Noname manuscript No.
(will be inserted by the editor)

Empirical Study on the Discrepancy between Performance Testing Results from Virtual and Physical Environments

Muhammad Moiz Arif · Weiyi Shang · Emad Shihab

Received: date / Accepted: date

Abstract Performance plays an indispensable role for software reliability. Performance tests are typically conducted in order to ensure the performance of software systems. Such performance tests often consume large amounts of computing resources, such as long running time in the performance testing environments. Making it worse, the ever evolving field environment requires frequently updating the performance testing environment. In practice, virtual machines (VMs) are widely exploited to provide flexible and less costly environments for performance tests. However, the use of VMs may introduce extra overheads (e.g., a higher than expected memory utilization) to the testing environment, leading to un-realistic performance testing results. Yet, to the best of our knowledge, little research has studied such overhead in the context of performance testing activities, nor has investigated the impact on performance testing results.

In this paper, we perform a case study on two open source systems, i.e., Dell DVD Store and CloudStore, to evaluate the discrepancy between the performance testing results from virtual and physical environments. We conduct the same performance tests in both virtual and physical environments. In order to evaluate the discrepancy, we compare the performance testing results based on the three aspects that are typically examined for performance testing results: 1) individual performance metric, 2) relationship among performance metrics and 3) performance models that are build from the performance metrics. Our results illustrate the discrepancy between performance testing results from virtual and physical environments. In particular, individual performance metrics from virtual and physical environments do not share the same shape of distribution. The correlations among performance metrics in virtual environment are different from the correlations in physical environment. Finally, the statistical models that are built based on the performance counters from virtual envi-

Muhammad Moiz Arif , Weiyi Shang, Emad Shihab
Department of Computer Science and Software Engineering
Concordia University
Montreal, Quebec, Canada
E-mail: {mo_ari, shang, eshihab}@cse.concordia.ca

ronment are different from the models that are build from the physical environment. Normalizing performance metrics based on deviance may reduced such discrepancy. Practitioners should be aware of such discrepancy and may leverage normalization techniques when analyzing performance testing results from virtual environments.

1 Introduction

Software performance engineering engulfs performance assurance activities during the development and operation of software systems. These activities ensure that the software meets the desired performance requirements [49]. Software performance engineering plays a vital role in the reliability of large software systems in the field. Failures in large software systems that are due to performance issues are common [18, 19], leading to eventual decline in the quality of the system with reputational issues and monetary and temporal losses [2]. For instance, Amazon estimates that a one-second page-load slowdown can cost up to $1.6 billion [3].

In order to mitigate performance issues and ensure software reliability, practitioners often conduct performance tests [49]. Performance tests apply a workload (e.g., mimicking users' behavior in the field) on the software system [5, 46], and monitor performance metrics, such as CPU usage, that are generated from the tests. Practitioners use such metrics to gauge the performance of the software system and identify potential performance issues (such as memory leaks [45]) and throughput bottlenecks [32].

Since performance tests are often performed on large-scale software systems, the performance tests often require many resources. Moreover, performance tests often need to run for a long period of time in order to build statistical confidence on the results [5]. In addition, such testing environment needs to be easily configurable such that a specific environment can be mimicked, reducing false performance issues that are due to environmental issues. Therefore, to address such challenges, virtual environments (i.e., VMs) are often leveraged for performance testing [1, 12, 13]. The flexibility of virtual environments enables practitioners to easily prepare, customize and run/update performance testing environments in an efficient manner.

However, a major question that lingers is that **are performance tests executed in a virtual environment representative of what happens in the physical environment?**. This question is particularly important since 1) virtual environments are highly leveraged in practice [38, 50] and 2) prior work has shown that using virtual machines imposes a hidden overhead that is rarely considered [36], impacting the reliability of performance test results performed in virtual environments. To the best of our knowledge, the discrepancy between performance testing results in virtual and physical environments has rarely been studied. Evaluating and quantifying such discrepancy would help practitioners and researchers in the area of software performance and reliability better understand how their results from virtual environments apply to the physical environment.

In this paper, we perform a study where performance tests are conducted on virtual and physical environments. Our study focuses on the discrepancy between the two environments and highlights how to address them. We compare performance

testing results from virtual and physical environments based on the three widely examined aspects, i.e., individual performance metric, the relationship between the performance metrics and models that predict performance. Our study is conducted on two open-source systems, DS2 [17] and CloudStore [14].

We find that 1) performance metrics have different shapes of distributions in virtual environments compared to physical environments, 2) there are large differences in correlations among performance metrics measured in virtual and physical environments, and 3) statistical models using performance metrics from virtual environments do not apply to physical environments (i.e., produce high prediction error) and vice versa. Then, we examined the feasibility of using normalizations to help alleviate the discrepancy between performance metrics. We find that in some cases, normalizing performance metrics based on deviance may reduce the prediction error when using performance metrics collected from one environment and applying it on another. Our findings show that practitioners cannot assume that their performance tests that are observed on one environment will necessarily apply to another environment. We do not only discuss the discrepancy but also more importantly we analyze the impact of discrepancy on performance testing results. Moreover, practitioners who leverage both, virtual and physical environments, need to apply normalization technique to reduce the discrepancy that may arise due to the environment (i.e., virtual vs. physical).

The rest of the paper is organized as follows. Section 2 presents the background and related work to this paper. Section 3 presents the case study step. Section 4 presents the results of our cases study, followed by a discussion of our results in Section 5. Section 6 discusses the threats to validity of our findings. Finally, Section 7 concludes this paper.

2 Background and Related Work

The work that is most related to the paper can be categorized into two main areas: 1) analyzing performance metrics from performance testing, using individual performance metrics, relationship among performance metrics, and statistical modelling based on performance metrics, and 2) analysis of VM overhead. Here, we discuss the state-of-the-art and how our work relates to prior work.

2.1 Analyzing performance metrics from performance testing

Individual performance metrics

Nguyen *et al*. [38] introduce the concept of using control charts [43] in order to detect performance regression. Control charts use a predefined threshold to detect performance anomalies. However control charts assume that the output follows a uni-model distribution which may be an inappropriate assumption for performance. Nguyen *et al*. propose an approach to normalize performance metrics between heterogeneous environments in order to build robust control charts.

Malik *et al*. [33, 34] propose approaches that cluster performance metrics using Principal Component Analysis (PCA). Each component generated by PCA is mapped

to performance metrics by a weight value. The weight value measures how much a metric contributes to the component. For every performance metric, a comparison is performed on the weight value of each component to detect performance regressions.

Heger *et al*. [23] present an approach that uses software development history and unit tests to diagnose the root cause of performance regressions. In the first step of their approach, they leverage Analysis of Variance (ANOVA) to compare the response time of the system to detect performance regressions. Similarly, Jiang *et al*. [27] extract response time from system logs. Instead of conducting statistical tests, Jiang *et al*. visualize the trend of response time during performance tests, in order to identify performance issues.

Relationship among performance metrics

Malik *et al*. [32] leverage Spearman's rank correlation to capture the relationship among performance metrics. The deviance of correlation is calculated in order to pinpoint which subsystem should take responsibility of performance deviation.

Foo *et al*. [19] propose an approach that leverages association rules in order to address the limitations of manually detecting performance regression in large scale software systems. Association rules capture the historical relationship among performance metrics and generate rules based on the results of prior performance tests. Deviation in the association rules are considered signs of performance regressions.

