Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 1, January 2015, pg.367 – 373

SURVEY ARTICLE

A Survey on Dynamic Resource Allocation Method for Cloud Environment

Keerti Priya K[1], Ganya Poonacha C[2], Prof.Ashwini[3], Dr. Prakash Sheelvanthmath[4]

Student, M.Tech (Software Engineering), NHCE, Bangalore, India[1]

Student, M.Tech (Software Engineering), EPCET, Bangalore, India[2]

Jr. Assistant Professor, Information Science & Engineering Department, NHCE, Bangalore, India[3]

Head of department, Computer Science & Engineering Department, EPCET, Bangalore, India[4]

keertipriya.k@gmail.com; ganyacp@gmail.com

Abstract- Cloud computing describes a verity of computing concepts that involves a large number of computers connected through a network called internet. Cloud computing is pretty famous among users of cloud by offering a different variety of resources. Cloud computing is an on demand service because it offers dynamic flexible resource allocation and guarantees services to public in pay as-you-use manner. In this paper, we describe different techniques available for dynamic resource allocation for cloud users who use cloud environment.

Keywords: Cloud computing, Dynamic Resource Allocation, Virtual machine, Data center

I. INTRODUCTION

Cloud computing can be the delivery of computing services over a proprietary network or the Internet. These services basically include infrastructures which can be servers, storage devices, etc. It includes software applications as well as development platforms. The Cloud points out to multiple data centers located anywhere in the world that house the hardware which is necessary to offer cloud services. Cloud computing basically collects services and resources which are needed to perform functions as well as to meet dynamically changing needs. The platform of cloud computing always guarantees subscribes that it sticks to the service level agreement by providing needs and by providing resources as service. However, needs of daily subscribers' are increasing for computing resources and their needs have platform irrelevance and dynamic heterogeneity. But in the cloud computing environment, when resources are shared and if they are not properly distributed then it will result into wastage of resources. Another important use of cloud computing platform is, it is capable to balance the load amongst several servers dynamically just to avoid hotspots and improve utilization of resources.

Hence, the main problems to be solved are how to manage the resources efficiently. In the platform of cloud, resource allocation can take place in two places:

- In the cloud as soon as the application is uploaded, the load balancer gives physical computer with their instances which they have requested for, in order to attempt balancing the computational load of multiple applications across the physical computers.
- When multiple incoming requests are received by an application, each request should be assigned to a particular application instance in order to balance the computational load across a set of instances of the same application.

For example, to control how incoming requests are handled, Elastic Load Balancing (ELB) approach is used by Amazon EC2.

In cloud computing environment, Dynamic Resource Allocation is a process of assigning resources which are available to the cloud applications which are in need over the internet. If the allocation is not managed carefully, resource allocation starves services. This problem is solved by resource provisioning by permitting the service providers to manage the resources for each individual module.

On a cloud computing platform, one way to manage dynamic resources is by using virtualization technology. The subscribers who demand more SLA can be guaranteed by accommodating all the services which are required within a virtual machine image and then mapping it on a physical server. This helps to solve the problem of heterogeneity of resources and platform irrelevance. The entire system load balancing of can be dynamically handled by using virtualization technology where it is possible to remap virtual machine and physical resources according to the changes in load.

Dynamic Resource Allocation Strategy (DRAS) is about integrating cloud provider activities for utilizing and providing scarce resources within the cloud environment limit to meet the cloud application needs. It requires the resources amount and type, needed by each and every application just to complete a job of user.

Thus, the main objective of this paper is to research of different techniques available determining how to improve resource utility, how to schedule the resources and how to achieve effective load balance in a cloud computing environment.

II. RESEARCH IN DYNAMIC RESOURCE ALLOCATON TECHNIQUES

The survey includes different methods which are implemented earlier. It also contains the advantages and disadvantages of each method.

