International Journal of Engineering & Technology, 7 (4.6) (2018) 53-59

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Group Tasks Scheduling Algorithm for Cloud Computing Networks based on QoS

Dr. K. Jairam Naik, B.Veda Vidhya

Associate Professor, Department of Computer Science & Engineering
Vardhaman College of Engineering, Hyderabad
Department of Computer Science and Engineering MLR Institute of Technology, Hyderabad
**Corresponding author E-mail: Jairam.524@gmail.com*

Abstract

This article introduces a Novel Group-Tasks Scheduling Algorithm (NGTSA) which is used for allocating the tasks in the network of cloud computing by means of pertaining quality of services to gratify user's desires. The tasks are categorized into five classes by the anticipated algorithm. Every one group contains the tasks with akin attributes (like, types of the users and tasks, size and latency of the task). Once the tasks are allocated to a precise group, scheduler starts assigning these tasks to accessible services. This assignment of tasks was performed in two steps: In Step-I is to decide which group tasks is to be scheduled foremost. Such decision will be based on the attributes of the tasks of each group. Hence, the groups which have higher task's attribute values are scheduled foremost. Step-II is for taking internal decision that is which task from the selected group is scheduled foremost. This decision will be based on time needed for task's execution. Therefore, the task which has the lowest time for execution will schedule foremost.

Keywords: *Cloud Computing, Task Scheduling, Latency, Load Balancing, Services, Execution Time.*

1. Introduction

Cloud Computing offers storage and computational resources for sharing, and also presents information and services using Internet to its users as per their requirements by means of assortment of applications [1]. Scheduling the existing tasks with the specified constraints for execution is the main objective in the cloud computing situations. Applying the QoS as per users necessities, harmonizing among the QoS and equality amongst the tasks is the primary requirement. For satisfying the obligations of cloud computing, many algorithms were 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 [2] is the one of these algorithms. This article launch's a novel algorithm called Novel Group-Tasks Scheduling Algorithm (NGTSA) that is pedestal on Cost-based algorithm's process, TSA and Min-Min algorithms. NGTSA employs the same approach of evaluation of priority to the tasks in TSA for both groups. However NGTSA employs Min-Min algorithm for scheduling tasks contained in each class [3]. The remained parts of this article are structured as: Section 2 presents the existing efforts, Sec.3 states the anticipated NGTSA, Sec.4 gives you the examination of simulation, Performance of anticipated NGTSA are illustrated in Sec.5 the, and the conclusion and future scope are presented in Sec.6.

2. Related Works

To initiate QoS in cloud computing environments [2] Task scheduling algorithms are primarily used. For computing the priority of tasks, these scheduling algorithms are used. They have to consider the following four Standardized/Normalized attributes of tasks:

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:

$$\mathcal{P}(i) = \alpha * NUT + \beta * NPT + \gamma * NTL + w * NLT \qquad (1)$$

Where,
Standardized values for the attribUtesUT,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 [2].
The tasks are arranged in the order of priority. Every task from the sorted task queue was scheduled against the required services which can com***P***lete the task at the earliest.
Min-Min algorithm [3], which relays on task's execution time to schedule task is one among the scheduling algorithms used in cloud computing. 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.

To 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 [4] were used. Task's Priorities basing with resources profit and cost is computed with this algorithm. 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) [5] 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 [6] were used. This was mainly used for the public or private resources, with a key intension of maximizing the User fulfillment and broker's profits.
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 [7], 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 [8] is used.
To allocate the resoUrces for the tasks arrived at uncertain run time interval, dynamic scheduling algorithms are used. Meaning that, it is as tough as numerous tasks at the same time coming. Genetic Algorithms [9] are used to avoid such scheduling difficulties.
To relate the QoS needs in Task Scheduling, the algorithm GTS [10] employs the thought of grouped tasks there in enhanced cost_based algorithm̧. Thereafter, uses the algorithm Ṁin_Min for scheduling tasks within each group.

