基于微服务架构的在线评判系统设计与实现

李西明,梁志才,刘龙浩,祝胜林

(华南农业大学 数学与信息学院,广东 广州 510642)

摘 要: 现有在线评判系统用于程序设计课程实践教学存在学生学习投入低、系统考核过程单一且缺乏动态扩缩性等问题。由此对现有的在线评判系统进行重新设计与实现,采用开卷训练与闭卷测试相结合的综合评价模式,应用微服务和Docker技术对系统进行重新架构。根据压力测试分析可知,新系统能够应对高并发情形,综合评价模式能够提高学生的学习投入,并实现系统设计目标。

关键词: 程序设计;在线评判系统;闭卷测试;弹性微服务;Docker

DOI: 10. 11907/rjdk. 221973 **开放科学(资源服务)标识码(OSID):**

中图分类号: TP311.52 **文献标识码:** A **文章编号:** 1672-7800(2023)008-0144-07

Design and Implementation of Online Evaluation System Based on Micro-Service Architecture

LI Ximing, LIANG Zhicai, LIU Longhao, ZHU Shenglin

(*College of Mathematics and Informatics*, *South China Agricultural University*, *Guangzhou* 510642, *China*)

Abstract: In view of the problems that the existing online evaluation system is used in the practical teaching of programming courses, such as low students' learning investment, single examination process and lack of dynamic expansion and contraction. Therefore, the existing online evaluation system is redesigned and implemented, and the comprehensive evaluation mode combining open-book training with closed-book testing is adopted, and the micro-service and Docker technology are applied to restructure the system. According to the analysis of stress test, the new system can cope with high concurrency, and the comprehensive evaluation mode can improve students' learning input, which has reached the system design goal.

Key Words: programming; online judge system; closed-book testing; micro-service architecture; Docker

0 引言

在线评判或评测系统(Online Judge,OJ)依据软件工程中的黑盒测试实现自动化线上程序评判。自2000年以来,OJ系统在程序设计与算法竞赛、竞赛培训和课程实践教学等方面得到了较为广泛的应用。2016年前后,计算机相关专业的求职面试或研究生复试也开始使用OJ系统,确保整个过程及时、公开和公正。国内已有部分高校提出使用在线评判系统辅助计算机类课程教学,主要分为以下4类[1]:①将信息学奥林匹克竞赛或ACM竞赛所采用的在线评测系统应用于计算机类课程教学;②以课程训练为主教学类的在线评测系统;③面向竞赛或基于慕课平台应用的在线评测系统;④联合机考和面试的研究生招生复试与考评系统。后疫情时代,在线教育与课堂教学相结合的混合教学模式将可能成为高校教学的"新常态"[2]。在线评测系统在线上远程实验教学、全过程评价和在线考核方面将有更广泛的应用[3]。

华南农业大学自2009年起上线在线评判系统,在专业竞赛网站和商业化在线评判系统功能基础上,扩展了实验教学管理和评价功能。在线评判系统不仅满足了正常的实验教学和竞赛训练需要,而且减轻了教师的全过程管理和

收稿日期: 2022-08-18

基金项目: 广州市科技计划重点实验室建设项目(201902010081)

作者简介: 李西明(1974-),男,博士,华南农业大学数学与信息学院副教授、硕士生导师,研究方向为计算机应用、软件架构;梁志才(1991-),男,华南农业大学数学与信息学院硕士研究生,研究方向为计算机应用、计算机视觉;刘龙浩(2000-),男,华南农业大学数学与信息学院学生,研究方向为软件工程;祝胜林(1969-),男,博士,华南农业大学数学与信息学院副教授、硕士生导师,研究方向为计算机应用、信息安全。本文通讯作者:祝胜林。

评价中数据登记与处理的负担。但是其也存在一些问题：①实验过程考核单一，最方便的是提交题目数，即通过的题目数，未通过的不计算在内，以避免学生只顾提交题目数量而不顾题目对错[4]；②传统的单体应用架构或面向服务架构的部署缺乏动态扩缩性。

为此，本文对现有的在线评判系统进行重新设计与实现，采用开卷训练与闭卷测试相结合的综合评价模式，应用微服务和Docker技术对系统进行重新架构。

1　问题分析与对策

1.1　过程考核单一

每个实验内容分为课堂练习和课下练习。课堂练习包括2~3道编程题，一般要求一次实验课完成；课下练习包括另外2~3道编程题，由学生课下自行完成。但实验成绩表只有提交题数，并没有区分课堂和课下题数，也没有限时要求。一般而言，有两种计算实验效果分的方法：

