Model-based performance self-adaptation

Emilio Incerto and Mirco Tribastone

IMT School for Advanced Studies Lucca, Italy
{emilio.incerto,mirco.tribastone}@imtlucca.it

ICPE'19 Tutorial
Mumbai, India

Motivation

IMT

- Performance is a fundamental aspect of computing systems:

  "[. . .] in many practical deployment scenarios, particularly mobile, ***performance is the new correctness****."*[1]

- Reasoning about performance is hard:
  - Large input spaces, measurement and testing becomes difficult.
  - Behavior dependent on interplay between software and hardware, can be machine dependent.
  - Usage profiles may be uncertain, performance analysis done at design time may not be sufficient.

[1] Mark Harman and Peter O'Hearn. "From Start-ups to Scale-ups: Opportunities and Open Problems for Static and Dynamic Program Analysis". In: *SCAM*. 2018.

Performance requirements at runtime

IMT

AWS Auto Scaling Overview Features Pricing Resources FAQs

AWS Auto Scaling

Application scaling to optimize performance and costs

Get started with AWS Auto Scaling

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, it's easy to setup application scaling for multiple resources across multiple services in minutes. The service provides a simple, powerful user interface that lets you build scaling plans for resources including Amazon EC2 instances and Spot Fleets, Amazon ECS tasks, Amazon DynamoDB tables and indexes, and Amazon Aurora Replicas. AWS Auto Scaling makes scaling simple with recommendations that allow you to optimize performance, costs, or balance between them. If you're already using Amazon EC2 Auto Scaling to dynamically scale your Amazon EC2 instances, you can now combine it with AWS Auto Scaling to scale additional resources for other AWS services. With AWS Auto Scaling, your applications always have the right resources at the right time.

It's easy to get started with AWS Auto Scaling using the AWS Management Console, Command Line Interface (CLI), or SDK. AWS Auto Scaling is available at no additional charge. You pay only for the AWS resources needed to run your applications and Amazon CloudWatch monitoring fees.

Introducing Predictive Scaling

Automatically scale your compute capacity in advance of traffic changes using ML technology.

Read the Blog>>

Amazon EC2 Auto Scaling has moved

You have several options for scaling with AWS.
Do you only need to manage Amazon EC2 instances?

Visit the Amazon EC2 Auto Scaling page »

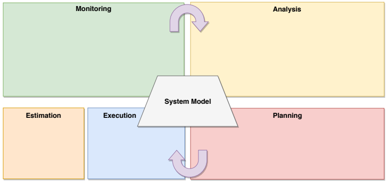
Amazon Web Services

Performance self-adaptation

Objective: Design systems that adapt themselves to changing environments while meeting desired performance-based service-level agreements (throughput, response time, utilization).

Three main activities:

1. monitor the system execution;
2. continuously update a model of the system;
3. trigger reconfigurations when required.



MAPE-K control loop

Performance self-adaptation

Objective: Design systems that adapt themselves to changing environments while meeting desired performance-based service-level agreements (throughput, response time, utilization)

Three main activities:

1. monitor the system execution;
2. continuously update a model of the system;
3. trigger reconfigurations when required.

Three main difficulties:

1. non-intrusive monitoring infrastructure;
2. ***efficient and accurate predictive models;***
3. effective and robust planner.

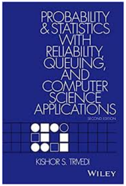
Table of Contents

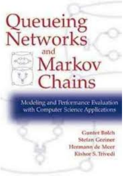
IMT

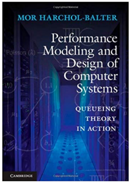
Table of Contents

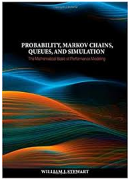
Efficient performance models

- We build on queuing networks, an established class of performance models for computing systems.
- Need to convince ICPE audience?
  - Varsha Apte's workshop was yesterday…
  - Kishor Trivedi speaks tomorrow…

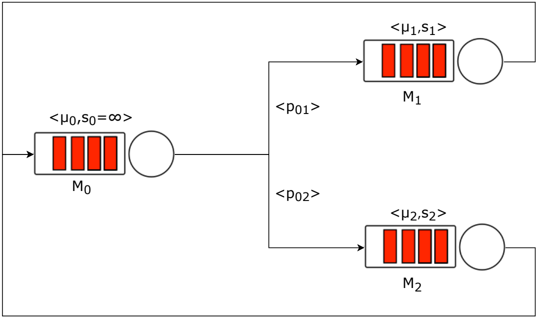








Queuing networks



A load balancing system

- Exponentially distributed service times—can be relaxed;[2]
- single class of users—multiple classes later;
- multiple servers.

[2] Matthias Kowal et al. "Scaling Size and Parameter Spaces in Variability-Aware Software Performance Models". In: *ASE*. 2015.

Queuing networks for self-adaptation?

- Queuing networks enjoy efficient analysis techniques (e.g., BCMP theorem), **but**:
  - In general, each parameterization requires a distinct analysis — large cost when exploring large parameter spaces.

    Our contribution:
    Efficient parametric analysis of queuing networks.

  - Analytical results assume time-homogeneous networks, i.e., parameters do not vary with time — adaptation requires changing the system, hence the parameters of its model.
  - Most analytical results available for the steady-state regime — adaptation might need to change parameters at all time points.

    Our contribution:
    Approximate queuing network analysis based on fluid models.

Parametric analysis of queuing networks

IMT

- Consider a queuing network with probability matrix P, vector of exogenous arrivals λ, and service rates μ.
- Consider concrete as well as symbolic parameters.
- Solve for stationary average performance indices using a computer algebra system.[3]

$$P = \begin{bmatrix} 0.0 & p^*_{1,2} & p^*_{1,3} & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.5 & 0.0 & 0.0 & 0.0 \end{bmatrix} \qquad \lambda = \begin{bmatrix} \lambda^*_1 \\ 0.0 \\ 0.0 \\ 0.1 \end{bmatrix} \qquad \mu = \begin{bmatrix} 1.5 \\ 2.0 \\ 3.0 \\ \mu^*_4 \end{bmatrix}$$

[3] Matthias Kowal, Ina Schaefer, and Mirco Tribastone. "Family-Based Performance Analysis of Variant-Rich Software Systems". In: *FASE*. 2014.

Parametric analysis of queuing networks

- Consider a queuing network with probability matrix P, vector of exogenous arrivals λ, and service rates μ.
- Consider concrete as well as symbolic parameters.
- Solve for stationary average performance indices using a computer algebra system.[4]

$$
\begin{aligned}
\mathrm{AQL} = &-\frac{20\lambda_1^* + 1}{80\lambda_1^* + 60p_{1,2}^* + 60p_{1,3}^* - 116} \\
&- \frac{p_{1,2}^*(20\lambda_1^* + 1)}{4(21p_{1,2}^* + 20p_{1,3}^* + 20\lambda_1^* p_{1,2}^* - 40)} \\
&- \frac{10\lambda_1^* p_{1,2}^* + 10\lambda_1^* p_{1,3}^* + 1}{20\mu_4^* p_{1,2}^* - 40\mu_4^* + 20\mu_4^* p_{1,3}^* + 40\lambda_1^* p_{1,2}^* + 40\lambda_1^* p_{1,3}^* + 4} \\
&- \frac{p_{1,3}^*(20\lambda_1^* + 1)}{4(30p_{1,2}^* + 31p_{1,3}^* + 20\lambda_1^* p_{1,2}^* - 60)}
\end{aligned}
$$

[4] Matthias Kowal, Ina Schaefer, and Mirco Tribastone. “Family-Based Performance Analysis of Variant-Rich Software Systems”. In: *FASE*. 2014.

