DYNAMIC RESOURCE ALLOCATION FOR GREEN CLOUDS

BY

M.S. MUMTAJ ZAREENA * M. MAHIL ** N. RUPAVATHY ***

* P.G. Scholar, Department of Computer Science and Engineering, Govt College of Engineering, Tirunelveli, India.
** Assistant Professor, Department of Computer Science and Engineering, Govt.College of Engineering, Tirunelveli, India.
*** P.G. Scholar, Department of Computer Science and Engineering, Govt.College of Engineering, Tirunelveli, India.

ABSTRACT

Cloud computing is an on demand service as it offers dynamic, flexible and efficient resource allocation for reliable and guaranteed services in 'pay-as-you-use' manner to the customers. Such a process of allocation and de-allocation of resources is the key to accommodate unpredictable demands. However, despite the recent growth of the Cloud Computing market, several problems with the process of resource allocation remain unaddressed. A system that uses virtualization technology and skewness algorithm to allocate data center resources dynamically based on application demands and support green computing by optimizing the number of servers in use had been presented. The result reveals the achievement of both overload avoidance and green computing for systems with multi resource constraints.

Keywords: Load Prediction, Virtualization, Resource Allocation, Green Computing.

INTRODUCTION

The live migration of virtual machines can improve global system utilization by load balancing across physical machines and can improve system serviceability and availability by moving applications off machines that need servicing or upgrades. Load balancing facilitates utilization of resource by providing a throughput with minimum response time by sharing the equal load between servers. To achieve load balancing and resource utilization, there are few algorithms used. Best example for load balancing is online shopping cart. Without load balancing, users could experience delays while ordering, transactions and buying.

Load balancing solutions usually apply redundant servers which help a better distribution of the communication traffic so that the online purchasing will be made easy. It also makes sure that every computing resource is distributed efficiently and reasonably. When one or more service fails, load balancing helps in provisioning and de-provisioning of instances of applications without fail. The goal of load balancing is to progress the performance by balancing the load among these various resources (network links, central processing units, disk drives.) to achieve optimal resource utilization, maximum throughput, maximum response time, and avoiding overload.

As an important cornerstone for clouds, virtualization plays a vital role in building this emerging infrastructure. Virtual machines (VMs) with a variety of workloads may run simultaneously on a physical machine in the cloud platform. Virtual Machine Monitors (VMMs) like Xen provide a mechanism for mapping virtual machines (VMs) to physical resources. This mapping is largely hidden from the cloud users. VM live migration technology makes it possible to change the mapping between VMs and PMs (physical machines) while applications are running. However, a policy issue remains as how to decide the mapping adaptively so that the resource demands of VMs are met while the number of PMs used is minimized.

A major challenge in resource provisioning technique is to determine the right amount of resources required for the execution of work in order to minimize the financial cost from the perspective of users and to maximize the resource utilization from the perspective of service providers. So, Cloud computing is one of the preferred options in today's enterprise. Resource provisioning means the selection, deployment, and run-time management of software(e.g., database management servers, load balancers) and hardware resources (e.g., CPU, storage and network) for ensuring guaranteed

performance for the applications. This resource provisioning takes Service Level Agreement (SLA) into consideration for providing service to the cloud users. This is an initial agreement between the cloud users and cloud service providers which ensures Quality of Service (QoS) parameters like performance, availability, reliability, response time etc.

Today's low power mechanisms assume that – like lighting – the absence of a user is a sufficient condition for curtailing operation. While this is largely true for disconnected laptop computers, it is not compatible with how users and programs expect their connected desktops to function. The success of the Internet in providing global connectivity and hosting a broad array of services has implicitly engendered an "always on" mode of computation. This work proposes an alternative energy-conservation approach called Green computing, that is attuned to server utilization patterns. In this approach, the entire system is to transit rapidly between a high-performance active state and a minimal-power nap state in response to instantaneous load has been intended. Rather than requiring components that provide fine-grain power-performance trade-offs, it focuses on two optimization goals such as Optimizing server utilization and Turning off idle servers.

