Energy-Efficient And Improved Qos -Driven Task Scheduling Algorithm In Mcc Environment Using Cloudsim Simulator

Dr.G.Anandharaj, K.Suganthi

Abstract: As an evolving and prospective computing paradigm, mobile cloud computing (MCC) can significantly improve computing capacity and save energy from intelligent mobile devices. Because of some intrinsic mobile device defects, such as restricted battery power, inadequate storage space, and mobile apps, many mobility management problems, quality of service (QoS), energy management, and safety problems are faced. A mobile device implementation is known as Task. The main focus of the task scheduling is to improve the effective use of resources and thus reduce the completion time of the task. This research explores relative analysis of energy-efficient and improved QoS-driven tasks Scheduling (E2IQDTS) algorithms for optimizing multi-objective problem in a Mobile Cloud Computing Environment scheduling parameters such as Makespam, Dynamic Offloading, Deadline-satisfied, Task Length, Priority, Delay-Sensitive, etc. Evolutionary algorithms such as Genetic Algorithm (GA), Genetic Programming (GP) and Differential Evolution (DF) are used.

Index Terms: Mobile Cloud Computing, QoS, Genetic Algorithm, Differential Evolution, Energy-efficient, Task Scheduling.

1. INTRODUCTION

Due to the fast development of mobile apps and the development of cloud computing, mobile cloud computing (MCC) has become an vital research sector. MCC relates to cloud computing being integrated into a mobile setting. MCC at its easiest refers to an environment where data storage and information handling takes place outside the mobile device. Mobile cloud applications shift computational energy and data storage away from mobile devices to strong, centralized cloud-based computation platforms that are then accessed via a thin-native client-based wireless link. Mobile cloud computing offers cloud information storage and handling facilities for portable consumers, avoiding the need for strong computer setup (e.g. CPU speed, memory capability, network and bandwidth, etc.) as all resource-intensive computing can be carried out in the cloud.

1.1 Objective

Cloud consumers and suppliers must have a powerful service-level contract in real-time cloud apps to guarantee timing of applications and application deadlines. A scheduler in real time must ensure that processes meet deadlines irrespective of system load or make span. Priority is given to the planning with deadlines of these periodic tasks. Through some strategy, each task in priority scheduling is provided precedence, so scheduler assigns tasks according to preferences to resources. Solving Mobile Cloud Computing (MCC) job planning is regarded as an NP-hard issue. The new algorithm based on the technique of task scheduling, which will effectively distribute the load between the virtual machine in order to minimize the overall response time (QoS). A comparison of this suggested algorithm of task scheduling method is conducted on CloudSim simulator that demonstrates that, with the assistance of Evolutionary algorithms, this will outperform the ideal energy-efficient Improved QoS-driven Tasks Scheduling by E2IQDTS Algorithm.

- *Dr.G.Anandharaj is finished Doctorate in Computer Applications in Anna University, India, PH-01123456789. E-mail: younganand @gmail.com*
- *K.Suganthi is currently pursuing M.Phil (Computer Science) Research Scholar in Adhiparasakthi College of Arts and Science, Kalavai, Tamil Nadu - 632506 PH-9841508044. E-mail: Sugan_wcc@yahoo.com*

1.2 Problem Definition

Task-offloading is not always effective, as it relies on several variables, such as wireless channel transmission bandwidth, task-offloading energy consumption on mobile devices, cloud assignment execution energy consumption, and so on. For instance, mobility as the mobile device's intrinsic characteristic can force mobile consumers to frequently modify the access point (AP) when users switch from one location to another. Sometimes this type of dynamics leaves the wireless link inaccessible, making the waiting period longer than anticipated, which can degrade customers' Quality of Experience (QoE), Even leading to the refusal of customers to acknowledge response time for important tasks in particular. In addition, energy consumption is another significant factor that greatly influences the choice to offload. For example, if the energy consumption caused by task offloading on mobile devices and data transmission via wireless channel were larger than task execution locally without offloading, it would make no sense to execute tasks remotely on the cloud side from the point of view of saving mobile device power consumption. The resource scheduling difficulties include dispersion, uncertainty, and resource heterogeneity that are not solved with traditional cloud environment algorithms. Therefore, by taking care of these characteristics of the Mobile cloud setting, there is a need to perform cloud functions efficiently.

1.3 Contribution

Under MCC settings, task scheduling is performed as a multi-objective optimization problem, taking into account both unconstrained and time-limited instances.

1) The suggested task scheduling algorithm to minimize the complete cost as well as the mean load in the MCC setting.

2) We research the compression of information that helps to save energy between the mobile device and the cloud.

3) The planning of tasks in MCC settings is performed as a multi-objective issue of optimization, taking into account both unconstrained and time-limited instances.

4) Simulations demonstrate that our suggested algorithm has improved overall cost efficiency, Mean load and timeline relative to other current algorithms for planning tasks.

