A Novel Heuristics based Energy Aware Resource Allocation and Job Prioritization in HPC Clouds

Thamarai Selvi Somasundaram[1], Kannan Govindarajan[1], T.D. Rohini[1], K. Kavithaa[1], R. Preethi[1]
[1]Department of Computer Technology, Anna University, Chennai, Tamil Nadu, India
Email: stselvi@annauniv.edu, kannan.gridlab@gmail.com, rohinitd@gmail.com, kavithaa.aswi@gmail.com, preethi.r@gmail.com
Website: www.annauniv.edu/care

Abstract - *Cloud Computing provides the computational, storage, network and database resources to the consumers in a pay-as-per usage mode. In recent years, the data centers play a major role in hosting the cloud applications in the cloud infrastructure. The data centers are consuming huge electrical power and emits large amount of carbon footprint. It is essential to incorporate the Energy Efficient Resource Management (EERM) mechanism to control the electric power consumption and reduce the carbon footprint emission. EERA comprises of matching the user application requests with available cloud resources and allocating the user application requests to the matched cloud resources in an efficient manner. This paper mainly focused on proposing a novel heuristics based Energy Aware Resource Allocation (EARA) mechanism to allocate the user applications to the cloud resources that consumes minimal energy and incorporating the prioritization mechanism based on the deadline. It is simulated using the CloudSim toolkit and by generating High Performance Computing (HPC) type of application requests with the generated Eucalyptus based private Cloud environment. The results prove the effectiveness of the proposed mechanism in the cloud infrastructure by maximizing the number of users completed their applications within deadline and minimize the energy consumption in the cloud resources.*

Keywords: Cloud Computing; Resource Management; Energy Efficiency; Eucalyptus; Heuristics.

1. Introduction

Cloud Computing [1] provides ondemand computing in terms of application, platform and infrastructure in a pay as per usage mode. The Cloud service models are categorized into three major types based on the applications, platform and infrastructure namely SaaS, PaaS and IaaS. The IaaS service delivery model is plays a major role in hosting the PaaS or SaaS in the data centers. The four major players of the Cloud are (i) Cloud Users (CUs) (ii) Cloud Service Providers (CSPs) (iii) Cloud Applications (CAs) and (iv) Cloud Service Brokers (CSPs). The CUs are submitting the jobs with software, hardware and QoS parameters. The requirements are varied in terms of hardware (Processor Speed, RAM Memory, Bandwidth and etc.), software (Java 1.6, apache tomcat-5.0.27, MPICH-1.2.7, Charm++ 3.x and etc.) and QoS (deadline, throughput and etc.). The CSPs are managing the huge datacenters for the purpose of computation, storage and etc. CSPs are managing the physical resources to host the Cloud applications in the virtual resources. The CRPs have to consider the user required parameters when they are selecting the resources to run the user applications. In this scenario, CRP's are facing the problem in the selection of resources to run the application. The CAs may be of different types such as web sites, web applications, high-performance computing applications and etc. In recent years, the huge datacenters are popular for hosting the CAs. The CSBs acts as the mediator between the CUs, CSPs and user's CA, so it is essential to incorporate the efficient Resource Management (RM) technique. RM is the challenging task due to the dynamic nature of Cloud Computing environment and ondemand user requirement. It mainly consists of five major functionalities are shown in Figure 1 and they are (i) Matchmaking the user job requests with available cloud resources (Resource Discovery) (ii) Allocating the user job requests with available cloud resources in an efficient manner (Resource Selection) (iii) Provisioning of virtual resources in the selected resources (Resource Provisioning) (iv) Running the user jobs in the created virtual resources (Running Application) and (v) Monitoring the running applications (Monitoring Applications).

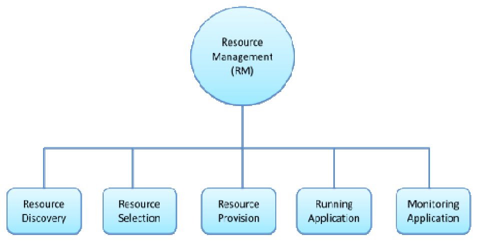


Figure 1: Functionalities of Resource Management (RM)