（1）总提交题数。

$$\rho = \frac{M}{N} * 100\% \tag{1}$$

其中，ρ为实验效果得分，满分100，M为截至时间前已提交且正确的题目数，N为在线评判系统要求提交的总题数。

（2）每实验提交题数。

$$\rho = \sum_{i=1}^{k} \frac{M_i}{N_i} * C_i \tag{2}$$

其中，ρ为实验效果得分，满分100，M_i为第i次实验已提交且正确的题目数，N_i为第i次实验应提交题目数，C_i为第i次实验满分，k为实验数。

由于题目公开，已提交学生可保留源代码。学习不认真的学生，为了提高自己的实验效果得分，存在用别人的源代码进行提交或截止日期前集中提交的现象。混合教学模式下在线学习和课堂学习在行为投入、认知投入上存在一致性，学生对课程的挫败感、焦虑感等消极情感往往与表层学习相伴而生[2]。因此，学生投入低必定会影响其深层学习效果。

为了督促学生增加实验课程的学习投入，促使其进入深层学习，将实验内容分为训练、每次实验小测和考试3部分，均在线完成。

（1）训练。采用开卷训练模式，题目是公开，可以反复提交，可在实验课上完成或课余时间完成。按提交题目数评价，当实验要求提交的题目全部提交后才可获得小测资格。

（2）小测。小测次数5次，每次满分15分，总分为75分。时间一般安排在下一次实验开始进行。题目来自于不公开的小测题库，从5道题中随机抽取一道，内容与本实验训练题目的知识点相关，但不是训练的原题。限时15分钟，限制提交次数为3次，多次提交会扣分。

（3）考试。仅有1次，满分25分。考试时间安排在最后一次实验课。考试题型有两种：客观题10小题，1分/小题，共10分；编程题1小题，15分/小题，共15分，题目不是训练的原题，而是来自不公开的考试题库，从5道题中随机抽取1道，但可公开复习范围。

该评价模式综合了开卷训练模式和闭卷测试模式的评价方法，简称综合评价模式。实验效果得分：

$$\rho = \sum_{i=1}^{5} E_i + T \tag{3}$$

其中，ρ为实验效果得分，满分100，E_i为第i次小测的得分，T为考试得分。

1.2　缺乏动态扩缩性

在线评测系统传统部署可以采用单体应用架构或面向服务架构：

（1）单体应用架构。在线评测系统按照模块化设计，开发完成后被打包并部署为一个具体的应用。虽然容易开发、调试和部署，但随着用户人数增加，一台服务器难以满足系统负载。通过增加服务器进行水平扩展，能够解决问题，但增加了较大成本投入；不需要扩展的服务也被扩展，浪费了资源；用户人数减少，也不能动态缩减。

（2）面向服务架构（Service-Oriented Architecture，SOA）。将在线评测系统中相近的功能聚合在一起，以服务的形式向外提供。按照功能细分为不同的子系统，各子系统依赖服务中间件调用所需服务[5]。虽然SOA架构降低了模块间的耦合度，提高了重用性和开发效率，但多数情况下，相互独立的服务仍然会部署在同一运行环境中，随着业务功能增多，服务会变得很复杂。因此，SOA架构无法支撑数据爆发式增长和业务的快速迭代[6]。

智能化程序设计管理系统不仅面向本校师生，还面向其他高校的师生，有些学期人数很多，有些学期人数较少，因而用户增量预期较大。为了应对这一问题，设计和实现基于微服务架构的在线评判系统，并成功将其部署在云平台上。微服务架构是SOA的一种变体[7]，每一个微服务都有自己的业务逻辑和适配器，完成某个特定功能，是一个独立的应用，可独立承担对外服务的职责，具备独立的运行进程，可独立部署[8]。微服务具有低成本、弹性伸缩等优势[9]。微服务架构是对软件系统进行整体和全面的考虑，包括开发、测试和部署等，可带来如下好处：①体积小、复杂度低、开发效率高；②单个微服务发生变更时，无需编译和部署整个应用，发布更高效；③技术选型灵活且多样化；④比较适合未来有一定扩展复杂度，且有很大用户增量预期的应用。

2　系统设计

系统设计是系统开发的重要部分，着重介绍在线评判系统的角色设计、模块设计、数据库设计和微服务架构的内容。根据系统角色和模块的划分对数据库进行详细设计。

2.1 角色

系统设计了3种角色，每种角色的功能包括如下：

（1）管理员。分为超级管理员和校级管理员，超级管理员可以创建学校和指定校级管理员，拥有后台管理的各项功能。校级管理员具有审核教师申请、管理题目、实验和考试等方面的功能。

