International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 15972-15978

Distributed Memory and CPU Management in Cloud Computing Environment

P.V.S.S. Gangadhar [1]
Research Scholar & Scientist-D at National Informatics Centre (NIC),
Department of Information Technology,
GITAM Institute of Technology (GIT), GITAM University,
Visakhapatnam, Andhra Pradesh, India.
Orcid: 0000-0002-8548-8492

Dr. M. Venkateswara Rao [2]
Professor, Department of Information Technology,
GITAM Institute of Technology (GIT), GITAM University,
Visakhapatnam, Andhra Pradesh, India.
Orcid: 0000-0002-7598-3473

Dr. Ashok Kumar Hota [3]
Scientist-F at National Informatics Centre,
Ministry of Electronics & Information Technology (MIETY),
Govt. of India, NIC-OSU, Bhubaneswar, Odisha, India.
Orcid: 0000-0002-9117-1504

Dr. V. Venkateswara Rao [4]
Professor, Department of Computer Science and Engineering,
Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh, India.
Orcid: 0000-0003-0131-4944

Abstract

To efficiently utilize the resources in a virtualized environment; they need to be over committed. Over commitment is the process of allocating more resources to a virtual machine that is physically present on the host. It is based on the principle that the majority of the Virtual Machines will use only a small percentage of the resources allocated to them at a given time. Nevertheless, circumstances may happen when the resources required by the virtual machines are more than the physical resources present on the machine. The performance of the Virtual Machines will be severely impacted in these situations. The majority cloud providers usually have Service Level Agreements (SLAs) with their clients and hence cannot allow virtual machines to deliver a poor quality of service.

In this paper, we first study troubles that relate to the capacity planning required when setting up applications into a virtualized data center. We show how models of virtualization overheads can be employed to precisely foresee the resource needs of virtualized applications, allowing them to be easily transitioned into a data center. We subsequently study how memory similarity can be used to direct placement when adding virtual machines to a data center, and exhibit how memory sharing can be exploited to reduce the memory footprints of virtual machines. This let for improved server consolidation, sinking hardware and power costs within the data center. Finally we investigate the problem of over commitment of the CPU and memory resources. We recommend a distributed resource scheduler (DRS) which uses methods such as memory ballooning and virtual machine live migration to solve the problems in CPU and memory over commitment. We design architecture for distributed resource scheduler (DRS) that is horizontally scalable and illustrate the techniques involved in monitoring and memory ballooning characteristics of the distributed resource scheduler (DRS).

Keywords: Cloud Computing, Virtualization, Virtual Machine Image, Distributed Resource Scheduler, Distributed Memory, Memory Management, Memory ballooning, virtual machine live migration.

INTRODUCTION

Modern data centers are comprised of tens of thousands of servers, and perform the processing for many Internet business applications. Data centers are increasingly using virtualization to simplify management and make better use of server resources. With the advent of large scale cloud computing, the users can get compute resources on demand

with flexible pricing models. Cloud vendors pool their massive hardware resources and provide virtual machines on top of it to the users. To best utilize the resources of a virtualized cloud infrastructure, resource over commitment is used. Allocating more virtual resources to a machine or a group of machines than are physically present is called resource over commitment. Since most applications will not use all of the resources allocated to them at all the times, most of the resources of a cloud provider will remain idle without over commitment. Hence, this approach is more profitable and less wasteful. Orthogonal to over commitment is the fact that cloud vendors have to satisfy some SLAs (Service Level Agreements) which they have with the users i.e. the promised resources should be available to the users whenever they need it. Distributed resource scheduling (DRS) is used to meet these SLAs. This paper talk about the issues faced by these huge data centers, and explains how virtualization can offer pioneering solutions.

