Analysis of Performance Regression Testing Data by Transaction Profiles

Shadi Ghaith
School of Computer Science and Informatics
University College Dublin, Ireland
shadi.ghaith@ucdconnect.ie

ABSTRACT

Performance regression testing is an important step in the software development lifecycle, especially for enterprise applications. Commonly the analysis of performance regression testing to find anomalies is carried out manually and therefore can be error-prone, time consuming and sensitive to the input load. In our research, we propose a new technique that overcomes the above problems which helps the performance testing teams to improve their process and speeds up the entire software production process.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering

General Terms

Measurement, Performance

Keywords

Application change, performance models, regression testing

1. INTRODUCTION

As part of the performance testing of enterprise applications, the testing teams need to detect if a new release performs worse than previous releases, in which case a regression anomaly is to be declared [1]. To conduct performance testing, the application is exposed to a field-like load, via a load generator software, for an extended period of time [2]. Accordingly, the Transactions Response Times (TRT) and the system Resources Utilization (RU) are gathered and analysed to look for performance anomalies [3].

The process of analysing performance data to look for regressions faces the following two major difficulties:

Manual Process: Many industries still lack of an automated and portable solution to analyse performance data looking for regressions. During a manual process, testing engineers have to compare the TRT and RU of different releases to capture bad increases. As a high volume of data can be produced by enterprise applications, this manual process is error-prone and time consuming.

Load Dependency: When comparing TRT and RU between current and previous runs, both runs should be done with the same load (number of users). Otherwise it will not be possible to determine if the change is due to software change or load variation. Yet, new performance testing runs may need to be done with an increased workload to account for varying field requirements [4]. In such a case, multiple runs with different loads would be required which are time and resource consuming.

The fact that performance testing is usually run as the last step of an already delayed project leaves very little time to run such tests, and to analyse them [1]. Hence, the lengthy manual process is not suitable neither do the running of multiple tests with various loads [5].

In this research work, we are targetting an automated and generic solution to fix the two problems above. The solution should be automated, independent of the load and should not impose any new procedure, like new setup or performance runs, to the performance testing process. Consequently, our approach will speed up the performance testing process and so will reduce the whole time and cost required to produce the software and so reduce the time to market.

2. PROPOSED APPROACH

2.1 Queueing Network Model

Computer systems can be represented by a Queueing Network Model (QNM) [6] as shown in Figure 1.

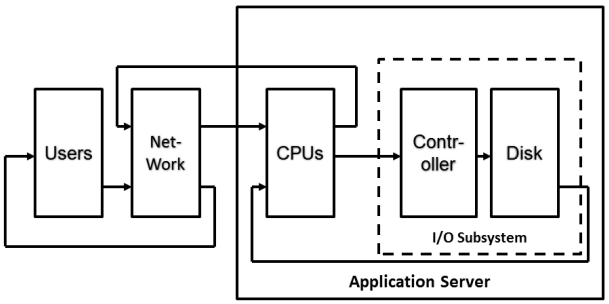

Figure 1: A Computer System Represented by a Queueing Network Model.

Each node in the network represents one hardware resource such as CPU, Disk I/O or Network. Each node is composed of a processing unit that will process the request if no other request is being served or the request will have to wait in the corresponding queue. Each request may visit any of the resources one or more times and require a service time to process the request (not including the time in the queues). The total time required to serve the request on any resource (execluding the time in the queues) during all the visits is known as Service Demands [6].

The QNM shown in Figure 1 can be represented as shown in Figure 2 if the BCMP theory [6] holds true. BCMP network is a class of queueing network for which a product-form equilibrium distribution exists. It is named after the authors of the paper where the network was first described.

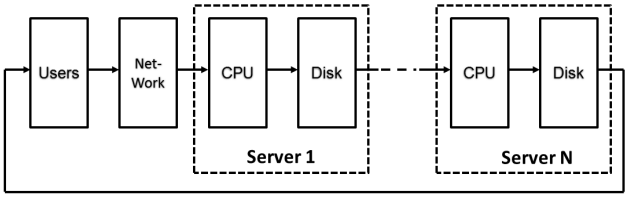

Figure 2: Representation of Computer System by Applying the BCMP Hypotheses.

2.2 Transaction Profile

Transaction Profile (TP) is defined as a series of service demands experienced by a transaction on all system resources [7]. It is considered as the minimum bound of TRT or the TRT when the request is the only one on the system. It can be represented as shown in Figure 3.