2 SYSTEM DESIGN

2.1 Existing System

Task uploading and scheduling with the interactions of dependence between tasks as a multi-objective optimization problem. We also bring into consideration the data movement between assignments, as well as some of the limitations placed on efficiency metrics by portable consumers such as execution price and execution moment limit. In particular, each task can be either uploaded to the mobile cloud or executed locally on the mobile device within the application. Instinctively, tasks requiring frequent interaction with mobile users should be performed on the mobile device, and tasks requiring complicated computing and consuming large amounts of energy on hand should be uploaded to the mobile cloud. Finding the ideal upload choices for task scheduling is an NP-complete issue. However, a mature strategy to obtaining an ideal alternative is the heuristic smart algorithm. To fix this issue, we implemented the genetic algorithm (GA) in Existing. GA's are robust algorithms, including planning issues, to solve NP-hard global optimization issues. GA's are population-based algorithms that operate iteratively over the enormous search space to achieve better alternatives. To fix the issue of optimization, we suggested a genetic algorithm-based task scheduling (GABTS).

2.1.1 Disadvantages

- ✓ An energy consumption optimization issue, taking into consideration task dependence, data transmission and certain constraints such as time and cost of reaction.
- ✓ Increases both cloud size and communication time.
- ✓ The amount of GA iterations also rises with an increase in the number of tasks.
- ✓ Increasing the amount of assignments implies increasing the duration of the chromosome, resulting in a wider genetic diversity of the population. As a consequence, the number of iterations is increasing.

2.2 Proposed System

In this section, we look at some present work on the MCC issue of task scheduling. Usually, to save power consumption, speed up application execution, or save storage room, the mobile application is partitioned into several parts, understands as tasks, and then these tasks are partly scheduled to be executed in the mobile cloud servers. The goal of optimization fell primarily into two classifications, either minimizing the complete execution time also called, makespan, or minimizing energy consumption. Because the issue of scheduling tasks is NP-hard, most works take heuristic methods to address this issue, which cannot guarantee to find the optimal solution, but it can find almost optimal solution. In, authors propose a task scheduling approach to guarantee a better accessibility to cloud network and speed up the processing time in Mobile Cloud Computing, taking into consideration some constraints such as the Energy efficient and Overall QOS and QOE for cloud usage. However, the details on how to obtain some metrics such as earliest start time or earliest finish time of tasks are not offered, and the algorithm complexity is unknown. Some works pay attention to the kinds of resources which the nodes in the MCC can provide, and schedule the tasks to the nodes in MCC combining it and the information on the amount and kinds of requested resources tasks need for execution, so as to find the most appropriate scheduling scheme. The proposed algorithm for task scheduling depending on the metrics of quality of service (QoS), such as load balancing, average execution, makespan, delay sensitive, and deadline satisfaction. First, they calculate the results of the tasks according to the QoS, and then tasks with higher priority are scheduled on the nodes first. An effective workflow task scheduling algorithm depending on bandwidth availability. For other techniques, Evolutionary algorithms are implemented to assign tasks based on a nonlinear programming model to each node in the cloud. Some are suitable for uploading to the MCC for the assignments acquired by partitioning the request while some are not Over the past few years, a lot of attention has been given to selecting suitable tasks to upload and ensuring that the task-precedence requirements and the time limit to complete the application are encountered. This began from a minimally delayed scheduling solution to present an algorithm and then conducts energy reduction through the application of the method of dynamic voltage and frequency scaling.

2.2.1 Advantages

- ✓ The E2IQDTS algorithm can decrease energy consumption and completion time of application efficiently.
- ✓ Minimizing the joint energy consumption and completion of the application within the time limit and task-priority requirement.

2.3 Hybrid Scheduling Algorithms

Many of these algorithms are new or created on top of some c urrent techniques that incorporate more scheduling parameter s to enhance efficiency. Below are some of the current journals in this group: They schedule tasks for distinct resources on the basis of their cost. Service costs vary depending on their complexity for distinct duties. The algorithm takes resource costs and energy capacity handling into account. They collective processing capability-based duties and select some of the finest resources to schedule them to reduce costs. Compared to other planning methods, this algorithm decreases the makespan and processing costs. The importance of the task is calculated in order to schedule them on various resources. Priorities are calculated for the tasks depending on the distinct characteristics of the tasks and are sorted on that basis. They are then allocated to the machine producing the best time to complete. This algorithm therefore increases efficiency by getting greater Time to complete. The tasks are divided into different groups and replicated to the system's local middleware. It allows the scheme prepared for failure and load balancing increases the use of response time and resources. Here, Lexi query technique is used to plan the duties to different assets together with cost reduction. The assignment of the task is focused on a probabilistic estimate computed on the basis of the resource availability and the task's execution time. Load balancing in each resource decreases the costs generated at the scheduler. They create an algorithm centered on the traditional min-min algorithm that

involves scheduling depending on server usage as well as taking user preference into consideration. Users are categorized as VIP and normal customers in two classifications. Load is balanced depending on the system's highest energy load and makespan. The technique demonstrates a nice profit in the proportion of customer fulfillment, makespan and use of resources. They offer a altered algorithm called the dual weighted least algorithm of association. In this method, the server weights (processing capacity) are dynamically calculated and the server load is calculated based on the characteristics of the tasks assigned to that server. Compared to the WLC technique, the software provides greater load balancing and system efficiency. An algorithm based on the divisible load principle is proposed with the aim of reducing the overall processing time of the tasks. Here homogeneous processors are used to calculate the load fractions and processing period for each task. The countable load is partitioned between different servers, enabling the assignments to be completed as quickly as possible within a short time. This technique enhances the benefits of cloud suppliers as well as service quality. Compared to other experimental techniques, these technique outcomes in outstanding performance, total cost, delay cost, effectiveness. They propose an algorithm that is a change to the enhanced Evolutionary algorithm It is focused on the anticipated execution moment in which it assigns on the machine a task with average execution time that provides minimum completion time. The hardest task determines the system's makespan and sometimes it can be too big to increase the system's makespan and generate load imbalance. Therefore, instead of selecting the largest Task, they select the largest average task or the nearest average task. This technique creates a fairer balance of loads and makespan than an enhanced Evolutionary method.
The primary parameters of planning regarded in the technique s mentioned are listed below:

3.1. Cloud User Preferred

1) Makespan: says of the last task's completion time. A good scheduling algorithm always tries to reduce the makespan
2) Cost: it is the sum of the amount that the user has paid to the provider to use the individual resource.
3) Time to perform: This is the precise moment it takes to perform the tasks. The main goal of a nice scheduling algorithm is to minimize execution delay.
4) Time to finish: the moment required to finish the full execution of a task is the completion time. It involves the cloud system's execution period and delay. Many of the current planning systems consider minimizing the finishing period of assignments.
5) Time to wait: time spent in the ready queue for a task to get an opportunity to execute.
6) Turnaround time: Time taken to complete the task after it has been submitted, i.e. the sum of the waiting time and the execution time of the task.
7) Tardiness: the limit in the performance of the job, i.e. the gap between the time of completion and the deadline for the task. The tardiness should be null for an ideal planning that demonstrates no delay in execution.
8) Fairness: this demonstrates that there is equal opportunity for all tasks to be executed.
9) Response time: time requested by the scheme to begin reaction (first response) after the task has been submitted.

3.2. Cloud provider preferred

1) Resource utilization: resources should be fully used to keep them as active as necessary in order to achieve highest profit.
2) Throughput: it reflects the amount of tasks per time unit finished.
3) Predictability: this reflects continuity in task response times. Unpredictable response times can degrade system performance.
4) Priority: giving priority to the task of finishing it as soon as possible. Priority may be provided based on time of arrival, time of execution, etc. Resources are given to finish the execution of the higher priority task.
5) Load balancing: is the technique by which the entire load is distributed across different nodes in the cloud network through distinct nodes and connections, there is no loading of nodes and links while overloading certain nodes or links. Most scheduling algorithms are attempting to maintain the load balanced in a cloud network to boost the system's effectiveness.
6) Deadline: the period from the time the task is submitted to the time it must be finished is specified. A excellent algorithm for planning always attempts to maintain the tasks performed within the deadline.
7) Energy Consumption: Cloud information center energy usage is an ongoing problem that should be looked at more carefully these days. Many scheduling algorithms have been created to reduce power consumption and improve efficiency, thus greening the cloud services.
8) Performance: Performance shows the general effectiveness of the scheduling algorithm to provide customers with excellent facilities according to their needs. A successful planning system should weigh both the customer end efficiency and the start of the cloud service provider.
9) Quality of Service: Service quality involves many user input limitations such as cost of quality meetings, deadlines, quality, cost, makespan, etc. All are described as a contract document between cloud user and cloud service provider in SLAs.

4. ARCHITECTURE DIAGRAM

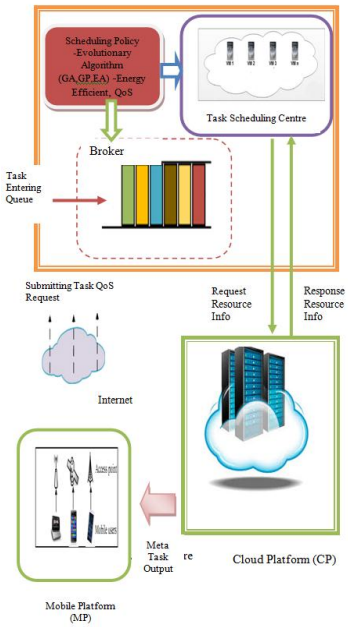

Figure 1 *Architecture Diagram*

Datacenter: It reflects the provider of the resource. It is accountable for handling the resources available, i.e. hosts, PEs, VMs, memory; on-premise hardware is a data center. It collects information within an organization's local network. Data centers are run by departments of in-house technology. Data centre has server memory ability. Data centre is a completely guaranteed program without third party interference. Broker: It is accountable for the mediation of the data center between the customer. It is the wishes of the user. It gives the planning cloudlets to the data center monitoring the cloudlet position and tells the customer of the current status of its demands; a broker that (1) gets all customer applications for computing; (2) handles processor registers (storage capability, network bandwidth) and computing expenses along with information request outcomes received from processors; and (3) Cloudlet: the customer demand (a cloud provider function) is represented. It is characterized by length, number of PEs (the number of PEs required to perform the cloudlet); Host: host is a physical resource characterized by a number of PE and RAM capacities (a computer); Virtual Machine (VM): it is a computer-based software emulation .Resource discovery and filtering: Data center Broker finds the assets in the network scheme and gathers resource condition data. Selection of resources: Target resource is chosen depending on specific task and resource demands. That's the phase that decides. Task assignment: Task assigned to the chosen resource

5. ALGORITHMS

5.1 Evolutionary Algorithms

Evolutionary algorithms are algorithms of stochastic search based on the natural selection strategy system. It begins with a collection of original approach, called the original population, and uses genetic operators to produce fresh solutions. The genetic algorithm strategy calculates the effect it will have on the scheme in future following the deployment of the new VM resource in the scheme, using historical information and present system status. The answer will then be picked up, which will have the slightest effect on the system. The benefit of this method is that it can manage a big search space that can be applied to complicated objective function and can prevent local optimal problem catching. Develop a cost-based scheduling task algorithm that provides a multi-objective QoS planning in the cloud computing environment.

