Global Journal of Computer Science and Technology
Cloud & Distributed
Volume 12 Issue 10 Version 1.0 July 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Going Back and Forth: Efficient Multi-Deployment and Multi-Snapshotting on Clouds

By Syeda Farhath Begum, Dr. Kahalid Mohiuddin & Ashiquee Rasool Mohammad
Osmania University, Hydrabad, India

Abstract - Cloud computing has changed the way people think of using resources. Especially, the IaaS (Infrastructure as a Service) allows users to make use of unlimited resources in pay per use fashion. Virtualization is the technology based on which the cloud service providers are able to provide or share computational resources and data centers to users. Though this approach is practical, it throws certain challenges in terms of designing and development of IaaS middleware. One such challenge is the need for deploying thousands of VM instances to meet the requirements of growing number of users. In the process another challenge is to snapshot multiple images and persisting them towards management tasks like stopping VMs temporarily and resuming them as and when required. The presence of data centers in different configurations enables the simultaneous deployment and snapshotting is important. This capability should be coupled with another feature that is the whole mechanism should be hypervisor independent. To achieve this, a new virtual file system is proposed in this paper. This is basing on lazy transfer scheme with VM optimization and object versioning that takes care of multi-snapshotting and multi-deployment simultaneously and effectively. The experiments have shown that the new filing system and related techniques have improved performance, and bandwidth utilization is reduced by 90%.

Keywords : Cloud Design, Cloud Storage Performance, Empirical Study, Multi-snapshotting, versioning, VM images, lazy propagation, cloning, multi-deployment.

GJCST-B Classification: C.2.1

GOING BACK AND FORTH EFFICIENT MULTI-DEPLOYMENT AND MULTI-SNAPSHOTTING ON CLOUDS

Strictly as per the compliance and regulations of:

Going Back and Forth: Efficient Multi-Deployment and Multi-Snapshotting on Clouds

Syeda Farhath Begum [α], Dr. Kahalid Mohiuddin [σ] & Ashiquee Rasool Mohammad [ρ]

Abstract - Cloud computing has changed the way people think of using resources. Especially, the IaaS (Infrastructure as a Service) allows users to make use of unlimited resources in pay per use fashion. Virtualization is the technology based on which the cloud service providers are able to provide or share computational resources and data centers to users. Though this approach is practical, it throws certain challenges in terms of designing and development of IaaS middleware. One such challenge is the need for deploying thousands of VM instances to meet the requirements of growing number of users. In the process another challenge is to snapshot multiple images and persisting them towards management tasks like stopping VMs temporarily and resuming them as and when required. The presence of data centers in different configurations enables the simultaneous deployment and snapshotting is important. This capability should be coupled with another feature that is the whole mechanism should be hypervisor independent. To achieve this, a new virtual file system is proposed in this paper. This is basing on lazy transfer scheme with VM optimization and object versioning that takes care of multi-snapshotting and multi-deployment simultaneously and effectively. The experiments have shown that the new filing system and related techniques have improved performance, and bandwidth utilization is reduced by 90%.

Keywords : Cloud Design, Cloud Storage Performance, Empirical Study, Multi-snapshotting, versioning, VM images, lazy propagation, cloning, multi-deployment.

I. Introduction

Nowadays, the emergence of Infrastructure as a Service (IaaS) cloud computing is a feasible substitute to the acquisition as well as physical resources management. With the help of IaaS, users can be able to lease storage and time of computation from datacenters that are very large. Leasing of computation time can be achieved by enabling users to deploy virtual machines (VMs) on the resources of the datacenter. As the user possess overall control on the configuration regarding Virtual Machines by making use of on-demand deployments, IaaS leasing is simply similar to purchase of hardware that is dedicated but with no long-term commitment as well as cost. The IaaS on-demand nature is complex to make such kind of leases more attractive, as it allows users for expanding or shrinking their resources with respect to their needs of computation, by making use of external resources for complementing their local resource base [15].

This emerging model results in new challenges with respect to the design as well as development of systems providing IaaS. One among frequently resulting patterns in the operation of IaaS is the necessity for deploying a huge number of VMs on most of the nodes relative to a datacenter at the same instant of time, starting from a collection of VM images that are stored previously in a fashion that is persistent. For instance, this pattern is occurred when the user needs the deployment of a virtual cluster that is used to execute a distributed application or a group of environments for supporting a workflow. This pattern is referred as multi deployment. Such kind of large deployment of most of the VMs at a time can take a longer time. This problem is in particular acute for VM images that are used in scientific computing in which image are large in size (from small number of gigabytes up to greater than 10 GB). A conventional deployment contains hundreds or else thousands of such kind of images. Before starting the instances of VM, conventional techniques of deployment [23] broadcast the images to the nodes , a process which could take time ranging from tens of minutes to approximately hours, not taking into account the time for booting the operating system alone. This could make the time of response of the IaaS installation very longer than that is acceptable and remove the on-demand benefits obtained from cloud computing. Once the instances of the Virtual Machines are being run, a same kind of challenge is applied to snapshotting the deployment. Most of the VM images which were changed locally need to be transferred in a concurrent manner for making storage stable with the reason to capture the VM state for using later (for instance in check pointing or online migration to another cluster or cloud). This pattern is referred to as multisnapshotting. The technique of conventional snapshotting works definitely on custom VM image file formats [9] for storage of only incremental differences in a new file which rely on the original VM image similar to backing file. When taking regular snapshots for a huge number of VMs, such kind of approaches form a huge number of files as well as interdependencies among them, that are difficult for managing and that get in the way with the ease-of-use basis behind clouds. Moreover, with emerging datacenter trends as well as tendencies for

Author α : Department of Computer Sci ence Osmania University, Hydrabad, India. E-mail : farhathbegum.syeda@gmail.com
Author σ ρ : Department of Information System, King Khalid University 61411, Abha, Saudi Arabia.
E-mail σ : drkhalidmk70@gmail.com
E-mail ρ: ashique.rasool@gmail.com

federating clouds [12], configurations have become more and more varied. Custom image formats are not standardized and might be used with particular hypervisors alone that limits the ability for easily migrating VMs among various hypervisors. Hence, multisnapshotting should be handled in a transparent and portable style which hides the interdependencies of additionaldifferences and exposes VM images that are standalone, by greater portability in various hypervisor configurations.

