



基于amqp实现的golang消息队列 MaxQ

2017-07-01

张培培

饿了么-基础框架组

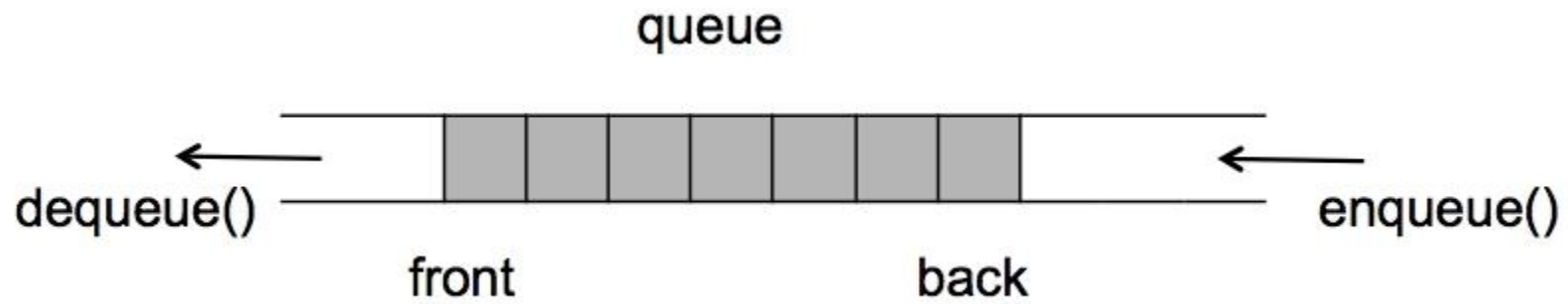


内容

1. 队列
2. IPG消息队列
3. AMQP协议
4. MaxQ架构模型
5. MaxQ相关特性
6. 使用场景和案例



1. 队列



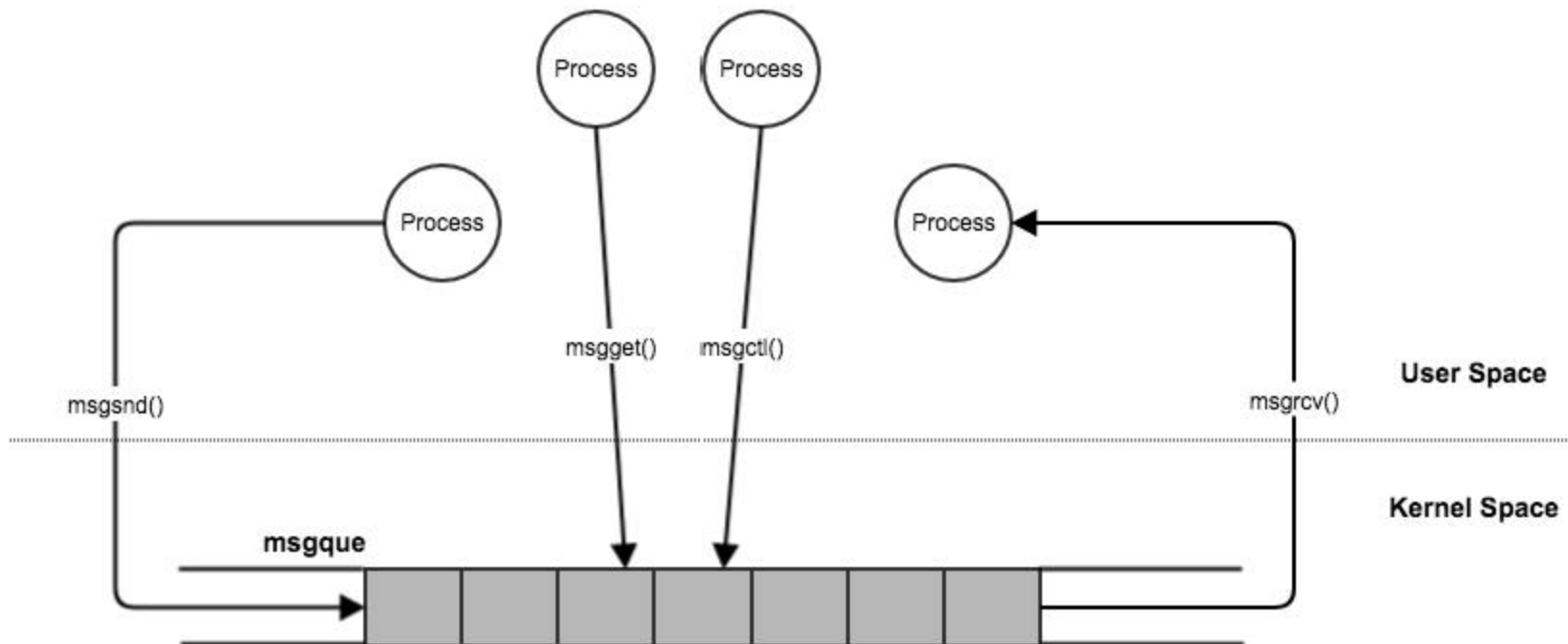
跟消息队列相比，有哪些共性？



- 生产者消费者
- 通信方式
- 存储方式