A. **"*Two-Tiered On-Demand Resource Allocation Mechanism for Virtual Machine Based Data Centers*"** [1] by Ying Song, Yuzhong Sun, this paper describes an algorithm which is well-designed on demand resource allocation which may minimize the waste of resources. It will also guarantee the hosted application's quality. In the local on-demand allocation of resources on each server optimizes the resource allocation to VMs within a server taking the allocation threshold into the account, where as 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 mechanism of two-tiered on-demand resource allocation is different from the traditional resource management in adding a resource management level for VMs. The main aim of on-demand resource allocation is to optimize the provision of dynamic resources for VMs. Each and every server hosts VMs which belongs to multiple application domains. Here, the workloads in VMs existing in the same server are time varying and different from each other, which is in turn resulting in the requirement of allocation of dynamic resource among VMs within a server. Each application workloads running on various servers are time varying and are different from other applications, which results in the requirement of on-demand resource allocation among applications. According to the technical support on the dynamic resource allocation to VMs within a server provided by the current VMMs, we can control resource allocation to VMs in each server independently. It is very much important to provide a global resource optimization which is based on the existing virtualization technology in such shared environment. Hence, two tiered on-demand resource allocation mechanism is proposed which is implemented by the local resource scheduler and global resource scheduler.

Here the problem is on-demand resource allocation to VMs and this is modeled using optimization theory. The two models are K-VM-1-PM (PM denotes physical machine) problem and the K-VM-N-PM problem to depict the resource allocation to K VMs within a server and to K VMs residing in N servers, respectively. These two particular models are general ones, and CPU or other resource allocations can, respectively, use them based on the K-VM-1-PM model, hence proposed a set of priority-based resource allocation algorithms to provide on-demand resources to the hosted applications. 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 at all idle memory of each VM is higher, then no memory needs to be reallocated. If IMi is lower, then MemFlow-L increases memory for VMi as long as there is another VM that can give some of its memory to VMi. To specifically address the single-point failure problem of the global scheduler running the global resource allocation algorithm, select a server dynamically for running it. In case the server running the global scheduler fails, then select another server randomly to run this scheduler. In case the global scheduler fails and no other global scheduler replaces it to work, then 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. With respect 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-PM model, because the constrained optimization in the K-VM -1-PM model is linear.

B. **"*Live Migration of Virtual Machines*"** [2] by Steven Hand, Christopher Clark, Andrew Warfield, Keir Fraser, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, this paper describes live migration of virtual machines. The Live OS migration is a good tool for cluster administrators, which allow separation of hardware and software considerations, and can also consolidate clustered hardware into a one coherent management domain. If a physical machine must be deleted from a service, then the administrator can transfer OS instances as well as the applications that they are running to various machine(s), which can free the first machine for maintenance. In a similar way, OS will be rearranged in a cluster to free load on congested hosts. In such cases, the combination of migration and virtualization improves manageability.

Here high-performance migration support for Xen [3] is implemented which is an open source VMM for commodity hardware. Here implementation and design addresses the issues and tradeoffs concerned in live local-area migration. As this paper target the migration of active OS which are hosting live services, it's important to cut off the downtime during which services are entirely inaccessible. Along with this we also consider the total migration time i.e, the time during which state on both the machines is synchronized and which thus could have an effect on reliableness. Here it's ensured that migration doesn't unnecessarily disrupt services which are active through resource contention (e.g., network bandwidth, CPU) with the migrating OS.

This is achieved by employing a pre-copy approach inside that page of memory which is iteratively copied to destination host from the source machine, and it should happen without having to stop the execution of the virtual machine that is being migrated. The page level protection hardware is used to make sure a consistent snapshot is transferred, and an algorithm referred to as rate-adaptive algorithm which is very much used to control the effect of migration traffic on the running services. The last phase pause the virtual machine that copies pages which are remaining to the destination, and resumes execution there. This paper uses an approach called 'pull' which faults in absent pages across the network because it adds a residual dependency which has an arbitrarily long duration. It also provide in general rather poor performance.

