Design and Implementation of Intelligent Academic Affairs System in Colleges and Universities Based on Microservice Architecture

Yuyan Gai, *Junfei Zhao, Qingxi Liu, Yingjie Wu, and Guohao Zhou

Harbin Institute of Information Technology, Harbin,150431, China

*Corresponding author: Junfei Zhao; email: zhaojunfei@gmail.com

Abstract. In order to solve the defects of traditional academic affair systems, for example bad expansion ability, weak support for concurrent processing, and not enough combination with intelligent teaching situations, this article puts forward a new type of intelligent academic affair system which is based on microservice structure. This system uses a front part/back part split construction structure. The forward end employs React plus Umi. js acts as the technology stack, while the back-end is constructed upon the Spring Boot + Spring Cloud Alibaba framework, hence it integrates key technologies which are Redis, Redisson, MinIO, FastGPT, WebRTC, and FFmpeg. It this includes the whole life cycle of academic affair services, containing course management, time arrangement, student sign-up management, and teaching task management. One mechanism of multi-dimension optimization has been designed, it includes preprocessing of concurrent requests based on Redis, resolution of conflicts of distributed locks based on Redisson, and support of intelligent decisions by FastGPT. The experiment outcomes give demonstration that this system can hold one hundred thousand concurrent user visit requests, the average time of response is 87.3 milliseconds, and the scheduling conflict ratio is brought down to 1.2 percent. It has realized the organic combination of academic affair handling and smart teaching services, hence greatly promoting the efficiency of academic affair handling and the level of teaching services, hence giving a dependable technical plan for the digitized change of higher education.

Keywords: Academic affairs system; microservice architecture; concurrency performance optimization; intelligent teaching integration; distributed lock

1 Introduction

1.1 Research Background

Along with the deep going of digital transformation inside higher education, the bottleneck problems of traditional academic affair systems are getting more and more obvious: these systems are for the most part constructed on the basis of monolithic

C. V. Garcés et al. (eds.), *Proceedings of the 6th International Conference on Internet, Education and Information Technology (IEIT 2026)*, Atlantis Highlights in Social Sciences, Education and Humanities 53,
https://doi.org/10.2991/978-94-6239-733-0_15

architectures, and hence as the quantity of students and courses goes up, the difficulty of function expansion and maintenance ascends sharply; The increasing of simultaneous visiting requests when busy time like class choosing often causes system break down or slow reply speed; At the same time, the combination of newly arisen teaching patterns such as net mutual teaching and intellectual arrangement with school academic management systems has already become an urgent demand that current universities must face. Current studies on academic affair systems mainly concentrate on optimizing functions of single modules or promoting local technologies, therefore they lack a systematic design for the whole academic affair business chain. Though certain systems have already employed distributed framework structures, they nevertheless still possess deficiencies with regard to the efficiency of concurrent processing and the integration of intelligent services.For example, the SpringBoot teaching system proposed in reference [1] improves development efficiency, but does not solve the high concurrency problem in key scenarios[1].

Recent research demonstrates significant advances in intelligent academic systems. Haider et al. proposed a self-learning genetic algorithm based on deep reinforcement learning for course scheduling, utilizing Q-learning for adaptive adjustment of crossover and mutation parameters to enhance search capability and optimization accuracy [2]. Dohrmann et al. developed an AI-based tool for curriculum-based course timetabling at the University of Potsdam, using answer set programming to achieve conflict-free scheduling for approximately 160 courses with solving times under 170 seconds [3]. Liu et al. presented a gradual optimization approach combining genetic algorithms with dynamic programming, demonstrating superior performance on 520 real scheduling tasks at Beijing Forestry University [4].In high-concurrency system design, Li et al. implemented a microservice-based automatic grading system using Spring Cloud architecture combined with Redis and Kafka, supporting peak concurrency of 5,000 requests per second with response times under 1 second [5]. Kaptosv conducted comprehensive evaluation of Redis caching strategies, showing that cache-aside mode significantly decreases average response time from 1146 ms to 323 ms and increases throughput up to 226 requests per second in high-traffic applications [6].These studies collectively confirm the effectiveness of combining intelligent algorithms with modern distributed architectures for educational applications. However, systematic integration of genetic algorithm optimization, microservice architecture, and high-concurrency processing within a unified academic affairs framework remains insufficiently explored. This paper addresses this gap through comprehensive lifecycle management system design.

