分布式连环炮

分布式Session方案?

spring session

利用独立部署的 Session 服务器统一管理
Session，服务器每次读写 Session 时，都访问 Session 服务器。
对于 Session 服务器，我们可以使用 Redis 或者 MongoDB 等内存数据库来保存 Session
中的数据，以此替换掉服务中的 HttpSession。达到 Session 共享的效果。

设计一个高并发系统?

服务拆分，一个服务拆分为多个服务，即微服务架构

缓存，高并发场景，大多读多写少，Redis进行缓存

消息队列，高并发的写场景，用MQ进行削峰和异步

分库分表，mycat，一个数据库拆分多个库，用多个库来抗更高的并发，将一个表拆分多个表。

读写分离

mysql的读写分离，Redis的读写分离

ES，分布式，可扩容，搜索类的操作可以使用ES承载

倒排索引是什么?

将文档id建立为索引，通过id快速可以快速查找数据。如数据库中的主键就会创建正排索引。

非结构化数据中我们往往会根据关键词查询数据。此时我们将数据中的关键词建立为索引，指向文档数据，这样的索引称为倒排索引。

如何保证消息传递不丢失?

生产者丢失数据

生产者发送给rabbitmq半路丢数据

RabbitMQ事务机制是同步的，所以吞吐量会降低，不可取

confirm模式

发送完消息不用管，如果出错回调本地出错接口

rabbimq丢失数据

开启RabbitMQ的持久化，同时持久化queue和queue中的数据

消费者丢失数据

将AutoAck关闭，每次确定处理完一条消息后，再发送ack给RabbitMQ

如何设计一个消息中间件架构?

首先MQ得支持可伸缩性，那就需要快速扩容，就可以增加吞吐量和容量，可以设计一个分布式的系统，参考kafka的设计理念，broker - > topic -> partition，每个partition放一台机器，那就存一部分数据，如果现在资源不够了，可以给topic增加partition，然后做数据迁移，增加机器，不就可以存放更多的数据，提高更高的吞吐量。

其次得考虑一下这个MQ的数据要不要落地磁盘？也就是需不需要保证消息持久化，因为这样可以保证数据的不丢失，那落地盘的时候怎么落？顺序写，这样没有磁盘随机读写的寻址开销，磁盘顺序读的性能是很高的，这就是kafka的思路。

其次需要考虑MQ的可用性？这个可以具体到我们上面提到的消息队列保证高可用，提出了多副本 ，leader 和follower模式，当一个leader宕机的时候，马上选取一个follower作为新的leader对外提供服务。

需不需要支持数据0丢失?

消息积压问题?

临时建立好原先10倍或者20倍的queue数量

积压大量消息

然后写一个临时的分发数据的consumer程序，这个程序部署上去消费积压的数据，消费之后不做耗时的处理，直接均匀轮询写入临时建立好的10倍数量的queue

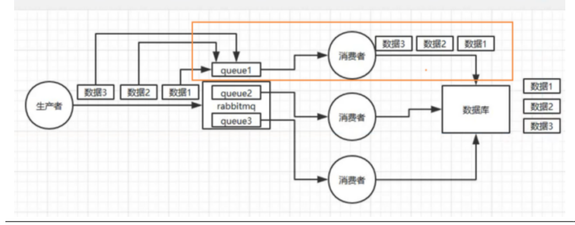
接着临时征用10倍机器来部署consumer，每一批consumer消费一个临时queue的数据

这种做法相当于临时将queue资源和consumer资源扩大了10倍，以正常的10倍速度

如何保证消息的顺序性?

搞多个Queue，每个消费者就消费其中的一个Queue。把需要保证顺序的数据发到1个Queue里去。

RabbitMQ保证消息顺序性



如何保证消息的重复消费？如何保证消息消费的幂等性?

幂等性就是一个数据，或者一个请求，给你执行多次，得保证对应的数据不会改变，并且不能出错，这就是幂等性。

比如那个数据要写库，首先根据主键查一下，如果这个数据已经有了，那就别插入了，执行update即可

如果用的是redis，那就没问题了，因为每次都是set操作，天然的幂等性

生产者发送每条消息的时候，需要加一个全局唯一的id，类似于订单id之后的东西，然后你这里消费到了之后，先根据这个id去redis中查找，之前消费过了么，如果没有消费过，那就进行处理，然后把这个id写入到redis中，如果消费过了，那就别处理了，保证别重复消费相同的消息即可。

你的项目中怎么使用消息队列的?

大广告新增、删除、修改时异步删除缓存，延时双删，先删除缓存，更新数据库，sleep500ms再删除缓存

商品的状态，上架新增商品异步放入redis和solr，删除或下架商品，商品从redis和solr中删除。购物车依赖商品缓存信息。

创建订单，同步消息进行库存判断，更新库存，生成订单信息，同步消息返回订单id。之后异步删除购物车（HttpClient请求），异步发送邮件给用户

为什么使用消息中间件?

就是一个系统或者一个模块，调用了多个系统，互相之间的调用很复杂，维护起来很麻烦。但是其实这个调用是不需要同步调用接口的，如果用MQ给他异步化解耦，也是可以的，这个时候可以考虑在自己的项目中，是不是可以运用这个MQ来进行系统的解耦。

大量的请求过来，然后MQ将其消化掉了，然后通过其它系统从MQ中取消息，在逐步进行消费，保证系统的有序运行。

消息中间件优缺点?

优点：异步解耦削峰

缺点：可用性降低，MQ挂了系统崩溃了，还有就是一致性问题。

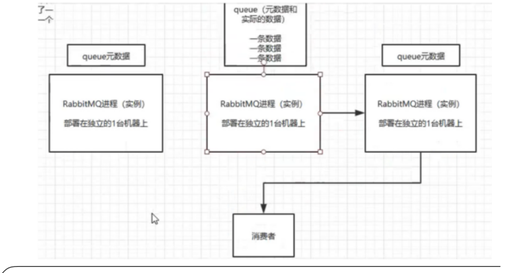
主流MQ?

kafka	吞吐量高（10W），大数据实时分析用的多，kafka是分布式的MQ
ActiveMQ	万级吞吐，主从架构实现高可用，但是偶尔消息丢失
RabbitMQ	万级吞吐，延迟最低，消息不丢失，erlang开发，并发能力强，性能非常好
RocketMQ	可用性高，社区维护方便，不像Rabbitmq限于erlang语言

如何保证消息队列高可用?

项目中rabbitmq的高可用性

单机模式



普通集群模式

多台机器上启动多个RabbitMQ实例，每台机器启动一个，但是创建的Queue，只会放在一个RabbitMQ实例上，但是每个实例都同步queue元数据

集群镜像模式

创建的queue无论元数据还是queue里的消息都会存在与多个实例中，然后每次你写消息到queu的时候，都会自动把消息推送到多个实例的queue中进行消息同步。

你任何一个机器宕机了，别的机器都可以用。坏处在于，性能开销提升，消息同步所有的机器，导致网络带宽压力和消耗增加，第二就是没有什么扩展性

kafka的高可用?

天然的分布式消息队列

多个broker组件，每个broker是一个节点，你创建一个topic，这个topic可以划分成多个partition，每个partition可以存在于不同的broker上，每个partition就放一部分数据。副本使用主从的方式leader和Follower，只有leader对外提供读写