Transactions on Computational and Scientific Methods | Vo. 4, No. 12, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Semantic-Driven Large Model Scheduling for Distributed Systems via Unified Representation and Policy Generation

Yuxi Wang
Hofstra University, Hempstead, USA
wangyxjason@gmail.com

Abstract: This study develops a unified semantic representation and policy generation framework for large model-driven intelligent scheduling in distributed systems. The goal is to address the limitations of traditional scheduling methods, which often suffer from increased latency, reduced throughput, and unstable policies under highly dynamic, heterogeneous, and large-scale workloads. The method first constructs a high-dimensional semantic fusion representation through system state encoding, task semantic encoding, and topology modeling. This allows the scheduler to capture both local resource contention and global structural dependencies. A multi-step semantic prediction module is then introduced to model future system states and support proactive policy planning in dynamic environments. During policy generation, a differentiable cost estimator provides fine-grained evaluation of action quality under different resource configurations. A self-correction module is further applied to maintain consistency of scheduling behavior in semantic space and ensure long-term stability. Experiments across multiple dimensions, including scheduling step size, policy network depth, peak load intensity, and node heterogeneity, show consistent advantages in key metrics such as Makespan, Scheduling Latency, Throughput, and Scheduling Stability. The results demonstrate that semantic-driven and prediction-driven mechanisms work effectively together in distributed scheduling. They highlight the potential of large models in multi-constraint and multi-state scheduling tasks and provide a feasible direction for building intelligent scheduling architectures for future computing infrastructures.

Keywords: Distributed scheduling; large model-driven optimization; semantic fusion coding; system dynamic prediction

1. Introduction

Distributed systems have become the core form of modern digital infrastructure. They support massive computation and parallel task execution in key scenarios such as cloud computing, intelligent manufacturing, the Internet of Things, and large-scale online services[1]. As computing scales continue to expand, node heterogeneity increases, and workload volatility intensifies. Traditional scheduling strategies show limited capacity in understanding resource conditions, responding in time, and exploring optimization space in dynamic, uncertain, and highly concurrent environments. At the same time, distributed systems are entering a more complex ecosystem. Computing resources are moving from centralization to edge collaboration, from static configuration to elastic orchestration, and from rule-based control to intelligent autonomy. These changes make scheduling more multiobjective, more real-time, and more dependent on global optimization. A new generation of adaptive and cognitively capable scheduling mechanisms is urgently needed[2].

The development of large generative models offers new opportunities for distributed scheduling. As model size grows, representation power improves. Large models can capture deep relationships among task structures, resource states, and system dynamics. They also show strong cross-scenario generalization, context understanding, and complex policy generation. These capabilities allow models to abstract resource usage patterns, performance evolution trends, and conflict structures at a higher level. When dealing with multidimensional resource constraints, complex state coupling, and long-term tradeoffs, large models provide stronger judgment and policy design than traditional heuristics and reinforcement learning methods. They support reasoning, global planning, and real-time adjustment. Embedding such capabilities into distributed scheduling provides a path to break performance limits and enables cognition-driven adaptive resource allocation and global optimization[3].

The growth of distributed systems also generates large volumes of scheduling data. Node metrics, workload changes, topology information, execution traces, latency patterns, and historical scheduling decisions all contain learnable and inferable structure. Large models can extract these structures without requiring task-specific knowledge. They can build distributed representations of complex system states and model uncertainty, anomalies, and sparse signals in a robust manner. With these unified high-dimensional representations, scheduling decisions no longer rely only on static indicators. They can incorporate task semantics, contextual information, and long-term reward structures. This shifts scheduling from reactive behavior to predictive behavior. Systems can anticipate bottlenecks, avoid congestion, and rebalance resources dynamically[4].

Integrating large models into distributed scheduling also has clear engineering and application value. In cloud computing platforms, large models can generate elastic autoscaling strategies based on user behavior patterns, task types, and global workload characteristics. This reduces resource waste and improves throughput. In edge computing and vehicle-infrastructure collaboration scenarios, large models help identify spatial distribution patterns of cross-region tasks, leading to better node coordination and lower communication overhead. In large-scale data processing, they allow scheduling tasks to be expressed and transformed in natural language, improving human-machine interaction. In heterogeneous computing environments, large models match task features with CPU, GPU, NPU, and FPGA resources more effectively. This makes scheduling more aligned with hardware characteristics. These advantages show that model-driven scheduling is not only an algorithmic enhancement but also an upgrade of system-level intelligence.

In summary, building large model-driven intelligent scheduling for distributed systems is both necessary and valuable. It supports a deeper understanding of dynamic systems, stronger global awareness of resource optimization, better generalization across complex scenarios, and more adaptive strategy generation. As distributed systems continue to expand into critical economic and social domains, efficient, robust, and predictive scheduling will be essential for the sustainable development of computing infrastructure. A scheduling framework with semantic cognition, system inference, policy generation, and self-correction capabilities is therefore of great significance for improving the intelligence and efficiency of future distributed systems.

