International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 12 Issue 2 –JANUARY 2015.

Vibrant Source Distribution for Cloud Computing Ambiance Using Virtual Machines

S Chiranjeevi[1] and D Rajesh Babu[2]

[1]*M.Tech CSE, Intell Engineering College, Affiliated to JNTUA University, ANANTAPURAMU, A.P., India*

[2]*Assistant Professor, CSE, Intell Engineering College, Affiliated to JNTUA University, ANANTAPURAMU, A.P, India*

***Abstract*— Virtualization might provide all features in data centers by switch on virtual machine migration to delete hotspots. We provide Sandpiper, a system that done the task of maintaining and capturing hotspots, finding a mapping of physical to virtual services and initializing the necessary migrations. As virtual command on Internet resources and services considerably increases in last 5 years, the vigour used by information centers, directly r linked to the number of hosted servers and their workload, is also been increasing. The problem for a cloud model is how can a cloud service provider best multiplex its virtual resources onto the physical hardware?. Most of the servers in many existing data centers are often severely underutilized due to over provisioning for the peak demand. Because of this, a cloud model offers a regular scale up and down in response to load variation. Sandpiper implements a black-box approach that is fully OS- and application-agnostic and a gray-box approach that exploits OS- and application-level statistics. We apply our techniques in Xen and conduct a detailed evaluation using a mix of CPU, network and memory-intensive applications. Our results show that Sandpiper is able to resolve single server hotspots within 20 seconds and scales well to larger, data center environments. We also show that the gray-box approach can help Sandpiper make more informed decisions, particularly in response to memory pressure.**

Keywords: Virtualization, green computing, resource utilization, virtual resources, hotspots.

I. INTRODUCTION

Virtualization data centers allow consolidation of one or multiple applications and sharing of multiple resources among all applications. Virtualization is process of disruptive change in enterprise data canters and giving rise to a new paradigm: shared virtualized infrastructure. In this new paradigm [3], many enterprise applications share dynamically allocated services and resources. These applications are also consolidated to decreases infrastructure and operating costs while simultaneously increasing resource utilization. Operating system virtualizations have attracted considerable interest in last 5 years, particularly from the data center and cluster computing communities. It have formerly shown that paravirtualization allows many OS instances to run simultaneously with in a single physical machine with more high performance, providing better useful of physical services and resources and isolating individual OS instances. The major objectives for virtual resources import complex resource and service level objectives, changing resource requirements over every time, distribution resource allocation, and resource dependencies. Since resources can now be dynamically adapted based on actual demand, the resource over-provisioning approach is wasteful. Current virtualization technologies are inadequate in achieving complex service level objectives (SLOs) for enterprise applications with time-varying multiple resource demands.

In this paper, we provide an adaptive algorithm to achieve following two goals as below:

Overload avoidance: The capacity that of a physical machine must be efficient to satisfy the resource needs of all VMs running on it.

Green computing: The number of PMs used must be minimized as long as they can still satisfy the needs of all VMs. Idle PMs can be turned off to save energy.

For achieving a balance between two goals, we develop a resource allocation system to avoid overload within the system, the technique of skewness to measure the minimum utilization of server.

II. EXISTING SYSTEM

Virtual machine monitors (VMMs) works as similar to Xen a mechanism for mapping virtual machines (VMs) to the physical resources. This mapping is largely hidden from the cloud users. A user with the Amazon EC2 service, for example, does not know where their VM instances have been run. It is totally related to cloud provider to make sure the underlying physical machines (PMs) has sufficient resources to meet their needs. VM live migration technology makes it possible to change the map between VMs and PMs While applications are running. The capacity of PMs can also be different because multiple generations of hardware coexist in a data center.

DISADVANTAGES OF EXISTING SYSTEM:

• A policy issue residue as how to decide the mapping adaptively so that the resource demands of VMs have met

while the number of PMs used have been minimized. • This is challenging when the resource needs of VMs are heterogeneous due to the diverse set of applications they run and vary with the time as the workloads grow and shrink. The two main disadvantages are overload avoidance and green computing.

