International Journal of Control and Automation
Vol. 13, No.1, (2020), pp. 334 - 342

A Hybrid MinMin & Round Robin Approach for Task Scheduling in Cloud Computing

Khaldun Ibraheem Arif
Department of Computer Science, College of Education for Pure Science, University of Thi Qar, Nassiriya, Iraq.
khaldun.arif@utq.edu.iq

Abstract
Cloud computing refers to the ability of using and sharing remote system resources over the internet. Task scheduling can be considered as one of major challenges of QoS which tries to distribute tasks to Virtual Machines (VMs) in an efficient manner. This paper offers a Hybrid Min-Min and Round Robin (RR) scheduling algorithm (HMMRR) of traditional algorithms for improving resource utilization and system performance through minimizing makespan (execution time) of all VMs and reducing average of response time (starvation) as well as waiting time of the system. The proposed algorithm has been compared with other existing algorithms such as RR, Min-Min and Max-Min where the experimental results show that the HMMRR algorithm outperforms others.

Keywords: Task scheduling, Makespan, Response time, Cloud computing, Quality of service (QoS), Waiting time

1. Introduction

Cloud computing enables the management delivery of services over the internet. It provides services as pay on demand basis [1, 2]. The services may be hardware, software, application platform, or database. These types of services are known as [5, 7, 18]: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), Database as a Service (DaaS). For the first type (SaaS), the providers put required application in the cloud in order to be available for others. In PaaS is similar to SaaS where makes computing platforms such as operating system or programming language available for users. While in Iaas, the service provider allows a whole computing infrastructure to be available as a service. With DaaS, the service provider stores a database system as a service.

Cloud computing is a form of parallel and distributed system where resources are shared among computers on demand and after paying for use [1, 14]. Cloud system is consists of some components like (see figure 1): Users: requesting for services, Broker: plays as an intermediate between user and service provider, and Hosts (Servers): providing services to the user. VM is a basic processing unit where cloud computing technique based on.The unit which is responsible on management all Virtual Machines (VMs) is called VM Manager (VMM) [6, 7]. However, cloud computing suffers from some challenges such as resource scheduling, load balancing, security and privacy, QoS management [8, 17]. Therefore, a most suitable mapping algorithm between clients' tasks and resources would reduce makespan and response time and waiting time at the same time would increase resource utilization and throughput (number of finished tasks per time unit) which is the objective of our research. Generally, there are various algorithms designed for task scheduling used in cloud, such as Round Robin (RR) [4, 5], Min-Min [5, 16], and Max-Min [3, 5]. However, these algorithms suffer from some drawbacks like high makespan, response time and waiting time. In this paper, we present a hybrid algorithm of Min-Min and RR (HMMRR) which attempts to overcome these demerits.

The remainder of this work is organized as follows. Section 2, reviews some of related work. In section 3, the proposed technique is discussed. Section 4, explains simulation environment. Experimental results are depicted in section 5. Obtained results are discussed in section 6. Section 7, includes conclusions and future work.

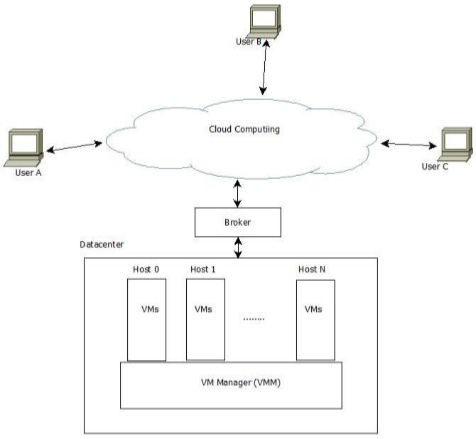

Figure 1. Cloud computing environment

2. Related Work

In last decade, there are many of task scheduling and load balancing algorithms. However, they are not always yield minimum makespan and response time as well as maximize resource utilization and throughput.

