Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2025.22841

Research Overview of Intelligent Computing Platform Technology

Xijun Wan

Xidian University, Xi'an, China
wxjlsy999@qq.com

Abstract: With the explosive growth of data and the increasing complexity of computing tasks, traditional computing models can no longer meet the demands. Parallel and distributed processing technologies have become key forces driving innovation in information technology. This paper reviews the research progress of intelligent computing platform technologies, focusing on parallel processing and distributed processing technologies. It primarily examines the basic principles and applications of multi-core processor technology, large-scale parallel algorithms, parallel programming, industrial computer networks, multi-agent systems, and cloud-edge-end architectures. Looking ahead, research on optimizing multi-core processors, innovating parallel algorithms, enhancing the intelligence and security of distributed systems, the deep integration of cloud-edge-end architectures with specific computing tasks, and the collaboration and planning of multi-agent systems will further improve the computational capacity, flexibility, and scalability of intelligent computing platforms.

Keywords: Intelligent Computing, Multi-core Processors, Parallel Algorithms, Parallel Programming, Industrial Computer Networks, Agent, Cloud Computing

1. Introduction

In the current digital wave, information technology has permeated every aspect of social life, reshaping the operational models of various industries. The massive increase in data and the complexity of computing tasks have placed stringent demands on the performance of computer systems. Traditional computing models can no longer meet these demands, and parallel processing and distributed processing technologies have emerged as critical forces driving continuous innovation in information technology. Exploring parallel and distributed processing technologies, uncovering their potential advantages, overcoming existing challenges, and exploring integrated innovation paths are not only frontier topics in computer science research but also practical necessities for driving the digital transformation of various industries and achieving intelligent upgrades.

The research on parallel processing technology dates back to the 1950s, when John von Neumann proposed the concept of "self-replicating cellular automata," which provided new ideas and methods for parallel computing systems, enabling the design of computer systems with self-repair and self-expansion capabilities [1]. In the 1970s, the concept of parallel computing began to take shape, and the first large-scale parallel processing systems (MPP systems) appeared. Early research mainly focused on theoretical and algorithmic aspects, and it was not until the 1990s that MPP systems became mainstream in high-performance computing, with wide applications in fields such as

DOI: 10.54254/2755-2721/2025.22841

scientific research. Since the early 21st century, MPP systems have integrated with cluster computing and cloud computing, and supercomputers have adopted large-scale cluster architectures [2]. Parallel processing technology has found new applications in big data processing and artificial intelligence, with accelerators such as GPUs and TPUs being widely used in parallel computing. Parallel processing technology has been extensively applied in fields such as avionics system computing platforms [3], parallel DCNN optimization algorithms based on ABWO (PDCNN-ABWO) [4], and reconfigurable computers based on multi-core processors [5].

Similarly, the research on distributed processing technology can be traced back to the 1960s and 1970s. As the concept of distributed systems began to take shape, research on theoretical foundations and model development also progressed. In 1969, the Advanced Research Projects Agency (ARPA) of the United States initiated the research on ARPANET, which laid the foundation for distributed networks. In the 1970s and 1980s, early distributed operating systems, such as the Andrew system developed by Carnegie Mellon University, emerged, focusing mainly on distributed file system management and user access control. At the same time, the theory of distributed databases began to rise, with research focusing on issues related to the distribution, storage, and querying of data across multiple nodes. Today, distributed processing technology has been widely applied in various fields. For example, in industrial computer networks, the arc furnace control system based on neural networks [6], the multi-agent-based MES system for manufacturing enterprises [7], and the cloud-based enterprise management distributed data mining system [8] all demonstrate the versatility of distributed processing technology.

Parallel and distributed processing technologies complement each other, both being essential technologies in the field of intelligent computing to improve computational efficiency and processing capabilities. The remaining sections of this paper are structured as follows: Section 2 provides a detailed introduction to parallel processing technology. Section 3 outlines the development and application of distributed processing technologies in industrial computer networks, multi-agent systems, and cloud-edge-end architectures. Section 4 concludes with a summary and outlook on the future development of intelligent computing platform technologies.

2. Parallel processing technology

2.1. Multi-core processor technology

Multi-core processors (typically using RISC technology, with super-scalar and superspeed pipelines, along with on-chip cache structures, and containing over a million transistors) are illustrated in the system block diagram shown in Figure 1. The aim of these processors is to quickly handle complex computing tasks, with a design focus on improving calculation speed and processing capacity. By employing advanced manufacturing processes, shrinking transistor sizes, and increasing integration, the number of cores and functional modules in the processor are increased, thus enhancing parallel processing capabilities. Additionally, optimizing the instruction set architecture allows the processor to execute instructions more efficiently, reducing instruction execution cycles. Superscalar technology enables the processor to execute multiple instructions in a single clock cycle, while pipelining divides the instruction execution process into multiple stages, enabling overlapping execution of instructions and increasing processor throughput.

