SOFAStack

消息队列 SOFAMQ

产品简介

产品版本：AntStack Plus 1.11.0

文档版本：20220930

蚂蚁集团
ANT GROUP

法律声明

蚂蚁集团版权所有©2022，并保留一切权利。

商标声明

免责声明

通用约定

格式	说明	样例
危险	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。	危险 重置操作将丢失用户配置数据。
警告	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。
注意	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。
说明	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	命令或代码。	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

1.概述 ---- 05
2.产品优势 ---- 07
3.产品架构 ---- 08
4.功能特性 ---- 10
5.应用场景 ---- 12
6.基础术语 ---- 16
7.使用限制 ---- 19

1.概述

SOFAStack 消息队列（SOFAStack MQ，简称 SOFAMQ）是基于 Apache RocketMQ 构建的分布式消息中间件，并与金融分布式架构 SOFAStack 深度集成，为分布式应用系统提供异步解耦和削峰填谷的能力，支持事务消息、顺序消息、定时消息等多种消息类型，并具备高可靠、高吞吐、低延时等金融级特性。

应用场景

- 异步解耦消息队列的生产消费模型可以解耦上下游业务系统，并支持下游多个消费者对同一消息进行消费和处理。以金融场景为例，支付中心作为支付宝主站最核心的系统，每笔支付数据的产生会引起几百个下游业务系统的关注，包括账户中心、用户中心、权益中心、流计算分析等，整体业务系统庞大而且复杂，在应用强耦合的情况下，任一应用故障都将可能对业务带来影响。通过消息队列进行异步通信和应用解耦，可以很好的提升业务的连续性。
- 削峰填谷应用分布式改造后，不同应用能承载的性能情况往往不一致，在诸如双 11、店庆、秒杀等大型活动时，将会带来较高的流量脉冲，可能导致系统超负荷甚至崩溃，影响用户体验。消息队列可提供强大的抗积压能力，实现削峰填谷，生产方生产消息后，消费方可以按照系统自身的承受能力进行消息的消费。
- 顺序收发指的是消息消费者按照消息发送的顺序进行消费，保证 FIFO。金融场景里需要保证顺序的应用场景非常多，例如证券交易过程中的时间优先原则（交易系统中的订单创建、支付、退款等流程）。
- 分布式事务一致性应用解耦往往带来多个应用之间的事务一致性的问题。例如支付转账成功后，需要生成账单，更新用户积分等，此时通过消息队列的分布式事务处理功能，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。