Along with incurring delays that are significant and raising issues of manageability, these patterns can also form huge network traffic which comes in the way through the execution of applications on resources that are leased and results in greater costs of utilization for the user.

In this paper a virtual file system that is distributed specifically that is optimized for patterns of multideployment as well as multi- snapshotting. As the patterns are considered complementary, they are investigated in conjunction. Our proposal provides a proper balance between performance, storage space, and finally consumption of network traffic, while treating snapshotting in a transparent manner and revealing standalone and even raw image files (understood by many hypervisors) to the outside.

The summary of our contributions are as follows:

- We present a flow of design principles which optimize patterns of multideployment as well as multisnapshotting and describe in which manner our design can be integrated with the resources of IaaS (Sections 2 and 3).
- We illustrate how to comprehend these principles of design by building a virtual file system which leverages distributed storage services that are versioning-based. To clear this point, we describe an implementation over BlobSeer, a service related to versioning storage particularly designed for maximum throughput under concurrency [17, 24].

Our approach is evaluated in a sequence of experiments each of which is conducted over hundreds of nodes that are provisioned on the Grid'5000 testbed, by making use of synthetic traces as well as real-life applications.

II. Related Work

Multideployment which depends on complete broadcast-dependent pre- propagation is a commonly utilized technique [28, 23, 11]. While this technique prevents read contention to the repository, it can incur great overhead in network traffic as well as execution time, as mentioned in Section 5.2. Moreover, as the VM images are completely copied on the compute nodes locally, multisnapshotting will not be feasible: greater amounts of data have been duplicated unnecessarily and can cause transfer delays that are not acceptable, without mentioning huge space of storage and utilization of network traffic.

For alleviating this problem, most of the hypervisors offer support of native copy-on-write by giving definition of formats of custom VM image file [12, 20] particularly designed for efficiently storing additional differences. Similar to our approach, this makes base images to be usable in the form of templates that are read-only for multiple logical instances that store modifications per instance. Moreover, deficiency of standardization and also the generation of more number of new files that are interdependent restrict the portability as well as manageability of the snapshots of VM image that result. Another approach that is different in nature for instantiating a huge number of VMs from the identical initial state has been proposed in [13]. The authors present a latest cloud abstraction: VM FORK. Basically this is considered as the equivalent of the fork call on operating systems like UNIX, cloning a VM at every instant into multiple replicas which are running on various hosts. While this is simply equal to CLONE followed by COMMIT in our method ,the main concern is on reducing the time as well as traffic of the network for spawning and running, on the fly, new remote instances of VM that share the identical state of a VM that is already running. Local modifications have been assumed tobe ephemeral, and no support is provided for storing the state persistently.

A similar one to our approach is Lithium [10], a replication system that is fork-consistent for virtual disks. Lithium supports instantaneous volume creation along with lazy space allocation and creation of writable snapshots instantaneously. Not similar to our approach is the one which is dependent on segment trees, Lithium is dependent on log structuring [22], that can potentially humiliate read performance when increasing the number of successive snapshots for the same image: the log of incremental differences is started growing, making it more costly for reconstructing the image.

Cluster volume managers for virtual disks like Parallel ax [16] allow compute nodes for sharing access to a block device that is single and globally visible, which in a collaborative manner managed for presenting individual virtual disk images to the Virtual Machines. While this allows frequent snapshotting that is not efficient like our approach, image sharing is intentionally not encouraged so as to remove the requirement for a distributed lock manager that is claimed for dramatically simplifying the design. Most of the storage systems, like Amazon S3 [5] (backed by Dynamo [8]), are particularly designed as highly accessible key-value repositories for infrastructures of cloud. They may be building blocks that are valuable for block level storage volumes [1] which host images of virtual machine; moreover, they have not been optimized for snapshotting. The intention of our approach is to complement existing platforms of cloud computing, from industry (Amazon Elastic Compute Cloud: EC2 [4]) as well as from academia (Nimbus [2, 12, 24], Open Nebula [3]). While the particulars for EC2 are not available publicly, it has been widely accepted that all of these platforms depend on many of the techniques mentioned above. Claims for instantiating multiple VMs in ―minutes, ‖ moreover, are not sufficient to meet our objectives of performance; So, our work is believed to be a welcome addition in this circumstance.

III. Description of Infrastructure and Other Components

a) About Cloud Infrastructure

Clusters are used in building IaaS cloud platforms. They are made up of hardware that makes use of less power and reduces cost per unit and provides high speed [4]. Many machines are interconnected and each machine is attached a disc storage. Virtualization technology is used in order to share physical resources well. The machines are able to run multiple VMs. Many nodes are dedicated for storage that is responsible for persistence. They might be having either distributed [5] or centralized [2] storage service. Such storage service is responsible to store VM instance images reliably. The manipulations of VMs include deleting, downloading, uploading and so on.

b) State of the Application

VM deployment state has two parts namely the state of all VM instances at any given point of time and the state of the channels between them meant for communications. They include sockets which have been open, network state and virtual topology. In order to make the sate persistent for future reuse and maintenance, it is essential that the VM instances are to be persisted and at the same time hundreds of VM instances are to be created to meet increasing demands of cloud users. However capturing the global state of such channels is difficult [14]. To avoid this problem, the second model is to get sum of all VM instances. This model discards any in-transit traffic in the network and assumes that fault tolerant network is used.

