Egyptian Informatics Journal (2017) **18**, 11–19

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

FULL-LENGTH ARTICLE

Grouped tasks scheduling algorithm based on QoS in cloud computing network

Hend Gamal El Din Hassan Ali *, **Imane Aly Saroit, Amira Mohamed Kotb**

Department of Information Technology, Faculty of Computer and Information, Cairo University, Egypt

Received 9 March 2016; revised 17 July 2016; accepted 25 July 2016
Available online 23 August 2016

KEYWORDS

Cloud computing;
Min-Min algorithm;
QoS;
Task scheduling algorithm

Abstract This paper proposes the grouped tasks scheduling (GTS) algorithm that is used to schedule tasks in cloud computing network by applying quality of service to satisfy user's needs. The proposed algorithm distributes tasks into five categories; each category has tasks with similar attributes (user type, task type, task size, and task latency). After adding tasks into right category, it starts scheduling these tasks into available services. Scheduling is done in two steps: first step is deciding which category will be scheduled first. This depends on the attributes of the tasks that belong to each category so the category that has tasks with high value of attributes will be scheduled first. Second step is deciding which task inside the chosen category will be scheduled first. This depends on the execution time of task so the task that has minimum execution time will be scheduled first.

1. Introduction

Cloud computing provides shared computing and storage resources, and also provides services and information to users over the Internet based on their demands using variety of applications [1]. The requirement in cloud computing environment is scheduling the current jobs/tasks to be executed with the given constraints. The meaning of constrains here is applying QoS that users need and balancing between these QoS and fairness among the tasks. Many algorithms are improved to satisfy the requirement of cloud computing. One of these algorithms is task scheduling algorithm that calculates priority to each task depending on its attributes and then schedules first tasks that have high priority [2]. This paper introduces a new algorithm named grouped tasks scheduling (GTS) algorithm which is based on the procedure of improved cost-based algorithm, TS algorithm and Min-Min algorithm. GTS algorithm uses the same way of calculating priority to tasks in TS algorithm to calculate priority to each category. However GTS algorithm uses the procedure of Min-Min algorithm to schedule tasks inside each category [3]. The rest of the paper is organized as follows: Section 2 gives the related work, Section 3 declares the proposed algorithm, Section 4 explains the simulation analysis, Section 5 illustrates the performance of proposed algorithm, and Section 6 gives the conclusion and future work.

* Corresponding author.
E-mail address: hendgamal@fci-cu.edu.eg (H. Gamal El Din Hassan Ali).
Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.eij.2016.07.002

2. State of art

Task scheduling algorithm is used to apply QoS in cloud computing [2]. This algorithm computes the priority of tasks according to four normalized attributes of tasks which are as follows:

1. TUserType (UT): shows the type of users (class A, class B, and class C).
2. TpriorExp(PT): shows the expected scheduled priority of tasks (urgent, high, medium, low priority).
3. TL: defines the length or load of tasks (normal, long).
4. LT: shows the latency of tasks.

TS algorithm assigns each attribute weight to calculate priorities of each task using this formula:

$$p(i) = \alpha * \mathrm{NUT} + \beta * \mathrm{NPT} + \gamma * \mathrm{NTL} + \omega * \mathrm{NLT} \quad (1)$$

where NUT, NPT, NTL, and NLT are normalized values of the four attributes UT, PT, TL, and LT respectively and α, β, y, and ω are the weights of them equal to 0.4, 0.3, 0.2, and 0.1 respectively [2].

Then tasks are sorted by priority and each task is scheduled onto service which can complete the task as soon as possible according to the sorted task queue.

Min-Min algorithm can be used in cloud computing. Min-Min algorithm depends on execution time for scheduling tasks. Tasks with minimum execution time will be scheduled first. Tasks with long execution time have high delay [3].

Improved cost-based scheduling algorithm is used in cloud computing to achieve minimum make span and cost and also to improve the computation/communication ratio [4]. This algorithm calculates priorities of tasks based on resources cost and profit. Then tasks are distributed into three groups (low, medium, and high) depending on their priorities and use job grouping algorithm to execute tasks in each group.

In [5], it is clear that divisible load theory (DLT) is used to design way to minimize the processing time for scheduling tasks in cloud computing. This is done by using homogeneous processors and deriving a closed form solution for the load fractions to be assigned to each processor.

Cloud brokering algorithm was also used to apply requests with different levels of non-functional requirements [6]. This is done to the private or public resources, with the aim to maximize user satisfaction and broker's revenues.

Self-adaptive QoS optimization solutions are needed in cloud computing because of the dynamic natural of cloud. Because adaptively optimizing QoS for DDDAS (Dynamic Data Driven Application System)-cloud based application is complex, so the decentralized architectural style which has dynamic QoS optimization central to adaptation was proposed in [7].

The Hybrid Cloud Optimized Cost scheduling (HCOC) algorithm is used to decide which resources should be leased from the public cloud and aggregated to the private cloud, to provide enough processing power to execute a workflow within a given execution time in [8].

Dynamic scheduling is that the task arrived in uncertain run time and resources are allocated to it. That is hard as several tasks arrive at the same time. To avoid this scheduling problem Genetic Algorithm is used in [9].

3. Proposed GTS algorithm

TS algorithm achieves low latency compared with Min-Min algorithm and also Min-Min algorithm achieves low execution time compared with TS algorithm. So the proposed algorithm aims to achieve low latency with minimum execution time.

