阿里云
消息队列 RocketMQ 版

产品简介

文档版本：20200606

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务时间约十分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 **注意：** 权重设置为0，该服务器不会再接受新请求。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	单击**设置** > **网络** > **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令。	执行cd /d C:/window命令，进入Windows系统文件夹。
斜体	表示参数、变量。	bae log list --instanceid Instance_ID
[]或者[a|b]	表示可选项，至多选择一个。	ipconfig [-all|-t]
{}或者{a|b}	表示必选项，至多选择一个。	switch {active|stand}

目录

法律声明.....I
通用约定.....I
1 什么是消息队列 RocketMQ 版？ 1
2 功能与特性..... 4
2.1 功能与特性概述.....4
3 适用场景..... 6
4 产品架构..... 14
5 名词解释..... 16
6 场景化案例..... 20
6.1 权益分发.....20
7 使用限制..... 22

1 什么是消息队列 RocketMQ 版?

消息队列 RocketMQ 版是阿里云基于 Apache RocketMQ 构建的低延迟、高并发、高可用、高可靠的分布式消息中间件。消息队列 RocketMQ 版既可为分布式应用系统提供异步解耦和削峰填谷的能力，同时也具备互联网应用所需的海量消息堆积、高吞吐、可靠重试等特性。

核心概念

- Topic：消息主题，一级消息类型，生产者向其发送消息。
- 生产者：也称为消息发布者，负责生产并发送消息至 Topic。
- 消费者：也称为消息订阅者，负责从 Topic 接收并消费消息。
- 消息：生产者向 Topic 发送并最终传送给消费者的数据和（可选）属性的组合。
- 消息属性：生产者可以为消息定义的属性，包含 Message Key 和 Tag。
- Group：一类生产者或消费者，这类生产者或消费者通常生产或消费同一类消息，且消息发布或订阅的逻辑一致。

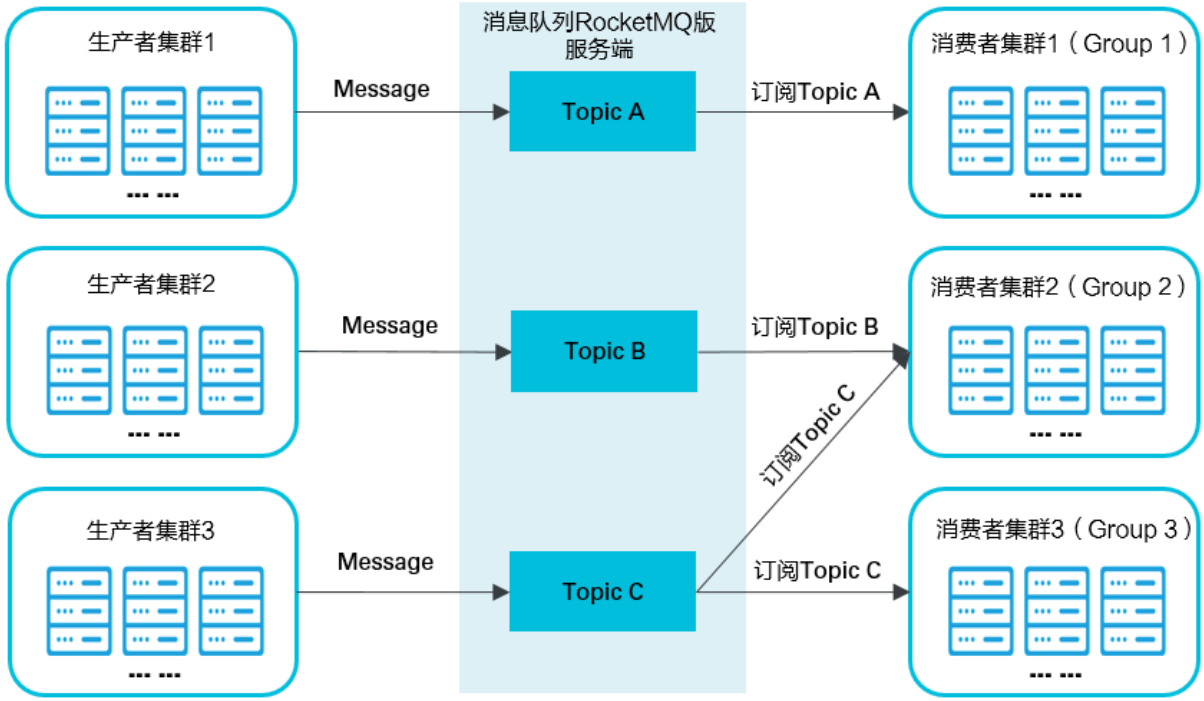
消息队列 RocketMQ 版涉及的概念的详细解释，请参见名词解释。

消息收发模型

消息队列 RocketMQ 版支持发布/订阅模型，消息生产者应用创建 Topic 并将消息发送到 Topic。消费者应用创建对 Topic 的订阅以便从其接收消息。通信可以是一对多（扇出）、多对一（扇入）和多对多。

具体通信如下图所示。

图 1-1: 消息收发模型



- 生产者集群：用来表示发送消息应用，一个生产者集群下包含多个生产者实例，可以是多台机器，也可以是一台机器的多个进程，或者一个进程的多个生产者对象。

  一个生产者集群可以发送多个 Topic 消息。发送分布式事务消息时，如果生产者中途意外宕机，Broker 会主动回调生产者集群的任意一台机器来确认事务状态。
- 消费者集群：用来表示消费消息应用，一个消费者集群下包含多个消费者实例，可以是多台机器，也可以是多个进程，或者是一个进程的多个消费者对象。

  一个消费者集群下的多个消费者以均摊方式消费消息。如果设置的是广播方式，那么这个消费者集群下的每个实例都消费全量数据。

  一个消费者集群对应一个 Group ID，一个 Group ID 可以订阅多个 Topic，如图 1-1: 消息收发模型 中的 Group 2 所示。Group 和 Topic 的订阅关系可以通过直接在程序中设置即可，具体设置方法可参见#unique_5中的资源申请流程优化部分。

应用场景

- 削峰填谷

  诸如秒杀、抢红包、企业开门红等大型活动时皆会带来较高的流量脉冲，或因没做相应的保护而导致系统超负荷甚至崩溃，或因限制太过导致请求大量失败而影响用户体验，消息队列 RocketMQ 版可提供削峰填谷的服务来解决该问题。