1.2 Research Objectives and Contributions

This paper has the purpose to carry out design and realization of a high property, expandable, and intelligent academic affair system for satisfying the multi-dimensional requirements of universities. Key core contributions contain putting forward a layered structure for the academic affairs system which is based on microservices, therefore realizing decoupling of business modules and flexible function expansion. A multi-layer simultaneous disposing mechanism is designed for key scenes such as course choosing, therefore effectively resolving the "course choosing avalanche" issue of

traditional systems. AI and real-time interactive teaching technologies are combined together, in order to realize the organic combination of academic affair management and intelligent teaching service. Comprehensive performance examination is carried out from many dimensions, which include concurrency, reliability, and response time, and the application worth of the system is verified by means of actual deployment inside universities.

2 Related work

2.1 Application of Microservice Architecture in Education Scenarios

Micro service architecture has become the main current solution for constructing large-scale distributed systems because of its merits including modular character, independent arrangement deployment, and flexible expansion capability. In the domain of clever campus areas, the micro-service-supported complete life-circle service flat forms have gotten use, thus achieving the combination of broken services into one body. But as for the usage of microservices inside academic affair systems, it is still necessary for us to solve the problems of service granularity division and the communication efficiency between every service.

2.2 Concurrency Processing Technology for Academic Affairs Systems

The systems for students to choose courses are a typical scene with many concurrent operations in the academic management systems. Current resolving methods mainly promote handling capacity via buffer storage and data base reading and writing division. Distributed locking mechanisms which base on Redisson can effectively solve problems of data consistency when concurrent access happens, but they need to do targeted optimization according to specific features of academic affairs work[7].

2.3 Integration of Intelligent Technology with Academic Affairs System

The incorporation of AI techniques such as big language models into the academic affairs system can supply services such as intelligent question answering and course suggestion; the usage of WebRTC and FFmpeg technical methods makes real-time interactive teaching and video resource processing be achieved, hence expanding the scope that academic affairs system can apply from pure management work to the aspect of teaching support[7].

3 System Architecture Design and Key Technologies

3.1 Overall Architecture Design

This system uses a five-layer structure, which contains a front-end display layer, an API gateway layer, a micro service layer, a middle layer, and an external connection layer.

The front end display level is constructed upon React+Umi. js frame, which combines AntDesign, AntV, and Quill parts to offer reaction-type interfaces for students, teachers, managers, and other such roles, thus supporting visit from both PC and moving equipment. The API gateway layer, it is mainly constructed upon Spring Cloud Gateway, it integrates Sa-Token for authentication and authorization work, and processes request routing, load balancing, traffic control, so that unified management of external access can be realized. The micro-service level is separated into 12 core business service items (course management, arrangement management, student sign-up management, etc.) according to the business chain of academic affairs, each service is independently developed and deployed on the basis of Spring Boot. The middle layer mainly combines middle software like Redis, Redisson, MinIO, and FastGPT, thus offering the system basic functions like caching, distributed locks, object storage, and intelligent calculation. The outside integration level combines Alibaba Cloud SMS SDK, DingTalk SDK, OCR identification, and Nexus private server to achieve functions including message notice, third-party cooperation, and automatic information input. The entire framework structure design of the system is displayed in Figure 1.

