中间件复习

题型：概念题20’ 5个，简答题40 8道‘，综合题20’ 2个，设计题20‘ 一个

1. 什么是中间件，什么是分布式互操作

答：中间件是处于系统软件（操作系统和网络软件）和应用软件之间的**一种起连接作用的分布式软件**，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**）。

分布式互操作：★

在分布式软件系统中，有许多独立的、网络连接的、通讯的，并且物理上分离的计算节点，有着通用的数据结构和传输标注设置，使之可以互换数据和执行命令的解决方案。软件的互操作往往通过标准和通用接口，或者通过专门的适配系统实现两种异构系统之间的互操作。

1. orb总线★

   答：Object Request Broker，对象请求代理（对象总线），是一个中间件。它在对象间建立C/S的关系。通过ORB，**客户可以透明的调用同一台机器上或者网络上的一个服务对象的方法**。ORB截获这一调用，同时负责查找实现服务的对象并向其传递参数、调用方法并返回最终结果。

   它为用户提供与其他分布式网络环境中对象通信的接口。

2. 负载均衡

   答：负载均衡是指将工作任务、访问请求等负载进行平衡，分摊到多个服务器和组件等操作元上进行执行，是解决高性能，单点故障，提高可用性和可扩展性，进行水平伸缩的终极解决方案。

3. iaas paas saas三种**云计算服务** 以及caas(container as a service), faas(function as a service)★

   答：IaaS（Infrastructure as a Service，基础设施即服务）、PaaS（Platform as a Service，平台即服务）以及SaaS（Software as a Service，软件即服务）

   IaaS（基础设施即服务）——云计算直接把所有云计算机基础设施（基本计算、存储、处理、网络）作为一种服务，提供给用户，用户能够在基础设施上部署和运行各种系统软件（操作系统）和应用软件。

   PaaS（平台即服务）——提供用户将云端基础设施部署与创建至客户端，或者借此获得使 用编程语言、程序库与服务。**用户不需要管理与控制云端基础设施，但需要控制上层的应用程序部署与 应用托管的环境。**

   SaaS（软件即服务）云服务提供商把 IT 系统的应用软件层作为服务出租出去，而消费者可以使用任何云终端设备接入计算机网络，然后通过网页浏览器或者编程接口使用云端的软件。

   CaaS (通讯即服务)是将传统电信的能力如消息、语音、视频、会议、通信协同等封装成API或者SDK通过 互联网对外开放，提供给第三方使用，将电信能力真正作为服务对外提供。

   FaaS (功能即服务) 是一种云计算服务，它允许开发人员以功能的形式来构建、计算、运行和管理这些应用包，无需维护自己的基础架构。

4. 什么是容器 ;为什么需要容器;容器与vm的关系???★

   容器：为特定组件提供服务的运行时环境，比如JAVA虚拟机，封装底层J2EEAPI服务：

   容器可以管理对象的生命周期、对象与对象之 间的依赖关系。 ➢ 通过容器的配置（如XML配置文件），可以定 义对象的名称、产生方式、对象间的关联关系 等。 ➢ 在启动容器之后，容器自动的生成这些对象， 而无需用户编码来产生对象，或建立对象与对 象之间的依赖关系。

   虚拟机：虚拟整个计算机，包括OS和磁盘，一台物理计算机上模拟多台计算机运行任务

   容器：在OS之上的虚拟，容器共享OS

5. 微服务架构的概念，主要纵向

   答：微服务是一种独立测试、独立部署、独立运行的软件架构模式。每个服务运行在其独立的进程中，服务与服务之间采用轻量级的通信机制沟通。每个微服务之间独立部署。

   特点：小且专注于做一件事；运行在独立的进程中、轻量级的通信机制、松耦合，独立部署。

6. SOA的概念，主要横向

   答：面向服务的体系结构是一个组件模型。它通过定义良好的接口和契约，将应用程序的不同功能单元（称为服务）联系起来，可以将松散耦合的、粗粒度的应用组件根据需求进行分布式的部署、组合和使用。

