特别放送（四）| 20道经典的Kafka面试题详解

2020-06-11 胡夕

Kafka核心源码解读 进入课程 >

讲述：胡夕

时长 28:03　大小 25.70M

你好，我是胡夕。这一期的“特别放送”，我想跟你分享一些常见的 Kafka 面试题。

无论是作为面试官，还是应聘者，我都接触过很多 Kafka 面试题。有的题目侧重于基础的概念考核，有的关注实际场景下的解决方案，有的属于“炫技式”，有的可算是深入思考后的“灵魂拷问”。“炫技”类的问题属于冷门的 Kafka 组件知识考核，而“灵魂拷问”类的问题大多是对 Kafka 设计原理的深入思考，有很高的技术难度。

每类题目的应对方法其实不太一样。今天，我就按照这 4 种类别，具体讲解 20 道面试题。不过，我不打算只给出答案，我会把面试题的考核初衷也一并写出来。同时，我还会给你分享一些面试小技巧，希望能够帮你更顺利地获取心仪的 offer。

那么，话不多说，我们现在开始吧。

基础题目

1.Apache Kafka 是什么?

这是一道很常见的题目，看似很无聊，其实考核的知识点很多。

首先，它考验的是，你对 Kafka 的定位认知是否准确。Apache Kafka 一路发展到现在，已经由最初的分布式提交日志系统逐渐演变成了实时流处理框架。因此，这道题你最好这么回答：**Apach Kafka 是一款分布式流处理框架，用于实时构建流处理应用。它有一个核心的功能广为人知，即作为企业级的消息引擎被广泛使用。**

其实，这里暗含了一个小技巧。Kafka 被定位为实时流处理框架，在国内的接受度还不是很高，因此，在回答这道题的时候，你一定要先明确它的流处理框架地位，这样能给面试官留下一个很专业的印象。

2. 什么是消费者组?

从某种程度上说，这可是个“送命题”。消费者组是 Kafka 独有的概念，如果面试官问这个，就说明他对此是有一定了解的。我先给出标准答案：**关于它的定义，官网上的介绍言简意赅，即消费者组是 Kafka 提供的可扩展且具有容错性的消费者机制。**切记，一定要加上前面那句，以显示你对官网很熟悉。

另外，你最好再解释下消费者组的原理：**在 Kafka 中，消费者组是一个由多个消费者实例构成的组。多个实例共同订阅若干个主题，实现共同消费。同一个组下的每个实例都配置有相同的组 ID，被分配不同的订阅分区。当某个实例挂掉的时候，其他实例会自动地承担起它负责消费的分区。**

此时，又有一个小技巧给到你：消费者组的题目，能够帮你在某种程度上掌控下面的面试方向。

如果你擅长**位移值原理**，就不妨再提一下消费者组的位移提交机制；

如果你擅长 Kafka **Broker**，可以提一下消费者组与 Broker 之间的交互；

如果你擅长与消费者组完全不相关的 **Producer**，那么就可以这么说：“消费者组要消费的数据完全来自于 Producer 端生产的消息，我对 Producer 还是比较熟悉的。”

使用这个策略的话，面试官可能会被你的话术所影响，偏离他之前想问的知识路径。当然了，如果你什么都不擅长，那就继续往下看题目吧。

3. 在 Kafka 中，ZooKeeper 的作用是什么?

这是一道能够帮助你脱颖而出的题目。碰到这个题目，请在心中暗笑三声。

先说标准答案：**目前，Kafka 使用 ZooKeeper 存放集群元数据、成员管理、Controller 选举，以及其他一些管理类任务。之后，等 KIP-500 提案完成后，Kafka 将完全不再依赖于 ZooKeeper。**

记住，**一定要突出“目前”**，以彰显你非常了解社区的演进计划。“存放元数据”是指主题分区的所有数据都保存在 ZooKeeper 中，且以它保存的数据为权威，其他“人”都要与它保持对齐。“成员管理”是指 Broker 节点的注册、注销以及属性变更，等等。“Controller 选举”是指选举集群 Controller，而其他管理类任务包括但不限于主题删除、参数配置等。

