分布式消息队列

马宏亮

人人聚财

问题

- 响应慢(同步代码过多)
- 高并发（发红包问题）
- 应用代码高度耦合，难以维护

MQ的应用场景

- 异步消息
- 系统解耦
- 流量削峰
- 日志收集
- 消息总线

MQ调研选型

名称	Active MQ	Rabbit MQ	Rocket MQ/kafka	Joram	Hornet Q	OpenM Q	MuleM Q	Sonic MQ	ZeroM Q
关注度	高	高	高	中	中	中	低	低	中
文档	一般	一般	多	少	少	少	少	少	少
开发语言	java	Erlang	java	java	java	java	java	java	c
是否开源	是	是	是	是	是	是	否	否	是
社区活跃度	较高	较高	较高	低	低	低	低	低	中

优缺点

名称	ActiveMQ	RabbitMQ	RocketMQ
缺点	没有大规模数据的应用场景实例，bug太多，每次版本发布都有很多bugfix	由于是erlang开发，和java客户端匹配度不是很好	Master-slave不支持自动切换，不过部署为双master同步双写也可以满足HA，性能虽略有下降，但仍满足业务需求。
优点	知名度较高的产品，在很多公司得到应用。各种协议支持较好，有多种语言的客户端	由于erlang语言的特性， mq性能较好	阿里主流消息队列框架，在阿里集团内部有大量的应用在使用，每天都产生海量的消息，并且顺利支持了多次天猫双十一海量消息考验,p2p公司：红岭创投也是其使用者。

人人聚财

为什么选择rocketMQ

RocketMQ在阿里集团内部有大量的应用在使用，每天都产生海量的消息，并且顺利支持了多次天猫双十一海量消息考验

采用**Java**语言编写

支持异步实时刷盘，同步刷盘，同步**Replication**，异步**Replication**

单机支持最高**5**万个队列，**Load**不会发生明显变化

- 队列多有什么好处？

单机可以创建更多**Topic**，因为每个**Topic**都是由一批队列组成

Consumer的集群规模和队列数成正比，队列越多，**Consumer**集群可以越大

RocketMQ单机写入TPS单实例约7万条/秒

消息的投递延时通常在几个毫秒

消费失败支持定时重试，每次重试间隔时间顺延

支持严格的消息顺序

支持根据**Message Id**查询消息，也支持根据消息内容查询消息（发送消息时指定一个**Message Key**，任意字符串，例如指定为订单**Id**）