Jiang *et al*. [25] use normalized mutual information as a similarity measure to cluster correlated performance metrics. Since metrics in one cluster are highly correlated, the uncertainty among metrics in the cluster should be low. Jiang *et al*. leverage entropy from information theory to monitor the uncertainty of each cluster. A significant change in the entropy is considered as a sign of a performance fault.

Statistical modelling based on performance metrics

Xiong *et al*. [50] proposed a model-driven approach named *vPerfGuard* to detect software performance regressions in a cloud-environment. The approach builds models between workload metrics and a performance metric, such as CPU. The models can be used to detect workload changes and assist in identifying performance bottlenecks. Since the usage of *vPerfGuard* is typically in a virtual environment, our study may help the future evaluation of *vPerfGuard*. Similarly, Shang *et al.* [42] propose an approach of including only a limited number of performance metrics for building statistical models. The approach leverages an automatic clustering technique in order to find the number of models to be build for the performance testing results. By building statistical models for each cluster, their approach is applicable to detect injected performance regressions.

Cohen *et al*. [15] propose an approach that builds probabilistic models, such as Tree-Augmented Bayesian Networks, to examine the causes that target the changes in the system's response time. Cohen *et al*. [16] also proposed that system faults can be detected by building statistical models based on performance metrics. The approaches of Cohen *et al*. [15,16] were improved by Bodik *et al.* [9] by using logistic regression models.

Jiang *et al*. [26] propose an approach that improves the Ordinary Least Squares regression models that are built from performance metrics and use the model to detect faults in a system.

In our work, we compare performance testing results from both virtual and physical environments based on all the above three types of analyses. Our findings can help better evaluate and understand the findings from the aforementioned research.

2.2 Analysis of VM overheads

Kraft *et al*. [30] discuss the issues that are related to disk I/O in a virtual environment. They examine the performance degradation of disk request response time by recommending a trace-driven approach. Kraft *et al.* [30] emphasize on the latencies existing in virtual machines request for disc IO due to increment in time associated with request queues.

Aravind *et al*. [36] audit the performance overheads in Xen virtual machines. They uncover the origins of overheads that might exist in the network I/O causing a peculiar system behavior. However, there study is limited to Xen virtual machine only while mainly focusing on network related performance overheads.

Brosig *et al*. [10] predict the performance overheads of virtualized environments using Petri-nets in Xenserver. The authors worked on visualization overheads with respect to queuing networks only.

Huber *et al*. [24] present a similar kind of study between cloud-like environments. They compare the performance of virtual environments and study the degradation between the two. They further use regression based models to investigate CPU and memory performance. However, to calculate performance overhead they scale their model between two virtualized environments. On the contrary, we investigate whether there exists a uniformity between the relationships in physical and virtual environments.

where does the following related work fits in?

Luo *et al*. [31] converge the set of inputs that may cause software regression. They apply genetic algorithms to detect such combinations.

Prior research on virtual environment overhead is rather general and is based on operating system point of view without considering the impact of such overhead on software reliability ensuring tasks. In this paper, we evaluate the discrepancy between virtual and physical environments by focusing on the impact of performance testing results and answer that whether such discrepancy will specifically impact the results and practices in software performance testing.

3 Case Study Setup

In this section, we present our case study setup. The goal of our case study is to evaluate the discrepancy between performance testing results from virtual and physical environments. An overview of our case study setup is shown in Figure 1.

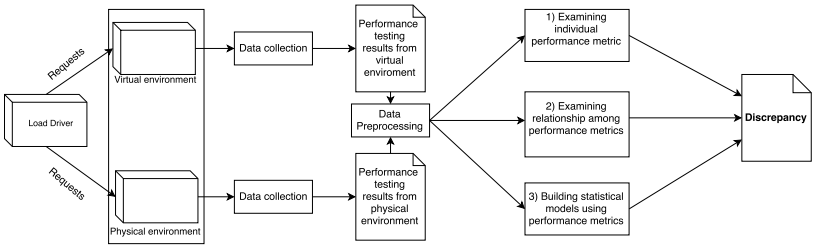

Fig. 1: An overview of our case study setup.

3.1 Subject Systems

Dell DVD Store (DS2) [17] is an online multi-tier e-commerce web application that are widely used in performance testing and prior performance engineering research [27, 38, 42]. We deploy DS2 on an Apache (Version 3.0.0) web application server with MySQL 5.6 database server [40]. CloudStore [14], our second subject system is an open source application based on the TPC-W benchmark [47]. CloudStore is widely used to evaluate the performance of cloud computing infrastructure when hosting web-based software systems and is leveraged in prior research [6]. We deploy CloudStore on *Apache Tomcat* [8] (version 7.0.65) with MySQL 5.6 database server [40].

3.2 Environmental Setup

The performance tests of the two subject systems are conducted on three machines in a lab environment. Each machine has an Intel i5 4690 Haswell Quad-Core 3.50 GHz CPU, with 8 GB of memory, connected to a local gigabyte ethernet. The first machine hosts the web server and application server (Apache and Tomcat). The second machine hosts the MySQL 5.6 database server. The load drivers (e.g., JMeter) was deployed on the third machine. We separate the load driver, the web/application server and the database server on different machines in order to avoid interference among these processes. The operating systems on the three machines are Windows 7. We disable all other processes and unrelated system services to minimize their performance impact. Since our goal is to compare performance metrics in virtual and physical environments, we setup the two different environments, which we detail next.

Virtual environment. We install one Virtual Box (version 5.0.16) and create only one virtual machine on one physical machine to avoid the interference between virtual machines. For each virtual machine, we allocate two cores and 3GB of memory.

Physical environment. To make the physical environment similar to the virtual environment, we only enable two cores and 3GB memory for each machine for the physical environment.

3.3 Performance tests

DS2 is released with a dedicated load driver program that is designed to exercise DS2 for performance testing. We used the load driver to conduct performance testing on DS2. We used Apache JMeter [7] to generate a workload for conducting performance tests on CloudStore. For both subject systems, the workload of the performance tests is variated periodically in order to avoid bias from a consistent workload. The workload variation was introduced by the number of threads. A higher number of threads represented a higher number of users accessing the system. Each performance test is run after a 15 minutes warming up period of the system and lasts for 9 hours.

3.4 Data collection and preprocessing

Performance metrics. We used *PerfMon* [37] to record the values of performance metrics. *PerfMon* is a performance monitoring tool used to observe and record performance metrics such as CPU utilization, memory usage and disk IOs. We record all the available performance metrics that can be monitored on a single process by *PerfMon*. We recorded the performance metrics of both the processes, web server and the database server, with an interval of 10 seconds. In total, we recorded 44 performance metrics.

System throughput. We used the web server access logs from Apache and Tomcat to calculate the throughput of the system by measuring the number of requests per minute. The two data sets were then concatenated and mapped against requests using their respective timestamps.

In order to combine the two datasets of performance metrics and system throughput, and to minimize noise of performance metrics recording, we calculate the mean values of the performance metrics in every minute and combine the datasets of performance metrics and system throughput based on the time stamp on a per minute basis. A similar approach has been applied to address mining performance metrics challenges [19].

4 Case Study Results

The goal of our study is to evaluate the discrepancy between performance testing results from virtual and physical environments, particularly considering the impact of discrepancy on the analysis of such results. We do not predict the applications' performance or throughput based on the amount of workdone by the underlying architecture. Instead, our experiments are set in context of analyzing performance testing data, based on the related work. Shown in Section 2, prior research examine performance testing results in three types of approaches: 1) examining individual performance metric, 2) examining relationship among performance metrics and 3) building statistical models using performance metrics. Therefore, our experiments focus on examining the discrepancy that may impact on such three approaches.

