旁观者-郑昀

随笔-628 文章-0 评论-1096

曾任职明天高软、神州泰岳、窝窝团等公司。曾任玩聚网CEO和创始人。现任云纵首席科学家。中国语义应用探索者。智能语义聚合应用框架的提出者和实践者。

查看我的中文简历

昵称：旁观者
园龄：11年6个月
荣誉：推荐博客
粉丝：926
关注：0
+加关注

搜索

找找看

最新随笔

1. 全栈的好处：七天和两周
2. 如何成为一位牛逼的高手
3. 被生活一次又一次推倒的那些日子
4. 有些事儿，工程师可能今生仅此一次
5. 大BOSS随时都会到来
6. 私有云的难处
7. 今日长缨在手 何时缚住苍龙
8. 生无可恋的一叶知秋#百度刘超事件#
9. Cloud Engine：大杀器如何炼成
10. 人生做出的选择越多，友谊的小船翻得越快？

随笔分类(812)

ActiveMQ(4)
docker(5)
dotNET(2)
EnterpriseLibrary(8)
HybridApp(1)
Java(97)
LCS2005(5)
Lotus Domino(7)
MongoDB(4)
mysql(6)
PHP(7)
Python(25)
Redis(2)
Search Engine(4)
SIP(8)
Social(7)
sync4j(5)
WAP(8)
web2.0体验笔记(24)
电商课题(31)
个性化阅读(21)
技术英雄会(8)

#研发中间件介绍#异步消息可靠推送Notify

郑昀 基于朱传志的设计文档 最后更新于2014/11/11

关键词：异步消息、订阅者集群、可伸缩、Push模式、Pull模式

本文档适用人员：研发

电商系统为什么需要 **NotifyServer**？

如子柳所说，电商系统『需要两种中间件系统，一种是实时调用的中间件（淘宝的HSF，高性能服务框架）、一种是异步消息通知的中间件（淘宝的Notify）』。那么用传统的 ActiveMQ/RabbitMQ 来实现 异步消息发布和订阅 不行吗？

2013年之前我们确实用的是 ActiveMQ，当然主要是订阅者 Pull 模式，选 MySQL 做消息持久化存储，SA 还为此反复测试了各种高可用方案，如下图所示，ActiveMQ 5.8，mq主从+mysql互为主从+MMM。

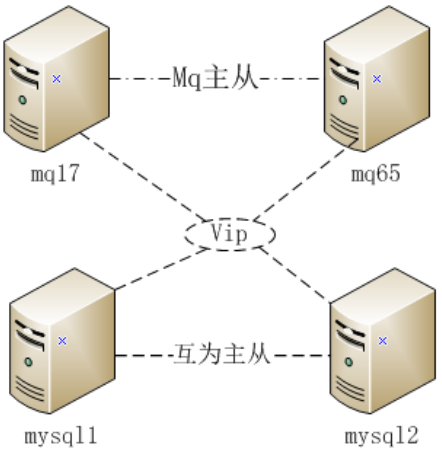

图1 mq 高可用

它有三个问题。

第一，它对上游发布者要求可能不是那么高，但要求下游实现消息订阅时要健壮，比如订阅者把消息读走了后它挂了也得不丢弃消息继续处理，比如非常重要的消息不能只有一个单点订阅者，必须有订阅者集群，但又不能重复处理消息。我在《#研发中间件介绍#JobCenter》中说过，对每一位开发者维护者提出高要求，这不是我们的解题思路。我在《职场培训第五期：职场的真相》中给出了解题思路：『要摒弃单纯依靠员工之间互相提醒、依靠个人认真细致来规避相同错误的固有思路，铁打营盘流水兵，靠人终归是靠不住的，最好靠遵循规则的机器』。是的，异步消息的可靠推送（Push），应该是消息中间件的职责。

第二，**ActiveMQ** 的高可用方案在可伸缩上不那么灵活，不适合电商业务。譬如说，我一开始用一组 [(mq1+mysql1(master角色)),(mq2+mysql2)] 来支撑所有业务的异步消息，但突然七夕节一个销售验证高峰即将到来，需要尽量平滑地把某些消息队列转移出去，用另一组支撑；或者我看某个消息队列的消息量比较大，想追加一个 mysql 节点单独存储它的消息。总之就是线上尽量平滑地扩容 mq server和 database，这事儿还得咱们自己从头搞才顺手。