不过，抛出 KIP-500 也可能是个双刃剑。碰到非常资深的面试官，他可能会进一步追问你 KIP-500 是做的。一言以蔽之：**KIP-500 思想，是使用社区自研的基于 Raft 的共识算法，替代 ZooKeeper，实现 Controller 自选举**。你可能会担心，如果他继续追问的话，该怎么办呢？别怕，在下一期“特别发送”，我会专门讨论这件事。

4. 解释下 Kafka 中位移（offset）的作用

这也是一道常见的面试题。位移概念本身并不复杂，你可以这么回答：**在 Kafka 中，每个主题分区下的每条消息都被赋予了一个唯一的 ID 数值，用于标识它在分区中的位置。这个 ID 数值，就被称为位移，或者叫偏移量。一旦消息被写入到分区日志，它的位移值将不能被修改。**

答完这些之后，你还可以把整个面试方向转移到你希望的地方。常见方法有以下 3 种：

如果你深谙 Broker 底层日志写入的逻辑，可以强调下消息在日志中的存放格式；

如果你明白位移值一旦被确定不能修改，可以强调下“Log Cleaner 组件都不能影响位移值”这件事情；

如果你对消费者的概念还算熟悉，可以再详细说说位移值和消费者位移值之间的区别。

5. 阐述下 Kafka 中的领导者副本（Leader Replica）和追随者副本（Follower Replica）的区别

这道题表面上是考核你对 Leader 和 Follower 区别的理解，但很容易引申到 Kafka 的同步机制上。因此，我建议你主动出击，一次性地把隐含的考点也答出来，也许能够暂时把面试官“唬住”，并体现你的专业性。

你可以这么回答：**Kafka 副本当前分为领导者副本和追随者副本。只有 Leader 副本才能对外提供读写服务，响应 Clients 端的请求。Follower 副本只是采用拉（PULL）的方式，被动地同步 Leader 副本中的数据，并且在 Leader 副本所在的 Broker 宕机后，随时准备应聘 Leader 副本。**

通常来说，回答到这个程度，其实才只说了 60%，因此，我建议你再回答两个额外的加分项。

强调 Follower 副本也能对外提供读服务。自 Kafka 2.4 版本开始，社区通过引入新的 Broker 端参数，允许 Follower 副本有限度地提供读服务。

强调 Leader 和 Follower 的消息序列在实际场景中不一致。很多原因都可能造成 Leader 和 Follower 保存的消息序列不一致，比如程序 Bug、网络问题等。这是很严重的错误，必须要完全规避。你可以补充下，之前确保一致性的主要手段是高水位机制，但高水位值无法保证 Leader 连续变更场景下的数据一致性，因此，社区引入了 Leader Epoch 机制，来修复高水位值的弊端。关于“Leader Epoch 机制”，国内的资料不是很多，它的普及度远不如高水位，不妨大胆地把这个概念秀出来，力求惊艳一把。上一季专栏的第 27 节课讲的就是 Leader Epoch 机制的原理，推荐你赶紧去学习下。

实操题目

6. 如何设置 Kafka 能接收的最大消息的大小？

这道题除了要回答消费者端的参数设置之外，一定要加上 Broker 端的设置，这样才算完整。毕竟，如果 Producer 都不能向 Broker 端发送数据很大的消息，又何来消费一说呢？因此，你需要同时设置 Broker 端参数和 Consumer 端参数。

Broker 端参数：message.max.bytes、max.message.bytes（主题级别）和 replica.fetch.max.bytes。

Consumer 端参数：fetch.message.max.bytes。

Broker 端的最后一个参数比较容易遗漏。我们必须调整 Follower 副本能够接收的最大消息的大小，否则，副本同步就会失败。因此，把这个答出来的话，就是一个加分项。

7. 监控 Kafka 的框架都有哪些?