更多信息请参见 应用场景。

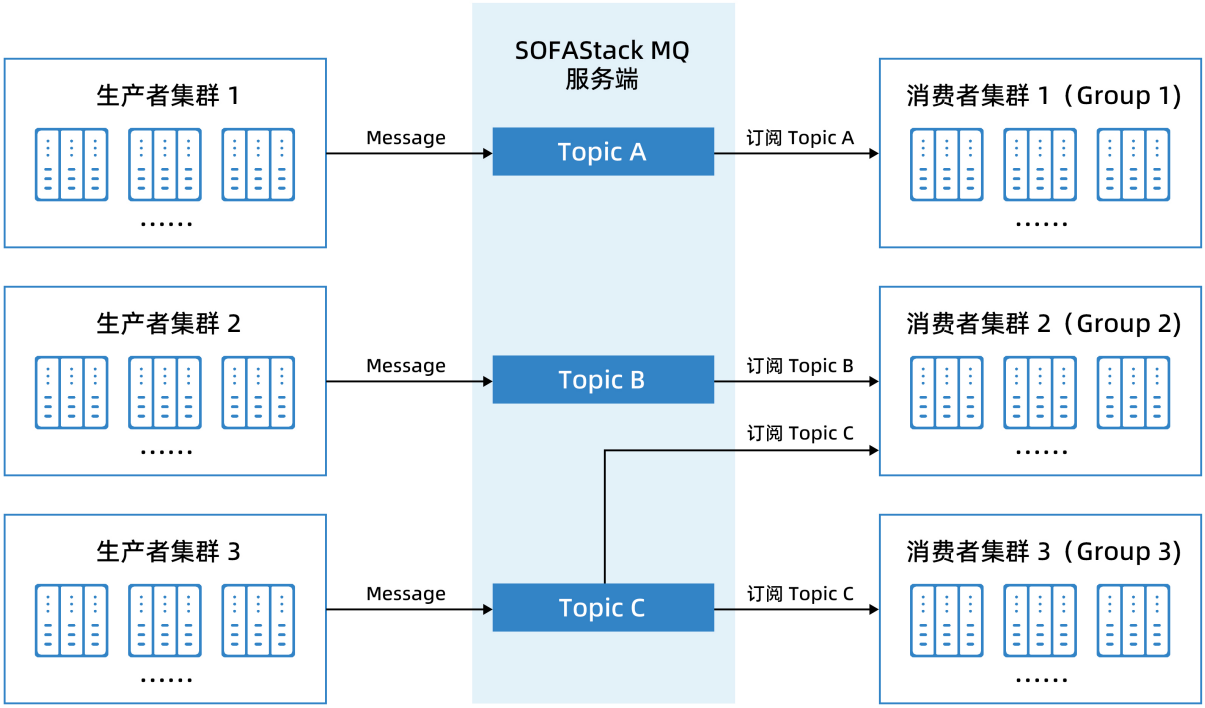
核心概念

- 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。

更多概念解释请参见 基础术语。

消息收发模型

消息队列支持发布/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。



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，如上图中的 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。

2.产品优势

- **功能齐全**
  - 支持多种消息类型：普通消息、定时消息、分区顺序消息、事务消息
  - 支持多种消费模式：Pub/Sub 模式、Tag 过滤
  - 支持 TCP Java SDK
- **运维体系便捷**
  - 支持多维度消息查询
  - 支持全链路消息轨迹
- **性能优越**
  - 海量消息堆积能力
  - 毫秒级端到端延迟
  - 千万级高并发处理能力
- **服务可靠**
  - 99.9% 服务高可用
  - 99.99999% 数据高可靠
  - 支持消息重投机制

3.产品架构

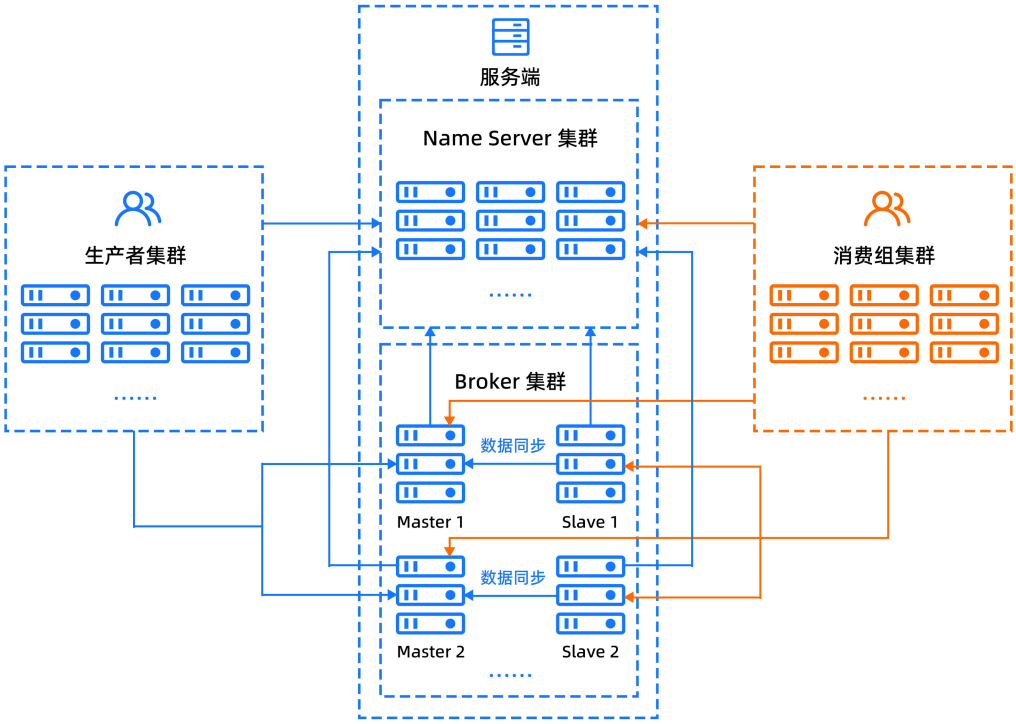
本文介绍 SOFAStack 消息队列的系统部署架构，方便您更好地理解消息队列的高可用性。

