Prioritized Energy Efficient Resource Scheduling in Cloud Computing

Jayanta Datta[1], *Subhamita* Mukherjee[2], and *Indrajit* Pan[1*]

[1] RCC Institute of Information Technology, Kolkata, 700015, West Bengal, India
[2] Techno Main Salt Lake, Kolkata, 700091, West Bengal, India

Abstract. Resource scheduling in cloud computing is one of the impactful area of research. Cloud service providers maintain its efficacy through proper resource management schemes. Users experience seamless cloud services when cloud service provider manages its resources efficiently. Another aspect in resource management is energy efficient schemes. Energy efficiency largely depends on employment of minimum number of resource servers. This article discusses energy efficient resource scheduling mechanism on multi-layer prioritized resource requests. Resource requests are segregated into three categories and those are scheduled on resource servers based on their fast availability. The work focuses on scheduling resource requests on previously active servers as much as possible before activating any idle servers. Proposed scheduling scheme has been tested on a standard benchmark datasets and performance of the proposed method establishes its efficacy.[*]

1 Introduction

Cloud computing is the delivery of on-demand computing services from applications to storage and processing power. Services are rendered typically over the internet and on a pay-as-you-go basis [1]. The popular models of cloud computing are;

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Resource Allocation involves multiple decision making like what, how many, where and when to make the resource available to the user. Typically, users decide the type and quantum of the resources before placing the request. Cloud service providers (CSP) then schedule the requested resource on a particular server site on their network [2]. Resource request scheduling is the process of allocating resources from their network pool to different users based on their requests and placement time.

Energy efficient design is one of the focus areas in any engineering design. Energy efficient design in cloud computing is focused on energy saving schemes. Energy saving schemes is

[*] Corresponding author : indrajit.pan@rcciit.org.in

mostly focused on switching on and off of CSP infrastructure involving hardware servers and network servers.
Proposed scheme in this article focuses on resource requests with multi-layer priorities. Basically request priorities are divided into three categories, based on which requests are placed in the ready queue. Priority factors are considered above the arrival time of requests and accordingly those are placed in the ready queue. Requests placed in the front of the queue are considered first for scheduling. This is termed as prioritized first scheduling mechanism. Another focus of this model is energy conservation. Energy conservation is done through fast reallocation and scheduling mechanism. The process of prioritized first scheduling triggers the fast allocation and scheduling mechanism to envisage prioritized energy efficient resource scheduling mechanism proposed in this article.
Rest of the article is organized as in section 2 focuses on related work; section 3 elaborates basics of cloud computing; the proposed model is depicted in section 4 followed by experimental results and discussion in section 5. Finally the conclusion is given in section 6.

2 Related Work

Sushil et al. [1] discussed that CPU scheduling introduces some issues such as starvation, large average waiting time, turnaround time and its practical implementation. Authors have used both average and variable time quantum to remove these problems. Some processes are served with average time quantum and others with variable time quantum. This approach not only provides the minimum average waiting time and turnaround time but also tries to prevent the starvation problem.
The article in [2] offers a generic scheduling algorithm that reduces the waiting time of the overall system. However the tasks enter in the cloud environment and the users have to wait until the resources are available that leads to more queue length and increased waiting time. This article introduces a task scheduling algorithm based on a generic algorithm using a queuing model to minimize the waiting time and queue length of the system.
Zhao et al. [3] proposed cloud workflow scheduling by bringing cloud and workflow together in cloud workflow management system. Security improvement in workflow execution has been introduced by Wang et al. [4] through an attack-defense game model-based scientific work-flow scheduling method.
A resource provisioning method is discussed by Rodriguez et al. [5] where workflow execution cost has been minimized under some pre-defined deadline constraint. Li et al. [6] introduced a cost and energy-aware scheduling (CEAS) algorithm for cloud scientific workflows. Tong et al. [7] proposed a deep Q-learning-based workflow scheduling algorithm in the cloud for solving the problem of handling directed acyclic graph (DAG). Zhang et al. [8] introduced an efficient priority and relative distance (EPRD) algorithm to minimize the finish time with the deadline constraint for workflow scheduling. The article in [9] introduced a Pareto-based list multi-objective workflow scheduling algorithm. Zhu et al. [10] proposed an evolutionary algorithm-based multi-objective scheduling method.
The article in [11] proposed a hybrid approach for energy aware scheduling of deadline constrained workflows (HAED) using intelligent water drops algorithm and genetic algorithm to implement multi-objective workflow scheduling. However, all the above algorithms neglect the FT problem that has a direct impact on the quality of service (QoS) of the workflow execution.
Pandey et al. [12] proposed a task replication method and an architecture-based solution that relies on task categorization and authorized access to the categories of tasks. Authors proposed for different levels of trust to improve the robustness of mobile device clouds.