最后一个问题是所有开源系统的典型问题，伴随着开源系统以及各种 **Driver** 的版本升级，我们会一路踏入它埋下的每一个大大小小的坑。当然，不是说我们自己写的中间件就没有 Bug，但 ActiveMQ 确实让人摊手，如下面的 RCA 案例所示。

- RCA：ActiveMQ 的生产者流量控制导致订单中心大量线程挂起；
- RCA：PHP连MQ超时导致主库连接被打满，引发众多应用数据不一致——原因在于 PHP::Stomp 包的默认重试次数和默认超时时间；
- RCA：调小 ActiveMQ之持久化 MySQL 的 wait_timeout 导致发送 MQ 消息频频失败。

最终我们还是选择自己来面对如下场景，采用 Push 模式（NotifyServer 主动向下游 Push 消息）：

最新评论

阅读排行榜

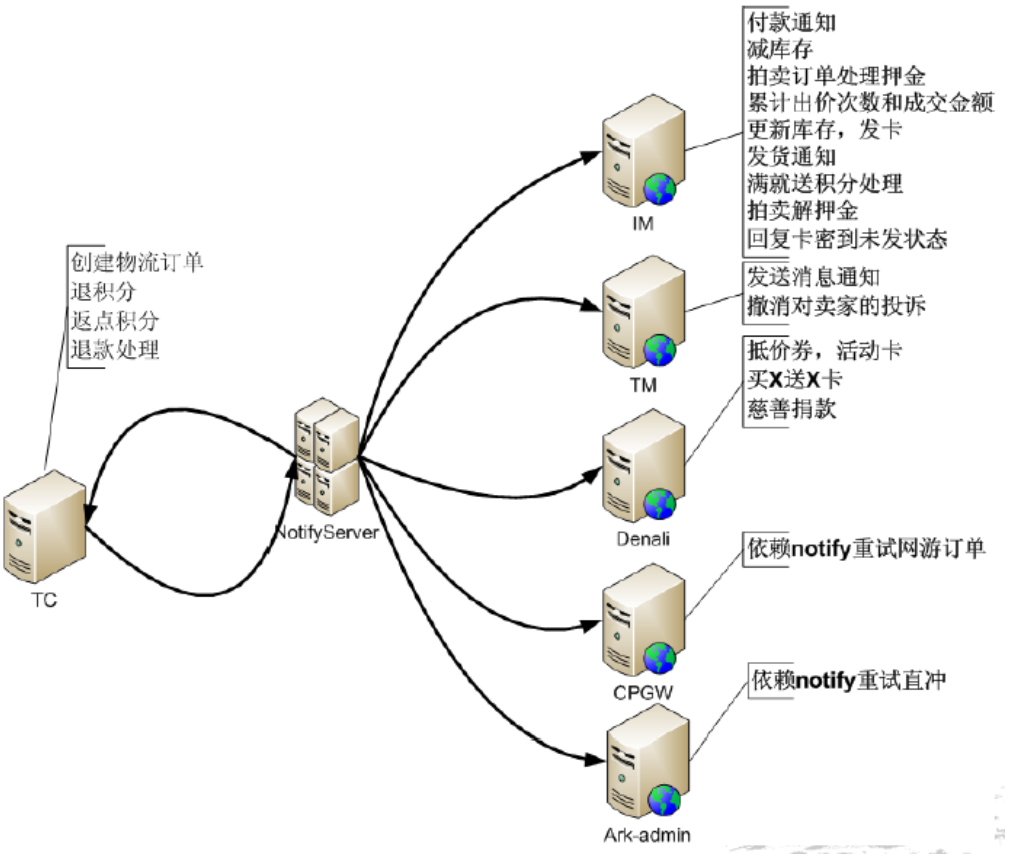

图2 一个异步消息需要很多订阅者集群分头处理

淘宝是怎么考虑这些问题的？

- 可靠性：
  - 消息的投递分为两个阶段
    - 发布者向Broker发送消息
    - Broker向订阅者投递消息
  - 因此，消息有可能在三个地方丢失
    - 发布者到Broker之间
    - Broker本身
    - 从Broker到订阅者
- 稳定性
  - 监视
    - Broker内存使用
    - 消息收发功能
    - 消息堆积情况
    - 存储的插入速度
    - 各个任务队列长度
    - 其他各项即时统计数据等
  - 控制
    - 自动移除失效存储节点
    - 优雅降级的控制
    - 添加新存储节点
    - 添加新**Broker**

