Subvention Scheme for the Cloud Computing Scheduling Algorithm

Abdellatif I. Moustafa*, Reem M. Ganadily, Shahad Y. Alzahrani, Majid M. Alotaibi
Dept. of Computer Eng., College of Comp. & Info. Sys., Umm AL Qura Univ., Makkah, Saudi Arabia.

* Corresponding author. Tel.: (+966) 5561 10917; email: aisemeia@uqu.edu.sa
Manuscript submitted January 5, 2019; accepted March 11, 2019.
doi: 10.17706/jcp.14.9.571-579

Abstract: Resource management and task scheduling are necessary for satisfying proper job (task) scheduling. Scheduling defines the process or methodology for data flows which are implemented to insure resource high utilization. One of the most used algorithms is shortest job first (SJF) in which deciding jobs order is queued to be processed, with cloud computing more than one virtual machine could be used. Distributed scheduling schemes need to be efficient for better performance. In this research, we proposed a new, efficient approach for allocating and scheduling users' tasks in the cloud environment. The main goal is to enhance the performance metrics such as: 1-reducing the waiting time and 2- increasing the throughput of a given task set to have an efficient mapping of tasks using the same shortest job first (SJF) scheduling algorithm. The proposed policy is tested to conduct performance evaluation.

Key words: Cloud computing, quality of service, scheduling algorithm, SJF algorithm, throughput, waiting time.

1. Introduction

Cloud Computing (CC) is a group of services provided over the Internet; whether software or hardware, they can be reached from anywhere, including office programs or space to store/process files. CC appears in some types of computing, such as parallel computing, distributed computing, grid computing, and utility computing. It is defined as the execution of these computer science concepts. Because of the high performance and low cost of services, with the data available when needed, these computing services are widely desirable; such services can be accessed whenever the Internet is available.

With the advent of the latest trends and advances in communication, computation, and technology, CC have received much interest and attracted the attention of researchers in many disciplines [1]. Therefore, CC has brought forward a dominant platform to serve businesses and individuals with an effective, low-cost paradigm [2]. It has recently come to be considered an interesting field of study. CC is a new technological environment that permits clients to use various huge pools of resources at any time and from anywhere, delivered as a service over the Internet [2].

The need for computing is increasing daily, as the associated technologies are also evolving. CC applies the concept of Virtual Machines (VMs), which divides physical resources into several logical units to enable application (software or hardware) abstraction. Thus, dynamic and resources on the cloud are controlled through virtualization that maps the VM's services to physical servers. Job scheduling can be done efficiently by employing both static and dynamic parameters [3].

CC is a large computation service platform that includes datacenters, storage, networks, firewalls, and

software. The main advantages of the cloud are its fast computing power, low cost of services, high performance, scalability, accessibility and availability [4]. CC is determined by the commercial-centered model; the architecture of CC is fully distributed and difficult to manage. It is like a huge server that owns unlimited resources and services. CC allocates an abundant amount of resources to do different tasks (services) for customers [5].

The CC environment is highly dynamic according to the rapid changes in resource utilization and acquisition over time. Typically, cloud service providers leverage computing resource times either peak load or non-peak resources. Job scheduling (a "job" refers to a user processing or resource request) is one of the most well-known optimization problems faced by CC service providers.

It is known that CC includes two components, namely the client side and server side. Clients request services and resources and servers respond to them. All the requests from clients first go to the master processor of the server side, which consists of several slave processors. Then, the master processor sends all the requests to one of the available slave processors with free time. The other slave processors could be busy with their assigned jobs, and they will return to the idle state once they finish them [6]. The concept of virtualization is directly connected to cloud systems; it refers to something that is not real but can support many services in reality. End users can employ and benefit from these provided services over a remote datacenter, either partially or fully [7].

Many concepts are implemented by clouds, such as virtualization, distributed application design, grids, and enterprise information technology (IT) management. Thus, the task of job scheduling is challenging, since it is necessary to arrange and manage this access to the pool of resources with satisfactory Quality of Service (QoS), reasonable waiting time, good throughput, effective CPU utilization, optimal CPU speed, and optimized scheduling algorithm. In a cloud environment, the case is quite different; the demand for resources varies frequently, according to the dynamic request of resources [8].