Background and Motivation

Internet and business applications are increasingly being moved to large data centers that hold massive server and storage clusters. Present data centers can hold tens or hundreds of thousands of servers, and strategies are previously being completed for data centers investing in excess of a million servers [65]. A few data centers are placed to run applications for a solitary company, such as the search-engine groups run by Google. Supplementary data centers are controlled by service providers that are able to rent storage and computation resources to other customers at very low cost due to their large scale. Cloud computing, that refers to hosting platforms that rent data center resources to customers, is becoming increasingly popular for running Internet websites or business applications. In all of these data centers, the massive amounts of computation power required to drive these systems results in many challenging and interesting distributed systems and resource management problems. Virtualization promises to dramatically change how data centers operate by breaking the bond between physical servers and the resource shares granted to applications. Virtualization can be used to "slice" a single physical host into one or more virtual machines (VMs) that share its resources. This can be helpful in a deploying environment where clients or applications do not require the full power of a single server. In this contect, virtualization provides an easy way to isolate and partition server resources. The concept layer sandwiched between the Virtual Machine and its material host also let for superior control over resource management.

The CPU and memory allocated to a virtual machine can be dynamically attuned and live- migration methods let VMs to be transparently moved between physical hosts without impacting any running applications. As data centers continue to deploy virtualized services, new problems have emerged such as determining optimal VM placements and dealing with virtualization overheads. At the same time, virtualization allows for new and better solutions to existing data center problems by permitting for quick, elastic resource provisioning. The central theme of this paper is to explore how virtualization can allow for application agnostic solutions when dealing with challenges related to application deployment, resource management, and reliability. In particular, we attempt to reply the subsequent questions:

- How can we change applications running on resident hardware to virtual machines while guarantee they take delivery of enough resources regardless of virtualization overheads?
- On what servers must we position new Virtual Machines in categorizing to find the greatest level of server consolidation?
- How can we powerfully manage server resources regardless of extremely varying application workloads?
- How can we successfully connect and manage numerous data centers?
- How can we ensure application reliability regardless of unexpected disasters that can bring down entire data centers?

The data-center atmosphere creates these challenges particularly difficult since it requires solutions with high scalability and extreme speed to respond quickly to fluctuating Internet workloads.

RELATED WORK

Auto-Ballooning in Xen

Xen is a popular open-source, bare-metal hypervisor which was developed by University of Cambridge Computer Laboratory in 2003 and was the first hypervisor to support paravirtualization. Support for full vritualization was later added to it. Xen has autoballooning feature which works via the autoballoon driver that exists in the Linux kernel. Autoballooning implementation in our work has some key differences to autoballooning in Xen, which have been discussed later. It is important to understand both techniques for comparison. To understand Xen hypervisor's method of autoballooning, it is first essential to understand transcendent memory in Linux.

Transcendent Memory

Transcendent memory (tmem) [13] is a type of memory which the linux kernel cannot directly enumerate, track or directly address, but helps in more efficient utilization of memory by a single kernel or load-balancing of memory between multiple kernels in a virtualized environment. The implementation of tmem is divided into two parts - frontend and backend.

Frontend provides the interfaces for different types of data which can be stored provided by tmem to the kernel, while backend is the underlying implementation of storage/retrieval methods. Two basic operations provided by the frontend are 'put' and 'get'. If the kernel wants to save some data into tmem, it uses the 'put' operation while 'get' is used to retrieve the data.

Tmem Frontends

There are two tmem frontends in the Linux kernel which cover two major types of kernel memory - anonymous pages and file backed pages.

Cleancache: Cleancache is used for storing pages which are backed by files on a disk. A kernel can choose to reclaim such pages at the times of memory pressure. These pages are evicted from memory. If the same page is to be used again, a page fault happens and it is fetched from the disk. Before evicting such a page, the kernel can choose to store it in cleancache. If the operation succeeds, the page can possibly be reclaimed from tmem at any later time, otherwise it has to be reclaimed from disk.

Frontswap: Frontswap is used for storing swap pages. Linux swap subsystem stores anonymous pages in a swap device when it needs to evict them.

