Indonesian Journal of Electrical Engineering and Computer Science
Vol. 18, No. 2, May 2020, pp. 1081~1088
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v18.i2.pp1081-1088

Optimization model for QoS based task scheduling in cloud computing environment

Sirisha Potluri, Katta Subba Rao
Department of CSE, KL University, India

Article Info

Article history:

Received Jul 26, 2019
Revised Oct 28, 2019
Accepted Nov 11, 2019

Keywords:

Cloud computing
Quality of service
Task scheduling

ABSTRACT

Shortest job first task scheduling algorithm allocates task based on the length of the task, i.e the task that will have small execution time will be scheduled first and the longer tasks will be executed later based on system availability. Min- Min algorithm will schedule short tasks parallel and long tasks will follow them. Short tasks will be executed until the system is free to schedule and execute longer tasks. Task Particle optimization model can be used for allocating the tasks in the network of cloud computing network by applying Quality of Service (QoS) to satisfy user's needs. The tasks are categorized into different groups. Every one group contains the tasks with attributes (types of users and tasks, size and latency of the task). Once the task is allocated to a particular group, scheduler starts assigning these tasks to accessible services. The proposed optimization model includes Resource and load balancing Optimization, Non-linear objective function, Resource allocation model, Queuing Cost Model, Cloud cost estimation model and Task Particle optimization model for task scheduling in cloud computing environement. The main objectives identified are as follows. To propose an efficient task scheduling algorithm which maps the tasks to resources by using a dynamic load based distributed queue for dependent tasks so as to reduce cost, execution and tardiness time and to improve resource utilization and fault tolerance. To develop a multi-objective optimization based VM consolidation technique by considering the precedence of tasks, load balancing and fault tolerance and to aim for efficient resource allocation and performance of data center operations. To achieve a better migration performance model to efficiently model the requirements of memory, networking and task scheduling. To propose a QoS based resource allocation model using fitness function to optimize execution cost, execution time, energy consumption and task rejection ratio and to increase the throughput. QoS parameters such as reliability, availability, degree of imbalance, performance and SLA violation and response time for cloud services can be used to deliver better cloud services.

Corresponding Author:

Sirisha Potluri,
Department of CSE,
KL University, Green Fields,
Vaddeswaram, Guntur, Andhra Pradesh 522502, India.
Email: sirisha.vegunta@gmail.com

1. INTRODUCTION

Cloud computing is a practise of using a network of remote servers hosted on the internet to store, manage and process data, rather than a local server or a personal computer. The main task of cloud computing is to schedule the existing tasks with specified constraints for execution. The main requirement for this is applying the Quality of Service (QoS) as per user requirements, coordinating among the QoS and equality among the tasks. To satisfy the obligations of cloud computing, many algorithms have been

enhanced. Priorities that are evaluated by the task scheduling algorithm for each task based on its attributes. Subsequently, the first tasks is scheduled which have utmost priority is the one of these algorithms. Shortest job first algorithm schedules the tasks with least execution time and the schedules longer tasks later whereas Min- min algorithm schedules short task in parallel and long tasks after them. But longer tasks will be scheduled only when the system is free for long enough to schedule and execute them.

2. RESEARCH METHOD

2.1. Shortest Job First Algorithm

Shortest Job First scheduling typically select the job with shortest execution time. This is the best approach to minimize waiting time [1]. This is used in Batch Systems. It is of two types: Non Preemptive and Pre-emptive. To successfully implement it, the burst time/duration time of the processes should be known to the processor in advance, which is practically not feasible all the time. The advantage of this algorithm is it gives better results if the jobs are available all at atime.

2.1.1. Non Preemptive SJF

If the arrival time for processes are different, which means all the processes are not available in the ready queue at time 0, and some jobs arrive after some time, in such situation, sometimes process with short burst time have to wait for the current process's execution to finish, because in Non Preemptive SJF, on arrival of a process with short duration, the existing job/process execution is not halted/stopped to execute the short job first. This leads to the problem of Starvation, where a shorter process has to wait for a long time until the current longer process gets executed. This happens if shorter jobs keep coming, but this can be solved using the concept of aging.