Dhari A. et al. [2] proposed an algorithm for load balancing called LBDA (Load Balancing Decision Algorithm). It consisted of three stages as follows: 1- Calculating VM capacity and its current load. 2- Computing time required to achieve tasks over each VM. 3- Making decision to distribute tasks to VMs. LBDA was compared with existing Max-Min, SJF, and RR algorithms and the results proved that it outperforms others.

Elmougy S. et al. [9] proposed a novel hybrid algorithm from both SJF and RR in which it partitions ready queue into two small queues (Q1, Q2) and puts short tasks in Q1 whereas long ones in Q2. Mapping tasks to resources is achieved mutually two tasks from first queue (Q1) and one task from second queue (Q2). The presented algorithm is compared with other scheduling algorithms such as SJF, RR, and TSPBRR. Obtained results indicate that the proposed algorithm outperforms others with respect to waiting time, response time and in somewhat the starvation of large tasks.

Alworafi M. A. et al. [10] presented a hybrid algorithm (HSLJF) as a combination of SJF and LJF schemes. Initially, sorting tasks in ready queue according to their burst time in ascending order. Then takes one task from front of queue (short) and one from the tail of queue (long) and so on. The taken task would assign to a VM having minimum completion time. The extracted results show that HSLJF is better than SJF, LJF and RR in terms of makespan, response time and actual execution time.

In [11], Zouaoui S. et al. introduced a new algorithm for improving real time of operating system as related to CPU performance. The proposed approach is a combination of RR and priority based (PB) algorithms which reduced starvation as well as obtaining advantage of priority scheduling. The experimental results show the superiority of the proposed scheme as compared with RR with respect to average waiting time and average turnaround time.

AL-MAYTAMI B. A. et al. [12] proposed a new scheduling algorithm based on Directed Acyclic Graph (DAG) and the Prediction of Tasks Computation Time algorithm (PTCT). The presented algorithm improves makespan and minimizes the computation as well as complexity by applying Principle Components Analysis (PCA). Also, it reduces Expected Time to Compute (ETC) matrix. The introduced algorithm is compared with Min-Min, QoS-Guide and MiM-MaM algorithms and the simulated results clarify superiority of proposed algorithm over others according to efficiency, speedup, and scheduling length ratio.

Parsa S. et al. [13] proposed a new algorithm for task scheduling suitable for grid environment based on both Min-Min and Max-Min scheduling algorithms called RASA. This algorithm executes Min-Min and Max-Min alternatively. RASA compared with traditional algorithms Min-Min and Max-Min and the experimental results demonstrate that proposed approach had achieved better than others in terms of makespan.

Banerjee S. et al. [14] presented a new policy for distribution cloudlets to the VMs with load balancing technique. The experimental results proved that it improves completion time of cloudlets and minimizes makespan of the VM(s) and host(s) as well as improves QoS. These results were obtained using CloudSim 3.0.3 toolkit.

ABU KHURMA R. et al. [15] review various task scheduling algorithms suitable for cloud computing such as: RR, MaxMin, MinMin, FCFS, MCT, PSO, GA as well as case study on Modified RR (MRR). The results demonstrate that MRR performs better than RR since it reduces average waiting time and promote merits of RR such as fairness and reducing starvation.

3. Proposed Work

Task scheduling is considered one of the important fields which affects the performance of overall system. Therefore, finding an efficient algorithm for mapping tasks to VMs and satisfaction QoS parameters is critical. So we present a combination of two traditional algorithms namely Min-Min and RR for getting merits of them. The Min-Min algorithm selects each time the shortest task from meta-task queue and assigns it to the most fast VM (if available) while RR distributes tasks to VMs through circular queue. Therefore, HMMRR attempts to sort tasks in ascending order according to their lengths (burst time) that are generated randomly in range 300-1000 MI and arranges VMs in descending order with respect to their speed (MIPS) that are also generated randomly then applying RR as shown in example 1.