（2）教师。负责创建班级、导入所教班级的学生名单；获得授权可以创建题目、实验、试卷或考试。

（3）学生。登录系统认证成功，可以查看实验和提交实验、参加考试、查看完成情况或成绩，也可以查看系统公开题目的列表。

2.2 模块

为了便于系统开发，将在线评判系统划分为不同的子模块。如图1所示，系统包括实验管理、授课班级、上机考试、题库管理模块。实验管理模块包括选择实验和历史记录模块。授课班级包括查看成绩和班级管理模块。上机考试包括选择题、判断题和编程题模块。题库管理包括选择题、编程题、判断题和题库列表模块。

2.3 数据库

根据上述分析，对数据库进行详细设计。系统涉及的实体有学校、班级、用户、题目、考试和实验等，它们的实体关系（E-R）图，如图2所示。每个用户拥有一个角色，其中管理员角色可以管理班级信息、学生信息、实验和题库，教师角色可以管理上机考试信息，学生用户则可以做实验题目和参加上机考试。

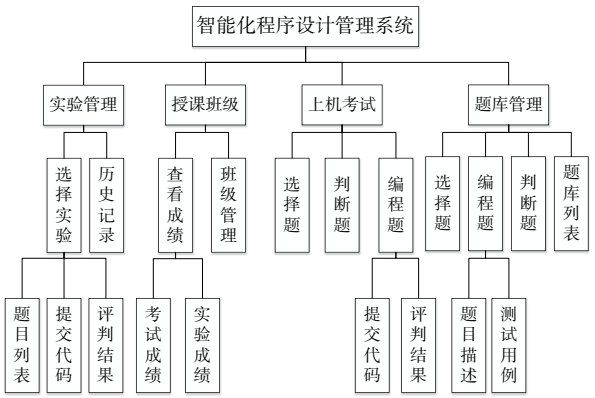

Fig. 1 System modules

图1 系统模块

2.4 微服务架构

整个系统运行在Virtual Private Cloud（VPC）上，除互联网（Internet）、防火墙（firewall）、数据库（MySQL）和云文件存储（Cloud File Storage）外，其他都是微服务，每个微服务都构成各自的集群，微服务之间可以自由通信。系统微服务架构如图3所示。

各部件及其作用如下：

（1）Nginx。Nginx实现动静分离和反向代理，并解决跨域无法访问的问题[10]。

（2）RocketMQ。RocketMQ是Spring AMQP（Advanced Message Queuing Protocol）的默认通信中间件，应用程序对应用程序的异步通信。其中，NameServer是微服务注册中心，拥有完整的路由信息，支持Broker的动态注册和发现，