The cloud resources issue is one of the main topics that should be focused on to achieve high utilization of resources and fulfill customers' needs and requirements. Resource governance can be divided into two domains, namely resource management and task scheduling; managing and scheduling are necessary for satisfying proper resource allocation. Scheduling defines the processor methodology and data flows implemented to access system resources. The scheduling parameters are concerned mainly with [9]:

Throughput: represents the total number of completed processes per time unit;

Turnaround: represents the time required to complete the process from its arrival; and

Response time: represents the period of time started since submission to the first response produced.

Scheduling algorithms deployed in distributed systems aim to schedule received jobs and map them to the available resources in a reduced time and compute the optimal scheduling of upcoming jobs to be executed under logic constraints. One of the key advantages of scheduling approaches is accomplishing the optimal system throughput with high-performance computing [8].

The Job Scheduler (JS) performs a job assignment task based on fairness and a load-balancing procedure to achieve a fruitful division of resources per request with an optimal execution time [10]. Task scheduling is close to the creation of a mapping relationship between tasks and resources through finding the optimal cloud task scheduling strategy and suitable mapping relationship between users' requests and available resources [11].

2. Literature Review

The authors in [12] involved a comparison between the first-in-first-out (FIFO) algorithm and the one proposed in. The proposed algorithm was implemented in a decentralized computing environment based on distributed grid scheduling of multi-agent systems that dynamically utilize resources (on demand) on

the global task queue basis. The experiments showed that the proposed algorithm has proved its robustness, adaptability, scalability, and alleviation of computing. However, the proposed approach still needs to be improved and compared with other metrics, such as response time. Thus, a real-world scenario should be developed by collecting data on the exact distribution of process demand.

A novel Simulated Annealing (SA) algorithm was proposed in [13] to schedule tasks in a CC environment due to its capacity to produce comparative results with minimum time of execution. The SA approach is adapted in the study to realize the potential performance of CC with a competitive choice. The proposed approach confirmed its capacity to guarantee QoS, execution time, cost, and deadline, as well as obtaining maximum profits for cloud service providers.

A scheduling algorithm based on a heuristic approach called a Genetic Algorithm (GA) was presented in [14] to reduce the total waiting time for cloud users. The nodes in the cloud environment are divided into two types, namely the Cloud Service Provider (CSP) and Cloud User (CU); the CSP stores all the CU's requests in the Request Queue (RQ) and Buffer Queue (BQ), which interact with the GA Queue Sequencer (GAQS). The main role of the GAQS, the JS, is selecting the best schedule of tasks and allocating a specific resource from the Resource Pool (RP) with the minimum Waiting Time (WT) based on the Round Robin (RR) scheduling algorithm. The results illustrate the enhanced performance of the proposed scheduling algorithm based on GA with the minimum optimum average waiting time and maximum throughput.

The proposed algorithm in [15] has improved the efficiency and performance of datacenters, mainly in terms of cloud simulation. A combination of RR and shortest job first (SJF) was presented to improve the process of load balancing in a cloud environment. Cloud Simulator (CloudSim) was used to simulate the proposed algorithm implemented in Java language for load balancing and scheduling in datacenters consisting of VMs.

An efficient scheduling algorithm was proposed in [16] to enhance the load balancing for efficient job scheduling and decreased response time, as well as improved availability of VMs to serve the requests coming from user nodes. The results showed the improved performance based on the minimized waiting time and turnaround time in SJF. However, SJF still faces many difficulties in forecasting the burst time of the process. To handle this issue, the experiments still need to be conducted with a Dynamic Time Quantum SJF schedule.

3. Methodology

One of the fundamental issues in the cloud environment is task scheduling, which plays a key role in the efficiency of all the CC facilities. Our aim is improving an algorithm for task scheduling on the cloud, considering load balancing, runtime factors and resource availability. The main goal is:

Reducing the waiting time for tasks and;

Increasing the throughput for the overall system

The scheduling algorithm has to satisfy efficient mapping of tasks to the resources (VMs). The proposed policy is simulated using Cloudsim-3. In order to get the completely random cloudlet length and randomly sorted we created a JAVA random length cloudlet generator that provides cloudlet sets of different digit-length (4-9 digit), each set of random count of cloudlets.

