SnapMig: Accelerating VM Live Storage Migration by Leveraging the Existing VM Snapshots in the Cloud

Yaodong Yang, Bo Mao, *Member, IEEE*, Hong Jiang, *Fellow, IEEE*, Yuekun Yang, Hao Luo, and Suzhen Wu, *Member, IEEE*

Abstract—Virtual Machine (VM) live storage migration is becoming increasingly important and indispensable in the current cloud data centers, for the purposes of load balance, hardware maintenance and system upgrade. Nevertheless, conventional VM migration approaches induce significant extra storage and network traffic to the source server that is already heavily loaded or scheduled for upgrade or repair. As a result, both the VM performance perceived by the application/user and the migration performance are degraded significantly. In this paper, we aim to address this problem by proposing a novel scheme, called *SnapMig*, to improve the VM live storage migration efficiency and eliminate its performance impact on user applications at the source server by effectively leveraging the existing VM snapshots in backup servers. By outsourcing the task of transferring VM base image and snapshots to the destination server to backup servers, the source server only needs to migrate the latest state changes to the destination server, leading to simultaneous improvement on VM performance, migration time and multiple-VM migration efficiency. Our lightweight prototype implementation of the SnapMig scheme demonstrates that, compared with the state-of-the-art approaches, SnagMig can significantly reduce the migration time and improve the source-server VM performance at the same time. Moreover, the performance improvement provided by SnapMig becomes much more pronounced with multiple concurrent VM migrations.

Index Terms—Virtual machine, live storage migration, cloud computing, VM snapshot, performance evaluation

1 Introduction

The cloud computing technology is revolutionizing how businesses are conducted in the Enterprise IT departments [1]. With the cloud, enterprises rent IT resources from the cloud providers on an on-demand and pay-as-you-go basis, which brings in numerous benefits ranging from cost savings, to higher level of reliability, availability and scalability. Netflix, a leading video service company in the world, has shutdown all of its own data centers and run all the video services in the Amazon AWS cloud platform [2]. In the last quarter of 2014 alone, five of the largest cloud computing providers, including Amazon, Microsoft, IBM, Google and Salesforce, saw their revenue of the cloud infrastructure service surge by between 37 and 96 percent. Despite of its total market of $16 billion, cloud computing as an industry is still considered in its infancy, since $16 billion is only a tiny fraction of the almost $4 trillion of IT spending for companies globally [1].

As one of the most popular products in the cloud computing market, Virtual Machines (VMs) have been extensively deployed to run different services for customers, such as web service, mail service, database service and printing service [3], [4]. The underlying hypervisors for the virtualized environment are responsible for the management of physical devices and provision of all sorts of virtual devices to VMs. At the same time, hypervisors are supposed to guarantee the isolation and fairness among different VMs on top of a single shared physical server, while improving the overall performance for all the VMs with the accessible physical resources [5], [6].

The VM live storage migration is a built-in module within modern hypervisors, designed to facilitate management tasks such as load balance, hardware maintenance and system upgrade. It can migrate a running live VM from one physical server to another, either within the same cluster or across different data centers globally [7]. At the same time, this process is transparent to the applications running within the migrating VM and other co-scheduled VMs on the same physical server. Most modern hypervisors, including VMware ESX, Windows Hyper-V, Citrix XEN and QEMU-KVM, provide live storage migration as a core functionality. This is primarily because of the increasing need

- *Y. Yang was with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68503, and now with the Microsoft, Redmond, WA 98052. E-mail: yaodong.yangy@gmail.com.*
- *B. Mao is with the Software School of Xiamen University, Xiamen, Fujian 361005, China. E-mail: maobo@xmu.edu.cn.*
- *H. Jiang is with the Computer Science and Engineering Department, University of Texas at Arlington, Arlington, TX 76019. E-mail: hong.jiang@uta.edu.*
- *Y. Yang and H. Luo were with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68503. E-mail: {yuyang, hluo}@cse.unl.edu.*
- *S. Wu is with the Computer Science Department, Xiamen University, Xiamen, Fujian 361005, China. E-mail: suzhen@xmu.edu.cn.*

Manuscript received 16 July 2017; revised 21 Nov. 2017; accepted 25 Dec. 2017. Date of publication 8 Jan. 2018; date of current version 11 May 2018. (Corresponding author: Bo Mao.)
Recommended for acceptance by Z. Chen.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2790389

for dynamic resource allocation, load balancing, server consolidation, system maintenance and upgrade, VM mobility and manageability in cloud data centers. For instance, each running VM has its own resource requirement in the runtime, e.g., the memory footprint, the network bandwidth and the storage throughput. If the physical server cannot provide such resources to the VM, the VM will be live migrated to another server that has required resource available. In addition, it is a known fact that load balancing among hundreds of thousands of servers is a big concern in data centers [8], [9], [10]. With the support of VM live storage migration, the mapping between VMs and the hosting physical servers can be dynamically adjusted, so that a better level of load-balancing and energy efficiency can be achieved in the runtime.

VMs may encounter two types of failures: system crash and data loss. VM snapshots, which preserve the VM state at a previous time, can be employed to roll back the VM to a previous consistent state before the system crash happens. Snapshot backup that transfers previous VM snapshots to backup servers is routinely used to restore VM states when hypervisors encounter data-loss failures [11], [12], [13]. Both of these two schemes are extensively deployed in the cloud computing products [14].

However, VM live storage migration is a nontrivial process. All the state information of the migrating VMs, including the memory footprint, the states of the virtual network, the virtual disk images and snapshot information, must be migrated to the destination servers [15]. The total size of such state information is usually on the order of tens of GBs, of which the largest part is the virtual disk images and snapshots that account for about 80 to 95 percent. Snapshots provide more flexibility and reliability for users, which comes at the expense of more data transmission during the VM live storage migration process. Moreover, as migrating VMs are running applications for customers during the migration period, the VM states keep changing in the migration period and all these updates must be migrated to the destination servers as well. More importantly, when multiple VMs live migrate at the same time, it is more challenging to migrate VMs quickly and provide reasonable IO performance for all the VMs simultaneously [16], [17], [18].

During the VM live storage migration, migration threads, which represent additional workload, are introduced in the source servers and they consume significant amount of storage resource. However, the overall system resource remains unchanged. To make things worse, the migration threads and VM threads interfere, instead of cooperate, with each other, which leads the performance of all these threads to degrade significantly [19]. There are several VM migration schemes proposed in the literature and industry products, such as Dirty Block Tracking [20], IOMirroring [20], workload-aware [21] and redundant data reduction [22], [23]. However, all of them ignore the fact that migrating VMs' base images and previous snapshots are already in the backup servers (through the regular backup operations), and backup servers can help migrate this resource-hungry (i.e., network and storage bandwidth) information to the destination servers on behalf of the source servers. In this paper, *we argue that by leveraging the VM backup snapshots in the VM migration process, the overall migration efficiency can be improved significantly*.

Motivated by the observation of the VM snapshots-backup process and its resulting snapshots of the VM state information available in the backup servers, we propose a novel VM live storage migration middleware, called *SnapMig*, to improve both the VM performance and migration performance simultaneously. By leveraging the backup servers to transfer migrating VMs' base images and previous snapshots, the source servers only need to migrate the latest VM state changes to the destination servers. Consequently, the otherwise severe interference between the I/O traffic generated by the user applications within the VMs (including the migrating VMs) in source servers and the I/O traffic induced by the VM migration process is significantly reduced, leading to substantial performance improvements to both the VM threads and migration threads. In addition, after the migration, recent VM snapshots made available in the destination servers by the backup servers allow the users to roll back their VMs to any of the previous states freely. Moreover, SnapMig is orthogonal to existing migration approaches, and it is regarded as a performance optimization middleware for the existing migration approaches. Finally, we observe that the performance advantages of SnapMig become more pronounced with the concurrent live storage migrations of multiple VMs with many snapshots.

The rest of this paper is organized as follows. Background and motivation are presented in Section 2. The design of the SnapMig is illustrated in Section 3. The performance evaluation of a lightweight SnapMig prototype implementation is presented in Section 4 and the related work is provided in Section 5. Section 6 concludes this paper.