Tmem Backends

There are multiple backends for tmem - zcache, transcendent memory for Xen, and RAMster. Tmem was originally developed for Xen with Xen backend. The other backends were created later. For autoballooning, we are only concerned with the Xen backend.

Frontswap Self Shrinking

When kernel swaps out a page, it assumes that the page will go to disk and may remain there for long time even if it is not used again as kernel assumes disk space is less costly and abundant.

Auto-Balloon Mechanism

Autoballooning in Xen requires transcendent memory. In each guest, a autoballoon driver is present. A thread of the driver runs periodically in some fixed time interval which sets the target size of the guest.

Target=Committedpages+Reservedpages+BalloonreservedpagesTarget=Committedpages+Reservedpages+Balloonreservedpages

Committed pages are the pages which are used by some process and may reside either in memory or swap. Reserved pages are the pages reserved from normal usage by the kernel. Balloon reserved pages form the memory reserved by the balloon driver to provide some extra memory for the kernel caches and some room to grow for any process that may demand more memory in the future. It defaults to 10% of the total memory size of the guest. If the target is less than the current RAM, the guest is ballooned down, else it is ballooned up. There is a hysteresis counter which represents the number of iterations it will take for the machine to balloon down to target. So, each time self-ballooning process runs,

1) Memory=Current−Current−TargethysteresiscounterMemory=Current−Current−Targethysteresiscounter
2) Hysteresis counter is different for ballooning up and down. There is also amin usable MB parameter, below which machines cannot be ballooned. the consideration of the risks (the value of the economic effects of the implementation of the innovation is proportional to the degree of the risk: the higher the risk, the higher the potential impact of the innovation).

Memory Management in VMware ESX

VMware was founded in 1999 when it launched its proprietary hypervisor which used binary translation techniques for x86 virtualization. They later released an enterprise class bare metal hypervisor called VMware ESXi. ESX has very robust memory management and it introduced several novel techniques as early as 2003 which are still being used and have been implemented in other platforms.

Memory Reclamation Techniques

ESX uses several memory reclamation techniques.

Content-Based Page Sharing: In a virtualized environment, several guests might be running common OS and some common applications which means that there would be lots of pages having the same content. Instead of keeping separate copies of these pages for separate guests, the duplicate pages are deleted and only one copy of the page is kept which is marked Copy-On-Write(COW). When any of the guests attempts to write to that page, a new copy of the page is created by the kernel which is then modified. Linux kernel has KSM(Kernel Same-Page Merging) [15] which performs the same task and is used in virtualization by QEMU-KVM.

Memory Ballooning:

Page compression: In this case, the content of the page is compressed and stored. When the page is accessed, its content is decompressed.

Demand Paging: ESX server has a swap daemon which handles hypervisor level swapping. The swap daemon gets the target swap levels of each VM from the swapping policy and selects the pages that need to be swapped.

Memory Reclamation Policies

The ESX server defines four memory states depending upon which it employs the memory reclamation techniques. .

High: More than 6% of the total memory of the hypervisor is free at this point. Only page sharing is employed in this state.

Soft: Free memory is between 6% and 4%. Page sharing is active. Balloon driver also starts reclaiming idle memory..

Hard: Free memory is between 4% and 2%. The hypervisor now aggressively starts reclaiming memory through swapping. Page compression also becomes active. So, on page reclamation, if it is compressible or shareable, it is compressed/shared otherwise it is swapped out.

Low: Below 1% free memory. Along with memory reclamation, ESX also blocks any new memory allocation by any guest.

VIRTUALIZATION AND RESOURCE MANAGEMENT

With the advent of large scale cloud computing, the users can get compute resources on demand with flexible pricing models. Cloud vendors pool their massive hardware resources and provide virtual machines on top of it to the users. To best utilize the resources of a virtualized cloud infrastructure, resource over commitment is used. Allocating more virtual resources to a machine or a group of machines than are physically present is called resource over commitment.

