Autonomic Cloud Controllers

Pooyan Jamshidi, Claus Pahl, Giovani Estrada

Imperial College London, UK
IC4, Dublin City University, Ireland
Intel, Ireland

p.jamshidi@imperial.ac.uk

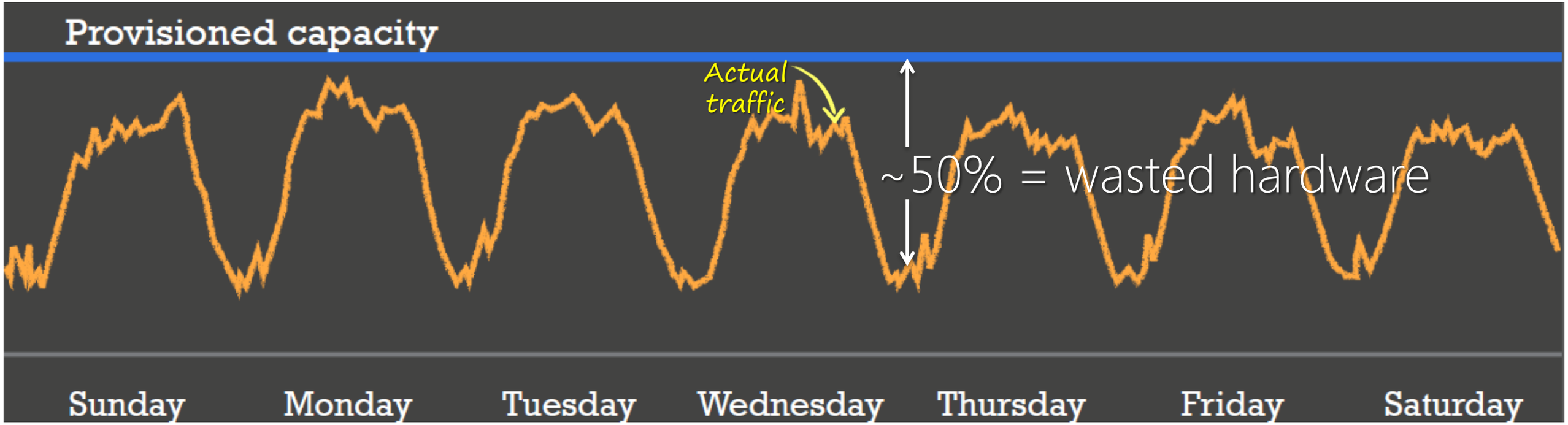
DCU

ucc
Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

AIT
Institiúid Teicneolaíochta Bhaile Átha Luain
Athlone Institute of Technology

Motivation

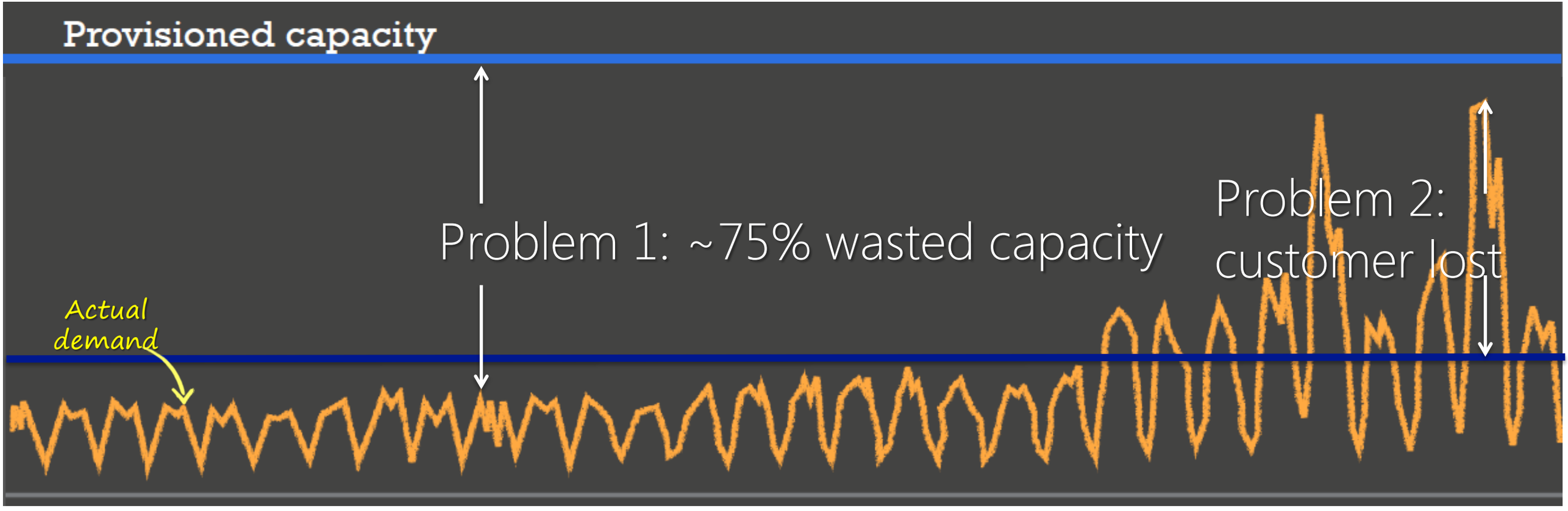
Typical weekly traffic to Web-based applications (e.g., Amazon.com)



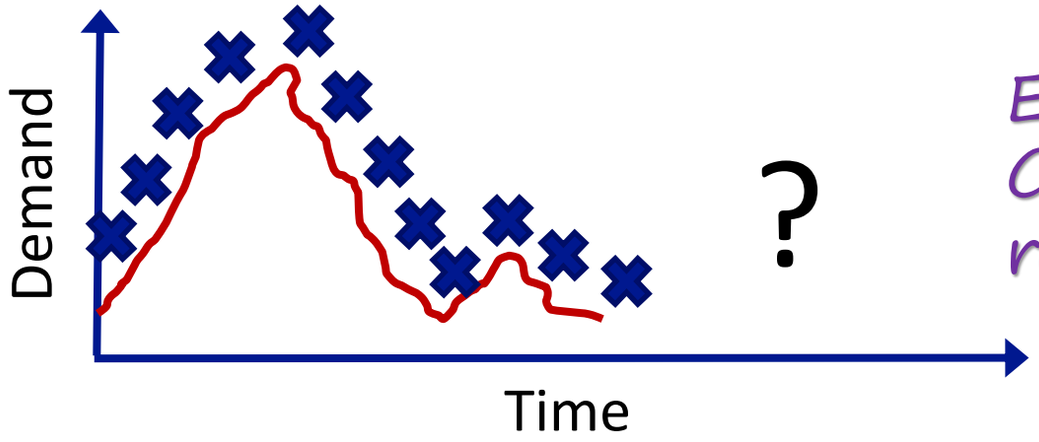
Motivation

Traffic in an unexpected burst in requests (e.g. end of year traffic to Amazon.com)

users dissatisfaction

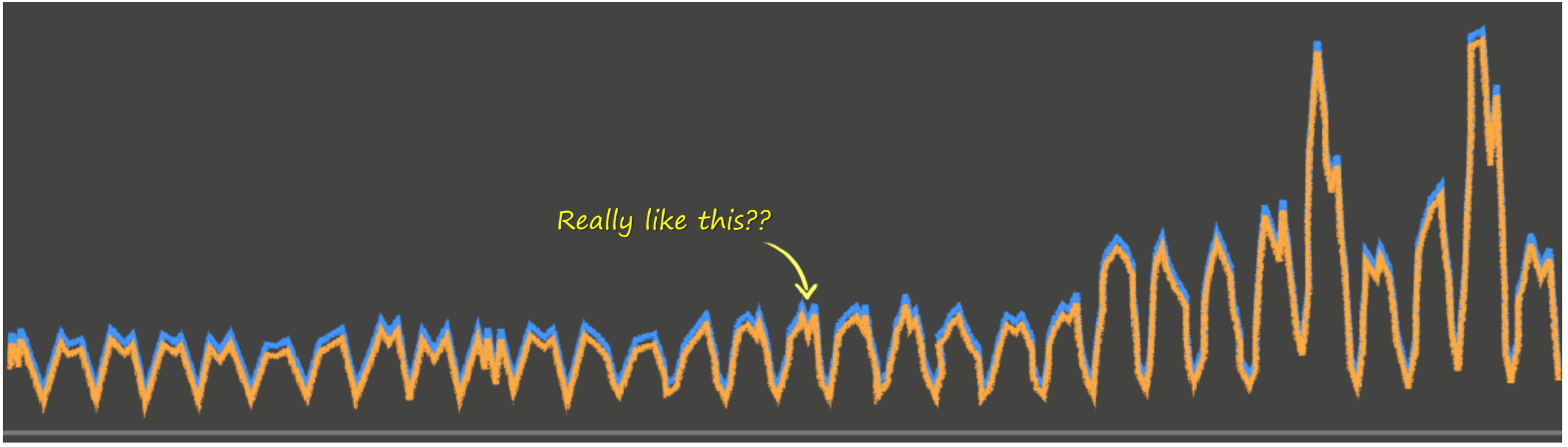


Motivation



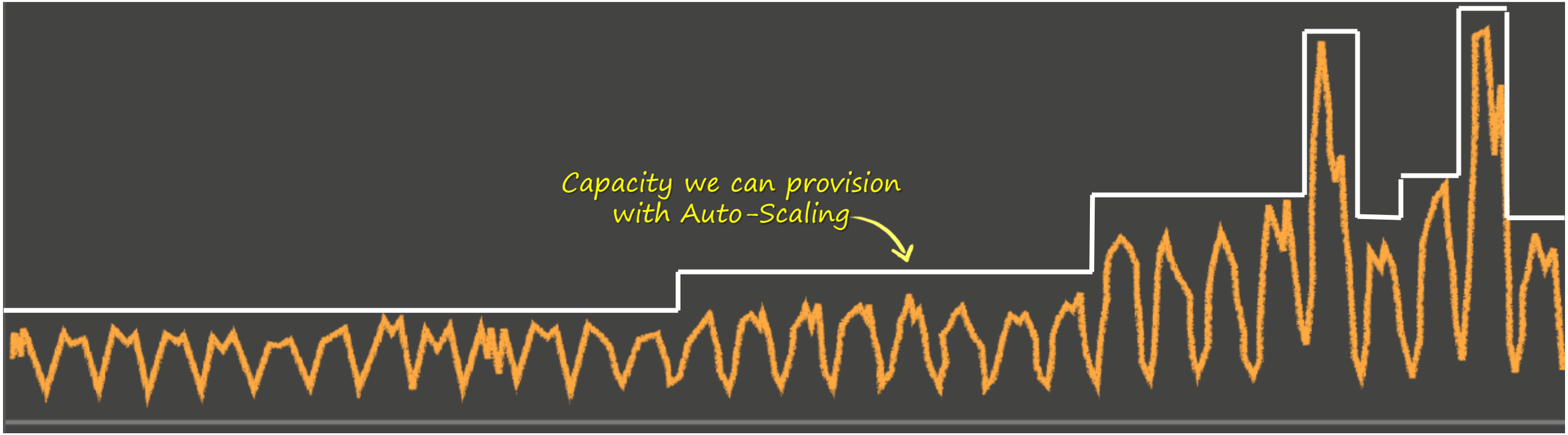
Enacting change in the Cloud resources are not real-time

Auto-scaling enables you to realize this ideal on-demand provisioning



Motivation

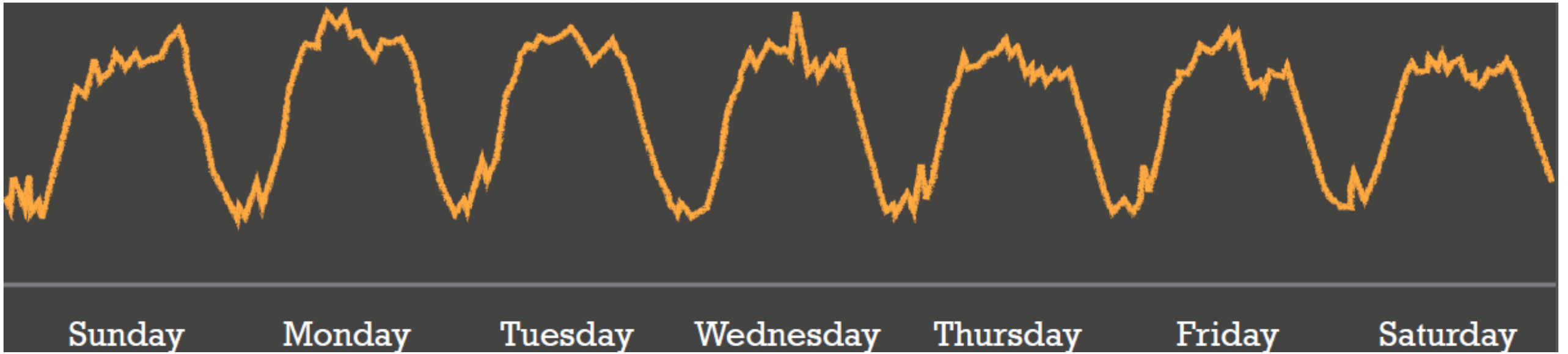
A realistic figure of dynamic provisioning

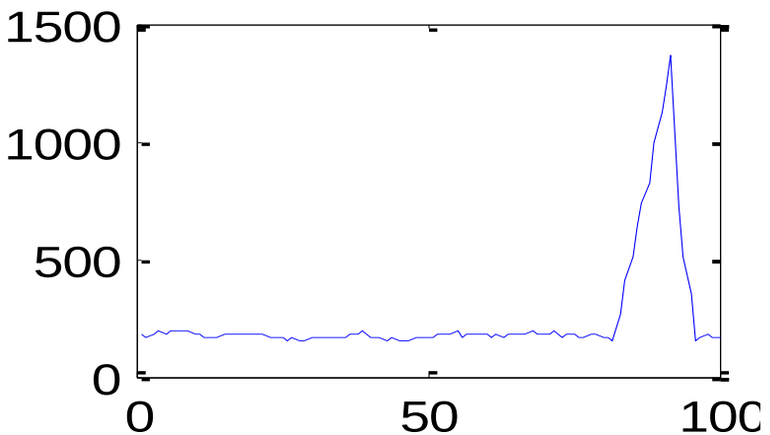


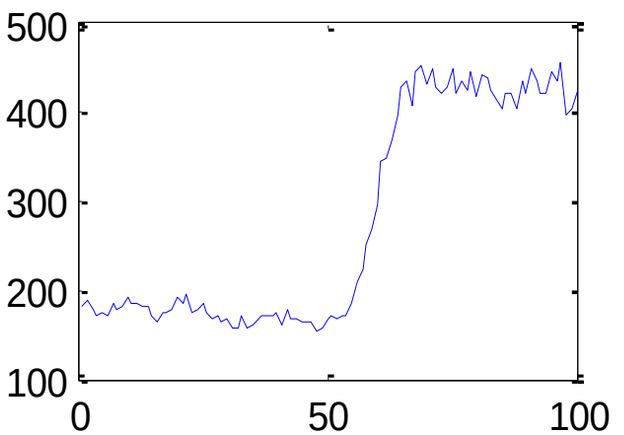
Conclusions (Impact on Industry)

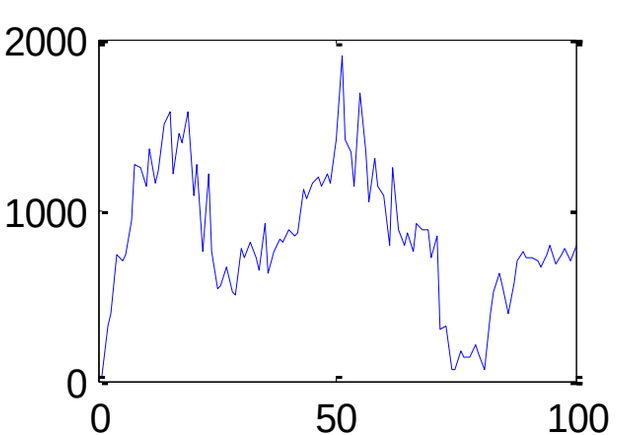
- User dependency –> autonomic solution
- Uncertainty –> robust solution
- Overhead –> low runtime overhead

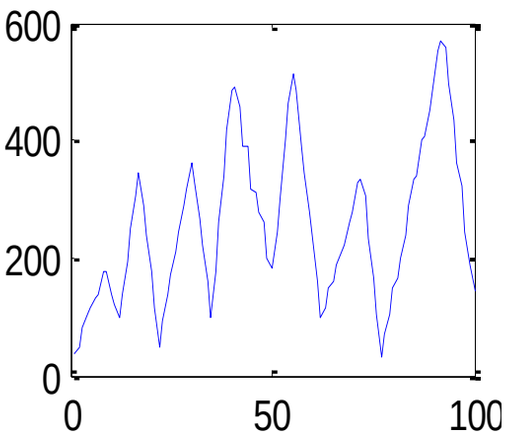
Predictable vs. Unpredictable Demand









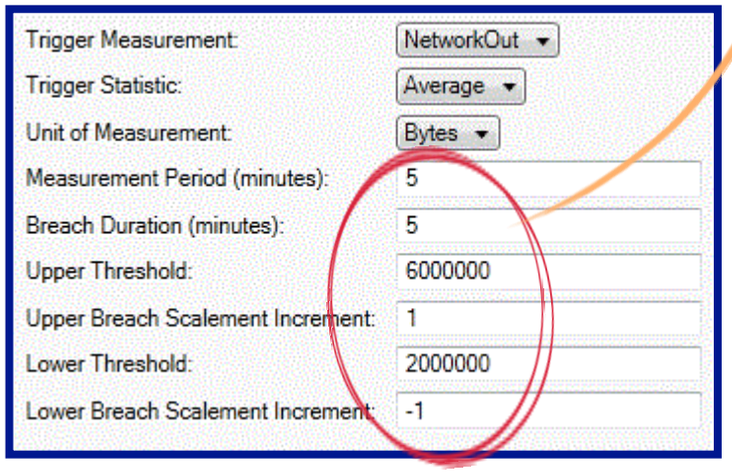


An Example of Auto-scaling Rule

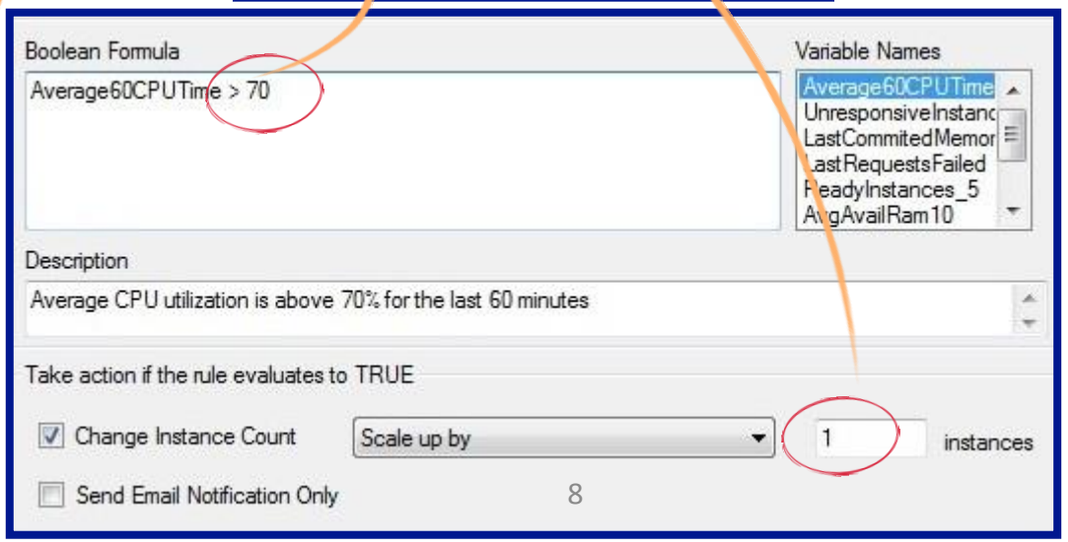
Microsoft Azure Auto-scaling Application Block

```
<rule name="Example Scaling Rule" rank="100">
  <when>
    <greater operand="CPU_RoleA" than="80"/>
  </when>
  <actions>
    <scale target="WorkerRoleA" by="2"/>
  </actions>
</rule>
```

