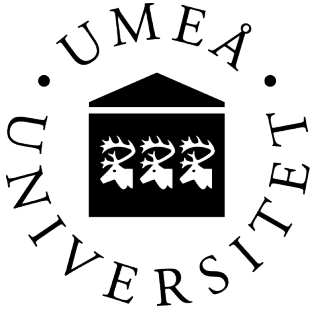
How I Learned to Stop Worrying and Love Capacity Shortages

Cristian Klein
Umeå University



2014-05-08, LCCC Workshop on Cloud Control
Lund, Sweden

Cloud Computing Definition

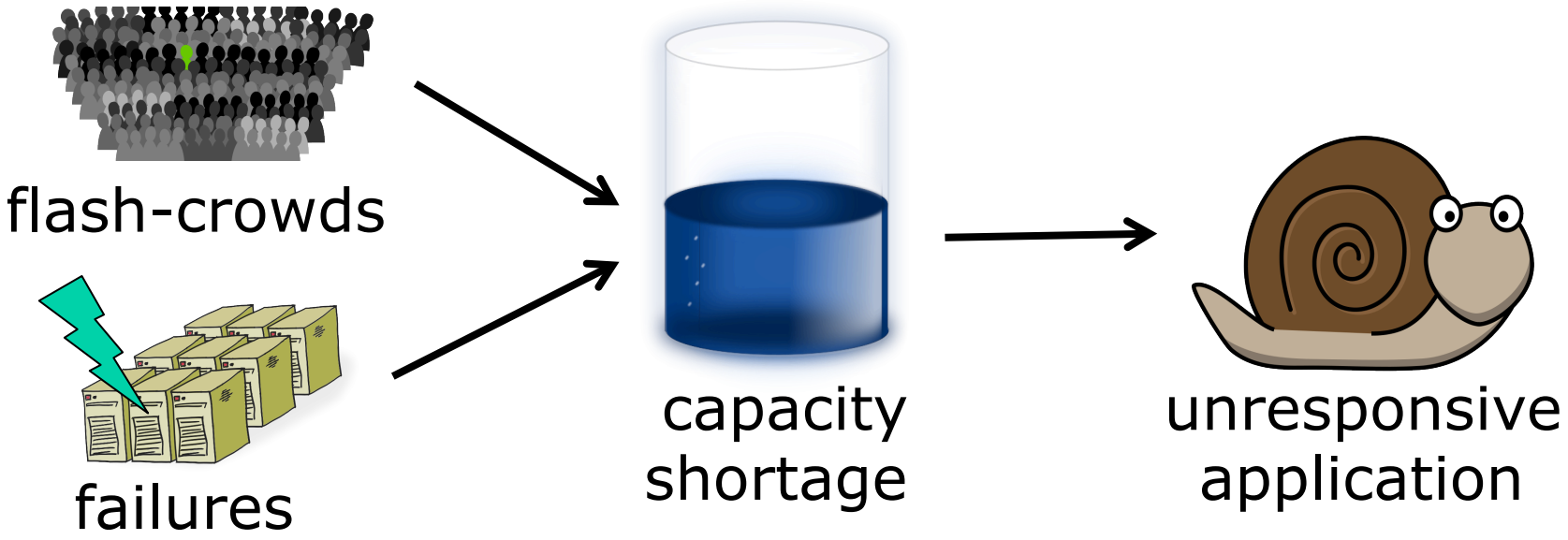
Rapid provisioning (and release) from a **shared pool** of resources

- 3 deployment models
  - Public cloud ("our stuff")
  - **Private cloud ("my stuff")**
  - Hybrid cloud (combine the two above)
- 3 service models (what is leased)
  - Software-as-a-Service (SaaS)
  - Platform-as-a-Service (PaaS)
  - **Infrastructure-as-a-Service (IaaS)**

Infrastructure-as-a-Service (IaaS)

- Data-center management
  - Lease CPU, memory, storage
  - Allocate **capacity**
  - Packed as **Virtual Machine (VM)**
- 3 stakeholders
  - Infrastructure Provider (IP)
  - Service/Application Provider (SP)
  - End-user

Problem: Unexpected Events

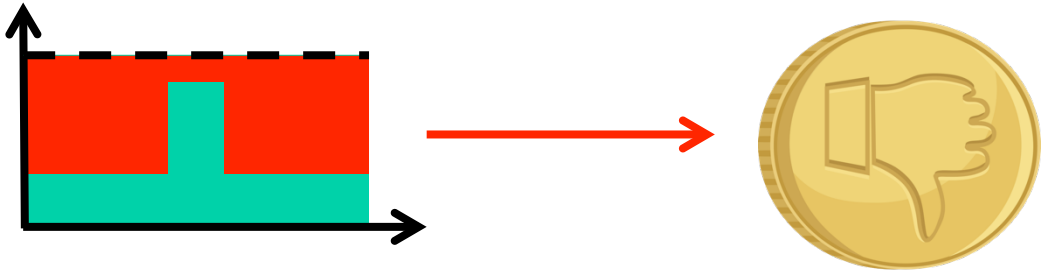


- 82% of end-users give up on a lost payment transaction*
- 25% of end-users leave if load time > 4s**
- 1% reduced sale per 100ms load time**
- 20% reduced income if 0.5s longer load time***

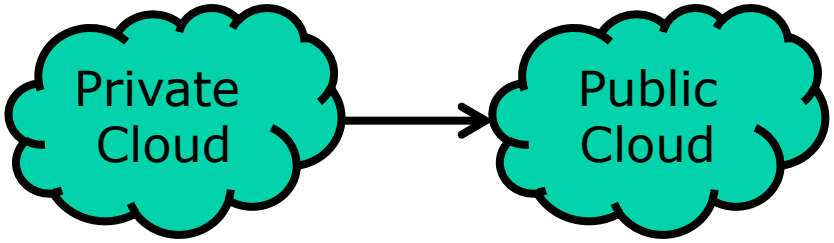
* JupiterResearch ** Amazon ***Google

State-of-Practice

- Large spare capacity
  - May be economically impractical

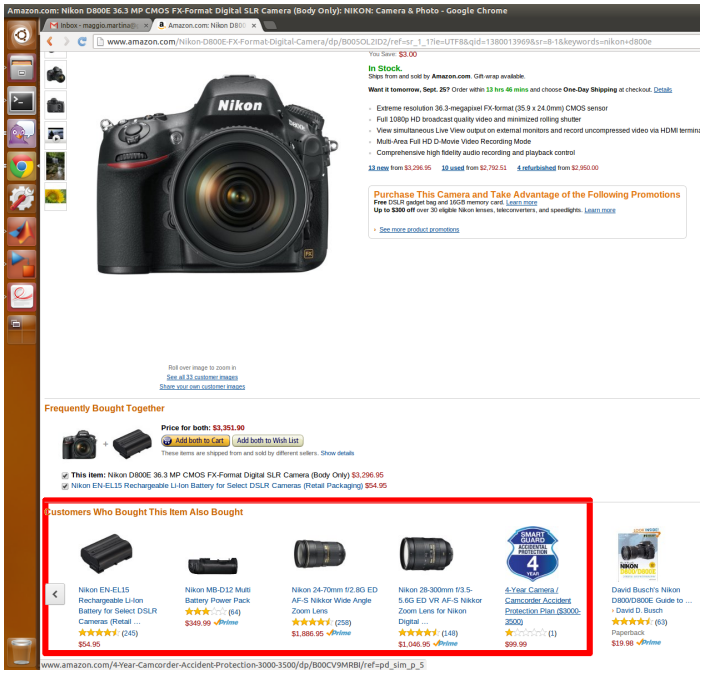


- Cloud bursting
  - Lease capacity from a public cloud
  - Does not really solve the problem