The GTS algorithm uses the idea of grouped tasks present in improved cost-based algorithm to apply QoS present in TS algorithm and then uses Min-Min algorithm to schedule tasks inside each group.

The main idea of GTS algorithm is to divide all tasks into categories/classes based on its attributes. The attributes of tasks are used as explained in TS algorithm. Each category will have tasks with similar attributes. These categories will be ordered to schedule based on weights that are given to attributes of tasks in TS algorithm. In this case the categories are subject to the scheduling, while tasks are not. The first scheduled category will have tasks with high value of attributes/high priority than other categories. Then in the chosen category the task with minimum execution time will be scheduled first.

The input of GTS algorithm is number of independent tasks n and number of services m.

Each task has four attributes:

1. TUserType (UT): shows the type of users (class A, class B, and class C).
2. TpriorExp(PT): shows the expected scheduled priority of tasks (urgent, high, medium, and low priority).
3. TL: defines the length or load of tasks (normal, long).
4. LT: shows the latency of tasks.

GTS algorithm has five categories:

1. CUrgentUser&Task: includes tasks with user belong to class A and expected scheduled priority of task is urgent.
2. CUrgentUser: includes tasks with user belonging to class A.
3. CUrgentTask: includes tasks with expected scheduled priority of task is urgent.
4. CLongTask: includes long tasks.
5. CNormalTask: includes all remaining tasks.

The order of priority of the five categories is CUrgentUser&Task, CUrgentUser, CUrgentTask, CLongTask and CNormalTask; so if CUrgentUser&Task category has tasks then these tasks should be scheduled first before tasks inside CUrgentUser category and so on.

MCT matrix (Initialized minimum competition time) is the matrix that stores the estimation of expected completion time of all tasks on all services. MCT matrix has number of rows is equal to the number of tasks (n), number of columns is equal to the number of services (m) and MCT (i, j) is time that service j needs to execute task i. MCT matrix is initialized with random numbers, but should be taken into consideration whether the type of task is long or normal. Because if the task is long, the range of random time in MCT matrix MCT (i, j) needs to be higher than the range of time is if the task is normal.

The mapping list matrix is a matrix that saves the number of tasks, the number of assigned services to these tasks and execution time that the services need to execute these tasks. Mapping list matrix is considered the output of the algorithm.

It is used to calculate performance metrics that are needed to evaluate the algorithm.

Fig. 1 shows the flowchart illustrating the steps of the GTS algorithm. After defining the input of algorithm, initializing MCT and mapping list matrix and distributing tasks into five classes, will start scheduling. If there is a new task need to execute, first it is needed to define which category it belongs based on its attributes and then this task is added to the decided category.

4. Simulation analysis

As explained proposed GTS algorithm used Min-Min algorithm to schedule tasks inside chosen category, while calculation of priority assigned to tasks in TS algorithm is used to decide which category in GTS algorithm will be scheduled first. To evaluate the behavior of GTS algorithm and compare

