1. **RabbitMQ**
2. 简介

是一个在AMP（高级消息队列协议）基础上完成的，可复用的企业级消息系统，是当前最主流的消息中间件之一

1. 主要特性
2. 可靠性：提供了多种技术可以在性能和可靠性之间进行权衡，这些技术包括持久性机制、投递确认、发布者证实、高可用机制。
3. 灵活的路由：消息在到达路由前是通过交换机进行路由的。如果你有复杂的路由需求，可以将这些交换机组合起来使用，甚至可以实现自己的交换机路由，当做插件来使用。
4. 消息集群：在相同局域网中的多个MQ服务器可以聚合在一起，作为一个独立的逻辑代理来使用。
5. 队列高可用：队列可以在集群中的机器上进行镜像，以确保硬件问题下消息的安全。
6. 支持多种协议：支持多种消息队列协议。
7. 支持多种语音：用Erlang语言编写，支持几乎所有语言。
8. 管理界面：有一个易用的用户界面，使得用户可以监控和管理消息broker的许多方面。
9. 跟踪机制：如果消息异常，提供消息跟踪机制。使用者可以找出发生了什么。
10. 插件机制：提供了许多插件，多方面进行扩展，也可以编写自己的插件。
11. 优点
12. 由于Erlang语言的特性，消息队列性能较好，支持高并发。
13. 健壮、稳定、易用、跨平台、支持多种语言、文档齐全。
14. 有消息确认机制和持久化机制，可靠性高。
15. 高度可定制的路由。
16. 管理界面较丰富，在互联网公司也有较大规模的应用，社区活跃度高。
17. 缺点
18. 不利于二次开发和维护。
19. 实现了代理架构，意味着消息在发送到客户端之前可以在中央节点上排队，使得其运行速度较慢，因中央节点增加了延迟，消息封装后也比较大。
20. 需要学习比较复杂的接口和协议，学习和维护成本较高。
21. **RocketMQ**
22. 简介

出自阿里的开源产品，用java语言实现，在设计时参考了Kafka，并做出了自己的一些改进，消息可靠性上比Kafka更好。

1. 主要特性
2. 基于队列模型：具有高性能、高可靠、高实时、分布式等特点。
3. Producer、Consumer、队列都支持分布式。
4. Producer向一些队列轮流发送消息，队列集合称为Topic。Consumer如果做广播消费，则一个Consumer实例消费这个Topic对应的所有队列；如果做集群消费，则多个Consumer实例平均消费这个Topic对应的队列集合。
5. 能够保证严格的消息顺序。
6. 提供丰富的消息拉取模式。
7. 高效的订阅者水平扩展能力。
8. 实时的消息订阅机制。
9. 亿级消息堆积能力。
10. 较少的外部依赖。
11. 优点
12. 单机支持1万以上持久化队列。
13. 所有消息都是持久化的，先写入系统pagecache，然后刷盘，可以保证内存与磁盘都有一份数据，而访问时，直接从内存读取。
14. 模型简单，接口易用。
15. 性能非常好，可以允许大量堆积消息在broker中。
16. 支持多种消费模式，包括集群消费、广播消费等。
17. 各个环节分布式扩展设计，支持主从和高可用。
18. 开发度较活跃，版本更新快。
19. 缺点
20. 支持的客户端语言不多，目前就java较成熟。
21. 社区关注度及成熟度不高。
22. 没有WEB管理界面，提供了一个CLI（命令行界面）管理工具带来查询、管理和诊断各种问题。
23. 没有在MQ核心里实现JMS等接口
24. **ActiveMQ**
25. 简介

是Apache出品，非常快速，支持多种 语言的客户端协议，而且可以非常容易的嵌入到企业的应用环境中，并有许多高级功能。

1. 主要特性
2. 服从JMS规范：同步或异步的消息分发，一次和仅一次的消息分发，消息接收和订阅等等。
3. 连接灵活性：HTTP/S、IP多播、SSL、TCP、UDP等等协议
4. 支持的协议种类多：OpenWire、STOMP、REST、XMPP、AMQP
5. 持久化插件和安全插件：提供了多种持久化选择，安全性也可以依赖用户需求进行自定义鉴权和授权。
6. 支持多种客户端语言：java、C/C++、.NET、PHP、Python、Ruby等
7. 代理集群：多个MQ代理可以组成一个集群来提供服务。
8. 异常简单的管理：ActiveMQ是以开发者思维被设计的，不需要专门的管理员，它提供了简单的管理特性。可以监控不同层面的的数据。通过处理JMX的告警消息，通过使用命令脚本、甚至可以通过监控各种类型的日志。
9. 优点
10. 跨平台（JAVA编写与平台无关，几乎可以运行在任何的JVM上）。
11. 可以使用JDBC：可以将数据持久化到数据库。
12. 支持JMS规范。
13. 支持自动重连和错误重试机制。
14. 有安全机制：支持基于shiro、jaas等多种安全配置机制，可以队Queue/Topic进行认证和授权。
15. 监控完善：包括WebConsole、JMX、Shell命令，Jolokia的RESTful API。
16. 界面友善：提供的WebConsole可以满足大部分情况，还有很多第三方组件可以使用。
17. 缺点
    1. 社区活跃度不及RabbitMQ。
    2. 会出现莫名的问题，会丢失消息。
    3. 不适合用于上千个队列的应用场景。
18. **Kafka**
19. 简介

是一个分布式消息发布订阅系统。

1. 主要特性
2. 可以在O（1）的系统开销下进行消息持久化。
3. 高吞吐：在一台普通的服务器上可以达到10W/s的吞吐速度。
4. 完全的分布式系统：Broker、Producer、Consumer都原生自动支持分布式，自动实现负载均衡。
5. 支持同步和异步复制两种高可用机制。
6. 支持数据批量发送和拉去。
7. 零拷贝技术：减少IO操作步骤，提高系统吞吐量。
8. 数据迁移和扩容对用户透明。
9. 无需停机就可扩展机器。
10. 丰富的消息拉去模型、高效订阅者水平扩展、实时的消息订阅、亿级的消息堆积能力、定期删除机制。
11. 优点
12. 客户端语言丰富：java、.Net、PHP、Python、Go等多种语言。
13. 高性能：单机写入TPS约在100万条/秒，消息大小10字节。
14. 提供完全分布式架构，并有replica机制，拥有较高的可用性和可靠性，理论上支持消息无限堆积。
15. 支持批量操作。
16. 消费者采用pull方式获取消息，消息有序，通过控制能够保证所有消息被消费且仅被消费一次。
17. 有优秀的第三方KafkaWeb管理界面。
18. 在日志领域比较成熟，被多家公司和多个开源项目使用。
19. 缺点
20. 单机超过64个队列/分区时，加载时会发送明显的飙高现象，队列越多，负载越高，发送消息响应时间变长。
21. 使用短轮询方式，实时性取决于轮询间隔时间。
22. 消息失败不支持重试。
23. 支持消息顺序，但是一台代理宕机，就会产生消息乱序。
24. 社区更新较慢。

