第一章 软件架构设计

1 软件架构的概念

架构设计就是_____，即将满足需求的职责分配到组件上。

软件架构风格是描述某一特定应用领域中系统组织方式的惯用模式。架构风格定义一个系统家族，即一个体系结构定义一个词汇表和一组约束。词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构件和连接件组合起来的。

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，由构成系统的元素的描述、这些元素的相互作用、指导元素集成的模式以及这些模式的约束组成。

架构的本质：

软件架构为软件系统提供了一个_____。

软件架构风格是特定应用领域的_____，架构定义_____。

架构的作用：

软件架构是_____，明确了对系统实现的约束条件，决定了开发和维护组织的组织结构，制约着系统的质量属性。

软件架构使推理和控制的更改更加简单，**有助于循序渐进的原型设计**，可以作为培训的基础。

软件架构是_____**的模型**，通过研究软件架构可能预测软件的质量。

软件架构 = 软件体系结构

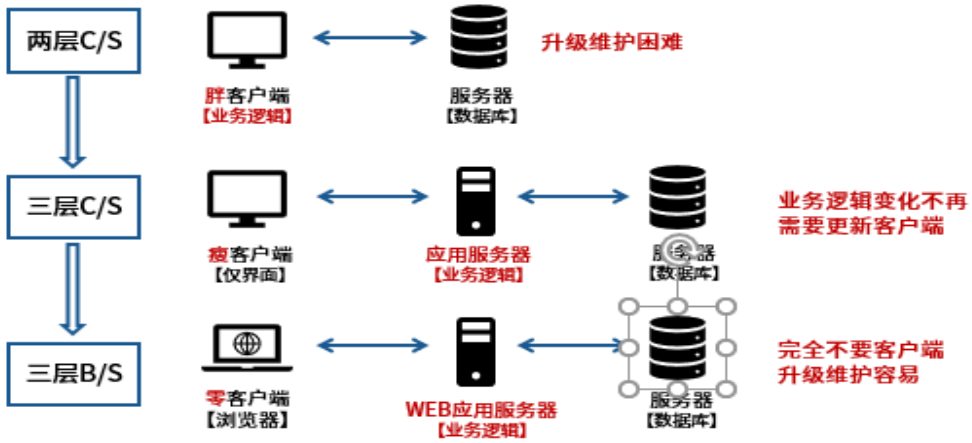
2 软件架构风格

架构风格定义了_____

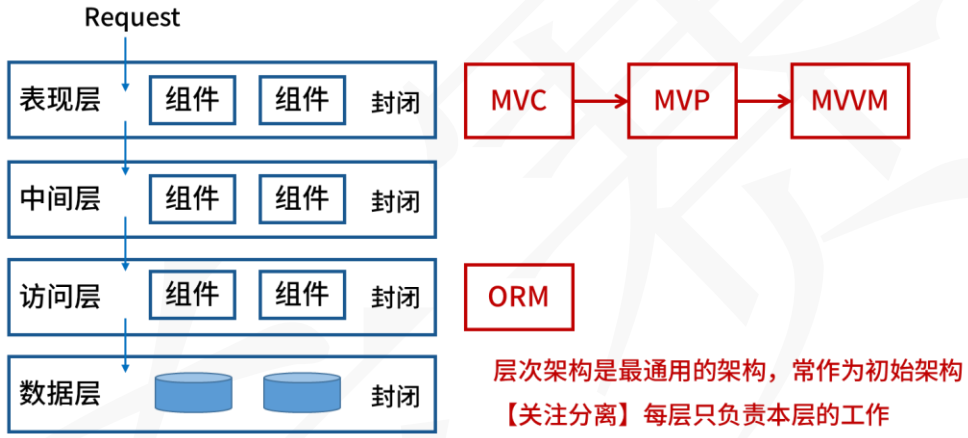
五大架构风格	子风格
数据流风格 【Data Flow】	_____ _____
调用/返回风 【Call/Return】	_____ _____
独立构件风格 【Independent Components】	_____ _____
虚拟机风格 【Virtual Machine】	_____ _____
以数据为中心 【Data-centered】	_____ _____

3 典型架构应用

3.1 层次架构



层次架构是最_____的架构，常作为初始架构，【关注分离】每层只负责本层的工作。



（1）MVC

Model（模型）是_____。通常模型对象负责在数据库中存取数据。

View（视图）是_____。通常视图是依据模型数据创建的。

Controller（控制器）是_____。通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

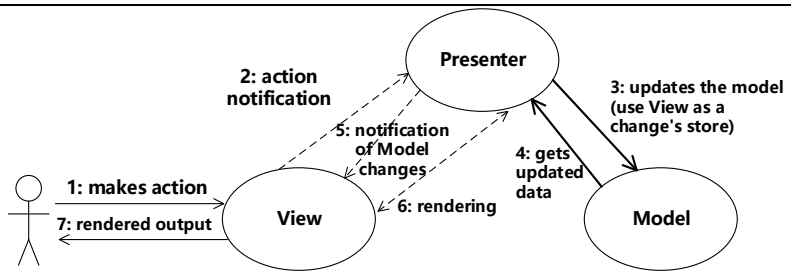
J2EE 体系结构中：

视图（View）：_____

控制（Controller）：_____

模型（Model）：_____

（2）MVP



MVP 与 MVC 关系：**MVP 是 MVC 的变种。**

MVP 的优点：_____

_____

3.2 富互联网应用（RIA）

RIA 结合了_____反应速度快、交互性强的优点，以及 B/S 架构_____

_____的特性。

RIA 简化并改进了 B/S 架构的用户交互。

数据能够被缓存在客户端，从而可以实现一个比基于 HTML 的响应速度更快且数据往返于服务器的次数更少的用户界面。

优点：_____。

3.3 REST

REST 含义：_____

_____

REST 的5个原则：

_____。

_____。

_____。

_____。

_____。

3.4 微服务-混合风格

（1）什么是微服务

_____。

（2）微服务的优势

优点	解读
【_____】	小服务（且专注于做一件事情） 化整为零，易于小团队开发
【_____】	独立开发 独立测试及**独立部署（简单部署）** **独立运行（每个服务运行在其独立进程中）**
【_____】	**支持异构（如：每个服务使用不同数据库）**
【_____】	**故障被隔离在单个服务中**，通过重试、平稳退化等机制实现应用层容错
【_____】	可根据需求独立扩展

（3）微服务面临的挑战

_____【更复杂】

_____【服务间依赖测试】

_____

（4）微服务与 SOA 的对比

微服务	SOA
能拆分的就拆分	_____
_____	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
_____	**粗粒度**
两句话可以解释明白	几百字只相当于 SOA 的目录
_____	**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_____，如 HTTP	企业服务生产总线（ESB）充当了服务之间通信的角色

微服务架构实现	SOA 实现
_____级，自底向上开展实施	企业级，自顶向下开展实施
一个系统被拆分成多个服务，粒度细	服务由_____组成，粒度_____
无集中式总线，松散的服务架构	企业服务总线，集中式的服务架构
集成方式_____（HTTP/REST/JSON）	集成方式_____（ESB/WS/SOAP）
服务能_____	单块架构系统，相互依赖，部署复杂

