Efficient Dynamic Heuristic Earliest Finnish Scheduling Algorithm for Load balancing in cloud computing

[1]Priya M M.E, [2]Arunachalam K, [3]Kirubakaran S,[4]Ramalingam M, [5]Yuvaraja K
[1]*Assistant Professor, Department of Computer Science Engineering, Tagore Engineering college, Deviyakurichi*
[2,3,4,5.]*UG Students, Department of Computer Science Engineering, Tagore Engineering college, Deviyakurichi*

***Abstract*—Cloud computing, serving as the dominant paradigm for large-scale infrastructure provisioning, presents significant challenges in service and task allocation. Cloud computing is the largest constraints service providing environment for large scale infrastructure. Many more request on service allocation and task allocating is complex because of improper scheduling leads poor computing and non-balanced cost efficiency. The inherent complexity of managing numerous simultaneous requests frequently results in inefficient scheduling, To resolve this problem, Dynamic Heuristic Earliest Finish scheduling algorithm (DHEFSA) is applied for balance the load to improve the computing resource. Initially the task computing time is based on Absolute Mean time (AMT) is estimated and rank eth priority task. Next to priority. Next to crate the Minmax priority pattern (MMPP) based on the time task to complete time to allocate the task finally the Dynamic Heuristic Earliest Finish scheduling algorithm (DHEFSA) allocates the workload to migrate the resource response and complete the workload efficiently. This approach is anticipated to yield superior computational performance, improved scheduling accuracy, and a more balanced workload distribution when compared to existing scheduling methodologies. The proposed system produces higher performance in computing to improves scheduling accuracy to balance the workload compared to the existing system.**

***Keywords*—Cloud Computing, Resource Allocation, Task Scheduling, Heuristic Algorithm, Load Balancing, Dynamic Scheduling, AMT, DHEFSA.**

I. INTRODUCTION

Cloud computing has emerged as the dominant paradigm for large-scale infrastructure provisioning, offering on-demand access to a vast array of computing resources. This ubiquitous accessibility, however, presents significant challenges in efficient resource management. The dynamic and heterogeneous nature of cloud environments, coupled with the ever-increasing volume of service requests, makes service allocation and task scheduling exceedingly complex. Improper scheduling strategies often lead to suboptimal resource utilization, manifesting as poor computing performance, imbalanced workloads, and ultimately, reduced cost efficiency. This introduction focuses on the crucial role of effective scheduling algorithms in mitigating these challenges, specifically introducing the Dynamic Heuristic Earliest Finish Scheduling Algorithm (DHEFSA) as a promising approach to balance load, improve computing resource utilization, and enhance overall performance. The core challenge in cloud resource allocation lies in the need to dynamically assign tasks to virtual machines (VMs) while minimizing completion time and optimizing resource consumption. Traditional scheduling algorithms, designed for static and homogeneous environments, often fall short in addressing the inherent variability and heterogeneity of cloud infrastructure. Consequently, the need for intelligent and adaptive scheduling strategies is paramount. Keywords such as cloud computing, large-scale infrastructure, service allocation, task scheduling, resource management, performance optimization, and cost efficiency are all central to understanding the scope and importance of this problem.

The DHEFSA algorithm, proposed as a solution to this intricate scheduling problem, leverages a multi-phased approach to effectively distribute workload across available resources. The initial phase focuses on estimating the task computing time using an Absolute Mean Time (AMT) metric. This AMT calculation serves as a baseline for understanding the computational requirements of each task, allowing for a more informed prioritization process. Accurate estimation of task completion time is crucial for effective scheduling, as it allows the system to anticipate resource demands and proactively avoid bottlenecks. Furthermore, the system incorporates a prioritization mechanism to rank tasks based on their importance and urgency. This prioritization ensures that critical tasks are allocated resources promptly, minimizing delays and maximizing overall system responsiveness.

