Downtime Analysis of Virtual Machine Live Migration

Felix Salfner
Department of Computer Science
Humboldt-Universität zu Berlin, Germany
salfner@informatik.hu-berlin.de

Peter Tröger, Andreas Polze
Operating Systems and Middleware Group
Hasso-Plattner-Institute at University Potsdam, Germany
{peter.troeger,andreas.polze}@hpi.uni-potsdam.de

***Abstract*—Virtual machine environments like VMware, XEN, KVM, and Hyper-V support live migration of guest operating systems, which is used in data centers to provide uninterrupted service during maintenance or to move computation away from failure-prone hosts. The duration of migration, as well as the virtual machine downtime during this process are essential when assessing if service availability agreements might be violated.**

We present the result of an experimental study that analyzed virtual machine live migration downtime and duration. We show that total migration time as well as downtime are dominated by specific memory utilization patterns inside the virtualized guest. We experienced that downtime involved by live migration can vary by a factor of more than 23, which can have significant impact on service availability.

***Keywords*-virtual machine, live migration, downtime**

I. Introduction

Virtualization as a concept for isolation and multiprogramming has been known since the late 60's [1]. Today, many virtualization products for commodity server and desktop environments exist. Most platforms support *live migration*, which allows to move a running virtual machine (VM) to a new physical host with minimal service interruption. This renders live migration a very attractive tool for various scenarios in dependable computing. Currently the predominant use of live migration is in data centers or compute clouds where VMs can be moved across physical hosts for load balancing, server consolidation or maintenance. In all these cases knowing the downtime involved by moving the VM is essential when service availability guarantees have to be fulfilled: the time of service interruption must not exceed the clients retry intervals.

A second area where live migration is used is proactive fault management. VMs are moved away from a physical node that has been predicted to show a failure in the near future (see, e.g., [2]). In addition to the downtime involved in moving the VM, the total duration of migration is an important characteristic. This is because the entire migration has to have finished before the failure occurs.

However, the majority of existing work that builds on live migration of VMs simply assumes some fixed (in many cases arbitrary) duration of the live migration and the downtime involved by it. According to the experiments presented in this paper, such an assumed value has to be chosen very carefully since migration time as well as downtime can vary by an order of magnitude or more, depending on the memory workload. It is the goal of this paper to systematically investigate the factors determining the time needed for VM live migration.

II. Virtual Machine Live Migration

Within the different existing virtualization frameworks with live migration support, the basic principle is that the virtualization cluster management actively moves a virtualized system while it is still executing and is still changing the hardware's and software's state. Today's products realize this by a delta-copying approach where modified memory regions are incrementally transferred until a lower threshold for data to be moved is reached. In the subsequent phase in which the VM is stopped, the remaining resources are copied and reconfigured and the VM is resumed on the destination host. This leads to the two characteristics investigated in this paper:

- *migration time* is the time from start of the live migration process until the virtualization framework notifies that the source host can be deactivated.
- *downtime* or *blackout time* is the phase during migration when there is a user-perceptible service unavailability.

The most difficult procedure in live migration is the transfer of main memory state. As live migration environments typically share storage within the migration cluster, swapped out memory pages do not have to be considered. Read-only pages from the working set (such as code pages) need to be migrated only once, whereas data pages could be modified again after their initial transfer. Transfer of writable memory can happen theoretically in three phases [3]: In the initial *push phase*, in which the source machine's actively used set of pages is copied to the destination host in rounds. In the subsequent *stop-and-copy phase*, in which the source VM is suspended, remaining memory regions are transfered, and the VM is resumed again on the destination host. The length of this phase is the VM downtime. The last step is the *pull phase*, where the VM running on the destination host might access memory regions that are still not migrated, which are then pulled from the source host. The end of the pull phase marks the end of the migration time. The time of transition from one phase to the next is controlled by adaptive algorithms that take into account various aspects such as the number of dirty memory pages, etc. Most live migration products combine the first two phases in a so-called *pre-copy* approach.

