Comparison of Average Completion Time and Makespan for Various Task Scheduling Techniques

Sushant Kumar Vishnoi
Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Odisha-769008, India
sushantkumarvishnoi@gmail.com

Sanjeev Patel
Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Odisha-769008, India
patels@nitrkl.ac.in

***Abstract*—Cloud computing is becoming a trending and important technology that everyone is using globally on a large scale. It allows users to get access to resources (Storage and computational) online over the internet. With the increasing number of users, maintaining the Quality of service is challenging. The jobs are becoming more and more intensive and hence designing and implementing the correct task scheduling algorithm is very necessary to fulfill the increasing demand of users. An algorithm is needed to be designed to map the request of users to a particular Virtual Machine to utilize the resources of virtual machines efficiently. In this paper, we have observed from the results that better makespan and optimum or efficient resource usage are required for both customers and cloud service scheduling algorithms, which are considered suitable for large-scale systems. In this paper, the performance of 5 scheduling algorithms - FCFS, SJF scheduling, Round Robin scheduling, general priority algorithm, and Particle Swarm Optimization (PSO) is analyzed and compared. The performance of the various task scheduling algorithms has been carried out on the Cloudsim simulator. It has been observed from results that PSO outperforms the other compared task scheduling techniques and gives 48.3% less makespan as compared to FCFS.**

I. Introduction

Cloud Computing is the delivery of resources or services over a network that provides flexible resources and economies of scale. A cloud can be considered as a distributed system like a collection of virtualized and linked systems. In this developing market of information technology-oriented companies and agencies, to satisfy everyday developing needs, cloud computing is the best option. Figure 1 depicts the relationships between the cloud actors. Cloud service providers aim to have high levels of customer satisfaction. One of the top technologies for providing scalable computing services in a sustainable and fault-tolerant manner is cloud computing. SaaS, IaaS, and PaaS have transformed large-scale computing during the last ten years. Furthermore, many large corporations have contributed to private and hybrid cloud arrangements while some have worked to create more affordable and dependable public cloud solutions.

It is challenging to assess the performance of such platforms due to the high infrastructure costs. This gap is filled in part by simulation studies and technologies like CloudSim. These technologies have made it possible to model these environments to optimize various performance, reliability, and control techniques. To create our power-aware data center model, CloudSim is used. Cloud computing focuses on the development of grid computing, virtualization, and internet generation. Cloud Computing has many problems and challenges from diverse issues in the cloud-making plans, it plays a completely vital role in identifying effective implementation. Scheduling refers to fixed policies for controlling the order of tasks to be finished via a computer machine.

There are special kinds of scheduling algorithms in an allocated computing device, and scheduling tasks is one of them. the number one benefit of scheduling is the achievement of immoderate computing energy. Scheduling controls CPU memory availability, and an incredible scheduling coverage presents the most beneficial and useful resource usage. The resource consumption of most PCs is drastically underutilized because modern computer hardware was only intended to operate a single operating system and one application. Virtualization technology is employed to improve resource consumption. You can operate numerous virtual machines on a single physical system using virtualization,

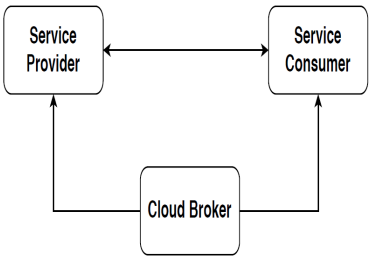

Fig. 1. Cloud Stakeholders

The organization of the paper is as follows: section II discusses related work. Section III presents a scheduling problem. Thereafter, some existing scheduling techniques are explained in section IV. Section V discusses the results. In the last section, conclusion and future direction of a research are included.

II. Related Work

The virtual machine (VM) placement is a very interesting problem of cloud computing where the mapping between available VM and user request is carried out. In the literature, many approaches are presented recently [13], [22]. Next interesting topic of cloud computing is task scheduling. The related work based on task scheduling is carried out in this section. In paper [5], the authors have tried to explain and give an overview of the cloudsim toolkit. An algorithm for load balancing was proposed by the author in the paper [11]. This work discussed load-balanced two-level task scheduling. The authors have presented an efficient job scheduling

system based upon genetic simulated Annealing process [8]. Using Cloudsim and fundamental Operating System scheduling algorithms such as FCFS, SJF, and priority algorithm, the paper [7] analyses and rates the effectiveness of various task scheduling techniques in a cloud-simulated setting, firstly testing is done which scheduling policy performs best under various conditions.