7. 组件的概念，面向组件编程的概念

   答：组件就是数据和方法的封装对象，是系统中一种物理的、可代替的部件、它封装了实现并提供了一系列可用的接口。

   面向组件编程是一种软件工程实践。设计时通常要求组件之间高内聚、低耦合，着眼于独立工作的可替换的代码模块，以复用为基础，将低耦合的独立部件组合为一个系统。并且无须非常熟悉其内部工作原理。

8. RPC　IDL　IIOP　GIOP ★

   RPC(Remote Procedure Call)：远程过程调用，是分布式计算中的一个计算机通信协议，该协议允许运行于一台计算机的程序调用另一个地址空间（通常为一个开放网络的一台计算机）的子程序，而程序员就像调用本地程序一样，无需关注细节。

   IDL(Interface Definition Language)：是用于描述分布式对象接口的定义语言，利用IDL进行接口定义之后，就确定了客户端与服务器之间的接口，这样即使客户端和 服务器 独立进行开发，也能够正确地 定义和调用所需要的分布式方法。 **定义CORBA接口**

   GIOP(General Inter-ORB Protocol)：通用对象请求代理间通信协议，是分布式计算领域的一种抽象 协议，对象请求代理（ORB）通过该协议进行通信。 **规定了ORB之间的文法和传输格式**

   IIOP(Internet Inter-ORB Protocol),互联网内部对象请求代理协议，它是一个用于CORBA 2.0及兼容平台上的协议。用来在CORBA对象请求代理之间交流的协议。Java中使得程序可以和其他语言的CORBA实现实现互操作性的协议。 **支持两阶段提交 如何在TCP/IP网络交换GIOP信息**

9. 两段锁

   两端锁协议：所有的事务必须分两个阶段对数据项加锁和解锁。即**事务分两个阶段**，第一个阶段是获得锁。事务可以获得任何数据项上的任何类型的锁，但是不能释放；第二阶段是释放锁，事务可以释放任何数据项上的任何类型的锁，但不能申请。

   获得锁的阶段称为扩展阶段，在这个阶段可以进行加锁操作，**对任何数据进行读操作之前要申请获得S锁，在进行写操作之前要申请并获得X锁，加锁不成功，则事务进入等待状态**，直到加锁成功才继续执行。加锁后无法解锁。

   释放锁的阶段称为收缩阶段，当事务释放一个锁后，事务进入收缩阶段，在该阶段只能解锁不能加锁。

   提交见另复习

简答题

1. 什么是高内聚低耦合（必考）★

   高内聚是指一个软件模块是由相关性很强的代码组成，只负责一项任务，也就是常说的单一责任原则。

   低耦合，粗浅的理解是：一个完整的系统，模块与模块之间，尽可能的使其独立存在。

（1） 高内聚指的是块内联系高，低耦合指的是块间联系低

（2） **如何解耦**：一般采用插入第三者的方式

2. 命名的时候，名字和地址如何解耦

   命名服务：

   在一个集中的位置存储信息。比如主机名+地址，用户，计算机，应用程序就可以通过网络进行通信。

   如果不使用的话，比如3台计算机通信，每台计算机都需要保存其余计算机的网络地址，耦合程度大。

   如果设置一台可公共访问的服务器存储信息，则可以解耦。

3. 消息型中间件如何帮助解耦 ★

   答：消息中间件是在分布式系统中完成消息的发送和接收的软件。消息的收发两方完全是松耦合的。

   消息中间件通过消息代理实现解耦。在基于消息代理的分布式应用系统中，消息的发送方称为出版者，消息的接收方称为订阅者，不同的消息通过不同的主题进行区分。

   ①时间解耦：发布方和订阅方无需同时在线就能进行消息传输，消息中间件通过存储转发提供了这种异步传输的能力。

   ②空间解耦：发布方和订阅方都无需知道对方的物理地址、端口等。

   ③流程解耦：发布方和订阅方在发送和接收数据时并不阻塞各自的控制流程。

4. 介绍producer和consumer

   答：Producer和Consumer分别是消息队列中业务的发起方和处理方，Producer负责生产消息传输到Broker，而Consumer负责从Broker中获取消息并进行业务逻辑处理。

