消息队列 CMQ 版

案例分享

腾讯云

【联系我们】

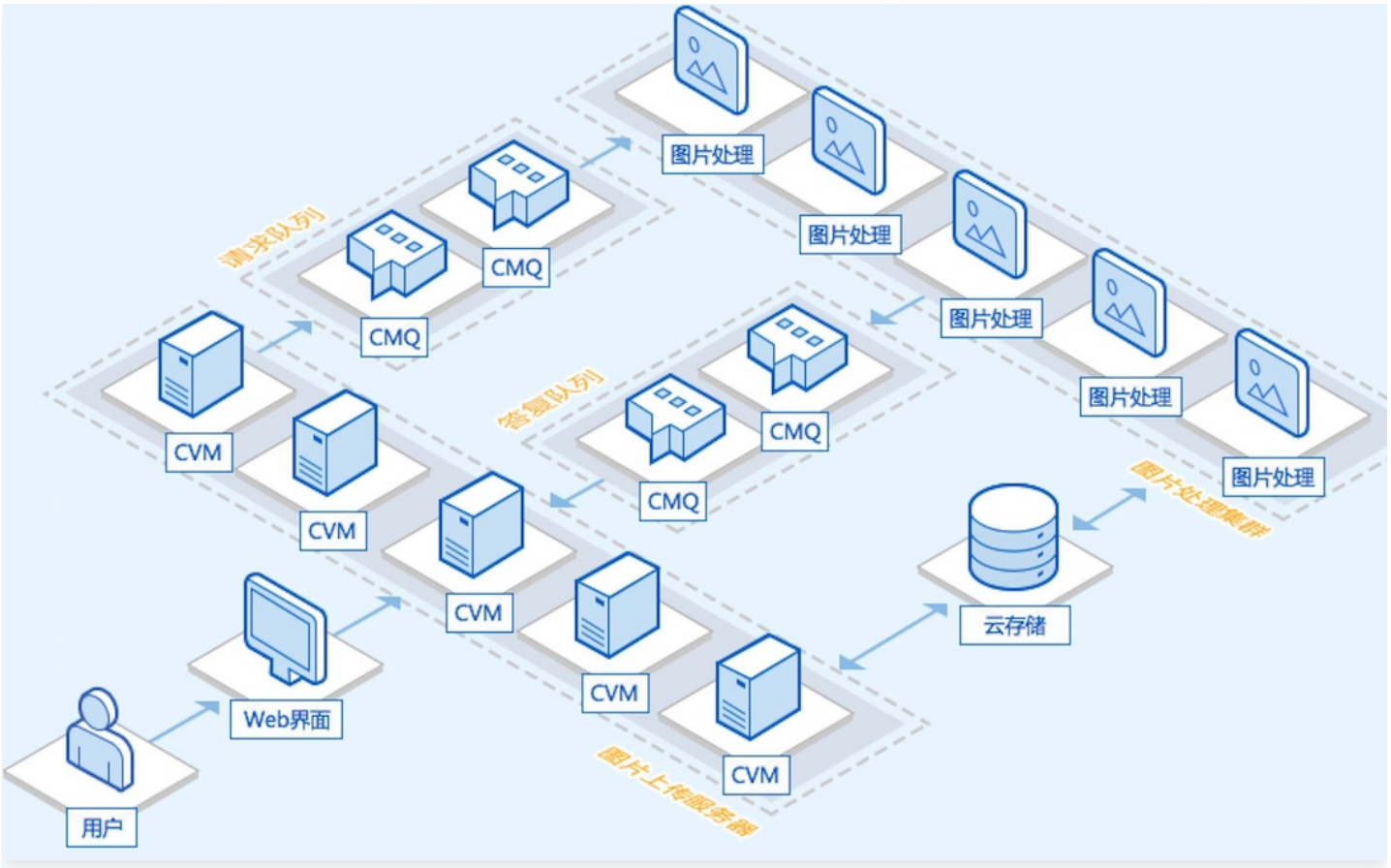
我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