1. Related Work

For data center environments, the automatic scaling of web applications has been previously studied. In the work [2], an online application placement controller has been employed to decide the allocation of application and its load among the running instances. In order to balance load across the machines, it is essential to maximize the satisfied application demand as well as minimize the starts and stops of each application.

It significantly and consistently outperforms the existing state-of-the-art algorithm. Approaches like load dispatching and server provisioning for connection oriented internet services are in [3]. The performance and power models of connection servers, based on a real data trace collected from the deployed Windows Live Messenger, have also been characterized in this paper.

All works above do not use virtual machines and require that the applications be structured in a multi-tier architecture with load balancing provided through a front-end dispatcher. In our work, the virtual machines are treated like a black box.

A better solution to avoid turning on and off of same machines is to rotate servers in and out of the active clusters deliberately. Resource management is done only at the granularity of whole VMs. Quincy, a fair scheduling algorithm approach [4], though maximizing data locality maintains fairness among different jobs. The locality is achieved cent percent just by relaxing the fairness slightly using a simple algorithm called delay scheduling in [5]. In work [6], a novel VectorDot scheme has been developed to address the complexity introduced by the data center topology and the multidimensional nature of the loads on resources.

The design and implementation of a system that uses virtual machine technology to provide fast, transparent application migration has been illustrated in [1]. It aspects the problem of service degradation and is entirely not appropriate for Wireless Area Networks. An Auto Control, a feed-back control system to dynamically allocate computational resources to applications in shared virtualized environments is elucidated in [6]. Auto Control consists of an on-line model estimator that captures application-level performance and resource allocation and an MIMO resource controller that determines the appropriate allocation of multiple resources. On the other hand, it doesn't deal with bottleneck problems.

2. Scope of the Paper

A Resource Allocation System (RAS) is a mechanism that considers the current status of each resource in the Cloud environment, to better allocate physical and/or virtual resources to applications based on their demand. Generally, resources are located in a datacenter which are shared by multiple clients, thereby avoiding both the under-provisioning and overprovisioning of resources. The objective of this paper is not only to allocate such resources by establishing virtualization in a datacenter environment, but also considers the power consumption

of physical machines. The pre-step of allocating these resources dynamically is load prediction. The Fast Up and Slow Down(FUSD) predicts the load of physical as well as virtual machines that helps to migrate the virtual servers. Skewness algorithm measures the unevenness of resource utilization of a server. It helps to combine different types of workload and improves resource consumption. It not only prevents the system from overload, but also saves energy. While considering power saving strategies, the algorithm is executed periodically and the system is made to shut down when its utilization lies below the threshold. So, hibernate mode can be proposed instead of turning the server off or switching into sleep mode. Because sleep puts your work and settings in memory and draws a small amount of power, whereas hibernation puts your open documents and programs on disk, and then turns off the server. Of all the power-saving states, hibernation uses the least amount of power.

3. Proposed Methodology and Discussion

This paper focuses on the concept of virtualization technology to allocate data center resources dynamically based on the application demands and support green computing by optimizing the number of servers in use. It introduces skewness, an algorithm that measures the unevenness in the multi-dimensional resource utilization of server. The entire description of this work is outlined as Overall Flow Diagram for Dynamic Resource Allocation as shown in Figure1. Initially, the tasks in the datacenter are initiated whose load is predicted for each physical machine by consolidating the resources utilized by individual virtual machines. Once the load of each physical server has been detected, the overloaded servers are assumed to be hotspot.

The next step is to mitigate the hotspots to the under loaded physical servers. Hotspot mitigation can be achieved by migrating the virtual machines of the overloaded servers. The resources required by each physical or virtual server are met by the VM migration. Finally, the underutilized machines are made to cold spot whose power is turned off in order to enhance green computing. The results achieve both overload avoidance as well as green computing.