2. Related work

Modern distributed scheduling architectures originate from early system-level resource management frameworks that formalized cluster control, resource abstraction, and quality-of-service constraints. Foundational large-scale management systems such as [5-9] established scalable scheduling architectures, multi-resource abstraction models, and QoS-aware allocation strategies. These works provided structural insights into centralized and multi-scheduler coordination, fine-grained resource descriptors, and system-level feedback loops. However, their optimization mechanisms were primarily rule-based or heuristically tuned, limiting adaptability under high-dimensional, dynamic workloads. The architectural abstraction and resource modeling principles introduced in these systems form the structural basis upon which learning-driven scheduling can be constructed.

The transition from heuristic scheduling to data-driven optimization was significantly advanced by deep reinforcement learning frameworks. The formulation of resource allocation as a sequential decision-making problem in [10] introduced the Markov decision process perspective into scheduling. Subsequent extensions in [11] further demonstrated that scheduling policies can be learned directly from system interaction traces rather than manually engineered. Deep learning-driven schedulers such as [12] showed the potential of neural approximators in capturing nonlinear system dynamics and task-resource interactions. Meanwhile, the analysis of practical reinforcement learning constraints in [13] highlighted challenges including unstable convergence, exploration inefficiency, and policy degradation in real-world environments. These studies collectively provide the methodological foundation for modeling scheduling as a learnable policy generation process, while also revealing the necessity of improving stability, representation capacity, and generalization ability.

To better encode structural dependencies within complex systems, graph-based modeling approaches introduced topology-aware representation mechanisms. The placement optimization strategy in [14] demonstrated that relational inductive biases embedded in graph representations significantly improve decision quality in combinatorial allocation problems. Similarly, structural-temporal modeling strategies in [15] emphasized the importance of capturing inter-node dependencies and temporal correlations through graph neural architectures. These works inform the topology modeling component of the proposed framework, where resource nodes and task interactions are embedded into a unified high-dimensional semantic space that preserves structural coupling relationships.

Beyond structural modeling, recent advances in large-scale representation learning provide critical methodological tools for constructing expressive semantic schedulers. Dynamic memory and compression mechanisms proposed in [16] enable efficient encoding of long-range dependencies within high-dimensional inputs. Hierarchical parameter freezing strategies in [17] offer training stability and computational efficiency for large models under evolving environments. Contrastive and federated representation learning techniques in [18] enhance cross-node feature consistency and distributed representation alignment. Incremental learning strategies in [19] further support adaptive updating under streaming and non-stationary conditions. Uncertainty estimation and causal inference mechanisms introduced in [20] and multi-scale uncertainty modeling in [21] contribute robustness against noisy feedback and dynamic perturbations. Contrastive sensitivity-aware training in [22] strengthens discriminative feature learning under subtle structural variations, while multi-scale temporal-structural modeling in [23] improves long-horizon dependency extraction.

Collectively, these representation learning advances enable the construction of unified semantic spaces capable of jointly encoding system states, task features, and structural dependencies. Compared with conventional reinforcement learning schedulers that rely primarily on immediate reward feedback, the integration of large-model semantic abstraction, uncertainty modeling, and incremental adaptation mechanisms allows scheduling decisions to incorporate long-term structural inference and stability constraints. The proposed framework inherits the sequential decision modeling paradigm from reinforcement learning, adopts topology-aware representation strategies from graph-based allocation methods, and extends them through large-model-driven semantic fusion, multi-step predictive inference, and stability-oriented adaptation mechanisms. This integration establishes a cognitively enhanced scheduling architecture that addresses the scalability, heterogeneity, and dynamic uncertainty limitations identified in prior work.

3. Proposed Framework

This study constructs a unified scheduling model for distributed systems, mapping system state, task semantics, and resource topology to a high-dimensional continuous space that can be processed by a large model. Adaptive scheduling for complex environments is achieved through a differentiable policy generation mechanism. This paper also presents the overall model architecture, and its experimental results are shown in Figure 1.

