THE DESIGN AND IMPLEMENTATION OF CLOUD-SCALE LIVE MIGRATION

BY LONGHAO SHU

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

Written under the direction of

Dr. Thu D. Nguyen

and approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2014

ABSTRACT OF THE THESIS

THE DESIGN AND IMPLEMENTATION OF CLOUD-SCLAE LIVE MIGRATION

by LONGHAO SHU

Thesis Director: Dr. Thu D. Nguyen

Live migration, short for live virtual machine (VM) migration, enables a running virtual machine to move between two physical hosts without perceptible interruption in service. Live migration is an efficient tool for system administrators to perform system maintenance, load balancing, and fault management while allowing end-users to avoid costly service downtimes. Today, live migration between hosts connected by a local area network (LAN) has become a critical feature of enterprise class virtual infrastructure.

We also expect live wide area network (WAN) migration, e.g., Cloud-scale live migration, to extend the scope of provisioning compute resources from a single data center to multiple geographically disparate data centers. Currently, Cloud-scale live migration is possible only through ad-hoc solutions using network file systems, proprietary storage array replication or software replicated block devices used in concert with more well known approaches for migrating memory. But this loose aggregation of mechanisms makes migration architectures complex, inflexible, and unreliable and performs poorly compared with live LAN migration

in general.

To overcome those deficiencies, we present a Cloud-scale live migration framework that integrates support for memory and storage migration over WAN and maintains much of the simplicity and reliability of live LAN migration. The main challenge for implementing Cloud-scale live migration is how to deal with the large VM data (especially VM storage data) transferring over WAN. To solve this problem, we propose a new "Migration over FedEx" solution to combine the benefits of both live LAN migration and transferring large amount of data via shipping portable storage devices containing the data. Our solution capped the total migration time into a bounded time period without increasing the downtime compared with traditional live migration. In the meanwhile, the total migration cost is greatly reduced especially for migrating large number of VMs.

Acknowledgements

First and foremost I would like to express my thanks to my internship mentor Min Cai, for his insightful advice and support, for providing me an excellent research environment at VMWare Inc.

I also would like to express my gratitude to my commitee members Dr. Thu Nguyen, Dr. Ricardo Bianchini and Dr. Abhishek Bhattacharjee for reviewing my work and their feedbacks.

Last but not least, I would like to express my eternal gratitude to my parents and girlfriend, Yaqin, for their everlasting love and support.

Dedication

To my parents

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Operating system virtualization is a powerful technique allowing multiple OS instances to run concurrently on a single physical machine with high performance [2]. The resulting resource isolation not only provides secure execution environment for each guest OS but also make live migration become reality [4][23]. Live migration is an alternative solution for process migration [20] but not restricted to that. Migrating the VM with its applications as a whole unit simplifies the process-level migration approach. Moreover, live migration allows the maintenance of the original host without perceptible downtime of the service. By load balancing VMs, we can efficiently use the computing resources and reduce the energy consumption. In addition to that, migrating whole memory state can avoid dealing with consistentcy problems, e.g. TCP connection etc., caused by partially memory migration [6]. Overall, by separating the software and hardware considerations, live migration becomes an extremely effective tool for datacenter administrators to improve manageability.

However, those benefits for live migration are only limited to migration between hosts within the same LAN. Most of the mainstream virtual machine hypervisors usually do not support live migration over WAN. Some of them support live migration between different datacenters which shared storage, it avoids transferring the persistent state of VMs over WAN. There are some state of the art systems that do allow live WAN migration [3][9][11][19][31], but it is very expensive to migrate large amounts of persistent data over WAN connections, especially when migrating large number of VMs rather than a single VM instance. Some

previous research showed that VMs running identical operating systems revealed content similarity for their memory [7][21][30][32] and storage [17][24][25]. Some techniques [26][29] which leverage this characteristic have been developed to reduce network traffic when migrating VMs between data centers interconnected by WAN.

In this dissertation, we propose "Migration over FedEx", a Cloud-scale live migration framework leveraging the benefits of both live LAN migration and transferring large amounts of data via shipping portable storage devices containing those data by courier service. With this framework, we only need to migrate the memory, device state, and dirty storage over WAN, which greatly reduced the total migration time and monetary cost. We also design a LAN Storage Replication Protocol (LSRP) to export/import the VM's base storage. The LSRP protocol also contains a lighweight delta consolication protocol which can consolidate the dirty storage data into the existing base disk with low cost. Compared with traditional live WAN migration schemes, our approach caps the total migration time into a bounded time period and greatly reduces the total migration cost, especially when migrating large number of VMs.

This dissertation is organized as follows. Chapter 2 describes the background of live migration and the metrics used to evaluate a live migration scheme. Chapter 3 first investigates the necessity for using courier service to migrate VM's base storage, then introduces the Migration over FedEx framework as well as its workflow steps. Chapter 4 elaborates the LAN Storage Replication Protocol specification and how those migration components coordinate with each other. Chapter 5 evaluates our approach. Chapter 6 concludes the thesis and discusses the possible future work.

Chapter 2
Background

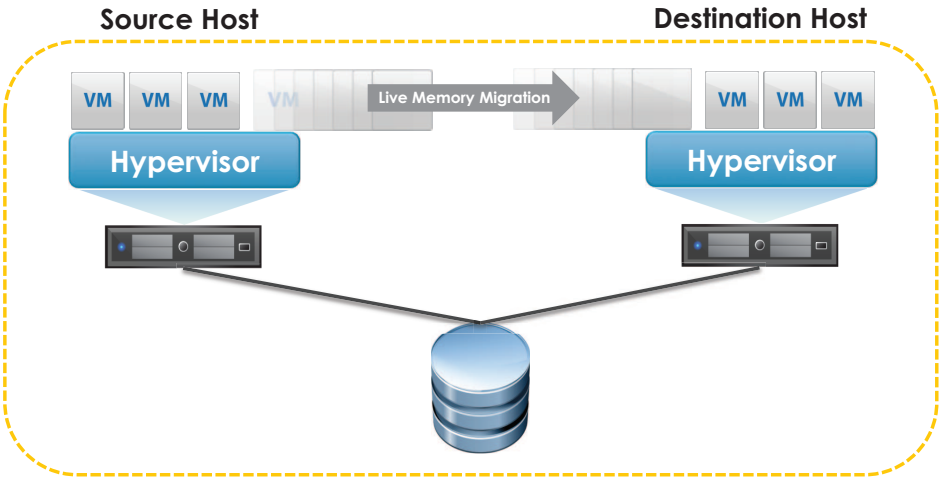
This chapter provides background for live migration. We first talk about the benefits obtained by using live migration as well as its classification. Then we introduce the metrics for evaluating a live migration scheme and the technique evolution for live memory migration and live storage migration. Finally, we discussed the necessity of live WAN migration. Later in this chapter, we briefly introduce our approach.

2.1 Live Migration

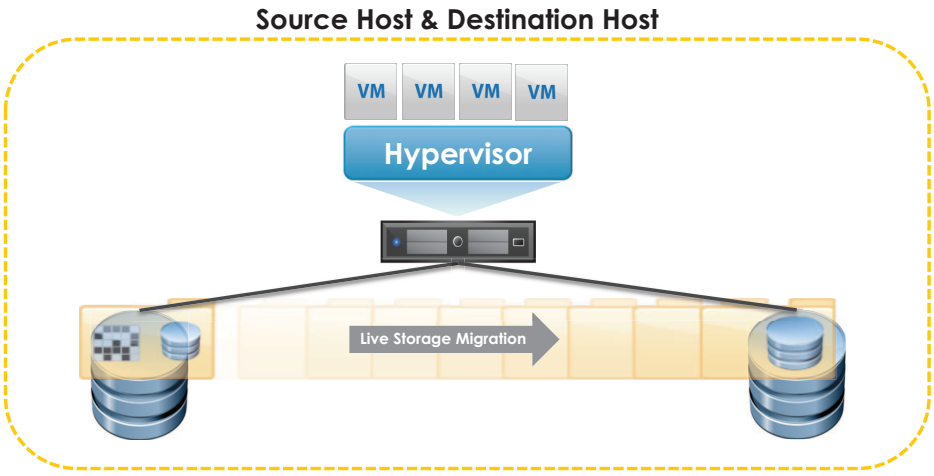
Live migration is a process in which a running virtual machine instance (memory, storage, and other device state) is entirely moved from one physical machine to another without interrupting the execution of software running within the virtual machine. The most significant advantages of live migration are the facts that it facilitates system maintenance, load balancing, and fault management. For example, system administrators can move all virtual machines on a host to another one when maintenance for the original host is needed. By load balancing VMs among hosts, we can optimize the utilization of available compute and storage resources. Also, we can recover from system failure by switching the execution to the migrated instance of the failed one. With these important applications, live migration has become the critical feature of enterprise class virtual infrastructure.

According to different storage and network connection characteristics, live migration can be classified into different catagories, i.e., live memory migration

and live storage migration (Figure 2.1), live unified migration over LAN and live unified migration over WAN (Figure 2.2). Live memory migration moves a virtual machine between two hosts which share storage, thus there is no need for migrating VM storage. Due to the constraints of high latency and limited bandwidth of WAN link, migrating VMs over WAN is much more challenging than migrating over LAN.

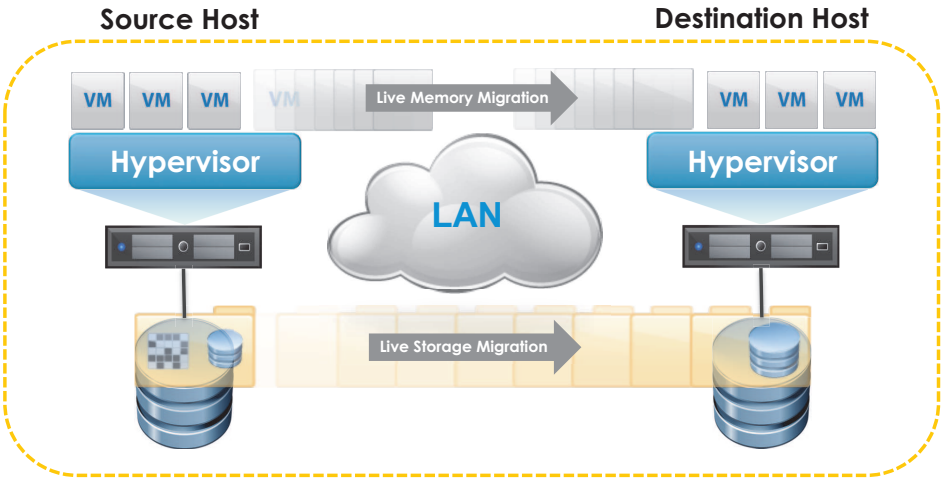


(a) Live Memory Migration

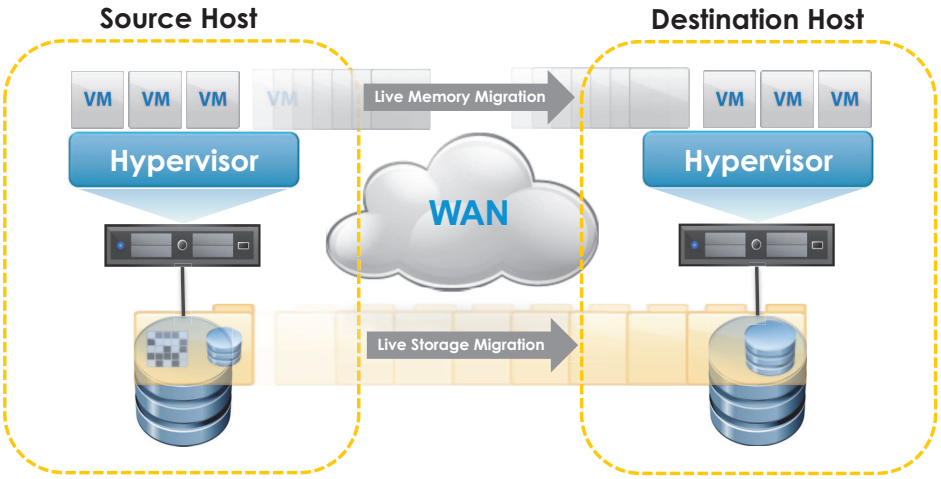


(b) Live Storage Migration

Figure 2.1: Live migration classification according to existence of shared storage



(a) Live Unified Migration over LAN



(b) Live Unified Migration over WAN

Figure 2.2: Live migration classification according to network connection type

2.1.1 Metrics

The efficiency of a live migration scheme can be evaluated using three different metrics: **downtime**, **total migration time**, and **total transferred bytes**. Downtime is the time period during which the service is interrupted because of the suspension of the original VM. Total migration time is the time duration taken by moving a entire VM instance from the source host to the destination host. Total transferred bytes is the total number of bytes transferred over the network from the source host to the destination host. A good live migration scheme is always

seeking a relatively reasonable trade-off among the requirements of minimizing the downtime, the total migration time, and the total transferred bytes.

