Symbolic Performance Adaptation

Emilio Incerto
Gran Sasso Science Institute
Viale Francesco Crispi, 7
L'Aquila, Italy
emilio.incerto@gssi.infn.it

Mirco Tribastone
IMT School for Advanced Studies
Piazza S. Francesco, 19
Lucca, Italy
mirco.tribastone@imtlucca.it

Catia Trubiani
Gran Sasso Science Institute
Viale Francesco Crispi, 7
L'Aquila, Italy
catia.trubiani@gssi.infn.it

ABSTRACT

Quality-of-Service attributes such as performance and reliability heavily depend on the run-time conditions under which software is executed (e.g., workload fluctuation and resources availability). Therefore, it is important to design systems able to adapt their setting and behavior due to these run-time variabilities. In this paper we propose a novel approach based on queuing networks as the quantitative model to represent system configurations. To find a model that fits with continuous changes in run-time conditions we rely on an innovative combination of symbolic analysis and satisfiability modulo theory (SMT). Through symbolic analysis we represent all possible system configurations as a set of non-linear real constraints. By formulating an SMT problem we are able to devise feasible system configurations at a small computational cost. We study the effectiveness and scalability of our approach on a three-tier web system featuring different levels of redundancy.

Keywords

Performance-based Adaptation; Queueing Networks; Symbolic Analysis; Satisfiability Modulo Theories

1. INTRODUCTION

In the software development process it is fundamental to quantify Quality-of-Service (QoS) properties, since they represent what end users expect from the software system. Early detection of QoS violations is indeed beneficial since it reduces effort in later development phases where bugs are harder to find and more expensive to fix [33].

Especially at run-time, QoS evaluation of software systems is not a straightforward task since there are many sources of variability and uncertainty that make the process of guaranteeing QoS requirements very challenging. In this paper we consider performance requirements such as response time, throughput, and utilization of resources, for software systems exposed to run-time variabilities such as server failures or changes in workload levels. Our key idea is to explicitly consider such uncertainties as first-class citizens in the analysis process. In particular, we propose a novel use of software performance models [11] that represent not only the system topology (e.g., software components, hardware devices) and parameters (e.g., workload, service rates), but also variability through a symbolic representation of different system configurations [30]. This provides a symbolic evaluation of QoS predictions that can be leveraged at run-time in order to pro-actively adapt the system under study to changing conditions.

Specifically, we consider queuing networks (QN) as performance models because they have been recognized as a well-assessed formalism in the area of software performance engineering [12, 3, 2, 37]. The popularity of QN is also due to the immediate mapping between its constituent elements and features of software systems: (i) a queueing *service center* may either model a software component or a hardware resource; (ii) each service center can composed of multiple *servers*, modeling software concurrent threads or hardware cores; (iii) *routing probabilities* between service stations model the operational profile. Based on previous work by one of these authors [29, 30], allowing symbolic specifications for the number of servers, their service times, and the routing probabilities, it is possible to obtain a closed-form symbolic expressions for typical QN steady-state performance indices such as utilization and response times. The actual performance of a specific QN can then be obtained by simply plugging its concrete parameters into the symbolic expression. Notably, topological variations to a model (such as the removal/addition of a service centre) can be accommodated by allowing certain parameters to be zero.

The previous work [29, 30] was mainly concerned with analyzing many variants of a performance model effectively, but did not consider the problem of exploring these variants in order to find a configuration that satisfies certain QoS requirements. We remark that a naive strategy for doing this may be unfeasible when the parameter space is very large, or even impossible if it is infinite — for instance when allowing rates and probabilities to vary within a range. In our approach, which we call *Symbolic Performance Adaptation* (SPA), we tackle this issue by leveraging the symbolic QN solution as the underlying engine for adaptation. In particular, we consider the solution of [30] as it covers more expressive models than [29] because it allows for multiple-server queuing stations and generally distributed service times.

Technically, the main novel contribution of this paper is to encode the QoS satisfaction problem itself symbolically as a satisfiability modulo theories (SMT) problem [5]. This

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

allows us to formulate statements such as the existence of an assignment for the symbolic variables that satisfy a QoS constraint, specified in first-order logic. The peculiarity of this formulation is that, if the SMT problem is satisfiable, i.e., an assignment exists, then an SMT solver is able to produce a *witness*, i.e., one possible assignment. Instead, in the case the problem is not satisfiable then the modeler has the definite proof that no assignment can ensure the given QoS. This mandates a revision of the system and/or a relaxation of the QoS requirements. In other words, in doing so we exploit the symbolic variables as the run-time "knobs" for guiding the QoS-based adaptation process.

In this way, SPA allows to perform QoS analysis at two stages: (i) at design-time we can provide assurances that the system can meet desired QoS; (ii) at runtime, after detecting changing conditions, these can be encoded as further constraints on the model. These, in turn, trigger a new evaluation of the satisfiability problem, leading to a possible new configuration that meets the newly imposed constraints. Thus, it becomes crucial to evaluate the effectiveness and scalability of SPA in re-configuring the system at runtime. Here we present a numerical evaluation on a realistic QN model of a three-tiered distributed web system with service replications. We consider the challenging situation where the system presents various degrees of freedom in the adaptation capability. SPA can find a re-configuration that satisfies the desired QoS in face of random changes of workloads and injection of faults within few hundred milliseconds.

The remainder of this paper is organized as follows. Section 2 presents related work; Section 3 provides foundational concepts; Section 4 describes SPA by means of an illustrative example; Section 5 shows SPA at work on the three-tier web system; Section 6 points out advantages and limitations of our approach; Section 7 discusses relevant open issues and provides directions for future research.

2. RELATED WORK

A summary of the state-of-the-art and research challenges when developing, deploying and managing self-adaptive software systems is given in [15]. In that respect, we focus on the challenge of supplementing traditional V&V methods in the requirements and design stages of development with run-time assurances. SPA deals with systems changing dynamically at run-time, in fact we use QN models including these uncertainties and providing a symbolic adaptation.

In the literature several approaches have been proposed to deal with the problem of capturing run-time variabilities. There are two main streams of research in this direction: (i) measurements-based approaches that monitor the actual system behavior variations and update parameters of analysis models accordingly; (ii) prediction-based approaches that use model-based predictions to derive missing parameters of system specifications.

Measurement-based approaches. In [45] extended Kalman filters are used to track changes in parameters of QN models. Such filters model the dynamics of parameter changes that are triggered by measuring system performance and updating the performance model when prediction estimates are far from the actual measurements. KAMI (Keep Alive Models with Implementations) models system evolution by run-time adaptation [19] with continuous parameter estimation. Bayesian estimators are exploited to produce updated model parameters enabling continuous automatic verification of requirements at run-time. However, KAMI modifies models through the estimation of numerical parameters but it does not take into account structural changes. Instead, our SPA relies on QN models that anticipate structural modifications, besides others, since the topology of our performance models is symbolically represented. An adaptive filter to accurately learn time-varying transition probabilities of DTMCs is proposed in [22]. It provides robustness to noise and fast adaptation to changes with a low overhead. However, this work focuses on learning and updating input parameters of analysis models similarly to [45]. In [23] authors present an approach for run-time quantitative verification and sensitivity analysis, where the self-adaptation is achieved by bringing software models and probabilistic model checking to run-time, thus to support perpetual automatic reasoning about changes. In principle, all these approaches [45, 19, 22, 23] can be used to keep the parameters of our symbolic model updated. That is, they may represent the input of our adaptation process which can be triggered by a detection of parameter changes.

