Efficient Dynamic Heuristic Task Scheduling Algorithm for Commercial Cloud Environment

M.Krishna Sudha [*1], Dr. S. Sukumaran [2]

1(Ph.D scholar, Department of Computer Science, Erode Arts and Science College, Erode, Tamilnadu, India)
2(Associate Professor, Department of Computer Science, Erode Arts and Science College, Tamilnadu, India.)
phdeasc@gmail.com

Abstract- Latest distributed computing paradigm is cloud computing and it offers tremendous opportunities to solve large-scale problems. It is critical to evaluate the performance of cloud environments as the adoption and deployment of cloud computing increases. The number of active machines are matched dynamically to resource demands inorder to increase resource utilization. However based on extensive studies, existing solutions have not fully considered the heterogeneity of both workload and resource utilization found in computing environments. Currently, effective and efficient scheduling algorithms are needed for the better use of tremendous capabilities in large scale distributed system. Many such algorithms have been designed and implemented. New scheduling algorithm based on dynamic conventional scheduling Heuristics is designed to use their cons and at the same time overcome their pros. The cloud simulator Cloudsim is used to evaluate the proposed scheduling algorithm by comparing to its performance against the basic heuristics. Variety of scenarios is taken into consideration to show that the simulation results of the new heuristic algorithm leads to significant performance gain. Maximization in resource utilization and reduction in makespan are the two basic criteria used for the performance evaluation of the proposed Algorithm. The proposed algorithm proves that it is efficient than the other heuristic algorithms by reducing the make span and increasing the resource utilization.

Keywords: Task Scheduling, Min-Min, Max-Min, NP-Problem, Heuristic Algorithm, Clustering

I. INTRODUCTION

Cloud computing is a collection of computing and communication resources located over distributed datacenters that is shared by many different users [1]. Task scheduling and other constraints which have direct effect on user consumption of resources are controlled by cloud infrastructure. In a commercial cloud environment [1], [2] individual providers are focused towards increasing their own revenue and do not care about the utility of users and other providers.

Mapping tasks to machines is an NP-Complete problem; therefore the use of heuristics is one of the suitable approaches. Existing scheduling heuristics can be divided into two classes: On-line mode (dynamic) and Batch-mode (static) heuristics [20]. In the on-line mode, a task is mapped onto a host as soon as it arrives at the scheduler. In the batch mode, tasks are not mapped on to hosts immediately and they are collected in to a set of tasks that is examined for mapping at prescheduled times called mapping events.

To make use of great capabilities of the distributed system, effective and efficient scheduling algorithms are needed. Depending on their goals, these algorithms assign tasks to the best machines which produce better quality of service [17]. In this work the problem of dynamic scheduling of task in commercial cloud environment using algorithemic design is addressed.Considering that scheduling in distributed systems is an NP-hard problem, and the challenge is to propose an efficient scheduling mechanism that yields low completion time (makespan) and high resource utilization compared to the state of the art applications. To address the scheduling problem in a commercial cloud environment a new algorithm for scheduling single tasks on a set of resources is proposed. Afterwards the proposed algorithm is tailored to dynamically schedule tasks.

II. LITERATURE REVIEW

Several researchers have pointed out the presence of performance problems with the cloud. Recently, [21] highlighted the long tail of response time in Amazon EC2 and related it to the co-scheduling of CPU bound and latency-sensitive tasks on the same hypervisor. In contrast, our focus is on designing systems to adapt to short term variability in the cloud.

Live VM migration techniques [18], [19] handle work load increase on a VM by migrating the VM to a less loaded physical server. Although migrations may involve short downtimes, they may incur additional overhead that may impact application's performance (e.g., due to iterative memory pre-copying, network bandwidth overhead, cache warm-up, etc.) Further, VM migration across data-centers in WAN environments is even more challenging (due to disk and memory state transfer, network reconfigurations, etc.) [12].