3. The Proposed NGTSA Algorithm

Compared with TSA, the Ṁin_Ṁin algorithm attains lesser execution time. And the TSA attains low latency when contrasted with Min-Min. So the anticipated NGTSA intends to get little latency and with smallest execution time.
The NGTSA employs the thought of tasks that are grouped there in enhanced cost_based Algo, to pertain QoS here in TSA and afterward employs Ṁin_Ṁin algorithm for scheduling tasks within every group.
The key thought of NGTSA is to separate all tasks into groups according to their attributes. As enlightened in TSA, the attributes of tasks are used. Tasks amid analogous attributes are exists in each groups. Based on the weights that specified to task's attributes in TSA, these groups will be prearranged for scheduling. Herein, the groups are subjects to the scheduling whereas tasks are not. Compared to other groups, the foremost scheduled group will contain tasks amid higher values of attribute or with priority wise high. Subsequently, the task with least time for execution would be scheduled foremost in the selected group.
Let, ɳ, ɱ be the number of independent tasks and services respectively. ɳ, ɱ are the input of NGTSA Algorithm. Every one task contains four attributes:
1) TaskUsersType (UT): give you an idea regarding the kind of users (user class - A, B & C).
2) TaskPriorExp (PT): give you an idea regarding the probable priority of tasks scheduled (low, medium, high, vital priority).
3) TaskLength (TL): characterizes the load or length of tasks (Normally loaded, lengthy).
4) LatencyTask (LT): give you an idea regarding the task's latency.

The Ą̧lgorithm NGTSA 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.

ṀĊŢ matrix is a $n \times m$ matrix which provisions the assessment of anticipated time for completion of all tasks on every one of services (Į̧nitialized Ṁinimum Ċompetition Ţime). The row of ṀĊŢ 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, ṀĊŢ *(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 ṀĊŢ 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 ṀĊŢ matrix ṀĊŢ *(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.
Algorithm 1 demonstrates the steps of the NGTSA algorithm. Scheduling will starts with defining the inputs for the NGTSA algorithm. Thereafter, initialize Ṁapping_list matrix and ṀĊŢ 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 vales. After that, this task is placed in to the determined class.

Algorithm 1: The NGTSA Algorithm

```
1   Start
2    - Task ← n && Task(n) to each task's four attributes UT,
    PT, LT and TL
     - Service ← m; Service(m)
3   Separate the tasks set in to one of five cstegories based on
    its attribute value
    C_UrgentUser&Task, C_UrgentUser, C_UrgentTask,
    C_LongTask and C_NormalTask
4    - Initialize MCT[n][m] Matrix with random number
     - Initialize Random List Matrix considering SR
5   FOR (TaskClass=1; TaskClass ≤ 5; TaskClass ++)
6   IF ((UrgentUser && Task has Tasks) && ( C_UrgentUser)
    && (C_UrgentTask) &&
    (C_LongTask) && (C_NormalTask))
    Then
7   Lookup for Minimum value in the given class in MCT Matrix
8   Record in Mapping-list matrix (#task, #Service, Minimum
    Value)
9   Update MCT Matrix
10  Delete Scheduled Task from the Class
    Else
11          Mapping_list
12  END-FOR
```

4. The Study of Simulation Results

Like described above in the proposed NGTSA algorithm, the Min-Min algorithm is used for scheduling the tasks inside the chosen class. Whereas, the TSA algorithm is used to make a decision which class in NGTSA will be scheduled first according to the assigned calculated priorities to tasks. There are three simulation programs were developed to assess NGTSA algorithm's behavior and to contrast its performance among existing algorithms like Ṁin-Ṁin, TSA and GST. Since the process of computing the priority of tasks is different in an Ḭmproved cost-based algo, it can't be contrasted with NGTSA algorithm. In NGTSA algorithm the attributes UT, PT, TL and LT are used [4] to compute the priority of tasks. But, in Ḭmproved Ċost-based algorithm the profit and cost ɑttributes are used.

Simulation programs were executed in CloudSim and developed using Java programming language. For obtaining accurate results, there have been used the same simulation parameter by all the three simulation programs as shown in Table 1. For all the three algorithms, six performance metrics have been evaluated. They are as explained below:

A. Execution Time Span

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

Duration of Time spent from the first task's processing begins to the end of processing by the last task is called Execution time span and is stated using equation (2). Where, Ts and Te represents the times of the first task starting last task ending respectively.

