CrossMark

Empirical study on the discrepancy between performance testing results from virtual and physical environments

Muhammad Moiz Arif[1] · **Weiyi Shang**[1] · **Emad Shihab**[1]

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 confounding overhead (e.g., a higher than expected memory utilization with unstable I/O traffic) to the testing environment and lead to unrealistic performance testing results. Yet, little research has studied the 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 (DS2) 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) single performance metric (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) A single metric from virtual and physical environments do not follow the

---

✉ Muhammad Moiz Arif
  mo_ari@encs.concordia.ca; m.moizarif@gmail.com

  Weiyi Shang
  shang@encs.concordia.ca

  Emad Shihab
  eshihab@encs.concordia.ca

[1] Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada

---

🌲 Springer

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 metrics from virtual environments are different from the models built from physical environments suggesting that practitioners cannot use the performance testing results across virtual and physical environments. In order to assist the practitioners leverage performance testing results in both environments, we investigate ways to reduce the discrepancy. We find that such discrepancy can be reduced by normalizing performance metrics based on deviance. Overall, we suggest that practitioners should not use the performance testing results from virtual environment with the simple assumption of straightforward performance overhead. Instead, practitioners should consider leveraging normalization techniques to reduce the discrepancy before examining performance testing results from virtual and physical environments.

Keywords Software performance engineering · Software performance analysis and testing on virtual 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 (Woodside et al. 2007). Often however, failures in large software systems are due to performance issues rather than functional bugs (Dean and Barroso 2013; Foo et al. 2010). Such failures lead to the eventual decline in quality of the system with reputational and monetary consequences (CA Technologies 2011). For instance, Amazon estimates that a one-second page-load slowdown can cost up to $1.6 billion (Eeton 2012).

In order to mitigate performance issues and ensure software reliability, practitioners often conduct performance tests (Woodside et al. 2007). Performance tests apply a workload (e.g., mimicking users' behavior in the field) on the software system (Jain 1990; Syer et al. 2017), and monitor performance metrics, such as CPU usage, that are generated based on the tests. Practitioners use such metrics to gage the performance of the software system and identify potential performance issues (such as memory leaks (Syer et al. 2013) and throughput bottlenecks (Malik et al. 2010a)).

Since performance tests are often performed on large-scale software systems, the performance tests often require many resources (Jain 1990). Moreover, performance tests often need to run for a long period of time in order to build statistical confidence on the results (Jain 1990). Such testing environments need 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. Hence, due to their flexibility, virtual environments enable practitioners to easily prepare, customize, use and update performance testing environments in an efficient manner. Therefore, to address such challenges, virtual environments (VMs) are often leveraged for performance testing (Chen and Noble 2001; VMWare 2016). The use of VMs in performance testing are widely discussed (Dee 2014; Kearon 2012; Tintin 2011), and even well documented (Merrill 2009) by practitioners. In addition, many software systems are released both on-premise (physical) and on cloud (virtual) environment (e.g., SugarCRM 2017 and BlackBerry Enterprize Server 2014). Hence, it is important to conduct performance testing on both the virtual (for cloud deployment) and physical environments (for on-premise deployment).

Prior studies show that virtual environments are widely exploited in practice (Cito et al. 2015; Nguyen et al. 2012; Xiong et al. 2013). Studies have investigated the overhead that is associated with virtual environments (Menon et al. 2005). 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 both, the 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 such 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 (Jaffe and Muirhead 2011) and CloudStore (CloudScale-Project 2014), 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 analyzing 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:

– *single performance metric*: the trends and distributions of each performance metric
– *the relationship between the performance metrics*: the correlations between every two performance metrics
– *statistical performance models*: the models that are built using performance metrics to predict the overall performance of the system

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, i.e a change in correlation values. 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 3 presents the case study setup. Section 4 presents the results of our case 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

In this section, we discuss the motivation and related work of this paper in broadly three subsections: 1) analyzing performance metrics from performance testing, 2) analysis of VM overhead and 3) performance testing and bug detection.

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) single performance metric, 2) the relationship between performance metrics, and 3) statistical modeling based on performance metrics.

2.1.1 Single Performance Metric

Nguyen et al. (2012) introduce the concept of using control charts (Shewhart 1931) 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 and workloads in order to build robust control charts.

Malik et al. (2010b, 2013) 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. (2013) 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. (2009) 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 Between Performance Metrics

Malik et al. (2010a) leverage Spearman's rank correlation to capture the relationship between performance metrics. The deviance of correlation is calculated in order to pinpoint which subsystem should take responsibility of the performance deviation.

Foo et al. (2010) 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. (2009a) 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. (2013) 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. (2015) 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. (2004) 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. (2005) also proposed that system faults can be detected by building statistical models based on performance metrics. The approaches of Cohen et al. (2004, 2005) were improved by Bodík et al. (2008) by using logistic regression models.

