International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Enhancing Load Balancing in Heterogeneous High Performance Cloud Computing with MMWRR+: A Novel Task Allocation Approach

Suvarna N A

Research Scholar, SoE, GD Goenka University, Gurugram, Haryana - 122102, India
Email: *suvarna07.aradhya[at]gmail.com*

Abstract: ***Cloud computing has emerged as a leading paradigm for providing scalable, on - demand and pervasive computing resources. But load balancing through efficient task allocation remains a challenging problem in cloud environments due to the dynamic and unpredictable nature of workloads across diverse virtualized resources. Addressing this, we present an innovative task allocation approach by amalgamating two state - of - the - art scheduling algorithms, namely Max - Min and Weighted Round - Robin to achieve effective load balancing in heterogeneous high performance distributed computing (HPDC). The proposed method MMWRR+ emphasizes resource awareness, scalability and dynamic adaptability, alleviating resource contention and optimizing resource utilization. Comparative evaluations demonstrate that the proposed approach outperforms the referred individual algorithms and prior research - based improvement techniques to these algorithms. Moreover, it effectively mitigates resource imbalances, efficiently manages diverse workloads and empowers cloud systems to handle fault tolerance.***

Keywords: Load Balancing; Resource Optimization; Max - Min; Heuristic Algorithm; Fault Tolerance

1. Introduction

Load balancing in cloud systems is defined as the practice of distributing incoming network traffic, computational tasks, or application workloads across multiple resources, such as virtual machines (VMs), servers, or datacentres, to optimize resource utilization, improve system performance, and ensure high availability, reliability and fault tolerance. The primary goal of load balancing is to prevent any single resource from becoming overwhelmed by an excessive amount of work while others remain underutilized. In totality, load balancing ensures that all resources are always kept productive. The challenges to load balancing are posed by the nature of distributed systems, such as: lack of centralized control, coordination between heterogeneous nodes, network structures, automated services, VM migrations and wide geographical distribution of resources. To address all the above challenges and meet the best of all performance metrics is highly computational and NP hard problem. Only selective challenges can be targeted, and few metrics can be aimed for improvement. This research article considers only the time critical parameters such as make span, waiting time, response time for improved performance along with primary goal of optimal resource utilization. Inherently, the framework adopted also facilitate the easy detection and mitigation of faults. This paper proposes a hybrid, resource - aware, scalable, heuristic, and dynamic task allocation method to achieve efficient load balancing in cloud computing. The method combines the advantages of both Max - Min and Weighted Round - Robin to exploit their complementary features and overcome their limitations.

Max - Min, Min - Min and Weighted Round - Robin are the popularly known and vastly used heuristic task scheduling algorithms in cloud systems. These algorithms are used in our research as benchmark to investigate and compare the effects on load balancing.

The Max - Min, Min - Min, Min - Max, and Max - Max algorithms are all different ways of implementing the minimax strategy in job scheduling. They are based on game theory approaches to optimization for job scheduling. Min - Max and Max - Max are less common because they do not correspond to well - known or widely used strategies in the context of task scheduling.

In this research, extensive simulations are conducted in a representative cloud computing environment to evaluate the performance of the proposed resource - aware hybrid task allocation method. The results demonstrate the superiority of the method compared to existing load balancing techniques in terms of resource utilization, response time, makespan and system throughput.

The contributions of this research are twofold. Firstly, the proposed method offers a novel approach to load balancing in cloud computing by using centralized task allocation technique for batch processing. Secondly, the resource - awareness aspect of the method ensures efficient utilization of heterogeneous cloud resources, leading to improved system performance.

The remainder of this article is organized as follows: Section 2 provides an overview of related work in the field of task scheduling and load balancing in cloud computing. Section 3 describes the methodology and algorithms employed in the proposed allocation method. Section 4 presents the simulation setup and performance evaluation results. Finally, Section 5 concludes the article with a summary of findings and future research directions.

2. Related Works

Load balancing in cloud systems is not just a critical issue but also the main objective of task - allocation, as it ensures

that resources are utilized efficiently and that users experience a consistent level of performance [1], [18]. Algorithms for load balancing through task allocation are broadly classified as: Centralized Algorithms, Decentralized Algorithms, Task Specific Algorithms and Game - Theoretic Algorithms [1]. These load balancing algorithms are built to work suitably across different network architectures and operating mechanisms [2], [7].

Each algorithm emphasizes on certain specific QoS parameters. Because, in a heterogeneous cluster, the load balancing through task allocation to improve all the QoS parameters is highly computational and is a NP Hard problem [6]. Therefore, only sub optimal solution exists that helps to improve few specific QoS parameters.

The task allocation algorithms are broadly classified as primitive, heuristic, or metaheuristic based on the level of sophistication and complexity of the algorithms and the approach they use to solve the task allocation problems [6].

Game - theoretic task allocation algorithms are based on principles of game theory, which studies the strategic interactions between decision - making agents. These heuristic algorithms aim to allocate tasks efficiently while considering the self - interested behaviour of individual agents. These algorithms offer a principled approach to address task allocation problems in scenarios where agents act strategically, helping achieve efficient and stable task assignments [36].

In recent years, there has been a growing interest in developing resource - aware, dynamic, fault detectable and task migratable load balancing algorithms [3], [7], [22]. These algorithms consider the resources available in the neighbourhood when making load balancing decisions. This can help to improve load balancing performance by ensuring that tasks are allocated to resources that have the capacity to meet their requirements [5], [9], [13].

Focussing on the research towards the improvements to the existing heuristic Min - Min and Max - Min algorithms along with the fundamental insights into Weighted Round - Robin, several researchers have presented modified algorithms based on these.

