RC23451 (W0411-188) November 22, 2004
Computer Science

IBM Research Report

A Comprehensive Toolset for Workload Characterization, Performance Modeling and On-line Control

Li Zhang, Zhen Liu, Anton Riabov, Monty Schulman, Cathy Xia, Fan Zhang
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

A Comprehensive Toolset for Workload Characterization, Performance Modeling and On-line Control

Li Zhang, Zhen Liu, Anton Riabov, Monty Schulman, Cathy Xia, Fan Zhang [a]

[a] *IBM Thomas J. Watson Research CenterP.O. Box 704, Yorktown Heights, NY 10598*

Keywords: Performance analysis, performance prediction, capacity planning, Web service modeling, queueing networks, on-line control.

Abstract

With the advances of computer hardware and software technologies, electronic businesses are moving towards the on-demand era, where services and applications can be deployed or accommodated in a dynamic and autonomic fashion. This leads to a more flexible and efficient way to manage various system resources. For on-demand services and applications, performance modeling and analysis play key roles in many aspects of such an autonomic system. In this paper, we present a comprehensive toolset developed for workload characterization, performance modeling and analysis, and on-line control. The development of the toolset is based on state-of-the art techniques in statistical analysis, queueing theory, scheduling techniques, and on-line control methodologies. Built on a flexible software architecture, this toolset provides significant value for key business processes. This includes capacity planning, performance prediction, performance engineering and on-line control of system resources.

1 Introduction

As e-businesses evolve and are being adopted by more and more industries, increasing portions of the business processes are being handled by computers, through Web interfaces and Web services. Complex business logic is built into these enterprise systems with routers, Web servers, authentication servers, application servers,

Email address:
{zhangli,zhenl,riabov,schulman,cathyx,fzhang}@us.ibm.com (Li Zhang, Zhen Liu, Anton Riabov, Monty Schulman, Cathy Xia, Fan Zhang).

back-end databases, etc. These enterprise systems, controlled by sophisticated software, perform a variety of business functions including authentication/verification, ordering, approval, billing, account management, etc. In order for such complex systems to perform critical business functions reliably and efficiently, system administrators need to be able to monitor and manage the whole system effectively. There are many challenging issues in the analysis and management of these large distributed systems. Here, we present a rich set of performance modeling and analysis tools called COMPASS, to assist in dynamic capacity planning and in the efficient management of highly accessed, commercial Web systems. COMPASS stands for Control and Optimization based on Modeling, Prediction and AnalySiS. The main components of this set of tools are workload characterization, system and application modeling and analysis, and on-line optimal control. Each component can function as an independent module. More importantly, these components also work in coordination to provide better understanding and management of the underlying system.

In order to better manage such complex service systems we need to first understand what the requests to the system are, how these requests arrive, what the key characteristics are, and how the requests will change over time. Workload characterization is mainly concerned with the analysis of the request arrivals processes. It also includes on-line monitoring and prediction services for the request arrival and system usage measurements.

We next need to understand how various requests are served by the system. We need to be able to build models for the system architecture, specify the service components for each types of request, and quantify the speed and overhead for all type of requests at each service component. These details form system application modeling and analysis.

The systems, which contain many different types of resources, and may have extra capacity, typically have various resource control and scheduling mechanisms. Administrators can tune these mechanisms to achieve more efficient system usage, lower system cost and increased overall profit. The on-line optimal control component provides efficient algorithms to dynamically adjust the control policies on-line based on the recently observed arrival processes, the system and application models, and the specified control objective function.

Based on advanced statistics, stochastic processes, queueing, control and optimization theories, the use of COMPASS tools can lead to significantly improved solutions for a range of mission critical business processes including capacity planning, performance engineering, life cycle management, and business process management, etc. For example, a number of fundamental problems for capacity planning, performance prediction, and service level agreement provisioning include: What is the capacity of the current system? What is the current request traffic volume? What level of response times are users experiencing? What can be done to improve the system's performance? Where is the potential bottleneck for the system? When will the servers run out of capacity? and so on. The answers are often obtained through benchmarking, on-line monitoring, system modeling and analysis. The COMPASS tools will apply sophisticated statistics and modeling techniques for analyzing the request arrival patterns to the system, forecast how these arrivals will change over time, construct system models for the request service processes, and trigger appropriate control actions. The use of COMPASS tools will lead the current system toward a better operating state. In general, the systems will be managed more efficiently, in an autonomic, on-demand fashion.

The rest of the paper is organized as follows. In Section 2, we present the overall architecture and interfaces for the toolset. Sections 3 though 5 present the main functional modules of the toolset. We summarize in the end with discussions.

2 The Overall Architecture

In this section we first describe technologies used in COMPASS implementation. Next, we list high-level components of our implementation and explain how these components interact in different typical usage scenarios. In this section, we provide only general descriptions of algorithm families, without describing the details of methods used for modeling, analyzing and controlling target systems. We leave all relevant detailed descriptions for following sections, and focus on overall COMPASS toolkit architecture.

The implementation of COMPASS tools and algorithms is based on the Java 2 Standard Edition platform [1] [18]. Java was chosen as the language used for implementation, because it satisfied our requirements of portability, short development cycle, development of an extensive GUI, and compatibility with other performance analysis tools. We used platform version 1.4.1 in our development and testing. At least version 1.4.1, or higher, is required to run COMPASS.

In our implementation, we make use of several API sets included in the Java 2 SE platform. All user interface code is based on the portable and lightweight Swing library. JDBC is used for operations with large data arrays, which can be stored in a JDBC-compatible relational database. DOM parser provided by Java API for XML Processing (JAXP) is used to process XML files. We use Remote Method Invocation (RMI) API in some of our implementations of measurement and control components, in order to communicate to systems that are being monitored or controlled by COMPASS. Finally, Java Native Interface (JNI) is used on the target system to invoke kernel control code written in C.

