中间件题目

一、概念解释

1、什么是中间件（狭义、广义）？

中间件(Middleware)是一种**软件**，处于**系统软件**（操作系统和网络软件）与**应用软件**之间，它能使应用软件之间进行跨网络的**协同工作**（也就是互操作）。

狭义中间件如常见的远程服务框架、消息队列、缓存等；广义的中间件是操作系统之上业务逻辑之下的所有可复用的后台组件。

2、英文名词解释

(1)云计算3个as?

IaaS基础设施即服务：向客户提供**处理、存储、网络**以及**其他基础计算资源**
PaaS平台即服务：把**应用程序的开发和运行环境**作为一种服务提供给用户
SaaS软件即服务：服务的提供者将软件应用部署在服务器上，通过互联网分发给最终用户。

(2)RPC

远程过程调用协议(Remote Procedure Call Protocol)，一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。

(3)IDL

接口定义语言IDL(Interface Definition Language)，是用来描述**软件组件接口**的一种计算机语言。定义了软件组件之间的接口规范，包括接口的方法、参数、返回类型等信息。

(4)ORB

对象请求代理（ORB，Object Request Broker）是对象之间建立客户端/服务端（Client/Server）关系的中间件。使用ORB，客户可以透明地调用一个服务对象上的方法，这个服务对象可以在本地，也可以在通过网络连接的其他机器上。

是一种中间件技术，用于分布式对象通信。它允许**分布式应用程序中的对象**进行通信，而无需了解彼此的实现细节。

(5)白页黄页绿页

命名(Naming)服务（白页）：通过**外部名字**定位构件
维护了（分布或集中式）系统中资源的名字与地址之间的映射关系（binding）

目录(Directory)服务（黄页）：通过**服务特性**定位构件
维护了（分布或集中式）系统中资源的名字与及其多种信息的映射关系

合约(Contract)服务（绿页）：通过**技术规范**定位构件
维护了企业以及企业提供的服务及其技术规范

(6)两段锁

两阶段封锁(Two-Phase Locking)两段锁协议的内容：
1.在对任何数据进行读、写操作之前，事务首先要获得对该数据的所有封锁，访问完后释放所有锁。
2.在释放一个封锁之后，事务不再获得任何其他封锁。

“两段”锁的含义：
事务分为两个阶段：
第一阶段是获得封锁，也称为扩展阶段（Growing Phase）；
第二阶段是释放封锁，也称为收缩阶段（Shrinking Phase）。

3、Web Service

Web服务（Web Service）是一种**基于网络的计算技术**，用于在不同应用程序之间进行**互操作和数据交换**。它通过标准的**网络协议**（如HTTP、HTTPS）和标准的**数据格式**（如XML、JSON）来实现**跨平台**、**跨语言**的通信。

4、微服务

微服务（或称微服务架构）是一种**云原生架构**方法，在单个应用中包含众多**松散耦合且可单独部署**的小型组件或服务。每个服务都是一个独立的组件，**能够独立开发、部署和扩展**。

5、负载均衡（什么是负载什么是均衡）？

负载（Load）是指在计算机系统中，对于**某种资源**（如CPU、内存、磁盘等）的**使用程度或者压力**的度量。
负载均衡（Load Balancing）是一种分布式系统设计技术，用于将**工作负载**（即任务或请求）均匀地分摊到**多个处理单元**（如服务器、计算节点等）上，以提高系统的**性能、可伸缩性和可用性**。

6、什么是容器什么是VM，区别与联系?

容器是一种轻量级的虚拟化技术，它将**应用程序及其所有依赖项**打包在一起，包括代码、运行时环境、系统工具、库和设置。

虚拟机是一种软件仿真的计算机系统，它在物理计算机上创建了一个**独立的、完整的操作系统环境**。

区别：
容器提供进程级别的隔离，而虚拟机提供操作系统级别的隔离。
由于容器共享操作系统内核，因此它们更加轻量级，启动更快，并且占用更少的资源。
容器的部署速度更快。
容器镜像可以在不同的平台和环境中轻松迁移。

联系：
都提供了一种在单个物理计算机上运行多个应用程序或服务的方式。

二、简答题

1、什么是高内聚低耦合，说明如何使用消息型中间件来解耦?

内聚指的是模块内部各个元素之间的关联程度。高内聚意味着一个模块内的功能高度相关且集中。

耦合指的是模块之间的依赖程度。低耦合意味着模块之间的依赖关系较少。

如何解耦：模块之间使用MQ进行通信。

2、计算架构，从单机向云计算演进，其商业逻辑?

3、什么是对象池技术，什么是云原生，举例其优点?