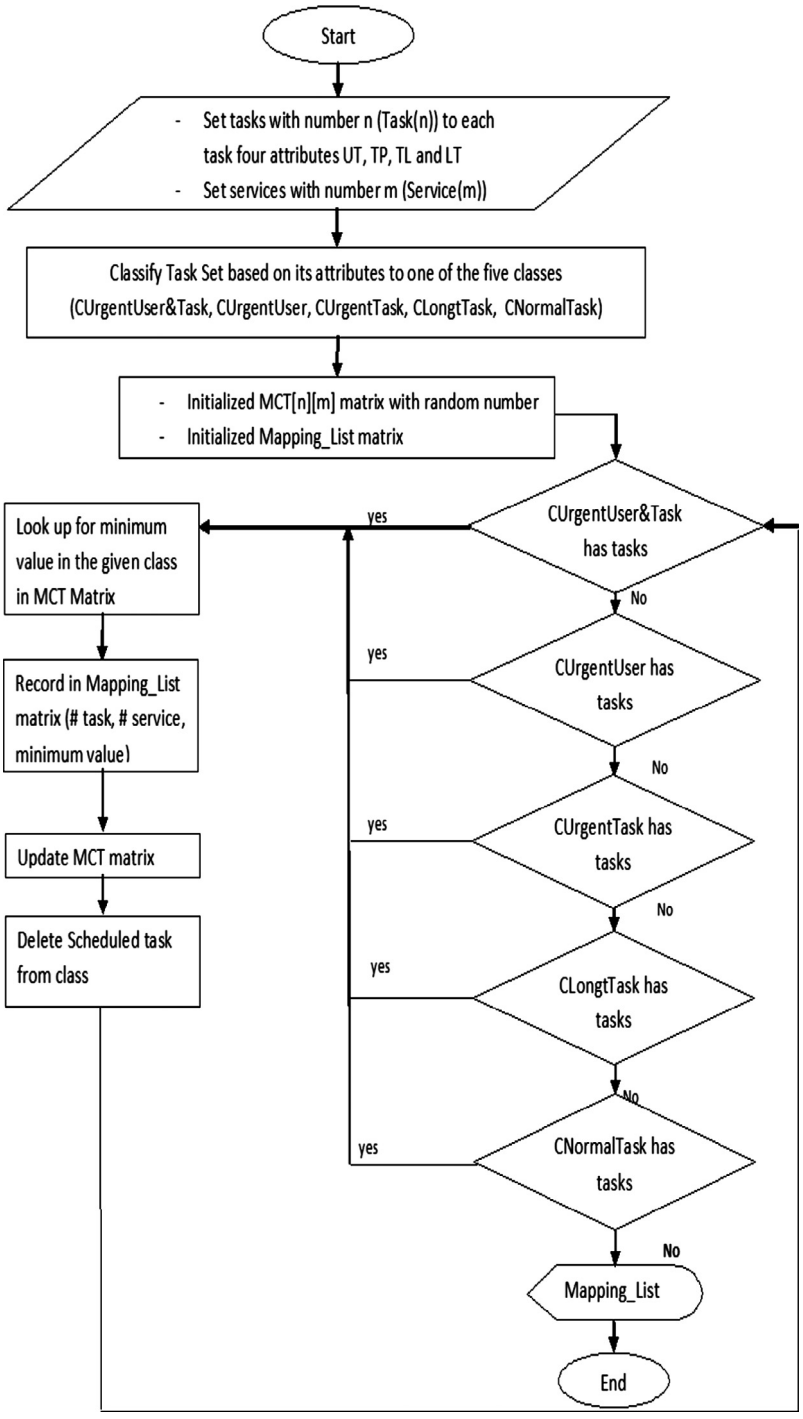

Figure 1 Flowchart of GTS algorithm.

its performance with that of Min-Min and TS algorithm, three simulation programs were developed to simulate the three algorithms (GTS, Min-Min, and TS Algorithm). Improved cost-based algorithm can't be compared with GTS algorithm because the way of calculating priority is different. In Improved cost-based algorithm the cost and profit attributes are used to calculate priority but in GTS algorithm the UT, PT, TL and LT attributes are used [4].

The simulation programs were developed using Java. To obtain accurate results the three simulation programs will use the same simulation parameter as shown in Table 1 [2].

Six performance metrics will be evaluated for the three algorithms:

1. Execution time span

Execution time span is the time duration taken from beginning of first task, start processing and end with last task finished the processing.

$$Execution\ Time = \mathrm{Te} - \mathrm{Ts} \tag{2}$$

where Te is time of ending last task, and Ts is time of start first task.

2. Average latency of tasks

Average latency is the ratio of total waiting time of tasks and number of these tasks. Note that there are four averages of latency used, each according to type of tasks.

$$Average\ Latency\ of\ Long\ Tasks = \frac{Total\ Waiting\ Time\ of\ Long\ Tasks}{Number\ of\ Long\ Tasks} \tag{3}$$

$$Average\ Latency\ of\ Tasks\ with\ Urgent\ Users = \frac{Total\ Waiting\ Time\ of\ Tasks\ with\ Urgent\ Users}{Number\ of\ Tasks\ with\ Urgent\ Users} \tag{4}$$

$$Average\ Latency\ of\ Tasks\ with\ Expected\ Urgent\ priority = \frac{Total\ Waiting\ Time\ of\ Tasks\ with\ Expected\ Urgent\ priority}{Number\ of\ Tasks\ with\ Expected\ Urgent\ priority} \tag{5}$$

Table 1 Simulation parameters.

Simulation parameters	Value
Number of tasks	200, 400, 800, 1200, 2400
Number of services	50,100
Percentage of tasks (Users Є Class A)	10%
Percentage of tasks (Users Є Class B)	20%
Percentage of tasks (Users Є Class C)	70%
Percentage of tasks (Size Є Long)	10%
Percentage of tasks (Size Є Normal)	90%
Percentage of tasks (Task Є Urgent)	25%
Percentage of tasks (Task Є High)	25%
Percentage of tasks (Task Є Medium)	25%
Percentage of tasks (Task Є Low)	25%
Percentage of tasks (User Є Class A, Task Є Urgent)	2.5%

$$Average\ Latency\ of\ Expected\ Urgent\ priority\ of\ Tasks\ with\ Urgent\ Users = \frac{Total\ Waiting\ Time\ of\ Expected\ Urgent\ priority\ Tasks\ with\ Users\ Urgent}{Number\ of\ Expected\ Urgent\ priority\ Tasks\ with\ Users\ Urgent} \tag{6}$$

3. Load balancing

Load balancing is the load in each service which was measured by using standard deviation.

$$Load\ Balancing = \sqrt{\frac{\sum_{i=1}^{n}(xi - x)^2}{n}} \tag{7}$$

where n is the number of services, xi is the number of tasks loading in service i and x is the average load of all services.

In order to achieve accurate results, at each group of tasks the average of five runs for the three simulation programs will be taken. At each run of five times different MCT matrix will be used.

5. Result analysis

In this section the six performance metrics will be studied.

5.1. Execution time span

As shown in Fig. 2 the performance of Min-Min algorithm is slightly lower than that of GTS algorithm; however, TS algorithm achieves the highest performance. This is because Min-Min algorithm searches in the whole MCT matrix for the services executing tasks faster. So the search is done on level of the whole matrix. But GTS algorithm searches in MCT matrix for service executing task with minimum execution time in the