[1] Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

We also use two modules that are not included in standard Java 2 Platform. We make use of advanced mathematical functions included in the freely available open source JSci library [6]. To achieve compatibility with Agent Building and Learning Environment (ABLE) we utilize open-source ABLE API [1, 2].

Figure 1 presents an overview of software components comprising of the current COMPASS toolkit. From the end-user point of view, there are three options for invoking COMPASS tools: end-system configuration mode, on-line measurement, analysis and control mode, and off-line analysis mode. Back-end algorithms and methodologies support the two analysis modes and the GUI elements. The GUI elements are responsible for the configuration of algorithm parameters. These algorithms and methodologies are often shared between on-line and off-line implementations, with minor differences in parts responsible for control flow.

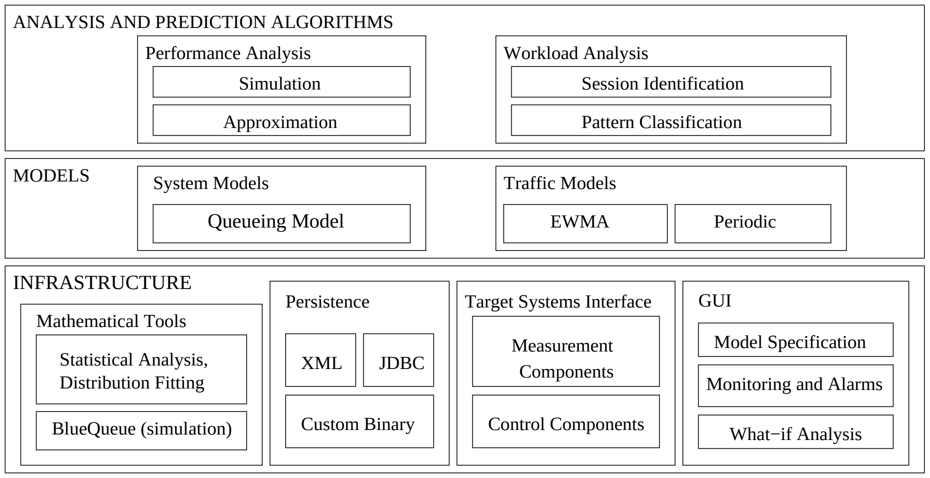


Fig. 1. COMPASS Components

The core part of COMPASS toolkit consists of several families of analysis and prediction algorithms (shown on top in figure 1). Methods used for system performance analysis include simulation and approximation. Workload (traffic) analysis is based on session identification and pattern classification algorithms. Most algorithms can be used in both on-line and off-line modes, depending on the user-specified configuration. The most trivial example of this is when the traffic analysis algorithms are used in off-line mode to infer traffic model parameters based on web server logs. In on-line mode, performance analysis algorithms can initiate control actions for improving user-perceived system response time or for maximizing profit based on a service level agreement contract. A possible action can be raising alarms to warn about potential system malfunctions, or about unusually high loads. More sophisticated control algorithms use the abstract system control interface (listed in infrastructure section on figure 1), in order to change configuration parameters of the target system in runtime. Algorithm components are designed to conform to a specified interface, and are made to be easily interchangeable and interconnectable, in order to achieve maximal code re-use in our implementation.

Our algorithms are based on a set of mathematical models, which are used to describe system and traffic behavior. Initial model parameters and target system configuration are specified with the assistance of the COMPASS GUI. Afterwards, system or traffic analysis algorithms can update model parameters based on observed system behavior.

Implementation of COMPASS analysis and optimization algorithms is supported by a set of utility classes. Mathematical utility packages include the home-grown discrete event queueing network simulation library, BlueQueue, and statistical analysis tools, which include probability distribution fitting algorithms. Algorithm parameters, time series of measurements, control actions and predictions can be persisted to a relational database via JDBC interface, or to custom binary files. Persistence utilities also provide XML serialization infrastructure for Java objects. Measurements and control actions are taken through a set of interfaces that form a target system abstraction layer. This layer, consisting of measurement and control interfaces, allows COMPASS to interact with a variety of platforms and software packages. This makes it easier to add support for new systems.

The Java Swing-based graphical user interface is essential to the COMPASS toolkit. Most of COMPASS operations can be performed or monitored via the graphical interface. These operations include system configuration setup, on-line system and workload monitoring, and off-line what-if analysis. COMPASS uses custom GUI components for displaying charts and plots, which allow to efficiently display easy to read representation of system status, forecast and other types of data.

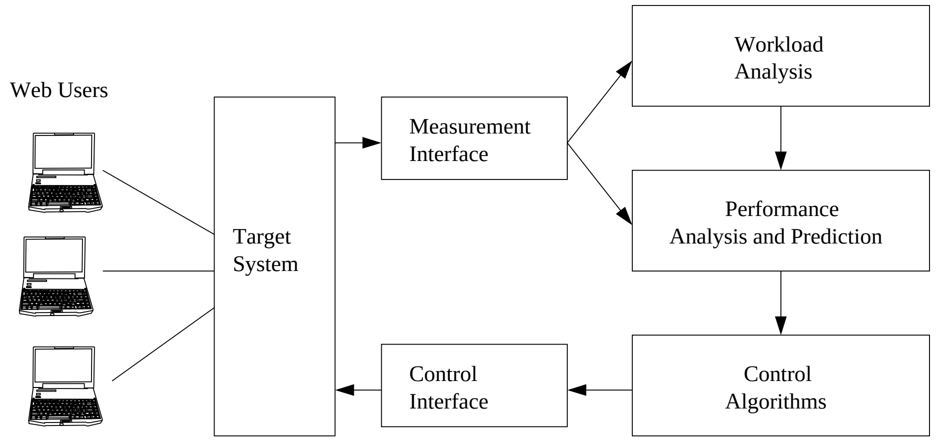


Fig. 2. Data flow in an on-line mode

In on-line analysis mode, algorithm components of COMPASS are connected through a flexible data interface. This interface supports automatic recording of time series data, raising alarms based on user specified sets of predicates, and allows GUI components to receive data for monitoring. Monitoring, saving time series data and alarms are optional. They may be enabled or disabled, according to user specifications at any particular data transfer point.

