RESEARCH PAPER IMPLEMENTATION REPORT

Name :	Pramod B S
Register Number :	15CO234
Email-id :	prmdbs3@gmail.com
Phone number :	9480041659

Abstract

Cloud computing delivers a computing environment where different resources are delivered as a service to the customer or multiple tenants over the internet. Task scheduling is an essential and most important part in a cloud computing environment. The task scheduling mainly focuses to enhance the efficient utilization of resources and hence reduction in task completion time. Task scheduling is used to allocate certain tasks to particular resources at a particular time instance. Many different techniques have been proposed to solve the problems of task scheduling. Task scheduling improves the efficient utilization of resource and yields less response time so that the execution of submitted tasks takes place within a possible minimum time.

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 present do not achieve the required standards and requirements as some of them only concentrated on waiting time or response time reduction or even both, thus neglecting the starved processes completely.

In this paper, a novel hybrid task scheduling algorithm named (SRDQ) combining Shortest-Job-First (SJF) and Round Robin (RR) scheduling algorithms considering a dynamic variable task quantum is proposed.

The proposed algorithms mainly relies on two basic keys

1) A dynamic task quantum to balance waiting time between short and long tasks
2) Splitting the ready queue into two sub-queues, Q1 for the short tasks and the other Q2 for the long ones.

Assigning tasks to resources from Q1 or Q2 are done mutually i.e two tasks from Q1 and one task from Q2 .

For evaluating the SRDQ algorithm’s performance, three different datasets were utilized during the algorithm simulation conducted using CloudSim environment toolkit 3.0.3 against two different scheduling algorithms SJF and RR. Here, the same datasets have been used and the implementation has been done in Python 3.6.

Experimentations and the obtained results from the tests indicate the superiority of the proposed algorithm over the state of art in reducing waiting time, response time and partially the starvation of long tasks.

<u>Introduction</u>

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 requirements. 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 for multi-tenant computing which allows the users to share resources.

One of the goals of cloud computing is to use the resources efficiently and gain maximum profit. Scheduling is a critical problem in Cloud computing, because a cloud provider has to serve many users in Cloud computing system. So scheduling is the major issue in establishing Cloud computing systems. The scheduling algorithms should order the jobs in a way where balance between improving the performance and quality of service and at the same time maintaining the efficiency and fairness among the jobs.

In cloud computing, available resources must be checked and scheduled using an efficient task scheduler to be assigned to clients based on their requests for maximum efficiency of existing resources by the Cloud Service Provider. Thus, 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.

Job Scheduling of cloud computing refers to dispatch the computing tasks to resource pooling between different resource users according to certain rules of resource use under a given cloud circumstances. At present there is not a uniform standard for job scheduling in cloud computing. Resource management and job scheduling are the key technologies of cloud computing that plays a vital role in an efficient cloud resource management.

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.

Conventional task scheduling algorithms as Shortest-Job-First (SJF), Round Robin (RR), First-Come-First-Serve (FCFS), Multilevel queue scheduling (MQ), Max-Min and Min-Min had achieved breathtaking results over years in different computer systems types but always suffer from big dilemmas caused by higher waiting time in RR and FCFS algorithms and starvation faced in SJF and Max-Min algorithms.

Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive).

Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an Operating System concept. In order to schedule processes fairly, a round-robin scheduler generally employs time-sharing, giving each job a time slot or quantum (its allowance of CPU time), and interrupting the job if it is not completed by then. The job is resumed next time a time slot is assigned to that process.

If the process terminates or changes its state to waiting during its attributed time quantum, the scheduler selects the first process in the ready queue to execute. In the absence of time-sharing, or if the quanta were large relative to the sizes of the jobs, a process that produced large jobs would be favoured over other processes. Round Robin algorithm is a pre-emptive algorithm as the scheduler forces the process out of the CPU once the time quota expires.

A different approach to CPU scheduling is the shortest-jobfirst (SJF) scheduling algorithm. This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst.

Starvation problem is one of the major challenges faced by task scheduling algorithms, especially in cloud where, a task may wait for one or more of its requested resources for a very long time and thus violate the Service Level Agreement (SLA) provided by the CSP. Starvation is frequently brought on by lapses in a scheduling calculation or 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.

Many hybrids were introduced to solve the starvation problem as FCFS-RR and SRTF-RR. One interesting thing to note is that RR is always a common denominator in these hybrids due to it’s task quantum nature. Unfortunately, none of these till now solve or nearly approach to solve the problem.

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. A proper balance needs to be achieved between these two extremes.

To tackle the important starvation problem in task scheduling, this paper introduces a novel hybrid scheduling technique of SJF and RR with Dynamic Quantum, henceforth referred to as SRDQ applied through having two Queues to schedule processes for execution. The proposed algorithm is designed to be a unit based algorithm based on an effectively queuing data structure for optimizing the execution time.
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 is explained in detail.

