Scalable Performance Evaluation of Computer Systems

Mirco Tribastone
Institut für Informatik
Ludwig-Maximilian-Universität München, Deutschland
email: tribastone@pst.ifi.lmu.de

November 21, 2011

Abstract

The present paper provides an overview of recent and ongoing research conducted at the Chair of Programming and Software Engineering of LMU Munich on performance evaluation of large-scale computer systems.

1 Introduction

The performance modelling of large-scale systems using discrete-state approaches like continuous-time Markov chains (CTMCs) is fundamentally hampered by the well-known problem of state-space explosion, which causes exponential growth of the reachable state space as a function of the number of the components which constitute the model. Consider for instance a model of a client/server system comprising N clients and M servers, and suppose furthermore that the behaviour is as follows:

- A client communicates with a server to exchange some data.
- Upon receiving the data, the client goes to a state in which it makes use of the data locally.
- The server, instead, goes to a state where it logs some details of the communication.

At some level of abstraction, the overall system may be described by a discrete-state model with descriptor $\xi = (C_e, C_u, S_e, S_l)$, where C_e gives the number of clients that wish to exchange data, C_u denotes the number of clients that are using the data, S_e is the number of servers that are available for transferring the data, and S_l is the number of servers that are logging. A model where the state descripor is a vector of nonnegative integers representing component counts is henceforth called a *population model*.

If we assume that the individual activities are governed by exponential distributions then the models admits a continuous-time Markov chain (CTMC) representation. For instance, let μ be the exponentially distributed transfer rate for a single client/server communication and λ_u and λ_l the individual rates for using the data and logging, respectively. The CTMC model is characterised by the infinitesimal generator Q with the following entries $q(\xi, \xi + j)$, which give the transition rate from state ξ to state $\xi + j$, where $j \in \mathbb{Z}^4$ is called a *jump*:

$$q(\xi, \xi+j) = \begin{cases} \mu \min(C_e, S_e) & \text{if } j = (-1, 1, -1, 1) \ , \\ \lambda_u \, C_u & \text{if } j = (1, -1, 0, 0) \ , \\ \lambda_l \, C_l & \text{if } j = (0, 0, 1, -1) \ . \end{cases} \tag{1}$$

The first transition expresses that the overall transfer rate depends on how many client/server pairs there are in the system at any state. The last two transitions capture the fact that the other activities are independent, i.e., the total rate is given by the individual rate times the number of components capable of performing that activity. This model shows a cyclic behaviour—after using the data and logging, clients and servers move back to a state where they are willing to exchange further information.

Given some initial state $(N, 0, M, 0)$ (i.e., all clients and servers are initially willing to transfer the data), the whole state space of the CTMC with generic transitions as in (1) may be constructed. In this context, the problem of state explosion arises from the cardinality of the state space as a function of N and M, as shown in Table 1. It is not difficult to imagine that explicit enumeration of the state space for realistic models of large-scale systems soon becomes intractable, particularly because of memory costs.

2 Fluid Approximation

In order to tackle the problem of state explosion, we seek an approximation to the CTMC which is independent from the actual population sizes (e.g., N and M). The approximation considered here is a system of coupled ordinary differential equations (ODEs) whose size is equal to the number of *component types* present in the model, hereafter denoted by d. In our running example, there will be $d = 4$ ODEs, one for each element of the population vector. Using a standard approach, we consider the transition rates of a CTMC population model as real functions with domain in $\mathbb{R}^d$. From transitions as in (1), a vector field $V(\xi) : \mathbb{R}^d \to \mathbb{R}^d$ may be constructed by multiplying each (functional) rate by its associated jump, and summing across all jumps. In our running example,

(N, M)	$(10, 10)$	$(20, 10)$	$(50, 50)$	$(100, 100)$	(x, y)
State space size	121	231	2601	10201	$xy + x + y + 1$

Table 1: State space size as a function of initial population levels.

