Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique

Samir Elmougy[1], Shahenda Sarhan[1*] and Manar Joundy[1,2]

Abstract

Cloud computing is a ubiquitous network access model to a shared pool of configurable computing resources where available resources must be checked and scheduled using an efficient task scheduler to be assigned to clients. Most of the existing task schedulers, did not achieve the required standards and requirements as some of them only concentrated on waiting time or response time reduction or even both neglecting the starved processes at all. In this paper, we propose a novel hybrid task scheduling algorithm named (SRDQ) combining Shortest-Job-First (SJF) and Round Robin (RR) schedulers considering a dynamic variable task quantum. The proposed algorithms mainly relies on two basic keys the first having a dynamic task quantum to balance waiting time between short and long tasks while the second involves splitting the ready queue into two sub-queues, Q1 for the short tasks and the other for the long ones.

Assigning tasks to resources from Q_1 or Q_2 are done mutually two tasks from Q_1 and one task from Q_2. For evaluation purpose, three different datasets were utilized during the algorithm simulation conducted using CloudSim environment toolkit 3.0.3 against three different scheduling algorithms SJF, RR and Time Slice Priority Based RR (TSPBRR) Experimentations results and tests indicated the superiority of the proposed algorithm over the state of art in reducing waiting time, response time and partially the starvation of long tasks.

Keywords: Task scheduling, Shortest job first, Round Robin, Dynamic quantum, Starvation

Introduction

The appearance of cloud computing systems represent a revolution in modern information technology (IT) that needs to have an efficient and powerful architecture to be applied in different systems that require complex computing and big-scale. Cloud is a platform that can support elastic applications in order to manage limited virtual machines and computing servers to application services at a given instance of time. The cloud is a suitable environment of multi-tenant computing which allows the users to share resources. In cloud, available resources must be checked and scheduled using an efficient task scheduler to be assigned to clients based on their requests [1–3].

Having an efficient task scheduler became an urgent need with the rapid growth of modern computer systems aiming to reach and achieve the optimal performance. Task scheduling algorithms are responsible for mapping jobs submitted to cloud environment onto available resources in such a way that the total response time and latency are minimized and the throughput and utilization of resources are maximized [3, 4]. Conventional task scheduling algorithms as Shortest-Job-First (SJF) [5], Round Robin (RR) [6], and First-Come-First-Serve (FCFS) [7], Multilevel queue scheduling (MQ) [8], Max-Min [9] and Min-Min [10] had achieved breathtaking results over years in different computer systems types but always suffer from big dilemmas as higher waiting time in RR and FCFS and starvation in SJF and Max-Min.

Starvation problem is one of the major challenges that face task scheduling in cloud where, a task may wait for one or more of its requested resources for a very long time. Starvation is frequently brought on by lapses in a

* Correspondence: shahenda_sarhan@yahoo.com
[1]Department of Computer Science, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt
Full list of author information is available at the end of the article

scheduling calculation, by resource spills, and can be deliberately created by means of a refusal of-administration assault. For example, if an ineffectively planned multi-tasking framework dependably switches between the initial two tasks while a third never gets the chance to run, then the third task is starving [11]. Many hybrids were introduced to solve the starvation problem as FCFS&RR and SRTF&RR [12] (note that RR is always a common denominator in these hybrids) but, no one till now solves or nearly approach to solve it. A hybrid of SJF and RR is one of the most used and powerful hybrids for solving starvation where we can benefit from SJF performance in reducing the turnaround time and from RR in reducing task waiting time. But the task quantum value was always the obstacle in having the optimum hybrid. Different researches were proceeded to find the best methodology to calculate the task quantum value as having small quantum leads to reducing throughput and increasing response time while having long quantum caused a high increase in turnaround time.

From this point and confessing the importance of starvation problem in task scheduling, we introduce in this paper a novel hybrid scheduling technique of SJF and RR with Dynamic Quantum, we called SRDQ applied through having two Queues to schedule processes for execution. The proposed algorithm is designed to be a unit based algorithm based on effectively queuing data structure and optimizing the execution time as possible. In this proposed technique, a time quantum value is statically and dynamically determined towards detecting the impact of quantum dynamicity over starvation and response time reduction which will be described in details in Section 3.3.

The remainder of this paper is organized as 5 main sections where, in Section 2, the concepts of task scheduling and some related work are presented. In Section 3.3, the proposed technique is presented enhanced with examples. While, the simulations settings are analyzed in Section 4. Section 5 comprises the discussion and results and finally, our main conclusions and future work are discussed in Section 6.

Related work

Task scheduling algorithms vary in their technique in scheduling tasks among cloud nodes statically, dynamically, in batches or even online, eventually they are all trying to achieve the optimal distribution of tasks overs cloud nodes. Through this section different task scheduling algorithms applied in cloud environment with suitable verification and different aims will be presented and discussed in details. As Fang et al., in [13] introduced a two levels task scheduling mechanism based on load balancing in cloud computing. Through the first level a task description of each virtual machine (VM) is created including network resources, storage resources and other resources based on the needs of the tasks created by the user applications. In the second level scheduler assigns the adequate resources to each VM considering its load to achieve load balancing among VMs. According to the authors, this task scheduling mechanism can not only meet user's requirements, but also get high resource utilization, which was proved by simulation results in the CloudSim toolkit, although this model did not consider the network bandwidth usability and its impact on VMs load.

In [14] Lin et al., proposed a Dynamic Round-Robin (DRR) algorithm for energy-aware virtual machine scheduling and consolidation during which VMs are moved from the retired physical machine to other physical machines in duty . According to the authors, the proposed algorithm was compared to GREEDY, ROUNDROBIN and POWERSAVE schedulers showing superiority in reducing the amount of consumed power although it did not consider the load and resources of the destination physical machines to which the VMs will be migrated to. Also did not mention any thing about the rules to consider when the physical machine should retire and forced its VMs to migrate.

A year after, Ghanbari et al., in [15] introduced a scheduling algorithm based on job priority named (PJSC) where each job is assigned resources based on its priority, in other words higher priority jobs gain access to resources first. Simulation results as clarified by the authors indicated that PJSC has reasonable complexity but sufferer from increasing makespan. In addition to that PJSC may cause Job starvation as the jobs with less priority may never gain access to the resources they need.

In [16] Maguluri et al., considered a stochastic model for load balancing and scheduling in cloud computing clusters. Their primary contribution was the development of frame-based non-preemptive VM configuration policies. These policies can achieve a nearly optimal throughput through selecting sufficiently long frame durations. Simulations indicate that long frame durations are not only good from a throughput perspective but also seem to provide good delay performance but also may cause starvation.

