A Quantitative Study of Virtual Machine Live Migration

Wenjin Hu, Andrew Hicks, Long Zhang, Eli M. Dow,
Vinay Soni, Hao Jiang, Ronny Bull, Jeanna N. Matthews
Department of Computer Science
Clarkson University
Potsdam NY13699
{huwj, hicksac, lozhang, dowem, sonivr, hajiang, bullrl, jnm}@clarkson.edu

ABSTRACT

Virtual machine (VM) live migration is a critical feature for managing virtualized environments, enabling dynamic load balancing, consolidation for power management, preparation for planned maintenance, and other management features. However, not all virtual machine live migration is created equal. Variants include memory migration, which relies on shared backend storage between the source and destination of the migration, and storage migration, which migrates storage state as well as memory state. We have developed an automated testing framework that measures important performance characteristics of live migration, including total migration time, the time a VM is unresponsive during migration, and the amount of data transferred over the network during migration. We apply this testing framework and present the results of studying live migration, both memory migration and storage migration, in various virtualization systems including KVM, XenServer, VMware, and Hyper-V. The results provide important data to guide the migration decisions of both system administrators and autonomic cloud management systems.

Categories and Subject Descriptors

D.4.8 [Performance]: Measurements – *benchmark, quantification.*

General Terms

Management, Measurement, Performance, Design, Experimentation.

Keywords

Virtualization, Live migration, Performance, Quantification.

1. INTRODUCTION

Virtualization has played a core role in cloud computing, enabling the portability of workloads from local data centers to managed cloud computing environments, whether public cloud or private cloud. Vendors of data center class virtualization systems like KVM, XenServer, VMware, and Hyper-V, value live migration as an important selling point in their offerings to customers. Virtual machine live migration provides cloud service providers with substantial flexibility in managing their backend infrastructure and has been a substantial area of active research in recent years [15, 16, 17, 18, 19, 20, 21, 22, 23].

Live migration allows a running VM to be moved from one physical host to another. VMs may be moved to better balance the load of current workloads across available servers. VMs may be moved to consolidate VMs onto a smaller number of physical servers, allowing some servers to be shut down completely, thus saving power. VMs may be moved to enable preventative maintenance on some servers or to facilitate the upgrade of some machines.

The high-level goal of live migration is to minimize service disruption, enabling the VM to continue serving requests as it migrates, and hiding the decisions of backend management and placement from users. However, not all virtual machine live migration is created equal. One live migration technology could seek to minimize the time a VM is unresponsive, while another could seek to minimize the end-to-end migration time or the network resources consumed by the migration.

Live migration technologies can vary substantially from vendor to vendor. Variants include memory migration that relies on shared backend storage between the source and destination of the migration, and storage migration that migrates storage state as well as memory state. Migration technologies vary in the degree to which they can maintain on-going network connections during migration, and the requirements placed on the networking infrastructure at source and destination.

Many of the diverse properties of live migration can be unclear without actual hands-on experimentation. We find that the technical descriptions provided by vendors can be incomplete and in some cases inaccurate.

To facilitate hands-on experimentation and testing of live migration on various hypervisors, we have developed an automated testing framework that can be used with any virtualization system. Our test suite measures important performance characteristics of live migration, including total migration time, downtime, or the time a VM is unresponsive to network access during migration, and the amount of data transferred over the network during migration.

We apply this testing framework and present the results of studying live migration, both memory migration and storage migration, in various virtualization systems including KVM, XenServer, VMware, and Hyper-V.

2. BACKGROUND

2.1 What Is Migrated

For all forms of live migration, it is important to consider how migration handles four key aspects: CPU state, memory state, and storage content.

CPU state: The VM's CPU state needs to be context switched from one host to another. This is a small amount of information to transfer and represents a lower bound for minimizing the live migration downtime.

Memory content: The VM's memory state also needs to be transferred to the destination host. This is a larger amount of information than the CPU state. It includes the memory state of both the guest OS and all running processes within the VM. The VM may be configured with more memory than is currently in active use. Hypervisors that can identify and avoid transferring the contents of unused memory have the potential to substantially reduce migration time. Compression and other techniques have the ability to speed up transfer as well.

Storage content: Storage content is an optional part of live migration. The on-disk VM image need not be transferred if it is accessible to both the source and the destination machines through Network Attached Storage (NAS). Transferring only the memory contents is called *memory migration*. If the storage cannot be accessed by the destination host, then a new storage virtual disk needs to be registered on the destination host and the storage content needs to be synchronized from the source to the destination. This is called *storage migration* or sometimes shared-nothing migration. Storage is by far the largest amount of information to be transferred, and the time to transfer the full disk image over the network can be substantial. As with memory, hypervisors that can identify and avoid transferring the contents of unused disk blocks have the potential to greatly reduce migration time.

2.2 Characterization of Memory and Storage Contents to Be Migrated

To help better understand memory migration and storage migration, we present definitions for various categories of memory content and storage content.

- VM Configured Memory

  Configured Memory is the amount of memory that can be given to the virtual machine by the hypervisor. To the VM guest, this looks like the amount of "physical memory" available for use.

- Hypervisor Allocated Memory

  Allocated Memory is the amount of physical memory on the underlying hardware that the hypervisor has actually allocated to a guest VM. This is less than the VM Configured Memory and is reported from the perspective of the hypervisor. Allocated Memory indicates the portion of VM Configured Memory that the VM has actively used. If a VM uses memory and then frees it, the allocated memory may or may not be reduced from the perspective of the hypervisor.

- VM Used Memory

  Used Memory is the memory currently and actively used by VM's OS and all running processes. These are memory pages actively used by the VM's OS and resident inside the VM memory. It is reported from the perspective of the guest VM.

- Application Requested Memory

  Requested Memory is the memory that applications running inside VM have requested from the VM's OS. The requested memory is not guaranteed to be resident in memory because memory not currently in use may be swapped out to the VM's disk when VM Configured Memory is all used.

- Application Actively Dirtied Memory

  Dirtied Memory is the memory that applications are actively modifying via writing to in-memory pages. Dirtied Memory is part of the Requested Memory of an application that is actively being used thus it is typically resident in memory rather than being swapped out to disk.