2.1.2. Preemptive Shortest Job First

In this algorithm, the jobs are kept in a queue whenever they enter into the system. The job with the shortest execution time is identified for the list then the existing process is preempted are taken out from execution, and the newly identified shorter job is given with high priority and then executed first. The Preemptive SJF is also known as Shortest Remaining Time First, because at any given point of time, the job with the shortest remaining time is executed first.

2.2. MAX-MIN algorithm

This algorithm works in the opposite direction of Min-Min algorithm. While Min-Min algorithm selects and assigns cloudlet to a virtual machine (vm) with minimum completion time first before considering cloudlets with maximum execution time, Max-Min does the opposite by selecting and assigning cloudlets with maximum completion time first. Max-Min algorithm gave priority to cloudlets with maximum completion time. However, its disadvantage is that, it sometimes leaves the short tasks unattended or waited for so long before execution. when we have muchmore numbers of cloudlets with maximum completion time [2-3].

2.2.1. Pseudo code for Max-Min algorithm

//Phase 1: Calculation of minimum completion time of each task

1. For all the submitted tasks (ti) in metatask (MT)
2. For all the resources R
3. Compute completion time j CTij= ETij + r
4. End of step 2 loop. j
5. End of step 1 loop.

//Phase 2: Assigning task ti

6. For each task in MT, find the task t with maximum completion time to the resource which gives minimum completion time. i
7. Assign t with maximum completion time and that resource on which it is calculated i to resource Rj
8. Remove task ti from MT. that has m minimum completion time
9. Update resource Rj ready time (rj)
10. Update completion time of all unmapped tasks in MT
11. Repeat step 6-10, until all the tasks in metatask (MT) have been mapped
12. End of step 6 loop

2.3. MIN-MIN Algorithm

Min-Min scheduling algorithm used in cloud and grid computing environments in order to minimize makespan,cost and maximize profit and resource utilization. This is done by selecting a cloudlet in the cloudlet list with the lowest execution time and assign it to virtual machine that produces its minimum completion time. This algorithm gave priority to tasks with minimum completion time. However, assigning cloudlets in this manner lead to increase of total response time of the system when cloudlets with minimum completion time are much more in number. Thus,making it inefficient.Researchers have recommended Min-Min as one of the best methods of scheduling in cloud and grid computing.These researchers have contributed tremendously by makingMin-Min an efficient task scheduling algorithm. The efficiency of this algorithm has made cloud computing acceptable in both educational institutions and industries as a preferred platform for data storing and information dissemination [4-8].

2.3.1. Pseudo Code for Min-Min Algorithm

//Phase 1: Calculation of minimum completion time of each task.

1. For all the submitted tasks (ti) in metatask (MT)
2. For all the resources R
3. Compute completion time j CTij= ETij + r
4. End of step 2 loop. j
5. End of step 1 loop.

//Phase 2: Assigning task ti

6. For each task in MT, find the task t with minimum completion time to the resource having minimum completion time. T
7. Assign t with minimum completion time and that resource on which it is calculated i to resource Rj
8. Remove task t that has m minimum completion time i
9. Update resource R from MT j ready time of r
10. Update completion time of all unmapped tasks in MT j
11. Repeat step 6-10, until all the tasks in metatask (MT) have been mapped
12. End of step 6 loop.

3. RELATED WORK

To initiate QoS in cloud computing environments, task scheduling algorithms are used For computing the priority of tasks, these scheduling algorithms are used the following four Standardized/Normalized attributes of tasks are considered :

1. TaskUsersType (UT): demonstrates the kind of users (User class-A, B & C).
2. TaskPriorExp (PT): demonstrates the probable task's priority for scheduling (low, medium, high, vital priority).
3. TaskLength (TL): demonstrates the load or length of the tasks (Normally loaded, lengthy).
4. LatencyTask (LT): demonstrates task's latency.