Gulati et al., in [17] studied the effect of enhancing the performance of Round Robin with a dynamic approach through varying the vital parameters of host bandwidth, cloudlet long length, VM image size and VM bandwidth. Experimental results indicated that Load had been optimized through setting dynamic round robin by proportionately varying all the previous parameters.

Agha and Jassbi in [18] proposed a RR based technique to obtain quantum time in each cycle of RR using arithmetic-harmonic mean (HARM), which is calculated by dividing the number of observations by the reciprocal of each number in series. According to the proposed technique if the burst time of a process is smaller than the previous one, HARM should be utilized for calculating quantum otherwise the arithmetic mean is utilized. The simulation results indicated that in some cases the proposed algorithm can provide better scheduling criteria and improve the average Turnaround Time (TT) and Average Waiting Time (AWT). These results according to the authors may indicate enhancement in RR performance but still missing the consideration of the arrival time of each process to verify the values of TT and AWT besides a real time implementation.

Tsai et al., in [19] introduced an optimization technique named Improved Differential Evolution Algorithm (IDEA) trying to optimize the scheduling of series of subtasks on multiple resources based on cost and time models on cloud computing environment. The proposed technique makes benefit of the Differential Evolution Algorithm (DEA) abilities in global exploration of macro-space and using fewer control parameters and Taguchi method systematic reasoning abilities in exploiting the better individuals on micro-space to be potential offspring. Experimental results were conducted using five-task five-resource and the ten-task ten-resource problems indicating the effectiveness of the IDEA in optimizing task scheduling and resource allocation while considering cost compared to the original DEA, NSGA-II, SPEA2 and IBEA.

Ergu et al., in [20] introduced a model for task-oriented resource allocation where, tasks are pairwise compared based on network bandwidth, complete time, task costs, and reliability of task. Resources are allocated to tasks based on task weight calculated using analysis hierarchy process. Furthermore, an induced bias matrix is used to identify the inconsistent elements and improve the consistency ratio when conflicting weights in various tasks are assigned. According to the authors testing was proceeded using two theoretical and not real time evaluation examples which indicated that the proposed model still needs more testing.

Karthick et al., in [21] introduced a scheduling technique that dynamically schedule jobs through depicting the concept of clustering jobs based on burst time in order to reduce starvation. Compared to other traditional techniques as FCFS and SJF, the proposed technique effectively utilizes the unused free space in an economic way although it hadn't considered consumed energy and the increasing number of submitted jobs.

In [22], Lakra and Yadav, introduced a multi-objective task scheduling algorithm for mapping tasks to VMs via non-dominated sorting after quantifying the Quality of Service values of tasks and VMs. The proposed algorithm mainly considered improving the throughput of the datacenter and reducing the cost without violating the Service Level Agreement (SLA) for an application in cloud SaaS environment. The experiments results indicated an accepted performance of the proposed algorithm although it did not consider many of the Quality of Service factors including awareness of VMS energy.

While Dash et al., in [23] presented a dynamic quantum scheduling algorithm based on RR, named Dynamic Average Burst Round Robin (DABRR). The proposed algorithm was tested and compared to traditional RR, Dynamic Quantum with Re-adjusted Round Robin (DQRRR), Improved Round Robin with Varying time Quantum (IRRVQ), Self Adjustment Round Robin (SARR), and Modified Round Robin (MRR) indicating the superiority of the DABRA. However the authors did not clarify DABRA's response to new arrival processes also sorting processes in an ascending order based on their burst time may cause starvation to processes with long burst time.

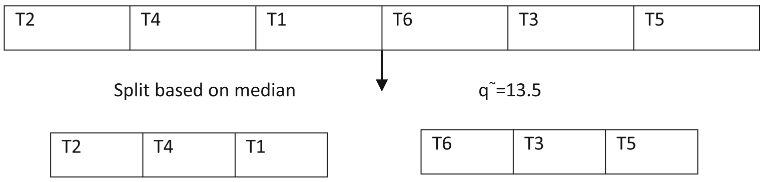
Tunc et al., in [24] presented a new metric called Value of Service mainly consider completed tasks within deadline through balancing its value of completing within deadline with energy consumption. They defined the proposed metric as the sum of the values for all tasks that are executed during a given period of time including task arrival time. Each resource was assigned to a task based on the number of the homogeneous cores and amount of memory. The experiments were conducted using IBM blade server using Keyboard-Video-Mouse (KVM), indicating an improvement in performance enhanced by a noticeable reduction in energy consumption only in case of completing tasks within deadline. The authors did not really clarify how their model reacts to the increasing number of tasks especially with the same arrival time.

In the same year Abdul Razaque et al., in [25] introduced a nonlinear programming divisible task scheduling algorithm, allocating the workflow of tasks to VMs based on the availability of network bandwidth. The problem here with this algorithm, it only considered a single criteria of a network for allocating tasks to VMs while neglecting the VMs energy consumption that may cause these machines to retire forcing tasks to terminate.

Recently bio-inspired algorithms as Ant Colony Optimization (ACO), Cuckoo Search (CS), genetic algorithm (GA), Particle Swarm Optimization (PSO) and Bees Life Algorithm (BLA) etc.. have played major role in scheduling tasks over cloud nodes as Mizan et al., in [26] developed a modified job scheduling algorithm based on BLA and greedy algorithm for minimizing make span in hybrid cloud. Based on the authors claim experiments were conducted indicating that the proposed algorithm has outperformed both greedy algorithm and firefly algorithm in make span reduction.

Table 1 Submitted tasks burst and arrival

Task	Burst time	Arrival time
T1	12	0
T2	8	0
T3	23	1
T4	10	2
T5	30	3
T6	15	4



In [27] Ge and Wei utilized genetic algorithm (GA) as an optimization technique utilized by the master node to schedule the waiting tasks to computing nodes. Before the scheduling procedure takes place all tasks in the job queue have to be evaluated first. Based on the authors the simulation results indicated reduction in make span and better balanced load across all cloud nodes for the GA over FIFO.

Raju et al., in [28] presented a hybrid algorithm combining the advantages of Ant Colony Optimization (ACO) and Cuckoo Search (CS) in order to reduce make span, based on Job execution within specified time interval. The experimental results clarified that the proposed algorithm achieved better results in terms of makspan reduction over the original ant colony optimization algorithm but not over other related algorithms as RR.