Model 3, which is simplified version of model 2 is that the VM state is represented only by the virtual disk attached to it. It stores only minimal information pertaining to state and such information is reused later. It has the benefits like reduction in size and portability across systems. Model 3 is widely used mechanism in practice and the same is considered in this work.

c) Application Phases

Any VM may not access the whole image. Some utilities and applications are never used. To model this behavior the VM life cycle has been divided into three phases namely boot phase, application phase and shutdown phase. The boot phase reads configuration files, launches processes that represent initial state of VM. The application phase is in either negligible virtual disk access that need not be persisted or data–intensive which needs dedicated storage. The shutdown phase generates very negligible disk access and this phase is not there when VM instance terminates prematurely due to some hardware failure.

IV. Our Methodology and Architecture

In order to optimize the process of multisnapshotting and multideployment, a new filing system is proposed. The following sub sections describe it.

a) Overview of Design

The design of the proposed approach depends on the principles like optimizing multisnapshotting, reducing contention, optimizing VM disk access, and aggregating storage space.

i. Aggregating local Storage

The existing approaches [5, 2, 3] are not capable of making use of storage space available in local hard discs of nodes. To overcome this shortcoming, the proposed approach aggregates storage space from local hard disks and forms a common pool which is used in a distributed fashion. Its advantages are high scalability and freeing memory for reducing overhead in managing VMs.

ii. Optimizing VM access and Reducing Contention

On demand VM image mirroring facilitates to make use of locally available VM image for output. However, it can get from global VM instance the required information in the form of mirroring. It improves performance. Moreover our approach supports reduction of contention as the VM image is split into number of equal sized pieces. While reading values if any piece is not available in the local disk, it is obtained from remote disk thus reducing contention.

iii. Optimizing Multi-Deployment and Snapshotting

When full VM image is saved every time, it consumes lot of resources even though small changes are made. To avoid this certain file formats can be used to incrementally save to other virtual machine. Its drawbacks include limitation of migration capabilities and also the risk of ending up with so many VM instances. By using shadowing and cloning these problems are overcome by the proposed approach.

Global Journal of Computer Science and Technology (B) Volume XII Issue X Version I July 2012

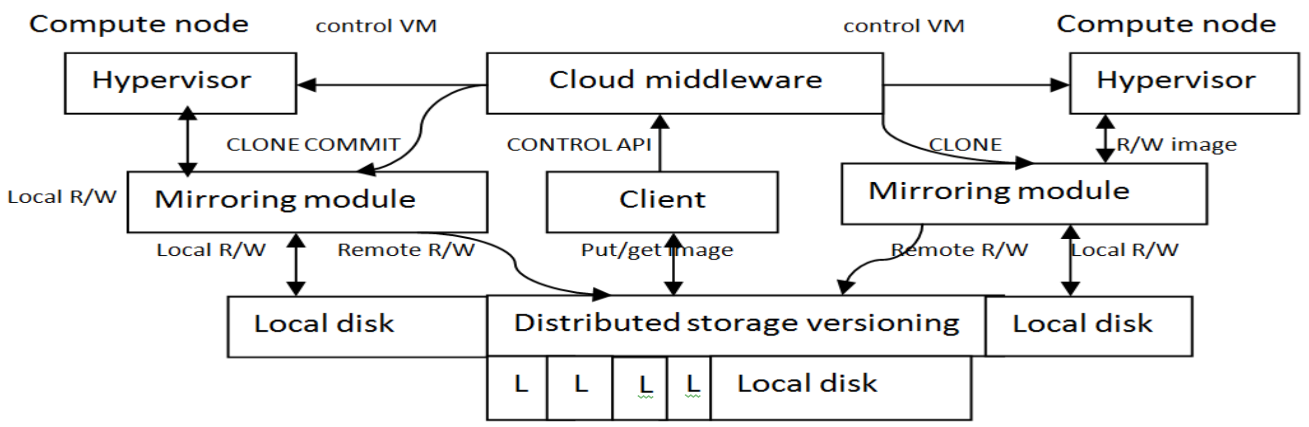

Fig. 1: Shows proposed architecture

V. Proposed Cloud Architecture

Fig. 1 shows the architecture of the proposed system. It has cloud middleware, compute nodes or hypervisors, clients mirroring modules. The cloud middleware facilitates communication to mirroring modules and also hypervisor concurrently. COMMIT is used to save changes permanently while CLONE is used to make another copy. Local disks are involved to form a distributed file system which improves the overall performance of multisnapshotting.

VI. Implimentation Details

The proposed system implementation mainly has two modules namely distributed versioning storage service and mirroring module. The former is meant for improving management of repository while the latter for trapping IO access and runs in each compute node.

a) Software Reused

Some of the components are reused in the proposed system. For instance BlobSheer [17, 18, 19] and FUSE are reused. The BlobSheer is meant for working with LOB objects while the FUSE is meant for implementing mirroring module.

As can be seen in figure 2, the fuse module is made up of many components like hypervisor, cloud middleware, BlobSheer etc.