B. Ţask's 𝓐verag Ĺatency

The ɤatio of sum of the waiting time of ţasks and the ŋumber of ţasks is called Average Latency. In this, average of the four latencies is used and each one is as per the kind of tasks.

$$\mathcal{A}verage\ Ĺatenco\ of\ the\ Ĺorger\ Tasks = \frac{Total\ Ẉaiting\ Time\ of\ the\ Lorger\ Tasks}{Ɗumber\ of\ Lorger\ Ţasks} \tag{3}$$

$$\mathcal{A}verage\ Ĺatency\ of\ thr\ Ţasks\ with\ Urgent\ Users = \frac{Ţotal\ Waiting\ Ţime\ of\ the\ Ţasks\ with\ Urgent\ Users}{Ŋumber\ of\ Ţasks\ with\ Urgent\ Users} \tag{4}$$

$$\mathcal{A}verage\ latency\ of\ the\ Ţasks\ with\ expected\ Urgent\ Priority = \frac{Ţotal\ Waiting\ Ţime\ of\ the\ Ţasks\ with\ Expected\ Urgent\ Users}{Ŋumber\ of\ Ţasks\ with\ Expected\ Urgent\ Priority} \tag{5}$$

$$\mathcal{A}verage\ Ĺatency\ of\ Expected\ Urgent\ priority\ of\ Ţasks\ with\ Urgent\ Users = \frac{Total\ Waiting\ Time\ of\ Ecpected\ Urgent\ Priority\ Tasks\ with\ Urgent\ Users}{Ɗumber\ of\ Expected\ Urgent\ Priority\ Ţasks\ with\ Urgent\ Users} \tag{6}$$

C. Load balancing

It indicates the load assigned to each service in the cloud should be at its normal level. Means, no service should be overloaded or underloaded with tasks. Deviation of load is measured by using the standard deviation as given below.

Table 1: Parameters used for Simulation

Parameters of experiment Simulation	Values
Ṅo.of of tasks used in simulation	Ḭ1-200/ Ḭ2-400/ Ḭ3-800/ Ḭ4- 1200/ Ḭ5-2400
Ń́umber of services used in simulation	50,100
% of ***T***asks (Users belongs to Ċlass Á)	10
% of ***T***asks (Users belongs to Ċlass Ḇ)	20
% of ***T***asks (Users belongs to Ċlass)	70
% of ***T***asks (Şize belongs to Ļonger)	10
% of ***T***asks (Şize belongs to Ńormal)	90
% of ***T***asks (Ţasks belongs to Urgent)	25
% of ***T***asks ((Ţasks belongs to High)	25
% of ***T***asks ((Ţasks belongs to Ṁedium)	25
% of ***T***asks ((Ţasks belongs to Low)	25
% of ***T***asks (Users belongs to Class A && ***T***asks belongs to Urgent)	3

$$Ştandard\ Ļoad\ Ḋeviation = \sqrt{\frac{\sum_{i=1}^{n}(x_i - x)^2}{\eta}} \tag{7}$$

Here, η represents the entire amount of services; x is the entire service's average load, and x_i is the amount of loaded tasks to the service i. In turn to attain exact results at every class of tasks, for every three programs simulation an average of 10 consecutive running's was measured. Different MCT matrix was generated for each run (of 10 times). This Algorithm considers the Success Rate (SR) of the processors [11-15]. The SR value of a processor increases by 1 for every job successfully executed by it. The SR is formulated as $SR = \frac{NS}{NS+NF}$. Where, NS, NF represents the number of jobs successfully executed or failed by the processor respectively.

5. Analysing the Results

The study of various performance metrics are described in this section.

5.1. Execution time span

Though, TSA algorithm attains the highest performance, the performance of Min-Min algorithm is somewhat inferior to the GTS algorithm as depicted in Fig.1. Because the algorithm Ṁin-Ṁin takes very less time for searching the services needed for tasks execution in the entire MCT matrix. It means the searching is performed on the whole MCT matrix. But, the proposed NGTSA searches the service needed for task executing in MCT matrix with a minimum time execution in a specifically selected class quicker. It means the searching is performed on a specific group of tasks.
As well, the TSA algorithm searches quickly for the service in MCT matrix with top priority. It means the searching is performed on the level of one task. Fast execution can be initiated with the increase in the range of search. As stated in Fig. 2, it is noted that reduction in execution time for 3 algorithms instigated by the increase in ŋumber of services to 100 from 50.