3.1 Load Prediction

The goal is to predict the future resource needs of VMs. One solution is to look inside a VM for application level statistics, e.g. by parsing logs of pending requests. But the problem that occurs in such environment is, it requires the modification of the VM which may not always be possible. Instead, we make our prediction based on the past external behaviors of VMs as in [7].

3.1.1 Fast Up and Slow Down Algorithm

The initial step is to estimate the load of an individual virtual machine running on the physical machines.The FUSD formula is used to predict the CPU load on the server. The load is measured every minute and prediction is done in the next minute. The estimated load of a server E(t) at a time t is defined as

$$E(t) = -|\propto| * E(t-1) + (1+|\propto|) * O(t) \quad (1)$$

$O(t) \rightarrow$ Obserbed load at time t

$\propto \rightarrow$ a constant whose range is $-1 \leq \alpha < 0$

On the other hand, when the observed resource usage is going down, reduction in estimation must be conserved. Hence, the two parameters, $\uparrow \alpha$ and $\downarrow \alpha$ to control how quickly E(t) adapts to changes, when O(t) is increasing or decreasing respectively. It is still quite acceptable nevertheless. The prediction algorithm plays an important role in improving the stability and performance of our resource allocation decisions. During the end of this module, the load of each virtual machine residing in the physical machines are predicted. The load of individual

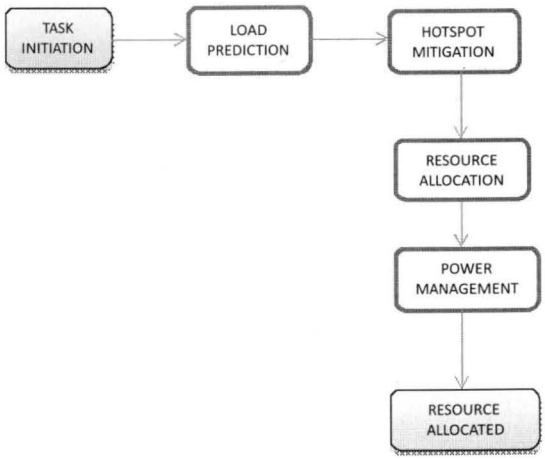


Figure 1. Overall Flow Diagram for Dynamic Resource Allocation

physical machines is predicted by consolidating the predicted load of virtual machines. The module results in predicting the load of the servers in the data center.

3.2 The Skewness Algorithm

Skewness is a measure of the asymmetry or unevenness of the probability distribution. A distribution may either be positively or negatively skewed. The concept of skewness is introduced to compute the unevenness in the utilization of multiple resources on a server. There are a number of ways of measuring skewness: In favor of performance and stability, a pragmatic algorithm is designed i.e. skewness into the aforementioned Cloud system. It is inspired by the fact that, if a PM runs too many memory-intensive VMs with light CPU load, much CPU resources will be wasted because it does not have enough memory for an extra VM. The concept of skewness is used to qualify the unevenness in the utilization of multiple resources on a server.

Let n be the number of resources and r_i be the utilization of the i-th resource. We define the resource skewness of a server p as

$$skewness(p) = \sqrt{\sum_{i=1}^{n} (\frac{r_i}{\bar{r}} - 1)^2} \quad (2)$$

Where, r is the average utilization of all resources for server p.

In practice, not all types of resources are performance critical and hence we only need to consider bottleneck resources. By minimizing the skewness, we can combine different types of workloads nicely and improve the overall utilization of server resources.

3.3 Hotspot Alleviation

As the load of the servers has been predicted or in other words, the utilization of all resources of each server has been determined, the next step is to mitigate the hotspots. First, the clarification of hotspot must be understood which is given below.

3.3.1 Hot and cold spots

The algorithm has been executed periodically and the resource allocation status based on the predicted future resource demands of VMs was evaluated. A server is said to be a hot spot if the utilization of any of its resources is above a hot threshold. This indicates that the server is overloaded and hence some VMs running on it should be migrated away. For example, the hot thresholds for CPU and memory resources be 90% and 80%, respectively. Thus a server is a hot spot if either its CPU usage is above 90% or its memory usage is above 80%.

