Optimizing AI-driven Geographic Simulation Task Scheduling through Intelligent Runtime Estimation for Distributed Heterogeneous Clusters

Wanhao Li[a,c,d], Min Chen[a,c,d], Fengyuan Zhang*[b], Peilong Ma[a,c,d], Zaiyang Ma[a,c,d], Yongning Wen[a,c,d], Songshan Yue[a,c,d], Guonian Lü[a,c,d]

[a] *School of Geography, Nanjing Normal University, Nanjing, China*

[b] *School of Environment, Nanjing Normal University, Nanjing, China*

[c] *Key Laboratory of Virtual Geographic Environment (Ministry of Education of PRC), Nanjing Normal University, Nanjing, China*

[d] *State Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing, China*

Abstract

Geographic simulation (geo-simulation), as an essential approach for characterizing geographic processes and conducting predictive analysis, plays a central role in solving geographical and environmental issues. With the rapid development and wide application of artificial intelligence (AI), AI-driven geo-analysis models demonstrate higher accuracy and broader applicability in geo-simulations. However, their increasing dependence on massive computing resources and diverse runtime environments introduces significant heterogeneity, making efficient task scheduling in distributed clusters a major challenge. The core difficulty lies in the lack of effective identification of task characteristics and the accurate evaluation and rational utilization of computing resources, which often results in insufficient resource utilization or load imbalance. To address this issue, this study proposes an AI-driven task scheduling optimization framework via large language models (LLMs) for geo-simulation based on intelligent runtime estimation. The framework integrates real-time monitoring of computing resources, a historical task knowledge base, and large model prediction capabilities to achieve adaptive and resource-aware scheduling in distributed heterogeneous environments. Experimental results show that the proposed method can select the most suitable computing nodes for AI-driven geo-simulation tasks, significantly reduce runtime, alleviate the imbalance in task execution time caused by low-performance computing nodes, and improve overall system throughput and task success rate. This research provides a novel approach for efficient scheduling of AI-driven geo-simulation tasks and demonstrates potential application value in complex task scenarios.

Keywords: AI-driven geographic simulation; distributed heterogeneous clusters; intelligent scheduling; large language models; resource-aware computing

1. Introduction

Geographic simulation (geo-simulation) is a scientific method that uses geographical models to describe the relationship between geographic phenomena and physical laws (Lin et al., 2012; Xu et al., 2024), which plays a critical role in predicting geographic phenomena and processes that can help people solve geo-analysis problems (Belete et al., 2017; Chen, Qian, et al., 2023). With the rapid development and widespread application of artificial intelligence (AI) methods such as Random Forests, Transformers, Convolutional Neural Networks (CNNs), and Graph Neural Networks (GNNs), AI models have gradually transformed the way of geographic reasoning and prediction, significantly improving accuracy (Kuglitsch et al., 2022; Wang et al., 2023; Bao et al., 2025; Shi et

al., 2025). Meanwhile, an increasing number of researchers applies AI-driven geo-analysis models for complex geographical simulation, such as cross-view geolocation, urban expansion simulation, and traffic flow prediction (Dai et al., 2022; Karimi et al., 2024; Zhang et al., 2024). Moreover, the emergence of large language models (LLMs) (e.g., ChatGPT and Deepseek) has greatly advanced developments across various fields and further accelerated the AI-driven progress of scientific research (Sun et al., 2023; Bi et al., 2024; Zhao et al., 2024). These advances show AI-driven geo-analysis models can contribute to the technological innovation in geography (Chen, Claramunt, et al., 2023; Lü et al., 2025).

With the wide application of AI methods in geo-simulation, the computational requirements of models have significant heterogeneity with computing resources (Hu et al., 2019; Hu et al., 2024). To date, these models often follow different development styles for various model types (Deng et al., 2019; Liu et al., 2025; Lu et al., 2025). Therefore, they show different resource requirements in the computing environment (Li et al., 2025). For instance, CNNs are widely used for object detection and recognition in remote sensing imagery, requiring substantial GPU and memory, whereas Random Forests and other classical machine learning methods primarily depend on CPUs and RAM (Buschjager et al., 2018; Derry et al., 2023). Beyond computing resources, differences also exist in I/O intensity, working set size, network bandwidth requirements (such as distributed training or data scheduling), software dependencies (such as specific libraries, drivers, CUDA/cuDNN versions), as well as fault tolerance and checkpointing mechanisms (Chaturvedi et al., 2021). Due to heterogeneity, the same model may show different runtime across different computing nodes, often leading to inefficient execution and unbalanced resource utilization within the cluster. Consequently, it becomes difficult to efficiently execute diverse AI-driven geo-simulation tasks, which are directly executable computational models and workflows for simulating complex geographic processes, on a unified platform.

To address this challenge, distributed and layered architectures provide a suitable solution for effective geo-simulation tasks. They can integrate large-scale computing nodes and coordinate heterogeneous resources, thereby enabling the parallelization and efficient execution of geo-simulation tasks (Duan et al., 2023; Ma et al., 2025; Qin et al., 2025). Representative modeling frameworks such as OpenMI (Open Modeling Interface), OMS (The Object Modeling System), CSDMS (Community Surface Dynamics Modeling System), and OpenGMS (The Open Geographic Modeling and Simulation) have been developed to establish generalized modeling systems, providing standardized interfaces and distributed execution environments, thereby facilitating the coupling of multi-source geographic models and enabling cross-platform execution (Westen et al., 2007; David et al., 2013; Peckham et al., 2013; Chen et al., 2020). For large-scale geo-simulation tasks, researchers have leveraged supercomputers and distributed clusters to enhance parallel computing performance (Kim et al., 2015; Georganos et al., 2021), while employing model environment partitioning and resource virtualization to alleviate compatibility issues in heterogeneous computing environments (Liu et al., 2016; Sun et al., 2019). Meanwhile, with the advancement of cloud computing and containerization technologies, several efforts have attempted to migrate geographic models onto cloud platforms (Tarboton et al., 2024). In geo-distributed cloud systems, performance disparities and competition for shared resources often lead to asymmetric computational capacities among physical nodes. To mitigate this issue, efficient computing frameworks have been introduced (Zaharia et al., 2010; Dafir et al., 2020), and further studies have optimized load balancing mechanisms by

incorporating the actual performance of cluster nodes and the characteristics of containerization, thereby improving the overall processing capacity of clusters (Wang et al., 2011; Li et al., 2021).

However, the existing methods mostly rely on static rules or manual configuration, and the exploration of innovative task scheduling optimization mechanisms remain critical for ensuring the efficient execution of complex geo-simulation tasks, especially in AI-driven geo-simulation tasks (Cao et al., 2024). It still has three challenges: at first, current frameworks lack the essential description and evaluation way for AI-driven geo-simulation tasks, which lead to unsuitable computing resource configurations for geo-simulation tasks. For example, some geo-analysis models require substantial computing resources such as network bandwidth and storage for simulation. Secondly, current computing resource evaluation methods can hardly monitor situation of cluster nodes, and often ignore multi-dimensional real-time indicators such as GPU utilization and network load. Thirdly, current frameworks lack the intelligent methods for AI-driven geo-simulation task execution. The frameworks, which ignore the estimation of potential task performance across computing nodes, could have ineffective decision-making in scheduling of high load and multi type tasks.

To address these challenges, this study proposes an intelligent scheduling framework tailored for AI-driven geo-simulation. Built upon the three-layer architecture of OpenGMS, the framework extends it by incorporating external LLMs as decision engines. By combining runtime information sensing, a historical task knowledge base, and real-time computing resource information, the framework enables intelligent runtime prediction and scheduling optimization. Transitioning from static and rule-based scheduling to dynamic and data-driven intelligent scheduling, this approach achieves up to a 3.18-fold speedup in high-concurrency geo-simulation tasks, increases the task success rate to 96.18%, and improves successful throughput by approximately 34.22%. These results demonstrate its capability to realize efficient task allocation and resource utilization in distributed heterogeneous environments, thereby supporting high-concurrency AI-driven geo-simulation tasks.

2. Background

2.1 OpenGMS and the Integration of Geo-simulation Resources

OpenGMS platform was proposed in an era of rapid development of open sharing of geo-simulation resources. Based on an open network environment, OpenGMS is dedicated to discovering and integrating resources related to geographic modeling and simulation research (Wen et al., 2016; Yue et al., 2016; Zhang et al., 2020). By aggregating computing resources across different network structures and adopting a service-oriented design, the platform encapsulates and deploys models and data resources as services that can be shared with users (Wang et al., 2018; Zhang et al., 2020). Moreover, OpenGMS establishes an open and collaborative distributed cyberspace platform, enabling efficient sharing and reuse of geo-simulation resources (Ma et al., 2024; Xu et al., 2024).

As a platform for contributing, sharing, and reusing geo-simulation resources (Chen et al., 2020), OpenGMS categorizes simulation resources into model resources, data resources, and computing resources (Yang et al., 2009; Reddy, 2018), and builds a three-layer architecture consisting of a model execution layer, a task management layer, and a data exchange layer (Figure 1). Within this platform, users can conduct geo-simulation studies to address complex environmental problems. The model execution layer, as the ultimate carrier of geo-simulation tasks, consists of multiple computing nodes equipped with model service containers, forming the computing resources of the distributed cyberspace. Multiple geographic models are deployed within the model service containers,

encapsulated into network-compliant model services according to the Model Description Document (MDL) (Yue et al., 2016; Zhang et al., 2020). The task management layer, directly oriented toward user requirements, mainly includes two core service modules: the management service container for parsing tasks and the task service container for assigning tasks. Based on geo-simulation tasks submitted by users, the task service container dispatches them to different computing nodes. The data exchange layer facilitates the transmission and exchange of data between users and computing nodes.

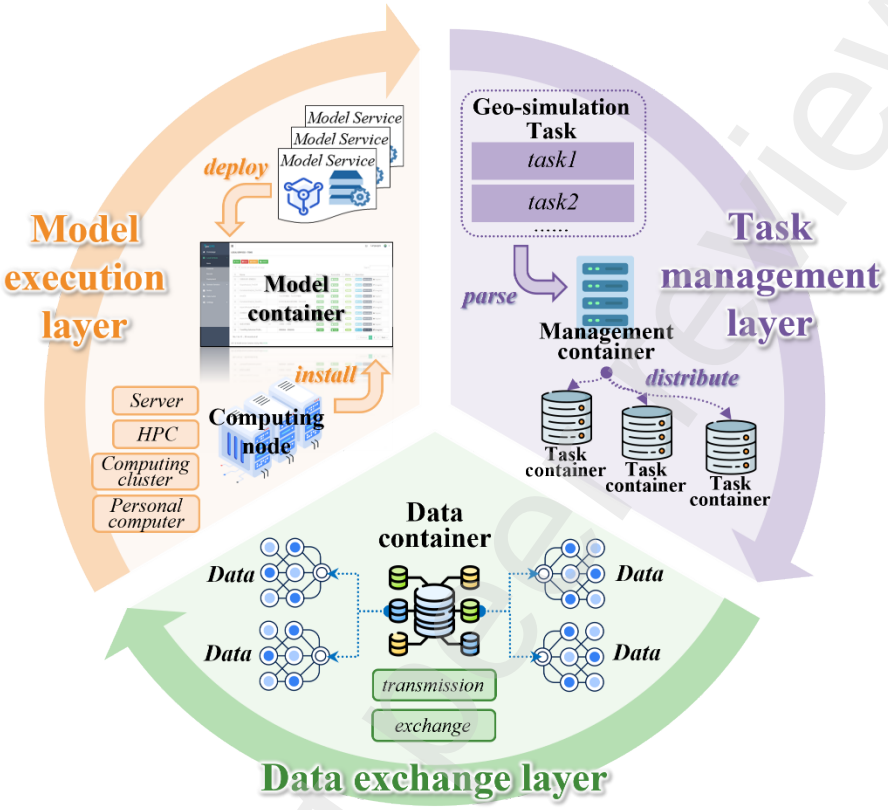

Figure 1 Distributed network architecture of OpenGMS.

2.2 Current Scheduling Strategies for Geo-simulation Tasks

The scheduling process of current geo-simulation tasks consists of three stages: task parsing, task subscription, and task execution (Zhang et al., 2020). First, during the task parsing stage, the management service container in the task management layer parses the MDL to obtain the geo-simulation models, related data, and execution logic required by the user. In the task subscription stage, the management service container sends the parsed task to its associated task service containers. Subsequently, each task service container checks whether the computing nodes it manages provide the target model service based on the model PID. The management service container prioritizes the containers with fewer running tasks to achieve load balancing. During further scheduling, the task service container considers both the resource utilization of computing nodes and the number of tasks currently running, and ultimately assigns the task to a specific computing node. In the task execution stage, the computing node invokes the corresponding model service and interacts with the data service container to download input data and upload output data, thereby completing the entire execution workflow.

Although the current scheduling strategy considers the operational status of the computing node to a certain extent, its evaluation dimension is still limited to the basic indicators such as CPU, memory and the number of tasks. With the growing demand of AI-driven geo-simulation tasks for computing

resources such as GPUs and total Video Memory (VRAM), relying solely on these limited metrics is no longer sufficient to meet practical requirements. Therefore, the evaluation system employed in the scheduling process needs to be more comprehensive. In addition, the weight assignments of these indicators are typically fixed and are not dynamically adjusted according to the characteristics of geo-simulation tasks. As a result, scheduling decisions remain static and subjective. Such a lack of task-aware strategies not only constrains the rationality of task scheduling but also hinders balanced resource utilization in complex computing environments.

3. LLM-Based Dynamic Scheduling Framework

This study proposes an intelligent scheduling framework designed to address the dynamic scheduling problem of AI-driven geo-simulation tasks in distributed heterogeneous clusters. Building upon the existing three-layer architecture of OpenGMS, the framework is enhanced by integrating an external LLM as a decision-making engine, thereby realizing dynamic intelligent scheduling driven by computing node information, model characteristics and historical simulation task data.

3.1 Overall Framework

As shown in Figure 2, the framework consists of three core layers: the model execution layer, the task scheduling layer, and the AI decision engine. The model execution layer is composed of diverse heterogeneous computing nodes, including high-performance computing clusters (HPCs), servers, and personal computers. Each node is equipped with model service containers responsible for receiving AI-driven geo-simulation tasks dispatched by the task service container. These model service containers also periodically transmit their hardware status and model execution information to the task scheduling layer, providing the necessary data foundation for subsequent decision-making.

