Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/119/2025.21640

Research on Grid Heterogeneous Resource Optimization Scheduling Based on Linear Programming

Weihan Li[1,a,*]

[1]*College of Science, University of Shanghai for Science and Technology, No. 516, Jungong Road, Shanghai, China*
a. 15608725818@163.com
**corresponding author*

Abstract: Grid computing serves as a critical infrastructure for managing large-scale data and complex computational tasks across diverse domains like scientific research and engineering. Efficient resource allocation in heterogeneous grid environments, however, poses a significant challenge due to the inherent diversity and dynamism of available resources. To address this, this paper introduces a resource optimization and scheduling method based on linear programming, specifically designed for heterogeneous grid settings. This approach focuses on enhancing resource utilization efficiency and minimizing task execution times by first analyzing critical factors such as the heterogeneity of computational resources, task priorities, and network bandwidth and latency. A linear programming model is then constructed to comprehensively integrate load balancing and priority-based scheduling. By judiciously defining the objective function and constraints, this model facilitates the optimal configuration of heterogeneous resources. As a well-established optimization technique, linear programming provides a novel perspective for grid resource scheduling. Ultimately, this research offers valuable theoretical analysis and practical insights for understanding and resolving resource management and optimization issues in heterogeneous grid environments, while also laying a solid foundation for future exploration of intelligent scheduling strategies based on machine learning and providing researchers with a viable modeling framework and optimization approach for advancing grid computing technologies.

Keywords: Grid computing, resource allocation, linear programming, load balancing

1. Introduction

With the rapid development of technology, distributed computing and large-scale data processing have gradually become important means of solving complex problems. As a typical distributed computing model, grid computing integrates geographically dispersed computing and data resources, providing powerful computing capabilities for scientific research, engineering calculations, and big data processing. However, grid computing faces the significant challenge of how to allocate resources efficiently. Due to the heterogeneity of computing resources, the complexity and dynamism of tasks, ensuring the efficient execution of computational tasks and avoiding resource waste have become key issues affecting the performance of grid computing systems.

In a grid computing environment, the rational allocation of computing resources directly determines the system's performance and computational efficiency. Factors such as the heterogeneity

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/119/2025.21640

of computing resources, task priorities, network bandwidth, and latency can all affect the outcome of resource allocation. Therefore, resource allocation is not merely a technical problem but also involves how to perform precise scheduling and optimization in a diversified resource environment. In particular, under high load and concurrent tasks, ensuring system stability and maximizing resource utilization through reasonable scheduling is one of the core issues in grid computing research.

Resource allocation in grid computing must comprehensively consider multiple factors, including but not limited to: the type and performance differences of computing resources, task priorities, network bandwidth for data transmission, and system load balancing. The balance of these factors directly determines the computational efficiency, task execution time, and overall stability of the grid system.

2. Resource Allocation Issues in Grid Computing

Rational resource allocation in grid computing can not only enhance the overall system performance but also ensure the efficient execution of tasks and the maximization of system resource utilization. Since the resource environment in grid computing is typically highly decentralized and heterogeneous, effective resource scheduling is crucial for optimizing system performance. For example, on the Grid5000 platform, the scheduling and allocation of computing resources is based on a priority-based scheduling algorithm [1]. Research has shown that after adopting this algorithm, the system's computing resource utilization increased by 15%, and task completion time was reduced by approximately 20% [2]. Furthermore, through a load balancing mechanism, the system effectively avoids overloading individual nodes, ensuring smooth task execution.

In the LIGO (Laser Interferometer Gravitational-Wave Observatory) project, large-scale data processing is achieved through grid computing at multiple global experimental sites. The project uses a resource scheduling strategy based on task priority and dependency. With this approach, the LIGO system ensures that high-priority tasks (such as real-time gravitational wave signal analysis) are prioritized for access to computing resources, ensuring the timely processing of real-time data and significantly improving the system’s processing efficiency [3]. These cases clearly demonstrate that rational resource allocation not only improves the efficiency of computing resource utilization but also ensures the on-time completion of tasks, enhancing the system's reliability and stability. Several key factors must be considered for computation:

2.1. Resource Heterogeneity

The resource environment in grid computing typically consists of multiple heterogeneous nodes, and the allocation and scheduling of resources must consider a variety of factors. The hardware and software environments of computing nodes are often different, manifesting in differences in computing power, memory, storage capacity, and other aspects. This heterogeneity poses a significant challenge for resource scheduling. The goal of heterogeneous resource scheduling is to execute each task on the most suitable computing node to maximize the system's overall performance. For example, on the EUGrid platform, a scheduling algorithm based on resource characteristics and task requirements is used, which can allocate tasks to the most appropriate resource nodes based on the performance characteristics of the computing nodes. This heterogeneous resource scheduling increased system computing efficiency by about 25%. According to Bal et al. [4], heterogeneous resource management can improve the utilization of computing resources through load prediction and task matching. Certain computation-intensive tasks (such as big data analysis) are better suited for execution on high-performance computing nodes (such as GPUs or high-frequency CPUs), while storage-intensive tasks should be allocated to nodes with larger storage capacities.

DOI: 10.54254/2755-2721/119/2025.21640

2.2. Load Balancing

Load balancing refers to how tasks are reasonably distributed among multiple computing nodes to prevent some nodes from being overloaded, which would degrade performance, while ensuring that all nodes' resources are fully utilized. The goal of load balancing is to optimize computational efficiency, reduce resource idleness, and avoid bottlenecks during task execution. On Amazon's Elastic Compute Cloud (EC2), an intelligent load balancing system dynamically distributes computational tasks. The EC2 platform uses a scheduling mechanism based on real-time traffic and system load, which allocates tasks reasonably across multiple computing nodes, thus enhancing the stability of the service. According to Amazon's data, the load balancing mechanism reduces the system's response time by more than 40%, while maintaining smooth system operation under high load. In certain high-concurrency grid computing environments, load balancing technology has reduced task completion times by 15% to 30%, significantly reducing resource waste [5].

2.3. Network Bandwidth and Latency

In distributed computing, the communication network bandwidth and latency between nodes are important factors that impact system performance. Network bandwidth limitations and latency issues may restrict the data transfer speed during task scheduling, affecting overall computational efficiency. Therefore, network bandwidth and latency must be considered when allocating resources. In systems using the GASNet (Global Address Space Networking) protocol, optimizing network bandwidth and latency improved data transmission efficiency between tasks by 25%, greatly reducing the time required for data transfer and thus enhancing overall computational efficiency. According to relevant studies [6], task data transmission time accounts for approximately 40% of the total execution time. If network bandwidth and latency are not considered, the system's overall performance may decrease by more than 25%. Therefore, network performance must be considered as a key factor in resource allocation.

2.4. Task Priorities

The priority of a task determines its scheduling order when resources are limited. High-priority tasks usually have a higher level of urgency or importance and need to access computing resources first. Proper allocation of priorities can effectively improve system response speed and ensure that critical tasks are completed on time. For example, in some high-performance computing applications (such as meteorological forecasting and genome analysis), task priorities directly affect the system's performance. By setting dynamic priorities and task dependencies, the system can schedule tasks based on their urgency, ensuring that high-priority tasks are completed in the shortest possible time. For example, in meteorological forecasting, disaster warning tasks generally have high priority to ensure real-time processing and prediction. After adopting priority scheduling, task completion time was reduced by an average of 20% to 40%. Meanwhile, the processing time for high-priority tasks was also shortened by 15% to 30% [7].

3. Optimization Algorithms for Resource Allocation in Grid Computing

To achieve efficient resource allocation, grid computing systems employ various optimization algorithms. The core objective of these algorithms is to maximize resource utilization, minimize task execution time, and ensure system load balancing. Optimization algorithms in grid computing include task dependency-based scheduling algorithms [8], dynamic load balancing algorithms [5], and prediction-based resource allocation algorithms [7]. After applying these optimization algorithms, the system's computing resource utilization has improved by 15% to 30%.