4. Proposed Algorithm

The algorithm we used is SJF based. It enhances the waiting-time and throughput, hence it enhances the quality of service. The enhanced algorithm is to detect any idle VM, as soon as detecting Idle VM when it finishes the tasks that was assigned to it. We build the modified algorithm to collect and sort all unprocessed including under execution tasks in the cloudlets in a queue. The sorted queue is resubmitted

to the VMs. These steps are repeated until all cloudlets are processed all tasks assigned to them. The proposed modified algorithm is summarized in the following steps:

1- A user sends the task (T) to the cloud.
2- Every task is stored in the job queue (JQ).
3- The tasks are rearranged in the queue according to the shortest time needed (STQ).
4- Divides the number of tasks by the number of VMs.
5- Submitted the groups of tasks that were ascending order among the number of VMs.

The first group will be completed first, and the first hardware resource (VM) will be idle while the rest of the devices are still working; We will collect all unprocessed including under execution tasks on the cloudlets, Store all in job queue then resubmit all equally to available VMs. Halting the process at that point and repeat the process (steps 2–5) for the rest until all the tasks are finished Fig. 1 and Fig. 2 Illustrate processing of the algorithm. For instance, if we used 100 tasks which are divided among the three VMs. With the proposed algorithm, three rounds are achieved to finish the whole tasks, Fig. 3 Illustrates the work flow of the algorithm.

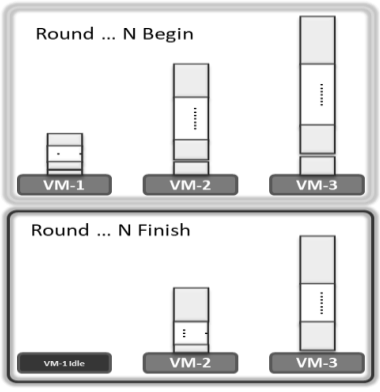

Fig. 1. Processing round begin and finish.

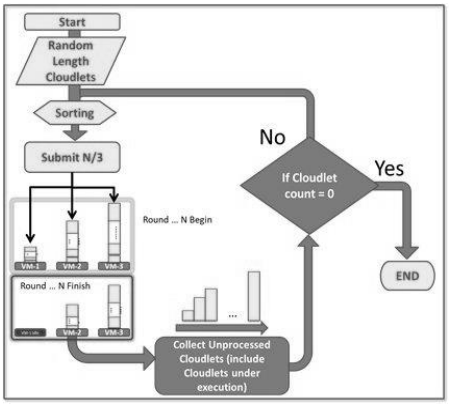

Fig. 2. Enhanced SJF Illustration.

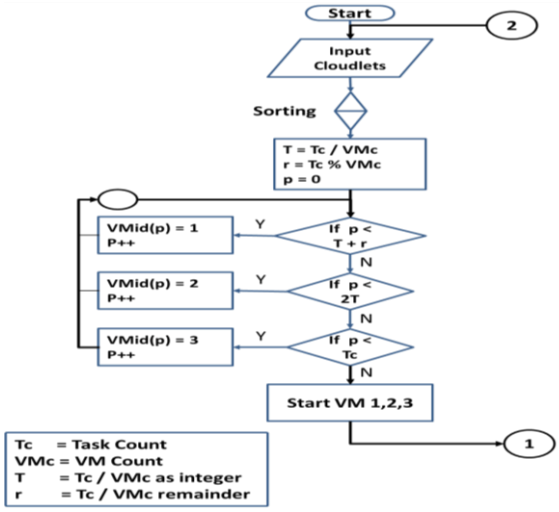

Fig. 3-a. MOD-SJF Flowchart of the algorithm for task scheduling.