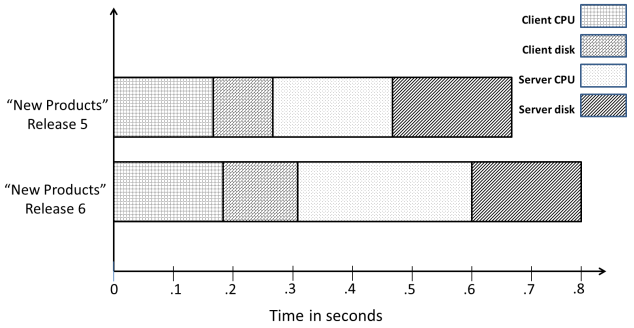

Figure 3: Comparison of TP for "New Products" Transaction Between Two Releases.

On one hand the TP represents the performance of the transaction and may change only if the application changes in a way that affects its performance. Such changes include more visits to a certain resource (e.g. the hard disk), an increase in the processing time of a certain resource, requiring new resources such as introducing a call to the database server. These changes affect the service demands and consequently the TP. On the other hand load applied to the system does not affect the TP which is defined as the time required to process the transaction on the resources' processing unit excluding the time on the queue. Therefore in this paper, we propose to use the TP concept in regression testing to highlight performance regressions caused by application changes and overcome the two challenges of the performance regression testing, namely the manual process and load dependency.

The central premise here is:-

An automated (or visual) comparison of the TPs between the two releases will highlight the performance regressions caused by application changes as opposed to those caused by load variations.

This approach is depicted in Figure 3. It shows the TP comparison of the "New Products" transaction between two releases and shows that the new release contains a regression in the application code that caused an increase of the CPU utilization on the server machine.

2.3 TP Run Report

In order to utilise the TP approach in performance regression testing, we designed the TP Run Report shown in Figure 4.

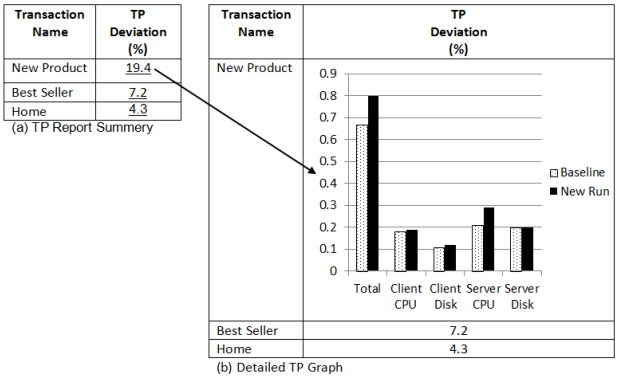

Figure 4: TP Run Report.

The TP Report Summary shown in Figure 4a shows the TPs which values in the new run deviates from their values in the previous run by a pre-set threshold of 3%. Each of these transactions corresponds to a potential performance regression caused by an application change. To get more information about each of these regressions, the user can click on the link under any of those deviation values to get the corresponding Detailed TP Graph shown in Figure 4b. This gives detailed information about the per-component contribution to the TP deviation.

2.4 Our Approach to Obtain TP

It is possible to measure the TP using a special setup equipped by a load generator software such as JMeter [8]. We think that introducing such a step to the testing process is considered as an extra overhead that makes the approach less attractive. So we propose to calculate the TP using data already produced by performance testing process (TRT and RU) and the QNM of the testing system.

To explain our approach, we first describe the use of TP in capacity management process [7] which is depicted in Figure 5. At the left hand side we see the inputs in this process which are the workload information and the service demands. The service demands are measured as mentioned above. The QNM in the centre of Figure 5 is generated to represent the system for which the capacity is to be calculated. This model is solved via a variety of tools such as Java Modelling Tool (JMT). Upon solving the model, the RU and TRT for the various resources/transactions are calculated which forms the output of the capacity management process. The input workload information and the QNM are then varied to evaluate the system capacity.

It can be noticed that the output parameters of the capacity management process (TRT and RU) are the data available to the performance regression testing process. The QNM is also available knowing the testing system. Also, the workload information can be found from the load gen-

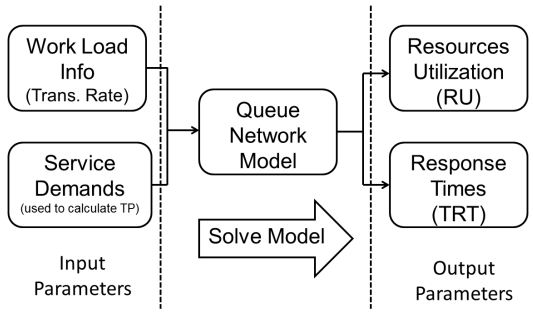

Figure 5: Capacity Management Approach.

erator scripts. Given these data, we propose the reverse process shown in Figure 6 to calculate the service demands and hence the corresponding TPs.

