Performance Evaluation with Java Modelling Tools: a Hands-on Introduction

Giuliano Casale
Imperial College London
London, UK
g.casale@imperial.ac.uk

Giuseppe Serazzi
Politecnico di Milano
Milan, Italy
giuseppe.serazzi@polimi.it

Lulai Zhu*
Imperial College London
London, UK
lulai.zhu15@imperial.ac.uk

ABSTRACT

The goal of this tutorial is to introduce Java Modelling Tools (JMT), an open source framework for discrete-event simulation and analysis of queueing networks, both product-form and extended, generalized stochastic Petri nets (GSPNs), and queueing Petri nets (QPNs). Thanks to a user-friendly graphical interface, JMT is well-suited to teach performance modeling in academia and to help research students familiarize with classic modeling formalisms used in performance evaluation. The tutorial introduces established and novel features of the JMT suite and illustrates them on case studies.

Keywords

Tutorial, simulation, analysis, queueing networks, Petri nets

1. INTRODUCTION

Java Modelling Tools[1] (JMT) is a tool suite for performance evaluation, capacity planning and workload characterization of computer and communication systems, first released under open source license in 2006 [5]. A number of state-of-the-art algorithms are available for exact, approximate and asymptotic analysis of queueing networks (QNs), stochastic Petri nets (SPNs), and hybrid models combining elements from both. Users can define and analyse models through a rich graphical interface or by means of guided wizards, making the tool appealing in particular for teaching performance evaluation, for performance modeling in the industry, and for research students to experiment with classic performance modelling formalisms. In particular, JMT consists of six applications, of which the most popular ones are *JSIMgraph* and *JSIMwiz*, the discrete-event simulation tools, and *JMVA*, the analytical solver.

JSIMgraph and JSIMwiz offer graphical and wizard-based interfaces to JSIM, the discrete-event simulation engine of JMT. They offer a visual way to specify networks of queues and Petri nets, configure the simulation, and inspect analytical results (averages, distributions, detailed logs). JSIM features heuristics for transient filtering that accelerate the computation of steady-state estimates and confidence intervals, such as MSER-5 and spectral analysis [5].

JMVA is instead an analytical solver for product-form networks. In addition to exact solution methods like mean value analysis, JMVA offers approximate solvers based on fixed-point iteration (e.g., Bard-Schweitzer, Linearizer) as well as normalizing constant based algorithms (e.g., RECAL, CoMoM). The tool also supports load-dependent stations, such as multi-server queues, and what-if analysis.

Other applications within JMT include *JABA*, an analytical solver for bottleneck analysis in product-form queueing networks with up to three service classes, *JMCH*, a visual Markov-chain simulator, and *JWAT*, a tool for workload analysis based on external data files, including clustering analysis and statistical workload analysis.

2. RECENT ADVANCES

2.1 Petri net extensions

The focus of JMT is particularly on QNs, where jobs visit a network of queueing stations demanding a certain amount of service at each visit and experiencing delays due to contention by other jobs. Although QNs are an established class of formalisms for performance modeling purposes, they are not always suitable for describing systems that feature synchronizations.

Recently, an extension has been contributed to JSIM for supporting stochastic Petri nets (SPNs). *Place* and *Transition* nodes can be included in simulation models alongside standard queueing network elements. With these new elements, JSIM supports most canonical types of PNs including Generalized Stochastic Petri Nets (GSPNs), Colored Petri Nets (CPNs) and Queueing Petri Nets (QPNs) [4].

High compatibility has been achieved between PN and QN stations. For example, a job can traverse a queue and later act as a token in a PN transition. The classes of jobs in QNs and the colors of tokens in CPNs are treated identically by JMT. Such tokens can be fired by *multi-mode* transitions, where each mode is specified by an enabling condition, an inhibiting condition, a timing strategy, and a firing outcome.

The simultaneous availability of PNs and QNs also allows in particular the evaluation in JMT of queueing Petri net (QPN) models. QPNs are a hybrid formalism that reconciles PNs with QNs by introducing the *queueing place*, which is comprised of two parts: a queue and a depository [3]. In JMT, a queueing place can be obtained by connecting a queue to a place. Other hybrid structures can also be easily obtained by combining different PN and QN stations

*This work is partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869 (DICE).
[1]JMT homepage: http://jmt.sf.net