A max-min task scheduler was suggested in the paper [17] as a way to increase resource usage and response time. The max-min approach allocates the largest task to the VM with the shortest execution time to reduce makespan, but max-min only functions well when there are a lot of large-size tasks. An enhanced max-min was proposed in the paper [10]. By replacing the completion time selection criteria with the execution time, the author claims that the execution and waiting times can be decreased even more. In [4], an attempt was made to enhance the min-min for resource utilization and makespan. In the suggested technique, the balancer actions start once the min-min has finished. In order to decrease Makespan and improve load balancing across resources, a task scheduling approach combining max-min, min-min, and genetic algorithms was proposed [2].

In [6] and [23] the genetic algorithm has been successfully and frequently used to solve scheduling issues. Various modifications are made to Ant colony optimization(ACO) which are proposed [14] and [16]. ACO algorithm was utilized by the author in [14] to address the load balancing issue. The goal of the algorithm was to keep the load of the system balanced and makespan minimum.

Kennedy and Eberhart have created the optimization method known as Particle Swarm Optimization (PSO) [12]. The PSO method is comparable to other population-based algorithms although PSO contains simple mathematical operators and converges more quickly than other optimization algorithms like the firefly algorithm, cuckoo algorithm, etc. When compared to another technique known as Best Resource Selection-based mapping, the PSO-based mapping algorithm is significantly less expensive. Due to the PSO algorithm's lower compute and transmission costs, it is employed for workflow applications. To reduce the cost of workflow execution while still maintaining deadline constraints, Rodriguez and Buyya [20] have used the PSO method, in which the impact of VM performance variance is taken into account. It creates a scheduling solution by mapping jobs onto VM instances using a particle. For the cloud workflow scheduling with deadline restriction, a GA-based strategy was proposed in [15], where an adaptive penalty function is utilized.

III. Scheduling Problem Formulation

The initial assumption is made that each task submitted by user is independent of the other. In this paper, there are n tasks in consideration and these tasks are processed across m computational resources, the main goal or objective is to minimize the makespan so that the completion time of all tasks can be minimized. If there are more tasks compared to resources, then a proper task scheduling algorithm is needed to distribute tasks based on the scheduling technique. In this paper it is considered that the number of tasks is more than resources, hence one task cannot be allocated to more than one resource or that task cannot be migrated to a different resource.

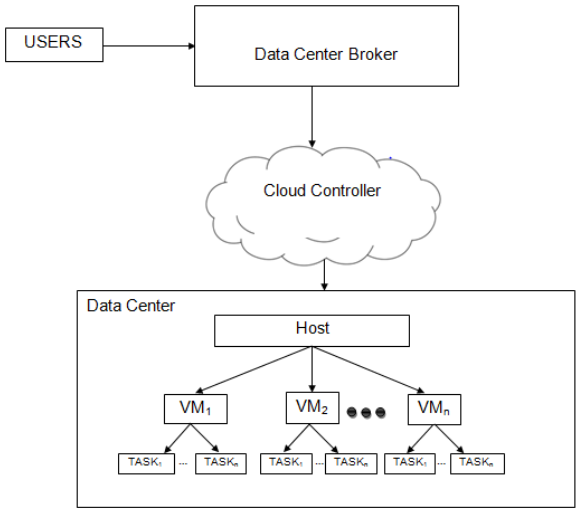

Fig. 2. Structure of Task scheduling

Following are some assumptions on basis of which the task scheduling model is formulated:

- All the tasks to be scheduled should be available before the scheduling i.e. at time zero.
- All the tasks should be executed without preemption.
- Tasks should be executed on a single VM at one time.

The set of tasks is specified as T_i = {1, 2, 3, 4, 5, ..., n} where n tasks are independent, and R_j = {1, 2, 3, 4,..., m} which is a collection of computational resources. The number of tasks should be greater than the number of resources i.e n $>$ m. Such circumstances give rise to m^n ways for allocating these tasks among the available machines. In order to ensure the best possible usage of the computing resources, our goal is to schedule submitted tasks on the available Virtual Machines(VMs). Here, Table 1 shows an example of mapping submitted tasks to the VMs that are available for their execution. We assume that the task i's computational node j's execution time is known and equal to job RunTime$_{ij}$. A matrix is needed to be calculated such that if task i is executed on resource j then element Xij in the matrix is 1 otherwise 0 and it minimizes the overall cost of task execution on computing resources.

