Modelling the Live Migration Time of Virtual Machines

Kateryna Rybina(✉), Waltenegus Dargie, Subramanya Umashankar, and Alexander Schill

Computer Networks Group, Technical University of Dresden, Nöthnitzer Str. 46, 01187 Dresden, Germany
{kateryna.rybina,waltenegus.dargie,alexander.schill}@tu-dresden.de

Abstract. Dynamic server consolidation in data centres enables the efficient usage of resources, because it aims to minimise the underutilisation or overloading of physical servers, both of which produce a disproportional amount of energy consumption. Server consolidation takes place by migrating virtual machines from one server to another while the virtual machines are still executing. However, live migration comes with corresponding costs in terms of execution latency and additional resource and power consumption. Whether or not these costs are significant depends on how long a migration lasts. In this paper we propose models to estimate the time it takes to live migrate virtual machines at runtime. Our models are built using simple and multiple linear regressions. The paper reveals useful insights into the most important parameters which are strongly correlated with the migration time. These are: Instructions retired, last level cache line misses, and dirtying memory pages.

Keywords: Virtual machines · Live migration · Service consolidation · Migration time · Linear regression model

1 Introduction

The advent of server virtualisation and cloud computing has enabled great flexibility in managing computing resources. It is now possible to create an abstract partitioning of a single physical server into multiple, non-overlapping, and non-interfering computing environments (virtual machines), so that they can be used by multiple independent users. The portion of these partitions can be dynamically adapted (or resized) to the need of the individual users. The physical servers themselves can also be managed by dynamically (live) migrating virtual machines from one server to another without actually stopping or suspending the virtual machines.

One of the advantages of virtual machine migration is dynamic consolidation of servers in a cloud infrastructure or data centre. Due to the fluctuation of incoming workloads, resources may not be utilised uniformly across all servers.

C. Debruyne et al. (Eds.): OTM 2015 Conferences, LNCS 9415, pp. 575–593, 2015.
DOI: 10.1007/978-3-319-26148-5_39

Some of them may be overloaded while others are underutilised or even idle. This imbalance not only creates dissimilar quality of service for different users but it is also inefficient because the power consumption of idle and underutilised servers exceeds 50% of their peak power consumption [1], [2]. By aggregating the virtual machines of data centres on a few number of machines, the rest can be switched off. Similarly, when servers are overloaded, additional servers can be switched on and virtual machines from overloaded servers can be migrated to them, thus seamlessly balancing the load of the data centres.

Aggregating virtual machines, however, introduces some costs. To begin with, a background process should continuously or at a regular interval estimate the size of the incoming workloads and the amount of resources required to handle them. Secondly, the live migration of virtual machines requires additional resources to iteratively transfer the content of the virtual machines and to coordinate the migration. Thirdly, the quality of service execution within the migrated virtual machines may degrade, since the virtual machines should now share resources (such as CPU and network bandwidth) with the migration process. Of all these costs, the third is the most significant one because it directly affects the service level agreement between the computing platform provider and the platform users. The cost is more pronounced if the migration takes a long time and the deterioration of service quality is perceived by the platform users (for example, in terms of increased response time and jitter).

Migration time depends on many factors including the activity and RAM utilisation of the virtual machines, the available CPU cycles and network bandwidth during migration, and the activity of co-located virtual machines. Several studies have been made in the past to estimate migration time and to determine the conditions that initiate VM migration. The model of Strunk [7] estimates the migration time of an idle virtual machine using a simple linear regression with an independent variable expressing the ratio of the active memory occupied by the VM to the available network bandwidth during migration. Clark et al. [6] and Liu et al. [4] investigate the impact of VM memory size, memory page dirtying rate, and network bandwidth on migration time. Akoush et al. [3] investigate the upper and lower bounds of migration time and the runtime parameters that influence migration time. Similar to Clark et al. they too investigate the impact of network bandwidth, memory page dirtying rate, VM memory size, and pre- and post-migration overheads on migration time. Moreover, they experimentally show that (1) network bandwidth is inversely proportional to migration time; (2) a non-linear dependency exists between memory page dirtying rate and migration time due to a stop conditions defined by pre-copy migration strategies; and (3) migration time linearly increases with the VM RAM size. Wu et al. [5] investigate the relationship between migration time and the amount of CPU resources available for migration. The authors propose separate models for different types of workloads (CPU intensive, memory read and write intensive, disk I/O intensive, and network I/O (send-receive) intensive workloads). Likewise, Verma et al. [8] propose an application-aware model to estimate migration time. The model accounts for CPU resource contention on the source server by co-located virtual machines and by the migration process itself.

In this paper we experimentally investigate the scope and usefulness of several resource utilisation metrics to estimate migration time. Unlike previous approaches, (1) we provide adequate and quantitative justification for the selection of the relevant metrics; (2) the metrics we identify estimate the migration time of different virtual machines with comparable accuracy regardless of the workload they process, and (3) the models we propose are all lightweight, linear models that are easy to comprehend.

The remaining part of this paper is organised as follows: In Section 2 we provide a brief background regarding virtual machine migration and linear regression. In Section 3 we introduce our experiment setting, the selection strategy of resource utilisation metrics and benchmarks, and the training and testing datasets. In Section 4 we introduce our approach and provide quantitative justification to the models we propose. We also provide experiment results and discus the results. Finally, in Section 5, we point out concluding remarks and outline future work.

2 Background

2.1 Virtual Machine Migration

In order to estimate the migration time of a virtual machine, it is essential to understand how migration takes place. During the live migration of a virtual machine, its RAM content is copied from the source to the destination server without stopping the execution of the virtual machine. Since the virtual machine is active, its memory content on the source server can change any time (i.e., the memory pages can be dirtied) and this change has to be synchronised with the content of the destination server. This is done by iteratively copying the dirty pages to the destination server. The iteration, however, does not go on indefinitely. Upon reaching a pre-defined threshold (stop-condition) by the migration algorithm, the VM is briefly stopped, all the updated pages are copied to the destination server for one final time; and the VM is started on the destination machine. The total VM migration time referred to as migration time is the time interval between the initialisation of the VM migration at the source server and the starting of the VM at the destination server. Obviously, this time is a function of the memory size of the virtual machine, the available network bandwidth, the memory update rate of the applications or services the virtual machine hosts, the CPU load, and the additional resources the virtual machine monitors on the two servers require to coordinate migration. Consequently, a model that estimates the migration time of a virtual machine should take these parameters into consideration.