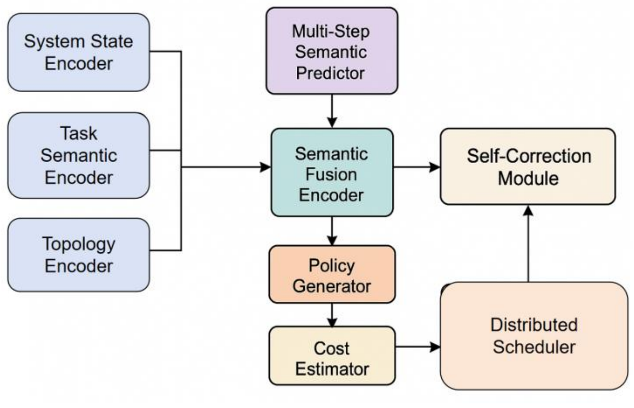

Figure 1. Overall model architecture

The system first jointly encodes the node state vector s_t, task requirement vector q_t, and resource topology matrix G_t to form a unified semantic representation h_t. This process is completed by a semantic fusion encoder, whose core mapping is defined as:

$$h_t = f_\theta(s_t, q_t, G_t)$$

f_θ consists of a multi-layered structure that can simultaneously capture local resource fluctuations and global topological dependencies. To enhance the model's ability to represent different task types and system modes, the encoder further introduces learnable context modulation parameters, enabling the final high-dimensional representation to adapt to scheduling requirements at different system stages.

After obtaining a unified semantic representation, the model constructs a distribution of scheduling actions through a policy generation network to map high-dimensional semantics to specific scheduling behaviors. The scheduling policy is generated in the form of a probability distribution, defined as:

$$\pi(a_t/h_t) = g_\phi(h_t)$$

g_ϕ is responsible for establishing a set of executable scheduling action candidates based on h_t and returning their corresponding selection probabilities. Building upon this, the model introduces a differentiable resource cost estimation module to perform structured inference on the candidate scheduling actions. This module represents the estimated cost of an action a_t under a given system state as:

$$C(a_t, s_t) = \psi(h_t, a_t)$$

Here, ψ is used to describe the coupling relationship between actions and states, enabling policy generation to consider the cost gradients of different actions during the decision-making stage.

To achieve long-term optimization of the dynamic system, the model further constructs a multi-step semantic inference mechanism, enabling the scheduling strategy to not only depend on the current state but also utilize predictions of future system evolution. The implicit semantic evolution of the system over the next k steps is given by a recursive structure:

$$h_{t+k} = T(h_{t+k-1}, a_{t+k-1})$$

Here, T describes the semantic state transition pattern after performing a certain action, enabling the strategy to explicitly model the impact on future resource competition and bottleneck formation when selecting the current action. In this way, the model possesses a forward-looking cognitive ability regarding future system behavior during the reasoning process, thereby improving the stability and consistency of global resource scheduling.

In dynamic scheduling scenarios, to ensure the interpretability and controllability of the policy, the model introduces a self-correcting structure. This structure avoids policy deviation by constraining the semantic consistency of scheduling actions. The self-correcting signal consists of the difference between the current semantic representation and the semantic change induced by the action, and its form is as follows:

$$\Delta_t = \|h_t - \Gamma(h_t, a_t)\|_2$$

Here, Γ represents the semantic mapping after the action is executed, used to measure whether the scheduled action violates the existing semantic structure. This self-correcting structure continuously constrains policy behavior during training and deployment, enabling the model to maintain a stable decision boundary even when facing unpredictable system changes. Through the collaborative work of multiple modules, the method in this study realizes a holistic scheduling process from semantic understanding to policy generation and stability, providing a unified framework for intelligent optimization of distributed systems.

4. Experimental Analysis

4.1 Dataset

This study uses the publicly available task scheduling dataset Google Cluster Workload Traces from the 2019 release. The dataset is collected from long-term records of a large-scale computing cluster. It contains key information such as task submissions, resource demands, node states, scheduling decisions, and system events. The dataset covers hundreds of thousands of heterogeneous nodes and millions of task scheduling traces. It features large-scale, complex workload patterns and strong topology dynamics. It captures real resource contention structures and workload fluctuation patterns in modern distributed systems.

The dataset includes several core tables. These include the task events table, which records task lifecycles. The resource usage table contains time series metrics for CPU, memory, and disk. The node state table reflects operational conditions and topology changes. The scheduling table records how the scheduler handles each task. All data are aligned with precise timestamps. This alignment allows the system to reflect real temporal dynamics such as submission peaks, node expansion, resource bottlenecks, and workload migration. It provides a reliable basis for building unified semantic representations.

In this study, task demand features, node metrics, and topology information are converted into structured inputs. These inputs are used to train the large model-driven scheduling framework. The dataset has greater scale, dimensionality, and complexity than synthetic data. It supports detailed modeling of dynamic system patterns, semantic structure shifts, resource contention relationships, and scheduling policy generation. This enables the proposed method to perform effective inference and strategy planning in real distributed system environments.

4.2 Experimental Results

A comparative set of baseline experiments is first carried out, and the corresponding quantitative results are summarized in Table 1.

