Software Contention Aware Queueing Network Model of Three-Tier Web Systems (Work-In-Progress)

Shadi Ghaith, Miao Wang, Philip Perry and Liam Murphy
School of Computer Science and Informatics
University College Dublin, Ireland
shadi.ghaith@ucdconnect.ie

ABSTRACT

Using modelling to predict the performance characteristics of software applications typically uses Queueing Network Models representing the various system hardware resources. Leaving out the software resources, such as the limited number of threads, in such models leads to a reduced prediction accuracy. Accounting for Software Contention is a challenging task as existing techniques to model software components are complex and require deep knowledge of the software architecture. Furthermore, they also require complex measurement processes to obtain the model's service demands. In addition, solving the resultant model usually require simulation solvers which are often time consuming.

In this work, we aim to provide a simpler model for three-tier web software systems which accounts for Software Contention that can be solved by time efficient analytical solvers. We achieve this by expanding the existing "Two-Level Iterative Queuing Modelling of Software Contention" method to handle the number of threads at the Application Server tier and the number of Data Sources at the Database Server tier. This is done in a generic manner to allow for extending the solution to other software components like memory and critical sections. Initial results show that our technique clearly outperforms existing techniques.

Keywords

performance models, performance prediction, web applications, software contention

1. INTRODUCTION

Using modelling to predict application performance under various possible hardware (and software) configurations is becoming widely used in capacity management processes [1] [2]. It saves time and also removes the need to physically build and evaluate a number of possible alternative systems. A Queueing Network Model (QNM) which represents the various hardware and software resources of the system can be solved by either analytical techniques (which are fast but are limited to simple QNMs) or by simulation solvers (which can be applied to more complex QNMs but require a much longer time to run the model).

The work done by Kounev et al [2] shows that building QNMs to represent hardware resources only (such as CPU) is relatively simple and fast. Yet, they show that the calculated response times are much lower (more than 30%) than the real ones measured by conventional load testing. This large error is due to the absence of modelling contentions caused by software resources such as the limited number of threads [2]. This deviation can be offset by increasing the suggested hardware requirements which can result in wasted hardware resources. Some techniques were introduced to account for Software Contention (SC) [3] [4] [5] but suffer from increased complexity in the QNM and additional effort to measure the service demands as well as the increased time required to use simulation solvers. The time factor of performance prediction is important as simulation solvers may take hours for a medium size software system which is needed to be repeated many times varying hardware and software configuration options [6].

In his work [7] Menasce introduced a two layered iterative technique to model SC. The technique is simple and time efficient, but it is difficult to apply it to complex systems models. In addition it requires some specific instrumentation of the application code to measure various service demands required by the software modules on each hardware resource. In this paper, we aim to extend the technique proposed in [7] to model a general three-tier web system such as the one introduced in [2]. We also aim to reduce the collection difficulties encountered when measuring the service demands of the QNM. In doing so we expect to achieve the following objectives:

1. Generate a generic model for three-tier web systems which can account for contention caused by software resources. In particular, in this paper we consider the number of threads of the Application Server (AS) and the number of database Data Sources (DS).

2. The model is simple enough to be solved using time efficient analytical solvers.

3. The model is extendable to handle contention caused by memory management and critical sections.

Our work can improve the capacity planning and management process by enhancing its accuracy and significantly reducing the time required to solve the model. It will also be useful in other processes that rely on multiple solutions of

ICPE'14, March 22–26, 2014, Dublin, Ireland.

http://dx.doi.org/10.1145/2568088.2576760.

QNMs such as the performance regression testing technique [8] [9].

2. OVERVIEW

2.1 QNM of a Three-Tier Web System

A typical three-tier web system consists of the following three components [10]. First, the client component which usually contains client software such as the web browser that takes the user input and communicates with the server side over the network. After the server side performs the necessary functions, the client will then present the response to the user transaction and the user will launch the next transaction after thinking for a period of time known as Think Time (TT). Given that the client software is usually run on individual user machines, its hardware (and even software) service demands are very low and hence it is usually just represented by a delay station denoted TT. The second component is the AS which is usually composed of an application platform, such as the JEE container, over which the application is deployed. The platform usually assigns each request to a thread to allow for serving multiple clients at the same time. The number of available threads is limited [10] and so it is a key factor that affects the application performance. The third component is the Database Server (DBS) on which a relational database software (such as Oracle) is deployed and contains the application specific data. Before the AS communicates with the DBS (to store and read data) it requests a DS from the DBS to handle this communication. The number of DS's available from the DBS is also limited [10] and is another key factor for the performance of such systems. The three-tier web system is usually represented by a hardware QNM such as the one shown at the bottom of Figure 1.