4.1 Examining individual performance metric

Approach.

First, we compare every individual performance metric between virtual and physical environments. Since the performance tests are conducted in different environments, intuitively the scales of performance metrics are not the same. For example, the virtual environment may have higher CPU usage than the physical environment. Therefore, instead of comparing the values of each performance metric in both environments, we study whether the performance metric follow the same shape of distribution and the same trend in virtual and physical environments.

First, we plot a quantile-quantile plot (Q-Q plot) [39] for every performance metric in two environments. A Q-Q plot is a plot of the quantiles of the first data set against the quantiles of the second data set. We also plot a 45-degree reference line on the Q-Q plots. If the performance metrics in both environments follow the same shape of distribution, the points on the Q-Q plots should fall approximately along this reference (i.e., 45-degree) line. A large departure from the reference line indicates that the performance metrics in the virtual and physical environments come from populations with different shapes of distributions.

Second, to quantitatively measure the discrepancy, we perform a Kolmogorov-Smirnov test [44] between every performance metric in the virtual and physical environments. Since the scales of each performance metric in both environments are not the same, we first normalize the metrics based on their median values and their mean absolute deviation:

$$M_{normalized} = \frac{M - \tilde{M}}{MAD(M))} \tag{1}$$

where $M_{normalized}$ is the normalized value of the metric, M is the original value of the metric, $\tilde{M}$ is the median value of the metric and $MAD(M)$ is the median absolute deviation of the metric [48]. The Kolmogorov-Smirnov test gives a p-value as the test outcome. A p-value ≤ 0.05 means that the result is statistically significant, and we may reject the null hypothesis (i.e., two populations are from the same distribution). By rejecting the null hypothesis, we can accept the alternative hypothesis, which tells us the performance metrics in virtual and physical environments do not have the same distribution. We choose Kolmogorov-Smirnov test since it does not have any assumption on the distribution of the metrics.

Finally, we calculate Spearman's rank correlation between every performance metric in the virtual environment and and the corresponding performance metric in the physical environment, in order to assess whether the same performance metric in two environments follow the same trend during the test. Intuitively, two sets of performance testing results without discrepancy should show similar trend, i.e., when memory keeps increasing in the physical environment (like memory leak), the memory should also increase in the virtual environment. We choose Spearman's rank correlation since it does not have any assumption on the distribution of the metrics.

Results.

Most performance metrics do not follow the same shape of distribution in virtual and physical environments. Figure 2 and 3 show the Q-Q plots by comparing the quantiles of performance metrics from virtual and physical environments.

Due to the limited space, we only present Q-Q plot for CPU user time, IO data operations/sec and memory working set for both web sever and database server[1]. The results show that the lines on the Q-Q plot are not close to the 45-degree reference line. By looking closely on the Q-Q plots we find that the patterns of each performance metric from different subject systems are different. For example, the web cpu user time for DS2 in the virtual environment shows higher values than in the physical environment at the median to high range of the distribution; while the Q-Q plot of CloudStore shows web cpu user time with higher values at the low range of the distribution. In addition, the lines of the Q-Q plots for database memory working set show completely different shapes in DS2 and in CloudStore. The results imply that the discrepancies between virtual and physical environments are different across subject systems. The impact of the subject systems will be discussed in future work.

The majority of the performance metrics have statistically significantly different distribution (p-values lower than 0.05 in Kolmogorov-Smirnov tests). Only 13 and 12 metrics have p-values higher than 0.05, for DS2 and CloudStore, respectively, showing statistically in-significant difference between the distribution in virtual and physical environments. By looking closely at such metrics, we find that these metrics either do not highly relate to the execution of the subject system (e.g., web server CPU privileged time in DS2), or highly relate to the workload. Since the workload between the two environments are similar, it is expected that the metrics related to the workload follow the same shape of distribution. For example, the I/O operations are highly related with the workload. The metrics related to I/O operations may show statistically in-significant difference between the distribution in virtual and physical environments (e.g., web server I/O write operations per second in DS2).

Most performance metrics do not have the same trend in virtual and physical environments. Table 1 shows the Spearman correlation coefficient and corresponding p-value between the selected performance metrics as Q-Q plots. We find that for the web server memory working set in CloudStore and the database server memory working set in DS2, there exists strong (0.69) to moderate (0.46) correlation between virtual and physical environments, respectively. By examining the metrics, we find that both metrics have an increasing trend that may be caused by a memory leak. Such increasing trend may be the cause of the moderate to strong correlation. Instead of showing the selected metrics as the Q-Q plots, Table 2 shows a summary of the spearman's rank correlation of all the performance metrics. Most of the correlations have an absolute value of 0 to 0.3 (low correlation), or the correlation is not statistically significant (p-val>0.05).

Performance metrics typically do not follow the same distribution or follow the same trend in virtual and physical environments. System load metric (throughput) and some performance metrics that are correlated with system load may be exceptions.

4.2 Examining relationship among performance metrics

Approach.

[1] The complete results are shared online at http://das.encs.concordia.ca/wp-content/uploads/2016/04/Arif_results.zip

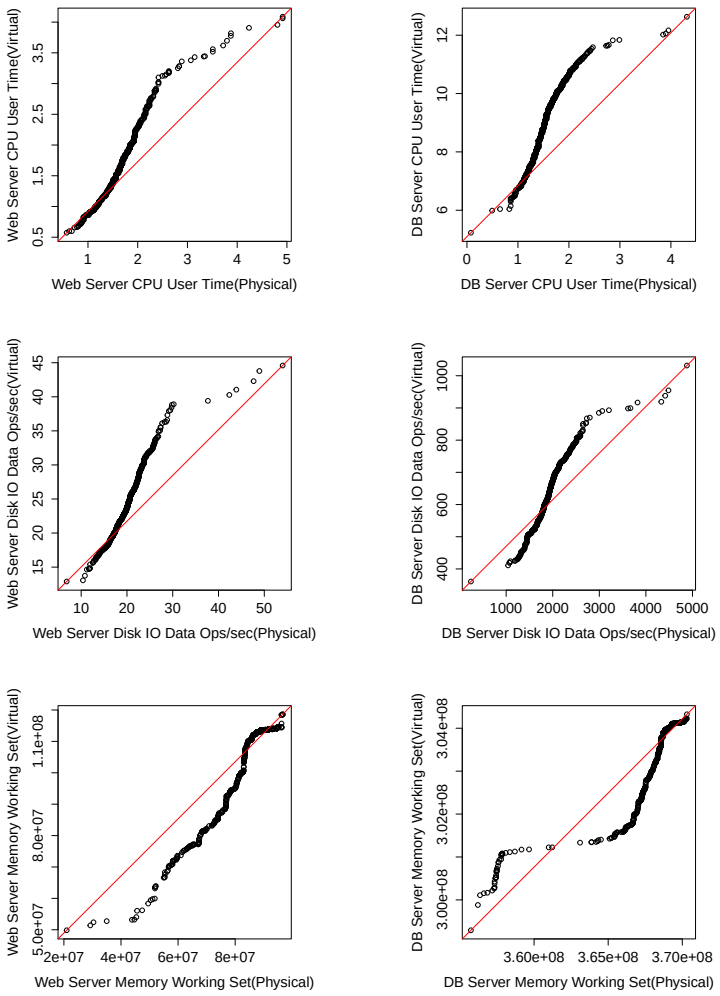

Fig. 2: Q-Q plots for DS2.

