Chameleon: Design and Evaluation of a Proactive, Application-Aware Elasticity Mechanism

André Bauer, Simon Spinner, Nikolas Herbst

Chair of Software Engineering
University of Würzburg
http://se.informatik.uni-wuerzburg.de/

SPEC RG Cloud Work Group
Würzburg, 25/10/2016

Agenda

1. Motivation
2. Related Work
3. Approach
4. Evaluation
5. Conclusion

Motivation

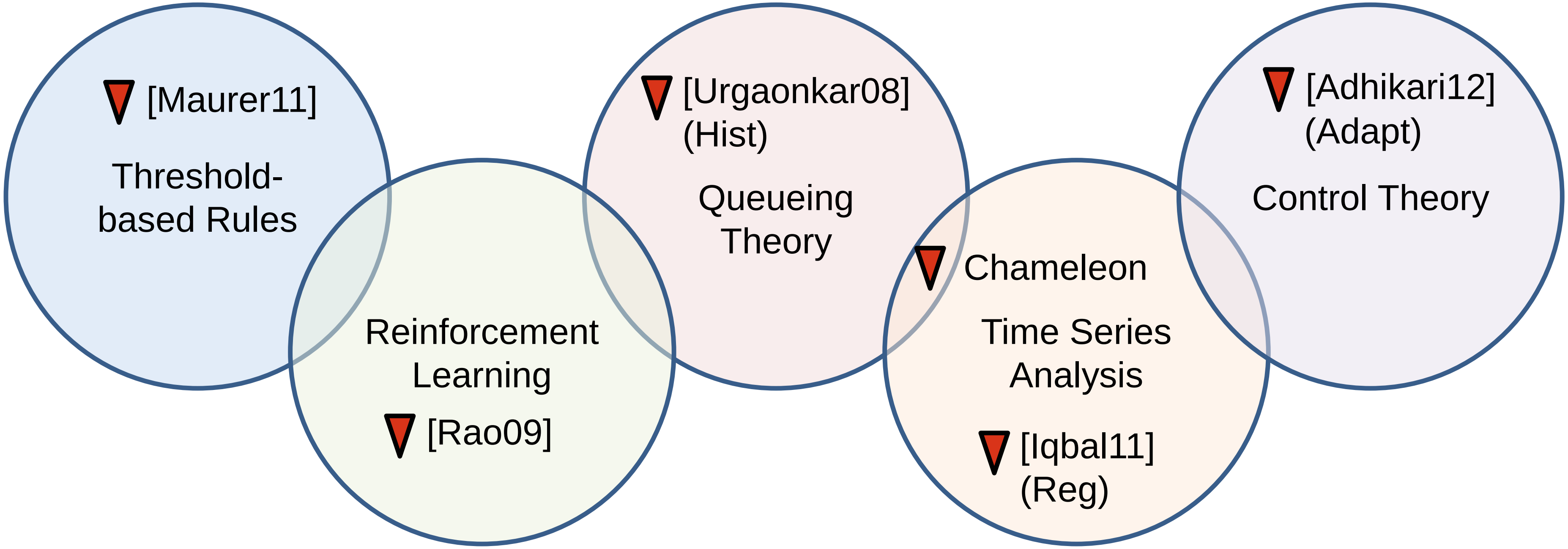
- Cloud computing is a paradigm which faces the increasing scale and complexity of modern internet services

- To guarantee a reliable service, most application run with a fixed amount of resources
  - High energy consumption, if the system is not fully utilized
  - Bad performance, if unexpected peaks appear

➡ High quality auto-scalers are required, which reconfigure the system regarding its load

State of the Art

- Auto-scalers can be classified into 5 groups [Lorido-Botran14]
- Prominent examples of each group:



- Mechanism are either generic or application-specific

➡ Delimitation: Chameleon is both generic **and** application-aware

In a Nutshell…

Problem

Existing auto-scalers are either application-specific or generic.
Bad performance can be translated into loss of money or customers

Idea

Controller combines time series analysis and query theory enriched with application knowledge

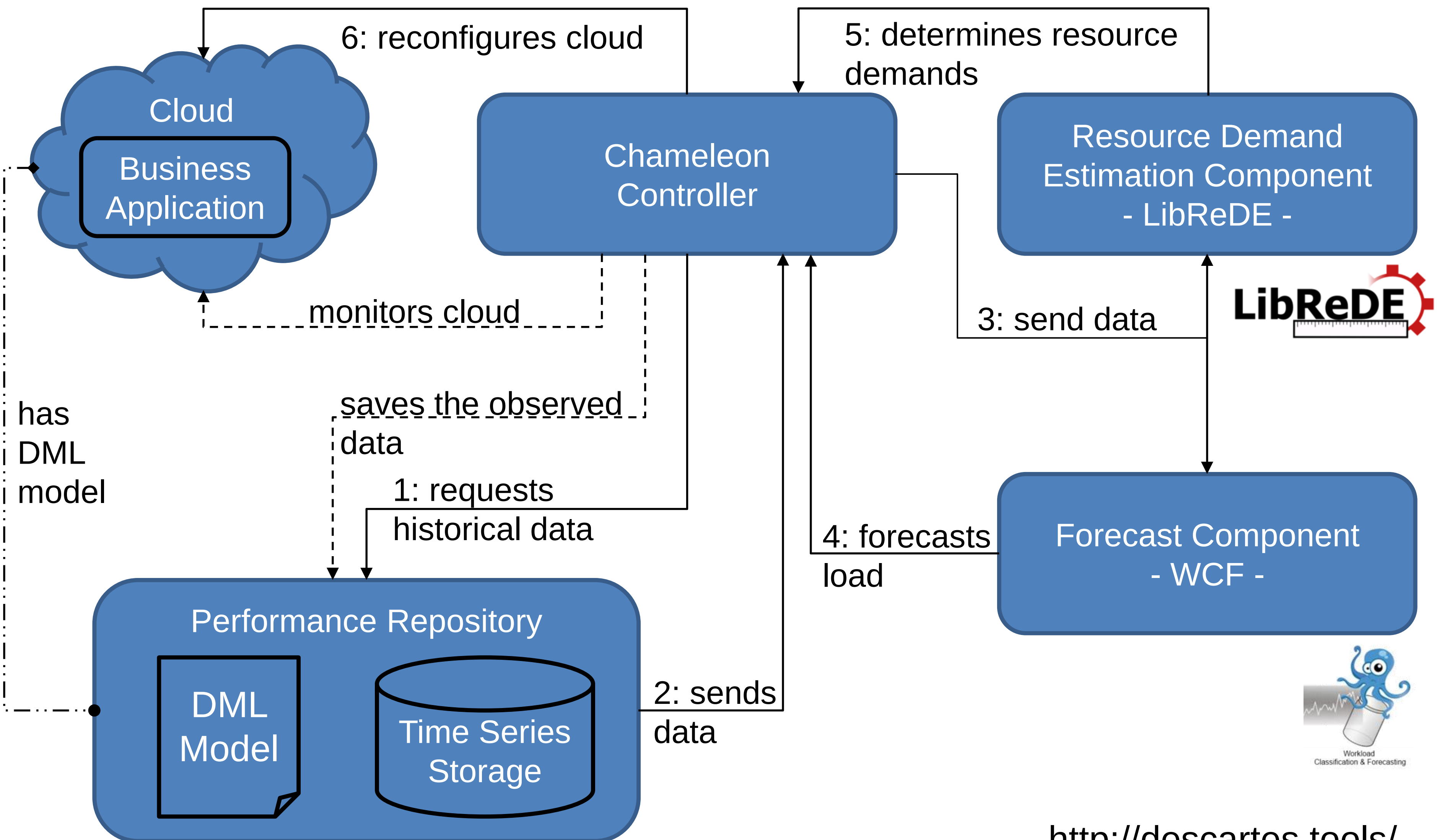
Benefit

Expected: More accurate scaling decisions compared to other controllers or standard reactive