In contrast, Dealer does not involve VM migration, and its lightweight nature enables adaptation at shorter time-scales. Dealer offers several advantages in that it can be used in PaaS cloud platforms (e.g., Microsoft Azure) where VM migrations are not possible. Moreover, techniques like [27] rely on VM-level utilization metrics which may not suffice to detect and adapt to all types of poor performance episodes in the cloud (e.g., storage or network related problems). For instance, a network congestion may negatively impact the performance between two inter-connected VMs; however, VM-level metrics such as network utilization may still be low. On the contrary, Dealer focuses on overall application performance and can detect and adapt to such issues by appropriately reassigning requests to the best combination of replicas.

In [17], a multiobjective list scheduling approach for workflow applications is proposed. Based on a set of objectives constraints and weights defined by user, the algorithm attempts to find an appropriate Pareto solution in the region of interest for

the users. The algorithm is customized and analyzed for four objectives: makespan, cost, reliability, and energy.

The heuristics presented in [22] do not consider task execution times when making their decisions. In [23], a survey of dynamic scheduling heuristics for distributed computing systems is provided. All heuristics, except one, in [23] schedule tasks on different machines using load sharing techniques, without considering task execution times. (The one heuristic in [23] that does not use load sharing, employs deadlines to schedule tasks, and therefore falls out of the problem domain discussed here.) The load balancing heuristic used in this research is representative of the load balancing techniques in [22] and [23].

III . HEURISTIC TASK SCHEDULING ALGORITHM

Minimum execution time (MET) can potentially create load imbalance across machines by assigning many tasks to few machines than to others, whereas the minimum completion time (MCT) tries to balance the load by assigning tasks with earliest completion time. If the tasks are arriving in a random mix, it is possible to use the MET at the expense of load balance until a given threshold and then use the MCT to smooth the load across the machines. The Heuristic Task Scheduling algorithm uses the MCT and MET heuristics in a cyclic fashion depending on the load distribution across the machines. The purpose is to have a heuristic with the desirable properties of both the MCT and the MET.

Proposed Dynamic Heuristic Task Scheduling algorithm is based on the data dependencies among applications and distributing the application among multiple data centers at the task level, leading to more flexible and more efficient task scheduling. In this section task scheduling is defined with respect to two objectives: minimising makespan and maximising resource utilization. Each Individual set of task *i* is denoted as a pair *(i,j)* meaning that the task i is assigned to resource *j*. The schedule of entire workflow is denoted as

$$sched=\{ (i,j)|i\epsilon[n]\} \quad \textbf{(1)}$$

To calculate the real completion time of a task, the resource considers a number of internal details such as virtual machine startup overhead, latency delay, current load, computing power, availability, ready time, communication bandwidth, and so on [15]. Here is an assumption that these calculations are performed by each provider internally and therfore ingore them in our model.

After the completion of each task the user will be aware of t^{real} i,j but will still not know c^{real} i,j. Based on the real completion time of a single task , the makespan is defined as the time required for executing the whole task without violating the control and dataflow dependencies betwen its tasks

$$Makespan\ (DAG, sched) = max_{i\epsilon[n]\wedge(i,j)\epsilon\ sched} \quad \textbf{(2)}$$

A. Dynamic Heuristic Task Scheduling Algorithm

The expected execution time e_{ij} of task t_i on machine m_j is defined as the amount of time taken by m_j to execute t_i given m_j has no load when t_i is assigned. The time e_{ij} includes the time to move the t_i code and data from each of their corresponding single fixed sources to machine m_j. The expected completion time c_{ij} of task t_i on machine m_j is defined as the wall-clock time at which m_j completes t_i.

Let the arrival time of the task t_i be b_i, and the execution t_i begins execution e_{ij}. From the above definitions, $c_{ij} = b_i + e_{ij}$. Let c_i be the completion time for task t_i, and is equal to c_{ij} where machine m_j is assigned to execute task t_i. The makespan [21] for the complete schedule is then defined as $max_{ti\ \epsilon\ K(ci)}$. Makespan is a measure of the throughput and does not measure the quality of service imparted to an individual task.