$V(\xi)$ is thus:

$$V(\xi) = (-1, 1, -1, 1)\mu \min(C_e, S_e) + (1, -1, 0, 0)\lambda_u C_u + (0, 0, 1, -1)\lambda_l C_l.$$

Define the initial value problem

$$\frac{dN(\xi, t)}{dt} = V\big(N(\xi, t)\big), \quad \text{with } N(\xi, 0) = (N, 0, M, 0)\,.$$

In coordinates, the differential equation model may be written thus

$$\begin{aligned}
\frac{dN(C_e, t)}{dt} &= -\mu \min\big(N(C_e, t), N(S_e, t)\big) + \lambda_u N(C_u, t)\,, \\
\frac{dN(C_u, t)}{dt} &= +\mu \min\big(N(C_e, t), N(S_e, t)\big) - \lambda_u N(C_u, t)\,, \\
\frac{dN(S_e, t)}{dt} &= -\mu \min\big(N(C_e, t), N(S_e, t)\big) + \lambda_l N(S_l, t)\,, \\
\frac{dN(S_l, t)}{dt} &= +\mu \min\big(N(C_e, t), N(S_e, t)\big) - \lambda_l N(S_l, t)\,.
\end{aligned}$$

The solution

$$N(\xi, t) \equiv \big(N(C_e, t), N(C_u, t), N(S_e, t), N(S_l, t)\big)$$

is the vector of real-valued functions giving the *fluid approximation* to the number of components of each type at time t. For the purposes of this paper, due to space constraints, the interpretation of this approximation may be given as

$$N(C_e, t) \approx \mathbb{E}\left[X(C_e, t)\right], \qquad N(C_u, t) \approx \mathbb{E}\left[X(C_u, t)\right],$$

where $X(\cdot, t)$ denotes the Markov process to be approximated and $\mathbb{E}[\cdot]$ is the usual expectation operator. A fundamental result from Kurtz (with which our expectation interpretation can be shown to be consistent) says that the deterministic trajectory becomes indistinguishable from a sample path of the Markov process as the initial population counts go to infinity, thus justifying this approximation for the analysis of large-scale systems [Kur70].[1]

The nature of the approximation may be appreciated pictorially using the plots in Figure 1, which consider the component C_s of the running example and compare the deterministic trajectory versus the mean trajectory of the Markov process as computed by stochastic simulation (the confidence intervals, within 1% of the mean, are not reported). Figure 1a shows the results for a system with only one client/server pair. Perhaps unexpectedly, the fluid approximation gets quite far off in many points. Instead, increasing the population counts to twenty components for each kind (see Figure 1b) results in a smoother stochastic mean trajectory and a generally better agreement with the fluid approximation. (In this case, a further increase of the population counts to 100 components for each kind led to visually identical trajectories.)

[1]The conditions required for this result to hold are enjoyed by all CTMCs considered in this paper.

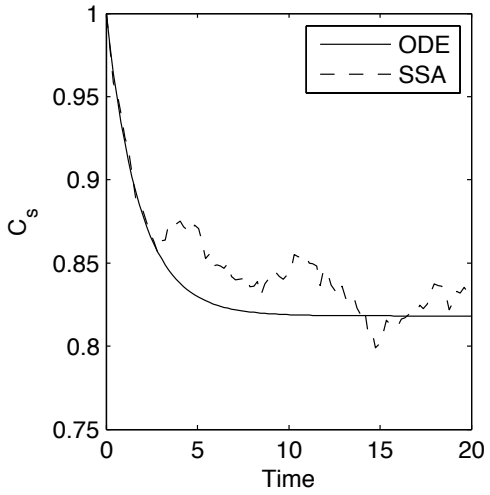
3 Relationship with Process Calculi and Applications

Main theoretical results We applied the fluid framework hitherto described in the context of the stochastic process algebra PEPA [Hil96]. References [TGH10, Tri10b] present a structured operational semantics which interprets a PEPA description as a population CTMC model in the form (1), from which the approximating system of ODEs may be derived.[2] A substantial numerical validation which compared fluid analysis against the numerical solution of the population CTMC model has demonstrated that this approach is accurate in practice and that the quality of the approximation improves sharply with increasing population sizes.