Jiang et al. (2009b) 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 that their approach is more efficient in successfully detecting the injected faults than the current linear-model approach.

On one hand, none of the prior research discusses the impact of their approaches results in virtual and physical environments, which motivates the empirical study that is conducted in this paper. On the other hand, since there are hardly two identical performance testing results, we do no compare the raw data of performance testing results from virtual and physical environments. Instead, we conduct our case study in the context of all the above three types of analyzes, in order to see the impact when practitioners use such analyzes on performance testing results. Our findings can help better evaluate and understand the findings from the aforementioned research.

2.2 Analysis of VM Overhead

Kraft et al. (2011) discuss the issues 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 disk IO due to increments in time associated with request queues.

Menon et al. (2005) 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. (2013) 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. (2011) 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. (2016) converge the set of inputs that may cause software regression. They apply genetic algorithms to detect such combinations. Netto et al. (2011) present a similar study to compare performance metrics generated via load tests between the two environments. However, the author did not analyze 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 analyzes and investigate whether such impact can be reduced in practice.

2.3 Performance Testing and Bug Detection

There exists much research on performance testing and bug detection. Nistor et al. (2013b) detect the presence of functional and loop-related performance bugs with the help of their developed tool. Jin et al. (2012) present a study on a wide range of performance bugs. The authors examined real-world performance bugs and developed rule-based performance bug detection tools. Nistor et al. (2013a) in another study highlight that automated tool based performance bug detection 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 users. Tsakiltsidis et al. (2016) use prediction models to detect and predict performance bugs based on extraction from source code repositories. Malik et al. (2010c) 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. (2012) 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 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) with the same hardware. 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) single performance metric 2) relationship between performance metrics and 3) statistical models based on the performance metrics. An overview of our case study setup is shown in Fig. 1.

3.1 Subject Systems

Dell DVD Store (DS2) (Jaffe and Muirhead 2011) is an online multi-tier e-commerce web application that is widely used in performance testing and prior performance engineering research (Shang et al. 2015; Nguyen et al. 2012; Jiang et al. 2009). We deploy DS2 (SLOC > 3,200) on an Apache (Version 3.0.0) web application server with MySQL 5.6 database server (Oracle 1998). CloudStore (CloudScale-Project 2014), our second subject system, is an open source application based on the TPC-W benchmark (TPC 2001). CloudStore (SLOC > 7,600) is widely used to evaluate the performance of cloud computing infrastructure when hosting web-based software systems and is leveraged in prior research (Ahmed
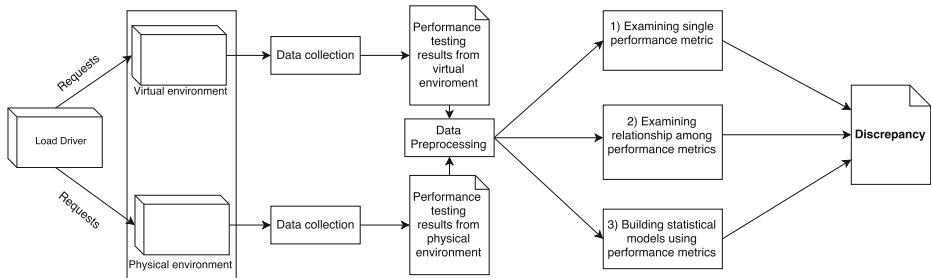
Fig. 1 An overview of our case study setup

et al. 2016). We deploy CloudStore on *Apache Tomcat* (Apache 2007) (version 7.0.65) with MySQL 5.6 database server (Oracle 1998).

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, 100GB SATA storage and connected to a local gigabyte ethernet. The first machine hosts the application servers (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 application 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, as follows:

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 not topping out and pushing our configuration for unrealistic results. Virtual machines typically have an option of using disk pass-through (Costantini 2015). However, disk pass-through prevents the quick deployment of an existing virtual machine image that's designed for performance testing and quick execution of performance tests (Srion 2015). Hence, we opt to disable disk pass-through since it is unlikely to be used in practice. The network of the virtual machine is set up based on network address translation (NAT) configuration (Tyson 2001). The network traffic of the workload was generated on a dedicated load machine to keep our experiments as close to the real-world as possible.

Physical Environment We used the same hardware as the virtual environment to set up our physical environments. To make the physical environment similar to the virtual environment, we only enable two cores and three gigabytes of 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 (Apache 2008) 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 represents 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. 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 after every test.