Example 1: Suppose we have four tasks {t0, t1, t2, t3} and two VMs {vm0, vm1} as depicted in tables 1 and 2 where MIPS refers to (Million Instruction Per Seconds), MI denotes (Million Instruction), and Mbps means (Million bit per second). Completion time of tasks on each VM are calculated in table 3. While the mapping of tasks to VMs according to RR are as in the following set: {(t0, vm0), (t1, vm1), (t2, vm0), (t3, vm1)}

Table 1. Resources Specifications

Resources (VMs)	VM speed (MIPS)	Bandwidth (Mbps)
vm0	400	45
vm1	300	80

Table 2. Tasks Specifications

Tasks	Instruction length (MI)	Data length (Mb)
t0	800	75
t1	600	30
t2	500	23
t3	700	19

ISSN: 2005-4297 IJCA

Table 3. Completion Time of Tasks Calculated in Seconds

Tasks / VMs	vm0	vm1
t0	2	2.7
t1	1.5	2
t2	1.25	1.7
t3	1.75	2.3

Figure 2 illustrates Gantt chart of running Min-Min algorithm where the value of makespan is 6.5 seconds.

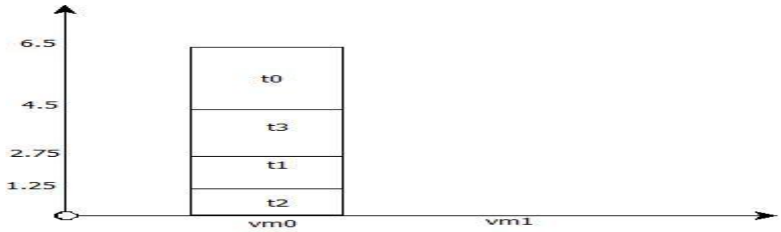

Figure 2. Gantt Chart of MinMin Algorithm

After applying our algorithm, mapping set would be {(t2, vm0), (t3, vm0), (t1, vm1), (t0, vm1)} and makespan value is 4.7 seconds.
The steps of the proposed HMMRR algorithm will be described as follows. Initially, tasks in the ready queue will be sorted in ascending order according to their length and VMs will be arranged in descending order in terms of their MIPS. Then it calculates the maximum capacity of every VM as shown in Eq. 1 [14]. Next, in order to achieve Completion Time (CT_{ij}) of each task (i) over every VM (j) by using Eq. 2 [6].

$VM_j.capacity = (VM_j.MIPS / \sum_{k=0}^{M-1} VMk.MIPS) * 100$ (1)
$CT_{ij} = task_i / (VM_j * PE)$ for i=1,2,…,N; j=1,2,…,M (2)

Where N refers to number of tasks and M indicates the number of VMs while PE refers to the number of processing elements (cores).
Loading of every VM can be computed as in Eq. 3 [6, 10].
$Load\ VM_j = \sum_{i=0}^{N-1} CTij$ (3)

The mapping relation between $task_i$ and VM_j may be formulated as in Eq. 4
$Assign_{ij} = \begin{cases} 1 & if\ taski\ is\ assigned\ to\ VMj \\ 0 & otherwise \end{cases}$ (4)

Makespan is calculated as in Eq. 5 [6, 14]
$Makespan = \max \{Load\ VM_j\}$ for j = 1,2,…,M (5)

While Response time (RT) which represents the difference between starting time of a $task_i$ and its submitting time to the ready queue as formulated in Eq. 6 [19].
$RT_i = task_i.start\_time – task_i.submit\_time$ (6)

Hence, the average of RT can be calculated as in Eq. 7 [10]
$Avg\_RT = \sum_{i=0}^{N-1} RT / N$ (7)

To compute the Average Resource Utilization (ARU) which indicates the ratio of utilization resources (see Eq. 8 [1, 10]).
$ARU = (\sum_{j=0}^{M-1} Load\ VMj) / (Makespan * M)$ (8)
Also, Waiting Time (WT) for a task (i) can be calculated as in Eq. 9 [19].
$WT_i = TT_i – (TL_i – ART_i)$ (9)

Where TT stands for Turnaround Time, TL denotes Task Length and ART refers to ARrival Time.

