Software Performance Modeling for the Cloud

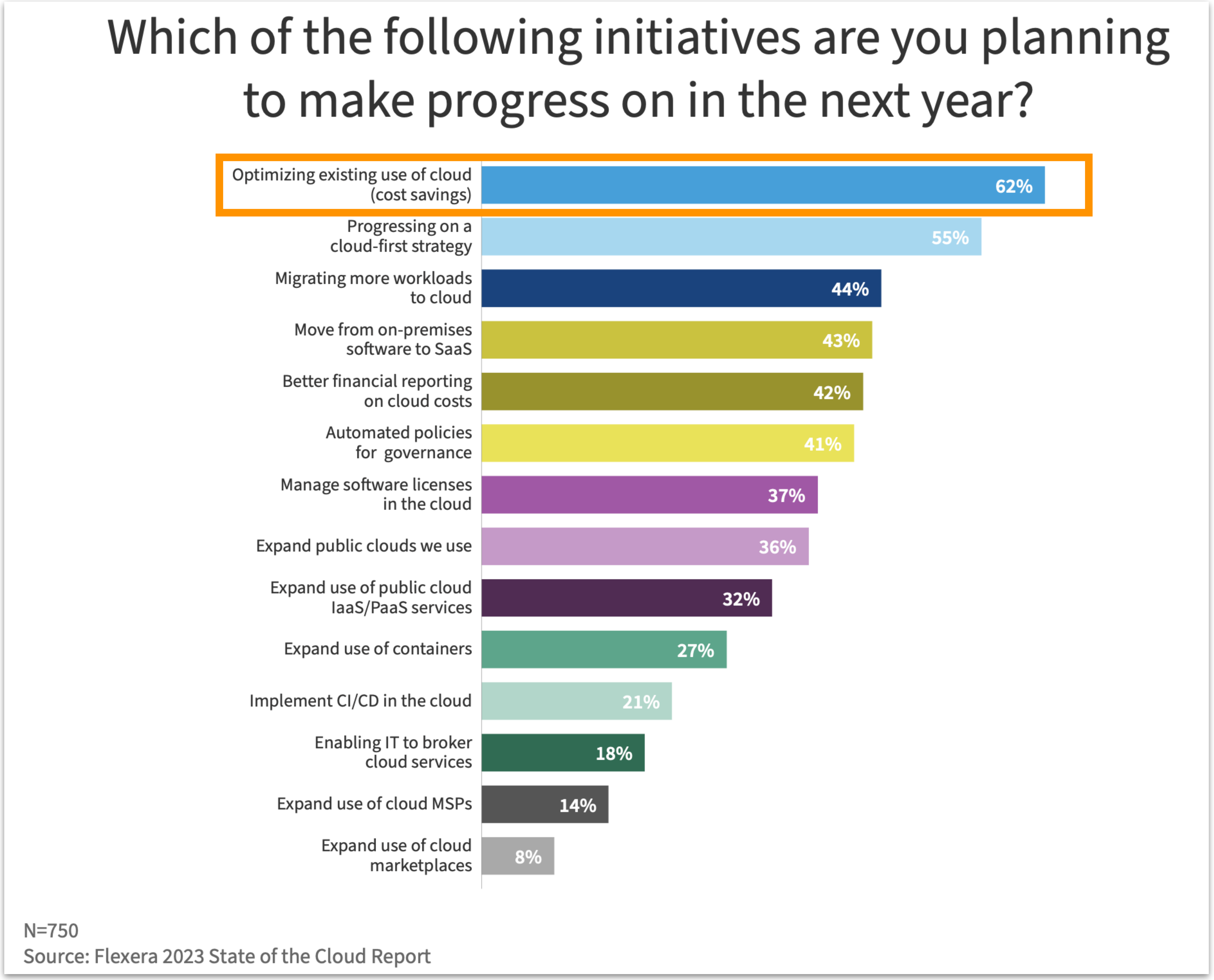
ASMTA/EPEW 2024

Mirco Tribastone

Venice, 14 June 2024

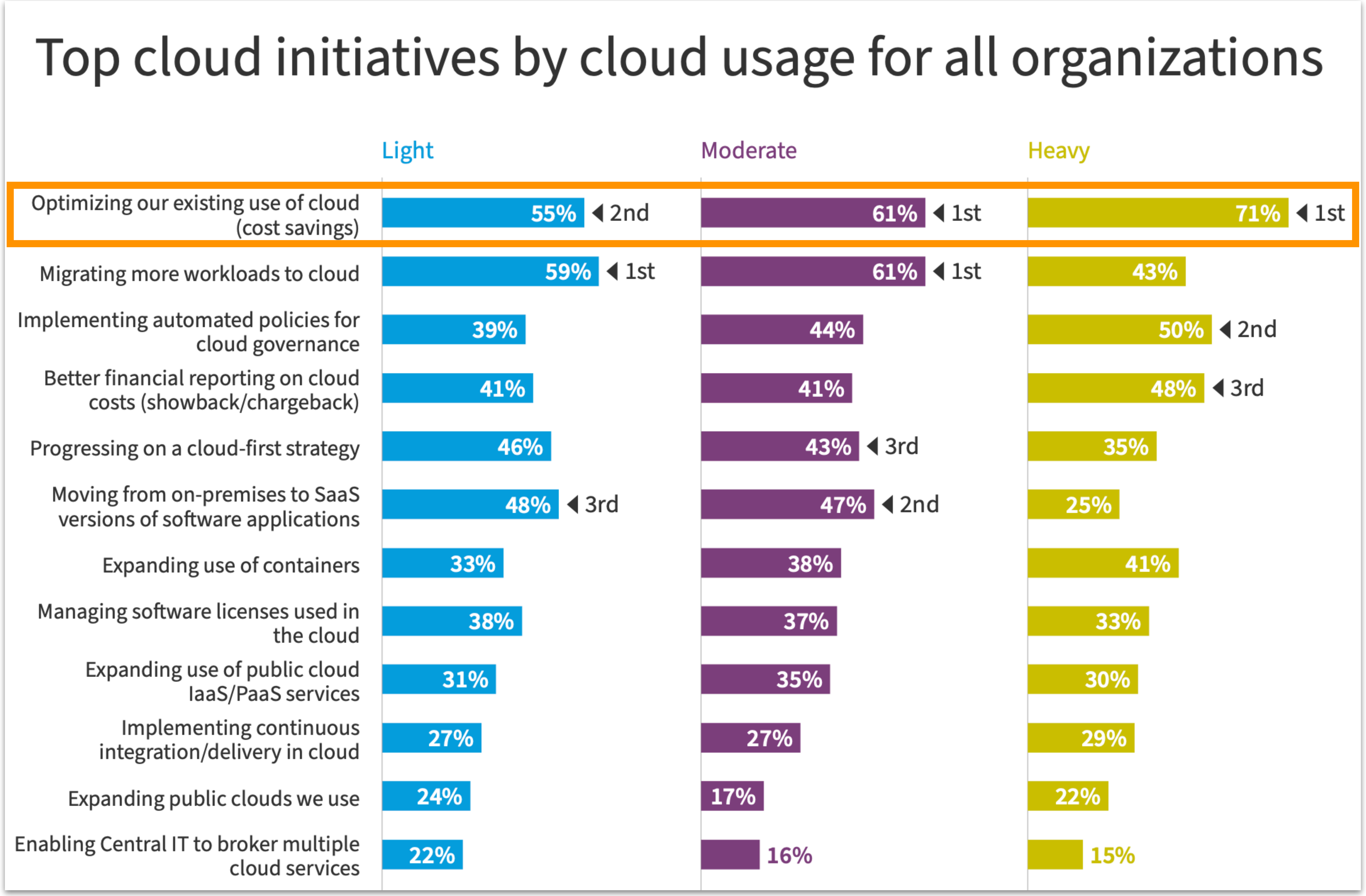
Motivation

Cloud Optimization: User’s View



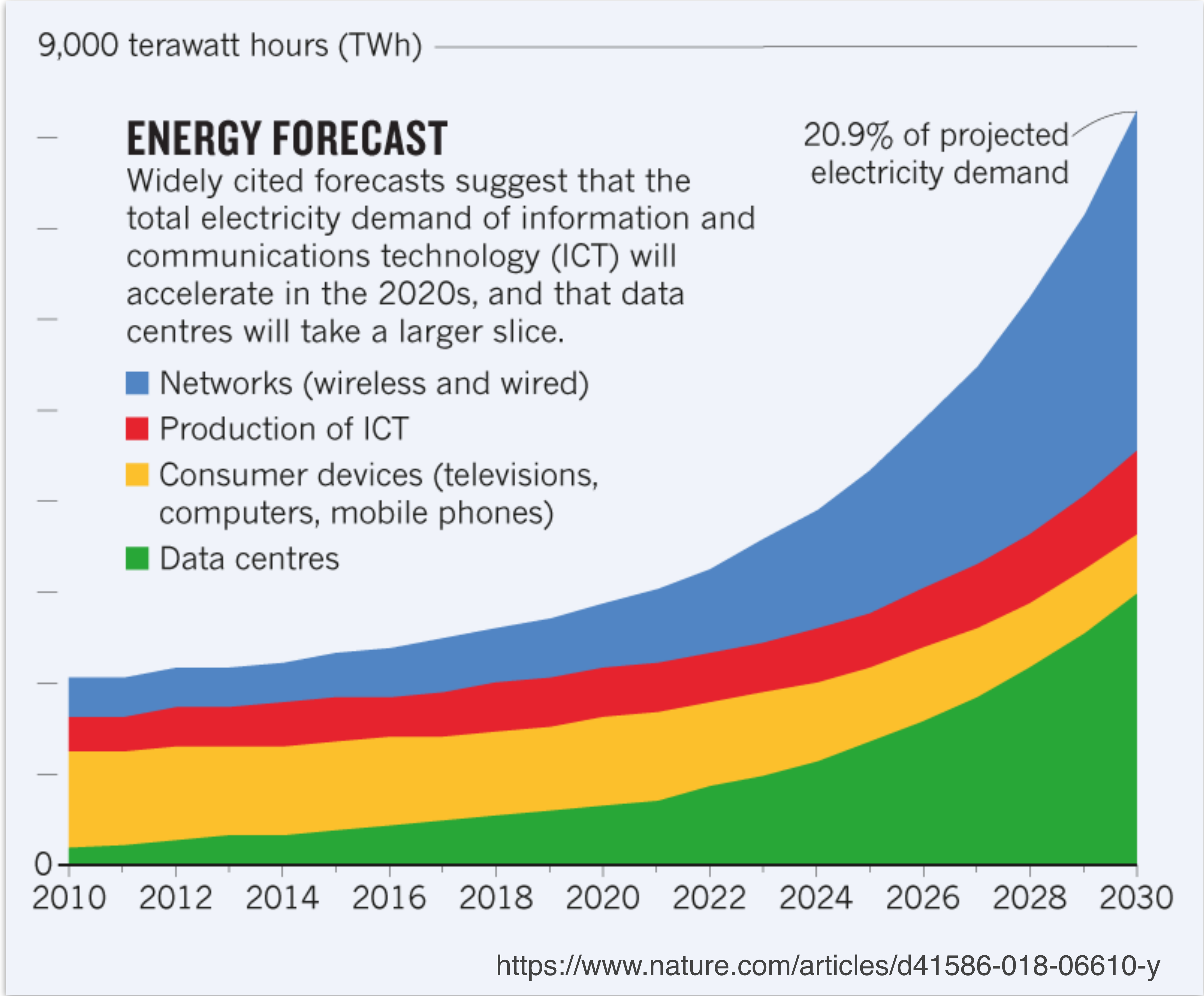
Motivation

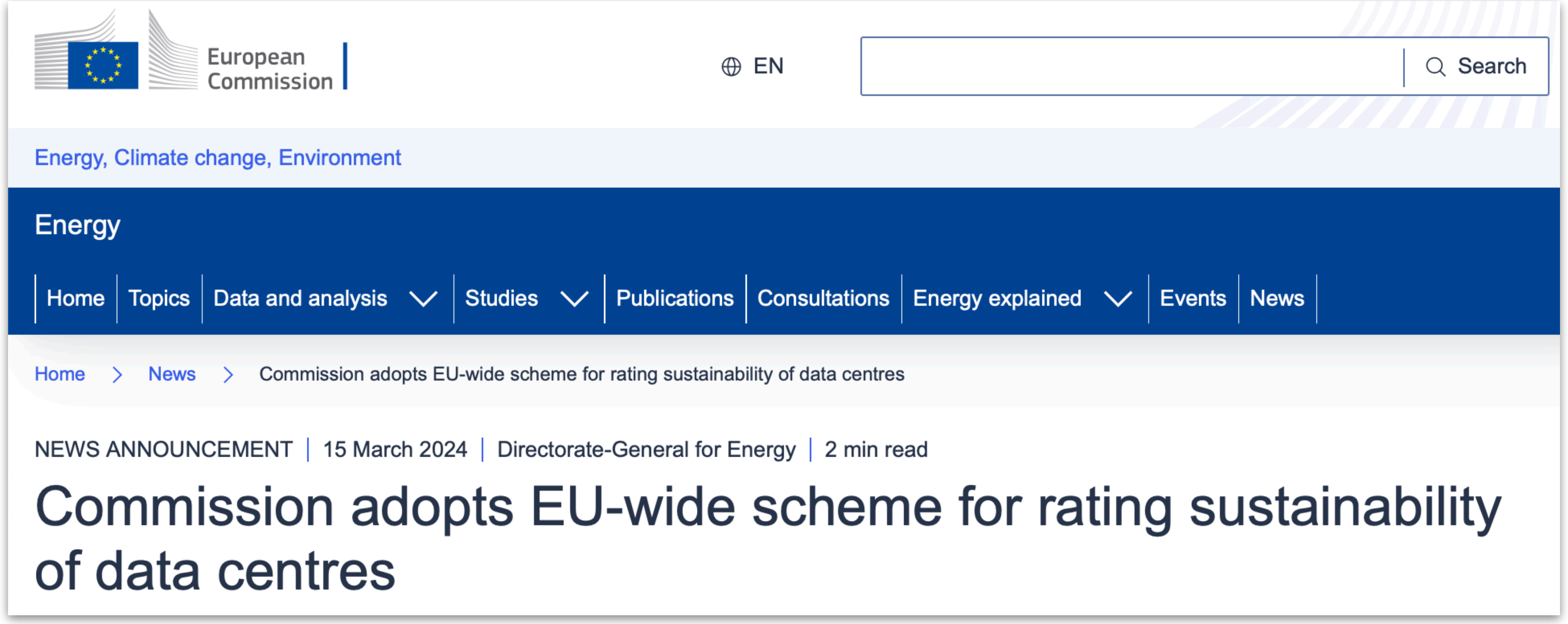
Cloud Optimization: User's View



Motivation

Cloud Optimization: Provider's View




European Commission

EN

Search

Energy, Climate change, Environment

Energy

Home | Topics | Data and analysis | Studies | Publications | Consultations | Energy explained | Events | News

Home > News > Commission adopts EU-wide scheme for rating sustainability of data centres

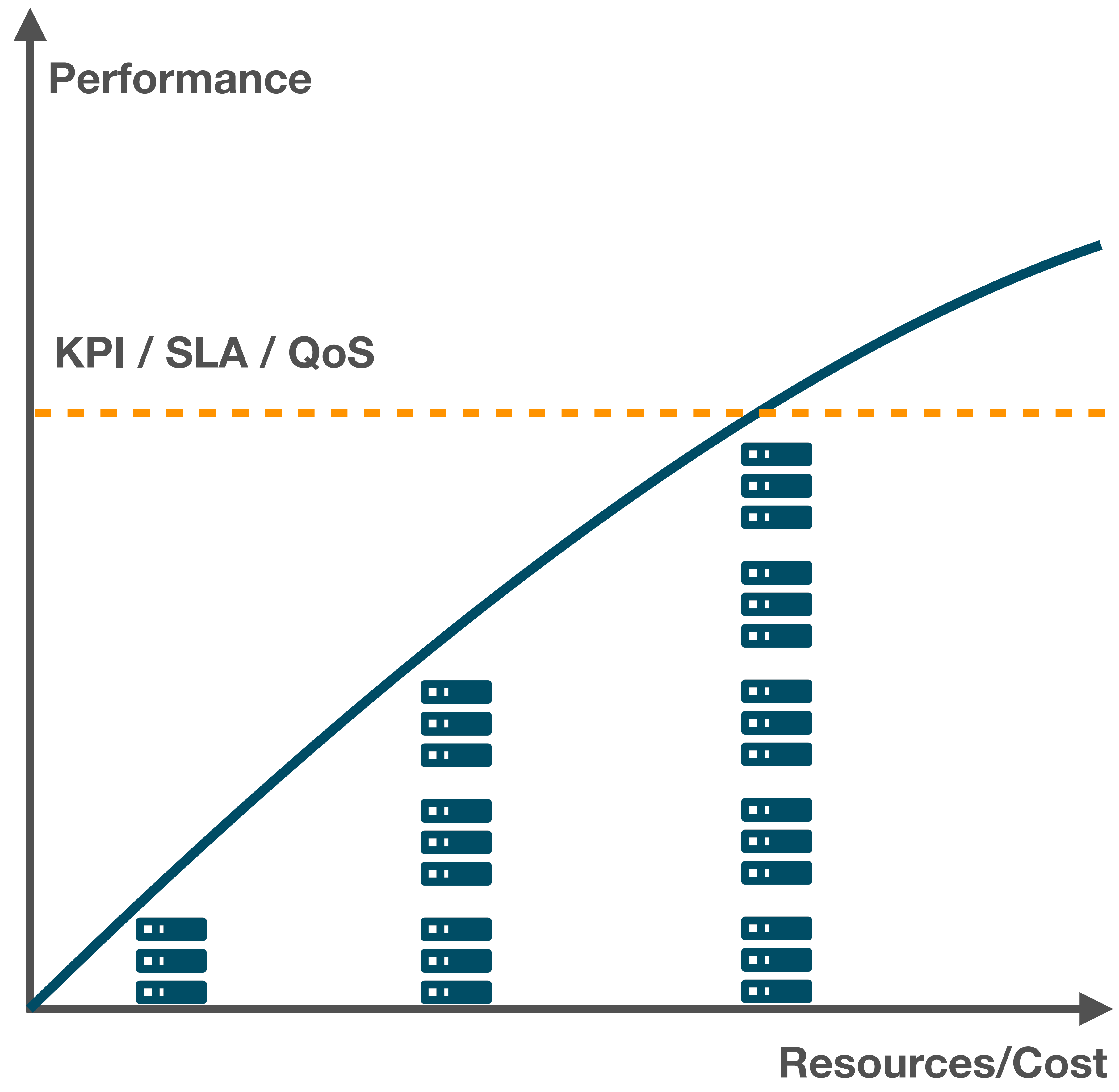
NEWS ANNOUNCEMENT | 15 March 2024 | Directorate-General for Energy | 2 min read

