Journal of Computer Science and Applications.
ISSN 2231-1270 Volume 6, Number 2 (2014), pp. 69-77

http://www.irphouse.com

# A Hybrid Approach for Optimizing Resource Allocation in Cloud Computing

**K. Raja[1] and M. Prabhakar[2]**

*Madanapalle Institute of Technology & Science,*
*Computer Science and Engineering,*
*Jawaharlal Nehru Technological University Anantapur, India*
*kukatlaraja@gmail.com*

ABSTRACT

Cloud computing is a new model of computing that has emerged as a technology which leverages the lives of people and organizations across the globe. Commoditization of computing resources has been made possible with virtualization and cloud computing technologies. Cloud is able to provide plethora of services to its users. Dynamic resource allocation is essential for the success of cloud as vast number of users need resources on the fly. Dynamic resource allocation has been a challenging problem in cloud computing that needs to be addressed. Many solutions came into existence for dynamic resource allocation. In this paper we propose an algorithm that combines two possible cases for optimizing resource allocation. The first case is, when a VM over commitment is witnessed, the VM's capacity gets increased by using either free memory of PM or idle memory of other VM. The second case is that a VM migration technique is employed so as to move over-committed VM to other PM. Thus the proposed algorithm achieves optimal resource allocation dynamically. We built a prototype application, a custom simulator, which demonstrates the proof of concept. The empirical results are encouraging.

**Keywords**— computing, virtualization, load balancing, optimal resource allocation

## I. INTRODUCTION

Virtualization is the technology that enabled cloud computing to be affordable and viable. Resource virtualization at various levels made cloud computing possible. As cloud resources are costly and need huge investment, from the service provider point

of view it is essential to have dynamic resource allocation so as to optimize the utilization of resources. If there is no such mechanism, it ends up with consumption of more resources that ultimately leads to unsuccessful implementation of cloud. Many algorithms or techniques came into existence. MUSE is included where replicas are maintained for optimal performance; load balancing strategies ; integrated load dispatching approach; delay scheduling [17], HARMONY where multiple resource layers are used [12]; and skewness algorithm for dynamic resource allocation [15]. These works focused on various strategies. However, in this paper we use hybrid approach to solve the problem of dynamic resource allocation.

In this paper we proposed a novel architecture to solve the problem of dynamic resource allocation. Our contributions in this paper are as described here.

- We proposed architecture for cloud infrastructure to achieve optimal resource allocation.
- We proposed an algorithm by name over-commitment mitigation algorithm for dynamic resource allocation and optimization of resource allocation and utilization. This algorithm has different strategies to handle the problem of dynamic resource allocation.
- We built a web based prototype application that demonstrates the proof of concept. The application simulates the cloud environment and dynamic resource allocation as per the proposed algorithm.

The remainder of the paper is structured as follows. Section II provides review of literature that exists about the prior works on dynamic resource allocation problem. Section III presented the proposed arhictecture. Secion IV provides the algorithm proposd. Section V presents experimental results while section VI concludes the paper.

## II. RELATED WORKS

With respect to data centers and resource allocation to various applications at application level was given importance to research in [15] and [14]. Load dispatching and server provisioning were introduced in [10]. However, these works did not use virtual machines. Nevertheless, these experiments were done in multi-tier architectures. The work proposed in [17] targeted a cloud environment that is Amazon EC2-style environment where experiments are made with dynamic resource allocation. There was concept of VMs and their allocations. In [14] MapReduce is explored which is a framework for distributed programming. Scheduling algorithm was used in [17] to minimize cost while executing jobs. The delay scheduling concept was introduced in [12] for effectiveness with local data processing. Dynamic priorities to jobs was explored in [15] for optimal resource allocation.

