Efficient Resource Utilization Algorithm (ERUA) for Service Request Scheduling in Cloud

Ramkumar N, Nivethitha S
rkjeeth@gmail.com, nivethithasomu@gmail.com
School of Computing, SASTRA University, Thanjavur, Tamil Nadu.

Abstract **- Cloud computing provides us with the massive pool of resources in terms of pay-as-you-use policy. Cloud delivers these resources on demand through the use of network resources under different load conditions. As the users will be charged based on their usage the effective utilization of resources poses a major challenge. To accomplish this, a service request scheduling algorithm which reduces the waiting time of the task in the scheduler and maximizes the Quality of Service (QoS) is needed. Our proposed algorithm named Effective Resource Utilization Algorithm (ERUA) is based on 3-tier cloud architecture (Consumer, Service Provider and the Resource Provider) which benefits both the user (QoS) and the service provider (Cost) through effective schedule reallocation based on utilization ratio leading to better resource utilization. Performance analysis made with the existing scheduling techniques shows that our algorithm gives out a more optimized schedule and enhances the efficiency rate.**

Keywords**: Cloud; Service Request; Service Provider; Consumer; Scheduler Units**

I INTRODUCTION

Cloud computing an emerging and an enabling technology which made us to think beyond what is possible. Realizing the services and amenities provided by the cloud many organizations decided to jump into cloud in order to reduce the infrastructure cost and energy consumption. Cloud makes them to move their business with different range and style of services. It had the changed the traditional way of using the resource infrastructure. Service request scheduling is the most crucial area with respect to the profit of the service provider and the QoS of the user.

Cloud computing services are offered based on 3-tier architecture. The entire architecture of a cloud with respect to service request scheduling comprises of the resource provider, the service providers and the consumers. In order to service the request given by the consumer, the service provider needs either to procure new hardware resources or to rent it from resource provider. However, getting resource on rental basis incurs less cost than buying a new one.

The service provider hires resources from the resource provider and creates Virtual Machine (VM) instances dynamically to serve the consumers. Resource provider takes on the responsibility of dispatching the VM's to the physical server. Charges for the running instance are based on the flat rate (/time unit). Users submit their request for processing an application consists of one or more services. These services along with the time and cost parameters are sent to the service provider. In general the actual processing time of a request is much longer than its estimated time as there incurs some delay at the service provider site. As the cloud is a form of "pay-as-you-use" utility, the service provider needs to reduce the response time and delay. Over here service request scheduling becomes an essential element to reduce maximize the profit of service provider and to improve the QoS offered to the user.

Earlier research contributions towards service request scheduling algorithms were on SERver CONsolidation [1], optimized service scheduling algorithm [2], scheduling policy based on priority and admission control [3], integration of VM for sorting tasks based on the profit [4], multiple pheromone algorithm [5], gang scheduling on VM [6], utility model to balance the profit between the user and the service provider [7], dynamic service request resource allocation through gi-First In First Out (FIFO) [8], Service Level Agreement (SLA) creation, management and usage in utility computing [9], scheduling dynamic user request to maximize the profit of the service provider [10], Ant Colony Optimization (ACO) [11], Particle Swam Optimization (PSO) [12], dynamic distribution of user request between the application services in a decentralized way [13], scheduling algorithm based on genetic algorithm to reduce the waiting time [14], task consolidation heuristics with respect to idle and active energy consumption [15], pricing model based on processor – sharing through max_profit and max_utility [16], optimized service request – resource mapping using genetic algorithm [17], dynamic priority scheduling algorithm [18].

Our algorithm ERUA for service request scheduling schedules the task units based on the utilization ratio of the queue. It always ensures that the utilization ratio always falls within 1 leading to better resource utilization

and enhancing the efficiency through enabling the task units to finish up its execution within their deadline. With our sample set of data, ERUA proves to be more optimal than the existing algorithms for service request scheduling.