The temperature of a hot spot p is defined as the square sum of its resource utilization beyond the hot threshold temperature,

$$temperature(p) = \sum_{r \in R} (r - r_t)^2 \quad (3)$$

R→set of overloaded resources in server p

r_t→hot threshold for resource

The point to be noted is that, only overloaded resources are considered in the calculation. The temperature of a hot spot reflects its degree of overload. If a server is not a hot spot, its temperature is zero. A server is defined as a cold spot if the utilizations of all its resources are below a cold threshold. This indicates that the server is mostly idle and a potential candidate is to turn off to save energy. However, we do so only when the average resource utilization of all actively used servers (i.e., APMs) in the system is below a green computing threshold. A server is actively used if it has at least one VM running. Otherwise, it is inactive. Finally, we define the warm threshold to be a level of resource utilization that is sufficiently high to justify having the server running but not as high as to risk becoming a hot spot in the face of temporary fluctuation of application resource demands.

3.3.2 VM migration

Once the hotspots have been determined, the next step is to list the hot spots in the system in descending temperature (i.e., we handle the hottest one first). The first and foremost goal is to eliminate all hot spots as much as possible. Otherwise, keep their temperature as low as possible.

The elimination of hotspots can be done by the migration of virtual machines. For each server p, which of its VMs should be migrated away is decided. The list of VMs based on the resulting temperature of the server is sorted if that VM is migrated away. The aim is to migrate away the

VM that can reduce the server's temperature the most.

In case of ties, selecting the VM whose removal can reduce the skewness of the server the most. For each VM in the list, the destination server to accommodate those virtual machines has to be found. The server not to become a hot spot after accepting this VM should also be a great consideration. Among all such servers, select the one whose skewness can be reduced the most by accepting this VM. Note that this reduction can be negative which means that we select the server whose skewness increases the least. If a destination server is found, the migration of the VM to that server is recorded and the predicted load of related servers was predicted [10].

Otherwise, move on to the next VM in the list and try to find a destination server for it. As long as a destination server for any of its VMs is found, consider the run of this algorithm a success and then move on to the next hot spot. Note that each run of the algorithm migrates away at most one VM from the overloaded server. This does not necessarily eliminate the hot spot, but at least reduces its temperature. If it remains a hot spot in the next decision run, the algorithm will repeat this process. It is possible to design the algorithm so that it can migrate away multiple VMs during each run. But this can add more load on the related servers during a period when they are already overloaded.

3.4 Resource Allocation

The first and foremost objective is to implement resource allocation mechanisms that provide an automated provisioning of resources and at the same time aims for the best utilization of available resources. The problem of selecting the most suitable physical and virtual resources to accommodate the application is a major problem. Various solutions have been evolved to overcome these problems. This work deals with the allocation of resource to the applications by migrating the virtual machines from one host to another. Energy saving can be achieved through many strategies, such as consolidating VMs according to current resource utilization, turning off or setting CPU hibernation when servers become idle, moving workloads or VMs, and so on.

Queuing information can also be used to implement resource allocation and energy management mechanism in virtualized datacenters. It assumes that to attend each application, there is a certain number of VMs hosted by the same number of servers. The servers' queuing information is used to make online control decisions, according to changes in the application's workload. The energy saving is done by switching idle servers into inactive mode when their workload is low, and turning them on again when workload increases.

3.5 Power Management

When the resource utilization of active servers is too low, some of them can be turned off to save energy. It can be handled by green computing algorithm. The challenge here is to reduce the number of active servers during low load without sacrificing performance either now or in the future. The oscillations in the system need to be avoided.

3.5.1 Green cloud

The green computing algorithm is invoked when the average utilizations of all resources on active servers are below the green computing threshold. The list of cold spots in the system based on the ascending order of their memory size has to be sorted. Since it is mandatory to migrate away all its VMs before shutting down an under-utilized server, the memory size of a cold spot is defined as the aggregate memory size of all VMs running on it [8]. The thing to be assumed is that all VMs connect to share back-end storage. Hence, the cost of a VM live migration is determined mostly by its memory footprint.