案例分享
在线图片处理案例

最近更新时间：2024-10-14 15:11:12

某美图公司在腾讯云搭建了在线图片处理服务，该服务可以让用户上传照片，并指定需要对这些照片执行的操作，例如裁剪、红眼处理、牙齿美白、重新着色、对比度调节、生成缩略图等。用户上传图片后，提交任务，然后等待图片处理完，下载处理后的图片。不同的操作会耗费不同的处理时间，从几秒到几分钟不等，而且用户可能一次上传几张也可能是几十张甚至几百张图片，所以总的处理时间就跟上传的图片个数、图片的大小、用户选择的操作有关。



使用 TDMQ CMQ 版实现了上述需求，用户的图片存储在腾讯云存储中（CBS/COS等），用户的每一个操作请求都会作为一个消息存入请求队列（Request Queue）中，消息内容为：图片索引，由图片名称+用户请求的操作类型+图片存储的位置索引 key 等组成。

运行在 CVM 的图片处理服务从 Request Queue 中获取消息（图片索引），图片处理服务器从云端下载数据，并进行图片编辑，完毕后把处理结果发送到结果队列（Response Queue）中，结果图片存储到云存储中。流程结束，客户已将原图片、编辑处理后的图片，都存储在云端存储，可随时下载使用。

可扩展的、高可靠的进一步思考：

- 如果因为 bug 或其他原因导致图片处理服务暂时不可用。但是系统利用 TDMQ CMQ 版使得错误对用户透明，一方面用户可以继续上传照片，web server 可以继续发消息到 Request Queue，消息会被保存在队列中直到图片处理服务可用后取走；另一方面。图片处理服务在实现时不用记住崩溃前在处理的消息，而且其崩溃时处理的消息还可以被重新处理。因为 TDMQ CMQ 版提供的接收消息（包括接收顺序队列消息和接收并发队列消

息）特性保证消息在接收后仍然在队列中，直到消息的接收者显式来删除它。本特性保证了图片处理服务与图片上传服务的解耦。

- 如果单个图片处理服务不能满足用户需求（用户虽然能够上传照片，但是却长时间拿不到处理的结果），利用 TDMQ CMQ 版并启动多个图片处理服务便可以满足不断增长的用户访问需求。 TDMQ CMQ 版的两个特性让这个需求成为可能：
  - 单个 TDMQ CMQ 版队列是可以让多个 server 同时共享访问的（即发送消息、接收并发队列消息、删除并发队列消息功能）。
  - 一个消息不会同时被多个服务接收，这是通过针对消息的短暂锁来保证的，消息的接收者可以指定消息被锁定的时间，接收者处理完消息需要主动删除消息，如果接收者处理消息失败，那么另一个服务可以在这个消息的锁失效后重新获得这个消息。

