A Machine Learning-based Approach to Live Migration Modeling

Changyeon Jo, Changmin Ahn, and Bernhard Egger
Department of Computer Science and Engineering
Seoul National University
Seoul, Korea
Email: {*changyeon,changmin,bernhard*}*@csap.snu.ac.kr*

***Abstract*—*Live migration* is one of the core technologies to increase the efficiency of data centers by enabling better power savings, a higher utilization, load balancing, and simplifying maintenance. With service-level agreements (SLA) in place, the overhead of live migration in terms of resources consumed on the host plus the performance reduction and downtime of the migrated VM poses a major obstacle to effectively apply live migration. With various live migration algorithms available, an important question is then which of the algorithms can provide optimal performance while respecting the SLAs. In this work, we propose a versatile model that is able to accurately predict the key metrics of live migration. The machine-learned model is trained with data from over 10,000 VM migrations and evaluated for the five live migration algorithms available in the latest QEMU/KVM virtualization environment. The evaluation shows that the proposed model is able to predict the total migration time and the total transferred data with over 90% accuracy, and 90th percentile error of the downtime is 280ms.**

1. Introduction

Virtualization allows data center operators to better utilize their resources by running multiple virtual machines on one physical host. In order to adapt to fluctuating workloads in virtual machines and optimize the utilization of hardware resources, virtual machines can be *live migrated* [3], i.e., moved from one physical host to another while the virtual machine (VM) keeps running. To balance the load between servers, VMs running on over-committed hosts can be migrated to idle servers. On the other hand, the VMs of a lightly-loaded server can be consolidated onto another machine, and the now idle server can be turned off, thereby increasing the power efficiency of the data center.

Migrating a VM requires copying its volatile state from the source to a destination host. The simplest approach, *stop-and-copy*, stops the VM on the source, completely transfers the VM's state, and finally resumes the VM on the destination host. In the presence of service-level agreements (SLA) between the data center operator and the owner of the VM requiring a certain availability of service, the stop-and-copy approach is unfeasible due to its long period during which the VM remains unavailable.

This *downtime* is not the only key metric of live migration. Other important factors include the *total migration time*, the *total amount of data transferred*, and *the performance degradation* of the VM being migrated. Additionally, the amount of CPU and memory resources and the network bandwidth required by the migration may also be of interest, especially in a resource-constrained environment.

Over the past decade, a number of live migration techniques have been proposed [3], [4], [5], [7], [10], each of which aims at optimizing one or several of the above metrics. The proposed techniques range from copying the volatile state iteratively while the VM keeps running on the source host to moving the core of the VM immediately and fetch outstanding data on demand. A number of orthogonal optimizations such as data compression or CPU throttling have been proposed as well. The different techniques exhibit distinct characteristics in the key metrics for identical workloads. In addition, for a given technique, its performance shows a large variance subject to the workload running inside the VM and on the host. In order to apply live migration effectively, an important problem for data center operators is thus to select the best migration technique as a function of SLAs, the operator's optimization policy, plus the workload characteristics of the VM and the host.

In this work, we present a method to build accurate performance estimation models for live migration. Based on an analysis of a large dataset of live migration profiles of diverse workloads migrated with different live migration techniques under varying resource constraints, we employ Machine Learning techniques to automatically generate a performance prediction model. The model can estimate the total migration time, the total amount of data transferred, and the VM downtime for the different migration techniques. By virtue of the automatic approach, new migration algorithms and profile features can be easily added, rendering the proposed procedure flexible and extensible. We verify the feasibility of the proposed approach in the QEMU/KVM virtualization environment [6]. Based on over 10'000 migration profiles, the generated model predicts the total migration time, the total amount of transferred data, and the downtime with high accuracy. In ongoing research, we employ the model in a data center optimization framework to select the best algorithm for a given situation and constraints.

The remainder of this paper is organized as follows: Section 2 provides the necessary background on live migration and discusses the different live migration techniques. Section 3 details the data collection and model building process, and Section 4 evaluates the models. Section 5, finally, concludes the paper and discusses future work.