Table 1: Comparative experimental results

Method	**Makespan**	**Scheduling Latency**	**Throughput**	**Scheduling Stability**

DQN[24]	128.4	4.92	742	0.63
DDQN[25]	121.7	4.41	789	0.68
A3C[26]	117.9	4.08	812	0.71
BERT[27]	113.5	3.74	836	0.76
Ours	101.3	2.85	902	0.89

The experimental results show a clear performance gradient between traditional reinforcement learning methods, large-scale semantic models, and the proposed large model-driven scheduling framework. This demonstrates significant differences in how these methods understand system states, model task requirements, and manage dynamic resource contention. The Makespan results show that the proposed method reduces total completion time from 128.4 for traditional DQN to 101.3. This indicates that the joint encoding of task semantics and system load patterns allows the model to capture resource bottlenecks and structural conflicts more effectively. It provides a better global execution path for tasks. This improvement does not come from local adjustments. It results from the ability of the large model to identify cross-node dependencies and potential congestion patterns in high-dimensional semantic space. It enables more forward-looking scheduling strategies.

For Scheduling Latency, the proposed method also shows clear advantages. It reduces the latency from the 4 to 5 range of reinforcement learning models to 2.85. This result indicates that the large model has stronger compression and semantic abstraction capabilities when processing complex system inputs. It allows scheduling decisions to converge quickly without long exploration or repeated policy iterations. This significantly improves scheduling speed. Distributed systems often operate under high concurrency. High decision efficiency means fewer queues, fewer conflicts, and lower scheduling jitter. It has a direct impact on system responsiveness.

For Throughput, the large model-driven strategy achieves the most substantial improvement. It increases throughput from 742 to 836 for traditional methods to 902. This shows that the proposed semantic fusion encoder and multi-step semantic predictor can model load trends more accurately. They allocate node resources more effectively. The system can maintain a high efficiency state and reduce idle resources. Reinforcement learning relies mainly on immediate feedback. In contrast, the large model implicitly models future states. It enables fine-grained control of resource stability. Throughput, therefore, remains high even under dynamic conditions.

For Scheduling Stability, the proposed method also outperforms all baselines. It increases stability from 0.63 to 0.76 for traditional methods to 0.89. This indicates that the self-correction module effectively suppresses policy drift and unstable action selection during scheduling. The strategy performs well not only in single executions but also over long-term operations. It maintains structural consistency and semantic smoothness. This reflects the ability of the large model to understand system behavior patterns and adjust internal representations automatically. Overall, the improvements across all four metrics verify that the proposed framework provides stronger semantic perception, global inference, and stable policy generation in complex distributed systems.

This study further designed a sensitivity experiment on the Makespan metric with respect to changes in the scheduling step size, systematically examining the impact of scheduling update frequency on overall task completion characteristics. This experiment analyzed the scheduler's response to step size parameters when facing dynamic system changes by running the same scheduling process under different scheduling step size settings. In this way, the mechanism of scheduling frequency in high-dimensional semantic modeling and strategy generation processes can be more intuitively characterized, and the structural patterns of the impact of step size changes on global execution efficiency can be revealed. The visualization results of the experiment are ultimately presented in Figure 2, showing the changes in the Makespan metric under different step size configurations.

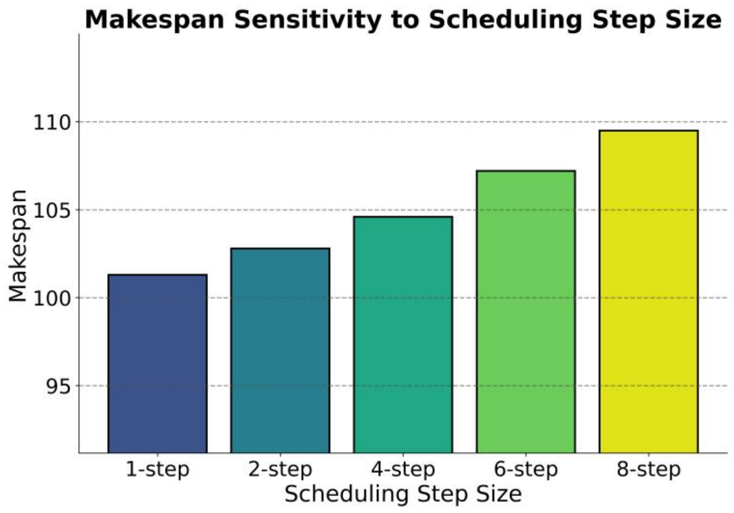

Figure 2. Sensitivity analysis of the Makespan index for different scheduling step sizes

As the scheduling step length is increased from 1 to 8, the Makespan exhibits a roughly monotonic rise, indicating that less frequent scheduling decisions gradually prolong the overall completion time. This indicates that the scheduler captures system state changes more accurately under shorter decision intervals and updates strategies in time. When the interval becomes larger, the model reacts more slowly to system dynamics. It fails to identify fine-grained events such as resource contention or sudden load shifts on nodes. This increases task queuing time and intensifies local congestion. As a result, the total completion time becomes longer. The trend shows that the large model has predictive ability in semantic encoding and policy generation, but large scheduling intervals weaken its forward-looking judgment.