DOI: 10.54254/2755-2721/2025.22841

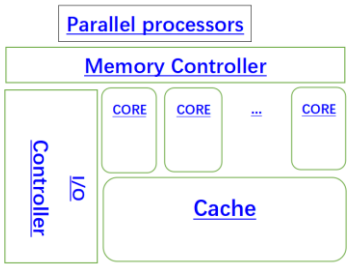

Figure 1: Block diagram of multi-core processor system

Multi-core processors are widely used in fields requiring high computational power, such as supercomputers, data centers, and artificial intelligence (AI) computing. In AI, multi-core processors accelerate complex neural network training and inference tasks, fostering advancements in speech recognition, image recognition, and other technologies.

By integrating multiple cores, multi-core processors enable thread-level and task-level parallelism within the chip, significantly boosting computational power. Compared to single-core processors, multi-core structures break down complex functional chips into multiple relatively simple processor cores, lowering the design complexity of each core and facilitating optimization. Multi-core processors make more efficient use of on-chip resources, fully leveraging parallelism in programs to enhance performance. Since the interconnection distance between processor cores is reduced, data transmission bandwidth is improved, allowing for more efficient resource sharing between cores, while also lowering chip power consumption.

Multi-core processors are now commonly used in desktop computers, servers, and mobile devices. In the server domain, multi-core processors can simultaneously handle multiple user requests, improving concurrency and response time. In mobile devices, multi-core processors help reduce power consumption while maintaining performance, extending battery life.

Although the integration limits of multi-core architecture chips are approaching, and their performance potential is vast, the specific development trends remain a topic of debate. While multi-core CPUs offer significant advantages, they also pose challenges such as shared resource management, system performance efficiency, algorithm and programming complexity, and code migration.

2.2. Large-scale parallel algorithms

The development of multi-core systems presents challenges for parallel computing models and algorithm scalability, with massive data computation becoming a critical area of research.

Computational models provide an abstraction of computer hardware and software, supporting algorithm design and programming. Traditional parallel computing models have become increasingly complex under the development of multi-core systems, due to the constant introduction of new parameters reflecting machine characteristics, making them difficult to apply in practical algorithm design. Hence, layered parallel computing models are needed to adapt to changes in system architecture and provide better support for algorithm design.

Scalability assessment is crucial in multi-core systems, with standards such as equal efficiency, equal speed, and average latency being used to evaluate performance. Modeling and calculating these metrics have become research hotspots. For example, in shared-memory multi-core systems, storage access bottlenecks can be seen as serial components that cannot be parallelized. These inherent serial parts must be carefully considered, as they significantly impact scalability.

With the development of multi-core systems and cloud computing, large-scale data processing has become even more crucial in parallel algorithm research. Unlike traditional scientific computing,

DOI: 10.54254/2755-2721/2025.22841

which is limited to a small number of typical algorithms, parallelization methods for numerous data-related non-numeric algorithms have become a key research focus. For instance, search engines that handle massive internet data rely on parallel technologies to process large-scale data-intensive applications, leading to the exploration of new parallel algorithms to enhance data processing efficiency.

2.3. Parallel programming

The advent of multi-core processors has reduced the cost of shared-memory parallel processing platforms, expanding the scope of parallel computing users from traditional fields to desktop and mobile computing domains. Currently, the price of a four-core system is only 1/5 to 1/10 of the price of a traditional 4-CPU system, which has driven the wider application of parallel programming.

The two main parallel programming models currently in use are message passing (e.g., MPI) and shared memory (e.g., OpenMP). Some applications also use multi-threaded APIs provided by operating systems or programming languages for shared-memory parallel programming. However, in desktop and mobile computing, many existing applications are serial, and developers are unfamiliar with parallel programming. This requires adapting existing applications to parallelism and developing new parallel applications.

As parallel programming becomes more widespread, there is an increasing demand for parallel programming languages, environments, and tools. Current parallel software is not yet capable of fully utilizing the performance of multi-core processors, and compilers, development languages, libraries, and tools often fall short in handling multi-core processors effectively. When programmers port existing applications to multi-core systems, they need to extract and explicitly state the parallelism, parameterize the application to adapt to varying cache sizes and hierarchical systems, and add code to achieve optimal NUMA (Non-Uniform Memory Access) effects [9]. Additionally, when developing high-performance computing systems, it is essential to consider both intra-node and inter-node parallelism, as well as parallelism within the chip.