- 堆积能力
- 消息可靠性
- 生产消费关系
- Pull/Push



2. IPC消息队列

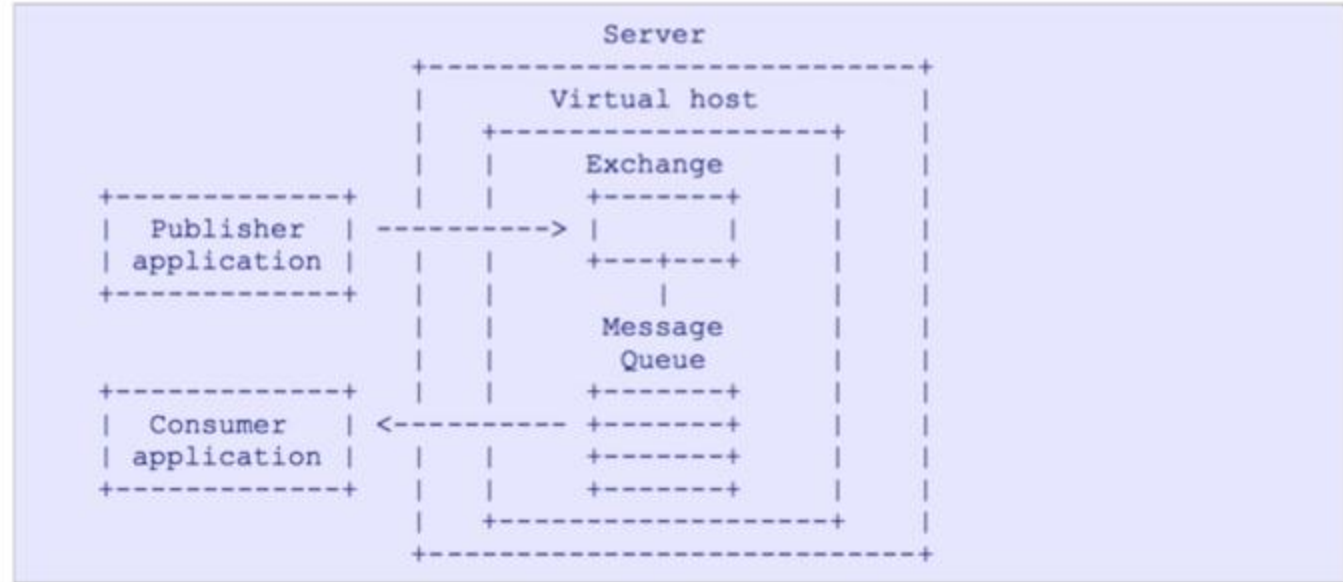


跟消息队列相比，有哪些共性？

3. AMQP(Advanced Message Queuing Protocol) 协议



- 七层应用协议
- 定义Broker实现
- 生产消费轻耦合
- 生产消费状态记录
- Push/Pull消费模型
- 消费者流控
- 事务支持



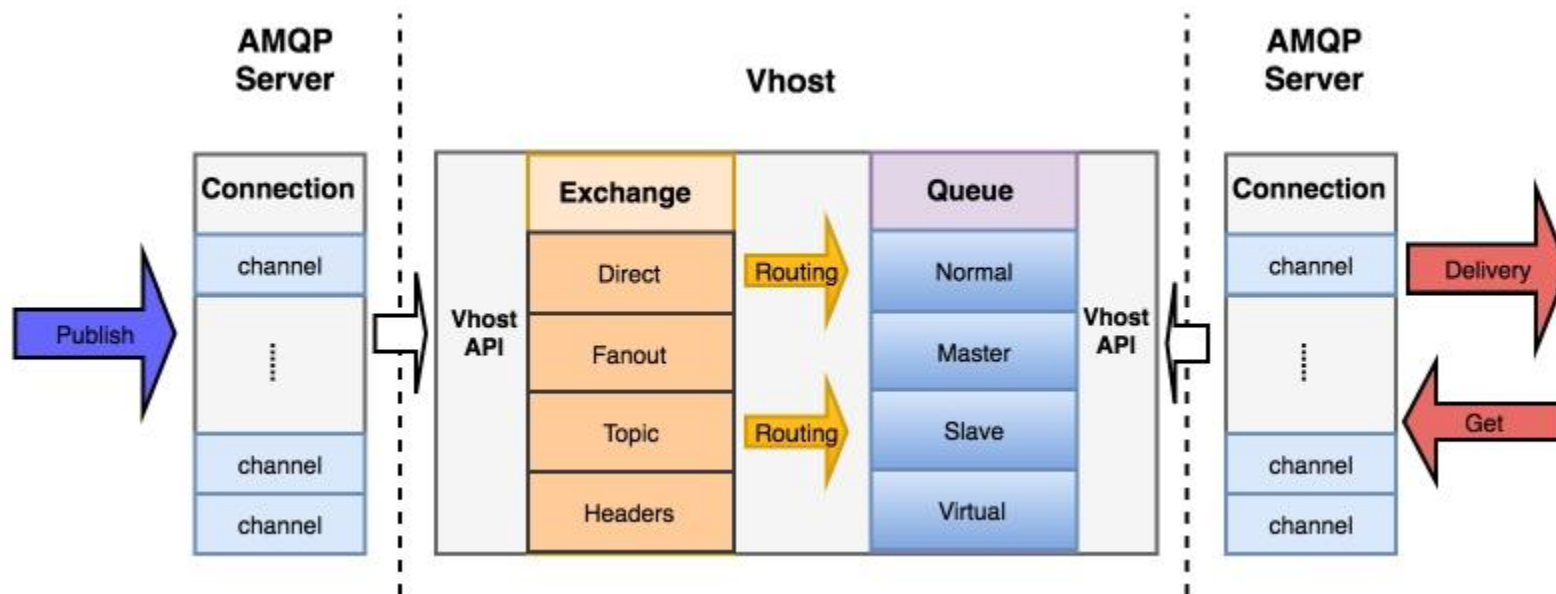


