International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Dynamic Priority Scheduling Algorithms for Flexible Task Management in Cloud Computing

[1]Rohith Sai Kamal Aakisetti, [2]Vanaja Ganta, [3]Pachipala Yellamma, [4]Chandana Siram, [5]Sri Harshani Gampa, [6]K. V. Brahma Rao

Submitted: 26/11/2023 **Revised**: 06/01/2024 **Accepted**: 16/01/2024

Abstract: Cloud computing on-demand resources revolutionized computing. However, adapting to dynamic workloads and resource availability is a challenge for traditional static scheduling algorithms. In response, Dynamic Priority Task-Based Scheduling (DPTS) is introduced. DPTS dynamically adjusts task priorities and scheduling decisions in real-time to optimize resource utilization by considering factors such as urgency importance and resource requirements. DPTS is a revolution in cloud scheduling because it's adaptable and efficient, unlike static algorithms that follow predictable patterns. It's like a dance that allows resources and tasks to work together smoothly. It's pretty cool how DPTS improves cloud performance and promises a future where everything works in perfect harmony. This new paradigm in cloud scheduling addresses the limitations of traditional approaches and enhances overall system efficiency. Simulations and comparative analyses demonstrate DPTS' effectiveness in optimizing resource utilization, minimizing task completion times, and improving cloud-based tasks' performance. As cloud computing evolves, DPTS contributes significantly to enhancing the efficiency and adaptability of cloud-based systems.

Keywords: *Resource-Aware Scheduling, HEFT, DHEFT, Task Scheduling Algorithms, Dynamic Priority Task-Based Scheduling.*

1. Introduction

In the world of cloud computing, efficient task scheduling is crucial for optimizing resource utilization and meeting user requirements. To address this, a priority-based Cloud Task Scheduling list of instructions has been developed. This step by step instructions aims to allocate priorities to tasks based on factors such as task characteristics, user requirements, and resource availability [1]. By considering these priorities during scheduling, the algorithm can optimize task execution and improve overall system performance. In this paper, we will explore the design, implementation, and progression of this priority-based algorithm, highlighting its benefits and potential applications [3]. The concept of cloud computing has gained significant popularity due to its ability to offer on-demand access to a variety of resources. Task scheduling is an essential aspect of cloud computing, aims to allocate available resources efficiently to ensure optimum performance. In recent years, various task scheduling algorithms which were proposed to address the challenges posed by dynamic and heterogeneous environments [4]. However, existing algorithms often fail to consider the dynamic nature of tasks and adequately prioritize them. In this essay we will thoroughly examine a cloud task scheduling algorithm that utilizes dynamic priority and the three queues. The algorithm aims to enhance task scheduling efficiency by considering factors such as execution time, waiting time, and task priority. By employing a three-queue structure and dynamically adjusting task priority, the proposed algorithm can effectively allocate resources and optimize task scheduling in the cloud computing environments. This essay examines the methodology and performance evaluation of the algorithm, providing valuable perception into its application in real-world scenarios [6]. Definition of cloud task scheduling: Another important aspect of cloud task scheduling is the definition of the term itself. Cloud task scheduling refers to the process of allocating computing resources to different tasks in the cloud computing environment. It involves determining the order and priority in which tasks should be executed to achieve efficiency and optimize resource utilization. In cloud computing, there are typically multiple tasks that need to be performed simultaneously, and these tasks can have varying priorities, deadlines, and resource requirements. Therefore, an effective cloud task scheduling algorithm is essential in ensuring that tasks will be executed in a timely manner whereas meeting the desired quality of service. The aim is to reduce execution time, maximize resource utilization, and meet the specified deadlines for the tasks.

This requires intelligent decision-making based on task characteristics, resource availability, and system constraints. A well-designed cloud task scheduling algorithm plays a key role in maximizing the overall performance and efficiency of a cloud computing system. Importance of efficient cloud task scheduling for improved resource utilization and customer fulfilment. In addition to resource utilization, efficient cloud task scheduling plays a

[1,2,3,4,5,6] *Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India*

crucial role in ensuring customer satisfaction in cloud computing systems. The efficient allocation of resources is essential to meet the increasing demand of cloud services and to achieve maximum utilization of available resources. Cloud task scheduling algorithms, such as the one based on three queues and dynamic priority, are designed to optimize resource allocation by considering various factors like processing power, memory requirements, and network bandwidth. By effectively prioritizing tasks based on their importance and urgency, the algorithm ensures that critical tasks are allocated necessary resources, thereby minimizing their response time and improving overall system performance. Moreover, by efficiently managing the allocation and execution of tasks, the algorithm allows for better utilization of resources, reducing resource wastage and cost. These improvements in resource utilization and customer satisfaction are essential for cloud.

The Service providers are to remain competitive in the market and deliver high-quality services to their customers. Deadline-Based Task Scheduling Algorithm: This algorithm considers task deadlines when assigning priorities [23]. Tasks with closer deadlines are given higher priorities, ensuring timely completion and avoiding any potential delays. Priority-Driven Task Scheduling Algorithm: This algorithm assigns priorities to tasks based on their importance and urgency [24]. It ensures that high-priority tasks are executed first, optimizing resource utilization and meeting user requirements. Dynamic Priority Task Scheduling Algorithm: This algorithm dynamically adjusts task priorities based on changing conditions and system requirements. It adapts to resource availability, workload variations, and user demands to optimize task scheduling in real-time. Resource-Aware Task Scheduling Algorithm: This algorithm takes into account the availability of resources when assigning priorities [25]. It aims to balance resource utilization and avoid resource bottlenecks by scheduling tasks based on resource availability. User-Preference Task Scheduling Algorithm: This algorithm considers user preferences when assigning priorities. It takes into account factors such as user profiles, preferences, and service-level agreements to ensure personalized task scheduling and enhance user satisfaction.

