RocketMQ 笔记以及相关面试题目

>

1、MQ的应用场景有哪些

答: 1) 异步解耦

2) 削峰填谷

2、MQ的缺点

答 : 1) 系统可用性降低

2)系统的复杂度提高

3)数据如何保证一致性

3、RocketMQ消息发送有几种方式以及各自特点

答:

发送方式	发送 TPS	发送结果反馈	可靠性
同步发送	快	有	不丢失
异步发送	快	有	不丢失
单向发送	最快	无	可能丢失

4、MessageQueueSelector的作用是什么?

答:MessageQueueSelector是队列选择器，实现其select接口可以实现自定义分区选择。

5、MessageListenerOrderly与MessageListenerConcurrently区别

答: MessageListenerOrderly:有序消费，同一队列的消息同一时刻只能一个线程消费，可保证消息在同一队列严格有序消费

MessageListenerConcurrently:并发消费

6、RocketMQ消费模式有哪两种?

答:集群消费模式 和广播消费模式

7、RocketMQ延迟消息实现原理?

答： 所有的延迟消息由producer发出之后，都会存放到同一个topic（**SCHEDULE_TOPIC_XXXX**）下，根据延迟level的个数，创建对应数量的队列，也就是说18个level对应了18个队列。注意，这并不是说这个内部主题只会有18个队列，因为Broker通常是集群模式部署的，因此每个节点都有18个队列。不同的延迟级别会对应不同的队列序号，当延迟时间到之后，由定时线程读取转换为普通的消息存的真实指定的topic下，此时对于consumer端此消息才可见，从而被consumer消费。

8、RocketMQ过滤消息有哪些实现方式?

答: TAG模式过滤 和SQL表达式过滤

9、什么是半事务消息?

答: 暂不能投递的消息，发送方已经成功地将消息发送到了消息队列 RocketMQ 版服务端，但是服务端未收到生产者对该消息的二次确认，此时该消息被标记成“暂不能投递”状态，处于该种状态下的消息即半事务消息。

10、什么是消息回查?

答: 由于网络闪断、生产者应用重启等原因，导致某条事务消息的二次确认丢失，消息队列 RocketMQ 版服务端通过扫描发现某条消息长期处于“半事务消息”时，需要主动向消息生产者询问该消息的最终状态（Commit 或是 Rollback），该询问过程即消息回查。

11、RocketMQ事务消息执行 执行本地事务有哪些返回状态?

答： LocalTransactionState.COMMIT_MESSAGE：提交消息，这个消息由prepared状态进入到commited状态，消费者可以消费这个消息；

LocalTransactionState.ROLLBACK_MESSAGE：回滚，这个消息将被删除，消费者不能消费这个消息； LocalTransactionState.UNKNOW：未知，这个状态有点意思，如果返回这个状态，这个消息既不提交，也不回滚，还是保持prepared状态，而最终决定这个消息命运的，是checkLocalTransaction这个方法。

12、RocketMQ架构中包含哪些主要角色

答: NameServer、Broker、Producer、Consumer

13 、nameserver 的特点有哪些

答： 1) nameserver 互相独立，彼此没有通信关系，单台 nameserver 挂掉，不影响其他

2) nameserver 不会有频繁的读写，所以性能开销非常小，稳定性很高。

14、Broker有哪些角色?

答: master 和 slave

Master和Slave的区别:在Broker的配置文件中，参数brokerId的值为0表明这个Broker是Master，大于0表明这个Broker是Slave，同时brokerRole参数也会说明这个Broker是Master还是Slave。

Master角色的Broker支持读和写，Slave角色的Broker仅支持读，也就是Proder只能和Master角色的Broker连接写入消息；Consumer可以连接Master角色的Broker，也可以连接Slave角色的Broker来读取消息。

15、Queue的作用是什么?

答: Message Queue：Topic的分区；用于并行发送和接收消息

16、什么是消费者组?

答: 一个消费者组，代表着一群topic相同，tag相同（即逻辑相同）的Consumer

17、消费者和Queue的订阅关系是什么?

答: 对于同一个消费组，一个分区只支持一个消费者来消费消息。多余的消费者不能消费消息。一个消费者可以订阅多个Queue

18、Offset 是什么? 有哪两种分类

答：Offset是指某个 Topic下的一条消息在某个 Message Queue里的 位置，通过 Offset的值可以定位到这条消息，或者指示 Consumer从这条消息 开始向后继续处理 。
Offset主要分为本地文件类型和 Broker代存 的类型两种 。

Broker代存： CLUSTERING 模式，使用 RemoteBrokerOffsetStore 。

本地文件类型：BROADCASTING模式下， 使用LocalfileOffsetStore 。

19、RocketMQ集群搭建方式有哪些以及各自优缺点?