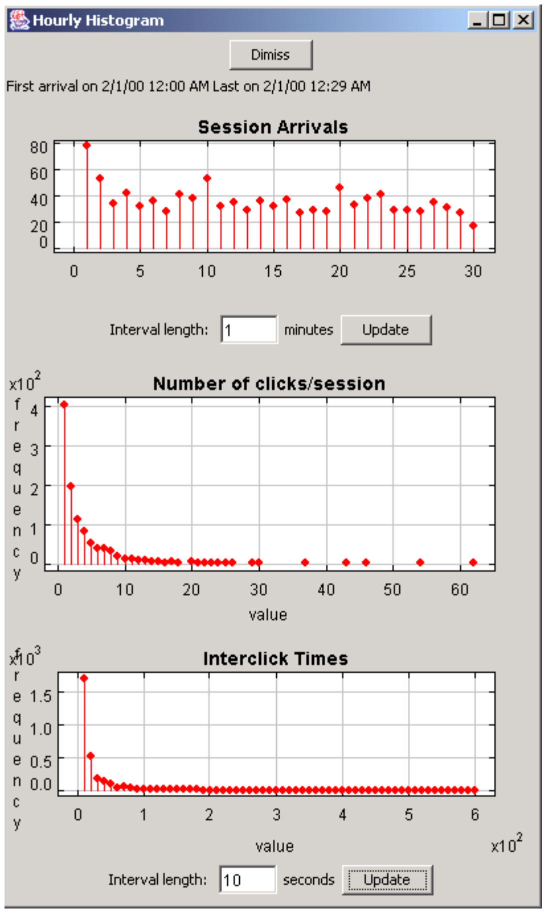
Figure 2 illustrates data flow between COMPASS components, when COMPASS is started in on-line analysis mode. Measurements are taken from the target system via system-specific components that conform to the COMPASS measurement interface. Measurements of traffic intensity are used to analyze workload and make predictions of future workload. System performance parameters, such as CPU utilization and memory, are measured and analyzed. The resulting model system performance, together with workload predictions, are used to make predictions about future system performance, make decisions about possible control actions, and raise alarms. Control actions, such as re-assigning system resources allocated to different types of tasks in the target system, are taken through the unified control interface. This interface allows easy customization for particular software configurations. The same sequence of algorithms can be used in training mode to infer system model parameters. This would be the case, if instead of connecting to a production system driven by web users, COMPASS components are connected to a test system, driven by a workload generator. In this scenario, the workload generator can also be controlled by COMPASS, and used to supply additional information to traffic modeling algorithms.

3 Workload Characterization

The workload characterization of COMPASS consists of three key components: traffic monitoring and analysis, workload profiling and classification, and traffic prediction.

Request arrival information is collected and passed to the workload characterization module through a flexible data flow interface. The analysis module analyzes arrival information and builds models for the arrival process. For example, such models for Web site usages may describe user arrivals in sessions. Within each session, users may visit multiple pages, with think times in between page views. When a page is loaded, a series of requests for embedded images is initiated. Algorithms from [10] are implemented to identify the session, and page view (or click) information. Figure 3 shows the analysis result of a customer workload. The top plot shows the number of user session arrivals every minute over the specified time window. The middle plot shows the distribution for the number of clicks in each session. The bottom plot shows the inter-click time distribution. The analysis module also provides the distribution fitting function. This function is used to calculate the best estimate of the distribution parameters for the session arrival processes, the inter-page-view, and number of page-views per session distributions. Users can select the format and type of mixture of distributions to use for the fitting of the distributions. The output result can be stored in a given file. This file also includes the goodness of fit measures, which is used for the fitting.

Key characteristics of the arrival process that have strong impacts on the server's