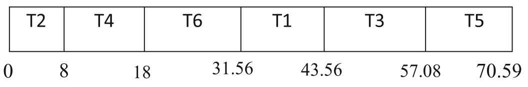
Ramezani et al., in [29] developed Multi-Objective Jswarm (MO-Jswarm) scheduling algorithm to determine the optimal task distribution over the virtual machines (VMs) attempting to balance between different conflicting objectives including task execution time, task transferring time, and task execution cost. According to the authors the proposed algorithm had the ability to enhance the QOS and to provide a balanced trade-off between the conflicted objectives.

Many other studies have investigated utilizing bio-inspired algorithms in task scheduling over cloud as in [30–40] but most of these studies have suffered from the intricacy and high time and space complexity.

Most of the previous studies concentrated on enhancing one or two of the Qos Standards either by minimizing or maximizing them although in cloud environment, it is highly recommended to consider various criteria as execution time, cost, bandwidth and energy consumption. Other studies even claimed to reach optimality in performance as in [41] and [42], while others have investigated task scheduling from load balancing prospective as in [43–47] concentrating on balancing workload with consumed energy. The experimentations results of all of these studies claimed that they had improved waiting, turnaround time or even throughput but none of them give a real solution to starvation or even approached to

Table 2 Task quantum calculations in first round

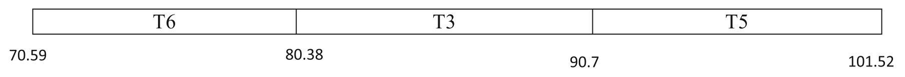
T	$B_{ij} = B_{i(j-1)} - q_{i(j-1)}$	$(B_{ij} + q_{i(j-1)})^2$ Q1, $\propto = 1$	$(B_{ij} - q_{i(j-1)})^2$ Q2, $\propto = 0$	$q_{ij} = q^{\sim} + \frac{q^{\sim}}{(B_{ij} + (-1)^{1-\propto} . q_{i(j-1)})^2}$
T2	8-0= 8	$(8+0)^2$= 64	--------	q_{21} = 13.71
T4	10-0= 10	$(10+0)^2$= 100	--------	q_{41} =13.63
T6	15-0= 15	--------	$(15-0)^2$= 225	q_{61} =13.56
T1	12-0= 12	$(12+0)^2$= 144	--------	q_{11} =13.59
T3	23-0= 23	--------	$(23-0)^2$= 529	q_{31} =13.52
T5	30-0= 30	--------	$(30-0)^2$= 900	q_{51} =13.51
	$\sum B_{i1}$=98			$\sum q_{i1}$=81.52



Gant chart based on table (2)

Table 3 Task quantum calculations in the second round

T	$B_{ij} = B_{i(j-1)} - q_{i(j-1)}$	$(B_{ij} + q_{i(j-1)})^2$ $\mathbf{Q1}, \propto = 1$	$(B_{ij} - q_{i(j-1)})^2$ $\mathbf{Q2}, \propto = 0$	$q_{ij} = q^{\sim} + \frac{q^{\sim}}{(B_{ij} + (-1)^{1-\propto}.\, q_{i(j-1)})^2}$
T2	--------	--------	--------	--------
T4	--------	--------	--------	--------
T6	15-13.56=1.44	--------	$(1.44\text{-}13.56)^2$=146.8	q_{62} =9.79
T1	--------	--------	--------	--------
T3	23-13.52= 9.48	--------	$(9.48\text{-}13.52)^2$= 16.32	q_{32} =10.32
T5	30-13.51= 16.49	--------	$(16.49\text{-}13.51)^2$= 8.88	q_{52} =10.82
	$\sum B_{i2}$=**27.41**	--------		$\sum q_{i2}$=**30.93**



Gant chart based on Table 3

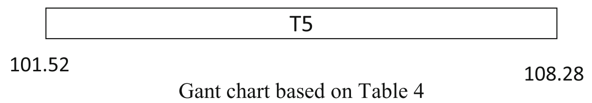
solve it. As the starvation problem is considered one of the major scheduling dilemmas, so in this paper we tried to overcome or partially overcome the starvation problem of long tasks though proposing a hybrid scheduling algorithm based on two tradition scheduling algorithms SJF and RR. These two algorithms were intentionally picked to make benefit of SJF fast secluding while solving its starvation problem using RR enhanced with dynamic quantum. Through the proposed algorithm the ready queue is split into two sub-queues Q_1 and Q_2 one for short tasks while the other is for long tasks, which will be discussed in details in next section.

SJF and RR with dynamic quantum hybrid algorithm (SRDQ)

As mentioned before, we are trying in this work to overcome the starvation problem by proposing a new hybrid scheduling technique based on SJF and RR scheduling techniques named SRDQ. SRDQ avoids the disadvantages of both of SJF and RR so that the evaluation of the performance metrics increases rather than decreases the probability of starvation occurrence as far as possible. In the RR stage of SRDQ, the time slice works on avoiding the traditional cons that lead to high waiting times and rare deadlines met. RR time slice or quantum setting is a very challenging process, as if the quantum is too short, too many context switches will lower the CPU efficiency while setting the quantum too long may cause poor response time and approximates First-Come-First-Serve (FCFS) algorithm. So in this paper, the researchers concentrated on calculating the optimal quantum interval at each round of RR algorithm while splitting the tasks ready queue into two sub-queues Q_1 for short tasks and Q_2 for long tasks using the median as the threshold of the tasks length in other words the tasks longer than the median to be inserted in Q_2 while the shorter ones to

Table 4 Task quantum calculations in the third round

T	$B_{ij} = B_{i(j-1)} - q_{i(j-1)}$	$(B_{ij} + q_{i(j-1)})^2$ $\mathbf{Q_1}, \propto = 1$	$(B_{ij} - q_{i(j-1)})^2$ $\mathbf{Q_2}, \propto = 0$	$q_{ij} = q^{\sim} + \frac{q^{\sim}}{(B_{ij} + (-1)^{1-\propto}.\, q_{i(j-1)})^2}$
T2	--------	--------	--------	--------
T4	--------	--------	--------	--------
T6	--------	--------	--------	--------
T1	--------	--------	--------	--------
T3	--------	--------	--------	--------
T5	16.49-9.73= 6.76	--------	$(6.76\text{-}9.76)^2$= 9	9.41
	$\sum B_{ij}$=**6.76**	--------		$\sum q_{ij}$=**9.41**



Gant chart based on Table 4

Table 5 The response, waiting and turnaround times of the SRSQ compared to SJF and RR