In addition, we should also consider the impact on other active services caused by any live migiration scheme as well. Taking live unified migration over LAN as an example, the memory and storage both are transferred in a succession of iterations. The resulting large traffic could easily consume the remaining bandwidth available between the source and destination and hence starving other active services. Therefore, some degree of **service degradation** will inevitably occur during any live migration process. However, this negative impact can be alleviated by throtting the data transferring rate by limiting the network and CPU resources used by the migration process.

2.1.2 Live Memory Migration

There are a number of ways to move the VM's memory state from one physical host to another, and those approaches differ from each other by transferring the memory state in different orders. The most widely used three techniques are: **Pre-Copy Memory Migration** [28], **Post-Copy Memory Migration** [12][13], and **Hybrid-Copy Memory Migration**.

Pre-copy based live memory migration is easy to design and does not require a fast network connection between hosts. Thus it is implemented in most mainstream hypervisors such as Xen [2], KVM [15], and VMware ESX [5]. The memory is transferred to the destination in a manner of successive iterations until the remaining amount of dirty memory is lower than a pre-defined threshold value.

Pre-copy memory migration includes two phases: **Warm-Up Phase** and **Stop-and-Copy Phase**.

- **Warm-Up Phase:** Starting the first iteration by copying the whole memory from source to destination while the VM is still running on the source. Any dirty memory generated in this process will be re-copied in the second iteration. This process will be repeated until the amount of dirty memory is lower than a pre-defined threshold. Note that the convergence of this iterative copying process depends on copying rate being faster than the rate of memory writes in the virtual machine.

- **Stop-and-Copy Phase:** Once the Warm-Up Phase stops, the VM on the source host will be suspened and the remaining dirty memory will be copied to the destination then the VM will be resumed on the destination host.

Post-copy based live memory migration starts by first suspending the source VM and copying a minimal subset of execution state to the destination host. Upon finishing, the execution control is immediately transferred and the VM is resumed at the destination side. A page-fault will be triggered if the VM tries to access any un-transferred data, which will be trapped at the destination and redirected towards the source over the network. At the same time, a thread will copy over the remaining memory in the background while the VM is running on the destination host.

Hybrid-copy based live memory migration is a combination of the other two approaches for seeking a trade-off between minimizing downtime and total migration time. It starts by executing some copying iterations of pre-copy warm-up phases before entering the post-copy phase. The initial warm-up iterations can reduce the possibility of page fault on the destination host since a large portion of the VM memory state has already been pre-copied.

2.1.3 Live Storage Migration

There are some very important use cases for live storage migration. For example, live storage migration enables storage maintenance by which the system administrator can upgrade storage arrays and file system. By storage load balancing, we can improve the storage performance and prolong the lifetime of storage devices. Earlier live migration did not move VM's storage, requiring that all virtual disks should reside in the same storage array accessible by both the source and destination hosts. Various software and hardware solutions have been developed to overcome this limitation. Among those software solutions, there are three approaches which are the most popular: Snapshotting, Dirty Block Tracking and Distributed Replicated Block Device (DRBD).

- **Snapshotting:** The migration begins by taking a snapshot of the base disk and copying the base disk to the destination. All writes from guest OS will be sent to this snapshot file. After the copying process is finished, another snapshot is taken and the old shapshot is consolidated into the base disk. By the time the consolidation is complete, some writes may have been sent to the new shapshot. Thus such process will be repeated until the amount of data in the snapshot file becomes small enough.

- **Dirty Block Tracking (DBT):** The migration begins by copying the base disk to the destination. In the meanwhile, a bitmap is used to track the dirty blocks on the source host. At the end of the first copy iteration, the bitmap is atomically obtained and cleared and the dirty blocks identified by the bitmap are copied to the destination. Such process is repeated until the number of dirty blocks stabilizes, then the source VM is suspended and the remaining dirty memory is transferred.

- **DRBD:** The migration begins by copying the base disk to the destination block by block concurrent with mirroring all writes from the guest OS on

the source host to the destination. The source and destination will become consistent immediately after the base disk is entirely copied to destination. Any writes to the block being copied will be synchronized.

	Advantages	**Disadvantages**
Snapshotting	simple to implement good robustness easy to converge	not atomic cost space lower performance
DBT	finer granularity atomic	harder to converge
DRBD	atomic easy to converge	harder to implement

Table 2.1: Comparisons of three typical live storage migration approaches

Those live storage migration approaches differ from each other by way of functionality, implementation complexity and performance. We compare those approaches in Table 2.1. Snapshooting is simple to implement and easy to tune. But it is not atomic, which may result in an intermediate state where multiple snapshots span over both source and destination storage volumes when cancelling a migration. DBT overcomes most of the performance inadequacies of snapshotting while increasing the complexity of tuning. Specifically, it is hard to get convergence when the write rate at the source is higher than the dirty rate. DRBD overcomes all of those deficiencies while maintaining better temporal and spatial performance compared with the other two approaches.

2.1.4 Live WAN Migration

Live WAN migration is key for providing a hybrid Cloud solution which can aggregate the resource capability from both private and public Cloud. This solution not only gives users the same experience of running workloads in private Cloud but also offers the benefits of public Cloud such as economies of scale and improved flexibility. Some use cases illustrate the necessity of live WAN migration, e.g. Cloud-scale live migration.

- **Cloud Bursting:** Cloud Bursting is an application deployment model in which an application running in a private Cloud bursts into a public Cloud when the demand for local computing resouces spikes. It allows an enterprise to dynamically harness Cloud servers and migrate the workload to those servers when the local workload transcends the confines of its private datacenter.

- **Data Center Consolidation:** Data Center Consolidation is an organization's strategy to reduce overall operating costs and IT footprint by shrinking the size of a single data center or merge several small data centers into a large one.

- **Follow The Renewable:** Follow The Renewable is conceptually similar with Follow The Sun but more coarse-grained, which is a type of global workflow model in which tasks are passed around seasonally between work sites that are many time zones apart. With Cloud-scale live migration, people can easily and freely move the tasks among different geographic locations by migrating the whole virtual machines.

- **Disaster Recovery as a Service:** DRaaS is a predetermined set of processes offered by a third-party vendor to help an enterprise develop and implement a disaster recovery plan. The idea is that when a disaster is expected you can start to move the virtual machines to the backup data center while they are running.

2.2 Related work

Pre-copy is a widely-used technique for migrating memory no matter for the traditional live LAN migration or state-of-art live WAN migration approaches. At the end of pre-copy process, the source VM will be finally suspended for

copying the remaining dirty pages. This phase is directly responsible for the downtime experienced during the whole migration process. Downtime can range from a few milliseconds to seconds or minutes, depending on page dirtying rate, network bandwidth and latency [1]. Minimizing downtime is the first priority for any live migration scheme.

The previous work for live memory migration can be classified into two types: pre-copy based and post-copy based approaches. For pre-copy based approaches, the hybrid-copy could be seen as an optimization of it. The existing work optimizes and improves the naive pre-copy by using techniques like compression, delta consolidation, and content deduplication. The performance of compression-based approaches really depends on the compression algorithm and the memory page characteristics [14]. Delta conslidation based approaches reduce the network traffic by sending and conslidating the different memory content between the source and destination memory [8][27][31]. Content deduplication based approaches avoid transferring the duplicated data which is cached on both source and destination side by sending an index of the corresponding data to the destination [33][31]. For post-copy based approaches, [16] fast deploy VMs on multiple hosts by using demand-paging and multicast distribution of data. Some approaches adopt checkpointing/recovery and trace/replay technique to improve the migration efficiency [18].

The previous work for live storage migration can also be classified into pre-copy based [3], post-copy based [10][11], and hybrid-copy based [19] approaches. In pre-copy scheme the storage is migrated prior to memory whereas the sequence is reversed in post-copy scheme. Pre-copy scheme causes no data access latency since each storage block has been copied over before the VM runs in the destination, but it may introduce excessive extra traffic. Post-copy scheme dose in a completely opposite way, the VM will start to run before the storage blocks are copied. The disk blocks can be copied either by a background process or on-demand. Unlike

pre-copy scheme, the post-copy scheme will cause extra WAN delays since the data can not be immediately obtained when a I/O request is issued by destination VM.

2.3 Our approach

We propose a live WAN migration framework and implement it as a Cloud-scale live migration solution, which has scaled out the geographical scope and network environment of traditional live migration by enabling VM mobility over WAN with lower cost. There are two big challenges in implementing Cloud-scale live migration. The first one is the contradiction between large WAN traffic and limited bandwidth. The second one is the security issue of a open migration protocol in a multi-tenant network environment. We solve the first problem by proposing a new "Migration over FedEx" solution to combine the benefits of both live LAN migration and transferring large amounts of data via shipping portable storage devices containing the storage data. We leave the second problem, migration protocol hardening, as our future work.

Chapter 3
Framework: Migration over FedEx

In this chapter, we present the overall design of our Migration over FedEx framework as well as the workflow steps. By comparing the bandwidth, unit transfer cost and efficiency of each mainstream WAN link type, we affirm the necessity of leveraging the capability of courier service (like FedEx first overnight etc.) to migrate large amounts of data among geographically disparate locations in a relatively short time period. Finally, we elaborate on each migration component of Migration over FedEx framework.

3.1 Overview

Cloud-scale live migration is a natural extension of the existing live migration in many virtual machine hypervisors (like vMotion in VMware ESX) from the perspectives of geographical scope and network connection type. Offering virtual machine mobility in hybrid Cloud environement is necessary for providing a complete hybrid Cloud solution:

- It enables enterprise users to ramp up quickly onto the public Cloud from their existing on-premise datacenters. A key advantage of Cloud-scale live migration is that existing enterprise workloads can be run seamlessly on public Clouds without any modification or redeployment.

- It allows ordinary users to seamlessly move workloads to public Cloud with minimal disruption. With Cloud-scale live migration, the public Cloud can

be a seamless extension of any private compute resources.

3.1.1 FedEx?!

For geographically separated sites, it is very time-consuming and monetarily expensive to transfer large numbers of virtual machines over WAN connections because of the limited bandwidth and high latency. Instead, it is much faster and more cost effective to export/import the virtual machine data to/from portable storage devices and shipping them using through FedEx or other courier services. Moreover, with the rapid development of hard disk drive (HDD) technology, the capacity per HDD has increased to the order of tens or even hundreds of terabytes while the price decreased to nearly 60 US dollars per terabyte. The high storage density and low price of modern HDD is another incentive for Migration over FedEx.