Prediction-based approaches. In [32] an approach is presented for computing unknown service-demand (SD) parameters in multi-class open QN models. In particular, a closed-form solution is given for the case of a single missing SD parameter; instead, a constrained non-linear minimization problem can be numerically solved for the case of multiple missing SD parameters. This approach is extended in [1] by considering input (i.e., arrival rates for each job class) and output metrics (i.e., average response time for each job class) to estimate the per-class service demands and number of servers for a system QN model. Our approach can be seen as a generalization with respect to the input parameters that can be derived from QN models; indeed as output of the adaptation process SPA may obtain not only service demands and number of servers but other parameters such as routing probabilities. On the other hand, we only consider single class closed QN models.

In [34] the authors present a declarative language (with a formal semantics defined in terms of a discrete-time Markov chain) that allows to explicitly represent uncertainty in the adaptation process by expressing alternative outcomes in the execution of the same adaptation action according to some probability distribution. By contrast, SPA performs adaptation by choosing nondeterministically from the possible feasible configurations returned by the SMT solver (i.e., picking the first witness), however with the guarantee that such a configuration meets the requirements. A case study based on a QN model is used in [34], however to compute the utility of a possible adaptation step it is necessary to analyse the model using traditional means.

The control-theoretic approach of [21] to adapt DTMCs for reliability requirements that need to be preserved over time considers specific changes that may lead to requirements violations (i.e., updates in certain probabilities associated with transitions), and a specific way to attempt adaptation (i.e., generating new values for other probabilities, which represent control variables). Our approach works at a different modeling level since it is concerned with performance instead of reliability, hence the adaptation process derives parameter values of different nature (e.g., number of servers, service demands).

In [28] a technique is presented for exploring the design space of complex systems by combining results from domain-specific languages, symbolic execution, and automated theorem proving in order to quickly move from a specification of a design space to a set of distinctive solutions. Similarly to [28], we use an SMT solver to solve a set of global design constraints and search for valid system instances, however our approach allows to encode both uncertainties and constraints in the SMT problem, thus to derive the adaptations directly from its formulation.

In [27] the authors address the problem of dynamic allocation of resources for cloud-based applications by considering unpredictable workloads. In [35] machine learning techniques, such as Support Vector Machine (SVM) and Neural Networks (NN), were utilized as time-series prediction techniques to model different workload patterns. However, in both approaches [27, 35] the uncertainty is limited to the workload specification. On the contrary, our approach allows to deal with run-time variabilities of different nature.

In previous work by one of these authors [41, 20] uncertainty in performance model parameters was investigated by considering not only the workload, but also the operational profile, software/hardware demand, etc. However, the performance analysis relied on lower/upper buonds provided by software designers. In this work we broaden the scope of our research since no bounds are required, the system configurations fulfilling QoS requirements are dynamically generated on the basis of run-time variabilities.

Active FORmal Models for Self-adaptation (ActivFORMS, [26]) uses an integrated formal model of the adaptation components and knowledge models. It ensures that the adaptation goals that are verified offline are then guaranteed at runtime by supporting dynamic adaptation. However, they use timed automata and timed computation tree logic (TCTL) [44] that do not allow to derive adaptations in a "family-based" fashion, thus leading to complex formulations in case of multiple QoS requirements.

A platform for deployment and autonomic management of web-based applications in cloud is presented in [4]. Performance control is performed with two strategies, i.e., when the system becomes overloaded/underloaded: (i) add/remove one worker to/from the web cluster; (ii) use a performance model to determine the number of worker instances that should be added/removed. In the conducted experimentation the first strategy showed poorer results than the second. This leads to assess the usefulness of our approach that makes use of performance models not constrained with pre-defined changes, in fact the adaptations take place on the basis of run-time conditions.

3. BACKGROUND

Queuing Networks. Queuing networks (QNs) are a class of models that has proven very popular in the software performance engineering community ([12, 13, 18]). They model customers/jobs being routed between different service centers/stations, where they compete for a pool of processing resources. After being served, a job leaves the station and goes to possibly another one according to a probability distribution (routing probabilities). Among the benefits of QN, there is the rather immediate association between their constituent elements and the components of a software system. A service centre may either represent a software device (e.g., a web server) or a hardware resource (e.g., CPU or disk). Routing probabilities, instead, can represent the operational profile of the workload. In order to model parallelism, each station can be composed of multiple, independent and identical servers (e.g., to represent thread concurrency levels for software resources or multiple cores for hardware resources). Service times are described by a probability distribution, where a commonly used one is the exponential. A QN may be closed or open depending on whether or not a fixed population of customers remains within the system.

Here we study closed QNs with a single class of users and service times modeled by Coxian distributions. This is a class of distributions that can be informally considered as a "composition" of exponential *stages*. It has the advantage of being able to approximate any given general distribution arbitrarily closely ([14, 38]). This keeps the stochastic model as a Continuous-Time Markov Chain (CTMC), hence easily amenable to numerical solution and a range of approximate analyses (e.g., [6]), but relaxes the assumption of exponentially distributed service times.

Symbolic Analysis. When *some* of the parameters (e.g., routing probabilities, service rates, etc.) of a QN are left unspecified, the model cannot be solved using conventional numerical algorithms. We make use of a recently presented result [30] which exploits symbolic computation to provide an approximation to steady-state performance indices. This approximation arises from a *fluid limit* result due to Kurtz [31] that allows to write a compact system of ordinary differential equations (ODEs) where, roughly speaking, each variable provides an estimate to the average number of jobs in each service centre. Thus, finding a stationary solution to these ODEs can provide an estimate to the steady-state indices of the QN. In particular, rewriting the stationary point in terms of the throughputs yields a linear system of equations for which a symbolic solution can be obtained. The output is constituted by expressions that depend on symbols representing the unspecified parameters. Plugging in concrete values in these symbolic expressions provides the performance indices of the corresponding QN where all parameters are instead specified.

The main advantage of this analysis is that the computation of the symbolic expression can be re-used among many models that share commonalities while exhibiting differences in topology and/or parameters. Indeed, the results of [30] have been motivated by software product lines applications where the symbolic solution represents the solution of the so-called *150%* model, i.e., an object that subsumes every concrete configuration of a product line model [40].

Satisfiability Modulo Theories. As discussed above, we take the results of [30] as the starting point of this paper. We interpret the symbolic expressions as a set of constraints that the parametric variables have to satisfy, and we augment them with further constraints that formally express QoS requirements. We encode this into a satisfiability problem, and we are able to automatically obtain an assignment of parameters that satisfy these constraints, or a formal certificate that no assignment can satisfy the required QoS.

The encoding is done by means of satisfiability modulo theories (SMT) which check the satisfiability of logical formulas over one or more theories [17]. This is needed because we combine boolean satisfiability with appropriate domains, such as those studied in convex optimization and term-manipulating symbolic systems [7]. More precisely, the

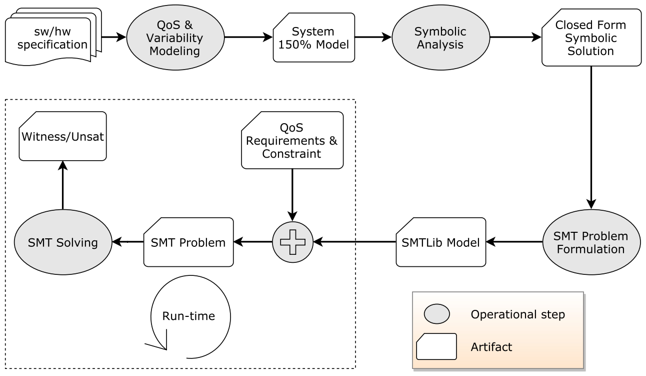

Figure 1: Overview of Symbolic Performance Adaptation.

propositional atoms that form the SAT formulae can be represented by uninterpreted functions, linear inequalities over the rational numbers or bit vector, or other elements belonging to a particular theory. In our case, the constraints involve inequalities of symbols interpreted as real values (e.g., routing probabilities and service rates). For instance, given the formula $3x + 2y - z \geqslant 4 \wedge x \geqslant 0 \wedge y \geqslant 0$ an SMT solver ([10, 16]) looks for an assignment of real variables x, y, z that satisfy the inequalities. The solution is based on the tight integration of propositional SAT solvers with dedicated procedures to reason about the theory component. It combines the power of the well-known constraint programming techniques such as linear or nonlinear programming with first-order-logic [9]. We exploit this combination to model quantitative properties of the systems.

