ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 5 May, 2014 Page No. 5614-5620

Survey Paper for Dynamic Resource Allocation using Migration in Cloud

Ms. Renu Krishnan, Ms. Silja Varghese

Department of Computer Science and Engineering
Nehru College of Engineering and Research Centre, Pampady, Thrissur, University of Calicut, Kerala, India.
renukrishnan.1990@gmail.com

Department of Computer Science and Engineering
Nehru College of Engineering and Research Centre, Pampady, Thrissur, University of caliicut, Kerala,India.
varghesesilja287@gmail.comg

ABSTRACT—*Cloud Computing is a rampant technology nowadays because of its scalability, flexibility, availability of resources and the other features. In cloud computing resource multiplexing is done through the virtualization technology. Virtualization technology is acts as a backbone for provisioning requirements of a cloud based solution. The problems arising in cloud computing using virtualization must be solved. In present cloud computing environment, load balancing is one of the challenging issues. To present a better approach for solving the problem of VM resource scheduling in a cloud computing environment, uses CPU and network usage calculation. A load predictor is used to predict the load in future, according to the this , it allocate the resources. For multiplexing of virtual machines to physical machines is managed using the Usher framework. Introducing the skewness algorithm to measure the resource utilization of server and minimizing the skewness can improve overall utilization of server and for load balancing the migration technology is used. Using this migration, can achieve the green computing.*

Keywords-Cloud computing, Virtualization, Migration, Green computing

I. INTRODUCTION

Cloud computing [1] is a new technology currently being studied in the academic world. The definition of the cloud computing from the Gartner: "A style of computing where massively scalable IT- related capabilities are provided as a service across the internet to multiple external customers using internet technologies".

The cloud computing platform guarantees subscribes that it sticks to the service level agreement by providing resources as service and by needs. However, day by day subscribers' needs are increasing for computing resources and their needs have dynamic heterogeneity and platform irrelevance. But in cloud computing environment, resources are shared and if they are not properly distributed then it will result into resource wastage. Another essential role of cloud computing platform is to dynamically balance the load amongst different servers in order to avoid hotspots and improve resource utilization. Therefore, the main problems to be solved are how as well as efficiently manage the resources.

Need of resources are increasing drastically day by day. So it is essential to allocate the resources properly but static allocations have some boundaries. So moving on to the dynamic resource allocation [2]. For using the virtualization techniques [3], can migrate virtual machines to physical machines effectively. By doing this some machines goes to the idle state and turning off these machines will lead to save the energy. So supports the green computing and thus resources can be dynamically allocated properly. On a cloud computing platform, dynamic resources can be effectively managed using virtualization technology. The subscribers with more demanding SLA can be guaranteed by accommodating all the required services within a virtual machine image and then mapping it on a physical server. This helps to solve problem of heterogeneity of resources and platform irrelevance. Load balancing of the entire system can be handled dynamically by using virtualization technology where it becomes possible to remap virtual machine and physical resources according to the change in load. Due to these advantages, virtualization technology is being comprehensively implemented in cloud computing. However, in order to achieve the best performance, the virtual machines have to fully utilize its services and resources by adapting to the cloud computing environment dynamically. The load balancing and proper allocation of resources must be guaranteed in order to improve resource utility. Thus, the

important objectives of this research are to be determine how to improve resource utility, how to schedule the resources and how to achieve effective load balance in a cloud computing environment. Dynamic resource allocation is done by using the virtualization technology. In this virtualization, migration of the VM's to PM's is required. For the better allocation, migration strategy is used here [4]. By defining the hot spot and cold spot, can balance the load and it avoids the overloading in the system's as well as it keep the energy efficiency.

A warm spot is defined which is below the hot spot and above the cold spot. Warm spot will balance the load and if allocate resources in as the warm spot condition, it will never goes to overload and the ideal condition. Here servers as a cold spot if the utilization of all its resources are below a cold threshold. This indicates that the server is mostly idle and a potential candidate to turnoff to save energy. After the allocation, the cloud controller finds idle system and by turning off this idle system, preserves the energy.

II. RELATED WORK

Survey includes the relative mechanisms and the methods which are implemented earlier and also the advantages and disadvantages of each method is described briefly. According to the survey of the earlier mechanism, it finds that the current system implemented has more advantages.