3.4 Data Collection and Preprocessing

Performance Metrics We used *PerfMon* (Microsoft Technet 2007) 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 run *PerfMon* on each of the application server and database server machines. We record all the available performance metrics that can be monitored on a single process by *PerfMon*. In order to minimize the influence of *Perfmon*, we monitor only the performance of the two processes of the application server and database server on the two dedicated machines. We recorded the performance metrics with an interval of 10 seconds. In total, we recorded 44 performance metrics.

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

Since an end user will consider a system as a whole, we combine the performance datasets from our application and database servers. 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 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 (Foo et al. 2010).

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. 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 examine performance testing results in three types of approaches: 1) examining a single performance metric, 2) examining the relationship between performance metrics

and 3) building statistical models using performance metrics. Therefore, our experiments are designed to answer three research questions, where each questions corresponds to one of the types of analysis above.

4.1 Are the Trend and Distribution of a Single Performance Metric Similar Across Environments?

Motivation The most intuitive approach of examining performance testing results is to examine every single performance metric. 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. Due to influences from testing environments, performance testing results are not expected to be identical in raw values. However, the shape of distribution and the trend of the metrics should be similar. For example, if in one environment, we observe the Memory has increasing trend while the increasing trend is not seen in another environment, we observe a discrepancy. In addition, the distribution differences between two test results should not be statistically significant. Therefore, we use quantile-quantile (Q-Q) plot and normalized Kolmogorov-Smirnov (KS) tests to examine the differences in trends and shape of the distributions.

Approach After running and collecting the performance metrics, we compare every single 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 (Q-Q) plot (NIST/SEMATECH 2003) 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, 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 (Stapleton 2008) 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 (Walker 1929). 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 a 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. Figures 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 application 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 application (web) server's 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 application (web) server's 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.

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., application 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., application 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's rank correlation coefficient and corresponding p-value between the selected performance metrics for which we shared the Q-Q plots. We find that for the application 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

---

[1] The complete results, data and scripts are shared online at http://das.encs.concordia.ca/members/moiz-arif/
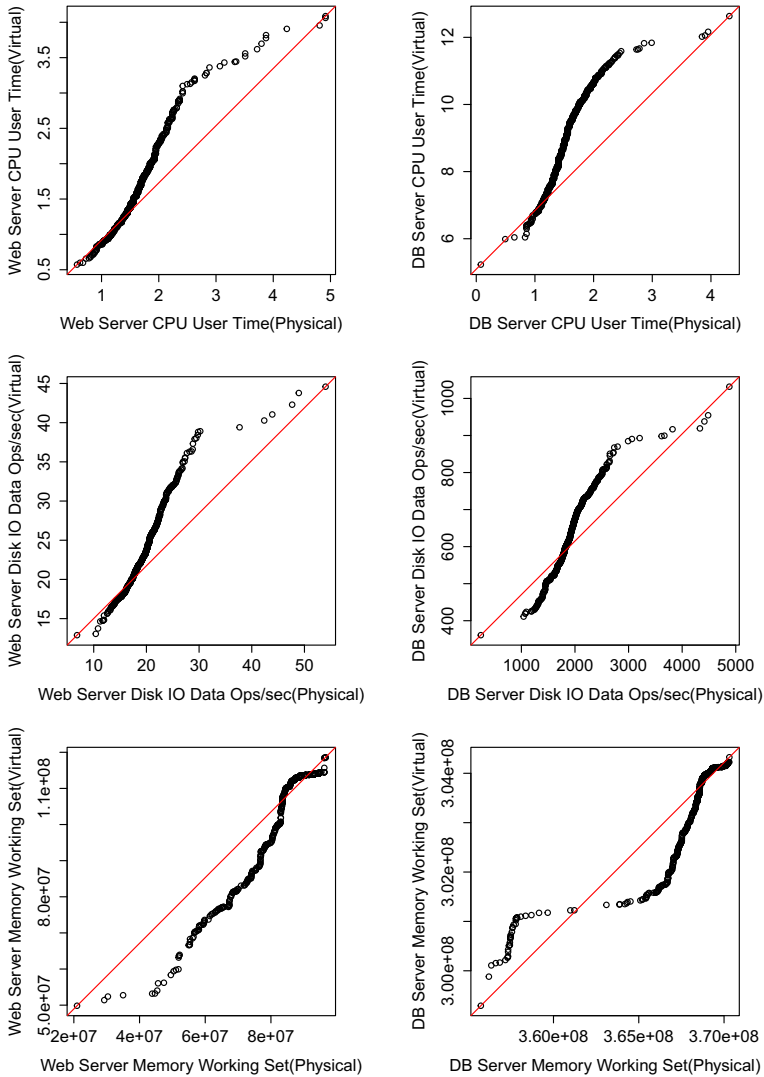
Fig. 2 Q-Q plots for DS2

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).