Commission adopts EU-wide scheme for rating sustainability of data centres

The Cost-Performance Trade-off
(Simplified)

- Adding more resources improves performance (up to a point)
- But it increases cost (pay-as-you-go) or energy consumption (on premise)
- Reducing resources decreases costs but also performance (up to a point)

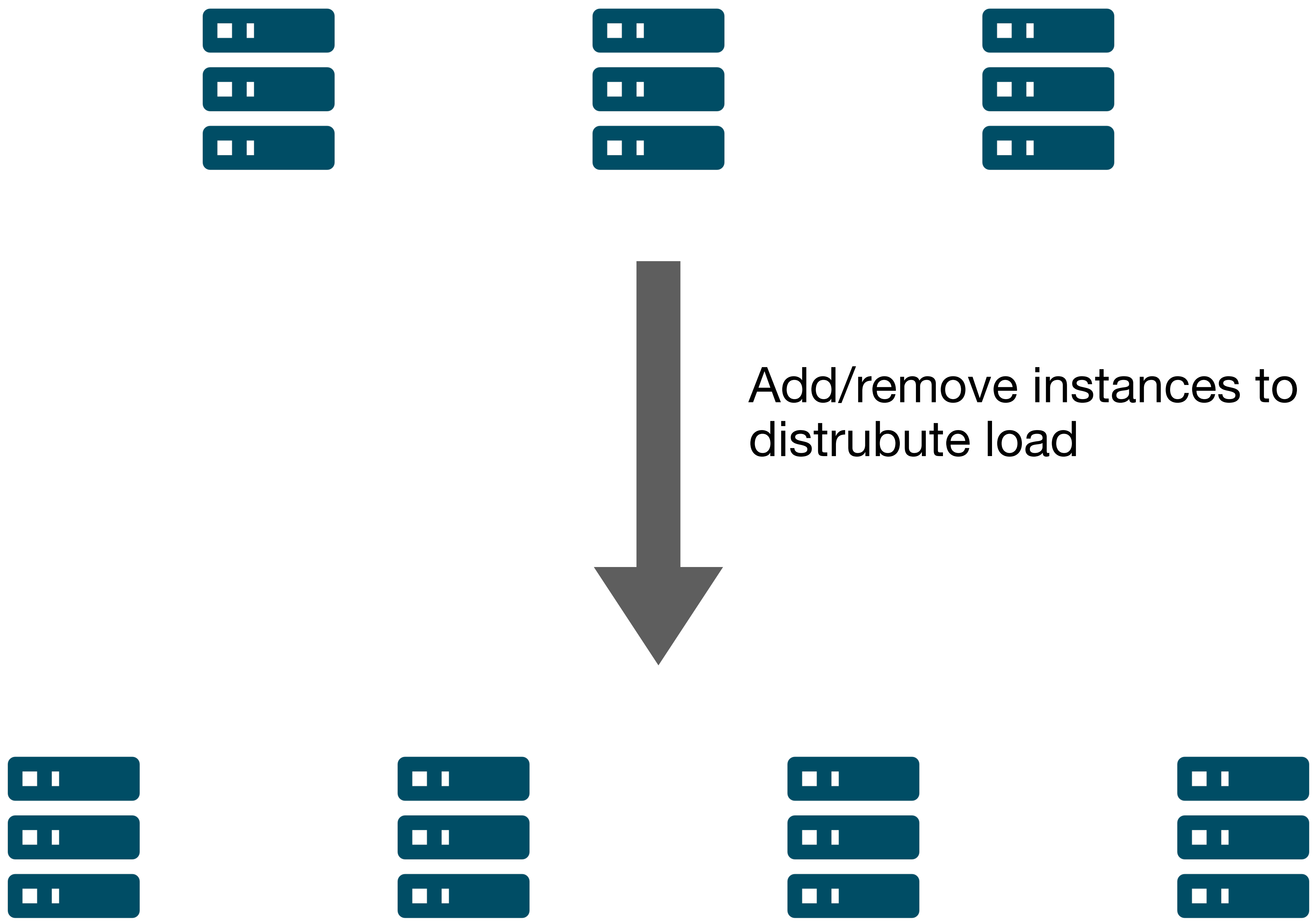
Objective: find the minimal amount of resources that meets desired key performance indicators



Elasticity in the Cloud

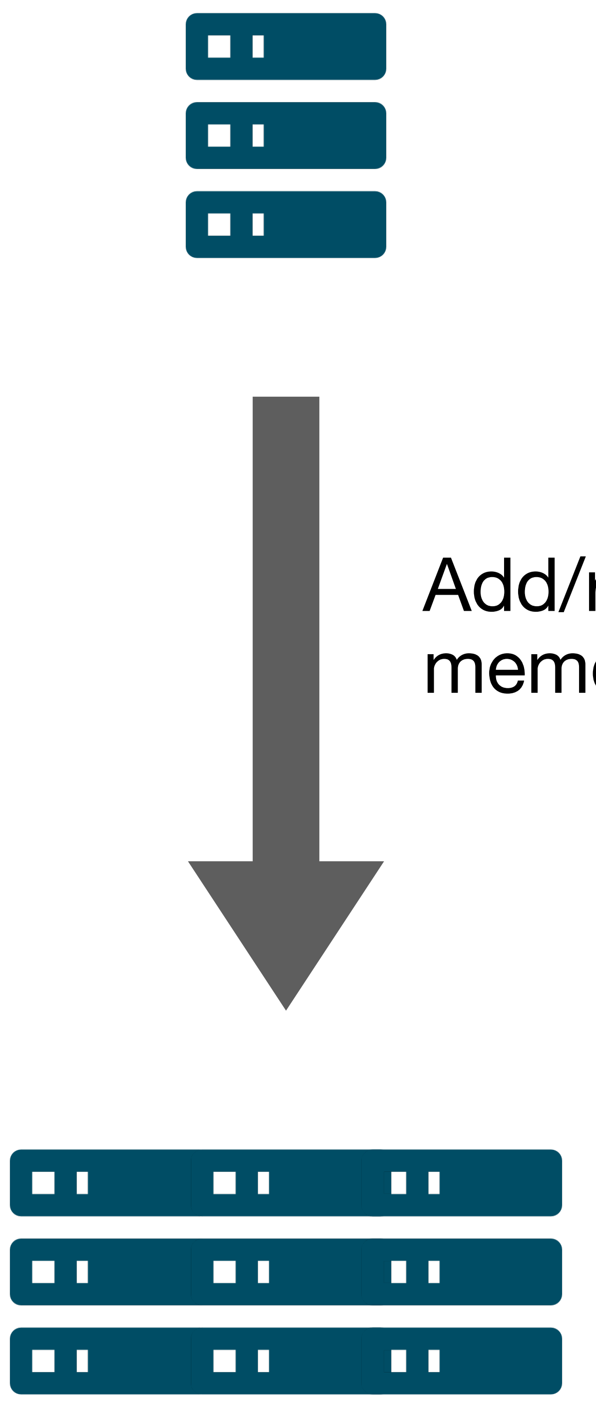
Ability to dynamically scale resources based on demand