Ying Song *et al.* [5] A two-tiered on-demand resource allocation mechanism, including the local and global resource allocation, based on a two-level control model. A well-designed on demand resource allocation algorithm may minimize the waste of resources as well as guarantee the quality of the hosted applications. The local on-demand resource allocation on each server optimizes the resource allocation to VMs within a server taking the allocation threshold into account, while the global on-demand resource allocation optimizes the resource allocation among applications at the macro level by adjusting the allocation threshold of each local resource allocation.

A novel two-tiered on-demand resource allocation mechanism with feedback to optimize the resource allocation for VM-based data centers. In order to guide the design of the on-demand resource allocation algorithm, model the resource allocation using optimization theory. Base on the two-tiered on-demand resource allocation mechanism and model, local and global resource allocation algorithms to optimize the dynamic resource provision for VMs.

The local and lazy on-demand memory allocation algorithm, based on the static priority and the periodically collected idle memory of each VMi, MemFlow-L determines whether there is memory overload in a VM or not. The activity refers to the threshold of idle memory for memory overload. If idle memory of each VM is higher than, no memory needs to be reallocated. If IMi is lower than, MemFlow-L increases memory for VMi, as long as there is another VM that can give some of its memory to VMi.

To address the single-point failure problem of the global scheduler running the global resource allocation algorithm, dynamically select a server to run it. If the server running the global scheduler fails, randomly select another server to run this scheduler. Even if the global scheduler fails and no other global scheduler replaces it to work, all the local schedulers could continue working to allocation resource to the hosted VMs without the optimization at the macro level by the global scheduler. In a word, the global scheduler's failure could not lead to the failure of the resource allocation in the system. As to the scalability problem, the computation scale of the global scheduler is in direct proportion to the number of applications corresponding to i in the K-VM-1-PMmodel, because the constrained optimization in the K-VM-1-PM model is linear. Thus, the global scheduler is not the bottleneck even in a large-scale computing environment.

Christopher Clark *et al.* [6] Live OS migration is a extremely powerful tool for cluster administrators, allowing separation of hardware and software considerations, and consolidating clustered hardware into a single coherent management domain. If a physical machine needs to be removed from service an administrator may migrate OS instances including the applications that they are running to alternative machine(s), freeing the original machine for maintenance. Similarly, OS instances may be rearranged across machines in a cluster to relieve load on congested hosts. In the situations, the combination of virtualization and migration significantly improves manageability.

Here achieve this by using a pre-copy approach in which pages of memory are iteratively copied from the source machine to the destination host, all without ever stopping the execution of the virtual machine being migrated. Page level protection hardware is used to ensure a consistent snapshot is transferred, and a rate-adaptive algorithm is used to control the impact of migration traffic on running services. The final phase pauses the virtual machine, copies any remaining pages to the destination, and resumes execution there. Here eschew a 'pull' approach which faults in missing pages across the network since this adds a residual dependency of arbitrarily long duration, as well as providing in general rather poor performance.

Live migration refers to the process of making running virtual machines or applications between different physical machines without disconnecting the client or application. Memory, storage and network connectivity of the virtual machines are transferred from the original host machine to the destination.

Migration processes have certain steps to perform.

Stage 0: Pre-Migration Begin with an active VM on physical host A. To speed any future migration, a target host may be preselected where the resources required to receive migration will be guaranteed.

Stage 1: Reservation A request is issued to migrate an from host A to host B. Initially confirm that the necessary resources are available on B and reserve a VM container of

that size. Failure to secure resources here means that the VM simply continues to run on A unaffected.

Stage 2: Iterative Pre-Copy During the first iteration, all pages are transferred from A to B. Subsequent iterations copy only those pages dirtied during the previous transfer phase.

Stage 3: Stop-and-Copy Suspend the running OS instance at A and redirect its network traffic to B. CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage there is a consistent suspended copy of the VM at both A and B. The copy at A is still considered to be primary and is resumed in case of failure.

Stage 4: Commitment Host B indicates to A that it has successfully received a consistent OS image. Host A acknowledges this message as commitment of the migration transaction: host A may now discard the original VM, and host B becomes the primary host.

Stage 5: Activation The migrated VM on B is now activated. Post-migration code runs to reattach device drivers to the new machine and advertise moved IP addresses.