The migration process has a set of following steps [2]:

Stage 0: *Pre-Migration*. It starts on physical host A with an active VM. To speed up migration future, a target host could be preselected whereas the resources which are required to receive migration will be guaranteed.

Stage 1: *Reservation*. Host A request is issued to migrate an OS from host A to host B. Here, first it is confirmed that the necessary resources are available on host B and reserve a VM container of the required size. Failure to secure resources here means that the VM simply continue to run on host A unaffected.

Stage 2: *Iterative Pre-Copy*. In the process of first iteration, all of the pages are transferred from host A to host B. Subsequent iterations copy particularly only those pages which were dirtied during the earlier transfer phase.

Stage 3: *Stop-and-Copy*. Here, the suspension of the running OS instance is performed at host A and redirect its network traffic to host B. As was described earlier, CPU state and any of the remaining memory pages which are inconsistent are then transferred. At the end of this particular stage there is a consistent suspended copy of the VM at both host A and host B. The copy at host A is still considered to be primary which will be resumed if there is any failure.

Stage 4: *Commitment.* Host B indicates to host A, that it has successfully received a consistent OS image. Host A will then acknowledge this message as commitment of the migration transaction: host A now can discard the original VM, and host B can become the primary host.

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

C. **"*Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment*"** [4] by Karve A.A., Chieu T.C., Mohindra A., Segal A. This paper describes dynamic scaling method with novel design of web applications. This method is moved into virtual machine instances that are installed dynamically on a cloud which will illustrate the powerful scaling capabilities of the Cloud Computing environment. Based on demand these virtual machines are started and provisioned by a provisioning subsystem. But the process of provisioning and de-provisioning of web server virtual machine instances control by a dynamic scaling algorithm is based on relevant threshold of web application. Here, the Service Monitor sub-system is responsible in collecting individual scaling indicators from the web applications and then calculating their moving average. According to the average shown in the scaling indicator, a dynamic scaling algorithm to be given in the next section is useful for triggering a scaling event to the Provisioning subsystem. A different concept can be used here is, Image-based provisioning. Image-based provisioning is a deployment and activation mechanism that clones a "golden" virtual image to create new virtual machine instances. A problem with cloning virtual images is with handling of network, operating system, and application specific customization. Automating the provisioning of new virtual machines from a "golden" image template [5] can be achieved by adding automation capabilities into the template image, which is combined with external automation scripts that control the deployment. The technique mentioned in this paper performs the automated image-based provisioning of Linux-based virtual appliances based-on Xen hypervisor technology [6] in Red Hat Enterprise Linux system [7]. The simplified provisioning process is used, that leverages the Linux disk image mount utility on the cloned Xen image and performs host name and network fix ups before booting. The whole process can be summarized in two simple steps. The first step is to copy the "golden" image appliance template to the target Linux host system, mount the image file to the host system, and to perform fix-ups of hostname and network in the mounted file system. The second step of this process involves the powering up of the virtual machine, and the remote execution of secure shell scripts on the VM for password change and for appliance specific customization.

The algorithm used here is scaling algorithm. The scaling algorithm is very well implemented in the Service Monitor sub-system. It is only used to trigger and control the scale-up or scale-down in the Provisioning sub-system which depends on the number of virtual machine instances which is based on the statistics of the scaling indicator.

D. ***"Usher: An Extensible Framework for Managing Clusters of Virtual Machines"*** [8] by Marvin McNett, Amin Vahdat, Diwaker Gupta and Geoffrey M. Voelker, this paper describes Usher as a cluster management system which is designed to decrease the administrative load of managing cluster resources where as at the same time it also improves the ability of users to request, control, customize their resources and also computing environment. Usher provides an easy idea of logical cluster of, virtual cluster or virtual machines. Users of Usher can create number of virtual clusters (VCs) whose size is arbitrary, where as Usher multiplexes individual virtual machines (VMs) on accessible physical machine hardware. It decouples logical machine resources from the physical machines, hence users will be able to create and use machines based on their requirements rather than based on assigned physical resources. The Usher core develops a basic virtual cluster and machine management mechanisms, like making, migrating and destroying VMs. Usher client use this core to control virtual clusters. These clients will function as an interface to the system for users and for use by higher-level cluster software package. As an example, associate Usher consumer known as "ush" provides associate active command shell for users that will interact with the system. We have conjointly enforced the associate adapter for execution management system at high level [9] which is operated as an associate Usher consumer, which manipulates and makes virtual clusters on its own behalf.