Impact on the interpretation of examining single performance metric. Practitioners often plot the trend of each important performance metrics, identify when the outliers exist or calculate the median or mean value of the metric to understand the performance of the system in general. However, based on our findings in this RQ, such analysis results may not be useful if they are from a virtual environment. For example, shown in Figs. 2 and 3 many differences between the two distribution are in the lower and higher ends of the plots, which corresponds to the high and low values of the metrics. Such values are often treated as
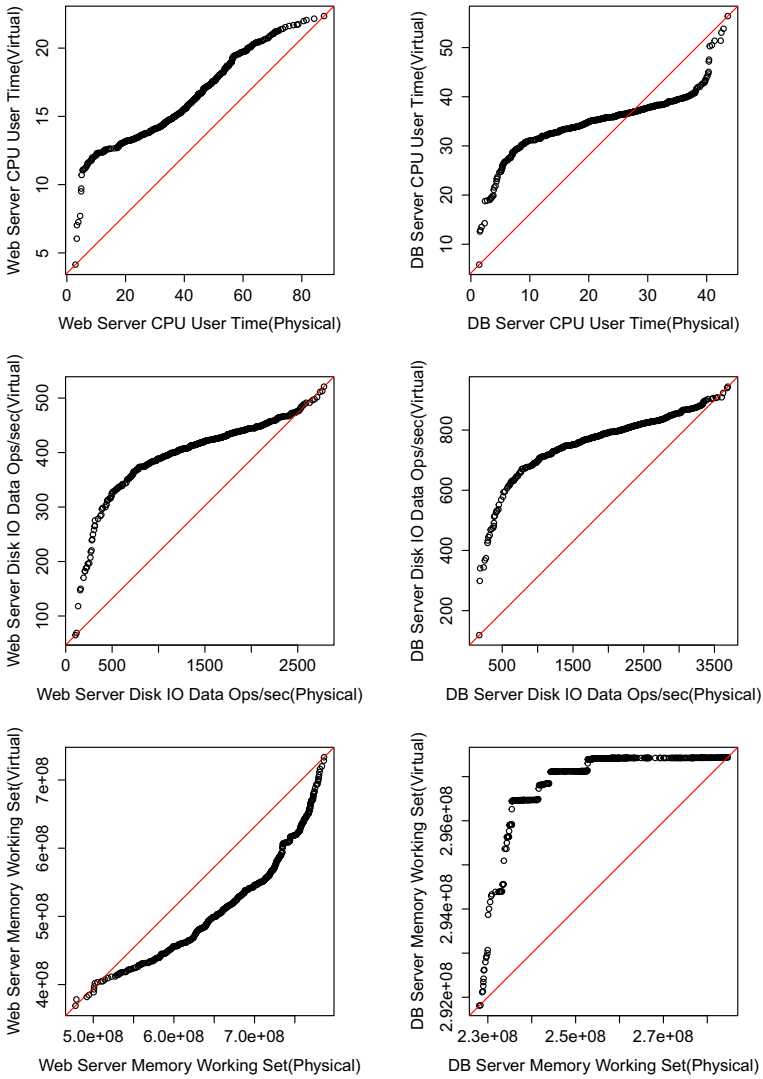
Fig. 3 Q-Q plots for CloudStore

outliers to be examined. However, if such outliers are due to the virtual environment rather than the system itself, the results may be misleading. In addition, since the distribution of the metrics are statistically different, the mean and median value of the metrics may also be misleading.

Findings: Performance metrics typically do not follow the same distribution in virtual and physical environments.
Actionable implications: Practitioners cannot assume a straightforward overhead from the virtual environment nor compare single performance metric after applying a simple scaling factor to the metric.

Table 1 Spearman's rank correlation coefficients and p-values of the highlighted performance metrics for which we shared the Q-Q plots, in virtual and physical environments

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

4.2 To what Extent does the Relationship Between the Performance Metrics Change Across Environments?

Motivation The relationship between two performance metrics may significantly change between two environments, which may be a hint of performance issues or system regressions. As found by Cohen et al. (2004), combinations of performance metrics are significantly more predictive toward performance issues than a single metric. A change in these combinations can reflect 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 relationship 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

Table 2 Summary of Spearman's rank correlation p-values and absolute coefficients of all the performance metrics, in virtual and physical environments

System	p-value>0.05	p-value<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

The numbers in the table are the number of metrics that fall into each category

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

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 correlations between system throughput and performance metrics change 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 criterion 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 Figs. 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 overhead in virtual machines (Menon et al. 2005; Kraft et al. 2011). 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.