Amazon auto scaling



Microsoft Azure Watch



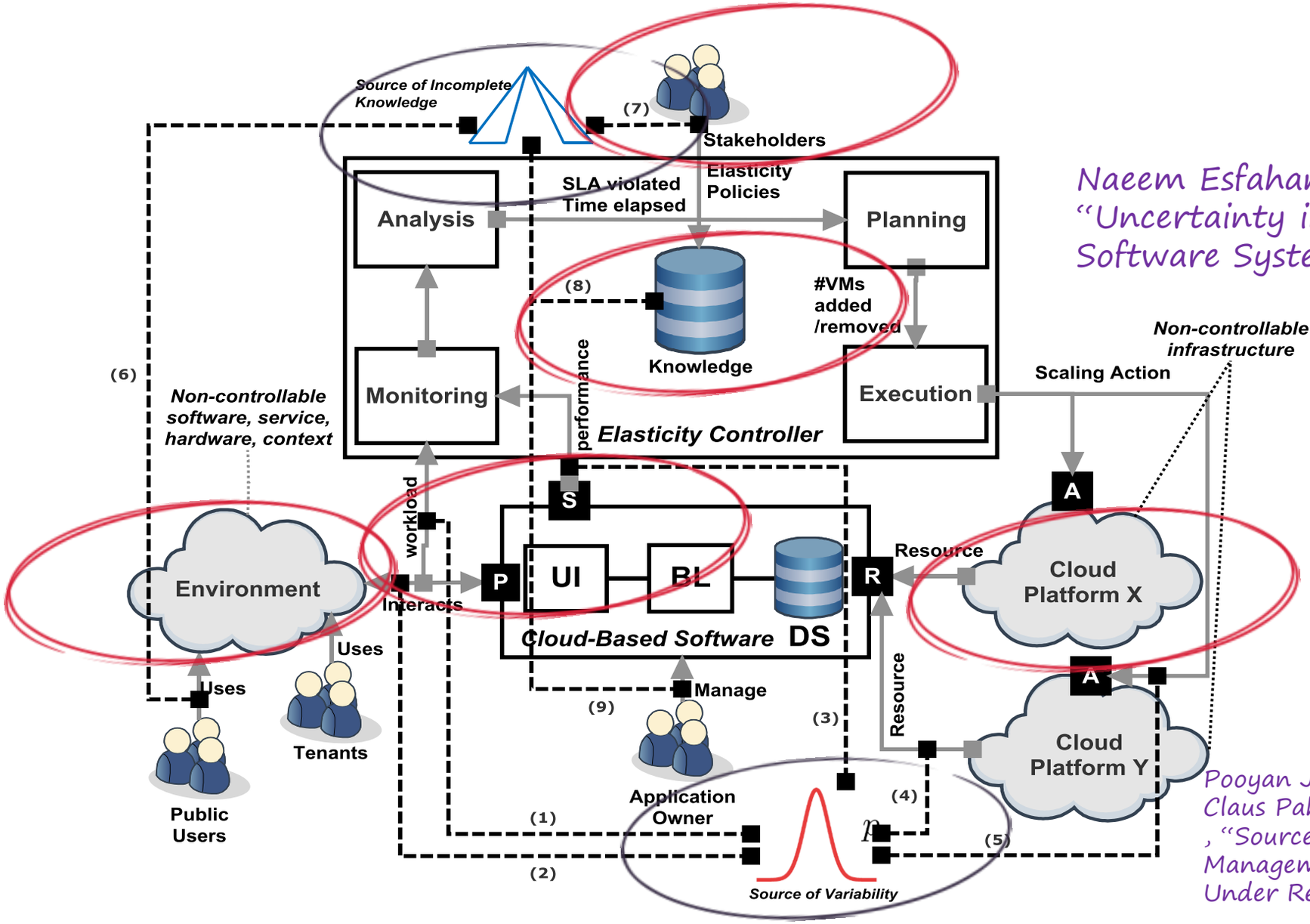
These quantitative values are required to be determined by the user

⇒ requires deep knowledge of application (CPU, memory, thresholds)

⇒ requires performance modeling expertise (when and how to scale)

⇒ A unified opinion of user(s) is required

Sources of Uncertainty in Elastic Software



Naeem Esfahani and Sam Malek, "Uncertainty in Self-Adaptive Software Systems"

Pooyan Jamshidi, Aakash Ahmad, Claus Pahl, Muhammad Ali Babar , "Sources of Uncertainty in Dynamic Management of Elastic Systems", Under Review

A concrete example of uncertainty in the cloud

Uncertainty related to enactment latency:
The same scaling action (adding/removing a VM with precisely the same size) took different time to be enacted on the cloud platform (here is Microsoft Azure) at different points and this difference were significant (up to couple of minutes).
The enactment latency would be also different on different cloud platforms.

ROLE NAME	QTY
LoyaltyManagementProcessing	2
LoyaltyManagementService	1

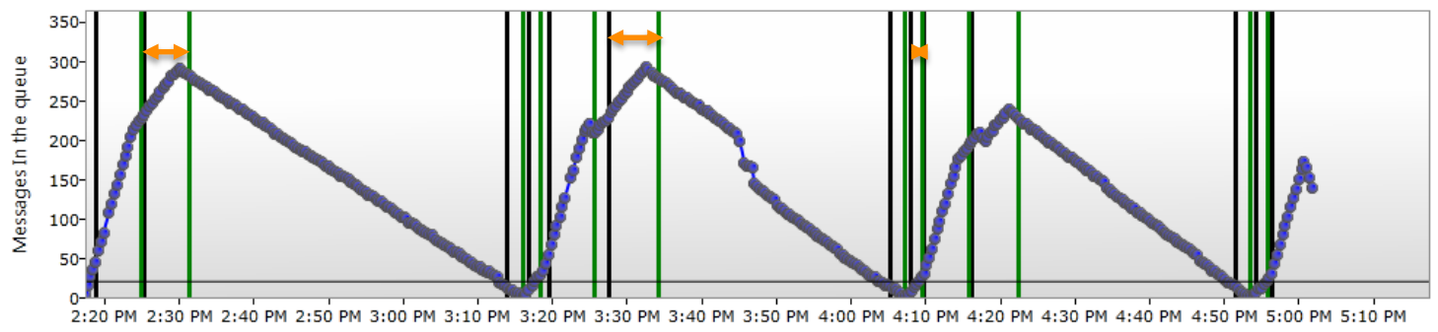
RECONFIGURING 1:58

GRAPH LEGEND

- Threshold for increasing role count
- Queue length
- Action Requested *
- Action Completed *
- Action Failed *

* Hover over line in graph for more information

STATS

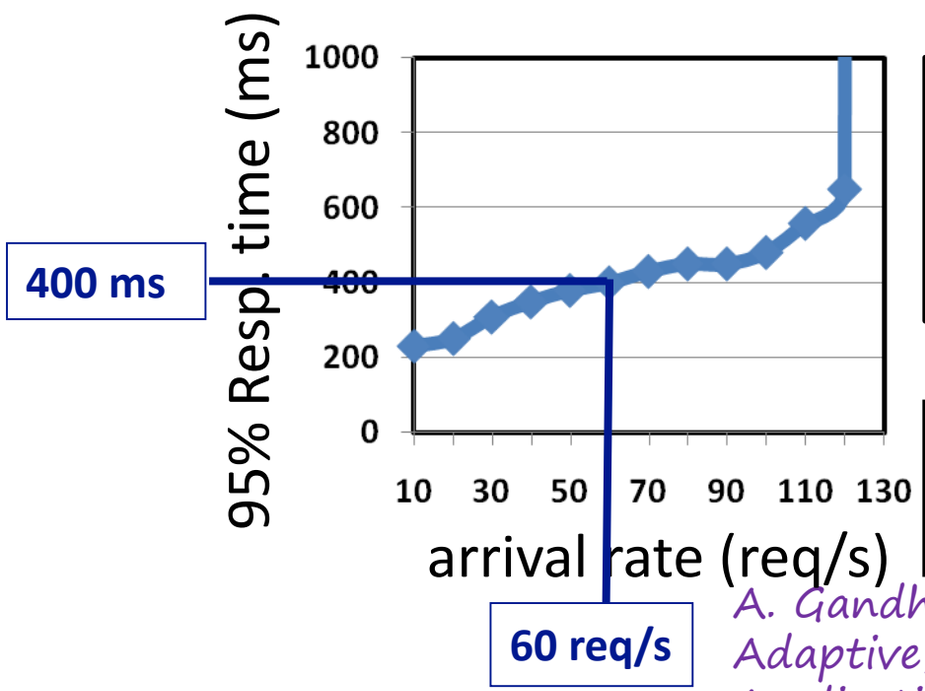


Graph time (min): 180 min

TYPE	STATUS	TIMESTAMP	COMMENTS
Increase	Completed	5/13/2014 3:25:37 PM	Metric 'QueueLength' is growing at rate 7 and has been above threshold of 20 for 60 seconds
Increase	Requested	5/13/2014 3:19:29 PM	Metric 'QueueLength' is growing at rate 7 and has been above threshold of 20 for 60 seconds
Decrease	Completed	5/13/2014 3:18:27 PM	Metric 'QueueLength' has been below threshold of 20 for 60 seconds
Decrease	Requested	5/13/2014 3:16:54 PM	Metric 'QueueLength' has been below threshold of 20 for 60 seconds
Decrease	Completed	5/13/2014 3:16:04 PM	Metric 'QueueLength' has been below threshold of 20 for 60 seconds
Decrease	Requested	5/13/2014 3:13:58 PM	Metric 'QueueLength' has been below threshold of 20 for 60 seconds
Increase	Completed	5/13/2014 2:31:25 PM	Metric 'QueueLength' is growing at rate 5 and has been above threshold of 20 for 60 seconds
Increase	Requested	5/13/2014 2:25:19 PM	Metric 'QueueLength' is growing at rate 5 and has been above threshold of 20 for 60 seconds
Increase	Completed	5/13/2014 2:24:55 PM	Metric 'QueueLength' is growing at rate 10.5 and has been above threshold of 20 for 60 seconds
Increase	Requested	5/13/2014 2:18:50 PM	Metric 'QueueLength' is growing at rate 10.5 and has been above threshold of 20 for 60 seconds

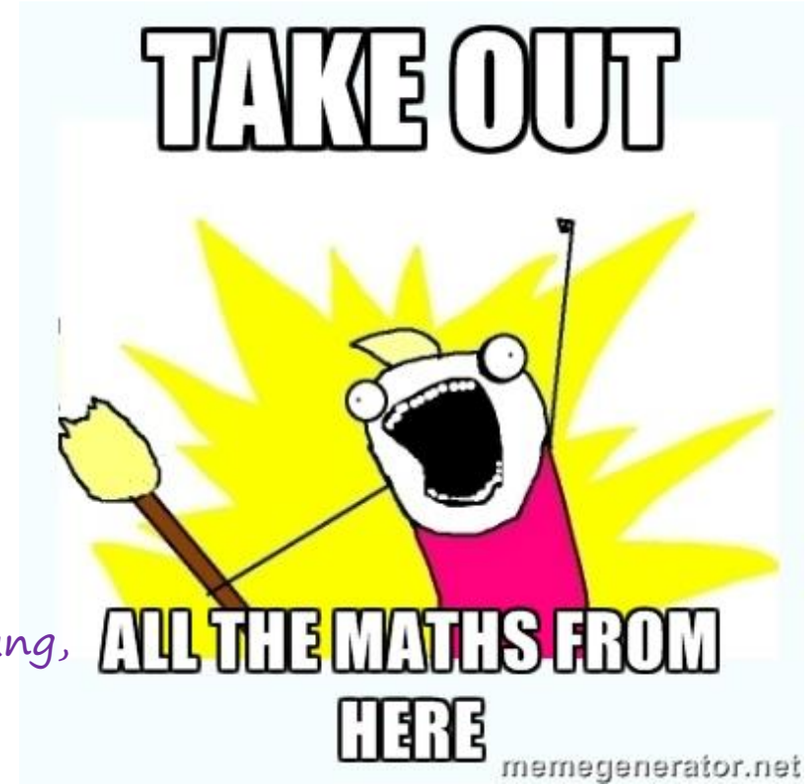
Goal!

- Take the burden away from the user
  - users specifies the thresholds through **qualitative linguistics**
  - the auto-scaling controller should be **fully responsible** for scaling decisions
  - the auto-scaling should be **robust** against uncertainty



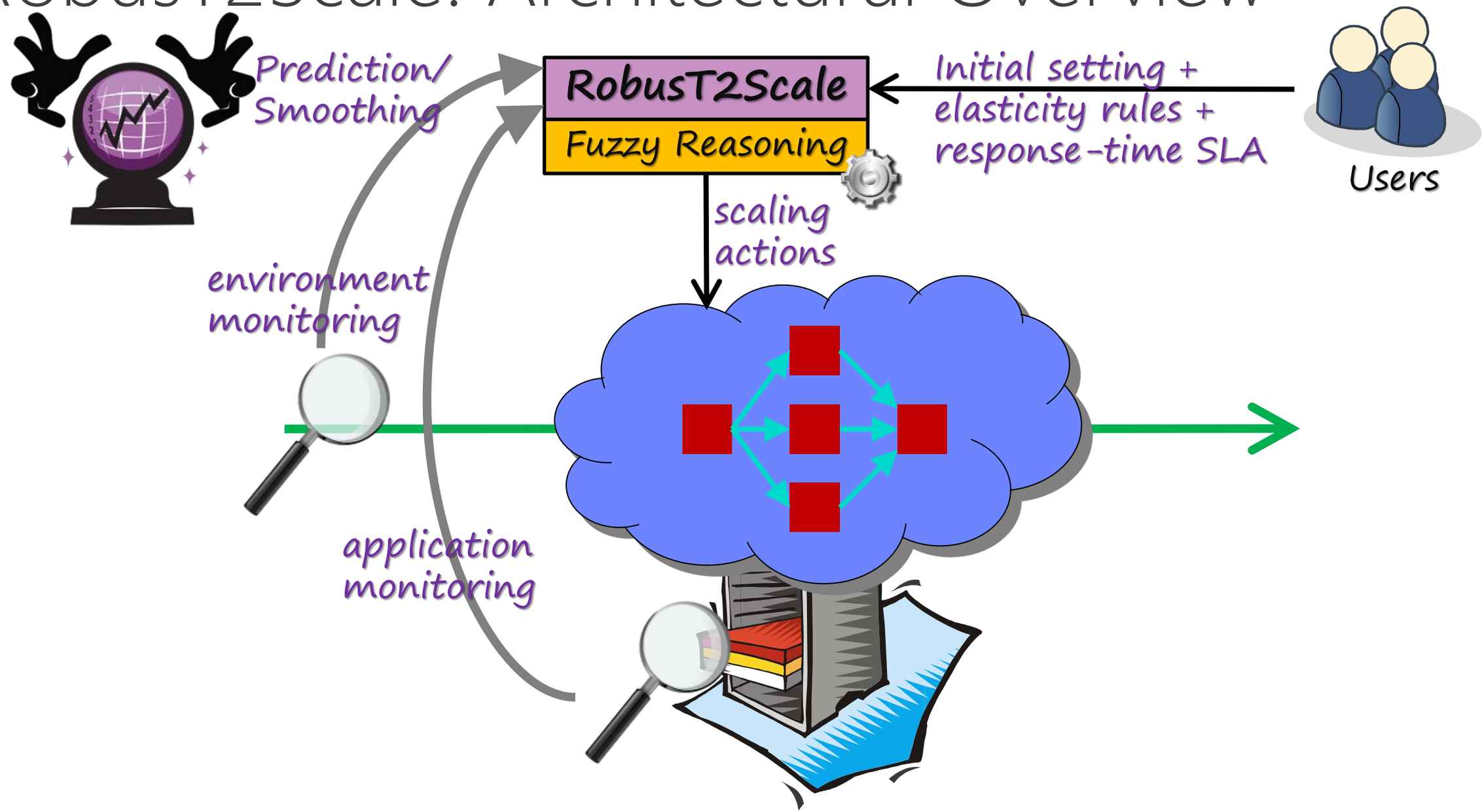
- Offline benchmarking
- Trial-and-error
- Expert knowledge