Reviewing Biased random sampling and weighted least connection heuristics, it can be seen that depending on the weight assigned to each tasks in MT (meta-task), one of these heuristics has better results than the other one [7]. The independent set of tasks that is considered for mapping at the mapping events is called a meta-task. A meta-task can include newly arrived tasks and the ones that were mapped in earlier mapping events but did not begin execution.

The Proposed algorithm uses iterative scheduling heuristic that first orders the tasks in a list and then schedules them in the established order. The tasks are ordered according to longest path from each task to the exit task, including the task itself. The rank of task according to the following iterative function is given below

$$rank(i)= \begin{cases} load(i)+max\{comm(i,i)+rank(i)\} \\ load(i) \end{cases} \quad \textbf{(3)}$$

While there are tasks in MT
for all tasks t_i in MT
for all machines m_j
MT = min(tr({f_i},$r_i\epsilon${R})) for each input task t_i
for all tasks t_i in MT
if ($CT_{ij} \leq= MT$)
Assign t_1 to resource m_1 that obtains CT_{ij} else
Assign t_s to resource m_1 that obtains CT_{s1}
Delete assigned task from MT.
Update resource availability information based on the mapping of tasks
End While , End for
For all Task$_i$ reselected
For all Machine j with
(CT_j >=MeanCT)
Compute New $CT_{i,j}$ = CT(task$_i$, host$_j$)
if($NewCT_{i,j}$ >= MT)
Update resource availability information based on the mapping of tasks
Compute $NewCT_{m,n}$ (goto 6) , End if , End for, End for

Fig 1: Dynamic Heuristic Task Scheduling Algorithm

IV . EXPERIMENTAL RESULTS

Task partitioning scheduling heuristics is used for computing static schedules (offline); and iterative rescheduling is used for computing dynamic schedules (online). Depending on what scheduling performance is desired in cloud there exist different performance metrics for evaluating these algorithms. The experimental testing of our heuristic is performed in 2 scenarios.

Scenario I: A few short tasks along with many long tasks.

Scenario II: A few long tasks along with many short tasks.

Number of resources is chosen to be 10. Four different numbers of tasks has been chosen: 200,400, 600 and 800, to be sure of efficiency of the proposed heuristic [10]. The task arrivals are modeled by a Poisson random process.

Task workloads and data sizes to be transmitted between the tasks from historical data and running this workflow is extracted using ASKALON environment in the Australian grid. An assumption is made such that the resources are fully connected by a 10 GB ethernet network. For each task, a set of compute resources {R} is formed by selecting only those resources that have minimum transfer time (MTT).

$$MTT = min\ (tr\ (\{f_i\} r_j \epsilon \{R\})) \quad \textbf{(4)}$$

where f_i = *input file*
t_i = *task*

The Transfer time value

$$tr\ (\{fi\},resource) = m_{ij} = m[file,resource] \quad \textbf{(5)}$$

is obtained from the task resource matrix for all the resources.

$$CT_{i,j} = CT(task_i, host_j) \quad \textbf{(6)}$$

Makespan: Makespan is a measure of the throughput of the heterogeneous computing systems, such as cloud. It can be calculated as the following relation:

$$Makespan = max\ (Ct_i) \quad \textbf{(7)}$$
$$where\ \ t_i \epsilon MT$$

The less the makespan of a scheduling algorithm, the better it works.

Average resource utilization rate: Average resource utilization of total resources is calculated through the following relation:

$$Ru = \sum_{J=1}^{m} ru_{j/m} \quad (8)$$

where ru is in the range 0 to 1.

The commercial cloud environment {m} comprises of m selfish cloud providers, such as amazon and go grid. Cloud information directory service is used to store information about the resources belonging to the existing cloud providers. Proposed scheduling algorithm is used directly in the layers of the cloud architecture by selecting the most adequate resources in terms of makespan and resource utilization.

An assumption is made such as the resource j is able to calculate the real completion time t^{real} i,j and the real resource utilization r^{real} i,j for executing every task i. For each resource the real completion time of a task is sum of two components: the time to transfer the input data and the effective execution time.