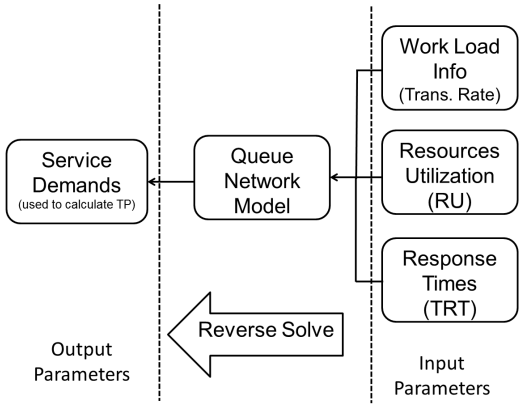

Figure 6: Proposed Performance Regression Testing Process.

The QNM can be reverse-solved using JMT (or similar tools) by applying a search based approach as shown in Figure 7. The model is first solved with an initial TP value (such as one from the previous run). The calculated TRT and RU are then compared with the target ones (available from performance testing). If they do not match the input TP is adjusted and the QNM is solved again. This is continued until the calculated and target TRT, RU match in which case the required TP is found.

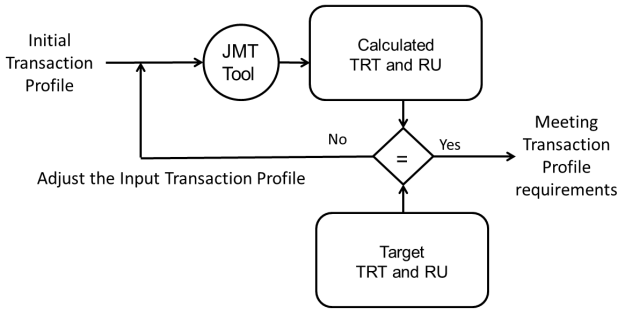

Figure 7: Reverse-solve Queueing Network Model.

2.5 System Overview

Figure 8 shows the entire proposed system. The performance regression New Run Data is used as an input to the process in addition to the TP of Previous Run(s). The Load Generator data is parsed to find the transaction types, TRT's and transaction rates. The System Monitoring data is parsed to get the resources types and RU's. The transaction and resources types are used to build the QNM as shown in Figure 2. Then all of the TRT, RU and the transaction rates along with the QNM are used to calculate the TPs of the various transactions as shown in Figure 6. Those TPs are then compared to the TP Previous Run(s) ones to generate the TP Run Report as shown in Figure 4.

3. METHODOLOGY AND CONTRIBUTION

Our research is conducted in the following four phases:

Validation of TP: In this phase we verified that the TP provides an accurate representation of the performance characteristics of the transactions of an open source web 2.0 application. This has been achieved by measuring the TP via JMeter software [8] in a way similar to the techniques used in the capacity management. From this experiment we found that TP truly represents the performance characteristics of the transaction and only changes when we introduce performance overheads to the software code for particular transactions. A detailed description of this work can be found in our previous work [9].

Implementation: We implemented the solution shown in Section 2.5. We parsed the data produced by the load generator and monitoring tools to obtain various resources and transactions information. We built the QNM of the system from these data as shown in Figure 2 and reversely solved the model to get the TP of the various transactions as described in Section 2.4. We then automatically produced the TP Run Report.

Evaluation: The solution was verified on a sample open source web 2.0 application (JPetstore 6.0 [10]) by getting the TP with and without an introduced common performance bugs, for example missing indexes on database tables. We found that the solution is capturing the performance regression anomalies with a precision of 80%. These results have been included into a paper which is currently under review.

In addition to evaluating our approach on the open source web 2.0 application (JPetstore 6.0 [10]), we plan to validate the technique on a real large scale enterprise application. For this application we have access to a repository of the information for the various performance runs data over multiple releases. The data includes performance regression anomalies found by the current manual approach in each run. We will generate the TP Run Report for each of these releases and compare its results (performance regression anomalies) to the real ones in the repository.

The results of this part will be published in short paper.

Enhancements: In this phase we plan to account for software contentions in the QNM to simulate real life applications where the software resources, such as threads and data sources, can form the bottleneck of the system. This work will be published in a separate paper.

The contribution of our work is a new tool added to the toolbox of performance testing engineers that helps them identifying the performance regression anomalies.

