Dynamic Priority-Based Round Robin: An Advanced Load Balancing Technique for Cloud Computing

Parupally Venu, Pachipala Yellamma*, Yama Rupesh, Yerrapothu Teja Naga Eswar, Maruboina Mahiddar Reddy
Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation,
Vaddeswaram, Guntur 522302, India

***Abstract*—An imbalance of load is an essential problem in cloud computing where the division of work between virtual machines is not well-optimized. Performance bottlenecks result from this unequal resource allocation, which keeps the system from operating at its full capability. Managing this load-balancing issue becomes critical to improving overall efficiency, resource utilization, and responsiveness as cloud infrastructures strive to respond to changing workloads and scale dynamically. Crossing the load-balancing landscape introduced a new strategy to effectively improve the load-balancing factor and ways to improve load-balancing performance by understanding how existing algorithms work, an effective method of load balancing. The "Dynamic Priority Based Round Robin" algorithm is a new approach that combines three different algorithms to improve cloud load balancing. This method improves load balancing by taking the best aspects of previous algorithms and improving them. It works remarkably well and responds quickly to commands, greatly reducing processing time. This DPBRR algorithm also plays an important role in improving cloud load balancing in many ways, including improving resource consumption, inefficiency, scalability, fault tolerance, cost optimization, and other aspects. Since it is a combination of algorithms, it may have its drawbacks, but its cloud computing enhancements are very useful for doing many tasks quickly. Strength and adaptability are quite effective, as is adaptability.**

Keywords—Load balancer; traffic distribution; cloud computing; resource utilization; scalability; Dynamic Priority Based Round Robin (DPBRR)

I. Introduction

A cutting-edge paradigm in information technology, cloud computing marks a radical shift in the direction of the democratization of data access and decentralization of computational capacity. This cutting-edge technical framework breaks through conventional barriers by providing individuals and businesses with access to an infinite resource, from powerful computer power to endless storage. Cloud computing is a fascinating and amazing computing era that has replaced traditional methods. Cloud computing eliminates this necessity and offers services based on user demand and usage, in contrast to traditional computing, which requires users to maintain internal infrastructure. Users are spared from having to buy and maintain processing, storage, and other hardware. Through the internet, data is accessed and stored [1]. These days, cloud computing is essential because it offers pay-as-you-go on-demand services. To provide high-quality services, suppliers are taking advantage of service models like SaaS, PaaS, IaaS. Over the past five years, the public cloud computing markets have grown significantly, expanding by 21.5% [4].

*Corresponding Author.

Workload control is essential for load balancing problems since job arrival patterns are unpredictable and cloud node capacity varies. It also helps to preserve system stability. Schemes for load balancing can be either static or dynamic, depending on how essential System dynamics are [11]. Load balancing is a technique used in distributed systems to distribute the workload among multiple resources. As a result, there is an increase in scalability and effective resource usage. Multiple networked computing devices that collaborate are part of distributed systems. The need for efficient resource allocation grows as workload demand rises. This problem is solved by load balancing, which divides up incoming requests or tasks among the servers, virtual machines, or containers that are available. The task load is typically divided among several virtual machines (VM’s) when using the load balancing technique. Without it, there could be SLA violations, assert wastage, execution corruption, and uneven burdens during server construction. Thus, employing an appropriate load balancing technique can enhance server utilization and provide more Quality of Service (QoS) assurance [12]. In a cloud environment, load balancing is accomplished in two steps: first the job is divided among the nodes; second, the virtual machine is monitored, and load balancing activities are carried out using task migration or virtual machine migration approach [5]. Load balancing in the cloud occurs in two stages: first, at the level of physical machines, where the load balancer distributes the load among the VMs connected to each physical device and manages the load of physical machines; second, at the level of virtual machines, where the load balancer manages and balances the load across all virtual machines using different LB algorithms [9].

The paper's content is organized as follows: Section II offers a comprehensive review of the literature on several key Load Balancing methods. In Section III, we review the current state of the art for the for DPBRR algorithm to enhance load balancing and introduce our proposed approach, to distribute tasks evenly among resources based on priority, the jobs with higher priorities will become more significant. We discuss the encryption process in detail. Section IV presents the results and a comparative analysis of our recommended methodology. A thorough explanation of the study paper's findings and closing thoughts is provided in Section V.