DOI: 10.54254/2755-2721/119/2025.21640

3.1. Task Scheduling Optimization Algorithms

Task scheduling optimization algorithms play a crucial role in grid computing [8]. The Shortest Job First (SJF) algorithm schedules tasks by prioritizing those with the shortest execution time, which reduces the average waiting time and improves the system's response speed. Based on the urgency and importance of tasks, the priority scheduling algorithm ensures that high-priority tasks are processed in a timely manner. By categorizing and sorting tasks, the system can allocate computing resources reasonably, ensuring that important tasks are completed first. Fair scheduling, through distributing computing resources among multiple tasks, guarantees each task receives a relatively fair share of resources and prevents any one task from monopolizing resources, thereby enhancing overall system performance.

3.2. Load Balancing Optimization Algorithms

Load balancing optimization algorithms dynamically adjust task allocation to prevent overload on computing nodes and improve system efficiency, especially in heterogeneous grid computing environments [5, 9]. These algorithms focus on achieving an even distribution of tasks based on the computing power, real-time load, and network conditions of nodes, which enhances resource utilization and computational efficiency. The weighted round-robin algorithm assigns tasks based on the processing capabilities of nodes, ensuring that high-performance nodes handle a greater load, making it suitable for heterogeneous environments. In the Grid5000 platform application, this method improved system load balancing by 20% and reduced task execution time by 15%. However, when the resource load is uniform, the hash load balancing algorithm can distribute tasks across computing nodes using simple mapping, though it may cause load imbalance if there is significant variation in node performance. To address this, an improved hash algorithm incorporates real-time node status information to make task allocation more accurate. Google's data centers, by optimizing the hash algorithm, reduced task response times by more than 10% and enhanced computing resource utilization [5].

In contrast, adaptive load balancing algorithms can dynamically adjust task distribution according to real-time fluctuations in system load, making them more flexible, particularly in environments with significant load variations. In the Xen virtualization environment, adaptive load balancing effectively avoided resource overload or idleness by dynamically allocating tasks, increasing the system's CPU load balancing rate by 25% and improving response speed by 18%. In summary, while weighted round-robin, hash, and adaptive load balancing algorithms each have their advantages, their effectiveness in practical applications is closely related to the specific environment. Future approaches that combine multiple algorithms and leverage intelligent scheduling methods could provide more efficient and flexible load balancing solutions for grid computing.

3.3. Dynamic Resource Allocation and Scheduling

Dynamic resource allocation and scheduling algorithms are designed to address fluctuations in resource demand and changes in system load, ensuring that resources are optimally utilized in environments with high load and frequent task changes [5, 7]. Unlike static allocation methods, dynamic scheduling monitors system status in real-time, predicts load changes, and adjusts resource allocation strategies based on demand, thus enhancing system flexibility and adaptability. Common dynamic resource scheduling algorithms include load-based scheduling, priority-based scheduling, and hybrid scheduling strategies.

Load-based scheduling methods dynamically adjust task allocation by monitoring real-time node load conditions, ensuring that resources are always in optimal use. For example, in some cloud

DOI: 10.54254/2755-2721/119/2025.21640

computing platforms, dynamic scheduling algorithms can automatically adjust tasks based on CPU and memory load conditions of nodes to prevent overload or resource idleness. Priority-based scheduling considers the urgency and importance of tasks, ensuring that high-priority tasks are allocated resources first. This method is widely used in real-time data processing in large-scale scientific computing to ensure timely task completion. Hybrid scheduling strategies combine load monitoring with priority scheduling, enabling flexible adjustment of resource allocation strategies in complex environments and improving overall system efficiency and stability.

In practice, the effectiveness of dynamic resource allocation and scheduling strategies usually depends on the accuracy of load prediction and the real-time responsiveness of the scheduling algorithm. In recent years, dynamic scheduling methods that combine machine learning and other intelligent algorithms have become a research focus. By intelligently predicting system load fluctuations, dynamic scheduling not only increases resource utilization but also maintains system balance and stability amid changes in task demands.

4. Application of Optimization Algorithms in Grid Computing

System optimization is becoming increasingly important in large-scale grid computing, especially in environments with heterogeneous resources and dynamic load changes. The key challenge is how to improve resource utilization and system efficiency through optimized scheduling strategies, which has become a hot topic in current research. Optimization strategies in grid computing cover multiple aspects, including task scheduling, resource allocation, load balancing, and energy efficiency optimization. In response to these challenges, researchers have proposed various optimization models based on advanced theories and methods, achieving significant application results in various fields. This paper explores the related theories of system optimization by combining the latest research progress and, through case studies, demonstrates the effects and advantages of these optimization methods in practical applications.

