A Machine Learning Approach to Live Migration Modeling

Changyeon Jo
Seoul National University
Seoul, Korea
changyeon@csap.snu.ac.kr

Youngsu Cho
Seoul National University
Seoul, Korea
youngsu@csap.snu.ac.kr

Bernhard Egger
Seoul National University
Seoul, Korea
bernhard@csap.snu.ac.kr

ABSTRACT

Live migration is one of the key technologies to improve data center utilization, power efficiency, and maintenance. Various live migration algorithms have been proposed; each exhibiting distinct characteristics in terms of completion time, amount of data transferred, virtual machine (VM) downtime, and VM performance degradation. To make matters worse, not only the migration algorithm but also the applications running inside the migrated VM affect the different performance metrics. With service-level agreements and operational constraints in place, choosing the optimal live migration technique has so far been an open question. In this work, we propose an adaptive machine learning-based model that is able to predict with high accuracy the key characteristics of live migration in dependence of the migration algorithm and the workload running inside the VM. We discuss the important input parameters for accurately modeling the target metrics, and describe how to profile them with little overhead. Compared to existing work, we are not only able to model all commonly used migration algorithms but also predict important metrics that have not been considered so far such as the performance degradation of the VM. In a comparison with the state-of-the-art, we show that the proposed model outperforms existing work by a factor 2 to 5.

CCS CONCEPTS

• **Computing methodologies** → **Modeling methodologies**; • **General and reference** → *Performance*; Metrics;

KEYWORDS

live migration, performance modeling, machine learning, virtualization

ACM Reference Format:
Changyeon Jo, Youngsu Cho, and Bernhard Egger. 2017. A Machine Learning Approach to Live Migration Modeling. In *Proceedings of SoCC '17, Santa Clara, CA, USA, September 24–27, 2017,* 14 pages.
https://doi.org/10.1145/3127479.3129262

SoCC '17, September 24–27, 2017, Santa Clara, CA, USA

ACM ISBN 978-1-4503-5028-0/17/09…$15.00
https://doi.org/10.1145/3127479.3129262

1 INTRODUCTION

The past decade has seen a greatly increased demand for techniques for dynamic management of resources in data centers. The goal is to minimize energy consumption while maximizing hardware resource utilization in order to reduce the operational cost and the environmental impact [40]. Virtualization is one of the key technologies for efficient data center operation [23] as it enables better utilization of resources by running multiple virtual machines on one physical host. To adapt to fluctuating workloads and dynamically optimize the resource utilization, virtual machines are *live migrated* [10, 37], i.e., moved from one physical host to another while the virtual machine (VM) keeps running. Live migration enables load balancing, consolidation, fault tolerance, and eases maintenance [32].

Live migration of a VM is a relatively costly operation as it involves sending several gigabytes of volatile state of the running VM from the source to the destination host. Over the years, a number of live migration algorithms have been proposed [10, 19–22, 28, 37, 44, 47]. As we demonstrate later in this paper, each of the algorithms exhibits different performance characteristics that not only depend on the algorithm itself but also on the state of the host system, the interconnection network, and to a larger extent on the workload running inside the VM itself.

With the major cloud platform providers Amazon, Google, and Microsoft all employing virtualization techniques in their data centers [3, 16, 31], an important problem is to select the best migration technique as a function of the operation policies, the characteristics of the workload in the VM, the state of the involved hosts, and existing service-level agreements (SLAs). There have been a significant number of attempts to model performance of live migration [1, 13, 26, 27, 34, 51, 53], however, given the different migration algorithms and vast parameter space, analytical or simple probabilistic models do not achieve satisfactory prediction accuracy. Another shortcoming of existing work is that not all commonly used live migration algorithms are modeled and that important performance metrics are missing.

In this paper, we employ machine learning (ML) techniques to obtain a versatile model that is able to accurately predict key metrics of different live migration algorithms. Given the resource usage of the physical hosts and the characteristics of the VM's workload,

the presented model predicts six key metrics of live migration (total VM migration time, total amount of data transferred, VM downtime, performance degradation of the VM, and CPU and memory usage on the physical hosts) with high accuracy. The model can be integrated into existing migration frameworks to select the best live migration algorithm for the migration of a VM.

We show what input features are relevant to model live migration and describe an efficient implementation to profile these features.

Based on the profiles of over 40,000 live migrations of VMs executing a wide variety of workloads, we employ machine learning techniques to generate the prediction model. The generated model outperforms the state-of-the-art in live migration modeling by a factor two to five in terms of its prediction accuracy. By virtue of the automatic approach, new migration algorithms and profile features can be easily added rendering the proposed procedure extensible and flexible.

In summary, the main contributions of this work are as follows:

- We show that among the existing live migration algorithms there is no one-size-fits-all technique. Choosing the 'correct' algorithm can significantly improve resource efficiency and reduce SLA violations.
- We present a machine learning-based modeling approach to predict important performance metrics of live migration algorithms for a given VM. The presented work currently is the only approach that can predict several target metrics for all commonly used live migration algorithms in a flexible and automated manner.
- The model achieves a high prediction accuracy on all target metrics. It is the first to predict the performance degradation of a VM under migration and outperforms the state-of-the-art by a factor 2 for the total migration time and a factor 5 for the downtime.
- We demonstrate how employing the model in an existing live migration framework to automatically select the proper live migration algorithm can significantly reduce the total number of SLA violations and improve resource utilization.

The remainder of this work is organized as follows. Section 2 provides the necessary background on live migration. Section 3 demonstrates selecting the optimal technique given SLAs and constraints is an important and non-trivial problem. Sections 4 and 5 introduce the ML model and the model parameters and discuss profiling. The experimental setup and the evaluation are presented in Sections 6-8. Section 9 discusses related work, and Section 10, finally, concludes this paper.

2 BACKGROUND

2.1 Live Migration

Migrating a running VM requires moving its entire state from one physical host to another. In intra-datacenter migration, permanent storage is typically provided by network-attached storage (NAS) and does not need to be moved. The volatile state of a VM comprises its memory and the state of the virtual CPUs (VCPUs) and devices. The memory, in the order of gigabytes, constitutes by far the largest part of the volatile state of a VM.

The simplest way of migrating a VM is to completely stop the machine on the source host, transfer the entire state to the destination, and then restart the VM on the destination host. The long downtime of this *stop-and-copy* technique is impractical for most environments; it is thus common practice to migrate the VMs while running to minimize the perceptible effects of the migration [17].

We distinguish the following phases of the live migration process (Figure 1):

(1) **Profile**. The VM monitor (VMM) profiles the VM and the system state of the physical host. Profiling may cause a small performance reduction of the VM.
(2) **Prepare**. Live migration is initiated. The VMM puts the VM into a managed mode suitable for live migration, which typically results in slightly reduced performance in the VM. The source host starts sending (parts of) the volatile state to the destination host.
(3) **Stop**. The VM is stopped both on the source and the destination host and thus not available to the user.
(4) **Resume**. The VM is restarted on the destination host. The parts of the volatile state that are still missing are fetched from the source host. VM performance may still be reduced until the resume phase ends.

Not all phases are present in all migration algorithms. The stop-and-copy technique, for example, comprises a long stop phase with very short preparation and resume phases. The profile phase is specific to modeling approaches. Existing live migration algorithms are very sensitive to the workload running in the VM and the state of the physical host. During the profile phase, the VMM gathers the necessary parameters affecting live migration performance and makes them available to the model. The duration and the overhead of this phase are discussed in Section 7.

2.2 Live Migration Metrics

In this work, we compare and predict the following metrics for the different live migration algorithms.

(1) **Total migration time (TT)**: time period from initiation to completion of the live migration.
(2) **Total amount of transferred data (TD)**: total amount of data transferred to the destination host.
(3) **Downtime (DT)**: time interval during which the VM is stopped and unavailable to the user.
(4) **Performance degradation (PERF)**: reduced performance of the VM during live migration measured in the number of executed instructions per second (IPS).
(5) **Host CPU utilization (CPU)**: total CPU resources consumed by the VM during live migration on the source host.
(6) **Host memory utilization (MEM)**: memory resources consumed by the VM during live migration on the source host.

