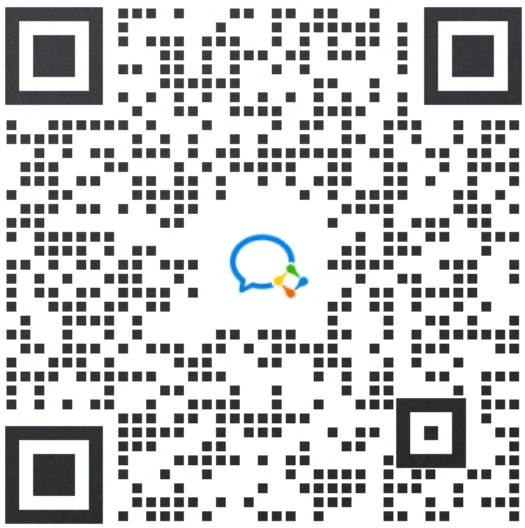
7. RabbitMQ常见面试题

版权说明



1. 面试题分析

本章节内容由最近半年的面试题分析得来, 分析时包含所有消息队列相关的面试题, 重点讨论RabbitMQ相关的

分析结果如下:

整体: 452条, 其中通用问题占: 53%, RabbitMQ : 24.32% RocketMQ: 19.25%, Kafka:3.38%, 其他 0.05%

详细:

> 只摘出可以移植到RabbitMQ的相关问题(包括通用问题里的部分)

问题	频率
可靠性保证	74
如何保证消息消费的幂等性	43
MQ的作用及应用场景	39
Kafka, RabbitMQ, RocketMQ区别	35
MQ如何保证消息的顺序	27
死信队列	23
延迟队列	20
MQ积压的原因, 如何处理	9
RabbitMQ的基本架构, 核心流程简单介绍	7
RabbitMQ工作模式	5
Rabbitmq的特性	3
RabbitMQ是推送式还是拉取式	1

2. 常见面试题

2.1 MQ的作用及应用场景

> 类似问题:
>
> 项目什么场景下使用到了MQ, 为什么需要MQ
>
> RabbitMQ 的作用？使用场景有哪些
>
> RabbitMQ的主要应用场景
>
> 消息队列解耦应用程序的例子
>
> 消息队列的应用场景
>
> 为什么说消息队列可以削峰

消息队列（MQ）是一种应用程序间的通信方法，它允许系统组件以异步的方式进行交互, 在不同的应用场景下可以展现不同的作用, 常见的应用场景如下:

- **异步解耦:** 在业务流程中, 一些操作可能非常耗时, 但并不需要即时返回结果. 可以借助MQ把这些操作异步化, 比如 用户注册后发送注册短信或邮件通知, 可以作为异步任务处理, 而不必等待这些操作完成后才告知用户注册成功.
- **流量削峰:** 在访问量剧增的情况下, 应用仍然需要继续发挥作用, 但是是这样的突发流量并不常见. 如果以能处理这类峰值为标准而投入资源,无疑是巨大的浪费. 使用MQ能够使关键组件支撑突发访问压力, 不会因为突发流量而崩溃. 比如秒杀或者促销活动, 可以使用MQ来控制流量, 将请求排队, 然后系统根据自己的处理能力逐步处理这些请求.
- **异步通信:** 在很多时候应用不需要立即处理消息, MQ提供了异步处理机制, 允许应用把一些消息放入MQ中, 但并不立即处理它,在需要的时候再慢慢处理.
- **消息分发:** 当多个系统需要对同一数据做出响应时, 可以使用MQ进行消息分发. 比如支付成功后, 支付系统可以向MQ发送消息, 其他系统订阅该消息, 而无需轮询数据库.
- **延迟通知:** 在需要在特定时间后发送通知的场景中, 可以使用MQ的延迟消息功能, 比如在电子商务平台中，如果用户下单后一定时间内未支付，可以使用延迟队列在超时后自动取消订单
- ……

2.2 了解过哪些MQ,以及他们的区别

> 类似问题:
>
> 了解过哪些MQ, 与其他同类产品的对比
>
> kafka 和 RabbitMQ的对比
>
> 对比其他消息队列，不同mq分别用在什么场景
>
> kafka和rocketmq比较
>
> 消息队列除了使用RabbitMQ,可以用RocketMQ吗?