For evaluating the priorities of each one task, the task scheduling algorithm uses each attribute's weight and calculates the priority using the following formula:

$$P(i) = \alpha * NUT + \beta * NPT + \gamma * NTL + w * NLT$$

Where,

Standardized values for the attributes UT, PT, TL, and LT are respectively NUT, NPT, NTL, and NLT. The parameters a, b, y, and x indicates weights of the attributes and equivalent to 0.4, 0.3, 0.2 and 0.1 correspondingly.The tasks are arranged in the order of priority. Every task from the sorted task queue was scheduled against the required services which can comPlete the task at the earliest.Shortest job first algorithm, which depends on task's execution time to schedule task . According to this, firstly the task with minimum time for execution might be scheduled and later the tasks with longer execution time will be scheduled.Min- Min algorithm will be similar to shortest job first algorithm, to but attain the minimum cost, make_span and as well to get better the communication or computation ratio in cloud computing environment, an improved scheduling algorithm bases on cost-based were used. In this algorithm when the task is scheduled on a particular resource then profit and cost parameters are calculated.. Subsequently, depending on priorities of each task's, the tasks are dispersed into 3 groups namely high, medium and low. Then, to execute tasks in each one group, job grouping algorithm was used.For designing a system to reduce

the processing time for scheduling tasks, a divisible load theory (DLT) is used in cloud computing environment. This can be made with identical processors and draw from a bunged form way out for the load fractions to be allocate to every processors. To relate requests through diverse levels of non-functional requirements, a cloud brokering algorithm were used. Because of the self-motivated nature of clouds, a solution with optimization of Self-adaptive QoS is desirable in cloud computing environments. An architectural approach with decentralization which has dynamic optimization of QoS vital to the alteration was anticipated in , since optimizing QoS adaptively for Dynamic Data Driven Application System (DDDAS) a cloud basing application is difficult.To make a decision of what resources must be rented as of the public cloud and aggregated to the private cloud, and adequate processing power being granted for executing the workflow contained by a particular execution_time, a Hybrid Cloud Optimized Cost scheduling (HCOC) algorithm is used.To allocate the resources for the tasks arrived at uncertain runtime interval, dynamic scheduling algorithms are used. Meaning that, it is as tough as numerous tasks at the same time coming.Genetic Algorithms are used to avoid such scheduling difficulties.

3.1. Makespan and Cost

The first approximation technique we have seen was through rounding and relaxation of IPs and LPs. In this section, we'll see an example of a greedy algorithm that guarantees a constant factor approximation ratio. The problem we are interested is the Minimum Makespan Scheduling Problem, defined as follows: Suppose we have n jobs each of which take time ti to process, and m identical machines on which we schedule the jobs. Jobs cannot be split between machines. For a given scheduling, let Aj be the set of jobs assigned to machine j. Let Tj = P i∈Aj ti be the load of machine j. The minimum makespan scheduling problem asks for an assignment of jobs to machines that minimizes the makespan, where the makespan is defined as the maximum load over all machines (i.e. max j Tj). The greedy algorithm we came up with in class was to sort the jobs so that t1 ≥ t2 ≥ ... ≥ tn, and iteratively allocate the next job to the machine with the least load:

Algorithm 1 Greedy Approximation Algorithm for Job Scheduling on identical Machines:

1: Aj ← ∅,Tj ← 0, ∀j
2: for i = 1 ... n do
3: j ← min k Tk
4: Aj = Aj ∪ {i}
5: Tj = Tj + ti
6: end fo

4. PROPOSED TASK SCHEDULING ALGORITHM

Compared with TSA, the Ṁin_Ṁin algorithm attains lesser execution time. Subsequently, the task with least time for execution would be scheduled foremost in the selected group [9-13]. Let, n, m be the number of independent tasks and services respectively. n, m is the input of the group based task scheduling algorithm Algorithm. Every task contains four attributes:

1. TaskUsersType: give you an idea regarding the kind of users (user class - A, B & C).
2. TaskPriorExp: give you an idea regarding the probable priority of tasks scheduled (low, medium, high, vital priority).
3. TaskLength: characterizes the load or length of tasks (Normally loaded, lengthy).
4. LatencyTask: give you an idea regarding the task's latency.