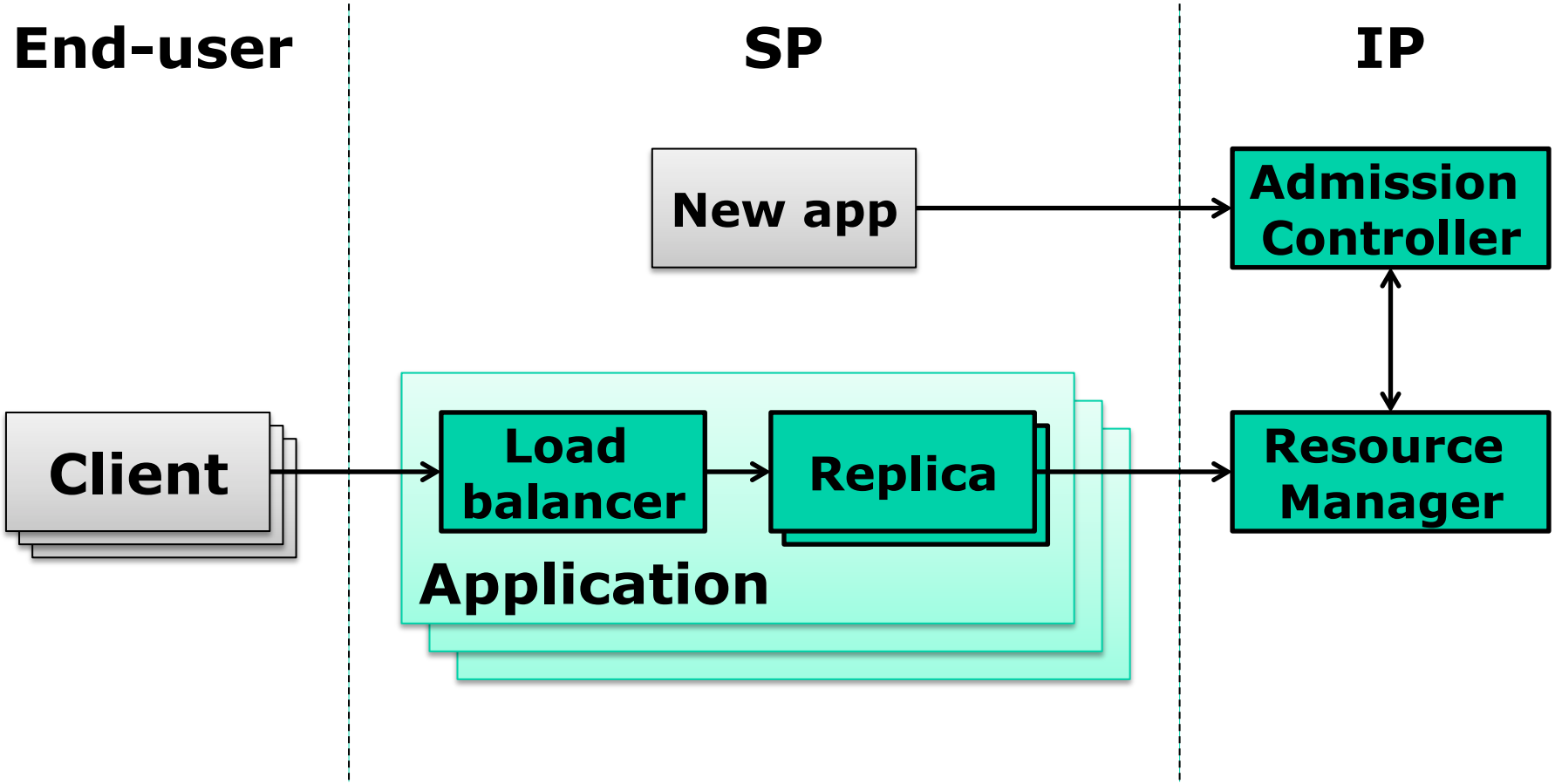
Brownout: Idea

- Disable **optional** content
  - Minimally intrusive
- E.g. recommendations
  - 50% increase in sales *
- Challenge
  - Maximize optional content
  - Avoid high response times

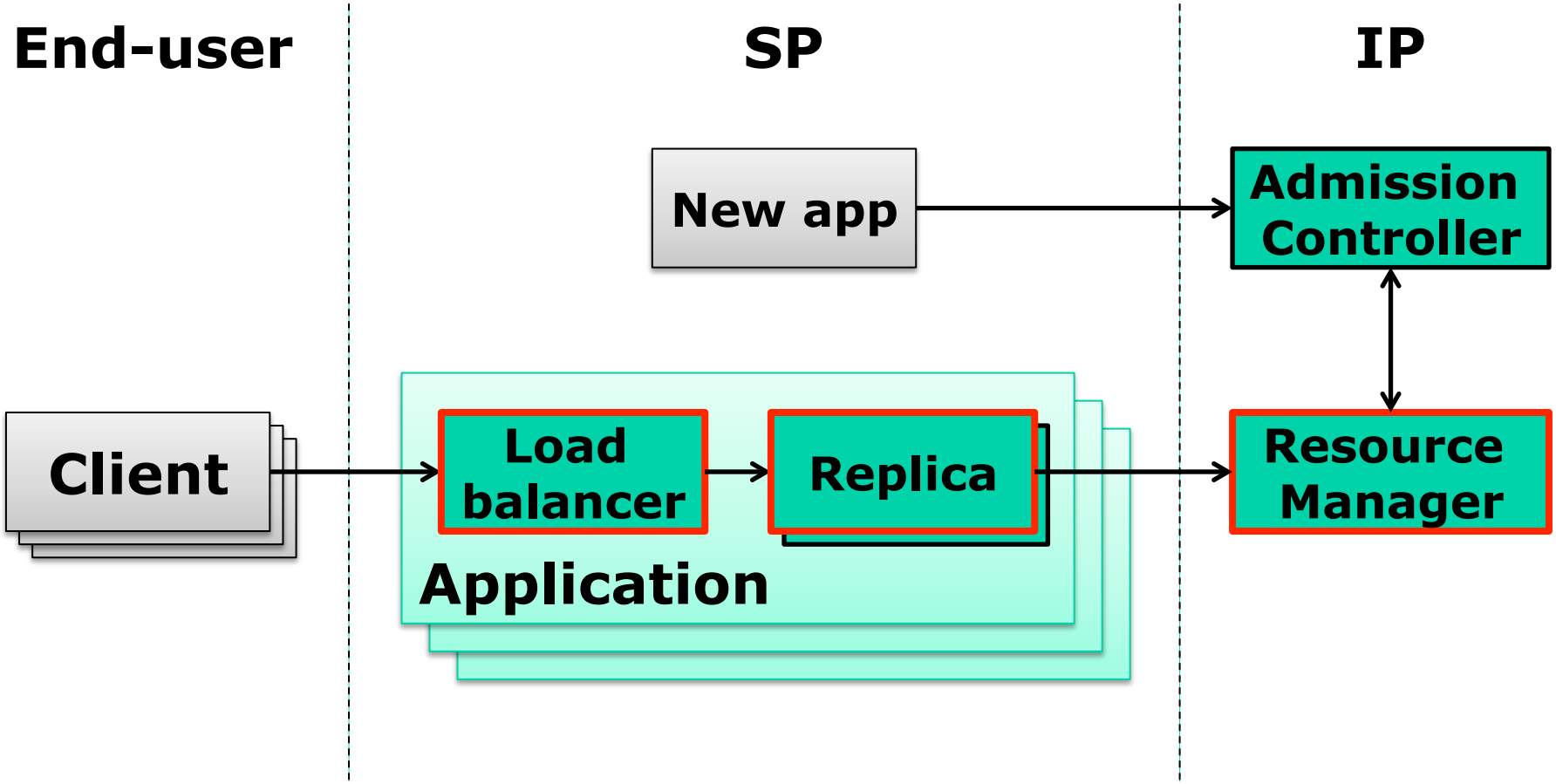


* D. Fleder et al., “Recommender systems and their effects on consumers: the fragmentation debate,” in Electronic Commerce, 2010. DOI: 10.1145/1807342. 1807378.

