Postprint: Construction of a High-Performance Application Service Platform Based on Node.js

Authors: Li Qian

Date: 2023-10-08T00:00:00+00:00

Abstract

This paper takes the Marxist view of journalism learning competition organized by Xinhua News Agency as a case study, systematically analyzes the design challenges of high-performance application service platforms, and develops a competition answering application service platform based on Node.js. The platform achieves a high-performance responsive architecture, hierarchical management of participants/organizations, activities, and questions, multi-dimensional statistical analysis, and data security assurance through comprehensive security designs. It can handle high-concurrency business scenarios, effectively ensuring the smooth execution of the Xinhua News Agency Marxist view of journalism learning competition, while possessing the capability to support additional similar activities.

Full Text

Building a High-Performance Application Service Platform Based on Node.js

Abstract

This paper examines the design challenges of high-performance application service platforms through the lens of the Xinhua News Agency Marxist Journalism Learning Competition. We present a competition answering platform built on Node.js that achieves high-performance responsiveness, hierarchical management of participants, organizations, activities, and questions, multi-dimensional statistical analysis, and robust data security. The platform successfully handles high-concurrency scenarios, ensuring the smooth execution of the Xinhua competition while demonstrating the capability to support similar large-scale events.

Keywords: High Performance; High Concurrency; Application Service Platform; Node.js; CDN; System Security

The Xinhua News Agency' s "Mobile Online Learning Competition on Marxist Journalism" recently concluded successfully. During the 15-day online answering period, 70 subsidiary units participated with full engagement, and over 12,000 staff members downloaded the competition APP, achieving nearly 100% participation. This created a strong learning atmosphere and effectively advanced Marxist journalism education throughout the agency. Behind this success, technical support was critical. Within just three months, our three-person R&D team completed the full independent development of the competition platform from conception to deployment. This platform enabled rapid construction of the entire competition framework. So how exactly does this platform support such events technologically?

1. Platform Design Challenges

Our vision for this "platform" was not a one-off backend application, but a reusable competition answering platform that could support multiple events for different users. Through extensive discussions with the event organizers, we established the business scenario for this Marxist journalism learning competition: the competition consists of multiple chapters with randomized questions in a progressive 闯关 (level-clearing) format. Participants can challenge higher-ranked competitors, with both parties receiving potential bonus points. Additionally, participants who complete all chapters can continue answering questions independently to earn unlimited bonus points. All questions display correct answers in real-time for learning purposes. Based on this business scenario and our platform vision, we identified the following key design challenges.

1.1 Diverse Activity Modes

Competition answering activities include various modes such as level-clearing, challenges, and practice, each requiring flexible configuration capabilities.

1.2 Large Numbers of Participating Units and Personnel

A single competition may target one or multiple units, and each unit may have numerous sub-departments and personnel, resulting in substantial numbers of involved individuals and organizational units. Without effective management functions for participating units and personnel, the platform would incur high management costs and reduced organizational efficiency.

1.3 High Concurrent Access

Activities experience varying degrees of concurrent access. For instance, longer-duration competitions typically generate significant concurrent traffic at the beginning and near the end. The system must guarantee normal operation and data integrity under high concurrency while maintaining high performance and responding to data requests promptly.

1.4 Multi-Dimensional Activity Statistics

Statistics during and after competitions are crucial for such events. For participants, intuitive activity statistics are essential for verifying their performance and rankings. For organizers, multi-dimensional statistics serve as important indicators of the competition' s success.

1.5 Flexible and Rapid Response to Business Changes

A platform' s longevity depends significantly on its ability to adapt to evolving business requirements and enable rapid iterative updates.

1.6 Data Security Imperatives

User data and competition data must be well-protected. Any unauthorized data access, tampering, or illegal requests must be strictly prevented.

2. Platform Design

Throughout the design process, our team conducted thorough research and multiple brainstorming sessions to address these challenges comprehensively from functional design, technology selection, and deployment perspectives.