The concepts of task scheduling and some related work are discussed followed by a detailed explanation of the proposed scheduling technique with examples of datasets and the corresponding results. The implementation in Python and results are presented and finally, conclusions and future work are discussed.

Explanation of paper

Related Work

Task scheduling algorithms in clouds vary in their technique in scheduling tasks among cloud nodes statically, dynamically, in batches or even online but eventually they are all trying to achieve the optimal distribution of tasks overs cloud nodes. In this section different task scheduling algorithms applied in cloud environment with suitable verification and different intentions will be presented and discussed in details.

Fang et al., in [13] was the first to introduce a two level 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, the scheduler assigns the adequate resources to each VM considering its load in order to achieve load balancing among VMs.
According to the authors, this task scheduling mechanism can not only meet user's requirements, but also get higher resource utilization, which was proved by simulation results in the CloudSim toolkit.

However, 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, thus showing superiority in reducing the amount of consumed power.

It did not however, consider the load and resources of the destination physical machines to which the VMs will be migrated to. Also it did not mention anything about the rules to follow when the physical machine should retire and force 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 suffered from increasing makespan i.e total length of the schedule.

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 miss 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 also uses 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 of 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 done using two theoretical (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 doesn’t consider 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.

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, is that 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 such 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.

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 that were conducted indicated that the proposed algorithm has outperformed both greedy algorithm and firefly algorithm in make span reduction.

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 makespan, based on Job execution within specified time interval.

The experimental results clarified that the proposed algorithm achieved better results in terms of makespan 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 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 while others have investigated task scheduling from load balancing prospective 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 solve it.

As the starvation problem is considered one of the major scheduling dilemmas, hence in this paper an attempt to overcome or partially overcome the starvation problem of long tasks though proposing a hybrid scheduling algorithm based on two traditional scheduling algorithms SJF and RR.
These two algorithms were intentionally picked to make benefit of SJF fast scheduling while solving its starvation problem using RR enhanced with dynamic quantum.

Through the proposed algorithm the ready queue is split into two sub-queues Q1 and Q2 one for short tasks while the other is for long tasks, which will be discussed in detail in next section.

SRDQ Algorithm

We are trying 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 disadvantages that lead to higher waiting times. 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 authors concentrated on calculating the optimal quantum interval at each round of RR algorithm while splitting the tasks ready queue into two subqueues Q1 for short tasks and Q2 for long tasks using the median as the threshold of the tasks length. So, the tasks longer than the median are to be inserted in Q2 while the shorter ones are to be inserted in Q1.
Tasks will be executed mutually i.e two short tasks from Q1 and one long task from Q2

will be executed in turns, 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.

The SRDQ algorithm proposed consists of following 6 steps :

1. Arrange all submitted tasks, T_i, $i = 1, 2, \ldots$, number of submitted tasks, according to their burst time.
2. Compute the median, $\tilde{q}$, 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.

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:

$$q_{ij} = \tilde{q} + \frac{\tilde{q}}{\left(B_{ij} + (-1)^{1-\propto} \cdot q_{i(j-1)}\right)^2} \quad (1)$$

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:

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

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

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

-

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~ will be updated dynamically as following :

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

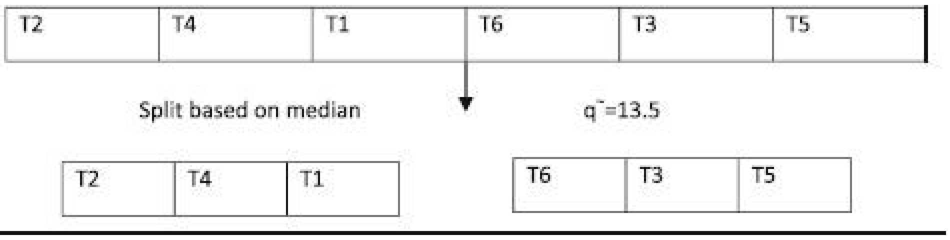
Where $B_{\text{terminated}}$ is the finished task burst time.

<u>Example</u>

The execution of tasks with arrival time and burst time as follows is shown :

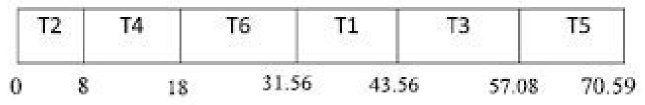
atime=[0,0,1,2,3,4]
btime=[12,8,23,10,30,15]

The median is 13.5, initial qbar value.



