DYNAMIC TASK SCHEDULING BASED ON BURST TIME REQUIREMENT FOR CLOUD ENVIRONMENT

Linz Tom[1] and Bindu V.R.[2]

[1]Dept. of Computer Science, Assumption College, Changanacherry, Kerala
[2]School of Computer Sciences, Mahatma Gandhi University, Kottayam, Kerala

ABSTRACT

Cloud computing has an indispensable role in the modern digital scenario. The fundamental challenge of cloud systems is to accommodate user requirements which keep on varying. This dynamic cloud environment demands the necessity of complex algorithms to resolve the trouble of task allotment. The overall performance of cloud systems is rooted in the efficiency of task scheduling algorithms. The dynamic property of cloud systems makes it challenging to find an optimal solution satisfying all the evaluation metrics. The new approach is formulated on the Round Robin and the Shortest Job First algorithms. The Round Robin method reduces starvation, and the Shortest Job First decreases the average waiting time. In this work, the advantages of both algorithms are incorporated to improve the makespan of user tasks.

KEYWORDS

Cloud computing, dynamic task scheduling, response time, waiting time, Round Robin algorithm.

1. INTRODUCTION

Advancement in the field of internet technology generated the computing paradigm called cloud computing. It blends computer software and infrastructure resources that can be customized according to user requirements. These resources are accessed depending upon the requirement. Commercially there exist innumerable service providers. Microsoft Azure, Amazon EC2, Google App Engine are some of the giant resource providers. They offer a pay-as-you-use model. Different cloud deployment models based on accessibility and location are private, public, hybrid, community, and federated. Cloud service models include IaaS, PaaS, and SaaS. A Cloud computing environment enables users to have good performance per price.

Cloud computing may also refer to the global access of configurable resources whereby users can access cloud computing services from anywhere using the internet. Attributes of cloud systems are scalability, accessibility, and flexibility[1]. Self-service provisioning is deployed in this environment. The salient feature is the provision with which users can demand their required resources anytime and anywhere. Virtualization techniques are prioritized and employed in cloud computing.

Task scheduling is the process by which a stream of tasks is assigned to the available resources. It can influence the resource utilization as well as the processing cost of a cloud system. The scalability of the cloud environment makes the scheduling process very complex [2]. It is complex to design an optimal task scheduling algorithm as every client's size and resource requirements differ. There exist different metrics to evaluate the efficiency of scheduling algorithms. Some of them are makespan, Waiting time, throughput, resource utilization, completion time, and energy consumption. The ultimate aim of scheduling is to minimize the average waiting time, response time, and execution time, thereby improving system throughput and quality of service.

Scheduling can be done statically or dynamically. Static scheduling can be applied if the information of tasks for scheduling is known initially. If not, dynamic scheduling is only possible. There are heuristic and metaheuristic algorithms available for scheduling. First Come First Serve(FCFS), Shortest Job First(SJF), and Round Robin(RR) are some of the traditional algorithms [3] employed for task scheduling in a cloud environment. Each of these methods hasits pros and cons. This article tries to combine Round Robin and Shortest Job First to lessen the overall makespan of user-submitted tasks.

The rest of the paper is structured as different sections. Section 2 features a description of related works. Section 3 introduces the system model, and the proposed method is described in section 4. Experimental setup and analysis of algorithm evaluation are elaborated in section 5. Finally, in section 6 conclusion and future scope are explained briefly.

2. RELATED WORK

To increase the performance of cloud systems, researchers have proposed different task scheduling algorithms. Different metrics used for evaluating the efficiency of algorithms include makespan, waiting time, throughput, load balancing, resource utilization, energy consumption response time, and many more. Some of the algorithms are single objective, while others are multi-objective. This section describes the different scheduling algorithms and their contribution to system performance.

The article [4] proposed an improved Shortest Job First scheduling for minimizing competition time and average response time. This method maximizes resource utilization too. They compared the proposed algorithm with the Shortest Job First and First Come First Served methods. In [5], a two-stage dynamic cloud task scheduling has been proposed. In this method, by using historical data of task scheduling VM's are pre-created. Each VM has a different set of resources; when a new task arrives, the most fitting VM's are allocated, so that time for scheduling is reduced. This method is compared with min-min and max-min for guarantee ratio, makespan, failure rate, and utilization rate. The proposed algorithm achieved a lower failure rate, higher utilization rate, and VM load balancing.

The authors [6] developed an EDA-GA hybrid task scheduling algorithm, and the method used the estimation of distribution algorithm and genetic algorithm. The advantage of this hybrid method is fast convergence speed and good ability for searching. This algorithm is compared with EDA and GA algorithms. The experimental analysis showed that the new method exhibited more excellent performance for the metrics like task completion time and load balancing. The article [7] introduced a hybrid bio-inspired task scheduling algorithm. In this work, VM allocation of the task is done using modified particle swarm optimization and distribution of resources for the tasks using the hybrid bio-inspired algorithm. The proposed method is compared with Throttled, RR, ACO, and Exact algorithms. This new method reduced execution time and increased resource utilization.

This article [8] put forward a many-objective optimization algorithm based on hybrid angles. The four objectives laid on for performance evaluation are time, cost, resource utilization, and load balancing. The proposed MaOEA-HA algorithm is compared with NSGAIII, GrEA, KnEA, VaEA, and Two-Arch2. It is found that this algorithm achieved greater performance compared to other algorithms. An improved firework algorithm [9] was suggested for task scheduling. The algorithm experimented with clustering of tasks, completion time, and server load. With the increase in the number of tasks, the execution time gets reduced compared to other algorithms,and also a faster convergence speed is achieved.

