Energy Efficient Heuristic Resource Allocation for Cloud Computing

Dilip Kumar, and Bibhudatta Sahoo

Abstract—Minimizing the energy consumption in cloud computing environment is one of the key research issues. Power consumed by computing resources and storage in cloud can be optimized through energy aware resource allocation. As the resource utilization by the tasks are directly related to energy consumption, the task consolidation is being used to optimize the energy consumption. An energy efficient heuristic algorithm has been proposed and compared with three energy-aware task consolidation heuristics by varying number of tasks. The proposed task consolidation algorithm minimizes total energy consumed by the cloud computing system.

Keywords—Cloud Computing, task consolidation, energy aware, virtual machine, energy-efficient resource allocation, resource utilization.

I. INTRODUCTION

Cloud computing infrastructures are designed to support the accessibility and deployment of various service-oriented applications by the users[1][2]. Cloud computing services are made available through the server farms or data centers. To meet the growing demand for computations and large volume of data, the data centers host high performance servers and large high speed mass storage devices [3]. These resources are the major source of the power consumption in data center along with air conditioning and cooling equipment [4]. More over the energy consumption in cloud are proportional to the resource utilization and data centers are almost the world’s highest consumers of electricity [5]. Due to the high energy consumption by data centers, it requires efficient technology to design green data center [6]. Cloud data center, on the other hand, can reduce the energy consumed through server consolidation, whereby different workloads can share the same server using actualization and unused servers can be switched off.

Power management represents a collection of IT processes and supporting technologies geared toward optimizing data center performance against cost and structural constraints. This includes increasing the deployable number of servers per rack, when racks are subject to power or thermal limitations, and making power consumption more predictable and easier to plan for. Power management comes in two categories: static and dynamic. Static power management deals with fixed power caps to manage aggregate power, while policies under dynamic power management take advantage of additional degrees of freedom inherent in virtualized cloud data centers, as well as dynamic behaviors supported by advanced platform power management technologies[7].

Generally, clouds are deployed to customers giving them three levels of access: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Clouds use virtualization technology in distributed data centers to allocate resources to customers as they need them. The task originated by the customer can differ greatly from customer to customer. Entities in the Cloud are autonomous and self-interested; however, they are willing to share their resources and services to achieve their individual and collective goals. In such open environment, the scheduling decision is a challenge given the decentralized nature of the environment. Each entity has specific requirements and objectives that need to achieve.

In this paper, we propose a heuristic algorithm that could be applied to the centralized controller of a local cloud that is power aware. We capture the Cloud scheduling model based on the complete requirement of the environment. We further create a mapping between the Cloud resources and the combinatorial allocation problem and propose an adequate economic-based optimization model based on the characteristic and the structure of the Cloud.

Cloud computing is based on the concept of virtualization that encapsulates various services that can meet the user requirement in a cloud computing environment [8]. Virtual machines(VMs) are designed to run on a server to provide multiple OS environment in the support of various application. Virtual Machines(VMs) are used to meet the resource requirement and run time support for the applications. In particular executing an application on required resource can be made available through two step: creating instance of virtual machine as required by the application (VM provisioning) and scheduling the request to the physical resources other wise known as resource provisioning [4].

Server consolidation are allowing the multiple servers running on a single physical server simultaneously to minimize the energy consumed in a data center [9]. Running the multiple servers on a single physical server are realized through virtual machine concept. The task consolidation also know as server/workload consolidation problem [10]. Task consolidation problem addressed in this paper is to assign n task to a set of r resources in cloud computing environment. This energy efficient load management maintains the utilization of
all compute nodes and distributes virtual machines in a way that is power efficient. The goal of this algorithm is to maintain availability to compute nodes while reducing the total power consumed by the cloud. In this paper, we implement and evaluate three task consolidation algorithm based on greedy heuristic. These algorithms assigns task to the servers so as to minimize the total energy consumed by the system. Our proposed heuristic MaxMaxUtil allocate the task to the server that consumes minimum energy and shows performance improvement.

The remainder of this paper is organized as follows. The next section discusses related research outcomes on energy aware scheduling and resource allocation for cloud computing systems. In Section 3 we define the model of cloud computing system, task model and energy consumption of the system. Based on this system model, we have defined the problem to minimizing the energy in cloud computing environment. Section 4 discusses the heuristic algorithms used in this study with the illustration. Section 5 discusses our simulation set up and analyses our simulation results. Finally, conclusions and directions for future research are discussed in Section 6.

