Model-Driven Performance Evaluation of Web Application Portals

Nilabja Roy and Douglas C. Schmidt
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37203, USA
{nilabjar, Schmidt } @ dre.vanderbilt.edu

Introduction

Emerging trends and challenges. The advent of web-based applications, such as shopping, social networking, photos, videos, music, gaming, and chat, are increasing the popularity and accesibility of the Internet. There is also an increasing focus on application integration platforms, such as Sun's Java Composite Application Platform Suite, Facebook's Application Platform and Oracle Application Development Framework…, where a single portal can provide many services. These integrated web sites are referred in this paper as *web application portals*, which are Internet sites that provide multiple services to users. For example, users of social networking sites, such as Facebook and MySpace, upload recent photos and videos, exchange messages and chat with each other, and play online games with friends.

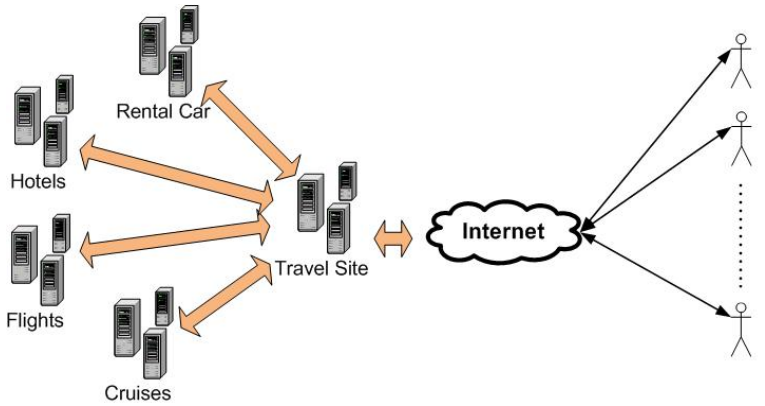

Figure 1: A Travel Site That Provides Interface to Hotels, Flights, Rental Cars, and Cruises

Figure 1 shows the architecture of web application portals, such as www.priceline.com or www.hotwire.com, that help users build vacation packages with choices for flights, hotels, rental cars, and cruises. Users submit requests to the portal, which in turn contacts various service providers for each service and forwards responses to users. Web application portals should be scalable to support a variety of services—and the large number of customers accessing the services simultaneously. The scalability requirements for these sites typically grow as the number of service providers increases.

Competition between providers of web application portals is also growing. Providers have different marketing differentiators and focus on different features and services. For example, IMDb (www.imdb.com) focuses on detailed movie information, whereas yahoo-movies (www.movies.yahoo.com) focus on movie ratings and theater timings. Although both features are useful, they serve different sets of users with different interests. With multiple services, differentiated focus, and large customer bases, such web portals are a complex composition of different sources of non-determinism, which complicates the evaluation of web application portal performance.

Regardless of the features and services offered by web application portals, successful providers must ensure key quality of service (QoS) properties, such as end-to-end request response time, system availability, and scalability. For example, online travel sites, such as Expedia or Orbitz, aim to provide the best travel deals to customers within a reasonable time frame, which may vary from person-to-person and from application-to-application. In particular, user submitting travel queries may be willing to wait longer for the best deals than users searching for a phone number. Given the proliferation of web-based application portals in the Internet, users who are not satisfied with one provider can often switch to alternative providers, which incentivises providers to enhance the QoS of their web application portals.

To remain viable in today's competitive environment, therefore, developers and administrators of web application portals must address a number of issues, such as (1) what software/hardware architecture will provide the necessary performance at scale, (2) how should the software be modularized, (3) how can applications and systems be configured to ensure high performance, (4) how will a particular application design perform under particular usage patterns, and (5) how many (and what type) of machines are required to achieve the required performance. Addressing these issues can help developers of web application portals design systems that can provide the required QoS for current and planned usage models.

In many cases, however, web portal application performance is evaluated late in the system lifecycle, i.e., after the software is developed and deployed on the target hardware. At this point, it is hard to correct mistakes in the system design that yield poor performance. What is needed, therefore, are techniques that analyze and predict the performance of such web based applications earlier in the lifecycle and can help guide developers choices of alternative portal designs. Such techniques need not provide exact predictions, but they do need to accurately capture general trends and provide quantifiable numbers that enable developers to select the most appropriate alternative designs.

