A Receding Horizon Approach for the Runtime Management of IaaS Cloud Systems

www.modaclouds.eu

Danilo Ardagna, Michele Ciavotta
{danilo.ardagna, michele.ciavotta}@polimi.it
Politecnico di Milano
Riccardo Lancellotti
riccardo.lancellotti@unimore.it
Università di Modena e Reggio Emilia

Agenda

* Introduction
* Problem
  * Problem statement and design assumption
  * Receding Horizon algorithm
* Experimental Analysis
* Conclusions

MODAClouds

Introduction

The advent of Cloud Computing changed dramatically the ICT industry

- Google, Amazon, Microsoft, Salesforce, Oracle, SAP, SoftLayer, Rackspace etc…
- Cost-effective solutions
- Computational power
- Reliability
- Auto-scaling

New business paradigms appeared on the market

- IaaS, PaaS, SaaS
- But also DaaS, BDaaS, HDaaS, etc…

Introduction: challenges

The growing popularity of Cloud Computing opens new challenges

- Vendor lock-in
- Design for Quality of Service (QoS) guarantees
- Managing the lifecycle of a Cloud application
- Managing Elasticity
  - Resource Provisioning
  - Self-adaptation

Introduction: resource provisioning

Resource Provisioning: mechanism for leasing and releasing virtual cloud resources to guarantee adequate QoS

… it requires management solutions that support

* Performance prediction,
* Monitoring of Service Level Agreements (SLA),
* Adaptive re-configuration actions.

Tools currently supplied by IaaS providers, are often too basic and inadequate for

* Highly variable workload,
* Applications with a dynamic behavior characterized by uncertainty.

Introduction: our approach

Proposal: a fast and effective Capacity Allocation technique

* based on the Receding Horizon control strategy
* integrated within MODAClouds runtime platform
* that minimizes the execution costs of a Cloud application,
* guaranteeing QoS constraints expressed in terms of average response time

Agenda

* Introduction
* Problem
  * Problem statement and design assumption
  * Receding Horizon algorithm
* Experimental Analysis
* Conclusions

MODAClouds

Problem: design assumptions

Perspective of a **Software-as-a-Service (SaaS)** provider hosting his/her applications on an **Infrastructure-as-a-Service (IaaS)** provider

Applications are single **tier** hosted in virtual machines (VMs) that are dynamically instantiated by the IaaS provider

Each VM hosts a single **WS application**

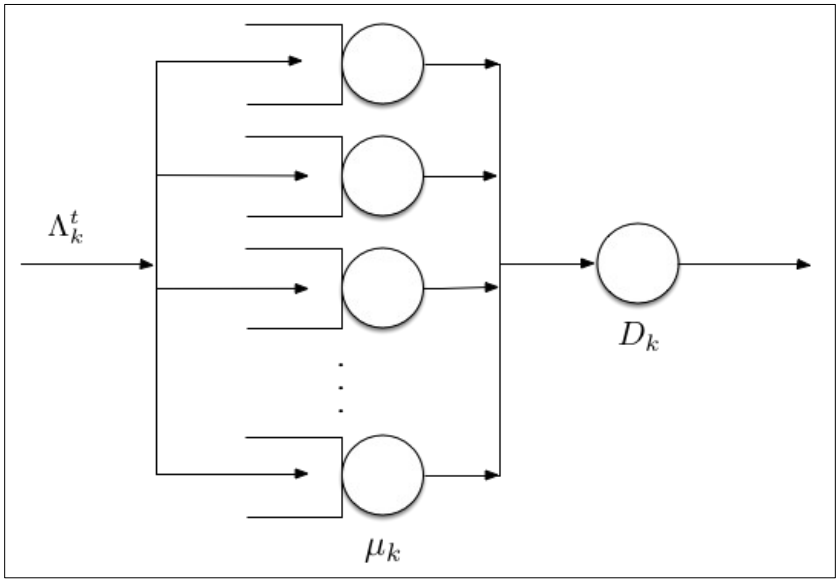
Multiple **homogeneous** VMs implementing the same WS application can run in parallel

MODAClouds

Problem: design assumptions

Each **WS class** hosted in a VM is modeled as an **M/G/1 queue** in tandem with a delay center

SLA based on the average response time R_k: every WS class has to provide a response time lower than a threshold $\overline{R_k}$



