Computer Science and Application 计算机科学与应用, 2023, 13(12), 2180-2190
Published Online December 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.1312218

Hans汉斯

消息队列技术综述

谢星星，张翠平*，田小燕，成洁宇，王焕迪，张树顺

北京信息科技大学计算机学院，北京

收稿日期：2023年11月10日；录用日期：2023年12月8日；发布日期：2023年12月18日

摘　要

随着互联网的发展，业务系统在超高并发场景中，需要处理大量的数据和请求，采用消息队列的技术可以实现流量削峰、解耦、异步处理，解决可能的服务器问题。消息队列用于计算机系统的异步通信，通过消息队列可以构建高效、可靠和弹性的分布式系统，满足现代应用需求。本文总结对比了现有主流消息队列设计模式的原理与优劣，研究分析了消息队列的基本应用方法和创新技术应用方向，并据此对该技术的主要发展挑战与可能的发展方向做出分析，以此为消息队列技术的相关研究提供相对可靠的技术参考。

关键词

消息队列，异步通信，消息传递，分布式系统

Review of Message Queue Techniques

Xingxing Xie, Cuiping Zhang*, Xiaoyan Tian, Jieyu Cheng, Huandi Wang, Shushun Zhang

School of Computer Science, Beijing University of Information Technology, Beijing

Received: Nov. 10th, 2023; accepted: Dec. 8th, 2023; published: Dec. 18th, 2023

Abstract

With the development of the Internet, the business system needs to deal with a large number of data and requests in the ultra-high concurrency scenario. The technology of message queue can realize traffic peak cutting, decoupling, asynchronous processing, and solve possible server problems. Message queue is used for the asynchronous communication of computer systems. Through the message queue, an efficient, reliable and flexible distributed system can be built to meet the needs of modern applications. This paper summarizes the comparison of the principle and advantages of the existing mainstream information queue design mode, analyzes the basic application method and innovative technology application direction of information queue technology, and

*通讯作者。

文章引用：谢星星，张翠平，田小燕，成洁宇，王焕迪，张树顺. 消息队列技术综述[J]. 计算机科学与应用，2023, 13(12): 2180-2190. DOI: 10.12677/csa.2023.1312218

analyzes the main development challenges of the technology, so as to provide relatively reliable technical reference.

Keywords

Message Queue, Asynchronous Communication, Message Transmission, Distributed System

1. 引言

消息队列的研究经历了几十年的演进，随着技术的不断发展和应用需求的不断演变，尽管相关领域的专家和学者对消息队列的研究已经取得了很大的进展，但是对消息队列的研究和创新仍然是一个持续进行的活跃领域。消息队列的主要使用目的是实现异步通信和解耦，它提供了一种可靠、高效的通信机制。因此，对于消息队列的研究具有一定的现实意义，本文基于对消息队列技术在进程间通信的应用出发，研究了消息队列技术在分布式系统的通信问题，通过对消息队列技术的原理与应用的总结，分析了现代互联网对于消息队列技术的需求，以给相关技术开发者提供技术参考，同时归纳了消息技术的不足以及发展方向，以对该技术的完善提供可能的改进方向。

2. 消息队列技术研究背景与现状

目前较受欢迎的消息队列系统有 Apache Kafka、RabbitMQ、ActiveMQ、RocketMQ 等等，其中 Apache Kafka 在国外得到了广泛的研究和使用，研究人员关注 Kafka 的性能优化、可靠性保障、实时数据处理等方面；同时还探索了 Kafka 与其他系统(如流处理框架)的整合。

而 RocketMQ 是阿里巴巴研发的一款纯 Java 开发的分布式、高性能、高可靠、高实时的消息中间件。2016 年，阿里巴巴向 Apache 软件基金会捐赠 RocketMQ，并在 2017 年正式成为 Apache 顶级项目，这是国内首个互联网中间件在 Apache 上的顶级项目。RocketMQ 广泛应用于阿里巴巴内部的生产系统,满足线上海量消息堆积的需求。经历多年双十一的洗礼，在可用性、可靠性和稳定性方面都有着非常稳定的表现，证明了其是一款非常优秀的消息中间件。

3. 消息队列技术基本理论

3.1. 消息队列

3.1.1. 消息队列的定义和特点

消息队列是一种异步、非实时的通信模式，传递的内容可以如文本、图像、声音等各种各样的媒体形式。队列是用来存储消息的公用存储空间，可以是存储即将发送的消息，也可以是存储接收到的消息。它有俩种存在于内存中或物理文件中对应于消息的发送方式，分别是快递方式(express)和可恢复模式(recoverable) [1]。如图 1 所示。