T	Response time			Waiting time			Turnaround time		
	SRDQ	SJF	RR	SRDQ	SJF	RR	SRDQ	SJF	RR
T1	31.56	18	0	31.56	18	48	43.56	30	60
T2	0	0	5	0	0	30	8	8	38
T3	43.56	45	10	67.7	44	65	90.7	67	88
T4	8	8	15	8	6	38	18	16	48
T5	57.8	68	20	78.28	65	68	108.28	95	98
T6	18	30	25	65.30	26	60	80	41	75
Average time	26.486	28.166	12.5	41.806	26.5	51.5	58.09	42.83	67.833

be inserted in Q_1. Tasks will be executed mutually, two short tasks from Q_1 and one long task from Q_2 will be executed which will lead to reducing long tasks waiting time without the disruption of the SJF in preferring short tasks. SRDQ is designed to be a unit based algorithm based on queuing data structure effectively and optimizing the execution time as possible. SRDQ involves 6 main steps as following:

1. Arrange all submitted tasks, T_i, $i = 1, 2, ..,$ number of submitted tasks, according to their burst time.
2. Compute the median, $q^{\sim}$, of the burst times of all tasks.
3. If a burst time of a task T, $B(T)$, is less than or equal to the median, insert T into a Q_1 otherwise insert T into Q_2.

Table 6 CloudSim simulation parameters

Number of cloud hosts	1
Number of cloud users	1
Million Instructions Per Second (MIPS)	1000
Number of CPUs per Host	10
Host memory (MB)	2048
Host storage	1,000,000
Host Bandwidth	10,000
Number of CPUs per VM	1,2,3
Virtual Machine Size (MB)	10,000
Virtual Machine Memory (MB)	512
Virtual Machine Bandwidth	1000
System architecture	"×86"
Operating system	"Windows 7"
time zone this resource located	10.0
the cost of using processing	3.0
the cost of using memory	0.05
the cost of using storage	0.001
the cost of using bandwidth	0.0

4. The quantum (q_{ij}) is calculated based on the current executed task source queue (whether it is from Q_1 or Q_2), and the round to be executed in, as following:

$$q_{ij} = q^{\sim} + \frac{q^{\sim}}{\left(B_{ij} + (-1)^{1-\propto} . q_{i(j-1)}\right)^2} \tag{1}$$

where q_{ij} is the quantum at iteration j, i:1, 2, . . ., n and, B_{ij} is burst time of task i at iteration j, $q_{i(j-1)}$, and $\propto$ is a binary selector $\propto$={0,1}. In the first round, j = 1, $q_{i(j-1)}$ is set to zero as there is no previous rounds. On the other hand, based on the source queue, $\propto$ will be set to either zero or one as in:

a. If the resource is taken from Q_1, $\propto$ will be set to one and thus Eq. (1) will be modified as follows:

$$\mathrm{q_{ij}} = \mathrm{q}^{\sim} + \frac{\mathrm{q}^{\sim}}{\left(\mathrm{B_{ij}} + \mathrm{q_{i(j-1)}}\right)^2} \tag{2}$$

b. If the resource is taken from Q_2, $\propto$ will be set to zero and thus Eq. (1) will be modified as follows:

$$\mathrm{q_{ij}} = \mathrm{q}^{\sim} + \frac{\mathrm{q}^{\sim}}{\left(\mathrm{B_{ij}} - \mathrm{q_{i(j-1)}}\right)^2} \tag{3}$$

5. The first two tasks of Q_1 are assigned to the resources followed by the first task of Q_2.
6. Step 4 is continuously repeated till the $Q_1$and Q_2 are empty.
7. In case of the of a new task arrival or a task is finished $q^{\sim}$ will be updated dynamically as following:

Table 7 Data sets

Task	Dataset1		Dataset2		Dataset3	
	Burst Time	Arrival Time	Burst Time	Arrival Time	Burst Time	Arrival Time
T1	49	0	251	0	33	0
T2	98	1	177	1	201	1
T3	143	2	152	2	98	2
T4	187	3	299	3	116	3
T5	244	4	47	4	11	4
T6	252	4	84	5	100	5
T7	199	4	244	3	33	6
T8	67	5	124	3	78	7
T9	83	3	55	4	18	4
T10	75	6	180	6	64	8

a. In case of a new task arrival, it will be inserted in Q_1 or Q_2 based on its burst time and $q^{\sim}$. In this case, $q^{\sim}$ will be updated as follow:

$$\tilde{q} = \tilde{q} + \frac{\tilde{q}}{B_{new}} \quad (4)$$

Where B_{new} is the new task burst time.

b. In case of a task is finished, $q^{\sim}$ will be updated as:

$$\tilde{q} = \tilde{q} - \frac{\tilde{q}}{B_{terminated}} \quad (5)$$

Where $B_{\text{terminated}}$ is the finished task burst time.

For more explanation, the following illustrative example discusses the case of executing 6 tasks using SRDQ (Table 1).

Round (1) (Table 2)
$q^{\sim} = 13.5$

Round (2)
T2, T4 and T1 are all finished in the first round so $q^{\sim}$ will be updated as following:

$$\tilde{q} = \tilde{q} - \frac{\tilde{q}}{B_{\text{terminated}}}$$

- After finishing T2, $B_{\text{terminated}} = 8$, $q^{\sim} = 13.5 - (13.5/8) = 11.81$
- After finishing T4, $B_{\text{terminated}} = 10$, $q^{\sim} = 11.81 - (11.81/10) = 10.62$

