Research on Multi-Printer Intelligent Scheduling and State-Aware Methods for the Campus Unattended Cloud Printing Platform "Zhiyun Yiyin"

Zijie Yi, Baohua Ning, Weiping Ning, Xiaomei Yang*, Ziguang Lu
Guangxi Vocational Normal University, Nanning, Guangxi, China
**Corresponding Author*

Abstract: With the continuous advancement of smart campus development, campus printing services have placed higher demands on unattended operation, intelligent management, and high-concurrency processing. Traditional cloud printing systems still suffer from low scheduling efficiency and unbalanced resource utilization in multi-printer collaborative scheduling and device-state awareness. To address these issues, this paper investigates the campus-oriented unattended intelligent cloud printing platform Zhiyun Yiyin and proposes a multi-printer intelligent scheduling method that integrates print-task characteristics with device-state awareness. Specifically, the proposed method constructs a printer state-awareness model to obtain real-time operational status information and dynamically allocates print tasks according to task priority and device load, thereby improving overall system efficiency and stability. Based on the existing architecture of the campus unattended cloud printing platform, the functional modules for multi-printer intelligent scheduling and state awareness were implemented and experimentally validated. The results show that the proposed method outperforms traditional scheduling strategies in terms of print-task response time, device load balancing, and system stability. It can effectively satisfy high-concurrency printing demands in campus scenarios and demonstrates practical applicability.

Keywords: Unattended Campus Printing; Printer State Awareness; Multi-Printer Scheduling; Parallel Printing; Load Balancing

1. System Model

1.1 System Architecture and Workflow

This study focuses on an unattended intelligent cloud printing platform deployed in campus environments. The platform uses a WeChat Mini Program as the user interaction interface, while the backend system is responsible for order management, task scheduling, and printer control. After a user submits a print order through the Mini Program, the system parses the order and assigns the print tasks to specific printers according to the selected allocation mode.

In a multi-printer environment, the system must handle multiple concurrent orders, while different printers may vary in performance, current load, and availability. Without a proper scheduling and allocation mechanism, some printers may remain under heavy load for long periods whereas others stay idle, resulting in longer user waiting times and lower resource utilization. Therefore, it is necessary to establish a unified model of print tasks and printer resources at the system level, which provides the basis for subsequent scheduling-strategy design [1-3].

1.2 Print Task Model

In the proposed platform, an order is a print request submitted by a user and may contain one or more files together with their associated print settings. Each order is converted into one or more subtasks, and each subtask usually corresponds to one file in the order. If an order contains multiple files, the system splits the order by file so that the print job of each file is processed as an independent task.

The platform also calculates the total number of printed pages for each order. This calculation comprehensively considers the page count of each file, the user-specified print range, and the number of copies. For duplex printing, the system converts the page count according to the rule that one sheet contains two printed pages,

thereby ensuring accurate statistics for subsequent scheduling optimization and cost calculation [4].

When placing an order, users may either specify a target printer or select the intelligent allocation mode, in which the system automatically chooses a printer. If the user specifies a printer, all tasks in the order are bound to the queue of that device and can only be processed by that printer. In contrast, under the intelligent allocation mode, the platform does not pre-assign a specific device. Instead, it dynamically selects an appropriate printer according to real-time operating conditions at the time of execution.

The choice between these two modes directly affects printing efficiency and resource utilization. Binding an order to a fixed printer may cause it to wait in a long queue when that device is busy, whereas intelligent allocation enables the system to assign tasks to an idle printer or to the printer with the shortest expected completion time, thereby reducing waiting time. For orders containing multiple tasks, the intelligent allocation mode also allows different tasks to be assigned to multiple printers for parallel execution, further improving overall service efficiency.

Each file in an order is associated with a set of print parameters that precisely describe its printing requirements, including color mode, paper size, print range, simplex/duplex option, page orientation, and number of copies.

