ISSN 2347 - 3983

Volume 8. No. 8, August 2020

International Journal of Emerging Trends in Engineering Research

Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter20882020.pdf

https://doi.org/10.30534/ijeter/2020/20882020

A Novel Approach to Comparative Analysis and Performance Evaluation of Energy Aware Multi-Layered Job Scheduling in Heterogeneous Cloud

Vinod Kumar Saroha[1], **Dr. Sanjeev Rana**[2]

[1]PhD Research Scholar, CSE, MMDU, Mullana; India; Email: vnd.saroha@gmail.com

[2]Professor, CSE Department, MMDU, Mullana; India; Email: Dr.sanjeevrana@yahoo.com

ABSTRACT

One of the latest apt platform in today's trending technological scenario for imparting application based utilities and services rendered by various distributed resources situated remotely is the Cloud Computing paradigm. The routine task operations and services involve efficient energy utilization with minimum dissipation via load balancing and process allocation. A heuristics defined approach has been implemented in this research work for execution of task scheduling activities with optimal resource sharing. Here a robust scheduling technique is used by the scheduler in the heterogeneous cloud network for mapping the available resources to execute scalable tasks optimally. The research area takes into consideration various performance based parameters like Make-Span, Throughput, Average Response Time (ART) etc. for analysis and comparison with the standard scheduling procedures. There was a large energy loss using the previous scheduling and load balancing standard algorithms in reference to the above mentioned parameters. Hence a algorithm named Energy Efficient Multi layered Scheduling (EEMLS) is proposed that outperforms the earlier algorithms in common practice; viz. Max-Min, Round Robin, Opportunistic Load Balancing (OLB), Artificial Bee Colony (ABC) and Minimum Completion Time (MCT) by containing the energy loss considerably during the process work flow of task execution. The entire scenario is best implemented in cloud environment using the Cloudsim simulator for obtaining the results to show better performance with energy saving.

Key words: Average Response Time, Load Balancing, Make-Span, Multilayer Scheduling, Resource Allocation, Throughput

1. INTRODUCTION AND RELATED WORK

The latest emerging paradigm in the technological world has been the cloud computing; that incorporates different virtual machines, data center units, task schedulers etc. for the efficient process execution on different machines scattered at remote locations [4]. The large number of virtual machines (VMs) [5] are the vital units of processing and are placed statically in the data centers. The cloud computing leaves a substantial carbon foot prints during the routine operations execution and there is further a considerable energy loss in the workflow. The load balancing in sync with the effective task scheduling and resources allocation check the energy wastage to a great extent. The job scheduling approach in cloud computing network helps in the allocation of the desired resources to the available tasks for the optimal performance and better resources utilization. At any particular instant of time; a desired resource is assigned for specific jobs. The CPU utilization, the bandwidth, the execution time, the memory storage space etc are certain types of resources which are time and again accessed by the scheduler.

1.1 The important categories of task scheduling are best explained below:

Dynamic Scheduling: This type of scheduling is more flexible than the static one; as here the execution time of any process is known in advance to the scheduler. The jobs here are allocated to the resources by the scheduler for a particular duration of time. A accurate and stable algorithm supported with the load balancing features helps in the execution of different tasks [18].

Static Scheduling: Here the information regarding the jobs execution and the availability of resources is known in the beginning when the jobs are scheduled. The scheduler schedules the task to the appropriate resource; depending on the capability of the resource [11].

Centralized Scheduling: As mentioned in the dynamic job scheduling, it is the responsibility of distributed or the centralized scheduler to take a call globally. The major benefits and advantages of the centralized scheduling are its ease in implementing, utilization of the available resource optimally, overall control on all the shared resources with convenience of implementation, output performance and efficiency.

Distributed or Decentralized Scheduling: In this type of scheduling, there is no centralized controlled domain entity and hence the scheduling is less strong and less effective when compared with the centralized scheduling. The schedulers that control and manage the task requests in a queue are the localized ones which adapt to the number and type of the resources available.

Pre-Emptive Scheduling: Here, interruption of every job takes place; during the migration and execution of jobs from one resource to another resource. For the allocation of new job requests to the original resource; it is left idle. Pre emptive scheduling is best suited for the jobs where the priority limits are attached to requests.

Co-operative scheduling: Here in this type of scheduling process, there are many schedulers attached with the system [14]. Depending upon the fixed set of algorithms, rules of framework and is the system users; every scheduler is granted with the onus to perform a particular activity that leads to the overall efficiency and enhance output performance.

Non Pre-Emptive Scheduling: In this type of scheduling, initially the execution of the scheduled job requests is allowed to finish of its cycle; before that there is no provision of the re-allocation of the shared resources [13].

