Research Journal of Applied Sciences, Engineering and Technology 14(3): 124-131, 2017
DOI:10.19026/rjaset.14.4155
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: December 14, 2016 Accepted: February 14, 2017 Published: March 15, 2017

**Research Article**

# Dynamic Job Scheduling Algorithms Based on Round Robin for Cloud Environment

[1]Bossy Mohamed, [2]Noha E. AL-Attar, [1]Wael Awad and [3]Fatma A. Omara
[1]Department of Mathematics and Computer Science, Faculty of Science,
Port Said University, Port Said, Egypt
[2]Faculty of Engineering, Delta University for Science and Technology, Dakahlia, Egypt
[3]Department of Computer Science, Faculty of Computers and Information, Cairo University, Cairo, Egypt

**Abstract:** This study attempts to solve the problem of the static scheduling algorithms by developing a dynamic version of Round Robin scheduling algorithm; Dynamic Priority Round Robin and Enhanced Dynamic Priority Round Robin. The proposed algorithms have been developed based on a dynamic manner of choosing the quantum time according to the current status of the requested jobs in attempting to fulfill the user's requirements and improve the overall system performance and resource utilization. The implementation of the developed algorithms is done by the Cloudsim simulator. The results record that the two versions of dynamic scheduling algorithms achieve high performance and resource utilization for the Cloud system comparing with the static scheduling algorithms like Round Robin and others. Accordingly, they decrease the idle waiting, computational and turnaround time of the requested jobs. By comparing the proposed algorithms with their corresponding static Round Robin versions, it is found that; Dynamic Priority Round Robin (DPRR) algorithm has enhanced the saving in idle waiting time, the response time and turnaround time are by 25, 51 and 32%, respectively. Similarly, the idle waiting time, response time and turnaround time are decreased in the proposed Enhance Dynamic Priority Round Robin (EDPRR) algorithm by 51, 44 and 30%, respectively. Furthermore, the resource utilization has also improved by 18% and 5% for the both of developed algorithms (DPRR and EDPRR) respectively.

**Keywords:** Cloud computing, dynamic scheduling, round robin, static scheduling

## INTRODUCTION

Over the last few years, there is a persistent increase in the size and specifications of data used by different users. This usage increasing led to the appearance of what is called big data (Hashem *et al*., 2015). Usually, the big data is characterized by five main features:

- Volume; which represents the size of the data that need to be processed
- Velocity; which refers to the rate of the data growth and usage
- Variety; which means that there are different types and formats of the data used in processing,
- Veracity; which means that both data results and analysis have to be accurate
- Value; which is the added value and contribution offered by data processing (Chen *et al*., 2012).

Due to the spread use of the big data in many fields (e.g., healthcare, science, engineering, finance, business and different society problems), it is a big challenge for data users and providers to handle this growth rate by the traditional computing infrastructure. Thus, Cloud Computing may be one of the best solutions to face this challenge (Hashem *et al*., 2015; Janakiraman, 2016).

Now a day, Cloud Computing gains popularity in processing, managing and analyzing the big data. In other words, it is considered as a way to get enough storage and computing power to handle big data without purchasing infrastructure assets (Chang and Wills, 2016). Due to the pay-as-you-go fashion in Cloud Computing, the user can ask for the resources only as he needs and the cloud providers will provision the required resources dynamically and process large data sets in a parallel way (Katyal and Mishra, 2013).

Generally, when a user requests a service from the Cloud environment, the cloud provider treats this service as a job to be executed. In order to execute the required job, many processes need to be carried out in order to deliver the requested service as; job scheduling, resources provisioning and resource allocation. Simply, the master job controller in the

**Corresponding Author:** Bossy Mohamed, Department of Mathematics and Computer Science, Faculty of Science, Port Said University, Port Said, Egypt

Cloud manages the order of executing the existed jobs according to their priority, then the provisioning and allocation processes begin to allocate the suitable resources to the selected jobs (Vernier, 2016).

The job scheduling is a decision making process which manages the order of executing the user's jobs using a set of strategies. The main goal of the job scheduling process is to increase the resource utilization and minimize processing time of the jobs, which in turn increases the system throughput (Salot, 2013; Singh *et al*., 2014).

Generally, job scheduling algorithms can be divided into two main types; Preemptive and Non-Preemptive algorithms. In *Preemptive algorithms*; the current running jobs are temporarily suspended according to a specific condition. These jobs wait to be re-executed after a specific predefined time called "time quantum". This type of algorithms focuses on decreasing the response time for the required jobs. Round Robin (RR) is one of the significant examples of the preemptive algorithms (Singh *et al*., 2014).