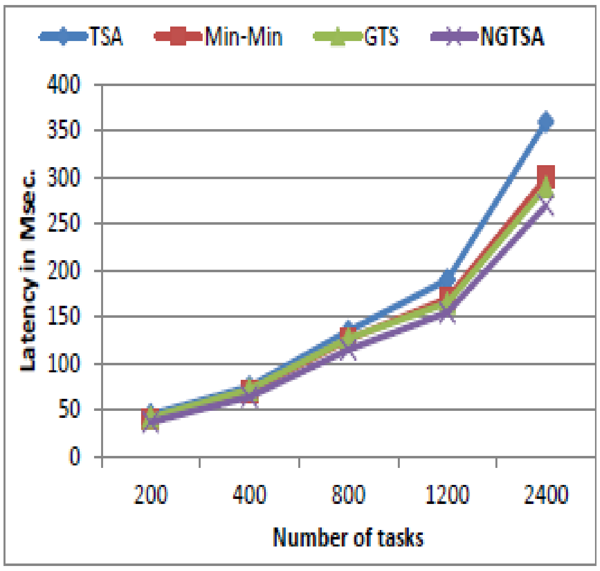

Figure 1. Time span of execution when the ɳᴜmber of 𝑺ervices = 50

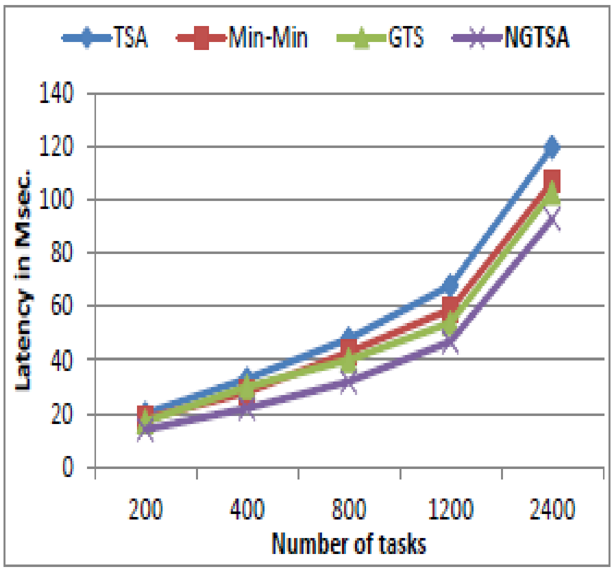

Figure 2. Time span of execution when the ɳᴜmber of 𝑺ervices = 100

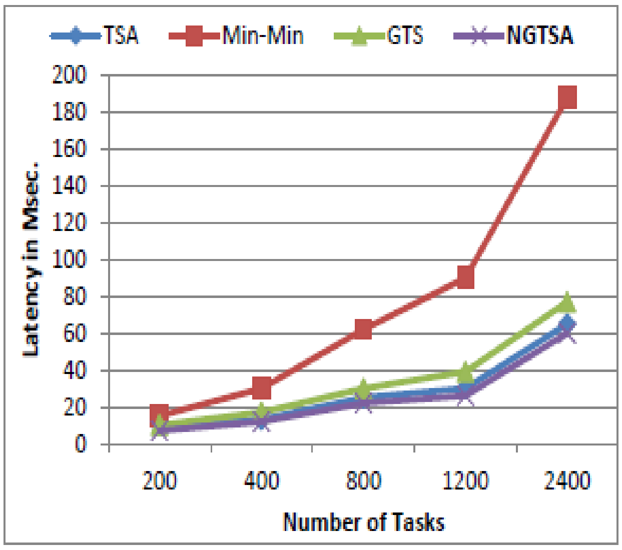

Figure 3. Latency for long tasks when the ɳᴜmber of 𝑺ervices = 50.

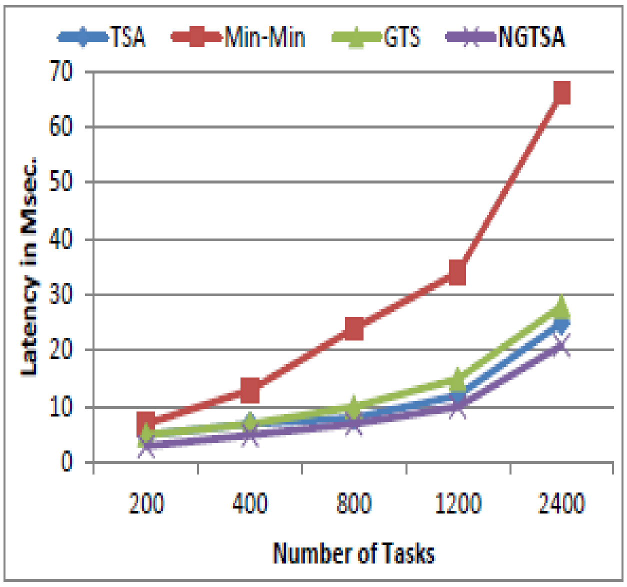

Figure 4. Latency for long tasks when ɳᴜmber of 𝑺ervices = 100.