Numerical vs symbolic analysis

Variables				*Runtimes (s)*			
Nodes	*p*	*m*	*g*	*SYM*	*NUM*	*NUM/SYM*	*PC*
4	1	0	0	0.011	0.049	4.47	0.545
4	0	1	0	0.004	0.043	10.78	0.185
4	0	0	1	0.009	0.045	4.92	0.190
4	1	1	1	0.011	0.046	4.13	0.214
4	2	2	2	0.014	0.049	3.60	0.319
4	4	4	4	0.020	0.049	2.43	0.310
24	1	0	0	0.011	0.066	5.96	1.141
24	0	1	0	0.004	0.063	14.60	1.124
24	0	0	1	0.011	0.067	6.29	1.152
24	1	1	1	0.013	0.068	5.04	1.244
24	2	2	2	0.016	0.068	4.31	1.100
24	0	6	0	0.007	0.065	9.94	1.004
24	4	4	4	0.099	0.070	0.70	1.904

Numerical vs symbolic analysis

Variables				*Runtimes (s)*			
Nodes	*p*	*m*	*g*	*SYM*	*NUM*	*NUM/SYM*	*PC*
142	1	0	0	0.016	6.540	397.34	5.425
142	0	1	0	0.005	8.107	1664.28	4.908
142	0	0	1	0.015	10.808	726.41	4.836
142	1	1	1	0.017	10.069	601.51	5.113
142	2	2	2	0.019	10.052	526.50	4.936
142	0	6	0	0.007	7.802	1191.79	5.137
142	4	4	4	0.026	10.985	429.83	4.942
302	1	0	0	0.019	19.186	1007.95	11.026
302	0	1	0	0.006	13.680	2292.46	11.192
302	0	0	1	0.018	13.399	728.73	11.050
302	1	1	1	0.021	19.520	909.12	11.591
302	2	2	2	0.024	19.459	820.30	11.214
302	0	6	0	0.006	13.896	2258.21	11.089
302	4	4	4	0.030	14.879	495.06	11.718

Queuing networks for self-adaptation?

- Queuing networks enjoy efficient analysis techniques (e.g., BCMP theorem), **but**:
  - In general, each parameterization requires a distinct analysis — large cost when exploring large parameter spaces.

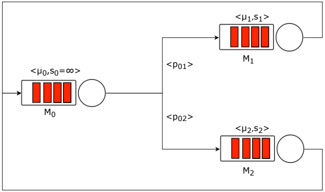
    Our contribution:
    Efficient parametric analysis of queuing networks.

  - Analytical results assume time-homogeneous networks, i.e., parameters do not vary with time — adaptation requires changing the system, hence the parameters of its model.
  - Most analytical results available for the steady-state regime — adaptation might need to change parameters at all time points.

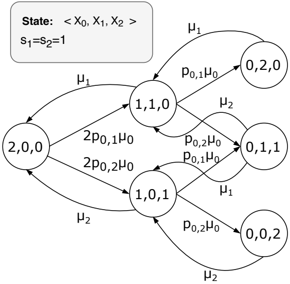
    Our contribution:
    Approximate queuing network analysis based on fluid models.

Transient solutions for queuing networks?

Queuing network



Underlying continuous-time Markov chain



Forward (master) equations

$$\dot{\pi}_{(2,0,0)} = -2\mu_0\pi_{(2,0,0)} + \mu_1\pi_{(1,1,0)} + \mu_2\pi_{(1,0,1)}$$

$$\vdots$$

$$\dot{\pi}_{(0,0,2)} = -\mu_2\pi_{(0,0,2)} + p_{0,2}\mu_0\pi_{(1,0,1)}$$

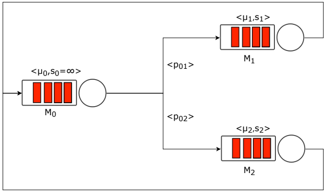
State explosion

Number of states is exponential!

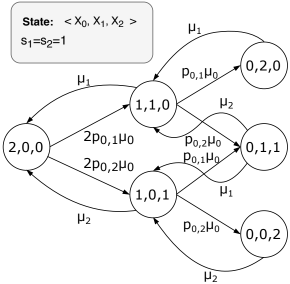
Transient solutions for queuing networks?

IMT

Queuing network



Underlying continuous-time Markov chain



Forward (master) equations

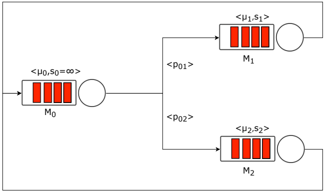
$$\dot{\pi}_{(2,0,0)} = -2\mu_0\pi_{(2,0,0)} + \mu_1\pi_{(1,1,0)} + \mu_2\pi_{(1,0,1)}$$

$$\vdots$$

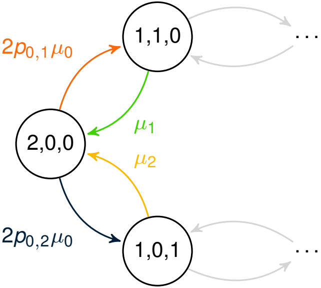
$$\dot{\pi}_{(0,0,2)} = -\mu_2\pi_{(0,0,2)} + p_{0,2}\mu_0\pi_{(1,0,1)}$$

Clients	*States*	*Transitions*
2	6	71
10	66	220
20	231	840
40	861	3280
80	3321	12960

Markov population processes



$$
\begin{aligned}
Q_0 &\to Q_1, && p_{0,1}\mu_0 Q_0 \\
Q_0 &\to Q_2, && p_{0,2}\mu_0 Q_0 \\
Q_1 &\to Q_0, && \mu_1 \min\{Q_1, s_1\} \\
Q_2 &\to Q_0, && \mu_2 \min\{Q_2, s_2\}
\end{aligned}
$$

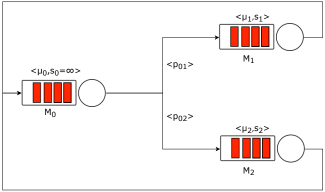


$$\Downarrow$$

$$
\begin{aligned}
l_1 &= (-1, +1, 0), && f_1(x) = p_{0,1}\mu_0 x_0 \\
l_2 &= (-1, 0, -1), && f_2(x) = p_{0,2}\mu_0 x_0 \\
l_3 &= (+1, -1, 0), && f_3(x) = \mu_1 \min\{x_1, s_1\} \\
l_4 &= (+1, 0, -1), && f_4(x) = \mu_2 \min\{x_2, s_2\}
\end{aligned}
$$

Jumps and rate functions

Markov population processes



$$
\begin{aligned}
Q_0 &\rightarrow Q_1, \qquad & p_{0,1}\mu_0 Q_0 \\
Q_0 &\rightarrow Q_2, \qquad & p_{0,2}\mu_0 Q_0 \\
Q_1 &\rightarrow Q_0, \qquad & \mu_1 \min\{Q_1, s_1\} \\
Q_2 &\rightarrow Q_0, \qquad & \mu_2 \min\{Q_2, s_2\}
\end{aligned}
$$

$$\Downarrow$$

Master Equation

$$\dot{\pi}_n(t) = -\sum_i f_i(n)\pi_n(t) + \\ + \sum_i f_i(n - l_i)\pi_{n-l_i}(t)$$

for all states $n = (n_0, n_1, n_2)$.

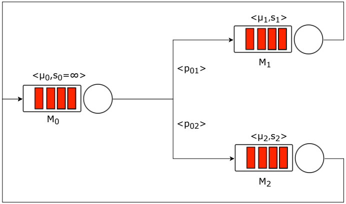
$$
\begin{aligned}
l_1 &= (-1, +1, 0), \quad & f_1(x) &= p_{0,1}\mu_0 x_0 \\
l_2 &= (-1, 0, -1), \quad & f_2(x) &= p_{0,2}\mu_0 x_0 \\
l_3 &= (+1, -1, 0), \quad & f_3(x) &= \mu_1 \min\{x_1, s_1\} \\
l_4 &= (+1, 0, -1), \quad & f_4(x) &= \mu_2 \min\{x_2, s_2\}
\end{aligned}
$$

Jumps and rate functions

Equations for the average

Master equation

$$\dot{\pi}_n(t) = -\sum_i f_i(n)\pi_n(t) + \sum_i f_i(n - l_i)\pi_{n-l_i}(t), \quad \text{for all states } n.$$



$$\begin{aligned} l_1 &= (-1, +1, 0), & f_1(x) &= p_{0,1}\mu_0 x_0 \\ l_2 &= (-1, 0, -1), & f_2(x) &= p_{0,2}\mu_0 x_0 \\ l_3 &= (+1, -1, 0), & f_3(x) &= \mu_1 \min\{x_1, s_1\} \\ l_4 &= (+1, 0, -1), & f_4(x) &= \mu_2 \min\{x_2, s_2\} \end{aligned}$$

- Call $X(t) = (X_0(t), X_1(t), X_2(t))$ the queue length process.
- The true equations for the average queue lengths are:

$$\mathbb{E}[\dot{X}](t) = \sum_i l_i \mathbb{E}\left[f_i(X(t))\right]$$

- Problem: equations are not closed!

