International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 2, Special Issue
September 2015

Improving Performance Effciently in Many-Task Computing in Cloud

R.Siva Malar
Lecturer, Department of Computer Science and Information System,
Jazan University, Ministry of Higher Education,
Jazan, Kingdom of Saudi Arabia. .

Abstract*—Cloud Computing has demonstrated applicability to a wide-range of problems in several domains, including scientific one. Many-Task Computing (MTC) is a practical paradigm for developing loosely coupled and complex scientific application. Many-Task Computing aims to bridge the gap between High Throughput Computing (HTC) and High Performance Computing (HPC). Cloud Computing services for Many-Task Computing was introduced in the existing system which investigates the presence of MTC component in the existing scientific computing workloads. It also analyse the empirical performance evaluation of Cloud Computing services for scientific computing workloads. However, the compute performance of the tested clouds is low in the existing system. In this paper, Data Aware Scheduling algorithm has been proposed to improve the efficiency and performance of Many-Task Computing. We have evaluated our approach by executing simulated experiments and also analysed base on various parameters like caching performance, Execution time, workload and utilization. The experimental result shows that the proposed Data Aware Scheduling algorithm can achieve much more better performance and the utilization of system resources.*

Keywords— Cloud Computing, Task scheduling, Many-Task Computing, High Throughput Computing, High Performance Computing.

I. INTRODUCTION

Cloud Computing [1] has emerged as an alternative computing model where Web-based services allow for different kinds of users to obtain a large variety of resources, such as software and hardware. Cloud Computing has demonstrated applicability to a wide-range of problems in several domains, including scientific ones. The Cloud Computing [2] paradigm holds great promise for the performance of scientific computing community. Clouds can be a cheap alternative to supercomputers and specialized clusters, a much more reliable platform than grids, and a much more scalable platform than the largest of commodity clusters. However, clouds raise important challenges in many aspects of scientific computing, including performance, which is the focus of this work.

Scientific Computing [3] is the field of study concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyze and solve scientific problems. In practical use, it is typically the application of computer simulation and other forms of computation problems in various scientific disciplines. The field is distinct from computer science (the study of computation, computers and information processing). It is also different from theory and experiment which are the traditional forms of science and engineering. The scientific computing approach is to gain understanding, mainly through the analysis of mathematical models implemented on computers. Scientific Computing is the reconstruction or prediction of phenomena and processes, especially from science and engineering, on supercomputers.

Recently, the scientific computing community has started to focus on Many-Task Computing (MTC) [4], that is, on high-performance execution of loosely coupled applications comprising many tasks. Performance wise, scientific workloads often require High-Performance Computing (HPC) of High-Throughput Computing (HTC) capabilities. Many-Task Computing is reminiscent to High Throughput Computing, but it differs in the emphasis of using many computing resources over short periods of time to accomplish many computational tasks (i.e. including both dependent and independent tasks), where the primary metrics are measured in seconds (e.g. FLOPS, tasks/s, MB/s I/O rates), as opposed to operations (e.g. jobs) per month. MTC denotes high-performance computations comprising multiple distinct activities, coupled via file system operations. Tasks may be small or large, uniprocessor or multiprocessor, compute-intensive or data-intensive. The set of tasks may be static or dynamic, homogeneous or heterogeneous, loosely coupled or tightly coupled. The aggregate number of tasks, quantity of computing, and volumes of data may be extremely large. MTC is used for bridge the gap between High Throughput Computing and High Performance Computing.

Recently the paradigm of Many Task Computing was proposed [5] to solve the problem of executing multiple parallel tasks in multiple processors. This paradigm consists on several computing resources used over short periods of time to accomplish many computational tasks. Some existing SWfMS are already exploring workflow parallelism using MTC in homogeneous computing environments, such as computational clusters [6][7]. This type of environment relies on centralized control of resources, which eases parallelism and exploits high speed communication networks which brings high performance to scientific experiments.

Due to the move of scientific experiments to clouds and increasing demands of those experiments for parallelization, executing parallel scientific workflows in clouds is still a challenge. Although clouds provide elastic resources that can be used to execute parallel instances of a specific scientific workflow, it is still difficult for scientists to express a parallel computing paradigm for the workflow on the cloud. It is possibly feasible for scientists to do this task in an ad-hoc way. However, ad-hoc approaches are not scalable; they may be exhaustive for scientists and error – prone and may become a barrier to execute the entire experiment. In this case, a specialized scheduling algorithm is required to simplify this execution, thus structuring the parallel execution and isolating scientists from its complexity. Existing approaches for executing scientific workflows using parallel processing are mainly focused on homogeneous environments [8] [9] whereas on the cloud, the scientist has to manage some new important aspects such as initialization of virtualized instances, scheduling workflow activities over different cloud environments, impact of data transferring and management of instance images.