2. Background

2.1. Live Migration

Live migrating a VM requires moving its entire state from one physical host to another. In intra-datacenter migration, the permanent storage is accessed via NAS and does not need to be moved. The volatile state consists of the state of the VM's VCPUs, devices, and its memory contents. The latter constitute by far the largest part of the volatile state. The simplest way of migrating a VM is *stop-and-copy*, i.e., stopping the VM on the source, transferring the entire state to the destination, and then resuming the VM. The long downtime renders this approach impractical; it is thus common practice to live-migrate VMs. We distinguish the following three phases (Figure 1):

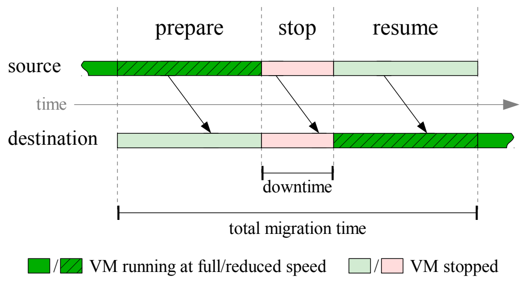


Figure 1. Phases and metrics in live migration.

1) **Prepare**. Live migration is initiated. The virtual machine monitor (VMM) puts the VM into a managed mode which typically results in a slightly reduced performance. The source host starts sending (parts of) the volatile state to the destination host.
2) **Stop**. The VM is stopped both on the source and the destination host and thus not available to the user. Small amounts of volatile state, e.g., register values of VCPUs, are transferred.
3) **Resume**. The VM is restarted on the destination host. Missing parts of volatile state are fetched. VM performance may still be reduced in this phase.

2.2. Live migration metrics

We compare the different live migration algorithms using the following metrics (Figure 1):

1) **total migration time:** time period from initiation to completion of the migration.
2) **total amount of transferred data:** total amount of data transferred to the destination host.
3) **downtime**: time interval during which the VM is stopped.

The first two metrics are of interest to data center operators in order to estimate the required resources for the live migration, whereas the downtime may affect SLAs and/or the quality of service (QoS) experienced by the users.

2.3. Live Migration Algorithms

QEMU [2] is a popular virtualization platform used both in industry and academy. The Google Compute Engine [1], for example, employs QEMU/KVM at the core of their virtualization services. With the addition of a data compression live migration technique, the latest QEMU platform now supports the following five live migration techniques.

Pre-copy [3] is the default migration algorithm for most virtualization platforms. At the heart of pre-copy lies an iterative transfer of memory pages. In each step, the modified memory pages are transferred to the destination host until the amount of modified data in the last iteration falls below a given threshold (stop-and-copy condition). The VM is then stopped, the remaining pages plus the VCPU and device state transferred, and the VM restarted on the destination host. If the network bandwidth is lower than the memory dirty rate, the algorithm does not converge and continuously sends large amounts of data over the network. The memory size, the number of modified pages in the working set, the page dirty rate, and the page transfer rate are the key features for migration cost estimation of pre-copy [8].

CPU throttling [7] is a technique enforcing convergence of the pre-copy process by deliberately decreasing the allotted CPU time of a VM in order to reduce its page dirty rate. CPU throttling can significantly degrade the performance of the workload running in the VM. The model thus requires the VM's CPU utilization as an extra parameter in addition to the features used in the estimation of the pre-copy method.

Delta compression [10] is an optimization for pre-copy that applies delta compression to modified pages during live migration. This technique may require a significant amount of additional memory to store the original memory pages for comparison with the modified ones. To estimate the performance of delta compression, the amount of modified data in written-to pages is included.

In the **data compression** optimization [5], memory pages are compressed using the `zlib` algorithm before transmission. Data compression requires a significant amount of computation time, and may thus not be a viable option if the CPU utilization on the server is high. In addition, the compression performance is highly dependent on word-level duplication of memory contents. For an accurate estimation, the degree of word-level duplication needs to be detected efficiently at runtime.

