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 Large software systems often undergo performance tests to ensure their capability to handle expected loads. These performance tests often consume large amounts of computing resources and time since heavy loads need to be generated. Making it worse, the ever evolving field requires frequent updates to 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 overhead (e.g., a higher than expected memory utilization) to the testing environment and lead to unrealistic performance testing results. Yet, little or no research has studied the overhead and impact on test results of using VMs in performance testing activities.

To evaluate the discrepancy between the performance testing results from virtual and physical environments, we perform a case study on two open source systems - namely Dell DVD Store and CloudStore. We conduct the same performance tests in both virtual and physical environments and compare the performance testing results based on the three aspects that are typically examined for performance testing results: 1) individual performance metrics (e.g. CPU Time from virtual environment vs. CPU Time from physical environment), 2) the relationship among performance metrics (e.g. correlation between CPU and I/O) and 3) performance models that are built to predict system performance. Our results show that 1) individual performance metrics from virtual and physical environments do not follow the same distribution hence practitioners cannot simply use a scaling factor to compare the performance between environments, 2) correlations among performance metrics in virtual environments are different from those in physical environments 3) statistical models built based on the performance counters from virtual environments are different from the models built from physical environments suggesting that practitioners cannot use the

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

performance testing results across virtual and physical environments. In order to assist the practitioners leverage performance testing results in both environments, we also investigate ways to transform results from virtual and physical environments. Performance metrics based on deviance may reduce the discrepancy between performance metrics. Overall, we recommend that practitioners should not simply assume that performance testing results done on virtual environments will be the same in physical environments.

1 Introduction

Software performance assurance activities play a vital role in the development of large software systems. These activities ensure that the software meets the desired performance requirements [65]. Often however, failures in large software systems are due to performance issues rather than functional bugs [15, 18]. Such failures lead to the eventual decline in quality of the system with reputational and monetary consequences [7]. For instance, Amazon estimates that a one-second page-load slowdown can cost up to $1.6 billion [17].

In order to mitigate performance issues and ensure software reliability, practitioners often conduct performance tests [65]. Performance tests apply a workload (e.g., mimicking users' behavior in the field) on the software system [23, 57], and monitor performance metrics, such as CPU usage, that are generated based on the tests. Practitioners use such metrics to gauge the performance of the software system and identify potential performance issues (such as memory leaks [56] and throughput bottlenecks [35]).

Since performance tests are often performed on large-scale software systems, the performance tests often require many resources [23]. Moreover, performance tests often need to run for a long period of time in order to build statistical confidence on the results [23]. In addition, such testing environment needs to be easily configurable such that a specific environment can be mimicked, reducing false performance issues. For example, issues that are related to the environment. Therefore, to address such challenges, virtual environments (i.e., VMs) are often leveraged for performance testing [9,61]. Additionally, based on our extensive collaboration with industrial practitioners, we find that a large amount of the performance testing is conducted in virtual environments as supported by the online discussions and posts by developers and testers [16, 29, 58]. In addition, the developers of SugarCRM are documenting their experiences in conducting load testing (which has a big overlap with performance testing) on virtual environments [41]. Some of the software systems are released in both on-premise(physical) and cloud (virtual) environment. For such systems consistency across solutions is key for example, comparing the new tests with older tests for a version or having the knowledge of the discrepancies present between the two environments [51, 64]. Hence, due to the flexibility of virtual environments enables practitioners to easily prepare, customize, use and update performance testing environments in an efficient manner.

Prior studies show that virtual environments are widely exploited in practice [10, 44, 66]. Studies have investigated the overhead that is associated with virtual envi-

ronments [40]. Such overheads may not impose effect on the results of performance tests carried out in physical and virtual environments. For example, if the performance (e.g., throughput) of the system follows the same trend (or distribution) in physical and virtual environments, such overhead would not significantly impact the outcome for the practitioners who examine the performance testing results. Our work is one of the first works that examine the discrepancy between performance testing results in virtual and physical environments. Exploring, identifying and minimizing such discrepancy will help practitioners and researchers understand and leverage performance testing results from virtual and physical environments. Without knowing if there exists a discrepancy between the performance testing results from the two environments practitioners cannot rely on the performance assurance activities carried out in the virtual environment or vice versa. Once the discrepancy is identified, the performance results could be evaluated more accurately.

In this paper, we perform a study on two open-source systems, DS2 [14] and CloudStore [11], where performance tests are conducted using virtual and physical environments. Our study focuses on the discrepancy between the two environments, the impact of discrepancy on performance testing results and highlights potential opportunities to minimize the discrepancy. In particular, 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.

We find that 1) performance metrics have different shapes of distributions and trends 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. The overhead from virtual environments does not only impact the scale of the performance metrics, but also impacts the relationship among performance metrics. On the other hand, we find that practitioners who leverage both, virtual and physical environments, may be able to reduce the discrepancy that may arise due to the environment (i.e., virtual vs. physical) by applying normalization techniques.

The rest of the paper is organized as follows. Section 2 presents the background and related work. Section 4 presents the case study setup. Section 5 presents the results of our cases study, followed by a discussion of our results in Section 6. Section 7 discusses the threats to validity of our findings. Finally, Section 8 concludes this paper.

2 Background and Related Work

In this section, we discuss the motivation and related work of this paper in broadly two main subsections: 1) analyzing performance metrics from performance testing and 2) analysis of VM overhead.

2.1 Analyzing performance metrics from performance testing

Prior research has proposed a slew of techniques to analyze performance testing results, i.e. performance metrics. Such techniques typically examine three different aspects of the metrics: 1) individual performance metrics, 2) the relationship among performance metrics, and 3) statistical modeling based on performance metrics.