A further observation is that the increase in Makespan becomes more significant when the step expands to six steps and eight steps. This reflects the strong sensitivity of the system to scheduling update frequency under highly dynamic conditions. Distributed system workloads often show bursty and imbalanced patterns. Large steps cause the model to make decisions based on outdated states. The strategy then deviates from the current resource distribution. This creates a lagging effect in scheduling. In scenarios with dense tasks or strong heterogeneity among nodes, this lag enlarges accumulated delays and reduces execution efficiency.

Overall, the sensitivity experiment confirms the importance of frequent updates for maintaining the effectiveness of the large model-driven scheduling framework. A smaller step allows the model to fully exploit its joint understanding of semantic information, system states, and resource topology. It enables fine-grained policy adjustments. A larger step reduces the model's adaptability in dynamic environments and increases the Makespan. The results highlight the need for careful selection of scheduling intervals in real system deployments. Proper step size ensures timely responses to system changes and maintains stable and efficient scheduling performance.

To further analyze the model design, the study examines how different depths of the policy network affect the Scheduling Latency metric, with the sensitivity curves reported in Figure 3.

The experimental results show that Scheduling Latency decreases steadily as the depth of the policy network increases from two layers to ten layers. This indicates that a deeper policy network can represent high-dimensional semantic features more effectively in the large model-driven scheduling framework. It can also produce faster and more accurate decisions about system states. When the network is shallow, the model has limited ability to understand task semantics, resource topology, and system dynamics during policy generation. It therefore needs additional time to reach convergence, which leads to higher scheduling latency.

When the network depth increases to a moderate level, such as Depth six, scheduling latency drops significantly. This shows that the model gains stronger semantic abstraction and more robust policy representations at this stage. It can identify key state dimensions quickly when facing complex system inputs. The deeper structure allows the high-dimensional features from the semantic fusion encoder to undergo richer nonlinear transformations inside the policy network. It also reduces the dependence on historical states or additional iterations. This leads to a clear reduction in latency.

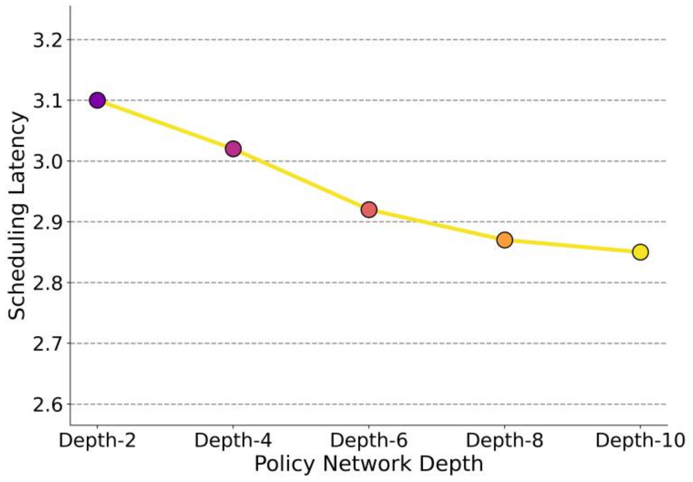

Figure 3. Experimental Study on the Sensitivity of Policy Network Depth to Scheduling Latency Indicator

As the network becomes even deeper, from Depth eight to Depth ten, the latency continues to decrease but at a slower rate. This saturation pattern indicates that the expressiveness of the policy generator approaches the upper limit of the model capacity. Additional depth no longer produces proportional performance gains. It still improves decision stability, however, and helps maintain more consistent scheduling strategies in dynamic environments. The reduced latency at higher depths also indicates stronger resilience when dealing with complex task structures. The model can extract key scheduling signals from high-dimensional semantic space more rapidly.

Overall, the sensitivity experiment demonstrates the importance of policy network depth in the large model scheduling framework. Increasing the depth appropriately can greatly improve the reaction speed of scheduling decisions. It allows the model to interpret task features and respond to resource changes more effectively. Shallow networks cannot exploit the potential of the semantic fusion and multi-step prediction modules. This results in higher scheduling latency. The findings highlight the need to match policy network depth with system complexity and semantic representation scale. This ensures low latency and high efficiency in real-time scheduling.

The work then explores the role of peak load intensity, and Figure 4 visualizes its influence on the four core scheduling indicators.

