DISCRETE-TIME CONTROL THEORY FOR SOFTWARE ENGINEERING

From wikipedia:

Control Theory

"Control theory is an interdisciplinary branch of engineering and mathematics that deals with the behavior of dynamical systems with inputs, and how their behavior is modified by feedback."

CONTROL: PHYSICAL SYSTEMS

- Starting point: physical plant
- What can we measure?
- What can we change?
- What are the physical laws that govern the system's dynamic?

CONTROL: COMPUTING SYSTEMS

- What can we measure? What do we want to control?
- What can we change in the system?
- What is the physics? **There is no physics.**

Formal Guarantees

- Robustness to changes
- Even if the number of users increases, the latency experienced by these users should be bounded

Formal Guarantees

- Stability
    - Not only the latency should be bounded but it should also be a specific value

Formal Guarantees

- Transient behavior
- We would like to have an estimate of when we reach stability and before that we would like to be sure that we don’t overdo

Applying The Theory

CONTROLLING THE CLOUD

BROWNOUT

BROWNOUT

A brownout is an intentional or unintentional drop in voltage in an electrical power supply system. Intentional brownouts are used for load reduction in emergency conditions.

BROWNOUT

A brownout is an intentional or unintentional drop in voltage in an electrical power supply system. Intentional brownouts are used for load reduction in emergency conditions.