The **post-copy** algorithm [4], finally, is diametric to pre-copy in the sense that the VM is restarted on the destination host before the vast majority of the volatile state has been migrated. The algorithm immediately stops the VM on the source host, transfers only the execution context (VCPU registers and device state), and resumes the VM on the destination machine. The contents of the VM's memory are transferred from the source machine on demand and in the background. If the VM accesses a page that has not yet been transferred it incurs a large performance penalty. The main advantage of post-copy is that each memory page is transferred exactly once. The severe performance degradation of post-copy during the resume phase, however, limits its applicability in environments with strict SLAs. For the post-copy algorithm, the downtime is constant and the total migration time depends linearly on the VM's memory size and the page transfer rate.

3. Data Collection and Modeling

3.1. Workloads and Test setup

In order to identify the key features strongly correlated to the performance metrics of live migration, we collected data from a wide range of applications running inside a VM. The application set includes online transaction processing, a video streaming server and data-parallel applications representing workloads in data center environments.

The test environment consists of identical host machines each comprising an 8-core AMD FX-8300 processor and 16 GB of memory. Three distinct switched gigabit networks are available for VM service (client access), VM management (data transfer during migration), and access to shared storage. The shared storage service is provided by a NAS device. VMs are configured with 4 VCPUs and 2 GB of memory. Both the host and guest VMs run Ubuntu server 14.04 LTS. Virtualization is provided by QEMU/KVM [2].

3.2. Data collection

Table 1 lists the features collected during live migration of a VM. Features such as the page dirty rate or the working set's entropy are collected directly in the virtual machine monitor (VMM). Others, such as the number of instructions-per-second (IPS), by querying the hardware's performance monitor unit (PMU). The host provides statistics about the CPU utilization and network utilization. Data collection starts 20 seconds prior to live migration in order to detect a VM's working set. The employed data collection methods are completely transparent to the VM and do not require any modifications of the guest operating system or the workloads running inside the VM.

3.3. Support Vector Regression Model

The proposed model is based on support vector regression (SVR) [11]. During training, the estimator takes two input parameters, the vector of features and a target value. A subset of the features in Table 1 make up the feature vector, and the target value is the measured target metric we want to predict. The learning algorithm then performs regression on the training data by minimizing the error using support vectors.

The model's performance is evaluated with four different features vectors: (1) page dirty rate (DR) only, (2) DR+VM size (VMSize), (3) DR+VMSize+Working set size (WSSize), and automatically extracted features using the recursive feature elimination with cross validation (RFECV) technique. RFECV identifies the relevant features by repeatedly removing features with small weights.

4. Model Evaluation

4.1. Model Training

We use the SVM package from the *Sci-Kit Learning* Python library [9] to build and evaluate the migration models. The model is trained with the 10,000+ live migration profiles obtained by running diverse workloads with different live migration algorithms under varying network bandwidth limitations. We first classify the dataset by live migration algorithm and network bandwidth, leading to about 400 profiles per configuration. The SVM model is trained with each set separately to predict the three metrics *total migration time*, *downtime* and *total amount of transferred data*. The final model consists of 75 sub estimators, predicting the three key metrics of live migration algorithms for five different network configurations.