The group based task task scheduling algorithm has five Classes:

1. C_UrgentUser&Ţask: contains tasks_with_user be owned by group A, and anticipated scheduling priority of the task is vital / urgent.
2. C_UrgentUser: contains tasks_with_user be owned by group A.
3. C_UrgentTask: contains tasks_with anticipated scheduling priority of task is vital / urgent.
4. C_LongTask: contains lengthy tasks.
5. C_NormalTask: contains all left over tasks.

The sorting of the priority of 5 classes in descending order is C_NormalTask, C_LongTask, C_UrgentTask, C_UrgentUser, C_UrgentUser&Task. It means the C_UrgentUser&Task class tasks must be firstly scheduled earlier than tasks contained in C_UrgentUser class and so on. The C_NormalTask class contains the remaining tasks with normal priority and may be scheduled after the previous four class tasks completion. MCT matrix is n ×m matrix which provisions the assessment of anticipated time for completion of all tasks on every one of services (Initialized Minimum Competition Time). The row of MCT matrix represents the tasks and the numbers of rows are equivalent to the number of tasks (n). Whereas, the columns of MCT matrix represents the services and the numbers of columns are equivalent to the number of services

(m). Let, MCT (i, j) is the minimum completion time that service j needs to execute task i. Though the type of task is longer or normal, the MCT matrix is initialized with random numbers. The range of time is in the normal tasks needs to be smaller in longer task than the range of random time using MCT matrix MCT (i, j). The numbers of tasks, services allocated to these tasks and time of execution that the services required to execute these tasks are saved in the mapping list. Mapping list is a matrix used for storing the above said information and it is treated as the output of the algorithm. Using this, performance metrics is computed and that are required to assess the algorithm. Thereafter, initialize Mapping_list matrix and MCT matrix. Tasks are distributed into 5 classes. Whenever a new task enters the system for execution, first it is required to make a decision to which class it belongs. This decision was on the basis of its attribute values. After that, this task is placed in to the determined class.

The proposed optimization model has to be designed which includes Resource and load balancing Optimization, Non-linear objective function, Resource allocation model, Queuing Cost Model, Cloud cost estimation model and Task Particle optimization model for task scheduling in cloud computing environement. The overall architecture of the proposed model is as shown in Figure 1.

The main objectives of the proposed optimization model are identified as follows [14-21].

a. To propose an efficient task scheduling algorithm which maps the tasks to resources by using a dynamic load based distributed queue for dependent tasks so as to reduce cost, execution and tardiness time and to improve resource utilization and fault tolerance.
b. To develop a multi-objective optimization based VM consolidation technique by considering the precedence of tasks, load balancing and fault tolerance and to aim for efficient resource allocation and performance of data center operations.
c. To achieve a better migration performance model to efficiently model the requirements of memory, networking and task scheduling.
d. To propose a QoS based resource allocation model using fitness function to optimize execution cost, execution time, energy consumption and task rejection ratio and to increase the throughput. QoS parameters such as reliability, availability, degree of imbalance, performance and SLA violation and response time for cloud services can be used to deliver better cloud services.

