Job scheduling reservations on cloud resources

Ardi Pujiyanta [a,1,*], Fiftin Noviyanto [a,2]

[a] Department of Informatics, Universitas Ahmad Dahlan, Kec. Banguntapan, Kab. Bantul, Daerah Istimewa Yogyakarta 55191, Indonesia
[1] ardipujiyanta@tif.uad.ac.id; [2] fiftin.noviyanto@tif.uad.ac.id
* corresponding author

ARTICLE INFO

Article history
Received November 20, 2023
Revised January 17, 2024
Accepted March 11, 2024
Available online August 31, 2024

Keywords
Cloud computing
Idle time
Virtual view
Waiting time
Makespan

ABSTRACT

The current application of cloud computing focuses more on research problems. One of the main problems in the cloud is job allocation. Jobs are dynamically allocated to server processors. All cloud-virtualized hardware is available to users on demand and can be dynamically upgraded. Resource scheduling is critical in cloud research due to its large execution time and resource costs. The differences in resource scheduling criteria and parameters used cause various categories of Resource Scheduling Algorithms. Resource scheduling has a goal: identifying the right resources to schedule workloads on time, improving resource utilization effectiveness, and, in other words, minimizing workload completion time. Mapping the right workloads to resources will result in good scheduling. Another goal of resource scheduling is to identify adequate and appropriate workloads. So, it can support scheduling multiple workloads and meet various QoS needs in cloud computing. The aim of this research is to determine the value of waiting time, idle time, and makespan in cloud resources. The proposed method is to sort the arrival times of jobs with the least workload and place them in a virtual view before scheduling them on cloud resources. Experimental results show that the proposed method has an idle time of 25.3% and FCFS is 43.1%, while for backfilling, it is 31.5%. The average makespan reduction for FCFS is 16.73%, and for backfilling, it is 12.87%. The average decrease in AWT for FCFS was 13.3%, and for backfilling, it was 12.03%. The results of this research can be applied to cloud rentals with flexible times.

1. Introduction

Cloud computing uses virtualization technology to offer services. The services provided can be in the form of storage computing via the internet network. Task allocation is one of the main problems in the cloud [1]–[4]. Task allocation can be done dynamically on the server processor. An unlimited collection of resources in the cloud is used for various computing needs [5]. Distributed platforms are utilized efficiently to get the best resource management services from cloud systems [6]. There are many techniques for effective resource management in the cloud, such as cloud job scheduling, resource migration, etc. Choosing the right technique will help save costs and good response time. So it benefits cloud users [7]–[9]. The cloud provides virtualized computing hardware similar to a public utility, so it is called Infrastructure as a Service (IaaS). Services are available to users on demand and can be improved dynamically. The cloud computing service model refers to applications and software platforms, hence the name Software-as-a-Service (SaaS) [10].

Scheduling is the distribution of certain work on resources to be completed efficiently. The main objectives are (i) reducing deadlines and maximizing resource utilization, (ii) optimizing the server in executing tasks, and (iii) working on higher-priority jobs first and reducing completion time. Another advantage of scheduling is that it increases system throughput and improves performance [11]. The required level of service quality can be met by the minimum number of resources used and the workload can be maintained or by minimizing the completion time of the workload (maximizing throughput) [12]. Mapping workloads to resources is necessary for scheduling [13]. So, identifying sufficient workloads will support the scheduling of several workloads. QoS requirements, such as CPU utilization, availability, reliability, security, etc., will be met [14], [15].

In backfilling scheduling, two things are generally measured: the accuracy of predictions and the measurement of scheduling performance [16]. In dynamic cloud scheduling, a backfilling algorithm is used to divide tasks into two queues [17]. The proposed method is the Simple Backfilling Algorithm (SBA) and DCBA in cloud computing. Both algorithms provide good performance for balanced or moderate workloads and also provide better performance when the workload becomes heavier. This method can also be implemented for all cloud tasks in future work [18]–[20]. The applied technique combines FCFS with a backfilling algorithm. It works by scanning the queue in real time. The proposed algorithm allows jobs at the back of the queue to be processed without delaying the head of the queue. Further experimental results show that the number of initial reservations accepted by a cluster must be below a threshold to maintain cluster performance [21].

