AN EMPIRICAL STUDY ON THE DISCREPANCY BETWEEN PERFORMANCE TESTING RESULTS FROM VIRTUAL AND PHYSICAL ENVIRONMENTS

Muhammad Moiz Arif

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science in Software Engineering

Concordia University

Montréal, Québec, Canada

August 2017

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Muhammad Moiz Arif**

Entitled: **An Empirical Study on the Discrepancy between Performance Testing Results from Virtual and Physical Environments**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Software Engineering

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. R. Jayakumar _____ Chair

Dr. T. Eavis _____ Examiner

Dr. J. Rilling _____ Examiner

Dr. E. Shihab _____ Supervisor

Dr. W. Shang _____ Co-supervisor

Approved _____

Chair of Department or Graduate Program Director

_____ 20 _____ _____

Dean

Faculty of Engineering and Computer Science

Abstract

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

Muhammad Moiz Arif

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 in order to exercise the system extensively and build confidence on results. Making it worse, the ever evolving field environments require 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.

In this thesis, we 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 usage from virtual environment vs. CPU usage from physical environment), 2) the relationship between two performance metrics (e.g. correlation between CPU usage and I/O traffic) and

3) statistical 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 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 and 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 may 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 a straightforward performance overhead. Instead, practitioners and future research should investigate leveraging normalization techniques to reduce the discrepancy before examining performance testing results from virtual and physical environments.

Acknowledgments

I would like to show my gratitude towards the people whose support and encouragement proved to be colossal throughout this journey. However, first, I thank Allah, for it is He who provided me with wisdom, health, capabilities, and desire to achieve my goals.

My sincerest and deepest gratitude goes to my academic supervisors, Dr. Emad Shihab and Dr. Weiyi Shang. From the time of of accepting me as a student to the never giving up attitude till the end, I thank you. All my work, results, and the learning experiences would have been meaningless if it was not for your guidance, mentorship and undying support. I consider myself very lucky to have had the opportunity to learn from you.

I would like to acknowledge my thesis committee, Dr. Juergen Rilling and Dr. Todd Eavis for taking the time out to read my thesis and providing valuable and constructive feedback. I also thank the Department of Engineering and Computer Science at Concordia University, for making my graduate experience immaculate.

I am grateful to Dr. Thanh Nguyen with whom I worked under at Blackberry. His work remains the core of my thesis. I am indebted for the opportunity, exposures and practical suggestions provided.

I owe a debt of gratitude to all my fellow colleagues, Rabe Abdalkareem, Davood Mazinanian, Maaz Hafeez Ur Rehman, Shahriar Rostami, Maxime Lamothe, Guilherme Padua, Jinfu Chen, Kundi Yao, Suhaib Mujahid, Giancarlo Sierra, & Olivier Nourry. It was a pleasure to share the same labs as you guys. I wish you all immense success in your journeys.

It is a pleasure to thank my friends who inspired and supported me tirelessly through thick and thin. A especial thanks to Everton da Silva Maldonado, Ahmed Sukhera, Syed Ali Sattam, Mehran Hassani, & Ahmad Al-Sheikh Hassan. You guys always fueled me with utmost motivation. I am proud to call you my closest friends.

Finally, to my parents, I do not think I can thank you enough for all that you have done for me. I thank my sisters, Nada, Aymen, and especially Beenish. I am grateful for the unfailing support and the continuous encouragement throughout the process of researching and completing my graduate studies.

Dedication

To my parents.

Related Publications

The following publications are related to this thesis:

1. **Muhammad Moiz Arif**, Weiyi Shang, & Emad Shihab. Empirical Study on the Discrepancy between Performance Testing Results from Virtual and Physical Environments. In *Empirical Software Engineering Journal (EMSE)*, 2017. [**Major Revision**]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 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 [WFP07]. Often however, failures in large software systems are due to performance issues rather than functional bugs [DB13, FJA+10]. Such failures lead to the eventual decline in quality of the system with reputational and monetary consequences [Tec]. For instance, Amazon estimates that a one-second page-load slowdown can cost up to $1.6 billion [Eet].

In order to mitigate performance issues and ensure software reliability, practitioners often conduct performance tests [WFP07]. Performance tests apply a workload (e.g., mimicking users' behavior in the field) on the software system [Jai90, SSJH16], 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 [SJN+13] and throughput bottlenecks [MAH10]).

Since performance tests are often performed on large-scale software systems, the

performance tests often require many resources [Jai90]. Moreover, performance tests often need to run for a long period of time in order to build statistical confidence on the results [Jai90]. Such testing environments need to be easily configurable such that a specific environment can be mimicked, reducing false performance issues, e.g. issues that are related to the environment. Hence, due to their flexibility, virtual environments (VMs) enable practitioners to easily prepare, customize, use and update performance testing environments in an efficient manner. Therefore, to address such challenges, virtual environments are often leveraged for performance testing [CN01, VMW]. The use of VMs in performance testing are widely discussed [Dee14, Kea12, Tin11], and even well documented [Mer09] by practitioners. In addition, many software systems are released both on-premise (physical) and on cloud (virtual) environment (e.g., SugarCRM [Sug17] and BlackBerry Enterprise Server [Bla14]). 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 [CLFG15, NAJ+12, XPZG13]. Studies have investigated the overhead that is associated with virtual environments [MST+05] and concluded that the computational overhead may effect the system performance in a virtual environment. 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.

We perform a study on two open-source systems, Dell DVD Store (DS2) [DJ] and CloudStore [Clo], and the performance tests are conducted on 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.

- *single performance metric*: the trends and distributions of each performance metric. Such analysis is to identify the trends and shape of the distributions of performance metrics. 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. We investigate such distributions and trends.

- *the relationship between the performance metrics*: the correlations between every two performance metrics. Combinations of performance metrics are significantly more predictive towards performance issues than individual metrics. We believe that a change in these combinations of relationships can reflect the discrepancy of performance in the two environments.

- *statistical performance models*: the models that are built using performance metrics to predict the overall performance of the system. Our third type of analysis is used to see the combination of all performance metrics all together

by constructing a statistical model.

The goal of the thesis is to empirically demonstrate the evidence of discrepancy present between the two environments with the data generated by our performance testing results.

1.2 Research Hypothesis

Prior research and our industrial experience lead us to an investigation based on the following hypothesis:

> *For large-scale software systems, performance assurance activities are carried primarily in virtual environments. There is little research on the presence of performance discrepancy in a virtual environment compared to physical. We hypothesize that for software testing activities there exists a discrepancy between physical and virtual environments.*
>
> *Furthermore, we believe that the practitioners should be aware and reduce such existing discrepancies when analyzing software performance in a foreign environment.*

1.3 Thesis Overview

Chapter 2: Literature Review: In this chapter, we discuss the background of performance assurance activities. We also discuss performance testing carried out in virtual environments and the associated overhead caused by it. The chapter is divided in two parts:

Part I: Discusses the role of software performance testing and how a software performance test is carried out, from setting up the system to the point of data

collection and analysis. Furthermore, this part also addresses the differences and similarities between software performance testing, load testing and stress testing.

Part II: Addresses the methodologies that are used to analyze performance testing results. We present the work sub divided into three categories: single performance metric, relationship among performance metrics, statistical modeling based on performance metrics. This part also addresses the work that points to the overheads present in virtual environments. We present the state-of-the-art approaches used to analyze performance testing results and how such approaches help us in detection of overheads. Additionally, we also discuss the role of performance testing in bug detection.

From the literature review, we deduced that much prior work has been affiliated to the generation of performance alarms and the detection of performance issues however little work has highlighted the testing done in virtual environments.

Chapter 3: Studying The Presence Of Discrepancy: In this chapter, we perform case studies on two open source subject systems. We generate the performance metrics by applying realistic and varying workloads in identically set up physical and virtual environments. Next, it is followed up by data cleansing. Finally, we analyze the performance metrics in three possible ways: individually, as a collection, and as an input to statistical models.

Chapter 4: Discussion On The Impact From Other Factors: Furthermore, we elaborate and solidify the conclusion drawn by discussing possibilities on how the use of virtual environments may affect our results: by changing the type of the virtual machine, by modifying the allocated resources and testing the repeatability(hence the stability) of our chosen virtual environment. Lastly, we discuss the threats to validity for this thesis.

Chapter 5: Summary, Contributions And Future Work: We conclude the thesis by summarizing our work. We also highlight how our findings can be leveraged by the practitioners, serving as the stepping stone towards future work.

1.4 Thesis Contributions

The major contributions of thesis are as follows:

- An extensive review of the state of the art analysis in software performance activities. Such a review is necessary for the practitioners to be aware of the discrepancies associated to software performance across different environments.
- We find the performance metrics have different shapes of distributions and trends in virtual environments compared to physical environments. There are large differences in correlations among performance metrics measured in virtual and physical environments.
- We highlight statistical models using performance metrics from virtual environments do not apply to physical environments (i.e., produce high prediction error) and vice versa.
- We examined the feasibility of using normalization techniques 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.

Chapter 2

Background And Literature Review

Software systems are expected to serve millions of concurrent requests [AJ00]. However, the systems are first tested to ensure that they are working correctly under a certain load(s). This load is also known as the rate at which the system is processing the requests. In this chapter, we discuss the motivation behind our work and similar studies in the field of performance engineering. We later survey the state-of-the-art literature that is related to our work.

2.1 Background

Generally, performance assurance activities are carried out by analyzing the system responses on a variety of workloads. For example, to detect performance issues, the system performance is analyzed after applying a workload. This workload profile depicts the normal workload of the system once it is functional in the field [AW95].

2.1.1 Performance Testing

Performance testing is mainly focused to detect primarily performance related problems with the software system, e.g. response time, throughput and resource utilization [BLG11a, BLG11b, Gor00].

The goal behind performance testing may be to test performance requirements [PG11] or exploratory (e.g. to answer questions such as how do various configurations impact the performance of the system [MV00, MAD94, MA01, PG11]).

What is a test execution?

As shown in Figure 1, the life cycle of a typical performance test execution is made up of four phases:

- *Setup*, that is system and test execution setup.
- *Testing*, that is the actual phase where the test is run and terminated at the end of the time frame.
- *Data and Metrics Collection*, that is the phase where the performance metrics and executions logs are collected [JH15].
- *Data Analysis*, the data is further analyzed and refined to draw conclusions in the fourth and final phase (discussed in Section 2.2).

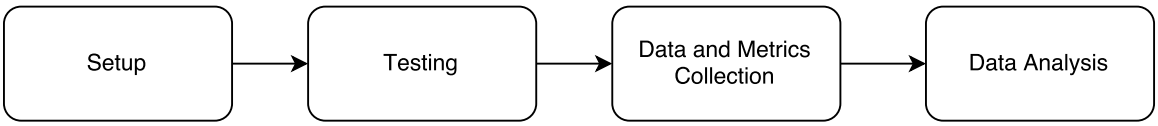


Figure 1: Test execution phases

Before the phase of text execution, practitioners have to develop the test for the system. It is based on either the realistic usage of the system when functional or with

the goal to uncover problems [JH15]. Once the test is developed, it is followed by its execution.

Setup

The term *setup* is further divided into two sub-terms. The setup for the system and the setup for the text execution system. The former deals with making sure that the system or *subject under test* (SUT) is operational and fully functional. This may also include setting up servers and other functionalities attached to the system for example, database servers. The latter engulfs configuring and using the load test drivers (for example, WebLoad [Ltd], HP LoadRunner [Dev], Apache Jmeter [Fou]) in addition to setting up the testing environment. In order to record the performance metrics, performance monitoring tools , e.g. Perfmon [Mic], and Psutil [Rod] are set up.

Testing

Following the setup, the system is tested by applying load. The results are concurrently recorded. Practitioners may terminate the load based on the following techniques:

- Continuous: Testing until it is stopped manually by practitioners [Sta06].
- Time-Based: The test runs for a specific duration of time [Sta06].
- Counter-Based: The load is stopped after a certain number of requests sent or received [Sta06].
- Statistic-Based: A comparatively newer technique where the metric of interest is captures till it is statistically stable. This serves a high confidence level when analyzing the metric [MKMS10, SAP11].

Data and Metrics Collection

During the course of loading the system and running the tests, performance data is monitored and recored. The performance data collected is in the form of logs and performance metrics. These performance metrics may be recorded at a high-level (e.g. throughput and response time) or at a low-level (e.g. CPU and memory usage). It is necessary that the monitoring or recording tool does not induce a large overhead on the system. This may lead to biased results [MDHS10].

Data Analysis

After the performance data is collected, various analysis techniques are used to understand the performance of the system. We discuss in detail the approaches to analyze data in Section 2.2.

2.1.2 What are the differences between load testing, stress testing and performance testing?

Although there are similarities between the three types of testing techniques i.e. load, stress and performance testing, (for e.g. all of the three are carried out after functional testing) we now differentiate between the application of the aforementioned tests.

Load Testing

The rate at which the requests are submitted to a system is called the load [Bei84]. This system is more commonly known as system under test or *SUT*. Load testing is carried out later than conventional testing in a software's life. This may be done on a prototype or a working system than a design or a model. Load testing is used to detect workload related problems. For example, deadlocks, buffer overflows, high response times and low throughput [AW95, BLG11a, BLG11b].

In some exceptional cases where the non-functional requirements are not present, one of ways to determine if the test has passed is by comparing it to the results of the previous version. This is also known as the "no-worse-than-before" principle. As the name suggests, the version being tested should be, if not better, equal to the previous version. [DRSS01]

Although there may be some similarities present between performance and load testing, however performance testing is detailed than load testing as it may be used to cover designs [CM99, DPE04, DPE05], algorithms [CCW09, CCW07], and system configurations [HCM05, PG11, SM05].

Stress Testing

Stress testing is testing the application under "stress" or extreme load. This can be used to detect how resilient the system is or to detect further load-related problems. For example, memory leaks and deadlocks. These "stressful" conditions for the system can be either load related (normal [ZC02, KKB11, Cha10] or heavy load [Dil09, KKB11, HMHR01]) or limited resources allocated/failures (for example, disk or database failures) [AK09]. We noted that in some cases it may also be used to detect the competency of the SUT [Gar10, Gar08, GBL06, GBL08].

Having stated all of the above, there are instances where the interest of a performance test may overlap with load testing or stress testing and vice-versa. For example, to check robustness of the system when put in extreme conditions, or to check how an algorithm works when handling large files. We observe that the terms performance testing [Dil09, Men02a, Men02b], load testing [ARW96, BC08, Ghe, MFB$^+$] and stress testing [BC08, YP96, BC06] are also used interchangeably.

We focus on performance testing, which is to detect performance related behavior of our *SUT*.

2.2 Literature Review

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

2.2.1 Analyzing performance testing results

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.

Single performance metric

Nguyen *et al.* [NAJ+12] introduce the concept of using control charts [She31] 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 tests. 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.* [MJA+10b, MHH13] 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.* [HHF13] 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.* [JHHF09] extract response time from system logs. Instead of conducting statistical tests, Jiang *et al.* visualize the trend of response time during performance tests, in order to identify performance issues.