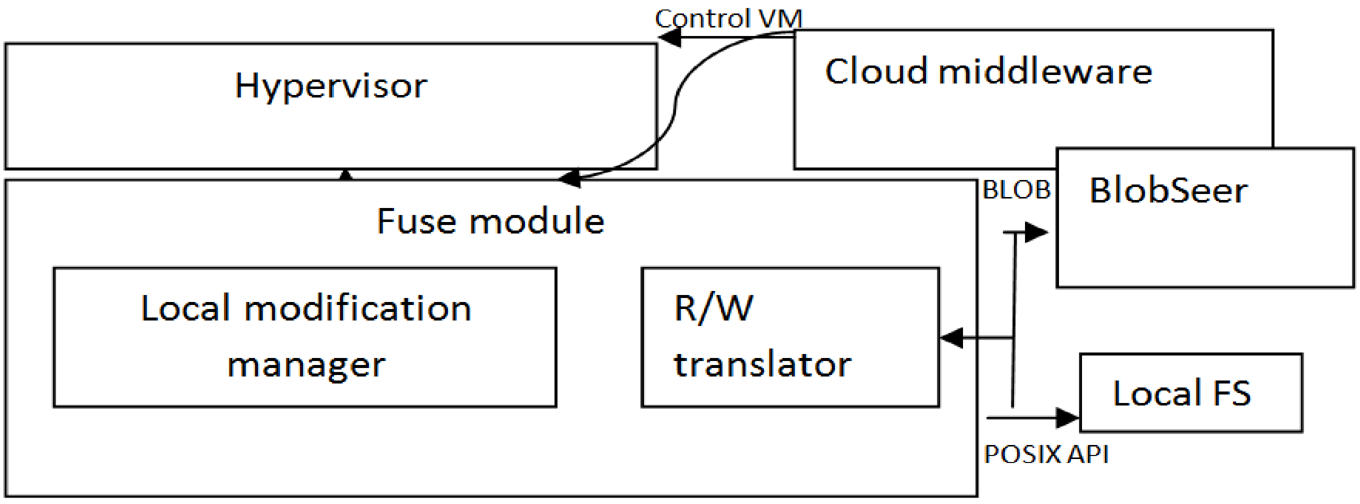

Fig. 2: Fuse Model

b) The Approach

Figure 2 presents FUSE module. Its sub modules are local modification manager and R/W translator. The former is for tracking local content while the latter is meant for translating original requests into remote read and write requests. On opening VM first time, the local disk has an empty file created in order to mirror BLOB image. The storage has been optimized. The local file gets closed after unmapping when VM image is closed. For remote access of VM image through POSIX the commands like COMMIT and CLONE have been implemented as part of FUSE module. COMMIT save local changes into BLOB image permanently. CLONE is meant for cloning VM image. Finally these are integrated with Nimbus cloud.

VII. Evaluations

Experiments and results on multi-deployment and multi-snapshotting are described in the following sub sections.

a) Emperical Setup

Grid'5000 was used to perform experiments. iNancy with 120 clusters was used. Each one is with x86 64 CPU with virtualization support, local HDD worth 250 GB and 8GB of RAM with Internet connection. KVM 0.12.5 was the hypervisor and the OS is Red Hot Linux.

b) Multideployment Performance

The following sub sections throw light into the experimental results. The observations are done in a multideployment pattern when a single VM is used to have ―n number of VM instances.