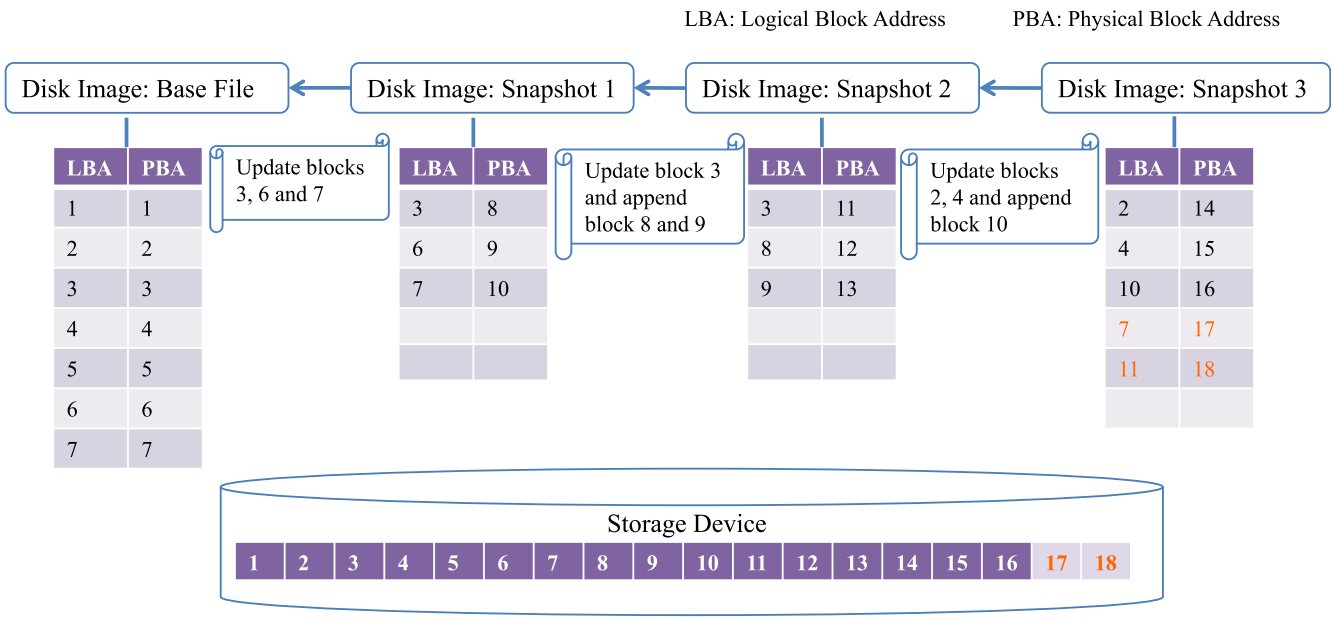
2 Background and Motivation

In this section, we provide the necessary background information for the SnapMig research, including VM snapshots, live migration of VM snapshots and VM snapshot backup, which then helps motivate this research.

2.1 VM Snapshot

For the VM reliability and availability purposes, VM snapshots are widely employed to restore VMs for customers upon system crash or data loss [11], [12], [24], [25]. Currently most modern virtualization platforms, including VMware, Hyper-V, and KVM, support snapshots in their products.

There are two types of snapshots: disk snapshots and system snapshots. A disk snapshot retains the state of the corresponding VM's virtual disk image at a specific time stamp. Given a disk snapshot, the user can roll back the VM to a previous consistent state freely, but the VM needs to reboot and applications are required to restart. The creation process for a disk snapshot includes two phases: (1) flushing only the in-memory buffer cache data to virtual disks, and (2) taking a snapshot for each virtual disk. Therefore, snapshot generation incurs negligible performance overhead for the running applications within in the VM. Disk Snapshots are largely adopted for VM backup and disaster recovery. A system snapshot contains the state information of the RAM and other virtual devices of the VM, besides virtual disk images.



Requests	Step_1	Step_2	Step_3	Step_4	Step_5	Step_6
Read block 1	Search Snapshot 3	Search Snapshot 2	Search Snapshot 1	Search Base File	Get PBA 1	Read PBA 1 from Storage Device
Read block 6	Search Snapshot 3	Search Snapshot 2	Search Snapshot 1	Get PBA 9	Read PBA 9 from Storage Device	
Update block 7	Allocate block 17 in Storage Device	Write data to physical block 17	Add an entry to Snapshot 3 (7->17)			
Append block 11	Allocate block 18 In Storage Device	Write data to physical block 18	Add an entry to Snapshot 3 (11->18)			

Fig. 1. The structure of VM snapshots and the workflow for read/write requests.

With the support of system snapshots, users can roll back the VM to a previous running state. Applications will be resumed at the last execution point, so that it is unnecessary to reboot the VM in the restore period. However, a system snapshot takes a longer time to create than a disk snapshot, since the state of the memory footprint and all virtual devices will be recorded in the snapshot. In addition, the running VM will be stunned during the creation phase, which will cause significant performance degradation for the running applications inside the VM [26]. System snapshots are mainly used to perform risky operations in a testing environment. In this work, we only focus on the disk snapshots, because they are widely used in the VM environment. Moreover, a migration scheme based on disk snapshots is also easy to be extended to one based on system snapshots.

For the purpose of storage efficiency, every VM snapshot only records the state changes of the VM image made since the most recent snapshot in the actual implementation, assuming that it has the access to all the previous snapshots and the base image. As shown in Fig. 1, each snapshot and the base image maintain a mapping table from the Logical Block Address (LBA) to the Physical Block Address (PBA) as their metadata. Each table (except the one in the base image) records the updates and new writes since the most recent snapshot. For the update (block 7) / write (block 11) requests, new entries are added to the mapping table of snapshot 3 (the current snapshot). For the read requests (block 1 and 6), older snapshots (snapshots 1, 2 and 3) and/or the base image have been queried in order to get the latest version of the accessed data blocks. Normally, production servers hold several snapshots for each VM, so that the VM can be restored to any of the stored previous snapshots quickly upon system crash or data corruption. Besides the snapshot support in modern virtualization platforms, major storage vendors like EMC also provide storage optimization for VM snapshots [14].

2.2 The Live Migration of VM Snapshots

As indicated in the previous section, there are several existing snapshots for each VM created by the user or system automatically, and each snapshot holds the changes of the VM image since the last snapshot or base image (if this is the first snapshot). The size of each VM snapshot varies significantly, and it largely depends on the write traffic to the running VM. Take the Aliyun cluster use case as an example [27], [28], the size of each VM is 40 GB, and each VM snapshot is 8 GB on average, which means 20 percent of the virtual disk images have been changed since the last snapshot. Suppose a VM has 2 snapshots, it will consume $40\,GB + 2 \times 8\,GB = 56\,GB$ physical storage capacity, even if its logical capacity is still 40 GB. When it comes to the VM live migration, these snapshots will be either migrated to the destination server, a scheme called *FullMig*, or discarded at source server, a scheme called *SelectiveMig*. If we migrate these snapshots to the destination server (FullMig), the user can make use of these snapshots for VM restore at the destination server, as they did before the migration [26].

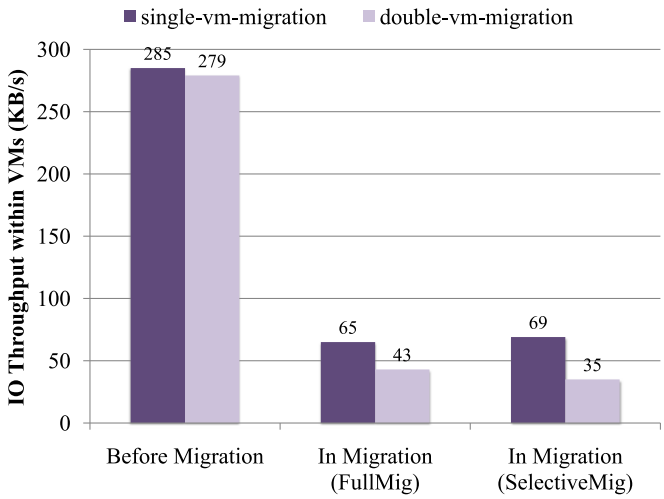


Fig. 2. The VM IO performance before and during the live storage migrations.

However, the total amount of data transmission increases and a longer migration time is unavoidable. If we discard the snapshots, and just migrate the current VM state to the destination server (SelectiveMig), the total amount of data transmission is much smaller than the FullMig scheme [29]. However, users can not roll back to any of the previous snapshots at the destination server.

The VM IO performance and VM live storage migration performance are interactive. Fig. 2 shows that the VM IO performance drops from $\frac{285}{65} = 4.38\times$ (reduction) to $\frac{279}{35} = 7.97\times$ (reduction), from migrating a single VM to migrating two VMs. At the same time, the migration time increases by $\frac{3805}{1425} = 2.67\times$ and $\frac{2903}{1059} = 2.74\times$ for the FullMig and SelectiveMig schemes respectively, as we increases the number of snapshots for each migrating VM and the number of concurrent VM migrations, as shown in Fig. 3. Therefore, both VM IO performance and migration performance are adversely affected by the VM live storage migration significantly. Ideally, we would like a solution that can deliver faster VM live storage migration and preserve all the previous snapshots simultaneously. More importantly, we would like to reduce the traffic for the source server, as the performance degradation for the migrating/co-located VMs in the source server in the migration period is crucial for the overall system performance.

