Performance optimization with JMT: Java Modelling Tools

D. Cerotti, M. Gribaudo, P. Piazzolla and G. Serazzi
Dip. di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Via Ponzio 34/5, 20133 Milano, Italy
{cerotti, gribaudo, piazzolla, serazzi}@elet.polimi.it

ABSTRACT

In this work, the main features of the Java Modelling Tool (JMT) are introduced. The standard algorithms for the solution of queueing networks with analytical, simulative and asymptotic techniques are reviewed. A new approach, based on command line interface and a scripting language, that increases the batch-analysis features of the tools is introduced. Two applications, for the optimization of a multi-tier system and a virtual environment are described. This proves how the JMT tool suite can be used to solve capacity planning problems.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Tools

Keywords

Performance optimization, Queuing model tool

1. INTRODUCTION

The Java Modeling Tools (JMT[1]) [2] is a set of instruments to support performance optimization studies. In this work we use these tools to solve two case studies related to performance optimization. The models are described by queuing networks, and are solved with analytical and asymptotic techniques. The first case study concerns the identification of the optimal operational point of a closed system with a two-class workload and two resources. The second case study investigates the best configuration of resources for a complex multiclass system by interfacing the tool with an external application that comprises several modules implemented in `python`. The problem approached is the evaluation of the efficacy of an allocation/deallocation policy for the Virtual Machines (VM) in a cloud environment subject to different types of Service Level Agreements (SLAs).

[1]JMT is available at `http://jmt.sourceforge.net`

ValueTools'13, December 10 – 12 2013, Turin, Italy

2. OVERVIEW OF THE JMT

JMT is a free open source suite consisting of seven tools for performance evaluation, capacity planning, workload characterization, and modelling of computer and communication systems. The suite implements several algorithms for the exact, approximate, asymptotic and simulative analysis of queueing network, with or without product-form solution. An XML data layer enables full interoperability of the computational engines. The components are the following:

- JMVA: for the exact and approximate analysis of single-class or multiclass product-form queueing networks, processing open, closed or mixed workloads. Several exact and approximate algorithms are available, such as MVA, RECAL (Recursion by Chain Algorithm), MoM (Method of Moments), Chow, Bard-Schweitzer, AQL, Linearizer, De Souza-Muntz Linearizer.
- JSIM: a discrete-event simulator for the analysis of queueing networks. It supports non-exponential, load-dependent, service and inter-arrival times. Finite capacity regions (blocking), fork-join (parallelism), class switch, priority, automatic transient detection and `What-if` analyses are also provided.
- JSIMwiz and JSIMgraph: two user interfaces for the JSIM engine. The first is wizard-based interface while the second is a graphical. Both interfaces support state-dependent routing strategies, such as routing to the server with minimum utilization, or with the shortest response time or with minimum queue length, and load dependent routing. JSIMgraph allows to export network topologies in vectorial or raster image formats.
- JMCH: it applies both analytical and simulation techniques to solve M/M/c or M/M/c/k queues. It allows to dynamically change the arrival rate, the service time and the queue size of the system, while showing an animation of the underlying Markov Chain.
- JABA: for the identification of bottlenecks in multiclass closed product-form networks using convex hull algorithms. Possible applications are: identification of

VALUETOOLS 2013, December 10-12, Torino, Italy

DOI 10.4108/icst.valuetools.2013.254399

potential bottlenecks and of the optimal load, throughput maximization and response time minimization.

- JWAT: supports the workload characterization. It can import data files and perform analysis using statistical techniques for multivariate data: means, correlations, histograms, boxplots, scatterplots. Algorithms for data scaling, sample extraction, outlier filtering, k-means and fuzzy k-means clustering are provided.

3. OPTIMAL LOAD OF A MULTI-TIER SYSTEM

The first case study considers the performance analysis and optimization of a multi-class closed queuing network. Initially, we study the behavior of performance metrics such as response times and throughputs as a function of the different mix of customer classes in execution. Then, we determine the load corresponding to the optimal operational point of the system analyzing the bottleneck switching. Corresponding to this load, the ratio of throughput to response time, i.e., the *system power* [5], is maximized. Such analysis are performed with the JMVA and JABA tools.