The task scheduling layer acts as the management and interaction hub of the framework, receiving geo-simulation tasks submitted by the management layer. Unlike current static allocation strategies, this layer undertakes the dual responsibilities of aggregating data and invoking the AI engine. It collects real-time status information from the model execution layer, extracts key performance metrics, and forwards them to the AI decision engine for analysis. Based on the evaluation scores returned by the AI engine, the scheduling layer ultimately assigns geo-simulation tasks to the most suitable computing nodes.

The AI decision engine is powered by LLM and is responsible for processing complex runtime predictions and scheduling decisions. By analyzing multi-dimensional data provided by the scheduling layer, the engine intelligently predicts the runtime of AI-driven geo-simulation tasks on different nodes and dynamically adjusts the weights of evaluation metrics. It then assesses the suitability of each computing node and returns optimized scoring results, providing scientific evidence for the final scheduling decisions.

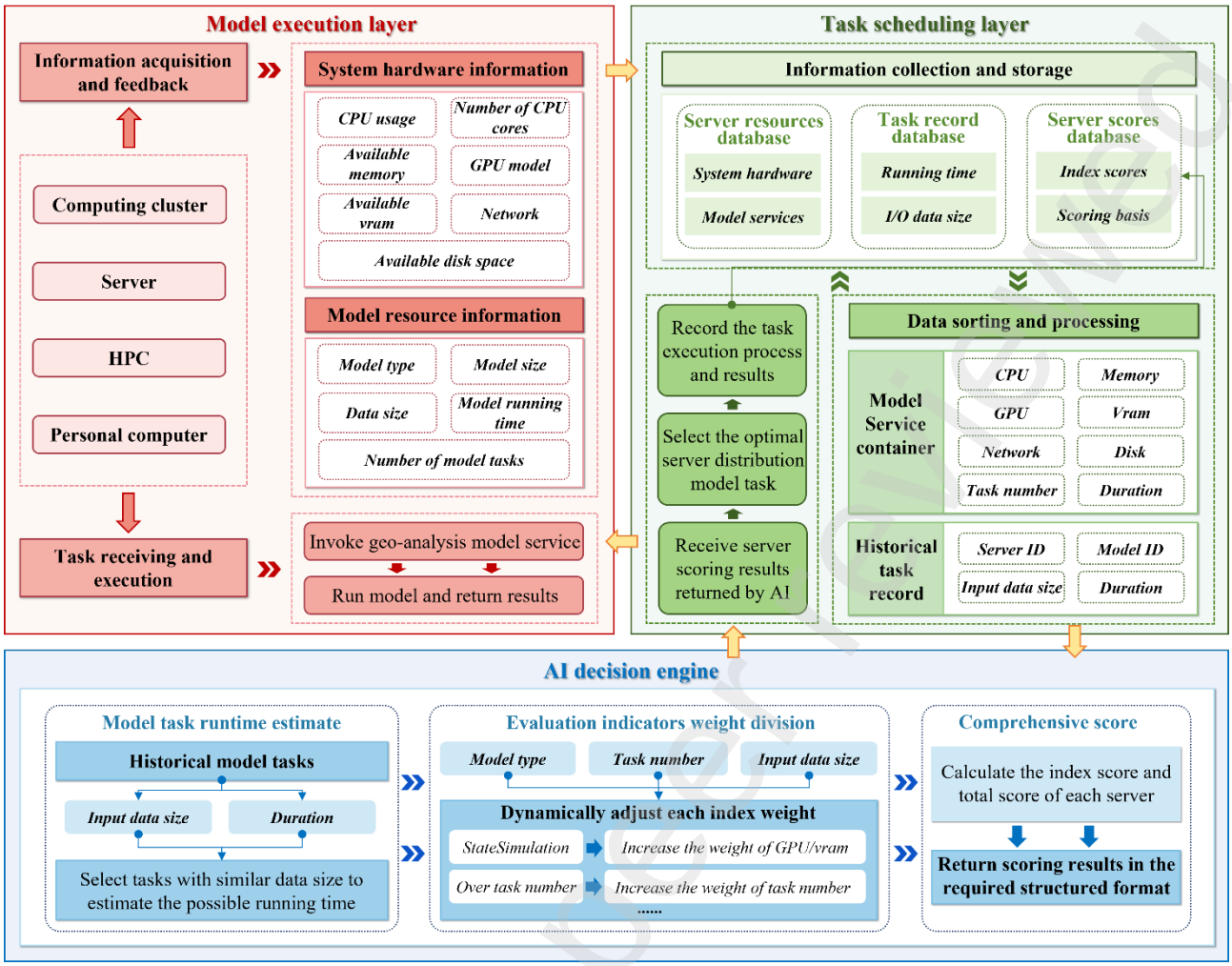

Figure 2 Design of the AI-based dynamic scheduling framework.

3.2 Runtime Information Awareness and Data Flow

To ensure the effective operation of the AI decision engine, the framework establishes a continuous and bidirectional data flow across the three layers. The key information involved includes configuration and deployment details of computing nodes, historical task records, and detailed evaluation scores generated by the AI engine. Together, these elements enable real-time interaction among the model execution layer, task scheduling layer, and AI decision engine. The acquisition and transmission of this information form the core of high-quality scheduling in high-concurrency scenarios.

The configuration of computing nodes and the details of deployed models are essential for the AI to evaluate node performance. This information is periodically collected by the model service containers deployed on each node and sent to the task service container. It includes hardware status such as CPU utilization, number of CPU cores, available memory, GPU type, available GPU memory, network, and remaining disk capacity, thereby providing a comprehensive overview of node performance. In addition, model-related information, such as model IDs and package sizes, is critical for determining whether a node can execute a specific simulation task. All collected data is processed and stored by the task service container in a persistent database, forming the basis for scheduling decisions.

Historical task information provides the data foundation for predicting the runtime of AI-driven geo-simulation tasks on different computing nodes. The execution details of each task are

comprehensively recorded, forming a complete task knowledge base. Moreover, these extensive historical records are structured into traceable execution dataset, which serve to guide the prediction of runtime for subsequent simulation tasks. These records span the entire of each task: when the task service container receives a new geo-simulation task, it begins by logging the task's metadata; after the AI decision engine determines the optimal computing node, the task service container assigns the corresponding server ID; once the model service container completes execution, it reports back the final status, actual runtime, and output results, which are then used to update the database. By constructing this structured historical dataset, LLM is provided with rich training data and is able to continuously learn from new task executions, dynamically estimate the expected runtime of new tasks, and progressively improve prediction accuracy.

The detailed evaluation scores are generated by the LLM-driven decision engine. After receiving real-time configuration and model information from the task service container, along with historical records of geo-simulation tasks, the engine invokes the LLM to analyze task characteristics and dynamically adjust the weights of evaluation metrics. Based on the estimated runtime, dynamic weights, and hardware indicators, it calculates both detailed scores and an overall score for each computing node. These results are formatted into structured data and returned to the task service container, which stores them in the database for traceable decision support and future optimization.

3.3 LLM-Driven Dynamic Scheduling Algorithm

The scheduling algorithm proposed in this study fundamentally differs from current static strategies, as it leverages structured outputs from the AI decision engine as the basis for task assignment. The current OpenGMS scheduling strategy fails to fully consider the heterogeneity of computing node performance, and also ignores the characteristics of geographical simulation tasks. In contrast, the proposed dynamic algorithm comprehensively integrates the hardware configurations of distributed nodes with the intrinsic characteristics of geographic analysis models, while incorporating the predictive power of AI to achieve rapid task scheduling under high concurrency.

Upon receiving node configuration data and historical task records, the AI decision engine executes three key steps: runtime estimation, metric weighting, and evaluation:

(1) Runtime estimation: The AI conducts an in-depth analysis of the characteristics of the current geographical simulation task, including the model type and the size of input data. This analysis is combined with the real-time status of the servers on which the model is deployed, such as hardware configuration and workload while also taking into account historical runtime data of similar tasks executed on the same computing nodes. Based on this comprehensive assessment, the AI ultimately predicts the potential runtime of the task on each server.

(2) Metric weight assignment: The AI dynamically adjusts the weights of evaluation metrics depending on the task type. For example, CPU and GPU weights are increased for computationally intensive tasks, while disk and network bandwidth weights are emphasized for I/O-intensive iterative tasks. When a node's task load approaches its capacity, the weight of current workload is raised accordingly. This flexible adjustment of weights allows the system to dynamically reflect the configuration advantages of different computing nodes, thereby providing support for decision-making in geo-simulation task scheduling. Finally, the LLM calculates the overall score of each server and transmits the structured results back to the task service container.

(3) Evaluation: The AI computes the overall score for each node and sends the structured results to the task service container. The scheduling layer then selects the node with the highest score to execute the task. Moreover, after each task is completed, the system records the AI's decision,

predicted runtime, and actual runtime in the database, continuously enriching the knowledge base. This feedback loop enables the LLM and scheduling strategy to evolve and improve over time.

4. Implementation

The study further implements the proposed LLM-based dynamic scheduling framework by structuring representations of computing nodes and models information, historical task records, and AI evaluation scores. In addition, LLM-oriented prompt engineering and a standardized output structure are designed to collectively support precise and adaptive scheduling in distributed heterogeneous environments.

4.1 Representation of Runtime Information

To transform complex information in distributed heterogeneous environments into inputs that can be understood by LLM, this study structures key data regarding computing nodes, geo-simulation tasks, and AI evaluation results. This structured representation provides the data foundation for dynamic scheduling of AI-driven geo-simulation tasks and AI-based decision-making.

(1) Computing Node Information

The real-time status of computing nodes is the core basis for AI to make dynamic decisions in AI-driven geo-simulation task scheduling. Node information is periodically collected by the model service container and transmitted to the task service container. As shown in Figure 3, the data structure mainly includes:

- **System hardware information:** Basic operating system information reflects the underlying runtime environment and overall stability of a computing node, serving as a prerequisite for deploying and executing geo-simulation tasks. This includes system type, system name, IP, MAC address, and system status. Computing resource information measures the node's core computational capacity, directly determining the efficiency and parallel processing ability for compute-intensive tasks. It includes CPU utilization, number of cores, total memory, available memory, disk capacity, and remaining disk capacity. Some nodes are configured with GPUs, which reflect performance in parallel computing and graphics processing—critical resources for high-performance computing and deep learning-based simulation tasks. This includes GPU model, total VRAM, and available VRAM. Network resources represent the node's performance in data transmission and communication, which is an important determinant of efficiency in distributed execution. It includes network card name, bandwidth, receiving rate, and sending rate.
- **Model resource information:** This describes the static attributes and runtime states of models deployed on a node, serving as an important basis for scheduling and adaptation decisions. It includes unique identifiers and functional attributes (e.g., model ID, type, name, category, description), storage and scale information (e.g., package size, version, number of running tasks), as well as configuration files specifying model workflows and input/output data.

These data are preprocessed and standardized before being stored in the model service container's database, and are periodically transmitted to the task service container, thereby providing a comprehensive foundation for AI engines to evaluate node suitability.

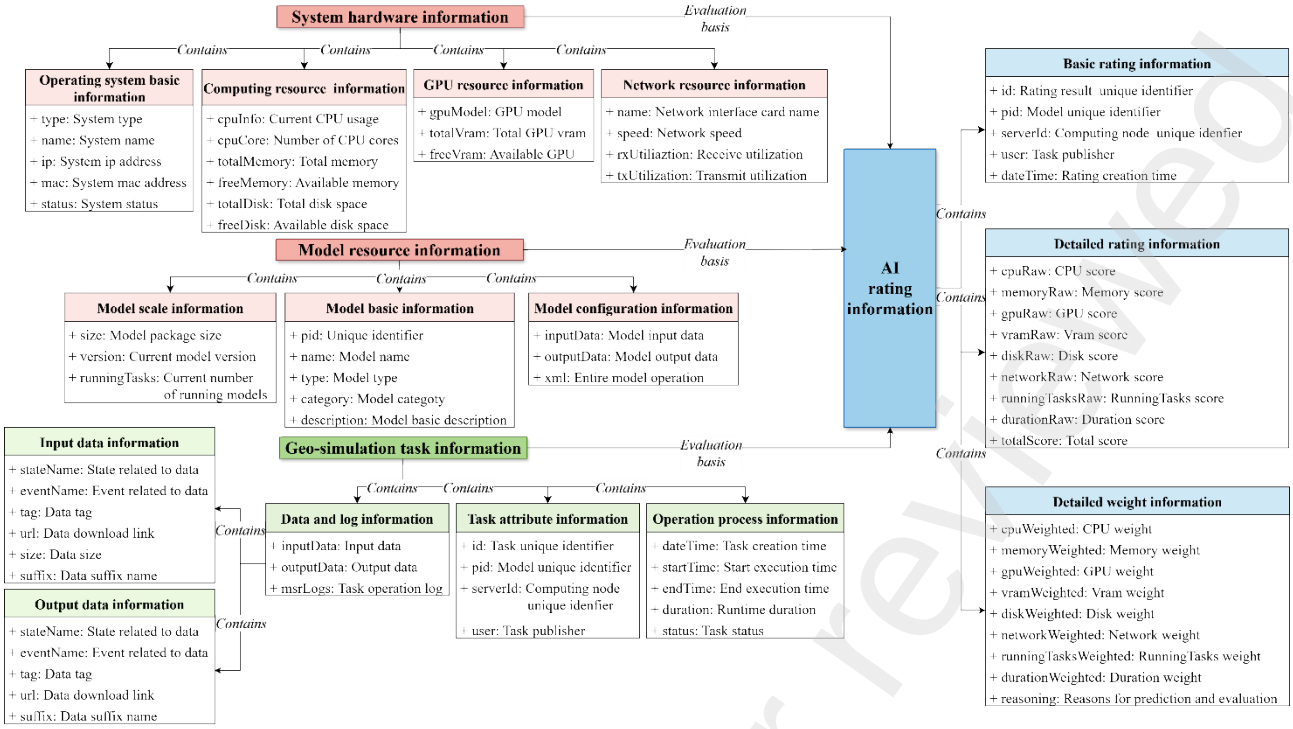

Figure 3 The logical structure of runtime information, including computing node information, geo-simulation task information, and AI evaluation information.