In his research [22], the M/G/1 queuing system was used. Strategic customers must decide whether to reserve a server first (and thus receive higher priority) or ignore the reservation. Server reservations in advance are subject to a fee. This study characterizes customer behavior strategies, equilibrium outcomes, and revenue maximization policies. Customers will be charged according to the amount of resources used. The main problem CPs face is choosing the right PM so that the new VM host still meets end user requirements. The distribution characteristics and scalability of cloud resources are taken into consideration [23]. In this paper [24], Static Independent Task Scheduling on Virtual Servers is proposed. Tasks are allocated to VMs by measuring the availability of each resource. Processing power, cost, and amount of processing are used in grouping tasks.

The literature review shows that the proposed architecture and scheduling algorithm will be influenced by factors such as idle time, waiting time, resource availability, and time horizon. So it becomes a challenge and limitation for existing resource scheduling algorithms. The aim of the research that will be carried out is to try to overcome the things above. The factors mentioned above focus on mapping jobs to virtual machines for an optimal schedule. This research proposes an FCFS slot-free method used to identify idle resources by utilizing user-submitted parameters to reduce resource execution delays, increase makespan values, and reduce job waiting times.

2. Method

2.1. System Architecture for Cloud Computing

Fig. 1 shows our cloud service system, where the number of virtual machines (VMs) is equal to the number of machines in the logical view. Virtual machines (VMs) are a subset of cloud resources that can be allocated to cloud services. The proposed system consists of cloud system information (CSI), Logical view(LV), Local scheduler (LS).

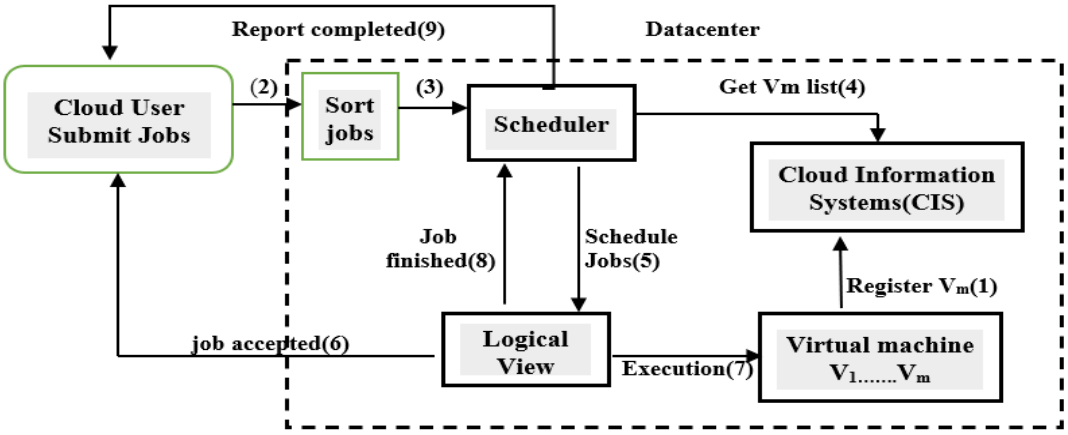

Fig. 1. Proposed job allocation flow in the cloud

Fig. 1 can be explained as follows: the number and status of Virtual Machines in a logical view are registered in the CIS, with the initial status of the Virtual Machine being free (List resource (1)). The user submits a reservation (2), then the work enters the task pool to be accommodated and sorted based on priority and then submitted to the scheduler (3). In the next step, the scheduler will check the Virtual Machine status on CIS (4), and whether a Virtual Machine status can be used. If the Virtual Machine status is free, then schedule job (5) from the logical view. All jobs that have been scheduled in the logical view will be sent to the user that the job is accepted (job accepted) (6) and executed (7) at a certain timeslot and a certain Virtual Machine number in the physical view. Virtual Machines that have finished executing in the logical view are also finished executing on the physical view. The status of the Virtual Machine on the CIS is changed by the logical view to free, and the scheduler gets a notification from the logical view that a job has finished executing (8). The scheduler then informs the user that the job has finished executing (9).

The function of each component in Fig. 1 above is as follows: 1) Cloud Information Systems(CIS): Stores a list of existing virtual machine resource information; 2) Logical view (LV): This component functions to place a list of jobs that will be executed on the virtual machine; 3) Local scheduler (LS): This component functions to schedule incoming jobs that will be executed on the virtual machine.