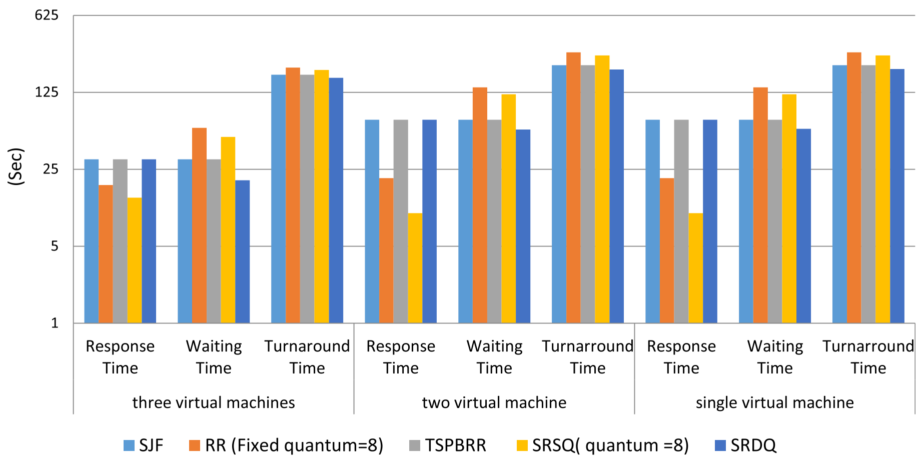


Fig. 1 First Experimentation Results using Dataset1, on (1, 2, 3) VMs

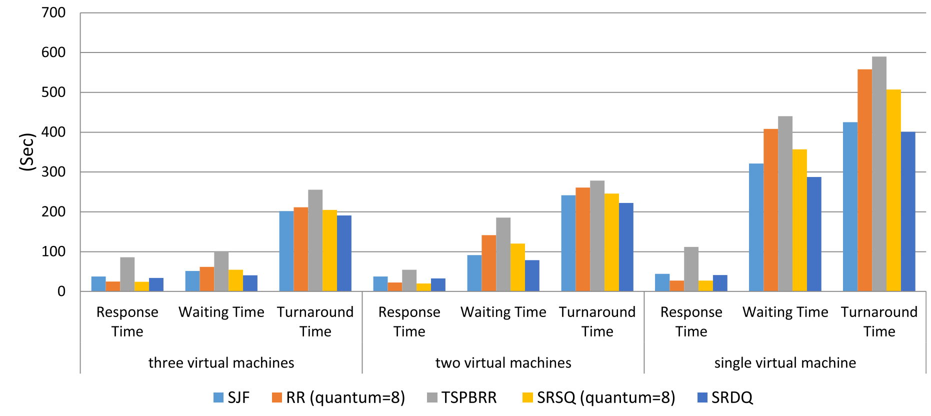


Fig. 2 Second Experimentation Results using Dataset2, on (1, 2, 3) VMs

– After finishing T1, $B_{terminated}$ = 12, $q^{\sim}$ = 10.62 –(10.62/12) = 9.73

As three tasks finished in the same round, $q^{\sim}$ will be updated three times and we obtain $q^{\sim}$ = 9.73 in round 2 (Table 3).

Round (3)

T6 and T3 are finished in the second round so $q^{\sim}$ will be updated as following:

$$\tilde{q} = \tilde{q} - \frac{\tilde{q}}{B_{terminated}}$$

– After finishing T6, $B_{terminated}$ = 15, $q^{\sim}$ = 9.73 –(9.73/15) = 9.08

– After finishing T3, $B_{terminated}$ = 23, $q^{\sim}$ = 9.08 –(9.08/23) = 8.47

Thus $q^{\sim}$ = 8.47 in this round (Table 4).

Table 5 demonstrates the response, waiting and turnaround times of the SRSQ compared to SJF and RR. We can detect that SRDQ achieved less response time compared to SJF but with higher turnaround and waiting time. Although, RR had really achieved good response time but with comparable waiting time to SRDQ. We can finally say that SRDQ is the balancing point between SJF and RR, in which we tried to overcome or at least reduce RR and SJF problems especially the starvation dilemma.

Simulation settings

Simulation environment

The proposed hybrid algorithm was implemented and tested in the CloudSim environment toolkit 3.0.3 which provides a generalized and extensible simulation framework that enables modeling, simulation, and experimentation of emerging Cloud computing infrastructures and application services, allowing its users to focus on specific system design issues that they want to investigate, without getting

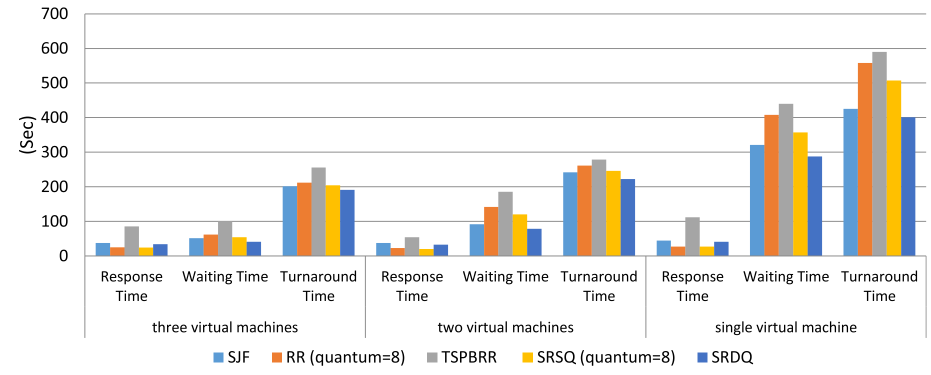


Fig. 3 Third Experimentation Results using Dataset3, on (1,2,3) VMs

Table 8 Cloudlet specification

Cloudlet ID	VM ID	Arrival	Brust time
0	0	0	12
5	0	8	13
2	0	5	44
7	0	11	92
4	0	8	101
8	0	13	144
3	0	7	157
9	0	15	158
6	0	19	179
1	0	5	210

concerned about the low level details related to Cloud-based infrastructures and services [4, 48]. The simulation settings and parameters employed in the CloudSim experiments are summarized in Table 6.

Performance metrics

The following metrics were considered through the evaluation process [49]:

- *Waiting Time:* Average time a process spends in the run queue.
- Response Time: Average time elapsed from when a process is submitted until useful output is obtained.
- Turnaround Time: Average time elapsed from when a process is submitted to when it has completed.

Experimental results & discussion

For evaluation purposes, three different datasets were utilized through testing the proposed algorithm against three different scheduling algorithms: traditional SJF, traditional RR and Time Slice Priority Based RR (TSPBRR) [50]. It was tested in two cases the first SRSQ with static task quantum through each iteration, while changing from one iteration to the next and the second SRSQ with dynamic quantum through the same iteration and from one iteration to the next. The algorithms performances were evaluated based on turnaround time, waiting time and response time. Each dataset consists of randomly generated and dynamically shuffled ten tasks denoted as T_1, $T_{2,\ldots\ldots}T_{10}$ and each task is characterized by its arrival time and burst time, as shown in Table 7.

To evaluate SRDQ a simulated Cloud computing environment consists of a single data center, a broker and a user, constructed by cloud-based interface provided by CloudSim, series of experiments are performed. The allocation of VMs (Virtual Machine) to hosts utilizes the default FCFS algorithm, while for allocating the cloudlets (tasks) to the virtual machines space-shared policy is used so that the tasks are executed sequentially in each VM. By using this policy each task unit had its own dedicated core therefore number of incoming tasks or queue size did not affect execution time of individual task units as the proposed algorithm is a non-primitive technique.