4. THE APPROACH

Overview. Our idea is to see a self-adaptive system as a highly configurable one, which is able to change its configuration at run-time in response to internal or external events. In this view, the role of self-adaptation becomes finding a suitable system configuration such that QoS requirements are continuously fulfilled. To this end, we define a suitable SMT encoding of the symbolic solution of the QN model augmented with QoS requirements. Then, we use an SMT solver, specifically the well known Z3 [16] in our case, for devising a feasible system configuration. This allows us to enable the run-time adaptation capabilities of the system through a white-box approach.

Figure 1 depicts an overview of SPA, where the grey ellipse boxes represent operational steps while the rounded rectangles identify different kind of artifacts (e.g., models). The first step is to model the QoS properties of the system using a symbolic version of QNs that allows designers to single out the adaption knobs that are accessible in the system. Instantiating these knobs with concrete variables leads to all possible system configurations, compactly and symbolically expressed in a single model. Then, symbolic evaluation yields a set of real non-linear equations (i.e., the symbolic solution). Finally, augmented with QoS requirements and constraints, we formulate an SMT problem that, once solved, is able to efficiently devise feasible system configurations, or to assess their absence. At run-time, the SMT problem is continuously updated with monitored information (e.g., workload fluctuations) in order to devise feasible system configurations taking into account the actual run-time variabilities. We remark that an adaptation knob represents a naturally continuous variable such a routing probability, and the whole configuration space becomes uncountably infinite. Yet the power of the SMT encoding will allow us to make formal and precise statements about the existence (or the absence) of a feasible configuration.

In the remainder of this section we explain in detail all the operational steps belonging to our approach. For clarifying the discussion we use a Load Balancing System (LBS) as a running example.

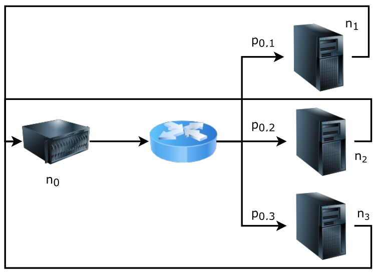

Figure 2: Running example: LBS with four nodes.

Running example. In performance engineering load balancing is a very well-known and established design technique [39]. Its goal is to distribute workloads across multiple computing resources, such as computers, clusters, network links or disk drives. It is useful to optimize resource usage, maximize throughput, minimize response time, and avoid overload of any single resource. It is recognized as the key ingredient for implementing redundancy pattern suitable to improve system's QoS characteristics.

Figure 2 depicts one possible instance of load balancing system that is composed of four nodes. Nodes n_1, n_2, and n_3 represent service nodes devoted to the actual processing of the incoming requests, while n_0 is the scheduler node responsible for forwarding each request to a particular service node according to the probabilities $p_{0,1}$, $p_{0,2}$, and $p_{0,3}$, respectively. For such systems self-adaptation is a fundamental technique allowing them to continuously meet their QoS requirements while executing in an uncertain environment. For instance, it could be necessary to recompute at run-time these probabilities due to stations or links faults, or to scale-up service node resources (e.g., CPU speed) due to workload peaks. Our goal is to endow such systems with self-adaptation capabilities, thus to efficiently explore the system's configurations space guided by QoS requirements (e.g., response time upper bound) and resource design constraints (e.g., maximum number of replication).

4.1 Symbolic QN

We start with a closed single class QN with multiple and independent servers. This is formally specified by a set of stations denoted by $S = \{0, 1, \ldots, n\}$ and by the following parameters:

- s_i denotes the server multiplicity at the i-th station, with $i \in S$;
- e_i is the service time distribution at the i-th station, with $i \in S$; for an arbitrary Coxian distributed service

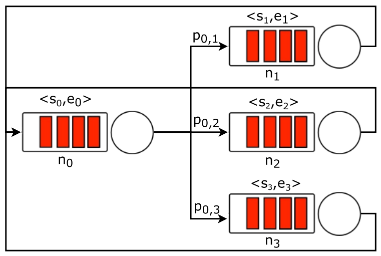

Figure 3: LBS Queuing Networks Model.

time, e_i is a pair $e_i = (\mu_i, \pi_i)$ where μ_i is a vector of rates, specifying the exponential rate at each stage, and π_i is a vector of probabilities (of the same length as μ_i), specifying the probability with which service is ended at each stage. To model exponential distributions, for simplicity we let e_i simply denote the mean service time.

- $p_{i,j}$ is the *routing probability*, i.e., the probability with which a request served in station i will be routed to station j;
- C is the number of clients that are simultaneously interacting with the system.

These parameters are enough to solve a QN model using standard techniques [6], which provide typical performance indices such as *throughput*, i.e., how many jobs per unit time are processed by a station; *utilization*, i.e., the probability that a server is busy; and *response time*, i.e., how it takes for a job to traverse a routing path in the QN.

The QN for the LBS system, with usual graphical notation, is shown in Figure 3.

To explicitly include variability, the modeler identifies parameters to be left unspecified in the form of *symbolic variables*. These will represent the "knobs" of the model that can be changed at run-time. Following [30], we set up a linear system of equations based on the routing probability matrix $P = (p_{ij})_{i,j \in S}$ as follows:

$$P'\mathcal{T} = \mathcal{T}, \qquad \mathcal{T} = (\mathcal{T}_i)_{i \in S} \tag{1}$$

where $'$ denotes matrix transposition and $\mathcal{T}$ is the vector of unknown steady-state *throughputs*, for each queuing station.

It is worth to remark that due to the unspecified parameters, these equations can only be solved symbolically using a computer algebra system. We obtain a symbolic expression for all throughputs $\mathcal{T}_i$, computed exploiting the following theorem:

Theorem 1. *Let QN_s be a parametric QN model of the system S, and let P be the routing probability matrix of QN_s with (1) its traffic equations. Denote by ζ the unique solution of (1) when $\zeta_1 = 1$. Then for all $i \in S$,*

$$\mathcal{T}_i = \begin{cases} \zeta_i \left(\sum_{j \in S} e_j\, \zeta_j\right)^{-1} C & \textit{if } (2) \\ \zeta_i \, \min\{s_j\, (e_j\, \zeta_j)^{-1} | j \in S \wedge s_j < \infty\} & \textit{else.} \end{cases}$$

$$s_i > \frac{e_i\, \zeta_i\, C}{\sum_{j \in S} e_j\, \zeta_j}, \qquad \forall i \in S \tag{2}$$

For illustration purposes we consider the running example with the parameter choice summarized in Table 1, where the symbol '—' refers to the variable parameters. For simplicity, we assume exponentially distributed service times. With this, the system of equations (1) reads

Table 1: Parameter choices for the running example; '—' indicates that the variable is considered symbolically.