4.1. Task Scheduling Optimization

Task scheduling is one of the most fundamental and critical optimization issues in grid computing systems [8]. As the scale of tasks continues to grow, how to efficiently schedule computational tasks to maximize resource utilization has become a key factor in improving system performance. Common task scheduling optimization methods primarily include heuristic algorithms [9, 10], game theory, linear programming, and other scheduling strategies.

Among these, heuristic algorithm-based scheduling methods are widely used in grid computing. Heuristic algorithms design task prioritization and task dependencies to achieve more efficient resource scheduling. For instance, a task scheduling method based on genetic algorithms simulates natural selection mechanisms to find optimal solutions within a limited search space. Research shows that using genetic algorithm-based scheduling strategies can effectively increase system resource utilization and reduce task execution time. In the "Grid5000" platform application, optimizing task scheduling strategies reduced task completion time by about 18%, while improving computing resource utilization by 25% [1, 2]. Moreover, game theory models are also widely applied to resource competition and task scheduling problems. By simulating the game process between multiple tasks, these models optimize the resource allocation for each task to achieve the overall optimization goal. Studies from both domestic and international researchers show that game theory models not only optimize system scheduling but also ensure system stability and robustness under dynamic load conditions.

DOI: 10.54254/2755-2721/119/2025.21640

4.2. Resource Allocation and Load Balancing

Resource allocation and load balancing are another critical optimization issue in grid computing systems. In environments with heterogeneous resources and significant load fluctuations, how to efficiently schedule and allocate computational resources, and avoid node overload or idleness, is crucial to enhancing system performance. To tackle this challenge, researchers have proposed optimization models based on fluid networks [5], ant colony algorithms [11, 12], and particle swarm optimization (PSO) [12, 13], among others.

The fluid network method is widely applied in resource allocation problems, especially in heterogeneous computing environments. By establishing an optimization model that minimizes the transmission delay between nodes in a network, the fluid network method dynamically adjusts network traffic to achieve load balancing. This method is especially suitable for environments with uneven resource loads and high communication delays between nodes.

Ant colony algorithms and particle swarm optimization (PSO) simulate the collective behavior and individual intelligence found in nature to optimize resource allocation and task scheduling. Studies have shown that ant colony algorithms can provide global optimal solutions in large-scale grid computing, particularly in multi-task parallel computing, where they help avoid resource wastage and increase system throughput. Particle swarm optimization, by simulating the motion trajectory of a particle swarm, quickly finds optimal resource allocation solutions, and performs particularly well in high-load, complex task environments. Several studies have shown that load balancing methods based on ant colony and particle swarm optimization algorithms significantly outperform traditional scheduling algorithms in terms of execution time and system stability.

4.3. Machine Learning-Based Intelligent Scheduling

In recent years, machine learning-based intelligent scheduling methods have gradually become a research hotspot in grid computing. Machine learning enables automatic adjustment of scheduling strategies by learning from a large amount of historical data, thereby achieving optimal allocation of system resources. The introduction of deep learning and reinforcement learning methods has enabled intelligent scheduling to not only adjust in real time but also continuously optimize decision-making strategies to adapt to complex and changing computational environments.

Reinforcement learning, as an unsupervised learning method, can gradually optimize task scheduling strategies through interactions with the environment. Unlike traditional rule-based scheduling methods, reinforcement learning can self-adjust based on the dynamic state of the system, achieving more flexible and efficient resource allocation. In recent years, reinforcement learning has achieved significant results in grid computing applications. For example, in some large-scale cloud computing platforms, using reinforcement learning for resource scheduling has significantly improved resource utilization and task execution efficiency. In specific cases, reinforcement learning models dynamically adjust scheduling strategies based on factors such as node load and task requirements, maximizing task execution parallelism and reducing task completion time.