II. RELATED WORK

The energy efficiency of clouds has become one of most crucial research issues [10]. Advancements in hardware technologies [11], such as low power CPUs, solid state drives, and energy efficient computer monitors have helped relieve this energy issue to a certain degree. In the meantime, there also have been a considerable amount of research conducted using software approaches, such as scheduling and resource allocation [10] and task consolidation [12]. Current research in the area of cloud computing load balancing focuses on the availability of resources. The specific load balancing approach depends on the type of resource offered. Galloway et al. [13] has proposed a load balancing algorithm for infrastructure as service (IaaS) for cloud infrastructure implemented on cluster. There are many proposed systems utilizing market-based resource management for various computing areas [14, 5] Kusic et al. [15] have stated the problem of continuous consolidation as a sequential optimization and addressed it using Limited Look ahead Control (LLC). The proposed model requires simulation-based learning for the application specific adjustments. Due to complexity of the model the optimization controller execution time reaches 30 minutes even for a small number of nodes (e.g. 15), that is not suitable for large-scale real-world systems. Srikantaiah et al. [12] have studied the problem of requests scheduling for multi-tiered web-applications in virtualized heterogeneous systems in order to minimize energy consumption, while meeting performance requirements. To handle the optimization over multiple resources, the authors have proposed a heuristic for multidimensional bin packing problem as an algorithm for workload consolidation. Song et al. [16] have proposed resource allocation to applications according to their priorities in multi-application virtualized cluster. The approach requires machine-learning to obtain utility functions for the applications and defined application priorities. Cardosa et al. [17] have explored the problem of power efficient allocation of VMs in virtualized heterogeneous computing environments. They have leveraged min, max and shares parameters of VMM that represent minimum, maximum and proportion of CPU allocated to VMs sharing the same resource. The approach suits only enterprise environments or private Clouds. Verma et al. [18] have formulated the problem of dynamic placement of applications in virtualized heterogeneous systems as continuous optimization: at each time frame the placement of VMs is optimized to minimize power consumption and maximize performance. The authors have applied a heuristic for bin packing problem with variable bin sizes and costs. The authors have introduced the notion of cost of VM live migration, but the information about the cost calculation is not provided. Calheiros et al. [19] have investigated the problem of mapping VMs on physical nodes optimizing network communication between VMs, however, the problem has not been explored in the context of energy consumption minimization. The studies show that software-driven thermal management and temperature aware workload placement bring additional energy savings.

III. SYSTEM MODEL

The concept of cloud computing has been emerged from the concept of heterogeneous distributed computing, grid computing, utility computing and autonomic computing [20, 21]. Figure 1 depicts the system model of cloud computing system, which has been referred in this paper. The cloud computing system is consists of fully interconnected set of m resources denoted as R. These resources are to be allocated on demand to run applications time to time. We have assumed the centralized cloud is hosted on a data center that is composed of large number of heterogeneous servers. Each of server may be assigned to perform different or similar functions.
These Virtualization technologies allow the creation of multiple virtual hosts on any of the available servers. There for a task can be flexibly assigned to any server. Servers can be modeled as a system that consumes energy in idle state to perform maintenance functions and to have all the subsystems ready while it waits for task to arrive. Once a task arrives, a server processes the task and it may spend an additional amount of energy, which depends on the number of resources demanded by the task, it is represented as resource utilization in work load model.

Although a cloud can span across multiple geographical locations (i.e., distributed), the cloud model in our study is assumed to be confined to a particular physical location. We assume that resources are homogeneous in terms of their computing capability and capacity; this can be justified by using virtualization technologies [10]. It is also assumed that a message can be transmitted from one resource to another while a task is being executed on the recipient resource, which is possible in many systems [10]. The maximum and minimum energy consumption of the server in cloud computing system are denoted as pick load state and idle state.

A. Energy Model for Centralized Cloud Infrastructure

A cloud computing infrastructure can be model as H is a set of physical Servers/host H1, H2, H3, ..., Hn. The resources of cloud infrastructure can be used by the Virtualization technology, which allows one to create several Virtual Machines (VMs) on a physical server/host and therefore, reduces amount of hardware in use and improves the utilization of resources. So we have assumed the computing resource/node in cloud model as virtual machine. A computing resources R is a set of virtual machines R1, R2, R3, ... Rm. The energy consumed by the resource Rj for executing a task or services runs on the resource. The energy consumed by a resource can proportional to the processor associated with the resource [10]. Let the set of task can be represented as T= {t1, t2, t3, ..., tk}. Each task tj has an expected time to compute on resource Rj and dent ed as tj. A task on cloud computing environment can be characterized by inconsistent expected time to compute (ETC) model as suggested by Ali et al in [10].