Nowadays scientific applications are becoming complex and it is composed of various application components and it requires heterogeneous set of resources. The High Performance Computing Clouds (HPCCs) or Science Cloud provides a great platform for researchers to test their ideas using simulation process. The scientific applications requires large amount of computational steps and customized execution environment and also it processes and generates huge amount of data. Cloud Computing provide benefits to the scientific applications using the concept of resource provisioning through virtualization technology and it provides different operating systems with different software configurations.However, it is very difficult to incorporate the efficient RM mechanism in every cloud provider site and also it is very tedious for the cloud user's to search the suitable cloud resources that are geographically distributed in nature to run their applications. These drawbacks can be achieved by integrating the efficient RM mechanism in CSB to efficiently manage the user requests and Cloud resources. In recent days, the data centers are consuming more amount of electrical energy and emitting large amount of carbon foot prints. The high energy consumption increases the running cost of the data centers. So, it is essential to decrease the high energy consumption of the data centers that will maximize the revenue of the resource providers, reduce the carbon emission and running cost of the data centers. To achieve the above objectives, we have proposed the energy efficient resource management mechanism that is mainly aimed to improve the maximum number of users completed their jobs within deadline and minimize the consumption of energy in the datacenters. These two factors influence the revenue of the cloud resource providers in an impressive manner. The maximum number of users completed within the deadline is achieved by giving more priority to the jobs nearer to the deadline. The resource selection process is carried out by employing the optimization algorithm of Particle Swarm Optimization [2]. The proposed approach selects the resources that consume less energy. In addition to that, it accommodates or allocates the maximum number of user requests in the datacenters that will increase the revenue of the service providers and increase the utilization of the resources. In brief the contributions of the research work are summarized below.

a. To design and develop the matchmaking algorithm for matching the HPC user requests with available cloud resources. (A)
b. To design and develop a Particle Swarm Optimization based Energy Aware Resource Allocation (PSOEAR) mechanism for allocating the user requests to the Cloud resources in a near optimal manner. (B)
c. Integration of (A) and (B) with Cloud Service Broker (CSB) for matchmaking, allocating and provisioning for the HPC user requests. (C)
d. The proposed work is simulated and the results have been analyzed in the simulation based cloud environment. (D)

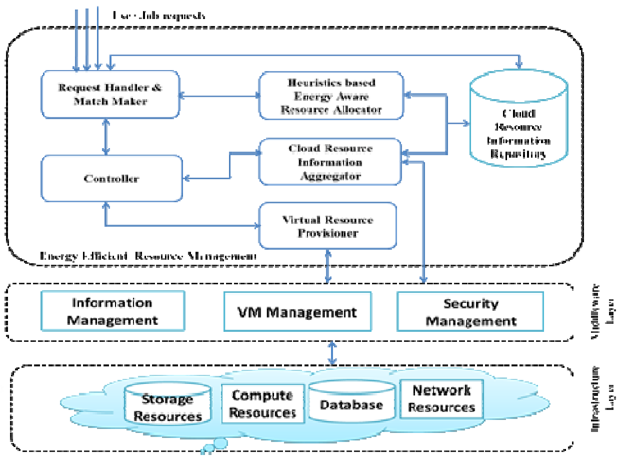
The rest of the paper is organized as follows: Section 2 presents the high-level architecture of proposed framework; Section 3 presents the proposed system model and its description. Section 4 describes the simulation results and its inferences observed from the simulation. Section 5 describes the related works closely related to our proposed work. Section 6 concludes the proposed work and explores the feasibility of future work.

2. Proposed High-Level Architecture

The proposed high-level architecture for cloud resource management framework with energy aware allocation is shown in Figure 2. It consists of the five major components and the functionalities of each component are described in detail.

2.1 Request Handler & Match Maker

The user submits the job requirements as an XML file or through Graphical User Interface (GUI). The parser in the request handler parses the job requirements and the parsed information is updated in the User Job Request Pool (UJRP). Once the job requests are parsed it invokes the Match Maker to match the user job requirements with the available cloud resources. The matchmaker component filters the potential resources that are capable of creating virtual resources and run the job. Finally, it generates the matched resource list and the generated list is sent to the Particle Swarm Optimization based Energy Aware Resource Allocator (PSOEARA). The matchmaking algorithm for HPC job request is shown below.



