<软件在线评测系统>

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <日/月/年> | <x.x> | <详细信息> | <姓名> |
| 25/6/2023 | 0.5 | 编写1、10两部分 | 常朋宇 |
| 28/6/2023 | 1.0 | 编写第一版 | 常朋宇 吴昶霖 |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 技术视图 5

8. 数据视图（可选） 5

9. 核心算法设计（可选） 5

10. 质量属性的设计 5

软件架构文档

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

本文档旨在设计一个Online Judge系统的软件架构，为开发人员提供指导和参考。Online Judge系统是一个用于评判程序员提交的代码在各种编程问题上的正确性和性能的系统。它提供了一个在线平台，让用户可以提交自己的代码解决问题，并通过系统自动评测来获得反馈和排名。

在本文档中，我们将详细讨论Online Judge系统的软件架构，以确保系统具备可扩展性、可靠性和高性能等重要质量属性。我们将通过不同视图来描述系统的各个方面，包括用例视图、逻辑视图、进程视图、部署视图、实现视图和技术视图。此外，还将包括数据视图和核心算法设计（如果适用）。

在用例视图中，我们将定义系统的功能需求，并描述用户与系统之间的交互。通过列举典型的用例并描述其执行流程，我们将帮助开发人员深入了解系统的功能和行为。

在逻辑视图中，我们将讨论系统的整体架构和模块之间的关系。我们将标识出构成系统的重要设计包，并说明它们在架构方面的重要性。

进程视图将展示系统中运行的进程及其之间的交互。我们将描述每个进程的功能和职责，以及它们之间的通信方式，以便确保系统的并发性和可扩展性。

部署视图将描述系统在物理环境中的部署情况。我们将提供系统的部署架构图，并说明每个组件的 部署位置和配置信息，以确保系统的可用性和可靠性。

实现视图将涵盖系统的具体实现细节，包括所选的编程语言、框架和库。我们将提供类图或模块图，并说明每个组件的功能和接口，以帮助开发人员理解和实现系统。

技术视图将描述系统使用的关键技术和工具，如数据库、服务器和消息队列。我们将提供技术架构图，并说明每个技术组件的作用和配置，以确保系统的性能和可维护性。

此外，本文档还将考虑系统的质量属性设计，包括性能、可靠性、安全性等。我们将列举重要的质量属性，并说明系统设计如何满足这些需求。

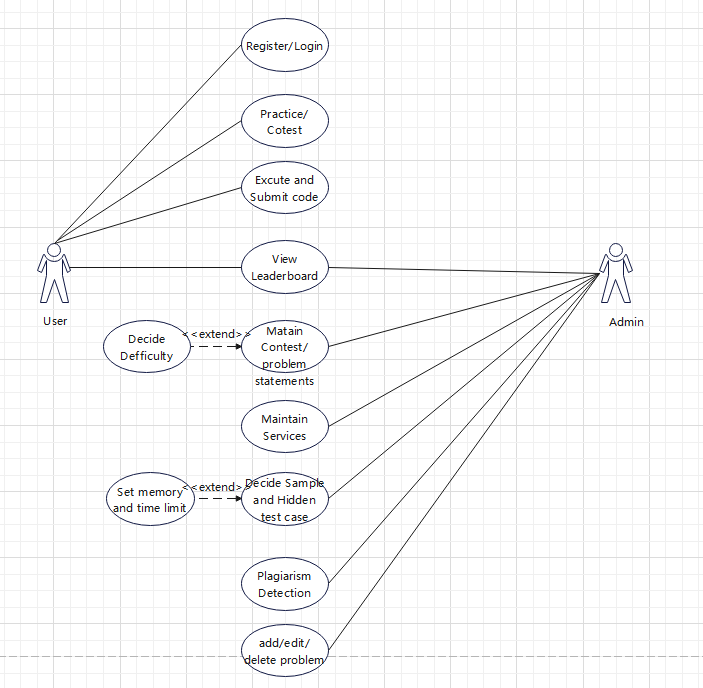
通过本文档的设计和讨论，我们将为开发人员提供一个清晰的软件架构指导，以实现一个高效、可靠和可扩展的Online Judge系统。