3.1 The system considered

We consider a closed network consisting of two load-independent queue stations and two classes of customers. Let N be the total number of customers in the system. The network population is given by $\vec{N} = \{N_r\}$, with $\sum_{r=1}^{2} N_r =$ N. A *population mix* is a vector $\vec{\beta} = \{\beta_r\}$ whose components are the fraction of customers in execution that belong to each class. Thus, it we have $\beta_r = N_r/N$.
Let $L_{ir} = V_{ir}\, S_{ir}$ be the total *service demand* of a class c_r customer to resource i, where V_{ir} and S_{ir} are the average number of visits and the average service time of c_r customers at resource i, respectively. The matrix of the service demands considered in the study is shown in Fig.1.

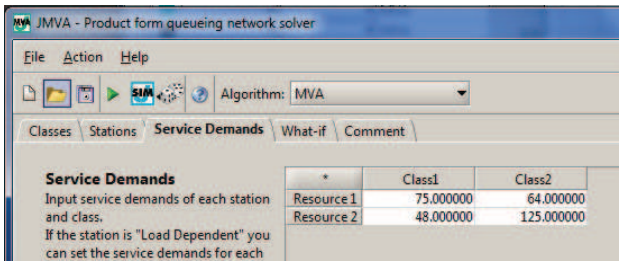

Figure 1: Service demands of the system with two classes of customers and two resources.

For each customer class there is a resource that is most utilized, i.e., the *bottleneck* for that class, and we assume that such resource is different for each class. The bottleneck of c_1 is *Resource 1*, $L_{1,1} = 75$ time units, while *Resource 2* is the bottleneck of c_2, $L_{2,2} = 125$ time units. For a two-station network with this type of workload it is known [6] that there exists a proportion of customers of each class such that both stations are equally utilized for all population sizes. It is also shown that this equiutilization condition provides an *optimal operational point* of the system where the system power or equivalently the sum of the station utilizations is maximized. This means that with this workload we are able to extract from the system the maximum processing capacity. Furthermore, it exists a set of population mixes, i.e., a *common saturation sector* [1], such that both stations saturate when the population size increases.

3.2 The results obtained

We first use the `What-if` feature of JMVA as it allows to derive the values of the performance indices for all the possible mix of jobs in execution. Figure 2 shows the behavior of the throughput at the levels of the global workload (aggregated for the system) and of the single class obtained for a system with two resources, two classes of customers, N=500 customers globally and service demands of Fig.1.
Figure 3 shows the behavior of the response times, at the

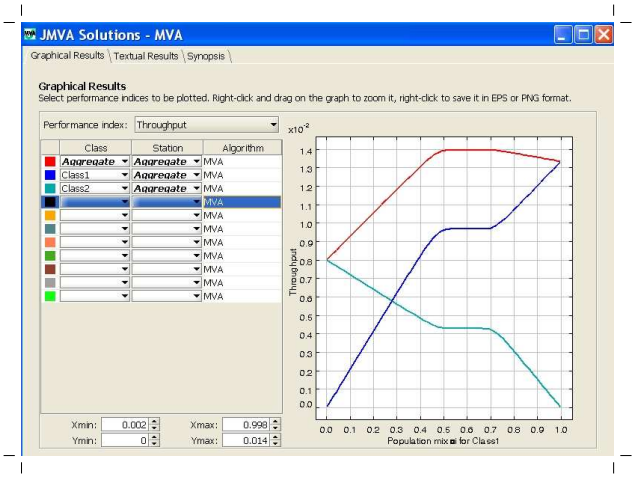

Figure 2: Behavior of the throughput *vs* all possible mix of the two classes of customers.