User Job requests
Request Handler & Match Maker
Heuristics based Energy Aware Resource Allocator
Cloud Resource Information Repository
Cloud Resource Information Aggregator
Controller
Virtual Resource Provisioner
Energy Efficient Resource Management
Information Management
VM Management
Security Management
Middleware Layer
Storage Resources
Compute Resources
Database
Network Resources
Infrastructure Layer

Figure 2: Cloud Resource Management Framework for Energy Aware Resource Allocation

Algorithm 1 Matchmaking Algorithm

```
Input  : Fetch the job requests with hardware, software and
         QoS requirements.
Output: Matchmaking the job requests with available cloud
         resources and generate the matched resource list.
Step 1 : Submit the job requests with requirements, parse the
         requirements and store it in the Broker Queue (BQ).
Step 2 : Match the job requirements with available cloud
         resources and generate the resource list that are
         capable of creating the virtual instances and run the
         job.
```

```
Step 3 : For (I= 1 to N 'Job Request')
          {
           For (J = 1 to M 'Datacenters')
           {
           For (K=1 to O 'Hosts')
           {
             Match the job requirements with available cloud
                resources.
             Generate the matched host list that is capable of
                satisfying the user job requests.
            }
             Generate Matched Datacenter List that has
             capable hosts to create the virtual instances
             and run the jobs.
          }
         }
Step 4 : End
```

2.2 Particle Swarm Optimization based Energy Aware Resource Allocator (PSOEARA)

PSOEARA is implemented with Particle Swarm Optimization (PSO) and Energy Aware Resource Allocation algorithm. PSO is a population based stochastic optimization technique. It is initialized with a group of random particles or solutions. Each particle is updated by the two best values known as pbest (the personal best) and gbest (the global best) in every iteration. Pbest represents the best solution achieved by one particle and gbest represents the best value obtained by any particle in the population. PSOEAR mainly consists of three fold processes such as (1) Initial job assignment to the matched resource list (2) Calculation of Expected Completion Time (ECT) and Energy Consumption (EC) of the job (3) Final job assignment to the selected cloud resource.

2.2.1 Initial job assignment to the matched resource list - PSOEARA takes the batch of jobs as input and each job is randomly allocated to the matched resource list. In each assignment the ECT and the EC is computed for each job.

2.2.2 Calculation of ECT and EC of the jobs - The ECT is computed using the Equation (1).

$$ECT_{ij} = ST_{ij} + BT_{ij} + EET_{ij} \quad - (1)$$

$$EETij = \text{Job Length} / \text{MIPS of VM} \quad - (2)$$

$$\text{Job Length} = \text{Million Instructions (MI)} / 60 \quad - (3)$$

$$\text{MIPS of VM} = \text{Job Length}/\text{Deadline} + x \quad - (4)$$

Where ECT_{ij} represents the expected completion time of job i on resource j, ST_{ij} represent the start time of job i on resource j, BT_{ij} represents the boot time of the virtual instances for the ith job on resource i, EETij represents the estimated execution time of job i on resource j. The EC is computed using the Equation (5) and it is given below.

$$EC_{ij}\,(\text{in watts}) = \sum_{i=1,j=1}^{N,M} RMIP_i / AMIP_j * 100 \quad - (5)$$

Where EC_{ij} represents the energy consumption of the particular job, RMIPS represents the Requested Millions of Instructions per Second for job i and AMIPS represents the Available Millions of Instructions Per Second in resource j. Where i= 1 To N represents number of VMs, j=1 To M represents the number of hosts.

2.2.3 Final job assignment to the selected cloud resource - In the final assignment process the jobs are assigned to the cloud resource which completes the job within the deadline and consumes less energy.

Algorithm 2 PSOEARA Algorithm

Input: Fetch the job requests with matched host list and datacenter list.

Output: Optimal selection of cloud resources that consumes less energy and completes the job requests within deadline.

```
Step 1   Get the 'N' number of job requests with matched
         datacenter list and the host list.
Step 4   For (I= 1 to N 'Job Request')
         {
        For (J=1 to M 'Virtual Machines' of each job)
        {
         For (K = 1 to O 'Matched Datacenter List')
         {
            GetMatchedHostList ( );
            For (L = 1 to P 'MatchedHostList')
           {
           Compute the Expected Completion Time (ETC)
           using the Equation (1);
           Compute the Present Energy Consumption (PEC)
           using the Equation (5);
           }
     If (K==0) {
     Pbest = PEC;
     Chosen DC= DC (0)
    }
    Else
    {
     If (PEC < Pbest && ECT < Deadline)  //Compare the
                              energy consumption difference
    {
    Gbest = PEC;
    Chosen DC = DC (K);
    }
    Else If
    {
     Gbest = Pbest;
    }
     }
     }
}}
```

2.3 Cloud Resource Information Aggregator

This work is extension of our previous work Cloud Monitoring and Discovery Service (CMDS) [3]. CMDS will aggregate the cloud resource information such as processor, memory and network using the external information providers Ganglia, NWS and our own user-defined script. CMDS is extended to aggregate the energy and load information from the cloud resources. The collected information is updated in the Cloud Resource Information Repository (CRIP).