To address failures that may occur during task execution, the platform incorporates a complete exception-handling and reassignment mechanism. When a print task fails because of printer malfunction, paper shortage, paper jam, or network interruption, the system triggers an exception-handling procedure. First, it marks the task and its associated order as abnormal and records the cause of failure. The abnormal status is represented by dedicated fields indicating that the task was not completed normally, while error codes or textual descriptions are also retained for troubleshooting and subsequent analysis.

After an abnormal status is recorded, the platform immediately attempts automatic reassignment. The failed task is returned to the pending scheduling queue, and the system filters other eligible printers for reallocation. The intelligent scheduling engine then dispatches the task to a new suitable device without manual intervention, allowing the print job to continue on an alternative printer. This automatic transfer mechanism significantly improves system reliability, because when one printer fails, tasks can be shifted to other available devices, thereby minimizing the impact on order completion time and improving both service robustness and user experience [5].

1.3 Print Queue State Model

In a multi-printer environment, the execution efficiency of print tasks depends not only on the attributes of the order itself, but also on the real-time operating state of each printer and its queue condition. To support the design of subsequent intelligent scheduling and load-balancing strategies, this paper establishes a unified model of printers and their task queues, and treats device-state and queue information as important inputs to scheduling decisions.

1.3.1 Printer State Modeling

The system models printer state from the following three aspects.

(1) Availability status: As shown in Figure 1, the printer state transition diagram includes four states: idle, printing, busy, and abnormal. When faults such as paper jams, paper shortages, or low ink occur, the printer is marked as abnormal and is temporarily excluded from new task allocation.

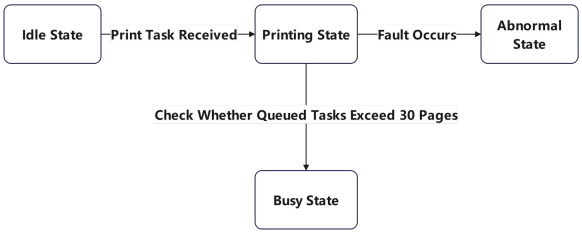

Figure 1. Printer State Transition Diagram

(2) Current task information: When a printer is busy, the system records the information of the task currently being executed, including the start JobId, end JobId, and the number of pages already completed. This information is used to estimate the remaining workload of the device and determine when it may accept new tasks.

(3) Printer identification information: Each printer has a unique identifier in the system, such as a printer number or printer name, which is used to distinguish different devices during scheduling and to establish the mapping between tasks and printers.

Through the above state model, the system can perceive the operating condition of each printer in real time during scheduling, thereby providing the basis for dynamic allocation in a multi-

printer environment.

1.3.2 Queue State Modeling

In addition to the operating state of an individual printer, the queuing condition of print tasks is another important factor affecting user waiting time. Therefore, this paper further models printer queue state in order to describe the queue pressure experienced by tasks in the system.

After a print task is assigned to a printer, it enters the queue of that device. For queue-state modeling, the system focuses on the following indicators.

(1) Number of tasks ahead in the queue:

This indicates how many tasks still need to be processed before the current task and reflects both queue length and device load.

(2) Total number of queued pages ahead:

This denotes the total number of pages of all pending tasks ahead of the current task. Compared with simply counting the number of tasks, this metric more accurately reflects the actual workload of the printer.

(3) Estimated waiting time:

Based on the total number of queued pages ahead, the historical printing speed of the printer, and its current operating state, the system estimates the waiting time of the task and expresses it in seconds or minutes. This indicator can be used both to display expected waiting time to users and to support scheduling decisions.

By modeling queue state in this way, the system can quantitatively characterize the queue pressure borne by different printers at the same moment, rather than relying only on coarse-grained states such as “idle” or “busy.”

1.3.3 State Awareness and Scheduling Inputs

By integrating printer operating state with queue state, this paper provides a unified abstraction of printer status and task queue information based on the existing system interfaces, and uses them as state inputs for the design of subsequent scheduling strategies.

