Optimizing Virtual Machine Live Storage Migration in Heterogeneous Storage Environment

Ruijin Zhou
Intelligent Design of Efficient Architectures Laboratory
University of Florida
USA
zhourj@ufl.edu

Fang Liu
State Key Laboratory of High Performance Computing
National University of Defense Technology
China
liufang@nudt.edu.cn

Chao Li
Intelligent Design of Efficient Architectures Laboratory
University of Florida
USA
chaol@ufl.edu

Tao Li
Intelligent Design of Efficient Architectures Laboratory
University of Florida
USA
taoli@ece.ufl.edu

Abstract

Virtual machine (VM) live storage migration techniques significantly increase the mobility and manageability of virtual machines in the era of cloud computing. On the other hand, as solid state drives (SSDs) become increasingly popular in data centers, VM live storage migration will inevitably encounter heterogeneous storage environments. Nevertheless, conventional migration mechanisms do not consider the speed discrepancy and SSD's wear-out issue, which not only causes significant performance degradation but also shortens SSD's lifetime. This paper, for the first time, addresses the efficiency of VM live storage migration in heterogeneous storage environments from a multi-dimensional perspective, i.e., user experience, device wearing, and manageability. We derive a flexible metric (migration cost), which captures various design preference. Based on that, we propose and prototype three new storage migration strategies, namely: 1) Low Redundancy (LR), which generates the least amount of redundant writes; 2) Source-based Low Redundancy (SLR), which keeps the balance between IO performance and write redundancy; and 3) Asynchronous IO Mirroring, which seeks the highest IO performance. The evaluation of our prototyped system shows that our techniques outperform existing live storage migration by a significant margin. Furthermore, by adaptively mixing our proposed schemes, the cost of massive VM live storage migration can be even lower than that of only using the best of individual mechanism.

Categories and Subject Descriptors D.4.8 [Operating Systems]: Performance; C.4 [Computer System Organization]: Performance of Systems

General Terms Management, Performance, Design

Keywords Live VM Storage Migration, Solid State Drive, Virtualization

VEE'13, March 16–17, 2013, Houston, Texas, USA.

1. Introduction

Nowadays, virtualization technology has been widely adopted as the base infrastructure for cloud computing. Major cloud providers, such as Amazon (EC2) [1] and Microsoft (Azure) [2], are selling their computing resources in the form of virtual machines (VMs). Load balancing has become essential for effectively managing large volumes of VMs in cloud computing environment. The cornerstone for moving virtual machines on the fly is the VM live migration, which only transfers CPU and memory states of VMs from one host to another. To allow the movement of persistent storage with VMs, several live storage migration techniques have been proposed, including Dirty Block Tracking (DBT) and IO Mirroring [10][11][12]. VM live storage migration significantly increases the mobility and manageability of virtual machines during disaster recovery, storage maintenance, and storage upgrade.

Meanwhile, Flash-based solid state drives (SSDs) have become one of the most popular storage media due to their high performance, silent operations and shock resistance [20, 21]. With the decrease in price, they become more affordable to be used in data centers. Currently, many leading Internet service provision companies, such as Facebook, Amazon and Dropbox, are starting to integrate SSDs into their cloud storage systems [3][4][5]. The storage media for data centers becomes more diverse as both SSDs and HDDs are being used to support cloud storage. Consequently, storage management, especially VM live storage migration, becomes more complex and challenging.

Although SSDs deliver higher IO performance, their limited lifetime is an inevitable issue. Our analysis shows that existing VM live storage migration schemes do not fully exploit the high performance characteristics of SSDs but aggravate the wear out problem. Even worse, during massive storage migrations, SSDs will be worn out significantly due to large volume of write operations. In this paper, we address the efficiency/cost of VM live storage migration (Migration Cost, MC) in heterogeneous storage environments from a multi-dimensional perspective, which incorporates user experience (IO penalty), cluster management (migration time) and device usage (degree of wear). The weights on IO penalty and SSD lifetime are also considered to reflect different design preferences. We propose and prototype three VM

live storage migration mechanisms to minimize the migration cost, namely: 1) Low Redundancy (LR), which generates near zero redundant writes; 2) Source-based Low Redundancy (SLR), which aims to leverage faster source disk while still maintaining low redundancy merit; 3) Asynchronous IO Mirroring (AIO), which targets high IO performance. The empirical evaluation of our prototyped systems shows that they yield stable and short disk downtimes (around 200ms). Although the cost varies with different weights and storage media, on average, the migration costs for LR, SLR and AIO are 51%, 22% and 21% lower than those for traditional methods (i.e. DBT and IO Mirroring). Furthermore, by adaptively invoking our schemes during massive storage migration, the cost can be further reduced by 48% compared to using the best individual mechanism.

The rest of this paper is organized as follows: Sections 2 and 3 provide background and motivation for this work. Section 4 describes the evaluation metric. Section 5 discusses our proposed designs. Sections 6 and 7 present our prototypes and experimental results. Section 8 discusses related work and Section 9 concludes the paper.

2. Background

2.1. VM Live Storage Migration Techniques

Live storage migration for virtual machines is defined as the migration of VM disk images without service interruption to the running workload. The two mainstream techniques are dirty block tracking (DBT) and IO Mirroring. The DBT technique, which is widely adopted by many VM vendors (e.g. Xen and VMware ESX), is a well-known mechanism that uses bitmap to track write requests while the VM image is being copied. Once the entire image is copied to the destination, a merge process is initiated to patch all the dirty blocks (i.e. data blocks that are recorded in bitmap) from the original image to the new image. In order to prevent further write requests, the VM is paused until all the dirty blocks are patched to the new disk image. To mitigate downtime introduced by the merge process, incremental DBT, which keeps the VM running while iteratively patching dirty blocks to the new image, is proposed and used in several projects [11][12][13][14]. If the number of dirty blocks is stable for several iterations, the VM is suspended and the remaining dirty blocks are copied to the destination. Nevertheless, incremental DBT also has disadvantage: in case that the number of dirty blocks are not converged due to intensive write requests, the migration time and even the downtime can be significantly long. Note that in this paper, we refer incremental DBT as DBT.

To address the issue of long migration time and downtime, VMware proposed IO Mirroring technique [10] to eliminate the iteratively merge process. With IO Mirroring, all the write requests to the data blocks that have been copied will be duplicated and issued to both source and destination disks. The two write requests are synchronized and then the write completion acknowledgement is asserted (synchronous write). Write requests to the data blocks that have not yet been copied will only be issued to the source disk while the writes to the data blocks that are currently being copied will be buffered and later issued to both source and destination when the being copied phase completes. By doing so, the data blocks will always be synchronized during the migration process. Note that once the process of copying VM disk image completes, merging is not needed, which leads to shorter migration time and lower downtime. However, IO Mirroring also raises some concerns: 1) workload IO performance is limited by the slower disk due to the synchronized write requests; 2) since the disk bandwidth is consumed by the duplicated IO requests, the progress of copying the VM image will be slowed down.

2.2. Storage Migration in Heterogeneous Storage Environments

Historically, mechanical hard disk drives (HDDs) are used as the primary storage media due to their large capacity and high stability in the long run. Recently, solid-state drives (SSDs), which have high IO performance [20], are emerging as promising storage media. The IOPS (IO per second) for VM running on SSDs in our experiments is 3.3X higher than that on HDDs. However, SSDs also have their limitations such as low capacity, high price tag, and limited lifetime. The more the writes and erases are performed, the shorter the remaining SSD lifetime will be [7, 22]. In the commercial market, cloud storage providers, such as Morphlabs, Storm on Demand, CloudSigma and CleverKite, are selling the SSD powered cloud [4]. On the other hand, device manufacturers, such as Intel and Samsung, are researching on reliable SSD for data centers [6, 27]. Thus, from the perspective of both seller and manufacturer, SSDs have been accredited as an indispensable component for cloud and data center storage. A data center will be equipped with several disk arrays. Some of the disk arrays are SSDs while the others are HDDs. Those disk arrays are connected to servers via Fibre Channel [8, 9, 26]. Our work focuses on the storage migration between different disk arrays.