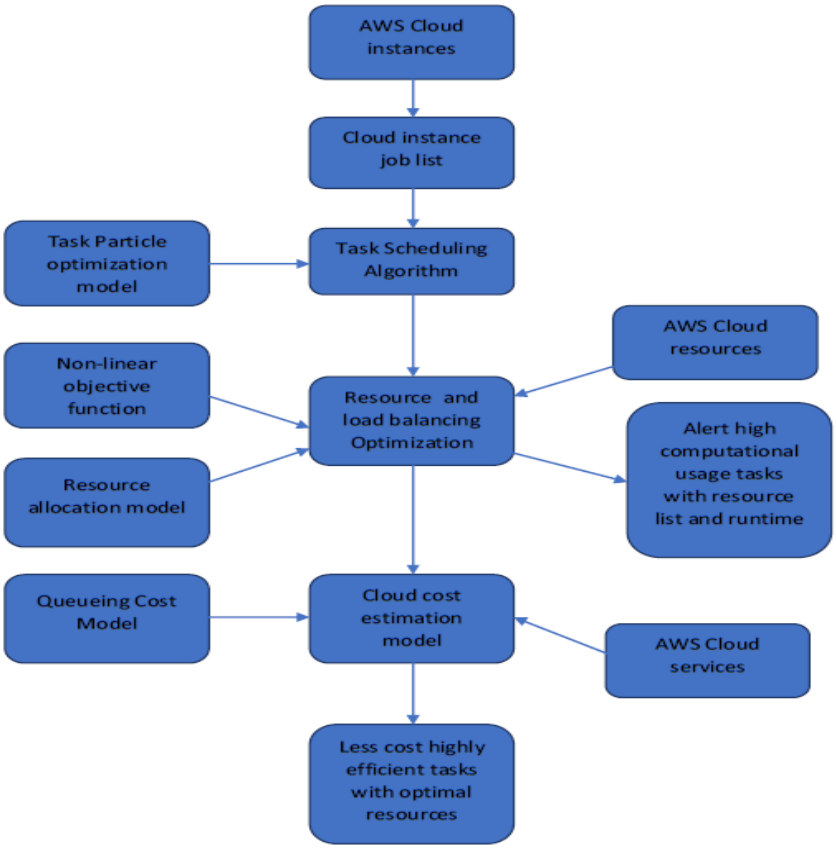

Figure 1. Proposed Task Particle optimization model

AWS cloud instances: A set of AWS cloud instances will be taken to create a job list.

Cloud instance job list: A set of jobs in a list will be given as input to task scheduling algorithm.

Task particle optimization model: This is the optimization model which we are going to use as schedule the given jobs using objective functions.

Resource and load balancing optimization: This module makes use of non-linear objective function, resource allocation model and AWS cloud resources to schedule the given tasks.

Cloud cost estimation model: This model makes use of AWS cloud services and queueing cost model to estimate the cost of the proposed model.

5. RESULTS AND ANALYSIS

Efficient scheduling algorithm can yield more desirable services to users and increase the performance provided by cloud environment. The main improvement of this task scheduling in a distributed cloud environment is to reduce the resource response time and execution time of tasks and to maximize the resource utilization. In this paper, a study related to different existing task scheduling algorithms in a cloud environment has been presented. A short description of each algorithm methodology has been presented and most algorithms consider on one or two parameters. More satisfactory results can be achieved by adding more metrics to existing algorithms. The major problem in task scheduling is load balancing, response time, resource utilization and memory storage. Efficient scheduling algorithm can be achieved by combining different parameters to existing algorithms which will improve their overall performance of cloud environment. Shortest job first for dependent tasks scheduling does the work in average time according to our implementation it took approximately 3 sec to execute 40 independent tasks. But for Dependent task scheduling it took 2.25 sec for 7 dependent tasks. Which is longer than time taken for independent tasks as shown in Figures 2 and 3.

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
0	SUCCESS	2	0	2.12	0.1	2.22
1	SUCCESS	2	0	1.17	2.22	3.39
3	SUCCESS	2	0	1.64	3.39	5.03
5	SUCCESS	2	0	1.16	5.03	6.19
2	SUCCESS	2	0	1.41	6.19	7.59
4	SUCCESS	2	0	2.21	7.59	9.81
6	SUCCESS	2	0	2.25	9.81	12.05