- 异步解耦

  交易系统作为淘宝/天猫主站最核心的系统，每笔交易订单数据的产生会引起几百个下游业务系统的关注，包括物流、购物车、积分、流计算分析等等，整体业务系统庞大而且复杂，消息队列 RocketMQ 版可实现异步通信和应用解耦，确保主站业务的连续性。

- 顺序收发

  细数日常中需要保证顺序的应用场景非常多，例如证券交易过程时间优先原则，交易系统中的订单创建、支付、退款等流程，航班中的旅客登机消息处理等等。与先进先出（First In First Out，缩写 FIFO）原理类似，消息队列 RocketMQ 版提供的顺序消息即保证消息 FIFO。

- 分布式事务一致性

  交易系统、支付红包等场景需要确保数据的最终一致性，大量引入消息队列 RocketMQ 版的分布式事务，既可以实现系统之间的解耦，又可以保证最终的数据一致性。

- 大数据分析

  数据在“流动”中产生价值，传统数据分析大多是基于批量计算模型，而无法做到实时的数据分析，利用阿里云消息队列 RocketMQ 版与流式计算引擎相结合，可以很方便的实现将业务数据进行实时分析。

- 分布式缓存同步

  天猫双 11 大促，各个分会场琳琅满目的商品需要实时感知价格变化，大量并发访问数据库导致会场页面响应时间长，集中式缓存因为带宽瓶颈限制商品变更的访问流量，通过消息队列 RocketMQ 版构建分布式缓存，实时通知商品数据的变化。

更多信息请参见适用场景。

2 功能与特性

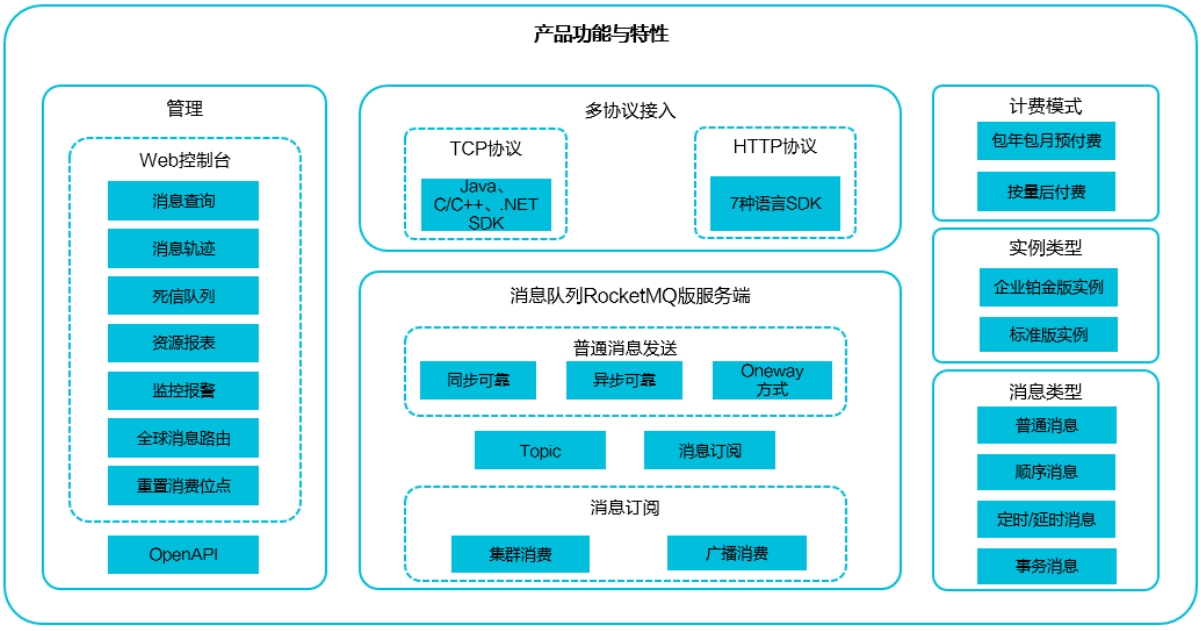
2.1 功能与特性概述

本文列举消息队列 RocketMQ 版所支持的所有功能与特性。

概览

消息队列 RocketMQ 版在阿里云多个地域（Region）提供了高可用消息云服务。单个地域内采用多机房部署，可用性极高，即使整个机房都不可用，仍然可以为应用提供消息发布服务。

消息队列 RocketMQ 版提供 TCP 和 HTTP 协议的多语言接入方式，方便不同编程语言开发的应用快速接入消息队列 RocketMQ 版消息云服务。您可以将应用部署在阿里云 ECS、企业自建云，或者嵌入到移动端、物联网设备中与消息队列 RocketMQ 版建立连接进行消息收发；同时，本地开发者也可以通过公网接入消息队列 RocketMQ 版服务进行消息收发。



多协议接入

- HTTP 协议：采用 RESTful 风格，方便易用，快速接入，跨网络能力强。支持 Java、C++、.NET、Go、Python、Node.js 和 PHP 七种语言客户端。
- TCP 协议：区别于 HTTP 简单的接入方式，提供更为专业、可靠、稳定的 TCP 协议的 SDK 接入服务。支持的语言包括 Java、C/C++ 以及 .NET。

管理工具

- Web 控制台：支持 Topic 管理、Group 管理、消息查询、消息轨迹展示和查询、资源报表以及监控报警管理。
- OpenAPI：提供开放的 API 便于将消息队列 RocketMQ 版管理工具集成到自己的控制台。消息队列 RocketMQ 版的 API 详情请参见OpenAPI 参考。

消息类型

- #unique_12：消息队列 RocketMQ 版中无特性的消息，区别于有特性的定时和延时消息、顺序消息和事务消息。
- #unique_13：实现类似 X/Open XA 的分布事务功能，以达到事务最终一致性状态。
- #unique_14：允许消息生产者对指定消息进行定时（延时）投递，最长支持 40 天。
- #unique_15：允许消息消费者按照消息发送的顺序对消息进行消费。