The paper's organization of the information follows: Section 2 offers a comprehensive literature review of existing research on Dynamic Priority Task Scheduling Algorithms. In Section 3, we discuss the current methodology for dynamically adjusts task priorities based on the current system state and introduced our proposed methodology use DPTS. We explain the Dynamic Priority Task Scheduling process in detail. Section 4 presents the results and comparative analysis of our proposed methodology. Finally, Section 5 provides a detailed explanation of the outcomes and the ending statements of the research paper.

2. Literature Review

“Cost Based Task Scheduling Algorithm” was proposed by Shikha, Garg. Cloud computing is offers different services to the users, such as Software as a Service (SaaS) and Infrastructure as a Service (IaaS). This paper center of attention is on task scheduling in a cloud computing environment, specifically at the platform and infrastructure levels. The goal is to allocate 'm' functions to 'n' virtual machines, where 'm' is superior than 'n' [5]. The algorithm evaluates the processing cost of the each task on each virtual machine and uses the Shortest Job First (SJF) algorithm to allocate tasks based on the minimum processing load. The total processing cost is calculated to achieve optimal results. The algorithm starts by reading the number of virtual machines (n) and tasks (m) and assigns tasks to virtual machines based on their processing cost. It selects virtual machines with the minimum processing load for task allocation. The algorithm continues until all tasks are assigned. An example implementation with 4 virtual machines and 10 tasks is provided, demonstrating task allocation based on processing costs.

“A Survey On Task Scheduling Model Using Optimization Technique was proposed by S.Radha, A. Nandhini, T.V.Pavithra and G.Umarani Srikanth”. The objective is to optimize cloud computing systems through efficient task scheduling [7]. The main aim of a task scheduling algorithm is to reduce makespan (total task time) and increase resource utilization. This paper explores different algorithms like Max-Min, Genetic Algorithm, PSO, Ant Colony Algorithm, and Bee Colony Algorithm [8]. Hybrid Cuckoo Algorithm, which combines Genetic Algorithm and Cuckoo Algorithm, enhances energy efficiency, execution time, and the resource utilization. This approach eliminates the need for traditional task scheduling algorithms, further reducing scheduling time.

“Enhanced Max-min Task Scheduling Algorithm in Cloud Computing” was proposed by Bhoi Upendra, Ramanuj Purvi N. In order to achieve reduced waiting time, reduced makespan , optimal resource utilization, and better performance, efficient scheduling is of utmost importance. The authors propose the SA (Scheduling Algorithm) as a solution to enhance traditional scheduling approaches. Through evaluation using the CloudSim framework, the SA algorithm demonstrates superior performance compared to existing SJF algorithms; consistently reducing processing times [10].

"Dynamic Fair Priority Optimization Task Scheduling Algorithm: Concepts and Implementations" was proposed by Deepika Saxena, R.K. Chauhan, Ramesh Kait. This paper Concepts and Implementations" explores a task scheduling algorithm that aims to optimize task scheduling

at both the system and user levels in cloud computing. It introduces the concept of "Weighted Fair Queuing" to enhance the quality of service (QoS) in task scheduling. The paper addresses the challenges of resource allocation, task execution order, and overhead minimization, VM monitoring, and cost considerations in cloud task scheduling. The proposed algorithm classifies tasks into deadline-based and reduced cost-based groups and applies dynamic optimization and priority equity. It utilizes three priority queues which are (high, mid, low) with assigned weights, implementing a round-robin approach [11]. The algorithm aims to benefit both users and service providers by providing fairness and efficiency at the priority level.

"A Task Scheduling Algorithm with Improved Makespan Based on Prediction of Tasks Computation Time algorithm" was proposed P. Fan, by B. A. Al-Maytami, P. Liatsis, T. Baker, and A. Hussain. The paper presents a new scheduling algorithm that combines Directed Acyclic Graphs (DAGs) and the Prediction of the Tasks Computation Time (PTCT) algorithm [13]. The main objective is to enhance task scheduling performance and minimize computational costs in cloud computing. The use of Principal Component Analysis (PCA) to minimize matrix size is a unique approach in cloud computing context. PTCT algorithm is compared to other state-of-the-art scheduling algorithms, such as Min-Min, Max-Min, QoS-Guide, and Min-Max, and simulation results demonstrate its superior performance in terms of speedup, efficiency and schedule length ratio. The proposed solution addresses the challenges of task scheduling in heterogeneous cloud computing environments by leveraging PCA and PTCT to improve efficiency and reduce computational costs.