*****Datacenter: Datacenter_0*****

Figure 2. Result analysis of dependent task scheduling

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
22	SUCCESS	2	0	1.03	0.1	1.13
36	SUCCESS	2	0	1.15	1.13	2.27
33	SUCCESS	2	0	1.2	2.27	3.48
16	SUCCESS	2	0	1.24	3.48	4.72
1	SUCCESS	2	0	1.28	4.72	5.99
38	SUCCESS	2	0	1.3	5.99	7.29
0	SUCCESS	2	0	1.34	7.29	8.64
2	SUCCESS	2	0	1.45	8.64	10.09
5	SUCCESS	2	0	1.48	10.09	11.57
23	SUCCESS	2	0	1.49	11.57	13.07
19	SUCCESS	2	0	1.58	13.07	14.65
39	SUCCESS	2	0	1.59	14.65	16.23
10	SUCCESS	2	0	1.59	16.23	17.83
9	SUCCESS	2	0	1.6	17.83	19.43
32	SUCCESS	2	0	1.68	19.43	21.11
31	SUCCESS	2	0	1.68	21.11	22.79
24	SUCCESS	2	0	1.77	22.79	24.57
12	SUCCESS	2	0	1.97	24.57	26.54
35	SUCCESS	2	0	1.97	26.54	28.51
27	SUCCESS	2	0	1.99	28.51	30.5
15	SUCCESS	2	0	2.09	30.5	32.59
20	SUCCESS	2	0	2.16	32.59	34.74
30	SUCCESS	2	0	2.24	34.74	36.99
4	SUCCESS	2	0	2.3	36.99	39.29
37	SUCCESS	2	0	2.42	39.29	41.71
6	SUCCESS	2	0	2.43	41.71	44.14
25	SUCCESS	2	0	2.52	44.14	46.66
18	SUCCESS	2	0	2.55	46.66	49.2
14	SUCCESS	2	0	2.61	49.2	51.81
3	SUCCESS	2	0	2.62	51.81	54.43
26	SUCCESS	2	0	2.71	54.43	57.14

Figure 3. Result analysis of independent task scheduling

It seems that the Max-Min outperforms the Min-Min when number of large sized tasks is more than the short length task. But when short length tasks outnumber the long length task, Min-Min can be better choice.

6. CONCLUSION

Efficient scheduling algorithm can yield more desirable services to users and increase the performance provided by cloud environment. An efficient algorithm can be developed which can efficiently maps the tasks to resources by using a dynamic load based distributed queue for dependent tasks so as to reduce cost, execution and tardiness time and to improve resource utilization and fault tolerance. We can incorporate a multi-objective optimization based VM consolidation technique by considering the precedence of tasks, load balancing and fault tolerance and so as to aim for efficient resource allocation and performance of data center operations. Using this technique we can achieve a better migration performance model to efficiently model the requirements of memory, networking and task scheduling. This model can be served as a QoS based resource allocation model using fitness function to optimize execution cost, execution time, energy consumption and task rejection ratio and will increase the throughput. QoS parameters such as reliability, availability, degree of imbalance, performance and SLA violation and response time for cloud services can be used in this model to deliver better cloud services.

REFERENCES

[1] M.A. Alworafi, A. Dhari, A. Asma, A. Hashmi, A.B. Darem, Suresha, "*An Improved SJF Scheduling Algorithm in Cloud Computing Environment*", *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, 2017.

[2] N.Sharma, S.Tyagi, "A Comparative Analysis of Min-Min and Max-Min Algorithms based on the Makespan Parameter", *International Journal of Advanced Research in Computer Science*, Volume 8, No. 3, March-April 2017, pp:1038-1041.

[3] B.H.Malik, M.Amir, B.Mazhar, S.Ali, R.Jalil, J.Khalid, "Comparison of Task Scheduling Algorithms in Cloud Environment", *(IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 9, No. 5, 2018, pp:384-390.

[4] H. E. Xiaoshan, et al., "QoS Guided Min-Min Heuristic for Grid Task Scheduling," *Journal of Computer Science and Technology*, Vol/issue: 18(4), pp. 442–451, 2003.