2.2 Two-Level Iterative Queueing Modelling of Software Contentions

The work in [7] by Menasce is one of the simplest approximations to handle SC. Yet, it provides very good results, as shown in the paper and as we found in our experiments. This approach requires an iterative solution of two QNMs: One is the Hardware Queueing Network (HQN) and the other is the Software Queueing Network (SQN).

The HQN is a typical QNM such as the one shown at the bottom of Figure 1, where hardware resources such as CPU and hard disk are modelled as load independent queueing stations and the user TT node is modelled by a delay station. The SQN also contains the TT station as well as a set of nodes each of which represents a certain software module. Each software module either causes no queueing in which case it is represented by a delay station, or it can cause queueing (such as a critical section) in which case it is represented as a load independent queue station. For example, the critical section is represented by a station that has one processing unit and a queue. A simple SQN is shown at the top of Figure 1.

Each software module in the SQN has a service demand on each hardware node in the HQN. These are measured using a single user test by instrumenting the software code. The total service demand on each module in the SQN is calculated by the summation of all service demands for that module on each station in the HQN. While, the total service demands for each hardware station in the HQN is the summation of all service demands for that station caused by each software module in the SQN.

The SQN is first solved with the initial (single user) modules' total service demands using the total number of users accessing the system. Then the number of blocked users is calculated by finding out the number of users in each queue station within the SQN. The HQN is then solved with the total number of users excluding the users in the queues of the SQN and the total service demands of the hardware stations. After solving the HQN, the service demand of each module in the SQN is set to the sum of the fraction of the residence time (queueing and serving time) of each station in the HQN proportional to this module contribution to the original total service demands of each HQN station. Then the above is repeated again by solving the SQN with the new service demands and solving the HQN with the total number of users excluding the blocked users. This continues until the number of blocked users is reasonably stable.

The approximation above has the following drawbacks:

1. Finding the service demands per software module for each hardware resource is complicated. It requires instrumentation at the code level on the border of each software module and on each access to hardware resources within each module. This requires a detailed investigation of the code which will not be always possible, especially in industrial environments. On the contrary, the hardware service demands used for modelling the hardware resources, ignoring the SC, can be easily measured with a single user test [5].

2. The current approach has a single level (depth) of SC modules. But, in practice, and taking the example of the critical section in the AS where its code may access the database and such a call will suffer from another SC, i.e. the DS. Hence, a new level of SC is possible and needs to be modelled.

3. METHODOLOGY

In this section, we will extend and apply the core idea of the Two-Level Iterative Queueing Modelling of Software Contention to the three-tier web system (both introduced in the previous section). We start by modelling the number of threads in the AS in a way that simplifies the service demands measurement step. Then we show how we can work around the nesting of the software resources by converting the HQN and SQN to product form QNMs.

3.1 Applying Two-Level Iterative Queueing Modelling to Number of AS Threads

As discussed in Section 2.1, the AS dispatches each new job to a new thread which executes the required server module. During its execution, the thread accesses the hardware resources (CPU and hard disk) on both the AS and the DBS (we describe the DS contention in the next section). Given that we approximate that most of the AS code is executed within the thread (except the ignorable demands of the dispatching module), the SQN will only have one module. This module can be represented by a single-queue multi-server station, where the number of servers equals the number of available threads in the AS. Figure 1 shows both the HQN and the SQN in this case. Hence, the service demand for that single software module equals the sum of service demands over all the hardware resources both at the AS and

the DBS. This simplification allows us to use the service demands measured by a single user test [5] on all system hardware resources.

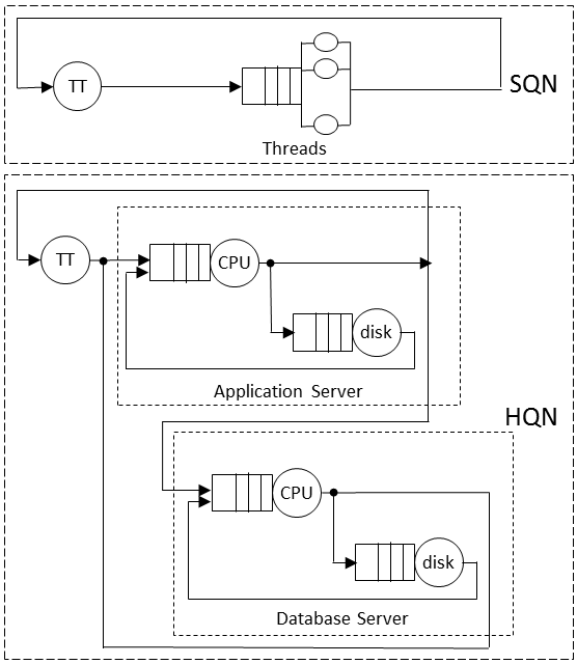

Figure 1: SQN-HQN of a Simple Three-Tier Web System with Application Server Threads.

