Performance of Memory Virtualization Using Hybrid Live Migration of Virtual Machines

Pvss Gangadhar, Ashok Kumar Hota, M. Venkateswara Rao, and V. Venkateswara Rao

***Abstract*—Today, Infrastructure-as-a-service providers are trying to minimize the cost of data center operations, while maintaining the Service Level Agreements. This can be achieved by one of the advanced state-of-the-art services of virtualization - the live migration capability. Live migration is defined as the process of transferring an active virtual machine from one physical machine to another without any disconnection. This is achieved by transferring all of the encapsulated states of the VM from one host to another. It has become an essential tool for efficient management of resources in a data center by enabling server consolidation and load balancing. There are two classical migration techniques, namely - pre-copy and post-copy, which employ different memory transfer mechanism during the offloading of a VM. In this paper, we propose a novel hybrid live migration technique by combining the existing pre-copy and post-copy approaches. Compare to its counterparts, our hybrid technique is a fast, efficient and a reliable migration technique.**

***Index Terms*—Cloud computing, IaaS, virtualization, virtual machine, data center, physical memory, load balancing, cluster, resource allocation, writable working set.**

I. Introduction

Virtualization [1], [2] has become one of the key technologies in the era of Cloud Computing. It is loosely defined as an abstraction of the computing resources that can be achieved by either dividing the resources into multiple computing environments or merging various resource components into one. Various concepts and techniques such as time sharing, hardware and software partitioning, simulation, emulation can be used to apply, the division of the resources, Thus, Virtualization technology has enabled the efficient utilization of hardware resources by abstracting away the underlying resources such as processors, main memory, secondary storage and networking. Today, with the help of Virtualization, the data centers are continuously employing the virtualized architecture to execute multiple applications that are mapped on to the physical machines. This has been enabled with the help of virtual machines that form the software abstraction of a physical machine. This abstraction is achieved by the various virtualization techniques [3], [4] such as:

- Full Virtualization: It is defined as the isolated execution of the unmodified guest OS by simulating the hardware resources including full instruction set, input/output operations, interrupts, and memory access.
- Para Virtualization: It is defined as the isolated execution of the guest OS with modified resources in the form of hooks. The para virtualized interface is used to decrease the performance degradation caused by the time spent by the guest in performing certain operations which are substantially more difficult to execute in a virtual environment compared to a non-virtualized environment.
- Hardware Assisted Virtualization: It is defined as the execution of the guest OS with the capabilities provided by the hardware, primarily from the host processor. The resources can be either fully virtualized or para virtualized in this case, Therefore, the cloud providers that are providing the infrastructure resources completely rely on Full Virtualization with hardware assistance. This is because of the client requirements of unmodified guests as a part of the service level agreements (SLAs). Overall, Virtualization helps in enabling agility, dynamism, adaptability [5,6] within a data center. Therefore, VMs form the basic building blocks of the Infrastructure-as-a-Service (IaaS) [6] with benefits such as flexible resource provisioning, monitoring and administration by the system administrators. Resource provisioning allows the provisioning of the infrastructure based resources either at the start or during the life-cycle of a service [7]. At the IaaS level, these resources consist of servers, network, and self-service for the clouds. Server provisioning, a part of resource provisioning, consists of well defined configuration of the servers based on the requirements of the client. It consists of the following two types:

1) Static server provisioning: The configuration of the VMs at the beginning of the life-cycle of a service constitutes the static part of server provisioning. During this process, a physical machine is selected from a pool of physical nodes. Today, a VM is instantiated by using the existing template based provisioning. A physical machine is selected from a pool of physical nodes, during this process, Today, the existing template based provisioning is used to instantiate, a VM [8] before executing other services provisioning depending upon the requirements.
2) Dynamic server provisioning: From both the client's point of view and cloud provider, the dynamic resource provisioning plays an important role in the allocation or

Manuscript received July 17, 2018; revised September 4, 2018.

Pvss Gangadhar is with Gitam University, Visakhapatnam, India (e-mail: pvss.gangadhar@nic.in)

Ashok Kumar Hota is with F, NIC, MEITY, Govt. of India (e-mail: ak.hota@nic.in).