MODAClouds

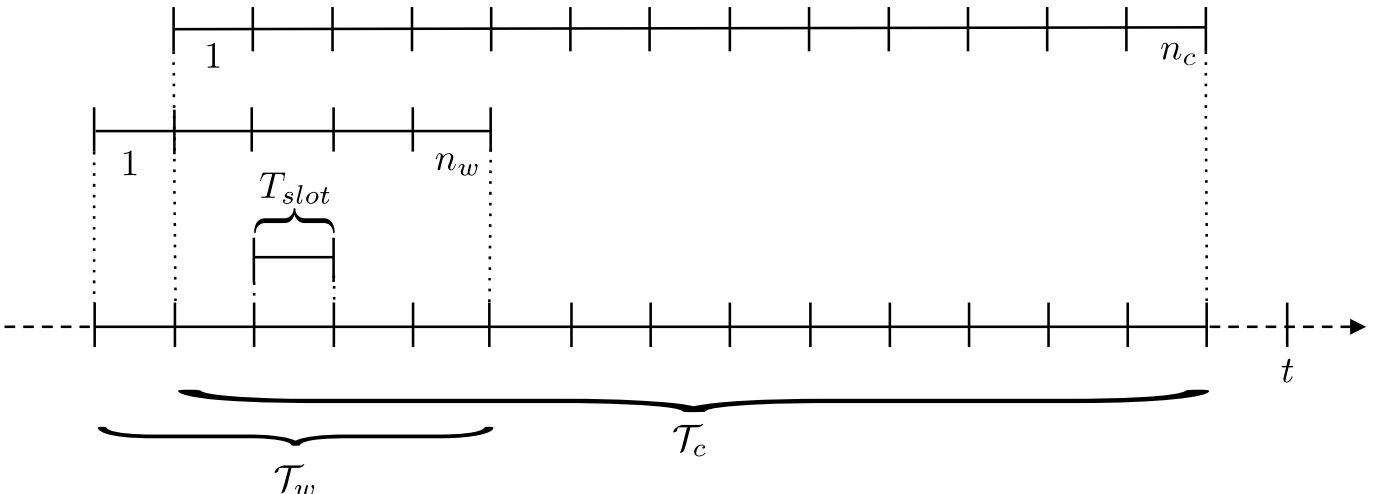
Problem: design assumptions

IaaS providers **charge** software providers on an **hourly basis**

* *reserved* VMs (ρ time-unit cost)
* *on demand* VMs (δ time-unit cost $\rho < \delta$)

Time management:

* Time slots: T_{slot} (5, 10 min)
* Time window: T_w (1-5 T_{slot})
* Charging interval: T_c (60 min)



MODAClouds

Problem: formulation

Time unit costs

Time management

Freely available VMs

Workload prediction

CA plan

System parameters

$\mathcal{K}$	Set of WS applications
δ	Time unit cost (measured in dollars) for *on-demand* VMs
ρ	Time unit cost (measured in dollars) for *reserved* VMs
$\mathcal{T}_w$	Set of time slots within the sliding time window
$\mathcal{T}_c$	Set of time slots within a charging interval
T_{slot}	Short-term CA time slot, measured in minutes
n_c	Number of time slots within the charging interval $\mathcal{T}_c$
n_w	Number of time slots within the time window $\mathcal{T}_w$
$\overline{r}_k^t$	Number of *reserved* VMs freely available at time slot t in the interval under analysis, for request class k
$\overline{d}_k^t$	Number of *on-demand* VMs available for free at time slot t in the interval under analysis, for request class k
Λ_k^t	Real local arrival rate (measured in requests/sec) for request class k, at time slot t
$\widehat{\Lambda}_k^t$	Local arrival rate prediction (measured in requests/sec) for request class k, at time slot t
$\overline{R}_k$	Average response time threshold for request class k
W	Maximum number of *reserved* instances available

Decision Variables

d_k^t	Number of *on-demand* VMs to be allocated for request class k at time slot t
r_k^t	Number of *reserved* VMs to be allocated for request class k at time slot t

MODAClouds

Problem: formulation

The CA problem can be formulated as:

$$(\mathrm{P}) \qquad \min_{r_k^t, d_k^t} \sum_{k \in \mathcal{K}} \left(\rho \sum_{t=1}^{n_w} r_k^t + \delta \sum_{t=1}^{n_w} d_k^t \right)$$