TABLE I
Mapping of Tasks to Virtual Machines

Tasks	T_1	T_2	T_3	...	T_n
Virtual Machine	M_2	M_3	M_1	...	M_m

Makespan is the amount of time it takes for the final task to complete its execution on a processing resource or the longest amount of time needed by the tasks in a given schedule to be completed. A matrix is created of order m$\times$ n which is called Expected Time to Complete(ETC) using Eq. (1) for each job in order to calculate the same.

$$ETC_{i,j} = T_i/M_j \quad (1)$$

where, T_i/M_j denotes the completion time of ith task on jth machine. For all tasks, i = {1,2,3,....,n} and j = {1,2,3,....,n}

$$makespan = max \sum_{j=1}^{m} FT_j \quad (2)$$

where
m : Number of virtual machines.
n : Number of tasks.
FT_j : Finishing time of M_j

IV. Algorithms for Task Scheduling

A. First Come First Serve

The resource with the shortest waiting time in the queue is the one targeted by FCFS for parallel processing. FCFS scheduling algorithm is offered by cloudsim simulator for internal task scheduling. The virtual machine-provided component is in charge of allocating application-specific virtual machines to cloud-based data centers. A provisioned virtual machine's default policy simply allocates a virtual machine to the host based on FCFS policy. FCFS has the drawback of being not preemptive [19]. The drawback of FCFS is that the jobs at the end of the queue have to wait for their execution.

B. Round Robin Scheduling

Fairness is the main goal of the Round Robin (RR) algorithm. Tasks are kept in RR's queue, which is a ring. Every task in the queue will run alternatively with the same execution time [1]. The task is returned to the queue where it is waiting for another turn if it cannot be finished in the allotted time. The benefit of the RR algorithm is that tasks are completed consecutively without having to wait for the completion of the preceding one. However, RR will take a long time to complete all tasks if the load is found to be heavy. The RR scheduling technique for internal purposes is supported by the Cloud Sim Toolkit.

C. Generalized Priority Algorithm

You must specify a cloudlet attribute such as size, memory, bandwidth planning policies, etc. The priority is determined according to the request of the user. The tasks are originally ranked in the suggested method according to their size, with the work with the largest size receiving the top position [1]. Additionally, priority is also given to Virtual machines based on their million instructions per second(MIPS) value. The machine with the highest MIPS is given the highest priority. Because of this, a crucial consideration in task prioritization is task size, or in the case of virtual machines, MIPS. Compared to Round Robin scheduling and FCFS, this strategy is more effective. Take into account five distinct computations of VMs, denoted by their ID and MIPS values as V = ((0, 350), (1, 1500), (2, 350), (3, 900), and (4, 350)). Here, VM1 gets the highest priority because of its highest MIPS, followed by VM3, which receives the second priority, and then VM0, VM2, and VM4 get the remaining priorities.

Steps for Algorithm

- Step-1: Create virtual machines (VMs) in various data centers based on the host/physical server's processing capacity in terms of CPU cost, memory, and storage.
- Step-2: Distribute cloudlet length based on computational capacity.
- Step-3: The VM Load Balancer tool keeps track of the VM's index table; at the moment, there is no allocation.
- Step-4: Cloudlet constrained by respective MIPS and length.
- Step-5: The highest cloudlet length will get the highest virtual machine MIPS.
- Step-6: Datacenter broker makes a request to the VM indicated by its id.
- Step-7: Update available resources.

D. Min-Min Algorithm

It is a heuristic approach [18] that begins with the set of all unmapped tasks and finds the shortest estimated duration for each task in the meta-task. The task is chosen and allocated to the appropriate source with a minimum predicted completion time. Until the meta-task is not empty, this procedure is repeated. You must finish minor tasks before moving on to a larger one [9].