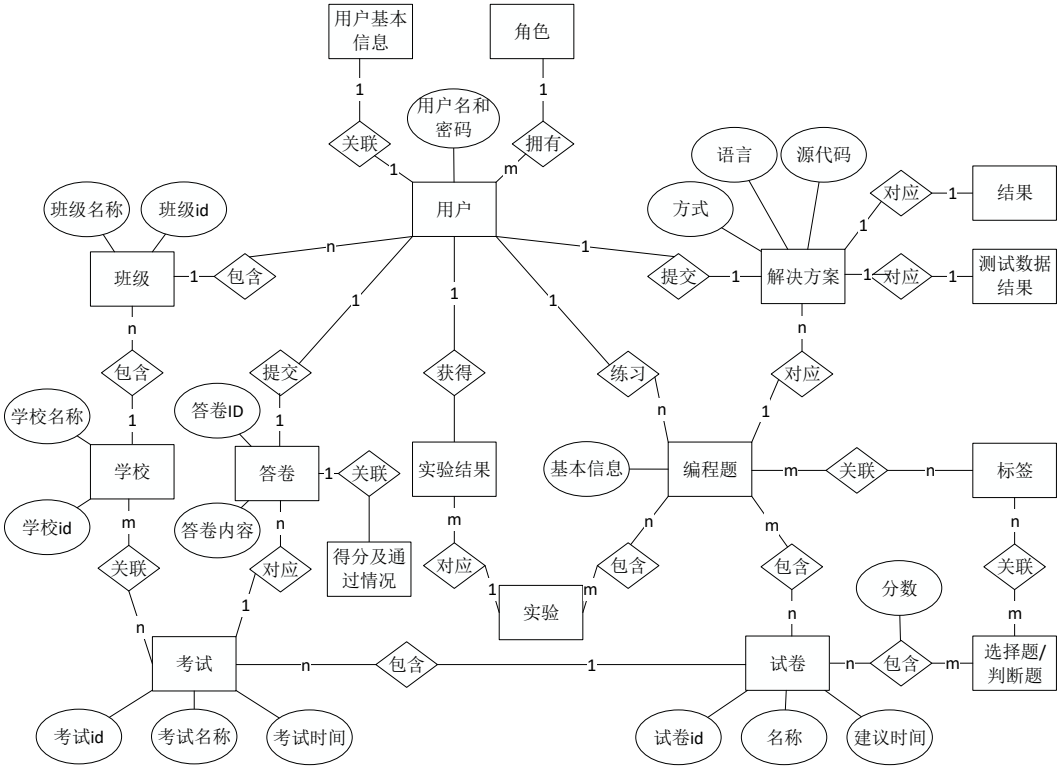

Fig. 2 System E-R diagram

图2 系统E-R图

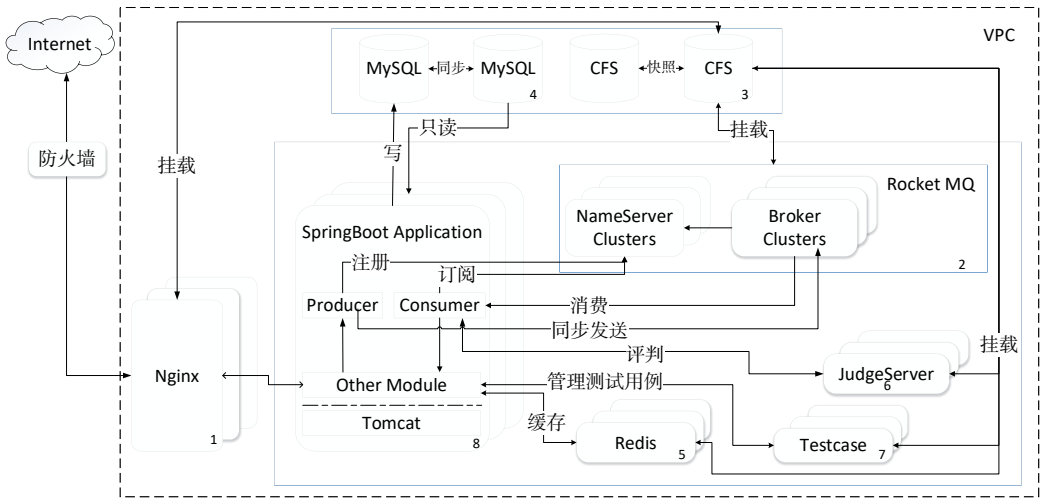

Fig. 3　System micro-services architecture

图 3　系统微服务架构

维护 Topic 和 Broker 之间的关系，以及与其他 NameServer 之间的相互通信。Broker 负责将消息队列中的评判消息主动推送到消费者[11]。

（3）CFS。CFS（Cloud File Storage，云文件存储）用来存储所有微服务的配置文件和测试用例文件，最终挂载到相应的微服务容器上，这样有利于解决文件冗余和不一致性的问题。

（4）MySQL。MySQL 提供数据存储服务，采用读写分离的主从结构，主 MySQL 负责写，从 MySQL 负责读（查询），可以较好地提高系统响应速度[12]。

（5）Redis。Redis 用于缓存 Docker 容器目录。Redis 完全基于内存，具有良好的并行读写性能[13]，因此使用 Redis 缓存数据，从缓存中恢复判题所需的环境，无需再次下载所需组件进行构建，从而提高判题效率。

（6）JudgeServer。JudgeServer 微服务提供 API 供后台管理微服务调用，完成评判用户的程序并将评判结果返回。

（7）Testcase。Testcase 微服务面向管理员，实现题目管理。

（8）SpringBoot Application。SpringBoot Application 采用 SpringBoot 框架[14]实现，属于后台管理微服务，是系统的中心枢纽。Producer 注册到 NameServer 上获取 Broker 的路由信息，Consumer 则从 Broker 上订阅消息。当有消息时，Broker 会将它推送到 Consumer，则 Consumer 调用评判微服务进行评判，然后将结果返回给 Other Modules。

3　实现与部署

3.1　在线评判流程实现

在线评判是整个系统的核心功能，流程比较复杂，以时序图形式表示，如图 4 所示。其中，ID（提交编号）是由服务端根据本次提交代码、提交用户、时间和题号保存到数据库而生成。提交用户根据返回的 ID 查询评判结果。

