QingCloud Technologies

浅入浅出消息中间件

2022.08.03

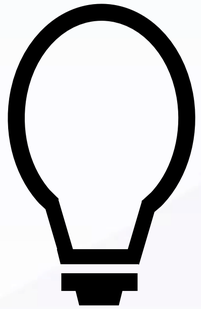
目录

什么是消息中间件

消息服务应用场景

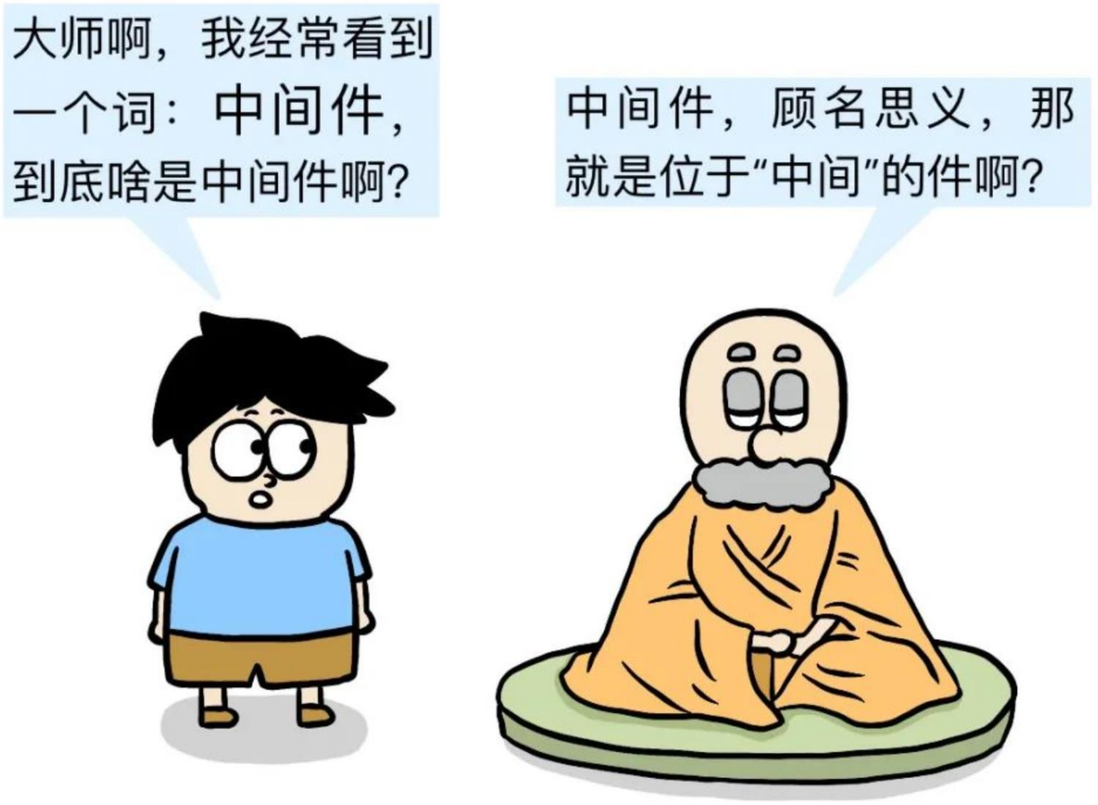
各消息产品分析

消息产品发展趋势



什么是消息中间件

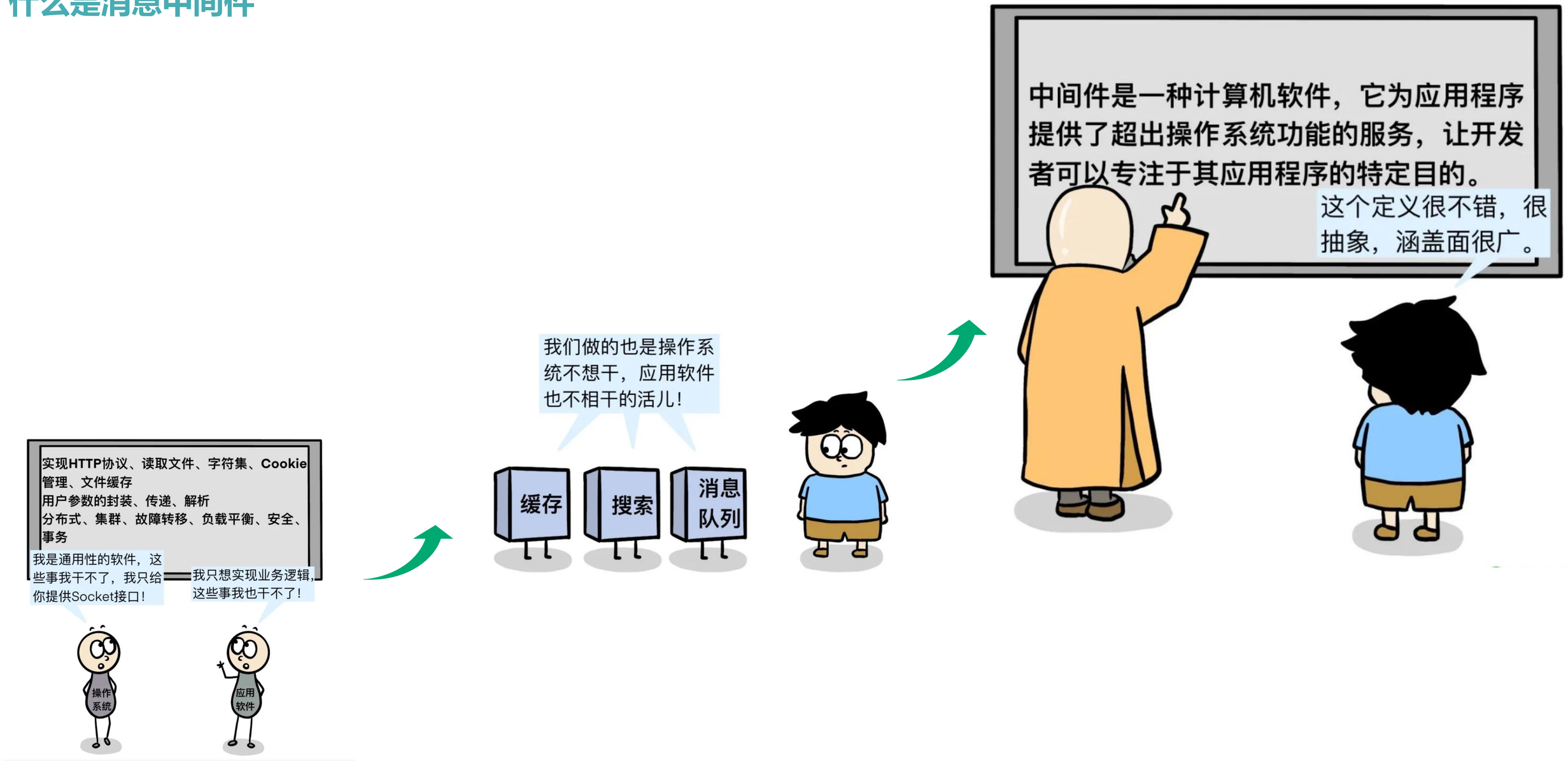
什么是消息中间件



首先

什么是中间件

什么是消息中间件


实现HTTP协议、读取文件、字符集、Cookie管理、文件缓存
用户参数的封装、传递、解析
分布式、集群、故障转移、负载平衡、安全、事务
我是通用性的软件，这些事我干不了，我只给你提供Socket接口！
我只想实现业务逻辑，这些事我也干不了！
操作系统
应用软件
缓存
搜索
消息队列
我们做的也是操作系统不想干，应用软件也不相干的活儿！
中间件是一种计算机软件，它为应用程序提供了超出操作系统功能的服务，让开发者可以专注于其应用程序的特定目的。
这个定义很不错，很抽象，涵盖面很广。