Thus, Average Waiting Time (AWT) may be computed as in Eq. 10

$$AWT = \sum_{i=0}^{N-1} WTi \, / \, N \tag{10}$$

See also tables 7 and 8.

The steps of our algorithm can be listed as follows.

////////////////////////////// Proposed Algorithm (HMMRR) //

Input: List of tasks (N tasks) and List of VMs (M VMs)

Output: Makespan, ART, ARU, and AWT

1. Arrange tasks according to their lengths in ascending order.
2. Arrange VMs with respect to their MIPS in descending order.
3. Calculate the maximum capacity of each VM from Eq. 1
4. For (i=0; i<N; i++)
5. For (j=0; j<M; j++) {
6. Computer CT_{ij} from Eq. 2 ;
7. set $load_j = 0$; }
8. For (i=0; i<N; i++)
9. For (j=i mod M; j<M; j++)
10. If $((load_j + CT_{ij}) \leq VM_j.capacity)$ {
11. $Task_i.start = load_j$;
12. $load_j = load_j + CT_{ij}$;
13. $Task_i.finish = load_j$;
14. $Assign_{ij} = 1$;
15. Break; }
16. Else $Assign_{ij} = 0$;
17. Calculate makespan according to Eq. 5;
18. Calculate Avg. of response time from Eq. 6 and 7;
19. Compute ARU from Eq. 8;
20. Compute AWT from Eq. 9 and 10;
21. End

---

4. Simulation Environment

The proposed algorithm with other traditional algorithms were executed and tested using Borland C++ version 5.0 on a computer included Intel Core i5 processor, 2.50 GHz CPU speed, and 4 GB RAM. The presented algorithm was compared with RR, Min-Min, and Max-Min. Table 4 describes simulation parameters of the cloud.

Table 4. Parameters of Simulation

Parameter	Value
No. of datacenters	1
No. of hosts	1
Host memory	1024 MB
Host storage	2 TB
Host bandwidth	10000
VM MIPS	10000
VM image	10000 MB
VM memory	512 MB

VM bandwidth	1000
No. of PE per VM	2
System architecture	"x86"
Operating system	"Windows 7"

5. Experimental Results

The results of comparison the proposed algorithm with other algorithms in terms of makespan is illustrated in table 5 and figure 3 measured in seconds. It discovers that our algorithm (HMMRR) yields minimum value compared with others.

Table 5. Comparison With Respect to Makespan (in Sec.)

# of tasks	# of VMs	RR	MinMin	MaxMin	HMMRR
500	5	7.416	23.279	23.279	7.369
500	10	7.064	15.040	15.040	5.082
1000	10	8.901	21.329	21.372	6.764
1000	15	5.906	13.365	13.399	4.922
1500	15	8.817	12.021	12.022	5.757
1500	20	6.229	8.523	8.525	4.304
2000	20	8.644	11.798	11.803	4.960
2000	25	7.113	9.161	9.163	4.123

Table 6 describes average of response time of the proposed and others measured in seconds. It is obvious that HMMRR leads to optimal value; see also figure 4. The measurement of ARU and AWT of the proposed and studied algorithms are shown in tables 7 and 8 for various number of tasks and VMs; also see the figures 5 and 6. In figure 7, run time results of the proposed algorithm are illustrated when the number of tasks are 2000 and number of VMs are 25.

Table 6. Comparison With Respect to Average Response Time (in Sec.)

# of tasks	# of VMs	RR	MinMin	MaxMin	HMMRR
500	5	3.13	7.882	14.302	2.012
500	10	1.522	4.902	9.01	1.336
1000	10	3.149	6.809	8.608	3.383
1000	15	1.82	4.541	5.473	1.797
1500	15	2.575	4.169	4.595	1.721
1500	20	1.663	3.118	3.64	1.165
2000	20	2.651	3.218	2.95	1.571
2000	25	2.061	2.584	2.427	1.146

Table 7. Comparison According to Resource Utilization (in Sec.)