消息队列具有先进先出(FIFO)的属性，这意味着消息在队列中按照它们被发送的顺序进行有序处理。当消息被发送到队列时，它们会按照发送顺序排列，并依次由消费者取出和处理。这保证了消息在队列中的顺序性和一致性，即最早进入队列的消息将首先被处理，后续消息则按照它们进入队列的顺序逐一处理[2]。

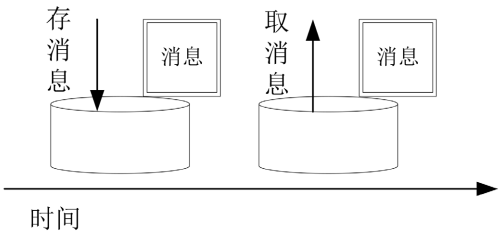

Figure 1. Message queue first transmission process
图 1. 消息队列 1 次传输过程

3.1.2. 消息队列的基本原理

消息队列是一种用于在应用程序之间异步传递消息的中间件。它基于发布–订阅模式或者点对点模式，起到解耦、异步、削峰填谷和可靠传递的作用。其中消息队列的基本原理是生产者将消息发送到消息队列中，然后消费者从队列中获取消息并进行处理。这种异步的方式使得生产者和消费者无需直接通信，提供了解耦、异步和伸缩性的优势[3]。同时，消息队列还提供了消息的持久化、优先级控制、消息过滤和重试机制等特性，确保消息的可靠传递，如图 2 所示。

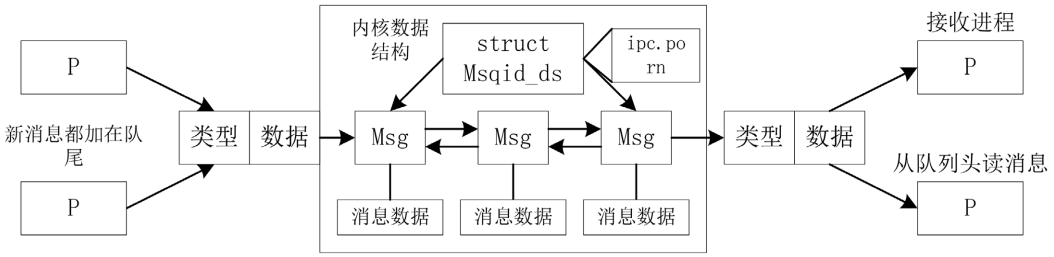

Figure 2. Principle of message queuing
图 2. 消息队列原理

3.1.3. 消息队列优势

消息队列有三个优势：首先它不用与发送和接收的进程同时进行，因此可以避免 open 和 close 时可能会带来的一些问题。其次，消息阻塞和同步等问题可以通过消息队列解决。最后服务器可以通过判断消息数据的类型来有选择性地接收消息数据，这样可以区别于命名管道中的默认消息接收[4]。

通过引入消息队列，系统可以更容易地实现水平扩展和分布式处理。生产者可以通过发送消息到消息队列的方式，将负载分散到多个消费者上，从而提高系统的处理能力和整体吞吐量。并且消息队列可以实现了生产者和消费者之间的解耦，使得它们可以异步地进行通信。生产者将消息发送到队列中后即可继续处理其他任务，而不需要等待消费者的实时响应。这种异步通信方式提高了系统的效率和并发性能[5]。

同时消息队列可以作为缓冲区，在生产者和消费者之间平衡流量，可以有效地削峰填谷。当生产者生成消息的速度超过消费者处理的速度时，消息会被存储在队列中，进而减少了对消费者的压力。这种能力可以防止系统因突发高负载而崩溃，并提高了系统的稳定性。

3.1.4. 消息中间件

消息中间件是一种独立的系统软件或服务程序，用于在分布式系统中提供消息传递和协调的功能[6]。它通过消息队列技术为程序提供一种异步通信方式。消息中间件充当了消息的传递和交换的中转站，使不同组件(应用程序、服务、系统)能够通过发送和接收消息来进行通信。消息队列中间件的主要特点是采用了队列化的消息传递机制。消息队列中间件主要特点有：多样化的通信方式、具备异步传输机制及松

耦合结构、消息传输可靠性强、消息优先级、流量易控制、支持多平台多传输协议。这些特点使得消息队列中间件成为构建分布式系统、微服务架构和异步通信的重要工具，能够提供可靠、高效和可伸缩的消息传递机制。

3.2. 消息队列的设计模式

周士杰曾为应用程序提供了五种消息队列设计方案，其所提出的五种设计模式分别是无确认更新、伪同步更新、异步确认更新、异步查询和伪同步查询[7]。

笔者认为其中伪同步更新和伪同步查询因为其处理的对象不同，而设计方案却基本一致，都是在底层使用了异步处理来提高性能和吞吐量，因此可以将他们划分为一类。异步查询和异步确认更新同理，所以周士杰所提出的五种设计模式可以简化为三种。而在今日，其所提出的五种设计模式更确切的说是五种编程模式，是针对数据的执行模式，MQ 可以与它们结合使用，但它们并不仅仅局限于消息队列。而现今消息队列的五种设计模式并没有具体的人在某个特定时间提出来，它们是在软件工程领域的实践中逐渐演化和形成的常见模式，它们分别是：