II. Literature Review

The research paper titled “A Hybrid Algorithm for Scheduling Scientific Workflows in Cloud Computing” by Muhammad Tahir, Muhammad sardaraz in the year 2019 [1] [10]. The authors put forth an algorithm that uses the PSO algorithm to schedule scientific operations in two key stages: task preparation and task scheduling [23]. Throughout the scheduling process, the load balance of cloud resources is tracked by this hybrid method.

The research paper titled “Load balancing in cloud computing – A hierarchical taxonomical classification” by G Kavitha, S Afzal in the year 2019 [2]. It primarily categorized the many load balancing algorithm types with clarity and supplied details about the technique employed, the complexity of the algorithm, and its benefits and drawbacks. It primarily demonstrates the functionality and complexity of the load balancing algorithms, with task scheduling, virtual machine scheduling, and resource scheduling serving as the primary criteria. It also compares the algorithms' operational processes and provides percentages to help readers understand the algorithms, covering all the major LB algorithms.

The research paper titled “A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment” by Rajkumar Buyya, Ram Mohana Reddy G, Shridhar G D in the year 2020[3]. By combining modified CSO and PSO algorithms for job scheduling, the authors presented a hybrid with a bio-effect method for asset allocation. Compared to current scheduling and techniques, this algorithm aims to increase response time, reliability and resource utilization.

The research paper titled “A Hybrid Model for Load Balancing in Cloud Using File Type Formatting” by Adnan Sohail, M Junaid, Ahmed, H Alhakami, Imran Ali Khan, Abdullah Baz in the year 2020 [4]. The main goal is to use support vector machine (SVM) technology, which looks at the quantity and quality of files stored in the cloud. It works based on how well it manages processing speed and SLA.

The research paper titled “Dynamic load balancing algorithm for balancing the workload among virtual machines in cloud computing” by S C Sharma, Mohit Kumar in the year 2017 [5]. Cloud computing technologies are used to reduce waiting time and increase resource usage. Demonstrate that the suggested approach outperforms current algorithms regarding work allocation, workload monitoring and dynamic control. Incoming tasks are also assigned to the most appropriate virtual machine.

The research paper titled “A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications” by Azween Abdullah, N Z Jhanjhi, M A Alzain, Dalia Abdulkareem Shafiq in the year 2021[6]. Increase cloud asset efficiency rate 78% relative to current traffic distribution algorithms to achieve decent performance and improve metrics. The proposed method prioritizes virtual machines (VMs) for workload scheduling, underutilization, and quality of service (QoS) balancing. The algorithm mainly focuses on optimizing IaaS cloud models to ensure high performance and minimize underload.

The research paper titled “Cloud Dynamic Load Balancing and Reactive Fault Tolerance Techniques: A Systematic Literature Review (SLR)” by A Yousif, M Bakri Bashir, T M Tawfeeg, Alzubair Hassan, Samar M alqhtani, Awad Ali, Rafik Hamza in the year 2022 [7]. To make cloud load balancing more efficient, the authors of this study focus on implementing fault management techniques along with real time workload management. Adaptive load distribution jobs are dynamically assigned to the virtual machine based on current state of the system. (VMs). In addition, a reactive failover method reacts to system failures to keep cloud services stable.

The research paper titled “Dynamic Resource Allocation Using an Adaptive Multi-Objective Teaching-Learning Based Optimization Algorithm in Cloud” by Mohammadreza Ramezanpour, R Khorsand, Ali Moazeni in the year 2023 [8]. The authors recommend a dynamic resource allocation based on the strategy of AMO-TLBO method. By dynamically allocating resources based on application capacity, this strategy saves costs and optimizes resource consumption. The algorithm has been greatly improved using web-based technology. In addition, the recommended strategy outperforms several popular algorithms such as NSGA-II, MOPSO and TLBO.

The research paper titled “Load Balancing in Cloud Environment: A State-of-the-Art Review” by Durgaprasad G, Yogesh Lohumi, M Zubair Khan, Prakash Srivastava, Abdulrahman in the year 2023 [9]. Efficiency, scalability and performance are negatively affected by the load imbalance that cloud computing constantly faces. This essay examines load balancing and improving service quality in an on-demand computing setting. In addition to explaining load balancing solutions, the taxonomy and classification of load balancing algorithms is discussed. A common well-solved problem was server load imbalance.

