Challenges with Applying Performance Testing Methods for Systems Deployed on Shared Environments with Indeterminate Competing Workloads: Position Paper

André B. Bondi
Red Bank, New Jersey USA
bondia@acm.org

ABSTRACT

There is a tendency to move production environments from corporate-owned data centers to cloud-based services. Users who do not maintain a private production environment might not wish to maintain a private performance test environment either. The application of performance engineering methods to the development and delivery of software systems is complicated when the form and or parameters of the target deployment environment cannot be controlled or determined. The difficulty of diagnosing the causes of performance issues during testing or production may be increased by the presence of highly variable workloads on the target platform that compete with the application of interest for resources in ways that might be hard to determine. In particular, performance tests might be conducted in virtualized environments that introduce factors influencing customer-affecting metrics (such as transaction response time) and observed resource usage. Observed resource usage metrics in virtualized environments can have different meanings from those in a native environment. Virtual machines may suffer delays in execution. We explore factors that exacerbate these complications. We argue that these complexities reinforce the case for rigorously using software performance engineering methods rather than diminishing it. We also explore possible performance testing methods for mitigating the risk associated with these complexities.

General Terms

Performance; Measurement; Virtualized Environment.

Keywords

Software performance engineering; Performance measurement and testing; Cloud performance.

1. INTRODUCTION

Recent press reports highlight the tendency of corporations to shift the applications in their own data centers to a cloud-based shared environment [17]. We suppose that they will conduct performance tests in a similar environment, assuming that they do comprehensive performance testing at all. The analysis of performance test results in a shared environment that the testers cannot control may be muddied by interference from activities of an unknown or undisclosed nature. The response times experienced by users of an application will be impacted by the demand for processing power and other resources by the application of interest and, in addition, by the unknown and seemingly random demands and contention for those resource by other applications.

Murphy [13] has raised the concern that the performance engineering practices developed over the years for physical environments might not be applied by cloud practitioners, or that they might not translate well to a cloud environment. The focus in [13] is the performance engineering practices that would precede performance testing, as well as on measurement concerns. The outputs of performance monitoring of the cloud might not be the same as those in more traditional environment, and they might not have the same meaning. For example, the properties of the resource usage and delay metrics of VMware's virtual environment are discussed in [4]. Murphy argues that the added complexity of a cloud environment increases the need to apply performance engineering practices at all stages of the software life cycle. In particular, he identifies the need to ensure that the benefit of the cloud is exploited by building applications so that they can be horizontally scaled to exploit the potential for parallelism. Doing so enhances the load scalability of a system.

Applying performance engineering practices throughout the software life cycle should reduce the performance risk inherent in software projects [3], [5]. The potential impact of performance risk was vividly illustrated by the difficult rollout of the US government's health insurance web site, healthcare.gov, in October 2013.

The performance testing practices that have evolved for a purely physical environment may have to be adapted for a cloud environment. To mitigate the effect of background variability on the measured performance and resource usage of the system under test in an uncontrolled, shared environment, we propose that identical load tests of an application or service be repeated at different times, for an extended period each time. The times could be chosen to reflect known traffic variation or to reflect different configurations. Kosmann *et al* [9] used repeated experiments at constant loads to see whether they could detect whether the cloud provider would dynamically alter the configuration in response to the changing workload. Repetition could also be used to evaluate the effect of mechanisms to ensure fairness for multiple tenants, such as those proposed by Shue *et al* [16]. This approach is analogous to those used for agricultural

ICPE'16 Companion, March 12–18, 2016, Delft, The Netherlands.

ACM 978-1-4503-4147-9/16/03…$15.00.
DOI: http://dx.doi.org/10.1145/2859889.2859895

experiments or clinical trials [1]. There, fields or patients are assigned distinct treatments in a systematic manner. Multiple patients are treated and multiple fields cultivated to allow for variability in the effects. The results of the treatments are subjected to regression analysis, an analysis of variance, or some other ad hoc quantitative analysis. In an analysis of variance, variation is decomposed into explained variation, i.e. variation due to treatment differences, and unexplained variation, i.e. variation due to random effects. In the case of a cloud environment, random effects are due at least in part to random activity that is outside the control of the experimenter.