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

2.3 Live Migration Algorithms

Live migration algorithms can be classified by the point in time when the volatile state of the VM is copied to the destination host: during the prepare phase (pre-copy), during the stop phase (stop-and-copy), or in the resume phase (post-copy). Other algorithms are hybrids of these three basic approaches or optimize a certain aspect of a method. We model all five techniques that are supported by the major virtualization environments [4, 46]:

Pre-copy (PRE) [10] iteratively transfers the volatile state of a VM during the *prepare* phase. In each iteration, memory pages that have been modified by the running VM since the previous

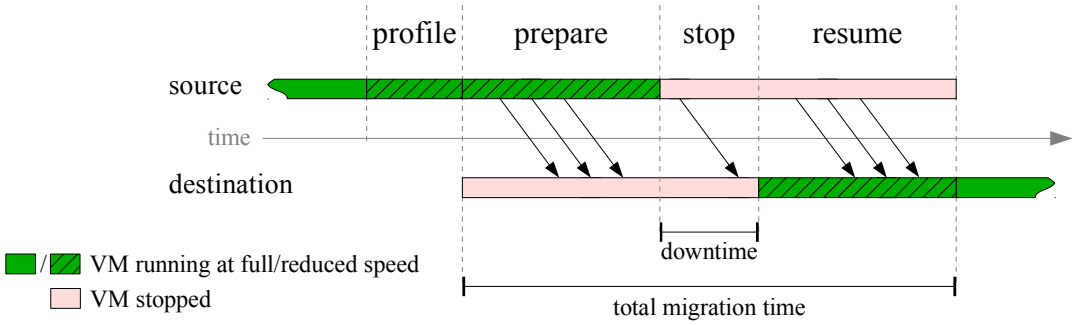


Figure 1: Phases and metrics of live migration

iteration are sent to the destination host. When the number of these dirtied pages falls below a given threshold, the VM is stopped. The remaining dirty pages and the VCPU plus device state is transferred, and the VM is restarted on the destination host. A limitation of this algorithm is that it never converges if the memory dirty rate is higher than the available network bandwidth. Thus, if the number of dirtied pages per iteration remains stable or a certain maximum number of iterations has been reached, a stop condition is activated to prevent such non-terminating migrations. Pre-copy is the default migration algorithm for most virtualization platforms.

CPU throttling (THR) [28] is a technique enforcing convergence of the pre-copy process by deliberately decreasing the allotted CPU time of a VM to reduce its page dirty rate. CPU throttling can significantly degrade the performance of the workload running in the VM.

Delta compression (DLTC) [44] is another optimization for pre-copy that applies delta compression to partially modified pages during live migration. This technique may require a significant amount of additional memory to store memory pages for future delta computations.

The **data compression (DTC)** optimization [20] compresses memory pages before transmission. This technique requires a significant amount of additional computation resources and may thus not be a viable option if the CPU utilization on the host is high.

The **post-copy (POST)** algorithm [19] is diametrical to pre-copy in the sense that the VM is immediately restarted on the destination host after transferring only the execution context (VCPU registers and device states). The memory contents are transferred by a background process. Accesses to not-yet-copied pages trigger a page fault in the VMM that then fetches the accessed page on demand. These page faults can cause a prohibitive performance penalty in the VM during the resume phase. Post-copy is attractive because it has a short and constant downtime and transfers each memory page only once. The potentially severe performance degradation during the resume phase, however, limits its applicability in environments with strict SLAs.

Note that CPU throttling and memory compression are independent optimizations that can, in principle, be applied to both the pre- and post-copy algorithm. In this work, we employ the techniques as provided by the virtualization framework: the underlying algorithm for THR, DLTC, and DTC is pre-copy and no optimizations are applied to post-copy.

3 MOTIVATION

IaaS (Infrastructure as a Service) platforms typically offer various service level agreements to customers such as guaranteeing a certain availability or a minimal performance of the VM [3, 16]. In addition to guaranteeing SLAs, operators are interested in minimizing the impact of live migration on the data center by constraining the amount of time, network bandwidth, or host CPU and memory usage the operation is allowed to consume. Given that all major virtualization solutions such as QEMU/KVM [4], VMware [48], or Xen [46] support various live migration algorithms that are used in production by data center operators [17], an important question for efficient use of live migration is which algorithm to select in a given situation.

The performance of live migration algorithms shows not only a strong variation between the different algorithms but also heavily depends on the workloads running inside the VM. Figure 2 displays the results of migrating 512 VMs running individual workloads for the six modeled metrics with the five live migration algorithms. The sixth bar, labeled OPT, represents an oracle technique that picks the best algorithm with respect to the given metric for each migration. The whiskers depict the standard deviation of the individual VM migrations. All numbers are normalized to the average performance of pre-copy (PRE), and in all graphs except performance lower is better. Note that the additional CPU load caused by live migration is negative with respect to pre-copy for throttling (THR) and post-copy (POST). The reduced performance of the VM caused by throttling (for THR) or page faults (for PRE) on the destination host leads to an overall reduced CPU load on the source and destination host.

The downtime exhibits the strongest variation between the different algorithms with post-copy (POST) being over 1,000 times faster than data compression (DTC). POST also shows the best performance for the total migration time and the total transferred traffic, outperforming all other algorithms by at least factor two. With respect to performance degradation observed in the VM, however, POST performs worst with an average performance degradation of 30%. Looking at the variance within the different algorithms for the 512 migrations, we observe that each metric is influenced heavily by the executed workload in the VM and the resource availability of the physical host. POST, seemingly the ideal choice when considering only the total time, the downtime, and the total traffic of the migration, can experience extreme performance degradation of over 90% in certain cases.

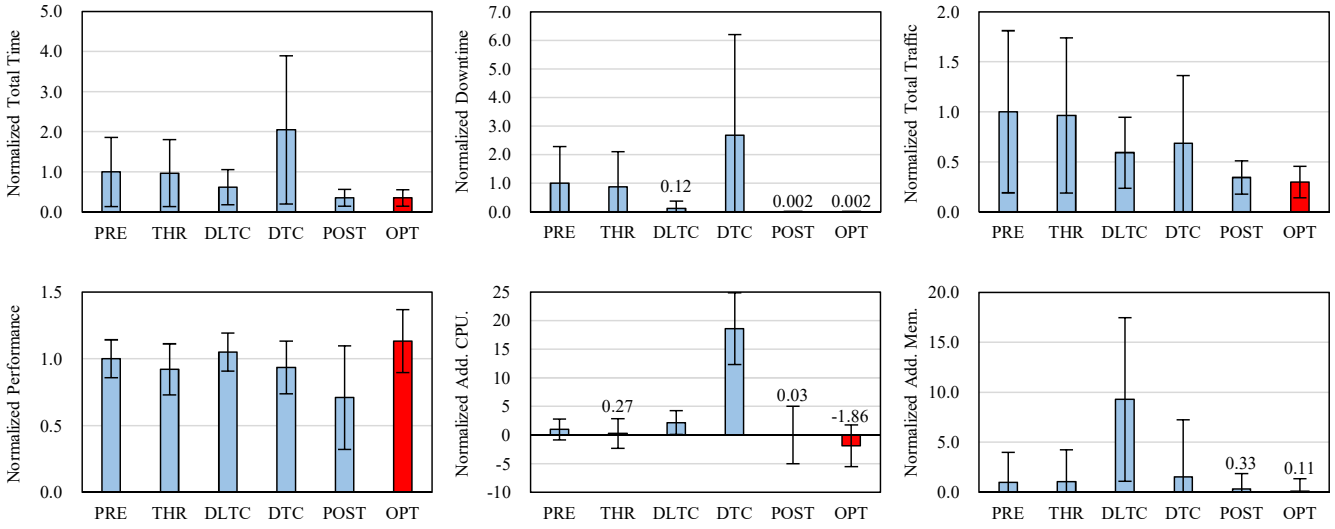


Figure 2: Relative benefit of the optimization techniques

3.1 Meeting Complex SLA Requirements

Given that the performance of the different metrics of live migration depends on the workload running inside the VM as well as the resource state of the host (this includes co-located VMs), the question arises how to know which algorithm is least likely to violate the SLAs while at the same time satisfy other operational constraints of a data center. The additional system resources consumed by a live migration algorithm should not cause a system slowdown or affect other VMs; this is especially true when live migration is used to eliminate hot spots: the source host is already overloaded, any additional consumption of resources must be minimized. The problem becomes more complex if SLAs are in place that guarantee, for example, no more than a 15% performance degradation.

To the best of our knowledge, there is currently no model available that can predict migration metrics of different live migration algorithms under consideration of SLA and user constraints. Also, no models exist that can predict the performance degradation of a VM and the additional consumption of system resources during migration. Some analytical models predicting the total time, the downtime, and the total traffic of migration for the original pre-copy algorithm have been proposed for Xen[1, 13, 26, 27, 34, 51, 53] and KVM[2, 25, 38, 54]. Extending these analytic models for other techniques and or metrics is impractical due to the large number of parameters that need to be considered.

3.2 Learning from Big Data

Modern data centers run hundreds of thousands of servers to accommodate millions of users worldwide. The servers in a data center generate lots of information from VM performance logs to data from hardware sensors. This opens new possibilities for data center management systems. Ferdaus *et al.* [15] were of the first to employ machine-learning to predict the power usage effectiveness of data centers by using 200,000 training samples collected in Google data centers over a period of two years. The model considers 43 distinct input parameters; building an analytic model considering such a large number of parameters would be impractical at the least.

Predicting the performance of live migration given the state of the VM and the underlying physical host is even less suited for analytical analysis. Not only are there various live migration algorithms to choose from, but there are several metrics of interest for each technique. For n algorithms and m metrics, one would have to construct $n \times m$ analytical models that each depend on a different set of parameters. Machine learning is an ideal technique to automatically generate models for the different metrics and the available live migration algorithms using profiling data gathered in data centers. Such a setup also allows for easy addition of new algorithms or metrics.