Relationship between performance metrics

Malik *et al.* [MAH10] 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.* [FJA+10] 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.* [JMRW09a] 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 a sign of a performance fault.

Statistical modeling based on performance metrics

Xiong *et al.* [XPZG13] 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.* [SHNF15] 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.* [CGK+04] 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.* [CZG+05] also proposed that system faults can be detected by building statistical models based on performance metrics. The approaches of Cohen *et al.* [CGK+04, CZG+05] were improved by Bodik *et al.* [BGF08] by using logistic regression models.

Jiang *et al.* [JMRW09b] 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 thesis. 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 analyses, in order to see the impact when practitioners use such analyses on performance testing results. Our findings can help better evaluate and understand the findings from the aforementioned research.

2.2.2 Analysis of VM overhead

Kraft *et al.* [KCK+11] 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 disc IO due to increments in time associated with request queues.

Aravind *et al.* [MST+05] 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.* [BGHK13] 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.* [HvQHK11] 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.* [LPG16] converge the set of inputs that may cause software regression. They apply genetic algorithms to detect such combinations. Netto *et al.* [NMV+11] 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 thesis, we evaluate the discrepancy between virtual and physical environments by

focusing on the impact of performance testing results analyses and investigate whether such impact can be reduced in practice.

2.2.3 Performance testing and bug detection

There exists much research on performance testing and bug detection. Nistor *et al.* [NSML13] detect the presence of functional and loop-related performance bugs with the help of their developed tool. Jin *et al.* [JSS+12] 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.* [NJT13] 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.* [TMM16] use prediction models to detect and predict performance bugs based on extractions from source code repositories. Malik *et al.* [MJA+10a] 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.* [ZAH12] 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, generalizing the findings across environments remains an open topic.

In chapter 3, we carry out performance tests in virtual and physical environments and analyze the discrepancy based on the types of analysis we discuss in section 2.2.1.

Chapter 3

Studying The Presence Of Discrepancy

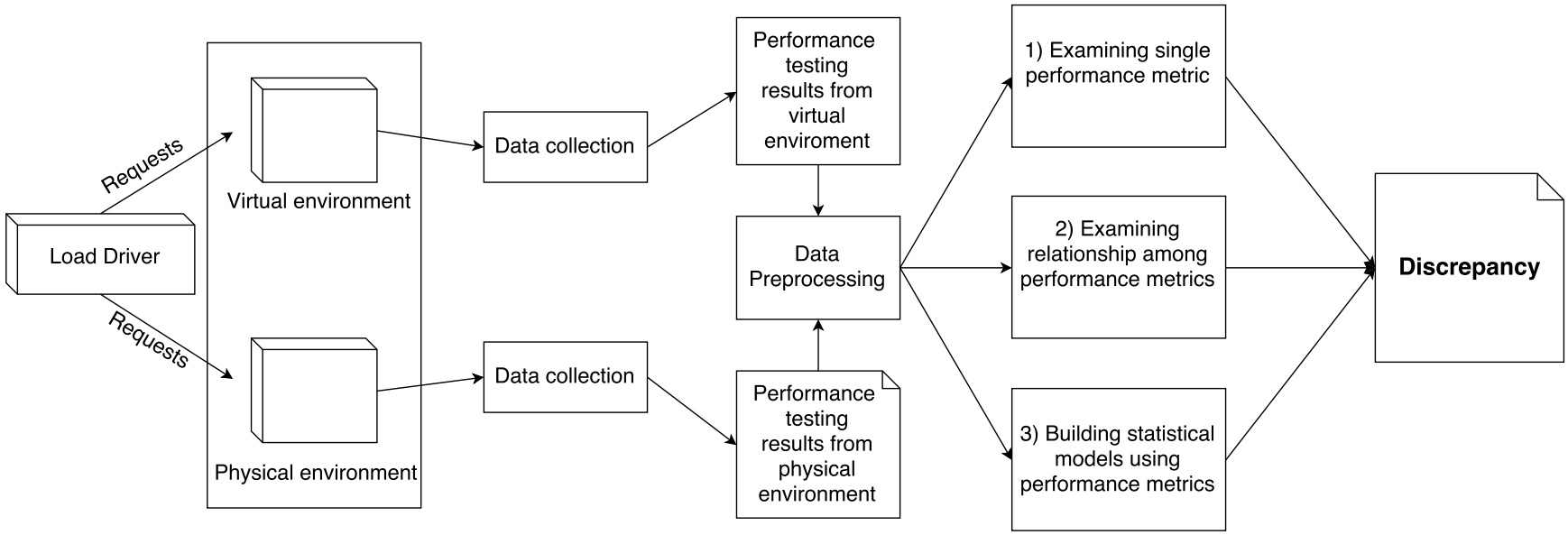


Figure 2: Overview of our case study setup.

3.1 Approach

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 Figure 2.

3.1.1 Subject Systems

Dell DVD Store (DS2) [DJ] is an online multi-tier e-commerce web application that is widely used in performance testing and prior performance engineering research [SHNF15, NAJ+12, JHHF09]. We deploy DS2 (SLOC > 3,200) on an Apache (Version 3.0.0) web application server with MySQL 5.6 database server [Ora]. CloudStore [Clo], our second subject system, is an open source application based on the TPC-W benchmark [TPC]. 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 [ABC+16]. We deploy CloudStore on *Apache Tomcat* [Apa] (version 7.0.65) with MySQL 5.6 database server [Ora].

3.1.2 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 the 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 any 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[Cos15]. 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 [Sri15]. 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[Tys01]. 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.

3.1.3 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 [Fou] 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.

3.1.4 Data collection and preprocessing

Performance metrics. We used *PerfMon* [Mic] 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 [FJA+10].

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 2.2, 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.

3.1.5 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 2.2.1, 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 the distribution and the trend of the metrics should be similar. For example, if in one environment, we observe that Memory has an 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 [NIS] 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 the 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 [Sta08] 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 [Wal29]. 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.

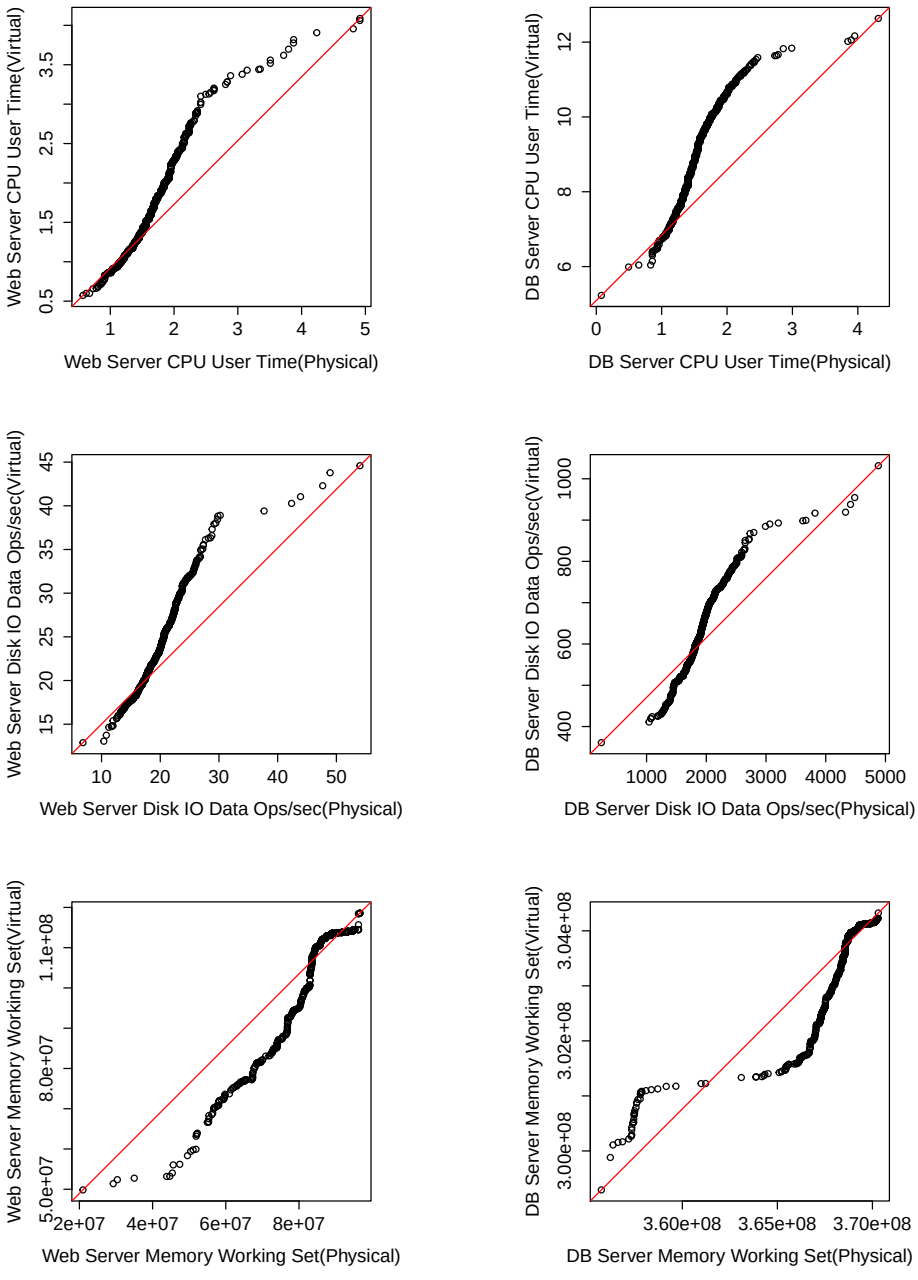
Finally, we calculate Spearman's rank correlation between every performance metric in the virtual environment and the corresponding performance metric in the physical environment, in order to assess whether the same performance metrics in two environments follow the same trend during the test. Intuitively, 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 the distribution in virtual and physical environments. Figure 3 and 4 show the Q-Q plots by comparing the quantiles of performance metrics from virtual and physical

environments. We only present Q-Q plots for CPU user time, IO data operations/sec and memory working set for both application sever and database server. For complete results please refer Chapter 6.

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 the 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 is similar, it is expected that the metrics related to the workload follow the same shape of the 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).



]

Figure 3: Q-Q plots for DS2.

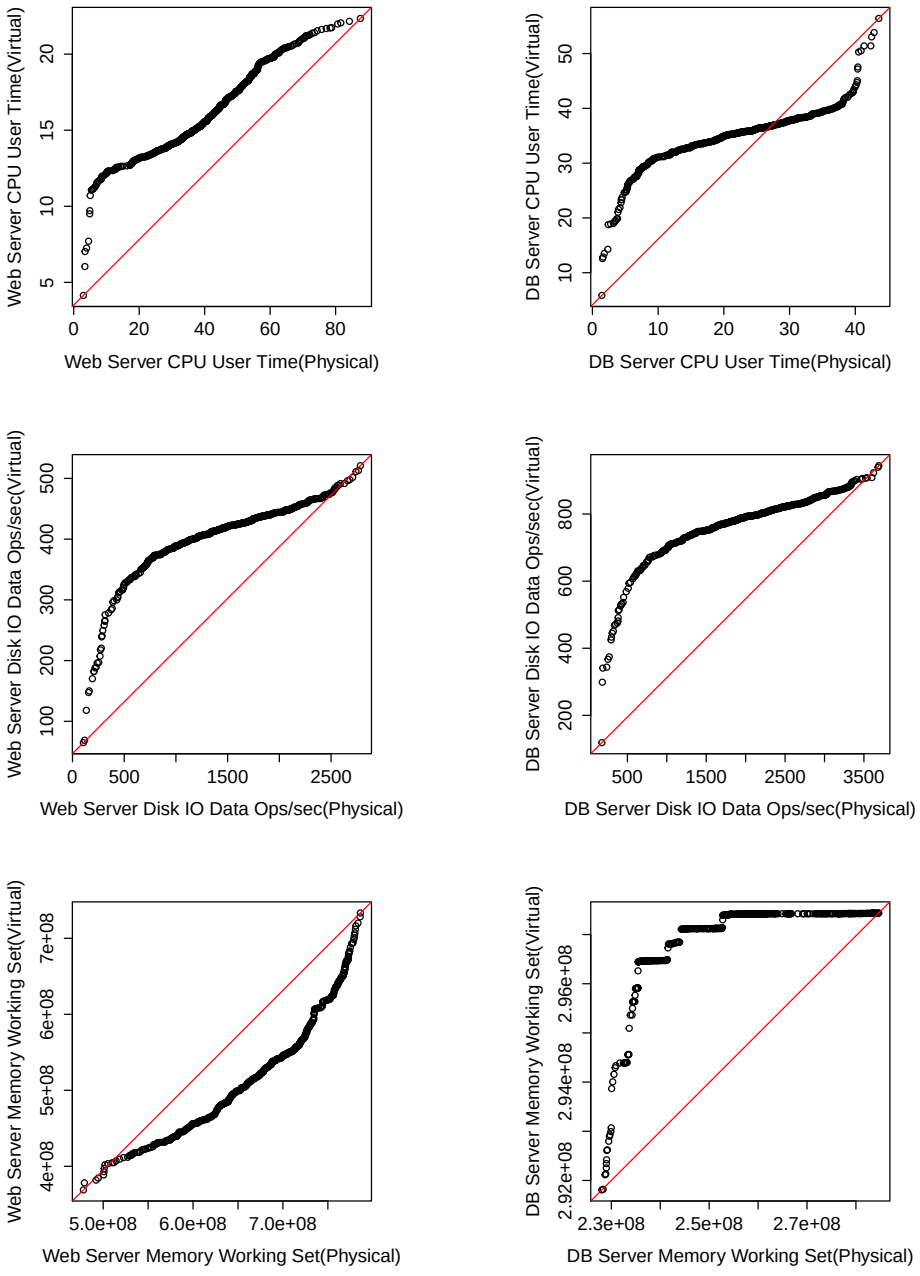


Figure 4: Q-Q plots for CloudStore.

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 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 Figures 3 and 4 many differences between the two distribution are in the lower and higher ends of the plots, which correspond to the high and low values of the metrics. Such values are often treated as outliers. 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.

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

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

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

Three metrics are constant. Therefore, we do not calculate the correlation on those metrics.