Horizontal Scaling



Add/remove instances to
distrubute load

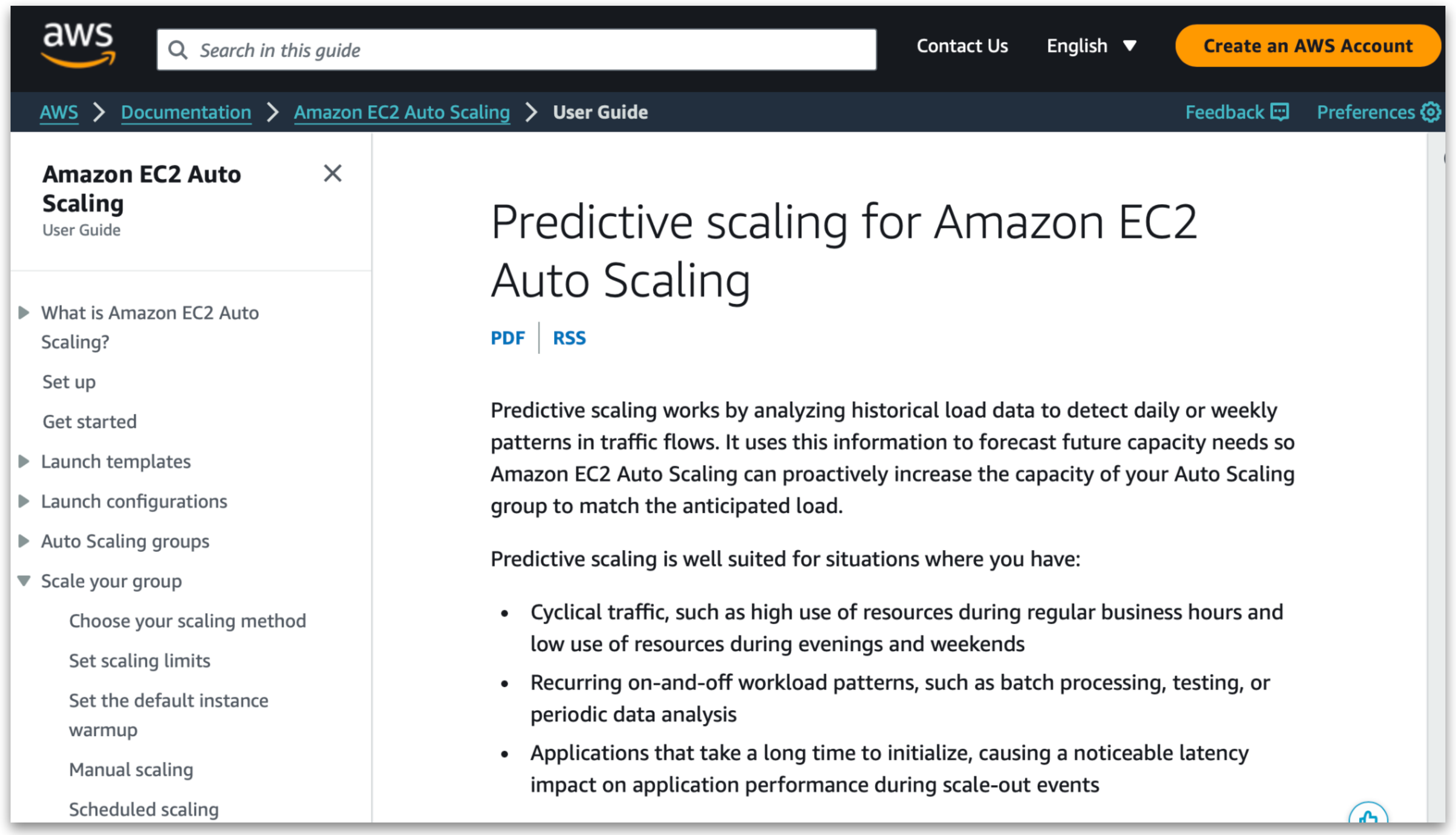
Vertical Scaling

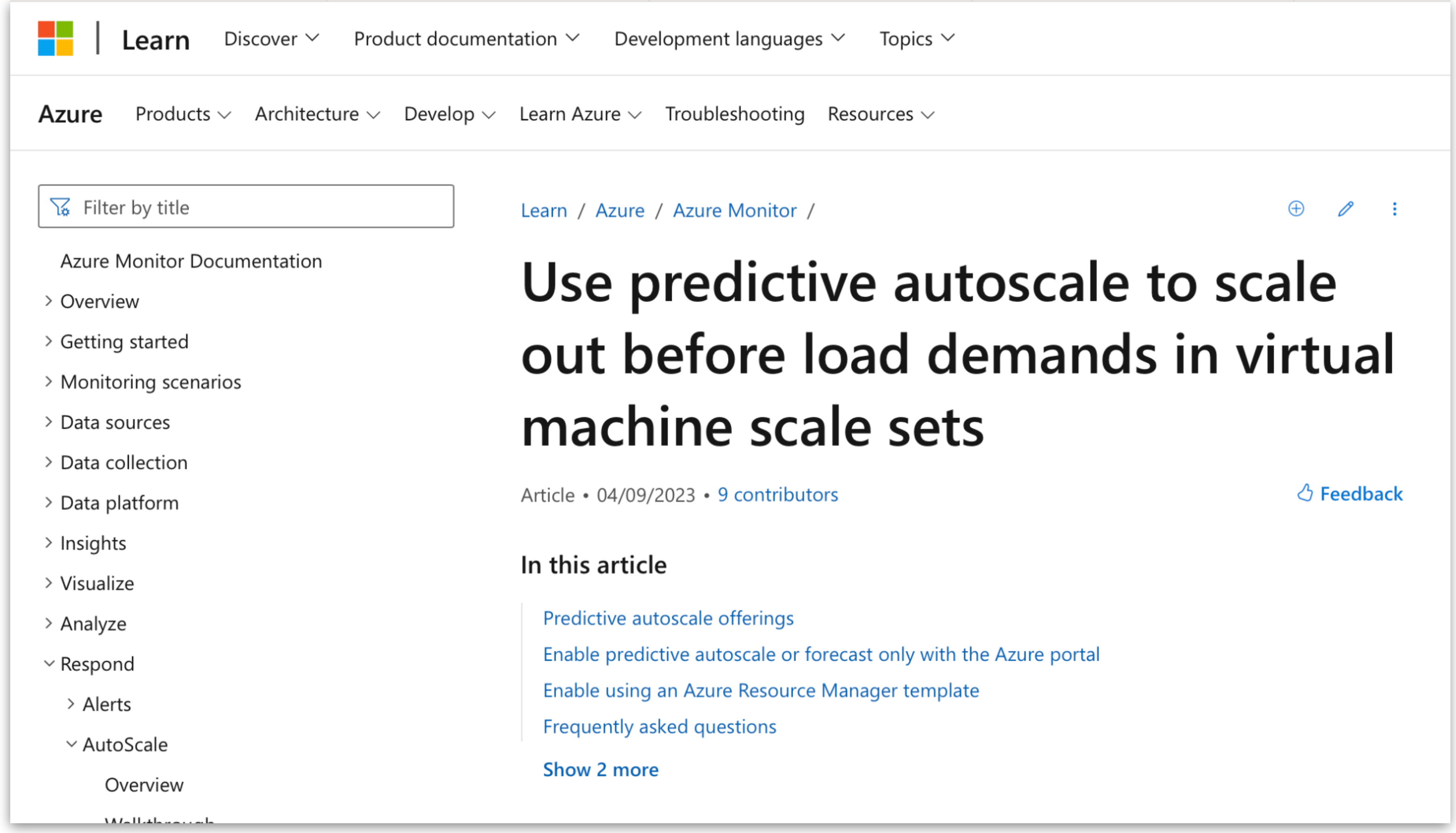


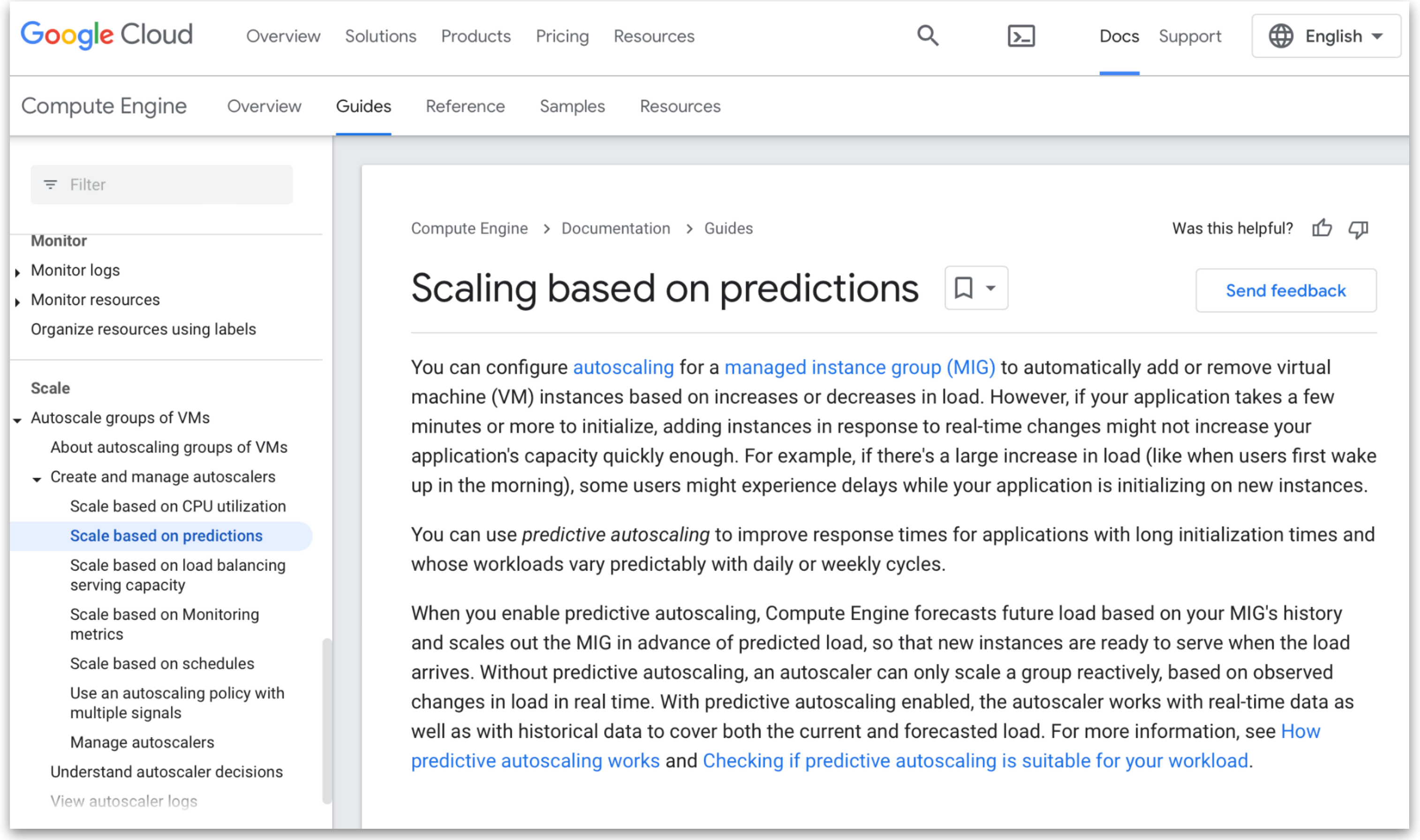
Add/remove more power (CPU,
memory, etc.) to existing instances

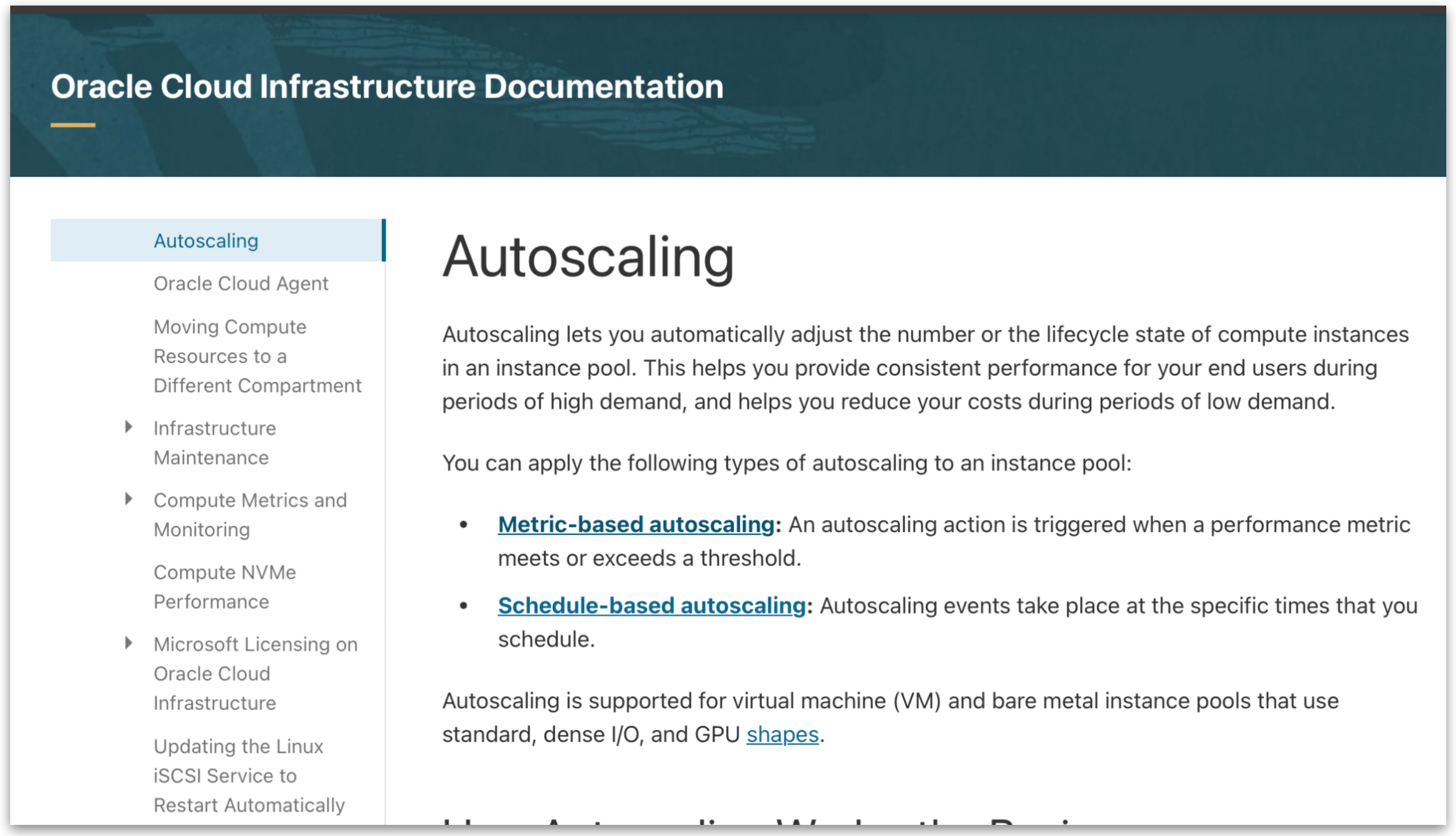
Autoscalers

Cloud techonology to achieve elasticity









Autoscalers

Limitations

Proper configuration	Requires careful configuration of thresholds, cooldown periods, and scaling policies to avoid unnecessary scaling actions
Latency	There may be some latency in scaling actions, which could impact performance during rapid spikes in demand
Complexity	Complex, multi-tiered applications, can add operational complexity

Autoscalers

State of the art in industry

		Target performance metric			
Autoscaler	**Predictive**	**Utilization**	**Throughput**	**Response time**	**Automatic rules**
AWS Tracking Scaling	x	✔	✔	✔	✔
AWS Predictive Scaling	✔	✔	✔	v	x
Azure Predictive Autoscale	✔	✔	x	x	x
GCP MIG Autoscaling	x	✔	x	x	x
GCP Predictive Autoscaling	✔	✔	x	x	x
Oracle Autoscale	x	✔	✔	x	x

Autoscalers

State of the art in academia

Black-box modeling

Regression
M. Wajahat, A. Karve, A. Kochut, and A. Gandhi, “MIscale: A machine learning based application-agnostic autoscaler,” Sustain. Comput.: Inform. Sys., vol. 22, pp. 287–299, 2019.

Reinforcement Learning
W. Iqbal, M. N. Dailey, and D. Carrera, “Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications,” IEEE Sys. J., vol. 10, no. 4, pp. 1435–1446, 2015.

Deep Learning
C. Meng, J. Tong, M. Pan, and Y. Yu, “HRA: An intelligent holistic resource autoscaling framework for multi-service applications,” in IEEE Int. Conf. Web Services, 2022, pp. 129–139.

Classification
H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, “FIRM: An intelligent fine-grained resource management framework for SLO-Oriented microservices,” in Proc. 14th USENIX Symp. Operating Sys. Des. Implementation, 2020.

Limitation: may be expensive (up to terabytes of training data reported)

Autoscalers

State of the art in academia

White-box modeling

M/M/G queues
V.Tadakamalla and D.A. Menascè, “Autonomic Elasticity Control for Multi-Server Queues Under Generic Workload Surges in Cloud Environments,” IEEE Transactions Cloud Computing, vol. 10, no. 2, pp. 984–995, 2022.

Analytic models
L. Funari, L. Petrucci, and A. Detti, “Storage-saving scheduling policies for clusters running containers,” IEEE Transactions Cloud Computing, 2021.

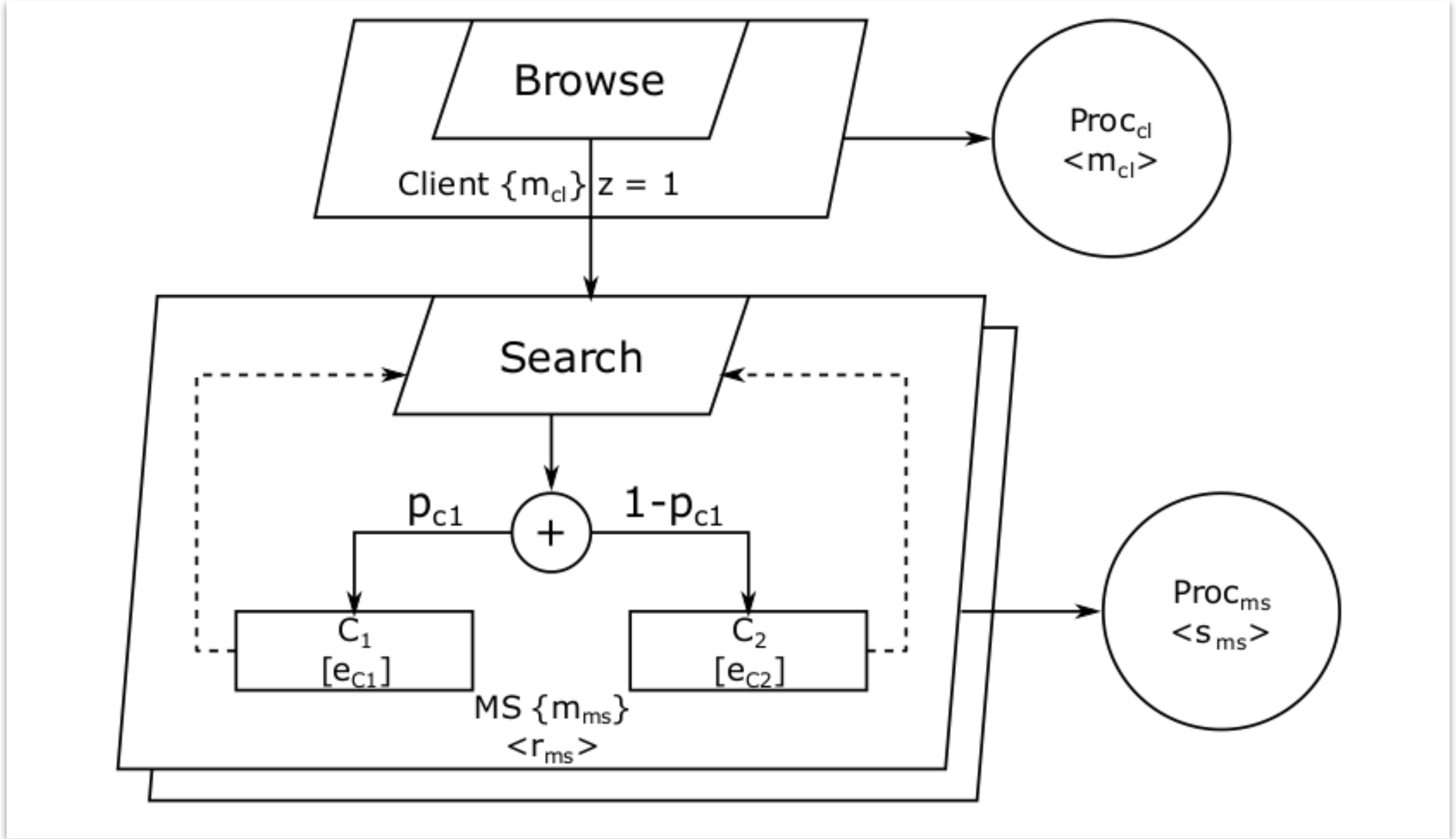
“Flat” queueing networks
J. Tong, M. Wei, M. Pan, and Y. Yu, “A holistic auto-scaling algorithm for multi-service applications based on balanced queuing network,” in IEEE Int. Conf. Web Services.

Layered queuing networks
A.U.Gias, G.Casale, and M.Woodside, “Atom: Model-driven autoscaling for microservices,” in IEEE 39th Int. Conf. Distrib. Comput. Syst., 2019.

And our work presented today…

Layered Queuing Networks (LQNs)

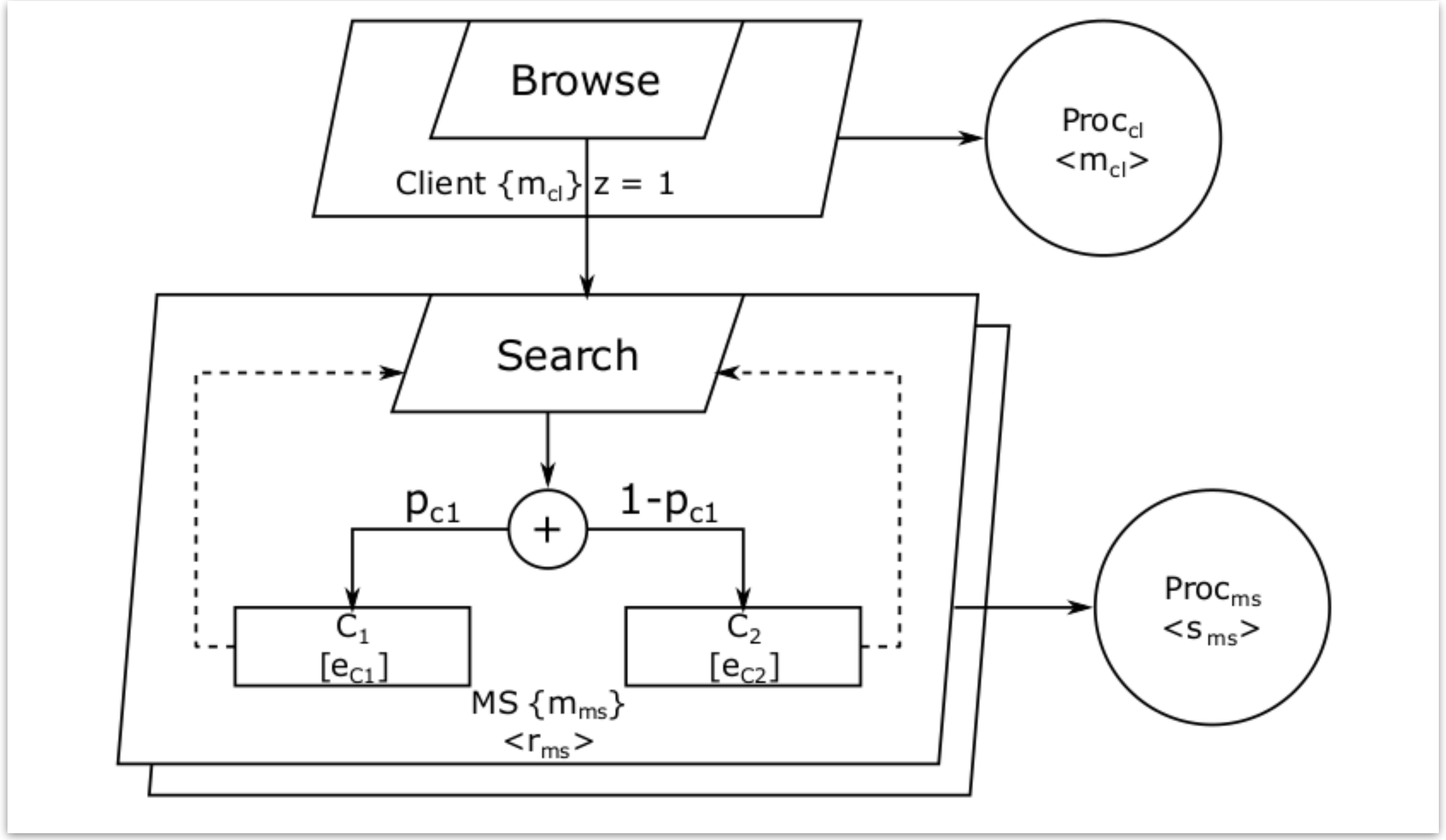
- Fork/join behaviour
- Synchronous and asynchronous calls
- Simultaneous resource possession (nesting/layering)
- Complex server logic



G. Franks, T. Al-Omari, M. Woodside, O. Das, and S. Derisavi, “Enhanced modeling and solution of layered queueing networks,” *IEEE Trans. Soft. Eng.*, vol. 35, no. 2, pp. 148–161, 2008.