答:

1）多 master 模式：

优点：所有模式中性能最高

缺点：单个 master 节点宕机期间，未被消费的消息在节点恢复之前不可用，消息的实时性就受到影响。

2) 多master多slave异步复制模式：

优点： 性能最好，在 master 宕机时，消费者可以从 slave读取消息，消息的实时性不会受影响，性能几乎和多 master 一样。

缺点：使用异步复制的同步方式有可能会有消息丢失的问题。

3) 多 master 多 slave 同步双写模式：

优点：同步双写的同步模式能保证数据不丢失。

缺点： 比异步复制的性能差10%。

20、消息的存储是如何构成

答: 消息的存储是由consumequeue和commitlog配合完成的。

commitlog：消息真正的物理存储文件

consumequeue: 是消息的逻辑队列，类似数据库的索引文件，存储的是指向物理存储的地址。

21、consume Queue的存储内容是什么?

答: consume Queue中每个消息索引信息长度为20bytes，包括8位长度的offset，记录commitLog中消息内容的位移；4位长度的size，记录具体消息内容的长度；8位长度的tagHashCode，记录消息的tag的哈希值（订阅时如果指定tag，会根据HashCode快速查找订阅的消息）

22、IndexFile的作用是什么?

答: IndexFile（索引文件）提供了一种可以通过key或时间区间来查询消息的方法。Index文件的存储位置是：$HOME \store\index${fileName}，文件名fileName是以创建时的时间戳命名的，固定的单个IndexFile文件大小约为400M，一个IndexFile可以保存 2000W个索引，IndexFile的底层存储设计为在文件系统中实现HashMap结构，故rocketmq的索引文件其底层实现为hash索引。

23、消息存储的清理机制是什么?

- 按时间清理，RocketMQ默认会清理3天前的commitLog文件；
- 按磁盘水位清理：当磁盘使用量到达磁盘容量75%，开始清理最老的commitLog文件。

24、RocketMQ刷盘方式有哪些?

答： 同步刷盘、异步刷盘，都是通过broker.conf配置文件里的flushDiskType参数设置的，这个参数配置成ASYNC_FLUSH、SYNC_FLUSH中的一个。

同步：可靠性高、安全性高

异步：效率性能高，高吞吐量

25、消息主从复制有哪两种方式?

答:同步和异步两种复制方式

26、集群模式下，消费者分配queue队列的策略哪些?

答:

算法名称	含义
AllocateMessageQueueAveragely	平均分配算法
AllocateMessageQueueAveragelyByCircle	基于环形平均分配算法
AllocateMachineRoomNearby	基于机房临近原则算法
AllocateMessageQueueByMachineRoom	基于机房分配算法
AllocateMessageQueueConsistentHash	基于一致性hash算法
AllocateMessageQueueByConfig	基于配置分配算法

27、Consumer消费消息有哪些方式?

答:MQPullConsumer和MQPushConsumer

push的方式是:消息发送到broker后，则broker会主动把消息推送给consumer即topic中；

pull的方式是:消息投递到broker后，消费端需要主动去broker上拉消息，即需要手动写代码实现。

28、集群消费方式下，消息消费失败后期望消息重试

答:　　需要在消息监听器接口的实现中明确进行配置（三种方式任选一种）：

1)消费端返回ConsumeConcurrentlyStatus.RECONSUME_LATER（推荐）

2)返回 Null

3) 抛出异常

29、死信队列是什么?

答: 当达到最大重试次数（默认16次），消息还是消费失败，RocketMQ不会将该消息丢弃而是会把它保存到死信队列中。

这种不能被消费者正常处理的消息我们一般称之为 死信消息（Dead-Letter Message），将存储死信消息的队列称之为 死信队列（Dead-Letter Queue，DLQ）

30、RocketMQ如何处理消息幂等

答: RocketMQ能够保证消息不丢失但不保证消息不重复。

最好的方式是以业务唯一标识作为幂等处理的关键依据如：订单号、流水号等作为幂等处理的关键依据。而业务的唯一标识可以通过消息 Key 设置。在消费端通过对key判断，确保消息的唯一性

31、如何快速处理消息堆积

答:1)消费端扩容；--通用方式

2）服务降级；--快速失败，不一定适用所有业务场景

3)跳过非重要消息：发生消息堆积时，如果消费速度一直追不上发送速度，可以选择丢弃不重要的消息

4)异常监控。--属于运维层面措施

31、RocketMQ提供哪些消息查询方式

答: Message Key 、 Unique Key 、Message Id

32、Rebalance的危害哪些?

答: Rebalance危害：

- **消费暂停：**
- **消费突增：**
- 重复消费：

33、Rebalance的触发场景有哪些?

答:

类别	典型场景
队列信息变化	broker宕机 broker升级等运维操作 队列扩容/缩容
消费者组信息变化	日常发布过程中的停止与启动 消费者异常宕机 网络异常导致消费者与Broker断开连接 主动进行消费者数量扩容/缩容 Topic订阅信息发生变化