目前业界有很多的MQ产品, 例如RabbitMQ, RocketMQ, ActiveMQ, Kafka, ZeroMQ等,

简单介绍其中3种:

1. Kafaka

Kafka一开始的目的就是用于日志收集和传输，追求高吞吐量, 性能卓越, 单机吞吐达到十万级, 在日志领域比较成熟, 功能较为简单, 主要支持简单的 MQ 功能. 适合大数据处理, 日志聚合, 实时分析等场景

2. RabbitMQ

采用Erlang语言开发, MQ 功能比较完备, 且几乎支持所有主流语言, 开源提供的界面也非常友好, 性能较好, 吞吐量能达到万级, 社区活跃度较高，文档更新频繁, 比较适合中小型公司, 数据量没那么大, 且并发没那么高的场景.

3. RocketMQ

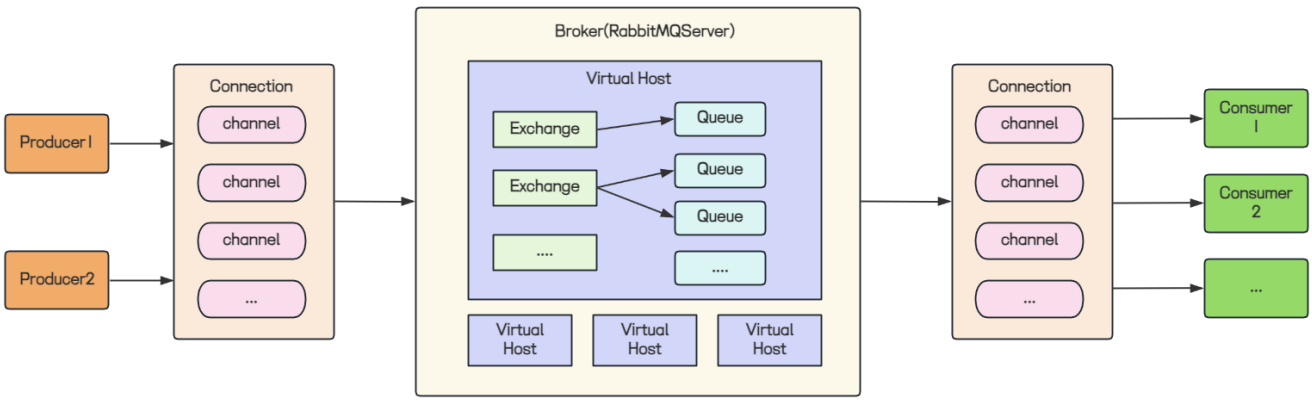
采用Java语言开发, 由阿里巴巴开源, 后捐赠给了Apache. 在可用性, 可靠性以及稳定性等方面都非常出色, 吞吐量能达到十万级, 在Alibaba集团内部广泛使用, 但支持的客户端语言不多, 产品较新文档较少, 且社区活跃度一般. 适合于大规模分布式系统, 可靠性要求高, 且并发大的场景, 比如互联网金融.

这些消息队列, 各有侧重, 没有好坏, 只有适合不适合, 在实际选型时, 需要结合自身需求以及MQ产品特征, 综合考虑

2.3 介绍下RabbitMQ的核心概念及工作流程

> 相关面试题:
>
> RabbitMQ的核心流程简单介绍一下
>
> 讲下RabbitMQ的结构

我们先来看RabbitMQ的工作流程



RabbitMQ是一个消息中间件, 也是一个生产者消费者模型. 它负责接收, 存储并转发消息. 根据工作流程图, 来介绍它的核心概念:

- Producer: 生产者, 向RabbitMQ发送消息
- Consumer: 消费者, 从RabbitMQ接收消息
- Broker: 消息队列服务器或服务实例, 也就是RabbitMQ Server
- Connection: 网络连接, 它允许客户端与 RabbitMQ通信
- Channel: 连接里的一个虚拟通道, 发送或者接收消息都是通过通道进行的.
- Exchange: 交换机. 负责接收生产者发送的消息, 并根据路由算法和规则将消息路由到一个或多个队列
- Queue: 消息队列, 存储消息直到它们被消费者消费