2.2. Proposed Algorithm

In this section, we present a new scheduling algorithm that maximizes resource utilization, minimum makespan, and minimizes delay time in the cloud.

Step 1: Create $V_M = V_{M1}, V_{M2}, V_{M3}, \dots, V_J$ into a set of resources. Step 2: Register the number of virtual machines on the Cloud Information System (CIS). Step 3: Sort jobs $B = B_1, B_2, B_3, \dots, B_i$ in ascending order. Step 4: Read the list of available virtual machines $V_M = V_{M1}, V_{M2}, V_{M3}, \dots, V_J$. Step 5: Schedule ordered jobs $B = B_1, B_2, B_3, \dots, B_i$, in the logical view (the number of machines), in the logical view is equal to the number of virtual machines created). Step 6: Inform the user that the job was accepted and will be executed. Step 7: Schedule and execute job $B = B_1, B_2, B_3, \dots, B_i$ on the available virtual machines. Step 8: Mark or delete jobs that have finished executing on the virtual machine and tell the scheduler that the jobs have finished executing. Step 9: Inform the user that the work has been completed

Notation Explanation, time of the earliest start time of the job (twet): the fastest execution time of a job. Start time to execute the job (tmd): the time a job starts to be executed. Completion time to execute the job (tasa): earliest execution time until the end of job execution. The end time to execute

the job (tpa): the latest execution start time. Execution time of the job (te): execution time. Relaxed time (ts): the difference between the actual execution start time and the earliest execution start time.

Algorithm: Job scheduling algorithm

Input: Job (jobId, $t_{asa}, t_{pa}, t_e, numJob$)

Output: RIT, AWT, Makespan

Begin()

// Declaration and Initialization; Virtual logical (V_L)

update(Table(Vm))

CIS←Register Vm

B_i← Sort jobs

Read Vm

// Schedule ordered jobs B_i

Free←($t_{pa} - t_{asa}$)

Note ← false;

If (!Note) then

start ← t_{wet};

finish ← t_{wet} + t_e ;

flex ← start - t_{wet};

while(!Note AND ($t_{pa} - t_{asa}$)<=Free)

min ← minR(start, finish);

If (min > 0) then

allocV_L (Id, jobId, t_{wet}, start, t_{pa}, t_e);

suk ← true;

else

start ← time + 1;

finish ← start + t_e - 1;

flex = start - t_{wet};

End

End while

End

RIT ←$Finish_{previous} - start_{current}$ // calculate RIT

TotalRIT← $\sum_{i-0}^{Size}$ RIT //calculate the total RIT

Makespan ← $max_{i \in job\ i} F_i$

WT ← $Start_{resr}$-$Start_{new}$

//job accepten V_M ← V_L// execute job V_L on the virtual machine

Lines 2 to 9 describe the declaration and initialization of the values used in the algorithm. Lines 10 to 13, calculation of the value of the job sent by the user. In lines 14 to 19, the loop is used to search for unused space or empty space in the virtual view. If there is free or unused space, the job will be allocated. In lines 21 to 25, the idle time, waiting time, and makespan values will be calculated. Line 26 of the job, will be executed on the virtual machine according to the existing virtual view.

2.3. Illustration of Method

Table 1. Shows the jobs sent by the user, with the job attribute tweet being the execution start time, ts being the flexible time, te being the time required to execute, and CN being the number of resolutions required.

Table 1. Jobs submitted by users

Submit_Job	t_{wet}	t_s	t_e	CN
1	0	0	2	2
2	0	0	3	2
3	0	3	4	1
4	0	0	4	1
5	1	5	5	1
6	1	6	3	2
7	1	6	3	1
8	2	9	5	1
9	2	9	3	1
10	3	7	3	1
11	3	8	4	2

All jobs sent by users will find a place on a logical node; if there is an empty place, then the job will be placed on a logical node; if there is no empty place, the job will be shifted according to the flexible time required. For example, for job number 3, execution time starts from t=0 to t=3. If there is an empty space in the slot range, the job will be placed in the logical view. If there are no empty places then the job will be rejected. Fig. 2 shows that job number 3 is placed in slot number 2, meaning that the job is shifted to the limit of slot number 2 for initial execution. The user will be notified that his job was accepted and will be executed.