Algorithm 1:

Step 1: Create the datacenter.

Step 2: Create VMs and allocate them to the host in datacenters according to the space shared schedule or timeshared schedule.

Step 3: Create the cloudlets and allocate to the VMs according to the space shared schedule or timeshared schedule.

Step 4: Create the broker and submit the cloudlet list to it.

Step 5: Start and print the result after the.

Algorithm 2

For Task Offloading - Distributed Multi-User Offloading Algorithm

Input: $\mathcal{K}, \mathcal{N}, \eta_n, C_\nu, c_\varepsilon, \rho_\nu, L;$

1: $\mathbf{s}_n^0$ is selected with probability $\frac{1}{|\mathcal{S}_n|}$ and MUs initialize their $Q_n^0(\mathbf{S}^0)$.

2: **for** $t = 1, \ldots, T_{\max}$ **do**

3: **for** $n = 1, \ldots, N$ **do**

- With probability $\epsilon_n^t \in (0,1)$, $\mathbf{s}_n^t$ is chosen with probability $\frac{1}{|\mathcal{S}_n|}$.
- With probability $1 - \epsilon_n^t$, each MU
  - With probability η_n, $\mathbf{s}_n^t \leftarrow \mathbf{s}_n^{t-1}$;
  - With probability $1 - \eta_n$, $\mathbf{s}_n^t$ is chosen with probability $\frac{1}{|\mathcal{B}_n(\mathbf{s}_{-n}^{t-1})|}$. If $|\mathcal{B}_n(\mathbf{s}_{-n}^{t-1})| = \emptyset$, $\mathbf{s}_n^t \leftarrow \mathbf{s}_n^{t-1}$.

4: **end for**

5: Each MU observes $\mathcal{U}_n^t(\mathbf{S}^t)$ in the "Result Returning" duration.

6: Each MU updates its Q-function by following (49).

7: **end for**

Algorithm 3:

Weisfeiler-Lehman Algorithm

1: procedure WEISFEILER-LEHMAN

2: for each tress t $\in T$ do

3: H $\leftarrow depth(\mathrm{Q})$

4: for h $\leftarrow 1; h \leq H; h++$ ***do***

5: for each node N $\in$ Q

6: s $\leftarrow CreateSet(label\ so\ f\{N\} \cup \sum N.children)$

7: if hashtable.get(s) $\neq null$ **then**

8: label(N) $\leftarrow hashtable.get(s)$

9: else

10: hashtable.put(s, label(N))

11: end if

12: if IsLea f($\sum N.children$) then

13: N.IsLea f = true

14: end if

15: end for

16: end for

17: end for

18: end procedure

6. EXPERIMENTAL RESULTS

This section introduces the simulation tests by using CloudSim 3.0.3 tool to show the effectiveness of the resource allocation strategies established in this chapter. Mean response time is the metrics used to evaluate the efficiency of task allocation strategies. This section includes the execution portion of this study involving the design and simulation method of the suggested algorithm using CloudSim 3.0.3 tool as well as the specifications for software and hardware for this experiment. Net beans, Java and CloudSim3.0.3 are the tools used for these studies. We created the suggested Java language engine. Then the simulation CloudSim3.0.3 is used to evaluate the engine created. First, one data centre, one scheduler and five virtual computers are used to evaluate the engine created. First, one data centre, one scheduler and five virtual computers are used to evaluate the proposed method. We also created evolutionary algorithms such as genetic algorithms, Java evolutionary programming using their pseudo coordinates for the intent of simulation. Because we attribute our suggested algorithm to the Genetic algorithm to test for superior results? Virtual machines are used for this evaluation in various dimensions. Table 1 demonstrates the efficiency

contrast between the algorithms proposed, Genetic Algorithm using different workflows with regard to makespan. The following figure indicates the graphic depiction of the outcomes where VM= 10.

Sr. No	Entities	Parameters	Values
1	User	No of users	5
2	Cloudlet	No of cloudlets	100–1000
		Length	2000
3	Host	No of Host	2
		RAM	2048MB
		Storage	1000000
		Bandwidth	10000
4	Virtual Machine	No of VMs	15
		Type of Policy	Time Share
		RAM	512MB
		Bandwidth	1000
		MIPS	1000
		Size	10000
		VMM	Xen
		Operating System	Linux
		No of CPUs	2
5	Data Center	No of Data Centers	2