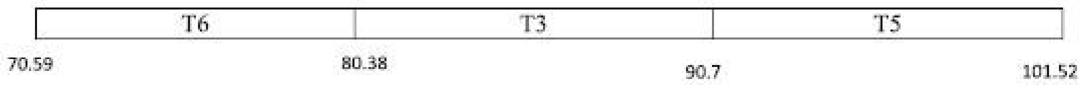
Task quantum calculations in first round

T	$B_{ij} = B_{i(j-1)} - q_{i(j-1)}$	$(B_{ij} + q_{i(j-1)})^2$ Q1, ∝= 1	$(B_{ij} - q_{i(j-1)})^2$ Q2, ∝= 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



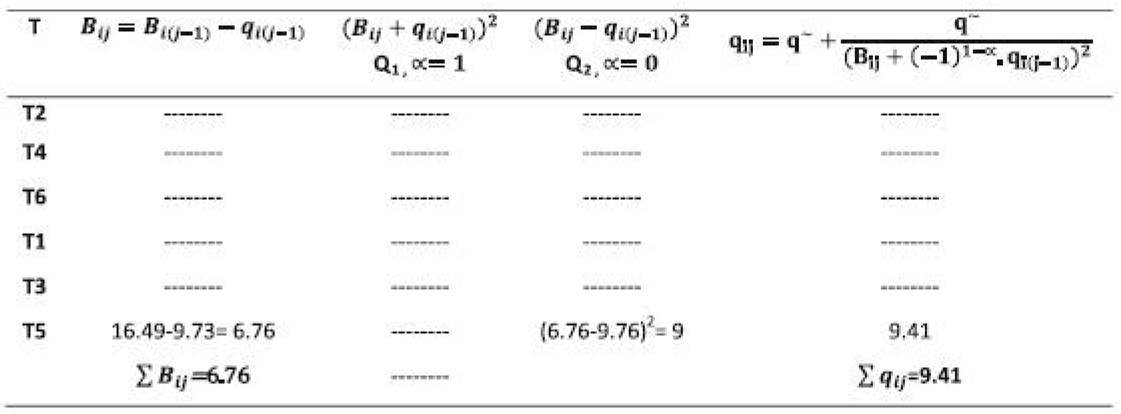
Task quantum calculations in the second round

T	$B_{ij} = B_{i(j-1)} - q_{i(j-1)}$	$(B_{ij} + q_{i(j-1)})^2$ Q1, ∝= 1	$(B_{ij} - q_{i(j-1)})^2$ Q2, ∝= 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	--------	$(1.44-13.56)^2$=146.8	q_{62} =9.79
T1	--------	--------	--------	--------
T3	23-13.52= 9.48	--------	$(9.48-13.52)^2$= 16.32	q_{32} =10.32
T5	30-13.51= 16.49	--------	$(16.49-13.51)^2$= 8.88	q_{52} =10.82
	$\sum B_{i2}$=27.41	--------		$\sum q_{i2}$=30.93

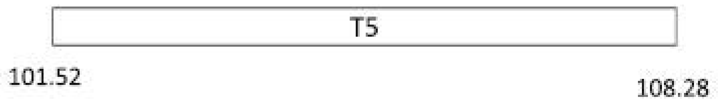


The final round of computations has only T5 left :

Task quantum calculations in the third round



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



<u>**Results**</u>

The implementation was done in Python 3.6 on a machine running Ubuntu 18.04 OS. An important assumption made in the paper is that the smallest burst time is greater than the largest arrival time of the processes.

Performance metrics used for evaluating were :

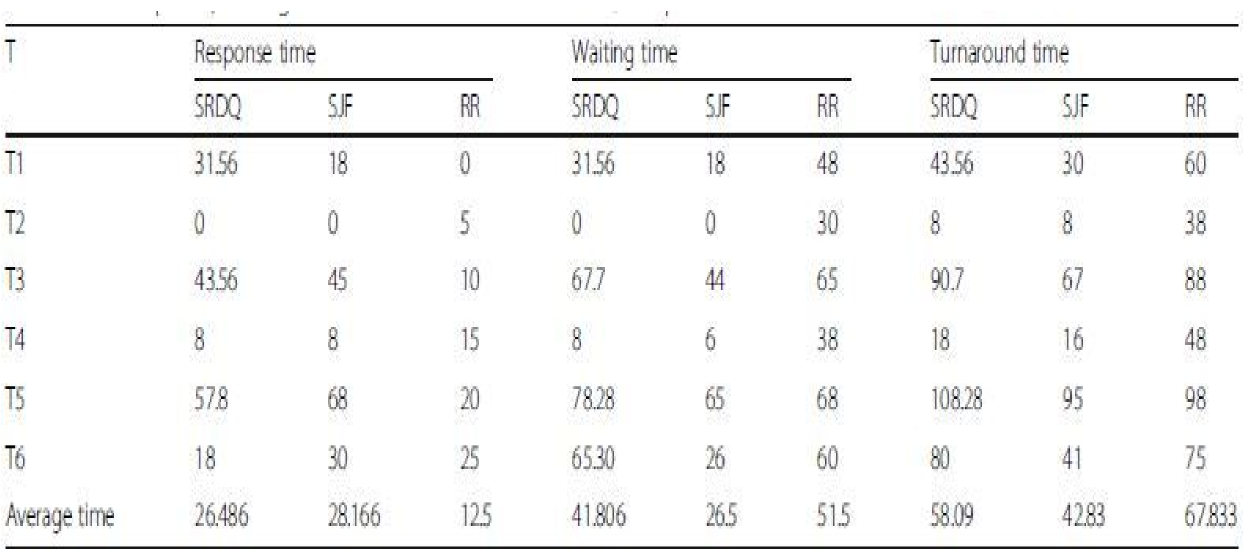
Waiting Time : Average time a process spends in the run queue.