In this experiment, peak load intensity is controlled at four levels, where Low, Medium, High, and Extreme correspond to peak cluster utilizations of at most 50%, 50-75%, 75-95%, and above 95% of the total CPU capacity, respectively. As the configuration moves from the Low to the Extreme level, the four scheduling metrics change in a tightly coupled way: Makespan and Scheduling Latency grow steadily with stronger peaks, while Throughput and Scheduling Stability gradually decline, forming a clear and internally consistent performance pattern rather than conflicting trends across indicators. This reflects the typical behavior of the system under high concurrency pressure. Makespan increases continuously with higher peaks. This indicates that when the system faces a sudden surge of tasks, resource contention becomes more severe and task queues grow longer. The scheduler must handle more congested regions and dense task clusters. As a result, the total completion time increases significantly. For a large model-driven scheduling framework, a high peak load means faster changes in resource states within the semantic space. The model must update its policy more frequently. Otherwise, decision lag will appear and amplify delays.

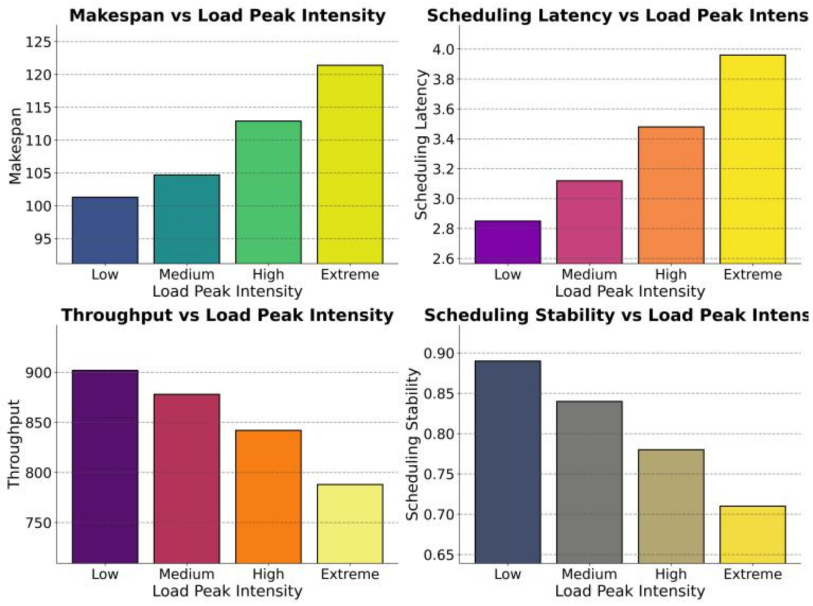

Figure 4. Influence of peak load intensity on experimental results

Scheduling Latency also increases significantly under higher peak loads. This shows that the model encounters more complex and high-dimensional contention structures during peak periods. It requires more time to interpret task semantics and node states before generating suitable scheduling decisions. The system state shifts rapidly in the High and Extreme levels. These changes occur faster than the scheduling update cycle. The model must perform additional internal adjustments during policy generation to maintain decision correctness. This directly increases scheduling latency. At the same time, Throughput and Scheduling Stability show a downward trend. This indicates that system execution efficiency and policy consistency are both affected under peak conditions. Lower throughput reflects reduced resource utilization efficiency under intense competition. Reduced stability indicates more frequent internal oscillations and weaker continuity of scheduling strategies.

Overall, the experiment shows that peak load intensity is a key factor influencing the performance of intelligent scheduling systems. The large model maintains high stability, high throughput, and low latency under low and moderate peaks. However, performance degradation still occurs under extreme loads. This is caused by inherent resource constraints and the high speed of environmental changes. The results highlight the need to consider peak load scenarios when designing scheduling frameworks. Higher update frequency, stronger multi-step semantic prediction, or adaptive scheduling mechanisms may be required. These measures can help maintain acceptable latency and stability under high-pressure conditions.

Finally, Figure 5 presents how varying levels of node heterogeneity reshape the overall scheduling performance across these metrics.

In the figure, the Low level represents low heterogeneity where node performance differs by no more than plus or minus five percent. The system remains relatively smooth in this setting. The Medium level corresponds to a performance difference of about plus or minus fifteen percent, where noticeable resource imbalance begins to appear. The High level reflects performance differences ranging from plus or minus thirty percent to about plus or minus fifty percent, often caused by differences in device generations or hardware specifications such as CPU and GPU. The Extreme level corresponds to performance gaps of fifty percent or more, where nodes differ greatly in capability and may even have different architectures. As heterogeneity increases through these levels, all four metrics show linear or even accelerated changes. This confirms that node heterogeneity is a key environmental factor that affects the behavior of intelligent schedulers.