Dynamic provisioning that dynamically turns mainly on a minimum number of servers required to the needs for satisfying the application specific quality of service. Load dispatching distributes current load among the running machines. Dynamic provisioning and load dispatching are two interdependent techniques. The Collective project [3] has been previously explored VM migration as a tool to provide mobility to the users who works on different physical hosts at different timings, citing as an example the transfer of an OS instance to a home Personal computer while a user drives from work home. Their work aims to optimize for slow (e.g., ADSL) links and longer time spans, and so stops OS execution for the duration of the transfer, with a set of enhancements to reduce the transmitted image size. In contrast, our efforts are alarmed with the migration of live, in-service OS instances on fast networks with only tens of milliseconds of downtime. Other projects that have explore migration over longer time spans by stopping and then transferring include Internet Suspend/Resume [4] and μDenali [5].Zap [6] uses partial OS virtualization to allow the migration of process domains (pods), essentially process groups, using a modified Linux kernel. Their approach is to isolate all process-to-kernel interfaces, such as file handles and sockets, into a contained namespace that can be migrated. Their approach is significantly faster than results in the Collective work, largely due to the smaller units of migration. However, migration in their system is still on the order of seconds at best, and does not allow live migration; pods are totally suspended, copied, and then resumed every time. Furthermore, they do not address the problem of maintaining open connections for existing services. The live migration system presented here has considerable shared tradition with the previous work on NomadBIOS [7], a virtualization and migration system built on top of the L4 microkernel [8]. NomadBIOS uses pre-copy migration to achieve very short best-case migration downtimes, but makes no attempt at adapting to the writable working set

The cloud model besides reducing the hardware cost [2] should also save on electricity which contributes to a significant portion of the operational expenses in large data centers. Virtual machine monitors (VMMs) like Xen provide a mechanism for mapping virtual machines (VMs) to physical resources. Xen is a freely available open source VMM for commodity hardware. In Xen, we consider the migration of active OSes hosting live services, it is critically important to minimize the downtime during which services are entirely out of stock. Secondly, we must consider the total migration time, during which state on both machines is coordinated and which hence may be affected the factor reliability.

VMware provides OS migration support, dubbed VMotion, to their Virtual Centre management software. As this is commercial software and strictly disallows the publication of third-party benchmarks. Moving the original contents of a VM's memory from one physical host to an additional can be approached in any number of ways. If it is a live service it must balance both downtime and total migration time. Iteratively scanning and sending a VM's memory image between two hosts in a cluster could easily consume the entire bandwidth available between them and hence starve the active services of resources. This service degradation will occurred to some extent during any live migration scheme. Also, a key challenge in managing the migration of OS instances is what to do about the resources that are associated with the physical machine that they are migrating away from.

In this paper, we propose a work in which we present a system that uses virtualization technology to allocate data center resources dynamically based mainly on application demands and it also supports green computing by optimizing the number of servers in use. We introduce the concept of "skewness" to symbolize the unevenness in the multidimensional resource utilization of a server. By minimizing skewness, we can join different types of workloads adequately and improve the overall utilization of server resources.

III. PROPOSED SYSTEM

In this paper, we present the design and implementation of an automated resource management system that achieve a good balance between the two goals. Two goals are overload avoidance and green computing.

1. Overload avoidance:

The capability of a PM should be enough to satisfy the resource needs of all VMs running on it. Otherwise, the PM is overfull and can lead to insolvent performance of its VMs.

2. Green computing:

The number of PMs used should be minimized as long as they can still satisfy the needs of all VMs. Idle PMs can be bowed off to save energy.

ADVANTAGES OF PROPOSED SYSTEM:

We make the following contributions: We build up a resource allocation system that can avoid overload in the system efficiently while minimize the number of servers used.