Action

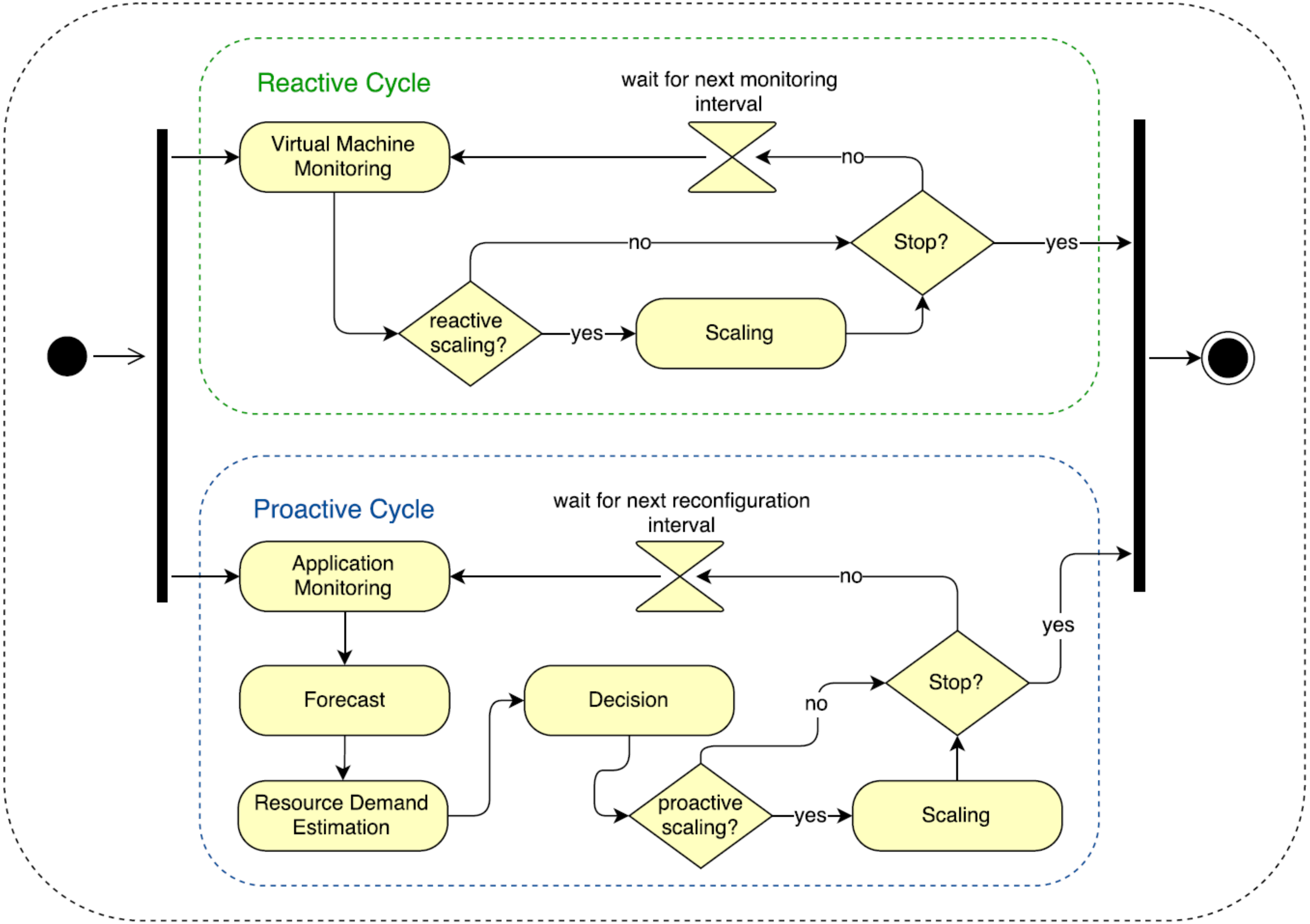
Taking forecasting, resource demand estimation and application model into account to increase the performance of the reconfiguration

Data Flow of Chameleon



http://descartes.tools/

Chameleon as State Machine



Assumptions & Evaluation Overview

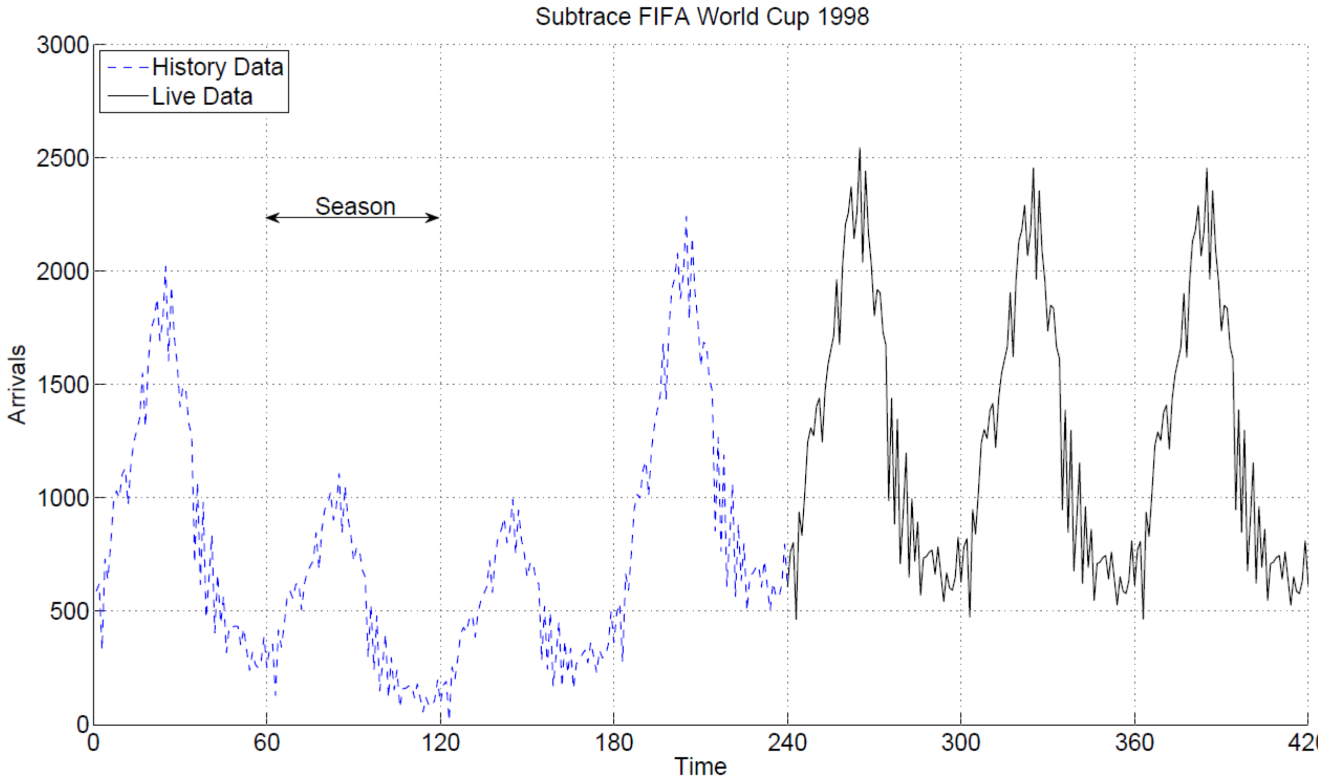
- Assumptions
  - Availability of at least 4 days of historical data
  - The SLO is the response time
  - Chameleon is limited to scale CPUs
  - A DML model is already provided

- Chameleon is driven with two different scenarios
  1. CPU intensive scenario
  2. The business application SPECjEnterprise2010

- Both scenarios are deployed in a private CloudStack environment

Workload

- FIFA world cup 1998 is used for an authentic workload
  - A sequence of 7 days were cropped out
  - 4 days as historical data for WCF
  - 3 days for the evaluation