In *Non-Preemptive algorithms*; once a job is selected to start its execution, it continues in running until it finishes completely. Non-Preemptive algorithms try to decrease the idle waiting time for the user's jobs. First-Come-First-Served (FCFS) and Shortest Job First (SJF) are examples of the non-preempt algorithms (Katyal and Mishra, 2013). First-Come-First-Served is considered the simplest and the default scheduling algorithm. The idea of FCFS is that it does not determine any priorities or conditions to execute the jobs; just the first arrived job will be the first finished job without any interruption. This means that the short jobs may wait for a long time before starting its execution (Chen *et al*., 2013) Shortest Job First (SJF) algorithm tries to solve the problem of executing the short jobs. In SJF algorithm, the jobs with small processing time will have the highest priority to be executed, so they do not have to wait the early arrived large jobs to finish their execution (Siahaan, 2016).

In Cloud Computing, the Master Virtual Machine (MVM) controller is responsible for scheduling the incoming jobs and sending them to Slaves Virtual Machines (SVMs) depending on the chosen scheduling algorithm (Chen *et al*., 2013). Due to the existence of multiple independent users who usually are more than the Cloud available resources, the job scheduling process becomes one of the main challenges that face the Cloud Computing community (Ernemann *et al*., 2002) Choosing the suitable scheduling algorithm depends on six main parameters; Resource Utilization; the ratio between the total busy time of the virtual machine and the total execution time of the parallel application, Throughput; the number of completed jobs within a certain period of time, System Efficiency; the busy time of each virtual machine, Response time; the time taken to start responding to the job request), Idle Waiting time; the waiting time taken by each job to complete its execution and Turnaround time; the total amount of time taken by the job) (Samal and Mishra, 2013).

One of the most famous scheduling algorithms in Cloud Computing is Round Robin (RR) algorithm because it is suitable for job scheduling in time-sharing systems due to its simplicity, fairness and generality (Siahaan, 2016).

In Round Robin, jobs are queued with the same style in FCFS algorithm, but RR assigns a fixed equal quantity of execution time to each job which is known as Time Quantum (TQ). TQ is a fixed interval of time used to interrupt the long jobs to give a chance to other jobs to start their execution (Pinedo, 2012). After the end of TQ, the uncompleted jobs will be stopped and traced to the tail of the job queue (Salot, 2013)

RR algorithms are categorized into two types; Static RR and Dynamic RR (Aravind and Chelladurai, 2016). In Static RR, a fixed Time Quantum is set. It starts from the beginning of the jobs execution time till they are finished. Static RR ignores any information about deadline time, computation time, precedence constraints and any expected delay time (Stankovic *et al*., 1998). Thus, Static RR may suffer from many drawbacks due to the fixed choice of TQ as it may affect the performance of the system and resources' utilization (e.g., If TQ is large, small jobs may finish their execution after a small amount of time and unutilized resources cannot be used until the end of TQ. On the other hand, if TQ is small, the big jobs will be surely interrupted after the end of the TQ and trace to the tail of the job queue waiting for their new turn, this may decrease the system efficiency) (Samal and Mishra, 2013).

Dynamic RR is basically based on calculating TQ according to the real processing time of the current jobs, which means that the TQ will be dynamically changed every turn (Noon *et al*., 2011). Dynamic RR tries to enhance the static RR algorithm by increasing the performance and stability of the system and provides a self-adaptation system in which the system adapts itself according to the incoming jobs (Matarneh, 2009).

In this study, two enhanced algorithms for dynamic Round Robin algorithm have been proposed in order to overcome RR drawbacks in the Cloud Computing environment.

## LITERATURE REVIEW

Many researches have been conducted in the field of Cloud Computing scheduling algorithms. Some of the researchers handled only the static scheduling algorithm as Aravind and Chelladurai (2016). They have focused on addressing scheduling fairness between CPU-bound jobs and I/O-bound jobs by developing a Fair-Share Round Robin Scheduling Algorithm (FSRR). FSRR algorithm is considered an

enhanced version of the classical round robin to decrease the unfairness problem in classical RR.

Other researchers tried to enhance the job scheduling specially in Cloud Computing as Li (2016) who has developed the Priority Based Weight Round Robin (PBWRR) algorithm. According to PBWRR algorithm, a weight for each job has been suggested according to its presence in the waiting queue in order to avoid long waiting time and minimize the response time.