2.1.1 Individual performance metrics

Nguyen *et al*. [44] introduce the concept of using control charts [53] in order to detect performance regressions. 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*. [36, 37] 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*. [21] 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*. [26] 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.

2.1.2 Relationship among performance metrics

Malik *et al*. [35] 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 the performance deviation.

Foo *et al*. [18] propose an approach that leverages association rules in order to address the limitations of manually detecting performance regressions 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. Deviations in the association rules are considered signs of performance regressions.

Jiang *et al*. [24] 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.

2.1.3 Statistical modeling based on performance metrics

Xiong *et al*. [66] 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 assists 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.* [52] 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*. [12] 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*. [13] also proposed that system faults can be detected by building statistical models based on performance metrics. The approaches of Cohen *et al*. [12,13] were improved by Bodik *et al.* [5] by using logistic regression models.

Jiang *et al*. [25] 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. The authors conclude their approach is more efficient than the current linear-model approach.

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. Prior research does not explicitly mention the environment. Other prior research is performed on either virtual or physical environments only. For example, the research conducted by Xiong *et al*. [66] is only conducted in the virtual environment. However, none of them discuss the applicability of their approaches cross-environments.

2.2 Analysis of VM overhead

Kraft *et al*. [31] 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.* emphasize on the latencies existing in virtual machine requests for disc IO due to increments in time associated with request queues.

Aravind *et al*. [40] audit the performance overhead in Xen virtual machines. They uncover the origins of overhead 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 overhead.

Brosig *et al*. [6] predict the performance overhead of virtualized environments using Petri-nets in Xen server. The authors focused on the visualization overhead with respect to queuing networks only. The authors were able to accurately predict server utilization but had significant errors for multiple VMs.

Huber *et al*. [22] present a study on cloud-like environments. The authors compare the performance of virtual environments and study the degradation between the two environments. Huber *et al*. further categorize factors that influence the overhead and use regression based models to evaluate the overhead. However, the modeling only considers CPU and memory.

Luo *et al*. [34] converge the set of inputs that may cause software regression. They apply genetic algorithms to detect such combinations. Netto *et al*. [43] present a similar study to compare performance metrics generated via load tests between the two environments. However, the author did not analyse the results from a statistical perspective.

Prior research focused on the overhead of virtual environments without considering the impact of such overhead on performance testing and assurance activities. In this paper, we evaluate the discrepancy between virtual and physical environments by focusing on the impact of performance testing results and investigate whether such impact can be minimized in practice.

2.3 Performance testing and bug detection

There has been much work on performance testing and bug detection. Nistor *et al*. [46] detect the presence of functional and loop-related performance bugs with the help of their developed tool . Jin *et al*. [27] present a study on wide range of performance bugs. . The authors examine rule-based performance bug detection in performance bug patches . Nistor *et al*. [45] in another study highlight that performance bug detection based on tools is limited . The authors also comment that performance bugs are mostly detected by code reasoning rather than seeing the effects of the system by the end user . Tsakiltsidis *et al*. [60] use prediction models to detect and predict performance bugs based on extraction from source code repositories. Malik *et al*. [38] present a study to uncover functional bugs via load testing. The authors propose an approach to reduce the large amount of performance metrics at the end of a load test by principal component analysis. Zaman *et al*. [67] study the tracking and fixing of performance bugs.

However, none of the above mentioned performance bug detection approach has been applied in different environments. In most of the cases, the environment is not explicitly mentioned. Hence, to generalize the findings across environments remains an open topic.

3 A Motivating Example

Ben is a performance engineer for a large-scale distributed system which serves users globally. The system has the options of running it on cloud or/and on-premise. After every update, it is Ben's duty to ensure that the performance of the system passes the performance tests as it is of critical importance.

In order to carry out a performance test Ben has to exercise the system. Due to the ever-evolving environment and user requirements Ben decides that because of the flexibility offered, he will deploy his system in a virtual environment. After the deployment, Ben applies a workload to exercise the system. The workload is as close to the field like load as possible. During the course of this test, the performance metrics are monitored and collected.

To cover the performance test for an on-premise deployment of the system and to save resources, Ben decides to use a model-based approach. On the basis of his prior experience, he uses the system throughput as the dependent variable and the rest of performance metrics as the independent variables. He then applies the model to predict his throughput of the same version of the system in a physical or his on-premise environment. Ben expects a prediction percentage error less than 10% in order to consider the two systems in harmony.

However, when Ben applies his model to the same version of the system in a physical environment he sees a very high percentage error close to 100%. He examines it further and sees that the magnitude of some of the performance metrics especially I/O performance metrics from the virtual environment is nowhere close to the I/O performance metrics from the physical environment. He concludes that the system's I/O performance in the virtual environment is being hampered.

From this example we note that if Ben would have known the discrepancy present between the results of physical and virtual environment, he would have not been surprised with the high prediction percentage error he observed. He may apply our suggested approach, normalization by deviance (as discussed in section 5.3.5), in order to reduce and and re-analyze the performance tests between the two environments.

4 Case Study Setup

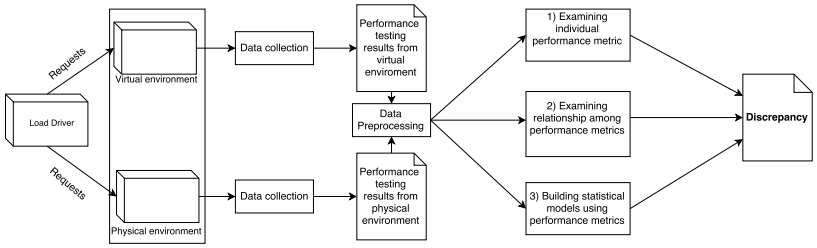

Fig. 1: An overview of our case study setup.