In this paper, Data Aware Scheduling Algorithm for Cloud environment to support MTC paradigm in scientific workflow has been introduced. The proposed Data Aware Scheduling algorithm improves the efficiency and performance of Many-Task Computing. Our approach has been evaluated by executing simulated experiments and also analysed base on various parameters like caching performance, Execution time, workload and utilization. The experimental result shows that the proposed Data Aware Scheduling algorithm can achieve much better performance and the utilization of system resources.

This paper is organized as follows. Section II describes metrics and methods of Cloud environment, Section III describes the Data Aware Scheduling Algorithm for Many - Task Computing and Section IV describes the performance evaluation. Section V shows the experimental results and conclusion in Section VI.

II METRICS AND METHODS

A. *Cloud Computing*

Cloud Computing is a marketing term for technologies that provide computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. A parallel to this concept can be drawn with the electricity grid, wherein end-users consume power without needing to understand the component devices or infrastructure required to provide the service. Cloud Computing [10] describes a new supplement, consumption, and delivery model for IT services based on Internet protocols, and it typically involves provisioning of dynamically scalable and often virtualized resources. It is a product and consequence of the ease-of-access to remote computing sites provided by the Internet. This may take the form of web-based tools or applications that users can access and use through a web browser as if the programs were installed locally on their own computers.

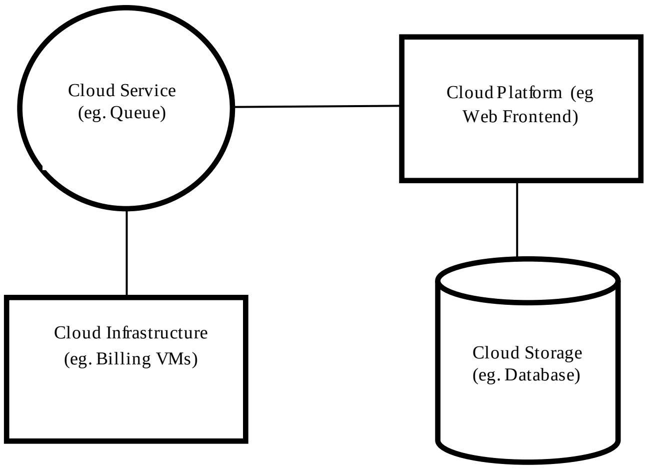

Figure 1. Cloud Computing Architecture

Cloud Computing is Internet-based development and use of computer technology [11]. The Cloud is a metaphor for the Internet and is an abstraction for the complex infrastructure it conceals. Cloud computing is a model for enabling ubiquitous, convenient,, on-demand network access to a shared pool of configurable computing resources (eg., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Figure 1 shows the architecture of the Cloud Computing.

The majority of Cloud Computing infrastructure consists of reliable services delivered through data centers and built on servers with different levels of virtualization technologies. The Cloud appears as a single point of access for all the computing needs of consumers.

Cloud Computing is the development of grid computing, parallel computing and distributed computing. It is a new pattern of business Computing. The basic mechanism of Cloud Computing is to dispatch the Computing tasks to resource pooling which constitutes by massive computers. It enables a variety of applications to gain Computing power, storage and a variety of software services according to their needs. The commercialization and the virtualization technology adopted by Cloud Computing have poured into new features for Cloud architecture. By using Clouds as the application hosting platform, IT companies are freed from the trivial task of setting up basic hardware and software infrastructures. Thus they can focus more on innovation and creation of business values for their application services [12].

Some of the traditional and emerging Cloud – based application services includes social networking, web hosting, content delivery, and real time instrumented data processing. Each of these application types has different composition, configuration, and deployment requirements. Quantifying the performance of provisioning (scheduling and allocation) policies in a real Cloud Computing environment (Amazon EC2 [13], Microsoft Azure [14], Google App Engine [15] for different application models. The Cloud Computing holds great promise for the performance of scientific computing community: Recently, the Scientific Computing community has started to focus on Many-Task Computing, that is, on high-performance execution of loosely coupled applications comprising Many Tasks.