To ensure that web portal applications meet their QoS requirements developers and administrators must address various issues. For example, application performance depends on multiple factors, such as the underlying middleware, operating system, third party-tools, and hardware, each of which must be properly understood and quantified. Performance also depends on the hardware used to deploy applications. Developers who deploy applications therefore need techniques and tools that can provide quantifiable numbers to help choose the right configurations. Likewise, administrators need techniques and tools that can provide quantifiable numbers that help choose the right configuration and deployment options.

A common technique for predicting the performance of applications in large-scale systems involves the creation and evaluation of analytical and/or simulation models [5]. Analytical modeling involves creating a mathematical model of the system that can predict application performance under various conditions. Simulation modeling involves creating a model of the actual system, implementing the model using a simulation package (such as C++Sim), executing the model, and analyzing execution output. Developers of web application portals face the following two challenges, however, when trying to create and evaluate analytical and/or simulation models in their domain:

- The performance of large-scale, multi-tiered web application portals are not accurately predicted using conventional analytic and simulation modeling techniques, such as processor usage measurement, simple queuing modeling, or exponential arrival rates.
- A large gap has historically existed between the performance problems of interest to domain experts and the expression of these problems in conventional performance modeling technologies. For example, admission control policies in web servers, multi-threaded software contention etc. cannot be directly analyzed using conventional queuing theory concepts.

Solution approach → Model-driven techniques and tools that can predict web application portal performance accurately and help close the gap between domain-oriented performance problems and conventional performance modeling technologies. Model-driven tools based on the OMG's Model-Driven Architecture (MDA) technologies [3] provide a structured process for software development and evaluation based on specifying and transforming platform-independent models (PIMs) into platform-specific models (PSMs) that capture key performance attributes, such as execution time and processor utilization [1]. The use of model-driven tools enables the refinement of performance models throughout the software development life cycle to predict performance accurately.

This book chapter explores the current research in the area of analytic and simulation modeling and describes how new techniques (such as regression based modeling and innovative workload prediction) can be used to accurately evaluate and predict modern-day web application portal performance. It also discusses how they can be combined with model-driven tools and applied to web application portals. These model-driven techniques and tools provide developers and administrators with the flexibility to tune key model parameters to enable detailed sensitivity analysis. For example, the scalability of an admission control policy in a web server can be evaluated by increasing the number of concurrent client requests until the response time reaches a certain upper bound, which accurately estimates application behavior as the number of clients increase.

The chapter covers the general steps to follow when conducting a model-driven performance evaluation, which are shown in Figure 2 and summarized below.

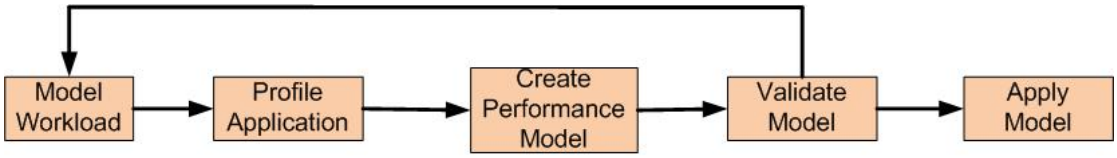

Figure 2: Steps to Use Models to Improve Web Application Portal Performance

Workload modeling, which entails modeling the incoming workload to the system. Workload is affected by various factors, such as time of day, time of month, holiday season, etc [6]. A web portal application will have a variety of workload, such as holiday crowd, evening crowd, lunch time requests, etc. It is hard to characterize these different workload patterns. Modeling techniques, such as customer behavior modeling, fitting request data to distributions, correlations, and auto-correlations, can be used to accurately model the modern day workload pattern of a web application portal. The initial workload modeling parameters such as customer behavior modeling can be input as a part of PIM of the MDA process. These parameters can be subsequently transformed into real values (such as the number of customers) in the steady state of the PSM.