Benchmarking

- Evaluation with Bungee experiment controller
  - Perform each scenario with Chameleon
  - Perform each scenario with other auto-scalers
    1. Hist [Urgaonkar08]
    2. Reg [Iqbal11]
    3. Adapt [Adhikari12]
    4. ConPaaS [Pierre12]
  - Compare the results with benchmarking metrics

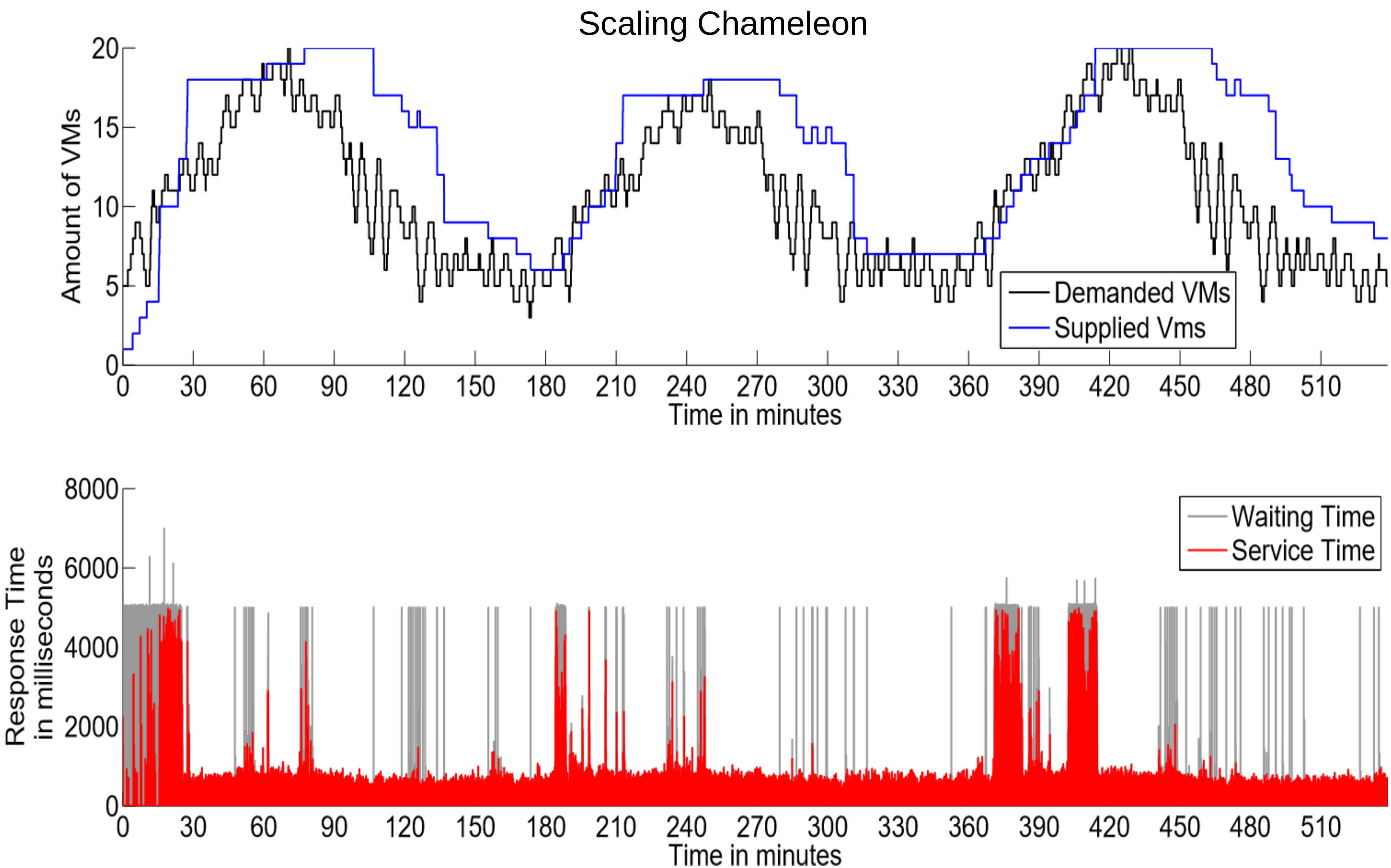
Server speed | Capacity | Load requests

- Benchmarking Metrics
  - Elasticity metrics
  - User metrics
    - Number of SLO violations
    - Average response time
    - 90th percentile of response time

CPU Intensive Scenario

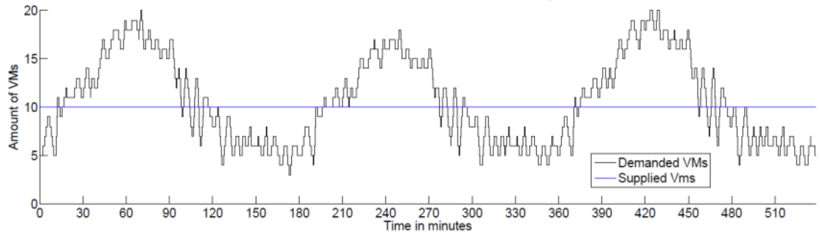
- Workload is accelerated 8 times: 3 d → 9 h
  - Between 40 and 200 request/second
  - 3.5 million requests
- Setup contains up to 20 WildFly slaves, 1 Domain controller and 1 Citrix Netscaler (LB)
  - 2 cores @2.6GHz
  - 4 GB memory
- Each slave is a http application
  - Request parameter n, here 600
  - LU decomposition of a random generated nxn matrix
- SLO: 90% of responses < 5 seconds