3.5 云原生架构风格

（1）云计算基本概念：

云计算是_____。

云计算优点：_____。

（2）分类

按服务类型分类：

_____	基于多租户技术实现，直接提供**应用程序**
_____	**虚拟中间件服务器**、运行环境和操作系统
_____	包括**服务器**、存储和网络等服务

按部署方式分类：

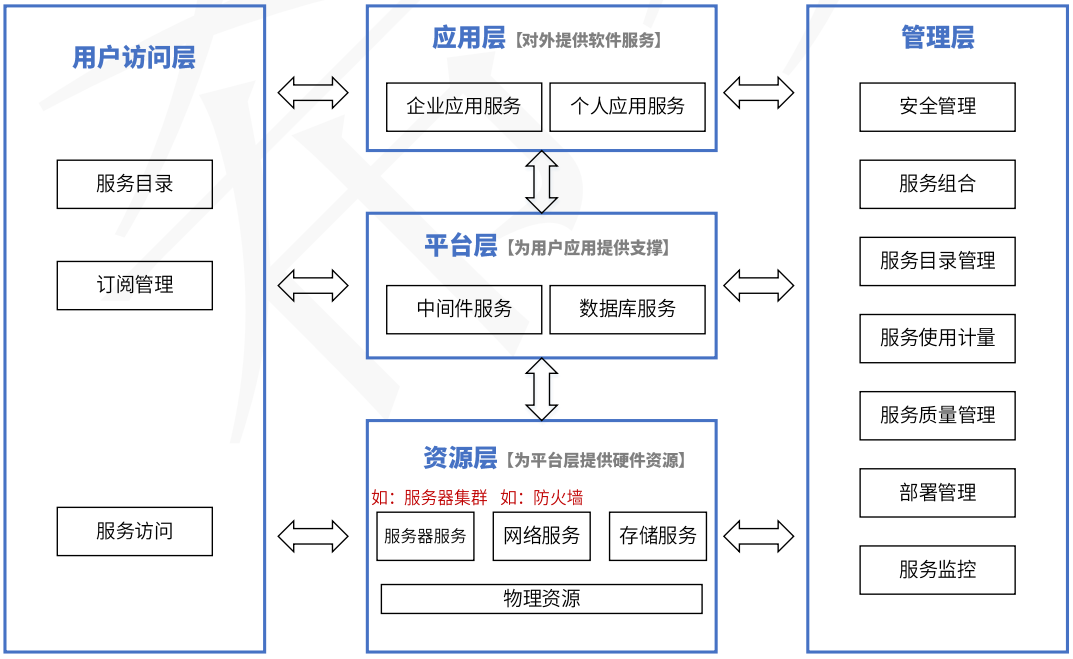
_____：面向互联网用户需求，通过开放网络提供云计算服务

_____：面向企业内部提供云计算服务

_____：兼顾以上两种情况的云计算服务

（3）云计算架构

【云原生】是基于_____的分布式云，以容器、微服务、DevOps 等技术为基础建立的一套云技术产品体系。



【管理层】提供_____。

【用户访问层】方便用户使用云计算服务所需的各种支撑服务，针对每个层次的云计算服务都需要提供相应的访问接口。

【应用层】提供_____，如：财务管理，客户关系管理，商业智能。

【平台层】为用户提供_____，使用户可以构建自己的应用。

【资源层】提供_____，从而隐藏物理资源的复杂性。如：服务器，存储。

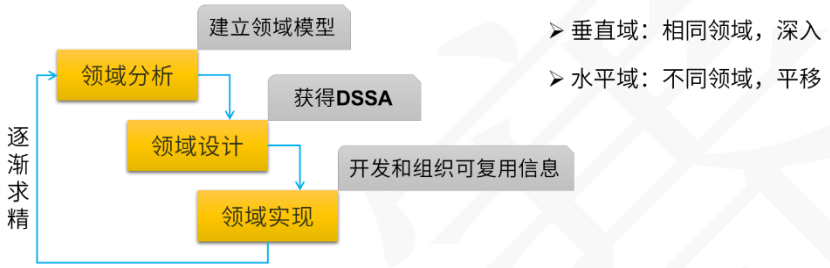
3.6 边缘计算

边缘计算是指_____。

边缘计算的本质：_____。

4 特定领域软件架构（DSSA）

定义：_____。



垂直域：_____。

水平域：_____。

5 基于架构的软件开发方法

（1）基于架构的软件设计（ABSD）

ABSD 能很好的支持软件重用。

ABSD 方法是架构驱动，即强调**由**_____的组合驱动架构设计。

ABSD 方法有三个基础。第一个基础是_____。在功能分解中，ABSD 方法使用已有的基于模块的内聚和耦合技术；第二个基础是_____；第三个基础是_____。软件模板利用了一些软件系统的结构。

视角与视图：**从不同的视角来检查，所以会有不同的视图。**

_____用来捕获功能需求、特定场景**【刺激、环境、响应】用来捕获**_____。

6 架构评估

6.1 架构设计重点关注非功能设计（质量属性）

（1）性能

性能（performance）是指_____，即_____，或者_____。例如：a.同时支持1000并发；b.响应时间小于1s；c.显示分辨率达到4K。

代表参数：_____ 设计策略：_____

（2）可用性

可用性（availability）是_____。例如：a.主服务器故障，1分钟内切换至备用服务器；b.系统故障，1小时内修复；c.系统支持7×24小时工作。

代表参数：_____ 设计策略：_____

（3）安全性

安全性（security）是指_____。安全性又可划分为_____**【信息不泄露给未授权的用户】**、_____**【防止信息被篡改】**、_____**【不可抵赖】**及_____**【对信息的传播及内容具有控制的能力】**等特性。例如：a.可抵御 SQL 注入攻击；b.对计算机的操作都有完整记录；c.用户信息数据库授权必须保证99.9%可用。

设计策略：_____

（4）可修改性

可修改性（modifiability）是指_____。通常以某些具体的变更为基准，通过考察这些变更的代价衡量可修改性。（可扩展性与之相近）例如：a.更改系统报表模块，必须在2人周内完成；b.对 Web 界面风格进行修改，修改必须在4人月内完成。

主要策略：_____

（5）易用性

易用性关注的是_____。例如：a.界面友好；b.新用户学习使用系统时间不超过2小时。

（6）可测试性

软件可测试性是指_____。

6.2 软件架构评估方法

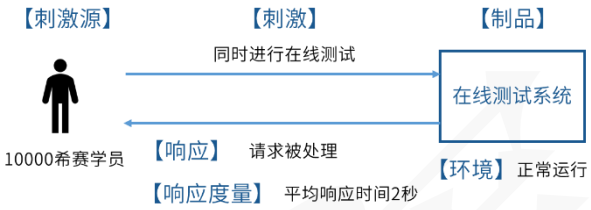
风险点：_____。

非风险点：_____。

敏感点：_____。

权衡点：_____。

场景：_____。场景可从六个方面进行描述：刺激源、**刺激**、制品、环境、**响应**、响应度量。



刺激源（Source）：_____。

刺激（Stimulus）：_____。

环境（Environment）：_____。

制品（Artifact）：_____。

响应（Response）：_____。

响应度量（Measurement）：_____。

7 产品线

7.1 特点

_____。

7.2 建立方式

	演化方式	革命方式
_____	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
_____	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

将现有产品演化为产品线

用软件产品线替代现有产品集

全新软件产品线的演化

全新软件产品线的开发

7.3 成功实施产品线主要取决因素

_____

_____

_____

_____

8 大型网站系统架构演化

8.1 维度

维度	涉及技术内容
从_____来看	MVC，MVP，MVVM，REST，Webservice，微服务
从_____来看	集群（负载均衡）、CDN
从_____来看	MemCache，Redis，Squid
从_____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
从_____来看	Hibernate，Mybatis
从_____来看	Hadoop，FastDFS，区块链
从_____来看	XML，JSON
从_____来看	Apache，WebSphere，WebLogic，Tomcat，JBOSS，IIS
从_____来看	SQL 注入攻击
其它	静态化，有状态与无状态，响应式 Web 设计，中台

8.2 缓存

（1）**MemCache**：MemCache 是_____，用于_____以减轻数据库负载。MemCache 通过在内存里维护一个统一的巨大的 hash 表，能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等。

（2）**Redis**：Redis 是_____，并提供多种语言的 API。

（3）Squid：Squid 是_____，Squid 支持 FTP、gopher、HTTPS 和 HTTP 协议。和一般的代理缓存软件不同，Squid 用一个单独的、非模块化的、I/O 驱动的进程来处理所有的客户端请求。

（4）**Redis 和 MemCache 对比：**

工作	MemCache	Redis
数据类型	简单 key/value 结构	_____
持久性	_____	支持
分布式存储	客户端哈希分片/一致性哈希	_____
多线程支持	支持	_____
内存管理	_____	无
事务支持	_____	有限支持
数据容灾	不支持，不能做数据恢复	_____

（5）Redis 集群切片方式

集群切片方式	核心特点
_____	在客户端通过 key 的 hash 值对应到不同的服务器。
_____	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。
_____	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。

（6）Redis 分布存储方案

分布式存储方案	核心特点
_____	一主多从，**故障时手动切换**。
_____	有哨兵的一主多从，**主节点故障自动选择新的主节点**。
_____	分节点对等集群，分 slots，不同 slots 的信息存储到不同节点。

（7）Redis 分片方案

分片方案	分片方式	说明
_____	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	_____	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。
_____	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。

（8）Redis 持久化

RDB： _____。

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。

对比维度	RDB 持久化	AOF 持久化
备份量		轻量级**增量备份**，一次只保存一个修改命令
保存间隔时间	保存**间隔时间长**	
还原速度		
阻塞情况	save 会阻塞，但 bgsave 或者自动不会阻塞	
数据体积	同等数据体积：	同等数据体积：大
安全性		

（9）淘汰机制

淘汰作用范围	机制名	策略
	noeviction	禁止驱逐数据，内存不足以容纳新入数据时，新写入操作就会报错。系统默认的一种淘汰策略。
	volatile-random	随机移除某个 key
	volatile-lru	优先移除最近未使用的 key
	volatile-ttl	ttl 值小的 key 优先移除
	allkeys-random	随机移除某个 key
	allkeys-lru	优先移除最近未使用的 key