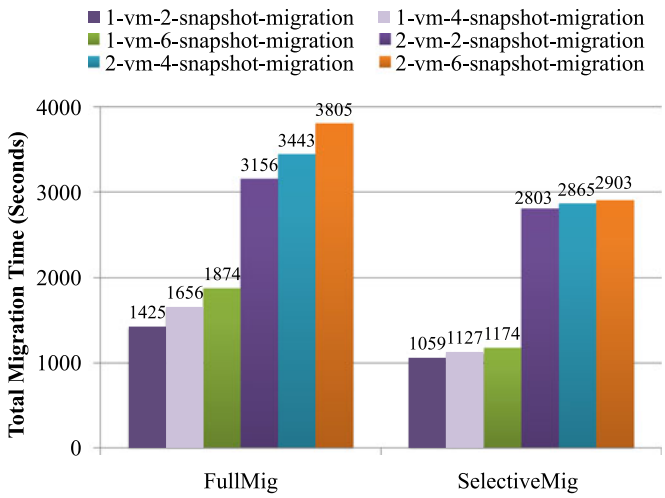


Fig. 3. The migration performance when migrating a single VM and migrating two VMs.

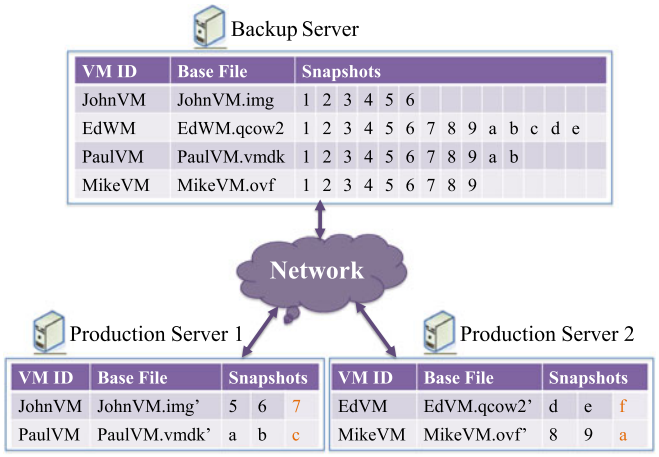


Fig. 4. The distribution of VM snapshots among production servers and the backup server.

2.3 VM Snapshot Backup

For the VM reliability and availability purposes, there are many mechanisms employed to protect VMs, such as snapshots of the datastores via storage systems, replication of storage volumes/LUNs, snapshots of virtual machines and replication within virtualized applications [14]. Among all these mechanisms, the most widely employed is the VM snapshot backup [30], [31]. In the runtime, a series of VM snapshots are created, and then these snapshots will be transferred to backup servers in the backup window regularly. Within backup servers, these snapshots will be further processed to reduce the storage space consumption [32]. For instance, by employing the deduplication technology, the redundant data blocks will be identified and removed.

As indicated in Fig. 4, for a particular VM (JohnVM), the production server holds several most recent snapshots (snapshots 5, 6 and 7) for a single VM, while the backup server holds all the previous VM snapshots (snapshots 1-6) and the base image (JohnVM.img) at the time of the last backup operation. The newly created snapshot (snapshot 7) after the last backup operation will only reside in the production server. The previous snapshots (1-4) are merged into the base image (JohnVM.img') in the production server. In this scenario, if the production server encounters any data loss or power outage issue, there is still an extra copy of the VM and its snapshots in the backup server. Another production server can resume the VM with the support from the backup server. When there is no data corruption occurring in the production server, the snapshots in the backup server will remain idle most of the time.

2.4 Motivation

During the VM live storage migration, the source server will be quite busy, e.g., executing the scheduled maintenance task, running many co-located VMs, or waiting to shutdown soon. In general, whenever VM live storage migration is invoked, IO-intensive migration threads will be introduced to the source server. In order to achieve satisfactory VM IO performances for all the migrating/co-located VMs in the source server and migrate VMs to the destination servers

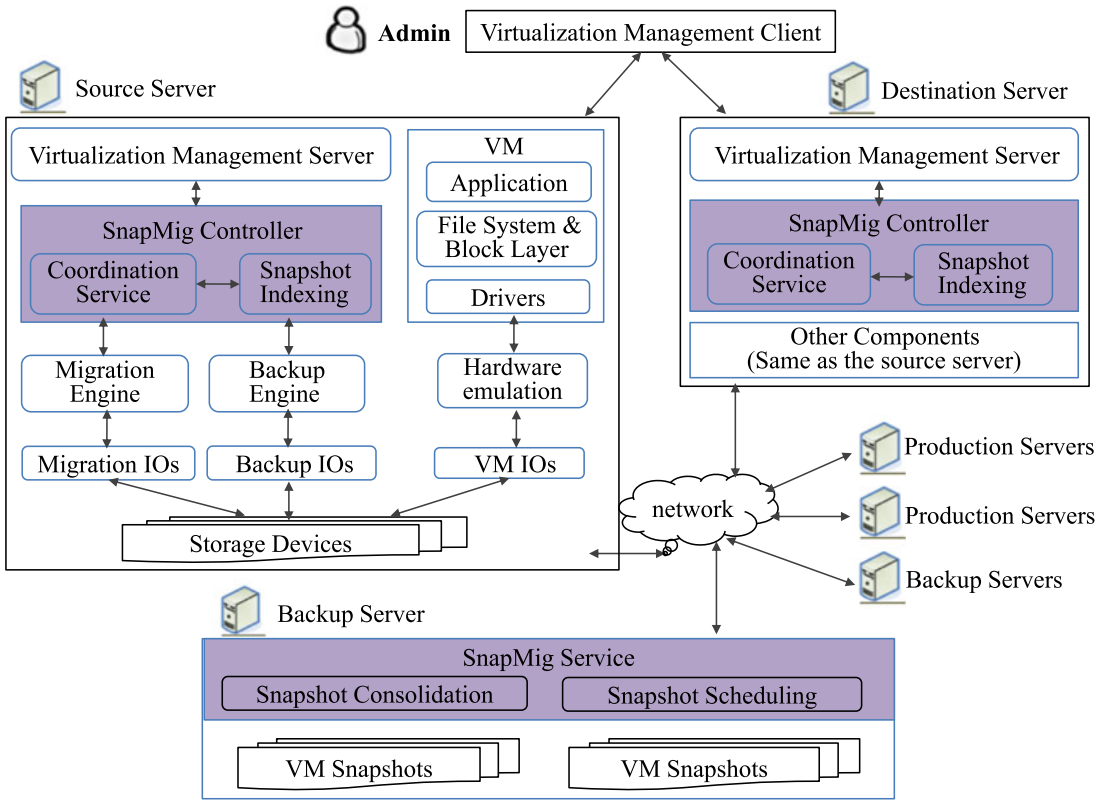


Fig. 5. Architecture of the SnapMig system.

quickly, it is vitally important to eliminate the unnecessary IO traffic to the source server.

On the other hand, we also observe that for the purpose of the reliability and availability, VM snapshot backup servers are widely used in the cloud computing environments. In order to address the performance issues of VM live storage migrations, we propose the SnapMig scheme to leverage the snapshots stored in the largely idle backup servers. In the SnapMig scheme, backup servers transfer the base images and recent snapshots of the migrating VMs to the destination server(s), while the source server only migrates information about the latest changes to the migrating VMs. The migrating VMs will be resumed once their states are reconstructed successfully in the destination server(s).

The benefits of SnapMig are fourfold: 1) *Better VM IO performance during the migration*, because the much reduced, if not completely eliminated, migration traffic involved in the source server allows the migrating/co-located VMs to achieve much better IO performance, as if there were no migration at all for most of the time; 2) *Shorter migration time*, because the backup server(s) that are idle most of the time can transfer the VM images to the destination server(s) at a much higher speed than any source server that has a very heavy running workload; 3) *All the previous snapshots are available in destination server(s)*, so that the users can freely roll back VMs to any of the previous states; and 4) *The performance improvement will be much more pronounced in the scenarios where multiple VMs with many snapshots are live migrated concurrently*. As a result, the contention between the user I/Os and the migration I/Os on the source servers are substantially alleviated, if not completely eliminated, resulting in a significant performance improvements on both the user performance and the VM live storage migration.