其实，目前业界并没有公认的解决方案，各家都有各自的监控之道。所以，面试官其实是在考察你对监控框架的了解广度，或者说，你是否知道很多能监控 Kafka 的框架或方法。下面这些就是 Kafka 发展历程上比较有名气的监控系统。

Kafka Manager：应该算是最有名的专属 Kafka 监控框架了，是独立的监控系统。

Kafka Monitor：LinkedIn 开源的免费框架，支持对集群进行系统测试，并实时监控测试结果。

CruiseControl：也是 LinkedIn 公司开源的监控框架，用于实时监测资源使用率，以及提供常用运维操作等。无 UI 界面，只提供 REST API。

JMX 监控：由于 Kafka 提供的监控指标都是基于 JMX 的，因此，市面上任何能够集成 JMX 的框架都可以使用，比如 Zabbix 和 Prometheus。

已有大数据平台自己的监控体系：像 Cloudera 提供的 CDH 这类大数据平台，天然就提供 Kafka 监控方案。

JMXTool：社区提供的命令行工具，能够实时监控 JMX 指标。答上这一条，属于绝对的加分项，因为知道的人很少，而且会给人一种你对 Kafka 工具非常熟悉的感觉。如果你暂时不了解它的用法，可以在命令行以无参数方式执行一下`kafka-run-class.sh kafka.tools.JmxTool`，学习下它的用法。

8.Broker 的 Heap Size 如何设置?

如何设置 Heap Size 的问题，其实和 Kafka 关系不大，它是一类非常通用的面试题目。一旦你应对不当，面试方向很有可能被引到 JVM 和 GC 上去，那样的话，你被问住的几率就会增大。因此，我建议你简单地介绍一下 Heap Size 的设置方法，并把重点放在 Kafka Broker 堆大小设置的最佳实践上。

比如，你可以这样回复：**任何 Java 进程 JVM 堆大小的设置都需要仔细地进行考量和测试。一个常见的做法是，以默认的初始 JVM 堆大小运行程序，当系统达到稳定状态后，手**

动触发一次 Full GC，然后通过 JVM 工具查看 GC 后的存活对象大小。之后，将堆大小设置成存活对象总大小的 1.5~2 倍。对于 Kafka 而言，这个方法也是适用的。不过，业界有个最佳实践，那就是将 Broker 的 Heap Size 固定为 6GB。经过很多公司的验证，这个大小是足够且良好的。

9. 如何估算 Kafka 集群的机器数量?

这道题目考查的是**机器数量和所用资源之间的关联关系。**所谓资源，也就是 CPU、内存、磁盘和带宽。

通常来说，CPU 和内存资源的充足是比较容易保证的，因此，你需要从磁盘空间和带宽占用两个维度去评估机器数量。

在预估磁盘的占用时，你一定不要忘记计算副本同步的开销。如果一条消息占用 1KB 的磁盘空间，那么，在有 3 个副本的主题中，你就需要 3KB 的总空间来保存这条消息。显式地将这些考虑因素答出来，能够彰显你考虑问题的全面性，是一个难得的加分项。

对于评估带宽来说，常见的带宽有 1Gbps 和 10Gbps，但你要切记，**这两个数字仅仅是最大值。**因此，你最好和面试官确认一下给定的带宽是多少。然后，明确阐述出当带宽占用接近总带宽的 90% 时，丢包情形就会发生。这样能显示出你的网络基本功。

10.Leader 总是 -1，怎么破?

在生产环境中，你一定碰到过“某个主题分区不能工作了”的情形。使用命令行查看状态的话，会发现 Leader 是 -1，于是，你使用各种命令都无济于事，最后只能用“重启大法”。

但是，有没有什么办法，可以不重启集群，就能解决此事呢？这就是此题的由来。

我直接给答案：**删除 ZooKeeper 节点 /controller，触发 Controller 重选举。Controller 重选举能够为所有主题分区重刷分区状态，可以有效解决因不一致导致的 Leader 不可用问题。**我几乎可以断定，当面试官问出此题时，要么就是他真的不知道怎么解决在向你寻求答案，要么他就是在等你说出这个答案。所以，**千万别一上来就说“来个重启”之类的话。**

炫技式题目

11.LEO、LSO、AR、ISR、HW 都表示什么含义?

