Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Descartes research

Chameleon: A Hybrid, Proactive Auto-Scaling Mechanism on a Level-Playing Field

André Bauer

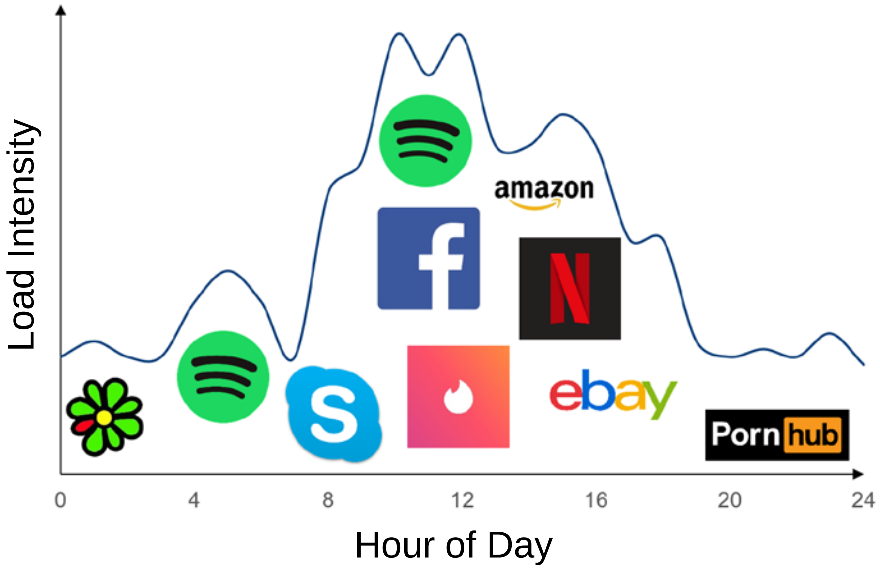
Workshop on Hot Topics in Cloud Computing Performance
Umeå, June 16, 2019

Auto-Scaling (AS) of Cloud Infrastructures

- Cloud infrastructure providers have to face changing requirements

- 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

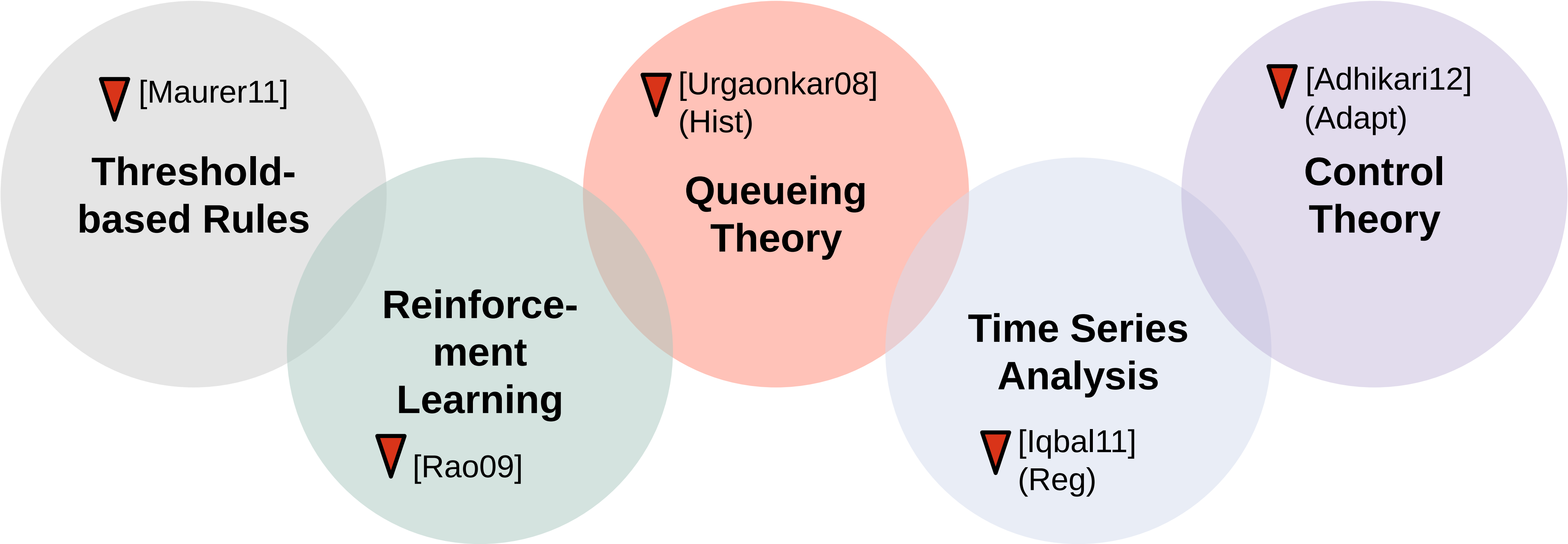
- High quality auto-scalers are required, which reconfigure the system regarding its load



Related Work on Auto-Scaling Methods

Auto-scalers can be classified into 5 groups [Lorido-Botran14]

Prominent examples are:



→ Predictive models from different disciplines are applied mostly in isolation.

→ Smart integration of multiple predictive/proactive with reactive mechanisms is missing.