Costly and not systematic

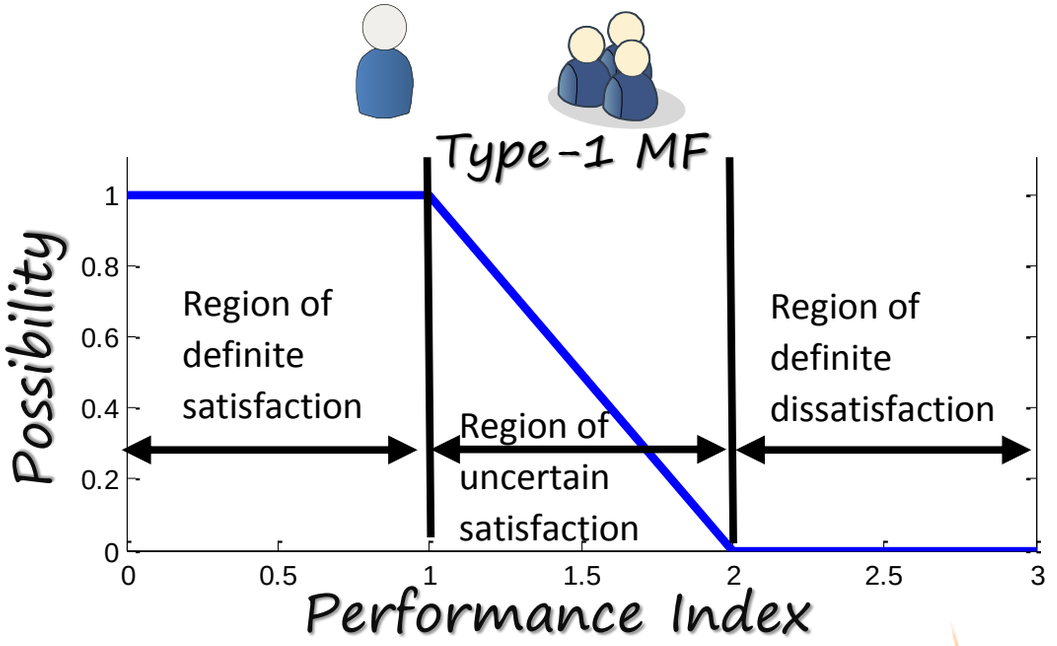


A. Gandhi, P. Dube, A. Karve, A. Kochut, L. Zhang, Adaptive, "Model-driven Autoscaling for Cloud Applications", ICAC'14

RobusT2Scale: Architectural Overview

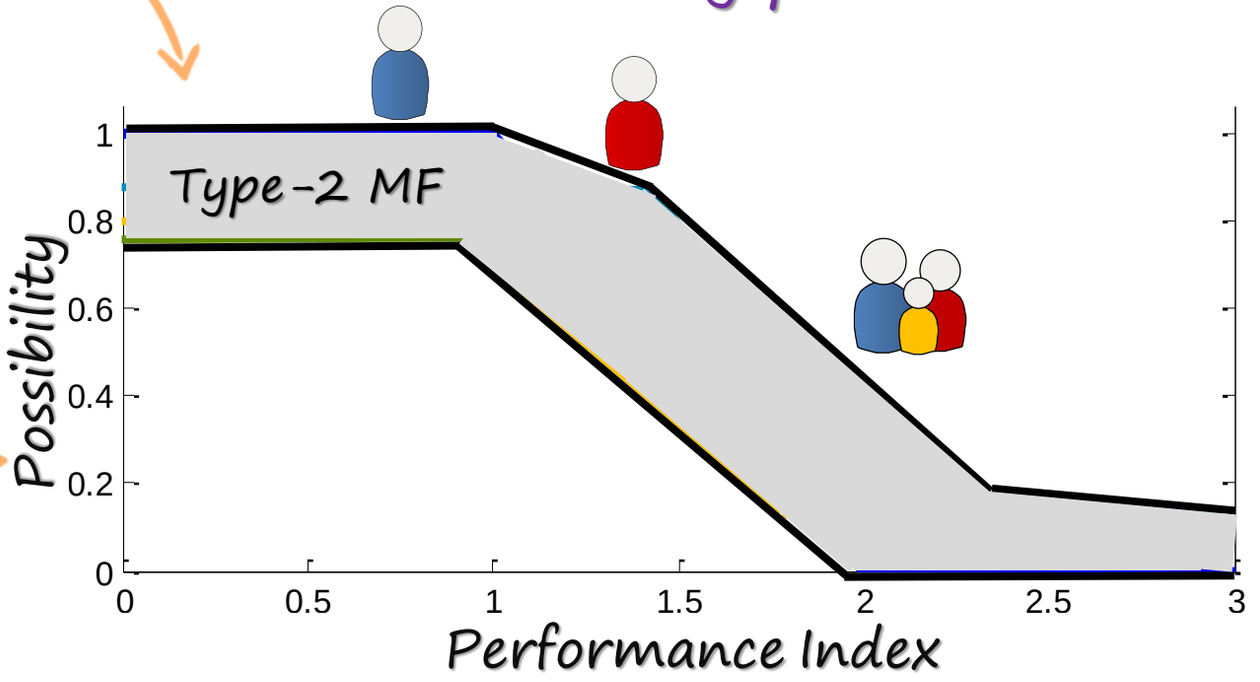


Why we decided to use type-2 fuzzy logic?



Different users often recommend different elasticity policies

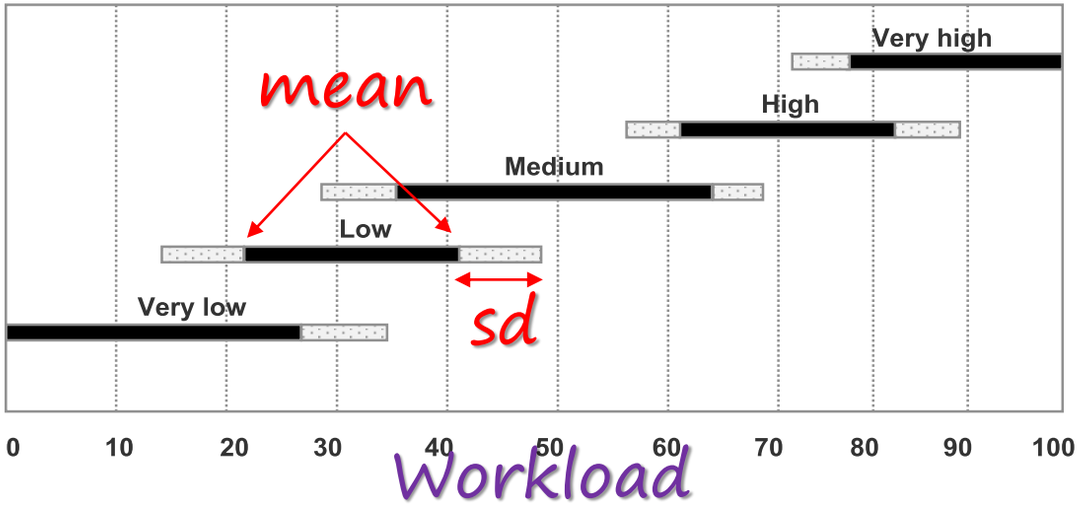
words can mean different things to different people



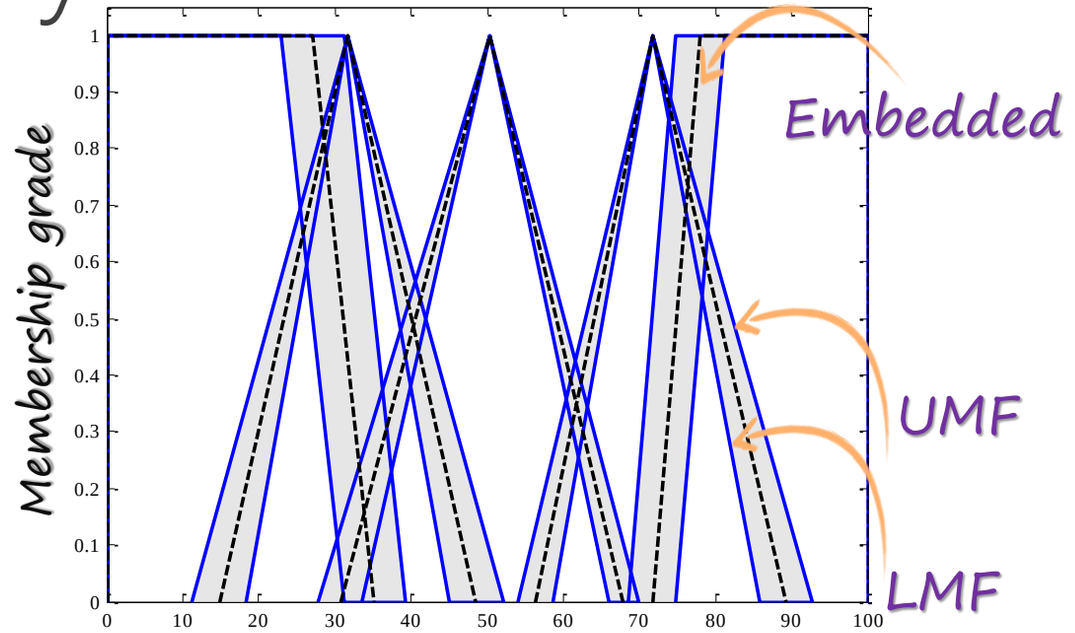
How we designed the fuzzy controller?

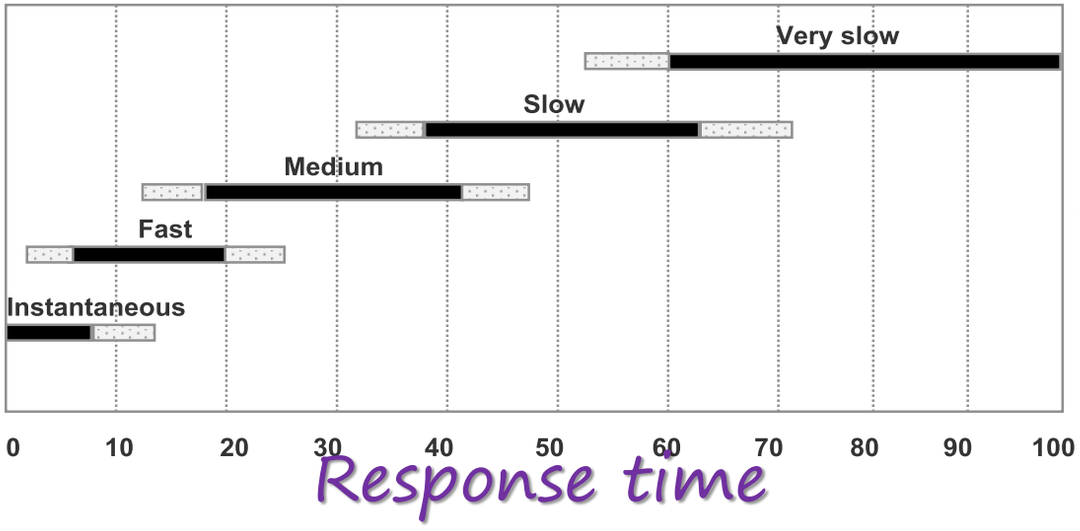
- The fuzzy logic controller is completely defined by its "membership functions" and "fuzzy rules".
- Knowledge elicitation through a survey of 10 experts.

Survey processing and fuzzy MF construction

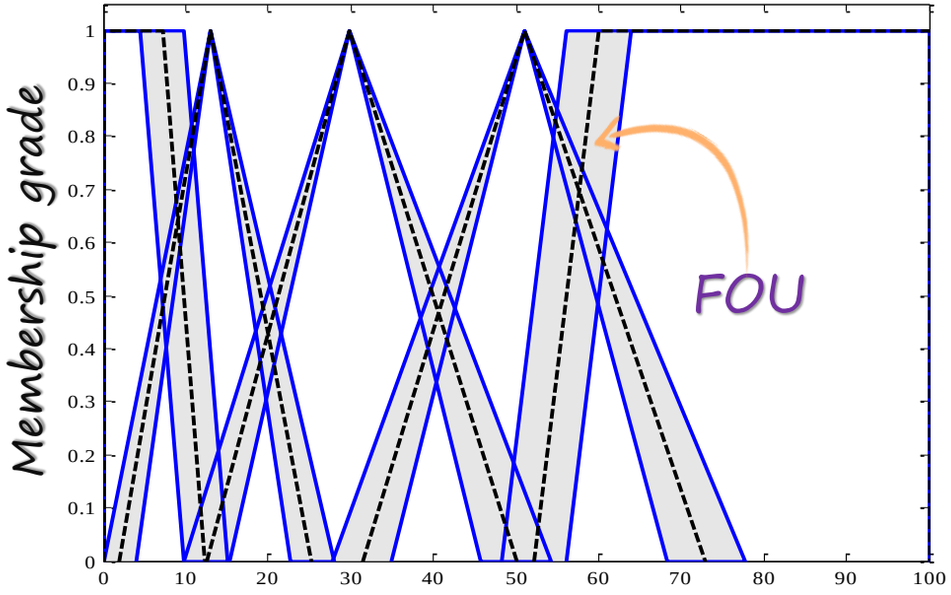


=>





=>



Fuzzy rule elicitation

Rule (l)	Antecedents		Consequent					c_{avg}^{l}
	Workload	Response-time	Normal (-2)	Effort (-1)	Medium Effort (0)	High Effort (+1)	Maximum Effort (+2)	
1	Very low	Instantaneous	7	2	1	0	0	-1.6
2	Very low	Fast	5	4	1	0	0	-1.4
3	Very low	Medium	0	2	6	2	0	0
4	Very low	Slow	0	0	4	6	0	0.6
5	Very low	Very slow	0	0	0	6	4	1.4
6	Low	Instantaneous	5	3	2	0	0	-1.3
7	Low	Fast	2	7	1	0	0	-1.1
8	Low	Medium	0	1	5	3	1	0.4
9	Low	Slow	0	0	1	8	1	1
10	Low	Very slow	0	0	0	4	6	1.6
11	Medium	Instantaneous	6	4	0	0	0	-1.6
12	Medium	Fast	2	5	3	0	0	-0.9
13	Medium	Medium	0	0	5	4	1	0.6
14	Medium	Slow	0	0	1	7	2	1.1
15	Medium	Very slow	0	0	1	3	6	1.5
16	High	Instantaneous	8	2	0	0	0	-1.8
17	High	Fast	4	6	0	0	0	-1.4
18	High	Medium	0	1	5	3	1	0.4
19	High	Slow	0	0	1	7	2	1.1
20	High	Very slow	0	0	0	6	4	1.4
21	Very high	Instantaneous	9	1	0	0	0	-1.9
22	Very high	Fast	3	6	1	0	0	-1.2
23	Very high	Medium	0	1	4	4	1	0.5
24	Very high	Slow	0	0	1	8	1	1
25	Very high	Very slow	0	0	0	4	6	1.6

R^l: IF (the workload (x_1) is $\tilde{F}_{i_1}$, AND the response-time (x_2) is $\tilde{G}_{i_2}$), THEN (add/remove c_{avg}^{l} instances).

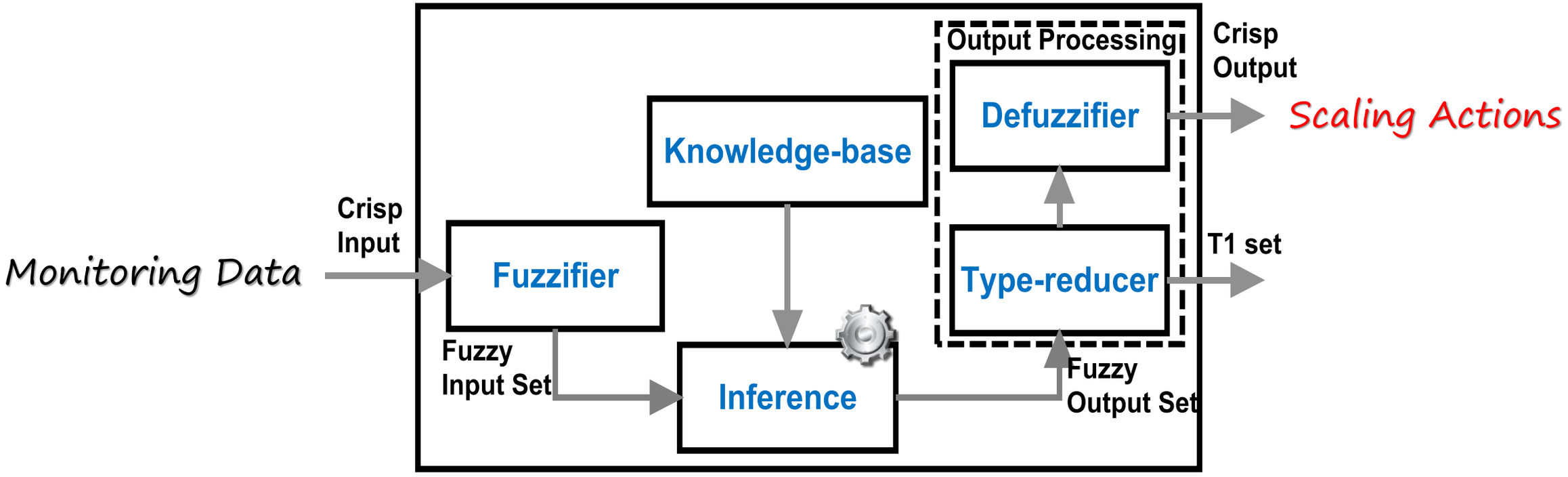
$$c_{avg}^{l} = \frac{\sum_{u=1}^{N_l} w_u^l \times C}{\sum_{u=1}^{N_l} w_u^l}$$

Rule ()	Antecedents		Consequent					
	Work load	Response -time	-2	-1	0	+1	+2	
12	Medium	Fast	2	5	3	0	0	-0.9

10 experts' responses

Goal: pre-computations of costly calculations to make a runtime efficient elasticity reasoning based on fuzzy inference