消息队列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。

> **说明**
>
> 文中所提及的消息队列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：

- **NameServer**：是一个几乎无状态节点，可集群部署，在消息队列中提供命名服务，该组件内置 zookeeper 用来负责 Broker 的选主和集群管理。
- **Broker**：消息中转角色，负责存储和转发消息。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 Producer**：与 Name Server 集群中的其中一个节点（随机）建立长连接（Keep-alive），定期从 Name Server 读取 Topic 路由信息，并与提供 Topic 服务的 Master Broker 建立长连接，且定时向 Master Broker 发送心跳。

- **消费者 Consumer**：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。

SOFAMQ 还包含如下两个未在图中展示的组件，主要功能如下：

- **Console**：SOFAMQ 的图形化管理控制台。
- **Router**：LDC 场景下，根据配置的 Router 规则，进行逻辑单元之间消息转发，使 SOFAMQ 支持 LDC 能力的组件。

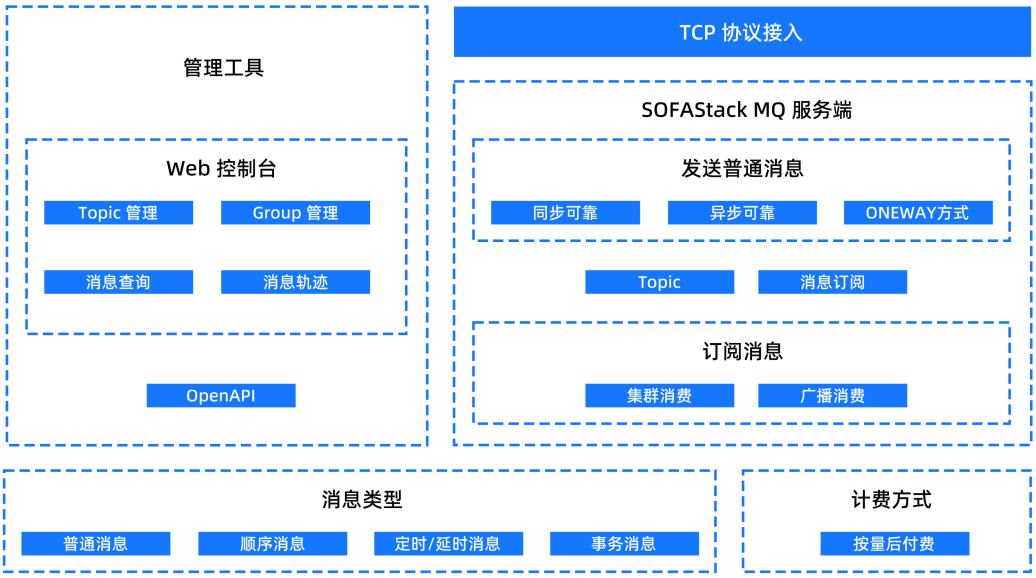
更多信息

消息队列中的概念详情，请参见 基础术语。

4.功能特性

SOFAStack 消息队列在阿里云多个地域（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。

消息队列提供 TCP 协议语言接入方式，方便应用快速接入消息队列消息云服务。在网络联通的情况下，您可以将应用部署在阿里云 ECS、企业自建云，与消息队列建立连接进行消息收发。



TCP 协议接入

提供更为专业、可靠、稳定的 TCP 协议的 SDK 接入服务。支持 Java 语言。

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。
- OpenAPI：提供开放的 API 便于将消息队列管理工具集成到自己的控制台。

消息类型

- 普通消息：消息队列中无特性的消息，区别于有特性的定时/延时消息、顺序消息和事务消息。
- 事务消息：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。
- 定时和延时消息：允许消息生产者对指定消息进行定时（延时）投递。
- 顺序消息：允许消息消费者按照消息发送的顺序对消息进行消费。

有关消息类型的详细信息，参见 消息类型。

特性功能

- 消息查询：消息队列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。

- 查询消息轨迹：通过消息轨迹，能清晰定位消息从生产者发出，经由消息队列服务端，投递给消息消费者的完整链路，方便定位排查问题。
- 集群消费和广播消费：当使用集群消费模式时，消息队列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。
- 重置消费位点：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。

5.应用场景

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

在互联网金融场景里，其业务涉及广泛，如支付交易、收费计息、商户结算、业务营销、会员积分、风险核查等；同时，也会涉及许多业务峰值时刻，如双 11、秒杀、周年庆等。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。