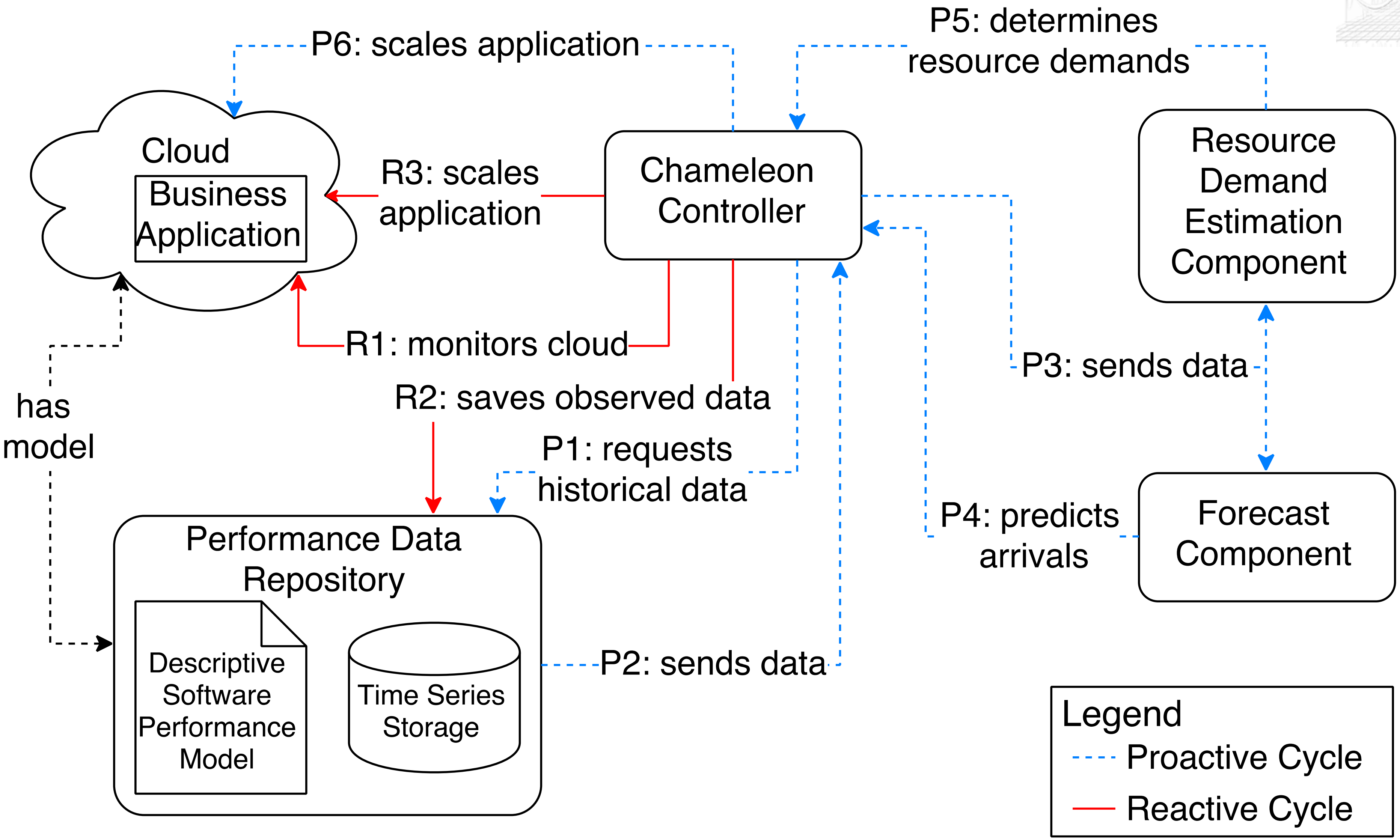
Challenges of Auto-Scaling

Based on related work,
we identify following challenges:

- Knowledge: models, history
- Awareness of own and system's performance and its boundaries
  - ➢ Descriptive performance model
- Guide to detect need/demand
  - ➢ Resource demand estimation
- Proactive planning of actions
  - ➢ Time series forecasting
- Reliable fallback options
  - ➢ Reactive cycle as fallback

A **resource demand** is the time a unit of work (e.g., request) spends obtaining service from a resource (e.g., CPU or hard disk) in a system (excluding waiting time). [Spinner15]

Chameleon Hybrid Auto-Scaler

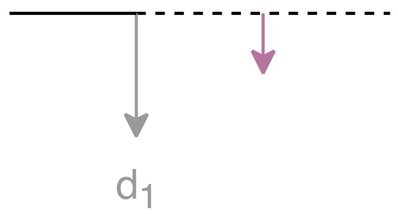


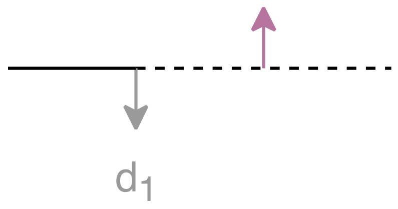
[Spinner17]

[Herbst13]

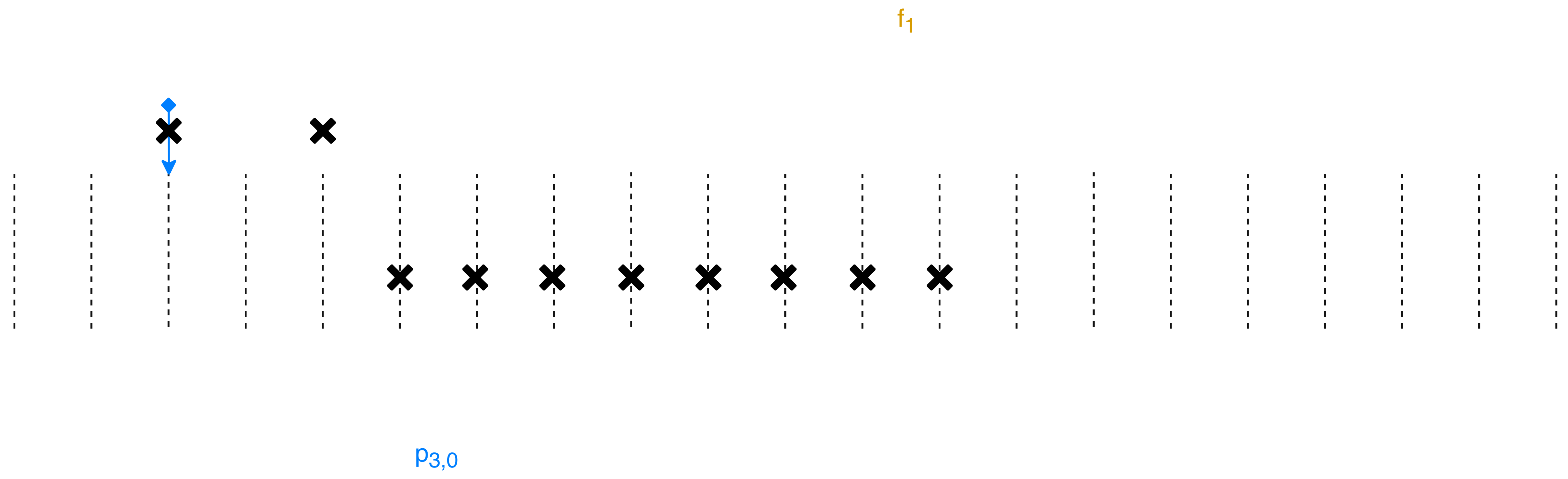
Chameleon Auto-Scaler: Decision Logic

- Simplification: Each service modelled as M/M/1/∞ queue
- Input: observed (reactive) and forecast (proactive) arrival rate
- Resource demand estimations
  based on monitored utilization, throughput and response time, e.g., service demand law
- Target utilization & response time
  → # resources add/remove
- Check “trustworthiness” of proactive scaling decisions
- Resolve conflicts in between proactive and reactive
- Optimize proactive scaling decisions pairwise





Chameleon: Example



Assumptions and Limitations

- Forecasting
  - 2 days of historical data is required
- Monitoring
  - Requests per second, response time and utilization are gathered by a monitoring infrastructure
- SLO
  - Response time of the application
- Use case
  - CPU intensive, request-based applications due to resource demand estimation
- Descriptive model
  - Can be transformed into a queuing network

Evaluation Setup

- Scaling a Java web application
  - Re-implementation of LU worklet from Rating Tool SERT^{TM}2
  - LU decomposition of nxn matrix, where n is GET parameter
- 3 different Environments
  - Private CloudStack
  - AWS EC2 IaaS cloud
  - Distributed ASCI Supercomputer 4 (DAS-4)
- 5 real-world traces
  - FIFA, BibSonomy, IBM, Wikipedia, and Retailrocket
  - 3 days each 3.2 hours → 9.6 hours experiment
- More than 400 hours of experiments

Benchmarking

- Evaluation with Bungee experiment controller [Herbst15]
  - Perform each scenario with Chameleon
  - Perform each scenario with standard reactive auto-scaler
  - Perform each scenario with sota auto-scalers
    - Hist [Urgaonkar08]
    - Reg [Iqbal11]
    - Adapt [Adhikari12]
    - ConPaaS [Pierre12]
  - Compare the results with benchmarking metrics
    - Individual elasticity metrics
    - Aggregate elasticity metrics
    - User metrics

