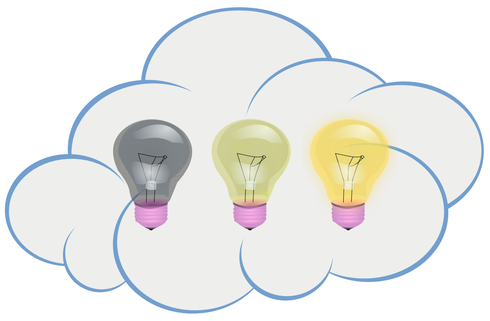
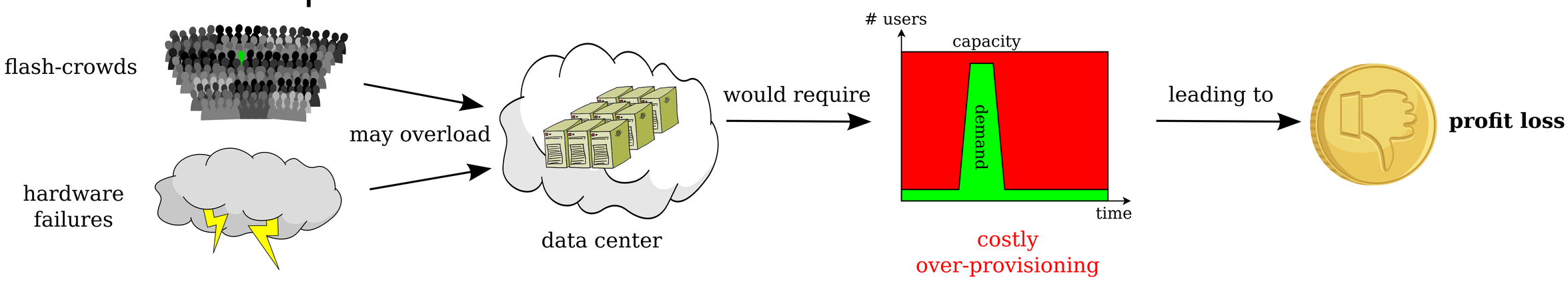


Introducing Service-level Awareness in Clouds

Cristian Klein[1], Martina Maggio[2], Karl-Erik Årzén[2], Francisco Hernández-Rodriguez[1]
[1]Umeå University, Sweden, [2]Lund University, Sweden
cristian.klein@cs.umu.se, martina.maggio@control.lth.se, karlerik@control.lth.se, francisco@cs.umu.se

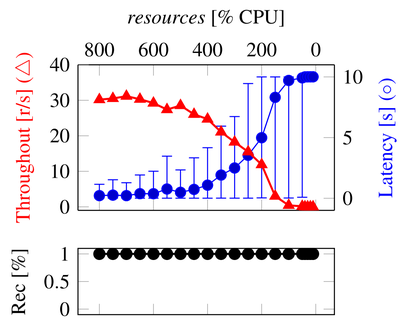
https://github.com/cristiklein/cloudish

Problem: unexpected events

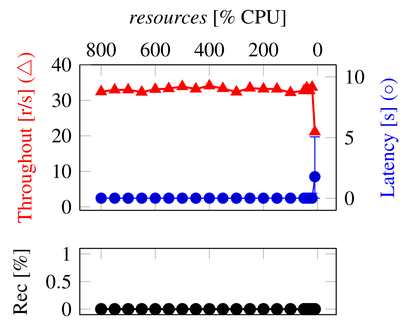


Idea: reduce **service-level** to avoid data center overload

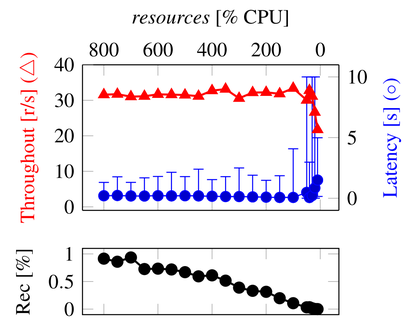
Downgrade **user experience** to reduce capacity requirements and serve more users