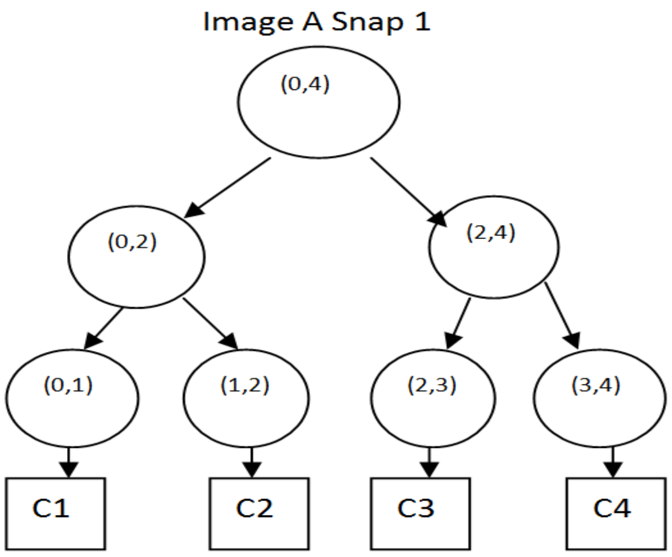

Fig. 3 : Segmentation of chunk details of VM image A

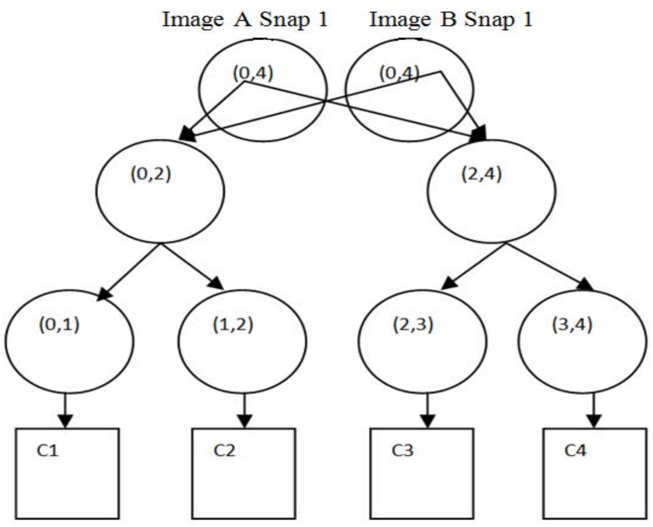

Fig. 4 : Segmentation of chunk details of VM image A

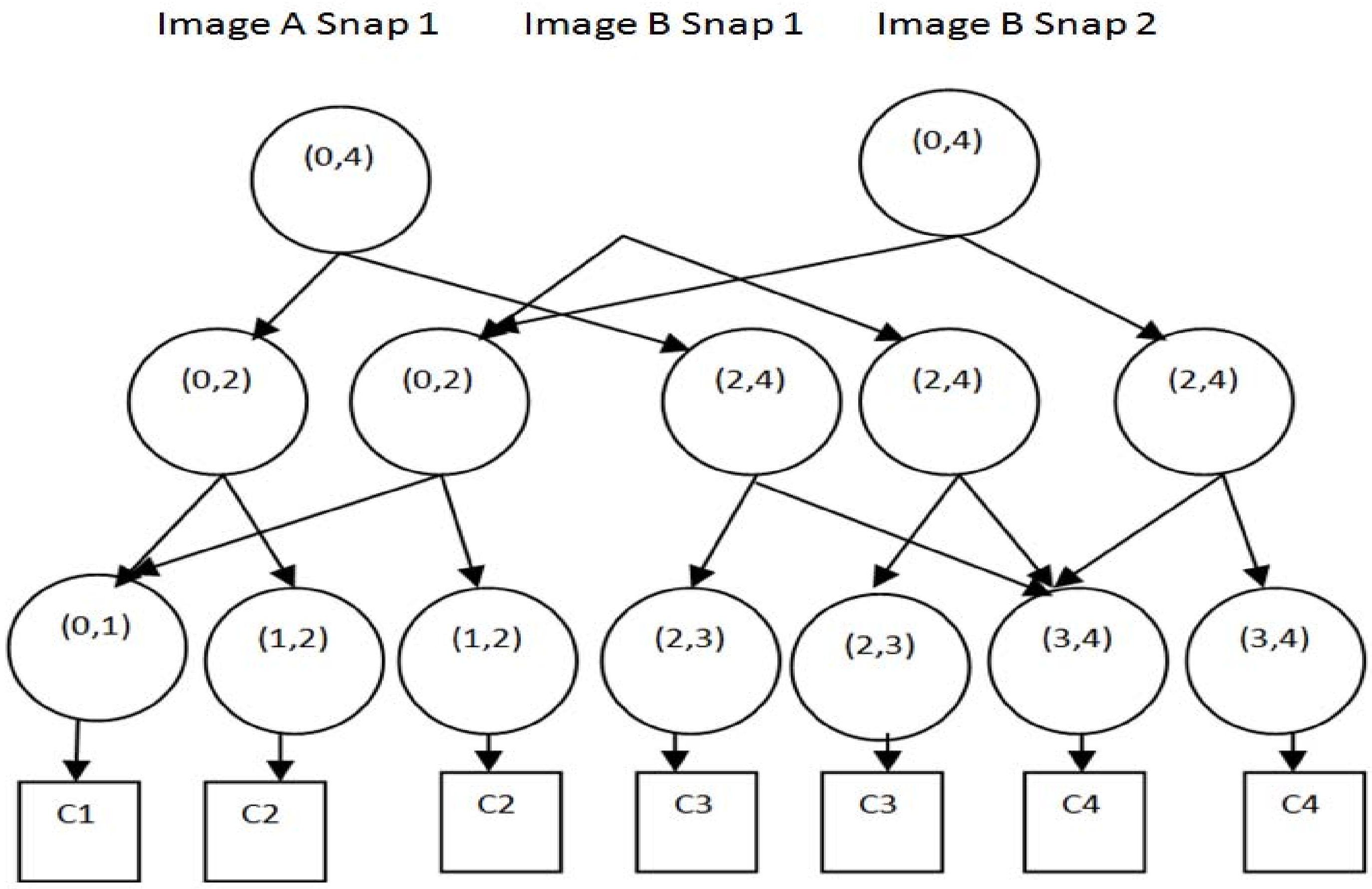

Fig. 5: Segmentation and chunk composition of consecutive snaps

i. *Propogation*

As given in [21, 23], it is part of cloud and has phases like broadcasting of VM image, and launching of VM instances concurrently. The drawback in the propagation approach is the overhead incurred in the initialization phase. Taktuk [7] has been used to overcome this downside. Taktuk is a broadcasting tool which is highly scalable. NFS server is used to store VM images.

ii. *Comparing Qcow2 Over PVFS*

PVFS [6] is used to compare our work. This tool is meant for metadata management with high performance. For comparison it was deployed in compute nodes. In order to initialize VM instances qcow2[9] images are created in the compute node in the local system while PVFS is used as backup image. The performance is measured on average time take to boot each instance and total network traffic.

Figure 6, 7, 8 and 9 shows the results of comparison of other works and our approach.