Normally, those memories sizes can be ordered as below: VM Configured Memory $\geq$ Hypervisor Allocated Memory $\geq$ VM Used Memory $\geq$ Application Requested Memory $\geq$ Application Actively Dirtied Memory. Their relationship is illustrated in Figure 1. If the VM's swap disk is actively in use, then it is also possible to see Application Requested Memory > VM Used Memory because part of the requested memory pages go into swap instead of being resident in memory.

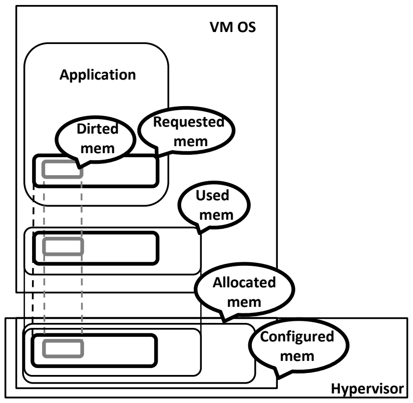

Figure 1 – Illustrating the categories of memory content

Practically in live migration, Hypervisor Allocated Memory is the most accurate parameter to estimate the amount of memory data to be transferred. Configured Memory serves as an upper bound, but it is a less accurate guide when predicting migration time. Dirtied Memory is also a key parameter. When memory is dirtied after it is migrated, a new copy must be sent, increasing the total amount of data to be transferred over the network.

We also break down the various categories of storage content to be considered in migration:

- Virtual Disk Size

  Virtual Disk size is the size of the "physical" disk offered to the VM for its use. Like VM Configured Memory, virtual disk size is typically defined when the VM is created. To the VM guest, this looks like the size of its "physical disk". Most hypervisors offer options to allocate all the space when created, or to dynamically expand the actual storage space as it is used.

- VM Used Blocks

  Used Blocks size is the actual system and user data size stored in VM image and used by VM's OS. It is the size of data actually contained in the VM's file system. Typically, the Virtual Disk will not be completely filled with data.

- Hypervisor Allocated Blocks

  Allocated Blocks is the space actually allocated by the hypervisor for a VM's storage content. If the hypervisor pre-allocates the entire virtual disk size initially, then Allocated Blocks will be same as the Virtual Disk size. If the hypervisor allocates space as it is used, then the Allocated Blocks may be the same size as the Used Blocks. However, if a VM frees the blocks it is using, then the hypervisor may not shrink the Allocated Blocks. The file system inside the VM understands which disk blocks are used and which ones are freed, but it is harder for the hypervisor to have this level of visibility. Avoiding transferring unused or garbage-collected blocks could substantially reduce storage migration time in some case, but we are not aware of any hypervisor that has implemented a disk space garbage collection mechanism.

In general, the disk size follows the order: Virtual Disk Size $\geq$ Allocated Blocks $\geq$ VM Used Blocks.

In the result and analysis section, we will match these definitions of memory and storage contents to the data we have collected and give further analysis and explanation.

3. DESIGN OF BENCHMARK

In this section, we describe the design of an automated testing framework that measures important performance characteristics of live migration, including total migration time, downtime or the time a VM is unresponsive during migration, and the amount of data transferred over the network during migration. Our testing framework will allow cloud administrators to compare the pros and cons of the live migration feature across different hypervisors, and to gather key data to guide the migration decisions of both system administrators and autonomic cloud systems.

Our primary goal is to provide datacenter administrators with clear information about the factors that affect live migration, and to point out common pitfalls of live migration such as migrations that do not complete. Test results can also give hypervisor developers an objective view of how to optimize their live migration and provide insight into how their migration implementation compares to those of other hypervisors.

3.1 Test Architecture for Live Migration

To make our testing more objective and fair to each different virtualization, we set up a 3rd-party benchmark server to evaluate all virtualization platforms in a consistent manner. All the test actions are taken either in a benchmark server or benchmark VM, independent of the underlying hypervisors or host platforms.

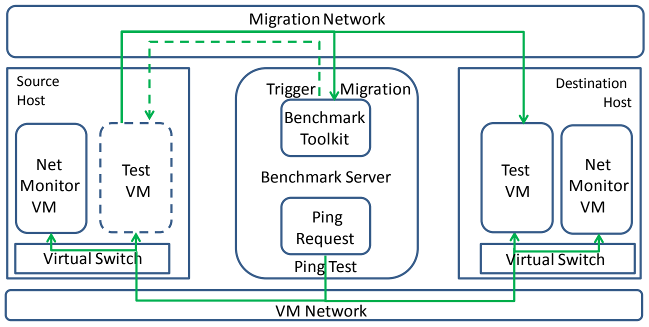

Figure 2 – Live migration test framework

The more detailed test framework is shown in Figure 2. Each of the source and destination hosts was configured identically, with two distinct physical network adapters, cabled through independent identical switches.

There are two separate networks set up to isolate migration traffic from VM web service traffic. The VM web service goes through one network while the VM migrating data goes through a separate one. A ping test is used to monitor the accessibility of services within the migrating benchmark VM. All cloud management traffic, including the actual migration traffic, goes through a different network segment than the ping probes. The migration command is sent from the benchmark server to the source host over the network.

The benchmark can capture traces of the network traffic to and from the VM as well as to and from the hypervisor, including the migration traffic. Those captured packets with timestamps can be further used to probe the exact behavior of live migration. We have used this to build a live migration's ping graph and probed into what happens to the VM packets before and after the migration downtime.

3.2 Workload Stress Test in VM

Since VM migration performance varies with the workload running inside the migrating VM, our benchmark suite includes a set of workload stress tests that run inside the migrating VM. These tests allow us to probe into the relationship of VM activities and live migration performance. The stress tests are classified by the primary resource consumed in the test: CPU, memory access (read and write), disk access (read and write), and network access (send and receive from both TCP and UDP). While tuning the stress test inside VM and evaluating live migration performance, we would be able to observe the pattern of how exactly each activity in VM will affect the VM migration.

The *CPU Stress Test* performs intensive mathematical calculations (FFT), but uses a trivial amount of memory and disk I/O. It can be configured to use different scheduling priorities ranging from 1 (75% user CPU cycles) to 10 (100% system CPU cycles).

The *Memory Stress Test* is programmed to allocate a specified memory size and then dirty a specified portion of the total allocated memory. We can control both the amount of memory that is written or dirtied as well as the rate at which data is dirtied.