Live VM migration concept was explored in [5], [10] and [12]. In this approach a VM is migrated from one physical resource to another physical resource. In [12] load balancing mechanisms are explored. The load balancing concept has been around and it is used in all cloud environments where server machines are given requests based on their load. In this paper we combined many strategies and built a hybrid algorithm

that can cope with various runtime requirements and resource dynamics for optimal resource allocation and utilization.

## III. HYBRID APPROACH FOR OVERLOAD AVOIDANCE AND DYNAMIC RESOURCE ALLOCATION

In cloud, virtualization plays an important role in resource utilization. In fact the virtualization technology made the cloud computing paradigm affordable as it can improve the utilization of physical resources. With less physical resources it is possible to serve more number of requests from clients. In this process virtualization mechanism in cloud data center is crucial. Therefore we considered the optimal resource allocation problem as non-trivial and studied the existing algorithms that are used for dynamic resource allocation. In this section we propose an architecture that can have our proposed algorithm running to allocate resources dynamically in order to optimize the usage of cloud resources. The architecture focuses on VM assignments, overload estimation and other features that will leverage the functionality of cloud data center. Figure 1 shows an overview of our architecture.

### A.System Architecture

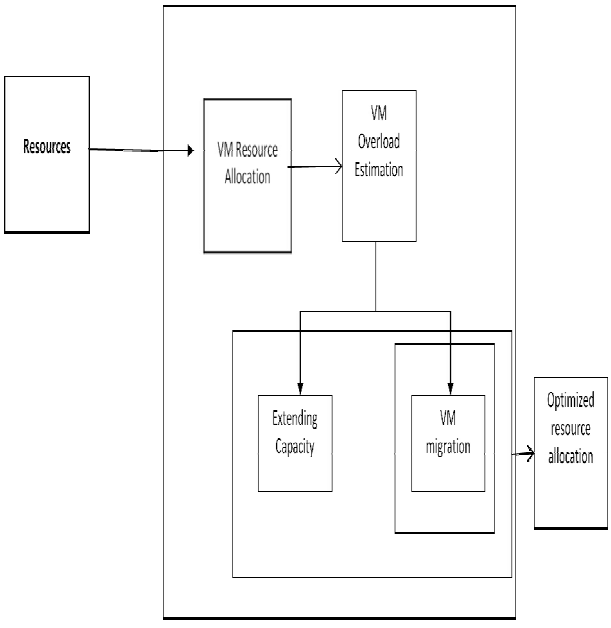


**Figure 1 – Architectural overview of the proposed system**

Here is the description about the proposed architecture. There are number of physical resources on top of which number of virtual machines is assigned. The physical resources have certain capacity limit. However, the physical resources are available so as to allocation of new physical machine is possible for business continuity and disaster mitigation. Nevertheless, it is imperative that the data center

should have strategies to allocate resources optimally. Towards it we could identify two strategies which are very useful. The first strategy is known as extending the capacity of physical machine while the second strategy is migrating VM from one physical machine to another physical machine. When any VM experiences over commitment our algorithm (described in the next section) starts working. It combines two possible cases for optimizing resource allocation. The first case is, when a VM over commitment is witnessed, the VM's capacity gets increased by using either free memory of PM or idle memory of other VM. The second case is that a VM migration technique is employed so as to move over-committed VM to other PM. Thus the proposed algorithm achieves optimal resource allocation dynamically.