VM live storage migration will be more sophisticated and challenging on heterogeneous storage environments. For instance, if a user requests more disk space, his/her VM image may need to be migrated from small capacity disk (SSD) to large capacity disk (HDD). On the other hand, if the user requests to upgrade IO performance, his/her VM image may need to be migrated from slow disk (HDD) to fast disk (SSD). Since VM live storage migration will inevitably be performed on various types of storage media, it should consider the characteristics of different storage devices, such as the high bandwidth, limited lifetime for SSD and the low access speed, large capacity for HDDs. Nevertheless, existing live storage migration schemes do not take the underlying storage media into consideration, which manifests several disadvantages, such as: 1) not fully exploiting the high performance of SSDs, 2) having longer migration time since the redundant write requests occupy a significant fraction of IO bandwidth, 3) quickly wearing down SSDs and reducing the remaining SSD lifetime. Even worse, large volume of redundant IO operations, which are generated during massive storage

migration, will not only saturate the disk bandwidth but also severely affect the SSD's lifetime.

3. Performance Characterization

This section analyzes the behaviors of existing live storage migration schemes (e.g. DBT and IO Mirroring) in heterogeneous storage environments.

3.1. A Characterization of the Two Basic Processes for VM Live Storage Migration

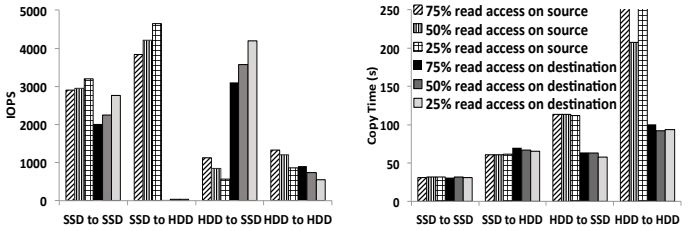
In general, VM live storage migration involves two processes, namely 1) copy process that moves the VM image and 2) VM IO request handling process that ensures consistency between the two disk images. During storage migration, processes 1) and 2) compete for IO bandwidth, which largely affects the performance. Table 1 shows the performance of copying a 5GB image of an idle VM and Table 2 shows the measured IOPS of the same VM that is running but is not being migrated. As expected, the SSD effectively speeds up the copy process and the VM on SSDs achieves much higher IOPS than that on HDDs. Nevertheless, the resource competition between image copy and VM IO requests still exists even when SSDs are involved. Note that the VM IO requests can be issued to source or destination during the migration. The performance characteristics on copying VM image with running workloads are shown in Figures 1 (a) and (b). SSDs have faster access speed and better capability of handling intensive IO requests. In the case where both a SSD and a HDD are involved, the IOPS will be higher and the copy time will be shorter if the VM IO requests are directed to the SSD. In the case where both source and destination are SSDs, the copy time and IOPS will be similar no matter where the VM IO requests are issued. In the case where both source and destination are HDDs, higher IOPS can be achieved at the cost of longer migration time.

Table 1. The time to copy a 5GB idle VM image

SSD to SSD	SSD to HDD	HDD to SSD	HDD to HDD
30s	59s	55s	81s

Table 2. The IOPS of running VM

	75% Read	50% Read	25% Read
VM on SSD	5500	5900	6495
VM on HDD	2230	1744	1445



(a) IOPS of VM (b) Time to copy VM image

Figure 1. Copying VM image while workloads are running inside

In the case that a VM is migrated from a HDD to a SSD and all the IO requests from VM are issued to the source disk (HDD) as in DBT, the performance for both copy process and IO requests will be significantly low, as shown in Figure 1. The reason is that the reads from the copy process and the IO requests from VM will saturate the bandwidth of the HDD. In the case that a VM is migrated from a SSD to a HDD, issuing all the VM IO requests to the source disk (SSD) will yield high IOPS and short copy time because the SSD has better capability for handling large volume of IO accesses. However, simply filtering all the IO requests to the SSD will exacerbate the degree of wearing for the SSD, which shortens the remaining SSD lifetime. Thus, blindly using SSD is also not a wise decision.

To summarize, when VM storage migration occurs in the storage environment that involves SSDs, redirecting the workload IO requests to SSDs will benefit the migration time and IOPS but the SSDs will be worn out quickly. When a VM is migrated between HDDs, there is trade off between migration time and IOPS.

3.2. A Characterization of Existing Live Storage Migration Schemes

We use Xen as our virtual machine monitor and implement the two existing storage migration techniques in Xen blktap2 modules (details in Section 6).

There are three well-known metrics to measure the performance of live storage migration: 1) downtime, 2) migration time, and 3) IO penalty. Downtime measures the time it takes to pause the VM and switchover between source and destination disks. Migration time represents the overall time it takes to accomplish the storage migration operation, which should be minimized to guarantee smooth and quick storage maintenance. IO penalty shows the performance degradation the user will experience during the VM live storage migration.

As can be seen in Table 3, DBT tends to have longer downtime than IO Mirroring because DBT needs to patch the last copy of dirty blocks during the downtime period while this is not necessary for IO Mirroring. Besides, when varying the underlying storage media, the downtime of DBT is less stable than that of IO Mirroring. Worse, when the destination is slower than the source (e.g. from SSD to HDD), DBT takes long time or even fails to complete since the last iteration of the merge process writes dirty data blocks to the slower disk.

Table 3. The Downtime of Migrating 30GB VM Image while Running IOmeter with 50% Reads / 50% Writes

	SSD to SSD	SSD to HDD	HDD to SSD	HDD to HDD
DBT	232ms	4000ms	266ms	900ms
IO Mirroring	198ms	249ms	298ms	199ms

Migration time for existing schemes is shown in Figure 2. We also use two emulated scenarios for comparison purposes. Copying disk image while VM IO requests are only issued to the source is denoted as Emulated_S and copying disk image while VM IO requests are only issued to the destination is referred as Emulated_D. In terms of migration time, the biggest difference between IO Mirroring and DBT is the iterative merge process. As

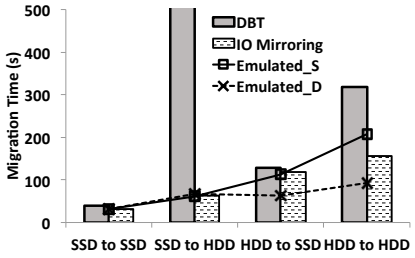


Figure 2. The migration time for a 5GB VM image under heterogeneous storage environment (Iometer is running inside the VM with 50% read, 50% write)

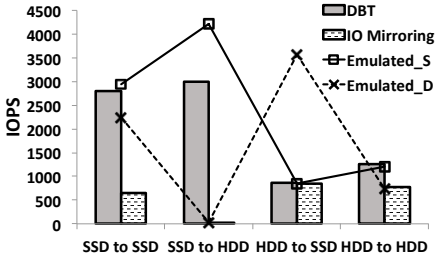


Figure 3. The IOPS for migrating VM images under heterogeneous storage environment (IOmeter is used to measure IOPS)

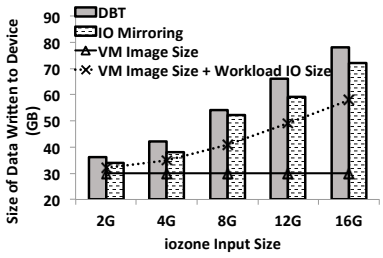


Figure 4. The total size of data that written to device when migrating 30GB VM from SSD to SSD (IOzone is set to perform write and rewrite operation)

can be seen in Figure 2, DBT exhibits longer migration time than IO Mirroring because it needs to iteratively merge dirty blocks to destination after copying the image. The slower the destination storage media are, the longer the merge process will be. Both migration schemes manifest longer migration time compared to the better of the two emulated scenarios.