levels of the global system and of the single class, for the same system. To investigate on the bottleneck migration as a function of the workload we use the JABA tool that is based on the application of the convex polytopes techniques to service demands [3]. Figure 4 shows the bottlenecks as a function of the mixes of classes in execution. The common saturation sector, i.e., the set of population mix that, when the number of customers is sufficiently large, saturate both resources, is delimited by the mixes $\vec{\beta}^{-} = (0.465, 0.535)$ and $\vec{\beta}^{+} = (0.726, 0.274)$. As shown in Fig.2, as N increases for all the mixes included in the common saturation sector the throughput, per-class and of the system, are constant. The optimal operational point, i.e., the equiutilization point, is obtained with the mix [6]:

$$\vec{\beta}^{*} = \left(\beta_1^{*} = \frac{log\,(L_{22}/L_{12})}{log((L_{11}L_{22})/(L_{12}L_{21}))}\ ,\ 1-\beta_1^{*}\right) \qquad (1)$$

According to the values of the service demands of the study, see Fig.1, the equiutilization point is obtained with the mix $\vec{\beta}^{*} = (0.6, 0.4)$. The mix $\vec{\beta} = (0.693, 0.307)$ corresponding to the equiload of the two resources, is clearly highlighted. With this mix, the response times of the two classes coincide with the system response time.

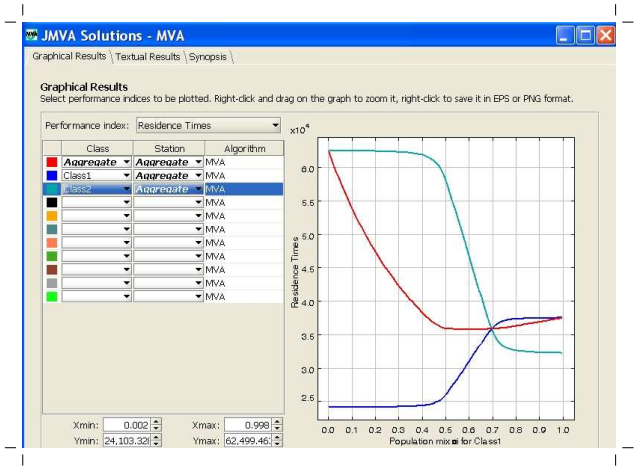

Figure 3: Behavior of the response times *vs* all possible mix of the two classes of customers.

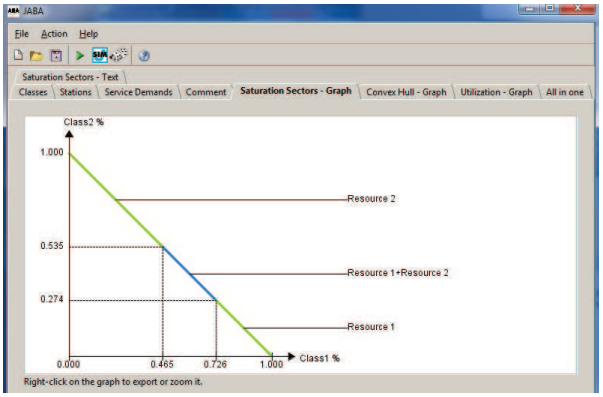

Figure 4: Identification of the bottlenecks as a function of the mixes of classes in execution.

4. REPLICATION POLICIES IN CLOUD ENVIRONMENTS

In the second case study we present an extension of the `What-if` analysis of JMT, which allows to dynamically change the number of resources in a model. We apply this new feature to evaluate different Virtual Machines (VMs) replication policies in a cloud environment subject to Service Level Agreements (SLAs) [4]. Replication techniques partition the load among a number of servers executing the same applications. As a consequence, each server has to handle a reduced flow of requests and the performance are improved. However, in order to reduce energy consumption, the number of replica should be kept at their minimum to satisfy the constraints. Several models, each with a different number of servers, must be investigated to determine the optimal number of replicas.

4.1 Command line for JMVA and Python

JMVA has been designed to be flexible. For this purpose, one important feature is the complete separation between GUI and computation engine obtained through an XML layer, as shown in Figure 5. This architecture allows to reuse the analytic engine in external tools by simply providing a suitable XML input file and calling the required engine with command lines as the following:

```
java -cp JMT.JAR jmt.commandline.Jmt mva <File.xml>
```

where $JMT.JAR$ is the archive of the JMT suite and $< File.xml >$ contains the definition of the model to be analyzed.

