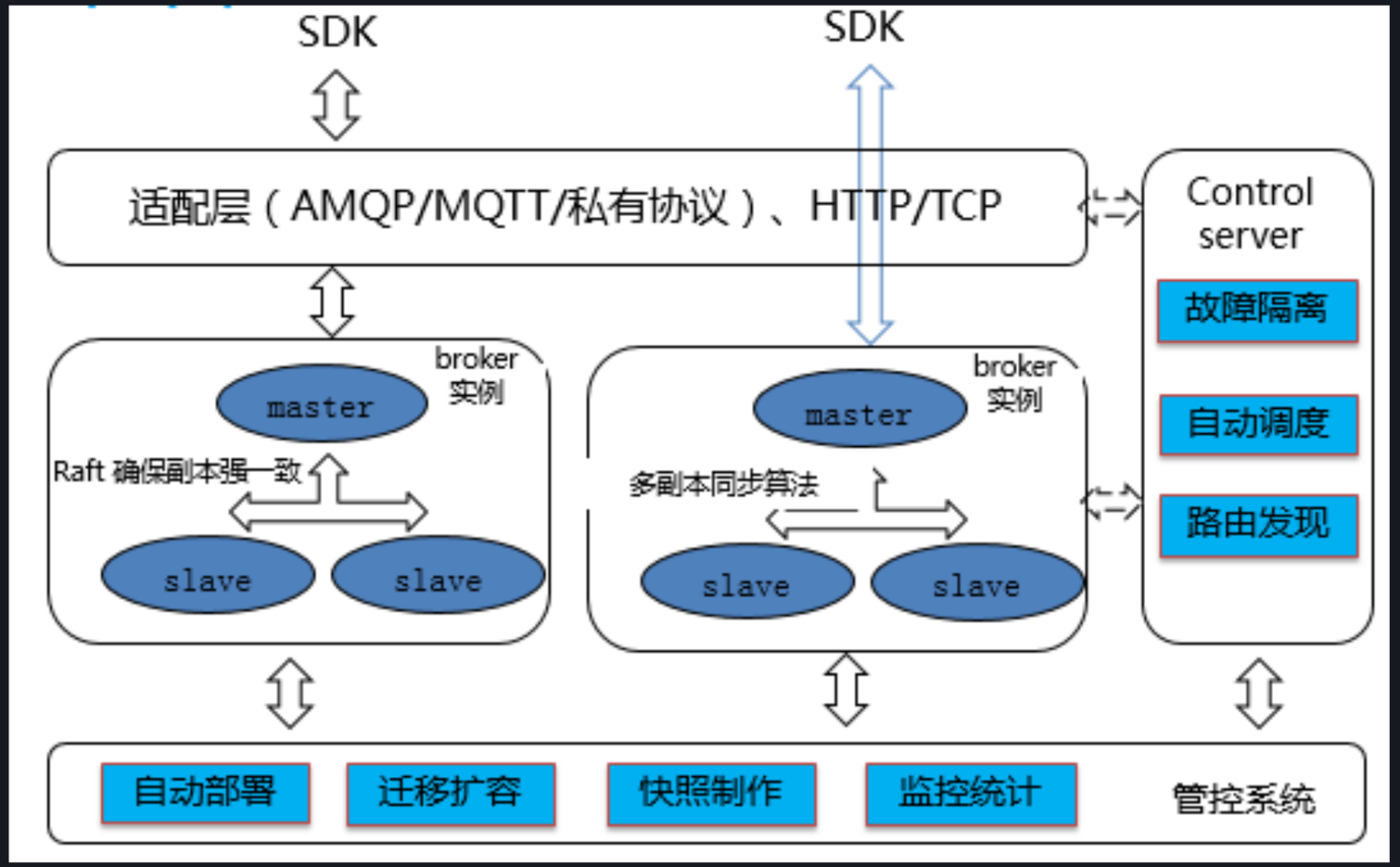
- 开发阶段：方案设计、代码设计、LLT（单元测试、模块测试）；
- 测试阶段：测试设计、自动化脚本、HLT（系统测试、联调测试）；
- 灰度阶段：灰度规则和策略、灰度测试设计、监控告警;
- 上线阶段：定时在线测试设计、日常运营 监控告警

