

Automated Multi-paradigm Analysis of Extended and Layered Queueing Models with LINE

Giuliano Casale

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

ABSTRACT

LINE is an open source library to analyze systems that can be modeled by means of queueing theory. Recently, a new major release of the tool (version 2.0.0) has introduced several novel features, which are the focus of this demonstration. These include, among others, an object-oriented modeling language aligned with the abstraction of the Java Modelling Tools (JMT) simulator and a set of native solvers based on state-of-the-art analytical and simulation-based solution paradigms.

CCS CONCEPTS

• **Mathematics of computing** → **Queueing theory**; • **Computing methodologies** → **Modeling and simulation**;

KEYWORDS

Queueing network, steady-state, transient, simulation, stochastic model, JMT, layered queueing network

ACM Reference format:

Giuliano Casale. 2019. Automated Multi-paradigm Analysis of Extended and Layered Queueing Models with LINE. In *Proceedings of Tenth ACM/SPEC International Conference on Performance Engineering, Mumbai, India, April 7–11, 2019 (ICPE '19 Companion)*, 2 pages. <https://doi.org/10.1145/3302541.3311959>

1 INTRODUCTION

LINE (<http://line-solver.sf.net>) offers state-of-the-art performance and reliability analysis methods for systems that can be modeled using queueing theory and stochastic modeling [6]. The goal of the tool is to simplify the computation of quality-of-service metrics in complex systems such as software applications, business processes, and computer networks. LINE decomposes a system model into one or more stochastic models, typically extended queueing networks, that are subsequently analyzed for the desired metrics, such as mean throughputs and mean (or percentiles) of response times.

A new version of the tool (2.0.0) has been recently released, which features primarily two changes from early versions, namely a completely new solution engine supporting multiple solution

methods and an object-oriented modeling language to specify stochastic models with ease, abstracting the model data structures from the particular solution technique chosen for their evaluation. The object-oriented modeling language primarily offers two classes of stochastic models: (i) Network models, which are extended queueing networks, such as open, closed and mixed queueing networks, possibly featuring multiple classes, class-switching, finite buffer capacities, priorities, caching, non-exponential service and arrival distributions, and multiple types of scheduling and routing policies; (ii) LayeredNetwork models, which are layered queueing networks [4], i.e., models consisting of layers, each corresponding to a Network object, which interact with each other through synchronous and asynchronous calls. For both classes of models, LINE also supports the definition of a class of random environments to describe systems that change with the environment (e.g., breakdown and repair).

Main performance metrics returned by LINE include average response times, queue-lengths, throughputs, and utilizations. Such metrics can either refer to an individual station, or when meaningful to the system as a whole (e.g., system response time, system throughput). For response times it is also possible to obtain percentiles, either analytically (typically via mean-field approximation) or via simulation.

LINE has been implemented in MATLAB 2018a and consists of approximately 140 classes. A porting to GNU Octave is currently ongoing. LINE 2.0.0 is open-source and distributed under the BSD-3 license.

2 SOLUTION METHODS

Solvers available in LINE can be either native or external. Each native solver encodes a general solution paradigm and may offer multiple solution methods differing for accuracy and computational cost. Native solvers are described next; for brevity we do not discuss in details external solvers interfaced with LINE such as LQNS [4].

CTMC solver. This is a solver that computes performance metrics in an exact manner by explicit generation of the infinitesimal generator of the continuous-time Markov chain (CTMC) for the model. As the CTMC typically incurs state-space explosion, this solver can successfully analyze only small models through their global balance equations. In order to cope with open models, which typically need infinite state spaces, the CTMC solver uses a truncation approach.

FLUID solver. This solver analyzes the model by means of mean-field approximations, leveraging a representation of the queueing network as a system of ordinary differential equations (ODEs). The resulting fluid model is approximate, but if the servers all use processor sharing or infinite-server scheduling, it becomes exact as the scale of the system grows to infinity [6].

JMT solver. This is a solver that uses a model-to-model transformation to export a LINE model as a JMT simulation (JSIM) or

Work supported by the European Union's Horizon 2020 research and innovation program under grant agreement 825040 (RADON) and by EPSRC under grant EP/M009211/1 (OptiMAM).

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ICPE '19 Companion, April 7–11, 2019, Mumbai, India

© 2019 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-6286-3/19/04...\$15.00

<https://doi.org/10.1145/3302541.3311959>

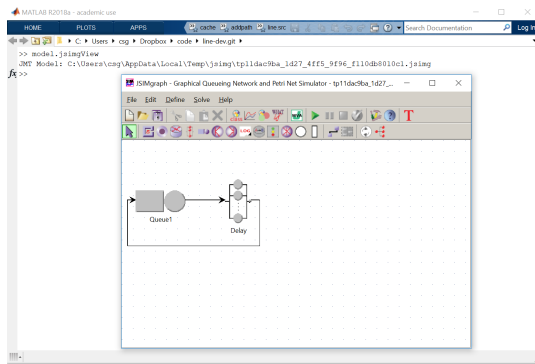


Figure 1: Automated model export from LINE to JMT

analytical (JMVA) model [1]. The JSIM export method can analyze also non-Markovian models, in particular those involving deterministic or Pareto distributions, or empirical traces. A tight integration with LINE is offered, for example it is possible to visualize LINE models through JSIMgraph, JSIMwiz and JMVA, see e.g. Figure 1.