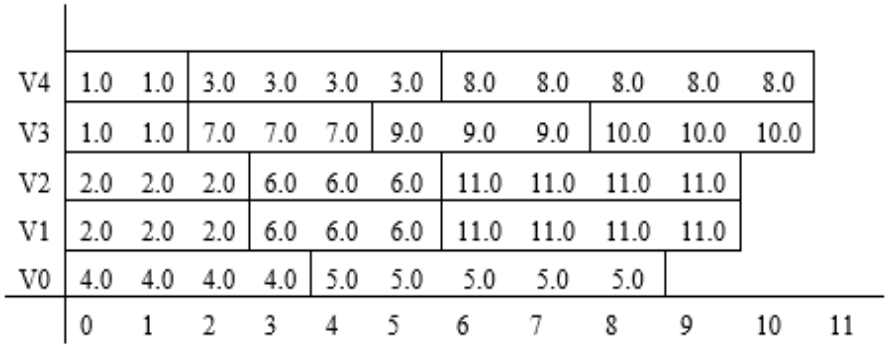

Fig. 2. Job placement in the logical view

2.4. Performance Metrics

Scheduling is a list of tasks that determines how competing tasks access one or more reusable resources. These resources can be hardware, such as processors, communications lines, storage devices,

or software. Task scheduling is assigning tasks to specific resources by starting and ending task times with certain limits. Task scheduling is an integrated part of cloud computing. The purpose of task scheduling is to allocate resources for task implementation. Task scheduling guides resource allocation because there are many nodes on which tasks can run. The problem is how to assign tasks to those resources. This assignment is known as task allocation to the resources scheduled by the scheduler. Performance metrics are used to measure certain attributes in a proposed or used scheduling algorithm.

2.4.1. Resource Idle Time (RIT)

A resource may not be usable even if a reservation request is available [25]. Delay times occur because scheduling policies do not match the allocation of reservation requests. RIT is calculated by applying the formula below;

$$RIT = Finish_{previous} - Start_{current} \quad (1)$$

When there is a reservation request with a conflict, the following equation calculates the total resource idle time.

$$Total\ RIT = \sum_{i-0}^{Size} RIT \quad (2)$$

2.4.2. Makespan

Makespan: Last job completion time. Users want to shorten their application completion time [26]–[28].

$$Makespan = max_{i \in job\ i} F_i \quad (3)$$

Where F_i indicates the completion time of job i

2.4.3. Waiting Time

Sometimes, a resource is unavailable when a reservation requires it, but the resource can be booked at a different time [29], [30]. The difference between the expected and actual start times is the waiting time, as shown in Equation 3.

$$Waiting\ Time\ (WT) = Start_{reser} - Start_{new} \quad (4)$$

Total Waiting Time (TWT) is the total waiting time in a timeslot, shown by equation 4

$$Waiting\ Time\ (WT) = Start_{reser} - Start_{new} \quad (5)$$

The size value refers to the reservation length of a particular timeslot. So the Average Waiting Time is shown by equation 5.

$$Average\ Waiting\ Time\ (AWT) = \frac{TWT}{No\ reservasi} \quad (6)$$

2.5. Workload

Configure the entities used in the simulation environment. Randomly generated workloads with different job sizes from 100 to 800. The number of virtual machines used is 30. The number of data centers is 1, with the number of hosts being 30. The scheduler is space shared, which only allows one job to run at a certain time within the resource certain. Sets of executed jobs are independent of each other.

3. Results and Discussion

The device uses Java Developer, Windows 11 operating system, 11th Gen Intel(R) Core(TM) CPU i3-1115G4 @ 3.00GHz 3.00 GHz. The backfilling approach is proposed as a comparison because this strategy shifts reservations early, making room for new reservations to be allocated. Viewed from the other side, the next job must wait in the waiting room queue until the previous job has finished executing, so there is no certainty about the time the job will be executed.

Therefore, resource usage may be inefficient, and jobs may have to wait for quite a long period of time. The leading job queue will wait if the required time is greater than the required computing resource time. Backfilling allows jobs that have execution times shorter than the execution times of jobs at the front of the queue to move forward and execute on idle computing resources. The delay time, waiting time, and job waiting time matrices are used as performance comparisons to make resource use more efficient.