When using IO Mirroring scheme, the IOPS inside the VM highly depends on the slowest disk due to synchronous write. On the other hand, the IOPS inside the VM when using DBT scheme only depends on the source disk. As can be seen from Figure 3, when the source disk is a SSD, DBT has advantage over IO Mirroring in terms of IO performance. If the source is a HDD, both DBT and IO Mirroring yield similar IOPS. However, neither of them reaches the performance of the two emulated scenarios (indicated by the dash-cross or solid-square lines, whichever is higher). In other words, conventional schemes do not fully exploit the high performance of SSDs.

Since introducing a large volume of write and erase cycles will unavoidably diminish the lifetime of SSDs [7], we add another measurement: the amount of data that is written to SSD, in our characterization. If only the copying of a VM disk image is considered (e.g. experiments shown in Figure 1), then the total amount of data written to SSD is determined by the workload IO traffic plus the VM image size. The live storage migration schemes introduce overhead and redundant data writes, further wearing off SSDs and shortening the SSD's lifetime.

Figure 4 shows the amount of data written to disk when migrating VM images from SSDs to SSDs. As can be seen, both mechanisms generate extra amount of data writes compared to simply copying the disk image of the running VM (dashed line). The heavier the workload's IO traffic is, the higher the extra volume of data write (the amount that exceeds the dashed line) will be. For DBT, the redundant data writes come from the merge process; for IO Mirroring, the redundant data writes come from the duplicated writes to data blocks that have been copied. DBT tends to have more redundant data writes than IO Mirroring.

To sum up, a good live storage migration technique in heterogeneous storage environments should have negligible downtime, short migration time, low IO penalty, and less redundant writes to the SSD. The major issue with contemporary methods is that they do not take the underlying device characteristics into consideration. We believe that once the underlying storage becomes heterogeneous, new methods for live storage migration are needed and our analysis shows that there is still plenty of room for improvement in terms of migration time, IO penalty, and redundant writes. In this paper, we are motivated to explore better VM live storage migration schemes.

4. Metric for Live Storage Migration in Heterogeneous Environments

VM live storage migration behavior has impacts on three aspects: 1) VM user experience (IO penalty and downtime), 2) storage maintenance (migration time), and 3) SSD lifetime (extra amount of data writes). Thus, in this section, we propose a more comprehensive metric, which takes all three aspects into consideration.

For the VM user experience, the existing metric, IO penalty, can indicate the performance degradation the user will experience. Note that the value of traditional IO penalty may be negative when destination disk is faster than source disk. In order to avoid this (i.e. negative penalty), we define I/O penalty as:

$$\lambda_{IO} = \frac{Best\ IO\ Performance - IO\ Performance\ during\ migration}{Best\ IO\ Performance} \quad (1)$$

In equation (1), *Best IO Performance* means the best IO performance achieved from available storage media. For example, if VM is migrated from HDD to SSD, the *Best IO Performance* is the performance we can get from SSD. Ideally, when workloads always run on a faster disk, λ_{IO} will be close to 0. From the user perspective, the less the IO penalty is, the better the storage migration scheme will be. As a "live" storage migration, the disk downtime should be close to 0. Otherwise, workloads inside the VM may crash due to intolerably long interrupts. We believe that downtime should be used as a separate metric to quantify whether a storage migration design is "live" or not.

From the perspective of the data center administrator, migration time means how long he/she should wait until the next

management can be performed. Longer migration time means higher possibility to fail the scheduled maintenance plan. Ideally, migration time should be close to the time it takes to simply copy the entire VM disk image without interference. We define the migration time factor $\lambda_{\text{migration time}}$ as:

$$\lambda_{migration\ time} = \frac{migration\ time}{image\ copy\ time} \quad (2)$$

Since storage migration will inevitably introduce IO competition and runtime overhead, *migration time* will always be longer than *image copy time*. In other words, $\lambda_{\text{migration time}}$ is always greater than 1. The smaller the $\lambda_{\text{migration time}}$ is, the better the storage migration scheme is for the data center manager.

From the device point of view, the amount of data writes can indicate the degree of wear brought by live storage migration. Besides, the larger the amount of data writes is, the more quickly the SSD will be worn out, which leads to shorter remaining lifetime for the SSD. Thus, we define a wear-out factor $\lambda_{\text{wear out}}$ to indicate the SSD lifetime penalty per time unit during storage migration. There is one caveat: HDD does not have the wear out issue. Thus, $\lambda_{\text{wear out}}$ should equal to 0 when all the write requests are issued to HDD.

$$\lambda_{wear\ out} = \frac{data\ written\ to\ SSD\ per\ time\ unit}{Workload\ Data\ Writes\ per\ time\ unit} \quad (3)$$

In equation (3), the denominator is the amount of data writes per time unit generated by the workloads while the nominator is the amount of data (generated by workload) written to SSD per time unit. For normal executing VMs, $\lambda_{\text{wear out}}$ equals to 1 if the VM is running on a SSD, 0 if the VM is running on a HDD. Any redundant writes, such as merge process in DBT and write request duplication in IO Mirroring, will make $\lambda_{\text{wear out}}$ greater. The smaller the $\lambda_{\text{wear out}}$ is, the less severely the SSD wears off during live storage migration.

Note that migration time indicates not only how long the user will suffer from the I/O penalty but also how long the device will be worn. In other words, the longer the migration time is, the higher overall penalty ($\lambda_{IO} * \lambda_{migration\ time}$) the user will observe and the higher overall degree of wearing ($\lambda_{wear\ out} * \lambda_{migration\ time}$) the SSD could receive. Therefore, we define the migration cost (MC) for VM live storage migration as:

$$\begin{aligned} MC &= \gamma * \lambda_{wear\ out} * \lambda_{migration\ time} \\ &\quad + (1-\gamma) * \lambda_{IO} * \lambda_{migration\ time} \\ &= \lambda_{migration\ time} * (\gamma * \lambda_{wear\ out} + (1-\gamma) * \lambda_{IO}) \end{aligned} \quad (4)$$

where γ defines the weight on the lifetime of SSD while $(1-\gamma)$ is the weight on the IO performance ($0 < \gamma < 1$). $\gamma > 50\%$ means wear out issue has higher priority while $\gamma < 50\%$ means IO performance is desired more.

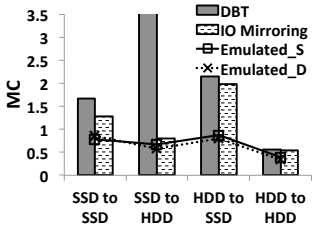


Figure 5. MC for current storage migration schemes when $\gamma = 0.5$

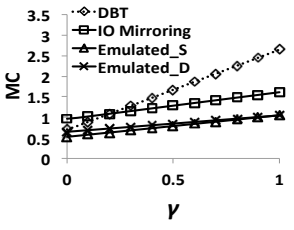


Figure 6 (a). MC when migrating a VM from SSD to SSD

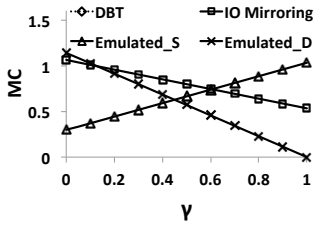


Figure 6 (b). MC when migrating a VM from SSD to HDD *(the value of DBT is too high to be shown)*