The *Disk Write Stress Test* is programmed to keep writing page units sequentially into a file to a specified size, and then rewrite new data from the beginning of the file.

The *Disk Read Stress Test* is programmed to first create a file with specified data, and then keep reading data sequentially by page unit from file into memory sequentially, and then iteratively read from the beginning of the file again.

The *TCP Send/Receive Stress Test* is programmed to intensively send sequence data through an established TCP connection. The client will wait to receive and examine the sequence numbers. If the TCP connection is broken due to the migration, the TCP connection will be reestablished and reset the sequence number. The TCP connection will consume the VM's Internet bandwidth, but won't affect the migration network bandwidth.

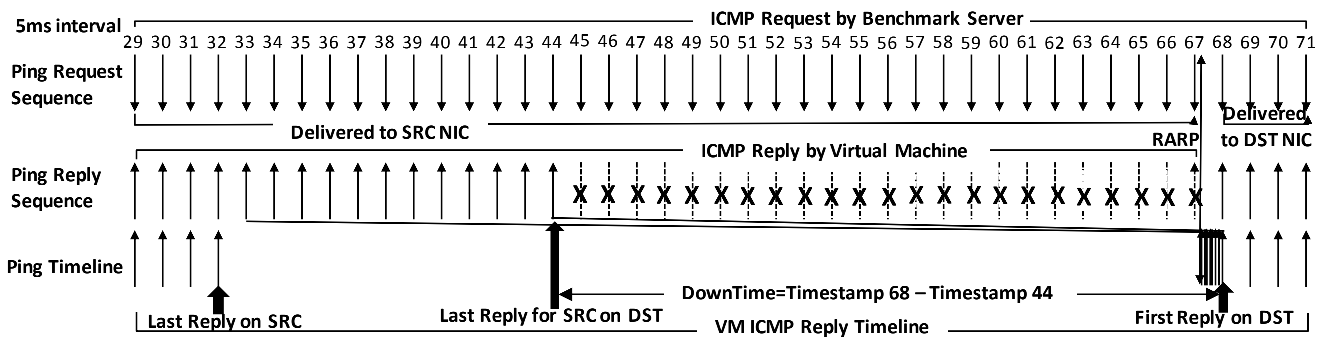

Figure 3 – VMware downtime ping pattern

The *UDP Send/Receive Stress Test* is programmed to intensively send sequence data through UDP protocol with no guarantee of it being received. The client will receive and examine the sequence numbers. If some of the UDP packets are lost, the UDP client will receive the discontinued sequence numbers and record them to a file with their timestamps.

3.3 Profiling Live Migration

We collected three primary statistics on each migration, specifically how long it takes to complete the migration from start to finish (total migration time), how much time the VM in unresponsive during migration (VM downtime), and how much data is transferred over the network.

Total time is relatively simple to measure. We simply initiate each migration using a command line utility and note the time before and after the migration command completes.

Measuring VM downtime is more complicated. We ping the VM during migration, controlling the interval of the pings to pinpoint the downtime experienced during migration. The ping interval is specifically configured to be at millisecond level, much smaller than for normal ping. As expected, a contiguous block of ping probes will receive no response from the migrating VM. We note the timestamp and sequence number of ping requests that receive no response in order to compute the VM downtime. From our test framework, we can build the ping sequence diagram. The ping pattern during VMware memory migration is shown in Figure 3 as one example. We see that some ping packets are received at the source but the corresponding responses are sent from the destination (e.g. sequence numbers 33-44). The detailed pattern of which ping requests are responded to at the source, which are responded to at the destination, and which receive no response at all, is quite interesting and subtle. For example, we see evidence that even after the VM stops responding to ping requests on the source, it is still receiving and buffering them for later responses on the destination. The responses to these pings are delayed substantially and tend to be issued in a cluster at the destination. We measure the total data transferred using counters on the managed switch between the source and destination machines; our benchmarking scripts capture the send and receive counts on the switch before and after migration through SNMP protocol. We also see that all the hypervisors use an ARP packet to notify and redirect the packets to the new host. Note that we isolate the migration traffic by sending it over a different network path than any management or measurement traffic.

3.4 Benchmark Toolkit

A shell script is used to initiate the workload in the VM, to collect statistics to profile VM before, during and after the migration, to start the migration and finally to cleanup/reset the system. The pseudo code for measuring VM migration time and other data is displayed in Table 1.

We run each migration twice – once without ping measurements and once with ping measurements so that we can account for any difference in results as a result of the additional VM activity due to ping response. The ping script will set the ping interval of 1 ms, dropping the ping test inside the VM before live migration starts, and then the ping responses are recorded into a file. Once migration is done, we analyze the ping responses and pick out the lost sequence numbers and then calculate the downtime by the lost sequence timestamps. So for evaluating migration downtime, line (2) is replaced with ping scripts, including the migration commands. For each virtualization, the code line (1) (2) (3) will be replaced with the corresponding hypervisor command interface as shown in Table 2.

Table 1. Pseudo code for live migration evaluation

```
Power On VM from Hypervisor                          (1)
        Start workload in VM
        Record VM CPU, Memory, Disk stats
        Record Start Time
        Record Initial network switch registers
                Start to migrate VM                  (2)
        Record End Time
        Record Final network switch registers
        Calculate live migration stats
                Migrate VM back                      (3)
Power Off VM
```

Table 2. Virtualization live migration command API

Virtualization	**Command Interface**
KVM	ssh + virsh
VMware	Java web service SDK
Hyper-V	telnet + powershell
XenServer	ssh + xe

4. RESULTS

In this section, we present results from testing live migration in Redhat KVM 3.2, Citrix XenServer 6.1, VMware vSphere 5.1, and Microsoft Hyper-V 2012. [1]

4.1 Testbed Specifications

In our experiment, VMs are migrated between two enterprise servers as described in Table 3. The full test environment consists of two servers, one network attached storage, and a managed switch.

Table 3. Server specification

PC Model	HP DL165 G7
CPU	2x AMD Opteron 6220 8 core 3.0GHz
Memory	64GB DDR3 1600MHz
NIC	1 Gb/s

The NAS is Iomega px4-300r 7200 rpm with 12 TB storage configured in a RAID mirror with capacity of 6 TB. It is used as a shared NFS server to host the VM images. The managed switch is a 1 Gb/s Netgear ProSafe 108T.