The remainder of the paper is sectioned as follows: Section 2 enlightens the concept of service request scheduling and our proposed algorithm, Section 3 discusses about the results and interpretation and Section 4 concludes this paper.

II PROPOSED METHODOLOGY

2.1 Scheduling Process

The process of scheduling can be viewed as service request scheduling (service provider and the Consumer) and resource scheduling (service provider and resource provider). The process of service request scheduling occurs as:

a) Users submit their request to the service provider.
b) Service provider executes the request.
c) Process the request in the service request architecture.
d) Dynamic VM generation and dispatch at the resource provider site.

2.2 System Architecture

The major components in the service request scheduling are (Figure 1):

i. **Classifier**: Receives user request, process and classifies into smaller task units. These task units can be scheduled directly onto the scheduler but before that it needs to get assigned with random priorities. Priority can either be based on system state or the task characteristics. Once each task gets its unique priority these task units can be sent to the scheduler component to be scheduled.

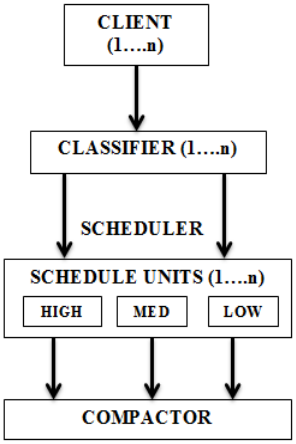

Figure 1. System Architecture of ERUA.

ii. **Scheduler**: Each scheduler contains several schedule units, each having its own priority based on the system design and the real situation. Scheduler pushes up the task units into appropriate schedule units based on the idleness and the saturation of each and every schedule unit. Scheduler units execute the task units based on the algorithm. The task unit with the lower deadline will be scheduled first to optimize the result.

iii. **Compactor**: Summarizes the completed task units during each cycle and sends it to the resource provider.

2.3 The Process of Service Request Scheduling

Users submit their request for executing their application which consists of one or more services to the SP. Now the SP has to perform the service request scheduling process with these requests and has to operate on a massive set of data. So the SP requires a scheduler to efficiently schedule these request maximizing the QoS to the user and the profit on the SP site. The process of service scheduling starts here. Each request will be spliced into task units and are assigned with some random priority in the classifier. Classifier pushes these task units into an appropriate scheduler units based on the state of the scheduler units. Scheduler units execute the task unit based on some algorithm. Our algorithm considers utilization ratio as the deciding factor for priority reassignment. Let us consider an example for priority reassignment (Figure 2).

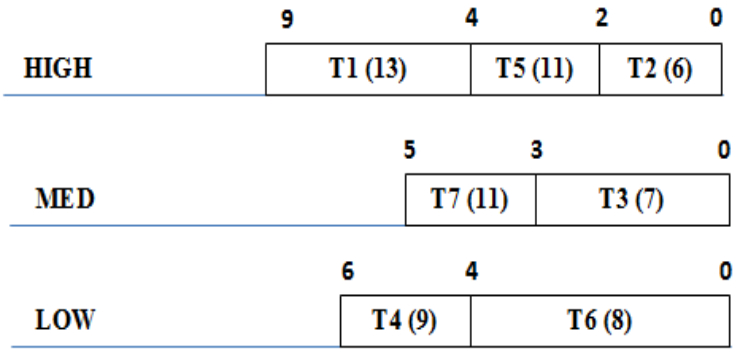

Figure 2. Example of scheduling task in prioritized queue.

Task units T2 (6), T3 (14) and T6 (8) having the lowest deadline will be assigned for execution at first in high, medium and low priority scheduler units respectively. Now after a cycle the remaining task units will be high queue - T5 (11) and T1 (13), medium queue – T7 (11) and low queue – T4 (9) and T6 (8). If there is a task T8 (19) with the execution time of 7 ms in the higher priority queue that needs to be executed after T1 (13), it can be scheduled to execution in the medium priority scheduler unit after T7 (11) as it frees up after 5 ms. This can be done by analysing the remaining jobs and the completion time of the current jobs scheduled in the queue, thus minimizing the delay of 1 ms, while enhancing the processor utilization. Now, T8 (14) completes its execution by 12 ms within its deadline. Whenever the queue frees up irrespective of the priority class, the tasks can be scheduled onto any one of them based on the state of the scheduler units. Priority reassignment based on deadline gives us a better way of maximizing the throughput and the performance of the system through effective resource utilization.