(1) 简单模式

一个生产者只对应一个消费者，这种模式实现了解耦，允许生产者和消费者在时间和空间上解耦，从而提高系统的可扩展性和灵活性，如图 3 所示。

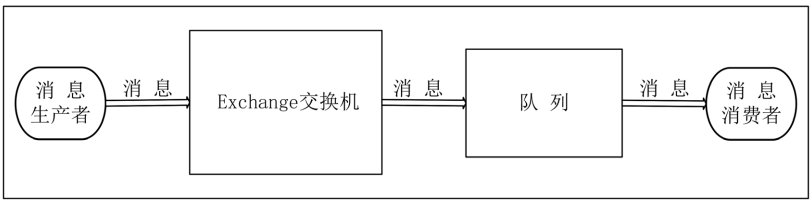

Figure 3. Simple mode
图 3. 简单模式

(2) 工作队列模式

一个生产者发送消息到一个队列中，消费者们共享一个队列，每个消费者获取的消息是唯一的，如图 4 所示。

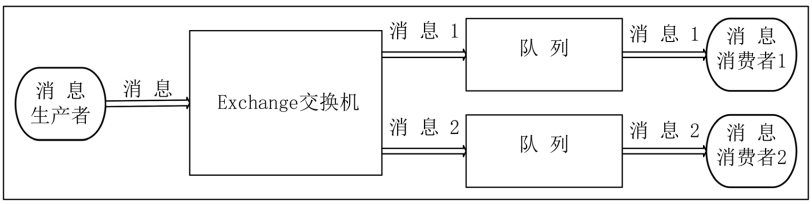

Figure 4. Work queue mode
图 4. 工作队列模式

(3) 发布/订阅模式

一个生产者发送消息，多个消费者只能获取同样消息，其在消息队列服务器中增加了交换机。生产者将消息发送到交换机，并且每一个消费者都对应着一个自己的专属队列，消息通过交换机后才能到达每个消费者的队列，消费者再从队列中获取消息。但是交换机没有存储消息功能，因此消息传输可靠性较低。这种模式适用于广播消息、事件通知等场景，如图 5 所示。