Table 1. Simulation Parameters setting of cloudsim

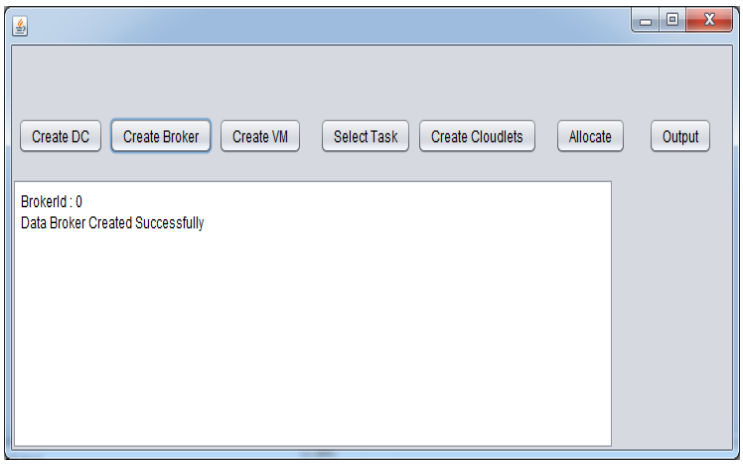

Figure 2 *creating Data Centre and Broker*

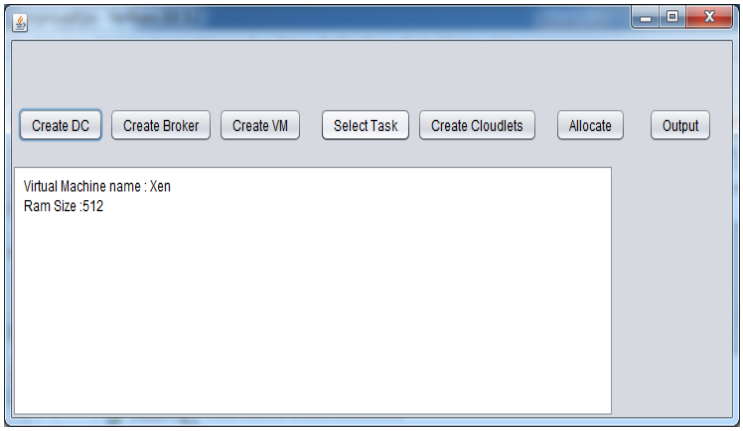

Figure 3 *creating Virtual Machine*

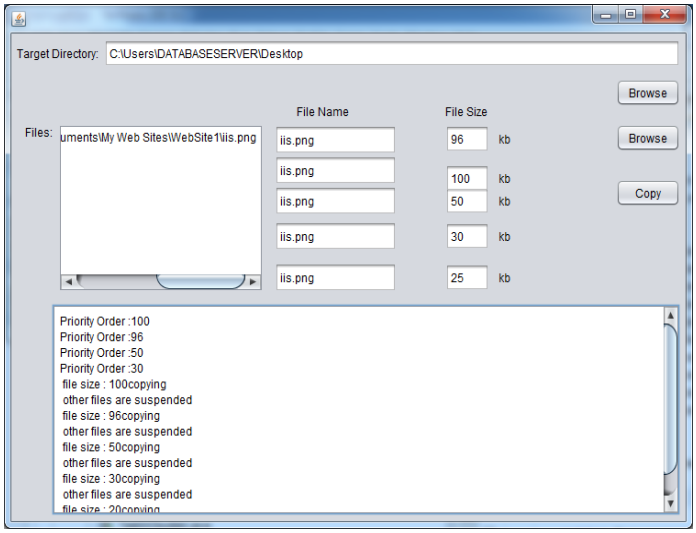

Figure 4 *Assign Task and Schedule Task based Priority*

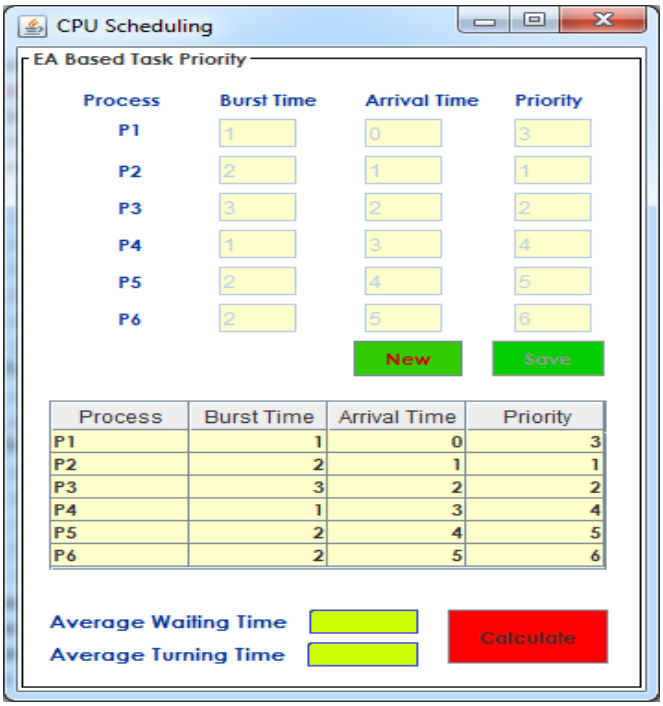

Figure 5 *CPU Scheduling Based on Waiting and Arrival Time*

Formula for Waiting Time
(Waiting time – Arrival Time)
Formula for Waiting Time
((Waiting time – Arrival Time) + Burst Time):

Waiting	Arrival Time	Waiting Time	Turning Time
0	0	(0-0) =0	(0-0) +1=1
1	1	(1-1) =0	(1-1) +2=2
0	2	(0-2) =-2	(0-2) +3=1
6	3	(6-3) =3	(6-3) +1=4
7	4	(7-4) =3	(7-4) +2=5
9	5	(9-5) =4	(9-5) +2=6