We use a separate benchmark server for starting the live migrations and collecting the data. The benchmark server is an IBM xSeries 2X Intel Xeon 3.2 GHZ machine with 2 GB of memory and 1 Gb/s NIC. We run Ubuntu 12.04 server on this benchmark server.

The VM to be migrated has 1 vCPU, 2 GB of memory, 40 GB of storage and 1 virtual network interface in bridge mode. We run Ubuntu 12.04 64 bit in the migrating VM.

Table 4. VM image specification

	Memory migration media	**Memory migration image**	**Storage migration media**	**Storage migration image**
KVM	NFS	QCOW2	EXT4	QCOW2
VMware	NFS	VMDK	VMFS	VMDK
Hyper-V	iSCSI	VHD	NTFS	VHD
XenServer	NFS	VHD	EXT3	LVM

The format of VM image varies from hypervisor to hypervisor. Each virtualization supports its own image format. In our tests, we use the format that each virtualization favors as described in Table 4.

For memory migration, the VM images are generally hosted on an NFS share exported by the NAS. In the case of Hyper-V, we also collected results with the NAS exporting the VM image over iSCSI. We did this because Hyper-V does not officially support NFS.

For storage migration, the VM image is stored on a disk local to each server. For each Hypervisor, the file system format for the local disk varies. KVM is EXT4, VMware is VMFS, Hyper-V is NTFS, XenServer is an LVM volume. In each case, we used the format preferred by the hypervisor under test.

4.2 Results and Analysis

In this section, we present results from using our full test suite with a wide variety of parameters.

4.2.1 Live migration baseline

First, we present some baseline measurements for memory migration across the four hypervisors. For each virtualization, we ran one VM on a host as specified above, with no stress tests running inside, then collected the data while migrating it to the destination.

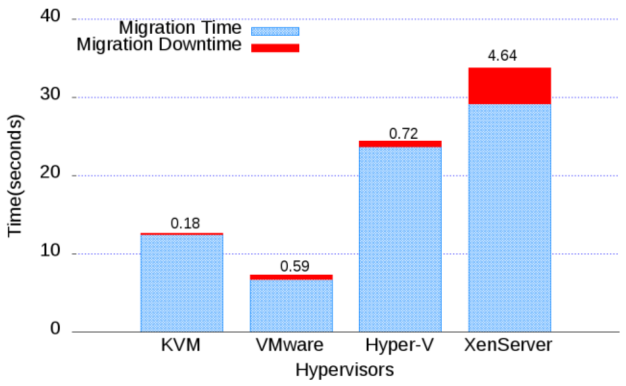

Figure 4 – Memory migration baseline

Figure 4 portrays the total live migration time broken down into downtime and migration time. Downtime represents the time that the VM being migrated was unresponsive to ping requests. Table 5 shows more detailed results for this baseline.

Table 5. Memory migration baseline

	Down Time (sec)	**Migration Time (sec)**	**VM Used Memory (MB)**	**Migration Data (MB)**
KVM	0.18	12.45	124	247
VMware	0.59	6.71	234	354
Hyper-V	0.72	23.71	135	2266
XenServer	4.64	29.17	85	2255

From these results, we see that VMware has the lowest total migration time, while KVM has the lowest downtime. XenServer has the worst total time as well as the worst downtime. It is the only virtualization system showing downtime of more than 1 second. XenServer's downtime also matches with other

[1] For memory migration, both Hyper-V and VMware require the source and destination hosts to be in the same virtual cluster. Hyper-V also requires that VM images be hosted on Samba 3.0 or iSCSI media, but those limitations are removed for storage migration.

For storage migration, KVM and Hyper-V handle the complete migration in one step while VMware and Citrix can handle the migration in two separate steps: disk migration and memory migration. This allows VMware and Citrix to keep the VM running on the same host, but migrate the VM's disk storage to a new place, separating disk migration from memory migration.

researcher's result [10]. Comparing the migration data with migration time, their estimate transfer speeds are 20 MB/s (KVM), 41 MB/s (VMware), 96 MB/s (Hyper-V), and 62 MB/s (XenServer), which indicates that Hyper-V has the best network throughput while still maintaining similar downtime as VMware. KVM successfully syncs the dirty memory data to achieve the minimal downtime. Notice that VM used memory varies between hypervisors even when the same version of Ubuntu is installed.

For cloud administrators, the commercial advertisements of only milliseconds of downtime are not always true. When we migrate VMs, we still need to evaluate how sensitive the workload is to downtime. A downtime of 4 seconds could easily cause service interruptions for some workloads.

Referring back to the categories of memory content that we defined in Section 2.2, KVM and VMware take less migration time because they transfer only Allocated Memory rather than Configured memory, while Hyper-V and XenServer migrate the whole Configured Memory even though they use the least memory.
The take-away for virtualization developers is that live migration only needs to migrate the in-used memory which will save a lot of migrating time, but the hypervisor has to abstract a VM's free memory information from the VM memory. An agent inside VM can be used to aid hypervisors in this [15].

4.2.2 Memory migration vs. Storage migration

In this section, we added measurements of storage migration in Figure 5 to the memory migration baselines in the previous section. Not surprisingly, the total migration time increases dramatically, but it is interesting to note that the downtime does not increase substantially relative to memory migration. A few simple back-of-the-envelope calculations on the size of the virtual disk and the network bandwidth can be used to illustrate the point. Transferring a 40 GB virtual disk over a 1 Gb/s network will take at least 8 minutes, while transferring a 1 TB virtual disk over a 1 Mb/s link could take over 2 weeks! Storage migration in a production environment could take longer as it experiences more contention for available network resources. Similarly, the longer migration takes, the more blocks will be rewritten during the time it takes to transfer other blocks and thus need to be retransmitted. Migrating cold storage blocks first, rather than transferring blocks in physical order could be a good strategy for minimizing this effect.

For cloud administrators, it can be tempting to offer storage migration to customers with large disk space, but it is important to recognize how time-consuming storage migration can be. One best practice is to allocate a small system disk for VM and mount large user data from shared storage.