Approximate equations for the average

IMT

True equations

$$\mathbb{E}[\dot{X}](t) = \sum_i l_i \mathbb{E}\left[f_i(X(t))\right]$$

$$\begin{aligned}
l_1 &= (-1, +1, 0), \quad & f_1(x) &= p_{0,1}\mu_0 x_0 \\
l_2 &= (-1, 0, -1), \quad & f_2(x) &= p_{0,2}\mu_0 x_0 \\
l_3 &= (+1, -1, 0), \quad & f_3(x) &= \mu_1 \min\{x_1, s_1\} \\
l_4 &= (+1, 0, -1), \quad & f_4(x) &= \mu_2 \min\{x_2, s_2\}
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[\dot{X}_0](t) &= -p_{0,1}\mu_0\mathbb{E}[X_0](t) - p_{0,2}\mu_0\mathbb{E}[X_0](t) \\
&\quad + \mathbb{E}\left[\mu_1 \min\{X_1(t), s_1\}\right] + \mathbb{E}\left[\mu_2 \min\{X_2(t), s_2\}\right] \\
\mathbb{E}[\dot{X}_1](t) &= +p_{0,1}\mu_0\mathbb{E}[X_0](t) - \mathbb{E}\left[\mu_1 \min\{X_1(t), s_1\}\right] \\
\mathbb{E}[\dot{X}_2](t) &= +p_{0,2}\mu_0\mathbb{E}[X_0](t) - \mathbb{E}\left[\mu_2 \min\{X_2(t), s_2\}\right]
\end{aligned}$$

How to close the equations? $\mathbb{E}[\min\{X, Y\}] \approx \min\{\mathbb{E}[X], \mathbb{E}[Y]\}$

Approximate equations for the average

IMT

True equations

$$\mathbb{E}[\dot{X}](t) = \sum_i l_i \mathbb{E}\left[f_i(X(t))\right]$$

$$\begin{aligned}
l_1 &= (-1, +1, 0), & f_1(x) &= p_{0,1}\mu_0 x_0 \\
l_2 &= (-1, 0, -1), & f_2(x) &= p_{0,2}\mu_0 x_0 \\
l_3 &= (+1, -1, 0), & f_3(x) &= \mu_1 \min\{x_1, s_1\} \\
l_4 &= (+1, 0, -1), & f_4(x) &= \mu_2 \min\{x_2, s_2\}
\end{aligned}$$

Fluid Approximation

$$\begin{aligned}
\dot{X}_0(t) &= -p_{0,1}\mu_0 X_0(t) - p_{0,2}\mu_0 X_0(t) \\
&\qquad + \mu_1 \min\{X_1(t), s_1\} + \mu_2 \min\{X_2(t), s_2\} \\
\dot{X}_1(t) &= +p_{0,1}\mu_0 X_0(t) - \mu_1 \min\{X_1(t), s_1\} \\
\dot{X}_2(t) &= +p_{0,2}\mu_0 X_0(t) - \mu_2 \min\{X_2(t), s_2\}
\end{aligned}$$

How to close the equations? $\mathbb{E}[\min\{X, Y\}] \approx \min\{\mathbb{E}[X], \mathbb{E}[Y]\}$

Another interpretation: fluid limits

IMT

Markov population process

$$
\begin{aligned}
l_1 &= (-1,+1,0), & f_1(x) &= p_{0,1}\mu_0 x_0 \\
l_2 &= (-1,0,-1), & f_2(x) &= p_{0,2}\mu_0 x_0 \\
l_3 &= (+1,-1,0), & f_3(x) &= \mu_1 \min\{x_1, K s_1\} \\
l_4 &= (+1,0,-1), & f_4(x) &= \mu_2 \min\{x_2, K s_2\}
\end{aligned}
$$

CTMC family

$$
\begin{aligned}
X^1(0) &= (Q_0, Q_1, Q_2) \\
X^2(0) &= 2(Q_0, Q_1, Q_2) \\
&\dots \\
X^K(0) &= K(Q_0, Q_1, Q_2)
\end{aligned}
$$

- Fix some initial condition (Q_0, Q_1, Q_2)
- Consider Markov chains with increasing clients as well as servers
- In the limit as K goes to infinity one sample path of $X^K(t)/K$ will converge to the fluid equations[5]

[5] T. G. Kurtz. "Solutions of ordinary differential equations as limits of pure Markov processes". In: *J. Appl. Prob.* (1970).

Fluid approximation of queuing networks

- Let $1, \ldots, N$ be the queuing stations;
- let $P = (p_{i,j})_{1 \leq i,j \leq N}$ be the routing probability matrix;
- let $s_1, \ldots, s_N$ be the server multiplicities;
- let $\mu_1, \ldots, \mu_N$ be the service rates.

The fluid approximation is given by[6]

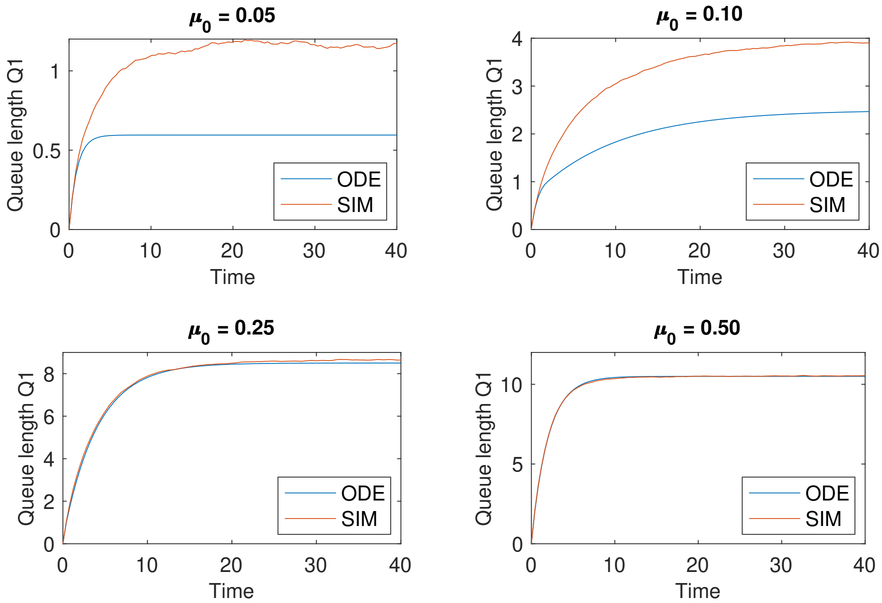
$$\dot{X}_i(t) = -\mu_i \min\{X_i(t), s_i\} + \sum_j p_{j,i}\mu_j \min\{X_j(t), s_j\}, \; i = 1, \ldots, N.$$

It can be also defined for time-varying parameters:

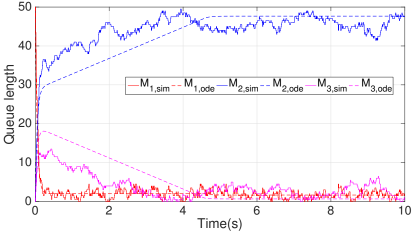
$$\dot{X}_i(t) = -\mu_i(t) \min\{X_i(t), s_i(t)\} + \sum_j p_{j,i}(t)\mu_j(t) \min\{X_j(t), s_j(t)\}$$

[6] Mirco Tribastone. "A Fluid Model for Layered Queueing Networks". In: *IEEE Trans. Software Eng.* (2013).