5.2. Avrage Latency for long Ţasks

The latency of TSA algorithm is smallest as shown in Fig. 3. Because when the tasks are long, their priority is increases by 20%. So that, the long tasks that have expected urgent priority percentage and urgent users also increases. The NGTSA is having reasonable Performance lies between the two algo's, however when compared with Min-Min algorithm, it is much nearer to TSA algorithm. Latency of NGTSA is greatly nearer to the latency of TSA because in addition to the fourth (includes long tasks) category, the long tasks are also founded in the first, second and third categories. Because the Min-Min algorithm looks for smallest execution time, its latency is the highest. So when the longer task executes at the end their latency will be the larger. As illustrated in Fig. 4 decrease in latency leaded by the increase in number of services.

5.3. Áverage Ĺatency of Ţasks with Ư rgent Users

Latency in NGTSA is the smallest as shown in Fig. 5. This is because the tasks with urgent users are dispersed to the classes which can lead to low latency i.e. scheduled first and second.

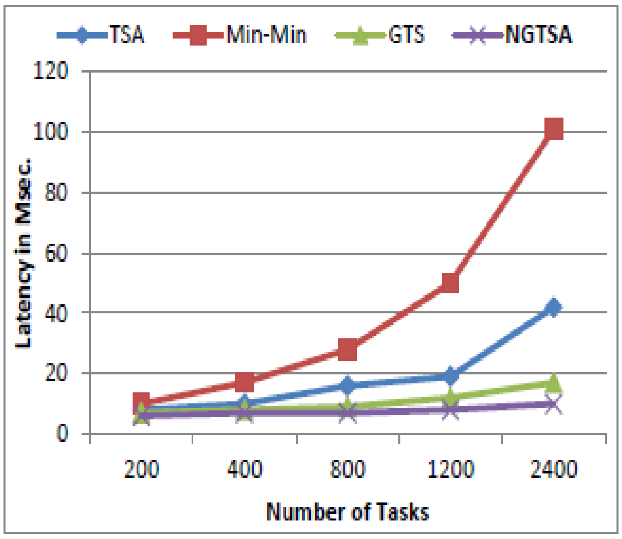

Figure 5. Latency of tasks when urgent users with ɳumber of *S*ervices = 50.

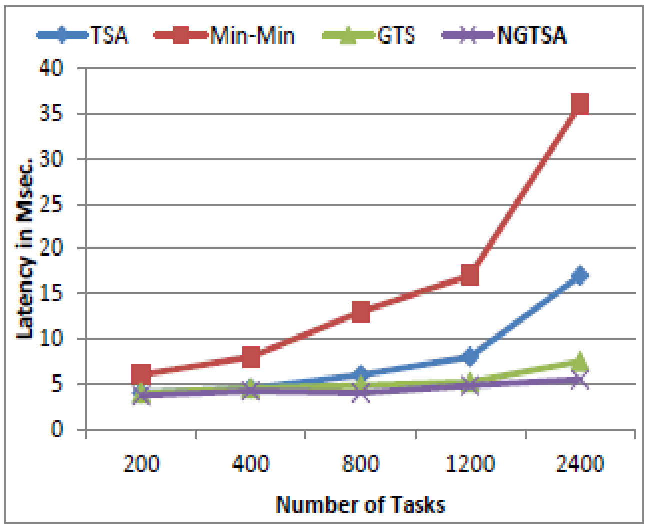

Figure 6. Latency of tasks when urgent users with ɳumber of *S*ervices = 100.

Because the Ṁin-Ṁin algorithm chooses tasks with least execution time the latency of Min-Min algorithm is the uppermost. And this type of tasks has 33% of lengthy tasks. So there is a higher latency. In case if it urgent user, the priority of tasks enhances only with 40%. Because of this, the performance of TSA algorithm is reasonable and lies between the performances of the other two algorithms. Decrease in latency lead by enhance in ɳumber of services as represented in Fig. 6.

5.4. Ǻverage Ĺatency of Ţasks with Ėxpected Urgent Priority

Because of the tasks with expected urgent priority are dispersed in to first and third scheduled classes, it is obvious that the NGTSA attains smallest latency contrasted to the other two algorithms as shown in Fig. 7. This also denotes that, early on these tasks will be scheduled. In this case the latency of TSA algorithm will be the highest. This is for the reason that tasks with higher priority will be executed earl. Current tasks have to wait because of high percentage of long tasks.