Batch / Offline Mode: Here in this type of scheduling process; the job requests are executed at specific intervals of time; which are consecutive in nature. These requests are saved as a common group of problems and executed at right instance.

Hybrid scheduling Model: In this type of scheduling both the Static scheduling mode as well as the Dynamic scheduling mode are combined with each other. This model caters primarily to the virtual machines of the system.

An efficient job scheduling approach aims in completion of the task in stipulated duration of time thereby taking less response time. Hence there is always requirement of timely access and reallocation of the shared resources among the varied jobs. In such approach more tasks are submitted to the cloud center for execution with fewer rejections [14]. The overall exercise helps in the smooth energy workflow for the execution of job requests at the cloud center and hence the better performance.

1.2 Some of the standard task scheduling algorithms employed in cloud computing environment are as below:

Round Robin (RR) algorithm: This type of algorithm is static in nature and is a popular for its load balancing feature. The round robin technique is followed here for the allocation of jobs to the resources. initially a primary node is chosen at a random; after then the jobs are assigned to the available resources in the round robin fashion. The tasks are assigned to the processors in a sequential circular manner; if there is no priority attached. The unit quantum is used for fixed time interval for different executable processes.

Min-Min Algorithm: This algorithm aims at addressing the sorting of the unmapped tasks in the order of the ascending order of their completion intervals of time. The longer duration tasks are delayed while the smaller tasks are executed in the beginning. The process is repeated over time such that all the tasks of the unmapped set are mapped for the corresponding available resources for sharing [1] [2].

Minimum Completion Time (MCT): Here, in this type of scheduling procedure, that specific task is assigned to the desired resource in random fashion; which has the value of completion time as minimal. However, the striking difference between the Min-Min and MCT is that during the time of mapping decision; Min-Min considers all the unmapped tasks where as the MCT recognizes a single task a specific time interval [9].

Max-Min Algorithm: Here in this type of algorithm, the smaller tasks of less time duration are delayed for longer; while the larger task is scheduled for its execution initially. In this procedure; the tasks are sorted in their descending order of the completion time and is exactly opposite to the Min-Min procedure in this respect [2].

Multi-level Queue Scheduling Algorithm: In this type of procedure; the initialized queue is demarcated into n number of queues; with different tasks in multi level scheduling [15]. The whole exercise involves the use of a simulator for assembling scheduling tasks and strategies. It takes into consideration both the non-preemptive and preemptive methods.

Shortest Job First Algorithm: In this algorithm; waiting time of task scheduling is optimized; while taking into consideration a non-preemptive scheduling approach. This procedure is best employed in the batch type processing systems; where the CPU timings are known from the start. It is rather difficult to be implemented in the interactive systems as CPU time is not known; but conveniently used in batch processing systems [12].

Opportunistic load balancing algorithm: In this type of procedure; the present workload of the virtual machines is not taken into consideration; hence it is a static feature dependant load balancing algorithm [17]. This algorithm is based on the approach of keeping the nodes busy; while the jobs are allocated to particular nodes following the random fashion. Here the leftover unexecuted tasks are transferred to the available node randomly.

Artificial Bee Colony (ABC) Algorithm: Here In this ABC procedure, a generic approach is being investigated in reference to the bees in search for their food source (nectar). The position of the nectar food source provides an optimal solution for which the quality of solution is most precise. At the onset; scattered food positions are created where the food source is randomly distributed [16]. The search operations are carried by the scout bees, the employed ones and the onlookers as well; for locating the nectar. The employed bee traces the new nectar position; while it forgets the old food source position. These new location of nectar or food source are evenly explored for the customized results.

There is considerable a large energy consumption from the process workflows and routine operations in a cloud computing network. The earlier research done in this area failed to bring the satisfactory results in terms of the standard parameters of Response Time, Makespan and Throughput. Also they resulted in large energy loss. Our improvised EEMLS algorithm addressed to this area of concern and provided optimal solution with results that rendered less energy dissipation as well as augmented the performance in the cloud data centers.

Remaining part of the paper is divided into further sections or sub parts for ease of reading. Section-1 explains the task scheduling, its introduction, types of scheduling and urgency of the EEMLS algorithm. Section 2 describes the literature work regarding the problem. The EEMLS approach has been elucidated in section 3 while section 4 gives the implementation details and section 5 studies the evaluation and performance analysis of the existing algorithms with the proposed EEMLS algorithm. Section 6 and 7 explains Conclusion and References part respectively.

2. LITERATURE SURVEY

Lot of research work has already been accomplished in the area of task scheduling; our work prolongs the existing work further.