The paper introduces PTCT, a novel task scheduling algorithm for cloud computing. It addresses the need for well ordered scheduling in heterogeneous systems by utilizing DAGs and PCA to reduce the dimensionality of the ETC matrix [13]. PTCT aims to minimize makespan, improve resource utilization, and consider QoS constraints. Simulation results show its superiority over other algorithms in terms of the efficiency (well ordered), schedule length ratio and speedup. PTCT offers a fixed approach for upgraded task scheduling in cloud computing environments."Improvement of the Dynamic Priority Scheduling Algorithm Based on a Heapsort," was proposed by Q. Zhu, S. Meng, and F. Xia. The algorithm proposed here takes into account parameters like task deadline, task value and energy consumption. It uses a technique called hierarchy process (FAHP) to prioritize tasks. To efficiently sort these tasks in order of priority the algorithm employs heap sort, which's well known for its low time complexity [15]. The results, from the experiments indicate that this enhanced approach leads to a decrease in the frequency of missed deadlines, by an average of 0.1789 thereby improving the performance of scheduling.

The algorithm is specifically designed for real time systems. Has an application, in industries such, as industrial control and data center resource scheduling. Its primary objective is to improve task scheduling in overloaded systems by considering factors and utilizing heap sort for sorting. Ultimately reducing the deadline miss rate and improving scheduling performance [16].

The Table.1, It provides a literature review based on the comparison of methodologies and drawbacks for the existing methodologies. It provides a comparison of different approaches, including Analytic Hierarchy Process (AHP), Induced Bias Matrix Method (IBMM), Task Scheduling Algorithm (SA) and more.

Table 1 Comparing of Related Work/Comparing the methodologies and drawbacks proposed by different authors

Author	***Paper Title***	***Methodology***	***Drawbacks/Limitations***
R. K. Dash [1]	Task Scheduling in Cloud Computing: A Priority-Based Heuristic Approach	Analytic Hierarchy Process (AHP), GGWO, Bacteria Foraging (BF) algorithm, GSA, MGGS, NSGA, GGWO.	Lack of Real-World Validation, Sensitivity to Parameters, Computational Complexity.
D. Ergu [2]	The analytic hierarchy process: Task scheduling and resource allocation in cloud computing environment	Analytic Hierarchy Process (AHP), Pairwise Comparison Matrix Technique, Induced Bias Matrix Method (IBMM).	Lack of empirical evidence, Limited scope of research, Inconsistency identification process, Lack of exploration on dynamic resource allocation.
S. Huang [3]	A Johnson's-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of	Simulation Environment, Task and Machine Setup, Johnson's Rule-based Decoding Genetic Algorithm(GA) Execution	The JRGA aims to optimize the makespan of tasks, it does not provide any formal proof of optimality guarantees of the

	Cloud Computing		proposed algorithm.
Y.J. An [4]	Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times	Heuristic Algorithm, Maggu and Das's Algorithm, Lower Bound Calculation	It does not explore other types of scheduling problems or provide a comprehensive analysis of scheduling methodologies.
Manshi Bhonsle and Yogita Chawla [6]	Dynamically optimized cost based task scheduling in Cloud Computing	Processing cost matrix (PCM) processing cost matrix (PCM) to determine the cost of processing each task on each virtual machine. Shortest Job First (SJF) algorithm to select tasks based on minimum processing load	Scalability Concerns, Assumption of Known Processing Costs, Limited Comparative Analysis, Real-world Validation, Single Objective Focus
Arora, Sumit and Anand, Sami. [9]	Improved Task Scheduling Algorithm in Cloud Environment. International Journal of Computer Applications	CloudSim Framework, Task Scheduling Algorithm (SA)	Simplistic Problem Sets, Algorithm Complexity Explanation, Limited Evaluation Metrics, Limited Comparison
Sateesh Kumar Peddoju and Monika Chaudhary [12]	A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment	Task Grouping, Prioritization, Greedy Allocation	Simulation Dependency, Limited Evaluation Metrics, Lack of Comparative Analysis, Homogeneous Resource Assumption
M.Hussin, J.Y.Maipan-uku, A.Abdullah and A.Muhammed.[14]	Max-Average: An Extended Max-Min Scheduling Algorithm for Grid Computing Environment	Evaluation of Data Quality Dimensions (DQDs), Ant Lion Optimization (ALO), Regulation of Security Risks in Cloud Storage	Simulation Environment Dependency, Algorithm Complexity and Practical Feasibility, Limited Comparison and Generalizability, Limited Evaluation Metrics
Y. Liu, W. Jing, X. Sun and W. Wei [18]	Enhancing energy-efficient and qos dynamic virtual machine consolidation method in cloud environment	SLA time per active host (SLATAH), performance degradation due to the migrations (PDM), energy and SLAV (ESV), and VMM. The proposed EQVC method is compared with the DTHMF method and the RUA method.	EQVC method are not explicitly discussed in this content.
S. Pang, W. Li, X. Wang H. He and Z. Shan [19]	An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing	The algorithm combines the advantages of Genetic Algorithm (GA) and Estimation of Distribution Algorithm (EDA) which helps to optimize the task completion time and load balancing degree	Its performance in a more extensive array of real-world cloud computing scenarios remains uncertain.
Y. Su and Y. Yu [24]	Cloud Task Scheduling Algorithm Based on Three Queues and Dynamic Priority	Proposal of TQ algorithm, which categorizes jobs into two queues based on their type (CPU-intensive or I/O-intensive)	It focuses only on comparing the performance of different scheduling algorithms in Hadoop platform.