[5] S. Anousha, M. Ahmadi, "*An Improved Min-Min Task Scheduling Algorithm in Grid Computing*", International Conference on Grid and Pervasive Computing, pp. 103-113, May 2013.

[6] L. Singh, et al., "An Improved Min-Min Task Scheduling Algorithm with Grid Utilization and Minimized Makespan", *International Journal of Computers & Technology*, Vol.14, No. 8, 2015, pp. 5960-5966.

[7] S. Anousha, et al., "*An Improved Min-Min Task Scheduling Algorithm in Grid Computing*", *International Conference on Grid and Pervasive Computing*, pp. 103-113, 2013.

[8] T. Kokilavani, D.I. G.Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", *International Journal of Computer Applications*, Vol. 20, No. 2, pp. 43-49, 2011.

[9] H.G.E.D.H. Ali, et al., "Grouped tasks scheduling algorithm based on QoS in cloud computing network", *Egyptian Informatics Journal*, (2017) 18, pp. 11–19.

[10] Xhafa, F., Barolli, L., Durresi, A., "*Immediate mode scheduling of independent jobs in computational grids"*, 21st International Conference on Advanced Networking and Applications (AINA 2007) (2007).

[11] Liu, Q., Liao, Y, "*Grouping-based fine-grained job scheduling in grid computing*", First International Workshop on Education Technology and Computer Science, pp. 556-559 (2009).

[12] Mishra, M.K., Mohanty, P., Mund, G.B, "*A modified grouping-based job scheduling in computational grid"*, International Conference on Current Trends in Technology, Nuicone, pp. 1-6 (2011).

[13] F. Dong, J. Luo, L. Gao, and L. Ge, "*A Grid Task Scheduling Algorithm Based on QoS Priority Grouping*", In the Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), IEEE, 2006.

[14] EnricoBarbierato, MarcoGribaudo, MauroIacono, AgnieszkaJakóbik, "Exploiting CloudSim in a multiformalism modeling approach for cloud based systems", *Simulation Modelling Practice and Theory*, Volume 93, pp. 133-147(2019).

[15] David A.Monge, ElinaPacini, CristianMateos, CarlosGarcía Garino, "Meta-heuristic based autoscaling of cloud-based parameter sweep experiments with unreliable virtual machines instances", *Computers & Electrical Engineering*, Volume 69, pp. 364-377 (2018).

[16] NajmeMansouri, BehnamMohammad Hasani Zade, Mohammad MasoudJavidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory", *Computers & Industrial Engineering*, Volume 130, pp. 597-633 (2019).

[17] BelaShrimali,HirenPatel, "Multi-objective optimization oriented policy for performance and energy efficient resource allocation in Cloud environment", *Journal of King Saud University-Computer and Information Sciences*, Available online 6 December 2017.

[18] Muhammad Zakarya, LeeGillam, "Modelling resource heterogeneities in cloud simulations and quantifying their accuracy", *Simulation Modelling Practice and Theory*, Volume 94, pp. 43-65 (2019).

[19] Taskeen Zaidi, Rampratap Rampratap, "Virtual Machine Allocation Policy in Cloud Computing Environment using CloudSim", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 8, No. 1, pp. 344-354 (2018).
[20] Sirisha Potluri, K.Subba Rao, "Quality of Service based Task Scheduling Algorithms in Cloud Computing", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 7, No. 2, pp. 1088-1095(2017).
[21] S. Pal, et al., "Efficient Architectural Framework for Cloud Computing," *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, vol/issue: 1(2), pp. 66-73 (2012).

BIOGRAPHIES OF AUTHORS

Sirisha Potluri
Research Scholar,
Department of CSE,
KL University,
Green Fields, Vaddeswaram, Guntur, Andhra Pradesh 522502, India,
Email: sirisha.vegunta@gmail.com

Dr. Katta Subba Rao
Professor,
Department of CSE,
KL University,
Green Fields, Vaddeswaram, Guntur, Andhra Pradesh 522502,India,
Email:subbarao_cse@kluniversity.in