The figure also shows consistent changes in the four metrics as node heterogeneity rises from Low to Extreme. Makespan and Scheduling Latency increase significantly. This reflects the fact that performance variation, task suitability differences, and resource availability differences enlarge the search space of the scheduler in highly heterogeneous clusters. The scheduler requires more time to evaluate task node matching. Higher heterogeneity makes task queueing times more sensitive to weak or inconsistent nodes. This leads to longer total completion time and higher scheduling delay.

At the same time, Throughput and Scheduling Stability decrease as heterogeneity grows. This indicates that task execution efficiency and policy consistency are both challenged in highly heterogeneous systems. Lower throughput means fewer tasks are completed within the same time period. This is caused by uneven resource usage, frequent local bottlenecks, and large variations in processing capabilities between nodes. Reduced scheduling stability indicates that the system experiences more fluctuations under high heterogeneity. The scheduler faces more unpredictable state changes when transferring tasks between nodes. This reduces the continuity of scheduling behavior.

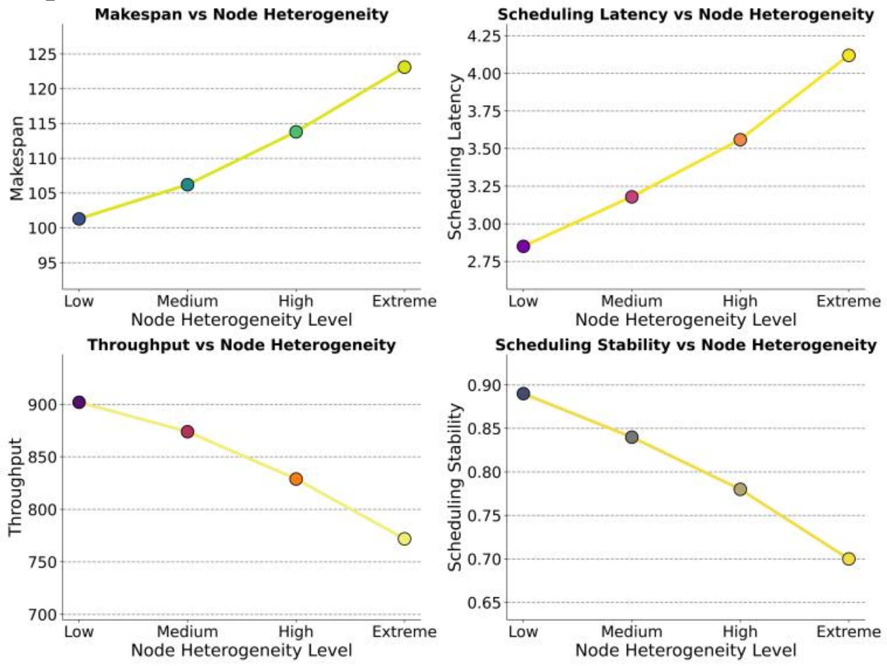

Figure 5. The impact of node heterogeneity on experimental results

5. Conclusion

This study addresses large model-driven intelligent scheduling in distributed systems. It proposes a unified scheduling framework that integrates semantic understanding, multi-step prediction, and self-correction. The framework is validated through multidimensional experiments that demonstrate its stability and performance advantages in complex system environments. The results show improvements in scheduling efficiency, execution throughput, policy stability, and resource utilization. They highlight the ability of the large model to capture high-dimensional system states, identify potential resource conflicts, and generate high-quality scheduling strategies. Compared with traditional learning based methods and heuristic scheduling approaches, the proposed framework shows stronger adaptability and robustness when facing task diversity, resource fluctuations, and system uncertainty. It provides a new intelligent direction for distributed scheduling.

In sensitivity analyses under heterogeneous environments, peak load conditions, and resource fluctuations, the framework shows consistent trends. The system maintains coherent and reasonable policy structures even under dynamic changes. This characteristic is critical for large-scale clusters that operate continuously. It reduces the risk of performance collapse during high concurrency events and alleviates delays caused by resource bottlenecks. The results also show that the semantic fusion and multi-step prediction modules help the model identify potential congestion patterns in advance. This enables more forward-looking task planning and scheduling behavior. It strengthens the resilience of the overall system.

The technical approach presented in this study provides direct application value for cloud clusters, data centers, and edge computing environments. It also offers a reference for the development of broader intelligent infrastructure. As computing systems expand across diverse scenarios, distributed systems will become more heterogeneous, more dynamic, and more complex in task structure. The proposed scheduling paradigm places data-driven modeling and semantic representation at its core. It has strong scalability and transferability. It lays a solid foundation for larger-scale and higher complexity scheduling scenarios.