(2) Geo-simulation Task Information

Task information characterizes the fundamental attributes and runtime process of simulation tasks, serving as the primary basis for AI runtime estimation and metric weighting. As shown in Figure 3, the data structure includes: task attribute information that uniquely identifies tasks and their associations with models and computing nodes, including task ID, model ID, node ID, and task publisher; execution process information that reflects the task's lifecycle, including creation time, start time, end time, runtime duration, and task status; data and log information that records model outputs and execution details, including input data, output data, and runtime logs, supporting result traceability and performance optimization. After standardization and structuring, these data provide the AI decision engine with a complete foundation for understanding task requirements, evaluating execution performance, and optimizing scheduling.

(3) AI Evaluation Information

AI evaluation information records the comprehensive assessment results produced by LLM regarding node suitability and task execution efficiency. This not only supports rapid parsing and utilization at the task scheduling layer but also informs subsequent model refinement and scheduling strategy optimization. As shown in Figure 3, the structure includes: the basic attributes of evaluation entries, such as evaluation result ID, model ID, server ID, task publisher, and evaluation timestamp; the predicted runtime of geo-simulation tasks on different computing nodes derived from input data and historical model execution records; and the quantitative evaluation results of key computing node resources, including CPU, memory, GPU, VRAM, disk, network, number of running tasks, and predicted runtime, together with the corresponding dynamic metric weights and overall scores.

4.2 LLM-Driven Intelligent Task Scheduling

The core of intelligent scheduling lies in leveraging LLM to perform real-time evaluation and decision-making based on the characteristics of simulation tasks and computing nodes (Liang et al.,

2024). In this study, prompt engineering is employed to constrain model behavior, combined with strictly defined structured outputs, ensuring accuracy and usability of the results.

(1) Prompt Engineering for Multi-Indicator Optimization

Prompt engineering is the practice of formulating and adjusting prompts to control the behavior of LLMs and obtain desired responses without changing the internal structure of the model (Zixuan Zhou et al., 2024; Liang et al., 2025). To achieve efficient adaptation between AI-driven geo-simulation tasks and computing nodes, a prompt template considering multiple evaluation metrics is designed to guide the LLM in reasonable weight assignment and node selection (Figure 4). The embedded key information includes: the runtime durations of historical similar tasks and the attributes of the servers; the type and resource requirements of the task to be scheduled; and the real-time configuration information of the computing nodes (including CPU, memory, GPU, VRAM, disk, network, and workload). On this basis, the prompt explicitly requires the LLM to accomplish three core tasks:

- **Task runtime prediction:** Estimate the runtime of the current task across different servers based on historical similar geo-simulation task size and duration.
- **Dynamic weight assignment:** Adjust the weights of eight indicators (CPU, memory, GPU, VRAM, disk, network, number of running tasks, predicted runtime) according to task characteristics, ensuring they sum to 1.0. For example, CPU and GPU weights are increased for compute-intensive tasks, disk and network weights for I/O-intensive tasks, and task-load weights when node utilization approaches capacity.
- **Node scoring:** Combine the dynamic weights with a predefined formula to calculate indicator scores and overall scores for each node, thereby quantifying their suitability.

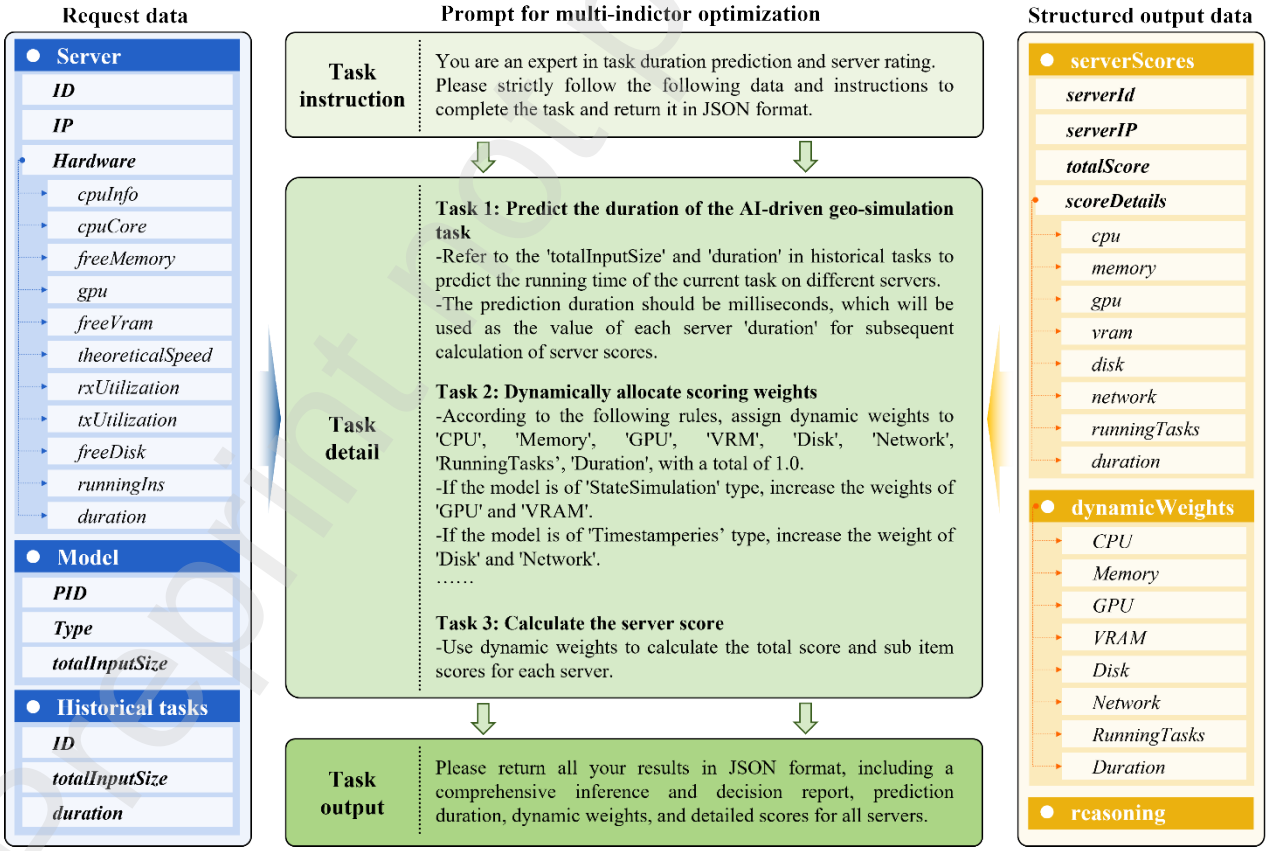

Figure 4 Workflow of prompt engineering for multi-indicator optimization.

(2) Structure and Analysis of LLM Outputs

To ensure the evaluation results can be directly parsed and utilized by the system, a unified JSON output structure is designed for the LLM. As shown in the Figure 4, the structure includes predicted runtimes of tasks across nodes, weight distributions for each indicator, per-node indicator scores and total scores, as well as explanatory reasoning for decisions. After receiving the LLM's output, the task service container ranks nodes based on total scores and selects the optimal one. Meanwhile, the predictions, weight distributions, and AI reasoning reports are logged in the database. Actual runtime and resource usage are subsequently fed back to the AI decision module, forming a closed-loop cycle of prediction–execution–feedback–optimization.

5. Experiment

The study introduces the Gemini 2.5 Flash-Lite into the AI decision engine, and conducts a series of controlled experiments to comprehensively evaluate the performance of the proposed AI-driven dynamic scheduling framework in a distributed heterogeneous cluster environment, comparing it with the current OpenGMS static scheduling strategy to verify its advantages in complex scenarios.

5.1 Experimental Setup and Data Preparation

(1) Test Platform Configuration: OpenGMS Distributed Heterogeneous Cluster

The experiments are carried out on the OpenGMS distributed heterogeneous cluster. This cluster is composed of multiple types of computing nodes contributed by different participants, with some nodes deploying model service containers to support model deployment and invocation. As shown in Table 1, the hardware configurations of the computing nodes vary significantly, covering different CPU architectures, core counts, memory and disk capacities, as well as the presence or absence of GPUs, fully reflecting the "heterogeneous" nature of the system. Such diversity provides the hardware foundation for testing the adaptability and decision-making capability of the scheduling framework under complex environments. Therefore, the proposed dynamic scheduling framework is validated within the OpenGMS distributed heterogeneous cluster environment.

Table 1 Computing node information in the OpenGMS distributed heterogeneous cluster environment.

Server IP	System type	CPU model	core	Total memory	Total disk	GPU model	Total vram
172.21.252.205	Windows_NT	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	16	4G	200G	/	/
172.21.252.206	Windows_NT	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	16	4G	500G	/	/
172.21.252.207	Windows_NT	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	4	8G	400G	/	/
172.21.252.212	Windows_NT	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	8	4G	452G	/	/
172.21.252.236	Linux	Intel Xeon E5-2620 v4	32	6G	1510 789G	/	/

223.2.41.213	Windows _NT	AMD Ryzen 5 4600H with Radeon Graphics	12	16G	185G	NVIDIA GeForce GTX 1650	4096
223.2.45.222	Windows _NT	AMD Ryzen 9 9950X 16-Core Processor	32	32G	778G	NVIDIA GeForce RTX 5090	32607

(2) Benchmark Suite: AI-Driven Geo-simulation Models

This study selects three representative types of geo-simulation models, as shown in Table 2: (i) Vision-LSTM model, which employs a pretrained ResNet18 model to txtract features from and subsequently performs image classification(Huang et al., 2023); (ii) UrbanVCA model, a simulation and prediction model of urban land use change based on real plots and Vector-based cellular automata(CA), which realizes enables dynamic parcel classification, land-use data matching, overall development probability calculation, and simulation of urban land-use change(Yao et al., 2021); (iii) UrbanM2M model, which is built upon a convolutional long short-term memory (ConvLSTM) deep learning framework and simulates future urban expansion scenarios using time-series urban land grid data and spatial variables(Zihao Zhou et al., 2024). These models exhibit distinct characteristics in terms of computing resource demands, aiming to comprehensively evaluate the adaptability of the scheduling algorithm.

In addition, the experiments are designed with multiple configurations from both model and data perspectives to thoroughly validate the robustness of the dynamic scheduling framework. Specifically: for UrbanM2M, three study areas—Area around Taihu Lake, Suzhou City, and Kunshan City—are selected (Figure 5(b)) to examine adaptability across different geographical scales; for the UrbanVCA model, experiments are conducted in two spatiotemporal ranges, Shunde District of Foshan City and Shanghua Village (Figure 5(c)); for the Vision-LSTM model, three datasets consisting of 289, 578, and 1186 street-view images from Shenzhen are used (Figure 5(d)) to test performance under different data scales.

Table 2 Information on geo-simulation models used in the experiments.

Model	Type	Description	Core requirements	Research scope
Vision-LSTM	SimpleCalculation	A model for image classification that extracts features from images using a pre trained ResNet18 model, and then classifies the images using an LSTM mode.	CPU、GPU	289 street view images
			CPU、GPU	578 street view images
			CPU、GPU、network	1186 street view images
UrabnVCA	TimeSeries	A simulation and prediction model for urban land use change based on real land parcels and vector cellular automata, which realizes vector dynamic land parcel classification, land use data	CPU、memory、network	Shanghua
			CPU、memory、network	Shunde

		matching, overall development probability calculation, and simulation functions.		
UrbanM2M	StateSimulation	A deep learning model based on ConvLSTM uses time-series urban land grid data and spatial variables to simulate future urban expansion scenarios.	CPU、GPU、memory、vram CPU、GPU、memory、vram、network CPU、GPU、memory、vram、network、disk	Kunshan Suzhou Area around Taihu Lake

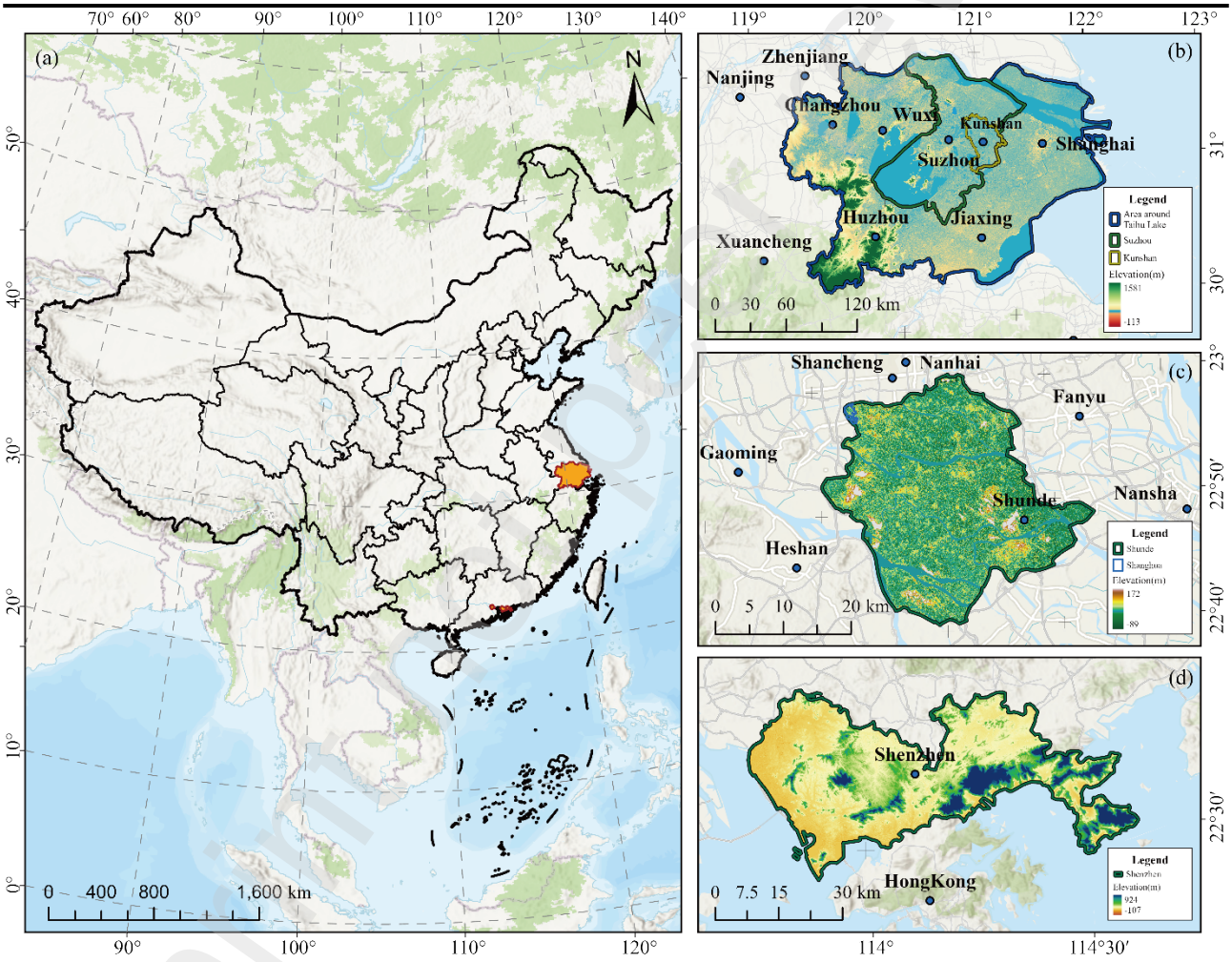

Figure 5 Study areas of the experiments (a. Location of the study area at the national scale; b. Three-level study areas for UrbanM2M; c. Two-level study areas for UrbanVCA; d. Street-view image selection range for Vision-LSTM).