Response Time **:** Average time elapsed from when a process is submitted until useful output is obtained.

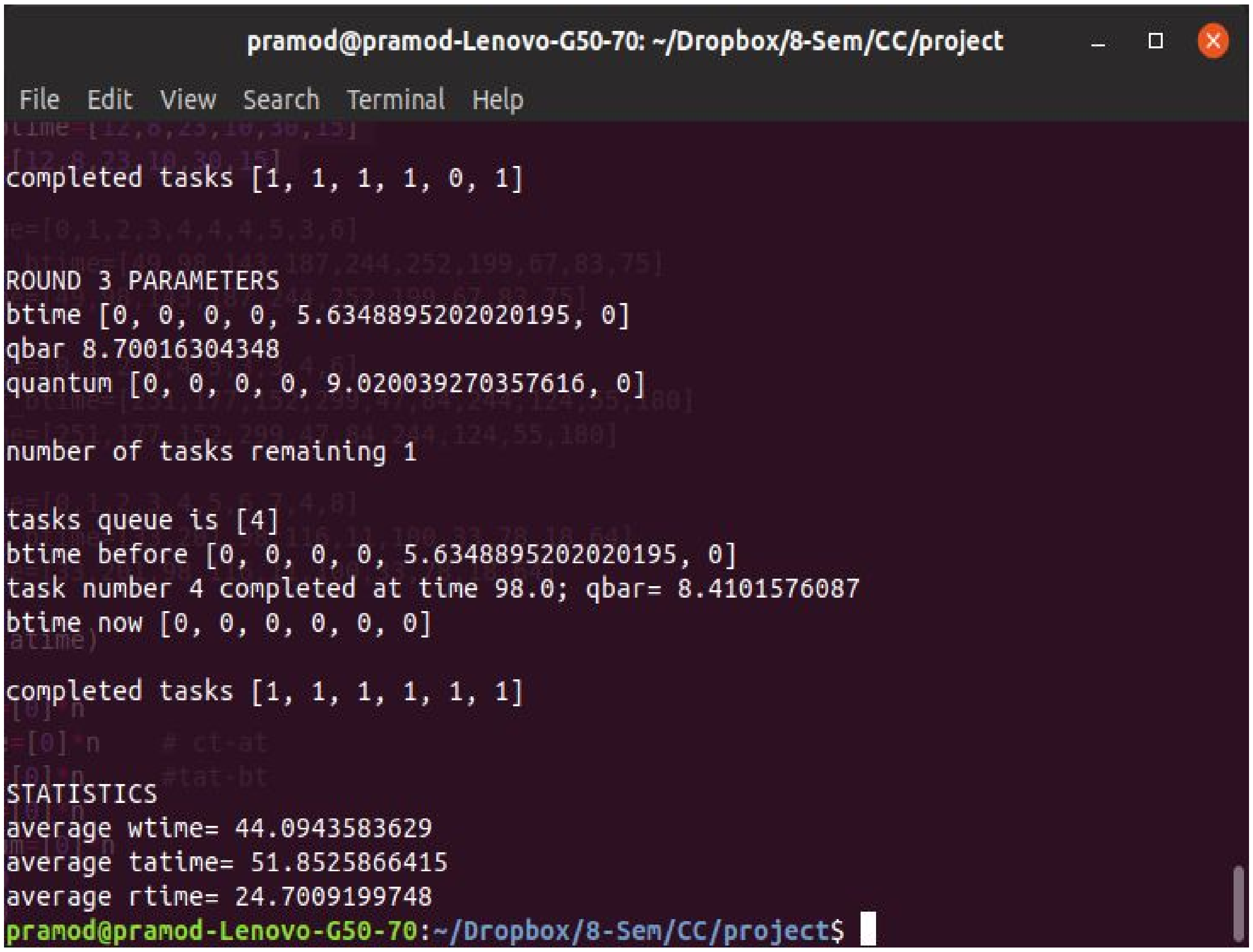
Turnaround Time : Average time elapsed from when a process is submitted to when it has completed.