8.3 服务集群

（1）应用层负载均衡

http 重定向： 。

特点： 。

反向代理服务器：在 。常用的 apache，nginx 都可以充当反向代理服务器。

特点： 。

（2）传输层负载均衡

DNS 域名解析负载均衡： 。

特点： 。

基于 NAT 的负载均衡：_____。

特点：_____。

（3）硬件负载均衡：F5

（4）软件负载均衡：_____

（5）算法分类

请列举出3种静态算法（不考虑动态负载）：

_____；

_____；

_____。

请列举出3种动态算法（考虑动态负载）

_____；

_____；

_____。

Session 有状态和无状态问题

_____（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。

_____（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。

8.4 数据库读写分离

主从数据库结构特点：

一般：一主多从，也可以多主多从。

主库做_____，**从库做**_____。

请列出主从复制步骤：

8.5 响应式 Web 设计

（1）概念

响应式 WEB 设计是一种网络页面设计布局，其理念是：_____

_____。

（2）方法与策略

采用流式布局和弹性化设计：_____

_。

响应式图片：_____。

8.6 中台

概念：_____

_____

_____。中台又可以进一步细分，比如业务中台，数据中台，XX 中台。本质上，都是对企业通用能力在不同层面的沉淀，并对外能力开放。

业务中台：提供_____，例如学员中心、课程中心之类的开箱即用可重用能力。

数据中台：提供_____，帮助企业从数据中学习改进，调整方向。

技术中台：提供_____，帮助解决基础技术平台的复用。如：中间件，分布式存储，AI，负载均衡等基础设施。

数据中台必备的4个核心能力

1、_____

2、_____

3、_____

4、_____

第二章 项目管理

1 进度网络图-关键路径法（PERT）

（1）总时差（松弛时间）：

含义：_____。活动的总时差等于该活动最迟完成时间与最早完成时间之差，或该活动最迟开始时间与最早开始时间之差。

（2）自由时差：

含义：_____。

对于有紧后活动的活动，其自由时差等于_____。

对于没有紧后活动的活动，也就是以网络计划终点节点为完成节点的活动，其自由时差等于_____。

对于网络计划中以终点节点为完成节点的活动，其自由时差与总时差相等。此外，由于活动的自由时差是其总时差的构成部分，所以，当活动的总时差为_____时，其自由时差必然为零，可不必进行专门计算。

2 软件质量管理

2.1 软件质量控制与质量保证

（1）**质量保证**一般是每隔一定时间（例如，每个阶段末）进行的，主要通过系统的质量审计和过程分析来保证项目的质量。独特工具包括：_____。

（2）**质量控制**是实时监控项目的具体结果，以判断它们是否符合相关质量标准，制定有效方案，以消除产生质量问题的原因。

（3）**质量保证的主要目标**

【事前预防】工作。

尽量在刚刚引入缺陷时即将其捕获，而不是让缺陷扩散到下一个阶段。

作用于_____而不是_____。

贯穿于_____，而不是只集中于一点。

3 软件配置管理

（1）**产品配置是指**_____。

（2）关于配置项

_____**（可交付成果）：**需求文档、设计文档、源代码、可执行代码测试用例、运行软件所需数据等

_____**：各类计划**（如项目管理计划，进度管理计划）、各类报告

软件配置管理核心内容包括【_____】和【_____】。

按软件过程活动将软件工具分为：

_____：需求分析工具、设计工具、编码与排错工具。

_____：版本控制工具（VSS、CVS、SCCS、SVN）、文档分析工具、开发信息库工具、逆向工程工具、再工程工具。

软件管理和软件支持工具：_____

_____。

第三章 计算机系统基础知识

1 存储系统

时间局部性：_____。

空间局部性：_____。

工作集理论：_____。

2 操作系统概述

2.1 特殊的操作系统

分类	特点
_____	单道批：一次一个作业入内存，作业由程序、数据、作业说明书组成 多道批：一次多个作业入内存，特点：多道、宏观上并行微观上串行
分时操作系统	采用时间片轮转的方式为多个用户提供服务，每个用户感觉独占系统 特点：_____
实时操作系统	实时控制系统和实时信息系统 _____要求不高，_____要求高（**规定时间内响应并处理**）
网络操作系统	方便有效共享网络资源，提供服务软件和有关协议的集合 主要的网络操作系统有：_____
_____	任意两台计算机可以通过通信交换信息 是网络操作系统的更高级形式，具有透明性、可靠性和高性能等特性
微机操作系统	Windows：Microsoft 开发的图形用户界面、多任务、多线程操作系统 Linux：免费使用和自由传播的类 Unix 操作系统，多用户、多任务、多线程和多 CPU 的操作系统
嵌入式操作系统	运行在智能芯片环境中 特点：_____

2.2 存储管理

（1）页式存储：将程序与内存均划分为同样大小的块，以页为单位将程序调入内存。

优点：_____

缺点：_____

（2）段式存储：按用户作业中的自然段来划分逻辑空间，然后调入内存，段的长度可以不一样。

优点：_____

缺点：_____

（3）段页式存储：段式与页式的综合体。先分段，再分页。1个程序有若干个段，每个段中可以有若干页，每个页的大小相同，但每个段的大小不同。

优点：_____

缺点：_____

2.3 磁盘管理

（1）存取时间=_____，寻道时间是指磁头移动到磁道所需的时间；等待时间为_____。

（2）读取磁盘数据的时间应包括以下三个部分：

_____。

_____。

_____。

（3）磁盘移臂调度算法：

_____ FCFS（谁先申请先服务谁）；

最短寻道时间优先 SSTF（申请时判断与磁头当前位置的距离，谁短先服务谁）；

_____ SCAN（电梯算法，双向扫描）；

循环扫描 CSCAN（单向扫描）。

3 文件系统

（1）文件（File）是具有符号名的、在逻辑上具有完整意义的一组相关信息项的集合，例如，一个源程序、一个目标程序、编译程序、一批待加工的数据和各种文档等都可以各自组成一个文件。

一个文件包括文件体和文件说明。

- 文件体是文件真实的内容；
- 文件说明是操作系统为了管理文件所用到的信息，包括文件名、文件内部标识、文件类型、文件存储地址、文件长度、访问权限、建立时间和访问时间等。

（2）文件的类型

按文件的性质和用途分类可将文件分为_____。

按信息保存期限分类可将文件分为_____。

按文件的保护方式分类可将文件分为_____。

4 性能评估

方法	描述	特点
时钟频率法	以时钟频率高低衡量速度	仅考虑 CPU
指令执行速度法	表示机器运算速度的单位是 MIPS	仅考虑 CPU
等效指令速度法（吉普森混合法）	_____	仅考虑 CPU，综合考虑指令比例不同的问题。
数据处理速率法（PDR）	PDR 值的方法来衡量机器性能，PDR 值越大，机器性能越好。	PDR = L/R 仅考虑 CPU+存储
_____（CTP）	CTP 用 MTOPS 表示。CTP 的估算方法是，首先算出处理部件每个计算单元的有效计算率，再按不同字长加以调整，得出该计算单元的理论性能，所有组成该处理部件的计算单元的理论性能之和即为 CTP。	仅考虑 CPU+存储
基准程序法	_____	综合考虑多部分，基准程序法是目前一致承认的测试系统性能的较好方法。

【测试精确度排名】 真实的程序 > _____ **>** _____ **> 合成基准程序**

常见的 Web 服务器性能评测方法有 _____。

系统监视：进行系统监视通常有3种方式：一是通过_____，如 UNIX/Linux 系统中的 W、ps、last，Windows 中的 netstat 等；二是_____；三是_____，如 **Windows 的 Perfmon 应用程序。**

第四章 嵌入式系统

1 嵌入式系统概述

1.1 基本概念

（1）嵌入式系统是以_____为中心、以_____为基础，并将可配置与可裁剪的软、硬件集成于一体的专用计算机系统，需要满足应用对功能、可靠性、成本、体积和功耗等方面的严格要求。

（2）从计算机角度看，嵌入式系统是指嵌入各种设备及应用产品内部的计算机系统。它主要完成信号控制的功能，体积小、结构紧凑，可作为一个部件埋藏于所控制的装置中。

（3）一般嵌入式系统由_____组成。

嵌入式系统初始化过程：片级初始化→_____初始化→系统级初始化

1.2 请列出嵌入式系统组成部件

_____

_____

_____

_____

2 嵌入式硬件

2.1 嵌入式微处理器分类

通常嵌入式处理器的选择还要根据使用场景不同选择不同类型的处理器，从处理器分类看，大致可分为 MPU、MCU、DSP、GPU、SoC

（1）_____：将微处理器装配在专门设计的电路板上，只保留与嵌入式应用有关的母板功能。微处理器一般以某一种微处理内核为核心，每一种衍生产品的处理器内核都是一样的，不同的是存储器和外设的配置和封装。

（2）_____（Micro Control Unit，MCU）：又称单片机。与 MPU 相比 MCU 的最大优点在于单片化，体积大大减小，从而使功耗和成本下降，可靠性提高。

（3）_____**（Digital Signal Processor，DSP）：**DSP 处理器对系统结构和指令进行了特殊设计（通常，DSP 采用一种哈佛结构），使其适合于执行 DSP 算法，编译效率高，指令执行速度也高。