III. Experiment Design and Load Model

Live migration duration is influenced by load factors from inside the VM and from the underlying physical host. For our investigations we assumed a typical (and recommended) setup where applications only run in VMs and there are no applications running on the physical host directly (except for the hypervisor).

Experiments carried out prior to the ones presented here have shown no impact on migration time and downtime when running multiple VMs on one physical host. Results shown here have therefore been measured with just one VM per physical host. We assumed a model of VM operation without *over-commitment*, which is a VM configuration where the sum of virtual memory of all VMs on a host does not exceed the amount of physical memory.

The focus of this work is on application-specific effects on live migration. Since in most scenarios the migration network is not a controllable parameter we did not investigate effects of the migration network on migration performance. Additional tests investigating network usage showed that the migration frameworks handle network capacity issues carefully so that this assumption appears valid.

The goal of our experiments are to investigate the effects of the following factors on migration time as well as downtime: CPU load, configured memory size of the VM, utilized amount of memory, and memory modification pattern for two different virtualization products.

A. Load Generators

In order to be able to analyze the effects of each factor and its combinations we used three different load generators - one for CPU load and two for memory load.

The *CPU load generator* was based on *burnP6* and *cpulimit* generating a controllable CPU utilization in the running VM.

The *locked pages generator* is used for analysis of static memory allocation that cannot be swapped out. This is achieved by allocating a given amount of memory, writing random data to it and locking it in physical memory using the according system call.

In order to investigate the effect of modifying memory pages while the VM is migrated, we programmed a *dirty pages generator*, simulating memory write access of applications running in the VM. It implements a cyclic memory modification pattern by continuously writing pre-computed random data to locked memory locations. This pattern is motivated by server applications, which modify memory regions based on incoming requests. These modifications can be expected to have comparable characteristics for the majority of requests, e.g., by always reading some data, storing logging information, and returning the result.

Table I
Investigated parameters

HYPERVISOR	The virtualization framework used
VMSIZE	Amount of main memory statically configured for the VM
LOAD	CPU utilization of the virtualized operating system
WSET	Working set, the sum of utilized memory
PERIOD	The period for one memory modification cycle
BPC	Blocks per cycle. Number of modified blocks per cycle
FILL	Filling degree. The average percentage of a memory block being actively modified

A list of all parameters investigated in our experiments is provided by Table I.

B. Technical Setup and Issues

All tests were performed on two Fujitsu Primergy RX300 S5 machines with a shared iSCSI drive, the migration was performed for a VM running Linux 2.6.26-2 (64 bit). All VMs were configured to have one virtual CPU and a varying amount of (virtualized) physical RAM. In all cases, the virtualization guest tools / drivers were installed. Native operating system swapping was activated, but not aggressively in use due to the explicit limitation of the allocated amount of memory.

We conducted all experiments with the two hypervisors VMware and Xen. Experiments for VMware were performed with ESX 4.0.0, using the vCenter server software for migration coordination. High availability features were deactivated. Experiments for Xen were performed with Citrix XenServer 5.6 (Xen core 3.4.2). Both Xen hosts were configured to form a pool, the test scripts were executed in the dom0 partition of the pool master.

Total migration time was measured by capturing the runtime of the product command-line tool that triggers a migration. Downtime was measured by a high-speed ping (50 ms) from another host, since the virtualization products do not expose this performance metric by themselves. The downtime is expressed as the number of lost Ping messages multiplied by the ping interval.

Live migration, similar to every performance-critical software feature, is influenced by a manifold of hardware / software factors. We are aware of the fact that new product versions, node and networking hardware as well as special optimization switches can lead to better or worse results. Nevertheless, the point of our investigations is to identify major impact factors when using live migration for dependability purposes.

IV. Single Variable Experiments

Since the number of parameters (also called *factors*) is too large for an investigation of all combinations of factors with each factor tested at multiple levels, we first aimed at reducing the set of factors. In order to do so, we investigated each parameter individually. As will be shown in this section, this analysis helped to eliminate the two parameters LOAD and FILL.