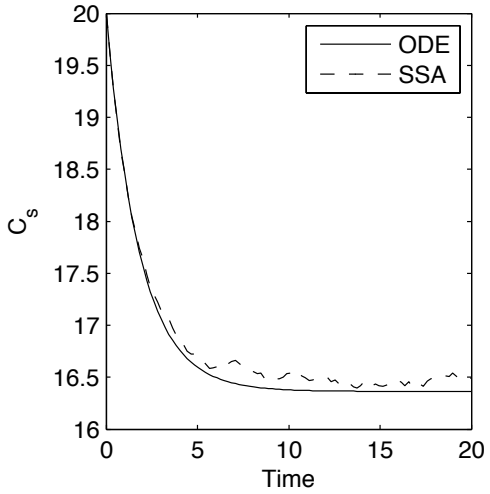
Comparative studies The differential interpretation of PEPA provides a framework that is somewhat analogous to established approximation methods in queueing networks based on mean-value analysis, in the sense that both approaches aim at reducing the computational cost of the analysis by providing estimates for the expected values of the performance metrics of interest. The relationship between these two techniques was examined in more detail in a comparative study between PEPA and the Layered Queueing Network (LQN) model [FOW+09]. General templates of translation of LQN elements into corresponding PEPA components were applied to a substantial case study of a distributed computer system [Tri10a]. A comparison of the approximation accuracy via numerical tests showed that the PEPA fluid analysis is competitive with the LQN approximation methods.

[2]Incidentally, the running example introduced in Section may be modelled in PEPA thus

$$\begin{aligned}
C_e &\stackrel{def}{=} (exchange, \mu).C_u \\
C_u &\stackrel{def}{=} (use, \lambda_u).C_e \\
S_e &\stackrel{def}{=} (exchange, \mu).S_l \\
S_l &\stackrel{def}{=} (log, \lambda_l).S_e \\
System &\stackrel{def}{=} C_e[N] \underset{\{exchange\}}{\bowtie} S_e[M]
\end{aligned}$$



(a) $N = 1, M = 1$



(b) $N = 20, M = 20$

Figure 1: Comparisons between the fluid approximation and stochastic simulation of the running example. Rates were set as follows: $\mu = 0.1$, $\lambda_u = 0.45$, $\lambda_l = 0.55$.

Applications to software performance engineering Importantly, the aforementioned study has highlighted that a process-algebraic approach to performance modelling can concisely capture typical patterns of behaviour that commonly arise in hardware/software systems. These patterns include conditional branching, looping, and fork/join synchronisations. This observation has prompted work toward the integration of performance prediction into model-driven development processes. References [TG08a, TG08b] present work on automatic translations of UML activity and sequence diagrams, respectively, into PEPA models which are amenable to fluid analysis. More recently [GGK+10], research conducted within the context of the EU-funded SENSORIA project has led to quantitative analysis of service-oriented systems described with the UML4SOA profile [MSK08].

Hybrid semantics The fluid semantics of PEPA essentially treats all the entities as continuos, based on assumption that all component types are present in large copies. However, in some cases system may exhibit a form of *multi-scale* phenomenon whereby some components have large populations whereas others have much smaller multiplicities. For instance, the client/server system in our running example may be modified such that it consists of N clients, with N large, and only $M = 1$ server (perhaps abstracting away from the real server multiplicity and considering an equivalent, but faster, single-server case). Thus, the fluid approximation may be found to be too coarse to capture the behaviour of components with small populations.

To deal with such situations, we proposed an hybrid semantics lying between these two extremes, treating parts of the system as discrete and stochastic and others as continuous and deterministic by inducing a binary partition on the set of actions [BGHT10]. The underlying mathematical object for the quantitative evaluation is a stochastic hybrid automaton [Dav93]. By varying which actions are to be considered *continuous*, the modeller has access to models with varying degrees of stochasticity. At the extremes of this lattice we recover the fully stochastic semantics, when all actions are kept discrete, and its deterministic fluid approximation, when all actions are made continuous.

