# 系统架构设计师考试背记精要

## 1、架构的本质： （1）软件架构为软件系统提供了一个结构、行为和属性的高级抽象。（2）软件架构风格是特定应用领域的惯用模式，架构定义一个词汇表和一组约束。

## 2、数据流风格：适合于分阶段做数据处理，交互性差，包括：批处理序列、管理过滤器。

3、调用/返回风格：一般系统都要用到，包括：主程序/子程序，面向对象，层次结构（分层越多，性能越差）。

## 4、独立构件风格：构件是独立的过程，连接件是消息传递。包括：进程通信，事件驱动系统（隐式调用）。应用场景，通过事件触发操作。

## 5、虚拟机风格：包括解释器与基于规则的系统，有自定义场景时使用该风格。

6、仓库风格（以数据为中心的风格）：以共享数据源为中心，其它构件围绕中心进行处理。包括：数据库系统、黑板系统（语言处理，信号处理），超文本系统。

7、闭环控制架构（过程控制）：定速巡航，空调温控。

8、MVC：视图（JSP），控制器（Servlet），模型（EJB）。

9、SOA：粗粒度，松耦合，标准化。Webservice与ESB是SOA的实现技术。

10、ESB：位置透明性、消息路由、服务注册命名、消息转换、多传输协议、日志与监控。

## 11、REST的5大原则：所有事物抽象为资源、资源唯一标识、通过接口操作资源、操作不改变资源标识、操作无状态。

## 12、微服务特点：小, 且专注于做⼀件事情；轻量级的通信机制；松耦合、独立部署。

13、微服务优势：技术异构性、弹性、扩展、简化部署、与结构相匹配、可组合性、对可替代性的优化。

14、微服务与SOA对比：

|  |  |
| --- | --- |
| 微服务 | SOA |
| 能拆分的就拆分 | 是整体的，服务能放一起的都放一起 |
| 纵向业务划分 | 是水平分多层 |
| 由单一组织负责 | 按层级划分不同部门的组织负责 |
| 细粒度 | 粗粒度 |
| 两句话可以解释明白 | 几百字只相当于SOA的目录 |
| 独立的子公司 | 类似大公司里面划分了一些业务单元（BU） |
| 组件小 | 存在较复杂的组件 |
| 业务逻辑存在于每一个服务中 | 业务逻辑横跨多个业务领域 |
| 使用轻量级的通信方式，如HTTP | 企业服务总线（ESB）充当了服务之间通信的角色 |

15、MDA的3种核心模型：平台独立模型（PIM），平台相关模型（PSM），代码Code:。

16、ADL的三个基本元素：构件，连接件，架构配置。

17、DSSA基本活动：领域分析（建立领域模型），领域设计（获得DSSA），领域实现（开发和组织可复用信息）。

18、DSSA角色：领域专家（有经验的用户、分析、设计、实现人员，“给建议”），领域分析人员（有经验的分析师，完成领域模型），领域设计人员（有经验的设计师，完成DSSA），领域实现人员（有经验的程序员完成代码编写）。