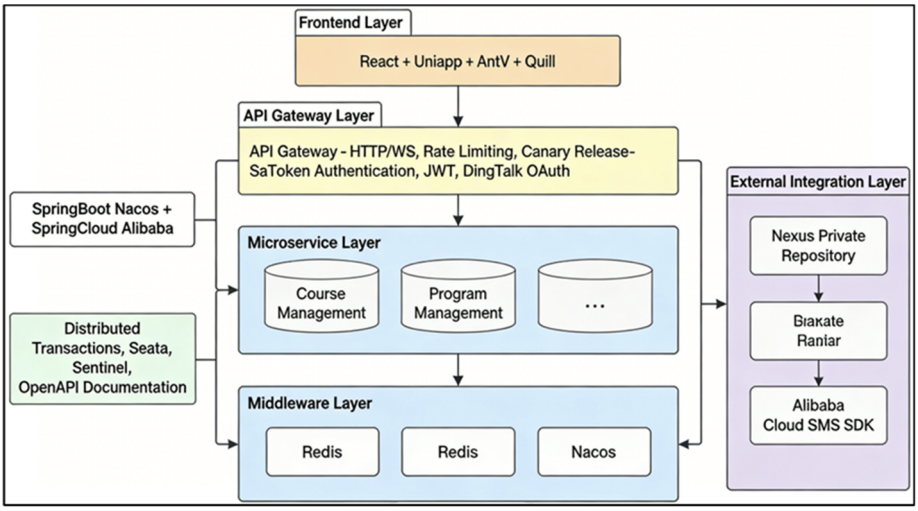

Fig. 1. System Overall Architecture Design

3.2 Key Technology Mechanisms

The system comprises four key technical modules: microservice partitioning and communication, high-concurrency course selection processing, intelligent course scheduling algorithm, and real-time interactive teaching support. A stable technical mechanism is built through the collaboration of core technologies: Nacos, Redis, FastGPT, and WebRTC. The core technical mechanism architecture diagram is shown in Figure 2.

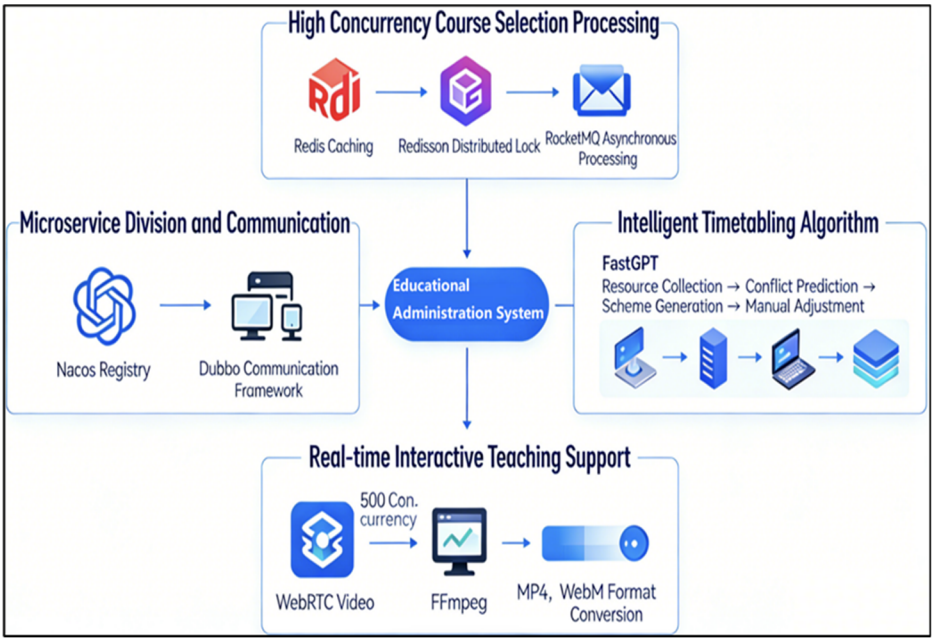

Fig. 2. Core Technology Mechanism Architecture Diagram

High-concurrency course selection mechanism. For the course choosing situation, a three-layer concurrent processing mechanism has been designed by us. The pre-processing storage area saves course quotas and student choosing conditions in a Redis cluster for real-time searching, hence decreasing database visit pressure[8]. The distributed lock controlling mechanism utilizes a Redisson fair lock for controlling concurrent course selection requests and therefore prevents over-selection[9]. The asynchronous handling mechanism, after cache pre-deduplication achieves success, transmits a message to a RocketMQ queue for asynchronous database renewing, thus promoting request response velocity.