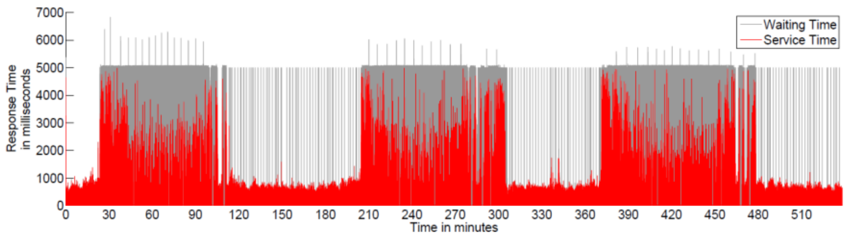
Evaluation Part I



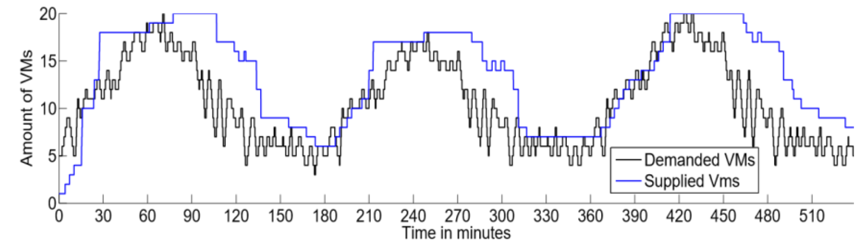
Evaluation Part II

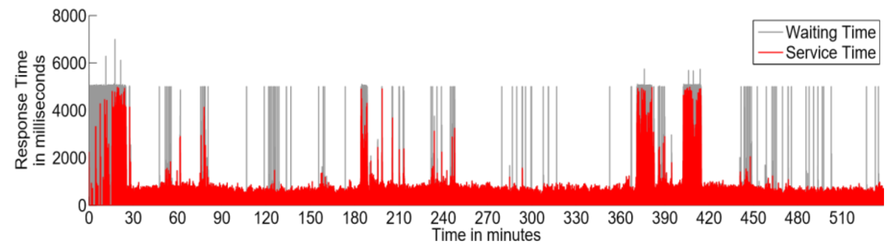
No Auto Scaling



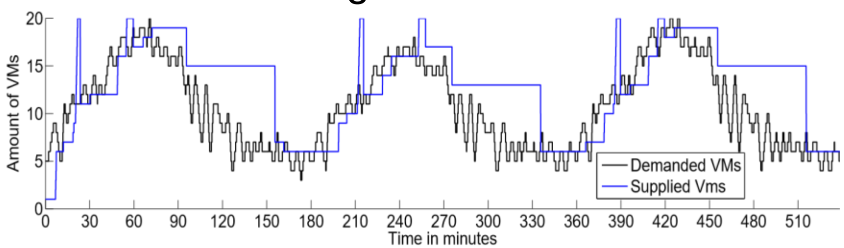


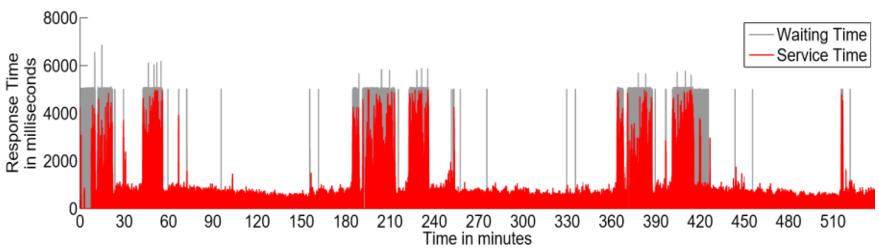
Scaling Chameleon



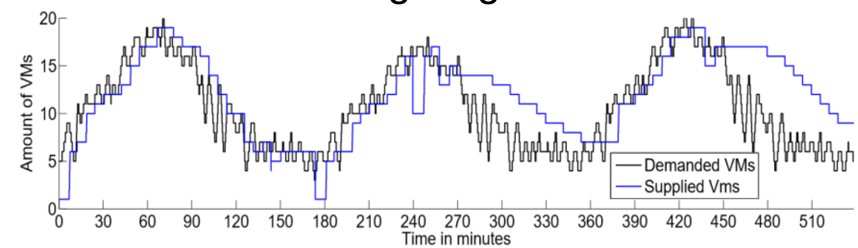


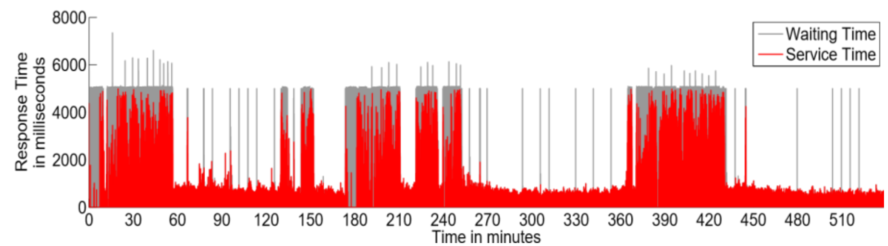
Scaling Hist





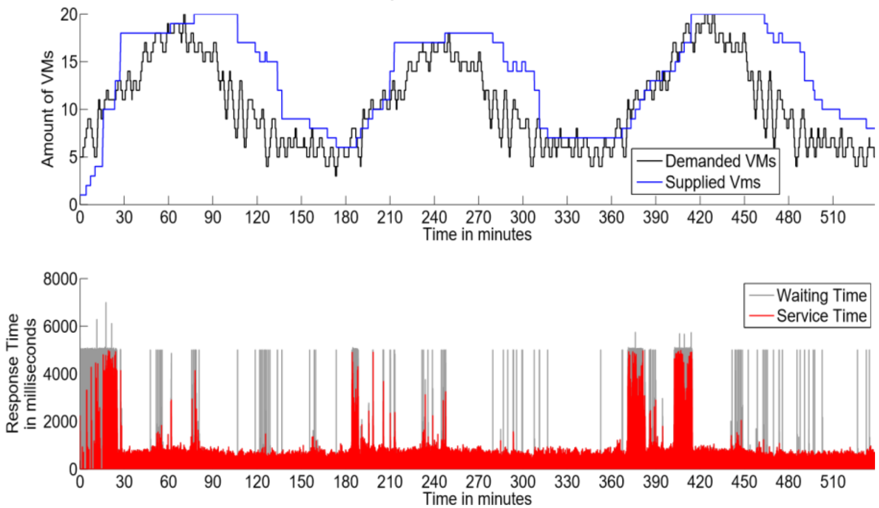
Scaling Reg



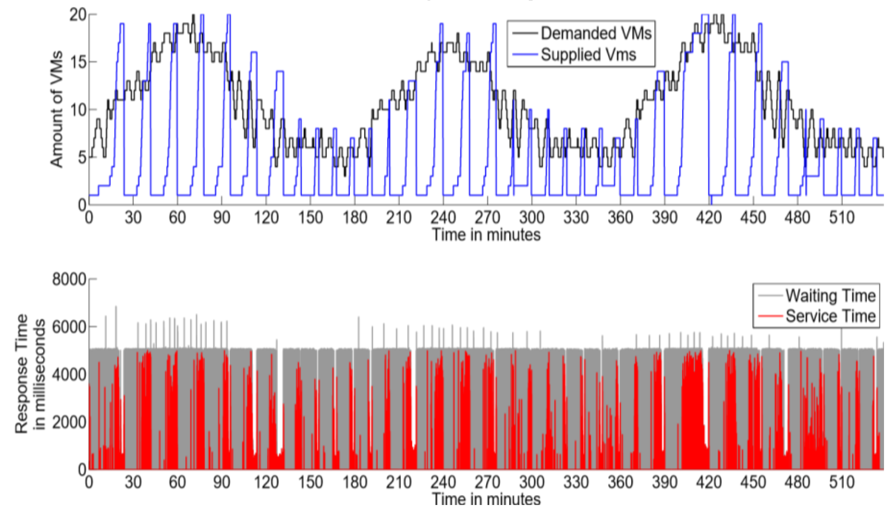


Evaluation Part III

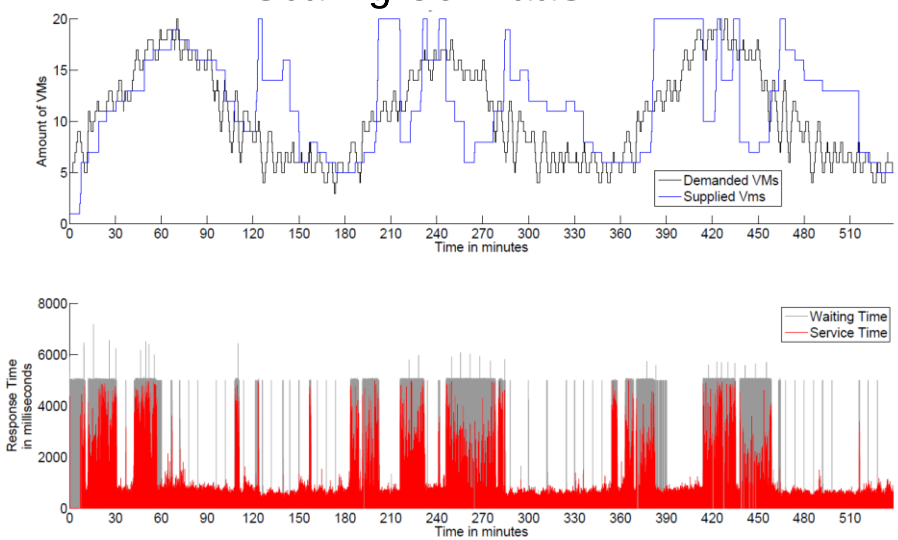
Scaling Chameleon



Scaling Adapt



Scaling ConPaaS



Metric Comparison

Metrics	No Scaler	Chameleon	Hist	Adapt	Reg	ConPaaS
$accuracy_o$	0.282	0.350	0.387	0.079	0.289	0.387
$accuracy_u$	0.147	0.030	0.055	0.613	0.078	0.105
$timeshare_o$	0.453	0.755	0.608	0.181	0.52	0.535
$timeshare_u$	0.491	0.129	0.285	0.788	0.371	0.355
jitter	-0.924	-0.723	-0.654	-0.615	-0.719	-0.466
instability	0.478	0.215	0.22	0.418	0.196	0.235
#Adaptions	0	85	110	618	93	204
Avg. #VMs	10	12.5	13.4	8.1	11.1	13.2
$score_{NoScaler}$	**1**	**1.330**	**1.151**	**0.896**	**1.177**	**1.143**
Avg. response time	1251 ms	671 ms	1267 ms	4339 ms	2018 ms	2082 ms
P_{90} response time	5002 ms	1051 ms	5002 ms	5002 ms	5002 ms	5002 ms
Violations	18.07%	7.50%	19.24%	82.75%	32.25%	35.98%