Finally, a better answer to cloud computing load balancing issues and resource allocation concerns is offered by load balancing algorithms that integrate dynamic priority-based round robin (DPBRR). Task distribution, resource allocation, and workload modification are increasingly important considerations when working with virtual machines. As a result of the round robin process, which distributes jobs among the available virtual machines in a cyclic fashion to give each VM an equal chance, the priorities are dynamically changed in response to workload patterns. Its fault tolerance and high adaptability allow it to function as a reliable balancing load technique in environments of cloud computing by dynamically reallocating and redistributing resources to VMs with lower priority when any VM becomes overwhelmed.

The table highlights the drawbacks of each methodology and limitations. The proposed load balancing algorithm combines the dynamic nature with a priority based round robin algorithm offers a better solution for scalability, resource allocation, task distribution, fault tolerance and more, this can improve the balancing load in cloud environments.

The Table I provides a literature review on existing methodologies for enhancing balancing of load in virtualized computing environments. It provides a comparison of different algorithms, including Improved WRR, PMHEFT, CMODLB, Dynamic load balancing algorithms, SARM, MOABCQ and more.

TABLE I. LITERATURE REVIEW ON EXISTING METHODOLOGIES

S:NO	AUTHORS	TITLE	APPLIED METHODOLGY	DRAWBACKS
1	A Pravin , N Manikandan	'An Efficient Improved Weighted Round Robin Load Balancing Algorithm in Cloud Computing' [13].	Improved WRR	For weight assignment, the updated WRR mostly relies on static server specs. It may not adapt to dynamic changes.
2	S C Jain , M Sohani	'A Predictive Priority-Based Dynamic Resource Provisioning Scheme With Load Balancing in Heterogeneous Cloud Computing' [12].	PMHEFT	When the system gets bigger or there are more jobs to complete, its scalability could become an issue.
3	N Panwar, Sarita Negi , M M Singh Rautham , V Kunwar Singh	'CMODLB: an efficient load balancing approach in cloud computing environment' [14].	CMODLB	There may be a large processing overhead when several machine learning and optimization techniques are used.
4	Xuqing Ke, Lingjun Zhong, Wei Hou, Limin Meng,	'Dynamic Load Balancing Algorithm Based on Optimal Matching of Weighted Bipartite Graph' [15].	KUHN-MUNKRES	It can be more communication-intensive to update the weighted bipartite graph appropriately.
5	Fan Hong, Jiang Zhang, Tom Peterka, H Guo, Xiaoru Yuan	'Dynamic Load Balancing Based on Constrained K-D tree Decomposition for Parallel Particle Tracing' [16].	DECOMPOSITION K-D TREE	The method might not be as flexible when it comes to dynamic shifts in the workload allocation or system attributes.
6	Sun-Yuan Hsich, Rajkumar Buyya, Chih-Heng Ke, Albert Y Zomaya, W-K Chung, Yun Li	'Dynamic Parallel Flow Algorithms With Centralized Scheduling for Load Balancing in Cloud Data Center Networks' [17].	CDPFSMP & CDPFS	As a result of the centralization, the network may experience a bottleneck or single point of failure.
7	J-P Yang	'Elastic Load Balancing Using Self-Adaptive Replication Management' [20].	SARM	It might rely on how well certain thresholds and parameters are adjusted.
8	W Kimpan, B kruekaew	'Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning'[21] [22].	MOABCQ	There may be trade-offs between objectives, and these should be properly weighed. Extended schedule periods due to slower convergence can affect overall system responsiveness and performance.

III. PROPOSED METHODOLOGY

In this section, we will go over how to enhance load balancing to guarantee cloud computing in this section [18] [19]. Improving application performance through faster response times and lower network latency is the primary goal shared by all users. The suggested method is applied to enhance load balancing performance to resolve this issue. This section describes the methods to make load balancing better.

