Energy-Efficient Server-Consolidation Based Resource Allocation in Cloud

P. Seenuvasan
Department of Information Technology
University College of Engineering Villupuram
Villupuram, India

R. Geetharamani
Department of Information Science and Technology
Anna University Chennai, India

P. Varalakshmi
Department of Computer Science and Engineering
Anna University, Chennai, India

Renuka Ramachadran
Department of Information Technology
MIT Campus, Anna University, Chennai, India

Abstract

In a cloud computing environment, the load is shared between the servers dynamically. The nodes may be upgraded, replaced and added in data center. This results in unequal distribution of load among the servers. Hence an effective load balancing algorithm is required. Due to the high demand for energy, conservation of energy in datacenters is also a major concern. In addition to the operating cost of the data centers, the cooling systems also demand a higher portion of energy. We propose energy aware server-consolidation based method to schedule and process the submitted user jobs. This method improves the resource utilization
while conserving the energy. This paper aims at achieving server consolidation while allocating the resources to the jobs. It is done by minimizing the number of active servers and distributing the load equally among them. The proposed algorithm is compared against the existing algorithms and results show that given jobs are executed in minimum number of active servers and energy is conserved efficiently.

**Keywords** - Server Consolidation, Idle Servers, Green Computing, CloudSim

1. **Introduction**

In Cloud computing, services are used from the resources which are spread across the globe, rather than remote servers or local machines. Generally the cloud consists of a group of distributed servers known as masters, providing services and resources to different clients in a network with scalability and reliability. Energy consumption represents a significant cost in data center operation. A major portion of energy is also utilized in powering the cooling systems. A large fraction of the energy is also used to power idle servers when the workload is low. Thus determining the number of active servers for servicing the clients is a major challenge in data centers. Dynamic consolidation of virtual machines (VMs) using live migration and switching idle nodes to the sleep mode allow Cloud providers to optimize resource usage and reduce energy consumption. On the other hand, users require as many servers as possible in order to minimize the make span and maximize the throughput. The existing or proposed approaches to achieve energy efficiency can be broadly categorized into scaling, scheduling, and consolidation approaches.

The scaling approaches primarily target the processor subsystem of a server and aim to adapt its operational voltage and frequency according to the task arrival rate at the processor. Scheduling approaches can be low-level or high-level. The low-level approaches refer to energy-aware schedulers within the operating system whereas the high-level approaches refer to load-balancing (or load management) approaches at the application layer. Energy-aware schedulers aim to schedule tasks (jobs) in a way hardware resources are efficiently utilized and unnecessary contention between jobs for hardware resources (such as memory or CPU) are avoided. The load-balancing solutions at the application layer are relevant to a cluster environment (such as a data center) where multiple servers are used to handle a large amount of user requests simultaneously. Hence, an energy-efficient load-balancer aims at selecting \( m \) out of \( n \) servers, \( m \leq n \), which can handle the overall workload of the entire cluster without violating a Service-Level Agreement (SLA). The last category of approaches, consolidation, particularly concerns virtualized environments. Similar to the load balancing approaches, the aim is to balance the demand for computing resources with the supply of resources, but this is mainly achieved through a runtime migration of virtual machines. Then, the idle or the underutilized servers are switched off and the overloaded servers are relieved of a portion of their load.
The growing cost of electrical energy required to power the data centers and servers is a major concern. In addition to the operational centers, considerable amount of energy is also used in cooling systems. Dynamic consolidation of virtual machines (VMs) using live migration and switching idle nodes to the sleep mode allow Cloud providers to optimize resource usage and reduce energy consumption. On the other hand, users require as many servers as possible in order to reduce the execution time of all the jobs. Thus, a new energy based server consolidated load balancing algorithm is proposed which minimizes the energy consumption and maintains the throughput to an optimum level. This algorithm aims to minimize the number of active servers and distribute the load equally among them. Our major concern is to minimize the energy consumption in the servers.

The rest of the paper is organized as follows. Section II presents an overview of the works related to our work and a comparison of those with the proposed system. Section III describes the overall architecture of the system. The result analysis and performance evaluation which includes energy consumption of the submitted jobs and the number of tasks executed in a particular time are discussed in Section IV. Section V concise the conclusion and the future enhancements in this work.

2. Related Work

In a distributed file system, the file chunks are not distributed equally among the nodes. A load-balancing algorithm is proposed in [1], in which each node has global knowledge regarding the system, which results in low movement cost and fast convergence. It is then extended for the situation that the global knowledge is not available to each node without degrading its performance. However it is a challenge for each node to have such global knowledge in a large-scale and dynamic computing environment. It also introduced a load balancing algorithm without having global knowledge.