3.5.2 Server consolidation

For a cold spot p, we have to check whether all of its VMs can be migrated to somewhere else. For each VM on p, a destination server to accommodate it has to be found. The resource utilizations of the server after accepting the VM must be below the warm threshold. Though the energy can be saved by consolidating the under-utilized servers, overdoing it may create hot spots in the future. The warm threshold is designed to prevent that. If multiple servers satisfy the above criterion, prefer the one that is not a current cold spot. This is because increasing load on a

cold spot reduces the likelihood that it can be eliminated.

However, if necessary, a cold spot as the destination server is also accepted. All things being equal, a destination server is selected whose skewness can be reduced the most, by accepting this VM. If destination servers for all VMs on a cold spot is found, the sequence of migrations were recorded and the predicted load of related servers were updated. Otherwise, VM migration is not allowed. The list of cold spots is also updated because some of them may no longer be cold due to the proposed VM migrations in the above process.

The above consolidation adds extra load onto the related servers. This is not as serious a problem as in the hot spot mitigation case because green computing is initiated only when the load in the system is low. It should restrict the number of cold spots that can be eliminated in each run of the algorithm to be no more than a certain percentage of active servers in the system. This is called the consolidation limit.

4. Pseudo Code

Step 1: Create hosts

Step 2: Create few virtual machines in each host

Step 3: Evaluate the existing load as well as the future load in each individual virtual machines

Step 4: Estimate the load in each host by aggregating the load of virtual machines running on it.

Step 5: Calculate the average utilization of each hosts (p) and the temperature (t) as well

Step 6: if t > hot threshold, then

sort VM in descending order

else if t < cold threshold , then

sort VM in ascending order

Step 7: Pick out a VM to migrate to the appropriate destination host or turn it OFF (VM idle)

Step 8: Repeat step8, until the load gets balanced in the environment.

Step 9: End.

5. Simulation and Results

The simulation study involves the deterministic small datacenter environment with two physical hosts (servers), Host A and Host B. Each host consists of two and three instances (virtual machines) respectively. The proposed algorithm is implemented with OPEN STACK cloud computing software. The implementation is done with a single controller node and two compute nodes (Host A and Host B).

The observed and the predicted load for every 5 minutes in a physical server have been depicted as Observed and Expected Prediction in Figure 2. Once the load has been predicted for each host, the hotspots are listed. The HostB is found to be a hotspot, hence the VM with the maximum load of 70% in the HostB is picked and migrated to the appropriate destination host. The acceptance of this VM should not make the destination again a hotspot. All these constraints are fingered safely using skewness algorithm and elucidated in Figure 3 as Resource Allocation. Before Migration, Figure 3(a) illustrates the status of HostA and HostB before the migration of overloaded vm3. Similarly Figure 3(b), illustrates the status of HostA and HostB after the migration of overloaded vm3. In the same way, as the process progresses, HostA is found to be cold spot. The VM with 2% of utilization is determined. Hence, the VM is considered idle and the host is turned into sleep mode [9]. The migration of underutilized vma1 from HostA to HostB and switching the idle server Host A to standby mode is depicted in Figure 4. Thus the power consumption is diminished to achieve green computing for systems with multi resource constraints.