"Classification-Based and Energy-Efficient Dynamic Task Scheduling Scheme for Virtualized Cloud Data Center,"

was proposed by F. Zhang, S. Pirbhulal, R. M. Parizi, Z. Liu, K. -K. R. Choo and A. Marahatta. This paper focuses on the challenges posed by rapid growth of the cloud data centers (CDCs), such as inefficient resource utilization and high energy consumption [17]. To address these challenges, the authors propose a energy-efficient dynamic scheduling scheme (EDS) for real-time tasks which are in virtualized CDCs. The EDS scheme aims to optimize energy efficiency, Ensure task scheduling and resource allocation to maintain a ratio of task guarantees minimize response time and maximize resource utilization. This can be achieved through adjustments, in task scheduling and efficient resource provisioning [25]. Additionally, the paper introduces a task merging strategy to maximize resource utilization by merging similar types of tasks and scheduling them on the same physical host [26]. Experimental output supports the effectiveness of the preferred approach in improving efficiency and reducing energy consumption in CDCs.

“Intelligent model design of cluster supply chain with horizontal cooperation" was proposed by J. H. Park, S. Ma, J. Li, C. Liu, N. Xiong and S. Cho.This paper focuses on some of the problems in task scheduling in cloud computing. The authors propose an EDA-GA hybrid scheduling algorithm that combines the Genetic Algorithm (GA) and Estimation of Distribution Algorithm (EDA) [20]. The algorithm aims to improve the system load balancing and to minimize the task completion time [27]. It formulates a multi-objective task scheduling model considering both task completion time and load balancing. Experimental results using the CloudSim simulation platform indicates the effectiveness of the hybrid approach in achieving efficient scheduling in cloud environments.

3. Proposed Methodology

DPTS stands out among its counterparts due to its adaptability and responsiveness to real-time changes in system dynamics. Unlike static scheduling algorithms, which allocate priorities based on predetermined factors, DPTS dynamically adjusts task priorities based on the current system state. This dynamic nature enables DPTS to optimize resource utilization, improve throughput, and minimize latency. One of the key strengths of DPTS lies in its ability to prioritize the tasks which are based on their urgency, importance and ensuring that critical tasks receive immediate attention. This feature makes it particularly well-suited for applications where timely execution is paramount, such as real-time systems, cloud computing, and edge computing environments. This predictive element enhances the algorithm's decision-making process, allowing it to anticipate future resource requirements and allocate them judiciously. As a result, DPTS minimizes the risk of bottlenecks and ensures a smoother flow of tasks through the system. The algorithm's versatility is further demonstrated in its seamless integration with multi-core and distributed systems. DPTS optimally distributes tasks across available cores, promoting parallelism and enhancing overall system performance. Its adaptability to diverse computing environments positions DPTS as a reliable choice for the wide range of applications, from embedded systems to the large-scale data centers.

3.1. Dynamic Adjustments

The priority of a task and its impact on scheduling using a weighted combination of various factors. Let's denote the priority of a task as P_i and the efficiency factor as E_i, representing the efficiency of resource utilization and task completion time. The overall priority (OP_i) of a task can be calculated using the following formula:

$$OP_i = w_1 * P_i + w_2 * E_i$$

Where:

- P_i is the dynamic priority of task based on its characteristics and system conditions.
- E_i is the efficiency factor of task, indicating how well the task utilizes resources and minimizes completion time.
- w_1 and w_2 are weight coefficients representing the importance of priority and efficiency in the overall priority calculation.

The formula dynamically adjusts task priorities (P_i) based on both inherent characteristics and their impact on resource utilization and completion time (E_i). Tuning weights w1 and w2 allows customization to specific cloud computing goals. The overall priority (OP_i) guides the scheduling algorithm, prioritizing tasks with higher values for optimized resource use and minimized completion times, thereby enhancing system performance.

3.2. Resource Allocation

To represent the resource allocation component of the DPTS algorithm mathematically, we can use a weighted sum approach that considers multiple factors, which are CPU usage, memory requirements, and network bandwidth. Let's denote the resource allocation score for task ii as Ra_i, and calculate it using the following formula:

$$Ra_i = w_1 * CPU_i + w_2 * Memory_i + w_3 * Bandwidth_i$$

Where:

- CPU_i Represents the CPU usage of task.
- $Memory_i$ Represents the memory requirements of task.
- $Bandwidth_i$ Represents the network bandwidth requirements of task.

- w_1,w_2, and w_3 are weight coefficients representing the importance of CPU usage, memory, and network bandwidth, respectively.

The DPTS algorithm utilizes a formula to inform resource allocation decisions, incorporating task characteristics with adjustable weights (w_1,w_2,w_3) for CPU usage, memory, and network bandwidth importance. The resource allocation algorithm prioritizes tasks based on higher resource allocation scores, ensuring efficient utilization by addressing individual task needs for CPU, memory, and network resources. This approach optimizes resource allocation for tasks with diverse requirements within the system.

3.3. Priority Assignment Formula

The priority assignment formula in the provided pseudocode can be expressed as follows:

$$task.priority = W_u * task.u + W_{imp} * task.imp + W_{res} * calculate\_resource\_score(task.resource\_requirements, available\_resources) + (1 - system\_load)$$

Where:

- W_u is the weight assigned to task urgency,
- W_{imp} is the weight assigned to task importance,
- W_{res} is the weight assigned to resource requirements,
- task.u is the urgency of the task,
- task.imp is the importance of the task,
- task.resource_requirements is a dictionary representing the resource requirements of the task (e.g., {'CPU': 20, 'Memory': 30, 'Bandwidth': 10}),
- available_resources is a dictionary representing the available resources in the system,
- calculate_resource_score is a function that calculates a resource score based on task requirements and available resources,
- system load is a value representing the overall system load.