The four datasets given in the paper were used. The SRDQ was used on the following four datasets and the following results were obtained :

1) atime=[0,0,1,2,3,4]
   btime=[12,8,23,10,30,15]



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



```
completed tasks [1, 1, 1, 1, 0, 1]

ROUND 3 PARAMETERS
btime [0, 0, 0, 0, 5.6348895202020195, 0]
qbar 8.70016304348
quantum [0, 0, 0, 0, 9.020039270357616, 0]

number of tasks remaining 1

tasks queue is [4]
btime before [0, 0, 0, 0, 5.6348895202020195, 0]
task number 4 completed at time 98.0; qbar= 8.4101576087
btime now [0, 0, 0, 0, 0, 0]

completed tasks [1, 1, 1, 1, 1, 1]

STATISTICS
average wtime= 44.0943583629
average tatime= 51.8525866415
average rtime= 24.7009199748
pramod@pramod-Lenovo-G50-70:~/Dropbox/8-Sem/CC/project$
```

The computations in 2nd and final round were shown in earlier tables and the updation of qbar is shown below :

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

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

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

- After finishing T1, $B_{terminated} = 12$, $\tilde{q} = 10.62 - (10.62/12) = 9.73$

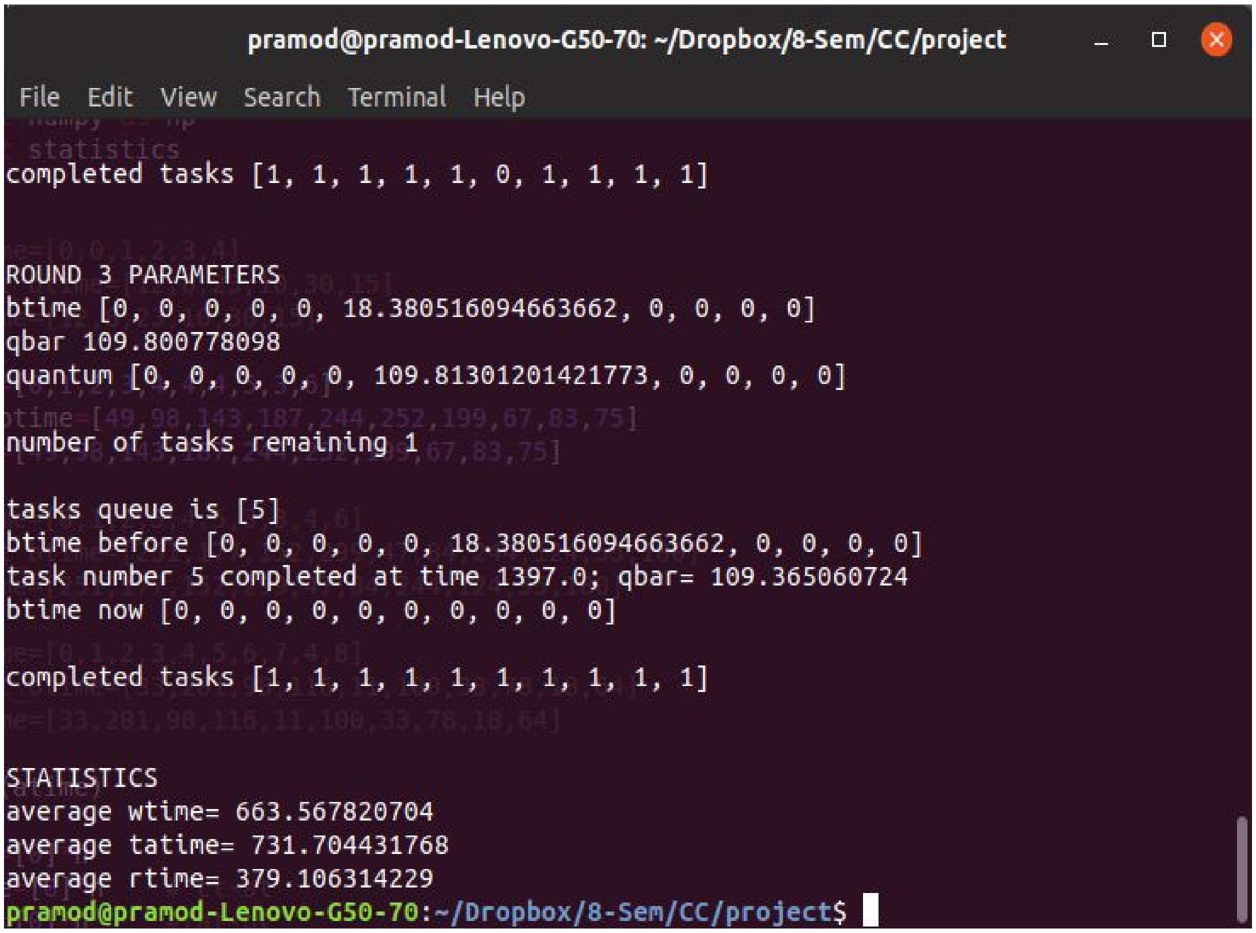
As three tasks finished in the same round, $\tilde{q}$ will be updated three times and we obtain $\tilde{q} = 9.73$ in round 2 (Table 3).