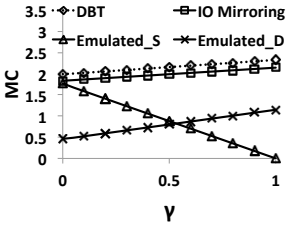


Figure 6 (c). MC when migrating a VM from HDD to SSD

Assuming neutral preference on wear-out issue and IO performance ($\gamma = 0.5$), Figure 5 shows the migration cost for existing storage migration schemes. As can be seen, no matter what type of underlying storage media are used, both DBT and IO Mirroring have extra migration cost compared to the two emulated scenarios (Emulated_S and Emulated_D). Further zooming in on each case, Figures 6 (a), (b) and (c) show how migration cost varies when the weight γ changes. In the case where both source and destination are SSDs, the larger the weight γ on $\lambda_{\text{wear out}}$ is, the higher the migration cost will be. In this situation, no matter where the VM IO requests are issued, the total writes to the SSD could not be reduced. When the source is SSD and the destination is HDD as shown in Figure 6 (b), the migration cost for DBT is way higher than that for IO Mirroring due to the long migration time (ranging from 11 to 21). Furthermore, since the source disk is SSD, running a VM on source (Emulated_S) will gain higher IO performance (lower λ_{IO}) than on the destination (Emulated_D), which leads to lower migration cost if the user cares more about the IO performance ($\gamma \to 0$). In contrast, if one prefers to extend the SSD lifetime ($\gamma \to 1$), Emulated_D will offer lower migration cost since $\lambda_{\text{wear out}}$ is 0 when all the IO requests are issued to the destination (HDD). In the case where a VM is migrated from HDD to SSD as shown in Figure 6 (c), although Emulated_D yields higher IO performance, Emulated_S will provide lower migration cost if longer SSD lifetime is preferred. Despite the value of γ, current storage migration schemes always yield higher migration cost than simply copying the running VM (Emulated_S or Emulated_D). Thus, we believe the migration cost can be reduced by adaptively balancing the three factors: $\lambda_{\text{wear out}}$, $\lambda_{\text{migration time}}$ and λ_{IO}.

5. VM Live Storage Migration Schemes under Heterogeneous Environments

5.1. Low Redundancy Live Storage Migration Mechanism (LR)

Since a VM will eventually run on the destination disk, we propose to issue all the write requests to the destination during the entire storage migration process, as shown in Figure 7. By doing so, the updated data will appear on the destination disk during the copy process and the read requests should be aware of which storage has the latest value. To implement this design, we leverage the bitmap from DBT and partition the disk image into 2 regions: copied region, whose data has already been copied to destination and to-be-copied region, whose data is going to be copied in the future. All the writes will be issued to the destination directly while the writes to the to-be-copied region also need to be recorded in the bitmap ("set" in the Figure 7). The reads to the copied area will fetch data from the destination while reads to the to-be-copied area need to check the bitmap first. If the data block is recorded in the bitmap, reads will be issued to the destination. However, if not recorded in the bitmap, data blocks will be read from the source. On the other hand, the copy process could skip the data blocks that are recorded in the bitmap. There is one caveat: requests to the data blocks, which are now being copied by copy process, will be deferred and put into a queue. They will be handled the same way as the requests to the copied area after the data blocks are released by the copy process.

The advantages of this design are: 1) there is zero redundant writes because it eliminates the merge process in DBT and duplicated writes in IO Mirroring; 2) copy process does not need to copy the entire disk image because the data blocks that have been written to the destination by VM write requests can be skipped; 3) the resource competition on disk is mitigated due to smaller volume of writes, which leads to higher IO performance and faster copy process. The benefits of this design become more evident when the destination disk (SSD) is faster than the source (HDD): when the storage migration begins, all the write requests are issued to faster disk (destination), which results in higher IO performance. In addition, the competition between copy process and VM IO requests is handled by SSD rather than HDD, which further improves the IO performance and migration time.

Note that implementing LR scheme introduces additional overhead (e.g. intercepting all IO requests, skipping data blocks in copy process, data recovery upon failure). With our implementation, the total cost of filtering IO requests and setting/checking the bitmap is less than 2us. In terms of the data recovery, our scheme forks a new process to compress and log the updates to the system hard drives (disk for OS). Upon a failure, the source disk image is recovered using the logged data. Since the logging process is parallel to the migration process and system hard drive is normally separated from data hard drives, the performance impact on our storage migration scheme is negligible.

5.2. Alternative Designs

LR scheme essentially runs the VM on the destination at the beginning of storage migration. If the destination has a slower hard drive (HDD) than the source (SSD), all the disk IO burden will be laid on the slow hard drives. Therefore, in this section, we propose two alternative designs to further exploit the IO performance from faster disk.

5.2.1. Source-based, Low Redundancy Storage Migration Mechanism (SLR)

In the situation where the destination disk is slower than the source, issuing the VM IO requests to the source can achieve better IO performance. Thus, based on LR design, we further propose source-based, low redundancy storage migration mechanism (SLR), as shown in Figure 8. We flip over the LR design by issuing as many requests as possible to the source disk. Similar to the LR design, all the IO requests are intercepted and the VM disk image is divided into 2 regions: copied and to-be-copied. The IO requests that are issued to the to-be-copied region will be issued to the source storage (faster disk). For those write requests that are issued to the copied area, SLR issues them to the destination and records them in the bitmap since doing so keeps the destination image up-to-date without invoking the merge process. All the reads falling in the copied area should first check the bitmap to find out whether the requested data has already been updated by writes or not. If so, the latest data is on the destination. Otherwise, the data will be fetched from the faster source disk (SSD). By doing so, most of the IO requests are now issued to the faster source disk.

Compared with traditional storage migration schemes, SLR has the edge on low redundant writes. Similar to LR, there is no merge process or duplicated write requests. Note that in SLR, most IO requests are issued to the source disk, which will yield better IO performance than LR. In addition, the copy process does not need to be interrupted every time to check the bitmap, which results in better migration time.

The implementation overhead of SLR is less than that of LR due to the simpler copy process. Also, the recovery process is both simpler and easier since we only need to log the write requests issued to the destination, which is significantly less than the LR. However, SLR has its limitation: not all the IO requests are issued to the (fast) source disk. Write requests and fraction of read requests to the copied region are issued to the (slow) destination. Thus, in the scenario that the workload keeps updating the data blocks in the copied area, the IO performance of SLR will become similar to running the VM on the slow disk.

5.2.2. Asynchronous IO Mirroring Storage Migration Mechanism (AIO)

In this section, we propose Asynchronous IO Mirroring (shown in Figure 9) as an alterative version of IO Mirroring design, to fully leverage the faster disk (SSD) and achieve higher IO performance.

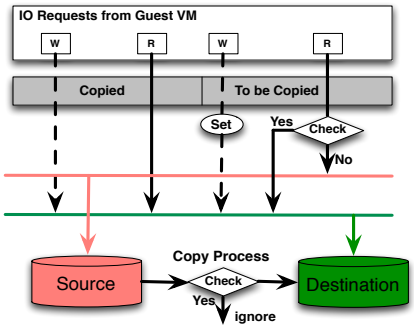


Figure 7. Low redundancy storage migration (LR)

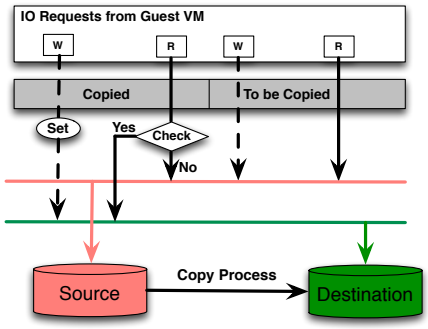


Figure 8. Source-based, low redundancy storage migration (SLR)

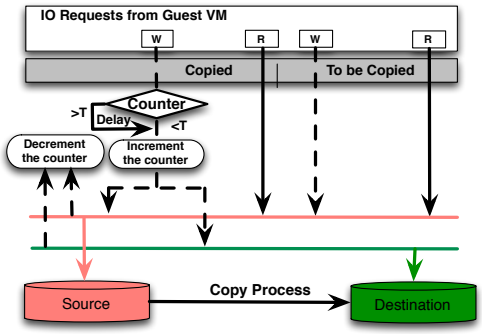


Figure 9. Asynchronous IO mirroring storage migration (AIO)

In the original IO Mirroring mechanism, the write requests are duplicated in the copied region and issued to both the source and the destination. Due to the synchronization requirement, the slower storage determines IO performance. With our AIO design, IO request is marked as completion and returned to the VM as soon as one of the duplicated IO operations is accomplished. A counter is used to track the number of unfinished IO operations. The counter will be incremented upon writes and decremented when both requests are completed. The requests to the faster disk will first check the counter to see how many requests are still pending. If the counter is larger than a threshold (T), the write request will be delayed by sleeping for a certain period. The pending write requests can be completed during that period so that the counter becomes smaller than the threshold (T). In real execution, delay will not occur frequently because the pending write requests can be completed during the CPU computation, memory access and disk reads. In the worst case (workload continuously perform writes), delay will keep occurring, which makes the overall performance as low as running on the slower disk.