5.2 Experimental Scenario I: Decision Process Analysis of a Single Model Task

This experiment aims to demonstrate the decision-making mechanism and advantages of the proposed dynamic scheduling framework at the individual task level, validating its suitability for integrating resources and intelligently allocating tasks in a distributed heterogeneous environment.

(1) Performance of Geo-simulation Models across Different Computing Nodes

To quantify the impact of the heterogeneous cluster on task runtime, benchmark tests were conducted. Specifically, three geo-simulation models and their corresponding study areas were selected, resulting in eight AI-driven geo-simulation tasks in total. Each task was executed independently seven times across seven computing nodes in the OpenGMS distributed heterogeneous cluster (with all nodes in an idle state) to obtain stable results. As shown in Figure 6, significant differences exist in the runtimes of different geographic tasks across nodes with varying performance levels.

Among them, the task with the largest runtime difference was the UrbanM2M model for the area around Taihu Lake. On the high-performance node 223.2.45.222, the average runtime was only 104.63s, while on node 172.21.252.206, the average runtime reached 6737.40s, demonstrates the inefficiency and infeasibility of current static scheduling strategy that relies on random allocation. Moreover, for deep learning models such as UrbanM2M and Vision-LSTM, runtime was highly dependent on GPU performance. The high-performance GPU node 223.2.45.222 consistently held an absolute advantage, with runtimes far lower than those on other nodes. For instance, the Vision-LSTM model with 1186 street-view images required an average of 12.63s on the fastest node, compared to 63.66s on the slowest. This clearly proves that scheduling frameworks must prioritize hardware dependencies of tasks; otherwise, substantial runtime waste will occur.

In contrast, for iterative tasks like UrbanVCA, which rely heavily on CPU and network, although runtime differences still existed, the performance gap was less pronounced compared to GPU-intensive tasks. The node 172.21.252.236, with its large number of CPU cores, demonstrated a clear advantage. This indicates that even for CPU-intensive geographic tasks, intelligent scheduling frameworks are still required to select the optimal nodes due to differences in operating systems and CPU performance.

In summary, the benchmark test results strongly confirm the high heterogeneity of the OpenGMS distributed cluster and the resulting significant uncertainty in geographic task runtimes. Such complex and non-linear performance differences cannot be reasonably addressed through simple rule-based scheduling. To achieve optimal task allocation, AI-based runtime prediction and comprehensive evaluation of task characteristics, node configurations, and historical data are essential.

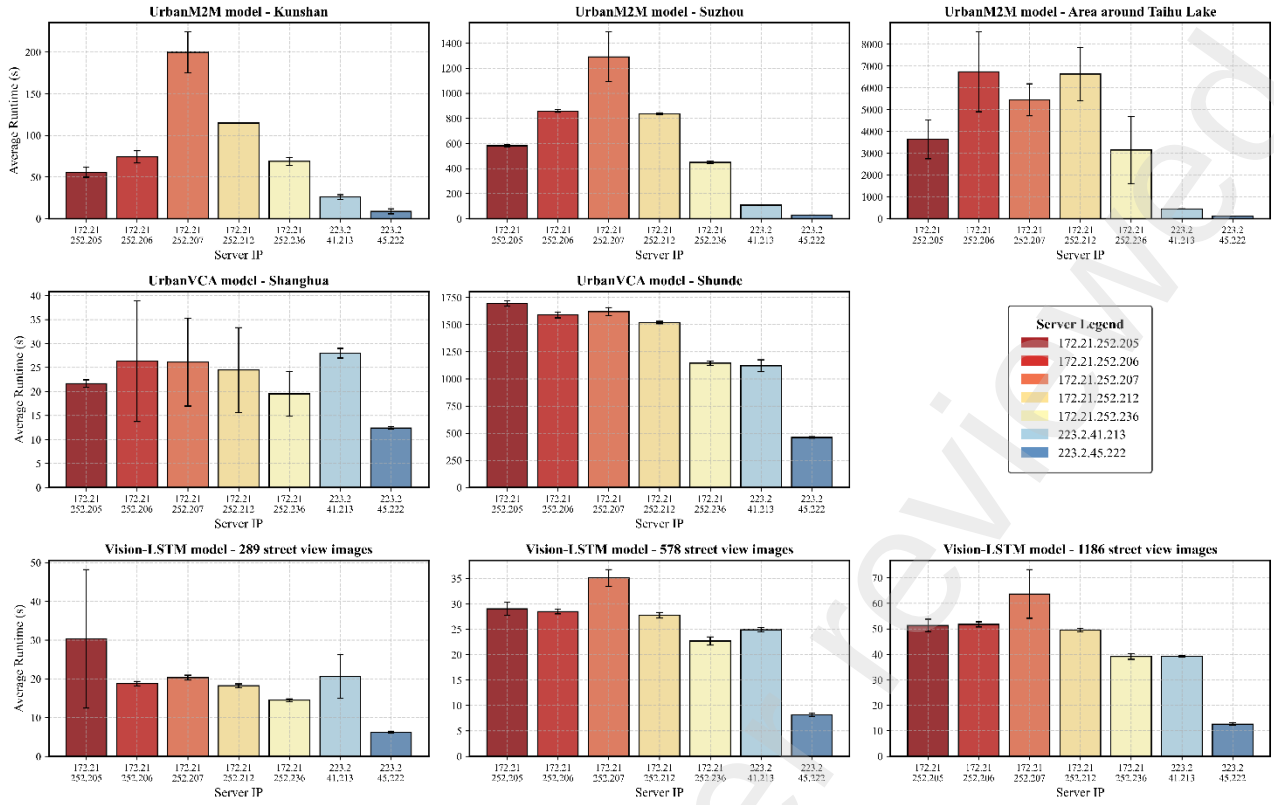

Figure 6 Average running time of AI-driven geo-simulation tasks across different computing nodes.

(2) Performance Comparison between AI-based Dynamic Scheduling and Current Scheduling Strategies

Building on the above experiments, this study further designed a heterogeneous-idle scheduling scenario, where all computing nodes were idle, to compare the performance of the current static scheduling strategy with the AI-driven dynamic scheduling strategy. This was done to verify the advantages of LLM-driven dynamic scheduling in task–resource matching.

Table 3 presents the task allocation, actual runtime, task dispatch time, data transfer time, and total execution time under the current static scheduling strategy. Due to the lack of awareness of task characteristics and computing node resources, GPU-intensive tasks were often randomly allocated to low-performance nodes, resulting in substantial runtime waste. For example, the UrbanM2M (Area around Taihu Lake) task, which required high GPU, VRAM, and disk resources, was allocated to node 172.21.252.207, resulting in a total execution time of 5108.07s, of which 4843.32s were spent on computation. Similarly, the UrbanVCA (Shunde) task, which is not GPU-intensive but requires high memory and network bandwidth, was assigned to node 172.21.252.206, with a total time of 1644.67s, including 1578.72s runtime.

Furthermore, high-performance GPU nodes such as 223.2.45.222 and 223.2.41.213 were not utilized under this strategy. These observations indicate that current scheduling strategy is highly prone to assigning geographic tasks to underperforming nodes, leading to substantial resource waste.

Table 3 Task allocation and runtime of AI-driven geo-simulation tasks under the current scheduling strategy.

Model	UrbanM2M model			UrbanVCA model		Vision-LSTM model		
Scope	Kunshan	Suzhou	Area around Taihu	Shanghua	Shunde	289 street view	578 street view	1186 street view

			Lake			images	images	images
Computing node	172.21.252.212	172.21.252.212	172.21.252.207	172.21.252.236	172.21.252.206	172.21.252.206	172.21.252.207	172.21.252.206
Transmission time(s)	12.57	54.57	264.75	26.95	65.95	42.86	34.54	37.15
Run time(s)	129.08	896.68	4843.32	32.29	1578.72	41.31	62.81	952.86
Total time(s)	141.65	951.25	5108.07	59.24	1644.67	84.17	97.35	990.01

Table 4 records the allocation of AI-driven geo-simulation tasks and the breakdown of execution time under the AI-driven dynamic scheduling strategy. The experiments demonstrate that, because the AI scheduling strategy leverages the LLM to accurately determine the resource requirements of AI-driven geo-simulation tasks, all tasks were scheduled to the optimal node 223.2.45.222, leading to substantial runtime optimization. For example, the UrbanM2M (Area around Taihu Lake) task and the Vision-LSTM (1186 street view images) task achieved speedups of 23.76× and 24.92× in total execution time, respectively. Furthermore, even for the UrbanVCA (Shunde) task, which primarily relies on CPU resources, a 2.59× acceleration was achieved, proving that AI decision-making is equally effective in fine-grained allocation of general-purpose computing resources.

Figure 7 illustrates the comprehensive evaluation of different computing nodes across eight dimensions—CPU, memory, GPU, VRAM, disk, network, number of running tasks, and duration—under the AI dynamic scheduling framework. It is evident that high-performance nodes scored significantly better on key indicators, thus ranking much higher in the overall evaluation and ensuring efficient task–resource matching. Taking the UrbanM2M (Area around Taihu Lake) task as an example, the AI decision engine correctly identified the geo-simulation task type as "StateSimulation", and therefore assigned higher dynamic weights to GPU and VRAM dimensions. Since the total input data size was 984MB, the task was categorized as "big data," which in turn increased the weights for network bandwidth and disk capacity. Consequently, the optimal node 223.2.45.222, equipped with an RTX 5090 GPU and 32GB of VRAM, achieved the highest scores in these key dimensions. Although its available disk space (300GB) was not the largest among all nodes, it was still relatively high, contributing to its outstanding overall score, which appeared as a nearly perfect octagon in the radar chart. This intelligent evaluation and dynamic weighting mechanism, tailored to task-specific requirements, is the core of the AI decision engine in ensuring efficient matching between tasks and optimal resources.

The results above show that the AI-driven dynamic scheduling framework not only overcomes the problems of resource waste and mismatch inherent in static strategies, but also significantly improves task execution efficiency in heterogeneous clusters, validating its practical application value in complex computational environments.

Table 4 Allocation and execution times of AI-driven geo-simulation tasks under the AI scheduling strategy.

Model	UrbanM2M model			UrbanVCA model		Vision-LSTM model		
Scope	Kunshan	Suzhou	Area around Taihu Lake	Shanghua	Shunde	289 street view images	578 street view images	1186 street view images
Computing node	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222

Transmission time(s)	19.93	56.66	110.19	28.56	61.29	14.48	24.03	24.86
Run time(s)	8.33	35.41	104.79	12.61	573.87	6.36	10.39	14.87
Total time(s)	28.26	82.07	214.98	41.17	635.16	20.84	34.42	39.73

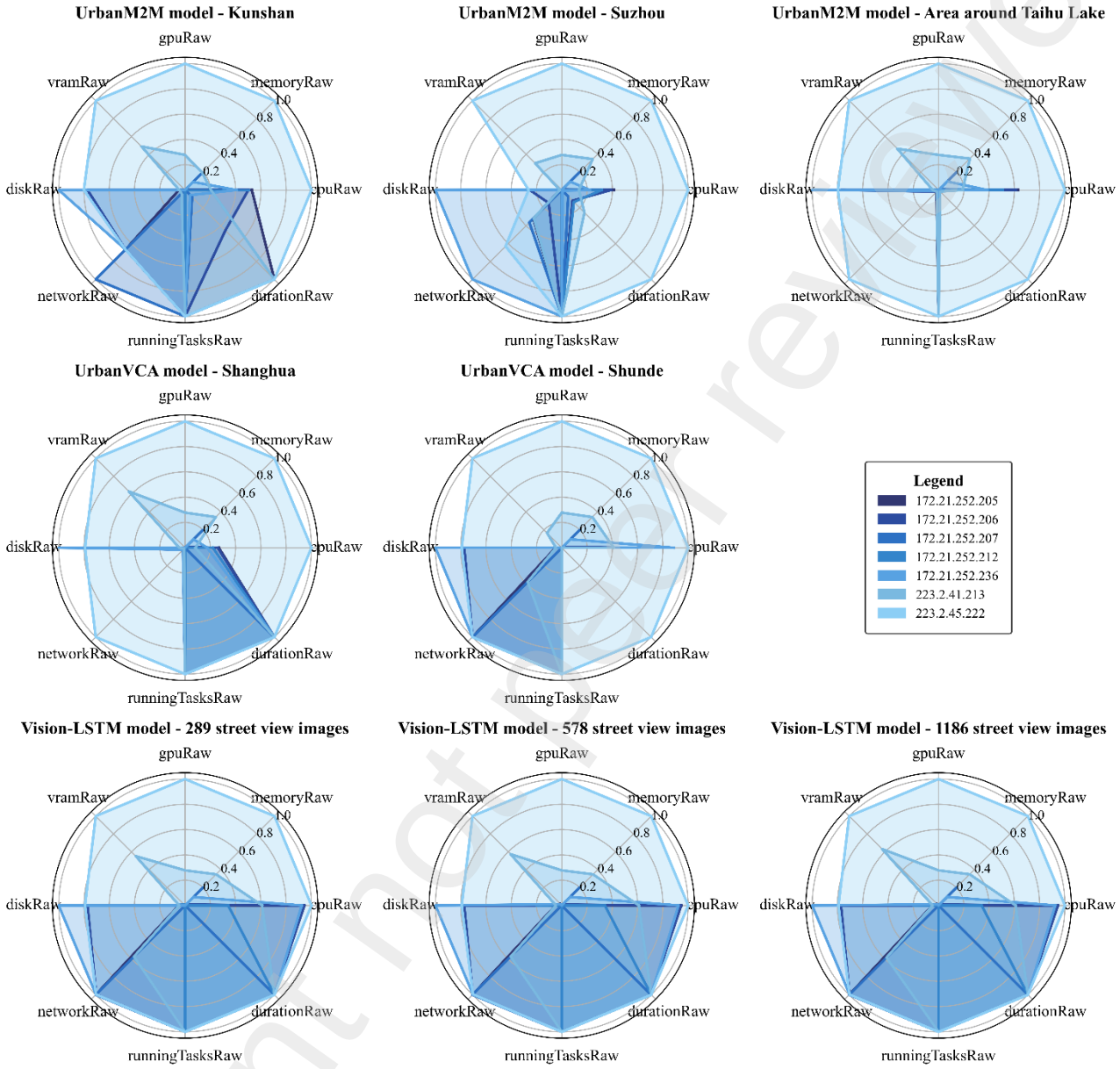

Figure 7 Evaluation of different computing nodes for a single model task.