The SQN can be solved analytically by modelling the single-queue multi-server station in the SQN as a load dependent station as detailed in [11]. Analytically solving the HQN with the non-blocked users is also straightforward. The performance of this solution is much better than the simulation techniques used to model the number of threads such as the Finite Capacity Region (FCR) of the Java Modelling Tool (JMT) [4] described in the Related Work section. The number of iteration was found to be small for such systems (i.e. less than 15 in the case study below).

If we omit the handling of the number of AS threads, the same technique can be used to model the DS, except that the service demands of the module in the SQN is the sum of the DBS CPU and hard disk. While the AS hardware stations, CPU and hard disk, will appear in the SQN and handled in a similar way to the TT station, i.e. they will not contribute to the DS module service demands in the SQN.

Trying to model both the AS threads and the number of DS's in the DBS results in a nested SQN modules, making the entire technique not viable in its current form.

3.2 Tiered Two-Level Iterative Queueing Modelling of Software Contention

The HQN in Figure 1 shows that the AS makes multiple calls to the DBS. The DBS returns the execution to the AS after each call and finally the AS returns back to the user (TT station). Given that the number of DS's on the DBS is also limited, the SC effect happens at both tiers of the system resulting in a complex nested SQN. To rectify this, we consider the approximation used in [2] by Kounev et al which approximates the QNM of a system similar to the HQN of our work with a model in which each station is visited only once. This makes the HQN simpler as the AS will make one call to the DBS which will serve the job and return back to the TT station. This simplification is validated also with their results. We provided the theory behind this approximation based on the BCMP approximation [12] in our previous work [8].

In the same manner as explained for the HQN, and by applying the same theoretical background [8], we believe that the same approximation is valid on the SQN level. This means that we assume the job visits each software module (the AS threads and the DBS DS's) only once. The SQN contains two single-queue multi-server stations each of them is visited once as shown in Figure 2. Each software module only relies on the hardware service demands within the same tier i.e. the threads module only relies on the CPU and hard disk service demands of the AS and the DS module only relies on the CPU and hard disk service demands of the DBS. The blocked users are calculated to include users waiting in the queues of both single-queue multi-server stations representing the threads and DS modules. The SQN and HQN are iteratively solved in the same manner as explained above. This approximation can be generalized to n-tier systems and has been validated against the three-tier web system described in this paper.

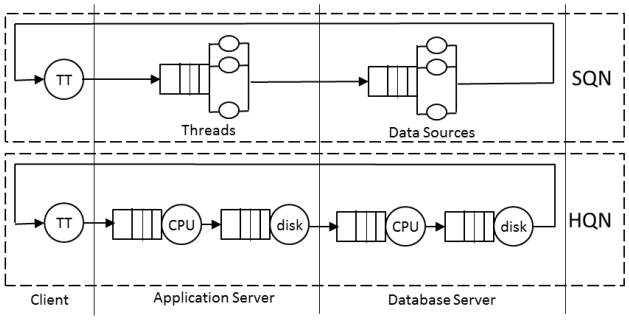

Figure 2: SQN-HQN of a General Three-Tier Web System with Tiered Software Modules (Threads and Data Sources).

4. VALIDATION

We verified the method developed in this paper on the TPC-W workbench application deployed on the IBM WebSphere Application Server and IBM DB2 Database Server with 500 users and three transaction types, as follows:

- Performed a load run using a load generator and measured each transaction response time and the resources utilization (CPU and hard disk on AS and DBS).
- Measured the various transactions service demands on all hardware resources by a single-user test.
- Predicted the response time of each transaction and resources utilization (CPU and hard disk on AS and DBS) by solving the QNM without taking the SC into account (i.e. similar to [2]).
- Used the technique presented in this paper to predict the same values.

- Solved the HQN model shown in Figure 2 using the FCR feature of the JMT.