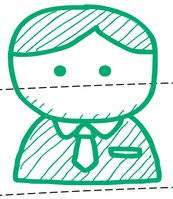
消息

MQ 全称 Message Queue（消息队列），是在消息的传输过程中保存消息的容器，多用于分布式系统之间进行通信。

Producer Queue Consumer

消息中间件演变史

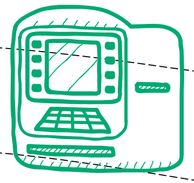
TIB
诞生于1985年
服务于金融机构和新闻机构



一个在美国的印度人Vivek Ranadive，设想了一种通用软件总线，创办了一家公司Teknekron

初见曙光

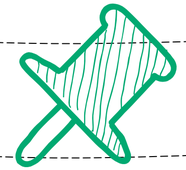
MQ/Wesphere MQ
诞生于1993年
商业消息队列平台玩家



Teknekron的业务发展引起了当时最牛气的IT公司IBM的注意，于是他们也开始研发了自己消息队列软件

各自为战

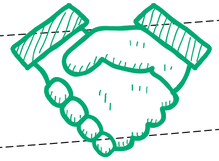
JMS
诞生于2001年
本质是一套Java API



为了能够让消息在各个消息队列平台间互融互通， JMS (Java Message Service) 应运而生

劫制天下

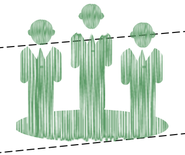
AMQP/RabbitMQ
规范发布于2006年，同年RabbitMQ面世



由 Cisco 、 Redhat 、 iMatix 等联合制定了 AMQP 的公开标准，由此 AMQP 登上了历史的舞台

一统江湖

Kafka/RocketMQ
2010年发布，次年RocketMQ面世



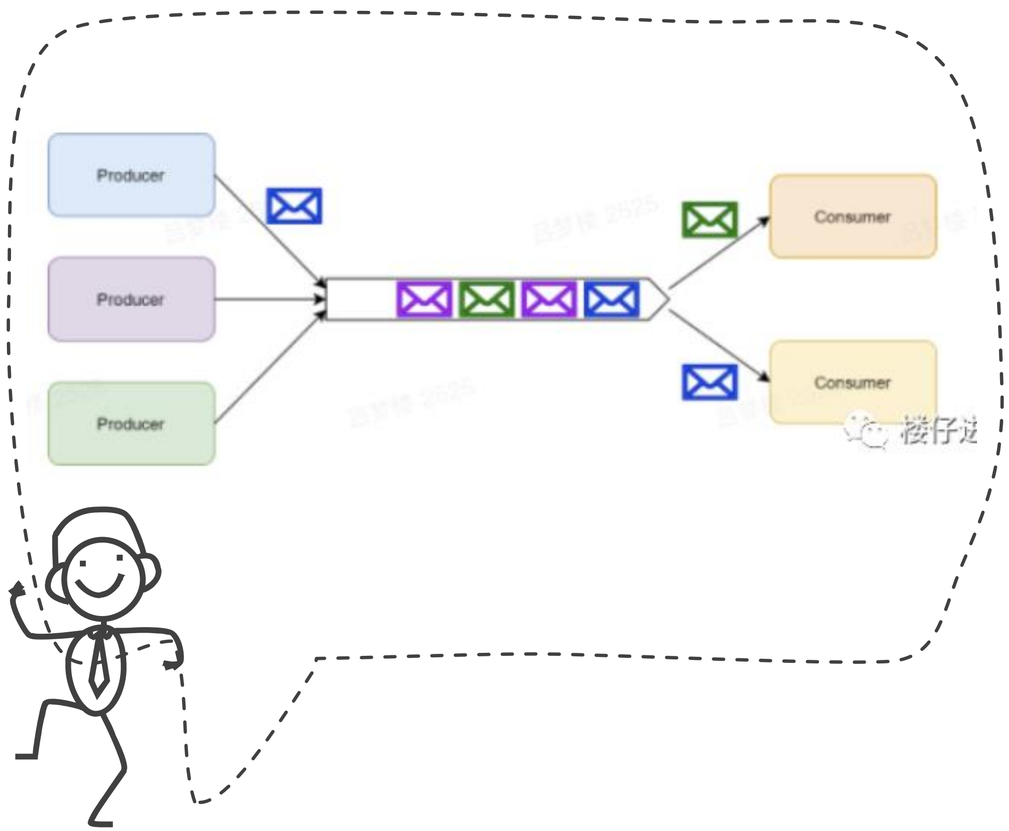
LinkedIn在实现消息队列的时候觉得AMQP规范并不适合自己，所以Kafka并不支持AMQP协议。RocketMQ在实现上借鉴了Kakfa的思想，所以也不支持AMQP协议