Intelligent scheduling algorithm. Integrating FastGPT to build an intelligent scheduling model, this system takes into account factors such as teacher time, classroom resources, and course conflicts, and generates a Pareto optimal solution through iterative optimization. The genetic algorithm which is optimized by FastGPT uses a multi-group evolution strategy that has the parameters as follow. The quantity of population is established as 200 individual persons, every one of which is coded as a multi-dimension chromosome that includes teacher distribution work, classroom arrangement assignments, and time slot placing plans. The fitness function carries out balance on three goals: teacher's preference satisfying with 40 percent weight, classroom use efficiency with 30 percent, and course conflict reducing to minimum with 30 percent. The iteration process that carries out 150 generations has adaptive working item ratios. The probability of crossover decreases in a linear way from 85% to 65%, for the purpose of encouraging early-stage exploration and later-stage convergence. Mutation probability has a reduction from 15% to 5%, it keeps diversity at the beginning and thus stabilizes

those elite solutions. FastGPT carries out dynamic adjustment of selection pressure that is between 1.2 and 2.0 on basis of historical conflict patterns. Elitism keeps the uppermost 5% solutions, tournament selection adopts competition scale 3, and termination is triggered when maximum generations are reached or when fitness improvement stays under 0.01% through 20 continuous generations. This setting guarantees the most excellent solutions in the three-minute processing time interval.First, it collects resource constraint information such as teacher schedules, classroom capacity, and course requirements; second, it builds a conflict probability model based on historical data to predict potential scheduling conflicts in advance; third, it uses a genetic algorithm optimized by FastGPT to generate multiple scheduling schemes within 3 minutes; and finally, it supports drag-and-drop manual adjustments, with the system automatically avoiding new conflicts arising from the adjustments.

4 System Implementation and Functional Modules

4.1 Core Functional Modules

The system covers the entire lifecycle of academic affairs management and includes 12 core functional modules, covering course management, scheduling, student registration management, and teaching task management. The core functional modules of the system are shown in Table 1.

Table 1. Core Functional Modules of the System

Serial Number	CoreModules	Font size and style
1	CourseManagement	Course information entry, modification, and query; management of supporting resources such as syllabi and textbooks.
2	PlanManagement	Teaching plan development, adjustment, and execution progress tracking.
3	CourseOpeningManagement	Course offering approval; matching of teaching resources such as teachers and classrooms.
4	ScheduleManagement	Intelligent timetable generation; supports conflict detection and manual adjustments.
5	TeachingManagement	Teaching task allocation, lesson plan management, and teaching progress monitoring.
6	WorkloadManagement	Automatic calculation of teacher workload based on class hours/class size; supports custom rules.
7	TaskBookManagement	Online generation, review, and archiving of teaching assignments.
8	StudentStatusManagement	Full-process electronic management of student records, including enrollment, transfer, and leave of absence.
9	StudentStatusDataManagement	Student record statistics, export, and historical data archiving.

4.2 Deployment Architecture

This system has been put into container form through the use of Docker and Kubernetes. The dispose structure contains three API gateway nodes in order to realize load balancing and high availability; six micro service nodes, whose core services are deployed on two nodes; a Redis grouping (3 main nodes and 6 secondary nodes) for high-speed data storing and distributed lock controlling; and one MinIO cluster (4 nodes) which is used for depositing non-structured data like teaching videos and attached documents. The topology of system deployment is displayed in Figure 3.