TABLE OF

CONTENTS 大纲

- 背景介绍
- 从开发、运维维度解析核心原理
- 分布式消息系统对测试的挑战
- 微信红包中如何使用消息中间件

架构介绍



典型三层架构

- 适配层：
  协议解析
  弹性伸缩
- 存储层：
  多副本
  强一致
  高可靠
  高可用
  高性能
  低延时

- 运营难点：如何做到发布变更对业务透明？
  如何做到弹性扩展？
  节点故障如何处理？ 集群故障如何处理，园区级别故障如何处理？

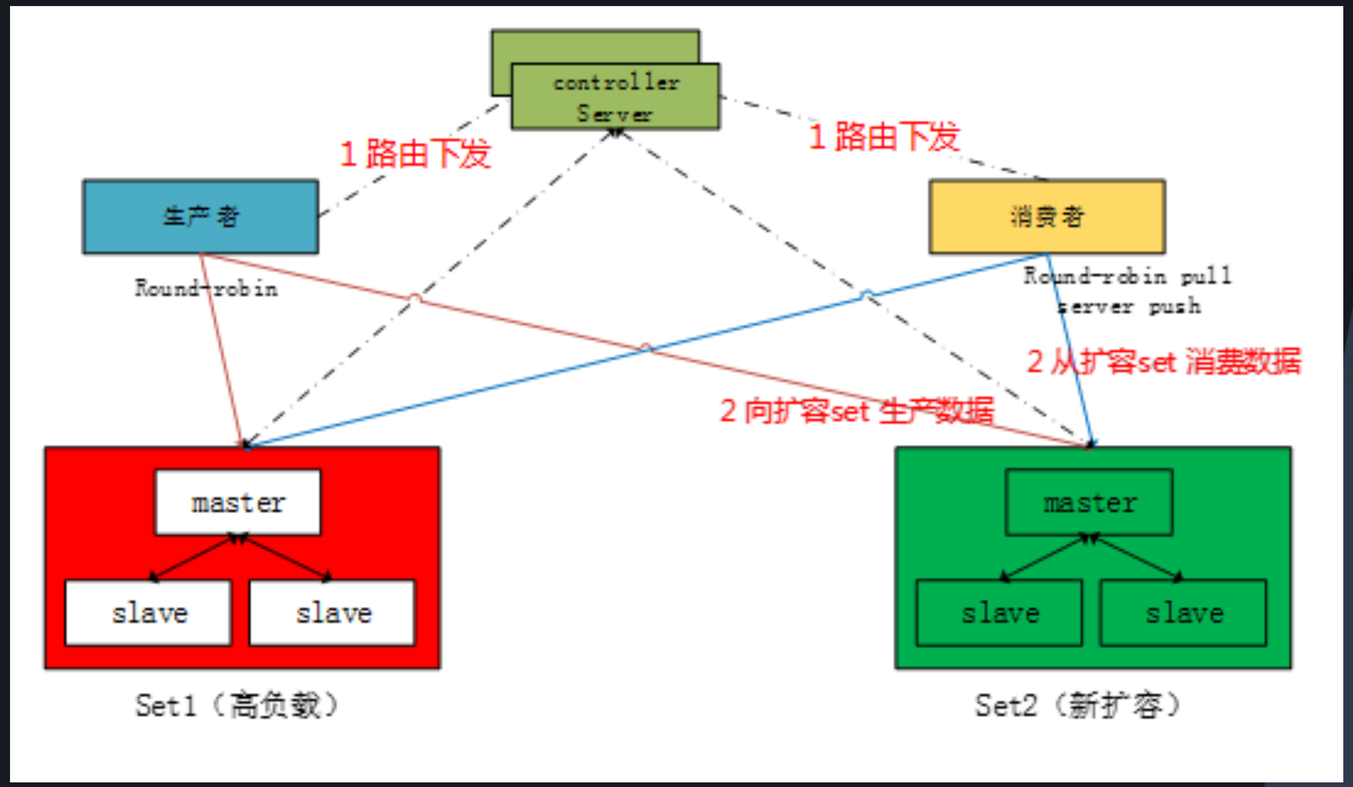
弹性伸缩

- 问题：性能和堆积不受限与set，支持透明scale out， 可以无限扩展

  ✓Controller server: 向生产者 消费者 下发扩缩容路由信息;

  ✓生产: 获取路由信息，轮询多个 broker set 实现消息生产

  ✓消费: 获取路由信息，通过轮询主动拉 或者 server 推的方式消费数据



高可靠(1/2)

- 数据可靠性：系统有n个数据对象，在1年内会损坏m个对象，可靠性为1-m/n
- 可靠性相关因子：副本数、磁盘故障率、修复率
- 常见 master+Nslave数据多副本问题：

