# **Description: ALI TASK SCHEDULING ALGORITHMS IN CLOUD COMPUTING: A COMPARATIVE STUDY**

***International Journal of Intelligent Computing and***

***Information Science***

# 

|  |  |  |
| --- | --- | --- |
| M. Joundy | S. Sarhan | S. Elmougy |
| Department of Computer Science, [Faculty of Computers and Information](http://www.fci-cu.edu.eg/), Mansoura University- , Egypt | | |
| [manarali8182@gmail.com](mailto:manarali8182@gmail.com) | [Shahenda\_sarhan@yahoo.com](mailto:Shahenda_sarhan@yahoo.com) | [samirelmougy@yahoo.com](mailto:samirelmougy@yahoo.com) |

# ***Abstract:*** *Cloud computing is emerging as a replacement for traditional physical hardware computing in the area of parallel and distributed computing. Clouds consist of a collection of virtualized resources that can be provisioned on demand, depending on the users’ needs. Cloud computing faces the grand quantity of the user groups, as well as the quantity of tasks and massive data, so the processing is also very significant. Scheduling tasks efficiently has become an important problem to be solved in the field of cloud computing. The main aim of this paper is to present a competitive study of some of the standard and hybrid scheduling algorithms over the cloud computing heterogeneous environment. The comparative analysis of SJF, RR, Max-Min, Min-Min and two hybrid techniques is provided in this work to illustrate the performance evaluation in three different datasets. The experimental results shows the benefit of the Min-Min algorithms which overcomes the rest in two of three datasets achieving the least turnaround and waiting time*

***Keywords:*** *Task scheduling, Cloud Computing, Shortest Job First, Round Robin, Min-Min, Max-Min.*

1. **Introduction**

The idea of task scheduling is to assign a set of tasks or processes to a set of computer processors considering, executing more than one process at a time and transmitting multiple flows simultaneously to achieve the optimal performance [1]. In recent years, with the growth of Internet and its services, a new computing model, called Cloud Computing has appeared. Cloud computing has been used to build efficient and powerful architectures in different systems that require complex and multi-tenant computing which allows users to share the resources[2]. Task scheduling in cloud computing environment is considered to be efficient when achieving the optimal assignment of a set of subtasks or different tasks over the available set of resources. Task scheduling algorithms in cloud computing is aiming to minimize the makespan of tasks efficiently. Cloud computing refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner. Cloud computing utilizes the computing resources on the network to facilitate the execution of complicated tasks that require large-scale computation and usually uses low-power hosts to achieve a high usability. Thus, the selecting nodes for executing a task in the cloud computing must be considered [3].

In cloud computing, user applications are run on virtual systems where distributed resources are allocated dynamically. Dynamic load-balancing mechanism has to allocate tasks to the processors dynamically as they arrive. Redistribution of tasks has to take place when some processors are overloaded. Every application is completely different in nature and independent where some require more CPU time to compute complex task, and some others may need more memory to store data [4]. Different scheduling algorithms can be used depending on the type of the task to be scheduled. The scheduling algorithms can utilize better executing efficiency and maintain the load balancing of system. The efficiency of the cloud depends on the algorithms used for task scheduling. Scheduling algorithm could be either a preemptive or non-preemptive type. In non-preemptive scheduling algorithm, no force can stop the execution of a task. On the other hand, a task execution could be stooped in the preemptive scheduling algorithm because of many factors. In order to evaluate scheduling algorithms efficiency, a set of criteria’s are needed including both of turnaround time and waiting time [5]. Turnaround time refers to the total time which is spent to finish the task while the waiting time is the aggregate time a task has been waiting in ready queue. CPU scheduling algorithm does not influence the measure of time among which a task executes or does information return; it influences just the measure of time that a task spends waiting in ready queue. Often, a task can deliver some yield genuinely early and can keep figuring new results while past results are being delivered to the client [6].

As the number of users of cloud computing systems increased, the tasks to be scheduled in cloud increased proportionally. Therefore, there is a need for better algorithms to schedule tasks on these systems. Many researchers have proposed many different preemptive and non-preemptive task scheduling algorithms whether standard scheduling algorithms as Shortest-Job-First (SJF), Round Robin (RR), Max-Min and Min-Min or hybrids of them as RR and FCFS (First-Come First-Serve) and RR and SRTF(Shortest-Remaining-Time-First). So In this paper, we had tested a set of those algorithms using three different datasets to evaluate their performance. The remainder of this paper is organized as following. Section 2 presents the concepts of task scheduling and some related works are discussed. Experimental simulations and results analysis are analyzed in Section 3. Finally, conclusion discussion and future work are given in Section 4.