Marvin McNett *et al.* [7] Usher balances these imposing requirements using a combination of abstraction and architecture. Usher provides a simple abstraction of a logical cluster of virtual machines, or virtual cluster. Usher users can create any number of virtual clusters of arbitrary size, while Usher multiplexes individual virtual machines on available physical machine hardware. The Usher core implements basic virtual cluster and machine management mechanisms, such as creating, destroying, and migrating VMs. Usher clients use this core to manipulate virtual clusters. These clients serve as interfaces to the system for users as well as for use by higher-level cluster software. For example, an Usher client called ush provides an interactive command shell for users to interact with the system. And also implemented an adapter for a high-level execution management system, which operates as an Usher client that creates and manipulates virtual clusters on its own behalf. Two modules are there, first modules enable Usher to interact with broader site infrastructure, such as authentication, storage, and host address and naming services. Usher implements default behavior for common situations, e.g., newly created VMs in Usher can use a site's DHCP service to obtain addresses and domain names. Additionally, sites can customize Usher to implement more specialized policies; at UCSD, an Usher VM identity module allocates IP address ranges to VMs within the same virtual cluster. Second, pluggable modules enable system administrators to express site-specific policies for the placement, scheduling, and use of VMs. As a result, Usher allows administrators to decide how to configure their virtual machine environments and determine the appropriate management policies.

On the other hand, Usher provides a framework that allows system administrators to express site-specific policies depending upon their needs and goals. By default, the Usher core provides, in essence, a general-purpose, best-effort computing environment. It imposes no restrictions on the number and kind of virtual clusters and machines, and performs simple load balancing across physical machines. Here believe this usage model is important because it is widely applicable and natural to use. Requiring users to explicitly specify their resource requirements for their needs, for example, can be awkward and challenging since users often do not know when or for how long they will need resources. Further, allocating and reserving resources can limit resource utilization; guaranteed resources that go idle cannot be used for other purposes. However, sites can specify more elaborate policies in Usher for controlling the placement, scheduling, and migration of VMs if desired. Such policies can range from batch schedulers to allocation of dedicated physical resources. Usher maintains a clean separation between policy and mechanism. The Usher core provides a minimal set of mechanisms essential for virtual machine management. Usher provides a set of hooks to integrate with existing infrastructure. A Plugin API to enhance Usher functionality.

A running Usher system consists of three main components: local node managers, a centralized controller, and clients. A client consists of an application that utilizes the Usher client library to send virtual machine management requests to the controller. One LNM runs on each physical node and interacts directly with the VMM to perform management operations such as creating, deleting, and migrating VMs on behalf of the controller. The local node managers also collect resource usage data from the VMMs and monitor local events. LNMs report resource usage updates and events back to the controller for use by plugins and clients.

The controller is the central component of the Usher system. It receives authenticated requests from clients and issues authorized commands to the LNMs. It also communicates with the LNMs to collect usage data and manage virtual machines running on each physical node. The controller provides event notification to clients and plugins registered to receive notification for a particular event. Plugin modules can perform a wide range of tasks, such as maintaining persistent system-wide state information, performing DDNS updates, or doing external environment preparation and cleanup.

The client library provides an API for applications to communicate with the Usher controller. Essentially, clients submit requests to the controller when they need to manipulate their VMs or request additional VMs. The controller can grant or deny these requests as its operational policy dictates. One purpose of clients is to serve as the user interface to the system, and users use clients to manage their VMs and monitor system state. More generally, arbitrary applications can use the client library to register call backs for events of interest in the Usher system.

Typically, a few services also support a running Usher system. Depending upon the functionality desired and the infrastructure provided by a particular site, these services might include a combination of the following: a database server for maintaining state information or logging, a NAS server to serve VM file systems, an authentication server to provide authentication for Usher and VMs created by Usher, a DHCP server to manage IP addresses, and a DNS server for name resolution of all Usher created VMs. Note that an

administrator may configure Usher to use any set of support services desired, not necessarily the preceding list.

Xiaoyun Zhu *et al*. [8] AutoControl a resource control system that automatically adapts to dynamic changes in a shared virtualized infrastructure to achieve application SLOs. AutoControl is a combination of an online model estimator and a novel multi-input, multi-output resource controller. The model estimator captures the complex relationship between application performance and resource allocation, while the MIMO controller allocates the right amount of resources to achieve application SLOs. Virtualization is causing a disruptive change in enterprise data centers and giving rise to a new paradigm: shared virtualized infrastructure. In this new paradigm, multiple enterprise applications share dynamically allocated resources. These applications are also consolidated to reduce infrastructure and operating costs while simultaneously increasing resource utilization. As a result, data center administrators are faced with growing challenges to meet service level objectives in the presence of dynamic resource sharing and unpredictable interactions across many applications. These challenges include:

Complex SLOs: It is non-trivial to convert individual application SLOs to corresponding re-source shares in the shared virtualized platform.