✓数据一致性问题：异步复制会导致slave上数据丢失，同步复制到所有slave导致系统可用性低

✓同步复制性能问题：每个请求都发起一次主从数据同步，导致系统性能低下

✓持久化问题：在master、slave cache中还未来得及持久化到磁盘的数据存在异常情况下丢失的可能

✓刷盘性能问题：每个请求都强制刷盘一次，导致系统IO负载高

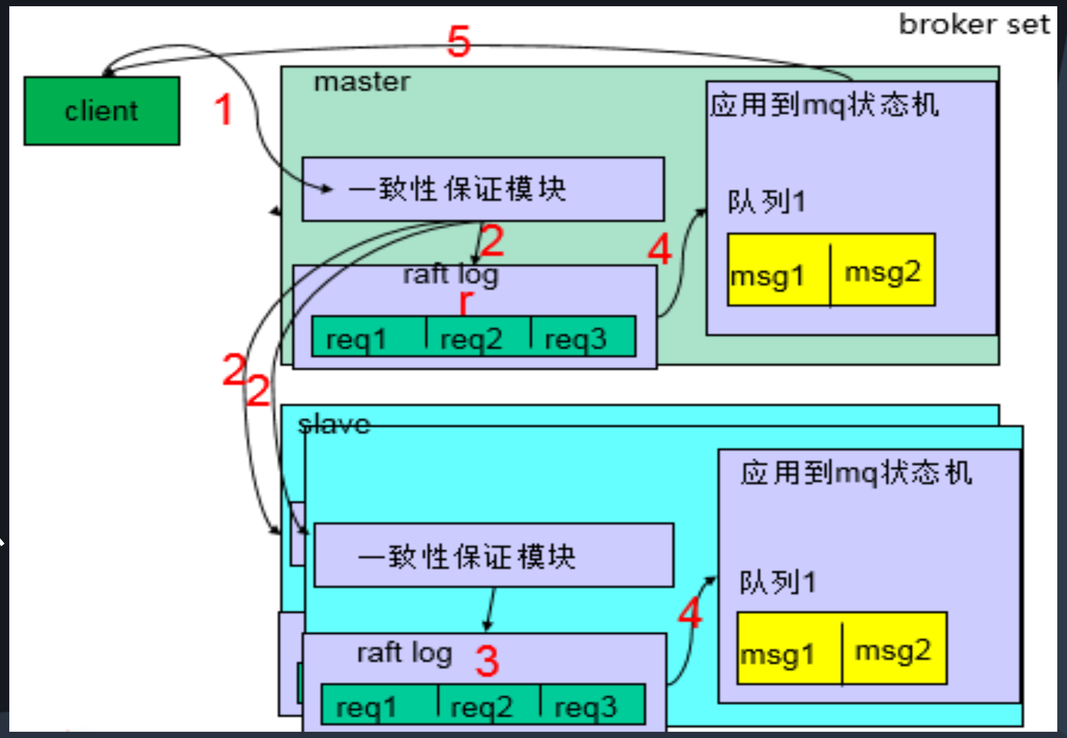
高可靠(2/2)

➢ 问题：异步复制无法保证强一致，同步复制到所有slave导致可用性低

✓方案：master同时向所有slave同步数据，salve 收到数据存储成功后向master返回成功，master 收到过半数节点返回成功后应用到mq状态机，返回用户成功。

举例：一个set 有3个节点组成，自动选举出一个master，两个slave

- Master 负责消息的生产消费请求，收到请求后先通过raft一致性模块写raft log到本地并同步给所有slave
- Slave 收到master发来的raft log持久化到本地同时返回master 成功信息
- Master 收到set中过半节点的成功信息后将请求信息提交到mq 状态机
- Mq 状态机处理请求信息后返回用户成功



高性能(1/2)

- 问题：同步复制性能问题

  Master每收到一个请求都向所有slave各发起一次网络IO, slave 处理成功后回复master成功。Master 和slave 还需要对收到的请求同步刷盘

- 分析：单次请求同步复制下，同步刷盘耗时：

✓$T_{total}=T_{creat\_raft\_log}+T_{master\_fsync\_raft\_log}+T_{replicate\_raft\_log}+T_{apply\_raft\_log}$

✓$T_{creat\_raft\_log}$、 $T_{write\_raft\_log}$ $T_{apply\_raft\_log}$ 受限单机处理性能

✓$T_{replicate\_raft\_log}=T_{raft\_log网络传输}+T_{slave\_fsync\_raft\_log}$，$T_{raft_log网络传输}$取决于RTT值