Referring to table 1. So the job placement for the backfilling algorithm is shown in Fig. 3. Where job number 4 will be rejected or not executed, because the job must be executed right away, namely in slot number 0. Meanwhile, slot number 0 is already occupied by job number 3. Based on Fig. 2 and Fig. 3, it can be concluded that the proposed method has better job acceptance flexibility than the backfilling algorithm.

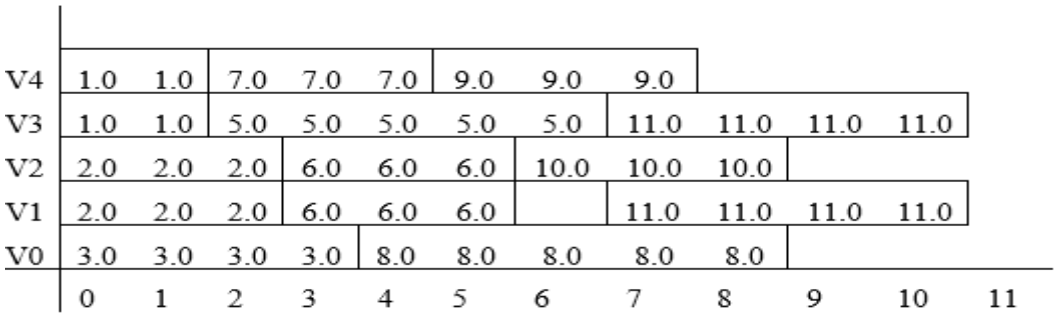

Fig. 3. Backfilling algorithm job placement

Experiments are carried out to represent a realistic cloud scheduling environment, considering different computing scenarios. The parameters to be observed are resource utilization and job waiting time. The parameters used in the experiment are shown in Table 2. The FCFS-Slotfree method will be compared with FCFS backfilling to measure delay time, job waiting time, and makespan.

Table 2. Job Experiment Parameters

Parameter name	Parameter value
Job execution time duration	Fixed
The number of resources the job requires	Fixed
Execution start time	Changed
Execution end time	Changed

Fig. 4 shows the idle time each algorithm generates for workloads of different sizes. This shows that the proposed algorithm shows significant improvement in idle time. If we look at the percentage, it can be seen that the proposed algorithm produces better idle time than other algorithms. We can observe that the average idle time for the proposed method is 25.3%, FCFS is 43.1%, and backfilling is 31.5%. In general, it can be observed that the proposed idle time is smaller than that of FCFS and backfilling.

Implementing advance reservation in the proposed scheduling system increases resource utilization by 17.8% for FCFS and 6.27% for backfilling. This is caused by fragmentation or idle time. The proposed strategy uses FCFS-Slotfree to schedule common job deadlines that can minimize the initial idle period, regardless of the size of the initial and final period of unemployment. The results show that FCFS-Slotfree provides the best system utilization compared to other strategies. FCFS-Slotfree provides a better allocation policy according to reservation requests.

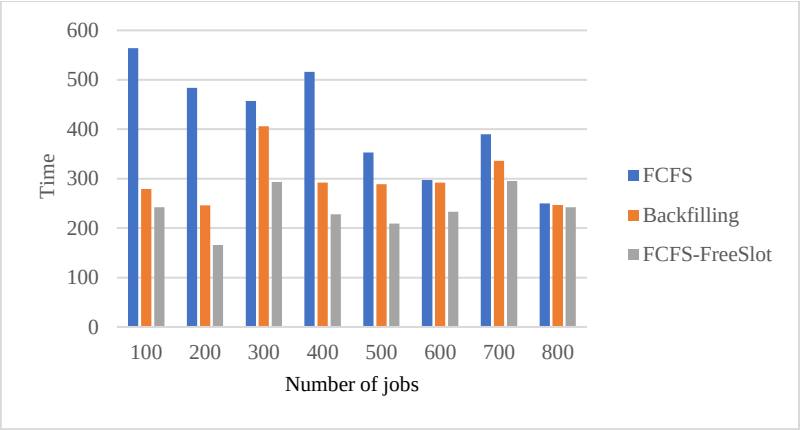

Fig. 4. Idle time results with different workloads

Fig. 5 shows the makespan produced by each algorithm for workloads of different sizes. This shows that the proposed algorithm can reduce the average makespan value significantly. If we look at the percentages, it can be seen that the proposed algorithm produces better makespan than other algorithms. We can observe that the average makespan reduction for FCFS is 16.73%, while for backfilling, it is 12.87%. This is because the execution delay time value of the proposed method is smaller compared to the FCFS and backfilling methods. The proposed method can place jobs early when they are about to be executed.