Akilandeswari. P and H. Srimathi (2016): The authors in their research work presented the utility advantages of the subscription services; based on pay for use idea. The services existed in all forms of cloud computing; in the distributed, parallel or cluster based cloud computing models. Scalability, elasticity and better performance are some of the important features offered by the cloud computing services. The user subscribes to the services of a cloud service provider and can customize, deploy his software to pay for the subscription charges. The scalability helps in increasing the complexity of the task scheduling in the cloud computing whereas, the elasticity caters to different resources viz. storage, memory, CPU etc. the throughput has been observed to be effected by many scheduling algorithms; scalability helps the scheduling to complete the complex task executions at different times.

Sushil Kumar Saroj, Aravendra Kumar Sharma (2016): The authors here in this study; gave the presentation how the shared resources are efficiently used in the data centers with optimal CPU scheduling for the processes that help to augment the system performance and the output. The CPU is transferred among varied processes for the desired resources. But there are some drawbacks visible in parameters viz. starvation, turnaround time, higher average waiting time etc. have found difference in their values when implemented in the real time conditions. Here both the variable and average time constraints are taken into consideration; while feeding some processes with the average time quantum while some with the variable time quantum and then comparing the results.

Rajveer Kaur et al.,(2014): With context to this paper, the authors have presented the problems involved with heavy load and traffic congestion. In such problems; the effective job scheduling plays a very important role for the solutions related to traffic or load congestion. The exercise of task mapping is performed depending upon the type and nature of resource. The earlier methods of cloud networking, task scheduling and resource allocation are discussed along with their comparative analysis.

Shridhar Domanal, Ram Mohana Reddy Guddeti, and Rajkumar Buyya (2016): The authors here presented a novel approach towards managing the shared resources effectively and a robust job scheduling in the cloud computing environment. For this, they proposed a new bio-inspired algorithm. In the traditional type of job scheduling algorithms example first come first serve, round robin, max-min, min-min, ant colony optimization etc. multiple job requests per instances were submitted to the cloud centers to allocate the desired resources intelligently. But in this study; MPSO algorithm (Modified Particle Swarm Optimization) is employed for the allocation of resources to the virtual machines in efficient manner and the hybrid bio inspired algorithm helps in allocating CPU, memory as desired by the processes. Hybrid bio inspired algorithm may be considered as the combination of modified versions of CSO and PSO. Regarding the parameters of average response time, processor utilization and reliability; the hybrid Bio Inspired algorithm outperforms the other conventional forms of algorithms viz. CSO, ACO and round robin.

3. ENERGY AWARE FRAMEWORK IN HETEROGENEOUS ENVIRONMENT

The principle on which the Energy Efficient Multi layered Scheduling (EEMLS) works in smooth workflow execution of processes is that the job request from the client with less time duration is assigned the highest priority [19] will be sent to the most efficient resource (Highest configured server). This paper represents the effective job scheduling paradigm in multi-layer format as the servers are prioritized on basis of their configuration whilst the client requests are prioritized in different layers categorized on the basis of their job processing time.

3.1 EEMLS Algorithm 1 (Multi layered client-server establishment with Scheduling)

1. Establishment of server-client database with different configuration on the cloud in the beginning.

2. The parameters viz. processor types, processing speed, RAMs and hard disk storage classify the servers.

3. The processing need (time) prioritises the client jobs. Here, maximum five jobs are send by the client of different or same processing time.

4. Hence on basis of processing need (time); the client requests are classified as highest, middle and of lowest priority.

5. The highest configuration server attends the client job request of lesser processing time.

6. The servers are assigned priorities on basis of their configurational parameters; i.e. the highest priority is assigned to the server with higher configurations.

7. Initialise Server_1 on the datacenter of cloud.

8. Accept the Client_IDs, jobs number, and processing (speed) time for every job.

9. Scheduler initializes scheduling with categorization of jobs in three layers depending their processing time.

10. Particular server is allocated a job; depending upon their priorities.

3.2 EEMLS Algorithm 2 (Load Balancing with Energy efficiency)

1. Load balancing is executed after the scheduling of client job requests on cloud datacenter.

2. The load of the high configuration servers (If over loaded) is migrated to the nearby server having configuration (priority) just less.

3. The job requests assigned to the under loaded servers are transferred to the nearby server that has enough capacity to handle that request.

4. The critical job requests are entertained first with energy saving in the overall processes workflow is achieved by the migration of job requests among overloaded and under loaded servers invariably.