When the user selects the intelligent allocation mode, the system gives priority to printers that are available and subject to lower queue pressure, and assigns tasks to the device with the shorter expected completion time. When a printer abnormality is detected, the system automatically removes that device from the candidate printer set to prevent abnormal states from negatively affecting the overall scheduling results, as shown in Figure 2. By introducing printer and queue state awareness, the system is able to dynamically coordinate multiple printer resources, thereby laying the foundation for the subsequent design of load-balancing and parallel-printing strategies.

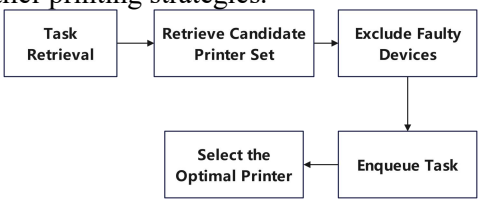

Figure 2. Scheduling Decision Process

1.4 Problem Description

Based on the analysis of the print task model and the printer and queue state model presented above, the multi-printer task allocation problem in the campus unattended printing platform can be abstracted as an online task scheduling problem in a multi-resource environment. In this system, let the set of print tasks be denoted by T, and the set of printers by P. Task is characterized by its page count n_i and submission time t_i^{submit}, while printer j is characterized by its printing speed v_j, the total number of queued pages ahead q_j, and the availability state variable a_j, where $a_j = 1$ indicates that printer j is available and $a_j = 0$ otherwise.

When a task arrives at the system, the scheduling module needs to select an appropriate printer from the candidate printer set under the given constraints. The task allocation process should satisfy the following constraints:

(1) Tasks can only be assigned to printers satisfying $a_j = 1$;

(2) if the user specifies a printer, the device-binding constraint must be satisfied;

(3) if a printer becomes abnormal during execution, task reassignment should be supported;

(4) when parallel printing is enabled, the split subtasks must remain independent while ensuring result consistency.

During the scheduling stage, let W_j denote the expected waiting time of printer . It is estimated based on the total number of queued pages ahead and the printing speed as follows:

$$W_j = \frac{q_j}{v_j} \quad (1)$$

Let C_{ij} denote the expected completion time of task i on printer j. It is defined as the sum of the waiting time and the printing time of the task on printer j:

$$C_{ij} = W_j + \frac{n_i}{v_j} \quad (2)$$

The scheduling strategy then selects the printer j^* that minimizes C_{ij} among all available printers:

$$j^* = \arg \min_{j \in P,\ a_j=1} C_{ij} \quad (3)$$

The overall optimization objective is to reduce average user waiting time, improve printer utilization, and enhance both load balancing and parallel processing efficiency in multi-printer scenarios.

2. Scheduling Strategy Design

2.1 Scheduling Rationale

In the multi-printer scenario of the campus unattended printing platform, the core objective of the scheduling strategy is to allocate print tasks reasonably while satisfying user requirements and system constraints, so as to reduce user waiting time, shorten order completion time, and improve printer utilization. Based on the print task model and the printer and queue state model established in Chapter 1, this paper adopts a state-aware multi-printer intelligent allocation strategy [6], which optimizes task assignment by utilizing real-time status information available in the system.

When the user selects the intelligent allocation mode, the system does not pre-bind tasks to a specific printer. Instead, it dynamically selects an appropriate device according to the current operating state and queue load of each printer. This strategy takes printer availability, queue pressure, and remaining workload as the main decision-making factors, thereby avoiding load imbalance caused by fixed rules or manual selection. At the same time, for orders containing multiple files or a large number of pages, the system supports splitting the order into multiple subtasks and assigning them to multiple printers for parallel execution when the device conditions are satisfied, thereby further improving the overall operating efficiency of the system, as shown in Figure 3.