We calculate Spearman's rank correlation coefficient among all the metrics from each performance test in one environment to study the relationship among these metrics. Then we study whether such correlation coefficient is different between the virtual and physical environment.

First, we compare the changes in correlation between the performance metrics and the system throughput. For example, in one environment, the system throughput may be highly correlated with CPU; while in another environment, such correlation is low. In such a case, we would consider that there exist discrepancy in the correlation coefficient between CPU and the system throughput. Second, for every two metrics,

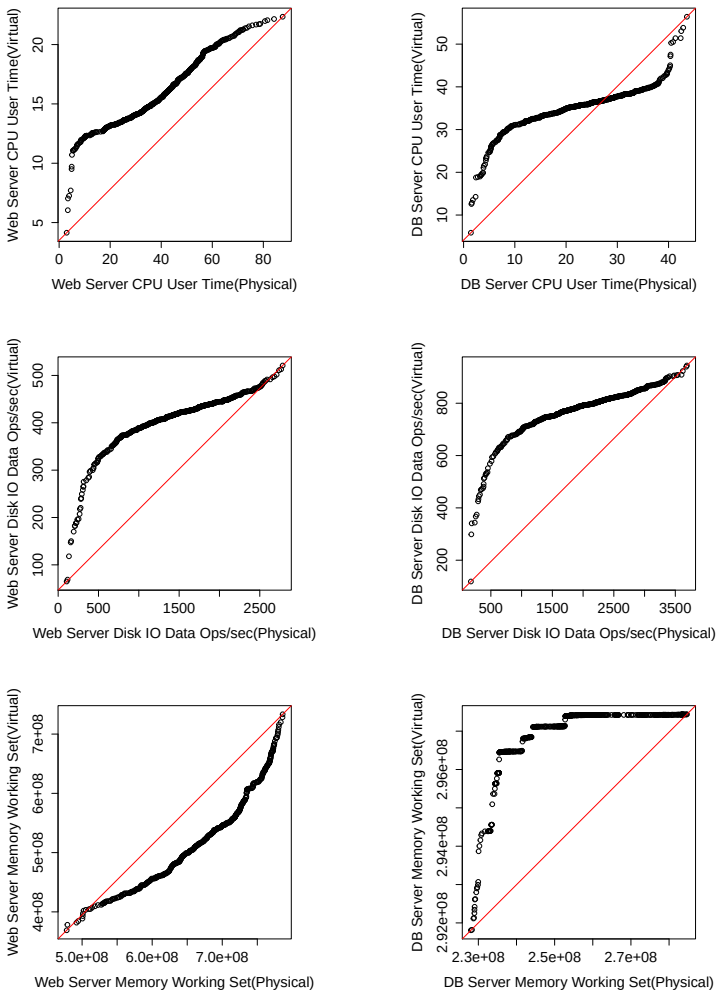

Fig. 3: Q-Q plots for CloudStore.

we calculate the absolute difference between the correlation in two environments. For example, if CPU and Memory have a correlation of 0.3 in the virtual environment and 0.5 in the physical environment, we report the absolute difference in correlation as 0.2 ($|0.3-0.5|$). Since we have 44 metrics in total, we plot a heatmap in order to visualize the 1,936 absolute difference values between every pair of performance metrics. The lighter the color for each block in the heatmap, the larger the absolute difference in correlation a pair of performance metrics have. With the heatmap, we can quickly spot the metrics that have large discrepancy in correlation coefficient.

Results.

Table 1: Spearman's rank correlation coefficients and p-values of the highlighted performance metrics.

Performance Metrics	DS2		CloudStore	
	coef.	p-value	coef.	p-value
Web Servers' User Times	0.08	0.07	-0.04	0.33
DB Servers User Times	-0.05	0.30	0.10	0.02
Web Servers' IO Data Ops/sec	0.25	0.00	0.13	0.00
DB Servers' IO Data Ops/sec	-0.14	0.00	0.13	0.00
Web Servers' Memory Working Set	0.22	0.00	0.69	0.00
DB Servers' Memory Working Set	0.46	0.00	-0.16	0.00

Table 2: Summary of spearman's rank correlation p-values and absolute coefficients of all the performance metrics in DS2 and CloudStore. The numbers in the table are the number of metrics that fall into each category.

System	p-value>0.05	p-value<0.05			
		0.0∼0.3	0.3∼0.5	0.5∼0.7	0.7∼1
DS2	8	28	4	0	1
CloudStore	5	25	4	4	3

Three metrics are constant. Therefore, we do no calculate the correlation on those metrics.

The correlation between system throughput and performance metrics changes between virtual and physical environments. Table 3 presents the top ten metrics with the highest correlation to system throughput in the physical environment for DS2. We find that for these ten metrics, the difference in correlation coefficients in virtual and physical environments is up to 0.78 and the rank changes from #9 to #40.
There exist differences in correlation among the performance metrics from virtual and physical environments. Figure 4 present the heatmap showing the changes in correlation coefficient among the performance metrics from virtual and physical environments. By looking at the heatmap, we can find hotspots (with lighter color), which have larger correlation differences. Due to the limited space, we do not show all the metric names in our heatmaps. Instead, we enlarge the heatmap by showing one of the hotspots for each subject system in Figure 4 and Figure 5. We find that the hotspots correspond to the changes in correlation among I/O related metrics. Prior research on virtual machines has similar findings about I/O overheads in virtual machines [30, 36].

The correlations between performance metrics and system load may change considerably between virtual and physical environments. The correlation among performance metrics may also change considerably between virtual and physical environments. The correlations that are related with the I/O metrics have the largest discrepancy.

4.3 Building statistical models using performance metrics

Approach.

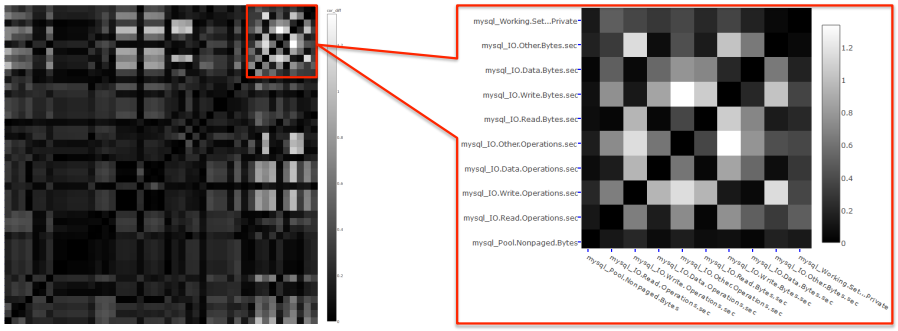

Fig. 4: Heatmap of correlation changes for DS2.