## 参考资料

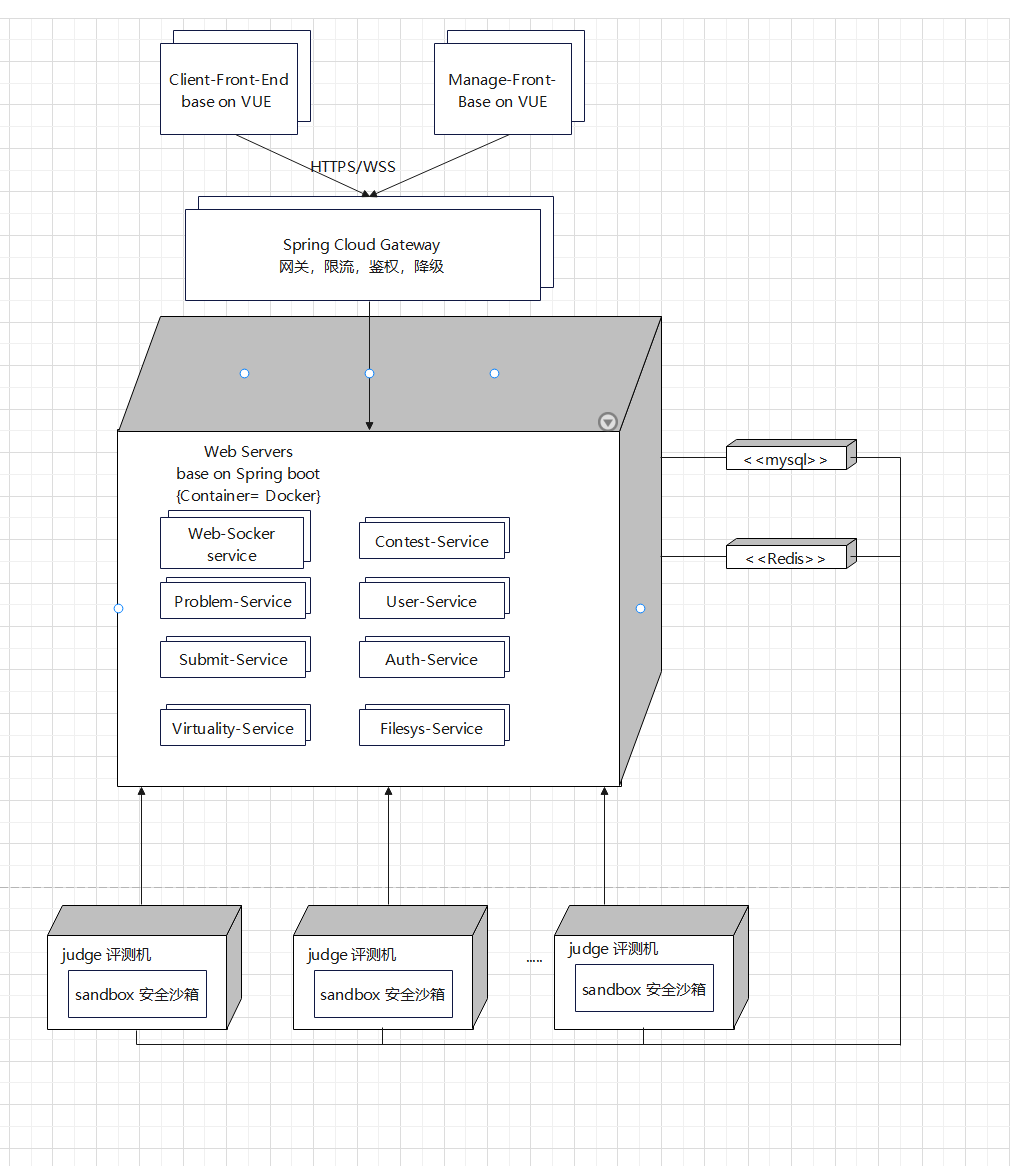
<软件需求规约> 开发团队出版

<软件项目计划> 开发团队出版

# 用例视图



# 逻辑视图



## 概述

本系统采用微服务架构，将OnlineJudge的各个功能分为若干个弱耦合的服务。逻辑架构图如上。首先有客户端和管理圆的前端连接到Spring Cloud的网关，连接到云服务器，拥有各种服务，然后服务器通过数据库链接judge评测机，评测机在自己的沙箱运行评测代码，再将结果返回给web服务器。

