CPU Resizing Vertical Scaling on Cloud

Sakthi Saravanankumar P., Mahendran Ellappan, and Mehanathen N.

***Abstract*—In a cloud computing, provisioned virtual machines are insufficient for computing the services it needs to be fine tuned by adding the additional resources or boosting the existing resources. A static method of resource allocation becomes inefficient under the different load. Based on the heavy load virtual machine can be adopted for dynamically changes of its behavior. There are two ways of scaling in the cloud scenario as horizontal scaling and vertical scaling. In a horizontal scaling, if the virtual machine resources are suffer from providing a service a new virtual machine was created and launched immediately. A threshold value has been maintained for scale out and scale down the virtual machine. Horizontal scaling is a traditional and well suitable approach for cloud computing environment. But the limitation of this approach is required separate load balancer to distribute the load between the virtual machines. In a vertical scaling resources are boosted by maximizing the virtual machine capabilities without shutting down the virtual machines. It's not like a traditional scaling approach to set the threshold and adding the resources. Here the resources are monitored by certain interval and based on the analysis resources are added to existing virtual machine. But lots of challenges are there in this approach. The limitation of this approach is we can't scale up the virtual machine up to the physical machine capability. Here we presented and analysis our approach for increasing the CPU capability of virtual machine. Likewise we can increase other resources like memory, vDisk and bandwidth. presented architecture resides on top of the virtual machine monitor and acts based on the scheduling algorithm.**

***Index Terms*—Cloud computing, vertical scaling, horizontal scaling.**

I. Introduction

Cloud computing technology shows a rapid growth in the field of information technology and provides an efficient and flexible way to access resources over internet. This includes on demand services, attracts industries and organizations. Cloud uses virtualization technologies for managing resources and provides customized environment as needed by the user. There were various definitions for cloud around the world and [1] list some of the basic definitions.

Scaling is defined as increasing or decreasing the capacity (i.e. ram, CPU, file system, bandwidth, etc…) of resources for better performance. Scaling is categorized into two types namely Horizontal and Vertical scaling. Horizontal scaling is scale-out the computing power by creating new virtual machines, and vertical scaling is scale-in increasing the computing power (i.e. ram, cpu, ..) of the existing virtual machine. Vertical scaling facilitates the users to increasing the performance on-demand. Dynamic scaling of resources is feasible by using virtualization technology. In general horizontal scaling refers to increasing or decreasing the no of virtual machines, when the loads of the services are varied. In horizontal scaling a load balancer is used for communication overheads. Vertical scaling refers to increase the effective "sizes" of individual virtual machine servers [2]-[4]. By increasing or decreasing the computing power of virtual machine is necessary for reducing the communication overhead. The focuses on the vertical /horizontal scaling approach for reducing the minimum turnaround time and for improving the cost efficiency [5].

Eucalyptus [6] is the cloud middleware for managing the IaaS cloud resources. It is organized in a hierarchical manner such as cloud controller, cluster controller and node controller. The cloud controller is responsible for handling the user requests and the cluster controller is managing the node controllers. The node controller is responsible for creating and destroying the virtual machines. This study proposes a vertical/horizontal scaling mechanism for different resource scaling situations in cloud.

The rest of the paper is organized as follows: Section II discusses the related work to the proposed work. Section III describes the proposed architecture and functions of internal components; Section IV describes the implementation and design details. Section V describes the experimental results and the experimental carried out on our proposed work. Section VI concludes the proposed work and explores the possibilities for future work.

II. Related Work

CPU resizing is an important factor of vertical scaling. It is a not like a traditional approach to maintain the threshold value and made changes in the cpu size depends upon the behavior of the value. Here CPU scheduler plays on important role for cpu resizing. Already scheduler was implemented such a way that balancing the cpu between the virtual machine. For our experiments we have used a xen virtual machine monitor. Xen is an open source virtualization monitor. It is referred as a para-virtualization hypervisor to manage the un-privileged domain (guest) from the most privileged domain (dom0). Responsibility of hypervisor includes memory management and cpu scheduling of running virtual machines [7]. Xen has many types of scheduler to manage the virtual machine running on cpu. The major scheduler (from xen version > 3.0) was Barrow Virtual Time (BVT), Simple Earliest Deadline First, (SEDF), ARINC653 scheduler and Credit scheduler [8]. All the CPU schedulers implemented with some polices like proportional

Manuscript received October 11, 2014; revised December 15, 2014.