The authors implemented the prediction of task computation time [10] in the article. The quality measures used for performance evaluation are makespan, speedup, and efficiency. Near-optimal allocation of tasks to processors is achieved in this method. The algorithm is compared with min-min, max-min QoS_guided, and MoM-MoM algorithms. The proposed method achieves the reduction of makespan and execution time. This article[11] compared two heuristic algorithms ICPCP and SCS, with two metaheuristic algorithms, particle swarm and CSO. Metrics used for evaluation are makespan, execution time, and cost for execution. An improved max-min algorithm is introduced [12] by the authors. With the use of this method, the competition time of tasks gets reduced. This algorithm is compared with max-min, RR, and min-min algorithms. According to this strategy, tasks are clustered using some machine learning techniques. Simulation results showed that processing time gets reduced and QoS gets improved.

In article[13], the authors suggested improving the Cat Swarm Optimization technique by applying linear descending inertia weight. LDIW enhanced convergence speed and was trapped in local searching. Experimental results showed significant improvement of makespan. A hybrid of Best-fit particle swarm optimization and Tabu search algorithm [14] is proposed in this article. Instead of the random method, the initial population is generated using the best-fit policy. TS avoids trapping in optimum local search. Metrics used for performance evaluation are execution time, utilization of resources, and cost. Simulation results exhibited that the proposed method outperforms the PSO method. The work [15] introduced an algorithm that dynamically schedules tasks according to their types. This strategy aims to reduce makespan and average waiting time.

This article [16] suggested a combined approach of cuckoo search and oppositional-based learning. The proposed strategy tried to minimize execution cost and makespan. This method is compared with particle swarm optimization, genetic algorithm, and improved differential evolution algorithm. The authors [17] tried to compare the most popular static scheduling algorithms like First Come First Serve, Max-Min, and Shortest Job First. The CloudSim simulator is used for studying various metrics like the complexity of algorithms, resource availability, task execution time, waiting time, and finish time.

The Authors suggested a dynamic particle swarm optimization algorithm in this work [18], and the work emphasized the utilization of bandwidth and memory. This algorithm is simulated on Cloudsim and compared with the PSO algorithm. The modified algorithm optimized the makespan and utilization of bandwidth. This article [19] provided a dynamic scheduling algorithm that considered the processing power, cost of execution, and the number of tasks currently running. The proposed algorithm is compared with Round Robin and minimum completion time algorithms. This strategy includes two versions. In the first version, no new VMs are created. But for the second version, new VMs are created whenever it is necessary.

The Authors proposed [20] an intelligent algorithm based on Imperialist competitive algorithm and firefly algorithm. This metaheuristic approach strives to improve makespan, CPU time, and load-balancing. In this article [21], a hybrid method based on an estimation of distribution algorithm and genetic algorithm is developed. The sampling and probability model of EDA and mutation and crossover of GA is incorporated in the new algorithm to find the optimal scheduling solution of tasks. Simulation results showed that the new algorithm reduced the competition time and improved the load-balancing. The authors suggested a multi-objective task scheduling algorithm [22], and the different objectives are load balancing, execution time, and cost. Each objective is given an equal priority. A rank strategy is applied to improve completion time and makespan.

An improved particle swarm optimization is suggested in this article [23]. The disadvantage of PSO is that as the number of tasks increases, the efficiency decreases. The IPSO algorithm gives optimum scheduling even for a large number of tasks. The incoming tasks are divided into batches dynamically. After obtaining a sub-optimal solution for each set, the result is combined to form the final solution. The authors [24] suggested a pair-based scheduling algorithm based on the Hungarian algorithm. The utmost aim of this strategy is to reduce layover time. The simulated algorithm is compared with FCFS, the Hungarian method with lease time, and converse lease time.

This article [25] compared different task scheduling algorithms based on the parameters like response time, load-balancing, makespan, etc. This work suggested that existing algorithms may incorporate more parameters to improve their overall performance. The authors [26] reviewed many task scheduling algorithms. According to this, an efficient scheduler should consider the varying nature of the cloud computing environment. This work [27] proposed an enhancement of the backfilling algorithm. The VIKOR method is used to resolve the conflicts among the same type of tasks. Experimental results showed an improvement in resource utilization and task rejection rate.

The authors [28] proposed a combination of a fuzzy model and PSO technique. In this work, the fuzzy strategy is applied for fitness evaluation, and the disadvantages of PSO are overpowered by applying crossover and mutation. The simulation indicated that the proposed method reduced execution time and the degree of imbalance. An online scheduling strategy is used in this work [29]. According to this method, a set of tasks are assigned to the VM by considering their processing capacity. Experimental results showed that the proposed strategy improved execution time, degree of load-balancing, and resource utilization. The article [30] proposed a model for task scheduling with load balancing. The simulation results manifested that the new model minimized the makespan and improved resource utilization. A new metaheuristic method is implemented in this work [31]. By applying the whale optimization technique, the convergence speed has improved.