Fluid LQN

$$\dot{x}_b = -T_b(x) + T_{c_1}(x) + T_{c_2}(x)$$
$$\dot{x}_{b \to s} = T_b(x) - T_{c_1}(x) - T_{c_2}(x)$$
$$\dot{x}_{ms} = T_b(x) - T_{ms}(x)$$
$$\dot{x}_{c_1} = p_{c_1} T_{ms}(x) - T_{c_1}(x)$$
$$\dot{x}_{c_2} = (1 - p_{c_1}) T_{ms}(x) - T_{c_2}(x)$$

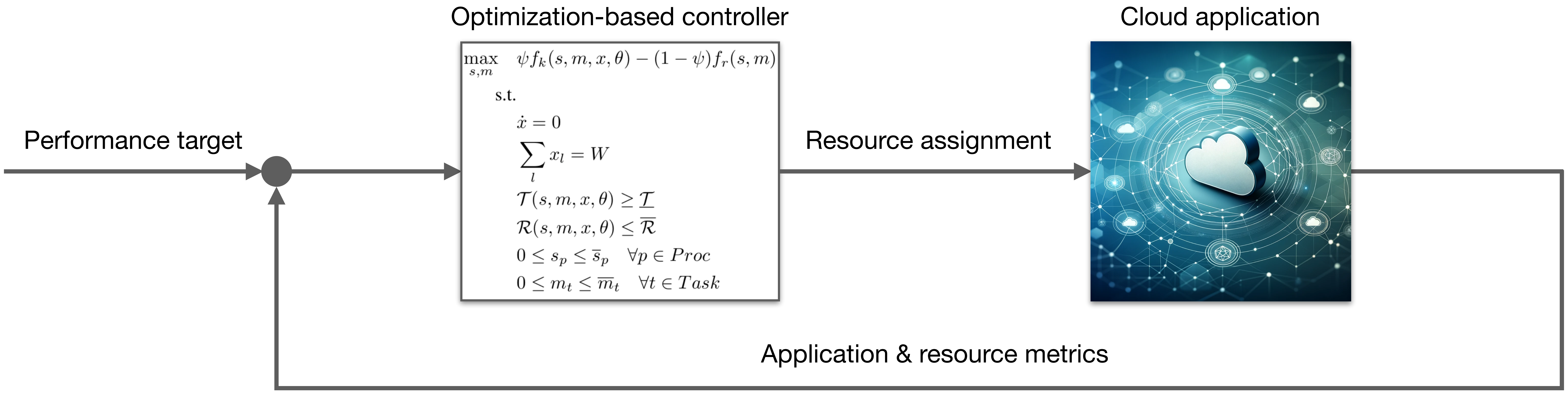


Avoids state space explosion, number of equations proportial to number of LQN components, asymptotically exact in the limit of large population sizes (appropriate for cloud-like scenarios)

M. Tribastone, “A fluid model for layered queueing networks,” IEEE Trans. Soft. Eng., vol. 39, no. 6, pp. 744–756, 2012

Waizmann, Tabea, and Mirco Tribastone. "DiffLQN: Differential Equation Analysis of Layered Queuing Networks." ACM/SPEC on International Conference on Performance Engineering. 2016.

Autoscaling via Feedback Control



Optimization-based Control

Emilio Incerto, Mirco Tribastone, Catia Trubiani. Software performance self-adaptation through efficient model predictive control. ASE 2017
Emilio Incerto, Mirco Tribastone, Catia Trubiani. Combined Vertical and Horizontal Autoscaling through Model Predictive Control. Euro-Par 2018
Emilio Incerto, Roberto Pizziol, Mirco Tribastone. μOpt: An Efficient Optimal Autoscaler for Microservice Applications. ACSOS 2023 **(best paper award)**

Learning Models

Emilio Incerto, Annalisa Napolitano, Mirco Tribastone. Statistical Learning of Markov Chains of Programs. MASCOTS 2020
Giulio Garbi, Emilio Incerto, Mirco Tribastone. Learning Queuing Networks by Recurrent Neural Networks. ICPE 2020
Giulio Garbi, Emilio Incerto, Mirco Tribastone. μP: A Development Framework for Predicting Performance of Microservices by Design. CLOUD 2023

Improving Fluid Approximation

Nicolas Gast, Luca Bortolussi, Mirco Tribastone. Size expansions of mean field approximation: Transient and steady-state analysis. PERFORMACE 2019
Francesca Randone, Luca Bortolussi, Mirco Tribastone. Refining Mean-field Approximations by Dynamic State Truncation. SIGMETRICS 2021
Francesca Randone, Luca Bortolussi, Mirco Tribastone. Jump Longer to Jump Less: Improving Dynamic Boundary Projection with h-Scaling. QEST 2022

µOpt: Optimal Autoscaler for Microservice Applications

Objective Function

Maximize performance while minimizing cost

Decision variables

Concurrency levels and CPU cores

Constraints

Steady-state condition of fluid model
Application load *W*
Throughput and response time threshold
Maximum concurrency and core allocation

Solution

By smooth approximation of nondifferentiable constraints

$$\max_{s,m} \quad \psi f_k(s,m,x,\theta) - (1-\psi) f_r(s,m)$$

$$\text{s.t.}$$

$$\dot{x} = 0$$

$$\sum_l x_l = W$$

$$\mathcal{T}(s,m,x,\theta) \geq \underline{\mathcal{T}}$$

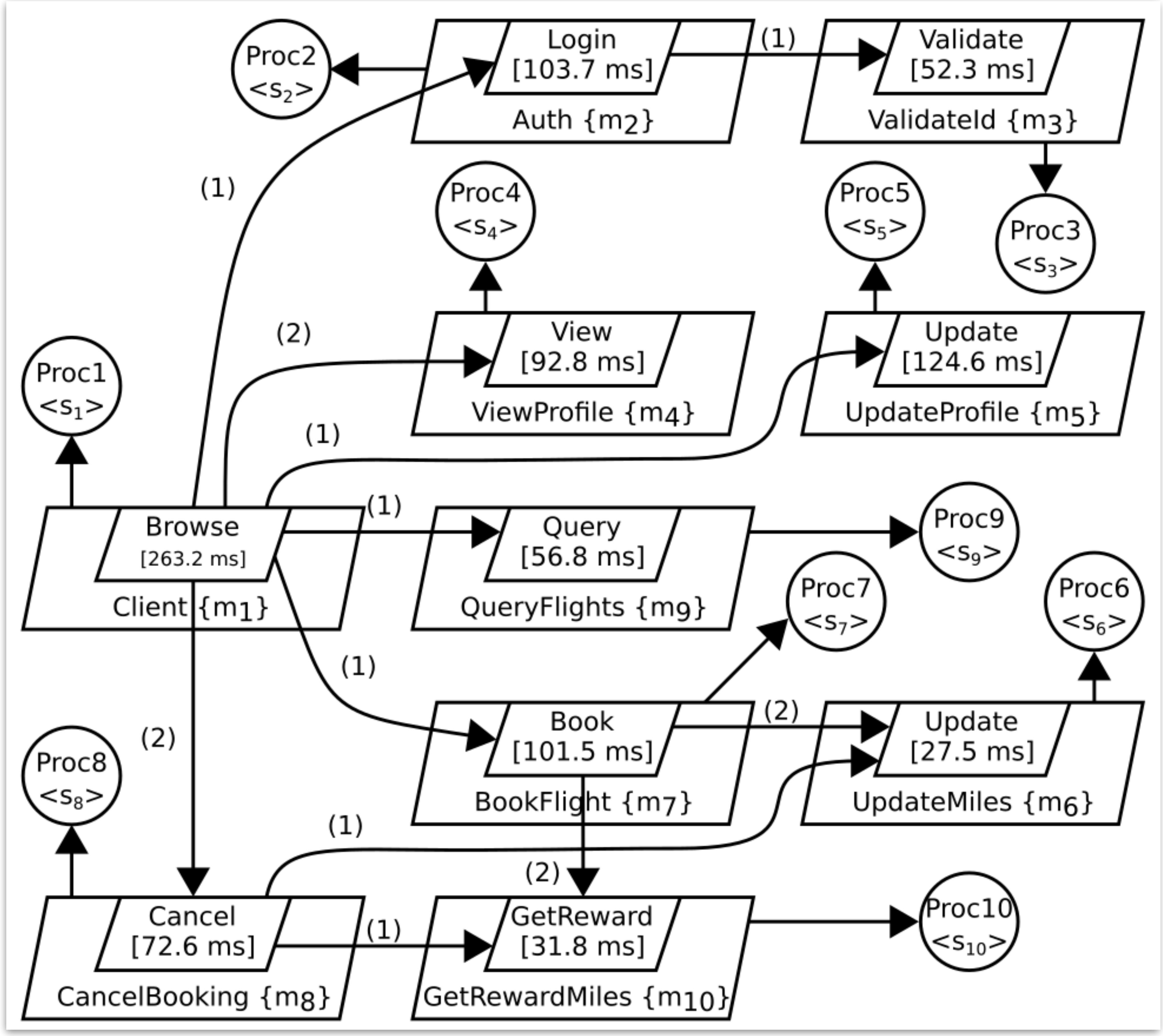
$$\mathcal{R}(s,m,x,\theta) \leq \overline{\mathcal{R}}$$

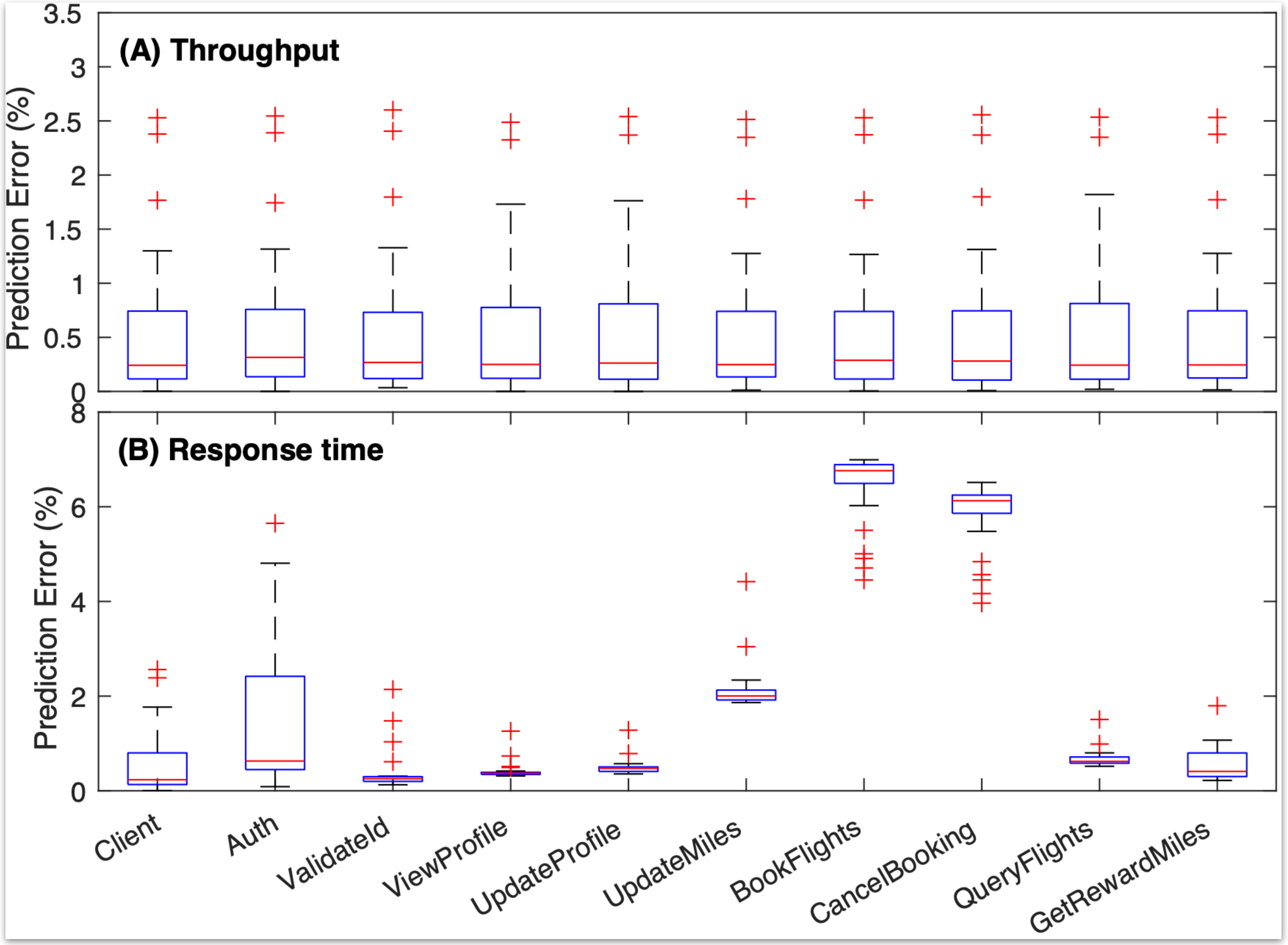
$$0 \leq s_p \leq \overline{s}_p \quad \forall p \in Proc$$

$$0 \leq m_t \leq \overline{m}_t \quad \forall t \in Task$$

µOpt: ACME Air Case Study

Open-loop Validation





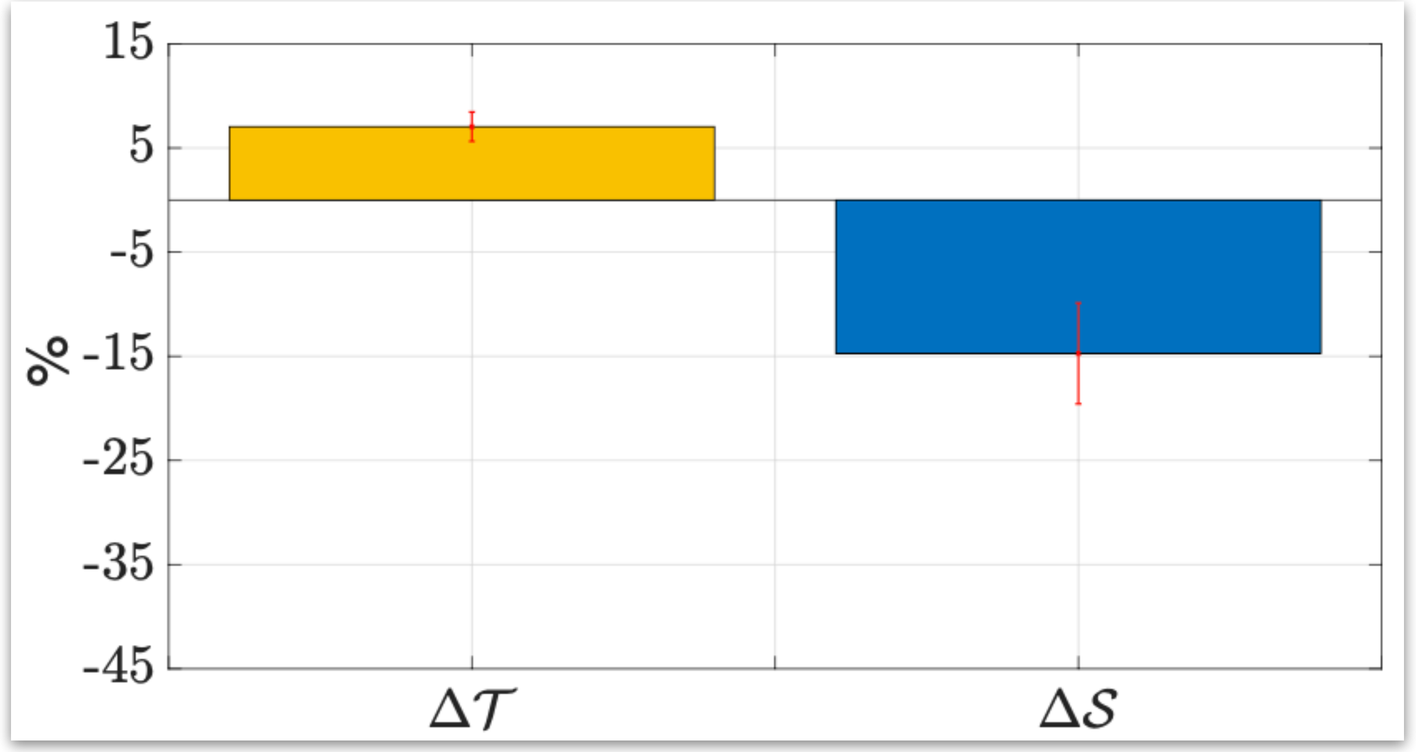
µOpt: ACME Air Case Study

Closed-loop Validation

Comparison with ATOM, an LQN-based autoscaler using black-box optimization with genetic algoritms

