Resource Provisioning for Virtualized Server Applications

Eva Kalyvianaki, Themistoklis Charalambous, Steven Hand

Computer Laboratory
University of Cambridge
ek264@cl.cam.ac.uk

ICAC
June 2009

Outline

What?

Self-managing feedback controllers that online update the CPU allocations for virtualized applications:

- applicable to many systems
- operational across different management conditions
- self-configurable with regards to the input parameters

Virtualized Data Center

What is virtualization?

A technique to transforms **one** machine into **multiple** virtual ones.

Characteristics:

- Virtual Machine (VM): virtual execution environment
- performance isolation
- dynamic resource allocation

Server Consolidation:

- server application deployment in VMs
- VM deployment on any physical server
- increase machine utilisation, $140B more than needed
- reduce power, 50% of spendings on server management

What and How?

What?

On-demand CPU provision each VM, avoid over- and under-utilisation, so as to have other applications co-located

Challenges:

- unknown workloads
- variable workloads
- application diversity

How?

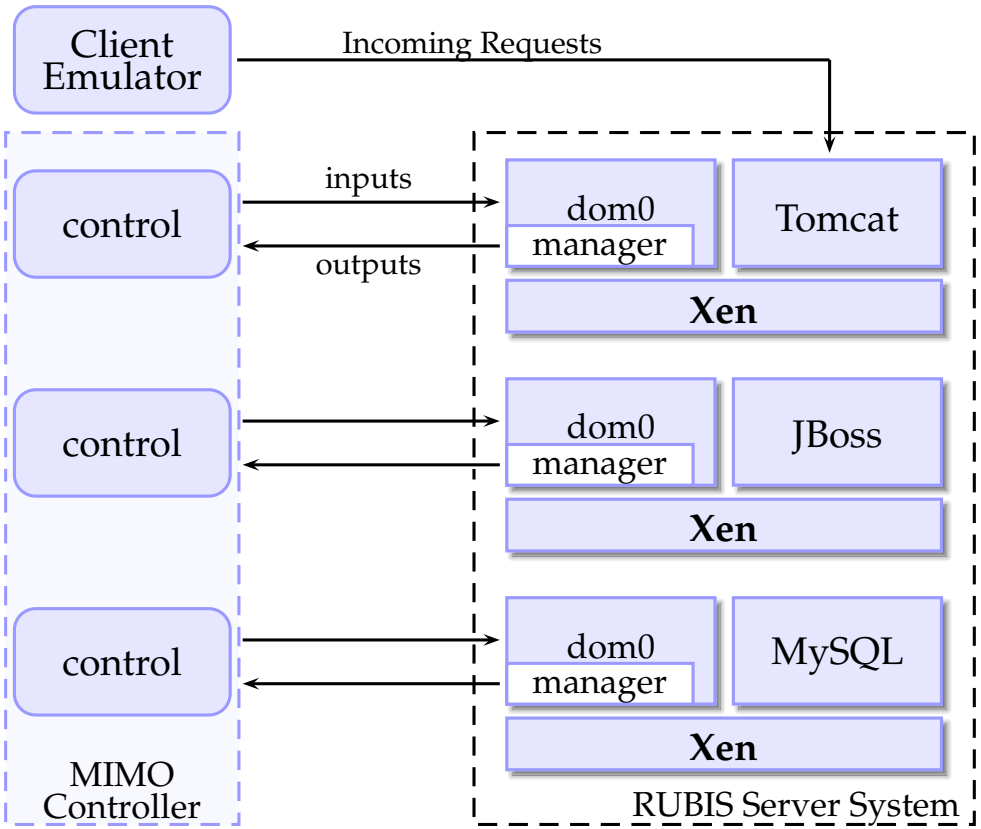
Using feedback control to periodically update the allocations

Properties:

- short history window
- control theory to deal with noise

System Overview

Feedback controllers to online update the CPU allocations for virtualized applications



- 3-tier Rubis auction benchmark
- control loop/interval, e.g., every a few seconds
- CPU allocation: max CPU resource a VM can use
- SISO and MIMO controllers

System Identification

Observations:

1. a good indication of the required allocation is the utilisation
2. when any of the components is not CPU-bounded the server has good performance

Control Signals:

input: CPU allocation
output: mean utilisation over some interval

Model: the allocation follows the utilisation: a tracking problem

multiplicative model: $u = c * a, 0 < c < 1$

Kalman Filter - simple definition

What is?

A data **processing** method, that estimates the state of a **linear** system using **noisy** measurements in a **recursive** manner.