Saeed Parsa et. al. [21] suggested a modified scheduling policy based on Max - Min and Min - Min called as “RASA”. In this algorithm, tasks are scheduled to appropriate resources using one of the two strategies: Max - Min and Min - Min alternatively. Resources are characterized with two variables, namely processing speed and communication speed.

An improved Max - Min algorithm for Elastic Cloud called (ECMM) is suggested by Xiaofang Li, et al. [30]. Main idea is to maintain an executing task status table and a virtual machine status table to estimate the workload in real time inside a load balancer for a batch of tasks. The algorithm selects the task with the longest execution time (Max) and assigns it to virtual machine with the shortest completion time (Min). Meanwhile, it also updates the number of tasks and the total task execution time of the virtual machine in the virtual machine status table. The process cycles until all tasks are allocated.

Belal Ali Al - Maytami et al [6]. suggests an improvement in Max - Min scheduling algorithm using Directed Acyclic Graph (DAG) based on the Prediction of Tasks Computation Time algorithm (PTCT) to estimate the preeminent scheduling algorithm for prominent cloud data. It employs Principal Components Analysis (PCA) to reduce the Expected Time to Compute (ETC) matrix.

S. VaaheedhaKfatheen et al. [31] proposes another improvement to Max - Min called as Mim - Mam. It segregates the tasks into two groups: 1. Above the average task length (MinETC) 2. Below the average task length (MaxETC). The number of tasks in both the resulted ETC’s is now used to take decision. If number of tasks in MinETC is less than or equal to MaxETC then Max - Min algorithm is used to schedule the MaxETC. Otherwise, Min - Min algorithm is used to schedule the MinETC.

O. M. Elzeki, et al. [33] recommends an improvement to Max - Min. The algorithm calculates the expected completion time of the submitted tasks on each resource. Then the task with the overall maximum expected execution time is assigned toa slowest resource so that many shorter jobs can be completed simultaneously on the fastest machine to improve the makespan. Finally, this scheduled taskis removed from meta - tasks and all calculated times are updated and the processing is repeated until all submitted tasks are executed.

Pandaba Pradhan et. al [35] advised an improvement to Max - Min and Min - Min through an algorithm called “IMM”. Algorithm assigns the first task according to Max - Min strategy and then assigns tasks using Min - Min strategy till the total task length of the assignments equal that of earlier Max - Min assignment. The technique is the same as that of [33].

D. Chitra Devi, et al. [8] advises an improvement to Weighted Round - Robin. it allocates the jobs to the most suitable VMs based on the VM’s information like its processing capacity, load on the VMs, and length of the arrived tasks with its priority. The static scheduler of this algorithm uses the processing capacity of the VMs, the number of incoming tasks, and the length of each task to decide the allocation on the appropriate VM. The dynamic scheduler of this algorithm additionally uses the load on each of the VMs to decide the allocation of the task to the appropriate VM. There is a probability at run timethat, in some of the cases that the task may take longer execution time based on the run time data. In such situations, the load balancer rescues the scheduling controller and rearranges the jobs through task migration to circumvent the overload on the VMs.

In summary, traditional task allocation methods often fail to consider the dynamic nature of cloud resources and the varying workload demands imposed by users. Therefore, there is growing need for novel task allocation methods that can effectively address these challenges and achieve optimal resource utilization and load balancing for time - critical applications in cloud computing.

3. Problem Formulation

In this research, the existing traditional algorithms: Max - Min, Min - Min and Weighted Round - Robin are investigated in depth along with the suggested improvements by earlier researchers to devise an improved algorithm "MMWRR+"which is resource - aware, dynamic, and scalable. The algorithm is designed for the following situations.

a) A subsystem of cloud/cluster consisting of a limited number of nodes.
b) Batch processing for scheduling of tasks at the broker level.
c) Only time - critical parameters are considered for improvement.

3.1 System Model

The algorithms are implemented on CloudSim, a Java - based simulation toolkit for modelling and simulating cloud computing systems. The broad architecture of cloud system and details of the entities within it are as under.

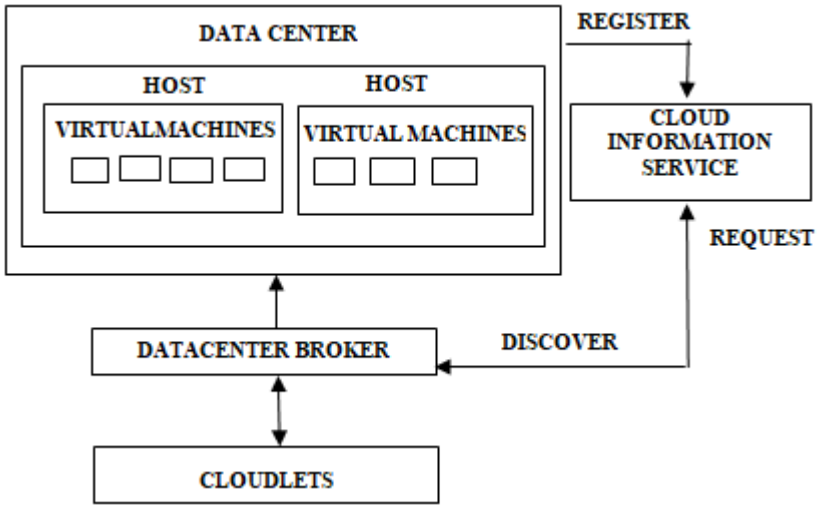

Figure 1: Architecture of the Cloud System

The cloud entities are emulated on a representative cloud setup (CloudSim) for implementation of the scheduling algorithms under study. Entire cloud system is a huge number of geographically distributed network of computing devices. For the system under study, a section or cluster of this network at one location ordatacenter with few computing nodes are considered. Tasks/Cloudlets are batch processed dynamically in real time with centralized task allocation at the broker level.