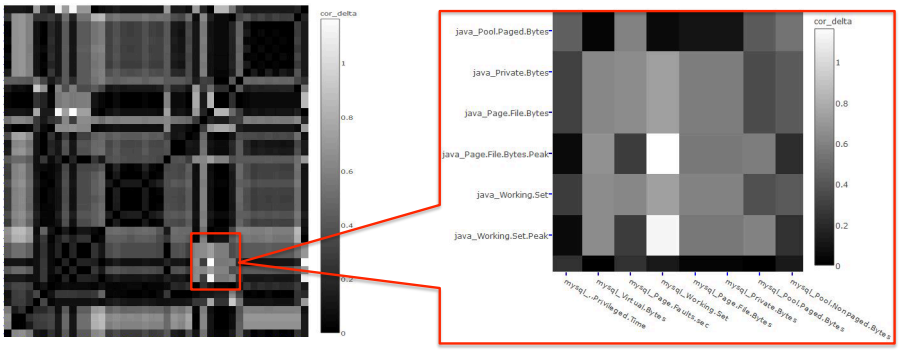

Fig. 5: Heatmap of correlation changes for CloudStore.

We follow a model building approach that is similar to the approach from prior research [16,42,50]. We first build statistical models using performance metrics from one environment, then we validate our performance model with the metric values from the same environment and also from a different environment.

4.3.1 B-1: Reducing counters

We first remove performance counters that have constant values in the test results, since such metrics cannot be included in a statistical model. We then perform a correlation analysis on the performance metrics. We used the Spearman's rank correlation coefficient among all performance metrics from one environment. We find the two performance metrics that have a higher than 0.75 correlation. From these two performance metrics, we remove the metric that has a higher average correlation with all other metrics. We repeat this step until there is no correlation higher than 0.75.

We then perform redundancy analysis [22] on the performance metrics. The redundancy analysis would consider a performance metric redundant if it can be predicted from a combination of other metrics. We use each performance metric as a dependent variable and use the rest of the metrics as independent variables to build

Table 3: Top ten metrics with highest correlation coefficient to system throughput in the physical environment for DS2.

Rank	Performance Metrics	Coef. PE	Coef. VE	Rank in VE
1	Web IO Other Ops/sec	0.91	0.62	10
2	Web IO Other Bytes/sec	0.91	0.62	12
3	Web IO Write Ops/sec	0.91	0.63	9
4	Web IO Data Ops/sec	0.91	0.63	8
5	Web IO Write Bytes/sec	0.90	0.62	11
6	Web IO Data Bytes/sec	0.90	0.61	13
7	DB IO Other Ops/sec	0.84	0.75	3
8	DB IO Data Ops/sec	0.83	0.07	41
9	DB IO Other Bytes/sec	0.83	0.15	40
10	DB IO Read Ops/sec	0.82	0.15	39

PE in the table is short for physical environment; while VE is short for virtual environment.

Table 4: CloudStore: Top 10 highly correlated metrics with load (Physical Server)

Rank	Performance Metrics	Cor in Physical Environment	Cor in Virtual Environment	Rank in Virtual Environment
1	DB Server IO Other Bytes/sec	0.98	0.73	10
2	DB Server IO Read Ops/sec	0.98	0.84	7
3	DB Server IO Read Bytes/sec	0.98	0.93	5
4	DB Server IO Write Ops/sec	0.98	0.97	2
5	DB Server IO Data Ops/sec	0.98	0.92	6
6	DB Server IO Data Bytes/sec	0.98	0.96	4
7	DB Server IO Write Bytes/sec	0.98	0.96	3
8	Web Server IO Other Bytes/sec	0.98	0.68	16
9	DB Server IO Other Ops/sec	0.98	0.98	1
10	Web Server IO Other Ops/sec	0.98	0.70	14

Table 5: CloudStore: Top 10 highly correlated metrics with load (Virtual Server)

Rank	Performance Metrics	Cor in Virtual Environment	Cor in Physical Environment	Rank in Physical Environment
1	DB Server IO Other Ops/sec	0.98	0.98	9
2	DB Server IO Write Ops/sec	0.97	0.98	4
3	DB Server IO Write Bytes/Sec	0.96	0.98	7
4	DB Server IO Data Bytes/sec	0.96	0.98	6
5	DB Server IO Read Bytes/sec	0.93	0.98	3
6	DB Server IO Data Ops/sec	0.92	0.98	5
7	DB Server IO Read Ops/sec	0.83	0.98	2
8	DB Server User Time	0.78	0.89	22
9	DB Server Processor Time	0.76	0.91	21
10	DB Server IO Other Bytes/sec	0.72	0.97	1

a regression model. We calculate the R^2 of each model and if the R^2 is larger than a threshold (0.9), the current dependent variable (i.e., performance metric) is considered redundant. We then remove the performance metric with the highest R^2 and repeat the process until no performance metric can be predicted with R^2 higher than the threshold. For example, if CPU can be linearly modeled by the rest of the performance metrics with $R^2 > 0.9$, we remove the metric for CPU.

4.3.2 B-2: Building statistical models

In the second step, we build a regression model [20] using the performance metrics that are left after the reduction in the first step as independent variables and use the system throughput as dependent variable. Similar models have been built in prior research [16, 50].

4.3.3 B-3: Identifying insignificant metrics

Not all the metrics in the model are statistically significant. Therefore in this step, we only keep the metrics that have a statistically significant contribution to the model. We leverage the *stepwise* function that adds the independent variables one by one to the model to exclude any metrics that are not contributing to the model [28].

4.3.4 B-4: Finalizing statistical models

After removing all the insignificant metrics, we have all the metrics that significantly contribute to the model. We use these metrics as independent variables to build the final model.

4.3.5 V-1: Validating model fit

Before we validate the model with internal and external data, we first examine how good the model fit is. If the models have a poor fit to the data, then our findings from the model may be biased by the noise from the poor model quality. We calculate the R^2 of each model to measure fit. If the model perfectly fits the data, the R^2 of the model is 1, while a zero R^2 value indicates that the model is completely random. We also would like to estimate the impact that each independent variable has on the model fit. We follow a "drop one" approach [11], which measures the impact of an independent variable on a model by measuring the difference in the performance of models built using: (1) all independent variables (the full model), and (2) all independent variables except for the one under test (the dropped model). A Wald statistic [21] is reported by comparing the performance of these two models. A larger Wald statistic indicates that an independent variable has a larger impact on the model's performance, i.e., model fit. A similar approach has been leveraged by prior research in [35]. We then rank the independent variables by their impact on model fit.

4.3.6 V-2: Internal validation

We validate our models with the performance testing data that is from the same environment. We leverage a standard 10-fold cross validation process, which starts by partitioning the performance data to 10 partitions. We take one partition (fold) at a time as the test set, and trains on the remaining nine partitions [29,41], similar to prior research [33]. For every data point in the testing data, we calculate the absolute percentage error. For example, for a data point with a throughput value of 100 requests per minute, if our predicted value is 110 requests per minute, the absolute percentage

error is 0.1 ($\frac{|110-100|}{100}$). After the ten-fold cross validation, we have a distribution of absolute percentage error for all the data records.

4.3.7 V-3: External validation

To evaluate whether the model built using performance testing data in one environment (e.g., virtual environment) can apply to another environment (e.g., physical environment), we test the model using the data from a different environment. For example, when we build a model using the data from a virtual environment, we test the model using the data from a physical environment.

Since the performance testing data is generated from different environments, directly applying the data on the model would intuitively generate large amounts of error. We adopt two approaches in order to normalize the data in different environments: (1) **Normalization by deviance.** The first approach we use is the same when we compare the distribution of each individual performance counter shown in Equation 1 from Section 4.1 by calculating the relative deviance of a metric value from its median value. (2) **Normalization by load.** The second approach that we adopt is an approach that is proposed by Nguyen *et al.* [38]. The approach uses the load of the system to normalize the performance metric values across different environments.