对象池技术预先创建一组对象并将它们存储在一个池中。需要时可以复用这些对象，而不需要每次都创建新的对象。

优点：减少对象的创建和销毁频率、避免了重复分配内存和资源的开销、加快系统响应速度。

云原生（Cloud Native）是一种设计、构建和运行应用程序的方法，充分利用云环境提供的弹性、高可用性和按需资源分配等特性，从而实现快速迭代和高效交付。

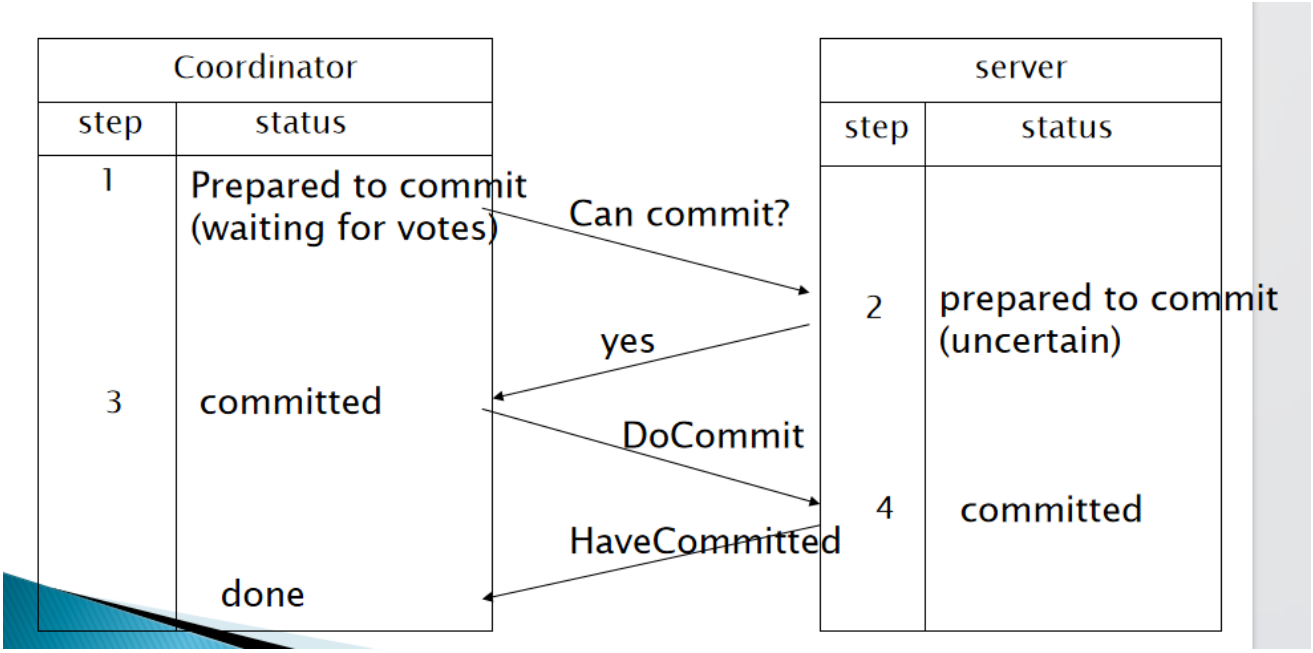
优点：弹性和伸缩性、高可用性和容错性、快速部署和更新、自动化运维。

4、画图说明两段式提交的过程，列举三段式及其特点?

两阶段:

事务发起阶段:
事务的发起者提出一个request（比如用户下单购买某个商品），要求其依赖的服务（事务的执行者）本地执行业务逻辑。执行成功本地事务不提交但要告诉发起者本地已经执行成功；执行失败执行者告诉发起者本地作业执行失败

事务提交/回滚阶段:
事务发起者根据事务发起阶段收集到的信息决定提交/回滚
如果全部执行者都反馈成功那么发起者通知所有执行者提交事务
如果存在执行者反馈失败，则发起者通知所有参与者取消事务



三阶段:
CanCommit(询问阶段):
事务协调者向参与者发送事务执行请求,询问是否可以完成指令,参与者只需要回答是不是即可,不需要做正真的事务操作,这个阶段会有超时终止机制。
PreCommit(准备阶段):
事务协调者会根据参与者的反馈结果决定是否继续执行,如果在询问阶段所有参与者都返回可以执行操作,则事务协调者会向所有参与者发送PreCommit请求,参与者收到请求后会写redo和undo日志,执行事务操作但是不提交事务,然后返回ACK响应等待事务协调者的下一步通知。如果询问阶段任意参与者返回不能执行操作的结果,那么事务协调者会向所有参与者发送事务中断请求。
DoCommit(提交或回滚阶段):
这个阶段也会存在两种结果，根据上一步骤的执行结果来决定DoCommit的执行方式。如果每个参与者在PreCommit阶段都返回成功,那么事务协调者会向所有的参与者发起事务提交指令。反之,如果参与者中的任一个参与者返回失败,那么事务协调者就会发起中转指令来回滚事务。

5、单机应用的架构和微服务架构的特点?