In [5], we argued that performance modeling methods could be used to structure performance tests to verify that performance requirements could be met and to verify and establish the bounds of system scalability. In a physical environment, performance tests at various offered loads can be used to determine whether hardware utilizations increase linearly with the offered load of a given type, and whether response time requirements can be met. Since throughput and response time respectively reflect the user demand and the user experience, these performance metrics will be of interest in both physical and cloud environments. For a physical environment, we have argued that, for ease of analysis, it is essential to conduct performance tests in a clean, dedicated test environment in which nothing else is going on, and that deviations from basic modeling assumptions and predictions were indicative of performance problems within the system under test. The challenge in a cloud environment is that the absence of a dedicated performance test environment will make it difficult to follow this advice, while complicating the design and analysis of performance tests. Shue *et al* [16] have proposed to mitigate the effect of resource contention by multiple tenants with heterogeneous resource usage patterns in a cloud environment through the use of fairness algorithms analogous to the rules used for time slicing in operating systems or to mediate packet dispatch in computer networks. They have described the effects that various types of tenants have on each others' resource usage through controlled experiments, but they have not attempted to describe a methodology for assessing performance and resource usage in the presence of competing workloads. It would be interesting to conduct replicated experiments with and without the fairness algorithms in place to see the effect on user response time and other performance metrics.

Making inferences about the results of performance tests conducted in an environment that is shared with systems and traffic over which the testers have no control is much more difficult. The tendency to move production to cloud or other externally hosted virtualized environments means that dedicated production and performance testing environments may not be available. At CMG2015, the author gave a talk [6] that was a highly abbreviated version of the tutorial he is delivering at the present conference [7]. At question time, more than one attendee asked how one should conduct performance tests in a shared environment so as to be able to make inferences about a system's performance and scalability. It seemed that the competing workloads on the target platform were unknown and possibly outside the testers' control. Time and lack of direct experience or preparation did not permit a detailed answer. Instead, the author pointed out that the lack of control over the environment has the potential to vary over time, e.g., by season or by time of day. The effects of this could be explored and mitigated for the purpose of performance evaluation by running the performance tests under like loads at different times of day, as has been done by Kosmann *et al* [9] to compare performance of a benchmark on different cloud architectures. This enables the estimation of means and variances, and hence of error bounds on performance metrics and resource usage metrics. The system under test would be subjected to constant transaction rates for prolonged periods, as one would in a dedicated controlled environment. A careful experimental design would be needed to do this effectively. We explore this and related questions in the remainder of this paper.

2. CHALLENGES IN TESTING FOR PERFORMANCE EVALUATION, PROBLEM DETECTION, AND DIAGNOSIS

When performance testing is done in a dedicated environment over which the testing team has complete control, the consequent reduction in ambient variability makes it possible to be confident in the results of performance tests without the need to repeat them to obtain confidence bounds on the measured values. This may not be the case when performance testing is done on shared platforms over which one does not have control. Techniques for quantifying and predicting the performance impact of competing workloads in mainframe environments when resource demands and throughputs are known have been available since the 1970s (see for example [15] and [12]). Their applicability is limited to systems in steady state. If the input parameters are not known, they can be varied for the purpose of sensitivity analysis so that a range of possible impacts can be determined. In shared environments whose use one cannot easily control, and in which the origins of the contending workloads might not be known or disclosed, evaluating the performance of a system or application poses a number of challenges, whether it is under a performance test or in production.

Our challenge is to determine whether performance degradation is caused by a property of the system of interest, by configuration issues in the system's virtual environment, or by resource contention from applications running in other guest machines on the same platform. Separating these issues is challenging because the resource usage metrics in a virtual environment could be dependent on the virtual machines' internal clocks, while contending virtual machines could impede each other's progress. An additional consequence of this confounding of factors is that it makes it difficult to know whether variations in a designated customer-affecting metric such as average response time are sufficient grounds to trigger software rejuvenation [2]. The problem is compounded by the possibility of considerable variability within and among the workloads and resource demands of other resident guests.

2.1 Compensating for Unknown Competing Applications on the Same Platform

In a shared environment with unknown competing workloads, the evaluation of performance metrics under different controlled load levels is analogous to making inferences about the yields of a crop planted in fields in different locations. Comparing the performance with two or more sets of distinct configuration settings is analogous to comparing the yields of two or more strains of the crop by planting them in grouped fields in different locations and subjecting the measured yields to an analysis of variance [1]. Similarly, if the background traffic in the target environment varies predictably by time of day, one can run load tests on the target platform under different configurations at like

times of day to determine whether one configuration or another yields better performance. Running load tests of a single configuration under like loads at different times of the day enables one to see which performance characteristics are dependent or independent of background activity. With this approach, one is attempting to separate out the random effect of "dirt in the test tubes." The load level and configuration settings are additional factors whose effect can be separated out using an analysis of variance as well. Examples of factors that we can control and about whose impact we wish to learn include configuration settings, hardware choices, choices of algorithms, software modifications, scheduling rules, and the choice of a virtual isolation mechanism as proposed by Shue *et al* [16].

In a virtualized environment, the performance measures and resource usage measures of a system under constant load might not be constant over time between load ramp up and ramp down, as would be the case in a well-behaved system being tested on a dedicated platform. Possible reasons for this include resource contention by other virtual machines, lack of access to physical resources by processes in a virtual machine because of stealing by other virtual machines (known as stealing), autonomic shifting of a virtual machine to another host, or periodicity of processing due to fairness algorithms that mediate access to hardware resources (as proposed in [16]). We note in passing that periodicity of access in the presence of randomly varying background loads could indicate that the fairness algorithm is working as intended, especially if the algorithm behaves in a periodic manner.