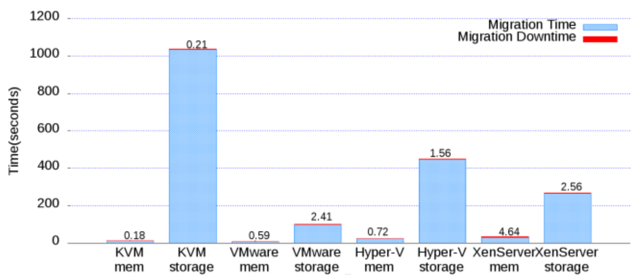

Figure 5 – Memory migration vs. storage migration on different virtualizations

The take-away for virtualization developers is that most storage migration time is spent moving static storage data to the disk. Since backups or snapshot VM images are a common routine for most administrators, it will save a dramatic amount of time if virtualization can support delta-migration where the remote host already has a snapshot image or recent "stale" image, so that the hypervisor only need to migrate the delta or new data, and then merge that with the snapshot to bring it up-to-date.

Notice also that KVM has dramatically higher total migration time. Regardless of how full the virtual disk is with valid data, KVM will transfer the entire Virtual Disk, rather than only the Used Blocks or Allocated Blocks. This would be an important optimization for KVM. It can save cloud administrators gigabytes of data and hours of time over the network.

4.2.3 How VM Stress activity impacts live migration

In this section, we consider the impact of an activity running inside the VM during migration. If the VM is running with a lot of activities going on inside, how badly will it prolong the migration? How much will the migration impact the services inside?

We used our benchmark to run a series of stress tests in VM and evaluated the live migration performance. We determined each resource's impact and showed how live migration varies.

The CPU's activity's impact is trivial. Our result shows that no matter how intensive the CPU workload is in the VM, when the CPU state is suspended and migrates, the migration time and downtime are almost the same. Live migration is believed to be independent of CPU workloads.

Not surprisingly, memory writes impact memory migration time substantially while memory reads do not. For memory writes, both the total amount of memory written (dirty memory size) and the rate with which memory is written matter.

Disk writes have limited impact on memory migration. But somewhat surprisingly, disk reads affect memory migration substantially because a disk read typically translates into a memory write as the data read from disk is cached in memory. Disk writes don't have the same impact on memory migration, but of course, disk writes affect storage migration while disk reads do not.

Memory content synchronization is the most critical part. Here we use our benchmark to vary the dirty memory size inside the migrating VM to further our exploration in live migration.

We dirty each memory page at the most intensive rate, and then gradually increase the dirty memory size exponentially up to the configured memory size. As the dirty memory size goes up, the migration time, migration data, and downtime responds as below in Figure 6, 7, and 8.

4.2.3.1 Dirty Memory size vs. Memory migration

As the dirty size increases, the used memory and dirty data required to be synced increase. So for virtualizations, this is a general principle for the live migration: the dirty memory size increases, the migration time and data also increase as shown in Figures 6 and 7.

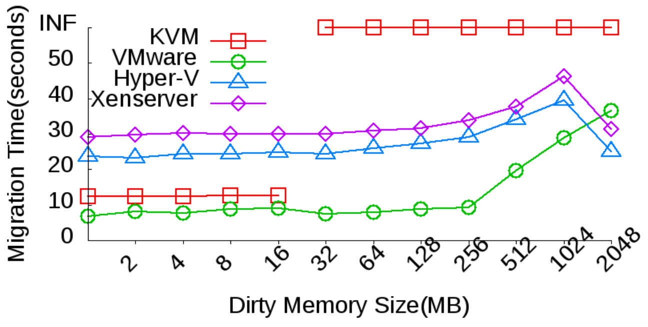

Figure 6– Memory migration time comparison

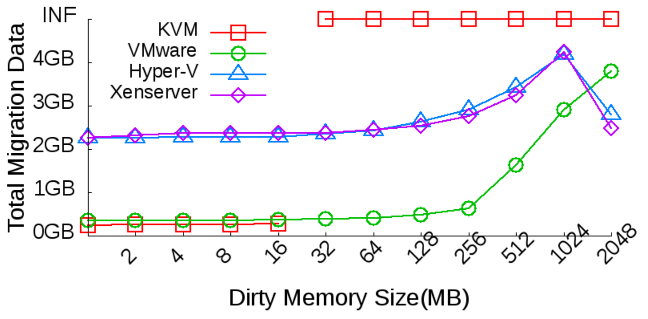

Figure 7 – Memory migration total transferred data comparison

Surprisingly, KVM is unable to finish when the dirty memory size increases beyond 32 MB at least with default migration settings. Recall from our baseline measurements, that KVM achieved the lowest downtime. KVM sacrifices the migration network throughput and migration time to maintain this low downtime. As the dirty memory size increases, KVM finally cannot keep pace with synchronizing the large dirty memory at high speed and thus fails to complete the migration. It also fails to detect that no progress is being made and change its strategy.

For cloud administrators, it is better to be aware of this situation and add a timeout option to the VM migration for safety. If a migration does not finish in a fixed time, it would be better to timeout and initiate a cold migration of the VM state than to continue attempting live migration unsuccessfully [11].

For all systems, there is a tradeoff between minimizing total migration time and downtime. Syncing the dirty pages fast minimizes the downtime, but will generate more migration data and prolong the migration time, while syncing lazily causes more dirty memory pages to be migrated during the migration downtime and will introduce more downtime. An optional approach may be for the hypervisor to slow down the VM dirty speed while migrating the VM [12].

Another interesting result is seen as the dirty size becomes the same as the configured memory size and swap the space is used to load excessive memory pages onto virtual disk. For XenServer and Hyper-V, the migration time and data drop dramatically, instead of continuing to increase, opposite of the VMware pattern. Swap does not impact VMware but does help Hyper-V and XenServer to reduce the dirty memory impact. One hypothesis is that so many pages have been dirtied in a short amount of time that even recently dirtied pages fall out of the least recently used heuristics and are therefore swapped out to disk. This saves the sync time for the memory since those dirty memory pages are on the disk and thus shared rather than being transferred during memory migration.