## 在构架方面具有重要意义的设计包

在逻辑架构中，主要包括以下服务：

用户服务：负责用户的注册、登录、权限管理等功能。它处理与用户相关的请求，并管理用户的信息和权限。

题目服务：负责题目的管理，包括题目的添加、修改、删除等功能。它提供了一个题库，供用户选择并提交代码进行评测。

评测服务：负责对用户提交的代码进行评测，并返回评测结果。它与编译器和执行环境交互，将用户的代码编译并在执行环境中运行，然后根据预定义的测试用例进行评测。

排名服务：负责计算用户在各个题目上的排名，并提供排名榜单。它根据用户提交的代码评测结果进行排名计算，并将排名信息展示给用户。

数据服务：负责存储和管理系统的数据，包括用户信息、题目信息、评测结果等。它提供了数据的持久化存储，并提供接口供其他服务进行数据的读取和写入。

每个服务可以独立开发、测试和部署，通过使用适当的通信协议和接口定义，实现了服务之间的解耦和灵活性。

# 进程视图

进程之间的主要通信模式采用网络通信的方式，REST，socket等。

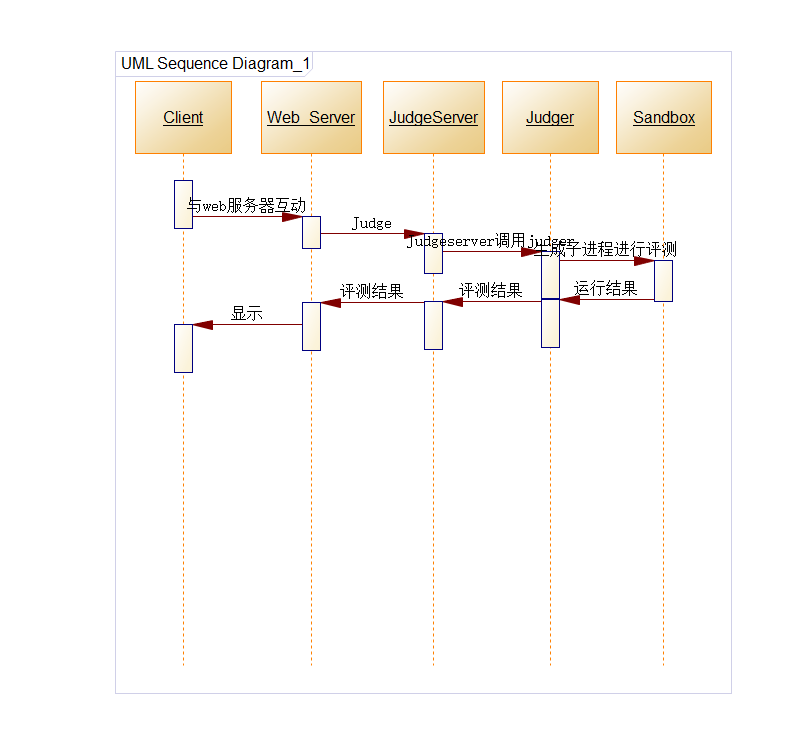
在进程视图中，系统可以分解为以下进程组：

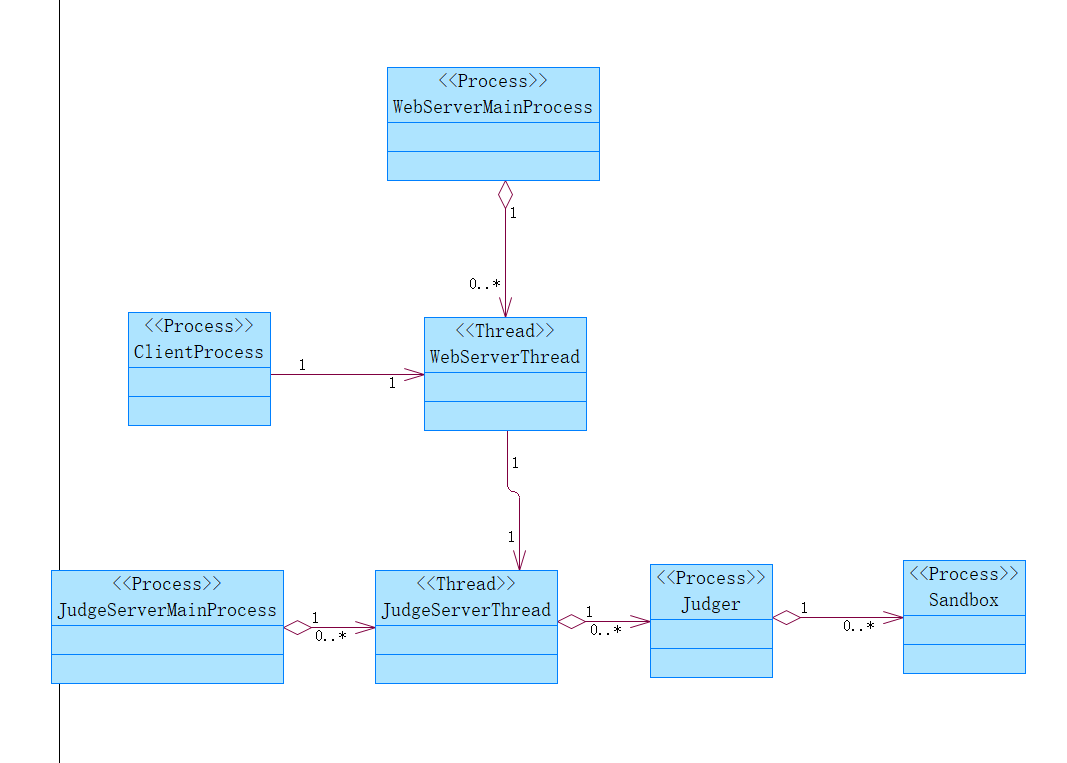
用户接口进程组：包括用户界面、Web服务器等进程，用于接收用户的请求，并将请求传递给相应的服务进行处理。用户界面可以是一个Web页面或移动应用，通过HTTP或其他适当的协议与后端服务进行通信。

服务进程组：包括用户服务、题目服务、评测服务、排名服务等进程，每个进程独立运行，负责相应功能的处理。这些进程之间通过网络通信进行交互，使用REST或Socket等协议进行消息传递和数据交换。