2.2 Estimation Error

We begin our investigation on migration time by assuming that a linear dependency can be established between migration time and resource utilisation during migration. In other words, migration time can be expressed as a linear combination of independent parameters that describe resource utilisation. The strength of this expression can be tested by examining such useful metrics as residual standard error, prediction error, mean absolute percentage error, and coefficient of determination (R^2).

Given two random variables X and Y, we wish to express one of them (Y) in terms of the other (X). A linear regression assumes that a conditional expectation $\mathbf{E}\{Y \mid X = x\}$ is a linear function of x [14]:

$$Y = G(x) = \mathbf{E}\{Y \mid X = x\} = \beta_0 + \beta_1 x \ . \tag{1}$$

where β_0 and β_1 are intercept and slope, respectively. Because Y is related to a single independent variable (predictor), the relation is said to be Simple Linear Regression (SLR). A Multiple Linear Regression (MLR) relates the dependent variable with more than one independent variables and assumes that the conditional expectation of the dependent variable is a linear function of the independent variables (predictors) $x_{1i}, ..., x_{ki}$ [13], [14]:

$$Y_i = G(x) = \beta_0 + \beta_1 x_{1i} + ... + \beta_k x_{ki} \quad i = 1...n \ . \tag{2}$$

where n is the number of observations (samples), k is the number of independent variables, β_0 is an intercept, β_j is the regression coefficient for the j-th independent variable, showing the expected change of the dependent (response) variable when the corresponding predictor changes by a unite value while all the other predictors remain constant; and x_{ji} is the j-th independent variable's value for the i-th observation. The linear regression model minimises the sum of squared residuals. In other words, the model parameters are so selected that the sum of squared differences between the actual values of the dependent variable and the fitted values by the model (which lie on the fitted regression line or plane) are minimised.

The model's error is a measure of how well the model fits the measured data. The residual standard error of the linear regression model (as calculated in R statistical tool [15]) is defined as:

$$Res_{st.err} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-k-1}} = \sqrt{\frac{\sum_{i=1}^{n} r_i^2}{n-k-1}} \ . \tag{3}$$

where y_i is the actual (measured) value of the dependent variable and $\hat{y}_i$ is the fitted by the model value; n is the sample size, k is the number of independent variables, minus 1 accounts for the estimated intercept, and r_i^2 denotes the squared residual for the i-th observation. The residual standard error of the model is calculated on the training data. The model's standard error of estimate from a sample (prediction error) is calculated on the testing data. It quantifies the departure of the predicted (estimated) by the trained model value $\ddot{y}_i$ of the dependent variable from the actual (or measured) value:[1]

[1] Available at http://onlinestatbook.com/2/regression/accuracy.html

$$Pred_{st.err} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \ddot{y}_i)^2}{n-k-1}} \ . \tag{4}$$

The percentage error[2] is another metric that analyses how close the predicted value $\ddot{y}_i$ is to the true (measured) value y_i and it gives the difference between the predicted and the true value as a percentage of the true value. It is calculated as follows:

$$Perc = \frac{\ddot{y}_i - y_i}{y_i} * 100\% \ . \tag{5}$$

In case of n observations the mean absolute percentage error of the model (its prediction accuracy) will be calculated as a mean of the absolute values of the percentage errors of these n observations as shown in equation below:

$$Perc_m = \frac{\sum_{i=1}^{n}\left|\frac{\ddot{y}_i - y_i}{y_i}\right|}{n} * 100\% \ . \tag{6}$$

3 Experiment

3.1 Experiment Settings

The hardware setup of our experiments consists of two (source and destination) homogeneous servers which are interconnected via a 1 Gbps Ethernet switch, a client server used to trigger the experiments, a network attached storage (NAS), and two power analysers. Both servers consist of two Intel 15-680 Dual Core 3.6 GHz processors, 4 GB DDR3-1333 SDRAM and a 1 Gbit/s Ethernet NIC. The NAS employs an Intel Xeon E5620 Quad-Core 2.4 GHz processor, 10 GB DDR3-1333 SDRAM memory, and 1 Gbps Ethernet NIC. Fedora (Linux kernel v. 2.6.38, x86 64) was installed as the host operating system on both physical servers. KVM[3] was used as a hypervisor and libvirt[4], as a toolkit to manage virtual machines. As a NAS we employed FreeNAS[5], which is a FreeBSD-based operating system (v. 8.0.1, AMD 64). In our experiments each time one VM was running in isolation on the source server and migrated between the source and the destination physical servers. For different experiments we varied the network bandwidth from 70 MBps to 100 MBps in steps of 10 MBps. The VM was allocated 4 GB RAM, 4 virtual CPUs and 20 GB disc space on the NAS. Ubuntu 14.04.2 LTS (Linux kernel 3.16.0-30-generic) was the operating system installed on the VM.

Inside the migrated virtual machine we executed benchmarks from the SPEC CPU2006 benchmark suite [9] (more information is given in Subsection 3.4). While a benchmark is still executed we migrated the virtual machine back

[2] Available at http://mathworld.wolfram.com/PercentageError.html
[3] http://www.linux-kvm.org/page/Main_Page
[4] Libvirt: The virtualization API. http://libvirt.org/
[5] FreeNAS: FreeBSD-based operating system. http://www.freenas.org/

and forth 20 times each time with the same network bandwidth. We carefully recorded the beginning and end of a migration time and the values of the resource utilisation parameters of the servers as well as the VM. The data analysis were realised with R statistical tool [15].