✓$T_{total}=T_{creat\_raft\_log}+T_{master\_fsync\_raft\_log}+T_{raft\_log网络传输}+T_{slave\_fsync\_raft\_log}+T_{apply\_raft\_log}$

其中$T_{master_fsync_raft_log}$与slave的 $T_{raft\_log网络传输}$ $T_{slave\_fsync\_raft\_log}$ 是并行发生的。

fsync_raft_log时间取决于磁盘性能，raft_log 网络传输时间取决于网络RTT。 由此可见这两个值是硬件相关的，因此我们只能通过提高每次批量发送raft_log 和批量fsync 刷盘来提高QPS(在牺牲一定请求延时情况下，可配置)

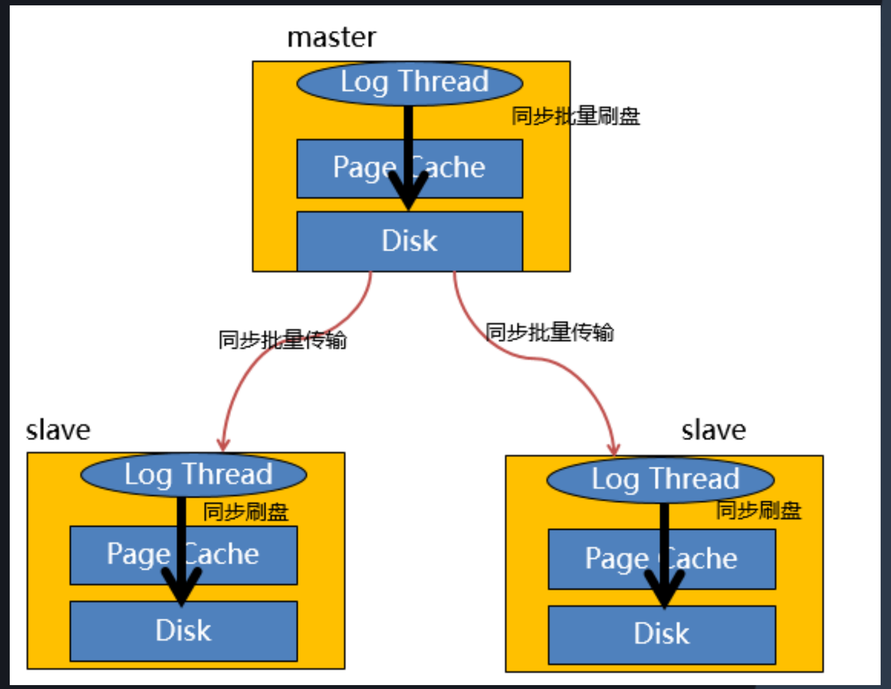
高性能(1/2)

- 解决思路：批量同步、批量刷盘提升性能
- 具体方法：

✓批量fsync：master 通过定时、定量两个维度合并请求批量刷盘，提高QPS(可配置)

