2020 年系统架构设计师论文真题

试题一 论数据分片技术及其应用

数据分片就是按照一定的规则，将数据集划分成相互独立正交的数据子集。然后将数据子集分布到不同的节点上，通过设计合理的数据分片规则，可将系统中的数据分布在不同的物理数据库中，达到提升应用系统数据处理速度的目的。

请围绕“论数据分片技术及其应用”论题，依次从以下三个方面进行论述。

1.概要叙述你参与管理和开发软件的项目以及承担的工作

2. Hash 分片，一致性 Hash 分片和按照数据范围分片是三种常用的数据分片方式

3.具体阐述你参与管理和开发的项目，且采用了哪些分片方式，并且具体说明其实现过程和应用效果。

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、Redis数据分片方案

1、范围分片：按数据范围值来做分片。如：按用户编号分片，0-999999映射到实例A；1000000-1999999映射到实例B。

2、hash分片：通过对key进行hash操作，可以把数据分配到不同实例，这类似于取余操作，余数相同的，放在一个实例上。

3、一致性hash分片：hash分片的改进，可以有效解决重新分配节点带来的无法命中问题。

Redis的持久化解决方案

Redis的持久化主要有两种方式：RDB和AOF。

RDB：传统数据库中快照的思想。指定时间间隔将数据进行快照存储。

AOF：传统数据库中日志的思想，把每条改变数据集的命令追加到AOF文件末尾，这样出问题了，可以重新执行AOF文件中的命令来重建数据集。

三、第三个问题要根据项目的实际情况来写自己是怎么做的，遇到什么样的问题，如何解决的。同时文章收尾要对效果进行评价。

试题二 论云原生架构及其应用论题，依次从以下三个方面进行论述：

1.概要叙述你参与管理和开发的软件项目以及承担的主要工作。

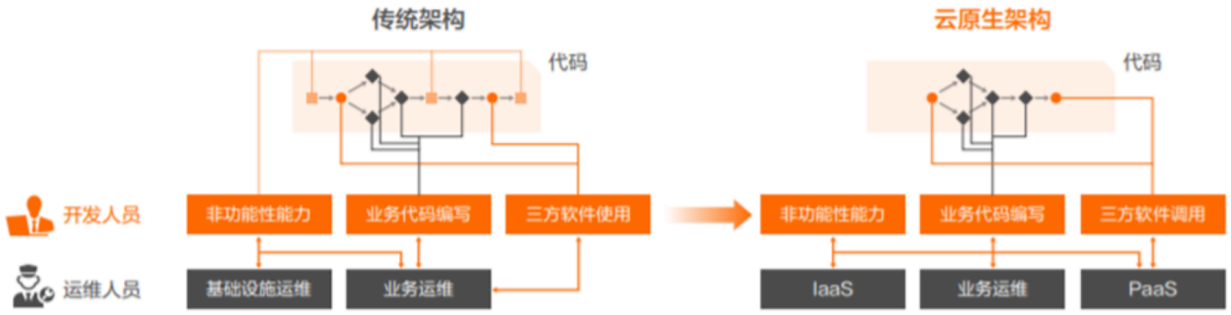
2.服务化，强性，可观测性和自动化是云原生架构重复的四类设计原则，请简要对这四类设计原则的内涵进行阐述。

3.具体阐述你参与管理和开发的项目是如何向采用云原生架构的，并且围绕上述四类设计原则详细论述在项目设计与实现过程中遇到了哪些实际问题，是如何解决的。

大纲：近年来，随着数字化转型不断深入，科技创新与业务不断融合，各行各业正在从大工业时代以容器和微服务架构为代表的云原生技术作为云计算服务的新模式已经逐渐成为企业持续发展的主流选择。

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、从技术的角度，云原生架构是基于云原生技术的一组架构原则和设计模式的集合，旨在将云应用中的非业务代码部分进行最大化的剥离，从而让云设施接管应用中原有的大量非功能特性，使业务不再有非功能性业务中断困扰的同时，具备轻量、敏捷、高度自动化的特点。