3.2 Selection of Parameters

The complete list of parameters (independent variables) we examined to model and estimate the migration time (the dependent variable) is shown in Table 1. We employed *dstat* to record the CPU and RAM utilisation of the servers as well as the VM (CPU_{util_server}, CPU_{util_vm}, MEM_{server}, and MEM_{vm}). Likewise, we employed the *Intel PCM* tool (Intel Performance Monitoring Counters) [10] to monitor last level cache line misses ($L3_{miss}$) and the total number of instructions retired $INST$ [12]. We also recorded the total amount of "dirty memory" (in kilobytes) waiting to be written back to the disk [11]. From these statistics we derived two additional parameters, namely, the total number of "dirty" pages of the source server during migration ($DirtyPages_{server}$) and the number of "dirty" pages in the source server per second per migration ($DirtyPages_{server\_per\_sec}$). The former was derived as follows: Using timestamps we extracted the *Dirty* statistics of the source server in KB which corresponded to the migration duration; then we calculated the positive increase in the number of *Dirty* memory in KB, summed it and divided the sum by the page-size. The page-size in our system was 4 KB. The latter was derived by dividing the number of "dirty" pages during migration by the migration duration (in seconds). Finally, we adopted an additional parameter from Poellabauer et al. [16] (memory access rate (MAR)), which is derived as the ratio of data cache misses to the instructions executed:

$$MAR = \frac{L3_{miss}}{INST} \ . \tag{7}$$

where $L3_{miss}$ refers to the total number of last level cache line misses during migration and $INST$ refers to the total number of instructions retired during migration. Memory access rate is proportional to the last level cache line misses. Consequently, if a benchmark modifies a memory page during migration, this page will have to be resent, resulting in an increase in the migration time. Thus, it is of interest to examine the strength of correlation between the migration time, on the one hand, and the $L3_{miss}$ and MAR parameters, on the other. The total number of instructions retired $INST$ is another parameter which potentially correlates well with the migration time.

3.3 Dataset

Our complete dataset consists of 880 observations containing 13 variables (12 of which are the independent variables and one, t_{mig}, the dependent variable). These correspond to 11 benchmarks $\times$ 4 different network bandwidths $\times$ 20 migrations per a configuration. One of our tasks was selecting from the long list of independent variables a handful of those which are strongly correlated

Table 1. Resource utilisation parameters (independent variables) used for modelling the VM migration time (t_{mig}).

Name of variable	Description
t_{mig}	Total VM migration time in seconds
BW	Network bandwidth available for migration in MBps
$L3_{miss}$[12]	Total number of L3 cache line misses during VM migration
$INST$ [12]	Total number of instructions retired during migration
MAR [16]	Ratio of total number of L3 cache line misses to the total number of instructions retired during migration
CPU_{util_server}	Mean total CPU utilisation of the source server during the VM migration
CPU_{util_vm}	Mean total CPU utilisation of the VM during the migration process
MEM_{server}	Mean active memory utilisation of the source server during the VM migration in MB
MEM_{vm}	Mean active memory utilisation of the VM during the migration process in MB
$MEMtoBW_{server}$	Ratio of active memory utilised by the source server to the network bandwidth available for migration
$MEMtoBW_{vm}$	Ratio of active memory utilised by the VM to the network bandwidth available for migration
$DirtyPages_{server}$	Number of “dirty” pages observed in the source server during the migration process
$DirtyPages_{server\_per\_sec}$	Number of “dirty” pages observed in the source server per second during the migration process

with the migration time. We divided the dataset into **training dataset** and **testing dataset**. The training data are used to build relationship between the dependent and independent variables whereas the test data are used for testing the estimation accuracy of our models. As a rule three fourth of the dataset is used for setting up (training) the model and one fourth is used for testing. We randomly divided the dataset thus: The measurements pertaining to the network bandwidth of 70 MBps, 80 MBps, and 100 MBps belong to the **training data** and the measurements pertaining to the network bandwidth of 90 MBps belong to the **testing data**.

3.4 Benchmarks

Jaleel et al. [17] made an extensive analysis of the benchmarks from the SPEC CPU2006 benchmark suite with regard to their resource utilisation characteristics. Based on this study we selected eleven benchmarks, six of which are predominantly CPU intensive (even though they also utilise a sizeable memory); and the other five are memory intensive. The CPU intensive benchmarks used the maximum CPU time, keeping the CPU busy most of the time during the benchmarks execution. A benchmark is considered to be memory intensive if it has a large number of reads/writes operations from/to the memory subsystem. Hence: **libquantum**, **gromacs**, **h264ref**, **namd**, **sphinx3**, and **soplex** belong to the *CPU intensive* benchmarks; and **bzip2**, **astar**, **mcf**, **gcc**, and **perlbench** belong to the *memory intensive* benchmarks.

4 Modelling Migration Time

Fig. 1 displays the VM migration time for different *CPU* intensive benchmarks for 20 migrations realised at a fixed network bandwidth of 70 MBps. As can be seen, the migration time for different benchmarks and migrations was different. The shortest migration time was 26.25 seconds (for **sphinx3**) and the longest was 69.4 seconds (for **libquantum**). The migration time of **sphinx3** exhibited low variation (variance = 0.14) while the migration time of **libquantum** exhibited the largest variation (variance = 36.26). Similarly, Fig. 2 shows the VM migration time for the *memory* intensive benchmarks, which were migrated with a network bandwidth of 80 MBps. Unlike the previous case, the migration times were significantly longer and the variances between the different migrations for some of the benchmarks were considerably larger than the variances we observed in the *CPU* intensive benchmarks. For example, the variances of **astar**, **mcf**, and **perlbench** were 8400.3, 2799.7, and 100.8, respectively. From this it can be concluded that the migration time is strongly influenced by the operating point at which the migration starts and the specific operations the benchmarks execute during migration.