Many courier services provide national-wide over-night delivery service. Taking FedEx First Overnight as an example, it costs less than $200 to ship 50TB of hard drives from San Francisco to New York. The equivalent Internet bandwidth needed to move the same amount of data in 15 hours is more than 7Gbps which is about three times the bandwidth of an OC-48. Table 3.1 shows the cost of different WAN link types and the time to migrate 1,000 VMs with 50GB disk each. Note that this assumes the WAN bandwidth is consistent end-to-end from the enterprise datacenter all the way to the public Cloud. However, no ISP gurantees the same bandwidth in the Internet backbone as the access links. In addition, the inbound link to the public Cloud provider will become the bottleneck if there are lots of migration traffic from different edge datacenters.

Many Cloud providers including Amazon AWS and VMware vCHS already support the export and import of large data sets using portable storage devices. However, those export and import services only support offline VM relocation since the VMs have to be powered off before export in the source site, and will

WAN Link	Bandwidth (Mbps)	Monthly Cost	Time to Migrate 1K VMs
T3	44.76	$\sim$ \$5K	103 days
OC3	155	$\sim$ \$20K	29 days
OC12	622	$\sim$ \$200K	7 days

Table 3.1: Comparison of different WAN connection types, their costs, and the time required to move 1K VMs with 50GB disk each.

need to be powered on after import in the destination site. Migration over FedEx combines the benefits of both live migration and high efficient data movement via portable storage devices so that we can live migrate large numbers of VMs between geographically distributed locations.

3.1.2 Workflow

We show our Migration over FedEx framework in Figure 3.1. It basically contains 4 interrelated entities: **Source Host**, **Destination Host**, **Source Terminal**, and **Destination Terminal**, respectively. The source and destination hosts are located in two geographically separated data centers and connected by WAN. The source and destination terminals are two distinct machines connected to the source and destination hosts via a LAN, respectively, and function like two intermediate data transfer stations. The Cloud administrator can operate on the source/destination terminal to export/import a VM's base storage. The source and destination terminals are indispensible for Migration over FedEx because the source/destination host can not be physically accessed due to security and management consideration. By bringing in the source and destination terminals, we leave the source and destination data centers as independent objects from the point of view of migration operator and the design is technically simplified since there is no need for modifying the software stack of the virtual machine infrastructure on both sites.

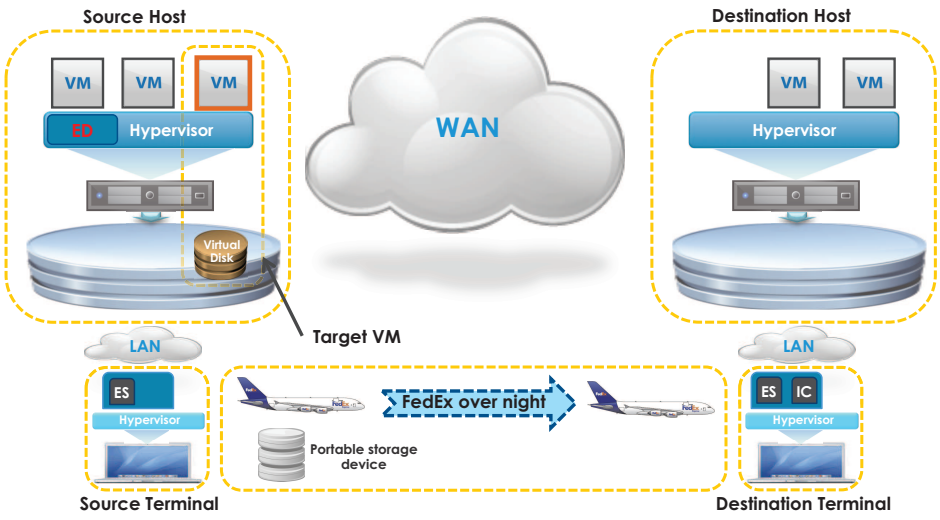


Figure 3.1: Migration over FedEx framework

The Migration over FedEx framework consists of two independent migration components: **Storage Migration over FedEx** and **Memory and Dirty-Storage Migration over WAN**. Storage Migration over FedEx takes care of base disk migration, which includes three successive steps: **Export**, **Shipping**, and **Import**. Those storage migration phases are associated with three storage migration components: **Export Driver (ED)**, **Export Server (ES)**, and **Import Client (IC)**. The Export Driver is implemented as a kernel module of virtual machine hypervisor and is in charge of reading out the base storage data block by block and sending them to the Export Server. The Export Server is a virtual machine running on the source/destination terminal (source ES and destination ES). The source ES is responsible for forwarding the data received from the ED to an export destination which could be either the source terminal itself or another machine within the same LAN. The destination ES takes charge of receiving the base storage data from the IC and transferring them to the destination host. After the portable storage devices containing the storage data are delivered, the Import Client, another VM running on destination terminal, will read out the data from portable storage devices and send them to the destination ES. All data transfers are performed using our LAN Storage Replication Protocol

(LSRP), which will be described in detail in the next chapter.

The complete workflow of Migration over FedEx includes 6 phases as follows:

1. Enable migration for a target VM and setup the Export Server, e.g. specify export destination and target disk group, etc.

2. **Storage Migration over FedEx - Export:** The Source Export Server exports the base storage of the target VM to portable storage devices associated with export destination (Figure 3.2)

3. **Storage Migration over FedEx - Shipping:** Shipping the portable storage devices containing the VM's base storage to the destination site via FedEx or other courier services (Figure 3.3)

4. **Storage Migration over FedEx - Import:** The Import Client reads and forwards the base storage data to the destination Export Server. The Export Server will import the data to destination host (Figure 3.4)

5. **Memory and Dirty-Storage Migration over WAN:** The Source host identifies the dirty storage blocks by looking up the bitmap maintained by the Export Driver and transfers those data to the destination host over the WAN concurrent with mirroring any new writes from the guest OS to the destination. Upon finishing dirty storage transfer, the memory and device state will then be transferred over the WAN (Figure 3.5)

6. Destroy source VM and resume destination VM if the migration succeeds

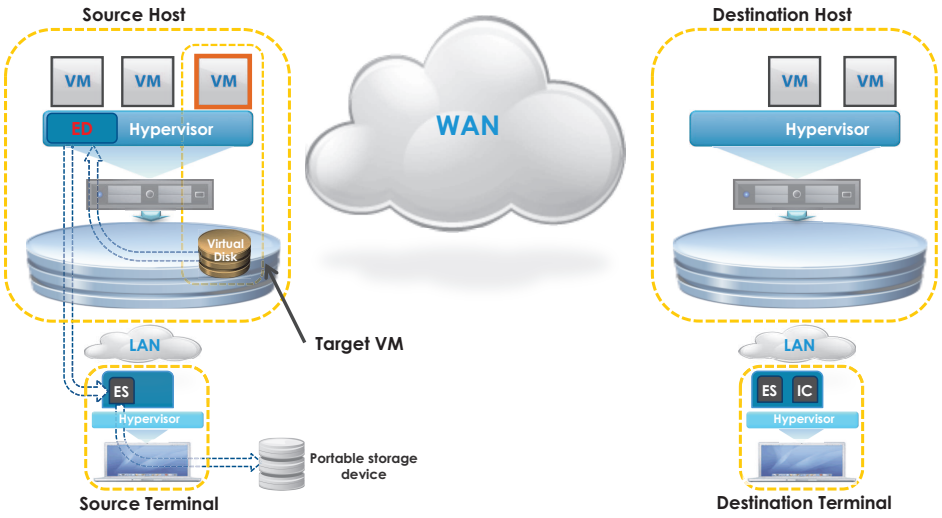


Figure 3.2: Exporting base storage to portable storage devices

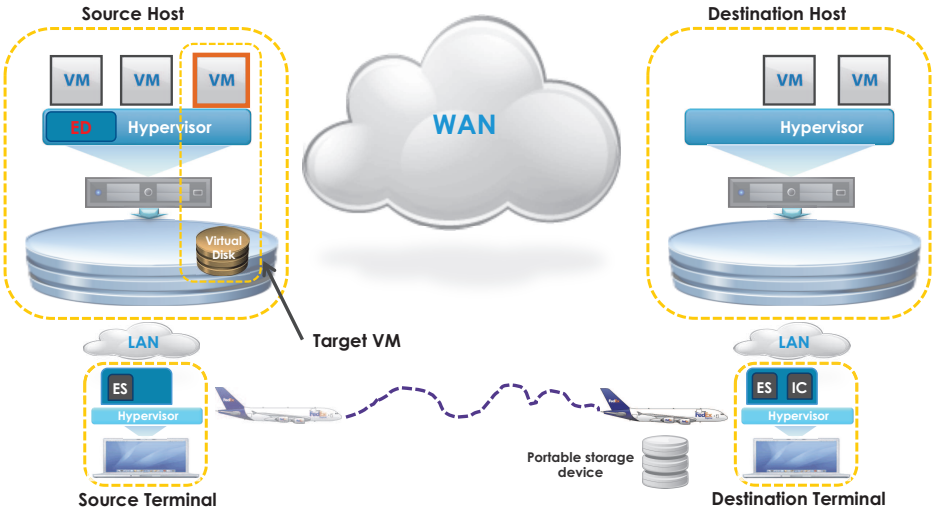


Figure 3.3: Ship the portable storage devices to destination site

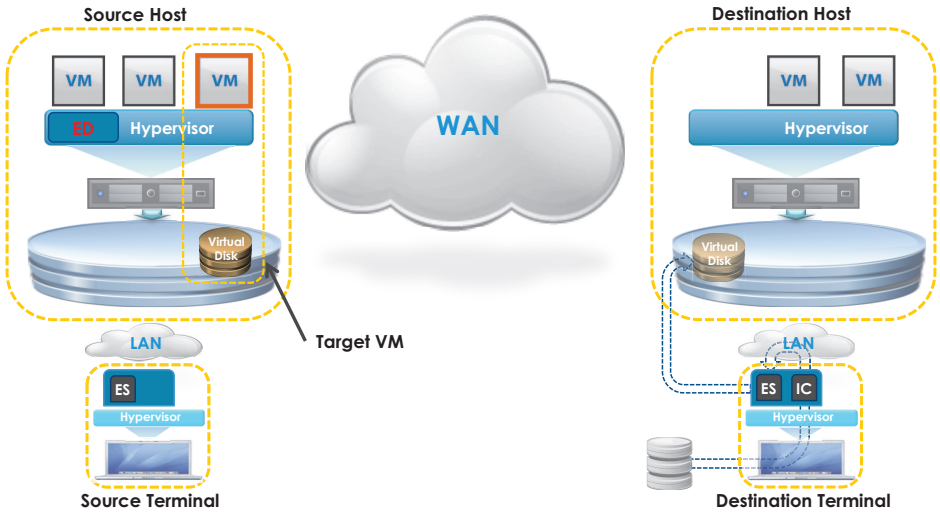


Figure 3.4: Importing base storage to destination site

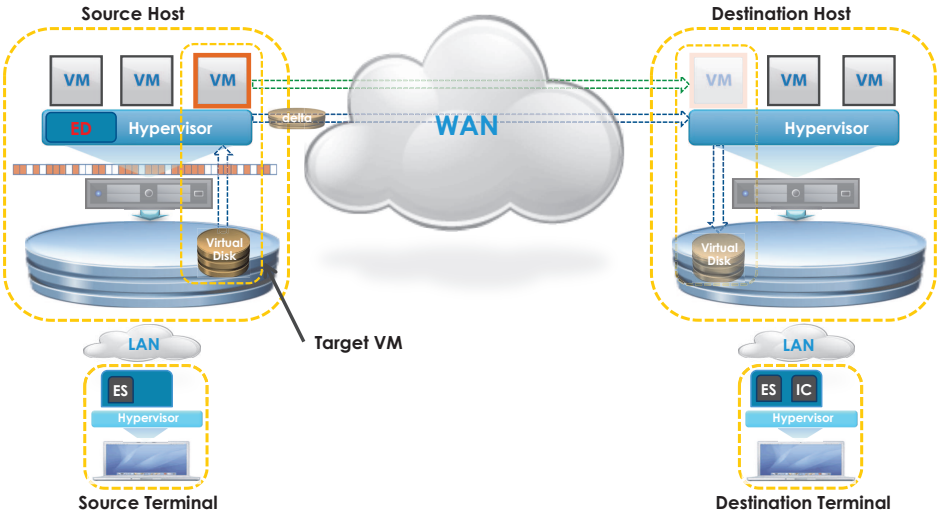


Figure 3.5: Migrating memory, device state, and dirty storage blocks to destination site

The amount of dirty storage data to be transferred completely depends on the guest write IO workload. Our study showed that most guest IO writes exhibit some degree of locality and will repeatedly write to a few hot disk regions. In addition, to tolerate possible damage of storage devices during shipment, we could export source VM disks to multiple devices using RAID-5 kind of striping with distributed parity. Therefore, on the destination side, we can tolerate at least a single device failure without losing the disk content.