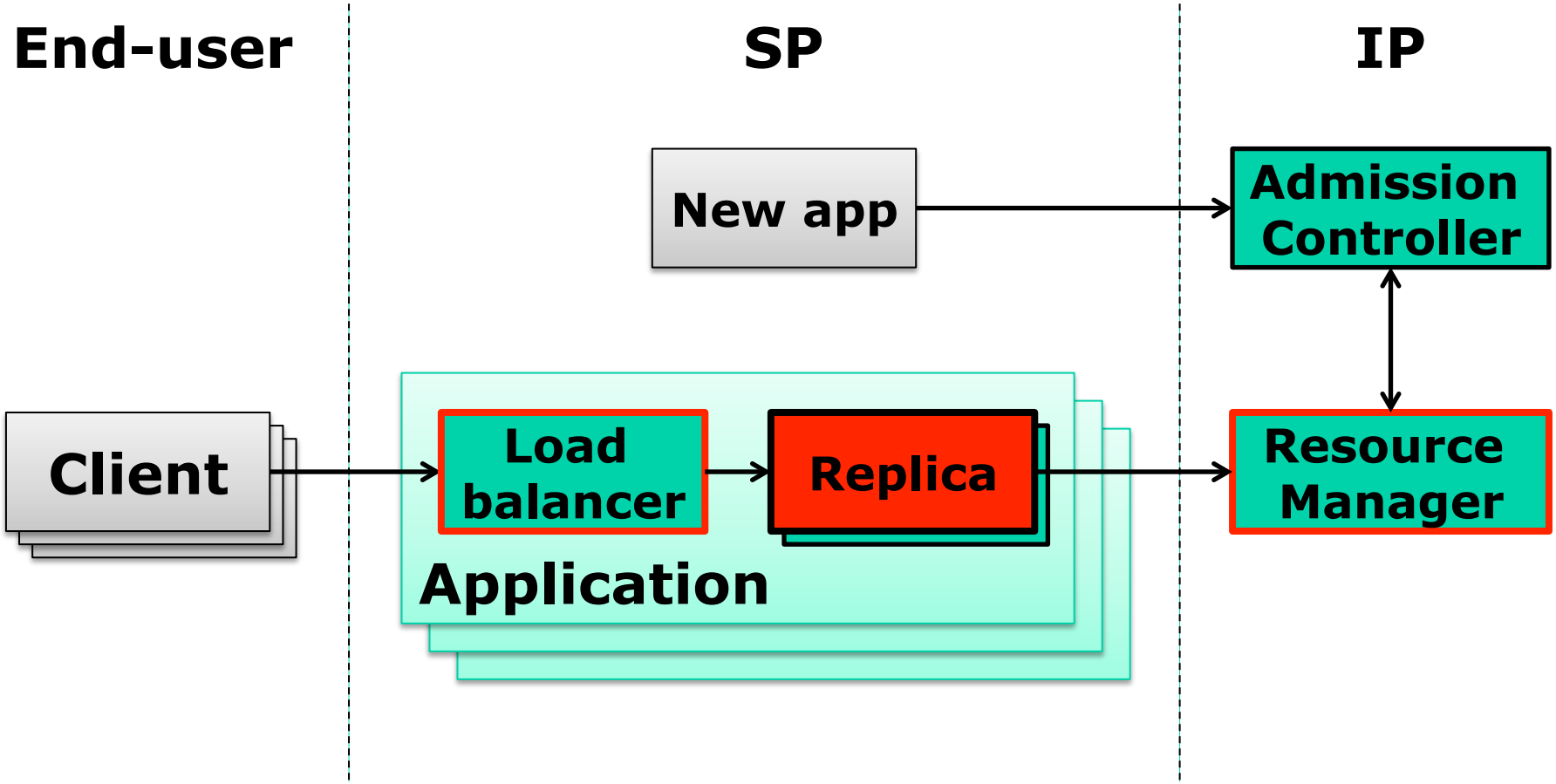
Cloud Architecture



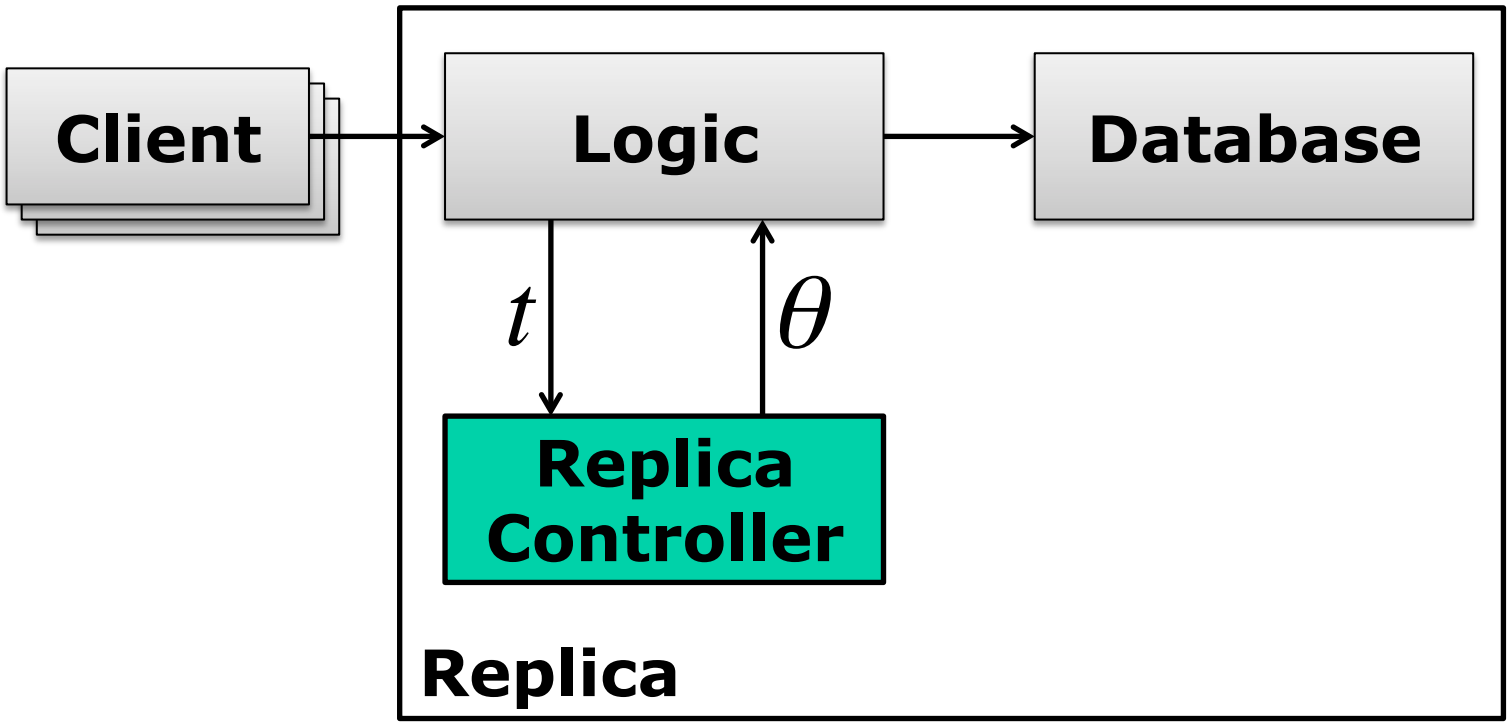
Cloud Architecture



Cloud Architecture



Brownout: Inside a Replica



t = response times

θ = probability of serving optional content (**dimmer**)

C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén,
"Brownout: building more robust cloud applications", ICSE, 2014

Replica Controller (1)

- Need to adapt to changes
  - Number of users
  - Available capacity
- Not all requests take the same time
  - E.g., cached in memory, disk
- Need to reject disturbances
  - E.g., NTP daemon firing up, cron jobs

M. Maggio, C. Klein, K-E Årzén,
"Control strategies for predictable brownouts in cloud computing", IFAC, 2014

Replica Controller (2)

- Start from a simple model

$$t^{k+1} = \alpha^k \cdot \Theta^k + \delta t^k$$

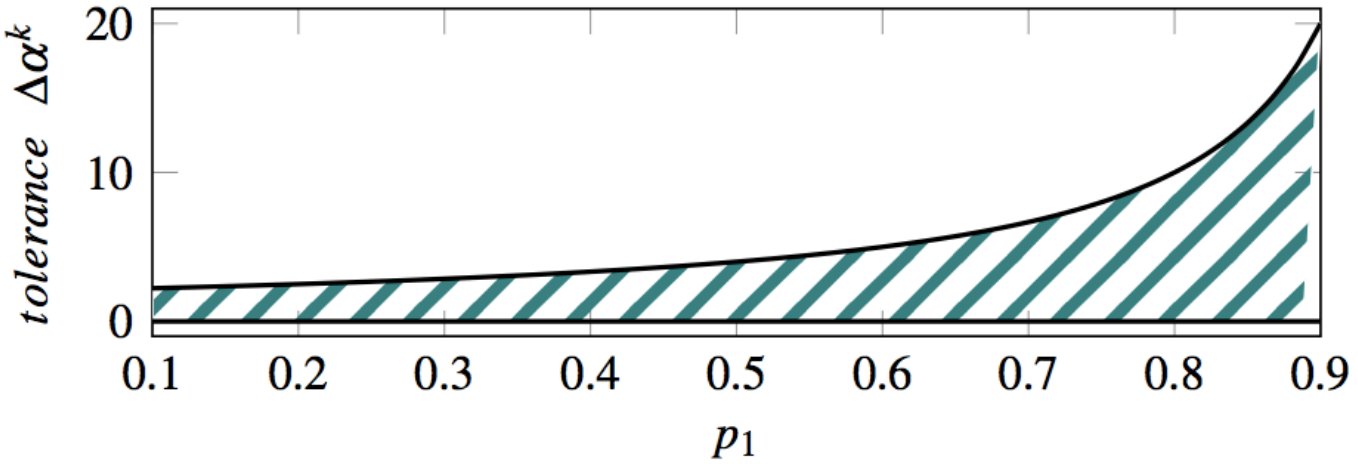
- Adaptive PI controller

$$\Theta^{k+1} = \Theta^k + \frac{1 - p_1}{\alpha} \cdot e^{k+1}$$

- α estimated using RLS

Robustness to Model Uncertainties

$$\alpha = \tilde{\alpha} \cdot \Delta\alpha$$



Evaluation

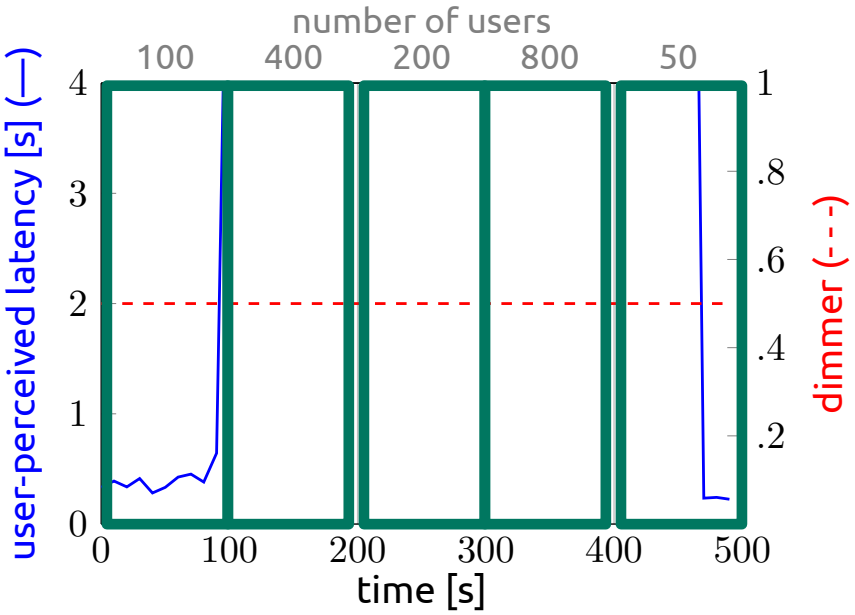
- RUBiS: eBay-like prototype
  - Added a recommender
- RUBBoS: Slashdot-like prototype
  - Added a recommender
  - Marked comments as optional

- Effort in lines of code:

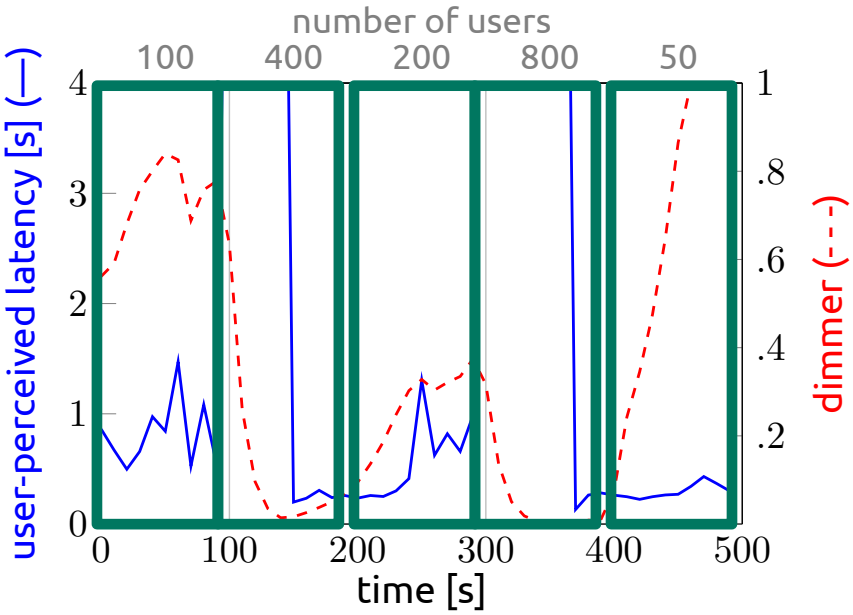
Modification	RUBiS	RUBBoS
Recommender	37	22
Dimmer	3	6
Reporting response time to controller	5	5
Controller	120	120
Total	*165*	*153*