Total Waiting Time 8 Total Turning Time 19

Average Waiting Time = Total Waiting Time / 6
= 8 / 6
= 1.333

Average Turning Time = Total Turning Time / 6
= 19 / 6
= 3.166

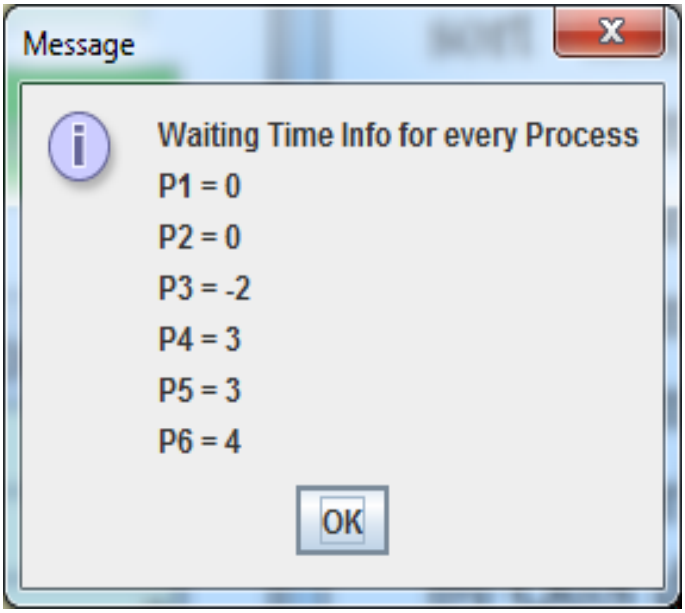

Figure 6 *Calculating Waiting Time*

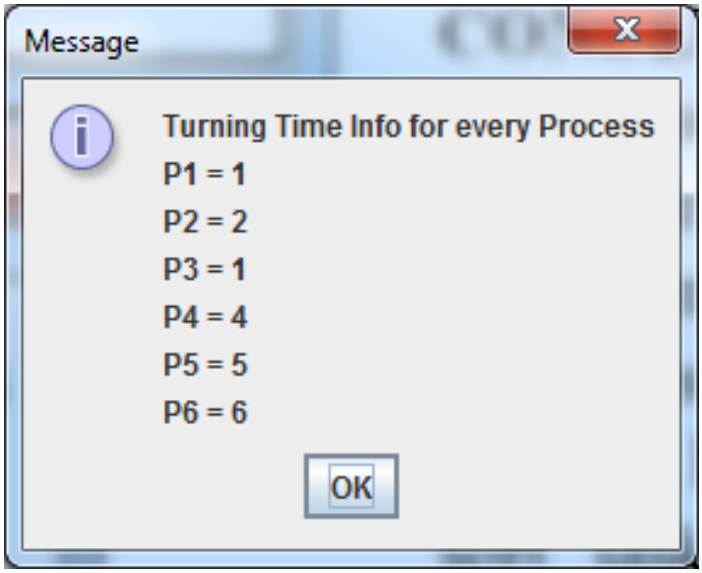

Figure 7 *Calculating Turning Time*

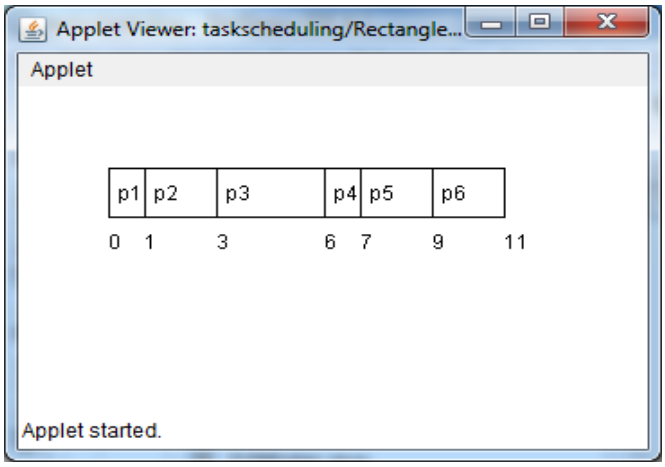

Figure 8 *Gantt chart*

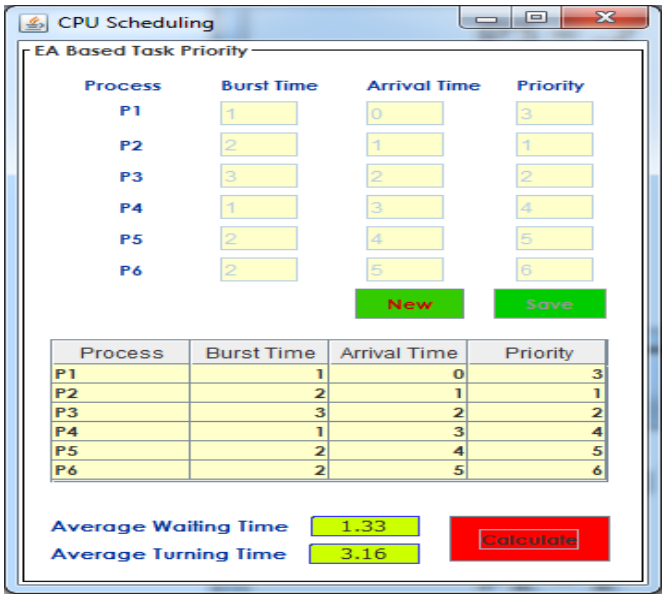

Figure 9 *Calculating Avg Waiting & Turning Time*

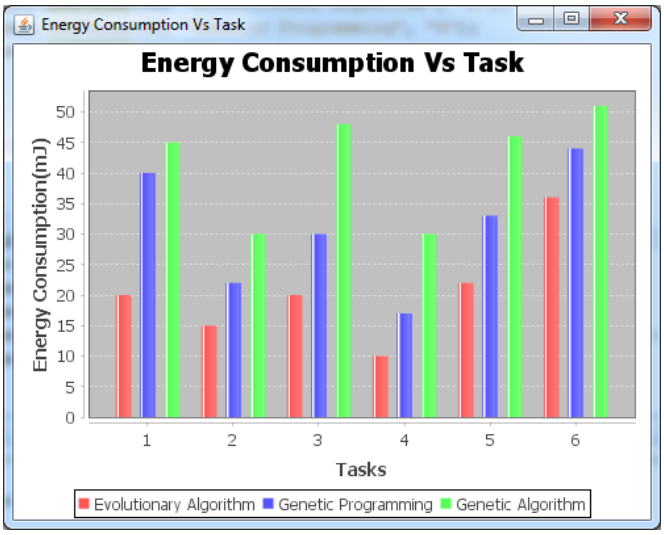

Figure 10 *Energy Consumption Vs Task*

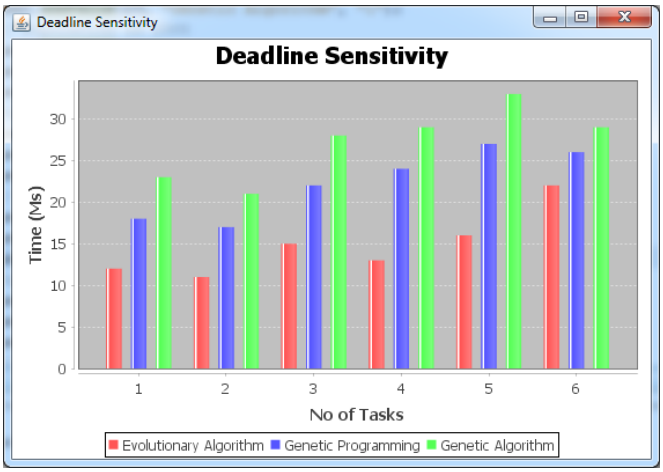

Figure 11 *Deadline Sensitivity Vs Task*

Data Centre Size	Genetic Algorithm (MS)	Genetic Programming (MS)	Differential evolution (MS)
VM=10	228.94	198.5	94.73
VM=30	241.49	202.37	98.36
VM=50	298.32	217.3	115.23
VM=100	314.43	234.43	134.32

Table 2. Three Algorithm Comparison Table

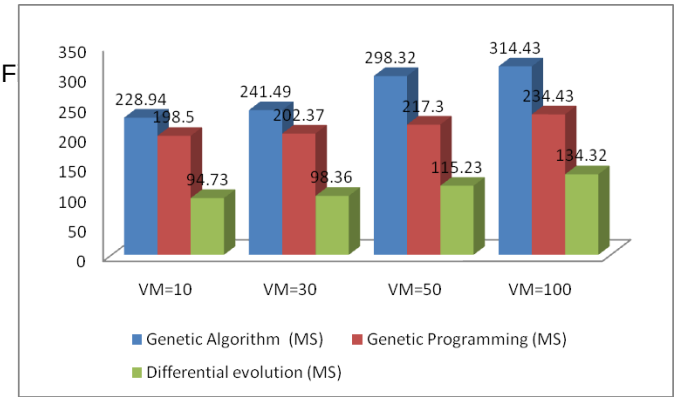

Figure 13 *Comparisons of Three Algorithms*

Results of the above assessments indicate that the suggested algorithm performs assignments with reduced makespan and greater efficiency relative to Genetic and Differential Evolutionary scheduling algorithms. The performance of the proposed algorithm is better than the genetic algorithm and the algorithm for 10, 30 and 50 and 100 virtual machines for genetic programming and differential evolution. Results show that after experimenting it with CloudSim Simulator, the suggested algorithm performs faster in terms of makespan. As we have improved a amount of virtual machines used for simulation, the overall planning duration for differential evolution is still lower than the Genetic Algorithm and Genetic Programming.

7 CONCLUSION

In this work, an energy-efficient and improved QoS –driven task scheduling protocol for portable Mobile cloud computation was suggested. The proposed algorithm blends many task attributes, including Deadline-satisfied characteristics, and Task Length, Priority, Delay-Sensitive in queue to prioritize the priority and type task. Comparing here Evolutionary algorithms such as Genetic Algorithm (GA), Genetic Program (GP) and Differential Evolution (DF). These algorithms schedule each task on a system that has a minimum time to complete. The test outcome demonstrates that the algorithm achieves both priority and deadline-satisfied performance and load sharing via QoS driving.