Let u(i,j) be the resource usage by the task tj when executed on Rj. Then the utilization matrix of the resource Rj at a given time τ, denoted as Uj, and defined as

\[ U_j(\tau) = \sum_{i=1}^{k} u_{i,j} \]  

Where k is the number of tasks running on resource Rj at time τ.

The Energy Consumption Ei of a resource Rj at a time τ is defined as

\[ E_i(\tau) = (p_{\text{max}} - p_{\text{min}}) \times U_i(\tau) + p_{\text{min}} \]  

Where \( p_{\text{max}} \) is the power consumption at the peak load and \( p_{\text{min}} \) is the minimum power consumption in the inactive mode.

B. Task Model

A task represents a user’s computing or service request. A task may range from simple activities such as document viewing, to complex activities involving multiple steps, such as 3D rendering. A task is an independent scheduling entity and its execution cannot be preempted. The tasks can be executed in any node. Formally, each arriving task tj is associate with a task ID, arrival time and expected time to compute on different computing node.

We have taken following assumption for creating work load:
- Task arrival time is considered.
- Tasks arrival rate at any particular time is Poisson distribution.
- Task processing time on resources are exponentially distribution with mean \( \mu = 10 \).
- Resource utilization by task is normally distribution between 10% and 100%.

Example of task model is shown in figure 2. The first column contains the task ID, second column represent the task arrival time, third column represent the resource utilization by task tj and the rest columns are characterized by Expected Time to Compute (ETC) on resource Rj. For example, task ID 16 has arrived at 3 unit of time, resource utilized by this task is 90% and expected time to compute on resource is 12 units of time.

C. Task Execution

- The resource allocated to a particular task must sufficiently provide the resource usage for that task.
- If resources are not sufficiently provide the resource usage for a particular task then task putted in waiting queue.

D. Problem statement

The resource utilization is directly proportional to the energy consumed by physical resources. The process of assigning a set of n tasks (service requests or simply services) to a set of r cloud resources are aiming to maximize resource utilization, ultimately to minimize energy consumption.

The total energy consumed by the cloud infrastructure at instant of time, τ is denoted as \( E(\tau) \) and defined as follows.

\[ E(\tau) = \sum_{i=1}^{m} E_i(\tau) \]  

Where, m is the total number of resources in cloud infrastructure.

Figure 2: Example of Tasks Table

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Arrival Time</th>
<th>Resource Utilization(%)</th>
<th>Task Execution Time on VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>58</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>62</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>31</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>55</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>92</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>29</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>57</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>31</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>54</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>60</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>54</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>31</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>48</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>99</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>13</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>90</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>32</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>22</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>18</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>23</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10</td>
</tr>
</tbody>
</table>

Figure 2: Example of Tasks Table
\[ E = \int_{\tau_0}^{\tau} E(\tau) d\tau \]  
(4)

or,

\[ E = \Delta \tau \sum_{\tau} E(\tau) \]  
(5)

The objective of our research work is to minimize \( E \) on cloud computing infrastructure by efficient allocation set of tasks \( T \).

IV. TASK CONSOLIDATION HEURISTIC ALGORITHMS

Heuristic and meta-heuristic algorithms are the effective technology for scheduling in HDCS due to their ability to deliver high quality solutions in reasonable time. In this section, we present the greedy heuristic algorithms for task allocation in a data center. The general form of task allocation algorithm for the resource utilization of cloud server resources is presented in Algorithm-1. This algorithm allocate task to the physical resource and maintain the utilization matrix. The Algorithm-1 operates by finding the task which uses maximum resource from the currently available task in task queue. The function TaskChoosingPolicy() returns the task from the task queue tempQ and the function ResourceChoosingPolicy() returns the resource for the task \( t_j \) for which maximum threshold value less than or equal to 100%. If no such fit found it returns null. If resource \( R_i \) is found such that utilization is maximum for task \( t_j \) and utilization is not exceeding 100%. After allocating tasks \( t_i \) to resource \( R_i \), the task is removed from the task queue mainQ and temporary queue tempQ. If no suitable fit is found then the task \( t_j \) will be removed from temporary queue but not from main queue, this process proceeds to a new iteration. The heuristic algorithms are simple to realize with very little computational cost in comparison to the effort by resource allocation algorithm. The three different heuristic algorithm used in this paper are described as follows. The algorithm FCFS Max Util has been adapted from heuristic algorithm presented by Lee and Zomaya [10].