M. Venkateswara Rao is with the Dept. of IT, Gitam University, Visakhapatnam, India (e-mail: mandapti_venkat@yahoo.com).

V. Venkateswara Rao is with the Dept. of CSE, Sri Vasavi Engineering College, Tadepalligudem, India (e-mail: venkatvedula2017@gmail.com).

removal of the resources during the life-cycle of a service. Currently, there are multiple ways to modify the resources either horizontally or vertically.

a) Horizontal scaling: It refers to adding multiple independent physical resources to provide more resources in terms of computing power, memory, disk space and even network bandwidth.

This form of scaling employs multiple instances of the applications running on different physical servers.

b) Vertical scaling: It refers to the addition of the resources on the same physical machine by adding CPUs, memory or even disk for that particular application residing on a single physical instance.

Currently, vertical scaling is not supported by cloud providers since, it requires changes to the guest OS or VMs running on the bare physical machines, which may result in several security issues [9].

A VM, a software implementation of a physical machine, always uses the same physical resources (CPU, memory and I/O) utilized by a physical machine. At the beginning of the life-cycle of a service, the resource-usage profile provided by the client determines initial provisioning of a VM, these profiles try to include the estimations to meet the future load requirements. However, either due to the changes in the workload conditions on VMs or load on the physical machines can lead to “hotspots” - not enough resources to meet the load spikes / demands or “coldspots” – inefficient utilization of the provisioned resources [10]. Thus, to mitigate these issues, live migration of VMs plays an important role in the dynamic resources management of the physical machines inside a data center.

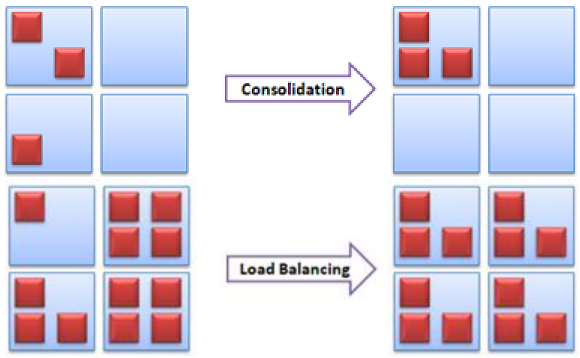


Fig. 1. Load balancing scenario of the virtual machines for equal distribution of the resources.

Live migration [11]-[13] is the process of transferring a running virtual machine from one physical machine to another over the network. This will alleviate the hotspots to meet the SLA guarantee and handle coldspots for the efficient utilization. Therefore, from a cloud provider’s point of view, live migration plays a very important role in the following scenarios:

- Server consolidation: Fig. 1 shows, with the help of server consolidation, we can avoid server sprawl. The VMs residing on a lightly loaded physical machines can be packed onto a fewer physical machines while maintaining the SLA. Not only will this lead to low power usage by the data center but it will also lead to higher resources usage by a host [14].
- Load Balancing: The primary purpose of load balancing is to ensure the equal distribution of the resources resulting in almost equal residual resource capacity across the physical machines (Fig. 1). This can be achieved via live migration of the VMs which will mitigate the resource utilization discrepancy.
- Hotspot mitigation: During the life-cycle service of a VM, if the resource utilization increases for some time window, it will result in hotspot in the future. To mitigate this issue, either additional resources are locally allocated (vertical scaling) or globally i.e. among the physical machines.
- If the local resources are not sufficient, VMs can be migrated to another physical node to offload the resources, thus resulting in hotspot mitigation .
- System maintenance: The physical resources need to be physical repaired 2 replaced or removed to keep the servers running smoothly inside a data center, Thus, during the maintenance, the cloud provider will try to migrate the VMs on to different physical machines while adhering the SLAs [15].

A. *Motivation*

From the above section, it is quite clear that one of the most important services of virtuaization is live migration. From a cloud provider perspective, a VM migration should be transparent enough to have an unnoticeable fast migration by neither exposing the latency to the user nor affecting the collocated VMs on the same physical machine. However, existing migration approaches, such as pre-copy [16] and post-copy [17], are inefficient for highly utilized physical machines inside a data center [18]. These approaches perform quite poorly on workloads with large working set size or even memory intensive workloads. Furthermore, they can incur significant performance overhead by consuming huge amount of network bandwidth. Besides that, the post-copy approach is more efficient than the pre-copy is but does not provide any destination VM guarantee during the migration period. This becomes another concern from a cloud provider perspective.

