A Study on the Reliability of The Performance Tests Done in Virtual Environment

Muhammad Moiz Arif, Weiyi Shang, Emad Shihab
{mo_ari, shang, eshihab}@encs.concorida.ca

Department of Computer Science and Software Engineering
Concordia University
Montreal, Quebec

Abstract

Performance testing plays an indispensable role for large software systems. Performance issues cause failures of these systems in the field more often than feature bugs. Such performance tests typically require large amounts of resources, such as long running time on performance testing environments. Making it worse, the ever evolving field environment requires frequently updating the performance testing environment. To address such challenges, virtual machines (VMs) are widely exploited to provide a flexible and less costly environments for performance tests. However, the use of VMs may introduce extra overheads (e.g., a higher than expected memory utilization) to the environment, leading to un-realistic performance testing results. Yet, there exists no study that qualifies such overhead in the context of performance testing activities, or investigates the impact of performance testing results.

In this paper, we perform a case study on two open source system, i.e.,Dell DVD and Cloud Store, to measure the impact of leveraging VMs for performance testing activities. In particular, we conduct two performance testing activities, including performance modeling and performance regression detection. The two activities are conducted in both environments that consists of physical machines or VMs. We observed that the comparison between the virtual and physical environments is not straight forward. We also observed that the performance of our subject systems between two environments is not identical.

(insert implications here once we have the results)

Keywords: Performance engineering, Performance testing Performance modeling Performance analysis

1. Introduction

In the software ecosystem, performance assurance activities play a vital role [36]. In essence, these activities ensure a consistent software functionality. The trend of dedicating a large chunk of costs, in some cases even exceeding the cost of development [4] to such performance assurance activities is now not unusual. In fact, most of the problems in the field are due to performance related issues [14] . A failure here would not only include an eventual decline in the quality of the software but also monetary and temporal losses. That is why companies like *Facebook, Amazon and Google* are committed to achieve excellence in this regard. [16]

The objective behind performance regression testing is to identify if there exists a lapse in performance for the newer version of the software compared to the previous versions. The system is tested by applying a fixed load which is congruent to a field-like load [36] [14] [18]. The performance analysts then look for deviations between metrics values compared to the earlier versions. Examples of factors causing performance lapse may be because of high CPU utilization or a memory leak. [18]. As there are no benchmarks for measuring the software performance cross environments, and with little or no time dedicated to performance assurance activities practitioners often find it hard to test and analyze the results of regression testing.

The need for performance testing environments to advance and evolve is continually augmenting and so is the cost associated with it [3] [4]. Consequently, one of the key problems practitioners face is lack of resources available for performance assurance activities in the dedicated or physical environment. That's why testing the system in a virtual environment comes to the foreground [44]. Before the system is deployed in the dedicated environment, a performance engineer tests the newer version of the software to look for performance loopholes in the virtual environment. The choice of running the performance assurance activities in a virtual machine is also based on the fact that majority of the client-server based systems are complex [22]. This enforces

a virtual set up of the environment which saves resources and is easier to refurbish according to the desired needs [33][35]

Regrettably, the road that leads to a comparison between the performance of the system in a dedicated and virtual environment remains undiscovered. Whether virtual machines are applicable in performance assurance activities or if they can be relied to behave equivalent to the physical servers still remains questionable. There have been limited instances of diagnosis of performance overheads [24] in the domain of performance engineering however no concrete conclusions have been drawn yet.

The goals of our study are to examine whether the performance metrics generated from our physical and virtual environments are homogeneous and correlated. Additionally, if they do not belong to the same population, then how impactful and effective are the prevalent discrepancies for a model-based regression testing approach.

To study this we set up two subject systems on an identical physical and a virtual environment. We use the performance metrics generated from both the environments to determine and compare the distributions. This is achieved by comparing plots and finding correlation between the metrics cross-environments. We use generalized linear regression models, as used in the performance assurance activities [36], to determine the extent of the comparison between metrics. If the metrics are transferable between the environments, we should expect to see a low percentage error. We chose to build our regression models based on multiple performance metrics to see the effect of the clustered performance metrics.

We diffuse our findings by answering the following research questions:

- RQ1: Do performance metrics from physical and virtual environments belong to the same population?
- RQ2: Is the metric correlation with load same cross-environments?
- RQ3: Do performance metrics from different environments impact performance modeling?

This paper is organized as follows. In section 2 we discuss background and related work. Section 3 demonstrates an example serving as a motivation for this paper. Sec-

tion 4 discusses the approach and the statistical techniques we adopted for this study. We present our case study results in section 5. Followed by discussion in section 6 and threats to validity in section 7. Section 8 concludes this paper.

2. Background and Related Work

To the best of our knowledge, the extent of the related work comparing the performance assurance activities carried out in the physical and virtual environments is limited. In this section, the related work we discuss use statistical techniques to detect performance regression on virtual environments only.

2.1. VM overheads

Kraft *et al.* [21] discuss the issues relative to performance modeling of disk I/O. They examine the performance degradation of disk request response time by recommending a trace-driven approach. Kraft *et al.* [21] emphasize on the latencies existing in virtual machines request for disc IO due to increment in time associated with request queues.

Aravind *et al.* [24] audit the performance overheads in Xen virtual machines. They uncover the origins of overheads that might exist in the network I/O causing a peculiar system behavior. However, there study is limited to Xen virtual machine only while mainly focusing on network related performance overheads.

2.2. Performance regression detection

Shang *et al.* [36] came up with a methodology to counter the approach of including only a limited number of performance metrics for the performance regression models. They recommend to use a an automatic clustering technique in order to select a subset of performance metrics out of the entire set of metrics. Ensued by building peformance models for each cluster. These models engulf the relationships of performance counters within each cluster. Shang *et al.* [36] also demonstrated that their approach is applicable to a system with injected performance regression. We use the same technique in our study to inject performance regression in the target system nonetheless the limitation of their study to perform their experiments in a virtual environment persists.