The advantages of AIO are: 1) all the IO requests will be issued and handled by the faster disk (SSD) under regular IO access patterns, which yields high IO performance; and 2) Data recovery is not needed since the source disk always has up-to-date data upon failure.

Compared to LR and SLR, AIO duplicates IO requests and generates the same amount of redundant writes to the storage devices as IO Mirroring does. The disk resource competition between the copy process and IO requests becomes more intensive, which may lead to longer migration time.

5.3. An Analysis of Migration Cost of the Proposed Design and Further Extension to Massive Storage Migration

The migration cost (MC) is always expected to be as low as possible. But, which scheme should be used highly depends on the weight γ in equation (4). LR design with lowest redundant writes on device will benefit the lifetime of SSDs, which will result in lower migration cost (MC) for a user who cares more about the lifetime ($\gamma \to 1$). SLR design takes advantage of the higher IO performance of the source disk while still maintaining the low IO pressure on the device. Those who care about both the IO performance and lifetime of SSDs will tend to use SLR when migrating from SSDs to HDDs. AIO, which maximally exploits high IO performance from the faster disk, is preferred when high IO performance has top priority ($\gamma \to 0$).

During the entire life cycle of data center, the weight γ could vary dramatically. When the user demands high IO performance for his/her VM, γ will be close to 0 to guarantee low IO penalty. On the other hand, when SSD is worn out after a period, the lifetime would be the most desired, which will cause $\gamma \to 1$. Once γ is set, a certain migration method (LR, SLR or AIO) could be chosen based on the MC in equation (4).

Upon storage upgrade or disaster recovery, massive storage migration will be triggered to move all VMs on the storage. The total migration cost for massive storage migration on heterogeneous storage can be further reduced by adaptively mixing our proposed three designs according to different weights (γ). For example, if the SSD's lifetime has top priority ($\gamma \to 1$), VMs that are migrated to or from the SSD will be migrated via LR, which has the least amount of writes to the SSD. Besides, by mixing our three designs in different ways, the overall IO penalty and migration time can be optimized accordingly.

6. Prototype and Experimental Setup

We implemented DBT, IO Mirroring, and prototyped our proposed designs in *blktap2* backend driver of Xen 4.1.2 [15], which intercepts every disk IO request coming from the VM. In order to track IO requests, we implemented a bitmap filter layer in *blktap2* driver module (/xen/tools/blktap2/drivers). The data sector is dirty if the corresponding bit is set to one. We also implemented the image copy function in *blktap2* control module (/xen/tools/blktap2/control), which can be executed in a separate process alongside with the *blktap2* driver module. Data structures, such as bitmap and copy address offset, are shared between both processes. We also integrated our command line interface into *tap-ctl* Linux command so that we can trigger the storage migration using generic Linux commands. The codes for the existing and the proposed schemes are integrated into *blktap* module, which will be enabled once the migration command is issued. When all the data is copied to the destination, we pause the disk driver and modify

the file descriptor (*fd*) so that all the upcoming IO requests will be directed to the new disk image. We log the execution statistics for each scheme under the /var/log directory. We have also implemented additional design specific functions: 1) IO duplication is implemented by *memcpy*-ing the entire IO requests including address offset, size and data content. 2) Long writes are implemented by injecting constant delay into write requests. Besides, the delay is set to be 1ms and threshold is set to be 100 in our experiments. 3) IO requests to the destination disk are handled by a new IO ring structure so that the requests to the source and the destination do not compete for software resources (e.g. queue).

We evaluated the downtime, migration time, IO performance, and the migration cost by using disk or file system benchmarks: Iometer [16], Dbench [17], IOzone [18] and Linux kernel 2.6.32 compilation. Our workloads run in a VM with 1 vCPU and 2GB RAM. We also vary the input parameters of our benchmarks (e.g. outstanding IO (OIO) for Iometer, process number (proc) for Dbench, input size for IOzone) to simulate different I/O size and access patterns. Besides, kernel compilation, which is known as a comprehensive benchmark on CPU, memory and IO, is also used in our evaluation. Our virtual machines ran on self-customized servers with hardware specifications shown in Table 4. Two SSD disk arrays and Two HDD disk arrays are used as data drives while the local storage uses HDD to host Xen+OS and log files. Data Drives are connected to the server via 6Gb/s Data Link interface. Similar platform configurations were used in [10] from VMware. Our experimental VM has a 10GB system disk running Debian Squeeze 6.0 and a separate data disk (size ranging from 2GB to 30GB). To avoid the interference of OS behavior, our migration schemes and benchmarks are performed on the data disk. In order to ensure the significance of the results, we execute the workloads for at least five times. Then, we take the average of the five results and show it in this paper. Besides, we have tested our prototypes many times to guarantee its robustness.

Table 4. Hardware Specs for Our Experiment Platform

CPU	3.4GHz Intel core i7
Motherboard	ASUS Maximus V Extreme
Physical Memroy	8 GB
Hard Drives	Seagate 7200 rpm hard drives
	Avearge Data Rate:125MB/s
Solid State Drives	Intel 520 Series MLC Internal SSD
	4KB Random Reads 50,000 IOPS
	4KB Random Writes 60,000 IOPS
Disk Interconnect	6Gb/s Data Link

7. Evaluation Results and Analysis

7.1. Downtime

Table 5 shows the downtime when we migrate the VM with kernel compilation benchmark running inside. As can be seen, the downtime for DBT is worse when the destination is an HDD (rather than an SSD) because the final copy of dirty data block has to be written to the HDD. Besides, when workloads have intensive IO requests and large IO working set, the merge process of DBT cannot even converge when the destination is a HDD. IO Mirroring and our proposed designs yield stable downtime. The reason is these designs do not have a merge process and the destination has the up-to-date data when the entire image is copied.

Table 5. The downtime of live storage migration

	DBT	IO Mirroring	LR-Design	SLR-Design	AIO-Design
SSD to SSD	232ms	198ms	197ms	120ms	124ms
HDD to SSD	266ms	249ms	197ms	140ms	200ms
SSD to HDD	4000ms	298ms	200ms	150ms	201ms
HDD to HDD	900ms	199ms	246ms	179ms	260ms

7.2. Impact on SSD Lifetime

Assume the model of write requests during the storage migration as: α% of write requests are in the copied area while (1-α%) of write requests are in the to-be-copied area. β% is the percentage of writes that are performed on different block addresses (a.k.a the write working set size of workloads). Table 6 summarizes the percentage of write requests that are issued to the SSD under different scenarios:

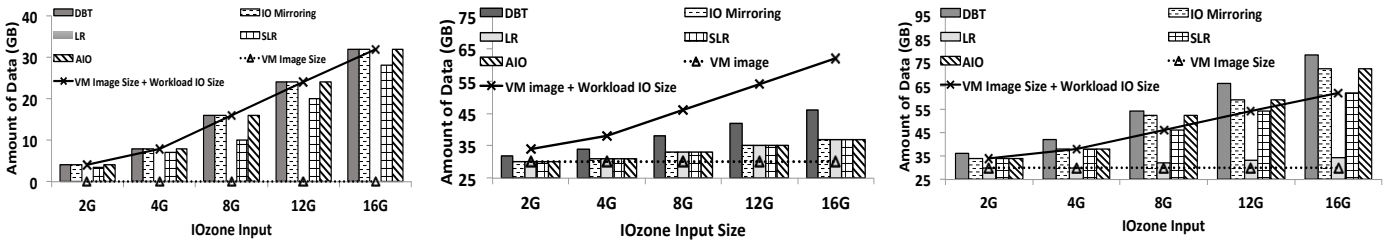
Table 6. Percentage of writes requests that are issued to SSD