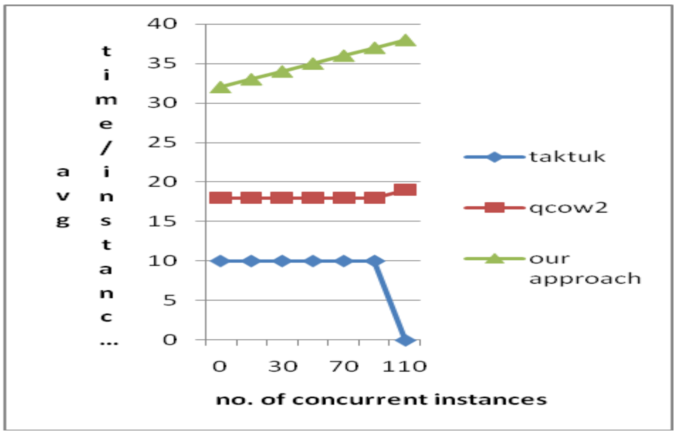

Fig. 6: Performance in terms of no. of concurrent instances

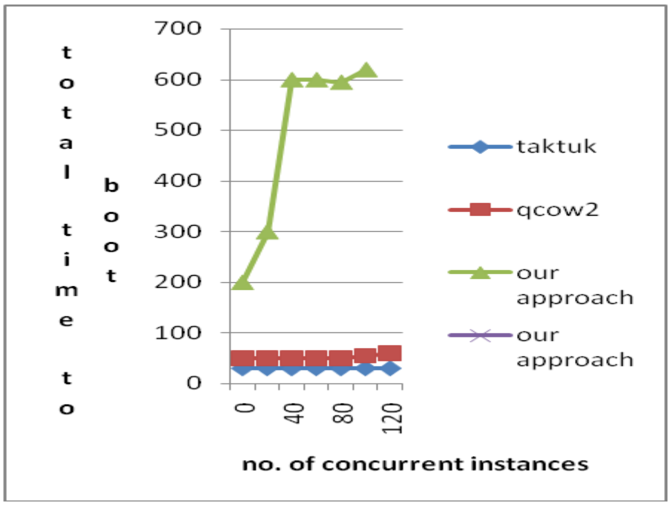

Fig. 7: Performance in terms of no. of concurrent instances

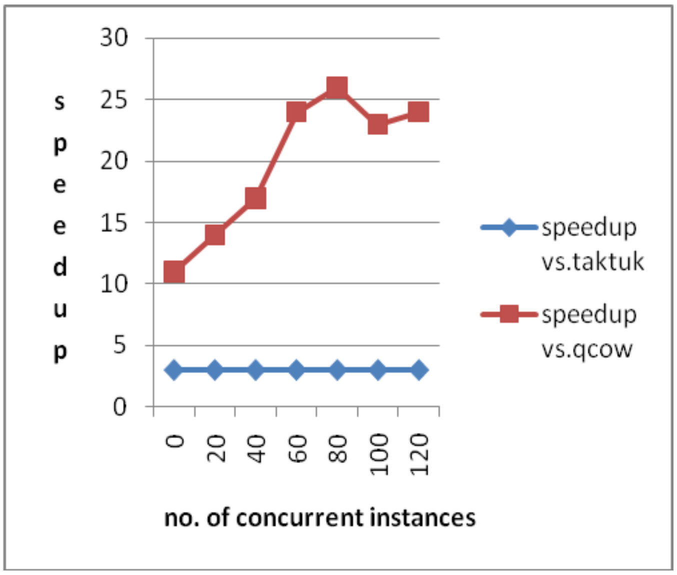

Fig. 8: Performance in terms of no. of concurrent instances

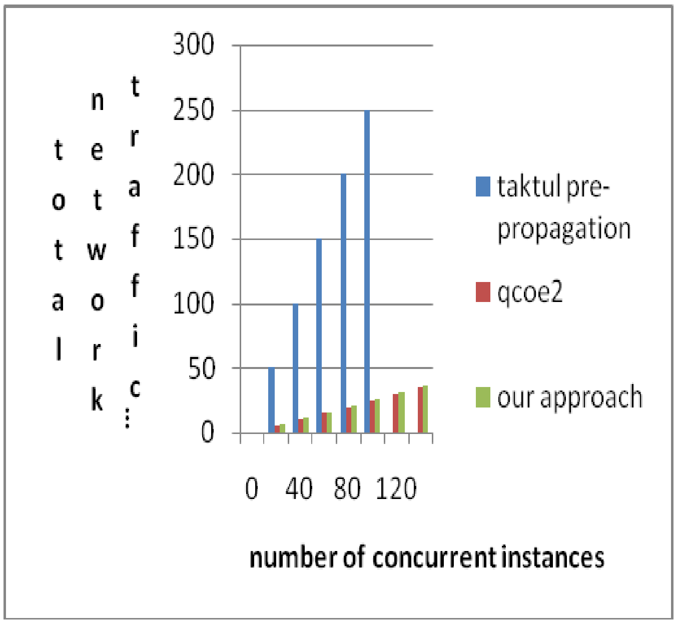

Fig. 9: Shows performance in terms of no. of concurrent instances

iii. *Multi-Snapshotting Performance*

The performance of our approach in case of multisnapshotting is described in this section. The comparison is made between qcow2 over PVFS and our approach. Fig. 10 and 11 show the performance of multi-snapshotting of our approach and qcow2 over PVFS. When overall performance is considered, our approach is taking relatively less time for instance creation and completion.