In Fig. 1, the algorithm of load balancing starts by initializing the distributed system and carefully determining the nodes and their capacity as well as the tasks executed on every node. The load balancing algorithm begins by setting up the distributed system and then calculating the nodes and their capacity together with the tasks that are on each node. When the system detects the disparities in workload distribution load balancing is activated before the activation of load balancing it must meet the set of conditions or its threshold value. The process of monitoring the task priorities, workloads and system status continues with the emphasis on gathering information on resource usage and other metrics. Tasks are distributed according to predetermined standards, which can be critical or urgent, which means that priority tasks are processed in a balancing load procedure. An algorithm is then introduced to analyze the workload of each node to account for prioritized tasks and computing load. Considering the priorities of the tasks and the general load of the system, the purpose of this decision process is to compare the current state of the given system with the set thresholds or requirements. Tasks are chosen for migrations considering the current workload distribution and task priorities, aiming to transfer less important tasks from nodes with high load to nodes with low load. Tasks should be selected for migration based on the current workload distribution and task priorities so that less important tasks can be moved from burdened nodes to less burdened nodes. A relocation strategy is formulated, so that the tasks are migrated along with their priority information intact, and that could involve tasks segmenting or transferring them in their entirety. Communication among the nodes involved in the migration process guarantees the process is done effectively so that the data and state information are transferred smoothly. Tasks' execution on target nodes obeys their priorities. There is ongoing monitoring of the system responses to the load balancer. Based on the results, necessary changes are made in the balancing load strategy. The feedback medium is set to continue to ameliorate the cargo balancing strategy by conforming to dynamic system conditions and returning to the regulator when necessary. Eventually, the termination condition, which is either balancing load or reaching a system stability thing, decides when the algorithm stops working.

A. Proposed DPBRR Algorithm

Initialization: Count the number of nodes in the distributed system, then also the capacities of the nodes and the tasks that are running on each node. Establish a threshold or set of requirements to kick off load balancing once imbalances are discovered.

1) Monitoring: Monitor the workload, tasks priorities and system status around the clock. Find out the data on resource utilization and other important indicators.

2) Task prioritization: Tasks should be sorted according to the priority levels that are defined beforehand such as criticality, importance, or urgency. While balancing load, tasks with more priority should be given preference.

3) Load evaluation: Look at the workload on each node at the instant, considering the priorities of the assigned tasks and the load of the computations.

4) Decision making: Evaluate the necessity of load balancing by comparing the current condition of the system with the previously defined boundaries or limitations. Involve yourself in the decision-making process by considering both task priorities and the overall burden.

5) Task selection: Ascertain which jobs need to be transferred in agreement with the current load allocation and their priorities. Firstly, the movement of less significant tasks from overworked to underworked nodes should be given the highest priority.

6) Migration strategy: Create a task migration plan which should be done with the consideration of priority information. Such a division of labor could be achieved by breaking the task into smaller pieces or by shifting the entire task.

7) Communication: Let know the concerned nodes about the migration plan. Ensure that all the required data and state information is transferred without a glitch.

8) Task execution: Ensure the priority of the assigned priorities when performing the migrated tasks on the target nodes. In addition, monitor the system's reaction and adjust as required.

9) Feedback loop: Design a feedback mechanism that allows for the continuous adjustment of the load-balancing strategy as the system adapts. Repeat the method and return to the observation step.

10) Termination condition: Give the algorithm a stop condition. It might be based on attaining a predetermined degree of load balance, system stability or other specific requirements.

Algorithmic steps for the proposed DPBRR Algorithm
Step 1: Develop the Task Priority Queue where each task has priority level.
Step 2: Implement the Round Robin server list by yourself.
Step 2.1: Each server receives a unique server ID.
Step 2.2: To start the process, the CSI (Current Server Index) must be set to 0.
Step 3: The Receive Incoming Requests and Tasks step is the most important step.
Step 3.1: If the task is a high priority, then put it into the priority queue and check whether the resources are available or not.
Step 4: Monitor the Priority Queue and the resources availability all the time.
Step 4.1: Work on the server assigned to you to complete the task and check the status of the task from time to time.
Step 5: When the job is done, disconnect from the server or resource.
Step 5.1: If there are any outstanding high-priority tasks in the Priority Queue, direct the next available server to the highest-priority task.
Step 6: After the high-priority tasks are completed, servers get distributed to lower-priority tasks in Round Robin manner.
Step 6.1: Therefore, we must increase the CSI (server index) while considering resource availability and fairness.
Step 7: Keep an eye on the status of the newly arrived tasks, resource availability, and task completions.
Step 8: Create the termination conditions and terminate the algorithm when the termination criteria are fulfilled.