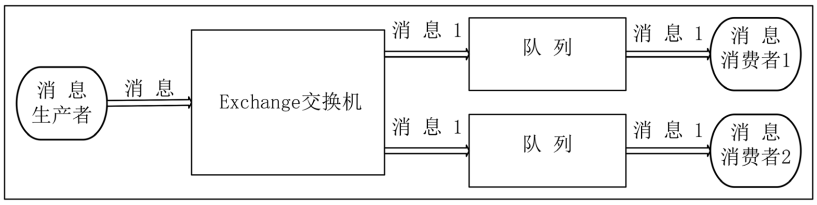

Figure 5. Publish/subscribe mode
图 5. 发布/订阅模式

(4) 路由模式

路由模式相比发布/订阅模式，其通过定义不用的路由 key 使得程序将消息发送到不同的队列中，这样每个消费者得到的信息不一样，如图 6 所示。

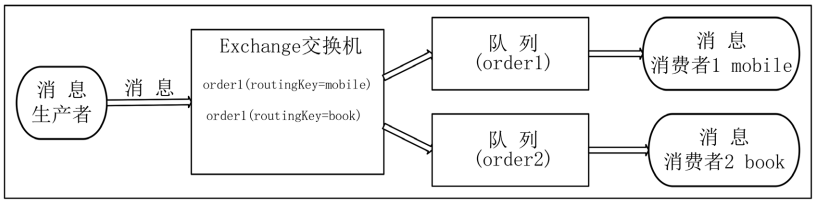

Figure 6. Routing mode
图 6. 路由模式

(5) 通配符模式

使用通配符来实现路由 key 将消息发送到一类相同的 key 中，如图 7 所示。

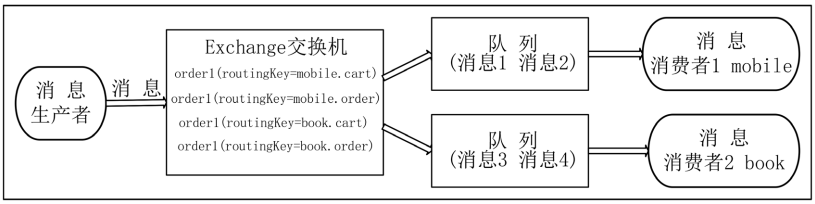

Figure 7. Wildcard mode
图 7. 通配符模式

这五种设计模式对比来看，点对点模式和发布/订阅模式是应用场景最多的俩种模式，其中点对点模式把队列作为中间存储，使通信具有异步性。而发布/订阅模式通过主题关联应用组件，使通信具有匿名性[8]。路由模式对于多个发送者和接收者之间的一对一通信非常高效，并且点对点模式通常使用路由模式来实现，但路由模式不仅仅用于点对点通信，它还可以用于更复杂的消息路由和传递场景。工作队列模式也称为任务队列模式，通常用于分发和处理耗时的任务。在这种模式下，多个工作者(或消费者)从一个队列中获取任务并处理它们，有助于实现并行处理和负载均衡。通配符模式扩展了发布订阅模式，使订阅者能够使用通配符匹配多个主题，适用于广播消息给多个订阅者并实现松散耦合的应用程序。

3.3. 常见消息队列系统

3.3.1. 系统描述

RabbitMQ 是一个基于 AMQP 协议可复用的企业级消息系统，它支持多种消息传递模式，如：发布–订阅模式、点对点模式。它提供了更高级的特性，如消息路由、消息确认和可靠性等，如图 8 所示。

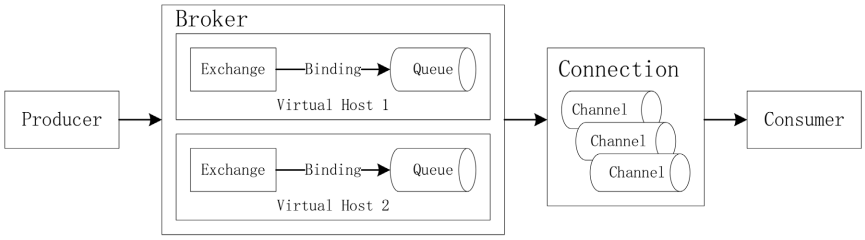

Figure 8. RabbitMQ architecture diagram
图 8. RabbitMQ 架构图

Apache Kafka 是一个高吞吐量和可扩展的分布式消息队列系统，采用发布－订阅模式传递消息，使用 Scala 语言开发。适用于实时流处理和大规模数据处理，同时它提供持久性、高可用性和容错能力，如图 9 所示。

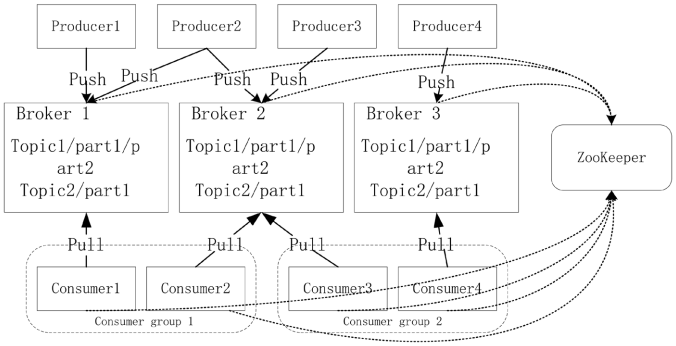

Figure 9. Apache Kafka architecture diagram
图 9. Apache Kafka 架构图

ActiveMQ 是一个开源的、基于 Java 的消息队列系统。它支持 JMS (Java Message Service)标准，提供了可靠的消息传递和事务支持，并且适用于各种企业应用。

RocketMQ 是阿里巴巴开源的分布式消息队列系统。它支持同步和异步消息传递模式，并提供了消息顺序和事务消息的支持，适用于高吞吐量、高可靠性和实时性要求较高的场景[9]，如图 10 所示。