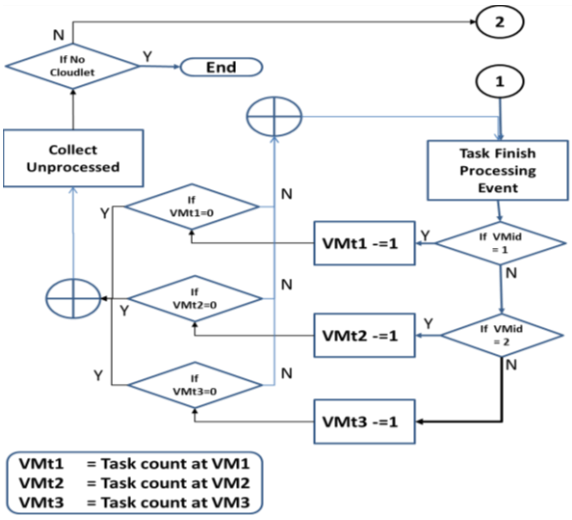

Fig. 3-b. MOD-SJF Collect unprocessed cloudlets.

For instance, with 100 tasks which are divided among the three and five VMs. With the proposed algorithm, three rounds are achieved to finish the whole tasks, Fig. 2 illustrates that and Table 1 (a-b) show

the distribution.

It is true that the fairness is affected as the assigned tasks, but on the other hand, there is fairness in the execution time distribution which shorten the whole tasks' execution. In Fig. 5 The finish time in the proposed algorithm 33-52% than the SJF. In Fig. 6 the throughput in the proposed algorithm was calculated and SJF algorithm based on the following equation:

$$\text{Throughput} = \text{Nums of Exec. Tasks} / \text{Total Finish time} \quad (1)$$

Half of the SJF algorithm is being checked in terms of and the proposed algorithm shows enhancements as the expiration time. The Waiting Time in the proposed algorithm is also out performed as well as the Turnaround Time Fig. 7, 8 represent them, obtained based on :

$$\text{Turnaround Time} = \text{End Time} - \text{Arrival Time} \quad (2)$$

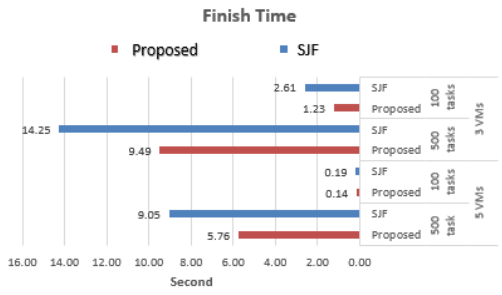

Fig. 4. Finish Time (Sec).

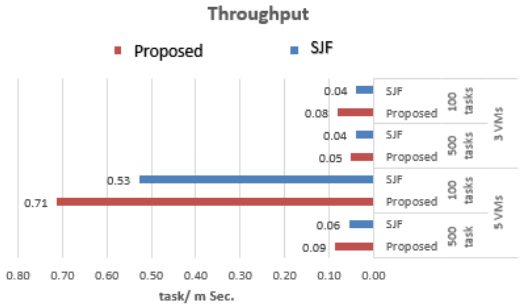

Fig. 5. Throughput (Task/m Sec.)

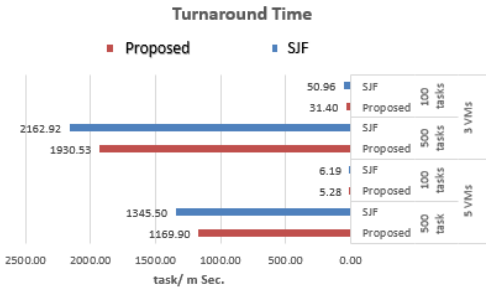

Fig. 6. Turnaround Time (Sec).

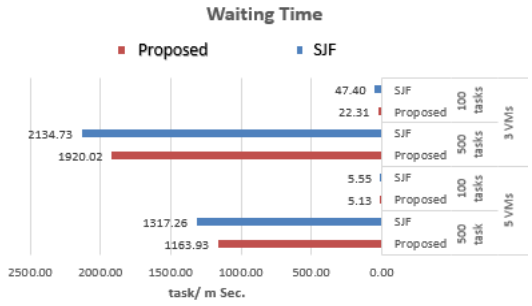

Fig. 7. Waiting Time (Sec).