Since most applications will not use all of the resources allocated to them at all the times, most of the resources of a cloud provider will remain idle without over commitment. Hence, this approach is more profitable and less wasteful. Orthogonal to over commitment is the fact that cloud vendors have to satisfy some SLAs (Service Level Agreements) which they have with the users i.e. the promised resources should be available to the users whenever they need it. Distributed resource scheduling (DRS) is used to meet these SLAs.

Virtualization

Virtualization is one of the driving technologies behind IaaS (Infrastructure as a Service). Virtualization makes it possible to run multiple operating systems with different configurations on a physical machine at the same time.

To run virtual machines on a system, a software layer called hypervisor or virtual machine monitor (VMM) is required. The hypervisor has the control of all the hardware resources and can take away resources from one VM to give it to another. The hypervisor also maintains the state of all the VMs at all the times. It does these by trapping all the privileged instructions executed by the guest VM and emulating the resource they access. The hypervisor is responsible for emulating all the hardware devices and providing proper resource isolation between multiple machines running on the same physical machine.

There are three different techniques used for virtualization [1] which mainly differ in the way they trap the privileged instructions executed by the guest kernel.

Full Virtualization with Binary Translation: In this approach, user mode code runs directly on CPU without any translation, but the non-virtualizable instructions [2] in the guest kernel code are translated on the fly to code which has the intended effect on the virtual hardware.

Hardware Assisted Full Virtualization: To make virtualization simpler, hardware vendors have developed new features in the hardware to support virtualization. Intel VT-x and AMD-V are two technologies developed by Intel and AMD respectively which provide special instructions in their ISA (Instruction Set Architecture) for virtual machines and a new ring privilege level for VM. Privileged and sensitive calls are set to automatically trap to the VMM, removing the need for either binary translation or paravirtualization. It also has modified MMU with support for multi level page tables [3] and tagged TLBs.

Paravirtualization. This technique requires modification of the guest kernel. The non-virtualizable/privileged instructions in the source code of the guest kernel are replaced with hyper calls which directly call the hypervisor [4]. The hypervisor provides hyper call interfaces for kernel operations like memory management, interrupt handling, and communication to devices. It differs from full virtualization, where unmodified guest kernel is used and the guest OS does not know that it is running in a virtualized environment.

Hypervisors can be bare-metal hypervisors or hosted hypervisors. A Bare-metal hypervisor runs directly on the physical hardware while the hosted hypervisor runs on top of conventional operating systems. There are several hypervisors available in the market with VMWare ESX and Xen [4] being the popular bare-metal hypervisors, while KVM-QEMU [5, 6] being a popular hosted hypervisor which runs on top of the Linux operating system. KVM is a kernel module providing support for hardware assisted virtualization in Linux while, QEMU is a user space emulator. KVM uses QEMU mainly for emulating the hardware [7]. So, both these pieces of software work together as a complete hypervisor for linux. KVM-QEMU and Xen are open source while ESX is proprietary. For the purpose of this paper, we will refer to KVM-QEMU wherever hypervisor is used unless specified otherwise.

Virtualization provides a number of benefits other than resource isolation, which makes it the fundamental technology behind IaaS.

It provides the ability to treat disks of virtual machine as files which can be easily snapshotted for backup and restore.

It provides ease of creation of new machines and deployment of applications through pre-built images of the file system of the machine.

Virtual machines can be easily migrated or relocated if the physical machines may require maintenance or develop some failure.

Ease in increasing the resource capacity (RAM or CPU cores) of the machine at runtime by CPU or memory hotplug [8], or otherwise.

Since the hardware resources are emulated by the hypervisor, there is an opportunity for over commitment of CPU and memory resources here.