媒体协议，下载LiveMediaVideo[1][修正版，增加了带宽测试包](23123)
9. [J2ME]手机看交通监视器实时录像 实现说明(19828)
10. #研发解决方案介绍#Tracing（鹰眼）(19148)

评论排行榜

1. [J2ME]手机看交通监视器实时录像 实现说明(62)
2. [SMS&WAP]实例讲解制作OTA短信来自动配置手机WAP书签[附源码](36)
3. 郑昀的2016中文简历(34)
4. 推荐阅读：《我在赶集网的两个月（完整版）》(31)
5. [J2ME]手机流媒体之实作[附源码][与RTSP/MMS协议无关](31)

- 限制
  - 有可能产生重复消息
  - 对订阅者的要求
    - 幂等性 f(f(x)) = f(x)
    - 重复调用多次产生的业务结果与调用一次产生的业务结果相同

它内部两个消息中间件产品的区别为：

	Notify	Metamorphosis
模型	Push	Pull
服务端	消息存储 处理请求 保存推送轨迹 保存订阅关系 消费者负载均衡 集中式	消息存储 处理请求 分布式
客户端	处理响应和请求	处理响应和请求 保存pull状态，如拉取位置的偏移量offset 异常情况下的消息暂存和recover
实时性	较好，收到数据后可立即发送给客户端	取决于pull的间隔时间
消费者故障	消费者故障情况下，服务端堆积消息，重复推送耗费资源。 保存推送轨迹压力很大。	消费者故障，对服务端无影响
其他	对消息推送有更多控制，能实现多样化的推送机制。 当消费者数量增多的时候，推送压力大，性能天花板。 消费者处理能力差异，导致堆消息	需要在客户端实现消息过滤，浪费资源。 需要在不同客户端之间协调，做负载均衡。

图3 消息中间件对比

以上资料出自于《消息中间件-Notify的概念和原理.pdf》。

窝窝如何实现 **NotifyServer** 的？

2013年2月，经过几轮的讨论，技术选型初步确定，研发2部传志开始构建 NotifyServer。

他设计了如下概念：

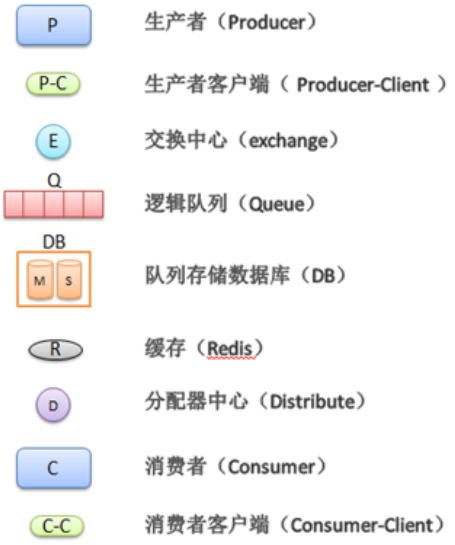

图4 notifyserver 的几个角色概念

技术模型可以描述为：

- 模块关系
  - 各个模块（队列、生产者、交换中心、DB、消息体缓存、队列缓存、日志缓存、分配中心、消费者）存在一定的对应关系，通过这些对应关系能够更好路由和分流消息，动态扩展系统，改善系统瓶颈。
  - 这些对应关系都存储在控制中心关系数据库中，通过控制台界面来进行配置，各模块在启动和定时到控制中心来更新这些关系，用于消息的分配。
  - 这些关系都遵守一定规则，添加更改不会影响系统的稳定性，如：一个队列必须对应两个以上的交换中心来处理消息，如果DB中还有消息没有消费完毕不允许直接删除，等等。
- 模块监控