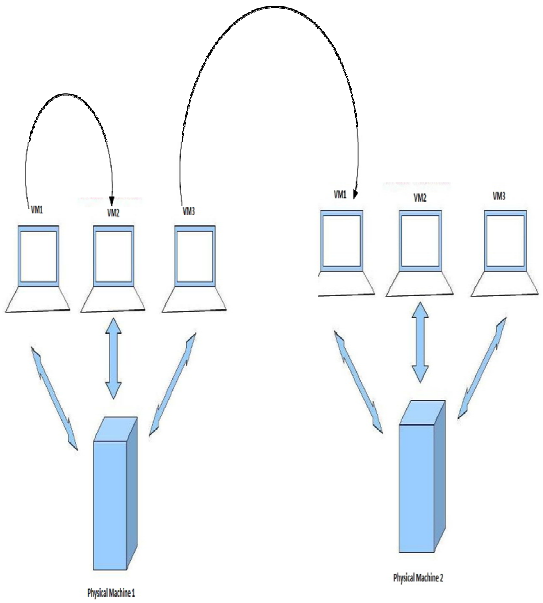


**Figure 2 – Live VM migration**

As shown in Figure 2, the VM3 of physical machine 1 is migrated to physical machine 2 as per the runtime dynamics according to the proposed algorithm which is described in ensuing section.

**B.Algorithm :** *algorithm for overload avoidance and optimizing resource utilization*
This algorithm is meant for achieving optimal resource utilization in cloud computing environment. This is in tune with the architecture proposed in the previous section. Optimized resource allocation is the goal of this algorithm. It takes the available resources as input and allocates resources optimally. The algorithm is strategically executed in cloud data center so as to ensure optimal resource utilization that can increase the scalability of the cloud.

**Listing 1 – Overload mitigation algorithm**

**Algorithm** : Over-Commitment Mitigation Algorithm
**Inputs** : Available Resources Allocation
**Outputs** : Achieve Overload Avoidance in an over-committed cloud
Initially Available Resources are allocated to cloud clients based on their demand.
Capacity of physical machine P;
Initialization:
Assign definite number of VMs to each physical machine;
Cloud is over-committed if 90% of its capacity P is used
**Checking Free Capacity of all PMs**
First we calculate the capacities of PMs after resource allocation to VMs is done
**Observing over-commitment of cloud**
for h = 1 *to* $N_{pm}$ do
IF VM 'h' capacity requirement is more than its allocated capacity THEN
report machine h is overloaded;
OverList = OverList ∪ h;
**Avoiding over-commitment problem**
This can be done either extending over loaded VM capacity or by migrating the VM to another physical machine having sufficient resources.
**CASE 1: VM CAPACITY EXTENSION**
When VM gets overloaded
First Check PM free memory
IF PM has available free memory THEN
Allocate the free memory to overloaded VM
Overloaded VM capacity= original VM capacity **U** free memory of PM
END
IF Free memory of PM is not available THEN
Check for unused memory of other VMs running on the Physical machine
IF available, then reallocate the unused memory to overloaded VM
Overloaded VM capacity= original VM capacity **U** unused memory of other VM on PM
END IF
END
IF above two cases fails THEN VMs are exchanged
Find best VM to exchange
Check any VM on other PM is ready to exchange
IF VM on other PM is ready to exchange with overloaded VM THEN
X = exchange(overloaded VM, VM on other VM)
Two VMs are successfully Exchanged
END IF
END
**CASE 2: LIVE VM MIGRATION**
IF capacity extension fails THEN
Prepare OverList with overloaded VMs

```
for each VM in OverList do
check overload comparing each other
Identify over loaded VM to migrate
Identify other PM with available resources
Use Sequence migration if possible
   END IF
END
```

As seen in Listing 1, the algorithm combines two possible cases for optimizing resource allocation. The first case is, when a VM over commitment is witnessed, the VM's capacity gets increased by using either free memory of PM or idle memory of other VM. The second case is that a VM migration technique is employed so as to move over-committed VM to other PM. Thus the proposed algorithm achieves optimal resource allocation dynamically. This will increase resource utilization and avoids unnecessary deployment of physical resources. This phenomenon is also linked to the possible reduction of investment risks in cloud computing from the stand point of cloud service provider.