Memory Overcommitment and Ballooning

In memory overcommitment, more memory is allocated to the virtual machines(VM) than is physically present in hardware. This is possible because hypervisors allocate memory to the virtual machines on demand. KVM-QEMU treats all the running VMs as processes of the host system and uses malloc to allocate memory for a VM's RAM. Linux uses demand paging for its processes, so a VM on bootup will allocate only the amount of memory required by it for booting up, and not its whole capacity.

On demand memory allocation in itself is not enough to make memory overcommitment a viable option. There is no way for the hypervisor to free a memory page that has been freed by the guest OS. Hence a page once allocated to a VM always remains allocated. The hypervisor should be able to reclaim free memory from the guest machines, otherwise the memory consumption of guest machines will always keep on increasing till they use up all their memory capacity. If the memory is overcommitted, all the guests trying to use their maximum capacity will lead to swapping and very poor performance.

There exists a mechanism called memory ballooning to reclaim free memory from guest machines. This is possible through a device driver that exists in guest operating system and a backend virtual device in the hypervisor which talks to that device driver. The balloon driver takes a target memory from the balloon device. If the target memory is less than the current memory of the VM, it allocates (current−target)(current−target) pages from the machine and gives them back to the hypervisor. This process is called balloon inflation. If the target memory is more than the current memory, the balloon driver frees required pages from the balloon. This process is called balloon deflation. Memory ballooning is an opportunistic reclamation technique and does not guarantee reclamation. The hypervisor has limited control over the success of reclamation and the amount of memory reclaimed, as it depends on the balloon driver which is loaded inside the guest operating system.

Virtual Machine Live Migration

VM live migration [11] is a technique to migrate a running VM from one host to another without shutting it down. Live migration involves migrating the disk, memory and CPU states of the running VM to the destination host and resuming the VM there. Migrating disk can be as easy as just copying the disk to destination or using a file-system shared over the network like Network File System (NFS). There are two popular techniques for migrating the VM memory and state:

Pre-Copy Live Migration: It follows an iterative page copying technique wherein first, all the pages of the VM are copied to the destination. From next iteration onward, only the pages which were dirtied during the previous iteration are copied to the destination. This process continues till the page dirtying speed is less than the page transfer speed. Then the VM at the source is stopped, remaining dirty pages and VM state is copied to the destination, and the VM is resumed at the destination. QEMU-KVM uses pre-copy live migration [12].

Post-Copy Live Migration: The VM is stopped at the source, its state is copied to the destination, and the VM resumed there. The pages of the VM are transferred to the destination in background, with the pages that are immediately demanded by the VM via page fault given the highest priority in transfer. Thus, the performance of the VM is degraded before its working set is transferred.

The migration time of a VM is the time taken to resume the VM on the destination after triggering the migration. There is also a small amount of downtime involved in live migration in which the VM is neither running on the host, nor on the destination. Live migration of VMs connected to a network is especially tricky because the new VM has to assigned the same IP address and all the packets have to rerouted to the new destination without any delay to prevent packet loss and downtime of any service running inside the VM which uses the network.

Resource Management in Cloud

Resource management is an essential technique to utilize the underlying hardware of the cloud efficiently. The role of the resource manager is to manage the allocation of physical resources to the virtual machines deployed on a cluster of nodes in a cloud. Different resource management systems may have different aims depending upon the needs. For a private cloud like in an educational institution, the most common aim might be to maximize performance of the virtual machines while minimizing the operational costs of the cloud infrastructure.

Minimizing operational costs involves minimizing the number of physical machines used. This can be achieved through overcommitment of resources. Resource overcommitment comes with some new problems like hotspot elimination and where to schedule new incoming VMs to minimize chances of

hotspot. If the total capacity of the virtual machines running on a physical machine is more than the total capacity of the physical machines, a situation may arise wherein the VMs may want to use a sum total of more resources than are present. Not satisfying those resource requirements may lead to violation of SLAs and poor performance of the VMs. This situation is called a hotspot. A distributed resource scheduler (DRS) is a piece of software that runs on different nodes in the cluster and handles dynamic resource allocation to different VMs in the cluster.