3.2 System Configuration

Virtual Machines (VMs): Four VMs of following configuration (Table - 1) are used. Onlyparameter varied is MIPS (Execution Speed in Millions of Instructions Per Second) to depict heterogeneous systems. The MIPs of four VMs are set to 1000, 2000, 3000 and 4000 respectively.

Table 1: Configuration of Virtual Machines

Image Size in MB	RAM in MB	BW in Gbps	PEs	VMM
10, 000	512	1000	1	Xen

Hosts: Two Hosts of the following configuration are used.

Table 2: Configuration of Hosts

RAM in MB	Storage	BW	PEs	MIPs	VM Allocation Policy
4096	1 GB	10, 000 Gbps	1	10, 000	First Fit

Datacenter: One Datacenter is created with architecture "X86", Linux Operating System and "XEN" VMM.

Cloudlets (Tasks): Tasks with random task lengths (between 100, 000 and 500, 000 million instructions) are applied in three different data sizes (100, 500 and 1000).

3.3 Proposed Algorithm (MMWRR+)

The algorithm is designed to exploit the advantages of Weighted Round - Robin and Max - Min.

Notations:
Number of tasks = n (namely, T_1, T_2, ……. . T_n)
Number of Machines = m (namely, VM_1, VM_2 …. . VM_m)
Length of each task = L_i (Millions of Instructions, i ranging from
1 ton; Also, $L_i > L_{i+1}$ for $1 \le i \le n$)

Total length of tasks, $L_T = \sum_{i=1}^{n} Li$

Processing rate (Speed) of each machine

$VM_i = \mu_I$ Millions of Instructions Per Second, i ranging from 1 to m; Also, $\mu_i > \mu_{i+1}$ for $1 \le i \le m$

Arrival rate of the tasks, λ = n/unit time = n ($\because$ Batch Processing)

Utilization of each VM, $\rho_i = \lambda/\mu_i$

Now, the problem is to assign the tasks to any one of the available machines so as utilize the machines to the optimum extent to complete the execution of tasks at the earliest possible.

Algorithm of MMWRR+:

1) Compute the total length of the tasks
   $L_T = \sum_{i=1}^{n} Li$

2) Compute the total available processing power of the virtual machines
   $\mu_T = \sum_{i=1}^{m} \mu i$

3) Compute the share of processing power of each virtual machine
   $\rho_i = \mu_i / \sum_{i=1}^{m} \mu i$

4) For (i = 1 to m) // for each machine VM_i
   {Tasks assigned to VM_i = TP_i = { } // Empty List
   For (j = 1 to n) // for each task T_j
   {If (Total Length of Tasks in $TP_i < \rho_i * L_T$)
   {If (Length of $T_j < ((\rho_i * L_T) - $ Length of $TP_i)$)
   Add T_j to TP_i
   }
   }
   }

5) Repeat steps 1 to 4 for the remaining tasks and leftover resources (CPU computing time).
6) Execute the tasks assigned to each VM in the reverse order.

The flow chart of the devised algorithm is indicated in figure - 2 below.

3.4 Simulation of algorithms

The state - of - the - art Max - Min, Min - Min, and Weighted Round - Robin and the earlier proposed improvements by the researchers to these algorithms along with the proposed algorithm (MMWRR+) are implemented on the simulation platform. The results are obtained for three different data sizesof 100, 500, and 1000, where each dataset is a collection of tasks with random task lengths. The graphs of makespan, waiting time and resource utilization are tabulated in the next section.

Flow Chart of MMWRR+:

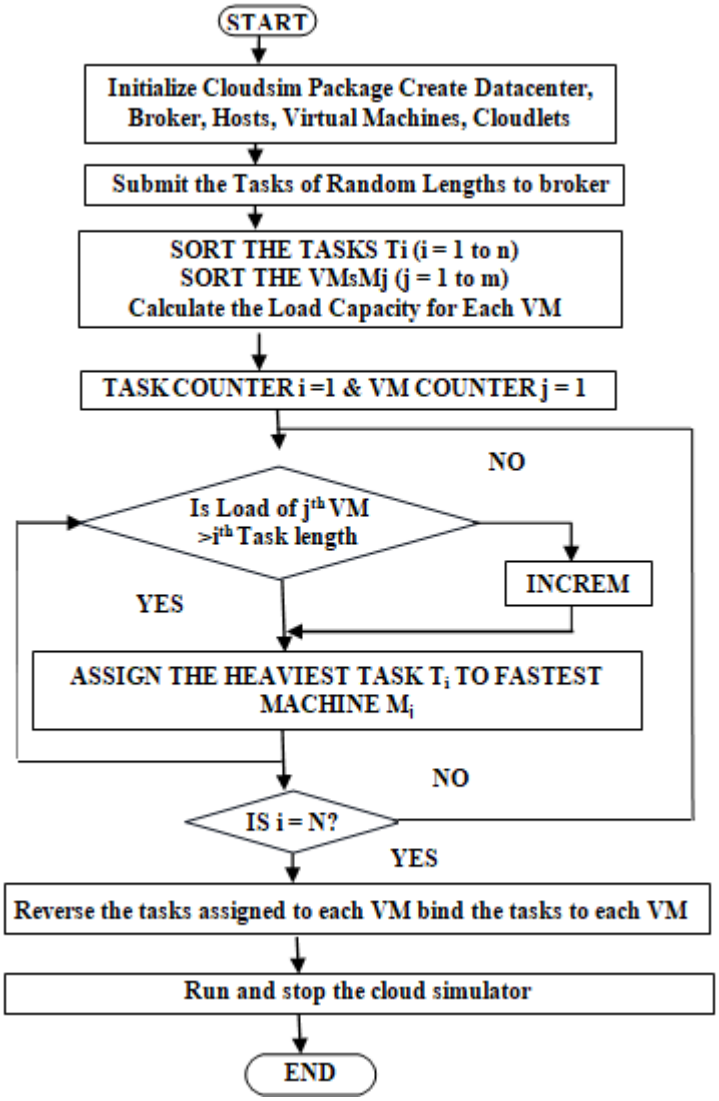

Figure 2: Flow chart of the proposed algorithm

4. Results

Table 3: Summary of Simulation Results

Summary of Results						
Makespan in Seconds						
Data Size	**Max - Min**	**Min - Min**	**WRR**	**RASA [21]**	**IMM [33]**	**MMWRR+**
100	3147	3211	3387	7887	3199	3179
500	106438	107041	107076	185870	104068	106686
1000	208055	215926	215755	276858	282395	215483
Average Wait Time in Seconds						
100	1839	1177	1465	2039	1376	1384
500	69640	35932	51135	67491	42882	50410
1000	136979	73586	107125	125485	88134	96981
	Resource (Computing Time) Utilized in Seconds					
VM	**Max - Min**	**Min - Min**	**WRR**	**RASA [21]**	**IMM [33]**	**MMWRR+**
Data Size = 100, Threshold = 3053 Seconds						
VM0	3098	2868	3390	7887	3182	3099
VM1	3151	3187	3194	2625	3101	3179
VM2	3157	3121	3228	2613	3117	3133
VM3	3165	3231	3018	2638	3199	3141
Data Size =500, Threshold = 106498 Seconds						
VM0	106462	107068	106827	276858	**103617**	106404
VM1	106469	106982	107125	87593	**103248**	106203
VM2	106511	106004	107118	87270	104068	106461
VM3	106512	106484	105637	87782	103969	106481
Data Size =1000, Threshold = 215304 Seconds						
VM0	210388	215979	215721	480341	215685	215210
VM1	210380	216010	215859	185833	215308	214892
VM2	210388	214759	215508	185878	215272	215282
VM3	210364	215191	214770	185850	215231	215302

4.1 Makespan (Time to complete the execution of all tasks)

Makespan obtained from the implementation of traditional and earlier suggested improved algorithms for three different data sizes listed in Table - 3. The graph so obtained from this data is indicated in Figure - 3.

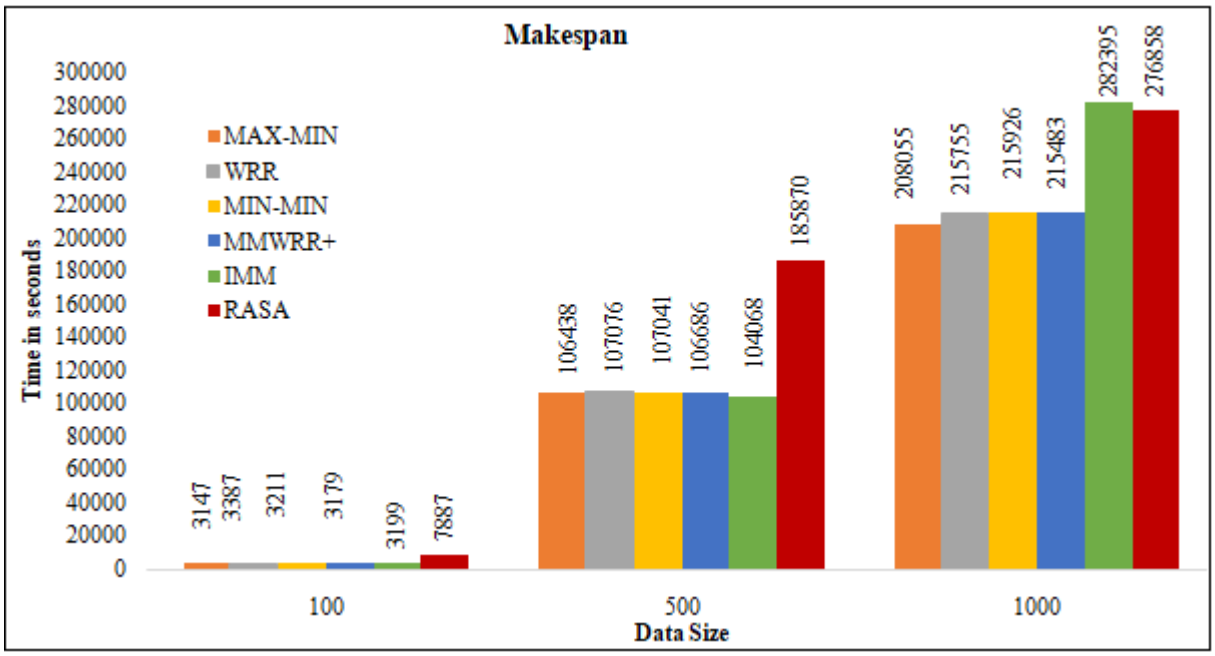

Figure 3: Makespan of the different algorithms

It is observed from the graph of makespan that MAX - MIN performs best for makespan. By prioritizing the assignment of tasks with the maximum completion time, MAX - MIN aims to balance the workload more effectively, preventing the occurrence of heavily loaded processors and idle time. This load balancing strategy led to improved overall efficiency and a reduced makespan. On the other hand, Min - Min makes task assignments based on the earliest

completion time of the tasks. This early commitment to suboptimal choices has led to a snowball effect, where subsequent task assignments become increasingly constrained and less efficient, ultimately leading to a larger makespan. WRR also performs badly for makespan as doesn't consider the varying processing times of tasks, leading to imbalanced resource utilization and longer overall completion time. IMM is not exhibiting consistent performance with varying data size. RASA is performing poor compared to all other algorithms.

4.2 Average Waiting Time:

The results of waiting time obtained from implementation of same set of algorithms for three different data sizes are listed in Table - 3. The graph so obtained from this data is shown in Figure - 4 below.