A. FCFS to Random Utilized (FCFSRandomUtil)

The first heuristic algorithm is known as FCFSRandomUtil. This algorithm selects the task in first come first serve (FCFS) basis and the resource is selected in random (using uniform distribution) among the available VMs. The task is assigned to the Virtual Machine \( R_i \) if \( R_i \) utilization is not exceeding threshold value 100% including the current task. Iteration continues till all tasks are allocated to VMs. The example in Figure 3 shows time required for the allocation of 20 tasks to 10 VMs.

B. FCFS to Maximum Utilized (FCFSMaxUtil)

The task selection process of the FCFSMaxUtil algorithm also follows FCFS principle. To allocate the selected task, the VM with maximum utilization is selected among the available VMs. The utilization of selected VM is computed by adding the assigned task. The task is assigned to the Virtual Machine \( R_i \), if \( R_i \) utilization is not exceeding 100% including the current task. Figure-4 the outcome of MaxUtil algorithm for 20 tasks to 10 VM.

C. Maximum to Maximum Utilized (MaxMaxUtil)

The pseudo-code for the proposed MaxMaxUtil algorithm for the Maximum utilization of cloud server resources is presented in Algorithm-2. This algorithm allocate task (which required the maximum resource utilization) to the currently maximum utilizing resources. First the algorithm operated on task queue, which is the resulted on arrival of task till the time

\begin{algorithm}
\caption{General Task Allocation Algorithm}
\begin{algorithmic}[1]
\State \textbf{Input:} Task Matrix
\State \textbf{Output:} Utilization Matrix
\State 1: Initialize \( \tau \)
\State 2: Initialize Utilization Matrix, \( U' \leftarrow \emptyset \)
\State 3: \( R' \leftarrow \emptyset \)
\State 4: while mainQ \neq \emptyset do
\State 5: \hspace{1em} \textbf{if} \ i \neq \text{Null} \hspace{1em} \textbf{then}
\State 6: \hspace{2em} Assign task \( t_j \) to \( R_i \)
\State 7: \hspace{2em} Update Utilization Matrix \( U(\tau, i) \)
\State 8: \hspace{2em} Remove task \( t_j \) from mainQ and tempQ.
\State 9: \hspace{1em} \textbf{else}
\State 10: \hspace{2em} Remove task \( t_j \) from tempQ.
\State 11: \hspace{1em} \textbf{end if}
\State 12: \hspace{1em} Increment \( \tau \).
\State 13: \textbf{end while}
\State 14: \textbf{return} \( U \).
\State 15: \textbf{End Algorithm}
\end{algorithmic}
\end{algorithm}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{algorithm1.png}
\caption{Algorithm 1: General Task Allocation Algorithm}
\end{figure}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
Time/VM & M1 & M2 & M3 & M4 & M5 & M6 & M7 & M8 & M9 & M10 \\
\hline
\hline
\end{tabular}
\caption{Table for 20 tasks}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{algorithm2.png}
\caption{Example of FCFS to Random Utilization Tasks allocation}
\end{figure}
of selection. The task is selected from the task queue having maximum resource utilization.

<table>
<thead>
<tr>
<th>Time/VM</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
<th>M10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,3]</td>
<td>[2]</td>
<td>[4]</td>
<td>[5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[1,3]</td>
<td>[2]</td>
<td>[4]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>[1,3,12]</td>
<td>[2]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>4</td>
<td>[1,3,12]</td>
<td>[2]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>5</td>
<td>[1,3,12]</td>
<td>[2]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>6</td>
<td>[1,3,15]</td>
<td>[16]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>7</td>
<td>[1,3,15]</td>
<td>[16]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>8</td>
<td>[1,3,15]</td>
<td>[16]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>9</td>
<td>[1,3,15]</td>
<td>[16]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>10</td>
<td>[1,3,15]</td>
<td>[16]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>11</td>
<td>[1,3,15]</td>
<td>[16]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>12</td>
<td>[1,3,15]</td>
<td>[16]</td>
<td>[4,12]</td>
<td>[5]</td>
<td>[6,8]</td>
<td>[7,13]</td>
<td>[9]</td>
<td>[10]</td>
<td>[11]</td>
<td>[14]</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>[16]</td>
<td>[12]</td>
<td>[17]</td>
<td>[18]</td>
<td>[19]</td>
<td>-</td>
<td>[20]</td>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>[16]</td>
<td>[12]</td>
<td>[17]</td>
<td>[18]</td>
<td>[19]</td>
<td>-</td>
<td>[20]</td>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>[16]</td>
<td>[12]</td>
<td>[17]</td>
<td>[18]</td>
<td>[19]</td>
<td>-</td>
<td>[20]</td>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>[16]</td>
<td>[12]</td>
<td>[17]</td>
<td>[18]</td>
<td>[19]</td>
<td>-</td>
<td>[20]</td>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>-</td>
<td>[16]</td>
<td>[12]</td>
<td>[17]</td>
<td>[18]</td>
<td>[19]</td>
<td>-</td>
<td>[20]</td>
<td>[11]</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>[17]</td>
<td>[18]</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>[11]</td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>[17]</td>
<td>[18]</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>[11]</td>
</tr>
</tbody>
</table>