上图展示了在代码中通常包括三部分：业务代码、三方软件、处理非功能特性的代码。其中“业务代码”指实现业务逻辑的代码；“三方软件”是业务代码中依赖的所有三方库，包括业务库和基础库；“处理非功能性的代码”指实现高可用、安全、可观测性等非功能性能力的代码。三部分中只有业务代码是核心，是对业务真正带来价值的，另外两个部分都只算附属物，但随着软件规模的增大、业务模块规模变大、部署环境增多、分布式复杂性增强，使得今天的软件构建变得越来越复杂，对开发人员的技能要求也越来越高。云原生架构相比较传统架构进了一大步，从业务代码中剥离了大量非功能性特性（不会是所有，比如易用性还不能剥离）到 IaaS 和 PaaS 中，从而减少业务代码开发人

员的技术关注范围，通过云厂商的专业性提升应用的非功能性能力。

云原生架构原则：

云原生架构本身作为一种架构，也有若干架构原则作为应用架构的核心架构控制面，通过遵从这些架构原则可以让技术主管和架构师在做技术选择时不会出现大的偏差。

1、 服务化原则：

当代码规模超出小团队的合作范围时，就有必要进行服务化拆分了，包括拆分为微服务架构、小服务（Mini Service）架构，通过服务化架构把不同生命周期的模块分离出来，分别进行业务迭代，避免迭代频繁模块被慢速模块拖慢，从而加快整体的进度和稳定性。同时服务化架构以面向接口编程，服务内部的功能高度内聚，模块间通过公共功能模块的提取增加软件的复用程度。

分布式环境下的限流降级、熔断隔仓、灰度、反压、零信任安全等，本质上都是基于服务流量（而非网络流量）的控制策略，所以云原生架构强调使用服务化的目的还在于从架构层面抽象化业务模块之间的关系，标准化服务流量的传输，从而帮助业务模块进行基于服务流量的策略控制和治理，不管这些服务是基于什么语言开发的。

2、弹性原则：

大部分系统部署上线需要根据业务量的估算，准备一定规模的机器，从提出采购申请，到供应商洽谈、机器部署上电、软件部署、性能压测，往往需要好几个月甚至一年的周期；而这期间如果业务发生变化了，重新调整也非常困难。弹性则是指系统的部署规模可以随着业务量的变化自动伸缩，无须根据事先的容量规划准备固定的硬件和软件资源。好的弹性能力不仅缩短了从采购到上线的时间，让企业不用操心额外软硬件资源的成本支出（闲置成本），降低了企业的 IT 成本，更关键的是当业务规模面临海量突发性扩张的时候，不再因为平时软硬件资源储备不足而“说不”，保障了企业收益。

3、可观测原则：

今天大部分企业的软件规模都在不断增长，原来单机可以对应用做完所有调试，但在分布式环境下需要对多个主机上的信息做关联，才可能回答清楚服务为什么宕机、哪些服务违反了其定义的 SLO、目前的故障影响哪些用户、最近这次变更对哪些服务指标带来了影响等等，这些都要求系统具备更强的可观测能力。可观测性与监控、业务探活、APM 等系统提供的能力不同，前者是在云这样的分布式系统中，主动通过日志、链路跟踪和度量等手段，让一次 APP 点击背后的多次服务调用的耗时、返回值和参数都清晰可见，甚至可以下钻到每次三方软件调用、SQL 请求、节点拓扑、网络响应等，这样的能力可以使运维、开发和业务人员实时掌握软件运行情况，并结合多个维度的数据指标，获得前所未有的关联分析能力，不断对业务健康度和用户体验进行数字化衡量和持续优化。

4、自动化原则：

技术往往是把“双刃剑”，容器、微服务、DevOps、大量第三方组件的使用，在降低分布式复杂性和提升迭代速度的同时，因为整体增大了软件技术栈的复杂度和组件规模，所以不可避免地带来了软件交付的复杂性，如果这里控制不当，应用就无法体会到云原生技术的优势。通过 IaC（Infrastructure asCode）、GitOps、OAM（Open Application Model）、Kubernetes operator 和大量自动化交付工具在 CI/CD 流水线中的实践，一方面标准化企业内部的软件交付过程，另一方面在标准化的基础上进行自动化，通过配置数据自描述和面向终态的交付过程，让自动化工具理解交付目标和环境差异，实现整个软件交付和运维的自动化。