First arrival on 2/1/00 12:00 AM Last on 2/1/00 12:29 AM

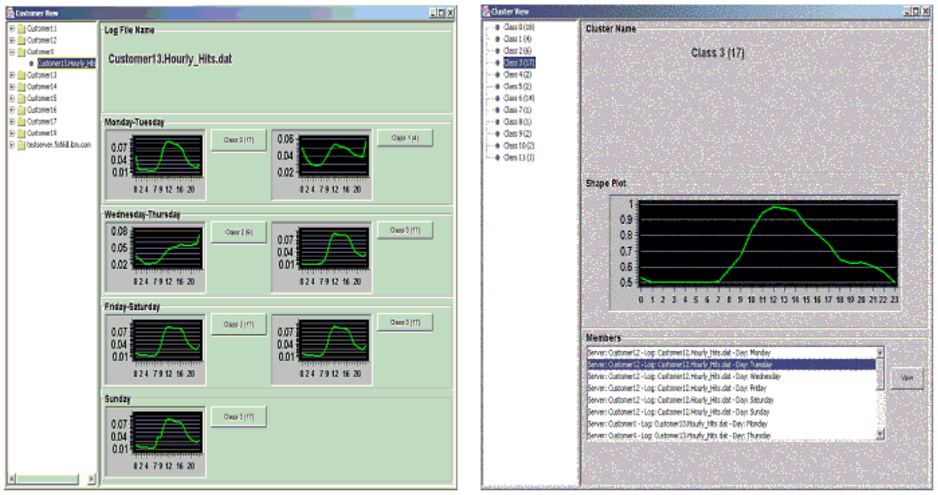
Interval length: 1 minutes

Interval length: 10 seconds

Fig. 3. Example of A Customer Workload

performance are also extracted by the workload characterization module. Studies [15, 16, 17] have shown that the correlation characteristics, such as short-range and long-range dependencies, have significant impact on user response times. The burstiness characteristics, such as the variability and heavy-tailness of the request distributions, also have significant impact. The user response times, under long-range dependent and heavy-tailed request processes, can degrade by orders of magnitude and have fundamentally different decay rates, when compared with the traditional Poisson models. These key parameters include correlation factors, marginal distributions, identified user access patterns, page visit sequences, think times, and various matching distribution parameters. These parameters are calculated by the workload characterization module to establish a complete workload profile, which is used as input to the other modeling modules.

The profiling and classification module further builds a profile for the arrival process for each customer system. It represents arrival patterns over time for each customer system. The clustering engine construct groups for the customer systems that have similar arrival patterns. For a new customer system, its pattern can be mapped into the most similar group by the classification engine. Clustering and classification algorithms are explained in detail in [12]. The left figure in Figure 4 is the customer view panel. It shows for a given customer, the normalized request patterns for different days of the week and their corresponding classes. The right figure in Figure 4 is the cluster view. It shows for a given class, the common request pattern and the list of class members. Each member of the class corresponds to a one-day pattern for a customer site.



Server: Customer12 - Log: Customer12.Hourly_Hits.dat - Day: Monday
Server: Customer12 - Log: Customer12.Hourly_Hits.dat - Day: Tuesday
Server: Customer12 - Log: Customer12.Hourly_Hits.dat - Day: Wednesday
Server: Customer12 - Log: Customer12.Hourly_Hits.dat - Day: Friday
Server: Customer12 - Log: Customer12.Hourly_Hits.dat - Day: Saturday
Server: Customer12 - Log: Customer12.Hourly_Hits.dat - Day: Sunday
Server: Customer8 - Log: Customer13.Hourly_Hits.dat - Day: Monday
Server: Customer8 - Log: Customer13.Hourly_Hits.dat - Day: Thursday

Fig. 4. Example of Customer View and Cluster View in Profiling

The prediction module makes predictions of the request arrival process, using the time series models. It's predictions are based on the past access patterns, and the constantly changing volume. Figure 5 provides the prediction results for two anonymous customer Web systems, based on one of our adaptive, template-based algorithms. The top plot in Figure 5 shows the measurements and the predictions. The closer the measurement and prediction lines appear, the better the prediction will be. The prediction errors are also analyzed and plotted in the middle plot of Figure 5, to demonstrate the effectiveness of the prediction algorithm. This plot shows the percentage of relative error against the fraction of time, where the relative error is below a given value. The lower the curve appears, the better the prediction will be. One can also implement other prediction algorithms under our flexible framework, and compare with the collection of prediction algorithms in the repository. Figure 6 shows the prediction accuracy for a subset of available customer Web sites. The algorithm predicts to above eighty percent accuracy over eighty percent of time for most of the customer Web sites.

The request volume usually changes over time for a live running system. Given the changing system load as the result of the changing arrival volume, the system may need to take certain actions to better accommodate for the changing situation. A change point detection algorithm is provided to detect on-line the changing state of the arrival process. Within each state, the arrival process remains relatively unchanged. The vertical lines in the bottom plots of Figure 5 illustrate the changing points as detected by our algorithm.