2.2 Identification of Deadlocks and Software Bottlenecks

Identifying software bottlenecks and deadlocks is challenging in a cloud environment because one of the symptoms is reduced CPU usage. The distinction is difficult because CPU usage by the application of interest could be reduced by competition from other applications or from other virtual machines. Deadlocks trigger sudden drops in CPU utilization followed by an increase when the deadlock is resolved. A software bottleneck is sometimes recognized when the CPU utilization cannot be increased despite an increase in the offered load. In some virtualized environments, one of the symptoms of a virtual machine being deprived of CPU by another is the portion of steal time, i.e., the time a virtual machine that has processes that are ready to run is blocked waiting for one or more physical CPUs. If deadlock occurs under a constant offered load, CPU usage drops. Since deadlocked processes will not use a virtual CPU either, the steal time should also drop to zero. If there is a software bottleneck, we should not expect the steal time to increase with the load offered to the virtual machine of interest either, even though the response time will rise with the offered load.

2.3 Placement of Load Generators

To avoid confounding the resource consumption of the system under test in a physical environment with the resource consumption of load generators, the latter should always be on a separate platform. Kosmann *et al* [9] mention that their load generators (Emulated Browser or EBs) were running in a separate cloud environment from that of the system under test. That environment itself could have been hosting some other system under load or under test. Thus, it is not impossible that the load they drove to their system under test could have been affected by the presence of other activity in its environment. This could slow down the processing of the next input data to be transmitted by an EB. The other application on the EB's host could also slow down the delivery of the response. Since each EB only generates a new request to the system under test when its previous request has been completed, the request rate can be reduced by competing activity in both the load generation environment and, as they point out, in the target environment. Thus, every effort should be made to deploy load generators on dedicated hosts.

3. RELATIONSHIP TO ARCHITECTURAL AND OTHER ASPECTS OF SPE

3.1 Engineering Process, Architecture, and Scalability

The apparent expandability of resources in a cloud environment does not mean that software performance engineering practices can be safely omitted during the software lifecycle [13]. Indeed, the greater complexity of diagnosing performance issues through measurement and performance testing in a cloud environment only reinforces the need for performance requirements specification, architecture reviews, and performance test planning. For the cloud environment, performance requirements should include specifications of metrics describing how the system responds to changes in the offered or background load, in terms of scaling, elasticity, and efficiency [11].

Without taking these steps, one cannot reduce the risk of performance issues that would arise even in the absence of the other workloads. It follows that rigorous performance-oriented architecture reviews and scalability assessments are all the more necessary to reduce the risk of performance and scalability issues that are inherently caused by architectural failings. Some of these might be detected through the use of automated tools such as CloudScale [10].

3.2 Instrumentation Challenges

When writing performance requirements and planning performance tests, one must bear cloud-related instrumentation challenges in mind:

1. The performance engineering process must include the identification and understanding of resource usage metrics that are peculiar to a virtualized environment, such as the per cent of the time that a guest machine is ready for execution as well as the proportion of time that processors are executing on behalf of that machine.
2. Because the observations of native instrumentation of hardware resource utilizations might not be available, and because the instrumentation of resource usage in the virtual machines might not have the same meaning as the native instrumentation, it will be necessary to heavily instrument application platforms such as database systems and application servers to uncover bottlenecks there, and examine their outputs from the start [14], rather than drilling down after (virtual) hardware resource usage patterns have indicated a problem.
3. Consequently, architectural and design choices and choices of platforms should be influenced not only by their functional characteristics, but also by the kind of performance instrumentation available for them.
4. The evaluation of resource usage measurements must be accompanied by the analysis of pertinent logs in application platforms such as databases, web servers,

and application servers. These logs must be equipped with accurate time stamps.

4. Configuration Issue: Virtual Processors

Configuration issues with a guest can be masked by concerns about performance generally. For example, in VMware, if a guest is configured with N virtual processors, it will not execute until that number of processors is available on the host [4]. A rigorous performance architecture review and a desk analysis that involves basic performance modeling can help us determine a suitable choice of N that is large enough to offer parallel execution of threads but small enough to reduce the risk of being starved of hardware CPU access. It should be as small as possible to prevent starvation due to excessive request rates by other guest machines requiring only one processor. Interestingly, there is an analogy with the problem of scheduling the allocation of magnetic tape drives to competing processes that require different numbers of them at different times [8]. If six drives are present, a deadlock might ensue if process A requires four of them and then two, while process B might require three of them and then four. Of course, there is a key difference between the magnetic tape allocation problem and the present one: the former occurs on a much longer time scale, with very infrequent requests for acquisition and release of discrete resources, while in a virtualized environment, it is the other way round. A performance model of the system in a dedicated environment may be needed to ascertain varying patterns in the demand for CPU usage and the like. Otherwise, to ascertain usage patterns and estimate relative device loadings, whether physical or otherwise, it may be necessary to run performance tests at a times when a guest is not competing with other guests for resources.