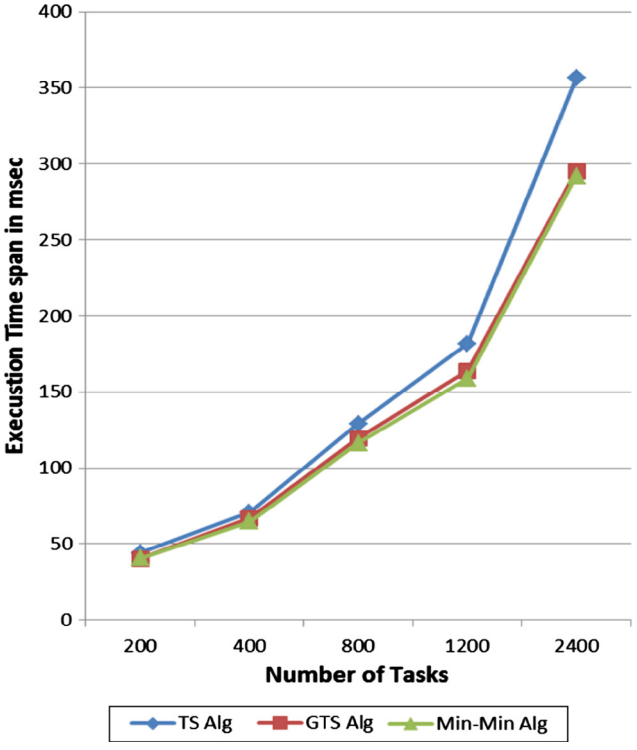

Figure 2 Execution time span with number of services 50.

chosen class faster. So the search is done on the level of group of tasks. Also TS algorithm searches in MCT matrix for service executing task with high priority faster. So the search is done on level of one task. Note that increasing the range of search, leads to fast execution. It's clear from Fig. 3 that increase in number of services from 50 to 100 leads to decrease in execution time for three algorithms.

5.2. Average latency of long tasks

Fig. 4 shows that latency of TS algorithm is the lowest. Because priority of tasks increases with 20% when it is long, also percentage of long tasks that have urgent users and its expected urgent priority are high. The performance of GTS algorithm is moderated between the two algorithms but it is much closer to TS algorithm than to Min-Min algorithm. Because the order of the category that includes long tasks is fourth but also first, second and third categories include long tasks so latency of GTS is much closer to latency of TS algorithm. Latency of Min-Min algorithm is the highest because Min-Min algorithm searches for minimum execution time so long tasks will be executed finally and its latency will be the highest. As shown in Fig. 5 increase in number of services leads to decrease in latency.

5.3. Average latency of tasks with urgent users

As shown in Fig. 6 latency in GTS algorithm is the lowest because tasks with urgent users will be distributed in first and second scheduled categories, which lead to low latency. Latency of Min-Min algorithm is the highest because Min-Min algorithm selects tasks with minimum execution time and this kind of tasks has 33% of long tasks. So latency will be high. Performance of TS algorithm is moderate between the other two algorithms, because priority of tasks increases with only 40% when its user is in urgent. Also it's clear that increase in number of services leads to decrease in latency as shown in Fig. 7.

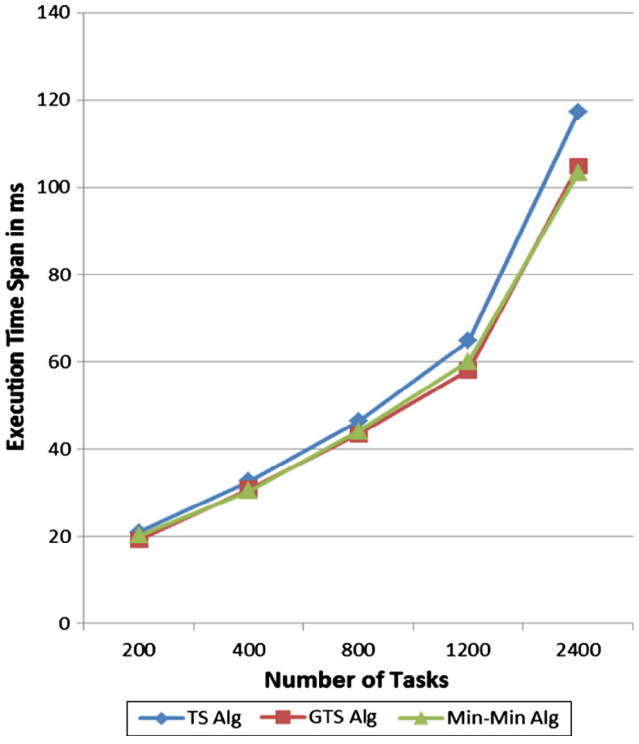

Figure 3 Execution time span with number of services 100.

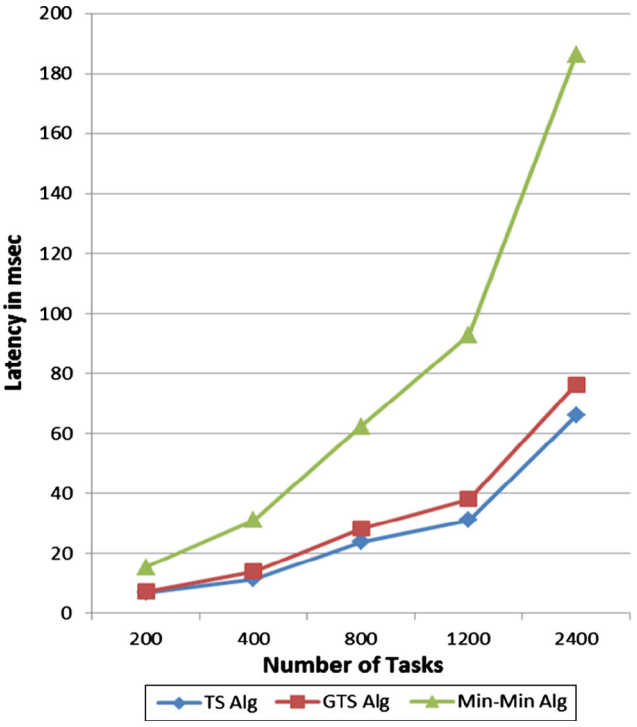

Figure 4 Latency of long tasks with number of services 50.