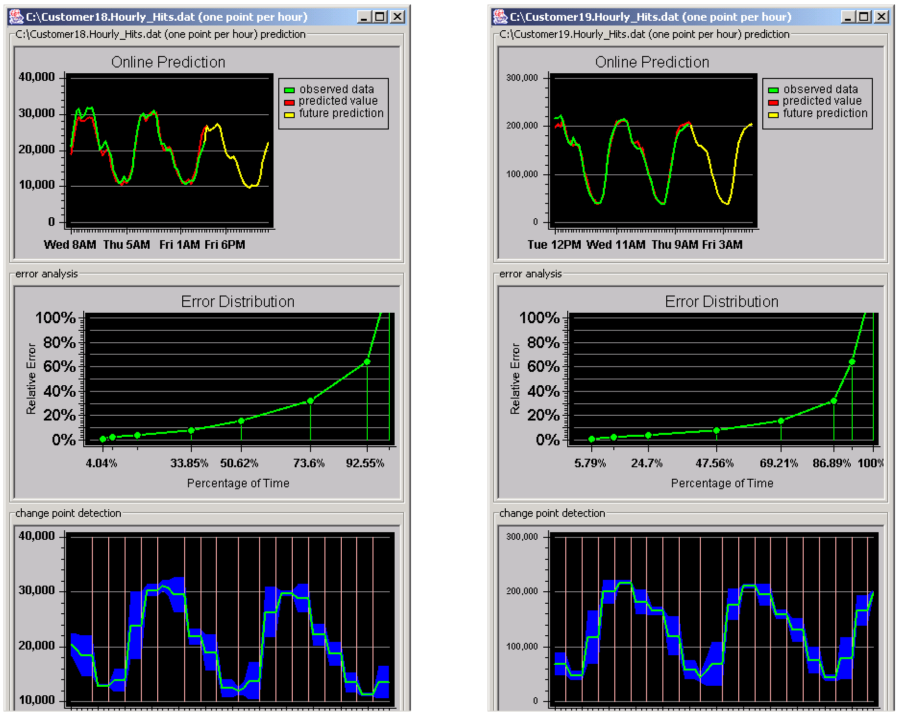

C:\Customer18.Hourly_Hits.dat (one point per hour)
C:\Customer18.Hourly_Hits.dat (one point per hour) prediction
Online Prediction
observed data
predicted value
future prediction
Wed 8AM Thu 5AM Fri 1AM Fri 6PM
error analysis
Error Distribution
Relative Error
Percentage of Time
4.04% 33.85% 50.62% 73.6% 92.55%
change point detection
C:\Customer19.Hourly_Hits.dat (one point per hour)
C:\Customer19.Hourly_Hits.dat (one point per hour) prediction
Tue 12PM Wed 11AM Thu 9AM Fri 3AM
5.79% 24.7% 47.56% 69.21% 86.89% 100%

Fig. 5. Predictions for Two Customer Web Site

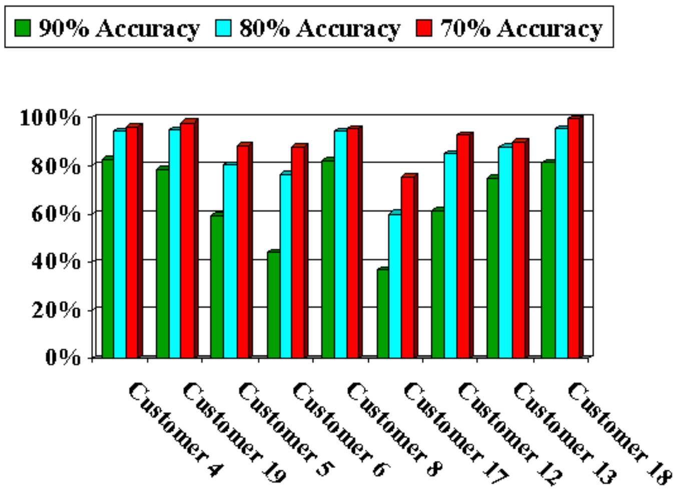


Fig. 6. Prediction Accuracy

A centralized repository is set up so that all of the measurement data, analysis results, and prediction algorithms can be stored for later re-use and comparisons. One can also re-play the historical data from the repository in order to drive another system for benchmarking, and evaluate the prediction/change-point detection algorithms.

The workload models in the repository can be used to scale up the traffic volume, and generate synthetic, realistic workloads. A set of clients can then generate web requests according to these synthetic workloads. An example that uses this scenario is a realistic benchmarking tool named Bluestone, which is currently being developed.

4 System and Application Modeling and Analysis

Queueing network models are commonly used to model the request serving process in many service systems, including manufacturing and Web server systems [9, 13]. More functions provided by the Web service infrastructures have resulted in more complex systems, as well as more complicated user access patterns. To model how the user requests, or *transactions*, are served by such complex systems, a single type of request stream feeding into a single black box is far from adequate.

The system and application module constructs flexible queueing network models to capture the Web serving process. Each of the multiple components within the server system can be represented as a queue, or as a more complex sub-queueing network. For example, one can use a queue to model the network component within a system, and use a single server queue to model a database, etc. Different routing mechanisms, such as round robin and probabilistic routing, can be used to approximate the load balancing schemes in the system [7]. Common service policies, such as processor sharing or priority policies, can be used for each queueing or server component within the model to mimic the components service behavior. Users can be categorized into multiple classes based on their access behaviors. For given routing and service parameters of such a queueing system, the system and application modeling module readily obtains the performance related measures such as throughput, utilization and response times, via simulations and queueing network theories. The workload profile feeding into the system can be the original as well as the forecasted profile from the workload models.

Figure 7 shows a sample system architecture. The two types of sources on the left represent two classes of arrival streams to a queue. The splitter represents a load balancing router, which spreads these jobs to a single server and a complex server station. The server station again has two servers, with a splitter to balance the load. After finishing service from either the top-level server or the server station, the jobs complete their service at the system and exit. The system editor supports many different kinds of queueing disciplines, load-balancing algorithms, and server processing policies. The supported queueing disciplines include first-come-first-serve (or FIFO), priority (or HOL) and time-dependent-priority (or TD) [9]. A range of supported load balancing algorithms include round robin (RR), class-dependent probabilistic routings (PRV, DET, and PRM in Figure 7), and send-to-first-available (STF). Each server can serve one job at a time (Single Job), or can serve multiple jobs concurrently according to the processor sharing, discriminatory processor sharing [4], or weighted fair queueing policies [3]. Figure 8 shows an example of