特性功能

- #unique_16：消息队列 RocketMQ 版提供了三种消息查询的方式，分别是按 Message ID、Message Key 以及 Topic 查询。
- #unique_17：通过消息轨迹，能清晰定位消息从生产者发出，经由消息队列 RocketMQ 版服务端，投递给消息消费者的完整链路，方便定位排查问题。
- #unique_18：当使用集群消费模式时，消息队列 RocketMQ 版认为任意一条消息只需要被消费者集群内的任意一个消费者处理即可；当使用广播消费模式时，消息队列 RocketMQ 版会将每条消息推送给消费者集群内所有注册过的消费者，保证消息至少被每台机器消费一次。
- #unique_19：根据时间或位点重置消费进度，允许用户进行消息回溯或者丢弃堆积消息。
- #unique_20：将无法正常消费的消息储存到特殊的死信队列供后续处理。
- #unique_21：用于全球不同地域之间的消息同步，保证地域之间的数据一致性。
- #unique_22：消息生产和消费数据的统计功能。通过该功能，您可查询在一段时间范围内发送至某 Topic 的消息总量或者 TPS（消息生产数据），也可查询在一个时间段内某 Topic 投递给某 Group ID 的消息总量或 TPS（消息消费数据）。
- #unique_23：您可使用消息队列 RocketMQ 版提供的监控报警功能，监控某 Group ID 订阅的某 Topic 的消息消费状态并接收报警短信，帮助您实时掌握消息消费状态，以便及时处理消费异常。

3 适用场景

本文为您介绍消息队列 RocketMQ 版的适用场景，以便您更好地判断如何在业务中使用消息队列 RocketMQ 版。

例如，针对一家互联网电商企业，其业务涉及广泛，如注册、订单、库存、物流等；同时，也会涉及许多业务峰值时刻，如秒杀活动、周年庆、定期特惠等。这些活动都对分布性系统中的各项微服务应用的处理性能带来很大的挑战。

消息队列 RocketMQ 版作为分布式系统中的重要组件，可用于应对这些挑战，例如解决应用的异步解耦。

下文先以用户注册为场景说明消息队列 RocketMQ 版如何实现以下功能：

- 异步解耦
- 分布式事务的数据一致性
- 消息的顺序收发

最后，再以电商的秒杀场景和价格同步场景分别说明消息队列 RocketMQ 版所实现的削峰填谷和大规模机器的缓存同步。

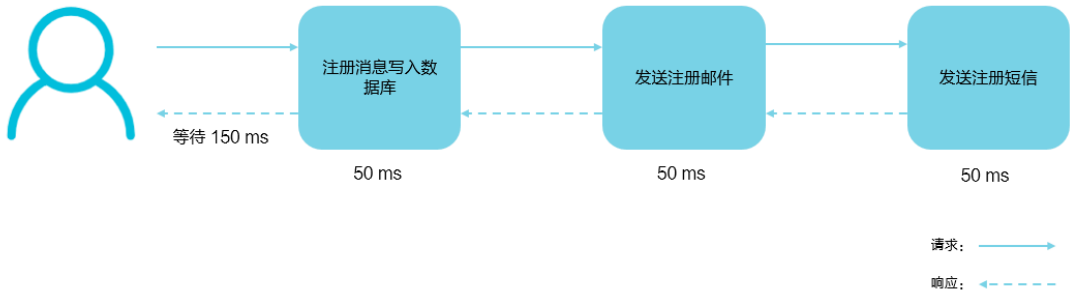
异步解耦

传统处理

最常见的一个场景是用户注册后，需要发送注册邮件和短信通知，以告知用户注册成功。传统的做法有以下两种：

- 串行方式

  串行方式下的注册流程如下图所示。



  数据流动如下所述：

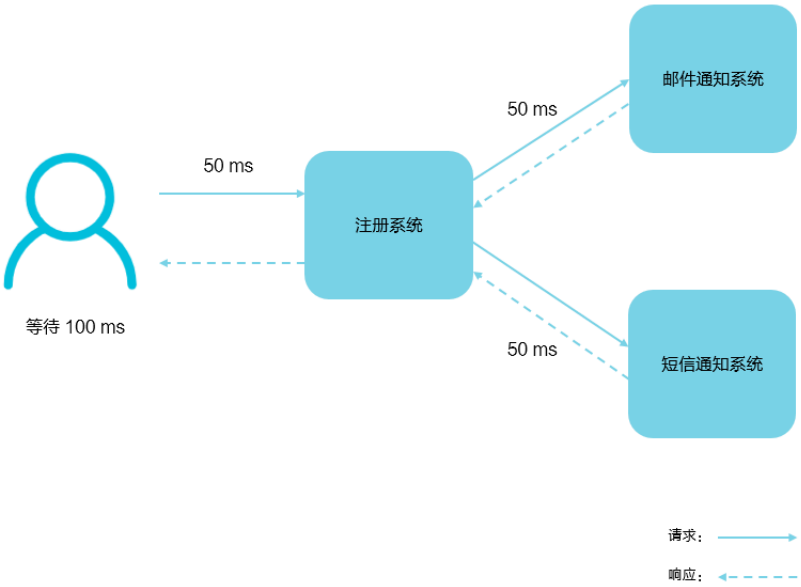
  1. 用户在注册页面填写账号和密码并提交注册信息，这些注册信息首先会被写入注册系统成功。
  2. 注册信息写入注册系统成功后，再发送请求至邮件通知系统。邮件通知系统收到请求后向用户发送邮件通知。
  3. 邮件通知系统接收注册系统请求后再向下游的短信通知系统发送请求。短信通知系统收到请求后向用户发送短信通知。

  以上三个任务全部完成后，才返回注册结果到客户端，用户才能使用账号登录。

  假设每个任务耗时分别为 50 ms，则用户需要在注册页面等待总共需要 150 ms 才能登录。

- 并行方式

  并行方式下的注册流程如下图所示。



  数据流动如下所述：

  1. 用户在注册页面填写账号和密码并提交注册信息，这些注册信息首先会被写入注册系统成功。
  2. 注册信息写入注册系统成功后，再同时发送请求至邮件和短信通知系统。邮件和短信通知系统收到请求后分别向用户发送邮件和短信通知。