3 Design and Implementation

In this section, we first introduce the design objectives for our SnapMig system, and then present the design and implementation of the proposed SnapMig by introducing the SnapMig architecture, its key functional modules and workflow.

3.1 Design Objectives

The design of SnapMig aims to achieve the following three objectives:

- *Accelerating the VM live storage migration performance:* By leveraging backup servers to migrate the existing VM base image and previous snapshots to the destination server, the VM live storage process can be significantly accelerated.
- *Improving the VM IO performance:* By removing most of the migration I/O requests from the source server, the IO interference between VM IO requests and migration IO requests can be largely reduced. Therefore, VM IO performance will be significantly improved.
- *Providing high extendibility:* SnagMig is quite simple and can be further extended with other functionalities, such as the faster VM reboot in the destination server [33] and the load balancing among backup servers.

3.2 SnapMig Architecture

Fig. 5 shows an architectural overview of the distributed virtualization system that includes three distinct constituent components: *management clients*, *production servers* and *backup servers*, connected by a high speed network. *Management clients* provide a console for system administrators to perform

various management jobs, such as VM creation, VM snapshot backup and VM live migration. They can reside anywhere as long as the network connection to other servers is available. *Production servers* host many live VMs and perform jobs as requested by the management clients. There are two layers in the production servers: virtualization management server (VMS) and the hypervisor module. The VMS listens to the incoming requests from the management clients and performs jobs by calling the corresponding functionalities within the hypervisor module. By design, the VMS, such as open source libvirt [34], can support most modern hypervisors. *Backup servers* store VM snapshots from the production servers through regular backup operations, and they are usually equipped with various functionalities for the purpose of reliability and space efficiency, such as redundant data elimination and data compression [32].

The SnapMig middleware is integrated into this distributed virtualization system and it consists of two functional components: SnapMig Controller and SnapMig Service. The SnapMig Controller contains the *Snapshot Indexing* and *Coordination Service* modules and resides in the production servers, while the *Snapshot Consolidation* and *Snapshot Scheduling* modules are included in the SnapMig Service that resides the backup servers, as shown in Fig. 5. The responsibilities of these four modules are elaborated below.

Algorithm 1. SnapMig VM Live Storage Migration

```
1: input:
2: SS: Source Server
3: DS: Destination Server
4: BS: Backup Server
5: initialization: a thread created for each migrating VM
6: procedure VM_MIGRATION(SS, DS, BS)
7:    Query_snapshots(BS)
8:    if Base_image & latest snapshots in BS then
9:       /*Hit case: most data is in backup server*/
10:      Snapshot_Consolidation(BS, DS)
11:      Snapshot_Scheduling(BS, DS)
12:      State_Migration(SS, DS)
13:      VM_Construction(DS)
14:   else
15:      /*Miss case: act as traditional migration*/
16:      Full_Migration(SS, DS)
17:      VM_Construction(DS)
18:   end if
19: end procedure
```

The *Snapshot Indexing* module tracks the distribution and placement of VM snapshots among the backup servers. There may be multiple copies of a single VM snapshot in several backup servers for increased reliability, especially for VMs with higher priorities. In addition, VM snapshots may be migrated among the backup servers for the purposes of load balancing and reliability. Once the VM live migration starts, this module will query backup servers for the distribution and placement of the current snapshots of the migrating VM and pass it to the Coordination Service module.

The *Coordination Service* module controls the VM live migration workflow. With the snapshots distribution and placement information from the Snapshot Indexing module, the Coordination Service module will instruct the corresponding backup servers to migrate the VM base image and previous snapshots to the destination server. Once these backup servers finish the transmission, this module will invoke the native migration engine within the source server, which will live migrate the latest snapshots and in-memory state to the destination server. Meanwhile, this module will re-configure the VM and its snapshots in the destination server. Finally, it will log the overall migration progress, so that the migration can be resumed upon migration failures.

TABLE 1
An Example of VM Snapshots Distribution and Placement During the Migration Process

Server Name	VM ID	Base File	Snapshots
Source Server	JohnVM	JohnVM.img′	5 6 7
Destination Server	JohnVM	JohnVM.img′	**5 6 7**
Backup Servers	JohnVM	JohnVM.img	1 2 3 4 5 6

The *Snapshot Consolidation* module is designed to eliminate the unnecessary data transmission from the backup servers to the destination servers. It merges some older snapshots to the base image or a single snapshot, before migrating them to the destination server. As shown in Table 1, for example, snapshots 1-4 are unnecessary for the destination server, it would be better to merge them to the base image before the snapshot migration.

The *Snapshot Scheduling* module is an advanced feature designed for the further reduction of the total migration time. With the analysis of the migrating VM's working set, the sequence of blocks in the base image and snapshots can be scheduled, so that hot blocks (i.e., frequently accessed) that are within the VM's working set are migrated before others. Once these hot blocks are ready in the destination server, the source server starts the live migration of the latest state changes, and then the VM can be restarted in the destination before all the blocks are migrated to the destination server. Although this module is under implementation and thus not shown in the evaluation results of Section 4, we expect it will further reduce the total migration time significantly. Algorithm 1 explains the SnapMig scheme in details.

The design of SnapMig is quite flexible. First, it supports all types of modern hypervisors, as SnapMig communicates with hypervisors through the standard Virtualization Management API. Second, the SnapMig scheme is orthogonal to most state-of-the-art VM live storage migration approaches included in the Migration Engine in Fig. 5, and it can be integrated into these existing approaches to further improve the VM live storage migration performance. Finally, the SnapMig scheme is scalable to the architecture of the cluster. Backup servers and production servers can be freely added to or removed from the cluster in the runtime.

3.3 SnapMig Workflow

Fig. 6 shows the workflow of the VM live storage migration in the SnapMig scheme by way of an example. In this example, the migrating VM has a single disk image (named JohnVM.img) and 7 disk snapshots (1-7) in total, as indicated in Table 1. Snapshot 7 is created after the last daily backup operation, so it only resides in the source server. The base image and all other snapshots, 1-6, are already stored in the backup servers through the regular backup

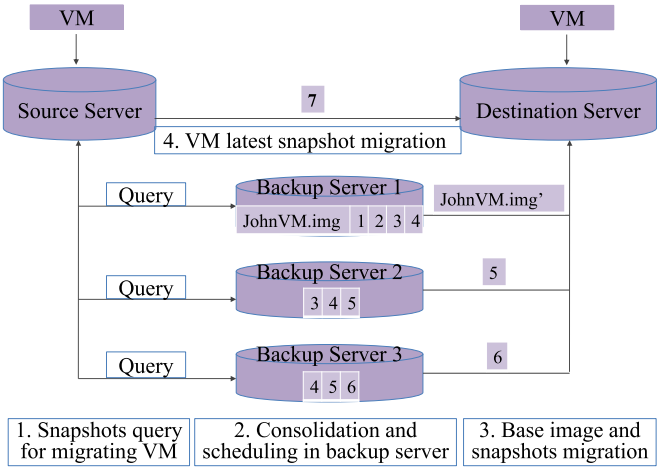


Fig. 6. The workflow of the SnapMig scheme.

operations. There are several copies for the base image and some important snapshots in the backup servers for the reliability purpose. In order to save storage space in the source server and allow users to roll back to recent snapshots, the older snapshots (1-4) are merged to the base image. Only newer snapshots (5-7) are retained in the source server.

As indicated in Section 2.2, FullMig and SelectiveMig are the state-of-the-art migration schemes. In the FullMig scheme, the VM base image and snapshots 5-7 are migrated to the destination server first, and then a conventional migration scheme within the migration engine, such as Dirty Block Tracking or IOMirroring [20], is called to migrate the in-memory state and the latest VM state changes. Different from the FullMig scheme, SelectiveMig only migrates the latest virtual disk image to the destination server and discards all the existing snapshots, such as snapshots 5-7. For instance, if a specific data block is updated both in snapshot 5 and snapshot 7, SelectiveMig only migrates the latest version of this data block (from snapshot 7).

The workflow of our SnapMig scheme can be traced as follows. At the beginning of the migration, the Coordination Service module queries the Snapshot Indexing module for the latest distribution and placement of the migrating VM's snapshots. Then it will notify the corresponding backup servers to start the base image and snapshots migration. These backup servers will consolidate all the unnecessary snapshots first and start the migration to the destination server. Once the base image and snapshots are available in the destination server, the source server will migrate the latest state changes and the in-memory state to the destination. The VM will be resumed in the destination server at the end of the process.