5. Exit.

The requirement and availability aspects determine the allocation of resources. Here, in our research work undertaken, the capping of server is done in a way that a server can process five job request with optimal resource utilisation. The job requests get migrated to next immediate available server for more tasks and processes in the pipeline. There is uniform distribution of the tasks assigned keeping in view their priorities, resource availability and requirements checks overloading and results in energy saving. The server is multilayered because of different configuration parameters of processing speed and memory; while the client is multilayered due to its classification of job requests.

4. IMPLEMENTATION WORK USING PROPOSED EEMLS ALGORITHM

The proposed framework uses the client-server model approach in layered structure wherein the Java and Cloud-Sim acts as the frontend and backend respectively.

At the very onset; there is establishment of Server-Client database in the cloud environment. There is categorization of servers on the basis of internal memory, processing speed and time. Whereas the processing need (time) of the client job requests classifies the clients on the basis of their job requests in three different priorities; viz. moderate priority, of high (Critical) priority, and with lowest priority. The servers are however categorized as lower, intermediate and of higher priority on the basis of their configuration parameters.

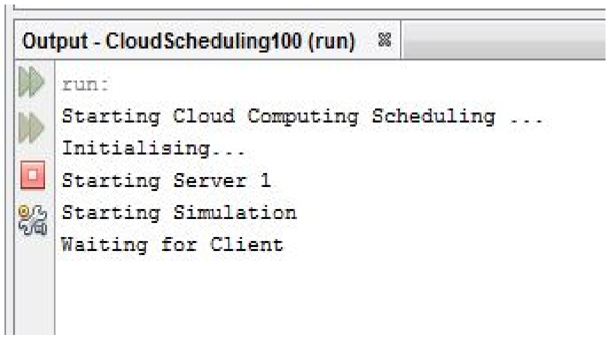
In our work, the server capacity to execute client job requests is five for the successive clients; while we have considered five servers for initial demonstration. Hence, for better interpretation of scalability results; the server numbers are increased to 10 for 50 task execution; 40 for 200 tasks execution, 100 for 500 tasks execution, 200 servers for 1000 execution and so on. In this manner, the tasks scheduling with execution is performed for 15, 50, 100, 250, 500, 1000 and 2000 tasks respectively. For even large number of tasks execution; the same may hold true; following the heuristic approach for scalable tasks. For client side in our work; a client is capable of sending 5 job requests of different processing speed (time units taken). If here are more than five job requests; than the same are transferred to the nearby servers which balances the load distribution and saves energy.

The servers of different configuration that are uniformly distributed, are considered in our implementation work. The processing speed of servers is 4.0 GHz, 3.2 GHz, 2.5 GHz for the same type of servers placed in same type group with similar processors i.e. i7 or i5 or i3 configuration servers. For similar type servers placed in same type; RAMs of 8TB, 4TB or 2TB is taken into consideration.

Our proposed EEMLS algorithm involves 5 client requests execution for a specific server. That is for example, in a 500 task execution; 100 servers will be utilised where every client ID will send 5 job requests. The ratio of tasks execution to the server deployed is 5:1.

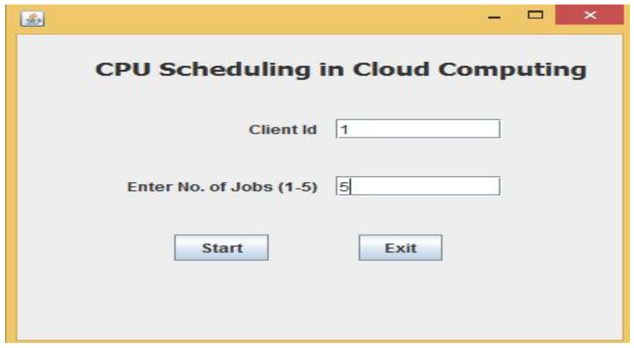
The initialisation of the client-server database is shown in the Fig.1 screenshot. This initialisation entails the server_1 to accept the first request from Client_1 ID for 100 task execution.

SNAPSHOTS OF WORKING:



```
run:
Starting Cloud Computing Scheduling ...
Initialising...
Starting Server 1
Starting Simulation
Waiting for Client
```

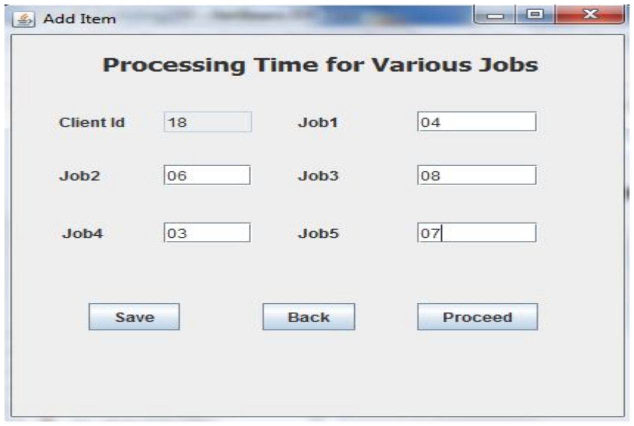
Figure 1: Initializing the 100 tasks execution in heterogeneous cloud