We found that the value in the proposed algorithm, is about 40% less than that in the SJF. Table 1, illustrate those comparisons. The total waiting is improved with a range of 10–52%.. The turnaround time is improved with a range of 10–38%. On the other hand,The the finish-time is decreased by 33-52% with 25-100% increasing on the throughput.some of the performance metrics are decreased as shown in Table 1 and Table 4.

Table 1. Performance of Tasks with VMs

a- 100 Cloudlets Submitted to 3 VMs

Algorithm	T (Finish)	Throughput	Waiting	Turnaround
SJF	26.08	0.04	47.40	50.96
Proposed	12.26	0.08	22.31	31.40
Enhancement %	52.99	100.00	52.93	38.37

b- 500 Cloudlets Submitted to 3 VMs

Algorithm	T (Finish)	Throughput	Waiting	Turnaround
SJF	1425.21	0.04	2134.73	2162.92
Proposed	948.84	0.05	1920.02	1930.53
Enhancement %	33.42	25.00	10.06	10.74

c- 100 Cloudlets Submitted to 5 VMs

Algorithm	T (Finish)	Throughput	Waiting	Turnaround
SJF	1.90	0.53	5.55	6.19
Proposed	1.40	0.71	5.13	5.28
Enhancement %	26.09	33.96	7.64	14.71

d- 500 Cloudlets Submitted to 5 VMs

Algorithm	T (Finish)	Throughput	Waiting	Turnaround
SJF	905.00	0.06	1317.26	1345.50
Proposed	574.31	0.09	1163.93	1169.90
Enhancement %	36.54	50.00	11.64	13.05

Another scenario, increasing the VM to 5 and test the 100 and 500 tasks In Table 3 are considered and we obtained the results as illustrated in Table 1-(c,d) We found that the value of the proposed out performs total waiting and the turnaround time are also improved with a range of 14% in 100-task and 13% for 500-task. On the other hand,The the finish-time is decreased by 26-36% and 33-55% increasing on the throughput for 100 and 500 tasks respectively, Table 4 shows that.

When we consider more task request, 500 tasks, into groups according to the number of devices. The first VM finished its tasks because it had the shortest tasks from a task list. The tasks that had not yet been implemented were redistributed, and the distribution process was repeated until all the tasks were completed. When this process was compared to the results from the SJF algorithm, it showed better performance, as shown in Table 2.

Table 2. Parameters Test for 3 VMs with 100/500 Tasks

SJF	Sum		VM3		VM2		VM1	
No. of tasks	500	100	500	100	500	100	500	100
No of executed tasks	500	100	167	33	166	33	167	34
Throughput (task/ m Sec.)	0.04	0.04	0.0100	0.01	0.01	0.01	0.01	0.0127
Finish Time (Sec.)	14	3	14	3	9	0.77	4	0.18
Sum Waiting Time (Sec.)	2135	47	1115	37	724	8	296	2
Turnaround Time (Sec.)	2163	51	1129	40	734	9	301	3
Proposed algorithm	Sum	Sum	VM 3	VM 3	VM 2	VM 2	VM 1	VM 1
No of executed tasks	500	100	111	14	146	31	243	55
Throughput (task/ m Sec.)	0.05	0.08	0.0100	0.04	0.02	0.03	0.03	0.0114
Finish Time (Sec.)	9	1.23	9	1.15	9	1.23	9	1.18
Sum Waiting Time (Sec.)	1920	22	523	5	580	10	818	7
Turnaround Time (Sec.)	1931	31	0.17	11	589	5	819	15