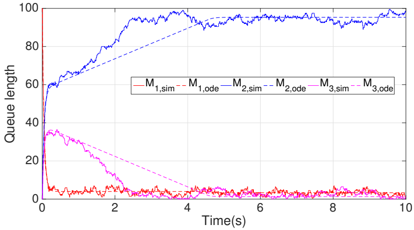
Mean approximation accuracy



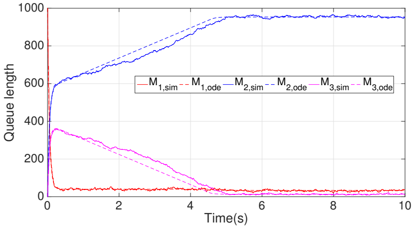
Fluid limit accuracy



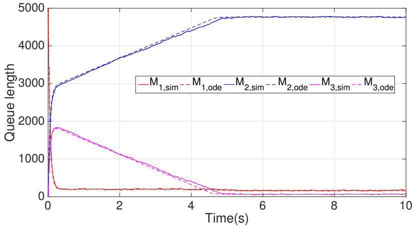
(a) K=1



(b) K=2



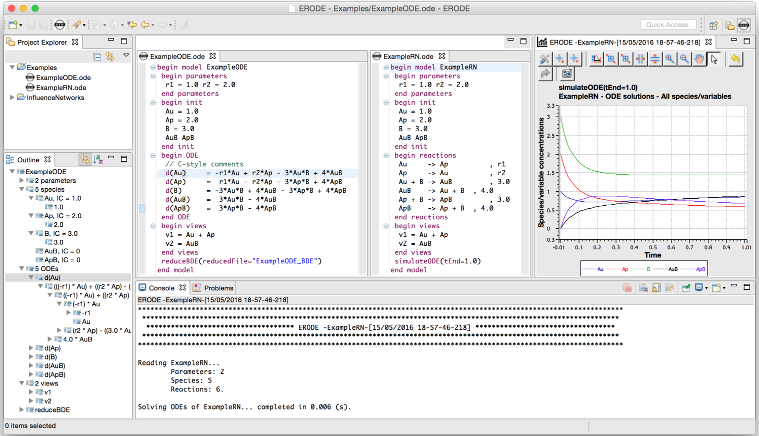
(c) K=20



(d) K=100

Comparison between a sample path of the CTMC and the fluid model

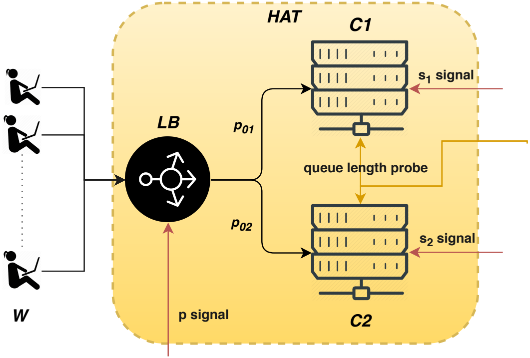
Interlude: ERODE session



https://sysma.imtlucca.it/tools/erode/

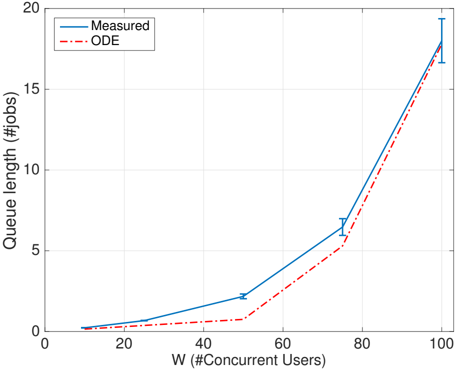
L. Cardelli, M. Tschaikowski, M. Tribastone, and A. Vandin. “ERODE: A tool for the evaluation and reduction of ordinary differential equations,” *TACAS’17*.

Running example: HAT architecture

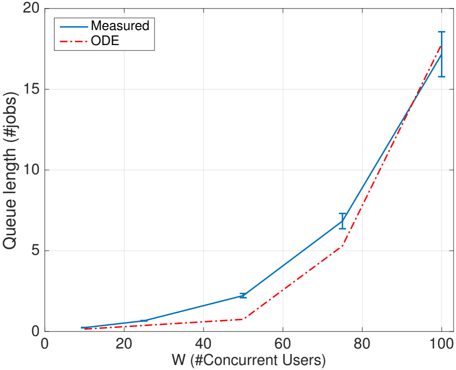


- In house developed web application
- It resembles CPU intensive behaviour
- Dispatcher (i.e., LB) and Computation Node (i.e., C1, C2)

Model validation



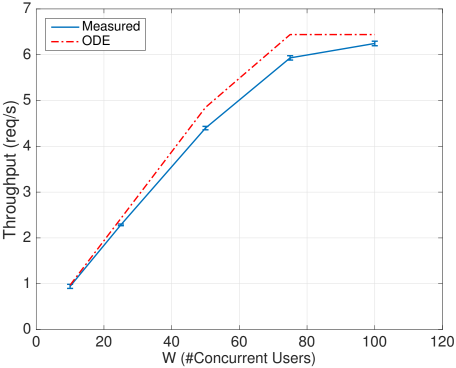
(a) N_1 queue length



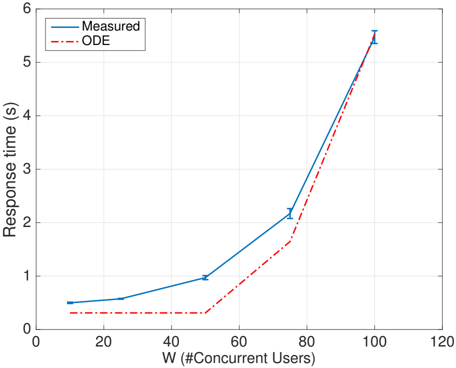
(b) N_2 queue length

Model validation

IMT



(a) System throughput



(b) System response time

Performance self-adaptation

Objective: Design systems that adapt themselves to changing environments while meeting desired performance-based service-level agreements (throughput, response time, utilization)

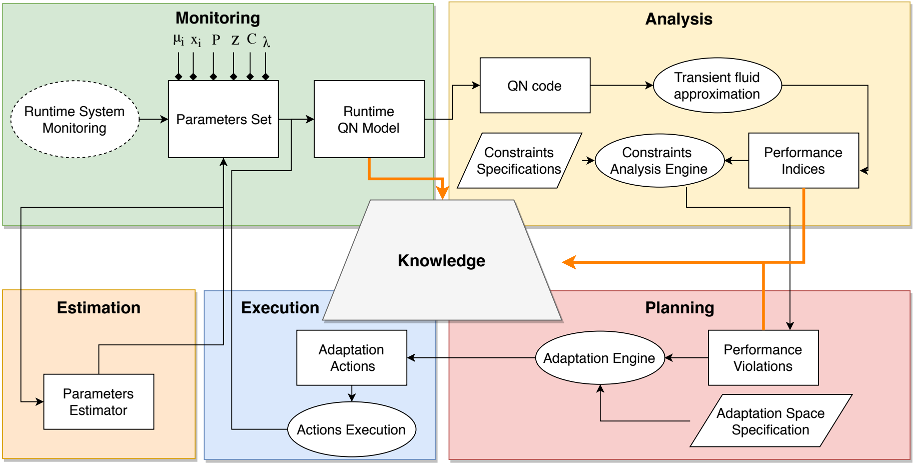
Three main activities:

1. monitor the system execution;
2. continuously update a model of the system;
3. trigger reconfigurations when required.

Three main difficulties:

1. non-intrusive monitoring infrastructure;
2. efficient and accurate predictive models;
3. ***effective and robust planner.***

A Performance-driven MAPE-K control loop



An MAPE-K framework based on fluid QN

Table of Contents

Symbolic performance adaptation

- **Idea:** Encode performance-driven adaptation as a satisfiability modulo theories (SMT) problem.[7]
- **Goal:** Query an SMT solver for a feasible assignment of the system parameters needed to satisfy QoS requirements or getting a formal proof of its non-existence.
- We rely on a combination of:
  - **queuing networks**: quantitative model to represent QoS attributes of the system;
  - **symbolic analysis**: represent all possible system configuration as a set of nonlinear real constraints;
  - **SMT**: devise feasible system configurations.

[7] Emilio Incerto, Mirco Tribastone, and Catia Trubiani. "Symbolic Performance Adaptation". In: *SEAMS*. 2016.