Profile application, which profiles the individual application components during unit testing. In early stages of an application's lifecycle for example at the PIM level, usage data can be approximated using data for similar components from previous projects, operational experience, or representative workload models. The objective at this point is to build an approximate model of the application's performance, not an exact one. Later at the PSM level, actual profiled values can be inserted which will provide a more accurate model-driven performance evaluation.

Create a performance model, which can be a simulation model or an analytical model, each of which has their pros and cons. If a system can be modeled using analytic techniques, the result is accurate. But in most cases like web application portals exact analytic models cannot be built and approximate models need to be built. This will obviously give approximate results [5]. Simulation can be used here and might give better results but depends upon the number of iterations performed since it uses a random number generator and needs quite a few iterations to get a nice set of data that represents the distribution functions used to model the various random variables. Both simulation and analytical models can be generated from PIM or the PSM level. The models at the PIM level will be approximate while those at the PSM level will be more accurate.

Validate model, which evaluates both simulation and analytical models against actual application execution traces after the model is built. If there is a large discrepancy, the workload model must be revisited both at the PIM and the PSM level and the earlier steps repeated, as shown in Figure 2.

Apply model, which can use validated models to help guide application configuration decisions by developers and administrators. For example, various alternative configurations can be considered to determine which architectures to choose. This can be added as a part of an interpreter which can be used both at the PIM and the PSM level.

Outline of the Rest of the Proposed Chapter

Section 2: Applying MDA to Performance Evaluation. This section discusses the technical challenges and solution approaches associated with transforming MDA models to corresponding performance models that can be evaluated using the analytical and/or simulation techniques described in this chapter [9,10]. The main points that will be pursued include:

- Model transformations at each level from PIM to PSM
- Auto-generation of analytical and/or simulation performance models at each stage of the MDA process
- The input and output expected for the performance models at each level
- Representation of the performance models.

Auto-generation of performance models will require techniques (such as abstract interpretation [2]) that can help by parsing code and determining the stack depth of application at different points. Other techniques

such as bytecode parsing will also be discussed. The input to the model and output produced will vary depending on what level the model is in. Obviously a model at the PSM level will be more accurate than a corresponding one at the PIM level. The performance models can be represented using UML[4]. The pros and cons of this approach will be discussed.

Section 3: Workload Modeling. This section summarizes the fundamentals of workload modeling and surveys the latest papers and research in this area, including:

- Fitting statistical distributions to model incoming load on an application
- Customer behavior modeling to quantify load on for each service
- Approximation techniques to manage large user base
- Abstraction of the main features of a workload

Workload modeling can be usd to understand and quantify web application portal load, which consists of customer request arrivals and their resource requirements Customer request arrivals can be recorded from web server logs and resource requirements can be obtained by profiling the live system and gathering data. Using them, both arrival and resource usage can be characterized using probability distributions to create the workload model. The section will show how conventional techniques (such as distribution fitting) can be applied to evaluate certain web application portal performance challenges, such as estimating the arrival of chat session requests. In addition, this section will discuss techniques (such as data filtering) that can be used to clean the noise and errors in measured data and will ensure improved workload models.

Section 4: Web Application Portal Profiling. This section describes various profiling techniques that can be used to measure web portal application data, including:

- Compiler-based instrumentation
- OS- and middleware-based profiling
- VM-based profiling,
- Hardware-based profiling

The pros and cons of these different techniques will be evaluated and we will recommend when to select one technique versus another. Recent research on profiling will also be discussed in this context.

This section will also present a detailed discussion on data analysis techniques that can be applied on the data measured via profiling. Since web application portals have many sources of non-determinism the data gathered via profiling contain inaccuracies, such as high variance or outliers in processor usage measurement…. Statistical techniques can be applied to prune away key elements of non-determinism in the data. This section will also survey recent research on applying such techniques in the area of web application portals.