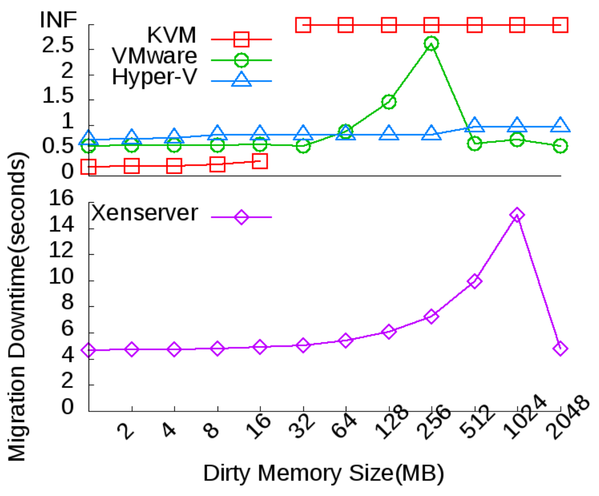

Figure 8 – Memory migration downtime comparison

VMware's default setting is hosting the VM swap with the local host, separate from VM images. So the swap also needs to be synchronized to the destination. VMware does offer an option to place the swap within the VM folder, which can facilitate the migration, but it is not set as a default.

In Figure 8, we again see that KVM fails to complete migration above 32 MB of Dirty Memory Size. Hyper-V's downtime increases, but only slightly from 0.72s to 0.96s. VMware's downtime can be considered as independent from dirty memory size variations, remaining at a constant 0.61s if we don't take into account the downtime spike introduced around 256 MB/s dirty memory size. This spike is yet unknown to us without the insight of a VMware migration algorithm, but it is reliably and consistently reproduced in our testbed. The dirty memory size becomes a large factor in XenServer's downtime (isolated in its own figure so the comparison of the other 3 is more clear). The downtime for XenServer goes up to over 15 seconds, which is not acceptable for most web service applications.

4.2.3.2 Dirty memory size vs. Storage migration

Applying the same tests to storage migration in the same testbed, we see the difference between memory migration and storage migration under the parameter of dirty memory size in Figure 9, 10, and 11.

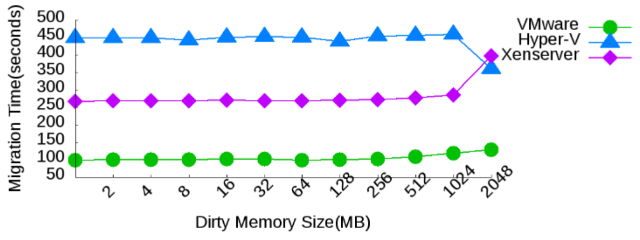

Figure 9 – Storage migration time comparison

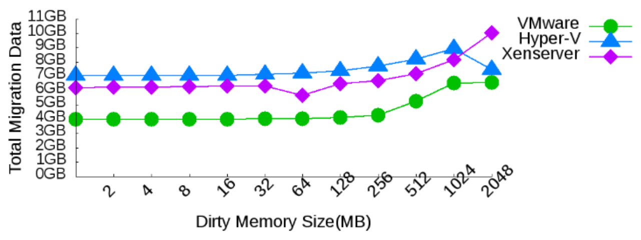

Figure 10 – Storage migration total transferred data comparison

KVM's storage migration is still unsuccessful when the dirty rate is above 32 MB/s dirty memory size, the same as memory migration. It is natural to conclude that KVM storage migration is impacted by the same memory content synchronization failure. We removed the KVM data in the figures below to provide a better visual comparison of the other virtualization systems.

In general, the trend of storage migration time and data follows the same pattern of memory migration as the dirty memory size changes, but takes more time and data to transfer storage content since little disk I/O is involved. The line level is differentiable due to a difference in VM base image size. VMware's VM size is 2.2 GB, while Hyper-V's is 3.5 GB, and XenServer's is 1.2 GB.

Different behavior happens when swap space is being accessed and the dirty memory size equals the configured memory size. As we hypothesized, the drop in memory migration is due to disk I/O. In XenServer's storage migration, those dirtied and swapped to the disk also need to be synced, so the storage migration time and data increase. We hypothesized that XenServer's storage sync is in asynchronous mode while Hyper-V syncs disk I/O simultaneously to the source and destination hosts.

One take-away message for virtualization developers is that if a hypervisor can have access to the destination's local storage remotely at the same time, then storage migration will be more flexible in handling the dirty synchronization issue since storage requests can tolerate longer latency than memory requests.

The downtime that storage migration introduces is somewhat different than for memory migration as shown in Table 6.

Table 6. Downtime baseline for memory vs. storage migration

	Memory Downtime (sec)	Storage Downtime (sec)
KVM	0.18	0.21
VMware	0.59	2.41
Hyper-V	0.72	1.56
XenServer	4.64	2.56

For VMware and Hyper-V, VM's storage switch during storage migration does introduce a lot more downtime. KVM retains its low downtime; XenServer has less downtime in storage migration than memory migration. We suspect that XenServer's storage migration uses a slightly different algorithm from its memory migration.

Furthermore, when we vary the dirty memory size, we can see the storage migration downtime pattern in the figure below. In general, the memory migration and storage migration downtime patterns affected by dirty memory follow a similar pattern.

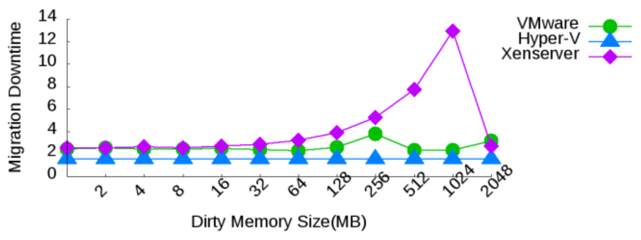

Figure 11 – Storage migration downtime comparison

Obviously, dirty memory size is a big factor that dramatically impacts XenServer migration downtime. However, swap content syncing won't impact the downtime that much, since downtime still drops when swap is in use.

Although Hyper-V's downtime during memory migration is gradually increasing, storage migration downtime is independent of dirty memory size.

The downtime spike still exists in VMware storage migration. Additionally, VMware storage migration introduces two phases of downtime: the first is 0.11s and the second is around 2.3s. This hasn't been observed in other hypervisors. Moreover, contrary to XenServer, disk I/O introduces more downtime to storage migration as swap space is used by the VM.