Belonging to the same background in the domain of performance engineering is pair-wise analysis. Nguyen *et al.* [28] introduce the concept of using control charts in order to detect performance regression, providing a solution for keeping numerous counters, only to make the tasks like book keeping and data analysis tedious for the practitioners. Control charts use a predefined threshold to detect performance anomalies. However control charts assume that the output follow a uni-modal distribution which an inappropriate assumption for the performance load. Nguyen *et al.* propose an approach to scale the metrics accordingly. However, the experiments are only carried out on the virtual machines in contrast to our approach.

Model-based approached use the target counters(e.g. DISC I/O and CPU Utilization) to build models and these models are then used to detect performance regression int he system. What make the model-based approach celebrated is the inclusion and comparison of numerous performance counters at the same time. Xiong *et al.* [49] proposed a model-driven approach to sofdetect software performance regression. The devised framework called *vPerfGuard* helps in detecting performance anomalies in a cloud-environment. Xiong *et al.* [49] only used virtual environments to test their framework, therefor our study is crucial to such state of the art research as it lays down the platform for putting confidence in such performance assurance practices. Jiang *et al.* [17] used an improved least square regression models to detect system faults. Cohen *et al.* [9] adapted an approach that includes fabricating probabilistic model, e.g. Tree-Augmented Bayesian Networks, to examine the causes that target the changes in the system's response time. Cohen *et al.* [10] also proposed that system faults can be detected by building statistical models based on performance metrics. The works of Cohen *et al:* 's [9] [10] were improved by Bodik *et al.* [5] by using logistic regression models.

Previous literature has its limitations as most of the performance regression testing and regression modelling is performed in virtual environments. Through our work we validate the usage of virtual environments in the field of performance engineering.

3. A Motivating Example

After presenting the background and challenges in the previous section, we illustrate an example of conventional performance testing activity.

Ben is a performance engineer who is working on a large-scale distributed software system. This system is capable of generating thousands of web requests and serving a number of clients all around the globe. Before the software is deployed, or even after a modification, it is Ben's responsibility to investigate the performance of the software versus the performance benchmarks.

A classic performance testing scenario would include Ben using a virtual machine to set up the desired environment. A virtual machine is a complete segregated and secure copy of the underlying physical architecture [42]. It can be tailored according the needs of required software hence saving resources. This is followed by deploying the system in a real world environment. He tests the system with a predefined load profile (for e.g. a set of request for logging in, browsing and logging out with x amount of users in y time) which is used to exercise the system extensively. During the testing phase, various counters are recorded. He will be comparing the results from a previously passed test in the virtual environment succeeded by the analysis of the results. [18] Or alternatively, exercising the older version of the system with the same load and then quantitatively comparing the counters to examine any signs of performance regression [42].

Ben investigates if there exists performance regression by selecting counters based on his experience and intuition for e.g. *disk reads/second*. Ben choses a model based approach, one of the techniques used to detect performance regression as discussed by *Shang et al.* [36]. Through this approach, Ben will predict the load and then compare it to the actual load. Subsequently, he considers the load as the dependent variable and the selected performance counter as the independent variable. He uses his chosen variables to build a linear regression model. In this example, Ben uses his model for the prediction of the web requests/sec, as the load, for the system under test. He will observe if there are any unexpected deviance between the predicted and the actual load. This will be followed by comparing the errors to a predefined acceptable threshold. If

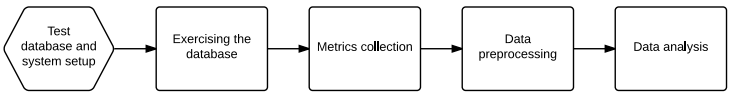

Figure 1: Approach Overview

he concludes that the system is not performing up to the mark, Ben will conclude that the system is exhibiting performance regression.

Ben did not discover any performance regression and proceeds to pass this system, allowing it to be deployed in the dedicated environment. However, the performance metrics generated by the physical environment are contradictory to what they were in the virtual environment. The system's performance metrics for CPU utilization are different than expectations. Ben overlooked these as he only chose disk reads/sec for his regression model. Further investigation showed that if 50% of the CPU is dedicated to the virtual machine, this will be considered 100% by the performance counters in the virtual environment thus mismatching the behavior of the software system in the physical environment where it has access to 100% of the underlying physical architecture. [45]

The above mentioned example, we note that a system passing a performance test in the virtual environment is not bound to follow the same course in the physical environment. Ben, according to his experience, build his performance model based on his chosen performance metrics because of which he failed to take into account into the deviation of the metric responsible for CPU utilization.

4. Approach

In this section we discuss the steps in our approach. The primary intention behind our experiment was to observe the performance of the system under test in a virtual

environment. Additionally, to analyze whether the metric values depict a similar behavior when compared cross-environment. Once the performance metrics are recorded and processed, we then analyze our results using graphical techniques and regression models. Figure 1 shows the synopsis of our approach.

4.1. System Setup

We chose Dell DVD Store (DS2) [12] as our test database. The DVD store is an online e-commerce web application with a multitier architecture, consisting of scripts for MySQL Server, *PHP* web pages and a load driver program written in *C-sharp*. [36] [28] [28]. *WAMP* [6] web application server were used to deploy and exercise the system while the database was set up on MySQL Server 5.6 [30]

Based on TPC-W standards [43], a performance benchmark originally proposed by the Transaction Processing Performance Council, our second subject system under test was also an open source application by CloudScale project [7] called CloudStore [8]. Just like DS2, CloudStore serves as online e-commerce website. An online book store, CloudStore is used as a standard in the domain of cloud computing. CloudStore's *JSP* pages were deployed on *Tomcat* [2] and MySQL Server 5.6 [30] was used to set up the database. The workload generator was based on scripts written for Apache Jmeter. [1]

Our system setup included three machines in a lab environment, Pentium i5 each with an 8GB of memory. To make the systems' configuration identical prior to exercising the subject system, we chose 2 cores and 3GB of memory dedicated to each environment to avoid crashes on the guest operating system. We opted for single tenancy of the guest operating system to avoid any unwanted noise. The first machine was dedicated to the database server, the second machine was dedicated to the web server and the third machine was used to run the load driver.