4.2. Prediction Accuracy

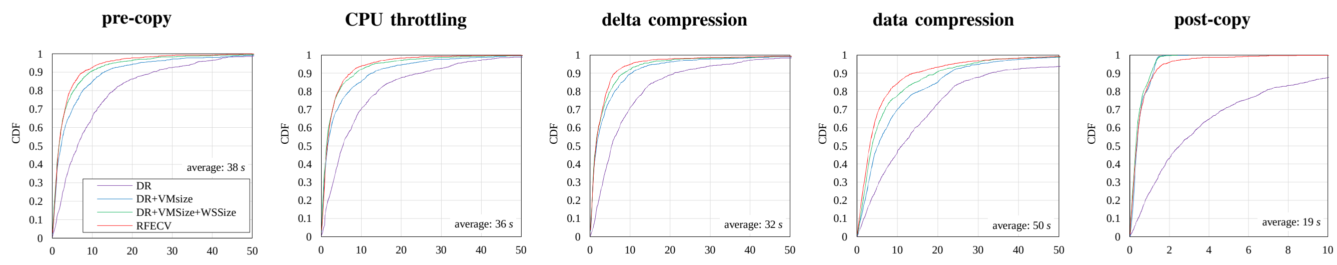
Figure 2 shows the cumulative distribution function (CDF) of the absolute prediction error for the five live migration techniques `pre-copy`, `CPU throttling`, `delta compression`, `data compression`, and `post-copy` in three rows for (a) the total migration time, (b) the downtime, and (c) the total amount of transferred data. The units of the x-axis for the three cases are seconds, milliseconds, and megabytes, respectively. Each plot shows the CDF of the absolute error for a prediction with four different input features vectors. The first, `DR`, only takes the page dirty rate as an input parameter. `DR+VMSize` is an input vector composed of the page dirty rate and the VM's memory size. `DR+VMSize+WSSize` also includes the VM's working set size. `RFECV`, finally, shows the results of the feature vector obtained by the RFECV technique.

Total Migration Time. The model shows a good prediction accuracy of the total migration time. The average relative error is below 10% for all algorithms. The 90th-percentile has an average absolute error of 7 seconds for an average total migration time of 35 seconds (Figure 2 (a)). We observe a gradual improvement of the prediction accuracy as more features are added to the models for the pre-copy-based algorithms (columns one to four). `RFECV` achieves the highest accuracy, although the difference to using only the three features `DR+VMSize+WSSize` is surprisingly small. For `post-copy` (column 5), the size of the VM's memory is the dominant factor, the additional input features included by `RFECV` even reduce accuracy due to the increased noise in the data.

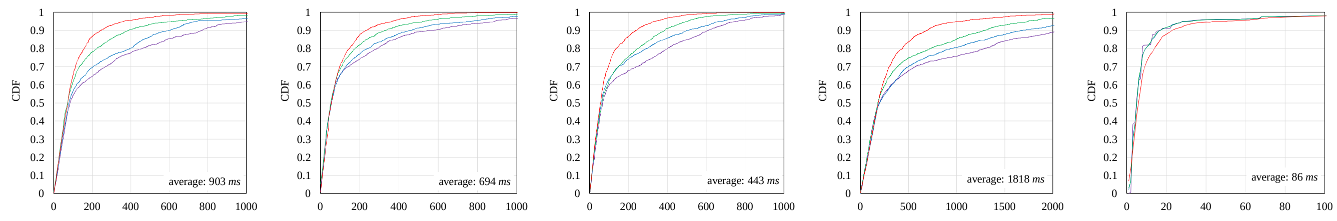
Downtime. Similar to the estimation of the total migration time, the proposed model is able to predict the downtime with a relatively small absolute error of 280ms for the 90th-percentile over all techniques. For pre-copy-based algorithms, `RFECV` again achieves the highest accuracy, yet suffers from increased noise in the `post-copy` technique. Since the absolute downtime is short, an absolute error of only 280ms translates to a large relative error of 50%.

Feature	Description	Source
Page Dirty Rate	Average page dirty rate	VMM
VM Size	Number of allocated pages to the VM	VMM
Writable Working Set Size	Size of modified pages for given period	VMM
Working Set Entropy	Shannon's entropy of the working set (byte level)	VMM
Cache Misses	Number of memory accesses not served by any caches for a given period	PMU
IPS	Ratio between the number of retired instructions and unhalted cycles	PMU
L2 Cache WB Count	L2 cache write-back count for a given period	PMU
Storage NIC Utilization	Utilization of the NIC dedicated for shared storage	host
Available CPU resource on Host	Utilization of processors in the host	host

