Virt-LM: A Benchmark for Live Migration of Virtual Machine*

Dawei Huang, Deshi Ye, Qinming He, Jianhai Chen, Kejiang Ye
College of Computer Science, Zhejiang University
Hangzhou, 310027, China
{tossboyhdw,yedeshi,hqm,chenjh919,yekejiang}@zju.edu.cn

ABSTRACT

Virtualization technology has been widely applied in data centers and IT infrastructures, with advantages of server consolidation and live migration. Through live migration, data centers could flexibly move virtual machines among different physical machines to balance workloads, reduce energy consumption and enhance service availability.

Today's data centers can grow to a huge scale. This implies that frequent live migration would be desireble for the economic use of hardware resources. Then, the performance of the live migration strategy will be an issue. So, we need a reliant evaluation method to choose the software and hardware environments that will produce the best live migration performance.

However, there is not a complete live migration benchmark available currently. In addition, the existing evaluation methodologies select different metrics, different workloads and different test means. Thus, it is difficult to compare their results.

In this paper we first survey the current research and their evaluation methods on live migration. We then summarize the critical issues for the live migration evaluation and also raise other unreported potential problems.

We propose our solutions and present an implementation in our live migration benchmark – Virt-LM. This is a benchmark for comparing live migration performance among different software and hardware environments in a data center scenario. We detail its design and provide some experimental results to validate its effectiveness.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: PERFORMANCE OF SYSTEMS—*Measurement techniques, Design studies*; D.2.8 [**SOFTWARE ENGINEERING**]: Metrics—*Performance measures*

General Terms

Design, Measurement, Performance

Keywords

Benchmark, Data Center, Live Migration, Performance, Virtual Machine

*This work was supported by National 973 Fundamental Research Project of China (No. 2007CB310900).

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

1. INTRODUCTION

Virtualization technology has been widely applied in data centers and IT infrastructures, since it provides a solution to the challenges of a data center: efficient hardware utilization, reduced energy consumption, service availability and reliability, and effective host management. Virtualization enables server consolidation without interfering with server isolation. Even more excitingly, virtualization enables live migration.

Live migration means that a whole VM (Virtual Machine) can be moved between different physical machines without disconnecting the client or application. Consequently, a data center can flexibly remap hardware resources among virtual machines, to balance workload, save energy and enhance service availability.

1.1 Motives of Live Migration Benchmark

Many of today's data centers have grown to a massive scale. At Microsoft Management Summit 2008, Debra Chrapaty said Microsoft was adding 10,000 servers a month. At 2008 MYSQL User Conference and Expo, Jeff Rothschld revealed that Facebook's data center possessed more than 30,000 servers. Such a huge scale implies there should be frequent live migration to ensure economic use of the hardware resources.

However, live migration costs. It consumes hardware resources and network bandwidth. This will influence the QoS(Quality of Service). Many researchers are attracted to finding new migration strategies that have less impact. Data center administrators also have to configure a better hardware and software environment for live migration. In both cases, they must compare performance among different strategies and environments to validate their conclusion. A reliable and impartial evaluation approach is needed.

1.2 Limitations of Existing Evaluation Methodologies

However, there is not an available live migration benchmark, not to mention an authenticated one. On virtualization, existing benchmarks are all dedicated to the server con-

solidation scenario. VMware developed VMmark [11]. Intel developed vConsolidate [2]. SPEC released SPECvirt_sc2010 [15].

In current research on live migration, evaluation methodologies are designed just to support their viewpoints. They choose different metrics, select different workloads and employ different testing means. Thus, their performance results are difficult to compare. Moreover, they don't emphasize some important aspects concerning their meaningfulness, fairness and reliability.

To date recent research has only migrated VMs from an idle physical machine in their tests. However, in a real data center scenario, a VM is migrated from a busy machine to balance workloads. A busy machine consumes even more resources when enabling live migration, and its performance will degrade. Furthermore, existing research tests a migration at only one or at particular times in the whole run of a workload on a VM. However, when a VM server runs in a real scenario, it could be migrated at any moment and most likely will produces different results. Consequently, applying the results from this research would not achieve a full fairness and reliability. We will discuss more details and more issues in Section 3.

1.3 Our Contribution

To overcome these limitations, we improve the existing evaluation methods and present an implementation in our live migration benchmark – Virt-LM. Our major contributions are listed as follows.

i) This is a new design and implementation of a live migration benchmark. Our benchmark Virt-LM can compare live migration performance among different software and hardware environments in a data center scenario.

ii) We define its specific design goals of the Virt-LM. We give the reasons for its metrics and workloads choices, and consider the issues about its meaningfulness, reliability, comparability and repeatability. Its design ensures it is easy to use, stable to run and compatible with some VMM(Virtual Machine Monitor) platforms.

iii) The critical problems encountered when evaluating live migration performance are briefly explained. We identified the problems in Section 3.3.1 and 3.8 that have never been mentioned in the literature. We provide feasible solutions for these problems, and give their implementations in Virt-LM.