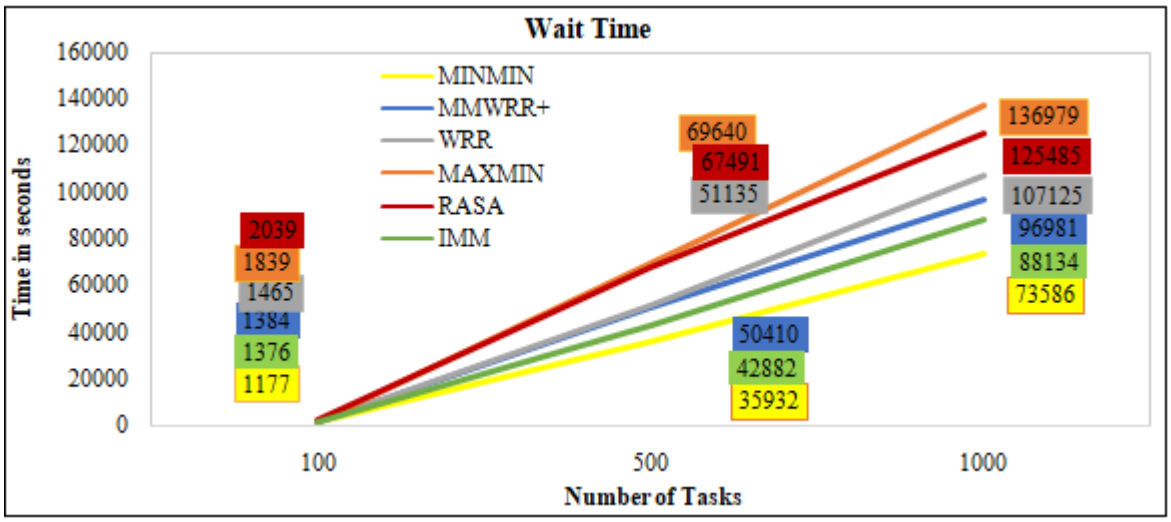

Figure 4: Average Wait Time of the different algorithms

The trend here is just opposite to that of the makespan graph. The Max - Min algorithm exhibits more wait time compared to all other algorithms. It is because it prioritizes processes with the maximum completion time. This strategy has led to longer wait times for tasks on processors with higher loads, potentially increasing the overall waiting time before tasks are completed. The waiting time is minimum in the Min - Min algorithm because it assigns tasks to processors based on the earliest completion time. Weightedround - robin performs badly for waiting time because it unfairly prioritizes tasks with higher weights, causing longer waiting times for lower - weighted tasks. IMM closely follows the best performing Min - Min. RASA and Max - Min are exhibiting longer wait times as longer jobs are scheduled prior to shorter jobs. The proposed MMWRR+ performs close to the best (Min - Min & IMM).

4.3 Load Balancing& Resource Utilization

Load Balancing is directly related to Resource Utilization. It is a very important performance metric, as it affects the overall time of completion (Makespan). Any server, which is under - utilized will wait for the over utilized server to complete its work. Load Balancing is to keep all the servers busy all the time or in other words, all servers finish the jobs at the same time.

Sample Calculation of Threshold CPU time (Resource utilization):

Using the M/G/c queuing model,
Resource Utilization = $\rho = \lambda / (c * \mu)$ where λ is the arrival rate of the tasks, μ is the service rate of the server and c is the number of servers. In batch processing, all tasks arrive at the same time. Therefore λ = total number of tasks at the server queue.

Total Task Lengths of 100 tasks = 31531862 (Millions of Instructions or MI)
Average MIs processed by four VMs in one second = 10, 000/4 = 2500 MI/Sec
(Where 1000, 2000, 3000, 4000 are the processing speeds of VMs, their sum is 10, 000 MIs)

Therefore, ρ = 31531862/ (4 * 2500) = 3153 seconds. The value, ρ also indicates the upper threshold of the workload to the VMs. This threshold value for data size of 500 and 1000 is also similarly calculated as 106498 seconds and 215304 seconds respectively.

The graphs below in figures: 5, 6 and 7 are the results to show the load imbalancing effect of the algorithms. Each graph is related to a specific data size.