19、DSSA三层次模型：领域架构师对应领域开发环境，应用工程师对应领域特定的应用开发环境，操作员对应应用执行环境。

20、ABSD方法是架构驱动，即强调由业务、质量和功能需求的组合驱动架构设计。

21、ABSD方法有三个基础：功能的分解，通过选择架构风格来实现质量和业务需求，软件模板的使用。

22、ABSD开发过程：

（1）架构需求（需求获取、生成类图、对类进行分组、打包成构件、需求评审）

（2）架构设计（提出架构模型、映射构件、分析构件相互作用，产生架构，设计评审）

（3）架构文档化：从使用者角度编写，分发给所有相关开发人员，保证开发者手中版本最新。

（4）架构复审：标识潜在的风险，及早发现架构设计中的缺陷和错误。

（5）架构实现（复审后的文档化架构，分析与设计，构件实现，构件组装，系统测试）

（6）架构演化（需求变化归类，架构演化计划，构件变动，更新构件相互作用，构件组装与测试，技术评审，演化后的架构）

23、架构评审四大质量属性：

（1）性能：代表参数（响应时间、吞吐量），设计策略（优先级队列、资源调度）。

（2）可用性：尽可能少的出错与尽快的恢复。代表参数（故障间隔时间，故障修复时间），设计策略（冗余、心跳线）。

（3）安全性：破坏机密性、完整性、不可否认性及可控性等特性。设计策略（追踪审计）

（4）可修改性：新增功能多少人月能完成，设计策略（信息隐藏，低耦合）

24、风险点：系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。

非风险点：一般以某种做法，“是可以实现的”、“是可以接受的”方式进行描述。

敏感点：指为了实现某种特定的质量属性，一个或多个构件所具有的特性。

权衡点：影响多个质量属性的特性，是多个质量属性的敏感点。

25、基本场景的评估方法：ATAM，SAAM，CBAM。

26. SAAM：最初用于分析架构可修改性，后扩展到其他质量属性。

SAAM五个步骤：即场景开发、体系结构描述、单个场景评估、场景交互和总体评估。

27、ATAM四大阶段：场景和需求收集、结构视图场景实现、属性模型构造和分析、折中。

ATAM：在SAAM的基础上发展起来的，主要针对性能、实用性、安全性和可修改性，在系统开发之前，对这些质量属性

进行评价和折中。

28、产品线技术应用场景：有多年行业开发经验，做过多个同类产品。

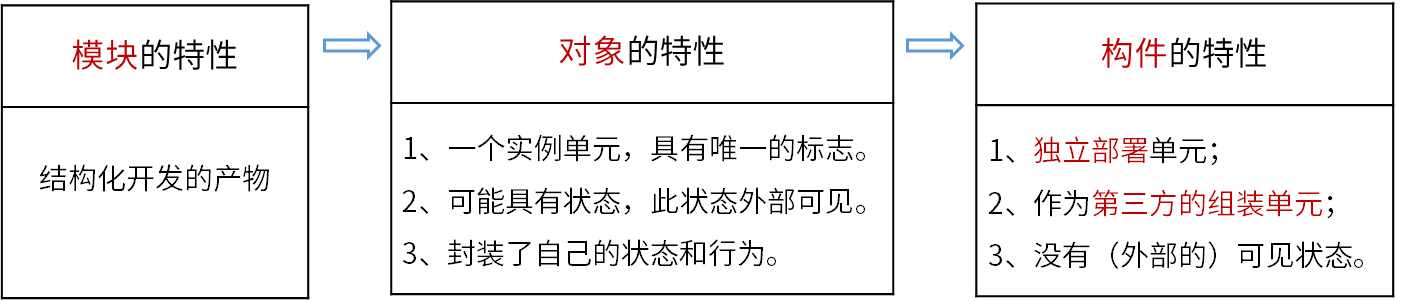
建立产品线的四种方式：基于现有产品演化式（风险最低），基于现有产品革命式，全新产品线演化式，全新产品线革命式（风险最高）。

|  |  |  |
| --- | --- | --- |
|  | 演化方式 | 革命方式 |
| 基于现有产品 | 基于现有产品架构设计产品线的架构，经演化现有构件，开发产品线构件 | 核心资源的开发基于现有产品集的需求和可预测的、将来需求的超集 |
| 全新产品线 | 产品线核心资源随产品新成员的需求而演化 | 开发满足所有预期产品线成员的需求的核心资源 |

29、软件产品线组织结构类型：设立独立的核心资源小组，不设立独立的核心资源小组，动态的组织结构。

30、产品线实施成功的决定因素：对该领域具备长期和深厚的经验；一个用于构建产品的好的核心资源库；好的产品线架构；好的管理（软件资源、人员组织、过程）支持。

31、构件、对象、模块的对比：



32、中间件：中间件是一种独立的系统软件或服务程序，可以帮助分布式应用软件在不同的技术之间共享资源。

33、中间件功能：客户机与服务器之间的连接和通信，客户机与应用层之间的高效率通信；应用层不同服务之间的互操作，应用层与数据库之间的连接和控制；多层架构的应用开发和运行的平台，应用开发框架，模块化的应用开发；屏蔽硬件、操作系统、网络和数据库的差异；应用的负载均衡和高可用性、安全机制与管理功能，交易管理机制，保证交易的一致性、一组通用的服务去执行不同的功能，避免重复的工作和使应用之间可以协作。

34、采用中间件技术的优点：面向需求；业务的分隔和包容性；设计与实现隔离；隔离复杂的系统资源；符合标准的交互模型；软件复用；提供对应用构件的管理。

35、主要的中间件：远程过程调用；对象请求代理；远程方法调用；面向消息的中间件；事务处理监控器。

36、中间件技术-Corba(公共对象请求代理体系结构)（代理模式）：

伺服对象（Servant）：CORBA对象的真正实现，负责完成客户端请求。  
对象适配器（Object Adapter）：用于屏蔽ORB内核的实现细节，为服务器对象的实现者提供抽象接口，以便他们使用ORB内部的某些功能。  
对象请求代理（Object Request Broker）：解释调用并负责查找实现该请求的对象，将参数传给找到的对象，并调用方法返回结果。客户方不需要了解服务对象的位置、通信方式、实现、激活或存储机制。