Server speed | Capacity | Load requests

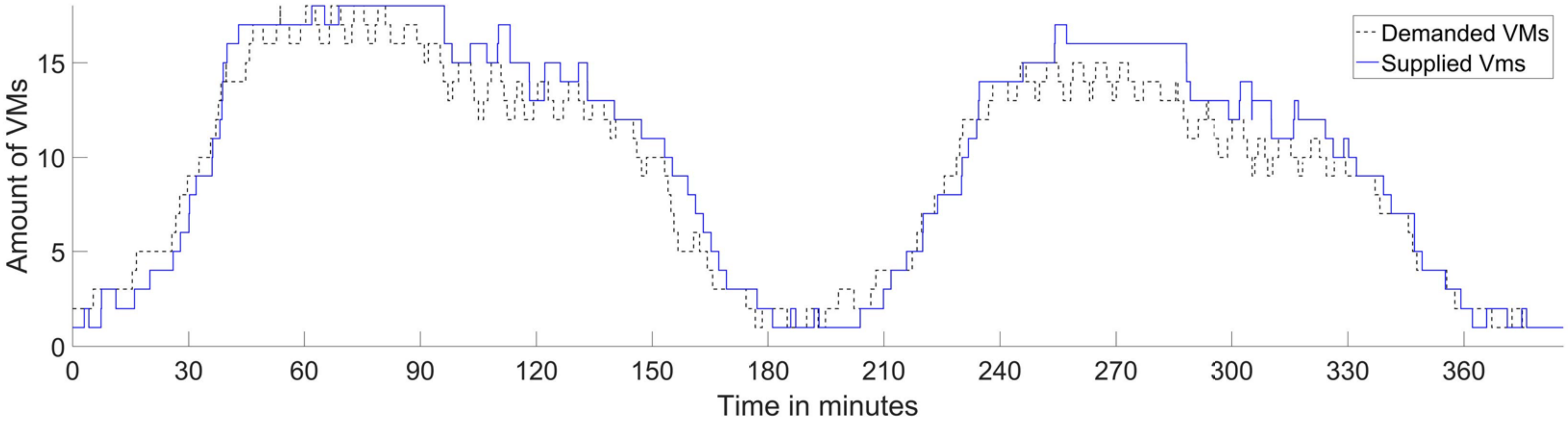
Cloud Elasticity Benchmark BUNGEE

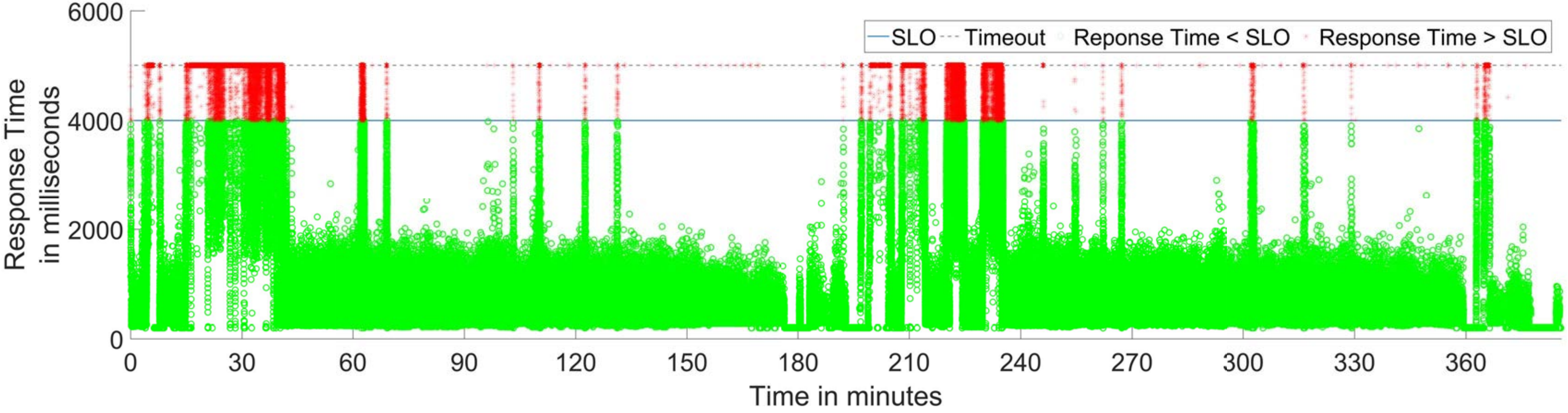
Elasticity Metrics

- Accuracy
- Timeshare
- Instability
- AS deviation
- Elasticity speed-up
- Pairwise competition

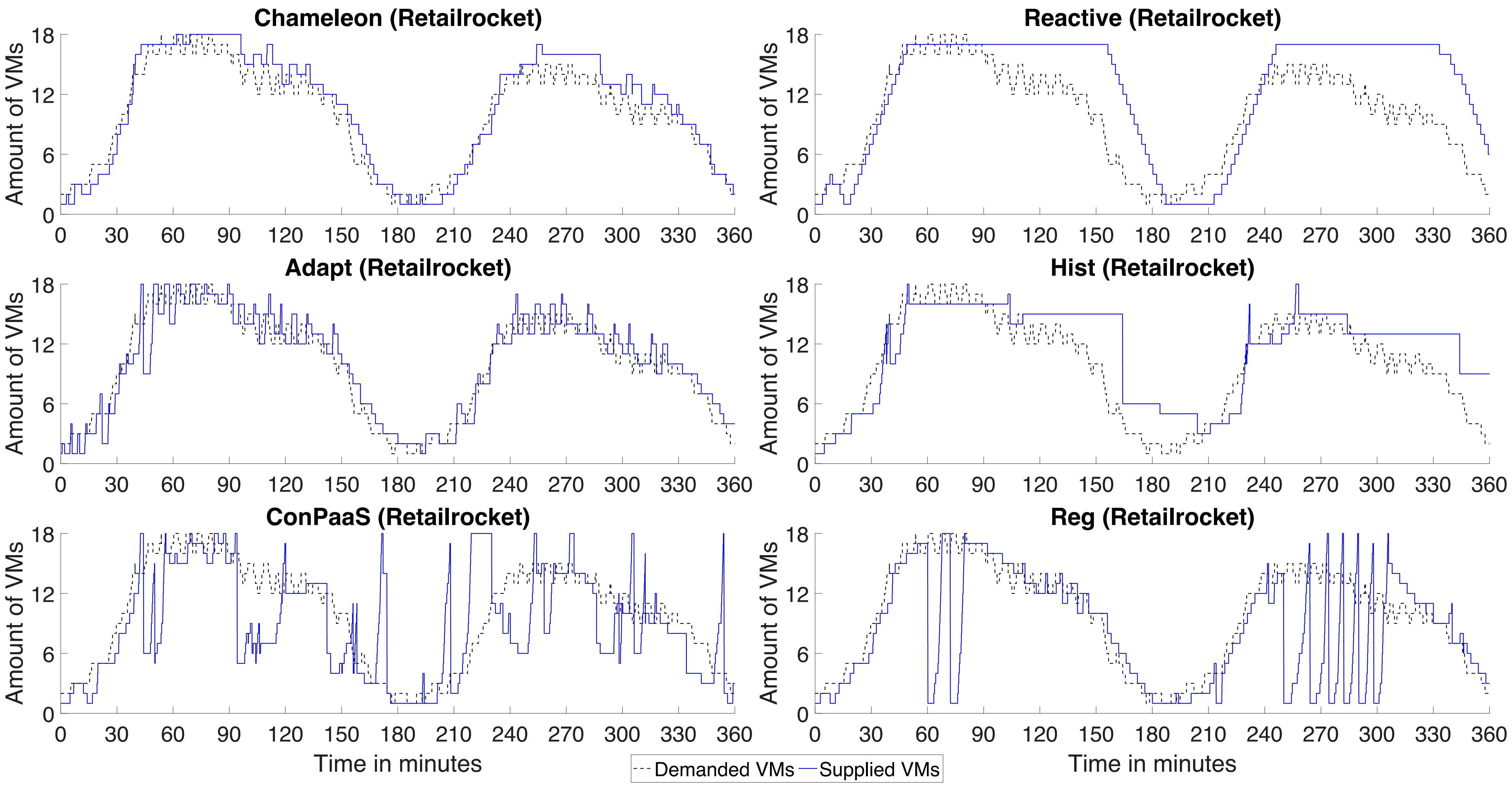
$$\kappa_a[\%] := \frac{1}{(n-1)\cdot|x|}\cdot\sum_{i=1;i\neq a}^{n}\sum_{j=1}^{|x|}\omega(i,j) \qquad \text{where} \qquad \omega(i,j) := \begin{cases} 0, & x_a(j) > x_i(j) \\ 0.5, & x_a(j) = x_i(j) \\ 1, & x_a(j) < x_i(j) \end{cases}$$