Impact on the interpretation of examining correlations between performance metric. When a system is reported to have performance issues, correlations between metrics are

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

often used in practice, as describe in the motivation of this RQ. However, since such correlation can be inconsistent in virtual and physical environment, existing knowledge of assumptions of correlation may not exist or new correlation may emerge, due to the use of virtual environment. For example, practitioners of a database-centric system may have the knowledge that I/O traffic is correlated with CPU and system throughput. Examining these three metrics together can help diagnose performance issues, while if no such correlation exists in the virtual environment, these three metrics together may not be as useful in performance issue diagnosis.

> **Findings:** 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.
> **Actionable implications:** Practitioners should always verify whether the inconsistency of correlations between performance metrics (especially I/O metrics) are due to virtual environments.
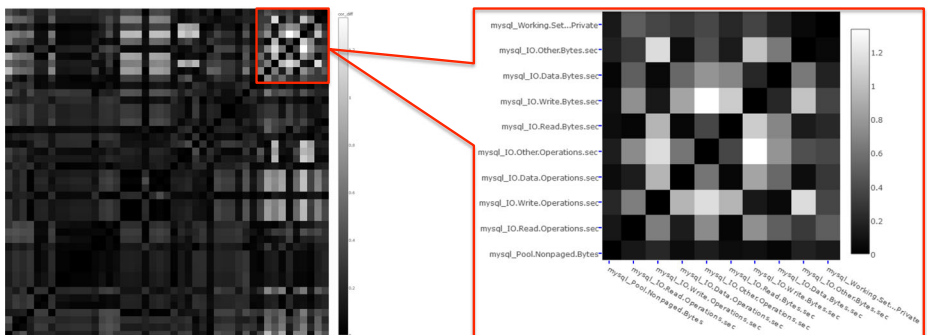

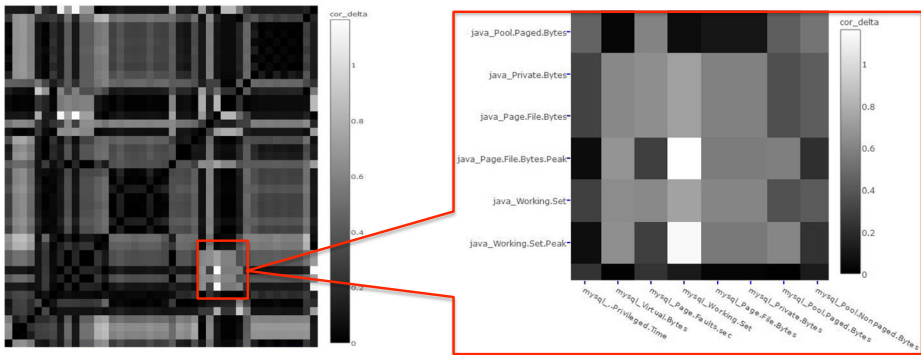**Fig. 4** Heatmap of correlation changes for DS2

Fig. 5 Heatmap of correlation changes for CloudStore

4.3 Can Statistical Performance Models be Applied Across Virtual and Physical Environments?

Motivation As discussed in the last research question (see Section 4.2), 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 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 examine the relationship among a set of performance metrics (Xiong et al. 2013; Cohen et al. 2004). Moreover, 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 while not contributing to the examination of performance testing results. It is necessary to remove such 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 can apply across 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 (Shang et al. 2015; Cohen et al. 2005; Xiong et al. 2013). 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 environments.

4.3.1 B-1: Reducing Metrics

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 (Kuhn 2008). 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, as 0.75 is considered to be a high correlation (Syer et al. 2017). 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 (Harrell 2001). 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) (Andale 2012). 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) (Syer et al. 2017), 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 (Kabacoff 2011).

4.3.2 B-2: Building Statistical Models

In the second step, we build a linear regression model (Freedman 2009) 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 simple explanation. Hence, it is easier to interpret the discrepancy that is illustrated by the model. Similar models have been built in prior research (Cohen et al. 2005; Xiong et al. 2013; Shang et al. 2015).

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.

4.3.3 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 would also like to estimate the impact that each independent variable has on the model fit. We follow a "drop one" approach (Chambers et al. 1990), 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 (Harrell 2001). 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 Mcintosh et al. (2016). We then rank the independent variables by their impact on model fit.

4.3.4 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 (Refaeilzadeh et al. 2009; Kohavi 1995), similar to prior research (Malik et al. 2013). 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.

4.3.5 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 single performance metric shown in (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. (2012). 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.

4.3.6 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 different. 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 (66.9 to 94.6% R^2 values). 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 application 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 statistically 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

Table 5 Summary of statistical models built for DS2

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%

The metrics listed in the table are the significant independent variables

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 (Nguyen et al. 2012), 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.