Building upon the task prioritization, the algorithm then constructs a Minmax priority pattern based on the estimated task completion times. This pattern aims to minimize the maximum completion time of any individual task while simultaneously maximizing the minimum completion time across all tasks. This strategy ensures that no single task disproportionately consumes resources and delays others, thereby fostering a more equitable and balanced resource distribution. The Minmax pattern effectively addresses the potential for starvation and ensures that all tasks progress towards completion in a timely manner. The concept of priority scheduling, combined with the Minmax pattern, represents a significant advancement in ensuring fairness and efficiency in resource allocation.

The final phase involves the application of the DHEFSA itself. This algorithm dynamically allocates workload to VMs based on the previously established priorities and estimated completion times. It takes into account the current resource utilization of each VM and intelligently migrates workloads to ensure a balanced distribution across the infrastructure. The Dynamic Heuristic nature of the algorithm allows it to adapt to changing conditions and dynamically adjust the scheduling strategy to optimize performance in real-time. This adaptability is crucial in dynamic cloud environments where resource availability and task requirements fluctuate constantly. The algorithm's objective is to minimize the overall execution time of the workload while ensuring efficient resource utilization and minimizing the risk of resource contention. The focus on load balancing and resource response is fundamental to the success of DHEFSA.

By effectively distributing the workload and intelligently allocating resources, the DHEFSA promises to significantly improve the performance of cloud computing environments. The proposed system is expected to exhibit higher computing performance, improved scheduling accuracy, and a more balanced workload distribution compared to existing scheduling algorithms. This translates to reduced execution times, lower operational costs, and improved overall system reliability. The anticipated improvements in scheduling accuracy are particularly important, as they directly impact the efficiency of resource utilization and the responsiveness of the system to user requests. Ultimately, the DHEFSA aims to provide a more robust and efficient platform for deploying and executing large-scale applications in the cloud.

II. BACKGROUND STUDY

Atakan Aral et al: This work provides a general overview of how to handle resource allocation problems in Inter-Clouds, Mobile Clouds and related solutions. Every cloud data center needs new ideas to handle the task of assigning computing and network resources to cloud services.

Dimitri Mazmanov et al: The use of cloud computing supports less cost and more flexibility when deploying three-tier web and video streaming applications, compared to in-house systems. These sorts of applications usually have clear and easily understood requirements for performance. Strict specifications for computation, storage and network resources are necessary for enterprise use-cases to be used with reasonable spending. They are written into agreements called Service Level Agreements (SLAs) between the application and the cloud provider. In addition, these distributed platforms introduce new ways of setting up cloud resources for highly responsive applications

Ahmad Anbar et al: The use of cloud computing is increasingly dominating science computing. However, some challenges related to such applications must be solved, before they can use the cloud as their IT platform. Using a Partitioned Global Address Space (PGAS) language based on Distributed Shared Memory (DSM) will assist in the adoption process. The initial use of a PGAS language, Unified Parallel C, to program a representative private cloud system based on Eucalyptus is looked at in this paper.

Matthew B et al: Over the years, the device people choose for capturing multimedia has become more often a smartphone. It is simple to find smartphones and use them to take quality photos and videos. Because of this, cloud storage is seeing more use. Most people keep their files with a single storage service. They require a solution that will respond instantly, be reliable all the time and be safe. Filling your site with content from various sources can make it less expensive and more effective. Distributed Indexed Storage in the Cloud (DISC) is offered as a framework to create a better cloud service for customers.

Nishant Nikam et al: Cloud computing makes it possible for users (clients) to transfer big chunks of their data to cloud servers. Cloud storage systems that protect data transfer them to different servers so that they can be safely and uninterrupted. Their approach suggests constructing this scheme by organizing data into data blocks, encoding them (with error correction) and placing authentication information (tags) on each encoded block.

Ling Liu et al: By first evaluating Hadoop performance in multi-datacenter and single-datacenter environments, they noticed that having data stored and processed in more than one center can cause delays which makes geographically distributed Cloud architecture and hybrid Cloud services necessary to support unexpected surges in demand. In addition, the characterization of the problem as seen in geographically spread cloud data centers is presented along with general optimization strategies.