- **optimal filter**
- **noisy measurements:** CPU utilisations
- **recursive computations:** reduce computation overhead

System Model for the Kalman controllers

Real Utilisation Signal

$$v_{k+1} = v_k + t_k$$

t normally distributed

Allocation Signal

$$a_{k+1} = a_k + z_k$$

z models the system evolution

normally distributed with variance Q

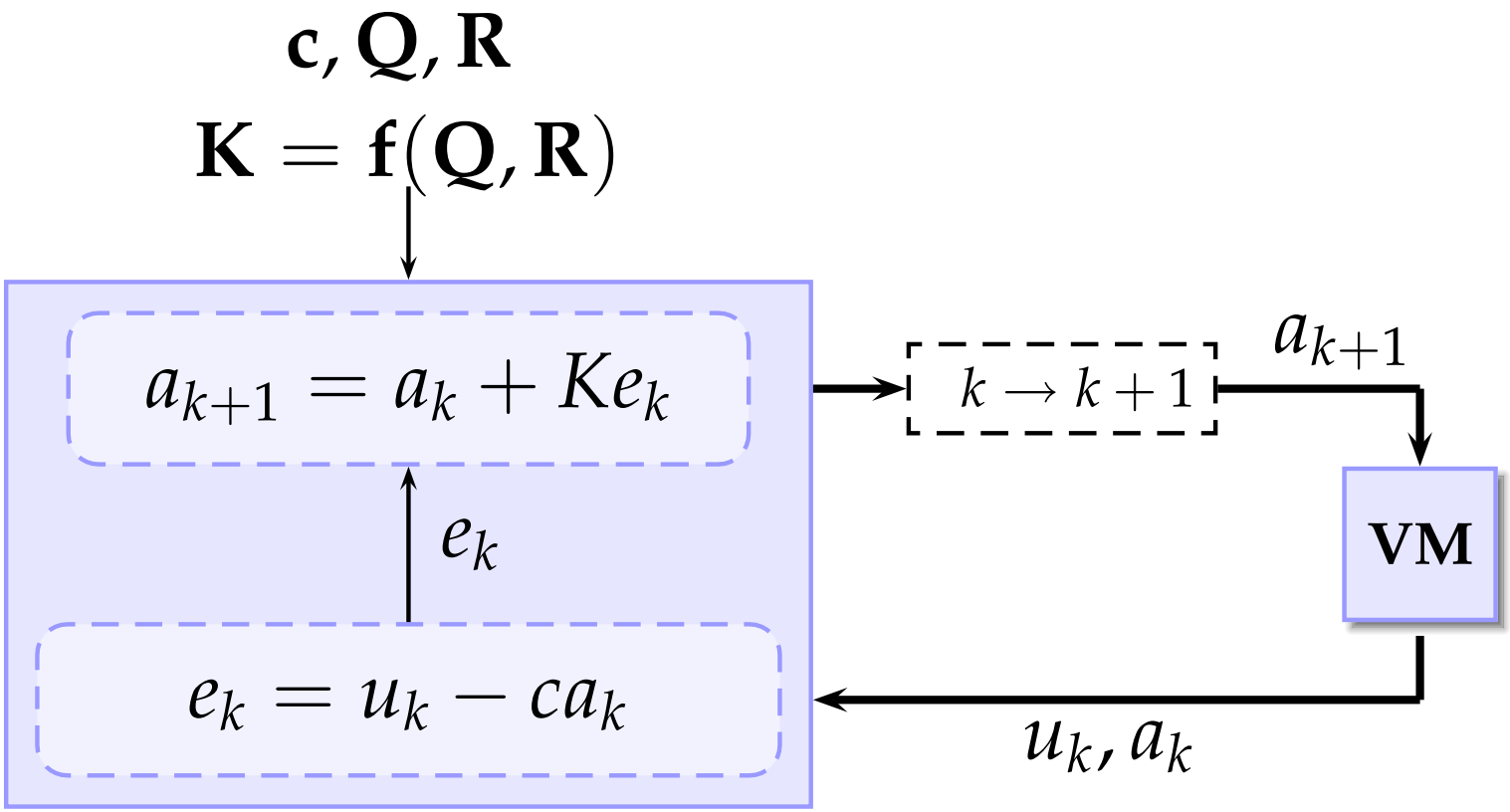
Observed Utilisation (measurement)