## C. EXPERIMENTAL RESULTS

We built a prototype web application, a custom simulator, which demonstrates the proof of concept. The empirical results are encouraging. The environment used for simulation study include JDK 1.7, Tomcat 7.0, MY SQL, Chrome running in Windows 7 machine with 4 GB RAM, core 2 dual processor. The prototype demonstrates our architecture and the underlying algorithm. Local machine resources are utilized in order to simulate the cloud. The study reveals that it is possible to achieve successful optimization of resource allocation and utilizations by using the proposed algorithm named "Over-Commitment Mitigation Algorithm". Here are some snapshots to understand how the simulation results reveal the dynamic resource allocation as per the proposed solution.

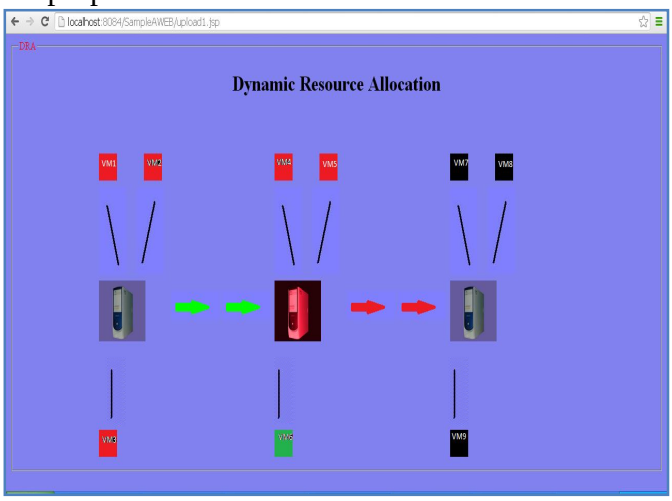


**Figure 2 – Dynamic resource allocation**

As seen in Figure 2, it is evident that resource allocation could not be possible with physical machine as indicated in red color VMs. The resource allocation is done for a VM on physical machine 2.

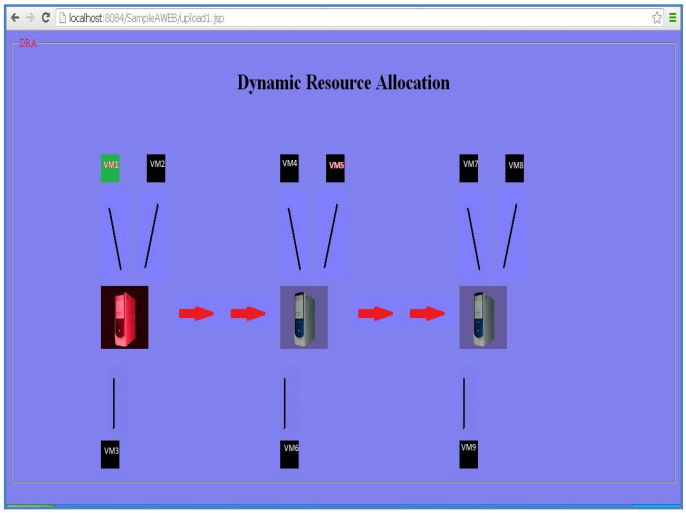


**Figure 3 – Dynamic resource allocation**

As seen in Figure 3, it is evident that resource allocation could be possible with physical machine 1 as indicated in green color VM. The resource allocation is done for a VM on physical machine 1.