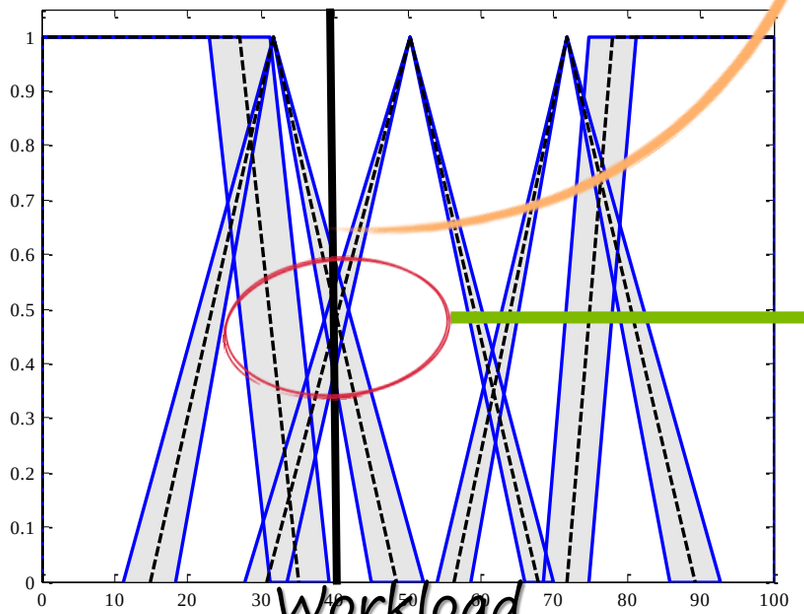
Elasticity Reasoning @ Runtime

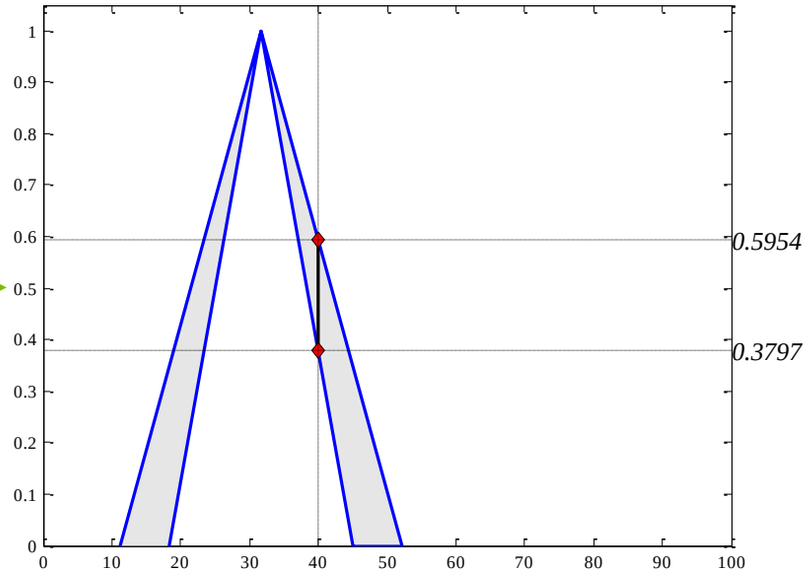


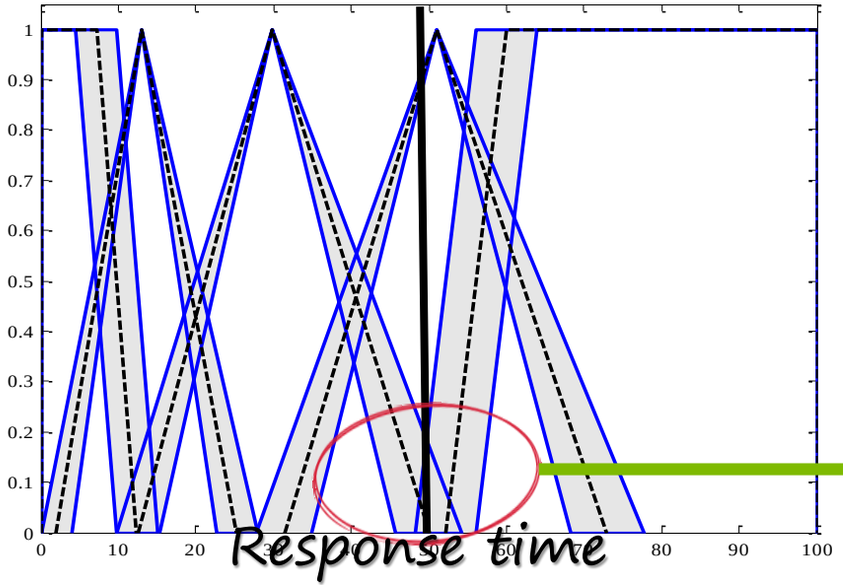
Liang, Q., Mendel, J. M. (2000). Interval type-2 fuzzy logic systems: theory and design. Fuzzy Systems, IEEE Transactions on, 8(5), 535-550.