5.3 Experimental Scenario II: Decision-Making Process under Equal Task Volume

This experiment aims to simulate batch and repeated execution of the same geo-simulation model. Specifically, three core tasks—Vision-LSTM (289 street view images), UrbanVCA (Shanghua), and UrbanM2M (Suzhou)—were each executed 400 times concurrently. The focus was to compare the AI dynamic scheduling strategy and the current static scheduling strategy in terms of total completion time and execution success rate under high-concurrency conditions.

(1) Vision-LSTM Model

As shown in Figure 8(a), under the current static scheduling strategy, the execution process of Vision-LSTM tasks can roughly be divided into three phases. In the first phase, due to the lack of task awareness and fine-grained resource evaluation, the static strategy randomly distributes tasks

across computing nodes, resulting in a relatively balanced distribution of tasks. However, as execution progresses, some low-performance nodes exhibit significantly prolonged runtimes, leading to the second phase. At this stage, in order to maintain overall throughput, subsequent tasks are redirected to unoccupied high-performance nodes. In the third phase, tasks initially assigned to low-performance nodes remain unfinished, causing their resources to be locked for extended periods. As a result, high-performance nodes receive a growing accumulation of tasks and gradually dominate execution. For example, node 172.21.252.206 was initially assigned only 4 tasks but had an average runtime of 6614.18s, resulting in almost no subsequent assignments. Similarly, node 172.21.252.212 was heavily assigned in the first phase but had an average runtime of 4841.41s, limiting its scheduling advantage. In contrast, node 223.2.45.222 continuously received tasks in later phases, ultimately handling 231 tasks with an average runtime of just 35.15s, highlighting its significant performance advantage.

Figure 8(b) illustrates the AI dynamic scheduling strategy, where tasks are primarily concentrated on a few high-performance nodes. Although Vision-LSTM does not strongly depend on GPU or VRAM, the AI scheduler still prioritized nodes with the highest overall performance scores, significantly reducing average runtime per task. The execution can be divided into two phases: In the first phase, node 223.2.45.222, with outstanding GPU and VRAM capabilities, achieved the highest score and became the primary executor. As task volume increased, this node approached its load limit, leading to the second phase, where secondary nodes were introduced for load balancing. For instance, node 172.21.252.236 offered large disk capacity and high memory, while node 223.2.41.213 had GPU resources and historically faster execution for this task, thus receiving assignments. Over time, as the system approached saturation, other nodes such as 172.21.252.207 (with strong CPU and memory performance) gradually took on tasks to maintain balance. Notably, node 223.2.42.222 was allocated 309 tasks with an average runtime of just 13.39s, showing stable efficiency even under heavy load, thereby demonstrating the AI strategy's advantage in optimal resource matching and efficient task execution.

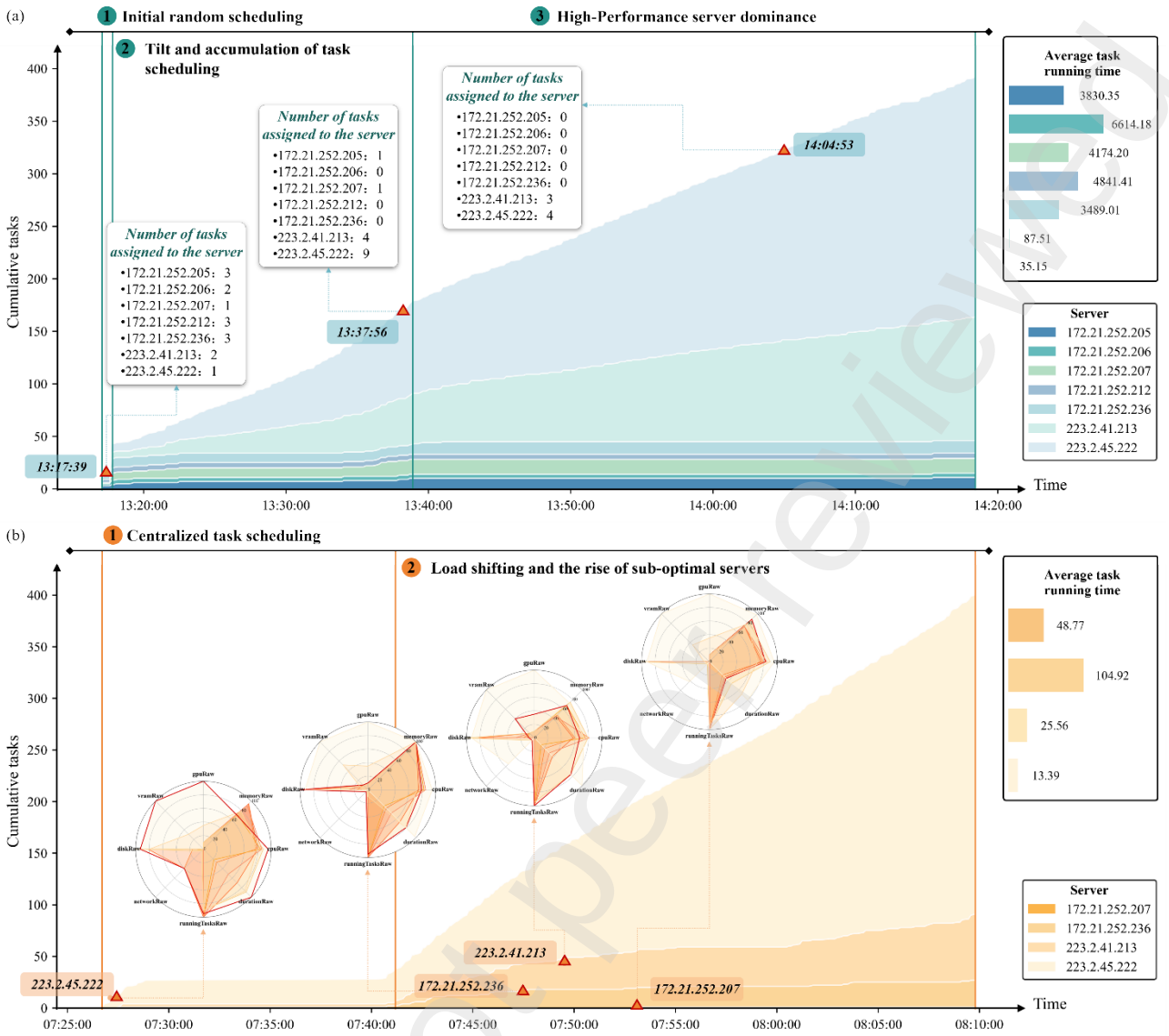

Figure 8 Execution of 400 Vision-LSTM tasks (a. Current scheduling; b. AI scheduling).

(2) UrbanVCA Model

As shown in Figure 9(a), under the static scheduling strategy, UrbanVCA tasks were distributed relatively randomly and evenly. Since this model primarily relies on GPU, memory, and network, but the static scheduler does not account for task features, tasks were nearly equally distributed across nodes in the initial stage, and the cumulative task curves rose in a smooth, linear fashion. Moreover, runtime differences across nodes were not significant, reducing the likelihood of resource blocking. Although such balanced distribution prevents individual nodes from being overloaded, some low-performance nodes reveal computational bottlenecks in the later stages. For instance, node 172.21.252.236 had an average runtime of 268.31s, whereas nodes 223.2.45.222, 223.2.41.213 and 172.21.252.236 achieved much lower runtimes. Overall, the static strategy failed to account for performance disparities, limiting overall efficiency.

By contrast, Figure 9(b) shows the AI dynamic scheduling results. Initially, tasks were concentrated on the optimal node 223.2.45.222, which scored highest across CPU, GPU, and VRAM dimensions. As task volume increased, the scheduler gradually introduced secondary nodes to share the load. For example, node 172.21.252.236 contributed due to its high memory and network, while node 223.2.41.513 was utilized thanks to its strong memory and favorable historical task performance.

Additionally, node 172.21.252.207, with strong CPU capacity, was assigned tasks. Overall, the AI strategy effectively identified CPU and memory intensive requirements and allocated resources accordingly. Unlike the static strategy, which had runtimes exceeding 200s in some cases, AI scheduling reduced average task runtimes to 18–82s. This demonstrates that AI-driven scheduling can accurately distinguish node performance in heterogeneous environments, preventing low-performance nodes from becoming bottlenecks, and substantially improving both efficiency and stability.

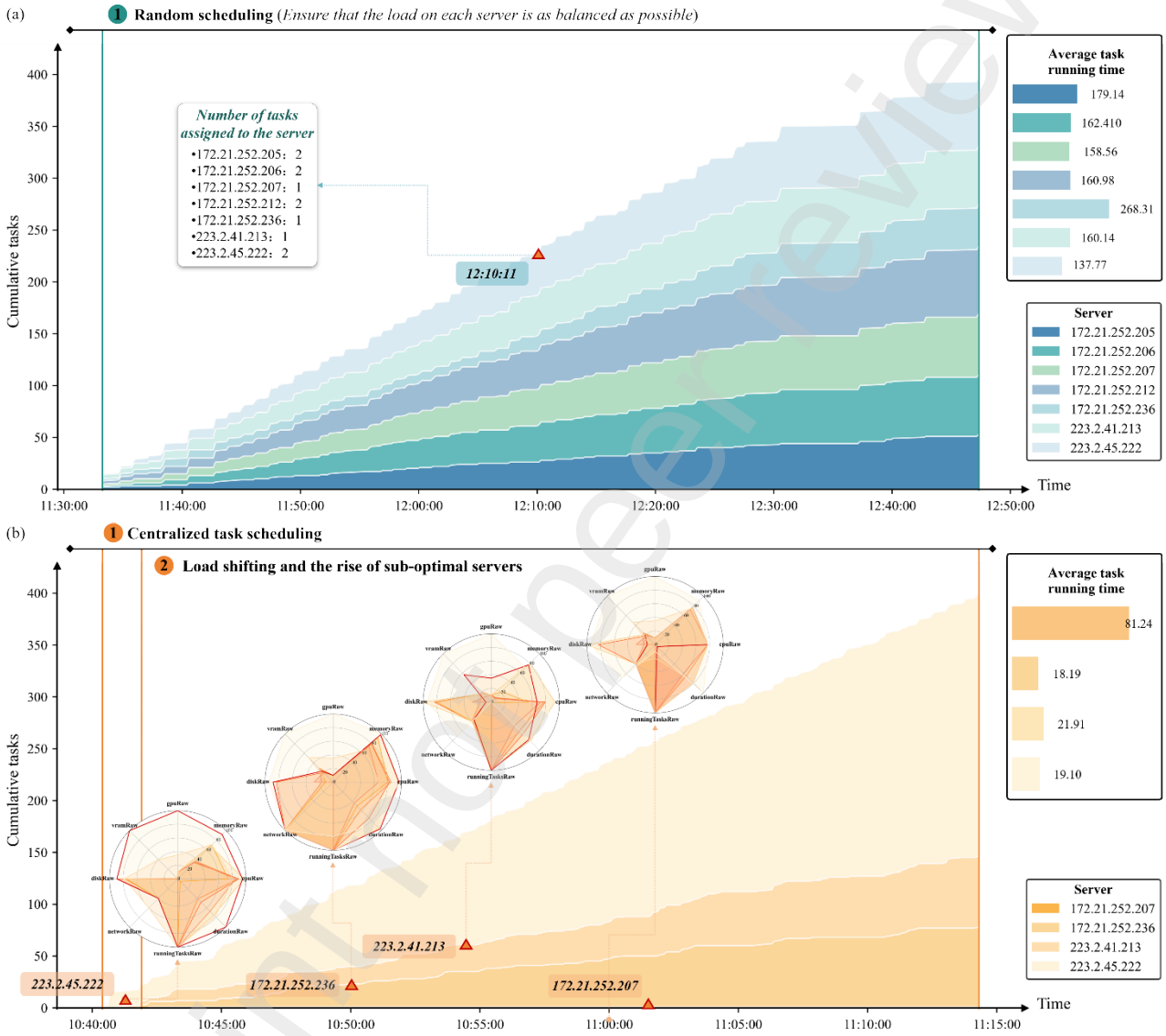

Figure 9 Execution of 400 UrbanVCA tasks (a. Current scheduling; b. AI scheduling).

(3) UrbanM2M Model

From Figure 10(a), it can be observed that under the static scheduling strategy, the task allocation of the UrbanM2M model exhibits a “broad and scattered” pattern. The current scheduling strategy fails to account for the characteristics of AI-driven geo-simulation tasks, instead distributing tasks across servers in an approximately even manner, which leads to significant disparities in execution efficiency among different nodes. The entire execution process can be divided into three stages: in the first stage, tasks are relatively evenly distributed among computing nodes, with little performance variation across them; in the second stage, the differences between nodes gradually widen, with high-performance nodes such as 223.2.45.222 and 223.2.41.213 being assigned an increasing number of

tasks; in the third stage, due to prolonged execution times on low-performance nodes occupying resources, and with the growing number of tasks, nodes like 223.2.45.222 and 223.2.41.213 undertake the majority of the workload and become dominant. Under this strategy, some low-performance nodes, such as 172.21.252.207, become a burden because of their long execution times, with an average runtime as high as 1403.05 seconds. Since UrbanM2M geo-simulation tasks have high dependency on GPU and VRAM, the overall task completion time is significantly prolonged under this scheduling strategy due to performance bottlenecks, with the cumulative task curve flattening at the tail, exhibiting a clear "long-tail effect." This indicates that static scheduling suffers from severe inefficiencies in resource utilization and is incapable of meeting the complex and diverse computational requirements of the UrbanM2M model tasks.