Total cost

Subject to the conditions:

$$R_k(r_k^1, \overline{r}_k^1, \ldots, r_k^t, \overline{r}_k^t, d_k^1, \overline{d}_k^1, \ldots, d_k^t, \overline{d}_k^t, \widehat{\Lambda}_k^1, \ldots, \widehat{\Lambda}_k^t) \leq \overline{R}_k$$

$$\forall k \in \mathcal{K}, \ \forall t \in \mathcal{T}_w$$

Response time

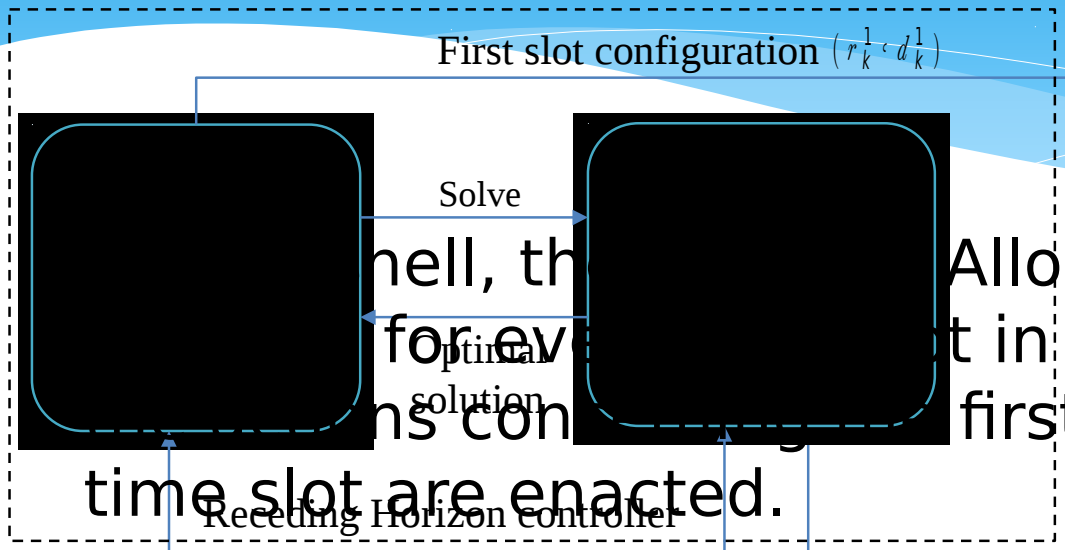
$$\sum_{k \in \mathcal{K}} (r_k^t + \overline{r}_k^t) \leq W, \ \forall t \in \mathcal{T}_w$$

limited number of reserved VMs

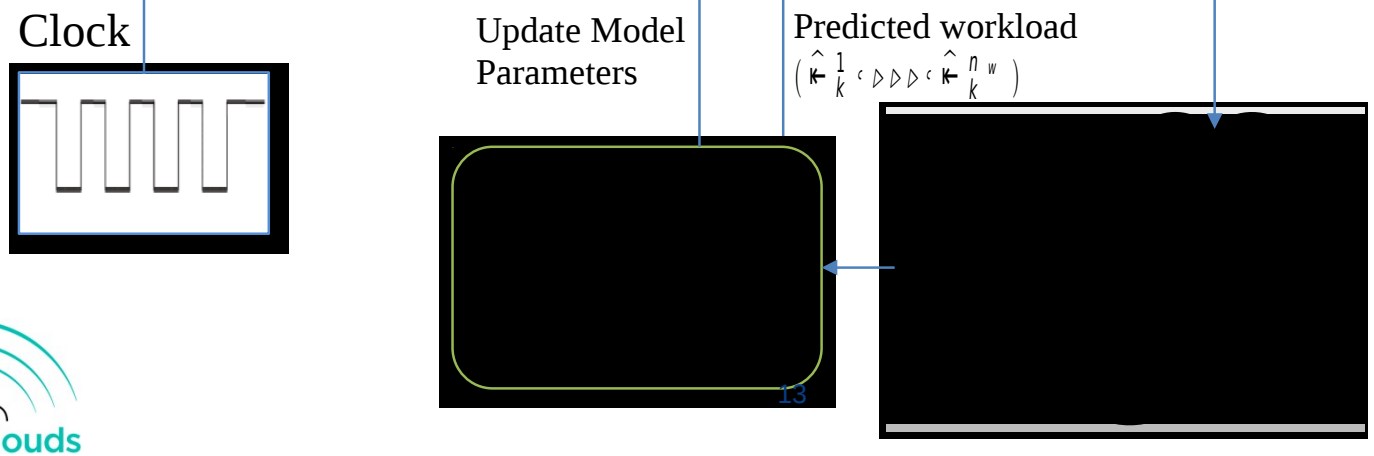
$$r_k^t \geq 0, \ r_k^t \in \mathbb{N} \quad \forall k \in \mathcal{K}, \ \forall t \in \mathcal{T}_w$$

$$d_k^t \geq 0, \ d_k^t \in \mathbb{N} \quad \forall k \in \mathcal{K}, \ \forall t \in \mathcal{T}_w$$