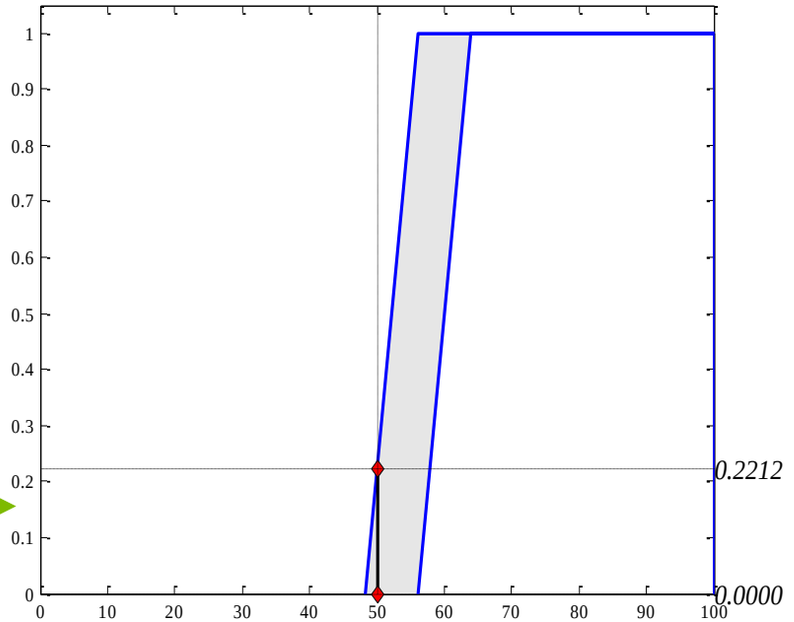
Fuzzification

Monitoring data

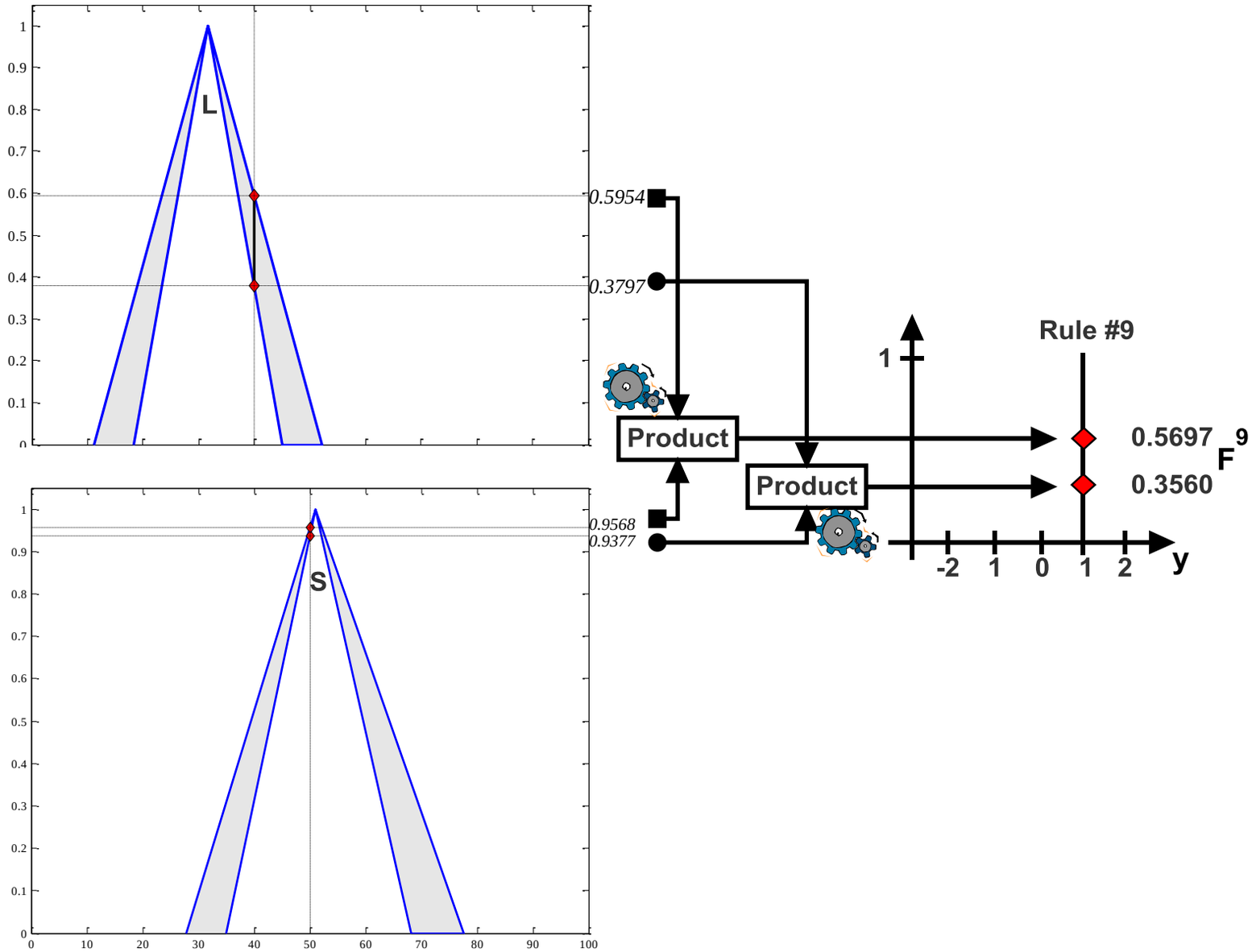




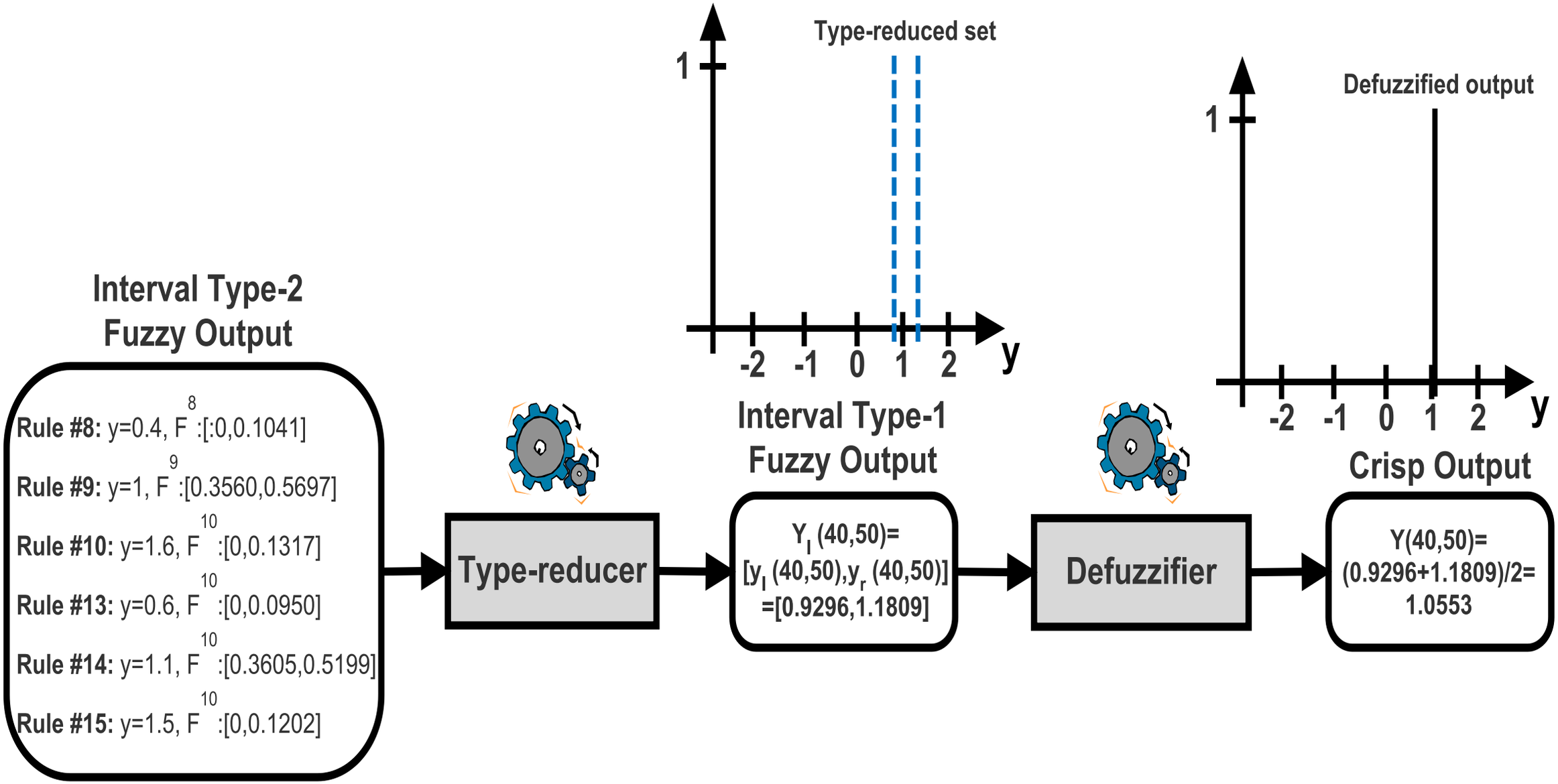




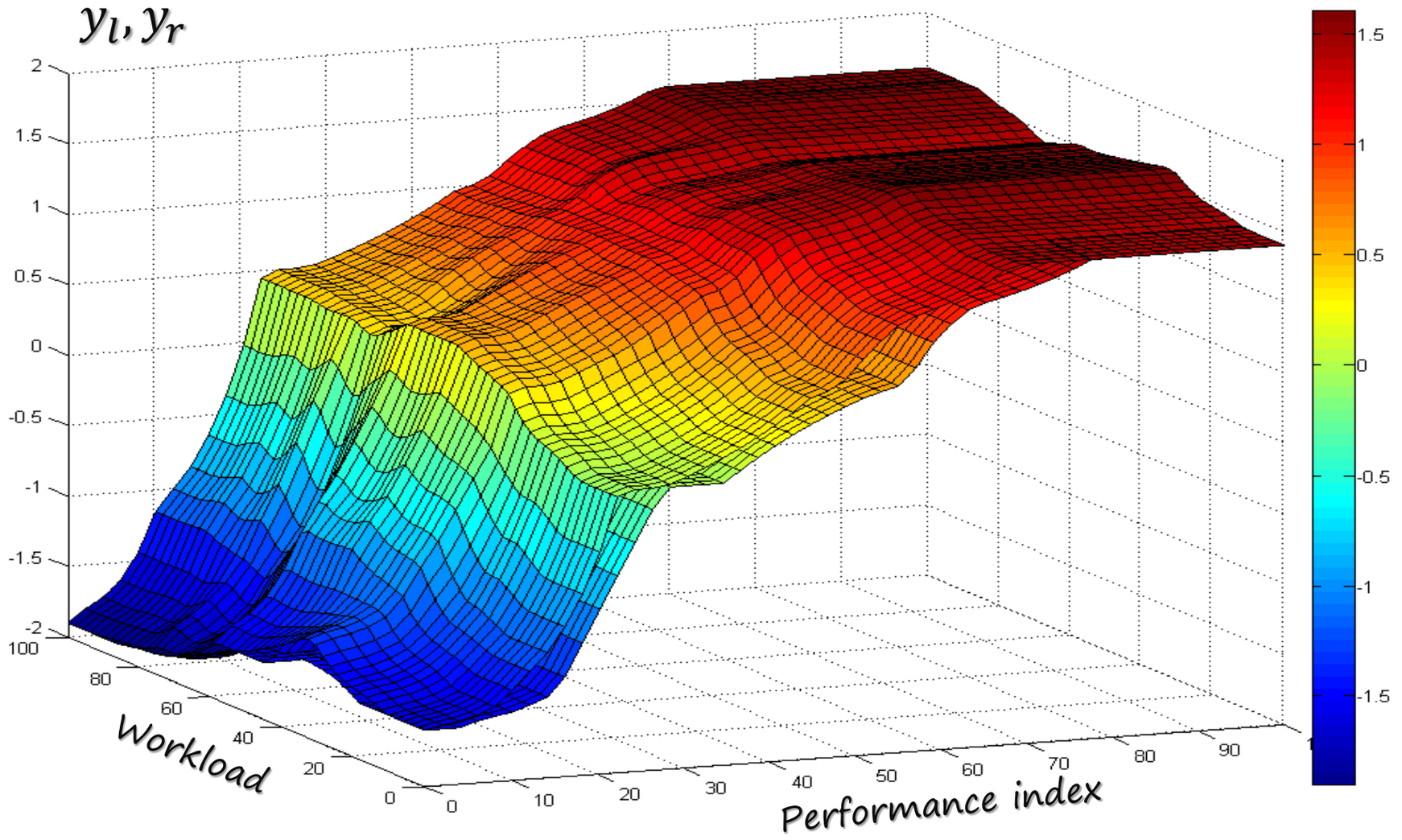
Inference Mechanism



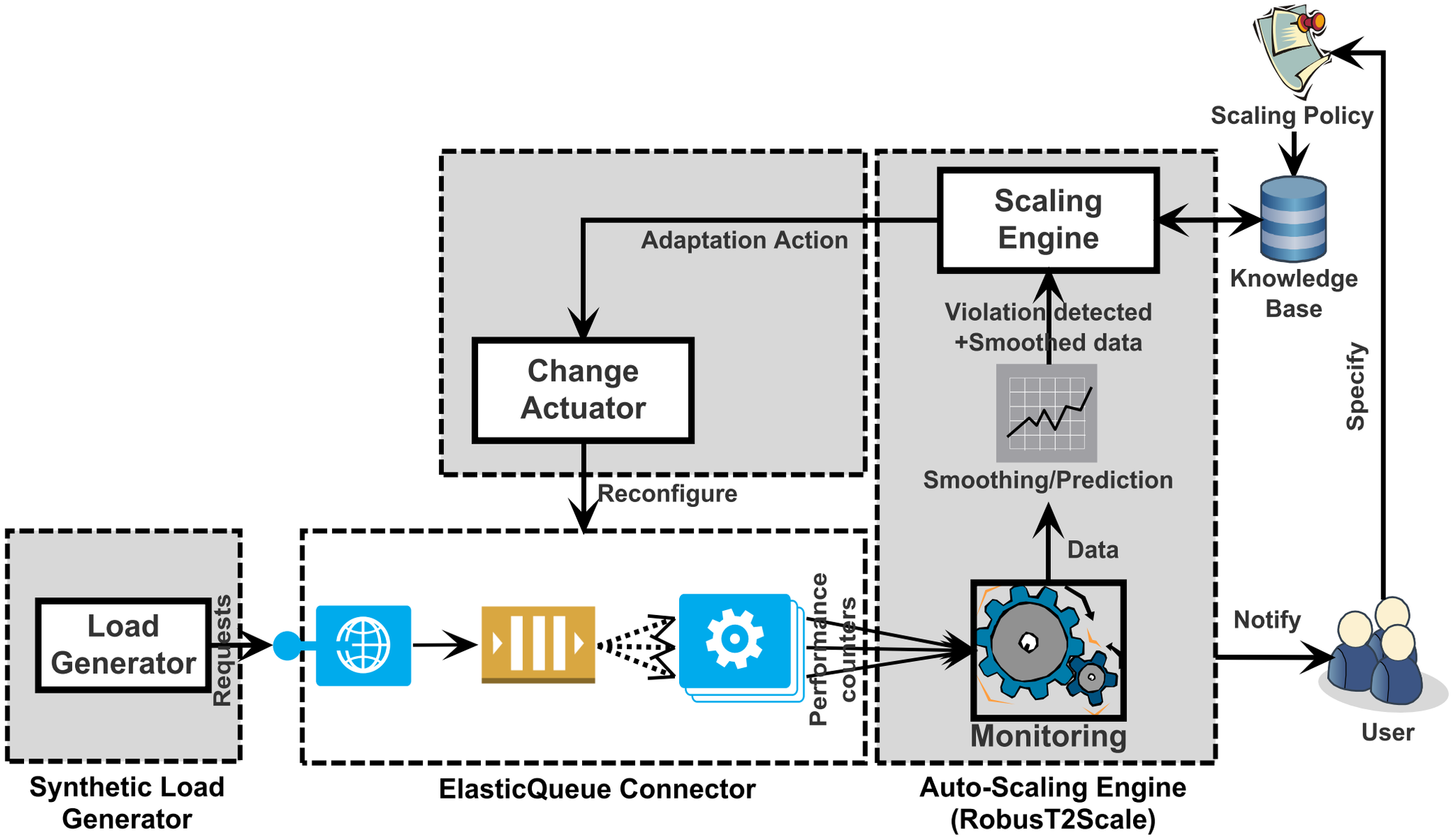
Output Processing



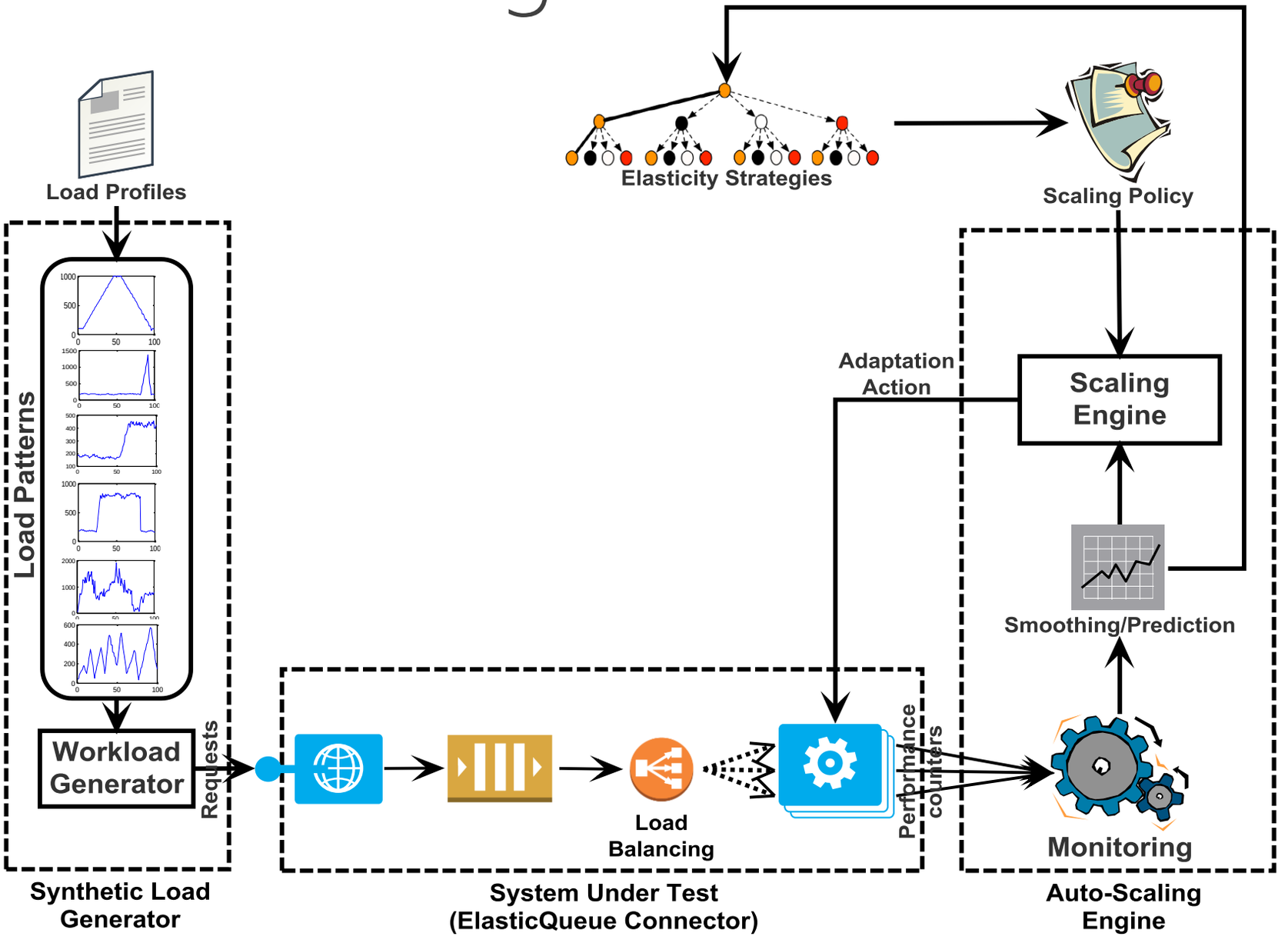
Control Surface



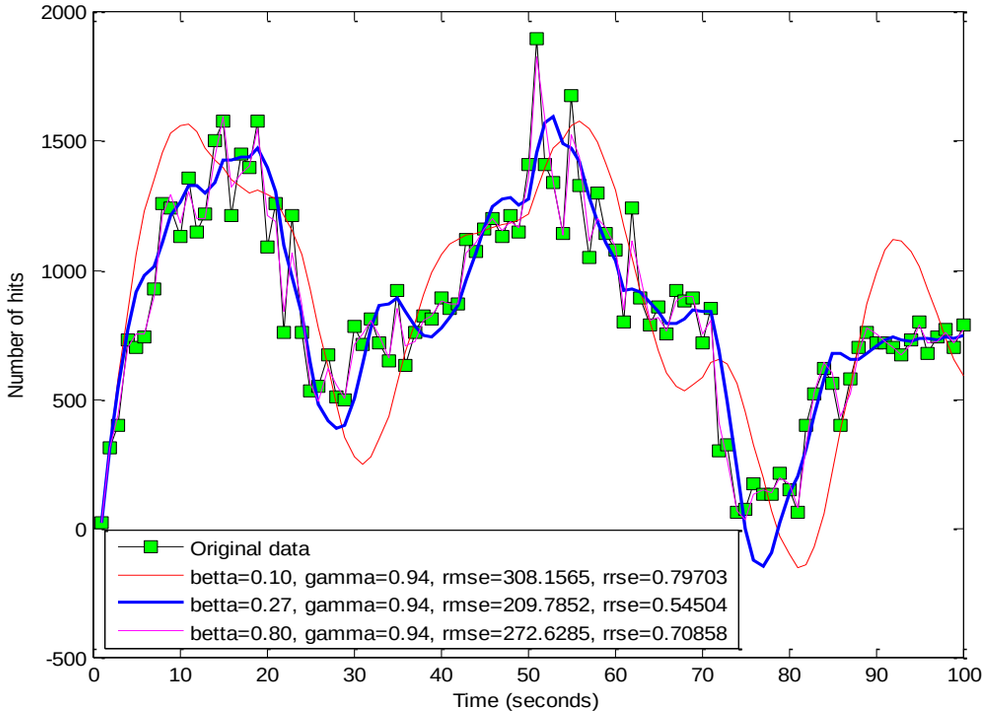
Tool Chain Architecture

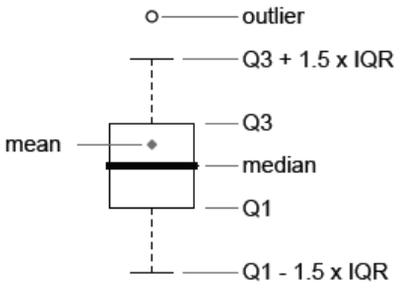


Experimental Setting: Process View

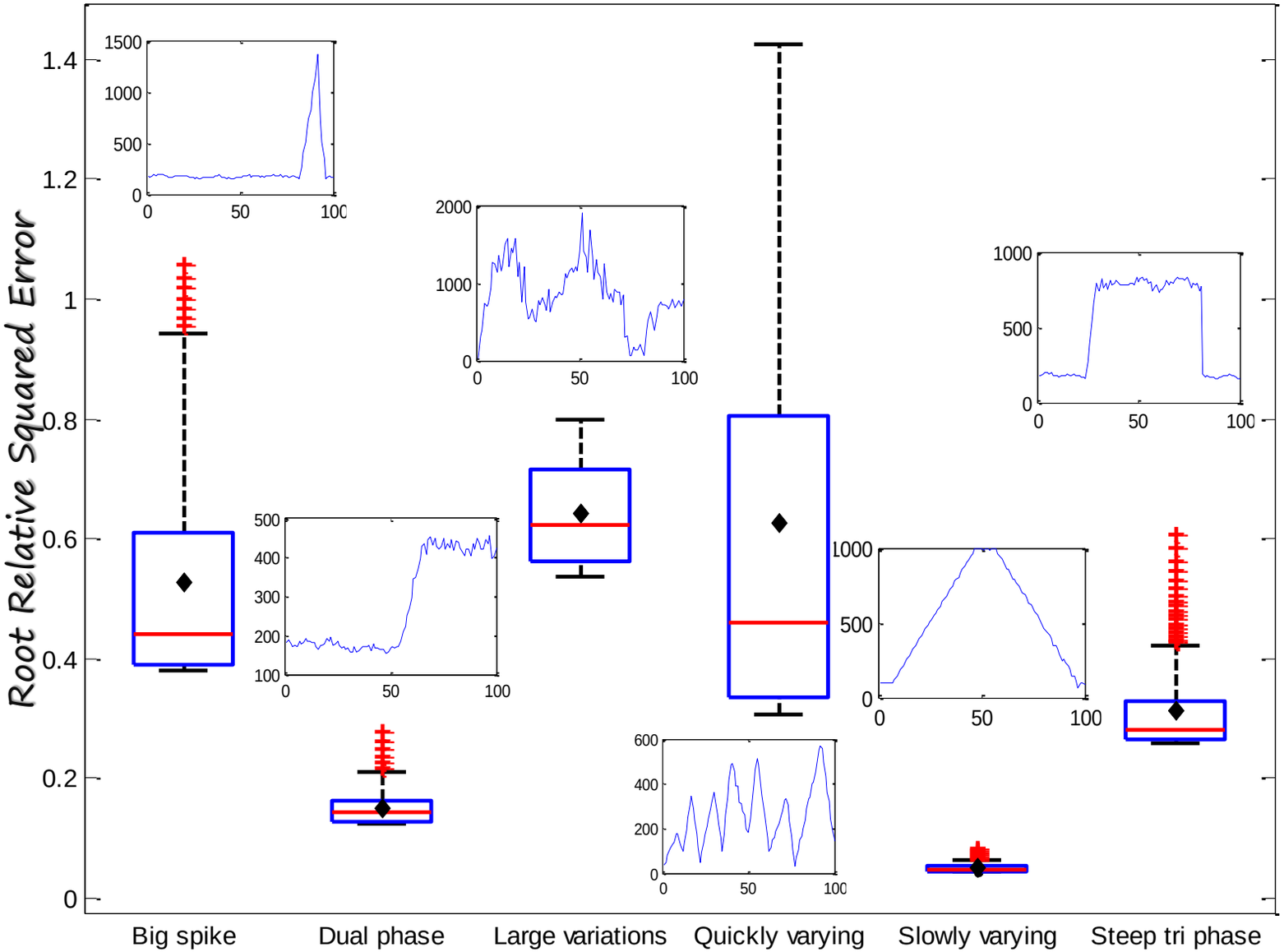


Estimation Errors w.r.t. Workload Patterns





Q1 = cuts off lowest 25% of data
median = cuts data set in half
Q3 = cuts off highest 25% of data
IQR = Q3 - Q1



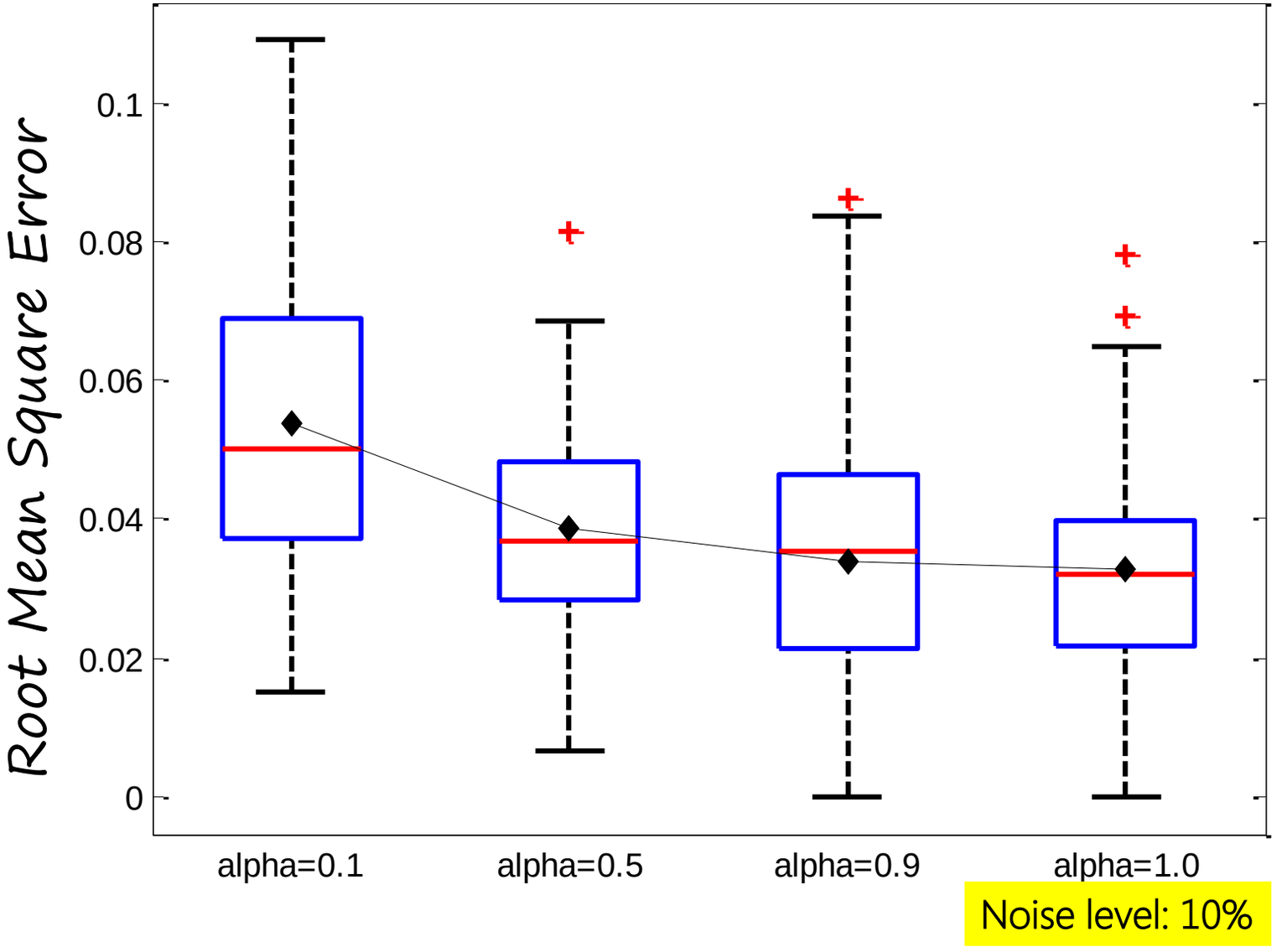
Effectiveness of RobusT2Scale

SUT	Criteria	Big spike	Dual phase	Large variations	Quickly varying	Slowly varying	Steep tri phase
RobusT2Scale		973ms	537ms	509ms	451ms	423ms	498ms
		3.2	3.8	5.1	5.3	3.7	3.9
Overprovisioning		354ms	411ms	395ms	446ms	371ms	491ms
		6	6	6	6	6	6
Under provisioning		1465ms	1832ms	1789ms	1594ms	1898ms	2194ms
		2	2	2	2	2	2

- RobusT2Scale is superior to under-provisioning in terms of guaranteeing the SLA and does not require excessive resources
- RobusT2Scale is superior to over-provisioning in terms of guaranteeing required resources while guaranteeing the SLA