5. 计算架构，cs,bs架构，分布式，云计算，端边云协同，云编译，演进背后的驱动力（1技术上驱动力：计算分配，要求，2商业驱动力：计算资源由service提供者掏钱还是用户掏钱）

   答：演进顺序：单机——C/S、B/S——分布式——云计算——端管边云融合，其背后驱动力有：

   技术上驱动力 ：计算要求、计算分配

   商业上的驱动力：单位计算资源越来越便宜，按量计价如果再加一句，就是谁掏钱的问题，是由Service Provider（消息提供者）来掏钱，还是由Consumer（客户）来掏这些计算资源的钱。

   （C/S架构：客户端/服务器架构，每一个客户端的实例都可以向一个服务器发出请求

   B/S架构：与C/S不同，客户端不需要安装专门的软件，只需要浏览器就可。浏览器和Web服务器交互，Web服务器和后端数据库交互。）

6. 池的概念，池在面向对象，为什么要建立对象池，什么是对象池，减少建立和关闭操作，保证连接数不至于过多，连接池，对象池，

   如何提高性能，提高性能的原理是什么（降低建立和退出的开销，对建立的资源实现一个高效的并发调度）

   为什么要数据库连接池：频繁建立和断开需要消耗大量资源

   答：池可以想象为一个容器，保存着各种软件需要的对象，分为资源池（对象池）和线程池。

   资源池提前保存大量的资源对象，以解决资源频繁分配和释放所造成的性能问题。

7. ★ 云原生的**概念**。以日志系统为例解释云原生的主要特点，主要有四个特点，相对于传统技术的优点。

   答：

云原生是一种构建和运行应用程序的方法，是一套技术体系和方法论。**云原生（CloudNative）是一个组合词，Cloud+Native。**Cloud表示应用程序位于云中，而不是传统的数据中心；Native表示应用程序从设计之初即考虑到云的环境，原生为云而设计，在云上以最佳姿势运行，充分利用和发挥云平台的弹性+分布式优势。

四个特点：DevOps+持续交付+微服务+容器。

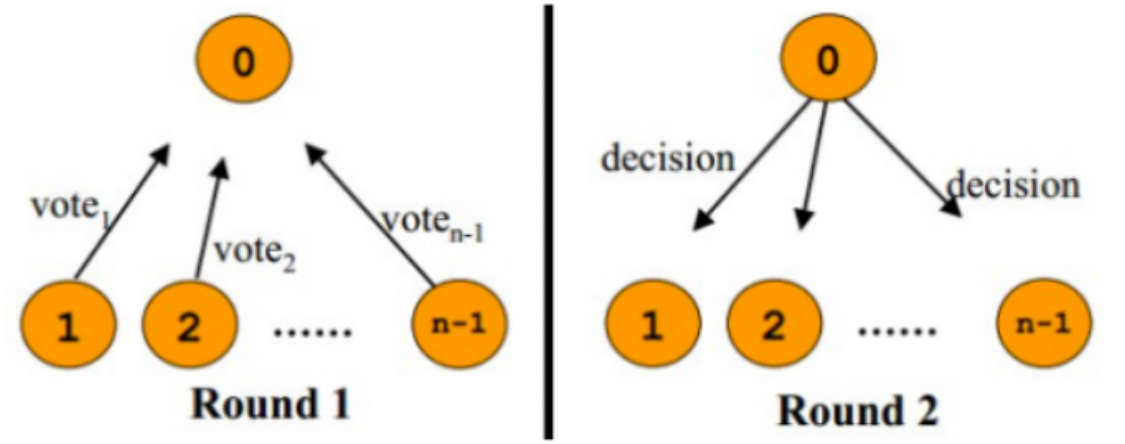
总而言之，符合云原生架构的应用程序应该是：采用开源堆栈（K8S+Docker）进行容器化，基于微服务架构提高灵活性和可维护性，借助敏捷方法、DevOps支持持续迭代和运维自动化，利用云平台设施实现弹性伸缩、动态调度、优化资源利用率。