II. Related Work

A. *Process Migration*

Process migration [19] is defined as the migration of processes from one computing environment to another. With the introduction of clusters of workstations for providing high performance facility, this concept gained momentum Since it was quite possible to transfer files between various machines in a distributed environment, this introduced a demand for dynamic allocation as well as re-allocation of the computational resources by migrating the process’ execution states. With the introduction of distributed computing, process migration showed various benefits such as load balancing and fault-tolerance However, because of the complexity involved in the transparent migration of the processes, it did not achieve widespread use.

B. *Virtual Machine Migration*

A VM is a software implementation of a physical machine which emulates the behavior of a physical machine. A running VM comprising of its states constitute as a process visible to the host OS. This process is a special process where every state information is in an encapsulated format and can be extracted and used, whenever required, by the host OS. This has been made possible by the virtual machine monitor (VMM). Thus VM migration is a special case of process migration where VMM plays an important role in providing the required VM's execution state, hence, bringing transparency to the migration process.

C. *Performance Metrics*

The following metrics can be used to measure the effectiveness of a live migration technique:

1) Performance degradation: The performance degradation of an application running on the VM due to the migration process.
2) Data transferred: The total data transferred during the migration process. Although it is lower bounded by the size of the VM and the device states, it can be larger than that depending on the migration technique used.
3) Total migration time: The time taken to completely finish all the three phases of migration: push, stop-and-copy and pull phases. Since the resources allocated to a VM on the source machine will be completely released only after the total migration time, this is an important metric.
4) Downtime: Downtime is the duration of the stop-and-copy phase. During the downtime, the VM is neither active on the source nor on the destination.
5) Perceivable Downtime: This is the time period during which the VM is unresponsive after resuming execution on the destination. This is a qualitative metric and is only relevant to VMs whose state is externally visible such as network based workloads.

III. Post-Copy Live Migration

In this section, we propose the design of the post-copy approach which tries to minimize the downtime in the pre-copy and the application performance degradation for the workloads with high dirtying rate. The post-copy approach consists of the two approaches - stop-and-copy phase and pull phase. It works as follows:

Stop-and-copy phase: As soon as the migration command is issued, the VM is suspended. Then the VM's states excluding the main memory pages are transferred to the destination. After this, the VM is immediately resumed for execution on the destination, while the older memory States of the VM at the source are present for the demand paging.

Pull phase: As the VM resumes its execution on the target machine, every time it accesses a page that is not yet transferred from the source, a page fault occurs. Then the page fault handler retrieves the corresponding page from the source over the network. Page faults of this kind are called network page faults.

IV. Hybrid Live Migration

In this section, we present the design, implementation and evaluation of the hybrid live migration approach. This approach consists of both the pre-copy and post-copy approaches. The hybrid algorithm utilizes all the three described phases of the process migration steps, namely - push phase, stop-and copy phase and pull phase. Thus, our basic hybrid approach tries to provide the best of two worlds – pre-copy and post-copy, by outperforming the both of these approaches in terms of the total data transfer, total migration time, and the application performance degradation. With the introduction of the push phase, our hybrid approach decreases both of the perceivable downtime as well as the number of network faults that are quite high in case of post-copy approach. In addition, we have o introduced a histogram based learning phase which not only improves the performance metrics but also reduces the resource consumption of the source. This phase is introduced prior to the push phase that assists the migration daemon in restricting the transfer of the writable working set, during the push phase. Depending upon the availability of the CPU and network bandwidth, we incorporate a compression technique using a real-time Compression /decompression technique - as well as parallelize the push phase to utilize the unused bandwidth. We have implemented the hybrid approach on the top of QEMU/KVM and have demonstrated our results on the same.

A. *Hybrid Approach for Live Migration*

In this Section, we discuss the design of the basic hybrid approach along with the introduction of a new phase - learning phase, prior to the push phase and other optimizations incorporated in the push phase. The hybrid approach tries to utilize the network bandwidth optimally by a single transfer of the VM's memory to the destination in the push phase followed by only transferring the device state's along with the dirty bitmap in the stop-and-copy phase and then the dirty pages in the pull phase. Our approach helps us to achieve lesser downtime when compared to the pre-copy technique and drastic reduction in the perceivable downtime when compared to the post-copy approach. We introduce the learning phase which comes prior to the push phase to estimate the WWS.

B. *Basic Hybrid Migration Algorithm*

The various phases of the basic hybrid migration algorithm are as follows:

1) Push phase: The entire memory is transferred to the destination without suspending the VM.

There are no multiple iterations and hence no multiple page retransmissions across iterations.

If all the bytes in a page contain the same value, only single byte is transferred.