  以上三个任务全部完成后，才返回注册结果到客户端，用户才能使用账号登录。

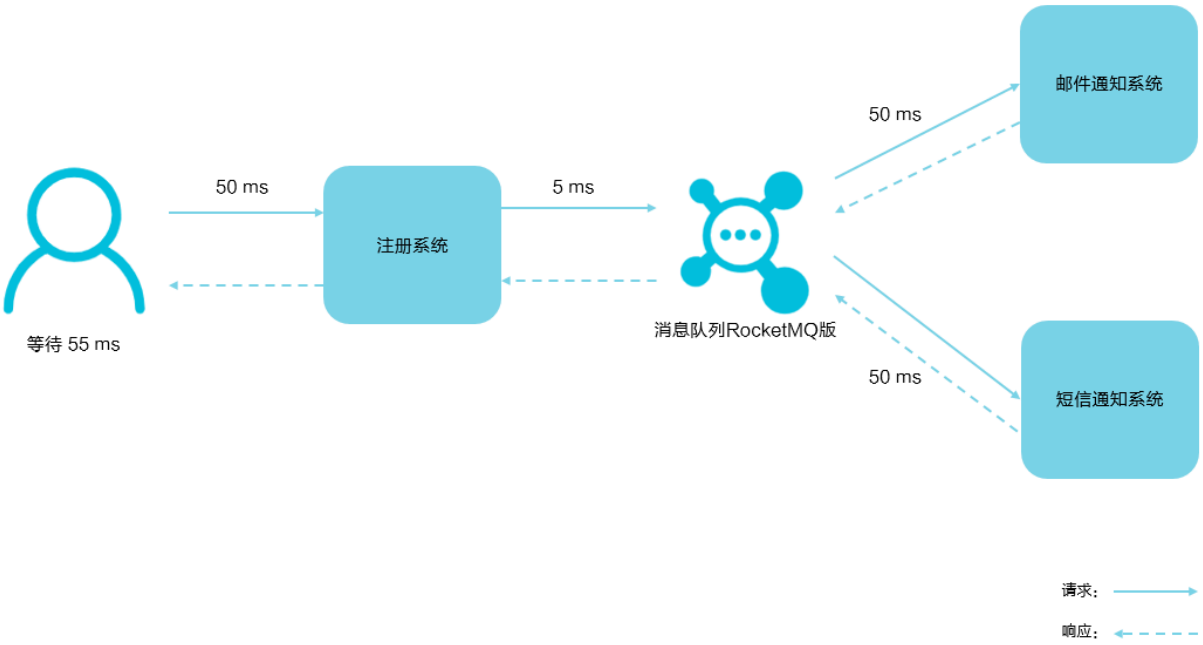
  假设每个任务耗时分别为 50 ms，其中，邮件和短信通知并行完成，则用户需要在注册页面等待总共需要 100 ms 才能登录。

以下就注册场景中使用了消息队列 RocketMQ 版的效果进行说明。

异步解耦

对于用户来说，注册功能实际只需要注册系统存储用户的账户信息后，该用户便可以登录，后续的注册短信和邮件不是即时需要关注的步骤。

对于注册系统而言，发送注册成功的短信和邮件通知并不一定要绑定在一起同步完成，所以实际当数据写入注册系统后，注册系统就可以把其他的操作放入对应的消息队列 RocketMQ 版中然后马上返回用户结果，由消息队列 RocketMQ 版异步地进行这些操作。



数据流动如下所述：

1. 用户在注册页面填写账号和密码并提交注册信息，这些注册信息首先会被写入注册系统成功。
2. 注册信息写入注册系统成功后，再发送消息至消息队列 RocketMQ 版。消息队列 RocketMQ 版会马上返回响应给注册系统，注册完成。用户可立即登录。
3. 下游的邮件和短信通知系统订阅消息队列 RocketMQ 版的此类注册请求消息，即可向用户发送邮件和短信通知，完成所有的注册流程。

用户只需在注册页面等待注册数据写入注册系统和消息队列 RocketMQ 版的时间，即等待 55 ms 即可登录。

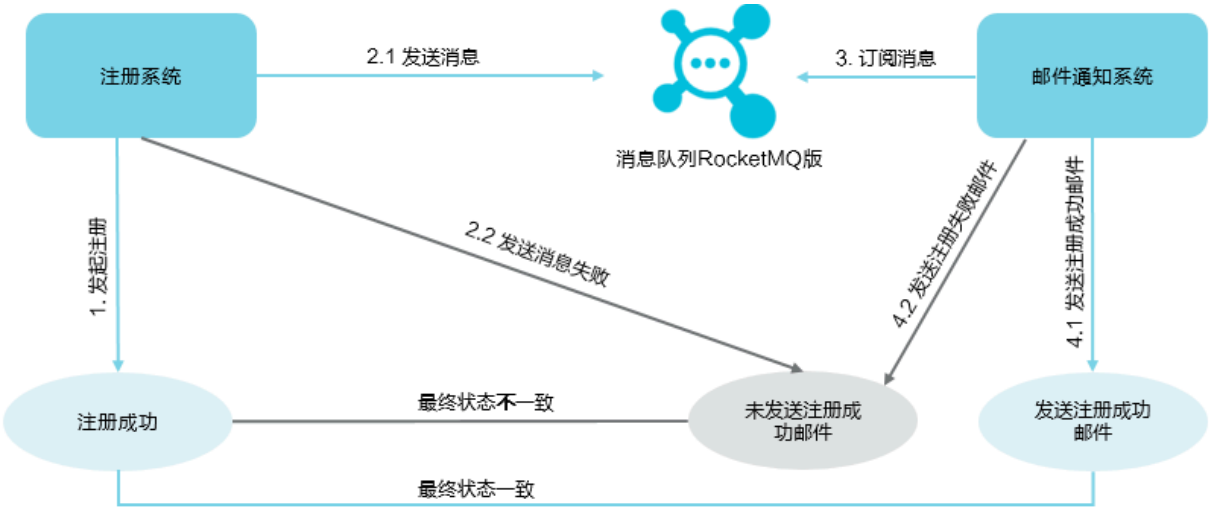
异步解耦是消息队列 RocketMQ 版的主要特点，主要目的是减少请求响应时间和解耦。主要的适用场景就是将比较耗时而且不需要即时（同步）返回结果的操作作为消息放入消息队列。同时，由于使用了消息队列 RocketMQ 版，只要保证消息格式不变，消息的发送方和接收方并不需要彼此联系，也不需要受对方的影响，即解耦和。