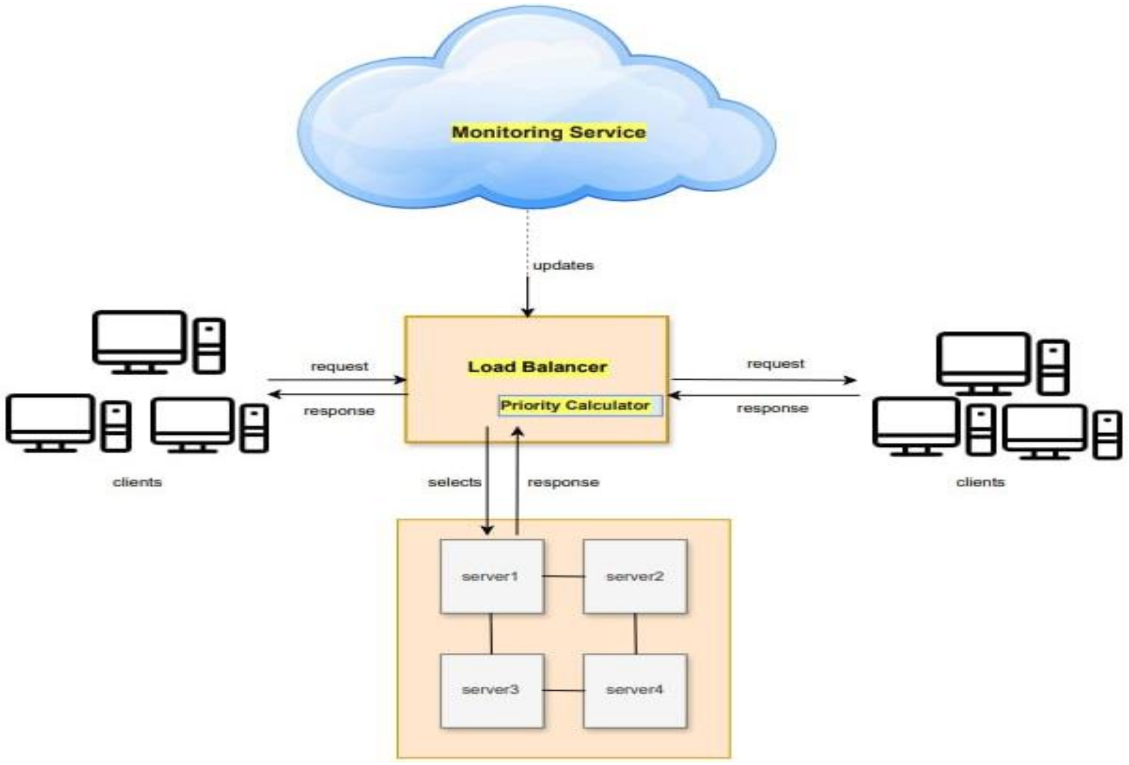

Fig. 1. A Proposed architecture for DPBRR algorithm to enhance load balancing.

- p be the priority
- a be the priority importance
- b be the resource that is available
- c be the load on the system
- q be the index of the resource
- s be the starting index
- j be the particular priority
- T be the total available resources
- Np be the new priority = Np
- Op be the old priority = Op

a) Priority Assignment Equation:

$$p = \alpha * a + \beta * b + \gamma * c$$

b) Task Distribution Equation:

$$q = (s + \Sigma(j)) \% T$$

c) Load Balancing Adjustment Equation:

$$Np = Op + \Delta p$$

These formulas will demonstrate how to compute the available resources, task allocation, and priority factors. When DPBRR applies the equation to distribute tasks evenly among resources based on priority, the jobs with higher priorities will become more significant. Tasks are dynamically altered, and priorities are modified based on changes in priorities as determined by the equations. These suggested equations will aid in effective resource usage and balanced workload distribution within the system.

IV. Result Analysis

This paragraph presents the suggested load-balancing technique as well as experimental results for various load-balancing algorithms. Several parameters are taken into consideration when analyzing it, such as throughput, response time, and resource usage. Existing methods like CMODLB, Kuhn-Munkres, and improved WRR were taken into consideration. As given below.

A. Throughput, Response Time, and Resource Utilization

The amount of material or items passing through a system or process is called throughput, The amount of time taken by a system to respond to a request or input is called Response Time, the efficiency in using the resources is called Resource Utilization, Analysis of CMODLB, Kuhn-Munkres, Improved WRR and proposed DPBRR in Table II, Table III and Table IV.