Parameter	**Value**	**Parameter**	**Value**
e_0	—	s_0	4
e_1	0.1	s_1	—
e_2	—	s_2	30
e_3	2.0	s_3	—
$p_{0,1}$	—	$p_{0,2}$	—
$p_{0,3}$	—	C	—

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ p_{0,1} & 0 & 0 & 0 \\ p_{0,2} & 0 & 0 & 0 \\ p_{0,3} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathcal{T}_0 \\ \mathcal{T}_1 \\ \mathcal{T}_2 \\ \mathcal{T}_3 \end{pmatrix} = \begin{pmatrix} \mathcal{T}_0 \\ \mathcal{T}_1 \\ \mathcal{T}_2 \\ \mathcal{T}_3 \end{pmatrix}$$

where $p_{0,1}$, $p_{0,2}$, and $p_{0,3}$ are the symbolic parameters. Applying the theorem to the running example yields:

$$\mathcal{T}_0 = \begin{cases} \dfrac{C}{(e_0 + \frac{p_{0,1}}{10} + 2p_{0,3} + e_2\, p_{0,1})} & \text{if } (*) \\ \min\left\{\dfrac{4}{e_0}, \dfrac{10 s_1}{p_{0,1}}, \dfrac{30}{e_2\, p_{0,2}}, \dfrac{s_3}{(2p_{0,3})}\right\} & \text{else,} \end{cases} \tag{3}$$

$$\mathcal{T}_1 = \begin{cases} \dfrac{C\, p_{0,1}}{(e_0 + \frac{p_{0,1}}{10} + 2p_{0,3} + e_2\, p_{0,1})} & \text{if } (*) \\ p_{0,1} \min\left\{\dfrac{4}{e_0}, \dfrac{10 s_1}{p_{0,1}}, \dfrac{30}{e_2\, p_{0,2}}, \dfrac{s_3}{(2p_{0,3})}\right\} & \text{else,} \end{cases}$$

$$\mathcal{T}_2 = \begin{cases} \dfrac{C\, p_{0,2}}{(e_0 + \frac{p_{0,1}}{10} + 2p_{0,3} + e_2\, p_{0,1})} & \text{if } (*) \\ p_{0,2} \min\left\{\dfrac{4}{e_0}, \dfrac{10 s_1}{p_{0,1}}, \dfrac{30}{e_2\, p_{0,2}}, \dfrac{s_3}{(2p_{0,3})}\right\} & \text{else,} \end{cases}$$

$$\mathcal{T}_3 = \begin{cases} \dfrac{C\, p_{0,3}}{(e_0 + \frac{p_{0,1}}{10} + 2p_{0,3} + e_2\, p_{0,1})} & \text{if } (*) \\ p_{0,3} \min\left\{\frac{4}{e_0}, \frac{10 s_1}{p_{0,1}}, \frac{30}{e_2\, p_{0,2}}, \frac{s_3}{(2p_{0,3})}\right\} & \text{else,} \end{cases}$$

$$(*) = \begin{cases} 4 > (C\, e_0)/(e_0 + \dfrac{p_{0,1}}{10} + 2\, p_{0,3} + e_2\, p_{0,2}) \\ s_1 > (C\, p_{0,1})/(10\, (e_0 + \dfrac{p_{0,1}}{10} + 2p_{0,3} + e_2\, p_{0,2}) \\ 30 > (C\, e_2\, p_{0,2})/(e_0 + \dfrac{p_{0,1}}{10} + 2p_{0,3} + e_2\, p_{0,2}) \\ s_3 > (2\, C\, p_{0,3})/(e_0 + \dfrac{p_{0,1}}{10} + 2p_{0,3} + e_2\, p_{0,2}) \end{cases}$$

We remark that the symbolic QN representation allows to encode arbitrarily complex and structural changes to the system. For instance, by setting $p_{0,1} = 0$ and $s_1 = 0$ it is possible to "power down" one server, which can model either a power failure or a downscaling of resources due to a lower workload. Moreover, we can express additional performance indices of the system over the specified symbolic variables such as $\mathcal{U}_i = \frac{\mathcal{T}_i e_i}{s_i}$ modeling in this way the utilization ratio at the i-th queuing station.

The important feature of this construction is that the performance of a concrete configuration, e.g., setting $C = 20, s_1 = 3, s_3 = 4, e_0 = 5, e_2 = 6, p_{0,1} = p_{0,2} = p_{0,3} = \frac{1}{3}$, can be obtained by simply plugging in these values in (3). For this case, it would yield

$$\mathcal{T}_0 = 0.80 \qquad \mathcal{T}_1 = 0.26 \qquad \mathcal{T}_2 = 0.26 \qquad \mathcal{T}_3 = 0.26.$$

4.2 SMT Problem Formulation

We observe that all symbolic expressions can be expressed in non-linear real arithmetic theory ($\mathcal{NRA}$). Thus, our idea is to encode the system's symbolic solution, combined with the QoS requirements and resources constraints, in an SMT problem suitable to efficiently devise feasible system configuration. Our formulation reflects Zave and Jackson's seminal work on bridging the gap between requirements and specifications [46]. More precisely, let $\mathcal{M}$ be the symbolic solution of the QN, let $\mathcal{Q}$ be a set of QoS requirements (e.g., an upper bound on the response time), $\mathcal{D}$ be a set of domain assumptions (e.g., workload levels), and $\mathcal{R}$ be a set of resource constraints (e.g., at most 10 servers are available). We encode $\mathcal{M}, \mathcal{Q}, \mathcal{D}$ and $\mathcal{R}$ as first-order-logic formulae over the $\mathcal{NRA}$ theory such that the QoS-based adaptation is turned into the satisfiability problem which we call the *adaptation problem*, $\mathcal{AP}$:

Find an assignment of the variables for model $\mathcal{M}$ that ensures $\mathcal{Q}$ subjected to constraints $\mathcal{D} \wedge \mathcal{R}$.

The SMT model encoding the symbolic solution $\mathcal{M}$ for the load balancing system (see Figure 3) is publicly available[1].

To completely specify the problem we consider in $\mathcal{M}$ the QoS requirements:

$$\mathcal{Q} := \left\{ \mathcal{T}_0 \geqslant \frac{C}{2} \right\},$$

the domain assumptions

$$\mathcal{D} := \{1 \leqslant C \leqslant 350\},$$

and the resource constraints

$$\mathcal{R} := \Big\{ \forall i, j \in S. \sum_{k \in S} p_{i,k} = 1.0 \wedge \big(0 \leqslant p_{i,j} \leqslant 1.0 \wedge 1 \leqslant s_i \leqslant 40 \wedge 0.02 \leqslant e_i \leqslant 10\big) \Big\}$$

This formulation requires that the ratio between the number of clients and the throughput of the reference station must be lower or equal than 2 time units (e.g., sec, ms) while the system is serving form 1 up to 350 users with available system resources spanning from 1 to 40 servers for each service station and a service time between 0.02 and 10 time units. Moreover, the formulation enforces that the symbols modeling the routing probabilities are associated to values that make the QN a valid closed one.

By embedding in $\mathcal{AP}$ the actual run-time values for $\mathcal{D}$ we are able to devise feasible system configurations that can be used for avoiding QoS violations consuming the available resources constrained by $\mathcal{R}$. Hereafter we report some numerical results about the solution of adaptation problem $\mathcal{AP}$ formulated in the previous section. We evaluated the self adaptation capabilities of the load balancing system for a series of 350 self-adaption steps. For each of them we generated a linear increasing workload C satisfying the domain assumptions $\mathcal{D}$ and collecting the information about the throughput of the scheduler node $\mathcal{T}_0$ and the self adaptation overhead (i.e., needed computation time).