- 控制台定期测试各个模块的健康状况。
- 各个模块会定期向控制中心发送一些监控数据，报告自己的运行状态。
- 控制台收集监控数据，以图表、拓扑图等形式向管理人员展示或报警。

- 消息跟踪
  - 每一条消息在进入系统后都会被分配一个唯一标识。
  - 各模块在处理消息时都会产生特定的日志信息，日志信息实时的传送到日志系统。
  - 唯一标识+日志+各模块信息和关系可以容易的跟踪每一条消息的执行情况。

那么最简单的消息消费泳道图如下所示：

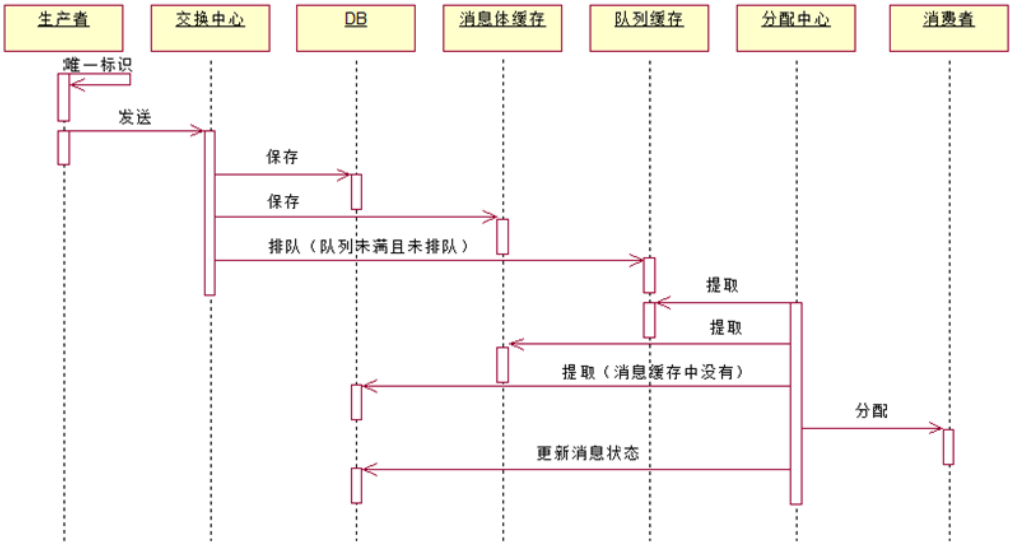

图5 消息消费

分配中心主导的慢速/重试Push，它会尽量从缓存（Redis）中拿消息体，尽量减少对 DB 的访问，尤其是消息体特别大的时候，效果会比较明显，如下图所示。

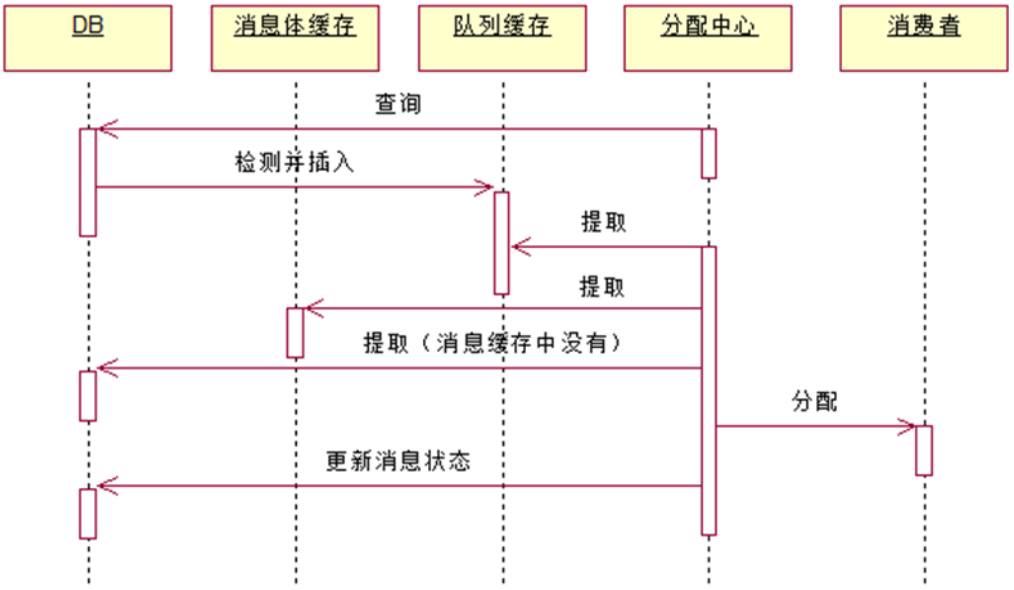

图6 重试的泳道图

对于可伸缩、高可用，他是这么考虑的：

- 吞吐量：
  - 交换中心、分配中心都采用平行结构。
  - 队列使用持久化方式缓存，使用缓存减少对DB的操作。
  - 交换中心与分配中心分离，性能互不影响。
  - 动态改变网络拓扑结构，分流系统瓶颈。

    - 动态控制吞吐量参数，调整系统性能。
- 扩展性
    - 交换中心、分配中心、DB、缓存可以动态添加或删除。
    - 队列路由路径可以动态改变。
- 可用性
    - 交换中心、分配中心采用平行方式提高可用性。
    - DB采用 master-master 方式提高可用性。
    - 缓存采用多点读写方式提过可用性。
- 一致性
    - 消息状态持久化在DB，未分配或消费失败的会再次被提取。
    - 分配中心采用快慢两种方式接收消息处理消息。
- 等幂性
    - 缓存保存正在处理的消息，防止重复分配。

与 JobCenter 一样，NotifyServer 也纳入在我们的 idcenter 体系下，这样可以共用一套帐号体系（LDAP），共用一个统一的权限分配：

图7 notifyserver 的入口

图8 notifyserver 的主界面

图9 notifyserver 系统队列（主要是配置信息）界面

图10 notifyserver 监控队列（主要是运行时状况）界面