In [3], a load balancing model is proposed which divides the public cloud into several cloud partitions. These partitions simplify the problem of load balancing when the environment is complex. Main Controller is responsible for choosing the suitable partition and the balancer is responsible for choosing the best strategy to be adopted. The load balancing strategy depends on the cloud partitioning concept. As soon as the cloud partitions are created, the process of load balancing starts. The load status of each and every partition is monitored. If it is normal, the partitioning is done locally. Otherwise the job should be transferred to another partition. A load balancing algorithm was also proposed in [6] which resulted in 90% system utilizations and movement of only 8% of the incoming load.

For dynamic resource allocation, resources were classified into three categories-hot, warm and cold in [2]. The resources in hot spots are arranged in descending order and the hottest resource is selected first to mitigate the load. As soon as all the hot spots are mitigated, the load is balanced among the warm spots.
If sufficient warm spots are available to accommodate more load, the load in the cold spots are migrated to warm spots and the corresponding servers are switched off thus contributing to green computing.

In [5], the power consumption of presently available Internet servers and data centers is not proportional to the work they accomplish. This problem is addressed in a number of ways, for example, by using dynamic voltage and frequency scaling (DVFS), selectively switching off idle or underutilized servers, and utilizing energy-aware task scheduling. It provides a comprehensive survey of existing or proposed approaches to estimate the power consumption of single-core as well as multi-core processors, virtual machines, and an entire server.

In [4], a very general model is proposed and proved that the optimal offline algorithm for dynamic right-sizing has a simple structure, and this structure is exploited to develop a new “lazy” online algorithm, which is proven to be competitive. He validated the algorithm using traces from two real data-center workloads and showed that significant cost savings are possible. Additionally, this new algorithm is contrasted with the more traditional approach of receding horizon control. The algorithm is motivated by the structure of the optimal offline solution and guarantees cost no larger than 3 times the optimal cost, under the following settings—arbitrary workloads, general delay cost, and general energy cost models provided that they result in a convex operating cost.

3. Proposed System

The overall architecture of the proposed system is shown in Fig.1. On the client side, the clients send the job request to the Cloud Providers. These jobs are sent to the job pool present in the cloud provider site. From the job pool, the jobs are classified into data intensive jobs and computation intensive jobs based on the requirements of processor and memory. These nodes are initially clustered based on the type of the Processor such as Dual Core, 4Core, 6Core etc. Nodes belonging to every cluster are further clustered based on their Memory size. Max-heap is constructed for the set of clusters of resources as nodes. The cluster node which has the highest rank will be in the root initially. Clustering the jobs and resources based on the above parameters will improvise the time taken for resource scheduling. After clustering the resources, the jobs are directed to the cluster with highest rank. The rank is calculated based on the processor and memory of the virtual machines. After the resource allocation, the resources are monitored. The jobs are allocated to a node until it reaches its hot threshold. After that, the next node in the cluster is selected for allocation. If all the nodes in the cluster have reached their hot threshold, the clusters are heapified and the cluster with next highest rank is then selected for allocation.

The workflow of the proposed system composed of clustering of jobs and resources, rank computation of clusters, resource allocation, resource monitoring, re-heapification and hot spot mitigation are discussed below.
A. Clustering of Jobs and Resources

All the jobs submitted to the cloud have a unique identifier. These jobs submitted to the data center are initially classified based on the type as data intensive and computational intensive jobs. These nodes are initially clustered based on the type of the Processor such as Dual Core, i3, i5 etc. Nodes belonging to every cluster are further clustered based on their memory. Clustering the jobs and resources based on the above parameters will improve the time taken for resource scheduling.

B. Rank Computation of Clusters

The rank is computed for each and every cluster of servers. For data intensive heap of clusters, the rank of individual node is (\( \text{Rank}_n \)) computed using (1) as

\[
\text{Rank}_n = np \times 0.1 + ms \times 0.9
\]

Here, \( np \) is the number of processors, and \( ms \) is the memory size of node in a cluster. The rank of the data intensive cluster is computed using (2) as the average of the ranks of all the nodes present in cluster.
$Rank_{dc} = \frac{\sum_{i=1}^{n} Rank_i}{n}$  \hspace{1cm} (2)

Here, n is the number of nodes present in cluster, $Rank_i$ is the rank of the $i^{th}$ node in cluster. And $Rank_{dc}$ is the rank of the data intensive cluster.

For computationally intensive heap of clusters, the rank of individual node ($Rank_n$) is calculated using (3) as

$Rank_n = np \times 0.9 + ms \times 0.1$  \hspace{1cm} (3)

Rank of computationally intensive cluster is computed using (4) as the average of the ranks of all the nodes present in cluster.

$Rank_{cc} = \frac{\sum_{i=1}^{n} Rank_i}{n}$  \hspace{1cm} (4)

Here $Rank_{cc}$ is the rank of computationally intensive cluster.

C. **Heapify the Cluster of Resources**

Based on the rank of every cluster, a max-heap is constructed. Hence the cluster with the highest rank is at the root. The clusters with the next rank are arranged in the next level. Finally the clusters with minimum rank are at the leaf nodes.