TABLE II. Computational Throughput Analysis of DPBRR vs. Conventional Algorithms

S.No	Algorithms	Throughput (requests/second)
1	Improved WRR	900 tasks/s
2	Kuhn-Munkres	950 tasks/s
3	CMODLB	980 tasks/s
4	DPBRR	1000 tasks/s

Here, it explores how DPBRR can more efficiently raise the throughput value than other algorithms.

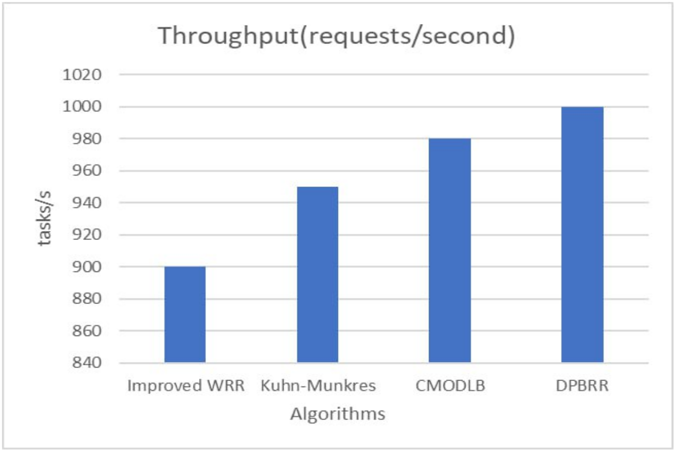

Fig. 2. Computational throughput analysis of DPBRR vs conventional algorithms.

Fig. 2 shows how DPBRR achieves a perfect spectrum since it is faster than other algorithms like Improved WRR, Kuhn-Munkres, CMODLB with a Throughput over 1000 tasks/second. The multi-node adaptive priority-oriented scheduling of this algorithm likely optimizes the resource allocation, with the consequent minimization of latency and improvement in throughput. This benefits DPBRR as businessmen's preferred compute tool in cases where fast processing, well-timed response, and request handling are taken as top priorities, hence, we are distinguishing it as the most efficient algorithm from the others.

Here, it explores how DPBRR can more efficiently decrease the Response time value than other algorithms.

TABLE III. Computational Response Time of DPBRR vs. Conventional Algorithms

S.No	Algorithms	Response Time (milliseconds)
1	Improved WRR	60ms
2	Kuhn-Munkres	55ms
3	CMODLB	52ms
4	DPBRR	50ms

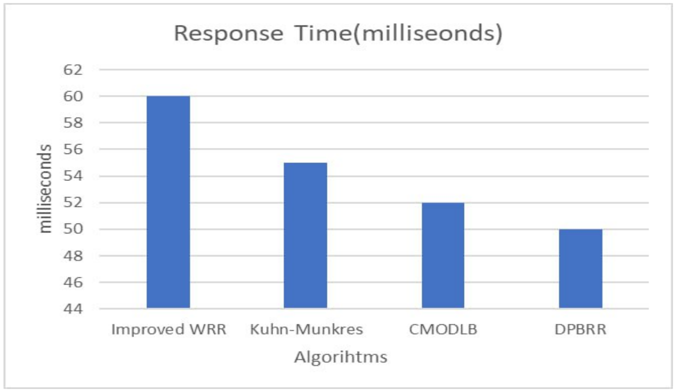

Fig. 3. Computational response time of DPBRR vs. conventional algorithms.

In analyzing the response times of various algorithms from Fig. 3, it is apparent that DPBRR emerges as the most efficient choice. When comparing the reaction times across different algorithms, DPBRR stands out as the most effective option. While Improved WRR exhibits a response time of 60 milliseconds, DPBRR demonstrates a significant improvement with a response time of only 50 milliseconds. Both Kuhn-Munkres and CMODLB show enhancements over Improved WRR, with response times of 55 and 52 milliseconds, respectively. However, DPBRR surpasses them all, showcasing its remarkable ability to handle requests swiftly and minimize system latency. This outstanding performance underscores DPBRR's suitability for scenarios where quick reaction times are crucial, establishing it as the optimal algorithm for maximizing system performance and ensuring prompt request processing.

Here, it explores how DPBRR can more efficiently optimize the Resource utilization value than other algorithms.