3.1.6 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 regression. As found by Cohen *et al.* [CGK+04], combinations of performance metrics are significantly more predictive towards 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 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

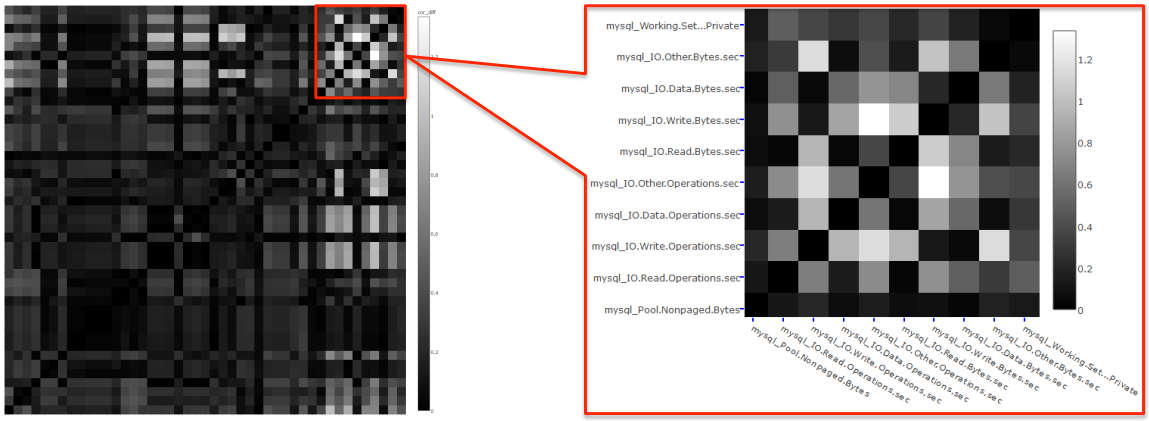


Figure 5: Heatmap of correlation changes for DS2.

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 5 and 6 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 5 and 6. 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 [MST+05, KCK+11]. 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.

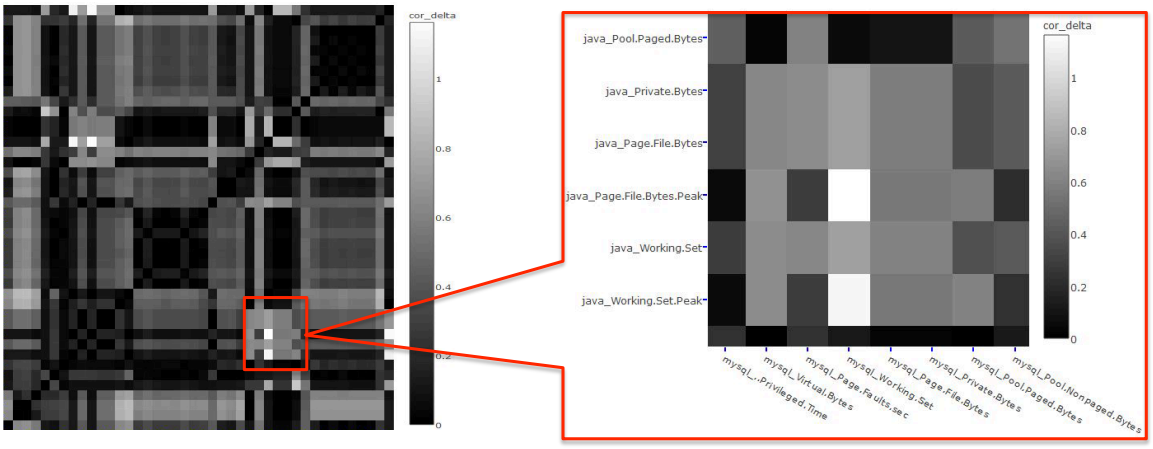


Figure 6: Heatmap of correlation changes for CloudStore.

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

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.

3.1.7 Can statistical performance models be applied across virtual and physical environments?

Motivation. As discussed in the last research question (see Section 3.1.6), the relationship among performance metrics is critical for examining performance testing results (see Section 2.2.1). 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 [XPZG13, CGK+04]. 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

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.

techniques are proposed to examine performance testing results (see Section 2.2.1). 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 [SHNF15, CZG+05, XPZG13]. 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.

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 [Kuh08]. 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 [SSJH16]. 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 [Har01]. 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) [And12]. 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)[SSJH16], 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 [Kab11].

B-2: Building statistical models

In the second step, we build a linear regression model [Fre09] 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 [CZG+05, XPZG13, SHNF15].

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.

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 [CHP90], 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 [Har01]. 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 [MKAH16]. We then rank the independent

variables by their impact on model fit.

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 into 10 partitions. We take one partition (fold) at a time as the test set, and train on the remaining nine partitions [RTL09, Koh95], similar to prior research [MHH13]. 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.

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 Equation 1 from Section 3.1.5 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.* [NAJ+12]. 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.

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 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 [NAJ+12], assumes a linear relationship between the performance metric and the system load. However, such an assumption may not be true in some performance testing results. For example, Table 3 shows that some I/O related metrics do have low correlation with the system load in virtual environments. On the other hand, the normalization based on deviance shows much lower prediction error. We think the reason is that the virtual environments

may introduce metric values with high variance. Normalizing based on the deviance controls such variance, leading to lower prediction errors.

Table 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%

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 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

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

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.

In Chapter 4, we further validate our findings of Chapter 3 by looking at external factors that may have affected the nature of our performance tests and the subsequent analysis.

Chapter 4

Discussion On The Impact From Other Factors

In the previous chapter, 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 hardware resources dedicated to the virtual environments. Therefore, in this section, we examine the impact of such factors to better understand our results and discuss the threats to validity for our findings.

4.1 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 [LC16].

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 3.1.7 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.

4.2 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 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 exists 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

4.3 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 resources 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 (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.

4.4 Threats to Validity

4.4.1 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 [JHHF09, NAJ+12, ABC+16]. 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. We also conduct the experiments only on a Windows OS. This choice was based on monitoring tools and the environments in our labs. We also plan to replicate this study in environments with other OS.

4.4.2 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 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 [Rod], 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 [CZG+05, SHNF15, XPZG13].

4.4.3 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[MJA+10b]. Future work may expand our list of performance metrics to address this threat.

Chapter 5

Summary, Contributions And Future Work

5.1 Summary of the Addressed Topics

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. In this thesis, 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. Following are the summaries of chapters covered in this thesis.

Chapter 2 contains a detailed literature review and examination of state-of-art approaches present for software regression detection and modeling. It is important to include such review as it will help build research gateways while defining the analogies that we have used in this domain.

Chapter 3 discusses the results of our investigation to find discrepancy between virtual and physical environments. In this chapter, we analyze our results, based on the

performance metrics of two open source subject systems (DS2 and CloudStore). Prior studies have also used our chosen subject systems in the field of software performance engineering. We evaluate the aforementioned discrepancy by conducting performance tests on two open source systems 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.

Chapter 4 concludes the work by adding a series of experiments carried out to address if there is a difference in the choice of virtual environments/configurations. Here, we sub divide the experiments into three categories: changing the type of virtual environment, changing the resources allocated to the virtual environment and investigating the stability of our virtual environment by repeating the set of our experiments. We evaluate that our virtual environment is stable. We conclude that altering the external factors has almost insignificant impact on our conclusion in Chapter 3. It reassures that there exists performance discrepancy even between different virtual environments and configurations.

5.2 Contributions

The goal of our thesis is to investigate if there exists a discrepancy between virtual and physical environments. If yes, to what extent it effects the performance assurance activities analyzed and compared in both environments. We reach our conclusions after an analysis on the performance metrics based on: 1) individual performance metrics 2) relationship among performance metrics and 3) statistical models based on the performance metrics.

The major contribution of this work includes:

- **A detailed state-of-the art review of the related literature** We cover in detail the performance assurance activities that are adapted to detect performance discrepancies and anomalies. We chose the most suitable sources to map our work to and and showed that the current approaches do not address the discrepancies present in virtual environments.
- **This is the first research attempt to evaluate the discrepancy between performance testing results in virtual and physical environments.** We show that the current approaches do not consider performance discrepancies present between virtual and physical environments. We provide a detailed analysis
- **Identifying the performance metrics that contribute most to the discrepancies.** We find that relationships among I/O related metrics have large differences between virtual and physical environments. We prove this with the help of analyzing performance metrics as a singular entity and also as an input to regression models.
- **Normalization via deviance** 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.

5.3 Actionable Contributions

- Practitioners should investigate the nature of overhead from VMs before assuming the overhead as straightforward.
- When leveraging existing techniques for performance assurance activities, practitioners should first verify the technique on VMs or in an environment with both

VM and physical machines.

- Practitioners should be careful when using historic data for performance assurance activities in heterogeneous environments. If there exists a discrepancy, normalization via deviance may address the discrepancy.

5.4 Future Work

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. 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.

Reproducing known performance regressions in heterogeneous environments: We conducted a set of experiments in a curated environment where there was no presence of performance regression in our subject systems. We believe that it will be interesting to see if these results still verify if performance regression is injected in our subject systems.

Replicating our experiments in cloud environments: Despite the fact that we carried out our experiments in different types of virtual environments, we also plan to examine the behavior of our subject systems in cloud environments. Issues like noise from other systems and how to isolate and monitor our system with and without regression, which may lead to numerous possibilities.

Designing automating techniques: Lastly, we plan to discover an approach that can lead us on to a precise normalizing factor between different environments. It will be appealing to find out a normalizing technique that incorporates the discrepancies present, dynamically.

Chapter 6

Appendix

Following is the complete set of Q-Q plots and heatmaps for both our subject systems.