Large-scale optimization for cloud workflow scheduling is proposed in this article [32]. A dynamic group learning methodology is implemented to enhance system balance, and an effective cost-performance optimization technique is suggested by the authors. A survey [33] of task scheduling algorithms indicated that an efficient method should reduce makespan, achieve load balancing and minimize energy consumption. The authors [34] proposed an improved Round Robin scheduling with varying time quantum. Experimental results showed an efficiency increase in turnaround and waiting time. The article [35] suggests that the key objective of any scheduling algorithm is to reduce execution time and thereby increase throughput. In the year 2016, the authors [36] emphasizes the importance of virtualization in cloud systems.

From the analysis of the above review, we deduce that the performance of scheduling algorithms heavily depends upon the input task size, which is unpredictable in cloud computing scenarios. A slight reduction in makespan improves overall system efficiency. In the proposed work, we tried to optimize multiple criteria like response time, context switches, waiting time, and makespan

3. SYSTEM MODEL

Task scheduling plays a crucial role in cloud system performance. Users submit their tasks to the cloud broker and are mapped onto the appropriate VMs for execution. The mapping is done to improve the QoS parameters. Fig1 shows the overview of the task scheduling framework. In cloud systems, task scheduling is done either in a centralized or distributed manner. Users submit the tasks(T_1, T_2,…, T_n) to the system, and the tasks are characterized by their input file size and output file size and put into a queue. Then the dynamic task scheduler component of the system calculates the expected completion time and allocates tasks to VMs. Details regarding the available computing resources are provided by the resource information system. All the required information regarding the capacity of hosts and virtual machines available in the cloud is stored and maintained in the resource information system. A host is characterized by storage, bandwidth, and memory and the resource information system provides the host information to the dynamic task scheduler for task allocation. It is the responsibility of the dynamic task scheduler to make decisions regarding scheduling strategy to implement task allocation. This scheduler schedules tasks dynamically, and the VMs run on the physical hosts.

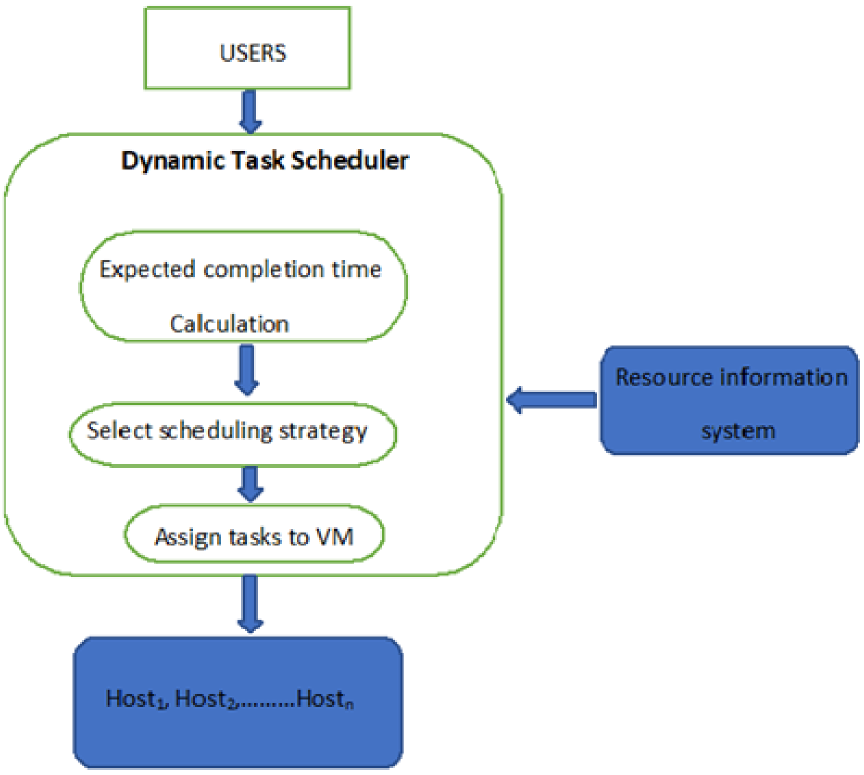
USERS
Dynamic Task Scheduler
Expected completion time Calculation
Select scheduling strategy
Assign tasks to VM
Resource information system
Host1, Host2,........Hostn

Fig 1. Task scheduling framework.

In this section, a hybrid Round Robin and the Shortest Job First method is described. The algorithms are determined based on the burst time requirements of the submitted tasks. In static Round Robin scheduling, different task attributes do not have any role in the scheduling policy. The method assigns the task a predefined time slice for execution. If the time- slice is too small, then the number of context switches is higher, reducing system utilization. At the same time, a large time slice increases the average waiting time and response time. A dynamic time-slicing strategy is implemented to improve overall system efficiency.

4. Proposed Method

A hybrid approach to the task scheduling problem is presented in this section. This method comprises two parts. Firstly dynamic property is incorporated in the Round Robin algorithm by varying the time slice. By this, the number of context switches gets reduced.

Steps for dynamic time slicing

Step1: Calculate the average burst time of tasks in the ready queue.

Step2: Divide the ready queue into multiple queues according to the average value.

Step3: Assign a time slice to each task by considering their burst time requirements.

Step4: Execute tasks on available VMs.

This method is a simple preemptive strategy that significantly reduces the processing overhead for context switching. Here, the authors are trying to improve the traditional method by dynamically varying time slices.