Compared with FullMig and SelectiveMig, SnapMig introduces negligible migration traffic in the source server, including only the read requests of storage system and network transmission. For instance, if a specific data block is updated in the base image and snapshots 5-6, SnapMig does not need to migrate this data block to the destination, since this data block will be migrated by the backup servers. Such significant traffic elimination in the source server will improve the overall system performance in several aspects: shorter migration time, better VM performance, and better multiple VM migrations, which will be evaluated quantitatively in Section 4.

3.4 Data Consistency

The SnapMig system is designed to be able to recover from hardware and software failures or system crashes during the VM live storage migration process, an important requirement for data consistency. In general, all modern hypervisors support the live recovery of running VMs with various mechanisms [26]. However, how to make sure that the live migration process can be resumed smoothly after system crashes is critically important.

The key to the recovery of the VM live storage migration process after a system crash is to restore the key data structures of the SnapMig system in memory correctly and efficiently. In the SnapMig system, we propose to store all the key data structures in battery-backed RAM, a *de facto* standard form of Non-Volatile RAM (NVRAM). Since an Uninterruptible Power Supply (UPS) is the linchpin of the data center backup chain which provides vital protection against power disruptions that would interfere with workloads or even cripple server hardware, storing the key data structures in such a form of NVRAM will be reliable and achieve the desired data consistency. Moreover, since only metadata is recorded as the key data structures for the SnapMig system, it will not incur significant extra hardware cost.

4 Performance Evaluation

In this section we present a detailed evaluation that compares our SnapMig scheme against two state-of-the-art VM migration schemes, FullMig and SelectiveMig. We focus on two key measures of VM live migration efficiency, the VM migration performance (migration time) and VM IO performance (IO throughput of user applications within the migrating and co-located VMs in the source server), under different configurations (e.g., single verses multiple co-located VMs in the source server, single verses multiple migrating VMs).

4.1 Experimental Platform

In order to evaluate the performance of our SnapMig scheme, we implement a light-weight prototype of SnapMig in a cluster to conduct VM live storage migration experiments. The cluster consists of three servers, a source server, a destination server and a backup server. Each server is configured with an Intel Xeon X3440 processor, 8 GB DDR memory and two 1 TB hard drives, 12.04 Ubuntu system, QEMU 2.4.50 system with KVM enabled and libvirt 1.2.20.

The SnapMig prototype is embedded in the libvirt platform [34]. In the source and destination servers, the host system and software are installed in one disk drive and all the VM virtual disk images are stored in a hardware RAID set. The backup server only stores and transfers VM virtual disk images and snapshots. These three servers are connected by a 1 Gbps Ethernet. The hardware information is described in details in Table 2.

4.2 Evaluation Methodology

From a user's perspective, the performance of the running VMs, including migrating VMs and co-located VMs, should not be affected by the migration process, at least to the extend of not violating the Service Level Agreement (SLA). Meanwhile, from a cloud service provider's perspective, the resource consumption for live storage migration should be

TABLE 2
Hardware Specifications in Our Experimental Platform

CPU	Intel(R) Xeon(R) CPU, X3440@2.53 GHz
MotherBoard	Winbond Electronics 0V52N7
Memory	8 GB, AMI CMX8GX3M2A1333C9
Hard Drives	1 TB Seagate ST31000524AS, SATA
RAID	4*160 GB, RocketRaid 2240
Network	1 Gbps Ethernet

minimized, for the purpose of improving overall performance and energy efficiency. For instance, to meet the SLA of user applications running on the migrating VM and/or co-located VMs in the source server, the migration should be completed within a reasonable time period and consume a reasonable amount of network/storage bandwidth of the source server. In this work, we focus on the following metrics to compare SnapMig system with the state-of-the-art solutions: (1) the IO throughput of migrating VMs, (2) the IO throughput of co-located VMs, (3) the migration time. We compare the performance of our SnapMig scheme with two state-of-arts schemes, FullMig and SelectiveMig, in different migration scenarios.

Two common VM live migration scenarios are evaluated in our experiments: migration of a single VM and simultaneous migrations of multiple VMs. Each VM is created with 1 virtual CPU, 2 GB memory, 20 GB disk image, 1 virtual network interface and Ubuntu 12.10 system. There are a number of disk snapshots for each VM, and each snapshot contains some updated data blocks to the VM since its last snapshot. The base image and older snapshots reside in the backup server through the daily backup operations, while newer snapshots sit in the source server only. During the live storage migration, IO requests are issued from the Fio tool [35] to both the migrating VMs and co-located VMs. The performance of the three migration schemes, FullMig, SelectiveMig and SnapMig, are compared under different user application workloads generated by the Fio benchmark.

4.3 Results Analysis

Migration of a Single VM. In this scenario, there is only one VM, mig-vm-1, migrating out of the source server, and the other three co-located VMs, co-located-vm-1, co-located-vm-2 and co-located-vm-3, are running in the source server. As indicated in Fig. 7, the IO throughput of the migrating VM under SnapMig is about 4.61× higher than that under FullMig and SelectiveMig. At the same time, the IO throughput of the co-located VMs under SnapMig is also significantly higher than that under FullMig and SelectiveMig, by about 4.33×, as shown in Fig. 7. Furthermore and importantly, the total migration time of the SnapMig scheme is significantly reduced from that of the FullMig and SelectiveMig schemes, from about 619 to 313 seconds, as shown in Fig. 9. From these results we draw the following observations. First, by leveraging the backup server to migrate the bulk of the VM images, the pressure for the storage device in the source server drops significantly. Therefore, both the migrating VMs and co-located VMs achieve better IO performance, as if there were no migration at all. Second, since the backup server has virtually no running workloads outside of its backup windows, thus much less busy than the source server, the VM images and older snapshots can be migrated from the backup server much faster than from the source server.

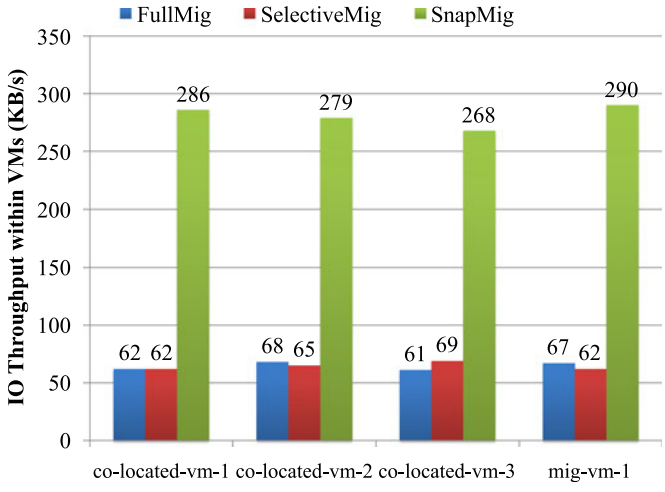


Fig. 7. VM performance comparison in a single-VM migration scenario.

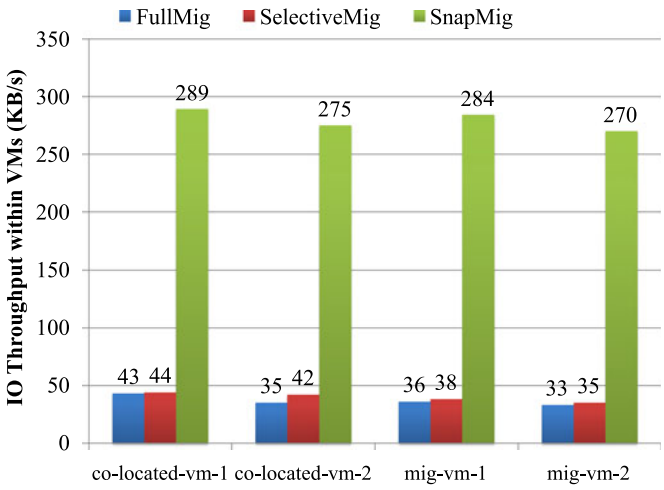


Fig. 8. Performance comparison in a multiple-VM migration scenario.