III RESULTS AND DISCUSSIONS

User submits their request to the service provider and it enters the scheduling architecture through the classifier. The classifier component split up the user request into several independent task units. Let us consider the following table which consists of several task units of a single request (Table 1). Now, the classifier assigns some initial priority (least deadline) to each task units and schedules them on to the schedule units. Here, we will be having three scheduler queues with high, medium and low priority respectively and this depends upon the design and the current load of the system.

The initial tasks scheduled to execution will be T2 (6) and T6 (11) in high priority queue, T3 (14) and T7 (17) in medium priority queue and T5 (8) and T10 (9) in low priority queue are shown in Figure 3. T8 (13) with the higher priority will be scheduled next to T6 (11) in the high priority queue (Figure 4). Always be sure about

$$\textbf{Utilization Ratio}_i\textbf{ (Queue) = (Execution Time }_i\textbf{ / Deadline }_i\textbf{)} \leq 1 \qquad \textbf{(i)}$$

TASK	EXECUTION TIME	DEAD LINE	PRIORITY
T1	8	50	HIGH
T2	2	6	HIGH
T3	3	14	MEDIUM
T4	4	12	LOW
T5	4	8	LOW
T6	2	11	HIGH
T7	2	17	MEDIUM
T8	5	13	HIGH
T9	3	45	HIGH
T10	2	9	LOW
T11	3	13	LOW

Table 1. Task Units Schedule.

The task units T2, T6, T8, T3, T7, T5 and T10 were scheduled on to execution within their deadline with the utilization ratio of 0.89 (T2, T6 and T8) on high priority queue, 0.33 (T3 and T7) on medium priority queue and 0.72 (T5 and T10) on low priority queue. To keep the queue busy, always ensure that the queue utilization

should be within 1. Write down the remaining task that needs to be scheduled (Table 2).

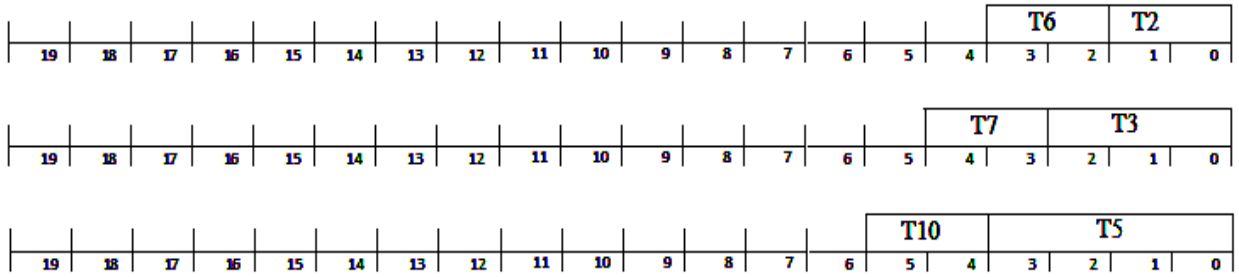

Figure 3. Initial Schedule Based on the Least Deadline.