一些基本概念

- Broker
- Connection
- Channel
- Virtual host
- Exchange
- Queue
- Binding



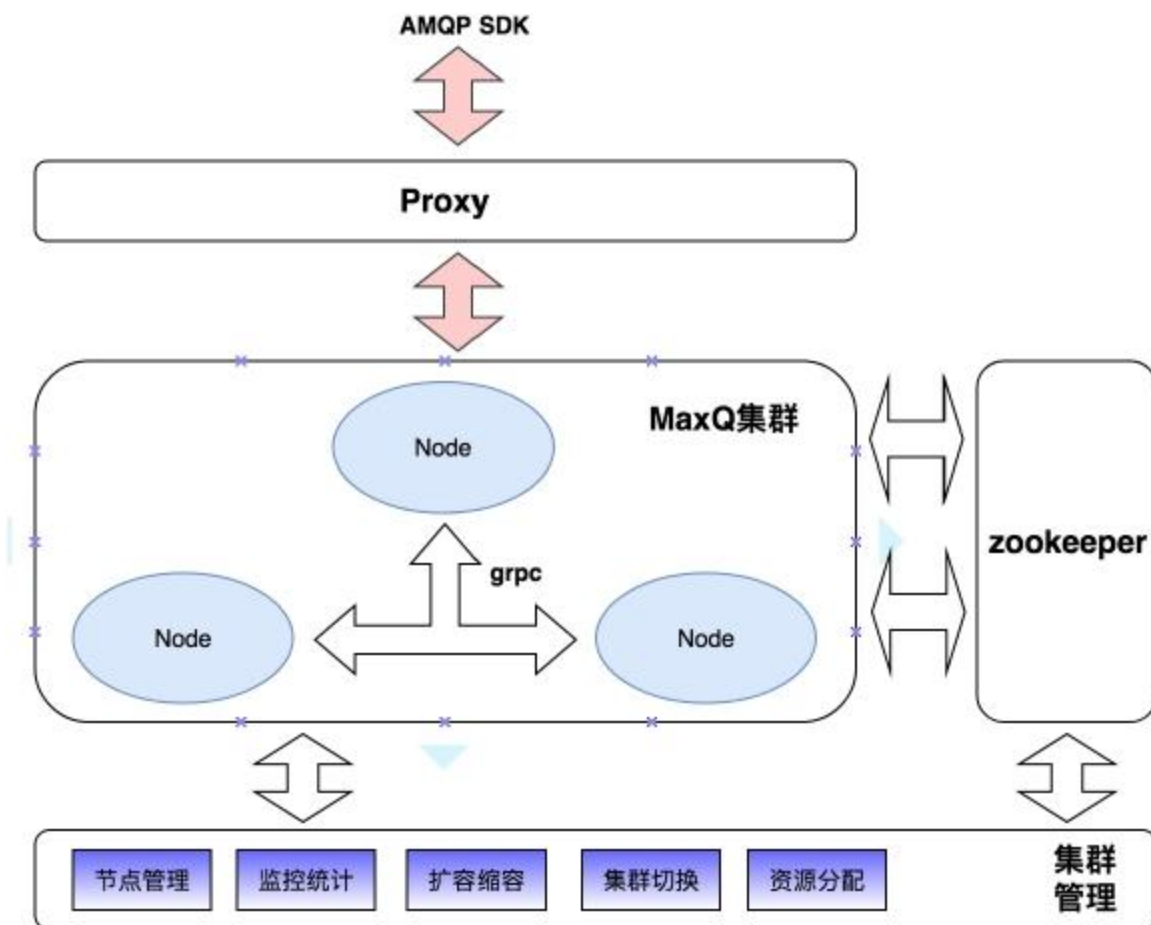
4. MaxQ - AMQP实现架构



- 按照协议spec自动生成frame encode/decode
- Exchange接口化
- Queue接口化——MaxQ集群



MaxQ 生产实现架构



满足生产环境的MaxQ集群:

1. 四层Proxy负载均衡
2. 集群中各Node通过grpc通信, publish、delivery、ack转发, HA消息同步
3. zookeeper存储元数据保证元数据一致性, Master queue选举
4. 集群管理维护所有MaxQ集群

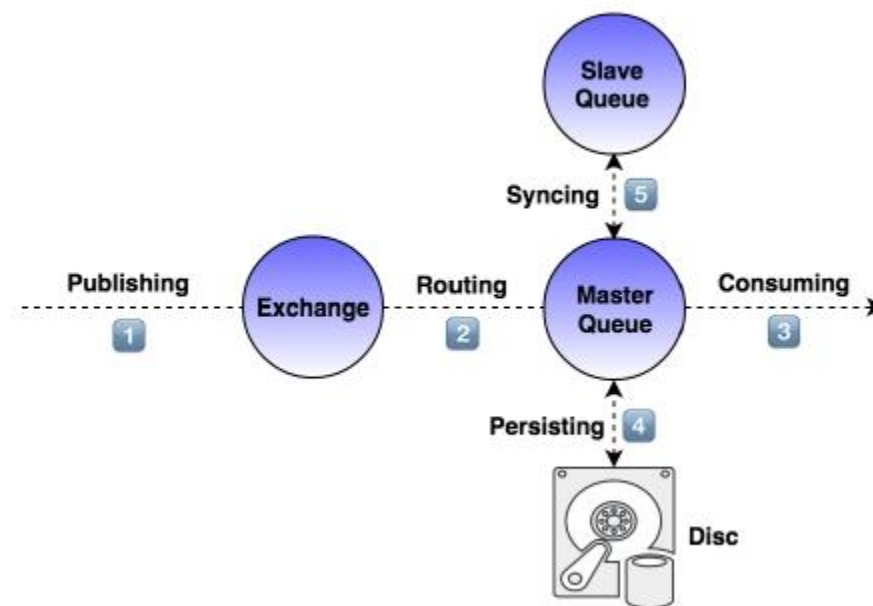


5. MaxQ相关特性

1. 消息可靠性
2. 容错性
3. 扩展性
4. 高并发



消息可靠性

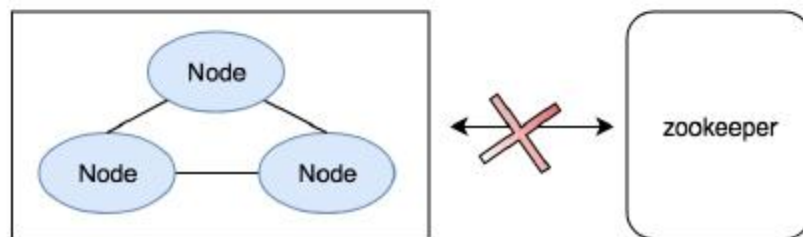


- 1) Publishing可靠性 -> Confirm
- 2) 消息Routing可靠性 -> Publish mandatory
- 3) Consuming可靠性 -> Ack
- 4) Persisting可靠性 -> RAID1
- 5) 分布式下的可靠性 -> Slave Queue