Simultaneous Migrations of Multiple VMs. In this VM migration scenario, two VMs, mig-vm-1 and mig-vm-2, are migrating from the same source server to the same destination server at the same time. As shown in Figs. 8 and 9, the

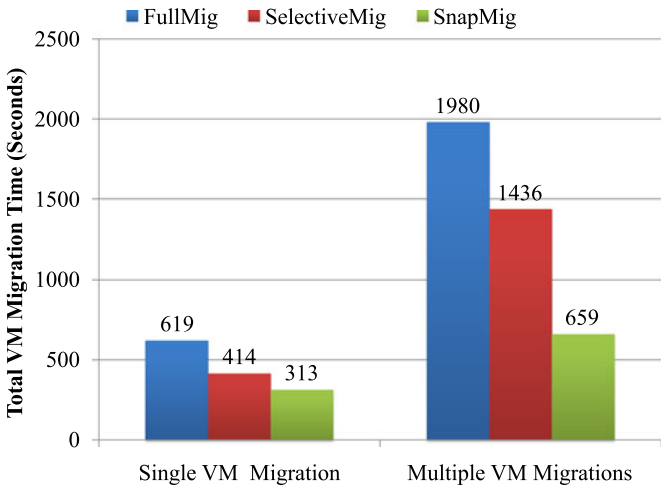
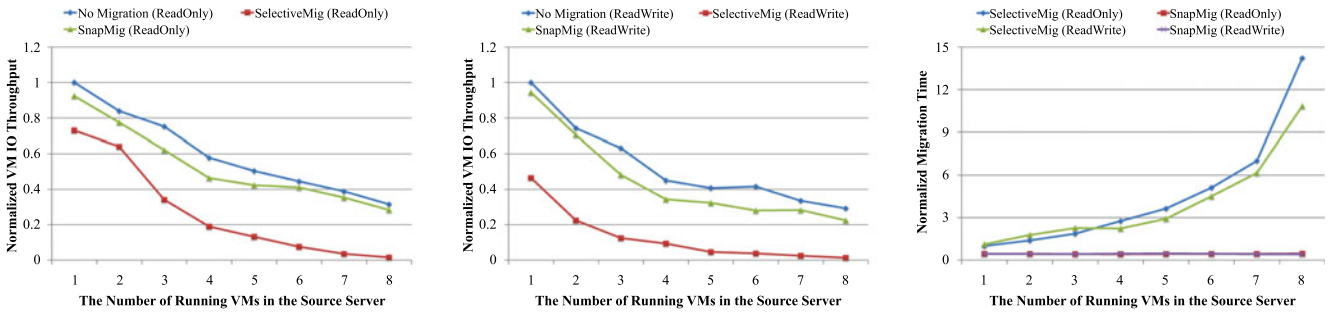


Fig. 9. Comparison of the total migration time of the three VM migration schemes.



(a) VM IO throughput under readonly workload (b) VM IO throughput under readwrite workload (c) VM migration speed

Fig. 10. Sensitivity studies of VM I/O throughput and VM migration speed under different number of co-located running VMs.

throughputs of all the four VMs under our SnapMig scheme, two migrating VMs and two co-located VMs in the source server are about 8× higher than those of these VMs under the FullMig and SelectiveMig schemes. Similar to, but much more pronounced than the case of single-VM migration, the total migration time of SnapMig is drastically reduced from that of FullMig and SelectiveMig, from 2,028 to 694 seconds. From these results, we notice that the performance advantages of the SnapMig system, in both VM performance and migration performance, become more pronounced as more VMs are being migrated simultaneously. For example, its IO throughput advantage over FullMig and SelectiveMig increases from 4.61× to 8× and migration time advantage over FullMig and SelectiveMig increases from $\frac{663}{351} = 1.89\times$ (reduction) to $\frac{2028}{694} = 2.92\times$ (reduction), from migrating one VM to migrating 2 VMs. The main reason is that, with more migrating VMs, there will be more hungry migrating threads competing for the same resources in the source server, which leads to more severe interference between the application IO traffic and the migration IO traffic in the source server in both the FullMig and SelectiveMig schemes and results in more serious degradations of both IO throughput and migration time. The SnapMig scheme, in contrast, almost completely avoids this traffic interference in the source server since the bulk of the migration traffic is diverted or outsourced to the backup server. In other words, while both FullMig and SelectiveMig are very sensitive to the increase in the number of concurrent migrating VMs, SnapMig is relatively insensitive to such increase.

While our evaluation is based on a small cluster in a local network connected by a 1 Gbps Ethernet, SnapMig should work well in a large-scale cluster environment. In such an environment, the contention for network bandwidth will be much more serious. In such a network-constrained environment, VM I/O requests and VM live storage migration I/O requests will interfere with each other in a more significant way. However, SnapMig leverages the existing VM snapshots in the backup servers to outsource the VM live storage migration I/O requests from the source servers to the backup servers. This should further increase the efficiency of SnapMig in a large-scale cluster environment such as a cloud. We will investigate the efficiency of SnapMig in large-scale clusters as our future work.

4.4 Sensitivity Studies

In order to investigate how the number of co-located VMs and the VM workload characteristics affect the performance of the SnapMig system, we conduct a single-VM migration experiment with the number of co-located VMs increasing from 1 to 7. In addition, each VM runs two types of workload: Read-Only workload and Read-Write workload during the migration. Due to the space limitation, we only discuss the comparison between the SnapMig and SelectiveMig schemes. It also shows similar trends for the comparison between the SnapMig and FullMig schemes. The performance results, normalized to that of a single VM running in the source server without any VM migration, are shown in Fig. 10.

From these results, we draw the following observations. First, as the number of co-located running VMs increases, the average VM throughput decreases, as shown in Figs. 10a and 10b. Since the overall storage resource is fixed, the more running VMs are involved, the less storage resource will be available to each VM. Second, in both workloads, the average VM IO performance under the SnapMig migration is very close to that in the No Migration scenario. The reason is that SnapMig effectively leverages the existing VM snapshots in the backup servers that are not affected by the workloads on the source servers. However, the average VM performance drops significantly in the SelectiveMig scenario because many more VMs shares the storage resources on the source servers. Finally, Fig. 10c shows that the migration time in the SnapMig scenario is less sensitive to the number of co-located running VMs. The reason why SnapMig (ReadOnly) and SnapMig (ReadWrite) perform almost the same is that the VM migration time with SnapMig is not affected so much by the workloads on the source servers [36], [37]. Increasing the number of co-located running VMs has little influence on the VM migration time with SnapMig. By contrast, the VM migration time with SelectiveMig soars as the number of co-located running VMs increases. This further confirms that SnapMig introduces negligible extra traffic to the source server.

4.5 Overhead Analysis

SnapMig requires two types of overhead, namely, accesses to the VM snapshots in the backup servers and additional memory space to store the metadata. In other words, SnapMig necessitates additional read traffic on the backup servers/systems to acquire snapshots, on top of the original backup traffic on these backup servers during the backup time window in case of system crashes. In order to reduce the IO pressure on these backup servers, we can isolate the backup window from the migration window by, for instance, giving the migration process much higher priority over the backup operations since the function of SnapMig is orthogonal to that of the backup servers. Second, SnapMig only records the snapshot metadata and keeps track the

migration progress of each snapshot. Thus, SnapMig does not increase the metadata size of the traditional migration schemes. The only difference is that the VM live storage data is mainly migrated from the backup servers rather than from the source servers to the destination server. As a result, the memory space overhead incurred by SnapMig is minimal, relative to the traditional migration schemes.

5 Related Work

VM live migration refers to techniques in which a VM is moved from one host to another with almost zero downtime. It has received great attention from both academia and industry [33], [38]. Based on the migrated resources, the existing studies on the VM live migration can be classified into three categories: Live Memory Migration, Live Storage Migration and Live Multiple-VM Migrations. In what follows, we summarize the representative studies within each category and outline their differences with our SnapMig scheme.

Live VM Memory Migration. VM live memory migration is very popular in the shared-storage environment, where both source and destination servers retain the access to the shared storage system. Among a variety of VM states being transfered, such as CPU state, network state, memory state and device sate, VM's memory state usually dominates in the migration time and a number of approaches have been proposed to accelerate the memory state migration [39]. Changyeon et al. [40] propose to track the duplicated memory pages in the source server in the runtime. When migration is triggered, instead of migrating these duplicated pages over the rate-limited connection to the destination server, the destination server directly fetches these pages from the shared storage server. Therefore, the total data transmission time is reduced significantly, and the live migration performance is improved as well [40]. Hai et al. [41], [42], [43] propose to classify memory pages to several types according to different characteristics, such as high word similarity, low word similarity, a large number of zero bytes, and then adopt different compression algorithms to compress memory pages with different properties.