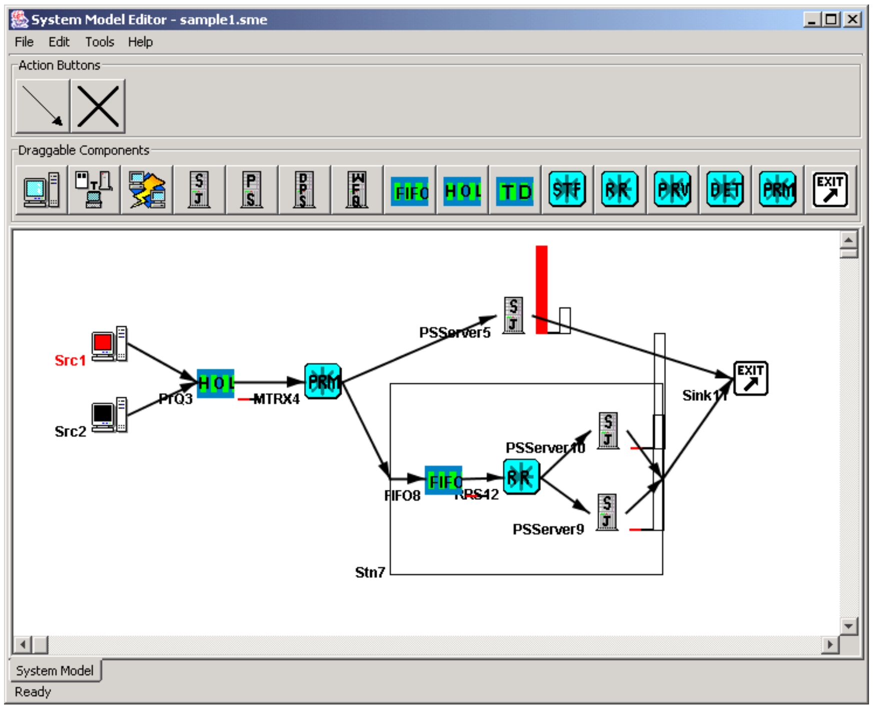
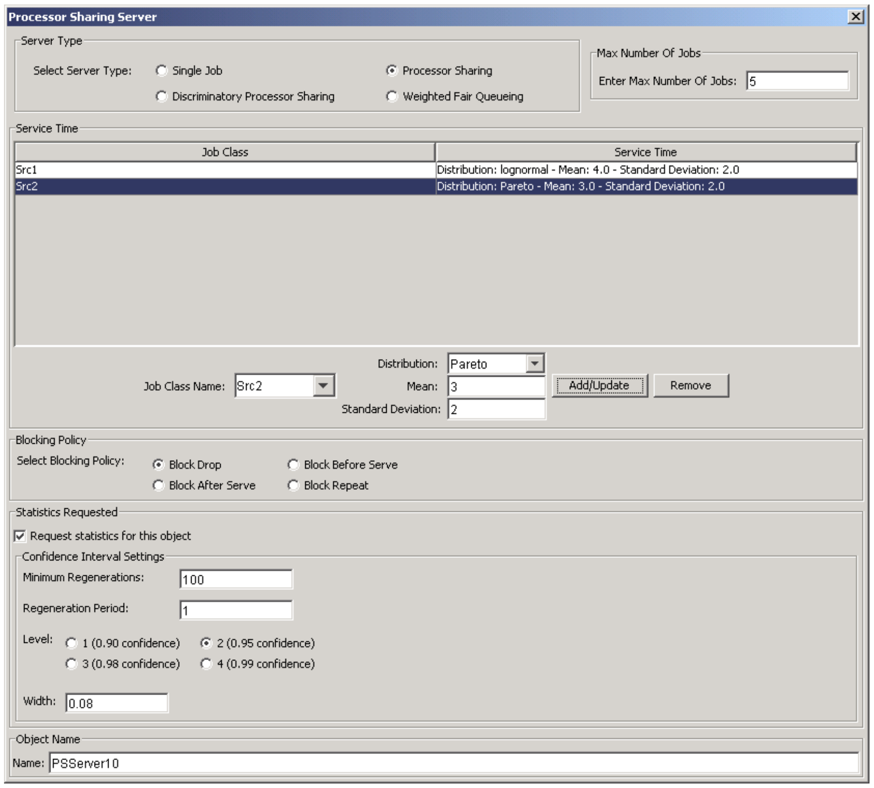


Fig. 7. The System Model Editor

the screen for server configuration. This panel allows users to specify the class-dependent service time distributions at the server, the maximum number of jobs the server can serve at any given time, and the blocking policies for new jobs arriving to a busy server.

The specified system is analyzed by running regenerative discrete event simulations, using our BlueQueue queueing network simulation class library. BlueQueue provides a set of components that facilitates the modeling of a queueing network, using a set of queueing, service and load balancing policies. Service and inter-arrival times can be modeled with random variables, following one of several commonly used families of probability distributions. BlueQueue models allow to specify pre-determined or random routings for multiple classes of jobs. Simulated parameters, such as response time, can be analyzed for individual components or arbitrary parts of the queueing network, using statistics collection objects.

The statistics objects collect per-class response time mean, standard deviation, and distribution information for each checked component (including queues and servers) in the system. The system begins a regeneration cycle when a job arrives to an empty system. The confidence intervals are calculated at the end of each regeneration cycle. The simulation stops when the specified confidence levels are reached for all of the checked components. The total running time of the simulation depends on the load of the simulated system. If the simulated system is heavily utilized, then it does not become empty very often. It takes a longer time for a heavily utilized system to observe the same number of regeneration cycles, compared with



Processor Sharing Server

Server Type

Select Server Type: Single Job [✓ Processor Sharing] Discriminatory Processor Sharing Weighted Fair Queueing

Max Number Of Jobs

Enter Max Number Of Jobs: 5

Service Time

Job Class	Service Time
Src1	Distribution: lognormal - Mean: 4.0 - Standard Deviation: 2.0
Src2	Distribution: Pareto - Mean: 3.0 - Standard Deviation: 2.0

Job Class Name: Src2

Distribution: Pareto

Mean: 3

Standard Deviation: 2

Add/Update Remove

Blocking Policy

Select Blocking Policy: [✓ Block Drop] Block Before Serve Block After Serve Block Repeat

Statistics Requested

[✓] Request statistics for this object

Confidence Interval Settings

Minimum Regenerations: 100

Regeneration Period: 1

Level: 1 (0.90 confidence) [✓ 2 (0.95 confidence)] 3 (0.98 confidence) 4 (0.99 confidence)

Width: 0.08

Object Name

Name: PSServer10

Fig. 8. The Server Specification Screen

an under-utilized system. Hence, simulating a heavily utilized system takes a longer time. During the simulation run, the per-class queue lengths and server utilizations are displayed and periodically refreshed as shown in Figure 7. The simulation can be interrupted in the middle of a run or continue until confidence is reached. The detailed simulation results are shown in a display window and can be saved in text format.