4 MODELING LIVE MIGRATION

One of the keys to success in machine learning is designing a good input feature set, in other words, a good representation of the input data to the model. Feature design requires domain specific knowledge and a careful analysis of what input features are likely to be correlated to the modeled output parameters. Since live migration requires copying the entire memory of a VM from one host to another, the size of the VM's memory and the network bandwidth are highly correlated to the total migration time and the amount of transferred data. For iterative algorithms based on pre-copy that repeatedly copy modified pages to the destination host, the rate with which the VM is dirtying pages and the working set of the VM are important. Algorithms that compress data profit from knowing the entropy to estimate the compressibility of data. Section 4.1 explains the process of feature selection in more detail.

Our model predicts six performance metrics (Section 2.2) for each of the five live migration algorithms (Section 2.3). We construct a dedicated sub-model for each combination of $\langle algorithm, metric \rangle$

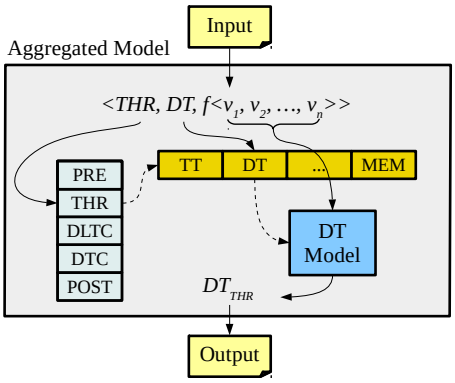


Figure 3: Overview of the model

resulting in a total of 30 sub-models. Each model is trained separately. The same input vector comprising of properties that represent the state of the VM, the source, and the destination host (Table 1) are fed to each submodel. The output of a model is a prediction for a given algorithm of the metric it was trained for.

For convenience, the sub-models are logically aggregated into a universal model as shown in Figure 3. The aggregated model takes the live migration algorithm, the desired metric, and the profiled input parameters as its input $\langle algorithm, metric, input\ vector \rangle$. Internally, *algorithm* and *metric* are used to select the appropriate sub-model, which is then fed the *inputvector*. The output of the selected sub-model equals the output of the aggregated model.

The remainder of this section discusses model feature selection and model generation. An evaluation of the importance of the selected features and the results of the modeling process are provided in Section 7.

4.1 Model Parameters

The selected input features are listed in Table 1. The first two columns describe the feature, and the third column shows where the parameter is profiled. The six composed features at the end of the table are computed using other features. The model parameters are chosen such that in their entirety they cover all aspects of the involved systems that can have an impact on the estimated metrics. Since live migration requires a transfer of the VM's memory to a destination host over a network, the size of the allocated memory, `VM.Size`, and the available network bandwidth for the transfer `PTR` are important features for all models. Note that `VM.Size` denotes the size of the actually allocated memory to the VM, not the assigned maximal memory size. Similarly, the CPU utilization `CPU.UTIL`, the number of retired instructions per second `IPS`, and the network utilization `NET.UTIL` of a VM are necessary to capture and estimate the performance degradation during live migration. Pre-copy-based algorithms that iteratively copy dirtied memory to the destination host, require the page dirty rate `PDR` and the working set size `WSS`. Note the relationship between the `PDR` and the `WSS`: the page dirty rate represents the number of pages dirtied in one profiling period, while the working set represents all pages dirtied during the entire profiling phase. For techniques involving compression of data, the entropy of both the working set and the non-working set, `WSE` and `NWSE`, plus the CPU and memory utilization of source and destination host, `SRC|DST.CPU` and `SRC|DST.MEM`, are included. To estimate the effectiveness of delta compression, the number of modified words per page `MWPP` is profiled.

Composed features represent special combinations of regular features that allow for a better prediction. For pre-copy-based algorithms, the **weighted relative page transfer rate** is a combined parameter computed as

$$R.PTR = max(WSS, WSS * (PDR/PTR)^2)$$

to reflect the fact that if the page dirty rate is higher than the page transfer rate the pre-copy algorithm does not converge and the remaining data will be transferred after the stop condition has been activated (Section 2.3). The **throttling benefit** given by

$$THR.BF = PDR * CPU.UTIL$$

enables the models to better account for the fact that the CPU throttling technique is ineffective for workloads that dirty many different pages with little CPU utilization. The run-length compression algorithm employed by the delta compression technique is more effective the fewer changed words exist in the delta of two pages. We approximate this **benefit of delta compression** by considering the working set size, the page dirty rate, and the number of modified words per page as follows

$$DLTC.BF = WSS * (\#words\ per\ page/MWPP)$$

For the data compression technique, the entropy of uncompressed data determines its effectiveness. We compute the **expected size of the compressed (non-)working set**, `E.WSS|NWSS`, by multiplying the size of the (non-)working set by the entropy

$$E.WSS|NWSS = WSS|NWSS * WSE|NWSE$$

Section 7.2 analyzes whether the selected features are both relevant and sufficient for accurate modeling.

4.2 Model Generation

The large number of features and their often inconspicuous but significant effects on a target metric render the task of manually generating separate models for all combinations of techniques and target metrics a difficult one. The different sub-models are generated automatically using supervised machine learning techniques. We employ three different techniques: linear regression, support vector regression (SVR) with non-linear kernels, and SVR with bootstrap aggregation. Linear regression is a common and simple regression technique fitting the given samples using a straight line with minimal error. Linear regression is a reasonable choice if there exists a clear linear relation between the data set and the target value. SVR [42] is a regression technique for complex data points that exhibit a non-linear relationship between the features and the target value. We use a radial basis function for the loss function with a penalty parameter $C = 10.0$, and the input features are standardized. The third modeling technique uses bootstrap aggregation [7] of the SVR model to improve the accuracy of the predictions. Bootstrap aggregation, also known as bagging, constructs multiple submodels with a subset of the full dataset and overfits the model to the dataset. After the submodel training, the average prediction of all submodels is used as the final value. Results are discussed in Section 7.

Feature	Description	Source
VM size (VM.Size)	Number of allocated pages to the VM	VMM
Page dirty rate (PDR)	Average number of pages modified per second	VMM
Working set size (WSS)	Number of modified pages during profiling period	VMM
Working set entropy (WSE)	Entropy of working set memory	VMM
Non-working set entropy (NWSE)	Entropy of non-working set memory	VMM
Modified words per page (MWPP)	Number of modified words in modified pages	VMM
Instructions per second (IPS)	Number of retired instructions per second	Source host
Page transfer rate (PTR)	Reserved bandwidth for live migration	Source host
CPU utilization of VM (CPU.UTIL)	CPU utilization of the VM process	Source host
Network utilization of VM (NET.UTIL)	Network utilization of the VM process	Source host
CPU utilization on host (SRC|DST.CPU)	CPU core utilization on the involved hosts	Src + dest host
Memory utilization on host (SRC|DST.MEM)	Memory utilization on the involved hosts	Src + dest host
Weighted relative page transfer rate (R.PTR)	Weighted relative page transfer rate to page dirty rate	Composed
Non-working set size (NWSS)	Number of not modified pages during profiling period	Composed
Benefit of delta compression (DLTC.BF)	Expected benefit of delta compression technique	Composed
Benefit of CPU throttling (THR.BF)	Expected benefit of CPU throttling technique	Composed
Compressed size of WSS (E.WSS)	Expected size of WSS after compression	Composed
Compressed size of NWSS (E.NWSS)	Expected size of NWSS after compression	Composed

Table 1: The 20 input features of the ML model

5 FEATURE PROFILING

In order for the model to make a prediction, the features listed in Table 1 need to be available. For performance reasons, not all features are measured continuously but gathered during a short period of profiling before the actual live migration is started (Section 2.1). The profiling period is further divided up into intervals. Data is gathered during the entire profiling period but measured at the end of each interval in order to detect variations in the individual features.

5.1 Lightweight Profiling

Some of the features are obtained easily because they are continuously monitored by the operating system (OS) or the VMM. Others, such as memory access patterns or entropy of data in memory, are notoriously difficult to profile without a high computational overhead [35]. In the following, we discuss our implementation for efficiently profiling the required values.

Features provided by the OS/VMM. The following features are either provided directly by the VMM, the OS or can be obtained easily through querying the CPU's performance monitor: `VM.Size`, `PTR`, `CPU.UTIL`, `NET.UTIL`, `IPS`, `SRC|DST.CPU`, and `SRC|DST.MEM`. Obtaining these features causes no significant overhead.

Page dirty rate and working set. To compute the page dirty rate, the write accesses of a VM to its memory need to be monitored. Thanks to the hardware support of modern CPUs, the overhead of profiling the `PDR` is minimal. The `PDR` is measured after every profiling interval. The working set is the union of all `PDRs` and computed at the end of the profiling period.