$$u_k = ca_k + w_k$$

w models the measurement noise

normally distributed with variance R

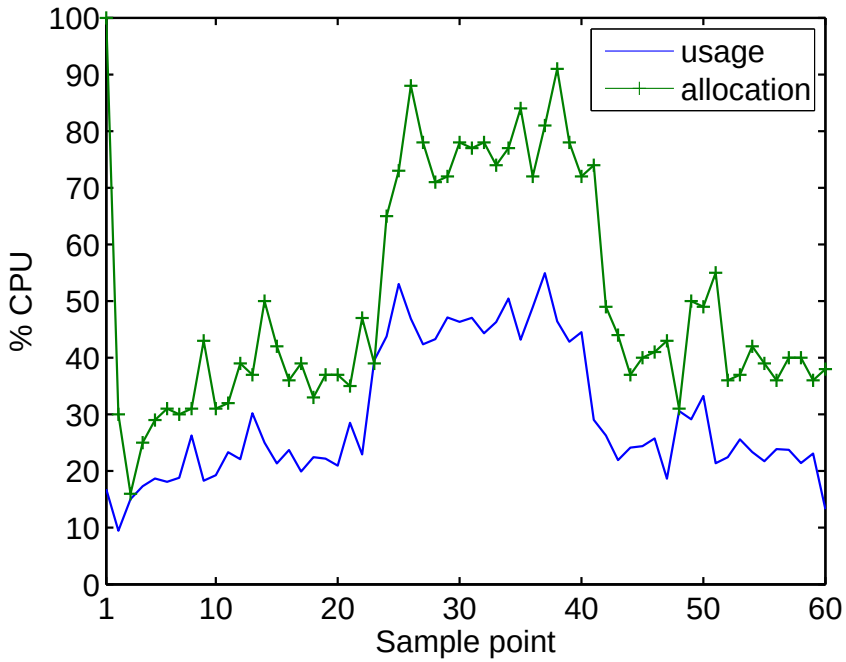
Kalman Basic Controller - KBC



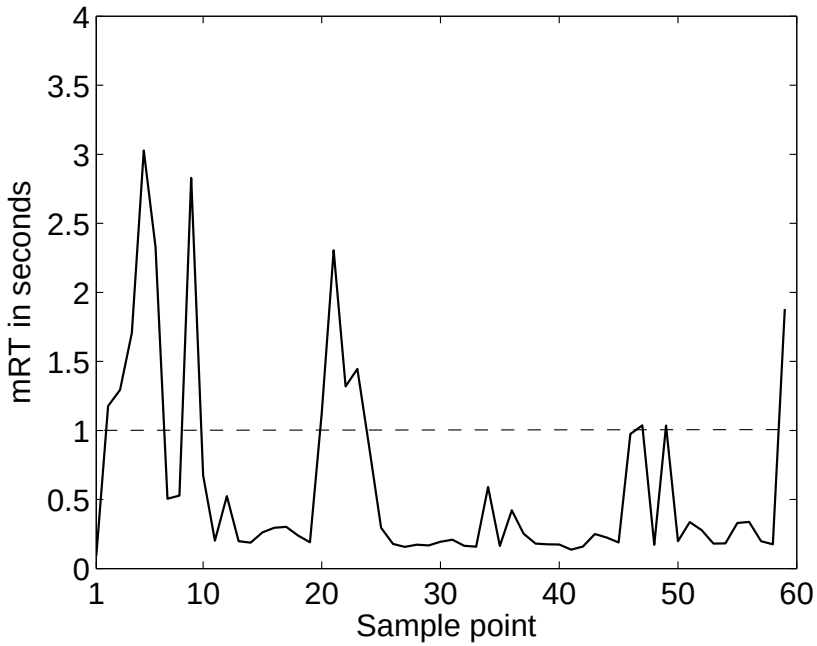
- controller goal: to maintain the allocation at $\frac{1}{c}$% of the utilisation
- Kalman gain K uses system information
- off-line computed input parameters, $c = 60\%$