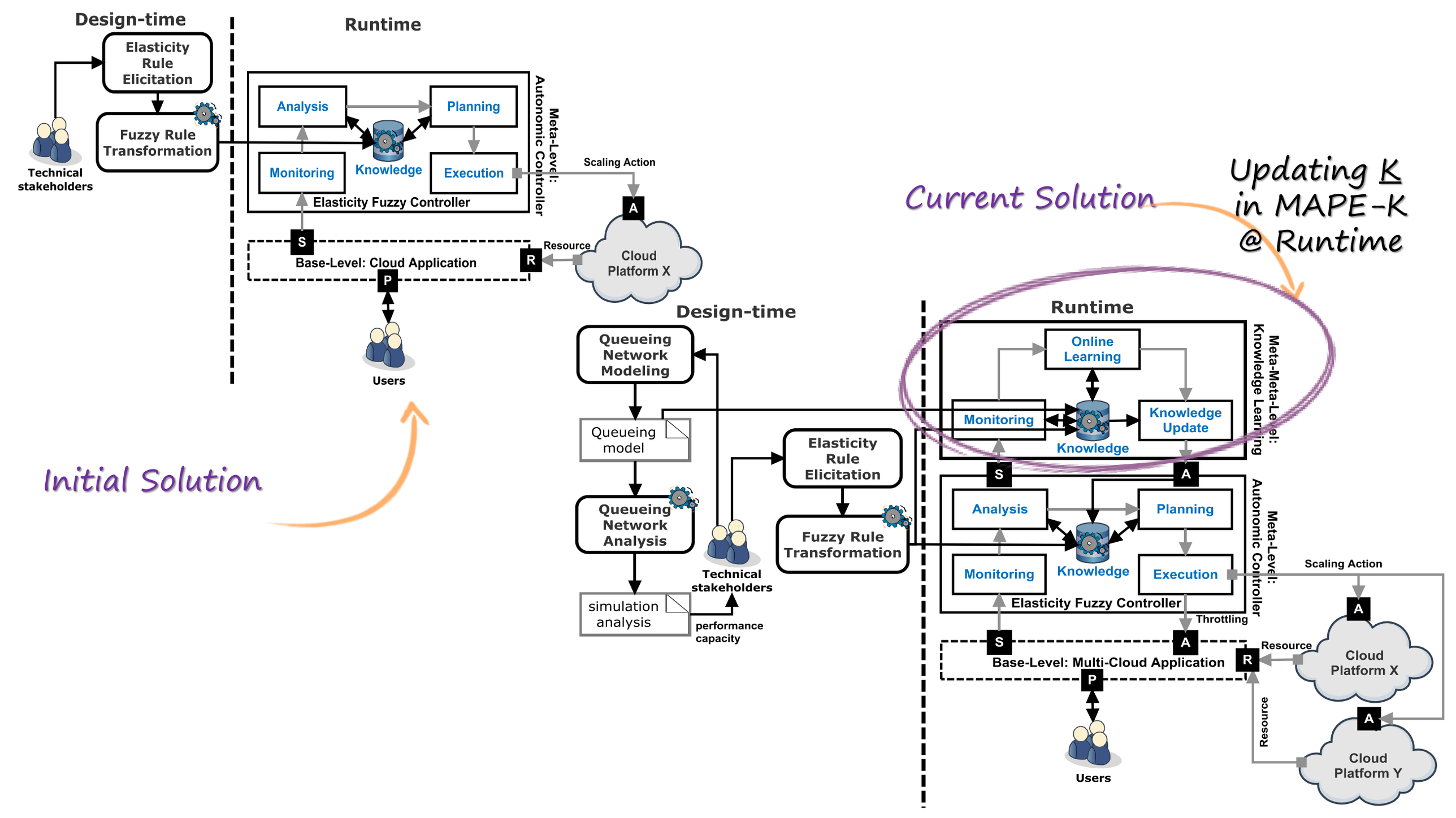
SLA: $rt_{95} \leq 600ms$
For every 10s control interval

Robustness of RobusT2Scale

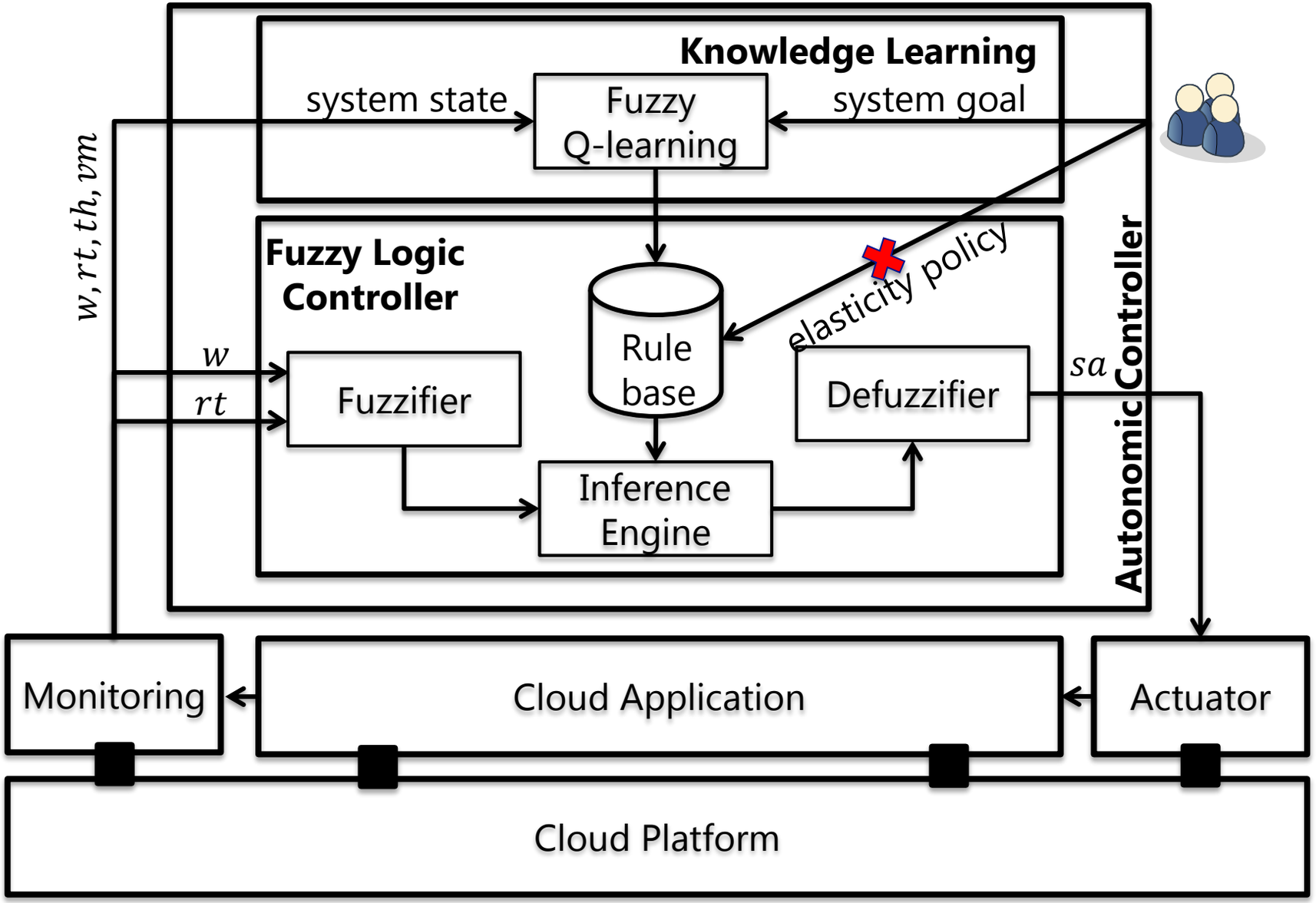


Part

Self-Learning Controller


Design-time
Runtime
Elasticity Rule Elicitation
Fuzzy Rule Transformation
Technical stakeholders
Analysis
Planning
Monitoring
Knowledge
Execution
Elasticity Fuzzy Controller
Meta-Level: Autonomic Controller
Scaling Action
A
S
R
P
Resource
Base-Level: Cloud Application
Cloud Platform X
Users
Initial Solution
Design-time
Queueing Network Modeling
Queueing model
Queueing Network Analysis
simulation analysis
Technical stakeholders
performance capacity
Elasticity Rule Elicitation
Fuzzy Rule Transformation
Current Solution
Updating K in MAPE-K @ Runtime
Runtime
Online Learning
Monitoring
Knowledge
Knowledge Update
Meta-Meta-Level: Knowledge Learning
Analysis
Planning
Monitoring
Knowledge
Execution
Elasticity Fuzzy Controller
Throttling
Meta-Level: Autonomic Controller
Scaling Action
Base-Level: Multi-Cloud Application
Resource
Cloud Platform X
Cloud Platform Y
Users

FQL4KE: Fuzzy Q-Learning for Knowledge Evolution



Algorithm 1 : Fuzzy Q-Learning

Require: γ, η

1: Initialize q-values:
$q[i,j] = 0,\ 1 < i < N\ ,\ 1 < j < J$

2: Select an action for each fired rule:
$a_i = argmax_k q[i,k]$ with probability $1-\epsilon$
$a_i = random\{a_k, k = 1, 2, \cdots, J\}$ with probability ϵ

3: Calculate the control action by the fuzzy controller:
$a = \sum_{i=1}^{N} \mu_i(x) \times a_i,$
where $\alpha_i(s)$ is the firing level of the rule i

4: Approximate the Q function from the current q-values and the firing level of the rules:
$Q(s(t),a) = \sum_{i=1}^{N} \alpha_i(s) \times q[i,a_i],$
where $Q(s(t),a)$ is the value of the Q function for the state current state $s(t)$ in iteration t and the action a

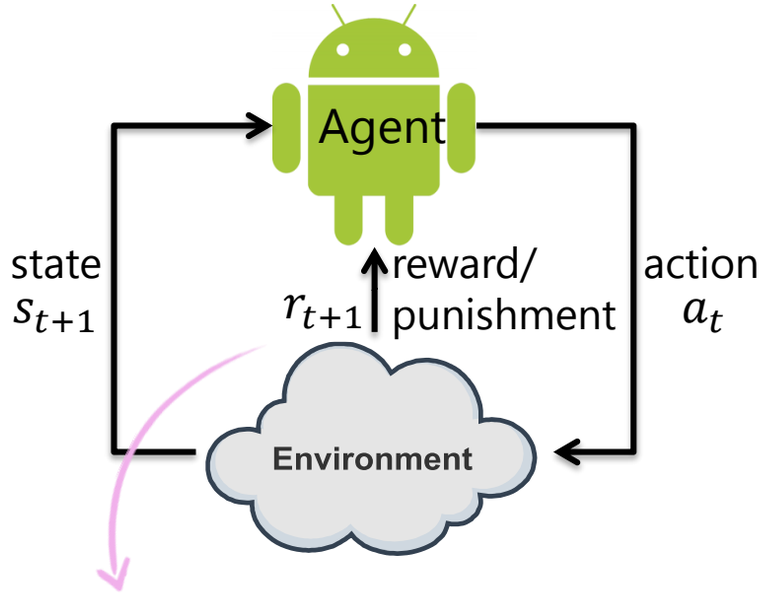
5: Take action a and let system goes to the next state $s(t+1)$.

6: Observe the reinforcement signal, $r(t+1)$ and compute the value for the new state:
$V(s(t+1)) = \sum_{i=1}^{N} \alpha_i(s(t+1)).max_k(q[i,q_k]).$

7: Calculate the error signal:
$\Delta Q = r(t+1) + \gamma \times V_t(s(t+1)) - Q(s(t),a),$
where γ is a discount factor

8: Update q-values:
$q[i,a_i] = q[i,a_i] + \eta \cdot \Delta Q \cdot \alpha_i(s(t)),$
where η is a learning rate

9: Repeat the process for the new state until it converges

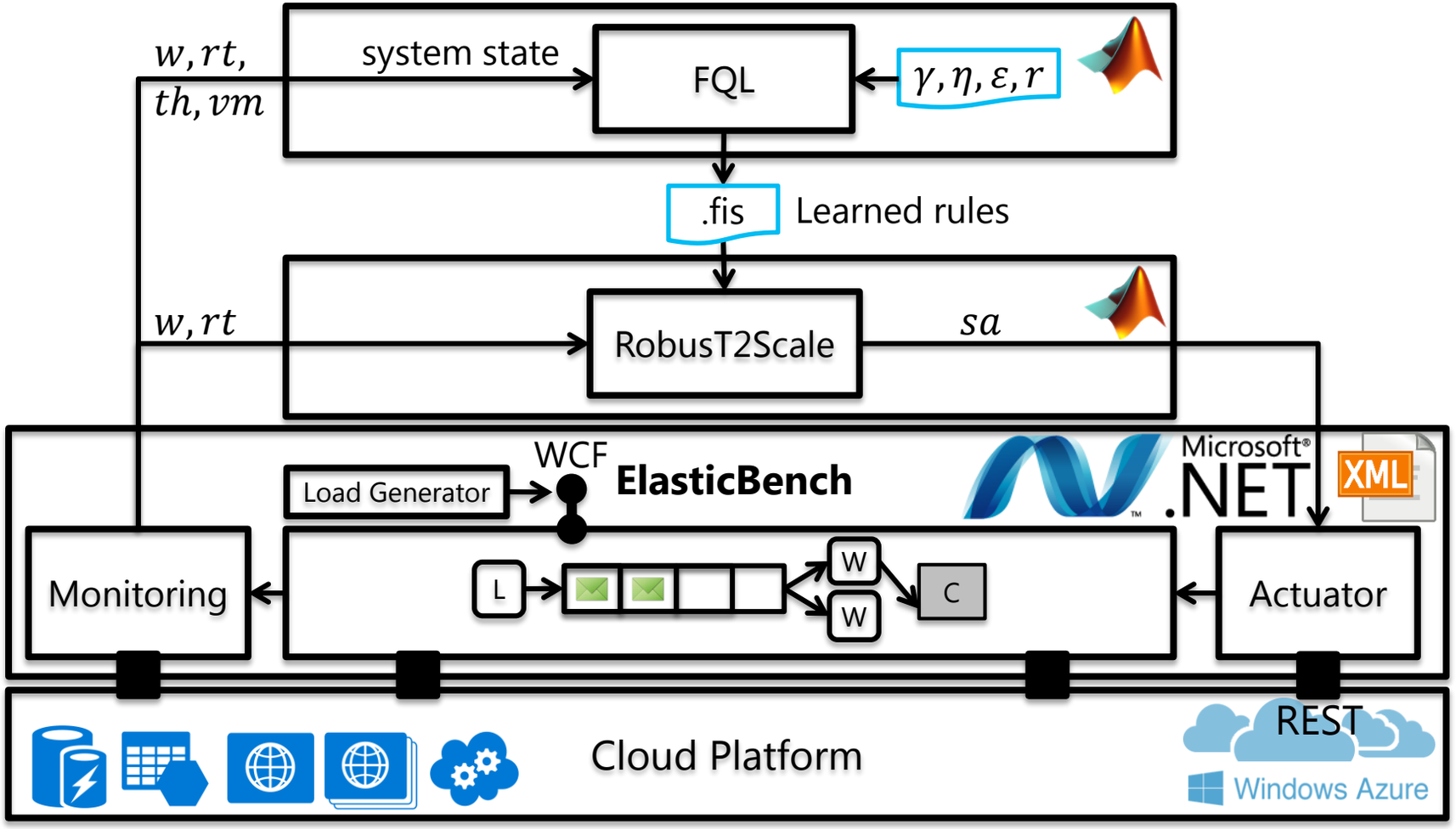


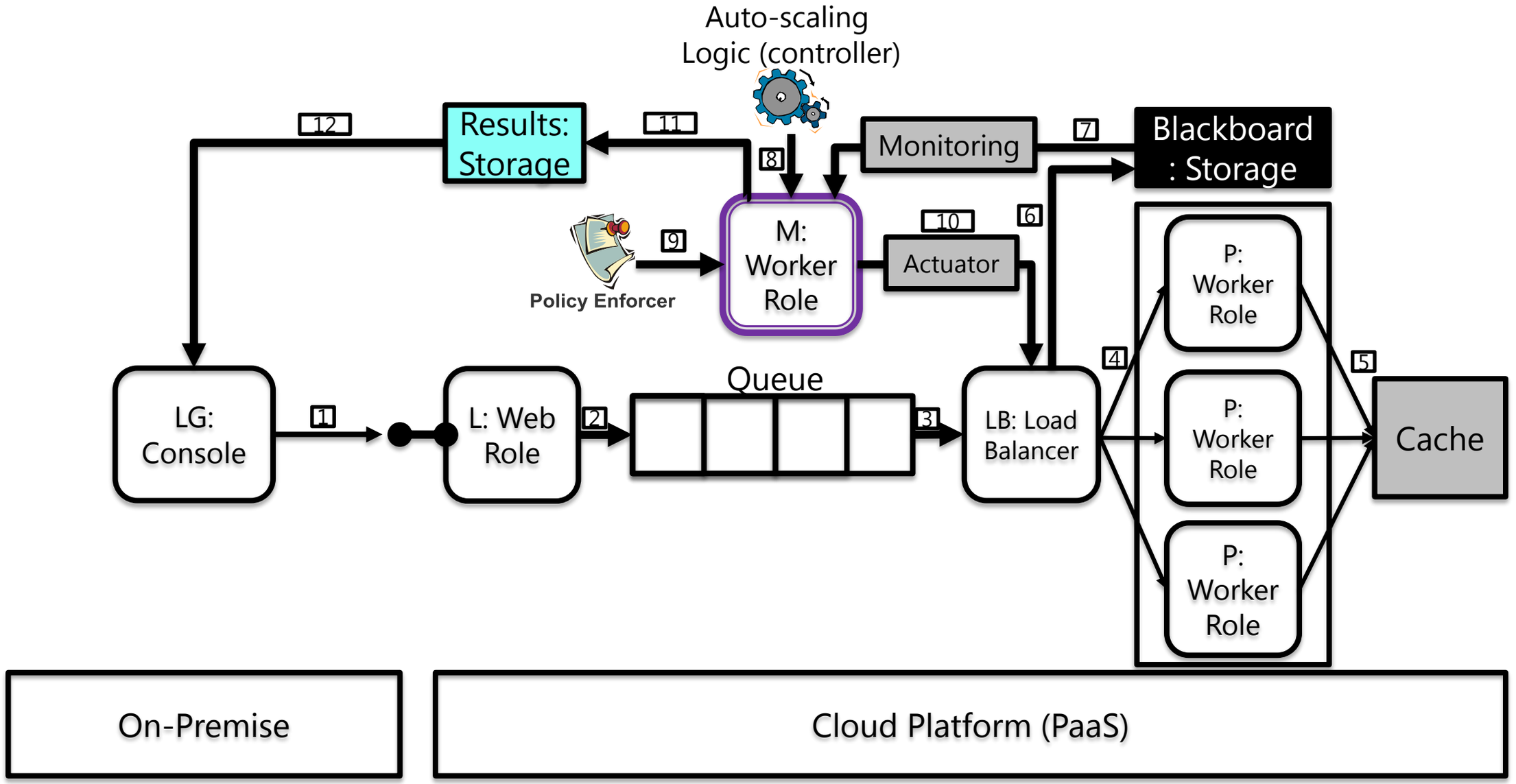
$$r(t) = U(t) - U(t-1),$$

$$U(t) = w_1 \cdot \frac{th(t)}{th_{max}} + w_2 \cdot (1 - \frac{vm(t)}{vm_{max}}) + w_3 \cdot (1 - H(t))$$