Following the set up of our subject systems on the respective servers, the systems were exercised with an aid of drivers. These drivers generated multi-type web requests and simulated real-time user behavior depending on the input parameters provided by us. We ran our performance tests for numerous hours while recording all the performance metrics generated for varying load applied on our software systems.

The load variation per run was introduced by the number of threads. A higher number of threads represented a higher number of users which would resultantly increase the number of requests per minute and vice-versa. As our study was based on exercising our systems and recording the performance metrics, and not stress testing [48], the respected limits were chosen in order to avoid the under-performance of the physical machine and system failure of the virtual machine.

4.2. Data Transformation

The values of all the available performance metrics were monitored, recorded every 10 seconds via *perfmon* [25][26]. *Perfmon* is a system performance monitor used to observe and record performance metrics like CPU utilization, Memory usage and disk IOs. For the reason that our web server and database server were on different machines, we recorded two data sets from two machines monitoring the application process resulting in a data set of 56 metrics. A set of every 6 recorded entries averaged out was labeled as a *single run* representing a minute. This way we had accumulated data from more than 500 runs. We used web server access logs to extract the requests that represented our load. Access logs record every web request that is received by the web server accompanied by the information such as IP addresses, URLs and timestamps [46]. With the help of a script we grouped the web requests per minute. The two data sets were then concatenated and mapped against requests according to the timestamps.

4.3. Data Cleansing and Data Analysis

Following the data transformation, we used R [34] to derive conclusions for our research questions with the help of Spearman correlation, q-q plots [29]; a graphical technique to detect if two sets of data belong to the same population, Mann-Whitney U test [27], and generalized linear regression models (GLM) [11] which are discussed later in detail in section 4. Spearman correlation is used to identify the level of association between two variables while Mann-Whitney U test test is used to verify the hypothesis that if the data sets belong to the same population. Both these methods do not strictly assume that the ordinal data is normally distributed [39] [13]. Due to the size of our data, heat maps [47] were used to visualize the correlations. We chose

model-based analysis as they can support the automatic selection and detection of performance abnormalities between heterogeneous environments. [36][28]. Contrary to the usual practice of ad-hoc selection of certain target performance metrics [15] to detect performance regression, we included exhaustive set of all 56 metrics to shape our models. For our linear regression models, we processed our data in two steps. First, we remove any metrics that showed no variance by a R script, for example a metric which has less than 20 unique values [31]. Next, we also remove the metrics which have high correlation amongst themselves to remove any *multicollinearity* present [23] so that the models do not over fit based on similar predictors(performance metrics) for the load. This approach will also magnify the metric which actually contribute the most to the model. The rationale behind aforementioned steps was to clean up data that may not contribute significantly to the model [37].

Once our data was organized our consequent footprint was to build the GLM using the physical and the virtual environment's performance metrics. Our models were built using the load as the dependent variable and the performance metrics as the independent variable i.e. the load is dependent on the change of values of the performance metrics. with the help. We, additionally, trained and tested our models for the same environment before using it cross-environments. This served as the benchmark for lower limit of the prediction percentage error. A model tested for the same environment was validated via ten-fold cross validation [38] [20]. Each fold is based on a random subset of values. The model is trained on 9 folds and tested on the 10^{th} fold. This is done 10 times with 10 different subsets of data each time. The accuracy of our models was determined by the percentage error of the predicted load versus the actual load values which was calculated as the absolute difference of the actual and predicted load values with respect to the actual load values.

Based on the results from the GLMs we then investigate about the metrics that are contributing the most and their respective correlation values using heat maps. Heat maps allow us to graphically view, in our case, the individual metrics and their correlation values.

5. Case Study Results

The objective of our case study is a comparative analysis of the two environments and to find out if there exists software performance regression between a dedicated server and the virtual environment. To analyze this, we divided our project into three research questions. Following the set up of the subject systems and data preprocessing, we used requests per minute as our independent variable and performance metrics as our dependent variables to answer our research questions.

5.1. ***Do performance metrics from physical and virtual environments belong to the same population?***

Motivation: Performance assurance activities are not only bounded by the idea that performance regression can only be identified relative to different versions of the software. A software system might also regress once it is transferred from one environment to another. According to the norm, practitioners often rely on performance metrics generated by the virtual environment prior to releasing it on a disparate environment. We address this question by comparing the metric values between the physical and virtual server.

Approach: As mentioned earlier in Section 3, after setting up the subject systems, we used the number of requests as our workload for the systems. A script was written to randomly alter the workload after every minute for CloudStore and every two minutes for DS2. The script was based on an underlying assumption that every unit of change in the performance metrics there will be a consequent unit of change in the workload which eventually introduced variety in our dataset. [41]. Subsequently, the requests per minute were mapped against the performance metrics recorded via *perfmon*.

The numerical values for requests per minute were extracted out of the web server's access logs. These requests were recorded per second and reached up to thousands in a minute, depending on the workload. We then, with the aid of a script, grouped and extracted the requests per minute for as long as the test systems were exercised. The metrics, on the contrary, were recorded every 10 seconds. A set of six records was averaged out as a minute according to the timestamps. This was called a single *run*.

The metrics generated for the web and database server were concatenated. Ultimately, we had a dataset consisting of 500 runs where every record represented a *run*; the metrics and requests generated for a given minute. This process was carried out for both of our environments. The two datasets, to analyze any discrepancies, were then analyzed using R's q-q plots and *cor* function.

Results: Figure 2 and 3 show the results from our q-q plots. For the sake of brevity, we chose one q-q plot to be displayed from each subset of metrics i.e. CPU, IO and memory. If the distributions of our performance metrics were similar, we should see the plots closer to the line y=x. For the sake of reference, we named this line 'Z'. [38]. We plotted the line Z on the same axes.

As seen in Figure 2 for the subject system DS2, the CPU user times for the web servers are closer to the line Z however the plots database servers' CPU user times are highly deviated from the same line as it not visible in the same plot. The disk IO operations/sec for the web servers gradually deviate form the line Z whereas the database server is, again, highly deviated. The same can be concluded about the memory working sets for both of our environments. Figure 3 shows the q-q plots for CloudStore. The web servers CPU user times are not congruent with the line Z. The database server CPU user times towards the tail of the plot are closer to the line Z however they still do not follow the line Z. The disk IO operations for both servers tends to follow the line Z initally but gradually moves away. Further on, the memory working set for both of our servers, as seen, are distant to the line Z.