Zhongni Zheng et al: Infrastructure as a Service cloud relies heavily on the process of resource scheduling. They suggest a scheduling algorithm that helps make the best use of resources for cloud scheduling problems. They look into placing Virtual Machines wherever possible to help find the best allocation quickly, as long as resources are put to maximum use. In mathematical terms, they regard the scheduling problem as being equivalent to an Unbalance Assignment Problem. By using the Parallel Genetic Algorithm, our scheduling policy was achieved more quickly than with an ordinary Genetic Algorithm. Our method, as tested in experiments, resulted in faster resource allocation and better use of the system’s resources.

K. Chandrasekaran et al: Many people turn to Map-Reduce to handle the analysis of large amounts of data using multiple machines. Data scientists are more often deploying Map-Reduce in the cloud together with other applications that make use of the same equipment. In such a circumstance, the success of the Map-Reduce tasks greatly depends on how efficiently they are scheduled. Besides being fast, Map-Reduce also takes into account energy usage and hitting SLA targets during job execution in clouds. In their analysis, these authors group Map-Reduce Scheduling within Cluster based Scheduling and Objective based Scheduling.

Ting He et al: The authors tackle the situation where low-priority work is done on computers in the cloud that are left open after high-priority jobs are finished. When high-priority functions are taken up by other servers, the best option for low-priority ones is to suspend what they are doing or transfer them to other open servers. Opportunistic scheduling seeks to schedule less important work onto available servers periodically and with the lowest total expense of waiting and migrating those tasks. In addition, they are designed to cope with running several lower priority jobs that must be synchronized. Assuming servers’ ability to perform low-priority tasks can be modeled as ON/OFF Markov chains, the team proved that solving the problem optimally is difficult because it involves solving an Markov Decision Process (MDP) that is exponentially complex and the best approach works only when all servers have the same behavior.

Zhang Xiaoqing et al: A cloud computing system relies heavily on the tasks scheduling problem as its main hurdle. In order to minimize expenditure when scheduling workflow tasks before the deadline and the budget, we suggest an algorithm for workflow tasks scheduling that uses a genetic algorithm in cloud computing. For each action in our algorithm, a top-down leveling method is used to assign priority. By doing this, each workflow task is assigned a level which allows different tasks to be worked on in parallel. The method for coding tasks scheduling tasks is a simple two dimension approach.

III. PROPOSED SYSTEM

The proposed system operates through a series of well-defined steps, beginning with the estimation of task computing time using the Absolute Mean Time (AMT) method. AMT provides a baseline for understanding the computational demands of each task, considering factors such as task size, complexity, and resource requirements.

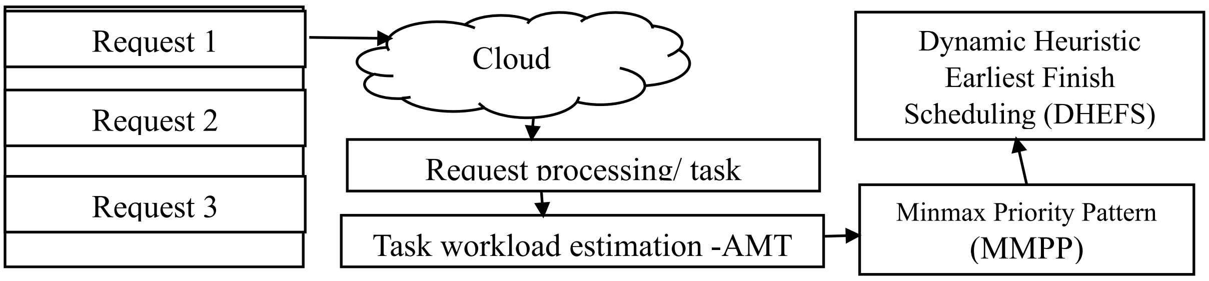
Request 1
Request 2
Request 3
Cloud
Request processing/ task
Task workload estimation -AMT
Minmax Priority Pattern (MMPP)
Dynamic Heuristic Earliest Finish Scheduling (DHEFS)