In contrast, Figure 10(b) highlights the significant optimization achieved by AI dynamic scheduling. The scheduler quickly identified node 223.2.45.222 as optimal in the initial stage, given its near-perfect radar chart across all dimensions, thus assigning the majority of tasks to it. As the task scale expanded, the system gradually introduced suboptimal servers to enable load migration and resource compensation. For example, node 223.2.41.213, equipped with GPU support, accelerated the execution of the geo-simulation task, while node 172.21.252.236, with its high storage capacity and CPU capability, also undertook part of the workload. In addition, due to the complexity of this geo-simulation task, nodes 172.21.252.206 and 172.21.252.212 undertook part of the subsequent workload because of their advantages in CPU performance and task-handling capacity. Since UrbanM2M is highly heterogeneous in resource demands across phases, the AI strategy's dynamic adjustment ability was particularly evident. Overall, AI scheduling not only reduced average runtime (node 223.2.45.222 processed 299 tasks with an average runtime of just 20.74s) but also enabled rapid task accumulation through multi-node collaboration, effectively mitigating the long-tail effect seen in static scheduling. This further proves the adaptability and superiority of AI-driven resource-aware scheduling for complex heterogeneous tasks.

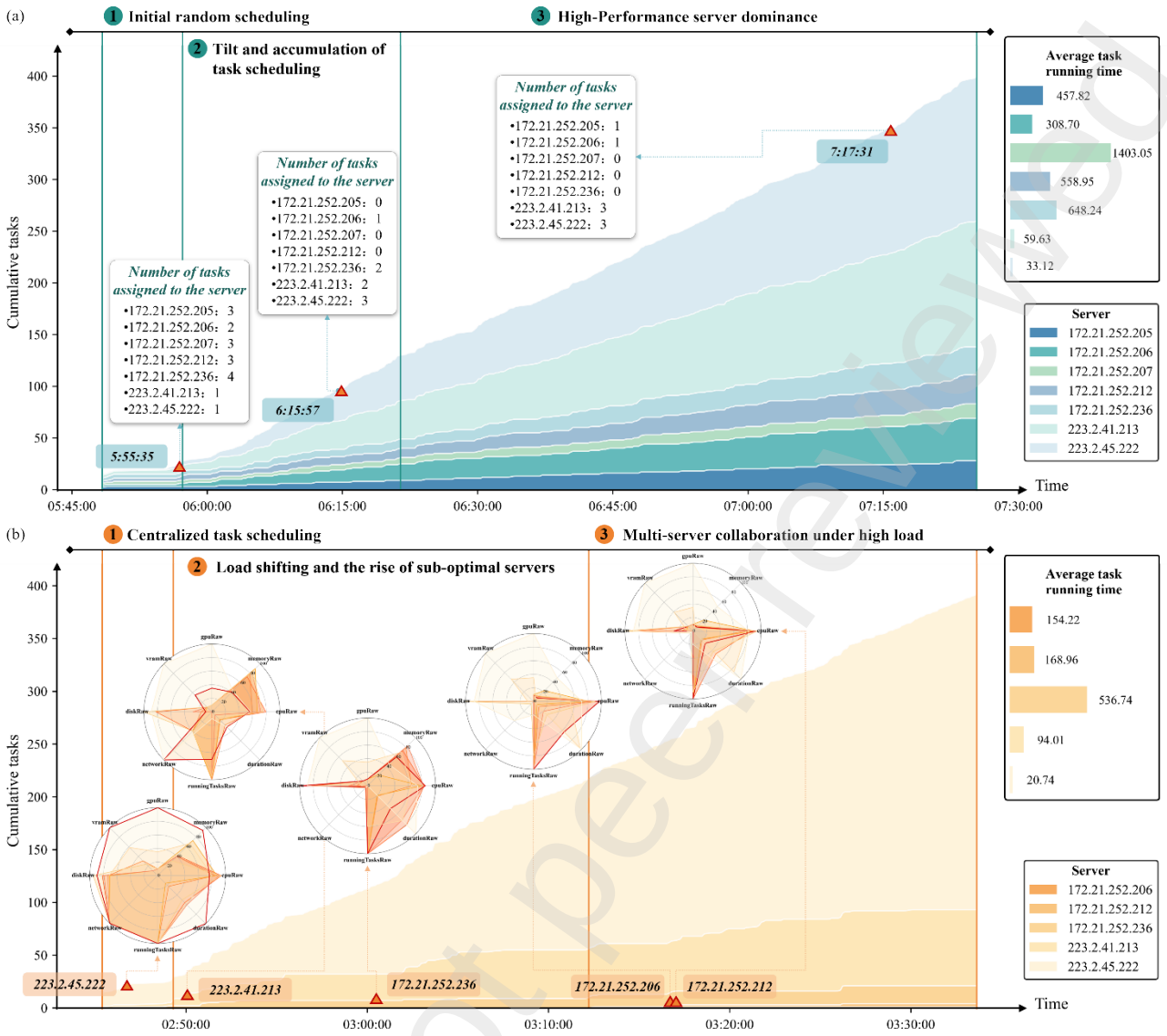

Figure 10 Execution of 400 UrbanM2M tasks (a. Current scheduling; b. AI scheduling).

(4) Comparative Analysis of AI Dynamic Scheduling and Current Scheduling

To further validate the applicability and advantages of the AI dynamic scheduling strategy in AI-driven geo-simulation tasks, this study compares its performance with that of current scheduling across three types of tasks, as shown in Table 5.

Table 5 Performance speedup comparison of AI dynamic scheduling strategy versus current scheduling strategy

Geo-simulation task	Characteristics	Scheduling policy	Number of successful tasks	Number of failed tasks	Total runtime
Vision-LSTM model(289 street images)	CPU、GPU	Current scheduling strategy	397	3	8414.81s
		AI dynamic scheduling strategy	397	3	2648.00s

UrabnVCA model(shanghua)	CPU、memory、network	Current scheduling strategy	387	13	4788.54s
		AI dynamic scheduling strategy	399	1	2123.40s
UrabnM2M model(kunshan)	CPU、GPU、memory、VRAM	Current scheduling strategy	391	9	6316.83s
		AI dynamic scheduling strategy	370	30	3934.37s

As can be seen from the Table 5, the AI dynamic scheduling strategy significantly reduced the total execution time for all three geo-simulation models. Specifically, the total execution time for the Vision-LSTM model dropped from 8414.81s to 2648.00s, achieving an acceleration ratio of approximately 3.18×; for the UrbanVCA model, the time was reduced from 4788.54s to 2123.40s, with a 2.25× speedup; and for the UrbanM2M model, the time decreased from 6316.83s to 3934.37s, representing a 1.61× improvement. At the same time, task success rates were also improved: for UrbanVCA, the number of failed tasks decreased from 13 to 1, further demonstrating the advantage of AI scheduling in multidimensional resource allocation. However, for the UrbanM2M model, due to its large task scale and stronger dependence on heterogeneous resources, the AI scheduling strategy experienced resource contention and scheduling deviation in some phases, resulting in the number of failed tasks increasing to 30. This indicates that although AI scheduling offers overall efficiency gains, further improvements are required in fault tolerance and redundant resource configuration when dealing with resource-sensitive, high-concurrency tasks.

In summary, the AI dynamic scheduling strategy demonstrates notable efficiency gains and resource adaptability across different types of geospatial simulation tasks, particularly excelling in scenarios with relatively balanced demands on CPU, GPU, and memory. Nevertheless, in heterogeneous high-load tasks with frequent resource contention, there remains room for improvement, and future research should focus on the integration of intelligent scheduling with fault-tolerance mechanisms.

5.4 Comparison of Task Execution Counts within Equal Time Frames

In real-world environments, the core performance indicators of a task scheduler are mainly reflected in task throughput per unit time and overall system stability. This experiment simulates a continuous 4-hour runtime, during which 8 types of AI-driven geo-simulation tasks are randomly invoked. It compares the AI-driven dynamic scheduling strategy with the current static scheduling strategy in terms of task-handling capacity and task state distribution within a fixed time window, in order to verify the performance advantages of AI-based scheduling under high-concurrency, long-duration scenarios.

(1) Current Static Scheduling Strategy

As shown in Figure 11, under the current scheduling strategy, the task allocation process exhibits obvious randomness and dispersion. This lack of task feature recognition and resource-awareness results in serious efficiency bottlenecks. In the early stages, because the scheduler failed to analyze task characteristics and compute node performance, complex simulation tasks—such as UrbanM2M

(Area around Taihu Lake), UrbanVCA (Shunde), and Vision-LSTM (1186 street view images), which have high dependency on CPU, GPU, and network—were assigned to low-performance compute nodes such as 172.21.252.205 and 172.21.252.206. This caused extremely long task runtimes, prolonged occupation of computing resources, and a higher probability of task failures, significantly slowing down the overall workflow. For example, node 172.21.252.212 continuously received two UrbanVCA (Shunde) tasks early on, consuming most of its computing resources and significantly reducing its parallel capacity. In addition, high-performance servers were underutilized during the early phase, and their computing power was not effectively harnessed. It was not until the later phase that the system gradually transferred more tasks to high-performance nodes, revealing a belated optimization trend in task scheduling.

From Figure 11(a), it can be seen that high-performance compute nodes continuously accepted more tasks in the later stage and eventually bore the primary workload. Moreover, Figure 11(b) shows that nodes 172.21.252.205 and 172.21.252.206, having undertaken resource-sensitive tasks early on, ended up with significantly fewer completed tasks. In contrast, node 223.2.45.222 kept receiving tasks during later stages, resulting in a cumulative task count far higher than other nodes.

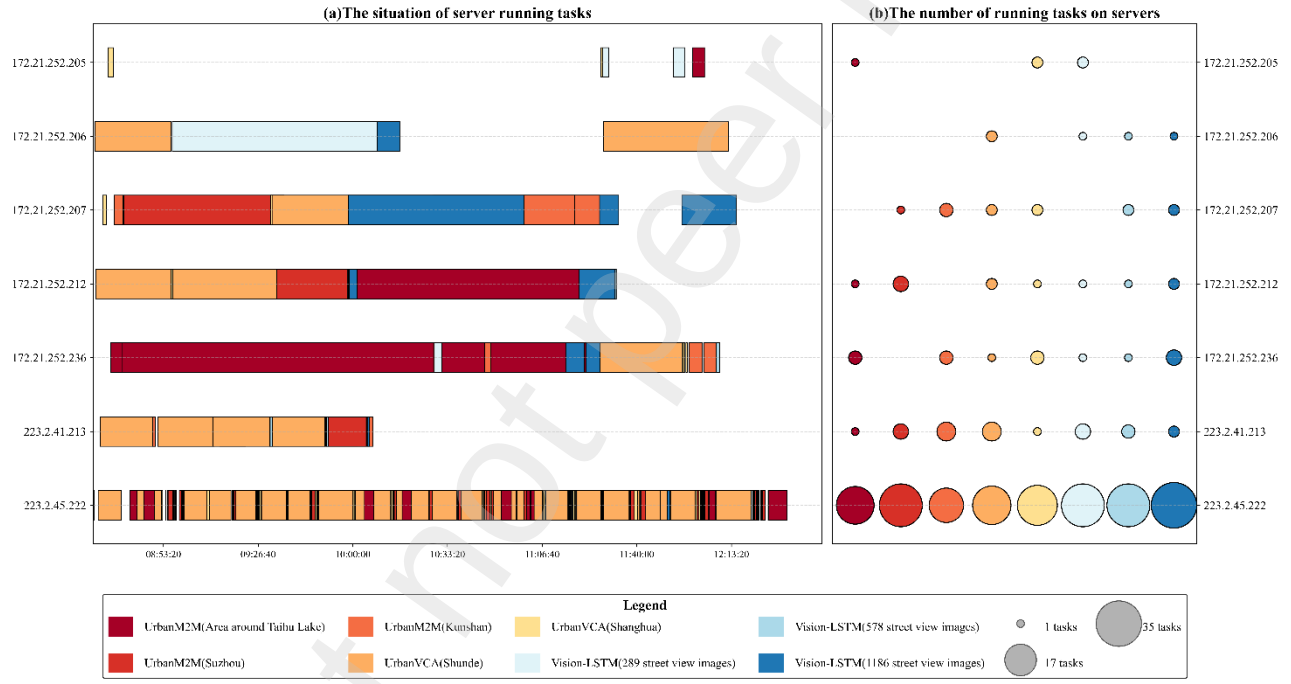

Figure 11 Task allocation within 4 hours under the current scheduling strategy (a. The situation of server running tasks, b. The number of running tasks on servers).

(2) AI Dynamic Scheduling Strategy

Figure 12 demonstrates the significant advantages of the AI dynamic scheduling strategy, whose core lies in precise awareness of heterogeneous resources and priority-based allocation. At the start of task execution, the AI scheduling framework, leveraging the LLM, quickly identifies the optimal compute nodes and concentrates tasks on high-performance servers. For instance, when multiple computationally intensive geospatial tasks such as UrbanM2M (Area around Taihu Lake) and Vision-LSTM (1186 street view images) were submitted early on, the LLM accurately recognized the model characteristics and, based on dynamic performance information of compute nodes, assigned these tasks to the high-performance node 223.2.45.222. Only when this high-performance node reached its load saturation threshold did the AI scheduler begin allocating tasks to secondary nodes, such as

172.21.252.236 and 223.2.41.213, chosen for their large disk capacity, high memory, and GPU configuration. Vision-LSTM (1186 street view images) and UrbanM2M (Suzhou) tasks were accordingly assigned to these supplementary nodes. As high-performance nodes gradually reached their maximum task capacity, the scheduler dynamically distributed additional tasks to other servers according to model-server compatibility, achieving capacity-aware load balancing. This ensured effective use of the overall cluster's resources while preventing low-performance nodes from becoming bottlenecks. Furthermore, as shown in Figure 12(b), the high-performance node 223.2.45.222 carried the majority of executed tasks. Since UrbanM2M (Area around Taihu Lake) is highly dependent on GPU and VRAM, this task was only assigned to GPU-enabled nodes 223.2.45.222 and 223.2.41.213, further demonstrating the AI strategy's advantage in resource-task matching.