（4）**图形处理器（Graphics Processing Unit，GPU）：**

GPU 是图形处理单元的缩写，是一种_____图形等图像的半导体芯片（处理器）。

GPU 可用于个人电脑、工作站、游戏机和一些移动设备上做图像和图形相关运算工作的微处理器。

它可减少对_____的依赖，并进行部分原本 CPU 的工作，尤其是在3D 图形处理中，GPU 采用了核心技术（如：硬件 T&L、纹理压缩等）保证了快速3D 渲染能力。

GPU 目前已广泛应用于各行各业，GPU 中集成了同时运行在 GHz 的频率上的成千上万个 core，可以高速处理图像数据。**最新的 GPU 峰值性能可高达100 TFlops 以上。**

（5）片上系统（System on Chip，SoC）：

追求产品系统最大包容的集成器件。

它是一个产品，是一个**有**_____**的集成电路**，其中包含完整系统并有嵌入软件的全部内容。

同时它**又是一种**_____，用以实现从确定系统功能开始，到软/硬件划分，并完成设计的整个过程。

成功实现了_____**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。

2.2 AI 芯片

人工智能（Artificial Intelligence，AI）芯片的定义：从广义上讲只要能够运行人工智能算法的芯片都叫作 AI 芯片。但是通常意义上的 AI 芯片指的是针对人工智能算法做了特殊加速设计的芯片，现阶段，这些人工智能算法一般以深度学习算法为主，也可以包括其它机器学习算法。

人工智能芯片四大类（按技术架构**分类**）：

- _____
- _____（现场可编程门阵列）
- _____（专用集成电路）
- 类脑芯片

AI 芯片的关键特征：

_____：AI 计算既不脱离传统计算，也具有**新的计算特质**

训练和推断：_____

_____：满足高效能机器学习的数据处理要求

数据精度：**降低精度**的设计

可重构的能力：_____

_____

开发工具：AI 芯片需要**软件工具链的支持**

2.3 嵌入式微处理器体系结构

体系结构分类	定义	特点	典型应用
冯·诺依曼结构	_____ _____ _____ _____ _____	指令与数据存储器合并在一起。 指令与数据都通过相同的_____传输。	一般用于 PC 处理器，如 I3、I5、I7处理器。 **注：_____属于冯·诺依曼结构**
哈佛结构	_____ _____ _____ _____	指令与数据分开存储，可以_____读取，有较高数据的吞吐率。 有___条总线：指令和数据的数据总线与地址总线。	一般用于嵌入式系统处理器。 **注：_____属于哈佛结构**

2.4 总线

总线的基本概念：_____。

特点：

挂接在总线上的多个部件只能分时向总线发送数据，但可同时从总线接收数据。

通过总线复用方式可以减少总线中信号线的数量，以较少的信号线传输更多的信息。

总线分类：

（1）从功能上来对总线进行划分：_____

（2）从数据传输的方式划分为_____

3 嵌入式操作系统

3.1 嵌入式操作系统的定义及特点

嵌入式操作系统（Embedded Operating System，EOS）是指_____。嵌入式操作系统是一种用途广泛的系统软件，负责嵌入式系统的全部软、硬件资源分配、任务调度、控制、协调并行活动等工作。通常包括与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面、标准化浏览器等。

根据系统对时间的敏感程度可将嵌入式系统划分为：

（1）_____

（2）_____：**能够在指定或者确定的时间内**完成系统功能和外部或内部、同步或异步时间做出响应的系统。

嵌入式操作系统具有一般操作系统的功能，同时具有嵌入式软件的特点，主要有：

_____

_____。

实时操作系统的最核心特点是_____强。

嵌入式实时操作系统实时性的评价指标

- □ _____
- □ _____
- □ _____

3.2 嵌入式实时操作系统调度算法

_____：系统为每个任务分配一个**相对固定**的优先顺序。

抢占式优先级调度算法：_____

_____。

_____（**EDF 算法**）：根据任务的截止时间头**端**来确定其优先级，对于时间期限最近的任务，分配最高的优先级。

最晚截止期调度算法：_____

_____。

3.3 操作系统内核架构

内核是操作系统的核心部分，它管理着系统的各种资源。内核可以看成连接应用程序和硬件的一座桥梁，是直接运行在硬件上的最基础的软件实体。

目前从内核架构来划分，可分为_____（Monolithic Kernel）和_____（Micro Kernel）。

第五章 新技术专题

1 大数据架构

对比内容	Lambda 架构	Kappa 架构
复杂度与开发、维护成本		
计算开销		
实时性		
历史数据处理能力		
使用场景		
选择依据		

2 边云协同

边云协同放大边缘计算与云计算的应用价值，主要包括六种协同：_____、_____、_____、_____、_____、_____。

答案

第一章 软件架构设计

1 软件架构的概念

架构设计就是需求分配，即将满足需求的职责分配到组件上。

软件架构风格是描述某一特定应用领域中系统组织方式的惯用模式。架构风格定义一个系统家族，即一个体系结构定义一个词汇表和一组约束。词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构件和连接件组合起来的。

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，由构成系统的元素的描述、这些元素的相互作用、指导元素集成的模式以及这些模式的约束组成。

架构的本质：

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。

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。

架构的作用：

软件架构是项目干系人进行交流的手段，明确了对系统实现的约束条件，决定了开发和维护组织的组织结构，制约着系统的质量属性。

软件架构使推理和控制的更改更加简单，有助于循序渐进的原型设计，可以作为培训的基础。

软件架构是可传递和可复用的模型，通过研究软件架构可能预测软件的质量。

软件架构 = 软件体系结构

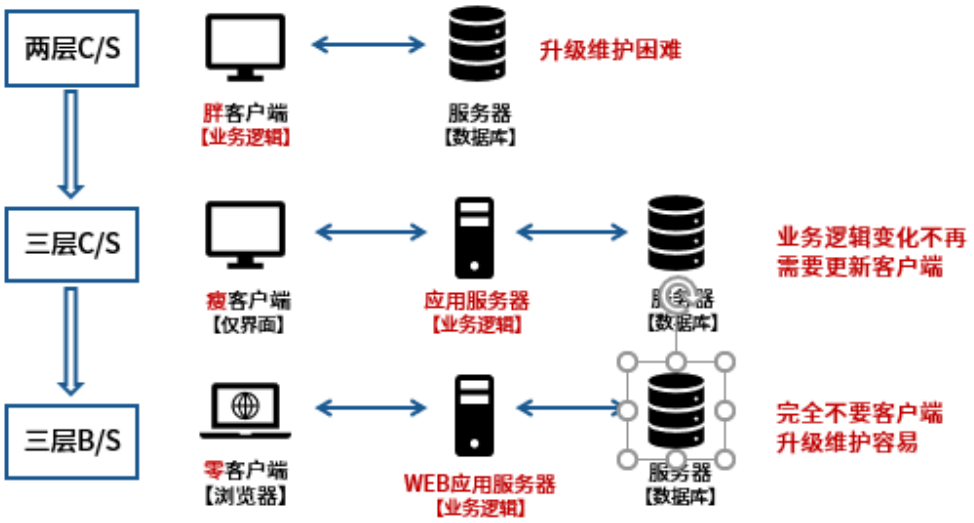
2 软件架构风格

架构风格定义了用于描述系统的术语表和一组指导构建系统的规则

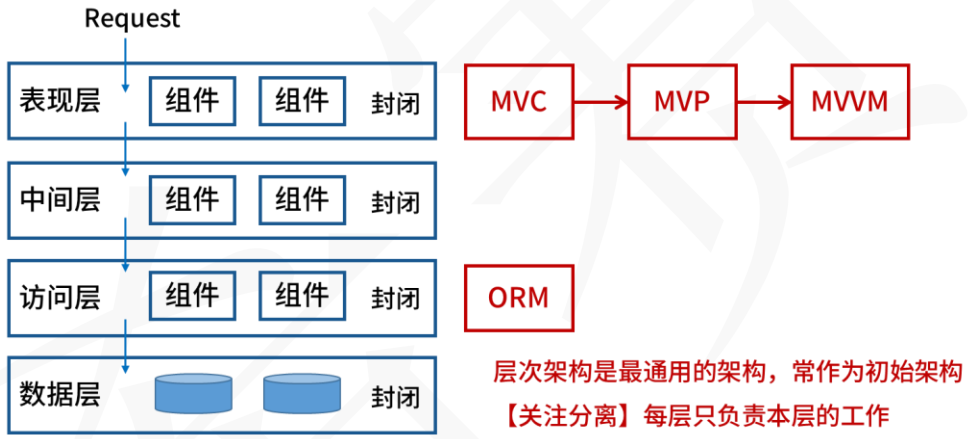
五大架构风格	子风格
数据流风格 【Data Flow】	批处理【Batch Sequential】、管道-过滤器【Pipes and Filters】
调用/返回风 【Call/Return】	主程序/子程序【Main Program and Subroutine】 面向对象【Object-oriented】、分层架构【Layered System】
独立构件风格 【Independent Components】	进程通信【Communicating Processes】、 事件驱动系统（隐式调用）【Event system】
虚拟机风格 【Virtual Machine】	解释器【interpreter】、规则系统【Rule-based System】
以数据为中心 【Data-centered】	数据库系统【Database System】、黑板系统【Blackboard System】、超文本系统【Hypertext System】