Figure 1: Proposed workflow Diagram DHEFS

This initial estimation is crucial for prioritizing tasks and making informed scheduling decisions. The system's performance is anticipated to surpass existing scheduling methodologies in several key areas. First, the AMT-based task computing time estimation provides a more accurate foundation for scheduling decisions, leading to improved resource utilization. Figure 1 show the Proposed workflow Diagram DHEFS. Second, the MMPP ensures a balanced workload distribution, preventing bottlenecks and maximizing system throughput. Third, the DHEFSA's dynamic adaptation to real-time system conditions allows for optimized resource allocation and efficient task completion. This holistic approach to scheduling promises superior computational performance, enhanced scheduling accuracy, and a more balanced workload distribution, ultimately resulting in improved cloud computing efficiency and cost-effectiveness.

Absolute Mean Time (AMT)

Following the AMT estimation, the system ranks tasks based on their priority. This prioritization is determined by a combination of factors, including the urgency of the task, its impact on overall system performance, and any predefined service level agreements (SLAs). High-priority tasks are given precedence in the scheduling process to ensure that critical operations are completed in a timely manner. Absolute Mean Time Estimation is a cloud-based scheduling strategy that maximizes task allocation by estimating a typical execution time for tasks on each available virtual machine. In this operation, when a task is scheduled, the execution time of the task is estimated on all of the mean execution time is calculated, which is then used for scheduling purposes.

$$MET_i = \frac{1}{m}\sum_{j=1}^{m} ET_{ij} \qquad (7)$$

Where MET_i represents mean execution time for task, and m is total number of virtual machines, ET_{ij} estimated execution time of task.

$$AMTD_{ij} = |ET_{ij} - MET_i| \qquad (8)$$

Where $AMTD_{ij}$ is absolute deviation of the task's execution time, and ET_{ij} is estimated execution time of task, and MET_i is mean execution time of task.

$$TPI_i = \frac{1}{MET_i + \epsilon} \qquad (9)$$

Where TPI_i represents priority score for task i, tasks with lower MET get higher priority, and ϵ is a small constant to division by zero, useful for scheduling order tasks need to be prioritized

Minmax Priority Pattern (MMPP)

Next, the system creates a Minmax Priority Pattern (MMPP) based on the estimated task completion time. This pattern strategically allocates resources to minimize the maximum completion time of any task, thereby reducing overall system latency and improving responsiveness. The MMPP ensures that no single task monopolizes resources, preventing bottlenecks and maintaining a balanced workload distribution. The Min-Min Max-Min Priority Algorithm is a cloud-based scheduling technique intended to improve the efficiency of task execution in distributed computing environments by blending the Min-Min and Max-Min approaches based on task priority. The tasks are analyzed first regarding estimated completion time across all virtual machines.

$$ECT_{ij} = RT_j + ET_{ij} \qquad (5)$$

Where ECT_{ij} represents is estimated completion time of task, and RT_j is ready time of tasks, ET_{ij} is estimated execution time of task. Select the task with minimum ECT and assign it to the corresponding.

$$ET_{ij} = \frac{T_i}{P_j} \qquad (6)$$

Where ET represents is execution time of task, T_i represents the volume quantity of work that needs to be processed, P_j and indicates the computational strength performance efficiency of a processing unit.

Dynamic Heuristic Earliest Finish Scheduling Algorithm (DHEFSA)

Finally, the Dynamic Heuristic Earliest Finish Scheduling Algorithm (DHEFSA) is applied to allocate the workload and migrate resources efficiently. DHEFSA dynamically adjusts the scheduling strategy based on real-time system conditions, such as resource availability, task dependencies, and workload fluctuations. This adaptive approach enables the system to respond effectively to changing demands and optimize resource allocation for maximum performance. DHEFSA aims to schedule each task to the resource that allows it to finish at the earliest possible time, considering current workload and resource capabilities. The "dynamic heuristic" component suggests that the algorithm uses rules of thumb that are adjusted as the system runs. The Dynamic Heuristic Earliest Finish Scheduling Algorithm is a cloud-based scheduling algorithm that is aimed at facilitating optimal resource task execution in an environment of distributed computing. Based on heuristic evaluations, the algorithm dynamically assigns tasks to virtual machines to minimize the makes pan or the total amount of time to perform all the tasks. The earliest finish time is calculated for each task across all the VMs and accounts for task dependencies, available resources, and execution costs. The DHEFS Scheduling Algorithm dynamically prioritizes the functions that can be completed as soon as possible while also considering improvements on each decision based on the system's current state. The DHEFS Scheduling Algorithm aims to realize better resource utilization and reduce system latency.