Experiment Example: Chameleon on CS, Retailrocket

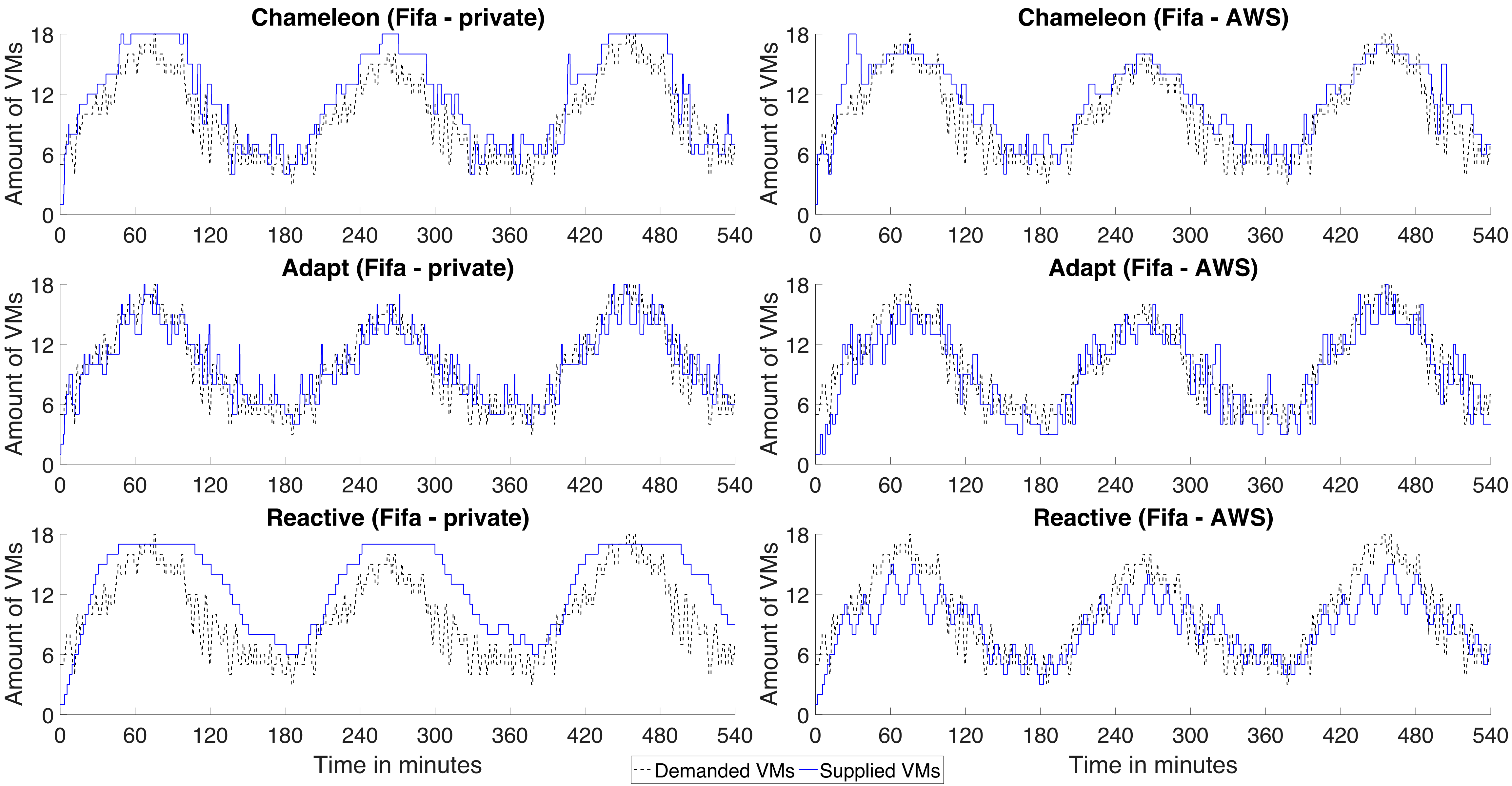




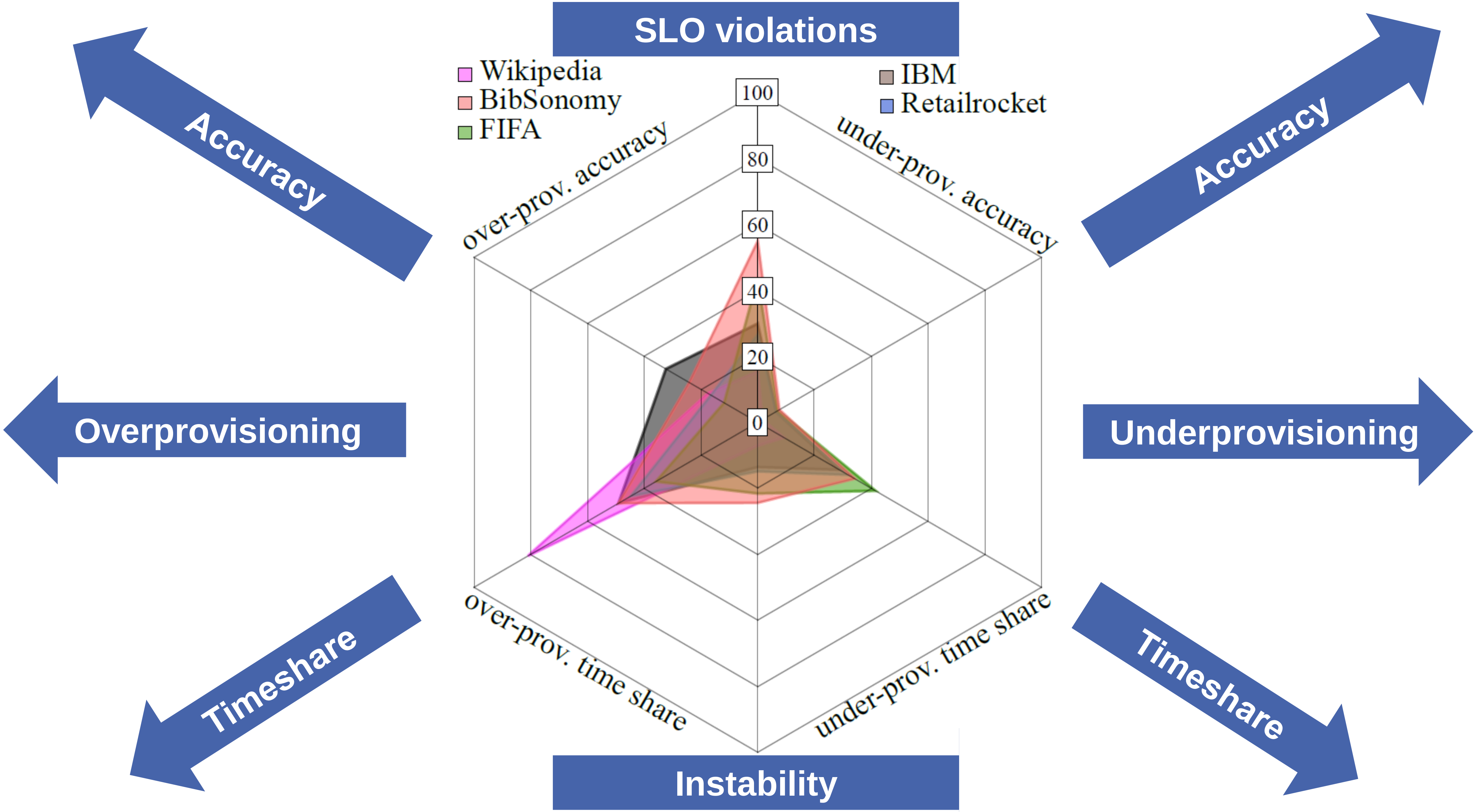
Experimental Evaluation: CS, Retailrocket



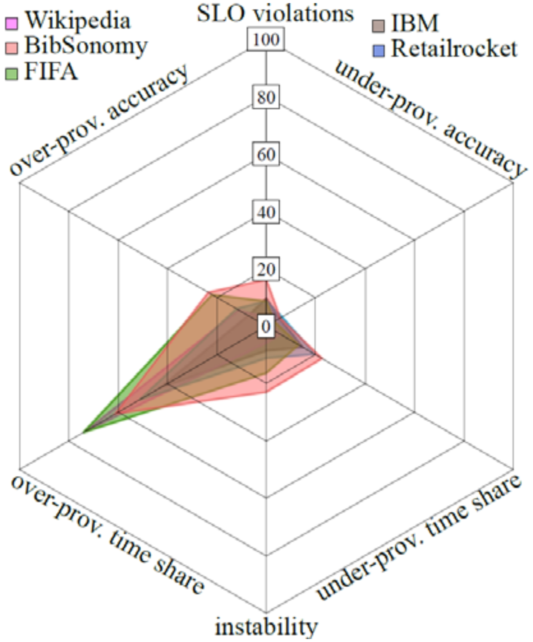
Experimental Evaluation: private CS vs. AWS EC2