The formula dynamically adjusts task priority based on urgency, importance, resource requirements, and system load, with adjustable weights reflecting the assigned importance in the dynamic priority assignment mechanism.

3.4. Proposed Algorithm

Step 1: Define a class called "Task" with attributes like task_id, priority, urgency, importance, and resource_requirements.

Step 2: Create the "update_task_priority" function that takes in parameters: task, user_parameters, and system_conditions.

Step 3: Extract the user-defined parameters (weight_urgency, weight_importance, weight_resources) from the user_parameters dictionary.

Step 4: Extract the system conditions (system_load, available_resources) from the system_conditions dictionary.

Step 5: Calculate the resource score by dividing the sum of task requirements by the sum of available resources.

Step 6: Update the task's priority by multiplying the weights with the corresponding attributes of the task and adding them together.

Step 7: Assign the calculated value to the task.priority attribute.

Step 8: Example usage: Define user_parameters, system_conditions, and create a task object.

Step 9: Call the "update_task_priority" function with the task object, user_parameters, and system_conditions.

Step 10: Print the updated priority of the task using the task.priority attribute.

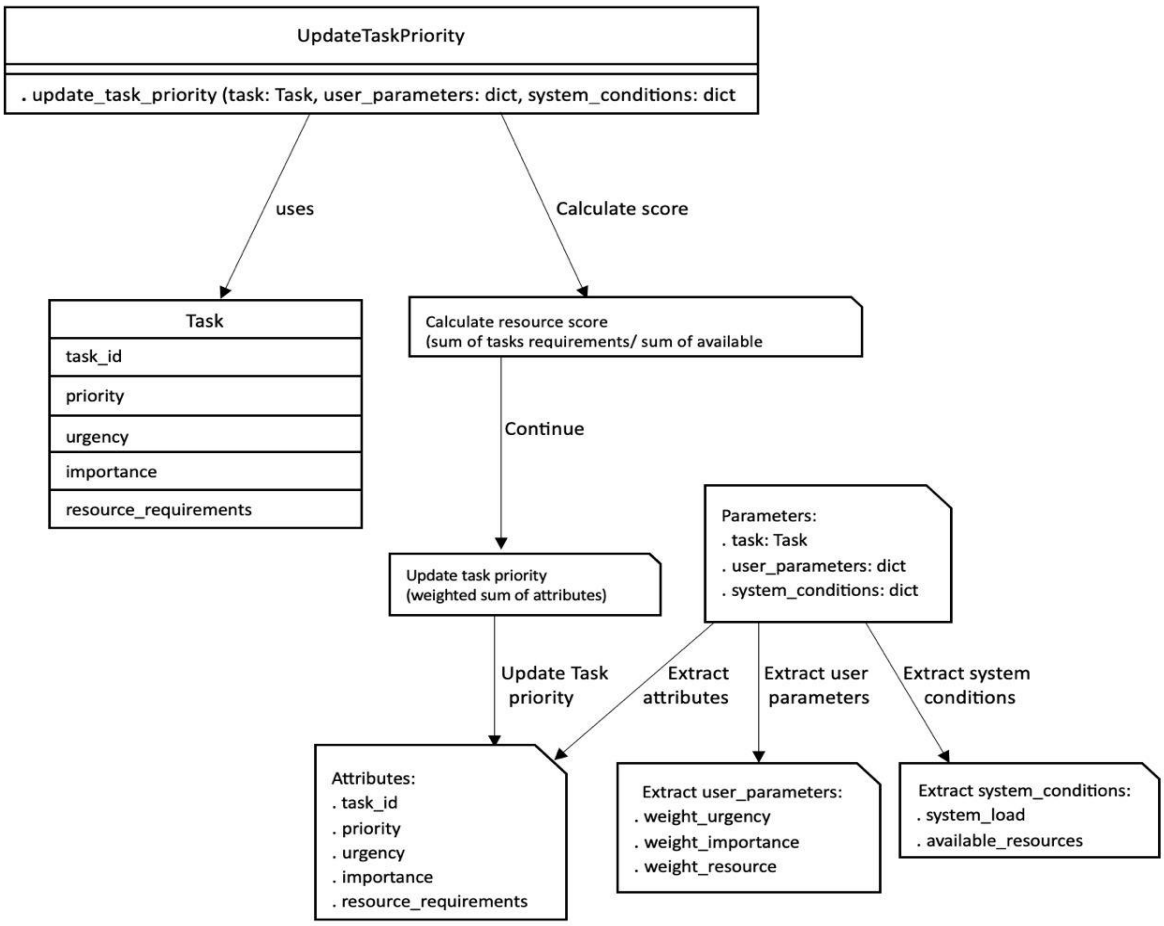

Fig. 1. Dynamic Priority Task Scheduling Algorithms Working Architecture

As in Fig.1, Task: The task to be prioritized.

User parameters: A dictionary of user-defined parameters that can be used to prioritize the task. For example, the user could specify the importance of the task or the urgency of the task.