Results: RUBiS, flash-crowd

Non-adaptive vs. self-adaptive



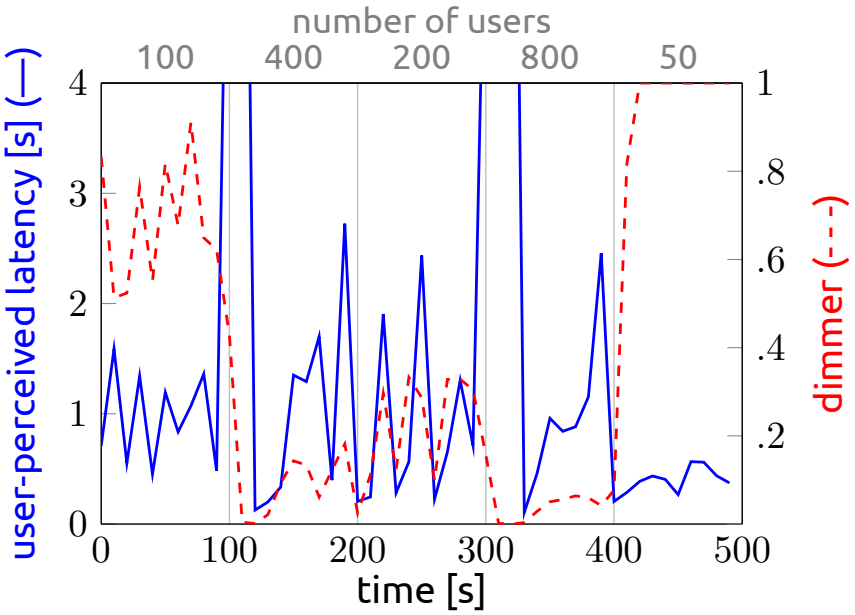
non-adaptive



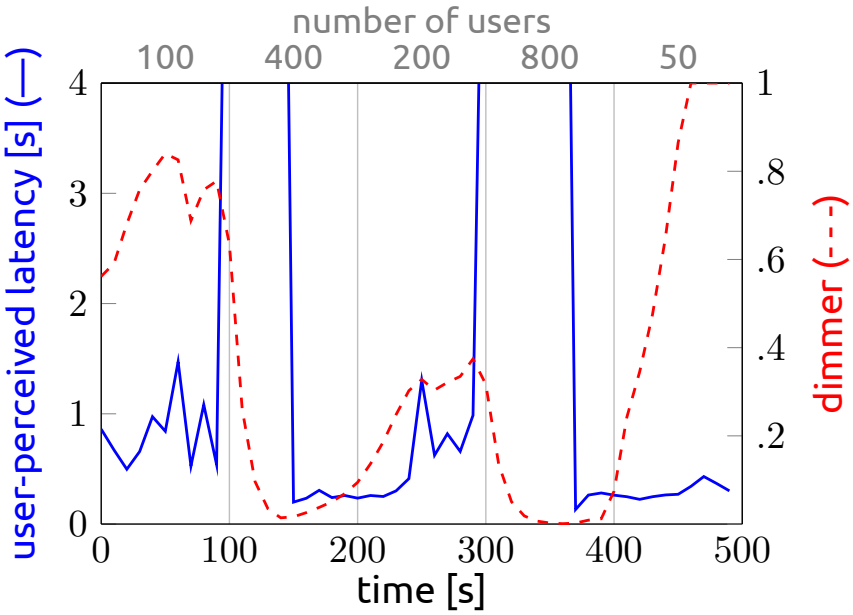
self-adaptive
pole 0.9

Results: RUBiS, flash-crowd

Self-adaptive: different pole values

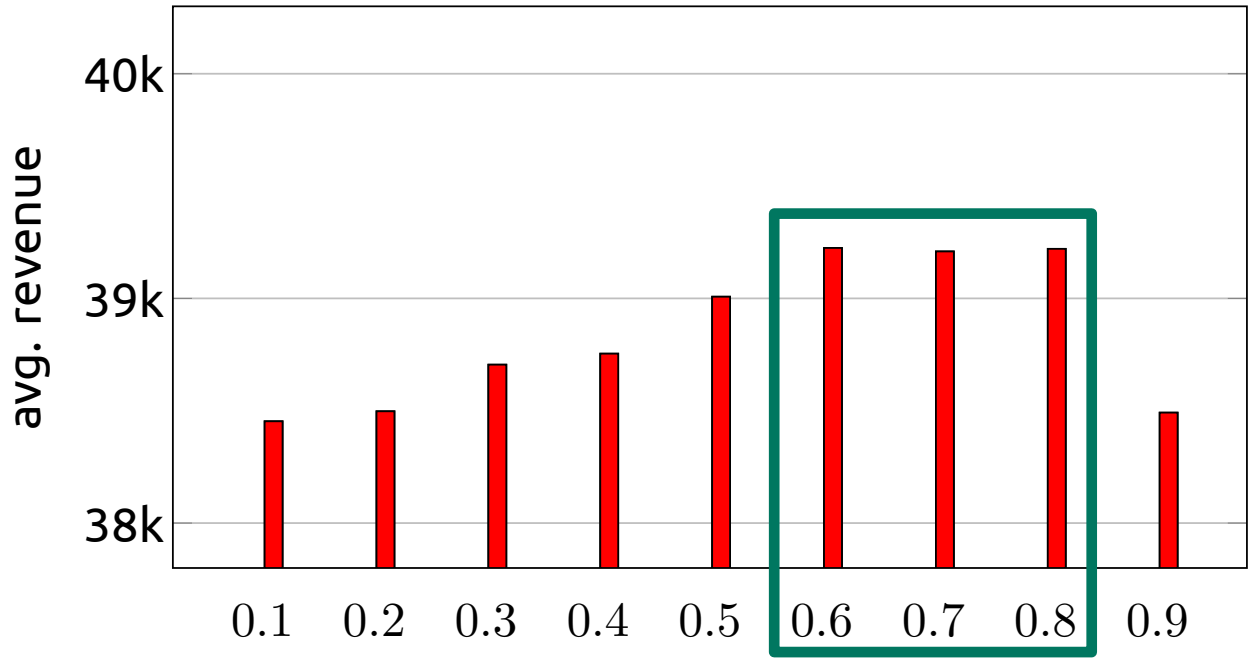


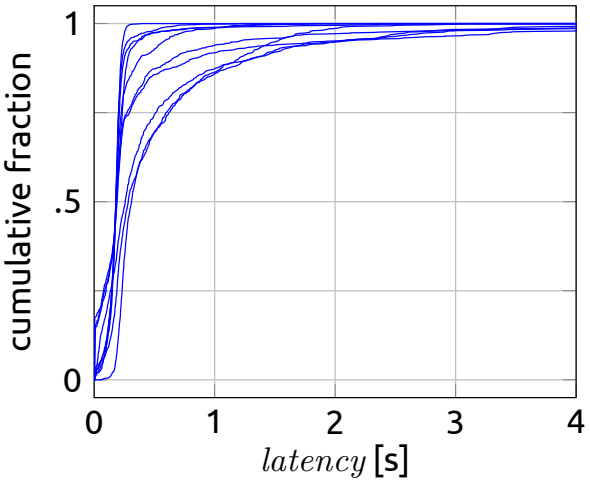
self-adaptive
pole 0.5



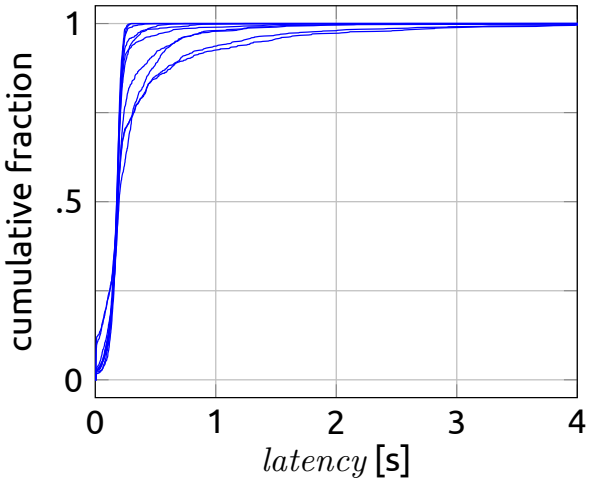
self-adaptive
pole 0.9

Results: revenue = 1×#requests + 0.5×#optional

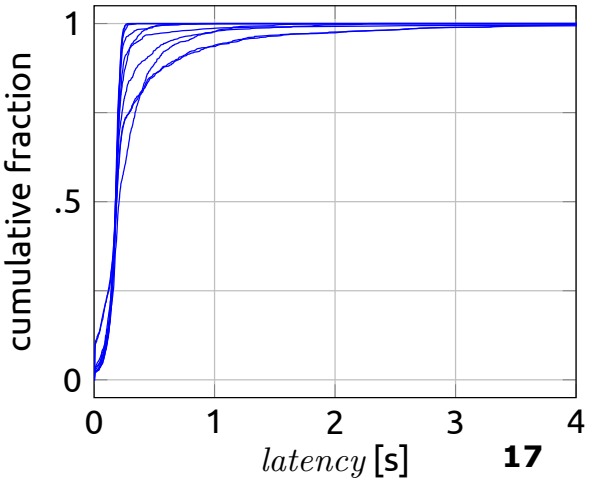




pole 0.6

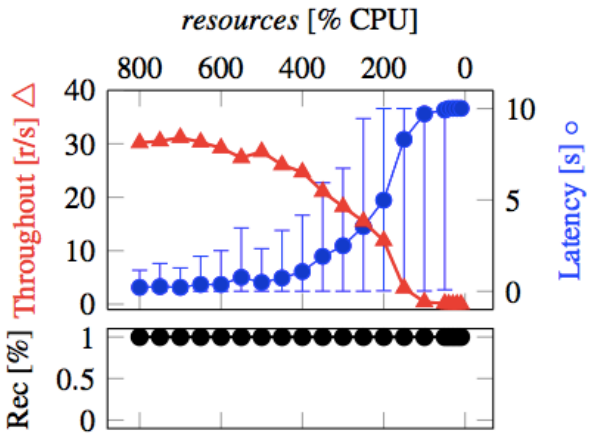


pole 0.7

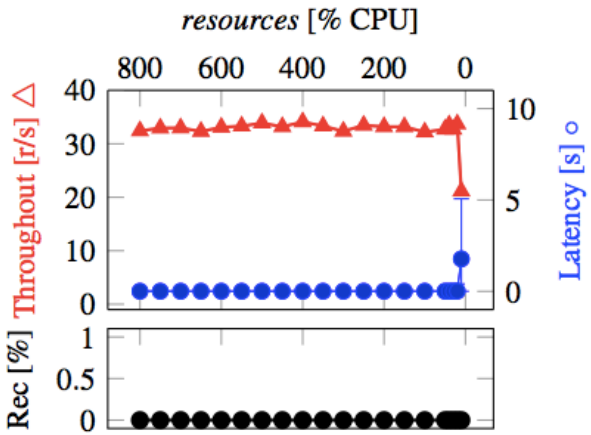


pole 0.8

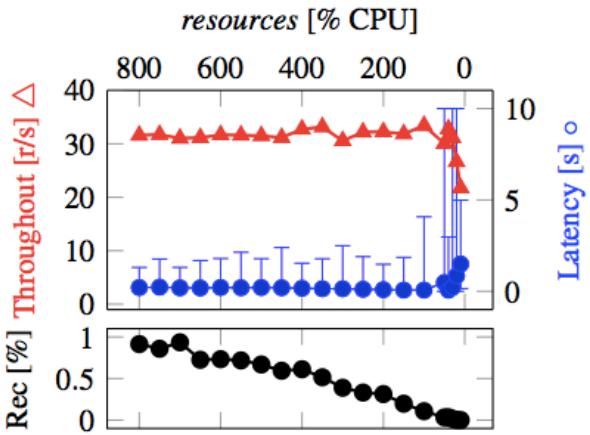
Results: Stress Test



(a) with all recommendations



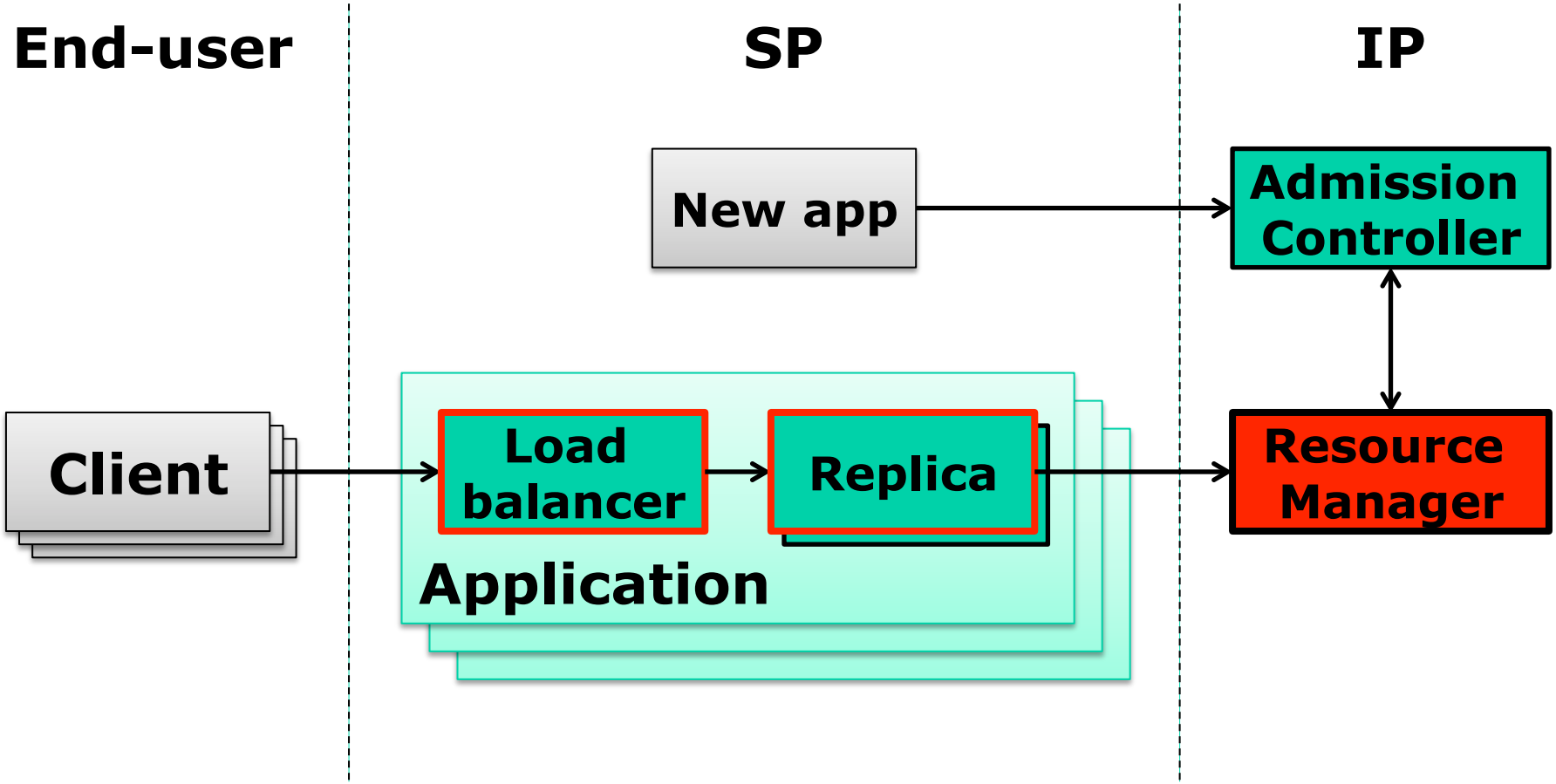
(b) with no recommendations



(c) self-adaptive

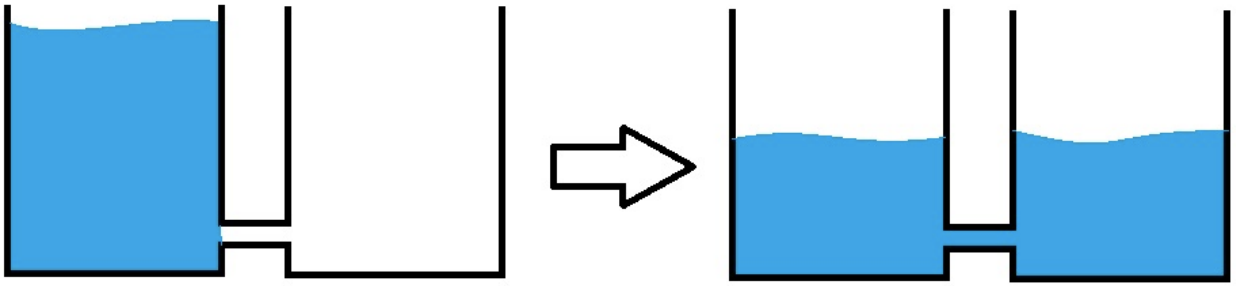
Can we do better?

