International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - EEECOS 2014, ISSN : 2348 2273
Available Online at www.ijeecse.com

Active Resource Allocation Management in Virtual Cloud Instances

E. Srinivasa Rao, Y. C. Ashok Kumar, K. Siva Rama Krishna.
Department of Computer Science and Engineering, Andhra Loyola institute of engineering and technology, Vijayawada
ycashokkumar@gmail.com,srinu.215@gmail.com

Abstract: ***Cloud computing supports enterprises to scale up and down their resource usage based on requirements along with popular opinion. One might say many of the perceived and interesting advantages Our uses in the cloud computing domain are inspired from resource multiplexing using virtualization technology with some side effects with respect to load management of resources prompting us to prescribe the concept of "cloud skewness" to estimate the anomalies and uneven factors in the multi-dimensional resource utilization of a virtual server instances. As a solution in this paper, we propose a cloud architecture that implements virtualization technology to allocate enterprise data center resources dynamically based on growing/shrinking application demands and dynamically controlling the number of active servers involved in the process. By reducing skewness factor, we can integrate different types of workloads nicely in queues and improve the overall utilization of server assets. Further we develop and use a combination of heuristic policies that can prevent overload in the system. Powered by real time cloud host simulations combined with simulation traces along with experimental results demonstrate that our proposed optimized algorithm achieves good performance.***

Keywords: Cloud computing, Cloud Resources, Green computing, Load Predictions, Virtual Instances.

I. INTRODUCTION

Cloud computing [1] is the next generation in computation. It is obviously the next natural step ahead in the evolution of on-demand IT services and products.

Near future possibly many users can have everything they need on the cloud from small apps to large hardware driven computational resources. Cloud Computing consolidates itself through rapid development and deployment of an increasing number of distributed applications in the form of cloud clusters. It can scale existing infrastructure on demand within minutes or even seconds, instead of days or weeks, thereby avoiding under-utilization (idle servers) and over-utilization a broad array of web-based services aimed at allowing users to obtain a wide range of functional capabilities on a 'pay-as-you-go' basis. Cloud computing platforms, such as those provided by Amazon EC2, Google, Microsoft, IBM, and HP, let developers deploy applications across computers/servers hosted by a central organization. These applications can manage and access a large network pool of computing resources that are deployed and managed by above mentioned cloud computing provider. It really is accessing resources and services needed to perform functions with dynamically changing needs. Simple the cloud is a virtualization of resources that maintains and manages itself.

Developers obtain the benefits of a managed computing platform, without having to dedicate resources to design, build, deploy and maintain the infrastructure. Yet, an important problem that must be highlighted in the cloud is how to manage QoS requirements and maintain SLA for cloud users that share cloud resources. The cloud computing technology enables the resource as a single point of access to the end user and is implemented as pay per usage. Though there are various benefits in cloud computing such as complete virtualized environment, prescribed and abstracted infrastructure, equipped with dynamic infrastructure, free of software and hardware installations and maintenance, pay as per consumption. An important concern is the order in which these requests are satisfied thus scheduling should be made in such a way that the resource should be utilized efficiently. This problem evolves the scheduling of the resources. This optimal allocation of resources should maximize the system utilization and overall performance. Cloud computing is sold on demand on the basis of time constraints basically specified in minutes or hours.

There are three primary classes of cloud computing service models (Fig. 1):

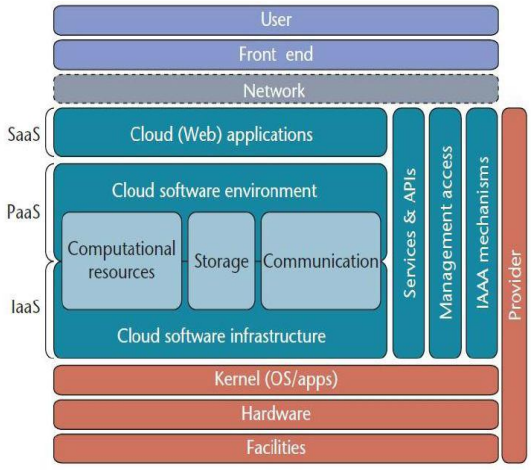

Fig. 1: Cloud Computing Architecture

In infrastructure as a service (IaaS) cloud service, a cloud based virtual server providing networking, computational and mass storage services and other infrastructure services is deployed. The client does not manage or control the enterprise data centre but may have control over the data or different operating systems placed into the infrastructure. For instance, Amazon web service (AWS). In platform as a service (PaaS) cloud service, the service levels are designed and offered in such a way where a computable platform upon which the user

can host and develop applications and services by using programming languages and API's of their choice and of course is supported by service provider. In this medium the user can control the deployed applications and sometimes even the application-hosting environment as well. However, the infrastructure (servers, OS, storage) is still in the control of the cloud provider. Examples include Google App Engine Framework and Windows Azure.

