Impediments to Analytical Modeling of Multi-Tiered Web Applications

Nilabja Roy, Aniruddha Gokhale and Larry Dowdy
*Dept. of EECS, Vanderbilt University, Nashville, TN 37235, USA
Email: nilabjar@dre.vanderbilt.edu,{a.gokhale,larry.dowdy}@vanderbilt.edu

***Abstract*—Service providers hosting multi-tiered applications require accurate analytical models of the applications they will host for different system management activities, such as capacity planning, configuration management, cost analysis and feedback control. Due to the complexity of real world scenarios, developing accurate analytical models is hard. This paper presents the commonly faced challenges in developing these analytical models that stem from real-world issues, such as excessive system activity, presence of multiple cores or processors, and concurrency management. Presence of multi-tiered applications further compounds the challenges faced. We sketch preliminary ideas based on application-specific and/or domain-specific modeling techniques to overcome these limitations.**

***Keywords*-Multi-tier applications, performance estimation, service deployment.**

I. Introduction

Accurate modeling of multi-tiered web applications is an important requirement for proper system management. These models can be used for a variety of important functions, such as performance evaluation, capacity planning, configuration management, admission control, and cost analysis. Accurate models aid in all of the above functions, which in turn help build better and reliable systems. Unfortunately, developing accurate system models for multi-tiered applications is hard. Consequently, often these models are approximate and may not truly represent the actual application.

Significant prior work exists that uses one or more of analytical modeling techniques for developing accurate models. For example, [1]–[5] have used analytical techniques and profiling to build models of multi-tiered web portals but have not accounted for increased system activity, such as page-faults which occur with increased load. The emerging trend towards multiple processors/cores has also not been considered by most of these works.

An additional issue concerns databases. Most prior work fail to account for database optimizations, *e.g.*, optimizations to handle similar (*i.e.*, overlapping) queries that run concurrently. Finally, resource allocation, which is a key issue in capacity planning, has been investigated only at the granularity of an entire tier-level, however, this coarse level of granularity is insufficient in minimizing the number of and efficiently using resources in the context of modern multi-tiered systems that are made up of finer-grained components.

This work has been supported by NSF SHF/CNS Award #0915976

This paper identifies some common scenarios that occur in realistic production environments and highlights the limitation in modeling these scenarios using modeling techniques developed in recent work on multi-tiered web applications. The cases illustrated in this paper clearly suggest the need for application-specific and/or domain-specific models, which are modified versions of traditional models catered to a specific scenario. The paper subsequently sketches preliminary ideas on a process that combines profile data and analytical modeling to develop more accurate models.

II. Challenges in Analytical Modeling of Multi-Tiered Applications

This section presents three scenarios that occur commonly in multi-tiered web applications under realistic production environments. The scenarios are shown in the context of the Rice University Bidding System (RUBiS) [6] which is a prototype application of an ebay like auction site. The experiments shows how modeling techniques used in recent research for multi-tiered web applications fall short in coming up with good estimates of system parameters that in turn limit the accuracy of the developed models.

A. System Activity at Heavy Load

Recent work [1]–[4] use queuing models to estimate performance of multi-tiered applications. But such models can introduce errors in estimating performance accurately when system activity, such as context switching and page faults, increases. To highlight this limitation and understand why these models fall short, we developed a similar queuing model and used mean value analysis algorithm to estimate response times observed by clients of RUBiS as system activity keeps increasing. We then compared these estimates against experimentally-measured response times for the `SearchByRegion` service offered by RUBiS as shown in Figure 1.[1]

Figure 1 shows that the two curves remain close to each other at low utilization for a small number of clients. However, as the number of clients increases (which in turn increases system activity), the disparity between the experimental and analytical results becomes significant.

This disparity is attributed primarily to the service demand used by the model, which is measured by executing only a single job [7] when there is minimal load. The service demand

[1]Similar comparisons were made for services like `SearchByCategory` but are not shown due to lack of space.

in the model does not account for excess system activity that takes place with a large load. This excess system activity needs to be taken into account in the model which potentially could be done by inflating the service demand as load increases. Thus our approach is to model the service demand as a function of the load and use it in the MVA calculation. This will consider the increased system activity and will improve the model estimation.