Receding Horizon Algorithm



hell, th Allocation problem
for ev t in but only
ns con first forthcoming
time slot are enacted.



MODAClouds

Receding Horizon Algorithm

Algorithm 1 Receding Horizon Algorithm

1: **procedure** SOLUTION ALGORITHM
2: **for all** $k \in \mathcal{K}$ **do**
3: **for** $w \leftarrow 1, n_w$ **do**
4: $\widehat{\Lambda}_k^w \leftarrow$ ***GetPrediction*** (w, k)
5: $\overline{r}_k^w \leftarrow N_{res,k}^{t+w}$
6: $\overline{d}_k^w \leftarrow N_{ond,k}^{t+w}$
7: **end for**
8: **end for**
9: ***Solve*** $(P, \overline{r}, \overline{d}, \widehat{\Lambda}) \leftarrow$
10: **for all** $k \in \mathcal{K}$ **do**
11: ***Scale*** $(k, r_k^1, d_k^1) \leftarrow$
12: **for** $j \leftarrow 1, n_c$ **do**
13: $N_{res,k}^{t+j} \leftarrow N_{res,k}^{t+j} + r_k^1$
14: $N_{ond,k}^{t+j} \leftarrow N_{ond,k}^{t+j} + d_k^1$
15: **end for**
16: **end for**
17: **end procedure**

Initialization

Solving the current model

Applying the changes according to the first time slot decisions

State update

MODAClouds

Agenda

* Introduction
* Problem
  * Problem statement and design assumption
  * Receding Horizon algorithm
* Experimental Analysis
* Conclusions

Experimental Analysis

Scalability:

* Large set of randomly generated instances
* Daily distribution of requests from real log traces

Comparison with state of the art approaches:

* Heuristic
* Oracle with perfect knowledge of the future

Time scale analysis:

* SLA violations

Experiment Design

Workload prediction

- Incoming workload has been obtained for traces of a very large dynamic web-based system
- Different workload for each WS class
- Prediction obtained by adding white noise to each sample
- Noise proportional to the arrival rate
- Inaccuracy increases with the time slot