To normalize our metrics, we first build a linear regression model with the one metric as independent variable and the throughput of the system as the dependent variable. With the linear regression model in one environment, the metric values can be represented by the system throughput. Then we normalize the metric value by the linear regression from the other environment. The details of the metric transformation are shown as follows:

$$throughput_p = \alpha_p \times M_p + \beta_p$$

$$throughput_v = \alpha_v \times M_v + \beta_v$$

$$M_{normalized} = \frac{(\alpha_v \times M_v) + \beta_v - \beta_p}{\alpha_p}$$

where $throughput_p$ and $throughput_v$ are the system throughput in the physical and virtual environment, respectively. M_p and M_v are the performance metrics from both environments, while $M_{normalized}$ is the metric after normalization. α and β are the coefficient and intercept values for the linear regression models. After normalization, we calculate the absolute percentage error for every data record in the testing data.

4.3.8 Identifying model discrepancy

In order to identify the discrepancy between the models build using data from the virtual and physical environments, we compare the two distributions of absolute percentage error based on our internal and external validation. If the two distributions are significantly different (e.g., the absolute percentage error from internal validation is much lower than that from external validation), the two models are considered to have a discrepancy. To be more concrete, in total for each subject system, we ended

up with four distributions of absolute percentage error: 1) modelling using the virtual environment and testing internally (on data from the virtual environment), 2) modelling using the virtual environment and testing externally (on data from the physical environment), 3) modelling using the physical environment and testing internally (on data from the physical environment), 2) modelling using the physical environment and testing externally (on data from the virtual environment). We compare distributions 1) and 2) and we compare distributions 3) and 4). Since normalization based on deviance will change the metrics values to be negative when the metric value is lower than median, such negative values cannot be used to calculate absolute percentage error. We perform a min-max normalization on the metric values before calculating the absolute percentage error. In addition, if the observed throughput value after normalization is zero (when the observed throughput value is the minimum value of both the observed and predicted throughput values), we cannot calculate the absolute percentage error for that particular data record. Therefore, we remove the data record if the throughput value after normalization is zero. In our case study, we only removed one data record when performing external validation with the model built in the physical environment.

Results.

The statistically significant independent variables in the models built by performance testing results in virtual and physical environments are different. Tables 6 and 7 show the summary of the statistical models built for the virtual and physical environments for the two subject systems. We find that all the models have good model fit (66.9% to 94.6%). However, some statistically significant independent variables in one model do not appear in the other model. For example, Web Server Virtual Bytes ranks #4 for the model built from physical environment data of CloudStore, while the metric is not significant in the model built from the virtual environment data. In fact, none of the significant variables in the model built from virtual environment are related to the web server's memory (see Table 7). We do observe some performance metrics that are significant in both models even with the same ranking. For example, Web Server IO Other Bytes/sec is the #1 significant metric for both models built from the virtual and physical environment data of DS2 (see Table 6).

The prediction error illustrates discrepancies between models built by performance testing results in the virtual and physical environments. Although the significant independent variables in the models built by the performance testing results in the virtual and physical environments are different, the model may have similar prediction results due to correlations between metrics. However, we find that the external prediction errors are higher than internal prediction errors for all four models from the virtual and physical environments for the two subject systems. In particular, Table 8 shows the prediction errors using normalization based on load is always higher than that of the internal validation. For example, the median absolute percentage error for CloudStore using normalization by load is 632% and 483% for the models built in the physical environment and virtual environment, respectively; while the median absolute percentage error in interval validation is only 2% and 10% for for the models built in physical environment and virtual environment, respectively. However, in some cases, the normalization by deviance can produce low absolute

percentage error in external validation. For example, the median absolute percentage error for CloudStore can be reduced to 9% using normalization by deviance.

We think the reason is that the normalization based on load performs better, even though is shown to be effective in prior research [38], assumes a linear relationship between the performance metric and the system load. However, such an assumption may not be true in some performance testing results. For example, Table 3 shows that some I/O related metrics do have low correlation with the system load in virtual environments. On the other hand, the normalization based on deviance shows much lower prediction error. We think the reason is that the virtual environments may introduce metric values with high variance. Normalizing based on the deviance controls such variance, leading to lower prediction errors.

Table 6: Summary of statistical models built for DS2. The metrics listed in the table are the significant independent variables.

Environment	**Physical**	**Virtual**
Ranking	1. Web Server IO Other Bytes/sec	1. Web Server IO Other Bytes/sec
	2. Web Server Page Faults/sec	2. DB server Working Set - Peak
	3. DB Server Page Faults/sec	3. Web Server Virtual Bytes
	4. DB Server IO Write Bytes/sec	4. Web Server Page Faults/sec
	5. Web Server IO Read Bytes/sec	5. DB Server Page Faults/sec
	6. DB Server User Time	6. DB Server IO Data Ops/sec
	7. DB Server Pool Paged Bytes	
	8. DB Server Privileged Time	
R^2	94.6%	66.90%

Table 7: Summary of statistical models built for CloudStore. The metrics listed in the table are the significant independent variables.

Environment	**Physical**	**Virtual**
Ranking	1. Web Server Privileged Time	1. Web Server IO Write Ops/sec
	2. DB Server Privileged Time	2. DB Server IO Read Ops/sec
	3. Web Server Page Faults/sec	3. Web Server Privileged Time
	4. Web Server Virtual Bytes	4. DB Server Privileged Time
	5. Web Server Page File Bytes Peak	5. DB Server IO Other Bytes/sec
	6. DB Server Pool Nonpaged Bytes	6. DB Server Pool Nonpaged Bytes
	7. DB Server Page Faults/sec	
	8. DB Server Working Set	
R^2	85.30%	90.20%

Table 8: Internal and external prediction errors for both subject systems.

DS2								
Model Built	**Validation**		**Min.**	**1st Quart.**	**Median**	**Mean**	**3rd Quart.**	**Max**
Physical	**Internal Validation**		0.00	0.01	0.02	0.03	0.05	0.30
	External Validation	**Normalization by Deviance**	0.00	0.08	0.25	0.36	0.49	13.65
		Normalization by Load	0.00	0.34	0.44	0.48	0.56	1.56
Virtual	**Internal Validation**		0.00	0.04	0.09	0.11	0.15	0.54
	External Validation	**Normalization by Deviance**	0.00	0.09	0.20	0.27	0.34	2.82
		Normalization by Load	0.00	0.06	0.13	0.17	0.23	0.92

CloudStore								
Model Built	**Validation**		**Min.**	**1st Quart.**	**Median**	**Mean**	**3rd Quart.**	**Max**
Physical	**Internal Validation**		0.00	0..05	0.10	0.16	0.18	2.68
	External Validation	**Normalization by Deviance**	0.00	0.04	0.09	0.17	0.17	2.29
		Normalization by Load	2.90	5.14	6.32	7.75	8.08	51.33
Virtual	**Internal Validation**		0.00	0.01	0.03	0.04	0.05	0.50
	External Validation	**Normalization by Deviance**	0.00	0.03	0.07	0.11	0.13	1.00
		Normalization by Load	4.07	4.64	4.83	5.13	5.10	33.36

The statistical models built by performance testing results in virtual and physical environments have discrepancy. The external prediction errors are higher than internal prediction errors, showing that one cannot apply the models from one environment on data from another environment. Normalizing the performance metrics by deviance may minimize such discrepancy and reduce the external prediction error. Practitioners may consider normalizing the performance testing result data from virtual environment by deviance before analysis.