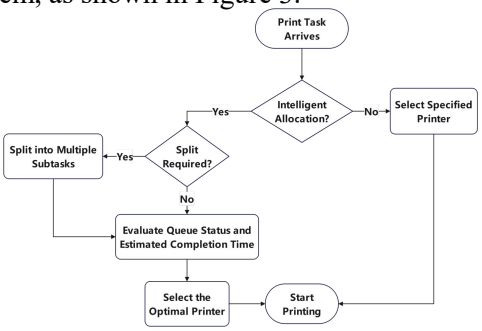

Figure 3. Intelligent Print Task Allocation and Scheduling Decision Flowchart

2.2 Intelligent Allocation Strategy

Under the intelligent allocation mode, the system obtains the real-time operating state and queue information of each printer before assigning print tasks, including whether the device is available, the number of tasks ahead in the queue, the total number of queued pages, and the estimated waiting time. Through comprehensive analysis of the above state information, the system can evaluate the current load of different printers and provide a basis for task allocation.

Specifically, when a new print task enters the scheduling stage, the system first filters the set of candidate printers that are available and satisfy the print-parameter constraints [7]. For example, for tasks requiring color printing or specific paper sizes, only printers supporting the corresponding functions are considered. For orders in which the user specifies a printer, the task is directly restricted to the designated device. The system then compares the queue states of the candidate printers, with particular attention to the number of queued pages and the estimated waiting time, so as to avoid assigning tasks to heavily loaded devices.

On this basis, this paper adopts an allocation principle oriented toward reducing queue pressure and shortening expected completion time. Print tasks are therefore preferentially assigned to printers with lighter queue loads and shorter expected completion times. When multiple printers are in similar states, the system may further consider task arrival order to maintain the stability and fairness of the scheduling process. By introducing printer state awareness, the system can dynamically adjust task-allocation results and effectively alleviate load imbalance in multi-printer scenarios.

2.3 Parallel Printing Strategy for Large-Scale Orders

For large-scale print orders with a large number of pages or copies, relying solely on a single printer for serial execution may result in excessive user waiting time and overload the device. To improve printing efficiency, the system introduces a parallel printing strategy in which multiple printers cooperate to process the same order, thereby accelerating task completion [8]. This strategy is state-aware in that it continuously acquires queue-related status information from each printer and uses it as the

basis for intelligent task allocation and dynamic load balancing.

The system determines whether to trigger the parallel printing strategy according to predefined conditions. First, at least two printers must currently be available; otherwise, task splitting is unnecessary. Second, the total number of pages in the order must exceed a minimum threshold, for example 10 pages, so as to avoid the communication and coordination overhead caused by splitting small orders. Third, the order must contain splittable print tasks, meaning that some files require multiple copies. If each file requires only one copy, the order cannot be effectively split across different devices. Meanwhile, the total number of copies must also be large enough to support multi-printer sharing, that is, the total number of copies should be no less than the number of available printers, ensuring that each selected device can be assigned at least part of the workload. Finally, the system evaluates the efficiency gain introduced by task splitting. Parallel printing is performed only when the expected completion time can be reduced by at least 15%, as controlled by the constant $MIN\_SPLIT\_BENEFIT\_RATIO = 0.15$. Through these rules, the system avoids unnecessary splitting when the expected benefit is limited or the trigger conditions are not satisfied.

After the trigger conditions are satisfied, the system enters the task planning and allocation stage of parallel printing. It first retrieves the list of eligible printers and filters out heavily loaded devices. For example, if the real-time number of queued pages on a printer exceeds a predefined threshold, such as 30 pages, that printer is temporarily excluded from assignment. In extreme cases where all printers are relatively busy, the system still selects the printer with the shortest queue to ensure that at least one feasible device remains available. The system also takes the color requirement of print tasks into account. Black-and-white tasks may be assigned to any suitable printer, including color printers, whereas color tasks can only be assigned to printers with color capability. After state-based filtering, the available printers are sorted in ascending order according to their current queue length, and priority is given to idle or lightly loaded devices. The system then calculates the optimal splitting plan based on the state of these printers, including the actual number of printers to be used and the workload assigned to each one. It evaluates the expected completion time under different numbers of participating printers and selects the scheme with the shortest completion time as the basis for parallel-printing decisions.