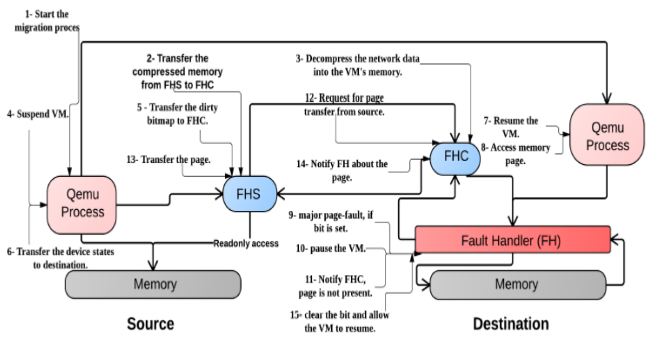
2) Stop-and-copy phase: The VM is suspended and the dirty bitmap along with the device states are transferred to the destination. The dirty bitmap indicates the pages that got dirtied during the push phase. Then the VM is resumed at the destination.
3) Pull phase: Whenever the VM accesses a page that got dirtied during the push phase, a major page fault gets generated which results in a network fault. This network fault is handled by retrieving the corresponding page from the source. The dirty bitmap transferred during the stop-and-copy phase is useful in deciding whether a

page fault can be resolved locally or requires a page transfer from the source.

A page gets transmitted in the push phase and gets retransmitted again in the pull phase, only if it gets dirtied again. No pages get transmitted during the stop-and-copy phase. Therefore, a page gets transmitted at most twice. To minimize the number of pages that get transmitted twice, we introduce a learning phase before the push phase.

```
Algorithm 1 Estimating WWS using adaptive histograms.
1: sum ← 0:0
2: for i ← 1 , nopages
do
3: hist[i] ← αdb[i] + (1 -α )hist[i]
4: sum ← sum + hist[i]
5: end for
6: average ← sum / nopages
7: for i ← 1, nopages
do
8: if hist[i] >= average then
9: wwsapx[i] ← 1
10: end if
11: end for
```

The following diagram shows Virtual Machine Migration flow


1- Start the migration proces
2- Transfer the compressed memory from FHS to FHC
3- Decompress the network data into the VM's memory.
4- Suspend VM.
5 - Transfer the dirty bitmap to FHC.
12- Request for page transfer from source.
7- Resume the VM.
8- Access memory page.
Qemu Process
13- Transfer the page.
14- Notify FH about the page.
FHC
FHS
Qemu Process
9- major page-fault, if bit is set.
Readonly access
10- pause the VM.
Fault Handler (FH)
6- Transfer the device states to destination.
11- Notify FHC, page is not present.
Memory
Memory
15- clear the bit and allow the VM to resume.
Source
Destination
Fig. 2. virtual machine migration flow.

V. Experimental Evaluation and Results

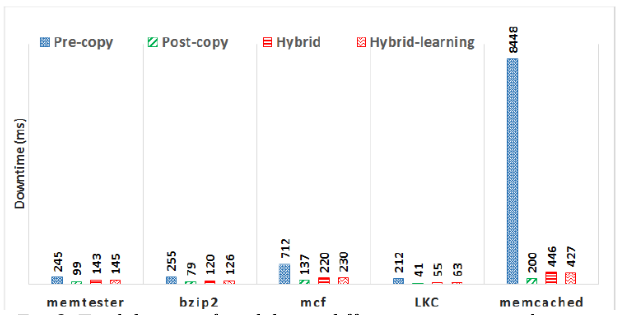


Fig. 3. Total data transferred during different migration mechanisms.

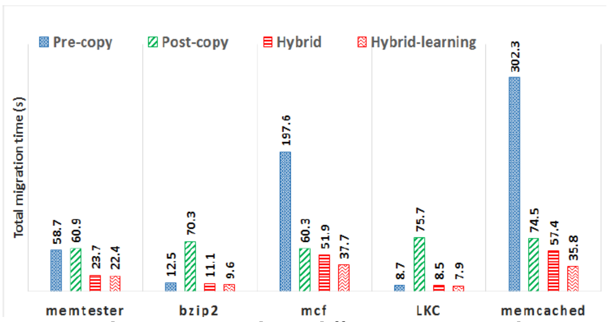


Fig. 4. Total migration time during different migration mechanisms.