3 典型架构应用

3.1 层次架构



层次架构是最通用的架构，常作为初始架构，【关注分离】每层只负责本层的工作。



（1）MVC

Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据。

View（视图）是应用程序中处理数据显示的部分。通常视图是依据模型数据创建的。

Controller（控制器）是应用程序中处理用户交互的部分。通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

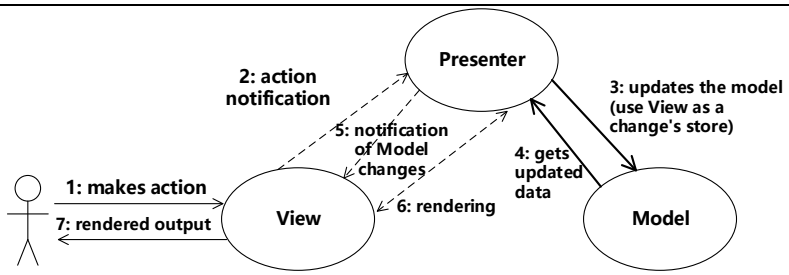
J2EE 体系结构中：

视图（View）：JSP

控制（Controller）：Servlet

模型（Model）：Entity Bean、Session Bean

（2）MVP



MVP 与 MVC 关系：MVP 是 MVC 的变种。

MVP 的优点：

模型与视图完全分离，我们可以修改视图而不影响模型。

可以更高效地使用模型，因为所有的交互都发生在一个地方——Presenter 内部。

我们可以将一个 Presenter 用于多个视图，而不需要改变 Presenter 的逻辑。这个特性非常的有用，因为视图的变化总是比模型的变化频繁。

如果我们把逻辑放在 Presenter 中，那么我们就可以脱离用户接口来测试这些逻辑（单元测试）。

3.2 富互联网应用（RIA）

RIA 结合了 C/S 架构反应速度快、交互性强的优点，以及 B/S 架构传播范围广及容易传播的特性。

RIA 简化并改进了 B/S 架构的用户交互。

数据能够被缓存在客户端，从而可以实现一个比基于 HTML 的响应速度更快且数据往返于服务器的次数更少的用户界面。

优点：反应速度快、易于传播、交互性强。

3.3 REST

REST 含义：REST（Representational State Transfer，表述性状态转移）是一种通常使用 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5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。

3.4 微服务-混合风格

（1）什么是微服务

微服务顾名思义，就是很小的服务，所以它属于面向服务架构的一种。

（2）微服务的优势

优点	解读
【复杂应用解耦】	小服务（且专注于做一件事情） 化整为零，易于小团队开发
【独立】	独立开发 独立测试及**独立部署（简单部署）** **独立运行（每个服务运行在其独立进程中）**
【技术选型灵活】	**支持异构（如：每个服务使用不同数据库）**
【容错】	**故障被隔离在单个服务中**，通过重试、平稳退化等机制实现应用层容错
【松耦合，易扩展】	可根据需求独立扩展

（3）微服务面临的挑战

分布式环境下的数据一致性【更复杂】

测试的复杂性【服务间依赖测试】

运维的复杂性

（4）微服务与 SOA 的对比

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	企业服务产总线（ESB）充当了服务之间通信的角色
微服务架构实现	SOA 实现
团队级，自底向上开展实施	企业级，自顶向下开展实施
一个系统被拆分成多个服务，粒度细	服务由多个子系统组成，粒度大
无集中式总线，松散的服务架构	企业服务总线，集中式的服务架构
集成方式简单（HTTP/REST/JSON）	集成方式复杂（ESB/WS/SOAP）
服务能独立部署	单块架构系统，相互依赖，部署复杂

3.5 云原生架构风格

（1）云计算基本概念：

云计算是集合了大量计算设备和资源，对用户屏蔽底层差异的分布式处理架构，其用户与提供实际服务的计算资源是相分离的。

云计算优点：超大规模、虚拟化、高可靠性、高可伸缩性、按需服务、成本低【前期投入低、综合使用成本也低】。

（2）分类

按服务类型分类：

SaaS【软件即服务】	基于多租户技术实现，直接提供**应用程序**
PaaS【平台即服务】	**虚拟中间件服务器**、运行环境和操作系统
IaaS【基础设施即服务】	包括**服务器**、存储和网络等服务

按部署方式分类：

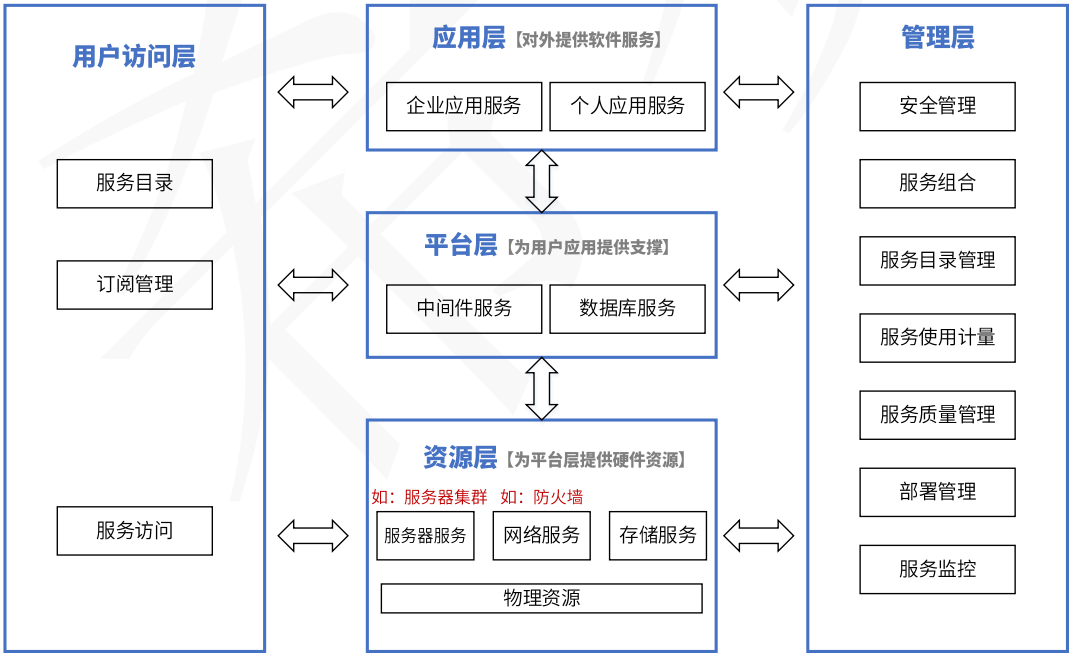
公有云：面向互联网用户需求，通过开放网络提供云计算服务

私有云：面向企业内部提供云计算服务

混合云：兼顾以上两种情况的云计算服务

（3）云计算架构

【云原生】是基于分布部署和统一运管的分布式云，以容器、微服务、DevOps 等技术为基础建立的一套云技术产品体系。



【管理层】提供对所有层次云计算服务的管理功能。

【用户访问层】方便用户使用云计算服务所需的各种支撑服务，针对每个层次的云计算服务都需要提供相应的访问接口。

【应用层】提供软件服务，如：财务管理，客户关系管理，商业智能。

【平台层】为用户提供对资源层服务的封装，使用户可以构建自己的应用。

【资源层】提供虚拟化的资源，从而隐藏物理资源的复杂性。如：服务器，存储。

3.6 边缘计算

边缘计算是指在靠近物或数据源头的一侧，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供最近端服务。

边缘计算的本质：计算处理职能的本地化。

4 特定领域软件架构（DSSA）

定义：特定领域软件架构以一个特定问题领域为对象，形成由领域参考模型、参考需求、参考架构等组成的开发基础架构，支持一个特定领域中多个应用的生成。

垂直域：某一个狭小领域或者说某个行业的共性抽象。

水平域：多个行业可通用的一些共性的抽象。

5 基于架构的软件开发方法

（1）基于架构的软件设计（ABSD）

ABSD 能很好的支持软件重用。

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

ABSD 方法有三个基础。第一个基础是功能的分解。在功能分解中，ABSD 方法使用已有的基于模块的内聚和耦合技术；第二个基础是通过选择架构风格来实现质量和业务需求；第三个基础是软件模板的使用。软件模板利用了一些软件系统的结构。

视角与视图：从不同的视角来检查，所以会有不同的视图。

用例用来捕获功能需求、特定场景【刺激、环境、响应】用来捕获质量需求。

6 架构评估

6.1 架构设计重点关注非功能设计（质量属性）

（1）性能

性能（performance）是指系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段时间内系统所能处理的事件的个数。例如：a.同时支持1000并发；b.响应时间小于1s；c.显示分辨率达到4K。