37、Bean 的分类：

（1）会话Bean：描述了与客户端的一个短暂的会话。

（2）实体Bean：持久化数据，O/R映射。

（3）消息驱动Bean：会话Bean+JMS，客户把消息发送给JMS目的地，然后，JMS提供者和EJB容器协作，把消息发送给消息驱动Bean。支持异步消息。

38、WEB设计维度：

（1）从架构来看：MVC，MVP，MVVM，REST，Webservice，微服务。

（2）从缓存来看：MemCache，Redis，Squid。

（3）从并发分流来看：集群（负载均衡）、CDN。

（4）从数据库来看：主从库（主从复制），内存数据库，反规范化技术，NoSQL，分区（分表）技术，视图与物化视图。

（5）从持久化来看：Hibernate，Mybatis。

（6）从分布存储来看：Hadoop，FastDFS，区块链。

（7）从数据编码看： XML，JSON。

（8）从Web应用服务器来看： Apache，WebSphere，WebLogic，Tomcat，JBOSS，IIS。

（9）其它：静态化，有状态与无状态，响应式Web设计。

39、集群：（1）应用服务器集群；（2）主从集群。

40、负载均衡技术：（1）应用层负载均衡：http重定向、反向代理服务器；（2）传输层负载均衡：DNS域名解析负载均衡、基于NAT的负载均衡；（3）硬件负载均衡：F5；（6）软件负载均衡：LVS、Nginx、HAproxy。

41、有状态和无状态：

（1）无状态服务（stateless service）对单次请求的处理，不依赖其他请求，也就是说，处理一次请求所需的全部信息，要么都包含在这个请求里，要么可以从外部获取到（比如说数据库），服务器本身不存储任何信息。

（2）有状态服务（stateful service）则相反，它会在自身保存一些数据，先后的请求是有关联的。

42、Redis 与 Memcache 能力比较

|  |  |  |
| --- | --- | --- |
| 工作 | MemCache | Redis |
| 数据类型 | 简单key/value结构 | 丰富的数据结构 |
| 持久性 | 不支持 | 支持 |
| 分布式存储 | 客户端哈希分片/一致性哈希 | 多种方式，主从、Sentinel、Cluster等 |
| 多线程支持 | 支持 | 不支持（Redis6.0开始支持） |
| 内存管理 | 私有内存池/内存池 | 无 |
| 事务支持 | 不支持 | 有限支持 |
| 数据容灾 | 不支持，不能做数据恢复 | 支持，可以在灾难发生时，恢复数据 |

43、Redis集群切片的常见方式

|  |  |
| --- | --- |
| 集群切片方式 | 核心特点 |
| 客户端分片 | 在客户端通过key的hash值对应到不同的服务器。 |
| 中间件实现分片 | 在应用软件和Redis中间，例如：Twemproxy、Codis等，由中间件实现服务到后台Redis节点的路由分派。 |
| 客户端服务端协作分片 | RedisCluster模式，客户端可采用一致性哈希，服务端提供错误节点的重定向服务slot上。不同的slot对应到不同服务器。 |

44、Redis分布式存储方案

|  |  |
| --- | --- |
| 分布式存储方案 | 核心特点 |
| 主从（Master/Slave）模式 | 一主多从，故障时手动切换。 |
| 哨兵（Sentinel）模式 | 有哨兵的一主多从，主节点故障自动选择新的主节点。 |
| 集群（Cluster）模式 | 分节点对等集群，分slots，不同slots的信息存储到不同节点。 |

45、Redis数据分片方案

|  |  |  |
| --- | --- | --- |
| 分片方案 | 分片方式 | 说明 |
| 范围分片 | 按数据范围值来做分片 | 例：按用户编号分片，0-999999映射到实例A；1000000-1999999映射到实例B。 |
| 哈希分片 | 通过对key进行hash运算分片 | 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。 |
| 一致性哈希分片 | 哈希分片的改进 | 可以有效解决重新分配节点带来的无法命中问题。 |