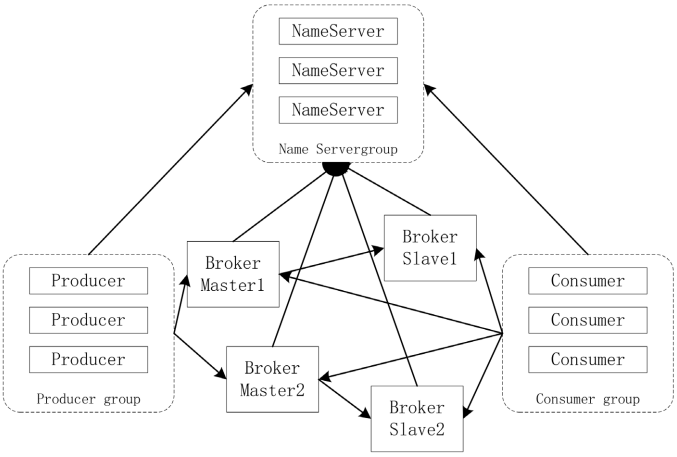

Figure 10. RocketMQ architecture diagram
图 10. RocketMQ 架构图

3.3.2. 系统比较和选择

在服务端发送消息的性能上，Kafka > RocketMQ > RabbitMQ > ActiveMQ。RabbitMQ 有负载均衡、数据持久化的特性，对路由也有很好的支持，而且其支持了多种协议，这便使得 RabbitMQ 更适合用于企业级的开发[10]。为了确保消息的高可靠性，RabbitMQ 需要加强其容错机制，提高故障恢复能力。同时，RabbitMQ 也需要优化其消息持久化策略和复制策略，以确保消息在出现故障时能够得到妥善处理。Kafka 消息系统具备高吞吐量的特性，也支持持久化，随着业务量的增长，Kafka 需要支持更大的消息处理能力和更高的吞吐量。因此，Kafka 需要进一步扩展其集群规模，提高硬件利用率，并优化分布式协调机制，以实现更高的吞吐量和更低的延迟。它依托 Zookeeper 进行管理，包括 Storm、Spark、Flink 等在内的很多系统都支持与 Kafka 耦合，基于 Kafka 消息中间件设计的各类数据处理系统在各行业得到广泛应用[11]。

ActiveMQ 允许应用程序之间通过消息进行通信，它具备可跨平台、多语言支持等特性，因此其在集成不同平台和使用不同语言编写应用时拥有显著的优势[12]。但是随着 ActiveMQ 的广泛应用，安全性问题也日益突出。因此，ActiveMQ 需要加强其安全机制，包括身份验证、授权和加密等，以保护用户数据的安全性。此外，ActiveMQ 还需要支持更多的安全协议和标准协议。

RocketMQ 基于 Kafka，改成了主从结构，并且相比 Kafka 其在事务性和可靠性方面做了调整和优化。[13] RocketMQ 需要进一步标准化其 API 和协议，以支持更多的应用场景和平台，并降低开发者的使用难度。

4. 消息队列技术的实现与应用

4.1. 一种共享内存的消息队列的实现方案

一般消息队列中存放的都是实际的消息数据，而王彦明却提出了一种新的思路，其设计了一个共享内存的消息队列，其所设计的消息队列中存放的是指向存放消息数据地址的指针而不是直接的消息数据，他认为共享内存队列的主要目的是提供一种路由方式并保证消息的传递，利用共享内存，可以在各个应用程序之间将数据共享。他提供了一种实现方式来达到让各分系统按照一种队列的模式来进行稳定的通信的目的。其所设计实现的消息队列分为发送队列和接收队列，并且一个发送队列可以被多个接收队列读取，这样有利于高效率的传输消息数据，而一个接收队列只能被一个发送队列接收，而由于此对应的关系特性，便有可能出现多个接受收队列中存取指向同一消息的指针的情况。因此，消息结构体中引入了 iReadCount 变量代表存放着同一消息数据所对应地址的指针的接收队列的数目。在消息需要发送出去的时候，iReadCount 变量通过遍历虚拟链路累加的方式来统计出具体的数值。之后，每当一个接受队列队首指针被读取，其队首位置后移一位，同时 iReadCount 变量的值减 1，当所有记录存放着该消息数据所对应的指针的接收队列被读取完，iReadCount 的值就减小为 0，至此，消息传输成功[14]。

4.2. 实际应用

消息队列在实际中的应用有异步处理、流量削峰、应用解耦和日志处理。

(1) 消息队列可以用于收集和分析日志数据。应用程序可以将日志消息发送到队列中，然后由专门的日志处理程序消费和分析日志数据，提供实时的监控和分析能力。

(2) 消息队列可以用来平衡系统的流量。当系统面临突发的高流量时，消息队列可以作为缓冲区，缓解系统压力，避免系统崩溃。这种能力对于电商促销活动、高流量网站等场景非常有用。

(3) 消息队列可以用于将不同的应用程序或服务之间进行解耦。通过消息队列，可以实现应用程序之间的松耦合，每个应用程序只需要关心发送和接收自己关心的消息，而不需要知道具体的实现细节[15]。

李海波与琚居森提出了针对高并发场景下短时间内流量暴增出现的系统性能问题，提出了一种基于页面控制、服务单元、数据缓存、流量控制技术的高性能系统架构。利用设计的消息队列实现了异步处理、应用解耦和流量削峰[16]。此系统在普通系统架构基础上，增加消息队列与数据缓存，通过微服务形式部署。此架构由表示层、服务单元层、缓存层和数据层组成，并基于 RateLimiter 流量控制器进行设计，实现限流，如图 11 所示。