In order to investigate a single factor we set all but the investigated parameters to fixed values and measured both downtime and total migration time at various levels of the investigated parameter.

A. VMSIZE

We investigated the influence of the configured VM main memory with different settings for both products. Specifically, we have investigated idle VMs with VMSIZE set to 512MB, 1GB, 2GB, 4GB and 8GB RAM. The virtual machines were idle, so no load generator was used. While VMware showed a nearly constant migration time, Xen had a linearly growing migration time with increasing VMSIZE.

B. LOAD

For the investigation of the influence of CPU load on migration time, we performed at least 10 migrations per CPU utilization degree, ranging from 0% to 100% artificial CPU load in steps of size ten using the CPU load generator. Results showed almost no impact with a 95% confidence interval of not more than -/+ 1s for all load values.

Both products also showed a constant (significantly smaller) downtime in all constellations, with a 95% confidence interval of not more than +/- 10% of the average downtime.

The results suggest that both virtualization frameworks reserve enough CPU time for their own management (migration) purposes. Live migration scenarios seem to only depend on non-CPU utilization factors. We could hence safely drop CPU load as an influencing factor in subsequent experiments.

C. WSET

Using the locked pages generator, we varied the size of the working set from zero to 90% of the virtual memory available to the VM. Results show that VMware downtime as well as migration time depend on WSET. Xen also shows dependency on WSET so that the effects of WSET are analyzed further in Section V.

D. FILL

In order to rely on the trap and page table mechanisms of the operating system, all VM migration approaches copy memory content in pages. Hence an entire page has to be migrated even when only a fraction of a page is written. We tested this assumption by “filling” pages to a varying degree using the dirty page generator. As expected, both virtualization toolkits showed no effect on downtime or migration time.

E. PERIOD and BPC

The parameters PERIOD and BPC determine how frequently memory pages are modified. In order to assess how total migration time and downtime are affected by them we conducted a series of experiments where we varied PERIOD for a number of settings for BPC using the dirty page generator. Results show that both downtime and migration time are strongly affected by the two parameters. To check for stochastic variability, we determined 95% confidence intervals, which never exceeded 5% of the average value. However, the influence of PERIOD and BPC are complex and will be further investigated in Section V.

To better understand the complex behavior we performed a source code analysis of Xen and had personal communication with VMware representatives. The behavior seems to be mainly related to the rate-adaptive migration control. The relevant aspect here is the dirty page diff set, the fraction of pages that are scheduled to be copied in each next round of the pre-copy phase. Both virtualization products obviously identify ”hot pages” in this set and shift such pages more aggressively to the stop-and-copy phase because for hot pages a block transfer in the stop-and-copy phase is potentially more effective (depending on “hotness” of the page, network link speed and other factors). Akoush et al. [4] made similar investigations in their live migration performance analysis.

F. Summary

Summing up these experiments, we observe that live migration duration as well as downtime can depend heavily on the investigated factors. On the other hand we saw that CPU load as well as the degree to which memory pages are filled do not influence migration performance significantly which allows us to exclude them from further investigations.

In order to deal with the mutual non-trivial dependencies seen in this section we subsequently devised experiments that investigate all combinations of factors, as will be presented in the next section.

V. Multi Parameter Experiments

From the experiments shown in the previous section we were able to conclude that the factors LOAD and FILL can be omitted from further analysis.

A second reduction in the number of factors can be achieved by leveraging on the fact that BPC (blocks per cycle) and PERIOD (duration of one cycle) can be combined into one factor $RATE = \frac{BPC}{PERIOD}$, which denotes the number of blocks that are modified per millisecond.

We have hence reduced the number of factors to the following three parameters: VMSIZE, WSET, and RATE.