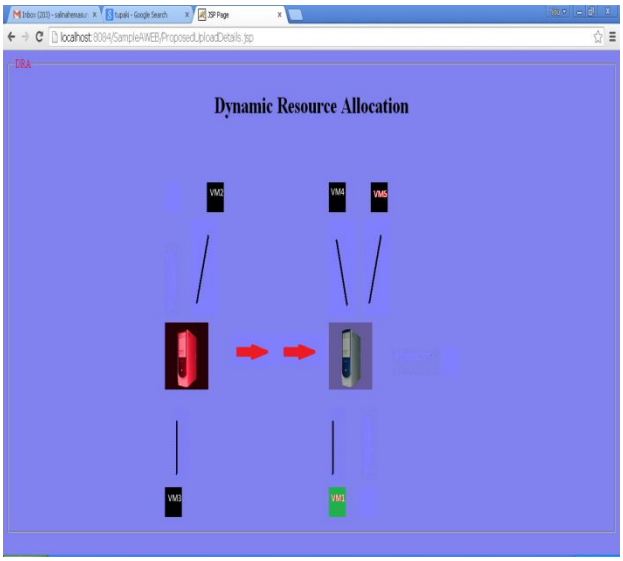


**Figure 4 – Dynamic resource allocation**

As seen in Figure 4, it is evident that VM migration took place. Initially there were 3 VMs with first physical machine. As one of the VMs experienced over-commitment and there was no possibility of extending resources for VM, it is migrated to other physical machine where resources are available. The migrated VM is shown in green color.

## IV. CONCLUSION

In this paper we studied the dynamic resource allocation in cloud computing environment. Since the resources are commercially shared by cloud in pay per use fashion, there is much research went in this area. From the literature it was understood that the existing algorithms focused on the dynamic resource allocation with different approaches. However, in this paper our focus is to propose an algorithm for mitigating over commitment of virtual machines. We consider some physical resources on which multiple VMs are running. The proposed algorithm named Overload Mitigation algorithm that combines two possible cases for optimizing resource allocation. The first case is, when a VM over commitment is witnessed, the VM's capacity gets increased by using either free memory of PM or idle memory of other VM. The second case is that a VM migration technique is employed so as to move over-committed VM to other PM. Thus the proposed algorithm achieves optimal resource allocation dynamically. We built a prototype application, a custom simulator, which demonstrates the proof of concept. The empirical results reveal the fact that the proposed algorithm is useful and can be used in real world cloud applications. An important direction for future work is to apply the algorithm to real cloud and perform empirical study to know its effectiveness.

## V. REFERENCES

[1] M. Armbrust *et al.*, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep., Feb 2009.

[2] "Amazon elastic compute cloud (Amazon EC2), http://aws.amazon.com/ec2/."

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. of the Symposium on Networked Systems Design and Implementation (NSDI'05)*, May 2005.

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

[5] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker, "Usher: An extensible framework for managing clusters of virtual machines," in *Proc. of the Large Installation System Administration Conference (LISA'07)*, Nov. 2007.

[6] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. Of the Symposium on Networked Systems Design and Implementation (NSDI'07)*, Apr. 2007.

[7] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in *Proc. of*

*the ACM European conference on Computer systems (EuroSys'09)*, 2009.

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

[9] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in *Proc. of the Symposium on Operating Systems Design and Implementation (OSDI'08)*, 2008.

[10] T. Sandholm and K. Lai, "Mapreduce optimization using regulated dynamic prioritization," in *Proc. of the international joint conference on Measurement and modeling of computer systems (SIGMETRICS'09)*, 2009.

[11] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proc. of the ACM/IEEE conference on Supercomputing*, 2008.

[12] R. Nathuji and K. Schwan, "Virtualpower: coordinated power management in virtualized enterprise systems," in *Proc. of the ACM SIGOPS symposium on Operating systems principles (SOSP'07)*, 2007.

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

[14] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: augmenting network interfaces to reduce pc energy usage," in *Proc. of the USENIX symposium on Networked systems design and implementation (NSDI'09)*, 2009.

[15] T. Das, P. Padala, V. N. Padmanabhan, R. Ramjee, and K. G. Shin, "Litegreen: saving energy in networked desktops using virtualization," in *Proc. of the USENIX Annual Technical Conference*, 2010.

[16] Y. Agarwal, S. Savage, and R. Gupta, "Sleepserver: a software-only approach for reducing the energy consumption of pcs within enterprise environments," in *Proc. of the USENIX Annual Technical Conference*, 2010.

[17] N. Bila, E. d. Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan, "Jettison: Efficient idle desktop consolidation with partial vm migration," in *Proc. of the ACM European conference on Computer systems (EuroSys'12)*, 2012.