Changing resource requirements over time: The intensity and the mix of enterprise application workloads change over time. As a result, the demand for individual resource types changes over the lifetime of the application. The utilization for both resources varies over time considerably, and the peaks of the two resource types occurred at different times of the day. This implies that static resource allocation can meet application SLOs only when the resources are allocated for peak demands, wasting resources.

Distributed resource allocation: Multi-tier applications spanning across multiple nodes require resource allocations across all tiers to be at appropriate levels to meet end-to-end application SLOs.

Resource dependencies: Application-level performance often depends on the application's ability to simultaneously access multiple system-level resources. Ina virtualized infrastructure, performance of a given application depends on other applications sharing resources, making it difficult to replicate its behavior in pre-production environments. Address the problem of managing the allocation of computational resources in a shared, virtualized infrastructure to achieve application-level SLOs.

Solution to this problem is AutoControl, an automated resource control and adaptation system. Main contributions are twofold: First, design an online model estimator to dynamically determine and capture the relationship between application level performance and the allocation of individual resource shares. Adaptive modeling approach captures the complex behavior of enterprise applications including varying resource demands overtime, resource demands from distributed application components, and shifting demands across multiple resources types. Second, design a two-layered, multi-input, multi output controller to automatically allocate multiple types of resources to multiple enterprise applications to achieve their SLOs. The first layer consists of a set of application controllers that automatically determines the amount of resources necessary to achieve individual application SLOs, using the estimated models and a feedback approach. The second layer is comprised of a set of node controllers that detect resource bottlenecks on the shared nodes and properly allocate resources of multiple types to individual applications. In overload cases, the node controllers can provide service differentiation by prioritizing allocation among different applications. AutoControl can detect and adapt to bottlenecks happening in both CPU and disk across multiple nodes. The AutoControl architecture allows the placement of AppControllers and NodeControllers in a distributed fashion. Node Controllers can be hosted in the physical node they are controlling. AppControllers can be hosted in a node where one of the application tiers is located. Here do not mandate this placement, how-ever, and the data center operator can choose to host a set of controllers in a node dedicated for control operations.

A model estimator that automatically learns in real time a model for the relationship between an application's resource allocation and its performance. An optimizer that predicts the resource allocation required for the application to meet its performance target based on the estimated model.

The main goal of the optimizer is to determine the source allocation required (ura) in order for the application to meet its target performance. An additional goal is to accomplish this in a stable manner, without causing large oscillations in the resource allocation.

Gong Chen *et al*. [9] Load skewing algorithms that allow significant amount of energy saving without sacrificing user experiences, i.e. maintaining very small number of SIDs. Understanding how power is consumed by connection servers provides insights on energy saving strategies. Connection servers are CPU, network, and memory intensive servers. There is almost no disk IO in normal operation, except occasional log writing. Since memory is typically pre-allocated to prevent run-time performance hit, the main contributor to the power consumption variations of a server is the CPU utilization .if pack connections and login requests to a portion of servers, and keep the rest of servers hibernating, here it can achieve significant power savings. However the consolidation of login requests results in high utilization of those servers, which may downgrade performance and user experiences. Hence, it is important to understand the user experience model before address the power saving schemes for large-scale Internet service.

The load of processing transactions is well under control when the number of connection and the login rates are bounded. So, will not consider transaction delays as a quality of service metric here. From this architectural description, it is clear that the DS and its load dispatching algorithm play a critical role in the shape of the load in connection servers. It motivates to focus on the interaction between CS and DS, the provisioning algorithms and load dispatching algorithms to achieve power saving while maintaining user experiences in terms of SNA and SID.A better alternative is to schedule

draining before turning a server off. More specifically, the dispatcher identifies servers that have the least amount of connections, and schedules them to connect to other servers at a controlled much slower pace that will not generate any significant burden for remaining active servers. In order to reduce the number of SIDs, also starve the servers for a period of time before doing scheduled draining or shutting it down. For Messenger servers, the natural departure rate caused by normal user log offs results in an exponential decay of the number of connections, with a time constant slightly less than an hour, meaning that the number of connections on a server decreases by half every hour. A two-hour starving time leads to number of SIDs less than a quarter of that without starving. The trade-off is that adding starving time reduces efficiency in saving energy. The goal is to maintain a small number of tail servers that have small number of connections. When user login requests ramp up, these servers will be used as reserve to handle login increases and surge, and give time for new servers to be turned on. When user login requests ramp down, these servers can be slowly drained and shut down. Since only tail servers are shut down, the number of SIDs can be greatly reduced, and no artificial surge of re-login requests or connection migrations will be created. On the hardware side, frequently turning on and off servers may raise reliability concern. Here it wants to avoid always turning on and off the same machines. This is where load prediction can help by looking ahead and avoiding short term decisions. A better solution is to rotate servers in and out of the active clusters deliberately.