➡ Chameleon achieves highest score and lowest SLO violations

Future Work

- For Chameleon:
  - Add vertical scaling and migration
  - Improve precision by dynamic method selection
  - Integration of a DML extractor and solver
  - Taking billing of reconfigurations into account
- For evaluation:
  - Distributed JMeter usage
  - Calibration search vs. bottleneck shadowing
  - Retaking the experiments in a public cloud
  - Multitier scaling of SPECjEnterprise2010
  - Scale another multi-tier application (e.g., RuBiS)
  - Evaluate energy savings by auto-scaling

Conclusion

Problem

Existing auto-scalers are either application-specific or generic.
Bad performance can be translated into loss of money or customers

Idea

Chameleon combines time series analysis and query theory enriched with application knowledge

Results

Chameleon achieves the best score in the CPU intensive scenario compared with 4 existing auto-scaler

Ongoing Work

Release Chameleon as open source, …

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Thank you
for your attention

andre.bauer@stud-mail.uni-wuerzburg.de
http://se.informatik.uni-wuerzburg.de

Literature

- [Adhikari12] R. Adhikari and R. Agrawal, “An Introductory Study on Time Series Modeling and Forecasting,” arXiv preprint arXiv:1302.6613, 2013.
- [Iqbal11] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, “Adaptive Resource Provisioning for Read Intensive Multi-tier Applications in the Cloud,” Future Generation Computer Systems, vol. 27, no. 6, pp. 871-879, 2011.
- [Lorido-Botran14] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, “A Review of Autoscaling Techniques for Elastic Applications in Cloud Environments,” Journal of Grid Computing, vol. 12, no. 4, pp. 559-592, 2014.
- [Maurer11] M. Maurer, I. Brandic, and R. Sakellariou, “Enacting Slas in Clouds Using Rules,“ in Euro-Par 2011 Parallel Processing. Springer, 2011, pp. 455-466.
- [Rao09] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin, “VCONF: a Reinforcement Learning Approach to Virtual Machines Auto-conguration,” in Proceedings of the 6th international conference on Autonomic computing. ACM, 2009,pp. 137-146.
- [Urgaonkar08] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, “Agile Dynamic Provisioning of Multi-tier Internet Applications,“ ACM Transactions on Autonomous and Adaptive Systems (TAAS), vol. 3, no. 1, p. 1, 2008.
- [Pierre12] Pierre, Guillaume, and Corina Stratan. "ConPaaS: a platform for hosting elastic cloud applications." *IEEE Internet Computing* 16.5 (2012): 88-92.

Elasticity Metrics

Let s_t, d_t be the supply, demand at time t, with $t \in [0, T]$

$$accuracy_{U_k} = \frac{1}{T}\sum_{t=1}^{T}\frac{max(d_t - s_t, 0)}{d_t}\Delta t$$

$$accuracy_{O_k} = \frac{1}{T}\sum_{t=1}^{T}\frac{max(s_t - d_t, 0)}{d_t}\Delta t$$

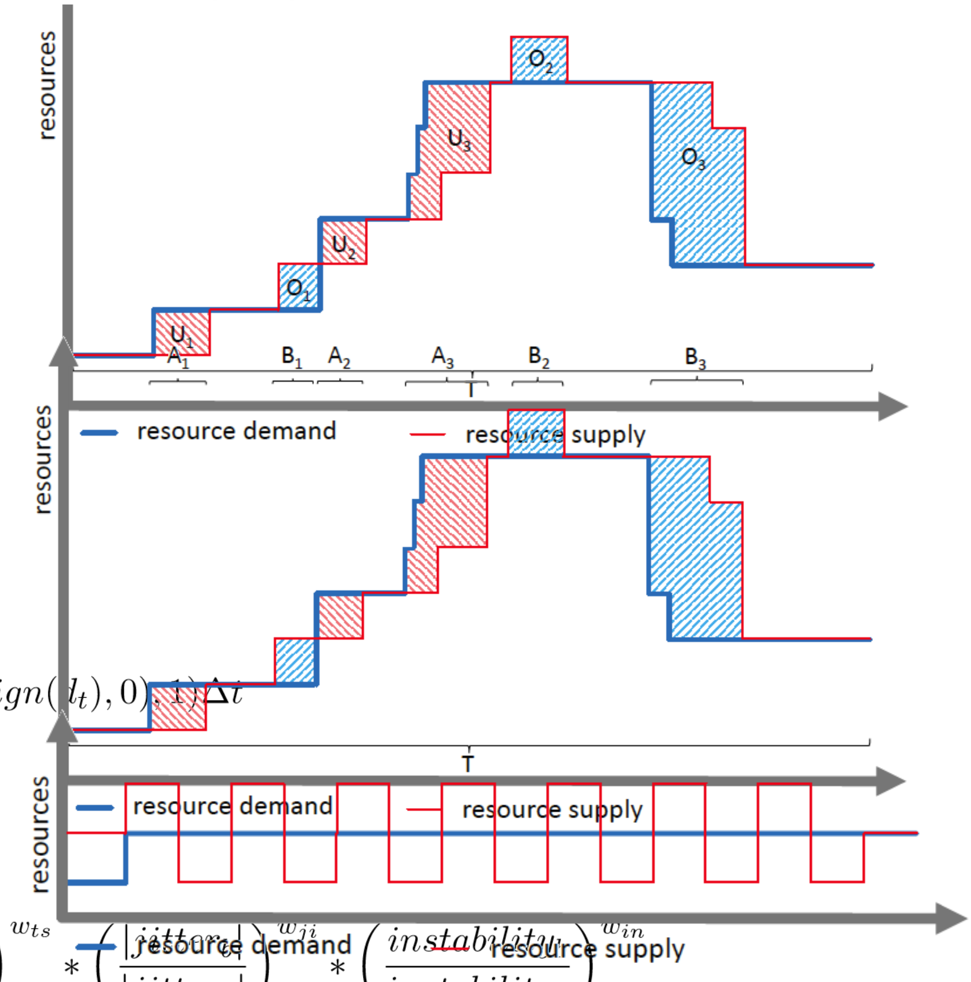
$$timeshare_{U_k} = \frac{1}{T}\sum_{t=1}^{T} max(sign(s_t - d_t), 0)\Delta t$$

$$timeshare_{O_k} = \frac{1}{T}\sum_{t=1}^{T} max(sign(s_t - d_t), 0)\Delta t$$

$$instability_k = \frac{1}{T-1}\sum_{t=2}^{T} min(max(sig(s_t) - sign(d_t), 0), 1)\Delta t$$