3.2　系统部署

3.2.1　Docker 容器

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用及依赖包并将其置于一个可移植的容器中，之后发布到任何流行的 Linux 或 Windows 操作系统的机器上，也可以实现虚拟化。容器完全使用沙箱机制，相互之间不会有任何接口[15]。容器技术是支撑现代云原生应用的基石，在云服务器上实施容器集群服务是业界的新兴趋势[16]。微服务因维护和部署门槛高，越来越依赖容器技术，Docker 技术已经成为当前部署微服务应用的首选。

Java 语言的 Docker 镜像文件制作步骤如下：

（1）编写 Dockerfile 文件。①指定基础镜像：FROM openjdk：8-jdk-slim；②指定镜像的维护者：LABEL maintainer=scau；③将源目录的文件复制到容器的默认目录：COPY target/*jar /backend.jar；④指定容器启动时执行镜像的入口命令：ENTRYPOINT [“java”，”-jar” “/backend.jar”]。

（2）构建指定标签的镜像文件。在安装有 Docker 的系统上，执行如下命令将在当前目录构建标签为 backend：v1 的镜像文件：docker build -t backend：v1。

（3）标记镜像文件。执行如下命令将其归入腾讯云镜像仓库：docker tag backend：v1 ccr.ccs.tencentyun.com/oj_hub/backend：v1。

（4）镜像上传。执行 docker push ccr.ccs.tencentyun.com/oj_hub/backend：v1 命令将镜像上传到腾讯云的镜像仓库。

3.2.2　微服务部署

微服务架构部署需要按照一定顺序才能确保系统正常运行。具体步骤如下：

（1）Nginx。①修改 Nginx 的配置文件 conf.d，设置 proxy_passhttp：//backend：8080/；为反向代理地址，并将 conf.d 保存到 CFS；②将前端的静态页面和 conf.d 分别挂载到 Docker 容器的/usr/share/nginx/html 和/etc/nginx/目录下；③定义服务名为 nginx，使用端口 8188；④启动 Nginx。

（2）RocketMQ。首先部署 NameServer：①定义其服务名

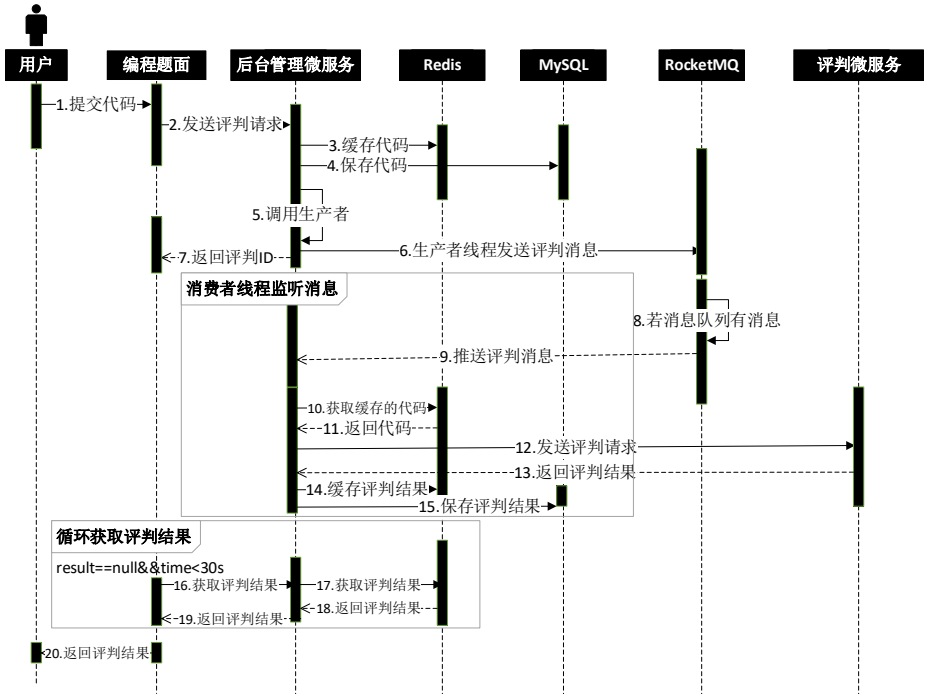

Fig. 4 Online judging sequence diagram

图4 在线评判时序图

为mq-namesrv；②设置端口9876；③启动NameServer。然后部署Broker，设置两个并行工作的主Broker和两个完成数据同步的从Broker，修改配置文件broker.conf及启动步骤：①添加NameServer的地址namesrvAddr=mq-namesrv：9876；②设置Broker的角色：主角色brokerRole= ASYNC_MASTER，从角色brokerRole=ASYNC_SLAVE；③定义Broker的服务名：主角色服务名broker-master：10911，从角色服务名broker-slave：10921；④挂载配置文件broker.conf；⑤设置启动命令为/home/rocketmq/rocketmq-4.5.0/bin/mqbroker；⑥设置启动参数为：-c /home/rocketmq/rocketmq-4.5.0/conf/broker.conf；⑦启动主Broker和从Broker后，创建评判消息的主题为"Judge"。