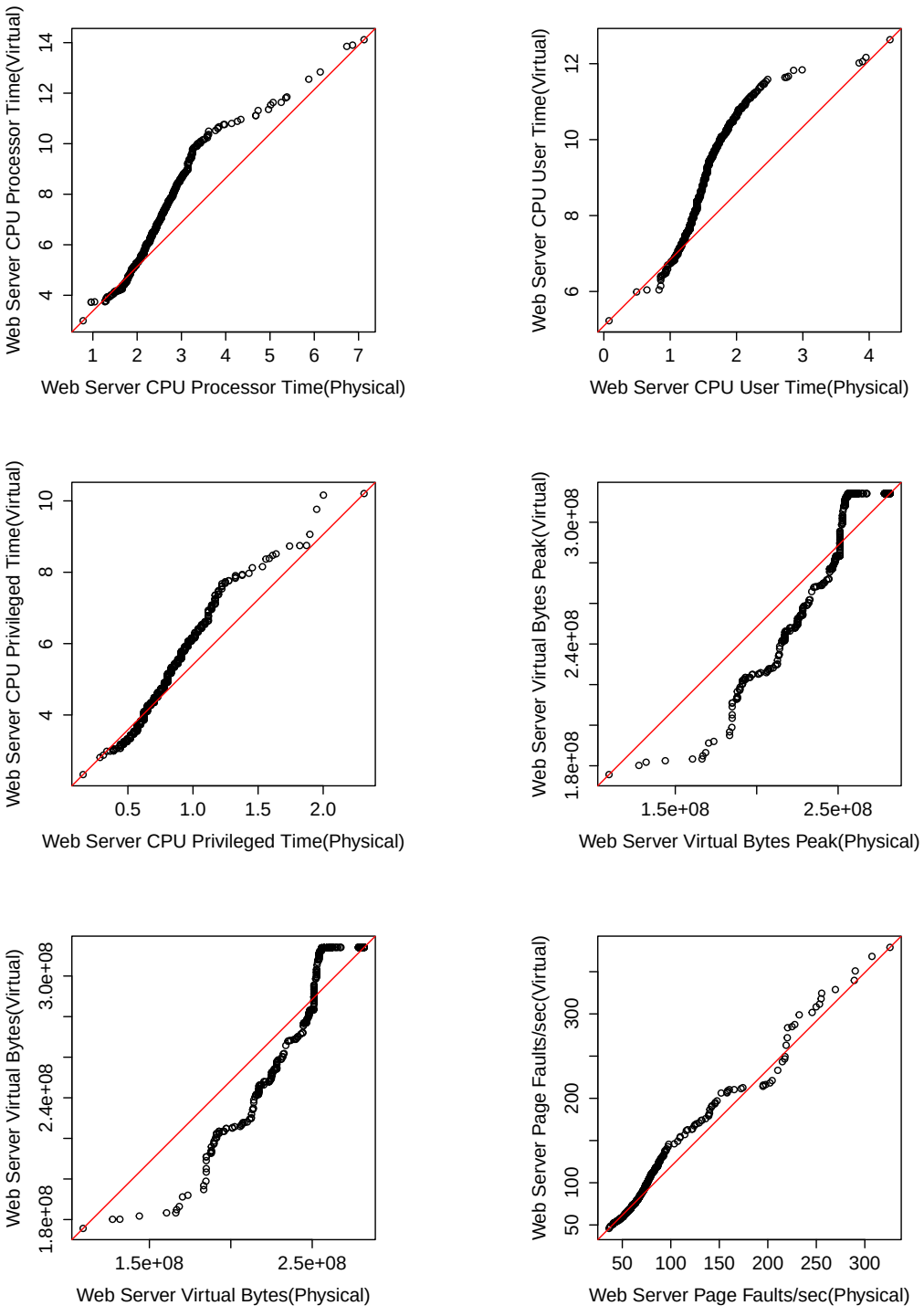


Figure 7: Q-Q plots for DS2's Web Server

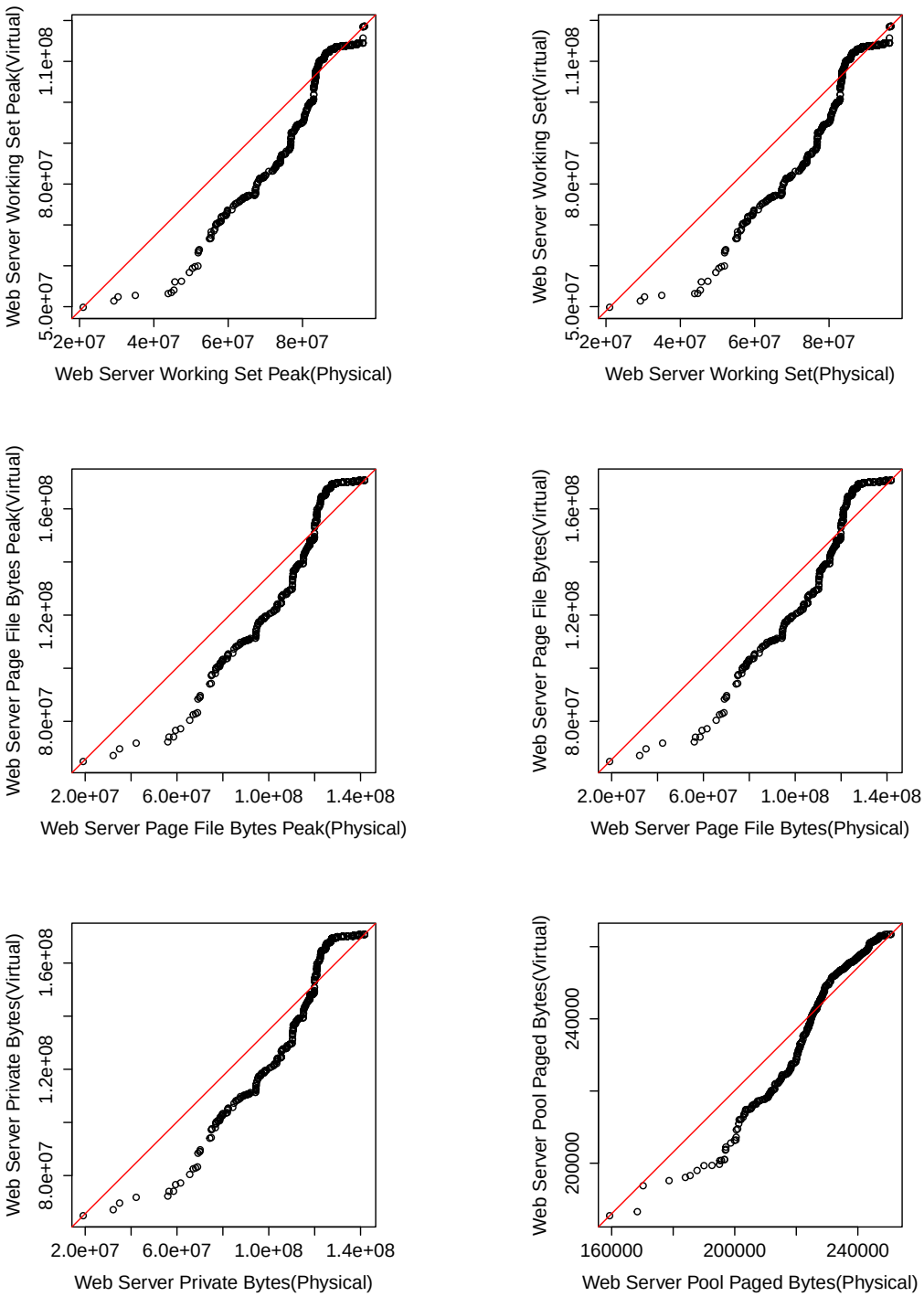


Figure 8: Q-Q plots for DS2's Web Server

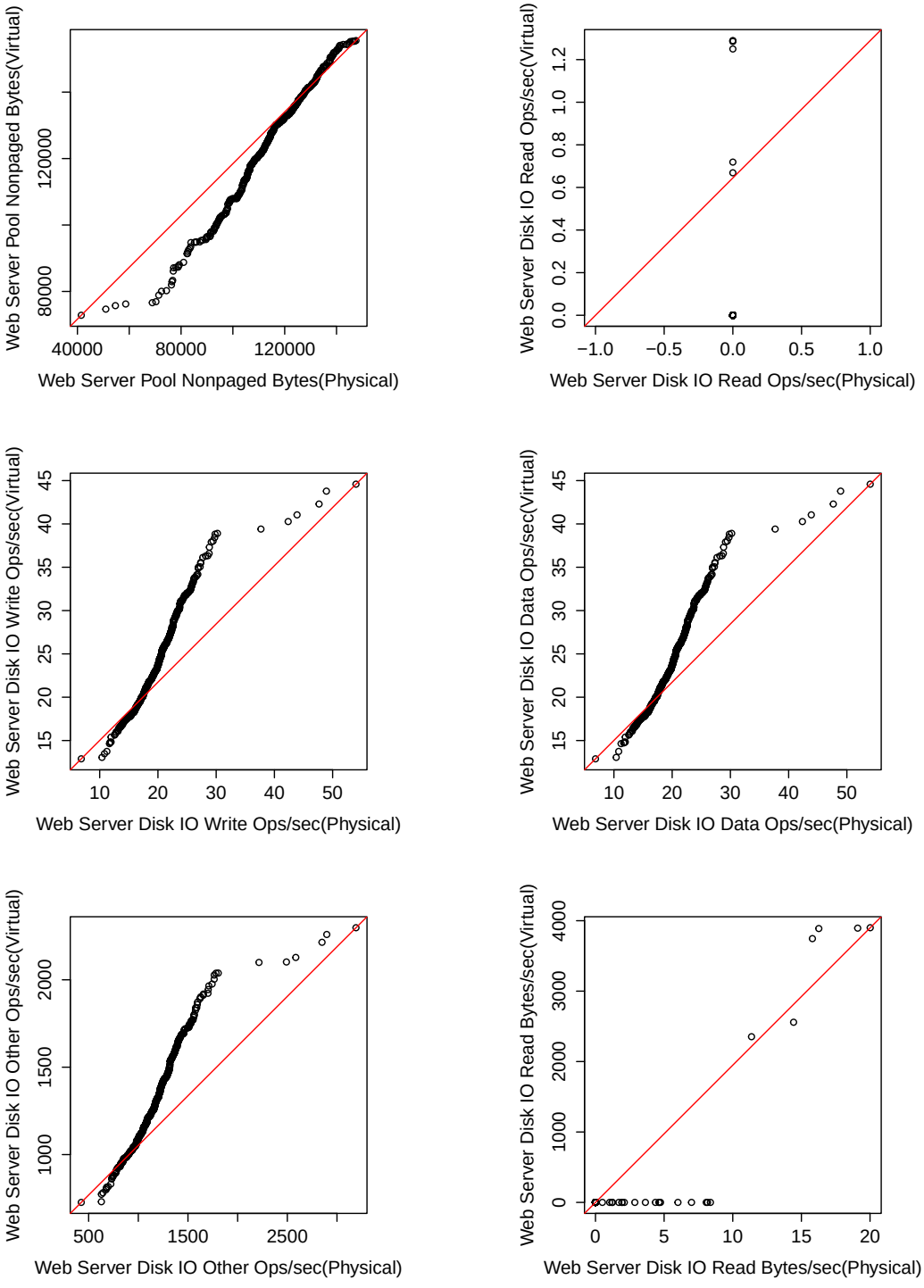


Figure 9: Q-Q plots for DS2's Web Server

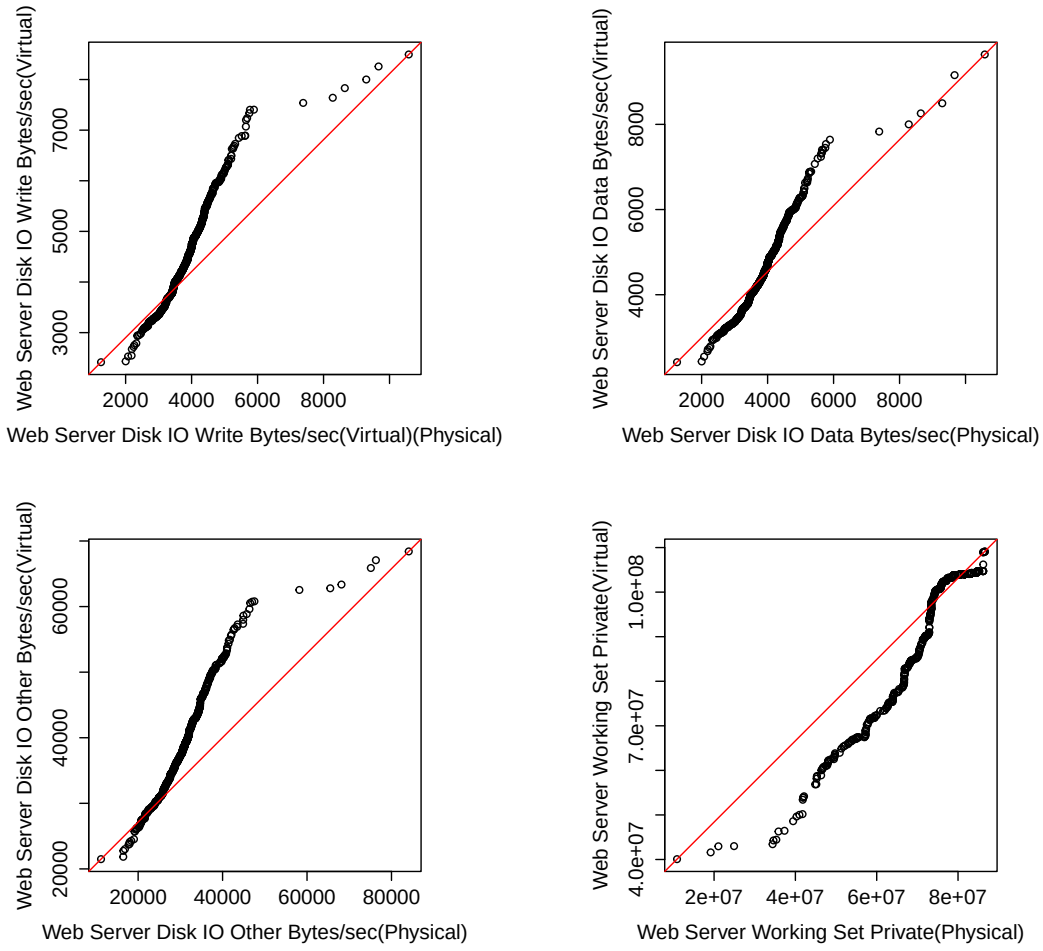


Figure 10: Q-Q plots for DS2's Web Server

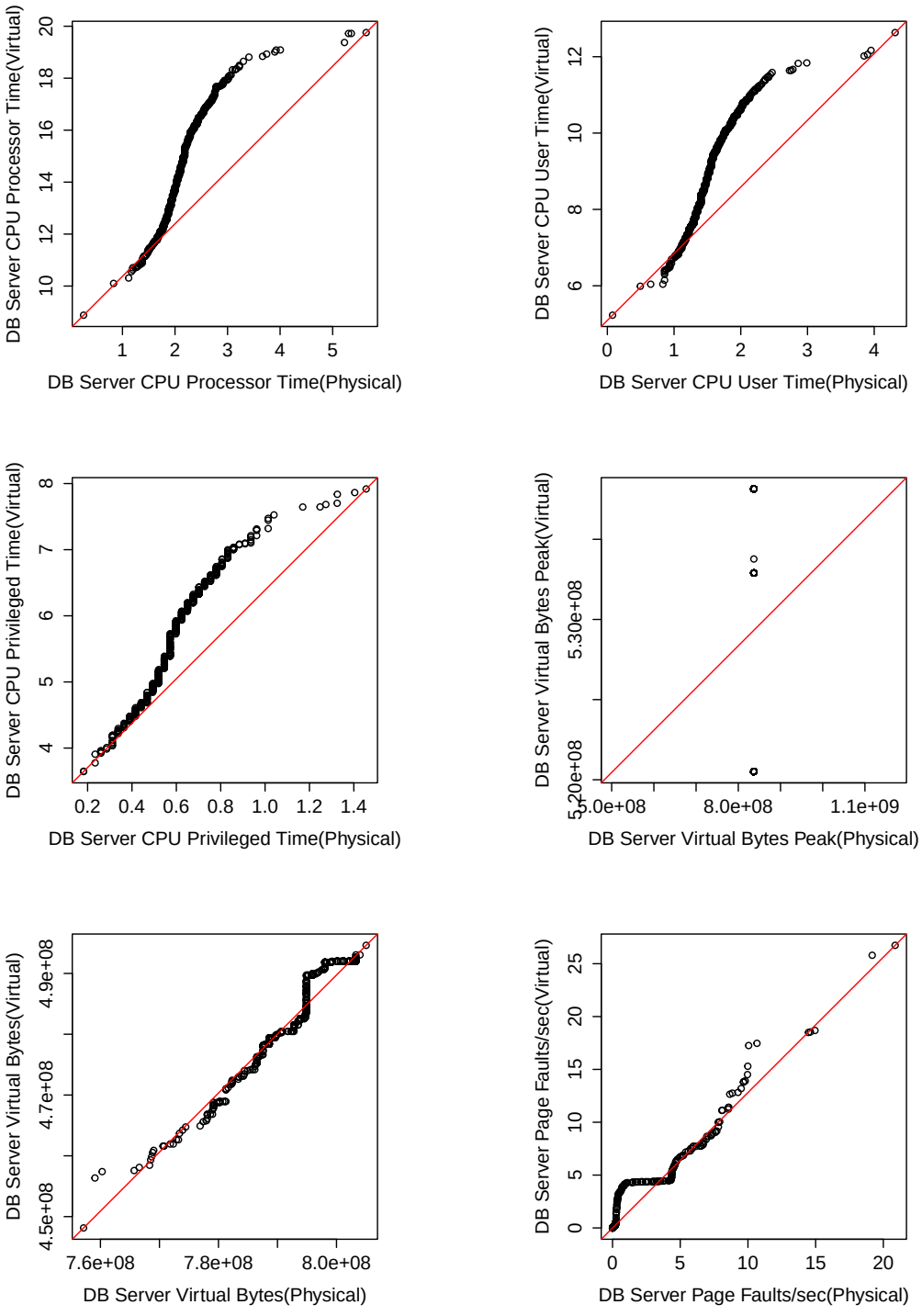


Figure 11: Q-Q plots for DS2's DB Server

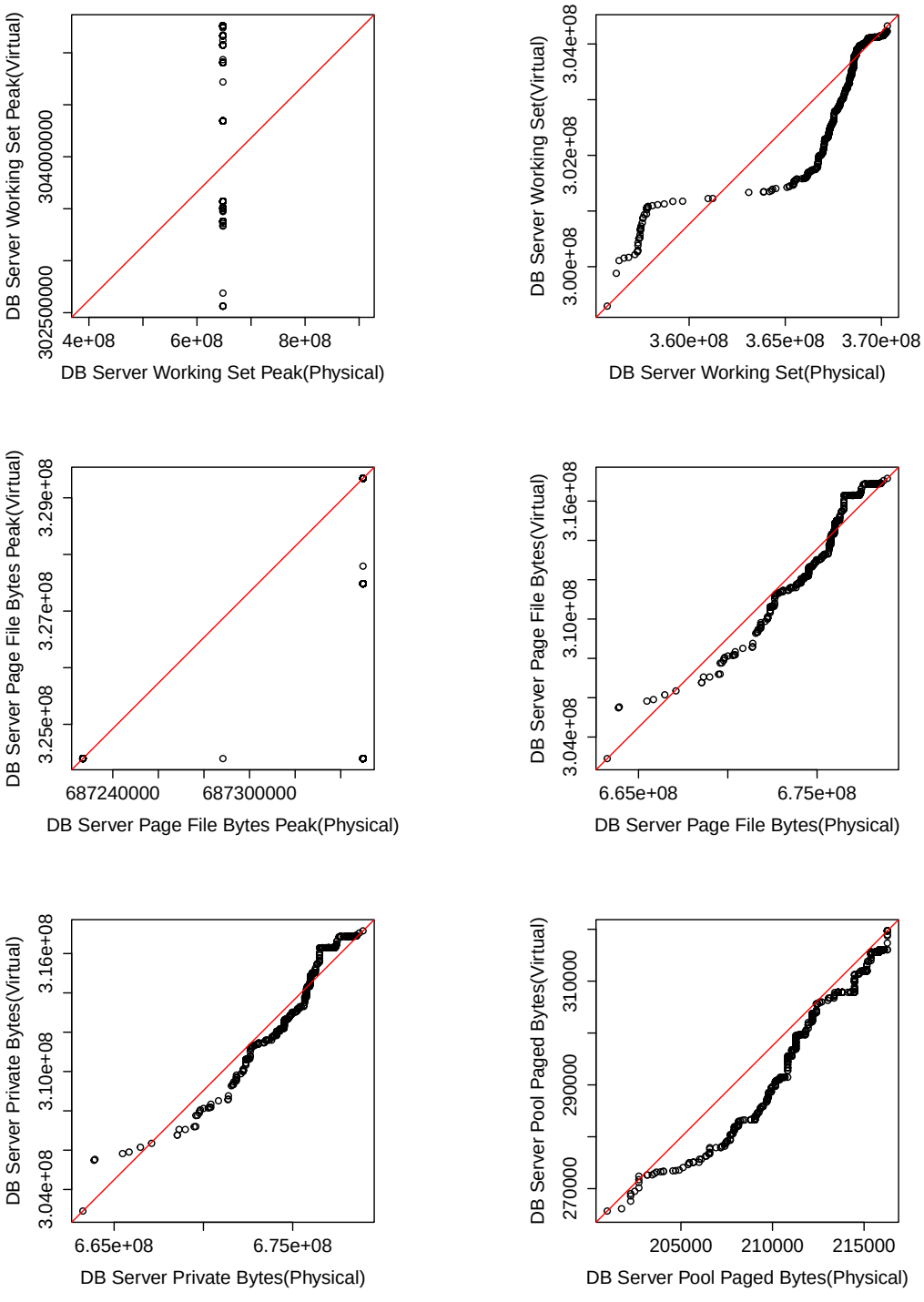


Figure 12: Q-Q plots for DS2's DB Server

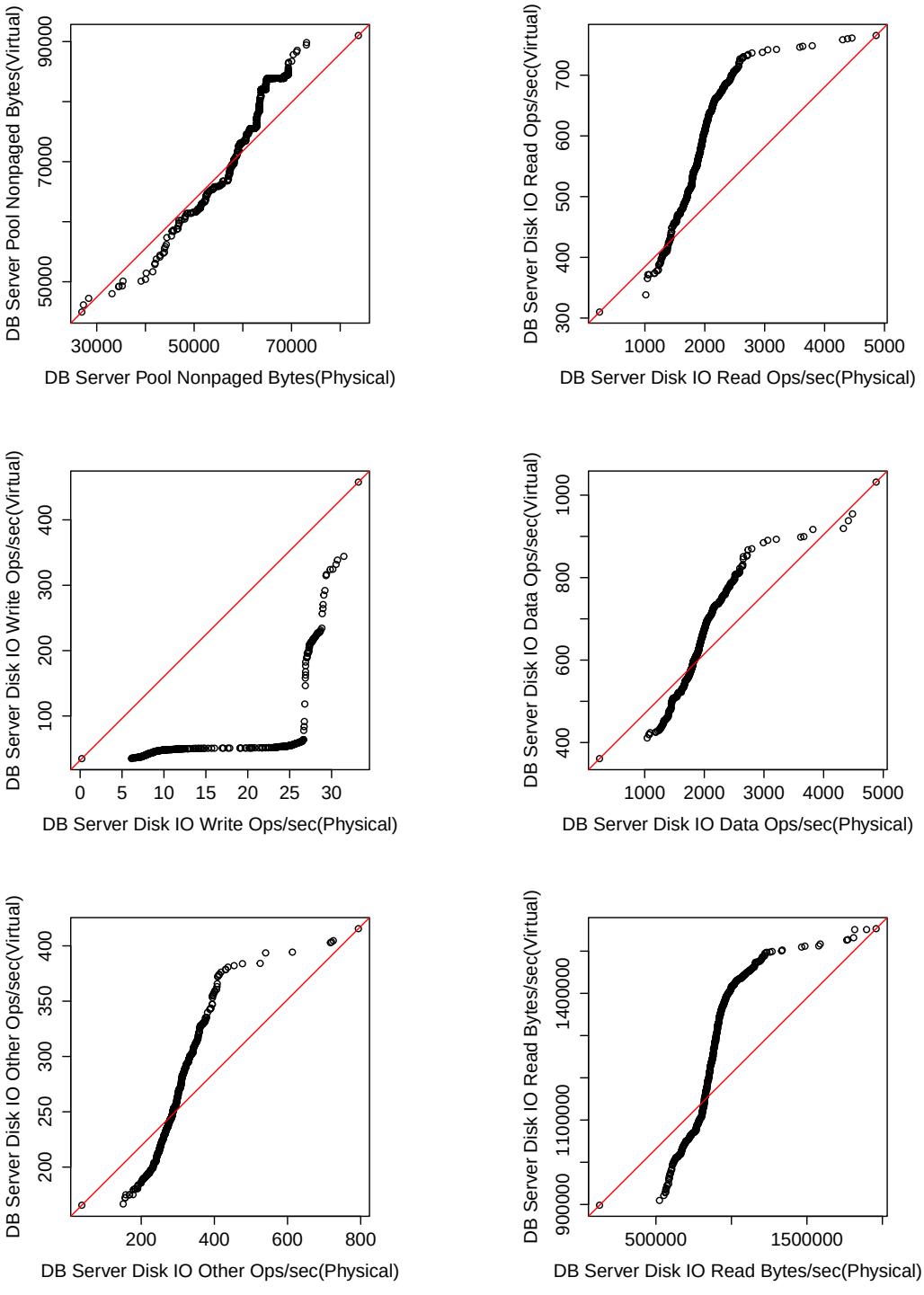


Figure 13: Q-Q plots for DS2's DB Server

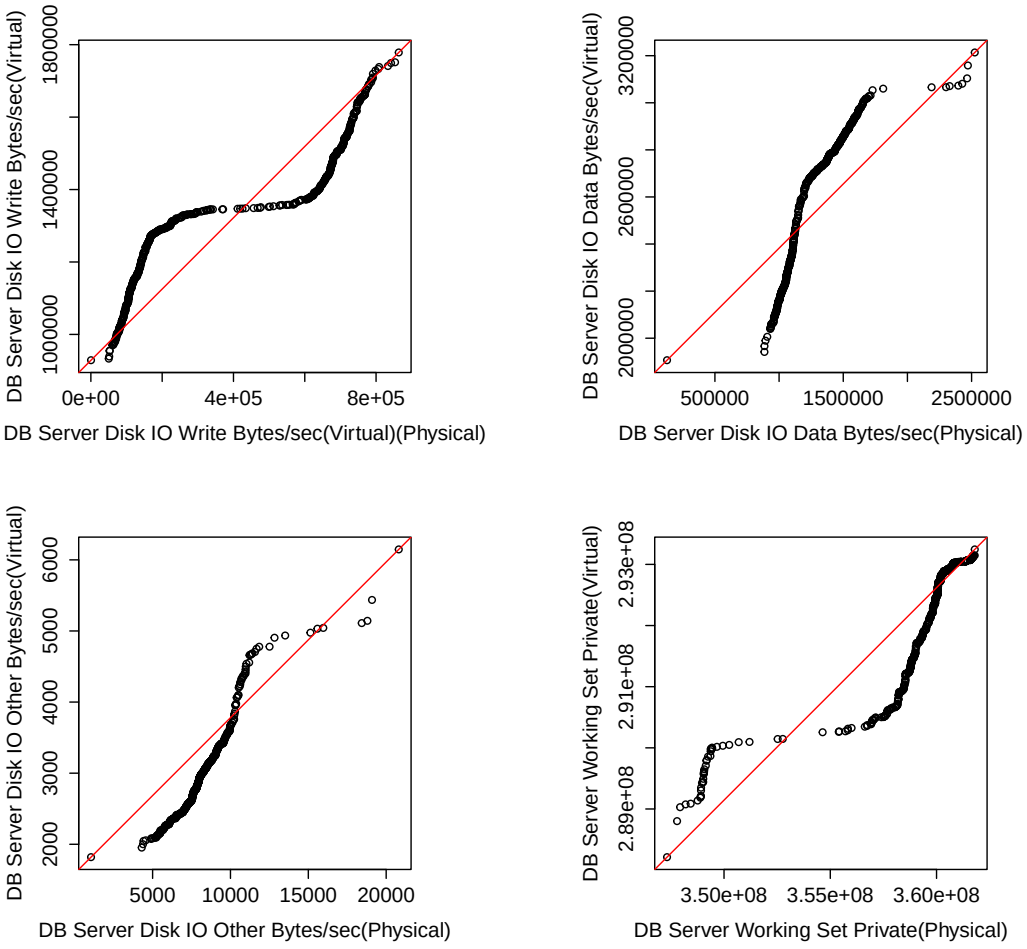


Figure 14: Q-Q plots for DS2's DB Server

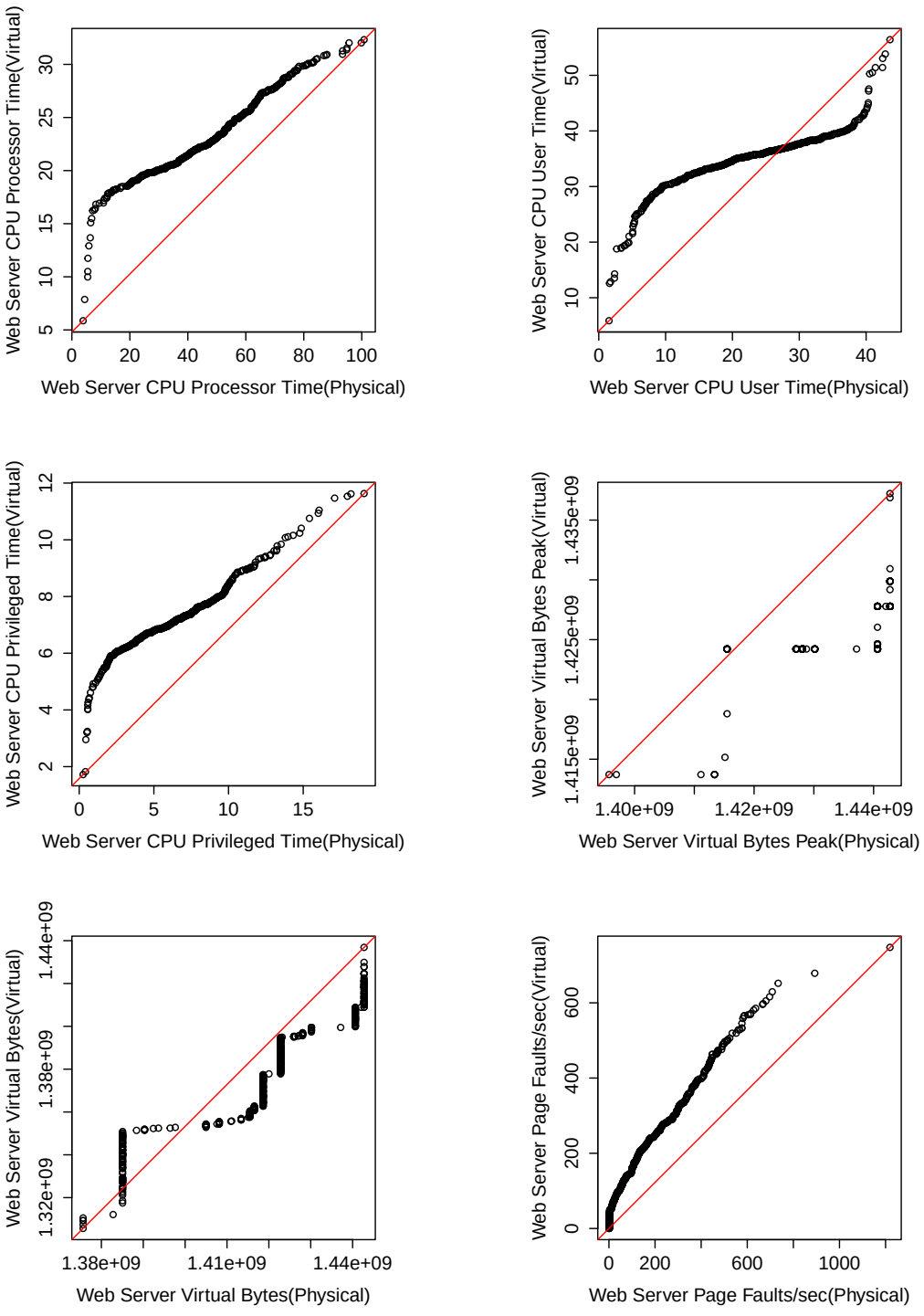


Figure 15: Q-Q plots for CloudStore's Web Server

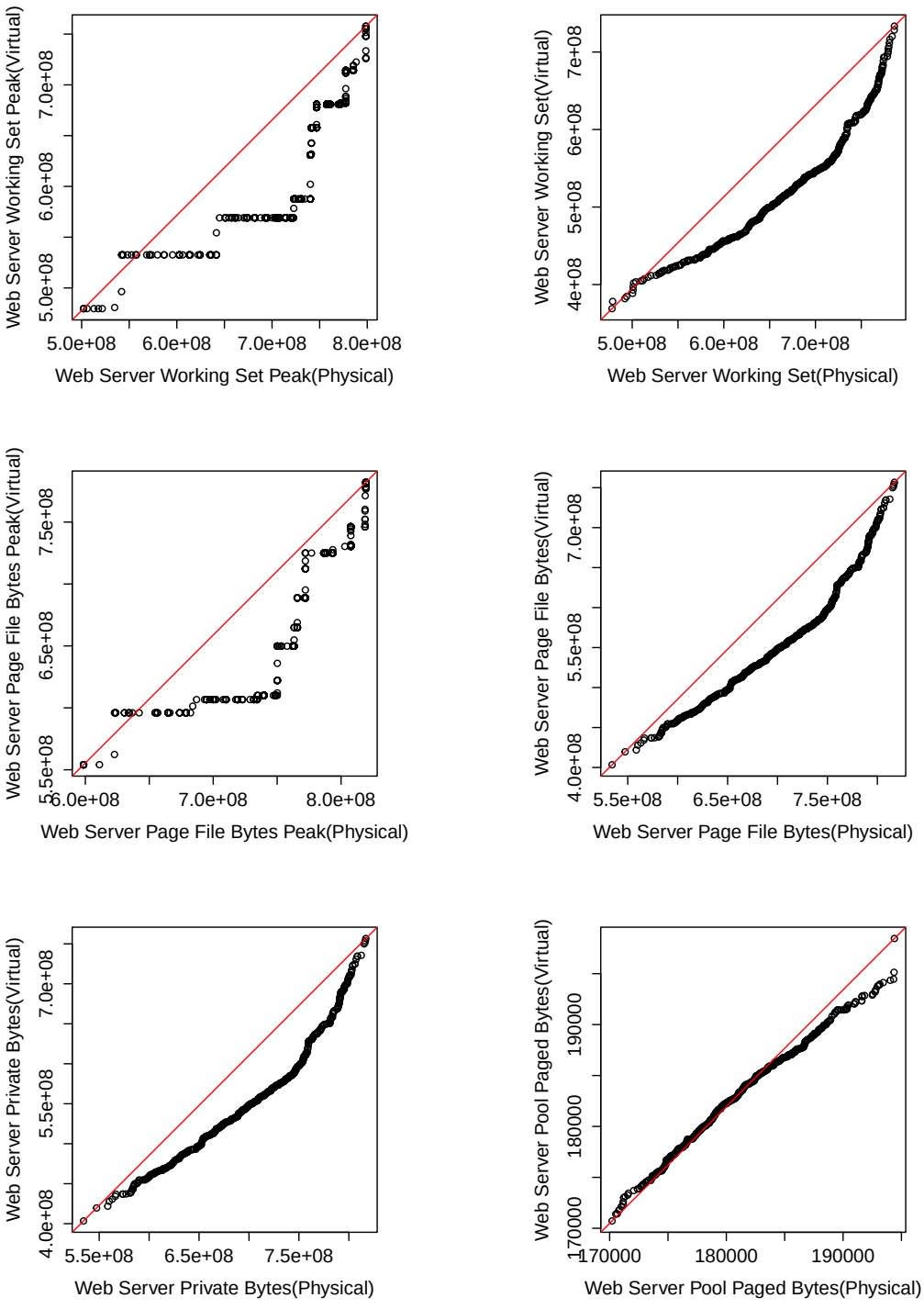


Figure 16: Q-Q plots for CloudStore's Web Server

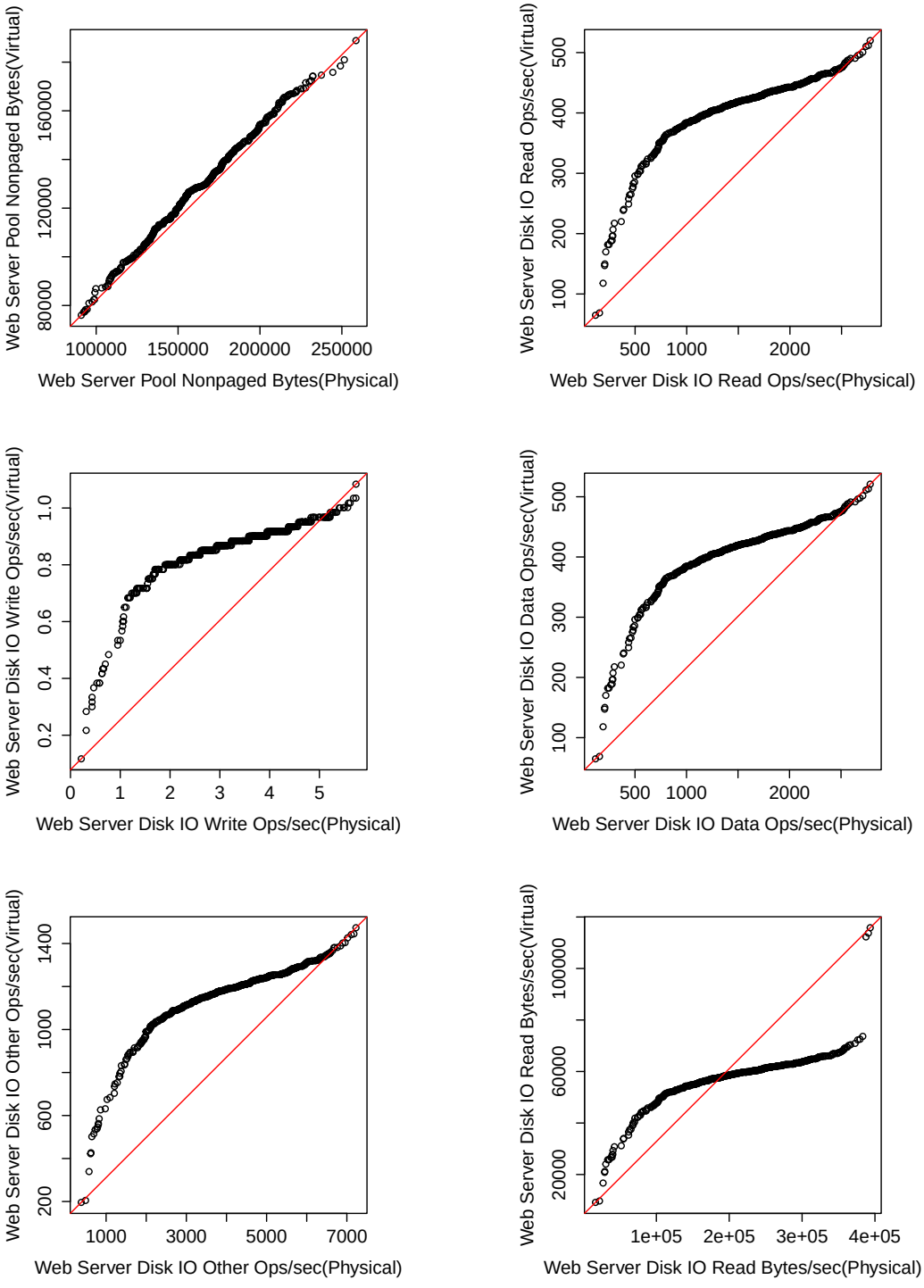


Figure 17: Q-Q plots for CloudStore's Web Server

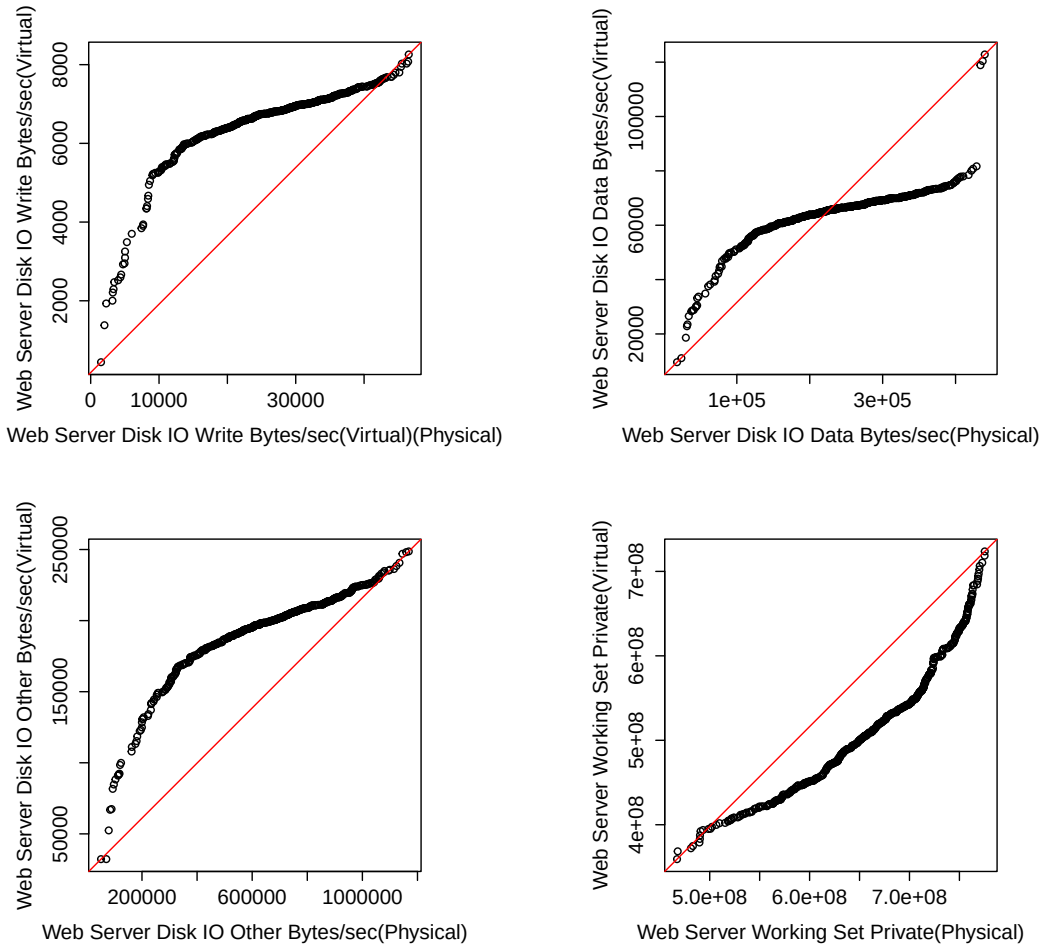


Figure 18: Q-Q plots for CloudStore's Web Server

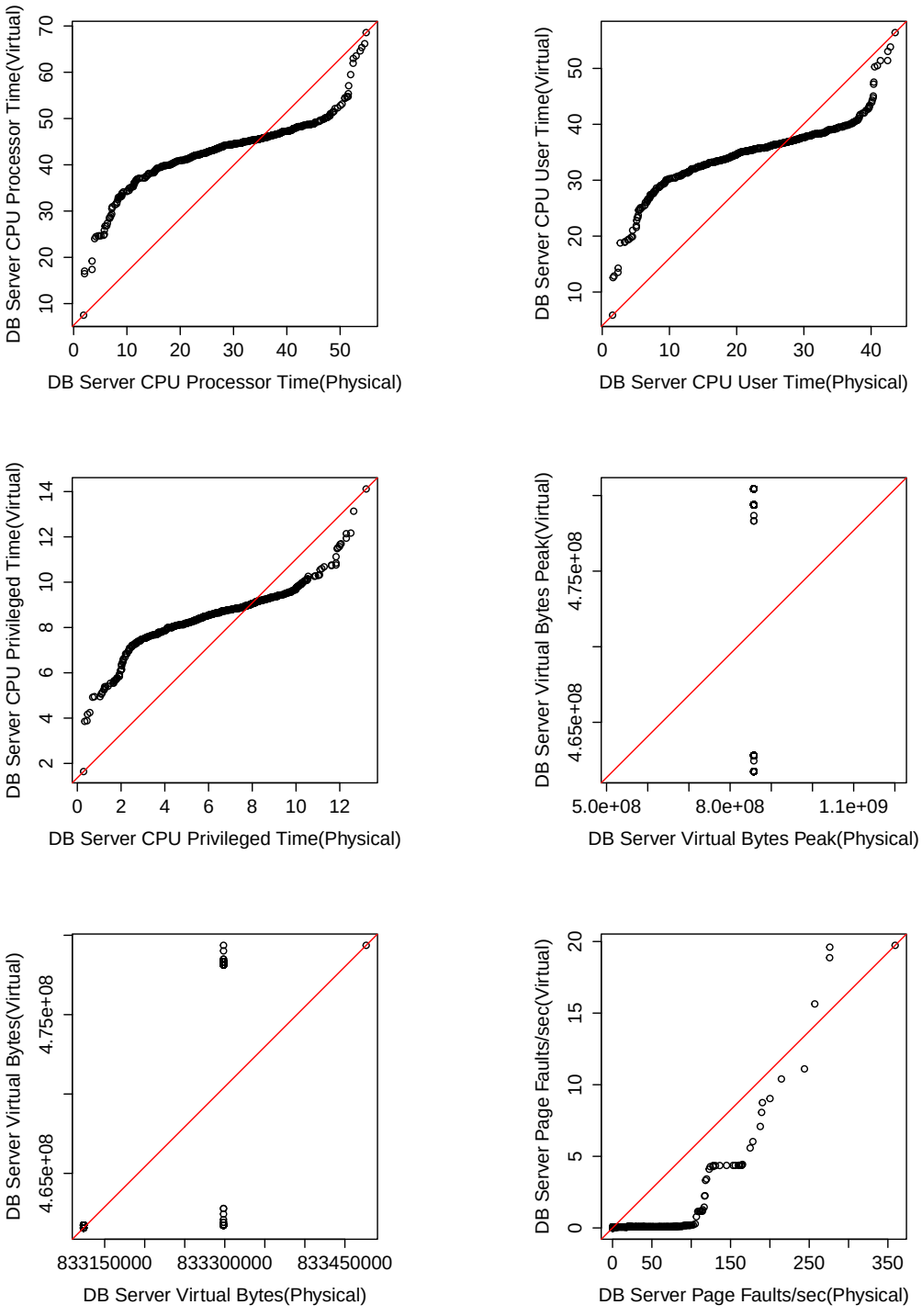


Figure 19: Q-Q plots for CloudStore's DB Server

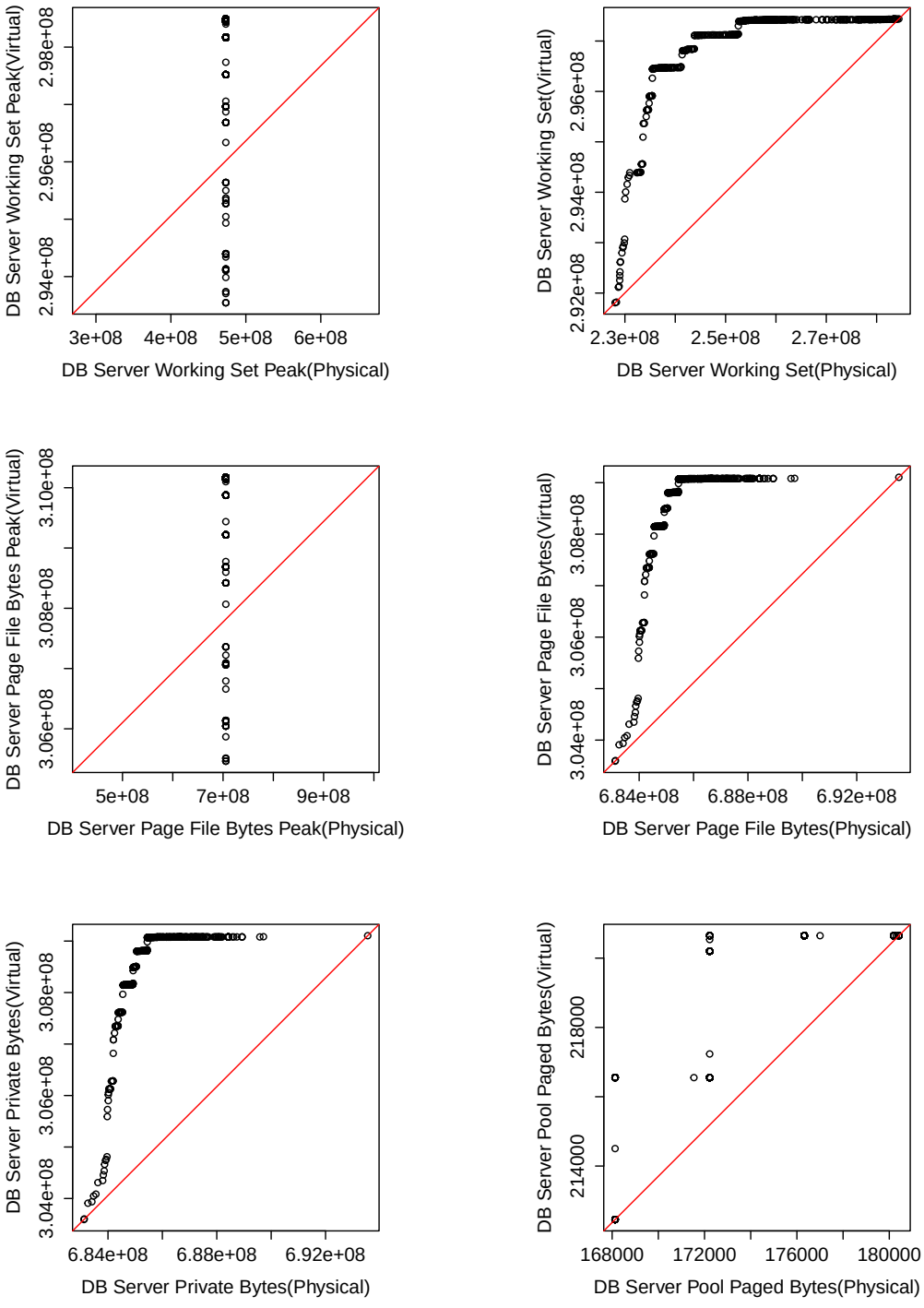


Figure 20: Q-Q plots for CloudStore's DB Server