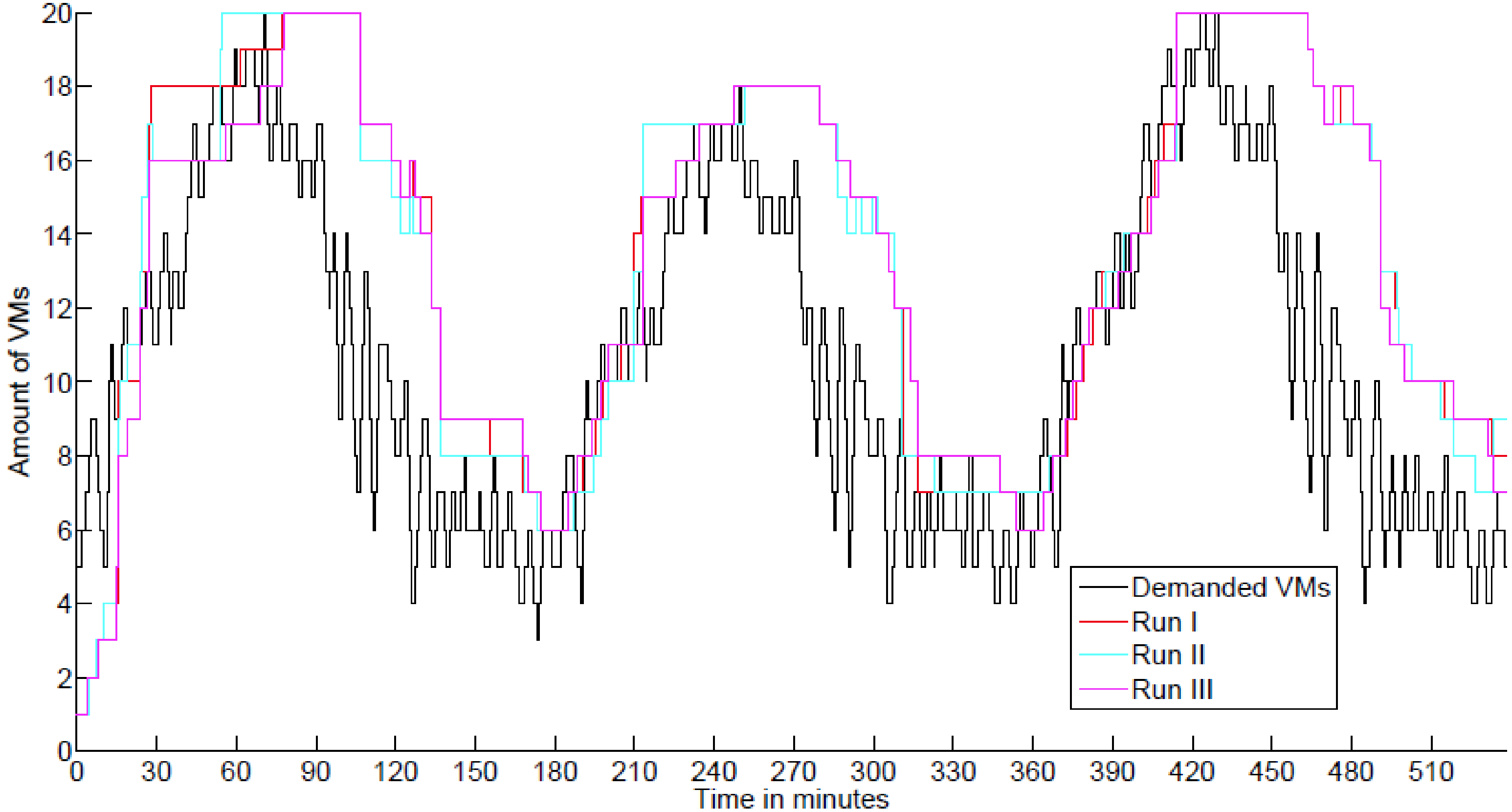
$$jitter_k = \frac{1}{T}\left(\sum_{t=2}^{T}|s_t - s_{t-1}| - \sum_{t=2}^{T}|d_t - d_{t-1}|\right)$$

$$score(\nu)_k = \left(\frac{accuracy_b}{accuracy_k}\right)^{w_{ac}} * \left(\frac{timeshare_b}{timeshare_k}\right)^{w_{ts}} * \left(\frac{|jitter_b|}{|jitter_k|}\right)^{w_{ji}} * \left(\frac{instability_b}{instability_k}\right)^{w_{in}}$$

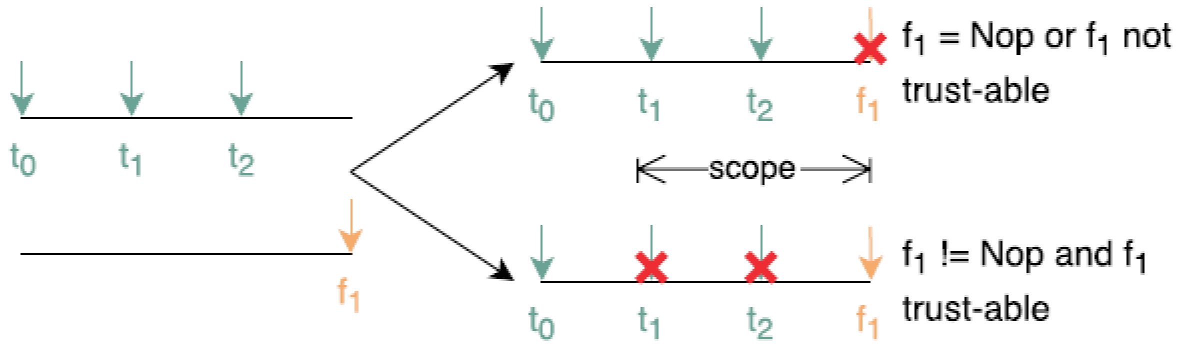


Reprodicibility

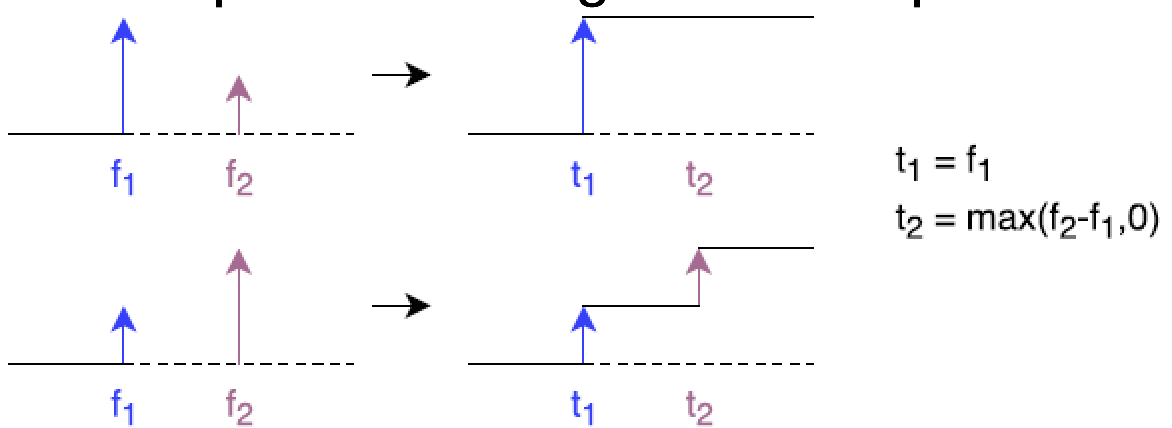
- Reproducibility of the scaling behaviour
  - 3 runs with the same setting
  - Elasticity score has a standard variation of 0.017