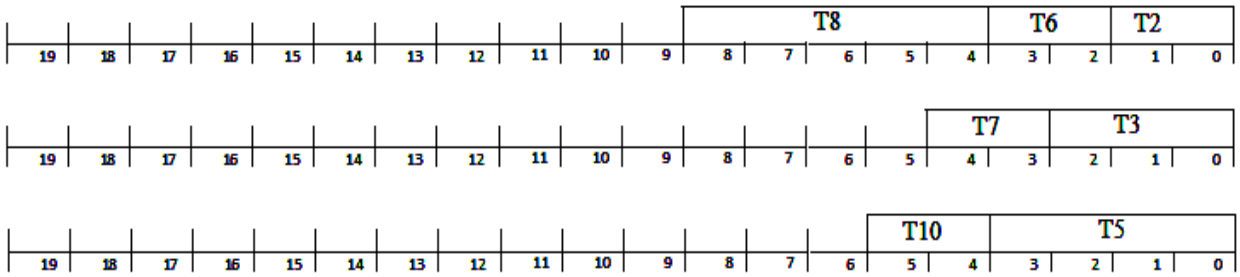

Figure 4. T8 with high priority scheduled on to high priority queue.

PRIORITY	TASK	EXECUTION TIME	DEAD LINE
HIGH PRIORITY	T1	8	50
	T9	3	45
LOW PRIORITY	T4	4	12
	T11	3	13

Table 2. Consolidated task details with priority reassignment based on deadline.

The task T4 (12) with the low priority will be scheduled on the medium priority queue after T7 (17) as T7 (17) completes 1ms before T10 (9) in the low priority queue (Figure 5). The task T11 (13) with the low priority will be scheduled on the low priority queue after T10 (9) (Figure 6). The task T9 (45) with the high priority will be scheduled on the high priority queue after T8 (13) as T9 (45) will have the least deadline than T1 (50) (Figure 7). The task T1 (50) with the high priority will be scheduled on the medium priority queue after T4 (12) as T4 (12) completes 4ms before T9 (45) (Figure 8).

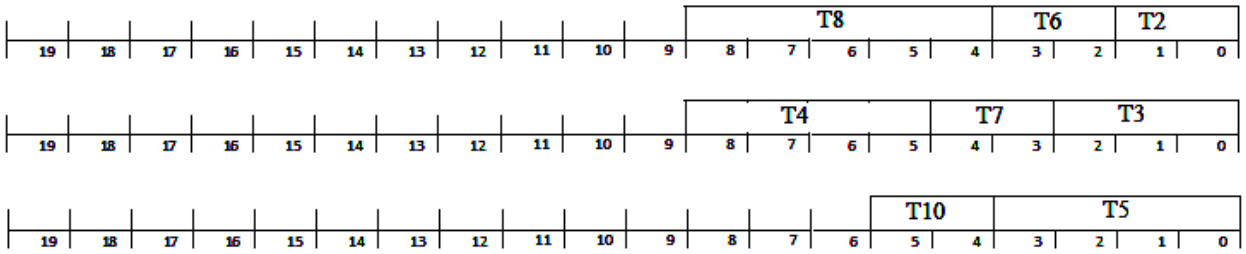

Figure 5. T4 with the lower priority scheduled on to idle medium priority queue (idle).

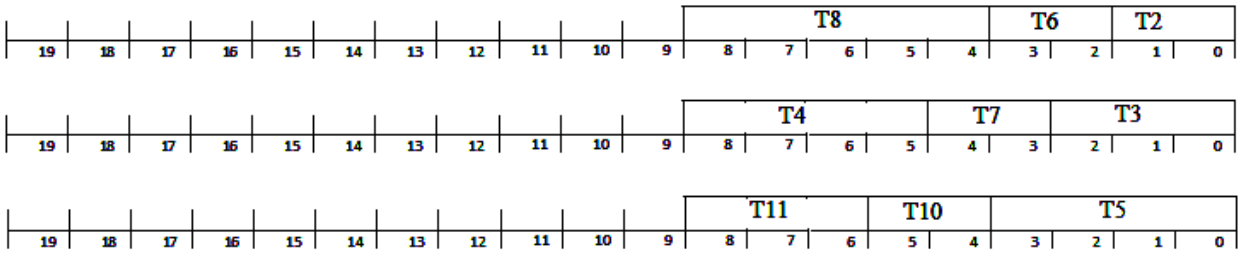

Figure 6. T11 with the lower priority scheduled on to low priority queue.