Finally Software as a service (SaaS) cloud service, where applications are running on a cloud infrastructure or platform which is accessible via a thin client interface (browser) or program interface.

The user only has the possibility to manage some application specific settings, because the cloud service provider don't accommodate cloud features; they only provide applications that are hosted and running in the cloud. SaaS is a parallel alternative to having the software run on remote machines and some good examples are online CRM systems (Sales Force CRM),online office applications such as (Google Docs), web mailing services (Google Mail) and even Social Network Sites (Facebook, Twitter) [5]. Resource Allocation (RA) is the process of allocating available resources to the up and running cloud applications over the web. Resource allocation introduces load problems and drains services if the allocation is not managed properly.

Resource provisioning solves that problem by allowing the service providers to manage the resources for each individual module. Resource Allocation Strategy (RAS) concentrates on integrating cloud service provider activities for utilizing and allocating important sparse resources within the limitations of cloud infrastructure so as to meet the needs of the deployed cloud application. It requires the stats pertaining to the type and amount of resources needed by each hosted application in order to complete a user's task.

II. RELATED WORK

Qiang Li and Qinfen Hao et al [2] prescribes cloud architecture, using feedback control theory, for adaptive management of virtualized resources, which is based on VM. This architecture suggests virtualization technology involves pooling all hardware resources into common shared space in cloud computing environment so that the deployed application can access the required resources as per their need to meet Service Level Agreement standards (SLA's) of the service provider. In the core of this virtualization technology resides the adaptive manager used mainly as multi-input multi-output (MIMO) resource manager, which further includes 3 handlers: CPU handlers, memory handlers and I/O handlers and their combined goal is to regulate multiple virtualized resources utilization within the cloud to uphold SLAs of application by using control inputs per Virtual Machine CPU, memory and I/O allocation.

Walsh et al. [3], suggested a general two-tier architecture that implements utility functions, that are adopted in the context of dynamic and autonomous resource allocations, with local agents and global arbiter being sub components. The responsibility of local agents is to estimate or forecast utilities, for a given workload or a range of resources, and later the results are transferred to the global arbiter. The global arbiter computes optimized allocation of resources based on the inputs obtained from the local agents. Jiayin Li et al[4], proposed an adaptive resource allocation algorithm for the cloud environment with pre-emptable tasks in which the system implements the resource allocation adaptively based on always changing actual task executions.

Some examples of these adaptive resource allocation algorithms are Adaptive list scheduling (ALS) and adaptive min-min scheduling (AMMS). Their main task includes static task scheduling, for static resource allocation, which is generated during offline. The online adaptive procedure constantly re-evaluates the remaining static resource allocations with respect to a pre defined threshold.

Resource allocation strategies based on distributed multi-criteria decisions in cloud is explained in [6]. According to it, the contributions of the Resource allocation are tow-fold, initially a distributed architecture is adopted where resource management is further divided into independent tasks, each of them will be performed by Autonomous Node Agents (NA) in a cycle of three major activities: (1) VM Placement where a suitable physical machine (PM) is found that is capable of running given VM's and then assign VM's to that PM, (2) Monitoring(MM) the total resources used by the hosted VM are monitored by Node Agents, (3) VM selection in which if a local accommodation is not possible, a VM need to migrate to another PM and process fits back to into placement and continues working. Finally using PROMETHEE method, NA can carry out the configurations in parallel through multi criteria decision analytics. This procedure is potentially more feasible in large data centers than in a centralized approaches. A real time example being Hadoop architecture.

III. PROPOSED SCHEME

In the proposed work, we present an architecture that uses virtualization technology to allocate data center resources dynamically based on application demands and optimizing further using Stochastic estimations for load predictions in the total number of servers in use.