Secondly, hybridization of improved Round Robin and Shortest Job First is achieved for reducing makespan, waiting time, and response time. In the proposed algorithm, TList is the set of tasks submitted to the cloud system, and VList, the set of available virtual machines created in the host machines. These two sets are kept sorted in descending order of the processing capacity, represented in MIPS. Initially, tasks in the TList are assigned to the VM's. In each VM, the scheduling algorithm selection is made depending on the completion time requirements of tasks. This process is repeated until TList becomes empty.

Algorithm: Dynamic Task Scheduling Based on Completion Time(DTBCT)

```
1.  Input: List of tasks (TList), List of virtual machines (VList).
2.  BEGIN
3.    Sort TList and VList in descending order of  MIPS.
4.    for each task in TList
5.      do
6.         Bind tasks to VM in sorted order
7.    end for
8.    Calculate the average burst time of tasks in each waiting queue
9.    Schedule tasks using Round Robin with dynamic time slices.
10.   Calculate the completion time for all tasks using Shortest Job First and Round  Robin.
11.   Migrate a task to a VM with minimum completion time.
12.   Set scheduler according to the expected completion time
13.   Repeat steps 8 to 12 until TList is empty.
14. END.
```

5. EXPERIMENTAL RESULTS

5.1. Experimental Setup

The CloudSim Toolkit (Simulation platform) is used [37] to evaluate the performance of the proposed strategy. This software simulates a real-time cloud computing environment. Users can directly conduct performance testing of different scheduling algorithms in CloudSim. The infrastructure of cloud systems can be simulated using the datacenter entity in CloudSim, and the data center comprises one or more hosts. Each of these hosts may be assigned several VMs. The user tasks are assigned to these VMs. All of these entities are characterized by some parameters. Table 1 lists the parameters and values used in this simulation environment.

Table 1. Simulation entity details

Entities	**Parameters involved**	**Values**
Task	Number of tasks	10 -1000
	Input file size	500-35000
	Output file size	500-35000
Host	Storage	1000000 MB
	Bandwidth	10000 Megabits/s
	RAM	2048MB
VM	Number of VMs	2-16
	Policy	Time-shared
	Number of PEs	1
	Bandwidth	10000 Megabits/s
Datacenter	Number of CPUs	2-4
	Number of data center	1-4
	Number of Hosts	2-8
	VMM	Xen
	Operating System	Linux
	System Architecture	X86

The metrics used for analyzing the algorithms are makespan, waiting time, response time, and context switches. Our method was compared with Round Robin, the Shortest Job First, and First Come First Serve algorithms.

5.2. Number of Context Switches

In the preemptive scheduling strategy, context switching refers to the event by which a task is moved from the execution state to the ready state. When the number of context switches increases, the efficiency of the algorithm decreases. In this proposed method, the authors tried to decrease the number of context switches of the Round Robin algorithm.

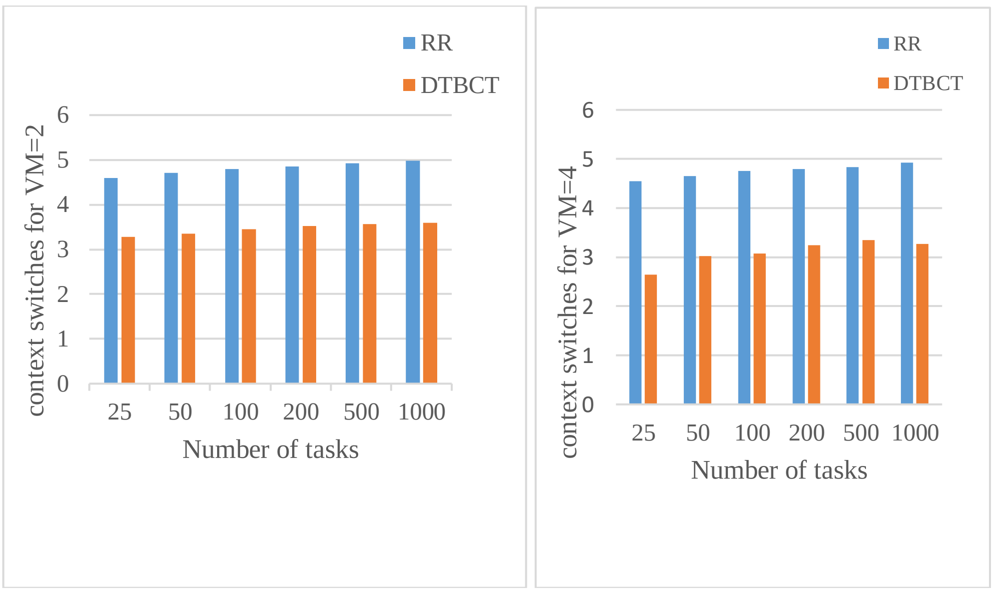

Fig 2. Average number of context switches with VM=2 & VM=4

Fig2 shows the average number of context switches for Round Robin (RR) and Dynamic Task Scheduling Based on Completion Time (DTBCT). By applying dynamic time-slicing in the proposed algorithm, we decreased the number of context switches, and the time for context switching can be used for productive computations. These algorithms are executed with the number of tasks 25, 50, 100,200,500, and 1000. For VMs = 2, context switches were reduced by17%, and for VM=4, the proposed method outperformed by 21%.