After determining the number of participating printers and the workload allocation scheme, the system proceeds to split and schedule the tasks within the order. The splitting granularity is defined at the copy level rather than the page level. In other words, each complete copy of the same file is printed by a single printer rather than being divided across multiple devices. For tasks that require multiple copies, the system distributes these copies among printers as evenly as possible according to their remaining capacity and ideal load ratio. During this process, the algorithm ensures that each selected printer receives at least one copy when the total number of copies is sufficient, and it corrects any discrepancy between the allocated total and the original requirement so that the final allocation remains fully consistent with the original order. For tasks that do not satisfy the splitting conditions, such as files requiring only a single copy, the system assigns the entire task to the printer with the lightest load, thereby maintaining balanced workload distribution. As a result, different files or copies within the same order can be executed in parallel by multiple printers, and each device undertakes part of the workload, significantly shortening the overall completion time.

After the initial allocation is completed, the system further checks whether the loads of the participating printers are balanced. If one printer is assigned substantially more pages than the others, for example, when the load difference exceeds 30% of the average, and the assigned task still contains copies that can be further divided, the system transfers part of the workload to the least-loaded printer to further optimize load balancing. This optimization process is repeated iteratively until the load difference is reduced to a reasonable range or no further splitting is possible, thereby improving the synchronization and efficiency of multi-printer task execution [9].

To ensure output correctness and order consistency under the parallel printing strategy, the system also introduces supporting mechanisms during implementation. First, each file task is always printed in full on a single

printer, which avoids content-order confusion and post-processing difficulties caused by cross-device output. Second, when an order is split across multiple devices, the system generates a corresponding sub-order for each printer and appends a suffix to the original order number for identification and association. In this way, even though a large order is divided into multiple subtasks executed in parallel, the backend can still track these subtasks and ensure that they all correspond to the same original order. The system also records the page count and fee information of each sub-order separately and uses a pre-calculated split price to ensure that the sum of all sub-order charges remains equal to the total price of the original order. Finally, after all subtasks are completed, the system aggregates the results according to the order-group identifier and delivers the complete printed output to the user. In summary, the proposed parallel printing strategy not only improves the processing efficiency of large-scale orders, but also guarantees the orderliness of the splitting process and the consistency of final results.

3. System Implementation and Evaluation

3.1 System Architecture and Functional Modules

To support the state-aware multi-printer intelligent scheduling strategy proposed in Chapter 2, the backend system implements several key functional modules for scheduling-oriented decision-making, including print task abstraction, printer status collection and queue information maintenance, and a state-based task dispatch mechanism. This chapter focuses on the system modules and implementation methods that are directly related to the realization of the scheduling strategy, while other general business functions are omitted.

3.2 Implementation of Printer Status Collection and Queue Information Management

To support the implementation of the state-aware scheduling strategy, the system needs to periodically collect the real-time status information of each printing device and make it available to both the scheduling module and the query interface [10]. To this end, the backend service includes a dedicated scheduled-task module that repeatedly polls each registered printer to obtain its current operating state and queue condition. The collected information includes the printer status code and description, such as idle, printing, and offline; whether the device is in an abnormal state, such as paper shortage or paper jam; and key queue-related indicators, including the total number of waiting tasks and the total number of queued pages. These data are obtained in real time through the operating system print queue interface or the printer-provided API and are encapsulated into a $PrinterQueueInfo$ object for further processing. The $PrinterQueueInfo$ object contains fields such as printer name, printer number, status description list, abnormal-state indicator ($isError$) and error message, total number of jobs ($totalJobs$), total number of pages ($totalPages$) information about the job currently being printed, and the ID list and detailed information of all waiting jobs in the queue. When a printer fault is detected, the system sets $isError$ to $true$ and records the reason for the fault, enabling the scheduling strategy to identify and avoid abnormal devices. This periodic collection mechanism ensures real-time awareness of printer status and provides data support for subsequent scheduling decisions [11].