5 Discussion

In the previous section, we find that there is a discrepancy between performance testing results from the virtual and physical environments. However, such discrepancy can also be due to other factors such 1) the instability of the virtual environments, 2) the virtual machine that we used or 3) the different hardware resources on the virtual environments. Therefore, in this section, we examine the impact of such factors to better understand our results.

5.1 Instability of virtual environment

If the performance of virtual machines are unstable, the observed discrepancy in Section 4 may due to the instability of virtual environment, i.e., the performance testing results are not repeatable from virtual environments. In order to study whether virtual environment is unstable, leading to discrepancy in performance testing results, we repeat the same performance tests on the virtual environments for both subject systems. We perform the data analysis in Section 4.3 by building statistical models using performance metrics. We find that external validation error (0.04 and 0.13 for CloudStore and DS2) is almost as low as the internal validation error (0.03 and 0.09 for CloudStore and DS2). Such low error shows that the performance testing results from the virtual environments are rather stable.

5.2 Virtual machine software for the virtual environment

We also investigated the impact of choosing different virtual machine software on our experimental results. We set up another virtual environment using VMWare (version 12) with the same allocated computing resources as when we set up Virtual Box. We repeat the performance tests for both subject systems. We train statistical models on the performance testing results from VMWare and test on the results from both the original virtual environment data (Virtual Box) and the results from the physical environments. We could not apply the normalization by deviance for the data from VMWare since some of the significant metrics in the model have a median absolute deviance of 0, making the normalized metric value to be infinite (see Equation 1). We only apply the normalization by load. The low error on Virtual Box in Table 9 shows that the performance testing results from the two different virtual machine software is similar. In addition, the high error when predicting with physical environment agrees with the results when testing with the performance testing results from the Virtual Box (see Table 8). Such results show that the discrepancy observed during our experiment also exits with the virtual environments that are set up with VMWare.

Table 9: Median absolute percentage error from building a model using VMWare data.

Validation type	**Median absolute percentage error**	
	CloudStore	**DS2**
External validation with Virtual Box results	0.07	0.10
External validation with physical normalization by load	7.52	1.63

5.3 Resource Allocation

We study the impact of changing the allocated resources on the results of our analysis. We only increase the allocated resources in order to ensure the execution of the subject systems. We increase the computing resources allocated to the virtual environments by increasing the CPU to be 3 cores and increasing the memory to be 5GB. We cannot allocate more resource to the virtual environment since we need to keep resources for the hosting OS. Similarly, we train statistical models on the new performance testing results and tested it on the performance testing results from the physical environment. Similar to the results shown in Table 8, the prediction error is high when normalizing based on load (1.57 for DS2 and 1.25 for CloudStore), while normalizing based on deviance can significantly reduce the error (0.09 for DS2 and 0.07 for CloudStore). Such results show the minimal impact to our findings by changing resource allocation. Moreover, our results demonstrate the ability of reducing discrepancy in performance testing results by using normalization based on deviance.

6 Threats to Validity

This section discusses the threats to our validity.

External validity. We chose two subject systems, CloudStore and DS2 for our study and two virtual machine software, VirtualBox and VMware. The two subject systems have years of history and prior performance engineering research have studied both systems [6, 27, 38]. The virtual machine software that we used are widely used in practice. Nevertheless more case studies on other subject systems in other domains with other virtual machine software are needed to evaluate our findings. We also present our results based on our subject systems only and do not generalize for all the virtual machines.

Internal Validity. Our approach is based on the recorded performance metrics. The quality of recorded performance metrics can impact the internal validity of our study. Replicating our study by other performance monitoring tools, such as psutil [4] may address this threat. Even though we build a statistical model using performance metrics and system throughput, we do not assume that there is causal relationship. The use of statistical models merely aims to capture the relationship among multiple metrics. Similar approaches have been used in the prior studies [16, 42, 50].

Construct Validity. We monitor the performance by recording performance metrics every 10 seconds and combine the performance metrics for every minute together as an average value. There may exist unfinished system requests when we record the system performance, leading to noise in our data. We choose a time interval (10 seconds) that is much higher than the response time of the requests (less than 0.1 second), in order to minimize the noise. Repeating our study by choosing other time interval sizes would address this threat. In addition, repeating our study by using other aggregation values (like median) to combine performance metrics values may further verify our findings. We exploit two approaches to normalize performance data from different environments. We also see that our R^2 value is high. Although a higher R^2 determines our model is accurate but it may also be an indication of overfit. There may exist other advance approaches to normalize performance data from heterogeneous environment. We plan to extend our study on other possible normalization approaches. There may exist other ways of examining performance testing results. We plan to extend our study by evaluating the discrepancy of using other ways of examining performance testing results in virtual and physical environments.

7 Conclusion

Performance assurance activities are vital in ensuring software reliability. Virtual environments are often used to conduct performance tests. However, the discrepancy between performance testing results in virtual and physical environments are never evaluated. We aimed to highlight that whether a discrepancy present between physical and virtual environments will impact the studies and tests carried out in the software domain. In this paper, we evaluate such discrepancy by conducting performance tests on two open source systems (DS2 and CloudStore) in both virtual and physical environments. By examining the performance testing results, we find that there

exists a discrepancy between performance testing results in virtual and physical environments when examining individual performance metrics, the relationship among performance metrics and building statistical models from performance metrics, even after we normalize performance metrics across different environments. The major contribution of this paper includes:

- Our paper is the first research attempt to evaluate the discrepancy between performance testing results in virtual and physical environments.
- We find that relationships among I/O related metrics have large differences between virtual and physical environments.
- We find that normalizing performance metrics based on deviance may reduce the discrepancy. Practitioners may exploit such normalization techniques when analyzing performance testing results from virtual environments.

Our results highlight the need of the awareness of discrepancy between performance testing results in virtual and physical environments, for both practitioners and researchers. Future research effort may focus on minimizing such discrepancy in order to improve the use of virtual environments in performance engineering and reliability assurance activities.

References

1. Accelerate software development and testing with the vmware virtualization platform. URL http://www.vmware.com/pdf/development_testing.pdf
2. The avoidable cost of downtime. URL http://www3.ca.com/~/media/files/articles/avoidable_cost_of_downtime_part_2_ita.aspx
3. How one second could cost amazon $1.6 billion in sales. http://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales
4. Psutil. https://github.com/giampaolo/psutil. Accessed: 2015-06-01
5. Book review: The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling by raj jain (john wiley & sons 1991). SIGMETRICS Perform. Eval. Rev. **19**(2), 5–11 (1991). Reviewer-Al-Jaar, Robert Y.
6. Ahmed, T.M., Bezemer, C.P., Chen, T.H., Hassan, A.E., Shang, W.: Studying the effectiveness of application performance management (apm) tools for detecting performance regressions for web applications: An experience report. In: MSR 2016: Proceedings of the 13th Working Conference on Mining Software Repositories (2016)
7. Apache: Jmeter. http://jmeter.apache.org/. Accessed: 2015-06-01
8. Apache: Tomcat. http://tomcat.apache.org/. Accessed: 2015-06-01
9. Bodík, P., Goldszmidt, M., Fox, A.: Hilighter: Automatically building robust signatures of performance behavior for small-and large-scale systems. In: SysML (2008)
10. Brosig, F., Gorsler, F., Huber, N., Kounev, S.: Evaluating approaches for performance prediction in virtualized environments. In: 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 404–408. IEEE (2013)
11. Chambers, J., Hastie, T., Pregibon, D.: Compstat: Proceedings in Computational Statistics, 9th Symposium held at Dubrovnik, Yugoslavia, 1990, chap. Statistical Models in S, pp. 317–321. Physica-Verlag HD, Heidelberg (1990)
12. Chen, P.M., Noble, B.D.: When virtual is better than real [operating system relocation to virtual machines]. In: Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on, pp. 133–138 (2001)
13. Chen, P.M., Noble, B.D.: When virtual is better than real [operating system relocation to virtual machines]. In: Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on, pp. 133–138 (2001)