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

可用性（availability）是系统能够正常运行的时间比例。经常用两次故障之间的时间长度或在出现故障时系统能够恢复正常的速度来表示。例如：a.主服务器故障，1分钟内切换至备用服务器；b.系统故障，1小时内修复；c.系统支持7×24小时工作。

代表参数：故障间隔时间　　设计策略：冗余、心跳线

（3）安全性

安全性（security）是指系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。安全性又可划分为机密性【信息不泄露给未授权的用户】、完整性【防止信息被篡改】、不可否认性【不可抵赖】及可控性【对信息的传播及内容具有控制的能力】等特性。例如：a.可抵御 SQL 注入攻击；b.对计算机的操作都有完整记录；c.用户信息数据库授权必须保证99.9%可用。

设计策略：追踪审计

（4）可修改性

可修改性（modifiability）是指能够快速地以较高的性能价格比对系统进行变更的能力。通常以某些具体的变更为基准，通过考察这些变更的代价衡量可修改性。（可扩展性与之相近）例如：a.更改系统报表模块，必须在2人周内完成；b.对 Web 界面风格进行修改，修改必须在4人月内完成。

主要策略：信息隐藏（二义性：良好的封装能够做到信息隐藏，一般归于可修改性策略；信息隐藏也能够体现在安全性当中）

（5）易用性

易用性关注的是对用户来说完成某个期望任务的容易程度和系统所提供的用户支持的种类。例如：a.界面友好；b.新用户学习使用系统时间不超过2小时。

（6）可测试性

软件可测试性是指通过测试揭示软件缺陷的容易程度。例如：a.提供远程调试接口，支持远程调试。

6.2 软件架构评估方法

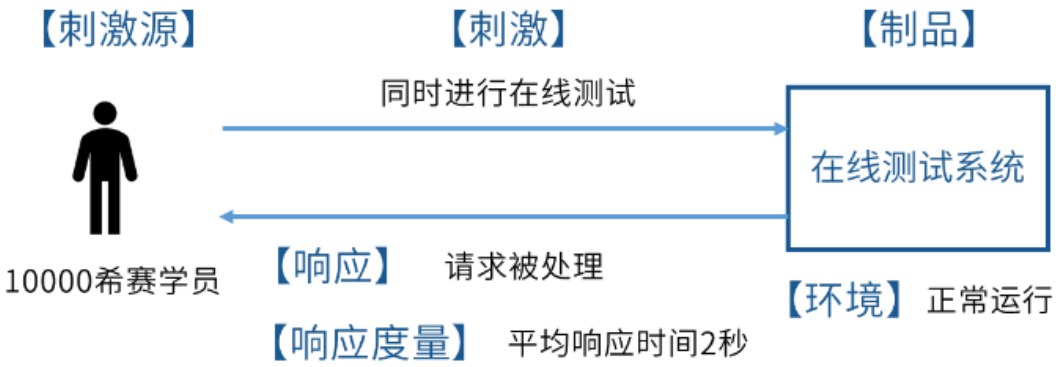
风险点：系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。

非风险点：是指不会带来隐患，一般以“XXX 要求是可以实现【或接受】的”方式表达。

敏感点：敏感点是一个或多个构件（和/或构件之间的关系）的特性，它能影响系统的某个质量属性。

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。

场景：场景是从风险承担者的角度与系统交互的简短描述。场景可从六个方面进行描述：刺激源、刺激、制品、环境、响应、响应度量。



刺激源（Source）：这是某个生成该刺激的实体（人、计算机系统或者任何其他刺激器）。

刺激（Stimulus）：该刺激是当刺激到达系统时需要考虑的条件。

环境（Environment）：该刺激在某些条件内发生。当激励发生时，系统可能处于过载、 运行或者其他情况。

制品（Artifact）：某个制品被激励。这可能是整个系统，也可能是系统的一部分。

响应（Response）：该响应是在激励到达后所采取的行动。

响应度量（Measurement）：当响应发生时，应当能够以某种方式对其进行度量，以对需求进行测试。

7 产品线

7.1 特点

核心资源、产品集合、过程驱动、特定领域、技术支持、以架构为中心。

7.2 建立方式

	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

将现有产品演化为产品线

用软件产品线替代现有产品集

全新软件产品线的演化

全新软件产品线的开发

7.3 成功实施产品线主要取决因素

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

8 大型网站系统架构演化

8.1 维度

维度	涉及技术内容
从架构来看	MVC，MVP，MVVM，REST，Webservice，微服务
从并发分流来看	集群（负载均衡）、CDN
从缓存来看	MemCache，Redis，Squid
从数据来看	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
从持久化来看	Hibernate，Mybatis
从分布存储来看	Hadoop，FastDFS，区块链
从数据编码来看	XML，JSON
从 Web 应用服务器来看	Apache，WebSphere，WebLogic，Tomcat，JBOSS，IIS
从安全性来看	SQL 注入攻击
其它	静态化，有状态与无状态，响应式 Web 设计，中台

8.2 缓存

（1）MemCache：MemCache 是一个高性能的分布式的内存对象缓存系统，用于动态 Web 应用以减轻数据库负载。MemCache 通过在内存里维护一个统一的巨大的 hash 表，能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等。

（2）Redis：Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API。

（3）Squid：Squid 是一个高性能的代理缓存服务器，Squid 支持 FTP、gopher、HTTPS 和 HTTP 协议。和一般的代理缓存软件不同，Squid 用一个单独的、非模块化的、I/O 驱动的进程来处理所有的客户端请求。

（4）Redis 和 MemCache 对比：

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	支持（Redis5.0 及以前版本不支持）
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

（5）Redis 集群切片方式

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。

（6）Redis 分布存储方案

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。

（7）Redis 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。

（8）Redis 持久化

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。

对比维度	RDB 持久化	AOF 持久化
备份量	重量级的**全量备份**，保存整个数据库	轻量级**增量备份**，一次只保存一个修改命令
保存间隔时间	保存**间隔时间长**	保存间隔时间短，默认 1 秒
还原速度	数据还原速度快	数据还原速度慢
阻塞情况	save 会阻塞，但 bgsave 或者自动不会阻塞	无论是平时还是 AOF 重写，都不会阻塞
数据体积	同等数据体积：小	同等数据体积：大
安全性	数据安全性：**低**，容易丢数据	数据安全性：**高**，根据策略决定

（9）淘汰机制

淘汰作用范围	机制名	策略
不淘汰	noeviction	禁止驱逐数据，内存不足以容纳新入数据时，新写入操作就会报错。系统默认的一种淘汰策略。
设置了过期时间的键空间	volatile-random	随机移除某个 key
	volatile-lru	优先移除最近未使用的 key
	volatile-ttl	ttl 值小的 key 优先移除
全键空间	allkeys-random	随机移除某个 key
	allkeys-lru	优先移除最近未使用的 key

8.3 服务集群

（1）应用层负载均衡

http 重定向。HTTP 重定向就是应用层的请求转发。用户的请求其实已经到了 HTTP 重定向负载均衡服务器，服务器根据算法要求用户重定向，用户收到重定向请求后，再次请求真正的集群。

特点：实现简单，但性能较差。

反向代理服务器。在用户的请求到达反向代理服务器时（已经到达网站机房），由反向代理服务器根据算法转发到具体的服务器。常用的 apache，nginx 都可以充当反向代理服务器。

特点：部署简单，但代理服务器可能成为性能的瓶颈。

（2）传输层负载均衡

DNS 域名解析负载均衡。DNS 域名解析负载均衡就是在用户请求 DNS 服务器，获取域名对应的 IP 地址时，DNS 服务器直接给出负载均衡后的服务器 IP。

特点：效率比 HTTP 重定向高，减少维护负载均衡服务器成本。但一个应用服务器故障，不能及时通知 DNS，而且 DNS 负载均衡的控制权在域名服务商那里，网站无法做更多的改善和更强大的管理。

基于 NAT 的负载均衡。基于 NAT 的负载均衡将一个外部 IP 地址映射为多个 IP 地址，对每次连接请求动态地转换为一个内部节点的地址。

特点：技术较为成熟，一般在网关位置，可以通过硬件实现。像四层交换机一般就采用了这种技术。

（3）硬件负载均衡：F5

（4）软件负载均衡：LVS、Nginx、HAproxy

（5）算法分类

静态算法（不考虑动态负载）：

（1）轮转算法：轮流将服务请求（任务）调度给不同的节点（即：服务器）。

（2）加权轮转算法：考虑不同节点处理能力的差异。

（3）源地址哈希散列算法：根据请求的源 IP 地址，作为散列键从静态分配的散列表找出对应的节点。

（4）目标地址哈希散列算法：根据请求目标 IP 做散列找出对应节点。

（5）随机算法：随机分配，简单，但不可控。

动态算法（考虑动态负载）

（1）最小连接数算法：新请求分配给当前活动请求数量最少的节点，每个节点处理能力相同的情况下。

（2）加权最小连接数算法：考虑节点处理能力不同，按最小连接数分配。

（3）加权百分比算法：考虑了节点的利用率、硬盘速率、进程个数等，使用利用率来表现剩余处理能力。

Session 有状态和无状态问题

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。

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。

8.4 数据库读写分离

主从数据库结构特点：

一般：一主多从，也可以多主多从。

主库做写操作，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（Slave）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