合久必分

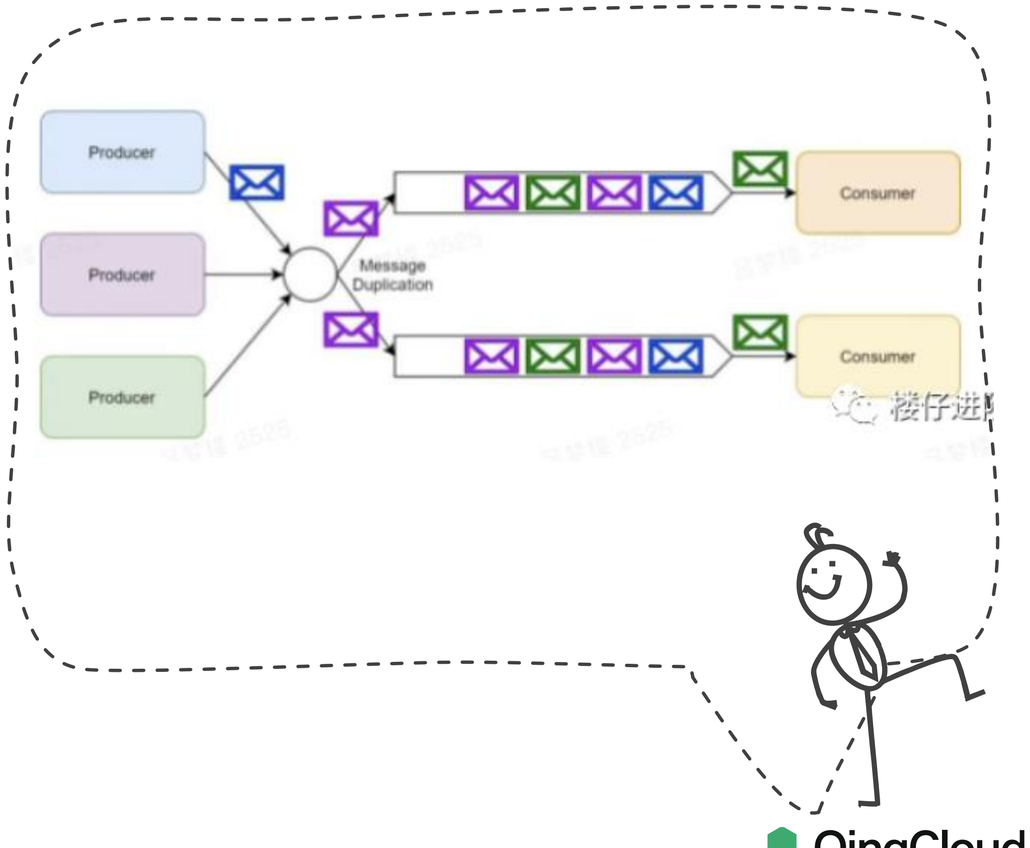
什么是消息中间件

消息队列

点对点模式



发布/订阅模式



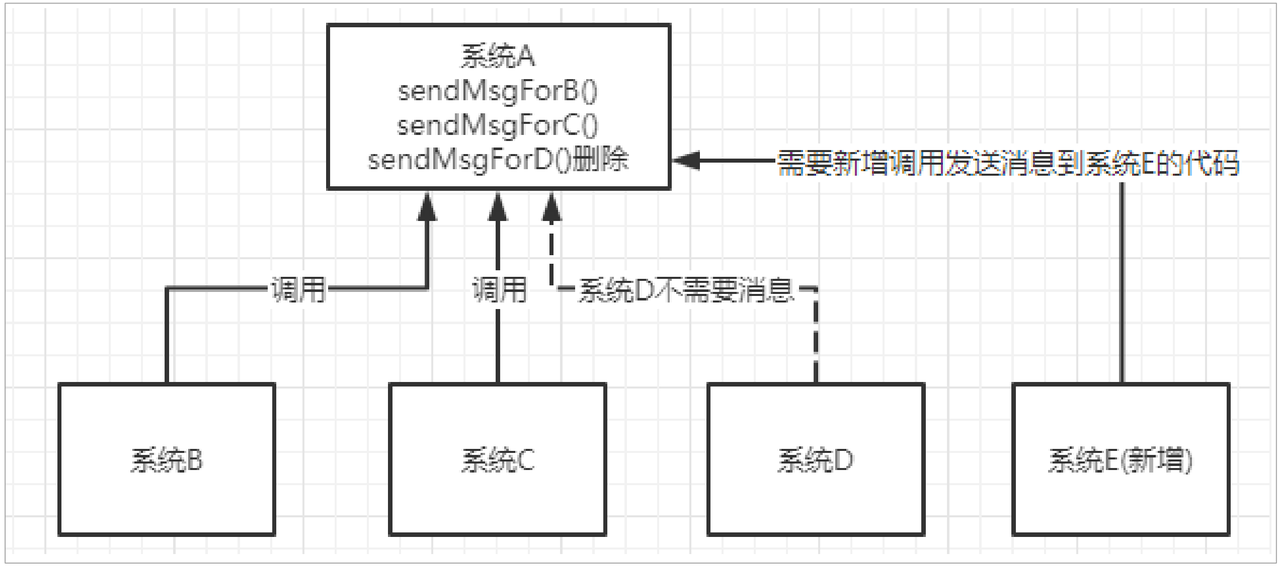
消息服务应用场景

消息服务应用场景

Message Queue 主要是用户程序与程序之间通信，实现异步+解耦,设计理念为了满足整体流量的削峰填谷。

核心应用场景：

解耦



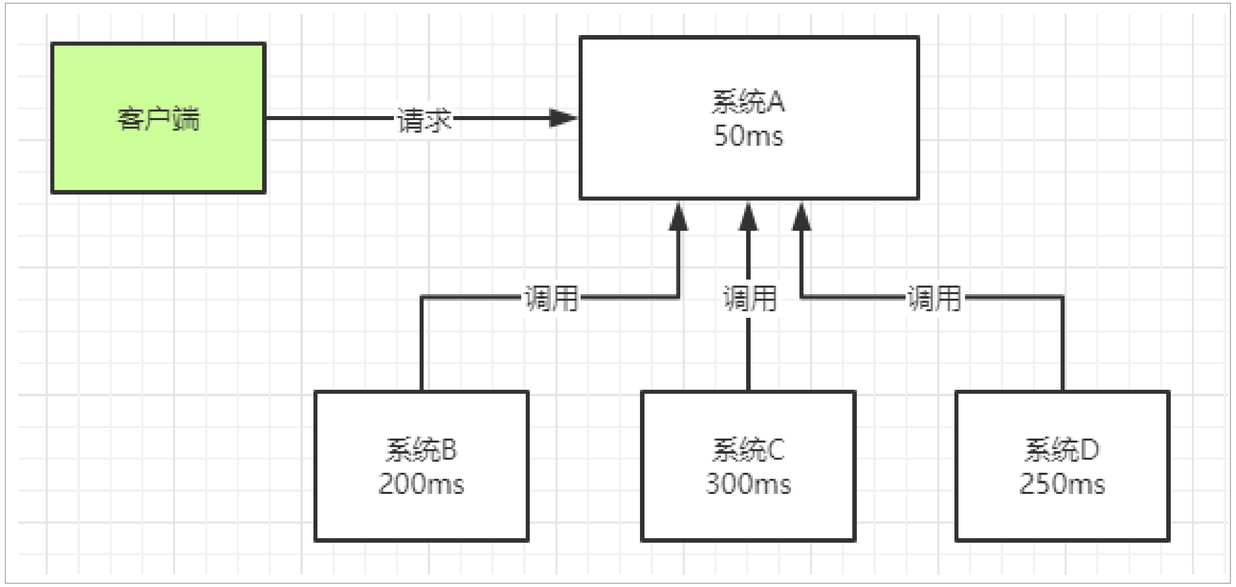
假设有系统B、C、D都需要系统A的数据，于是系统A调用三个方法发送数据到B、C、D。这时，系统D不需要了，那就需要在系统A把相关的代码删掉。假设这时有个新的系统E需要数据，这时系统A又要增加调用系统E的代码。