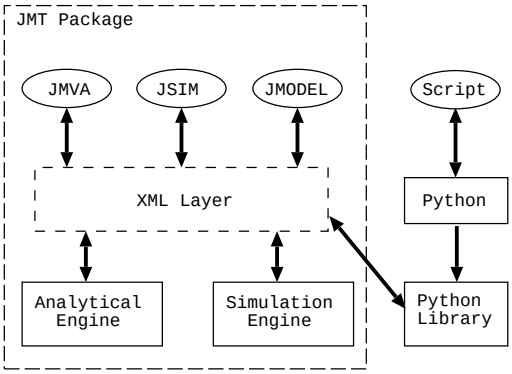

Figure 5: Python script interface.

In this work we integrate JMT with `python` modules. Figure 6 outlines the workflow: the user starts by creating a model with the GUI. The script reads the model produced by the GUI, changes the model, and then evaluates it using the MVA solution engine. Results are then read and evaluated: if the constraints are not met, the model is modified and solved in a new iteration.

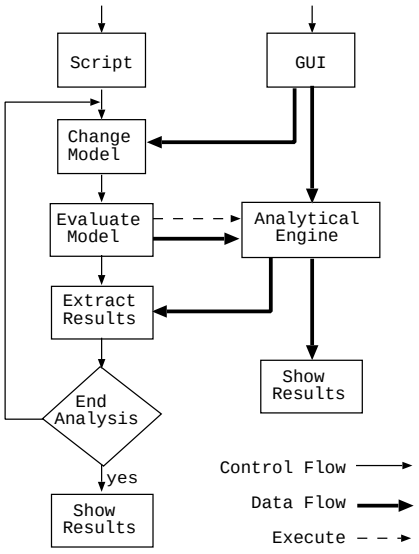

Figure 6: Python scripting workflow.

4.2 The cloud application

We want to study the optimal replication of a three-tier architecture deployed on a cloud environment. Each tier is deployed on one or more dedicated VMs. The workload consists of two classes of jobs, c_1 and c_2, characterized by different service demands at each tier. Each VM can be

replicated, and the workload is evenly partitioned among all the replicas. We can define the configuration of the system as a tuple $\mathbf{K} = (K_1, K_2, K_3)$ where K_i is the number of replicas of the original VM_i. Let $K = K_1 + K_2 + K_3$ be the total number of VMs in the configuration $\mathbf{K}$. The cloud system must be able to satisfy the requests with response times R_{c_1} and R_{c_2} for the two classes, below the two thresholds ρ_1 and ρ_2. For such reason, it adopts a reactive approach: as the number of jobs increases, whenever one of the per-class response time R_{c_i} reaches its threshold ρ_i, the technique adds a new replica of the resource that is bottleneck. We want to evaluate the behavior of this system with N jobs and a population mix $\vec{\beta}$.

The proposed replication scheme is shown in Algorithm 1. The input parameters are the maximum population size of the system N_{Max}, the population mix $\vec{\beta}$ and the two thresholds ρ_1 and ρ_2 (Algorithm 1: line 1). The algorithm iteratively generates and analyses, using MVA, the queueing network with an increasing population size and the given population mix (Algorithm 1: lines 4+5). Then, it considers the per-class residence times (Algorithm 1: line 6): whenever the SLA is violated, it proceeds to replicate the bottleneck server, i.e., the one with the maximum aggregate utilization. As said, the replication implies that the initial demand is shared uniformly among all the servers of the bottleneck server type. The replication is repeated until the SLA satisfaction. The whole process is repeated until the maximum population size is reached.

Algorithm 1 Advanced What-if Algorithm.