容错性

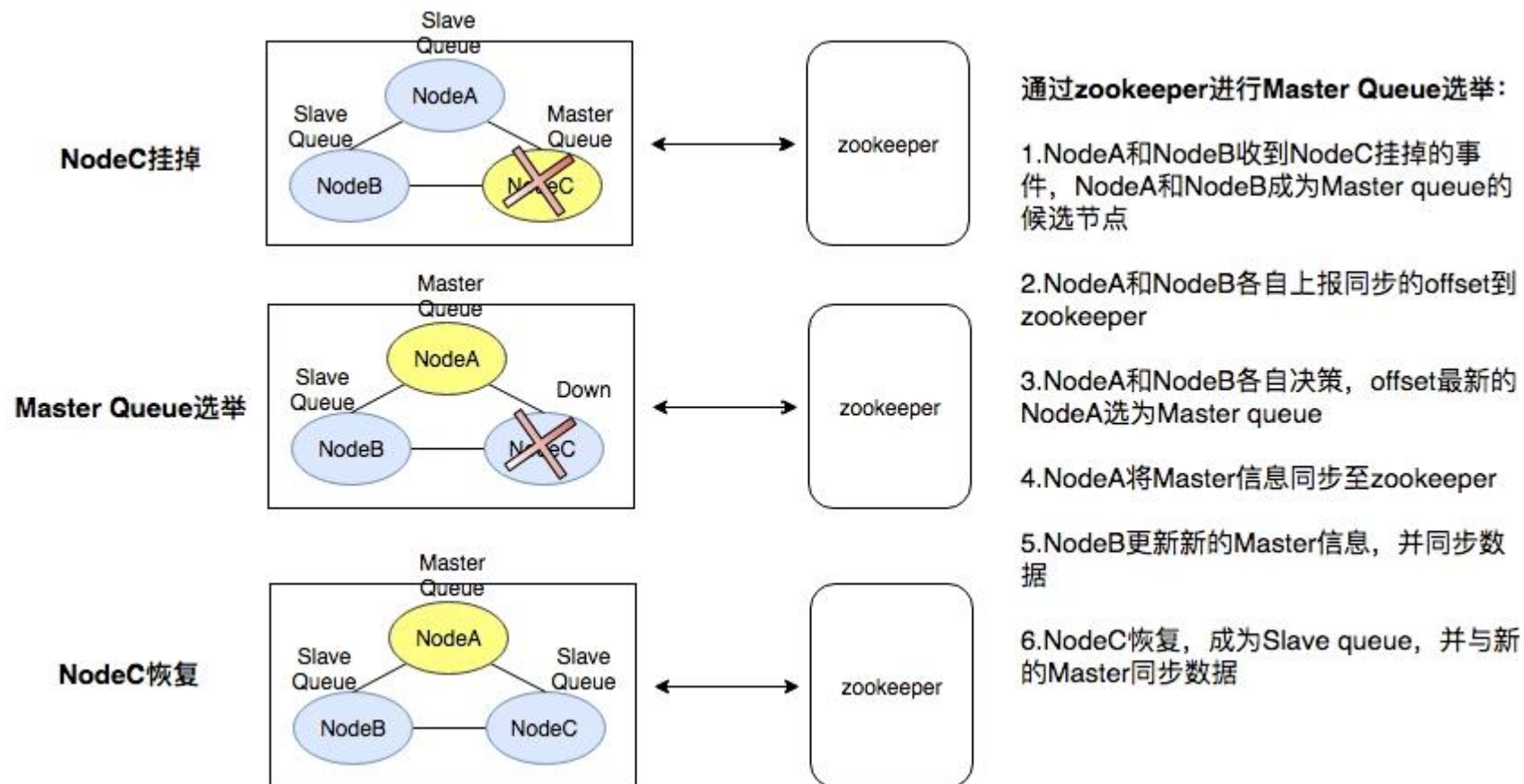
zookeeper不可用



- 元数据已缓存在内存中，不会有任何影响，生产方和消费方仍可正常生产和消费
- 服务自动降级，元数据不可变更
- zookeeper恢复，服务自愈

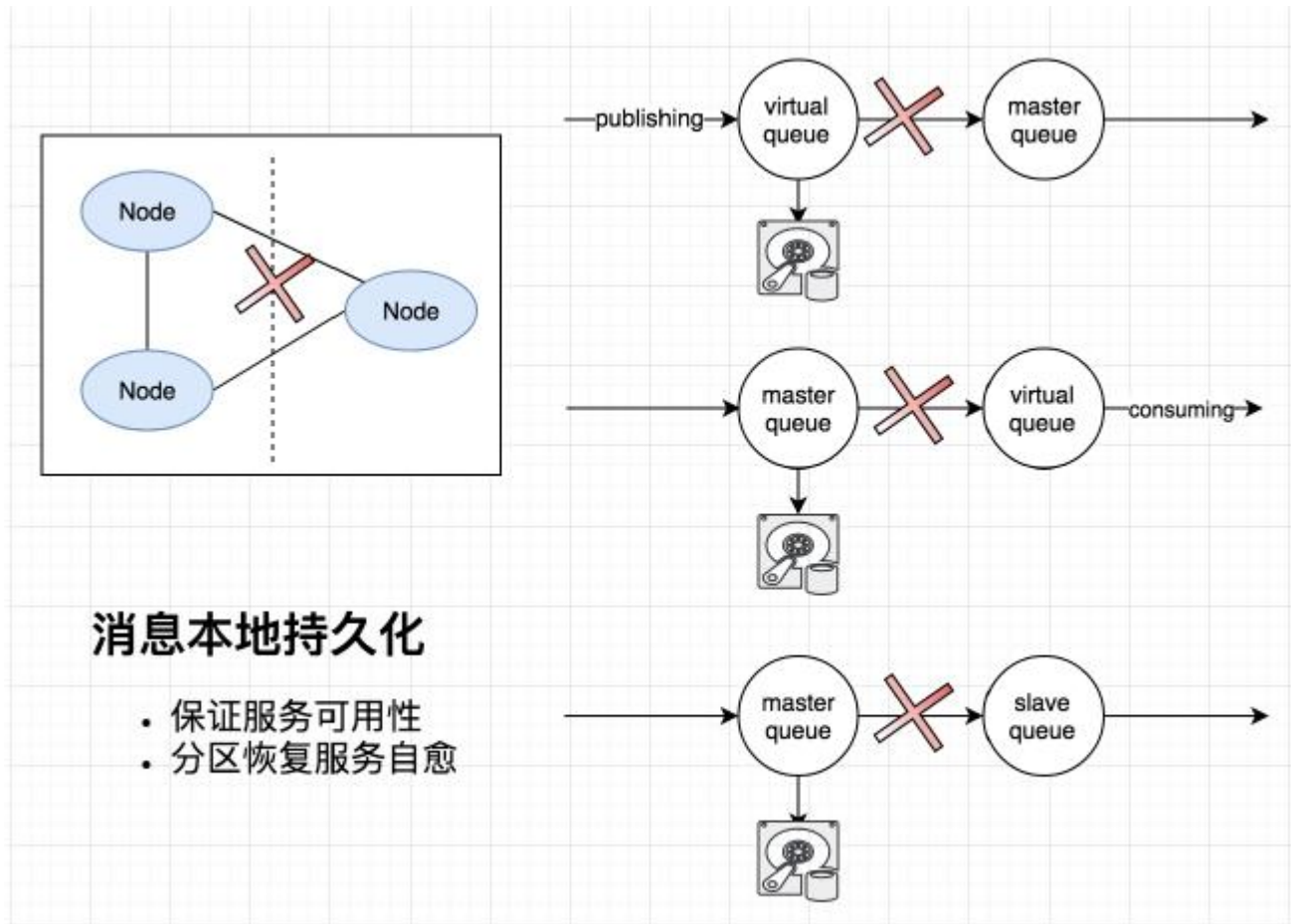