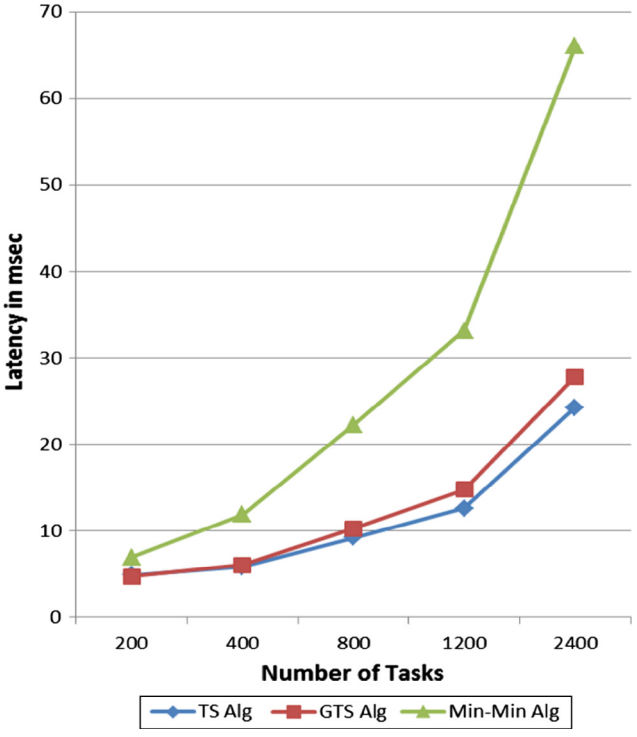

Figure 5 Latency of long tasks with number of services 100.

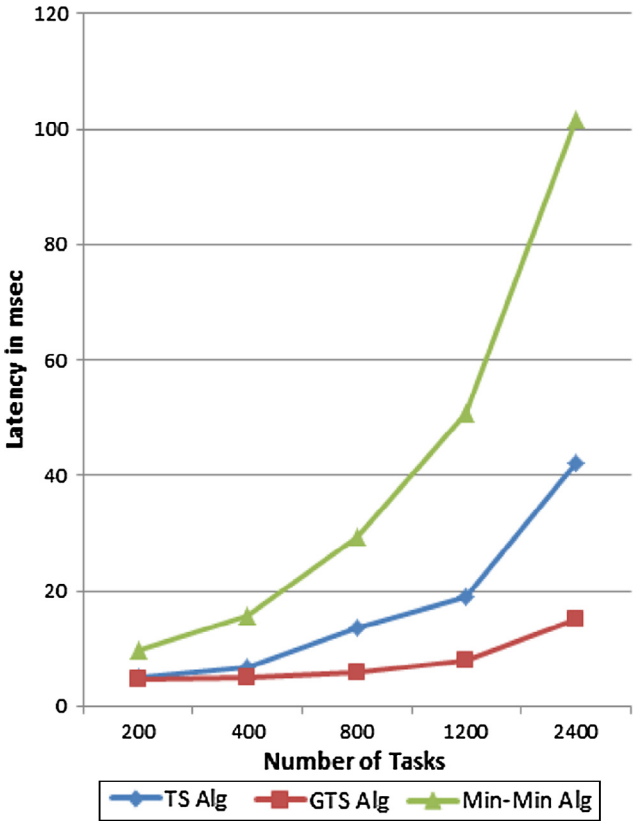

Figure 6 Latency of tasks with urgent users with number of services 50.

5.4. Average latency of tasks with expected urgent priority

From Fig. 8, it is clear that GTS algorithm achieves minimum latency compared to the other two algorithms. Because tasks with expected urgent priority will be distributed in first and third scheduled categories which also mean that these tasks will be scheduled early. Latency of TS algorithm in this case will be the highest. This is because tasks with high priority executed early, have high percentage of long tasks that lead the current tasks to wait. The performance of Min-Min algorithm is moderate between two algorithms because the percentage of long tasks in these tasks is down from 33% in the previous two types of tasks to 10%. So most of these tasks are executed early leading to lower latency. As shown in Fig. 9 increase in number of services leads to decrease in latency.

5.5. Average latency of expected urgent priority of tasks with urgent users

Fig. 10 shows that GTS algorithm achieves the lowest latency to these kinds of tasks than TS and Min-Min algorithm. This is because the category that has these tasks is scheduled first so the latency will be the lowest. Latency of TS algorithm is closer to latency of GTS algorithm. This is because the priority of tasks increases with 70% when its expected priority is urgent and its user is in urgent. So these tasks should have high priority which leads to be scheduled first. Latency in Min-Min algorithm is the highest because Min-Min algorithm executes first tasks having minimum execution time without consideration to any attributes of tasks. Percentage of long tasks in these