The goal of our case study is to evaluate the discrepancy between performance testing results from virtual and physical environments. We deploy our subject systems in two identical environments(physical and virtual). A load driver is used to exercise our subject systems. After the collection and processing of the performance metrics we analyze and draw conclusions based on: 1) individual performance metrics 2) relationship among performance metrics and 3) statistical models based on the performance metrics. An overview of our case study setup is shown in Figure 1.

4.1 Subject Systems

Dell DVD Store (DS2) [14] is an online multi-tier e-commerce web application that is widely used in performance testing and prior performance engineering research [26, 44, 52]. We deploy DS2 on an Apache (Version 3.0.0) web application server with MySQL 5.6 database server [48]. The file size of DS2 is less than an MB. CloudStore [11], our second subject system, with a size of 1.2 MB, is an open source application based on the TPC-W benchmark [59]. CloudStore is widely used to evaluate the performance of cloud computing infrastructure when hosting web-based software systems and is leveraged in prior research [1]. We deploy CloudStore on *Apache Tomcat* [4] (version 7.0.65) with MySQL 5.6 database server [48].

4.2 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 were deployed on the third machine. We separate the load driver, the web/application server and the database server on different machines in order to mimic real world scenario and avoid interference among these processes. For example, isolating the web and database driver would ensure that the processor is not overused. 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 three gigabytes of memory, which is well below capacity to make sure we are topping out and pushing our configuration for unrealistic results. We used a 100GB SATA storage. As an alternate, there is a possibility of using disk pass through. Since practitioners want to quickly deploy an existing virtual machine image thats designed for performance testing and start performance tests right away we chose not to use disk pass through [54]. We set up our network based on NAT. The network traffic was generated on another machine to keep our experiments as close to the real-world as possible.

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. We used the same hardware to set up our virtual and physical environments.

4.3 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 [3] to generate a workload to conduct the performance tests on CloudStore. For both subject systems, the workload of the performance tests is varied randomly and periodically in order to avoid bias from a consistent workload. The variation was identical across environments. The workload variation was introduced by the number of threads. A higher number of threads represent a higher number of users accessing the system. Each performance test is run after a 15 minute warming up period of the system and lasts for 9 hours. We chose to run the test 9 hours ensuring that our sample sizes have enough data points for our results to be statistically significant [63]. The nature of our performance tests was based on our related studies mentioned in section 2.2. To ensure the consistency between the performance tests, we restored the environments followed by a restart of the systems.

4.4 Data collection and preprocessing

Performance metrics. We used *PerfMon* [42] 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*. In order to reduce any influence by our recording tool, *Perfmon*, we monitor only the performance of the web and database applications on the two dedicated servers. We did not run *Perfmon* on the load driver machine. 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 the real world scenario, a user will consider a subject system as one box. To mimic this scenario we decided to combine the performance datasets from our application and database server. In order to combine the two datasets of performance metrics and system throughput, and to minimize noise of the performance metric recording, we calculate the mean values of the performance metrics in every minute. Then, we 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 [18].

5 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 work done by the underlying architecture or the overhead induced by the virtual environment. Instead, our experiments are set in the context of analyzing performance testing data, based on the related work. Shown in Section 2, prior research and practitioners examines performance testing results in three types of approaches: 1) examining individual performance metrics, 2) examining the relationship among performance metrics and 3) building statistical models using performance metrics. Therefore, our experiments focus on examining the discrepancy that may impact three such approaches.

5.1 Is the trend or distribution of individual performance metrics similar across environments?

Motivation. The most intuitive approach of examining performance testing results is to examine individual performance metrics. The term individual here refers to, for example, comparing a CPU performance metric's trend from one environment trend the same performance metric from the other environment. As shown in Section 2.1.1, prior studies propose different approaches that typically compare the distribution or trend of each performance metric from different tests. Primarily, this approach is used to identify the trends and distribution of performance metrics. As a result, we can look at the trends at a finer level that may not be represented numerically. Due to the difference between testing environments, performance testing results are expected to be different in raw value. However, the shape of distribution and the trend should be similar. For example, when there is higher low, CPU increases. If in one environment, we observe the CPU has increasing trend while not seeing the same trend in another environment, we observe a discrepancy. Therefore, we use QQ plot and normalized KS tests to examine the differences in trends and shape of the distributions.

Approach. After running and collecting the performance metrics, we compare every individual performance metric between the 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 follows the same shape of the distribution and the same trend in virtual and physical environments.

First, we plot a quantile-quantile plot (Q-Q plot) [47] 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 indi-

cates that the performance metrics in the virtual and physical environments come from populations with different shapes of distributions, which can lead to a different set of conclusions. For example, the virtual environment has a CPU's utilization spike at a certain time, but the spike is absent in the physical environment.

Second, to quantitatively measure the discrepancy, we perform a Kolmogorov-Smirnov test [55] 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 median 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 [62]. 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 to use the 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 the corresponding performance metric in the physical environment, in order to assess whether the same performance metrics 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 present between the subject systems. The impact of the subject systems warrants its own study.

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

The majority of the performance metrics had statistically significantly different distributions (p-values lower than 0.05 in Kolmogorov-Smirnov tests). Only 13 and 12 metrics (out of 44 for each environment) 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 differences between the distributions in the 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 for which we shared the 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 the 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 in virtual and physical environments. We conclude that there exists a discrepancy between physical and virtual environment. Practitioners cannot compare individual performance metric with a simple scaling factor applied to the metrics.