Through the migration statistics we collected, we have found each virtualization has the pros and cons. There are also a lot of optimizing points that could be discovered by using our benchmark tool. In general, for live migration implementation, the more information a hypervisor can obtain from the VM's OS during live migration, the more efficient and faster the hypervisors can accomplish live migration, such as how many memory pages are in use, or how many disk blocks are allocated with user data.

4.2.4 Live migration vs. Cold migration

Finally, we compared live migration to cold migration. In cold migration, the VM is shut down and its files transferred from source to destination. For cold migration, downtime is the same as total migration time.

We created a KVM VM image as Section 4.1 specified, the VM virtual disk size is capped as 40 GB (Configured when created), but only used 1.8 GB of space (*df* command inside VM). The image file size shows as 41 GB (*ls* command in Hypervisor). After we compressed the image with *gzip*, the compressed size turned out to be 570 MB. As cold migration, we used *scp* to copy the image over to the destination host. As storage live migration, we migrated the running VM with VM image to the destination. The actual measured network bandwidth for migration is 97 Mbps.

In Figure 12, one can see that live storage migration takes the longest transfer time and trivial downtime. To transfer the cold VM image itself it will take a fair amount of time, which is indicated by cold migration. The compression bar tells us that the high compression rate of *gzip* can greatly reduce the transfer time, but the time for compression/decompression must be added to the downtime time as well. The savings from transfer time can be completely eaten up by the compression and decompression overhead. Of course this depends on the network bandwidth available. Novel compression schemes could also be explored that are customized for the live migration scenario [13].

Although live migration takes the longest transfer time, the downtime is trivial, and the transfer time is still affordable compared to cold migration. It leads the Cloud administrators to realize that they need to care about live migration's performance and understand how to best utilize the feature in cloud management.

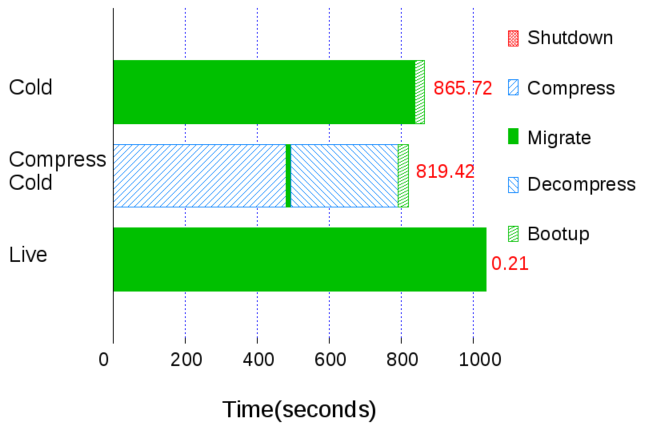

Figure 12 – Live migration vs. cold migration

5. LESSONS LEARNED

We have the live migration benchmark tools automated and it can be easily ported to test any kind of enterprise-level virtualization system and automatically finish all the stress tests. In the process of building our tools, we learned several other interesting lessons.

For example, we found that ping in Ubuntu has a new feature of ping at a high speed up to 1ms, which can better evaluate the downtime. But not all the Linux-based systems support this feature, not to mention Windows Hyper-V. That brings us to our plan that our benchmark tool should be independent of virtualization systems, and we developed a 3rd–party benchmark server that is responsible for all the tests. This leads to another problem of how to remotely manage the VMs. We use various remote APIs provided by different hypervisors.

Initially we hoped to tune all of our network parameters using the powerful network simulator, CORE [14]. CORE simulates network topology, variations in network speed, and outage situations. However, it turned out that software simulation cannot reach the optimal network performance and it may also drop the packets. We also attempted to use software to record the packets to network traffic, but all failed due to the fact that collecting network traffic is resource consuming. To solve this problem, we were required to use hardware support to record network traffic. We used a Netgear managed switch to count the data packets accurately and efficiently and can optionally augment these statistics with packet captures to help drill down into the details. For example, we can determine what happens to each ping packet during the migration process.

6. RELATED WORK

The closest related work is the Virt-LM [4] benchmark from Zhejiang University. It measures many of the same metrics including total time, downtime and includes the ability to sample the migration rate at various points during the migration. In contrast to their work, which utilizes the SPEC simulated workloads, our work uses microbenchmarks run in the VM to precisely control the workload parameters such as the amount of memory allocated, read and written, and the amount of disk space allocated, read and written. Furthermore, we specifically applied our benchmarks to studying both memory migration and storage migration. We also studied KVM, XenServer, VMware, and Hyper-V where their study examined only KVM and Xen. We found that including a Windows based hypervisor, Hyper-V, was essential to making our benchmark truly hypervisor agnostic.

Vmmark [5] benchmark is dedicated to the VMware platform with coarse metrics while our benchmark is compatible with diverse virtualization platforms and fine-granularity metrics of live migration performance. Other papers either focus on the migration cost model [6], simulating the real web service workload and evaluating by SLA criteria or math modeling [7], or dividing the migration time into separate phases and then formalize with mathematical models. We focus on the impact on live migration by the VM internal services, and the results could help further refine their migration model. Some studies are targeting the energy efficiency [8] of live migrations. Our testing approach from the angle of VM activities is similar to [9], but they are focusing more on correlating the migration time with the hypervisor's CPU activities, while our benchmark correlates VM internal resource activities with live migration performance. Additionally our benchmark provides finer-granularity tuning parameters to those resource activities. There have been numerous migration evaluations published as either whitepapers from virtualization vendors or community blog posts [1] [2] [3]. However, none of these have provided a thorough and consistent comparison across virtualization platforms.

There is also a rich set of research into optimizing live migration performance. Some customize specific virtualization systems to prototype their ideas [15, 16] or validate their ideas on multiple platforms [17]. They all use migration time, total migration data, and migration downtime to validate their enhancement. Our live migration benchmark could be applied to help quantify the impact of their changes on live migration performance and give an apples-to-apples comparison among all those optimizations.