5.3. Average Response Time

Response time is the duration from task allocation to start execution. This time plays a pivotal role in user satisfaction. As the task size increases, response time increases for the Shortest Job First. By the proposed task scheduling algorithm, the average response time is decreased. The graphs in the following figure show the performances of the proposed method Dynamic Task Scheduling Based on Completion Time (DTBCT) and Shortest Job First algorithm (SJF).

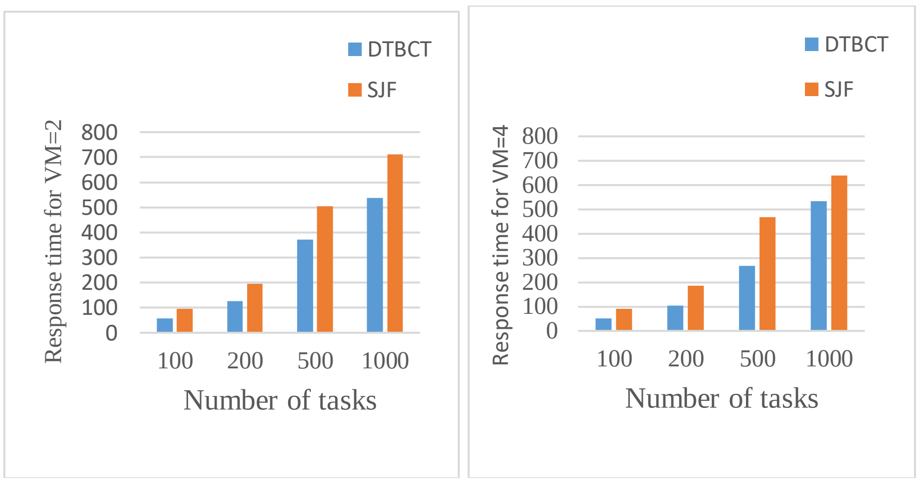

Fig 3. Average response time with VM=2 & VM=4

Fig3 illustrates the average response time obtained for the number of virtual machines equal to two and four. When the number of VMs increased from two to four, the average response time of both algorithms significantly reduced. Two algorithms have been executing with the number of tasks 100, 200, 500, and 1000. With the hybridization of two algorithms RR and SJF, in the proposed method DTBCT, the average response time has been reduced by 15% and 19% for VMs two and four, respectively.

5.4. Average Waiting Time

Waiting time is the total time the task is spent in the ready queue for execution. The QoS of the cloud system has improved by reducing the waiting time. Switching between RR and SJF by considering the task's expected completion time, the proposed method outperforms other algorithms. The following figures illustrate the performance enhancement of the proposed strategy.

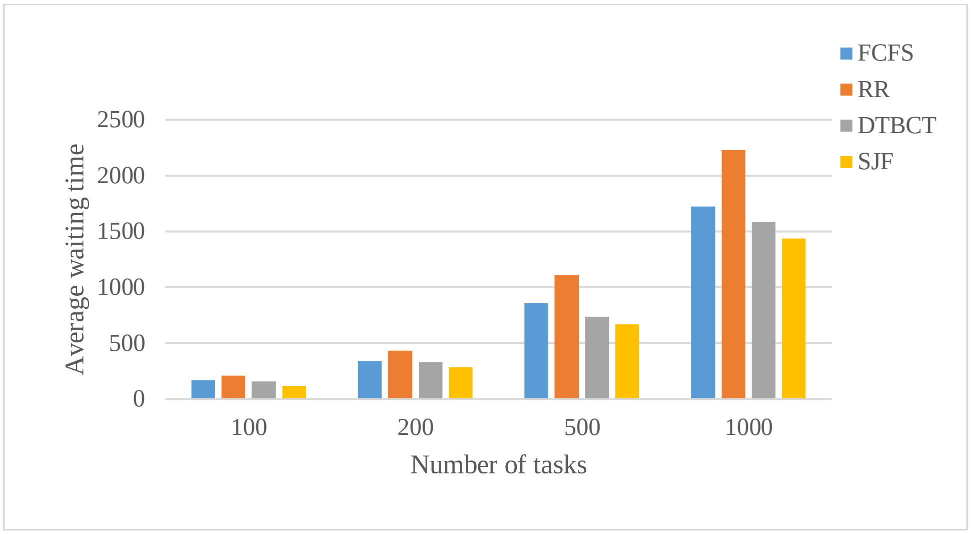

Fig 4. Average waiting time with VM=2

Fig4 illustrates the average waiting time obtained for two virtual machines. The waiting time is optimal for the Shortest Job First algorithm. But when compared with RR and FCFS, the average waiting time is less for DTBCT and tending towards the optimum value. The algorithms have been executing with the number of tasks 100, 200, 500, and 1000. When comparing the performance, DTBCT is 10% efficient than RR.