节点故障

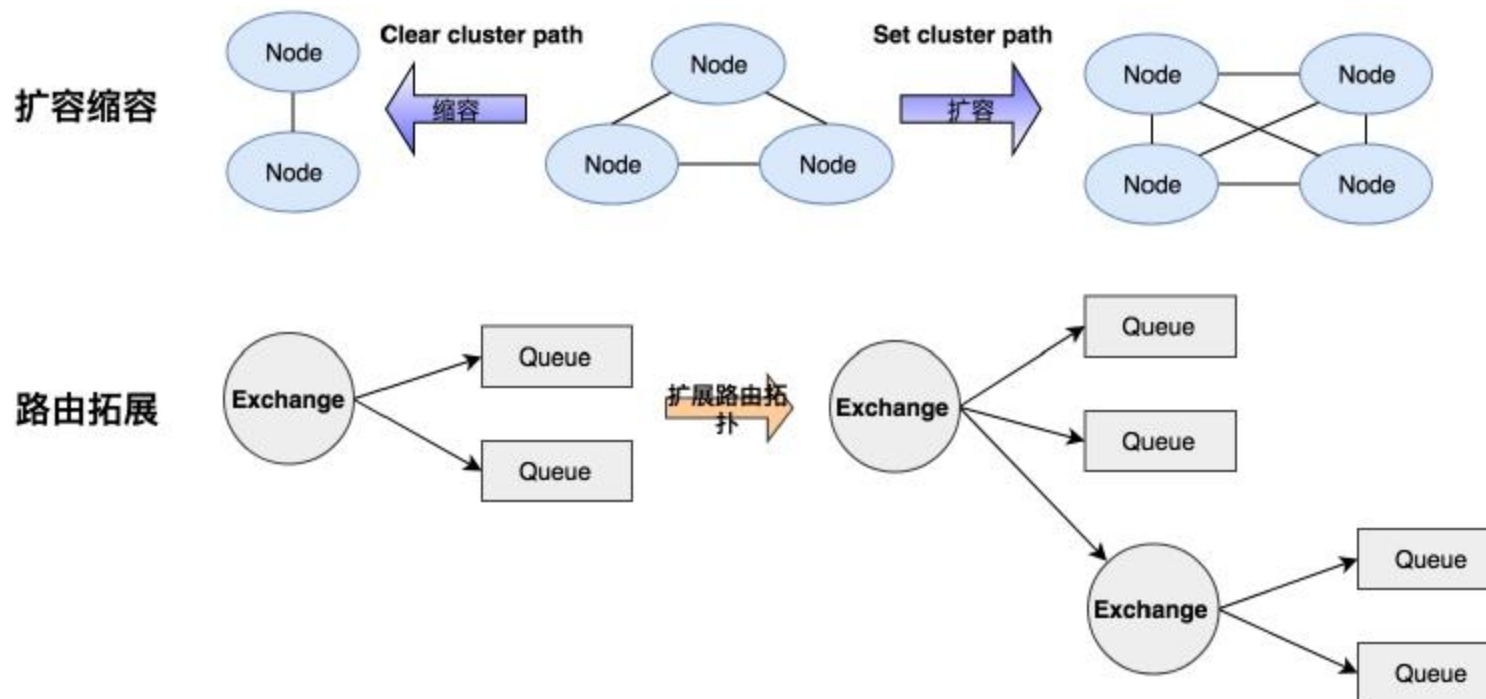




网络分区



扩展性

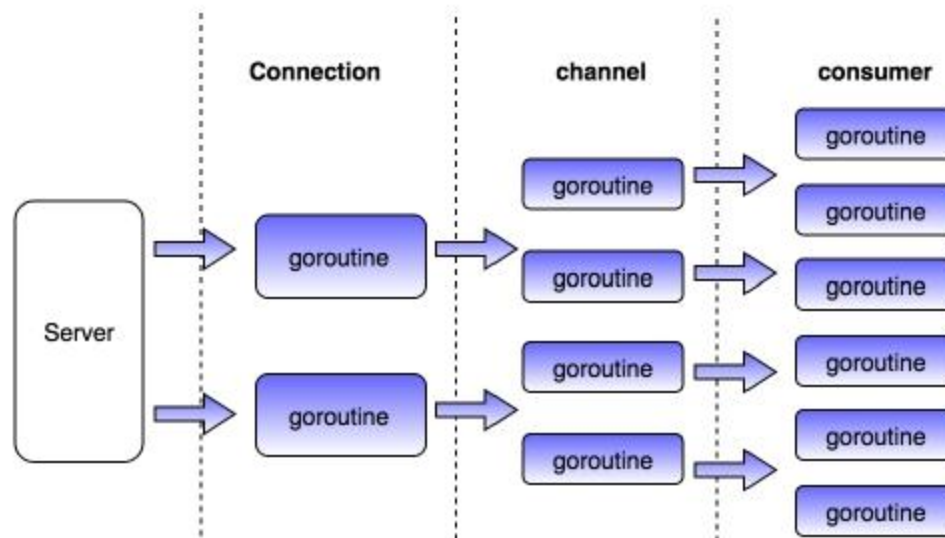


- HA、Exchange和Queue动态扩展属性参数