单机也可以支持亿级的消息堆积能力

人人聚财

特性

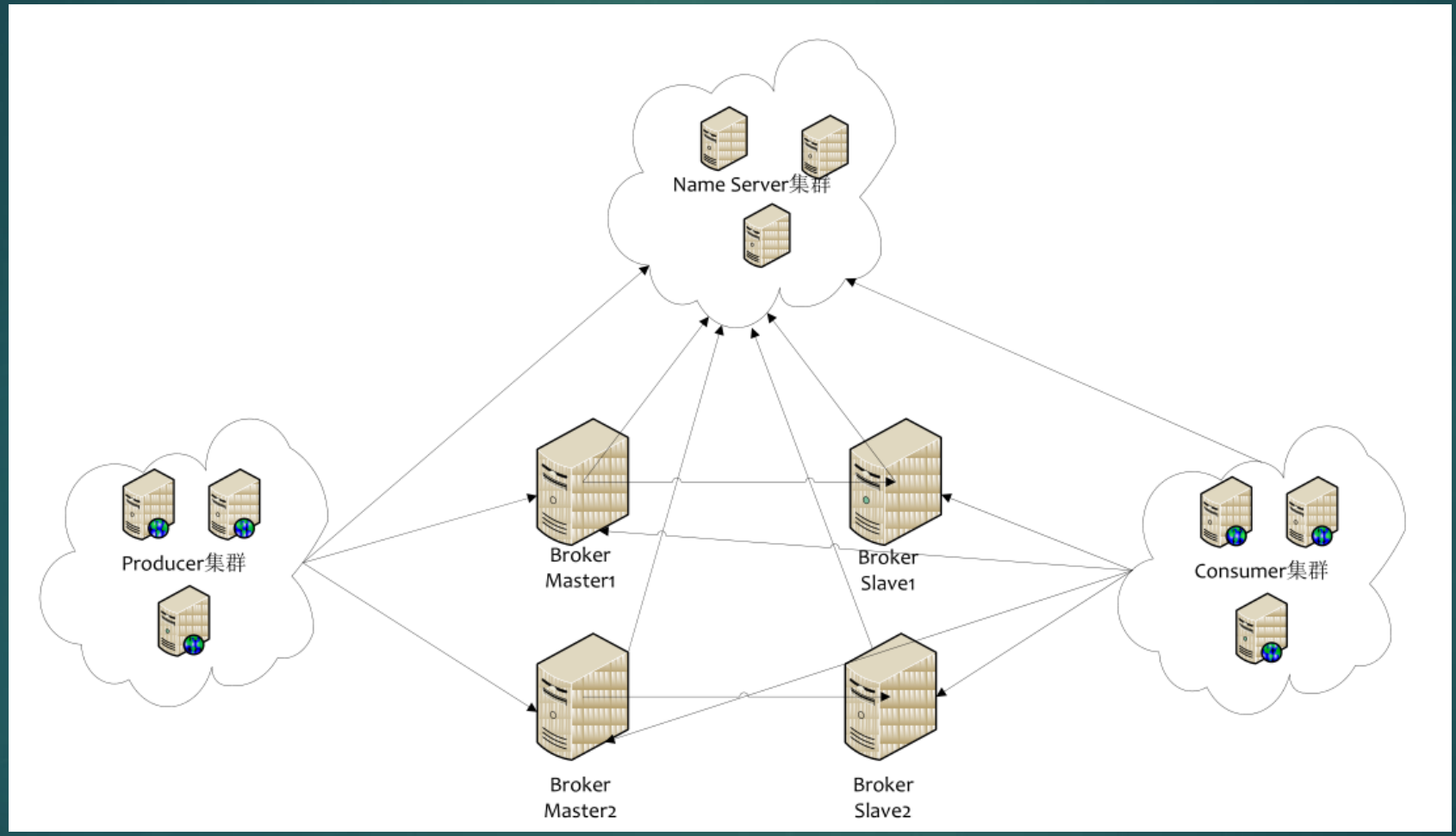
- 是一个队列模型的消息中间件，具有高性能、高可靠、高实时、分布式特点。
- **Producer**、**Consumer**、**broker**都可以分布式。
- **Producer** 向一些队列轮流发送消息，队列集合称为 **Topic**，**Consumer** 如果做广播消费，则一个 **consumer**实例消费这个 **Topic** 对应的所有队列，如果做集群消费，则多个 **Consumer** 实例平均消费这个 **topic** 对应的队列集合。
- 能够保证严格的消息顺序
- 提供丰富的消息拉取模式
- 高效的订阅者水平扩展能力
- 实时的消息订阅机制
- 亿级消息堆积能力
- 较少的依赖

消息可靠，可用性保证

- 影响消息可靠性的几种情况：
- **(1). Broker** 正常关闭
- **(2). Broker** 异常 **Crash**
- **(3). OS Crash**
- **(4).** 机器掉电，但是能立即恢复供电情况。
- **(5).** 机器无法开机（可能是 **cpu**、主板、内存等关键设备损坏）
- **(6).** 磁盘设备损坏。
- **(1)**、**(2)**、**(3)**、**(4)**四种情况都属于硬件资源可立即恢复情况，**RocketMQ** 在这四种情况下能保证消息不丢，或
- 者丢失少量数据（依赖刷盘方式是同步还是异步）。
- **(5)**、**(6)**属于单点故障，且无法恢复，一旦发生，在此单点上的消息全部丢失。**RocketMQ** 在这两种情况下，通
- 过异步复制，可保证 **99%**的消息不丢，但是仍然会有极少量的消息可能丢失。通过同步双写技术可以完全避免单点，
- 同步双写势必会影响性能，适合对消息可靠性要求极高的场合，例如与 **Money** 相关的应用。
- **RocketMQ** 从 **3.0** 版本开始支持同步双写。**(**目前最新稳定版是**:3.2.6)**