Liu *et al*. (2013) has presented a Generalized Priority Algorithm for efficient execution of jobs. This algorithm categorized each node's computing capacity into two types; foreground virtual machines (e.g., high CPU priority) and background virtual machines (e.g., low CPU priority). This algorithm aims to increase resources utilization in the system.

Another enhancement for RR algorithm has been introduced by Mohamed *et al*. (2016) They have introduced an Enhanced Priority Round Robin (EPRR) which is based on listing the uncompleted jobs in a priority queue depending on the remaining execution time (i.e., the job with small remaining time will have high priority). The advantage of this algorithm is to decrease the average waiting and turnaround time of the job.

Although, using the static job scheduling algorithms is the most popular in cloud computing, it cannot handle the rapid increase in the complicated applications that are handled by the Cloud environment. Thus, there is a significant need to use efficient dynamic scheduling algorithms to manage and control the available resources, as well as, increase the system throughput.

Lee *et al*. (2011) have proposed a Dynamic Priority Scheduling Algorithm (DPSA) which can change the priorities of the available jobs during the scheduling time. According to DPSA algorithm, the scheduler sets a threshold value that limits the waiting time for jobs in the job queue, if the job exceeds this threshold then it will be sent to execute immediately.

Matarneh (2009) has presented a Self-Adjustment Time Quantum in Round Robin (SARR) algorithm. The SARR algorithm depends on Burst Time of the now running processes. The idea of this algorithm is to adjust the Time Quantum according to the burst time of the running processing. It tries to minimize idle waiting time, but it still needs better adjustment criteria for Time Quantum.

## MATERIALS AND METHODS

The goal of this study is to enhance the Round Robin algorithm in such a way that it can deal with the dynamic nature of job scheduling. The original form of the RR algorithm is to use a static time quantum to execute the job within it using the preemption style. Determining this fixed time quantum is considered the main drawback in the RR algorithm. So, the work in this study will try to handle this problem by developing two versions of the Round Robin algorithm using dynamic quantum time. The two algorithms which have been introduced are "*Dynamic Priority Round Robin*" and "*Enhanced Dynamic Priority Round Robin*".

**Dynamic Priority Round-Robin (DPRR) algorithm:** Dynamic Priority Round Robin (DPRR) is based mainly on the dynamic change of the time quantum. At the beginning of the scheduling process, the time quantum is calculated according to the expected processing time of the requested jobs and then it is modified in every time slice depending on the processing time of the existed jobs. The Time Quantum (TQ) calculation is defined in Eq. (1):

$$TQ=AVG(\Sigma PT(J_N)/N \quad (1)$$

where,

$PT(J_N)$ = The processing time of every job in the system

$N$ = The number of requested jobs

DPRR uses two types of queues; Primary Job Queue (*PJQ*) which stores the incoming jobs which need to be processed in the Cloud system and Priority Queue (*PQ*) which is used to list the uncompleted jobs that will be executed depending on their priority which is determined according to the condition in Eq. (2):

$$T_{Remain}(j) <= TQ/2 \quad (2)$$

where, $T_{Remain}(j)$ is the remaining time of the uncompleted job (*j*).

Thus, DPRR algorithm is based on executing the uncompleted jobs firstly depending on their remaining time. According to Eq. (2), if the $T_{Remain}$ of the uncompleted job is less than or equal half of the current Time Quantum, then these jobs will be listed in the *PQ*, otherwise, it will be queued in the tail of the *PJQ*.

Also, DPRR algorithm cares about achieving good resource utilization during the run time. It determines any virtual machines that become free before the end of the Time Quantum. If a free VM is existed, the scheduler will assign a new job from the head of *PJQ* to be executed on the free VM.

The pseudo code of DPRR algorithm is illustrated in Fig. 1. DPRR algorithm assumes that, *m* is the number of the available Virtual Machines (*VMs*) and *N* is the total number of the requested jobs over the Cloud system. While *TQ* represents the calculated Time Quantum and *PT* is the expected Processing Time of the requested jobs.