The authors are with Center for Development of Advanced Computing, Chennai, India (e-mail: {sakthis, mahendrane, mehanathenn}@cdac.in).

share (PS) scheduler. PS scheduling polices guarantees that each virtual machine obtains certain percentage of cpu cycles for computing its task. Generally cpu schedulers are maintain the following attributes for scheduling. These are shares, limits and reservation. A virtual machine was assigned twice as many shares as another virtual machine, it consumes twice as many cpu cycles. Another attribute limit is used to restrict the physical resource usage of virtual machine. Also it ensures that virtual machine doesn't get more cpu cycles rather than its allotted. Reservation is guaranteed for reserve the resources of the shared environment [9]. Usually scheduling algorithm operate at two modes called pre-emptive (wc), non-pre-emptive (nwc). In wc (work conserving) mode if the two virtual machines are shared the cpu, one goes to the idle state then another one consume the entire cpu. In nwc (non work conserving) mode even the shared virtual machine goes for idle another one virtual machine does not allow getting the remaining idle cpus. Below, we briefly describe the above algorithms. BVT (Borrowed Virtual Time) is a fair-share scheduler, dispatching the runnable virtual machine with the earliest effective virtual time. The scheduler is accounting a running time in terms of minimum charging unit (mcu) typically the frequency of clock interrupt and the scheduler is configured with a context switch allowance C. It provides low latency support for real-time application by allowing low latency sensitive clients to wrap back in virtual time to gain scheduling priority. It then effectively borrows the virtual time from the future cpu allocation of virtual time. Each BVT includes a state variable called effective virtual time and actual virtual time. BVT is a pre-emptive works at work conserving mode only [10]. The Simple Earliest Deadline First (EDF) scheduler sets each domain to run for an n milliseconds slice every m milliseconds. SEDF is a fairness scheduler it depends on value of the period. The values of n and m are configurable by the administrator on a per-domain basis. It chooses the VCPU which has the closest deadline. SEDF works on both WC and NWC modes. It doesn't support global load balancing on multi-processor [11]. The Credit scheduler is default scheduler. In the scheduler each domain has two main properties associated with it, a weight and a cap. The weight determines the share of physical CPU time that the domain will get. The cap is mainly the maximum of CPU time the domain can get. It is widely configurable for the administrator and work-conserving [12], [13]. ARINC653 scheduler is a periodically repeating fixed time slice scheduler. The primary goal of this scheduler is isolating of domains. Each virtual machine has been assigned to single pCPU. It has two time frames. Overall scheduling time frame is major frames. It contains many minor frames to execute the task. CPU pool support also implemented in xen. But multicore support is not available [14]. Based on the evaluation of the above scheduler we take credit scheduler for our experimentation. It has global load balancing support on multicore architecture. Dynamic Scaling of Cloud Applications proposes an availability-aware policy by performing both vertical and horizontal scaling to explore how and where to allocate computing resource [15].

III. Proposed Architecture

The proposed cloud architecture for cloud environment for scaling is depicted in Fig. 1.

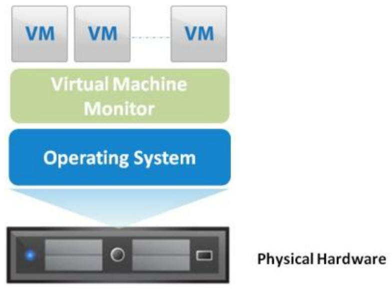

Fig. 1. Cloud architecture.

A. *Cloud Infrastructure*

The cloud has installed with Eucalyptus 4.X Middleware, euca2ools and XEN Hypervisors installed in Nodes. The physical infrastructure of cloud is given in Table I.