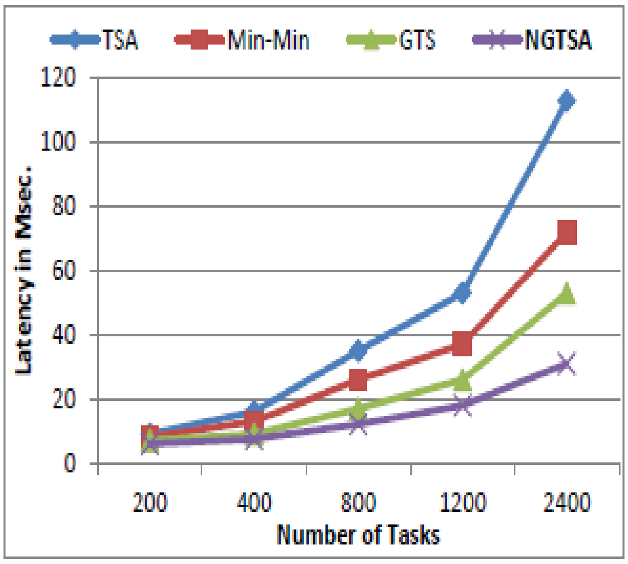

Figure 7. Latency of tasks when expected urgent priority with ɳumber of *S*ervices = 50.

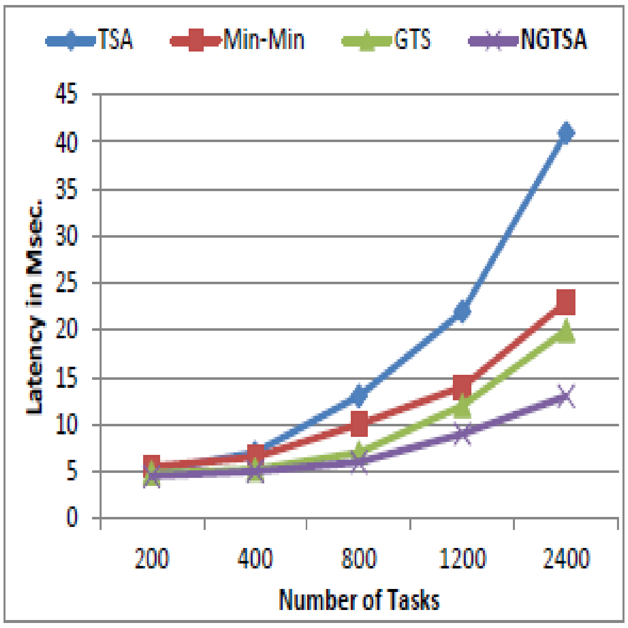

Figure 8. Latency of tasks when expected urgent priority with ɳumber of *S*ervices = 100.

Because the %ge of long tasks in these tasks is down, the performance of Min-Min algorithm is reasonable and lies between the two other algorithms. Hence, leading to lesser latency, many of these tasks are executed early on. Increase in the number of services direct to diminish in latency like illustrated in Fig. 8.

5.5. Ǻverage Ĺatency of Ėxpected Urgent priority of Ţasks with Urgent Users

The NGTSA algorithm attains smallest latency to these types of tasks compared to TSA, Ṁin-Ṁin algorithms as illustrated in Fig. 9 & 10.

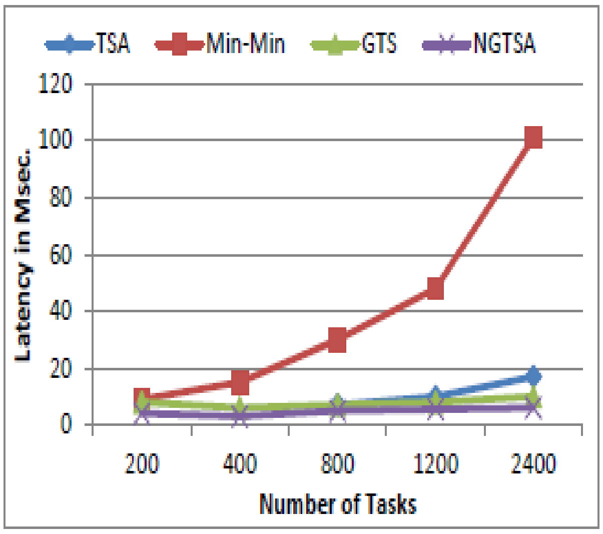

Figure 9. Latency of tasks when expected urgent priority and urgent users with ɳUmber of *S*ervices = 50.