Unfortunately, DPRR does not concern any criteria for selecting the jobs during the runtime to be run. The

```
1.  Begin w
2.  Arrived Jobs store in PJQ/* Store as FCFS*/
3.  Initialize PQ by NULL/* PQ initializer */
4.  S, R = 0/* R is No. of waiting jobs in PQ, S is No. of processed jobs on the available VM*/
5.  While (PJQ not equal NULL)/* PJQ has jobs */
6.  Do
7.  TQ=AVG(Σ PT(JN)/N/* Calculating Time Quantum */
8.  For each job J to VMm
9.  Issue_Job onto VMi to execute/* Execute jobs */
10. S = S+1/*Count No. of executing jobs (S) */
11. End For
12. For each processed job i to VMm/* locate the jobs on the free VM*/
13. If Tremain (Ji) is equal 0/* If a job ends before the end of TQ */
14. Set VMi to null/* free up the VM that has finished job*/
15. Issue_Job from PJQ onto VMi/*Put new jobs from PJQ on the free VM*/
16. S = S+1
17. End If
18. End For
19. Until(TQ equal 0)/* Until the Time Quantum finishes */
20.  For each processed job i to S/* Selected jobs *
21.  If TRemain(j) not equal 0 and less than or equal TQ/2
/* if the processed Job doesn't finish and its remaining time is less than TQ average */
22.  Add i to PQ/* Add job I to Priority queue */
23.  Else
24.  Add i to PJQ/* Add job i to the tail of Primary job queue */
25.  End If
26.  End For
27.  While (PQ is not NULL)
28.  For each Job i in PQ
29.  Sort jobs in PQ in ascending order according to T (i)
30.  R = Count (i)/* Count uncompleted processed jobs in PQ*/
31.  While (R is greater than VMm)
32.  TQPQ= AVG(Σ T(iR)/R)/* Calculate Time Quantum for jobs in PQ)
33.  Issue_Job from PQ onto VMi to execute
34.  If Tremain (i) is equal 0/* If a job ends before the end of TQ */
35.  Set VMi to null/* free up the VM that has finished job*/
36.  Go to 33/*Put new jobs from PQ on the free VM*/
37.  End If
38.  S = S+1
39.  End While
40.  For each job i in PQ to VMR
41.  Issue_Job from PQ onto VMi to execute
42.  End for
43.  VMfree = VMm−VMR/* calculate No. of free VMs */
44.  TQnew= AVG(Σ T(iR)/R)+ Σ T(jN)/N)
45.  Goto 8
46.  End For
47.  End While
48.  End While
49.  End
```

Fig. 1: Pseudo code of DPRR algorithm

scheduler in DPRR only checks for the free VMs and then selects the job from the head of *PJQ* without taking into consideration if the selected job has a proper processing time to be executed on this resource or not, which may lead to decrease the performance of the system.

**Enhanced Dynamic Priority Round-Robin (EDPRR) Algorithm:** Enhanced Dynamic Priority Round Robin (EDPRR) algorithm is considered an enhancement of DPRR algorithm. It is developed to overcome the main drawback (i.e., the scheduler selects the new job to assign in to free *VM* from the head of the *PJQ* without considering any criteria for choosing the suitable job). In EDPRR algorithm, after checking if there is an existed free virtual machine during the runtime and before the end of *TQ*, the scheduler selects a new job from *PJQ* to be executed but under a condition. This

```
12.  For each processed job i to VM_m
13.  If T_remain (i) is equal 0
14.  Set VM_i to null
15.  T_free(VM) = TQ- PT(i)
16.  For each job j in PJQ to VM_i
17.  If PT(j) is less than or equal T_free(VM)
18.  Issue_Job j from PJQ onto VM_i
19.  S = S+1
20.  Else
21.  Issue_Job from the head of PJQ onto VM_i
22.  S=S+1
23.  End If
24.  End For
25.  End if
26.  End for
```

Fig. 2: Pseudo code of the enhanced part in the proposed EDPRR algorithm

Table 1: A sample of ten requested jobs' specifications

| Job name | Required capacity |
|---|---|
| J1 | 200 |
| J2 | 60 |
| J3 | 80 |
| J4 | 20 |
| J5 | 100 |
| J6 | 70 |
| J7 | 50 |
| J8 | 30 |
| J9 | 150 |
| J10 | 40 |

condition concerns with selecting the suitable job that can be executed on the free resource (i.e., VM) during the remaining time of the *TQ*. Calculating the time of the free virtual machine (VM) is defined in Eq. (3):

$$T_{free}(VM) = TQ - PT_{finished\_job} \quad (3)$$

The pseudo code of the enhanced part in the proposed EDPRR algorithm is described in Fig. 2.