3 Energy efficient cloud computing

Energy efficiency of ICT modules deals with several factors. Some of the impactful factors among them are [13],

a) Hardware pattern/ configuration
b) Power consumption by physical machines
c) Volume of carbon emission which is directly related with power on time
d) Power off and cooling time, this cooling time increases proportionally with power on time

There is a need to focus on energy efficient design of cloud computing infrastructure with the on growing popularity of cloud computing. There are many factors associated with energy rating of any cloud computing infrastructure which involves

a) Physical layers in the physical system
b) Number of physical sites
c) Power on time of physical sites
d) Cooling time of those physical sites
e) Network and communication protocol and hardware

One aspect of energy efficient design of cloud computing infrastructure is hardware optimization. On other side software system can also contribute in energy efficient design. Balancing in software system should maintain focus on certain factors like

a) Performance of the system
b) Compliance with cloud service level agreement (Cloud SLA)
c) Quality of services (QoS)

Significant effectiveness in energy efficient design can be obtained through energy aware scheduling of resource services from cloud service providers (CSP). It requires multiplexing and de-multiplexing of resource service loads across various CSPs. Sometimes also there is a need to balance between strict energy efficient scheduling and quality of service aware scheduling.

This article proposes an energy aware scheduling mechanism so that the usage of number of physical machines can be reduced which in turn will impact the limited power on time for servers and limited cooling time.

4 Proposed Method

4.1 Proposed Model

- Minimize the number of servers used by migrating load of one server to another server then saves the energy as only required servers are active and remaining servers are not active.
- Maximize the resource utilization by giving services to new clients from the previously deallocated servers instead of starting the new server for new requests, thereby saving the energy.
- Schedule the client requests into multilayer jobs categories - critical, moderate and normal. The critical tasks are scheduled on fast servers so services are quick and less critical tasks are scheduled on moderate servers.

4.2 Scheduling Objective

A variety of factors must be considered when developing a scheduling discipline, such as what type of systems and what are user's needs. Following are the project goals that are to be accomplished:
Maximizing throughput: Distribute network traffic and incoming service requests across multiple servers on a selective basis. This ensures no single server bears too much load. By spreading the work evenly, responsiveness and throughput increase.
Enforcement of priorities: The scheduling mechanism should favour the higher-priority processes.

4.3 Proposed Algorithm

4.3.1 Assumptions

1. Requests with high priority are allocated first with the top priority; they are placed in front of ready queue before scheduling.
2. Requests with moderate priority are considered after high priority tasks.
3. Requests with low priority are allocated later and they are placed in the rear of the ready queue.
4. Each request comes with a deadline and their priorities.
5. Request with shorter deadlines are processed first to reduce number of failed/ not served requests.
6. The ready queue concept is implement to track the resource requests under consideration for scheduling.
7. Server activation is another aspect of consideration to maintain energy efficiency. Initially 25% of the whole servers are deployed to schedule live requests in ready queue. This number is increased or a new server is added only when there is an absolute need for serving resource request. Failed requests are usually avoided to maintain quality of services. A trade-off is done between minimum server usage and sustenance in quality of services.