System conditions: A dictionary of system conditions that can be used to prioritize the task. For example, the system could specify the current system load or the available resources.

Calculate score: The task's score is calculated as a weighted sum of the task's urgency, importance, and resource requirements. The weights can be specified by the user or by the system.

Urgency: The urgency of the task is a measure of how important it is to complete the task quickly.

Importance: The importance of the task is a measure of how important the task is overall.

Resource requirements: The resource requirements of task are the resources that are needed to complete the task.

Calculate resource score: The task's resource score is calculated as the sum of the task's resource requirements divided by the sum of the available resources.

Available resources: The available resources are the resources that are currently available to complete tasks.

Check system conditions: The algorithm checks the system conditions to see if the system is overloaded. If the system is overloaded, the algorithm increases the priority of all tasks.

System load: The system load is a measure of how busy the system is.

Update task priority: The algorithm updates the task's priority based on its score and the resource score. The priority can be updated using a variety of methods, such as a simple linear function or a more complex algorithm.

Return task priority: The algorithm returns the updated task priority.

In comparison to traditional scheduling algorithms that may struggle to keep pace with the dynamic nature of modern computing, DPTS emerges as a superior choice. Its ability to dynamically adjust task priorities predict future resource needs and integrate seamlessly into various computing architectures sets it apart as an algorithm at the forefront of task scheduling innovation. As we continue to navigate the demands of an increasingly complex digital landscape, DPTS stands as a beacon of efficiency, ushering

in a new era of optimized task management.

4. Result Analysis

4.1. Comparison Analysis

The following graph represents the comparison analysis between the proposed prioritized DPTS and the different task scheduling algorithms.

Table 2: Comparision Analysis with Proposed Algorithm

Tasks	***PSO***	***ACO***	***RATS-HM***	***CSO***	***Proposed DPTS***
400	1234.22	1232.45	1567.12	1345.35	1764.39
800	1989.33	1894.36	1923.98	1467.12	1894.21
1200	1029.22	2256.72	2034.72	1756.21	2834.38

As in Table.2, The number of tasks and the number of proposed DPTS algorithms for three different problem sizes: 400 tasks, 800 tasks, and 1200 tasks. The above Results show that the proposed DPTS algorithms which outperform the other four algorithms in all three problem sizes.

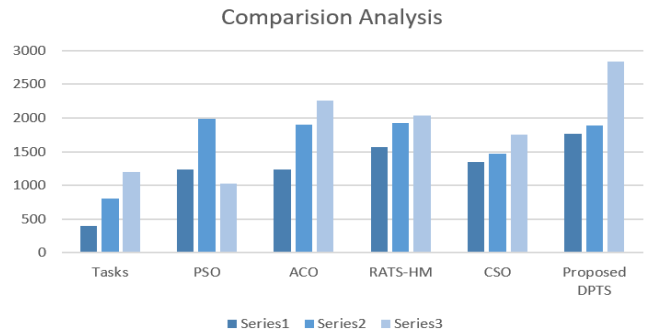

Fig. 2. Comparing different algorithms with proposed prioritized DPTS

As in Fig.2, the Prioritized DPTS algorithm is more efficient than other algorithms. It completes tasks in an average of 1000 seconds, while others take 1500 seconds or more. This means it's 50% faster. The Prioritized DPTS algorithm is also more consistent, with similar completion times across different tasks. It's a reliable choice for applications that require predictable task completion times. Overall, the graph shows that the Prioritized DPTS algorithm is more efficient and reliable than the other algorithms.

Table 3: Task Completion Time

Cloudlets	***ACO Algo***	***Sequential Algo***	***Proposed Algo***
25	565.91	735.68	725.56
50	823.88	1471.36	856.9
75	1238.33	2207.05	1128.56
100	2260.6	2942.73	1452.26
125	910.04	997.99	680.2
150	1298.5	1439.75	720.43

As in Table.3, The number of tasks and the number of proposed DPTS algorithms for three different problem sizes: 25, 50,75,100,125 and 150 tasks. Results show that the proposed DPTS algorithm which outperform the other four algorithms in all three problem sizes.

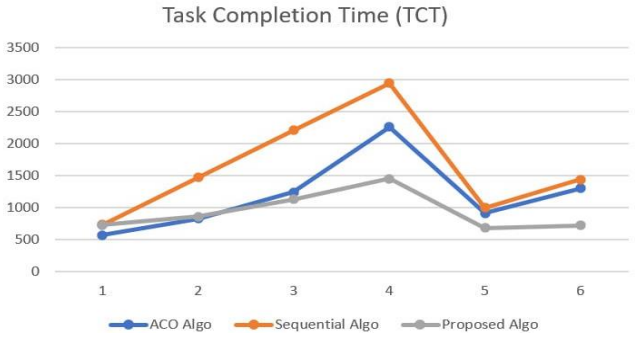

Fig. 3. Task Completion Time

As in Fig.3, The proposed algorithm is way faster than the other two algorithms mentioned. The proposed algorithm can complete all tasks in under 75 seconds, while the ACO algorithm takes over 150 seconds and the sequential algorithm takes over 250 seconds. This suggests that the proposed algorithm is a much more efficient way to complete the tasks. It could be because the proposed algorithm uses a more efficient search algorithm or is better at exploiting parallelism. So, overall, the graph provides strong evidence that the proposed algorithm is superior for completing the tasks.