Here $SRM_{im,jn}$ is the number of user request that must be directed between component *i* in data-center *m* to component *j* in data-center *n*, for every <component, data-center > pair. In determining the split ratio matrix, algorithm considers several factors including i) the total response time; ii) stability of the overall system; and iii) capacity constraints of application components.

In the discussion, a combination refers to an assignment of each component to exactly one data-center. The algorithm iteratively assigns a fraction of requests to each combination. The SRM matrix is easily computed once the fraction of requests assigned to each combination is determined.

Cloud Simulator is used for the evaluation of proposed algorithms as cloudsim is a new, generalized and extensible simulation toolkit and application which enables seamless modeling, simulation, and experimentation of emerging cloud computing system, infrastructures and application environments for single and internetworked clouds.

The existing distributed system simulators were not applicable to the cloud computing environment due to evaluating the performance of cloud provisioning policies, services, application workload, models and resources under varying system, user configurations and requirements. To overcome this challenge, CloudSim can be used. In simple words, CloudSim is a development toolkit for simulation of Cloud scenarios. CloudSim is not a framework as it does not provide a ready to use environment for execution of a complete scenario with a specific input. Instead, users of CloudSim have to develop the Cloud scenario it wishes to evaluate, define the required output, and provide the input parameters.

The algorithm begins by checking if the delay is unacceptably high. In such case, if $NewCT_{i,j} >= MeanCT$, the threshold is lowered. Otherwise the threshold remains unchanged and the component capacity is lowered to the threshold. If $CT_{i,j}$ is comparable to Delay, it is an indication the component can take more load. If $NewCT_{i,j} <= MeanCT$, then the threshold is too conservative, and hence it gets increased.

Further ComponentCapacity is set to slightly higher than the threshold to experiment if the component can absorb more requests. If however $NewCT_{i,j} = MeanCT$, then ComponentCapacity is set to Thrash to allow more requests be directed to that component.

All experiments were conducted on Microsoft Azure by deploying each application simultaneously in two data-centers. In all experiments, application traffic to one of the data-centers is controlled by Proposed Algorithm, while traffic to the other one was run without Proposed Algorithm . The objective was to not only study the effectiveness of Proposed Algorithm in enhancing performance of traffic but also ensure that Proposed Algorithm did not negatively impact performance of traffic.