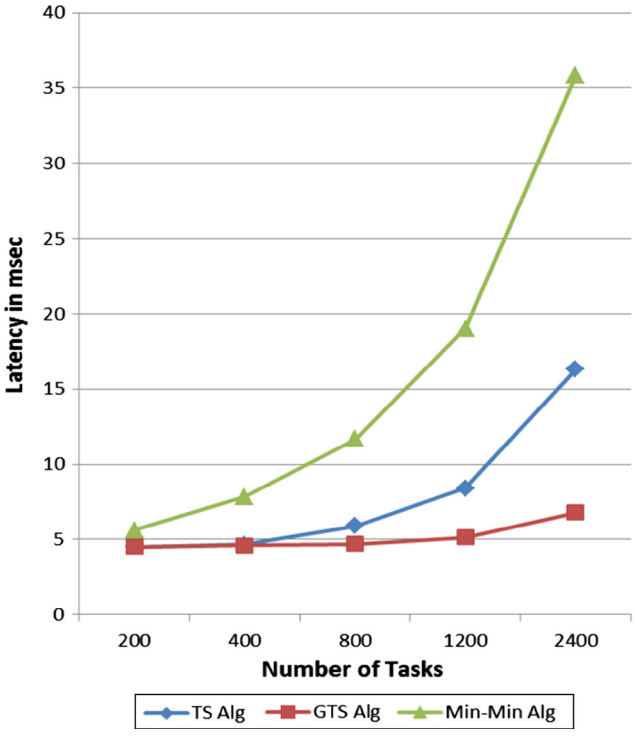

Figure 7 Latency of tasks with urgent users with number of services 100.

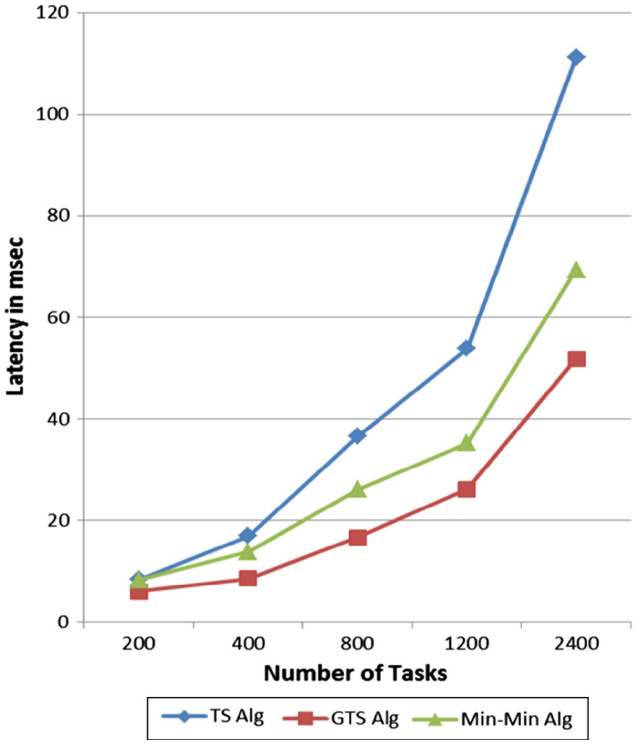

Figure 8 Latency of tasks with expected urgent priority with number of services 50.

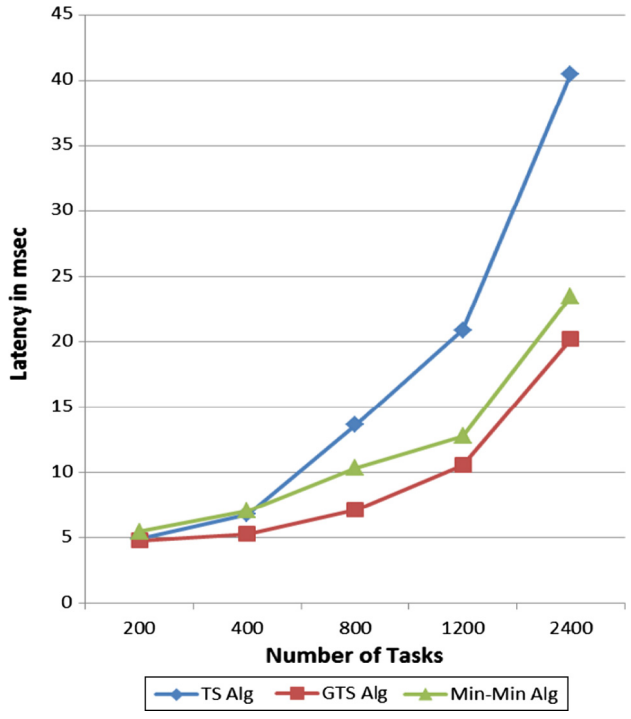

Figure 9 Latency of tasks with expected urgent priority with number of services 100.

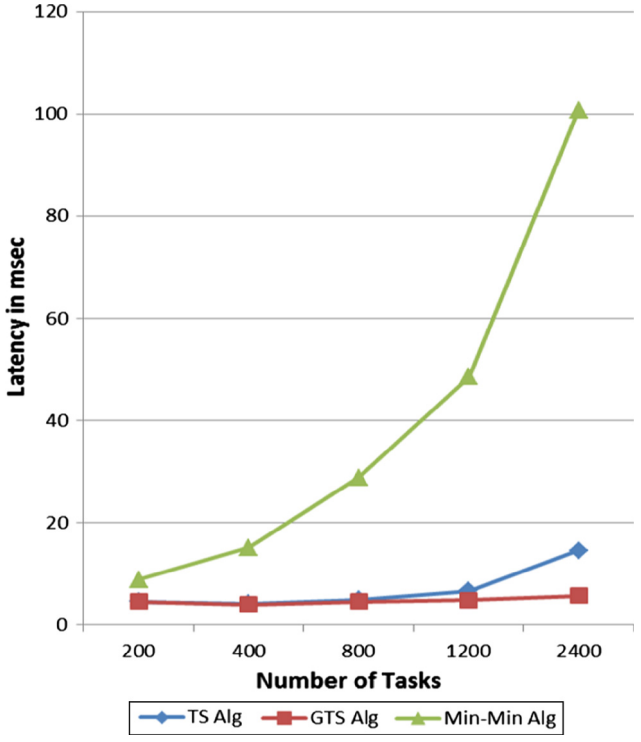

Figure 10 Latency of tasks with expected urgent priority and urgent users with number of services 50.

tasks is 33 so long tasks will be executed at the end which leads to increase in latency. As shown in Fig. 11 increase in number of services leads to decrease in latency.