7. SUMMARY

Live migration is an important feature in cloud management. Our live migration benchmark has helped us to examine contemporary enterprise-level virtualization and their live migration features. By quantifying the live migration characteristics, we have been able to find the gaps between ideal performance and actual performance under a wide variety of conditions. This can help cloud administrators more accurately predict live migration performance and understand better how to utilize live migration in their cloud management strategy. By comparing across different hypervisors, we also find there are some optimization spaces for virtualization developers to improve live migration performance. Our VM live migration benchmark is licensed under GPL3.0 and can be accessed through the link http://www.clarkson.edu/projects/gdc/vmmigration.

8. ACKNOWLEDGMENTS

We would like to thank VMware and Citrix for generously providing us enterprise licenses for our testing. We also thank NYSERDA for providing the hardware equipment and Clarkson Green DataCenter project for supporting this subproject.

9. REFERENCES

[1] Virtual Machine Migration Comparison: VMware vSphere vs Microsoft Hyper-V. 2011. A principled technologies test report, commissioned by VMware Inc. Principled Technologies Inc.

[2] Evaluating Microsoft Hyper-V Live Migration Performance Using IBM System x3650 M3 and IBM Systems Storage DS34002010. 2010. Whitepaper. IBM.

[3] Why Choose VMware for Server Virtualization? A Comparative Analysis for New Virtualization Customers. 2012. Whitepaper. VMware Inc.

[4] Dawei Huang, Deshi Ye, Qinming He, Jianhai Chen, and Kejiang Ye. 2011. Virt-LM: a benchmark for live migration of virtual machine. In *Proceedings of the 2nd ACM/SPEC International Conference on Performance engineering* (ICPE '11). ACM, New York, NY, USA, 307-316.

[5] Vikram Makhija, Bruce Herndon, Paula Smith, Lisa Roderick, Eric Zamost, and Jennifer Anderson. 2006. VMmark: A scalable benchmark for virtualized systems. Technical Report. VMware Inc.

[6] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. 2009. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation. In *Proceedings of the 1st International Conference on Cloud Computing* (CloudCom '09), Martin Gilje Jaatun, Gansen Zhao, and Chunming Rong (Eds.). Springer-Verlag, Berlin, Heidelberg, 254-265.

[7] Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. 2011. Power Consumption of Virtual Machine Live Migration in Clouds. In *Proceedings of the 2011 Third International Conference on Communications and Mobile Computing* (CMC '11). IEEE Computer Society, Washington, DC, USA, 122-125.

[8] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. 2010. Predicting the Performance of Virtual Machine Migration. In *Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (MASCOTS '10). IEEE Computer Society, Washington, DC, USA, 37-46.

[9] Yangyang Wu and Ming Zhao. 2011. Performance Modeling of Virtual Machine Live Migration. In *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing* (CLOUD '11). IEEE Computer Society, Washington, DC, USA, 492-499.

[10] Felix Salfner, Peter Troger, and Andreas Polze. 2011. Downtime Analysis of Virtual Machine Live Migration. In *Proceedings of the Fourth International Conference on Dependability* (DEPEND 2011). IARIA, Nice, France, 100–105.

[11] Virsh migrate man page, http://linux.die.net/man/1/virsh.

[12] Zhaobin Liu, Wenyu Qu, Weijiang Liu, and Keqiu Li. 2010. Xen Live Migration with Slowdown Scheduling Algorithm. In *Proceedings of the 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies* (PDCAT '10). IEEE Computer Society, Washington, DC, USA, 215-221.

[13] [1] Hai Hui, Li Deng, Song Wu, Xuanhua Shi, and Xiaodong Pan. 2009. Live virtual machine migration with adaptive memory compression. In *Proceedings of Cluster Computing and Workshops* (CLUSTER '09). IEEE Computer Society, Washington, DC, USA, 1-10.Jeff Ahrenholz. 2010. Comparison of CORE Network Emulation Platforms. In *Proceedings of IEEE MILCOM Conference* (MILCOM 2010). IEEE Communications Society, New York, NY, USA, 864-869.

[14] Jui-Hao Chiang, Han-Lin Li, and Tzi-cker Chiueh. 2013. Introspection-based memory de-duplication and migration. In *Proceedings of the 9th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (VEE '13). ACM, New York, NY, USA, 51-62.

[16] Changyeon Jo, Erik Gustafsson, Jeongseok Son, and Bernhard Egger. 2013. Efficient live migration of virtual machines using shared storage. In *Proceedings of the 9th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (VEE '13). ACM, New York, NY, USA, 41-50.

[17] Xiang Song, Jicheng Shi, Ran Liu, Jian Yang, and Haibo Chen. 2013. Parallelizing live migration of virtual machines. In *Proceedings of the 9th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (VEE '13). ACM, New York, NY, USA, 85-96.

[18] Lei Cui, Jianxin Li, Bo Li, Jinpeng Huai, Chunming Hu, Tianyu Wo, Hussain Al-Aqrabi, and Lu Liu. 2013. VMScatter: migrate virtual machines to many hosts. In *Proceedings of the 9th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (VEE '13). ACM, New York, NY, USA, 63-72.

[19] Jie Zheng, Tze Sing Eugene Ng, and Kunwadee Sripanidkulchai. 2011. Workload-aware live storage migration for clouds. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (VEE '11). ACM, New York, NY, USA, 133-144.

[20] Petter Svärd, Benoit Hudzia, Johan Tordsson, and Erik Elmroth. 2011. 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* (VEE '11). ACM, New York, NY, USA, 111-120.

[21] Xiang Zhang, Zhigang Huo, Jie Ma, and Dan Meng. 2010. Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration. In *Proceedings of the 2010 IEEE International Conference on Cluster Computing* (CLUSTER '10). IEEE Computer Society, Washington, DC, USA, 88-96.

[22] Katharina Haselhorst, Matthias Schmidt, Roland Schwarzkopf, Niels Fallenbeck, and Bernd Freisleben. 2011. Efficient Storage Synchronization for Live Migration in Cloud Infrastructures. In *Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing* (PDP '11). IEEE Computer Society, Washington, DC, USA, 511-518.

[23] Yanqing Ma, Hongbo Wang, Jiankang Dong, Yangyang Li, and Shiduan Cheng. 2012. ME2: Efficient Live Migration of Virtual Machine with Memory Exploration and Encoding. In *Proceedings of the 2012 IEEE International Conference on Cluster Computing* (CLUSTER '12). IEEE Computer Society, Washington, DC, USA, 610-613.