Memory ballooning and live migration of VM can help in mitigating hotspots. The basic idea is that if a VM is short on memory, ballooning can be used to take away some memory from another guest on the same host which has some free memory and give it to the needy guest. If none of the guests have any free memory, the host is overloaded. A guest has to be migrated from this host to another host while taking into account the overall load of the cluster. This might sound simple, but there are several challenges involved in this process. Some of the challenges are determining the amount of free memory a VM can give away without affecting its own performance, determining whether benefits of migration are more than performance loss, selecting which virtual machine to migrate such that maximum benefit is achieved out of the migration, selecting destination host to minimize the chances of future migrations, filtering intermittent spikes from resource usage graph of VMs to determine their actual load profile and distributed monitoring of VMs which can scale to a large number of machines. In this paper, we address large scale monitoring and ballooning of virtual machines.

CONCLUSION

Resource overcommitment is an essential technique for efficient use of resources in a cloud infrastructure. There are several hurdles in overcommitment which have not been addressed completely yet. Most of the studies in this field focus on just a part of the problem, but do not solve it as a whole. The sub problems that have attracted most attention is the optimal placement of virtual machines depending on their demands. But many of these studies do not take the dynamic nature of resource demand of the VMs or the performance degradation during live migration into account. Through this paper, we have tried to solve this problem by taking the whole picture into account, and not just a part of it.

In this paper, we have identified some of the major problems faced in overcommitment of CPU and memory resources in a virtualized environment. We have described some of the relevant work done in this area and their limitations.

REFERENCES

[1] Q. Ali, “Scaling web 2.0 applications using Docker containers on vSphere 6.0,” http://blogs.vmware.com/performance/2015/04/scaling-web-2-0-applicationsusing-docker-containers-vsphere-6-0.html, 2015, (Accessed on 03/21/2016).

[2] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian, “Resource Management in the Autonomic Service-Oriented Architecture,” in Proceedings of the 2006 IEEE International Conference on Autonomic Computing (ICAC 2006), June 2006, pp. 84–92.

[3] J. Anselmi, E. Amaldi, and P. Cremonesi, “Service Consolidation with End-to-End Response Time Constraints,” in Proceedings of 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2008.), September 2008, pp. 345–352.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., “A view of cloud computing,” Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[5] B. Arnold, S. A. Baset, P. Dettori, M. Kalantar, I. I. Mohomed, S. J. Nadgowda, M. Sabath, S. R. Seelam, M. Steinder, M. Spreitzer, and A. S. Youssef, “Building the ibm containers cloud service,” IBM Journal of Research and Development, vol. 60, no. 2-3, pp. 9:1–9:12, March 2016.

[6] M. Assuncao, M. Netto, B. Peterson, L. Renganarayana, J. Rofrano, C. Ward, and C. Young, “CloudAffinity: A framework for matching servers to cloudmates,” in Proceedings of the 2012 IEEE Network Operations and Management Symposium (NOMS 2012), April 2012, pp. 213–220.

[7] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, Discrete-event system simulation. Prentice Hall, 2010.

[8] P. Barham et al., “Xen and the art of virtualization,” in Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), October 2003, pp. 164–177.

[9] L. A. Barroso and U. H¨ olzle, “The case for energy-proportional computing,” Computer, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[10] C. L. Belady and D. Beaty, “Roadmap for datacom cooling,” ASHRAE journal, vol. 47, no. 12, p. 52, 2005.

[11] A. Beloglazov and R. Buyya, “Energy efficient allocation of virtual machines in cloud data centers,” in Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010), May 2010, pp. 577–578.

[12] "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, December 2010, pp. 4:1–4:6.

[13] F. Caglar, S. Shekhar, and A. Gokhale, "A performance Interference-aware virtual machine placement strategy for supporting soft realtime applications in the cloud," Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA, Tech. Rep. ISIS-13-105, 2013.