# **Related Work**

As the number of users of cloud computing systems increased, the tasks to be scheduled in cloud increased proportionally. Therefore, there is a need for better algorithms to schedule tasks on these systems. Here, in this study the researchers concentrated on a set of task scheduling algorithms that will be discussed in this section.

**2.1 Round Robin (RR)**

RR is one of the most preemptive straightforward scheduling algorithms used for tasks in a working framework in which the execution of a task is stopped after a specific period of a time called time quantum. RR scheduling is both basic and simple to execute, and does not suffer from starvation while suffering from a long waiting and response times [7].



Figure 1: Round Robin Algorithm

The main characteristics of RR are:

1. Using too short time quantum causes too many context switches and decreases the CPU efficiency.

2. Using too long time quantum cause, in some cases, poor response time and approximates FCFS performance.

3. Because of high waiting times, deadlocks are rarely met in a pure RR system.

As shown in Figure 1 [18], the main steps of applying RR are:

1. Keep the ready queue as a FIFO queue of processes.
2. A new process is added to the tail of the ready queue.
3. The CPU scheduler picks the first process from the ready queue, sets a timer to one time slot to cause interrupt, and dispatches the process.
4. If the process may have a CPU burst of less than one time quantum

* the process itself will release the CPU voluntarily, and
* the scheduler will then proceed to the next process in the ready queue.

Otherwise “if the CPU burst of the currently running process is longer than one time quantum”,

* the timer will go off and will cause an interrupt to the OS,
* a context switch will be executed, and the process will be put at the tail of the ready queue, and
* the CPU scheduler will then select the next process in the ready queue.

Many researchers had proposed different variations of RR algorithm as Lin et.al. in (2011) proposed an algorithm called Dynamic Round-Robin (DRR) for energy-aware virtual machine scheduling and consolidation. DRR was compared with the GREEDY, ROUNDROBIN and POWERSAVE scheduling and showed superiority in reducing the amount of consumed power [8]. [Yassein](http://dl.acm.org/author_page.cfm?id=94258634574&coll=DL&dl=ACM&trk=0&cfid=714367629&cftoken=83323653) et. al. in (2013) proposed an enhancement to the traditional RR, namely Randomized Round Robin RRR. The enhanced version of RR algorithms is based on random selection for processes that come from different users to achieve near optimal selection of jobs to be served. A simulation had been carried out using CloudSim simulator V 3.0 to test the performance of the proposed scheme in terms of different evaluation metrics such as average throughput and average turnaround time [9].

**2.2 Shortest-Job-First (SJF)**

SJF scheduling distributes the tasks according to the lowest burst time. It could be either preemptive or non-preemptive types, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst. Pre-emptive SJF scheduling is sometimes called shortest-remaining-time-first (SRTF). Although it gives optimal average waiting time, it is more suitable in a batch framework but suffers from starvation [10]. This algorithm is designed for maximum the throughput in most scenarios. This idea is illustrated as shown in the Figure 3 [18].

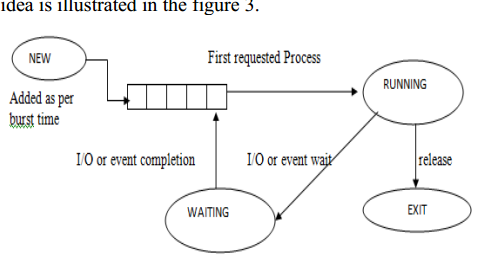


Figure 2: SJF algorithm

The main algorithm of SJF is

|  |
| --- |
| for *i* = 0 to *i* < main queue-size  If task *i+1*.length < task *i*.length then  add task *i+1* in front of task *i* in the queue  end if  If main queue-size == 0 then  task i last in the main queue  end if  end for |

Figure 3: SJF algorithm

The main characteristics of SJF are:

1. The real difficulty with the SJF algorithm is how to predicate the length of the next CPU request.
2. SJF minimizes the average waiting time because it services small processes before it services large ones. While it minimizes average wait time, it may processes with high service time requests. If the ready list is saturated, then processes with large service times tend to be left in the ready list while small processes receive service. In extreme cases, when the system has little idle time, the processes with large service time will never be served.