分布式事务的数据一致性

注册系统注册的流程中，用户入口在网页注册系统，通知系统在邮件系统，两个系统之间的数据需要保持最终一致。

普通消息处理

如上所述，注册系统和邮件通知系统之间通过消息队列进行异步处理。注册系统将注册信息写入注册系统之后，发送一条注册成功的消息到消息队列 RocketMQ 版，邮件通知系统订阅消息队列 RocketMQ 版的注册消息，做相应的业务处理，发送注册成功或者失败的邮件。



流程说明如下：

1. 注册系统发起注册。
2. 注册系统向消息队列 RocketMQ 版发送注册消息成功与否的消息。

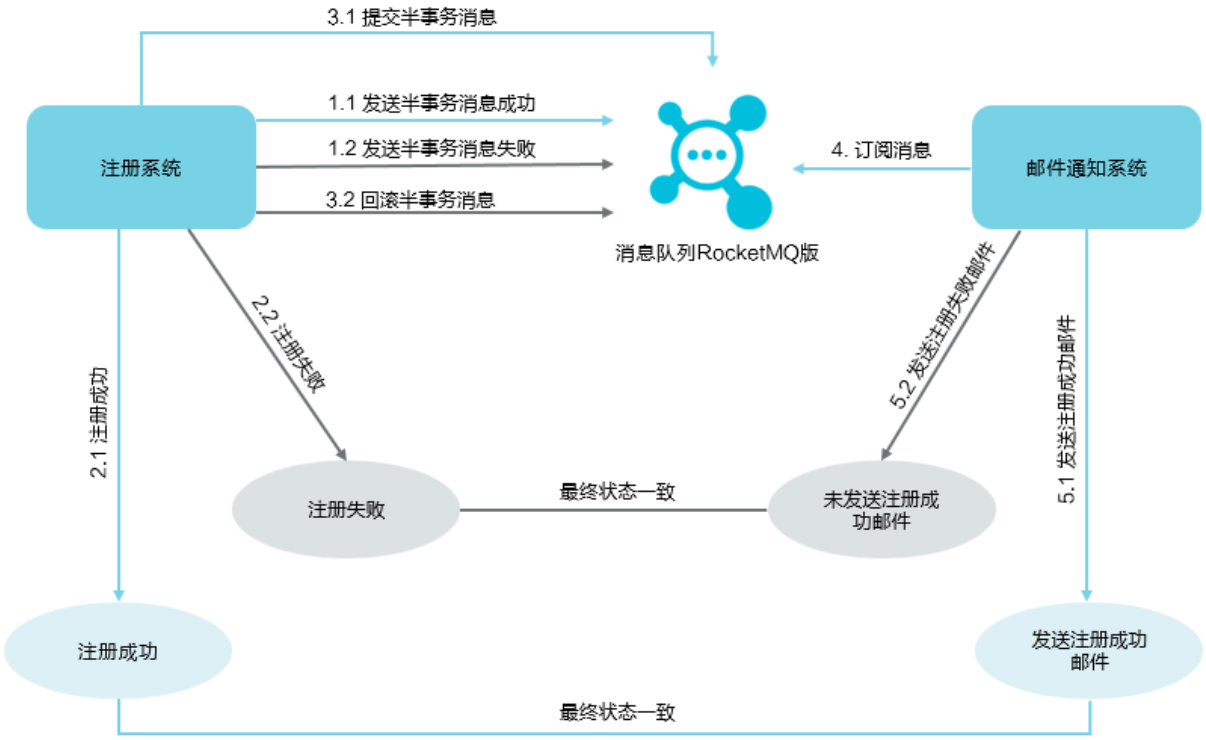
   2.1 消息发送成功，进入 3。

   2.2 消息发送失败，导致邮件通知系统未收到消息队列 RocketMQ 版发送的注册成功与否的消息，而无法发送邮件，最终邮件通知系统和注册系统之间的状态数据不一致。
3. 邮件通知系统收到消息队列 RocketMQ 版的注册成功消息。
4. 邮件通知系统发送注册成功邮件给用户。

在这样的情况下，虽然实现了系统间的解藕，上游系统不需要关心下游系统的业务处理结果；但是数据一致性不好处理，如何保证邮件通知系统状态与注册系统状态的最终一致。

事务消息处理

此时，需要有利用消息队列 RocketMQ 版所提供的事务消息来实现系统间的状态数据一致性。



流程说明如下：

1. 注册系统向消息队列 RocketMQ 版发送半事务消息。

   1.1 半事务消息发送成功，进入 2。

   1.2 半事务消息发送失败，注册系统不进行注册，流程结束。（最终注册系统与邮件通知系统数据一致）

2. 注册系统开始注册。

   2.1 注册成功，进入 3.1。

   2.2 注册失败，进行 3.2。

3. 注册系统向消息队列 RocketMQ 版发送半消息状态。

   3.1 提交半事务消息，产生注册成功消息，进入 4。

   3.2 回滚半事务消息，未产生注册成功消息，流程结束。（最终注册系统与邮件通知系统数据一致）

4. 邮件通知系统接收消息队列 RocketMQ 版的注册成功消息。
5. 邮件通知系统发送注册成功邮件。（最终注册系统与邮件通知系统数据一致）

分布式事务消息的更多详细内容请参见#unique_13。

消息的顺序收发

消息队列 RocketMQ 版顺序消息分为两种情况：

- 全局顺序：对于指定的一个 Topic，所有消息将按照严格的先入先出（FIFO）的顺序，进行顺序发布和顺序消费；
- 分区顺序：对于指定的一个 Topic，所有消息根据 Sharding Key 进行区块分区，同一个分区内的消息将按照严格的 FIFO 的顺序，进行顺发布和顺序消费，可以保证一个消息被一个进程消费。

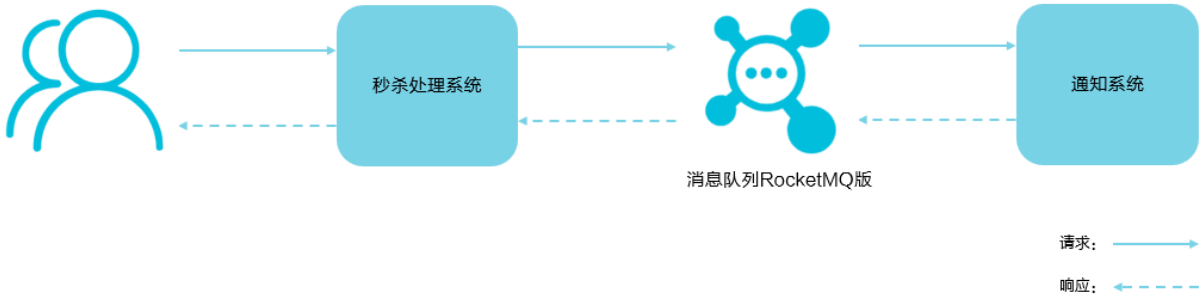
  在注册场景中，可使用用户 ID 作为 Sharding Key 来进行分区，同一个分区下的新建、更新或删除注册信息的消息必须按照 FIFO 的顺序发布和消费。