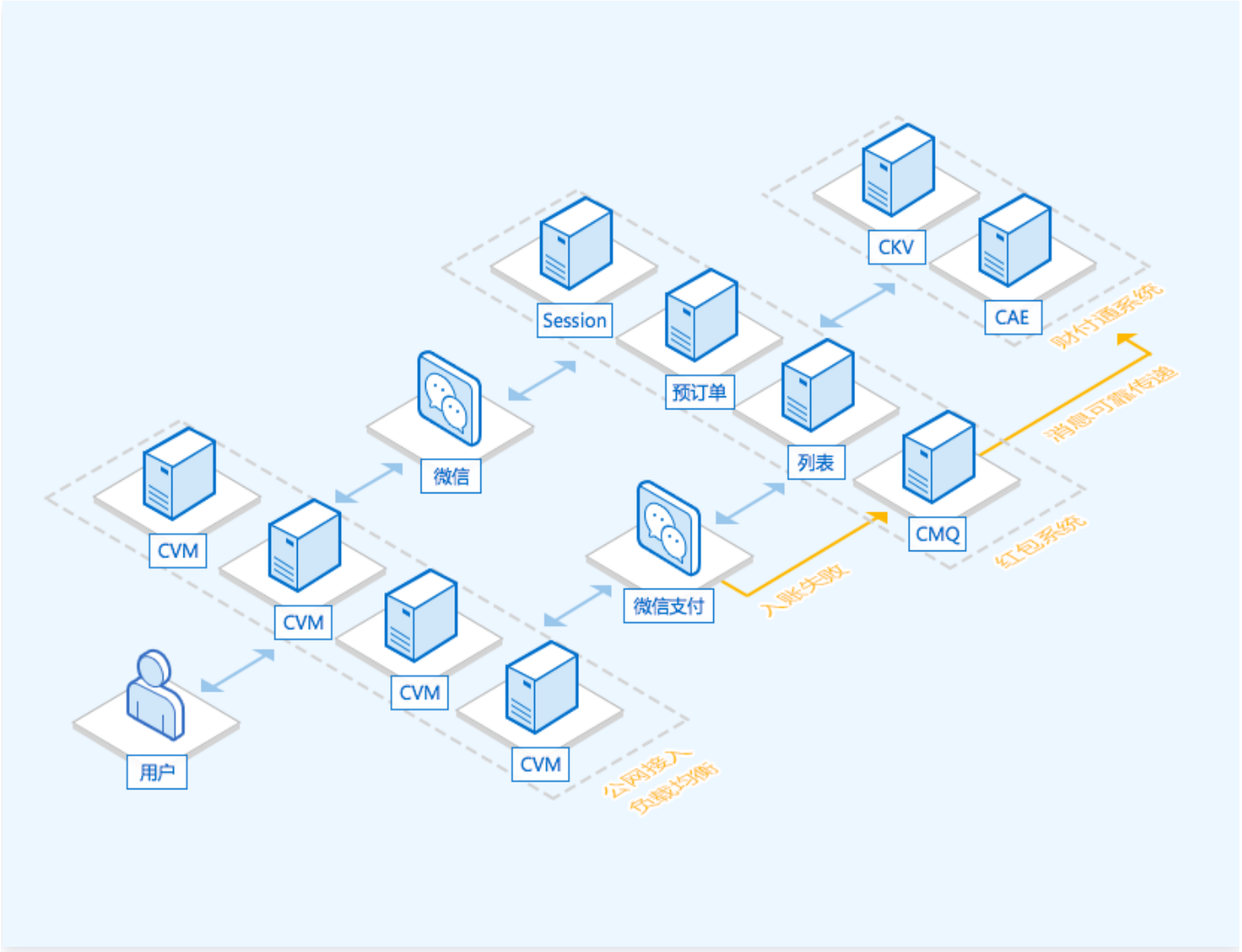
这两个特性保证了处理服务器的数量可以随着负载的变化而动态加减。

春晚微信红包案例

最近更新时间：2024-10-14 10:43:31

春晚红包活动涉及四个大型系统的联动，包括微信、微信支付、红包系统和财付通系统。以下简单介绍各个系统：

- 红包系统：个人红包的发、抢、拆和列表查看；
- 财付通系统：包括支付订单、异步入账流水的高性能存储，用户余额和账单的实时展示；
- 微信接入：确保微信用户公网接入的质量；
- 微信支付：在线交易的入口。



类似红包系统的分布式事务是关注的热点。举一个典型的例子，『用户 A 给用户 B 发了10元的红包』，有以下步骤：

1. 从 A 账号中把余额读出来。
2. 对 A 账号做减法操作（减10元）。

3. 把结果写回 A 账号中（一次确认）。
4. 从 B 账号中把余额读出来。
5. 拆开 A 发送给 B 的红包，读出数值。
6. 对 B 账号做加法操作（加10元）。
7. 把结果写到 B 账号中。

为了保证数据的一致性，上述步骤只有两种结果：都成功完成或者都不成功执行回滚。而且这个操作的过程中，对A、B 账号还需引入分布式锁机制来避免脏数据的问题。在微信红包这个庞大的分布式集群内，事情将变的异常复杂。

微信红包系统引入了 TDMQ CMQ 版以避免分布式事务增加对系统的开销。同样 A 用户给 B 用户发10元红包的场景，下面介绍引入TDMQ CMQ 版后的新策略。