3.2 Migration Components

Traditional live WAN migration usually takes days or even months to migrate large numbers of virtual machines. Instead of this inefficient way, our Cloud-scale live migration scheme employs **Storage Migration over FedEx** to migrate the base storage of target VMs. It takes at most one day to migrate the bulk of VM data to the destination anywhere in the nation. Then it utilizes the **Memory and Dirty-Storage Migration over WAN** component to migrate the dirty storage generated while the base storage was being exported, shipped, and imported. Upon finishing, the memory and other device state will be migrated.

3.2.1 Storage Migration over FedEx

Migrating a VM's base storage typically comprises the largest component of the overall migration time since the base storage may be in the tens or hundreds of gigabytes. Thus it is inefficient or even impossible to migrate large number of VMs at the same time over the WAN in an acceptable time period because of the WAN's limited bandwidth and high latency. The Storage Migration over FedEx is designed to overcome this limitation.

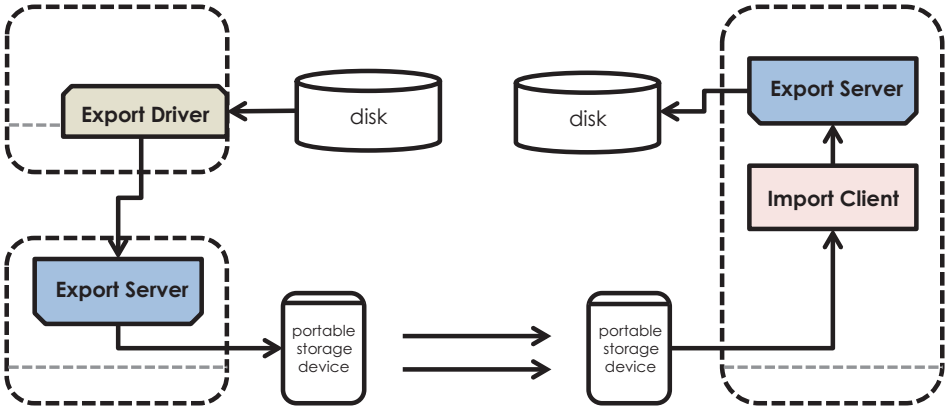


Figure 3.6: Data-flow path of Storage Migration over FedEx

Figure 3.6 shows the data-flow path of Storage Migration over FedEx. The Export Driver, Export Server, and Import Client together constitute the backbone of the client-server model. As the starting point, Export Driver is designed and implemented as a kernel module and we also implemented a layer in the VM kernel to include a framework that can be used for registering and unregistering the Export Driver after it is loaded into the kernel. In general, the Export Driver is responsible for keeping track of which regions of the virtual disk the guest is modifying and sending these updates to the destination site in a manner that results in a consistent replica. Specifically, it is responsible for:

- Sequentially reading the target base storage data block by block (block size is an experimental parameter).
- Packing each block read from the first step with LSRP header and transferring the package to Export Server.
- Maintaining a bitmap to track the dirty blocks of the target virtual disk after base storage migration starts.

We attach an instance of Export Driver to each disk that we are migrating. Each instance requires some configuration information that is specific to the

associated virtual SCSI device. Those options are stored in the virtual machine configuration file; when the virtual SCSI device is created these options are parsed and passed down as part of the driver attachment process. Each instance only cares about the device that it is attached to. The coordination of mutiple drivers is done in the management layer.

The Export Driver maintains meta-data about what regions of the virtual disk are empty as well as a set of blocks that have been updated that need to be re-send to the remote side. This meta-data is stored in a file on disk managed by the driver itself. When the driver is attached, we open the file and load the state from disk, when the driver is detached we flush any in-memory state to disk and close the file. All of the state kept in this file can be regenerated from the remote site, so in the event that this file is lost or not written out cleanly, we can recover by triggering a full replica creation. This operation allows the driver to rebuild its meta-data by comparing the state of the primary disk with the state of its remote copy. The process involves reading the entire disk contents on both sides, so it is relatively expensive.

Export Server is the server-side daemon process running in a virtual machine hosted by source/destination terminal indicated by the blue square in Figure 3.6. On the source side, it will first create a virtual disk (same capacity with the target virtual disk) in the portable storage device when storage migration starts. Then it receives the packages sent by Export Driver and unpacks them. Finally it writes the virtual disk data into the right place of virtual disk within the portable storage device. On the destination side, it will receive the packages sent by Import Client and unpack them, then transfer the data directly to the destination host.

Import Client is the client-side application running in a virtual machine (can be the same VM with Export Server or not) hosted by destination terminal. It is similar to the Export Driver but runs at the user level and does not maintain any bitmaps. The only work it will do is read the virtual disk data from the portable

storage devices and send them to the destination Export Server.

3.2.2 Memory and Dirty-Storage Migration over WAN

Most of the live memory migration architectures follow a similar iterative copy approach: they initially mark all memory pages as dirty and iteratively copy memory pages from source to destination. After a page is copied, a write trap will be installed and the corresponding page will be marked as clean. If the write trap is triggered, the page will be marked as dirty again. Then applying successive "iterative pre-copy" passes, each pass will copy remaining dirty pages left by the last iteration. At some point, the VM will be suspended and the remaining dirty pages along with the device state are sent to the destination. At the same time, DRBD disk replication system will mirror the new writes to the destination site.

Our Memory and Dirty-Storage Migration over WAN component acts analogously with traditional iterative copy approach, except that the dirty storage data is migrated prior to memory and device state. Specifically, it initially installs the DRBD disk replication system in the VM storage stack and mirrors IO to the destination host. In the meanwhile, by checking the bitmap maintained by Export Driver a separate thread identifies the dirty storage blocks and copies them from source to destination with a single pass, while guaranteeing synchronization with the DRBD mirroring. Finally, the virtual disks in source and destination will become consistent immediately after the initial dirty storage blocks are copied.

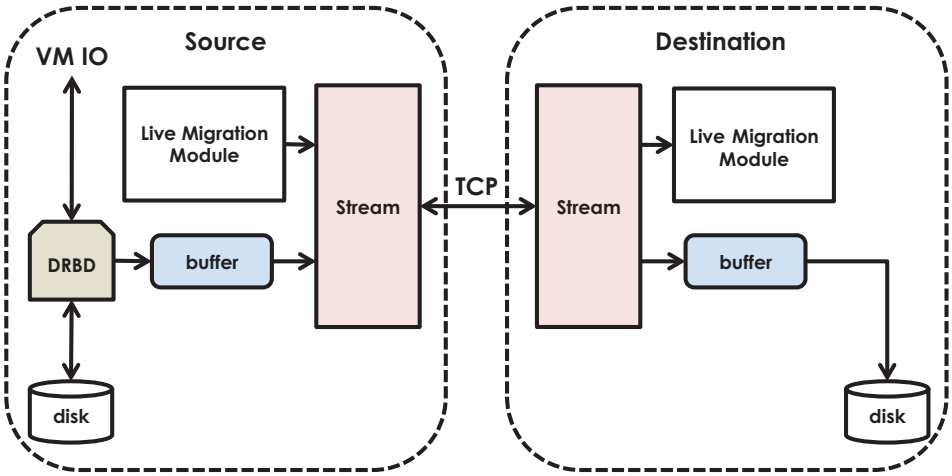


Figure 3.7: Data-flow path of Memory and Dirty-Storage Migration over WAN

Figure 3.7 depicts the data-flow path of the Memory and Dirty-Storage Migration over WAN. The DRBD disk replication system is used to interpose on all virtual disk writes and mirror them to both the source and destination disk. Disk buffering allows for the appearance of asynchronous IO mirroring. While the buffer has available space IOs behave asynchronously, meaning writes return immediately. Upon filling, subsequent write attempts are delayed until space becomes available in the buffer, effectively switching the guest IO to a temporary state of synchronous mirroring. Draining of the buffer is handled by the Streams bulk transport framework, discussed next.

The live migration module is responsible for many well-known tasks of live memory migration, such as locating memory pages for transmission, and appropriately handling the VM's virtual device state. It enqueues the memory pages and relevant device state for transmission by Streams.

Streams serve to abstract knowledge of the underlying network or communication channel from the rest of the live migration system. Delivery in Streams is generally out-of-order; this choice comes from how memory pre-copying in our live migration system works. A migration begins by linearly iterating over the VM's physical memory, queuing up each page for transmission as we install a

write trace on the page. We guarantee that we always make full passes over the VM's memory, never sending a given page more than once in any pre-copy iteration. Since we know we only transmit a given 4K guest memory page once per pre-copy iteration, we do not need to be concerned with the order the destination host receives those pages in. It will not harm correctness, for example, if the destination ends up receiving all of our transmitted pages backwards, provided it can distinguish between a page sent during the first pre-copy iteration and a page sent in the second iteration.

The relaxed ordering of many Streams consumers, such as pre-copy, make it easy for Streams to support mutipathing, as it can often buffer and transmit data in whatever order it wishes. In particular, it can opt to leverage multiple TCP connections between the source and destination, traversing different physical network interfaces.

For example, any number of live migration network interfaces can be configured, perhaps two 10GbE NIC at the source, one 10GbE NIC and two 1GbE NICs at the destination. Streams pair off adapters based on the expected maximum capacity of all adapters until either host is saturated. In the example given above, this would result in the source pairing its first 10GbE with the destination's sole 10GbE NIC and its second 10GbE with the destination's two 1GbE NICs, leaving the source pool with 8Gb of unused link speed capacity. We open one TCP connection per network adapter pair.

Streams can dynamically load-balance outgoing buffers over the TCP connections, servicing each connection round-robin as long as the socket has free space. This allows us to saturate any number of network connections, up to PCI bus limitations. Streams also allows for leveraging of non-network communications channels, such as a multi-writer buffer on shared storage, though we currently do not employ such channels by default. Streams rely on zero-copy transmit and single copy receive network APIs. We also decided to continue using TCP to

leverage hardware offload abilities that are not readily available for other transports. These choices proved critical to surpassing 10Gbps throughput, and almost doubled our single-adapter throughput with low CPU consumption.

Any ordering requirements that exist are expressed using a write barrier e.g., the Live Migration module will create a write barrier at the end of any-copy iteration - one linear pass over the VM's physical memory. Upon encountering such a barrier, the source host transmits the barrier over all communication channels to the destination host. The destination host, will pause reading data off each channel until all channels have read up to the barrier message. Once every channel has reached its barrier message, all channels will resume receiving page content. This guarantees that pages transmitted on the source on one side of a barrier message will not be interleaved with messages on the other side of this write barrier.