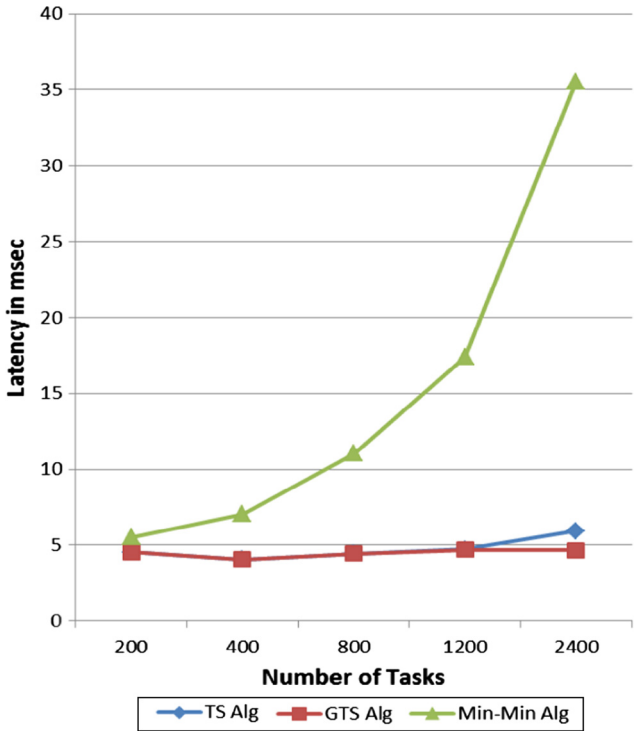

Figure 11 Latency of tasks with expected urgent priority and urgent users with number of services 100.

5.6. Load balancing

Load on services depends on timing of executing long tasks. So if long tasks are executed early, this will lead to two types of services: services executing one task but in a long time and other services executing more than one task in the same time. This leads to unbalancing. GTS and TS algorithms execute long tasks early so load balancing is high. However in Min-Min algorithm long tasks are executed at the end so load balancing is low as shown in Fig. 12. Also as shown in Fig. 13 increase in number of services leads to decrease in congestion on services.

6. Conclusion and future work

The objective of this paper was to define an algorithm used in cloud computing which gets minimum execution time to all tasks with low latency to tasks with high priority. GTS algorithm was proposed to schedule tasks into services with driven QoS. GTS algorithm combines many types of task attributes (user type, task priority, task size, latency of task) which are used to measure priority of tasks.

The experimental results show that GTS algorithm achieves minimum execution time to all tasks similar to Min-Min algorithm. Also minimum latency is obtained to different types of tasks compared with both Min-Min and TS algorithms.

Table 2 summarizes the previous comparison between three algorithms (GTS, TS, and Min-Min) using six performance metrics: execution time, load balancing, average latency to expected urgent priority of tasks with urgent users, average latency to tasks with urgent users, average latency to tasks with

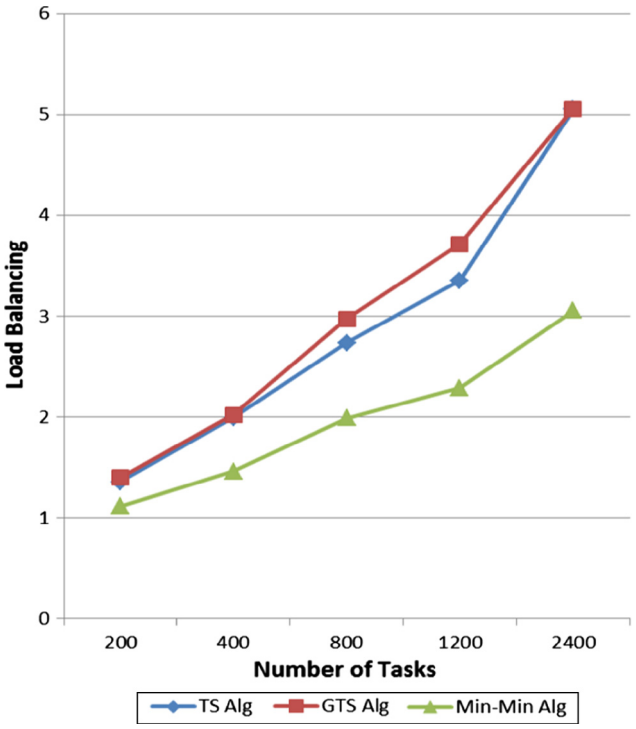

Figure 12 Load balancing with number of services 50.