消息队列作为分布式系统中的重要组件，可以很好的应对这些场景。

下文以支付转账为场景说明消息队列如何实现以下功能：

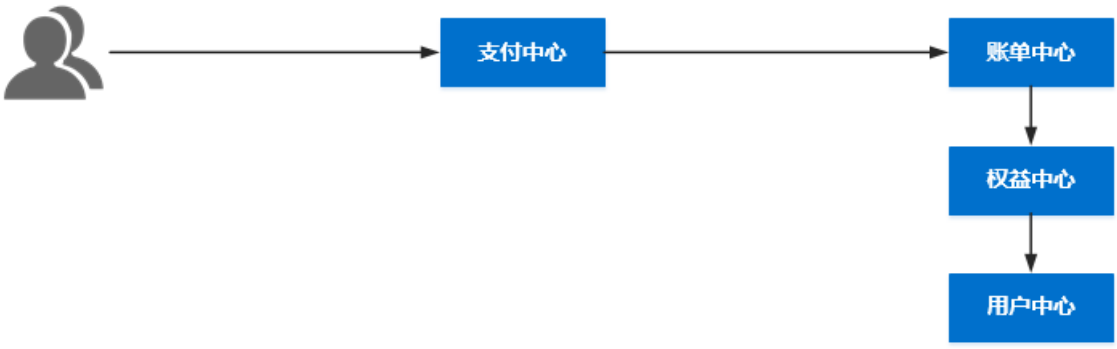
- 异步解耦
- 分布式事务的数据一致性
- 削峰填谷

异步解耦

传统处理方式

最常见的一个场景是支付转账成功后，需要生成交易双方的账单，并更新用户权益，发送用户通知。传统的做法有以下两种：

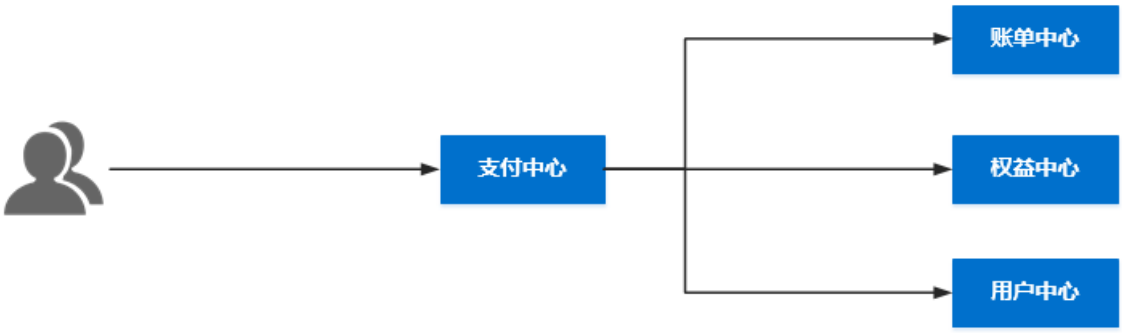
- 串行方式



流程说明如下：

1. 用户在支付中心，填写金额等相关信息，完成转账操作。
2. 转账成功后，再发送请求至账单中心，生成交易双方账单。
3. 账单生成成功后，再发送请求至权益中心，更新用户积分。
4. 积分更新成功后，再发送请求至用户中心，发送用户通知。