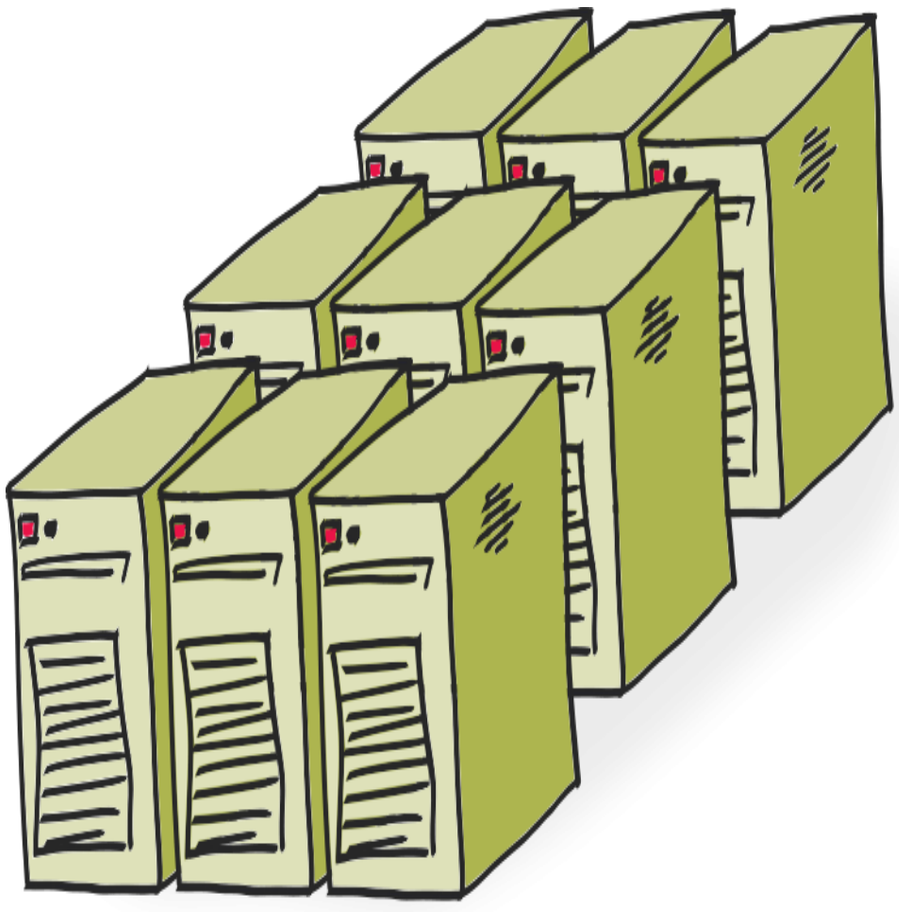
Tokyo Tower Brownout

PREDICTABILITY: MOTIVATION

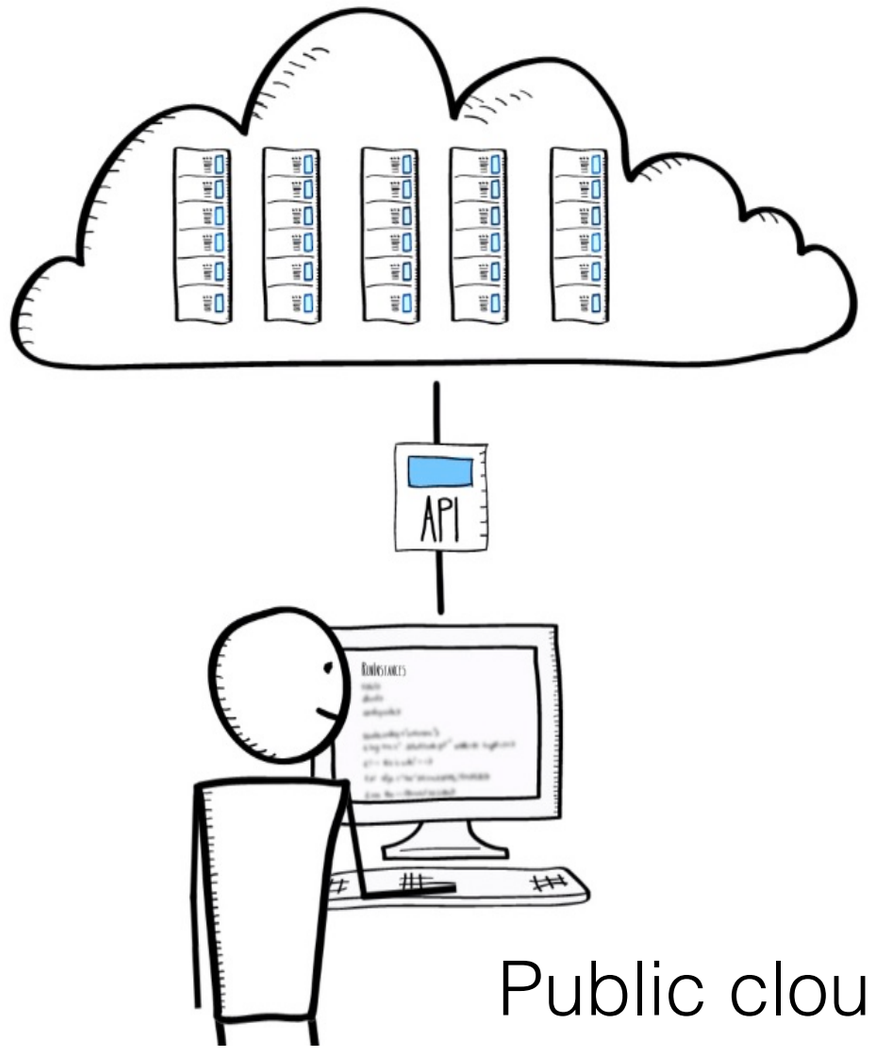
- ✓ 82% of end-users give up on a lost payment transaction*
- ✓ 25% of end-users leave if load time > 4s**
- ✓ 1% reduced sale per 100ms load time**
- ✓ 20% reduced income if 0.5s longer load time***