A. Dynamic heuristic task scheduling Algorithm – scenario I

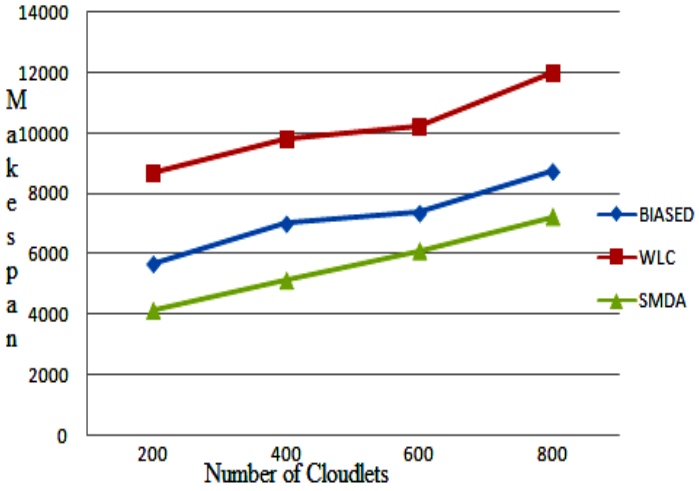

Fig 2: Shows Average Makespan Vs Number of Cloudlets

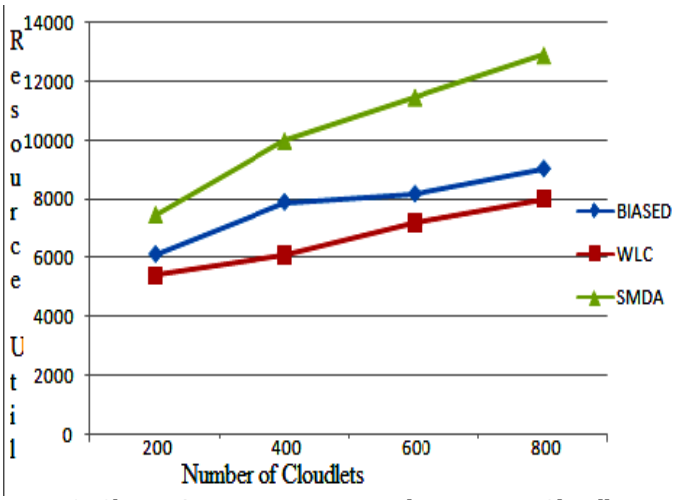

Fig 3: Shows Average Resource utilization Vs Cloudlets

B. Dynamic heuristic Task Scheduling Algorithm - Scenario II

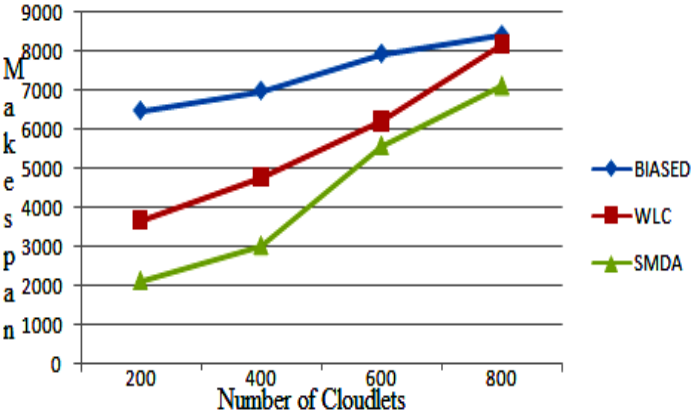

Fig 4: Shows Average Makespan Vs Number of Cloudlets

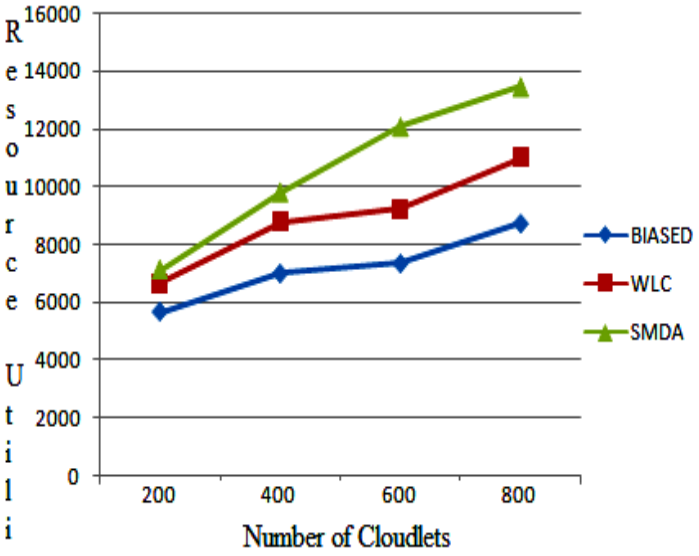

Fig 5: Shows Average Resource utilization Vs Cloudlets

C. Static Vs Dynamic Scheduling - Scenario I

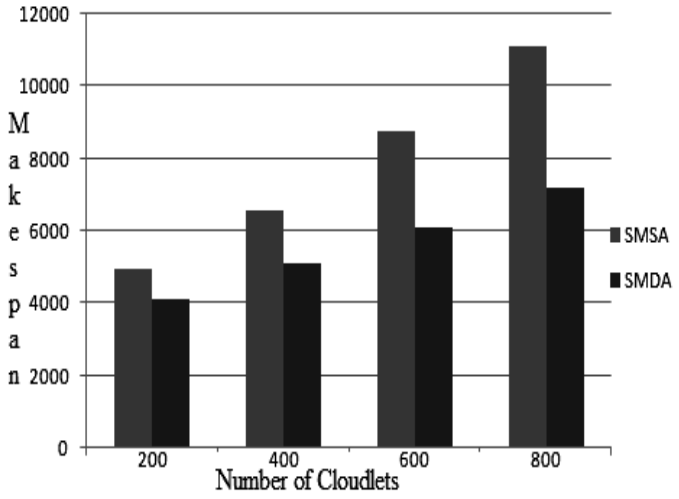

Fig 6: Shows Average Makespan Vs Cloudlets

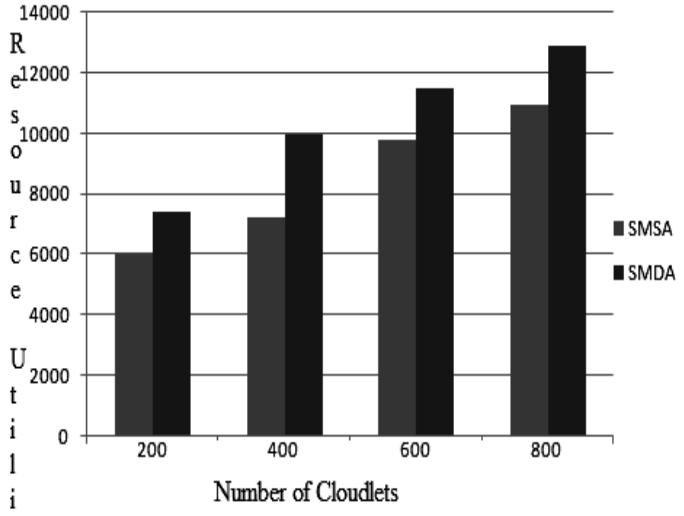

Fig 7: Shows Average Resource utilization Vs Cloudlets

D. Static vs dynamic scheduling - Scenario II

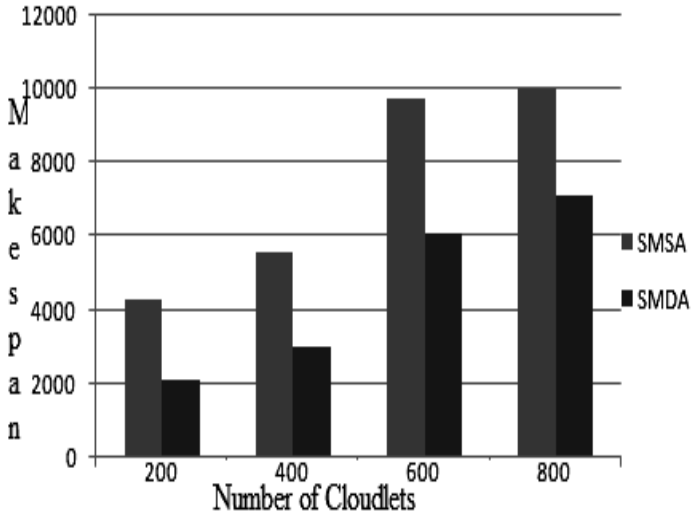