Here we present a detailed evaluation of the hybrid migration technique against the existing pre-copy and post-copy approaches. Fig. 3, Fig. 4, Fig. 5 and Fig. 6 shows how pre-copy, post-copy, hybrid and hybrid-learning migration algorithms compare against each other on various workloads. The performance metrics used are application degradation, data transferred, migration time and downtime.

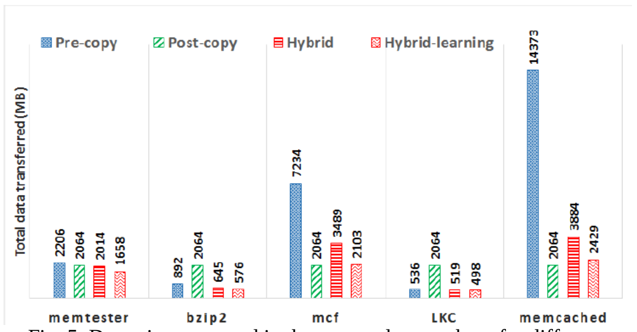


Fig. 5. Downtime occurred in the stop-and-copy phase for different migration mechanisms.

The following diagram shows Total migration time between the compression based hybrid learning and parallelized compression based hybrid learning.

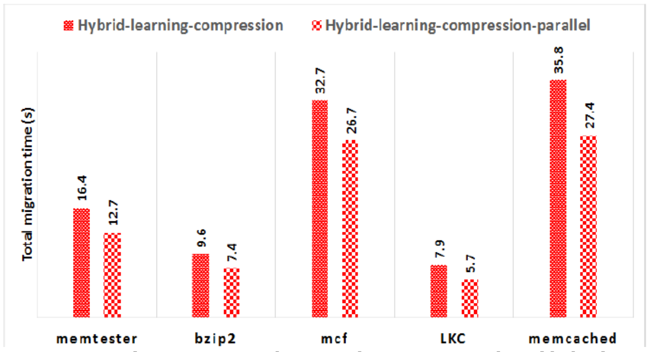


Fig. 6. Total migration time between the compression based hybrid learning and parallelized compression based hybrid learning.

VI. Conclusion

We have designed and implemented a hybrid migration scheme on the top of a KVM hypervisor, which not only utilizes both pre-copy and post-copy techniques, but also incorporates a novel learning phase prior to the push phase to estimate WWS. Our learning phase leads to effective utilization of the source machine CPU cycles and the available network bandwidth. Besides this, we also expedite the push phase with the introduction of the compression and parallel data transfer, drastically reducing the data transfer and total migration time for the push phase.

References

[1] 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. the Nineteenth ACM Symposium on Operating Systems Principles*, pp. 164–177, New York, NY, USA, 2003.

[2] Amazon. Amazon Elastic Compute Cloud (Amazon EC2). (2012), [Online]. Available: aws.amazon.com/ec2/

[3] 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.

[4] S. Al-Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu, "Vmflock: virtual machine co-migration for the cloud," in *Proc. the 20th International Symposium on High Performance Distributed Computing*, pp. 159–170, New York, NY, USA, 2011.

[5] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow *et al*., "Optimizing the migration of virtual computers," in *Proc. the 5th Symposium on Operating Systems Design and Implementation*, 2002, pp. 377–390.

[6] T. Yang, M. Hertz, E. D. Berger, S. F. Kaplan, and J. E. B. Moss, "Automatic heap sizing: taking real memory into account," in *Proc. the 4th International Symposium on Memory Management*, 2004, pp. 61–72.

[7] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian, "Resource management in the autonomic service-oriented architecture," in *Proc. the 2006 IEEE International Conference on Autonomic Computing*, June 2006, pp. 84–92.

[8] S. S. Seiden, "A guessing game and randomized online algorithms," in *Proc. the Thirty-Second Annual ACM Symposium on Theory of Computing*, 2000, pp. 592–601.

[9] A. Whitaker, M. Shaw, and S. D. Gribble, "Denali: Lightweight virtual machines for distributed and networked applications," in *Proc. the USENIX Annual Technical Conferenc*e, 2002.

[10] T. Sherwood, E. Perelman, and B. Calder, "Basic block distribution analysis to find periodic behavior and simulation points in applications," in *Proc. the 2001 International Conference on Parallel Architectures and Compilation Techniques*, 2001, pp. 3–14.

[11] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," in *Proc. the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2006, pp. 2–13.