[14] R. Calheiros and R. Buyya, "Energy-efficient scheduling of urgent Bag-of-Tasks applications in clouds through DVFS," in Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014), December 2014, pp. 342–349.

[15] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," Software: Practice and Experience, vol. 43, no. 5, pp. 595–612, 2013.

[16] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," in Proceedings of the 2011 IFIP/IEEE International Symposium on Integrated Network Management (IM 2011), May 2011, pp. 594–601.

[17] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz, "Energy efficiency for large-scale MapReduce workloads with significant interactive analysis," in Proceedings of the 7th ACM European Conference on Computer Systems, April 2012, pp. 43–56.

[18] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available Google cluster trace," University of California, Berkeley, Tech. Rep., 2010.

[19] P. DELFORGE, "Energy efficiency, data centers—NRDC," http://www.nrdc.org/ energy/data-center-efficiency-assessment.asp, (Accessed on 02/18/2016).

[20] B. H. Li, X. Chai, and L. Zhang, "New advances of the research on cloud simulation," in Advanced Methods, Techniques, andApplications inModeling and Simulation, vol. 4 of Proceedings in Information and Communications Technology, pp. 144–163, 2012.

[21] S. Jafer, Q. Liu, and G. Wainer, "Synchronization methods in parallel and distributed discrete-event simulation," Simulation Modelling Practice andTheory, vol. 30, pp. 54–73, 2013.

[22] R. Fujimoto, A. Malik, and A. Park, "Parallel and distributed simulation in the cloud," SCS Modeling and Simulation Magazine, pp. 1–10, 2010.

[23] A. W. Malik, A. J. Park, and R. M. Fujimoto, "An optimistic parallel simulation protocol for cloud computing environments," SCS M&S Magazine, vol. 4, 2010.

[24] A. J´avor and A. Fur, "Simulation on the Web with distributed models and intelligent agents," Simulation, vol. 88, no. 9, pp. 1080–1092, 2012.

[25] IEEE Std 1516.1-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), Framework and Rules Specification, 2010.

[26] IEEE Std 1516.2-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), Object Model Template (OMT) Specification, 2010.

[27] IEEE Standard, 1516.1-2010—IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification, 2010.

[28] Google, "Google App Engine", (2012), [online]. Available: cloud.google.com [Nov 1, 2012].

[29] Amazon, "Amazon Elastic Compute Cloud (Amazon EC2)", (2012), [online]. Available: aws.amazon.com/ec2/ [Nov 1, 2012].

[30] Microsoft, "Windows Azure.", (2012), [online]. Available:windowsazure.com [Nov 1, 2012].

[31] IBM, "SmartCloud." (2012), [online]. Available: ibm.com/cloudcomputing [Nov 1, 2012].

[32] P. Mell and T. Grance, "The NIST definition of cloud computing(draft)," NIST special publication, vol. 800, p. 145.

[33] A. Desai, "Virtual Machine." (2012), [online]. Available: http://searchservervirtualization.techtarget.com/definition/virtualmachin

[34] 34. VMWare, "vSphere ESX and ESXi Info Center.", (2012), [online]. Available: vmware.com/products/vsphere/esxi-and-esx [Nov 1, 2012].

[35] Microsoft, "Windows Virtual PC.", (2012), [online]. Available: http://www.microsoft.com/windows/virtual-pc/ [Nov 1, 2012].

[36] Xen, "Xen Hypervisor.", (2012), [online]. Available: http://www.xen.org/products/xenhyp.html [Nov 1, 2012].

[37] Microsoft, "Hyper-V Server 2012.", (2012), [online]. Available: microsoft.com/server-cloud/hyper-v-server/ [Nov 1, 2012].

[38] KVM, "Kernel-based Virtual Machine.", (2012), [online]. Available: linux-kvm.org [Nov 1, 2012].

[39] Oracle, "VirtualBox.", (2012), [online]. Available: virtualbox.org [Nov1, 2012]