Modified words per page. Computing the `MWPP` for all modified pages is too compute-intensive, we thus compute the values for a subset only. Experiments show that a sample of 1/32 of all modified pages provides sufficient accuracy at a moderate computational overhead of 1% additional load on one processor core.

Data entropy. The entropy of data stored in the memory is computed at the end of the profiling period by building a frequency histogram at byte granularity. Computing the entropy for the entire VM memory would be too costly (the entropy is computed for the working set and the non-working set), we revert to sampling every 32^{nd} page which takes 120ms for 2GB of memory on our test machines. This overhead is only incurred once since the entropy is only computed at the end of the profiling period.

5.2 Profiling Period and Interval

In our approach, we profile the VM for a few seconds before starting the migration (Section 2.1). As a consequence, profiling can delay the beginning of the migration and should thus be as short as possible. In related work, Nathan *et al.* [34] propose a maximum profiling period of `VM.Size/PTR`, i.e., dividing the VM memory size by the page transfer rate. For a 2GB VM transferred at 1 gigabit per second, this results in a rather long profiling period of 16 seconds. We thus employ *adaptive profiling*: data is collected at periodic intervals of one second with a minimal and maximal total profiling period of 3 and 20 seconds, respectively. We stop adaptive profiling using a simple heuristic: we monitor the increase of the working set size and terminate profiling when the working set size does not increase by more than 1 MB in the last interval. The effect of adaptive profiling on prediction accuracy is discussed in Section 7.

5.3 Profiling Overhead

Figure 4 shows the relative performance degradation (IPS) in the VM caused by profiling in dependence of the host's CPU load. Performance drops by 1-3% up to a utilization of 80% and up to 5% for a CPU utilization up to 97%.

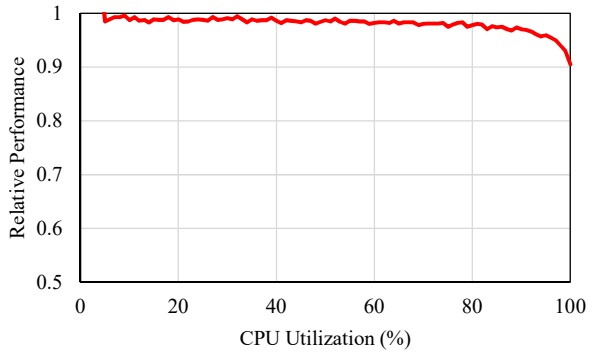


Figure 4: VM performance degradation caused by profiling in dependence of host CPU utilization

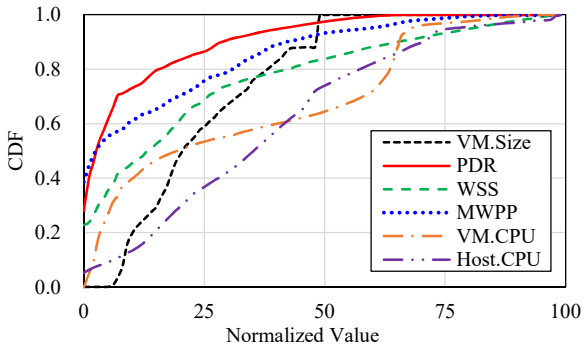


Figure 5: Statistics of the dataset used for the evaluation

6 EXPERIMENTAL SETUP

We now present an in-depth analysis of the model and show how it can be applied in data center to reduce the number of SLA violations caused by live migration.

6.1 Host and VM Setup

VMs with 1-2 VCPUs and up to 2GB of memory are deployed on and live migrated between heterogeneous machines comprising Intel Skylake i5-6600 quad-core processors with a varying clock rate and 8-32GB of memory. The machines are connected by three dedicated 1Gbit networks for shared storage, public networking, and migration traffic. Ubuntu server 14.04 LTS is installed both on the host machines and inside the VMs. The host machines run QEMU VMM version 2.3.5 with additional profiling capabilities as discussed in the previous section.

6.2 Workloads

We use 37 workloads from benchmark suites and applications representing real-world loads. The workloads are taken from the following benchmark suites: SPECWeb [43] simulates a web server hosting banking and e-commerce services. OLTPBench [14] is a database application for online transaction processing. Memcached [30] implements an in-memory key-value store, Dacapo [6] is a set of java applications, and PARSEC [5] contains a set of emerging multi-threaded workloads. `Bzip` is used as a compute- and data-intensive application, and `mplayer` [33] represents a multimedia workload. A synthetic random workload generator is used to generate a wider range of datapoints for model training. The workload generator can generate different CPU, I/O, and memory usage patterns, allowing us to control the page dirty rate, the working set size, the number of memory writes per page, or the data entropy. We employ the workload generator in random mode to produce memory dirty rates ranging from 15 to 1024 MB/s with working set sizes from 128 to 1536 MB, memory writes strides of 1 to 512 bytes, and a data entropy from 0.1 to 1.0.

6.3 Building the Data Set

The training data for the machine learning models has been generated over a period of several months by live migrating a total of over 40,000 VMs with the different live migration algorithms. For all migrations, the profiled model parameters, the values of the six target metrics, and other data for future work were recorded in a database. On each physical host, up to four VMs are co-located and compete for the shared resources. Each VM runs either one of the application benchmarks or the synthetic workload generator. A random VM of a random source host is selected and live migrated to one of the destination hosts. The available bandwidth of the migration network is set to a random value between 50 and 125MB/s to emulate a resource-constrained environment. In this work, we have limited ourselves to one concurrent live migration per host. The cumulative distribution function (CDF) of five selected features shown in Figure 5 demonstrate that the data covers a wide range of the parameter space.

7 MODEL EVALUATION

The submodels for the prediction metrics are trained and tested with the *sci-kit learn v0.17* [41] toolkit. Training and test data is generated using 10-fold cross-validation: the data set is first split into 10 equal-sized subsets. Each subset serves as the test set once while the union of the remaining nine forms the training data set. The reported values represent the average of the 10 evaluations.

7.1 Target Metric Prediction Accuracy

We report the prediction accuracy for three different regression techniques (linear regression, SVR with non-linear kernels, and SVR with bootstrap aggregation) for the five live-migration algorithms `PRE`, `THR`, `DLTC`, `DTC`, and `POST`, and the six target metrics. The results are shown in Table 2. The Mean Absolute Error (MAE) represents the average divergence of the predicted value to the actual value in absolute units of the metric (ms, MB, or %) while the Mean Relative Error (MRE) displays the average relative error of the prediction.

Linear regression, shown in the first six rows, does not achieve satisfactory accuracy: the average prediction error exceeds 10% for all algorithms. The main reason of the high error comes from the complex correlation of the features. A naive approach cannot capture the complexities and fails to train the model successfully.

The results of SVR are shown in the second six rows of Table 2. Compared to linear regression, the overall accuracy of the models has increased significantly. Especially target metrics that show a

Model	Target Metric	PRE		THR		DLTC		DTC		POST	
		MRE	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE	MAE
Linear regression	Total Time (ms)	0.16	3127	0.16	3043	0.17	2412	0.22	7758	0.12	1206
	Downtime (ms)	0.27	152	0.50	239	1.07	120	0.84	559	0.52	1
	Total Traffic (MB)	0.03	59.0	0.05	86.0	0.08	105.4	0.16	136.9	0.00	0.7
	Performance (%)	0.05	4.3	0.07	5.9	0.04	3.5	0.08	6.7	0.24	14.4
	CPU (%)	0.65	3.6	0.63	4.5	0.86	4.1	0.21	8.9	0.42	10.3
	MEM (MB)	4.04	37.2	4.00	35.6	0.40	43.8	3.60	55.9	1.45	22.1
SVR	Total Time (ms)	0.06	1197	0.06	1150	0.05	755	0.11	3864	0.03	289
	Downtime (ms)	0.23	128	0.28	136	0.35	40	0.50	333	0.29	1
	Total Traffic (MB)	0.06	94.4	0.06	94.4	0.04	56.2	0.09	78.4	0.02	18.5
	Performance (%)	0.04	3.5	0.05	3.7	0.03	3.1	0.05	4.2	0.11	6.9
	CPU (%)	0.50	2.8	0.42	3.0	0.61	2.9	0.10	4.1	0.19	4.5
	MEM (MB)	1.94	17.9	1.94	17.3	0.32	35.5	1.87	29.0	0.76	11.6
SVR with bagging	Total Time (ms)	0.06	1053	0.06	1017	0.05	726	0.10	3322	0.03	309
	Downtime (ms)	0.17	96	0.23	109	0.32	36	0.43	285	0.26	1
	Total Traffic (MB)	0.04	70.7	0.04	72.5	0.03	42.3	0.07	64.6	0.02	16.9
	Performance (%)	0.03	3.0	0.04	3.5	0.03	2.8	0.05	3.8	0.11	6.4
	CPU (%)	0.48	2.7	0.40	2.9	0.59	2.8	0.10	4.3	0.18	4.5
	MEM (MB)	1.79	16.4	1.74	15.5	0.27	29.3	1.67	25.9	0.71	10.9

Table 2: Accuracy of the different ML algorithms for the five live migration algorithms and the six target metrics (MRE: geometric mean relative error, MAE: geometric mean absolute error)

non-linear dependency based on several input features show big improvements. The relative prediction error of the downtime of the delta compression technique, for example, is reduced from 107% with linear regression to 35% with SVR. In certain cases, the relative accuracy decreases by a few percent, for example, the prediction of the total traffic with an error of only 3% for linear regression and 6% for SVR for pre-copy. Such cases show a relatively simple linear relationship between input features and the target value. The complex kernels of SVR fail to achieve the same results, however, the drop in accuracy is never severe.