- 并行方式

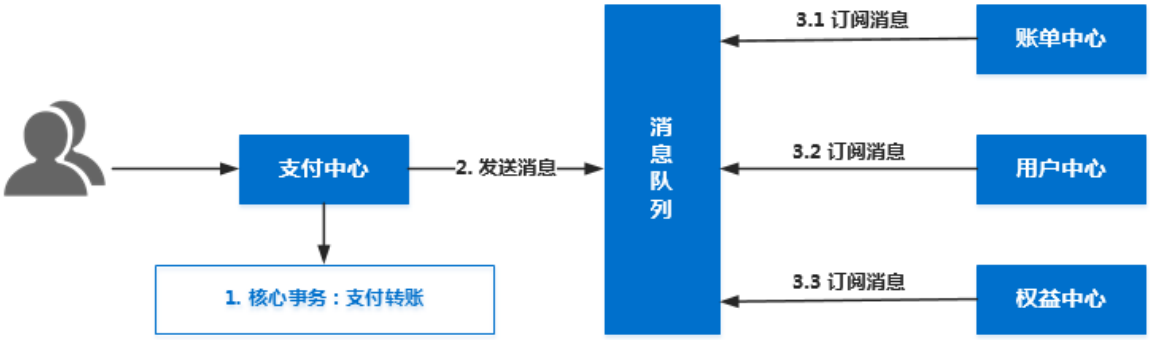


流程说明如下：

1. 用户在支付中心，填写金额等相关信息，完成转账操作。
2. 转账成功后，同时发送请求至账单中心、用户中心、权益中心完成相应操作。

异步解耦方式

对于用户来说，转账操作成功后，后续的账单、权益、积分等不是即时需要关注的步骤，由系统保证即可。



流程说明如下：

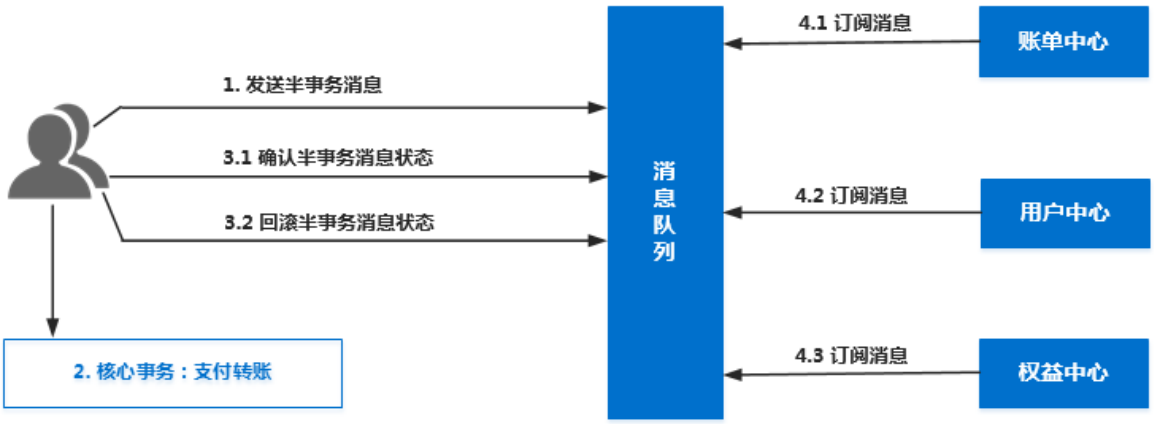
1. 用户在转账页面，填写金额等相关信息，完成转账操作。
2. 转账成功后，发送一条支付消息至消息队列。消息队列会马上返回响应给支付中心，转账完成。
3. 下游的账单中心、权益中心、用户中心等系统订阅消息队列的支付消息，完成后续的业务流程。

异步解耦是消息队列的主要特点和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，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

支付的流程中，用户入口在支付中心完成，账单、权益、通知系统在其他系统完成，多个系统之间的数据需要保持最终一致。

在这样的情况下，虽然实现了系统间的解耦，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，如何保证下游系统状态与支付系统状态的最终一致。此时，需要利用消息队列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. 支付中心向消息队列发送半事务消息。
   - 半事务消息发送成功，进入 2。
   - 半事务消息发送失败，支付中心不进行转账，流程结束。（最终支付中心与下游系统数据一致）
2. 支付中心开始转账流程。
   - 转账成功，进入 3.1。
   - 转账失败，进行 3.2。
3. 支付中心向消息队列发送半消息状态。
   - 3.1 提交半事务消息，产生支付成功消息，进入 4。
   - 3.2 回滚半事务消息，未产生支付成功消息，流程结束。（最终支付中心与下游系统数据一致）
4. 下游系统接收消息队列的支付成功的消息。
5. 下游系统处理相关业务逻辑。（最终支付中心与下游系统数据一致）