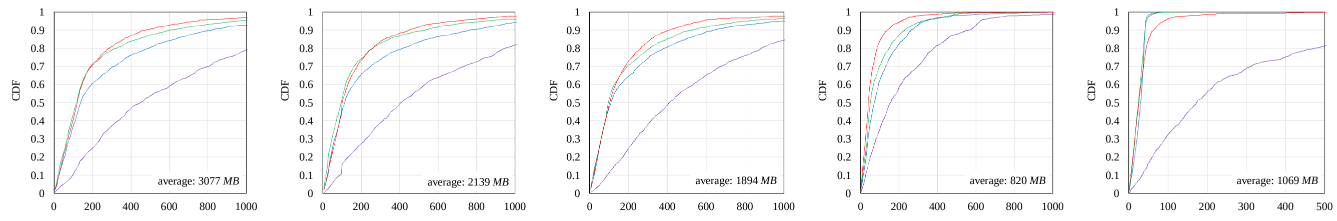
TABLE 1. LIST OF FEATURES OF THE ML MODEL



(a) total migration time (x-axis: seconds)



(b) downtime (x-axis: milliseconds)



(c) total transferred data (x-axis: megabytes)

Figure 2. Cumulative distribution function (CDF) of the absolute prediction error for the metrics (rows *a*-*c*) and live migration techniques (columns 1-5).

Total Transferred Data. The total transferred data shows a similar pattern as the total migration time. This is not an unexpected result since in most cases the total amount of transferred data can be computed by multiplying the total migration time with the page transfer rate. A notable exception is `data compression` where the high computational overhead can become the bottleneck. The 90th-percentile is 300 MB for 1640 MB of data transferred. The average relative error is only 7.3%.

5. Conclusion and Future Work

In this ongoing work, we have outlined a technique to build accurate prediction models that are able to estimate key parameters of different live migration algorithms under varying resource constraints and VM workloads. We show that with a moderate set of training data, an automated Machine Learning-based approach is able to accurately predict the total migration time, the downtime, and the total amount of data transferred for all live migration algorithms supported by QEMU/KVM.

This work will be extended as follows. Currently, all experiments were performed on lightly-loaded hosts. We are conducting a series of experiments under resource-constrained situations to better model live migration performance in the presence of hot spots. We expect that this will result in new input features to the model such as the host's available CPU load, the available memory, and network bandwidth.

In future work, we will integrate the model into a fully-automated data center management framework where it will play a key role in the VM placement and migration process with the goal of maximizing the data center's efficiency while at the same time respecting SLA constraints.

Acknowledgments

We thank Jinho Song for his advice on machine learning and Jinwoo Choi for assisting us in running the experiments. This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2012R1A1A1042938) and by the Promising-Pioneering Researcher Program through Seoul National University (SNU) in 2015. ICT at Seoul National University provided research facilities for this study.

References

[1] "Google Compute Engine," http://https://cloud.google.com/compute, 2015, online; accessed November 2015.

[2] F. Bellard, "Qemu, a fast and portable dynamic translator." in *USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 41–46.

[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-Volume 2*. USENIX Association, 2005, pp. 273–286.

[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*. ACM, 2009, pp. 51–60.

[5] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive, memory compression," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*. IEEE, 2009, pp. 1–10.

[6] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in *Proceedings of the Linux Symposium*, vol. 1, 2007, pp. 225–230.

[7] Z. Liu, W. Qu, W. Liu, and K. Li, "Xen live migration with slowdown scheduling algorithm," in *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on*. IEEE, 2010, pp. 215–221.

[8] S. Nathan, U. Bellur, and P. Kulkarni, "Towards a comprehensive performance model of virtual machine live migration," *ACM Symposium on Cloud Computing*, 2015.

[9] Scikit-learn.org, "scikit-learn: machine learning in python," 2015. [Online]. Available: http://scikit-learn.org/

[10] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth, "Evaluation of delta compression techniques for efficient live migration of large virtual machines," *ACM Sigplan Notices*, vol. 46, no. 7, pp. 111–120, 2011.

[11] V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems 9*. Citeseer, 1996.