*JupiterResearch, **Amazon, ***Google

STATE OF THE ART



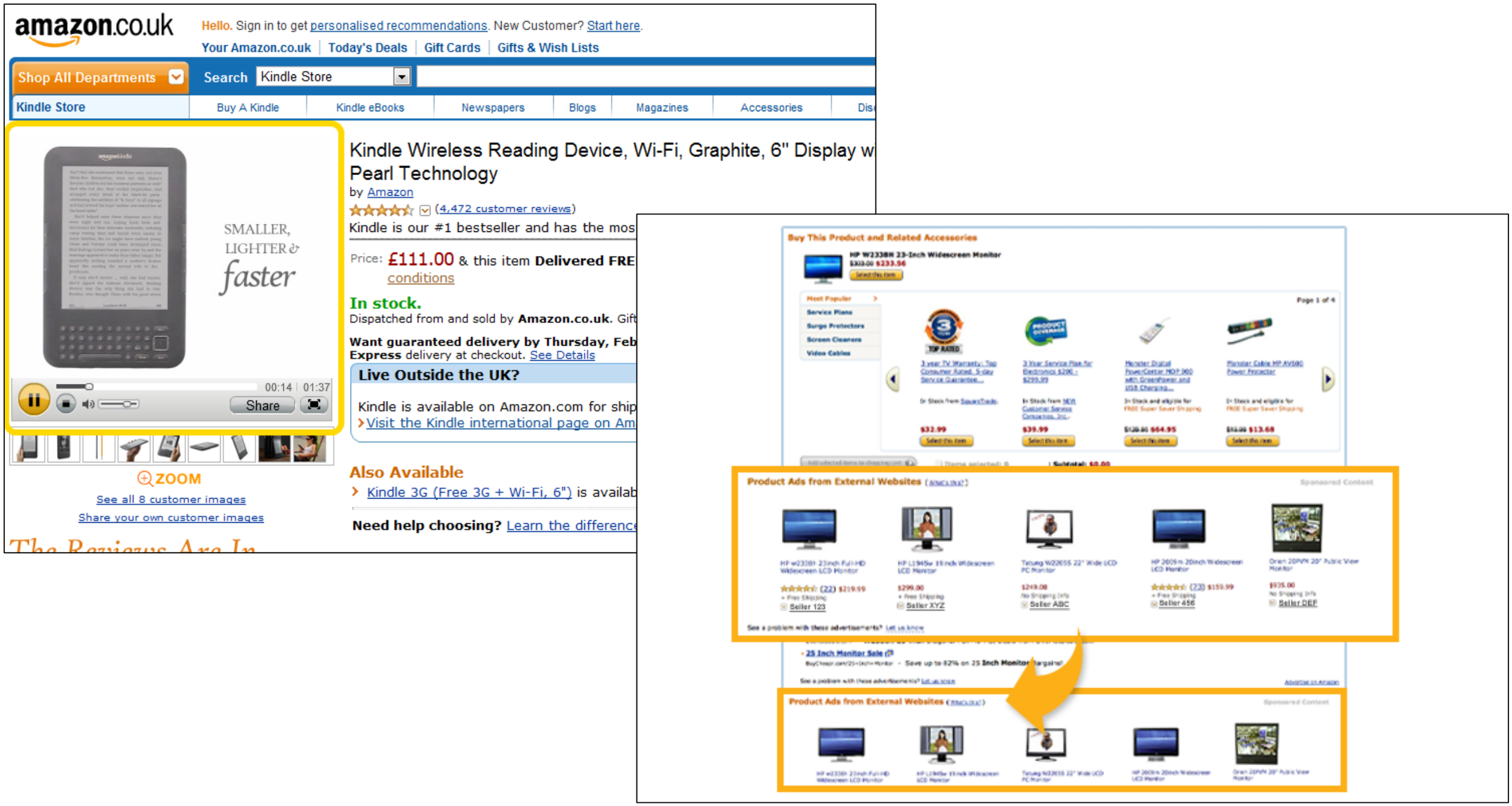
Increase capacity



Public cloud

BROWNOUT

Disable non-essential content on demand



BROWNOUT

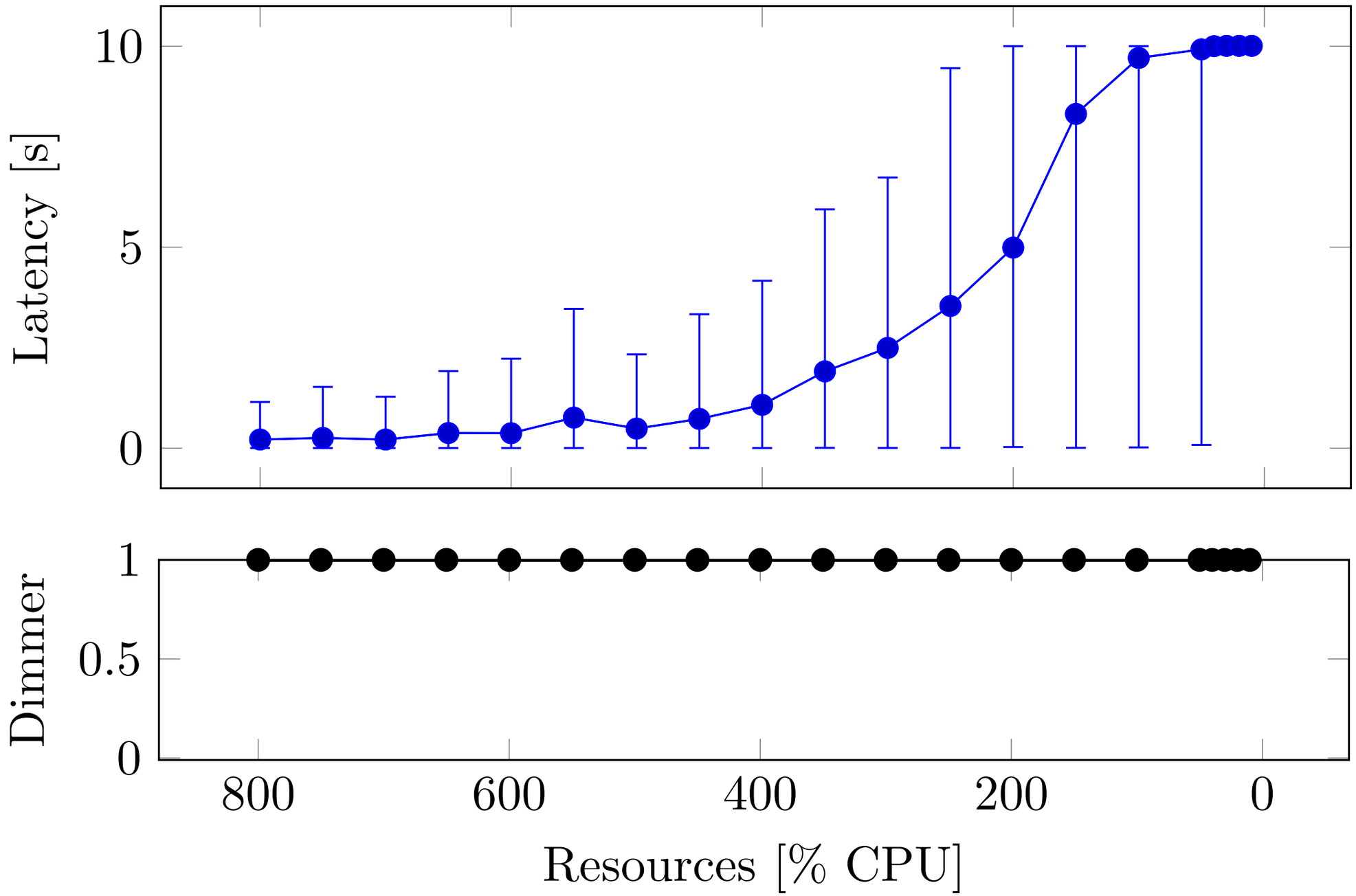
- Single replica
- Graceful degradation

➡ Minimally intrusive (<200 lines of code)