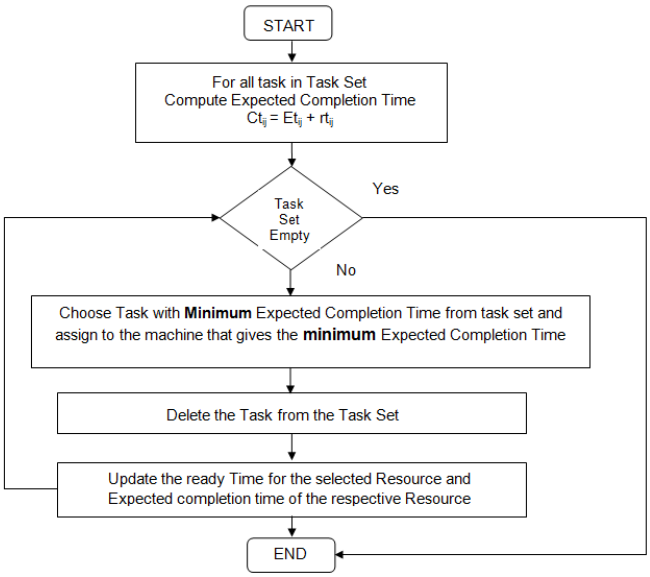

Fig. 3. Flow Chart of Min-Min Algorithm [9]

E. Genetic Algorithm

The 1970s' early years saw the greatest rise in the popularity of Genetic Algorithms (GA). Figure 4 illustrates how the genetic algorithm functions. It has also been widely recognized as one of the top optimization methods that may be used in a variety of study domains. In addition to this, several techniques and methods have been presented to address the issues associated with task scheduling [3].
Recent research has concentrated on decreasing task execution durations, dealing with bandwidth problems, lowering task execution costs, and so forth.

Phases of genetic algorithm
1) Encoding Chromosomes
2) Initial Population Generation

3) Fitness Function
The Genetic Algorithm working is depicted in Figure 4.

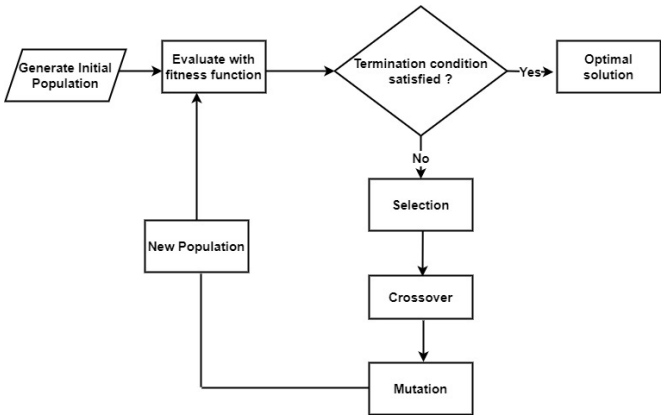

Fig. 4. Work Flow of Genetic Algorithm [3]

F. Particle Swarm Optimization

Kennedy and Eberhart have suggested the particle swarm optimization (PSO) meta-heuristic approach to address optimization issues. PSO is an optimization technique that relies on the social behavior of fish schooling and flocks of birds [21]. Similar to fish or birds in nature, PSO particles also have a position and velocity vector. The concept of population-based search is used in optimization techniques such as particle swarm optimization. The solution in this case is the particle's position, and the swarm of particles serves as the searching agent. Along with its own experience, the PSO particle also gains knowledge from other particles (social learning and cognitive learning). There are thus two options: gbest and pbest. Since there are changes in a location with regard to time (or iteration) and random weighted acceleration is used, velocity V and acceleration also play an important role in this situation.
A population of particles is taken into account as the initial space in this technique. Each member of the population, which corresponds to an individual in the evolutionary algorithms, represents a possible solution. A flock of particles is first created at random, and each one's position in the issue space indicates a viable solution. There is a velocity vector that is analogous to the position vector and is defined for each particle. Like the position vector, it is updated consistently and causes movement in the search space.
The formula for updating the position vector is based on the approach described by Kennedy and Eberhart below :

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{3}$$

and velocity vector is :

$$V_i(t+1) = \omega * V_i(t) + C_1 r_1 (pb_i - X_i(t)) + C_2 r_2 (gb_i - X_i(t)) \tag{4}$$

This algorithm's fitness function is applied to each particle's position vector during each generation, and the goodness of each particle is assessed. The pb and gb are two parameters that are specified in (4) and play critical roles in PSO. pb denotes the best position that a particle has experienced since the procedure began, and gb for the best position that has existed between particles since the algorithm's start.
The inertia weight is w. The right choices for inertia weight and acceleration coefficient should be made if we wish to achieve a balance between local and global search. After a particle advances toward its new position using the updated velocity vector, the velocity equation updates the particle's current velocity. Each cycle evaluates each particle using the previously determined proportion function. This procedure is repeated until the algorithm's termination requirement is met. The Flowchart of PSO is shown in Fig 5.