The collected printer queue information is then written into Redis in a timely manner to improve access efficiency. The system organizes Redis keys using a hierarchical namespace based on the merchant ID and printer name, for example, $printer: queue: < merchantId > : < printerName >$. After each collection cycle is completed, the corresponding $PrinterQueueInfo$ object is serialized and stored under the related key. This caching design allows both the scheduling module and the user query interface to directly obtain the latest queue state from Redis without repeatedly accessing the physical printing device. At the same time, the scheduling module can periodically retrieve the collection of queue keys for all printers under the same merchant by using wildcard queries in Redis, thereby obtaining a snapshot of all printer states at one time. If the key corresponding to a certain printer does not exist or has expired, the system treats that printer as offline or not reporting status and avoids it during scheduling. With the support of Redis in-memory caching, the state-aware scheduling strategy can obtain the latest load and health information of each printer with low latency,

thus enabling faster and more reliable task allocation.

In the user query interface, the system calculates and returns order queue information based on the cached printer queue data. Specifically, when a user initiates a query using an order number, the backend first locates the corresponding order record and its associated print-task mapping. If the order has not yet been assigned to a printer, that is, the print-task mapping is empty, the system determines that the order has not entered the print queue and directly returns the status Pending, while setting the number of queued tasks and queued pages to zero. Once the order has been mapped to a printer task, the system retrieves the current queue information of the corresponding printer from Redis. By comparing the start and end Job IDs of the print task and scanning the job ID list of the printer queue, the system identifies the position of the order within the queue. On this basis, it calculates the number of tasks ahead in the queue ($queueAhead$) and the total number of queued pages ahead ($queueAhead$). It then converts $pagesAhead$ into an estimated waiting time, such as $estimatedWaitSeconds$ [12], using a predefined printing-speed constant, and further converts it into $estimatedWaitMinutes$. These queue-related indicators are encapsulated into an $OrderQueueInfo$ object and returned to the frontend, including fields such as order number, printer name or number, $queueAhead$, $pagesAhead$, and estimated waiting time, thereby allowing users to clearly understand how many tasks and pages remain ahead of their order and how long they are expected to wait.

Based on the above queue information, the system dynamically determines the current order status and presents it to the user in an intuitive form. When $queueAhead = 0$, it indicates that the order has reached the head of the queue and is currently being printed; in this case, the system sets the $isPrinting$ flag to $true$ and returns a status description such as Printing. When $queueAhead > 0$, the task is still waiting in the queue, and the status is displayed as Queued (n tasks ahead), where n is the number of tasks ahead. If the corresponding Job ID cannot be found in the cached queue list, that is, $orderInQueue = false$, the system infers that the print job has been removed from the queue, which generally means that it has either been completed or terminated. In this case, $queueAhead$ and related indicators are set to zero, and the returned status is Completed. For orders that have never entered the queue, the system returns Pending, as described above. By combining database order records with real-time queue data, the system can accurately distinguish whether an order has not yet started printing, is waiting or printing in the queue [13], or has already been completed, thereby providing accurate and timely status feedback to users.

In summary, the printer status collection and queue information maintenance module provides a solid foundation for implementing state-aware scheduling in the system. On the one hand, the scheduling strategy can use the real-time data in Redis to understand the load and availability of each printer and intelligently assign new print tasks to idle or lightly loaded devices, while avoiding abnormal or busy printers, thereby improving overall printing efficiency and reliability. On the other hand, the user query interface uses the same set of status data to make order queue progress transparent, thereby improving users' expectations regarding waiting time. This unified mechanism for status collection and information sharing enables the system to perform globally optimized task scheduling while also providing fine-grained progress feedback, fully demonstrating the practical value of the proposed state-aware scheduling strategy in a real system [14].

4. Conclusion

This paper addresses the problems of load imbalance and queue congestion faced by the campus unattended intelligent cloud printing platform during peak periods. A multi-printer state-awareness model is constructed, and an intelligent scheduling and dynamic splitting strategy integrating queue information and task scale is proposed. By introducing printer-state evaluation and a benefit-threshold mechanism, the system achieves adaptive task allocation and parallel execution across multiple devices. Experimental results show that, while maintaining system stability, the proposed method effectively reduces average order waiting time and overall completion time and improves resource utilization. Future work will further incorporate predictive models and real-time load-trend analysis to enhance the adaptability and generalization capability of the system.