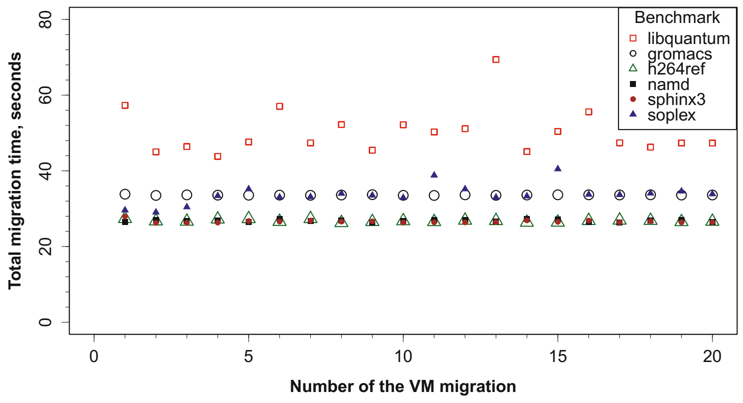

Fig. 1. The migration time of *CPU* intensive benchmarks from the SPEC CPU2006 benchmark suite. The migration bandwidth was 70 MBps.

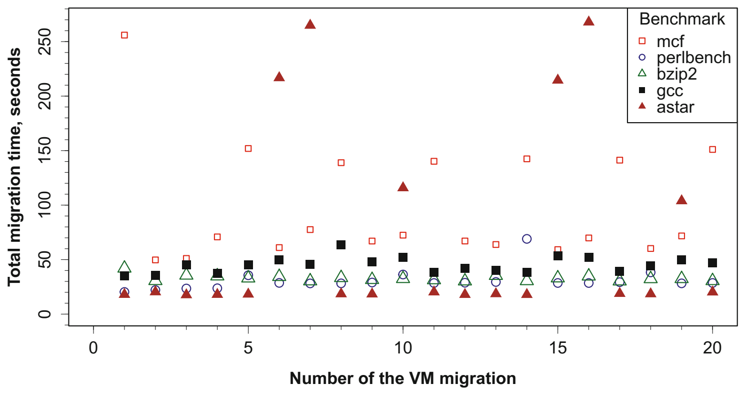

Fig. 2. The migration time of *memory* intensive benchmarks from SPEC CPU2006 benchmark suite. The migration bandwidth was 80 MBps.

4.1 Linear Dependency

To determine the existence of linear dependencies between the independent variables and the migration time, it suffices to calculate the Pearson's correlation coefficient [14] for each independent variable. Table 2 displays the results of our calculation using the training dataset which includes all the measurements we took for different network bandwidths: 70 MBps, 80 MBps and 100 MBps.

Table 2. Pearson correlation coefficients between the migration time and the resource utilisation parameters described in Table 1.

Pearson's correlation, ρ	Value
$cor(t_{mig}, INST)$	0.8604983
$cor(t_{mig}, L3_{miss})$	0.8106588
$cor(t_{mig}, DirtyPages_{server})$	0.8442557
$cor(t_{mig}, MAR)$	0.3986198
$cor(t_{mig}, MEM_{server})$	0.1925516
$cor(t_{mig}, MEMtoBW_{server})$	0.2797072
$cor(t_{mig}, DirtyPages_{server\_per\_sec})$	-0.2659009
$cor(t_{mig}, CPU_{util\_server})$	-0.08078284
$cor(t_{mig}, BW)$	-0.1088528
$cor(t_{mig}, CPU_{util\_vm})$	0.09879503
$cor(t_{mig}, MEM_{vm})$	0.4579728
$cor(t_{mig}, MEMtoBW_{vm})$	0.4950578

As can be seen from Table 2, some of the independent variables, namely, $INST$, $L3_{miss}$, and $DirtyPages_{server}$, display strong linear dependencies while some of the remaining parameters such as $MEMtoBW_{server}$, $MEMtoBW_{vm}$, and MAR are moderately correlated with the migration time. The strong correlations can be logically explained. The number of retired CPU instructions corresponds with the CPU activity level during migration. A large number of retired instructions signifies a high level of CPU activity, which in turn implies a longer migration time. If the data required by a benchmark execution are not available in the cache during VM migration, they have to be fetched from, in case of write access modified, and rewritten back to the main memory. These data have to be resent to the destination server, as they are no longer in sync with the memory content of the destination server. This process prolongs the migration time. Similarly, as the number of dirty pages $DirtyPages_{server}$ increases during migration, the size of data that have to be updated at the destination server increases, which in turn increases the migration time. From our dependency analysis, we concluded that most of the resource utilisation parameters are linearly correlated with the migration time and indeed a few of them show strong linear dependencies. Hence, it is possible to employ linear regression to express migration time in terms of these variables. The question: "How many of these independent variables are sufficient to estimate migration time?" can be answered by considering different combinations of the independent variables and by analysing:

1. the R^2 values and the residual standard error $Res_{st.err}$ (Equation 3) to test how well the models fit the training data; and,
2. the prediction error (Equation 4) and the mean absolute percentage error (Equation 6) of the models using the testing dataset. The latter is the prediction accuracy of the models, as it expresses their accuracy as a percentage.

4.2 Simple Linear Regression Models

The simplest approach is to setup a linear regression model consisting of a single independent variable. The strength of the model and the appropriateness of the independent variable can be judged by analysing R^2 which, for a single independent variable, is simply the square of the sample correlation coefficient (ρ^2) given in Table 2 [14]. R^2 expresses the portion of the total variance of the dependent variable (migration time) that can be captured and explained by the independent variable. A value of 1 implies the regression line perfectly fits the measured data. The adjusted *R-square* R^2_{Adj} is a more useful measure of goodness-of-fit when a model consists of more than one independent variable, as it includes the notion of number of degrees of freedom and penalises when irrelevant or insignificant independent variables are added into the model. Table 3 summarises the simple linear regression models (SLR) we constructed and tested using our independent variables. The independent variables BW, CPU_{util_server}, and CPU_{util_vm} have p-values equal to 0.005, 0.038 and 0.011, respectively, which are lower than the significance level of 0.05. All the other parameters are significant with the p-value lower than the smallest significance level (0.001) which indicates that these independent variables are appropriate for estimating the migration time.

The model that relates migration time with the total number of instructions retired ($INST$) during migration resulted in the highest R^2 value (0.74).

Table 3. Summary of the Simple linear regression models (SLR).