（3）Redis。①定义服务名为redis，并设置端口映射为6379；②启动Redis。

（4）Testcase。①定义服务名为testcase，使用默认端口80；②挂载CFS的测试用例文件；③启动Tesetcase。

（5）JudgeServer。①定义服务名为JudgeServer，使用默认端口80；②挂载CFS的测试用例文件；③启动JudgeServer。

（6）SpringBoot Application。①将（2）—（5）中部署的微服务的服务名和相关端口写入配置文件；②定义服务名为backend，并设置端口为8080；③设置CPU使用率超过60%进行扩容，低于60%时进行缩容；④启动后台管理微服务。

4 运行与测试

4.1 系统运行

系统已经实现并成功部署在腾讯云，用户可以通过PC端和移动端的浏览器http://120.79.57.23：8188登录系统。以系统管理员身份登录，系统后台部分界面如图5所示（整图扫OSID码可见）。

Fig. 5 System background interface

图5 系统后台界面

4.2 功能测试

针对系统的在线评判功能，设计了部分测试用例，具体如表1所示。"预期输出"有3种状态，其中"通过"表示源代码编译正常且通过了所有的测试用例；"编译出错"表示源代码存在语法错误等问题；"错误"表示源代码通过了编译但其输出与正确的答案不相符。

4.3 性能测试

使用Apache Jmeter对系统进行性能测试。

（1）压力测试。测试500并发量/s的统计数据的屏幕截图如图6所示，其中执行失败为0个，错误率为0.00%；高达99%的请求响应时长（Response Times）在898.84 ms以内，完全符合评判要求；吞吐量（Throughput）为209.82事务/秒，可以应对高并发情形。

（2）动态扩容。根据单台服务器不扩容和多台服务器

Table 1　Test case

表 1　测试用例

测试用例编号	测试标题	重要级别	预置条件	输入	操作步骤	预期输出	测试结果
oj-001	系统在线评判功能测试	高	1.登录系统 2.选择编程题	源代码	1.输入正确的源代码 2.点击提交按钮 3.等待评判结果	通过	正常
oj-002	系统在线评判功能测试	高	1.登录系统 2.选择编程题	源代码	1.输入语法错误的源代码 2.点击提交按钮 3.等待评判结果	编译出错	正常
oj-003	系统在线评判功能测试	高	1.登录系统 2.选择编程题	源代码	1.输入语法正确的但答案不正确的源代码 2.点击提交按钮 3.等待评判结果	错误	正常

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	500	0	0.00%	385.05	38	1332	372.00	667.80	767.90	898.84	209.82	73.56	128.68
HTTP Request	500	0	0.00%	385.05	38	1332	372.00	667.80	767.90	898.84	209.82	73.56	128.68

Fig. 6　Statistics testing 500 concurrency per second

图 6　测试 500 并发量/s 统计数据

动态扩容的原则,按照 100~500 并发量/s 请求进行测试,依据完成并发量的评判总用时进行验证,实际测试数据如图 7 所示。图 7 的纵坐标为评判的总用时,包括前端发送评判请求的时间、后台微服务接受评判请求后发送到评判微服务的请求时间和评判结束后写入数据库的时间 3 部分,单位为秒;图 7 的横坐标为每秒的并发量;图例 1*—4*代表运行评判服务器台数。可以看出,单台服务器不扩容,评判的总用时线性增大,会造成响应超时而无效;当多台服务器动态扩容时,评判的总用时则会减少,譬如:当扩容到 4 台服务器时,处理 500 并发量的评判请求,总用时是 131 s,平均 0.2 s 就可以处理一个评判请求,完全能及时响应请求。