46、缓存与数据库的协作

数据读取：根据key从缓存读取；若缓存中没有，则根据key在数据库中查找；读取到“值”之后，更新缓存。

数据写入：根据key值写数据库；根据key更新缓存。

47、REST概念：REST（Representational State Transfer，表述性状态转移）是一种只使用HTTP和XML进行基于Web通信的技术，可以降低开发的复杂性，提高系统的可伸缩性。

REST的五个原则：网络上的所有事物都被抽象为资源；每个资源对应一个唯一的资源标识；通过通用的连接件接口对资源进行操作；对资源的各种操作不会改变资源标识；所有的操作都是无状态的。

48、响应式Web设计：响应式WEB设计是一种网络页面设计布局，其理念是：集中创建页面的图片排版大小，可以智能地根据用户行为以及使用的设备环境进行相对应的布局。方法：采用流式布局和弹性化设计、响应式图片。

49、主从数据库结构特点：

一般：一主多从，也可以多主多从。

从库做写操作，从库做读操作。

主从复制步骤：

主库（Master）更新数据完成前，将操作写binlog日志文件。

从库（Salve）打开I/O线程与主库连接，做binlog dump process，并将事件写入中继日志。

从库执行中继日志事件，保持与主库一致。

50、反规范化的技术手段以及优缺点

|  |  |
| --- | --- |
| 技术手段 | 说明 |
| 增加派生性冗余列 | 已有单价和数量列，增加“总价”列 |
| 增加冗余列 | 已有学号列，增加“姓名”列 |
| 重新组表 | 把拆分的表重新组表 |
| 分割表 | 把用户表做水平分割，长沙的用户存在长沙，上海的用户存在上海 |

反规范化的优点：连接操作少，检索快、统计快；需要查的表减少，检索容易。

|  |  |
| --- | --- |
| 反规范化的缺点 | 解决方案 |
| 数据冗余，需要更大存储空间 | 无解 |
| 插入、更新、删除操作开销更大 | 无解 |
| 数据不一致  可能产生添加、修改、删除异常 | 1、触发器数据同步  2、应用程序数据同步  3、物化视图 |
| 更新和插入代码更难写 | 无解 |

51、视图的优点：（1）视图能简化用户的操作（2）视图机制可以使用户以不同的方式查询同一数据（3）视图对数据库重构提供了一定程度的逻辑独立性（4）视图可以对机密的数据提供安全保护其中物化视图：将视图的内容物理存储起来，其数据随原始表变化，同步更新。52、分表和分区

|  |  |  |
| --- | --- | --- |
|  | 分区 | 分表 |
| 共性 | 1、都针对数据表 2、都使用了分布式存储  3、都提升了查询效率 4、都低数据库的频繁I/O压力值 | |
| 差异 | 逻辑上还是一张表 | 逻辑上已是多张表。 |

53、分区的优点：（1）相对于单个文件系统或是硬盘，分区可以存储更多的数据。（2）数据管理比较方便，比如要清理或废弃某年的数据，就可以直接删除该日期的分区数据即可。（3）精准定位分区查询数据，不需要全表扫描查询，大大提高数据检索效率。（4）可跨多个分区磁盘查询，来提高查询的吞吐量。（5）在涉及聚合函数查询时，可以很容易进行数据的合并。54、关系型数据库和NoSQL对比

|  |  |  |
| --- | --- | --- |
| 对比维度 | 关系数据库 | NoSQL |
| 应用领域 | 面向通用领域 | 特定应用领域 |
| 数据容量 | 有限数据 | 海量数据 |
| 数据类型 | 结构化数据【二维表】 | 非结构化数据 |
| 并发支持 | 支持并发、但性能低 | 高并发 |
| 事务支持 | 高事务性 | 弱事务性 |
| 扩展方式 | 向上扩展 | 向外扩展 |

55、嵌入式微处理器分类

（1）嵌入式微控制器（MCU：Micro Controller Unit）：又称为单片机，片上外设资源一般比较丰富，适合于控制。

（2）嵌入式微处理器（EMPU：Embedded Micro Processing Unit）： 又称为单板机， 由通用计算机中的CPU发展而来，仅保留和嵌入式应用紧密相关的功能硬件。

（3）嵌入式DSP处理器（DSP：Digital Signal Processor）：专门用于信号处理方面的处理器。

（4）嵌入式片上系统（SOC）：追求产品系统最大包容的集成器件。

成功实现了软硬件的无缝结合，直接在微处理器片内嵌入操作系统的代码模块。

减小了系统的体积和功耗、提高了可靠性和设计生产效率。