TABLE I: Cloud Infrastructure

No of Servers	Processor	Ram (in GB)	CPU Cores	HDD
Server 1	Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz	32	48	1 TB
Server 2	Intel(R) Xeon(R) CPU X5460 @ 3.16GHz	16	8	500 GB

The proposed architecture for vertical scaling in Cloud environment is depicted in Fig. 2. The proposed architecture is deployed over the cloud site for scaling. The architecture mainly consist CPU Pool Management, Physical Core Controller, VM Introspection, VM Scheduling & Exception Handler.

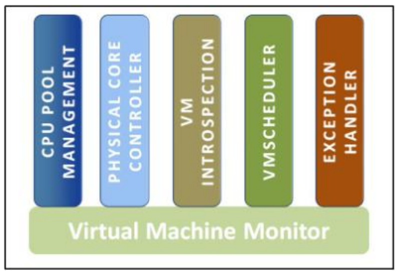

Fig. 2. Vertical scaling in cloud environment architecture.

B. *CPU Pool Management*

It is a CPU affinity; which dedicates a specific pool for a VM. It has a set of CPU pools and maintains the information of the number of domains that can run in a particular pool; it also has the information of the physical CPU which is grouped. The available physical CPU are grouped to a particular CPU pool, pcpu's are grouped by 2 powers N (2^n). In runtime the CPU cores can be added/*removed* from the pool. A scheduler is used to share the pCPU of the Virtual machines within the CPU pool.

C. *Physical Core Controller*

It resides in the virtual machine monitor. It is used to restrict the usage of each and every virtual machine CPU cycles in the CPU pool. Depends on the increase of the no of virtual Machines in the CPU pool, the CAPS value of credit scheduler can be adjusted and the availability of virtual machines and the virtual machines application without loss of generality.

D. VM Introspection

It monitors the following components CPU Pool Management, Physical core controller and VM Scheduling; it also monitors the state of the running virtual machines, CPU usage & demand of the CPU of a specific Virtual Machine.

E. VM Scheduling

It is the important component in the architecture. The running virtual machine CPU cores can be reduced / expanded based on the information from the VM introspection component from the specific CPU pool. If the virtual machine vcpu has been increased more than configured the cpu pool the VM gets migrated to another possible CPU pool.

F. Exception Handler

The role of exception handler is to handle the load of virtual machines sharing the same pCPU by many of the vCPU of different VM Fig. 3 illustrated. By default sharing of pcpu reduces the cpu cycles and availability for running the VMs. Due to this VMs didn't get the cpu cycles fully. This leads to performance degradation of the VMs.

IV. Implementation

In our implementation the CPU pool is created using CPU pool utility in XL tool stack. Initially the host system vCPUs are shared all the physical CPU. The host system requirements vCPUs are grouped and pinned into one pool then the host system will be moved into that pool. The pCPUs are unplugged and grouped into pools, based on the request of the virtual machines. Once the CPUs are grouped and named the components in our architecture will maintain all the information's of CPU pool such as time of creation, no of VMs in the CPU pool, allocation of vCPUs from the pCPUs. Physical core controller restrict the virtual machine CPU cycles. The value can be adjusted by the cap and weight value of XEN credit scheduler. Based on monitoring system status the cap and weight value can be changed. Initially it set by 0 it works as a WC mode and consumes all the cpu cycles. Libvirt package is used to interact with the hypervisior. It has the ability to manage virtual machines on XEN. The request made by eucalyptus is converted into xen managed configuration through libvirt api interface.

Introspection of particular VM can be done provide by various tools like xentop and xenmon. It will get the information based on some time slice value produced. VM scheduling is the most important component which is used to migrate the VM between the pool. Frequent switch of VM migration can be affect the performance of the physical machine CPUs. When VM is not sufficient to adopt the particular pool VM can be migrated to some other pool by provided the xl utility cpupool-migrate. All the components are implemented and run on top of the XEN hypervisor.

In this paper cpu resizing is not depending on the threshold value. It is purely depends on the scheduling on the vCPUs. Over a period of time the components has been monitoring a vCPUs consumption cpu cycles. By default the scheduler gives equal priority for each threads (vCPUs) running on pCPU. When there is more sharing by vCPUs, automatically scheduler reduces the cpu cycles of each virtual machine running on the pCPUs.