The rest of this paper is organized as follows. Section 2 investigates the background of live VM migration, and summarizes the limitations of existing evaluation methodologies. Section 3 defines six design goals of the Virt-LM. It also explains several critical problems associated with a live migration benchmark design, and provides some feasible solutions. Section 4 introduces the Virt-LM implementation details. Section 5 demonstrates experimental results to show its operation. Conclusion and future work are given in Section 6.

2. RELATED WORK

In this section, we introduce the typical live migration scenarios, present the live migration strategies of the current research, and finally examine the evaluation methods in their papers.

2.1 Background of Live Migration

In data centers, live migration facilitates load balancing, online maintenance and power management. Its typical scenarios are summarized below:

Load balancing. VMs are migrated from congested hosts to light ones to get optimal resource utilization, maximize workload throughput and avoid overload.

Online maintenance To free one physical machine for upgrade or maintenance, all its VM servers are migrated away without disconnecting clients. As a result, system reliability and availability is improved.

Power management. If many physical machines are light-loaded, their VMs can be consolidated into fewer ones. This frees some physical machines and saves power.

Varieties of live migration algorithms have been designed, and the prevailing ones are Pre-Copy techniques [3, 12, 18]. This have even been applied in WAN environments [1, 10, 5, 16]. The pre-Copy technique is designed to minimize VM service downtime. It copies memory pages to the destination node round after round while keeping the VM service still available. When the applications' writable working set becomes small enough, the VM is suspended and only its CPU state and dirty pages in the last round are sent out to the destination. In the pre-copy phase, although the VM service is still available, great service degradation can happen because the migration daemon continually consumes network bandwidth to transfer the dirty pages in each iteration.

Later research introduces approaches using memory compression [6] to reduce the network bandwidth consumption, and others that use remote direct memory access [5] to reduce migration overhead. Other novel strategies include Checking/Recovery and Trace/Replay approach [8] and Post-Copy technique [4].

In a WAN environment, VM image files and local block devices have to be transferred for migration. These implementations include Collective [14], Internet Suspend/Resume [7] and uDenali [17]. Some work combines Pre-Copy with writing throttling [1], while some migrate the whole VM system using block-bitmap [10].

In industry, VMware and XenSource implement VMotion [12] and XenMotion [3] in their virtualization products. They employ similar LAN schemes. Both migrate physical memory as well as network connections, and adopt the Pre-Copy algorithm to reduce migration downtime for QoS.

2.2 Existing Evaluation Methodologies

A huge scale data center may have to migrate its VMs time and again to optimize the hardware utilization, balance the workloads, save energy and keep the server maintained. They must choose a software and hardware environment to achieve the best live migration performance. However no live migration benchmark exists currently. Moreover, there are some limitations in the evaluation methods offered by the current research papers on live migration strategy design. These are given below:

Lack of Unified Metrics: There are no unified metrics among them, as shown in Table 1. We cannot compare the results produced by the different metrics. Moreover, their sufficiency is also not proven.

Lack of Unified Workloads: There are no unified workloads, as shown also in Table 1. They choose a small number of light workloads for fast evaluation, rather than the heavy workloads that resemble the real applications of a data center. In addition, VMs are migrated

from an idle machine. This is not sufficient to represent realistic load balance scenarios for live migration.

Lack of Efficient Benchmarking Methods: They employ their evaluation methods to support their conclusions instead of developing a complete migration benchmark. Hence usability, compatibility and stability are not emphasized in their tests.

3. PROBLEMS AND SOLUTIONS

In this section, we present the design goals of the Virt-LM, point out the problems encountered for each goal and provide some feasible solutions. We integrate all our solutions into the implementations of the Virt-LM. The existing literature only partly cover some goals, especially only a few works discusses stability and fairness.

3.1 Design goals of Virt-LM

The Virt-LM is designed to compare the live migration performance among different hardware and software platforms, especially in data center scenarios. We consider the following 6 goals:

1. **Metrics**. The performance metrics of a live migration benchmark must be observable, concise and sufficient to evaluate the live migration efficiency.
2. **Workloads**. Workloads must be typical and sufficient to reflect real applications of suitable size. Any workload must have a congruent behavior in any platform to faciliate the comparison of its performance results.
3. **Impartial Scoring Methodology**. The scoring method must be impartial and able to distinguish the performance results among different migration environments.
4. **Stability**. The benchmark must be stable to run. Its results have to be repeatable.
5. **Compatibility**. The benchmark shall be compatible to existing popular VMMs and easy to emigrate to future VMMs.
6. **Usability**. The benchmark shall provide a user-friendly interface that facilitates its configuration and running. Its overall test process should not last too long.

3.2 Metrics

To evaluate live migration performance, recent research (according to Table 1) usually selects metrics within following four metrics: downtime, total migration time, the amount of migrated data and migration overhead. Their definitions and measurement methods are summarized in Table 2.

Metrics selection is crucial in the design of a benchmark. A natural problem is as follows.