单机：开发和部署集中、数据库共享、性能好、架构简单、难以水平拓展。

微服务：服务边界清晰、自治、分布式数据管理、灵活的可拓展性、技术异构。

6、DNS进行负载均衡，有什么优缺点?

优点：简单（只需配置DNS记录）、成本低（无需购买服务器）、可以将用户的请求引导到最近的服务器。

缺点：

用户直接访问的DNS服务器常常没有所需的DNS信息，而需将DNS请求转发给保存着该信息的DNS服务器，并且缓存返回的域名与IP地址的映射，实际效果可能并不理想。

当一个新的服务节点加入时，DNS信息不能很快更新。

当服务节点崩溃时，DNS服务器也无法感知。

7、解释对象的序列化和反序列化?

序列化是将对象的状态转换为字节流的过程。

反序列化是将字节流恢复为原始对象的过程。

8、什么是控制反转，什么是依赖注入，说明其作用及其使用场景?

控制反转是一种设计原则，其核心思想是将对象的**创建和管理**职责从**应用程序代码**中抽离出来，交给一个**容器或框架**处理。

依赖注入是实现控制反转的一种具体方式。它通过**将依赖关系（即对象依赖的其他对象）注入到该对象**中，而不是让对象自行创建依赖。

作用：降低代码之间的耦合度、便于单元测试、更容易修改和扩展系统

使用场景：大型企业级应用程序、需要高可维护性和可扩展性的系统、复杂的依赖关系图谱

9、什么是高内聚低耦合

（前面有）

10、VM实现虚拟化的两种方式?

全虚拟化：允许未修改的客体机在虚拟监控器上运行。

半虚拟化：要求客体机系统修改，以便与虚拟机监控器合作，提高性能。

三、综合题

1、简要介绍消息型中间件，什么是Queue，什么是Topic?

消息型中间件（Message-Oriented Middleware, MOM）是一种用于**分布式系统**中的基础软件，旨在通过**消息传递机制**实现**系统之间的通信和数据交换**。

Queue（队列）模式：点对点的消息传递模型，消息生产者将消息发送到队列中，消息消费者从队列中读取并处理消息。

Topic（主题）模式：发布/订阅的消息传递模型，消息生产者将消息发布到主题中，多个消息消费者可以订阅同一个主题，并且每个订阅者都会收到所有发布到该主题的消息。

2、举例说明如何利用消息型中间件解耦、异步、削峰?

解耦：模块与模块之间使用MQ进行通信。

异步：MQ提供了异步处理机制，允许你把一个消息放入队列，但并不立即处理它。从而实现消息异步处理。

削峰：用户的请求，服务器接收后，首先写入消息队列。假如消息队列长度超过最大数量，则直接抛弃用户请求或跳转到错误页面。秒杀业务根据消息队列中的请求信息，再做后续处理。

3、技术和商业角度阐述微服务的意义？

技术：模块化松耦合使开发简单而集中、每个模块可以独立部署和拓展、使用的技术灵活。

商业：快速响应市场变化、提高开发效率、降低长期维护成本。

4、（查资料）什么是web服务，是无状态的还是有状态的，rest型的web服务可以有状态吗？

web服务是无状态的，在web应用中，经常会使用Session来维持登陆的上下文信息，虽然协议无状态，但借助Session，可以使http服务转换为有状态服务。

5、（查资料）技术商业角度阐述k8s产生和成为主流的原因，解释其与docker的关系？

Docker和K8S本质上都是创建容器的工具，Docker作用于单机，K8S作用于集群。

Docker是一种开源的容器化平台，它使开发人员能够将应用程序及其所有依赖项打包到一个独立的容器中，从而实现轻量级、可移植和可扩展的部署。

Kubernetes是一个用于**自动化容器化应用程序部署、扩展和管理**的开源平台。虽然Kubernetes可以与各种容器运行时配合使用，但它最常与Docker容器一起使用，因为Docker在容器技术中占据主导地位。