为了降低这种强耦合，就可以使用MQ，系统A只需要把数据发送到MQ，其他系统如果需要数据，则从MQ中获取即可。

消息服务应用场景

Message Queue.主要是用户程序与程序之间通信，实现异步+解耦,设计理念为了满足整体流量的削峰填谷。

核心应用场景：

异步



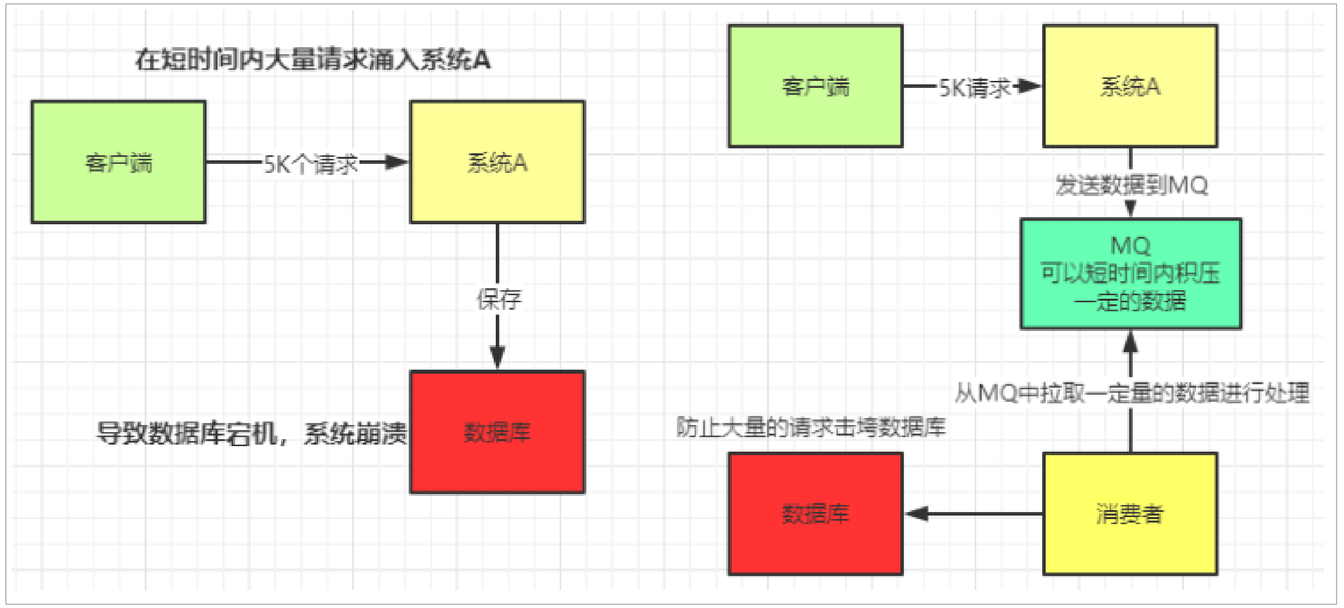
客户端请求发送进来，系统A会调用系统B、C、D三个系统，同步请求的话，响应时间就是系统A、B、C、D的总和，也就是800ms。如果使用MQ，系统A发送数据到MQ，然后就可以返回响应给客户端，不需要再等待系统B、C、D的响应，可以大大地提高性能。对于一些非必要的业务，比如发送短信，发送邮件等等，就可以采用MQ。