Because these tasks are first scheduled, hence the latency will be small. But, the latency of TSA was nearer to latency of NGTSA, because the priority of tasks rises with seventy percentages when its user is in urgent and expected priority is urgent. Because the Ṁin-Ṁin algorithm executes tasks encompass minimum execution time first with no consideration of any other task's attributes. Hence, latency in Ṁin-Ṁin is the largest.

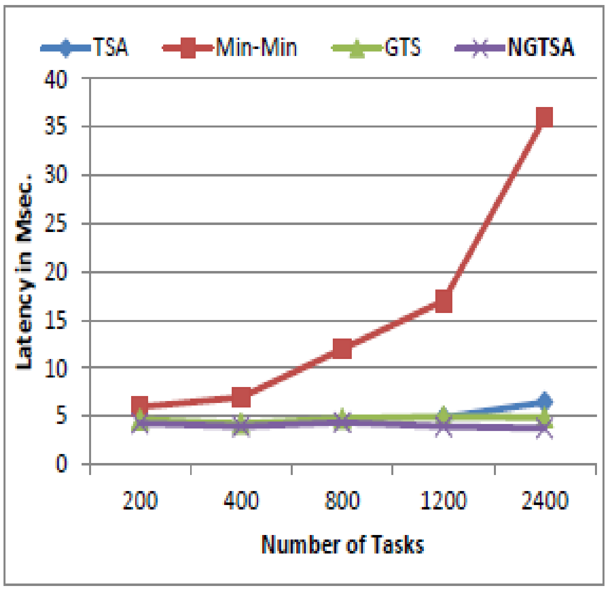

Figure 10. Latency of tasks when expected urgent priority and urgent ɳUmber with number of *S*ervices =100.

5.6. Balancing the Load

Timing of executing long tasks affects the loads on services. Long tasks are executed early by the NGTSA and TSA algorithms hence, load balancing is high. But, long tasks are handled at the ending in Min-Min algorithm, so balancing the load is stumpy as illustrated in Fig. 11. Moreover, the raise in number of services directs to reduce within congestion on services as illustrated in Fig. 12.

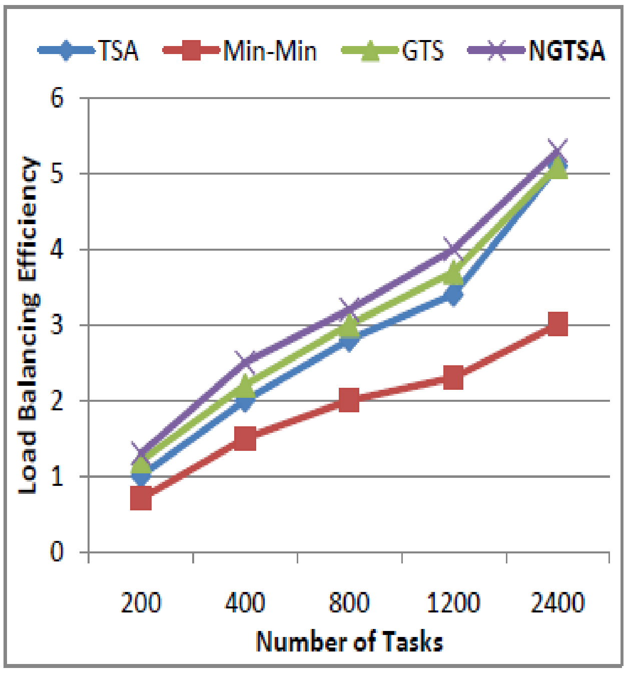

Figure 11. Load balancing when the ɳUmber of *S*ervices=50.

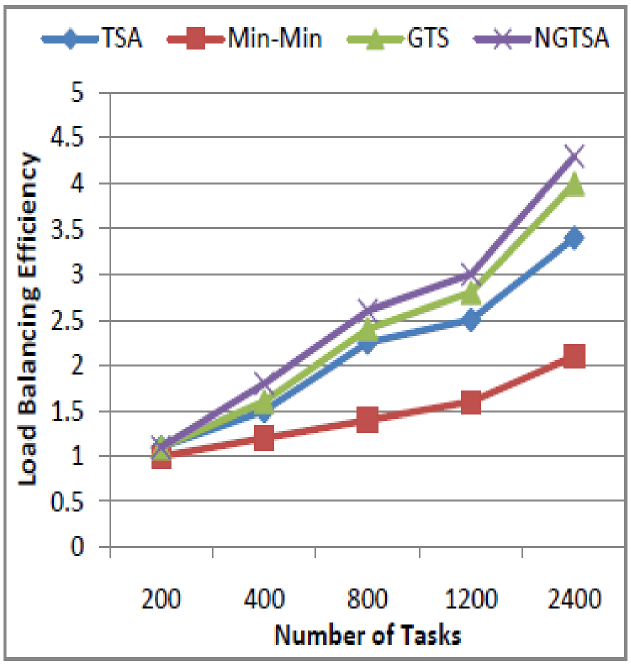

Figure 12. Load balancing when the ɳUmber of *S*ervices=100.

