Quantitative System Evaluation with Java Modeling Tools
(Tutorial Paper)

Giuliano Casale∗
Imperial College London
Dept. of Computing
London, SW7 2AZ, U.K.
g.casale@imperial.ac.uk

Giuseppe Serazzi
Politecnico di Milano
Dip. Elettronica e Informazione
I-20133 Milan, Italy
giuseppe.serazzi@polimi.it

ABSTRACT
Java Modelling Tools (JMT) is a suite of open source applications for performance evaluation and workload characterization of computer and communication systems based on queueing networks. JMT includes tools for workload characterization (JWAT), solution of queueing networks with analytical algorithms (JMVA), simulation of general-purpose queueing models (JSIM), bottleneck identification (JABA), and teaching support for Markov chain models underlying queueing systems (JMCH). This tutorial summarizes the main features of the tools that compose the suite. Furthermore, using a composite case study, we provide intuition on the versatility of JMT in dealing with the different aspects of quality-of-service (QoS) evaluation, what-if analysis, and software performance tuning.

Categories and Subject Descriptors
C.4 [Performance of Systems]: Modeling Techniques;
D.2.8 [Software Engineering]: Metrics—complexity measures, performance measures

General Terms
Performance

Keywords
Tools, Performance Evaluation, Modeling, Simulation, Load Balancing, Bottlenecks detection, Queueing Networks

1. INTRODUCTION
Ongoing work in the software performance modeling community has significantly stressed the importance of developing automated or semi-automated frameworks for performance optimization and management of complex applications [8,9]. This is especially important in the design phases of a large-scale software in order to avoid taking choices that result in poor quality of service (QoS).

The Java Modelling Tools suite (JMT) is here proposed as tool to visually help software and system performance engineers to predict the performance of a system and quickly answer what-if questions. JMT is released as an open source tool suite that can be downloaded free from http://jmt.sourceforge.net. Thanks to the availability of Java sources, JMT functionalities may be freely integrated or interfaced via XML with other tools, as done for example in Opedo [7].

JMT consists of six applications that communicate using XML with a core algorithmic module composed by the simulation engine (JSIMengine) and by a library of analytical functions for performance model evaluation.

• JSIMengine is a graphical design environment for queueing network models which is tightly coupled to the JSIMengine for running discrete-event simulation. JSIMwiz replaces the graphical framework of JSIMgraph with a set of wizards that guide the user through the definition of a queueing model. The tools generate XML specifications of simulation models, pretty-print visualization of complex networks, automatic model debugging, support for what-if analyses, and dynamical presentation of simulation state, performance metrics estimates and related confidence intervals. JSIMengine supports the evaluation of the most popular types of queueing models and several constructs that cannot be solved with exact analytical techniques like multiclass queueing networks with blocking, priorities, fork-and-join elements, burstiness, and state-dependent routing schemes.

• JMVA is a graphical user interfaces for the analytical evaluation of queueing network models. The tool relies on an implementation of the Mean value Analysis (MVA) algorithm for closed networks, together with its extensions for open and mixed networks.

• JABA is an analytical tool for automatic identification of the bottlenecks in multiclass closed queueing networks. The tool receives in input a set of service demands specifying the speed of each server in processing requests of the different classes. JABA identifies the mixes of requests of the different classes that saturate concurrently more than one resource. It uses efficient convex-hull algorithms. This saves the computational costs of a long simulation analysis over different mixes of requests.

• JWAT tool supports the workload characterization process. Algorithms for data scaling, sample extraction, outlier filtering, k-means and fuzzy k-means clustering for
identifying similarities in the input data are provided. These techniques allow the identification of cluster of customers having similar characteristics. The clusters centroids represent the mean values of the parameters of the classes (e.g., CPU time, number of I/Os, number of web pages accessed) that can be used for the workload parameterization. The characterization of time-varying workloads (e.g., burstiness analysis) and the fitting of input data with exponential and Pareto distributions are also supported.

- JMCH application is a graphical simulator of M/M/c and M/M/c/K queues. The simulation state is visualized both on the queue buffer and on a Markov model representing the state system.

2. LEARNING JMT BY EXAMPLE

In this section, we provide a case study to illustrate two applications of the JMT suite, namely JAVA and JMVA. The case study considers the performance analysis and optimization of the 3-tier enterprise system illustrated in Figure 1. This is composed by a Web Server (presentation tier), 2 Application Servers (business logic tier), and 3 Storage Servers (data tier) for DBs. The evaluation of performance metrics such as throughputs and response times is here obtained using the JMVA tool, which considers product-form queuing networks hence it implicitly takes assumptions regarding exponentiality of service times for first-come first-server queues and other properties of the queuing network model.

For systems where such assumptions are not satisfied, JMVA results can be equivalently replaced by the simulation-based estimated provided by JSIMgraph or JSIMwiz. The workload of the system consists of two classes of applications which have different requirements in terms of the amount of resources requested. Furthermore, each application may be subject to a different QoS constraints. The two application classes are data intensive: data processing for the first class and data updating for the second class. We assume that, for performance reasons, the maximum number of requests simultaneously in execution is limited to \[ N \]. Therefore we consider a closed queuing model. This is a convenient assumption to model load balancing based on admission control or to describe the performance of systems which use a finite number of software threads to serve requests. For the sake of illustration, let us set \[ N = N_1 + N_2 = 100 \] as the constant number of requests in execution, where \( N_1 \geq 0 \) and \( N_2 \geq 0 \) are the number of requests in execution for the two application classes, respectively. The objectives of the case study are twofold:

- to study the performance behavior of the system as a function of the different mix of requests in concurrent execution;
- to determine the optimal load balancing of the system that maximizes the global application throughput while satisfying the QoS constraints.