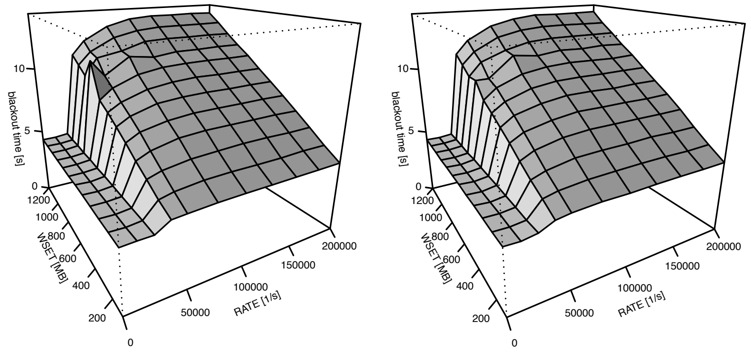

Figure 1. Mean downtime for Xen plotted over WSET and RATE for VMSIZE=4096 (left) and VMSIZE=8192 (right)

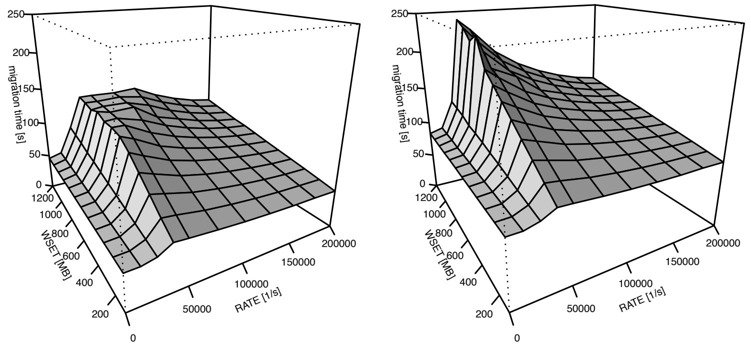

Figure 2. Mean migration time for Xen plotted over WSET and RATE for VMSIZE=4096 (left) and VMSIZE=8192 (right)

We performed experiments according to a full factorial design, meaning that all possible combinations of parameter levels have been measured in the experiment. More specifically, for Xen we investigated a total number of 528 parameter combinations (treatments), each with 20 measurements resulting in an overall number of 10560 migrations. In each experiment we measured migration time and downtime as response variables. In case of the VMware hypervisor, we performed experiments for 352 combinations resulting in 7040 migrations.

In the following we will discuss results for each virtualization framework separately.

A. Analysis of XenServer

As we have three factors (plus a response variable) we cannot present the entire results in one plot. Since VMSIZE has the least number of levels, we decided to plot the mean response, i.e. mean migration time or downtime, over WSET and RATE for fixed values of VMSIZE (see Figures 1 and 2). Comparing the two figures, we can see that downtime shows a very different behavior in comparison to migration time, although the first is part of the latter.

Downtime (Figure 1) in general increases with increasing WSET and increasing RATE. This is not surprising as an increased usage of memory (more pages written at an increasing rate) requires more memory to be transferred in the stop-and-copy phase. We can also conclude from the figure that WSET seems to have a linear effect on downtime, regardless of the values of VMSIZE and RATE.

Turning to total migration time (Figure 2) we observe that the mean migration time is more irregular. It came as a little surprise to us that for RATE levels ”above the jump” total migration time decreases with increasing RATE. In order to check that this behavior really occurs we have carried out separate experiments specifically targeted to this question with the same consistent result. The behavior can be explained by the documented stop conditions for the pre-copy phase in these products. The precopy phase of Xen stops (1) when a sufficiently small amount of memory is left on the source or (2) if an upper limit for the transferred data was reached or (3) if the time taken becomes too long (measured by the number of pre-copy rounds) [4]. Hence if the modification rate grows beyond a certain value close to the link speed, the algorithm will end the pre-copy phase earlier resulting in the observed behavior of constant downtime and decreasing overall migration time.