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

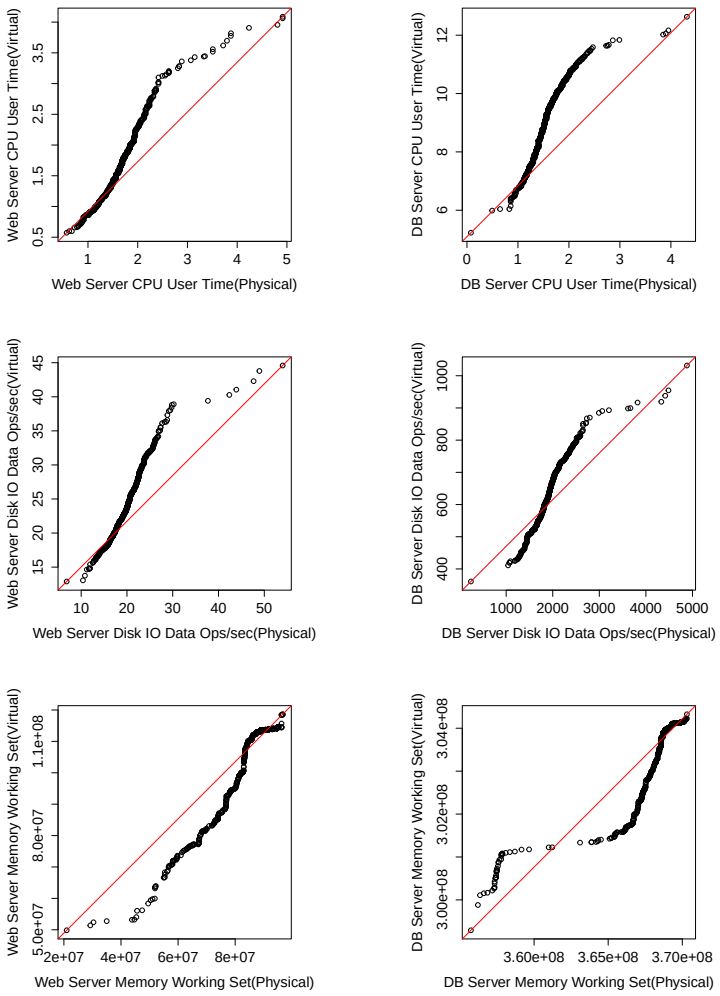

Fig. 2: Q-Q plots for DS2.

5.2 How much has the relation changed between the performance metrics across environment?

Motivation. The relationship between two performance counters may significantly change between two environments, which may be a hint of performance issues or system regressions. We have used the following approach to examine the relationships between the performance metrics. As claimed by Cohen et al. [12], combinations of performance metrics are significantly more predictive towards performance issues than individual metrics. A change in these combinations of relationships can reflect

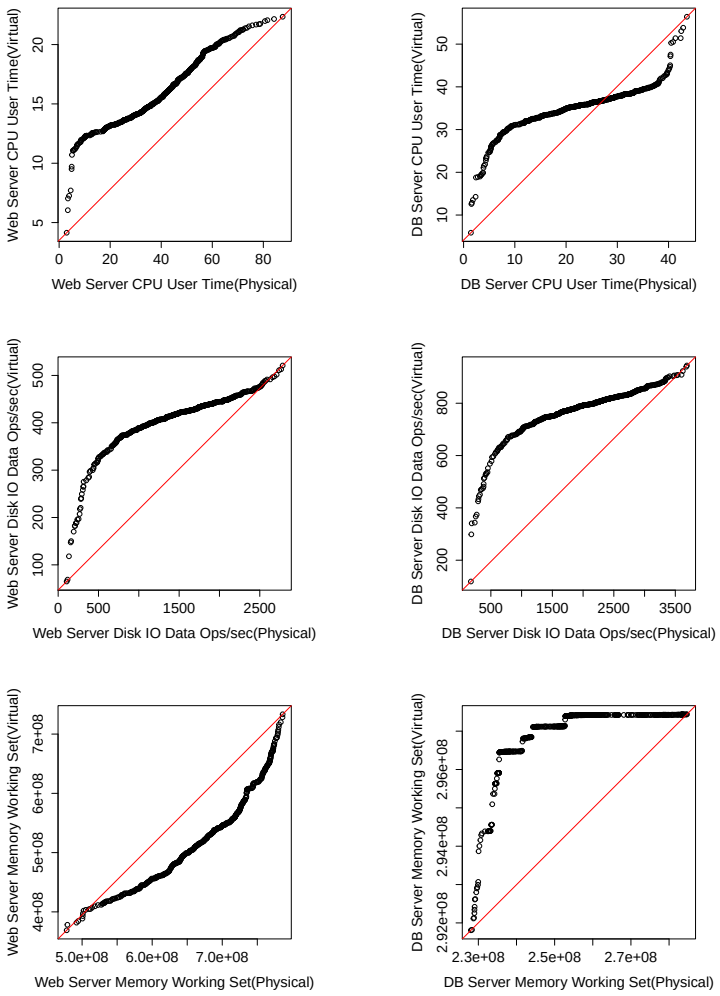

Fig. 3: Q-Q plots for CloudStore.

the discrepancy of performance and can help a practitioner identify the behavioral changes of a system between the two environments. For instance, in one release of the system, the CPU may be highly correlated with I/O while (e.g., when I/O is high, CPU is also high); while on a new release of the system, the correlation between CPU and I/O may become low. Such change to the correlation may expose a performance issue (e.g., the high CPU without I/O operation may be due to a performance bug). However, if there is a significant difference in correlations simply due to the platform being used, i.e., virtual vs. physical, then practitioners may need to be warned that a correlation discrepancy may be false. Therefore, we examine whether the relation-

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.

ship among performance metrics has a discrepancy between the virtual and physical environments.