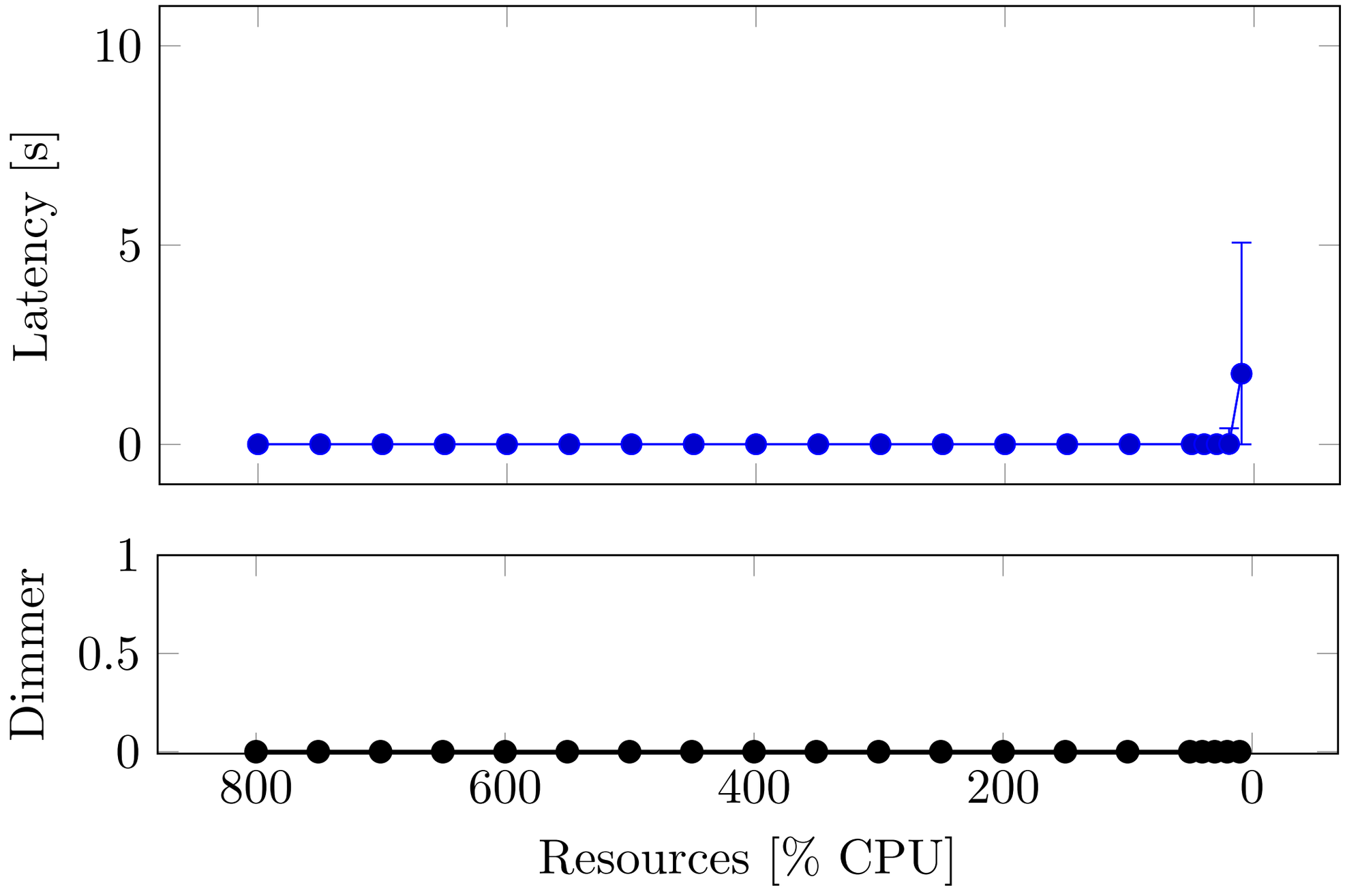
➡ Application developers **mark optional code**

➡ **Automatic control strategy** to select when the optional computations should be turned on and off