✓ 批量发送raft log到slave: 同上

✓ 严格同步刷盘：master slave 遵循fsync 后才返回成功的逻辑，确保异常断电、宕机情况下的数据100%不丢失。



节点可用性

- 问题：当set内节点间发生网络分区时如何确保数据一致性？

✓当slave 单独在一个分区时，master可以 得到过半数节点的应答，无任何影响；

✓ 当master 单独成为一个分区时由于得不到大多数节点的应答，超时后最合适的 salve 会 提升为master

✓具体过程：

关键变量：

- Log序号（index）
- Master任期号（term）

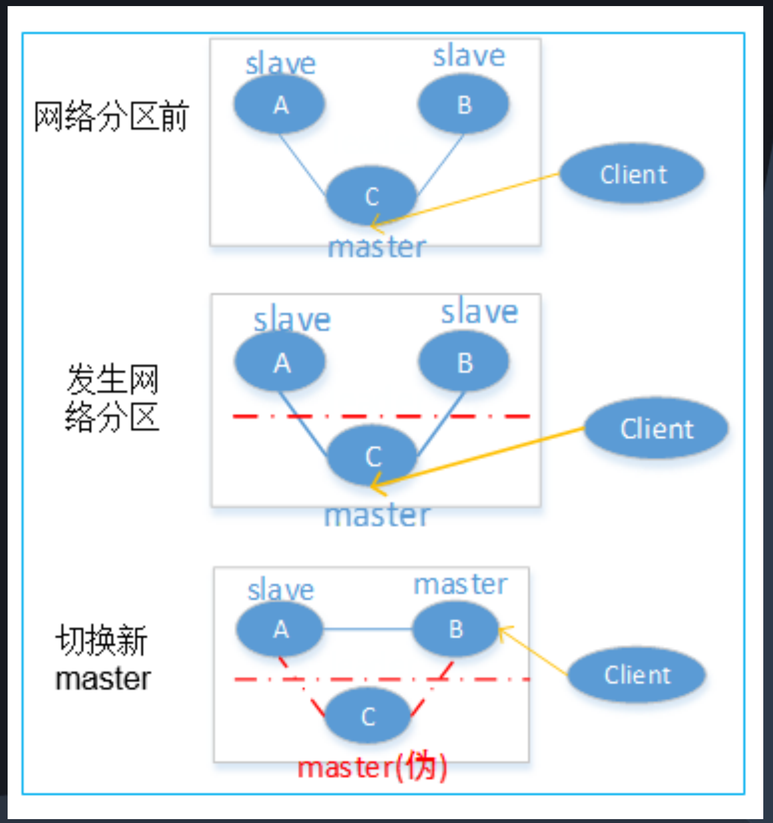
Raft Log同时记录了index和term。

Raft数据同步核心思想：