4.3.2 Proposed Algorithm

Step 1. Accept resource requests with priority of the request and their deadline
Step 2. Resource requests are arranged in the ready queue first with their deadlines. Requests having similar deadlines are rearranged in the ready queue following descending order of their priorities.
Step 3. Requests are allocated to the server with a focus on their deadline parameter in the best-fit allocation scheme.
Step 4. Server availability is checked for scheduling within deadline. If a resource request is not schedulable upon available server then next resource request is considered for allocating by placing the current resource request in another queue with halted tag.
Step 5. Once all available and schedulable requests from ready queue is scheduled on available servers then halted queue is considered for scheduling. Number of server is minimally increased through power on so that all possible resource requests are scheduled.

Step 6. Once any active server releases the allocated task then request migration takes place to gradually reduce the count of active servers.

Step 7. Overall runtime of each server is maintained and monitored along with average runtime of active servers. Minimization of average and cumulative run time of servers provide a measure for energy consumption through powering on and cooling subsequently.

Proposed algorithm is further explained through two flowcharts given below. Fig. 1 (a) describes the process of prioritized scheduling and fig. 1 (b) provides a schematic representation of dynamic server migration technique during the resource serving process.

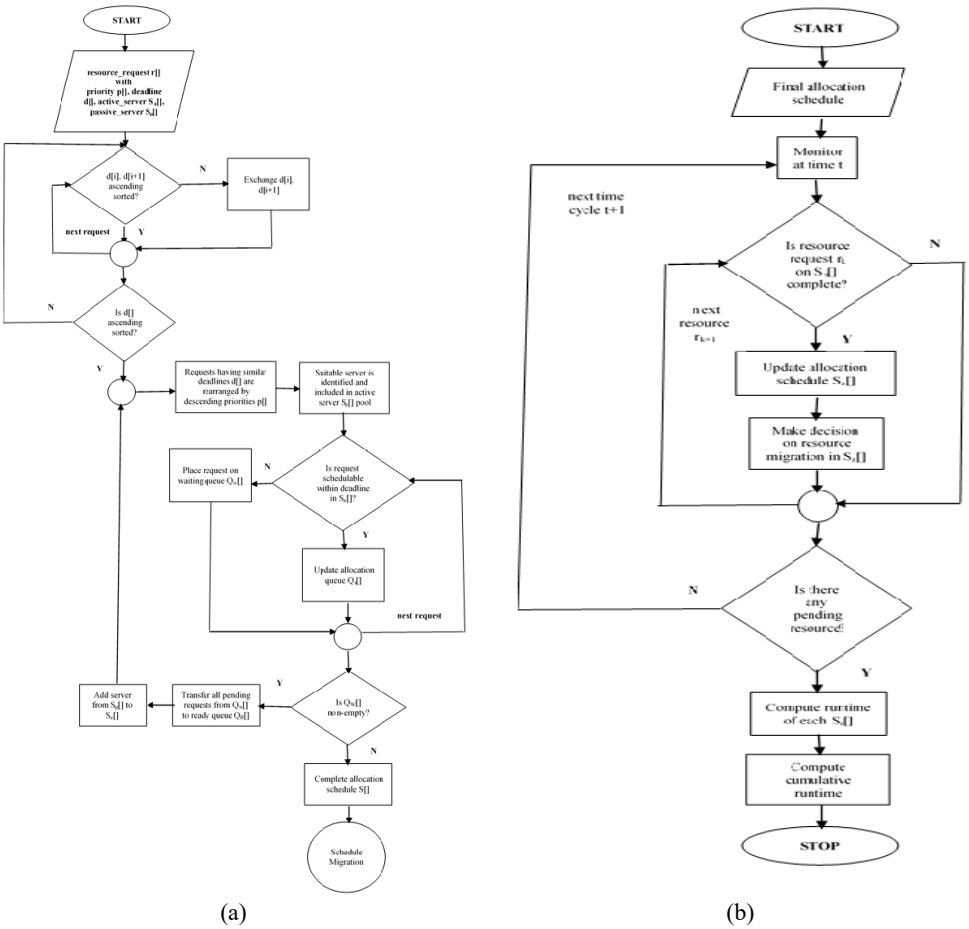

Fig. 1. Flowchart of (a) Prioritized Resource Scheduling (b) Dynamic Server Migration