Problem 1: "Is it reasonable to conclude that these four metrics are sufficient to represent the influence of live migration? "

Solution: We choose all these four metrics as the metrics in our benchmark. The reason is that they already cover the prime performance aspects regarding a living migration. Migration overhead reveals its consumption of the physical

Table 1: Selection of metrics and workloads on the recent research

Papers	Metrics	Workloads
VTDC'09 Nocentino et al. [13]	Amount of migrated pages	Small C programs written by the author
Cluster'08 Luo et al. [10]	Downtime; Total migration time; Amount of migrated data	SPECweb2005; Samba server; Bonnie++
Cluster'07 Huang et al. [5]	Downtime; Total migration time; Network contentions	SPEC CINT2000; NPB
IWVT'08 Liu et al. [9]	Downtime; Total Migration time; Throughput	Kernel-compile; Memtest86; Apache AB benchmark
HPDC'09 Liu et al. [8]	Downtime; Total Migration time; Total data transmitted	Kernel-build; Apache 2.0.63; SPECweb99; Unixbench;
Cluster'09 Jin et al. [6]	Down time, Total Migration time; Total data transferred; Network throughput;	Small C programs written by the author; Apache 2.0.63; Tomcat5-5.5.23 Kernel-compile; Mummer; Dbench; memtester;
NSDI'05 Clark et al. [3]	Downtime, Throughout, Transfer rate	SPEC CINT2000; Linux kernel compile; OSDB OLTP benchmark; SPECweb99 Quake 3 server MMuncher;
USENIX'05 Nelson et al. [12]	Downtime; Total migration time; Throughput;	Kernel-compile; lometer; Memtest86; dbhammer
VEE'07 Bradford [1]	Downtime; Total migration time; Transferred overhead;	Static web content (HTTP); Dynamic web application(phpBB bulletin board softeare); Streaming video; Diabolical load bonnie;
VEE'09 Hines et al. [4]	Downtime; Total migration time; Pages transferred; Application degradation;	Memory-intensive C program; Linux kernel compilation; Netperf TCP; SPECweb2005; Bit Torrent-Client;

Table 2: The definitions and measurement methods of the four metrics

Metric	Quantified definitions	Measurement methods
Downtime	The time during which the migrating VM's execution is stopped	VM is pinged during a migration. Thus its downtime equals the time when packages are lost.
Total migration time	The period during a migration from start to finish	Measure the duration of a migration command.
Amount of migrated data	The total amount of migrated data transferred during a whole migration period	Monitor the total amount of data transferred through the TCP port dedicated for a migration.
Migration overhead	Extra machine resources consumed to perform a migration.	We compare a workload's performance in the migration case to the non-migration case, and represent the overhead with its declined percentage.

machine resources. The amount of migrated data measures its consumption of the network bandwidth. Downtime shows the service availability. Total migration time represents the whole duration when the run of workloads is influenced by live migration.

3.3 Workloads

The selected workloads for a benchmark shall represent realistic applications as closely as possible. In a live migration benchmark, the VMM of a System Under Test will have already determined its live migration behavior, so the workload is not just the migration itself but includes the condition of the environment on the platform where the migration happens. First, we have to investigate the typical conditions of a live migration in a data center. Next, we will discuss separately the internal and external conditions of a migrating VM.

3.3.1 Workloads running externally to the VM

Workloads have to represent the real scenarios of live migration in data centers. Clearly, some workloads do run outside the VM when migration occurs in a data center.

Problem 2: "How do we select workloads externally to the migrating VM?"

Analysis: The typical migration scenarios are load balancing, online maintenance and proactive fault tolerance, and power management. In a load balancing scenario, VMs are migrated from a heavy-loaded machine to a light-loaded one. For a power management case, the VM is migrated away from a relevant light-loaded machine such that the light-loaded machine could be shut down to save energy. For an online maintenance and proactive fault tolerance case, migrations may occur under any load condition.

Solution: Our benchmark tests both the heavy-load and light-load cases. To create the scenario of a heavy load, we scale the workload by running an increasing number of VMs until it reaches limit of physical resources available. For a light-loaded environment, no other application is allowed to run on the platform. Table 3 gives their implementations.

Table 3: Implementation of Workload outside Environment

Environment where VM is migrated	How to represent and implement
Heavy-loaded	Run an increasing number of VMs until the machine is fully utilized – specifically until a newly added VM obviously undermines the workload performance on each VM.
Light-loaded	VM run alone on the machine

3.3.2 Workloads running internally to the VM

The workloads running internally to a VM are selected to resemble the real applications in a data center. Hence, a problem arises here.

Problem 3: "Which workloads are representative for the data center scenarios? "

Solution: Data centers' applications mainly consist of mail servers, application servers, file servers, web servers, database servers and standby servers. Thus, we choose the existing popular benchmarks. They are listed in Table 4.

Table 4: Representative workloads for data centers

Workloads	Representative benchmarks
Mail server	SPECmail2008
App server	SPECjAppServer2004
File server	Dbench
Web server	SPECweb2005
Database server	Sysbench using MYSQL
Standby Server	Idle VM

3.4 Scoring Methodology

The benchmark has to test various kinds of workloads and then collect a large amount of results. So the next problem is as follows.

Problem 4: "How do we integrate the many workloads' results to make an overall evaluation? "

Solution: To calculate the scores, we take two steps.

- Step 1: We select four sub-metrics which consist of downtime, total migration time, the amount of migrated data and migration overhead. For each sub-metric, we merge all the workloads' results into a single score.

- Step 2: Compute a final overall score.