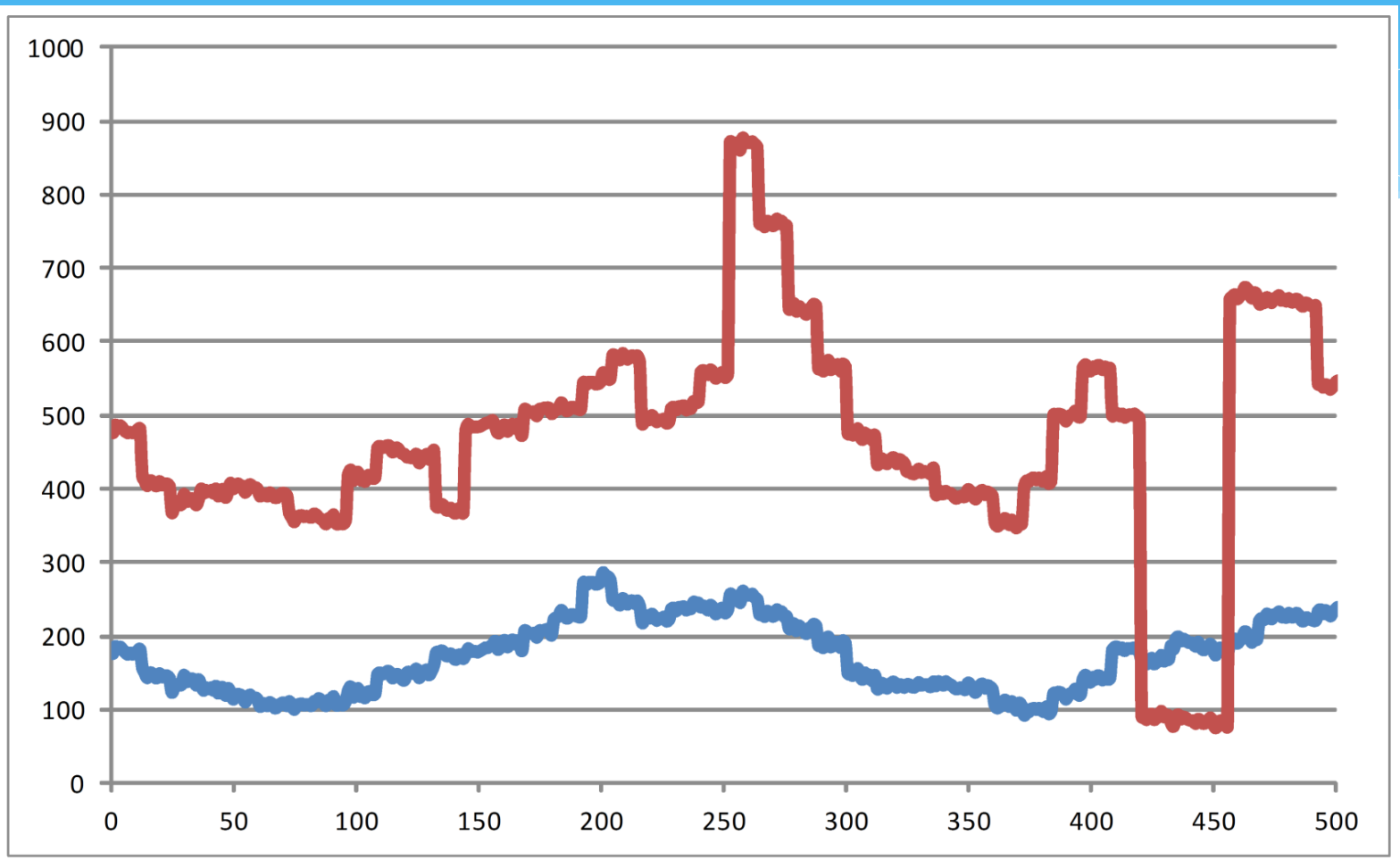
Performance parameters

- Service rate $\mu_k \in [200, 400] \quad req/sec$
- Queueing delay $D_k \in [0.001, 0.05] s$
- Reserved instances $W = 10$

Instance cost

- Randomly generated considering prices currently charged by common IaaS providers