Impact on the interpretation of examining statistical performance models. Statistical performance models are often used to interpret relationships among many system performance metrics. For example, what are the significant metrics that are associated with system load and what performance metrics are redundant. Since the statistical performance models have

Table 6 Summary of statistical models built for CloudStore

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%

The metrics listed in the table are the significant independent variables

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

large discrepancy, even after applying normalization techniques that is proposed by prior research, we cannot directly use the performance models built in the virtual environment. Even though our results show that normalizing by deviance can reduce the discrepancy, practitioners should still be aware of it when examining the performance models.

Findings: We find that the statistical models built by performance testing results in an environment cannot advocate for the other environment due to discrepancies present. Normalization technique for heterogeneous environments and workloads that is proposed by prior research may not work for virtual and physical environment.

Actionable implications: Normalizing the performance metrics by deviance may minimize such discrepancy and should be considered by practitioners before examining performance testing results.

---

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 as 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 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 Leitner and Cito (2016).

In order to study whether the virtual environment is stable, 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. As the previously mentioned approach, we build a model based on one of the runs, serving as our

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

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.

5.2 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 (1)). We only apply the normalization by load.

Table 8 shows that the performance testing results from the two different virtual machine software are 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.

5.3 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 (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.

6 Threats to Validity

6.1 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 (Jiang et al. 2009; Nguyen et al. 2012; Ahmed et al. 2016). 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.

6.2 Internal Validity

Our empirical study is based on the performance testing results on subject systems. The quality and the way of conducting the performance tests may introduce threats to the validity of our findings. In particular, our approach is based on the recorded performance metrics. The quality of recorded performance metrics can have an impact the internal validity of our study. We followed the approaches in the prior research to control the workload and to introduce the workload variation on our subject systems. However, we acknowledge that there exist other ways of control and vary workload. Our performance tests all last for 9 hours, while the length of the performance tests may impact the findings of the case study. Replicating our study by using other performance monitoring tools, such as psutil (Rodola 2009), using other approaches to control and to vary the workload of the system and running the performance tests for a longer period of time (for example, 72 hours), 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 (Cohen et al. 2005; Shang et al. 2015; Xiong et al. 2013).

6.3 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.

In our performance tests, we consider the subject systems as a whole from the users' point of view. We did not conduct isolated performance testing for each feature or component of the system. Isolated performance testing may unveil more discrepancies than our results. Future work may consider such isolated performance tests to address this threat.

In practice, the system performance may be interfered by other environmental issues. However, in our experiments, we opt for a more controlled environment to better understand the discrepancy without any environmental interference, hence we can limit the possibility 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.

We recorded 44 performance metrics that are readily available from *PerfMon* and calculated throughput of the subject system. However, there may exist other valuable performance metrics, such as system load. Prior study shows that most performance metrics are often correlated to each other (Malik et al. 2010b). Future work may expand our list of performance metrics to address this threat.

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 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 single performance metric, 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 one of the first research that attempts to evaluate the discrepancy in the context of analyzing 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 heterogeneous environments and workloads 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 and to reduce the discrepancy between performance testing results in virtual and physical environments, for both practitioners and researchers.

Future Work This paper is the first step to lay a ground for a deeper understanding of the discrepancy between performance testing results in virtual and physical environments and the impact of detecting performance issues with 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. Moreover, future research effort can focus on generating comparable performance testing results from different environments with different workload.

References

Ahmed TM, Bezemer CP, Chen TH, Hassan AE, Shang W (2016) 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

Andale (2012) Statistics how to - coefficient of determination (r squared). http://www.statisticshowto.com/what-is-a-coefficient-of-determination/. Accessed: 2017-04-04

Apache (2007) Tomcat. http://tomcat.apache.org/. Accessed: 2015-06-01

Apache (2008) Jmeter. http://jmeter.apache.org/. Accessed: 2015-06-01

Blackberry (2014) Blackberry enterprise server. https://ca.blackberry.com/enterprise. Accessed: 2017-04-04

Bodík P, Goldszmidt M, Fox A (2008) 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

Brosig F, Gorsler F, Huber N, Kounev S (2013) Evaluating approaches for performance prediction in virtualized environments. In: 2013 IEEE 21st international symposium on modelling, analysis and simulation of computer and telecommunication systems. IEEE, pp 404–408

CA Technologies (2011) The avoidable cost of downtime. http://www3.ca.com/~/media/files/articles/avoidable_cost_of_downtime_part_2_ita.aspx

Chambers J, Hastie T, Pregibon D (1990) Statistical models in S. In: Compstat: proceedings in computational statistics, 9th symposium held at Dubrovnik, Yugoslavia, 1990. Physica-Verlag HD, Heidelberg, pp 317–321

Chen PM, Noble BD (2001) 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

Cito J, Leitner P, Fritz T, Gall HC (2015) 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

CloudScale-Project (2014) Cloudstore. https://github.com/cloudscale-project/cloudstore. Accessed: 2015-06-01