在我看来，这纯属无聊的炫技。试问我不知道又能怎样呢？！不过既然问到了，我们就统一说一说。

LEO：Log End Offset。日志末端位移值或末端偏移量，表示日志下一条待插入消息的位移值。举个例子，如果日志有 10 条消息，位移值从 0 开始，那么，第 10 条消息的位移值就是 9。此时，LEO = 10。

LSO：Log Stable Offset。这是 Kafka 事务的概念。如果你没有使用到事务，那么这个值不存在（其实也不是不存在，只是设置成一个无意义的值）。该值控制了事务型消费者能够看到的消息范围。它经常与 Log Start Offset，即日志起始位移值相混淆，因为有些人将后者缩写成 LSO，这是不对的。在 Kafka 中，LSO 就是指代 Log Stable Offset。

AR：Assigned Replicas。AR 是主题被创建后，分区创建时被分配的副本集合，副本个数由副本因子决定。

ISR：In-Sync Replicas。Kafka 中特别重要的概念，指代的是 AR 中那些与 Leader 保持同步的副本集合。在 AR 中的副本可能不在 ISR 中，但 Leader 副本天然就包含在 ISR 中。关于 ISR，还有一个常见的面试题目是**如何判断副本是否应该属于 ISR**。目前的判断依据是：**Follower 副本的 LEO 落后 Leader LEO 的时间，是否超过了 Broker 端参数 replica.lag.time.max.ms 值**。如果超过了，副本就会被从 ISR 中移除。

HW：高水位值（High watermark）。这是控制消费者可读取消息范围的重要字段。一个普通消费者只能“看到”Leader 副本上介于 Log Start Offset 和 HW（不含）之间的所有消息。水位以上的消息是对消费者不可见的。关于 HW，问法有很多，我能想到的最高级的问法，就是让你完整地梳理下 Follower 副本拉取 Leader 副本、执行同步机制的详细步骤。这就是我们的第 20 道题的题目，一会儿我会给出答案和解析。

12.Kafka 能手动删除消息吗?

其实，Kafka 不需要用户手动删除消息。它本身提供了留存策略，能够自动删除过期消息。当然，它是支持手动删除消息的。因此，你最好从这两个维度去回答。

对于设置了 Key 且参数 cleanup.policy=compact 的主题而言，我们可以构造一条 <Key，null> 的消息发送给 Broker，依靠 Log Cleaner 组件提供的功能删除掉该 Key 的消息。

对于普通主题而言，我们可以使用 kafka-delete-records 命令，或编写程序调用 Admin.deleteRecords 方法来删除消息。这两种方法殊途同归，底层都是调用 Admin 的 deleteRecords 方法，通过将分区 Log Start Offset 值抬高的方式间接删除消息。

13.__consumer_offsets 是做什么用的?

这是一个内部主题，公开的官网资料很少涉及到。因此，我认为，此题属于面试官炫技一类的题目。你要小心这里的考点：该主题有 3 个重要的知识点，你一定要全部答出来，才会显得对这块知识非常熟悉。

它是一个内部主题，无需手动干预，由 Kafka 自行管理。当然，我们可以创建该主题。

它的主要作用是负责注册消费者以及保存位移值。可能你对保存位移值的功能很熟悉，但其实**该主题也是保存消费者元数据的地方。千万记得把这一点也回答上**。另外，这里的消费者泛指消费者组和独立消费者，而不仅仅是消费者组。

Kafka 的 GroupCoordinator 组件提供对该主题完整的管理功能，包括该主题的创建、写入、读取和 Leader 维护等。

14. 分区 Leader 选举策略有几种?