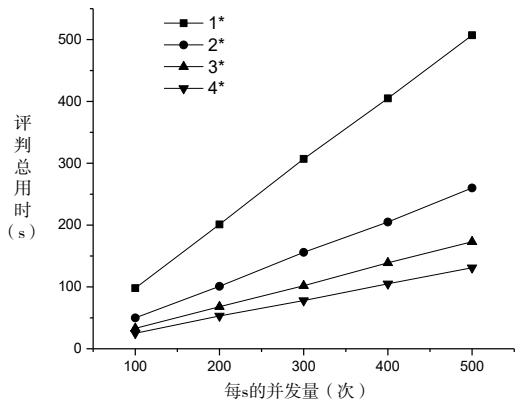

Fig. 7　Line chart of dynamic scaling performance

图 7　动态扩容性能折线图

5　系统应用

为了分析该系统应用效果,现选取 2021 年春季和 2022 年春季 Java 语言程序设计课程进行比较,课程对比数据表如表 2 所示。

说明:因为考试题目有 5 道,比较多且都是未做过的新题,大部分学生得分不高,所以 2*中考试部分的占比定为 10%。

对期末考试的编程题得分进行分析,得分情况如表 3 所示。其中,题目 A1 和 A2 是学生平时训练过的原题,两份试卷都有相类似的题目。题目 C 是 2021 春季考试的编程题,

Table 2　Course comparison data

表 2　课程对比数据

比较	2021 春季	2022 春季	说明
班级(个)	12	12	
学生(人)	353	342	
实验(个)	7	7	
题目(个)	30	30	
小测(次)	0	5	1 题/次
考试(次)	1*	1**	1*:5 道编程题;1**:10 道客观题和 1 道编程题
实验得分	2*	2**	2*:公式 2+考试*10%;2**: 公式 3

分值是 10 分,而在 2022 春季考试的编程题中不采用题目 C,此 10 分由题目 B1 和 B2 的分值构成,且 B1 和 B2 都不是原题,因此 B1+B2 与 C 等价。计算表 3 中 B1+B2,将它们的和与 C 对应,然后画成柱状图,如图 8 所示。

Table 3　Scores of programming questions in the final exam

表 3　期末考试编程题得分

题目	2021 春季	2022 春季	说明
A1	3.45	4.82	5 分,实验做过的原题
A2	3.63	4.91	5 分,实验做过的原题
B1		3.42	5 分,训练过的知识点,不是原题
B2		3.37	5 分,综合训练过的多个知识点,不是原题
C	5.78		10 分,综合训练过的多个知识点,不是原题
总分	12.86	16.52	

2022 年春季情况好于 2021 年春季情况:①每个小题的得分都有提高;②做过的题目复现效果不错,语法错误少;③知识点掌握较好,即使非原题,得分也有一定提高,而且

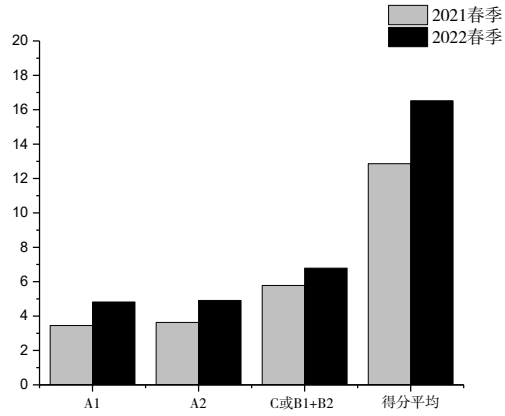

Fig. 8 Score comparison of programming questions in the final exam

图8 期末考试编程题得分比较

编程思路多样,死记硬背、生搬硬套的少了。因此,学生通过训练获得小测资格,增加了学习投入。

6 结语

本文应用微服务和Docker技术对现有在线评判系统进行重新设计与实现,通过增加小测和考试的过程管理环节提高了学生对编写程序的学习投入,通过微服务部署在云平台实现了动态扩缩。今后需在如下方面作进一步研究:①通过分析系统使用过程中CPU、RAM、外存配额、Docker容器参数等有关数据,设计一个更动态的扩缩策略;②通过对学生实验全过程数据进行分析,构建完善的和可视化的学生实验投入度评价模型;③通过加入查重算法,以判断学生是否在考试过程中直接抄袭其他用户的程序题源码。

参考文献:

[1] WAN H, WU D, LIU S R, et al. Practice of online program evaluation system for remote teaching and testing in fighting against epidemic[J]. Software Guide, 2021, 20(12): 230-234.
万海,吴迪,刘思然,等. 疫情下在线评测系统远程教学与机考实践[J]. 软件导刊,2021,20(12):230-234.

[2] LIU L, WANG Q. Research on the characteristics and influencing factors of students' learning engagement in blended teaching model[J]. Technology in Modern Education, 2021, 31(11): 80-86.
刘玲,汪琼. 混合教学模式下学生学习投入的特点及影响因素研究[J]. 现代教育技术,2021,31(11):80-86.