Cloud Architecture

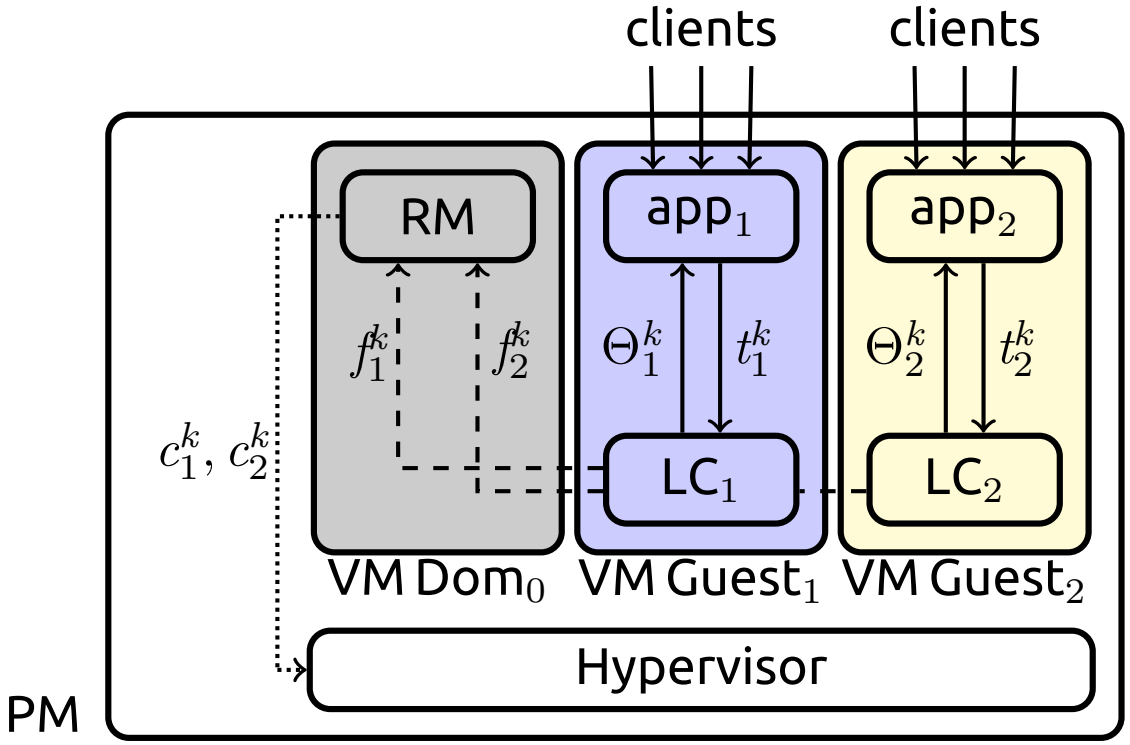


Goal

- Give capacity to application that “struggles”
- **Fairly** balances capacity among applications



Zoom: Inside a Physical Machine



C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén,
"Resource management for service level aware cloud applications", REACTION, 2013

Details

- Application sends **matching values**

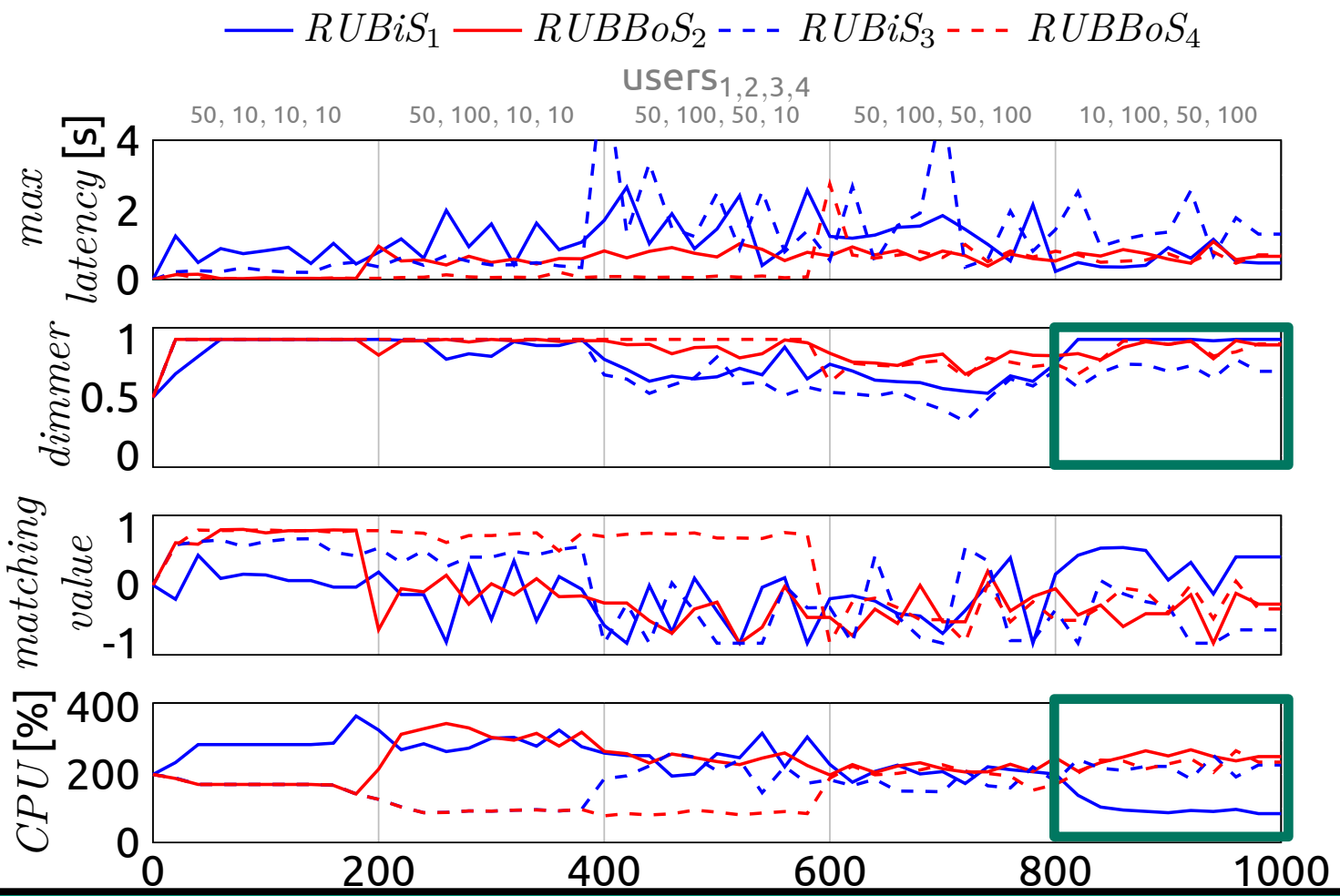
$$f_i^k = 1 - t_i^k / \bar{t}_i$$

- Resource manager computes **capacities**

$$c_i^{k+1} = c_i^k - \varepsilon_{rm} \left(f_i^k - c_i^k \cdot \sum_p f_p^k \right)$$

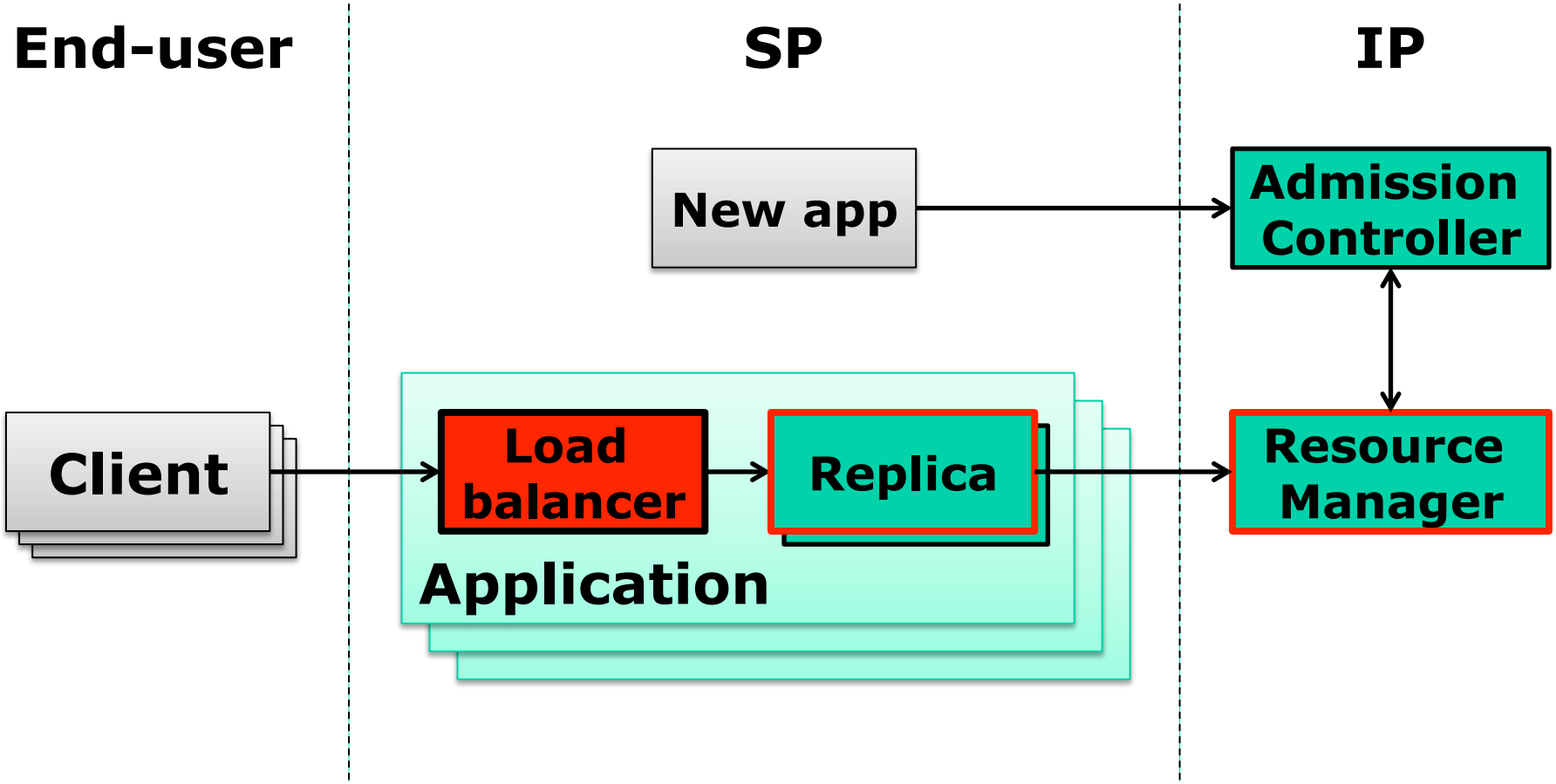
- Proven to **converge** and be **fair** using game theory