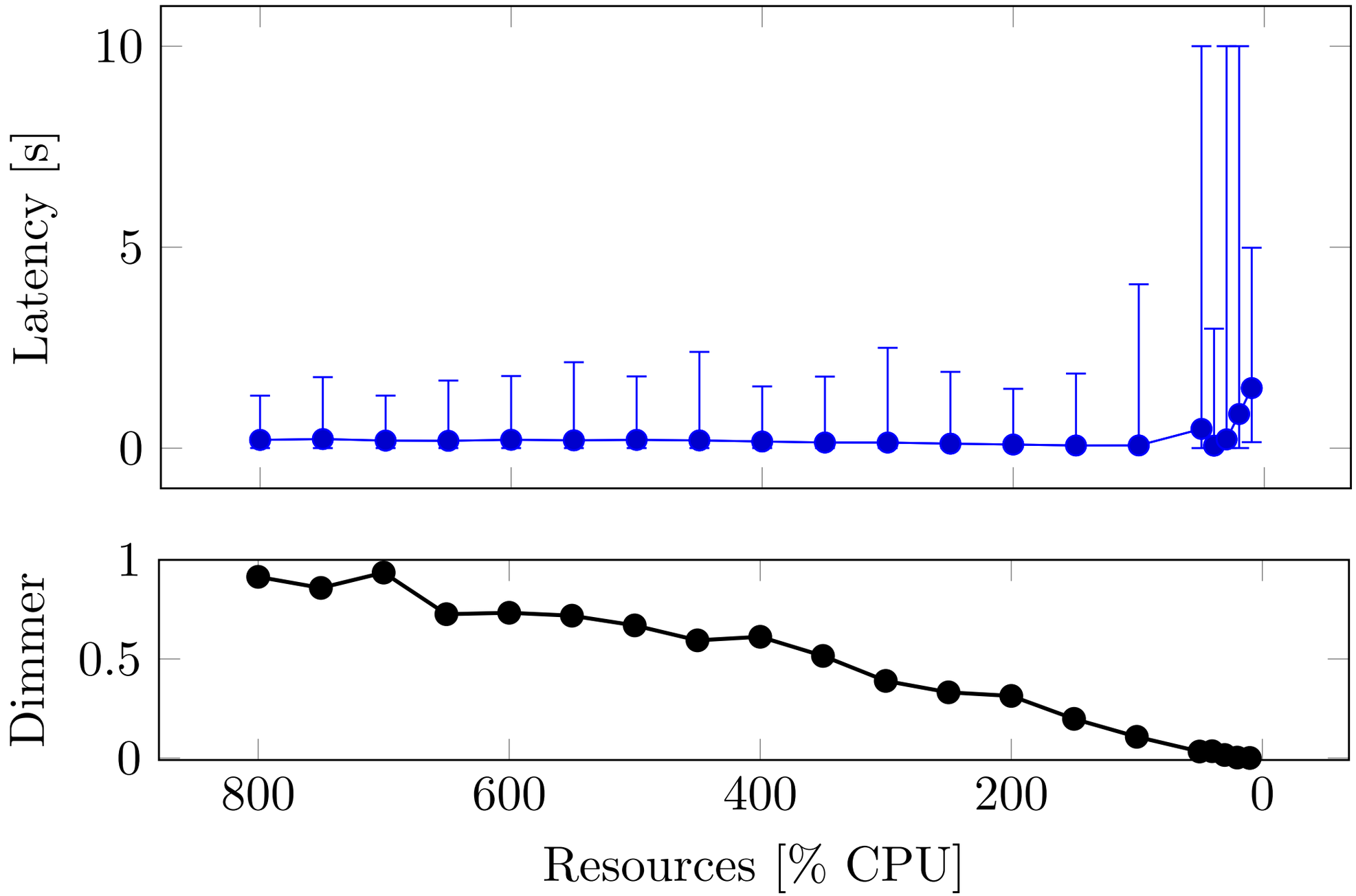
EXAMPLE: E-COMMERCE WEBSITE



EXAMPLE: E-COMMERCE WEBSITE



EXAMPLE: E-COMMERCE WEBSITE



AUTOMATIC CONTROL STRATEGY

- Compute **statistics** from response times
- Use statistics as a **feedback** signal
- Update the current model of the system

➡ Control strategy:
selects the probability
of executing the optional
code (dimmer)

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

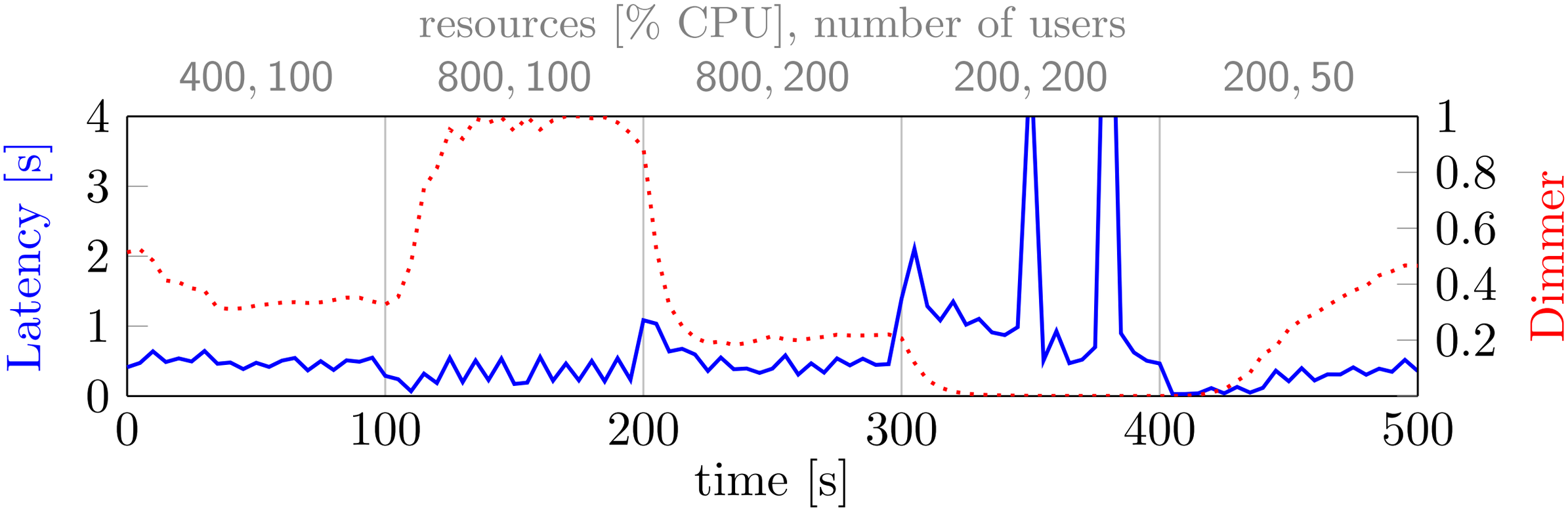
AUTOMATIC CONTROL STRATEGY

- Implemented and tested:
  - Adaptive PI controller
  - Adaptive PID controller
  - Deadbeat controller
  - Feedforward plus feedback controller

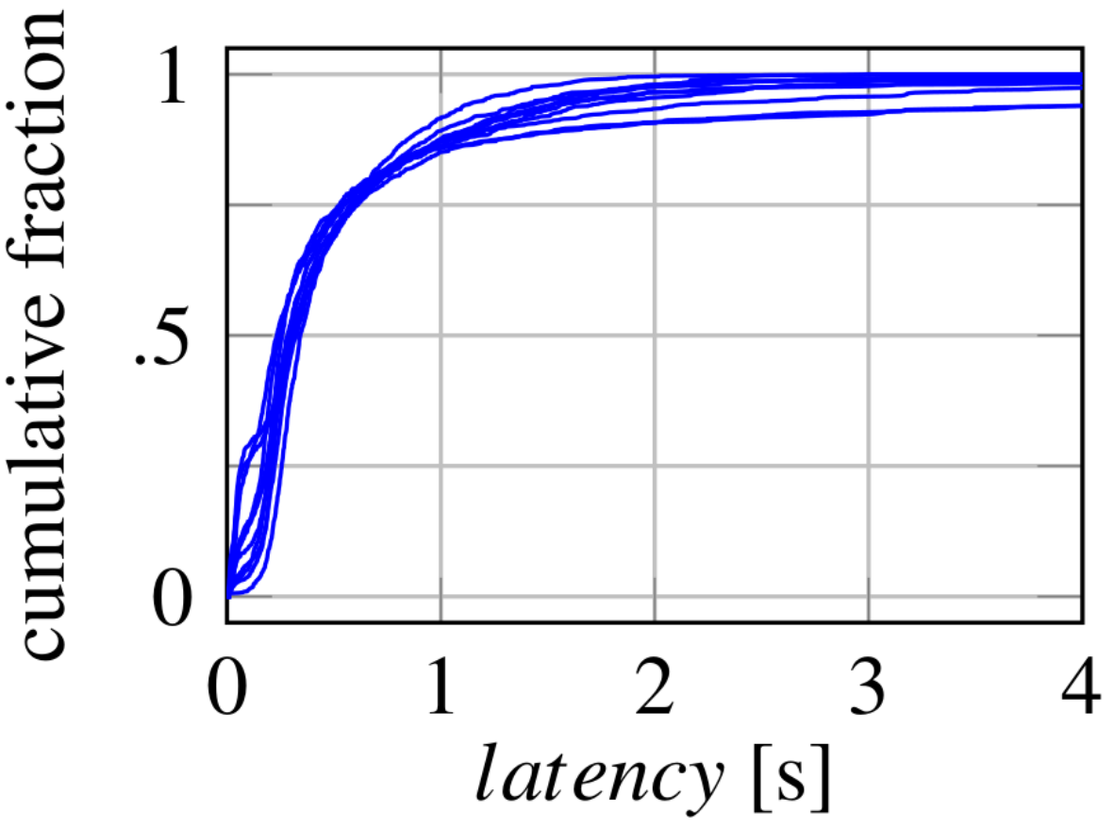
"Control strategies for predictable brownouts in cloud computing"
M. Maggio, C. Klein, K.-E. Årzén, IFAC WC 2014