Many researchers had proposed different variations of SJF as [Jia](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Jia%20Ru.QT.&newsearch=true) and Keung in 2013 proposed a scheduling algorithm integrated with task grouping, priority-aware and SJF (shortest-job-first) to reduce the waiting time and make span, as well as to maximize resource utilization. This proposed scheduling algorithm aims at reducing processing time, waiting time and overhead. In the experiment, tasks are generated using Gaussian distribution and resources are created using Random distribution as well as CloudSim framework is used to simulate the proposed algorithm under various conditions [11]. Yeboah in 2015 introduced an enhancement to the traditional RR with the SJF called (RRSJF) that selects processes based on shortest job first in a round robin fashion to give optimal selection of job. A simulation has been carried out using CloudSim simulator V 3.0 to test the performance of the proposed scheme in terms of different evaluation metrics such as average turnaround, average waiting time and context switches [12].

**2.3 MAX-MIN and MIN-MIN algorithms**

Both of MAX-MIN and MIN-MIN algorithms estimate the execution and completion times of each of the tasks on each of the grid resources. Estimating the execution time of each task on different resources, the Min-min algorithm selects the task with minimum completion time and assigns it to the resource on which the minimum execution time is achieved. Figure 4 shows Min-Min scheduling algorithm [19].

|  |
| --- |
| 1. for all tasks Ti in meta-task Mv  2. for all resources Rj  3. Cij=Eij+rj  4. do until all tasks in Mv are mapped  5. for each task in Mv find the earliest completion time and the resource  that obtains it  6. find the task Tk with the minimum earliest completion time  7. assigne task Tk to the resource Rl that gives the earliest completion time  8. delete task Tk from Mv  9. update rl  10 . update Cil for all i  11.end do |

Figure 4: Min-Min algorithm

The Max-Min is very similar to Min-Min, except that Max-Min assigns task with maximum expected completion time to the corresponding resource [5]. Figure 5 illustrates Max-Min algorithm [20]

|  |
| --- |
| 1. for all submitted tasks in meta-task; Ti  2. for all resources; Rj  3. Cij = Eij + rj  4. While meta-task is not empty  5. Find task Tkconsumes maximum completion time.  6. Assign Tk to the resource Rj which gives minimum execution time.  7. Remove Tk from meta-tasks set  8. Update rj for selected Rj  9. Update |

Figure 5: Max-Min algorithm

Recently, many researchers had introduced different variations of the MIN-MIN and MAX-MIN algorithms. In 2013, Liu et. al. introduced an improved min-min algorithm based on the min-min algorithm considering three constraints quality of service, the dynamic priority model and the cost of service. The proposed algorithm was compared with the traditional min-min algorithm showing that, it can make long tasks execution at reasonable time, increase resource utilization rate and meet users’ requirements [13]. Huankai et. al. in 2013 introduced an improved load balanced algorithm on the ground of Min-Min algorithm in order to reduce the makespan and increase the resource utilization (LBIMM). At the same time, cloud providers offer computer resources to users on a pay-per-use base. In order to accommodate the demands of different users, they may offer different levels of quality for services. Then the cost per resource unit depends on the services selected by the user. In return, the user receives guarantees regarding the provided resources. To observe the promised guarantees, user-priority was considered in the proposed PA-LBIMM so that user's demand could be satisfied more completely [14].

Also in (2013) [Devipriya](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Devipriya,%20S..QT.&newsearch=true) and [Ramesh](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Ramesh,%20C..QT.&newsearch=true) proposed a simple modification of the Max-min algorithm. This algorithm is built based on RASA algorithm and the concept of Max-Min strategy to outperform scheduling process of RASA in case of total complete time for all submitted jobs. The proposed Max-min algorithm is based on expected execution time instead of complete time, so the scheduling tasks within cloud environment using Improved Max-min can achieve lower makespan rather than original Max-min [15]. In 2015, Kaur and Kaur Proposed a holistic analysis algorithm based on Max-Min algorithm, which work on principle of sorting jobs based on completion time. The proposed algorithm also reviews job characteristics in method of size, pattern, payload ratio and available storage blocks in particular cluster of contribution of file systems. The observations show no significant overload due to addition of these constrains, as sorting operation remains same and efficient [16].

There are more and more different scheduling algorithms but in this paper we will stick with six main algorithms and hybrids including four standard algorithms RR (Quantum=8), SJF, MAX-MIN and MIN-MIN and two hybrids RRS and FCFS and RRS and SRTF [17], as they are the most used in this research area.

# **Simulation Settings**

## **Simulation Environment**

The evaluated algorithms were implemented and tested in C# engine environment. The C# engine was programmed by C # 2013 Express Editions as an integrated development tool. The system is a user friendly interface using XML and Windows Foundation Presentation (WPF). The simulation is running on Windows 7 Home Edition x64 bit with Intel Core i5 second generation, 3 MB cache and 4 GB of RAM.