In Step 1, we normalize the workload results by comparing them to their reference results (test results from a uniform reference system). If each workload test is treated as been equally important, we can add all the normalized results. More formally, given metric i and workload j, let P_{ij} be the result of workload j's metric i on the target platform, and R_{ij} be its reference result. We can obtain the normalized result N_{ij} through

$$N_{ij} = \frac{P_{ij}}{R_{ij}}. \quad (1)$$

Then, the overall score S_i for all the n workloads is

$$S_i = \sum_{j=1}^{n} N_{ij}. \quad (2)$$

To make it more readable, we scale S_i as

$$S'_i = 1000 \cdot \frac{\sum_{j=1}^{n} N_{ij}}{n}. \quad (3)$$

Thus S'_i of the reference system is always 1000, because each of its N_{ij} equals 1. In this method, a lower score indicates a higher performance.

For example, Table 5 gives an overall score on the "total migration time" metric (in seconds) from the results of some SPECJVM workloads, including "Compiler", "Compress", "Crypto" and "Serial". The "normalized results" in the third row are obtained by equation (1). The "single score" in the last row are derived by equation (3).

Table 5: An example to get a single score for "total migration time"

	Compiler	Compress	Crypto	Serial
Reference system	64.24	63.24	66.63	72.41
System under test	43.73(s)	25.41	24.07	28.27
Normalized results	0.68	0.40	0.36	0.39
Single score	458.55			

It is common to use a reference platform to normalize and compute benchmark scores. For example, SPEC_CPU2000 takes this approach and uses a SUN Ultra5_10 with a 300MHz processor as its reference platform. However, to choose a reasonable reference platform, we should gather and investigate data from a wide range of systems in the following research.

After Step 1, we obtain 4 distinct scores representing the 4 performance sub-metrics respectively. They have different properties, and some have more drastic variation than others. Furthermore, it is difficult to weigh their importance objectively. So we keep a four-dimensional final result for the moment.

3.5 Compatibility

The problem that arises in compatibility is to support as many virtualization platforms as possible.

Problem 5: "The benchmark shall be available for all kinds of typical platforms of data centers."

Analysis: Because tests do not interfere with the internal operation of VMM, the benchmark is easy to migrate to different VMMs.

Solution: For the workload components, we provide the VM images with pre-installed workloads for each VMM platform. For the management agent components, we only need to change the VMM-related sections such as the commands responsible for VM management. Up to now, Virt-LM has successfully been run on XEN and KVM.

3.6 Usability

Usability is expected since complicated operation procedures will be a burden on the end user.

Problem 6: "Can the benchmark be made easy to configure and run?"

Solution: Virt-LM has been designed especially to provide a user-friendly interface that facilitates the configuration and the running of benchmark.

Firstly, workloads are packaged into pre-created VM images so that the user doesn't have to install them.

Secondly, the management agent was designed to be as automatic as possible, so that the VM management, the performance measurement and other processes could be accomplished without a user's manual interaction.

Thus, users only have to distribute the VM images and the management agent, start the test and then wait for all tests to finish.

3.7 Stability problem during the test of migration overhead

Stability means the benchmark has a congruent behavior, whenever it runs and whatever platform it runs on.

Problem 7: "How can the benchmark results be made to be repeatable?"

Analysis: As stated in Section 3.2, we use the decline of workload performance to represent the migration overhead. However, because a VM is migrated between two machines, then the performance of its internal workloads involves both machines.

For example, assuming some workload gets a score S_A on some platform A, and a score S_B on B, if running without migration. Then, as the migration can happens at different times, its score could vary from $S_A + overhead$ to $S_B + overhead$. Therefore, because we cannot precisely control the migration moment, the results may be unrepeatable.

Fortunately, there are two ways to solve it. We can force the two machines' environments to be identical, and then S_A equals S_B. Or we can force the VM to be always run on one machine. To achieve this, the VM is paused once it is migrated to machine B, and the VM states are saved and resumes on machine A. This could be accomplished through a static migration.

Solution: In Virt-LM, we force the VM always run on one machine, because it concerns only one platform acting as the System Under Test, and costs only a little extra effort.

3.8 The problem of the migration point

Any migration procedure has two properties as below.

1. Each migration has a start point (*migration point*) and lasts only for a while.

2. Its performance is determined by its start point. Specifically it is determined by the memory state before the start point and what happens during the total migration time.

Problem 8: "Performance varies as migration can commence from different times. How do we control the migration start point to guarantee its result is repeatable?"

Analysis: Ideally, we expect that each time we run the benchmark, migration will always occur at the same instruction points of the workload. Thereby we need to be notified when the workload has reached a specific instruction. However, it is very hard to communicate with workloads for this information, especially when many workloads are not open source.

Solution: We use time points as opposed to instruction points in our benchmark since it is easy to monitor the running time. First, we shall fully run the workload without a live migration to help compute the "migration overhead". Then its total running time, as a byproduct in the first run, is divided into several sectors. In the second run, the VM is migrated when each time sector is reached.

Problem 9: "Performance varies as migration occurs at different times. How to fully represent a workload's variability in performance?"

Analysis:

In real scenarios, migration occurs randomly and thus could happen at any possible start point. To represent a workload's performance, we have to collect the performance data over a sufficient number of start points.