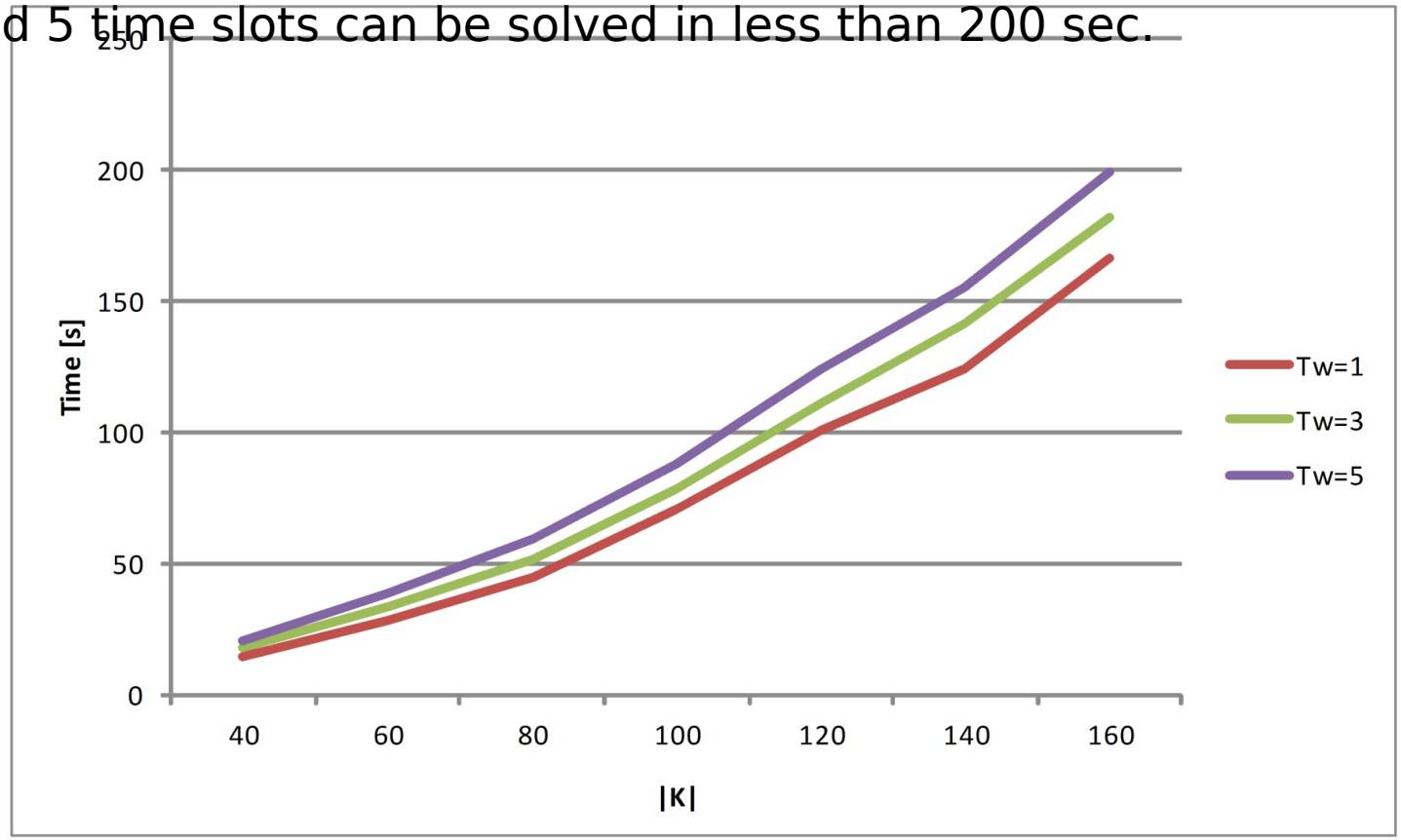
Experiment Design



MODAClouds

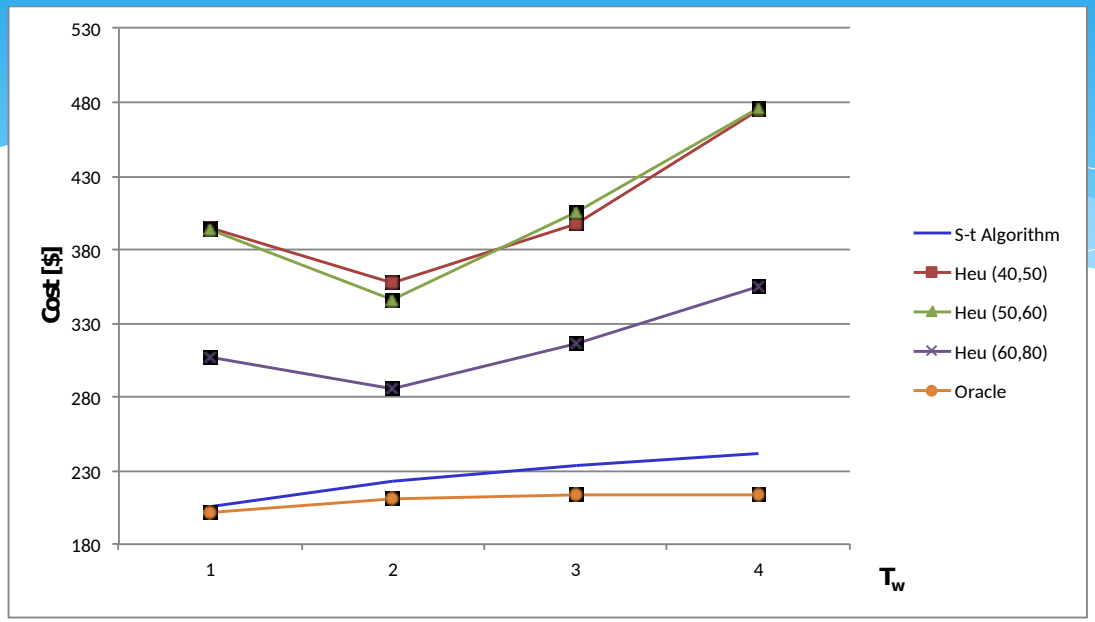
Scalability

The analysis demonstrated that our approach scales almost linearly with
respect to the number of request classes. Systems up to 160 classes and 5 time slots can be solved in less than 200 sec.