EXPERIMENTAL VALIDATION

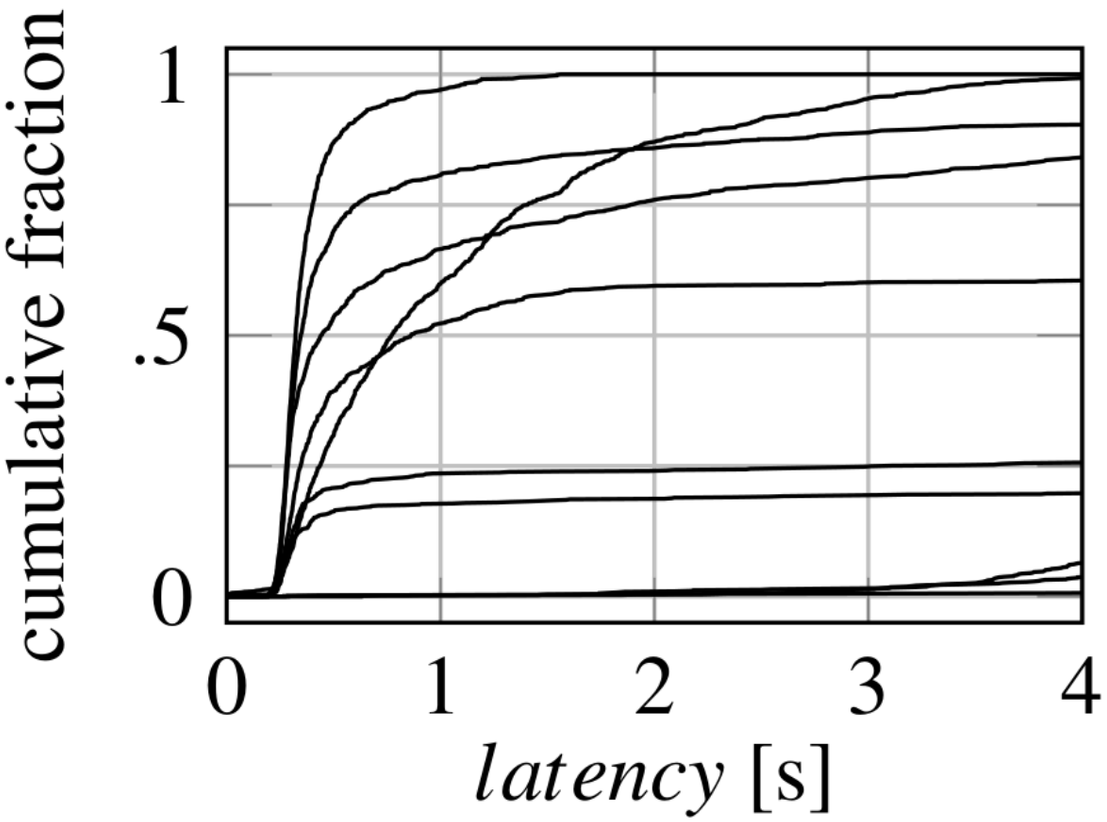
Experiment with RUBBoS and varying users/resources