## RESULTS AND DISCUSSION

In order to implement and evaluate the performance of the proposed scheduling algorithms (i.e., DPRR and EDPRR), CloudSim simulator is used (Buyya *et al*., 2009). It is a simple toolkit that is used to simulate the cloud environment. It gives the ability of a quick and easy change of the needed parameters to be assumed in the simulation that provides a high degree of configurability and flexibility (El-Attar *et al*., 2014).

The experimental study considers some assumptions as follows:

- It is operated on a single Data Center with 5, 8 and 10 homogenous VMs. Each VM has the same type of operating system and the same amount of processing and capacity (e.g., '1GHz' processor and '2GB' of RAM).
- The experiment is conducted with a varying number of jobs '100', '150' and '200'.
- Each job has 2 parameters; Name and Required Capacity (Table 1). Job's computation time will be calculated using job's required processing capacity which varies from one job to another.

According to the results of implementing the proposed algorithms (DPPR and EDPPR), the performance can be evaluated through some of the essential parameters that measure the efficiency of the scheduling process using the proposed algorithms. The evaluation parameters are; the response time, idle waiting time, turnaround time and utilization performance. In this evaluation, the comparison has been done among the two proposed algorithms (DPRR and EDPRR) as dynamic scheduling algorithms from a side and a static scheduling algorithm EPRR (i.e., it depends only on listing the uncompleted jobs in a priority queue with a static amount of time quantum TQ) which have been presented before in Matarneh (2009) on the other side. The comparative results of the three algorithms based on the four evaluation parameters (i.e., Idle Waiting time, Response Time, Turnaround Time and Resource Utilization) are presented in Table 2 to 5 respectively. Also, the enhancement percentages of the proposed algorithms are calculated according to Eq. (4):

$$Improving\ percentage = \frac{P1ij - P2ij}{P1ij}\ \% \quad (4)$$

where, *P1* is the default scheduling algorithm and *P2* represents the enhanced algorithm to be compared. While *i, j* are the jobs and the VMs respectively.

Resource utilization has been evaluated using Eq. (5) which defines the utilization function as the ratio between the total busy time of the virtual machines and the total finish execution time of the parallel application (Abdelkader and Omara, 2012):

$$Utilization = \frac{VM\ Busy\ Time}{Make\ Span} * 100 \quad (5)$$

As shown in Table 2 to 5, the performance evaluation has been done over two phases. First, the static scheduling (EPRR) algorithm is compared with the first proposed dynamic (DPRR) algorithm. According to this phase, it is found that the performance of the proposed DPRR outperforms the static EPRR algorithm by 79% and 65% for both response and turnaround time respectively. Furthermore, the proposed DPRR algorithm utilizes the resources that are freed up during the run time by selecting a new job from the PJQ. So, the idle waiting time is decreased in the proposed DPRR algorithm by 75% with respect to the static EPRR algorithm. Also, the resource utilization is enhanced by 17%.

On the other hand, for the dynamic algorithms, the second proposed EDPRR algorithm has achieved a

Table 2: The idle waiting time comparison between the static (EPRR) and dynamic (EDPRR and DPRR) scheduling algorithms

| No. of VMs | No. of Jobs | EPRR /sec | DPRR /sec | EDPRR /sec | Improve DPRR Vs. EPRR (%) | Improve EDPRR vs. DPRR (%) |
|---|---|---|---|---|---|---|
| 5 | 100 | 699 | 300 | 280 | 57 | 6 |
| | 150 | 1110 | 583 | 530 | 47 | 8 |
| | 200 | 2963 | 730 | 695 | 75 | 5 |
| 8 | 100 | 400 | 250 | 235 | 38 | 6 |
| | 150 | 620 | 500 | 470 | 20 | 6 |
| | 200 | 799 | 680 | 647 | 15 | 5 |
| 10 | 100 | 300 | 238 | 211 | 20 | 11 |
| | 150 | 515 | 400 | 380 | 22 | 5 |
| | 200 | 730 | 550 | 533 | 25 | 3 |