B. *HPC/HTC Computing*

The High Performance/High Throughput (HPC/HTC) Computing systems are research resources intended for testing and running large codes, parallel-processing codes, visualization and scientific applications. HPC (High Performance Computing) systems enable users to run a single instance of parallel software over many processors. HTC (High-Throughput Computing) serial systems are more suited to running multiple independent instances of software on multiple processors at the same time. High Performance Computing/High Throughput Computing (HPC/HTC) systems achieve enormous computing power by using hundreds of individual processors (cores) to break up huge computing projects into many smaller pieces.

There are many differences between High-Throughput Computing and High-Performance Computing. HPC tasks are characterized as needing large amounts of computing power for short periods of time, whereas HTC tasks also require large amounts of computing, but for much longer times (months and years, rather than hours and days). HPC environments are often measured in terms of FLOPS. The HTC community, however, is not concerned about operations per second, but rather operations per month or per year. Therefore, the HTC field is more interested in how many jobs can be completed over a long period of time instead of how fast an individual job can complete. As a general rule, HPC systems are tightly coupled parallel jobs, and as they must execute within a particular site with low-latency interconnects. Conversely, HTC systems are independent, sequential jobs that can be individually scheduled on many different computing resources across multiple administrative boundaries. HTC systems achieve this using various grid computing technologies and techniques.

C. *Many-Task Computing*

Clouds have been the preferred platform for loosely coupled applications that have been traditionally part of the high throughput computing class of applications, which are managed and executed through workflow systems or parallel programming systems. Various properties of new emerging applications, such as large number of tasks (i.e. Millions or more), relatively short per task execution times (i.e. seconds to minutes long), and data intensive tasks (i.e. tens of MB of I/O per CPU second of compute) have leads to the definition of a new class of applications called Many-Task Computing. MTC paradigm [16] embraces different types of high-performance applications involving many different tasks, and requiring large number of computational resources over short periods of time. MTC applications are composed of many tasks (both independent and dependent tasks) that can be individually scheduled on many different computing resources across multiple administrative boundaries to achieve some larger application goal.

MTC aims to bridge the gap between HTC and HPC. MTC is reminiscent of HTC, but it differs in the emphasis of using many computing resources over short periods of time to accomplish many computational tasks (i.e. including both dependent and independent tasks), where the primary metrics are measured in seconds (e.g. FLOPS, tasks/s, MB/s I/O rates), as opposed to operations (e.g. jobs) per month. MTC denotes high-performance computations comprising multiple distinct activities, coupled via file system operations.

Many-Task Computing applications span a broad range of possible configurations, but utilizing large numbers of computing resources over short periods of time to accomplish many computational tasks, where the primary metrics are in seconds". Here the system investigate the presence of a MTC component in Scientific Computing workloads and quantify the presence of these users in Scientific Computing environments. Also evaluate with well-known micro benchmarks and application kernels the performance of four commercial Cloud Computing services that can be used for Scientific Computing. Their method for identifying proto-MTC users with a pronounced MTC-like workload, which are potential MTC users in the future in existing system workloads is based on the identification of users with many submitted tasks and/or bags-of-tasks in the workload traces taken from real Scientific Computing infrastructures. The system didn't consider any the scheduling algorithms for Many-Task Computing. Therefore this system does not provide the optimal performance of the system.

III DATA AWARE SCHEDULING

Data-aware scheduling is central to the success of data diffusion, as harnessing data-locality in application access patterns is critical to performance and scalability. As demand increases, more resources are acquired, thus allowing faster response to subsequent requests that refer to the same data; when demand drops, resources are released.

There are four dispatch policies such as FA, MCH, MCU and GCC. The First-Available (FA) policy ignores data location information when selecting an executor for a task; it simply chooses the first available executor. The Max-Cache-Hit policy uses information about data location to dispatch each task to the executor with the largest amount of data needed by that task. If that executor is busy, task dispatch is delayed until the executor becomes available. The Max-Compute-Util (MCU) policy leverages data location information, attempting to maximize resource utilization even at the potential higher cost of data movement. The Good-Cache-Compute (GCC) policy is a hybrid MCH/MCU policy.