8. 两阶段提交，那张图必考⭐处理异常

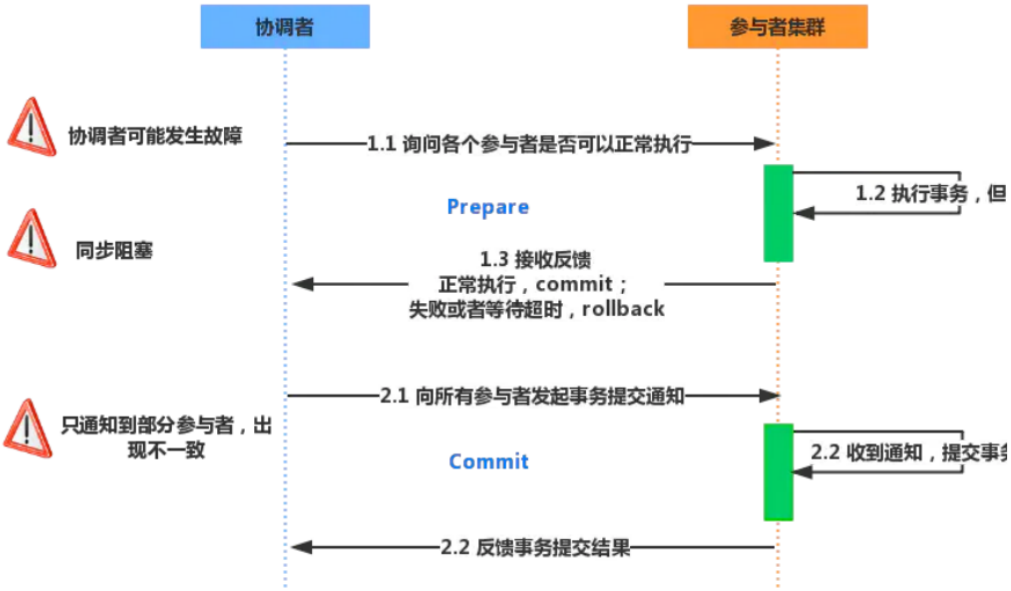
在分布式系统中，事务往往包含有多个参与者的活动，单个参与者上的活动是能够保证原子性的，而多个参与者 之间原子性的保证则需要通过两阶段提交来实现，**两阶段提交是分布式事务实现的关键**。

在分布式事务两阶段提交协议中，有一个主事务管理器负责充当分布式事务**协调器**的角色。

事务协调器负责整个事务，并使之与网络中的其他事务管理器（参与者）协同工作 协调者可以看做成事务的发起者，同时也是事务的一个参与者。对于一个分布式事务来说，一个事务是涉及到多个参与者的。



时间轴出现的特殊异常情况的处理，按时间轴列举出异常情况以及如何处理



第一阶段

首先，协调者在自身节点的日志中写入一条的日志记录，然后**所有参与者发送消息prepare T**，询问这些参与者 （包括自身），是否能够提交这个事务；

参与者在接受到这个prepare T 消息以后，会根据自身的情况，进行事务的**预处理**：

如果参与者能够提交该事务，则会将日志写入磁盘，并**返回给协调者一个ready T信息，同时自身进入预提交状态**；

如果不能提交该事务，则记录日志，**并返回一个not commit T信息给协调者，同时撤销在自身上所做的数据库 改**；

第二阶段

协调者会收集所有参与者的意见，如果收到参与者发来的not commit T信息，则标识着该事务不能提交，协调者 会将Abort T 记录到日志中，并向所有参与者发送一个Abort T 信息，让所有参与者撤销在自身上所有的预操作：