The main factors for scaling cpu virtually are, initial number of virtual cpu should be defined for launching virtual machine. The maximum no of virtual cpu required for scaling at running state. The constraints to be followed for resizing of vCPU. The total number of vCPU can't be exceeded the total number of pCPU in SMP host (NpCPU >= NvCPU). The number of vCPU shared by the pCPU increases a risk of reducing the cpu cycles. For example the following figure illustrates the VM1 of vCPU2, VM2 of vCPU2 and VM4 of vCPU1 shared the pCPU2. Here scheduler gives equal priority for all the running virtual machine. In case any virtual machine has a weight value as double as comparatively with other virtual machine, it gets the twice as cpu cycle. The remaining virtual machine automatically gets reduced cpu cycles. Another vCPU depends on this vCPU the process speed automatically reduced by itself. So here vCPU has been boosted by twice to get full cpu cycle. The weight value can be adjusted and regulated based on the cpu cycles. Equally this process can be monitored.

Load balancing between virtual cpus depends on weight, the weight range start from 1 to 65535 and the default is 256. A domain with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a guest machine. The cap optionally fixes the maximum amount of CPU a domain will be able to consume, even if the host system has idle CPU cycles. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc... The default, 0, means there is no upper cap.

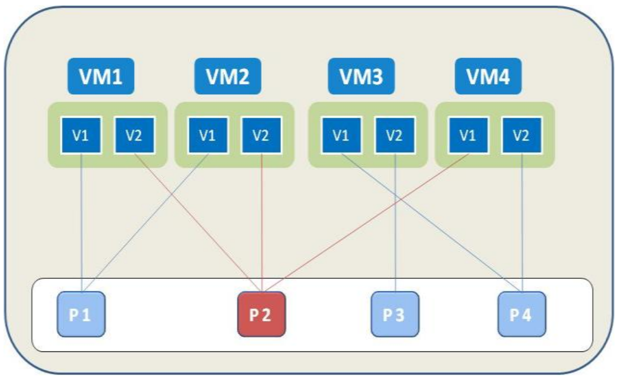

Fig. 3. Multiple vCPU shared single pCPU.

V. Experimental Results

Creation of virtual machines in cloud, our main objective and in our experiment we focused on the scaling and CPU resizing of machines. Any virtual machine is launched it should have the configuration parameter max vCPUs value is equivalent to maximum physical processor value. Once the value is fixed we cannot upgrade the CPU level. Our components are executed on top of the XEN hypervisor CPU pools are created based on the calculation of 2^n. Once the pool was created virtual machines are moved based on the availability of vCPUs. The virtual machines are created and the load increased to the VM using dd (disk write utility) which consumes more CPU. Then on-demand the cap value can be adjusted and additional vcpu can be given by physical core controller.

TABLE II: WEIGHT AND CAP VALUE OF CREDIT SCHEDULER

Domain	Priority	CAP
Guest 1	256	No
Guest 2	256	No
Guest 3	512	60%

From the Table II guest3 gets more cpu cycle rather than guest1 and guest 2. From our experiment initially we are providing the instruction of disk block (block size 10MB) write of each virtual machine with four threads. All the virtual machine consumes 98.0 % of cpu cycles running on different physical core. In guest3 additionally we are instructed four more threads and increase the load of cpu. Now the weight value has been adjusted and provide more cpu cycles (194.0%), even though the getting chance of cpu utilization is very less only. This scenario stats that vcpu also has been resized. Now the efficient utilization of cpu cycles guest3 vcpu has been increased and the virtual machine has been moved into high capability pool. All these process are automated by using python and shell script language. But we are facing some issues in different machines. Sometimes frequent context switching happened in a single core system. We observe that for single core earlier credit scheduler itself produce great performance.

VI. CONCLUSION

In this paper, we propose a vertical scaling and CPU resizing in cloud infrastructure using virtualization technology. In our experiment, we provide a definite approach for running virtual machines over cloud. In this vertical scaling, a variety of CPUs are grouped together and pinned for peak conditions. In our future work, this proposed vertical scaling and CPU resizing can be extended for various types of application in cloud environment.