2.4 Virtual Machine Provisioner

It is mainly responsible for interacting with Cloud middleware to provision the virtual machine instances. It fetches the virtual machine request with the parameters of type of instances, ram capacity and number of instances to be created to run the job.

2.5 Virtual Machine Energy Monitor

It is running in the cloud resources and it collects the energy consumed by the virtual machine instances. The collected information is updated to the VM Energy Aggregator.

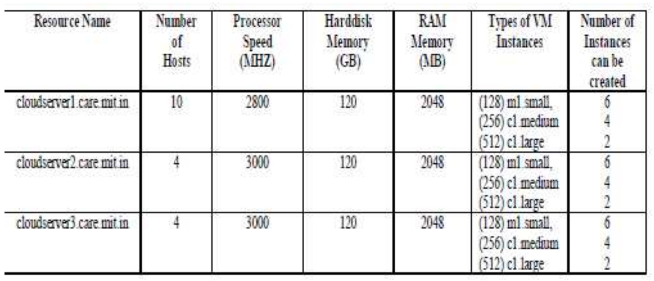
3. Implementation Details

In this paper we have simulated and compared the proposed PSO based energy aware resource allocation with DVFS and Round Robin. The simulation is carried out using the CloudSim [4] toolkit. The CloudSim source code is analyzed and incorporated with the major modifications in the classes DatacenterBroker.java, Host.java, Cloudlet.java and newly added PSOEARAllocationPolicy.java. The available resources in the cloud environment are represented as ‘A_{CR}’. Each cloud resource has ‘m_h’ number of hosts and every host is capable of hosting/creating ‘n_v’ number of virtual machine instances. The proposed system is accessed by ‘m_u’ number of users the users are arrived at a regular interval of ‘I’ in a Poisson distribution manner. Each user job request will require ‘N‘ number of nodes, ‘M’ amount of RAM memory, ‘P’ amount of processor speed. The sample HPC job request is shown in Table 1 and it is generated by doing the modifications in the Cloudlet.java class. We have generated the job requests for three types of HPC applications such as NAMD [5], Clustal [6] and FASTA [7]. The Host.java class is modified and the generated Eucalyptus based private cloud resources is shown in Table 2.

Table 1: Simulated HPC User Job Requests

User Name	**Job Type**	**Number Of Nodes**	**Processor Speed (MHZ)**	**RAM Memory (MB)**	**Disk Memory (GB)**
stselvi	NAMD	5	2200	512	10
preethi	CLUSTAL	10	2000	1024	20
Rohini	NAMD	5	2200	1024	10
kavitha	FASTA	5	2000	512	20

Table 2: Simulated Eucalyptus based Private Cloud Resources



Resource Name	Number of Hosts	Processor Speed (MHZ)	Harddisk Memory (GB)	RAM Memory (MB)	Types of VM Instances	Number of Instances can be created
cloudserver1.care.mit.in	10	2800	120	2048	(128) m1.small, (256) c1.medium (512) c1.large	6 4 2
cloudserver2.care.mit.in	4	3000	120	2048	(128) m1.small, (256) c1.medium (512) c1.large	6 4 2
cloudserver3.care.mit.in	4	3000	120	2048	(128) m1.small, (256) c1.medium (512) c1.large	6 4 2

4. Simulation Details and its Inferences

The experimentation is carried out by generating a Cloud Service Broker (CSB) with multiple Cloud Service Providers (CSPs). We have considered 5 CSPs each CSP maintains one datacenter. Each datacenter is generated with 1000, 2000, 3000, 1000, 2000 cloud hosts respectively. The cloud hosts has different capabilities in terms of number of processors, processor speed, ram speed, hard disk memory, bandwidth, latency, type of hypervisor and etc. The job request is generated randomly using the random access model that generates the job requests as Cloudlets in the range of 1000 to 10000 in the random fashion. The job parameters such as length of job (J_A), job arrival rate (A_A) and number of Job requests (N_J) also generated. The job requests are mapped with available cloud resources for creating virtual instances and running the applications. The experimental setup is shown in Figure 3. The simulation has been carried out for type of use cases (i) Use Case 1 - Resource Allocation within datacenter (ii) Use Case 2 - Resource Allocation across datacenters. The performance measures such as number of users completed within deadline, energy consumption of the datacenters are represented figuratively.