Acknowledgments

This work was supported by the China College Students' Innovation and Entrepreneurship Practice Program "AI-Enabled Intelligent Cloud Printing: 24h Air-Ground Integrated Intelligent Cloud Printing 3.0" (No. 202514684027S).

References

[1] Li Shi, Du Hongbo, Sun Jiaqi, et al. Print Terminal Based on “Internet + Sharing with Cloud”. Software, 2020, 41 (07): 72-75.

[2] Lin Q, Liu J, Tretter D. Printing in a Digital Age. IEEE MultiMedia, 2010, DOI: 10.1109/MMUL.2010.84.

[3] Hao Lijiang, Tian Luyun, Sun Peng, et al. Task Scheduling Algorithm for Intelligent Interpretation of Remote Sensing Data Based on Heterogeneous Platforms. Journal of Electronics & Information Technology, 2025, 47 (12): 4742-4753.

[4] Suman Lata, Dheerendra Singh, Sukhpreet Singh. A Hybrid Approach for Cloud Load Balancing Optimization. Journal of Electrical Systems, 2024, 20 (9s): 1-10.

[5] Athokpam Bikramjit Singh, Rio D’souza. A Hybrid Approach of Load Balancing in Cloud Computing by Optimization of Metaheuristic Techniques: An Execution Assessment. International Journal of Engineering Research in Electronics and Communication Engineering, 2022, 9 (11): 1-8.

[6] Yousef Sanjalawe, Salam R. Al-Emari, Salam Fraihat, Sharif Naser Makhadmeh. AI-driven job scheduling in cloud computing: a comprehensive review. Artificial Intelligence Review, 2025, DOI: 10.1007/s10462-025-11208-8.

[7] Yamari S., Benlahmar E. H. AI-Based Load Balancing in Cloud Computing: A Systematic Review and Future Directions. Mathematical Modeling and Computing, 2026, DOI: 10.23939/mmc2026.01.102.

[8] Simaiya S, Lilhore U K, Sharma Y K, et al. A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques. Science & Technology - Other Topics, 2024, DOI: 10.1038/s41598-024-51466-0.

[9] Li T., Ying S., Zhao Y., et al. Batch Jobs Load Balancing Scheduling in Cloud Computing Using Distributional Reinforcement Learning. IEEE Transactions on Parallel and Distributed Systems, 2023, DOI: 10.1109/TPDS.2023.3334519.

[10] Albalawi N. Dynamic scheduling strategies for cloud-based load balancing in parallel and distributed systems. Journal of Cloud Computing, 2025, DOI: 10.1186/s13677-025-00757-6.

[11] Wang J., Cheng W., Zhang W. AoI-Aware Resource Allocation for Smart Multi-QoS Provisioning. IEEE Systems Journal, 2024, DOI: 10.1109/JSYST.2024.3519536.

[12] Wang Y., Liu Q., Zhang D., et al. Adaptive Resource Scheduling and Management Optimization in Digital Education with Compute First Networking. IEEE Transactions on Consumer Electronics, 2025, DOI: 10.1109/TCE.2025.3581902.

[13] Chayon M. H. R., Shouvo N. H., Mamun M. A. A. An Efficient Scheduling Algorithm Based on Buffer Status and Queue Length to Satisfy the Real-time Users in 5G. 2024 27th International Conference on Computer and Information Technology (ICCIT), 2024, DOI: 10.1109/ICCIT64611.2024.11022462.

[14] Sheikh Umar Mushtaq, Sophiya Sheikh, Sheikh Mohammad Idrees. Enhanced priority based task scheduling with integrated fault tolerance in distributed systems. International Journal of Cognitive Computing in Engineering, 2025, 6: 152-169.