Results: 4 Applications



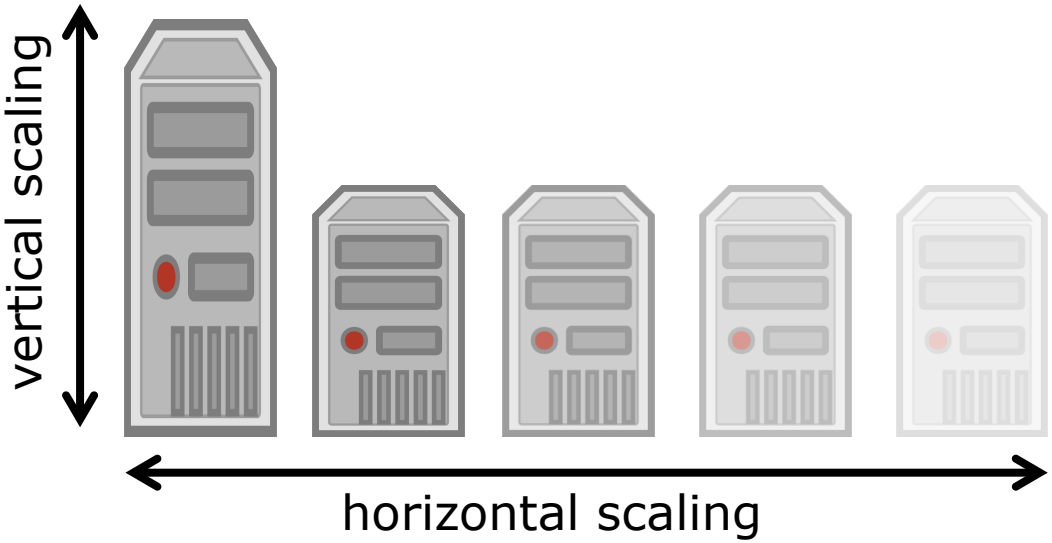
Applications that "struggle" get equal capacity
Other applications may run with maximum dimmer

Cloud Architecture



Why Replication?

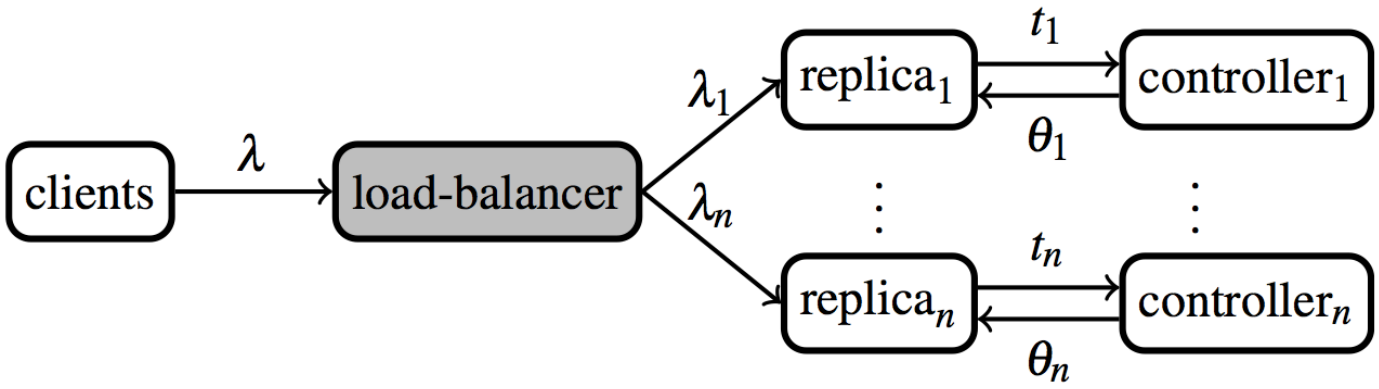
- Scale beyond a physical machine



- Resilience, **hide** infrastructure failures

Why Replication and Brownout?

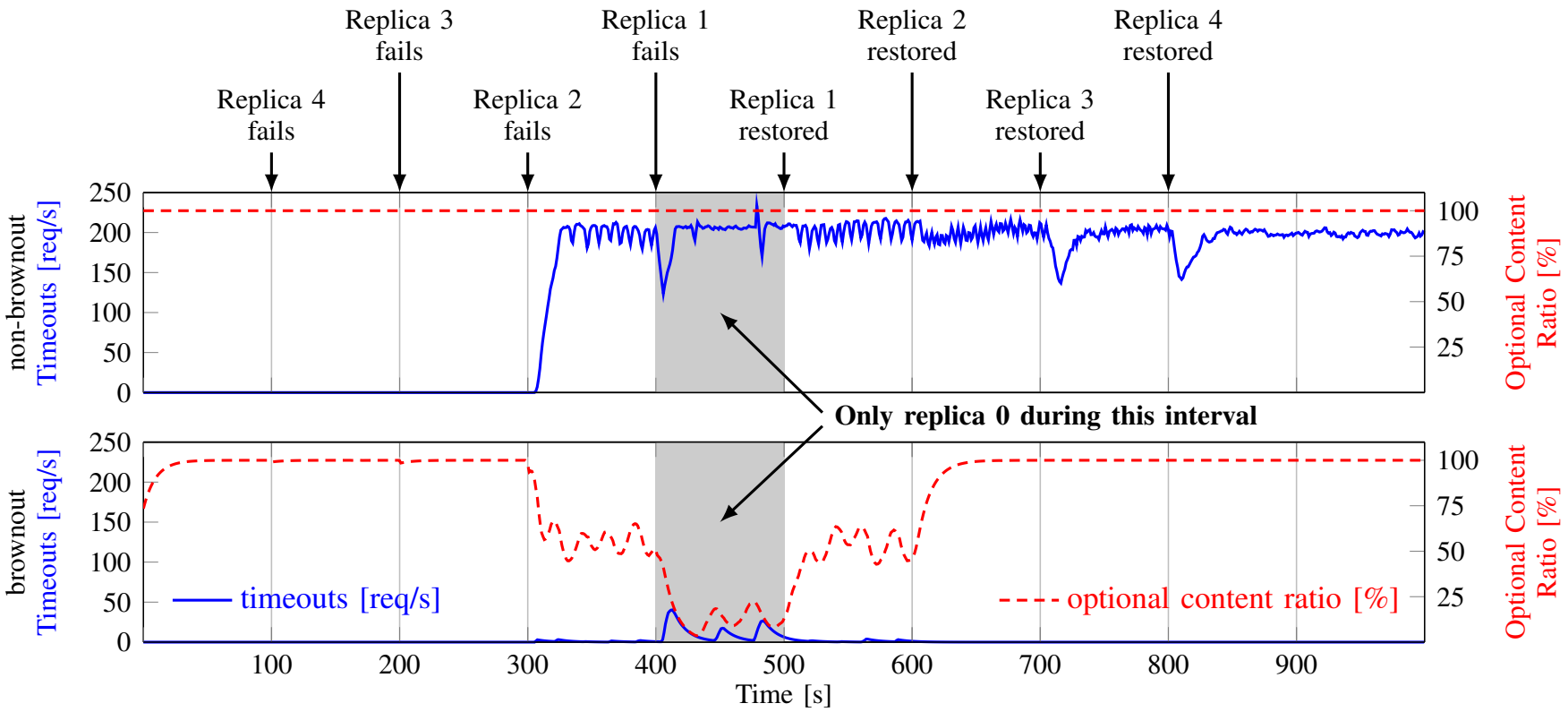
- Hide auto-scaling mishaps
- Hide failures leading to **capacity shortage**



- **Goal:** Maximize optional content served

Results: Replication and Brownout

Brownout-unaware load-balancing: shortest queue first



Maximizing Optional Content

	Weight-based (periodic)	Queue-based (event)
Brownout-unaware	WRR	SQF
Variation-based	**PIBH**	**PIBH+**
Equality-based	**EPBH**	**EPBH+**
Optimization-based	**COBLB**	

Tested using simulations

- SQF best brownout-unaware method
- Brownout-aware methods better
- Somewhat slow to react

Tested using experiments (lighttpd)

J. Dürango et al. "Control-theoretical load-balancing for cloud applications with brownout", (submitted to CDC 2014)
C. Klein et al. "Improving Cloud Service Resilience using Brownout-Aware Load-Balancing", (submitted to SRDS 2014)

Queue-Based Methods

- Track queue-lengths q_i and dimmers Θ_i
- Compute a queue offsets u_i
- Pick replica with lowest $q_i - u_i$