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

[13] P. Nagpurkar, C. Krintz, M. Hind *et al*., "Online phase detection algorithms," in *Proc. the International Symposium on Code Generation and Optimization*, 2006, pp. 111–123.

[14] T. Sherwood, S. Sair, and B. Calder, "Phase tracking and prediction," in *Proc. the 30th International Symposium on Computer Architecture*, 2003.

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

[16] VMware, Inc. (2009). VMware vMotion. [Online]. Available: www.vmware.com/files/pdf/VMware-VMotion-DS-EN

[17] 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.

[18] Oracle. (2012). VirtualBox. [Online]. Available: virtualbox.org

[19] B. Nicolae and F. Cappello, "A hybrid local storage transfer scheme for live migration of i/o intensive workloads," in *Proc. the 21st international symposium on High-Performance Parallel and Distributed Computing*, New York, NY, USA, 2012, pp. 85–96.

[20] A Kochut and K. Beaty, "On strategies for dynamic resource management in virtualized server environments," in *Proc. 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2007, pp. 193–200.

[21] O. George. (May 22, 2006). Introduction to server virtualization. TechRepublic. [Online]. Available: http://www.techrepublic.com/article/introduction-to-server-virtualization/6074941

[22] C. Christopher *et al.*, "Live migration of virtual machines," in *Proc.* the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation, 2005.

[23] J. P. Rajan. (March 31, 2010). How Live Migration works in Hyper-V R2. [Online]. Available: http://blogs.technet.com/b/ranjanajain/archive/2010/03/31/how-live-migration-works-in- hyper-v-r2.aspx

[24] Q. Ali. (2015). Scaling web 2.0 applications using Docker containers on vSphere 6.0. [Online]. Available: http://blogs.vmware.com/performance/2015/04/scaling-web-2-0-applicationsusing-docker-containers-vsphere-6-0.html

[25] J. Anselmi, E. Amaldi, and P. Cremonesi, "Service Consolidation with End-to-End Response Time Constraints," in *Proc. 34th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 345–352, September 2008.

[26] M. Armbrust, A. Fox, R. Griffith *et al*., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[27] M. Assuncao, M. Netto, B. Peterson, L. Renganarayana, J. Rofrano, C. Ward, and C. Young, "CloudAffinity: A framework for matching servers to cloudmates," in *Proc. 2012 IEEE Network Operations and Management Symposium*, April 2012, pp. 213–220.

[28] A. Schuster *et al*., "Deconstructing amazon ec2 spot instance pricing," *ACM Trans. Econ. Comput*., vol. 1, no. 3, pp. 16:1–16:20, Sept. 2013.

[29] A. Arcangeli, I. Eidus, and C. Wright, "Increasing memory density by using KSM," in *Proc. OLS '09 the Linux Symposium*, pp. 19–28, July 2009.

[30] 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, 2013.

[31] P. Delforge. Energy efficiency, data centers — NRDC. [Online]. Available: http://www.nrdc.org/energy/data-center-efficiency-assessment.asp

[32] B. H. Li, X. Chai, and L. Zhang, "New advances of the research on cloud simulation," in advanced methods, techniques, and applications in modeling and simulation," *Proceedings in Information and Communications Technology*, vol. 4, pp. 144–163, 2012.

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

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

[35] 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.

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

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

[38] Google. (2012). *Google App Engine*. [Online]. Available: cloud.google.com [Nov 1, 2012].

[39] Microsoft. (2012). *Windows Azure.* [Online]. Available: windowsazure.com

[40] IBM. (2012). SmartCloud. [Online]. Available: ibm.com/cloudcomputing

[41] P. Mell and T. Grance, "The NIST definition of cloud computing(draft)," *NIST Special Publication*, vol. 800, p. 145.

[42] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," I in *Proc. the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, New York, NY, USA, 2009, pp. 51–60.

[43] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi. "Enabling instantaneous relocation of virtual machines with a lightweight vmm extension," in *Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 73 –83, May 2010.

[44] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, "Reactive consolidation of virtual machines enabled by postcopy live migration," in *Proc. the 5th International Workshop on Virtualization Technologies in Distributed Computing*, New York, NY, USA, 2011, pp. 11–18.

[45] R. Ho, C.-L. Wang, and F. C. M. Lau, "Lightweight process migration and memory prefetching in openmo six," in *Proc. IEEE International Symposium on Parallel and Distributed Processing*, 2008, pp. 1–12.