分区的 Leader 副本选举对用户是完全透明的，它是由 Controller 独立完成的。你需要回答的是，在哪些场景下，需要执行分区 Leader 选举。每一种场景对应于一种选举策略。当前，Kafka 有 4 种分区 Leader 选举策略。

OfflinePartition Leader 选举：每当有分区上线时，就需要执行 Leader 选举。所谓的分区上线，可能是创建了新分区，也可能是之前的下线分区重新上线。这是最常见的分区 Leader 选举场景。

ReassignPartition Leader 选举：当你手动运行 kafka-reassign-partitions 命令，或者是调用 Admin 的 alterPartitionReassignments 方法执行分区副本重分配时，可能触发此类选举。假设原来的 AR 是[1，2，3]，Leader 是 1，当执行副本重分配后，副本集合 AR 被设置成[4，5，6]，显然，Leader 必须要变更，此时会发生 Reassign Partition Leader 选举。

PreferredReplicaPartition Leader 选举：当你手动运行 kafka-preferred-replica-election 命令，或自动触发了 Preferred Leader 选举时，该类策略被激活。所谓的 Preferred Leader，指的是 AR 中的第一个副本。比如 AR 是[3，2，1]，那么，Preferred Leader 就是 3。

ControlledShutdownPartition Leader 选举：当 Broker 正常关闭时，该 Broker 上的所有 Leader 副本都会下线，因此，需要为受影响的分区执行相应的 Leader 选举。

这 4 类选举策略的大致思想是类似的，即从 AR 中挑选首个在 ISR 中的副本，作为新 Leader。当然，个别策略有些微小差异。不过，回答到这种程度，应该足以应付面试官了。毕竟，微小差别对选举 Leader 这件事的影响很小。

15.Kafka 的哪些场景中使用了零拷贝（Zero Copy）？

Zero Copy 是特别容易被问到的高阶题目。在 Kafka 中，体现 Zero Copy 使用场景的地方有两处：**基于 mmap 的索引**和**日志文件读写所用的 TransportLayer**。

先说第一个。索引都是基于 MappedByteBuffer 的，也就是让用户态和内核态共享内核态的数据缓冲区，此时，数据不需要复制到用户态空间。不过，mmap 虽然避免了不必要的拷贝，但不一定就能保证很高的性能。在不同的操作系统下，mmap 的创建和销毁成本可能是不一样的。很高的创建和销毁开销会抵消 Zero Copy 带来的性能优势。由于这种不确定性，在 Kafka 中，只有索引应用了 mmap，最核心的日志并未使用 mmap 机制。

再说第二个。TransportLayer 是 Kafka 传输层的接口。它的某个实现类使用了 FileChannel 的 transferTo 方法。该方法底层使用 sendfile 实现了 Zero Copy。对 Kafka 而言，如果 I/O 通道使用普通的 PLAINTEXT，那么，Kafka 就可以利用 Zero Copy 特性，直接将页缓存中的数据发送到网卡的 Buffer 中，避免中间的多次拷贝。相反，如果 I/O 通道启用了 SSL，那么，Kafka 便无法利用 Zero Copy 特性了。

深度思考题

16.Kafka 为什么不支持读写分离?

这道题目考察的是你对 Leader/Follower 模型的思考。

Leader/Follower 模型并没有规定 Follower 副本不可以对外提供读服务。很多框架都是允许这么做的，只是 Kafka 最初为了避免不一致性的问题，而采用了让 Leader 统一提供服务的方式。

不过，在开始回答这道题时，你可以率先亮出观点：**自 Kafka 2.4 之后，Kafka 提供了有限度的读写分离，也就是说，Follower 副本能够对外提供读服务。**

说完这些之后，你可以再给出之前的版本不支持读写分离的理由。

场景不适用。读写分离适用于那种读负载很大，而写操作相对不频繁的场景，可 Kafka 不属于这样的场景。

同步机制。Kafka 采用 PULL 方式实现 Follower 的同步，因此，Follower 与 Leader 存在不一致性窗口。如果允许读 Follower 副本，就势必要处理消息滞后（Lagging）的问题。

17. 如何调优 Kafka?