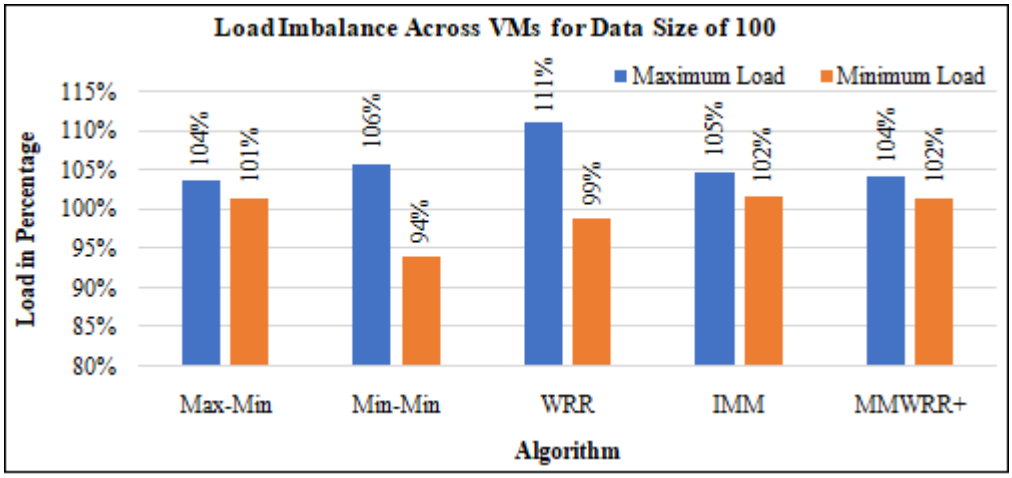

Figure 5: Load Imbalance for data size 100

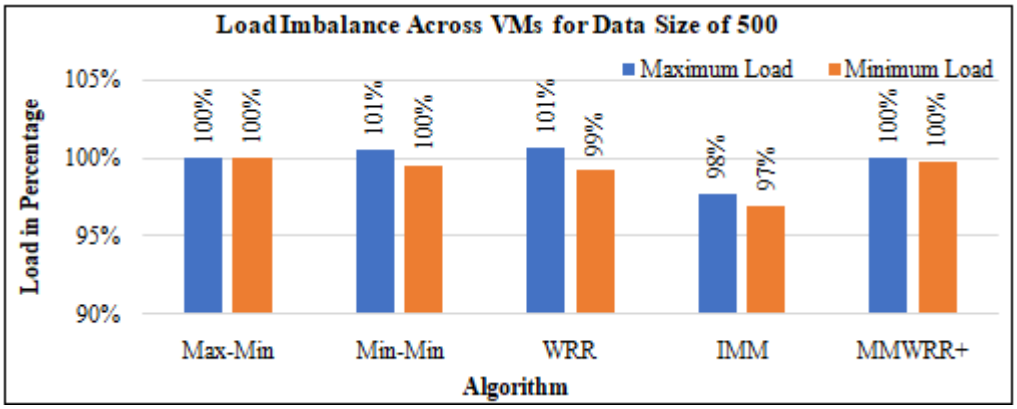

Figure 6: Load Imbalance for data size 500

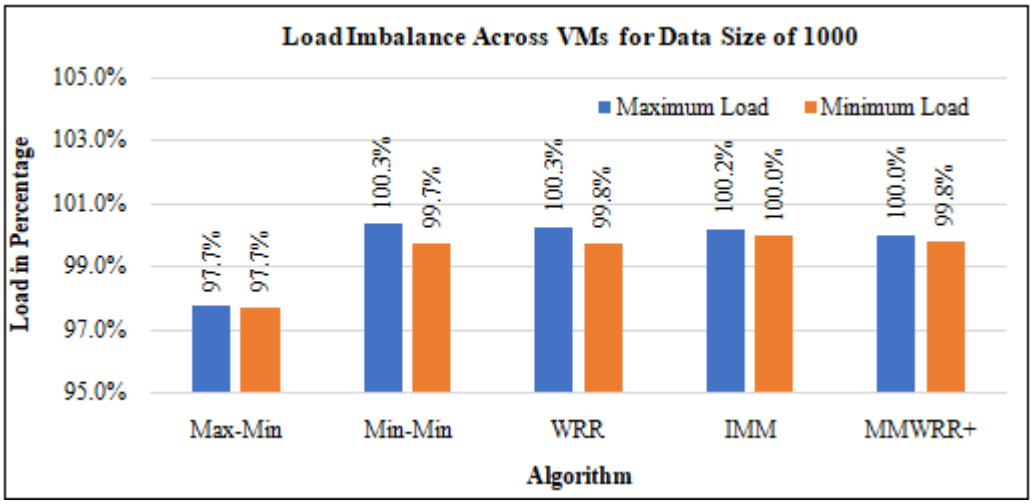

Figure 7: Load Imbalance for data size 1000

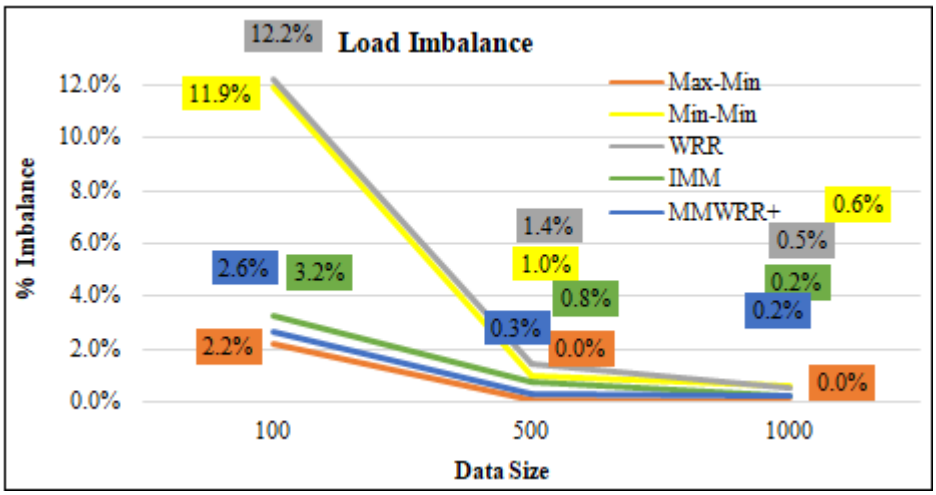

Figure 8: Summary of Load Imbalance as percentage deviation from Threshold

The workload distributed above/below the threshold value to each of the available four Virtual machines through the various algorithms to avail the CPU computing time is indicated in these graphs. Threshold indicates a 100% workload.

As is clear from the graph, the load balancing is poor for WRR. It is because it allocates tasks to processors based on fixed weights, without considering the dynamic workload or processing capabilities, leading to potential imbalances in processor utilization. Load balancing is very poor in RASA due to prioritization of smaller tasks. As it is very evident from the results table that RASA is not comparable to other algorithms, it is excluded from the graph. MAX - MIN performs best in load balancing as it prioritizes longer tasks. It also optimally utilizes the resources. MMWRR+ is very close to the best performing MAX - MIN, as far as load balancing is concerned. Next is IMM, close to MMWRR+ and performs almost equally good.

4.4 Fault Tolerance

Having done the scheduling at the broker level, maintaining a lookup table of VM allocation and tallying it with the information received from the VMs about finished tasks results in earlier detection of faults.