[1]http://rise4fun.com/Z3/1jA8

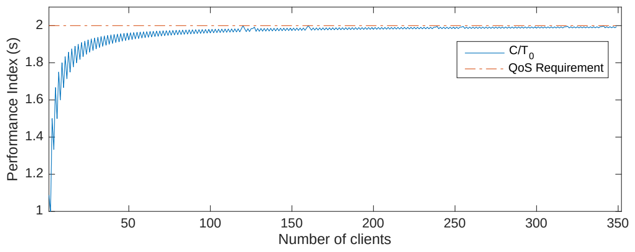

Figure 4: LBS $\mathcal{Q}$ Fulfillment.

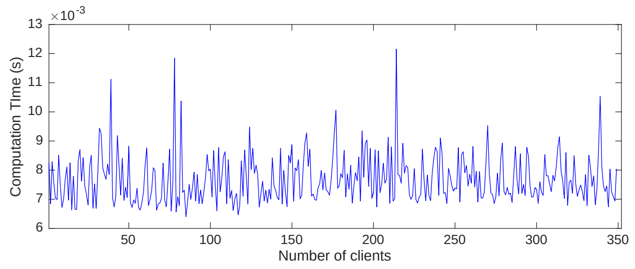

Figure 5: Self-adaption computation time

Figure 4 shows the effectiveness of our approach for allowing an LBS system to continuously meet its QoS requirements $\mathcal{Q}$ (red dotted line) despite a highly variable workload. In Figure 4 we can see that the blue line (i.e, $\frac{C}{\mathcal{T}_0}$) is always below its threshold (i.e., 2), thus assessing the fulfillment of $\mathcal{Q}$ for any value of the workload C. Instead, Figure 5 shows the computation time needed by the load balancing system to recompute a new configuration. For this model the solver was very efficient, yielding a new configuration in a few milliseconds on average.

We remark that the effectiveness of the SMT solver may depend on the choice of the domain used to encode inherently integer parameters such as server multiplicities and client populations. Declaring them as integers leads to nonlinear mixed integer constraints that may turn out to be more difficult to handle than using reals. In fact, in this case Z3's *nlsat* is a decision procedure for such formulae (that belong to the quantifier-free fragment of first-order logic), so the solver will return an answer (given enough resources). In our tests we chose an encoding into reals, appropriately rounding after the solution was obtained.

This is also somewhat justified by the inherent assumptions made for deriving closed form symbolic expressions, whereby one considers the fluid limit of large enough copies of servers and large populations of jobs. It is not difficult to show that the same systems of equations are valid for variables that represent *population densities*, i.e., proportions of server multiplicities for each client in the system [31]. Then, one would equivalently consider an SMT encoding where all variables are naturally reals, and then interpret the solution through rounding to an integer afterwards.

5. CASE STUDY: THREE TIER SYSTEM

In this section we evaluate the effectiveness of SPA on a more substantial case study. We consider a model of a three-tier system consisting of a data-management tier (encompassing one or several database servers), an application tier (business logic), and a client tier (interface functionality). We study the possibility of load balancing within tiers, especially for the application tier, by distributing requests across the different application server instances.

All the numerical results shown in this section are available for replicability at https://goo.gl/wmmnCK.

Model description. Our case study is based on a QN model presented and successfully validated against a real system in [42]. The QN is shown in Figure 6, where we can see that the model is composed of tiers T_1, T_2, T_3 plus a delay station d_s used for emulating the session-based workload typical of web applications. We modeled d_s as a queuing center with a high number of servers such that at least one is always available for a client. With this consideration its thinking time Z becomes the server service time. Each of the three tiers has a load balancing system with possibly different degrees of replication. In Figure 6 we consider the case where each tier has three replicas.

When a request arrives at tier T_i it triggers one or more requests at its subsequent tier T_{i+1} (e.g., a database search that triggers multiple queries at different product catalogs). This phenomenon is captured by allowing a request to make multiple visits to each of the queues during its overall execution. This is achieved by introducing a transition from each tier to its predecessor. A request, after some processing at tier T_i, either returns to T_{i-1} with a certain probability $p_{i,j}$ or proceeds to T_{i+1} with probability $(1 - p_{i,j})$. At the tier T_3 all requests return to T_2, while at the first tier T_1 a transition to the delay station denotes request completion. For a complete list of the system's parameters together with their semantics we refer the reader to the description of the load balancing system used as the running example of Section 4.

Throughout the remainder, without loss of generality, we fixed to 0.5 the probabilities that tier i makes multiple requests to tier $i+1$ and $Z = 0.2$, i.e., we assume that the operational profile of the user is known.

Experimental set-up. As said in Section 4, once QoS properties of the system and its variabilities are symbolically modeled, then a set of non linear real equations is derived from them. This represents the symbolic solution $\mathcal{M}$ for the throughput of each station in the system. By encoding $\mathcal{M}$, augmented with QoS requirements $\mathcal{Q}$, resource constraints $\mathcal{R}$, and domain assumptions $\mathcal{D}$, in a set of first-order-logic formulas we are able to formulate an SMT problem used as the core engine for the self-adaptation process.

In all our tests, we arbitrarily set $\mathcal{Q}$ to represent the requirement that the average response time for a user is no larger than 1.0 time units, together with the condition that all working servers operate at a utilization between 0.6 and 1.0. This encodes a trade-off between competing requirements forcing the solver to avoid trivial solutions: without a condition on the utilizations, the solver could have found an assignment that oversizes the system, at the cost of having servers that are inefficiently working at very low utilization levels. Furthermore, it is worth to point out that our utilization constraint can be interpreted as a QoS requirement related to energy efficiency by relating servers utilization to energy consumption, as done in [8].

In the following we formally specify these constraints. By taking station 1 as the reference station and applying Little's law, we obtain that the response time is given by $C/\mathcal{T}_1$.

```
(assert
    (and
        ;Constraints set
        $\forall i \in S$ (>= $s_i$ 1)
        $\forall i \in S$ (<= $s_i$ 40)
        $\forall i \in S$ (>= $e_i$ 0.06)
        $\forall i \in S$ (<= $e_i$ 10)
        ;Routing probabilities constraints
        $\forall i,j \in S$ (<= $p_{i,j}$ 1.0)
        $\forall i,j \in S$ (>= $p_{i,j}$  0.0)
        $\forall i \in S$ (= $\sum_{j \in S} p_{i,j}$ 1.0)
        ;Domain assumptions
        (>= C 1)
        ;System throughput
        $p_{i,j}, e_i, s_i, k \in S$ $\mathcal{T}_k = f(p_{i,j}, s_i, e_i)$
        ;QoS requirements
        (>= $\mathcal{T}_1$ C)
    )
)
```

Listing 1: SMT problem for the three tier system.