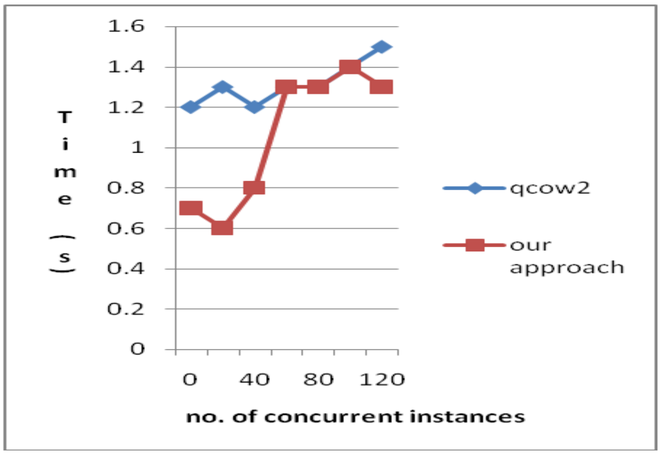

Fig. 10: Shows average time snapshot an instance

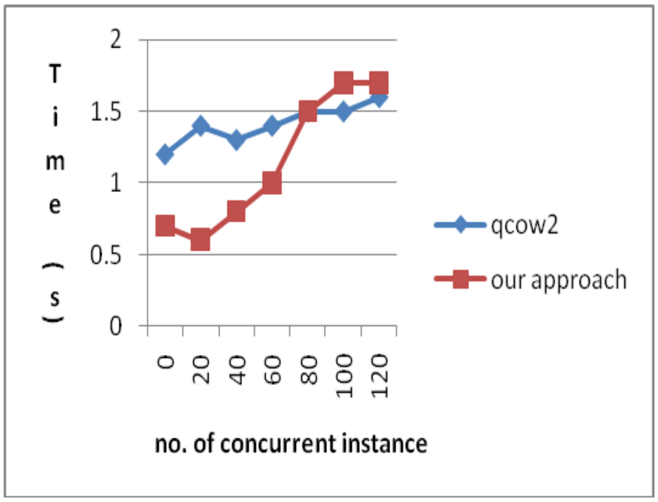

Fig. 11: Shows completion time to snapshot all instances

Figure 12 (a), (b) and (c) show the performance of access pattern, operation type and setting of local and our approach. The access patterns compared are Read, Write and Overwrite in block of 8 KB. The operation types considered are random seeks, file creation, and file deletion. The Fig. 12 (c) shows the time taken to finish simulation using 100 VM instances.

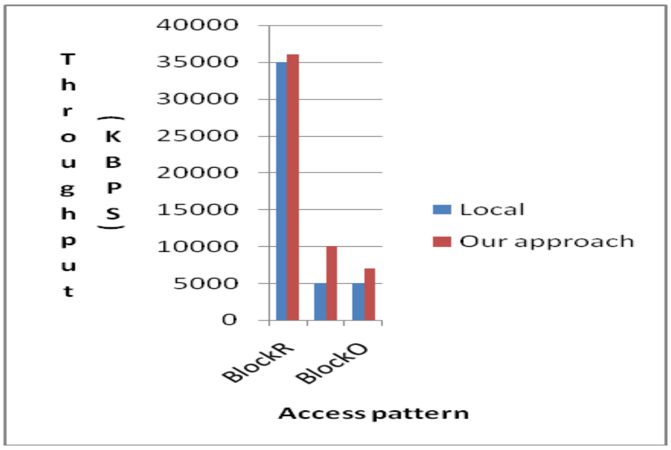

Fig. 12: (a) Access pattern in terms of throughput

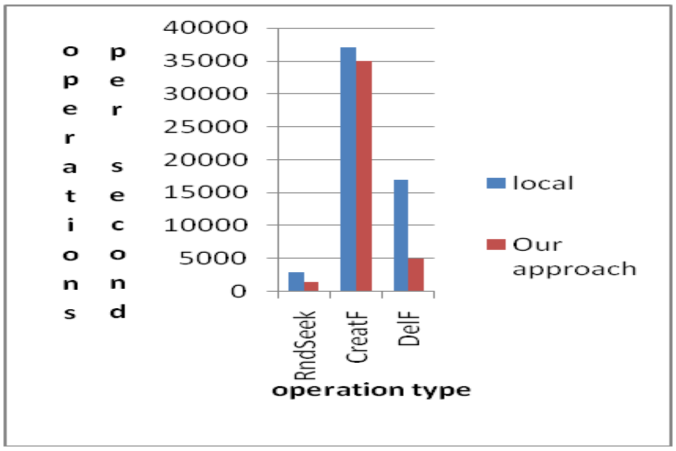

Fig. 12: (b) Shows operation type and operations per second

Our approach is showing better results and it is intended to help cloud platforms such as EC2, Nimbus etc. We believe that our work can be used in any existing cloud platform in order to improve its performance in terms of managing virtual machines and improving performance by using our techniques pertaining to multi-snapshotting and multi-deployment. Figure12 (c) shows time taken to finish simulation using 100 VM instances

VIII. Conclusion and Future Work

Since cloud computing is becoming more popular and efficient management of VM images, like image propagation for computing nodes and image snapshotting for the purpose of check- pointing or migration is difficult. The performance of these kind of operations affects in a direct manner the usability of the benefits provided by systems of cloud computing. This paper presented various techniques which integrate with middleware of the cloud for handling two patterns efficiently. They are multideployment and multi-snapshotting.

A lazy VM deployment scheme which fetches content of the VM image as required by the application that is executed inthe VM, thereby minimizing the pressure on the storage service of VM for deployment requests that are heavily concurrent. Moreover, we leverage object versioning for saving local VM image differences alone back to persistent storage when a snap-shot is generated, yet offer the illusion that the snapshot is a different, completely independent image. This has two crucial benefits. First, it does the management of updates of the hypervisor in an independent manner, thus greatly enhancing the portability of VM images and providing compensation for the deficiency of standardization of the VM image format. Second, it manages snapshotting in a transparent manner at the level of the repository of the VM image, simplifying to a great extent the snapshots management. We have given the demonstration of the advantages of our approach via experiments on number of nodes by making use of benchmarks as well as applications of real-life. When compared with simpler approaches depending on pre-propagation, our approach gives a best improvement in execution time as well as resource usage: the total time for performing a multi-deployment got reduced approximately to a factor of 25, and the storage and bandwidth usage got reduced by approximately 90%. When compared with approaches which make use of copy-on-write images (i.e., qcow2) depending on raw backing images that are stored in a distributed file system (i.e., PVFS), a speedup of multideployment by a factor of 2 and multi-snapshotting performance that is comparable are shown, each with the extra benefits of transparency as well as portability.