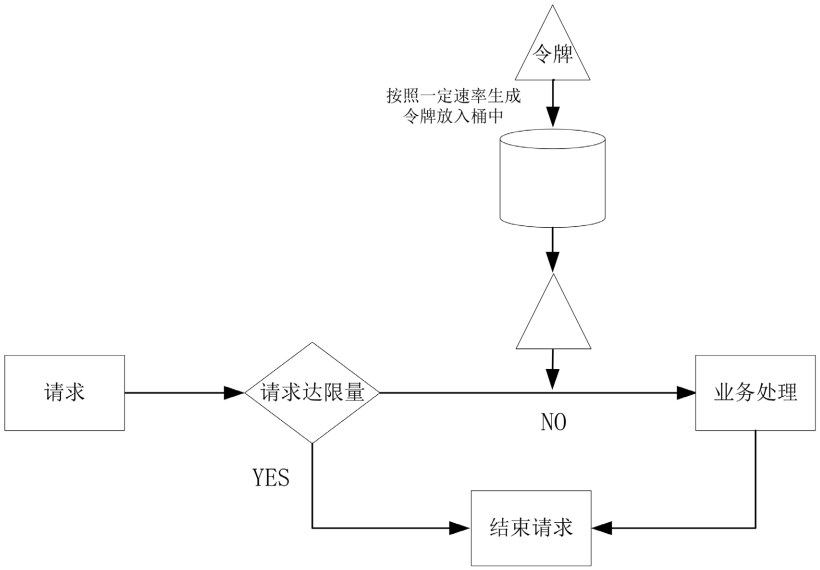

Figure 11. Operating principle of RateLimiter controller
图 11. RateLimiter 控制器运行原理

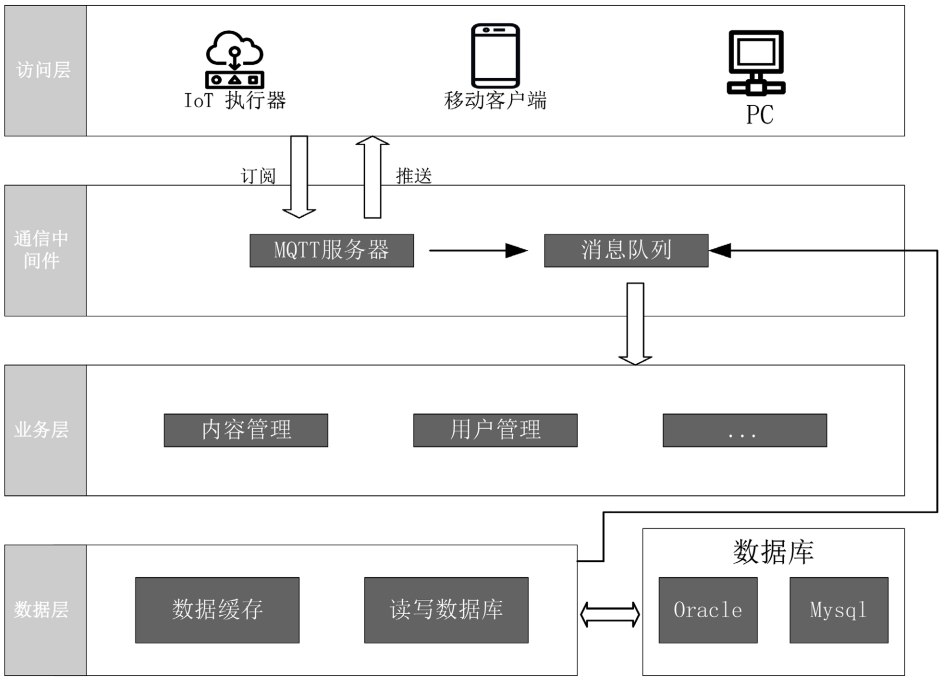

Figure 12. General information management system architecture
图 12. 通用信息管理系统架构

消息队列在物联网中的应用同样很广泛，王腾与郑静曾提出基于 MQTT 协议建立一个服务器中间消息队列，即在传统信息管理系统三层模型以外添加一个既独立于 MQTT 消息代理服务器，又不与三层之间进行耦合的消息数据封装、处理、传输模式层。该消息队列既可以通过多线程操作，同时也具备异常客户端的消息再处理流程[17]。传统的数据管理系统三层架构体系结构分别为表示层、业务层与数据层，以防数据丢失，在通用的信息管理系统的三层架构以外添加了一个通信中间件，其中包含通信接口设计，包括客户端的消息数据发送与接收，中间件以 MQTT 中间代理服务器为中转站，进行消息接收与转发，在收到消息后业务层处理数据，再通过数据层存储到数据库，再将该数据进行封装后通过通信中间件推送给作为订阅者的客户端，如图 12 所示。除此之外，本文中也举例介绍了消息队列实现过程中对消息序列化、如何进行多线程操作以及对再次入队的消息对象进行延时的核心代码。