工作流程:

1. 创建连接: Producer 连接到RabbitMQBroker, 建立一个连接(Connection), 开启一个信道(Channel)

2. 声明交换机和队列,以及绑定规则: Producer 声明一个交换机(Exchange)和队列, 并绑定Queue到Exchange
3. 发布消息: Producer 发送消息至RabbitMQ Broker
4. 消息存储: RabbitMQ Broker 接收消息, 并存入相应的队列(Queue)中, 如果未找到相应的队列, 则根据生产者的配置, 选择丢弃或者退回给生产者.
5. 消费消息: 消费者监听Queue, 当消息到达时, 从Queue中获取消息, 处理后, 向RabbitMQ发送消息确认
6. 消息删除: 消息被确认后, RabbitMQ 会把消息从Queue中删除.

2.4 RabbitMQ如何保证消息的可靠性

> 相关面试题:
>
> RabbitMQ消息丢失原因及其解决方案
>
> 如何保证消息不丢失
>
> 消息写入失败怎么办
>
> 消息消费失败如何处理
>
> MQ的主动ack和被动ack有什么区别
>
> RabbitMQ如何解决数据丢失问题, 如何保证一致性
>
> 消息队列怎么保证消费者的消息不丢失的?

从以下几个方面来回答

1. 发送方投递可靠性
2. RabbitMQ可靠性
3. 消费者可靠性

参考前面章节[**RabbitMQ高级特性-发送方确认-常见面试题**]

2.5 RabbitMQ如何保证消息的顺序性

> 相关面试题:
>
> RabbitMQ怎么保证消息的顺序性?
>
> 如何保证消息能够有序消费

- 单队列单消费者
- 分区消费
- 事务和消息确认机制
- 业务逻辑控制, 比如消费端内部实现消息排序逻辑
- ……

参考[**RabbitMQ应用问题-顺序性保障**]

2.6 如何保证消息消费时的幂等性

> 相关问题:
>
> RabbitMQ怎么保证消息不重复消费
>
> 消息或请求存在重复消费问题吗? 是怎么解决的?
>
> 怎么解决MQ重复消费的问题

- 全局唯一ID
- 业务逻辑判断
- ……

参考[**RabbitMQ应用问题-幂等性保障**]

2.7 RabbitMQ有哪些特性

- 发送方消息确认
- 持久化
- 消费端消息确认
- 重试机制
- TTL
- 死信队列
- ……

参考[**RabbitMQ高级特性**]

2.8 介绍下RabbitMQ的死信队列

> 类似问题:
>
> RabbitMQ的死信队列以及应用场景

从以下方面来回答:

- 死信队列的概念
- 死信的来源
- 死信队列的应用场景

参考[**RabbitMQ高级特性-死信队列**]

2.9 介绍下RabbitMQ的延迟队列

> 类似问题:
>
> rabbitmq延迟队列的实

从以下三个方面来回答

1. 概念
2. 应用场景
3. 实现方式

参考[**RabbitMQ高级特性-延迟队列**]

2.10 介绍下RabbitMQ的工作模式

> 相关面试题:
>
> RabbitMQ的几种模式, work模式怎么实现的能者多劳

1. Simple(简单模式)
2. Work Queue(工作队列)
3. Publish/Subscribe(发布/订阅)

4. Routing(路由模式)
5. Topics(通配符模式)
6. RPC(RPC通信)
7. Publisher Confirms(发布确认)

2.11 消息积压的原因, 如何处理

> 类似问题:
>
> MQ消息堆积问题
>
> 如果解决MQ的数据囤积?

消息积压的原因:

- 消息生产过快
- 消费者能力不足
- 网络问题
- RabbitMQ服务配置偏低
- ……

解决方案:

- 提高消费者效率, 比如增加机器, 优化业务逻辑
- 限制消费者生产速率
- 资源配置优化

参考[**RabbitMQ应用问题-消息积压问题**]

2.12 RabbitMQ是推模式还是拉模式

概念

RabbitMQ支持两种消息传递模式: 推模式(push)和拉模式(pull)

推模式：消息中间件主动将消息推送给消费者.
拉模式：消费者主动从消息中间件拉取消息.