Algorithm: Data Aware Scheduling for Many-Task Computing

```
Input: Request for object p at store X from sequence σ
1  if p is not on X then
2    if X is not full then /* No eviction required */
3      if p is on some store Y then
4        Transfer p from Y to X
5      else
6        Load p to X from persistent storage
7      end
8    else /* Eviction required to make space in X */
9      if all objects on X are local-marked then
10         local-unmark all /*Begins new local phase */
11       end
12       if p is on some store Y then
13         Select an arbitrary local-unmarked object q on
X
14         Exchange q and p on X and Y
           /* X now has p and Y has q */
15           if p was local-marked on Y then
16             local-mark q on Y
17           end
18     else /* p must be loaded from persistent storage */
19         if all objects in system are global-marked then
20         global-unmark and local-unmark all objects
             /*Begins new global phase & local phases
             at each store */
21           end
22           if all objects on X are global-marked then
23         Select an arbitrary local-unmarked object q on X
24           Select an arbitrary store Y with at least one
           global-unmarked object or empty space
```

```
25         Transfer $q$ to $Y$ , replacing an arbitrary global-
           unmarked object or empty space
26             else
27           Evict an arbitrary global-unmarked object $q$ on $X$
28           end
29           Load $p$ to $X$ from persistent storage
30           end
31         update its results
32       end
33   end
34   global-mark and local-mark $p$
```

Several variables are defined first in order to understand the scheduling algorithm pseudo-code and the algorithm is separated into two sections, as the first part decides which executor will be notified of available tasks, while the second part decides which task to be submitted to the respective executor. It provides high performance for Many Tasks Computing environment. Finally define the model and then analyze the computational time per task, caching performance, workload execution times, arrival rates, and node utilization.

Data-aware scheduling is not limited to the hurricane scenario but is useful across a wide variety of applications where there is a sense of relative importance of jobs. Application aware scheduling can be applied to regular day-to-day jobs submitted by users. Such scheduling not only helps improve utilization but also ensures that the limited amount of supercomputing time available to the scientists is spent doing more valuable science. We did not need any of the advanced features supported by these and using DAGMan was an obvious solution since the work management system was Condor.

A Directed Acyclic Graph (DAG) can be used to represent a set of computations where the input, output, or execution of one or more computations is dependent on one or more other computations. The computations are nodes (vertices) in the graph, and the edges (arcs) identify the dependencies. Condor/Stork finds machines for the execution of programs, but it does not schedule programs based on dependencies. The Directed Acyclic Graph Manager (DAGMan) is a meta-scheduler for the execution of programs (computations). DAGMan submits the programs to Condor/Stork in an order represented by a DAG and processes the results. A DAG input file describes the DAG, and further submit description file(s) are used by DAGMan when submitting programs to run under Condor/Stork. This keeps the number of software components to install low and keeps the software stack small. It is also often referred to the concept of data location aware computing, which aims at bringing computation to data. This model of computing is extremely popular when data sizes are large and transferring data between the compute units is prohibitively expensive. Schedulers are used to schedule data-transfers optimally such that no transfer is repeated and the computations can be performed on the transferred data without moving it to other locations. Hence data-aware scheduling algorithm is used that can perform data-placement optimizations.

IV PERFORMANCE EVALUATION

Finally evaluating the proposed approach with the existing approach for identify the utilization of resources and present the evaluation comparison by the parameter metrics such as the computational time per task, caching performance, workload execution times, arrival rates, and node utilization. Based on the comparison and the results from the experiment show the proposed approach works better than the other existing systems.

V RESULTS AND DISCUSSION

In this section, experiments are done to investigate the performance of the proposed algorithm.

Experimental Results

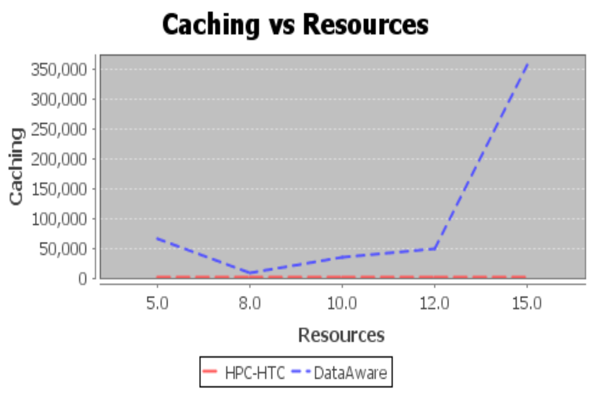

Figure 2.Caching performance vs No.of resources

From the above Figure 2, we can see that the caching performance of the Many-Tasks Scientific Computing will remain constant for any number of resources. When it is compared with our proposed

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 2, Special Issue
September 2015

method of Data Aware Scheduling algorithm it shows that if the number of resources increased the caching performance is increased linearly.