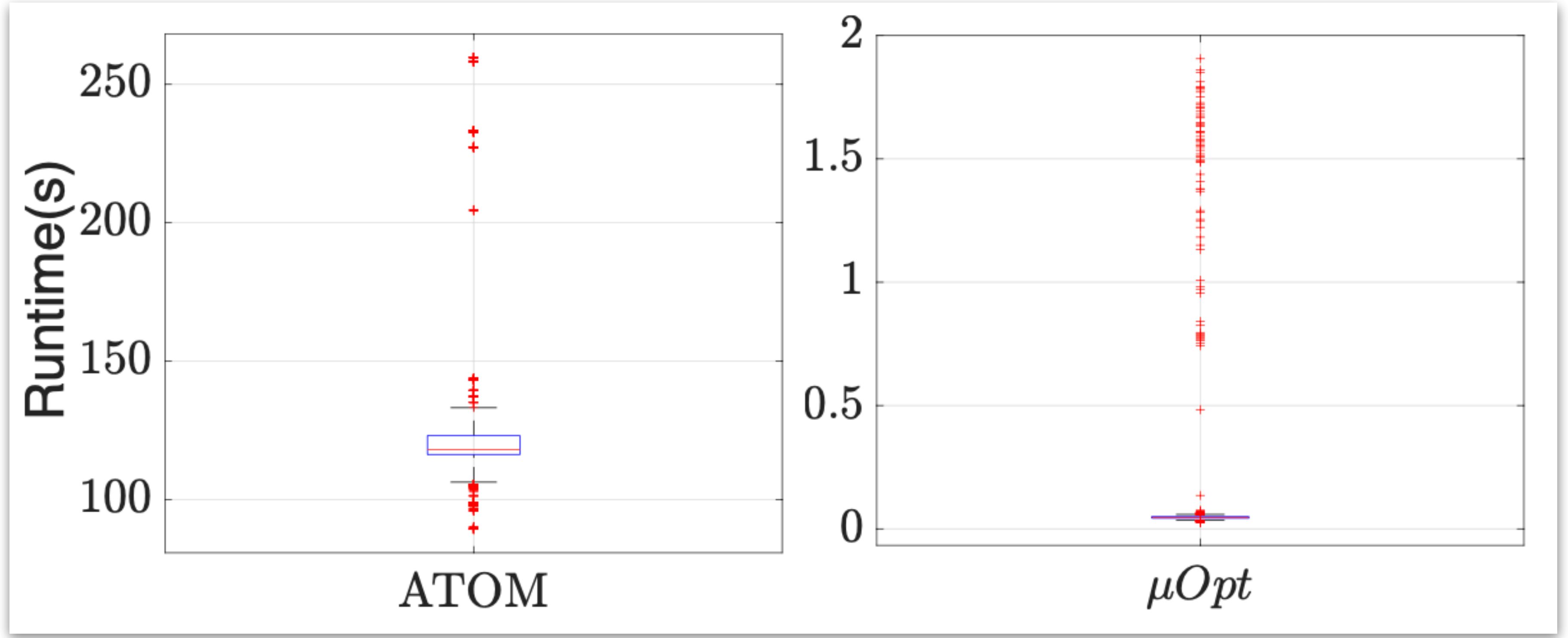
Evaluation on (re-scaled) Twitter trace

U.Gias, G.Casale, and M.Woodside, “ATOM: Model-driven autoscaling for microservices,” in 39th Int. Conf. Distrib. Comput. Syst., 2019

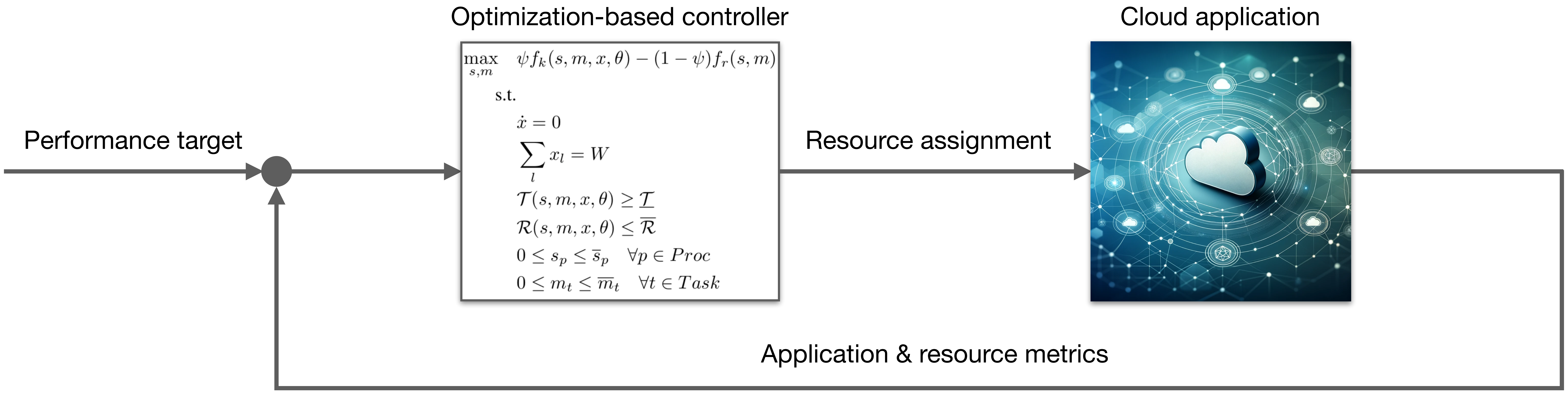


Throughput improvement

Core savings



Autoscaling via Feedback Control



Optimization-based Control

Emilio Incerto, Mirco Tribastone, Catia Trubiani. Software performance self-adaptation through efficient model predictive control. ASE 2017
Emilio Incerto, Mirco Tribastone, Catia Trubiani. Combined Vertical and Horizontal Autoscaling through Model Predictive Control. Euro-Par 2018
Emilio Incerto, Roberto Pizziol, Mirco Tribastone. μOpt: An Efficient Optimal Autoscaler for Microservice Applications. ACSOS 2023 **(best paper award)**

Learning Models

Emilio Incerto, Annalisa Napolitano, Mirco Tribastone. Statistical Learning of Markov Chains of Programs. MASCOTS 2020
Giulio Garbi, Emilio Incerto, Mirco Tribastone. Learning Queuing Networks by Recurrent Neural Networks. ICPE 2020
Giulio Garbi, Emilio Incerto, Mirco Tribastone. μP: A Development Framework for Predicting Performance of Microservices by Design. CLOUD 2023

Improving Fluid Approximation

Nicolas Gast, Luca Bortolussi, Mirco Tribastone. Size expansions of mean field approximation: Transient and steady-state analysis. PERFORMACE 2019
Francesca Randone, Luca Bortolussi, Mirco Tribastone. Refining Mean-field Approximations by Dynamic State Truncation. SIGMETRICS 2021
Francesca Randone, Luca Bortolussi, Mirco Tribastone. Jump Longer to Jump Less: Improving Dynamic Boundary Projection with h-Scaling. QEST 2022

µP: Predicting Performance of Microservices by Design

Motivation

Performance Engineering for Microservices: Research Challenges and Directions

Robert Heinrich,[1] André van Hoorn,[2] Holger Knoche,[3] Fei Li,[4]
Lucy Ellen Lwakatare,[5] Claus Pahl,[6] Stefan Schulte,[7] Johannes Wettinger[2]

[1] Karlsruhe Institute of Technology, Germany
[2] University of Stuttgart, Germany
[3] Kiel University, Germany
[4] Siemens AG, Austria
[5] University of Oulu, Finland
[6] Free University of Bozen-Bolzano, Italy
[7] TU Wien, Austria

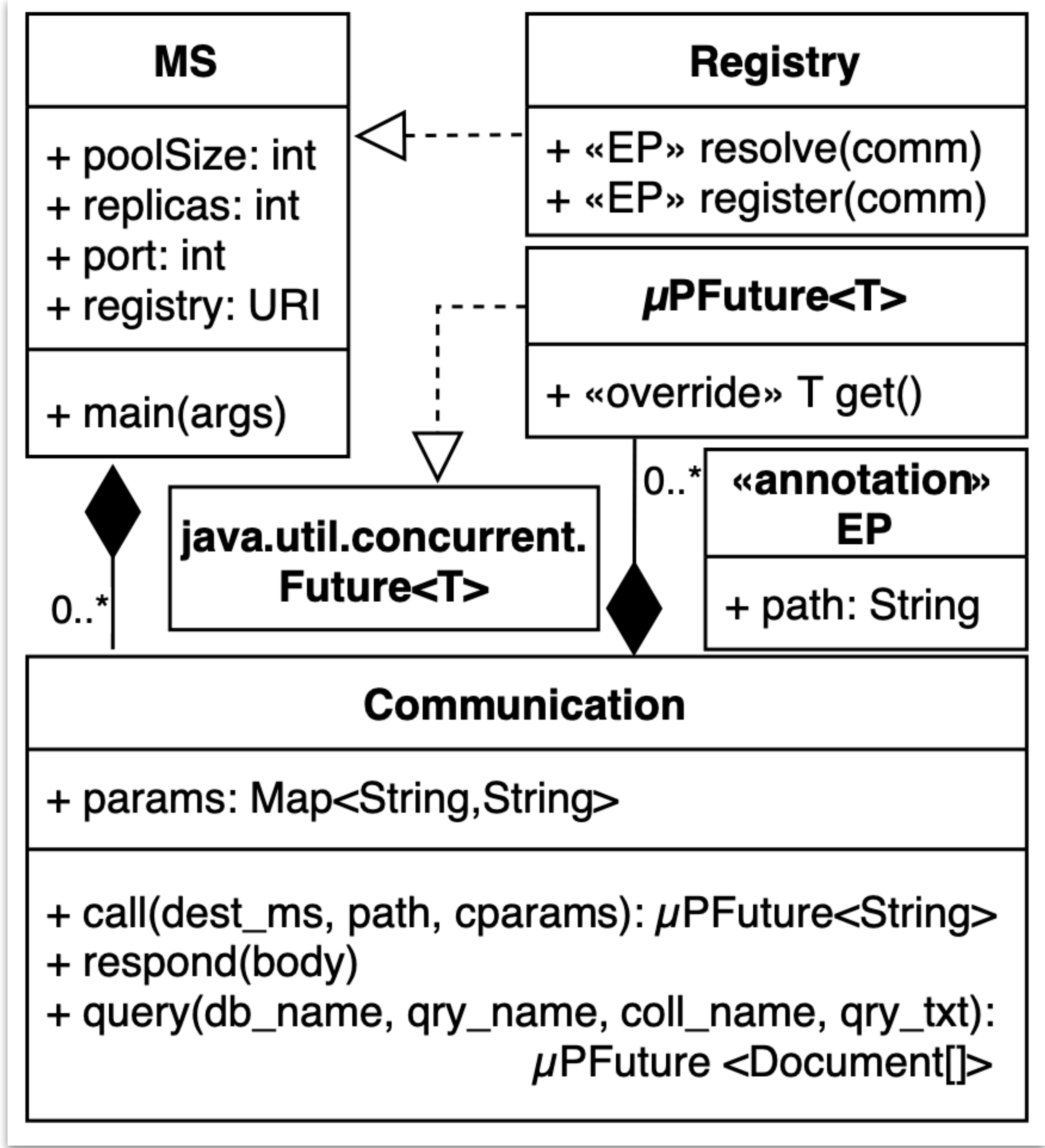
Thus, four key challenges emerge for performance modeling for microservices:

- Adopting performance modeling to shifted use cases
- Finding appropriate modeling abstractions
- Automated extraction of performance models
- Learning of infrastructure behavior and integration into performance models

This is an instance of a more general problem: how to learn appropriate abstractions of software systems for performance engineering models