2013年中旬，经过积分业务的试用后，传志的 NotifyServer 开始在内部推广，各种异步消息发布和订阅一点一点地搬进来，ActiveMQ 方案下线。

-over-

窝窝的解决方案介绍列表：

#研发解决方案#基于StatsD+Graphite的智能监控解决方案

#研发中间件介绍#定时任务调度与管理JobCenter

#研发解决方案介绍#Recsys-Evaluate（推荐评测）

#研发解决方案介绍#Tracing（鹰眼）

#研发解决方案介绍#基于持久化配置中心的业务降级

#研发中间件介绍#异步消息可靠推送Notify

#研发解决方案介绍#IdCenter（内部统一认证系统）

#研发解决方案介绍#基于ES的搜索+筛选+排序解决方案

#数据技术选型#即席查询Shib+Presto，集群任务调度HUE+Oozie

欢迎订阅我的微信订阅号『老兵笔记』，请扫描二维码关注：

分类：电商课题,研发解决方案

标签： notify， mq， java， 中间件

好文要顶 关注我 收藏该文

旁观者
关注 - 0
粉丝 - 926

0 0

荣誉：推荐博客

+加关注

« 上一篇：#研发解决方案介绍#基于持久化配置中心的业务降级
» 下一篇：#研发解决方案介绍#IdCenter（内部统一认证系统）

posted @ 2014-12-12 17:37 旁观者 阅读(10730) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新**IT**新闻：
· Siri必须借AI提高"智商" 否则很快就会被对手甩开
· 复盘美团点评收购钱袋宝：孙江涛的创投反思
· iPhone能诊断癌症？看看美国学者做的这款移动实验室
· 摩拜单车CEO王晓峰：我只是想让更多人骑车出行
· 物联网再次被黑客利用，厂商应重视智能设备的安全问题
» 更多新闻...

最新知识库文章：
· 陈皓：什么是工程师文化？
· 没那么难，谈CSS的设计模式
· 程序猿媳妇儿注意事项
· 可是姑娘，你为什么要编程呢？
· 知其所以然（以算法学习为例）
» 更多知识库文章...

历史上的今天：
2009-12-12 围观一个People Search
2008-12-12 随手小记：快速适应未必是个好策略