顺序消息的详细内容请参见#unique_15。

削峰填谷

流量削峰也是消息队列 RocketMQ 版的常用场景，一般在秒杀或团队抢购活动中使用广泛。

在秒杀或团队抢购活动中，由于用户请求量较大，导致流量暴增，秒杀的应用在处理如此大量的访问流量后，下游的通知系统无法承载海量的调用量，甚至会导致系统崩溃等问题而发生漏通知的情况。为解决这些问题，可在应用和下游通知系统之间加入消息队列 RocketMQ 版。



秒杀处理流程如下所述：

1. 用户发起海量秒杀请求到秒杀业务处理系统。
2. 秒杀处理系统按照秒杀处理逻辑将满足秒杀条件的请求发送至消息队列 RocketMQ 版。
3. 下游的通知系统订阅消息队列 RocketMQ 版的秒杀相关消息，再将秒杀成功的消息发送到相应用户。
4. 用户收到秒杀成功的通知。

大规模机器的缓存同步

双十一大促时，各个分会场会有玲琅满目的商品，每件商品的价格都会实时变化。使用缓存技术也无法满足对商品价格的访问需求，缓存服务器网卡满载。访问较多次商品价格查询影响会场页面的打开速度。

此时需要提供一种广播机制，一条消息本来只可以被集群的一台机器消费，如果使用消息队列 RocketMQ 版的广播消费模式，那么这条消息会被所有节点消费一次，相当于把价格信息同步到需要的每台机器上，取代缓存的作用。

广播消费的详细内容请参见#unique_18。

更多信息

您还可以访问应用场景获取更多有关消息队列 RocketMQ 版应用场景的信息。

4 产品架构

本文介绍消息队列 RocketMQ 版的系统部署架构，方便您更好地理解消息队列 RocketMQ 版的高可用性。

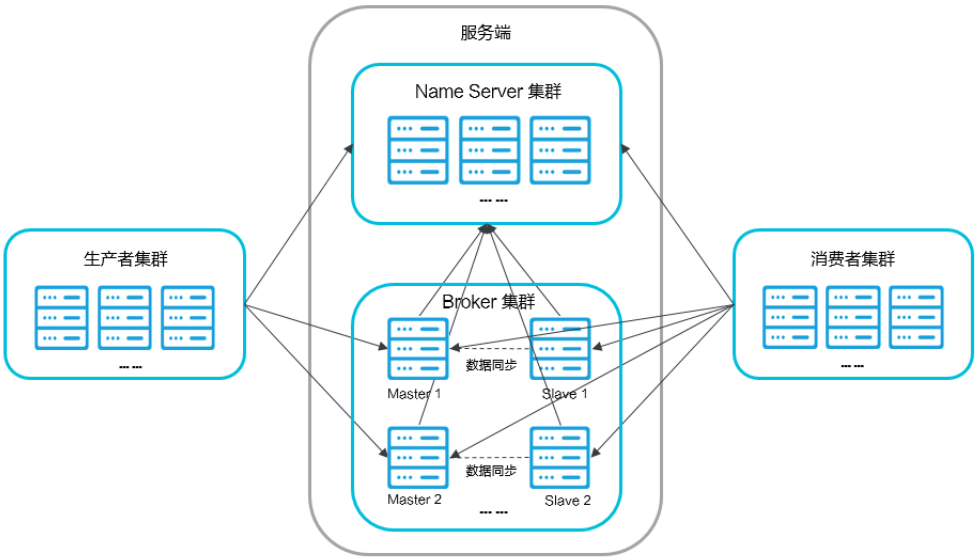
消息队列 RocketMQ 版在任何一个环境都是可扩展的，生产者必须是一个集群，消息服务器必须是一个集群，消费者也同样。集群级别的高可用，是消息队列 RocketMQ 版跟其他的消息服务器的主要区别，消息生产者发送一条消息到消息服务器，消息服务器会随机的选择一个消费者，只要这个消费者消费成功就认为是成功了。

注意：

文中所提及的消息队列 RocketMQ 版的服务端或者服务器包含 Name Server、Broker 等。服务端不等同于 Broker。

系统部署架构

系统部署架构如下图所示。



图中所涉及到的概念如下所述：

- Name Server：是一个几乎无状态节点，可集群部署，在消息队列 RocketMQ 版中提供命名服务，更新和发现 Broker 服务。
- Broker：消息中转角色，负责存储消息，转发消息。分为 Master Broker 和 Slave Broker，一个 Master Broker 可以对应多个 Slave Broker，但是一个 Slave Broker 只能对应一个 Master Broker。Broker 启动后需要完成一次将自己注册至 Name Server 的操作；随后每隔 30s 定期向 Name Server 上报 Topic 路由信息。

- 生产者：与 Name Server 集群中的其中一个节点（随机）建立长链接（Keep-alive），定期从 Name Server 读取 Topic 路由信息，并向提供 Topic 服务的 Master Broker 建立长链接，且定时向 Master Broker 发送心跳。
- 消费者：与 Name Server 集群中的其中一个节点（随机）建立长连接，定期从 Name Server 拉取 Topic 路由信息，并向提供 Topic 服务的 Master Broker、Slave Broker 建立长连接，且定时向 Master Broker、Slave Broker 发送心跳。Consumer 既可以从 Master Broker 订阅消息，也可以从 Slave Broker 订阅消息，订阅规则由 Broker 配置决定。