Our Mann-Whitney U Tests also concluded that for each of the performance metrics selected they do not belong to the same distributions with a p-value < 0.05, except the web server's user times for DS2.

Table 1 and 2 shows the Spearman correlation values between the selected performance metrics. If the correlation value is closer to 1 the metrics have a strong correlation and if the value is closer to 0 the metrics have a weak correlation. A negative correlation means that if one of the metrics is increasing in value, the other is decreasing. For both of our subject systems we observe that the correlation values are mostly positive and are closer to 0 representing weak correlations. This conclusion, however, is not applicable to the user times from both environments as the p-value > 0.05.

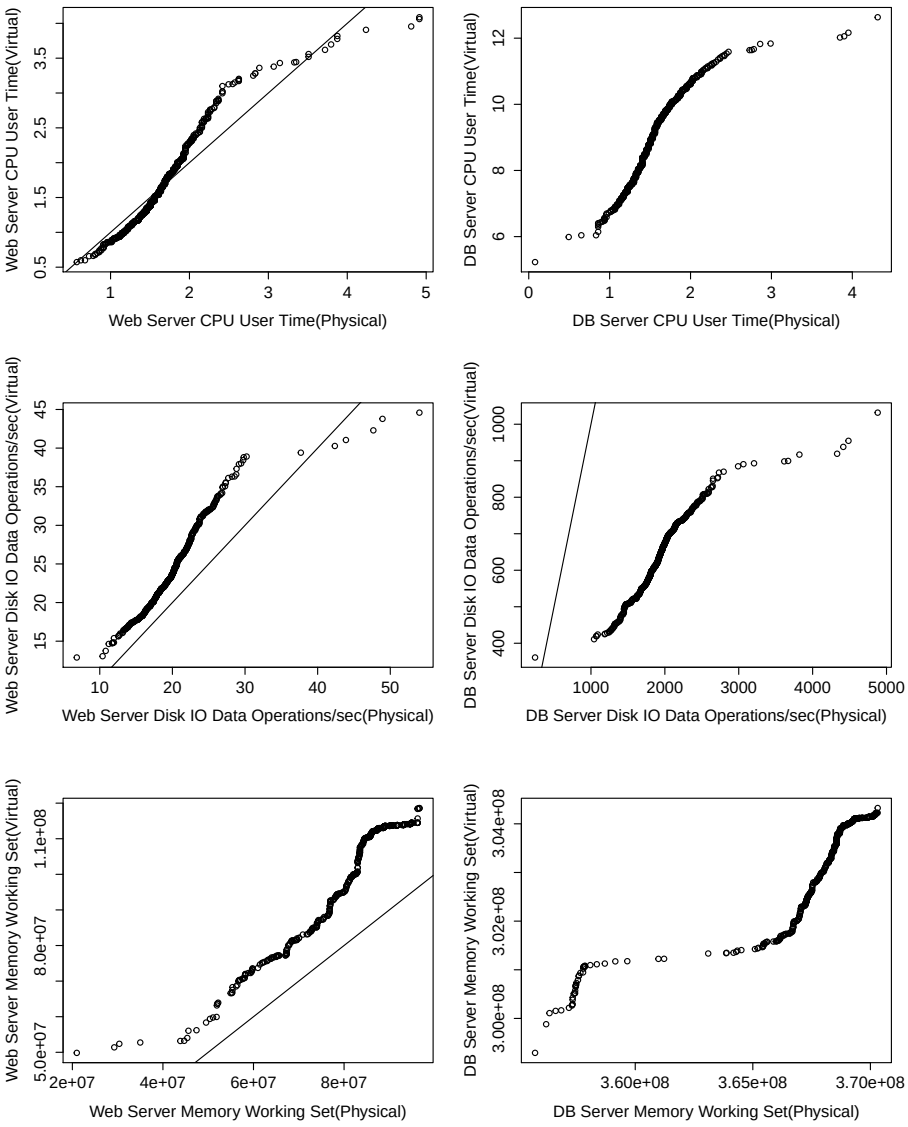

Figure 2: Q-Q plots: DS2

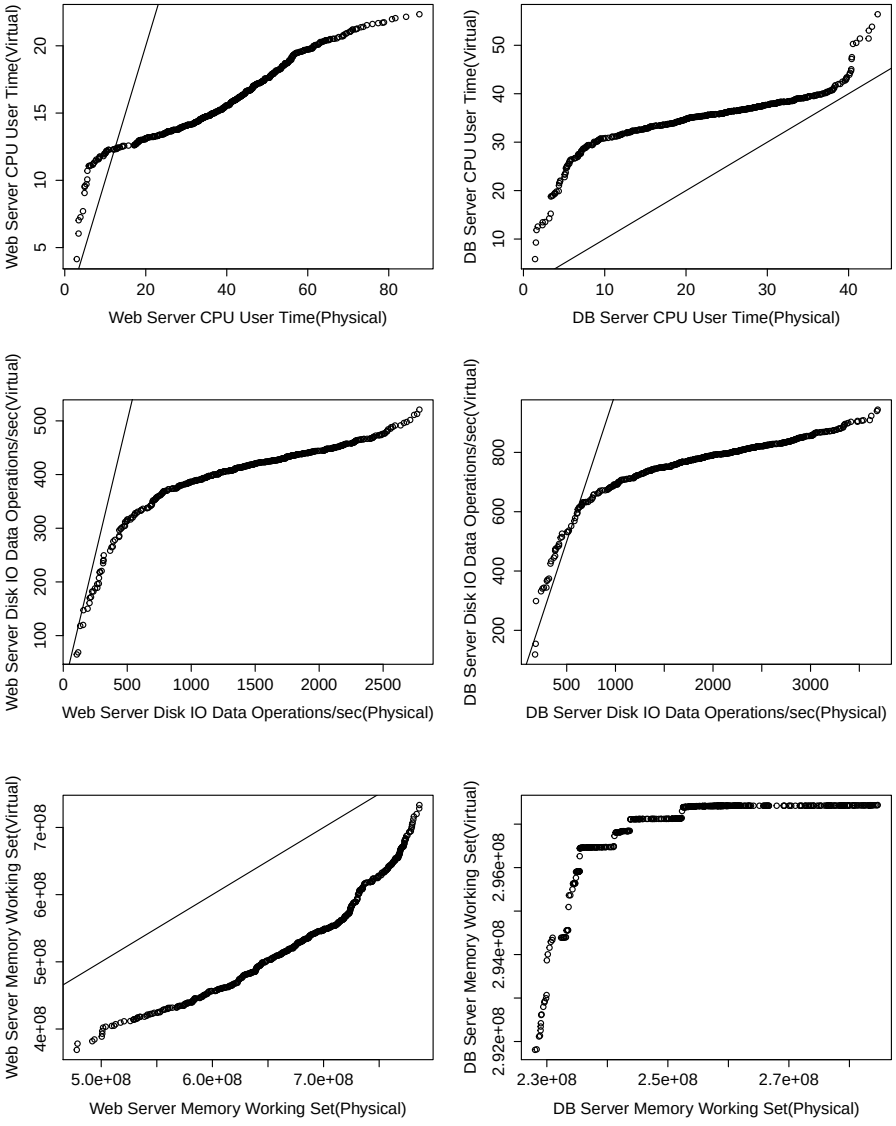

Figure 3: Q-Q plots: CloudStore

Table 1: DS2: Correlation Values

Performance Metrics	Cor	p-value
Web Servers' User Times	0.08	0.07
DB Servers' User Times	-0.05	0.30
Web Servers' IO Data Ops/Sec	0.25	0.000
DB Servers' IO Data Ops/Sec	-0.14	0.00
Web Servers' Memory Working Set	0.22	0.00
DB Servers' Memory Working Set	0.46	0.00

Table 2: CloudStore: Correlation Values

Performance Metrics	Cor	p-value
Web Servers' User Times	0.01	0.87
DB Servers' User Times	0.20	0.00
Web Servers' IO Data Ops/Sec	0.17	0.00
DB Servers' IO Data Ops/Sec	0.18	0.00
Web Servers' Memory Working Set	0.69	0.00
DB Servers' Memory Working Set	-0.13	0.00

5.2. *Is the correlation value between load and performance metrics same cross-environments?*

Motivation: Building on the conclusion from the previous research question, we next address the change in metric correlation values amongst themselves and versus the load. Our goal was to explore whether the change in correlation cross-environments is identical for both of our subject systems. This way we will be able to conclude whether a certain type of discrepancy is always present between the two environments or is it the unstable nature of the performance of subject systems in different environments.

Approach: We looked for the top 5 metrics which are highly correlated with the load in each of our environments. We used R's *cor* function to determine the Spearman value of the aforementioned associations. Next, we used heat maps to highlight the set of metrics which show a significant change in correlation values amongst themselves. The correlation values between the metrics of physical server were stored in matrix A. The correlation values between the metrics of the virtual server were store in matrix B. Matrix Z was the absolute difference between these two matrices. For example the Spearman correlation value for