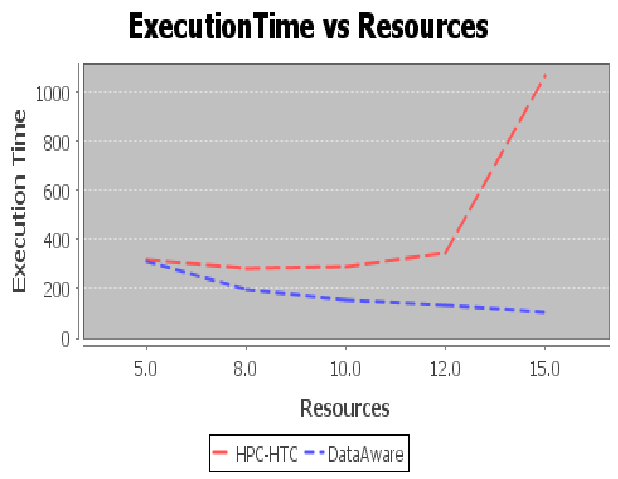

Figure 3.Execution time vs No.of resources

Although the execution time in both the cases is initially same, when number of resources increases, the execution time of our proposed algorithm decreases linearly. Figure 3 clearly shows that our proposed Data Aware Scheduling algorithm significantly outperforms the existing Many-task scientific computing method.

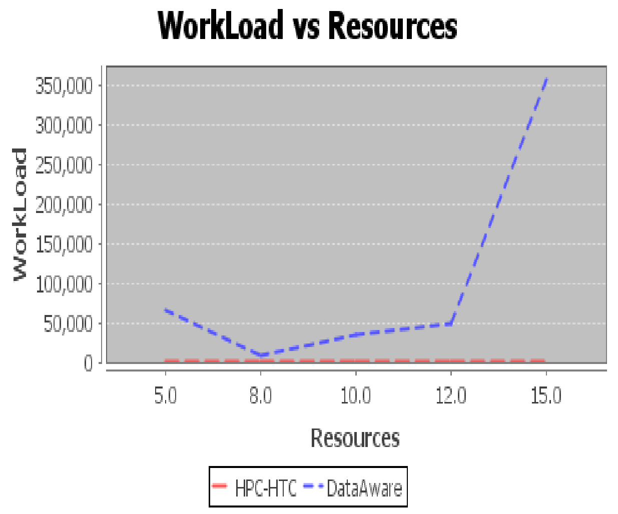

Figure 4.Workload vs No.of resources

Figure 4 analyse and compare the performance offered by Many-Tasks Scientific Computing with our proposed method of data aware scheduling algorithm. Here if the no of resources increased the Workload for resources is decreased linearly. Based on the comparison and the results from the experiment show the proposed approach works better than the other existing systems.

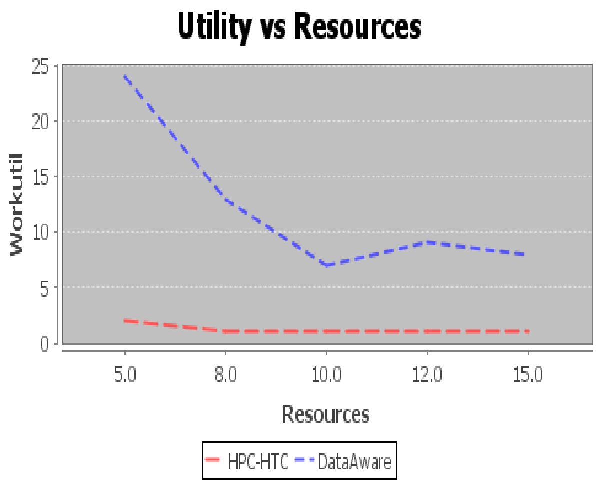

Figure 5.Utilization vs No.of resources

When we analyse and compare the performance offered by Many-Tasks Scientific Computing with our proposed method of data aware scheduling algorithm from Figure 5, it clearly shows that if the no of resources increased the utilization for resources is increased linearly. Based on the comparison and the results from the experiment show the proposed approach works better than the other existing systems.