如果协调者收到所有参与者发来prepare T信息，那么协调者会将Commit T日志写入磁盘，并向所有参与者发送一 个Commit T信息，提交该事务。

如果协调者迟迟未收到某个参与者发来的信息，则认为该参与者发送了一个VOTE_ABORT信息，从而取消该事 务的执行。

参与者接收到协调者发来的Abort T信息以后，参与者会终止提交，并将Abort T 记录到日志中；

如果参与者收到的是Commit T信息，则会将事务进行提交，并写入记录

糟糕的情况

如果参与者收不到协调者的Commit 或Abort指令，参与者将处于“状态未知”阶段，参与者将不知道要如何操作。 比如,如果所有的参与者完成第1阶段的回复后（可能全部yes，可能全部no，可能部分yes部分no），如果协调 者在这个时候宕机了, 那么所有的参与者将无所适从。可能需要数据库管理员的介入，防止数据库进入一个不 一致的状态。

9. 微服务架构。与单体结构比较，微服务架构优势：有利于进行纵向整合，而soa主要进行横向整合，
10. 负载均衡

负载均衡几种方式：轮询，dns

轮叫调度（1：1）、加权轮叫调度、最少链接调度、加权最小链接调度。

2层（数据链路层）：采用虚拟IP（VIP），集群中不同的机器采用相同IP地址

3层（网络层）VIP，但是集群中不同的机器采用不同的IP地址。

4层（传输层）修改IP+端口号

7层（应用层）DNS HTTP Redis等

11. 对象的序列化和反序列化过程，解释一下概念以及作用，主要是传输对象的转换

    序列化：把对象转换为字节序列的过程，反序列化就是把字节序列还原为对象的过程。

    对象序列化成的字节序列会包含对象的类型信息、对象的数据等，说白了就是包含了描述这个对象的所有信息， 能根据这些信息“复刻”出一个和原来一模一样的对象。

    为什么要序列化：凡是需要跨平台存储和网络传输的数据，都需要序列化。网络直接传输数据，但是无法直接传输对象，可在传输前序列化，传输完成后反序列化成对象。 所以所有可在网络上传输的对象都必须是可序列化的。

12. 面向切面编程包含哪些要素

AOP将应用程序分为两个部分：核心业务逻辑和横向的通用逻辑（即所谓的方面）

主要包含：Aspect（切面）、Join Point（连接点）、Advice（通知）、PointCut（切入点）、Introduction（引入）、Target Object（目标对象）、AOP proxy（AOP代理对象）、Weaving

13. 透明性的概念，有哪些透明性6-7个（位置的透明性，迁移的透明性）

    答：透明性是一种现象，即一个系统的分布情况对于用户和应用来说是隐藏的。

    主要包括：访问透明、位置透明、迁移透明、重定位透明、复制透明、并发透明、故障透明。

14. 解释applet，技术优点，特点：可以用java语言编程的方法来编写服务端程序；（为什么不能流行）
    applet缺点：依赖于jdk或jre，微软取消了jre则applet不能运行，而servelet可以运行

15. springcloud包含哪些核心组件，简要介绍组件的核心运行流程

    答：Spring Cloud包括了5大核心组件：

    服务发现组件Eureka、

    客服端负载均衡Ribbon、

    熔断器Hystrix、

    微服务网关 Zuul和

    分布式配置Spring Cloud Config。

    运行流程：

    1、请求统一通过API网关（Zuul）来访问内部服务

    2、网关接收到请求，从注册中心（Eureka）获取可用服务

    3、由Ribbon进行负载均衡后，分发到后端具体实例

    4、微服务之间通过消息总线进行通信处理业务

    5、Hystrix负责处理服务超时熔断

16. dns怎么实现负载均衡（属于第几层的负载均衡）,其基本原理是什么，有什么优缺点？ ★

    DNS属于第七层的负载均衡。其基本原理是为统一给域名配置多个不同的地址，查询域名的客户机可获得其中一个地址，对于同一个域名不同客户机会得到不同的地址，访问不同地址是的Web服务器。