Usher aids customizable modules for two purposes. 1st purpose is, the modules allows Usher to communicate with broader site infrastructure, like host address and naming services and authentication. Usher implements default behavior for common situations. 2nd purpose is, pluggable modules allow administrators of the system to make categorical site-specific policies for the scheduling, placement, and usage of VMs. As a result of this, Usher will provide permission to the administrators to make decision on how the virtual machine environment has to be configured and appropriate management policies have to be determined. For example, the administrators will install an Usher scheduling which is available and placement plugin that will perform the round-robin placement of VMs across the different physical machines and easy rebalancing as a response to the removal or addition of virtual and physical machines to support a general-purpose computing environment. With this plugin, users will dynamically add or take away VMs from VCs at any time while not having to specify the service level agreements [10, 11, 12], write configuration files, or get the resources on leases. With live migration of Virtual Machines, Usher will dynamically and transparently alter the mapping of virtual to physical machines in order to adapt to the changes in load among active Virtual Machines or the working set of active Virtual Machines, which will exploit all the affinities among the available VMs (e.g., to boost physical page sharing [13], or add and take away hardware with very little or no interruption). The Usher core provides, in essence, an all-purpose, best effort computing environment. It will impose no restrictions upon the number and type of virtual clusters and machines, and performs easy load balancing across physical machines.

Usher Summary

Running Usher system has 3 particularly important components: a centralized controller, local node managers (LNMs) and clients. A client has an application that uses the Usher client library which sends VM management requests to controller.

i. **Local Node Managers-** Here, the local node managers (LNMs) operates with the hardware. On every physical node in Usher system LNMs run as servers. Each of the LNM gives a remote API to the controller for manipulating the Virtual Machines that are present on its node and on invoking this API method the LNM will translate the operation into an equivalent operation of the VM management API which is exposed by the VMM that runs on the node. An administrator will register multiple authentication modules, and Usher can query each of them in return. This is often useful, for providing local password authentication if in any case LDAP or NIS authentication fails. Once receiving a user's secured documents as mentioned on top, the controller will then check them against the active authentication module chain. The user is authenticated only if at least one wins before reaching the end of the chain. Or else, the authentication will fail and hence the user must try again and again.

ii. **Usher Controller**-The usher controller is the main component of the Usher system which may either be bootstrapped into a VM which that runs within the system or that runs on a separate server. The controller provides as follows:

   - **User authentication**: The SSL-encrypted user authentication is used by usher. It is necessary for the users of the Usher system to be authenticated in order to make requests of the system. And the administrators will use any of the authentication modules which are included for using with various authentications. In conjunction with this, the controller maintains 3 alternative VMs that are: *lost VMs*, *missing VMs* and *unmanaged* VMs.

   - **VM operation request API**: This API provides the entry into the system for the VM management requests (via RPC) from the connecting clients. The authorization plugin is invoked by the controller in so as to examine whether or not the authenticated user will perform the operation before continuing. The controller also will be able to invoke different plugins for performing the preprocessing like creating placement decisions and checking resource handiness at this point.

   - **Consolidation of LNM monitoring data:** The controller takes up the responsibility of consolidating observing information that's sent by the native node managers into a format that is accessible by the remaining system. The clients make use of this data to in order to describe the system's state to users, and plugins make use of the above described data to create the policy decisions.