**3.2 Performance metrics**

 The following metrics were considered through the evaluation process:

* *Wait time:* Average time a process spends in the run queue.
* *Turnaround Time*: Average time elapsed from when a process is submitted to when it has completed.
* *Response Time*: Average time elapsed from when a process is submitted until first response
* *Throughput:*  Number of processes completed / time unit.

# **Experimental Results and Evaluation**

For evaluation purposes, three different datasets were used through testing the six evaluated algorithms [RR (Quantum=8), SJF, RRS and FCFS, RRS and SRTF, MAX-MIN and MIN-MIN]. Each dataset consists of 100% randomly generated and dynamically shuffled five tasks denoted as T1, T2, . . .T5 and each task is characterized by its arrival time and burst time, as shown in Table 1.

Table 1: Data sets

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Dataset1 | | Dataset2 | | Dataset3 | |
| Burst Time | Arrival Time | Burst Time | Arrival Time | Burst Time | Arrival Time |
| T1 | 50 | 0 | 250 | 0 | 30 | 0 |
| T2 | 100 | 1 | 200 | 1 | 300 | 1 |
| T3 | 150 | 2 | 150 | 2 | 200 | 2 |
| T4 | 200 | 3 | 100 | 3 | 20 | 3 |
| T5 | 250 | 4 | 50 | 4 | 10 | 4 |

Figure 6 represents the evaluation results of the six implemented algorithms showing that the RR and RRS and SRTF scored the highest turnaround time while SJF, Max-Min and Min-Min achieved the least turnaround time confirming the superiority of the Min-Min algorithm in Datasets 1 and 2 and Max-Min in Dataset 3.

Figure 6: Average Turnaround Time using three different datasets

In Figure 7, it is noticed that also SJF, Max-Min and Min-Min achieved the minimum waiting time while standard RR with quantum=8 achieved the worst waiting time. Confirming, also in this figure that the Min-Min algorithm had proved superiority over all in Datasets 1 and 2 while the Max-Min proved its superiority in Dataset 3. Finally, we can conclude from the experimental evaluation that the Min-Min performed effectively and won the competitive against the six implemented standard algorithms and hybrids in reducing the turnaround and waiting times.

Figure 7: Average Waiting Time Comparison using three different datasets

Response time in cloud environment represents a quick reply to users of services in which it is considered one of the most important metrics in this heterogeneous environment. Figure 8 represents the evaluation results of the six implemented algorithms showing that the SJF score the highest Response time while RR achieved the least Response time, Max-Min and Min-Min have an acceptable Response time, RRS&FCFS and RRS&SRTF have a close ranks confirming the advantage of the Min-Min and Max-Min algorithm over all datasets. RR stander algorithm as known one of the main advantage besides its fairness is the good Response time. In Figure 9 as one of QoS requirements, we measured the throughput to six competitive algorithms. Throughput is measured as total number of tasks finished per deep average of complete time. Reviewing Figure 9, it is noticed that Max-Min and Min-Min achieved the equal maximum throughput in Datasets 1 while standard RR with quantum=8 and SJF achieved the worst throughput. Confirming, also in this figure that the Min-Min algorithm had proved superiority over all Dataset 2 and 3 flowed by Max-Min. Finally, we can conclude from the experimental evaluation that the Min-Min performed effectively and won the competitive against the six implemented standard algorithms and the two hybrids algorithms in reducing the turnaround time, waiting time and throughput, while RR kept its superiority concerning with response time.

Figure 8: Average Response Time Comparison using three different datasets

Figure 9: Throughput Comparison using three different datasets

# **Conclusion and Future work**

With the appearance of the cloud computing environment, different and many trials of finding and proposing the best scheduling algorithm have been introduced. Some researchers tried to adapt the standard task scheduling algorithms as Shortest Job First, Round Robin, Min-Min and Max-Min while others refuged to proposing hybrids as RRS and FCFS and RRS and SRTF. In this study, we implemented six of the most used standard and hybrid algorithms using three different datasets to evaluate their performances. The experimentation results assure that the superiority of the Min-Min algorithm over the state of art algorithms and suitability of RR algorithm to build good hybrid techniques so we recommend researchers in this field to get benefit of this algorithm, SJF showing good waiting time but its bottleneck is starvation problem. In the future, the researchers are willing to proceed in introducing a hybrid of SJF and dynamic RR to be tested in the CloudSim environment to reduce the starvation problem in SJF.

**References**

# Nosrati, M.; Karimi, R., and Hariri, M., "Task Scheduling Algorithms Introduction", World Applied Programming, Vol. 2(6), pp. 394-398, 2012.