Fig 8: Shows Average Makespan Vs Cloudlets

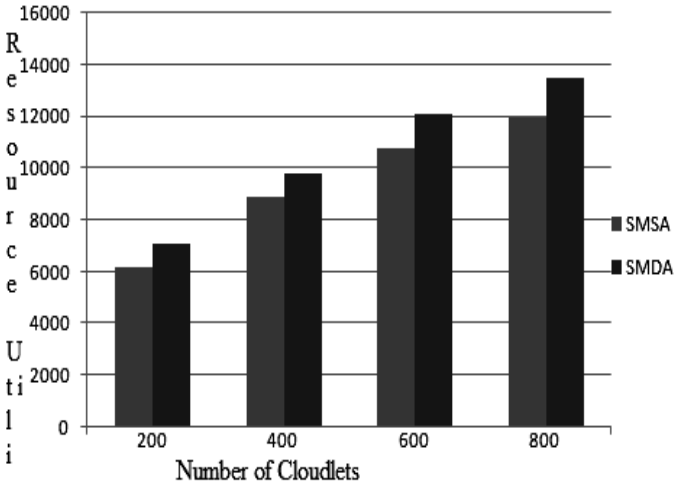

Fig 9: Shows Average Resource utilization Vs Cloudlets

V. CONCLUSION

Dynamic capacity provisioning has become a promising solution for data centers in recent years. However, existing work on this topic has not addressed a key challenge, which is the heterogeneity of workloads. In this paper, we first provide a characterization of both workload and machine heterogeneity found in one of the compute clusters. To achieve high computing throughput in a cloud environment, this new scheduling algorithm was proposed. We proved a lower bound on the expected makespan by developing a dynamic Heuristic Task scheduling algorithm, which is based on the Mean of minimum completion time of all unassigned tasks in a meta-task.