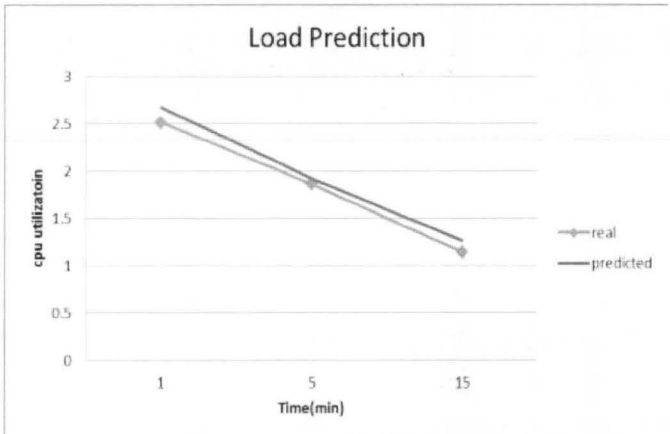
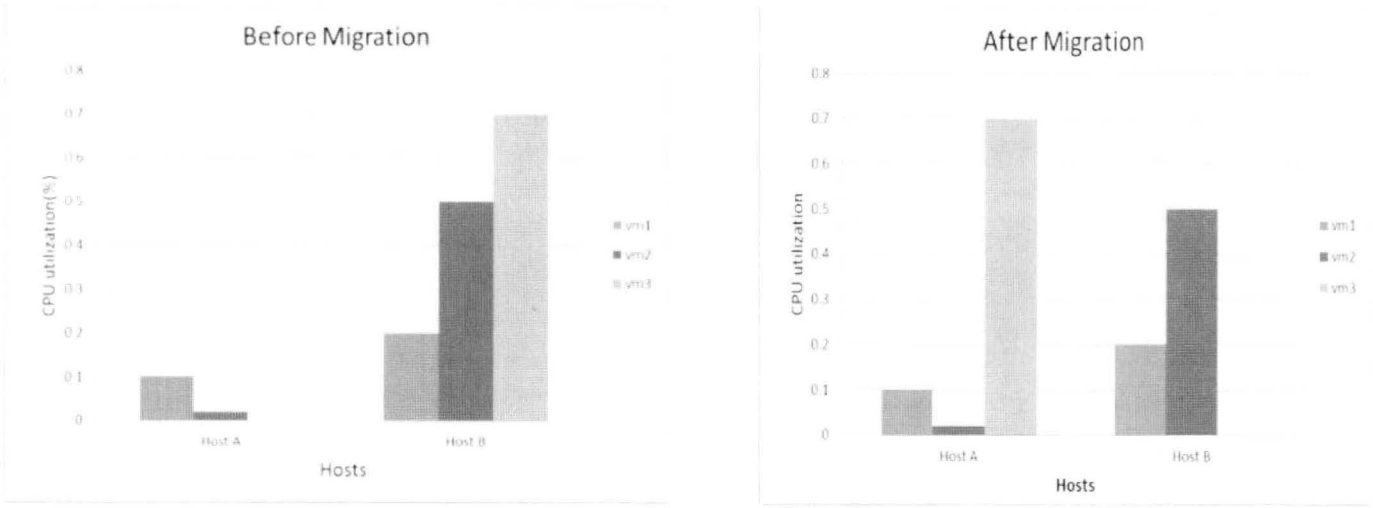


Figure 2. Observed and Expected Prediction



(a) Before migration

(b) After migration

Figure 3. Resource Allocation

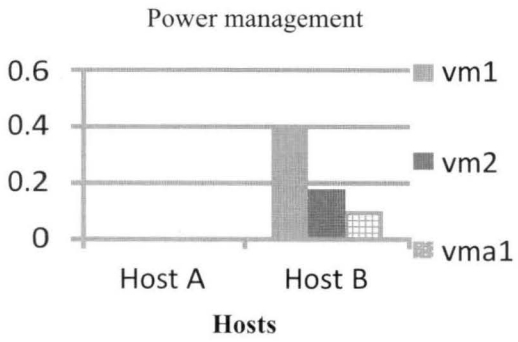


Figure 4. Cold spot

Conclusion and Future Enhancement

Dynamic resource allocation is the growing need of cloud providers for more number of users and with less response time. Cloud Computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the internet. Virtualization provides an efficient solution to the objectives of the cloud computing paradigm by facilitating the creation of Virtual Machines (VMs) over the underlying physical servers, leading to improved resource utilization and abstraction. Recent computers are sufficiently powerful to use virtualization to present the deception of many smaller VMs, each running a separate OS instance.