The function MaximumResourceUtilizationTask(tempQ) return the maximum resource utilizing task from the task queue tempQ and the function MaximumUtilizingResource(U, τ, j) return the resource which has maximum utilization of resources for task tj, but less than or equal to maximum threshold value 100% if no such fit found it return 0 value. If resource Rj is found such that utilization is maximum for task tj and utilization is not exceeding 100%. After allocating task tj to resource Rj, the task is removed from the main queue mainQ and temporary queue tempQ. If no suitable fit is found then the task tj will be removed from temporary queue but not from main queue, the iterative process continues till the successful allocation of all tasks to VMs.

**Algorithm 2: MaxMaxUtil Algorithm**

**Input:** Task Matrix  
**Output:** Utilization Matrix

1: Initialize τ  
2: Initialize Utilization Matrix, U* ← Ø  
3: R* ← Ø  
4: while mainQ ≠ Ø do  
5:   tempQ ← All jobs from mainQ where arrival time ≤ τ.  
6:   while tempQ ≠ Ø do  
7:     j ← MaximumResourceUtilizationTask(tempQ)  
8:     i ← MaximumUtilizedResource(U, τ, j)  
9:     if i ≠ Null then  
10:        Assign task tj to Ri  
11:        Update Utilization Matrix U(τ, i).  
12:        Remove task tj from mainQ and tempQ.  
13:     else  
14:        Remove task tj from tempQ.  
15:     end if  
16:   end while  
17: Increment τ.  
18: end while  
19: return U.  
20: End Algorithm

Example of Maximum to Maximum Utilized allocations and utilization are shown in figure-5 and figure-6 for allocation of 20 tasks to 10 VMs.

**V. SIMULATION RESULTS**

We have simulated the performance of three task consolidation heuristic with 5000 task. The tasks are generated for different set of VMs using inconsistent ETC generation.
The algorithm suggested by Ali et al. [22]. The experiment have been conducted using our simulator designed using Matlab 2012 for 5000 tasks. The tasks are arriving with a rate $\lambda$ to the central server queue having infinite queue length. We have considered task arrival interval is 1 and arrival rate to be 60 for our experiments. The performance of task consolidation algorithms are presented for 20, 40 and 60 virtual machines in Figures 7, 8 and 9 respectively. The three heuristic task consolidation algorithms are FCFSRandUtil, FCFSMaxUtil, and MaxMaxUtil. It is observed that the utilization of resources are better when resources are allocated using MaxMaxUtil heuristic. It is also observed that the performance of MaxMaxUtil algorithm is affected by the rate of heterogeneity of the task and computing node, as well as consistency of the tasks.

![Figure 7: Utilization Comparison for tasks on 20 VMs](image)

![Figure 8: Utilization Comparison for tasks on 40 VMs](image)

![Figure 9: Utilization Comparison for tasks on 60 VMs](image)

![Figure 10: Energy Consumption for 5000 tasks on 60 VMs](image)

![Figure 11: Energy Saving compared to FCFSRandomUtil for 5000 tasks on 60 VMs](image)

The Figure-10 presents the experiment result obtained for the total Energy consumption on 60 VMs by varying task size from 1000 to 5000. This experiment also considers the arrival rate of task to be 60 and inter arrival time to 1. The maximum and minimum energy consumption of the server in cloud computing system is taken as 30 and 20 units respectively. Our simulator shows that energy consumption and energy saving for tasks on 60 VMs in Figure-10 and Figure-11 respectively. The MaxMaxUtil heuristic saves up to 9.5% more energy in comparison to the FCFSRandomUtil as shown in Figure-11.

VI. CONCLUSION

Simulation experiments were conducted to examine the performance of simple heuristic based task consolidation algorithms to optimize the energy consumption in cloud computing system. An average case analysis is presented for
three different task consolidation algorithm with inconsistent ETC matrix. Simulation results prove the MaxMaxUtil heuristic algorithm is preferred over others.

REFERENCES