We introduce the concept of "skewness" to measure the uneven utilization of a server. By minimizing skewness, we can improve the overall utilization of servers in the face of multidimensional resource constraints.

We designed a load prediction algorithm .it could capture the future services and resource usages of applications accurately without seeing the inside VMs. The algorithm can

capture the rising trend of resource usage patterns and help reduce the placement churn significantly.

The architecture of the proposed system is given a sin Fig 1: Each PM runs the Xen hypervisor (VMM)[1] which supports a changed 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 assuming a backend storage. 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 collect the usage figures of resources for each VM on that node. The CPU and network usage can be calculated by monitoring the scheduling events in Xen [2]. The guest OS is required to install a break up swap partition. A working set prober (WS Prober) is implemented on each hypervisor to estimate the working set sizes of VMs running on it. The VM Scheduler is invoked periodically and receives from the LNM the resource demand the past of VMs, the capacity and the load history of PMs, and the current layout of VMs on PMs.

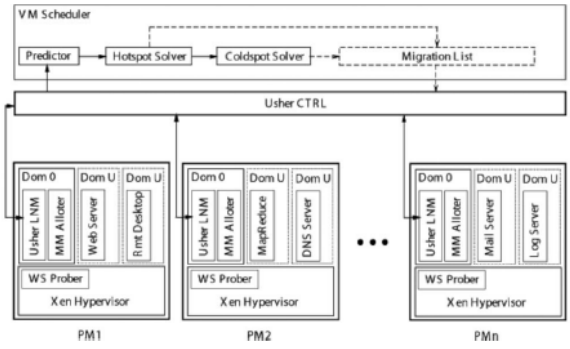


Fig. 1. System architecture.

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 dynamically used PMs (APMs) is below the green computing threshold. If so, some of those PMs might potentially be turned off to save energy.

While estimating the future resource needs of VMs we will calculate an exponentially weighted moving average (EWMA) using a TCP-like scheme given as

$$E(t) = \alpha * E(t-1) + (1-\alpha) * O(t), 0 \le \alpha \le 1,$$

We use the EWMA formula to expect the CPU load on the DNS server to measure the load every minute and predict the load in the next minute. To mirror the "acceleration," we take an innovative approach by setting α to a negative value. When $-1 \le \alpha \le 0$, the above formula can be transformed into the following:

$$\begin{aligned} E(t) &= -|\alpha| * E(t-1) + (1+|\alpha|) * O(t) \\ &= O(t) + |\alpha| * (O(t) - E(t-1)), \end{aligned}$$

THE SKEWNESS ALGORITHM

The method of skewness is used for quantify the unevenness in the utilization of single and multiple resources in the server. Consider n be the number of services and resources we consider and ri be the utilization of the ith resource. We define the resource skewness of a server p as

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

Where r is the average utilization of all resources for server p. To minimizing the skewness, we should combines different types of workloads and improve the overall CPU utilization of server resources.

Our algorithm performs and executes like clockwork to solve the resource allocation status based on the predicted future services and resource strain of VMs. We define a server as a hot spot if the utilization of any of its resources is the above a hot threshold. This indicates that the server is overloaded and hence some VMs operation on it should be migrated away. We define the temperature of a hot spot p as the square sum of its resource utilization beyond the hot threshold:

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

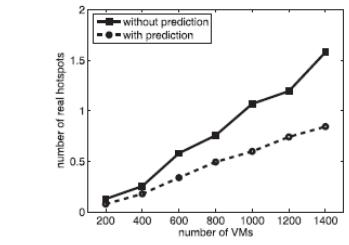
We define a server as a cold spot if the utilizations of all its resources are below a cold threshold. This indicates that the server is mostly idle and a potential candidate to turn off to save energy.