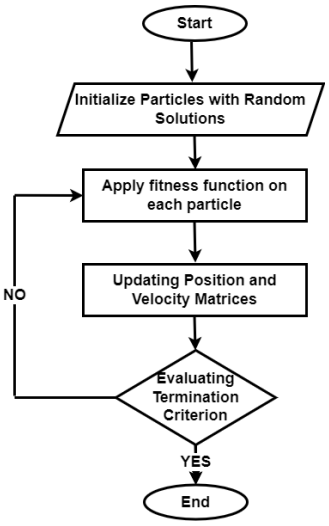

Fig. 5. Flowchart of PSO

G. Improved Particle Swarm Optimization

The initial population is created randomly in the traditional PSO algorithm, but randomness reduces the likelihood that the algorithm will converge to the best solution. Traditional task scheduling algorithms can be merged with standard Particle Swarm Optimization to make improved PSO to reduce the makespan of the task scheduling algorithm. To improve the behavior of the PSO algorithm, we merge the Priority algorithm into PSO, i.e., we generate the initial population while taking into account the Priority algorithm. In the initial population, the highest cloudlet length is given to the Virtual Machine with the highest MIPS. The flowchart for this modified PSO method is shown in Fig 6. All other phases are identical to those in the normal PSO algorithm.

V. Results and Discussions

A. Experimental Setup using CloudSim Simulator

One of the following methods can be used to evaluate the algorithm and protocols' performance:
• By using publicly accessible Cloud Management Platforms to create and manage clouds.
• By utilizing a simulator like TeachCloud, CloudSim, etc.

CloudSim makes it simple to model, simulate, and test out new application services. Event-based simulation is offered by CloudSim, in which various system entities communicate by sending events. Its extremely easy to compile the CloudSim example codes; a command prompt

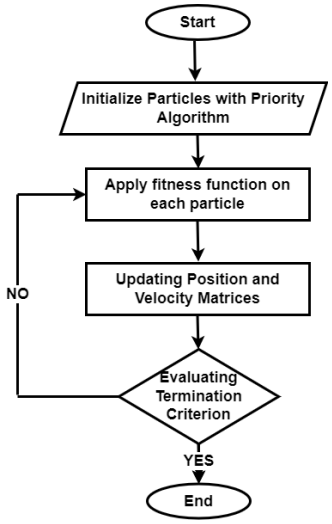

Fig. 6. Flowchart of Improved PSO

is used. For even easier work, CloudSim can be added to Eclipse.

Assumptions and Performance Metrics

• Tasks are not preemptive, meaning that no new task can take priority over an existing one.

• Tasks are independent of one another; that is, there is no order in which these would be completed.

• Tasks are computationally demanding.

TABLE II
VIRTUAL MACHINE PARAMETERS

Image Size	10000 MB
RAM	512 MB
MIPS	1000
Bandwidth	1000 Megabits/s
Processors in VMs	1
Count of VMs	50 VMs
Virtual Machine Monitor	"Xen"

Several performance measures were utilized to assess and compare the algorithms' performance. One of the key metrics was waiting time, which is the amount of time tasks wait to acquire a VM. The execution time, or the amount of time a process uses a virtual machine, was another statistic. The scheduling algorithm attempts to average waiting and execution times for jobs, as well as minimize the makespan.

B. Results Analysis

In this section comparison between different algorithms is shown. The Main metrics considered are makespan and execution time. Tabulation of results obtained by FCFS, SJF, Round Robin, the Generalized Priority algorithm and PSO are shown. Table 2 and Table 3 contain a tabulation of the findings from the FCFS, SJF, RR, General Priority algorithm, and PSO algorithm. Table 2 shows Comparisons of different task scheduling algorithms. It can be clearly seen that the makespan of the PSO algorithm is least as compared to FCFS, SJF, RR, and Priority algorithms. Table 3 depicts a comparison in the makespan time of various task scheduling algorithms when the number of tasks is varied. Figures 7 and 8 show the comparisons in the graphical format of different task scheduling algorithms where PSO has the least makespan.