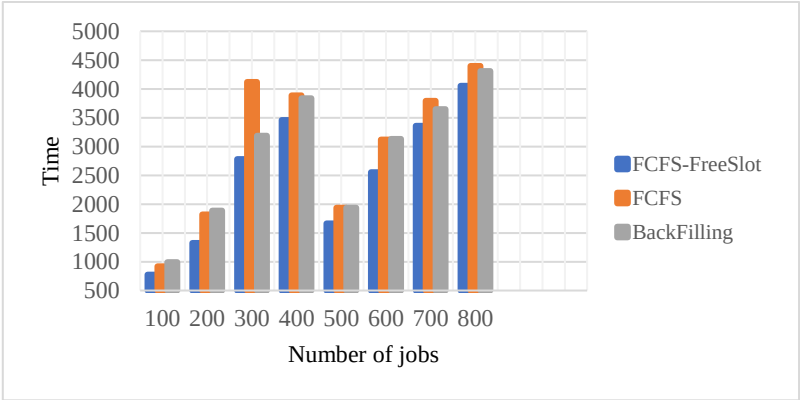

Fig. 5. Makespan results with different workload traces.

Fig. 6 shows the AWT generated by each algorithm for workloads of different sizes. This shows that the proposed algorithm shows significant improvement in AWT. If we look at the percentages, it can be seen that the proposed algorithm produces better AWT than other algorithms. We can observe that the average AWT reduction for FCFS is 13.3%, while for backfilling, it is 12.03%. In general, it can be observed that there is a significant improvement in AWT in the proposed algorithm. The leading job queue will wait if the required time exceeds the required compute node time. Backfilling allows jobs that have an execution time smaller than the execution time of jobs in the front queue to move forward

and execute on idle compute nodes. In the backfilling algorithm, the next job waits in the waiting room queue until the previous job has finished executing, so there is no certainty when the job will be executed.

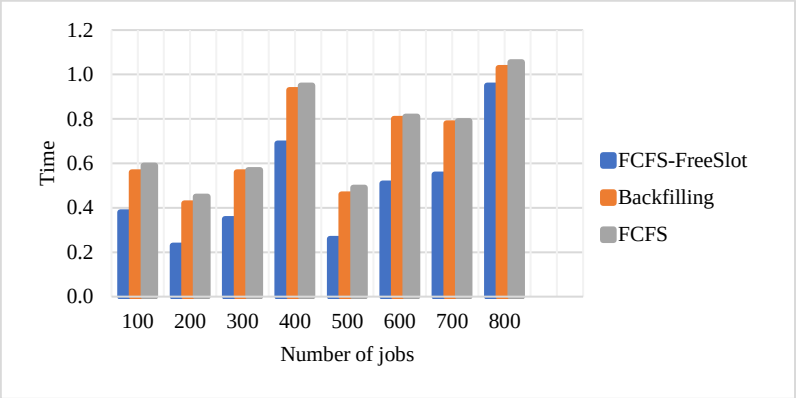

Fig. 6. Average waiting time results with different workloads

4. Conclusion

Job scheduling algorithms aim to provide better quality services such as delay time, wait time, cloud wait time, etc. Job scheduling simulation has been carried out using the proposed algorithm. Based on the simulation results, it is known that the proposed algorithm can work well. Compared with well-known algorithms such as FCFS and backfilling, the proposed algorithm has better performance. The algorithms were compared considering job sets of different sizes. After comparison, it can be seen that FCFS-Slotfree produces smaller delays, waiting times, and makespan values than FCFS and backfilling. The contribution of this research is that all work is completed in a shorter duration. This shows that in cloud computing, the proposed algorithm shows a better scheduling policy. The implication and potential future research direction is developing cloud scheduling with multiple sites and global scheduling in the cloud.