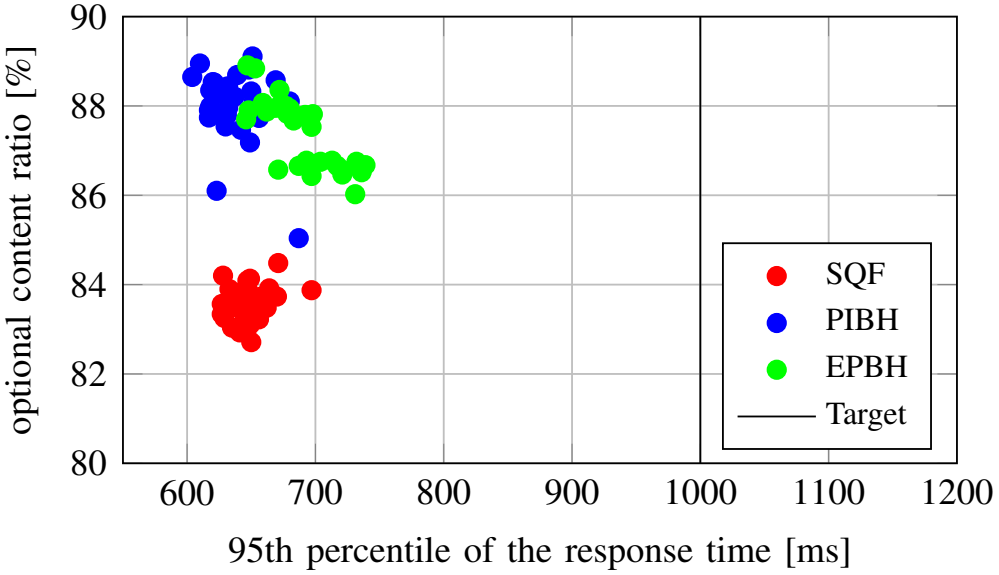
1. Variation-based:

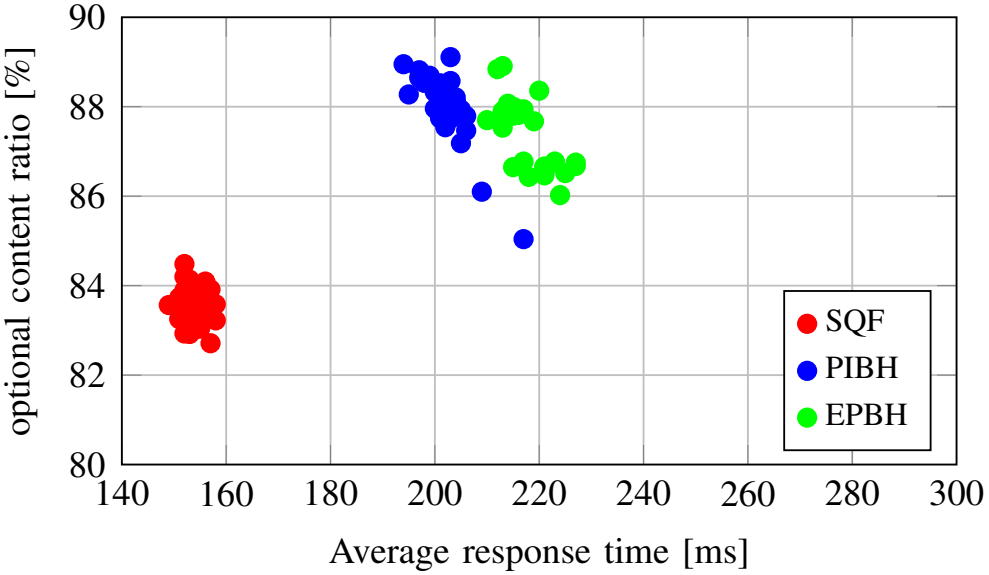
$$u_i(k+1) = (1-\gamma)\left[u_i(k) + \gamma_P \Delta\Theta_i(k) + \gamma_I \Theta_i(k)\right] + \gamma q_i(k)$$

2. Equality-based:

$$u_i(k+1) = u_i(k) + \gamma_e \left(\Theta_i(k) - \frac{1}{n} \sum_{j=1}^{n} \Theta_j(k) \right)$$

SQF vs. Brownout-Aware





Load-Balancing Recap

- Brownout improves resilience
  - SQF unexpectedly good
- Brownout-aware load-balancing methods
  - 5% more optional content

Conclusions

- Cloud applications need to tolerate
  - **Flash-crowds** (sudden increase in users)
  - Hardware **failures**
- Brownout
  - **Minimally intrusive** method
  - Cloud applications more **robust**

https://github.com/cloud-control

- Perspectives
  - **Improve** replica controller
    - Event-driven, queue-length-based
  - Combine brownout with **elasticity**

Acknowledgments

Umeå University


Erik Elmroth, PhD, Professor


Francisco Hernandez, PhD, Assistant Professor


Johan Tordsson, PhD, Assistant Professor


Luis Tomás, PhD, Post-doc

Lund University


Karl-Erik Årzén, PhD, Professor


Martina Maggio, PhD, Post-doc


Alessandro Vittorio Papadopoulos, PhD, Post-doc

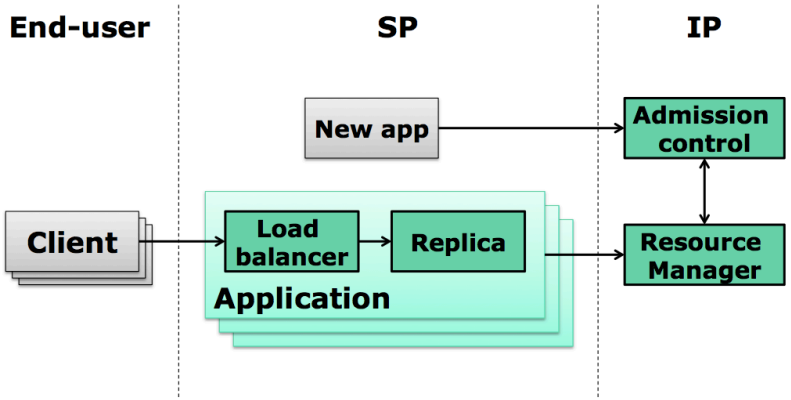

Manfred Dellkrantz, PhD Student

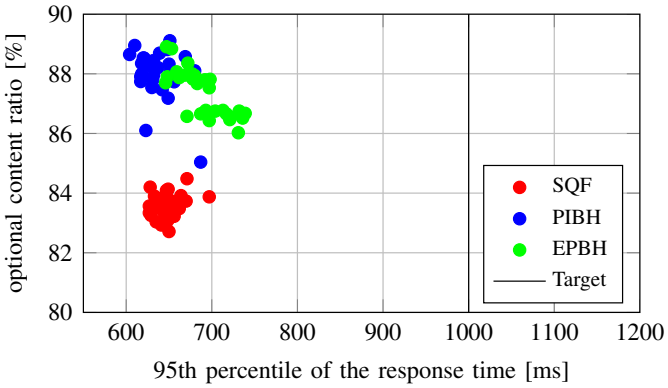

Jonas Dürango, PhD Student

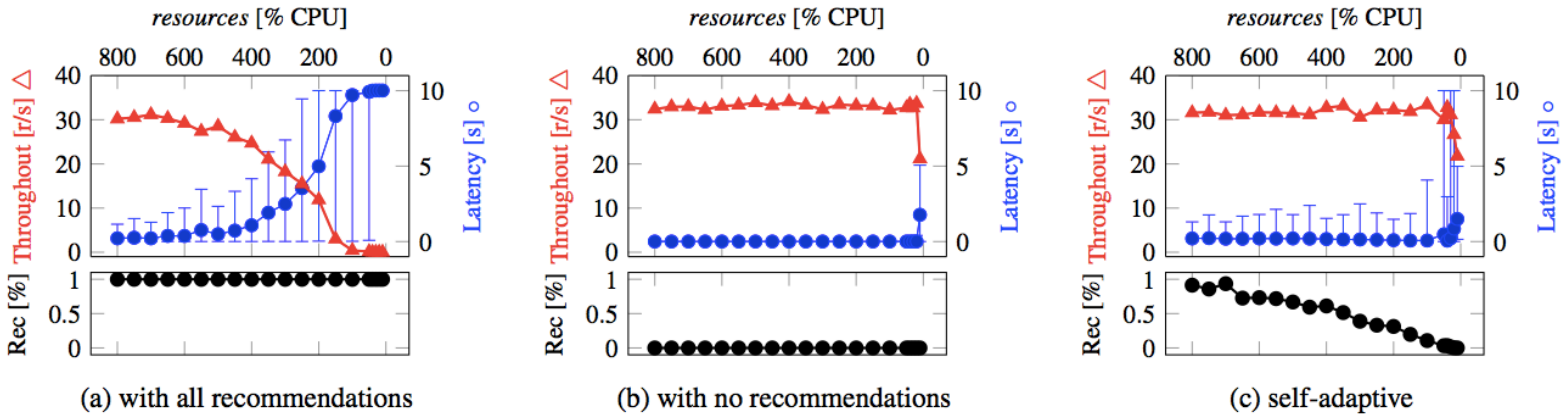
Thank you for your attention!

Cristian Klein

How I Learned to Stop Worrying
and Love Capacity Shortages







(a) with all recommendations (b) with no recommendations (c) self-adaptive

https://github.com/cloud-control

References

1. C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén, “Brownout: building more robust cloud applications”, ICSE, 2014
2. M. Maggio, C. Klein, K-E Årzén, “Control strategies for predictable brownouts in cloud computing”, IFAC, 2014
3. C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén, “Resource management for service level aware cloud applications”, REACTION, 2013
4. J. Dürango, M. Dellkrantz, M. Maggio, C. Klein, A. V. Papadopoulos, F. Hernández-Rodriguez, E. Elmroth, K-E Årzén, “Control-theoretical load-balancing for cloud applications with brownout”, (submitted to CDC 2014)
5. C. Klein, A. V. Papadopoulos, M. Dellkrantz, J. Dürango, M. Maggio, K-E Årzén, F. Hernández-Rodriguez, E. Elmroth, “Improving Cloud Service Resilience using Brownout-Aware Load-Balancing”, (submitted to SRDS 2014)
6. L. Tomás, C. Klein, J. Tordsson, F. Hernández-Rodriguez, “The straw that broke the camel's back: safe cloud overbooking with application brownout”, (submitted to CAC 2014)

Appendix

Cloud Computing

Results: 2 Applications