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

[Tutorial]

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

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

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

ABSTRACT

This tutorial focuses on Java Modelling Tools (JMT, http://jmt.sf.net), an open source framework for discrete-event simulation and analysis of queueing networks (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 performance modeling formalisms. 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

ACM Reference format:
Giuliano Casale, Giuseppe Serazzi, and Lulai Zhu. 2017. Performance Evaluation with Java Modelling Tools: a Hands-on Introduction. In *Proceedings of IFIP Performance, New York, NY, USA, Nov (IFIP Performance 2017),* 2 pages.
https://doi.org/

1 INTRODUCTION

Java Modelling Tools (JMT) is an integrated environment for performance evaluation, capacity planning and workload characterization of computer and communication systems, first released under GNU GPL open source license in 2006 [5]. JMT is entirely Java-based and runs on Windows, Linux, and Mac. 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, to promote performance modeling in the industry, and for research students to experiment with classic performance modelling formalisms.

*This work is partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869 (DICE).

IFIP Performance 2017, Nov, New York, NY, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/

1.1 Main features

JMT consists of six independent tools, namely:
JSIMgraph and JSIMwiz. These tools offer graphical and wizard-based interfaces to the discrete-event simulation engine of JMT. They are the most popular tools in the suite as they offer a visual way to specify networks of queues, configure the simulation, and inspect analytical results (averages, distributions, detailed logs). The simulation engine (JSIM) features heuristics for transient filtering that accelerate the computation of steady-state estimates and confidence intervals. More details can be found in [5].
JMVA. This is 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 approximate MVA, Linearizer) as well as normalizing constant algorithms (e.g., RECAL, CoMoM). The tool also supports load-dependent stations and sensitivity analysis (what-if analysis).
JABA. This is an analytical solver for visual bottleneck analysis in product-form queueing networks with up to three service classes.
JMCH. This is a visual Markov-chain simulator rich of animations.
JWAT. This is 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 almost all the 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 determines an enabling condition, an inhibiting condition, a timing strategy, and a firing outcome, defined similarly to the usual PN semantics.

The simultaneous availability of PNs and QNs also allows the evaluation in JMT of queueing Petri net models. QPNs are a hybrid

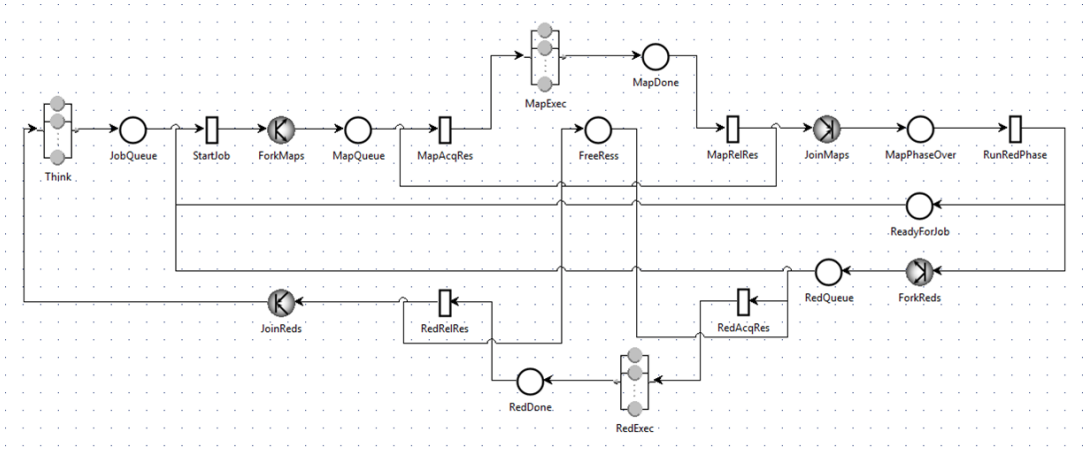

Figure 1: The hybrid model for the YARN capacity scheduler executing a single class of MapReduce jobs.

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 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 current implementation, 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 and parallelization

Recent versions of JSIM also improve 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 among non-preemptive policies 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].

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].

2.3 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 results of the hybrid and SWN models collected by JSIM and GreatSPN[1] match with each other to a high precision.

REFERENCES

[1] Samuli Aalto, Urtzi Ayesta, Sem Borst, Vishal Misra, and Rudesindo Núñez-Queija. 2007. Beyond processor sharing. *ACM SIGMETRICS Performance Evaluation Review* 34, 4 (2007), 36–43.
[2] Danilo Ardagna and *et al.* 2016. Modeling Performance of Hadoop Applications: A Journey from Queueing Networks to Stochastic Well Formed Nets. In *Algorithms and Architectures for Parallel Processing*. Springer, 599–613.
[3] Falko Bause. 1993. Queueing Petri Nets: A Formalism for the Combined Qualitative and Quantitative Analysis of Systems. In *PNPM*. IEEE, 14–23.
[4] Falko Bause and Pieter S. Kritzinger. 2002. Stochastic Petri Nets: An Introduction to the Theory. (2002).
[5] Marco Bertoli, Giuliano Casale, and Giuseppe Serazzi. 2009. JMT - Performance Engineering Tools for System Modeling. *ACM PER* 36, 4 (2009), 10–15.
[6] Giuliano Casale and *et al.* 2017. Generalized Synchronizations and Capacity Constraints for Java Modelling Tools. In *Proc. of ICPE*. ACM, 169–170.
[7] Salvatore Dipietro and *et al.* 2016. A Queueing Network Model for Performance Prediction of Apache Cassandra. In *Proc. of VALUETOOLS*. EAI.

[1] http://www.di.unito.it/ greatspn/index.html