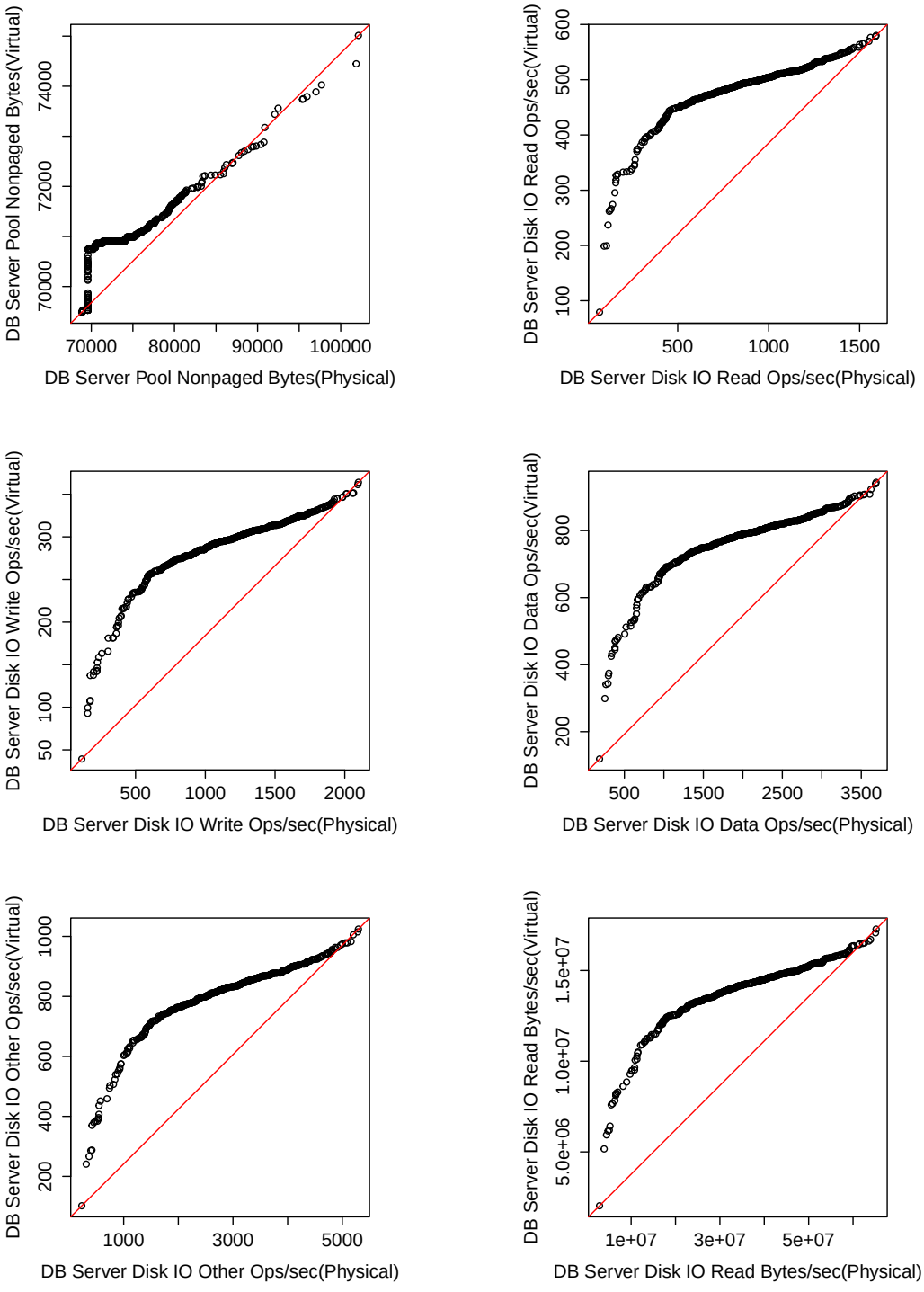


Figure 21: Q-Q plots for CloudStore's DB Server

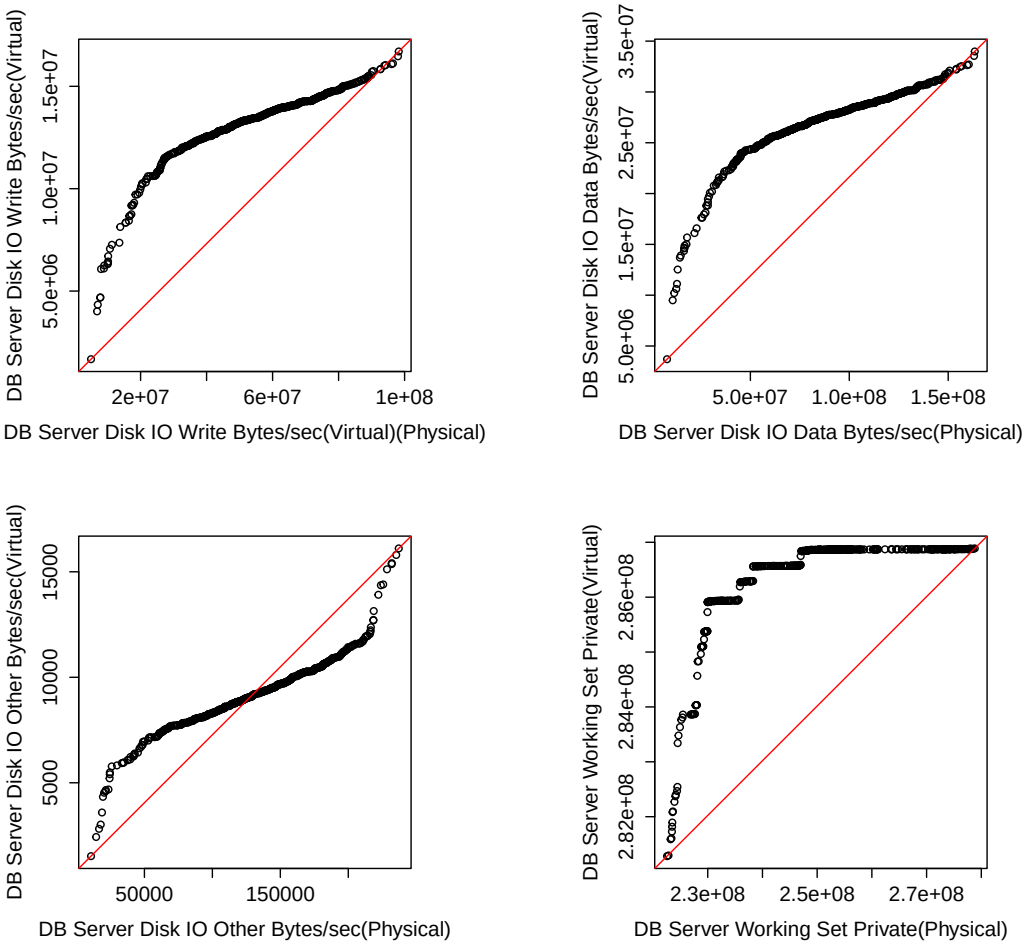


Figure 22: Q-Q plots for CloudStore's DB Server

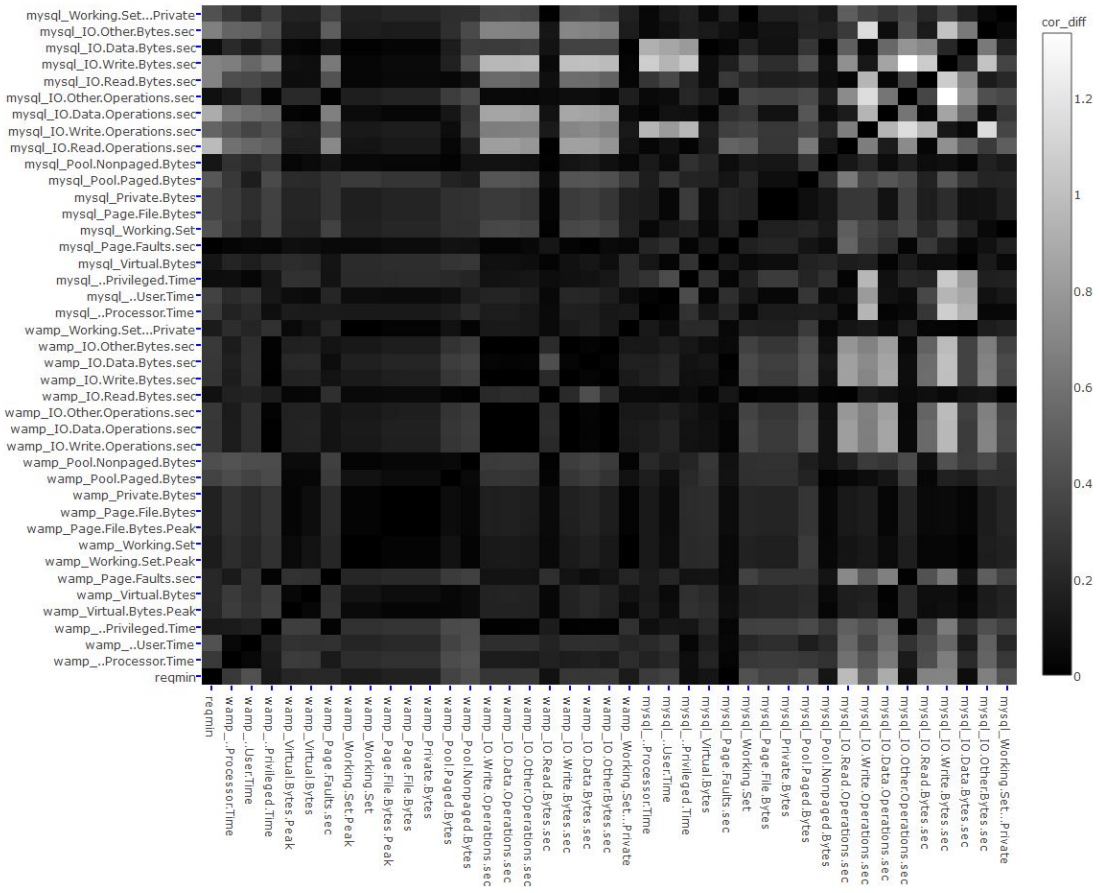


Figure 23: Heatmap (complete): DS2

Figure 24: Heatmap (complete): CloudStore

Bibliography

[ABC+16] Tarek M Ahmed, Cor-Paul Bezemer, Tse-Hsun Chen, Ahmed E Hassan, and Weiyi Shang. 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.

[AJ00] Martin Arlitt and Tai Jin. A workload characterization study of the 1998 world cup web site. *IEEE network*, 14(3):30–37, 2000.

[AK09] Mithun Acharya and Vamshidhar Kommineni. Mining health models for performance monitoring of services. In *Automated Software Engineering, 2009. ASE'09. 24th IEEE/ACM International Conference on*, pages 409–420. IEEE, 2009.

[And12] Andale. Statistics how to - coefficient of determination (r squared). `http://www.statisticshowto.com/what-is-a-coefficient-of-determination/`, 2012. Accessed: 2017-04-04.

[Apa] Apache. Tomcat. `http://tomcat.apache.org/`. Accessed: 2015-06-01.

[ARW96] A. Avritzer, J. P. Ros, and E. J. Weyuker. Reliability testing of rule-based systems. *IEEE Software*, 13(5):76–82, Sep 1996.