CloudStore's physical server between web server's User Time and database server's IO write/Bytes sec is 0.94. The correlation value for the same pair of metrics in the virtual environment is 0.26. Then the Matrix Z will record a value which is the absolute difference of the aforementioned Spearman values i.e. 0.68. If the metric value has changed significantly, according to the legend, this will be denoted as a 'hot zone' in the heatmap denoted by a lighter gradient of color.

Results: Figure 4 and 5 are the heatmaps for the change in correlation values amongst the metrics of two environments. For DS2, figure 4, the hot spots are mostly prevalent between the IO operations, for both the web and database server, cross-environments. This means the correlations amongst IO operations in the physical environment are not the same as the correlation between IO operations in the virtual environment. While CloudStore's heatmap, figure 5, shows that the change in correlation values is not as similar to DS2. Most of the hot spots are scattered across the heatmaps, contrary to DS2 where we can see clusters around most of the IO operations.

We also observed a similar change in correlation values between the processor times and other metrics. This trend may not be as strong as CloudStore's heatmap

for DS2, however these changes in values can be found in both of our subject system.

Table 3-6 are the top five highly correlated metrics with the load. In DS2, most of the IO operations from the web driver are highly correlated with the load in the physical environment. However, the database server is highly correlated than any of the metrics from the web driver in the virtual environment. Table 5 and 6 shows a much similar behavior of CloudStore in both the environments.

We primarily learned that the DS2 IO operations' behavior in one environment are not similar to that of the physical environment. We also learned that the change in nature of correlation cross-environments is non-uniform.

Table 3: DS2: Top 5 highly correlated metrics with load (Physical Server)

1. Web Server IO Other Operations/sec
2. Web Server IO Other Bytes/sec
3. Web Server IO Write Operations/sec
4. Web Server IO Data Operations/sec
5. Web Server IO Data Bytes/sec

Table 4: DS2: Top 5 highly correlated metrics with load (Virtual Server)

1. Database Server Handle Count
2. Database Server Working Set-Peak
3. Database Server Pool Paged Bytes
4. Database Server IO Other Operations/sec
5. Database Server Page File Bytes Peak

*5.3. **Do performance metrics from different environments impact performance modeling?***

Motivation: As discussed in earlier work [36] [28], performance tests require a large dedication of resources as it is carried out just before the system is on the brink