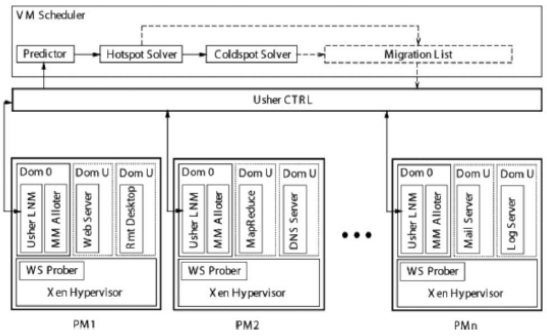
Comparisons of above described papers are shown in the table 1 below.

EXISTING SYSTEM	METHODS	ADVANTAGES	DISADVANTAGES
A Two Tired On-Demand Resource Allocation Mechanism for VM-Based Data Centers.	Two Tiered Allocation Mechanism 1)Local resource scheduler 2)Global resource scheduler	It address the problems of availability and scalability. If global resource allocation failure occurs then the local resource allocation will work, vice versa. So no failure of resource allocation is occurred.	Application workload scheduling is not considered. Mismatch between the on demand resource and workload dispatch.
Live Migration of Virtual machines.	Live migration: Migrating application into another system.	It is extremely powerful tool for clusters administrators. It will freeing the original machine for maintenance. Relieve the load on the congest hosts.	Sending of the VM's memory will consume the entire bandwidth. If only consider the live migration among the well-connected data center.
Usher:An Extensive Framework for Managing Clusters of Virtual Machines.	Usher Framework: Plugin API is for adding modules.	Provide a best effort computing environment. Performs load balancing across physical machines. Usher can be used for controlling the placement scheduling, and migration of VM's if desired.	For using other sites in usher, existing plugin's are not matching for this. For using the usher in another site needs to modify the existing plugin or rewrite it. No plugin's for managing clusters of physical machines is written.
Automated Control of Multiple Virtualized Resource.	AutoControl: An automatic control system	Performance assurance: All Applications can be meet their performance. Without human intervention allo automatically. Various workloads can be adopted. Scalability can be achieved.	AutoControl only does not deal the bottleneck problems. It does not control any memory control.
Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services.	Skewness algorithm	Load prediction will help to reduce the frequently turning on and off servers. Load prediction will reduce the power consumption . Load balancing is considered.	Sometimes load prediction may cause the failure. Migration of the load is not considered. Power is only considered as the parameter.

Table 1: Comparison table of existing systems

III. GENERAL SYSTEM MODEL

The architecture of the system is presented in Fig. 1. Each PM runs the Xen hypervisor (VMM) which supports a privileged domain 0 and one or more domain U.



Each VM in domain U encapsulates one or more applications such as Web server, remote desktop, DNS, Mail, Map/ Reduce, etc. We assume all PMs share a backend storage. The multiplexing of VMs to PMs is managed using the Usher framework. The main logic of our system is implemented as a set of plug-ins to Usher. Each node runs an Usher local node manager (LNM) on domain 0 which collects the usage statistics of resources for each VM on that node. The CPU and network usage can be calculated by monitoring the scheduling events in Xen. The memory usage within a VM, however, is not visible to the hypervisor. One approach is to infer memory shortage of a VM by observing its swap activities. Unfortunately, the guest OS is required to install a separate swap partition. Furthermore, it may be too late to adjust the memory allocation by the time swapping occurs. Instead we implemented a working set prober (WS Prober) on each hypervisor to estimate the working set sizes of VMs running on it. We use the random page sampling technique as in the VMware ESX Server. The VM Scheduler is invoked periodically and receives from the LNM the resource demand history of VMs, the capacity and the load history of PMs, and the current layout of VMs on PMs. The scheduler has several components. The predictor predicts the future resource demands of VMs and the future load of PMs based on past statistics. We compute the load of a PM by aggregating the resource usage of its VMs. The details of the load prediction algorithm will be described in the next section. The LNM at each node first attempts to satisfy the new demands locally by adjusting the resource allocation of VMs sharing the same VMM. Xen can change the CPU allocation among the VMs by adjusting their weights in its CPU scheduler. The MM Allotter on domain 0 of each node is responsible for adjusting the local memory allocation. The hot spot solver in our VM Scheduler detects if the resource utilization of any PM is above the hot threshold (i.e., a hot spot). If so, some VMs running on them will be migrated away to reduce their load. The cold spot solver checks if the average utilization of actively used PMs (APMs) is below the green computing threshold. If so, some of those PMs could potentially be turned off to save energy. It identifies the set of PMs whose utilization is below the cold threshold (i.e., cold spots) and then attempts to migrate awayall their VMs. It then compiles a migration list of VMs and passes it to the Usher CTRL for execution.