Table 3: The response time comparison between the static (EPRR) and dynamic (EDPRR and DPRR) scheduling algorithms

| No. of VMs | No. of Jobs | EPRR /Sec | DPRR /Sec | EDPRR /Sec | Improve DPRR Vs. EPRR (%) | Improve EDPRR Vs. DPRR (%) |
|---|---|---|---|---|---|---|
| 5 | 100 | 5632 | 1420 | 1320 | 75 | 7 |
| | 150 | 7985 | 1756 | 1542 | 78 | 12 |
| | 200 | 11230 | 2314 | 1996 | 79 | 13 |
| 8 | 100 | 3263 | 1100 | 1025 | 66 | 7 |
| | 150 | 4896 | 1270 | 989 | 74 | 12 |
| | 200 | 6120 | 1267 | 1170 | 79 | 8 |
| 10 | 100 | 2750 | 988 | 950 | 62 | 4 |
| | 150 | 4632 | 1023 | 965 | 77 | 6 |
| | 200 | 6000 | 1136 | 1110 | 78 | 4 |

Table 4: The turnaround time comparison between the static (EPRR) and dynamic (EDPRR and DPRR) scheduling algorithms

| No. of VMs | No. of Jobs | EPRR/Sec | DPRR /Sec | EDPRR /Sec | Improve DPRR Vs. EPRR (%) | Improve EDPRR Vs. DPRR (%) |
|---|---|---|---|---|---|---|
| 5 | 100 | 6978 | 2400 | 2210 | 65 | 8 |
| | 150 | 9786 | 2839 | 2546 | 71 | 10 |
| | 200 | 12740 | 4414 | 3568 | 65 | 19 |
| 8 | 100 | 4578 | 2088 | 1997 | 54 | 4 |
| | 150 | 5640 | 2640 | 2403 | 53 | 9 |
| | 200 | 7250 | 3136 | 2989 | 57 | 5 |
| 10 | 100 | 3500 | 1967 | 1845 | 43 | 6 |
| | 150 | 5100 | 2635 | 2330 | 48 | 12 |
| | 200 | 6436 | 3022 | 2478 | 53 | 18 |

Table 5: The resource utilization comparison between the static (EPRR) and dynamic (EDPRR and DPRR) scheduling algorithms

| No. of VMs | No. of Jobs | EPRR (%) | DPRR (%) | EDPRR (%) | Improve DPRR Vs. EPRR (%) | Improve EDPRR Vs. DPRR (%) |
|---|---|---|---|---|---|---|
| 5 | 100 | 67 | 80 | 83 | 16 | 4 |
| | 150 | 68 | 83 | 88 | 18 | 6 |
| | 200 | 68 | 82 | 89 | 17 | 8 |
| 8 | 100 | 68 | 86 | 88 | 21 | 2 |
| | 150 | 69 | 87 | 89 | 21 | 2 |
| | 200 | 69 | 85 | 89 | 18 | 4 |
| 10 | 100 | 70 | 88 | 90 | 20 | 2 |
| | 150 | 70 | 85 | 88 | 18 | 3 |
| | 200 | 71 | 86 | 90 | 17 | 4 |

better performance than that the first proposed (DPRR) algorithm. The overall performance of the system after using EDPRR algorithm is improved by decreasing the idle waiting time, the response time and the turnaround time of jobs with 5, 13 and 19%, respectively with respect to the proposed DPRR algorithm. Also, EDPRR algorithm improves the resources utilization by 8% with respect to DPRR algorithm.

The performance evaluation also handles the effect of changing the number of the available virtual machines. The proposed DPRR and EDPRR have been implemented using 5, 8 and 10 VMs, with the same amount of requested jobs (i.e., 100, 150 and 200). According to the implementation results, it is found that the increasing of the number of virtual machine enhances the scheduling process, where by increasing the number of virtual machine to 8 instead of 5, the idle waiting time, the response time and turnaround time are decreased in the proposed DPRR algorithm by 25, 51 and 32%, respectively. Similarly, the response time and turnaround time are decreased in the proposed EDPRR algorithm by 51, 44 and 30%, respectively. Furthermore, the resource utilization has also improved by 18% and 5% for DPRR and EDPRR algorithms respectively (Fig. 3 to 6).

Finally, as it is shown in the above comparison, the dynamic scheduling algorithms have enhanced the performance of the system better than the static ones.