1: $INPUT(NMAX, \beta, \rho_1, \rho_2)$
2: $NVals = [1..NMAX]$
3: **for all** $N \in NVals$ **do**
4: $m = GeneratePopulation(\beta, N)$
5: $EvaluateModel(m)$
6: $(R_{c_1}, R_{c_2}) = ExtractResidenceTimes(m)$
7: $success = (R_{c_1} < \rho_1) AND (R_{c_2} < \rho_2)$
8: **while** $NOTsuccess$ **do**
9: $BottleneckServer = IdentifyBottleneck(m)$
10: $m = AddStationServer(BottleneckServer)$
11: $AnalyseModel(m)$
12: $(R_{c_1}, R_{c_2}) = ExtractResults(m)$
13: $success = (R_{c_1} < \rho_1) AND (R_{c_2} < \rho_2)$
14: **end while**
15: **end for**

4.3 Results

Figure 7 presents the results that can be obtained from the proposed advanced JMT `What-if` analysis. The upper part of the Figure shows both the per-class response times R_{c_1} and R_{c_2} achieved by the system and their respective thresholds. The bottom part shows the total number of replicas K and the number K_i of replicas for VM_i needed to satisfy the SLA. For what concern the response times, we see a saw-tooth behavior: without replication, response time increases due to the increasing number of jobs. When the response time of a class reaches its threshold, a new VM is deployed, and the response times of both the classes abruptly drop. The benefits of replication decreases as the number of replica increases. Asymptotically, the response time of class c_1 tends to its threshold ρ_1, while the one of class c_2 tends to a value slightly below ρ_2. This suggests that the constraint on class c_1 is stronger that the one on class c_2. For what concern the number of replicas, we can see a linear trend for all types of VMs: in other words, asymptotically we have $K_{S_i} = \eta_i N$, where η_i is a constant related to the resource.

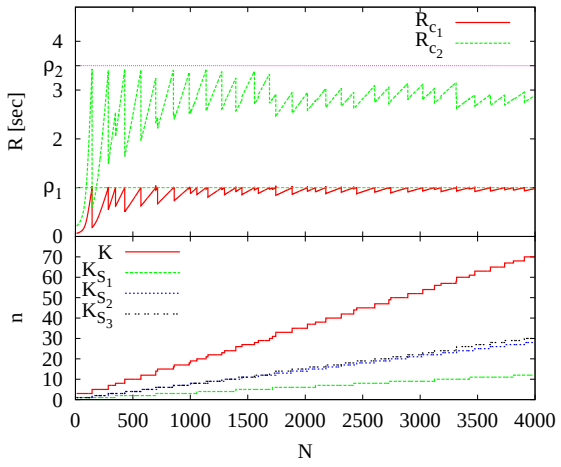

Figure 7: System Response Time per classes (c_1, c_2); ρ_{c_1} = 1 sec; ρ_{c_2} = 3.5 sec for a population mix $\vec{\beta} = (0.3, 0.7)$ and total population up to $N = 4000$ jobs.

5. CONCLUSIONS

In this work, we have proposed two applications of JMT tools to find the optimal operational points of two different systems. Combining the potentials of the tools with the ones `python` scripting, new complex analyses can be carried out, extending the standard applicability of the suite.

6. REFERENCES

[1] G. Balbo and G. Serazzi. Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Performance Evaluation*, 30(1):115–152, 1997.

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

[3] G. Casale and G. Serazzi. Bottlenecks identification in multiclass queueing networks using convex polytopes. In *Proc. of IEEE MASCOTS Symposium*, pages 223–230. IEEE Press, 2004.

[4] D. Cerotti, M. Gribaudo, P. Piazzolla, and G. Serazzi. Matching performance objectives for open and closed workloads by consolidation and replication. *Annals of Operations Research (To Appear 2014)*.

[5] L. Kleinrock. Power and deterministic rules of thumb for probabilistic problems in computer communications. *Proc. International Conference on Communication*, pages 43.1.1–43.1.10, June 1979.

[6] E. Rosti, F. Schiavoni, and G. Serazzi. Queueing network models with two classes of customers. In *MASCOTS*, pages 229–234. IEEE Computer Society, 1997.