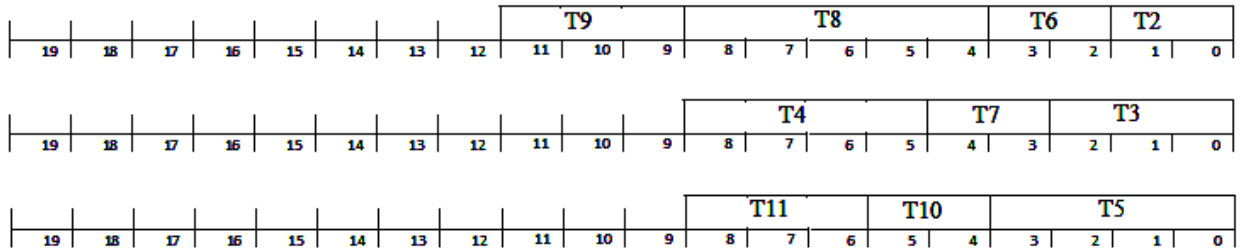

Figure 7. T9 with the higher priority scheduled on to the high priority queue.

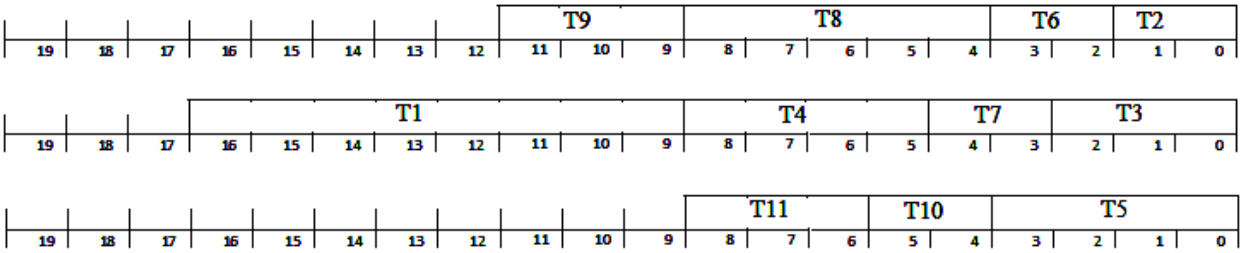

Figure 8. T1 with the higher priority scheduled on to the medium priority queue (idle).

When **D**ynamic **P**riority **S**cheduling **A**lgorithm (**DPSA**) is used to schedule the same set of tasks the utilization ratio ($U_i = e_i / d_i$) **1.12** (T1, T2, T6, T8 and T9) on high priority queue, 0.33 (T3 and T7) on medium priority queue and **1.28** (T5 and T10) on low priority queue. DPSA violates the condition for effective utilization by exceeding 1 affecting the QoS by prohibiting most of the tasks to meet their deadline. ERUA schedules task in such a way that the utilization ratio (U_i) of high priority queue (0.95), medium priority queue (0.82) and low priority queue (0.95).

VI PERFORMANCE ANALYSIS

The performance analysis made by comparing **ERUA** with **F**irst **C**ome **F**irst **S**erve (**FCFS**) (no priority), **S**tatic **P**riority **S**cheduling **A**lgorithm (**SPSA**) (fixed priority), **E**arliest **D**eadline **F**irst (**EDF**) and **DPSA** is shown (Figure 9). The efficiency of our algorithm can be measured using

$$\text{Efficiency}_i = \frac{\text{Number of Tasks Scheduled}}{\text{Total Number of Tasks}} \times 100 \quad \textbf{(ii)}$$