From a practical standpoint, the main advantage of hybrid analysis is that it may provide accurate estimates of distributions in the original model (which cannot be computed with the differential model) orders of magnitude faster than full stochastic simulation. By means of numerical tests on a case study, reference [BGHT10] also provides hints as to where the hybrid approach is particularly advantageous.

Tool support Many of the results present in this paper are available in the *PEPA Eclipse Plug-in Project*, a toolkit for PEPA within the Eclipse framework. The project allows the user to perform steady-state Markovian analysis, stochastic simulation, and fluid analysis of PEPA models [Tri07, TDG09]. Although one of the main points of strength is the integration with the Eclipse platform and its development environment, an application programming interface called *Pepato* can also be used by third-party Java-based applications. The plug-in project is under active development, and a number of extensions are now available that were not discussed in the papers cited above. For instance, with regards to Markovian analysis, software modules have been implemented to support transient analysis and the computation of response-time quantiles.

The tool may be downloaded from `http://www.dcs.ed.ac.uk/pepa/tools/plugin/index.html`.

References

[BGHT10] Luca Bortolussi, Vashti Galpin, Jane Hillston, and Mirco Tribastone. Hybrid Semantics for PEPA. In *Proceedings of the 7th International Conference on Quantitative Evaluation of Systems (QEST)*, Williamsburg, Virginia, USA, September 2010. in press.

[Dav93] M.H.A. Davis. *Markov Models and Optimization*. Chapman & Hall, 1993.

[FOW+09] Greg Franks, Tariq Omari, C. Murray Woodside, Olivia Das, and Salem Derisavi. Enhanced Modeling and Solution of Layered Queueing Networks. *IEEE Trans. Software Eng.*, 35(2):148–161, 2009.

[GGK+10] Stephen Gilmore, Laszlo Gonczy, Nora Koch, Philip Mayer, Mirco Tribastone, and Daniel Varro. Non-Functional Properties in the Model-Driven Development of Service-Oriented Systems, 2010. To appear in *Journal Software and System Modeling*.

[Hil96] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[Kur70] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure Markov processes. *J. Appl. Prob.*, 7(1):49–58, April 1970.

[MSK08] Philip Mayer, Andreas Schroeder, and Nora Koch. A Model-Driven Approach to Service Orchestration. In *IEEE SCC (2)*, pages 533–536. IEEE Computer Society, 2008.

[TDG09] M. Tribastone, A. Duguid, and S. Gilmore. The PEPA Eclipse Plug-in. *Performance Evaluation Review*, 36(4):28–33, March 2009.

[TG08a] M. Tribastone and S. Gilmore. Automatic Extraction of PEPA Performance Models from UML Activity Diagrams Annotated with the MARTE Profile. In *Proceedings of the Seventh International Workshop on Software and Performance (WOSP)*, Princeton, New Jersey, USA, June 2008. ACM.

[TG08b] Mirco Tribastone and Stephen Gilmore. Automatic Translation of UML Sequence Diagrams into PEPA Models. In *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008)*, pages 205–214, Saint-Malo, France, 14–17 September 2008. IEEE Computer Society.

[TGH10] M. Tribastone, S. Gilmore, and J. Hillston. Scalable Differential Analysis of Process Algebra Models, 2010. *Transactions on Software Engineering*, in press.

[Tri07] Mirco Tribastone. The PEPA Plug-in Project. In *Fourth International Conference on the Quantitative Evaluation of Systems*, pages 53–54, Edinburgh, United Kingdom, September 2007. IEEE Computer Society Press.

[Tri10a] Mirco Tribastone. Relating layered queueing networks and process algebra models. In *WOSP/SIPEW '10: Proceedings of the First Joint WOSP/SIPEW International Conference on Performance Engineering*, pages 183–194, New York, NY, USA, 2010. ACM.

[Tri10b] Mirco Tribastone. *Scalable Analysis of Stochastic Process Algebra Models*. PhD thesis, School of Informatics, The University of Edinburgh, 2010.