[46] KVM. (2012). Kernel-based virtual machine. [Online]. Available: linux-kvm.org

[47] T. Newhall, S. Finney, K. Ganchev, M. Spiegel, "Nswap: A network swap module for linux clusters," in *Proc. the 9th European Conference on Parallel Processing (Euro-Par)*, 2003, pp. 1160–1169.

[48] S. S. Pinter, Y. Aridor, S. Shultz, and S. Guenender, "Improving machine virtualization with 'hotplug memory'," in *Proc. 17th International Symposium on Computer Architecture and High Performance Computing*, 2005, pp. 168–175.

[49] T. H. Romer, W. H. Ohlrich, A. R. Karlin, and B. N. Bershad, "Reducing tlb and memory overhead using online superpage promotion," S*IGARCH Computer Architecture News*, vol. 23, no. 2, pp. 176–187, 1995.

[50] X. Shen, Y. Zhong, and C. Ding, "Locality phase prediction," in *Proc. 11th International Conference on Architectural Support for Programming Languages and Operating System*s, 2004.

[51] T. Sherwood, E. Perelman, B. Calder, "Basic block distribution analysis to find periodic behavior and simulation points in applications," in *Proc. 2001 International Conference on Parallel Architectures and Compilation Techniques*, 2001, pp. 3–14.

[52] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for hpc with xen virtualization," in *Proc. the 21st Annual International Conference on Supercomputing*, pp. 23–32, New York, NY, USA, 2007.

[53] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proc. the Annual Conference on USENIX Annual Technical Conference*, Berkeley, CA, USA, 2005, p. 25.

[54] R. Uhlig *et al*., "Intel virtualization technology," *Computer*, May 2005, vol. 38, no. 5, pp. 48–56.

[55] C. A. Waldspurger, "Memory resource management in VMware ESX server," *SIGOPS Operting Systems Review*, vol. 36, pp. 181–194, 2002.

[56] P. Werstein, X. Jia, Z. Huang, "A remote memory swapping system for cluster computers," in *Proc. Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 75–81, 2007.

[57] D. Williams, H. Jamjoom, Y. H. Liu, H. Weatherspoon, "Overdriver: handling memory overload in an oversubscribed cloud," in *Proc. 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2011, pp. 205–216.

[58] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. the 4th USENIX Conference on Networked Systems Design &#38; Implementation*, 2007, pp. 17–17.

[59] T. Yang, E. D. Berger, S. F. Kaplan, and J. E. B. Moss, "CRAMM: Virtual memory support for garbage collected applications," in *Proc. the 7th Symposium on Operating Systems Design and Implementation*, 2006, pp. 103–116.

[60] E. Zayas "Attacking the process migration bottleneck," *SIGOPS Operating Systems Review*, vol. 21, no. 5, pp. 13–24, 1987.

[61] A. Zeileis, C. Kleiber, W. Kramer, and K. Hornik, "Testing and dating of structural changes in practice," *Computational Statistics & Data Analysis*, vol. 44, no. 109–123, 2003.

[62] X. Zhang, S. Dwarkadas, K. Shen, "Towards practical page coloring-based multi-core cache management," in *Proc. the 4th ACM European Conference on Computer Systems*, 2009.

P. V. S. S. Gangadhar is a scientist-D in NIC & Ph.D Scholar. He is presently studying for a Ph.D at the Department of Information Technology, Gitam Institute of Technology, Gitam University, Vishakapatnam, AndhraPradesh, India. His research interests include e-governance, cloud computing, fuzzy logic and data mining.

Ashok Kumar Hota is a scientist-F at NIC, MEITY, OSU, Bhubaneswar, Govt of India. His research interests include e-governance, tribal informatics, cloud computing, data mining, and big data analytics. He has published several papers in International conferences journals.

Mandapati Venkateswara Rao is a professor at the Department of Information Technology, Gitam Institute of Technology, Gitam University, Vishakhapatnam, India. He got an M.Tech in CST and a PhD in robotics from Andhra University. His research interests includes robotics, cloud computing and image processing. He has published several papers in international conferences and journals.

Vedula Venkateswara Rao is a professor at the Department of Computer Science Engineering, Srivasavi Engineering College, Tadepalligudem, India. He received his master degree in computer science engineering from Jawahar Lal Nehru Technological University Kakinada; master degree in information technology from Punjabi University, Patiayala, India and PhD from Gitam University. His research interests include cloud computing and distributed systems, data mining, big data analytics and image processing. He has published several papers in international conferences and journals.