三、第三个问题要根据项目的实际情况来写自己是怎么做的，遇到什么样的问题，如何解决的。

试题三 围绕“论软件测试中缺陷管理及其应用”论题，依次从以下三个方面进行论述。

1.概要叙述你参与管理和开发的软件项目以及承担的工作

2.详细论述常见的缺陷种类及级别，论述缺陷管理和基本流程

3.结合你具体参与管理和开发的实际项目，说明是如何进行缺陷管理的。请具体说明实施过程及应用效果。

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、软件缺陷定义：

（1）软件未达到需求规格说明书的功能；

（2）软件出现了需求规格说明书指明不会出现的错误；

（3）软件功能超出需求规格说明书的范围；

（4）软件未达到需求规格说明书未指出但应达到的目标；

（5）测试工程师认为软件难以理解、不易使用、运行速度慢，或者最终用户认为不好。

缺陷类型：

（1）设计缺陷：由于软件设计或代码实现所产生的功能或流程的问题。

（2）界面问题：系统页面的展示的问题。

（3）数据问题：系统数据的来源，处理及处理结果的问题。

（4）需求问题：软件需求测试发现的问题，也包括之后需求变更的问题。

（5）安装部署：软件安装部署过程的错误。

（6）性能问题：软件性能相关的缺陷。

（7）文档问题：用户使用手册，软件帮助文档等出现的问题

（8）常识问题：系统用户的正常使用习惯相关问题。

（9）安全问题：系统漏洞安全问题。

（10）优化建议：针对操作过程逻辑或界面显示的优化性建议。

（11）其他：除前面分类的其他问题

严重程度定义

严重程度	定义和说明
致命	阻碍开发或测试工作继续进行的系统性故障， 例如： ➢ 实现的功能与产品定义或软件需求规格严重不符。 ➢ 系统无法执行、崩溃、冻结，死循环等。 ➢ 程序引起的死机，非法退出。 ➢ 主要功能模块严重错误。 ➢ 数据库链接错误，严重数据计算错误通讯错误等。
严重	系统出现的严重错误，但不影响当前测试工作的错误。 例如： ➢ 模块功能错误，模块功能未实现，乱码等； ➢ 功能错误，如链接模块有误，基本按键使用有误等。 ➢ 数据错误，如用户数据丢失、破坏、计算、保存有误等。
一般	不影响用户使用的非严重问题 ➢ 次要功能未实现或与需求不符。 ➢ 操作界面错误，如界面图表或字符的一般性错误，但不影响操作。 ➢ 提示信息错误，辅助说明不清楚。 ➢ 数据错误，数据边界、格式约束未实现或需求不一致
建议	测试过程中发现不利用户操作的优化建议。 ➢ 界面字符或提示与的显示不恰当。 ➢ 页面或操作习惯的优化性建议。 ➢ 功能操作更好的实现方式。

注：严重等级由创建者在创建缺陷时根据此定义来选择，之后都不能随意更改。

优先级的定义

优先级	定义和说明
立刻	阻碍测试工作无法进行，需要开发工程师马上解决问题。 ➢ 所有的致命问题都需要立刻解决； ➢ 时间急迫时影响版本上线的问题等；
紧急	不影响测试工作的严重问题，下个测试版本发版前必须解决。 ➢ 所有严重问题； ➢ 常用模块功能、业务逻辑或数据错误； ➢ 明显的性能问题；
尽快	一般性功能错误，版本发布前应该解决的问题。 ➢ 大多数一般问题； ➢ 页面显示，页面的字符，界面图标、文字显示、链接有误，不影响使用。如错别字，图标显示异常等；
一般	不影响版本上线的问题，部分问题允许不修改。 ➢ 非常用界面或字符的显示错误或不恰当； ➢ 用户使用习惯，语言表达等优化建议；