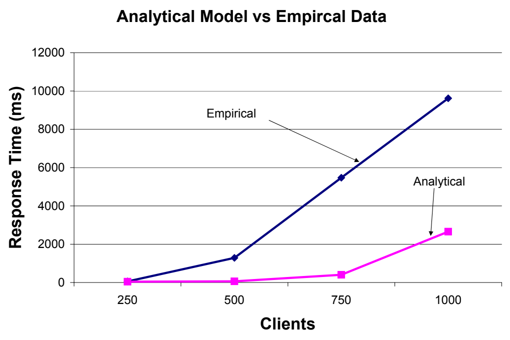

Fig. 1. **Comparison of Analytical vs Empirical Data**

B. *Multiprocessor effects*

With increasing availability and use of multi-processors and multi-cores for multi-tiered applications such as web portals, existing closed queuing network models [1], [4], [8] must now incorporate support for multiple servers. Although existing closed queuing networks can be solved efficiently using the mean value analysis (MVA) algorithm, accounting for multiple-server models requires computing the probability mass function of the queue sizes for each server. The mass function is used within MVA to calculate the total expected waiting time that a customer experiences on a server. This approach, however, significantly increases the complexity of the MVA solution. A potential solution has been suggested in [9] where a correction factor is used that estimates the server waiting time for multi-processors. But it is also suggested that this correction factor will be application and hardware specific. Thus there is the need to somehow measure such a correction factor in order to use it.

C. *Dependent Transactions*

Multi-tiered applications often comprise databases, which means that user invocations will result in database transactions that operate on related data. Modern day databases use different kinds of optimizations for multiple overlapping queries. These optimizations include caching of intermediate data [10] or using heuristics [11]. Such optimizations could result in unpredictable performance for concurrently executing and overlapping queries, which makes it hard to estimate the performance of multi-tiered applications at design-time. Nonetheless, it is important to model these effects since they can play a dominant role in the overall performance of the multi-tiered application. Note that such effects will be primarily application- and query-specific and can be modeled with the help of extensive profiling.

We highlight this challenge using a specific example in RUBiS and then describe how to model such behavior based on profiled data. We focus on two types of browsing queries in RUBiS: `SearchByRegion` and `SearchByCategory`. Both these queries work on the same table named *items*. However, `SearchByRegion` also uses an additional *users* table. We observed that when the two different queries execute all by themselves (*i.e.*, no concurrency), the `SearchByCategory` is much faster than the `SearchByRegion` as shown in Figure 2 (Lines `SearchByCategory` and `SearchByRegion`).

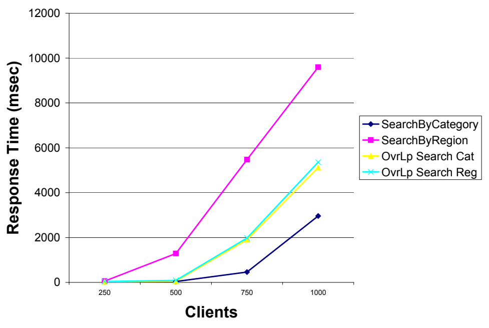

Fig. 2. **Concurrent Overlapping Queries Have Similar Response Times**

However, when the two queries execute concurrently on the same database, their individual response times become almost the same (Lines `OvrLp Search Cat` and `OvrLp Search Reg`). With a low number of clients the response times are somewhat different but as the number of clients increase (and hence the overlapping queries increase), the response times experienced by all the clients are almost the same. We surmise that this behavior is attributed to some optimization in the databases when multiple overlapping queries run concurrently. Intuitively it seems that since both queries are using the same table, there is some locking due to a software lock such as a semaphore or a mutex.

Although we can only surmise the exact cause of the behavior shown in Figure 2, we were able to reproduce this behavior across multiple experiments. Figure 3 shows a closed queuing model where the processor on which the query executes is modeled as a queue.

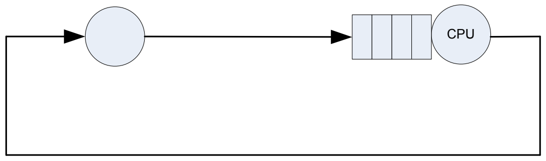

Fig. 3. **Traditional Closed Queuing Model**

An additional delay server models the client think time. This is the normal way in which such database queries can be modeled using queuing networks. Here we assume that only one resource is being used, which is the CPU. This model also assumes there is no internal blocking and thus works properly as long as there are no optimizations.

To model the blocking effect stemming from the use of software locking for overlapped query optimizations, we introduced another resource on which each query waits for some time. This resource could be thought of as a software lock where each query waits to grab the lock before accessing the table. Thus, the newly introduced resource simulates the blocking time spent by the queries when executed concurrently. Figure 4 shows the enhanced queuing model which introduces an extra queue compared to Figure 3.

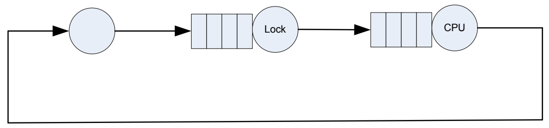

Fig. 4. **Additional Queue to Model Software Blocking**

Having developed the model, the next step is to estimate how long a query waits on the lock. Intuitively it appears that a query of one type will wait for the query of the other type to finish. Thus, it will wait for the length of the time equal to what the other query takes to finish its job. In other words, the waiting time for the query under consideration is equal to the service demand of the other concurrently running query on that processor. This might not be the perfect way to estimate the time spent on the lock and the actual time depends upon the optimizations in the database. But our objective is to build an approximate model. Thus, we assign the service demand of the query on the lock to be equal to the service demand of the other query on the processor.