STATISTICAL VALIDATION



With control

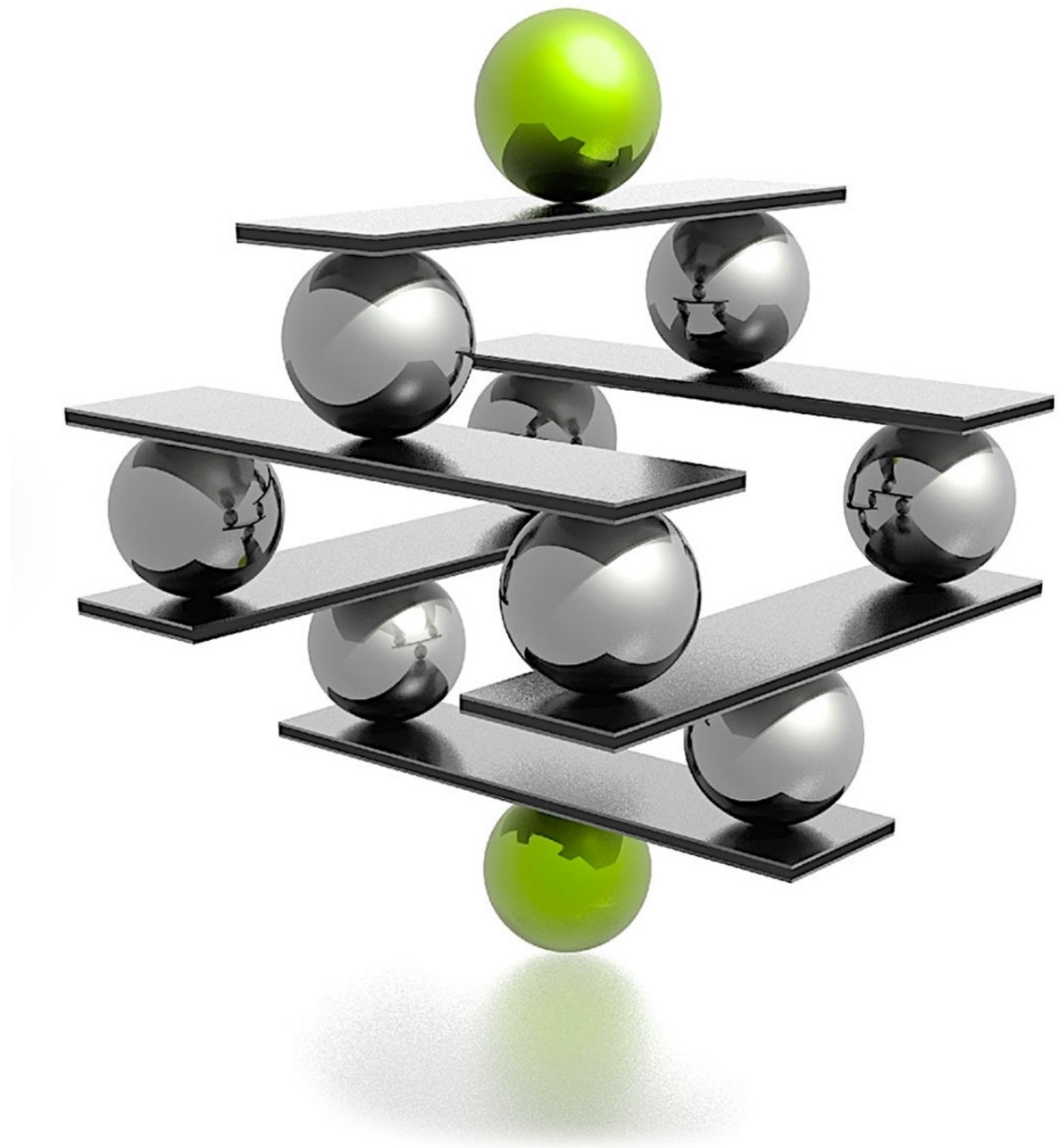


Without control

SCALING UP

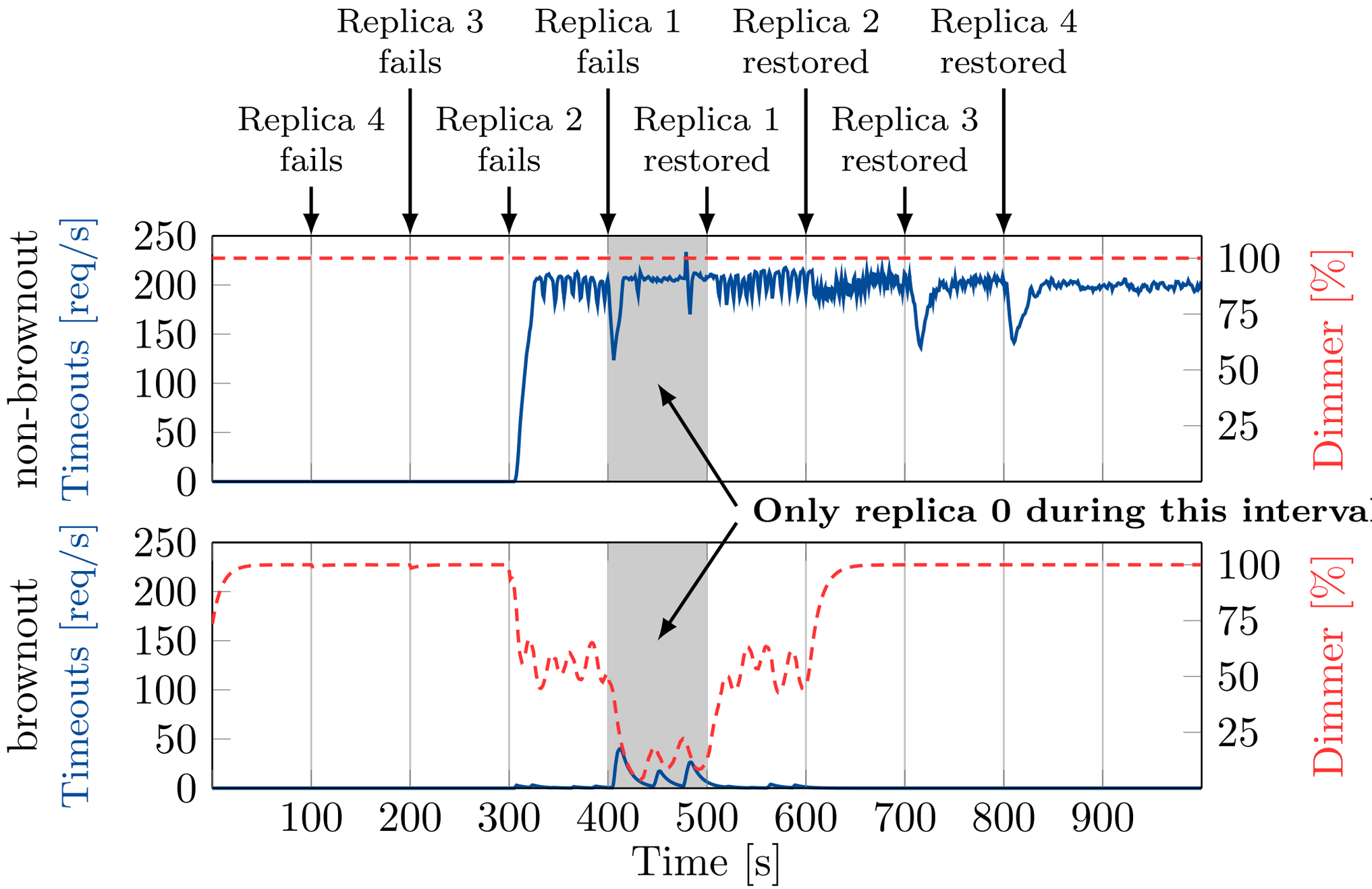
- Multiple replicas

- Needs load balancing:
  - Based on latencies
  - SQF
  - Brownout-aware



"Control-theoretical load-balancing for cloud applications with brownout"
J. Dürango, M. Dellkrantz, M. Maggio, C. Klein, A.V. Papadopoulos, F. Hernández-Rodriguez, E. Elmroth, K.-E. Årzén, CDC 2014

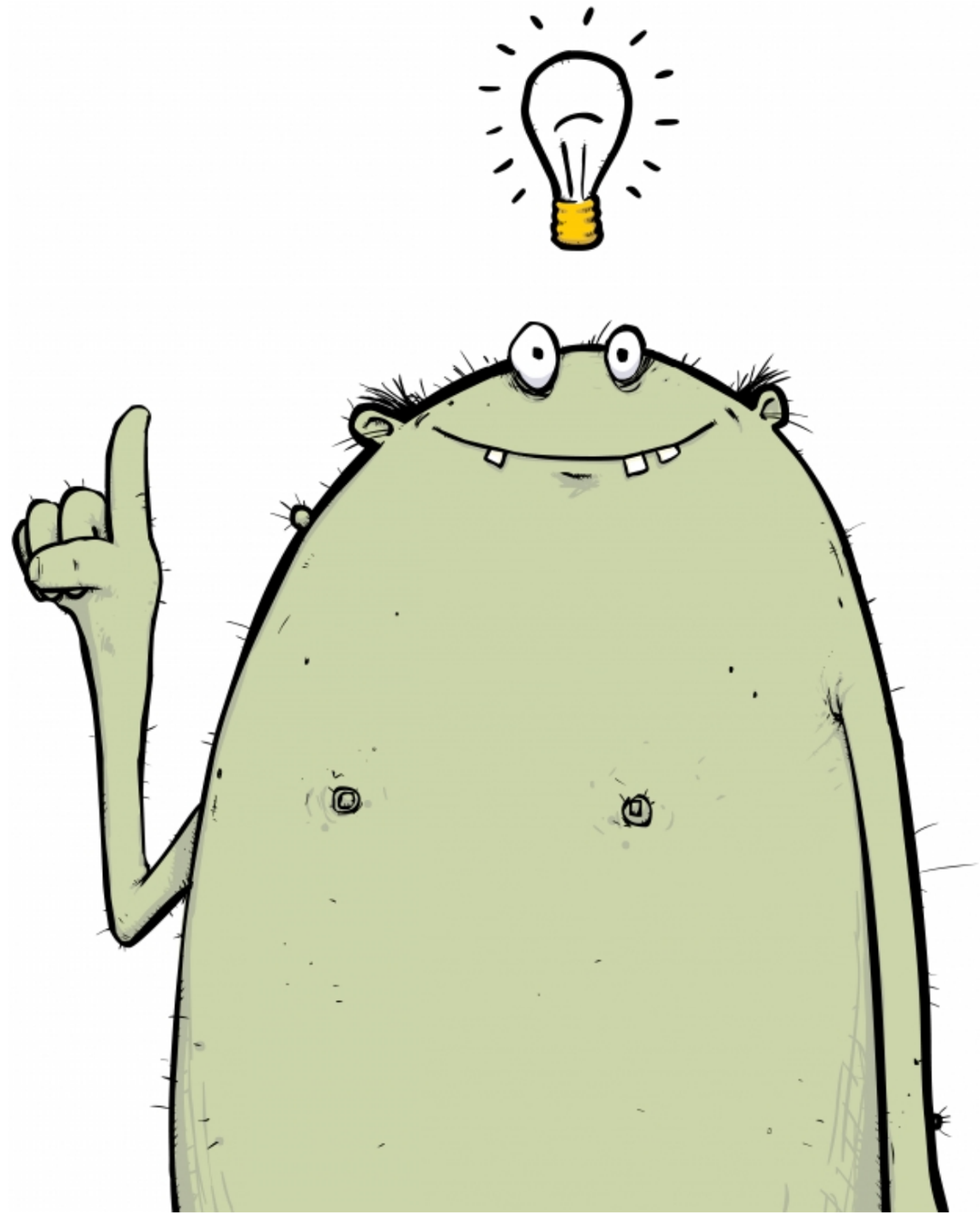
IMPROVING FAULT-TOLERANCE



"Improving Cloud Service Resilience using Brownout-Aware Load-Balancing"
C. Klein, A.V. Papadopoulos, M. Dellkrantz, J. Dürango, M. Maggio, K.-E. Årzén, F. Hernández-Rodriguez, and E. Elmroth, SRDS 2014

CONCLUSION

- Bounded response times and improved fault tolerance
- Formal guarantees
- Minimally intrusive
- Application developer should only **mark optional computations**



https://github.com/cloud-control/

Thanks for the attention!