1）Master通过选举产生，同时产生了一个新的term，且新term > 老term

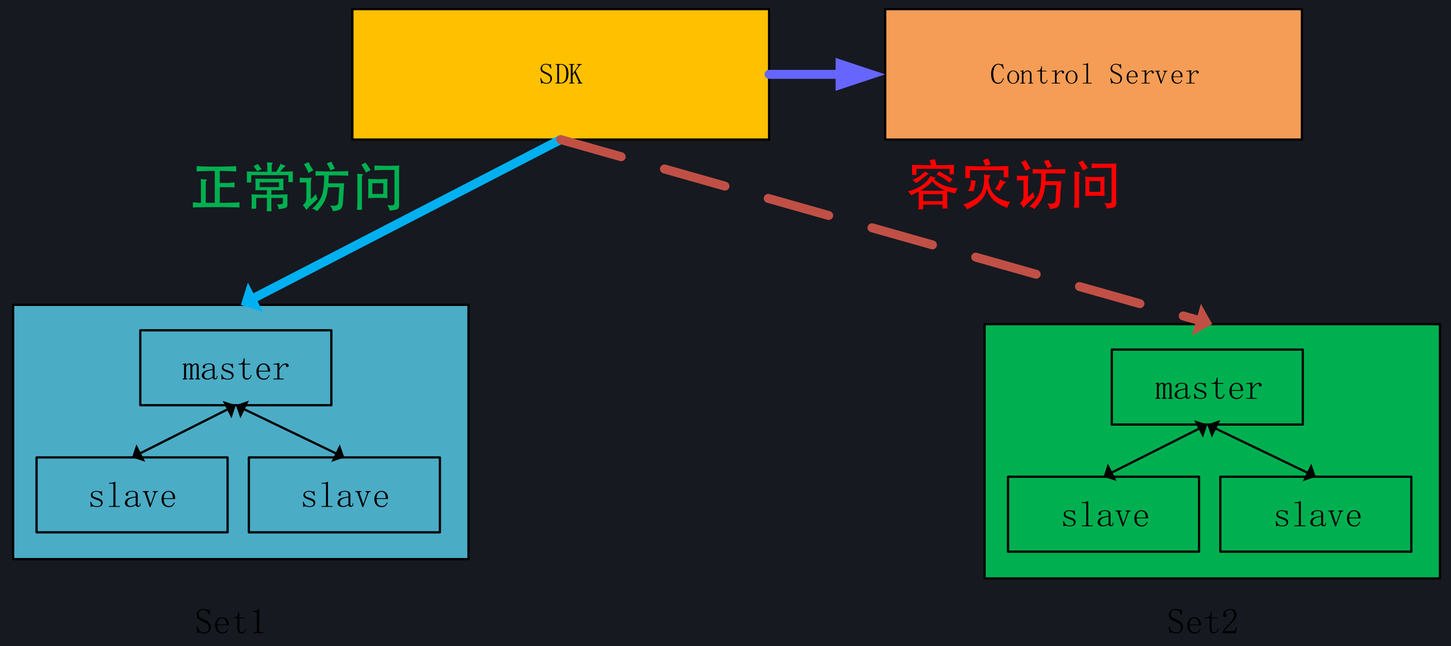
2）Slave只接受大于或者等于当前term的连续log。

3）Master根据Slave的上一条log和term，发送差量log，实现数据一致性。



集群可用性

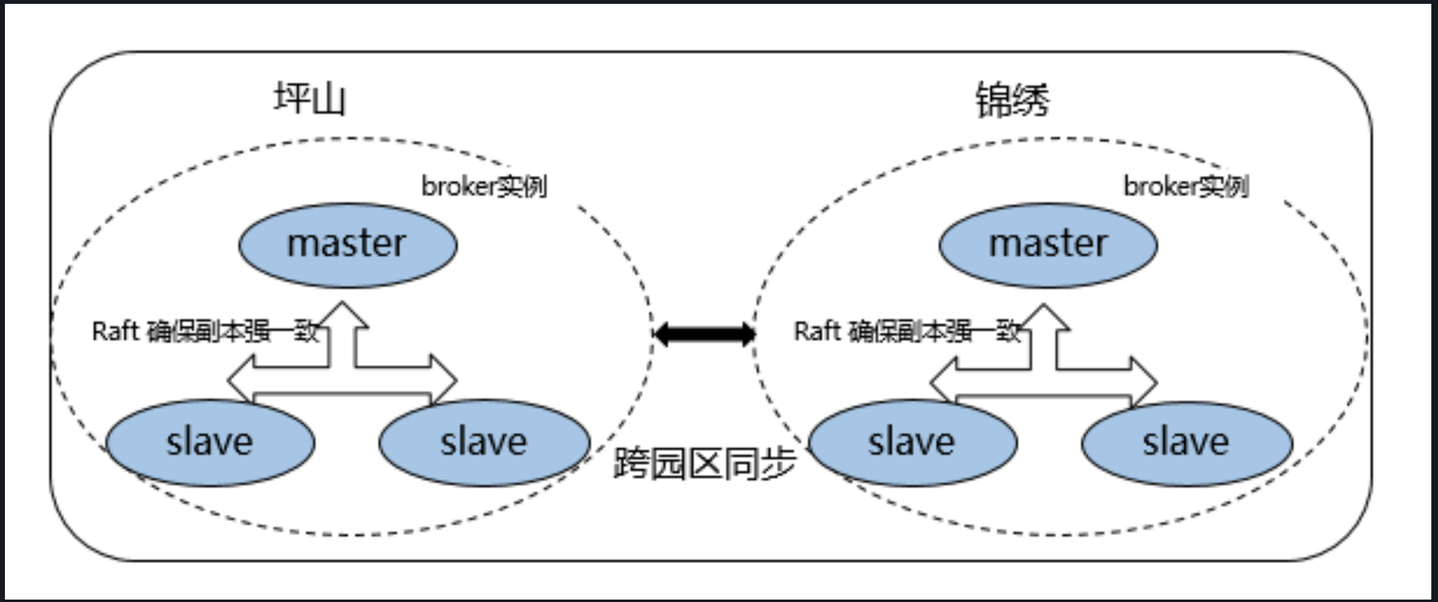
- 问题：raft解决了单节点故障导致的可用性问题，如果同一个set内多个节点故障，怎么办？
- 解决方案：双SET服务能力



- ✓ 分别在两个set创建Topic/Queue元数据
- ✓ 一个Set真正对外服务，另外一个standby
- ✓ Set内节点故障选举时间最大2s，Set 间故障切换时间5s
- ✓ 切换时数据顺序性问题