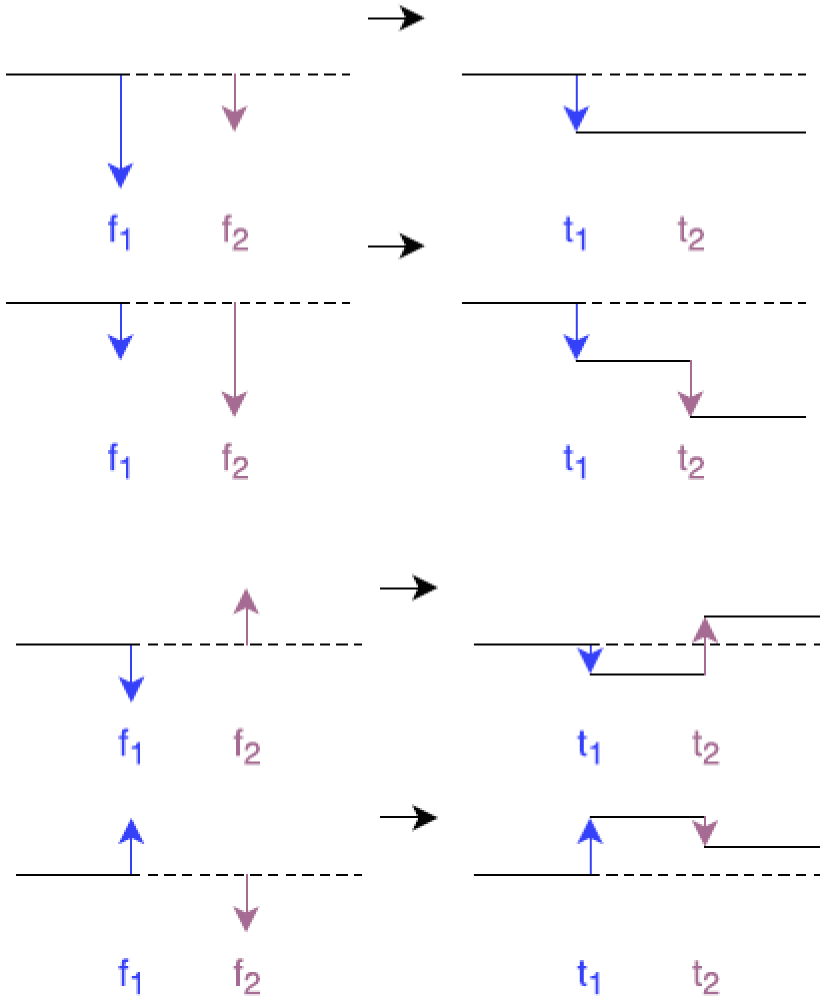


- Reactive scaling events vs. proactive scaling events



- Chameleon plans reconfiguration steps in advance



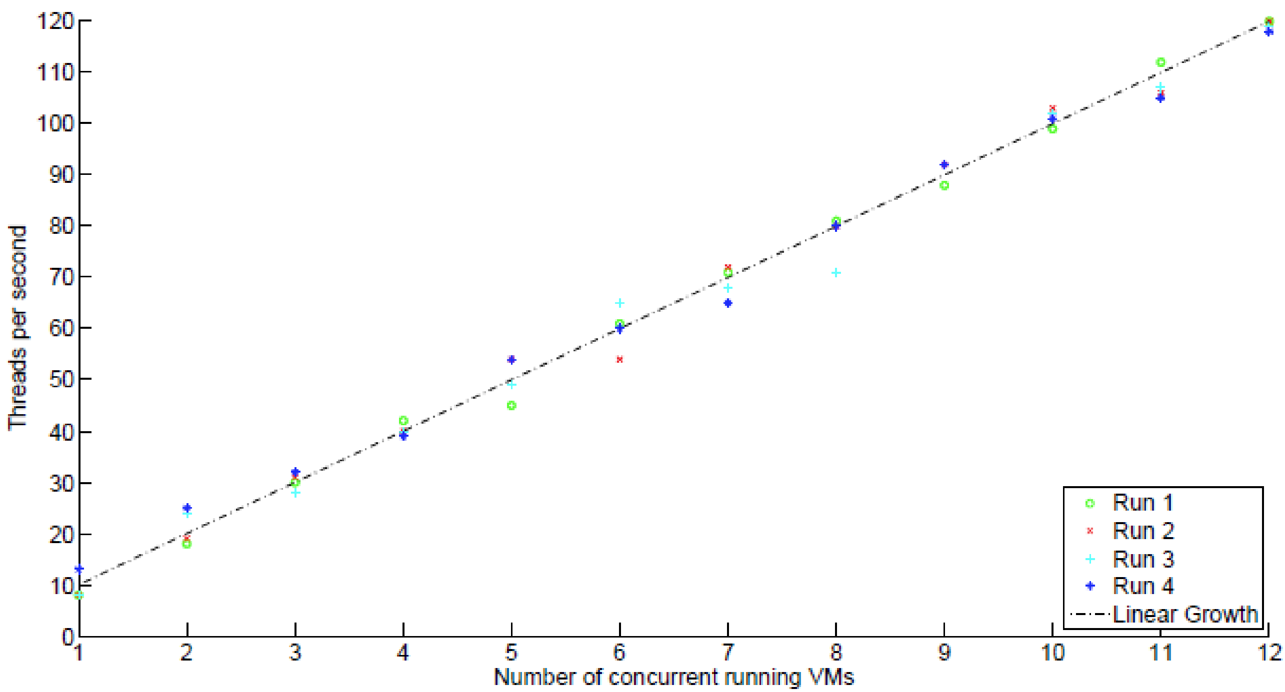


$$t_1 = \min(f_1, f_2)$$
$$t_2 = \max(f_2 - f_1, 0)$$

$$t_1 = \mathrm{floor}(d * f_1)$$
$$t_2 = \mathrm{ceil}(d * (f_2 + f_1))$$
$$0 < d \leq 1$$
$$t_1 = \mathrm{ceil}(d * f_1)$$
$$t_2 = \mathrm{floor}(d * (f_2 + f_1))$$