Capacity planners must identify sources of such database optimizations in their applications and use the suggested technique of modeling the extra locking queue to obtain accurate performance estimates. An alternative way of modeling this scenario would be to consider the two job classes as a single one. But then the parameters of each separate job class will not be available and cannot be used if they are required.

The model is validated and Figure 5 shows the performance estimation of the inter-dependent queries. It can be seen from the figure that our model accurately estimates the effect of the inter-dependent queries.

III. Conclusion

This paper presented real-world scenarios in the domain of multi-tiered web applications to highlight how prevalent modeling techniques cannot adequately model these applications and accurately estimate their performance. Such scenarios need domain specific models which can modify and extend traditional models for the specific scenario. In future work, we are investigating the following issues in order to develop more accurate analytical models:

- How does system activity in a machine increase with load? Is the increase linear?
- When does system activity saturate? Does it depend on the set of jobs of running?

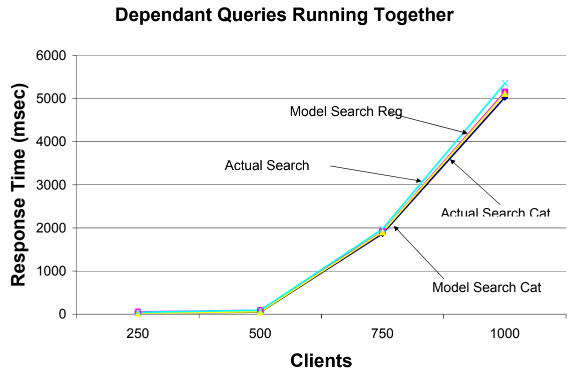

Fig. 5. **Model of Inter-Dependent Queries**

- Does the speed of a processor affect the correction factor (Section II-B)?
- Will the hardware architecture affect the correction factor?

Answers to the above questions will enable us to develop more robust application/domain specific analytical models which in turn can be used in system management activities.

References

[1] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, “An Analytical Model for Multi-tier Internet Services and its Applications,” *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 291–302, 2005.

[2] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, “Dynamic provisioning of multi-tier internet applications,” in *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, 2005, pp. 217–228.

[3] Q. Zhang, L. Cherkasova, G. Mathews, W. Greene, and E. Smirni, “R-capriccio: a capacity planning and anomaly detection tool for enterprise services with live workloads,” in *Middleware ’07: Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*. New York, NY, USA: Springer-Verlag New York, Inc., 2007, pp. 244–265.

[4] Q. Zhang, L. Cherkasova, N. Mi, and E. Smirni, “A regression-based analytic model for capacity planning of multi-tier applications,” *Cluster Computing*, vol. 11, no. 3, pp. 197–211, 2008.

[5] C. Stewart and K. Shen, “Performance modeling and system management for multi-component online services,” in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2 table of contents*. USENIX Association Berkeley, CA, USA, 2005, pp. 71–84.

[6] C. Amza, A. Ch, A. Cox, S. Elnikety, R. Gil, K. Rajamani, and W. Zwaenepoel, “Specification and Implementation of Dynamic Web Site Benchmarks,” in *5th IEEE Workshop on Workload Characterization*, 2002, pp. 3–13.

[7] D. A. Menasce, L. W. Dowdy, and V. A. F. Almeida, *Performance by Design: Computer Capacity Planning By Example*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[8] N. Roy, Y. Xue, A. Gokhale, L. Dowdy, and D. C. Schmidt, “A Component Assignment Framework for Improved Capacity and Assured Performance in Web Portals,” in *Proceedings of the 11th International Symposium on Distributed Objects, Middleware, and Applications (DOA’09)*, Nov. 2009, pp. 671–689.

[9] R. Suri, S. Sahu, and M. Vernon, “Approximate Mean Value Analysis for Closed Queuing Networks with Multiple-Server Stations,” in *Proceedings of the 2007 Industrial Engineering Research Conference*. Citeseer, 2007.

[10] A. Safaeei, M. Kamali, M. Haghjoo, and K. Izadi, “Caching Intermediate Results for Multiple-Query Optimization,” in *IEEE/ACS International Conference on Computer Systems and Applications, 2007. AICCSA’07*, 2007, pp. 412–415.

[11] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe, “Efficient and extensible algorithms for multi query optimization,” *SIGMOD Rec.*, vol. 29, no. 2, pp. 249–260, 2000.