SLR: $lm\ (t_{mig} \sim Predictor)$	R^2	R^2_{Adj}	$Res_{st.err}$ on 658 df	$Pred_{st.err}$	$Perc_m$
$INST$	0.7405	0.7401	17.05	15.35	30.79
$L3_{miss}$	0.6572	0.6566	19.6	12.07	20.6
$DirtyPages_{server}$	0.7128	0.7123	17.94	16.89	20.51
MAR	0.1589	0.1576	30.7	30.96	29.21
CPU_{util_server}	0.0065	0.005	33.36	34.15	37.06
MEM_{server}	0.03708	0.03561	32.85	33.68	39.35
$MEMtoBW_{server}$	0.07824	0.07684	32.14	33.41	29.21
$DirtyPages_{server\_per\_sec}$	0.0707	0.06929	32.27	31.74	31.22
BW	0.01185	0.01035	33.27	33.94	31.14
CPU_{util_vm}	0.0098	0.0083	33.31	34.03	37.8
MEM_{vm}	0.2097	0.2085	29.76	29.23	26.23
$MEMtoBW_{vm}$	0.2451	0.2439	29.08	29.33	25.46

Fig. 3 displays the linear dependence of migration time on $INST$. The black line is the best fit regression line (trained model) that regresses t_{mig} on $INST$. The expression for the SLR model is:

$$t_{mig} = -11.1 + 2.9 \times 10^{-4} \times INST\ . \tag{8}$$

where the intercept of -11.1 is just an adjustment constant; the regression coefficient 2.9×10^{-4} implies the expected increase of 2.9×10^{-4} seconds in migration time for a unit increase in the instructions retired. Thus, when the number of CPU instructions retired increases during migration by 100000, the total migration time is expected to increase by 29 seconds. The residual standard error $Res_{st.err}$ of the model on 658 degrees of freedom equals to 17.05 seconds. The prediction error of the trained model on the testing data (standard error of the estimate from a sample of 220 observations) equals to 15.35 seconds. But its prediction accuracy (the mean absolute percentage error) on the testing data is still quite low and equals to 30.79%.

The linear regression model which produced the second highest R^2 value (0.71) is the one that relates migration time with the number of "dirty" pages observed in the source server during migration ($DirtyPages_{server}$). The residual standard error and the prediction error of the model are 17.94 and 16.89 seconds, respectively. Though, its prediction accuracy on the testing dataset is better ($Perc_m$ = 20.51%). Fig. 4 shows the linear dependency of the total migration time on the total number of "dirty" pages observed at the source server during migration. The relationship is expressed as follows:

$$t_{mig} = -0.78 + 0.94 \times DirtyPages_{server}\ . \tag{9}$$

Consequently, migration time increases by 0.94 seconds when the number of dirty pages increases by a unit value (which corresponds to an update of 4 KB of memory at the source host). The model that relates migration time with the

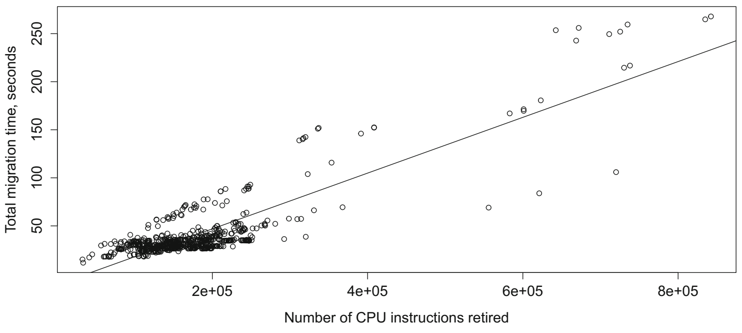

Fig. 3. Dependence of the total migration time on the total number of instructions retired during migration process. Regression line: $t_{mig} = -11.1 + 2.9 \times 10^{-4} \times INST$

total number of last level (L3) cache line misses resulted in the third highest R^2 value (0.65). The residual standard error and the prediction error of the model are 19.6 seconds and 12.07 seconds, respectively. Its mean absolute percentage error equals to 20.6%. Fig. 5 shows the dependency between the total migration time and the last level (L3) cache line misses. Accordingly, the dependence of t_{mig} on $L3_{miss}$ can be described by:

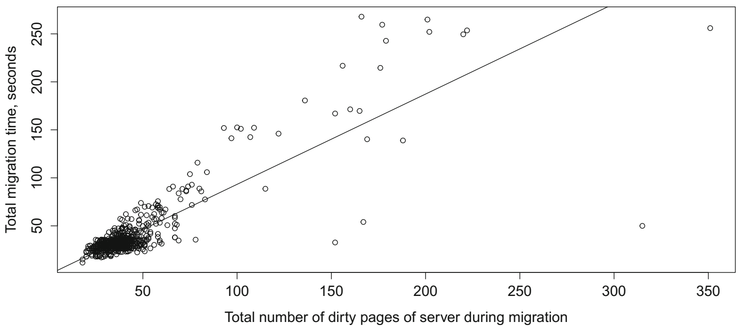

Fig. 4. Dependence of the total migration time on the "dirty" pages observed in the source server during migration. Regression line: $t_{mig} = -0.78 + 0.94 \times DirtyPages_{server}$.

$$t_{mig} = 25.2 + 8 \times 10^{-8} L3_{miss} \ . \tag{10}$$

The R^2 of models with all the other independent variables is so small that they cannot be considered alone to estimate migration time. In fact, the residual and prediction errors of the SLR models with even the best predictors are considerably high and their prediction accuracy is low ($Perc_m$ exceeds 20%) that none of the SLR models is adequate to estimate migration time.

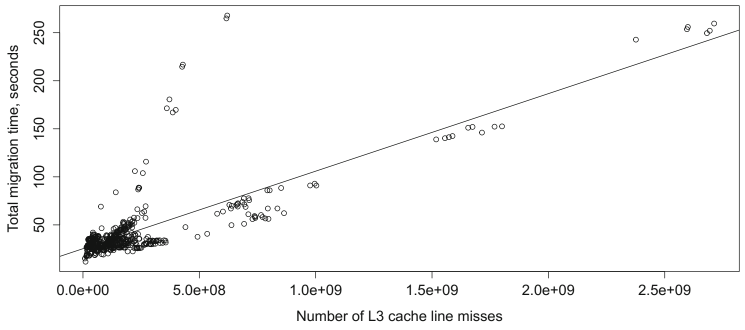

Fig. 5. Dependence of the total migration time on the last level (L3) cache line misses observed during the VM migration. Regression line: $t_{mig} = 25.2 + 8 \times 10^{-8} L3_{miss}$.