µP: Predicting Performance of Microservices by Design

Approach

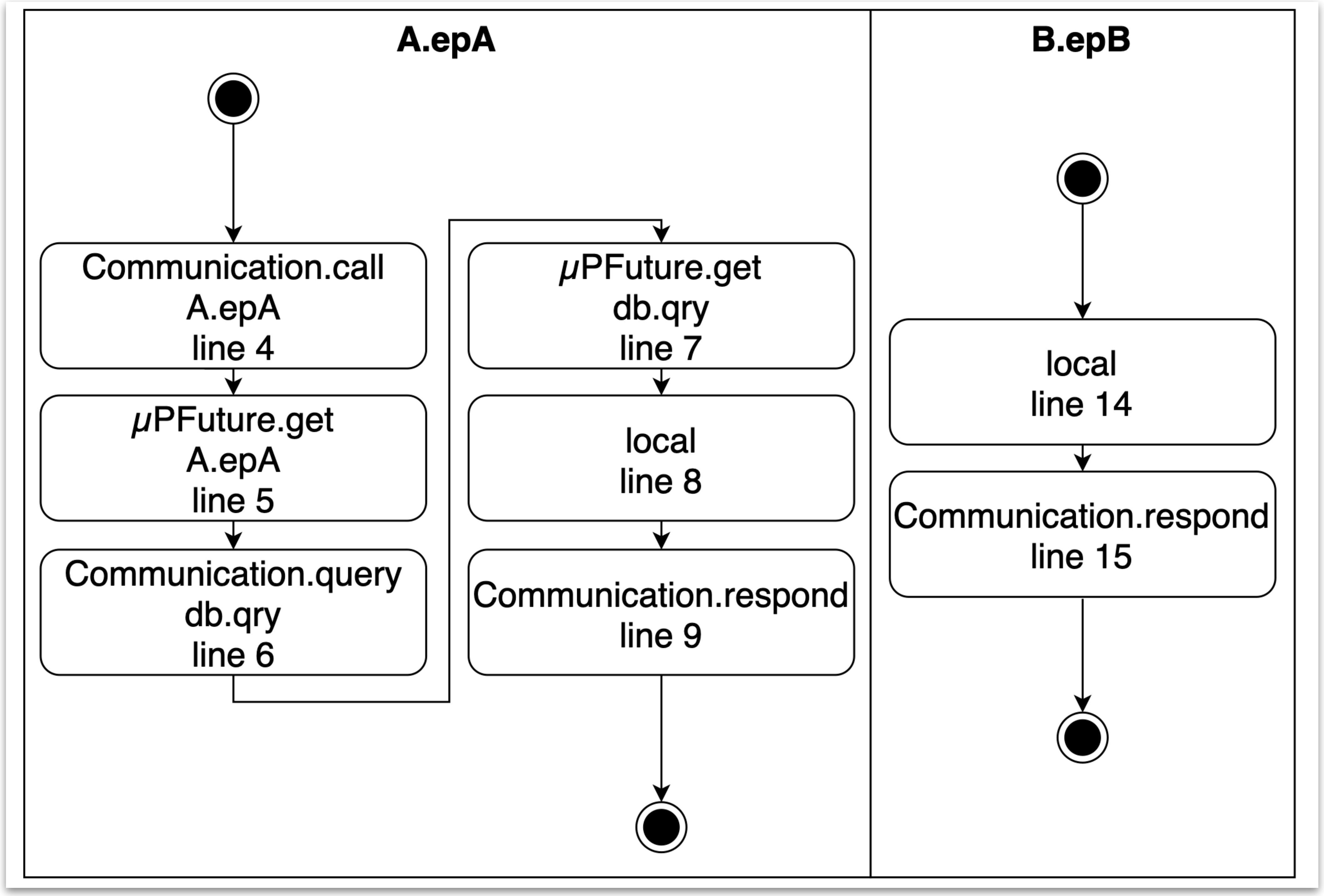


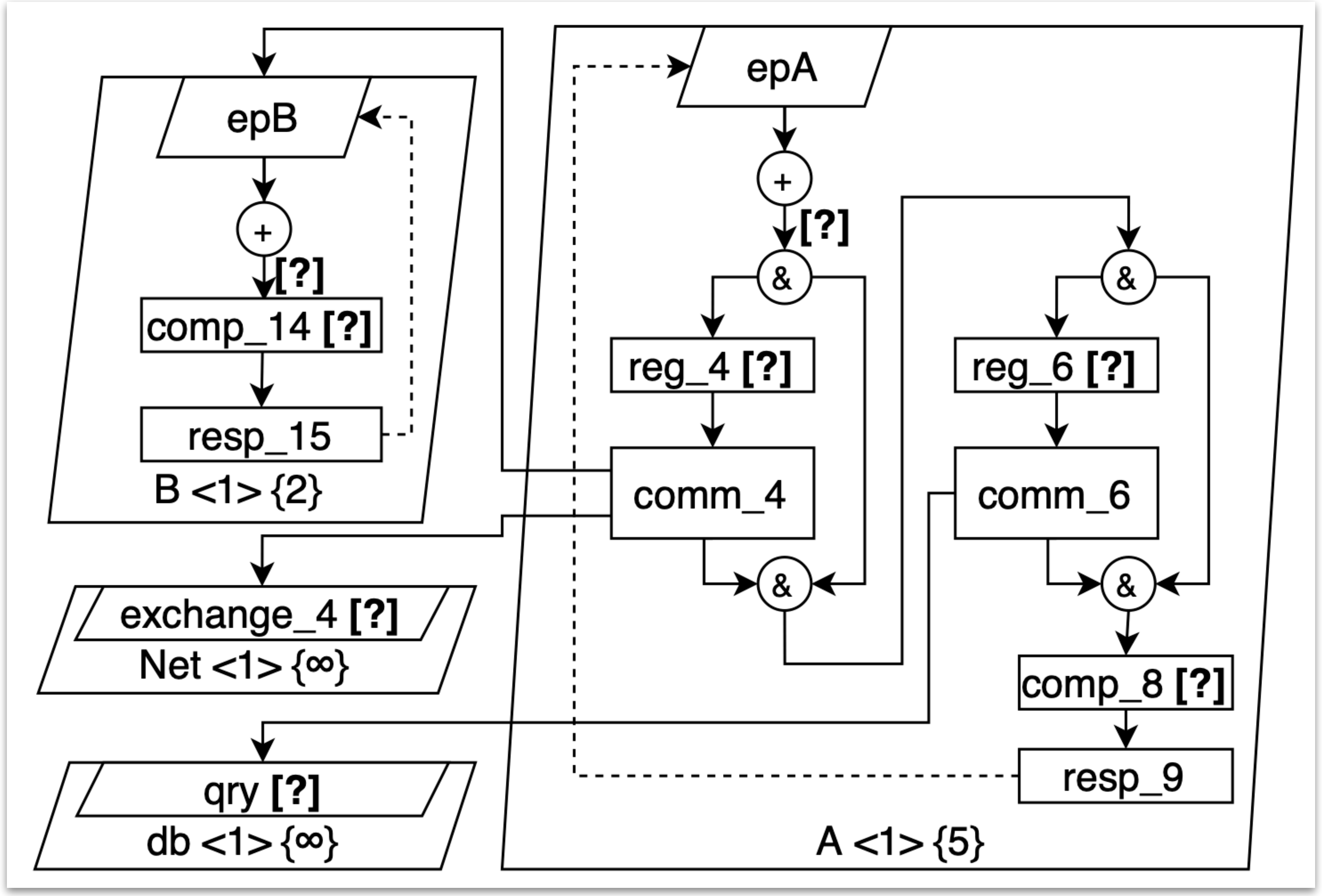

```
class A extends MS {
  poolSize = 5; replicas = 1;
  @EP("/epA/") public void epA(comm){
    bfuture = comm.call("B", "/epB/", comm.params);
    bstr = bfuture.get();
    dq = comm.query("db", "qry", "tr",
         "{\"_id\":\""+comm.params["word"]+"\"}");
    qstr = dq.get();
    ans = bstr + "=>" + qstr[0];
    comm.respond(ans);}
}

class B extends MS {
  poolSize = 2; replicas = 1;
  @EP("/epB/") public void epB(comm){
    hw = "hello" + "world";
    comm.respond(hw);}
}
```

μP: Predicting Performance of Microservices by Design

Static Analysis

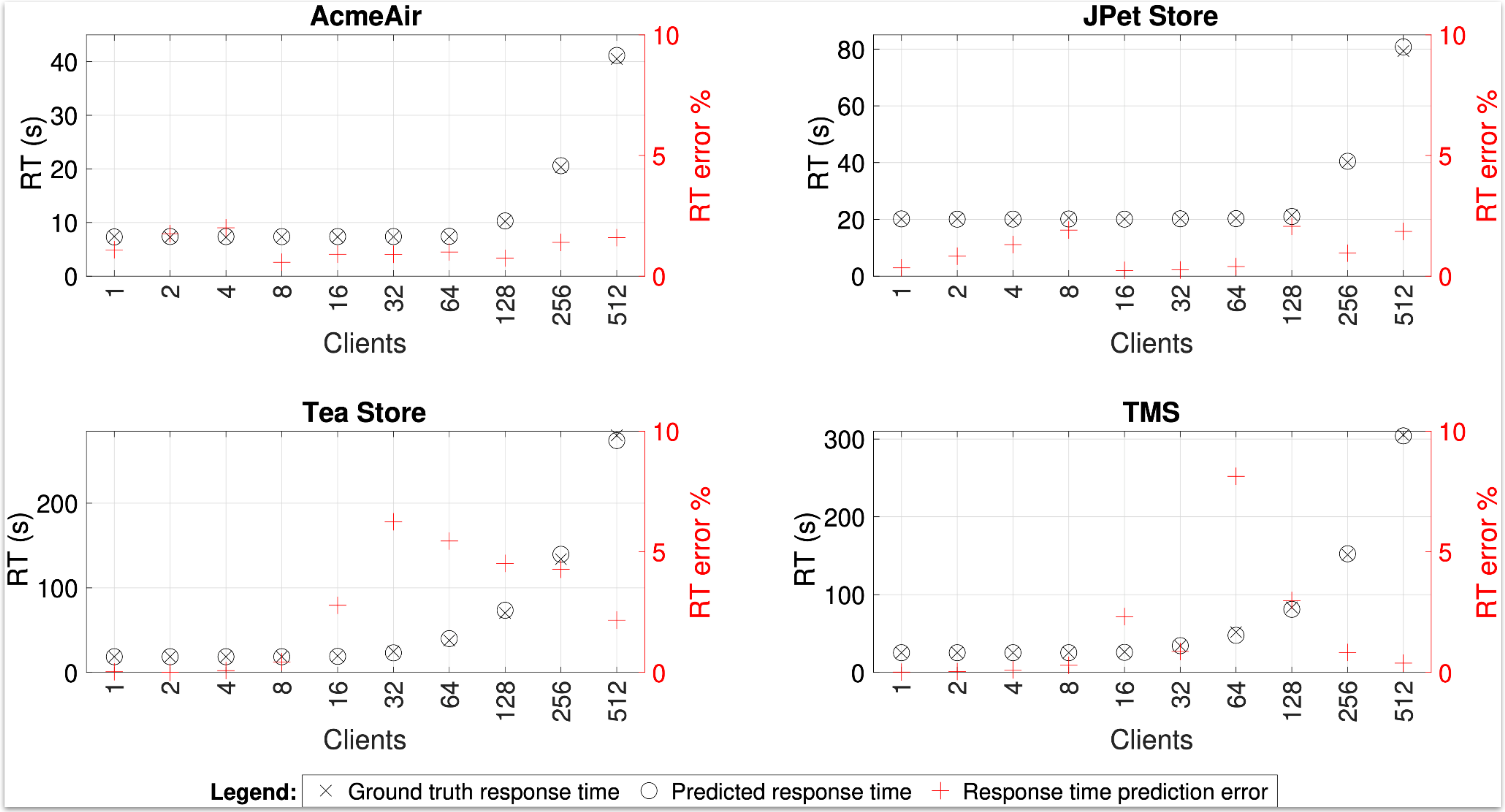




From code annotation to LQN model structure with “holes” (parameter values) to be filled by dynamic analysis with single-user explorative runs

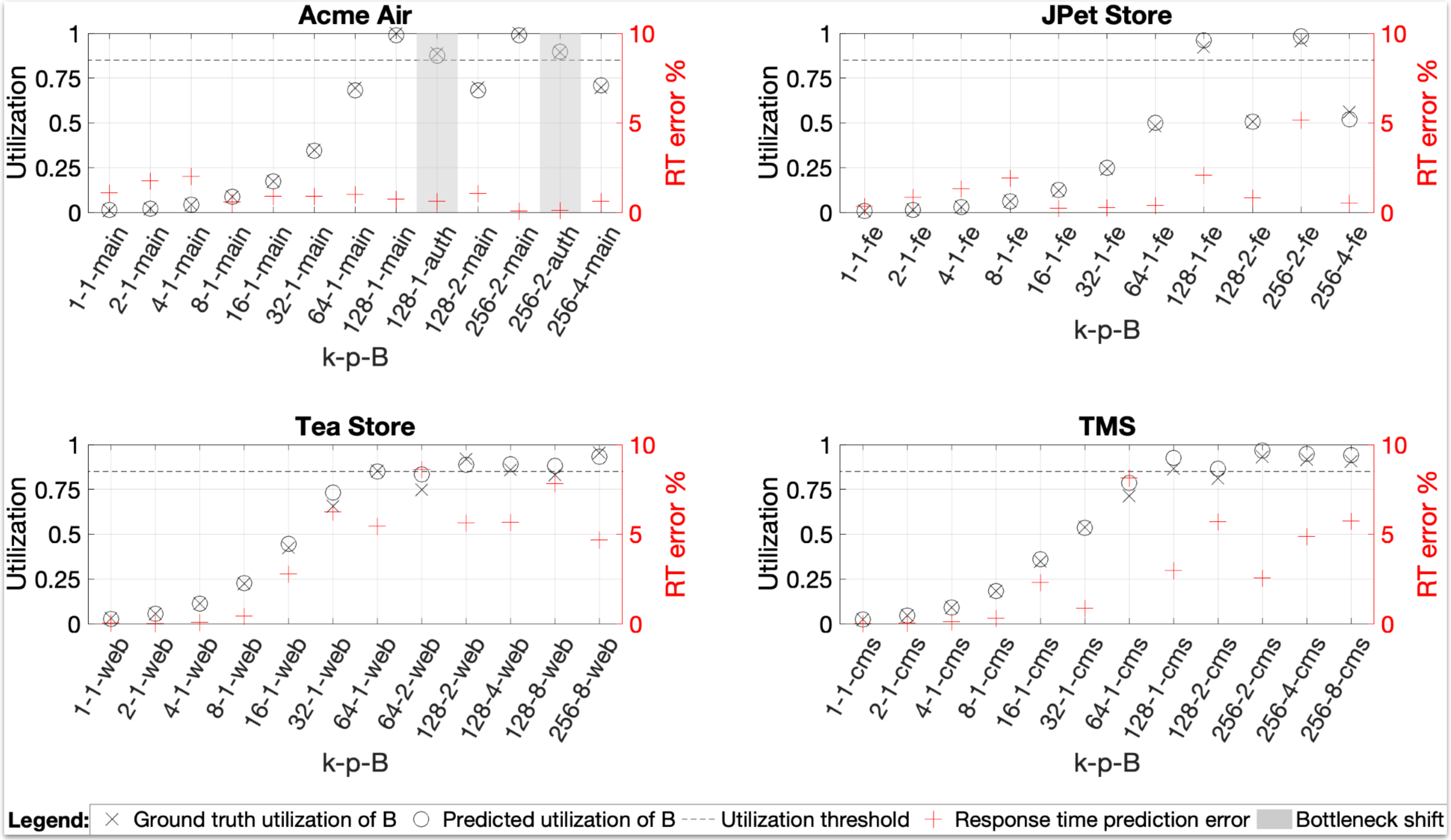
µP: Predicting Performance of Microservices by Design

Validation: Increasing Load



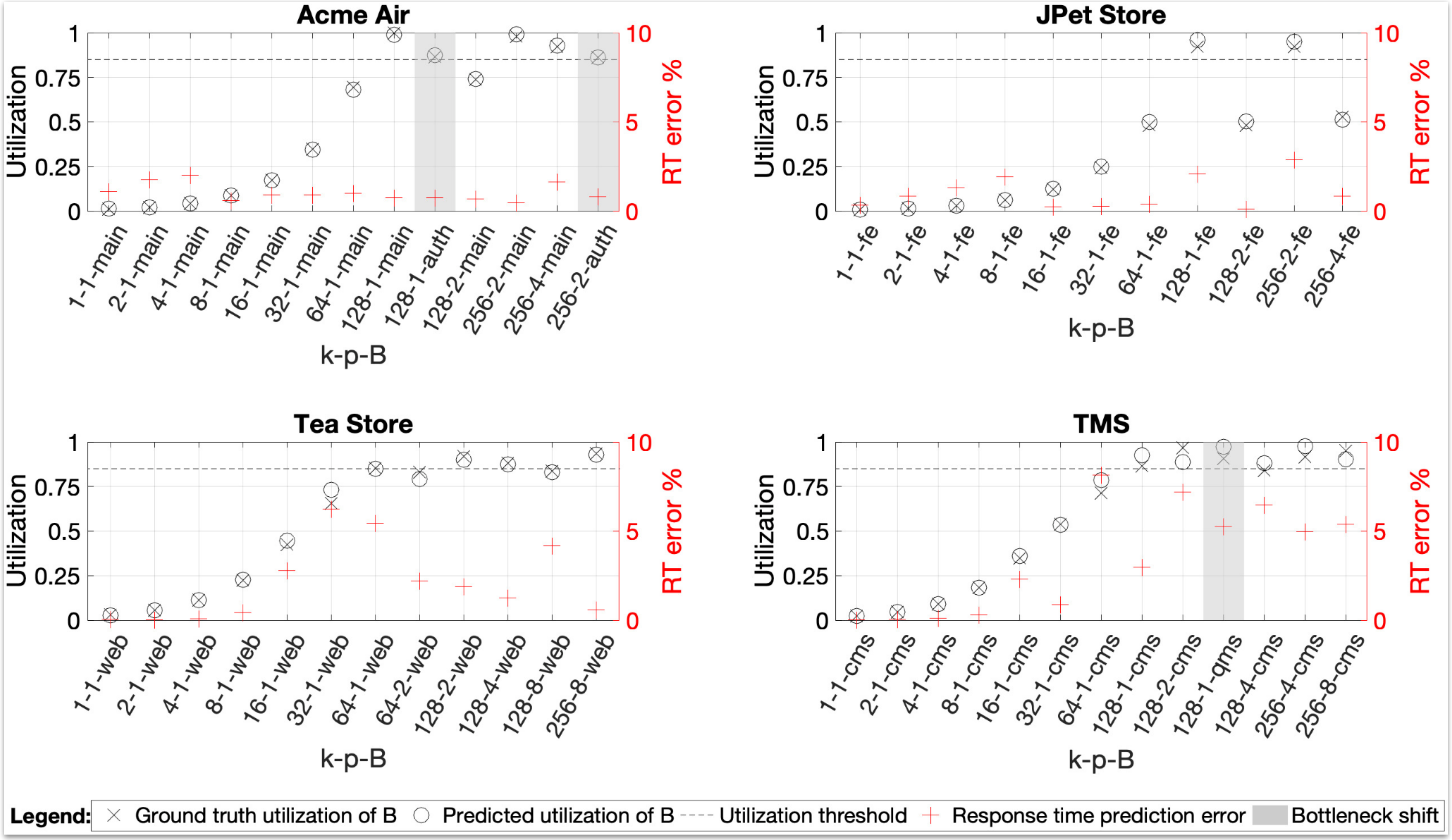
µP: Predicting Performance of Microservices by Design

Validation: Vertical Scaling (add more workers)



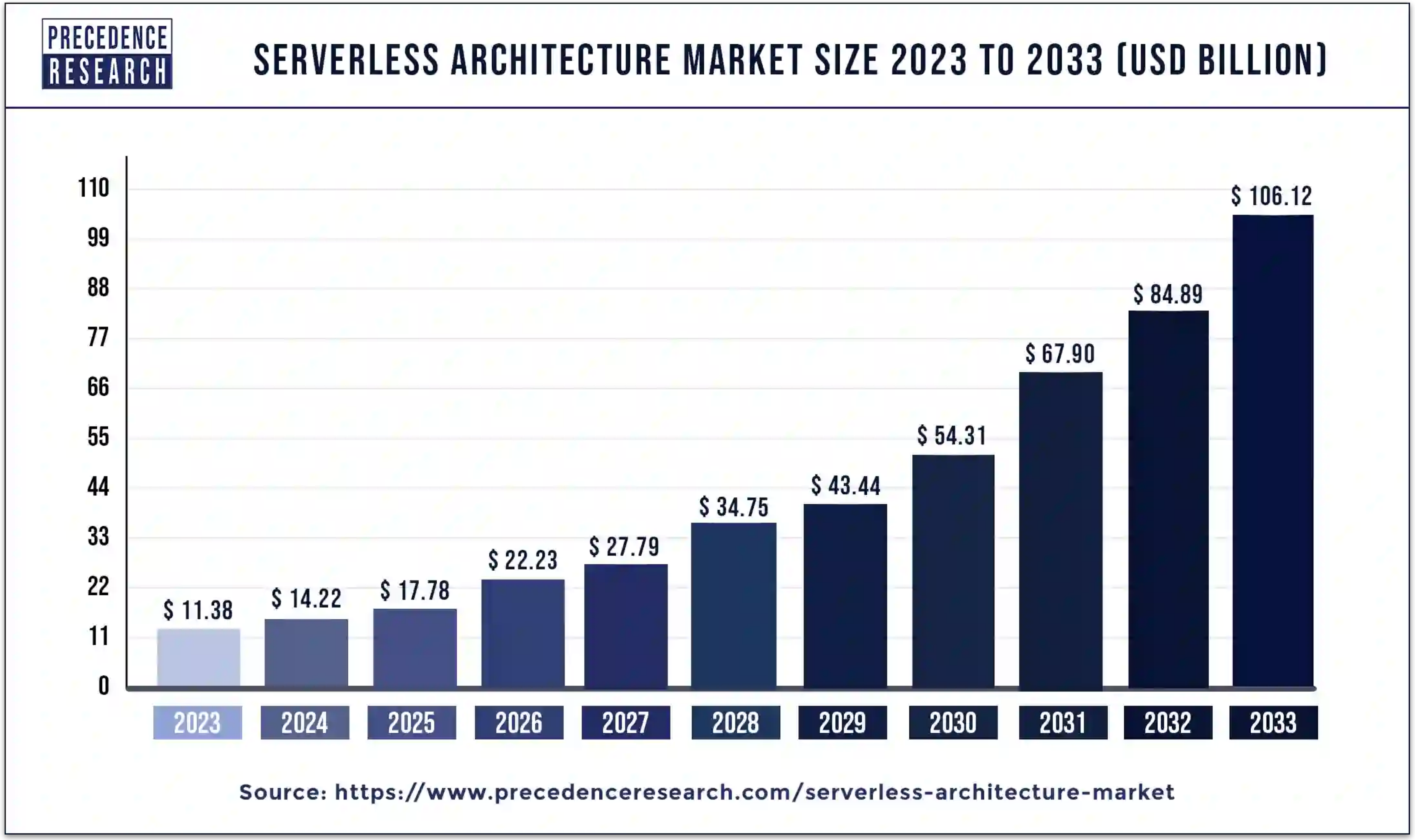
μP: Predicting Performance of Microservices by Design

Validation: Horizontal Scaling (add more instances)



Ongoing Work: Serverless Computing

Growing interest from academia and industry



Serverless Computing

FaaS: Function-as-a-Service

Benefits

Developers focus on code

Deployment and scaling is the platform's responsibility

Billing based on actual number of executed instances (*billable instances*)

Limitations

Developers must manually specify CPU and memory requirements, and maximum concurrency level for each function instance *(2nd gen.)*

Cold starts may negatively impact performance

Performance optimization by **trial and error**

Tune concurrency for your service

The number of concurrent requests that each instance can serve can be limited by the technology stack and the use of shared resources such as variables and database connections.