6. Conclusion and Future Scope

The purpose of this article was to characterize an algorithm to obtain lowest time for execution to all tasks through near to the ground latency to tasks in cloud computing with high priority. To schedule the tasks onto the services with ambitious QoS the NGTSA algorithm was anticipated. The proposed NGTSA algorithm unites many kinds of task attributes which are used to assess the task's priority like user type, task priority, size, latency. The simulation results states that NGTSA algorithm attains smallest execution time to all tasks like in Ṁin-Ṁin. Also smallest latency is achieved to diverse kinds of tasks contrasted with both TSA algorithms and Ṁin-Ṁin.

References

[1] Tilak Sujit et al. A survey of scheduling algo's in cloud env. Intl Jr Engg Inv, Sep-2012, PP 36–9.
[2] Wu Xiaonian et al, A task scheduling algo's based on QOS-driven in cloud comp., Intnl conf on info tech and quantitative management, in China.
[3] Liu Gang et al Proceedings of the 2012 intnl conf of modern comp science and app's, Zhenyu Du; 2013. pp. 47–52.
[4] Selvarani S et al Improved cost-based algo for task scheduling in cloud comp, Intnl conf. IEEE-2010.
[5] Abdullah Monir et al, Cost-based multi-QOS job scheduling using divisible load theory in cloud comp, Intnl conf on computational science. ICCS-2013.
[6] Quarati Alfonso et al, Hybrid clouds brokering: business opportunities, QoS and energy-saving issues. J Simul Model Pract Theory 2013, pp. 121–34.
[7] Chen Tao et al, Dynamic QOS optimization architecture for cloud-based DDDAS. Intnl Jr. Comp. Algorithm June-2013.
[8] Bittencourt et al , A cost optimization algorithm for workflow scheduling in hybrid clouds. Jr. Internet Serv Appl-2011.
[9] Ravichandran S et al, ER. Dynamic scheduling of data using genetic algorithm in cloud comp, Innl. Jr. Adv Engg &
[10] Tech 2013, pp. 327–34.
[11] Hend Gamal El Din Hassan Ali et al, Grouped tasks scheduling algorithm based on QoS in cloud computing network Cairo University Egyptian Informatics Journal (2017), pp.11–19
[12] K. Jairam Naik, Dr. A. Jagan, Dr. N. Satyanarayana, "An enhanced mechanism for balanced job scheduling based on deadline control in computational grid", 2nd International Conference on Emerging Trends in Electrical, Communication and Information Technologies 2015 (ICECIT 2015), SRIT, Anantapur,AP. 19-21Dec, 2015
[13] K. Jairam Naik, Dr. A. Jagan, Dr. N. Satyanarayana, "A novel algorithm for fault tolerant job Scheduling and load balancing in grid computing environment", International Conference on Green Computing and Internet of Things (ICGCIoT 2015), Galgotia University, Noida, 8-10 Oct, 2015(IEEE Explore)
[14] K. Jairam Naik, Dr. N. Satyanarayana, "A Novel Fault-tolerant Task Scheduling Algorithm for Computational Grids", 15th IEEE International Conference on Advanced Computing TechnologiesÂ (ICACT-2013), AITS, Rajampet, Andrapradesh, 10th – 11thÂ August 2013(IEEE Explore)
[15] K. Jairam Naik, Dr. A. Jagan, Dr. N. Satyanarayana, "A Cost Greedy Price Adjustment based Job Scheduling and Load Balancing in Grids", International Journal of Computing and ICT Research (IJCIR), Vol.10, Issue 1, June,2016
[16] K. Jairam Naik, Dr. A. Jagan, Dr. N. Satyanarayana, "A Novel Approach for Job Scheduling and Load Balancing in Grid Computing Environment", Annals. Computer Science Series Journal, Vol. XIII fasc.2, December, 2015