5. 发展趋势

5.1. 消息队列面临的挑战

在实践中，许多企业使用不同的消息队列系统，并且面临不同版本的兼容性问题。集成现有应用程序和服务，并保持与其他系统的兼容性是一个复杂的任务，需要细心规划和实施。

传统的消息中间件(Messae-0rented m 尽管消息队列提供了许多优势和好处，但在实践中也面临一些挑战。

(1) 数据一致性问题

因为消息队列是异步通信的，生产者和消费者之间存在一定的延迟。这可能导致数据的一致性问题，尤其是在需要强一致性的业务场景中。确保消息在队列中按照正确的顺序被消费，以及处理消息失败或重复消费的情况，对于保持数据一致性至关重要。

(2) 系统的复杂性

引入消息队列将增加系统的复杂性，需要处理更多的组件和依赖关系。正确配置、管理和监控消息队列是一项挑战，需要考虑各种因素，如消息丢失、重复消息、死信队列、消息积压等。[18]同时在消息队列中也可能发生的故障情况包括队列溢出、系统宕机等。处理这些故障并避免系统阻塞或消息丢失是一项具有挑战性的任务。需要合理地配置队列大小、考虑持久化和备份策略，并采取适当的监控和报警机制。

(3) 复杂的集成和版本兼容性问题

Middleware.MOM)基于消息队列和授权机，在不同企业计算平台能高效且可靠地交换数据，具有松合、异步可靠和持久化等特点，但由于大多基于远程过程调用(Remote Process Ca.PC)的体系结构而难以满足企业间大规模交互的需求[19]，这使得消息队列的发展受到一定约束。

(4) 实时性要求更高

随着企业对实时数据处理的需求增加，消息队列技术需要更加高效地处理高并发、低延迟的消息传递。

(5) 数据安全与隐私保护

在数据传输过程中，如何保证数据的安全性和隐私保护，防止数据泄露和攻击，成为亟待解决的问题。

(6) 边缘计算与物联网的发展

随着物联网设备的普及，如何有效管理海量设备产生的数据，以及如何在设备边缘进行计算和处理，将成为未来消息队列技术的面临的重要挑战。

5.2. 消息队列的未来发展趋势

(1) 数据实时性要求提高

随着企业对实时数据处理的需求增加，消息队列技术发展将更加注重实时性处理能力。根据 Gartner (美国高德纳咨询公司)发布的调研报告预测表明，到 2025 年，有超过一半的大型企业将使用实时数据流处理平台。消息队列技术如 Kafka，RabbitMQ 和 ActiveMQ 等消息队列也将通过改进其流处理能力来满足企业的需求。

(2) 可靠性要求提高

随着业务场景的复杂性和对数据一致性要求的进一步提高，消息队列技术的可靠性也一定会越来越受到重视。甚至一些消息队列产品已经开始引入分布式事务、双写数据模型等机制来提高消息的一致性和可靠性。

(3) 系统易用性

随着开源社区的发展，消息队列的易用性将得到进一步提升。更多的企业将选择开源的消息队列产品，如 RabbitMQ、Kafka 等，这些产品通常具有友好的用户界面和丰富的社区支持。

(4) 云原生消息队列发展

随着云原生应用的兴起，消息队列将更加适应云环境，提供弹性伸缩、容器集成、自动备份等功能，以满足云原生应用的高性能和高可用性需求。相信在未来大多数企业都将使用云原生消息队列来替代传统的消息队列，如图 13 所示。

(5) 人工智能与消息队列的融合

随着 ChatGPT 人工智能技术的兴起，人们会越来越重视产品与人工智能的结合应用。消息队列如果与人工智能技术进一步融合，其数据传输效率和传输方式也将会得到提高和改变，实现更加智能化和自动化的消息处理和传递。例如，通过自然语言处理技术，可以实现人与机器之间的通信，提高消息处理的效率；通过机器学习技术，可以自动优化消息队列的性能，提高系统的整体性能和稳定性。

(6) 性能优化

随着云计算的普及，消息队列的性能也在不断提高。目前，许多消息队列产品都提供了内置的负载均衡和故障转移功能，这使得它们能够更好地应对高并发和故障情况。另一个重要的趋势是利用更高效的算法和数据结构来提高消息队列的吞吐量和延迟。

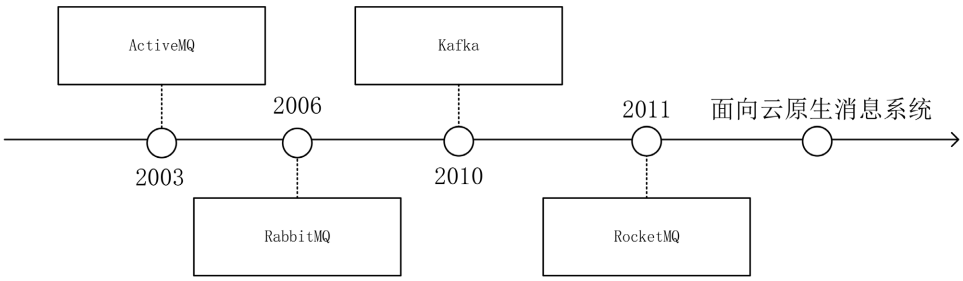

Figure 13. Development of message middleware towards cloud native
图 13. 消息中间件朝云原生发展