注：立刻、紧急、尽快的问题都要求在系统上线前解决。

缺陷处理过程：

正常处理过程

（1）创建问题

在测试管理系统中，所有用户都可以创建新问题，包括需求问题和软件缺陷等。创建问题时，需要描述清楚，并选择正确的选项，详细请参考5.4和5.5。

（2）指派问题

创建问题时，创建者通常要指派给该项目开发负责人，再由其指派任务，或直接指派给相应模块的开发工程师。

如果指派人是错误的，或者需要他人确认或帮助，则可以重新指派给合适的工程师，写上相关备注。

（3）确认问题

通常开发工程师收到新问题后，需要分析和确认此问题是否为Bug。如果是Bug，则选择“确认状态”；如果认为非Bug，则注明原因并指派回创建者。

当创建者收到确认指派时，需要进行及时确认。如果同意为非bug，则及时关闭它；如果不同意，则需要注明理由并指派回相关工程师。

如果问题确认指派次数大于6次时，需要进入“争议处理”流程，详细请参考5.2。

（4）解决问题

此为开发工程师的主要职责，包括Bug的复现、修改和修改验证。

开发工程师需要及时对确认状态Bug进行分析和解决，并自己验证通过，则操作为解决状态，解决方案规则请参考5.4中解决方案定义部分，在缺陷管理系统中解决方案选择相应的选项，解决后系统将自动指派回给创建者。

如果Bug无法解决或修改影响比较大，可申请进入“延期解决”流程，请参考5.2中延期处理部分。

（5）验证问题

创建者需要及时对解决状态的Bug在对应版本上面进行验证。如果验证通过，则可关闭Bug；如果验证不通过，则激活此Bug，系统将自动指派回给解决者。

验证通过准则：相同的操作步骤，进行一定次数的验证测试都没有发生。

验证不通过准则：相同的操作步骤，全部或部分实际结果还会发生，验证不通过则激活Bug。

（6）关闭问题

通过验证的Bug，验证者需要注明验证结果并进行关闭操作，系统将指派给Closed。

如果关闭状态的Bug在之后版本又会发生，则激活此Bug，系统将自动指派回给解决者。

特别处理过程

（1）客户问题

客户反馈的问题可以由客户直接反馈或项目经理、市场部等了解到的客户问题，经确认后的Bug提交到测试管理系统，按照以上处理流程进行处理，由创建者或测试组进行跟踪验证关闭。

创建客户问题时，创建者需要在Bug标题开头标记为[客户问题]，测试组负责检查和更正。

（2）争议处理

当开发和测试工程师对某问题有争议并且多次沟通无果时（暂定为6次），可以注明双方的理由，并指派给项目经理进行处理。项目经理可以召开评审会议，或者直接与双方沟通了解，并根据项目情况给出专业意见和最终决定。开发和测试工程师根据项目经理的最终决定执行。

（3）延期解决

当开发工程师对确认Bug进行解决时，发现或评估其解决时间紧或风险比较大等，可以说明原因或理由并指派给项目经理来确认。项目经理可以召开评审会议，或者直接沟通了解，并根据项目情况给出最终决定。如果不同意，项目经理将此Bug指派回开发工程师，开发工程师继续分析和解决。如果同意，项目经理需要在Bug标题开头标记为[延期解决]和在处理状态选择“延期解决”，然后注明解决时间计划并指派回开发工程师，开发工程师根据解决时间计划来规划和解决此Bug。

三、第三个问题要根据项目的实际情况来写自己是怎么做的，遇到什么样的问题，如何解决的。同时文章收尾要对效果进行评价。

试题四 围绕“论企业集成架构设计及应用”为题

1.概要叙述你参与的软件开发项目的及承担的主要工作

2.详细说明三类企业集成架构设计技术分列要解决的问题及其含义，并阐述每种技术具体包含了哪些集成架构。

3.根据你所参与的项目，说明用了哪些企业集成架构设计技术，实施效果如何。

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、企业信息集成是解决“孤岛”问题的需要，技术发展的同时也推动了集成架构等相关的研究。构建企业集成平台的首要目的是实现数据集成，即为平台上运行的各种应用、系统或服务，提供具有完整性、一致性和安全性的数据访问、信息查询及决策支持服务。

1、数据集成包括以下3种模式：数据联邦、数据复制和基于接口的数据集成。

数据联邦

数据联邦是指不同的应用共同访问一个全局虚拟数据库,通过全局虚拟数据库管理系统为不同的应用提供全局信息服务,实现不同的应用和数据源之间的信息共享和数据交换,其具体实现由客户端应用、全局信息服务和若干个局部数据源三部分组成。

数据复制模式

在数据复制模式中,通过底层应用数据源之间的一致性复制来实现(访问不同数据库的)不同应用之间的信息共享和互操作,其实现的关键是必须能够提供在两个或多个数据库系统之间实现数据转换和传输的基础结构(以屏蔽不同数据库间数据模型的差异)