3. Distributed processing technology

3.1. Industrial computer networks

The DCS (Distributed Control System), which emerged in the mid-1970s, is the prototype of industrial computer networks and has undergone three generations of development. The first generation achieved PCU (Process Control Unit) decentralization; the second generation adopted masterless communication and token passing; the third generation saw the standardization of communication protocols, with new technologies introduced to realize digitalization and automation, and the development of industrial networks was driven by Internet technology.

The entry of microprocessors into control systems has propelled the development of fieldbus systems. Traditional control methods had flaws, and fieldbus technology connects field devices to the bus, forming multiple product types and making the structure of industrial computer networks more complete.

Industrial computer networks differ from civilian networks in several ways. These networks require decentralized processing and centralized management, feature specialized networking methods, use diverse communication protocols with no unified standard, and emphasize optimization of resource allocation as well as real-time performance, reliability, and security. The aim is to achieve resource sharing and integrated industrial control management. A typical industrial computer network is shown below:

DOI: 10.54254/2755-2721/2025.22841

3.1.1. Integrated information and distributed control system I²DCS

The network topology of the Integrated Information and Distributed Control System (I²DCS) is shown in Figure 2.

System Scale: The I²DCS has a hierarchical structure, is flexible to build, provides a multi-task environment, and uses modular software. It has good openness and can be expanded according to needs, allowing users to develop their own algorithm programs.

Network Performance: The network is divided into I Net and Ethernet. I Net is used for real-time control, with high speed and long range, and uses token control. Ethernet is used for managing information communication, using TCP/IP and FTP protocols to realize data file communication and resource sharing [10].

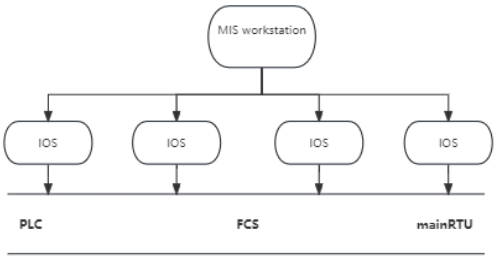

Figure 2: I²DCS network topology diagram

3.1.2. Rockwell automation industrial network

The network topology of Rockwell Automation is shown in Figure 3.

Ethernet: This is composed of a PLC-5 processor and an upper-level PC, with high transmission rates and long distances, using the TCP/IP protocol to realize the transmission of product management data and configuration files.

ControlNet™: It uses the CTDMA algorithm to transmit control information, has a flexible topology, high throughput, supports peer-to-peer communication and various types of information, reduces network traffic, and comes with dedicated configuration software.

DeviceNet™: This is an open network that supports CANBus technology, with multi-point branch connections. The transmission rate is selectable and is based on the producer/consumer model. It reduces the number of trunk-line taps and allows for many I/O point connections.

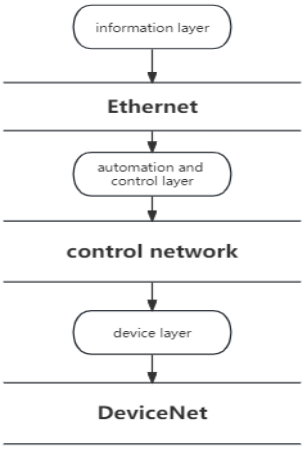

Figure 3: Rockwell automation industrial network topology diagram

DOI: 10.54254/2755-2721/2025.22841

3.2. Multi-agent systems

An Agent possesses autonomy, communication capabilities, reasoning and planning abilities, cooperation skills, and perception abilities, among others. A multi-agent system consists of multiple Agents that collaborate through interaction to solve problems, providing a unified model for complex systems and enjoying widespread application.

3.2.1. Cognitive models and theories of agents

This research mainly focuses on the formal description of Agent characteristics and the reasoning and decision-making processes. The BDI theory is an important foundation, and many scholars have conducted research from different perspectives to establish relevant theories and formal language systems.

The BDI theory, proposed by Bratman, includes core concepts such as Belief, Desire, and Intention. Belief refers to the Agent's perception of the state of the world, which is derived from its perceptions and experiences. Desire represents the goals or desired states that the Agent aims to achieve. Intention is the specific course of action or plan that the Agent chooses to pursue in order to fulfill its desires, reflecting a selection and confirmation of desires based on beliefs.