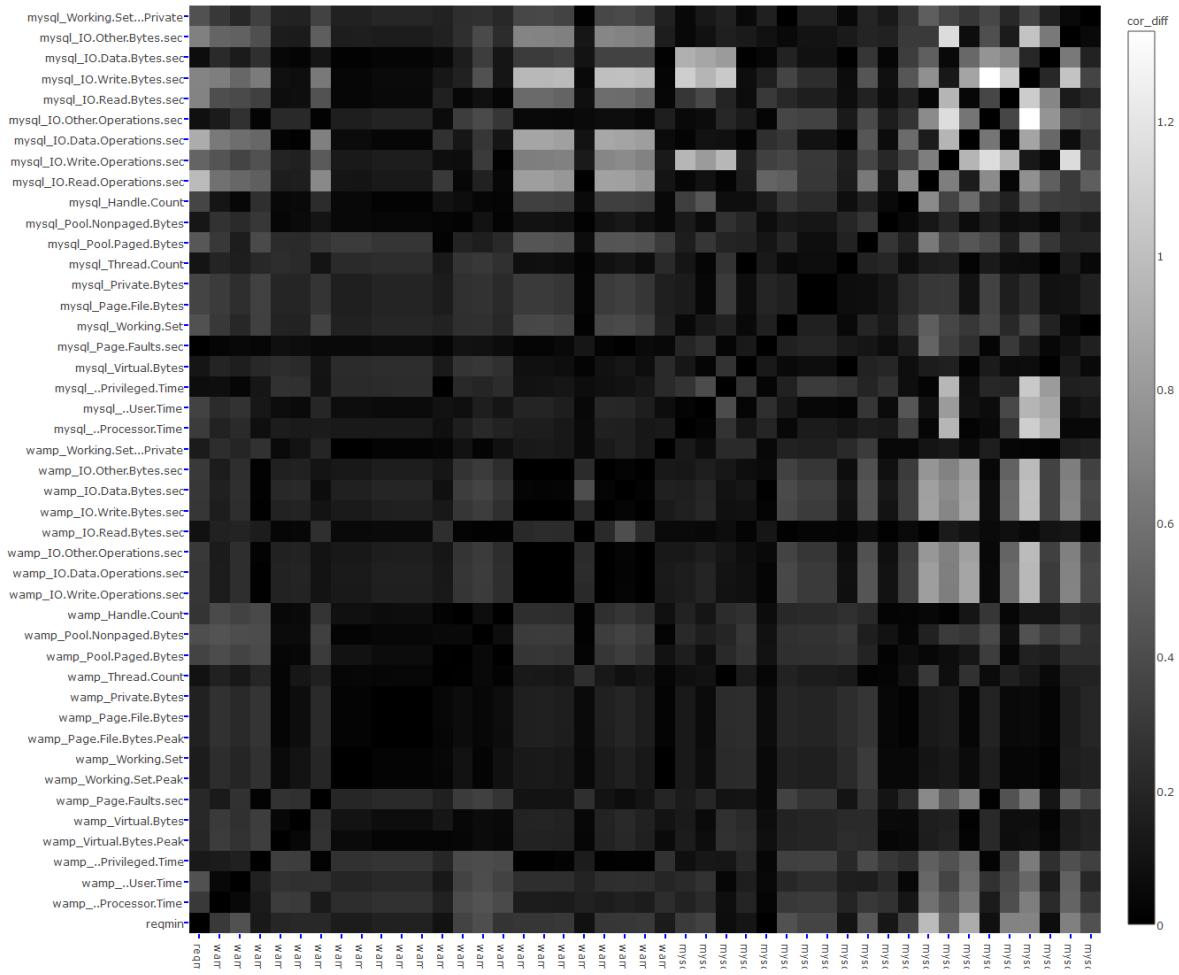

Figure 4: Heatmap: DS2

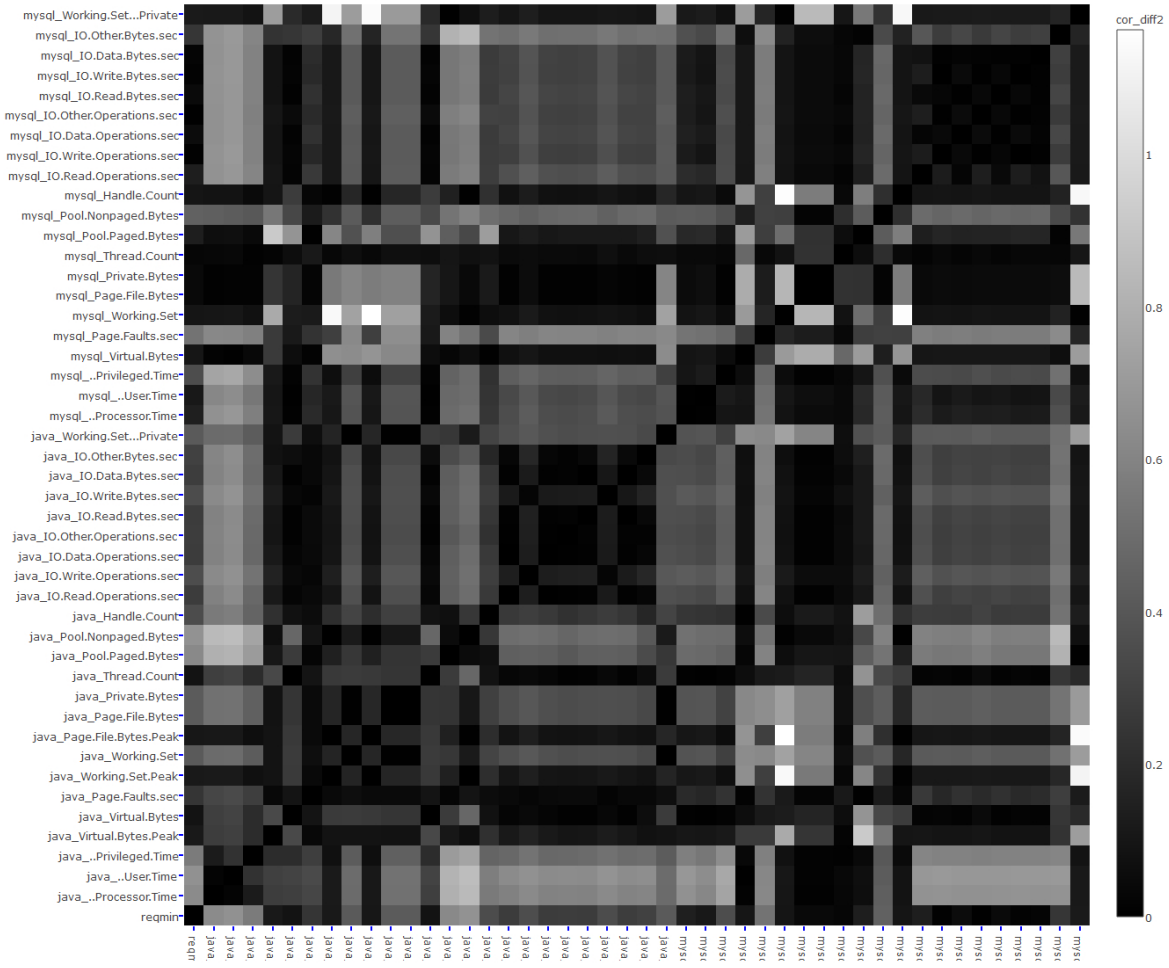

Figure 5: Heatmap: CloudStore

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

1. Database Server IO Other Bytes/sec
2. Database Server IO Read Operations/sec
3. Database Server IO Read Bytes/sec
4. Database Server IO Data Operations/sec
5. Database Server IO Write Operations/sec

Table 6: CloudStore: Top 5 highly correlated metrics with load (Virtual Server)

1. Database Server IO Other Operations/sec
2. Database Server IO Write Operations/sec
3. Database Server IO Write Bytes/sec
4. Database Server IO Data Bytes/sec
5. Database Server IO Read Bytes/sec

of deployment. This gives insufficient time to the performance engineers, leaving them with minimal resources. As a result they leverage on the performance assurance activities carried out in the virtual environment. The motivation behind this research question is to investigate whether the performance models generated from one environment can be applied and held representatives of the other. In essence, this will help the practitioners conclude the reliability of the performance activities carried out in the virtual environment.

Approach: We first validated our models with an environment using 10-fold cross validation. *K*-fold cross validation divides the data into *k* parts which are known as folds. The model is trained on *k*-1 folds and tested on the k^{th} fold. This process is iteratively repeated *k* times [32] [20].

We partitioned our results into two segments; the explanatory and the predictive part for our models, trailed by applying our model to predict load cross-environments. The explanatory part calculates the percentage of deviance explained by our models

i.e. the fit of the model while the predictive part explains the error percentage between the actual and predicted load values. Both of these parts were addressed by building generalized linear models that only incorporated the metrics which were selected via R *stepwise stepwise* or commonly knows as *stepwise* function, from the complete set of metrics from the dataset.

Explanatory Power

After removing any outliers for both of our subject systems, we built our GLM, initially, using all the physical server's metrics. We trained and tested our model on the same server i.e. physical. We then reduced our model, iteratively, using only the metrics that our contributing the most to the model. This was achieved using R's *stepwise* function. The *stepwise* function adds the independent variables one by one to the model to exclude any metrics that not contributing to the model [19]. Once the metrics were automatically selected out on the physical server, we used R's *ANOVA*, or *analysis of deviance* to rank the metrics according to their deviance value. Higher the deviance value for a given performance metric in the model, higher the contribution of the performance metric to the model. This approach was similarly applied to the set of metrics from the virtual server to build a generalized linear model. R's *deviancepercentage* was used to determine the explanatory part or fit of our model. It is used to calculate the percentage of deviance for a given GLM model.