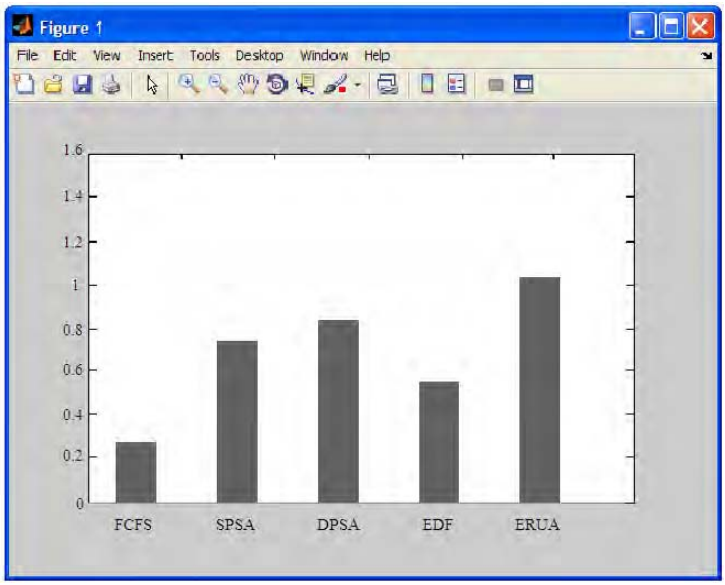

Figure 9. The Efficiency Comparison of Five Algorithms.

If the same set of tasks is to be scheduled using **FCFS**, tasks (**T2, T4, T5, T6, T7, T8, T10** and **T11**) miss its deadline. For **SPSA**, tasks (**T6** and **T8** – **High** priority queue and **T10** – **Low** priority queue) miss its deadline. For **EDF**, tasks (**T4, T8, T11, T3** and **T7**) miss its deadline. The efficiency of **FCFS** (**0.27**), **SPSA** (**0.72**), **EDF** (**0.54**), **DPSA** (**0.81**) and **ERUA** (**0.96**) are plotted in the graph to illustrate the optimality of ERUA. ERUA proves to be an optimal service request scheduling algorithm through effective resource utilization.

As per this schedule,

ALGORITHM	EFFICIENCY (%)
FCFS	**27%**
SPSA	**72%**
EDF	**54%**
DPSA	**82%**
ERUA	**98%**

Table 3. Efficiency (%)

V CONCLUSION

Users focus on the QoS whereas the service providers rely on maximizing their profit. To satisfy both the user and the service providers we need an efficient service request scheduling algorithm in a cloud computing platform. Our algorithm satisfies the requirement of both the users and the service providers through efficient schedule and priority reassignment. It services the SLA model of the user and the cost model for the service provider through dynamic resource reuse management. Our future work investigates on evaluating the users SLA model and the service provider profit model under different load condition.

VI REFERENCES

[1] Ana Juan Ferrer, Francisco Hernández, Johan Tordsson , Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsigri, Raül Sirvent, Jordi Guitart, Rosa M. Badia, Karim Djemamee, Wolfgang Ziegler, Theo Dimitrakos, Srijith K. Nair, George Kousiouris, Kleopatra Konstanteli, Theodora Varvarigou, Benoit Hudzia, Alexander Kipp, Stefan Wesnerj, Marcelo Corrales, Nikolaus Forgó, Tabassum Sharif and Craig Sheridan, "OPTIMIS: A holistic approach to cloud service provisioning", Future Generation Computer Systems, 2012, pp 66–77.

[2] Aziz Murtazaev, Sangyoon Oh, "Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing," IETE Technical Review, 2011, pp 212-231.

[3] Dr. M. Dakshayini and Dr. H. S. Guruprasad, "An Optimal Model for Priority based Service Scheduling Policy for Cloud Computing Environment", International Journal of Computer Applications, 2011, pp 0975–8887.

[4] Geetha J, Rajeshwari S B, Dr. N Uday Bhaska and Dr. P Chenna Reddy, "An Efficient Profit-based Job Scheduling Strategy for Service Providers in Cloud Computing Systems", International Journal of Application or Innovation in Engineering & Management (IJAIEM), 2013, pp 336-338.