A. System Overview:

The architecture of the system is detailed in the following Figure 2. The figure designates each physical machine (PM) that runs the Xen (or) Virtual box hypervisors (VMM) which supports a privileged admin domain 0 and one or more processing domains U [7]. Each VM in domain U encapsulates one or more applications such as control panel, Web server, Application server, Database server, Mailing Apps, Map/Reduce Apps, etc. We assume all PMs share a

backend storage. The multiplex management and migration of VMs in PMs is managed using the Usher framework prescribed in [8].

The main logic of the proposed architecture is implemented as a set of plug-in. Each node deploys an Usher local node manager (LNM) on domain 0 which accumulates the usage statistics of resources for each VM instance on that node. The statistics accumulated at each PM are then forwarded to the Usher central controller (Usher CTRL) where the original VM scheduler is still running. The VM Scheduler is the invoked at regular intervals and is programmed to receive updates from the LNM regarding the resource demand history of VMs, followed by the capacity and the load history of PMs, and finally the current layout semantics of VMs on PMs. The scheduler has several components mainly the stochastic load predictor that predicts the future resource demands of VMs and the future load of PMs based on past statistics. We estimate the load experienced by a PM by aggregating the resource usage of its individual VMs.

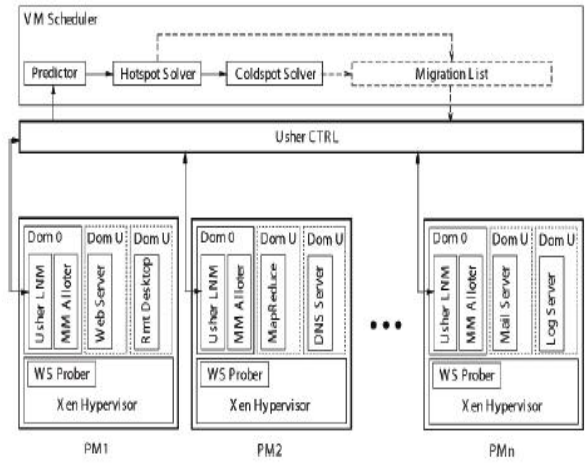

Fig. 2: System Architecture

The LNM at each node agent first tries to complete the new demands locally by adjusting the resource allocation of VMs sharing the same Hypervisor. The MM Allotter on domain 0 of each node agent is responsible for modifying the local memory allocations of individual VM's. The hot spot solver in our Hypervisor Scheduler checks if the resource consumption of any PM is above the prescribed hot limit (i.e., a hot spot). The cold spot solver detects if the average consumption of actively used PMs (APMs) is below the prescribed cold limit thus prompting the relocation to much needy VM's.

Skewness Algorithm with Stochastic estimations:

We implement the "skewness" factor to measure the imbalances in the multiplexed resource utilization of a PM. By reducing skewness, we can transfer resource between different types of workloads between different VM's nicely and improve the overall consumption of all PM's resources. Let n in question be the number of resources and ri be the consumption of the i-th resource.

We put forward the resource skewness of a server p as

$$\text{Skewness(p)} = \sqrt{\sum_{i=1}^{n}\left(\frac{r_i}{r}-1\right)^2}$$

Our algorithm is invoked periodically to check the resource allocation load status based on the predicted future resource demands of each VMs to determine if they grow or shrink. We define a PM as a hot spot if the consumption of any of its resources is above a pre-defined hot limit calculated based on overall resource statistics present in PM's. We define the CPU utilization of a hot spot p as the square sum of its resource consumption beyond the defined hot threshold:

$$\text{Temperature (p)} = \sum_{r\in R}\left(r - r_t\right)^2$$

where R is the combination of overloaded resources in a server p and r_t is the hot threshold factor for resource r. We define a VM instance as a cold spot if the consumptions of all its resources are below a cold threshold. This indicates that the VM instance is mostly idle and a potential candidate to switch off or hibernate to save energy. Finally, we define the a balanced warm threshold factor to be at level of resource utilization reading that is sufficiently high enough to justify having the VM instance running but not so high to risk becoming a hot spot in the face of temporary peaks of an application resource demands. Considering high computational time of plain Skewness Load Prediction algorithm we propose to use eager prediction based load balancing algorithm combined with Stochastic estimations for dynamic resource provisioning.

Prediction-based Load Balancing

```
1: nodes ← Servers with utilization statistics
2: while n ∈ nodes do
3:    if n is triggered then
4:       triggers ← triggers ∪ {n}
5:    end if
6: end while
7: Sort the imbalance scores of triggers in descending order.
8: while n ∈ triggers do
9:    if attractiveness(n) > threshold then
10:      nodeToMigrate ← node
11:   end if
12: end while
13: dest ← smallestLoad(nodeToMigrate)
```

Results on multi core processors justified this performance tweak in terms of efficiency in implementation parallel processing.