$$Avearge\ Service\ Time\ E(S) = \frac{\sum_{l=0}^{n} Si}{n} \quad (1)$$

Where $E(S)$ is average service time of tasks, Si is service time for the task, and n is total number of tasks.

$$Service\ rate\ \mu = \frac{1}{E(s)}\mu = \frac{1}{E(s)} \quad (2)$$

Where μ service rate of the system is, $E(S)$ is average service time. It indicates many tasks a server can handle per unit time.

$$p_o = \left[\sum_{n=0}^{s-1} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{1}{s!}\left(\frac{\lambda}{\mu}\right)^2 \left(\frac{s\mu}{s\mu-\lambda}\right)^{-1}\right] \quad (3)$$

Where p_o probability that there are zero tasks in the system is, λ represents arrival rate of tasks, and μ is service rate, s is number of servers. Used to assess the probability that the system is not processing any task.

$$L_q = \left[\frac{1}{(s-1)!*}\left(\frac{\lambda}{\mu}\right)^s * \left(\frac{\lambda\mu}{(s\mu-\lambda)^2}\right)\right] * p_o \quad (4)$$

Where L_q represents average number of tasks waiting in the queue, $\frac{1}{(s-1)!*}$ as defined above, this determines the expected queue length, evaluate efficiently the system manages task backlogs.

IV. RESULT AND DISCUSSION

The section demonstrates system performance by estimating maximum request capacity for the server supported by energy-aware routing, which improves scheduling efficiency while optimizing resource utilization. The evaluation analyzes the proposed scheduling framework, which combines Max-Min execution time evaluation and PP-BQS to handle critical user tasks when operating in dynamic cloud environments. Through this framework, researchers can perform an extensive performance review that examines scheduling technique efficiency levels. A performance comparison between the proposed model and conventional Multilevel Queue Scheduling (MQS), Multilevel Feedback Queue Scheduling (MFQS), Pre-emptive Scheduling algorithm (PSA) methods occurs. The study employs evaluation metrics that assess time complexity, energy consumption, delay tolerance, resource allocation, and self-scheduling performance. The proposed method achieves optimized workload distribution, which leads to reduced execution times and better overall cloud system responsiveness.

Table 1 Simulation Parameter Description

Parameter	Variable
Used Simulator	VS.net framework
Task assigning -workload	Request / response
Virtual machine-Migration	5
Single iteration	500 requests
Energy balancing mode	Switch over

A C# language-based simulator, developed using Visual Studio 6.0, generates simulation results that are compared to analytical performance based on the parameters of the DHEPS algorithm. The simulator is designed for self-programming performance analysis. The Load Balancer (LB) also optimizes User Request (UR) performance by directing incoming requests during scheduled sessions through its self-scheduling feature.