IFIP WG 7.3 Performance 2017. Nov. 14-16, 2017, New York, NY USA

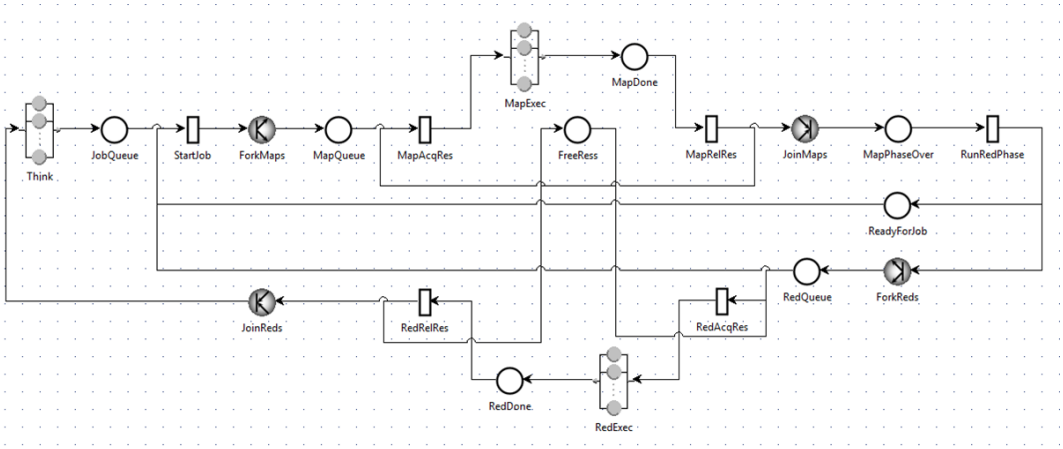

Figure 1: The hybrid model for the YARN capacity scheduler executing a single class of MapReduce jobs.

together. As part of the PN extension, JMT also allows users to import/export PNML models with JSIM. PNML (Petri Net Markup Language) is a proposal of an XML-based interchange format for Petri nets. In the latest release (JMT 1.0.2), JSIM supports import/export in PNML of core models and Place/Transition (P/T) nets. Core models are used for the topologies of PN models, while P/T nets have been extended to represent GSPN models.

2.2 Scheduling

Recent versions of JSIM also offer support for advanced scheduling disciplines. Alongside the classic processor sharing (PS), first-come first-served (FCFS), and last-come first-served (LCFS) policies, JMT now supports also generalized processor sharing (GPS), discriminatory processor sharing (DPS), and non-preemptive policies such as shortest job first (SJF), shortest expected processing time (SEPT), and their counterparts prioritizing longer jobs (LJF, LEPT). GPS and DPS are of great interest in a wide range of applications, for example network protocols [1].

2.3 Parallelization

Recently, JMT has also integrated novel policies for fork/join synchronizations [6]. These include quorum synchronization, which requires to wait for a subset of forked tasks, and guard synchronization, which joins tasks on the basis of the jobs they originated from and their class of service. Such features are useful to model distributed systems, for example Apache Cassandra [7]. An element called semaphore has also been introduced to represent a barrier synchronization among a percentage of tasks that does not alter the execution of the remaining tasks. Such primitive can be useful in modelling the shuffle phase of Apache Hadoop.

2.4 Advanced capacity constraints

JSIM supports since its early releases finite capacity regions (FCRs), which are subnetworks that limit the number and class of jobs within their stations. JSIM has extended this notion with the memory capacity, which consists of a finite pool of memory tokens that get assigned to entering jobs based on their memory weight parameter. This feature allows controlling parallelism not just in terms the number of executing jobs but also based on their weight. Furthermore, JSIM now allows specifying such constraints for groups of classes, as opposed to single classes.

3. CASE STUDY: YARN

YARN is a pluggable scheduler for Apache Hadoop that enables multi-tenant applications to share a large cluster under capacity constraints. We can use JMT to estimate the execution times of MapReduce jobs on the YARN capacity scheduler, based on two models recently proposed in [2]. The QN model proposed in [2] uses a finite capacity region to impose the YARN capacity constraints, but this cannot detail the granularity of the policy implemented by the scheduler. The stochastic well-formed net (SWN) model in [2] can exactly capture the scheduler behavior, but most structures in the model have to be duplicated when multiple classes of MapReduce jobs are running. The JSIM model shown in Figure 1 combines QNs and PNs to address both problems. This can be shown equivalent to the SWN model but avoids the duplication problem in the presence of multiple classes of jobs leveraging the multi-mode transitions of JMT. The simulation of the hybrid and SWN models have been validated against a popular Petri net tool, GreatSPN, matching to a high precision.

4. REFERENCES

[1] S. Aalto, U. Ayesta, S. Borst, V. Misra, and R. Núñez-Queija. Beyond processor sharing. *ACM SIGMETRICS Performance Evaluation Review*, 34(4):36–43, 2007.

[2] D. Ardagna and *et al.* Modeling performance of hadoop applications: A journey from queueing networks to stochastic well formed nets. In *Algorithms and Architectures for Parallel Processing*, pages 599–613. Springer, 2016.

[3] F. Bause. Queueing petri nets: A formalism for the combined qualitative and quantitative analysis of systems. In *PNPM*, pages 14–23. IEEE, 1993.

[4] F. Bause and P. S. Kritzinger. Stochastic petri nets: An introduction to the theory. 2002.

[5] M. Bertoli, G. Casale, and G. Serazzi. Jmt - performance engineering tools for system modeling. *ACM PER*, 36(4):10–15, 2009.

[6] G. Casale and *et al.* Generalized synchronizations and capacity constraints for java modelling tools. In *Proc. of ICPE*, pages 169–170. ACM, 2017.

[7] S. Dipietro and *et al.* A queueing network model for performance prediction of apache cassandra. In *Proc. of VALUETOOLS*. EAI, 2016.