- Exchange Binding Queue支持自定义扩展, 如: x-message-ttl x-expires x-max-length x-dead-letter-exchange



高并发

goroutine + channel + consumer



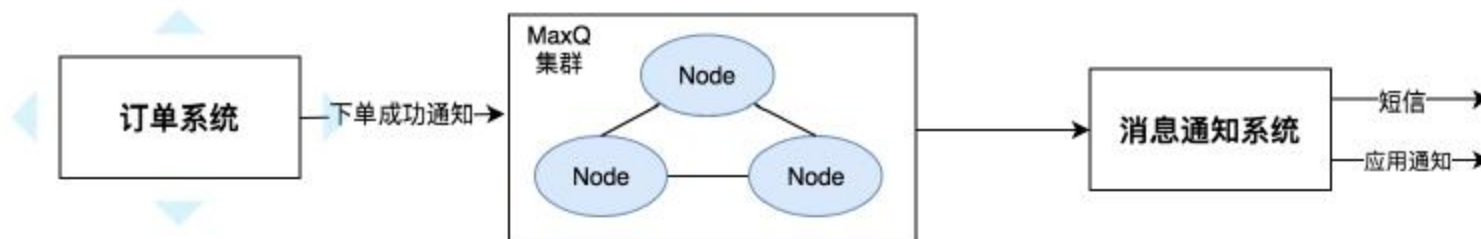
遇到的一些问题:

- goroutine 泄漏
- goroutine 卡死
- 连接 Half-Open



6. 使用场景和案例

异步解耦

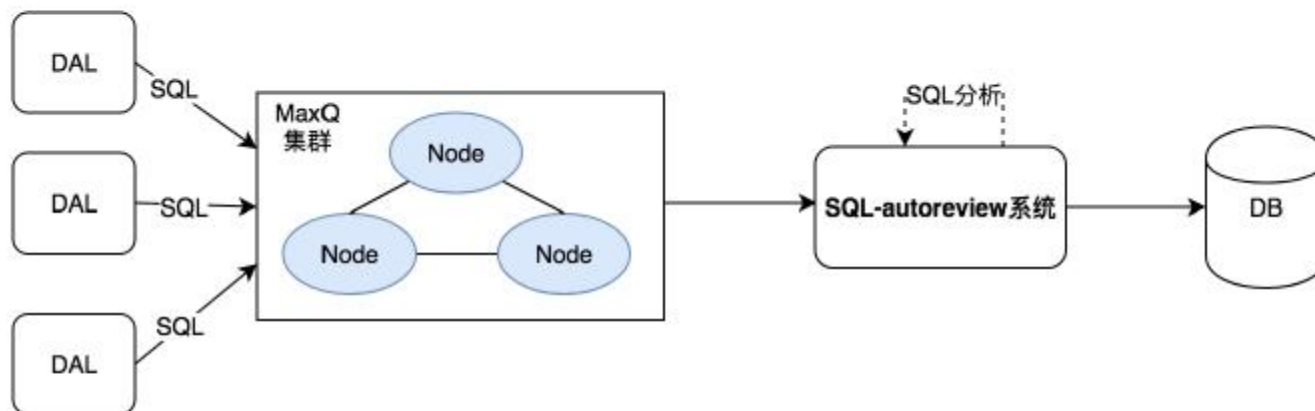


订单系统与消息通知系统解耦

1. 用户订单支付成功, 直接向MaxQ推送下单成功通知, 主流程迅速返回
2. 消息通知系统异步接收通知消息, 发送短信通知或应用通知



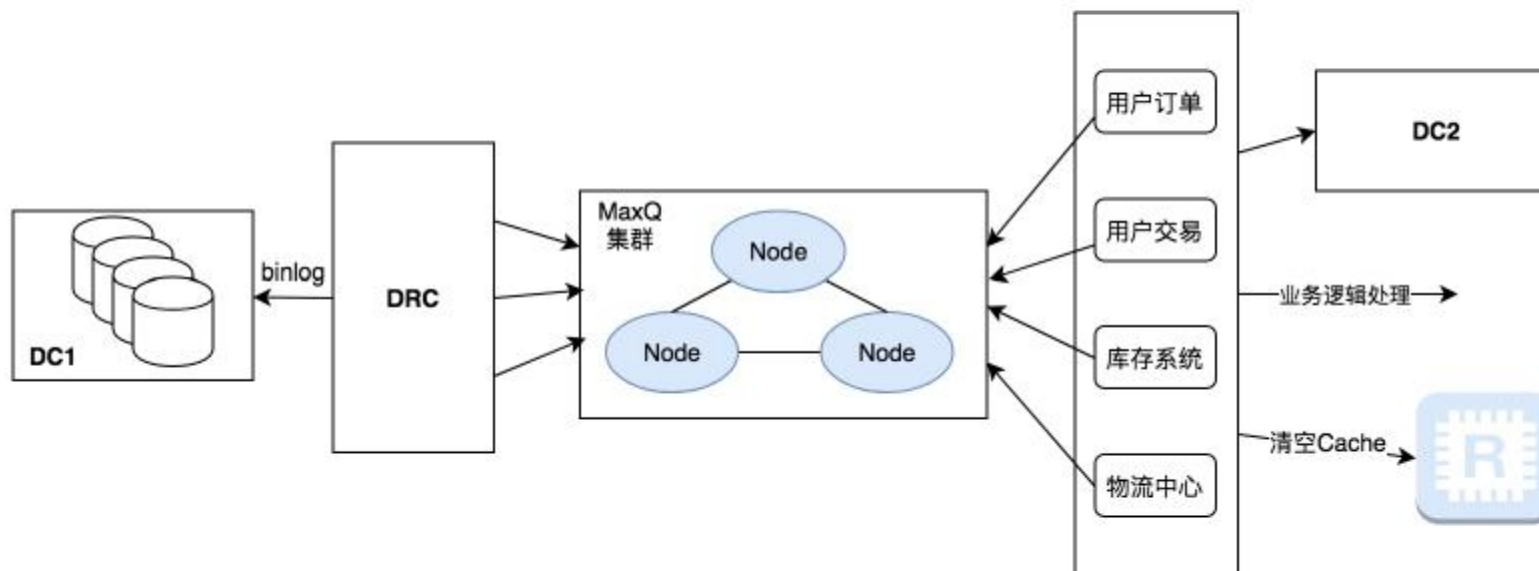
削峰填谷



SQL-autoreview系统分析优化SQL语句，并将结果落DB，属于慢消费，生产高峰期处理能力不够，可利用MaxQ的堆积能力，匀速消费和处理。



发布订阅



DC数据变更发布和订阅

1. DRC将DC的数据变更记录发布至MaxQ
2. 各业务系统订阅相关的数据变更，并进一步做业务处理



Q/A

- 1.为什么不用RabbitMQ
- 2.为什么基于AMQP实现?
- 3.与RabbitMQ功能性能上对比如何?

Thank you

张培培

饿了么-基础框架组



IT大咖说
知识分享平台