The summation of all the running VM's on a physical machine can be termed as the load of physical machine. If we assume that the T is the best time duration to monitor historical data i.e, from current time to the last T minutes will be the historical data zone, which will be used to solve the load balancing problem. By using the variation law of the physical machine load, we can split T into n subsequent time intervals. Therefore, T becomes $[(t_1 - t_0), (t_2 - t_1),\ldots, (t_n - t_{n-1})]$. The time interval k can be defined as $(t_k - t_{k-1})$. If the load of a VM is stable in all the periods then V (i, k) refers to the load of VM i in the interval k . By using the above definition, we can define the average load of VM on physical server P_i in the time cycle T is

$$\overline{V_l(l,T)} = \frac{1}{T}\sum_{k=1}^{n} V(i,k) \times (t_k - t_{k-1})$$

Utilizing the optimization of genetic algorithm and within this genetic algorithm we can produce the prediction, for that we can calculate the rank of each node and thus we can implement the prediction based algorithm.

By using this algorithm we generate different job scheduling sequence and select best sequence. Best sequence selection is based on rank.

IV CONCLUSION

Most of the firms are moving to cloud environment now a days. Moving to cloud is clearly a better alternative as they can add resources based on the traffic according to a pay-per-use model. But for cloud computing to be efficient, the individual servers that make up the datacenter cloud will need to be used optimally. Even an idle server consumes about half its maximum power. With the emergence of cloud computing in the past few years, Map Reduce has seen tremendous growth especially for large-scale data intensive computing. The lack of a separate power controller in Map Reduce frameworks post an interesting area of research to work on. It addresses the issue of job migration for clusters of nodes that run Map-Join-Reduce jobs. So migration cost based job reconfiguration algorithm is describes that dynamically reconfigures the job in accordance with the workload imposed on it. Currently the base paper has been implemented, which includes the cloud environment Map Reduce Framework. The work for implementation of a new framework Migration based Reconfiguration of nodes, a system that extends and improves Map Reduce runtime framework to efficiently process complex data analysis tasks with migration using Genetic Algorithm to be used for the future work.

REFERENCES

[1] M. Armbrust*et al.*, "*A View of Cloud Computing*,"*Commun. ACM*, vol. 53, no. 4, 2010, pp. 50–58..

[2] Zhen Xiao, Senior member, IEEE, weijia song and Qi chen "*Dynamic Resource allocation using Virtual Machines For Cloud Computing Environment*," IEEE Transaction on parallel and distributed systems, vol.24, No.6 june 2013.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "*Xen*

and the Art of Virtualization," Proc. ACM Symp. Operating Systems Principles (SOSP '03), Oct. 2003.

[4] M. Mishra and A. Sahoo, "*On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach*," *Proc. 4th Int'l. Conf. Cloud Computing*, 2011, pp. 275–82.

[5] Ying Song, Yuzhong Sun, Member, IEEE, and Weisong Shi, Senior Member, IEEE*A" Two-Tiered On-Demand Resource Allocation Mechanism for VM-Based Data Centers",* IEEE transactions on services computing, vol. 6, no. 1, january-march 2013.

[6] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield *"Live Migration of Virtual Machines",*University of Cambridge Computer Laboratory 15 JJ Thomson Avenue, Cambridge, UK.

[7] Marvin McNett, Diwaker Gupta, Amin Vahdat, and Geoffrey M. Voelker "*Usher: An Extensible Framework For Managing Clusters Of Virtual Machines",* University of California, San Diego.

[8] PradeepPadala, Kai-Yuan Hou Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, SharadSinghal, Arif Merchant "*Automated Control of Multiple Virtualized Resources",* The University of Michigan, Hewlett Packard Laboratories.

[9] Gong Chen, Wenbo He, Jie Liu, SumanNath, Leonidas Rigas, Lin Xiao, Feng Zhao "*Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services",*Dept. of Computer Science, University of Illinois, Urbana-Champaign, IL 61801