In CloudSim environment, evaluation experiments were performed in three cases using one VM, two VMs and three VMs. While the assumptions behind the proposed algorithm involve:

- All cloudlets which have to be processed are available,
- At runtime no more cloudlets are added,
- The environment is also static i.e. no more resources are added at runtime.

Finally the inner code of CloudSim was modified to test our proposed algorithm and also to compare it to the traditional RR and SJF. Then our own classes for the scheduling algorithm were defined to extend the basic

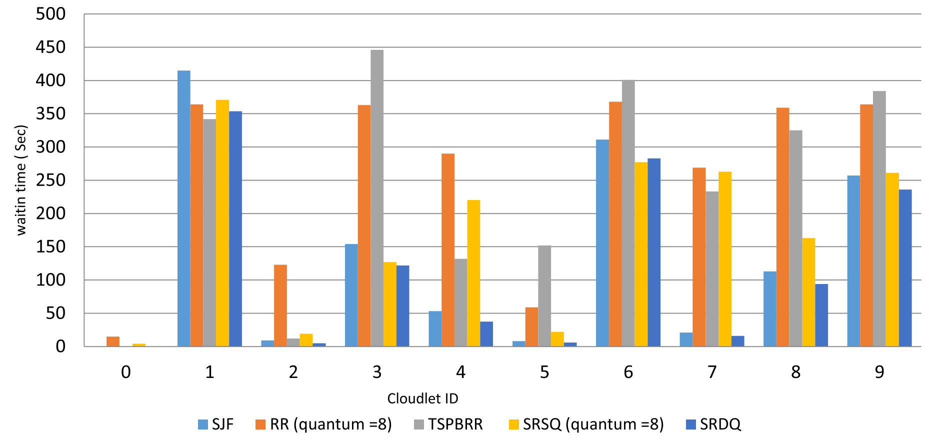


Fig. 4 Cloudlets waiting time on one VM

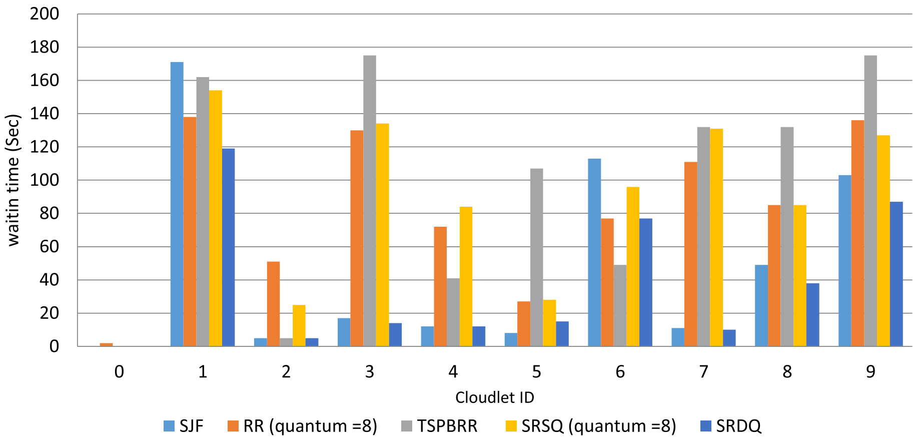


Fig. 5 Cloudlets waiting time on two VMs

CloudSim classes. The same datasets were used in three different cases one VM, Two VMs and three VMs. Each dataset was used in each case as shown in the next figures.

Figure 1 clarifies the experimentation results of the implemented algorithms using dataset1 on one, two and three VMs. It is noticed that SRSQ and RR have the least response time in all phases but suffer from the highest turnaround and waiting time. While SRDQ has the least waiting and turnaround time which is also nearly comparable to the SJF.

From Fig. 2, we can also notice that RR and SRSQ really achieved good comparable response time but still suffer from higher waiting and turnaround time, while TSPBRR suffered from elevated values compared to SJF and SRDQ. Finally the SRDQ is again the winner that achieved the least waiting and turnaround time.

Finally in the third and final experiments using dataset3 as shown in Fig. 3, we can notice that with the increasing number of VMs, SRDQ performance became better and exceeded the rest in all evaluation metrics, while TSPBRR performance was degrading.

A final test was done through the CloudSim environment using a randomized dataset of 10 cloudlets (tasks) with random arrival and long burst time generated by the environment given in Table 8 to detect the impact of the proposed algorithm on reducing the starvation problem.

It is noticed that the cloudlets (1, 6, 9, 3, and 8) burst time is long which means that theses cloudlets will suffer from starvation if the SJF scheduler was applied and also will suffer if the RR quantum was small. In the proposed algorithms with its two versions, we tried to balance between reducing cloudlets waiting time and increasing quantum value also tried to achieve fairness in selecting cloudlets for execution through having two short cloudlets from Q_1 and one long cloudlet from Q_2.

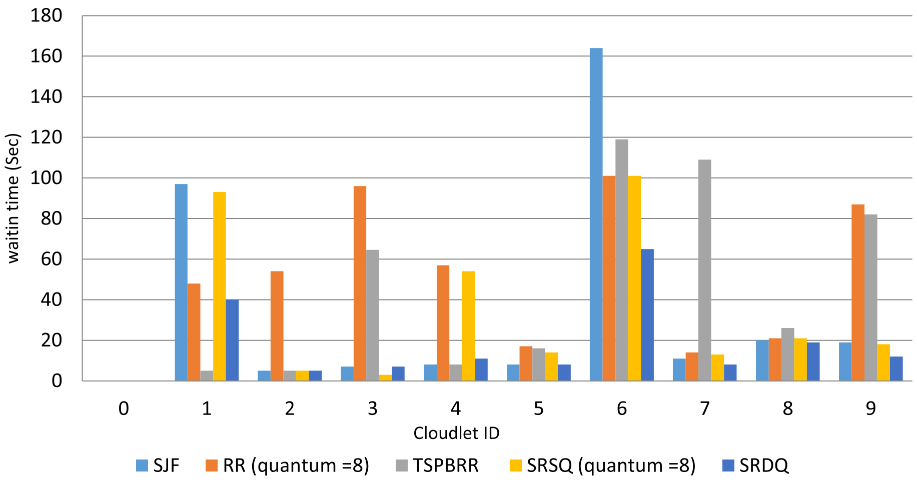


Fig. 6 Cloudlets waiting time on three VMs

Figures 4, 5 and 6 clarifies the waiting time of each cloudlets applied on a 1, 2 and 3 VMs, from which we can see that the proposed algorithm with its two versions had achieved much better reduction in cloudlets waiting time especially cloudlets with long burst time. We can also notice that SRDQ performance exceeds SRSQ or at least comparable to it. SJF achieved the worst waiting time especially with cloudlets with long burst time while TSPBRR achieved better waiting time in most cases that its traditional version.