Analytical analysis can be conducted for the system throughput and utilization analysis. Given the type of growth rate for each type of request, the analytical analysis module identifies the potential bottleneck components and the time horizon for these bottlenecks to reach their limits.

While drag and drop editing makes COMPASS tools user friendly, the XML import and export functions for model specification make the tools open and extensible. Using the extensible markup language (XML), the constructed models are easily described and customized for many different platforms in heterogeneous environments. Models can be built and saved in native binary format, or in XML format. We can load stored models, or import models from their XML descriptions. This makes it easier for users to communicate and share their work.

After the system model has been built, we can further feed the workload predictions from the workload characterization component in Section 3, and obtain the predicted performance measures. Combining the predicted performance measures with the bottleneck analysis results, we can then obtain a deeper understanding of how the system functions. In particular, many capacity planning related issues can be solved. For example, the report includes the projected performance measures, for a projected load or a projected time horizon. These performance measures include the user response times, and system utilizations. The report also includes the projected time for the system to experience performance problems, identification of the overload component, and recommendations for the best actions. Examples of such recommendations are adding front-end or back-end servers, so that the system can deliver the expected performance.

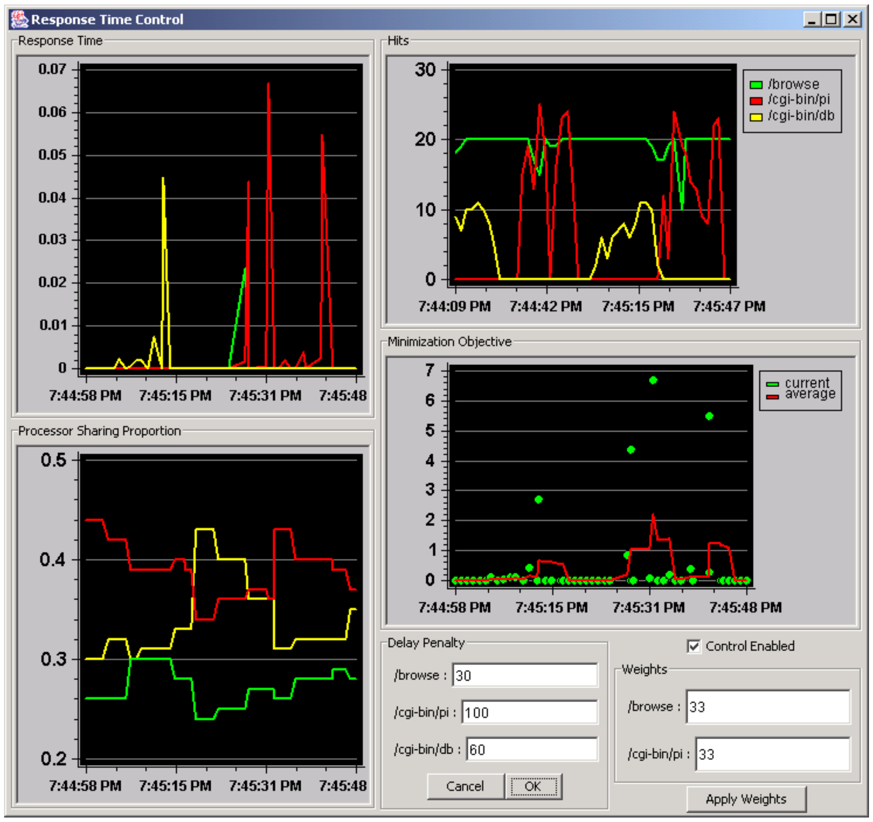
5 On-line Optimal Control

Based on the appropriate data of sufficient detail and accuracy, we can construct workload, system and performance models automatically. Based on these models and the measurements from the live system, various control actions can be taken.

Many systems are capable of performing real-time resource manangement and scheduling functions [5]. These systems can be configured to adjust system resources, such as network bandwidth and CPU, allocated to different types of jobs based on the user's class, or the type of requested service. These control mechanisms are used to achieve a certain QoS objective, as well as minimize overall system operating cost. The optimization functions in the modeling modules map the given QoS requirements into the most cost and operation efficient hardware and software configurations. The on-line optimal control module activates a controller using RMI to dynamically change the scheduling and resource allocation policies within the servers, based on the system and performance prediction models. Results of these actions are reflected in changes in the monitored performance measures. These performance measures, together with the changing workload, will again influence the control decisions. With all of these functions in place, the system will then be empowered with self-managing capabilities.

A sample test system has been set up for this on-line optimal control module. The system consists of a front end web server and a back-end database. A set of client machines are used to generate three classes of user requests. These three classes of requests are static images, CPU intensive CGI (Common Gateway Interface) scripts, and database queries. A proportional share scheduler is activited on the web server to provide proportional CPU allocation to the three classes of requests, based on the weight parameters assigned to them. These weights can be adjusted in real-time by the control module. Measurement agents collect throughput and response information from the web server, and then sends the collected information to the control module. The control module calculates the best parameter setting for the proportional share scheduler, in order to minimize a weighted sum of the response times. These control parameters are then passed on to the scheduler, and become active immediately.