Approach. We calculate Spearman's rank correlation coefficients among all the metrics from each performance test in each environment. Then we study whether such correlation coefficients are different between the virtual and physical environments.

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 consider there to be a discrepancy in the correlation coefficient between CPU and the system throughput. Second, for every pair of metrics, 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 between a pair of performance metrics. With the heatmap, we can quickly spot the metrics that have large discrepancy in correlation coefficients.

Results. The correlation between system throughput and performance metrics changes between virtual and physical environments. Tables 3 and 4 present the top ten metrics with the highest correlations to system throughput in the physical environment for DS2 and CloudStore, respectively. We chose system throughput to be our touchstone as it was kept identical between the environments. We find that for these top ten metric sets, the difference in correlation coefficients in virtual and physical environments is up to **0.78** and the rank changes from #9 to #40 in DS2 and #1 to #10 in CloudStore.

There exist differences in correlation among the performance metrics from virtual and physical environments. Figures 4 and 5 present the heatmap showing the changes in correlation coefficient among the performance metrics from virtual and physical environments. By looking at the heatmap, we find hotspots (with lighter color), which have larger correlation differences. For the sake of brevity, 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 Figures 4 and 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 ma-

chines [31, 40]. In such a situation, when practitioners observe that the relationship between I/O metrics and other metrics change, the change may not indicate a performance regression, but rather the change may be due to the use of a virtual environment.

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 I/O metrics have the largest discrepancy.

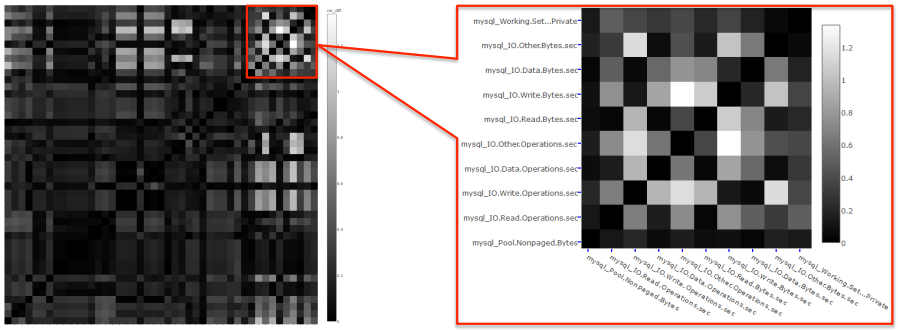

Fig. 4: Heatmap of correlation changes for DS2.

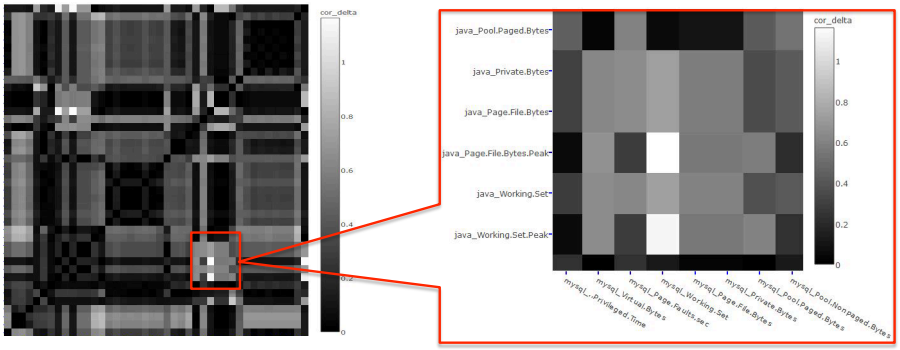

Fig. 5: Heatmap of correlation changes for CloudStore.

5.3 What is the effect on statistical models using the performance metrics?

Motivation. As discussed in the last step (see Section 5.2), the relationship among performance metrics is critical for examining performance testing results (see Section 2.1.2). However, thus far we have only examined the relationships between two

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: Top ten metrics with highest correlation coefficient to system throughput in the physical environment for CloudStore

Rank	Performance Metrics	Coef. PE	Coef. VE	Rank in VE
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

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

performance metrics. In order to capture the relationship among a large number of performance metrics, more complex modeling techniques are needed. Hence, we use statistical modeling techniques to see the collective impact of the performance metrics [12, 66]. This analysis may also serve as the basis of a future study (for example, fault detection). More importantly, some performance metrics do not have any impact with system performance, which are still examined. For example, for a software system that is CPU intensive, I/O operations may be irrelevant. Such performance metrics may expose large discrepancies between virtual and physical environments, but they do not impact the examination of performance testing results. It is necessary to remove the performance metrics that are not contributing or impacting the results of the performance analysis. To address the above issues, modeling techniques are proposed to examine performance testing results (see Section 2.1.3). In this step, we examine whether the modeling among performance metrics has a discrepancy

between the virtual and physical environments and whether we can minimize such discrepancy between performance models.

Approach.

We follow a model building approach that is similar to the approach from prior research [13,52,66]. We first build statistical models using performance metrics from one environment, then we test the accuracy of our performance model with the metric values from the same environment and also from a different environment. For example, if the model was built in a physical environment it was tested in both, physical and virtual environment.