When CPU instructions retired and the last level cache line misses are considered in combination, they captured 92% of the variance of t_{mig}. Fig. 6 shows the dependency of the total migration time on the two independent variables. The dashed plane is the regression plane which fits the measured data best and thus, minimises the sum of squared residuals. The expression that describes the multiple regression model with two independent variables is given as:

$$t_{mig} = -5.1 + 2.03 \times 10^{-4} INST + 4.98 \times 10^{-8} L3_{miss} \ . \tag{11}$$

The residual standard and prediction errors of the model are 9.17 seconds and 8.02 seconds, respectively. Its prediction accuracy is significantly improved as well ($Perc_m$ equals to 15.18%). However, combining other independent variables in the same way does not always produce the same remarkable improvement. For example, a combination of $MEMtoBW_{vm} + DirtyPages_{server}$ did not significantly improve R^2 that was achieved by $DirtyPages_{server}$ alone, though its prediction accuracy was still improved by 1.58%. Table 4 provides a summary of the estimation improvements we observed with multiple regression for these two cases. Each of the parameters in all presented MLR models are significant with p-value lower than the smallest significance level (0.001).

Table 4. Simple vs. Multiple linear regression models with two predictors.

SLR versus MLR: $lm\ (t_{mig} \sim Predictor(s))$	R^2	R^2_{Adj}	$Res_{st.err}$	$Pred_{st.err}$	$Perc_m$
$INST$	0.741	0.7401	17.05	15.35	30.79
$L3_{miss}$	0.657	0.6566	19.6	12.07	20.6
$INST + L3_{miss}$	**0.925**	**0.9248**	**9.171**	**8.02**	**15.18**
$MEMtoBW_{vm}$	0.245	0.2439	29.08	29.33	25.46
$DirtyPages_{server}$	0.713	0.7123	17.94	16.89	20.51
$MEMtoBW_{vm} + DirtyPages_{server}$	**0.744**	**0.7436**	**16.94**	**16.25**	**18.93**

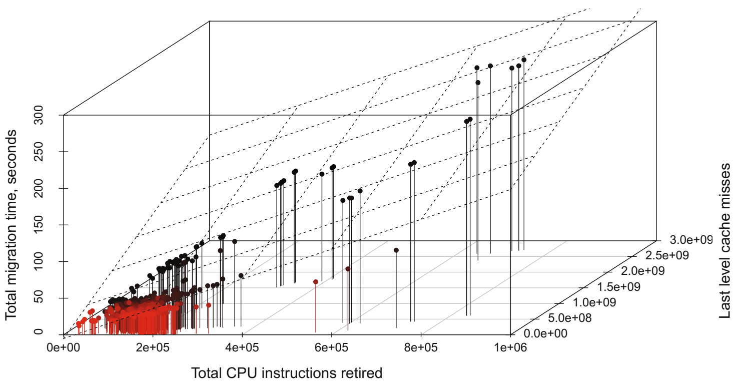

Fig. 6. 3D scatter plot with regression plane showing the dependence of the total VM migration time on CPU instructions retired and last level cache line misses. Multiple linear regression model: $t_{mig} = -5.1 + 2.03 \times 10^{-4} INST + 4.98 \times 10^{-8} L3_{miss}$.

4.3 Multiple Linear Regression Models

As the number of independent variables in a linear regression model increases, the strength of the model in accounting for the variance in migration time increases. Understandably, the model's complexity increases too. It is also possible that the model gets over-fitted and, as a result, looses its prediction power. Therefore, care must be taken to strike the right balance between expressiveness, complexity, and potential over-fitting. For this purpose, we identified the five most significant independent variables that can be combined together. These are: (1) Instructions retired, (2) last level cache line misses, (3) total "dirty" pages at the source server, (4) ratio of active memory used by the source server to network bandwidth, and (5) average CPU utilisation of the source server during migration.

We employed *all subsets regression* [13] in order to identify the best multiple linear regression model that balanced estimation accuracy with complexity. The method examines all possible models and compares the gain in the adjusted R-square. Since we have five independent variables, the *all subsets regression* considers all possibles models with one, two, three, four, and five independent variables which corresponds to 31 possible models. Fig. 7 depicts one best model for each subset size (one, two, three, four, and five independent variables) with respect to adjusted R-square measure.

As we already mentioned above, the best SLR model with respect to R^2_{Adj} is the one using $INST$ (depicted here as TI); the best MLR model consisting of two independent variables is the one using $INST$ and $L3_{miss}$ (depicted here as TL-TI); the best MLR model consisting of three independent variables is the one using $INST$, $L3_{miss}$, and $DirtyPages_{server}$ (depicted here as TL-TI-TD). Its R^2_{Adj} is 0.935 and $Perc_m$ equals to 13.39%. The adjusted R-square of the best model with four independent variables is 0.943 and it consists of $INST$, $L3_{miss}$, $DirtyPages_{server}$ and CPU_{util_server} as independent variables.

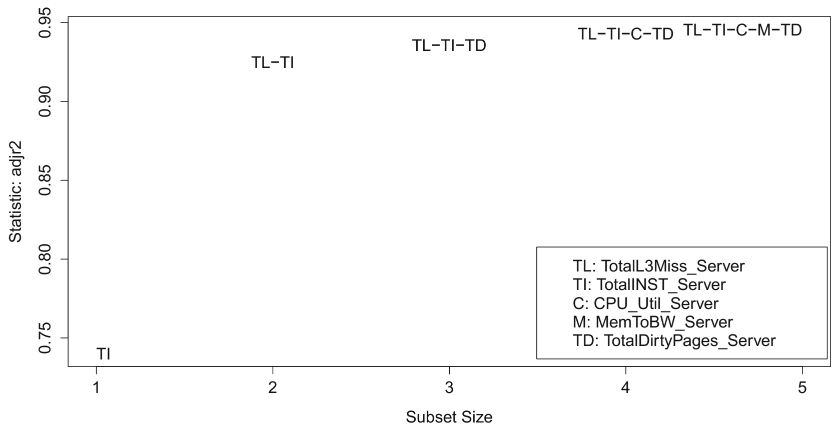

Fig. 7. Defining the best models with all subsets regression method.