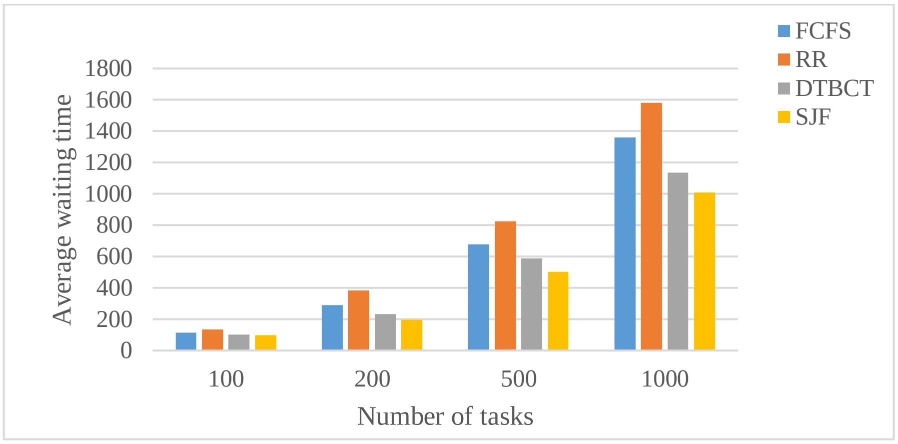

Fig 5. Average waiting time with VM=4

Fig5 shows the performance of algorithms for the average waiting time obtained for the number of virtual machines equal to four. The algorithms are executed with the number of tasks 100, 200, 500, and 1000. In terms of efficiency, the average waiting time value for DTBCT is much less than RR and FCFS. Since the average waiting time for SJF is always a minimum, we tried to achieve the same in the proposed method. On average, a 9% reduction in waiting time is achieved for DTBCT when compared with RR.

5.5. Average Makespan

Makespan is a seminal metric to evaluate the performance of scheduling algorithms. It is the time that elapses from the start execution of a set of tasks to their competition. In the proposed method, the makespan is improved by switching between RR and SJF. The new method is compared with FCFS, RR, and SJF.

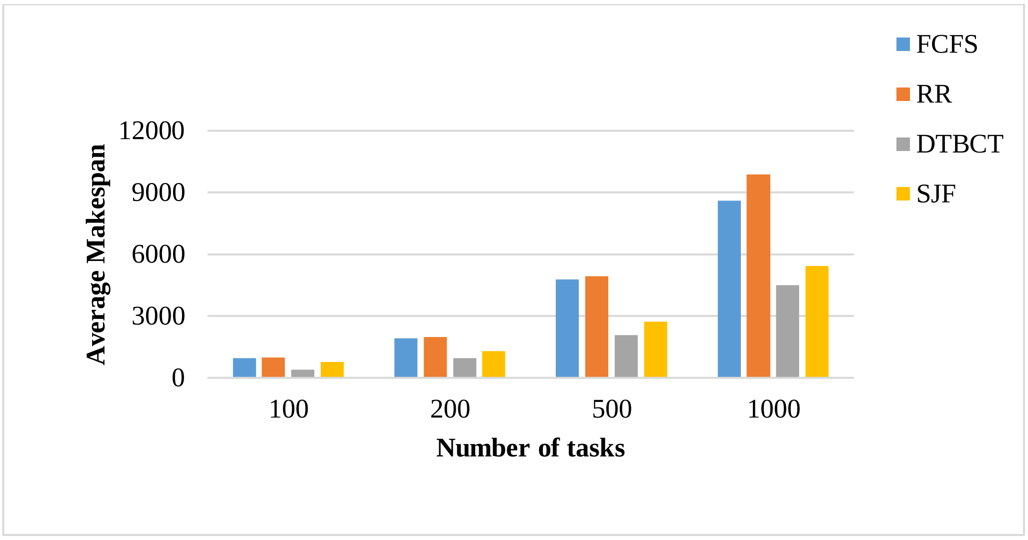

Fig 6. Average Makespan with VM=2

Fig6 demonstrates the comparative performance analysis of algorithms for the average makespan obtained for two virtual machines. The algorithms are executed with the number of tasks 100, 200, 500, and 1000. As the number of tasks increases, there is a significant decrease in the makespan for the proposed strategy when compared with other algorithms. For all the sets of tasks, a decrease in average makespan is achieved for the proposed method DTBCT.