As a result, we had two scenarios per subject system:

- Trained on physical, tested on physical.
- Trained on virtual, tested on virtual.

Predictive Power

The notion behind our predictive approach was to observe the percentage error between the actual and the predicted load. Our first set of predicted values were based on the model trained on the virtual environment's performance metrics and tested on the physical's server metrics. For the dataset of the virtual metrics, we wrote a script in R to remove the metrics that indicated practically zero fluctuation because else they will not have any impact on the GLM. This was trailed by the step to remove any highly correlated metrics and using *stepwise* for every fold [37]. We used mean absolute percentage

error (*MAPE*) to measure the error between the predicted and actual load values [40]. MAPE serves as the percentage measurement of the deviance of our forecasted values from our real values which makes it easier to interpret our results. If the error is 3, we say the forecasted value are off by 3%.

Based on our results, we then also scaled the virtual environment's metrics according to the physical environment. Our approach for scaling was based on by Nguyen *et al.* previous work [28]:

$$Load_{physical} = \alpha_{physical} \times Counter_{physical} + \beta_{physical}$$

$$Load_{virtual} = \alpha_{virtual} \times Counter_{virtual} + \beta_{virtual}$$

$$V_{scaledmetric} = (\frac{Load_{virtual} - \beta_{virtual}}{\alpha_{virtual}}) \times \alpha_{physical} + \beta_{physical} \quad (1)$$

To scale our metrics from the virtual environment, we first build a GLM with the selected counter and the load in the each of the environment. The selected counter was chosen on the basis of counters selected by *stepwise*, trained in virtual and tested in phyiscal environment. For every GLM there exists an intercept and a gradient value. We used the aforementioned values and applied them to the metric from the virtual environment as explained by equation 1. We also assumed that for each of the metric there will be a gradient and intercept value.

Results: Tables 7-10 show the results of our approach. In Table 7, for our subject system DS2, we see that the statistically significant metrics for the GLM from both of our environments are different. The significant of the performance metrics in an GLM is environment dependent. Due to automatic selection of metrics we see a high fit and lower *MAPE* values for our environments. Table 8 shows us the results when the model from the virtual was applied to predict load for the physical server and vice-versa. We observe a an immensely high *MAPE* value for the former while physical-virtual prediction is almost 50%. This means that the predicted and actual values of the workload have a mean absolute error percentage of almost 50%. When the same set

of metrics from the virtual environment were scaled the MAPE value for DS2 reduces drastically.