# of tasks	# of VMs	RR	MinMin	MaxMin	HMMRR
500	5	0.699	0.2	0.2	0.720
500	10	0.540	0.1	0.1	0.634
1000	10	0.636	0.126	0.126	0.657
1000	15	0.640	0.138	0.137	0.649
1500	15	0.475	0.231	0.231	0.655
1500	20	0.480	0.229	0.228	0.621
2000	20	0.525	0.337	0.336	0.689
2000	25	0.519	0.326	0.325	0.654

Table 8. Comparison According to Average Waiting Time (in Sec.)

# of tasks	# of VMs	RR	MinMin	MaxMin	HMMRR
500	5	2.925	6.377	12.810	1.816
500	10	1.611	4.102	8.259	1.265
1000	10	3.371	6.594	8.363	3.340
1000	15	1.901	4.640	5.690	1.893
1500	15	2.591	4.333	5.640	1.793
1500	20	1.799	3.270	4.079	1.277
2000	20	3.194	3.665	3.422	1.950
2000	25	2.554	3.032	2.895	1.540

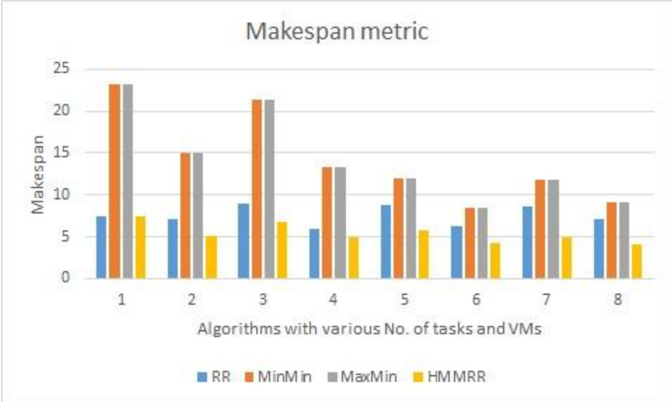

Figure 3. Results of Makespan

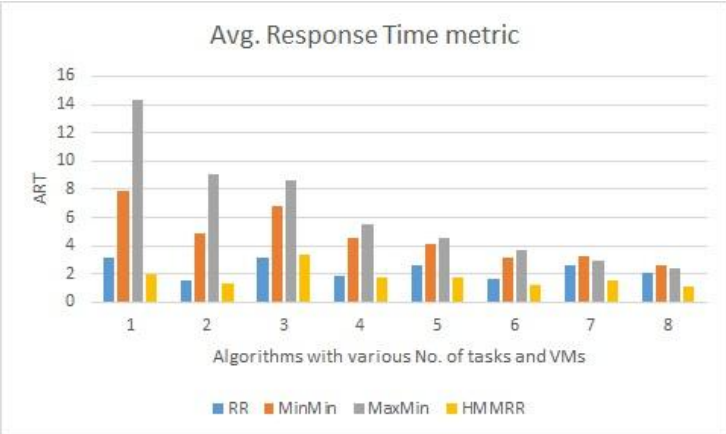

Figure 4. Results of Avg. Response Time

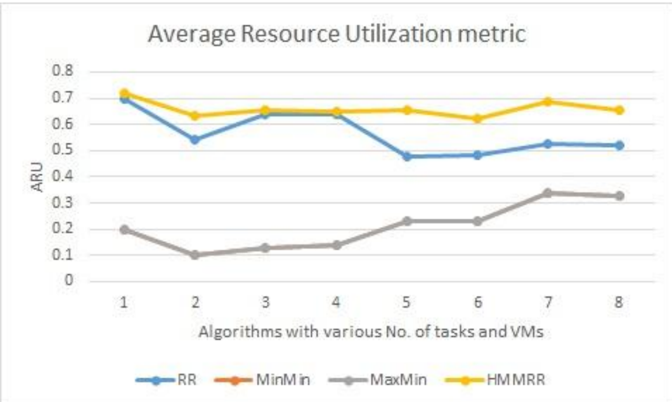

Figure 5. Results of ARU

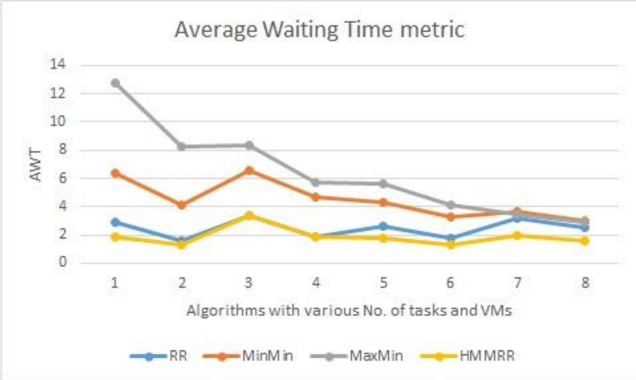

Figure 6. Results of AWT

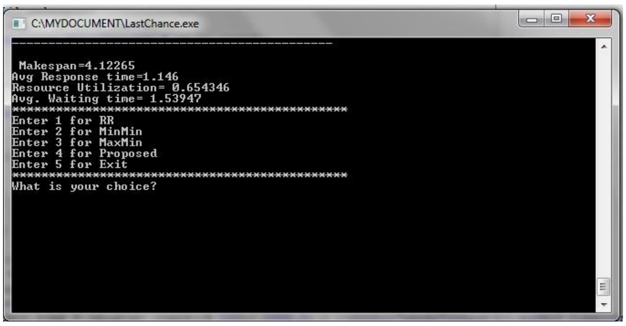

Figure 7. Execution of Proposed Algorithm

6. Discussion Experimental Results

The main reason behind selecting the combination of both Min-Min and RR is to get advantages of each one. Experimental results are carried out based on four quality metrics in order to measure system performance namely makespan, response time, waiting time and resource utilization. An optimal algorithm which minimizes the first three metrics and maximizes the last one. Figure 3 shows that our algorithm (HMMRR) decreases the value of makespan in all scenarios which leads to good load balancing and system performance in comparison with covered traditional algorithms. In figure 4, it is also seen that the proposed algorithm yields smallest value in all cases of average response time which is preferred in real time systems since the highest value would cause starvation (i.e., some tasks will wait in the ready queue (meta-task) for a long period of time). From figure 5 and 6, the comparative study shows that the presented algorithm has better results than others according to resource utilization and average waiting time respectively. So it is noticed that when increasing amount of tasks and VMs, the results become much suitable in terms of quality metrics.

7. Conclusion and Future Work

This paper presented a hybrid Min-Min RR scheduling algorithm for satisfaction QoS metrics in reducing makespan of VMs, average response time of processing tasks as well as average of response time whereas increasing resource utilization and overall system performance. It tries to get the advantages of traditional Min-Min and RR techniques since they yield good results especially in dealing with large amount of processing. In HMMRR, sorting tasks increasingly according to their length and arranging VMs decreasingly in terms of their speed (MIPS) in order to execute shortest task first over fastest VM which is the policy of Min-Min technique and distributing tasks over VMs under circular queue which is the policy of RR gave us encouraged results. The simulation results show that the proposed algorithm has superiority over existing algorithms in terms of minimizing makespan, average response time and average waiting time while on the other side, increasing resource utilization and performance particularly with a big number of tasks and resources. In future work, deadline constraints could be used to support real-time system as well as focusing on improving energy consumption in addition to task scheduling.