基于接口的数据集成模式

在基于接口的数据集成模式中,不同的应用系统之间利用适配器(或接口代理)提供的应用编程接口来实现相互调用。应用适配器或接口代理通过其开放或私有接口将业务信息从其所封装的具体应用系统中提取出来,进而实现不同的应用系统之间业务数的共享与互交换,接口调用的方式可以采用同步调用方法,也可以采用基于消息中间件的异步方法来实现。

2、应用集成

应用集成是指两个或多个应用系统根据业务逻辑的需要而进行的功能之间的相调用和互操作，应用集成模式包括集成适配器,集成信使、集成面板和集成代理4种。

适配器集成模式

采用在需要交互的系统之间加入适配器的解决方案来实现企业原有应用系统与新实施系统之间的互操作。在应用系统提供的API的基础上（在应用系统没有提供API的情况下，可以在其数据库表结构一致的条件下，直接完成对数据库的写入和读出），通过适配器完成不同系统间数据格式及访问方式的转换与映射，进而实现不同的系统之间业务功能及业务数据的集成。

信使集成模式

随着企业中业务应用系统个数的增多,应用系统间的接口问题变得越来越复杂。基于信使的集成结构中,系统之间的通信和数据交换通过信使(消息代理)来实现,每个应用只需要建立与集成信使之间的接口连接,就可实现与所有通过集成信使相联的应用系统间的交互,这种结构大大减少了接口连接数量,同时由于采用了信使(消息代理)作为信息交流的中介,可以将应用之间的交互对通信服务能力的依赖程度降到最低；另外,当某一系统发生改变时、只需要改变信使中相应的部分,从而降低系统维护工作量和系统升级的难度。

面板集成模式

面板集成模式和面向对象的软件设计方法中的面板模式很相似,它是从应用交互实现的层面来描述客户端应用和服务器端应用集成的一种方法。集成面板可以为一对多、多对一、多对多等多种应用提供集或接口,在种模式中包含有一个成多个客户端应用、一个集成面数,一个成多个眼务器应用,集成面板通过对服务器端应用功能的抽象和简化,为客户端应用访间与调用服务器端应用来种简化的公共接口,集成面板在得到客户应用服务请求后,将客户的眼务转换成服务器端应用能理解的形式,并将该请求提交给服务器端应用。

代理集成模式

面板集成模式实现了服务器应用交互逻辑的分离，在代理执政模式中，由于不存在很明显的客户端应用和服务器端应用的划分，它仅需要将待集成的应用间的交互逻辑从应用中分离出来，并对应文件的交互逻辑进行封装，进而是由集成代理来引导多个应用之间的交互。

3、企业集成

企业应用软件系统从功能逻辑上可以分为表示、业务逻辑和数据三个层次，其中表示层负责完成系统与用户交互的接口定义，业务逻辑层主要是根据具体业务规则完成相应业务的数据处理，数据层负责存储由业务逻辑层处理或产生的业务数据，它是系统相对稳定的部分。

从企业集成运行的实现策略上看，EAI主要有如下三种实现模式：

前端集成模式

所谓前端集成模式，是指EAI侧重于业务应用系统表示层的集成，它主要通过单一的用户入口，实现跨多个应用事物的运作，这种方式适用于用户启动的业务过程中，会产生多种多个跨应用的事物，而且这些事物都需要实时响应的情况（主要指B2C的环境）。另外，采用前端集成模式，还可以实现对已经运行的核心业务系统增加功能或特性的目的。

后端模式

后端集成模式主要侧重于应用系统数据层面的集成，它通过专门的数据维护及转换工具。实现不同业务义勇或数据源之间的信息交换，维护企业整体业务数据完整性和一致性，后端模式就像是一个方便多个应用系统之间数据自动交互的数据管道。

混合集成模式

混合集成模式是前端集成模式和后端集成模式的组合，客户通过基于Web浏览器的客户端（瘦客户）实现对业务应用或EAI服务器的访问，服务请求可以由前端应用系统执行，也可以通过EAI服务器将服务请求路由到后端，由后端的业务应用来执行。这种模式几乎具有前端集成模式和后端级模式的所有特征，主要应用于需要响应大量服务请求，又需要维护多个数据源的完整性和一致性的情况。

三、第三个问题要根据项目的实际情况来写自己是怎么做的，遇到什么样的问题，如何解决的。同时文章收尾要对效果进行评价。