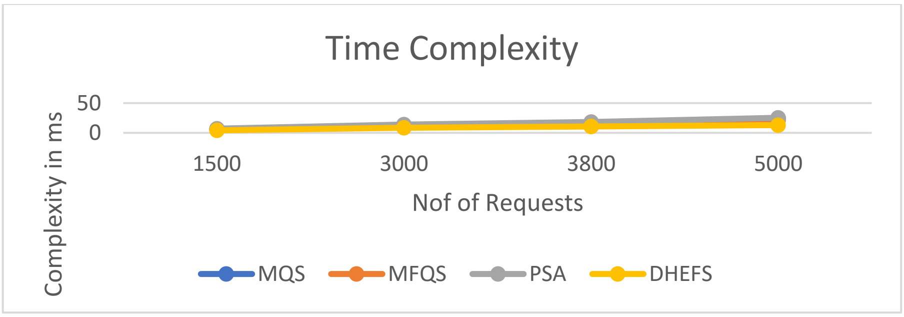

Figure 2. Analysis of Time Complexity in ms

The figure 2 shows how scheduling methods consume processing time when processing a growing number of requests. Execution times of the MQS method demonstrate increasing complexity until it reaches around 24.8 milliseconds with 5000 concurrent requests. The DHEFS technique delivers equivalent computational complexity levels when studying values from 13.1 ms to those measured in the PSA method. The most extensive solution comes from the collection of methods being evaluated. The proposed DHEPS method reduces execution duration to the minimal level thus it ensures both time efficiency and superior request speed.

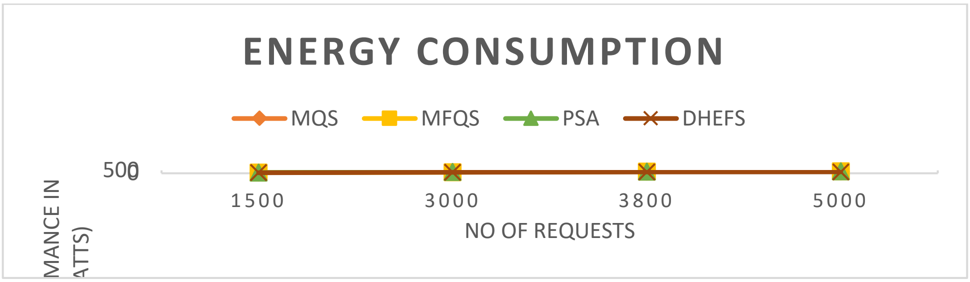

Figure 3. Analysis of Energy Consumption in (Watts)

Various scheduling techniques are evaluated for their energy consumption, as shown in Figure 3. The entire system consumes growing amounts of energy when requested operations increase. DHEPS requires the least power consumption among the scheduling methods, but MQS and PSA methods exhibit increased power usage. The DHEPS approach provides an optimized 250-watt balance through efficient workload management. It maintains lower energy consumption than existing methods to deliver an energy-efficient solution for cloud scheduling.

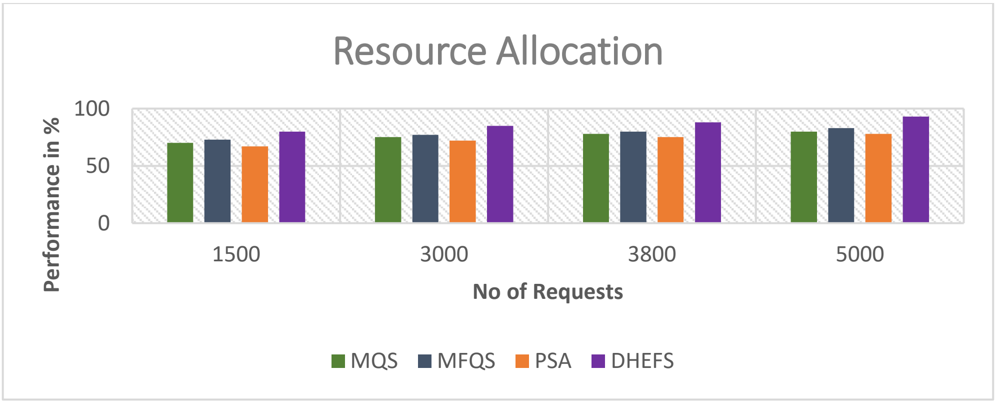

Figure 4. Analysis of Resource Allocation in %

Figure 4 shows the resource allocation efficiency of scheduling techniques. Research indicates that the DHEPS method performs better than the MQS and MFQS methods at allocating resources efficiently, with more than 93% success during 5000 request simulations. DHEPS resource utilization peaks when empty resources are eliminated while cloud workloads are distributed evenly throughout environments.