人人聚财

rocketMQ部署



人人聚财

rocketMQ性能

rocketmq性能报告

结论，单机**tps 7**万/秒，完全满足未来业务需要。

人人聚财

broker物理硬件要求

- **CPU** 两颗**x86_64cpu,**每颗**cpu12**核，共**24**核
- 内存 **48G**
- 网卡 千兆
- 磁盘 **RAID10 SAS 15000 1T**
- **Ext4** 文件系统
- 两台

RocketMQ版本

- 采用最新稳定版: **v3.2.6**
- 部署模式：双**master**结构(**nameserver**没有性能压力，可与**broker**部署在一起)
- 刷盘策略：同步刷盘
- **HA**策略：同步双写

运行之前的操作系统(centOS6.5)调优

```
#!/bin/sh
#
#  Execute  Only  Once  只可执行一次
#
#用户名
USER=admin
#磁盘盘符
DISK=sda
##在grub.conf中添加参数  默认注释掉  需要使用请取消注释
#sed  -i  's/kernel.*$/&  elevator=deadline/'  /etc/grub.conf

echo  'vm.overcommit_memory=1'  >>  /etc/sysctl.conf
echo  'vm.min_free_kbytes=5000000'  >>  /etc/sysctl.conf
echo  'vm.drop_caches=1'  >>  /etc/sysctl.conf
echo  'vm.zone_reclaim_mode=0'  >>  /etc/sysctl.conf
echo  'vm.max_map_count=655360'  >>  /etc/sysctl.conf
echo  'vm.dirty_background_ratio=50'  >>  /etc/sysctl.conf
echo  'vm.dirty_ratio=50'  >>  /etc/sysctl.conf
echo  'vm.page-cluster=3'  >>  /etc/sysctl.conf
echo  'vm.dirty_writeback_centisecs=360000'  >>  /etc/sysctl.conf
echo  'vm.swappiness=10'  >>  /etc/sysctl.conf
sysctl  -p
echo  "ulimit  -n  655350"  >>  /etc/profile
echo  "$USER  hard  nofile  655350"  >>  /etc/security/limits.conf
echo  'deadline'  >  /sys/block/$DISK/queue/scheduler
echo  "----------------------------------------------------------------"
sysctl  vm.overcommit_memory
sysctl  vm.min_free_kbytes
sysctl  vm.drop_caches
sysctl  vm.zone_reclaim_mode
sysctl  vm.max_map_count
sysctl  vm.dirty_background_ratio
sysctl  vm.dirty_ratio
sysctl  vm.page-cluster
sysctl  vm.dirty_writeback_centisecs
sysctl  vm.swappiness
su  -  $USER  -c  'ulimit  -n'
cat  /sys/block/$DISK/queue/scheduler
```