Table 3. Parameters Test for 5 VMs with 100/500 Tasks

SJF	Sum		VM5		VM4		VM3		VM2		VM1	
No. of tasks	500	100	500	100	500	100	500	100	500	100	500	100
No of executed tasks	500	100	99	19	100	20	100	20	100	20	101	21
Throughput (task/mSec.)	0.06	0.53	0.011	0.100	0.011	0.11	0.01	0.11	0.01	0.11	0.01	0.11
Finish Time (Sec.)	9	0.19	9	0.19	7	0.17	6	0.12	4	0.09	2	0.06
Sum Waiting Time (Sec.)	1317	6	430	2	351	2	270	1.10	179	0.77	88	0.51
Turnaround Time (Sec.)	1346	6	439	2	358	2	276	1.23	183	0.86	90	0.57
Proposed algorithm	Sum	Sum	VM 5	VM 5	VM 4	VM 4	VM 3	VM 3	VM 2	VM 2	VM 1	VM 1
No of executed tasks	500	100	63	14	71	14	85	17	112	22	169	33
Throughput (task/mSec.)	0.09	0.71	0.011	0.100	0.012	0.10	0.02	0.12	0.02	0.16	0.03	0.24
Finish Time (Sec.)	6	0.1	5.7	0.1	5.5	0.1	5.6	0.1	5.5	0.1	5.8	0.1
Sum Waiting Time (Sec.)	1164	5.1	172.9	0.9	189.4	0.7	208.6	0.9	250.4	1.0	342.7	1.6
Turnaround Time (Sec.)	1170	5.3	172.9	0.9	189.4	0.7	208.6	0.9	255.9	1.1	343.1	1.7

5. Conclusion and Future Work

Resource management and tasks scheduling are necessary for satisfying Cloud Computing Environment Quality of Service CC-QoS requirements. Scheduling algorithms are play very important roles in governing proper data flows which are implemented to insure best resource utilization and to achieve best performance.

We provided a modified algorithm (modified SJF) for distributing the tasks on cloud VMs, with ascending task lengths and divided them at first round and redistributed the rest when one of the VM had finished its assigned tasks, even if there were some tasks still in processing stage on the other VMs. Table 4 is summarizing the performance of the experiment's results, which had been obtained by the following equation:

$$\text{Ratio} = 100 * \text{ABS}\ (1- (\text{MOD SJF}/\text{SJF})) \quad (3)$$

Table 4. Optimization Ratios

Enhancement %	Finish Time	Throughput	Waiting Time	Turnaround Time
3 VMs 100 Tasks	52.99	100.00	52.93	38.37
3 VMs 500 Tasks	33.42	25.00	10.06	10.74
5 VMs 100 Tasks	26.09	33.96	7.64	14.71
5 VMs 500 Tasks	36.54	50.00	11.64	13.05

We found that the use of the proposed method was efficient, as it increases the Throughput and reduces the Finish, Waiting, and Turnaround Times in all cases.

In the future, we want to increase the number of tasks and devices, improve the results, and determine the role of this method if applied to the scheduling tasks in a condensed environment.

References

[1] Wang, N., Liang, H., Jia, Y., Ge, S., Xue, Y., & Wang, Z. (2016). Cloud computing research in the IS

discipline: A citation/co-citation analysis. *Decision Support Systems, 86,* 35-47.

[2] Shawish, A., & Salama, M. (2014). Cloud computing: Paradigms and technologies. *Inter-Cooperative Collective Intelligence: Techniques and Applications,* 39-67. Springer Berlin Heidelberg.

[3] Reddy, D. C. (2013). An efficient profit-based job scheduling strategy for service providers in cloud computing systems. *International Journal of Application or Innovation in Engineering & Management (IJAIEM), 2(1).*

[4] Arora, S., & Anand, S. (2014). Improved task scheduling algorithm in cloud environment. *International Journal of Computer Application, 96(3).*

[5] Li, B., Song, A. M., & Song, J. (2012). A distributed QoS-constraint task scheduling scheme in cloud computing environment: Model and algorithm. *Advances in Information Sciences and Service Sciences*, *4(5),* 283-291.

[6] Wang, X., Wang, Y., & Cui, Y. (2014). A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing. *Future Generation Computer Systems, 36,* 91-101.