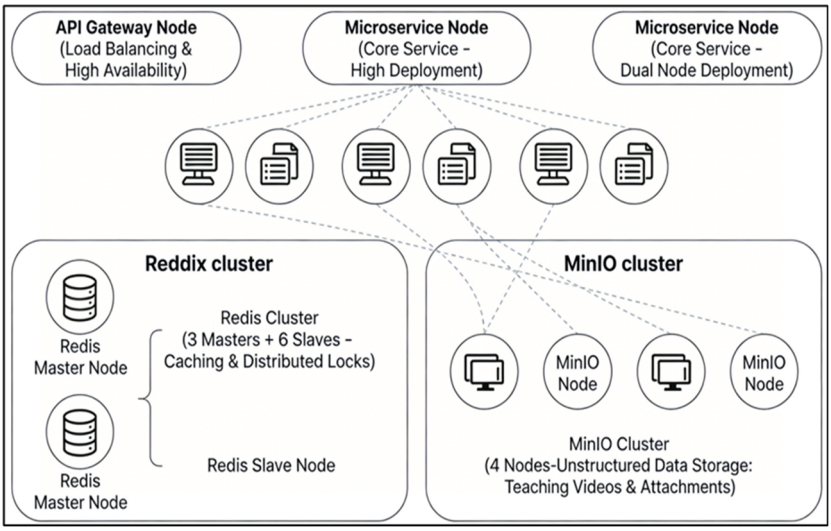

Fig. 3. System Deployment Topology Diagram

5 Modules Experimental Design and Results Analysis

5.1 Experimental Environment

The experimental environment was configured according to common standard deployment specifications for university information systems. The experimental environment is shown in Table 2.

Table 2. Experimental Environment

Hardware/Software	Configuration instructions
Server CPU	IntelXeonGold6248R（24 cores and 48 threads）
Server Memory	128GBDDR42933MHz
Hard Disk	4TBSSD+10TBHDD
Operating System	CentOS7.9
JDK Version	JDK17
Database Version	MySQL8.0、Redis7.0
Testing Tools	JMeter5.6、LoadRunner2023

5.2 Test Metrics and Methods

The testing of this system has covered four big indicators: concurrent working performance, dependability, function realization, and expansion ability. Concurrency working speed was tested through simulating 10000, 30000, 50000, 80000, and 100000 concurrent users visit key interfaces such as course choose and timetable inquire use JMeter (every session continues 30 minutes). We carried out a 72-hour uninterrupted running experiment, which simulates 5% service node malfunction, for the verification of system usability and data consistency. For 12 core modules, 200 test cases have been designed, which cover normal running, boundary situations, and non-normal input, to be used for testing the completion rate of core functions and the accuracy of conflict detection. We increased the quantity of concurrent users from 50,000 to 150,000, and recorded the efficiency of resource expansion and the time of service expansion, in order to finish the test of scalability.

5.3 Experimental Results and Analysis

Concurrency performance test results. When the quantity of simultaneous users is not larger than 80,000, the average system reaction time still keeps inside 100ms; When the quantity of concurrent users arrive at 100,000, the average responding time is only 87.3ms, and the throughput keeps above 1200TPS, this is obviously better than the traditional academic affair system (average responding time >300ms, throughput <500TPS). The test results of concurrent performance are displayed in Table 3.

Tab**le 3.** Concurrency Performance Test Results

Concurrent users	Average response time（ms）	Throughput（TPS）	Error rateError rate（%）
10000	23.7	4218	0.02
30000	41.2	3645	0.05
50000	58.6	2763	0.08
80000	75.4	1689	0.12
100000	87.3	1246	0.15

The outcomes of tests indicate that the three-layer concurrent handling mechanism which this article designs effectively promotes the system's processing capability under high concurrency: Redis cache lowers the pressure of database, distributed locks guarantee the consistency of data, and therefore resolves the "course selection avalanche" issue that traditional systems have.

Reliability test results. In a 72-hour uninterrupted running experiment, the system's usable rate attained 99.99%, and there was no loss of data or inconsistence of data. When 5% percent of the service nodes had the failure situation, the system can by itself complete the switch to the backup node in the time of 3 seconds, therefore it guarantees that the service does not get interrupted, hence it keeps 100% consistency of the data, thus it has verified the high reliability of the microservice architecture and the fault tolerance mechanism.