Live VM Storage Migration. In a non-shared-storage environment, such as distributed storage or local storage systems, VM live migration must also include VM virtual disk images whose size is much larger all the other VM stages mentioned above combined. In other words, in such an environment, the time to live migrate VM storage (virtual disk images) dominates the total migration time. Shrinker [22] is a distributed system that is capable of migrating a virtual cluster over WAN. It has two built-in services: Coordination Service (in the source site) and Indexing Service (in the destination site). The Coordination Service tracks the hash values of memory pages and virtual disk blocks that have already been transferred to the destination site, so that hypervisors in the source side can perform data deduplication by replacing duplicated transmission of memory pages and disk blocks with their corresponding hash values. The Indexing Service records the hash values and the location information of the memory pages and disk blocks in the destination side. Hypervisors in the destination can reconstruct the VM's memory and virtual disk images with the communication between Indexing Service and other hypervisors that hold the real data. Zhou et al. [15] take the speed discrepancy between HDDs and SSDs, and the wear-out issue of SSDs into consideration in order to optimize the live storage migration. Ali et al. [44] build a VM live storage migration system, called XvMotion, to migrate VMs over long distances across heterogeneous systems, with performance similar to that of VM migration in Local Area Network.

Live Multiple-VM Migrations. Given the wide deployment of VMs in current cloud data centers, it is not uncommon to migrate multiple VMs from/to a single server simultaneously. VMScatter [16] is a multicast-based VM live migration system, which can efficiently migrate a group of VMs from one shared source server to multiple destination servers. Timothy et al. [45] design and implement the Sandpiper system that contains two components: a hotspot detection algorithm and a hotspot migration algorithm. The hotspot detection algorithm will decide when to migrate VMs, while the hotspot migration algorithm will determine where to migrate and how much resources to allocate after the migration. Live Gang Migration [46] is inspired by the fact that co-located VMs often have many identical memory pages, such as the same operating system, applications and libraries, same Java Virtual Machine. In this approach [46], identical memory pages will be identified and deduplicated prior to the transmission of VM state, so that only a single memory page copy needs to be migrated.

However, none of the above studies leverages the existence of the VM snapshots to improve VM and migration performance. They focus on the performance optimization techniques of the aggregate IO streams in the source server, such as leveraging the fast read performance of SSD, removing the redundant data transmission and so on. In contrast, SnapMig takes a different approach by leveraging the existing base images and snapshots already stored in the largely idle backup servers for the transmission of the dominant VM state information, VM storage state (disk images). Thus, SnapMig is orthogonal to and can be used to further improve the above techniques. Moreover, since a large portion of the migration traffic is removed from the source server, the contention between the user I/Os and the migrations I/Os is substantially alleviated. Thus both the VM live storage migration performance and the user I/O performance are improved.

6 Conclusion

For the purposes of load balance, hardware maintenance and system upgrade, VM live storage migration is becoming increasingly important and indispensable in the Cloud. However, conventional VM migration approaches induce significant extra storage and network traffic to the source server that is already heavily loaded or scheduled for upgrade or repair. In this paper, we present SnapMig, a novel VM live storage migration middleware based on the observation of idle VM snapshots in backup servers. SnapMig effectively leverages the backup servers for the bulk migration of VM disk images and previous snapshots, which significantly reduces the migration traffic to the source server. Therefore the VM IO performance and migration performance are improved simultaneously. The experiments conducted on our lightweight prototype implementation of SnapMig indicate that SnapMig significantly outperforms existing live migration approaches, such as FullMig and SelectiveMig.

SnapMig is an ongoing research project that offers several directions for future research. Intuitively, the key to the

reduction of storage migration time is how to maximize the effective migration bandwidth and minimize the amount of data transferring. First, we will exploit the data redundancy or similarity within VM live storage migration of VMM to eliminate unnecessary transferring of redundant and similar data. Thus, a judicious combination of data deduplication and delta compression is preferred to maximize the effective migration bandwidth. Second, how to extract the liveness information of filesystem blocks of virtual disks, and migrate the live blocks only to reduce the amount of data transferring? By doing so, it can significantly reduce the storage migration time by minimizing the amount of data transferring.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 61772439, No. U1705261, No. 61472336 and No. 61402385, the US National Science Foundation under Grant No. CCF-1704504 and CCF-1629625.

References

[1] Amazon still dominates the 16 billion dollars cloud market. (2015, Feb.). [Online]. Available: http://www.businessinsider.com/synergy-research-amazon-dominates-16-billion-cloud-market-2015–2

[2] Netflix shuts down its last data center, but it still runs a big it operation. (2015, Aug.). [Online]. Available: http://arstechnica.com/information-technology/2015/08/netflix-shuts-down-its-last-data-center-but-still-runs-a-big-it-operation/

[3] The virtualization techniques. (2016, Sep.). [Online]. Available: https://www.fluxlabs.net/solutions/virtualization

[4] S. Rampersaud and D. Grosu, "Sharing-aware online virtual machine packing in heterogeneous resource clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 2046–2059, Jul. 2017.

[5] Z. Shen, et al., "VMAR: Optimizing I/O performance and resource utilization in the cloud," in *Proc. ACM/IFIP/USENIX 14th Int. Middleware Conf.*, 2013, pp. 183–203.

[6] B. Nicolae and F. Cappello, "A hybrid local storage transfer scheme for live migration of I/O intensive workloads," in *Proc. 21st Int. Symp. High-Perform. Parallel Distrib. Comput.*, 2012, pp. 85–96.

[7] T. Knauth and C. Fetzer, "VeCycle: Recycling VM checkpoints for faster migrations," in *Proc. ACM/IFIP/USENIX 16th Int. Middleware Conf.*, 2015, pp. 210–221.

[8] P. Patel, et al., "Ananta: Cloud scale load balancing," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2013, pp. 207–218.

[9] S. Angel, H. Ballani, T. Karagiannis, G. Ohea, and E. Thereska, "End-to-end performance isolation through virtual datacenters," in *Proc. 11th USENIX Conf. Operating Syst. Des. Implementation*, 2014, pp. 233–248.

[10] W. Zhang, H. Xie, and C. Hsu, "Automatic memory control of multiple virtual machines on a consolidated server," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 2–14, Jan.–Mar. 2017.

[11] L. Cui, B. Li, Y. Zhang, and J. Li, "HotSnap: A hot distributed snapshot system for virtual machine cluster," in *Proc. 27th USENIX Conf. Large Installation Syst. Admin.*, 2013, pp. 59–74.

[12] J. Li, H. Liu, L. Cui, B. Li, and T. Wo, "iROW: An efficient live snapshot system for virtual machine disk," in *Proc. 18th Int. Conf. Parallel Distrib. Syst.*, 2012, pp. 376–383.

[13] W. Xiao, Q. Yang, J. Ren, C. Xie, and H. Li, "Design and analysis of block-level snapshots for data protection and recovery," *IEEE Trans. Comput.*, vol. 58, no. 12, pp. 1615–1625, Dec. 2009.

[14] EMC recoverpoint for virtual machines gives administrators more agility and control. (2014, Aug.). [Online]. Available: https://www.emc.com/collateral/analyst-reports/esg-wp-emc-recoverpoint-for-vms.pdf

[15] R. Zhou, F. Liu, C. Li, and T. Li, "Optimizing virtual machine live storage migration in heterogeneous storage environment," in *Proc. 9th Int. Conf. Virtual Execution Environ.*, 2013, pp. 73–84.

[16] L. Cui, et al., "VMScatter: Migrate virtual machines to many hosts," in *Proc. 9th Int. Conf. Virtual Execution Environ.*, 2013, pp. 63–72.

[17] J. Zheng, T. Ng, K. Sripanidkulchai, and Z. Liu, "Comma: Coordinating the migration of multi-tier applications," in *Proc. 10th Int. Conf. Virtual Execution Environ.*, 2014, pp. 153–164.

[18] C. Ng, M. Ma, T. Wong, P. Lee, and J. Lui, "Live deduplication storage of virtual machine images in an open-source cloud," in *Proc. ACM/IFIP/USENIX 12th Int. Middleware Conf.*, 2011, pp. 80–99.

[19] Y. Yang, H. Jiang, B. Mao, L. Tian, Y. Yang, and J. Qian, "WAIO: Improving virtual machine live storage migration for the cloud by workload-aware IO outsourcing," in *Proc. 7th Int. Conf. Cloud Comput. Technol. Sci.*, 2015, pp. 314–321.

[20] A. Mashtizadeh, E. Celebi, T. Garfinkel, and M. Cai, "The design and evolution of live storage migration in VMware ESX," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2011, pp. 1–14.

[21] J. Zheng, T. Ng, and K. Sripanidkulchai, "Workload-aware live storage migration for clouds," in *Proc. 7th Int. Conf. Virtual Execution Environ.*, 2011, pp. 133–144.

[22] P. Riteau, C. Morin, and T. Priol, "Shrinker: Improving live migration of virtual clusters over WANs with distributed data deduplication and content-based addressing," in *Proc. 17th Int. Conf. Parallel Comput.*, 2011, pp. 431–442.