1. Lu, C. W., Hsieh, C. M.; Chang, C. H. and Yang, C. T., "An Improvement to Data Service in Cloud Computing with Content Sensitive Transaction Analysis and Adaptation ", 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops (COMPSACW), 22-26 July, pp. 463-468, Japan, 2013
2. Wang, S. C.; Yan, K. Q.; Wang, S. S., and Chen, C. W., "A Three-Phases Scheduling in a Hierarchical Cloud Computing Network", Third International Conference on Communications and Mobile Computing (CMC), 2011. pp. 114-117, 2011.
3. Tayal, Sandeep , "Tasks scheduling optimization for the cloud computing systems.", International Journal of Advanced Engineering Sciences And Technologies (IJAEST), Vol. 5(2). pp. 111-115, 2011.
4. Salot, P. , "A survey of various scheduling algorithm in cloud computing environment,", International Journal of Research and Engineering Technology (IJRET), Vol. 2(2), pp. 2319-1163, 2013.
5. Goel, N.; and Garg, R. B., "A Comparative Study of CPU Scheduling Algorithms", International Journal of Graphics and Image Processing, Vol. 2(4), pp. 245-251, 2012.
6. Mohammadi, A. and Akl, S. G. , "Scheduling algorithms for real-time systems", Technical Report No. 2005-499, School of Computing, Queens University, Kingston, Ontario, Canada, 2005.
7. Lin, Ching-Chi; Pangfeng Liu and Jan-Jan Wu., "Energy-aware virtual machine dynamic provision and scheduling for cloud computing", 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 736-737, Washnighton , USA, 2011.
8. Yassein, M. O. B. and Khamayseh, Y. M., and Hatamleh, A. M., "Intelligent Randomize Round Robin for Cloud Computing", International Journal of Cloud Applications and Computing (IJCAC), Vol. 3(1), pp. 27-33, 2013.
9. Gokilavani, M.; S. Selvi and C. Udhayakumar., "A Survey on Resource Allocation and Task Scheduling Algorithms in Cloud Environment.”, International Journal of Engineering and Innovative Technology (IJEIT), Vol. 3(4), pp. 173-179, October 2013.
10. Ru, J. and Keung, J., "An Empirical investigation on the simulation of priority and shortest-job-first scheduling for cloud-based software systems", 2013 22nd Australian Software Engineering Conference (ASWEC), pp. 78-87, Melbourne, VIC, Australia. 2013.
11. Yeboah, Thomas; Odabi I. Odabi, and Kamal Kant Hiran, "An Integration of Round Robin with Shortest Job First Algorithm for Cloud Computing Environment", International Conference On Management, Communication and Technology (ICMCT), Vol. 3(1), pp. 1-5, 2015.
12. Liu, G.; Li, J. and Xu, J., ” An improved min-min algorithm in cloud computing”, [Proceedings of the 2012 International Conference of Modern Computer Science and Applications](http://link.springer.com/book/10.1007/978-3-642-33030-8), Volume 191 of the series [Advances in Intelligent Systems and Computing](http://link.springer.com/bookseries/11156) pp 47-52, 2013.
13. Chen, H.; Wang, F.; Helian, N. and Akanmu, G., "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing", 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH), pp. 1-8. Feb. 21-Feb. 23, 2013, Bangalore, India, 2013.
14. Devipriya, S. and Ramesh, C., "Improved Max-min heuristic model for task scheduling in cloud", 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), pp. 883-888, Dec. 12-14, Chennai, India, 2013.
15. Kaur, Navdeep and Khushdeep Kaur, "Improved Max-Min Scheduling Algorithm"‏, IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 17, pp. 42-49, 2015.
16. Santra S.; Dey H;  [Dey H.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Dey,%20H..QT.&newsearch=true); [Majumdar S.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Majumdar,%20S..QT.&newsearch=true)and [Jha G.S.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Jha,%20G.S..QT.&newsearch=true), “New simulation toolkit for comparison of scheduling algorithm on cloud computing”, 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, pp. 466–469, Kanyakumari, India, 2014.
17. Kaur A.; Kinger S.,”A Survey of Resource Scheduling Algorithms in Green Computing”, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4) , pp.4886-4890, 2014.
18. Parsa S.; Entezari-Maleki R., “ RASA: A New Grid Task Scheduling Algorithm”, International Journal of Digital Content Technology and its Applications”, Vol. 3(4), pp. 91-99, December 2009.
19. Elzeki O. M.; Reshad M. Z.; Elsoud M. A.,” Improved Max-Min Algorithm in Cloud Computing”, International Journal of Computer Applications, Vol.50 (12), pp.22-27, July 2012.