To illustrate it, we test 483xalancbmk of SPECcpu2006, which is both CPU and memory intensive, on the same platform as mentioned in Section 5.1. The metric results vary at different migration points. Figure 1 and Figure 2 show results of its migration time and downtime.

Solution: Taking only a few random points, the results would not be satisfying because some metrics could vary drastically. For example, the results for downtime in figure 2. Therefore, a large sample of results spread over a whole run of a workload are required.

Ideally, we could test for every instruction point, but this costs too much time and effort. To find a tradeoff between the accuracy and the overhead time, over the whole run of each workload we collect as many migration point results as possible. This means each workload has to run only once or in some cases just a few times.

4. IMPLEMENTATION OF VIRT-LM

We introduce the implementations of Virt-LM in this section. We discuss its logical components, the user configuration and run procedures, and its internal processes. The details are given in the following subsections.

4.1 Logical components

The benchmark has four major logical components: System Under Test, Migration Target Platform, VM image storage and management agent. These logical components of the benchmark are illustrated in Figure 3.

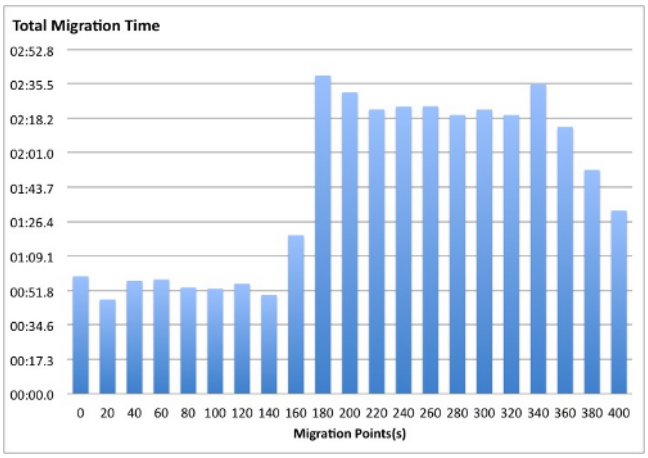

Figure 1: Total migration time results at different migration points

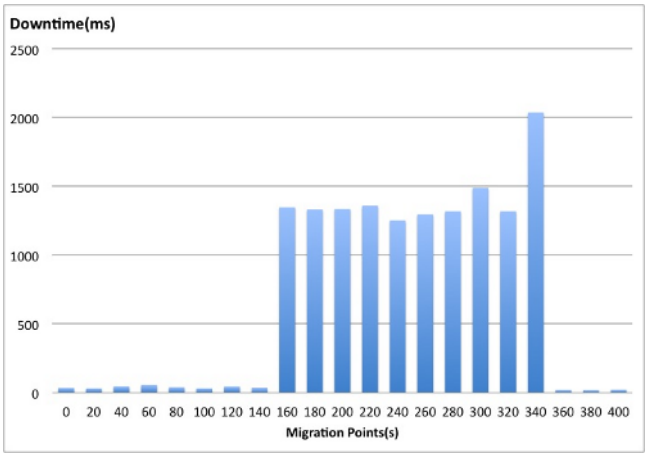

Figure 2: Downtime results at different migration points

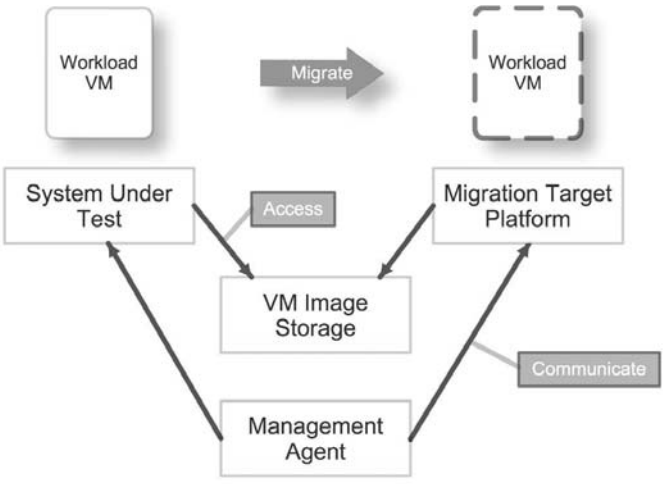

Figure 3: Logical components of Virt-LM

4.1.1 System Under Test

The SUT(System Under Test) plays a role as the evaluation target of the Virt-LM, and provides both the hardware and software environments where the migrating VM runs.

More than one virtual machine might run on the SUT, whereas only one particular VM is migrated. Next, we use "workload VM" to denote the migrating VM and distinguish it from other VMs running in the background.

4.1.2 Migration Target Platform

Because live migration must occur between two machines, the MTP (Migration Target Platform) provides the other environment that the workload VM would be migrated to. The MTP and SUT run the same VMM on different machines.

Furthermore, as mentioned in Section 3.7, to ensure a workload VM is always run on SUT, once a workload VM is migrated here, it can be moved back to the SUT through a static migration method.

4.1.3 VM Image Storage

VM images should be stored in a third party to ensure they are still accessible after migration. Physically it shall be located externally to the machine of the SUT; otherwise it may interfere with the test results.