Table 9 shows us the results from CloudStore. Again, the ranks of the metrics that prove to contribute most to the model are not the same compared to both environments. After the data validation, we see a *MAPE* value of almost 16% for physical server and almost 5% for virtual. As the GLM was based on automatic selection via *stepwise* regression, we see a high percentage fit for our models. Table 10 shows us the results of cross application of models. A model that was trained on virtual server and tested to predict the workload values for the physical server was off by almost 29% whereas from physical applied to virtual it was off by 293%. Similarly, when the metrics were scaled, instead of a an expected decrease in the MAPE value we see that the MAPE jumps from 28.95% to 276.04%.

Not only the *MAPE* values are high when applying models cross environments, they are inconsistent between two environments. We conclude that for both of our subject systems, the models can not directly be applied to predict load for the other environment. We also observed that scaling according to the physical environment may or may not work. This is dependent on the selection of metrics in both of the environments. What might be significant in on of the models may not be applicable to the other model in another environment.

6. Discussion

In our research questions we concluded that comparing the physical and virtual environment and using the performance metrics may not be as straight forward. One of the reasons that this might stand true is because of the instability of the virtual environment itself. We now try to explore the if there are any discrepancy present in the nature of our virtual environment.

6.1. Repeatability

In order to check whether the performance test done in virtual environment are repeatable or whether the environment itself is instable, we carried out two more tests for

Table 7: DS2: Ranking of Performance Counters and Prediction errors

Training-Testing	Physical - Physical	Virtual - Virtual
Ranking	1. Web Server Privileged Time	1. Web Server IO Other Bytes/sec
	2. Database Server User Time	2. Web Server Page Faults/sec
	3. Database Server IO Read Bytes/sec	3.Database Server Working Set-Peak
	4. Web Server Page Faults/sec	4. Web Server Handle Count
	5. Database Server Privileged Time	5. Database Server IO Data Operations Bytes/sec
	6. Database Server IO Write Operations Bytes/sec	
	7. Database Server Pool Paged Bytes	
	7. Web Server Working Set-Private	
Fit	85.80%	67.10%
MAPE	7.02%	10.49%

Table 8: DS2: Prediction errors cross-environments

Training - Testing	**MAPE**
Physical-Virtual	49.97%
Virtual - Physical	6033.16%
Virtual - Physical **(after scaling)**	16.04%

each of our subject system. We trained our models on the new repeated tests and tested it on the baseline test used in our research question. The experiment was a replica of our previous experiments.

Yes, the performance tests carried out in the virtual environment are repeatable.

Table 9: CloudStore: Ranking of Performance Counters and Prediction errors

Training-Testing	Physical - Physical	Virtual - Virtual
Ranking	1. Web Server Privileged Time	1. Web Server IO Data Bytes/sec
	2. Database Server Privileged Time	2. Database Server User Time
	3. Web Server Page Faults/sec	3.Database Server IO Other Bytes/sec
	4. Web Server Virtual Bytes	4.Database Server Page Faults/sec
	5. Database Server Pool Nonpaged Bytes	
	6. Database Server Page Faults/sec	
	7. Database Server Page File Bytes	
	8. Database Server Working Set	
Fit	85.20%	89.10%
MAPE	15.75%	4.70%

Table 10: CloudStore: Prediction errors cross-environments

Training - Testing	MAPE
Physical-Virtual	293.62%
Virtual - Physical	28.95%
Virtual - Physical **(after scaling)**	276.04%

6.2. Resource Allocation

Next, we altered the resources allocated to the guest OS. This helped us deduce whether resource allocation makes a difference in a model driven approach. We increased our resources from 2 cores and 3GB memory to 3 cores and 5GB memory.

Similar to our previous approaches, we trained our model on the new set of performance metrics and tested it on the baseline test.

Yes, a change in resource effects the performance of the subject system in the virtual environment.

6.3. Type of Virtual Environment

We also investigated if the our subject systems behave similarly in different virtual environments. For this, we set up another type of virtual environment with the same specifications for the sake of identicality.

Yes, a change in virtual environment increased the error percentages for both of our subject systems.

7. Threats to validity

This sections discusses the threats to our validity.

7.1. External validity

We chose two subject systems, CloudStore and DS2 for our study and two virtual environments, VirtualBox and VMware. All of the former mentioned entities have a credible history when it comes to the domain of software performance activities. [18] [28]. Also, we made sure that our virtual environment is set up exactly the same as our physical environment by keeping a constant checks aided by scripts. Having said that, this study can be boosted by additional subject systems being tested in other types of virtual environment.

7.2. Internal Validity

All of our models are dependent on the performance metrics' accuracy. Which means if the load on the server is beyond the capacity of the system to handle and builds up a queue, there is a possibility of noise sneaking in the recording process of the performance metrics. The scaling approach we have adopted assumes that the for a particular metric, there exists alpha and beta values. However, if the metric value is

constant it not possible to have an alpha and beta value associated with it in the model. Hence looking for an alternate scaling process is necessary for such a performance metric. We build performance regression models to compare the metrics from both of our environments. This models are accurate if there exists a high number of records for the performance counters. Additionally, we also assume that none of our dependent variable is correlated to the independent variable or vice-versa.

7.3. Construct Validity

We compared our distributions using Mann-Whitney *U* Tests. As this test is commonly used to compare two distributions, other tests like T-test can also be used for this purpose. We used R's *step()* function to build our generalized linear regression models. This function automatically selects the variables contributing most to the models however the models might be different if they are only based on p-values. To address this, we plan to build models with a predefined threshold for the p-values.

8. Conclusion and Future work

Performance assurance activities are vital in the software industry. Before a software is deployed, performance engineers often test the subject system in a virtual environment rather than a physical environment. However, the reliability on this phenomenon of testing and relying in an virtual environment is questionable. Our paper magnifies the impact of performance in difference environments. Additionally, we observed that the performance metrics from different environment do not belong to the same distribution. We tried to mitigate this impact by using scaled metrics in our performance regression models. As a result, the percentage error dropped drastically for one of our subject systems are increased for the other. We concluded that scaling may not apply to every subject system. We plan to see in what ways we can leverage the metrics from the virtual environment and use them for prediction of the metrics in the physical environment. We also plan to investigate the injection of regression in the system being hosted in a virtual environment will behave similar to that under regression in a physical environment.