KBC - Results 1/3

Tomcat CPU



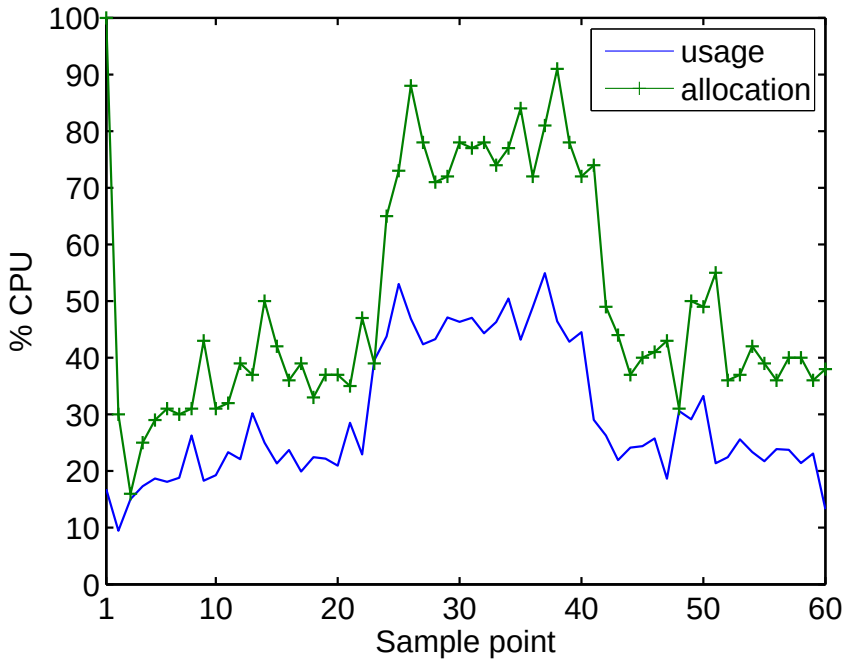
Response Time



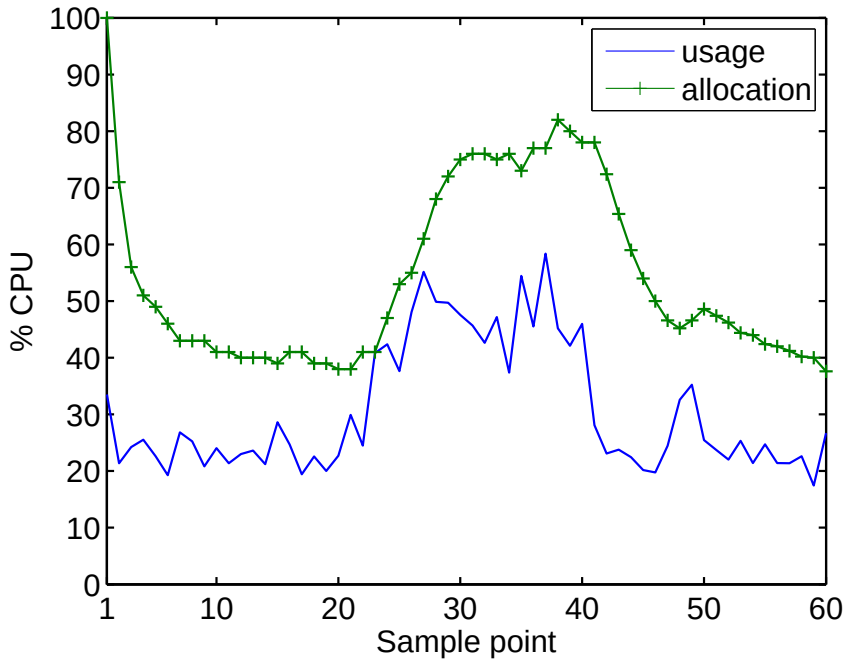
- follows the utilisation changes
- but prone to **all** fluctuations