CPU Scheduling in Cloud Computing

Client Id	1
Enter No. of Jobs (1-5)	5

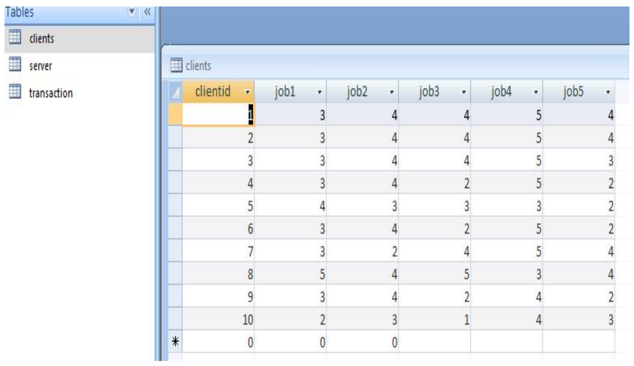
Figure 2: Client login screen with Type client id and no. of jobs to be processed.



Processing Time for Various Jobs

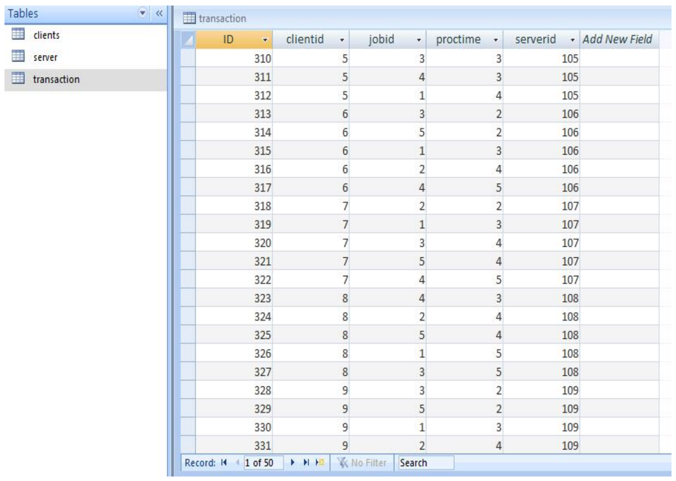
Client Id	18	Job1	04
Job2	06	Job3	08
Job4	03	Job5	07

Figure 3: Processing time for different jobs of client id_18



clientid	job1	job2	job3	job4	job5
1	3	4	4	5	4
2	3	4	4	5	4
3	3	4	4	5	3
4	3	4	2	5	2
5	4	3	3	3	2
6	3	4	2	5	2
7	3	2	4	5	4
8	5	4	5	3	4
9	3	4	2	4	2
10	2	3	1	4	3
0	0	0			

Figure 4: Implementation table for first 10 clients in 100 tasks execution (Above)



ID	clientid	jobid	proctime	serverid	Add New Field
310	5	3	3	105	
311	5	4	3	105	
312	5	1	4	105	
313	6	3	2	106	
314	6	5	2	106	
315	6	1	3	106	
316	6	2	4	106	
317	6	4	5	106	
318	7	2	2	107	
319	7	1	3	107	
320	7	3	4	107	
321	7	5	4	107	
322	7	4	5	107	
323	8	4	3	108	
324	8	2	4	108	
325	8	5	4	108	
326	8	1	5	108	
327	8	3	5	108	
328	9	3	2	109	
329	9	5	2	109	
330	9	1	3	109	
331	9	2	4	109	

Figure 5: Shows the part of transactions performed in 100 task execution (Server ID 310-349)

5. PERFORMANCE EVALUATION OF EEMLS ALGORITHM

The efficacy of our EEMLS has been analyzed by comparison with the standard laid algorithms viz. Round Robin procedure, Min-Min, Max-Min, Artificial Bee Colony etc. algorithms; and by involving different parametric values of Make Span, Throughput and Average Response Time in the heterogeneous cloud conditions is discussed in the subsequent sections ahead.