TABLE IV. RESOURCE UTILIZATION OF DPBRR VS. CONVENTIONAL ALGORITHMS

S.No	Algorithms	Resource Utilization (%)
1	Improved WRR	75%
2	Kuhn-Munkres	78%
3	CMODLB	72%
4	DPBRR	80%

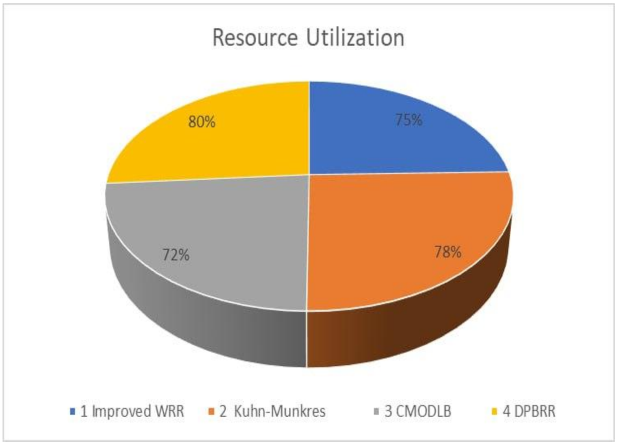

Fig. 4. Resource utilization of DPBRR vs conventional algorithms.

From Fig. 4, the comparisons of resource usage of different algorithms show trends in their effectiveness. The improved WRR consumes resources of 75%, while Kuhn-Munkres raises to 78%. But from the listed algorithms CMODLB has the highest resource consumption of all other algorithms. DPBRR manages to maintain a respectable 80% utilization rate by striking a compromise between resource efficiency and performance. It efficiently optimizes resource allocation and achieves competitive performance. DPBRR is a great option where increasing performance while optimizing resource utilization is crucial because of its balanced resource usage. It manages the system resources even though it does not have the lowest utilization rate.

V. CONCLUSION

In conclusion, the study of Dynamic Priority-Based Round Robin (DPBRR) load balancing is a useful way to solve the important problem of efficient load management in cloud computing settings. DPBRR aims to reduce response time and maximize resource utilization by implementing a dynamic adaptive mechanism that intelligently distributes work across cloud servers. Because DPBRR can dynamically adjust task priorities in response to real-time data, it is particularly useful for workload fluctuations and changing resource demands. Using advanced load balancing technology in virtualized computing, dynamic priority derived from Round Robin is a significant step forward in managing workload imbalances in distributed systems. The recommended approach shows cloud computing load balancing solutions with better performance and economy. By implementing several techniques, DPBRR helps reduce congestion and improve system flexibility and adaptability. The DPBRR study offers a functional and effective way to improve system performance and resource allocation in cloud services, solve important problems, and soon open the door for further developments.

REFERENCES

[1] Pachipala, Y., Dasari, D.B., Rao, V.V.R.M., Bethapudi, P., Srinivasarao, T. Workload prioritization and optimal task scheduling in cloud: introduction to hybrid optimization algorithm (2024) Wireless Networks.

[2] Kavitha .G, Shahbaz Afzal, 'Load balancing in cloud computing -A hierarchical taxonomical classification.', Open Access, 2019.

[3] Karimunnisa, S., Pachipala, Y. Deep Learning Approach for Workload Prediction and Balancing in Cloud Computing (2024) International Journal of Advanced Computer Science and Applications, 15 (4), pp. 754-763.

[4] M. Junaid, A. Sohail, A. Ahmed, A. Baz, I. A. Khan, and H. Alhakami, "A Hybrid Model for Load Balancing in Cloud Using File Type Formatting," IEEE Access, vol. 8, pp. 118135–118155, 2020, doi: 10.1109/access.2020.3003825.

[5] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," Procedia Computer Science, vol. 115, pp. 322–329, 2017, doi: 10.1016/j.procs.2017.09.141.

[6] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications," IEEE Access, vol. 9, pp. 41731–41744, 2021, doi: 10.1109/access.2021.3065308.

[7] T. M. Tawfeeg et al., "Cloud Dynamic Load Balancing and Reactive Fault Tolerance Techniques: A Systematic Literature Review (SLR)," IEEE Access, vol. 10, pp. 71853–71873, 2022, doi: 10.1109/access.2022.3188645.

[8] A. Moazeni, R. Khorsand, and M. Ramezanpour, "Dynamic Resource Allocation Using an Adaptive Multi-Objective Teaching-Learning Based Optimization Algorithm in Cloud," IEEE Access, vol. 11, pp. 23407–23419, 2023, doi: 10.1109/access.2023.3247639.