   - **Event notification**: Usher has to alert clients and plugin modules once varied events in the system occur. Events fall into one of 3 categories, which are VM operation requests, VM state changes and Errors and unexpected events. The change in the state of the VM is automatically being notified to the client and based on this notification, clients are allowed to take any action and safely ignore them. But the plugin modules, must explicitly register with the controller in order to receive the event notifications. Plugins are going to be allowed to register for any style of event within the system as specified above.

iii. **Clients and the Client API-** Usher client API is used by the client which enables interaction along with the Usher controller. It provides methods for manipulating or requesting VMs and performing state queries. Any application using this API is taken into account as a client. Client API provides techniques for clients to safely and securely authenticate and connect with the Usher controller. Once its connected, an application might call any of the strategies provided by the API. All the strategies are asynchronous or event-based calls to the controller.

Few other services support the running of an Usher system. Based on the functionality that is desired and the infrastructure which is provided by a specific website, the services could embody the following: a NAS server to serve VM file systems, a database server for maintaining state information, an authentication server to supply authentication for the Usher and Virtual Machines created by Usher, a DNS server for the name resolution of all Usher created VMs and a DHCP server in order to manage IP addresses.

E. ***"Automated Control of Multiple Virtualized Resources"*** [14] by Kai-Yuan Hou Kang G. Shin, Mustafa Uysal, Xiaoyun Zhu, Zhikui Wang, PradeepPadala, SharadSinghal, Arif Merchant, this paper describes AutoControl as combination of novel multi-input, multi output (MIMO) resource controller and an online model estimator. The MIMO controller assigns the correct amount of resources to achieve application SLOs, where-as the model estimator captures the complex relationship between resource allocation and application performance. AutoControl can also detect and adapt to disk I/O and CPU bottlenecks which occur across multiple nodes over time and multiple virtualized resources are allocated accordingly in order to achieve application SLOs.

There are 2 contributions: 1st, an online model estimator is designed to determine and capture the relationship between allocation of individual resource shares and application level performance dynamically. Adaptive modeling approach will capture the complicated behavior of enterprise applications as well as the resource demands from distributed application components, varying resource demands over time, and shifting demands across multiple resources types. The first layer includes set of application controllers which will be used for automatically determining the amount of resources that will be needed to achieve individual application SLOs, by making use of the feedback approach and estimated models. 2nd, Design a two-layered, multi-input, multi-output (MIMO) controller which automatically allocates different types of resources to various different enterprise applications in order to achieve their SLOs. The second layer consists of a set of node controllers. These node controllers will detect resource bottlenecks on the shared nodes and then it will properly allocate resources of multiple types to individual applications. In case of overload, the node controllers provide service differentiation which is done by prioritizing allocation among different applications.

AutoControl has a set of application controllers (AppControllers) and a set of node controllers (NodeControllers). One AppController for each hosted application, and one NodeController for each virtualized node. For each application, its AppController will periodically poll an application performance sensor for the measured performance. This is called as the control interval. After this the AppController will compare this measurement along with the application performance target. Based on the discrepancy, it automatically determines the resource allocations that is needed for the next control interval, and sends these requests to the NodeControllers for the nodes that host the application.

For each of the node, based on the collected requests from all relevant AppControllers, the corresponding Node Controller will determine whether it is having enough resource of each type in order to satisfy all the demands, and then it computes the actual resource allocation. The computed allocation values are stored in the resource schedulers in the virtualization layer for actuation. This will then allocate the corresponding portions of the node resources to the Virtual machines in real time. AutoControl can detect and adapt to bottlenecks happening in both CPU and disk across multiple nodes.

III. COMPARISION TABLE

EXISTING SYSTEM	ADVANTAGES	DISADVANTAGES
Two-Tiered On-Demand Resource Allocation Mechanism for VM Based Data Centers	-It addresses the problems of availability and scalability. -In case, if global resource allocation fails, then local resource allocation will work and if local resource allocation fails, then global resource allocation works.	-Application workload scheduling is not considered. -There can be a mismatch between on demand resource and workload dispatch.
Live Migration of Virtual Machines	-It is a very good tool for cluster administrators. -It will free the original machine for maintenance when ever required. -It will also reduce the load on congested node.	-Transferring the virtual machine's memory will consume entire bandwidth.
Usher: An Extensible Framework for	-Load balancing is performed among the various machines present.	-When other sites are used in Usher, the existing plugin's does not match for this.