2.1 Functional Design

The platform provides a comprehensive suite of functions and services including personnel management, organizational structure management, question bank management, activity management, statistical analysis, user authentication, and API services to support various competition formats. Through interactive configuration interfaces for activities and levels, the platform supports multiple concurrent competitions and levels, enabling organizers to easily adjust activity parameters. Hierarchical management and batch data import capabilities reduce the management overhead for organizers. Multi-dimensional statistical query functions based on organizational structure and activity levels facilitate real-time monitoring for organizers. API services provide diverse interfaces for competition terminals, enabling flexible user interaction implementations. All platform functions are segmented by role and content permissions, ensuring that only authorized users can perform specific operations.

2.2 Technical Framework

The platform adopts a B/S architecture.

2.2.1 Server-Side The server-side is developed using Node.js, MySQL, and the Express framework, providing RESTful API services. Node.js employs an event-driven, non-blocking I/O model that maximizes execution efficiency and reduces interface latency.

Table 1 Performance Comparison of Node.js and Java in Various Scenarios [**?**]

Scenario	Requests per Second (No Concurrency)	Requests per Second (With Concurrency)
I/O Intensive	Node.js exceeds Java by several times	Node.js performance improves by further multiples
Compute/I/O Balanced	Node.js exceeds Java by several times	Node.js performance improves by further multiples
Long-blocking I/O Requests	Node.js exceeds Java by several times	Node.js performance improves by further multiples

As the data demonstrates, except in compute-intensive scenarios, Node.js outperforms Java by several times in I/O-intensive, compute-I/O balanced, and long-blocking I/O request scenarios. Under concurrent conditions, Node.js performance can improve by additional multiples. Our analysis concluded that the competition answering platform primarily requires rapid request processing with minimal complex computation, making it a typical I/O-intensive application. To achieve optimal performance, we selected Node.js.

Express is a concise and flexible Node.js web application framework that enables rapid development of web request handling logic and API services. Its modular architecture facilitates subsequent functional expansion. We use the relational database MySQL, designing database tables with good request performance and scalability to accommodate potential business changes. The relational nature of MySQL makes it suitable for complex statistical query operations, enabling multi-dimensional data queries based on business requirements.

2.2.2 Browser-Side The browser-side is developed using HTML5, CSS3, and JavaScript, employing the Vue.js framework and Element UI library to create a clean, user-friendly, modular UI interface. Vue.js is lightweight and concise, enabling rapid iterative development. Its component-based architecture ensures excellent scalability for browser-side functionality. The Element UI library provides diverse UI components, delivering an elegant user experience and accelerating the development and assembly of the platform management interface.

2.3 Deployment Design

The platform utilizes CDN (Content Delivery Network) technology to accelerate static content delivery. CDN can automatically distinguish between dynamic and static content on Xinhua' s website, directing user requests to optimal nodes through intelligent load balancing systems [**?**]. This effectively avoids network congestion, enables fastest access, improves service quality, and provides

fast, stable support for accessing competition content while reducing application server load. The following two diagrams illustrate CDN hit rates (dark areas indicate hits) during the competition. CDN usage protected the origin servers and effectively avoided network congestion, supporting access from all 34 domestic provincial administrative regions and over 100 countries and regions worldwide.

Figure 1 [Figure 1: see original paper] CDN Hit Rate Diagram (Domestic)

Figure 2 [Figure 2: see original paper] CDN Hit Rate Diagram (Overseas)

The platform employs load balancing technology to distribute user requests across application servers. The system supports horizontal scaling of components such as load balancers, application servers, and database servers based on bottlenecks under heavy load. Regular system data backups ensure timely recovery in case of accidental data loss.

2.4 Security Design

The platform implements multiple security measures to ensure system and data security: (1) SSL certificates encrypt network requests; (2) Critical information is encrypted at the data source for protection; (3) Servers undergo regular vulnerability detection, baseline checks, and virus scanning.

Note: Figure translations are in progress. See original paper for figures.