Kubernetes通过提供诸如自动伸缩、负载均衡、服务发现等功能，使得在生产环境中部署和管理容器化应用程序变得更加容易和可靠。

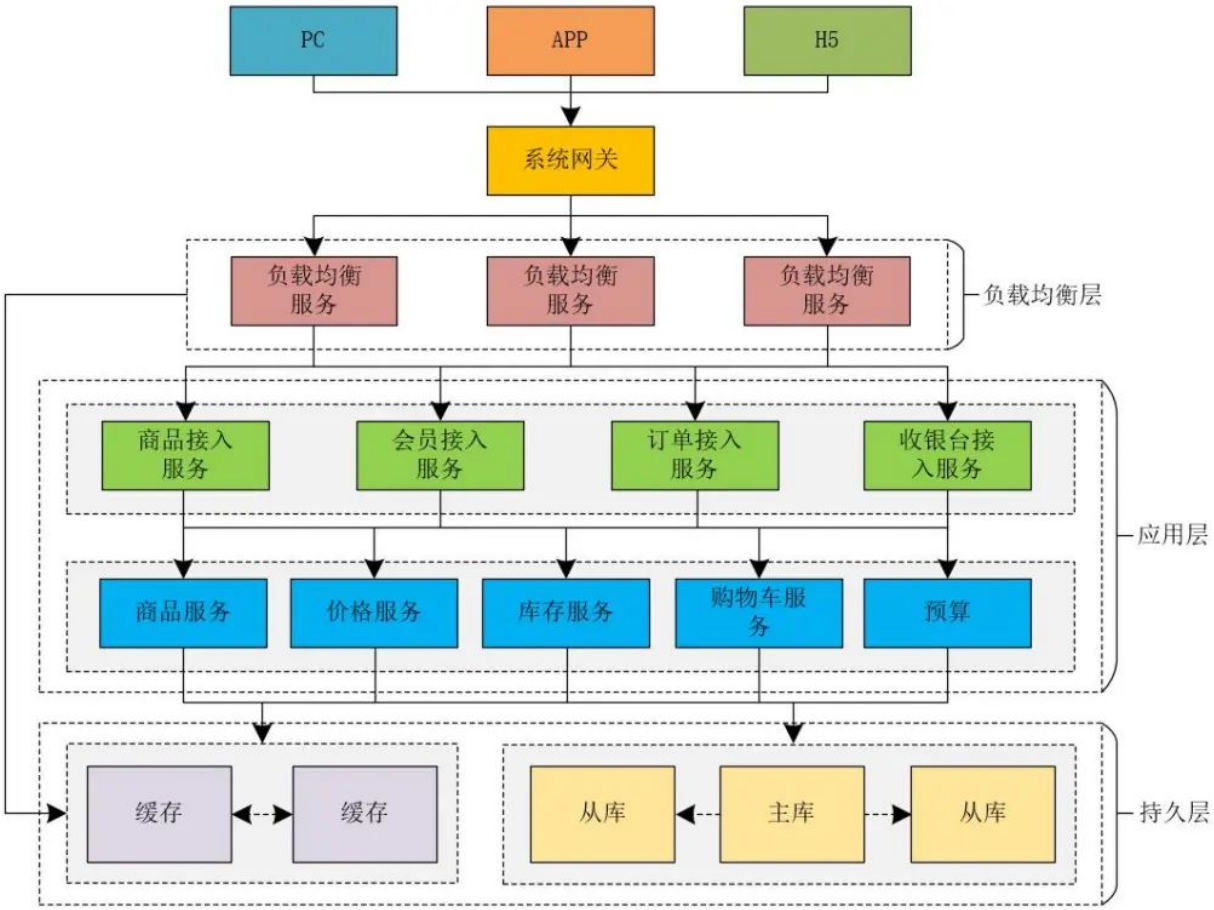
随着时代的发展，对系统的性能有了更高的要求，高可用、高并发都是基本要求。随着要求变高的的同时，单机显然性能就跟不上了，服务器集群管理就是发展趋势，所以 Kubernetes 为代表的云原生技术强势发展并成为主流。

四、设计题（二选一）

1、电子商务秒杀

2、大规模订单

(1)设计该系统结构，画出软件架构图，解释其主要特征。



(2)如果是基于微服务进行架构，系统中哪些功能应该设计为微服务架构，举三个例子。

秒杀活动管理服务、订单处理服务、库存管理服务

(3)从中间件的角度，可以从哪个角度出发解决大规模请求需求，举三个例子。（域名解析、数据库请求分为长短请求，微服务层次）

负载均衡（Nginx进行反向代理将请求分发到不同的后端服务实例）、缓存数据库的使用（将频繁访问的数据缓存在Redis中）、消息队列异步处理请求（RabbitMQ、Kafka）