(i) Use Case 1 - Resource Allocation within datacenter – In this use case, the resource allocation policy finds out the suitable resources for every job requests that consumes less energy within the single datacenter. If the job requests could not able to satisfy within single datacenter the resource allocation policy sends the message to the broker COULD NOT be ABLE TO CREATE required VIRTUAL MACHINE WITHIN SINGLE DATACENTER”. The broker invokes the resource co-allocation policy to satisfy the job request could not be processed in the single data center.

(i) Use Case 1 - Resource Allocation across datacenters – In this use case, the resource allocation policy finds out the suitable resources for every job requests that consumes less energy across the datacenters. If the job requests could not able to satisfy across the datacenters the resource allocation policy sends the message to the broker "COULD NOT be ABLE TO CREATE ENOUGH VIRTUAL MACHINES ACROSS THE DATACENTERS". The broker rejects the request and notifies to the user.

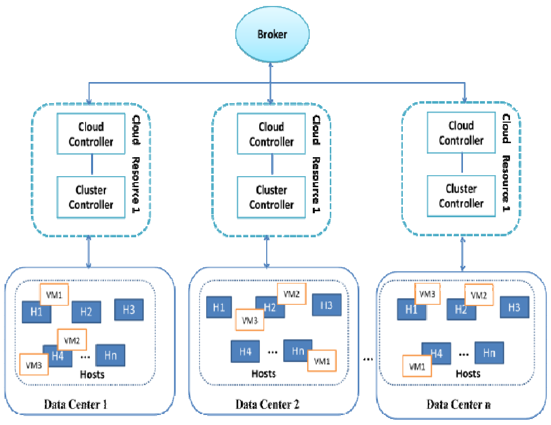


Figure 3: Experimental Setup

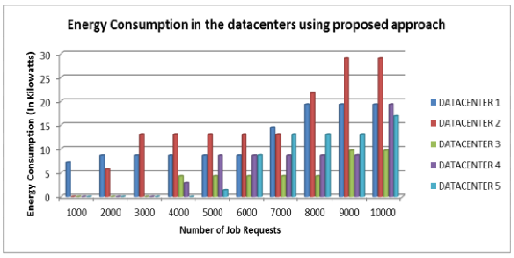


Figure 4: Energy Consumption using PSOEARA

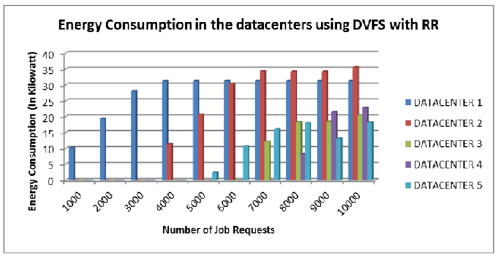


Figure 5: Energy Consumption using DVFS with RR

The job requests are generated in the order 1000 to 10000 cloudlets and the job rejection rate of PSOEARA is compared DVFS with RR. The proposed mechanism has the job rejection rate with an average of 10% and the RR has the job rejection rate of 35%.

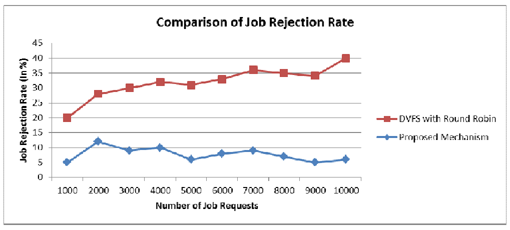


Figure 6: Comparison of Job Rejection Rate PSOEARA versus DVFS with RR

5. Related Works

Thamarai Selvi et. al [8] has proposed and implemented a Java based architectural framework to schedule and support the virtual resource management in the Grid environment. It handles the various scheduling scenarios of Physical, Coalloc, Virtual Cluster and etc. Eucalyptus [9] is the open source cloud middleware and it consists of cloud controller, cluster controller, node controller and storage controller. These components are arranged in a hierarchical fashion and eucalyptus has incorporated with Greedy, Round Robin, and Power Save scheduling algorithm. These three scheduling algorithms do not select the resource in a near optimal manner and also it does not have considered the priority. OpenNebula (2005) [10] is an open source cloud middleware that creates virtual machines in a physical cluster and its main focus is virtual resource management in the infrastructure. It has incorporated with rank based scheduling approach and does not consider the energy efficiency and deadline parameters and it is mainly working in the host level. Das et. al. [11] has built the commercialized computing system called Unity; the main aspects are application environment centric, computation of optimal configuration of resources in the datacenters, absence of the cost of components during the problem formulation.