8.1 Future Enhancement

For potential Task, the impact of other parameters like VMs, datacenters, memory, bandwidth and storage in mobile cloud settings should be investigated and expressed in actual physical environment. To deliver a cost-effective alternative that looks to be easier relative to other existing methods. In order to obtain greater efficiency and greater results, we will quickly expand the suggested model to operate in different conditions. To create this grid system algorithm and observe the time difference in the cloud and grid of mobile devices.

REFERENCES

[1] Conti, Marco, et al.: Future Internet research difficulties. Communications from computers, 34(18), 2115–2134 (2011).

[2] Carthik, Kumar, and Y. H. Lu. Cloud Mobile User Computing: Can Computer Offload Save Energy? Computer, 51-56 (2010), 43(4).

[3] Satyanarayan, Mahadev, et al.: Mobile Computing Case for VM-based Cloudlets. Pervasive computing of IEEE, 8(4), 14-23(2009).

[4] Xia, W. Liang, Qiufen, and W. Xu.: Maximizing the performance of internet application admissions in portable cloudlets. In: Local Computer Networks (LCN), 38th IEEE Conference, pp. 589-596. Sydney(2013)

[5] M.R. Garey & D. S. Johnson,: Computers and intractability: a NP-Completeness theory guide. New York: W.H. Freeman, 1979

[6] Hung P P, Bui T A, Huh E N: A New Mobile Cloud Computing Task Scheduling Optimization Approach. Electrical Engineering lecture notes, 301, pp.211-220 (2014).

[7] Chen L, Li Y, Zhao Y, et al. RAS: A Resource Attribute Selection Task Scheduling Algorithm in a Task Scheduling Framework. In: Internet and Distributed Computing Systems International Conference, pp. 106-119, Berlin (2013).

[8] Deng S, Huang L, Wu H, et al.: Mobile Cloud Computing Constraints-Driven Service Composition. In: IEEE International Web Services Conference, San Francisco (2016), pp. 228-235.

[9] Wang J, Tang J, Xue G, et al.: Smartphone planning of energy-efficient tasks in portable audience detection devices. Computer Networks, pp.100-109 (2017), 115(C).

[10] Awad, A.I., El-Hefnawy, N.A., Abdelkader, H.M.: Improved Particle SwarmOptimization for Cloud Computing Environments Task Scheduling. Computer of the procedures. Sci. 65, (2015) pp. 920–929.

[11] Chen, H., Zhu, X., Guo, H., Zhu, J., Qin, X., Wu, J.: Towards energy-efficient real-time planning in a cloud-based environment. J. Syst. Softw., 99 (2015), pp. 20–35.

[12] Wu, X., Deng, M., Zhang, R., Zeng, B., Zhou, S.: a cloud-driven QoS-based scheduling algorithm. Computer Procedia, 17, pp.1162–1169(2013).

[13] Lee, Y.C.; Wang, C.; Zomaya, A.Y.; Zhou, B.B. Profit-driven data access sensitivity planning for cloud services.

J. Distribute in parallel. Comput., 72 (2012), pp.591–602.
[14] Panda, S.K., Gupta, I., Jana, P.K.: Heterogeneous Multi-cloud Systems Allocation-aware Task Scheduling. Computer Procedia, 50, pp. 176–184 (2015).
[15] Razaque, A., Vennapusa, N., Soni, N., Janapati, G.: Cloud computation tasks. In: Long Island Systems, Applications and Technology Conference (LISAT) proceedings of the IEEE, pp.1-5, Farmingdale(2016).
[16] Sindhu, S. Cloud Task Scheduling. Int. J. Adv. Res. Comput. Eng. Technol. Last year, 4, 3019–3023.
[17] Mahmood A, Khan S.: Hard Real-Time Task Using an Adaptive Genetic Algorithm in Cloud Computing. Computers (2017), 6(2):15.
[18] Lin X, Wang Y, Xie Q, et al.: Mobile Cloud Computing Environment Task Scheduling for Dynamic Voltage and Frequency Scaling for Energy Minimization. IEEE Services Computing Transactions, 8(2), pp. 175-186 (2015).
[19] Guo S, Xiao B, Yang Y, et al.: Mobile web computation for energy-efficient vibrant downloading and asset planning. In: The International Computer Communications Conference of IEEE INFOCOM, pp. 1-9, San Francisco (2016).
[20] Tsai, J.T., Fang, J.C., Chou, J.H.: Optimized cloud computation system job planning and resource allocation using enhanced differential development algorithms. Comput. Oper. Res. 40 (2013), pp. 3045–3055.
[21] Liu, S., Quan, G., Ren, S.: On-line profitable and punishment planning of real-time facilities. In: Proceedings on Applied Computing at the 2011 ACM Symposium, pp.1476–1481, Taiwan (2011).
[22] Kim, K.H., Beloglazov, A., Buyya, R.: Power-aware delivery of real-time cloud services to virtual machines. Concurr. Comput. 23, pp. 1491–1505.
[23] Deniziak, S.; Ciopinski, L.; Pawinski, G.; Wieczorek, K.; Bak, S. Using evolutionary genetic programming to optimize real-time cloud applications. In the IEEE / ACM Seventh International Conference on Utility and Cloud Computing (UCC) Proceedings, pp. 774–779, London(2014).
[24] Holland J H. Natural and synthetic system adaptation [M]. Press from MIT, 1992.
[25] Goldberg, David E. Search, Optimization and Machine Learning Genetic Algorithms. xiii.7(1990):2104–2116.