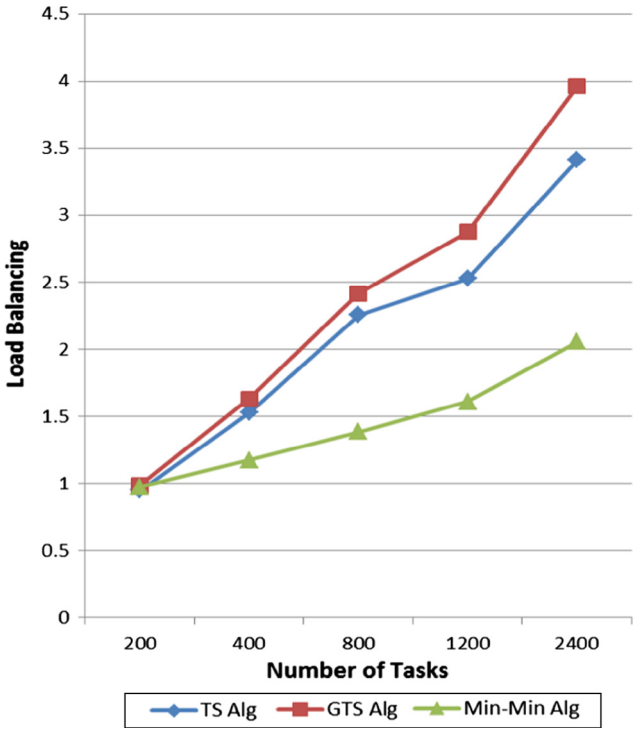

Figure 13 Load balancing with number of services 100.

expected urgent priority and average latency to long tasks. This is performed for number of services equal to 50 and 100. Also percentage of difference between GTS algorithm with Min-Min and TS algorithms is added.

It's clear from the table that increase in number of services from 50 to 100, leads to decrease in execution time, latency and pressure on congestion services. Also it appears that GTS algorithm achieves minimum execution time than TS algorithm

Table 2 Comparison between three algorithms.

Number of services	Performance metrics	Average value obtained using			Percentage of difference proposed GTS with	
		Min-Min	TS	GTS	Min-Min	TS
50 Services	Execution Time	134.52	156.32	137.20	More by 1.99%	Less by 12.23%
	Average Latency of Long Tasks	77.54	27.84	32.72	Less by 57.80%	More by 17.52%
	Average Latency of Tasks with Urgent Users	41.40	17.38	7.77	Less by 81.23%	Less by 55.29%
	Average Latency of Tasks with Expected Urgent Priority	30.51	45.38	21.84	Less by 28.41%	Less by 51.87%
	Average Latency of Expected Urgent Priority of Tasks with Urgent Users	40.39	6.94	4.72	Less by 88.31%	Less by 31.98%
	Load Balancing	1.98	2.89	3.03	More by 53.03%	More by 4.84%
100 Services	Execution Time	51.56	56.40	51.24	Less by 0.62%	Less by 9.14%
	Average Latency of Long Tasks	28.03	11.33	12.70	Less by 54.69%	More by 12.09%
	Average Latency of Tasks with Urgent Users	15.98	7.94	5.14	Less by 67.83%	Less by 35.26%
	Average Latency of Tasks with Expected Urgent Priority	11.82	17.33	9.58	Less by 18.95%	Less by 44.72%
	Average Latency of Expected Urgent Priority of Tasks with Urgent Users	15.30	4.72	4.46	Less by 70.84%	Less by 5.50%
	Load Balancing	1.43	2.13	2.37	More by 65.73%	More by 11.26%

with 12% and 9% with 50 and 100 services respectively. GTS algorithm gets low latency to tasks with expected urgent priority and its users are in urgent than both Min-Min and TS algorithms. GTS algorithm gets low latency to long tasks than Min-Min algorithm with 57% and 54% with 50 and 100 services respectively. GTS achieves load balancing almost the same as TS algorithm.

The GTS algorithm uses only four attributes stated before to apply QoS, and also the GTS algorithm only works for independent tasks and needs to queue all tasks first to make classification. So the future work is as follows:

1. Increment the number of attributes that apply QoS in the algorithm.
2. The algorithm can work with dependent tasks.
3. The algorithm can work in real time.

References

[1] Tilak Sujit, Patil Dipti. A survey of various scheduling algorithms in cloud environment. Int J Eng Invent 2012;1(September):36–9.

[2] Wu Xiaonian, Deng Mengqing, Zhang Runlian, Zeng Bing, Zhou Shengyuan. A task scheduling algorithm based on QOS-driven in cloud computing. In: International conference on information technology and quantitative management, China.

[3] Liu Gang, Li Jing, Xu Jianchao. In: Proceedings of the 2012 international conference of modern computer science and applications, Zhenyu Du; 2013. p. 47–52.

[4] Selvarani S, Sudha Sadhasivam G. Improved cost-based algorithm for task scheduling in cloud computing. In: International conference. IEEE; 2010.

[5] Abdullah Monir, Othman Mohamed. Cost-based multi-QOS job scheduling using divisible load theory in cloud computing. In: International conference on computational science. ICCS; 2013.

[6] Quarati Alfonso, Clematis Andrea, Galizia Antonella, D'Agostino Daniele. Hybrid clouds brokering: business opportunities, QoS and energy-saving issues. J Simul Model Pract Theory 2013;39 ():121–34.

[7] Chen Tao, Bahsoon Rami, Theodoropoulos Georgios. Dynamic QOS optimization architecture for cloud-based DDDAS. Int J Comput Algorithm 2013;02(June).

[8] Bittencourt Luiz Fernando, Madeira Edmundo Roberto Mauro. HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. J Internet Serv Appl 2011.

[9] Ravichandran S, Naganathan ER. Dynamic scheduling of data using genetic algorithm in cloud computing. Int J Adv Engg & Tech 2013;5(2):327–34.