5. CONCLUSION

The movement of production and testing from privately controlled corporate environments to shared environments such as data centers and the cloud poses challenges to the interpretation and instrumentation of performance test results. Repeated experiments with identical configuration settings of the system under test are needed to compensate for the much higher variability that a shared environment imposes compared with that of a dedicated test system which has been cleared of background activity. At the same time, uncertainty about the interaction between the system under test and the host environment and uncertainty about the meanings of resource usage measurements in virtualized environments complicate the interpretation of test results and necessitate a thorough and rigorous review of the software architecture of the application and of the virtual environment in which it runs.

6. ACKNOWLEDGMENTS

The author has benefited from discussions with Alberto Avritzer and Raj Tanikella, and from the comments of one of the referees.

7. REFERENCES

[1] Anderson, V. I., and R. A. McLean. 1974. *Design of Experiments: A Realistic Approach.* Marcel Dekker, 1974.

[2] Avritzer, A., A. B. Bondi, and E. Weyuker. 2005. Ensuring stable performance for systems that degrade. *Proceedings of the 5th International Workshop on Software and Performance,* 42–51.

[3] Bass, L., Nord, R. L., Wood, W., Zubrow, D. 2007. Risk themes discovered through architecture evaluations. *WICSA 2007*, Mumbai, January 2007. 14

[4] Bell, P. 2015. Understanding VMware capacity. *Proc CMG2015*, San Antonio, TX, Paper 333.

[5] Bondi, A. B. 2014. *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice.* Addison-Wesley, Upper Saddle River, NJ.

[6] Bondi, A. B. 2015. Integrating Software and Systems Performance Engineering Processes into Software Development Processes. *CMG2015* (abstract only). http://edas.info/p19899#S1569514067

[7] Bondi, A. B. 2016. Incorporating Software Performance Engineering Methods Practice into the Software Development Life Cycle. Tutorial presentation, *ICPE2016,* Delft, the Netherlands.

[8] Habermann, A. N. 1976. *Introduction to Operating System Design*. SRA.

[9] Kossmann, D., Kraska, T., and Loesing, S. 2010. An Evaluation of Alternative Architectures for Transaction Processing in the Cloud. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, (SIGMOD '10) Indianapolis, Indiana, 579—590

[10] Lehrig, S. and Becker, S. 2015. The CloudScale Method for Software Scalability, Elasticity, and Efficiency Engineering: A Tutorial. *Proc. 6th ACM/SPEC International Conference on Performance Engineering (ICPE2015)*, Austin, Texas, 329—331.

[11] Lehrig, S. and Becker, S. 2015. Beyond Simulation: Composing Scalability, Elasticity, and Efficiency Analyses from Preexisting Analysis Results. *Proc. Workshop on Challenges in Performance Methods for Software Development, WOSP-C 2015*, Austin, Texas, 29-34.

[12] Lazowska, E. D., J. Zahorjan, G. S. Graham, and K. C. Sevcik. 1984. *Quantitative System Performance*. Prentice Hall. Available free online at www.cs.washington.edu/homes/lazowska/qsp/.1

[13] Murphy, J. 2011. Performance Engineering for Cloud Computing. *Computer Performance Engineering: Lecture Notes in Computer Science,* vol. 6977, 1-9. Springer, Berlin/Heidelberg.

[14] Paliwal, S. 2014. Performance Challenges in Cloud Computing. https://www.cmg.org/wp-content/uploads/2014/03/1-Paliwal-Performance-Challenges-in-Cloud-Computing.pdf

[15] Reiser, M., and Kobayashi, H. 1975. Queueing networks with multiple closed chains: Theory and computational algorithms. IBM J. of R. & D., 19(3), 283–294.

[16] Shue, D., M.J. Freedman, A. Shaikh. 2012. Performance Isolation and Fairness for Multi-Tenant Cloud Storage. *Proc. 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12),* Hollywood, CA.349—362.

[17] Weinberger, M. 2015. "The guy who pushed Netflix into the future says we're still underestimating Amazon." Interview with Adrian Cockcroft. Business Insider, finance.yahoo.com, November 18, 2015. http://finance.yahoo.com/news/guy-pushed-netflix-future-says-182431978.html;_ylt=AwrC1C31_E1WgmkAQHjQtDMD;_ylu=X3oDMTByOHZyb21tBGNvbG8DYmYxBHBvcwMxBHZ0aWQDBHNlYwNzcg--