3.3 Conclusion

In this chapter, we first elaborated the necessity and feasibility for leveraging the capability of courier service to migrate the bulk of VM data - storage. Then we introduced our Migration over FedEx framework and its detailed workflow steps. After that, we talked about the design and working principles of each migration component. The Storage Migration over FedEx component is in charge of transferring the VM's base storage data, and the Memory and Dirty-Storage Migration over WAN is in charge of transferring the memory, device state, and dirty storage content.

Chapter 4
Protocol: LAN Storage Replication

For Cloud-scale live migration, the data transferred by Migration over FedEx falls into five catagories: base storage, dirty storage, memory, device state, and config files. In this chapter, we will elaborate the design and implementation of base storage migration mechanism. The base storage is actually migrated through the Storage Migration over FedEx component which leverages an application layer transport protocol to create a remote replica for the base storage of the target VM. We implemented this protocol, LAN Storage Replication Protocol, on top of Storage Migration over FedEx compoment as an independent service.

4.1 Overview

The motivation behind LSRP is that TCP/IP does not provide any ordering guarantees for data delivery across different connections. Besides, TCP/IP ensures in-order delivery of data on a single socket, but if a socket is closed the caller can make no assumptions about whether any of the data that was currently in-flight was processed by the remote site. This presents a problem, as updates from an older connection could end up with getting interleaved with updates from a newer connection. The following case is a example for such scenario:

1. Virtual machine M is being migrated from host A to host B and a storage data package P_1 is on the way to host B

2. Host A failed sometime when P_1 is in-flight and M is restarted on host C

3. Host C starts to migrate M to host B and another LSRP package P_2 is on the way to host B

Host B thought the connection between itself and host A is still valid since host A did not close it. Moreover, the in-flight package P_1 which is on the connection between host A and B could be delayed and get to destination later than P_2, in this case, if P_2 is stale it could end up corrupting the remote storage replica version.

To overcome the above defficiency, we designed an application layer transport protocol, LAN Storage Replication Protocol (LSRP), to perform consistent data transfer. The LSRP is developed to provide an efficient and consistent LAN storage replication mechanism at the granularity of virtual machine. VM-based replication simplifies the provisioning and management of storage relative to datastore-based solution, because the latter one requires specifying replication properties at the granularity of datastore. In a VM-based way, once the base storage migration is completed, we will immediately have a replica in the destination datastore. Next, we only need to migrate the dirty storage, memory, device state, and config files over the network. The use of LSRP to mgirate base storage gives the following benefits:

- Create consistent replica for the target VM's base storage with minimal performance impact by applying a "lightweight" Delta Consolidation protocol.
- Replicate a group of virtual disks of several different VMs in a way that guarantees the consistency of an application that spans those disks.
- Minimize the overall use of network bandwidth through using some sophisticated heuristics.

For each target VM, there is a *primary site* where the production copy of VM is executing and the *secondary site* where the replica of the VM lives. The

primary site can be specified for each target VM and it includes two components. The first component is LSRP Manager which is a host plugin providing interfaces for configuration and management purposes. The second component is Export Driver that runs as part of the VM kernel. Export Driver intercepts all I/O to the disks of target VMs that are powered on on the local host, tracks the dirty regions of disks, performs initial synchronization of target VM disks, coordinates the creation of consistent disk replicas and transfers the data to Export Server which can run either on secondary host or on an intermediate host.

4.2 Delta Consolidation Protocol

Our base storage migration mechanism follows an asynchronous replication model, where the replica state may fall behind the state of the primary host. When a new replica needs to be created, a straightforward way is to do another migration for all target virtual disks. Whereas, this naive approach is very inefficient and bandwidth intensive for migrating a large amounts of unchanged data. Instead of that, we choose to consolidate the changed data (delta content) into the secondary host as long as a new replica needs to be created (the dirty region of virtual disks in primary site is tracked by Export Driver). Nevertheless, we still need a protocol to ensure that the new replica is in a consistent state with the primary site.

We implemented the Delta Consolidation protocol to enforce the above consistency property. It will create an immutable image of the primary disk from which it copies any dirty regions. Typically, such an immutable image is created using some type of snapshot. However, generic snapshot mechanism is heavyweight and may impact the performance of the workload using the disk especially when the snapshot is created and/or removed. To minimize the performance impact of replica creation for the most common types of workloads, we utilize a lightweight log to record the under-affected data region. We show the basic principles of

Delta Consolidation protocol in Figure 4.2.

The Export Driver starts to migrate the dirty regions of a virtual disk as soon as the consolidation is required (step $w0$). The grayed-out blocks in the figure depict the dirty regions that need to be transferred to the remote site to result in a new consistent replica. During the period, Delta Consolidation is used to ensure the immutability of the dirty regions that need to be replicated. Instead of implementing copy-on-write functionality for all disk blocks, as is typically the case with generic snapshot mechanisms, we create a copy of the original dirty block contents, only if an overwrite to a dirty block is detected before the contents of that block are copied to the replica. Thus a write operation is allowed to be executed without any overhead (step $w''1$). If an overlap exists (step $w1$ and $w'1$), then the original block contents are copied (step $w2$ and $w'2$) in the lightweight log, before the write operation is allowed through (step $w4$ and $w'4$). Export Driver transfers the content of lightweight log to the secondary site in the same way it does for the dirty regions on the disk.

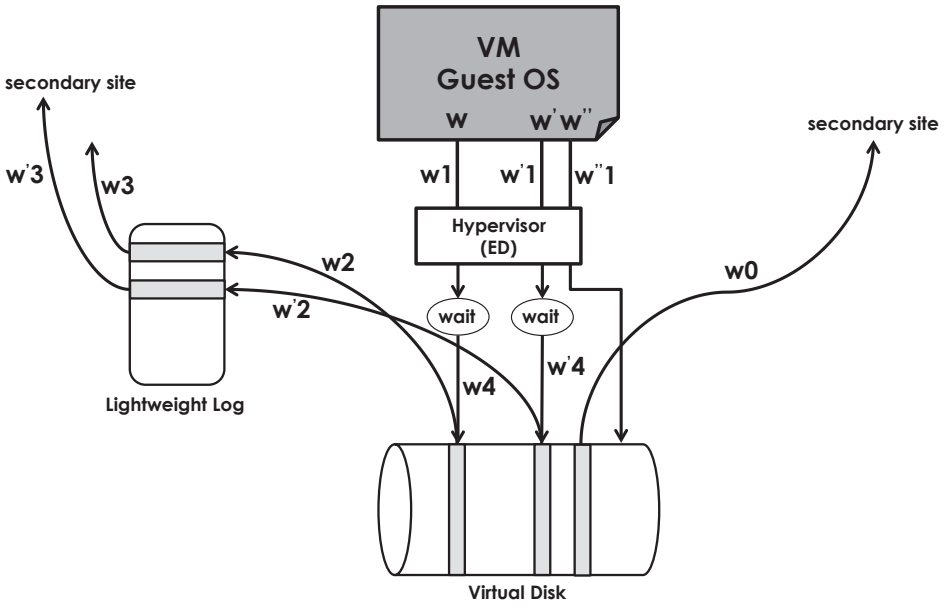


Figure 4.1: An overview of Delta Consolidation Protocol

With this approach, we can create an updated replica version when it is necessary by consolidating the dirty storage data into the remote replica in secondary site and there is no cost for snapshot initialization or collapse. There is an overhead due to additional I/O when an overwrite actually occurs, this overhead is incurred only the first time a dirty block is overwritten. Moreover, only the reading of the original data is synchronous with respect to the guest I/O, the write to the lightweight log is performed concurrently with the guest I/O execution. For typical workloads, such overwrites are limited, usually, there are just a few "hot" blocks in the working set of a workload.

4.3 Setting Up

Before base storage migration (export or import) starts, each migration component has to be set up first by migration administrator. LSRP Manager has provided various interfaces for configuring those components. After finishing the set-ups for each migration component, the administrator can start export by issuing a "start export" command in local terminal. LSRP Manager is in charge of receiving, translating those commands and control related component to perform corresponding actions.

The first component needs to be set up is Export Server. It is the core component for base storage migration no matter for export or import stage becasue that it connects the primary host and secondary host by constructing a migration group for the target storage and its remote replica path. The group is the unique identification for the target storage. The workflow steps for setting up Export Server is as follows:

- Launch the Export Server. Export Server can stay on either secondary host or an intermediate host
- Add the datastore host into the host list of Export Server, which will tell

the Export Server inside the VM which storage LUN it has access to

- Add a migration group by pairing the target virtual disks with its remote replica path

The migration group and datastore hosts added to Export Server will be stored in its internal database. Re-adding an identical group or datastore host is not allowed unless the original database is deleted, a new server is started, or upgrading to a server which uses a different database version. Creating replicas for a group virtual disks (either of the same or of multiple different VMs) at the same time is also supported by LSRP. This function can be enabled by adding all wanted target virtual disk ID and their remote replica disk path pairs in the third step of the above workflow. The Export Server can differetiate which target disk a replica belongs to by checking the group identification.

The second component needs to be set up is primary host. It is the client end where the LSRP Manager and Export Driver (or Import Client) live. The primary host must have some necessary management knowledge about the target VMs running on it. For example, it have to know which VM is currently configured as "replication enabled" status as well as which virtual disks of a targe VM are active for migration. LSRP Manager also provides any necessary interfaces for setting up the VMs or obtaining any configuration information. The setting up steps for primary host is as follows:

- Obtain the target virtual machine ID
- Enable the replication function for target VM

The virtual machine ID can be easily obtained by issuing a "get VM id" command. After enabling the replication mode, the target VM is then connected to the Export Server through a group of TCP ports. The virtual machine ID and TCP port group will be automatically set to default values if not be explicitly

specified in the parameter list. Each time the replication is configured, the Export Server have to be re-configured with the matching group information. The per-VM replication mechanism can also be turned off by executing a "disable replication" command.

The last component needs to be set up is the secondary host. For base storage export, secondary host is also where the Export Server lives. So all we have to do is to create a empty replica file (flat virtual disk file) since the LSRP expects pre-exsiting disk into which replica data will be copied. The replica disk file have to be the same size with the original disk.

4.4 LSRP Specification

The LSRP is an application layer transport protocol implemented to coordinate each LAN storage migration component: Export Driver, Export Server, Import Client, primary host, and secondary host. As a result, the base storage is migrated by creating a replica in the destination datastore. Besides, LSRP also supports to consolidate the delta content (dirty storage data) into the corresponding remote replica. Delta consolidation has some very important advantages, e.g., it allows the migration administrator to check out the base storage into the portable storage devices days or even months prior to the shipping date and consolidate the delta content periodically until when they decide to ship the portable storage devices to the destination site. Moreover, it is also helpful for looking for a time point when the amount of dirty storage becomes stable, which can effectively reduce the WAN traffic and shorten the WAN migration time.

The LAN storage migration components communicate with each other by sending messages. There are three types of messages: data message, control message, and event message. The data messages are dedicated used by the Export

Driver (or Import Client) and the Export Server for base storage migration purpose. The control messages are sent by the LSRP manager to the Export Driver, which is a way for the migration administrator to control the behaviour of Export Driver. Most of the control messages will be finally converted into corresponding data messages to perform the real actions. The event messages are sent by the Export Driver to the LSRP manager for notifying it if the requested operations are successfully executed.

