面试随记（kafka

- kafka设计：Kafka 将消息以 topic 为单位进行归纳，发布消息的程序称为 **Producer**，消费消息的程序称为 **Consumer**。它是以集群的方式运行，可以由一个或多个服务组成，每个服务叫做一个 **Broker**，Producer 通过网络将消息发送到 kafka 集群，集群向消费者提供消息，broker 在中间起到一个代理保存消息的中转站。
- **Kafka 中重要的组件**
  - *Producer*： 消息生产者
  - *Broker*：一个 Kafka 节点就是一个 Broker，多个Broker可组成一个Kafka 集群
    - 某个 Topic 下有 n 个Partition 且集群有 n 个Broker，那么每个 Broker会存储该 Topic 下的一个 Partition
    - 某个 Topic 下有 n 个Partition 且集群中有 m+n 个Broker，那么只有 n 个Broker 会存储该Topic下的一个 Partition
    - 某个Topic 下有 n 个Partition 且集群中的Broker数量小于 n，那么一个 Broker 会存储该 Topic 下的一个或多个 Partition，这种情况尽量避免，会导致集群数据不均匀
  - *Topic*：消息主题，每条发布到Kafka集群的消息都会归集于此，Kafka是面向Topic 的
  - *Partition*：Partition 是Topic在物理上的分区，一个Topic可以分为多个Partition，每个Partition是一个有序的不可变的记录序列。单一主题中的分区有序，但无法保证主题中所有分区的消息有
  - *Consumer*：Kafka集群中的消费者
  - *Consumer Group*：每个Consumer都属于一个Consumer Group，每条消息只能被Consumer Group中的一个Consumer消费，但可以被多个Consumer Group消费
  - *Replica*：Partition 的副本，用来保障Partition的高可用性
  - *Controller*： Kafka 集群中的其中一个服务器，用来进行Leader election以及各种Failover 操作。
  - *Zookeeper*：Kafka 通过Zookeeper来存储集群中的 meta 消息
- kafka为什么高性能
  - pagecache缓存
  - 磁盘顺序写
  - 零拷贝
  - pull的消费模式，与之对应的push模式难以处理不同速率的上游推送消息。pull模式消费者可以自主决定是否从broker中拉取数据，此外为了避免在没有数据的情况下消费者不断轮询，Kafka可以通过参数使消费者阻塞直到新消息到达
- kafka文件高效存储设计
  - 单个partition大文件分为多个小文件段，通过小文件段就容易定期清理已经消费完的消息，减少磁盘占用
- kafka优点
  - 高性能，吞吐量大，时延低
  - 高可用
  - 容错性高
  - 高扩展
- kafka应用场景
  - 日志聚合
  - 消息系统
  - 系统解耦
  - 流量削峰

    - 异步处理
- kafka中的分区
    - 主题是逻辑上的概念
    - 分区是物理上的概念
    - 一个主题可以有多个分区，一个分区只属于一个主题
    - 同一主题下的不同分区包含的消息是不同的，分区在存储层面可以看做一个可追加的日志文件，消息在被追加到分区日志文件的时候都会分配一个特定的偏移量（offset）。offset 是消息在分区中的唯一标识，kafka 通过它来保证消息在分区内的顺序性，不过 offset 并不跨越分区，也就是说，kafka保证的是分区有序而不是主题有序。
    - 为了容错性，一个分区可以有多个副本，一主多从。主副本负责读写，从副本只负责同步。同一分区的不同保存的消息一致，副本一般存储在不同的broker中
    - 分区可以方便在集群中扩展，提高并发，因为以分区为单位进行读写
- kafka中的消息
    - 分好区的消息不会直接发送到服务端，而是放入生产者的缓冲区，多条消息会被封装为一个批次（batch），默认大小为16kb
    - 发送线程启动后会从缓冲中获取可发送的批次，并发送到服务端
    - batch的大小可以根据下面三个维度进行发送
        - 累计的消息数量
        - 累计的时间间隔
        - 累计的数据大小
- kafka的负载均衡和故障转移
    - 负载均衡，每个broker都有均等机会成为客户端提供服务，可以负载均衡分布到集群中的机器上
    - 故障转移，通过会话机制实现，kafka启动后以会话的形式注册到zookeeper，一旦服务器出现问题，与zookeeper的会话就会断开，此时kafka集群就会选取另外一台服务器来继续提供服务
- zookeeper的作用
    - kafka使用zookeeper构建分布式系统，broker启动会在zookeeper上注册，或者说zookeeper维护了整个集群的状态，比如谁提供服务，故障转移恢复等
- kafka中消费者和消费者组的关系及负载均衡实现
    - 消费者组是kafka独有的可扩展具有容错性的消费者机制，一个消费者组有多个消费者，共享一个全局唯一的组id，祖内所有消费者协调消费订阅主题的所有分区，当然一个分区只能由同一个消费者组的一个消费者消费，消费者数量通常不超过分区数量
    - 消费者和消费者组的关系会被不断协调，比如增加新的消费者，为其分配分区
- Kafka中offset作用
    - 唯一区分分区中的每条消息，存储文件命名也是按照offset.Kafka命名
- queuefullexpection
    - 当生产者试图发送消息的速度快于broker处理的速度就会发生上述问题
    - 如何解决：首先判断生产者是否能够降低速率，如果不能，则需要添加足够的broker，或者生产阻塞
- replica，leader，follower
    - 为了保证分区的高可用性，采用备份机制，将相同的数据复制到多个broker中，这些备份就是replica，replica一般均匀分配到所有broker中
    - leader，负责提供读写的replica
    - follower，不对外提供服务，只向leader请求同步
- 如何判断一个broker是否有效

    - broker会在zookeeper上注册，zookeeper定期通过心跳机制检查每个节点的连接
    - 如果broker是个follower，必须定期同步与leader的写操作
- kafka的ack机制
    - 0，不需要leader broker回复，发送完就认为成功，最低延迟，但持久性可靠性差，broker发生故障时，很可能发生数据丢失
    - 1，默认配置，需要leader确认成功才继续发送，不过leader宕机，而follower尚未复制的情况下，消息就会丢失，拥有较好的持久性和延迟性
    - -1，leader不仅需要回复，并且只有在与所有follower都同步确认后才回复，可靠性最好，但延时性最差
- 消费者如何消费
    - 生产者将数据推送至broker，消费者从broker中主动去pull消息进行消费，由消费者自己去控制消费的进度和数量
- Kafka的日志保留期和数据清理策略
    - 保留期保留了Kafka集群所有已发布消息，超过保留的数据将按清理策略进行清理
    - 默认清理时间是七天，清理策略有删除和压缩
- kafka主从同步
    - 通过维护一个同步状态的副本集合，集合中的所有节点都是和leader高度一致，消息只有被每个节点读取并追加到日志中，才会被认为已提交
    - 主从同步分为同步复制和异步复制
- kafka什么情况会出现消息丢失/不一致
    - 由于kafka有ack机制，ack机制为0即异步的，不需要leader回复，leader挂了则消息丢失；为1时，假如leader挂了，follower没来得及同步，造成数据丢失
    - 消费时，也可能造成消息不同步
- kafka如何保证顺序消费
    - kafka消费单元是分区，分区中通过offset保证同一个分区中的消息的唯一有序，如果要保证topic中的消费顺序可以使所有消息都发往同一个分区