14. CloudScale-Project: Cloudstore. https://github.com/CloudScale-Project/CloudStore. Accessed: 2015-06-01
15. Cohen, I., Chase, J.S., Goldszmidt, M., Kelly, T., Symons, J.: Correlating instrumentation data to system states: A building block for automated diagnosis and control. In: OSDI, vol. 4, pp. 16–16 (2004)
16. Cohen, I., Zhang, S., Goldszmidt, M., Symons, J., Kelly, T., Fox, A.: Capturing, indexing, clustering, and retrieving system history. In: Proceedings of the Twentieth ACM Symposium on Operating Systems Principles, SOSP '05, pp. 105–118 (2005)
17. Dave Jaffe, T.M.: Dell dvd store. http://linux.dell.com/dvdstore/. Accessed: 2015-06-01
18. Dean, J., Barroso, L.A.: The tail at scale. Communications of the ACM **56**, 74–80 (2013)
19. Foo, K.C., Jiang, Z.M., Adams, B., Hassan, A.E., Zou, Y., Flora, P.: Mining performance regression testing repositories for automated performance analysis. In: Quality Software (QSIC), 2010 10th International Conference on, pp. 32–41. IEEE (2010)
20. Freedman, D.: Statistical models: theory and practice. Cambridge University Press (2009)
21. Harrell, F.: Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis. Springer (2015)
22. Harrell, F.E.: Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis. Springer (2001)
23. Heger, C., Happe, J., Farahbod, R.: Automated root cause isolation of performance regressions during software development. In: ICPE '13: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, pp. 27–38 (2013)
24. Huber, N., von Quast, M., Hauck, M., Kounev, S.: Evaluating and modeling virtualization performance overhead for cloud environments. In: CLOSER, pp. 563–573 (2011)
25. Jiang, M., Munawar, M., Reidemeister, T., Ward, P.: Automatic fault detection and diagnosis in complex software systems by information-theoretic monitoring. In: DSN '09: Proceedings of 2009 IEEE/IFIP International Conference on Dependable Systems Networks, pp. 285–294 (2009)
26. Jiang, M., Munawar, M.A., Reidemeister, T., Ward, P.A.: System monitoring with metric-correlation models: Problems and solutions. In: ICAC '09: Proceedings of the 6th International Conference on Autonomic Computing, pp. 13–22 (2009)
27. Jiang, Z.M., Hassan, A.E., Hamann, G., Flora, P.: Automated performance analysis of load tests. In: Software Maintenance, 2009. ICSM 2009. IEEE International Conference on, pp. 125–134 (2009)
28. Kabacoff, R.I.: R In Action. In: R In Action, pp. 207–213. Manning Publications Co., Staten Island, NY (2011)
29. Kohavi, R., et al.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai, vol. 14, pp. 1137–1145 (1995)
30. Kraft, S., Casale, G., Krishnamurthy, D., Greer, D., Kilpatrick, P.: Io performance prediction in consolidated virtualized environments. In: ACM SIGSOFT Software Engineering Notes, vol. 36, pp. 295–306. ACM (2011)
31. Luo, Q., Poshyvanyk, D., Grechanik, M.: Mining performance regression inducing code changes in evolving software. In: Proceedings of the 13th International Conference on Mining Software Repositories, MSR '16, pp. 25–36. ACM, New York, NY, USA (2016). URL http://doi.acm.org/10.1145/2901739.2901765
32. Malik, H., Adams, B., Hassan, A.E.: Pinpointing the subsystems responsible for the performance deviations in a load test. In: 2010 IEEE 21st International Symposium on Software Reliability Engineering, pp. 201–210 (2010)
33. Malik, H., Hemmati, H., Hassan, A.E.: Automatic detection of performance deviations in the load testing of large scale systems. In: 2013 35th International Conference on Software Engineering (ICSE), pp. 1012–1021 (2013)
34. Malik, H., Jiang, Z.M., Adams, B., Hassan, A.E., Flora, P., Hamann, G.: Automatic comparison of load tests to support the performance analysis of large enterprise systems. In: CSMR '10: Proceedings of the 2010 14th European Conference on Software Maintenance and Reengineering, pp. 222–231 (2010)
35. McIntosh, S., Kamei, Y., Adams, B., Hassan, A.E.: An Empirical Study of the Impact of Modern Code Review Practices on Software Quality. Empirical Software Engineering p. To appear (2015)
36. Menon, A., Santos, J.R., Turner, Y., Janakiraman, G.J., Zwaenepoel, W.: Diagnosing performance overheads in the xen virtual machine environment. In: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, pp. 13–23. ACM (2005)
37. Microsoft Technet: Windows performance counters. https://technet.microsoft.com/en-us/library/cc780836(v=ws.10).aspx. Accessed: 2015-06-01

38. Nguyen, T.H., Adams, B., Jiang, Z.M., Hassan, A.E., Nasser, M., Flora, P.: Automated detection of performance regressions using statistical process control techniques. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE '12, pp. 299–310. ACM, New York, NY, USA (2012)
39. NIST/SEMATECH: e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm. Accessed: 2015-06-01
40. Oracle: MYSQL server 5.6. https://www.mysql.com/. Accessed: 2015-06-01
41. Refaeilzadeh, P., Tang, L., Liu, H.: Encyclopedia of Database Systems, chap. Cross-Validation, pp. 532–538. Springer US, Boston, MA (2009)
42. Shang, W., Hassan, A.E., Nasser, M., Flora, P.: Automated detection of performance regressions using regression models on clustered performance counters. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, ICPE '15, pp. 15–26. ACM, New York, NY, USA (2015)
43. Shewhart, W.A.: Economic control of quality of manufactured product, vol. 509. ASQ Quality Press (1931)
44. Stapleton, J.H.: Models for Probability and Statistical Inference: Theory and Applications. WILEY (2008)
45. Syer, M.D., Jiang, Z.M., Nagappan, M., Hassan, A.E., Nasser, M., Flora, P.: Leveraging performance counters and execution logs to diagnose memory-related performance issues. In: Software Maintenance (ICSM), 2013 29th IEEE International Conference on, pp. 110–119 (2013)
46. Syer, M.D., Shang, W., Jiang, Z.M., Hassan, A.E.: Continuous validation of performance test workloads. Automated Software Engineering pp. 1–43 (2016)
47. TPC: TPC-W. www.tpc.org/tpcw. Accessed: 2015-06-01
48. Walker, H.M.: Studies in the history of statistical method: With special reference to certain educational problems. (1929)
49. Woodside, M., Franks, G., Petriu, D.C.: The future of software performance engineering. In: Future of Software Engineering, 2007. FOSE '07, pp. 171–187 (2007)
50. Xiong, P., Pu, C., Zhu, X., Griffith, R.: vperfguard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, ICPE '13, pp. 271–282. ACM, New York, NY, USA (2013)