消息服务应用场景

Message Queue.主要是用户程序与程序之间通信，实现异步+解耦,设计理念为了满足整体流量的削峰填谷。

核心应用场景：

削峰



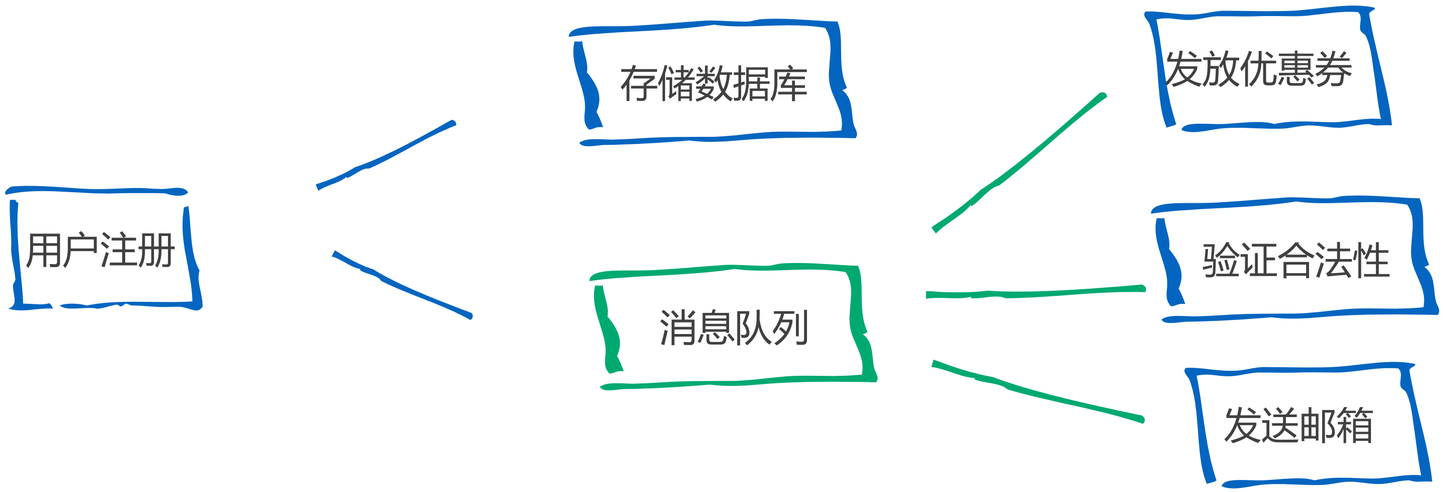
假设系统A在某一段时间请求数暴增，有5000个请求发送过来，系统A这时就会发送5000条SQL进入MySQL进行执行，MySQL对于如此庞大的请求当然处理不过来，MySQL就会崩溃，导致系统瘫痪。如果使用MQ，系统A不再是直接发送SQL到数据库，而是把数据发送到MQ，MQ短时间积压数据是可以接受的，然后由消费者每次拉取2000条进行处理，防止在请求峰值时期大量的请求直接发送到MySQL导致系统崩溃。

