**消息中间件篇**

1. **RabbitMQ如何保证消息不丢失？**

**候选人**：

我们使用RabbitMQ来确保MySQL和Redis间数据双写的一致性，这要求我们实现消息的高可用性，具体措施包括：

1. 开启生产者确认机制和消费者确认机制，确保消息能被送达队列和消息能够被消费者接受到
2. 启用持久化功能，保证消息在未消费前不会在队列中丢失，需要对交换机、队列和消息本身都进行持久化。

2. **RabbitMQ消息的重复消费问题如何解决？**

**候选人**：

我们遇到过消息重复消费的问题，处理方法是：

* 设置消费者为自动确认模式，如果服务在确认前宕机，重启后可能会再次消费同一消息。
* 通过业务唯一标识检查数据库中数据是否存在，若不存在则处理消息，若存在则忽略，避免重复消费。

3. **那你还知道其他的解决方案吗？**

**候选人**：

是的，这属于幂等性问题，可以通过以下方法解决：

* 使用Redis分布式锁来确保操作的幂等性。

4. **RabbitMQ中死信交换机了解吗？（RabbitMQ延迟队列有了解过吗？）**

**候选人**：

了解。我们项目中使用RabbitMQ实现延迟队列，主要通过死信交换机和TTL（消息存活时间）来实现。

* 消息若超时未消费则变为死信，队列可绑定死信交换机，实现延迟功能。
* 另一种方法是安装RabbitMQ的死信插件，简化配置，在声明交换机时指定为死信交换机，并设置消息超时时间。

5. **如果有100万消息堆积在MQ，如何解决？**

**候选人**：

若出现消息堆积，可采取以下措施：

1. 提高消费者消费能力，如使用多线程。
2. 增加消费者数量，采用工作队列模式，让多个消费者并行消费同一队列。
3. 扩大队列容量，使用RabbitMQ的惰性队列，支持数百万条消息存储，直接存盘而非内存。

6. **RabbitMQ的高可用机制了解吗？**

**候选人**：

我们项目在生产环境使用RabbitMQ集群，采用镜像队列模式，一主多从结构。

* 主节点处理所有操作并同步给从节点，若主节点宕机，从节点可接替为主节点，但需注意数据同步的完整性。

7. **那出现丢数据怎么解决呢？**

**候选人**：

使用仲裁队列，主从模式，基于Raft协议实现强一致性数据同步，简化配置，提高数据安全性。

**⏱ RAFT 的工作流程简化说明：**

**1. Leader 选举（Leader Election）**

* 初始时所有节点是 Follower。
* 如果 Follower 超时没收到 Leader 心跳，会变成 Candidate，发起投票。
* 获得超过半数票的 Candidate 成为新的 Leader。

**2. 日志复制（Log Replication）**

* Leader 接收客户端命令，写入日志，然后将日志复制给所有 Follower。
* 一旦超过半数节点写入成功，Leader 才提交日志。

**3. 安全性保障**

* RAFT 保证日志顺序一致性，任何一个已经被提交的日志条目在所有节点上最终都会被复制。

**🔄 心跳机制：**

* Leader 会定期给所有 Follower 发送心跳（空日志），防止他们超时成为 Candidate。

8. **Kafka是如何保证消息不丢失？**

**候选人**：

Kafka保证消息不丢失的措施包括：

1. 生产者使用异步回调发送消息，设置重试机制应对网络问题。
2. 在Broker中通过复制机制，设置acks参数为all，确保消息在所有副本中都得到确认。
3. 消费者手动提交消费成功的offset，避免自动提交可能导致的数据丢失或重复消费。

9. **Kafka中消息的重复消费问题如何解决？**

**候选人**：

通过以下方法解决Kafka中的重复消费问题：

* 禁用自动提交offset，手动控制offset提交时机。
* 确保消息消费的幂等性，例如通过唯一主键或分布式锁。

10. **Kafka是如何保证消费的顺序性？**

**候选人**：

Kafka默认不保证消息顺序性，但可以通过以下方法实现：

* 将消息存储在同一个分区，通过指定分区号或相同的业务key来实现。

11. **Kafka的高可用机制了解吗？**

**候选人**：

Kafka的高可用性主要通过以下机制实现：

* 集群部署，多broker实例，单点故障不影响整体服务。
* 复制机制，每个分区有多个副本，leader和follower，leader故障时从follower中选举新leader。

12. **解释一下复制机制中的ISR？**

**候选人**：

ISR（In-Sync Replicas）指与leader保持同步的follower副本。

* 当leader故障时，优先从ISR中选举新leader，因为它们数据一致性更高。

13. **Kafka数据清理机制了解吗？**

**候选人**：

Kafka的数据清理包括：

* 基于消息保留时间的清理。
* 基于topic数据大小的清理，可配置删除最旧消息。

14. **Kafka中实现高性能的设计有了解过吗？**

**候选人**：

Kafka高性能设计包括：

* 消息分区，提升数据处理能力。
* 顺序读写，提高磁盘操作效率。
* 页缓存，减少磁盘访问。
* 零拷贝，减少数据拷贝和上下文切换。
* 消息压缩，减少IO负载。
* 分批发送，降低网络开销。