Round (3)
T6 and T3 are finished in the second round so $\tilde{q}$ will be updated as following:

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

- After finishing T6, $B_{terminated} = 15$, $\tilde{q} = 9.73 - (9.73/15) = 9.08$
- After finishing T3, $B_{terminated} = 23$, $\tilde{q} = 9.08 - (9.08/23) = 8.47$

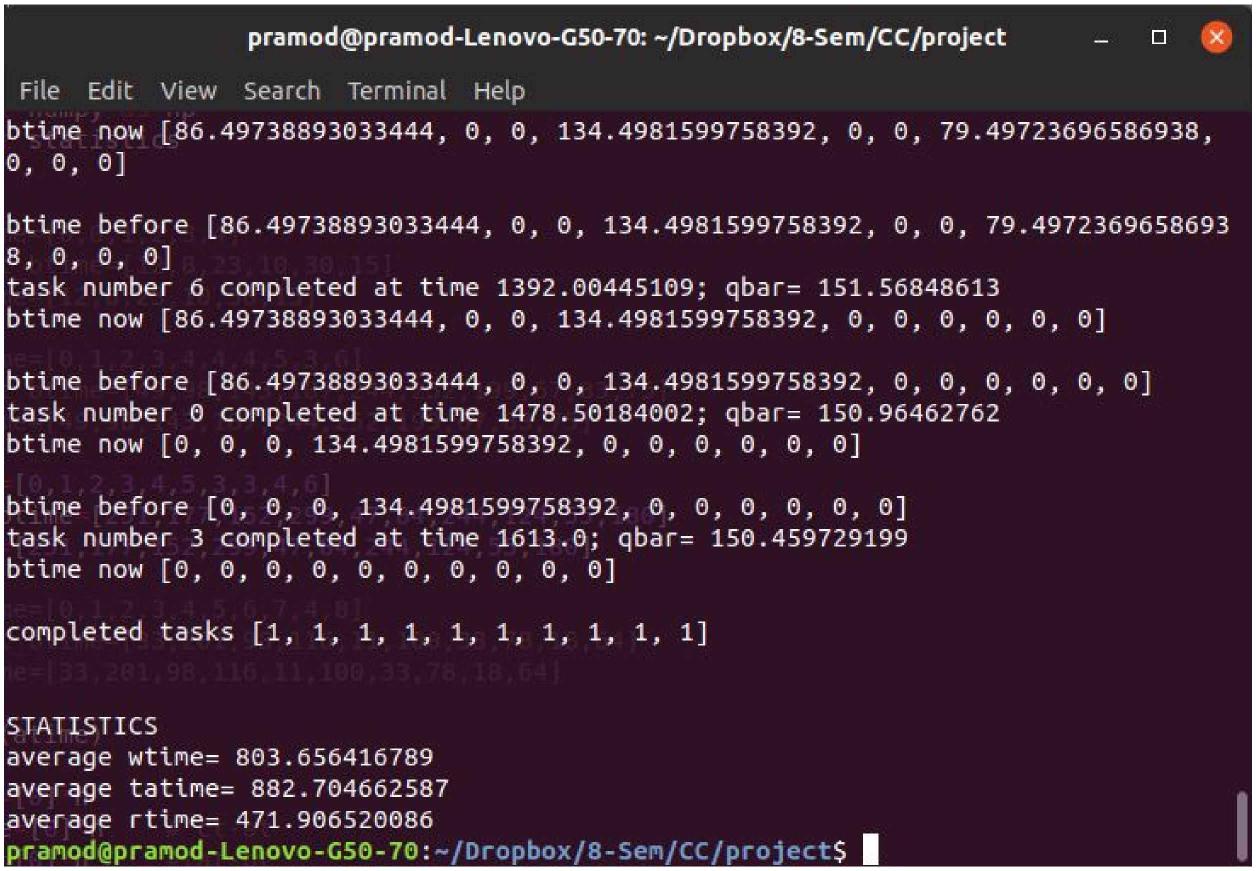
2) atime=[0,1,2,3,4,4,4,5,3,6]
btime=[49,98,143,187,244,252,199,67,83,75]