    优点：实现简单、成本低

    缺点：负载分配不均匀，未考虑每个Web服务器的负载情况，最慢的将成为系统的瓶颈。

    可靠性低：若某台出现故障，DNS仍会分配，导致不能相应

    变更生效时间长。

17. 数据库连接池的核心思想：池化的作用，对对象的池化 ★

    数据库连接池的核心思想的数据库连接的复用。

    通过建立一个数据库连接 池以及一套连接使用、分配、管理策略，连接池中的连 接可以得到高效、安全的复用，避免了数据库连接频繁 建立、关闭的开销。

综合题：

1. 消息型中间件的topic,queue概念，如何用消息型中间件实现解耦异步和削峰
2. cobra为什么在推广上会失败，说说可以跨平台跨语言的技术框架的未来发展
3. 什么是web服务，web服务是有状态还是无状态。rest风格的web服务功能可以具有状态么，给出答案并解释原因（这道题不一定是综合题，但必然以某种方式出现）★

   答：**Web服务（Web service）**是一种面向服务的架构的技术，通过标准的Web协议提供服务，目的是保证不同平台的应用服务可以互操作。是无状态的。客户端和服务器之间的交互在请求之间是无状态的。

   RESTful Web Service具有以下特点：

   无状态的

   每一个来自客户端的request必须包含所有必要的信息，即不能从服务器端获得任何保存的上下文信息。

   REST 的“客户机- 无状态- 服务器”约束禁止在服务器上保存会话状态。符合这一约束进行设计

   1，可以提高系统的可靠性和可伸缩性。它不需要昂贵的维护和支持费用，因为状态并不在服务器上维护。

   2，可以进行资源缓存。Web的高速缓存既可以驻留在客户主机中，也可以驻留在中间网络高速缓存服务器主机中。
4. 从技术和商业角度，阐述Kubernetes产生以及成为业界主流的原因，说明其主要特征，阐述其与docker的关系★

   答：

   技术角度：

   容器技术启动速度快，基本不消耗额外的系统资源。Docker是应用最广泛的容器技术，通过打包镜像、启动容器来创建一个服务。

   但随着应用的逐渐复杂，容器的数量越来越多，衍生了管理运维容器的问题，K8S问世。它可以使得应用的部署和运维更加方便。

   商业角度：

   K8S在诞生之初就为云时代而生，拥有超前的眼光和先进的设计理念，最初由Google天才工程师设计，诞生之后就飞速发展。随着全球的优秀工程师大力投入，打造了繁荣的社区和生态系统，它成为了企业容器编排系统的首选。