[7] Rahul, M. (2016). An efficient multi-objective genetic algorithm for optimization of task scheduling in cloud computing. *Asian Journal of Technology & Management Research.*

[8] Dubey, S., & Agrawal, S. (2013). QoS driven task scheduling in cloud computing. *International Journal of Computer Applications Technology and Research, 2(5).*

[9] Mathew, T., Sekaran, K. C., & Jose, J. (2014). Study and analysis of various task scheduling algorithms in the cloud computing environment. *Proceedings of International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 658-664).

[10] Lakshmi, R. D., & Srinivasu, N. (2016). A dynamic approach to task scheduling in cloud computing using genetic algorithm. *Journal of Theoretical and Applied Information Technology, 85(2).*

[11] Zhang, Y., & Xu, B. (2015). Task scheduling algorithm based on QoS constraints in cloud computing. *International Journal of Grid and Distributed Computing, 8(6),* 269-280.

[12] Banerjee, S., & Hecker, J. P. (2017). A multi-agent system approach to load-balancing and resource allocation for distributed computing. *Proceedings of First Complex Systems Digital Campus World E-Conference 2015* (pp. 41-54). Springer, Cham.

[13] Abdullah, M., & Othman, M. (2014). Simulated annealing approach to cost-based multi-quality of service job scheduling in cloud computing environment. *American Journal of Applied Sciences, 11(6),* 872.

[14] Lakshmi, R. D., & Srinivasu, N. (2016). A dynamic approach to task scheduling in cloud computing using genetic algorithm. *Journal of Theoretical and Applied Information Technology, 85(2).*

[15] Bishwkarma, M. K., & Vyas, K. (2016). Survey on round robin and shortest job first for cloud load balancing. *International Journal of Engineering Research and General Science, 4(1).*

[16] Mondal, R. K., Nandi, E., & Sarddar, D. (2015). Load balancing scheduling with shortest load. *International Journal of Grid and Distributed Computing*, *8(4),* 171-178.

Abdellatif I. Moustafa is an associate professor. He received his B.Sc. and M.Sc. in electrical engineering (communications & electronics) from AL Azhar University, Cairo, Egypt, in 1986 and 1992 respectively, and Ph.D from Stevens Inst. of Technology, Hoboken, NJ, USA in 2002. He worked as a consultant for lucent technologies, Holmdel, NJ, (1998-2001) and many other private networking companies. He worked an assistant professor at Faculty of Engineering, AL Azhar University, Cairo, Egypt (2002 – 2007) and an associate professor up till now. He joined College of Computers & Info. Sys., Umm AL Qura University,

Saudi Arabia for sabbatical years. His current interests include WSN, cloud computing, underwater sensors, networks modelling and simulations.

Reem Ganadily is a M.Sc. student at the Department of Computers & Info. Sys (CIS) of Umm AL Qura University (Saudi Arabia), where she received her B.Sc. degree, in Faculty of Computing and Information Technology (FCIT) from Umm AL Qura University University (UQU), Makkah, Saudi Arabia, in 2009. She is working as the trainer of computer sciences, multimedia and web design at Technical College, Technical and Vocational Training Corporation (TVTC), Jeddah, Saudi Arabia (2009– till now). Her current interests include cloud computing and simulations.

Shahad Alzahrani is a M.Sc. student at the Department of Computers & Info. Sys (CIS) of Umm AL Qura University (Saudi Arabia), where she received her B.Sc. degree, in Faculty of Computing and Information Technology (FCIT) from King Abdulaziz University (KAU), Jeddah, Saudi Arabia, in 2008. She is working as the trainer of computer sciences, multimedia and web design at Technical College, Technical and Vocational Training Corporation (TVTC), Jeddah, Saudi Arabia (2008 – till now). Her current interests include WSN, cloud computing and simulations.

Majid Alotaibi received the Ph.D. from The University of Queensland, Brisbane, Australia, in 2011. Currently, he is an assistant professor with the Department of Computer Engineering, Umm Al-Qura University, Makkah, Kingdom of Saudi Arabia. His current research interests include mobile computing, mobile and sensor networks, wireless technologies, Ad-hoc networks, computer networks (wired/wireless), RFID, antennas and propagation, radar, and nano electronics.