8.5 响应式 Web 设计

（1）概念

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。

（2）方法与策略

采用流式布局和弹性化设计：使用相对单位，设定百分比而非具体值的方式设置页面元素的大小。

响应式图片：不仅要同比的缩放图片，还要在小设备上降低图片自身的分辨率。

8.6 中台

概念：中台是一套结合互联网技术和行业特性，将企业核心能力以共享服务形式沉淀，形成“大中台、小前台”的组织和业务机制，供企业快速低成本的进行业务创新的企业架构。中台又可以进一步细分，比如业务中台，数据中台，XX 中台。本质上，都是对企业通用能力在不同层面的沉淀，并对外能力开放。

业务中台：提供重用服务，例如学员中心、课程中心之类的开箱即用可重用能力。

数据中台：提供数据整合分析能力，帮助企业从数据中学习改进，调整方向。

技术中台：提供技术重用组件能力，帮助解决基础技术平台的复用。如：中间件，分布式存储，AI，负载均衡等基础设施。

数据中台必备的4个核心能力

1、数据汇聚整合能力

2、数据提纯加工能力

3、数据服务可视化

4、价值变现方面

第二章 项目管理

1 进度网络图-关键路径法（PERT）

（1）总时差（松弛时间）：

在不延误总工期的前提下，该活动的机动时间。活动的总时差等于该活动最迟完成时间与最早完成时间之差，或该活动最迟开始时间与最早开始时间之差。

（2）自由时差：

在不影响紧后活动的最早开始时间前提下，该活动的机动时间。

对于有紧后活动的活动，其自由时差等于所有紧后活动最早开始时间减本活动最早完成时间所得之差的最小值。

对于没有紧后活动的活动，也就是以网络计划终点节点为完成节点的活动，其自由时差等于计划工期与本活动最早完成时间之差。

对于网络计划中以终点节点为完成节点的活动，其自由时差与总时差相等。此外，由于活动的自由时差是其总时差的构成部分，所以，当活动的总时差为零时，其自由时差必然为零，可不必进行专门计算。

2 软件质量管理

2.1 软件质量控制与质量保证

（1）质量保证一般是每隔一定时间（例如，每个阶段末）进行的，主要通过系统的质量审计和过程分析来保证项目的质量。独特工具包括：质量审计和过程分析。

（2）质量控制是实时监控项目的具体结果，以判断它们是否符合相关质量标准，制定有效方案，以消除产生质量问题的原因。

（3）质量保证的主要目标

【事前预防】工作。

尽量在刚刚引入缺陷时即将其捕获，而不是让缺陷扩散到下一个阶段。

作用于【过程】而【不是最终产品】。

贯穿于【所有的活动之中】，而不是只集中于一点。

3 软件配置管理

（1）产品配置是指一个产品在其生命周期各个阶段所产生的各种形式（机器可读或人工可读）和各种版本的文档、计算机程序、部件及数据的集合。

（2）关于配置项

基线配置项（可交付成果）：需求文档、设计文档、源代码、可执行代码测试用例、运行软件所需数据等

非基线配置项：各类计划（如项目管理计划，进度管理计划）、各类报告

软件配置管理核心内容包括【版本控制】和【变更控制】。

按软件过程活动将软件工具分为：

软件开发工具：需求分析工具、设计工具、编码与排错工具。

软件维护工具：版本控制工具（VSS、CVS、SCCS、SVN）、文档分析工具、开发信息库工具、逆向工程工具、再工程工具。

软件管理和软件支持工具：项目管理工具、配置管理工具、软件评价工具、软件开发工具的评价和选择。

第三章 计算机系统基础知识

1 存储系统

时间局部性：指程序中的某条指令一旦执行，不久以后该指令可能再次执行，典型原因是由于程序中存在着大量的循环操作。

空间局部性：指一旦程序访问了某个存储单元，不久以后，其附近的存储单元也将被访问，即程序在一段时间内所访问的地址可能集中在一定的范围内，其典型情况是程序顺序执行。

工作集理论：工作集是进程运行时被频繁访问的页面集合。

2 操作系统概述

2.1特殊的操作系统

分类	特点
批处理操作系统	单道批：一次一个作业入内存，作业由程序、数据、作业说明书组成 多道批：一次多个作业入内存，特点：多道、宏观上并行微观上串行
分时操作系统	采用时间片轮转的方式为多个用户提供服务，每个用户感觉独占系统 特点：多路性、独立性、交互性和及时性
实时操作系统	实时控制系统和实时信息系统 交互能力要求不高，可靠性要求高（**规定时间内响应并处理**）
网络操作系统	方便有效共享网络资源，提供服务软件和有关协议的集合 主要的网络操作系统有：Unix、Linux 和 Windows Server 系统
分布式操作系统	任意两台计算机可以通过通信交换信息 是网络操作系统的更高级形式，具有透明性、可靠性和高性能等特性
微机操作系统	Windows：Microsoft 开发的图形用户界面、多任务、多线程操作系统 Linux：免费使用和自由传播的类 Unix 操作系统，多用户、多任务、多线程和多 CPU 的操作系统
嵌入式操作系统	运行在智能芯片环境中 特点：微型化、**可定制（针对硬件变化配置）**、实时性、可靠性、**易移植性（HAL 和 BSP 支持）**

2.2 存储管理

（1）页式存储：将程序与内存均划分为同样大小的块，以页为单位将程序调入内存。

优点：利用率高，碎片小，分配及管理简单

缺点：增加了系统开销；可能产生抖动现象

（2）段式存储：按用户作业中的自然段来划分逻辑空间，然后调入内存，段的长度可以不一样。

优点：多道程序共享内存，各段程序修改互不影响

缺点：内存利用率低，内存碎片浪费大

（3）段页式存储：段式与页式的综合体。先分段，再分页。1个程序有若干个段，每个段中可以有若干页，每个页的大小相同，但每个段的大小不同。

优点：空间浪费小、存储共享容易、存储保护容易、能动态连接

缺点：由于管理软件的增加，复杂性和开销也随之增加，需要的硬件以及占用的内容也有所增加，使得执行速度大大下降

2.3 磁盘管理

（1）存取时间=寻道时间+等待时间，寻道时间是指磁头移动到磁道所需的时间；等待时间为等待读写的扇区转到磁头下方所用的时间。

（2）读取磁盘数据的时间应包括以下三个部分：

找磁道的时间。

找块（扇区）的时间，即旋转延迟时间。

传输时间。

（3）磁盘移臂调度算法：

先来先服务 FCFS（谁先申请先服务谁）；

最短寻道时间优先 SSTF（申请时判断与磁头当前位置的距离，谁短先服务谁）；

扫描算法 SCAN（电梯算法，双向扫描）；

循环扫描 CSCAN（单向扫描）。

3 文件系统

（1）文件（File）是具有符号名的、在逻辑上具有完整意义的一组相关信息项的集合，例如，一个源程序、一个目标程序、编译程序、一批待加工的数据和各种文档等都可以各自组成一个文件。

一个文件包括文件体和文件说明。

- 文件体是文件真实的内容；
- 文件说明是操作系统为了管理文件所用到的信息，包括文件名、文件内部标识、文件类型、文件存储地址、文件长度、访问权限、建立时间和访问时间等。

（2）文件的类型

按文件的性质和用途分类可将文件分为系统文件、库文件和用户文件。

按信息保存期限分类可将文件分为临时文件、档案文件和永久文件。

按文件的保护方式分类可将文件分为只读文件、读/写文件、可执行文件和不保护文件。

4 性能评估

方法	描述	特点
时钟频率法	以时钟频率高低衡量速度	仅考虑 CPU
指令执行速度法	表示机器运算速度的单位是 MIPS	仅考虑 CPU
等效指令速度法（吉普森混合法）	通过各类指令在程序中所占的比例（W_i）进行计算得到的。	仅考虑 CPU，**综合考虑指令比例不同的问题。**
数据处理速率法（PDR）	PDR 值的方法来衡量机器性能，PDR 值越大，机器性能越好。	PDR = L/R **仅考虑 CPU+存储**
综合理论性能法（CTP）	CTP 用 MTOPS 表示。CTP 的估算方法是，首先算出处理部件每个计算单元的有效计算率，再按不同字长加以调整，得出该计算单元的理论性能，所有组成该处理部件的计算单元的理论性能之和即为 CTP。	**仅考虑 CPU+存储**
基准程序法	**把应用程序中用得最多、最频繁的那部分核心程序作为评估计算机系统性能的标准程序，称为基准测试程序（benchmark）。**	**综合考虑多部分**，基准程序法是目前一致承认的测试系统性能的较好方法。