	DBT	IO Mirroring	LR-Design	SLR-Design	AIO-Design
SSD to SSD	1+ β%	1+ α%	α%	1	1+ α%
HDD to SSD	β%	α%	α%	α%	α%
SSD to HDD	1	1	0	1-α%	1
HDD to HDD	0	0	0	0	0

Note that (1-α%) of the writes are skipped by the copy process for LR design. To be fair and to keep the amount of data writes in the copy process the same across all five schemes, those (1- α%) writes are counted into the copy process for LR, which means only α% of writes is issued to the destination. As shown in Table 6, LR has the lowest amount of data writes to the SSD. SLR also has smaller amount of data writes than DBT and IO Mirroring while AIO has the same amount of data writes as IO Mirroring. Thus, using LR design can mitigate the wear-out issue for SSDs.

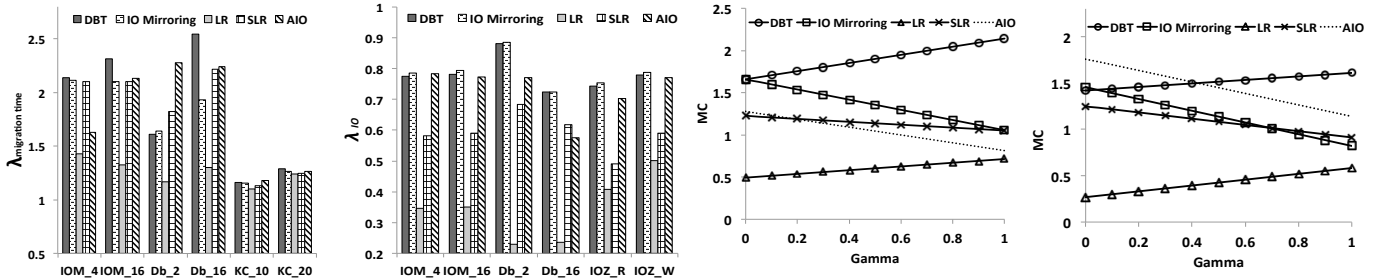
To verify this, we run IOzone with different input parameters when migrating 30GB VM image on an SSD involved storage environment. In the case of migrating a VM disk image from an SSD to an HDD as shown in Figure 10 (a), LR has zero writes to the SSD because all writes are issued to the destination. When migrating from an HDD to an SSD as shown in Figure 10 (b), DBT will have more data writes to the SSD than the other four designs because β% is larger than α% for this benchmark. When both source and destination are SSDs as shown in Figure 10 (c), LR produces the smallest amount of data writes while DBT yields the largest amount of data writes.

In general, DBT is the worst design while LR is the best in term of redundant data writes to SSDs. Since redundant data writes will affect the remaining lifetime of SSDs, LR can be awarded as SSD-preserved VM live storage migration design.



(a) From SSD to HDD (b) From HDD to SSD (c) From SSD to SSD

Figure 10. The total amount of data that is written to SSD when migrating 30 GB VM image



(a) $\lambda_{migration\ time}$ (b) λ_{IO} (c) MC for IOmeter OIO=4 (d) MC for Dbench proc=16

Figure 11. The case that VM is migrated from HDD to SSD

7.3. IO Penalty, Migration Time and Migration Cost

In this section, we compare DBT, IO Mirroring, LR, SLR and AIO in terms of migration time ($\lambda_{\text{migration time}}$), IO penalty ($\lambda_{IO}$) and Migration Cost (MC). The benchmarks with different input parameters are shown in Table 7.

Table 7. Benchmark list

Name	Benchmarks	Notes
IOM_4	IOmeter with OIO=4	IOmeter is configured as 75% reads and 25% writes
IOM_16	IOmeter with OIO=16	running in VM with 20GB disk image
Db_2	Dbench with proc=2	Dbench is configured to push the limits
Db_16	Dbench with proc =16	of throughput (-R 99999) running in VM with 20GB
IOZ_R	IOzone read file test	IOzone is used to test the IO Performance
IOZ_W	IOzone write file test	during the migration
KC_10	Kernel Compilation in 10GB image	Kernel Compilation is used to test our schemes
KC_20	Kernel Compilation in 20GB image	in comprehensive, real workloads

7.3.1. Migrating VMs from HDDs to SSDs

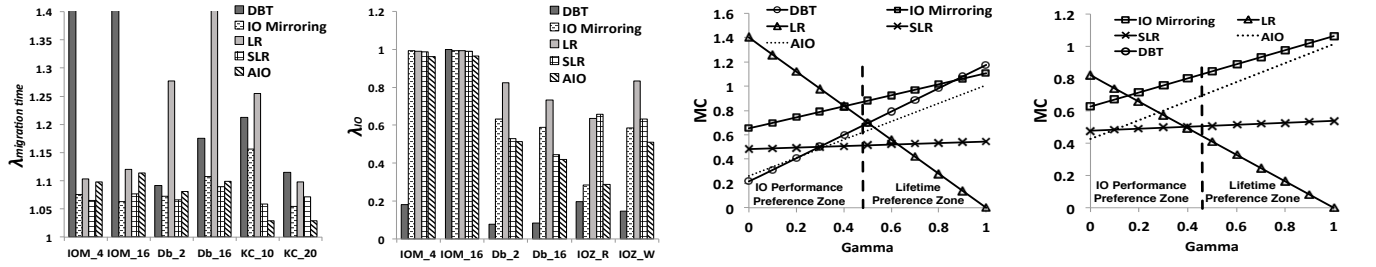
As shown in Figures 11 (a) and (b), LR exhibits advantages for both migration time (56% shorter than traditional design) and IO performance (at least 35% less IO penalty compared with traditional design). Issuing IO requests to the destination will not only benefits the IO performance but also decreases the migration time by skipping updated data blocks and mitigating IO contention. Although SLR and AIO are not designed specifically for this case, they both have better migration time and IO performance than the traditional design. By further considering the amount of redundant data writes ($\lambda_{\text{wear out}}$), we show the migration cost for two benchmarks in Figures 11 (c) and (d). As can be seen, LR has the lowest migration cost no matter what the weight γ is. Although LR does not exhibit advantages in terms of SSD data writes in this situation, it leverages the fast access speed from SSDs to the greatest extent possible, which not only mitigates the IO penalty but also decreases the migration time.

By further digging into results shown in Figures 11(c) and (d), we observe that as the weight gets close to 1, the migration cost for DBT and LR are increased because $\lambda_{\text{wear out}}$ is larger than λ_{IO} for these two designs. On the other hand, the migration cost for IO Mirroring, SLR and AIO are reduced when γ ->1. That is because those three designs have smaller λ_{IO} than $\lambda_{\text{wear out}}$. In this situation, the benefits of having high IO performance and shorter migration time overwhelm the effect of $\lambda_{\text{wear out}}$ when using LR. Thus, LR always yields the least migration cost when a VM is migrated from an HDD to an SSD.