Functional test results. In the 200 test cases that we have, 198 have passed, thus the completion rate of core functions reaches 99%. The intelligent arrangement module has realized conflict checking accuracy of 98.8%, it makes conflict rate drop to 1.2%, this figure is far lower than the conflict rate of manual arrangement (about 15%) and also lower than the conflict rate of traditional automatic arrangement systems (about 8%). This has proven that the intelligent arranging arithmetic which is integrated into FastGPT possesses outstanding practical working effect.

6 Conclusion

This intelligent academic affair handling system for universities, which is built on a microservice architecture, uses a front-end/back-end divided architecture and the Spring Boot + Spring Cloud Alibaba core framework. It has the integration of key technologies including Redis, FastGPT, and WebRTC, and 12 core function modules that cover the whole life cycle of academic work have been constructed. Via optimization mechanisms which include three-level concurrent processing, it on effective terms solves the problems that traditional academic affairs systems have: bad scalability, low concurrency support, and insufficient integration of intelligent teaching scenes. Experimental outcome make known that this system is able to support 100,000 concurrent user visit, having an average respond time of 87.3ms, usability of 99.99%, and a core function complete rate of 99%. It greatly promotes the work efficiency of academic affairs management and the level of teaching service, hence it offers a dependable technology scheme for the digitized transformation of higher education.

Acknowledgments.

We thank Harbin Institute of Information Technology for providing the experimental environment and data support, which enabled the research to proceed smoothly.

References

1. Li J W, Han S H. Research on Design Methods of Online Education Platform Based on Micro-services and TCC Distributed Transaction[J]. Journal of Jiangsu Shipping College, 2024, 21(1): 45-52.DOI:10.3969/j.issn.1671-9891.2020.02.011.
2. Haider S A, Ahmad K M, Zahid A, et al. Genetic and Deep Reinforcement Learning-Based Intelligent Course Scheduling for Smart Education[C]//Proceedings of the 2024 International Conference on Artificial Intelligence and Teacher Education (ICAITE 2024). Beijing, China: ACM, 2024: 1-8. DOI:10.1145/3702386.3702398
3. Dohrmann C, Lucke U, Schaub T, et al. AI-Based Tool for Curriculum-Based Course Timetabling at the University of Potsdam[C]//Proceedings of EUNIS 2024 Annual Congress. Athens, Greece, 2024: 188-199.
4. Liu Y, et al. Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming[J]. Algorithms, 2025, 18(3): 158. DOI:10.3390/a18030158

5. Li Z, et al. Design and Implementation of a High Concurrency Oriented Automatic Grading System for Application Essays[C]//Proceedings of the 2025 International Conference on Big Data and Informatization Education. ACM, 2025. DOI:10.1145/3729605.3729617
6. Kaptosv L. Using Redis for Caching Optimization in High-Traffic Web Applications[J]. International Journal of Emerging Trends in Computer Science and Information Technology, 2025, 5(4): 28-36. DOI:10.62225/2583049X.2025.5.4.4839
7. Smirnov N, Tomforde S. Real-time rate control of WebRTC video streams in 5G networks: Improving quality of experience with Deep Reinforcement Learning[J]. Journal of Systems Architecture, 2024, 148: 103066.DOI:10.1016/j.sysarc.2024.103066
8. Jordanov J, Simeonidis D, Petrov P. Containerized Microservices for Mobile Applications Deployed on Cloud Systems[J]. International Journal of Interactive Mobile Technologies (iJIM), 2024, 18(10): 48-58.DOI:10.3991/ijim.v18i10.8564
9. Liu H P, Zhu W H, Fu S Y, et al. A trend detection-based auto-scaling method for containers in high-concurrency scenarios[J]. IEEE Access, 2024, 12: 71821-71834. DOI:10.1109/ACCESS.2024.3397608