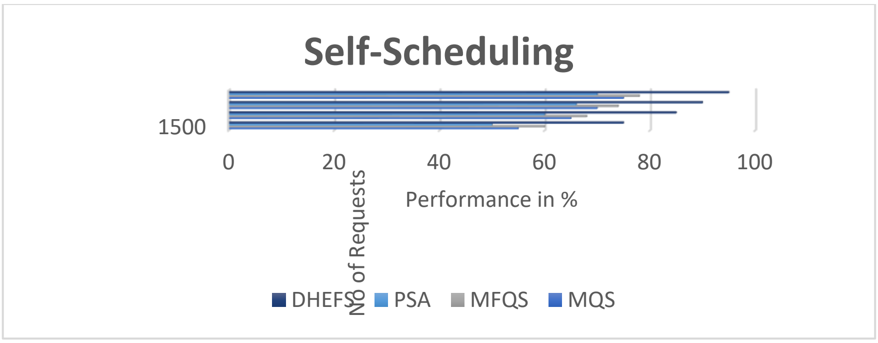

Figure 5. Analysis of Self-Scheduling in %

Figure 5 shows the performance of different scheduling techniques in self-scheduling. The scheduling results delivered by the DHEPS approach surpass other similar methods because it achieves 95% efficiency at 5000 request specifications. MCDM PSO-GA and MOTS's self-scheduling performance is below those of other methods. Through its task assignment management capabilities, the DHEPS method successfully executes cloud operations by minimizing operational execution delays.

V. CONCLUSION

To conclude that the proposed system, leveraging the Dynamic Heuristic Earliest Finish Scheduling Algorithm (DHEFSA), offers a comprehensive solution to the challenges of service and task allocation in cloud computing environments. By combining AMT-based task estimation, priority-based ranking, MMPP workload distribution, and dynamic heuristic scheduling, the system aims to achieve superior computational performance, improved scheduling accuracy, and a more balanced workload distribution, thus enhancing the overall efficiency and cost-effectiveness of cloud infrastructure.

REFERENCES

[1] Atakan Aral "Modeling and Optimization of Resource Allocation in Distributed Clouds," , IEEE 2016.
[2] Dimitri Mazmanov, Calin Curescu, Hjalmar Olsson, Andrew Ton, James Kempf. "Handling Performance Sensitive Native Cloud Applications with Distributed Cloud Computing and SLA Management," IEEE 2013.
[3] Ahmad Anbar, Vikram K. Narayana, and Tarek El-Ghazawi, "Distributed Shared Memory Programming in the Cloud," IEEE 2012.
[4] Matthew B. Hancock, Carlos A. Varela,"Augmenting Performance for Distributed Cloud Storage," IEEE 2015.
[5] Binanda Sengupta, Nishant Nikam, Sushmita Ruj, Srinivasan Narayanamurthy, Siddhartha Nandi, "An Efficient Secure Distributed Cloud Storage for Append-only Data," IEEE 2018.
[6] Qi Zhang, Ling Liu, Kisung Lee, Yang Zhou, Aameek Singh, Nagapramod Mandagere, Sandeep Gopisetty, "Improving Hadoop Service Provisioning in A Geographically Distributed Cloud," IEEE 2014.
[7] Zhongni Zheng, Rui Wang, Hai Zhong, Xuejie Zhang, "An Approach for Cloud Resource Scheduling Based on Parallel Genetic Algorithm," IEEE 2011.
[8] Sofia D'Souza, K. Chandrasekaran, "Analysis of MapReduce Scheduling and Its Improvements in Cloud Environment," IEEE 2015.
[9] Ting He, Shiyao Chen, Hyoil Kim, Lang Tong, and Kang-Won Lee,"Scheduling Parallel Tasks onto Opportunistically Available Cloud Resources," IEEE 2012.
[10] Yang Cui, Zhang Xiaoqing,"Workflow Tasks Scheduling Optimization Based on a Genetic Algorithm in Clouds," IEEE 2018.