消息服务应用场景

队列的主要作用举例：解耦，异步和并行，用户注册的案例来说明下 MQ 的作用

未使用MQ

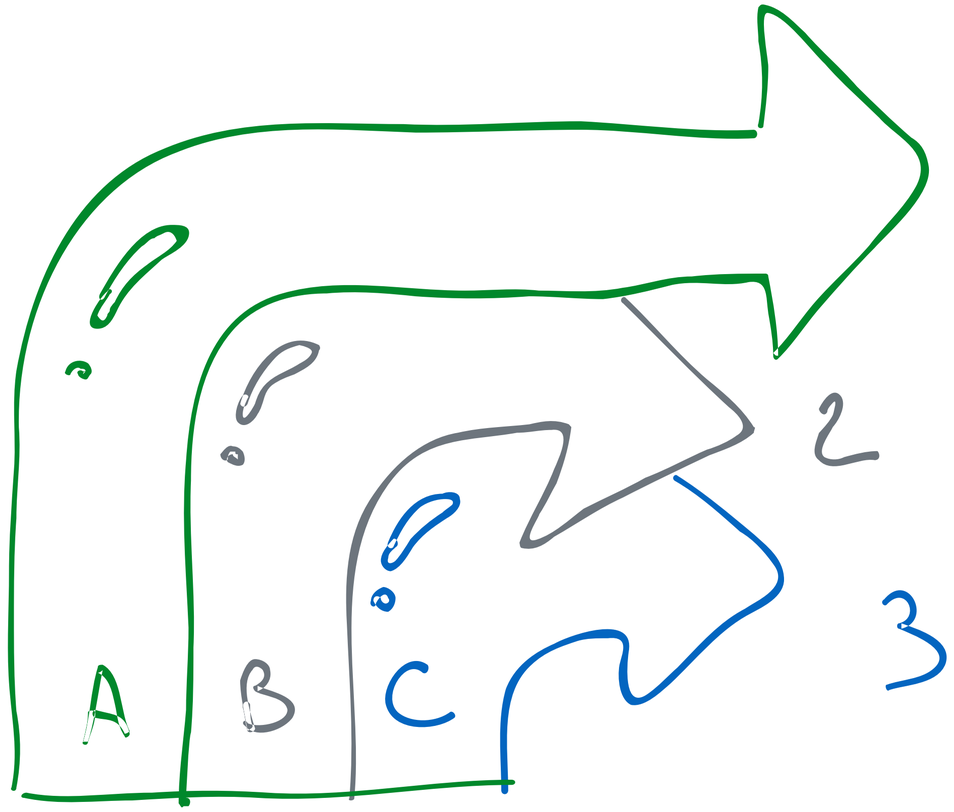
使用MQ



各消息产品分析

几款消息产品介绍

目前比较主流的 MQ 产品主要有三个，大有三足鼎立的格局



Kafka

Kafka 是一个分布式的，支持多分区、多副本，基于 Zookeeper 的分布式消息流平台

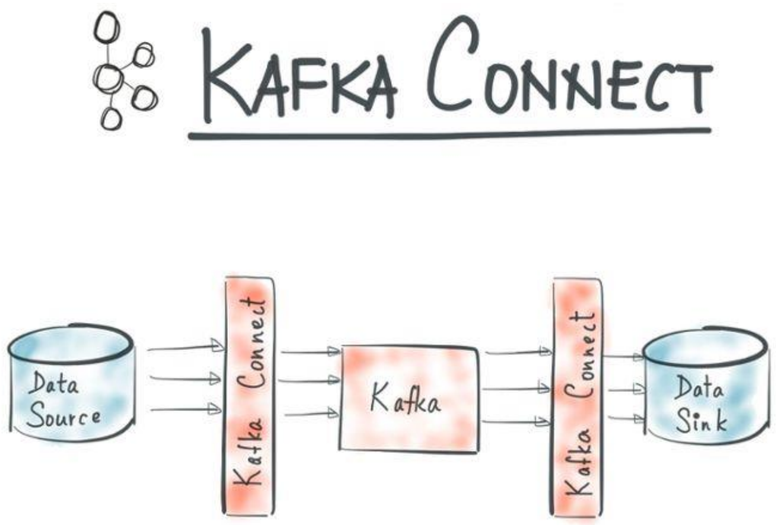
RabbitMQ

RabbitMQ 是一款使用Erlang语言开发的，实现AMQP(高级消息队列协议)的开源消息中间件

RocketMQ

RocketMQ是一个纯 Java、分布式、队列模型的开源消息中间件

几款消息产品介绍



Kafka 是一个分布式的，支持多分区、多副本，基于 Zookeeper 的分布式消息流平台

Apache Kafka 它最初由 LinkedIn 公司基于独特的设计实现为一个分布式的提交日志系统(a distributed commit log)，之后成为 Apache 项目的一部分。号称大数据的杀手锏，谈到大数据领域内的消息传输，则绕不开 Kafka，这款为大数据而生的消息中间件，以其百万级 TPS 的吞吐量名声大噪，迅速成为大数据领域的宠儿，在数据采集、传输、存储的过程中发挥着举足轻重的作用。