Biao Song [12] has discussed the heuristic based task selection and allocation framework in cloud environment. They have classified the resource allocation problem into two things such as heavy workload and light workload. In the heavy workload scenario they have consider the Quality of Service (QoS) is their major focus and in the light workload scenario the resource utilization is their main focus. They maintained the threshold value based on that value they will allocate the tasks to the resources with an objective of increasing the resource utilization. But they have not discusses anything about the energy efficiency and deadline of the job requests. Hai Zhong et. al [13] proposed the optimized resource scheduling for open-source cloud systems using the Improved Genetic Algorithm (IGA). They have derived the fitness algorithm using the dividend policy mechanism. They have compared their proposed approach with First Fit and RR. They claimed that their proposed algorithm increases the utilization of the cloud resources and saves much energy. The major difference of our work from their work is they have not discussed anything HPC job requests, energy efficiency in detail and deadline of the job requests.

6. Conclusion and Future Work

The datacenters are consuming huge amount of electric power and emits large amount of carbon footprints that pollutes the environment. This paper mainly aimed to provide an efficient resource management mechanism in the cloud service broker. It handles the user job requests as HPC applications based on the user required parameters it selects the cloud resources in a near optimal manner using the heuristics based energy aware resource allocation mechanism. The proposed work minimizes the consumption of power and maximizes the revenue of the CRP's. The main contributions of the proposed work are summarized as follows: ability of handling the HPC job requests in Cloud Service Broker, matchmaking the user job requests and allocating the user job requests to the available cloud resources that consumes less energy in an optimal manner and completes the job within deadline. It increases the maximum number of jobs completed within the deadline and minimize the consumption of energy in the datacenters. These two factors influence to maximize the revenue of the cloud resource providers. The proposed work is simulated using the CloudSim toolkit and compared with the most well-known algorithm DVFS using Round Robin. The results are evident that proposed work minimizes the consumption of energy in the datacenters and maximizes the number of users completed within the deadline.

As a future work the proposed work to be tested in the Eucalyptus based real private cloud environment for HPC applications. And also, it can be extended for decentralized mode incorporated with load balancing mechanism that will enhance the scalability and utilization of cloud resources further.

ACKNOWLEDGMENT

The authors sincerely thank the Ministry of communication and Information Technology, Government of India, for financially supporting the Centre for Advanced Computing Research and Education of Anna University Chennai, India in this project

REFERENCES

[1] NIST, National Institute of Standards and Technology (2011), http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

[2] Particle Swarm Optimization (PSO), http://www.swarmintelligence.org/.

[3] Thamarai Selvi Somasundaram, Kannan Govindarajan, "Cloud Monitoring and Discovery Service (CMDS) for IaaS resources", has been accepted in ICoAC 2011.

[4] CLOUDSIM, http://www.cloudbus.org/cloudsim/.

[5] NAMD, http://www.ks.uiuc.edu/Research/namd/.

[6] CLUSTAL, http://www.clustal.org/.

[7] FASTA, http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml.

[8] Thamarai Selvi Somasundaram, Balachandar R. Amarnath, R. Kumar, P. Balakrishnan, K. Rajendar, R.Rajiv, G. Kannan, G. Rajesh Britto, E. Mahendran & B. Madusudhanan, "CARE Resource Broker: A framework for scheduling and supporting virtual resource management", Journal: Future Generation Computer System, Volume 26, Issue 3, March 2010, Pages 337-347, doi:10.1016/j.physletb.2003.10.071.

[9] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov ,"The Eucalyptus Open source Cloud-computing System", Proceedings of Cloud Computing and Its Applications, October 2008.

[10] OpenNebula (2011): The Open Source Toolkit for Cloud Computing. http://opennebula.org/start.

[11] R. Das, J. Kephart, I. Whalley and P. Vyas, "Towards Commercialization of Utility-based Resource Allocateion," in ICAC '06:IEEE International Conference on Autonomic Computing, 2006, pp.287-290.

[12] Biao Song, Mohammad Mehedi Hassan, Eui-nam Huh: A Novel Heuristic-Based Task Selection and Allocation Framework in Dynamic Collaborative Cloud Service Platform. CloudCom 2010: 360-367.

[13] Hai Zhong, Kun Tao, Xuejie Zhang, "An approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems," The Fifth Annual ChinaGrid Conference, 2010. DOI 10.1109/ChinaGrid.2010.37.