4.4. Practical Applications of System Optimization

Through specific case studies, we can further verify the effectiveness of the aforementioned optimization methods in practical applications. For example, in the Grid5000 platform, by optimizing task scheduling and resource allocation strategies, the system's computing resource utilization and task completion efficiency were significantly improved. Additionally, certain large-scale scientific computing projects, such as LIGO (Laser Interferometer Gravitational-Wave Observatory) and CERN (European Organization for Nuclear Research), have adopted resource optimization methods based on load balancing and task priority scheduling. These optimization measures not only ensured

DOI: 10.54254/2755-2721/119/2025.21640

timely task completion but also improved the efficiency of computing resource usage, while ensuring system stability and robustness.

In the LIGO project, the priority-based scheduling strategy ensures the real-time processing of data and the rapid completion of high-priority tasks, ensuring the smooth progress of the project. Through dynamic task scheduling and resource optimization, the LIGO project effectively avoided node overload and resource idleness while processing large-scale data. Similarly, in the CERN project, by introducing multiple optimization algorithms, such as ant colony optimization, particle swarm optimization, and reinforcement learning models, the system can allocate resources in real time based on node load conditions, maximizing system computing efficiency.

With the continuous development of grid computing technologies, future system optimization will focus more on intelligence, dynamic adaptability, and self-adaptability. By integrating artificial intelligence and big data analytics, systems will be able to dynamically adjust scheduling strategies and resource allocation plans based on changing computational needs and load conditions. Furthermore, as quantum computing and edge computing mature, grid computing will further expand into more complex and diverse application scenarios. How to optimize resource management and task scheduling in this emerging environment will continue to be an important direction for future research.

5. Conclusion

This paper systematically studies the problem of optimizing resource scheduling in heterogeneous grid computing environments. It provides an in-depth analysis of the key factors that affect resource scheduling performance, including heterogeneity, task priority, network bandwidth, and latency. Based on this analysis, various optimization algorithms are examined, including task scheduling algorithms, load balancing algorithms, and dynamic resource scheduling strategies, with a discussion on their applicability.

To achieve efficient resource allocation, this paper proposes an optimization model based on linear programming. The model takes into account the heterogeneity of computing resources, task priorities, network conditions, and load balancing constraints, with the goal of minimizing task completion time and maximizing resource utilization. The effectiveness of this model was validated through simulation experiments on typical grid platforms. The results show that, compared to traditional scheduling algorithms, this method significantly reduces the average task waiting time and execution time, improving system performance and resource utilization.

Although the proposed method performed well in the simulation experiments, there are still limitations, such as the computational complexity of the linear programming model in highly dynamic, large-scale environments, and the fact that factors like storage and energy consumption were not considered. In conclusion, this paper provides an effective solution for resource optimization scheduling in heterogeneous grid environments and lays the foundation for future research on intelligent scheduling combined with machine learning.

References

[1] *Casanova, H., Legrand, A., & Quinson, M. (2008). SimGrid: a generic framework for large-scale distributed experiments. Proceedings of the Tenth International Conference on Computer Modeling and Simulation (UKSim 2008), 126-131.*

[2] *Kondo, D., et al. (2009). Characterizing and evaluating desktop grids: An empirical study. Journal of Grid Computing, 7(3), 361-388.*

[3] *Deelman, E., et al. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming, 13(3), 219-237.*

[4] *Bal, H. E., et al. (2000). Grid Computing: The New Frontier of High Performance Computing. Prentice Hall.*

DOI: 10.54254/2755-2721/119/2025.21640

[5] *Buyya, R. (Ed.). (2009). Grid Resource Management: State of the Art and Future Trends. Springer Science & Business Media.*
[6] *Ripeanu, M., et al. (2002). Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. IEEE Internet Computing, 6(1), 50-57.*
[7] *Foster, I., & Kesselman, C. (Eds.). (2003). The Grid: Blueprint for a New Computing Infrastructure (2nd ed.). Morgan Kaufmann.*
[8] *Pinedo, M. L. (2012). Scheduling: theory, algorithms, and systems. Springer Science & Business Media.*
[9] *Talbi, E.-G. (2009). Metaheuristics: From Design to Implementation. John Wiley & Sons.*
[10] *Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR), 35(3), 268-308.*
[11] *Dorigo, M., & Stützle, T. (2004). Ant Colony Optimization. MIT Press.*
[12] *Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press.*
[13] *Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942-1948.*