- 在上述案例中的第七步，B 用户拆开了红包，红包里有10块钱。在做最后的入账操作时由于当天并发压力大，常出现入账失败的情况。
- 红包团队把入账失败的请求，全部转入TDMQ CMQ 版。当B用户更新账户余额失败时，手机客户端显示等待状态。随后账户系统将不断从TDMQ CMQ 版重新拉取重试此更新操作。TDMQ CMQ 版保证了这 10 元的入账消息永远不丢，直至它被取出。
- 在除夕当天，用户红包的发、拆、入账等动作，转化为了十亿级别的海量请求。若使用传统的事务方式，会放大并发压力使系统崩溃。
- TDMQ CMQ 版消息队列保证了红包消息的可靠存储、传递，实时写三份保证数据不丢。资金入账失败时可多次重试，避免失败回滚和频繁轮询数据库等传统方式的弊端。

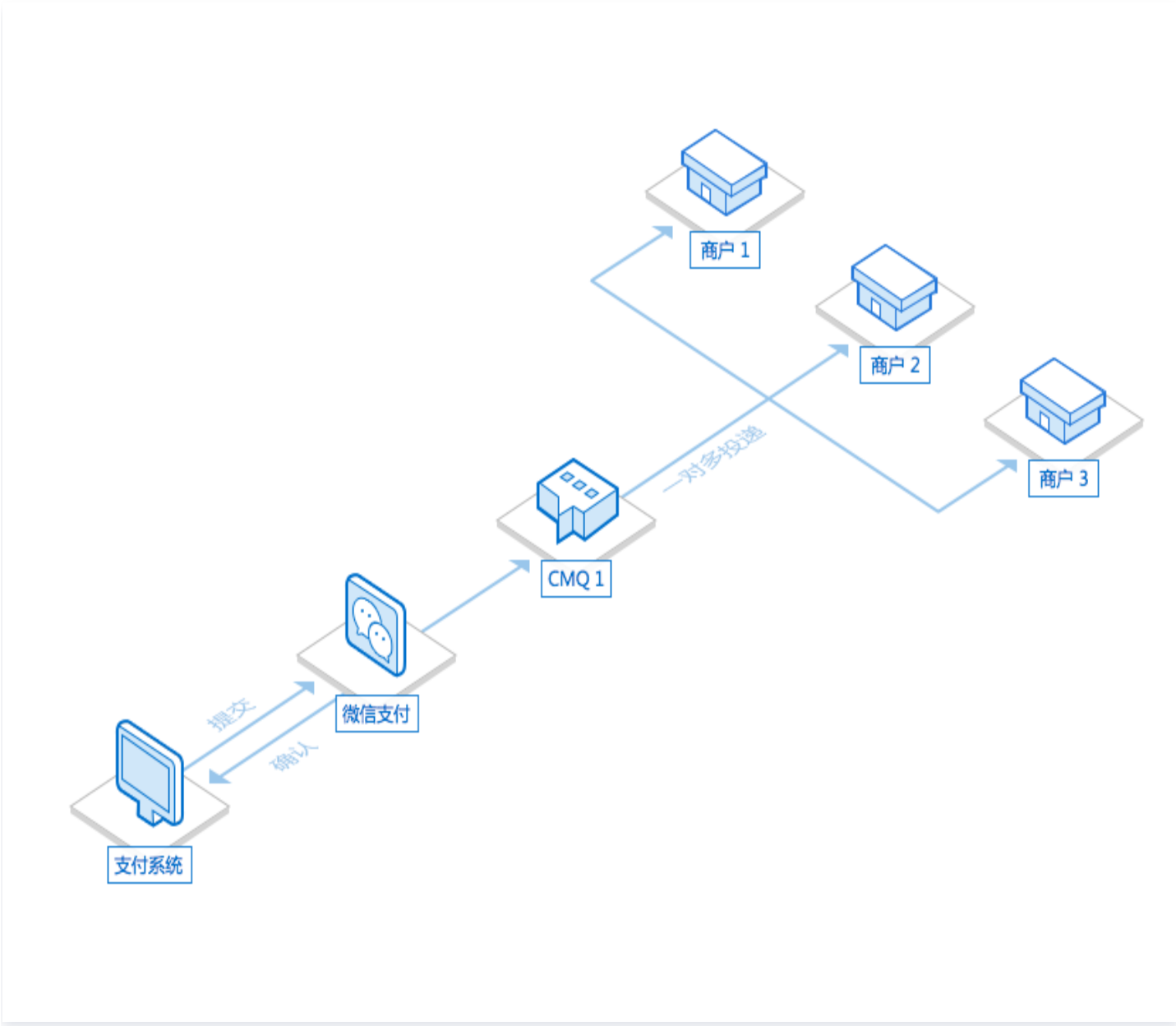
第三方支付案例

最近更新时间：2024-10-14 16:30:22

与微信支付紧密合作的第三方移动金融支付解决方案提供商，如深圳威富通等，促进了全国各行各业的线下商铺的发展，通过微信支付，提高效率，免除现金结算的低效率。支付系统主要架构如下：

1. 老百姓在便利店购物（如7-11）提交的支付请求，发送给微信支付，微信支付确认后会返回 ACK。
2. 返回的 ACK 确认后，微信支付系统会下发一条**订单支付成功的消息**，详细说明了消费的账户信息、时间、金额，终端信息等。该消息会发给威富通。
3. 威富通将该明细，写入 TDMQ CMQ 版，用作暂存。**订单支付成功的消息**作为威富通与商家（便利店）之间结算的重要凭证，必须可靠传递，保证不丢。
4. 异步的，将 TDMQ CMQ 版内的**订单支付消息**，返回给多商家的服务器（便利店），这个返回的过程不需要及时，可以异步处理。具体怎么做呢？是将该消息写到 Queue 里，然后一个 HTTP 代理，来拉消息。取出后，HTTP 发给商家。