Preliminaries: Satisfiability Modulo Theory

- Satisfiability Modulo Theory[8] checks the satisfiability of logical formulas over one ore more theory, i.e., if there exists an assignment or real number satisfying

$$3x + 2y - z \geq 4 \wedge x \geq 0 \wedge y \geq 0$$

- We interpret the symbolic expressions as a set of constraints that the parametric values have to satisfy
- We augment them with further constraints representing the QoS requirements $\mathcal{Q}$

Exploiting the Non-Linear Real Arithmetic theory $\mathcal{NRA}$ we encode this problem as an SMT problem suitable to devise the set of feasible parameters satisfying $\mathcal{Q}$

[8] Clark Barrett et al. "Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications". In: ed. by A. Biere et al. IOS Press, 2009. Chap. Satisfiability Modulo Theories.

The SMT adaptation problem

- $\mathcal{M}$ be the symbolic solution of the QN
- $\mathcal{Q}$ be a set of QoS requirements
- $\mathcal{D}$ be a set of domain assumptions
- $\mathcal{R}$ be a set of resource constraints

The QoS-based adaptation is turned into the satisfiability problem:

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

$$\mathcal{Q} := \{\mathcal{T}_0 \geq C/2\}$$

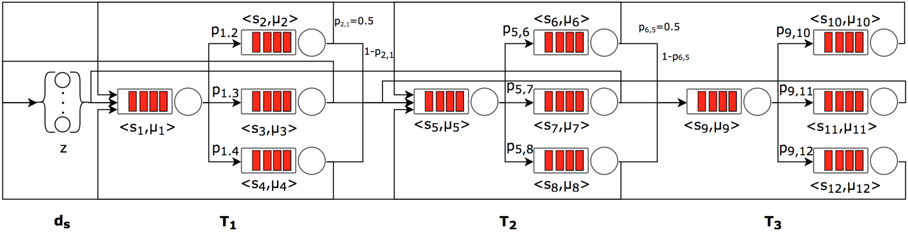
$$\mathcal{D} := \{C = 350\}$$

$$\mathcal{R} := \Big\{\forall i, j \in S. \sum_{k \in S} p_{i,k} = 1.0 \wedge$$

$$0 \leq p_{i,j} \leq 1.0 \wedge 1 \leq s_i \leq 40 \wedge 0.02 \leq \frac{1}{\mu_i} \leq 10\Big\}$$

Case study

IMT



QN model for a three-tier system[9]

Control objective

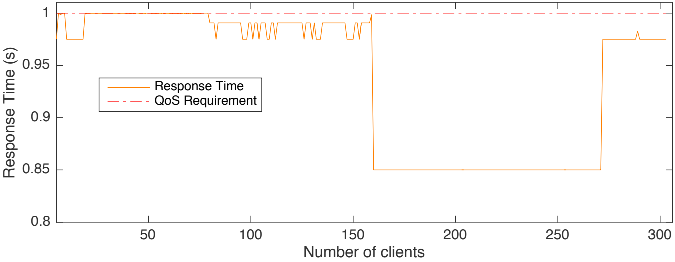
$$\mathcal{R}_s = C/\mathcal{T}_1 \leq 1 \qquad C \in [1, 304], \qquad Z = 0.$$

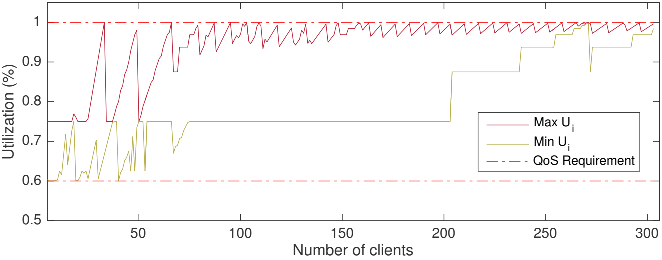
[9] Bhuvan Urgaonkar et al. “An analytical model for multi-tier internet services and its applications”. In: *SIGMETRICS*. 2005.

Case study: SMT problem formulation

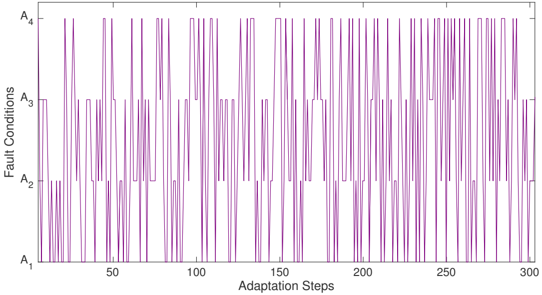
$$
\begin{aligned}
\mathcal{Q} &:= (\mathcal{T}_1 \geq C) \wedge \forall i \in S. \big((\mathcal{U}_i \leq 1.0 \wedge \mathcal{U}_i \geq 0.6) \vee \mathcal{U}_i = 0.0 \big) \\
\mathcal{D} \wedge \mathcal{R} &:= (Z = 0) \wedge \mathcal{R}_1 \wedge \mathcal{R}_2 \\
\mathcal{R}_1 &:= \forall i, j \in S. \big(0.0 \leq p_{i,j} \leq 1.0 \big) \wedge \forall i \in S. \sum_{j \in S} p_{i,j} = 1.0 \\
\mathcal{R}_2 &:= \forall i \in S. \Big((1.0 \leq s_i \leq 40.0) \wedge \big(0.06 \leq \frac{1}{\mu_i} \leq 10.0\big) \Big)
\end{aligned}
$$

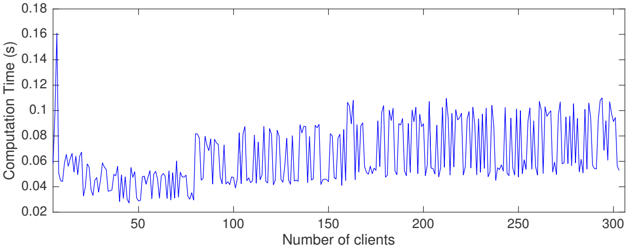
Case study: numerical results





Case study: numerical results





Discussion

- Advantages:
  - Computation pushed at design-time phase for the symbolic solution
  - Adaptation step is performed in a few ms even for realistic fully parametric three-tier models
  - The solver may return an unsatisfiability result, meaning that there is no configurations satisfying the requirements
- Limitations:
  - Single class QNs model
  - Closed form QNs model for the symbolic solution
  - Steady state analysis
  - Scalability of the SMT solver

Table of Contents

Self-adaptation via model predictive control

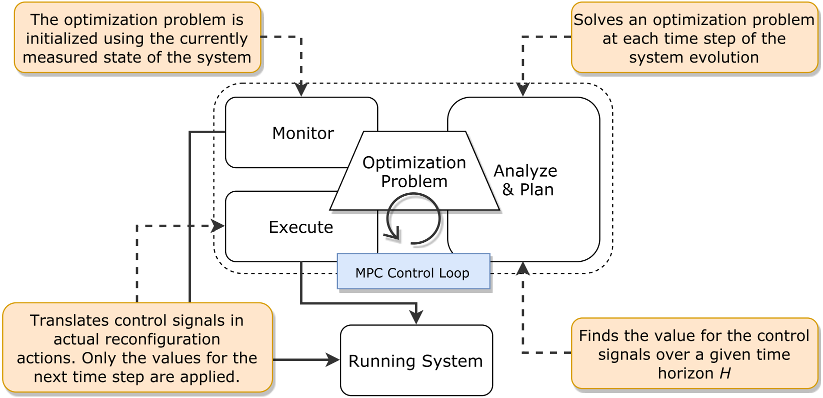
- **Idea**: Encode performance-driven self-adaptation as a model predictive control (MPC) problem[10]

- **Goals**: Get optimal assignments of the system parameters needed for steering an application toward a desired operating point (e.g., throughput)
  - fully automated
  - multiple adaptation knobs
  - considers actual run-time conditions
  - involves the solution of mixed integer nonlinear programs (MINLPs)

As a main technical result we formally translate the naive MINLP MPC formulation in a mixed integer quadratic programming (MIQP) one

[10] Emilio Incerto, Mirco Tribastone, and Catia Trubiani. "Software Performance Self-Adaptation through Efficient Model Predictive Control". In: *ASE'17*.