[AW95] A. Avritzer and E. R. Weyuker. The automatic generation of load test suites and the assessment of the resulting software. *IEEE Transactions on Software Engineering*, 21(9):705–716, 1995.

[BC06] M. S. Bayan and J. W. Cangussu. Automatic stress and load testing for embedded systems. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, volume 2, pages 229–233, Sept 2006.

[BC08] Mohamad Bayan and João W. Cangussu. Automatic feedback, control-based, stress and load testing. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 661–666. ACM, 2008.

[Bei84] Boris Beizer. *Software System Testing and Quality Assurance*. Van Nostrand Reinhold Co., New York, NY, USA, 1984.

[BGF08] Peter Bodík, Moises Goldszmidt, and Armando Fox. 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, pages 3–3, 2008.

[BGHK13] Fabian Brosig, Fabian Gorsler, Nikolaus Huber, and Samuel Kounev. Evaluating approaches for performance prediction in virtualized environments. In *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 404–408. IEEE, 2013.

[Bla14] BlackBerry. Blackberry enterprise server. `https://ca.blackberry.com/enterprise`, 2014. Accessed: 2017-04-04.

[BLG11a] Cornel Barna, Marin Litoiu, and Hamoun Ghanbari. Autonomic load-testing framework. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, pages 91–100. ACM, 2011.