   二者的关系：

   Docker是一个开源的应用容器引擎，开发者可以打包他们的应用及依赖到一个可移植的容器中。

   K8S是一个开源的容器集群管理系统，可以实现容器集群的自动化部署、自动扩缩容、维护等功能。

开放设计题：

1. ★

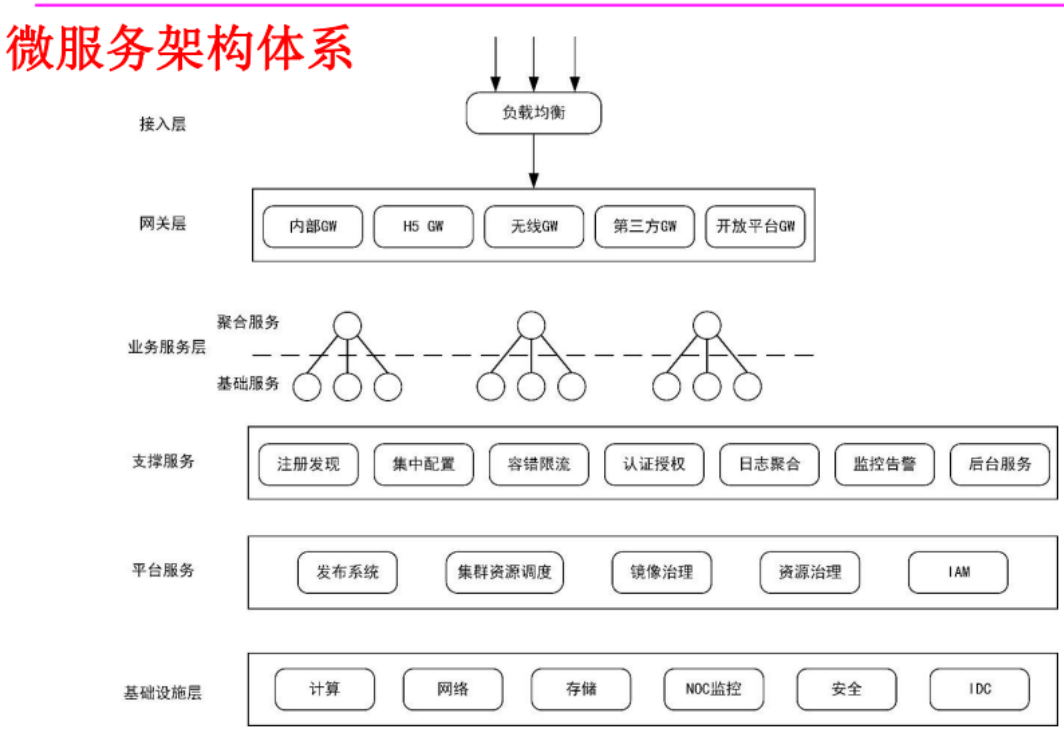
   （1）以大规模在线订餐，设计系统架构，画出整体架构图，解释其主要特征，

   （2）如果是基于微服务的设计，该如何对其系统功能服务进行划分（主要是纵向拆解）

   需要遵循的原则：高内聚低耦合，服务粒度适中（横向拆分）、围绕业务概念建模（确定功能边界）、演进式拆分（先粗后细）

   微服务的纵向拆分最多三层。分别是：1）基础服务层、2 ）组合服务层、3）控制层。

比如交易服务可以拆分成下单、拆单、校验、支付等。



（3）从中间件的角度谈谈，可以从哪些层次，哪些技术方案以应对短时间内大量访问的应用请求，

负载均衡、事件处理、数据访问、缓存技术、数据库拆分、消息队列、池化技术、程序优化

2. 以12306为例，订票高峰时，包括设计技术架构，技术特征，以及讨论有哪些技术方案以应对短时间内大量访问的应用请求，

   中间件在架构上提供了分布式互操作的便利性，使得程序员的注意力可以集中在业务逻辑上。

押题：

一、概念解释

1. 中间件
2. 分布式操作
3. CORBA对象总线
4. 容器
5. 负载均衡
6. XaaS/RPC/IDL/IIOP/GIOP

二、简答题

1. 解释对象序列化，反序列化的过程及作用

   答：

   序列化：把对象转换为字节序列的过程

   反序列化：字节序列恢复为对象的过程

   作用：（1）持久化对象，序列化后保存在一个文件中。（2）在网络上传输对象

2. 什么是“高内聚，低耦合”？消息型中间件是如何解耦的？命名服务是如何解耦的？
3. 在事务型中间件中，解释什么是两段式加锁，什么是两段式提交？简要介绍其 原理及过程。
4. 云原生的**概念**。以日志系统为例解释云原生的主要特点，主要有四个特点，相对于传统技术的优点。