Its prediction accuracy is also improved ($Perc_m = 12.12\%$). The model with all the five independent variables has R^2_{Adj} which is equal to 0.946. Its mean absolute percentage error is the lowest and equals to 10.14%. The residual standard error of the model on the 654 degrees of freedom is 7.8 seconds and the standard error of estimate on the testing data is comparatively low, namely, 5.4 seconds. The linear equation for the model consisting of all the five independent variables is given as:

$$t_{mig} = 32.5 + 4.52 \times 10^{-8} L3_{miss} + 1.8 \times 10^{-4} INST - 0.9 \times CPU_{util\_server} + \\ +0.17 \times DirtyPages_{server} - 0.29 \times MEMtoBW_{server} \ . \quad (12)$$

Table 5 summarises the adjusted R-square values, the residual standard errors, the standard errors of estimate for a dataset of 220 observations, and the mean absolute percentage errors of the best models with three, four, and five independent variables.

Table 5. The best Multiple linear regression models with three, four, and five independent variables.

The best Multiple linear regression models lm ($t_{mig} \sim Predictors$)	R^2	R^2_{Adj}	$Res_{st.err}$	$Pred_{st.err}$	$Perc_m$
$L3_{miss} + INST + DirtyPages_{server}$	0.936	0.9356	8.487	7.028	13.39
$L3_{miss} + INST + CPU_{util\_server} + DirtyPages_{server}$	0.943	0.943	7.984	5.811	12.12
$L3_{miss} + INST + DirtyPages_{server} + CPU_{util\_server} + MEMtoBW_{server}$	**0.946**	**0.9456**	**7.802**	**5.379**	**10.14**

To further assess the generalisability of the best model with five independent variables we realised a *10-fold cross-validation* of R^2 [15]. It allows us to see how well the model will perform on the unseen (testing) data. This method divides the training data into 10 sub-samples each of which serves as a testing group and the remaining 9 sub-samples (training group) are used to train the model. The performance (R^2) for each of the 10 prediction equations applied to the 10 testing groups is recorded and averaged, which gives us a new metric, namely *10-fold* cross-validated R^2 [13]. The results of the *10-fold cross-validation* of the best model with five independent variables are as follows: original R^2 equals to 0.946, 10-fold cross-validated R^2 is equal to 0.937. Thus, the difference is very small (0.009) and the model is performing well on the unseen data.

Adding a sixth independent variable (for instance, the ratio of active memory used by the VM to network bandwidth ($MEMtoBW_{VM}$) or the average CPU utilisation of the VM did not improve the adjusted R-square appreciably. Fig. 8 compares the relative importance (relative weights) of the independent variables in producing $R^2 = 0.94$ in the best MLR model with five independent variables – The CPU instructions retired (depicted in the plot as $TotalINST\_Server$) contributed 37.8%, the last level cache misses contributed 31.4%, number of “dirty” pages contributed 27.6%. The CPU utilisation of the source server and the ratio of active memory to network bandwidth contributed 0.7% and 2.3%, respectively. Thus, we can see that $TotalINST\_Server$ is the most important independent variable in estimating the migration time. The code for calculating the relative weights was adapted from Kabacoff [13].

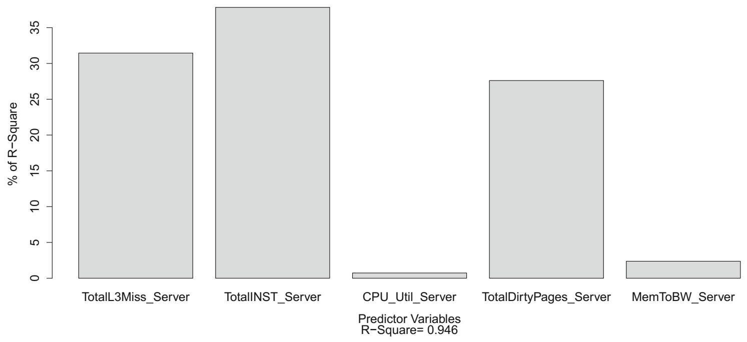

Fig. 8. Relative importance of independent (predictor) variables.

The distributions of the errors of the model built using training dataset and the distribution of its prediction errors on the testing dataset are displayed in Fig. 9 and Fig. 10, respectively. Both distributions tend to form a normal distribution curve with 0 mean. The prediction errors depicted in Fig. 10 are obtained by subtracting from the real values of the migration time of the testing dataset the values predicted by the model. Hence, the error is measured

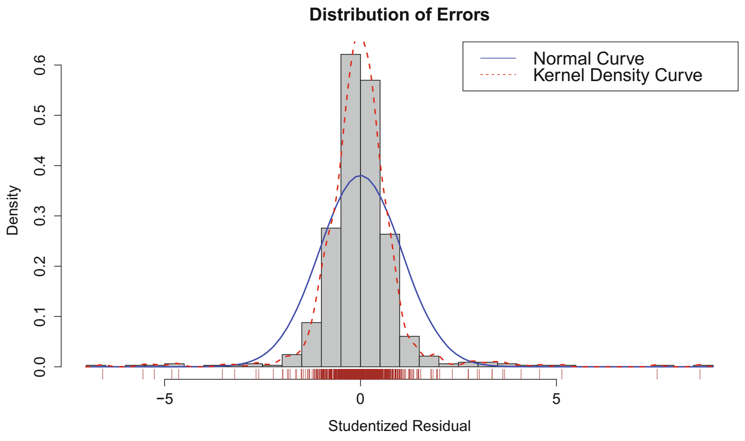

Fig. 9. Normality test of the trained model.