TABLE III
COMPARISON OF DIFFERENT TASK SCHEDULING ALGORITHMS

Algorithm	Simulator	Total completion time(ms)	Average completion time(ms)	Makespan(ms)
FCFS	Cloudsim	699922.64	3499.61	9237.59
SJF	Cloudsim	647210.91	3236.05	7536.21
Round Robin	Cloudsim	686135.66	3430.67	10725.75
Generalized Priority	Cloudsim	59190.68	2959.53	7127.94
PSO	Cloudsim	41129.58	2056.47	4769.51

TABLE IV
COMPARISON OF MAKESPAN FOR VARIOUS TASK SCHEDULING ALGORITHMS(MS)

Number of Tasks	FCFS	SJF	Round Robin	Priority Algo	PSO
30	5361.89	4220.27	4158.49	3906.37	2830.45
200	9726.26	9296.50	9908.94	8821.38	4247.67
400	13180.16	12895.98	13126.24	11905.59	5726.79

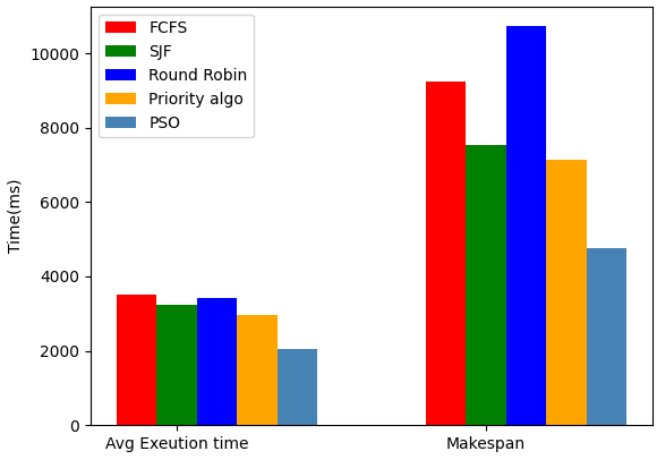

Fig. 7. Comparison of Different Task Scheduling Algorithms

VI. CONCLUSION

A cloud computing system's performance is always greatly influenced by effective scheduling algorithms. An effective scheduling algorithm must include customer requirements that fulfill their SLA-mandated obligations while also being advantageous to the cloud provider. It can be seen that the PSO algorithm outperformed FCFS, SJF, RR, and the General priority algorithm. The performance of cloud services may be enhanced by combining several different elements to create an effective scheduling algorithm. The benefit of task scheduling is the effective utilization of all resources, the achievement of user deadlines, and improved multiprogramming choices.

Traditional scheduling techniques result in slow throughput and lengthy response times. Makespan can be reduced in the future by using optimization techniques such as Ant

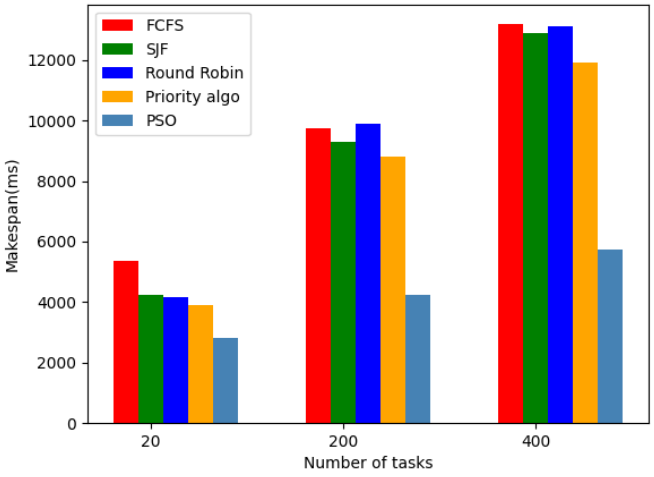

Fig. 8. Comparison of Makespan for Various Task Scheduling Algorithms(ms) at Different Task Load

Colony Optimization(ACO) and Particle Swarm Optimization(PSO) which will lead to more resource utilization. In future, a hybrid approach for task scheduling will be attempted by combining the PSO with the genetic algorithm to further improve the reduction in makespan.

REFERENCES

[1] Amit Agarwal and Saloni Jain. Efficient optimal algorithm of task scheduling in cloud computing environment. *arXiv preprint arXiv:1404.2076*, 2014.