[BLG11b] Cornel Barna, Marin Litoiu, and Hamoun Ghanbari. Model-based performance testing: Nier track. In *2011 33rd International Conference on Software Engineering (ICSE)*, pages 872–875, May 2011.

[CCW07] Joao W Cangussu, Kendra Cooper, and W Eric Wong. Reducing the number of test cases for performance evaluation of components. In *SEKE*, pages 145–150. Citeseer, 2007.

[CCW09] JoAO W Cangussu, Kendra Cooper, and W Eric Wong. A segment based approach for the reduction of the number of test cases for performance evaluation of components. *International Journal of Software Engineering and Knowledge Engineering*, 19(04):481–505, 2009.

[CGK+04] Ira Cohen, Moises Goldszmidt, Terence Kelly, Julie Symons, and Jeffrey S. Chase. 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, pages 16–16, 2004.

[Cha10] Anand Chakravarty. Stress testing an ai based web service: A case study. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 1004–1008. IEEE, 2010.

[CHP90] J. Chambers, T. Hastie, and D. Pregibon. *Compstat: Proceedings in Computational Statistics, 9th Symposium held at Dubrovnik, Yugoslavia, 1990*, chapter Statistical Models in S, pages 317–321. Physica-Verlag HD, Heidelberg, 1990.