跨园区可用

- 问题：金融业务要求具备跨园区容灾能力
- 思路： set内节点同园区部署，不同园区的set间异步同步数据



✓两中心部署：

1）两园区各部署一个set,园区间数据异步复制

2）性能高、园区间数据一致性受异步复制频率和园区间网络质量影响

Log Trace 系统

- 问题：如何满足业务日常定位问题需求？消息丢了，延时大了。。。

✓全路径日志追踪：

1) 支持对消息的生产 消费全流程trace，
2) 方便查看消息延时，是否丢失，投递几次等常见运维问题

TABLE OF

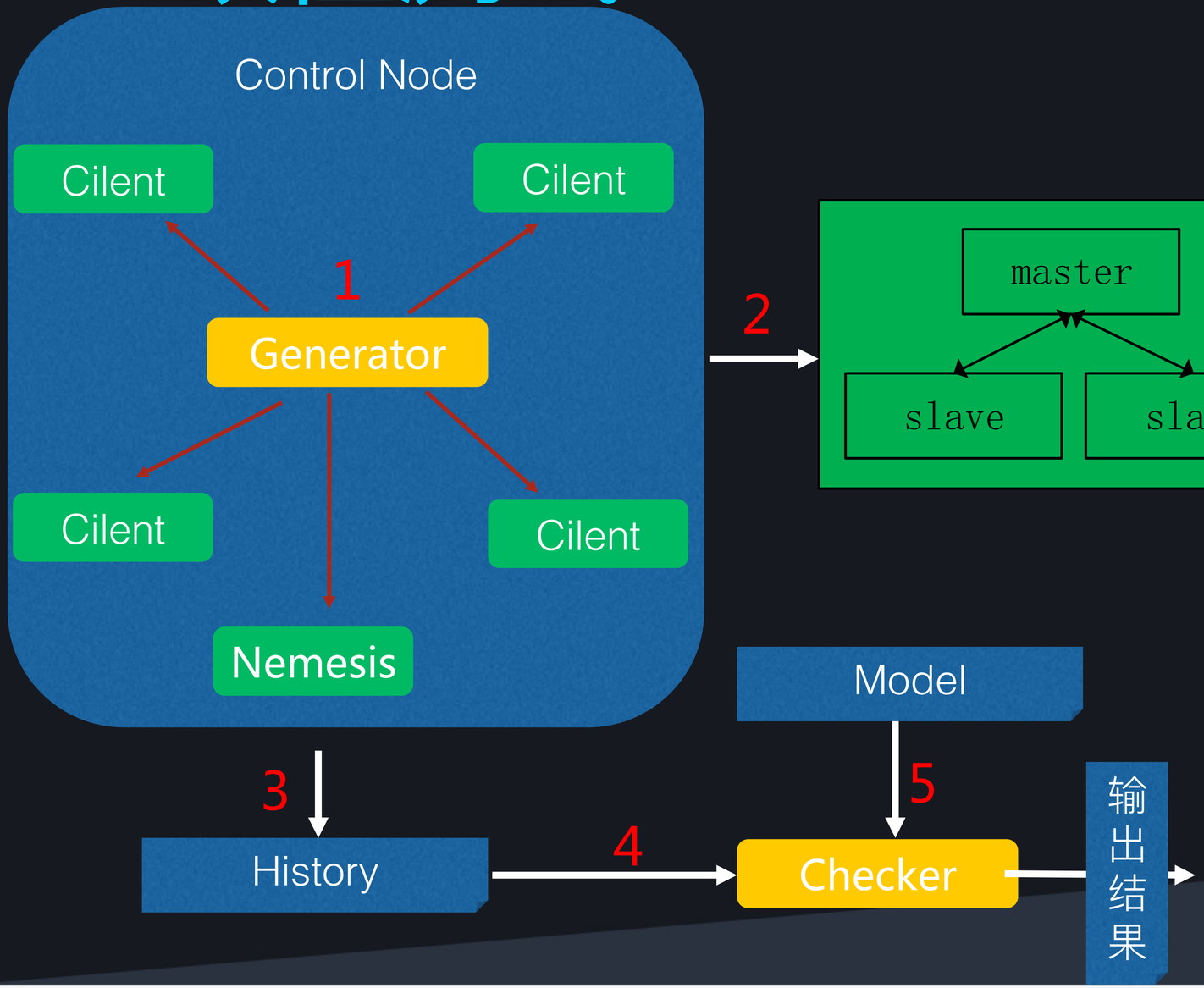
CONTENTS 大纲

- 背景介绍
- 从开发、运维维度解析核心原理
- 分布式消息系统对测试的挑战
- 微信红包中如何使用消息中间件

分布式消息系统测试

- 做好分布式系统的测试比做分布式系统本身更难
- 设计的时候就要考虑如何测试这个特性
  - ✓ 功能测试
  - ✓ 压力测试
  - ✓ 异常注入测试
  - ✓ 一致性测试