6. 总结

综上所述，消息队列是一种常见的通信模式，用于解耦和异步通信。它通过将消息发送到队列中，允许发送者和接收者解耦，并支持异步通信。同时消息队列提供了解耦性，使系统的组件能够独立演化

和扩展，提高系统的可靠性和可维护性。异步通信是消息队列的一大优势，发送者发送消息后即可继续执行其他操作，而无需等待接收者的回应。除此之外消息队列也提供持久化机制，确保消息在发送和接收过程中的可靠性，即使在发生故障或重启后也能正确传递和处理消息。

通过消息队列，系统可以实现水平扩展，处理更大的请求负载。并且消息队列适用于许多应用场景，如微服务架构、异步任务处理、日志处理和事件驱动架构等。对于保证顺序的应用场景，一些消息队列系统也提供特定策略来确保消息的顺序传递。在了解消息队列的好处的同时，我们也得知道消息队列也面临一些挑战，而我们需要注重消息队列的可靠性、扩展性、顺序性和性能。采取适当的措施和技术，监控系统状态，及时解决问题，并根据实际需求进行优化和调整，以确保消息队列的高效和稳定运行。

致　谢

感谢北京信息科技大学大学生创新创业训练计划项目的资金支持，感谢张翠平老师的指导。

基金项目

基金资助由北京信息科技大学促进高校分类发展－大学生创新创业训练计划项目－计算机学院(5112310855)支持。

参考文献

[1] 林学峰, 徐世新. 消息队列通信机制在卫星计费系统中的应用[J]. 计算机应用, 2006(S1): 214-215+218.

[2] 简广林, 王颖. 利用消息队列实现三层结构的通信平台[J]. 现代电力, 2003(2): 72-75.

[3] 林学峰, 徐世新. 消息队列通信机制在卫星计费系统中的应用[J]. 计算机应用, 2006(S1): 214-215+218.

[4] 李妍. 消息队列在 Linux 线程或进程间通信中的应用[J]. 电子世界, 2019(17): 146-147.

[5] 金琪. 高性能消息队列中无锁数据结构的设计与实现[D]: [硕士学位论文]. 南京: 南京邮电大学, 2021.

[6] 杨朝军. 基于消息队列消息传输系统的设计与研究[D]: [硕士学位论文]. 西安: 第四军医大学, 2009.

[7] 周世杰, 刘锦德, 秦志光. 消息队列技术研究: 综述与一个实例[J]. 计算机科学, 2002(2): 84-86.

[8] 何华海. 基于消息的中间件设计模式和系统结构[D]: [硕士学位论文]. 北京: 中国科学院研究生院(软件研究所), 2004.

[9] 吴璨, 王小宁, 肖海力, 等. 分布式消息系统研究综述[J]. 计算机科学, 2019, 46(S1): 1-5+34.

[10] 刘邦, 余华平. Kafka 分布式消息队列的高性能研究[J]. 电脑知识与技术, 2019, 15(32): 4-6.

[11] 冯冼. 基于 Kafka 的高并发消息机制优化研究[D]: [硕士学位论文]. 湘潭: 湘潭大学, 2022. https://doi.org/10.27426/d.cnki.gxtdu.2021.001882

[12] 吴艳艳. JMS 消息中间件的研究与实现[D]: [硕士学位论文]. 成都: 电子科技大学, 2009.

[13] 骆文亮. 基于异步消息处理的 RabbitMQ 运行原理探讨[J]. 数码世界, 2017(11): 400.

[14] 王彦明, 毛元泽, 刘一臻. 航空电子系统消息队列的设计与实现[J]. 信息通信, 2017(7): 86-87.

[15] 郭小丹. 浅析开源 ActiveMQ 的特点及应用[J]. 山东工业技术, 2015(18): 263+274.

[16] 李海波, 琚森超. 高并发条件下消息队列的设计与实现[J]. 电脑与信息技术, 2023, 31(3): 43-46+64.

[17] 王腾, 郑静. 基于 MQTT 协议的服务器中间消息队列设计与实现[J]. 电脑编程技巧与维护, 2022(3): 52-55.

[18] 王小霞. 消息中间件在数据交换中的应用研究及其面临的挑战[J]. 电子技术应用, 2005(1): 1-4.

[19] 梁霄. 面向消息中间件的设计与研究[J]. 信息与电脑(理论版), 2016(10): 34-35.