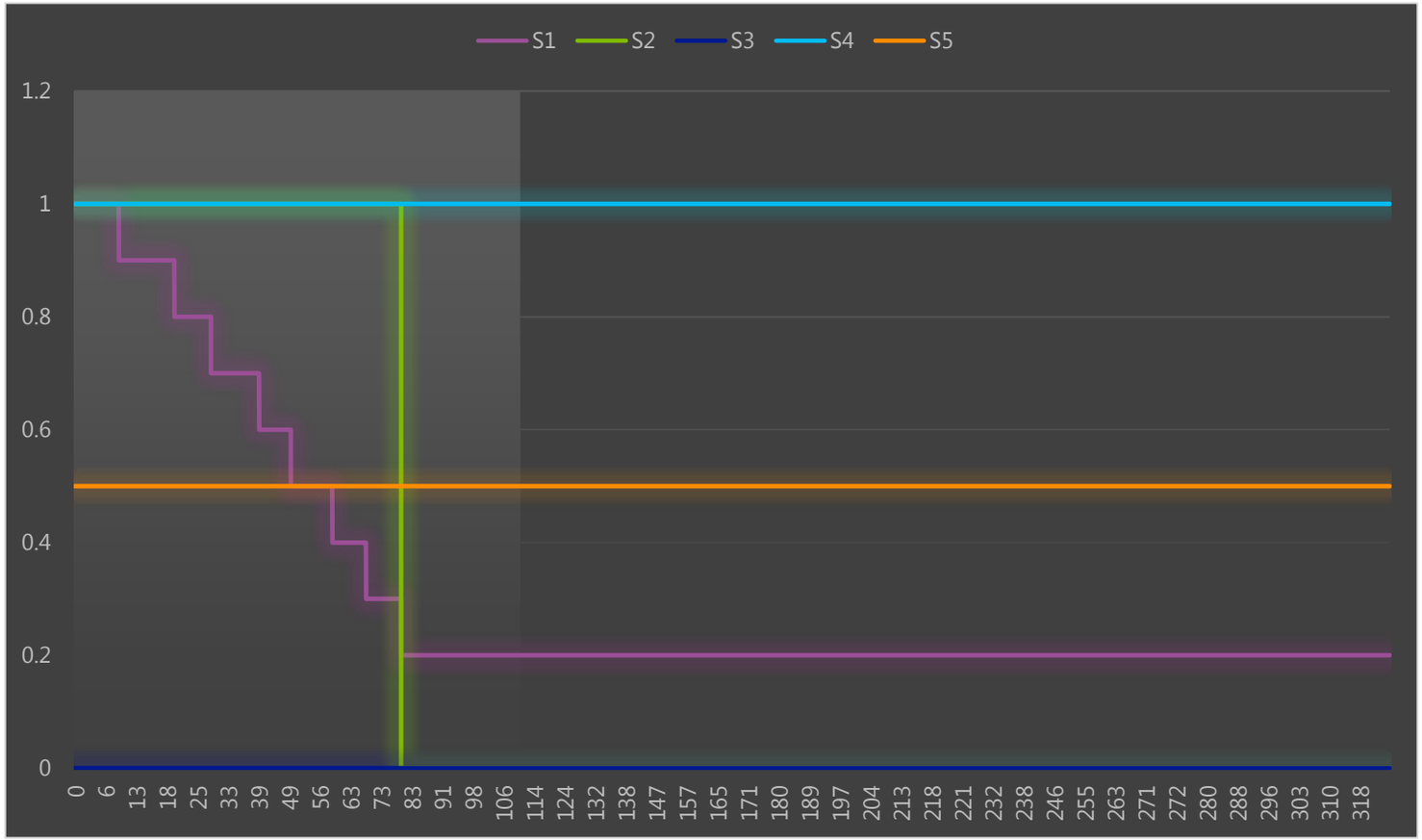
$$H(t) = \begin{cases} \frac{(rt(t) - rt_{des})}{rt_{des}} & rt_{des} \leq rt(t) \leq 2 \cdot rt_{des} \\ 1 & rt(t) \geq 2 \cdot rt_{des} \\ 0 & rt(t) \leq rt_{des} \end{cases}$$

ElasticBench: A Cloud Application Framework

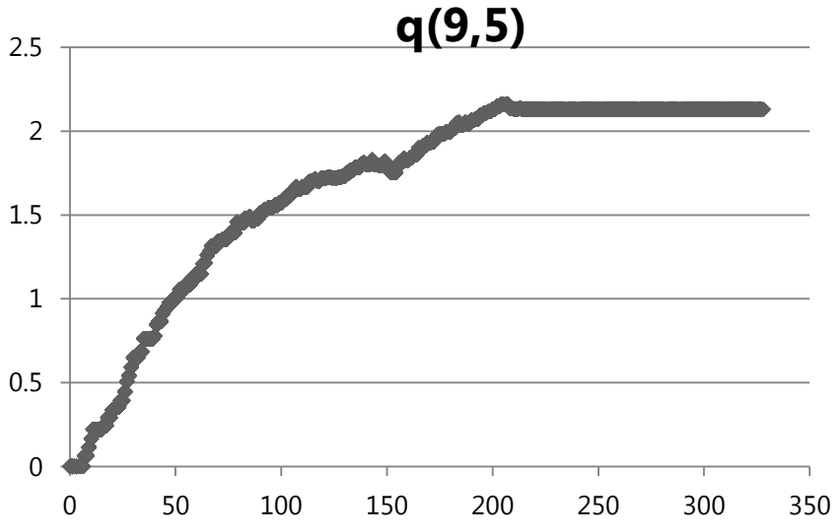


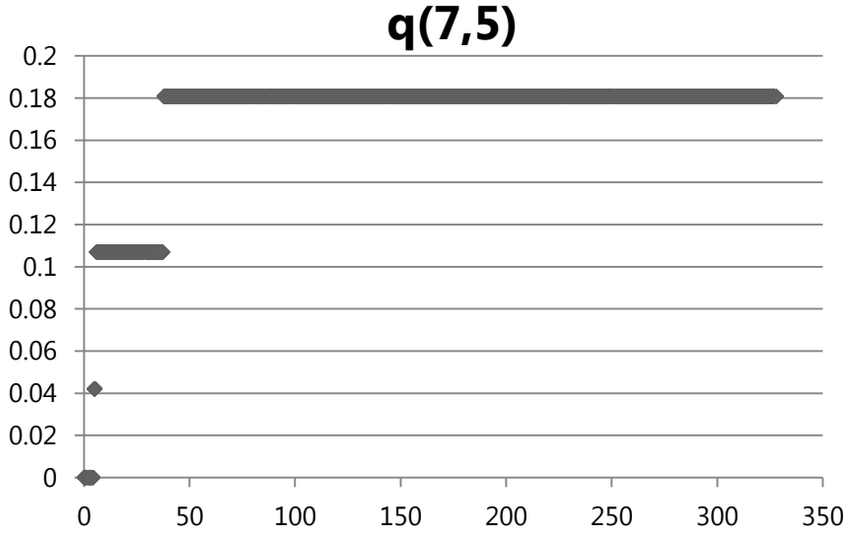

Auto-scaling
Logic (controller)
Results:
Storage
12
11
8
Monitoring
7
Blackboard
: Storage
9
M:
Worker
Role
10
Actuator
6
Policy Enforcer
P:
Worker
Role
4
5
LG:
Console
1
L: Web
Role
2
Queue
3
LB: Load
Balancer
P:
Worker
Role
Cache
P:
Worker
Role
On-Premise
Cloud Platform (PaaS)