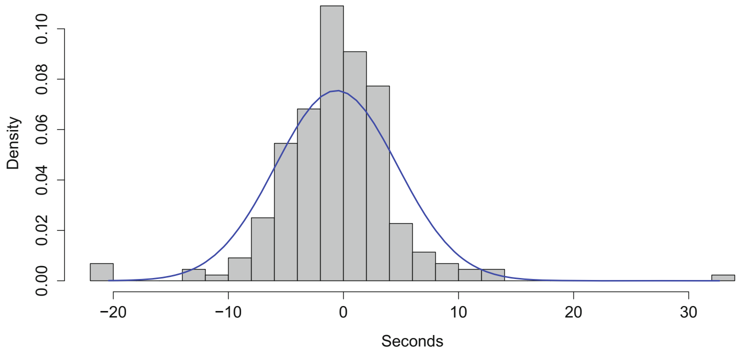

Fig. 10. Distribution of the prediction errors of the model.

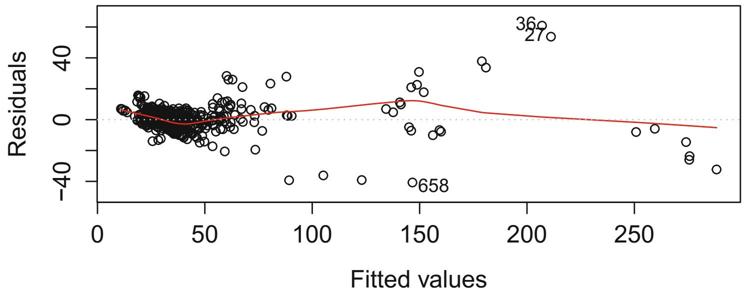

Fig. 11. Diagnostic plot of the linearity assumption.

in seconds. Finally, Fig. 11 illustrates the assumption pertaining to the existence of linearity between the dependent and independent variables for the best MLR model with five independent variables. If a model satisfies the linearity assumption, there will be no systematic relationship between the residuals and the fitted values [13]. It can be shown from the figure that our model satisfies this requirement. Though, due to high variances in migration time introduced mainly by *mcf* and *astar* memory intensive benchmarks the model faces the problem of non-constant variance which is often the case in practice. Nevertheless, the non-constant variance is not substantial in this case because the model behaves well on the testing data and its prediction accuracy equals to 10.12%. In order to satisfy additionally constant variance assumption in our future work we are planning to 1) build separate MLR models for CPU intensive benchmarks only and 2) investigate additional techniques such as weighted least squares.

5 Conclusion and Future Work

This paper extensively discussed migration time as a consequence of dynamic server consolidation in data centres and virtualised environments. We experimentally demonstrated that migration time can be adequately expressed as a linear combination of a few number of resource utilisation parameters, particularly, in terms of the total number of retired CPU instructions, the total number of L3 cache line misses, and the number of "dirty" pages observed in the source server during migration. By employing various simple and multiple linear regression models we closely and quantitatively examined the significance of these parameters in reducing residual and prediction errors as well as in improving R^2 and R^2_{Adj}. Furthermore, we experimentally showed that the expressive power and the complexity of the models depended on the number of independent variables they include. However, we also showed that increasing the number of independent variable beyond five could not appreciably increase the strength of the models.

This paper mainly focused on migration time. In future we will be considering other costs introduced by virtual machine migration, such as the energy overhead and how these costs can be estimated using a single unifying model.

Acknowledgments. This work is supported in a part by the German Research Foundation (DFG) in the Collaborative Research Center 912 Highly Adaptive Energy-Efficient Computing.

References

1. Barroso, L.A., Holzle, U.: The Case for Energy-Proportional Computing. IEEE Computer **40**(12), 33–37 (2007)
2. Dargie, W.: Analysis of the power consumption of a multimedia server under different DVFS policies. In: Fifth International Conference on Cloud Computing, Honolulu, HI, USA, pp. 779–785 (2012)

3. Akoush, S., Sohan, R., Rice, A., Moore, A.W., Hopper, A.: Predicting the performance of virtual machine migration. In: 2010 IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 37–46 (2010)
4. Liu, H., Xu, C., Jin, H., Gong, J., Liao, X.: Performance and energy modeling for live migration of virtual machines. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing, pp. 171–182. ACM, New York (2011)
5. Wu, Y., Zhao, M.: Performance modeling of virtual machine live migration. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 492–499 (2011)
6. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation, Vol. 2, pp. 273–286. USENIX Association, Berkeley (2005)
7. Strunk, A.: A lightweight model for estimating energy cost of live migration of virtual machines. In: 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), pp. 510–517 (2013)
8. Verma, A., Kumar, G., Koller, R., Sen, A.: CosMig: modeling the impact of reconfiguration in a cloud. In: 19th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) 2011, pp. 3–11. IEEE (2011)
9. SPEC CPU2006 Benchmark Descriptions. https://www.spec.org/cpu2006/publications/CPU2006benchmarks.pdf
10. Intel performance monitoring counters. https://software.intel.com/en-us/blogs/2014/07/24/developer-api-documentation-for-intel-performance-counter-monitor
11. Linux Dirty statistic from proc/meminfo/. http://www.centos.org/docs/5/html/5.2/Deployment_Guide/s2-proc-meminfo.html
12. Intel PCM column names. https://software.intel.com/en-us/blogs/2014/07/18/intel-pcm-column-names-decoder-ring
13. Kabacoff, R.I.: R in Action: Data Analysis and Graphics with R. MANNING Shelter Island (2011)
14. Baron, M.: Probability and Statistics for Computer Scientists, 2nd edn. CRC Press Taylor and Francis Group, New York (2014)
15. Core Team, R.: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna (2014). http://www.R-project.org/
16. Poellabauer, C., Singleton, L., Schwan, K.: Feedback-based dynamic voltage and frequency scaling for memory-bound real-time applications. In: 11th IEEE Symposium on Real Time and Embedded Technology and Applications, RTAS 2005, pp. 234–243 (2005)
17. Jaleel, A.: Memory characterization of workloads using instrumentation-driven simulation. http://www.jaleels.org/ajaleel/publications/SPECanalysis.pdf