RabbitMQ主要是基于推模式工作的, 它的核心设计是让消息队列中的消费者接收到由生产者发送的消息. 使用channel.basicConsume方法订阅队列, RabbitMQ就会把消息推送到订阅该队列的消费者, 如果只想从队列中获取单条消息而不是持续订阅，则可以使用channel.basicGet方法来进行消费消息.

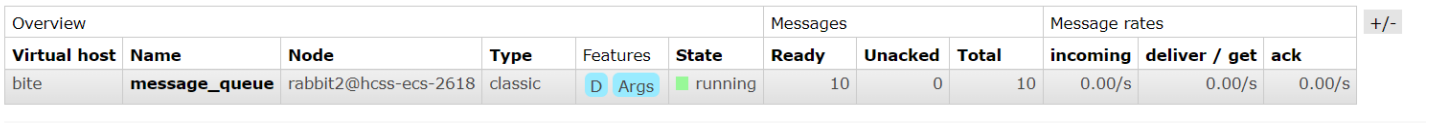
接下来通过代码演示下推模式和拉模式

实现

生产消息:

```
//1. 创建channel通道
Channel channel = connection.createChannel();
//2. 声明队列
channel.queueDeclare("message_queue",true,false,false,null);
//3. 通过channel发送消息到队列中
String msg = "hello message~~";
for (int i = 0; i < 10; i++) {
    channel.basicPublish("","message_queue",null,msg.getBytes());
}
```

运行程序, 生产10条消息成功



Overview						Messages			Message rates			+/-
Virtual host	**Name**	**Node**	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**	
bite	**message_queue**	rabbit2@hcss-ecs-2618	classic	D Args	running	10	0	10	0.00/s	0.00/s	0.00/s	

拉模式:

```
//1. 创建channel通道
Channel channel = connection.createChannel();
//2. 进行绑定，指定消费那个队列
channel.queueDeclare("message_queue",true,false,false,null);
//3. 使用pull模式, 获取消息
GetResponse getResponse = channel.basicGet("message_queue", true);
System.out.println(new String(getResponse.getBody()));
//4.释放资源
channel.close();
connection.close();
```

运行程序, 观察控制台, 只获取了一条消息:

```
hello message~~
```

队列中还剩余9条消息

Overview						Messages			Message rates			+/-
Virtual host	**Name**	**Node**	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**	
bite	**message_queue**	rabbit2@hcss-ecs-2618	classic	D Args	running	9	0	9	0.00/s	0.00/s	0.00/s	

推模式:

```
//1. 创建channel通道
Channel channel = connection.createChannel();
//2. 声明队列
channel.queueDeclare("message_queue",true,false,false,null);
//3. 接收消息, 并消费
DefaultConsumer consumer = new DefaultConsumer(channel) {
    @Override
    public void handleDelivery(String consumerTag, Envelope envelope,
AMQP.BasicProperties properties, byte[] body) throws IOException {
        System.out.println("接收到消息: " + new String(body));
    }
};
channel.basicConsume("message_queue", true, consumer);
//4. 等待消费结束, 再释放资源
Thread.sleep(2000);
channel.close();
connection.close();
```

运行程序, 观察控制台, 剩余消息全部推送到了消费者:

```
接收到消息: hello message~~
接收到消息: hello message~~
接收到消息: hello message~~
接收到消息: hello message~~
接收到消息: hello message~~
接收到消息: hello message~~
接收到消息: hello message~~
接收到消息: hello message~~
接收到消息: hello message~~
```

队列中没有消息了

Overview						Messages			Message rates			+/-
Virtual host	**Name**	**Node**	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**	
bite	**message_queue**	rabbit2@hcss-ecs-2618	classic	D Args	running	0	0	0	0.00/s	0.00/s	0.00/s	

总结

RabbitMQ 支持两种消息传递模式

推模式:

对消息的获取更加实时, 适合对数据实时性要求比较高时, 比如实时数据处理, 如监控系统, 报表系统等.

拉模式:

消费端可以按照自己的处理速度来消费, 避免消息积压, 适合需要流量控制, 或者需要大量计算资源的任务, 拉取模式允许消费者在准备好后再请求消息, 避免资源浪费.