7.3.2. Migrating VMs from SSDs to HDDs

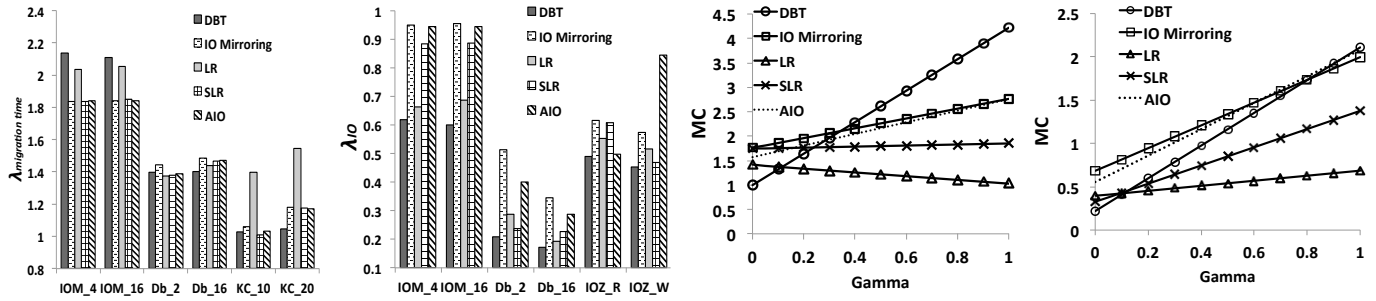
By analyzing Figures 12 (a) and (b), we found that in this situation, SLR has the shortest migration time on average while AIO has the least IO penalty. Since SLR uses the source (SSD) to handle IO requests and generates less redundant writes, the copy process can consume more disk bandwidth, which results in shorter migration time. On the other hand, AIO takes advantage of faster disk all the time, which would lead to lower IO penalty. Note that DBT is excluded from the comparison due to its frequent failure (merge process takes extremely long time). As discussed in section 7.2, LR has zero writes to the SSD in this situation, which means it has an edge in preserving SSD's lifetime.

Figures 12 (c) and (d) show migration cost as weight increases from 0 to 1. The area where γ is larger than 50% is defined as Lifetime Preference Zone, which includes users who consider SSD wear-out issue as top priority; the area where γ is less than 50% is defined as IO Performance Preference Zone, which includes those who prefer to have high IO performance. As can be seen in Figures



(a) $\lambda_{migration\ time}$ (b) λ_{IO} (c) MC for IOmeter OIO=4 (d) MC for Dbench proc=16

Figure 12. The case that VM is migrated from SSD to HDD



(a) $\lambda_{migration\ time}$ (b) λ_{IO} (c) MC for IOmeter OIO=4 (d) MC for Dbench proc=16

Figure 13. The case that VM is migrated from SSD to SSD

12 (c) and (d), LR has relatively low migration cost in Lifetime Preference Zone due to zero writes to SSDs. Besides, the more the γ is close to 1, the less the migration cost will be for LR. In IO Performance Preference Zone, AIO is more desired as γ becomes smaller. As SLR maintains the balance between migration time, IO penalty and the amount of SSD data writes, it has the lowest migration cost when γ is around 0.5.

Further investigating in the trend of migration cost, we found that the slope for LR is negative, which means that the migration cost for LR keeps decreasing as the weight moves towards 1. However, the migration cost will increase for the other four designs as γ increases. Thus, when migrating from SSDs to HDDs, the weight determines which design has the lowest migration cost. If IO performance is preferred, AIO is the most suitable decision; if SSD wear-out issue is the main concern, LR is the best policy; if the IO performance and SSD lifetime are equally important, SLR will yield the lowest migration cost.

7.3.3. Migrating VMs from SSDs to SSDs

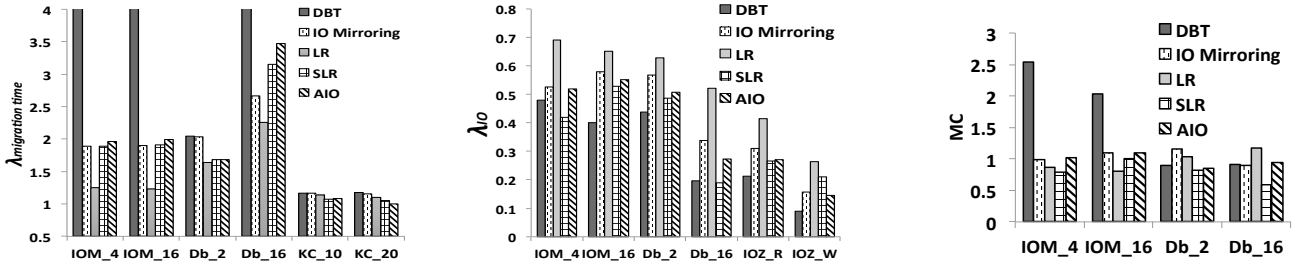
When a VM is migrated from SSDs to SSDs, issuing requests to source or destination does not affect the migration time and IO performance significantly. In terms of migration time as shown in Figure 13 (a), on average, SLR and AIO take slightly shorter time. LR has slightly longer migration time because of the overhead to check bitmap in the copy process. In terms of IO penalty, DBT has the least IO penalty; LR ranks second as shown in Figure 13(b). However, DBT generates the largest amount of data writes to the SSD while LR has the least amount of data writes to the SSD as discussed in Section 7.2. When considering migration cost, which is shown in Figures 13 (c) and (d), LR yields the least cost except when the weight (γ) is 0. Besides, the migration cost for DBT will increase faster than others as the weight increases because it has the largest slope rate. SLR, although yielding slightly more migration cost when γ =0, has a relatively low migration cost than DBT when γ lies between 0.1 and 1. In total, we believe that SLR will be the best choice when IO performance is desired. For the majority of cases (0.2<γ<1), LR is the best migration policy, which shows the lowest migration cost.

7.3.4. Migrating VMs from HDDs to HDDs

In the situation that no SSD is involved, the wear-out factor $\lambda_{\text{wear out}}$ will not affect migration cost at all. As shown in Figure 14 (a), LR takes the shortest time to migrate a VM while DBT takes the longest time and occasionally even fails to complete the migration. Besides, the downtime for DBT is longer than others as discussed in Section 7.1, which makes us exclude DBT from comparison. In terms of IO penalty, which is shown in Figure 14 (b), SLR is the best; AIO ranks second. There is a tradeoff between migration time and IO performance in this situation: lower IO penalty (λ_{IO}) can only be achieved by sacrificing the migration time $\lambda_{\text{Migration time}}$. Taking both factors into consideration, Figure 14(c) shows the migration cost (MC) for all the five designs. On average, SLR has the lowest migration cost; LR ranks the second.

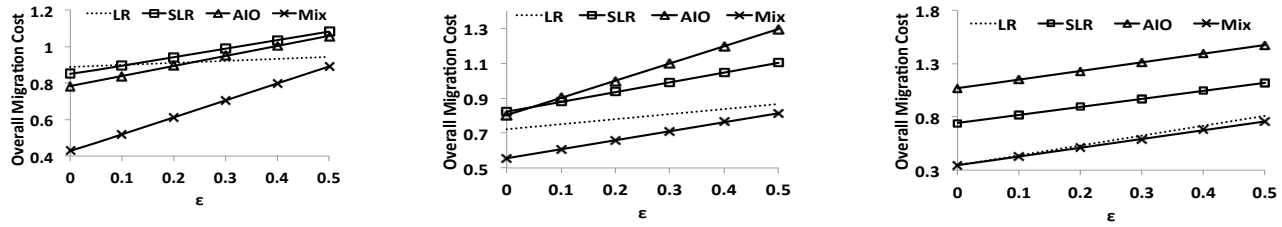
7.4. Towards Lower Migration Cost in Massive Storage Migration by Mixing Different Designs

As mentioned before, each individual design has its own strength. In this section, we show an example to demonstrate how to further reduce the overall migration cost in massive storage migration by



(a) $\lambda_{migration\ time}$ (b) λ_{IO} (c) MC

Figure 14. The case that VM is migrated from HDD to HDD



(1) γ= 10% (2) γ=50% (3) γ=80%

Figure 15. Overall MC of the massive storage migration on heterogeneous storage environment

mixing all three designs. Consider that we want to move all the VMs from one storage pool to another for the purpose of storage maintenance. Assume the percentage of VMs that will be migrated from SSDs to SSDs is the same as the percentage of VMs that will be migrated from HDDs to HDDs (ε). And the number of VMs that are migrated from SSDs to HDDs is the same as the number of VMs that are migrated from HDDs to SSDs, which will be (1-ε-ε)/2=0.5-ε. The migration cost of the massive storage migration will be the sum of cost for each individual storage migration. To simplify the discussion, we consider weight γ as 10%, 50% and 80%.