Declarations

Author contribution. All authors contributed equally as the main contributor to this paper. All authors read and approved the final paper

Funding statement. This research received a research grant from the Ministry of Research, Technology and Higher Education (Ristekdikti) of the Republic of Indonesia with contract number 001/PFR/LPPM UAD/VI/2023

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

References

[1] F. Alhaidari, T. Balharith, and E. AL-Yahyan, "Comparative Analysis for Task Scheduling Algorithms on Cloud Computing," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, Apr. 2019, pp. 1–6, doi: 10.1109/ICCISci.2019.8716470.

[2] M. Ibrahim *et al.*, "A Comparative Analysis of Task Scheduling Approaches in Cloud Computing," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, May 2020, pp. 681–684, doi: 10.1109/CCGrid49817.2020.00-23.

[3] P. M. Rekha and M. Dakshayini, "Efficient task allocation approach using genetic algorithm for cloud environment," *Cluster Comput.*, vol. 22, no. 4, pp. 1241–1251, Dec. 2019, doi: 10.1007/s10586-019-02909-1.

[4] Y. Su, Z. Bai, and D. Xie, "The optimizing resource allocation and task scheduling based on cloud computing and Ant Colony Optimization Algorithm," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–9, Aug. 2021, doi: 10.1007/s12652-021-03445-w.

[5] L. Golightly, V. Chang, Q. A. Xu, X. Gao, and B. S. C. Liu, "Adoption of cloud computing as innovation in the organization," *Int. J. Eng. Bus. Manag.*, vol. 14, p. 184797902210939, Jan. 2022, doi: 10.1177/18479790221093992.

[6] K. Braiki And H. Youssef, "Resource Management in Cloud Data Centers: A Survey," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, Jun. 2019, pp. 1007–1012, doi: 10.1109/IWCMC.2019.8766736.

[7] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *J. Netw. Comput. Appl.*, vol. 143, pp. 1–33, Oct. 2019, doi: 10.1016/j.jnca.2019.06.006.

[8] S. A. Murad, A. J. M. Muzahid, Z. R. M. Azmi, M. I. Hoque, and M. Kowsher, "A review on job scheduling technique in cloud computing and priority rule based intelligent framework," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2309–2331, Jun. 2022, doi: 10.1016/j.jksuci.2022.03.027.

[9] M. Usman Sana and Z. Li, "Efficiency aware scheduling techniques in cloud computing: a descriptive literature review," *PeerJ Comput. Sci.*, vol. 7, p. e509, May 2021, doi: 10.7717/peerj-cs.509.

[10] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review," *Cluster Comput.*, vol. 20, no. 3, pp. 2489–2533, Sep. 2017, doi: 10.1007/s10586-016-0684-4.

[11] A. A. Nayak and S. Shetty, "A Systematic Analysis on Task Scheduling Algorithms for Resource Allocation of Virtual Machines on Cloud Computing Environments," in *2023 International Conference on Recent Trends in Electronics and Communication (ICRTEC)*, Feb. 2023, pp. 1–6, doi: 10.1109/ICRTEC56977.2023.10111894.

[12] K. Pradeep, N. Gobalakrishnan, N. Manikandan, L. J. Ali, P. K., and K. . Vijayakumar, "A Review on Task Scheduling using Optimization Algorithm in Clouds," in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, Jun. 2021, pp. 935–938, doi: 10.1109/ICOEI51242.2021.9452837.

[13] B. Wang, C. Wang, Y. Song, J. Cao, X. Cui, and L. Zhang, "A survey and taxonomy on workload scheduling and resource provisioning in hybrid clouds," *Cluster Comput.*, vol. 23, no. 4, pp. 2809–2834, Dec. 2020, doi: 10.1007/s10586-020-03048-8.

[14] S. Varshney, R. Sandhu, and P. K. Gupta, "QoS Based Resource Provisioning in Cloud Computing Environment: A Technical Survey," in *Communications in Computer and Information Science*, vol. 1046, Springer, Singapore, 2019, pp. 711–723, doi: 10.1007/978-981-13-9942-8_66.