[5] 5 R.Gogulan, A.Kavitha and U.Karthick Kumar, "An Multiple Pheromone Algorithm for Cloud Scheduling With Various QOS Requirements", International Journal of Computer Science Issues (IJCSI), May 2012, pp 232-238.

[6] Ioannis A. Moschakis and Helen D. Karatza, "Evaluation of gang scheduling performance and cost in a cloud computing system", The Journal of Supercomputing, 2012, pp 975-992.

[7] J. Chen, C. Wang, B. Zhou, L. Sun, Y. Lee and A. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud", International Symposium on High Performance Distributed Computing, 2011, pp 229-238.

[8] Keerthana Boloor, Rada Chirkova and Yannis Viniotis, “Dynamic request allocation and scheduling for context aware applications subject to a percentile response time SLA in a distributed cloud “, IEEE International Conference on Cloud Computing Technology and Science, 2011, pp 464-472.
[9] Linlin Wu and Rajkumar Buyya , “Service Level Agreement (SLA) in Utility Computing Systems, Technical Report, 2010.
[10] Linlin Wu, Saurabh Kumar Garg and Rajkumar Buyya, “SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments”, IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2011, pp 195-204.
[11] Linan Zhu, Qingshui Li and Lingna He, “Study on Cloud Computing Resource Scheduling Strategy Based on the Ant Colony Optimization Algorithm”, International Journal of Computer Science Issues (IJCSI), 2012, pp 54-58.
[12] Noha El.Attar, Wael Awad and Fatma Omara, “Resource Provision for Services Workloads based on (RPOA)”, International Journal of Computer Science Issues (IJCSI), 2012, pp 553-560.
[13] Rajiv Ranjan, Liang Zhao, Xiaomin Wu, Anna Liu, Andres Quiroz and Manish Parashar, “Peer-to-Peer Cloud Provisioning: Service Discovery and Load-Balancing”, Cloud Computing Computer Communications and Networks, 2010, pp 195-217.
[14] Sourav Banerjee, Mainak Adhikary, Utpal Biswas, “Advanced Task Scheduling for Cloud Service Provider Using Genetic Algorithm”, IOSR Journal of Engineering (IOSRJEN), July 2012, PP 141-147.
[15] Young Choon Lee and Albert Y. Zomaya, “Energy Efficient Utilization of Resources in Cloud Computing Systems”, The Journal of Supercomputing, 2012, pp 268-280.
[16] Young Choon Lee, Chen Wang, Albert Y. Zomaya and Bing Bing Zhou, “Profit-Driven Service Request Scheduling In Clouds“, IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp 15-24.
[17] Zhipiao Liu, Shangguang Wang, Qibo Sun, Hua Zou and Fangchun Yang, “Cost-Aware Cloud Service Request Scheduling for SaaS Providers”, The Computer Journal, 2013, pp 1-1.
[18] Zhongyuan Lee, Ying Wang and Wen Zhou, “A dynamic priority scheduling algorithm on service request scheduling in cloud Computing”, International Conference on Electronic & Mechanical Engineering and Information Technology, 2011, pp 4665-4669.

Authors Profile

Ramkumar N B.E., M.Tech.,
He received his degree in Electronic and communication Engineering from Periyar Maniammai University, Thanjavur, Tamil Nadu, in 2012. He is currently pursuing his Master of Technology in Computer Science and Engineering at SASTRA University, Thanjavur, Tamil Nadu. His interests include Cloud Scheduling, Virtualization ,image processing and Wireless Sensor Networks.

Nivethitha S M.Sc., M.Tech., She received her degree in Software Engineering from Anna University, Chennai, Tamil Nadu, in 2011. She is currently pursuing her Master of Technology in Advanced Computing at SASTRA University, Thanjavur, Tamil Nadu. Her interests include Cloud Scheduling, Virtualization and Wireless Sensor Networks.