

**SICHUAN UNIVERSITY**



**题 目 个人作业-软件架构阐述**

**学生姓名 李亚宸**

**课程 软件项目管理**

**学号 2022141470276**

**学院 建筑与环境学院**

**专业 工程力学（力学-软件工程交叉学科实验班）**

**指导教师 毌攀良**

1. **C/S架构阐述**

客户端-服务器（Client-Server，C/S）架构是一种分布式系统架构，其中系统的功能划分为两个主要部分：**客户端**和**服务器**。客户端是用户与系统交互的界面，通常运行在用户的计算机或设备上，而服务器则是提供服务、存储数据并执行业务逻辑的核心，通常运行在远程服务器上。两者通过网络进行通信。

客户端-服务器 架构是一种非常常用的架构，适用于几乎所有的网络应用，其**主要特点**包括：

1.集中化管理：服务器集中管理数据与应用逻辑，便于维护和升级。

2.分布式计算：将任务划分为客户端和服务器部分，从而提高了系统效率。

3.可扩展性：支持横向扩展，增加更多服务器节点以处理更多的客户端请求。

4.多用户支持：允许多个客户端同时与服务器交互。

而常见的客户端-服务器模式包括：两层架构（客户端与服务器直接交互）和三层架构（引入中间层如应用服务器或数据库服务器）。

1. **C/S架构应用场景**

1.高性能与交互性要求高的应用：如桌面应用程序（例如微软Office、Adobe Photoshop），它们需要强大的本地计算资源来处理复杂任务。

2.对网络传输速度要求不是特别敏感的应用：如企业内部系统或局域网环境下的应用，网络传输速度可能不是主要考虑因素。

3.需要高度安全性和稳定性的应用：如银行系统，C-S架构通过客户端应用程序可以更好地控制数据安全，减少数据泄露的风险。

4.复杂图形界面和数据处理的应用：客户端应用程序可以支持复杂的图形界面和大量的数据处理，提供更丰富的用户体验和功能。

5.在线游戏：许多大型多人在线游戏（如《魔兽世界》）采用了C-S架构，游戏客户端负责图形渲染、用户输入处理，而服务器处理用户间的互动、数据同步和游戏世界状态。

6.企业级软件：如ERP（Enterprise Resource Planning，企业资源计划）系统、CRM（Customer Relationship Management，客户关系管理）系统等，也是基于C-S架构

**三、C/S架构的优缺点**

**优点：**

1.界面丰富：C/S架构的界面和操作可以很丰富，提供良好的用户体验。

2.安全性高：安全性能容易保证，实现多层认证也不难。

3.响应速度快：由于只有一层交互，因此响应速度较快。

4.数据处理能力强：客户端可以承担一部分的数据处理，减轻服务器负担。

**缺点：**

1.适用面窄：通常用于局域网中，不适合面向未知的用户群体。

2.维护成本高：系统升级时，所有客户端的程序都需要改变，维护成本较高。

3.用户群固定：由于程序需要安装才可使用，不适合快速部署和广泛分布。

**C/S架构与其他常见的软件架构比较，见下表：**

| **架构**  **特性** | **C/S架构** | **B/S**  **架构** | **P2P**  **架构** | **微服务架构** | **SOA架构** | **云架构** |
| --- | --- | --- | --- | --- | --- | --- |
| **用户界面** | **丰富** | **依赖浏览器** | **分布式** | **可定制** | **服务导向** | **通过浏览器** |
| **安全性** | **高** | **较低** | **依赖节点** | **高** | **高** | **依赖云服务提供商** |
| **响应速度** | **快** | **可能较慢** | **依赖网络** | **快** | **快** | **依赖网络** |
| **维护成本** | **高** | **低** | **低** | **中等** | **中等** | **低** |
| **适用场景** | **局域网，固定用户群** | **广域网，多用户** | **分布式网络** | **复杂应用** | **企业级应用** | **互联网应用** |
| **扩展性** | **有限** | **高** | **高** | **高** | **高** | **高** |

**四、C/S架构技术栈**

**1.客户端（前端）技术栈：**

（1）Java Swing/AWT：用于创建桌面应用程序的图形用户界面（GUI）工具包。

（2）JavaFX：用于创建丰富客户端应用程序的现代框架，提供高级的UI控件、媒体处理功能和动画支持。

（3）C# (Windows Forms/WPF)：用于开发Windows桌面应用程序，提供丰富的GUI组件和良好的性能。

（4）Qt (C++/Python)：跨平台的应用程序和UI框架，支持多种编程语言。

（5）Electron：允许使用Web技术（HTML, CSS, JavaScript）构建跨平台的桌面应用程序。

**2.服务器端（后端）技术栈：**

（1）Java EE (Enterprise Edition)：用于构建企业级应用，提供Servlets, JSP, Spring Framework, Spring Boot等技术。