Hence, $\mathcal{T}_1 \geqslant C$ encodes our desired response time constraint (i.e., $\leqslant 1.0$). Recalling that $\mathcal{U}_i$ denotes the utilization at queuing station i (again, a subformula from $\mathcal{M}$), we get

$$\mathcal{Q} := (\mathcal{T}_1 \geqslant C) \wedge \forall i \in S. \big((\mathcal{U}_i \leqslant 1.0 \wedge \mathcal{U}_i \geqslant 0.6) \vee \mathcal{U}_i = 0.0 \big),$$

where the condition $\mathcal{U}_i = 0.0$ allows the solver to explore the possibility not to activate one or more server replicas in a tier. Finally we complete the adaptation problem formulation by specifying $\mathcal{D} \wedge \mathcal{R}$ as

$$\mathcal{D} \wedge \mathcal{R} := (C \geqslant 1) \wedge \mathcal{R}_1 \wedge \mathcal{R}_2$$

where

$$\mathcal{R}_1 := \forall i,j \in S. \big(0.0 \leqslant p_{i,j} \leqslant 1.0 \big) \wedge \forall i \in S. \left(\sum_{j \in S} p_{i,j} = 1.0 \right),$$

$$\mathcal{R}_2 := \forall i \in S. \big((1.0 \leqslant s_i \leqslant 40.0) \wedge (0.06 \leqslant e_i \leqslant 10.0) \big),$$

Here $\mathcal{R}_1$ imposes that the symbolic variables $p_{i,j}$ are actual probabilities, such that the sum across all outgoing routing probabilities from a node is equal to 1. Instead, $\mathcal{R}_2$ gives further constraints on the service multiplicities and on the service times. To better describe our formulation, we report the initial specification of the case study, in Z3-like pseudo-code, in Listing 1. This will be expanded with further constraints to represent the runtime changes, as discussed hereafter.

Numerical results. We evaluate the effectiveness and efficiency of our approach by simulating adaptation steps on the QN model. Each step triggers the solution of the satisfiability problem by varying the workload C and simulating link/server faults by enforcing to 0.0 one or more of the routing probabilities $p_{1,4}$, $p_{5,8}$, and $p_{9,12}$. More specifically, we considered 300 adaptation steps where at each step we picked a linear increasing workload C in the range $[5, \ldots, 304]$, and we imposed uniformly at random to apply one of the following constraints with the idea of emulating from 0 up to 3 station or link faults: $\mathcal{A}_1$) *true*, i.e., no change to the routing probabilities; $\mathcal{A}_2$) $p_{1,4} = 0.0$, $\mathcal{A}_3$) $p_{1,4} = 0.0 \wedge p_{5,8} = 0.0$; $\mathcal{A}_4$) $p_{1,4} = 0.0 \wedge p_{5,8} = 0.0 \wedge p_{9,12} = 0.0$. We remark that at each iteration the internal status of the used SMT solver is cleared in order to ensure that it did not reuse any assignment computed at the previous step.

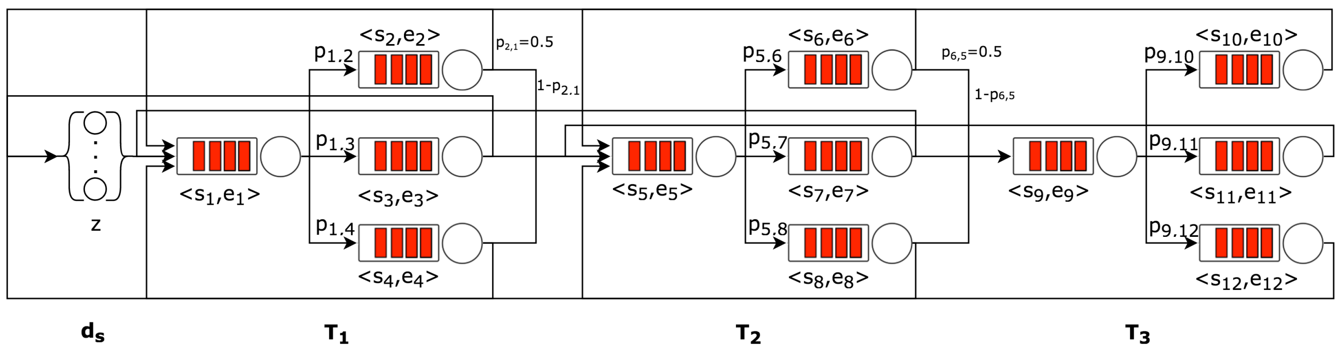

Figure 6: QN model for a three-tier system based on [42], where each tier has three replicated servers.

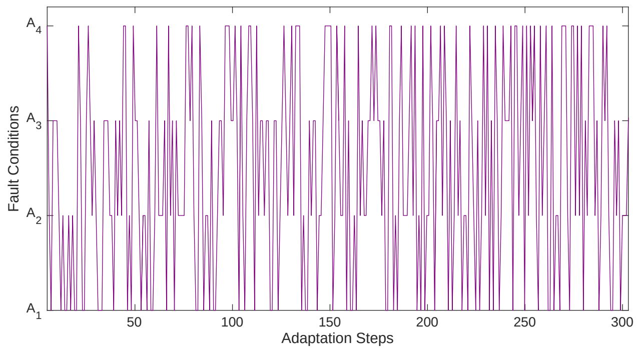

Figure 7: Number of faults for each adaptation step.

Figure 7 shows the traces of the adaptation steps considered in our experiment, where each j-th step added the constraint $(C = j) \wedge \mathcal{A}_i$ for $j \in 5, \ldots, 304$ and $i = 1, 2, 3, 4$, to the model reported in Listing 1.

To understand the efficiency of SPA, we calculated the computational time required by the operational steps of *symbolic analysis* and *SMT problem formulation* (see Figure 1) for our three tier system. We experienced that these two steps were executed with an average of 4.85 s, however we remark that this initial overhead is required once since the run-time adaptations apply to the same SMT problem formulation.

Figures 8, 9, and 10 show the numerical results of our experiments, executed on an ordinary laptop with a 2,6 GHz Intel Core i5 and 8 GB of RAM. On the x axis is reported the number of clients C at each adaptation step. The response time for each found configuration (Figure 8) and the utilization levels of the system (Figure 9) always satisfy the requirements (indicated by the red dotted line) even if in the system some faults occur (Figure 7). For presentation issue, Figure 9 shows only the maximum and minimum utilization levels measured at each adaptation step since they are enough for assessing the fulfillment of the QoS requirements. Moreover, the required computation time for the adaptation (Figure 10) shows that our approach can efficiently solve this problem within few hundred milliseconds in the worst case.

To evaluate the scalability of SPA, we considered the same experiment (i.e., managing symbolically all the system parameters s_i, e_i, p_{ij} constrained by the same $\mathcal{Q} \wedge \mathcal{R}$) on larger variants of the system characterized by increasing replication levels up to six replicas per tier. Figure 11 reports the average computation time (in seconds) across all adaptation steps for each replication level (where the case of 3 replicas summarizes the results previously reported). The complexity of the adaptation process grows up while increasing numbers of replicas per tier (adding a replica for each tier means to add in the model a number of new parameters equals to 3 times the number of tiers), but still remains reasonable for meaningful system configurations. However, we remark that all the results presented in this section are obtained for the extreme cases in which *all* the system parameters are treated symbolically. Reducing this variability can significantly lower the complexity of the adaptation process, making our approach scalable to larger models. Finally we evaluated when the adaptation problem is unsatisfiable since it may take longer than providing a witness. To experiment this point, in our case study we modified the original problem by requiring its satisfiability with a number of clients equal to: (i) 3 for the three replica system, and (ii) 2 for the four replica one. The SMT solver provided as output *unsat* for both cases, and it took 69.42 s and 224.26 s, respectively. This shows that the difficulty in not finding a witness is model-dependent; in practical uses it is possible to consider setting a time-out that is treated as *unsat*, at the cost of obtaining false negatives.