KBC - Results 2/3

Tomcat CPU, K=1.6, original



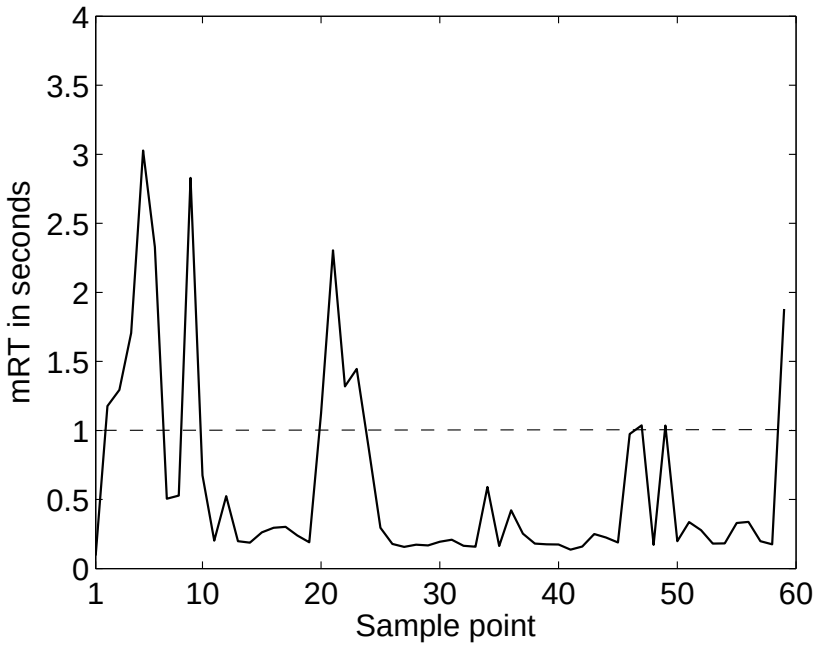
Tomcat CPU, K=0.4



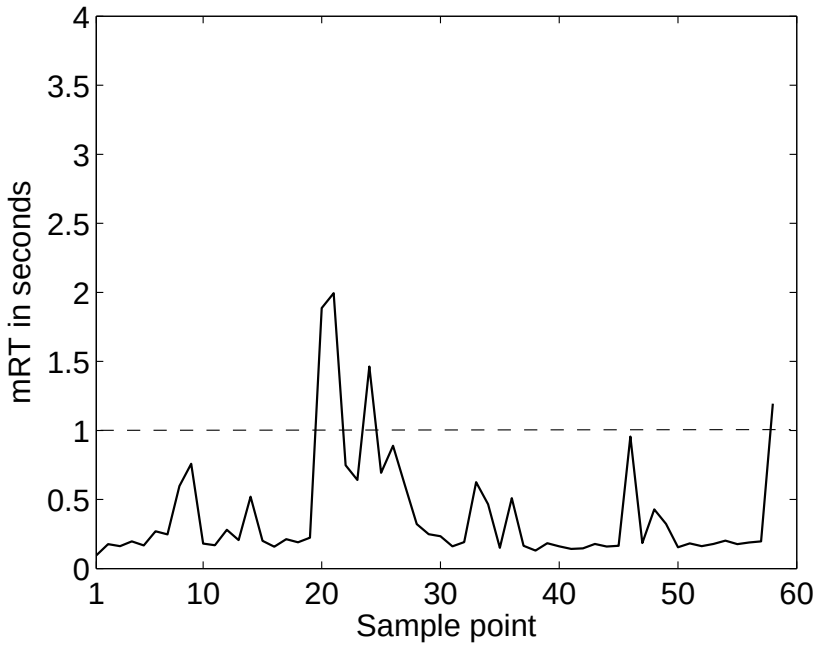
- depending on $K = f(Q, R)$, KBC filters out noise
- Q relates to system information, $Q = var(\frac{u}{c})$
- $R = 1$ relates to measurement accuracy
- $K = f(\uparrow Q, \downarrow R)$