References

[1] D. C. Devi and V. R. Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks," vol. 2016, 2016.
[2] A. Dhari and K. I. Arif, "An Efficient Load Balancing Scheme for Cloud Computing," *Indian J. Sci. Technol.*, vol. 10, no. 11, 2017.
[3] U. Bhoi, P. N. Ramanuj, and W. S. Email, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing," vol. 2, no. 4, pp. 259–264, 2013.
[4] M. A. F. Al-Husainy, "Best-job-first CPU scheduling algorithm," *Inf. Technol. J.*, vol. 6, no. 2, pp. 288–293, 2007.
[5] J. Y. Maipan-uku, A. Mishra, A. Abdulganiyu, and A. Abdulkadir, "An Extended Min-Min Scheduling Algorithm in Cloud Computing," *i-manager's J. Cloud Comput.*, vol. 5, no. 2, p. 20, 2018.
[6] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," *Procedia Comput. Sci.*, vol. 115, pp. 322–329, 2017.
[7] A. Hota, S. Mohapatra, and S. Mohanty, "Survey of different load balancing approach-based algorithms in cloud computing: a comprehensive review," in *Computational Intelligence in Data Mining*, Springer, 2019, pp. 99–110.
[8] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egypt. informatics J.*, vol. 16, no. 3, pp. 275–295, 2015.
[9] S. Elmougy, S. Sarhan, and M. Joundy, "A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique," *J. Cloud Comput.*, vol. 6, no. 1, p. 12, 2017.
[10] M. A. Alworafi, A. Dhari, S. A. El-booz, A. A. Nasr, A. Arpitha, and S. Mallappa, *An Enhanced Task Scheduling in Cloud Computing Based on Hybrid Approach*. Springer Singapore, 2019.
[11] S. Zouaoui, L. Boussaid, and A. Mtibaa, "Priority based round robin (PBRR) CPU scheduling algorithm," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 1, p. 190, 2019.
[12] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, "A Task Scheduling Algorithm With Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing," *IEEE Access*, vol. 7, pp. 160916–160926, 2019.
[13] S. Parsa and R. Entezari-Maleki, "RASA: a new grid task scheduling algorithm," *Int. J. Digit. Content Technol. its Appl.*, vol. 3, no. 4, pp. 91–99, 2009.
[14] S. Banerjee, M. Adhikari, S. Kar, and U. Biswas, "Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud," *Arab. J. Sci. Eng.*, vol. 40, no. 5, pp. 1409–1425, 2015.
[15] R. A. B. U. KHURMA, H. AL HARAHSHEH, and A. SHARIEH, "TASK SCHEDULING ALGORITHM IN CLOUD COMPUTING BASED ON MODIFIED ROUND ROBIN ALGORITHM.," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 17, 2018.
[16] K. Etminani, M. Naghibzadeh, and N. R. Yanehsari, "A hybrid min-min max-min algorithm with improved performance," *Dep. Comput. Eng. Univ. Mashad*, 2009.
[17] X. He, X. Sun, and G. Laszewski, "A QoS guided scheduling algorithm for grid computing," in *Proc. of the Int'l Workshop on Grid and Cooperative Computing*, 2002.
[18] K. I. Arif and A. S. Ketab, "Dynamic Time Quantum for an Efficient Round Robin in Cloud Computing," *J. Comput. Theor. Nanosci.*, vol. 16, no. 5–6, pp. 2404–2409, 2019.
[19] A. Silberschatz, G. Gagne, and P. B. Galvin, *Operating system concepts*. Wiley, 2018.

Khaldun Ibraheem Arif was born in Baghdad, Iraq in 1973. He received B.Sc. degree and M.Sc. degree in computer science at University of Basrah, Iraq from 1995 to 1997. He got Ph.D. degree in IT at Second University of Naples, Italy in 2010. He is currently Assistant Professor and he had occupied head of computer science department at University of Thi_Qar during period 2014-2017. He published more than 17 articles at various scientific journals. Interested topics are Computer Security, Load Balancing, Task Scheduling and Cloud Computing.