（2）NET Framework：微软的框架，用于构建Windows服务和Web服务。

（3）Node.js：基于Chrome的V8 JavaScript引擎，用于构建可扩展的网络应用。

（4）Python：以其简洁的语法和强大的库支持，适用于快速开发。

（5）Ruby on Rails：Ruby的服务器端Web应用框架，强调约定优于配置。

**3.数据库技术：**

（1）关系型数据库：如MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database。

（2）NoSQL数据库：如MongoDB, Cassandra, Redis。

（3）文件存储：如SQLite, XML, JSON文件。

**4.通信协议：**

（1）TCP/IP：用于建立可靠的网络连接。

（2）HTTP(S)：用于客户端和服务器之间的数据交换。

（3）WebSocket：提供实时双向通信的能力。

（4）RPC (Remote Procedure Call)：如gRPC, XML-RPC, JSON-RPC等，用于远程调用服务器端的方法。

（5）SOAP/REST：SOAP是基于XML的协议，REST是基于HTTP的风格，用于构建API。

**5.开发工具：**

（1）版本控制系统：如Git, SVN。

（2）构建工具：如Maven, Gradle, Ant (Java), MSBuild (.NET), CMake (C++, C)。

**6.其他工具：**

（1）API开发工具：如Postman, Swagger。

（2）容器化和虚拟化工具：如Docker, Kubernetes。

**五、C/S架构开源市场实现案例**

1.Apache HTTP Server

Apache HTTP Server 是最广泛使用的开源 Web 服务器之一，它采用了客户端-服务器架构。其实现方式如下：

①客户端：用户通过浏览器向 Apache 服务器发送 HTTP 请求。

②服务器：Apache 接收请求，解析并根据配置返回静态资源（如 HTML、CSS 文件）或调用后端服务生成动态内容。

③扩展功能：Apache 支持模块化扩展（如 mod\_php、mod\_wsgi），以便与其他服务如 PHP、Python 等集成，增强业务逻辑能力。

2.MySQL 数据库

MySQL 是一个基于 C/S 架构的开源关系型数据库。

①客户端：开发人员或应用通过 SQL 客户端工具（如 MySQL Workbench 或自定义应用程序）发送查询。

②服务器：MySQL 服务器接收查询并执行相关数据库操作，如数据插入、更新或检索。  
 ③MySQL 的多线程设计确保了高并发性能，同时支持通过 API 接口或语言驱动（如 Python 的 mysql.connector）访问数据库。

3.Nginx

Nginx 是一款高性能开源服务器，适用于反向代理、负载均衡和静态资源服务。

①客户端：用户的浏览器通过 HTTP 请求访问 Nginx。

②服务器：Nginx 处理静态资源请求或将动态请求转发到后端服务器（如 Flask 或 Django 应用）。通过负载均衡功能，Nginx 将请求分发到多个后端实例。

**六、基于架构实现功能的基础**

为了在上述开源架构的基础上进一步开发或定制，需要以下准备：

1.理解底层实现与配置

熟悉基础的网络协议（如 HTTP、TCP/IP）和数据库语言（如 SQL）。

阅读官方文档和源码，了解如 Apache 的模块加载机制、MySQL 的查询优化机制或 Nginx 的配置语法。

2.学习编程语言

针对特定项目学习相关语言：如 Apache 和 Nginx 使用 C 开发，MySQL 使用 C++ 实现。

掌握脚本语言（如 Python、PHP、JavaScript）以实现客户端交互功能。

3.部署与测试技能

掌握虚拟化和容器化工具（如 Docker），便于快速部署和管理服务。

学习测试工具（如 Postman 用于 API 测试，JMeter 用于性能测试），确保系统的可靠性与稳定性。

4.社区参与

参与相关开源社区（如 GitHub 上的项目仓库），学习开发者的最佳实践。

提交 Bug 或贡献代码，以获取项目开发经验。

[参考文献]：

【1】Kurose, J. F., & Ross, K. W. Computer Networking: A Top-Down Approach. Pearson Education, 2021.

【2】software engineering structure practice-4