The design, implementation, and evaluation of a resource management system for cloud computing services have been presented. This system multiplexes virtual to physical resources adaptively based on the changing demand. The skewness metric is used to combine VMs with different resource characteristics appropriately so that the capacities of servers are well utilized. The algorithm achieves both overload avoidance and green computing for systems with multi-resource constraints.

As we could notice that, Cloud Computing is still in its maturation phase and there are many challenges to be solved or improved. About resource discovery and monitoring, the need of automating discovery and allocation processes, and make the monitoring process more active, is being able to re-allocate resources according to demand or to the current status of the Cloud (in order to optimize resource usage). An adaptive Cloud [11] that is able to self-manage its resources and self-adapt its configurations according to different circumstances can be designed.

References

[1]. **M. Nelson, B.-H. Lim, and G. Hutchins, (2005).** "Fast transparent migration for virtual machines," in *Proc. of the USENIX Annual Technical Conference*, pp. 391-394.

[2]. **C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, (2007).** "A scalable application placement controller for enterprise data centers," in *Proc. Of the International*

World Wide Web Conference (WWW'07), pp. 331-340.

[3]. G. Chen, H. Wenbo, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, (2008). "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. of the USENIX Symposium on Networked Systems Design and Implementation* (NSDI'08), pp. 337-350.

[4]. M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and Goldberg, (2009). "Quincy: Fair scheduling for distributed computing clusters," in *Proc. of the ACM Symposium on Operating System Principles (SOSP'09)*, pp. 261-276.

[5]. M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, (2010). "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proc. of the European conference on Computer systems (EuroSys'10)*, pp. 265-278.

[6]. P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, (2009). "Automated control of multiple virtualized resources," in *Proc. of the ACM European conference on Computer systems (EuroSys'09)*, pp. 13-26.

[7]. Prasad Saripalli, GVR Kiran, Ravi Shankar R, Harish Narware and Nitin Bindal, (2011). "Load Prediction and Hot Spot Detection Models for Autonomic Cloud Computing", in *the Proc. of Fourth IEEE International Conference*, pp. 397-402.

[8]. J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, (2001). "Managing energy and server resources in hosting centers," in *Proc. of the ACM Symposium on Operating System Principles (SOSP'01)*, pp. 103-116.

[9]. D. Meisner, B. T. Gold, and T. F. Wenisch, (2009). "Powernap: eliminating server idle power," in *Proc. of the international conference on Architectural support for programming languages and operating systems (ASPLOS'09)*, pp. 205-216.

[10]. N. Bobroff, A. Kochut, and K. Beaty, (2007). "Dynamic placement of virtual machines for managing sla violations," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management (IM'07), pp. 119-128.*

[11]. Yexi Jiang,Chand-shing Perng, Tao Li and Rong Chang, (2012). *"Self Adaptive cloud Capacity Planning"* in *Proc. of the IEEE 9th international conference*, pp. 73-80.

ABOUT THE AUTHORS

Mumtaj Zareena. M.S is presently pursuing her Master's degree in Computer Science and Engineering from Government College of Engineering, Tirunelveli. She received her Bachelor's degree in Computer Science and Engineering from Government College of Engineering, Tirunelveli in 2013. Her research interest includes virtualization in cloud computing and openstack, a cloud computing software.

M. Mahil is currently pursuing her PhD in Cloud Computing. She received her Bachelor's degree in Computer Science and Engineering from C.S.I. Institute of Technology, TamilNadu and Master's degree in Computer Science and Engineering from Manonmaniam Sundaranar University. Her research interest includes distributed and parallel computing, grid computing and cloud computing.

N. Rupavathy is presently pursuing her Master's degree in Computer Science and Engineering from Government College of Engineering, Tirunelveli. She received her Bachelor's degree in Information Technology from Sri Sivasubramaniya Nadar College of Engineering, Tamilnadu in 2013. Her research interest includes wireless sensor networks and cloud computing.