回答任何调优问题的第一步，就是**确定优化目标，并且定量给出目标**！这点特别重要。对于 Kafka 而言，常见的优化目标是吞吐量、延时、持久性和可用性。每一个方向的优化思路都是不同的，甚至是相反的。

确定了目标之后，还要明确优化的维度。有些调优属于通用的优化思路，比如对操作系统、JVM 等的优化；有些则是有针对性的，比如要优化 Kafka 的 TPS。我们需要从 3 个方向去考虑。

Producer 端：增加 batch.size、linger.ms，启用压缩，关闭重试等。

Broker 端：增加 num.replica.fetchers，提升 Follower 同步 TPS，避免 Broker Full GC 等。

Consumer：增加 fetch.min.bytes 等

18.Controller 发生网络分区（Network Partitioning）时，Kafka 会怎么样?

这道题目能够诱发我们对分布式系统设计、CAP 理论、一致性等多方面的思考。不过，针对故障定位和分析的这类问题，我建议你首先言明“实用至上”的观点，即不论怎么进行理

论分析，永远都要以实际结果为准。一旦发生 Controller 网络分区，那么，第一要务就是查看集群是否出现“脑裂”，即同时出现两个甚至是多个 Controller 组件。这可以根据 Broker 端监控指标 ActiveControllerCount 来判断。

现在，我们分析下，一旦出现这种情况，Kafka 会怎么样。

由于 Controller 会给 Broker 发送 3 类请求，即 LeaderAndIsrRequest、StopReplicaRequest 和 UpdateMetadataRequest，因此，一旦出现网络分区，这些请求将不能顺利到达 Broker 端。这将影响主题的创建、修改、删除操作的信息同步，表现为集群仿佛僵住了一样，无法感知到后面的所有操作。因此，网络分区通常都是非常严重的问题，要赶快修复。

19.Java Consumer 为什么采用单线程来获取消息?

在回答之前，如果先把这句话说出来，一定会加分：**Java Consumer 是双线程的设计。一个线程是用户主线程，负责获取消息；另一个线程是心跳线程，负责向 Kafka 汇报消费者存活情况。将心跳单独放入专属的线程，能够有效地规避因消息处理速度慢而被视为下线的“假死”情况。**

单线程获取消息的设计能够避免阻塞式的消息获取方式。单线程轮询方式容易实现异步非阻塞式，这样便于将消费者扩展成支持实时流处理的操作算子。因为很多实时流处理操作算子都不能是阻塞式的。另外一个可能的好处是，可以简化代码的开发。多线程交互的代码是非常容易出错的。

20. 简述 Follower 副本消息同步的完整流程

首先，Follower 发送 FETCH 请求给 Leader。接着，Leader 会读取底层日志文件中的消息数据，再更新它内存中的 Follower 副本的 LEO 值，更新为 FETCH 请求中的 fetchOffset 值。最后，尝试更新分区高水位值。Follower 接收到 FETCH 响应之后，会把消息写入到底层日志，接着更新 LEO 和 HW 值。

Leader 和 Follower 的 HW 值更新时机是不同的，Follower 的 HW 更新永远落后于 Leader 的 HW。这种时间上的错配是造成各种不一致的原因。

好了，今天的面试题分享就先到这里啦。你有遇到过什么经典的面试题吗？或者是有什么好的面试经验吗？

欢迎你在留言区分享。如果你觉得今天的内容对你有所帮助，也欢迎分享给你的朋友。

上一篇　特别放送（三）| 我是怎么度过日常一天的?

精选留言 (2)

写留言

镜子中间
2020-06-11

胡大大给力！为了我们面试不被虐真是用心良苦！

展开

作者回复: 嗯嗯，加油加油!

Satellite

2020-06-15

胡老师你好，看到有设置单条消息大小的题，想到个问题 大小应该也是有个限制的吧，例如消息都是10M以上的，这种合理么 会不会对集群压力很大（broker异常 复制副本等）

作者回复: 确实压力会很大，通常也不是很建议传输这么大的消息。