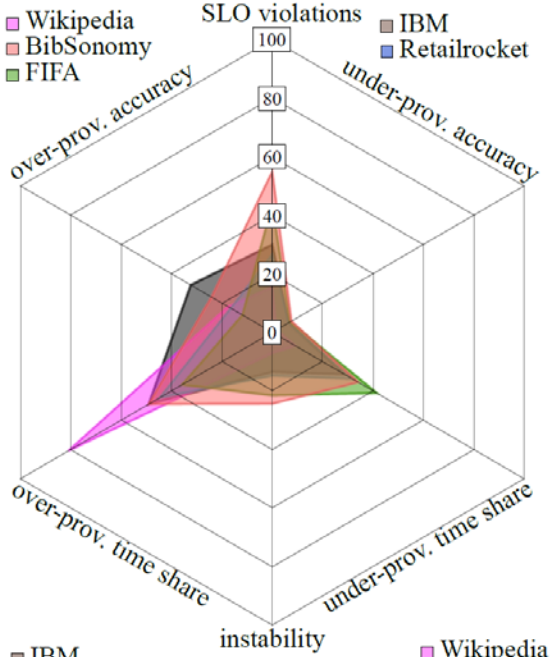
Result Visualization: Explanation



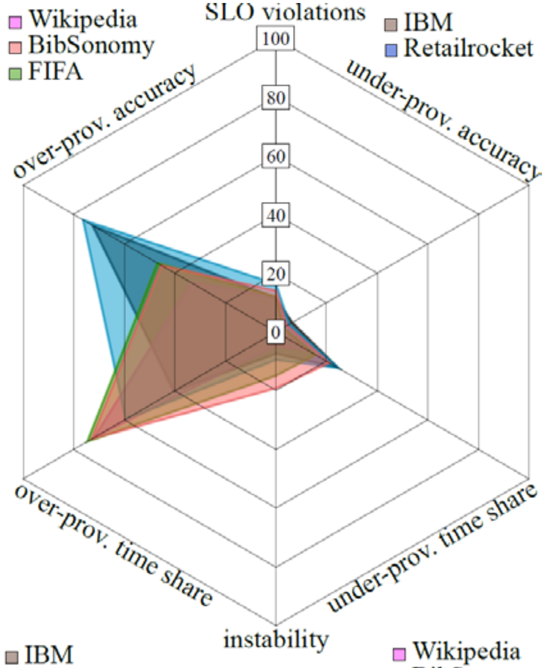
Chameleon



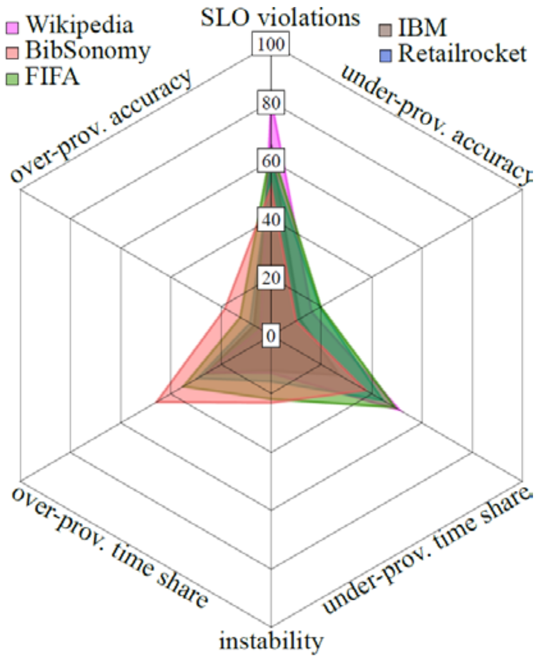
Adapt



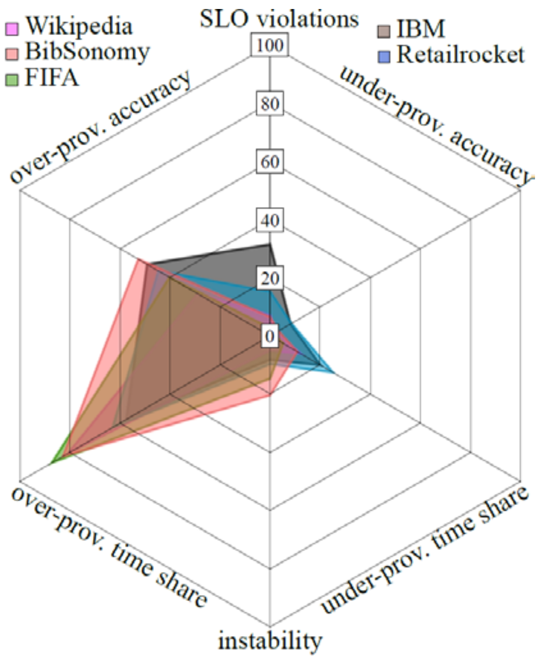
Hist



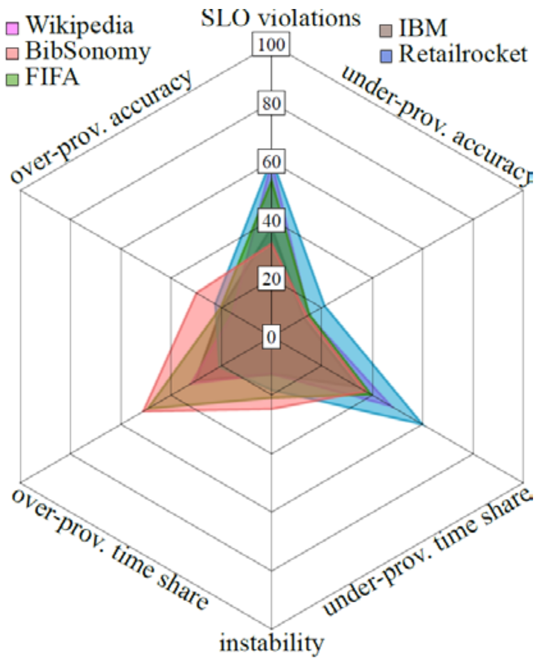
Reg



Reactive



ConPaas



Summary of all Experiments: Average Metrics

Metric	Chameleon	Adapt	Hist	ConPaaS	Reg	Reactive
$\overline{\theta}_U$ (avg. accuracy$_U$)	**3.63%**	6.45%	4.70%	15.55%	15.69%	6.98%
$\overline{\theta}_O$ (avg. accuracy$_O$)	17.88%	19.94%	52.64%	25.98%	**10.51%**	34.47%
$\overline{\tau}_U$ (avg. time share$_U$)	**13.32%**	30.43%	22.75%	42.04%	43.71%	25.41%
$\overline{\tau}_O$ (avg. time share$_O$)	65.06%	51.41%	62.35%	41.69%	**33.42%**	62.08%
$\overline{\upsilon}$ (avg. instability)	13.91%	16.60%	**11.95%**	17.42%	17.02%	12.99%
$\overline{\psi}$ (avg. SLO violations)	**10.29%**	32.76%	15.59%	44.11%	60.16%	21.96%
$\overline{\sigma}$ (avg. as deviation)	**39.63%**	46.90%	46.43%	54.03%	63.46%	48.14%
$\overline{\kappa}$ (avg. pairwise comp.)	**69.44%**	50.00%	58.33%	36.51%	42.46%	55.56%
$\overline{\epsilon}$ (avg. elastic speedup)	**2.02**	1.48	1.38	1.10	1.41	1.49

Auto-Scaler Benchmark Competition: Findings

- **Chameleon** outperforms in the evaluated scenarios