pramod@pramod-Lenovo-G50-70: ~/Dropbox/8-Sem/CC/project

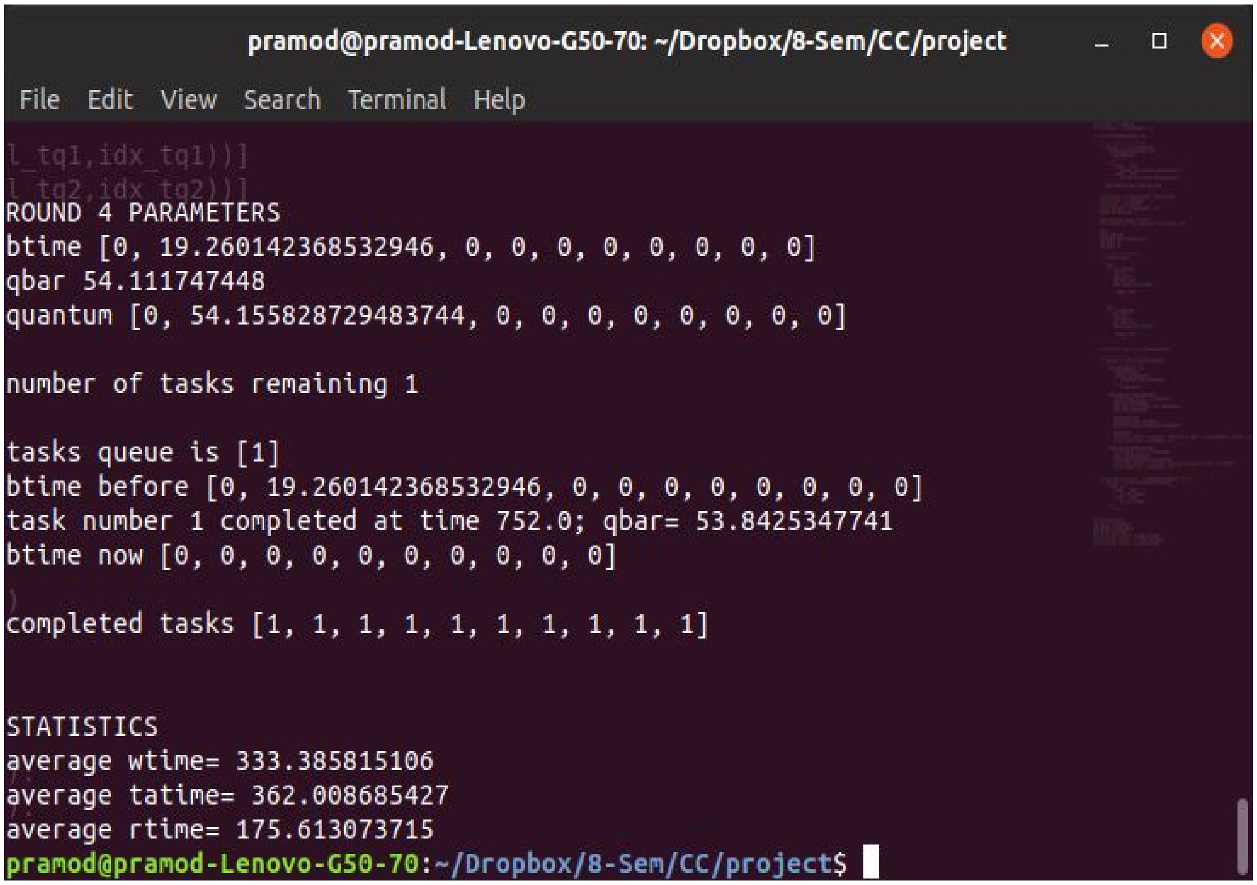
```
completed tasks [1, 1, 1, 1, 1, 0, 1, 1, 1, 1]


ROUND 3 PARAMETERS
btime [0, 0, 0, 0, 0, 18.380516094663662, 0, 0, 0, 0]
qbar 109.800778098
quantum [0, 0, 0, 0, 0, 109.81301201421773, 0, 0, 0, 0]

number of tasks remaining 1

tasks queue is [5]
btime before [0, 0, 0, 0, 0, 18.380516094663662, 0, 0, 0, 0]
task number 5 completed at time 1397.0; qbar= 109.365060724
btime now [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

completed tasks [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]


STATISTICS
average wtime= 663.567820704
average tatime= 731.704431768
average rtime= 379.106314229
pramod@pramod-Lenovo-G50-70:~/Dropbox/8-Sem/CC/project$
```