【测试精确度排名】 真实的程序 > 核心程序 > 小型基准程序 > 合成基准程序

常见的 Web 服务器性能评测方法有基准性能测试、压力测试和可靠性测试。

系统监视：进行系统监视通常有3种方式：一是通过系统本身提供的命令，如 UNIX/Linux 系统中的 W、ps、last，Windows 中的 netstat 等；二是通过系统记录文件查阅系统在特定时间内的运行状态；三是集成命令、文件记录和可视化技术的监控工具，如 Windows 的 Perfmon 应用程序。

第四章 嵌入式系统

1 嵌入式系统概述

1.1 基本概念

（1）嵌入式系统是以应用为中心、以计算机技术为基础，并将可配置与可裁剪的软、硬件集成于一体的专用计算机系统，需要满足应用对功能、可靠性、成本、体积和功耗等方面的严格要求。

（2）从计算机角度看，嵌入式系统是指嵌入各种设备及应用产品内部的计算机系统。它主要完成信号控制的功能，体积小、结构紧凑，可作为一个部件埋藏于所控制的装置中。

（3）一般嵌入式系统由嵌入式处理器、相关支撑硬件、嵌入式操作系统、支撑软件以及应用软件组成。

嵌入式系统初始化过程：片级初始化→板级初始化→系统级初始化

1.2 嵌入式系统组成部件

嵌入式微处理器（MCU）、存储器（RAM/ROM）、内（外）总线逻辑、定时/计数器、看门狗电路（定时器溢出则中断，系统复位处理）、I/O 接口（串口、网络、USB、JTAG 接口--用来进行 CPU 调试的常用接口）、外部设备（UART、LED 等）、其他部件

2 嵌入式硬件

2.1 嵌入式微处理器分类

通常嵌入式处理器的选择还要根据使用场景不同选择不同类型的处理器，从处理器分类看，大致可分为 MPU、MCU、DSP、GPU、SoC

（1）微处理器（Micro Processor Unit，MPU）：将微处理器装配在专门设计的电路板上，只保留与嵌入式应用有关的母板功能。微处理器一般以某一种微处理内核为核心，每一种衍生产品的处理器内核都是一样的，不同的是存储器和外设的配置和封装。

（2）微控制器（Micro Control Unit，MCU）：又称单片机。与 MPU 相比 MCU 的最大优点在于单片化，体积大大减小，从而使功耗和成本下降，可靠性提高。

（3）信号处理器（Digital Signal Processor，DSP）：DSP 处理器对系统结构和指令进行了特殊设计（通常，DSP 采用一种哈佛结构），使其适合于执行 DSP 算法，编译效率高，指令执行速度也高。

（4）图形处理器（Graphics Processing Unit，GPU）：

GPU 是图形处理单元的缩写，是一种可执行染3D 图形等图像的半导体芯片（处理器）。

GPU 可用于个人电脑、工作站、游戏机和一些移动设备上做图像和图形相关运算工作的微处理器。

它可减少对 CPU 的依赖，并进行部分原本 CPU 的工作，尤其是在3D 图形处理中，GPU 采用了核心技术（如： 硬件 T&L、纹理压缩等）保证了快速3D 渲染能力。

GPU 目前已广泛应用于各行各业，GPU 中集成了同时运行在 GHz 的频率上的成千上万个 core，可以高速处理图像数据。最新的 GPU 峰值性能可高达100 TFlops 以上。

（5）片上系统（System on Chip，SoC）：

追求产品系统最大包容的集成器件。

它是一个产品，是一个有专用目标的集成电路，其中包含完整系统并有嵌入软件的全部内容。

同时它又是一种技术，用以实现从确定系统功能开始，到软/硬件划分，并完成设计的整个过程。

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。

2.2 AI 芯片

人工智能（Artificial Intelligence，AI）芯片的定义：从广义上讲只要能够运行人工智能算法的芯片都叫作 AI 芯片。但是通常意义上的 AI 芯片指的是针对人工智能算法做了特殊加速设计的芯片，现阶段，这些人工智能算法一般以深度学习算法为主，也可以包括其它机器学习算法。

人工智能芯片四大类（按技术架构分类）：

GPU

FPGA（现场可编程门阵列）

ASIC（专用集成电路）

类脑芯片

AI 芯片的关键特征：

新型的计算范式：AI 计算既不脱离传统计算，也具有新的计算特质

训练和推断：AI 系统通常涉及训练和推断过程

大数据处理能力：满足高效能机器学习的数据处理要求

数据精度：降低精度的设计

可重构的能力：针对特定领域而不针对特定应用的设计，可以通过重新配置，适应新的 AI 算法、架构和任务

开发工具：AI 芯片需要软件工具链的支持

2.3嵌入式微处理器体系结构

体系结构分类	定义	特点	典型应用
冯·诺依曼结构	冯·诺依曼结构也称普林斯顿结构，是一种将程序指令存储器和数据存储器合并在一起的存储器结构。	指令与数据存储器合并在一起。 指令与数据都通过相同的数据总线传输。	一般用于 PC 处理器，如 I3、I5、I7处理器。 **注：常规计算机属于冯·诺依曼结构**
哈佛结构	哈佛结构是一种并行体系结构，它的主要特点是将程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个独立的存储器，每个存储器独立编址、独立访问。	指令与数据分开存储，可以并行读取，有较高数据的吞吐率。 有4条总线：指令和数据的数据总线与地址总线。	一般用于嵌入式系统处理器。 **注：DSP 属于哈佛结构**

2.4 总线

总线的基本概念：总线是一组能为多个部件分时共享的信息传送线，用来连接多个部件并为之提供信息交换通路。

特点：

挂接在总线上的多个部件只能分时向总线发送数据，但可同时从总线接收数据。

通过总线复用方式可以减少总线中信号线的数量，以较少的信号线传输更多的信息。

总线分类：

（1）从功能上来对总线进行划分：数据总线、地址总线和控制总线

（2）从数据传输的方式划分为并行总线和串行总线

3 嵌入式操作系统

3.1 嵌入式操作系统的定义及特点

嵌入式操作系统（Embedded Operating System，EOS）是指用于嵌入式系统的操作系统。嵌入式操作系统是一种用途广泛的系统软件，负责嵌入式系统的全部软、硬件资源分配、任务调度、控制、协调并行活动等工作。通常包括与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面、标准化浏览器等。

根据系统对时间的敏感程度可将嵌入式系统划分为：

（1）嵌入式非实时系统

（2）嵌入式实时系统：能够在指定或者确定的时间内完成系统功能和外部或内部、同步或异步时间做出响应的系统。

嵌入式操作系统具有一般操作系统的功能，同时具有嵌入式软件的特点，主要有：

（1）微型化 （2）代码质量高 （3）专业化 （4）实时性强（5）可裁减、可配置。

实时操作系统的最核心特点是实时性强。

嵌入式实时操作系统实时性的评价指标

中断响应和延迟时间

任务切换时间

信号量混洗时间

3.2 嵌入式实时操作系统调度算法

抢占式优先级调度算法：根据任务的紧急程度确定该任务的优先级。大多数 RTOS 调度算法都是抢占方式（可剥夺方式）。

最早截止期调度算法（EDF 算法）：根据任务的截止时间头端来确定其优先级，对于时间期限最近的任务，分配最高的优先级。

最晚截止期调度算法：根据任务的截止时间末端来确定其优先级，对于时间期限最近的任务，分配最高的优先级。

3.3 操作系统内核架构

内核是操作系统的核心部分，它管理着系统的各种资源。内核可以看成连接应用程序和硬件的一座桥梁，是直接运行在硬件上的最基础的软件实体。

目前从内核架构来划分，可分为宏内核（Monolithic Kernel）和微内核（Micro Kernel）。

第五章 数学与经济管理

1 大数据架构

对比内容	Lambda 架构	Kappa 架构
复杂度与开发、维护成本	需要维护两套系统（引擎），复杂度高，开发、维护成本高	只需要维护一套系统（引擎），复杂度低，开发、维护成本低
计算开销	需要一直运行批处理和实时计算，计算开销大	必要时进行全量计算，计算开销相对较小
实时性	满足实时性	满足实时性
历史数据处理能力	批式全量处理，吞吐量大，历史数据处理能力强	流式全量处理，吞吐量相对较低，历史数据处理能力相对较弱
使用场景	直接支持批处理，更适合对历史数据分析查询的场景，期望尽快得到分析结果，批处理可以更直接高效地满足这些需求。	不是 Lambda 的替代架构，而是简化，Kappa 放弃了对批处理的支持，更擅长业务本身为增量数据写入场景的分析需求
选择依据	根据两种架构对比分析，将业务需求、技术要求、系统复杂度、开发维护成本和历史数据处理能力作为选择考虑因素。 计算开销虽然存在一定差别，但是相差不是很大，所以不作为考虑因素。	

2 边云协同

边云协同放大边缘计算与云计算的应用价值，主要包括六种协同：资源协同、数据协同、智能协同、应用管理协同、业务管理协同、服务协同。

制作于 23 年 11 月 适用于第 2 版教材