MPC philosophy



The main idea is to encode the discrete time version the ODE model as constraints of the optimization problem

Naive MPC formulation

$$\min_{\mu,s,p,x} \sum_{k=0}^{H-1} \sum_{i=1}^{M} w_{i,k}(m_i(k) - \hat{r}_i(k))^2 + w_{s_{i,k}} \Delta s_{i,k}^2$$

s.t.

$$x_i(k+1) = (-\mu_i(k) \min\{x_i(k), s_i(k)\} +$$
$$+ \sum_{j \in S} p_{j,i}(k) \mu_j(k) \min\{x_j(k), s_j(k)\}) \Delta t + x_i(k)$$

$$s_i(k) \in \{\underline{s}_i, \underline{s}_i + 1, \ldots, \overline{s}_i\}$$

$$0 \leq p_{i,j}(k) \leq 1, \sum_{j \in S} p_{i,j}(k) = 1$$

for $1 \leq i \leq M, 0 \leq k \leq H-1$,

$\hat{r}_i(k)$ set point

MIP formulation

IMT

- We reformulate the system relying on "virtual" adaptation knobs which are related to the original ones
- $\min\{x_i(k), s_i(k)\} = \alpha_i(k)$ can be encoded through standard MIP techniques[11]
- $\mu_i(k)\alpha_i(k)$ can be encoded by adding slack variables $\gamma_i(k)$ and proper bound constraints

$$\mu_i(k)\alpha_i(k) = \hat{\mu}_i\alpha_i(k) + \gamma_i(k) \leftrightarrow \mu_i(k) = \hat{\mu}_i + \frac{\gamma_i(k)}{\alpha_i(k)}$$

$$\begin{cases} \mu_i(k) \leq \bar{\mu}_i \leftrightarrow \gamma_i(k) \leq (\bar{\mu}_i - \hat{\mu}_i)\alpha_i(k) \\ \mu_i(k) \geq \underline{\mu}_i \leftrightarrow \gamma_i(k) \geq (\underline{\mu}_i - \hat{\mu}_i)\alpha_i(k) \end{cases}$$

- The new linear time system is

$$\dot{x}_i(t) = \gamma_i(t) + \sum_{j \in S}(-\gamma_j(t) + \zeta_{j,i}(t)), \quad i, j \in S$$

[11] `www.gurobi.com/documentation/8.1/refman/constraints.html`

MIP formulation

Theorem

Denoting by $\mathcal{S} = \{\mu_i^*(k), p_{i,j}^*(k), s_i^*(k), x_i^*(k)\}$ *an optimal solution of the non-linear adaptation problem, there exists an MPC problem based on the MIP formulation with linear dynamics such that its optimal solution* $\mathcal{S}' = \{\gamma_i'(k), x_i'(k), \zeta_{i,j}'(k), s_i'(k)\}$ *satisfies:*

$$\mu_i^*(k) = \begin{cases} -\frac{\gamma_i'(k)}{x_i'(k)\Delta t} & \text{if } x_i'(k) \leq s_i'(k) \\ -\frac{\gamma_i'(k)}{s_i'(k)\Delta t} & \text{if } x_i'(k) > s_i'(k) \end{cases}$$

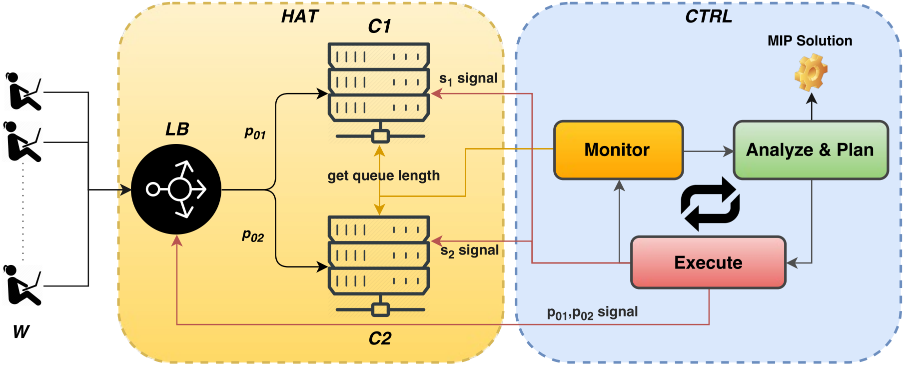
$$p_{i,j}^*(k) = \frac{\gamma_i'(k) - \zeta_{i,j}'(k)}{\gamma_i'(k)}, \quad s_i^*(k) = s_i'(k),$$

for all $k = 0, \ldots, H-1$.

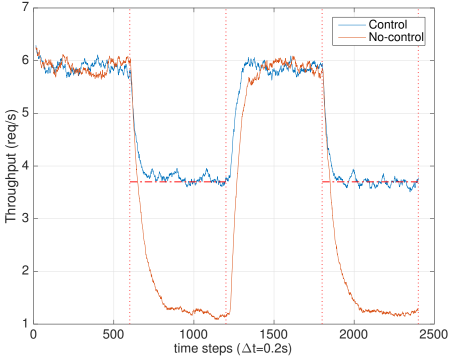
Evaluation

- We evaluate the effectiveness and the scalability of the MPC approach on a real system
  - an in-house developed web application
- By studying two non trivial adaptation scenarios
  - hardware degradation
  - workload fluctuation
- Scalability comparison with probabilistic model checking

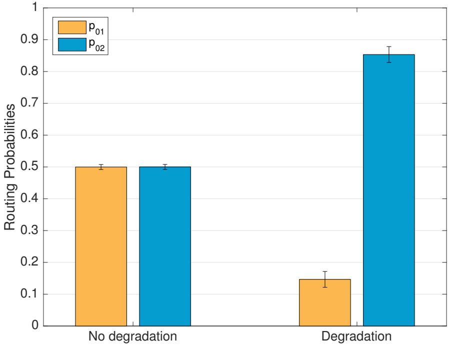
HAT architecture



Numerical evaluation: hardware degradation

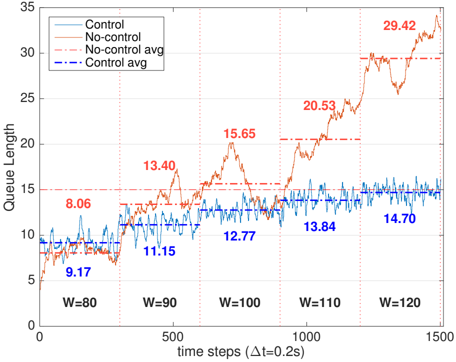


(a) System throughput

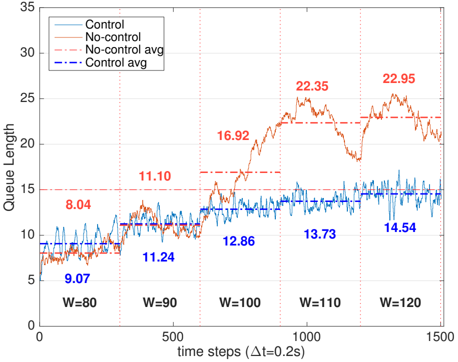


(b) Optimal control signals

Numerical evaluation: workload fluctuation

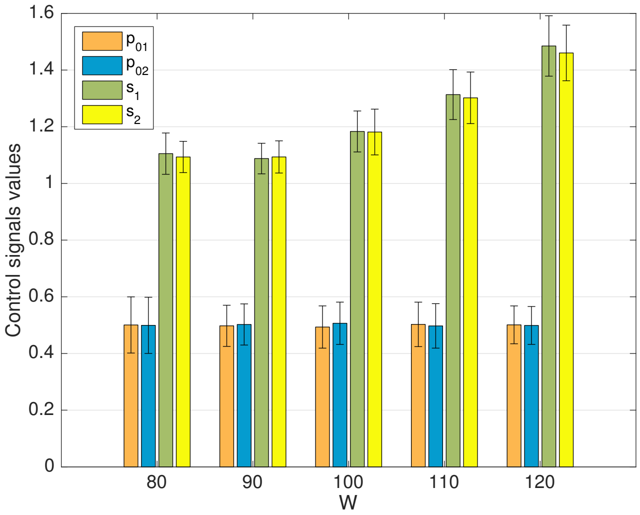


(a) C_1 queue length



(b) C_2 queue length

Numerical evaluation: workload fluctuation



Statistics of optimal control signals

MPC scalability evaluation

Comparison against MDP (TO: timeout after 120 s)[12]

	MIP	*Markov Decision Processes*		
W	Runtime(s)	Runtime(s)	# States	# Transitions
80	0.0037	71	3 018 789	334 732 743
90	0.0036	87	3 805 074	421 958 628
100	0.0040	TO	4 682 259	519 272 613
110	0.0038	TO	5 650 344	626 674 698
120	0.0041	TO	6 709 329	744 164 883

[12] Marta Kwiatkowska, Gethin Norman, and David Parker. “PRISM 4.0: Verification of Probabilistic Real-time Systems”. In: *CAV*. 2011.