CODE	MESSAGE NAME	TYPE
D1	LSRP_DATA_GET_SERVER_STATE	Data
D2	LSRP_DATA_HANDSHAKE	Data
D3	LSRP_DATA_INIT_SESSION	Data
D4	LSRP_DATA_REPLICATION_START	Data
D5	LSRP_DATA_REPLCIATION_CHECKSUM	Data
D6	LSRP_DATA_REPLICATION_UPDATE	Data
D7	LSRP_DATA_REPLICATION_COMPLETE	Data
D8	LSRP_DATA_TRANSFER_CONFIG_FILE	Data
D9	LSRP_DATA_DELTA_CONSOLICDATION_START	Data
D10	LSRP_DATA_DELTA_CONSOLIDATION_UPDATE	Data
D11	LSRP_DATA_DELTA_CONSOLIDATION_COMPLETE	Data
C1	LSRP_CONTROL_GET_SERVER_STATE	Control
C2	LSRP_CONTROL_REPLICATION_START	Control
C3	LSRP_CONTROL_TRANSFER_CONFIG_FILE	Control
C4	LSRP_CONTROL_GROUP_PREPARE_AND_COMMIT	Control
C5	LSRP_CONTROL_DELTA_CONSOLIDATION_START	Control
C6	LSRP_CONTROL_DELTA_CONSOLIDATION_UPDATE	Control
E1	LSRP_EVENT_GET_SERVER_STATE	Event
E2	LSRP_EVENT_REPLICATION_COMPLETE	Event
E3	LSRP_EVENT_TRANSFER_CONFIG_FILE	Event
E4	LSRP_EVENT_DELTA_CONSOLIDATION_RESULT	Event
E5	LSRP_EVENT_DELTA_CONSOLIDATION_COMPLETED	Event

Table 4.1: LSRP messages

We list all messages and their corresponding types in Table 4.2. There are 11 data messages, 6 control messages, and 5 event messages in total. LSRP is initialized as soon as the base storage migration starts. In the process of initialization, a connection between the Export Driver and Export Server will be constructed as well as some sanity checks and identity confirmation. Once the LSRP initialization is finished, the base storage migration starts immediately. Then all storage data will be sent to the Export Server block by block, each

block will be sent as a LSRP package. Delta consolidation is activated by an independent command, which can only issued after the base storage migration is completed.

4.4.1 LSRP Initialization

LSRP is immediately initialized as soon as a base storage replication is launched by the LSRP manager. The initialization process includes two parts: Export Driver initialization and LSRP manager initialization. Upon successful initialization, some important server-side states for the target VM will be returned back to the Export Driver and a session will be constructed for each migration group between the Export Driver and the Export Server. All disk-related traffic is done within the context of the latest sessions established between the Export Driver and the Export Server. The Export Driver will establish a new session with the Export Server when it needs to start or resume replication for a disk that is replicated to the corresponding target. Specifically, the Export Driver has to establish a new session whenever the VM is powered on.

The notion of a session is separate from that of a connection in the network substrate. If an underlying connection for a session gets reset it can simply be re-established. The important point is that it is up to the Export Server to determine whether a disk's session is no longer valid. For the layer above a resume communication with the remote site, it has to re-establish the session. From an implementation perspective, sessions are represented by session IDs. The ID for a specific disk session is monotonically increasing over time.

CODE	MESSAGE NAME	TYPE
D1	LSRP_PROTOCOL_GET_SERVER_STATE	Data
D2	LSRP_PROTOCOL_HANDSHAKE	Data
D3	LSRP_PROTOCOL_INIT_SESSION	Data
C1	LSRP_CONTROL_GET_SERVER_STATE	Control
E1	LSRP_EVENT_GET_SERVER_STATE	Event

Table 4.2: LSRP initialization messages

We show all involved messages for LSRP initialization in Table 4.3. Three data messages are all going to be sent from the Export Driver to the Export Server for requesting states, handshaking, and initializing session. The Export Driver will notify the LSRP manager by sending event message when the state information is successfully received from the Export Server. The Export Driver side initialization has following two steps:

- Export Driver issues a handshaking message $D2$ to Export Server to confirm that both side are using the same LSRP version etc.. $D2$ includes a unique identification number which is expected to be returned by Export Server
- Upon receiving expected response, the Export Driver then issues a session initialization message $D3$ to the Export Server, in the meanwhile, requesting the virtual disk information (e.g., capacity) which is provided by the administrator when setting up the Export Server

LSRP Manager side initialization has following 5 steps:

- LSRP Manager issues a ioctl message $C1$ to the Export Driver for requesting status information (number of virtual disks etc..) from the Export Server
- Upon receiving requests from the LSRP manager, Export Driver issues message $P1$ to the Export Server to request the needed information
- Export Server responds to the Export Driver with requested information
- Upon receiving responds, Export Driver posts an event message $E1$ to notify LSRP Manager that the requested information is ready
- LSRP Manager processes the event and finish initialization

We show the complete protocol flow of LSRP initialization in Figure 4.3. On the local side, the LSRP manager and Export Driver both reside in primary host

which is indicated by a grey dotted circle. On the remote side, the Export Server communicates with primary host by exchanging WAN messages (LSRP data messages) which are indicated by red dotted arrows. Accordingly, the solid arrows (green and blue) represents LAN messages. The message types are differentiated by three colors. The green arrow is control message sent from the LSRP manager to Export Driver, the blue arrow is the event message sent from the Export Driver to LSRP manager, the red arrow is data message sent between the Export Driver and Export Server.

Figure 4.2: Protocol flow of LSRP initializtion

4.4.2 Full Replica Creation

This protocol is utilized to replicate the base storage of the target VM. The replication process is exactly the way we migrate the base storage of the target VM. Full replica creation ensures that a consistent replica from the primary site is always reflected on an immutable image in the secondary site. It is concerned with on disk only and implemented by injecting special control message in the update stream of the disk to indicate that all updates up to that point constitue a consistent replica. After the Export Driver has received acknowledgements for all updates that complete a consistent replica, it sends a control message to the Export Server. Upon receiving the message, Export Server will take a snapshot of the replica file to capture the consistent replica. Only after the snapshot is created successfully does it act the control message back to the Export Driver. The Export Driver can then continue with any new updates that it may have to send to the secondary site.

The snapshot mechanism is not necessary for Full Replication Creation itself. But when we bring in Delta Consolidation, we will talk about it in the next section, it becomes essential for preventing the remote replica disk from corrupting because of the failure of delta consolidation. For example, we launched a delta consolidation operation in the primary host, without snapshot mechanism, the dirty storage data will be directly written into the remote replica disk. Unfortunately this delta consolidation operation finally failed because of networking or other problems. Therefore the Export Driver has to roll-back while the remote replica has been ruined, which causes that the replica disk is in an inconsistent status with the primary site.

A full replica creation is enabled by executing a “full replication start” command in the administrator terminal. The command will be translated into a RPC call by LSRP manager on primary host, this RPC call will first initialize LSRP itself and then start replica creation, namely, export/import the base storage.

CODE	MESSAGE NAME	TYPE
D4	LSRP_PROTOCOL_REPLICATION_START	Data
D5	LSRP_PROTOCOL_REPLICATION_CHECKSUM	Data
D6	LSRP_PROTOCOL_REPLICATION_UPDATE	Data
D7	LSRP_PROTOCOL_REPLICATION_COMPLETE	Data
C2	LSRP_CONTROL_REPLICATION_START	Control
E2	LSRP_EVENT_REPLICATION_COMPLETE	Event

Table 4.3: Full Replica Creation messages

We list all involved messages in Table 4.4. Each $D6$ message includes a LSRP header and a base storage data block. Upon received it from Export Driver, the Export Server will extract the data block and write it into corresponding place in the portable storage device (for export stage). The detailed workflow steps of full replica creation is as follows:

- LSRP manager decides to create a full replica for the target VM. It sends $C2$ to Export Driver indicating the set of virtual disks that is going to create a group consistent replicas for
- Upon receiving $C2$, the Export Driver then sends $D4$ to the Export Server to start full replication for each target virtual disk.
- Export Driver computes checksums for all blocks on disk, and sends $D5$ to Export Server for all blocks. It sets the bits for any blocks with non-matching checksums.
- Export Driver transfers data blocks with set bits by sending $D6$ to Export Server, in parallel with the above step.
- When transfer is complete, Export Driver will notify that all consistent data has been received by sending $D7$ to Export Server
- Upon receiving $D7$, the Export Server will update the database to indicate there is a snapshot and takes an actual snapshot. Then response back to Export Driver

- Upon receiving the response, Export Driver sends E2 to LSRP manager. LSRP manager will declare the base storage migration complete when received $E2$ for each target disk.

We show the complete protocol flow of LSRP initialization in Figure 4.4.

Figure 4.3: Protocol flow of Full Replica Creation

4.4.3 Delta Consolidation

Delta consoliation is a way to create a new replica version by sending the dirty storage data to remote site. If we decide that we want to transfer a dirty block and consolidate the content into remote replca. We will atomically mark this block clean in the bitmap. Once this is done we read the block from disk and

send it to the remote site. Based on the result of the transfer we need to update our state. If the transfer fails, we can continue to retry till it sucesses or give up and mark the block as dirty in the bitmap. If the transfer succeeds, once the remote site acknowledged that the block is on disk then we have nothing need to do in the primary site. We listed all involved messages for delta consolidation in Table 4.5 and the detailed workflow steps of delta consolidation is as follows:

CODE	MESSAGE NAME	TYPE
D8	LSRP_PROTOCOL_TRANSFER_CONFIG_FILE	Data
D9	LSRP_PROTOCOL_DELTA_CONSOLIDATION_START	Data
D10	LSRP_PROTOCOL_DELTA_CONSOLIDATION_UPDATE	Data
D11	LSRP_PROTOCOL_DELTA_CONSOLIDATION_COMPLETE	Data
C3	LSRP_CONTROL_TRANSFER_CONFIG_FILE	Control
C4	LSRP_CONTROL_GROUP_PREPARE_AND_COMMIT	Control
C5	LSRP_CONTROL_DELTA_CONSOLIDATION_START	Control
C6	LSRP_CONTROL_DELTA_CONSOLIDATION_UPDATE	Control
E3	LSRP_EVENT_TRANSFER_CONFIG_FILE	Event
E4	LSRP_EVENT_DELTA_CONSOLIDATION_RESULT	Event
E5	LSRP_EVENT_DELTA_CONSOLIDATION_COMPLETED	Event

Table 4.4: Delta Consolidation messages

- LSRP manager decides to consolidate the delta for the target VM. It sends $C5$ to Export Driver indicating the set of virtual disks that is going to be consolidated
- Upon receiving $C5$, the Export Driver then sends $D9$ to the Export Server to start delta consolidation for each target virtual disk
- After get responsed by Export Server, Export Driver will send $E4$ to LSRP manager to notify that delta consolidation starts
- Once $D9$ has succeeded, the LSRP manager will initiate file transfer by sending $C3$ to Export Driver
- Upon reveiving $C3$, the Export Driver will send $D8$ to Export Server to

transfer each chunk of config file and response to Export Driver when finishing transfer

- Export Driver will send $E3$ to LSRP manager to notify that config file transfer is finished
- Upon reveiving $E3$, LSRP manager will send $C4$ and $C6$ to Export Driver. $C3$'s PREPARE part does the following:
  - Any update which is in-flight will be marked as dirty and no new update will be sent
  - Update the dirty block bitmap which records the regions that must be sent to the remote site
  - Block the completion of any IO operations from being delivered to the guest
  - Start an asynchronous task that walks the transfer bitmap and sends each dirty region to Export Server by sending $D10$ (controlled by $C6$)
  - Acknowledges the PREPARE part is successfully executed
- Upon receving a successful response to PREAPRE for each target disk, $C4$'s COMMIT part will be executed:
  - Unblock completion of IO operations, if there are any queued completions it delivers them to the guest
  - Acknowledges the COMMIT is successfully executed
- Once the Export Driver has sent all of the dirty regions, it sends $D11$ to Export Server instructing the server to take a snapshot.
- Once get responsed by Export Server, the Export Driver will send $E5$ to LSRP manager to indicate that the delta consolidation is completed

- Once the Export Server receives $D11$, it writes an entry into it's database to mark that it has a new replica for the VM
- Once the LSRP manager has received $E5$ for all target virtual disks, it will claim that delta consolidation is completed

The complete protocol flow is shown in Figure 4.5.