Cohen I, Goldszmidt M, Kelly T, Symons J, Chase JS (2004) Correlating instrumentation data to system states: a building block for automated diagnosis and control. In: Proceedings of the 6th conference on symposium on operating systems design & implementation, OSDI'04, vol 6, pp 16–16

Cohen I, Zhang S, Goldszmidt M, Symons J, Kelly T, Fox A (2005) Capturing, indexing, clustering, and retrieving system history. In: Proceedings of the twentieth ACM symposium on operating systems principles, SOSP '05, pp 105–118

Costantini D (2015) How to configure a pass-through disk with hyper-v. http://thesolving.com/virtualization/how-to-configure-a-pass-through-disk-with-hyper-v/. Accessed: 2017-04-04

Dean J, Barroso LA (2013) The tail at scale. Commun ACM 56:74–80

Dee (2014) 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. Accessed: 2017-04-04

Eeton K (2012) 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

Foo KC, Jiang ZM, Adams B, Hassan AE, Zou Y, Flora P (2010) Mining performance regression testing repositories for automated performance analysis. In: 10th international conference on quality software (QSIC), 2010, pp 32–41

Freedman D (2009) Statistical models: theory and practice. Cambridge University Press

Harrell FE (2001) Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis. Springer

Heger C, Happe J, Farahbod R (2013) 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

Huber N, von Quast M, Hauck M, Kounev S (2011) 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

Jaffe D, Muirhead T (2011) Dell dvd store. http://linux.dell.com/dvdstore/. Accessed: 2015-06-01

Jain R (1990) The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling. Wiley

Jiang M, Munawar M, Reidemeister T, Ward P (2009a) 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

Jiang M, Munawar MA, Reidemeister T, Ward PA (2009b) System monitoring with metric-correlation models: problems and solutions. In: Proceedings of the 6th international conference on autonomic computing, pp 13–22

Jiang ZM, Hassan AE, Hamann G, Flora P (2009) Automated performance analysis of load tests. In: IEEE International conference on software maintenance, 2009. ICSM 2009, pp 125–134

Jin G, Song L, Shi X, Scherpelz J, Lu S (2012) Understanding and detecting real-world performance bugs. In: Proceedings of the 33rd ACM SIGPLAN conference on programming language design and implementation, PLDI '12. ACM, pp 77–88

Kabacoff RI (2011) R in action. In: R in action. Manning Publications Co., Staten Island, NY, pp 207–213

Kearon S (2012) 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. Accessed: 2017-04-04

Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th international joint conference on artificial intelligence, IJCAI'95, vol 2, pp 1137–1143

Kraft S, Casale G, Krishnamurthy D, Greer D, Kilpatrick P (2011) Io performance prediction in consolidated virtualized environments. SIGSOFT Softw Eng Notes 36(5):295–306

Kuhn M (2008) Building predictive models in r using the caret package. J Stat Softw Articles 28(5):1–26

Leitner P, Cito J (2016) 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

Luo Q, Poshyvanyk D, Grechanik M (2016) 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

Malik H, Adams B, Hassan AE (2010a) 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

Malik H, Jiang ZM, Adams B, Hassan AE, Flora P, Hamann G (2010b) 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

Malik H, Jiang ZM, Adams B, Hassan AE, Flora P, Hamann G (2010c) 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

Malik H, Hemmati H, Hassan AE (2013) 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

Mcintosh S, Kamei Y, Adams B, Hassan AE (2016) An empirical study of the impact of modern code review practices on software quality. Empirical Softw Engg 21(5):2146–2189

Menon A, Santos JR, Turner Y, Janakiraman GJ, Zwaenepoel W (2005) 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

Merrill CL (2009) Load testing sugarcrm in a virtual machine. http://www.webperformance.com/library/reports/virtualization2/. Accessed: 2017-04-04

Microsoft Technet (2007) Windows performance counters. https://technet.microsoft.com/en-us/library/cc780836(v=ws.10).aspx. Accessed: 2015-06-01

Netto MA, Menon S, Vieira HV, Costa LT, De Oliveira FM, Saad R, Zorzo A (2011) Evaluating load generation in virtualized environments for software performance testing. In: IEEE International symposium on parallel and distributed processing workshops and phd forum (IPDPSW), 2011. IEEE, pp 993–1000

Nguyen TH, Adams B, Jiang ZM, Hassan AE, Nasser M, Flora P (2012) 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

Nistor A, Jiang T, Tan L (2013a) Discovering, reporting, and fixing performance bugs. In: 2013 10th working conference on mining software repositories (MSR), pp 237–246

Nistor A, Song L, Marinov D, Lu S (2013b) Toddler: detecting performance problems via similar memory-access patterns. In: Proceedings of the 2013 international conference on software engineering, ICSE '13. IEEE Press, Piscataway, NJ, USA, pp 562–571