Rao and Georgeff, building on Bratman's work, presented a formal model for BDI Agents and introduced decision-theory concepts, making the BDI theory applicable to actual system development. By formalizing the Agent's mental state and decision-making process, this model helps developers create smarter and more autonomous Agents.

The BDI theory provides a clear logical framework for the behavior of Agents in multi-agent systems, making their actions more rational and predictable. In complex multi-agent interaction scenarios, Agents based on the BDI theory are better able to coordinate actions and achieve common goals.

3.2.2. Agent architectures

There are three types of Agent architectures: deliberative, reactive, and hybrid [11]. Deliberative architectures are based on symbolic reasoning, reactive architectures respond directly to stimuli, and hybrid architectures combine the advantages of both, making them commonly used in applications.

The deliberative architecture, as shown in Figure 4, is based on Simon and Newell's physical symbol system hypothesis. It maintains an internal representation of the world and modifies its state through symbolic reasoning. Its cognitive components primarily consist of a planner and a world model. This architecture modularizes cognitive functions, separating perception, learning, planning, and action, which reduces system complexity from an engineering perspective. However, due to the need to maintain a complex world model and perform symbolic reasoning, the decision-making process may be time-consuming, resulting in poor real-time performance in dynamic environments.

DOI: 10.54254/2755-2721/2025.22841

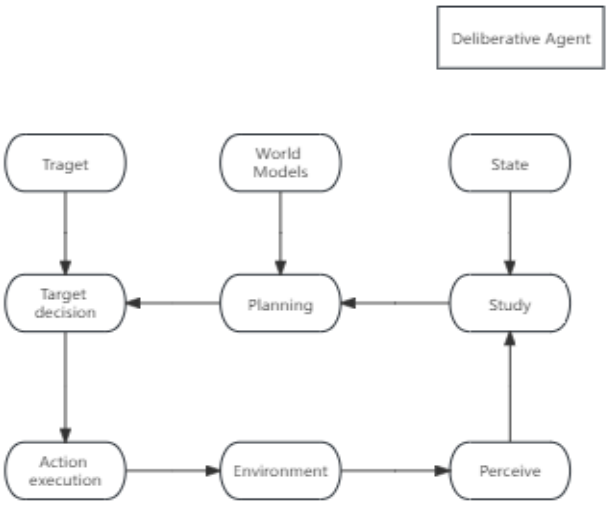

Figure 4: Deliberative agent architecture

The reactive architecture, as shown in Figure 5, originates from Brooks' ideas. Agents of this type do not rely on symbolic representations and directly generate actions based on sensory inputs. There is no world model represented symbolically, nor is complex symbolic reasoning performed. The perception module acquires environmental stimuli and directly passes them to the action execution module, which produces corresponding actions. The design assumption of this architecture is that the complexity of Agent behavior arises from environmental complexity. Its advantage is rapid response to environmental changes and the ability to quickly react to external stimuli. However, it lacks overall environmental perception and planning capabilities, making it difficult to complete complex tasks.

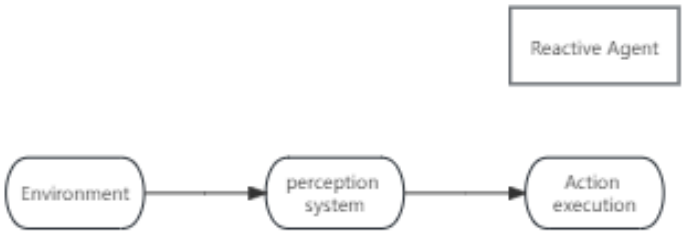

Figure 5: Reactive agent architecture

The hybrid architecture, as shown in Figure 6, combines the features of both deliberative and reactive architectures. It is the most commonly used in multi-agent system (MAS) applications and addresses the disadvantages of poor real-time performance in deliberative systems and the lack of planning capability in reactive systems. However, its design and implementation are more complex, requiring a reasonable division of layers and coordination of work between different layers.