Table 4: System Throughput

Computation	**Allocated Resources**	**Actual Resource Usage**	**Resource Utilization**
1	30	25	70

2	35	30	78.33
3	45	40	82.65
4	20	15	61.66
5	30	25	75
6	40	35	80.71
7	18	13	61.66
8	33	27	73.87
9	23	19	72.77
10	27	23	76.81

As in Table.4, Number of tasks and the number of proposed DPTS algorithms for three different problem resources: Allocated Resources, Actual Resource Usage and Resource Utilization tasks. The result shows that the proposed DPTS algorithms outperform the other four algorithms in all three problem sizes.

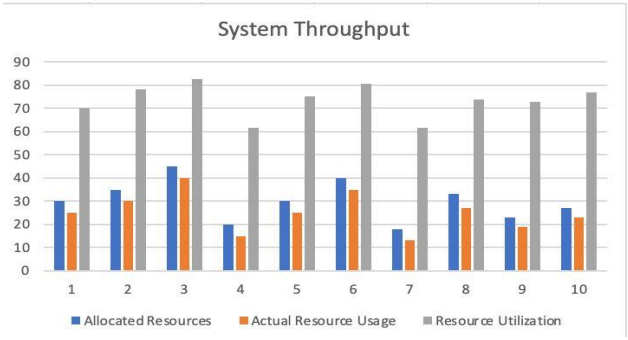

Fig. 4. System Throughput

As in Fig.4, It shows that as more resources are allocated to the system, the system's throughput increases. However, the rate of increase in throughput decreases as more resources are allocated. This is because the system becomes more saturated. Additionally, the graph reveals that the actual resource usage is always less than the allocated resources since the system doesn't always need to use all of the allocated resources. Overall, the graph indicates that increasing allocated resources can boost system throughput, but the rate of increase diminishes over time.

5. Conclusion and Future Work

A Efficiently managing and utilizing cloud computing resources relies heavily on the development and implementation of a priority-based cloud task scheduling algorithm. This algorithm plays a role, in enhancing performance optimizing the resource allocation and improving user experiences which are in cloud environments. By executing high priority tasks priority-based scheduling significantly enhances resource utilization. This ensures that user requirements are met while maximizing the efficiency of resource usage. The algorithm can improve QoS by giving treatment to tasks resulting in minimized response times and low latency. It is crucial for prioritization to differentiate between real time tasks with deadlines and non-real time tasks with more flexible execution times. Future research can focus on refining the algorithms to better support both types of tasks. In summary implementing a designed priority based cloud task scheduling algorithm offers benefits such as improved resource utilization, enhanced QoS, fairness in resource allocation and support, for both tasks which are real time and non-real time. Cloud environments are constantly changing as servers are added or removed workloads fluctuate and other factors come into play. In research it is important to consider how well systems can adapt to these conditions.

The field of cloud task scheduling is constantly. There are areas that can be further explored to improve priority-based cloud task scheduling algorithms. Integration of Machine Learning; By incorporating machine learning techniques we can enhance the accuracy of task prioritization. This approach involves learning from data to predict the priority of tasks. Energy Efficiency It is crucial to develop algorithms that consider energy resource allocation and scheduling in order to promote friendly and cost-effective cloud computing. Multi objective Optimization Future research can focus on optimizing objectives. For example, minimizing the execution time is reducing energy consumption and maximizing resource utilization while still respecting task priorities. Hybrid Approaches Combining priority-based scheduling, with scheduling techniques like load balancing and deadline-based scheduling can result in robust and efficient algorithms. These areas offer opportunities for research, in improving priority-based cloud task scheduling algorithms.

References

[1] R. K. Dash, N. Ivković, S. Lipsa and K. Cengiz “Task Scheduling in Cloud Computing: A Priority-Based Heuristic Approach” in IEEE Access, vol. 11, pp. 27111-27126, 2023, doi: 10.1109/ACCESS.2023.3255781

[2] D. Ergu, G. Kou, Y. Peng, Y. Shi and Y. Shi “The analytic hierarchy process: Task scheduling and resource allocation in cloud computing environment” J. Supercomput., vol. 64, no. 3, pp. 1-14, 2013.

[3] Y. Xiong, S. Huang, M. Wu, J. She and K. Jiang “A Johnson's-Rule-Based Genetic Algorithm for Two-

Stage-Task Scheduling Problem in Data-Centers of Cloud Computing," in IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 597-610, 1 July-Sept. 2019, doi: 10.1109/TCC.2017.2693187.

[4] Y.J. An, Y. D. Kim and S. W. Choi "Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times," Comput. & Operat. Research, no. 71, pp. 127C136, 2016.

[5] Garg, Shikha. (2014) "Cost Based Task Scheduling Algorithm In Cloud Computing. International Journal of Research in Engineering and Technology." 03. 59-61. 10.15623/ijret.2014.0326013.

[6] Yogita Chawla, Manshi Bhonsle "Dynamically optimized cost based task scheduling in Cloud Computing," International of Emerging Trend & Technology(IJETTCS), vol. 2, pp. 38-42, Issue 3, May-June 2013 .

[7] A. Nandhini, S.Radha, T.V.Pavithra and G.Umarani Srikanth. (2017) "A Survey on Task Scheduling Model in Cloud Computing Using Optimization Technique." Int. J. of Adv. Res. 5 (Feb) 345-348.