4.5 Discussion of Results

The performance of the proposed algorithm (MMWRR+) closely rivals the top - performing MAX - MIN in makespan and closely approaches the best (MIN - MIN) in waiting time. Additionally, it outperforms MIN - MIN and WRR in load balancing and its performance is nearly on par with the best (MAX - MIN). In totality, proposed algorithm gives better results when all the three - performance metrics is considered.

5. Conclusion

In conclusion, this research introduces an innovative and improved scheduling algorithm "MMWRR+" that outperforms traditional approaches like Weighted Round Robin and Max - Min. Through comprehensive experimentation and analysis, it has been demonstrated that the proposed algorithm effectively minimizes the makespan, reduces waiting times for tasks, and significantly enhances load balancing in parallel computing environments. By considering dynamic workload variations and optimizing task assignments based on a sophisticated weighting mechanism, the new algorithm achieves superior performance, making it a promising solution for task scheduling in diverse computing systems. The findings presented in this study contribute valuable insights to the field of parallel computing, offering a robust alternative that can lead to enhanced efficiency, resource utilization, and overall system performance. With its demonstrated advantages, this novel scheduling algorithm holds great potential for real - world applications, paving the way for more efficient and effective task scheduling in parallel computing environments.

6. Future Scope

Further research and practical implementation are recommended to fully explore the algorithm's capabilities and validate its performance under non time - critical scenarios including parameters such as energy consumption, network costs, data availability using redundancy, etc. Fault tolerance can be testedthrough fault injection at various levels and mitigating it through migration, replication, and monitoring.

References

[1] Yichuan Jiang, Senior Member, IEEE, "A Survey of Task Allocation and Load Balancing in Distributed Systems", IEEE Transactions on Parallel and Distributed Systems, Vol.27, no.2, February 2016.
[2] Pooja Kathalkar, A. V. Deorankar, "Challenges & Issues in Load Balancing in Cloud Computing", International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321 - 9653; IC Value: 45.98; SJ Impact Factor: 6.887, Volume 6 Issue IV, April 2018
[3] XiaoxunZhong, LianmingZhang, and YehuaWei, "Dynamic Load - Balancing Vertical Control for a Large - Scale Software - Defined Internet of Things", IEEE Access, September 23, 2019, Digital Object Identifier 10.1109/ACCESS.2019.2943173

[4] Hatim Gasmelseed Ahmed, RRamalakshmi, "Performance Analysis of Centralized and Distributed SDN Controllers for LoadBalancing Application", Proceedings of the 2nd International Conference on Trends in Electronics and Informatics (ICOEI 2018) IEEE Conference Record: # 42666; IEEE Xplore ISBN: 978 - 1 - 5386 - 3570 - 4.

[5] Suchintan Mishra, Manmath Narayan Sahoo, Sambit Bakshi, Senior Member, IEEE, Joel J. P. C. Rodrigues, Fellow, IEEE, "Dynamic Resource Allocation in Fog - Cloud Hybrid Systems using Multi - criteria AHP Techniques", DOI 10.1109/JIOT.2020.3001603, IEEE Internet ofThings Journal.

[6] Belal Ali Al - Maytami, Pingzhi Fan1, Abir Hussain, Thar Baker and Panos Liatsis, "A Task Scheduling Algorithm With ImprovedMakespan Based on Prediction of TasksComputation Time algorithm forCloud Computing", IEEE Access, October 21, 2019, Digital Object Identifier 10.1109/ACCESS.2019.2948704.

[7] Alireza Sadeghi Milani, Nima Jafari Navimipour, "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends", Journal of Network andComputerApplications71 (2016) 86–98.

[8] D. Chitra Devi and V. RhymendUthariaraj, "Load Balancing in Cloud Computing EnvironmentUsing Improved Weighted Round Robin Algorithm forNonpre - emptive Dependent Tasks", Hindawi Publishing Corporation, Scientific World Journal, Volume 2016, Article ID 3896065, 14 pages, http: //dx. doi. org/10.1155/2016/3896065.

[9] Vadivel R, SudalaiMuthu T, "An effective HPSO - MGA Optimization Algorithm for Demand Based Resource Allocation in Cloud Environment", 6th International Conference on Advanced Computing and Communication Systems, 2020.

[10] Ajay Jangra, Neeraj Mangla, " An efficient load balancing framework for deploying resource scheduling in cloud - based communication in healthcare", Measurement: Sensors 25 (2023) 100584.

[11] Majid Derakhshan, Zohreh Bateni, "Optimization of Tasks in Cloud Computing Based onMax - Min, Min - Min and Priority", 4th International Conference on Web Research (ICWR), 2018.

[12] Shanchen Pang, Wenhao Li, Hua He, Zhiguang Shan, And Xun Wang"An EDA - GA Hybrid Algorithm for Multi - ObjectiveTask Scheduling in Cloud Computing", Special section on innovation and application of intelligent processing ofData, information and knowledge as resources in edge computing, IEEE Access, October 2019.

[13] Altaf Hussain, Muhammad Aleem, Muhammad Azhar Iqbal, Muhammad Arshad Islam, "SLA-RALBA: cost-efficient and resource-aware loadbalancing algorithm for cloud computing ", The Journal of Supercomputing (2019) 75: 6777–6803, https: //doi. org/10.1007/s11227 - 019 - 02916 - 4, June 2019.

[14] He XioShan, Sun XionHe, Gregor Von Laszewski, "QoS Guided Min - Min Heuristic for Grid Task Scheduling", Journal of Computer Science and Technology, July 2003.

[15] Pillareddy VamsheedharReddy and Karri Ganesh Reddy, "A Multi - Objective Based Scheduling Frameworkfor Effective Resource Utilization in CloudComputing", April 2023. IEEE Access, 10.1109/ACCESS.2023.3266294.