5.3.1 B-1: Reducing counters

Mathematically, performance metrics that show little or no variation do not contribute to the statistical models hence we first remove performance metrics that have constant values in the test results. We then perform a correlation analysis on the performance metrics to remove multicollinearity based on statistical analysis [32]. We used the Spearman's rank correlation coefficient among all performance metrics from one environment. We find the pair of performance metrics that have a correlation higher than 0.75 [57]. 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 exists no correlation higher than 0.75.

We then perform redundancy analysis on the performance metrics. The redundancy analysis would consider a performance metric redundant if it can be predicted from a combination of other metrics [20]. We use each performance metric as a dependent variable and use the rest of the metrics as independent variables to build a regression model. We calculate the R^2 of each model. R^2, or the coefficient of multicollinearity, is used to analyze how a change in one of the variables (e.g. predictor) can be explained by the change in the second variable (e.g. response) [2]. We consider multicollinearity to be present if more than one predictor variable can explain the change in the response variable. If the R^2 is larger than a threshold (0.9) [57], 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.

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].

5.3.2 B-2: Building statistical models

In the second step, we build a linear regression model [19] using the performance metrics that are left after the reduction and removal of statistically insignificant metrics in the previous step as independent variables and use the system throughput as our dependent variable. We chose the linear regression model over other models because

of its uncomplicated explanation. Hence, it is easier to interpret the discrepancy that is illustrated by the model. Similar models have been built in prior research [13,52,66].

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.

5.3.3 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 model has 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 does not explain the variability of the response data. We also would like to estimate the impact that each independent variable has on the model fit. We follow a "drop one" approach [8], 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 is reported by comparing the performance of these two models [20]. 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 [39]. We then rank the independent variables by their impact on model fit.

5.3.4 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 train on the remaining nine partitions [30,49], similar to prior research [36]. 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 (*MAPE*) for all the data records.

5.3.5 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 the other 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 5.1 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.* [44]. The approach uses the load of the system to normalize the performance metric values across different environments. As there are varying inputs for the performance tests that we carried out, normalization by load helps in normalizing the multi-modal distribution that might be because of the trivial tasks like background processes(bookkeeping).

To normalize our metrics, we first build a linear regression model with the one metric as an 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.

5.3.6 Identifying model discrepancy

In order to identify the discrepancy between the models built 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) modeling using the virtual environment and testing internally (on data from the virtual environment), 2) modeling using the virtual environment and testing externally (on data from the physical environment), 3) modeling using the physical environment and testing internally (on data from the physical environment), 4) modeling 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 performance metrics leveraged by the models in virtual and physical environments are not the same. Tables 5 and 6 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 a good fit, denoted by the R^2 values (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 the 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 the virtual environment are related to the web server's memory (see Table 6). 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 5).

The prediction error illustrates discrepancies between models built in 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 7 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 internal validation is only 2% and 10% for the models built in the physical and virtual environments, 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.

One possible reason is that the normalization based on load performs better, even though it is shown to be effective in prior research [44], 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.

We find that the statistical models built by performance testing results in an environment cannot advocate for the other environment due to discrepancies present. Normalizing the performance metrics by deviance may minimize such discrepancy and should be considered by practitioners before examining performance testing results.

Table 5: Summary of statistical models built for DS2. The metrics listed in the table are the significant independent variables.

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

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

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

6 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.

6.1 Investigating the stability of virtual environments

Thus far, we perform our case studies in one virtual environment and compare the performance metrics to the physical environment. However, the stability of the results obtained from the virtual environment need to be validated, in particular since VMs tend to be highly sensitive to the environment that they run in [33].

Table 7: 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

In order to study whether the virtual environment is stable, we repeat the same performance tests, twice more, on the virtual environments for both subject systems. In total, we had results from three performance tests. We perform the data analysis in Section 5.3 by building statistical models using performance metrics. As the previously mentioned approach, we build a model based on one of the runs, serving as our training data for the model, and tested it on another run. In this case, we define external validation when a model is trained on a different run than it is tested on. We validate our model by predicting the throughput of a different run.

Prediction error values (see section 4.3.5) closer to 0 indicate that our model was able to successfully explain the variation of the throughput of a different run. This also means that the external validation error closer to 1 or higher depicts instability of the environment. We find the external validation error to be 0.04 and 0.13 for CloudStore and DS2, respectively. The internal validation error is 0.03 and 0.09 for CloudStore and DS2, respectively. Such low error values show that the performance testing results from the virtual environments are rather stable.

6.2 Investigating the Impact of Specific Virtual Machine Software

In all of our experiments, we used the Virtual Box software to setup our virtual environment. However, there exists a plethora of VM software (i.e., it can be argued that our chosen subject systems behave differently in another environment). The question that arises then is whether the choice of VM software impacts our findings. In order to address the aforementioned hypothesis, we set up another virtual environment using VMWare (version 12) with the same allocated computing resources as when we set up Virtual Box.

To investigate this phenomenon, 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.

Table 8 shows that the performance testing results from the two different virtual machine software is similar, as supported by the low percentage error when our model was tested on Virtual Box. 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 7). Such results show that the discrepancy observed during our experiment also exits with the virtual environments that are set up with VMWare.

Table 8: 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

6.3 Investigating the Impact of Allocated Resources

Another aspect that may impact our results is the resources allocated and the configuration of the virtual environment. We did not decrease the system resources as decreasing the resources may lead to crashes in the testing environment.

To investigate the impact of the allocated resources, 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. 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 7, the prediction error is high when we normalize by the load as per Equation 1 (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). We conclude that our findings still hold when the allocated resources are changed and this change has minimal impact on the results of our case studies.