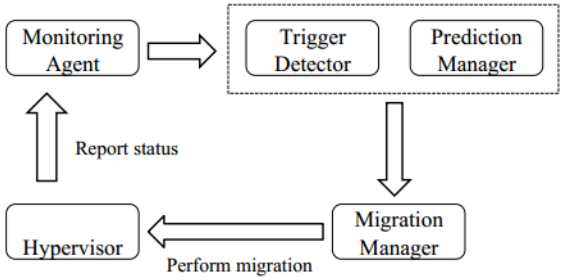

Fig. 3: Optimized System Architecture

Hotspot Mitigation:

We first identify and then sort the list of hot spots in the VM instances in descending CPU reads (i.e., we handle the hottest one first). Our aim is to reduce all hot spots if possible by re-allocation of resources. Otherwise, keep their CPU consumption as low as possible. For each PM say p, we initially decide which of its VMs resources should be migrated away to other VM's. We sort its primary list of VMs based on the resulting core CPU usage of each VM if that VM is migrated away. We attempt to migrate the VM that can reduce the load on PM's the most. In case of similar estimates, we select the VM whose migration can reduce the skewness of the PM the most. For each active VM in PM's list, we seek to find a destination PM that can accommodate it. The PM must not become a hot spot after welcoming this VM. It is important to note that this reduction can either be negative which means we select the PM whose skewness increases at the least. If a destination PM is found, we track and record the migration of the current VM to that PM and update the predicted shared load of concerned servers. If not, we move on to the next active VM in the list and explore a destination PM for it. As long as we are able to find a destination PM for any of its VMs, we consider this sort of functioning of the algorithm largely a success and then move on to find the next hot spot. Note that each execution of the algorithm migrates away at most one VM instance from the overloaded PM. This definitely does not by any means eliminate the hot spot, but at least attempts to reduce its temperature. If it remains a hot spot in the next algorithm invoke, the algorithm will go on repeating the mentioned process.

IV. Conclusion And Futurework

Cloud Computing has proven to be an ideal technology where computing services are offered over the internet with resources elsewhere, with this sort of on-demand practices elastic resources like computing power, memory tweaks, storage capacities and network virtualization offers an efficient solution to achieve the objectives laid out in the cloud computing paradigm by implementing creation, maintenance and migration of Virtual Machines (VMs) over the underlying physical machines(PMs), that ultimately leads to a more improved resource consumption and abstraction process. In this paper, we proposed a system that uses the concept of virtualization technology aided with some form of hypervisor to allocate enterprise data center resources dynamically based on application growing/shrinking demands and support smooth communications by optimizing the number of PMs in use. We introduce this concept of "skewness" with stochastic estimations to measure the imbalances in the multiplexed resource utilization of a PM. By reducing this skewness factor, we can club together different types of workloads nicely between different VMs and improve the overall PMs Utilization of resources. Exploration of live virtual machine migration technologies can be considered as a possible research that can further load reduction.

V. References

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

[2] "Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control," in First International Conference on Information Science and Engineering, April 2010, pp. 99-102.

[3] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility Functions in Autonomic Systems," in ICAC '04: Proceedings of the First International Conference on Autonomic Computing. IEEE Computer Society, pp. 70-77, 2004.

[4] Jiayin Li, Meikang Qiu, Jian-Wei Niu, Yu Chen, Zhong Ming, "Adaptive Resource Allocation for Preempt able Jobs in Cloud Systems," in 10th International Conference on Intelligent System Design and Application, Jan. 2011, pp. 31-36.

[5] P.T.Jaeger, J.Lin, and M. grimes, Cloud computing and information policy: Computing in a policy cloud? Journal of Information Technology and politics, 2009.

[6] Yazir Y.O., Matthews C., Farahbod R., Neville S., Guitouni A., Ganti S., Coady Y., "Dynamic resource allocation based on distributed multiple criteria decisions in computing cloud," in 3rd International Conference on Cloud Computing, Aug. 2010, pp. 91-98.

[7] 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," in Proc. of the ACM Symposium on Operating Systems Principles (SOSP'03), Oct. 2003.

[8] 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.