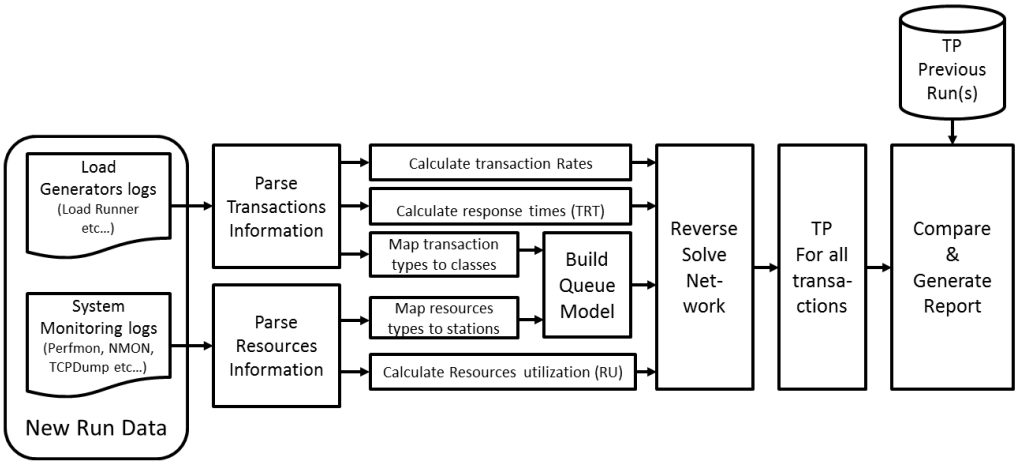

Figure 8: Outline of Proposed System.

4. RELATED WORK

Industry still adopts the manual approach to analyse performance data looking for performance regression [1]. The testing engineer will need to compare TRT, RU of the two releases looking for an anomalous behaviour based on his/her personal judgement. Such manual approach is error-prone and time consuming unlike our proposed approach which can be fully automated. Also, such manual approach requires running both tests with the same load which is time and resources consuming if the new release is expected to run with a different field load. Our approach does not depend on the load as the same TP can be inferred from test runs with different loads.

One of the first approaches to automate the performance regression testing process used statistical techniques [2]. They introduce the use of Statistical Process Control (SPC) charts in performance regression testing. The previous run data (baseline) is used to draw the Lower and Upper Control Limits (UCL, LCL) and the Centre Line (CL). The new runs data is then tested against those control limits and a violation ratio is calculated. If this violation ratio exceeds a certain pre-determined value a performance regression anomaly is declared. Such an approach still requires running the new release test with a similar load to the previous release. So the same load issue of the manual approach above still exists making our approach more favourable.

5. CONCLUSIONS

In this research work, we have proposed the usage of TP to detect performance regression anomalies. We have verified that a TP can realistically represent the transaction performance characteristics. The TP only changes when the software changes and is independent of variations in the load. Then we have also provided an overview of our proposed system along with implementation details. During the experiment, our system has been tested on an open source application and we plan to extend the evalation setup to a real enterprise application. We also plan to study some more aspects of the solution, mainly the software contentions.

6. ACKNOWLEDGMENTS

Supported, in part, by Science Foundation Ireland grant 10/CE/I1855.

7. REFERENCES

[1] J. Z. Ming. Automated analysis of load testing results. In *Proceedings of ISSTA*, New York, NY, USA, 2010. ACM.

[2] T H. D. Nguyen. Using control charts for detecting and understanding performance regressions in large software. In *Proceedings of ICST*, Washington, DC, USA, 2012. IEEE Computer Society.

[3] W. R. Adrion, M. A. Branstad, and J. C. Cherniavsky. Validation, verification, and testing of computer software. *ACM Comput. Surv.*, 14(2):159–192, June 1982.

[4] F. M. Bereznay. Did something change? using statistical techniques to interpret service and resource metrics. In *Proceedings of CMG*. Computer Measurement Group, 2006.

[5] T. Gao, Y. Ge, and J. Ni G. Wu. A reactivity-based framework of automated performance testing for web applications. In *Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on*, pages 593–597. IEEE, 2010.

[6] J. Walrand. *An introduction to queueing networks*, volume 165. Prentice Hall Englewood Cliffs, NJ, 1988.

[7] L. Grinshpan. *Solving Enterprise Applications Performance Puzzles*, pages 5–57. John Wiley and Sons, Inc., Hoboken, New Jersey, 2012.

[8] The Apache Software Foundation. Apache jmeter. http://jmeter.apache.org/, 2013.

[9] S. Ghaith, M. Wang, P. Perry, and J. Murphy. Profile-based, load-independent anomaly detection and analysis in performance regression testing of software systems. In *17th European Conference on Software Maintenance and Reengineering (CSMR'13)*, Italy, 2013.

[10] MyBatis. Jpetstore 6. http://www.mybatis.org/spring/sample.html/, 2013.