5 Experimental Results

Proposed method has been experimentally tested on CloudSim framework. CloudSim is more generalized and extensible simulation framework. Configuration of servers in simulated Cloud Service Providers comprises configuration of IBM X3850 [15] having thirty central processing units each having six cores, 32 GB DDR3 1066 Hz memory having power parameters as described in table 1.

Table 1. Configuration of host servers

Host	Power Rating	Peak Power	Standby Power	Start Duration	Idle Duration
X3850	1975W x 2	2765W x 2	98.75W x 2	2 min	20 min

Load of resource request is described in table 2.

Table 2. Resource request description

Resource Request	VMs #	Deadline	Priority	Median (%)	St. Dev (%)
R1	1052	22	H	6	17.09
R2	898	13	L	5	16.83
R3	1061	37	L	4	15.57
R4	1516	17	H	5	12.78
R5	1078	27	M	6	14.14
R6	1463	11	M	6	16.55
R7	1358	31	H	6	15.09
R8	1233	23	M	6	15.07
R9	1054	21	L	6	15.15
R10	1033	18	L	4	15.21

Table 2 represents ten different setups. Each setup contains requirement of multiple resource requests given in terms of VM#. Deadline column contains the maximum deadline among all resource requests under a particular set. Priority is given in terms of H for high, M for medium and L for low. These priority values are converted in numerical numbers 1 for low, 3 for medium and 5 for high. Proposed method was deployed on above setup and experimental analysis has recorded following parameters with respect of each resource request.

1. Percentage of successful completion
2. Average power on time

Table3. Experimental findings

Resource Request	Proposed Method		[15]	
	Success %	Avg. Power on time (mins)	Success %	Avg. Power on time (mins)
R1	97	3.306	90	5.212
R2	94	1.667	92	3.958
R3	98	5.608	92	7.368
R4	98	3.681	90	8.569
R5	97	4.158	92	5.968
R6	96	2.299	92	4.522
R7	97	6.014	92	10.234
R8	94	4.051	90	9.219
R9	94	4.051	92	8.526
R10	98	2.656	88	5.237

Overall observation is recorded in table 3. Table 3 also records another parameter in the last column which is the performance of algorithm proposed in [15]. Hence a comparative measure is evident from table 3 between proposed method and the method discussed in [15]. Percentage of successful completion is also a measure of quality of services of cloud service provider. Average power on time is measured in terms of minutes. It is the average of service time of all cores to access a particular resource request. Experimental data recorded in table 3 is graphically represented in fig. 2.

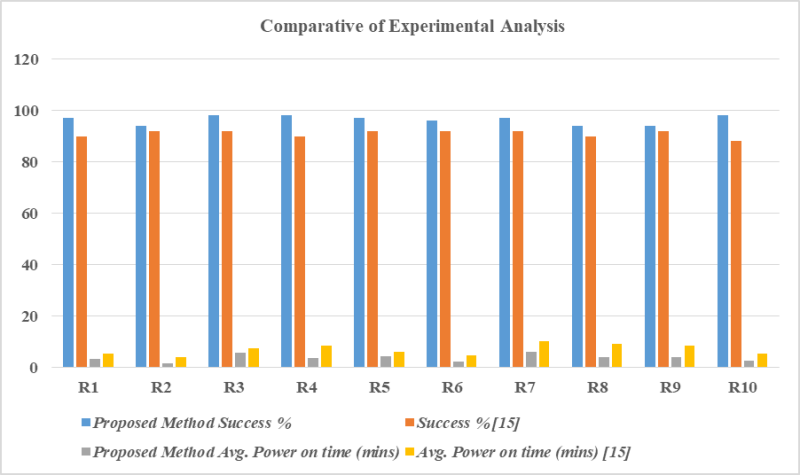

Fig. 2. Comparative Representation of Experimental Findings in Table 3.