RabbitMQ 是一款使用 Erlang 语言开发的，实现 AMQP(高级消息队列协议)的开源消息中间件

RabbitMQ 于 2007 年发布，是使用 Erlang 语言开发的开源消息队列系统，基于AMQP协议来实现。AMQP 的主要特征是面向消息、队列、路由（包括点对点和发布/订阅）、可靠性、安全。AMQP 协议更多用在企业系统内，对数据一致性、稳定性和可靠性要求很高的场景，对性能和吞吐量的要求还在其次。

RocketMQ 是一个纯 Java、分布式、队列模型的开源消息中间件

是阿里开源的消息中间件，它是纯 Java 开发，具有高吞吐量、高可用性、适合大规模分布式系统应用的特点。RocketMQ 思路起源于 Kafka，但并不是 Kafka 的一个 Copy，它对消息的可靠传输及事务性做了优化，目前在阿里集团被广泛应用于交易、充值、流计算、消息推送、日志流式处理、binglog 分发等场景。

如何选择消息产品

根据上面不同的衡量标准，可以看出在选型的时候，各个产品有自己的优势，需根据客户的需求来进行选择，下面概括下三款产品的推荐使用场景。

Kafka 大型公司建议可以选用，如果有日志采集功能，肯定是首选kafka。

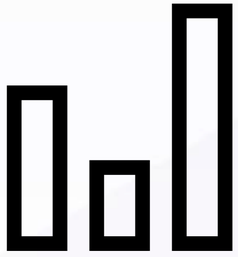
- 高吞吐、低延迟、高可用
- 处理活跃的流式数据
- 容错性非常高

RabbitMQ 如果你的数据量没有那么大，小公司优先选择功能比较完备的RabbitMQ。

- 支持丰富语言
- 社区活跃
- 管理界面丰富

RocketMQ RoketMQ在稳定性上可能更值得信赖，这些业务场景在阿里双11已经经历了多次考验，如果你的业务有上述并发场景，建议可以选择RocketMQ。

- 高吞吐、低延迟、高可用
- 顺序消费，消费能力强



消息产品发展趋势

消息产品发展趋势

回溯消息队列以前的发展：**被云、开源、商业所驱动**

消息系统的整体发展趋势可以分为两个方面：**技术架构和业务场景丰富度**

云原生

是消息队列架构的基本发展趋势
Pulsar等

连接

更多的数据源

Serverless

多语言函数化编程、免运维、弹性扩缩容、按量计费

Thank you!

几款消息产品介绍

特性	KAFKA	RocketMQ	RabbitMQ
特点	高吞吐、低延迟；kakfa 最大的特点就是收发消息非常快	高性能, 支持数据海量堆积, 支持主动推送	由于Erlang语言的并发能力，性能很好, 支持多种协议，重量级系统
消息方式	Consumer pull	Consumer pull/ broker push	broker push
数据可靠性	很好	很好	好
拉取模式	Pull/Push	Pull/Push	Pull/Push
发送性能	十万级	1万	1万
消息保证	理论上消息不会丢失	不保证(消费失败16次后丢弃)	保证
持久化能力	磁盘文件	消息默认保留三天，支持海量堆积	内存、文件，支持数据堆积,但数据堆积反过来影响生产速率
可用性	非常高（分布式、主从）	非常高（分布式、主从）	高（主从）
消息延时	毫秒级	毫秒级	微毫秒
性能稳定性	队列/分区多时性能明显下降 消息堆积时性能稳定	队列较多/消息堆积时性能稳定	消息堆积时性能不稳定
批量操作	支持	不支持	不支持
评价	优点： 在高吞吐、低延迟、高可用、集群热扩散、集群容错上有非常好的表现 Producer端提供缓存、压缩功能，可节省性能，提高效率 3、顺序消费、生态完整、容错率高等优势 缺点： 1、消费集群数目收到分区数目限制 2、单机topic多时，性能会降低 3、不支持事务	优点： 1、在高吞吐、低延迟、高可用表现很好，消息堆积时性能也很好 2、支持多种消费、支持broker消息过滤，支持事务，支持顺序消费，消费能力强。 3、集群规模50台左右，单日处理消息可上百亿 缺点： 1、生态不够完善，消息堆积、吞吐率不如kafka 2、不支持主从自动切换 3、客户端只支持java	优点： 1、支持多种客户端语言，支持amqp协议 2、异步消息传递：支持多种消息协议 3、社区活跃，管理界面丰富 缺点： 1、高吞吐、高可用略低于前两者 2、erlang语言有门槛，集群不支持动态扩展 3、不支持事务，消息吞吐有限，消息堆积时性能下降