4.1.4 Management Agent

The management agent is intended to manage the whole test process by communicating with other components. Once all tests are accomplished, it generates a final report about the live migration performance of the SUT.

However, it is not necessary to allocate four different machines for the four components. Because we have promised the performance result only for the SUT, we could allocate the SUT at one machine, and integrate the other three components at another machine for convenience.

4.2 Configuration and run procedures

Before running Virt-LM, the user has to configure and initiate each logical component.

First, VM images with pre-installed workloads are distributed into the file system of the VM Image Storage. Then the VMs' configuration files are put on the SUT. In addition, the management scripts shall be distributed and installed on the Management Agent. Figure 4 shows the Virt-LM configuration.

Next, the user launches VMMs on both the SUT and the MTP, and set up a clean environment. Each component's IP is given to Management Agent for communication. Finally, the user runs the executable script on the Management Agent. For each workload, the Management Agent will automatically reset the environment on the SUT, start a workload VM, run particular workloads inside and outside the workload VM, migrate it at the appropriate time, collect the metric results, and clean the environment for the next test.

Once the Management Agent accomplishes all the tests, it generates a final report about the live migration performance on the SUT.

4.3 Internal process

A complete test process consists of a lot of iterations. A test performs the same operations for each workload and it repeats the live migration many times in each workload test, as illustrated in the following flow diagram Figure 5.

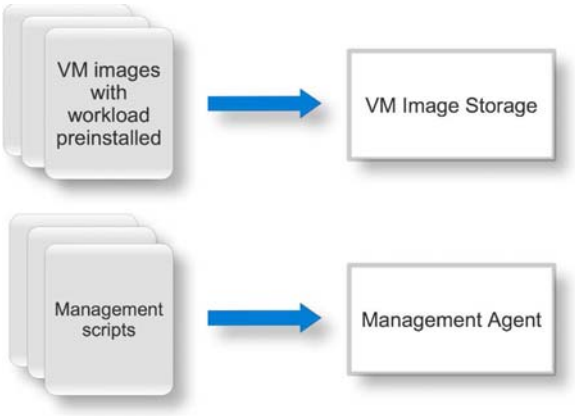

Figure 4: Virt-LM configuration

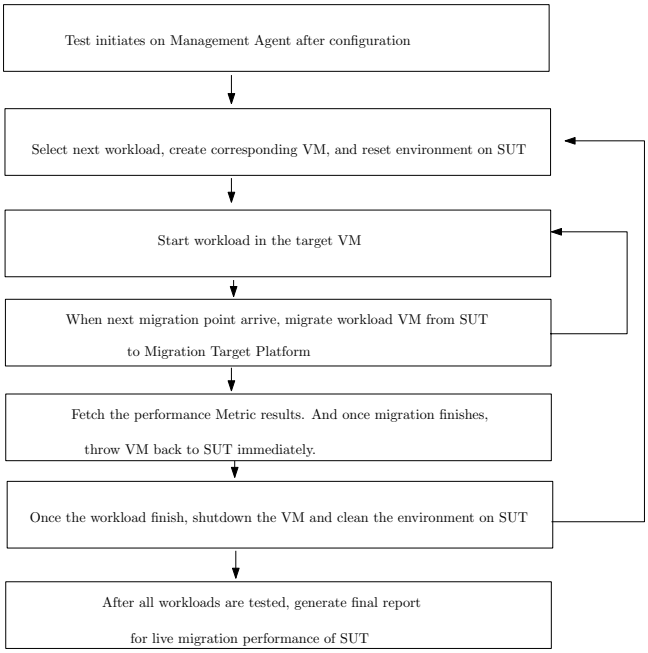

Figure 5: A brief flow diagram for Virt-LM internal process

The process includes three scopes: scope of the entire test, of each workload, of each migration measurement. Each scope includes its initialization, main jobs and the environment reset. All these steps progress automatically under control of Management Agent. Given the VM management interface of a VMM, they could be implemented without ambiguity.

However, to ensure performance is only determined by the SUT, to keep workload VM always running on the SUT, we move the VM back to SUT once a migration finishes through the static migration method. The VM is suspended when a static migration happens, so it will not run on MTP.

It is worth noticing how to repeat the measurements within each workload. As posed in Section 3.8, this concerns the strategy of selecting migration points, to fully represent the workload's performance.

Here we set a fixed frequency for each workload. The total running time of a workload is divided into serveral sectors, and then the benchmark start every round migration when each time sector is reached.

5. EXPERIMENTAL RESULTS

In this section we give the demonstration the experimental results of Virt-LM. We run the Virt-LM in two VMMs – Xen and KVM. We show the specific steps to obtain a final score from the raw metrics results and compare the performance results between Xen and KVM.

Disclaimer: The scope of the tests is limited. However, the intention is to demonstrate that this prototype application is working and the principle of operation is established.

5.1 Experimental Setup

A DELL OPTIPLEX 755 containing 2.4GHz Intel Core Quad CPU Q6600, configured with 2GB of memory and a SATA disk, was used as demonstration System Under Test hardware. Our experiments were conducted on two SUT platforms, one was XEN 3.3 on Linux kernel 2.6.27, and the other was KVM-84 on Linux kernel 2.6.27.