Conclusion

- Contribution: a model predictive control based self-adaptive approach to continuously meet performance requirements
- Advantages:
  - fully automated
  - efficient
  - it works during the transient regime
  - the proposed linearization technique can be straightforwardly applied to more expressive models (e.g., Petri Nets)
- Limitations:
  - single-class model
  - QN parametrization
  - throughput and queue length QoS requirements only

Adaptation of co-located applications

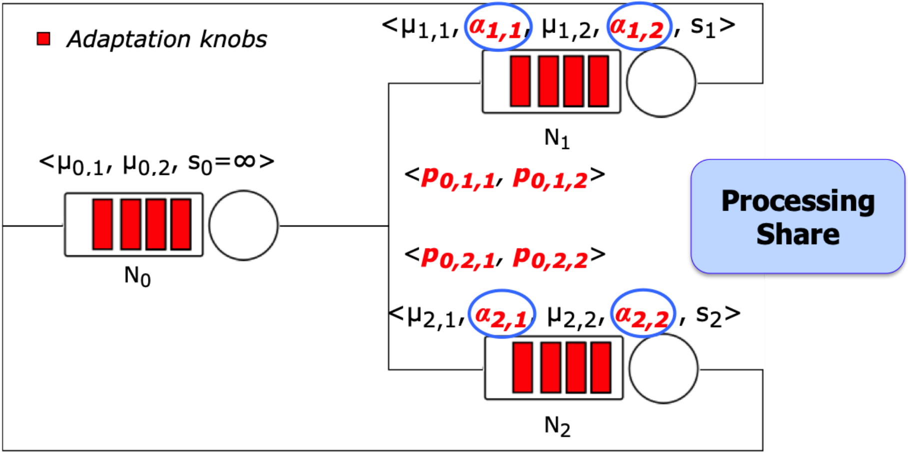
- Meeting performance targets of co-located applications (e.g., virtualized cloud environments) is challenging
- Scaling techniques:
  - *Vertical scaling*: assigns resource shares on each individual machine
  - *Horizontal scaling*: chooses the number of virtual machines employed
- State-of-the-art approaches apply vertical and horizontal scaling in an isolated fashion (i.e., symmetric load balancing)
- Ineffective when machines have different hardware characteristics (e.g., software aging or hardware degradation)

Adaptation of co-located applications

- A model based approach:
  - Multi-class QN as the quantitative model
  - Analysed by means of QNs fluid approximation
  - Combined horizontal and vertical scaling formulated as an MPC quadratic programming problem
- Main technical results:[13]
  - A multi-class model that enables an accurate representation of the **capped allocation paradigm**
  - The specification of latency-based requirements
  - Extension of [ASE'17]

[13] Emilio Incerto, Mirco Tribastone, and Catia Trubiani. "Combined Vertical and Horizontal Autoscaling Through Model Predictive Control". In: *EURO-PAR*. 2018.

The multi-class QN model



Multi class QN model of load balancer with two co-located applications

The multi-class fluid QN model

- For each station i and each class c we define the ODE:

$$\dot{x}_{i,c}(t) = -\mu_{i,c} \min\{x_{i,c}(t), \alpha_{i,c}(t) s_i\} + \sum_{j \in S} p_{j,i,c}(t) \mu_{j,c} \min\{x_{j,c}(t), \alpha_{j,c}(t) s_j\}$$

with $\alpha_{i,c}(t) \geq 0, \ \sum_{c \in C} \alpha_{i,c}(t) \leq 1$.

- We define the following metrics:
  - Class-c throughput at station i

$$T_{i,c}(t) = \mu_{i,c} \min\{x_{i,c}(t), \alpha_{i,c}(t) s_i\}$$

  - Instantaneous response time

$$R_{i,c}(t) = \frac{x_{i,c}(t)}{T_{i,c}(t)}$$

MPC QP formulation

- In [ASE'17] we employed MPC for performance runtime adaptation for single class queuing networks
- We showed how it could be formulated as a series of mixed-integer quadratic program (MIQP)
- Here we extend this formulation for controlling the multi-class QN under the cap allocation sharing
  - A positive side effect: quadratic programming (QP) formulation

MPC QP formulation

- The QP formulation is possible due to the following:

$$\min\{x_{i,c}(t), \alpha_{i,c}(t)s_i\} = \gamma_{i,c}(t)$$

$$\Updownarrow$$

$$\gamma_{i,c}(t) \geq 0, \qquad \gamma_{i,c}(t) \leq s_i$$

$$\gamma_{i,c}(t) \leq x_{i,c}(t) \qquad \sum_{c \in C} \gamma_{i,c}(t) \leq s_i$$

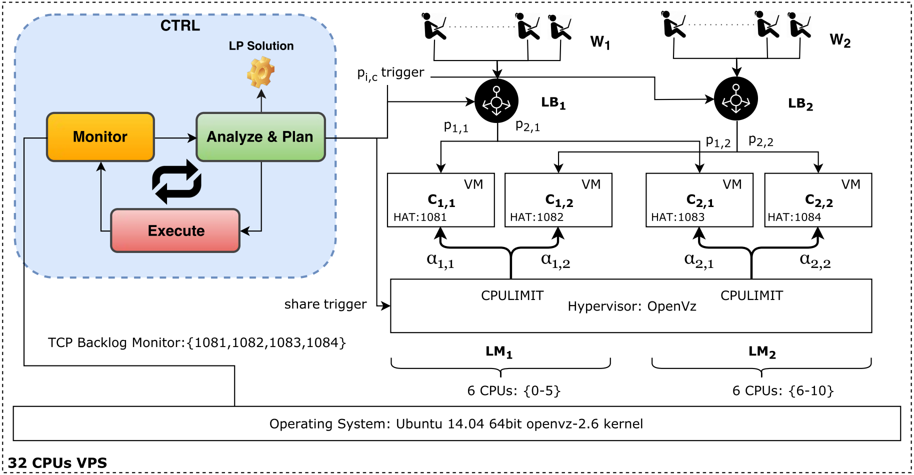
with $i \in S, c \in C$

- then if $\gamma^*_{i,c}(t) \neq x^*_{i,c}(t)$ the real actuator can be computed as $\alpha^*_{i,c}(t) = \frac{\gamma^*_{i,c}(t)}{s_i}$

Numerical evaluation: system description

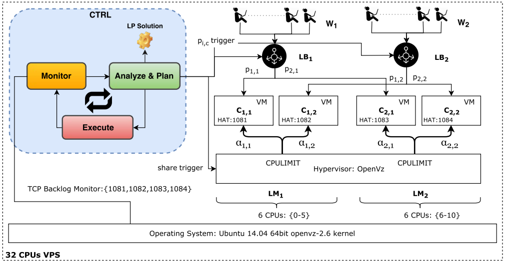
- We show that the combined vertical and horizontal adaptation can efficiently meet performance targets when either of the two techniques alone cannot
- We used the OpenVz hypervisor while horizontal scaling has been enabled through a NodeJS-based load balancer
- For emulating a multi class scenario, we ran two instances of the same load balanced HAT deployment consisting of two OpenVz virtual machines each

Numerical evaluation: system description

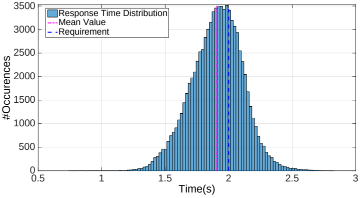


Numerical evaluation: hardware degradation

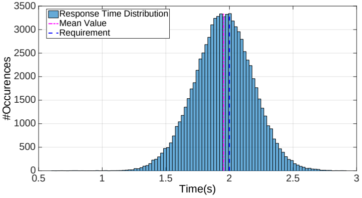
From a symmetric set-up, inject a degradation event such that service rate at node LM1 becomes 3 times smaller