ACKNOWLEDGMENT

The authors sincerely thank the Department of Electronics & Information Technology, Ministry of Communication and Information Technology, Government of India for financially supporting the Centre for Development of Advanced Computing.

REFERENCES

[1] Twenty Experts Define Cloud Computing. [Online]. Available: http://www. cloudcomputing .syscon.com/read/612375_p.htm.

[2] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. S. Yousif, “Autonomic resource management in virtualized data centers using fuzzy-logicbased approaches,” *Cluster Computing Journal,* vol. 11, 2008.

[3] 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. the 4th ACM European Conference on Computer Systems*, 2009, pp. 13–26,

[4] S. Blagodurov, D. Gmach, M. Arlitt, Y. Chen, C. Hyser, and A. Fedorova, “Maximizing server utilization while meeting critical SLAs via weight-based collocation management,” in *Proc. the International Symposium on Integrated Network Management*, 2013, pp. 277-285.

[5] Y. Liu, M. R. Shie, Y. F. Lee, Y. C. Lin, and K. C. Lai, "Vertical/horizontal resource scaling mechanism for federated clouds," in *Proc. the 2014 International Conference on Information Science and Applications (ICISA)*, vol. 1, no. 4, May 2014, pp. 6-9.

[6] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud computing system,” in *Proc.* the *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 124 –131.

[7] Wikipedia. [Online]. Available: http://www. en.wikipedia.org/wiki/Xen

[8] Scheduling. [Online]. Available: http://www. wiki.xen.org/wiki/Scheduling_in_Xen

[9] Reservations and CPU Scheduling. [Online]. Available: http:/www. /frankdenneman.nl/2010/06/08/reservations-and-cpu-scheduling

[10] K. J. Duda and D. R. Cheriton, “Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler,*”* in *Proc. the Seventeenth ACM Symposium on Operating Systems Principles (SOSP '99),* New York, NY, USA, pp 261-276, 2013.

[11] M. A. G. Giraldo, “Xen scheduling with load balancing on speed,” Master in Computer Engineering Project Report, Faculty of Computer Science Complutense University of Madrid, 2010.

[12] L. Cherkasova, D. Gupta, and Vahdat, “Comparison of the three CPU schedulers in Xen,” *SIGMETRICS Perform. Eval*, Rev. vol. 35, no. 2, 2007, pp. 42-51.

[13] Credit Scheduler. [Online]. Available: http://www. wiki.xen.org/wiki/Credit_Scheduler

[14] ARINC653 Scheduler. [Online]. Available: http://www. wiki.xenproject.org/wiki/ARINC653_Scheduler.

[15] W. T. Wang, H. P. Chen, and X. Chen, "An availability-aware virtual machine placement approach for dynamic scaling of cloud applications," in *Proc. the 2012 9th International Conference on the Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, 2012, pp. 509-516.

Sakthi Saravanakumar P. is working as an engineer at C-DAC, Chennai. He started his career in development of software application in the health care domain. Initially he worked in the platform of SaaS development. Later he is working in the cloud middle-ware development. He is pursing M.Tech in computer science and engineering. His major research interests include virtualization and cloud computing.

Mahendran Ellappan working as a senior engineer at CDAC, Chennai. He has a unique passion towards research and development. He pursued bachelor of engineering in computer science in 2004. He spent over 8 years of experience in grid and cloud computing. His major research areas include neural networks, virtualization grid computing and cloud computing. contributed. he has R&D publications towards international journals and conferences which include a wide area of grid computing, virtualization and cloud computing.

Mehanathen Nesamony holds a master of computer applications from Bharathidhasan University. He started his career as a project associate in IIT Madras, was working with TeNet (telecommunication networking group) for providing Internet connectivity and portals for the rural kiosks which has poor connectivity. His next assignment was with a private company Netlink Technologies (Unimity Solutions) worked with content management systems on open source technologies. Currently. He is working as senior engineer in Center for Development of Advanced Computing. He is working on Open Source technologies like PHP, MySql & Grails. His major research areas include cloud computing and foss technologies.