DOI: 10.54254/2755-2721/2025.22841

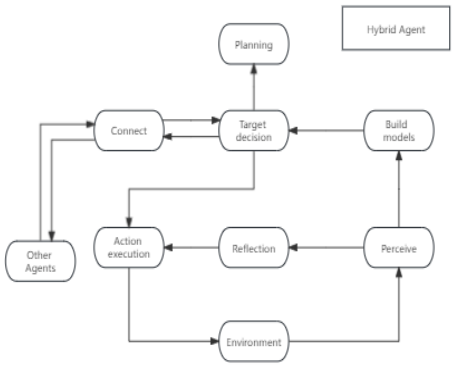

Figure 6: Hybrid agent architecture

3.2.3. Planning in MAS

Distributed multi-agent planning provides each agent with a model regarding the plans of other agents, enabling agents to consider the actions of others during the collaboration process and thus better coordinate to accomplish tasks. However, this requires that the participating agents do not have conflicting recognition and interests, as it would be difficult to reach a consistent global solution or plan.

In centralized multi-agent planning, a central entity aggregates the partial plans of other agents for analysis, removing inconsistencies and conflicts to merge them into new plans. The advantage of this approach is that it allows for overall coordination of the plans of all agents from a global perspective, theoretically leading to an optimized global plan. However, it demands high communication and computational resources because it needs to collect and process extensive planning information from agents. Furthermore, centralized planning requires precise models of tasks; otherwise, it is difficult to accurately analyze and coordinate the plans of different agents.

3.2.4. CSCW technology

CSCW (Computer-Supported Cooperative Work) technology originated in the mid-1980s, proposed by Irene Grief of MIT and Paul Cashman of Digital Equipment Corporation. MAS closely integrates with CSCW technology, where multiple agents interact and negotiate to complete complex tasks. In computer-supported collaborative design among different teams, each designer is encapsulated as an agent, and the maintenance of consistency in understanding and data across agents is achieved through intermediaries. CSCW technology serves as a supporting platform, providing a collaborative environment for the technological units.

3.2.5. CORBA technology

CORBA (Common Object Request Broker Architecture) technology, initiated and formulated by the Object Management Group (OMG), is a distributed object computing standard designed to achieve the sharing of information and resources across heterogeneous environments and to improve software reusability and system integration. The integration of agent technology with CORBA defines public facilities for agents, categorizing them into static agents and mobile agents. Mobile agents are akin to “smart messages” that can move across the network, carrying task code and data to execute on different nodes. Static agents interact via mobile agents, capable of completing certain tasks independently or encapsulating existing applications to provide new functionalities. In distributed systems, static agents can interact with other objects using CORBA’s object request broker

DOI: 10.54254/2755-2721/2025.22841

mechanism, while mobile agents can use CORBA's network communication capabilities to move between nodes and execute tasks, thereby enhancing system flexibility and scalability.

3.3. Cloud-edge-terminal architecture

Cloud computing, based on the internet, is a new computing model developed from distributed, parallel, and grid computing. Cloud computing enhances server-side parallel computing demands and drives research in fields like information storage, mining, and security.

To address the limitations of cloud computing, the industry has proposed a cloud-edge-terminal architecture. Terminal devices have limited computing capacity but offer strong privacy protection and are network-agnostic; the cloud is suited for resource-intensive tasks, and the edge node lies between the two. Optimizing computation and rationally offloading tasks is a key research focus.

(1) Edge Intelligent Computing Optimization in Cloud-Edge-Terminal Architecture: Cloud-based training and edge-based inference: To enable cloud-trained models to perform inference on edge devices, the industry has proposed various frameworks and lightweight techniques such as quantization, pruning, early exit, and lightweight models. These can reduce inference computation but introduce privacy risks in training.

Cloud-edge collaborative training: Federated learning algorithms can avoid privacy leakage. Three-layer federated learning balances communication and model accuracy through two aggregation stages. Two-layer federated learning optimizes resource use to improve efficiency in resource-limited environments.

Separate cloud-edge training: Small-sample learning methods like transfer learning and meta-learning can solve the problem of limited sample sizes and uneven data distribution on the edge side. These methods can also be combined with federated learning to enhance training efficiency and protect privacy.

(2) Offloading Solutions in Cloud-Edge-Terminal Architecture: Offloading solutions for task delay optimization: Task delay includes network transmission, waiting, and computation delays. Existing centralized scheduling, theoretically optimal strategies, and local scheduling algorithms based on edge servers all have their pros and cons.

Comprehensive delay and system energy consumption optimization for offloading: Mobile terminals have limited battery life. A research focus is on achieving minimal system energy consumption under delay constraints. Related studies consider multi-task scenarios, time-varying channels, and device mobility, but challenges remain, such as high computational complexity and not accounting for multi-channel fluctuations.