Depending on these results that are supported, we plan for exploring the multi-deployment as well as multi-snapshotting patterns in a more extensive manner. According to multideployment, one optimization that is possible is to build a scheme that is perfecting depending on last experience through the access pattern. According to multi-snapshotting, reductions that are interesting in time as well as storage space can be achieved by presenting deduplication schemes. We also intend for fully integrating the present work with Nimbus [2] and thereby explore its advantages for more critical applications of HPC in the real world.

References Références Referencias

1. Amazon elastic block storage (ebs). http://aws.amazon.com/ebs/.
2. Nimbus. http://www.nimbusproject.org/.
3. Opennebula. http://www.opennebula.org/.
4. Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2/.
5. Amazon Simple Storage Service (S3). http://aws.amazon.com/s3/.
6. P. H. Carns, W. B. Ligon, R. B. Ross, and R. Thakur. Pvfs: A parallel file system for Linux clusters. In Proceedings of the 4th Annual Linux Showcase and Conference, pages 317–327, Atlanta, GA, 2000. USENIX Association.
7. B. Claudel, G. Huard, and O. Richard. Taktuk, adaptive deployment of remote executions. In HPDC '09: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, pages 91–100, New York, 2009. ACM.
8. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles, pages 205–220, New York, 2007. ACM.
9. M. Gagn´e. Cooking with Linux—still searching for the ultimate Linux distro? Linux J., 2007(161):9, 2007.
10. J. G. Hansen and E. Jul. Scalable virtual machine storage using local disks. SIGOPS Oper. Syst. Rev., 44:71–79, December 2010.
11. M. Hibler, L. Stoller, J. Lepreau, R. Ricci, and C. Barb. Fast, scalable disk imaging with Frisbee. In ATC '03: Proceedings of the 2003 USENIX AnnualTechnical Conference, pages 283–296, San Antonio, TX, 2003.
12. K. Keahey, M. O. Tsugawa, A. M. Matsunaga, and J. A. B. Fortes. Sky computing. IEEE Internet Computing, 13(5):43–51, 2009.
13. H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell,P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan. SnowFlock: Rapid virtual machine cloning for cloud computing. In EuroSys '09: Proceedings of the 4th ACM European Conference on Computer Systems, pages 1–12, New York, 2009. ACM.
14. X. Liu, J. Huai, Q. Li, and T. Wo. Network state consistency of virtual machine in live migration. In SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing, pages 727–728, New York, 2010. ACM.
15. P. Marshall, K. Keahey, and T. Freeman. Elastic site: Using clouds to elastically extend site resources. In CCGRID '10: Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID '10, pages 43–52, Washington,DC, USA, 2010. IEEE Computer Society.
16. D. T. Meyer, G. Aggarwal, B. Cully, G. Lefebvre, M. J. Feeley, N. C. Hutchinson, and A. Warfield. Parallax: Virtual disks for virtual machines. SIGOPS Oper. Syst. Rev., 42(4):41–54, 2008.
17. B. Nicolae. BlobSeer: Towards E_cient Data Storage Management for Large-Scale, Distributed Systems. PhD thesis, University of Rennes 1, November 2010.
18. B. Nicolae, G. Antoniu, L. Boug´e, D. Moise, and A. Carpen-Amarie. BlobSeer: Next-generation data management for large scale infrastructures. J. Parallel Distrib. Comput., 71:169–184, February 2011.
19. B. Nicolae, D. Moise, G. Antoniu, L. Boug´e, and M. Dorier. Blobseer: Bringing high throughput under heavy concurrency to Hadoop map/reduce applications. In IPDPS '10: Proceedings of the 24th IEEE International Parallel and DistributedProcessing Symposium, pages 1–12, Atlanta, GA, 2010.
20. D. Reimer, A. Thomas, G. Ammons, T. Mummert, B. Alpern, and V. Bala. Opening black boxes: Using semantic information to combat virtual machine image sprawl. In VEE '08: Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on

Virtual Execution Environments, pages 111–120, New York, 2008. ACM.
21. A. Rodriguez, J. Carretero, B. Bergua, and F. Garcia. Resource selection for fast large-scale virtual appliances propagation. In ISCC '09: Proceedings of 14th IEEE Symposium on Computers and Communications, pages 824–829, 5-8 2009.
22. M. Rosenblum and J. K. Ousterhout. The design and implementation of a log-structured file system. ACM Trans. Comput. Syst., 10(1):26–52, 1992.
23. R. Wartel, T. Cass, B. Moreira, E. Roche, M. Guijarro, S. Goasguen, and U. Schwickerath. Image distribution mechanisms in large scale cloud providers. In CloudCom '10: Proceedings 2nd International Conference on Cloud Computing Technology and Science, Indianapolis, IN, 2010.
24. K. Keahey and T. Freeman. Science clouds: Early experiences in cloud computing for scientific applications. In CCA'08: Proceedings of the 1stConference on Cloud Computing and it's Applications, 2008.

This page is intentionally left blank