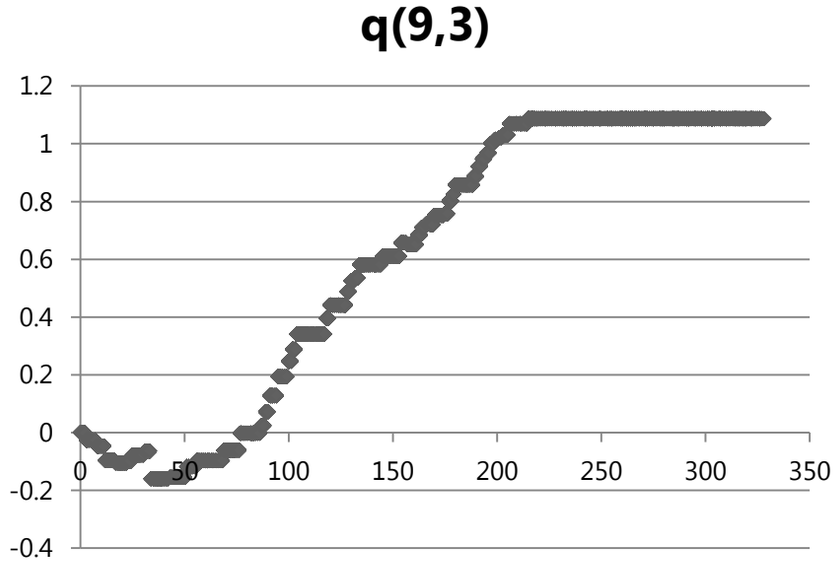
Learning Strategies

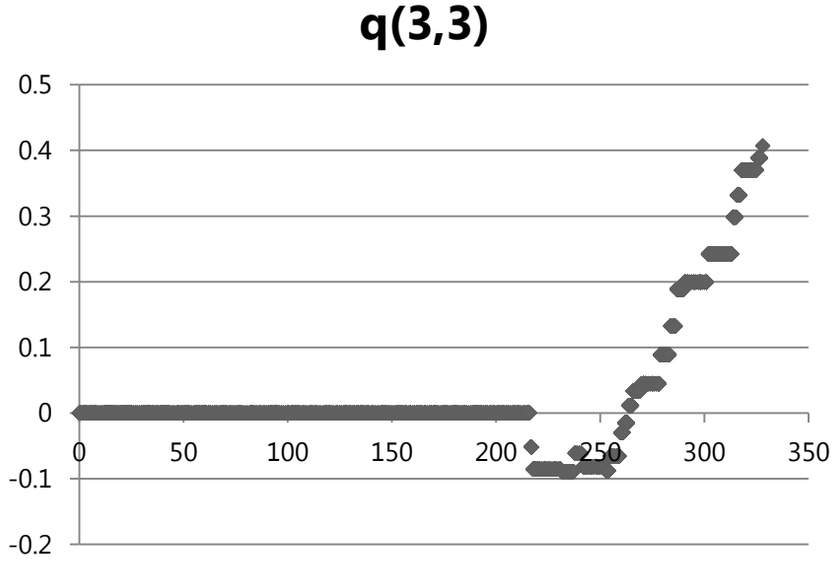


Q-value Evolution

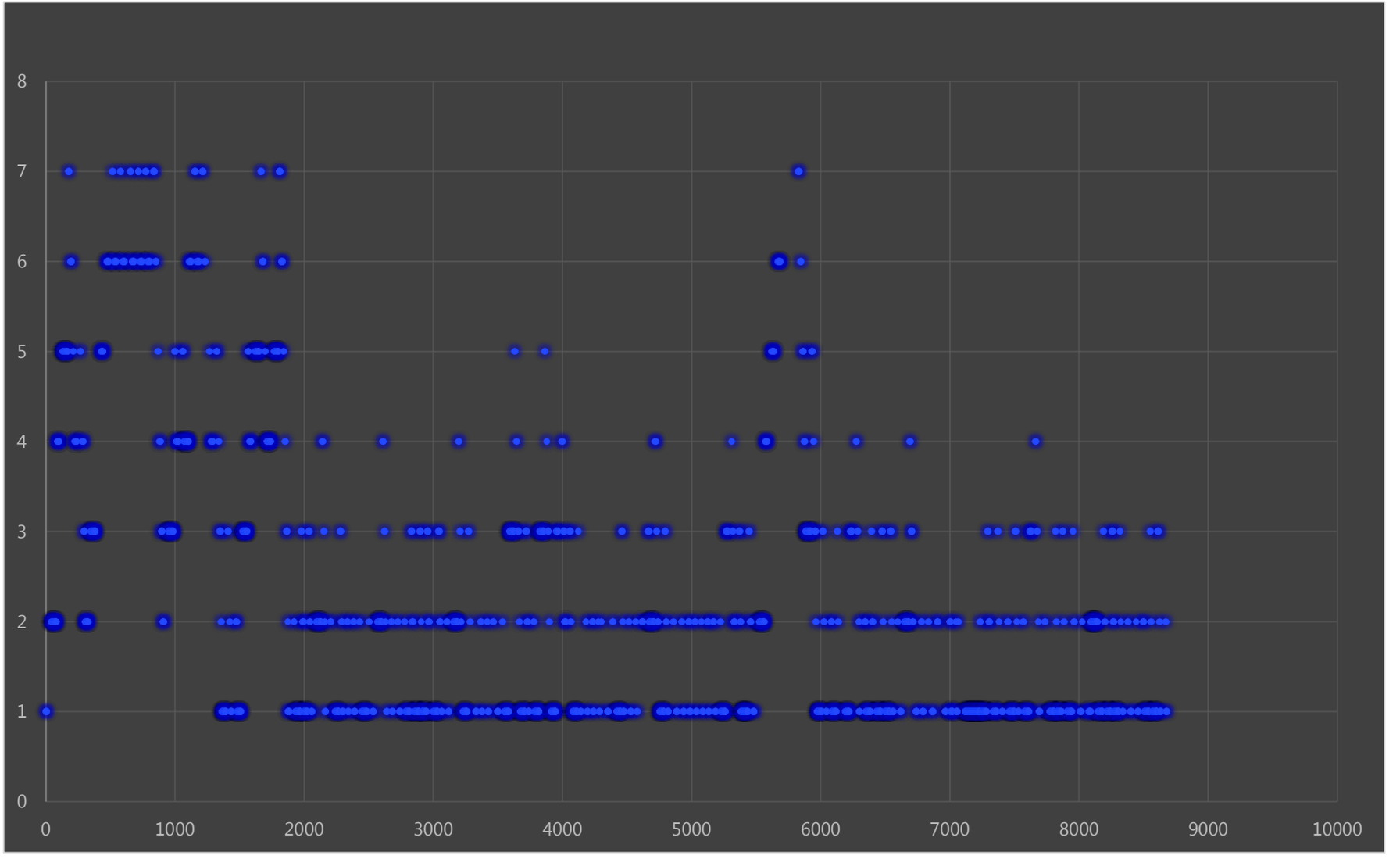




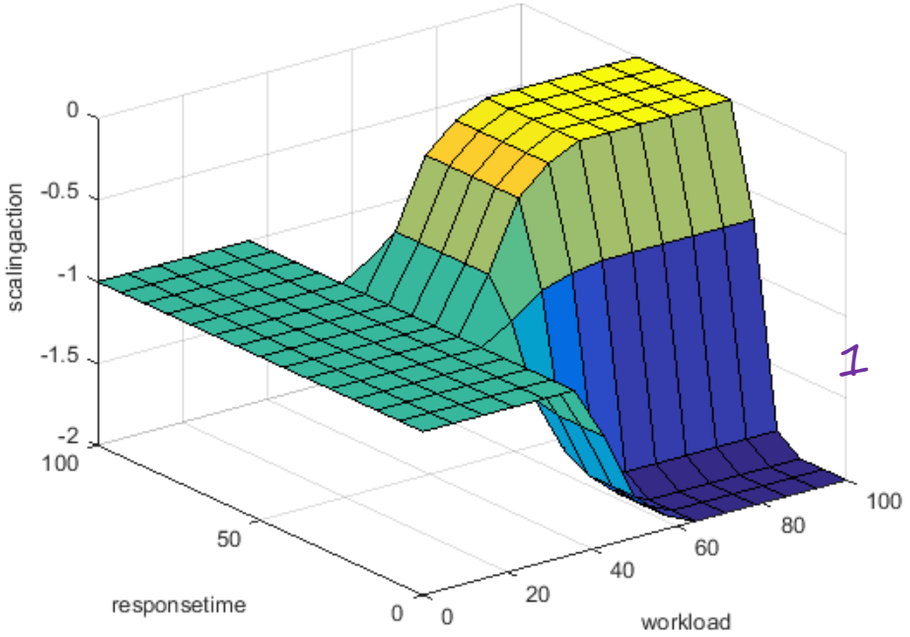




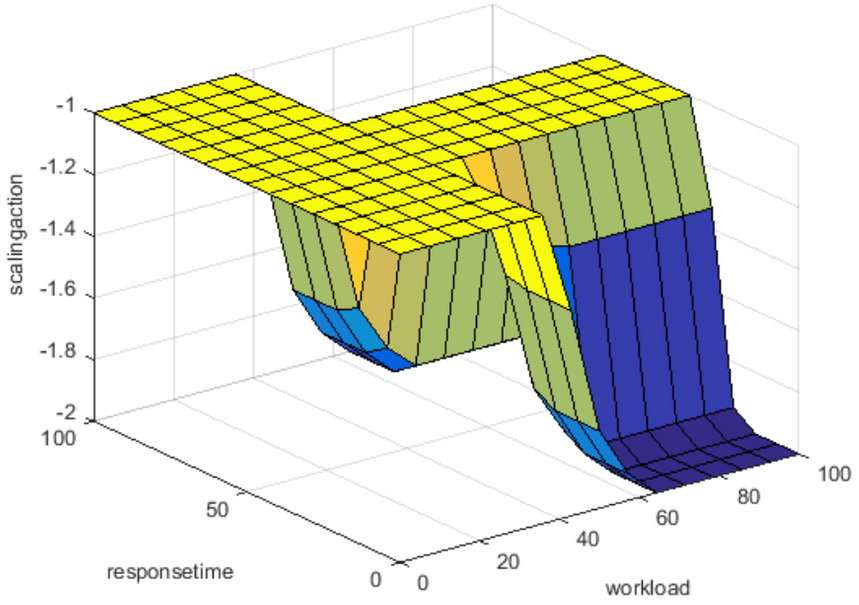
Temporal Evolution of Acquired Nodes



Control Surface Evolution

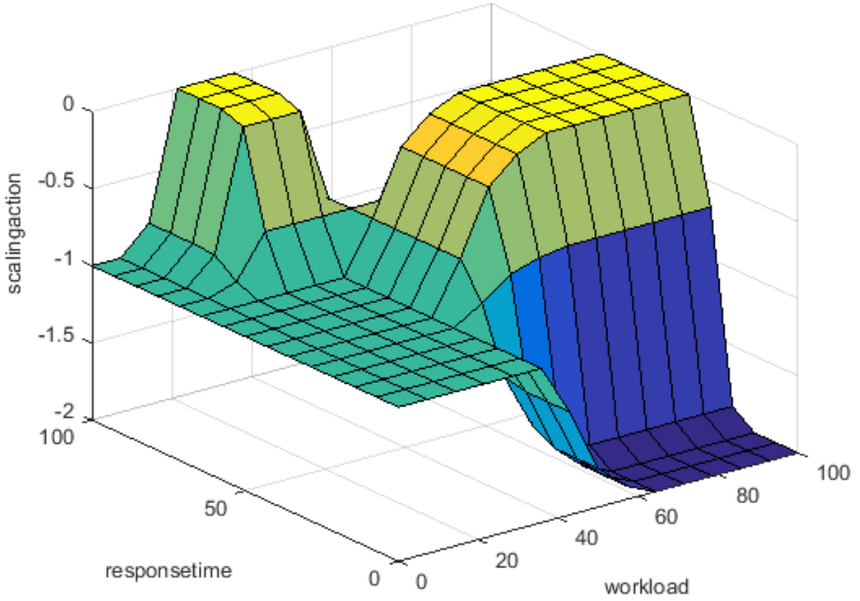


1

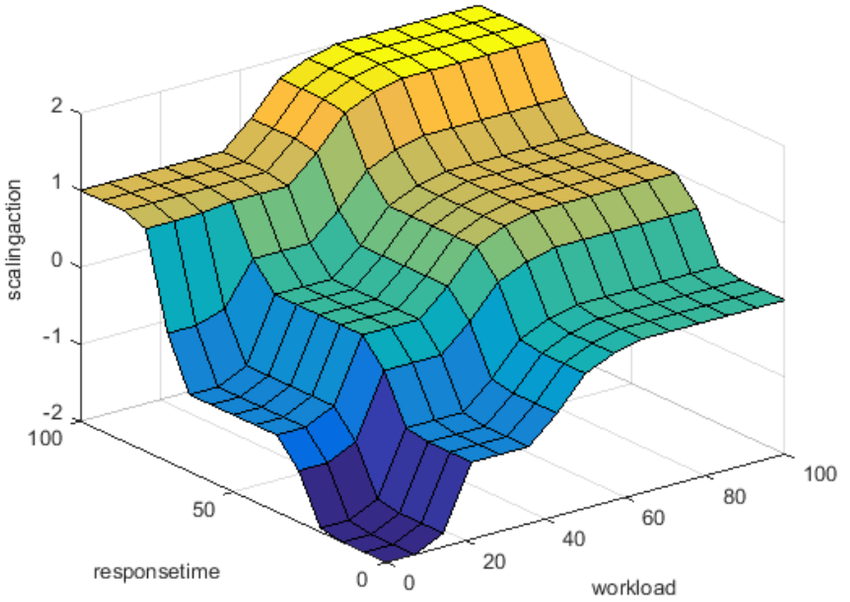


2

3



4

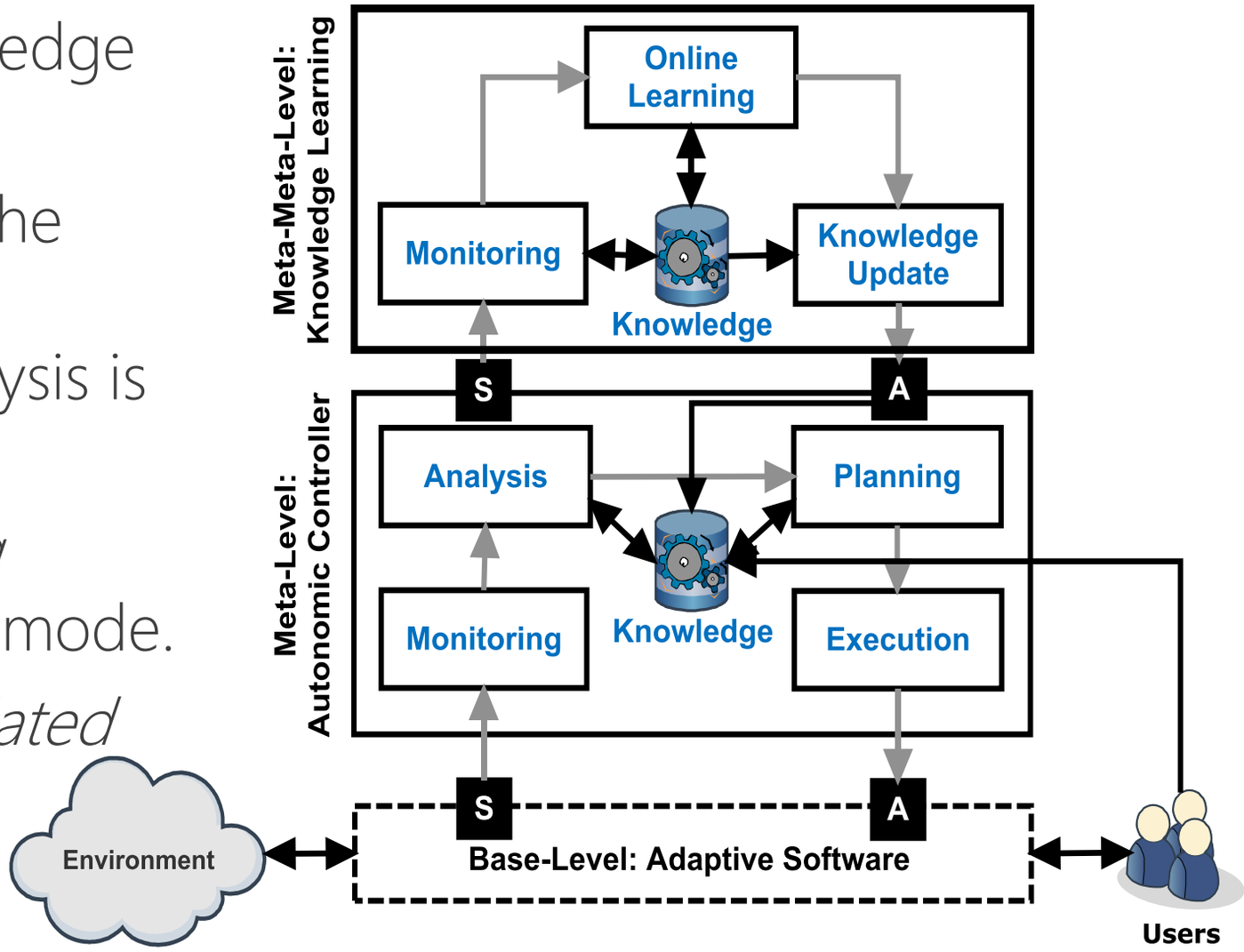


Exploration/exploitation strategies under different workloads

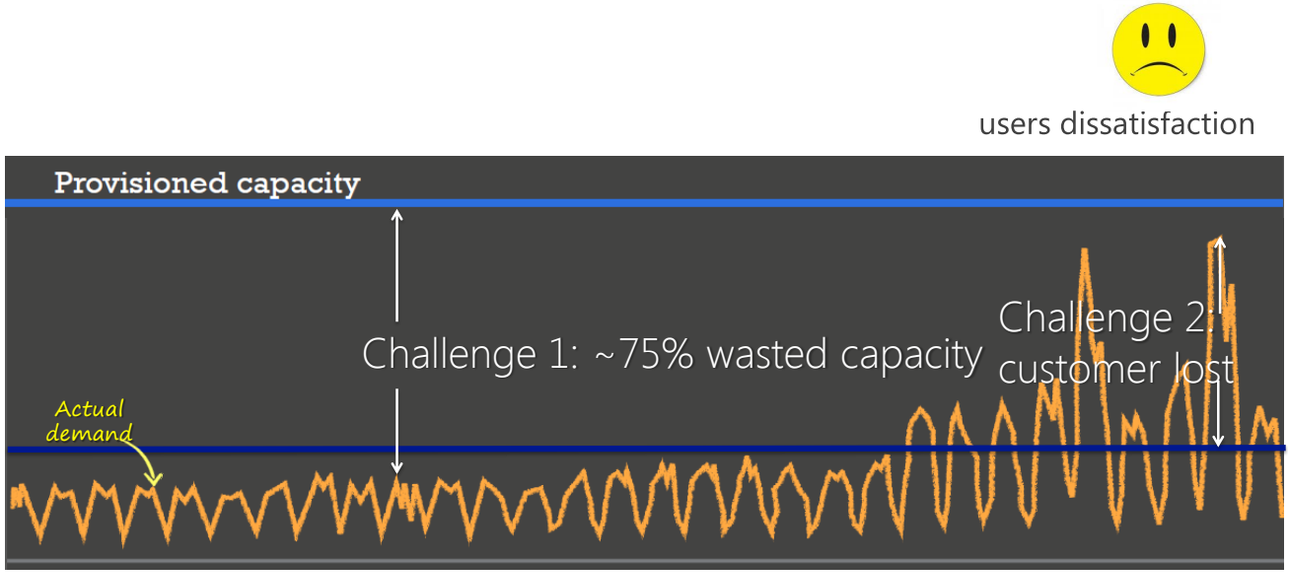
Strategy	Criteria	Big spike	Dual phase	Large variations
S1	$rt_{95\%}, \overline{vm}$	1212ms, 2.2	548ms, 3.6	991ms, 4.3
	node change	390	360	420
	convergence	32	34	40
S2	$rt_{95\%}, \overline{vm}$	1298ms, 2.3	609ms, 3.8	1191ms, 4.4
	node change	412	376	429
	convergence	38	36	87
S3	$rt_{95\%}, \overline{vm}$	1262ms, 2.4	701ms, 3.8	1203ms, 4.3
	node change	420	387	432
	convergence	30	29	68
S4	$rt_{95\%}, \overline{vm}$	1193ms, 3.2	723ms, 4.1	1594ms, 4.8
	node change	487	421	453
	convergence	328	328	328
S5	$rt_{95\%}, \overline{vm}$	1339ms, 3.2	729ms, 3.8	1233ms, 5.1
	node change	410	377	420
	convergence	–	–	–
Azure	$rt_{95\%}, \overline{vm}$	1409ms, 3.3	712ms, 4.0	1341ms, 5.5
	node change	330	299	367
	convergence	–	–	–
		Quickly varying	**Slowly varying**	**Steep tri phase**
S1	$rt_{95\%}, \overline{vm}$	1319ms, 4.4	512ms, 3.6	561ms, 3.4
	node change	432	355	375
	convergence	65	24	27
S2	$rt_{95\%}, \overline{vm}$	1350ms, 4.8	533ms, 3.6	603ms, 3.4
	node change	486	370	393
	convergence	98	45	28
S3	$rt_{95\%}, \overline{vm}$	1287ms, 4.9	507ms, 3.7	569ms, 3.4
	node change	512	372	412
	convergence	86	40	23
S4	$rt_{95\%}, \overline{vm}$	2098ms, 5.9	572ms, 5.0	722ms, 4.8
	node change	542	411	444
	convergence	328	328	328
S5	$rt_{95\%}, \overline{vm}$	1341ms, 5.3	567ms, 3.7	512ms, 3.9
	node change	479	366	390
	convergence	–	–	–
Azure	$rt_{95\%}, \overline{vm}$	1431ms, 5.4	1101ms, 3.7	1412ms, 4.0
	node change	398	287	231
	convergence	–	–	–

Online Knowledge Learning

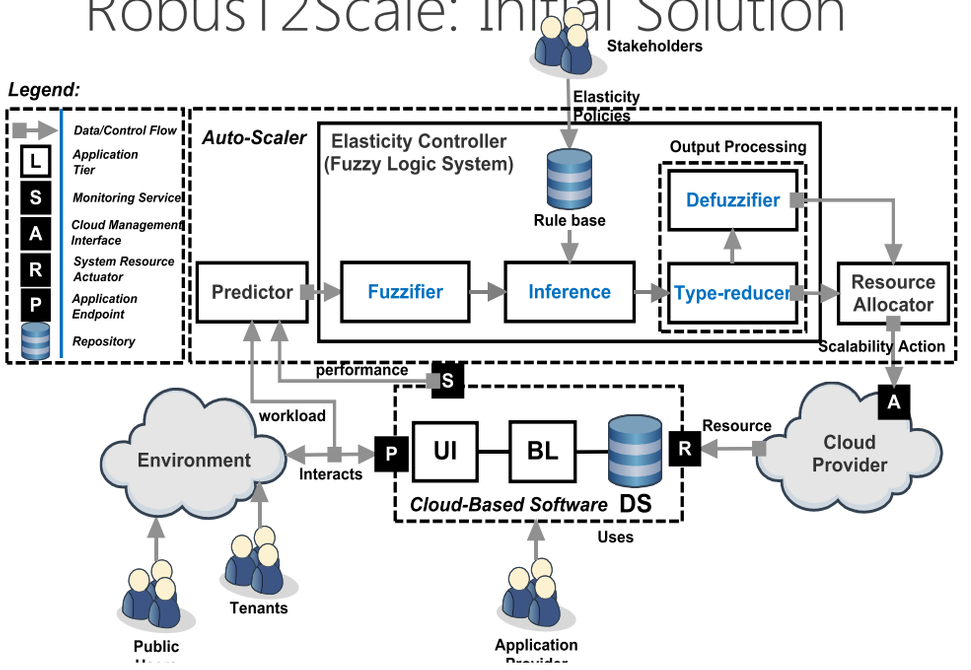
- **Implication**: No priori knowledge is needed, controller can gain knowledge at runtime
- The learning process is executed the system enters to the *monitored operation* mode: statistics for analysis is collected.
- Control is returned to the *learning process* which enters the learning mode.
- Finally the knowledge base is *updated*



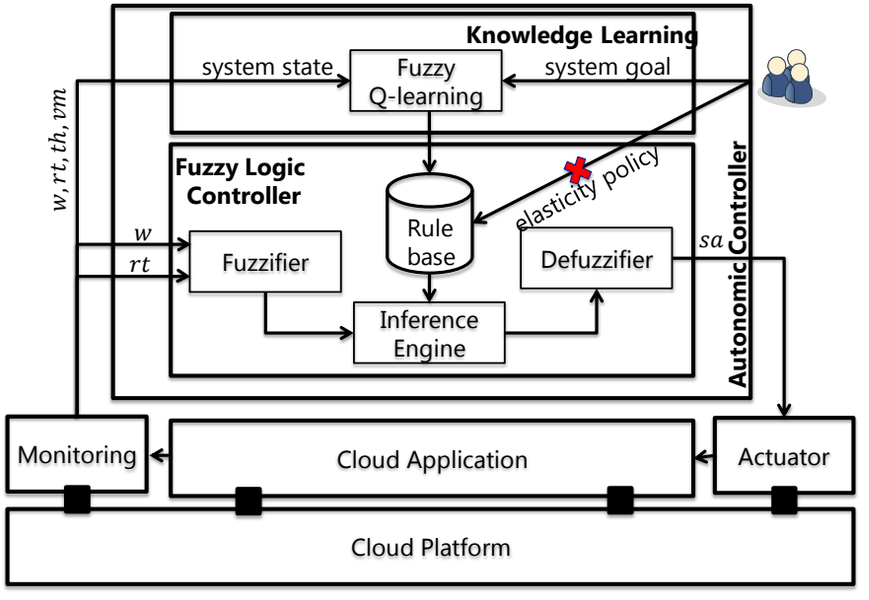
Motivations & Challenges



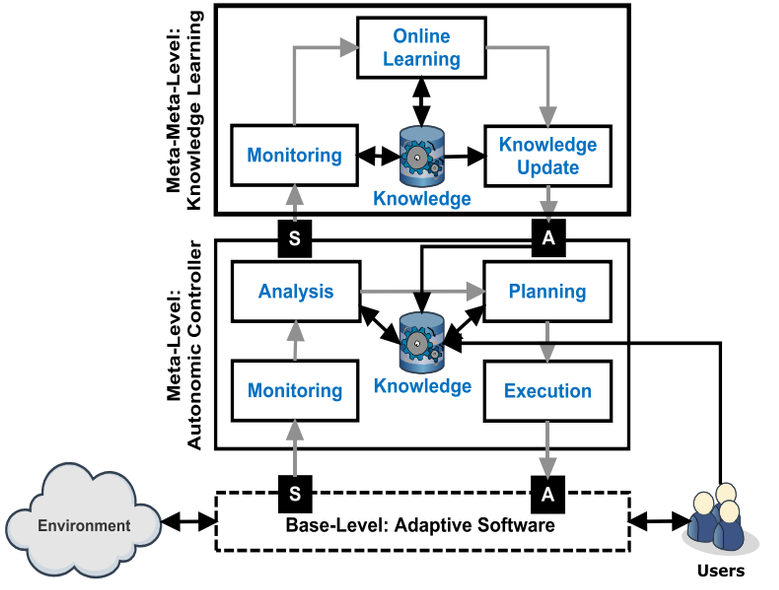
RobusT2Scale: Initial Solution



FQL4KE: Augmenting with Online Learning



Online Knowledge Learning



Thank you!

Claus Pahl

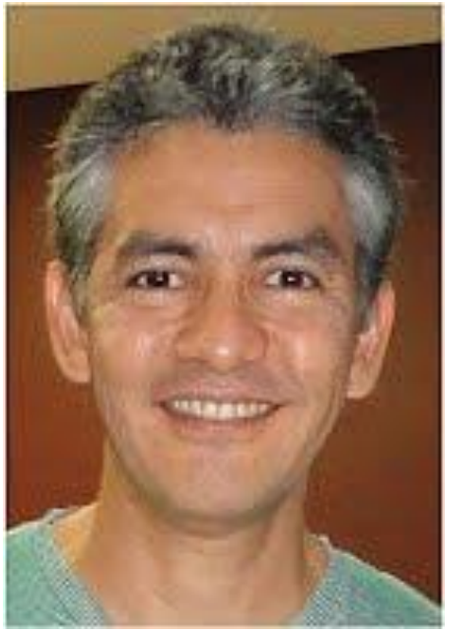
Aakash Ahmad

Soodeh Farokhi

Amir Sharifloo

Armin Balalaie

Hamed Jamshidi



Nabor Mendonca

Reza Teimourzadegan

Brian Carroll

Slides?

Code?

More Details?

https://github.com/pooyanjamshidi

http://www.slideshare.net/pooyanjamshidi/

http://computing.dcu.ie/~pjamshidi/PDF/SEAMS2014.pdf