Figure 4.4: Protocol flow of Delta Consolidation

4.4.4 Future

We discuss here a number of possible optimizations we could do in the future for the full replica creation protocol flow. There are two areas for improvement:

- Minimize the data transferred in export stage by avoiding transferring blocks that are the same in primary and secondary sites
- Avoid re-starting from scratch if full replica creation was going on either upon a failure or upon VM power off/on (because of live migration or disaster recovery)

For the first optimization area, our first potential solution could be avoiding transferring storage blocks which are all 0s. If one or more consecutive blocks are all 0s just send some meta-data indicating so, instead of the actual block contents. The second solution is to use content-based digests to transfer only blocks that differ between both sites. The basic idea is that as part of the initial handshake protocol, the Export Driver will request cryptographic digests for the contents of the secodary disk image. The Export Server scans the secondary disk replica and provides the digests at some predetermined granularity. As the filter scans and reads the contents of the primary disk, it also generates content-derived digests, which are compared with those provided by the Export Server. Blocks are send to the secondary only when the corresponding digests do not match. But there are still a number of issues needing to be considered.

- The lock-step of the two sites scanning through the disk replicas
- The properties of the hash functions used to calculate the digests
- The details of the digest-generation algorithm and how it is optimized to not affect negatively the performance of disk scanning
- the granularity of the blocks on which digests are calculated and their relation to the granularity of the blocks transferred
- The feasibility of defining the granularity of those blocks dynamically

4.5 Conclusion

In this chapter, we detailed introduced the LSRP, LAN Storage Replication Protocl, which is an application layer transport protocol to migrate base storage by creating replicas for the virtual disks of target VM. LSRP includes a lightweight Delta Consolidation protocol which can merge the dirty disk data into the replica in the "remote" site. Thus we can update the replica in the local portable storage device periodically till deciding to send those to the secondary site. This is a very efficient approach to reduce the amount of dirty storage data which will be migrated through WAN. We also discussed the improvements we could do in the future for LSRP protocol.

Chapter 5
Evaluation

5.1 System Configuration

We run all tests on a pair of HP ProLiant DL580 G7 servers. Each of them is a quad-socket machine configured with eight-core 2.67 GHz Intel Xeon E7-8837 processors, 256 GBs of RAM, and two 10GbE Intel network adapters. Both servers are connected to an EMC CX4-480 and an EMC CX3-40 SAN arrays using an 8Gb Fibre Channel (FC) switch. We created a 100GB virtual machine file system on a 15-disk RAID-0 volume on each FC array.

5.2 Benchmarks

We compare the performance of our Cloud-scale live migration approach for three benchmarks: an idle VM, Online Transaction Processing using Iometer, and the DVD Store verion 2 (DS2). The configurations of these three VMs are shown in Table 5.1.

	CPU#	RAM(G)	DISK(G)		
			BOOT	DATA	LOG
IDLE	2	8	16	24	N/A
OLTP	2	8	16	24	N/A
DS2	4	16	50	90	24

Table 5.1: Configurations of three benchmark VMs

Our synthetic workload uses Iometer to generate an IO pattern that simultes an OLTP workload with a 30% write, 70% read of 8KB IO commands to the 24GB

data disk. During the migration for IDLE and OLTP, both the data disk and boot disk are migrated at the same time, but for OLTP there is no workload running on the boot disk. The DVD Store version 2 workload is an online e-commerce test application with a web front-end and database. The test was configured with 50 million customers and tested with 12 DS2 users. During the migration, all the three disks are migrated at the same time.

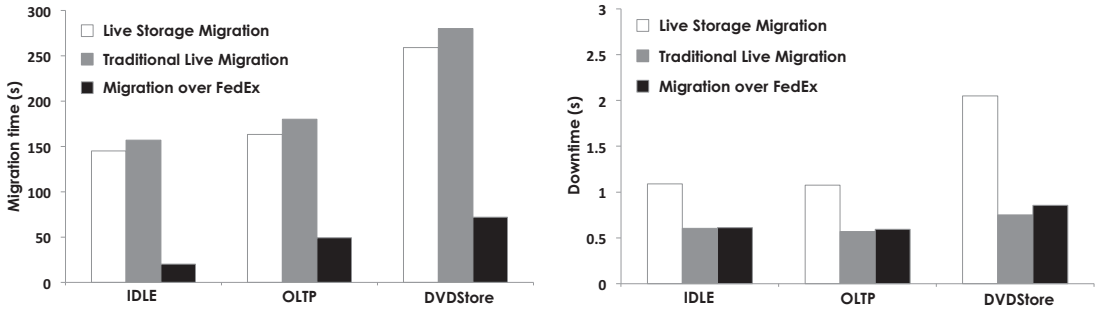
5.3 Migration over FedEx

We compare Migration over FedEx to local live storage migration and traditional live migration. Local live storage migration works by copying a virtual disk from one shared storage device to another on the same host. This is perhaps a bit of an apples to oranges comparison as Cloud-scale live migration must also deal with migrating memory and device state. We choose this comparison for two reasons. First, storage migration overhead generally dominates memory migration overhead. Next, storage migration is quite heavily optimized, so it represents a good best-case for comparison. Traditional live migration transfers everything over network while Migration over FedEx transfers the base storage data by shipping the portable storage device where the storage data is stored.

5.3.1 Migration Time and Downtime

Figure 5.1(a) shows the migration time for three workloads under three migration configurations. As expected, Migration ove FedEx is the fastest since it avoids transferring the bulk of storage data over network. The actual time for transferring the dirty storage data depends on the characteristics of running workloads. Taking the time needed for shipping the portable storge devices into consideration (assuming 12 hours), we list the threshold number of migrated VMs when Migration over FedEx exceeds traditional live migration in Table 5.2. Traditional

live migration is nearly 10% slower than local live storage migration because that it also needs to deal with the additional memory and device state. For example, there are 12 seconds difference between local storage migration and traditional live migration for idle VM and that difference grows to 17 seconds for the OLTP workload. The overhead is mostly from traditional live migration that requires about 9 to 10 seconds to migrate the memory state.



(a) Migration time (b) Downtime

Figure 5.1: Migration time & Downtime for three workloads under three migration configurations

	IDLE	**OLTP**	**DVDStore**
Threshold # of migrated VMs	345	361	251

Table 5.2: Threshold number of migrated VMs when Migration over FedEx exceeds traditional live migration

In Figure 5.1(b), we show the downtimes involved in three workloads against different migration configurations. As expected, the downtime of Migration over FedEx is a little bit higher for all three cases than that of traditional live migration. That is because Migration over FedEx transfers the dirty storage data over network. We expect that all Migration over FedEx have effectively a bounded downtime of a few seconds, which is composed of the transfer time for any remaining data and a few round trips for the resume handshake. Our local live storage migration in these tests suffered greater downtimes because it is very expensive to hand over the memory ownership when memory size is big. Compared

with traditional live migration, our approach does not incur apparent overhead but greatly shortens the total migration time especially when migrating large number of VMs.

5.3.2 Dissection

Figure 5.2 and 5.3 show dissections of DVD Store's reported workload throughput during migration time for traditional live migration and Migration over FedEx. These VMs are migrating from a higher datastore to a slower one, as shown by the throughput on the source and destination, during the migration the workload throughput reduces. We can notice a quick dip to zero in both of the figures as the VMs are suspended for switchover. Besides, there is a short period of time where the workload is not experiencing any performance penalty, and this is during the memory pre-copy.

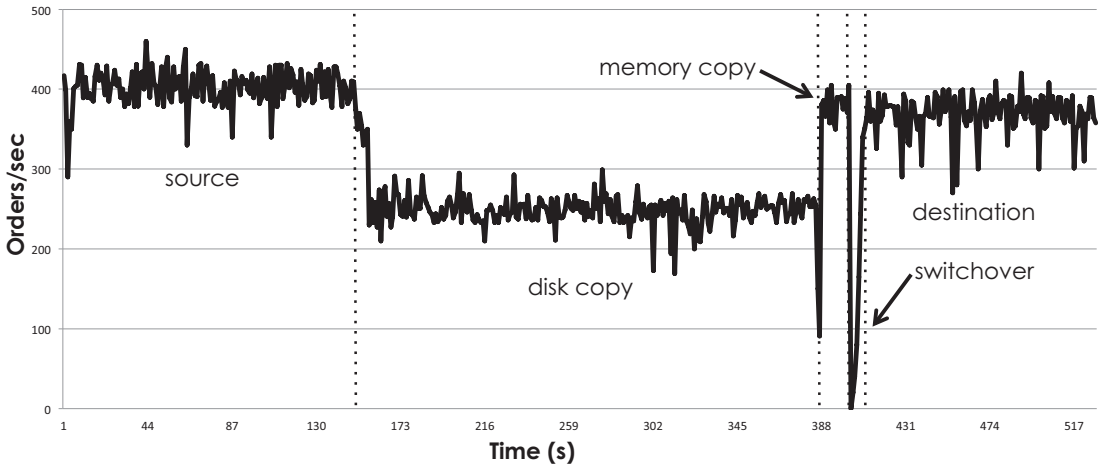


Figure 5.2: Illustrates the various phases of the migration against a plot of DVD Store Orders/sec for traditional live migration

For both traditional live migration and Migration over FedEx, the throughput dramatically decreased by almost 70 orders/sec when disk copy phase starts. This is mainly because that disk copy occupies nearly half of the disk bandwidth. In

addition, when the coming writes fall into the disk region where we are currently copying from, they will be queued until this disk region has been copied. This synchronization also causes negative effect on throughput to some extent. Memory copy does not introduce any overhead for the throughput that is because we have finished disk copy and any writes will be mirrored to both the source and destination sites.

Disk copy and memory copy dominate the migration time for both traditional live migration and Migration over FedEx. Thanks to the way we migrate storage by shipping the portable storage device, the storage migration time has been shrinked to one fifth compared with traditional live migration.