Figure 9 shows a sample control panel for the live system. The top two plots in Figure 9 provide the response time and request volume information, for each class of request. The lower left plot shows sharing proportions among the request classes for the system resources. The goal is for the system to minimize a weighted sum of the response times. This objective function over time is plotted on the lower right plot. These delay penalty weights can be specified, and changed on screen. They are used to distinguish the importance of different requests. The automatic control actions can be activated or deactivated on the screen. An improvement in the overall system performance has been observed in our system testbed, as a result of the control actions.



Delay Penalty

/browse : 30

/cgi-bin/pi : 100

/cgi-bin/db : 60

[✓] Control Enabled

Weights

/browse : 33

/cgi-bin/pi : 33

Fig. 9. The Control Panel

6 Summary

We have presented a comprehensive toolset for workload characterization, performance modeling and analysis, and on-line control, based on advanced techniques in statistical analysis, queueing theory and dynamic scheduling and control methodologies.

Workload characterization provides on-line monitoring and analysis of requests to the system. It also provides agents to collect and display system usage measurements. This module conducts profiling, traffic analysis and predictions for the incoming workload. The workload models can be the input to the system and application modeling module. They are also used to generate realistic benchmarks for testing purposes.

The use of system application modeling and analysis builds multi-class queueing network models for the request serving process. It provides a flexible way to build models for the general system architecture, specifies the service components, and quantifies the speed and overhead for each type of request at each service component. Simulation and analytical solutions are used to analyze the given system, and provide valuable throughput, response time, and bottleneck analysis.

The on-line optimal control component provides efficient algorithms to dynamically adjust resource control policies on-line, based on recently observed arrival processes, system and application models, and the control objective function.

Built on top of a flexible architecture, with a rich set of functional components working in coordination, this toolset provides significant value for key business processes. The significant value includes capacity planning, performance prediction, performance engineering and on-line control of system resources.

References

[1] J. P. Bigus, J. Bigus. Constructing Intelligent Agents with Java™: A Programmer's Guide to Smarter Applications. John Wiley & Sons; Book and CD-ROM edition (December 1997). ISBN 0471191353.

[2] J. P. Bigus, D. Schlosnagle, *et al*. Agent Building and Learning Environment. `http://www.alphaworks.ibm.com/tech/able/`

[3] A. Dmers, S. Keshav and S. Shenker. Analysis and Simulation of Fair Queueing System. Internetworking Research and Experience, Vol. 1, 1990.

[4] G. Fayolle, I. Mitrani, R. Iasnogorodski. Sharing a Processor Among Many Job Classes. J. ACM, Vol. 14, No. 2, 1967.

[5] L. L. Fong, M. H. Kalantar, D. P. Pazel, G. Goldszmidt, K. Appleby, T. Eilam, S. A. Fakhouri, S. M. Krishnakumar, S. Miller and J. A. Pershing. Dynamic Resource Management in an eUtility. In *Proceedings of NOMS 2002 IEEE/IFIP Network Operations and Management Symposium*, Piscataway, NJ, IEEE. 2002, p. 727-40, April 2002.

[6] M. Hale, *et al*. JSci - A science API for Java™. `http://jsci.sourceforge.net/`

[7] G. Hunt, G. Goldszmidt, R. King, and R. Mukherjee. Network dispatcher: A connection router for scalable internet services. In *Proceedings of the 7th International World Wide Web Conference*, April, 1998.

[8] A. K. Iyengar, M. S. Squillante, and L. Zhang. Analysis and characterization of large-scale web server access patterns and performance. *World Wide Web*, 2, June 1999.

[9] L. Kleinrock. *Queueing Systems Volume II: Computer Applications*. John Wiley and Sons, 1976.

[10] Z. Liu, N. Niclausse, and C. Jalpa-Villanueva. Web traffic modeling and performance comparison between HTTP 1.0 and HTTP 1.1. In E. Gelenbe, editor, *Systems Performance Evaluation: Methodologies and Applications*, pages 177–189. CRC Press, 2000.

[11] Z. Liu, M. S. Squillante, C. H. Xia, and L. Zhang. Preliminary analysis of various SurfAid customers. Technical report, IBM Research Division, July 2000. Revised, December 2000.

[12] Z. Liu, M. S. Squillante, C. H. Xia, S. Yu, and L. Zhang Web Traffic Profiling, Clustering and Classification for Commercial Web Sites. The 10th International Conference on Telecommunication Systems, Modeling and Analysis (ICTSM10), 2002.

[13] D. A. Menasce, and V. A. F. Almeida. *Capacity Planning for Web Performance: metrics, models, and methods.* Prentice Hall, 1998.

[14] A. K. Parekh, and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *IEEE Transactions on Networking*, Vol 1, No. 3, 1993.

[15] M. S. Squillante, B. Woo, and L. Zhang. Analysis of queues with dependent arrival processes and general service processes. Technical report, IBM Research Division, 2000.

[16] M. S. Squillante, D. D. Yao, and L. Zhang. Web traffic modeling and web server performance analysis. In *Proceedings of the IEEE Conference on Decision and Control*, December 1999.

[17] M. S. Squillante, D. D. Yao, and L. Zhang. Internet traffic: Periodicity, tail behavior and performance implications. In E. Gelenbe, editor, *Systems Performance Evaluation: Methodologies and Applications*. CRC Press, 2000.

[18] Sun Microsystems, Inc. Java™ 2 Platform, Standard Edition (J2SE™). `http://java.sun.com/j2se/`

[19] R. W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[20] L. Zhang, C. H. Xia, M. S. Squillante, and W. N. Mills III. Workload service sequirements analysis: A queueing network optimization approach. In *Tenth IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2002.