[3] YANG Z, ZHAO Y, CAI X J, et al. Practice and reflections on remote online testing during the epidemic[J]. Computer Education, 2020, 18(10): 8-11.
杨铮,赵毅,蔡新军,等. 疫情期间远程在线考试的实践与思考[J]. 计算机教育,2020,18(10):8-11.

[4] LIU X, LI B T, TANG Y, et al. OBE-based introduction to algorithm course assessment and achievement evaluation[J]. Computer Education, 2021, 19(1): 163-167.
柳欣,李保田,唐艳,等. 基于OBE的算法导论课程考核与达成度评价[J]. 计算机教育,2021,19(1):163-167.

[5] LI Y G, LI X M, WU Y, et al. A service-oriented TT&C service bus system of survey ship[J]. Computer Engineering and Science, 2020, 42(8): 1345-1351.
李永刚,李祥明,吴云,等. 面向服务的测量船测控服务总线系统[J]. 计算机工程与科学,2020,42(8):1345-1351.

[6] ZHANG M, LIU W, GUO Q. Design and implementation of video management system based on cloud native[J]. Software Guide, 2022, 21(5): 141-144.
张咪,刘文,郭庆. 基于云原生的视频管理系统设计与实现[J]. 软件导刊,2022,21(5):141-144.

[7] CUI C, ZHOU W. Construction design of the safety management system based on microservice architecture[J]. Software Guide, 2021, 20(2): 159-164.
崔灿,周伟. 基于微服务的工地安全管理系统架构设计[J]. 软件导刊,2021,20(2):159-164.

[8] SHI L P, ZHU Z, ZHOU J S, et al. Load balancing mechanism for microservice architecture in cloud-based systems[J]. Computer Engineering, 2021, 47(9): 44-50.
施凌鹏,朱征,周俊松,等. 面向微服务架构的云系统负载均衡机制[J]. 计算机工程,2021,47(9):44-50.

[9] DAI L C, WANG C. Reconstruction and optimization of online lawyer website under microservice architecture[J]. Software Guide, 2021, 20(9): 144-149.
代立晨,王晨. 微服务架构下的在线律师网站重构与优化[J]. 软件导刊,2021,20(9):144-149.

[10] JIAO Y, LI M, WANG H, et al. Optimization of online shopping system based on distributed high availability cluster[J]. Software Guide, 2022, 21(6): 183-187.
焦宇,李民,王欢,等. 基于分布式高可用集群的网购系统优化[J]. 软件导刊,2022,21(6):183-187.

[11] YUAN X C, FU G, BI J Z, et al. Survey on data caching technology of distributed dataflow system[J]. Big Data Research, 2020, 6(3): 101-116.
袁旭初,付国,毕继泽,等. 分布式数据流计算系统的数据缓存技术综述[J]. 大数据,2020,6(3):101-116.

[12] LIAO L F, ZHANG X P. Design and implementation of intelligent visualization command system based on iOS[J]. Computer Engineering and Design, 2020, 41(11): 3269-3274.
廖列法,张幸平. 基于iOS的智能可视化指挥系统设计与实现[J]. 计算机工程与设计,2020,41(11):3269-3274.

[13] LIN P R, CHEN Z R, SHI X Q. Highly concurrent multi-threaded competitive shared resource architecture[J]. Computer Engineering and Design, 2020, 41(11): 3282-3288.
林平荣,陈泽荣,施晓权. 高并发多线程竞争共享资源架构[J]. 计算机工程与设计,2020,41(11):3282-3288.

[14] HUANG C, BAI L P. Dual-architecture application paralleling and flow switching scheme based on Nginx-F5[J]. Computer Systems & Applications, 2022, 31(3): 351-355.
黄晨,柏路平. 基于Nginx-F5的双架构应用并行及流量切换方案[J]. 计算机系统应用,2022,31(3):351-355.

[15] LUO C, CUI Y, LIN Y S. Container migration method based on bandwidth prediction and adaptive compression[J]. Computer Engineering, 2022, 48(5): 200-207.
罗成,崔勇,林予松. 基于带宽预测与自适应压缩的容器迁移方法[J]. 计算机工程,2022,48(5):200-207.

[16] XIE D M, HUANG L, HUANG J J, et al. Design and implementation of container cluster service for muti-cloud[J]. Software Guide, 2022, 21(6): 169-175.
谢冬鸣,黄林,黄进军,等. 多云容器集群服务的设计与实现[J]. 软件导刊,2022,21(6):169-175.

(责任编辑:孙 娟)