NIST/SEMATECH (2003) E-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm. Accessed: 2015-06-01

Oracle (1998) MYSQL Server 5.6. https://www.mysql.com/. Accessed: 2015-06-01

Refaeilzadeh P, Tang L, Liu H (2009) Cross-validation . In: Encyclopedia of database systems. MA, Boston, pp 532–538

Rodola G (2009) Psutil. https://github.com/giampaolo/psutil. Accessed: 2015-06-01

Shang W, Hassan AE, Nasser M, Flora P (2015) 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

Shewhart WA (1931) Economic control of quality of manufactured product, vol 509. ASQ Quality Press

Srion E (2015) The time for hyper-v pass-through disks has passed. http://www.altaro.com/hyper-v/hyper-v-pass-through-disks/. Accessed: 2017-04-04

Stapleton JH (2008) Models for probability and statistical inference: theory and applications. Wiley

SugarCRM (2017) Sugarcrm. https://www.sugarcrm.com/. Accessed: 2017-04-04

Syer MD, Jiang ZM, Nagappan M, Hassan AE, Nasser M, Flora P (2013) 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

Syer MD, Shang W, Jiang ZM, Hassan AE (2017) Continuous validation of performance test workloads. Autom Softw Eng 24(1):189–231

Tintin (2011) 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. Accessed: 2017-04-04

TPC (2001) TPC-W. www.tpc.org/tpcw. Accessed: 2015-06-01

Tsakiltsidis S, Miranskyy A, Mazzawi E (2016) On automatic detection of performance bugs. In: 2016 IEEE international symposium on software reliability engineering workshops (ISSREW), pp 132–139

Tyson J (2001) How network address translation works. http://computer.howstuffworks.com/nat2.htm. Accessed: 2017-04-04

VMWare (2016) Accelerate software development and testing with the vmware virtualization platform. http://www.vmware.com/pdf/development_testing.pdf. Accessed: 2016-03-16

Walker HM (1929) Studies in the history of statistical method: with special reference to certain educational problems. Williams & Wilkins Co

Woodside M, Franks G, Petriu DC (2007) The future of software performance engineering. In: Future of software engineering, 2007, pp 171–187

Xiong P, Pu C, Zhu X, Griffith R (2013) 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

Zaman S, Adams B, Hassan AE (2012) A qualitative study on performance bugs. In: 2012 9th IEEE working conference on mining software repositories (MSR), pp 199–208

Muhammad Moiz Arif received his M.Sc. from Concordia University (Canada) in the domain of software engineering. He now works in the industry with his expertise in software performance analysis. His research interests include software performance engineering, big data software engineering, software engineering for ultra-large-scale systems, and empirical software engineering. Contact him at mo_ari@encs.concordia.ca; m.moizarif@gmail.com; http://das.encs.concordia.ca/members/moizarif/.
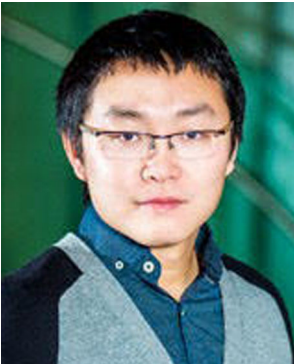
Weiyi Shang is an Assistant Professor and Concordia University Research Chair in Ultra-large-scale Systems at the Department of Computer Science and Software Engineering at Concordia University, Montreal. He has received his Ph.D. and M.Sc. degrees from Queen's University (Canada) and he obtained B.Eng. from Harbin Institute of Technology. His research interests include big data software engineering, software engineering for ultra-large-scale systems, software log mining, empirical software engineering, and software performance engineering. His work has been published at premier venues such as ICSE, FSE, ASE, ICSME, MSR and WCRE, as well as in major journals such as TSE, EMSE, JSS, JSEP and SCP. His work has won premium awards, such as SIGSOFT Distinguished paper award at ICSE 2013 and best paper award at WCRE 2011. His industrial experience includes helping improve quality and performance of ultra-large-scale systems in BlackBerry. Early tools and techniques developed by him are already integrated into products used by millions of users worldwide. Contact him at shang@encs.concordia.ca; http://users.encs.concordia.ca/~shang.

Emad Shihab is an Associate Professor in the Department of Computer Science and Software Engineering at Concordia University. He received his PhD from Queens University. Dr. Shihab's research interests are in Software Quality Assurance, Mining Software Repositories, Technical Debt, Mobile Applications and Software Architecture. He worked as a software research intern at Research in Motion in Waterloo, Ontario and Microsoft Research in Redmond, Washington. Dr. Shihab is a senior member of the IEEE and member of the ACM. More information can be found at http://das.encs.concordia.ca.