To optimize your service for maximum stable concurrency:

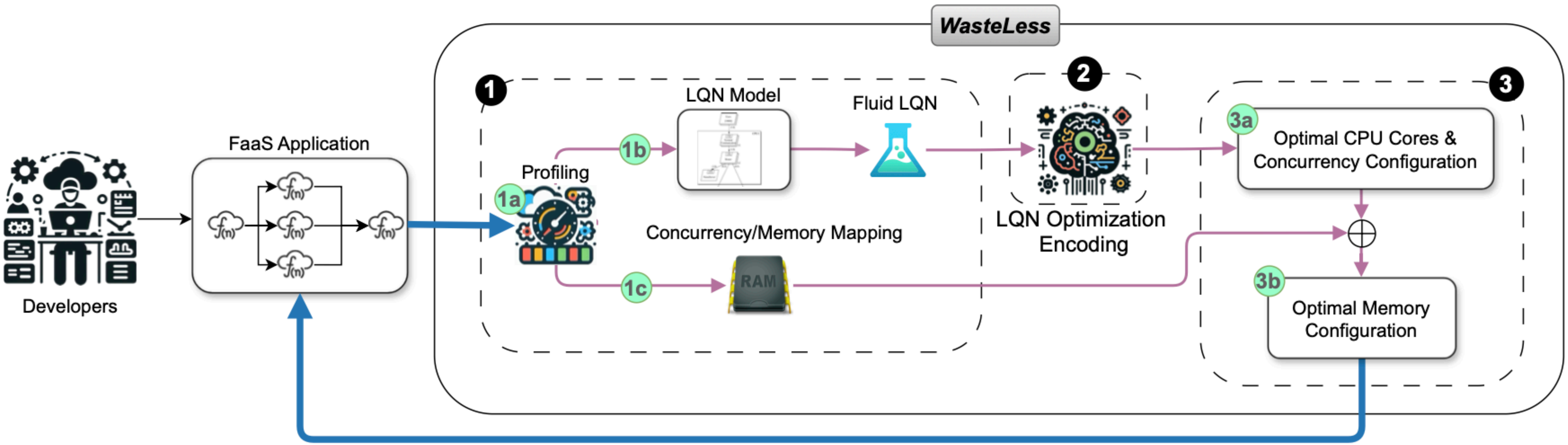
1. Optimize your service performance.
2. Set your expected level of concurrency support in any code-level concurrency configuration. Not all technology stacks require such a setting.
3. Deploy your service.
4. Set Cloud Run concurrency for your service equal or less than any code-level configuration. If there is no code-level configuration, use your expected concurrency.
5. Use load testing tools that support a configurable concurrency. You need to confirm that your service remains stable under expected load and concurrency.
6. If the service does poorly, go to step 1 to improve the service or step 2 to reduce the concurrency. If the service does well, go back to step 2 and increase the concurrency.

Continue iterating until you find the maximum stable concurrency.