In this work, we aim to migrate away the VM that can reduce the server's temperature the most. In case of ties, we select the VM whose removal can reduce the skewness of the server the most. For each VM in the list, we see if we can find a destination server to put up it. The server must not become a hot spot after accepting this VM. Among all such servers, we select one whose skewness can be reduced the most by accepting this VM. For a cold spot p, we check if we can migrate all its VMs somewhere else. Our green computing algorithm is invoked when the average utilizations of all resources on active servers are below the green computing threshold. We sort the list of cold spots in the system based on the rising order of their memory size. For each VM on p, we try to find a destination server to accommodate it. The resource utilizations of the server after accepting the VM must be below the warm threshold. While we can save energy by consolidating underutilized servers, overdoing it may create hot spots in the future. If we can find destination servers for all VMs on a cold spot, we record the sequence of migrations and update the predicted load of related servers. Otherwise, we do not migrate any of its VMs.

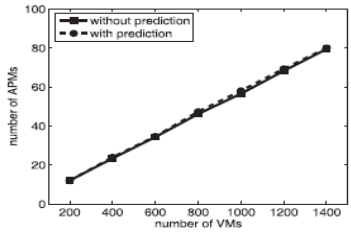
When load prediction is disabled, the algorithm simply uses the last observed load in its decision making. Fig. 2a shows

that load guess significantly reduces the average number of hot spots in the system during a decision run. Notably, prediction prevents over 46 percent hot spots in the simulation with 1,400 VMs. With prediction, the algorithm correctly foresees that the load of the PM will increase above the threshold shortly and hence takes no action.

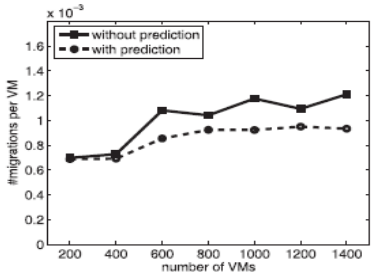
This leaves the PM in the "cold spot" state for a while. Fig. 2b which shows that the average numbers of APMs remain essentially the same with or without load prediction. Fig. 2c compares the average number of migrations per VM in each decision with and without load prediction.



(a) number of hot spots



(b) number of APMs



(c) number of migration

Fig 2: Effect of Load Prediction

IV. CONCLUSION

In this paper, we supported a feed- back control system and AutoControl, to change dynamically allocate computational resources to all applications in shared virtual-ized environments. AutoControl contains of an on-line model estimator that finds the relationship between application-level performance and resource allocation and a novel MIMO resource controller that finds appropriate allocation of multiple resources to achieve application-level SLOs. We solved Au-toControl using a testbed consisting of Xen virtual machines and various multi-tier and single-tier applications and benchmarks. Our experimental results documents confirm that AutoControl may detect CPU and disk utilization across multiple nodes and can adjust re-source allocation to achieve end-to-end application-level SLOs. In addition, AutoControl can achieve shifting resource bottlenecks and provide a level of service differentiation according to the priority of all applications. Finally, we showed that Auto-Control can show the performance targets for different application-level metrics, including throughput and response time

Self-ballooning was the main technique for virtual resources. it automatically t allows the hypervisor to reclaim unused memory. Looking into the performance considerations, the memory consumption is stable over the time. The network utilization stays very smaller.

REFERENCES

[1] Zhen Xiao, Senior Member, IEEE, Weijia Song, and Qi Chen,Dynamic Resource Allocation Using Virtual Machines for Cloud Computing environment, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 6, JUNE 2013

[2] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," Proc. USENIX Ann. TechnicalConf., 2005.

[3] M. McNett, D. Gupta, A. Vahdat, and G.M. Voelker, "Usher: An Extensible Framework for Managing Clusters of Virtual Machines,"Proc. Large Installation System Administration Conf.(LISA '07), Nov. 2007.

[4] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," Proc. Symp. Networked Systems Design and Implementation (NSDI '07), Apr. 2007.

[5] 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," Proc. ACM European conf. Computer Systems (EuroSys '09), 2009.