KBC - Results 3/3

Response Time, K=1.6, original



Response Time, K=0.4



for smaller K, KBC reacts slower to changes

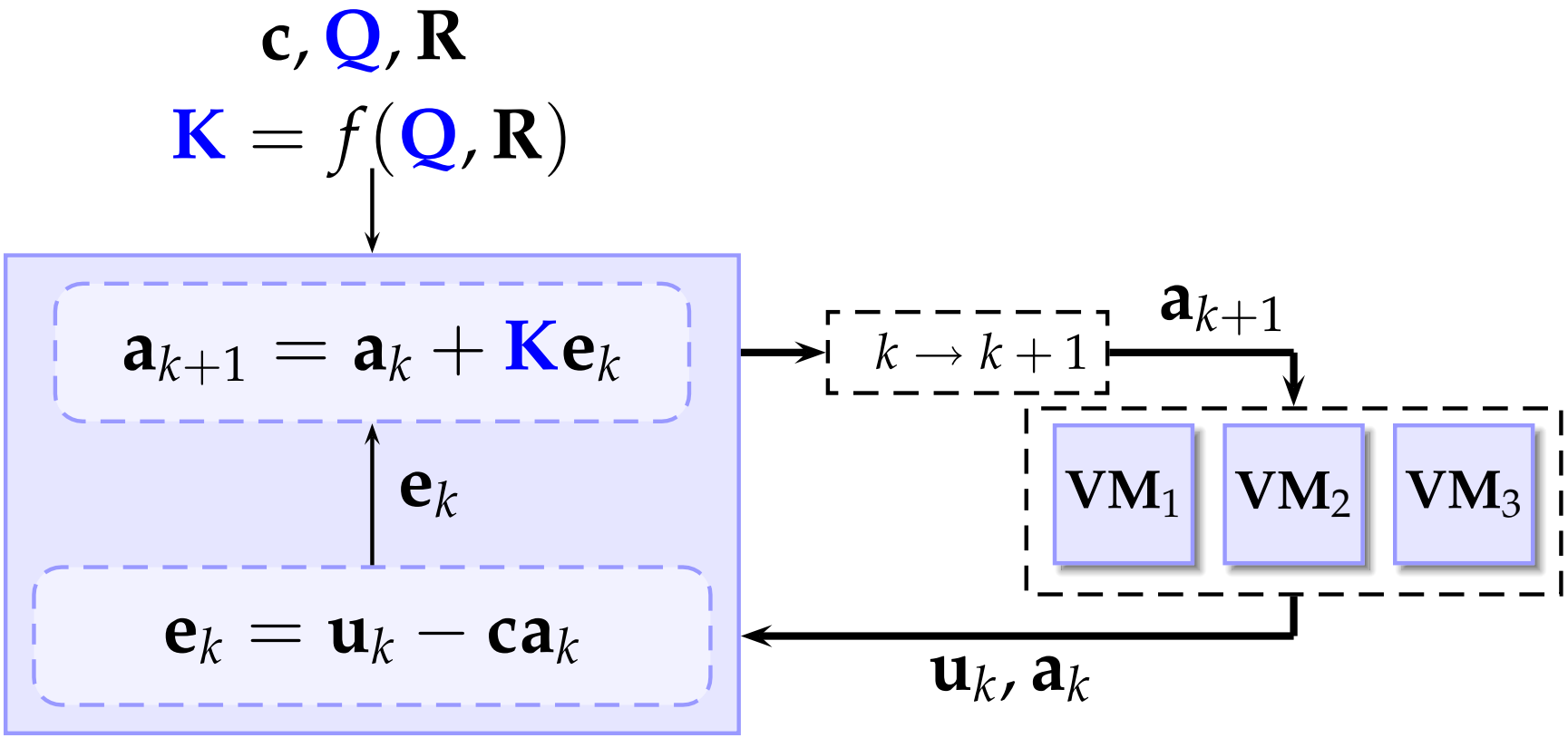
Resource Coupling

What is?

In **multi-tier** applications, the utilisations from the different components are related since requests are being processed by all the components.

- one controller for all the components, MIMO model
- for **faster** adaptations during workload increases
- update the allocations based on the saturation of **all** components
- resource coupling utilisation model

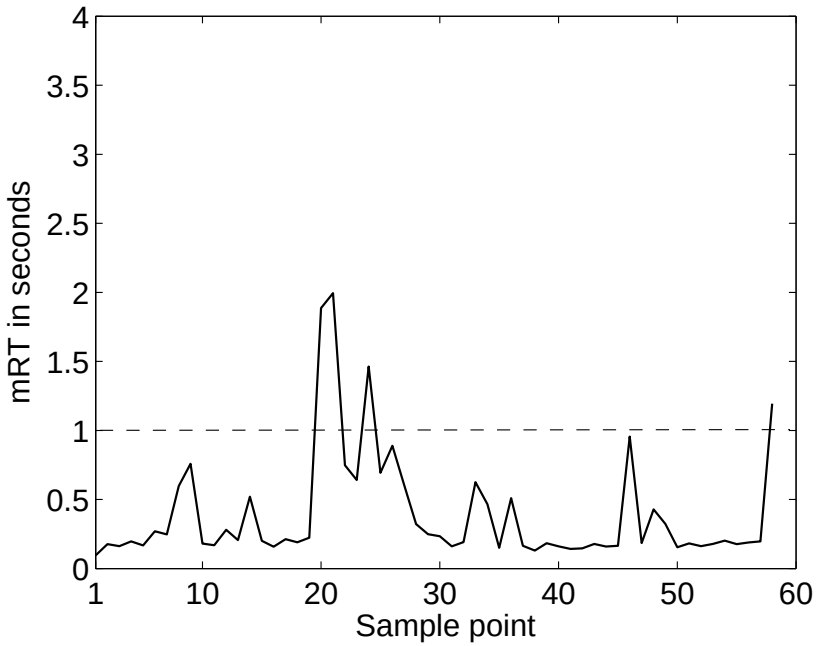
Process Noise Covariance Controller - PNCC