[15] S. S. Gill and R. Buyya, "Resource Provisioning Based Scheduling Framework for Execution of Heterogeneous and Clustered Workloads in Clouds: from Fundamental to Autonomic Offering," *J. Grid Comput.*, vol. 17, no. 3, pp. 385–417, Sep. 2019, doi: 10.1007/s10723-017-9424-0.

[16] M. Naghshnejad and M. Singhal, "A hybrid scheduling platform: a runtime prediction reliability aware scheduling platform to improve HPC scheduling performance," *J. Supercomput.*, vol. 76, no. 1, pp. 122–149, Jan. 2020, doi: 10.1007/s11227-019-03004-3.

[17] J. Natarajan, "Parallel Queue Scheduling in Dynamic Cloud Environment Using Backfilling Algorithm," *Int. J. Intell. Eng. Syst.*, vol. 11, no. 2, pp. 39–48, Apr. 2018, doi: 10.22266/ijies2018.0430.05.

[18] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends," *Swarm Evol. Comput.*, vol. 62, p. 100841, Apr. 2021, doi: 10.1016/j.swevo.2021.100841.

[19] W. Khallouli and J. Huang, "Cluster resource scheduling in cloud computing: literature review and research challenges," *J. Supercomput.*, vol. 78, no. 5, pp. 6898–6943, Apr. 2022, doi: 10.1007/s11227-021-04138-z.

[20] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Futur. Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019, doi: 10.1016/j.future.2018.09.014.

[21] R. Istrate, A. Poenaru, and F. Pop, "Advance Reservation System for Datacenters," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Mar. 2016, vol. 2016-May, pp. 637–644, doi: 10.1109/AINA.2016.106.

[22] J. Chamberlain, E. Simhon, and D. Starobinski, "Preemptible queues with advance reservations: Strategic behavior and revenue management," *Eur. J. Oper. Res.*, vol. 293, no. 2, pp. 561–578, Sep. 2021, doi: 10.1016/j.ejor.2020.12.044.

[23] K. P. N. Jayasena and B. S. Thisarasinghe, "Optimized task scheduling on fog computing environment using meta heuristic algorithms," in *2019 IEEE International Conference on Smart Cloud (SmartCloud)*, Dec. 2019, pp. 53–58, doi: 10.1109/SmartCloud.2019.00019.

[24] P. Mallik, A. K. Nayak, and R. Kumar Dalei, "Comparative Analysis of Various Task Scheduling Algorithms in Cloud Environment," in *2021 19th OITS International Conference on Information Technology (OCIT)*, Dec. 2021, pp. 37–41, doi: 10.1109/OCIT53463.2021.00019.

[25] E. Hosseini, M. Nickray, and S. Ghanbari, "Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process," *Comput. Networks*, vol. 206, p. 108752, Apr. 2022, doi: 10.1016/j.comnet.2021.108752.

[26] N. Chitgar, H. Jazayeriy, and M. Rabiei, "DSCTS: Dynamic Stochastic Cloud Task Scheduling," in *2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, Dec. 2019, pp. 1–5, doi: 10.1109/ICSPIS48872.2019.9066063.

[27] M. T. Alam Siddique, S. Sharmin, and T. Ahammad, "Performance Analysis and Comparison Among Different Task Scheduling Algorithms in Cloud Computing," in *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, Dec. 2020, pp. 1–6, doi: 10.1109/STI50764.2020.9350466.

[28] K. M. S. . Bandaranayake, K. P. . Jayasena, and B. T. G. S. Kumara, "An Efficient Task Scheduling Algorithm using Total Resource Execution Time Aware Algorithm in Cloud Computing," in *2020 IEEE International Conference on Smart Cloud (SmartCloud)*, Nov. 2020, pp. 29–34, doi: 10.1109/SmartCloud49737.2020.00015.

[29] R. K. R. Indukuri, S. V. Penmasta, M. V. R. Sundari, and G. J. Moses, "Performance Evaluation of Deadline Aware Multi-stage Scheduling in Cloud Computing," in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, Feb. 2016, pp. 229–234, doi: 10.1109/IACC.2016.51.

[30] P. Y. Zhang and M. C. Zhou, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 772–783, 2018, doi: 10.1109/TASE.2017.2693688.