Managing Clusters of Virtual Machines	-It also provides a best effort computing environment. -Usher will enable controlling of migration of VM's and also controlling of the placement scheduling as desired.	-If Usher is to be used in other site then the existing plugins will have to be modified or rewritten. -For managing the clusters of physical machines, no plugin's are written.
Automated Control of Multiple Virtualized Resources	-Scalability can be achieved. -Different types of workload can be adopted. -Provides performance assurance wherein all the applications can achieve good performance. -There is no need for human intervention.	-It does not control any memory.
Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment	-Dynamic scaling has the ability to add as well as remove capacity into the cloud infrastructure on a whim- ideally because the traffic patterns will change which can be adjusted accordingly.	

IV. CONCLUSION

The cloud service providers have a growing need for dynamic resource allocation. This survey paper gives a study of various dynamic resource allocation techniques in cloud environment. This paper also mentions advantages and disadvantages of each paper.

REFERENCES

[1] Yuzhong Sun, Ying Song, 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.

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

[3] K. Fraser, P. Barham, B. Dragovic, T. Harris, S. Hand, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, *Xen and the Art of Virtualization*, Proc. ACM Symp. Operating Systems Principles (SOSP '03), Oct. 2003.

[4] Chieu T.C., Karve A.A., Mohindra A., Segal A., *Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment*, in IEEE International Conference on e-Business Engineering, Dec. 2009, pp. 281-286.

[5] S. Smith, L. He, R. Willenborg, Q. Wang, *Automating deployment and activation of virtual images*, IBM developerWorks,Aug. 2007. http://www.ibm.com/developerworks/websphere/techjournal/0708_he/0708_he.html

[6] Xen Hypervisor, http://www.xen.org/

[7] Red Hat Enterprise Linux, http://www.redhat.com/rhel/

[8] Amin Vahdat, Marvin McNett, Diwaker Gupta, and Geoffrey M. Voelker, Usher*: An Extensible Framework for Managing Clusters of Virtual Machines*, 21st Large Installation System Administration Conference (LISA '07), University of California, San Diego.

[9] Ryan Braud, Albrecht, Jeannie, Christopher Tuttle,Darren Dao, Nikolay Topilski, Alex Amin Vahdat ,C. Snoeren, *Remote Control: Distributed Application Configuration, Management, and Visualization with Plush*, Proceedings of the Twenty-first USENIX Large Installation System Administration Conference (LISA), November, 2007.

[10] S. Fakhouri, L. Fong, Appleby, K., D. Pazel, S. Krishnakumar, M. K. G. Goldszmidt, J. Pershing, and B. Rochwerger, *Océano – SLA-based Management of a Computing Utility*, Proceedings of the IFIP/IEEE Symposium on Integrated Network Management, May, 2001.

[11] Junghwan Rhee, Ruth, Dongyan Xu, R. Kennell, and S. Goasguen, *Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure*, IEEE International Conference on Autonomic Computing, June, 2006.

[12] Wood, Arun Venkataramani, Timothy, Prashant Shenoy, and Mazin Yousif, *Black-box and Gray-box Strategies for Virtual Machine Migration*, Proceedings the Fourth Symposium on Networked Systems Design and Implementation (NSDI), April, 2007.

[13] Carl A., Waldspurger, *Memory Resource Management in VMware ESX Server*, Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02), December, 2002.

[14] Kai-Yuan Hou, Pradeep Padala, Kang G. Shin, Mustafa Uysal, Zhikui Wang, Xiaoyun Zhu, Sharad Singhal, Arif Merchant, *Automated Control of Multiple Virtualized Resources*, Hewlett Packard Laboratories, October, 2008.