Evaluation of our new heuristic was done through a simulation environment called Cloudsim. The performance of the proposed dynamic Heuristic Task scheduling algorithm was compared with existing dynamic scheduling algorithms, i.e., Biased Random sampling and weighted least count algorithms. The experimental results show that the Dynamic Heuristic Task Scheduling Algorithm outperforms the Existing Algorithm in terms of makespan and resource utilization.

REFERENCES

[1] I. Foster, "Software for Service-Oriented Systems", *International Conference on Network and Parallel Computing, Springer-Verlag*, pp. 2-13, Oct 2005.

[2] R. Raman , et.al., "Matchmaking: Distributed Resource Management for High Throughput Computing", *In Proc. 7th IEEE Int. Symposium on High Performance Distributed Computing, Chicago*,pp. 2341-2349, Jan 2009.

[3] E. Bagheri, M. Naghibzadeh, "A New Approach to Resource Discovery and Dissemination for Pervasive Computing Environments Based on Mobile Agents", *Scientia Iranica Journal*, Vol. 14, No. 6, pp. 612-624, , Mar 2007.

[4] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, R. Freund," Dynamic mapping of a class of independent tasks onto heterogeneous computing systems", *8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pp. 30-44, Jan 2002.

[5] F. Dong, S. G. Akl: Scheduling Algorithms for Grid Computing, "State of the Art and Open Problems", *Open Issues in Grid Scheduling Workshop, School of Computing, University Kingston, Ontario*, pp.89-96, Mar 2006.

[6] D. Fernández-Baca, "Allocating modules to processors in a distributed system", *IEEE Transactions on Software Engineering*, Vol. 15 , No. 11, pp. 1427-1436, Nov 2011.

[7] R. F. Freund, M. Gherrity, et.al. "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", *7th IEEE Heterogeneous Computing Workshop (HCW'98)*, pp. 184–199, June 2008.

[8] H. Casanova, A. Legrand, D. Zagorodnov, "Heuristics for Scheduling Sweep Applications in Cloud Environments", *In Proc. 9th heterogeneous Computing Workshop (HCW'00), Cancun, Mexico*, pp. 349-363, Nov 2010.

[9] X. He, X. Sun, G. Laszewski, "A QoS Guided Min-Min Heuristic for Grid Task Scheduling", *Journal of Computer Science and Technology*, Vol. 18, No. 4, pp. 442-451, June 2013.

[10] M. Wu, W. Shu, H. Zhang, "A Static Mapping Algorithm for Meta- Tasks on Heterogeneous Computing Systems", *In Proc. 9th Heterogeneous Computing Workshop, Cancun, Mexico*, pp. 375-385, Mar 2012.

[11] J. Cao, D. P. Spooner, S. A. Jarvis, G. R. Nudd, "Grid Load Balancing Using Intelligent Agents", *Future Generation Computer Systems*, Vol. 21, No. 1, pp. 135-149, Apr 2005.

[12] R. Buyya, M. Murshed: GridSim, "A toolkit for the modeling and simulation for grid computing", *Journal of Concurrency and Computation: Practice and Experience*, Vol. 14, No. 13-15, pp. 1175–1220, Oct 2007.

[13] K. Aida, A. Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi, "Performance evaluation model for scheduling in computing system", *Journal of High Performance Computing Applications*, Vol. 14, No. 3, pp. 268-279, Apr 2009.

[14] H. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, " The MicroGrid: A scientific tool for modeling computational Grids", *In Proc. IEEE Supercomputing*, pp. 53-75, July 2009.