数据库进程组：包括数据服务的数据库进程，负责存储和管理系统的数据。

每个进程组可以运行在独立的服务器上，实现了水平扩展和高可用性。

进程之间交互例子:以正常的用户提交代码的流程为例

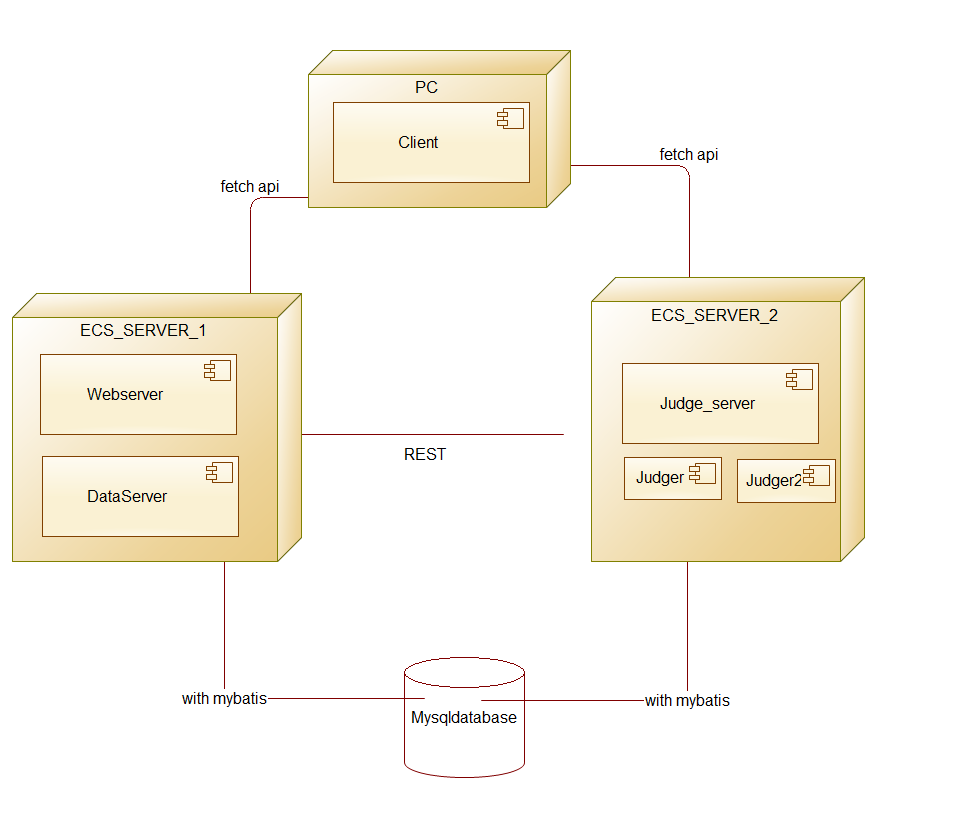


# 部署视图

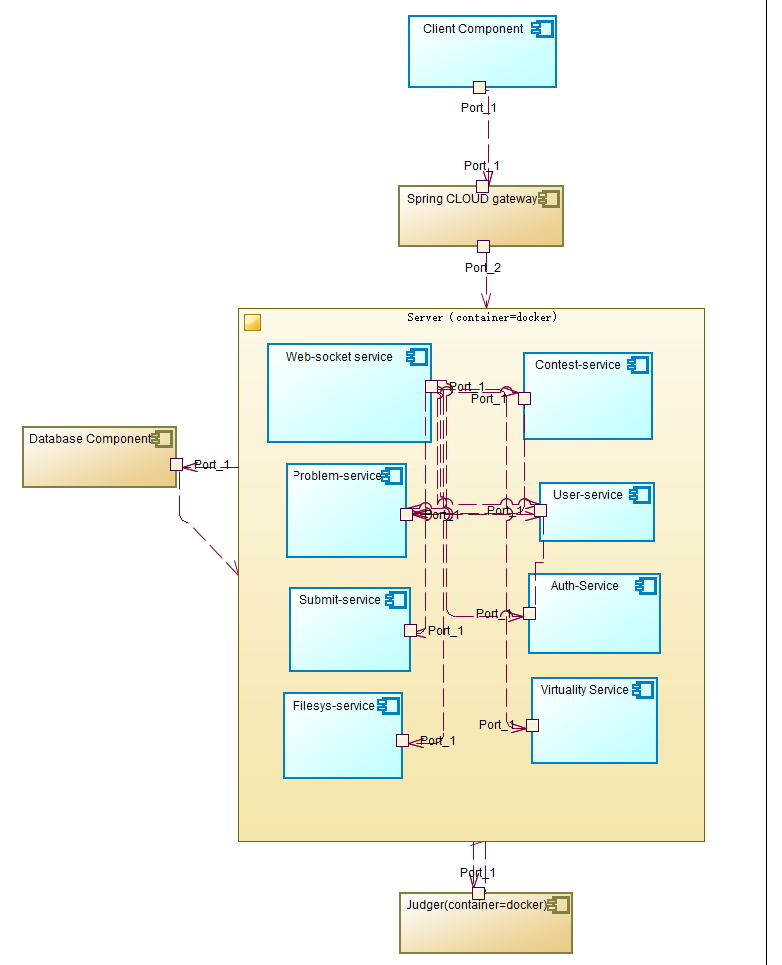
这次将服务器部署在两个华为弹性云服务器ECS上。具体状况如图:

规格：通⽤计算型s7 s7.large.4 2vCPUs 8GiB，镜像：Ubuntu18.04 server 64bit(40GiB)