6. DISCUSSION

Advantages. One major advantage of this approach is that a great amount of the computation effort is pushed to the design phase, where the symbolic expressions for the QN performance indices must be produced using a computer algebra system. The numerical experiments reported in this paper showed that the time taken by the satisfiability problem allows to perform adaptation steps in a few milliseconds even in a realistic three-tier model under the extreme-case condition where the system is able to self-adapt by varying any of its parameters. This scalability is mainly due to the fact that the complexity of the QN solution depends on the network topology and not on the size of the underlying continuous-time Markov chain. In more general approaches where such approximations are not feasible, carrying out run-time adaptation using formal verification techniques becomes more difficult and strategies to improve its effectiveness and responsiveness must be adopted [24].

Another advantage comes from the fact that the adap-

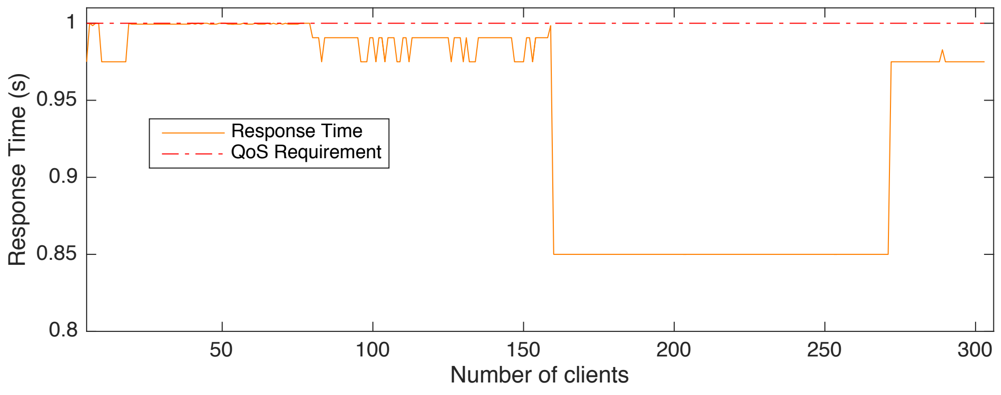

Figure 8: System response time.

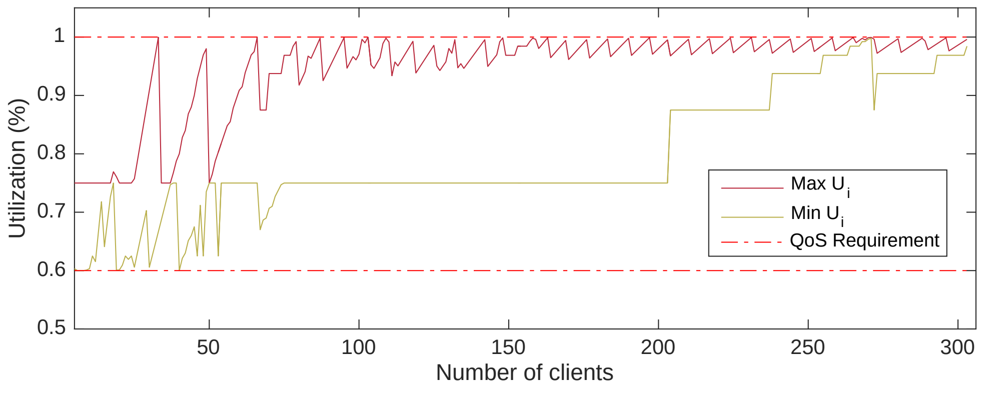

Figure 9: Minimum and maximum utilizations for each runtime configuration.

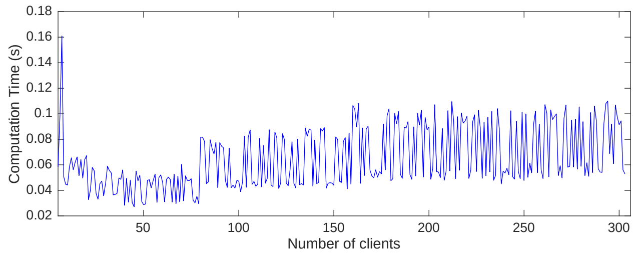

Figure 10: Computation time.

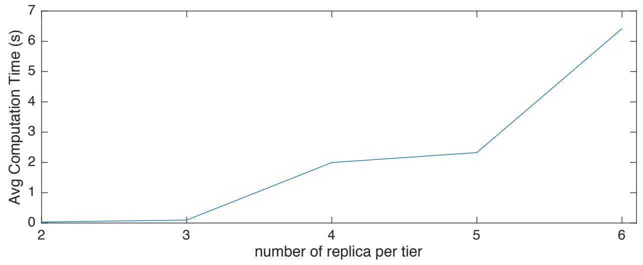

Figure 11: Scalability of SPA on the case study.

tation is symbolically performed: the solver may return an unsatisfiability result, meaning that there exists no configuration that can satisfy the current QoS requirements. This peculiar characteristic of our SMT approach can also be leveraged at design time (although this was not the focus of this paper). Indeed, it is not difficult to envisage a development process where the modeller uses this approach in order to find a rough sizing of the system, for instance using worst-case scenarios about the workload levels and service rates, which is then taken as the starting configuration that will be refined at runtime. Another possibility might be to include forecast knowledge describing load intensity variations over time, similarly to [43, 25], thus to pro-actively set system parameters [36] and anticipate QoS flaws by means of such additional source of information.

Limitations. The limitations of symbolic performance adaptation are currently in the assumptions behind the QN model. Despite being employed in many software performance models, QNs lack the necessary expressiveness to cover useful mechanisms such as simultaneous resource possession and layered services. The former is of increasing relevance especially in virtualized environments in order to model the case where more than one application/virtual machine shares the same hardware. The latter is typical in multi-tiered applications when one tier needs processing from tiers below in order to complete a request. Incidentally, our case study from [42] features a similar scenario, but it is successfully rendered as a "flat" QN by means of a careful interpretation of the routing probabilities.

A crucial feature for the derivation of a symbolic QN solution is the assumption of a closed network. However we argue that, in practice, this does not appear particularly limiting since it is possible to approximate exogenous arrivals by choosing a (single-server) station with a large enough initial population of jobs (to which jobs that exit the network are redirected).

Symbolic closed-form expressions are available for steady-state measures. Therefore, our performance adaptation can be effectively applied under the assumption that the duration of the transient dynamics is negligible with respect to the representative time scales of the system under consideration. The fluid solution of the QN, instantiated with the current parameters, may assist the modeler in deciding to which extent the assumption is acceptable in practice. Since it gives an approximation of steady state as the equilibrium point of a time-dependent trajectory, the fluid solution directly provides an estimate of the duration of the transient behavior. If, at runtime, the QoS requirements are not violated during that period of time, the modeler gains confidence that the new configuration converges to the desired steady state.

7. CONCLUSION

In this paper we have presented an approach for the adaptation of a software system using a queuing network (QN) as the underlying model employed to predict performance-related quality-of-service (QoS) metrics such as response times, throughput, and utilization. The main novelty lies in the use of a symbolic procedure as the core adaptation engine. Our methodology starts with the description of a system as well as all its possible configurations as a single "super-model" that encodes uncertainty and variability as symbolic variables. Building on previous results [29, 30], such a model enjoys a symbolic solution yielding an expression that gives the performance prediction of a concrete configuration by simply plugging in the numerical values related to that configuration. The key idea is to encode the model, the QoS constraints, and the necessary assumptions on the parameter spaces as first-order logical formulae over those variables, which thus represent adaptation "knobs" such as the number of servers at a queuing center or routing probabilities. In this way, one can leverage on efficient SMT solvers, in our case the well-known Z3 [16], in order to find an assignment of such variables that satisfies all the constraints. In terms of expressiveness, a challenging restriction is due to the single-class assumption which we aim to address in future work.