VI CONCLUSION

The system introduces Cloud Computing services for Many-Task Computing and its applications span a broad range of possible configurations, but utilizing large numbers of computing resources over short periods of time to accomplish many computational tasks, where the primary metrics are in seconds".

The presence of a MTC component in Scientific Computing workloads and quantify the presence of these users in Scientific Computing environments and evaluate with well-known micro benchmarks and application kernels the performance of four commercial Cloud Computing services that can be used for Scientific Computing.

In this paper, we present Data Aware Scheduling algorithm approach which provides high performance for Many Tasks Computing environment, define the model and then analyze the computational time per task, caching performance, workload execution times, arrival rates, and node utilization. Based on the comparison and the results from the experiment, the proposed approach shows the better result than the other existing systems.

REFERENCES

[1] Oliveria.D, Baiao.F, and Mattoso.M, 2010, "Towards a Taxonomy for Cloud Computing from an e-Science Prespective", Cloud Computing: Principles, Systems and Applications (to be published), Heidelberg: Springer-Verlag.

[2] Alexandru losup, Simon Ostermann, Nezih Yigitbasi.M, Radu Prodan, Thomas Fahringer, and Dick H.J.Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing", IEEE Transactions on Parallel and Distributed Systems, Vol.22, No.6, June 2011.

[3] Jonas Dias, Eduardo Ogasawara, Daniel de Oliveria, Esther Pacitti, and Marta Mattoso, "Improving Many –Task Computing in Scientificc Workflow Using P2P Techiques", IEEE, 2010.

[4] Raicu.I, Zhang.Z, Wilde.M, Foster.I.T, Beckman.P.H, Iskra.K, and Clifford.B, "Toward Loosely Coupled Programming on Petascale Systems," Proc.ACM Conf. Supercomputing(SC), p.22,2008.

[5] Lin.C, Lu.S, Fei.X, Pai.D, and Hua.J, 2009, A Task Abstraction and Mapping Approach to the Shimming Problem in Scientific Workflows, In: Proc. Services 2009, p.284-291.

[6] Tylor.I.J, Deelman.E, Gannon.D.B, Shields.M, and (Eds.), 2007,, Workflows for e-Science:Scientific Workflows for Grids. 1 ed. Springer.

[7] Mattoso.M, Werner.C, Travassos.G.H, Braganholo.V, Murta.L, Ogasawara.E, Oliveria.D, Cruz.S.M.S.D, and Martinho.W, 2010 , Towards Supporting the Life Cycle of Large Scale Scientific Experiments, To be published Int. J, Business Process Integration and Management, in Special Issue on Scientific Workflows.

[8] Walker.E and Guiang.C, 2007, Challenges in executing large parameter sweep studies across widely distributed computing environments, In: Workshop on Challenges of large applications in distributed environments, p.11-18, Monterey, California, USA.

[9] Raicu.I, Foster.I, and Yong Zhao, 2008, Many –task computing for grids and supercomputers, In: Workshop on Many-Task Computing on Grids and Supercomputers, P, 1-11.

[10] Amazon Web Services LLC, "Amazon Simple Storage Service," http : / / aws.amazon.com/s3/, 2011.

[11] Cloud computing and distributed computing. http://www.cncloudcomputing.com/.

[12] Knorr.E and Gruman.G, "What cloud computing really means," http://www.infoworld.com, 2010.

[13] Amazon Web Services LLC, "Amazon Elastic Compute Cloud (Amazon EC2)," http:// aws.amazon. com/ec2/,2011.

[14] Microsoft Azure, http://www.microsoft.com/ windowsazure (accessed 15.01.2011)

[15] Google App Engine, http://code.google.com/ appengine, (accessed 14.01.2011)

[16] Rafael Moreno-Vozmedia, Ruben S. Montero and Ignacio M. Llorente "Multicloud Deployment of Computing Clusters for Loosely Coupled MTC Applications", IEEE transactions on parallel and distributed systems, Vol.22, No.6, 2011.

R.Sivamalar received the MCA, M.Phil and M.Sc degrees from Bharathiar University, India in 2006, 2008 and 2009 respectively. She also received her M.E degree from Anna University, India in 2012. Since 2012 she has been working as lecturer in Computer Science department, Jazan University, Kingdom of Saudi Arabia. She is currently doing her Ph.D in Computer Science at Jodhpur National University, India. Her research interests are in the areas of resource management and scheduling in the area of Cloud Computing.