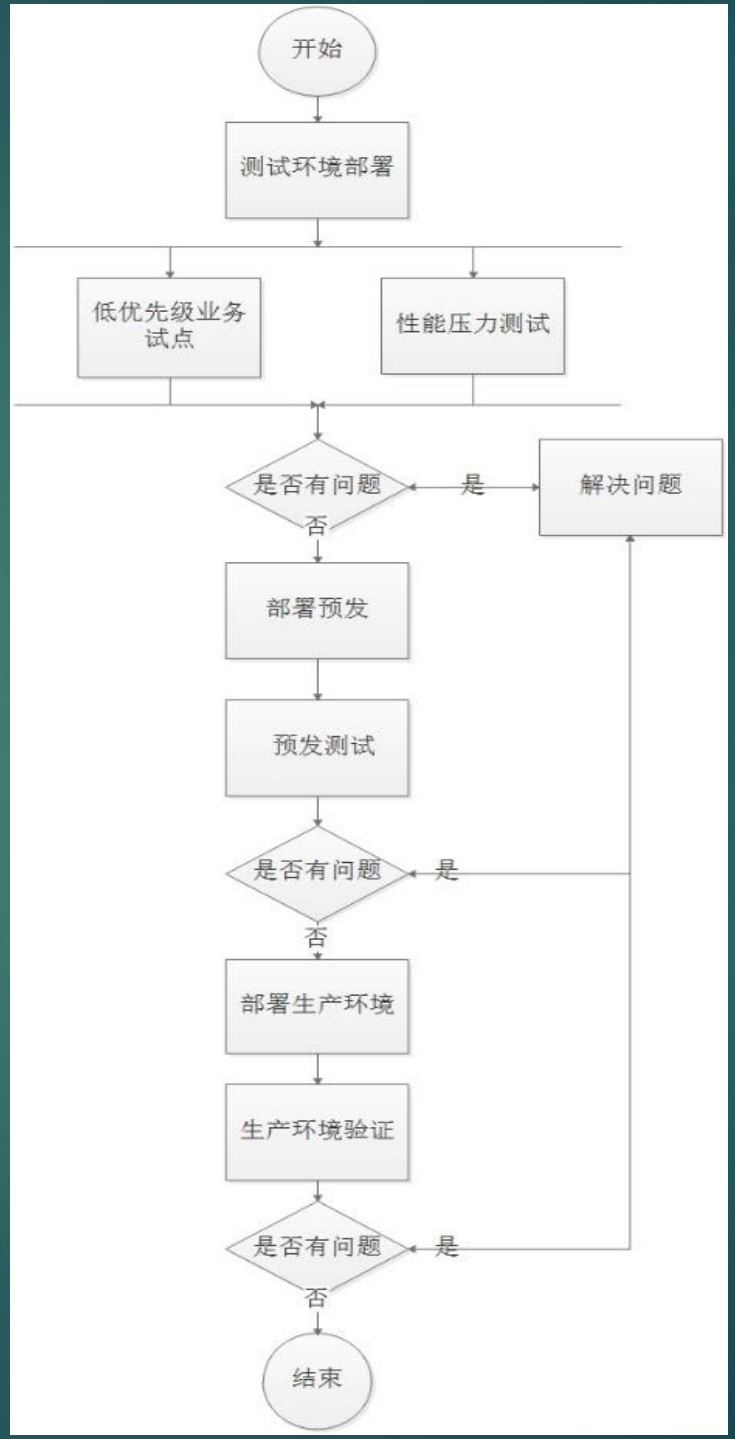
人人聚财

Jvm调优

```
-server -Xms4g -Xmx4g -Xmn2g -XX:PermSize=128m -
XX:MaxPermSize=320m -XX:+UseConcMarkSweepGC -
XX:+UseCMSCompactAtFullCollection -
XX:CMSInitiatingOccupancyFraction=70 -
XX:+CMSParallelRemarkEnabled -
XX:SoftRefLRUPolicyMSPerMB=0 -
XX:+CMSClassUnloadingEnabled -XX:SurvivorRatio=8 -
XX:+DisableExplicitGC -verbose:gc -
Xloggc:/root/rocketmq_gc.log -XX:+PrintGCDetails -XX:-
OmitStackTraceInFastThrow
```

人人聚财

执行计划


开始
测试环境部署
低优先级业务试点
性能压力测试
是否有问题
是
解决问题
否
部署预发
预发测试
是否有问题
是
否
部署生产环境
生产环境验证
是否有问题
是
否
结束

人人聚财