Section 5: Model-driven Performance Evaluation. This section discusses analytical and simulation modeling that can be used to predict web application portal performance. Analytical modeling techniques include:

- Queuing modeling
- Petri net based modeling
- Fluid flow models based on simple workload
- Models based on software architecture

Simulation modeling techniques include:

- Simulating queuing models and petri nets
- Models based on application flow

Both techniques will be compared and contrasted and instances of use of one versus the other will be shown in the context of web application portals. Model validation will also be discussed as a part of this section. This section will also discuss recent research targeted towards models that predict web portal application performance, ranging from complicated queuing models to simple fluid flow based models. Some models focus on component-based applications, whereas others focus on tier-based applications. The pros and cons of various approaches will be discussed and evaluated.

Section 6: Applying Model-driven Performance Evaluation in Practice. This section describes how analystic or simulation models can be used to benefit application developers and administrators [7,8]. Op-

timization techniques that can be used to determine optimal application configurations will be covered, along with heuristics that help determine the best configuration for the application for a particular objective such as throughput maximization. The section will also discuss heuristics that can determine near-optimal application configuration without entailing the performance degradation of an optimal algorithm. We will cover web portal application strategies for dynamic capacity planning during sporadic bursty loads, web server admission control, and component placement in service-oriented web portal applications.

Section 7: Concluding Remarks. Creating accurate performance models of web application portals is hard and must consider various factors, such as workload characterization, resource usage, and resource contention. This section will summarize lessons learned (both pro and con) from applying model-driven techniques and tools to model the performance of web application portals and recommend practical strategies that can be used to model production web application portals. The discussion will focus on how the techniques presented in the chapter can be leveraged to predict application performance throughout the lifecycle of a web application portal, as well as to configure and deploy applications scalably in web application portals.

References

1. Cortellessa, V. "Software Performance Model-driven Architecture," Proceedings of the 2006 ACM Symposium on Applied Computing, pp., 1218-1223 (2006).
2. Regehr, J., Reid, A., and Webb, K. "Eliminating Stack Overflow by Abstract Interpretation." *Transactions on Embedded Computing Systems*, 4(4):751–778, 2005.
3. Miller J.(editor), Model-Driven Architecture Guide, omg/2003-06-01 (2003).
4. Skene, J. and Emmerich, W. "Model Driven Performance Analysis of Enterprise Information Systems," Electronic Notes in Theoretical Computer Science, Vol 82, pp 147-157, 2003.
5. Menasce, D. A., Almedia V. A. F., and Dowdy. L. W., "Performance by Design: Computer Capacity Planning by Example." Prentice Hall, Upper Saddle River, NJ, 2004.
6. Menasce, D. A., Almedia V. A. F., and Dowdy. L. W., "Capacity Planning for Web Services: metrics, models, and methods." Prentice Hall, Upper Saddle River, NJ, 2001.
7. Urgaonkar, B. and Pacifici, G. and Shenoy, P. and Spreitzer, M. and Tantawi, A., "An Analytical Model for Multi-tier Internet Services and Its Applications" Proceedings of the 2005 ACM SIGMETRICS Performance Evaluation Review, pp 291-302 (2005).
8. Stewart, C. and Shen, K.," Performance Modeling and System Management for Multi-component Online Services", Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, pp 71-84 (2005)
9. Balsamo, S. and Simeoni, M., "Deriving Performance Models from Software Architecture Specifications", Proceedings of the European Simulation Multiconference, Analytical and Stochastic Modelling Techniques, pp 6-9,(2001).
10. Balsamo, S. and Simeoni, M., "On Transforming UML Models into Performance Models", Workshop on Transformations in the Unified Modeling Language,(2001)

Author Biographies

Nilabja Roy is a Ph.D. student in the Department of Electrical En*gineering and Computer Science at* Vanderbilt University. His research interests include performance evaluation of computer systems, modeling and analyzing large scale distributed enterprise and real time systems and resource allocation and application configuration using optimization techniques. He received his B.E. in Electrical Engineering from Jadavpur University, India and M.S in Computer Science from Vanderbilt University.

Dr. Douglas C. Schmidt is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University. He has published 9 books and over 400 papers that cover a range of topics, including patterns, optimization techniques, and empirical performance analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed real-time and embedded (DRE) middleware and applications. Dr. Schmidt has over fifteen years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are open-source middleware frameworks and model-driven tools that implement patterns and product-line architectures for high-performance DRE systems.