Reducing the waiting time indicates that the average time a cloudlet spends in the run queue is reduced which leads to reducing cloudlet starvation. SRDQ achieved this waiting time reduction and thus starvation through having the two sub-queues Q1 and Q2 where Q1 for nearly short tasks and Q2 for the rest depending on the tasks median. Many tests and trials have been done by the researchers to find the best methodology for selecting tasks from Q1 and Q2 to be assigned to resources and finally found that as clarified in the algorithm that having two tasks from Q1 and one task from Q2 really have a good impact on reducing task starvation.

From the simulations results, it is obvious that SRSQ and SRDQ had achieved a good performance compared to the traditional RR and SJF and also to TSPBRR in response, turnaround and waiting time. It is also obvious that SRDQ had superiority on SRSQ in reducing waiting and turnaround times while SRSQ exceeds in reducing response time. We can assure that the proposed algorithm in its both versions (SRSQ and SRDQ) had achieved a good reduction in the waiting time of each task and also the overall waiting average, from which we can say that it leads to reducing task starvation which is one of our first priorities.

But one last issue, the experiments results had shown that dynamicity in task quantum had a good impact on reducing task waiting time and turnaround time, while the dynamicity in each task quantum from round to round had a good impact on reducing response time so we can see that SRSQ had exceeds SRDQ in response as it only depends on having a static quantum for all tasks that did not change from round to round. SRDQ works as the balancing point with, waiting, turnaround and starvation reduction especially in tasks with long burst times and with a comparable performance in response to SJF, RR and TSPBRR. From all of the above, we can surely conclude that having the optimum task quantum value is nearly impossible.

Conclusions

Achieving optimality in scheduling tasks over computing nodes in cloud computing is the aim of all researchers interested in both scheduling and cloud. Balancing between throughput, waiting time and response time may provide a way to approach scheduling optimality but on another level it may causes long tasks starvation. Most of the previous studies have concentrated only on one side either starvation or throughput but not both so in this study we have tried to develop a hybrid algorithm based on SJF and dynamic quantum RR, while concentrating on splitting the ready queue into to sub-queues Q1 for short tasks and Q2 long tasks.

Three different datasets were utilized for evaluation conducted using CloudSim environment 3.0.3 in two different versions SJF&RR with Dynamic Quantum (SRDQ) and SJF&RR with Static Quantum (SRSQ) with 1,2 and 3 virtual machines. Experimentations results indicated that the proposed algorithm has outperformed the state of art in minimizing turnaround and waiting times with comparable response time in addition to partially reducing long tasks starvation.

In the future the researchers intend to proceed their experiments in finding a better task quantum calculation methodology that balance between the static and dynamic quantum values to achieved better reduction in waiting and thus reducing task starvation.

Authors' contributions

SS carried out the literature review. SE developed the mathematical models. SS and MJ jointly conducted the simulations and drafted the manuscript. SE provided useful remarks and critically reviewed the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Computer Science, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt. [2]University of Al-Qadisiyah, Al-Qadisiyah, Iraq.

Received: 17 January 2017 Accepted: 30 May 2017
Published online: 15 June 2017

References

1. Lu CW, Hsieh CM, Chang CH, Yang CT (2013, July) An improvement to data Service in Cloud Computing with Content Sensitive Transaction Analysis and Adaptation, Computer Software and applications Conference workshops (COMPSACW), 2013 IEEE 37th annual, vol 463-468
2. Azeez A, Perera S, Gamage D, Linton R, Siriwardana P, Leelaratne D, Fremantle P (2010, July) Multi-tenant SOA middleware for cloud computing, Cloud computing (cloud), 2010, IEEE 3rd International Conference on, pp 458–465
3. Demchenko Y, de Laat C (2011, March) Defining generic architecture for cloud infrastructure as a Service model, The International symposium on grids and clouds and the open grid forum Academia Sinica, pp 2–10
4. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41(1):23–50
5. Ganapathi A, Chen Y, Fox A, Katz R, Patterson D (2010, March) Statistics-driven workload modeling for the cloud, Data Engineering workshops (ICDEW), 2010 IEEE 26th International Conference on, pp 87–92
6. Garg SK, Buyya R (2011, December) Networkcloudsim: Modelling parallel applications in cloud simulations, Utility and cloud computing (UCC), 2011 fourth IEEE International Conference on, pp 105–113