In order to achieve these goals, we can follow a methodology that combines the features of JMVA and JABA. JMVA is used to estimate the performance of the system for a given parameter range. Conversely, JABA is used to drive in a computationally efficient manner the guessing of the optimal load balancing for the system. The main steps of the case study are:

1. Identify with JABA the bottlenecks of the system with respect to all the possible mixes of requests in execution. This allows to quickly identify what resources may limit the performance of the system under all possible workload mixes \( (N_1, N_2) \) under the assumption that \( N = N_1 + N_2 \) is asymptotically large.

2. Evaluate with JMVA the most important performance measures, e.g., throughput, response time, resource utilization, system power, per-class and at the system level with respect to all the possible mixes of requests in execution. This refines the information obtained in the previous step by considering the actual value \( N = 100 \) in place of the asymptotic one. For large models, it may be useful to restrict the range of analysis to a subset of workload mixes following the insights obtained from JABA.

3. Compute with JABA the optimal load balancing that maximizes the system throughput. JABA supports dynamic re-evaluation of a system’s asymptotic performance without the need of solving the models with a computationally intensive procedure as in JMVA. Hence, one could visually re-estimate the system load balancing in order to obtain a more performing configuration.

4. Evaluate with JMVA the new performance metric values with respect to all the possible mixes of requests in execution. This validates for \( N = 100 \) the load balancing reconfigurations suggested by JABA.

To characterize the requests of the applications in terms of processing requirements a set of service demands, one for each resource and for each class, is used. The service demand of a request of class \( r \) at resource \( i \), \( D_{i,r} \), is the total amount of time the request requires at that resource in order to be completely executed. The service demand value is computed ignoring contention by other requests and may be estimated.
directly from measured utilizations according to the relation $U_{i,r} = X_i D_{i,r}$, where $X_i$ is the system throughput for workload class $r$ and $U_{i,r}$ is its utilization at resource $i$. The parameters of our system, in ms, are shown in Fig. 2. The amount of work requested from the Web Server is much less demanding than the one requested from the Application and Storage Servers. The computations required by the business logic place a medium load on the Application Servers while the high number of data manipulated, uploaded and downloaded, generate a high load on the Storage Servers.

In the asymptotic analysis phase [2], JABA derives the set of bottlenecks as a function of all the possible mix of requests (see Fig. 3). Indeed, keeping the total population of the system constant, and large, and varying the population mix, we may observe a bottleneck migration phenomenon. When the fraction of class 1 requests is between 18.2% and 72.7% of the total population, two resources, namely StorageServer1 and StorageServer2, saturate concurrently. This is in contrast with the other segments of Fig. 3 where it is shown that some mixes of requests result in only StorageServer1 or StorageServer2 being saturated.

The identification of the interval of joint saturation for StorageServer1 and StorageServer2, referred to as common saturation sector, is important in order to find the load of the system that satisfy the performance criteria related to the QoS. Indeed, it can be shown that the equiutilization point, i.e., the mix that causes the two bottlenecks to be equally utilized, lies into this interval and provides the maximum system throughput [10]. Furthermore, the existence of such saturation sectors, despite being derived under the assumptions of product-form theory, has been independently observed to exist in real-world multi-tier applications [9].

The graph of Fig. 4 generated by JABA, provides a visual representation of the resources that may become saturated. The utilization of the three storage servers are shown in Fig. 7. As predicted, the StorageServer1 and StorageServer2 saturate together for all the mixes of the common saturation sector while the utilization of StorageServer3 is definitively lower. $U_{\text{Storage}3} = 0.58$, since its service demands are smaller with respect to the ones of the two bottlenecks. In Fig. 8 the Power measures, at the system level and per-class, are shown. The Power measure, first introduced in [6], is an interesting metric that combines the throughput $X$ and the response time $R$. This metric is the ratio $\Phi = X/R$ of throughput and response time and captures the level of efficiency in executing a workload. The maximum Power corresponds to the optimal operating point for the system, i.e., the point in which the throughput is maximized with the minimum response time. This concept is directly related with the one of QoS. JMVA plots the Power for the requests of each class and at the aggregate level. As shown in Fig. 8, the population mixes close to the extremes of the common saturation sector provides a better QoS to the requests of one class or of the other. When the fraction of requests of class 1 is about 0.21 the QoS of class 1 is maximized, while with a fraction of about 0.69 the QoS of class 2 is maximized.
In this paper, we have illustrated the application of JMT to a basic QoS optimization problem. The study aimed at the bottleneck identification, performance evaluation and optimization of an enterprise system and proves the simplicity of studying a system’s performance in a graphical way by means of JMT. Further examples and case studies are provided in the bibliography section of the JMT websites at http://jmt.sourceforge.net.

4. REFERENCES


Figure 7: Utilization of the three storage servers (red curve for Storage 1, blue curve for Storage 2, and light blue curve for Storage 3) as a function of the mix of requests in execution.


Figure 8: Behavior of the Power (the ratio of $X$ to $R$) at the system level (red curve) and per-class (blue curve for class 1 and light blue for class 2).

Figure 9: Service demands of the optimized system.
Figure 10: Identification of the bottlenecks in the optimized system. The load of the three Storage Servers is balanced.

Figure 11: Utilization of the storage servers (red curve for Storage 1, blue curve for Storage 2, and light blue curve for Storage 3) with the balanced load.

Figure 12: Throughput behavior of the optimized system (red curve for system throughput, blue curve for class 1 and light blue for class 2).

Figure 13: Response time of the system (red curve) and residence times of the three storage servers (light blue curve for Storage 1, blue curve for Storage 2, and black curve for Storage 3).