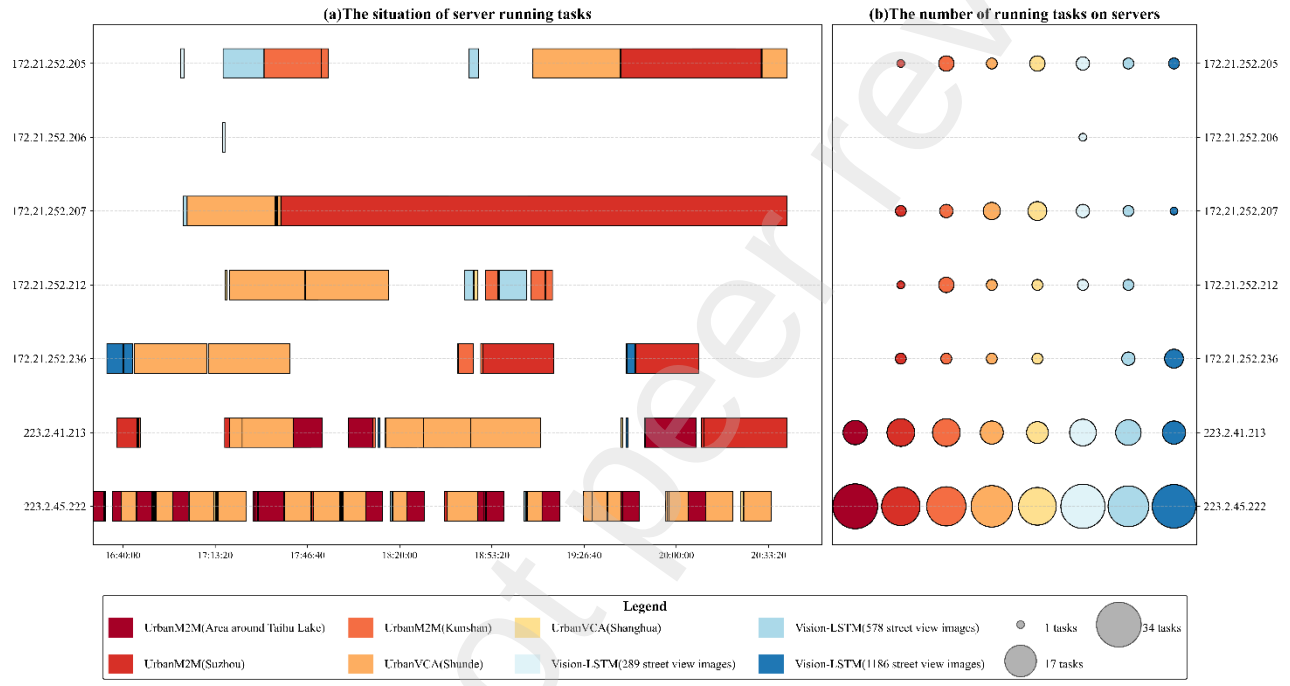

Figure 12 Task allocation within 4 hours under the AI dynamic scheduling strategy (a. Task execution on servers, b. Task counts on servers).

(3) Comparative Analysis of AI Dynamic Scheduling and Current Scheduling

Table 6 summarizes the task handling results of the two scheduling strategies within the 4-hour fixed runtime window, validating the advantages of the AI dynamic scheduling strategy in improving task throughput and system stability.

Table 6 Comparison of task states within 4 hours under AI dynamic scheduling and current scheduling strategies.

Scheduling policy	Number of successful tasks	Number of failed tasks	Success rate	Number of pending tasks	Number of running tasks	Total number of tasks processed
Current scheduling strategy	301	58	83.84%	112	7	478

AI dynamic scheduling strategy	404	16	96.18%	39	10	469

Under the current static scheduling strategy, a total of 478 tasks were processed, of which 58 failed, resulting in a success rate of 83.84%. The large number of pending tasks indicates that the system's processing capacity was heavily constrained by inefficient resource allocation. By contrast, within the same 4-hour period, the AI scheduling strategy successfully completed 404 tasks, achieving approximately a 34.22% increase in successful throughput compared to the static strategy. The number of failed tasks was reduced to 16, raising the success rate to 96.18%. This improvement in stability further demonstrates the AI scheduling strategy's effectiveness in anticipating task resource requirements, thereby avoiding memory overflows or timeout failures caused by assigning compute-intensive geo-simulation tasks to low-performance nodes. In addition, the number of pending tasks dropped to 39 under the AI scheduling strategy, proving that it enabled faster resource turnover within the cluster and effectively eliminated bottlenecks. By concentrating tasks on high-performance compute nodes, tasks could be processed quickly, significantly shortening their lifecycle and ensuring a healthy scheduling queue.

Overall, this experiment provides strong evidence of the AI-driven dynamic scheduling framework's superiority as a long-term, high-concurrency scheduling strategy. By setting task priority and capacity aware dynamic resources allocation, it significantly enhances the effective throughput, resource utilization, and overall operational reliability of heterogeneous compute clusters.

6. Conclusion

This study focuses on the problem of AI-driven scheduling for AI-driven geo-simulation tasks and proposes an intelligent scheduling framework for distributed heterogeneous clusters. Compared with the current static scheduling strategy, the framework incorporates Gemini as an external LLM for decision-making. By integrating runtime information awareness, a historical task knowledge base, and real-time computational resource data, it achieves dynamic, data-driven scheduling optimization. Through extended applications within the three-layer OpenGMS architecture, the study validates the effectiveness of this scheduling approach across multiple types of AI-driven geo-simulation tasks. Experimental results show that AI-based dynamic scheduling can accurately match computational resources to the requirements of AI-driven geo-simulation tasks, avoid load imbalance, significantly reduce overall runtime, mitigate the long-tail effect common in static scheduling, while also improving task success rates to some extent. These findings highlight the applicability and superiority of the proposed method in complex heterogeneous environments.

However, several limitations and challenges remain. An important limitation lies in the reliance of the current scheduling framework on the reasoning capabilities of the LLM to select computational resources. Although this approach often achieves favorable results, issues such as resource allocation bias and increased task failures may still occur in scenarios involving frequent resource contention and high task concurrency. Second, the framework's considerations of fault tolerance and resource redundancy are not yet sufficient. For example, sudden request capacity limits or network fluctuations affecting the LLM could disrupt node evaluation and lead to task creation failures. Meanwhile, low-performance nodes remain underutilized, causing resource idleness. Consequently, the framework's robustness under sudden heterogeneous resource failures or unstable network conditions requires further enhancement. Additionally, although the experimental design already covers several geo-

simulation models, its adaptability to larger-scale, cross-platform, and multi-environment scenarios remains to be further validated.

Future work may proceed in several directions. First, introducing fault-tolerance and elastic scaling mechanisms would allow failed tasks to be reassigned to alternative nodes, thereby improving robustness in complex and uncertain computational environments. Second, incorporating techniques such as reinforcement learning and adaptive optimization could enable the scheduling strategy to evolve continuously and self-adjust in dynamic environments. Overall, the AI-driven intelligent scheduling framework proposed in this study not only provides a new perspective for improving geo-simulation efficiency and resource utilization but also lays a foundation for building future-oriented intelligent scientific computing platforms.

CRediT authorship contribution statement

Wanhao Li: Writing – original draft, Validation, Resources, Methodology. Min Chen: Writing – review & editing, Validation, Funding acquisition, Conceptualization. Fengyuan Zhang: Writing – review & editing, Supervision, Funding acquisition. Peilong Ma: Writing – review & editing, Supervision, Methodology. Zaiyang Ma: Writing – review & editing, Methodology. Yongning Wen: Writing – review & editing, Methodology. Songshan Yue: Writing – review & editing, Methodology. Guonian Lv: Writing – review & editing, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We appreciate the detailed suggestions and comments from the anonymous reviewers. We express heartfelt thanks to the other members of the OpenGMS team. This work was supported by the NSF of China [grant numbers 42301539, 42401574, 42325107, 42361144884].

Data availability

Data will be made available on request.

References

Bao, Y., Lin, X., Jia, M., Xiao, Z., Wang, C., Liao, J., & Zhang, Y. J. I. J. o. D. E. (2025). Convergence in key month of phenology-based mangrove species classification using sentinel-2 imagery data: insights from structural and physiological indices. *18*(1), 2528651.

Belete, G. F., Voinov, A., Morales, J. J. E. M., & Software. (2017). Designing the distributed model integration framework–DMIF. *94*, 112-126.

Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., & Fu, Z. J. a. p. a. (2024). Deepseek llm: Scaling open-source language models with longtermism.

Buschjager, S., Chen, K.-H., Chen, J.-J., & Morik, K. (2018). *Realization of random forest for real-time evaluation through tree framing.* Paper presented at the 2018 IEEE International Conference on Data Mining (ICDM).

Cao, K., Zhou, C., Church, R., Li, X., & Li, W. (2024). Revisiting spatial optimization in the era of geospatial big data and GeoAI. *International Journal of Applied Earth Observation and*

Geoinformation, 129. doi:10.1016/j.jag.2024.103832

Chaturvedi, V., & de Vries, W. T. (2021). Machine Learning Algorithms for Urban Land Use Planning: A Review. *5*(3), 68.

Chen, M., Claramunt, C., Çöltekin, A., Liu, X., Peng, P., Robinson, A. C., Wang, D., Strobl, J., Wilson, J. P., Batty, M., Kwan, M.-P., Lotfian, M., Golay, F., Joost, S., Ingensand, J., Senousi, A. M., Cheng, T., Bandrova, T., Konecny, M., Torrens, P. M., Klippel, A., Li, S., Zhang, F., He, L., Wang, J., Ratti, C., Kolditz, O., Lin, H., & Lü, G. (2023). Artificial intelligence and visual analytics in geographical space and cyberspace: Research opportunities and challenges. *Earth-Science Reviews, 241*, 104438. doi:https://doi.org/10.1016/j.earscirev.2023.104438

Chen, M., Qian, Z., Boers, N., Jakeman, A. J., Kettner, A. J., Brandt, M., Kwan, M.-P., Batty, M., Li, W., Zhu, R., Luo, W., Ames, D. P., Barton, C. M., Cuddy, S. M., Koirala, S., Zhang, F., Ratti, C., Liu, J., Zhong, T., Liu, J., Wen, Y., Yue, S., Zhu, Z., Zhang, Z., Sun, Z., Lin, J., Ma, Z., He, Y., Xu, K., Zhang, C., Lin, H., & Lü, G. (2023). Iterative integration of deep learning in hybrid Earth surface system modelling. *Nature Reviews Earth & Environment, 4*(8), 568-581. doi:10.1038/s43017-023-00452-7

Chen, M., Voinov, A., Ames, D. P., Kettner, A. J., Goodall, J. L., Jakeman, A. J., Barton, M. C., Harpham, Q., Cuddy, S. M., DeLuca, C., Yue, S., Wang, J., Zhang, F., Wen, Y., & Lü, G. (2020). Position paper: Open web-distributed integrated geographic modelling and simulation to enable broader participation and applications. *Earth-Science Reviews, 207*. doi:10.1016/j.earscirev.2020.103223

Dafir, Z., Lamari, Y., & Slaoui, S. C. (2020). A survey on parallel clustering algorithms for Big Data. *Artificial Intelligence Review, 54*(4), 2411-2443. doi:10.1007/s10462-020-09918-2

Dai, M., Hu, J., Zhuang, J., & Zheng, E. (2022). A Transformer-Based Feature Segmentation and Region Alignment Method for UAV-View Geo-Localization. *IEEE Transactions on Circuits and Systems for Video Technology, 32*(7), 4376-4389. doi:10.1109/tcsvt.2021.3135013

David, O., Ascough, J. C., Lloyd, W., Green, T. R., Rojas, K. W., Leavesley, G. H., & Ahuja, L. R. (2013). A software engineering perspective on environmental modeling framework design: The Object Modeling System. *Environmental Modelling & Software, 39*, 201-213. doi:10.1016/j.envsoft.2012.03.006

Deng, J., Desjardins, M. R., & Delmelle, E. M. (2019). An interactive platform for the analysis of landscape patterns: a cloud-based parallel approach. *Annals of GIS, 25*(2), 99-111. doi:10.1080/19475683.2019.1615550

Derry, A., Krzywinski, M., & Altman, N. (2023). Convolutional neural networks. In: Nature Publishing Group US New York.

Duan, S., Wang, D., Ren, J., Lyu, F., Zhang, Y., Wu, H., & Shen, X. (2023). Distributed Artificial Intelligence Empowered by End-Edge-Cloud Computing: A Survey. *IEEE Communications Surveys & Tutorials, 25*(1), 591-624. doi:10.1109/comst.2022.3218527

Georganos, S., Grippa, T., Niang Gadiaga, A., Linard, C., Lennert, M., Vanhuysse, S., Mboga, N., Wolff, E., & Kalogirou, S. J. G. I. (2021). Geographical random forests: a spatial extension of the random forest algorithm to address spatial heterogeneity in remote sensing and population modelling. *36*(2), 121-136.

Hu, Y., Gao, S., Lunga, D., Li, W., Newsam, S., & Bhaduri, B. (2019). GeoAI at ACM SIGSPATIAL. *SIGSPATIAL Special, 11*(2), 5-15. doi:10.1145/3377000.3377002

Hu, Y., Goodchild, M., Zhu, A. X., Yuan, M., Aydin, O., Bhaduri, B., Gao, S., Li, W., Lunga, D., &

Newsam, S. (2024). A five-year milestone: reflections on advances and limitations in GeoAI research. *Annals of GIS, 30*(1), 1-14. doi:10.1080/19475683.2024.2309866

Huang, Y., Zhang, F., Gao, Y., Tu, W., Duarte, F., Ratti, C., Guo, D., & Liu, Y. (2023). Comprehensive urban space representation with varying numbers of street-level images. *Computers, Environment and Urban Systems, 106*. doi:10.1016/j.compenvurbsys.2023.102043

Karimi, F., & Sultana, S. J. S. (2024). Urban expansion prediction and land use/land cover change modeling for sustainable urban development. In (Vol. 16, pp. 2285): MDPI.