We had the following observations for the transaction type with the highest response time:

1. The response time calculated using the normal QNM (i.e. not taking the SC into account) is 18% lower than that measured with the load test. While the resource utilizations are within 4% range. This confirms the results achieved by Kounev et al [2].

2. The response time calculated using the presented tiered HQN-SQN (i.e. taking the SC into account) is 6.5% lower than that measured with the load test. While the resource utilizations are within 3% range. That is, we had an improvement of about 300% for the accuracy of response time prediction compared to [2].

3. The presented technique converged within 2 seconds and needed 13 iterations, while the FCR approach required just over 2 minutes. That is, the presented tiered SQN-HQN technique takes around 10% of the time required by simulation.

5. RELATED WORK

Modelling three-tier web systems has been explored previously, particularly by Liu et al [10]. They built a QNM based on the architecture of typical three-tier web applications. The model incorporates the threads at all three tiers and then a solution is approximated by using the MVA technique. The solution augments the hardware and software components within the same elements of the QNM, which makes it hard to use such a model in capacity management where it is always required to modify hardware and software parameters separately to achieve the optimum configuration.

Layered Queueing Networks (LQN) were introduced to handle the SC issue [3]. LQN is based on software resources representing various software operations as the main nodes of the QNM, and hardware resources (such as CPU) as leaf nodes. Software resources call each other forming a multi-layered QNM. It is assumed that the service demands for each software resource on each hardware resource are known (i.e. either estimated by developers or measured by instrumenting the code). This assumption makes the approach difficult to adopt in capacity management processes where access to the code is usually not available. Even if code access is available, instrumenting the code to measure various service demands requires deep knowledge of all parts of the code, such knowledge is rarely available in large applications which are usually developed by multiple teams each responsible for just one software module. In addition, for any LQN with moderate complexity, the expansive simulation techniques are the only possible way to solve them.

Existing tools, such as the JMT, introduced some capabilities to model SC such as the FCR feature of the JSIMgraph part of the JMT [4]. The FCR is used to specify the number of jobs within a certain block of the QNM which allows modelling of constraints such as the number of threads and DS's. The performance of JSIMgraph (simulation), specifically when introducing FCRs prevent it from being used in real capacity management projects.

6. CONCLUSIONS AND FUTURE WORK

Current modelling techniques to account for SC require complex models, hard to obtain service demands for and are expensive to solve. Hence, capacity management projects typically rely on hardware models and apply some assumptions and rules of thumbs to account for SC. In this paper we reduced some of the difficulties of the "Two-Level Iterative Queuing Modelling of Software Contention" method by Menasce and applied it to the common three-tier web systems. We achieved a simplification of the model where it is easier to obtain service demands and can be solved quickly utilizing basic analytical solvers. We showed that our technique provides three times more accurate results in 10% of the time compared to other solvers with SC capabilities.

In this short paper, we explained the technique to account for the number of threads and the DS's; in the future we plan to account for other factors, mainly the memory and critical sections. Also, the technique will be verified on more complex systems, like those with clustered servers. In addition, we plan to show the improvement on actual capacity management projects and on other fields utilizing QNMs, such as the regression testing data analysis.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

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

[2] S. Kounev and A. P. Buchmann. Performance modeling and evaluation of large-scale j2ee applications. In *Int. CMG Conference*, pages 273–283, 2003.

[3] J.A. Rolia and K.C. Sevcik. The method of layers. *Software Engineering, IEEE Transactions on*, 21(8):689–700, 1995.

[4] M.Bertoli, G.Casale, and G.Serazzi. Jmt - performance engineering tools for system modeling. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):10–15, March 2009.

[5] Samuel Kounev. J2ee performance and scalability-from measuring to predicting. In *Spec Benchmark Workshop*, page 12, 2006.

[6] CU Smith. *Performance Engineering of Software Systems*. Addison-Wesley, 1990.

[7] D. Menasce. Two-level iterative queuing modeling of software contention. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, pages 267–276, 2002.

[8] S. Ghaith, M. Wang, P. Perry, and J. Murphy. Automatic, load-independent detection of performance regressions by transaction profiles. In *Proceedings of the 2013 International Workshop on Joining AcadeMiA and Industry Contributions to testing Automation*, JAMAICA 2013, pages 59–64, New York, NY, USA, 2013. ACM.

[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)*, Genova, Italy, 2013.

[10] Xue Liu, J. Heo, and Lui Sha. Modeling 3-tiered web applications. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, pages 307–310, 2005.

[11] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 2006.

[12] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM, 22(2)*, pages 248–260, 1975.