MODAClouds

Cost – Normal traffic



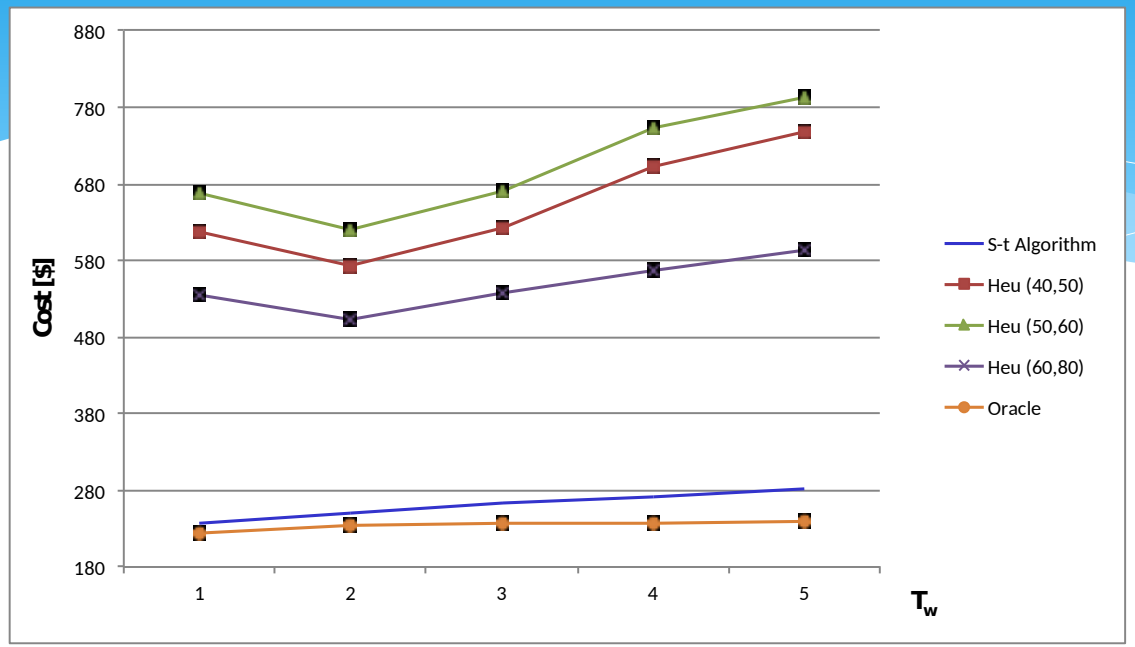
- 10 minutes time scale
- Low noise level

Costs comparison

Solution	$\mathcal{T}_w$ 1	2	3	4
Oracle	0.00%	0.00%	0.00%	0.00%
S-t Algorithm	2.00%	5.78%	9.31%	13.05%
Heu (40%, 50%)	95.81%	68.97%	85.51%	122.15%
Heu (50%, 60%)	95.09%	63.72%	85.59%	122.88%
Heu (60%, 80%)	52.39%	34.88%	47.54%	65.81%