7. Buyya R, Ranjan R, Calheiros RN (2009, June) Modeling and simulation of scalable cloud computing environments and the Cloudsim toolkit: challenges and opportunities, High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, pp 1–11
8. Das AK, Adhikary T, Razzaque MA, Hong CS (2013, January) An intelligent approach for virtual machine and QoS provisioning in cloud computing, Information Networking (ICOIN), 2013 International Conference on, pp 462–467
9. Bhoi U, Ramanuj PN (2013) Enhanced max-min task scheduling algorithm in cloud computing. International Journal of Application or Innovation in Engineering and Management (IJAIEM) 2(4):259–264
10. Wang SC, Yan KQ, Liao WP, Wang SS (2010) Towards a load balancing in a three-level cloud computing network, Computer Science and information technology (ICCSIT), 2010 3rd IEEE International Conference on, 1, pp 108–113
11. Venters W, Whitley EA (2012) A critical review of cloud computing: researching desires and realities. J Inf Technol 27(3):179–197
12. Santra S, Dey H, Majumdar S, Jha GS (2014, July) New simulation toolkit for comparison of scheduling algorithm on cloud computing, Control, instrumentation, communication and computational technologies (ICCICCT), 2014 International Conference on, pp 466–469
13. Fang Y, Wang F, Ge J (2010) A task scheduling algorithm based on load balancing in cloud computing. Web information systems and Mining. Springer, Berlin Heidelberg, pp 271–277
14. Lin CC, Liu P, Wu JJ (2011, July) Energy-aware virtual machine dynamic provision and scheduling for cloud computing, CLOUD computing (CLOUD), 2011 IEEE International Conference on, pp 736–737
15. Ghanbari S, Othman M (2012) A priority based job scheduling algorithm in cloud computing. Procedia Engineering 50:778–785
16. Maguluri ST, Srikant R, Ying L (2012) Stochastic models of load balancing and scheduling in cloud computing clusters, INFOCOM, 2012 Proceedings IEEE, pp 702–710
17. Gulati A, Chopra RK (2013) Dynamic round Robin for load balancing in a cloud computing. IJCSMC 2(6):274–278
18. Agha AEA, Jassbi SJ (2013) A new method to improve round Robin scheduling algorithm with quantum time based on harmonic-arithmetic mean (HARM). International Journal of Information Technology and Computer Science 5(7):56–62
19. Tsai JT, Fang JC, Chou JH (2013) Optimized task scheduling and resource allocation on cloud computing environment using improved Differential Evolution algorithm. Comput Oper Res 40(12):3045–3055
20. Ergu D, Kou G, Peng Y, Shi Y, Shi Y (2013) The analytic hierarchy process: task scheduling and resource allocation cloud computing environment. J Supercomput 64(3):835–848
21. Karthick AV, Ramaraj E, Subramanian RG (2014, February) An efficient multi queue job scheduling for cloud computing, Computing and communication technologies (WCCCT), 2014 world congress on, pp 164–166
22. Lakra AV, Yadav DK (2015) Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. Procedia Computer Science 48:107–113
23. Dash AR, Samantra SK (2016) An optimized round Robin CPU scheduling algorithm with dynamic time quantum. International Journal of Computer Science, Engineering and Information Technology (IJCSEIT) 5(1):7–26. doi:10.5121/ijcseit.2015.5102
24. Tunc C, Kumbhare N, Akoglu A, Hariri S, Machovec D, Siegel HJ (2016, December) Value of Service based task scheduling for cloud computing systems, Cloud and autonomic computing (ICCAC), 2016 International Conference on, pp 1–11. doi:10.1109/ICCAC.2016.22
25. Razaque A, Vennapusa NR, Soni N, Janapati GS (2016, April) Task scheduling in cloud computing, Long Island systems, applications and technology Conference (LISAT), 2016 IEEE, pp 1–5. doi:10.1109/LISAT.2016.7494149
26. Mizan, T., Al Masud, S. M. R., & Latip, R. (2012). Modified bees life algorithm for job scheduling in hybrid cloud.
27. Ge Y, Wei G (2010) GA-based task scheduler for the cloud computing systems. In Web Information Systems and Mining (WISM), 2010 International Conference on, Vol. 2. IEEE, Sanya, China, p. 181–186
28. Raju R, Babukarthik RG, Dhavachelvan P (2013) Hybrid ant Colony optimization and cuckoo search algorithm for job scheduling, Advances in computing and information technology. Springer, Berlin Heidelberg, pp 491–501
29. Ramezani F, Lu J, Hussain F (2013, December) Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization, *International Conference on Service-oriented computing*. Berlin, Springer Berlin Heidelberg, pp 237–251
30. Gan GN, Huang TL, Gao S (2010). Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on (pp. 60-63). IEEE, Guilin
31. Wang L, Siegel HJ, Roychowdhury VP, Maciejewski AA (1997) Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. Journal of parallel and distributed computing 47(1):8–22
32. Zhao C, Zhang S, Liu Q, Xie J, Hu J (2009) Independent tasks scheduling based on genetic algorithm in cloud computing. In Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on. IEEE, Beijing, p. 1–4
33. Gu J, Hu J, Zhao T, Sun G (2012) A new resource scheduling strategy based on genetic algorithm in cloud computing environment. Journal of Computers 7(1):42–52
34. Mocanu EM, Florea M, Andreica MI, Ţăpuş N (2012) Cloud computing—task scheduling based on genetic algorithms. In Systems Conference (SysCon), 2012 IEEE International. IEEE, Vancouver, p. 1–6
35. Kaur S, Verma A (2012) An efficient approach to genetic algorithm for task scheduling in cloud computing environment. International Journal of Information Technology and Computer Science (IJITCS) 4(10):74
36. Jang SH, Kim TY, Kim JK, Lee JS (2012) The study of genetic algorithm-based task scheduling for cloud computing. International Journal of Control and Automation 5(4):157–162
37. Kruekaew B, Kimpan W (2014) Virtual machine scheduling management on cloud computing using artificial bee colony, Proceedings of the International MultiConference of engineers and computer scientists, vol 1, pp 12–14
38. Bilgaiyan S, Sagnika S, Das M (2014) An analysis of task scheduling in cloud computing using evolutionary and swarm-based algorithms. Int J Comput Appl 89(2):11–18
39. Navimipour NJ, Milani FS (2015) Task scheduling in the cloud computing based on the cuckoo search algorithm. International Journal of Modeling and Optimization 5(1):44
40. Verma, A., & Kaushal, S. (2014) Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud. In Engineering and Computational Sciences (RAECS), 2014 Recent Advances in. IEEE, p. 1–6
41. Alla HB, Alla SB, Ezzati A, Mouhsen A (2017) A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing, *Advances in Ubiquitous Networking 2,* 397, 205–217. Springer, Singapore
42. Sebastio S, Gnecco G, Bemporad A (2017) Optimal distributed task scheduling in volunteer clouds. Comput Oper Res 81:231–246
43. Maharana D, Sahoo B, Sethi S (2017) Energy-efficient real-time tasks scheduling in cloud data centers. International Journal of Science Engineering and Advance Technology, IJSEAT 4(12):768–773
44. Liu Y, Xu X, Zhang L, Wang L, Zhong RY (2017) Workload-based multi-task scheduling in *Cloud Manufacturing*. Robot Comput Integr Manuf 45:3–20
45. Sofia AS, Kumar PG (2017) Energy efficient task scheduling to implement green cloud. Asian Journal of Research in Social Sciences and Humanities 7(2):443–458
46. Li, K. (2017). Scheduling parallel tasks with energy and time constraints on multiple Manycore processors In A cloud computing environment. Future generation computer systems, http://dx.doi.org/10.1016/j.future.2017.01.010.
47. Rimal BP, Maier M (2017) Workflow scheduling in multi-tenant cloud computing environments. IEEE Transactions on Parallel and Distributed Systems 28(1):290–304. doi:10.1109/TPDS.2016.2556668
48. http://www.cloudbus.org/cloudsim/ [Accessed at : 25/4/2015]
49. http://www.read.seas.harvard.edu/~kohler/class/05s-osp/notes/notes5.html. Accessed 25 Apr 2015.
50. Mohapatra S, Mohanty S, Rekha KS (2013) Analysis of different variants in round Robin algorithms for load balancing in cloud computing. Int J Comput Appl 69(22):17–21. doi:10.5120/12103-8221