  - Reliable slight over-provisioning, lowest SLO violations
  - Coupling of proactive and reactive scaling decisions improves the elasticity
- **Adapt**: closely follows the demand, high number of adaptations
- **Hist** and **Reactive**: high over-provisioning accuracy
- **Reactive**: accurate, timely CPU utilization metrics required – not always reliable
- **ConPaaS** and **Reg**: unstable behavior – often not reliable

In a Nutshell

- Cloud Infrastructure providers have to face changing requirements
- High quality auto-scaler are required
  - Predictive models from different disciplines are applied mostly in isolation
  - Smart integration of multiple predictive/proactive with reactive mechanisms is missing
- Design of a hybrid auto-scaler Chameleon
- More than 400 hour evaluation in 3 different environments with 5 real-world traces
- Chameleon outperforms other auto-scalers

References

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.

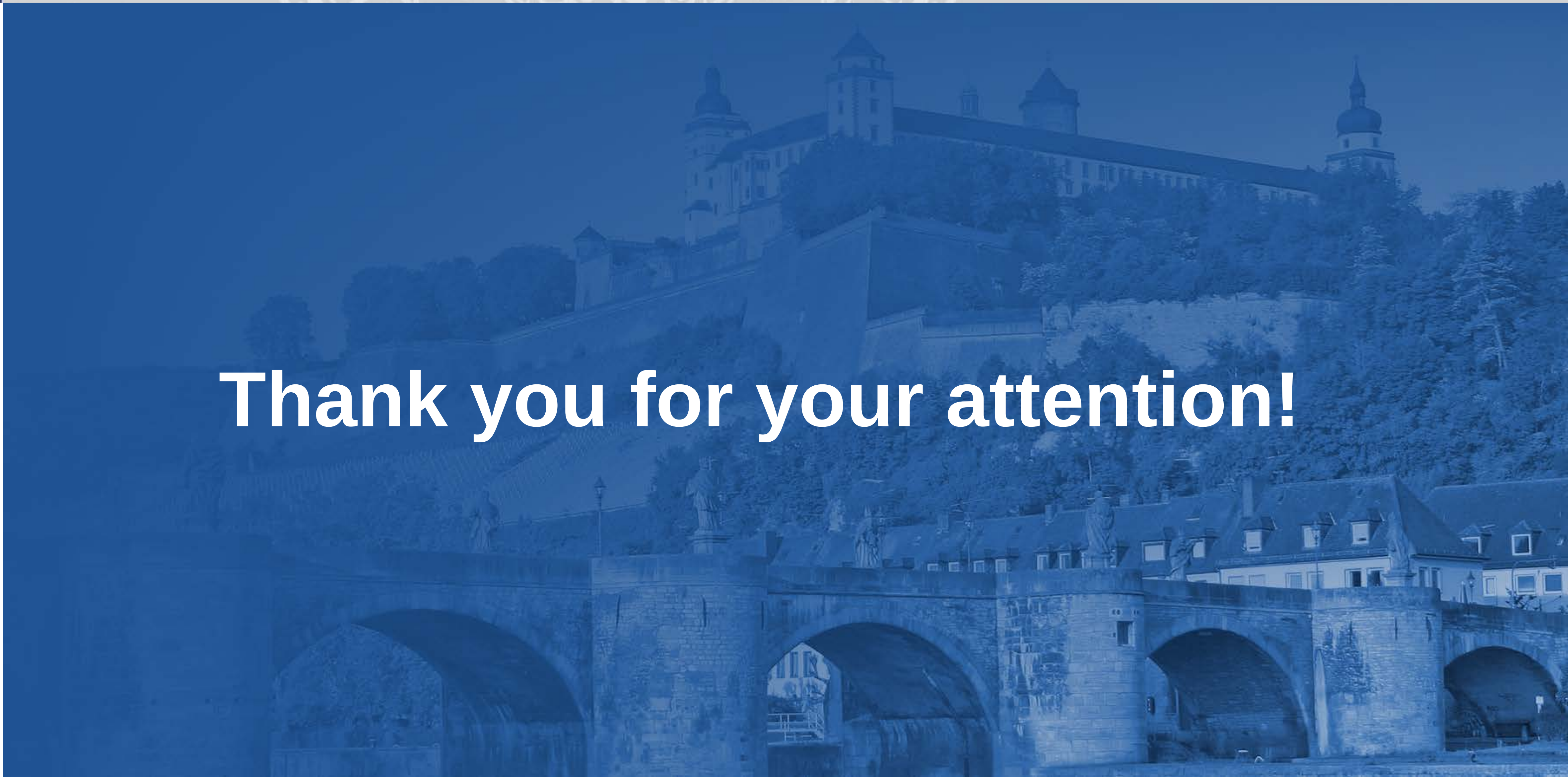
Herbst13 Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning. In Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE 2013), Prague, Czech Republic, April 21--24, 2013, pages 187--198. ACM, New York, NY, USA. April 2013.