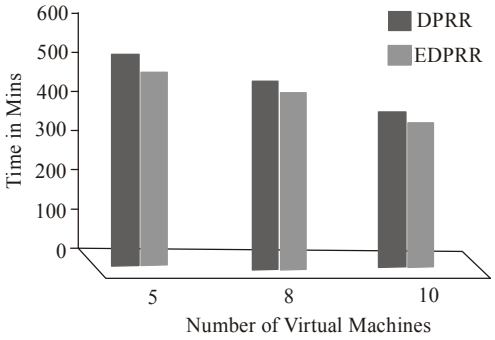


Fig. 3: Idle waiting time comparison

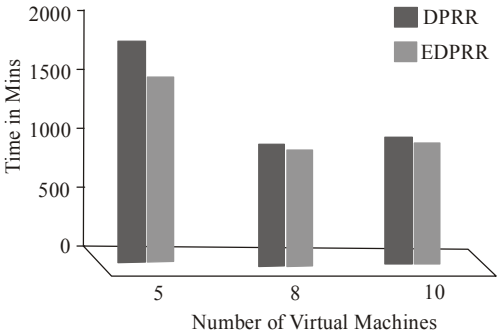


Fig. 4: Response time comparison

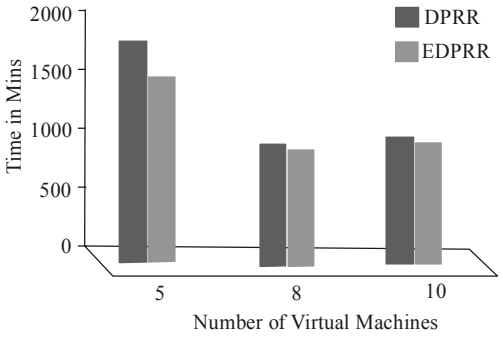


Fig. 5: Turnaround time comparison

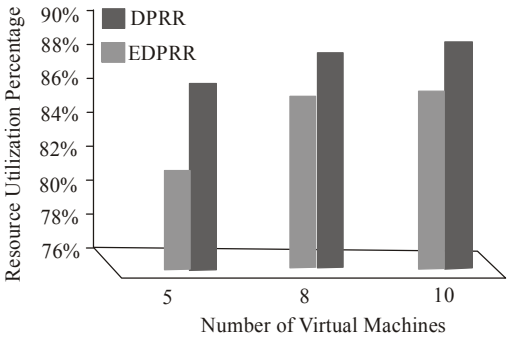


Fig. 6: Resource utilization comparison

The performance comparison between the static EPRR, DPRR and EDPRR algorithms is plotted in Fig. 7.

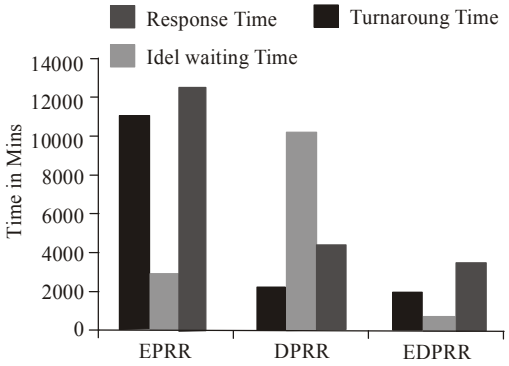


Fig. 7: Dynamic and static algorithms comparison

## CONCLUSION AND FUTURE RECOMMENDATION

In this study, two dynamic enhanced algorithms of Round Robin (Dynamic Priority Round Robin (DPRR) and Enhanced Dynamic Priority Round Robin (EDPRR)) have been proposed and developed to overcome the problem of fixed TQ. The proposed DPRR algorithm is based on calculating a dynamic quantum time according to the requested jobs processing time. Also, it utilizes the free resource by assigning a new job to it until the end of the time quantum. The main drawback of the proposed DPRR algorithm is that it does not concern any criteria to select jobs to be executed on the free resource. Thus, the second version of dynamic scheduling algorithms which is called EDPRR has been introduced. According to the proposed EDPRR algorithm, the selection of the new job to be executed on the free resource is based on its execution time to be suitable for execution upon the free resource. According to the implementation results of the proposed DPRR and EDPRR algorithms, it can be concluded that:

- The proposed DPRR and EDPRR algorithms outperform the static scheduling (EPRR) algorithm for both response and turnaround time respectively.
- Furthermore, the idle waiting time is decreased in the proposed DPRR algorithm with respect to the static EPRR algorithm. Also, the resource utilization is enhanced.
- In addition, the proposed EDPRR algorithm enhances the average of idle waiting time, response time, turnaround time and System Utilization relative to the proposed DPRR algorithm.

More investigation is needed to overcome the time overhead due to the selection of new jobs to be executed on the free VM(s). This will be considered in the future work.

## REFERENCES


Abdelkader, D.M. and F. Omara, 2012. Dynamic task scheduling algorithm with load balancing for heterogeneous computing system. Egypt. Inform. J., 13(2): 135-145.