更多信息

消息队列 RocketMQ 版中的概念详情，请参见名词解释。

5 名词解释

本文主要对消息队列 RocketMQ 版涉及的专有名词及术语进行定义和解析，方便您更好地理解相关概念并使用消息队列 RocketMQ 版。

Topic	消息主题，一级消息类型，通过 Topic 对消息进行分类。详情请参见 #unique_25。
消息（Message）	消息队列中信息传递的载体。
Message ID	消息的全局唯一标识，由消息队列 RocketMQ 版系统自动生成，唯一标识某条消息。
Message Key	消息的业务标识，由消息生产者（Producer）设置，唯一标识某个业务逻辑。
Tag	消息标签，二级消息类型，用来进一步区分某个 Topic 下的消息分类。详情请参见 #unique_25。
Producer	消息生产者，也称为消息发布者，负责生产并发送消息。
Producer 实例	Producer 的一个对象实例，不同的 Producer 实例可以运行在不同进程内或者不同机器上。Producer 实例线程安全，可在同一进程内多线程之间共享。
Consumer	消息消费者，也称为消息订阅者，负责接收并消费消息。可分为两类： • Push Consumer：消息由消息队列 RocketMQ 版推送至 Consumer。 • Pull Consumer：该类 Consumer 主动从消息队列 RocketMQ 版拉取消息。目前仅 TCP Java SDK 支持该类 Consumer。 **注意：** 如需使用 Pull Consumer，请确保您的消息队列 RocketMQ 版实例为企业铂金版。 详情请参见#unique_26和#unique_27。
分区	即 Topic Partition，物理上的概念。每个 Topic 包含一个或多个分区。
消费位点	每个 Topic 会有多个分区，每个分区会统计当前消息的总条数，这个称为最大位点 MaxOffset；分区的起始位置对应的位置叫做起始位点 MinOffset。

	消息队列 RocketMQ 版的 Pull Consumer 会按顺序依次消费分区内的每条消息，记录已经消费了的消息条数，称为消费位点 ConsumerOffset。剩余的未消费的条数（也称为消息堆积量）= 最大位点 MaxOffset - 消费位点 ConsumerOffset。
Consumer 实例	Consumer 的一个对象实例，不同的 Consumer 实例可以运行在不同进程内或者不同机器上。一个 Consumer 实例内配置线程池消费消息。
Group	一类 Producer 或 Consumer，这类 Producer 或 Consumer 通常生产或消费同一类消息，且消息发布或订阅的逻辑一致。
Group ID	Group 的标识。
队列	每个 Topic 下会由一到多个队列来存储消息。每个 Topic 对应队列数与消息类型以及实例所处地域（Region）相关，具体的队列数可提交工单咨询。  **注意：** 标准版实例不支持变更队列数，铂金版实例支持变更队列数。
Exactly-Once 投递语义	Exactly-Once 投递语义是指发送到消息系统的消息只能被 Consumer 处理且仅处理一次，即使 Producer 重试消息发送导致某消息重复投递，该消息在 Consumer 也只被消费一次。详情请参见 #unique_28。
集群消费	一个 Group ID 所标识的所有 Consumer 平均分摊消费消息。例如某个 Topic 有 9 条消息，一个 Group ID 有 3 个 Consumer 实例，那么在集群消费模式下每个实例平均分摊，只消费其中的 3 条消息。详情请参见#unique_18。
广播消费	一个 Group ID 所标识的所有 Consumer 都会各自消费某条消息一次。例如某个 Topic 有 9 条消息，一个 Group ID 有 3 个 Consumer 实例，那么在广播消费模式下每个实例都会各自消费 9 条消息。详情请参见#unique_18。
定时消息	Producer 将消息发送到消息队列 RocketMQ 版服务端，但并不期望这条消息立马投递，而是推迟到在当前时间点之后的某一个时间投递到 Consumer 进行消费，该消息即定时消息。详情请参见#unique_14。
延时消息	Producer 将消息发送到消息队列 RocketMQ 版服务端，但并不期望这条消息立马投递，而是延迟一定时间后才投递到 Consumer 进行消费，该消息即延时消息。详情请参见#unique_14。

事务消息 消息队列 RocketMQ 版提供类似 X/Open XA 的分布事务功能，通过消息队列 RocketMQ 版的事务消息能达到分布式事务的最终一致。详情请参见#unique_13。

顺序消息 消息队列 RocketMQ 版提供的一种按照顺序进行发布和消费的消息类型，分为全局顺序消息和分区顺序消息。详情请参见#unique_15。

全局顺序消息 对于指定的一个 Topic，所有消息按照严格的先入先出（FIFO）的顺序进行发布和消费。详情请参见#unique_15。

分区顺序消息 对于指定的一个 Topic，所有消息根据 Sharding Key 进行区块分区。同一个分区内的消息按照严格的 FIFO 顺序进行发布和消费。Sharding Key 是顺序消息中用来区分不同分区的关键字段，和普通消息的 Message Key 是完全不同的概念。详情请参见#unique_15。

消息堆积 Producer 已经将消息发送到消息队列 RocketMQ 版的服务端，但由于 Consumer 消费能力有限，未能在短时间内将所有消息正确消费掉，此时在消息队列 RocketMQ 版的服务端保存着未被消费的消息，该状态即消息堆积。

消息过滤 Consumer 可以根据消息标签（Tag）对消息进行过滤，确保 Consumer 最终只接收被过滤后的消息类型。消息过滤在消息队列 RocketMQ 版的服务端完成。详情请参见#unique_29。

消息轨迹 在一条消息从 Producer 发出到 Consumer 消费处理过程中，由各个相关节点的时间、地点等数据汇聚而成的完整链路信息。通过消息轨迹，您能清晰定位消息从 Producer 发出，经由消息队列 RocketMQ 版服务端，投递给 Consumer 的完整链路，方便定位排查问题。详情请参见#unique_17。

重置消费位点 以时间轴为坐标，在消息持久化存储的时间范围内（默认 3 天），重新设置 Consumer 对已订阅的 Topic 的消费进度，设置完成后 Consumer 将接收设定时间点之后由 Producer 发送到消息队列 RocketMQ 版服务端的消息。详情请参见#unique_19。

死信队列 死信队列用于处理无法被正常消费的消息。当一条消息初次消费失败，消息队列 RocketMQ 版会自动进行#unique_30；达到最大重试次数后，若消费依然失败，则表明 Consumer 在正常情况下无法正确地消费该消息。此时，消息队列 RocketMQ 版不会立刻将消息丢弃，而是将这条消息发送到该 Consumer 对应的特殊队列中。

消息队列 RocketMQ 版将这种正常情况下无法被消费的消息称为死信消息（Dead-Letter Message），将存储死信消息的特殊队列称为死信队列（Dead-Letter Queue）。