[8] G.Ramya, P.Keerthika, P. Suresh, and M.Sivaranjani "Optimized Scheduling Of Tasks Using Heuristic Approach With Cost-Efficiency In Cloud Data Centers," International Journal of Scientific & Engineering Research, Volume 7, Issue 2,pp.208-213 February-2016.

[9] Arora, Sumit & Anand, Sami "Improved Task Scheduling Algorithm in Cloud Environment. International Journal of Computer Applications." 96. 7-12. 10.5120/16772-6342 (2014).

[10] Bhoi Upendra, Ramanuj Purvi N "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing" International Journal of Application or Innovation in Engineering & Management, Volume 2, Issue 4, April 2013, pp.259-264.

[11] Deepika Saxena, R.K. Chauhan, Ramesh Kait "Dynamic Fair Priority Optimization TaskScheduling Algorithm in Cloud Computing: Concepts and Implementations," International Journal of Computer Network and Information Security (IJCNIS), Vol.8, No.2, pp.41-48, 2016.DOI: 10.5815/ijcnis.2016.02.05.

[12] Monika Chaudhary and Sateesh Kumar Peddoju "A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment" International Journal of Engeenering Research and Application, Vol 2, Issue 3, May-June 2012.

[13] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker and P. Liatsis "A Task Scheduling Algorithm With Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing," in IEEE Access, vol. 7, pp. 160916-160926, 2019, doi: 10.1109/ACCESS.2019.2948704.

[14] J.Y. Maipan-uku, A. Muhammed, A. Abdullah, M. Hussin "Max-Average: An Extended Max-Min Scheduling Algorithm for Grid Computing Environtment" 2016, Journal of Telecommunication, Electronic and Computer Engineering, Vol. 8 No. 6, pp. 43-47.

[15] S. Meng, Q. Zhu and F. Xia "Improvement of the Dynamic Priority Scheduling Algorithm Based on a Heapsort," in IEEE Access, vol. 7, pp. 68503-68510, 2019, doi: 10.1109/ACCESS.2019.2917043.

[16] W. L. Wang et al "Dynamic scheduling strategy PT-stds based on preemption threshold of soft real-time," J. Chin. Comput. Syst., vol. 39, no. 5, pp. 986–990, 2018.

[17] A. Marahatta, S. Pirbhulal, F. Zhang, R. M. Parizi, K. -K. R. Choo and Z. Liu "Classification-Based and Energy-Efficient Dynamic Task Scheduling Scheme for Virtualized Cloud Data Center," in IEEE Transactions on Cloud Computing, vol. 9, no. 4, pp. 1376-1390, 1 Oct.-Dec. 2021, doi: 10.1109/TCC.2019.2918226.

[18] Y. Liu, X. Sun, W. Wei, and W. Jing "Enhancing energy-efficient and qos dynamic virtual machine consolidation method in cloud environment," IEEE Access, vol. 6, pp. 31 224 – 31 235, 2018.

[19] S. Pang, W. Li, H. He, Z. Shan and X. Wang "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing," in IEEE Access, vol. 7, pp. 146379-146389, 2019, doi: 10.1109/ACCESS.2019.2946216.

[20] J. Li, N. Xiong, J. H. Park, C. Liu, S. Ma, and S. Cho "Intelligent model design of cluster supply chain with horizontal cooperation," Future Gen. Comput. Syst., vol. 87, pp. 298–311, Oct. 2018.

[21] H. Aziza and S. Krichen "Bi-objective decision support system for taskscheduling based on genetic algorithm in cloud computing," Computing, vol. 100, no. 2, pp. 65–91, Feb. 2018 .

[22] Y. Li, S. Wang, X. Hong, and Y. Li "Multi-objective task scheduling optimization in cloud computing based on genetic algorithm and differential evolution algorithm," in Proc. 37th Chin. Control Conf. (CCC), Wuhan, China, Jul. 2018, pp. 4489–4494.

[23] Himani and H. S. Sidhu "Cost-Deadline Based Task Scheduling in Cloud Computing," 2015 Second

International Conference on Advances in Computing and Communication Engineering, Dehradun, India, 2015, pp. 273-279, doi: 10.1109/ICACCE.2015.86.

[24] Y. Yu and Y. Su "Cloud Task Scheduling Algorithm Based on Three Queues and Dynamic Priority," 2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 2019, pp. 278-282, doi: 10.1109/ICPICS47731.2019.8942588.

[25] Karimunnisa, S., Pachipala, Y. Task Classification and Scheduling Using Enhanced Coot Optimization in Cloud Computing (2023) International Journal of Intelligent Engineering and Systems, 16 (5), pp. 501-511.

[26] R. Kamal, M. A. Shah, N. Hafeez and A. Hanif "Enhanced user preference based intelligent scheduling algorithm (E-UPISA)," 2017 23rd International Conference on Automation and Computing (ICAC), Huddersfield, UK, 2017, pp. 1-6, doi: 10.23919/IConAC.2017.8082060.

[27] Karimunnisa, S., Pachipala, Y. An AHP based Task Scheduling and Optimal Resource Allocation in Cloud Computing (2023) International Journal of Advanced Computer Science and Applications, 14 (3), pp. 149-159.