RUBiS with full recommendations

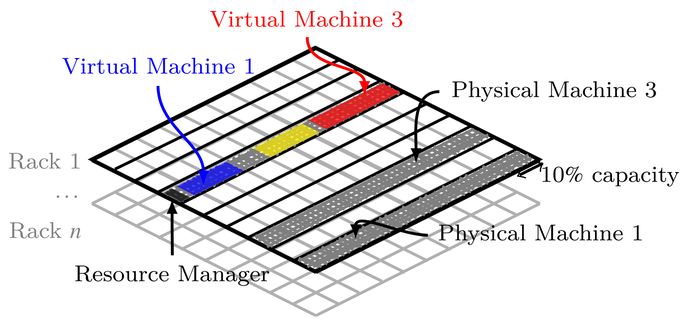


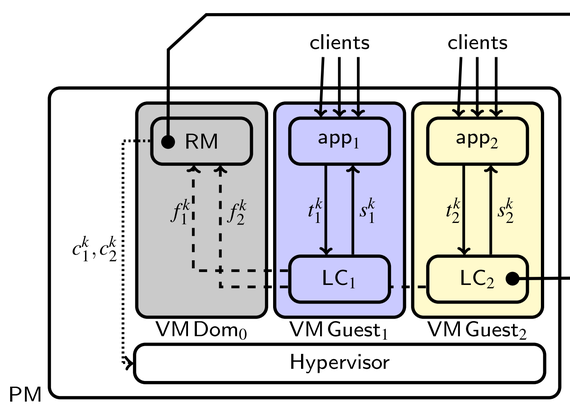
RUBiS with no recommendations



RUBiS with adaptive recommendations

Our Proposal





Resource Manager (RM): designed using **game theory**

$$c_i^{k+1} = c_i^k - \varepsilon_{rm}(f_i^k - c_i^k \cdot \sum_i f_i^k)$$

f_i^k: matching value sent by app. i at time index k

c_i^{k+1}: capacity allocation of app. i at time index $k+1$

Local Controller (LC): designed using **control theory**

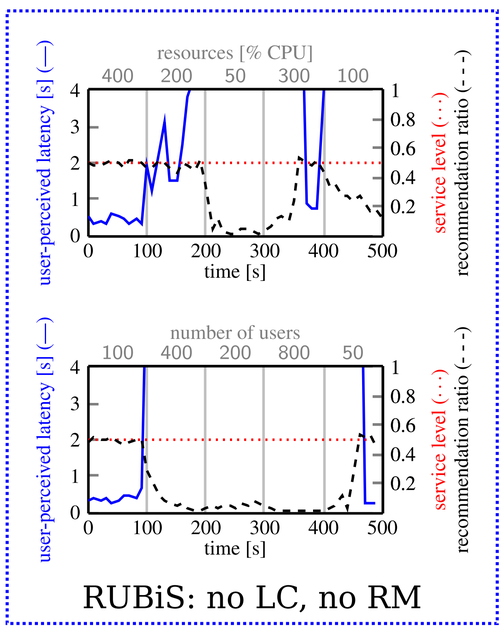
$$s_i^{k+1} = s_i^k + \frac{1-p_1}{\alpha} \cdot (t_i^k - \bar{t}_i) \qquad f_i^k = 1 - \frac{t_i^k}{\bar{t}_i}$$

s_i^{k+1}: service-level

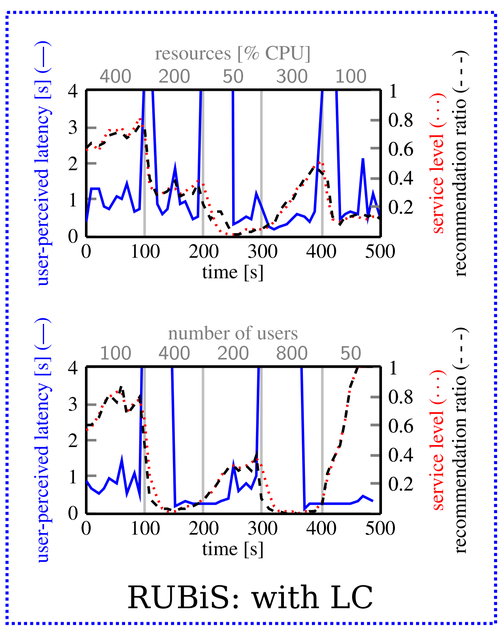
t_i^k: measured latency

$\bar{t}_i$: target latency

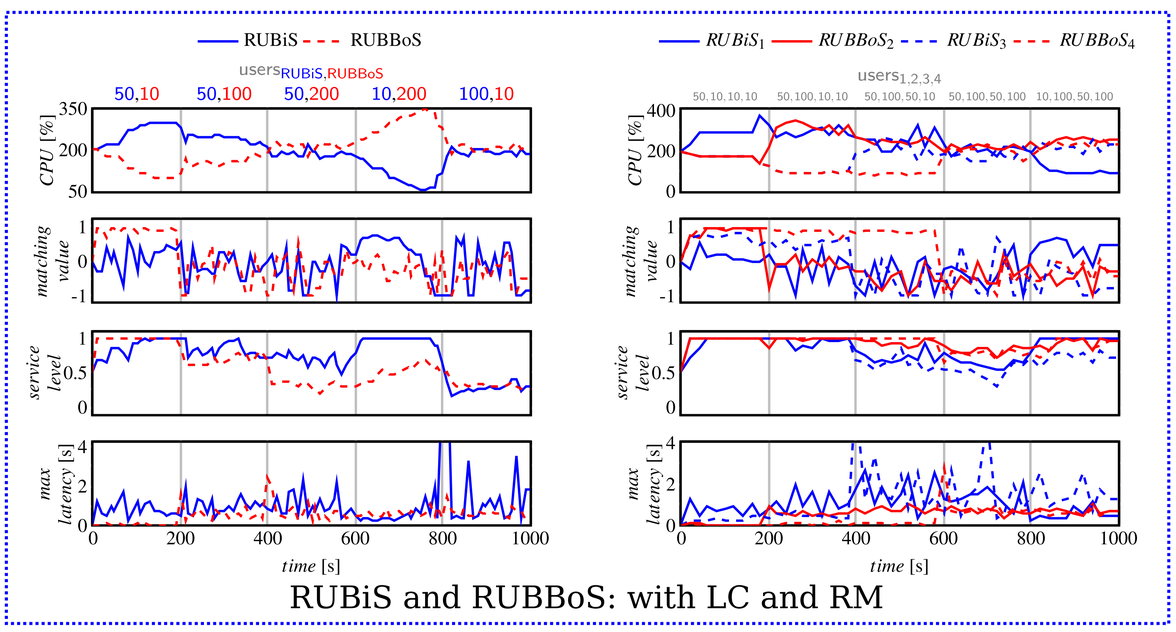
Results



RUBiS: no LC, no RM



RUBiS: with LC



RUBiS and RUBBoS: with LC and RM

Conclusions

- clouds can more robustly **withstand capacity shortages**
  - flash-crowds
  - hardware failures
- applications can reduce their service level (turn off optional features)
- infrastructure can rebalance resources among hosted applications

Perspectives

- combining with other mechanism
  - migration
  - horizontal scaling
  - overbooking
- devising a billing model

Vetenskapsrådet
SEVENTH FRAMEWORK PROGRAMME