采用容器化部署。

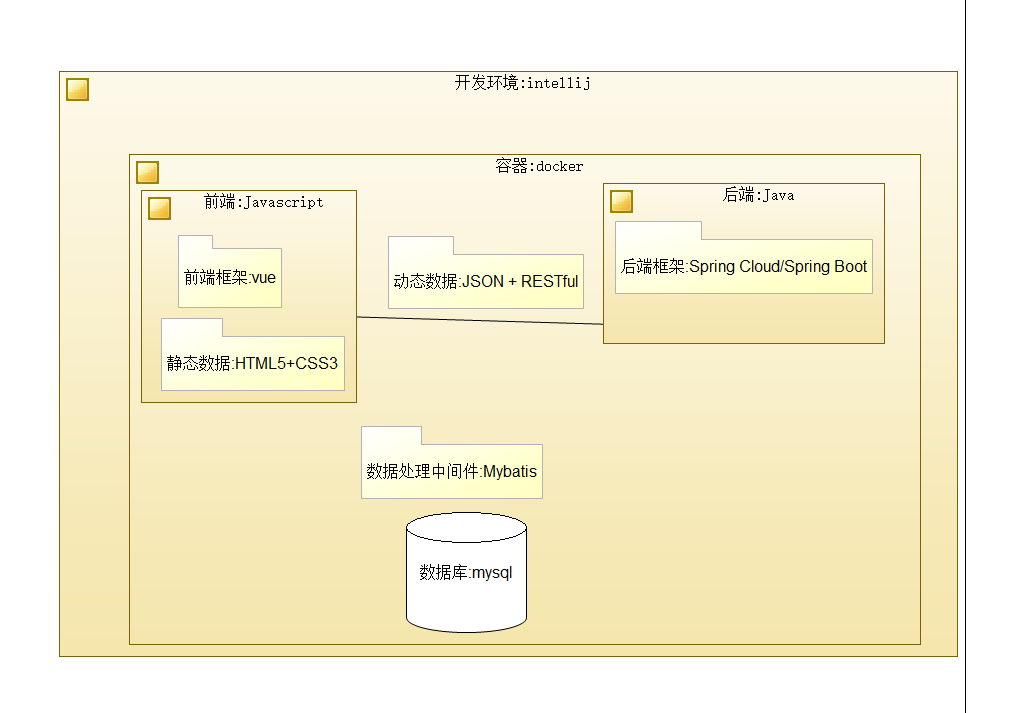


# 实现视图



实现模型整体分为：视图层、控制层、服务层、模型层、数据层。具有重要意义的构件如:Judger等。

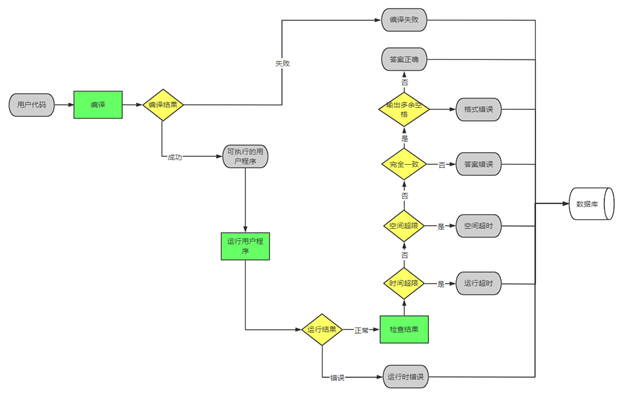
# 技术视图



我们采用IntelliJ作为开发的IDE，mysql（可能还有redis)作为数据库，数据处理中间件为mybatis/mybatis plus，前端采用vue的框架，数据采用ajax的方式才前后端传递。后端采用springcloud/springboot的框架。另外，一些服务器：如judger等将在docker当中运行。

# 数据视图（可选）

# 核心算法设计（可选）

将对评测算法进行设计:下图展示了一个用户代码是如何经过编译和运行和程序是如何进行数据库交互的一个流程。

# 质量属性的设计

易用性：1.为用户提供适当的反馈和协助

2.系统采用直观和用户友好的界面设计，提供清晰的导航、一致的布局和可见的操作反馈。

3.将用户接口与应用的其余部分分离。

可靠性：1.用户提交的代码将在沙箱内运行，防止其中的有害代码。具有安全性意义。

2.容错和异常处理：系统具备容错能力，能够捕获和处理异常情况，提供恰当的错误处理和恢复机制，保证系统的稳定性和可用性。

性能：

1.响应时间：系统设计考虑到减少响应时间，采用合适的算法和数据结构，以及优化数据库查询和缓存机制，以提供快速的用户响应。

2.能够处理大量并发请求和用户，通过水平扩展和负载均衡来提高系统的整体性能。

3.资源利用率：系统优化资源的使用，合理管理内存、存储和网络带宽等资源，以提高系统的效率和性能。

可移植性：1.接口和实现分离。

可扩展性：1.采用模块化的架构，能够做到模块的添加和修改。