Aravind, A. and J. Chelladurai, 2016. Fair-share Round Robin CPU Scheuling Algorithms. Retrieved from: http://web.unbc.ca/~csalex/papers/aj05.pdf. (Last Accessed on: July 19-Aug. 18, 2016 at 12 AM)

Buyya, R., R. Ranjan and R.N. Calheiros, 2009. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. Proceeding of the International Conference on High Performance Computing and Simulation (HPCS, 2009), pp: 1-11.

Chang, V. and G. Wills, 2016. A model to compare cloud and non-cloud storage of Big Data. Future Gener. Comp. Sy., 57: 56-76.

Chen, H., R.H.L. Chiang and V.C. Storey, 2012. Business intelligence and analytics: From big data to big impact. MIS Quart., 36(4): 1165-1188.

Chen, J., D. Wang and W. Zhao, 2013. A task scheduling algorithm for hadoop platform. J. Comput., 8(4): 929-936.

El-Attar, N., W. Awad and F. Omara, 2014. RPOAWLB: Resource provisioning optimization approach based on RPOA with load balance. Int. J. Comput. Appl., 105(7): 34-41.

Ernemann, C., V. Hamscher, U. Schwiegelshohn, R. Yahyapour and A. Streit, 2002. On advantages of grid computing for parallel job scheduling. Proceeding of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, China.

Hashem, I.A.T., I. Yaqoob, N.B. Anuar, S. Mokhtar, A. Gani and S.U. Khan, 2015. The rise of "big data" on cloud computing: Review and open research issues. Inform. Syst., 47: 98-115.

Janakiraman, P., 2016. Big Data Cloud Database and Computing. Retrieved from: https://www.qubole.com/resources/article/big-data-cloud-database-computing/.

Katyal, M. and A. Mishra, 2013. A comparative study of load balancing algorithms in cloud computing environment. Int. J. Distrib. Cloud Comput., 1(2): 14.

Lee, Z., Y. Wang and W. Zhou, 2011. A dynamic priority scheduling algorithm on service request scheduling in cloud computing. Proceeding of the International Conference on Electronic and Mechanical Engineering and Information Technology, China.

Li, H., 2016. PWBRR Algorithm of Hadoop Platform. Retrieved from: https://www.thelibrarybook.net/pdf-research-on-job-schedulingalgorithm-in-hadoop.html. (Last Accessed on: July 17, 2016 at 4 AM)

Liu, X., C. Wang, B.B. Zhou, J. Chen, T. Yang and A.Y. Zomaya, 2013. Priority-based consolidation of parallel workloads in the cloud. IEEE T. Parall. Distrib. Syst., 24(9): 1874-1883.

Matarneh, R.J., 2009. Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes. Am. J. Appl. Sci., 6(10): 1831-1837.

Mohamed, B., W. Awad, S.A. El Hafeez and F. Omara, 2016. Job schedulers based on round robin strategy on the cloud environment. Eur. J. Sci. Res., 141(2): 141-153.

Noon, A., A. Kalakech and S. Kadry, 2011. A new round robin based scheduling algorithm for operating systems: Dynamic quantum using the mean average. Int. J. Comput. Sci. Issue., 8(3): 224-229.

Pinedo, M., 2012. Scheduling: Theory, Algorithms and Systems. 4th Edn., Springer, New York, London.

Salot, P., 2013. A survey of various scheduling algorithm in cloud computing environment. Int. J. Res. Eng. Technol., 2(2): 131-135.

Samal, P. and P. Mishra, 2013. Analysis of variants in round robin algorithms for load balancing in cloud computing. Int. J. Comput. Sci. Inform. Technol., 4(3): 416-419.

Siahaan, A.P.U., 2016. Comparison analysis of CPU scheduling: FCFS, SJF and round robin. Int. J. Eng. Develop. Res., 4(3): 124-131.

Singh, P., V. Singh and A. Pandey, 2014. Analysis and comparison of CPU scheduling algorithms. Int. J. Emerg. Technol. Adv. Eng., 4(1): 91-95.

Stankovic, J.A., M. Spuri, K. Ramamritham and G. Buttazzo, 1998. Deadline Scheduling for Real-Time Systems. 1st Edn., In: EDF and Related Algorithms. The Springer International Series in Engineering and Computer Science. Springer, US, 460: 273.

Vernier, D., 2016. How Does Cloud Computing Work? Retrieved from: http://www.thoughtsoncloud.com/2014/02/how-does-cloud-computing-work/.