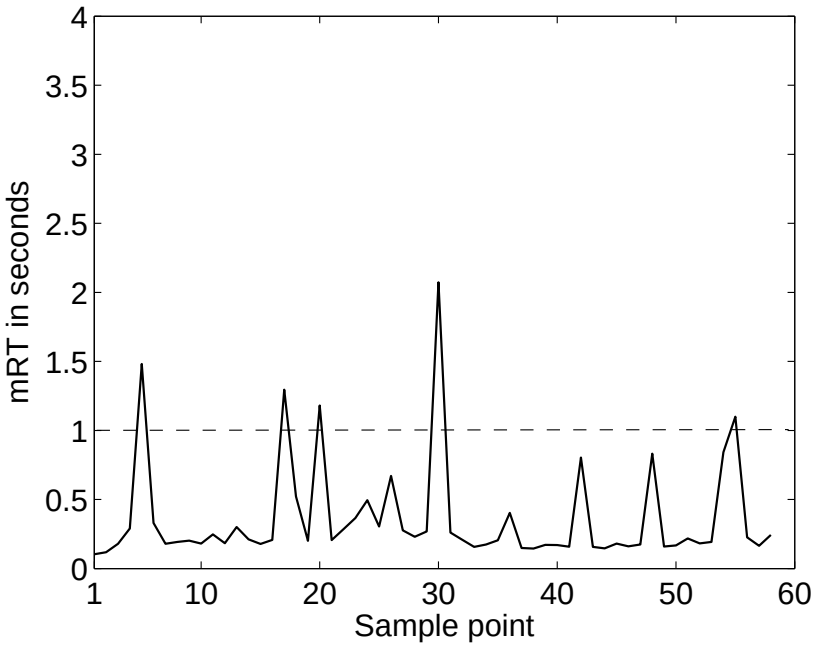
- $\mathbf{Q}$ is the covariance matrix:$\mathbf{Q}(i,i) = var(i)$,$\mathbf{Q}(i,j) = cov(i,j)$
- $a_{k+1} = a_k + K(1,1)e_k^1 + K(1,2)e_k^2 + K(1,3)e_k^3$

PNCC Results 1/2

KBC Tomcat CPU



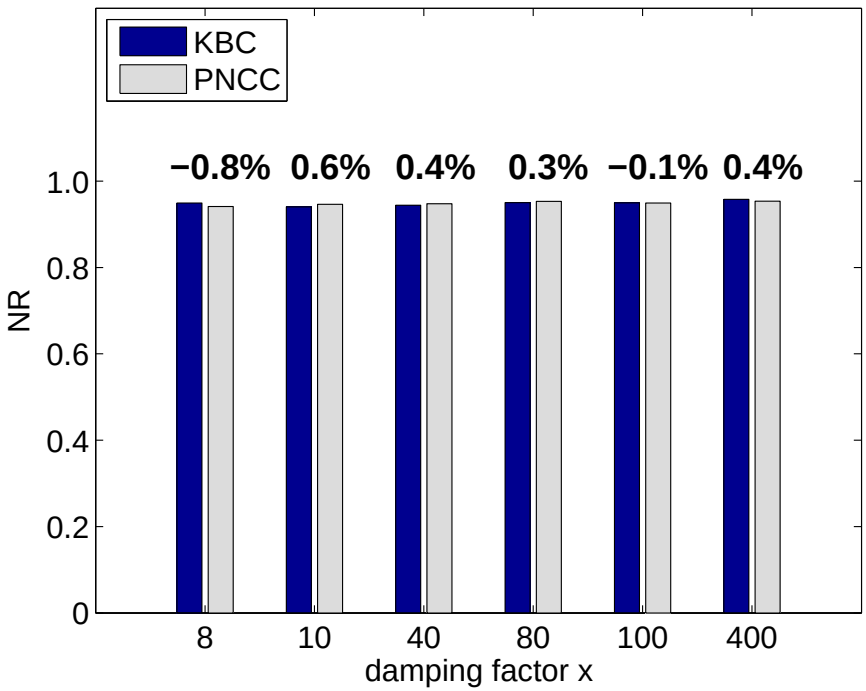
PNCC Tomcat CPU



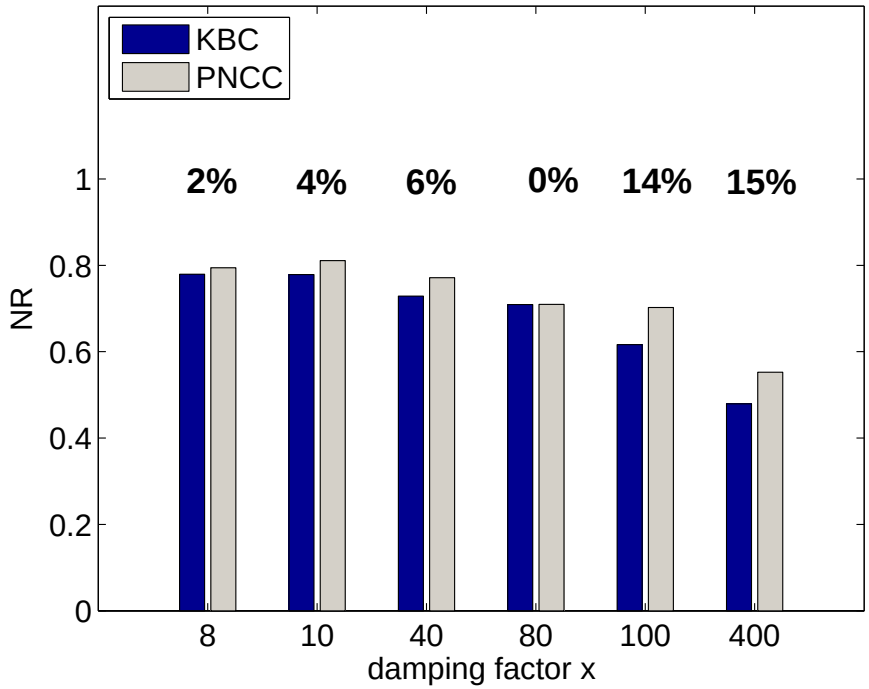
PNCC has better responses in workload increases