[23] A. Yoshihisa, G. Roxana, J. Kaustubh, and S. Mahadev, "Urgent virtual machine eviction with enlightened post-copy," in *Proc. 12th Int. Conf. Virtual Execution Environ.*, 2016, pp. 51–64.

[24] L. Cui, T. Wo, B. Li, J. Li, B. Shi, and J. Huai, "PARS: A page-aware replication system for efficiently storing virtual machine snapshots," in *Proc. 11th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, 2015, pp. 215–228.

[25] L. Cui, et al., "HotRestore: A fast restore system for virtual machine cluster," in *Proc. 28th USENIX Conf. Large Installation Syst. Admin.*, 2014, pp. 1–16.

[26] Understanding virtual machine snapshots in VMware ESXi and ESX. (2015, Mar.). [Online]. Available: http://kb.vmware.com

[27] W. Zhang, T. Yang, G. Narayanasamy, and H. Tang, "Low-cost data deduplication for virtual machine backup in cloud storage," in *Proc. 5th USENIX Conf. Hot Top. Storage File Syst.*, 2013, pp. 12–12.

[28] W. Zhang, H. Tang, H. Jiang, T. Yang, X. Li, and Y. Zeng, "Multi-level selective deduplication for VM snapshots in cloud storage," in *Proc. 5th IEEE Int. Conf. Cloud Comput.*, 2012, pp. 550–557.

[29] Linux kvm. (2016, Sep.). [Online]. Available: http://www.linux-kvm.org/page/Main_Page

[30] Microsoft: About virtual machine snapshots. (2017, Feb.). [Online]. Available: https://technet.microsoft.com/en-us/library/dd851843(v=ws.11).aspx

[31] VMware: Using the snapshot. (2017, Feb.). [Online]. Available: https://www.vmware.com/support/ws4/doc/preserve_snapshot_ws.html

[32] B. Zhu, K. Li, and R. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proc. 6th USENIX Conf. File Storage Technol.*, 2008, pp. 1–14.

[33] S. Nathan, U. Bellur, and P. Kulkarni, "On selecting the right optimizations for virtual machine migration," in *Proc. 12th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, 2016, pp. 37–49.

[34] libvirt: The virtualization API toolkit. (2017, Feb.). [Online]. Available: http://libvirt.org/index.html

[35] Fio. (2016, Sep.). [Online]. Available: https://github.com/axboe/fio

[36] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "WorkOut: I/O Workload outsourcing for boosting the RAID reconstruction performance," in *Proc. 7th USENIX Conf. File Storage Technol.*, Feb. 2009, pp. 239–252.

[37] S. Wu, H. Jiang, and B. Mao, "Proactive data migration for improved storage availability in large-scale data centers," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2637–2651, Sep. 2015.

[38] Q. Chen, L. Liang, Y. Xia, and H. Chen, "Mitigating sync amplification for copy-on-write virtual disk," in *Proc. 14th USENIX Conf. File Storage Technol.*, 2016, pp. 241–247.

[39] M. Nelsonand, B. Limand, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proc. USENIX Annu. Tech. Conf.*, 2005, pp. 391–394.

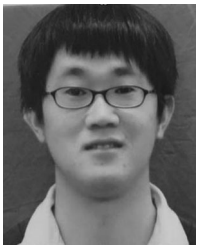
[40] C. Jon, E. Gustafsson, J. Son, and B. Egger, "Efficient live migration of virtual machines using shared storage," in *Proc. 9th Int. Conf. Virtual Execution Environ.*, 2013, pp. 41–50.

[41] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in *Proc. 18th Int. Symp. High Perform. Distrib. Comput.*, 2009, pp. 101–110.

[42] H. Liu, C. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machine," in *Proc. 20th ACM Int. Symp. High Perform. Distrib. Comput.*, 2011, pp. 101–110.

[43] H. Liu, H. Jin, X. Liao, W. Deng, B. He, and C. Xu, "Hotplug or ballooning: A comparative study on dynamic memory management techniques for virtual machines," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1350–1363, May 2015.

[44] A. Mashtizadeh, M. Cai, G. Tarasuk-Levin, R. Koller, T. Garfinkel, and S. Setty, "XvMotion: Unified virtual machine migration over long distance," in *Proc. USENIX Conf. USENIX Annu. Tech.*, 2014, pp. 97–108.
[45] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. 4th USENIX Symp. Netw. Syst. Des. Implementation*, 2007, pp. 229–242.
[46] U. Deshpande, X. Wang, and K. Gopalan, "Live gang migration of virtual machines," in *Proc. 20th Int. Symp. High Perform. Distrib. Comput.*, 2011, pp. 135–146.



Yaodong Yang received the BSc degree in computer engineering from Tianjin University, China, in 2008, the MASc degree in computer engineering from Huazhong University of Science and Technology, China, in 2011, and the PhD degree from the Computer Science and Engineering Department, University of Nebraska-Lincoln, in 2016. He is an intern in Tintri Storage and familiar with their flash products. His research interests include flash-based storage systems, VM storage migration, and cloud storage. He is currently a software engineer with Microsoft, Redmond.

Bo Mao received the BSc degree in computer science and technology from Northeast University, in 2005 and the PhD degree in computer architecture from the Huazhong University of Science and Technology, in 2010. His research interests include storage system, flash-based SSDs and disk arrays, data deduplication, cloud storage, and storage reliability. He was a postdoc researcher with the University of Nebraska-Lincoln between 2010 and 2013. After that he joined in the Software School of Xiamen University and becomes an associate professor since August 2015. He has more than 40 publications in international journals and conferences, including the *IEEE Transactions on Computers*, the *ACM Transactions on Storage*, the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, USENIX FAST, IPDPS, ICS, Cluster, USENIX LISA, MASCOTS, ICCD, and ICPADS. His research has been supported by NSFC and Huawei. He is a member of the IEEE, ACM, and USENIX.

Hong Jiang received the BSc degree in computer engineering from Huazhong University of Science and Technology, Wuhan, China, in 1982, the MASc degree in computer engineering from the University of Toronto, Toronto, Canada, in 1987, and the PhD degree in computer science from the Texas A&M University, College Station, Texas, in 1991. He is currently chair and Wendell H. Nedderman Endowed professor of Computer Science and Engineering Department, University of Texas at Arlington. Prior to joining University of Texas at Arlington, from January 2013 to August 2015, he served as a program director with National Science Foundation and he was with the University of Nebraska-Lincoln since 1991, where he was Willa Cather professor of Computer Science and Engineering. He has graduated 16 PhD students who upon their graduations either landed academic tenure-track positions in PhD-granting US institutions or were employed by major US IT corporations. His present research interests include computer architecture, computer storage systems and parallel I/O, high-performance computing, big data computing, cloud computing, performance evaluation. He recently served as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. He has more than 300 publications in major journals and international Conferences in these areas, including the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *Proceedings of IEEE*, the *ACM Transactions on Architecture and Code Optimization*, the *ACM Transactions on Storage*, the *Journal of Parallel and Distributed Computing*, ISCA, MICRO, USENIX ATC, FAST, EUROSYS, SOCC, LISA, SIGMETRICS, ICDCS, IPDPS, MIDDLEWARE, OOPLAS, ECOOP, SC, ICS, HPDC, INFOCOM, ICPP, etc., and his research has been supported by NSF, DOD, the State of Texas and the State of Nebraska, and industry. He is a fellow of the IEEE and member of the ACM.

Yuekun Yang received the BSc degree in computer science from the University of Nebraska-Lincoln, in 2016. He was an undergraduate research assistant with Abacus Distributed Storage Lab, University of Nebraska-Lincoln, working on the Virtual Machine Live Storage Migration projects. He is currently a software engineer with Megvii (Face++) Research Lab, in Redmond.

Hao Luo received the BSc degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2011 and the PhD degree in computer science and engineering from the University of Nebraska, Lincoln, in 2016. He is currently a software engineer with Twitter. His research interests include flash-based storage systems and cloud computing. He has published papers in the *IEEE Transactions on Computers*, ICS, MSST, HotStorage.

Suzhen Wu received the BSc and PhD degrees in computer science and technology and computer architecture from Huazhong University of Science and Technology, in 2005 and 2010 respectively. She is an associate professor in the Computer Science Department, Xiamen University since August 2014. Her research interests include computer architecture and storage system. She has more than 40 publications in journal and international conferences, including the *IEEE Transactions on Computers*, the *ACM Transactions on Storage*, the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, USENIX FAST, USENIX LISA, IPDPS, ICS, ICCD, MASCOTS, and ICPADS. Her research has been supported by NSFC, Huawei, Intel and Inspur. She is a member of the IEEE and ACM.