[CLFG15] Jürgen Cito, Philipp Leitner, Thomas Fritz, and Harald C. Gall. 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, pages 393–403, 2015.

[Clo] CloudScale-Project. Cloudstore. `https://github.com/CloudScale-Project/CloudStore`. Accessed: 2015-06-01.

[CM99] Peter Csurgay and Mazen Malek. Performance testing at early design phases. In *Testing of Communicating Systems*, pages 317–328. Springer, 1999.

[CN01] P. M. Chen and B. D. Noble. 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.*, pages 133–138, May 2001.

[Cos15] Davide Costantini. How to configure a pass-through disk with hyper-v. `http://thesolving.com/virtualization/how-to-configure-a-pass-through-disk-with-hyper-v/`, 2015. Accessed: 2017-04-04.

[CZG+05] Ira Cohen, Steve Zhang, Moises Goldszmidt, Julie Symons, Terence Kelly, and Armando Fox. Capturing, indexing, clustering, and retrieving system history. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, SOSP '05, pages 105–118, 2005.

[DB13] Jeffrey Dean and Luiz Andr Barroso. The tail at scale. *Communications of the ACM*, 56:74–80, 2013.

[Dee14] 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-ph 2014. Accessed: 2017-04-04.

[Dev] Hewlett Packard Enterprise Development. Loadrunner. http://www8.hp.com/us/en/software-solutions/loadrunner-load-testing/. Accessed: 2017-02-16.

[Dil09] Bruno Dillenseger. Clif, a framework based on fractal for flexible, distributed load testing. *annals of telecommunications - annales des télécommunications*, 64(1):101–120, 2009.

[DJ] Todd Muirhead Dave Jaffe. Dell dvd store. http://linux.dell.com/dvdstore/. Accessed: 2015-06-01.

[DPE04] Giovanni Denaro, Andrea Polini, and Wolfgang Emmerich. Early performance testing of distributed software applications. In *Proceedings of the 4th International Workshop on Software and Performance*, WOSP '04, pages 94–103. ACM, 2004.

[DPE05] Giovanni Denaro, Andrea Polini, and Wolfgang Emmerich. Performance testing of distributed component architectures. In *Testing Commercial-off-the-Shelf Components and Systems*, pages 293–314. Springer, 2005.

[DRSS01] Reiner R. Dumke, Claus Rautenstrauch, Andreas Schmietendorf, and André Scholz, editors. *Performance Engineering, State of the Art and Current Trends*, London, UK, UK, 2001. Springer-Verlag.

[Eet] Kit Eeton. 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.

[FJA+10] King Chun Foo, Zhen Ming Jiang, Bram Adams, Ahmed E Hassan, Ying Zou, and Parminder Flora. Mining performance regression testing repositories for automated performance analysis. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 32–41, 2010.

[Fou] Apache Software Foundation. Apache jmeter. http://jmeter.apache.org/. Accessed: 2015-06-01.

[Fre09] David Freedman. *Statistical models: theory and practice*. Cambridge University Press, 2009.

[Gar08] Vahid Garousi. Empirical analysis of a genetic algorithm-based stress test technique. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1743–1750. ACM, 2008.

[Gar10] Vahid Garousi. A genetic algorithm-based stress test requirements generator tool and its empirical evaluation. *IEEE Transactions on Software Engineering*, 36(6):778–797, 2010.

[GBL06] Vahid Garousi, Lionel C Briand, and Yvan Labiche. Traffic-aware stress testing of distributed systems based on uml models. In *Proceedings of the 28th international conference on Software engineering*, pages 391–400. ACM, 2006.

[GBL08] Vahid Garousi, Lionel C Briand, and Yvan Labiche. Traffic-aware stress testing of distributed real-time systems based on uml models using genetic algorithms. *Journal of Systems and Software*, 81(2):161–185, 2008.

[Ghe] Grig Gheorghiu. Performance vs. load vs. stress testing. http://agiletesting.blogspot.com/2005/02/performance-vs-load-vs-stress-testing.html. Accessed: 2017-02-16.

[Gor00] Ian Gorton. *Essential Software Architecture*. Springer, 2000.

[Har01] Frank E Harrell. *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer, 2001.

[HCM05] Dean S Hoskins, Charles J Colbourn, and Douglas C Montgomery. Software performance testing using covering arrays: efficient screening designs with categorical factors. In *Proceedings of the 5th international workshop on Software and performance*, pages 131–136. ACM, 2005.

[HHF13] Christoph Heger, Jens Happe, and Roozbeh Farahbod. Automated root cause isolation of performance regressions during software development. In *ICPE '13: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 27–38, 2013.

[HMHR01] Frank Huebner, Kathleen Meier-Hellstern, and Paul Reeser. Performance testing for ip services and systems. In *Performance Engineering*, pages 283–299. Springer, 2001.

[HvQHK11] Nikolaus Huber, Marcel von Quast, Michael Hauck, and Samuel Kounev. Evaluating and modeling virtualization performance overhead for cloud environments. In *Proceedings of the 1st International Conference on Cloud Computing and Services Science*, pages 563–573, 2011.

[Jai90] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1990.

[JH15] Zhen Ming Jiang and Ahmed E Hassan. A survey on load testing of large-scale software systems. *IEEE Transactions on Software Engineering*, 41(11):1091–1118, 2015.

[JHHF09] Z. M. Jiang, A. E. Hassan, G. Hamann, and P. Flora. Automated performance analysis of load tests. In *IEEE International Conference on Software Maintenance, 2009. ICSM 2009.*, pages 125–134, Sept 2009.

[JMRW09a] Miao Jiang, M.A Munawar, T. Reidemeister, and P.AS. Ward. 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*, pages 285–294, June 2009.

[JMRW09b] Miao Jiang, Mohammad A. Munawar, Thomas Reidemeister, and Paul A.S. Ward. System monitoring with metric-correlation models: Problems and solutions. In *Proceedings of the 6th International Conference on Autonomic Computing*, pages 13–22, 2009.

[JSS+12] Guoliang Jin, Linhai Song, Xiaoming Shi, Joel Scherpelz, and Shan Lu. Understanding and detecting real-world performance bugs. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '12, pages 77–88. ACM, 2012.

[Kab11] Robert I. Kabacoff. R In Action. In *R In Action*, pages 207–213. Manning Publications Co., Staten Island, NY, 2011.

[KCK+11] Stephan Kraft, Giuliano Casale, Diwakar Krishnamurthy, Des Greer, and Peter Kilpatrick. Io performance prediction in consolidated virtualized environments. *SIGSOFT Softw. Eng. Notes*, 36(5):295–306, September 2011.

[Kea12] Sean Kearon. 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.

[KKB11] Mitashree Kalita, Sanjoy Khanikar, and Tulshi Bezboruah. Investigation on performance testing and evaluation of prewebn: a java technique for implementing web application. *IET software*, 5(5):434–444, 2011.

[Koh95] Ron Kohavi. 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, pages 1137–1143, 1995.

[Kuh08] Max Kuhn. Building predictive models in r using the caret package. *Journal of Statistical Software, Articles*, 28(5):1–26, 2008.

[LC16] Philipp Leitner and Jürgen Cito. 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, April 2016.

[LPG16] Qi Luo, Denys Poshyvanyk, and Mark Grechanik. Mining performance regression inducing code changes in evolving software. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 25–36, 2016.

[Ltd] RadView Software Ltd. Webload. http://www.radview.com/webload-download/. Accessed: 2017-02-16.

[MA01] Daniel A. Menasce and Virgilio Almeida. *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[MAD94] Daniel A. Menascé, Virgílio A. F. Almeida, and Larry W. Dowdy. *Capacity Planning and Performance Modeling: From Mainframes to Client-server Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[MAH10] H. Malik, B. Adams, and A. E. Hassan. Pinpointing the subsystems responsible for the performance deviations in a load test. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pages 201–210, Nov 2010.

[MDHS10] Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F Sweeney. Evaluating the accuracy of java profilers. *ACM Sigplan Notices*, 45(6):187–197, 2010.

[Men02a] Daniel A. Menasce. Load testing, benchmarking, and application performance management for the web. In *Proc. 2002 Computer Management Group Conference*, pages 271–281, 2002.

[Men02b] Daniel A. Menascé. Load testing of web sites. *IEEE Internet Computing*, 6(4):70–74, July 2002.

[Mer09] Christopher L Merrill. Load testing sugarcrm in a virtual machine. `http://www.webperformance.com/library/reports/Virtualization2/`, 2009. Accessed: 2017-04-04.

[MFB+] J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. Performance Testing Guidance for Web Applications - patterns and practices. `http://msdn.microsoft.com/en-us/library/bb924375.aspx`. Accessed: 2017-02-16.

[MHH13] H. Malik, H. Hemmati, and A. E. Hassan. Automatic detection of performance deviations in the load testing of large scale systems. In *2013*

35th International Conference on Software Engineering (ICSE), pages 1012–1021, May 2013.

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

[MJA+10a] H. Malik, Z. M. Jiang, B. Adams, A. E. Hassan, P. Flora, and G. Hamann. Automatic comparison of load tests to support the performance analysis of large enterprise systems. In *2010 14th European Conference on Software Maintenance and Reengineering*, pages 222–231, March 2010.

[MJA+10b] Haroon Malik, Zhen Ming Jiang, Bram Adams, Ahmed E. Hassan, Parminder Flora, and Gilbert Hamann. 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*, pages 222–231, 2010.

[MKAH16] Shane Mcintosh, Yasutaka Kamei, Bram Adams, and Ahmed E. Hassan. An empirical study of the impact of modern code review practices on software quality. *Empirical Softw. Engg.*, 21(5):2146–2189, oct 2016.

[MKMS10] Rajesh Mansharamani, Amol Khanapurkar, Benny Mathew, and Rajesh Subramanyan. Performance testing: Far from steady state. In *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*, pages 341–346. IEEE, 2010.

[MST+05] Aravind Menon, Jose Renato Santos, Yoshio Turner, G John Janakiraman, and Willy Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *Proceedings of the 1st*

ACM/USENIX international conference on Virtual execution environments, pages 13–23, 2005.

[MV00] Daniel A. Menasce and A. F. Almeida Virgilio. *Scaling for E Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[NAJ+12] Thanh H.D. Nguyen, Bram Adams, Zhen Ming Jiang, Ahmed E. Hassan, Mohamed Nasser, and Parminder Flora. Automated detection of performance regressions using statistical process control techniques. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ICPE '12, pages 299–310, 2012.

[NIS] NIST/SEMATECH. e-Handbook of Statistical Methods. `http://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm`. Accessed: 2015-06-01.

[NJT13] A. Nistor, T. Jiang, and L. Tan. Discovering, reporting, and fixing performance bugs. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 237–246, May 2013.

[NMV+11] Marco AS Netto, Suzane Menon, Hugo V Vieira, Leandro T Costa, Flavio M De Oliveira, Rodrigo Saad, and Avelino Zorzo. 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*, pages 993–1000. IEEE, 2011.

[NSML13] Adrian Nistor, Linhai Song, Darko Marinov, and Shan Lu. Toddler: Detecting performance problems via similar memory-access patterns. In

Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pages 562–571, Piscataway, NJ, USA, 2013. IEEE Press.

[Ora] Oracle. MYSQL server 5.6. `https://www.mysql.com/`. Accessed: 2015-06-01.

[PG11] BA Pozin and Igor V Galakhov. Models in performance testing. *Programming and Computer Software*, 37(1):15–25, 2011.

[Rod] Giampaolo Rodola. Psutil. `https://github.com/giampaolo/psutil`. Accessed: 2015-06-01.

[RTL09] Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Encyclopedia of Database Systems*, chapter Cross-Validation, pages 532–538. Springer US, Boston, MA, 2009.

[SAP11] Niclas Snellman, Adnan Ashraf, and Ivan Porres. Towards automatic performance and scalability testing of rich internet applications in the cloud. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pages 161–169. IEEE, 2011.

[She31] Walter Andrew Shewhart. *Economic control of quality of manufactured product*, volume 509. ASQ Quality Press, 1931.

[SHNF15] Weiyi Shang, Ahmed E. Hassan, Mohamed Nasser, and Parminder Flora. 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, pages 15–26, 2015.

[SJN+13] M. D. Syer, Z. M. Jiang, M. Nagappan, A. E. Hassan, M. Nasser, and P. Flora. Leveraging performance counters and execution logs to

diagnose memory-related performance issues. In *29th IEEE International Conference on Software Maintenance (ICSM '13)*, pages 110–119, Sept 2013.

[SM05] Monchai Sopitkamol and Daniel A Menascé. A method for evaluating the impact of software configuration parameters on e-commerce sites. In *Proceedings of the 5th international workshop on Software and performance*, pages 53–64. ACM, 2005.

[Sri15] Eric Srion. 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.

[SSJH16] Mark D. Syer, Weiyi Shang, Zhen Ming Jiang, and Ahmed E. Hassan. Continuous validation of performance test workloads. *Automated Software Engineering*, pages 1–43, 2016.

[Sta06] N. Stankovic. Patterns and tools for performance testing. In *2006 IEEE International Conference on Electro/Information Technology*, pages 152–157, May 2006.

[Sta08] James H. Stapleton. *Models for Probability and Statistical Inference: Theory and Applications*. WILEY, 2008.

[Sug17] SugarCRM. Sugarcrm. `https://www.sugarcrm.com/`, 2017. Accessed: 2017-04-04.

[Tec] CA Technologies. The avoidable cost of downtime. `http://www3.ca.com/~/media/files/articles/avoidable_cost_of_downtime_part_2_ita.aspx`. Accessed: 2016-03-16.

[Tin11] 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.

[TMM16] S. Tsakiltsidis, A. Miranskyy, and E. Mazzawi. On automatic detection of performance bugs. In *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 132–139, Oct 2016.

[TPC] TPC. TPC-W. www.tpc.org/tpcw. Accessed: 2015-06-01.

[Tys01] Jeff Tyson. How network address translation works. http://computer.howstuffworks.com/nat2.htm, 2001. Accessed: 2017-04-04.

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

[Wal29] Helen M Walker. *Studies in the history of statistical method: With special reference to certain educational problems.* Williams & Wilkins Co, 1929.

[WFP07] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *Future of Software Engineering, 2007.*, pages 171–187, May 2007.

[XPZG13] Pengcheng Xiong, Calton Pu, Xiaoyun Zhu, and Rean Griffith. 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, pages 271–282, 2013.

[YP96] Cheer-Sun D. Yang and Lori L. Pollock. Towards a structural load testing tool. *SIGSOFT Softw. Eng. Notes*, 21(3):201–208, May 1996.

[ZAH12] S. Zaman, B. Adams, and A. E. Hassan. A qualitative study on performance bugs. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 199–208, June 2012.

[ZC02] Jian Zhang and Shing Chi Cheung. Automated test case generation for the stress testing of multimedia systems. *Software: Practice and Experience*, 32(15):1411–1435, 2002.