6 Conclusion

This article discusses a prioritized energy efficient scheduling algorithm. Proposed method considers resource requests with deadline and priorities. First deadlines are considered for scheduling, requests with shorter deadlines are scheduled first. If there are more than one resource request with similar deadlines then priorities are considered. Requests with higher priority are scheduled first.

Energy conservation is another aspect for this scheduling. Main focus was on minimal usage of resource servers. Initially it activated 25% of the total available servers. Maximum number of resource requests from ready queue is attempted to schedule on these active servers. Any request which was not possible to schedule on first attempt was placed in a halted queue. Later all jobs placed in the halted queue are scheduled by minimally allowing additional servers. Another aspect is server migration. Post scheduling with the time servers release some completed or served requests. Then load is balanced by rearranging resource requests on available servers so that additional servers can be released and sent to cool down mode.

Proposed method was deployed on benchmark data sets and found to be mostly effective in terms of successful scheduling, energy conservation and quality of services. Successful scheduling is measured by means of counting more number of successfully completed requests. Energy conservation is measured in terms of minimal total power on time through a cumulative measure by considering all servers which was powered on. Quality of service is a measure which is directly related with number of successfully served requests.

Experimental results were compared with a recently published work and found to be effective over them.

References

1. S. Saroj, A. Sharma, novel CPU scheduling with variable time quantum based on mean difference of brust time, in, 2016 International Conference on Computing, Communication and Automation, ICCCA, 1342-1347, (2016)
2. S. Saha, S. Pal, A novel scheduling algorithm for cloud computing environment, in international conference on computational intelligence in data mining (CIDM), 387-398, (2015)
3. Y. Zhao, X. Fei, Opportunities and challenges in running scientific workflows on the cloud, in 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 455–462, (2011)
4. Y. Wang, CLOSURE: a cloud scientific workflow scheduling algorithm based on attack-defense game model, Future Gener. Comput. Syst. 111, 460–474, (2020)
5. M. Rodriguez, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, IEEE Trans. Cloud Comput. 2 (2), 222–235, (2014)
6. Z. Li, Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds, IEEE Trans. Serv. Comput. 11 (4), 713–726, (2018)
7. Z. Tong, A scheduling scheme in the cloud computing environment using deep Q-learning, Inf. Sci. 512, 1170–1191, (2020)
8. L. Zhang, Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments, Inf. Sci. 531, 31–46, (2020)
9. J. Durillo, R. Prodan, MOHEFT: a multi-objective list-based method for workflow scheduling, in International Conference on Cloud Computing Technology and Science Proceedings, 185-192 (2012)
10. Z. Zhu, Evolutionary multi-objective workflow scheduling in cloud, IEEE Trans. Parallel Distrib. Syst. 27 (5), 1344–1357, (2016)
11. M. Kalra, Multi-objective energy aware scheduling of deadline constrained workflows in clouds using hybrid approach, Wireless Pers. Commun. 116 (3), 1743–1764, (2021)
12. P. Pandey, Robust orchestration of concurrent application workflows in mobile device clouds, J. Parallel Distrib. Comput. 120, 101–114, (2018)
13. A. Berl, E. Gelenbe, D. Girolamo, G. Giuliani, D. Meer, Q. Dang, Pentikousis, K.: Energy-efficient cloud computing, The Comp. J., vol. 53, no. 7, 1045 – 1051, (2010)
14. Amazon EC2 Instance Types, http://aws.amazon.com/ec2/instancetypes/
15. J. Cao, Y. Wu, M. Li, Energy efficient allocation of virtual machines in cloud computing environment based on demand forecast, R. Li, J. Cao and J. Bourgeois (Eds) GPC 2012, LNCS 7296, 137 – 151, (2012)