The last six rows of Table 2 shows the results for SVR with bootstrap aggregation. It is known [7] that in general bagging outperforms single models. We apply bagging to our dataset with 64 sub-models that use 90% of the full dataset and 80% of the features. The results show that SVR with bagging achieves high prediction accuracy for most metrics and clearly outperforms the other two techniques.

Certain metrics show notoriously high relative errors such as, for example, the prediction of the use of memory resources on the source host with up to 179% of error with SVR+bagging for pre-copy. The reason for the high errors of CPU and memory utilization on the source host is that these metric include the resource consumption of the VM itself which can vary greatly depending on the workload executed in the VM. We notice, however, that the absolute errors are reasonably small in all such cases: up to 2.8% for the CPU load for delta compression (relative error of 59%) and 16.4 MB for the memory utilization with pre-copy (relative error of 179%). In future work, we will separate the additional CPU and memory resources consumed by the VM from those caused by live migration.

Algo.	TT	DT	TD	PERF	CPU	MEM
PRE	0.97	0.98	0.98	0.20	0.20	0.79
THR	0.98	0.98	0.98	0.51	0.49	0.80
DLTC	0.95	0.86	0.95	0.16	0.36	0.88
DTC	0.95	0.96	0.98	0.23	0.80	0.70
POST	0.99	0.08	1.00	0.52	0.71	0.85

Table 3: Aggregated CoDs of the input features

7.2 Completeness of Input Features

An important question is whether the selected features include all relevant parameters. Table 3 shows the coefficient of determination (CoD, or R^2) [8] of the 20 features for each algorithm and metric with the SVR model. The coefficient of determination measures how well the prediction fits the measured value; an R^2 of 1 indicates a perfect fit of the prediction to the measured target value. We observe a high correlation for most of the metrics indicating that the selected features are sufficient to make an accurate prediction. Low R^2 values, such as for 0.08 for the downtime with post-copy are caused by an (almost) constant target value; even a small divergence in the predicted values then causes a low R^2 value. Note that the relative and absolute error of the downtime for post-copy are 26% and only 1ms, respectively. Similarly, the overall CoD for the performance metric is quite low because the performance degradation is usually close to zero except for CPU-throttling and post-copy, i.e., there is little impact on the target value by the feature values.

Algorithm	Learning Time (s)	Prediction Time (ms)
Linear	8.0	0.74
SVR	239.0	5.11
SVR.Bagg	6617.7	188.63

Table 4: Learning and prediction overhead of the model

7.3 Learning and Prediction Overhead

The time required to train and predict values with the proposed model are shown in Table 4. The learning time includes the total CPU time required to train the 30 sub-models for the respective machine learning technique on a modern desktop. The prediction time is the total CPU time required to predict all target metrics for all live migration algorithms, i.e., the total time to evaluate each of the 30 sub-models once. We observe that training can take some time, especially in the case of SVR with bagging. The prediction time is sufficiently low not to have an impact on the algorithm selection, especially in comparison to the entire duration of the live migration (several tens of seconds). In a data center setting, re-training of the models can be desirable have the models reflect the current set of workloads running on its VMs. A computational overhead of a few CPU hours every few days seems acceptable.

7.4 Data Set Size vs Model Accuracy

Machine learning requires a sufficient number of samples to accurately train a model. The training set used in this work comprises data from over 40,000 migrations in total or about 8,000 for each of the five live migration algorithms. Figure 6 shows the progression of the mean relative and absolute error for selected metrics in dependence of the number of training samples.

The analysis reveals that prediction accuracy increases rapidly, after about 2,000 training samples improvements are becoming small. This data suggests that the set of training data is sufficient for the target metrics and VM workloads used in our setup.

7.5 Adaptive Termination of Profiling

Here we analyze the effect of the profiling duration on prediction accuracy. We compare the recommended profiling period [34] with our shorter, adaptive profiling as described in Section 5.2. Using the proposed heuristics, the average profiling period is reduced from 20 to 7 seconds with no noticeable loss in prediction accuracy: with adaptive profiling, the average MRE is only 0.36% higher compared to full profiling.

7.6 Comparison to the State-of-the-Art

The current state-of-the-art in live migration performance modeling is the work presented by Nathan *et al.* in 2015 [34]. In that work, the authors analyze twelve existing performance models for pre-copy in depth and propose a new analytic model for pre-copy that outperforms all existing models by a significant margin. Nathan *et al.*'s work provides an analytical model only for the original pre-copy live migration algorithm and three target metrics: total migration time, downtime, and total amount of transferred data.

	Total Time (s)		Downtime (s)		Total Traffic (MB)	
	Nathan	Ours	Nathan	Ours	Nathan	Ours
MAE	4.01	2.19	0.86	0.24	330.1	178.2
90th.MAE	10.76	4.26	2.37	0.47	993.3	340.8

Table 5: Comparison of our model (Ours) to the state-of-the-art by Nathan *et al.* [34] for pre-copy and total time, downtime, and total traffic. The comparison is limited to pre-copy and the above three target metrics because [34] does not support other live migration algorithms or target metrics.

Our comparison with their work is thus limited to the pre-copy algorithm and the three metrics.

Table 5 shows the overall mean absolute error and the MAE of the 90^{th} percentile. The proposed machine-learning based models outperform the manually-generated analytical models by a factor 2 to 4 for the overall MAE and, more importantly, also show a significantly smaller error for the 90^{th} percentile. Furthermore, the presented machine learning-approach is not specifically tailored towards pre-copy or the above three metrics; the same aggregated model also supports an additional four live migration techniques and three prediction metrics, clearly demonstrating the advantages of machine learning when applied to this domain.

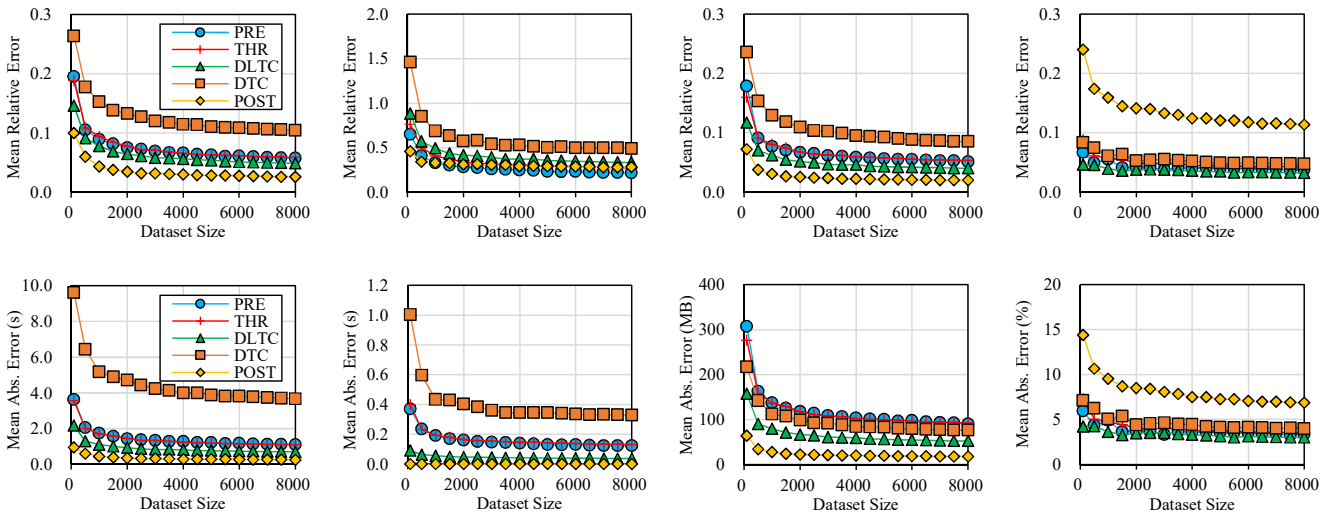
8 MODEL-GUIDED VM MIGRATION

In this section, we show one possible application of the presented model. We have integrated the proposed model into a VM live migration framework that automatically migrates VMs for load balancing or consolidation. The model is used to select the best live migration technique for a given workload under various operator- and user-provided SLA constraints. The goal is to maximize the benefit with regards to the defined objectives while meeting SLA constraints. These objectives and constraints can be defined in sophisticated ways to reflect the requirements of a data center.