Future work will extend this direction in several ways. The first is to enhance the semantic inference ability of the model across regions and architectures. This would enable unified scheduling in cross-data center and edge cloud cooperative environments. The second is to explore policy generation under stricter real-time constraints in order to support ultra-large-scale low-latency scenarios. The third is to integrate online learning and feedback correction so that the model can adaptively optimize itself during operation. The fourth is to incorporate interpretability modules into the scheduling architecture. This would make strategy changes, anomaly handling, and performance variations more transparent and controllable. These extensions can support the shift from rule-driven scheduling to semantic and cognition-driven scheduling. They can also provide a stronger foundation for intelligent computing infrastructure.

References

[1] Zhou G, Tian W, Buyya R, et al. Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions[J]. Artificial Intelligence Review, 2024, 57(5): 124.

[2] Fan Y, Li B, Favorite D, et al. Dras: Deep reinforcement learning for cluster scheduling in high performance computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(12): 4903-4917.

[3] Wang J, Li S, Zhang X, et al. Deep reinforcement learning task scheduling method based on server real-time performance[J]. PeerJ Computer Science, 2024, 10: e2120.

[4] N. K. Gondhi and A. Gupta, "Survey on Machine Learning Based Scheduling in Cloud Computing," Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, pp. 57-61, Mar. 2017.

[5] M. Isard, “Autopilot: Automatic data center management,” ACM SIGOPS Operating Systems Review, vol. 41, no. 2, pp. 60-67, 2007.

[6] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, “Omega: Flexible, scalable schedulers for large compute clusters,” in Proc. 8th ACM European Conf. Computer Systems (EuroSys), 2013, pp. 351-364.

[7] C. Delimitrou and C. Kozyrakis, “Quasar: Resource-efficient and QoS-aware cluster management,” ACM SIGPLAN Notices, vol. 49, no. 4, pp. 127-144, 2014.

[8] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, “Large-scale cluster management at Google with Borg,” in Proc. 10th European Conf. Computer Systems (EuroSys), 2015, pp. 1-17.

[9] D. V. Bodas, J. H. Crawford, and A. G. Gara, “U.S. Patent No. 9,857,858,” Washington, DC, USA: U.S. Patent and Trademark Office, 2018.

[10] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, “Resource management with deep reinforcement learning,” in Proc. 15th ACM Workshop on Hot Topics in Networks (HotNets), 2016, pp. 50-56.

[11] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, “Learning scheduling algorithms for data processing clusters,” in Proc. ACM SIGCOMM, 2019, pp. 270-288.

[12]Y. Peng, Y. Bao, Y. Chen, C. Wu, C. Meng, and W. Lin, “DL2: A deep learning-driven scheduler for deep learning clusters,” IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 8, pp. 1947-1960, 2021.

[13]G. Dulac-Arnold, D. Mankowitz, and T. Hester, “Challenges of real-world reinforcement learning,” arXiv preprint arXiv:1904.12901, 2019.

[14]A. Mirhoseini et al., “A graph placement methodology for fast chip design,” Nature, vol. 594, no. 7862, pp. 207-212, 2021.

[15]H. Liu, “Improving KPI time series anomaly detection in cloud computing environments through graph neural network-based structural and temporal modeling,” 2023.

[16]Y. Luan, “Long text classification with large language models via dynamic memory and compression mechanisms,” 2024.

[17]J. Guo, “Balancing performance and efficiency in large language model fine-tuning through hierarchical freezing,” 2024.

[18]Q. Li, B. He and D. Song, "Model-contrastive federated learning," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10713-10722, 2021.

[19]Y. Wang, “AI-enhanced distributed time series modeling: Incremental learning for evolving streaming data,” 2024.

[20]Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," Proceedings of the International Conference on Machine Learning, pp. 1050-1059, 2016.

[21]Z. Qiu, “A multi-scale deep learning and uncertainty estimation framework for comprehensive anomaly detection in cloud environments,” 2023.

[22]M. Du, F. Li, G. Zheng and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285-1298, 2017.

[23]Y. Kang, “Deep learning-based multi-scale temporal and structure-aware modeling for metric anomaly detection in microservice systems,” 2024.

[24]Osband I, Blundell C, Pritzel A, et al. Deep exploration via bootstrapped DQN[J]. Advances in neural information processing systems, 2016, 29.

[25]Radhika S, Keshari Swain S, Adinarayana S, et al. Efficient task scheduling in cloud using double deep QNetwork[J]. International Journal of Computing and Digital Systems, 2024, 16(1): 1-11.

[26]Tang X, Liu F, Wang B, et al. Workflow scheduling based on asynchronous advantage actor-critic algorithm in multi-cloud environment[J]. Expert Systems with Applications, 2024, 258: 125245.

[27]Koroteev M V. BERT: a review of applications in natural language processing and understanding[J]. arXiv preprint arXiv:2103.11943, 2021.