5.1 MAKESPAN PERFORMANCE METRICS

Make-span parameter may be defined as the maximum time of jobs completion out of the total number of the job requests received per unit time for tasks execution in the cloud environment. The ultimate aim of any evaluation system is to keep CPU in busy mode. This parameter is responsible for ascertaining the time used in job execution [16] and also determines the quality feature of the jobs allocated to available resources. Mathematically it is represented by the following expression:

Makespan = max (rtj); where the ready time of every scheduling resource is denoted by the term “rtj”. For optimal efficiency of any scheduling algorithm; the value of Make-span needs to be minimum. Further mathematically, it may be stated as;

Makespan = Max. { KTj | ¥ j are the elements of J}

where; KTj = The finishing time calculated for job j; that further is element of Job list “J”

j = represents a specific job from the entire jobs list of pool, while J = Entire list of jobs for execution

Table 1: Make Span time metrics comparative analysis using table

Algorithm	Make Span Time
Round Robin	24.5 ms
MAX-MIN	18.0 ms
ABC	17.8 ms
EEMLS (Proposed)	15.5 ms

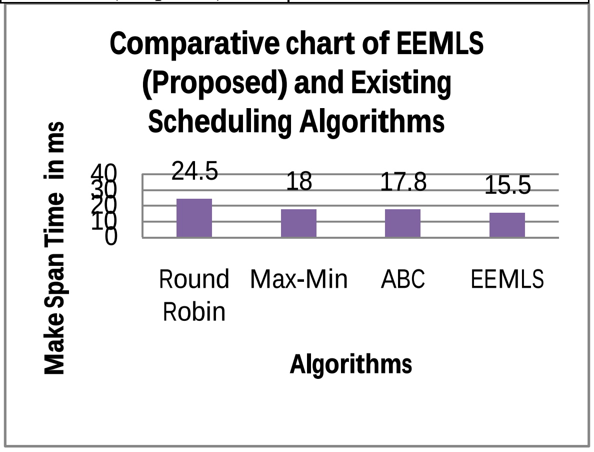


Figure 6: Make Span time metrics comparative analysis for different scheduling techniques using graph

It is very conclusive after noticing the above figure 6 and table7 that for the parameter of Make-span; our EEMLS procedure clearly outperforms the Artificial Bee Colony Optimization, Round Robin and Max-Min algorithms.

5.2 AVERAGE RESPONSE TIME (ART) PARAMETER

The Average Response Time or ART parameter is the total time duration between the time of making a job request to the time of delivery of response for that request. It helps in the measurement of the processing time of the submitted jobs. For optimal resource utilization; it is always desired that both CPU utilization and server throughput ought to be highest, where as the response time factor may be lowest.

The basic optimization criteria commonly deployed is as follows:

- Maximum CPU utilization
- Minimum Response time
- Maximum Throughput

Table 2: The average response time metrics comparative table analysis

Algorithm	Average Response Time
Round Robin	364.85 ms
ABC	362.67 ms
MAX-MIN	360.11 ms
EEMLS (Proposed)	357.5 ms

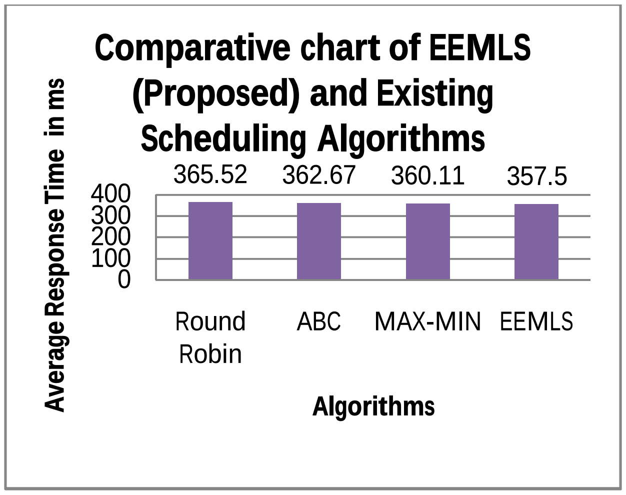


Figure 7: Average Response Time metrics for different algorithms using graph

It is clear from the above; that for parameter Average Response Time; our EEMLS procedure clearly outperforms the Artificial Bee Colony Optimization, Round Robin and Max-Min algorithms effectively.