The framework is evaluated with eight different policies as shown in Table 6, ranging from simple policies with a single objective (`A` to `D`) to more complex policies with SLA constraints. For `Policy E`, for example, the objective is to minimize the total time of migration while keeping the downtime of the VM below 3s and maintaining a performance of at least 85% in the VM.

8.1 Live Migration Algorithm Selection

To select the best live migration algorithm for a given policy, we first eliminate all algorithms that are expected to violate the given SLA constraints based on the model's predictions. If more than one algorithm remains, the one that maximizes the given objective is chosen in a second step. We do not allow for a migration not to happen, that is, if all algorithms have been eliminated in the first step, in the second step we choose the one that minimizes the *aggregated relative SLA violation error*. This metric is computed by summing up the relative errors of the expected target values and the given constraints. For example, if the model predicts a downtime of $3.1s$ and the given downtime constraint is $3s$, the relative error is $abs(1 - (3.1/3)) = 0.03$. We compute this SLA violation error for all algorithms and all SLA constraints and select the algorithm that is expected to yield the minimum total SLA error.



(a) Total Time (b) Downtime (c) Total Traffic (d) Performance

Figure 6: Impact of dataset size to the model accuracy

	Description	Policy							
		A	B	C	D	E	F	G	H
Objective	Minimize the total time	✓				✓			
	Minimize the downtime		✓				✓	✓	
	Minimize the total transferred data			✓					✓
	Maximize the relative performance				✓				
SLA	Keep the downtime less than 3.0s					✓			✓
	Keep the relative performance more than 85%					✓	✓	✓	✓
	Keep the additional memory usage less than 500MB							✓	

Table 6: Policies defining an objective and SLAs

8.2 Evaluation of Guided Migration

To compare guided migration against a fixed migration algorithm and an oracle selection, we migrate 500 VMs according to a predefined schedule. The schedule defines the VM, the migration time, and the destination host. Guided migration selects the algorithm as outlined in the previous paragraph. For the fixed selection, we migrate all 500 VMs with the same algorithms and all live migration algorithms. For the oracle selection, we use actual and not predicted values to select the best technique according to 8.1.

Objective score. We first compare the different selection methods with regards to the given objective. The bars in Figure 7 show how close the model-guided selection matches the oracle with respect to the objective of the different policies A to H. On average, the model-guided selection satisfies the objective function to 97% compared to the oracle selection. The lowest objective scores are obtained for policies that have minimizing downtime as an objective; this reflects the fact that the accuracy of the model is lower than for the other objectives.

SLA Violations. In addition to the objective scores, we measure the number of SLA violations of the model-guided selection with respect to the oracle. We use the *relative error SLA violation error*

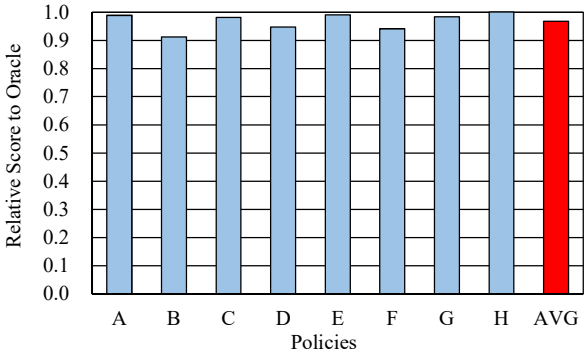


Figure 7: Objective scores

from 8.1 as the score metric. Figure 8 shows that the model-guided selection outperforms each static live migration algorithm by a wide margin and comes very close to the optimal result. Figure 9 displays the number and type of SLA violations for each selection method and the policies with SLA restrictions. Again, we observe that employing a single technique incurs many more SLA violations

	Policy A		Policy B		Policy C		Policy D		Policy E		Policy F		Policy G		Policy H	
	Ours	OPT	Ours	OPT	Ours	OPT	Ours	OPT	Ours	OPT	Ours	OPT	Ours	OPT	Ours	OPT
PRE	8	3	1	0	4	0	61	67	42	45	33	45	109	121	34	35
THR	19	1	0	0	3	0	37	44	44	42	27	24	43	42	24	23
DLTC	21	1	1	0	2	0	294	268	307	294	294	301	190	186	255	250
DTC	5	3	1	0	217	216	60	85	36	26	39	29	51	46	166	141
POST	460	505	510	513	287	297	61	49	84	106	120	114	120	118	34	64
Hit rate	0.89		0.99		0.88		0.58		0.64		0.71		0.59		0.69	

Table 7: Live migration algorithm selection under different policies for the proposed (Ours) and the oracle (OPT) model

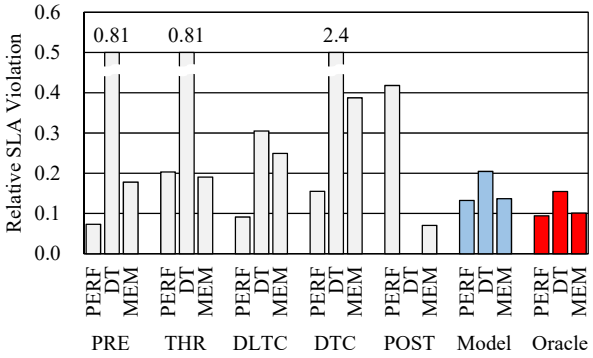


Figure 8: SLA violation scores

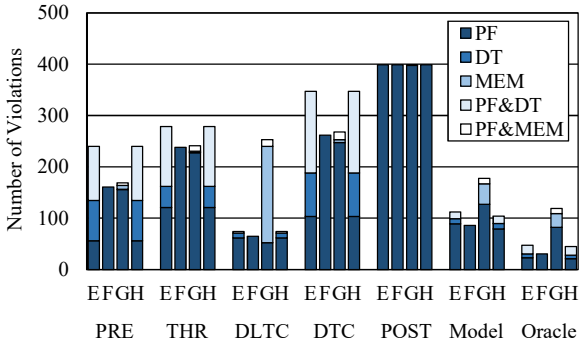


Figure 9: Number and type of violations

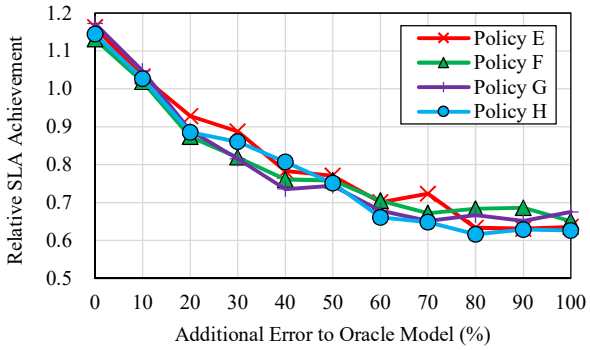


Figure 10: Oracle with errors

than the proposed approach. Note that a violation of an SLA even by the smallest margin counts as a violation; from Figure 8 we know that the relative violation error is close to that of the oracle selection.

Algorithm Selection. Table 7 compares the choices of the migration algorithm of our model-guided versus the oracle selection. We observe a similar distribution of the selected techniques between two for the different migration algorithms. We also count how many times the model-guided selection chose the same technique as the oracle; this is reflected in the *hit rate* defined as the number of correct predictions of our model with respect to the oracle. The hit rate varies for the different policies. We observe lower hit rates for the more complex policies E-H when the benefits of the algorithms are not obviously different (D). However, even a lower hit rate does not mean that the guided selection performed poorly. In many cases, the model-guided selection picked an algorithm performing only marginally worse than the one selected by the oracle. This is evident by looking at Figures 7 and 8 that reveal that the loss of benefit is not that significant.

Overall, we conclude that the model-guided migration selection outperforms the standard practice of selecting a single algorithm for all live migrations and that it performs reasonably well compared to the lower bound set by the oracle selection. In a dynamic scenario, where the VM migration schedule is determined based on hot/cold spot detection, we expect that our model-guided selection will result in fewer migrations and SLA violations.

Effect of modeling accuracy on SLA violations. To measure the importance of the model accuracy with respect to the number of SLA violations, we measure the number of violations of the oracle model with artificially introduced errors from 0 to 100 percent.

Figure 10 shows the score of respected SLAs normalized to our model. We observe that model accuracy has a similar effect on SLA violations independent of the policy. Higher model accuracy clearly leads to less SLA violations. With an error of 40% or more, algorithm selection performs similar to random guessing. We observe that our model performs similar to an oracle with a 10% error. We also notice that higher prediction accuracy has the potential to lead to up to 15% less SLA violations.,

8.3 Generality and Limitations

The results in this paper show that predicting diverse target metrics for a variety of live migrations using machine learning is feasible and leads to good results for our experimental setup. The presented features and the gathered dataset, however, may not be sufficient for more complex data center environments with heterogeneous nodes,

hierarchical networks, or special virtualization technologies such as PCI pass-through. Nevertheless, this work demonstrates that machine learning techniques can be employed to semi-automatically generate accurate prediction models for a wide range of features. The input features fed to the models may have to be extended and new live migration data may have to be collected for specific cases, but our results also show that a few thousand migrations suffice to achieve relatively accurate predictions of the target metrics.