一致性测试



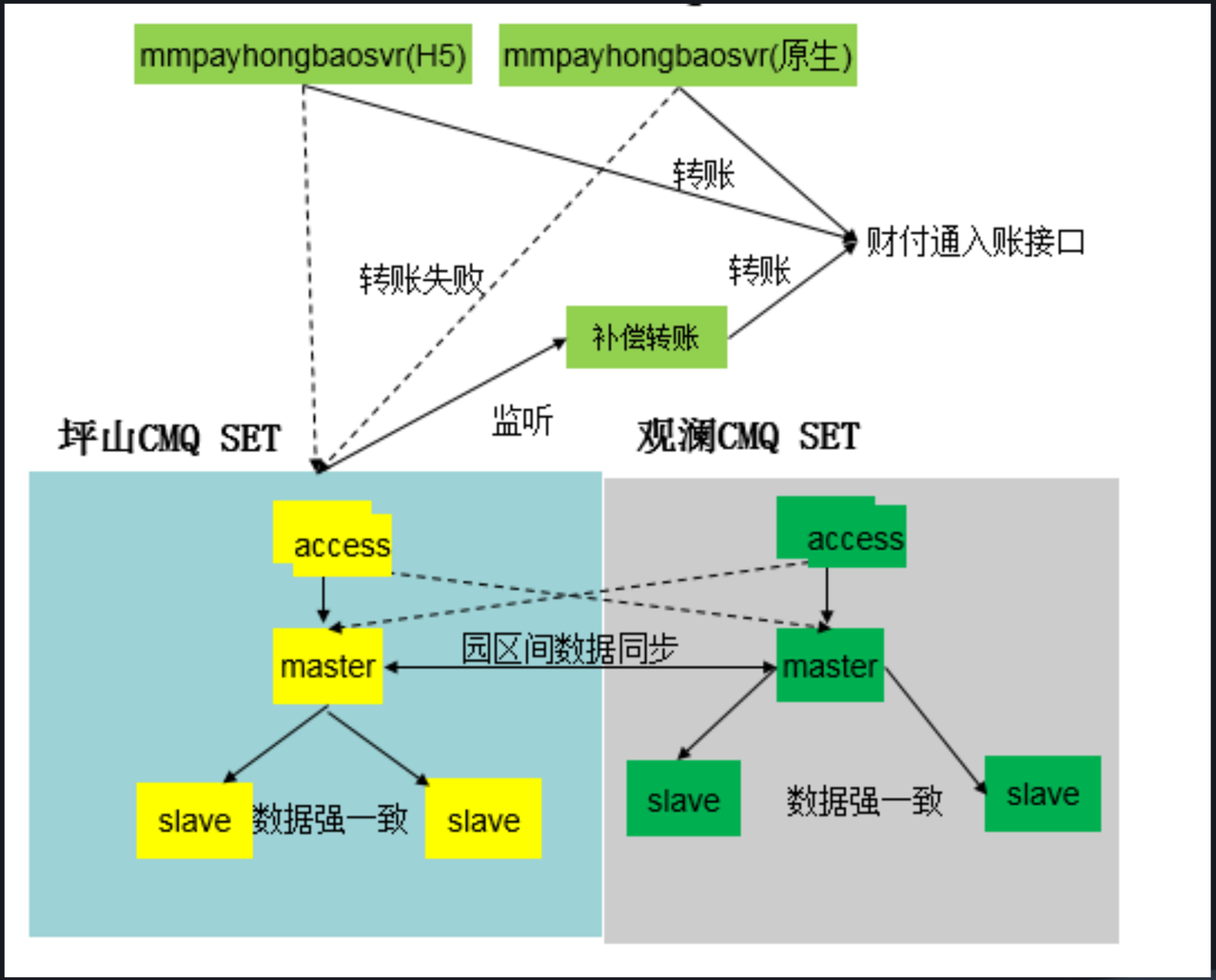
- 部署要测试的集群
- Control Node执行测试程序
  - ✓ 启动集群
  - ✓ 生产5个执行序列
  - ✓ 5个线程执行序列
    - 调用Client执行请求
    - 通过SSH登录N1-N2注入故障
- 记录执行结果
- 验证执行结果
- 停止集群分析结果

TABLE OF

CONTENTS 大纲

- 消息中间件应用场景
- 腾讯消息中间件核心技术原理解析
- 分布式系统对开发、测试、运维的挑战
- 微信红包中如何使用消息中间件

微信红包中如何使用消息中间件



- 目标：应对财付通转账接口异常，列表更新异常，回调写用户信息异常等场景，缓存失败请求，由独立模块补偿重试。

- 对消息中间件的要求：
  - 高可靠
  - 强一致
  - 高性能

THANK YOU

如有需求，欢迎至 [讲师交流会议室] 与我们的讲师进一步交流

ArchSummit
全 球 架 构 师 峰 会 2017