5.3 THROUGHPUT PARAMETER FOR COMPARATIVE ANALYSIS

One more parameter for comparative analysis and performance evaluation is Throughput. It is the measure to find the numbers of (jobs) processes are going to complete in given unit of time. For smaller time duration processes; the throughput is calculated per second basis; while for the lengthier processes/jobs; the throughput is measured on hourly basis. The throughput includes all such jobs those have reached their deadline time of execution or those that have been availing the services of resource and have reached the finishing time. This parameter reflects the efficiency of the overall processes that have been counted in light of their execution or finishing time attained. Mathematically;

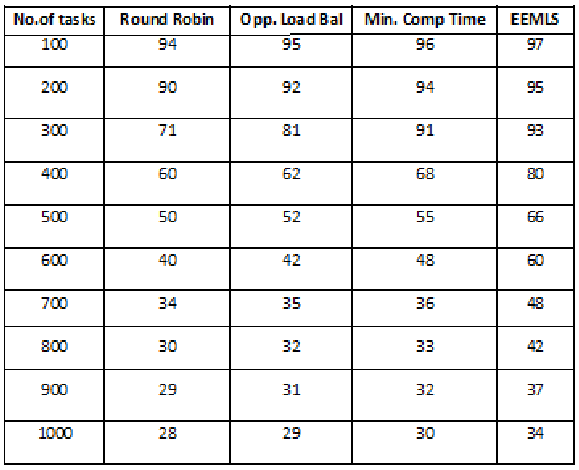
Vj = {1, if j job has finished of its execution}

= {0, if j job has not finished its execution}

Throughput J = ∑ Vj; ¥ j €J ;

In which; j = a particular job from the pool of jobs; and, J = List (Pool) of jobs (Denotes the number of jobs)

Table 3: Throughput metrics to show values of EEMLS algorithm with existing algorithms.



No.of tasks	Round Robin	Opp. Load Bal	Min. Comp Time	EEMLS
100	94	95	96	97
200	90	92	94	95
300	71	81	91	93
400	60	62	68	80
500	50	52	55	66
600	40	42	48	60
700	34	35	36	48
800	30	32	33	42
900	29	31	32	37
1000	28	29	30	34

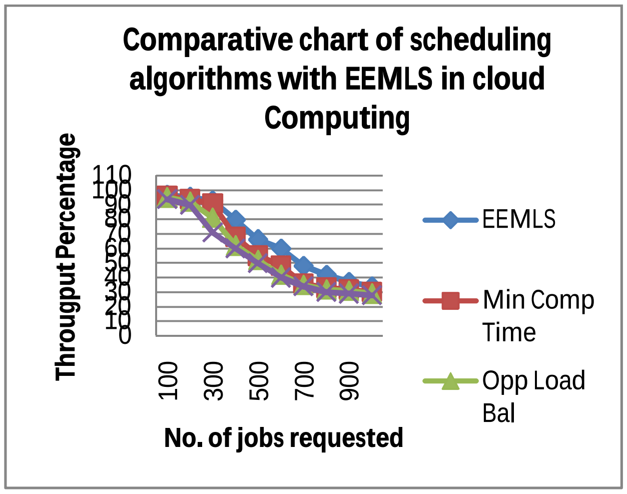


Figure 8: Throughput percentage graph to show comparison between EEMLS with illustrated scheduling algorithms.

As depicted from the above figures, the throughput parameter is inversely proportional to the number of jobs and it decreases as the number of jobs or we can say the scalability factor is increased. It is clearly visible from the graph that our work outperforms the other standard algorithms when this throughput parameter is taken into consideration.

Hence from the above study, it is very much conclusive that other standard illustrated algorithms are outperformed by our EEMLS algorithm. The imperative reason is that our work uses the effective load balancing technique with minimum completion time scheduling approach to allocate the jobs to the servers optimally; that further enhances the energy saving in the workflow of the operations and task scheduling.

6. CONCLUSION AND FUTURE SCOPE

Cloud computing platform is abundantly used nowadays for various online based application processing and services. However the biggest consideration during the routine tasks execution of the process workflow in cloud environment is the dissipation of energy. Our this research focuses on the mechanism to reduce and check the energy loss for enhanced performance and better output. Hence a energy aware multi layered job scheduling approach is employed here for optimal resource utilization on the cloud. The scheduler and the load balancer with help of its EEMLS algorithm works efficiently to manage the multiple job queues in sync with the cloud server for task provisioning. Our work has been evaluated using different performance indicators viz energy, network (processor) utilization and response time etc. It is very much conclusive that other standard illustrated algorithms are outperformed by our EEMLS algorithm. The imperative reason is that our work uses the effective load balancing technique with minimum completion time scheduling approach to allocate the jobs to the servers optimally; that further enhances the energy saving in the workflow of the operations and task scheduling. Our future work intends to check the energy loss in task scheduling procedures to restrict the carbon foot print for protecting our environment.