The other three components are integrated into another machine with an identical VMM and hardware. Two machines are connected over a single 100 Mbit network link.

5.2 Experimental Results

Here we illustrate how to get scores from raw measurement results and how to generate a final performance report.

We choose the workload SPEC-jvm2008 in the light-load environment to demonstrate, the other workloads can be done similarly. SPEC-jvm2008 is a benchmark suite for measuring the performance of Java Runtime Environment (JRE). It contains several real applications and benchmarks that focusing on core java functionality. SPEC-jvm2008 represents an application server's workload.

The configurations in the VM are Ubuntu 8.10 on Linux kernel 2.6.27, 512MB memory, and one CPU core.

As Section 3.8 and Section 4.3 shows, we test many migration points during a run of a workload VM. For each workload, we get several results of the same metric and calculate their averages. This experiment is repeated three times, and the standard deviation of each metric score is within 1 percentage. We also test the memory and CPU intensive sub-workloads of SPCE-CPU2006. Their results also show good repeatability even at every migration time point's results. It suggests that the migration time points are stable in a fixed software and hardware environment.

The following Table 6 shows metric results of SPEC-jvm2008's workloads on our XEN SUT.

By analogous calculations, we have the 4 metric results on SUT-KVM in Table 7.

To demonstrate how to normalize and integrate results, we use SUT-XEN as a reference system. According to Equation (2) and Equation (3), we have the final scores of "downtime", which are given in Table 8.

Likewise for other three metrics, and then we obtain final results in Table 9.

Table 6: Metric results on SUT-Xen

Workloads of SPEC-jvm-2008 on SUT-XEN	Downtime (ms)	Total migration time (s)	Amount of migrated data (MB)	Workload's own performance in non-migration case (ops/m)	Workload's own performance in migration case (ops/m)	Migration overhead (%)
Compiler	436	64.24	745	33.37	33.09	0.84
Compress	60	63.24	742	32.95	32.83	0.36
Crypto	119	66.63	748	32.07	31.81	0.81
Serial	247	72.41	844	24.67	24.43	0.97

Table 7: Metric results on SUT-KVM

workloads of SPEC-jvm-2008 on SUT-KVM	Downtime (ms)	Total migration time (s)	Amount of migrated data (MB)	workload's own performance in non-migration case (ops/m)	workload's own performance in migration case (ops/m)	Migration overhead (%)
Compiler	1748	43.73	501	40.27	37.05	8.00
Compress	1041	25.41	290	34.4	30.67	10.84
Crypto	876	24.07	286	29.86	28.56	4.35
Serial	1053	28.27	312	23.07	20.46	11.31

Table 8: "Downtime" final scores on SUT-XEN and SUT-KVM

	Downtime on SUT-XEN	Downtime on SUT-KVM	Downtime on reference system	normalized score on SUT-XEN	normalized score on SUT-KVM
Compiler	436	1748	436	1	4.01
Compress	60	1041	60	1	17.35
Crypto	119	876	119	1	7.36
Serial	247	1053	247	1	4.26
Final "downtime" score				1000.00	8245.92

Table 9: Final scores on SUT-XEN and SUT-KVM

	Downtime	Total migration time	Amount of migrated data	Migration overhead
SUT-XEN	1000	1000	1000	1000
SUT-KVM	8245	459	454	14075

5.3 Discussions and Analysis

In our scoring method, a lower score indicates a higher performance. The results in Table 9 show that, by sacrificing more downtime and migration overhead, SUT-KVM reduces the amount of migrated data and the total migration time. To interpret these phenomena, we could review and compare their raw data from Figure 6.

The clues are "amount of migrated data" and "downtime". Because our workload VMs are configured with 512 MB memory, it seems that the SUT-KVM intensively compresses the migrated data, which leads to its higher "migration overhead". On the other hand, the SUT-XEN restricts its "downtime" more harshly, which suggest it transfers many more dirty pages at pre-copy phase, and further increases "amount of migrated data". In summary, comparing with the SUT-KVM, the SUT-XEN has better performance at "downtime" and "migration overhead", but places more burden over network.

This pilot experiment demonstrates the principle of operation. One issue that has been not been tackled here is scalability. However, it was felt that this could raise more issues that there would have been space to deal with in this paper.

6. CONCLUSION AND FUTURE WORK

In this paper, we surveyed the current research and their evaluation methods on live migration, and summarized the critical issues for live migration evaluation. The current literature doesn't mention the issues in Section 3.3.1 and 3.8. Then we proposed our solutions and gave their implementations in our live migration benchmark – Virt-LM, a benchmark comparing live migration performance among different software and hardware environments in a data center scenario. We introduced its internals and provided some experimental results to validate its effectiveness.

However, many aspects of the design are far from satisfying, such as how to design a better strategy to select migration points, how to scale up the VMs to represent a heavy-load environment. Furthermore, a host in a data center could receive multiple migration requests at the same time. The VMs could be migrated one by one or concurrently. These scenarios also need to be tested to get the best number of the concurrent migrating VMs. It still remains for our future research to give solutions for those problems.