[16] Huankai Chen, Professor Frank Wang, Dr Na Helian, Gbola Akanmu, "User - Priority Guided Min - Min Scheduling AlgorithmFor Load Balancing in Cloud Computing", IEEE, National Conference on Parallel Computing Technologies, 2013.

[17] James Olmsted, Eyhab Al - Masri, "FogWeaver: Task Allocation Optimization Strategy across Hybrid Fog Environments", 3rd IEEE International Conference on Knowledge Innovation and Invention, 2020.

[18] Shu - Ching Wang, Kuo - Qin Yan, Wen - Pin Liao and Shun - Sheng Wang, "Towards a Load Balancing in a Three - level Cloud Computing Network", IEEE, 3rd International Conference on Computer Science and Information Technology, 2010.

[19] Sohaib Manzoor, Ze Chen, Yayu Gao, Xiaojun Hei andWenqingCheng, "Towards QoS - Aware Load Balancing for HighDensity Software Defined Wi - FiNetworks", IEEE Access, July 2020, DOI: 10.1109/ACCESS.2020.3004772.

[20] Weikun Wang, Giuliano Casale, "Evaluating Weighted Round Robin Load Balancing for Cloud Web Services", IEEE, 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2014.

[21] Saeed Parsa and Reza Entezari - Maleki, "RASA: A New Task Scheduling Algorithm in Grid Environment", World Applied Sciences Journal 7 (Special Issue of Computer & IT): 152 - 160, 2009, ISSN 1818.4952

[22] Rahul Mishra, Gaurav Mitawa, "Improved Round Robin Algorithm for effective Scheduling Process for CPU" (2021), Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021). IEEE Xplore Part Number: CFP21ONG - ART; 978 - 0 - 7381 - 1183 - 4

[23] BING HU, (Senior Member, IEEE), FUJIE FAN, (Student Member, IEEE), KWAN L. YEUNG, (Senior Member, IEEE), AND SUGIH JAMIN (2018), "Highest Rank First: A New Class of Single - iteration Scheduling Algorithms for Input - queued Switches" IEEE Access, DOI: 10.1109/ACCESS.2017

[24] Mohammad Oqail Ahmad and Rafiqul Zaman Khan (2019), "Cloud Computing Modelling and Simulation using CloudSim Environment", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277 - 3878, Volume - 8 Issue - 2, July 2019.

[25] Shahbaz Afzal, G. Kavitha (2019), "Load balancing in cloud computing – A hierarchical taxonomical classification", Journal of Cloud Computing: Advances, Systems and Applications https: //doi. org/10.1186/s13677 - 019 - 0146.

[26] Komal Mahajan, AnsuyiaMakroo and Deepak Dahiya (2013) "Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud Based Infrastructure Journal of Information Processing Systems", DOI: 10.3745/JIPS.2013.9.3.379.

[27] Altaf Hussain, Muhammad Aleem, Muhammad Arshad Islam, Muhammad Azhar Iqbal (2018), "A Rigorous Evaluation of State - of - the - Art Scheduling Algorithms for Cloud Computing", IEEE Access, DOI: 10.1109/ACCESS.2018.2884480.
[28] Mung Chiang, *Fellow*, *IEEE*, and Tao Zhang, *Fellow*, *IEEE* (2016), "Fog and IoT: An Overview of Research Opportunities", IEEE Internet of Things Journal, vol.3, no.6, December 2016.
[29] Bharat Khatavkar, Prabadevi Boopathy, "Efficient WMaxMin Static Algorithm For LoadBalancing In Cloud Computation", International Conference on Innovations in Power and Advanced Computing Technologies, 2017.
[30] Xiaofang Li, Yingchi Mao, Xianjian Xiao, Yanbin Zhuang, "An Improved Max - Min Task - Scheduling Algorithm for Elastic Cloud", International Symposium on Computer, Consumer and Control, 2014.
[31] S. VaaheedhaKfatheen, Dr. M. Nazreen Banu, "MiM - MaM: A new scheduling algorithm for grid environment", International Conference on Advances in Computer Engineering and Applications, 2015.
[32] Sikha Suhani Bhuyan, Ashis Kumar Mishra, "A Comparative Analysis Of Task Scheduling Algorithms Through CloudSim", International Journal of Creative Research Thoughts, 2022 IJCRT | Volume 10, Issue 7 July 2022 | ISSN: 2320 - 2882.
[33] O. M. Elzeki, M. Z. Reshad, M. A. Elsoud, "Improved Max - Min Algorithm in Cloud Computing", International Journal of Computer Applications (0975 – 8887) Volume 50 – No.12, July 2012.
[34] Kaushik Mishra and Santosh Kumar Majhi, "A State - of - Art on Cloud Load Balancing Algorithms", International Journal of Computing and Digital Systems ISSN (2210 - 142X) Int. J. Com. Dig. Sys.9, No.2 (Mar - 2020).
[35] Pandaba Pradhan, Prafulla Ku. Behera, B. N. B. Ray, "Improved Max - Min Algorithm for ResourceAllocation in Cloud Computing", Sixth International Conference on Parallel, Distributed and Grid Computing, 2020.
[36] Chandrasekhar Salimath, Bhupender Parashar, " Operations Research", Universities Press, 2014, ISBN 978 - 81 - 7371 - 931 - 8, Pages 297 - 326.

Author Profile

Suvarna N A, received her B. E degree in (E & C) Engineering from Mysore University in 1987 and M. Tech from UP Technical University in 2010. She is currently pursuing Ph. D from GD Goenka University, Gurugram, India, She has worked as Software Consultant for World Band Aided Project, Software Lead in IT Industry, and as Faculty in Engineering for more than a decade. Her research areas and interests include Grid Computing, Cloud Computing, Distributed Processing, Algorithms and Data Processing. She has also authored a book on "Programming Data Structures Using 'C'" on Amazon KDP.