Herbst15 Nikolas Roman Herbst, Samuel Kounev, Andreas Weber, and Henning Groenda. BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments. In Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015), Firenze, Italy, May 18--19, 2015.

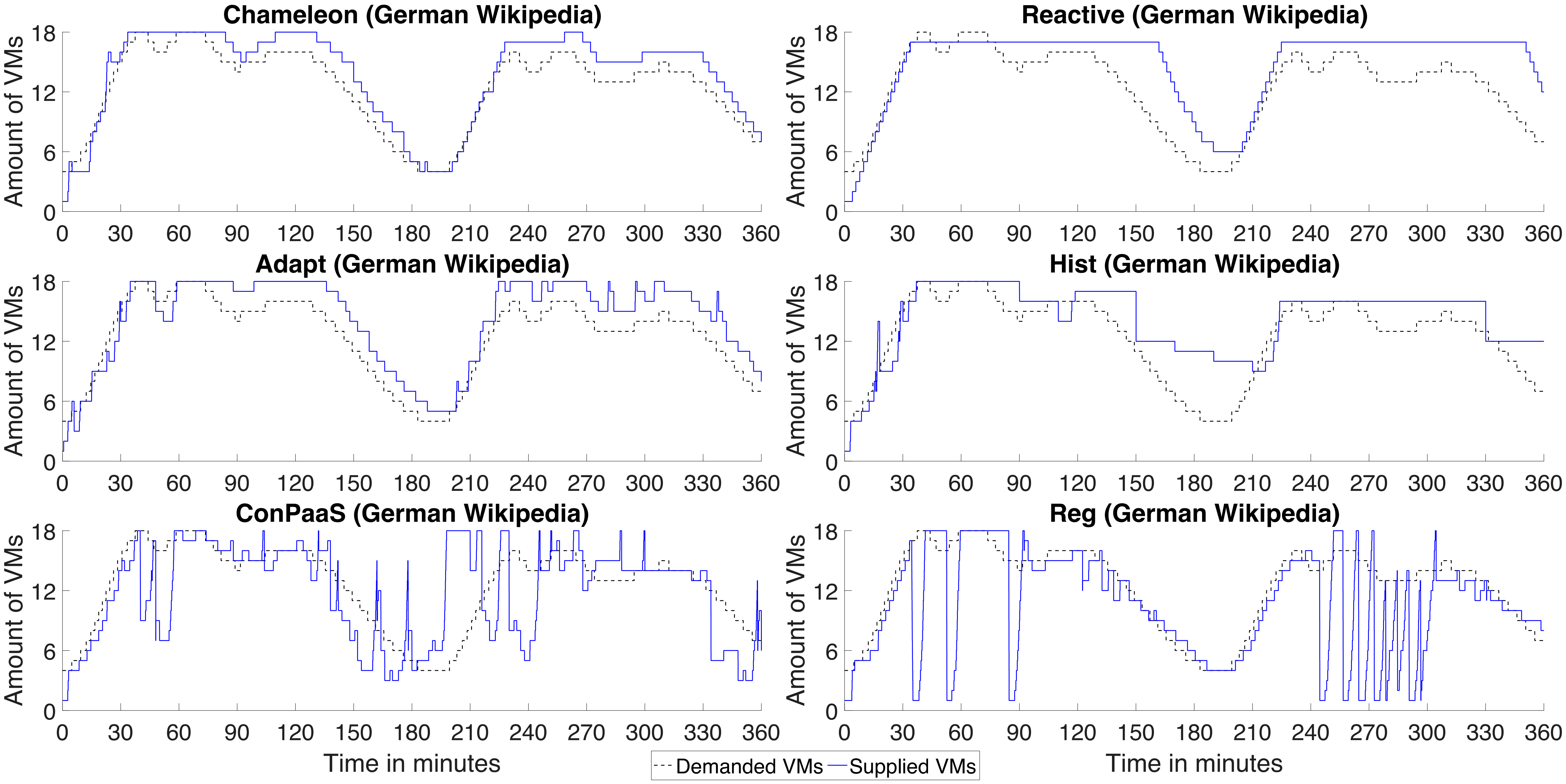
Bauer18 André Bauer, Nikolas Herbst, Simon Spinner, Ahmed Ali-Eldin, and Samuel Kounev. Chameleon: A Hybrid, Proactive Auto-Scaling Mechanism on a Level-Playing Field. IEEE Transactions on Parallel and Distributed Systems, 30(4):800 -- 813, April 2019, IEEE.

Spinner17 Simon Spinner, Antonio Filieri, Samuel Kounev, Martina Maggio, and Anders Robertsson. Run-time Models for Online Performance and Resource Management in Data Centers. In Self-Aware Computing Systems, Samuel Kounev, Jeffrey O. Kephart, Aleksandar Milenkoski, and Xiaoyun Zhu, editors. Springer Verlag, Berlin Heidelberg, Germany, 2017.