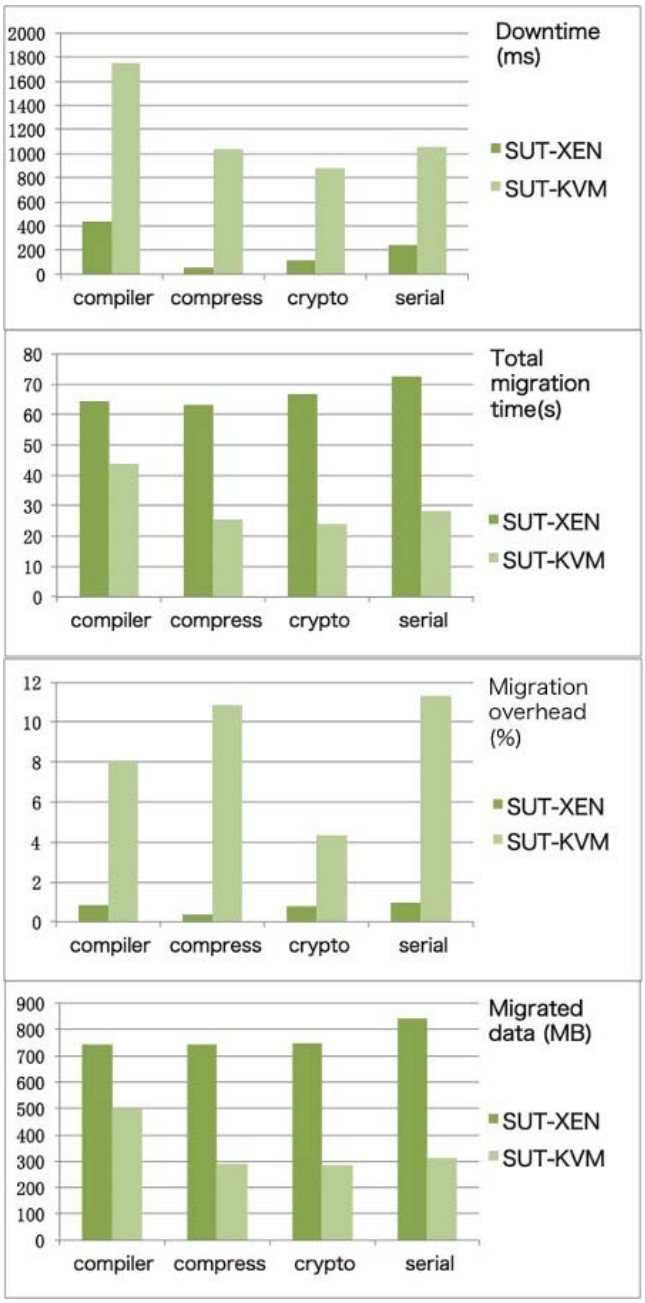

Figure 6: Comparison of metric results between SUT-XEN and SUT-KVM

7. REFERENCES

[1] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd International Conference on Virtual Execution Environments*, pages 169–179, 2007.

[2] J.P. Casazza, M. Greenfield, and K. Shi. Redefining server performance characterization for virtualization benchmarking. *Intel Technology Journal*, 10(3):243–251, 2006.

[3] 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 conference on Symposium on Networked Systems Design & Implementation*, pages 273–286, 2005.

[4] M.R. Hines and K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 51–60, 2009.

[5] 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 IEEE International Conference on Cluster Computing*, pages 11–20, 2007.

[6] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan. Live virtual machine migration with adaptive memory compression. In *Proceedings of 2009 IEEE International Conference on Cluster Computing*, pages 1–10, 2009.

[7] M. Kozuch and M. Satyanarayanan. Internet suspend/resume. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, pages 32–40, 2002.

[8] 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*, pages 101–110, 2009.

[9] P. Liu, Z. Yang, X. Song, Y. Zhou, H. Chen, and B. Zang. Heterogeneous live migration of virtual machines. In *Proceedings of the International Workshop on Virtualization Technology*, 2008.

[10] 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 the 2008 IEEE International Conference on Cluster Computing*, pages 99–106, 2008.

[11] V. Makhija, B. Herndon, P. Smith, L. Roderick, E. Zamost, and J. Anderson. VMmark: A scalable benchmark for virtualized systems. *VMware Inc, CA, Tech. Rep. VMware-TR-2006-002, September*, 2006.

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

[13] A. Nocentino and P.M. Ruth. Toward dependency-aware live virtual machine migration. In *Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing*, pages 59–66, 2009.

[14] C.P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M.S. Lam, and M. Rosenblum. Optimizing the migration of virtual computers. *ACM SIGOPS Operating Systems Review*, 36(SI):377–390, 2002.

[15] SPECvirt_sc2010. http://www.spec.org/virt_sc2010/.

[16] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. De Laat, J. Mambretti, I. Monga, B. Van Oudenaarde, and S. Raghunath. Seamless live migration of virtual machines over the MAN/WAN. *Future Generation Computer Systems*, 22(8):901–907, 2006.

[17] A. Whitaker, R.S. Cox, M. Shaw, and S.D. Grible. Constructing services with interposable virtual hardware. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation-Volume 1*, pages 13–13, 2004.

[18] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation*, pages 229–242, 2007.