9 RELATED WORK

Machine learning is a powerful tool that uses data to solve complex problems in real systems. Several researchers have applied machine learning to solve critical problems in data center management such as performance modeling of VMs [9, 24] or an interference-aware cluster management framework [11, 12, 39]. In this work, we focus on modeling key metrics of live migration to enable elastic management of data center resources.

Modeling live migration performance accurately has been addressed by a number of researchers in the past. Akoush *et al.* [1] propose a simulation-based live migration modeling approach. The model only predicts the total migration time and the downtime. Important parameters are missing which limits the prediction accuracy to only 90%. Liu *et al.*'s [27] online performance prediction model also requires profiling of all memory accesses of a VM. In addition to total migration time, downtime, and total traffic, the model also predicts the power consumption of a migration. Important input parameters are missing, resulting in reduced prediction accuracy. To the best of our knowledge, the most advanced work on live migration modeling has recently been presented by Nathan *et al.* [34]. The authors reveal serious performance problems of twelve existing models [1, 2, 13, 25–27, 29, 38, 50, 51, 53, 54] and propose an accurate analytic model that outperforms all existing models for Xen and KVM. Predicted metrics are limited to the total migration time, the downtime, and the total traffic of a live migration. In comparison to Nathan *al.*'s work, our machine learning-based approach is not only much more versatile in supporting all commonly available live migration algorithms and six target metrics but also significantly more accurate.

Although there have been many works on live migration modeling, we argue that all current approaches are far from ideal. First, there is no general model that can predict performance degradation under live migration. According to [18, 49, 52], the degree of performance degradation of a migrated workload is highly dependent on the workload itself. Ye *et al.* [52] predict the performance degradation of a VM based on a history of recorded migrations, rendering this method unsuitable for unknown workloads. Second, existing models do not properly capture the effect of limited resources on the performance of live migration. One attempt [36] only considers the reserved network bandwidth as an input parameter; other important parameters such as the utilization of CPU and memory are ignored. Third, existing models fail to predict the expected additional resource requirements of the live migration process. This can be important information, especially when there is a lack of resources on the servers involved in the migration. Last but not least, there is no model that supports all common live migration algorithms. Since pre-copy and post-copy live migration have been introduced, many optimization techniques such as data compression, delta compression, and CPU throttling have been proposed. There is some work on selecting the proper optimization technique and important parameters to the performance of optimization techniques [35, 45], but the current models only consider vanilla Xen and KVM migration techniques.

In this work, we address all of the aforementioned problems. We exploit the power of a diverse migration dataset and state-of-the-art machine learning techniques to model various live migration algorithms and predict important target metrics including the performance loss of the VM and the expected additional resource consumption during migration. Our model supports all common live migration algorithms and predicts six performance metrics with high accuracy which is a great step forward from the current state. Compared to existing work, the proposed model is more versatile, supports more migration algorithms, is able to predict more metrics, and demonstrates a higher accuracy for the predicted metrics.

10 CONCLUSION AND FUTURE WORK

This work, a machine learning-based technique to automatically construct accurate models that can predict key metrics of VM live migration under varying resource constraints and workloads for all commonly available migration algorithms has been proposed. We discuss the relevant parameters necessary to model the performance metrics and outline the key ideas to lightweight profiling of these input parameters. The proposed technique achieves a very high prediction accuracy for all target metrics and live migration algorithms for heterogeneous set of physical and virtual machines. Compared to the state-of-the-art, the presented model achieves an improvement of prediction accuracy of factor 2 to 5. The results show that accurate analytical models are difficult to build and require a big engineering effort. We strongly believe that the presented machine learning-based approach is the proper way to go forward since it allows for easy and automatic adaptation to new target metrics, live migration algorithms, and hardware environments. Employed in a live migration framework, the proposed model is shown to reduce the number of SLA and operator-specified constraint violations significantly.

In future work, we plan to adapt and test the model on more heterogeneous platforms and in more complex migration scenarios. The dataset and the algorithms used to build and evaluate the models in this work are available online to interested researchers at http://csap.snu.ac.kr/software/lmdataset.

ACKNOWLEDGEMENTS

We thank our shepherd Asim Kadav and the anonymous reviewers for their detailed feedback and guidance. Also, we thank Jeongseok Son for his contribution to the synthetic workload generator used to generate the dataset in the paper. This work has been supported in part by grants NRF-2015K1A3A1A14021288 and 2016R1A2B4009193 of the National Research Foundation of Korea and by the Promising-Pioneering Researcher Program of Seoul National University in 2015. ICT at Seoul National University provided research facilities for this study.

REFERENCES

[1] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. 2010. Predicting the Performance of Virtual Machine Migration. In *Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '10)*. IEEE Computer Society, Washington, DC, USA, 37–46. https://doi.org/10.1109/MASCOTS.2010.13

[2] Arwa Aldhalaan and Daniel A. Menascé. 2013. Analytic Performance Modeling and Optimization of Live VM Migration. In *Proceedings of 10th European Workshop (EPEW '13)*. 28–42. https://doi.org/10.1007/978-3-642-40725-3_4

[3] Amazon EC2 - Virtual Server Hosting 2017. https://aws.amazon.com/ec2/. (2017). Online; accessed August 2017.

[4] Fabrice Bellard. 2005. QEMU, a Fast and Portable Dynamic Translator. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC '05)*. USENIX Association, Berkeley, CA, USA, 41–41. http://dl.acm.org/citation.cfm?id=1247360.1247401

[5] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT '08)*. ACM, New York, NY, USA, 72–81. https://doi.org/10.1145/1454115.1454128

[6] Stephen M. Blackburn, Robin Garner, Chris Hoffmann, Asjad M. Khang, Kathryn S. McKinley, Rotem Bentzur, Amer Diwan, Daniel Feinberg, Daniel Frampton, Samuel Z. Guyer, Martin Hirzel, Antony Hosking, Maria Jump, Han Lee, J. Eliot B. Moss, Aashish Phansalkar, Darko Stefanović, Thomas VanDrunen, Daniel von Dincklage, and Ben Wiedermann. 2006. The DaCapo Benchmarks: Java Benchmarking Development and Analysis. In *Proceedings of the 21st Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications (OOPSLA '06)*. ACM, New York, NY, USA, 169–190. https://doi.org/10.1145/1167473.1167488

[7] Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24, 2 (1996), 123–140. https://doi.org/10.1007/BF00058655

[8] A. Colin Cameron and Frank A.G. Windmeijer. 1997. An R-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics* 77, 2 (1997), 329 – 342. https://doi.org/10.1016/S0304-4076(96)01818-0

[9] Ron C. Chiang, Jinho Hwang, H. Howie Huang, and Timothy Wood. 2014. Matrix: Achieving Predictable Virtual Machine Performance in the Clouds. In *11th International Conference on Autonomic Computing (ICAC 14)*. USENIX Association, Philadelphia, PA, 45–56. https://www.usenix.org/conference/icac14/technical-sessions/presentation/chiang

[10] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. 2005. Live Migration of Virtual Machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2 (NSDI'05)*. USENIX Association, Berkeley, CA, USA, 273–286. http://dl.acm.org/citation.cfm?id=1251203.1251223

[11] Christina Delimitrou and Christos Kozyrakis. 2013. QoS-Aware Scheduling in Heterogeneous Datacenters with Paragon. *ACM Trans. Comput. Syst.* 31, 4, Article 12 (Dec. 2013), 34 pages. https://doi.org/10.1145/2556583

[12] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-efficient and QoS-aware Cluster Management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 127–144. https://doi.org/10.1145/2541940.2541941

[13] Li Deng, Hai Jin, Huacai Chen, and Song Wu. 2013. Migration Cost Aware Mitigating Hot Nodes in the Cloud. In *Proceedings of the 2013 International Conference on Cloud Computing and Big Data (CLOUDCOM-ASIA '13)*. IEEE Computer Society, Washington, DC, USA, 197–204. https://doi.org/10.1109/CLOUDCOM-ASIA.2013.72

[14] Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, and Philippe Cudre-Mauroux. 2013. OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. *Proc. VLDB Endow.* 7, 4 (Dec. 2013), 277–288. https://doi.org/10.14778/2732240.2732246

[15] Jim Gao and Ratnesh Jamidar. 2014. Machine learning applications for data center optimization. *Google White Paper* (2014).

[16] Google Compute Engine 2017. https://cloud.google.com/compute. (2017). Online; accessed August 2017.

[17] Google Compute Engine uses Live Migration technology to service infrastructure without application downtime 2017. https://goo.gl/Ui3HFd. (2017). Online; accessed August 2017.