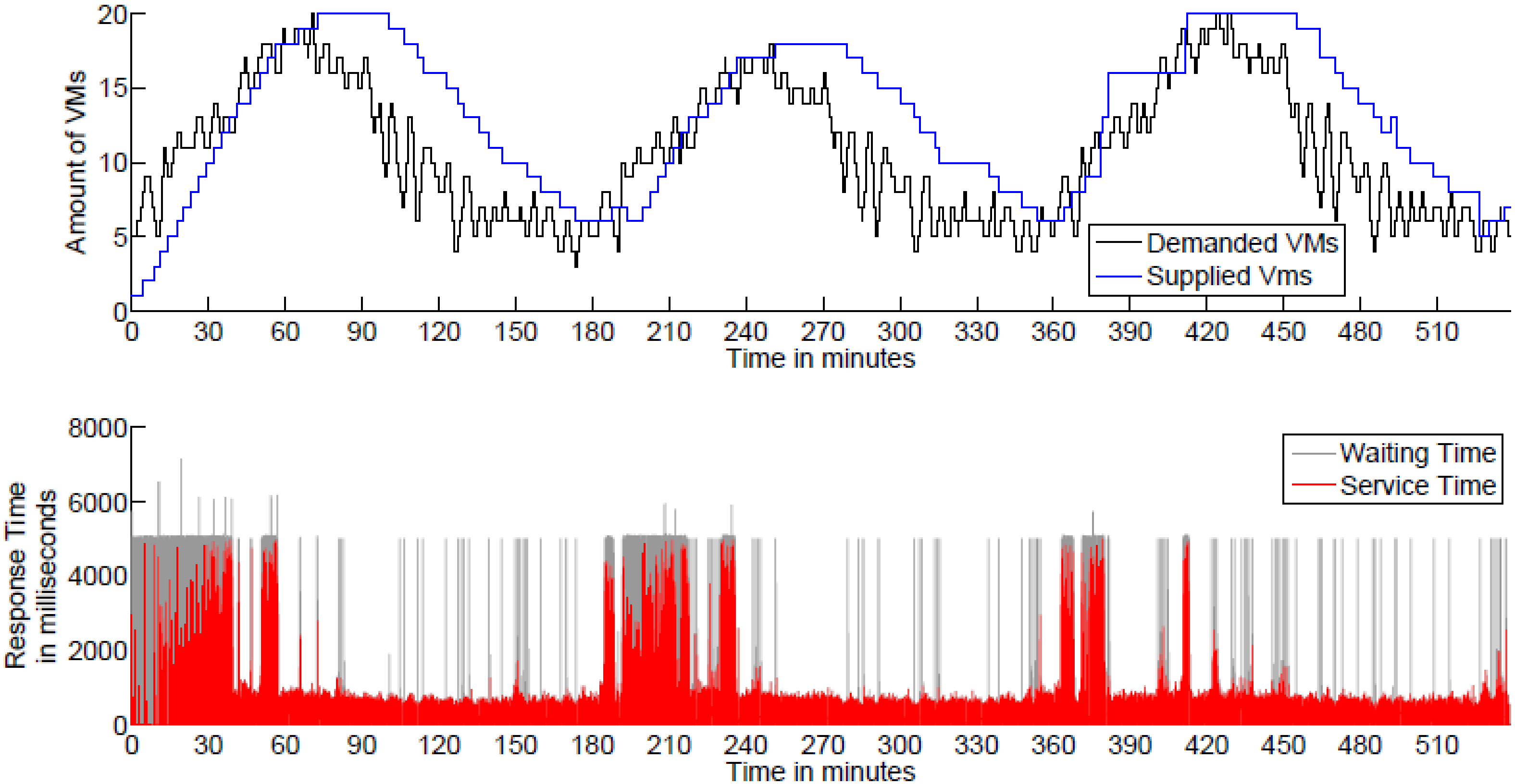
Resource Demand

- Resource demand for Chameleon is determined via LibReDE
- Resource demand for the existing auto-scalers is determined via the server speed



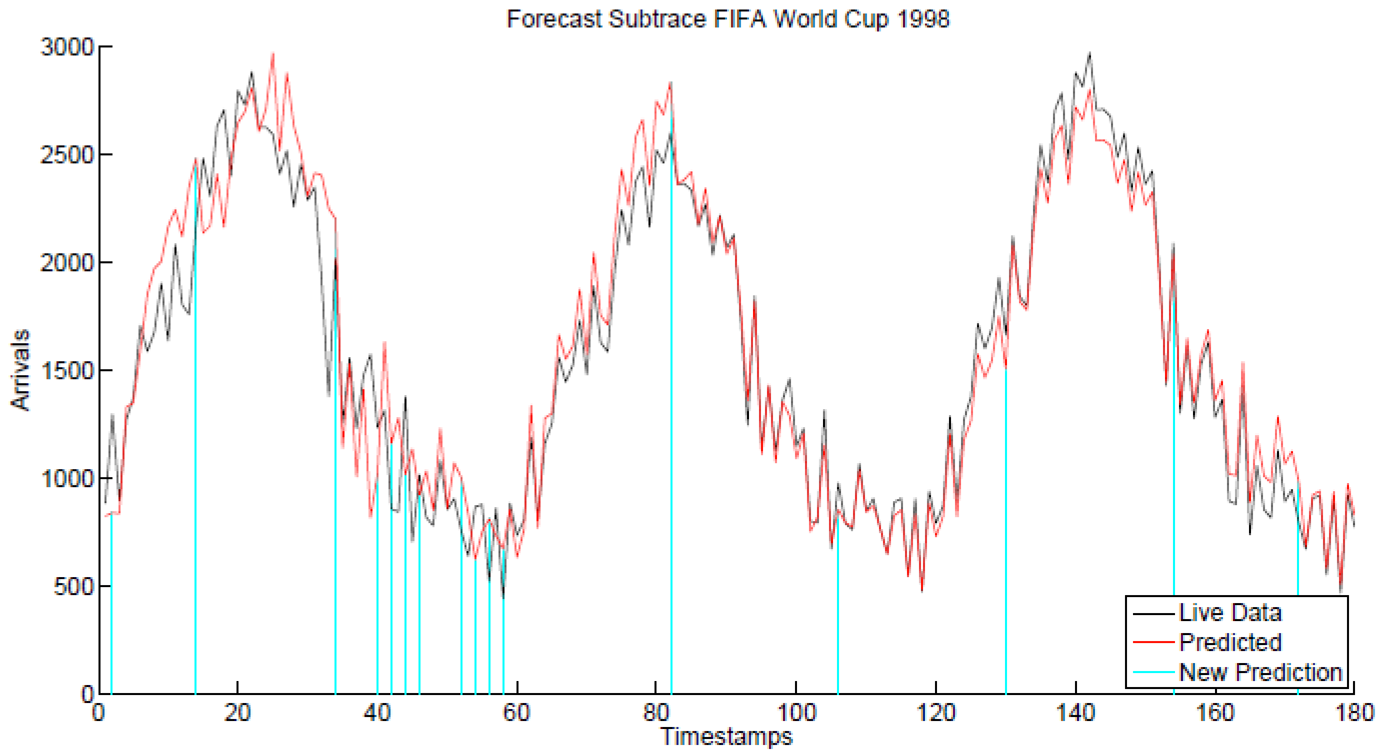
Without Historical knowledge

- Chameleon without historical data
  - Until minute 390: reactive scaling
  - Afterwards proactive scaling



Forecasting

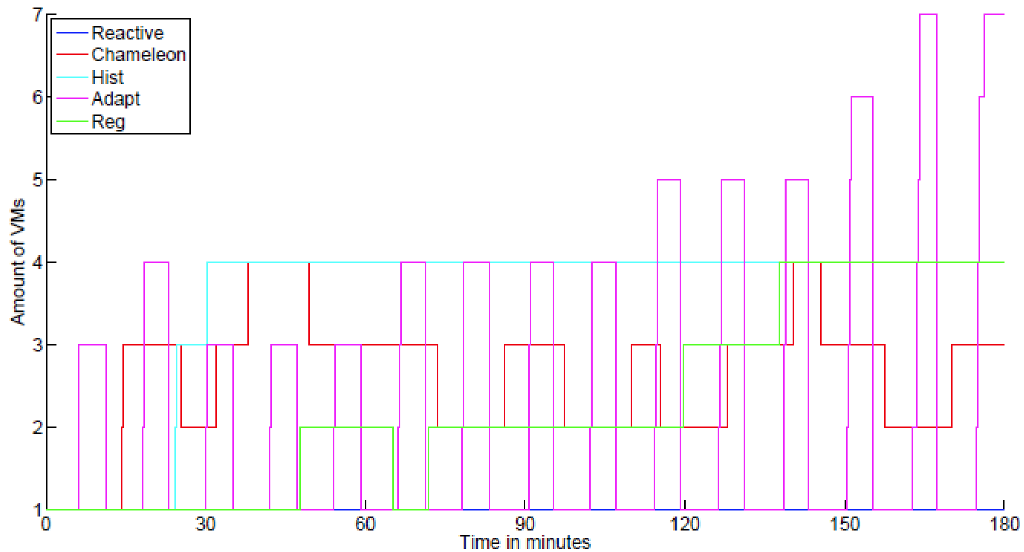
- Modified WCF forecast strategy:
  - Forecasts on demand
  - Forecast model contains the next 24 predictions
  - If MAPE > tolerance then build new forecast model



SPECjEnterprise2010

- Setup contains 8 different tiers (= WildFly slaves), 1 Domain controller and 1 Citrix Netscaler (LB)
- Problems with benchmarking
  - Very small response time (about 10 ms)
  - High load generation burden
  - Difficult calibration (reproducibility, hidden bottlenecks,…)
- SLO: 90% responses < 100 milliseconds
- Only user metrics can be determined
  - Number of SLO violations
  - Average & median response time
  - 90th percentile of response time

Metric Comparison



Metrics	Reactive	Chameleon	Hist	Adapt	Reg
Avg. response time	669 ms	476 ms	404 ms	592 ms	500 ms
P_{50} response time	9 ms	9 ms	7 ms	12 ms	10 ms
P_{90} response time	2005 ms	1876 ms	1754 ms	1899 ms	1915 ms
Violations	33,02 %	27,19 %	26,57 %	27,32 %	30,16 %

➡ Chameleon achieves the second best user metrics