Figures 15 (1), (2) and (3) show the migration cost for the massive storage migration when we deploy each single storage migration design (LR, SLR and AIO) and mix them together (Mix). As can be seen, under different weights γ and percentage of migrating between heterogeneous storage media (0.5-ε), mixing all the three designs (Mix) can always achieve lower migration cost. This is because Mix always chooses the migration scheme with the lowest migration cost for each single storage migration behavior. For instance, when the SSD lifetime is top priority (γ >50%), most of the VMs will be migrated using LR, which makes the migration cost of Mix close to that of LR (1% lower than LR but 69% lower than AIO). In the case that the IO performance is desired and most of the migration occurs on the same storage media (ε->0), Mix will be relatively close to AIO but still has the lowest migration cost (48% lower than AIO).

8. Related Work

To eliminate the merge process in DBT and achieve stable and low downtime and migration time across benchmarks, [10] proposed IO Mirroring storage migration mechanism, which aims to maintain data consistency between the new and old images during migration by duplicating IO requests in the copied region. Our proposed techniques differ from IO Mirroring in that they fully exploit the performance characteristics of SSDs while taking SSD lifetime into consideration. [11, 14] implemented live VM migration in wide area network (WAN) equipped with persistent storage. The block-level disk pre-copy mechanism is employed to transfer the VM disk image. The write IO requests are throttled to reduce the dirty block rate and IP tunneling is used to make the network switching transparent to guest OS. [12] proposed to couple VM memory copy with disk copy to hide the downtime of VM live migration into that of storage migration and [13] takes workload behavior into consideration when performing storage migration. Our proposed techniques are orthogonal to these schemes and can be combined together to further improve the efficiency of virtual machine storage migration in light of heterogeneous storage media. [23] leverages the copy-on-write features in virtual storage and puts the read-only templates of VM disk images on SSDs. However, the storage migration behavior and the wear-out issue are not considered in their work. [24] focuses on file system level implementation of hybrid SSD storage systems, which intends to improve the IO performance via faster SSDs. However, their work does not consider the migration, wear-out issue, and virtual disk image. [25] implements a virtualized flash storage layer for Fusion-io device, which will yield shorter IO response. However, both wear-out issue and migration behavior are not considered.

9. Conclusion

As SSDs emerge as an indispensable media for cloud storage, their strength and weakness should be taken into account in storage management, such as VM storage migration. Nevertheless, existing mainstream storage migration schemes (e.g. DBT and IO Mirroring) do not fully exploit the features of SSDs. Even worse, they wear out SSD devices severely. In this paper, we propose a new metric, migration cost, which characterizes the cost for storage migration mechanisms from multiple aspects: migration time, IO penalty, SSD lifetime and preference (on IO performance and SSD lifetime). We propose three new storage migration mechanisms to achieve lower migration cost. Each of the proposed schemes has its own strength: 1) LR has the lowest possible redundant writes; 2) SLR leverages IO performance of the source disk while still maintaining the attribute of low redundancy and 3) AIO aims at achieving the highest possible IO performance. Our prototype-based evaluation shows that all three designs yield stable downtime (around 200ms) and lower migration cost than traditional mechanisms (DBT and IO Mirroring). By mixing them in massive storage migration, the overall migration cost can be further reduced by 48% at most compared with the best of each individual mechanism.

Acknowledgments

This work is supported in part by NSF grants 1117261, 0937869, 0916384, 0845721(CAREER), 0834288, 0811611, 0720476, by SRC grants 2008-HJ-1798, 2007-RJ-1651G, by Microsoft Research Trustworthy Computing, Safe and Scalable Multi-core Computing Awards, by NASA/Florida Space Grant Consortium FSREGP Award 16296041-Y4, and by three IBM Faculty Awards. Fang Liu is supported by the National High-Tech Research and Development Program of China (No. 2013AA013201), the National Natural Science Foundation of China (NO.61170288, NO.61025009, NO.61232003).

References

[1] "Amazon EC2", http://aws.amazon.com/ec2/
[2] "Microsoft Azure", http://www.windowsazure.com/en-us/
[3] "Flash Drives Replace Disks at Amazon, Facebook, Dropbox", http://www.wired.com/wiredenterprise/2012/06/flash-data-centers/
[4] "Morphlabs, Dell DCS Team on SSD-Powered Cloud", http://www.datacenterknowledge.com/archives/2012/03/28/morphlabs-dell-dcs-team-on-ssd-powered-cloud/
[5] "SolidFire Develops All-SSD System for Cloud Storage Providers", http://searchstoragechannel.techtarget.com/news/2240037093/SolidFire-develops-all-SSD-system-for-cloud-storage-providers
[6] "Intel Takes Their SSD Reliability to the Datacenter", http://www.zdnet.com/blog/datacenter/intel-takes-their-ssd-reliability-to-the-datacenter/1316
[7] Gokul Soundararajan, Vijayan Prabhakaran, et al., Extending SSD Lifetimes with Disk-Based Write Caches, FAST 2010
[8] EMC, http://www.us.emc.com/index.htm
[9] Winchester Systems, http://www.winsys.com
[10] Ali Mashtizadeh, et al., The Design and Evolution of Live Storage Migration in VMware ESX, ATC 2011
[11] Robert Bradford, Evangelos Kotsovinos, Anja Feldmann, Harald Schioberg, Live Wide-Area Migration of Virtual Machines Including Local Persistent State, VEE 2007
[12] Yingwei Luo, Binbin Zhang, et al., Live and Incremental Whole-System Migration of Virtual Machines Using Block-Bitmap, ICCC 2008
[13] Jie Zheng, T. S. Eugene Ng, Kunwadee Sripanidkulchai, Workload-Aware Live Storage Migration for Clouds, VEE 2011
[14] Takahiro Hirofuchi, et al., A Live Storage Migration Mechanism over WAN for Relocatable Virtual Machine Services on Clouds, CCGRID 2009
[15] XEN Project http://www.xen.org, January 2009
[16] Iometer Project, http://www.iometer.org
[17] Dbench, http://dbench.samba.org
[18] IOzone File System Benchmark, http://www.iozone.org
[19] Aameek Singh, Madhukar Korupolu, et al., Server-Storage Virtualization: Integration and Load Balancing in Data Centers, SC 2008
[20] Seonyeong Park, A Comprehensive Study of Energy Efficiency and Performance of Flash-based SSD, Journal of Systems Architecture, 2011
[21] Guanying Wu and Xubin He, ΔFTL: Improving SSD Lifetime via Exploiting Content Locality, EuroSys 2012
[22]Youngjae Kim, et al., HybridStore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs, MASCOTS 2011
[23] Heeseung Jo, Youngjin Kwon, Hwanju Kim, Euiseong Seo, Joonwon Lee, and Seungryoul Maeng, SSD-HDD-Hybrid Virtual Disk in Consolidated Environments, VHPC 2009
[24] Feng Chen, David Koufaty, Xiaodong Zhang, Hystor: Making the Best Use of Solid State Drives in High Performance Storage Systems, ICS 2011
[25] William K. Josephson, et al., DFS: A File System for Virtualized Flash Storage, ACM Transactions on Storage, Sept 2010
[26] Bob Laliberte, Delivering Greater Effectiveness and Efficiency for SANs in Virtualized Data Centers. White Paper, EMC
[27] "Intel Launches DC S3700 SSD for Data Centers", http://hothardware.com/News/Intel-Launches-New-Datacenter-SSDs-Emphasizes-Data-Protection-High-Performance/