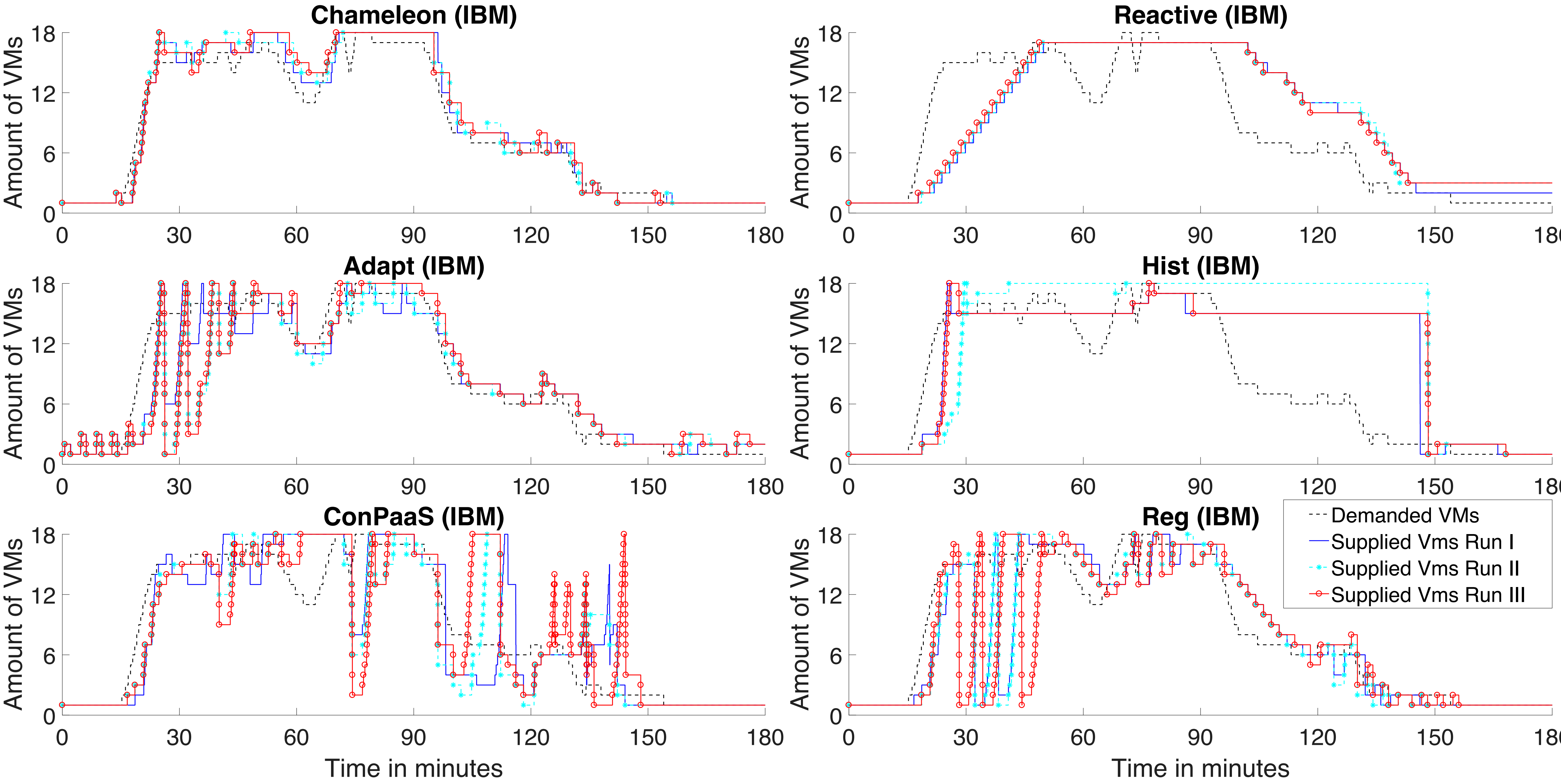
Thank you for your attention!



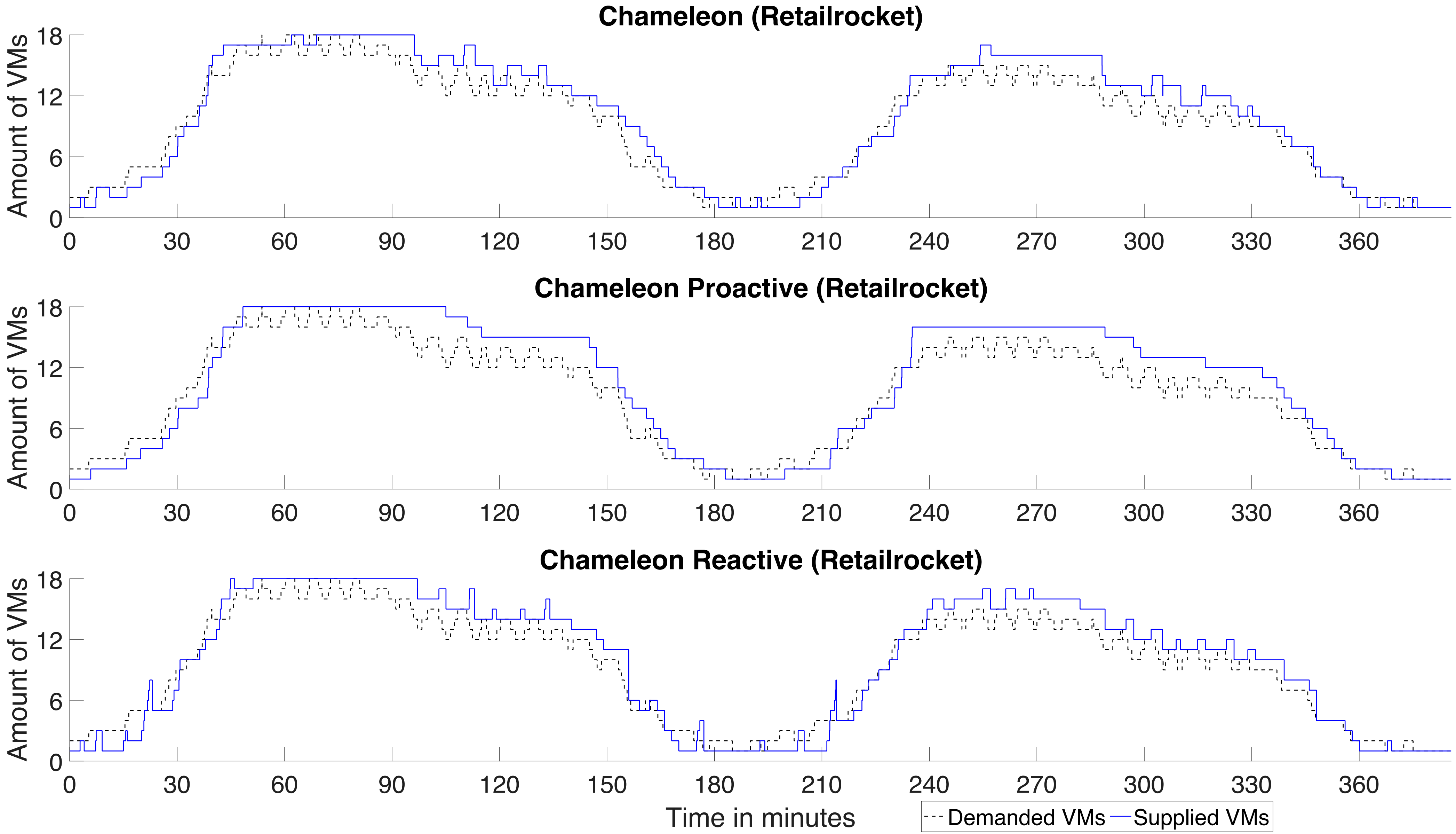
Experimental Evaluation: CS, Wiki



Experimental Evaluation: CS, IBM



Chameleon Components



Evaluation – Scaling Behavior