7 Threats to Validity

7.1 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 has studied both systems [1, 26, 44].

The virtual machine software that we used is 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.

7.2 Internal Validity.

Our approach is based on the recorded performance metrics however the quality of performance tests may still have room for improvement. 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 [50] may address this threat. We followed the approaches in the related work to introduce the workload variation on our subject systems however we acknowledge that this is not the only way. In the future, we also plan to run the tests for a longer period of time (for example, 72 hours). 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 [13, 52, 66]. We also consider the subject systems as a whole from the user's point of view. Hence, we do not test just one of the functionalities, for example an atomic SQL query. In the future we also plan to conduct performance tests on an isolated lower level components. The performance testing carried out on the subject systems was dependent on the use cases which were predefined in the testing suite which may also be another limitation of our subject systems.

In the real world, the systems may have different interferences to impact their performance. However, in our experiments, we opt for a more controlled environment to better understand the differences without any interference, hence we can limit the chance that the discrepancy is from handling interference rather than the environments. Future work can be applied to investigate the performance impact from different environments by handling interference, a better controlled experiment with having the knowledge of environment discrepancy. On the other hand, we agree that system load counters may illustrate valuable knowledge of the system. We include throughput metrics that are associated with system load. In addition, we will include more metrics in our future work.

7.3 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. 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.

8 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. Developers cannot assume an straightforward overhead from the virtual environment, (such as a simple increment of CPU).
- Prior approach that are proposed to normalize performance testing results with different loads may not work between physical and virtual 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 to be aware of the discrepancy between performance testing results in virtual and physical environments, for both practitioners and researchers. Our ultimate goal is to have investigate the impact on the detection of real world performance bugs. This paper is the first step to lay a ground to deeply understand such discrepancy. With the knowledge of such discrepancy, we can, in the future, better understand the existence and magnitude of impact on detecting real world performance bugs. In addition, another contribution of our paper, beside the road towards real world performance bugs, is the effort of identifying approaches that may minimize the discrepancy. In particular, we find that the approach previously proposed by for normalizing performance testing results with different loads

does not effectively reduce the discrepancy and we propose a new approach that is shown to be more effective. Our future work on evaluating the impact of real world performance bugs will based on the testing results that are processed with reduced discrepancy. 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. 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)
2. Andale: Statistics how to - coefficient of determination (r squared). http://www.statisticshowto.com/what-is-a-coefficient-of-determination/ (2012). Accessed: 2017-04-04
3. Apache: Jmeter. http://jmeter.apache.org/. Accessed: 2015-06-01
4. Apache: Tomcat. http://tomcat.apache.org/. Accessed: 2015-06-01
5. Bodík, P., Goldszmidt, M., Fox, A.: Hilighter: Automatically building robust signatures of performance behavior for small- and large-scale systems. In: Proceedings of the Third Conference on Tackling Computer Systems Problems with Machine Learning Techniques, SysML'08, pp. 3–3 (2008)
6. 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)
7. CA Technologies: The avoidable cost of downtime. http://www3.ca.com/~/media/files/articles/avoidable_cost_of_downtime_part_2_ita.aspx
8. 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)
9. Chen, P.M., Noble, B.D.: When virtual is better than real [operating system relocation to virtual machines]. In: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, 2001., pp. 133–138 (2001)
10. Cito, J., Leitner, P., Fritz, T., Gall, H.C.: The making of cloud applications: An empirical study on software development for the cloud. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, pp. 393–403 (2015)
11. CloudScale-Project: Cloudstore. https://github.com/CloudScale-Project/CloudStore. Accessed: 2015-06-01
12. Cohen, I., Goldszmidt, M., Kelly, T., Symons, J., Chase, J.S.: Correlating instrumentation data to system states: A building block for automated diagnosis and control. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6, OSDI'04, pp. 16–16 (2004)
13. 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)
14. Dave Jaffe, T.M.: Dell dvd store. http://linux.dell.com/dvdstore/. Accessed: 2015-06-01
15. Dean, J., Barroso, L.A.: The tail at scale. Communications of the ACM **56**, 74–80 (2013)
16. Dee: performance-testing systems on virtual machines that normally run on physical machines. http://sqa.stackexchange.com/questions/7709/performance-testing-systems-on-virtual-machines-that-normally-run-on-physical-ma (2014). Accessed: 2017-04-04
17. Eeton, K.: 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. Accessed: 2016-03-11
18. 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 (2010)
19. Freedman, D.: Statistical models: theory and practice. Cambridge University Press (2009)