3) atime=[0,1,2,3,4,5,3,3,4,6]
btime=[251,177,152,299,47,84,244,124,55,180]


pramod@pramod-Lenovo-G50-70: ~/Dropbox/8-Sem/CC/project
File Edit View Search Terminal Help

```
btime now [86.49738893033444, 0, 0, 134.4981599758392, 0, 0, 79.49723696586938,
0, 0, 0]

btime before [86.49738893033444, 0, 0, 134.4981599758392, 0, 0, 79.4972369658693
8, 0, 0, 0]
task number 6 completed at time 1392.00445109; qbar= 151.56848613
btime now [86.49738893033444, 0, 0, 134.4981599758392, 0, 0, 0, 0, 0, 0]

btime before [86.49738893033444, 0, 0, 134.4981599758392, 0, 0, 0, 0, 0, 0]
task number 0 completed at time 1478.50184002; qbar= 150.96462762
btime now [0, 0, 0, 134.4981599758392, 0, 0, 0, 0, 0, 0]

btime before [0, 0, 0, 134.4981599758392, 0, 0, 0, 0, 0, 0]
task number 3 completed at time 1613.0; qbar= 150.459729199
btime now [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

completed tasks [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]


STATISTICS
average wtime= 803.656416789
average tatime= 882.704662587
average rtime= 471.906520086
pramod@pramod-Lenovo-G50-70:~/Dropbox/8-Sem/CC/project$
```

4) atime=[0,1,2,3,4,5,6,7,4,8]
btime=[33,201,98,116,11,100,33,78,18,64]


pramod@pramod-Lenovo-G50-70: ~/Dropbox/8-Sem/CC/project
File Edit View Search Terminal Help

```
l_tq1,idx_tq1))]
l_tq2,idx_tq2))]
ROUND 4 PARAMETERS
btime [0, 19.260142368532946, 0, 0, 0, 0, 0, 0, 0, 0]
qbar 54.111747448
quantum [0, 54.155828729483744, 0, 0, 0, 0, 0, 0, 0, 0]

number of tasks remaining 1

tasks queue is [1]
btime before [0, 19.260142368532946, 0, 0, 0, 0, 0, 0, 0, 0]
task number 1 completed at time 752.0; qbar= 53.8425347741
btime now [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

completed tasks [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]


STATISTICS
average wtime= 333.385815106
average tatime= 362.008685427
average rtime= 175.613073715
pramod@pramod-Lenovo-G50-70:~/Dropbox/8-Sem/CC/project$
```

<u>Simulation used in paper - an overview</u>

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 concerned about the low level details related to Cloud-based infrastructures and services

The processes have been distributed various cloudlets, each comprising of VMs used to run the task in a simulated setting provided by CloudSim, which has been replicated in Python for this implementation here.

Problems in the paper

The following discrepancies have been found in the paper and an e-mail regarding the same has been sent to the authors :

1) Arithmetic mistake in calculation of qbar distorts the results obtained.
2) Round Robin algorithm’s Gantt chart and implementation is wrong.
3) Time quantum used for round robin used for comparison with SRDQ is unclear.

Conclusion

Achieving optimality in scheduling tasks over computing nodes in cloud computing is the aim of all researchers interested in both scheduling and cloud efficiency. Balancing between throughput, waiting time and response time may provide a way to approach scheduling optimality but on another level it may cost long tasks starvation.

Most of the previous studies have concentrated only on one side of improving scheduling, either starvation or throughput but not both. 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.

Dividing the tasks into two such queues makes a lot of sense to avoid starvation and simultaneously complete shorter jobs quicker. The updation of quantum oftime for round robin for Q1 and Q2 is also mathematically intuitive, in the sense that larger increments are made for Q2 and smaller increments for Q1 by controlling the changes made to the denominator of the increment either positively or negatively.

The authors utilized three different datasets were utilized 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.

The implementation in Python of SRDQ algorithm yielded promising results similar to those obtained in the paper. Unfortunately, the paper needs a lot of clarity and further work due to numerous errors and missing information. Additional clarity is expected to further improve the results achieved.

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.

References

The following referenced papers have been used extensively throughout the paper :

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

Other papers referred were :

https://pdfs.semanticscholar.org/fe70/2bc2833702e5312bed4ef2894d9cbc5d4d50.pdf
http://ieeexplore.ieee.org/document/7538374

The following websites were also referred :

https://www.tutorialspoint.com/operating_system/os_process_scheduling.htm
https://www.geeksforgeeks.org/gate-notes-operating-system-process-scheduling/
https://www.geeksforgeeks.org/starvation-aging-operating-systems/