REFERENCES

1. Neha, Dr. Sanjay and Swati. **A Comparative Analysis of Min-Min and Max-Min Algorithms based on the Makespan parameter** in *International Journal of Advanced Research in Computer Science*, Vol.8, No.3, March 2017, ISSN No. 0976-5697
2. Omar, Mohamed Abu and Reshad **Improved Max-Min Algorithm in Cloud Computing** in *International Journal of Computer Applications*, july 2012, DOI: 10.5120/7823-1009
3. Cao, Fei, Michelle M. Zhu, and Chase Q. Wu. **Energy-efficient resource management for scientific workflows in clouds** In 2014 *IEEE World Congress on Services*, pp. 402-409.
4. Shridhar Domanal, Ram Mohana Reddy Guddeti and Rajkumar Buyya **A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment**, *IEEE Transactions On Services Computing*, Vol. X, No. X, July 2016.
5. Neha Thakkar, Dr. Rajender Nath **Performance Analysis of Min-Min, Max-Min and Artificial Bee Colony Load Balancing Algorithms in Cloud Computing** in *IJIACS*, ISSN 2347 – 8616, Volume 7, Issue 4, April 2018
6. Vinod Kumar Saroha and Dr. Sanjeev Rana, **Implementing a multi layer job scheduling approach with effective load balancing and energy saving over a cloud** *ICIME 2018*, September 22–24, 2018, Salford, United Kingdom, ACM ISBN 978-1-4503-6489-8/18/09…$15.00
7. C.T. Ying and Y. Jiong, **Energy-aware Genetic Algorithms for Task Scheduling in Cloud Computing**, 2012 *Seventh China Grid Annual Conference*, (2012) September 20-23, Beijing, pp. 43-48, IEEE.
8. Isam Azawi Mohialdeen **Comparative Study of Scheduling Algorithms in Cloud Computing Environment**, *Journal of Computer Science*, 9 (2): 252-263, 2013, ISSN 1549-3636
9. Aparnaa, S. K., and K. Kousalya. **An Enhanced Adaptive Scoring Job Scheduling algorithm for minimizing job failure in heterogeneous grid network** in *IEEE*, *ICRTIT*, 2014, pp.1-6, 2014
10. Daljinder Singh, Madeep Devgan **Multilayer Hybrid Energy Efficient Approach in Green Cloud Computing** in *International Journal of Computer Applications* ISSN:0975 – 8887, Vol.6, August 2016.
11. Dwivedi, Sanjay K., and Ritesh Gupta. **A simulator based performance analysis of multilevel feedback queue scheduling** In *IEEE Computer and Communication Technology (ICCCT)*, 2014 Intl. Conference pp. 341-346
12. Kaur, Harmeet, and Rama Krishna Challa. **A new hybrid virtual machine scheduling scheme for public cloud** in *IEEE Advanced Computing & Communication Technologies*, pp. 495-500, 2015.
13. Gupta, P.K. and N. Rakesh, 2010. **Different job scheduling methodologies for web application and web server in a cloud computing environment**, *IEEE Xplore Press*, Goa, pp: 569-572. DOI: 10.1109/ICETET.2010
14. Yang, B., X. Xu, F. Tan and D.H. Park. **An utility based job scheduling algorithm for cloud computing considering reliability factor** *ICCSC,IEEE Xplore Press*, Hong Kong, pp:95-102. DOI: 10.1109/ CSC.2011.6138559
15. Babbar, Davender, and Phillip Krueger. **A performance comparison of processor allocation and job scheduling algorithms for mesh-connected multiprocessors**." In *Sixth IEEE Symposium* on, pp. 46-53, 1994
16. Antipas, Lawrence, etal. **Comparative Study of K-Power Means, Ant Colony Optimization, Kernel Power Density-based Estimation, and Gaussian Mixture Model for Wireless Propagation Multipath Clustering** in *IJETER* Vol. 8. No. 7, July 2020, ISSN 2347 - 3983 https://doi.org/10.30534/ijeter/2020/164872020
17. Bansal, Sunita, and Chittaranjan Hota. **Priority-based Job Scheduling in Distributed Systems** in *International Conference on Information Systems, Technology and Management*, pp.110-118. Springer Berlin Heidelberg, 2009
18. Aparnaa, S. K., and K. Kousalya. **An Enhanced Adaptive Scoring Job Scheduling algorithm for minimizing job failure in heterogeneous grid network**. In *IEEE, Recent Trends in Information Technology (ICRTIT)*, 2014
https://doi.org/10.1109/ICRTIT.2014.6996161
19. D. Srinivasa Rao, Ch.Rajasekhar, GBSR Naidu **An Energy-Efficient priority Scheduling Technique for Disaster Response Cellular Networks** in *IJETER,* Volume 8, No. 7, July 2020, ISSN 2347 – 3983
https://doi.org/10.30534/ijeter/2020/136872020