[2] Ismael Salih Aref, Juliet Kadum, and Amaal Kadum. Optimization of max-min and min-min task scheduling algorithms using ga in cloud computing. In *2022 5th International Conference on Engineering Technology and its Applications (IICETA)*, pages 238–242. IEEE, 2022.

[3] Sarah B Basahel and Mohammad Yamin. A novel genetic algorithm for efficient task scheduling in cloud environment. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 30–34. IEEE, 2022.

[4] Kadda Beghdad Bey, Farid Benhammadi, and Rédha Benaissa. Balancing heuristic for independent task scheduling in cloud computing. In *2015 12th International Symposium on Programming and Systems (ISPS)*, pages 1–6. IEEE, 2015.

[5] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *2009 international conference on high performance computing & simulation*, pages 1–11. IEEE, 2009.

[6] Ying Changtian and Yu Jiong. Energy-aware genetic algorithms for task scheduling in cloud computing. In *2012 Seventh ChinaGrid Annual Conference*, pages 43–48. IEEE, 2012.

[7] Monica Gahlawat and Priyanka Sharma. Analysis and performance assessment of cpu scheduling algorithms in cloud using cloud sim. *Int. J. Appl. Inf. Syst*, 5(9):5–8, 2013.

[8] Guo-ning Gan, Ting-lei Huang, and Shuai Gao. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In *2010 International Conference on Intelligent Computing and Integrated Systems*, pages 60–63. IEEE, 2010.

[9] Mubarak Haladu and Joshua Samual. Optimizing task scheduling and resource allocation in cloud data center, using enhanced min-min algorithm. *IOSR J. Comput. Eng.(IOSR-JCE)*, 18(4):18–25, 2016.

[10] XiaoShan He, XianHe Sun, and Gregor Von Laszewski. Qos guided min-min heuristic for grid task scheduling. *Journal of computer science and technology*, 18(4):442–451, 2003.

[11] R Jayarani, Sudha Sadhasivam, N Nagaveni, et al. Design and implementation of an efficient two-level scheduler for cloud computing environment. In *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 884–886. IEEE, 2009.

[12] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[13] Sudhanshu Kulshrestha and Sanjeev Patel. An efficient host overload detection algorithm for cloud data center based on exponential weighted moving average. *Int J Commun Syst.*, 34(4):1–30, 2020.

[14] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and Dan Wang. Cloud task scheduling based on load balancing ant colony optimization. In *2011 sixth annual ChinaGrid conference*, pages 3–9. IEEE, 2011.

[15] Li Liu, Miao Zhang, Rajkumar Buyya, and Qi Fan. Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing. *Concurrency and Computation: Practice and Experience*, 29(5):e3942, 2017.

[16] Yong Liu, XH Wang, CM Xing, and Shuo Wang. Resources scheduling strategy based on ant colony optimization algorithms in cloud computing. *Computer Technology and Development*, 21(9):19–23, 2011.

[17] Yingchi Mao, Xi Chen, and Xiaofang Li. Max–min task scheduling algorithm for load balance in cloud computing. In *Proceedings of International Conference on Computer Science and Information Technology*, pages 457–465. Springer, 2014.

[18] Shubham Mittal and Avita Katal. An optimized task scheduling algorithm in cloud computing. In *2016 IEEE 6th international conference on advanced computing (IACC)*, pages 197–202. IEEE, 2016.

[19] Abdul Razaque, Nikhileshwara Reddy Vennapusa, Nisargkumar Soni, Guna Sree Janapati, et al. Task scheduling in cloud computing. In *2016 IEEE long island systems, applications and technology conference (LISAT)*, pages 1–5. IEEE, 2016.

[20] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE transactions on cloud computing*, 2(2):222–235, 2014.

[21] Yuhui Shi, Russell Eberhart, and Yaobin Chen. Implementation of evolutionary fuzzy systems. *IEEE Transactions on fuzzy systems*, 7(2):109–119, 1999.

[22] Shilpa Sunil and Sanjeev Patel. Energy-efficient virtual machine placement algorithm based on power usage. *Computing*, pages 1–25, 2023.

[23] Chenhong Zhao, Shanshan Zhang, Qingfeng Liu, Jian Xie, and Jicheng Hu. Independent tasks scheduling based on genetic algorithm in cloud computing. In *2009 5th international conference on wireless communications, networking and mobile computing*, pages 1–4. IEEE, 2009.