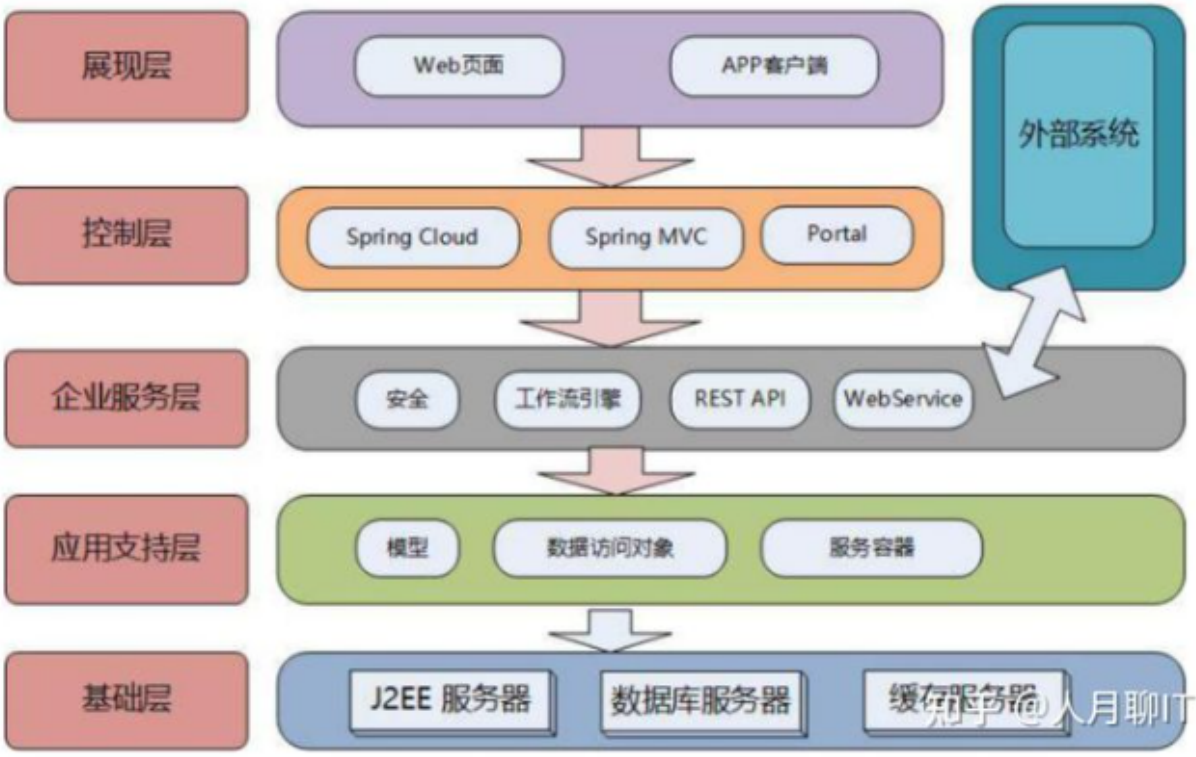
5. SpringCloud包含哪些核心组件，简要介绍组件的核心运行流程
6. DNS怎么实现负载均衡（属于第几层的负载均衡）,其基本原理是什么，有什么优缺点?
7. 数据库连接池的核心思想：池化的作用，对对象的池化

三、综合题

1. 从技术和商业角度阐述Kubernetes产生以及成为业界主流的原因，说明其主要特征，阐述其与docker的关系
2. 消息型中间件的topic,queue概念，如何用消息型中间件实现解耦异步和削峰
3. 什么是web服务，web服务是有状态还是无状态。rest风格的web服务功能可以具有状态么，给出答案并解释原因

四、设计题

（1）以大规模在线订餐，设计系统架构，画出整体架构图，解释其主要特征，



（2）如果是基于微服务的设计，该如何对其系统功能服务进行划分（主要是纵向拆解）

需要遵循的原则：高内聚低耦合，服务粒度适中（横向拆分）、围绕业务概念建模（确定功能边界）、演进式拆分（先粗后细）

微服务的纵向拆分最多三层。分别是：1）基础服务层、2 ）组合服务层、3）控制层。

比如交易服务可以拆分成下单、拆单、校验、支付等。

（3）从中间件的角度谈谈，可以从哪些层次，哪些技术方案以应对短时间内大量访问的应用请求，

负载均衡、事件处理、数据访问、缓存技术、数据库拆分、消息队列、池化技术、程序优化