5. 在未接入 TDMQ CMQ 版之前，假如威富通通知商户失败，威富通会重新向微信支付发起请求，微信随后会再次将同样的**订单支付消息**投递给威富通。接入 TDMQ CMQ 版之后，从微信的角度来看，威富通系统的成功率提升不少，微信对其评级会提升（可靠性、信誉）。
6. 最后，每笔**订单支付消息**，由另一个 Topic 不断向风控管理、活动管理、促销活动等系统投递。例如风控管理会持续分析 Topic 投递的每一笔订单支付情况，当商家 A 在短时间内交易额大幅上涨时（刷单嫌疑），会用回调接口，禁止商家 A 的后续交易。

参考图示如下：



起点文学网案例

最近更新时间：2024-10-14 15:11:25

阅文集团旗下的起点文学网，使用 TDMQ CMQ 版满足了3个核心需求：

1. 『仗义书财』的运营系统，里面抢红包月票的功能，消费者入账的时候是异步的。入账信息会先写到 MQ 里。消费者过来拉，且消费者确认已成功消费后，回调接口把 MQ 里的信息删掉。
2. 另一个场景是，起点文学网的各大系统，包括运维、告警、运营系统的日志流水，会先聚合到 TDMQ CMQ 版中，后端的大数据分析集群，会按处理能力，不断到 TDMQ CMQ 版中拉去，分析。 TDMQ CMQ 版理论上支持的消息堆积数量无上限，使用无后顾之忧。
3. 提供类似于 Kafka 的消息回溯能力。当业务成功消费，并删除消息后，使用消息回溯，可重新消费已删除的消息。可指定 offset 的位置进行调整。这便于起点文学网，做账单的对账、业务系统重试等。

起点文学的整体业务对 TDMQ CMQ 版的压力，API 请求的 QPS 超过10万，全天请求量超10亿次，客户担忧如此大的业务压力，TDMQ CMQ 版是否能稳定支持?

TDMQ CMQ 版后端的集群对用户来说是透明无感知的， TDMQ CMQ 版 controller server 可根据集群的负载情况实时对 queue 进行调度搬迁。如果某个 queue 的请求量超过当前集群的服务阈值，controller server 可以将 queue 路由分布到多个集群上来提高并发量，理论上可以达到无限的消息堆积以及超高的 QPS。

参考图示如下：