One peculiarity in Figures 1 and 2 is the abrupt change at a RATE level around $30{,}000\frac{1}{s}$. In order to analyze this further, we conducted additional “zoom-in” experiments that investigated a sub-range of values for RATE at greater level

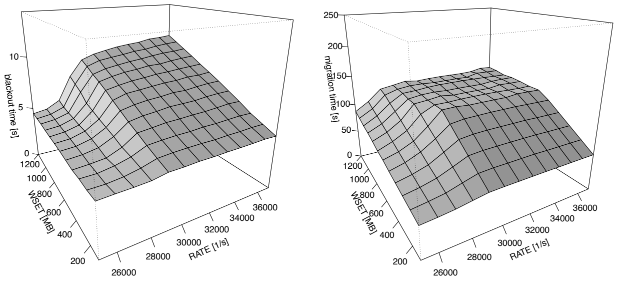

Figure 3. Mean downtime (left) and migration time (right) for Xen with additional WSET and RATE levels in the ”zoom-in area”, VMSIZE=4096

of detail (see Figure 3). As it can be seen from the plot, the change is not as abrupt as might have been concluded from Figures 1 and 2 and only appears to be abrupt due to the scale of the plot and due to a lack of intermediate measurement points.

The effect of VMSIZE can be observed by comparing the two plots in Figure 2. It can be seen that VMSIZE has a non-trivial effect on migration time: since the shapes look very different at different levels of VMSIZE, the effect does not appear to be linear, except for the case where RATE equals zero. There is no effect of WSET if RATE is zero.

The plots in Figures 1 to 3 show times averaged over all 20 measurements. In order to assess the variability in the data, we report the ratio of maximum to minimum values as well as standard deviations for the data in Table II. Specifically, two ratios and two standard deviations are reported: the ratio of the maximum treatment mean to the minimum treatment mean and the ratio of the maximum to the minimum values across all measurements. Regarding standard deviations we report the largest standard deviation computed within each treatment as well as the standard deviation for the overall data set. In addition, the table reports the mean time averaged across all measurements.

The table quantifies what has also been observable from the plots: Both migration time as well as downtime vary tremendously depending on the three investigated parameters.

B. Analysis of VMware

Due to space limitations we report results for VMWare only for VMSIZE equal to 4GB (see Figure 4). This is no severe limitation as the behavior is very similar for other values of VMSIZE.

As can easily be observed the behavior differs significantly from the one of Xen, which emphasizes that the choice of the hypervisor product can have significant impact on availability. The main reason for the different behavior seems to be the different rate-adaptive algorithms employed in the two virtualization products.

Variability of the data for VMware is also listed in Table II. Regarding the max:min ratio of downtime computed from treatment means we have observed a ratio of 16.27. This shows that due to different memory load the maximum mean downtime can be 16.27 times as large as the minimum mean downtime. If we instead consider the maximum and minimum value observed across all experiments, the factor even goes up to 23.83! The conclusion from this observation is that if service downtime is critical for meeting availability goals, a realistic assessment of availability can only be achieved if the maximum downtime for the application-specific memory load is used.

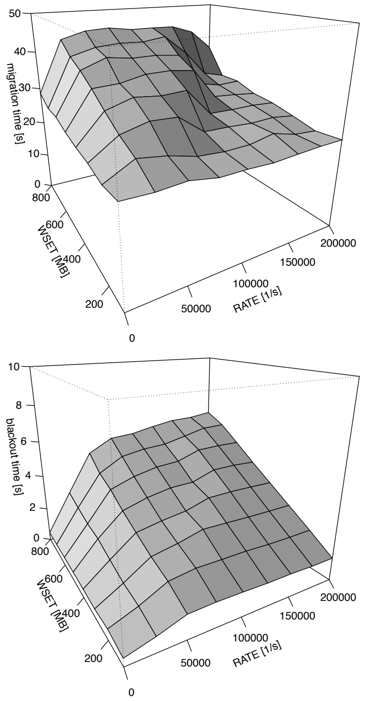

Figure 4. Mean migration time (top) and downtime (bottom) for VMware plotted over WSET and RATE, VMSIZE=4096

VI. Related Work

In the area of dependable computing, VM live migration has primarily been used as a tool. Two examples are *proactive fault tolerance* [2] and the approach to resource allocation proposed in [5].