[9] Y. Lohumi, D. Gangodkar, P. Srivastava, M. Z. Khan, A. Alahmadi, and A. H. Alahmadi, "Load Balancing in Cloud Environment: A State-of-the-Art Review," IEEE Access, vol. 11, pp. 134517–134530, 2023, doi: 10.1109/access.2023.3337146.

[10] L. Yang, Y. Xia, L. Ye, R. Gao, and Y. Zhan, "A Fully Hybrid Algorithm for Deadline Constrained Workflow Scheduling in Clouds," IEEE Transactions on Cloud Computing, vol. 11, no. 3, pp. 3197–3210, Jul. 2023, doi: 10.1109/tcc.2023.3269144.

[11] G. Xu, J. Pang, and X. Fu, "A load balancing model based on cloud partitioning for the public cloud," Tsinghua Science and Technology, vol. 18, no. 1, pp. 34–39, Feb. 2013, doi: 10.1109/tst.2013.6449405.

[12] M. Sohani and S. C. Jain, "A Predictive Priority-Based Dynamic Resource Provisioning Scheme With Load Balancing in Heterogeneous Cloud

Computing," IEEE Access, vol. 9, pp. 62653–62664, 2021, doi: 10.1109/access.2021.3074833.

[13] M. N and P. A, "An Efficient Improved Weighted Round Robin Load Balancing Algorithm in Cloud Computing," International Journal of Engineering &amp; Technology, vol. 7, no. 3.1, p. 110, Aug. 2018, doi: 10.14419/ijet.v7i3.1.16810.

[14] S. Negi, M. M. S. Rauthan, K. S. Vaisla, and N. Panwar, "CMODLB: an efficient load balancing approach in cloud computing environment," The Journal of Supercomputing, vol. 77, no. 8, pp. 8787–8839, Jan. 2021, doi: 10.1007/s11227-020-03601-7.

[15] W. Hou, L. Meng, X. Ke, and L. Zhong, "Dynamic Load Balancing Algorithm Based on Optimal Matching of Weighted Bipartite Graph," IEEE Access, vol. 10, pp. 127225–127236, 2022, doi: 10.1109/access.2022.3226885.

[16] J. Zhang, H. Guo, F. Hong, X. Yuan, and T. Peterka, "Dynamic Load Balancing Based on Constrained K-D Tree Decomposition for Parallel Particle Tracing," IEEE Transactions on Visualization and Computer Graphics, vol. 24, no. 1, pp. 954–963, Jan. 2018, doi: 10.1109/tvcg.2017.2744059.

[17] W.-K. Chung, Y. Li, C.-H. Ke, S.-Y. Hsieh, A. Y. Zomaya, and R. Buyya, "Dynamic Parallel Flow Algorithms With Centralized Scheduling for Load Balancing in Cloud Data Center Networks," IEEE Transactions on Cloud Computing, vol. 11, no. 1, pp. 1050–1064, Jan. 2023, doi: 10.1109/tcc.2021.3129768.

[18] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic Resource Provisioning With Fault Tolerance for Data-Intensive Meteorological Workflows in Cloud," IEEE Transactions on Industrial Informatics, vol. 16, no. 9, pp. 6172–6181, Sep. 2020, doi: 10.1109/tii.2019.2959258.

[19] 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.

[20] J.-P. Yang, "Elastic Load Balancing Using Self-Adaptive Replication Management," IEEE Access, vol. 5, pp. 7495–7504, 2017, doi: 10.1109/access.2016.2631490.

[21] Bhargavi, M., Pachipala, Y. Enhancing IoT Security and Privacy with Claims-based Identity Management (2023) International Journal of Advanced Computer Science and Applications, 14 (11), pp. 822-830.

[22] B. Kruekaew and W. Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning," IEEE Access, vol. 10, pp. 17803–17818, 2022, doi: 10.1109/access.2022.3149955.

[23] Aakisetti, R.S.K., Ganta, V., Yellamma, P., Siram, C., Gampa, S.H., Brahma Rao, K.V. Dynamic Priority Scheduling Algorithms for Flexible Task Management in Cloud Computing (2024) International Journal of Intelligent Systems and Applications in Engineering, 12 (13s), pp. 246-256.