(3) Challenges and Research Trends in Edge Intelligent Computing under Cloud-Edge-Terminal Architecture: Deep integration of cloud-edge-terminal architecture with specific computing tasks: Different computing tasks have distinct characteristics. Combining these tasks with the architecture can optimize resource consumption. For instance, in a forest fire monitoring system, tasks can be allocated reasonably based on business and device characteristics.

Solutions for terminal device disparities: Terminal devices differ in computing power, data structure, data distribution, and communication conditions, affecting algorithm deployment, data processing, and offloading strategies. Solutions need to be found.

Incentives and trusted computing in multi-entity collaborative edge intelligent computing: Multi-entity collaboration in edge intelligent computing requires an incentive mechanism. Currently, blockchain technology has been introduced but still faces resource occupation issues, and it must guard against malicious participants to ensure computing security and trustworthiness.

Intelligent offloading in edge computing under communication constraints: Offloading is related to the communication environment. Current research strategies are fixed, and there is a need to enhance node self-organization for intelligent offloading to improve system robustness [12].

DOI: 10.54254/2755-2721/2025.22841

4. Summary and outlook

This paper, in conjunction with the development of intelligent computing platform technologies, primarily elaborates on the basic principles of multi-core processor technology, large-scale parallel algorithms, parallel programming techniques, industrial computer networks, multi-agent systems, and cloud-edge architecture from the perspectives of parallel processing and distributed processing technologies. In recent years, with the deployment of various intelligent applications, multi-core processor technology has expanded from the traditional field of high-performance computing to desktop and mobile computing, greatly enhancing computing efficiency. The research on large-scale parallel algorithms has provided new solutions for processing massive data, driving the development of artificial intelligence and big data technologies. The continuous evolution of parallel programming models has enabled developers to better leverage the performance of multi-core processors, further promoting the popularization of parallel computing. The development of industrial computer networks has made industrial control systems more intelligent and automated, while research on multi-agent systems has provided new approaches for modeling and solving complex systems. The emergence of cloud-edge architecture offers new possibilities for the integration of cloud computing and edge computing. The combination of these technologies not only enhances the computing capacity of systems but also increases their flexibility and scalability.

Currently, intelligent computing platform technologies still face numerous challenges. Future research will primarily focus on the following areas: the optimization and expansion of multi-core processors, more efficient management of shared resources, optimization of system energy efficiency, and solving the complexity issues related to algorithms and programming; innovations in parallel algorithms to improve data processing efficiency and accuracy; the intelligence and security of distributed systems; the deep integration of cloud-edge architecture with specific computing tasks; and collaboration and planning issues in multi-agent systems.

References

[1] *Wei, S. X. (1995). An overview of the development of parallel processing technology. Aviation Manufacturing Engineering, 6, 1-18.*
[2] *Wang, L. (2012). An overview of parallel computing technology. Information Technology, 10, 1-5.*
[3] *Yang, H., Wang, X. D., He, P., Liu, Z., Yao, H. J., & Guo, Y. M. (2025). DAG schedulability analysis and optimization of multi-core processors for avionic system computing platforms. Small and Micro Computer Systems, 3, 1-10.*
[4] *Mao, Y. M., & Liu, Y. X. (2025). Parallel DCNN optimization algorithm based on ABWO. Computer Engineering and Design, 2, 1-7.*
[5] *Huang, B. (2013). Reconfigurable computer design based on multi-core processors. Computer Measurement and Control, 1, 1-3.*
[6] *Bi, Y. J. (2024). Design and application of a neural network-based arc furnace control system in industrial computer networks. Software Development and Applications, 8, 1-3.*
[7] *Liu, Z. Q. (2025). Design and application of a manufacturing enterprise MES system based on multi-agent systems. Information Systems Engineering, 2, 1-4.*
[8] *Hong, Y. H. (2025). Design of a distributed data mining system for enterprise management based on cloud computing. Journal of Jiamusi University, 1, 1-3.*
[9] *Chen, G. L., Sun, G. Z., Xu, Y., & Long, B. (2009). Integrated research status and development trends of parallel computing. Science Bulletin, 8, 1-7.*
[10] *Zhang, H. (2000). Industrial computer networks. Journal of Xi'an Jiaotong University, 12, 1-4.*
[11] *Li, H. G., & Wu, Q. D. (2003). An overview of multi-agent systems. Journal of Tongji University, 6, 1-5.*
[12] *Dong, Y. M., Zhang, J., Xie, C. Z., & Li, Z. Y. (2024). Key issues of edge intelligent computing under the cloud-edge-end architecture: Computation optimization and task offloading. Journal of Electronics and Information, 3, 1-12.*