Acknowledgement

This was was partially supported by the EU project QUANTICOL, 600708, and by the DFG project DAPS2 (SPP1593) through a Mercator Fellowship.

8. REFERENCES

[1] M. Awad and D. A. Menascé. On the predictive properties of performance models derived through input-output relationships. In *European Workshop on Performance Engineering*, pages 89–103, 2014.

[2] S. Balsamo and A. Marin. Queueing networks. In *International School on Formal Methods for Performance Evaluation SFM*, pages 34–82, 2007.

[3] S. Balsamo and M. Marzolla. Performance evaluation of UML software architectures with multiclass queueing network models. In *International Workshop on Software and Performance*, pages 37–42, 2005.

[4] C. Barna, H. Ghanbari, M. Litoiu, and M. Shtern. Hogna: A Platform for Self-Adaptive Applications in Cloud Environments. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 83–87, 2015.

[5] C. Barrett, R. Sebastiani, S. A. Seshia, S. A. S. Cesare Tinelli Clark Barrett, Roberto Sebastiani, and C. Tinelli. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*, chapter Satisfiability Modulo Theories. 2009.

[6] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications.* Wiley, 2005.

[7] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. Van Rossum, S. Schulz, and R. Sebastiani. MathSAT: Tight Integration of SAT and Mathematical Decision Procedures. *Journal of Automated Reasoning*, 35:265–293, 2005.

[8] D. Cerotti, M. Gribaudo, P. Piazzolla, R. Pinciroli, and G. Serazzi. Multi-class queuing networks models for energy optimization. In *International Conference on Performance Evaluation Methodologies and Tools*, pages 98–105, 2014.

[9] A. Cimatti. Beyond boolean sat: Satisfiability modulo theories. In *International Workshop on Discrete Event Systems*, pages 68–73, 2008.

[10] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The mathSAT5 SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107. Springer, 2013.

[11] V. Cortellessa, A. D. Marco, and P. Inverardi. *Model-Based Software Performance Analysis.* Springer, 2011.

[12] V. Cortellessa and R. Mirandola. Deriving a queueing network based performance model from UML diagrams. In *International Workshop on Software and Performance*, pages 58–70, 2000.

[13] V. Cortellessa and R. Mirandola. Prima-uml: a performance validation incremental methodology on early uml diagrams. *Science of Computer Programming*, 44(1):101–129, 2002.

[14] A. Cumani. On the canonical representation of homogeneous markov processes modelling failure-time distributions. *Microelectronics Reliability*, 22(3):583–602, 1982.

[15] R. De Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, and others. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.

[16] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[17] L. De Moura and N. Bjørner. Satisfiability modulo theories: An appetizer. In *Formal Methods: Foundations and Applications*, pages 23–36. 2009.

[18] A. Di Marco and P. Inverardi. Compositional generation of software architecture performance qn models. In *Working IEEE/IFIP Conference on Software Architectures*, pages 37–46, 2004.

[19] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time parameter adaptation. In *International Conference on Software Engineering*, pages 111–121, 2009.

[20] L. Etxeberria, C. Trubiani, V. Cortellessa, and G. Sagardui. Performance-based selection of software and hardware features under parameter uncertainty. In *International Conference on Quality of Software Architectures*, pages 23–32, 2014.

[21] A. Filieri, C. Ghezzi, A. Leva, and M. Maggio. Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements. In *ASE*, pages 283–292, 2011.

[22] A. Filieri, L. Grunske, and A. Leva. Lightweight adaptive filtering for efficient learning and updating of probabilistic models. In *ICSE*, pages 200–211, 2015.

[23] A. Filieri, G. Tamburrelli, and C. Ghezzi. Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. *IEEE Trans. Software Eng.*, 42(1):75–99, 2016.

[24] S. Gerasimou, R. Calinescu, and A. Banks. Efficient runtime quantitative verification using caching, lookahead, and nearly-optimal reconfiguration. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 115–124, 2014.

[25] N. Herbst, S. Kounev, A. Weber, and H. Groenda. BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 46–56, 2015.

[26] M. U. Iftikhar and D. Weyns. Activforms: Active formal models for self-adaptation. In *Proceedings of the International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 125–134. ACM, 2014.

[27] P. Jamshidi, A. Ahmad, and C. Pahl. Autonomic resource provisioning for cloud-based software. In *SEAMS*, pages 95–104, 2014.

[28] E. Kang, E. Jackson, and W. Schulte. An Approach for Effective Design Space Exploration. In *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, number 6662 in Lecture Notes in Computer Science, pages 33–54. Springer Berlin Heidelberg, Mar. 2010.

[29] M. Kowal, I. Schaefer, and M. Tribastone. Family-based performance analysis of variant-rich software systems. In *FASE*, 2014.

[30] M. Kowal, M. Tschaikowski, M. Tribastone, and I. Schaefer. Scaling size and parameter spaces in variability-aware software performance models. In *International Conference on Automated Software Engineering*, 2015.

[31] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of Applied Probability*, 7:pp. 49–58, 1970.

[32] D. A. Menascé. Computing missing service demand parameters for performance models. In *International Computer Measurement Group Conference*, pages 241–248, 2008.

[33] P. Mohan, A. U. Shankar, and K. JayaSriDevi. Quality flaws: Issues and challenges in software development. *Computer Engineering and Intelligent Systems*, 3(12):40–48, 2012.

[34] J. C. Moreno, A. Lopes, D. Garlan, and B. R. Schmerl. Impact models for architecture-based self-adaptive systems. In *International Symposium on Formal Aspects of Component Software*, pages 89–107, 2014.

[35] A. Nikravesh, S. Ajila, and C.-H. Lung. Towards an Autonomic Auto-scaling Prediction System for Cloud Resource Provisioning. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 35–45, 2015.

[36] L. Sabatucci and M. Cossentino. From Means-End Analysis to Proactive Means-End Reasoning. In *SEAMS*, pages 2–12, 2015.

[37] C. U. Smith, C. M. Lladó, and R. Puigjaner. Performance model interchange format (PMIF 2): A comprehensive approach to queueing network model interoperability. *Perform. Eval.*, 67(7):548–568, 2010.

[38] W. J. Stewart. *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling.* Princeton University Press, 2009.

[39] A. N. Tantawi and D. Towsley. Optimal static load balancing in distributed computer systems. *J. ACM*, 32(2):445–465, Apr. 1985.

[40] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. A classification and survey of analysis strategies for software product lines. *ACM Comput. Surv.*, 47(1):6:1–6:45, June 2014.

[41] C. Trubiani, I. Meedeniya, V. Cortellessa, A. Aleti, and L. Grunske. Model-based performance analysis of software architectures under uncertainty. In *International Conference on Quality of Software Architectures*, pages 69–78, 2013.

[42] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An analytical model for multi-tier internet services and its applications. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 291–302. ACM, 2005.

[43] J. Von Kistowski, N. Herbst, D. Zoller, S. Kounev, and A. Hotho. Modeling and extracting load intensity profiles. In *SEAMS*, pages 109–119, 2015.

[44] F. Wang, G.-D. Huang, and F. Yu. TCTL inevitability analysis of dense-time systems: From theory to engineering. *IEEE Transactions on Software Engineering*, 32(7):510–526, 2006.

[45] C. M. Woodside, T. Zheng, and M. Litoiu. The use of optimal filters to track parameters of performance models. In *International Conference on the Quantitative Evaluation of Systems*, pages 74–84, 2005.

[46] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6(1):1–30, 1997.