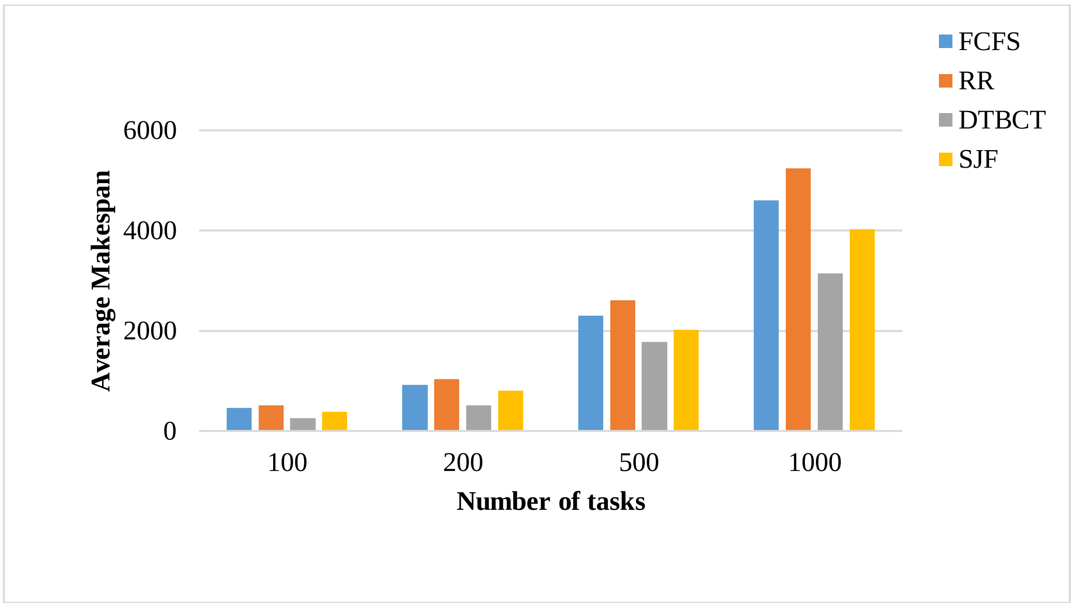

Fig 7. Average Makespan with VM=4

Fig7 shows the comparison of FCFS, RR, and SJF with the proposed method DTBCT for four VMs. The algorithms are executed with the number of tasks 100, 200, 500, and 1000. As with the case VM equal to two, there is a relative decrease in the average makespan for the proposed algorithm. The average makespan is reduced considerably for each set of tasks.

This section compared and analyzed our proposed algorithm with Round Robin, First Come First Served, and Shortest Job First methods. From the results, it is evident that the dynamic time-slicing reduced average waiting time and had a significant impact on decreasing makespan. Our method tried to reduce the number of context switches, average response time, and waiting time, which in turn decreased the average makespan. While considering the metrics, the proposed method DTBCT outperformed all algorithms.

6. CONCLUSION

Task Scheduling is becoming a complex activity with the increasing number of cloud system users day by day. The efficiency, as well as QoS of cloud computing systems, is dependent on the effectiveness of task scheduling algorithms. The scheduling algorithms enhancing a single metric is not enough to improve the overall system performance. In this work, a hybrid approach that improves system efficiency significantly is proposed. This method is compared with the existing static algorithms like RR, FCFS, and SJF. The proposed method incorporates the merits of RR and SJF to improve task scheduling in terms of response time and waiting time. Simulation results showed that the new method outperformed these algorithms in terms of the metrics average response time, number of context switches, average waiting time, and average makespan. For the time being, this work considered only four metrics, but in the future more metrics like task deadline and cost of processing, etc., can also be considered. Further, we intend to shift our focus to implement the algorithm in a real-time cloud environment.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of "DST-PURSE Phase II" from, Government of India.

REFERENCES

[1] Chavan, V., Kaveri, P. R., Peshkar, M., & Dhole, K., (2020) "Clustering and Scheduling Tasks for Higher Utilization of Cloud Resources.", 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM) (pp. 386-390).

[2] Qadir, O. A., & Ravi, D. G., (2020) "Dual Objective Task Scheduling Algorithm in Cloud Environment.", International Journal of Advanced Trends in Computer Science and Engineering, 9(3).

[3] Liu, R., Liu, S., & Wang, N., (2020) "Optimization Scheduling of Cloud Service Resources Based on Beetle Antennae Search Algorithm.", 2020 International Conference on Computer, Information and Telecommunication Systems (CITS) (pp. 1-5).

[4] Alworafi, M. A., Dhari, A., Al-Hashmi, A. A., &Darem, A. B., (2016) "An improved SJF scheduling algorithm in cloud computing environment.", 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT) (pp. 208-212).

[5] Zhang, P., & Zhou, M., (2017) "Dynamic cloud task scheduling based on a two-stage strategy." ,IEEE Transactions on Automation Science and Engineering, 15(2), 772-783.

[6] Pang, S., Li, W., He, H., Shan, Z., & Wang, X., (2019) "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing." IEEE Access, 7, 146379-146389.

[7] Domanal, S. G., Guddeti, R. M. R., &Buyya, R., (2017) "A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment.", IEEE transactions on services computing, 13(1), 3-15.

[8] Geng, S., Wu, D., Wang, P., & Cai, X., (2020) "Many-objective cloud task scheduling", IEEE Access, 8, 79079-79088.

[9] Wang, S., Zhao, T., & Pang, S., (2020) "Task scheduling algorithm based on improved firework algorithm in fog computing", IEEE Access, 8, 32385-32394.

[10] Al-Maytami, B. A., Fan, P., Hussain, A., Baker, T., &Liatsis, P., (2019) "A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing", IEEE Access, 7, 160916-160926.

[11] Maurya, A. K., & Tripathi, A. K., (2018) "Deadline-constrained algorithms for scheduling of bag-of-tasks and workflows in cloud computing environments.", Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications (pp. 6-10).

[12] Hung, T. C., Hieu, L. N., Hy, P. T., & Phi, N. X, . (2019) "MMSIA: improved max-min scheduling algorithm for load balancing on cloud computing.", Proceedings of the 3rd International Conference on Machine Learning and Soft Computing (pp. 60-64).

[13] Gabi, D., Ismail, A. S., &Dankolo, N. M., (2019)" Minimized makespan based improved cat swarm optimization for efficient task scheduling in cloud datacenter", Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference (pp. 16-20).

[14] Alkhashai, H. M., &Omara, F. A., (2016) "An enhanced task scheduling algorithm on cloud computing environment", International Journal of Grid and Distributed Computing, 9(7),91-100.

[15] Mazumder, A. M. R., Uddin, K. A., Arbe, N., Jahan, L., &Whaiduzzaman, M, (2019) "Dynamic task scheduling algorithms in cloud computing.", 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1280-1286). IEEE.

[16] Krishnadoss, P., & Jacob, P., (2018) "OCSA: task scheduling algorithm in cloud computing environment.", International Journal of Intelligent Engineering and Systems, 11(3), 271-279.