详情请参见#unique_20。

消息路由 消息路由常用于不同地域之间的消息同步，保证地域之间的数据一致性。消息队列 RocketMQ 版的全球消息路由功能依托阿里云优质基础设施实现的高速通道专线，可以高效地实现不同地域之间的消息同步复制。详情请参见#unique_21。

6 场景化案例

6.1 权益分发

在电商和新零售领域，权益分发适用于对特定的用户进行指定运营活动的场景。技术层面上，此种场景的 DB 和缓存的数据强一致性较难保证。针对该问题，阿里云消息队列 RocketMQ 版推出了权益分发解决方案。本文将以电商场景为例说明权益分发解决方案的背景信息、方案架构、以及方案优势。

背景信息

在电商平台运营决定对特定用户进行营销活动时，会针对不同的用户群设置不同的营销策略，可能会涉及但不局限于以下内容：

- 用户规则：针对什么样的用户下发权限，如新用户、会员等，结合业务需求设置具体的判断条件。
- 权限类型：红包、积分或是优惠券等不同类型的权益。
- 领取成本：需要付出成本才能获取相应权益，如消耗会员积分才能领取折扣券。
- 时间控制：发放权益的时间点，如早上 10:00 针对新用户下发 100 张优惠券。
- 库存：权益的个数是否充足、与配置的用户个数是否匹配等。

这些运营数据写入 DB 后，转换成技术策略，写入缓存，再转换成用户发放链路数据（变成一条一条的规则）。只有符合这些规则的用户才能领取对应的权益。

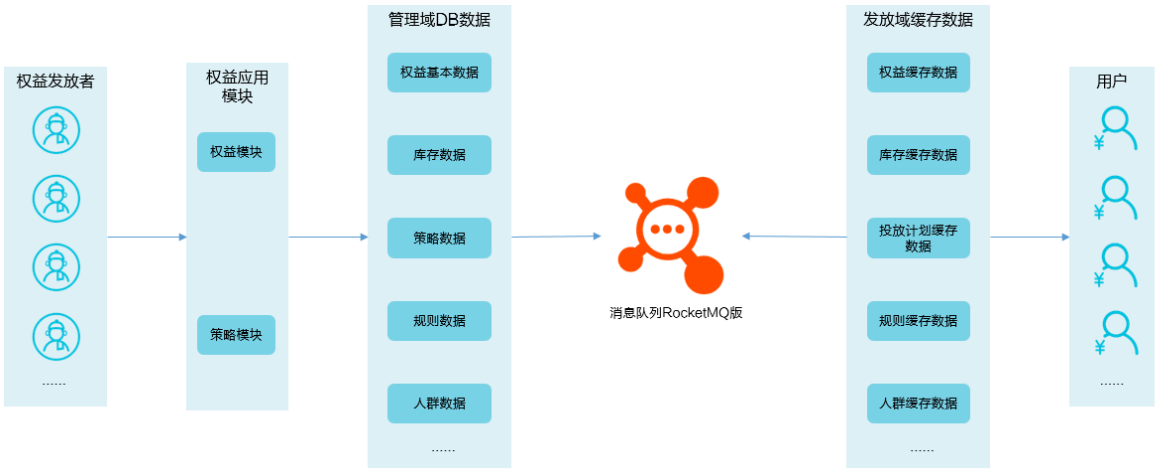
痛点

让用户通过访问缓存数据来领取对应的权益，这样既保证了用户高效的访问，也减轻了 DB 的压力。但是，这就会引发新的问题：

- 当规则特别多时，DB 写入缓存的数据量也会特别大，下游的缓存压力较大（此类情况较少）。
- DB 写入数据至缓存时只写一次，可能会因为网络抖动等原因导致数据写入失败，进而造成数据更新不及时的结果。例如，库存原定需设置 100,000，但运营策略变成 10 个，如果数据没有及时同步，那么就无法通知到下游，造成资损。

方案架构

因为消息队列 RocketMQ 版具有重试功能且能保证消息不丢失，所以推出此方案来确保 DB 和缓存数据的强一致性，即在 DB 和缓存间使用消息队列 RocketMQ 版，架构如下图所示。



方案优势

- 消息队列 RocketMQ 版拥有巨大的数据吞吐量。
- 消息队列 RocketMQ 版支持海量消息堆积，为下游应用减轻流量洪峰的冲击。
- 消息队列 RocketMQ 版可以帮助简化 DB 和缓存间的实现，大大减少代码开发量。如果没有消息队列 RocketMQ 版，实现方法则十分复杂。
- 消息队列 RocketMQ 版的实时消息收发以及重试功能可确保消息不丢，从而确保消息强一致性。

更多信息

消息队列 RocketMQ 版的功能与特性详情，请参见以下文档：

- 功能与特性概述
- #unique_30

7 使用限制

消息队列 RocketMQ 版对某些具体指标进行了约束和规范，您在使用消息队列 RocketMQ 版时注意不要超过相应的限制值，以免程序出现异常。

具体的限制项和限制值请参见下表。

限制项	限制值	说明
单个地域（Region）内的消息队列 RocketMQ 版实例数	• 标准版：8 个 • 企业铂金版：数量不限	无
Topic 名称长度	64 个字符	Topic 名称长度不得超过该限制，否则会导致无法发送或者订阅。
消息大小	• 普通和顺序消息：4 MB • 事务和定时/延时消息：64 KB **说明：** 其中，所有消息类型的消息属性大小均不能超过 16 KB。	消息大小不得超过其类型所对应的限制，否则消息会发送失败。
消息保存时间	3 天	消息最多保留 3 天，超过时间将自动滚动删除。
消费位点重置	3 天	支持重置消费 3 天之内任何时间点的消息。
单消息队列 RocketMQ 版实例的消息收发 TPS	• 标准版：5000 条/秒；如需更高规格，请提交工单 • 企业铂金版：参考所购买的规格	无
定时/延时消息的延时时长	40 天	msg.setStartDeliverTime 参数（单位：毫秒）可设置 40 天内的任何时刻，超过 40 天消息发送将失败。 **说明：** 定时消息的精度有 1s ~ 2s 的延迟误差。

限制项	限制值	说明
批量发送消息	不支持	消息队列 RocketMQ 版不支持批量发送消息。