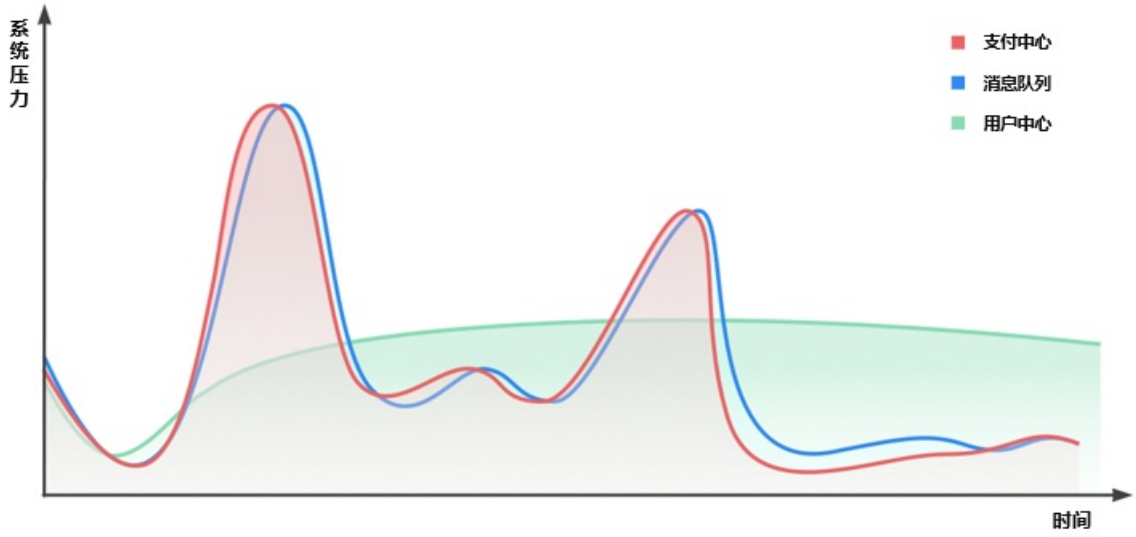
分布式事务消息的更多详细内容请参见 事务消息。

削峰填谷

流量削峰也是消息队列的常用场景，一般在秒杀或团队抢购活动中使用广泛。

以支付场景举例，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，支付中心在处理如此大量的访问流量后，下游的应用用户中心可能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。

引入消息队列后，用户中心作为消费方可以根据自身应用的能力进行消息的消费，不受大流量的影响。


系统压力
支付中心
消息队列
用户中心
时间

6.基础术语

本文主要对 SOFAStack 消息队列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。

中文	英文	释义
消息主题	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 Topic 与 Tag。
消息	Message	消息队列中信息传递的载体。
Schema	Schema	消息内容的结构描述，通过 Schema 对消息内容的属性以及类型进行描述。
Message ID	Message ID	消息的全局唯一标识，由消息队列系统自动生成，唯一标识某条消息。
Message Key	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。
消息标签	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 Topic 与 Tag。
消息生产者	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 instance	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。
消息消费者	Consumer	消息消费者，也称为消息订阅者，负责接收并消费消息。
Consumer 实例	Consumer instance	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	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 ID	Group 的标识。

中文	英文	释义
队列	Queue	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。
集群消费	Clustering consumption	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 集群消费和广播消费。
广播消费	Broadcasting consumption	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 集群消费和广播消费。
定时消息	Scheduled message	Producer 将消息发送到消息队列服务端，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 定时和延时消息。
延时消息	Delayed message	Producer 将消息发送到消息队列服务端，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 定时和延时消息。
事务消息	Transactional message	消息队列提供类似 X/Open XA 的分布事务功能，通过消息队列的事务消息能达到分布式事务的最终一致。详情请参见 事务消息。
顺序消息	Ordered message	消息队列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，当前仅支持分区顺序消息。详情请参见 顺序消息。
分区顺序消息	Partitionally ordered message	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 顺序消息。

中文	英文	释义
消息堆积	Message accumulation	Producer 已经将消息发送到消息队列的服务端，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	Message filtering	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的服务端完成。详情请参见 消息过滤。
消息轨迹	Message trace	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 查询消息轨迹。
重置消费位点	Reset consumption offset	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 发送到消息队列服务端的消息。详情请参见 重置消费位点。

7.使用限制

SOFAStack 消息队列对某些具体指标进行了约束和规范，在使用过程中请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
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：4 MB	消息大小不得超过其类型所对应的限制，否则消息会被丢弃。
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。
定时/延时消息的延时时长	40 天	• 默认最大限制 3 天，超过 3 天消息将发送失败。您可以修改 `timerMaxDelaySec` 参数（单位：秒）增加延时时间限制至 40 天，即 3456000。 • 您可以在 `msg.setStartDeliverTime` 参数（单位：毫秒）设置消息延时时间。