[15] H. Casanova, "A toolkit for the simulation of application scheduling", *In Proc. 1st International Symposium on Cluster Computing*,vol 2. no. 4, pp. 430-441, Dec 2012.

[16] J. Sherwani, N. Ali N, et.al., "Libra: A Computational Economy based Job Scheduling", *Journal of Software: Practice and Experience*, Vol. 34, No. 6, pp. 573-590, June 2011.

[17] J. Cropper, " A Cost-Benefit Perspective of Grid Computing", *In Proc. 4th Annual WCSURS*, vol 8, iss. 5, pp. 13-21, Dec 2013.

[18] C. S. Yeo, R. Buyya, "Pricing for Utility-driven Resource Management and Allocation in Clusters", *International Journal of High Performance Computing Applications*, Vol. 21, No. 4, pp. 405-418, Oct 2007.

[19] T.Fahringer, et.al., Aug 2005 , " Askalon: A grid application development and computing environment " *Prc.sixth IEEE/ACM int'l conf.grid computing*, pp.122-131, Aug 2011.

[20] Hamid Mohammadi fard,et.al., " A truthful dynamic workflow scheduling mechanism for commercial multicloud environment", *IEEE Transsactions on parallel and distributed systems*, vol.24, no.6, pp.1203-1212, June 2013.

[21] Mohammad hajjat, et.al., "Dynamic request splitting for intractive cloud applications", *IEEE Jornal on selected areas in communications* vol.31,no.12,pp. 2722-2737, Dec 2013.

[22] R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Adaptive load sharing in heterogeneous distributed systems", *Journal of Parallel and Distributed Computing*, vol 9, No. 4, pp 331–346, Aug. 2012.

[23] H. G. Rotithor, "Taxonomy of dynamic task scheduling schemes in distributed computing systems", *IEEE Proceedings on Computer and Digital Techniques*, vol 141, No. 1 ,pp 1–10, Jan. 2013.

[24] Wood T.et.al.,"Dynamic pooling of cloud resources by live WAN migration of virtual machines", IMC, vol 7, No.4, pp 546-552,july 2011.

[25] BARKER, S.,AND SHENOY, P. "Empirical evaluation of latency-sensitive application performance in the cloud", In MMSys, vol.13,no.1,pp. 222-237, Dec 2010.

[26] R. N. Calheiros et al., "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, Vol.41, No.1, pp.23-50, 2011.

[27] R. N. Calheiros et al., "CloudSim: a novel framework for modeling and simulation of cloud computing infrastructure and services," Technical Report of GRIDS Laboratory, The University of Melbourne, Australia, 2009.

Authors

Ms.M.Krishna sudha graduated in 2001 with a degree in Science from Bharathiar University, Coimbatore. She obtained her Master Degree in Science and M.Phil in Computer Science also from the Bharathiar University. She has 10 years of teaching experience starting from Lecturer to Department Head. At present she is working as Assistant Professor & Head of Information Technology in Sri Vasavi College College, Erode, Tamilnadu. She has guided more than 15 M.Phil research Scholars in various fields. Currently she is Guiding 4 M.Phil Scholars leading to Research. She published Many research papers in national and international journals and conferences. Her current research interests include Computer Networking, Cloud Computing. and Data Mining.

Dr. S. Sukumaran graduated in 1985 with a degree in Science from Bharathiar University, Coimbatore. He obtained his Master Degree in Science and M.Phil in Computer Science also from the Bharathiar University. He received the Ph.D degree in Computer Science from the Bharathiar University. He has 25 years of teaching experience starting from Lecturer to Associate Professor. At present he is working as Associate Professor of Computer Science in Erode Arts and Science College, Erode, Tamilnadu. He has guided for more than 40 M.Phil research Scholars in various fields and guided 2 Ph.D Scholar. Currently he is Guiding 8 M.Phil Scholars and 8 Ph.D Scholars. He is member of Board studies of various Autonomous Colleges and Universities. He published around 25 research papers in national and international journals and conferences. His current research interests include Image processing and Data Mining.