D. **Resource Allocation**

When the jobs arrive, they are directed to the respected root cluster of the heap (data intensive or computationally intensive heap). Since the nodes in the root cluster have higher rank, they have high end processors and high capacity memory. The jobs will be allocated to the cluster node in the corresponding heap until it reaches its hot spot. 80% of the capacity of the server is allocated means it will be considered as hot-spotted node in our implementation.

Further jobs will be allocated to the next node present in the cluster and continued in the same way until all the jobs will be allocated or all the nodes in the cluster will be exhausted. Based on the remaining capacity of the node, the rank of the individual cluster node will be reduced. So the overall rank of the entire cluster node will be decreased when all the nodes in cluster is exhausted. Then heap is reconstructed with next highly ranked cluster as root node of heap. Thus the jobs can be executed with appropriate clusters without violating the SLA in an efficient manner with less power consumption because we are going to allocate less number of high capacity nodes.

E. **Resource Monitoring**

After allocating the resources to jobs, the allocated capacity of the node is deducted from its original capacity. This process continues until the node reaches its cold spot threshold. 20% of the capacity of the server is considered as its cold
spot threshold. Then live migration is started from cold spotted node for server consolidation purposes so that reduction in power consumption is achieved. Once the original rank of the cluster is regained, again heap is reconstructed and this cluster node is again placed at the root.

F. Re-Heapification

When all the nodes in a cluster have reached its hot spot threshold and their rank got reduced due their lower remaining capacity. The clusters are heapified again according to which the cluster with the next highest rank comes to the root position and the hot spotted cluster node moves to the leaf nodes. When the execution continues in the leaf cluster with hot spotted nodes, and as soon as the jobs will be completed and remaining capacity of the node will be increased. The node’s rank also got increased, and continued in the same manner for all the other nodes in the cluster so that the overall cluster rank will be increased. Then the heap is reconstructed.

4. Implementation and Results

The proposed technique is implemented in CloudSim environment in order to simulate the huge number of cloudlets (user jobs), VMs, physical hosts and the data centers. Java has been used as the programming language and used the IDE Net Beans 7.1 to solve our purpose.

The jobs submitted by the user to the cloud are known as cloudlets. The Cloudlets are specified by id, the number of processors required and the required memory capacity. The processor requirements of cloudlets considered in our implementation are Dual Core, 4 Core, 6 Core. The memory requirements of the cloudlets are 2GB, 4GB and 6GB. Table I shows the RAM and processor specification of the virtual machines taken for consideration in CloudSim. A set of data intensive jobs is taken for simulation. On the server side, the virtual machines handle the requests. They are specified by vm id, number of CPUs, speed of the processor, image size, memory, network bandwidth, energy consumption of the node based on the processor and virtual machine monitor. The parameters considered for simulation are processor and memory. The processor varieties of VMs considered are Dual Core, 4 Core, 6 Core. The memory specifications of VMs considered are 2GB, 4GB and 6GB.

These VMs are allotted to the physical machines based on the processor and memory present in the physical machine. The physical hosts are specified by id, number of CPUs, speed of the processor, memory, storage and bandwidth.

Table I. CLUSTER OF VMS

<table>
<thead>
<tr>
<th>S.NO</th>
<th>CLUSTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DUAL CORE, 8 GB</td>
</tr>
<tr>
<td>2</td>
<td>DUAL CORE, 16 GB</td>
</tr>
</tbody>
</table>
The physical hosts are placed in large datacenters. Datacenters are specified by system architecture, operating system, cost of usage of processor, cost of usage memory, the cost of usage of bandwidth.

By our proposed method of energy based load balanced resource allocation, the energy consumption is considerably reduced compared to the existing load balanced resource allocation in energy aware task scheduling algorithm as shown in Fig.2. Because the jobs are allocated to the less number of appropriate active servers in cluster to improve the conservation of energy, the number of active servers is also less as compared to the number of active servers in existing system (energy aware task scheduling algorithm [5]) as shown as Fig.3. Though the energy efficiency is achieved by our proposed idea and the number of idle servers also increases, the throughput decreases as compared to the existing system (energy aware task scheduling algorithm [5]) as shown in Fig.4.
5. Conclusion and Future Work

Server consolidation in cloud has been achieved by minimizing the energy consumption in servers and maximizing the utilization of resources. While allocating the resources, the load is also balanced among the active servers by considering the hot and warm thresholds of every server. These thresholds vary according to the capacity of servers and based on threshold values, the cloudlets will be allocated to the clusters. The algorithm is run for various numbers of cloudlets. A graph is drawn to compare the consumption of energy in the servers in the existing and the proposed work. Results show that the energy consumption is reduced by the proposed idea. Our algorithm is considered for the servers in a single datacenter and energy conservation is achieved successfully. In future this work can be extended for the servers in different datacenters across many locations. Those modifications have to be performed for the successful extension of our system.

References


Received: December 15, 2014; Published: April 22, 2015