Kim, I.-H., Tsou, M.-H., & Feng, C.-C. (2015). Design and implementation strategy of a parallel agent-based Schelling model. *Computers, Environment and Urban Systems, 49*, 30-41. doi:10.1016/j.compenvurbsys.2014.09.004

Kuglitsch, M. M., Pelivan, I., Ceola, S., Menon, M., & Xoplaki, E. J. N. c. (2022). Facilitating adoption of AI in natural disaster management through collaboration. *13*(1), 1579.

Li, C., Liu, J., Li, W., & Luo, Y. (2021). Adaptive priority-based data placement and multi-task scheduling in geo-distributed cloud systems. *Knowledge-Based Systems, 224*. doi:10.1016/j.knosys.2021.107050

Li, Z., Ning, H., Gao, S., Janowicz, K., Li, W., Arundel, S. T., Yang, C., Bhaduri, B., Wang, S., Zhu, A. X., Gahegan, M., Shekhar, S., Ye, X., McKenzie, G., Cervone, G., & Hodgson, M. E. (2025). GIScience in the era of Artificial Intelligence: a research agenda towards Autonomous GIS. *Annals of GIS*, 1-36. doi:10.1080/19475683.2025.2552161

Liang, J., Hou, S., Zhao, A., Xu, Q., Xiang, L., Li, R., & Wu, H. (2025). Design and application of a semantic-driven geospatial modeling knowledge graph based on large language models. *Geo-spatial Information Science*, 1-20. doi:10.1080/10095020.2025.2483884

Liang, J., Zhao, A., Hou, S., Jin, F., & Wu, H. (2024). A GPT-enhanced framework on knowledge extraction and reuse for geographic analysis models in Google Earth Engine. *International Journal of Digital Earth, 17*(1). doi:10.1080/17538947.2024.2398063

Lin, H., Chen, M., & Lu, G. (2012). Virtual Geographic Environment: A Workspace for Computer-Aided Geographic Experiments. *Annals of the Association of American Geographers, 103*(3), 465-482. doi:10.1080/00045608.2012.689234

Liu, J., Zhu, A. X., Qin, C.-Z., Wu, H., & Jiang, J. (2016). A two-level parallelization method for distributed hydrological models. *Environmental Modelling & Software, 80*, 175-184. doi:10.1016/j.envsoft.2016.02.032

Liu, Y., Wang, X., Wang, Y., Huang, F., Huang, Y., Li, Y., Zhang, W., Gong, S., Mai, G., Yao, Y., Yue, Y., Li, H., & Zhang, F. (2025). Representation learning for geospatial data. *Annals of GIS*, 1-27. doi:10.1080/19475683.2025.2552157

Lu, F., Cheng, S., & Wang, P. (2025). GeoAI enabled urban computing: status and challenges. *Annals of GIS*, 1-19. doi:10.1080/19475683.2025.2552152

Lü, G., Xiong, J., Wu, M., Yuan, L., Li, J., Chen, M., Yu, Z., Zhou, L., Yue, S., Zhang, X., Li, X., Li, X., Chen, F., Zhou, C., Zhang, Z., & Gao, Y. (2025). Geographic information discipline development: Demands, Strategy and challenges. *Information Geography, 1*(1), 100012. doi:https://doi.org/10.1016/j.infgeo.2025.100012

Ma, P., Chen, M., Zhang, S., Zhu, Z., Qian, Z., Ma, Z., Zhang, F., Li, W., Yue, S., & Wen, Y. (2025). Facilitating sensitivity analysis of hydrological models through knowledge-driven configuration and distributed online model services. *Journal of Hydrology, 660*, 133406.

doi:https://doi.org/10.1016/j.jhydrol.2025.133406

Ma, Z., Wu, C., Chen, M., Li, H., Lin, J., Zheng, Z., Yue, S., Wen, Y., & Lü, G. (2024). Promoting forest landscape dynamic prediction with an online collaborative strategy. *Journal of Environmental Management, 352*, 120083. doi:https://doi.org/10.1016/j.jenvman.2024.120083

Peckham, S. D., Hutton, E. W. H., & Norris, B. (2013). A component-based approach to integrated modeling in the geosciences: The design of CSDMS. *Computers & Geosciences, 53*, 3-12. doi:10.1016/j.cageo.2012.04.002

Qin, C.-Z., Zhu, L.-J., & Zhu, A. X. (2025). Position paper: Domain knowledge-driven intelligentization of watershed modeling and scenario analysis for precise watershed management. *Information Geography, 1*(1), 100016. doi:https://doi.org/10.1016/j.infgeo.2025.100016

Reddy, G. O. (2018). Spatial data management, analysis, and modeling in GIS: principles and applications. In *Geospatial Technologies in Land Resources Mapping, Monitoring and Management* (pp. 127-142): Springer.

Shi, M., Janowicz, K., Verstegen, J., Currier, K., Wiedemann, N., Mai, G., Majic, I., Liu, Z., Zhu, R. J. C., & Science, G. I. (2025). Geography for AI sustainability and sustainability for GeoAI. *52*(4), 331-349.

Sun, A. Y., & Scanlon, B. R. (2019). How can Big Data and machine learning benefit environment and water management: a survey of methods, applications, and future directions. *Environmental Research Letters, 14*(7). doi:10.1088/1748-9326/ab1b7d

Sun, J., Xu, C., Tang, L., Wang, S., Lin, C., Gong, Y., Ni, L. M., Shum, H.-Y., & Guo, J. J. a. p. a. (2023). Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph.

Tarboton, D. G., Ames, D. P., Horsburgh, J. S., Goodall, J. L., Couch, A., Hooper, R., Bales, J., Wang, S., Castronova, A., Seul, M., Idaszak, R., Li, Z., Dash, P., Black, S., Ramirez, M., Yi, H., Calloway, C., & Cogswell, C. (2024). HydroShare retrospective: Science and technology advances of a comprehensive data and model publication environment for the water science domain. *Environmental Modelling & Software, 172*. doi:10.1016/j.envsoft.2023.105902

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., & Deac, A. J. N. (2023). Scientific discovery in the age of artificial intelligence. *620*(7972), 47-60.

Wang, H., Fu, X., Wang, G., Li, T., & Gao, J. (2011). A common parallel computing framework for modeling hydrological processes of river basins. *Parallel Computing, 37*(6-7), 302-315. doi:10.1016/j.parco.2011.05.003

Wang, J., Chen, M., Lü, G., Yue, S., Chen, K., & Wen, Y. (2018). A Study on Data Processing Services for the Operation of Geo‐Analysis Models in the Open Web Environment. *Earth and Space Science, 5*(12), 844-862. doi:10.1029/2018ea000459

Wen, Y., Chen, M., Yue, S., Zheng, P., Peng, G., & Lu, G. (2016). A model-service deployment strategy for collaboratively sharing geo-analysis models in an open web environment. *International Journal of Digital Earth, 10*(4), 405-425. doi:10.1080/17538947.2015.1131340

Westen, S. J. P., Gijsbers, P. J. A., & Gregersen, J. B. (2007). OpenMI: Open modelling interface. *Journal of Hydroinformatics, 9*(3), 175-191. doi:10.2166/hydro.2007.023

Xu, K., Chen, M., Yue, S., Zhang, F., Wang, J., Wen, Y., & Lü, G. (2024). The portal of OpenGMS: Bridging the contributors and users of geographic simulation resources. *Environmental*

Modelling & Software, 180. doi:10.1016/j.envsoft.2024.106142

Yang, C., & Raskin, R. J. I. J. o. G. I. S. (2009). Introduction to distributed geographic information processing research. In (Vol. 23, pp. 553-560): Taylor & Francis.

Yao, Y., Li, L., Liang, Z., Cheng, T., Sun, Z., Luo, P., Guan, Q., Zhai, Y., Kou, S., & Cai, Y. J. a. p. a. (2021). UrbanVCA: A vector-based cellular automata framework to simulate the urban land-use change at the land-parcel level.

Yue, S., Chen, M., Wen, Y., & Lu, G. (2016). Service-oriented model-encapsulation strategy for sharing and integrating heterogeneous geo-analysis models in an open web environment. *ISPRS Journal of Photogrammetry and Remote Sensing, 114*, 258-273. doi:10.1016/j.isprsjprs.2015.11.002

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). *Spark: Cluster computing with working sets.* Paper presented at the 2nd USENIX workshop on hot topics in cloud computing (HotCloud 10).

Zhang, F., Chen, M., Yue, S., Wen, Y., Lü, G., & Li, F. (2020). Service-oriented interface design for open distributed environmental simulations. *Environmental Research, 191*, 110225. doi:https://doi.org/10.1016/j.envres.2020.110225

Zhang, Y., Li, Y., & Zhang, F. (2024). Multi-level urban street representation with street-view imagery and hybrid semantic graph. *ISPRS Journal of Photogrammetry and Remote Sensing, 218*, 19-32. doi:10.1016/j.isprsjprs.2024.09.032

Zhao, T., Wang, S., Ouyang, C., Chen, M., Liu, C., Zhang, J., Yu, L., Wang, F., Xie, Y., Li, J., Wang, F., Grunwald, S., Wong, B. M., Zhang, F., Qian, Z., Xu, Y., Yu, C., Han, W., Sun, T., Shao, Z., Qian, T., Chen, Z., Zeng, J., Zhang, H., Letu, H., Zhang, B., Wang, L., Luo, L., Shi, C., Su, H., Zhang, H., Yin, S., Huang, N., Zhao, W., Li, N., Zheng, C., Zhou, Y., Huang, C., Feng, D., Xu, Q., Wu, Y., Hong, D., Wang, Z., Lin, Y., Zhang, T., Kumar, P., Plaza, A., Chanussot, J., Zhang, J., Shi, J., & Wang, L. (2024). Artificial intelligence for geoscience: Progress, challenges, and perspectives. *Innovation (Camb), 5*(5), 100691. doi:10.1016/j.xinn.2024.100691

Zhou, Z., Chen, Y., Liu, X., Zhang, X., & Zhang, H. (2024). A maps-to-maps approach for simulating urban land expansion based on convolutional long short-term memory neural networks. *International Journal of Geographical Information Science, 38*(3), 503-526. doi:10.1080/13658816.2023.2298296

Zhou, Z., Ning, X., Hong, K., Fu, T., Xu, J., Li, S., Lou, Y., Wang, L., Yuan, Z., & Li, X. J. a. p. a. (2024). A survey on efficient inference for large language models.

Server IP	System type	CPU model	core
172.21.252.205	Windows_NT	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	16
172.21.252.206	Windows_NT	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	16
172.21.252.207	Windows_NT	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	4
172.21.252.212	Windows_NT	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	8
172.21.252.236	Linux	Intel Xeon E5-2620 v4	32
223.2.41.213	Windows_NT	AMD Ryzen 5 4600H with Radeon Graphics	12
223.2.45.222	Windows_NT	AMD Ryzen 9 9950X 16-Core Processor	32

Total memory	Total disk	GPU model	Total vram
4G	200G	/	/
4G	500G	/	/
8G	400G	/	/
4G	452G	/	/
6G	1510789G	/	/
16G	185G	NVIDIA GeForce GTX 1650	4096
32G	778G	NVIDIA GeForce RTX 5090	32607

Model	Type
Vision-LSTM	SimpleCalculation
UrabnVCA	TimeSeries
UrbanM2M	StateSimulation

Description
A model for image classification that extracts features from images using a pre trained ResNet18 model, and then classifies the images using an LSTM mode.
A simulation and prediction model for urban land use change based on real land parcels and vector cellular automata, which realizes vector dynamic land parcel classification, land use data matching, overall development probability calculation, and simulation functions.
A deep learning model based on ConvLSTM uses time-series urban land grid data and spatial variables to simulate future urban expansion scenarios.

Core requirements	Research scope
CPU、GPU	289 street view images
CPU、GPU	578 street view images
CPU、GPU、network	1186 street view images
CPU、memory、network	Shanghua
CPU、memory、network	Shunde
CPU、GPU、memory、vram	Kunshan
CPU、GPU、memory、vram、network	Suzhou
CPU、GPU、memory、vram、network、disk	Area around Taihu Lake

Model	UrbanM2M model			UrbanVCA model		Vision-LSTM model		
Scope	Kunshan	Suzhou	Area around Taihu Lake	Shanghua	Shunde	289 street view images	578 street view images	1186 street view images
Computing node	172.21.252.212	172.21.252.212	172.21.252.207	172.21.252.236	172.21.252.206	172.21.252.206	172.21.252.207	172.21.252.206
Transmission time(s)	12.57	54.57	264.75	26.95	65.95	42.86	34.54	37.15
Run time(s)	129.08	896.68	4843.32	32.29	1578.72	41.31	62.81	952.86
Total time(s)	141.65	951.25	5108.07	59.24	1644.67	84.17	97.35	990.01

Model	UrbanM2M model			UrbanVCA model		Vision-LSTM model		
Scope	Kunshan	Suzhou	Area around Taihu Lake	Shanghua	Shunde	289 street view images	578 street view images	1186 street view images
Computing node	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222	223.2.45.222
Transmission time(s)	19.93	56.66	110.19	28.56	61.29	14.48	24.03	24.86
Run time(s)	8.33	35.41	104.79	12.61	573.87	6.36	10.39	14.87
Total time(s)	28.26	82.07	214.98	41.17	635.16	20.84	34.42	39.73

Geo-simulation task	Characteristics
Vision-LSTM model(289 street images)	CPU、GPU
UrabnVCA model(shanghua)	CPU、memory、network
UrabnM2M model(kunshan)	CPU、GPU、memory、VRAM

Scheduling policy	Number of successful tasks	Number of failed tasks
Current scheduling strategy	397	3
AI dynamic scheduling strategy	397	3
Current scheduling strategy	387	13
AI dynamic scheduling strategy	399	1
Current scheduling strategy	391	9
AI dynamic scheduling strategy	370	30

Total runtime
8414.81s
2648.00s
4788.54s
2123.40s
6316.83s
3934.37s

Scheduling policy	Number of successful tasks	Number of failed tasks	Success rate
Current scheduling strategy	301	58	83.84%
AI dynamic scheduling strategy	404	16	96.18%

Number of pending tasks	Number of running tasks	Total number of tasks processed
112	7	478
39	10	469