https://cloud.google.com/run/docs/tips/general#tuning-concurrency

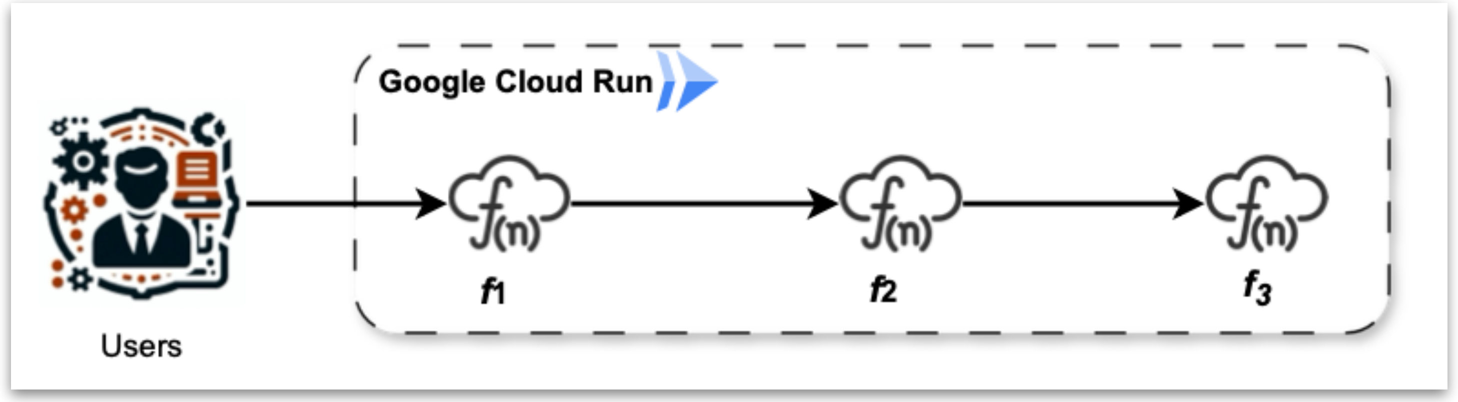
WasteLess

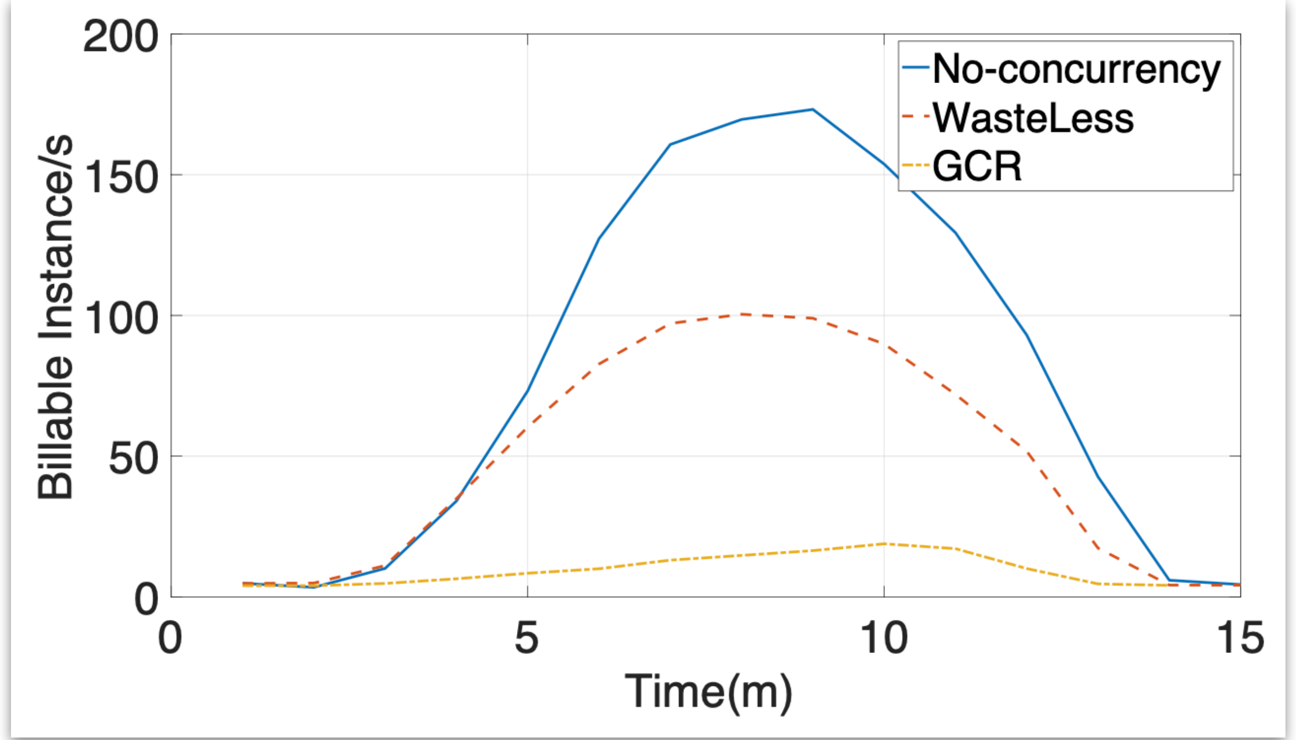
Model-driven Optimization for FaaS

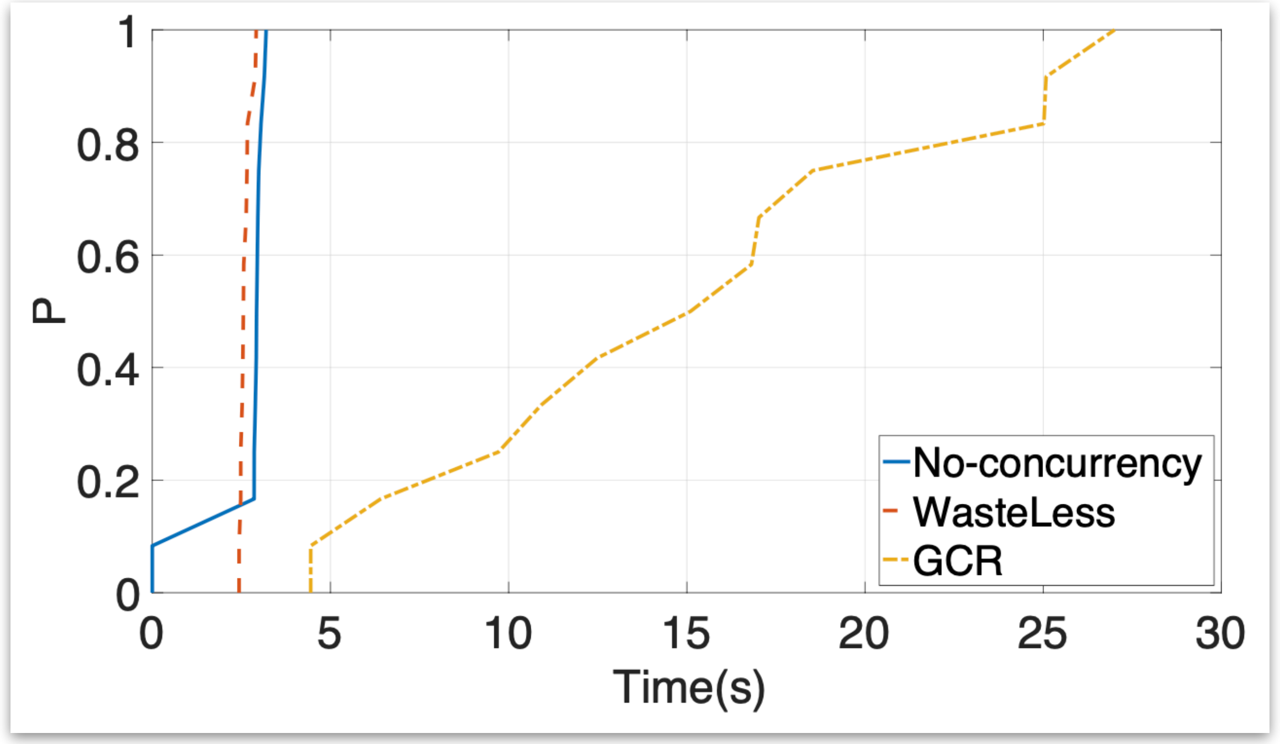


WasteLess: Validation Results

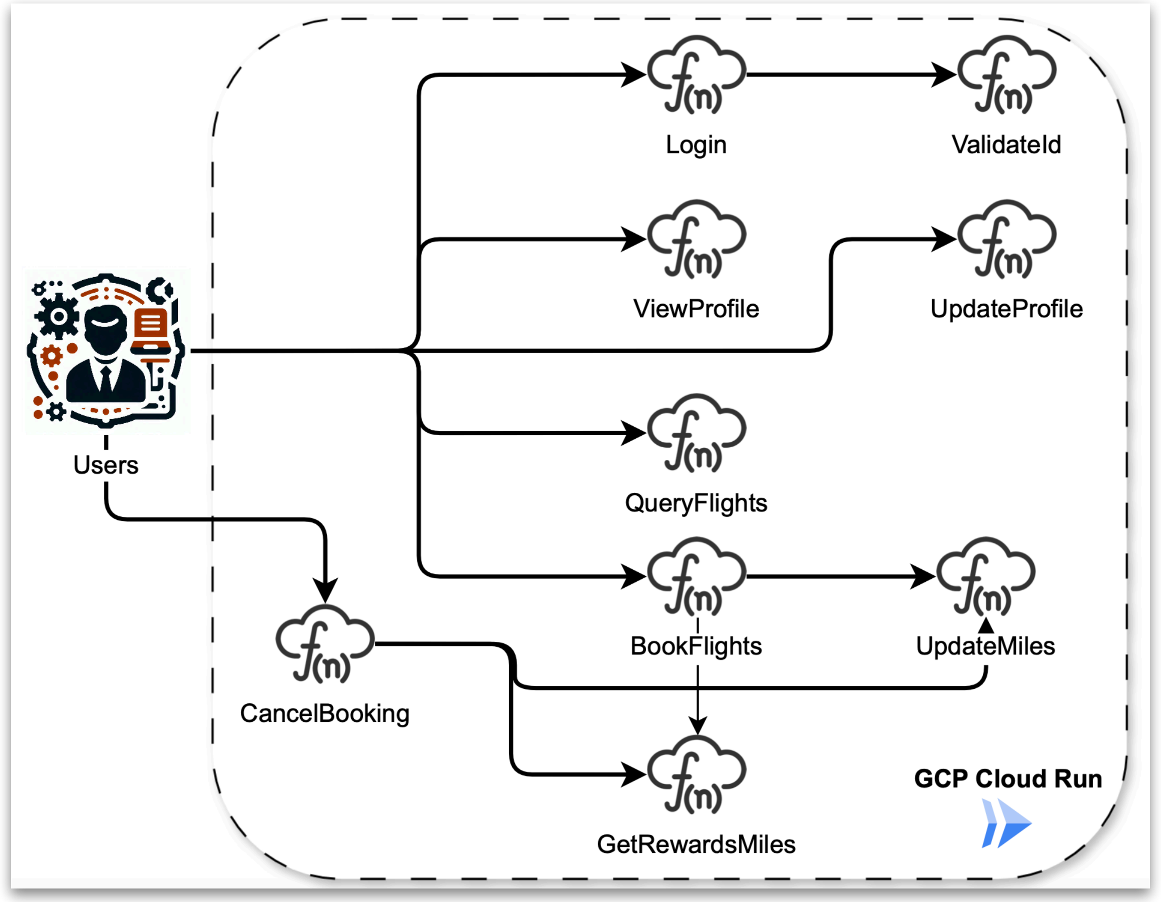
Simple three-tier app



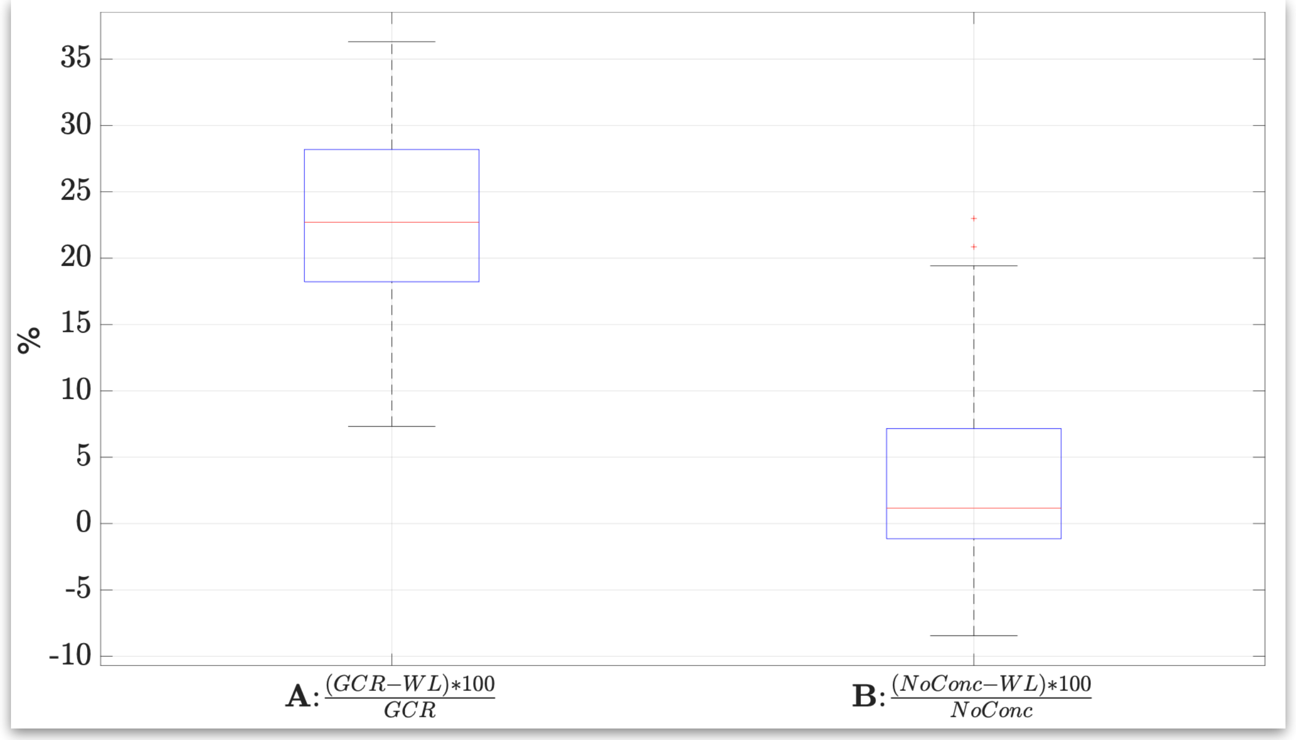




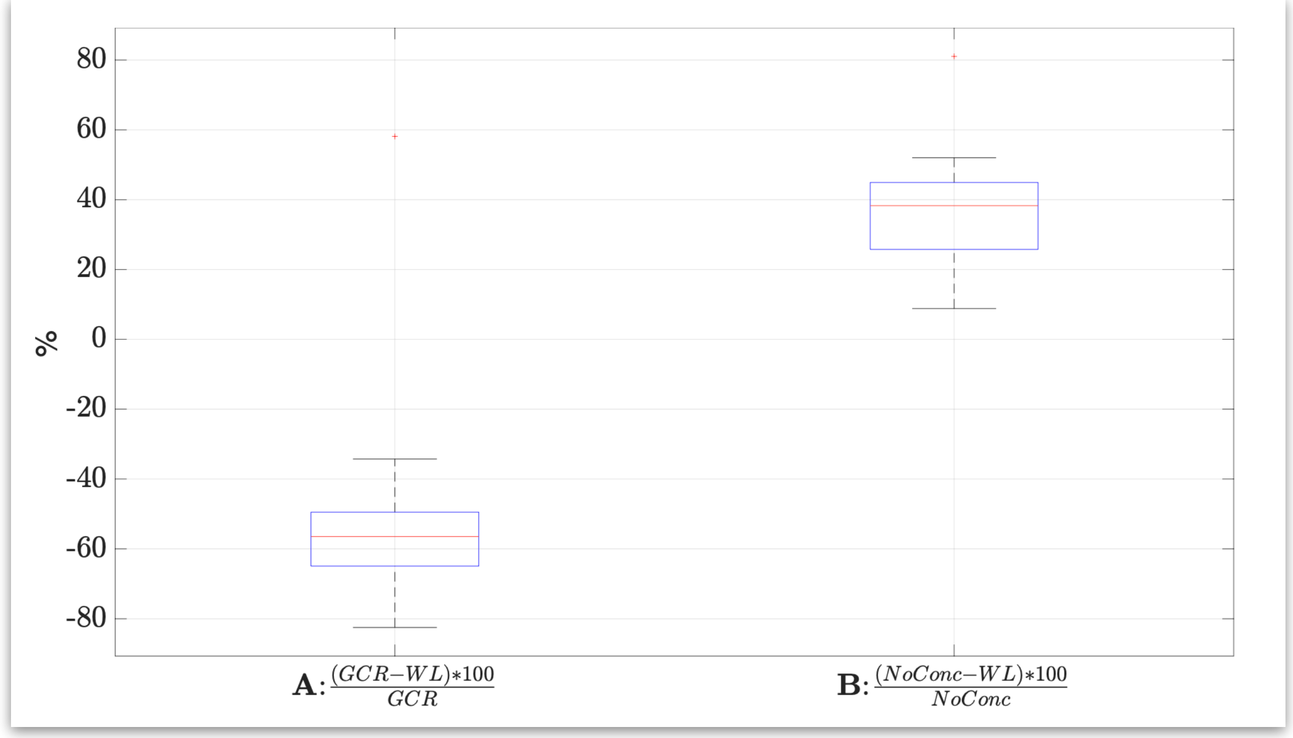
ACME Air (GCR Porting)

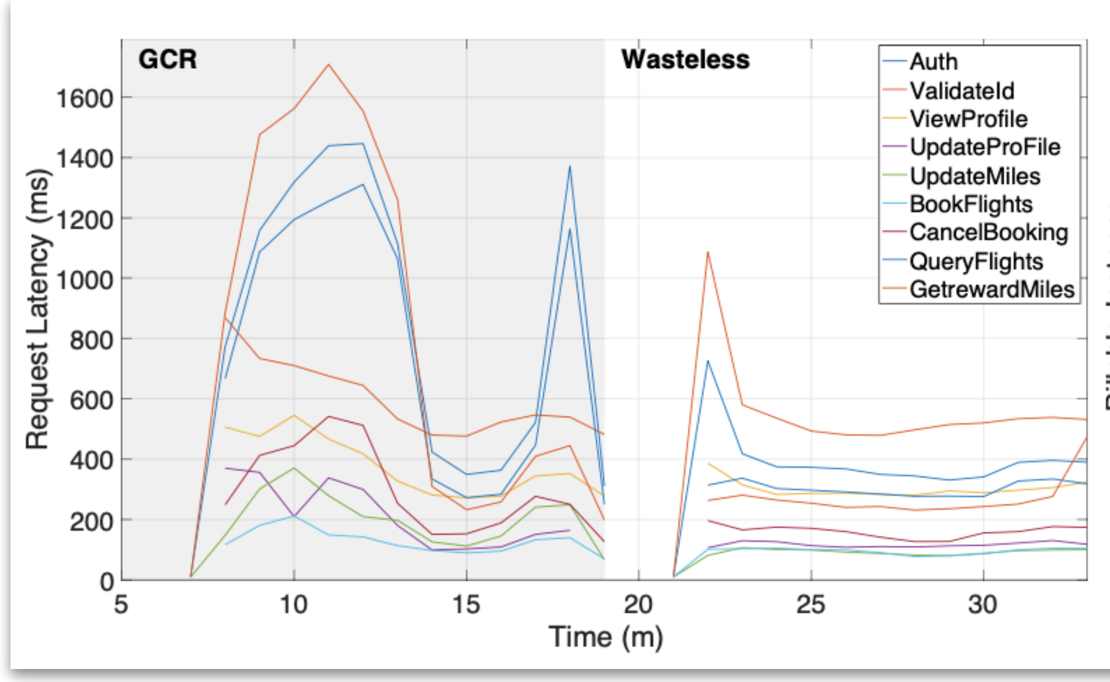


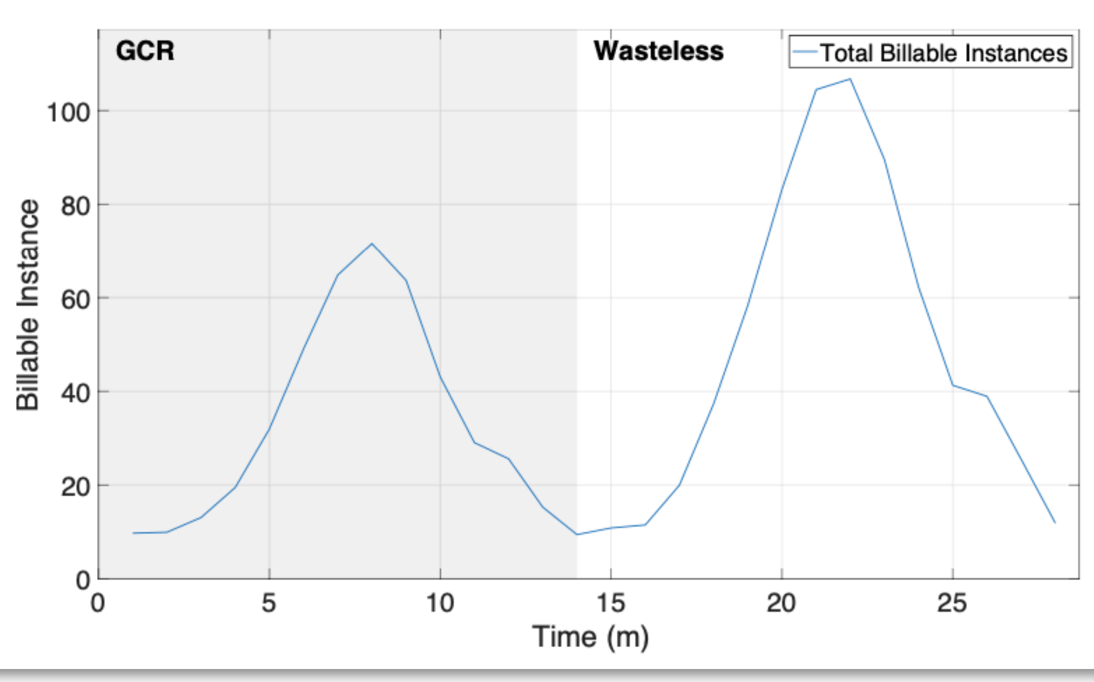
End-to-end response time

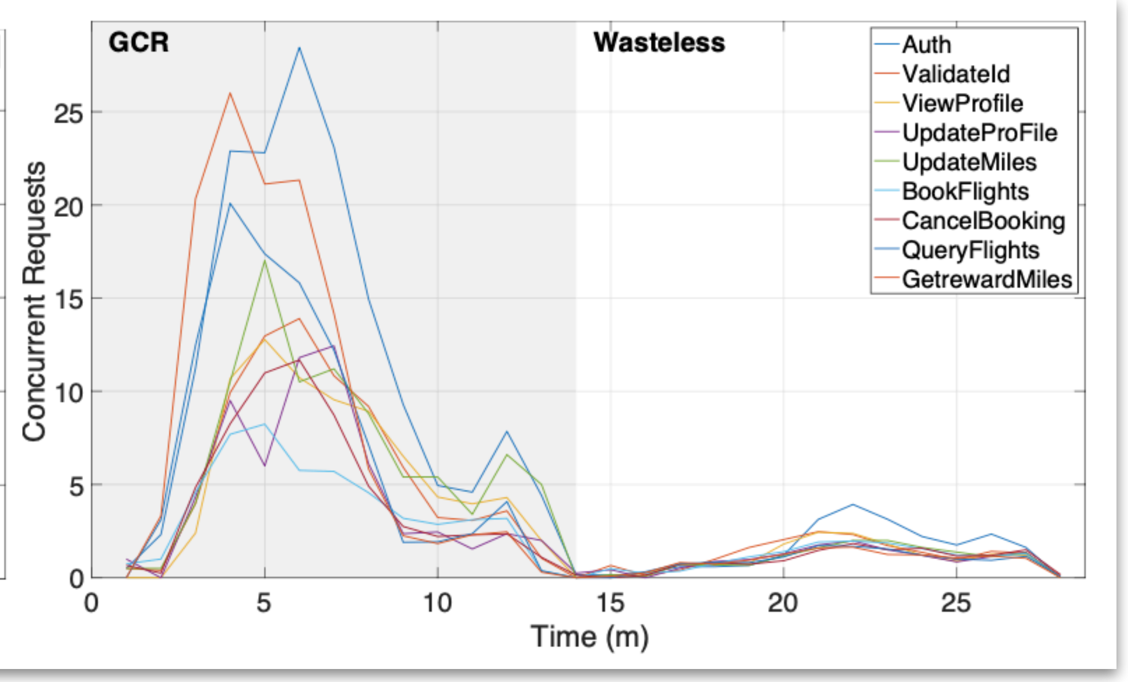


Billable Instances

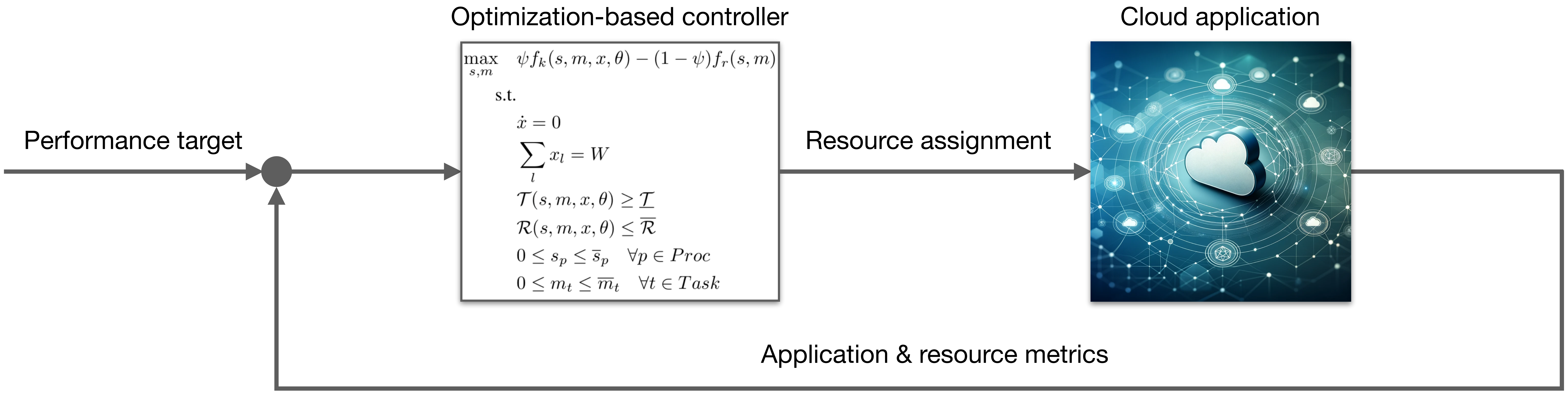








Limitations and Future Work



Model-based optimal autoscalers can be effective

Difficulty in comparing techniques developed in the literature (competition? more benchmarks?)

Modeling assumptions may be difficult to hold in practice:
full availability of the application under study; homogeneity of the platform on which the application runs; ability to isolate the application to execute calibration runs; …

Right-level of abstraction is difficult to find (art more than science?): Techniques to derive model abstractions automatically (from code, bytecode, etc.)? Using model simplification methods?

Francesca Randone, Luca Bortolussi, Emilio Incerto, Mirco Tribastone:. Inference of Probabilistic Programs with Moment-Matching Gaussian Mixtures. POPL 2024

Acknowledgements

- **Emilio Incerto (main collaborator)**
- Luca Bortolussi
- Giulio Garbi
- Nicolas Gast
- Roberto Pizziol
- Francesca Randone
- Gabriele Russo Russo
- Catia Trubiani
- Tabea Waizmann

Finanziato dall'Unione europea
NextGenerationEU

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

THE
Tuscany Health Ecosystem

Other References

Model simplification methods

- Luca Cardelli, Giuseppe Squillace, Mirco Tribastone, Max Tschaikowski, Andrea Vandin: Formal lumping of polynomial differential equations through approximate equivalences. J. Log. Algebraic Methods Program. 134: 100876 (2023)
- Luca Cardelli, Radu Grosu, Kim Guldstrand Larsen, Mirco Tribastone, Max Tschaikowski, Andrea Vandin: Algorithmic Minimization of Uncertain Continuous-Time Markov Chains. IEEE Trans. Autom. Control. 68(11): 6557-6572 (2023)
- Alexey Ovchinnikov, Isabel Cristina Pérez-Verona, Gleb Pogudin, Mirco Tribastone: CLUE: exact maximal reduction of kinetic models by constrained lumping of differential equations. Bioinform. 37(12): 1732-1738 (2021)
- L Cardelli, M Tribastone, M Tschaikowski, A Vandin. Symbolic computation of differential equivalences. Theoretical Computer Science 777, 132-154
- Luca Cardelli, Mirco Tribastone, Max Tschaikowski, Andrea Vandin: Guaranteed Error Bounds on Approximate Model Abstractions Through Reachability Analysis. QEST 2018: 104-121
- L Cardelli, IC Perez-Verona, M Tribastone, M Tschaikowski, A Vandin. Exact maximal reduction of stochastic reaction networks by species lumping. Bioinformatics 37 (15), 2175-2182
- Luca Cardelli, Mirco Tribastone, Max Tschaikowski, Andrea Vandin: Efficient Syntax-Driven Lumping of Differential Equations. TACAS 2016