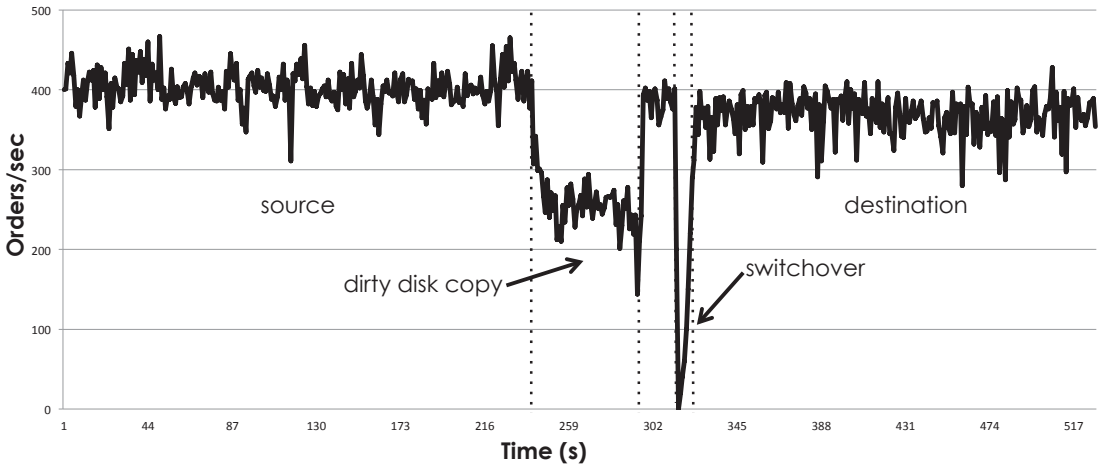


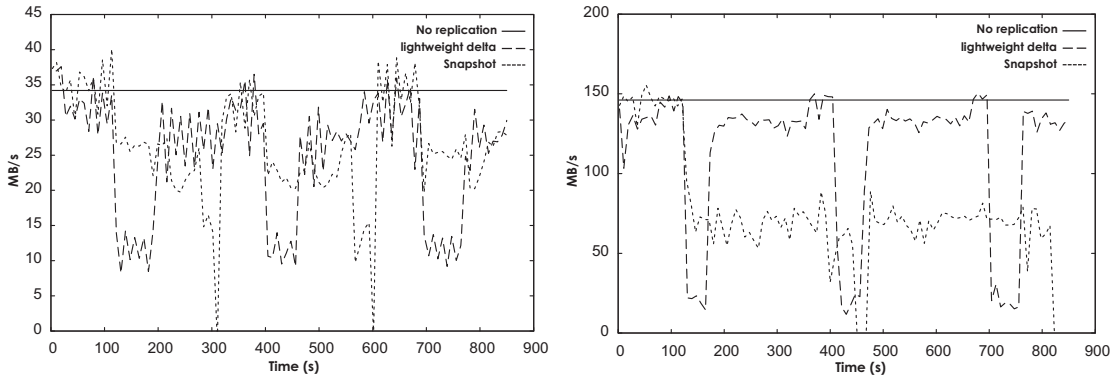
Figure 5.3: Illustrates the various phases of the migration against a plot of DVD Store Orders/sec for Migration over FedEx

5.4 Delta Consolidation

We investigate the advantages of Delta Consolidation protocol by comparing it to snapshot mechanism. DCP is a lightweight protocol used to creates a consistent storage replica.

5.4.1 Performance Results

Figure 5.4. presents experimental performance results that show the impact in terms of sustainable I/O throughput to a workload running in a VM. The graphs compare the use of DCP vs. snapshot as the mechanism to achieve primary disk immutability during delta consolidation. They capture an execution period of 15 minutes during which we create three replicas.



(a) Sequential writes, 4KB block size, 1GB disk (b) Sequential writes, 64KB block size, 1GB disk

Figure 5.4: DCP vs. Snapshot for creating consistent disk replicas

The workload in the VM tries to consume as much disk bandwidth as possible. In the case of 4KB writes, both mechanisms introduce some overhead, but snapshot approach drives the throughput down to 0 for a few seconds. In the 64KB case, the throughput of snapshot approach is much lower than DCP in most of the time and takes so long that it is not done in time for the next replica creation. The results indicate that DCP is a more appropriate protocol especially when consectively creating replica in short time interval.

Chapter 6
Discusstions and Futrue

We have shown our Cloud-scale live migration framework and the detailed design for LAN storage migration protocol. This is however a prototype design and needs to be studied further from some more important aspects e.g. performance and security. We list some future studies here.

6.1 VM Distribution Network

In addition to Migration over FedEx, we can optimize and accelerate the WAN traffic for memory and dirty-block migration using a distributed content de-duplication network. We have two observations to prove the necessity and feasibility of VMDN. The first observation is that in a virtualized environment, many VMs should exhibit the same similarity in their disk conten since those VMs are often deployed from a few common VM templates. For example, the VMs could have the same Windows OS and applications installed in the guest. Therefore, there will be lots of duplicated disk blocks for the same executable binaries as shown in [22]. Many VMs in an enterprise datacenter already use linked-clone features to explicitly share the same base disk image. The second observation is that due to the nature of Cloud-scale live migration, the WAN bandwidth will be very asymmetric between source and destination datacenters. If there are hundreds even thousands of users who want to extend their private datacenters to public Cloud, it is very difficult if not impossible to provide the inbound WAN bandwidth that is even close to the total aggregated bandwidth of those users.

We will study different cache strategies and algorithms to globally optimize the memory and dirty-block migration traffic. Our idea is: VMDN consists of a set of virtual applicances that can be deployed in both private Cloud and public Cloud. Each VMDN node will have a dedup engine as well as local cache for popular disk contens. The cache index of each VMDN node is propagated among other VMDN nodes so that any duplicated data transfer can be suppressed by only sending the content has rather than the actual content data.

6.2 Open Live Migration Protocol

Our Migration over FedEx framework is contructed on a open networking environment which may cause the communication to happen between untrusted hosts. So we have to redesign our memory and dirty-block migration protocol to prevent from malicious attacking. The original live migration protocol does not inherently require trust between the communicating hosts. Rather, the trust extends naturally from the type of VM metadata that must be transferred in order to move a running VM's state. In order to harden the protocol, it is critical to review each and every type of metadata transferring during a VM's migration, ensuring that proper sanity checks and content validation are performed.

There is some additional room for improvement here, we can ask the user to sacrifice some degree of VM fidelity in order to simplify the migrating. Such an approach allows us to greatly reduce the scope of any necessary migration protocol to just the core requirements of VM migration: moving the VM's memory state and virtual device checkpoint.

References

[1] S. Akoush, R. Sohan, A. Rice, A.W. Moore, and A. Hopper. Predicting the performance of virtual machine migration. In *Proceedings of the 2010 International Symposium on Modeling, Analysis, and Simulation of Computer Systems (MASCOTS2010)*, pages 37–46, 2010.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP19)*, pages 164–177, 2003.

[3] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schioberg. Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd international conference on Virtual Execution Environment (VEE07)*, pages 169–179, 2007.

[4] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd Symposium on Networked Systems Design & Implementation (NSDI05)*, pages 273–286, 2005.

[5] VMWare Inc. VMWare ESX. http://www.vmware.com.

[6] B. Gerofi, H. Fujita, and Y. Ishikawa. Live migration of process maintaining multiple network connections. *IPSJ Transactions on Advanced Computing Systems (ACS29)*, 3(1), 2010.

[7] D. Gupta, S. Lee, M. Vrable, S. Savage, A.C. Snoeren, G. Varghese, G.M. Voelker, and A. Vahdat. Difference engine: Harnessing memory redundancy in virtual machines. In *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI08)*, pages 309–322, 2008.

[8] S. Hacking and B. Hudzia. Improving the live migration process of large enterprise applications. In *Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing (VTDC09)*, pages 51–58, 2009.

[9] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall. The efficacy of live virtual machine migrations over the internet. In *Proceedings of the 3rd international workshop on Virtualization Technology in Distributed Computing (VTDC07)*, pages 1–7, 2007.

[10] T. Hirofuchi, H. Nakada, H. Ogawa, S. Itoh, and S. Sekiguchi. A live storage migration mechanis and its performance evaluation. In *Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing (VIDC09)*, 2009.

[11] T. Hirofuchi, H. Ogawa, H. Nakada, S. Itoh, and S. Sekiguchi. A live storage migration mechanism over wan for relocatable virtual machine services over clouds. In *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid09))*, pages 460–465, 2009.

[12] T. Hirofuchi and I. Yamahata. Yabusame: Postcopy live migration for qemu/kvm. In *Oliver & Boyd, London, OCLC*, pages 26–27.

[13] W. Huang, Q. Gao, J. Liu, and D.K. Panda. High performance virtual machine migration with rdma over modern interconnects. In *Proceedings of the 2007 International Conference on Cluster Computing (CLUSTER07))*, pages 11–20, 2007.

[14] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan. Live virtual machine migration with adaptive memory compression. In *Proceedings of the 2009 IEEE International Conference on Cluster Computing (CLUSTER09)*, 2009.

[15] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. Kvm: the linux virtual machine monitor. In *Proceedings of the 2007 Linux Symposium*, volume 1, pages 225–230, 2007.

[16] H.A. Lagar-Cavilla, J.A. Whitney, A.M. Scannell, P. Patchin, S.M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan. Snowflock: Rapid virtual machine cloning for cloud computing. In *Proceedings of the 4th ACM European Conference on Computer Systems (EuroSys09)*, pages 1–12, 2009.

[17] A. Liguori and E.V. Hensbergen. Experiences with content addressable storage and virtual disks. In *Proceedings of the First Workshop on I/O Virtualization (WIOV08))*, 2008.

[18] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu. Live migration of virtual machine based on full system trace and replay. In *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing (HPDC09)*, pages 101–110, 2009.

[19] Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen. Live and incremental whole-system migration of virtual machines using block-bitmap. In *Proceedings of 2008 IEEE International Conference on Cluster Computing (Cluster08)*, pages 99–106, 2008.

[20] D. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou. Process migration. In *ACM Computing Surveys*, volume 32, pages 241–299, 2005.

[21] G. Milos, D.G. Murray, S. Hand, and M. Fetterman. Satori: Enlightened page sharing. In *Proceedings of the 2009 USENIX Annual Technical Conference (USENIX09)*, 2009.

[22] A. Muthitacharoen, B. Chen, and D. Mazieres. A low-bandwidth network file system. In *Proceedings of 18th ACM Symposium on Operating Systems Principles (SOSP18)*, 2001.

[23] M. Nelson, B.H. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *Proceedings of the 2005 USENIX Annual Technical Conference (USENIX05)*, pages 391–394, 2005.

[24] N. Partho, M.A. Kozuch, D.R. O'Hallaron, J. Harkes, M. Satyanarayanan, N. Tolia, and M. Toups. Design tradeoffs in applying content addressable storage to enterprise-scale systems based on virtual machines. In *Proceedings of the 2006 USENIX Annual Techinal Conference (USENIX06))*, pages 1–6, 2006.

[25] S. Rhea, R. Cox, and A. Pesterev. Fast, inexpensive content-addressable storage in foundation. In *Proceedings of the 2008 USENIX Annual Techinal Conference (USENIX08))*, pages 144–156, 2008.

[26] C.P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M.S. Lam, and M. Rosenblum. Optimizing the migration of virtual computers. In *Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI02)*, pages 377–390, 2002.

[27] P. Svard, B. Hudzia, J. Tordsson, and E. Elmroth. Evaluation of delta compression techniques for efficient live migration of large virtual machines. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE11)*, pages 111–120, 2011.

[28] M.M. Theimer, K.A. Lantz, and D.R. Cheriton. Preemptable remote execution facilities for the v-system. In *Proceedings of the 10th ACM Symposium on Operating Systems Principles (SOSP85))*, pages 2–12, 1985.

[29] N. Tolia, T. Bressoud, M. Kozuch, and M. Satyanarayanan. Using content addressing to transfer virtual machine state. In *Technical Report, Intel Corporation*, 2002.

[30] C.A. Waldspurger. Memory resource management in vmware esx server. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI02)*, pages 181–194, 2002.

[31] T. Wood, K. Ramakrishnan, P. Shenoy, and J. van der Merwe. Cloudnet: Dynamic pooling of cloud resouces by live wan migration of virtual machines. In *Proceedings of the 7th international conference on Virtual Execution Environments (VEE09)*, 2011.

[32] T. Wood, G. Tarasuk-Levin, P. Shenoy, P. Desnoyers, E. Cecchet, and M. Corner. Memory buddies: Exploiting page sharing for smart colocation in virtualized data centers. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE09))*, pages 31–40, 2009.

[33] X. Zhang, Z. Huo, J. Ma, and D. Meng. Exploiting data deduplication to accelerate live virtual machine migration. In *Proceedings of the 2010 International Conference on Cluster Computing (CLUSTER10)*, pages 88–96, 2010.