20. Harrell, F.E.: Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis. Springer (2001)
21. 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)
22. Huber, N., von Quast, M., Hauck, M., Kounev, S.: Evaluating and modeling virtualization performance overhead for cloud environments. In: Proceedings of the 1st International Conference on Cloud Computing and Services Science, pp. 563–573 (2011)
23. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley (1990)
24. Jiang, M., Munawar, M., Reidemeister, T., Ward, P.: Automatic fault detection and diagnosis in complex software systems by information-theoretic monitoring. In: Proceedings of 2009 IEEE/IFIP International Conference on Dependable Systems Networks, pp. 285–294 (2009)
25. Jiang, M., Munawar, M.A., Reidemeister, T., Ward, P.A.: System monitoring with metric-correlation models: Problems and solutions. In: Proceedings of the 6th International Conference on Autonomic Computing, pp. 13–22 (2009)
26. Jiang, Z.M., Hassan, A.E., Hamann, G., Flora, P.: Automated performance analysis of load tests. In: IEEE International Conference on Software Maintenance, 2009. ICSM 2009., pp. 125–134 (2009)
27. Jin, G., Song, L., Shi, X., Scherpelz, J., Lu, S.: Understanding and detecting real-world performance bugs. In: Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '12, pp. 77–88. ACM (2012)
28. Kabacoff, R.I.: R In Action. In: R In Action, pp. 207–213. Manning Publications Co., Staten Island, NY (2011)
29. Kearon, S.: Can you use a virtual machine to performance test an application? http://stackoverflow.com/questions/8906954/can-you-use-a-virtual-machine-to-performance-test-an-application (2012). Accessed: 2017-04-04
30. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'95, pp. 1137–1143 (1995)
31. Kraft, S., Casale, G., Krishnamurthy, D., Greer, D., Kilpatrick, P.: Io performance prediction in consolidated virtualized environments. SIGSOFT Softw. Eng. Notes **36**(5), 295–306 (2011)
32. Kuhn, M.: findcorrelation. https://www.rdocumentation.org/packages/caret/versions/6.0-73/topics/findCorrelation/ (2017). Accessed: 2017-04-04
33. Leitner, P., Cito, J.: Patterns in the chaos—a study of performance variation and predictability in public iaas clouds. ACM Trans. Internet Technol. **16**(3), 15:1–15:23 (2016)
34. 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 (2016)
35. 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)
36. 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)
37. 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)
38. 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: 2010 14th European Conference on Software Maintenance and Reengineering, pp. 222–231 (2010)
39. Mcintosh, S., Kamei, Y., Adams, B., Hassan, A.E.: An empirical study of the impact of modern code review practices on software quality. Empirical Softw. Engg. **21**(5), 2146–2189 (2016)
40. 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 (2005)
41. Merrill, C.L.: Load testing sugarcrm in a virtual machine. http://www.webperformance.com/library/reports/Virtualization2/ (2009). Accessed: 2017-04-04

42. Microsoft Technet: Windows performance counters. https://technet.microsoft.com/en-us/library/cc780836(v=ws.10).aspx. Accessed: 2015-06-01
43. Netto, M.A., Menon, S., Vieira, H.V., Costa, L.T., De Oliveira, F.M., Saad, R., Zorzo, A.: Evaluating load generation in virtualized environments for software performance testing. In: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, pp. 993–1000. IEEE (2011)
44. 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 (2012)
45. Nistor, A., Jiang, T., Tan, L.: Discovering, reporting, and fixing performance bugs. In: 2013 10th Working Conference on Mining Software Repositories (MSR), pp. 237–246 (2013)
46. Nistor, A., Song, L., Marinov, D., Lu, S.: Toddler: Detecting performance problems via similar memory-access patterns. In: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pp. 562–571. IEEE Press, Piscataway, NJ, USA (2013)
47. NIST/SEMATECH: e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm. Accessed: 2015-06-01
48. Oracle: MYSQL server 5.6. https://www.mysql.com/. Accessed: 2015-06-01
49. Refaeilzadeh, P., Tang, L., Liu, H.: Encyclopedia of Database Systems, chap. Cross-Validation, pp. 532–538. Springer US, Boston, MA (2009)
50. Rodola, G.: Psutil. https://github.com/giampaolo/psutil. Accessed: 2015-06-01
51. Sage, S.: Blackberry enterprise server 12 now available. http://crackberry.com/blackberry-enterprise-server-12-now-available (2014). Accessed: 2017-04-04
52. 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 (2015)
53. Shewhart, W.A.: Economic control of quality of manufactured product, vol. 509. ASQ Quality Press (1931)
54. Srion, E.: The time for hyper-v pass-through disks has passed. http://www.altaro.com/hyper-v/hyper-v-pass-through-disks/ (2015). Accessed: 2017-04-04
55. Stapleton, J.H.: Models for Probability and Statistical Inference: Theory and Applications. WILEY (2008)
56. 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: 29th IEEE International Conference on Software Maintenance (ICSM '13), pp. 110–119 (2013)
57. Syer, M.D., Shang, W., Jiang, Z.M., Hassan, A.E.: Continuous validation of performance test workloads. Automated Software Engineering pp. 1–43 (2016)
58. Tintin: Performance test is not reliable on virtual machine? https://social.technet.microsoft.com/Forums/windowsserver/en-US/06c0e09b-c5b4-4e2c-90e3-61b06483fe5b/performance-test-is-not-reliable-on-virtual-machine?forum=winserverhyperv (2011). Accessed: 2017-04-04
59. TPC: TPC-W. www.tpc.org/tpcw. Accessed: 2015-06-01
60. Tsakiltsidis, S., Miranskyy, A., Mazzawi, E.: On automatic detection of performance bugs. In: 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 132–139 (2016)
61. VMWare: Accelerate software development and testing with the vmware virtualization platform. http://www.vmware.com/pdf/development_testing.pdf. Accessed: 2016-03-16
62. Walker, H.M.: Studies in the history of statistical method: With special reference to certain educational problems. Williams & Wilkins Co (1929)
63. Wikipedia: Statistical significance. https://en.wikipedia.org/wiki/Statistical_significance/ (2017). Accessed: 2017-04-04
64. Wikipedia: Sugarcrm. https://en.wikipedia.org/wiki/SugarCRM#Deployment_options (2017). Accessed: 2017-04-04
65. Woodside, M., Franks, G., Petriu, D.C.: The future of software performance engineering. In: Future of Software Engineering, 2007., pp. 171–187 (2007)
66. 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 (2013)
67. Zaman, S., Adams, B., Hassan, A.E.: A qualitative study on performance bugs. In: 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), pp. 199–208 (2012)