A second group of related work deals with various aspects of implementing VM live migration. Hines and Gopalan [6] discuss the modification of Xen for *post-copy* live migration. Sapuntzakis et al. [7] introduced several optimization approaches for VM live migration, among which *ballooning* is best-known, which forces the VM to swap out as much memory as possible.

This work, however, is somewhere in between using live migration as a tool and investigating aspects of its implementation: We have focused on the factors determining downtime and migration time from an application's perspective. A work that is closer related to ours is [3], which

Table II
DATA VARIABILITY

Hypervisor / Guest	time	Mean time [s]	Max:Min Ratio		Standard Deviation	
			Mean	Overall	Treatment Max [s]	Overall [s]
XenServer / CentOS	migration	89.73	9.01	9.10	6.32	39.08
	downtime	7.69	3.17	3.46	0.62	2.94
VMware / Linux	migration	30.93	2.24	2.96	7.72	7.51
	downtime	3.10	16.27	23.83	0.50	1.80

investigates the effect of the size of the *writable working set*. The migration times reported are much smaller than the ones reported here. This is probably due to the fact that the workloads used in their experiments do not result in significant memory load.

VII. CONCLUSION

With growing capacity of commodity server hardware and increased consolidation efforts, virtualization has become a standard approach for data center operation - not only on the mainframe and for UNIX systems, but also in the world of Intel servers. Live migration of virtual server workloads can be employed to implement workload-driven system management as well as mechanism to free server hardware that is due for maintenance and repair. However, in order to give guarantees on application availability or responsiveness as well as for proactive fault management, solid estimations either about the total duration of live migration or the length of service downtime are mandatory.

In this paper, we have reported about our research on major factors that influence migration time and migration-induced downtime. Our measurements are based on two representative virtualization products, namely VMware ESX 4.0.0 and Citrix XenServer 5.6. By carrying out a wide range of experiments, our analysis shows that performance of live migration can vary significantly depending on the memory load and memory access patterns of the guest system.

The results can be used, e.g., to investigate the applicability of VM live migration in the context of proactive fault management: If VMs are to be migrated away from failure-prone hosts the failure prediction algorithm needs to predict failures further in the future than the total duration of migration. Our results also help to assess if service availability assertions are violated by the downtime introduced by live migration of a VM running the service. Even if the absolute numbers may be different for future versions of the virtualization products our results highlight that application-specific investigations are crucial to assess the feasibility of live migration in a particular scenario.

A second area where our results are useful is to help to improve live migration features of virtualization products. For example, the observation that Xen migration time depends on the size of virtual RAM configured might be an indicator how live migration can be improved futher, or might even stimulate new ideas for VM live migration.

Future work will involve investigation of other virtualization approaches (e.g., KVM and Microsoft Hyper-V). We will also focus on the relationship between the load generators used in this work and real-world applications.

VIII. ACKNOWLEDGMENTS

We would like to thank Matthias Richly from Hasso Plattner Institute for the realization and management of the experimentation environment. All hardware resources were made available by the HPI FutureSOC lab infrastructure.

REFERENCES

[1] R. P. Goldberg, “Survey of Virtual Machine Research,” *IEEE Computer*, vol. 7, no. 6, pp. 34–45, 6 1974.

[2] C. Engelmann, G. R. Vallée, T. Naughton, and S. L. Scott, “Proactive Fault Tolerance Using Preemptive Migration,” in *Proceedings of 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, PDP, 2009*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 252–257.

[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 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, NSDI, 2005*. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286.

[4] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, “Predicting the Performance of Virtual Machine Migration,” *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, vol. 0, pp. 37–46, 2010.

[5] S. Fu, “Failure-aware resource management for high-availability computing clusters with distributed virtual machines,” *Journal of Parallel and Distributed Computing*, vol. 70, no. 4, pp. 384–393, 2010.

[6] M. R. Hines and K. Gopalan, “Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning,” in *Proceedings of 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, Washington, DC, USA*, ser. VEE ’09. New York, NY, USA: ACM, 2009, pp. 51–60, washington, DC, USA.

[7] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, “Optimizing the migration of virtual computers,” *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 377–390, 2002.