MODAClouds

Cost – Spiky traffic



5 minutes time scale

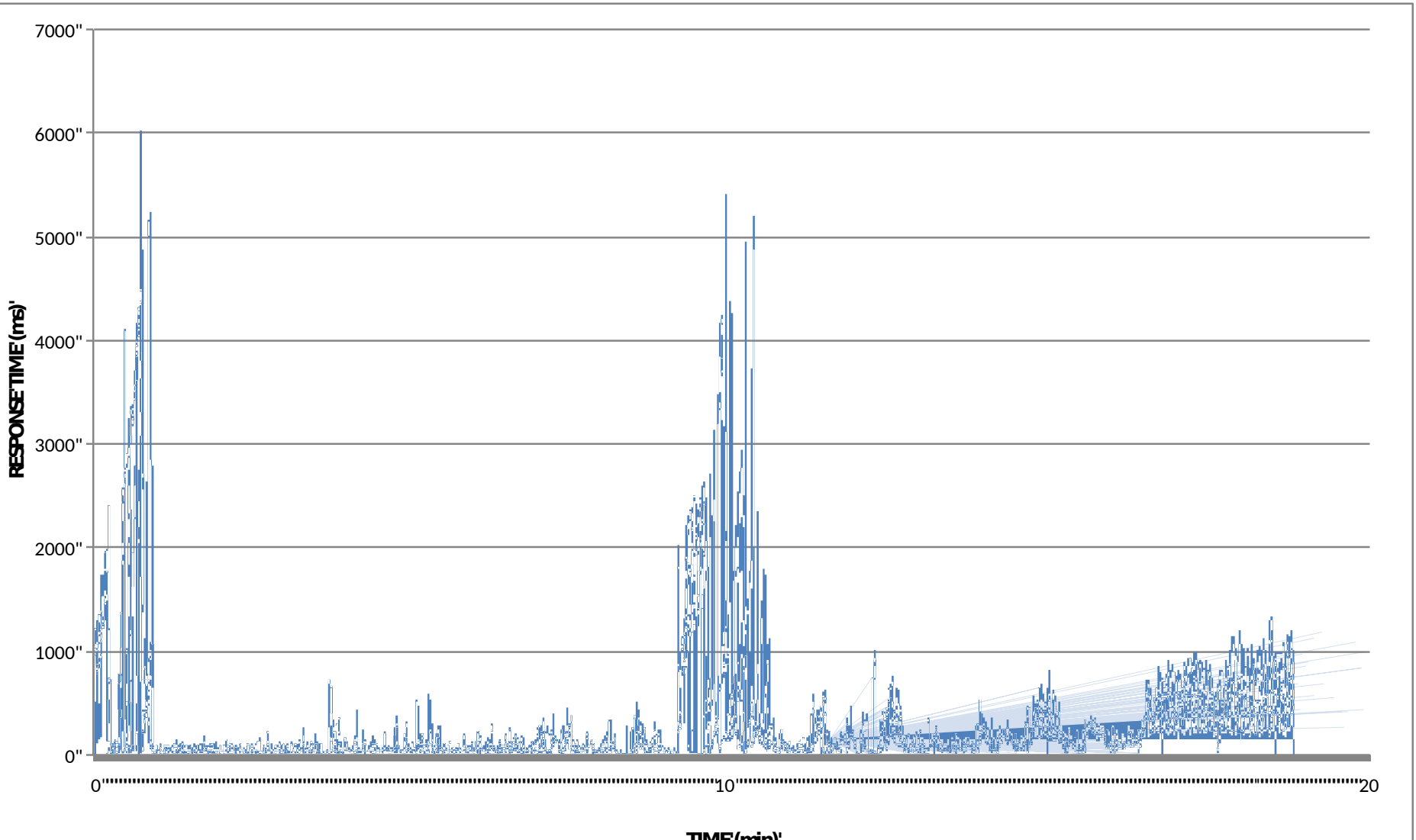
Low noise level

Costs comparison

Solution	$\mathcal{T}_w$ 1	2	3	4	5
Oracle	0.00%	0.00%	0.00%	0 .00%	0.00%
S-t Algorithm	5.10%	7.75%	11.02%	13.73%	16.86%
Heu (40%, 50%)	175.37%	145.68%	162.70%	195.95%	210.97%
Heu (50%, 60%)	197.36%	165.89%	183.58%	216.52%	229.00%
Heu (60%, 80%)	138.10%	115.46%	126.85%	138.88%	146.38%

MODAClouds

Time scale analysis



Time scale analysis

$\mathcal{T}_w$	SLA Violations [%]		Dropped Requests [%]	
	5 min	10 min	5 min	10 min
1	0.49	1.74	5.56	6.71
2	1.08	0.56	5.91	6.34
3	0.90	1.81	5.99	6.26
4	1.15	1.88	5.61	5.95

The values are related to a 24 hours analysis with low noise and averaged over 10 executions.

A control time granularity of 5 minutes tends to provide better performance if compared to granularity of 10 minutes both in terms of SLA violations and in terms of dropped requests.

MODAClouds

Agenda

* Introduction
* Problem
  * Problem statement and design assumption
  * Receding Horizon algorithm
* Experimental Analysis
* Conclusions

Conclusions and Future Works

We proposed optimization approach to achieve fast, scalable and effective capacity allocation based on a fine grained time scale

Our technique is able to minimize costs in a more efficient way than the current state of the art

The QoS defined into the SLA is almost always respected (less than 2% and 7 min)

Future works:

* development of an adaptive approach able to switch between different time scales according to the workload conditions
* Test on a real prototype environment

MODAClouds

Thank You!

Questions ?