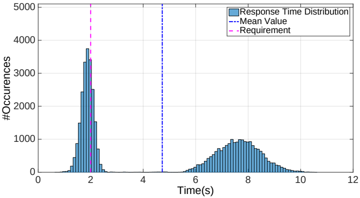
- Objective: set points $R_1 = 2\ s$ and $T_2 = 50\ r/s$
- Workload of 200 users and think times with average 1 s
- Control approaches evaluated in two separate 20-minute-long sessions



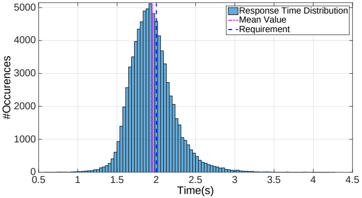
(a) Vertical scaling only



(b) Vertical & horizontal scaling



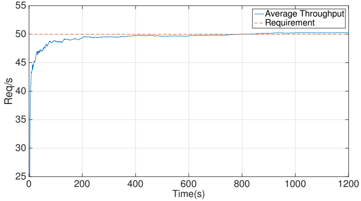
(c) Vertical scaling only



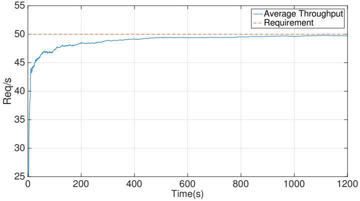
(d) Vertical & horizontal scaling

Response time distribution without (a,b) and with (c,d) degradation

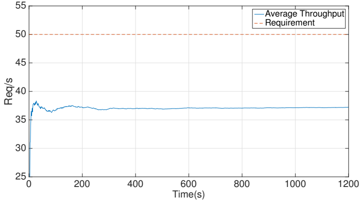
Numerical evaluation: hardware degradation



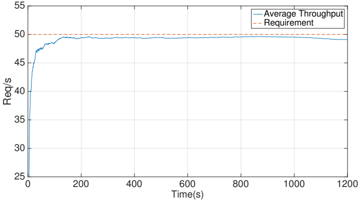
(a) Vertical scaling only



(b) Vertical & horizontal scaling



(c) Vertical scaling only



(d) Vertical & horizontal scaling

Class-2 average throughput with (a,b) and without (c,d) degradation

Discussion

- **Contribution**: an efficient approach for performance adaptation of distributed co-located applications.

- The main novelties lay:
  - The combined usage of vertical and horizontal scaling techniques
  - A multi-class fluid model for co-located applications under a capped resources allocation scheduler.

- Future works:
  - modeling response time distribution instead of its average only
  - include resource contention policies for network, memory, I/O
  - consider more expressive resource schedulers and system performance interactions

Table of Contents

Moving horizon estimation of service demands

IMT

- Well calibrated model parameters are necessary for computing accurate predictions
- When dealing with queuing networks service demands are fundamental
- The estimation need to be performed:
  - continuously
  - non intrusively

MHE: We formulate the estimation problem as a quadratic program solved according to the moving horizon paradigm.[14]

[14] Emilio Incerto, Annalisa Napolitano, and Mirco Tribastone. "Moving Horizon Estimation of Service Demands in Queuing Networks". In: *MASCOTS*. 2018.

Estimator formulation

$$
\begin{aligned}
&\underset{\bar{x},T}{\text{minimize}} \sum_{k=1}^{H} \sum_{i=1}^{M} (\overline{x}_i(k) - \tilde{x}_i(k))^2, \\
&\text{subject to:} \\
&\qquad \overline{x}_i(k+1) = \overline{x}_i(k) - T_i(k) + \sum_{j=1}^{M} p_{j,i} T_j(k) \\
&\qquad \overline{x}_i(0) = \tilde{x}_i(0), \quad 1 \leq i \leq M, 0 \leq k \leq H-1.
\end{aligned}
$$

$$
\mu_i^* := \frac{\sum_{k=0}^{H-1} T_i^*(k)}{\sum_{k=0}^{H-1} \min\{\overline{x}_i^*(k), s_i\}}, \qquad 1 \leq i \leq M.
$$

Numerical evaluation

Accuracy comparison between the queue length maximum likelihood estimation (QMLE)[16] and our approach (MHE).

	$x(0) = (3,0,0)$ $H = 2347, U_2 \approx 0.10$		$x(0) = (9,0,0)$ $H = 688, U_2 \approx 0.30$		$x(0) = (12,0,0)$ $H = 521, U_2 \approx 0.40$		$x(0) = (19,0,0)$ $H = 353, U_2 \approx 0.60$		$x(0) = (26,0,0)$ $H = 262, U_2 \approx 0.80$	
K	QMLE	MHE	QMLE	MHE	QMLE	MHE	QMLE	MHE	QMLE	MHE
1	0.52	9.25 ± 1.03	1.37	9.63 ± 1.06	2.07	7.90 ± 1.01	3.40	6.58 ± 0.81	5.15	4.89 ± 0.69
2	448.30	4.13 ± 0.62	126.54	3.93 ± 0.58	67.18	4.20 ± 0.63	5.46	3.90 ± 0.56	2.33	3.59 ± 0.54
5	184.02	2.26 ± 0.33	60.41	3.02 ± 0.43	42.09	2.76 ± 0.38	8.78	2.07 ± 0.33	1.65	2.06 ± 0.34
10	92.29	1.65 ± 0.27	30.53	1.99 ± 0.31	23.18	1.82 ± 0.31	9.50	2.09 ± 0.30	3.89	1.50 ± 0.24
20	45.18	1.37 ± 0.21	15.01	1.13 ± 0.19	11.32	1.36 ± 0.18	6.41	1.36 ± 0.19	5.81	1.17 ± 0.18
50	18.67	0.74 ± 0.10	6.08	0.81 ± 0.14	4.57	0.78 ± 0.11	2.72	0.81 ± 0.12	5.17	0.73 ± 0.10

[16] Weikun Wang et al. "Maximum likelihood estimation of closed queueing network demands from queue length data". In: *ICPE*. 2016.

MHE scalability analysis

	Errors				*Runtimes* (s)			
M	min	avg	95-th	max	min	avg	95-th	max
5	1.60	2.53	4.12	4.50	0.03	0.03	0.03	0.04
10	1.63	2.46	3.28	3.56	0.08	0.08	0.09	0.09
15	1.59	2.63	3.48	4.56	0.18	0.18	0.19	0.19
20	1.62	2.52	3.19	3.82	0.34	0.38	0.48	0.52

Conclusion

- We defined optimization methods based on fluid queuing networks enabling fast adaptation reactions under strict time constraints.
- Through an extensive numerical validation we have shown how complex multi-dimensional adaptation actions can be computed at a small computational cost in real-world scenarios.
- We developed the first transient estimation technique for service demands of QNs enabling effective and efficient parameter estimation in a minimally intrusive manner.

Future work

- The definition of model predictive control problem based on the fluid interpretation of layered queuing networks.
- The definition of performance-driven self-adaptation approaches considering higher-order statistics of the controlled quantities, e.g., variance
- The development of minimally intrusive estimation techniques for multi-class QNs with non-exponentially distributed service times

Further related work

IMT

- A number of results on self-adaptation by means of formal quantitative verification.[17] [18]
- Runtime use of discrete time Markov chains for reliability of self-adaptive systems.[19]
- Control-theoretic approaches for learning and updating a linear model from measurements and controlling it via a single parameter.[20]
- Control of software performance via service-rate adaption using a queuing model.[21]

[17] Carlo Ghezzi et al. "Managing non-functional uncertainty via model-driven adaptivity". In: *ICSE'13*.

[18] A. Filieri, G. Tamburrelli, and C. Ghezzi. "Supporting Self-adaptation via Quantitative Verification and Sensitivity Analysis at Run Time". In: *IEEE Trans. Software Eng.* (2016).

[19] Antonio Filieri et al. "Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements". In: *ASE'11*.

[20] Antonio Filieri, Henry Hoffmann, and Martina Maggio. "Automated Design of Self-adaptive Software with Control-theoretical Formal Guarantees". In: *ICSE'14*.

[21] Davide Arcelli et al. "Control theory for model-based performance-driven software adaptation". In: *QoSA'15*.