[17] Aladwani, T., (2020) "Types of Task Scheduling Algorithms in Cloud Computing Environment.", Scheduling Problems-New Applications and Trends. IntechOpen.

[18] Sudheer, M. S., & Krishna, M. D. V., (2019) "Dynamic PSO for task scheduling optimization in cloud computing.", Int J Recent TechnolEng, 8(2), 332-338.

[19] Ibrahim, E., El-Bahnasawy, N. A., &Omara, F. A., (2017) "Dynamic task scheduling in cloud computing based on the availability level of resources.", International Journal of Grid and Distributed Computing, 10(8), 21-36.

[20] Kashikolaei, S. M. G., Hosseinabadi, A. A. R., Saemi, B., Shareh, M. B., Sangaiah, A. K., &Bian, G. B., (2020) "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm.", The Journal of Supercomputing, 76(8), 6302-6329.

[21] Pang, S., Li, W., He, H., Shan, Z., & Wang, X., (2019) "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing.", IEEE Access, 7, 146379-146389.

[22] Langhnoja, H. K., &Joshiyara, H. A.. (2019) "Multi-Objective Based Integrated Task Scheduling In Cloud Computing", 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1306-1311). IEEE.

[23] Saleh, H., Nashaat, H., Saber, W., &Harb, H. M., (2018) "IPSO task scheduling algorithm for large scale data in cloud computing environment", IEEE Access, 7, 5412-5420.

[24] Panda, S. K., Nanda, S. S., &Bhoi, S. K, (2018) "A pair-based task scheduling algorithm for cloud computing environment", Journal of King Saud University-Computer and Information Sciences.

[25] Mazhar, B., Jalil, R., Khalid, J., Amir, M., Ali, S., & Malik, B. H., (2018) “Comparison of task scheduling algorithms in cloud environment", International Journal of Advanced Computer Science and Applications, 9(5), 384-390.

[26] Arora, N., (2017) "Review on Task Scheduling Algorithms in Cloud Computing Environment", International Journal of Advanced Research in Computer Science, 8(4).

[27] ayak, S. C., & Tripathy, C., (2018) "Deadline based task scheduling using multi-criteria decision-making in cloud environment.", Ain Shams Engineering Journal, 9(4), 3315-3324.

[28] Mansouri, N., Zade, B. M. H., &Javidi, M. M, (2019) "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory.". Computers & Industrial Engineering, 130, 597-633.

[29] Nasr, A. A., El-Bahnasawy, N. A., Attiya, G., & El-Sayed, A., (2018) "A new online scheduling approach for enhancing QOS in cloud.", Future Computing and Informatics Journal, 3(2), 424-435.

[30] Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J., (2020) "A woa-based optimization approach for task scheduling in cloud computing systems.", IEEE Systems Journal, 14(3), 3117-3128.

[31] Wang, Z. J., Zhan, Z. H., Yu, W. J., Lin, Y., Zhang, J., Gu, T. L., & Zhang, J., (2019) "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling.", IEEE transactions on cybernetics, 50(6), 2715-2729.
[32] Wan, B., Dang, J., Li, Z., Gong, H., Zhang, F., & Oh, S, (2020) "Modeling Analysis and Cost-Performance Ratio Optimization of Virtual Machine Scheduling in Cloud Computing", IEEE Transactions on Parallel and Distributed Systems, 31(7), 1518-1532.
[33] Tom, L., & Bindu, V. R., (2019) "Task Scheduling Algorithms in Cloud Computing: A Survey", Springer, Cham.International Conference on Inventive Computation Technologies (pp. 342-350).
[34] Alaa, F. I. A. D., Zoulikha, M. M., & Hayat, B. E. N. D. O. U. K. H. A. , (2020) "Improved Round Robin Scheduling Algorithm With Varying Time Quantum"Second International Conference on Embedded & Distributed Systems (EDiS) (pp. 33-37).
[35] Abijith, P., Kumar, K. A., Allen, R. S., Vijayakumar, D., Paul, A., & Kumar, G. P., (2021) "Intelligent Cloud Load Balancing Using Elephant Herd Optimization.", Intelligence in Big Data Technologies—Beyond the Hype (pp. 267-273).
[36] Hung, T. C., & Phi, N. X., (2016) "Study the effect of parameters to load balancing in cloud computing", International Journal of Computer Networks & Communications.33-45
[37] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., &Buyya, R., (2011) "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.", Software: Practice and experience, 41(1), 23-50.

AUTHORS

Linz Tom received her Master's Degree in Computer Science from Mahatma Gandhi University, Kerala, India, in 1996. Currently, she is a research scholar at Mahatma Gandhi University, Kerala, India, and she is an Associate Professor in the Department of Computer Science, Assumption College Autonomous, Changanacherry, Kerala, India. Her research interests include Cloud Computing and Distributed Systems.

Dr. Bindu V R received her Master's Degree in Computer Science from the University of Kerala, India, and Ph.D. degree in Computer Science from Mahatma Gandhi University, India. She joined the School of Computer Sciences, Mahatma Gandhi University, Kerala, India, in 1993 and is currently Professor and Dean, Faculty of Science, Mahatma Gandhi University. Her research interests include Digital Image Processing and Computer Vision, Machine Learning and Artificial intelligence, Cloud Computing, and Data Science. She has authored more than 40 research papers in reputed international journals and conference proceedings.