[18] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. 2009. Entropy: A Consolidation Manager for Clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*. ACM, New York, NY, USA, 41–50. https://doi.org/10.1145/1508293.1508300

[19] Michael R. Hines and Kartik Gopalan. 2009. 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 (VEE '09)*. ACM, New York, NY, USA, 51–60. https://doi.org/10.1145/1508293.1508301

[20] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, and Xiaodong Pan. 2009. Live virtual machine migration with adaptive, memory compression. In *2009 IEEE International Conference on Cluster Computing and Workshops*. 1–10. https://doi.org/10.1109/CLUSTR.2009.5289170

[21] Changyeon Jo and Bernhard Egger. 2013. Optimizing Live Migration for Virtual Desktop Clouds. In *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom '13)*, Vol. 1. 104–111. https://doi.org/10.1109/CloudCom.2013.21

[22] Changyeon Jo, Erik Gustafsson, Jeongseok Son, and Bernhard Egger. 2013. Efficient Live Migration of Virtual Machines Using Shared Storage. In *Proceedings of the 9th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '13)*. ACM, New York, NY, USA, 41–50. https://doi.org/10.1145/2451512.2451524

[23] Jonathan Koomey. 2011. Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times* 9 (2011).

[24] Sajib Kundu, Raju Rangaswami, Ajay Gulati, Ming Zhao, and Kaushik Dutta. 2012. Modeling Virtualized Applications Using Machine Learning Techniques. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (VEE '12)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/2151024.2151028

[25] Jianxin Li, Jieyu Zhao, Yi Li, Lei Cui, Bo Li, Lu Liu, and John Panneerselvam. 2014. iMIG: Toward an Adaptive Live Migration Method for KVM Virtual Machines. *Comput. J.* 58, 6 (2014), 1227. https://doi.org/10.1093/comjnl/bxu065

[26] Haikun Liu and Bingsheng He. 2015. VMbuddies: Coordinating Live Migration of Multi-Tier Applications in Cloud Environments. *IEEE Transactions on Parallel and Distributed Systems* 26, 4 (April 2015), 1192–1205. https://doi.org/10.1109/TPDS.2014.2316152

[27] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. 2013. Performance and energy modeling for live migration of virtual machines. *Cluster Computing* 16, 2 (2013), 249–264. https://doi.org/10.1007/s10586-011-0194-3

[28] Zhaobin Liu, Wenyu Qu, Weijiang Liu, and Keqiu Li. 2010. Xen Live Migration with Slowdown Scheduling Algorithm. In *Proceedings of the 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '10)*. IEEE Computer Society, Washington, DC, USA, 215–221. https://doi.org/10.1109/PDCAT.2010.88

[29] Vijay Mann, Akanksha Gupta, Partha Dutta, Anilkumar Vishnoi, Parantapa Bhattacharya, Rishabh Poddar, and Aakash Iyer. 2012. *Remedy: Network-Aware Steady State VM Management for Data Centers*. Springer Berlin Heidelberg, Berlin, Heidelberg, 190–204. https://doi.org/10.1007/978-3-642-30045-5_15

[30] Memcached - a distributed memory object caching system 2017. https://memcached.org/. (2017). Online; accessed August 2017.

[31] Microsoft Azure - Virtual Machines 2017. https://azure.microsoft.com/en-us/services/virtual-machines/. (2017). Online; accessed August 2017.

[32] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo. 2012. Dynamic resource management using virtual machine migrations. *IEEE Communications Magazine* 50, 9 (September 2012), 34–40. https://doi.org/10.1109/MCOM.2012.6295709

[33] MPlayer - The Movie Player 2017. http://www.mplayerhq.hu/design7/news.html. (2017). Online; accessed August 2017.

[34] Senthil Nathan, Umesh Bellur, and Purushottam Kulkarni. 2015. Towards a Comprehensive Performance Model of Virtual Machine Live Migration. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15)*. ACM, New York, NY, USA, 288–301. https://doi.org/10.1145/2806777.2806838

[35] Senthil Nathan, Umesh Bellur, and Purushottam Kulkarni. 2016. On Selecting the Right Optimizations for Virtual Machine Migration. In *Proceedings of the12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '16)*. ACM, New York, NY, USA, 37–49. https://doi.org/10.1145/2892242.2892247

[36] Senthil Nathan, Purushottam Kulkarni, and Umesh Bellur. 2013. Resource Availability Based Performance Benchmarking of Virtual Machine Migrations. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE '13)*. ACM, New York, NY, USA, 387–398. https://doi.org/10.1145/2479871.2479932

[37] Michael Nelson, Beng-Hong Lim, and Greg Hutchins. 2005. Fast Transparent Migration for Virtual Machines. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC '05)*. USENIX Association, Berkeley, CA, USA, 25–25. http://dl.acm.org/citation.cfm?id=1247360.1247385

[38] Hiep Nguyen, Zhiming Shen, Xiaohui Gu, Sethuraman Subbiah, and John Wilkes. 2013. AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*. USENIX, San Jose, CA, 69–82. https://www.usenix.org/conference/icac13/technical-sessions/presentation/nguyen

[39] Dejan Novaković, Nedeljko Vasić, Stanko Novaković, Dejan Kostić, and Ricardo Bianchini. 2013. DeepDive: Transparently Identifying and Managing Performance Interference in Virtualized Environments. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference (USENIX ATC'13)*. USENIX Association, Berkeley, CA, USA, 219–230. http://dl.acm.org/citation.cfm?id=2535461.2535489

[40] Greg Schulz. 2009. *The green and virtual data center*. Auerbach Publications Boston.

[41] scikit-learn - Machine Learning in Python 2017. http://scikit-learn.org/stable/. (2017). Online; accessed August 2017.

[42] C. E. Shannon. 2001. A Mathematical Theory of Communication. *SIGMOBILE Mob. Comput. Commun. Rev.* 5, 1 (Jan. 2001), 3–55. https://doi.org/10.1145/584091.584093

[43] SPECweb2009 2017. https://www.spec.org/web2009/. (2017). Online; accessed August 2017.

[44] Petter Svärd, Benoit Hudzia, Johan Tordsson, and Erik Elmroth. 2011. Evaluation of Delta Compression Techniques for Efficient Live Migration of Large Virtual Machines. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '11)*. ACM, New York, NY, USA, 111–120. https://doi.org/10.1145/1952682.1952698

[45] Petter Svärd, Benoit Hudzia, Steve Walsh, Johan Tordsson, and Erik Elmroth. 2015. Principles and Performance Characteristics of Algorithms for Live VM Migration. *SIGOPS Operating Systems Review - Special Issue on Repeatability and Sharing of Experimental Artifacts* 49, 1 (Jan. 2015), 142–155. https://doi.org/10.1145/2723872.2723894

[46] The Xen Project 2017. https://www.xenproject.org/. (2017). Online; accessed August 2017.

[47] Marvin M. Theimer, Keith A. Lantz, and David R. Cheriton. 1985. Preemptable Remote Execution Facilities for the V-system. In *Proceedings of the Tenth ACM Symposium on Operating Systems Principles (SOSP '85)*. ACM, New York, NY, USA, 2–12. https://doi.org/10.1145/323647.323629

[48] VMware Virtualization Solutions 2017. http://www.vmware.com/. (2017). Online; accessed August 2017.

[49] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. 2009. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation. In *Proceedings of the 1st International Conference on Cloud Computing (CloudCom '09)*. Springer-Verlag, Berlin, Heidelberg, 254–265. https://doi.org/10.1007/978-3-642-10665-1_23

[50] Yangyang Wu and Ming Zhao. 2011. Performance Modeling of Virtual Machine Live Migration. In *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD '11)*. IEEE Computer Society, Washington, DC, USA, 492–499. https://doi.org/10.1109/CLOUD.2011.109

[51] Fei Xu, Fangming Liu, Linghui Liu, Hai Jin, Bo Li, and Baochun Li. 2014. iAware: Making Live Migration of Virtual Machines Interference-Aware in the Cloud. *IEEE Trans. Comput.* 63, 12 (Dec 2014), 3012–3025. https://doi.org/10.1109/TC.2013.185

[52] Kejiang Ye, Zhaohui Wu, Chen Wang, Bing Bing Zhou, Weisheng Si, Xiaohong Jiang, and Albert Y Zomaya. 2015. Profiling-Based Workload Consolidation and Migration in Virtualized Data Centers. *IEEE Transactions on Parallel and Distributed Systems* 26, 3 (March 2015), 878–890. https://doi.org/10.1109/TPDS.2014.2313335

[53] Jiao Zhang, Fengyuan Ren, and Chuang Lin. 2014. Delay guaranteed live migration of Virtual Machines. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 574–582. https://doi.org/10.1109/INFOCOM.2014.6847982

[54] Jie Zheng, TS Ng, Kunwadee Sripanidkulchai, and Zhaolei Liu. 2013. Pacer: A Progress Management System for Live Virtual Machine Migration in Cloud Computing. *IEEE Transactions on Network and Service Management* 10, 4 (December 2013), 369–382. https://doi.org/10.1109/TNSM.2013.111013.130522