LN solver. LayeredNetwork models are treated in LINE as ensembles of extended queueing networks (i.e., Network objects) and the corresponding native solver is called LN. This solver can use any standard Network solver in LINE (CTMC, FLUID, JMT, ...) to analyze a given layer and its main role is to update the layered parameters after each iteration. Compared to solvers such as LQNS, LN is therefore parametric on the solution technique used in each layer, for example allowing to combine analytical and simulation solvers.

MAM solver. This is a matrix-analytic method solver that relies on quasi-birth death (QBD) processes to analyze open queueing systems with non-exponential arrival or service processes. Fitting functions are supplied to help the user parameterize such processes in terms of phase-type distributions. In LINE 2.0.0, the QBD solver implementation depends on the BUTools library [5].

MVA solver. This is a solver based on approximate and exact mean-value analysis [2]. The solver is best suited to analyze product-form models, but approximations for multi-server stations and non-exponential distributions are also featured.

NC solver. This solver uses a combination of methods based on the normalizing constant of state probability to solve a model. The solver is applicable in particular to models that admit a product-form solution [2] and it is particularly useful to compute marginal and joint state probabilities in queueing network models. Normalizing constants are either computed exactly or via approximations, such as the logistic expansion and logistic sampling methods [3].

SSA solver. This is a discrete-event simulator based on the model state space that, contrary to the CTMC solver, does not require to exhaustive state space generation. Each entity in the queueing network is seen as an agent synchronizing passive and active actions with other agents. The model simulation can be parallelized on multi-core machines.

3 EXAMPLE: PERCENTILE COMPUTATION

We now demonstrate the response time percentiles analysis in a simple model. We begin by instantiating a simple closed model consisting of a delay followed by a processor-sharing (PS) station:

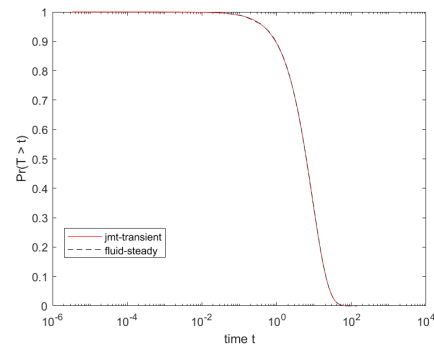


Figure 2: Simulated vs approximated response time tail

```
model = Network('model');
node{1} = Delay(model, 'Delay');
node{2} = Queue(model, 'Queue1', SchedStrategy.PS);
We assume that there exist a single service class consisting of 5
jobs that circulate between the two stations, requiring exponential
service times at both nodes with rates 1.0 and 0.5.
jobclass{1} = ClosedClass(model, 'Class1', 5, node{1}, 0);
node{1}.setService(jobclass{1}, Exp(1.0));
node{2}.setService(jobclass{1}, Exp(0.5));
model.link(Network.serialRouting(node{1},node{2}));
```

We now wish to compare the response time distribution at the PS queue computed analytically through a fluid approximation against the empirical distribution simulated by JMT. To do so, we call the getCdfRespT method on both the Fluid and JMT solvers:

```
RDfluid = SolverFluid(model).getCdfRespT();
RDsim = SolverJMT(model, 'seed', 23000, ...
    'samples', 1e4).getCdfRespT();
```

The returned data structures, RDfluid and RDsim, provide the cumulative distribution function (CDF) value $F(t) = Pr(T \leq t)$, where T is the random variable denoting the response time, while t is the desired percentile. A comparison between the complementary CDF $(1 - F(t))$ obtained through JMT and the corresponding fluid approximation is shown in Figure 2.

It is worth noting that, to produce the results from JSIMgraph in Figure 2, the JMT solver automatically exports the model to XML, adding in the process logger nodes to record the flow of individual jobs, and then importing back the results to estimate the empirical distribution from the logged data.

REFERENCES

- [1] M. Bertoli, G. Casale, and G. Serazzi. 2007. The JMT Simulator for Performance Evaluation of Non-Product-Form Queueing Networks. In *Proc. of the 40th Annual Simulation Symposium (ANSS)*. 3–10.
- [2] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. 2006. *Queueing Networks and Markov Chains*. Wiley.
- [3] G. Casale. 2017. Accelerating Performance Inference over Closed Systems by Asymptotic Methods. In *Proc. of ACM SIGMETRICS*. ACM Press.
- [4] G. Franks, T. Omari, M. C. Woodside, O. Das, and S. Derisavi. 2009. Enhanced Modeling and Solution of Layered Queueing Networks. *IEEE Trans. Software Eng.* 35, 2 (2009), 148–161.
- [5] G. Horváth and M. Telek. 2016. BuTools 2: a Rich Toolbox for Markovian Performance Evaluation. In *Proc. of VALUETOOLS*.
- [6] J. F. Pérez and G. Casale. 2017. Line: Evaluating Software Applications in Unreliable Environments. *IEEE Trans. Reliability* 66, 3 (2017), 837–853.