PNCC Results 2/2

NR, stable workload



NR, workload increase

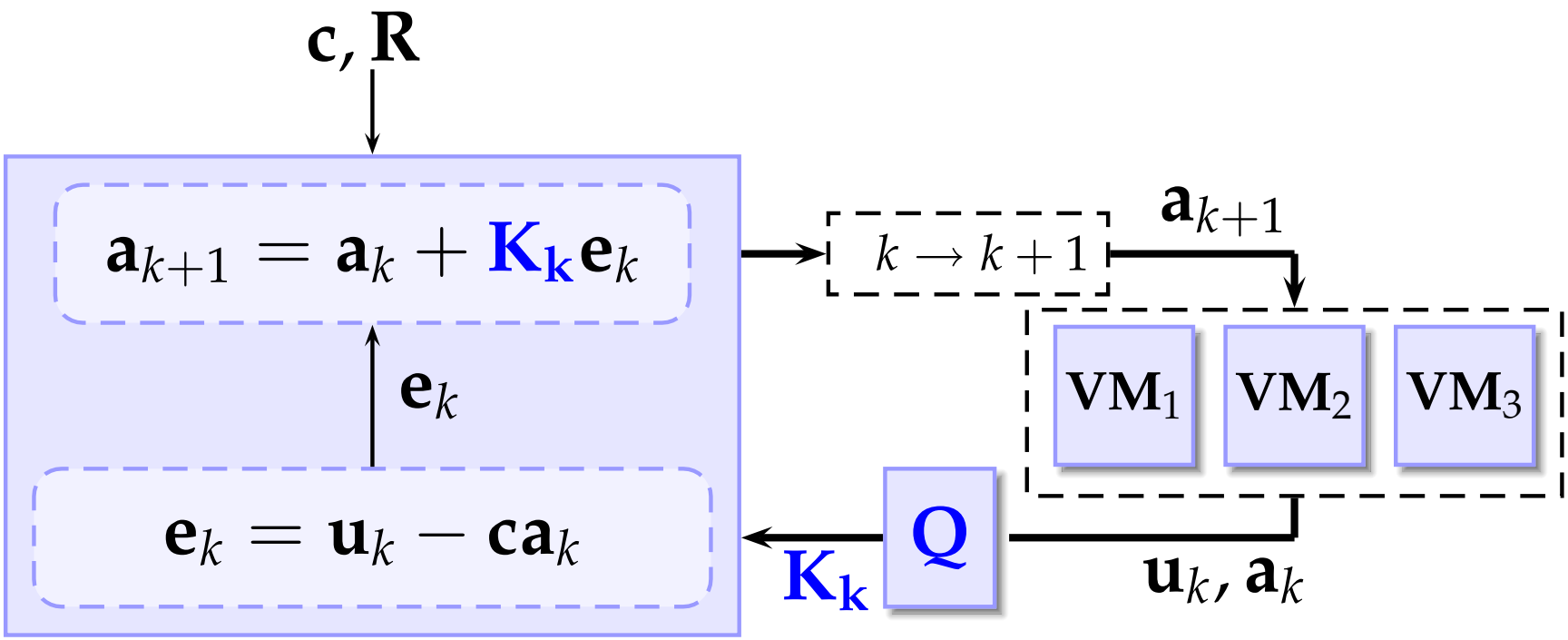


- NR: % of completed requests with mean response time $\leq$ 1sec
- $\uparrow$ damping factor $\rightarrow$ smoother filter
- PNCC allocates resources faster at the workload increase

Controllers

controller	filtering	setting of parameters	resource coupling	parameter adaptation
KBC	✓	server information	**x**	**x**
PNCC	✓	server information	✓	**x**

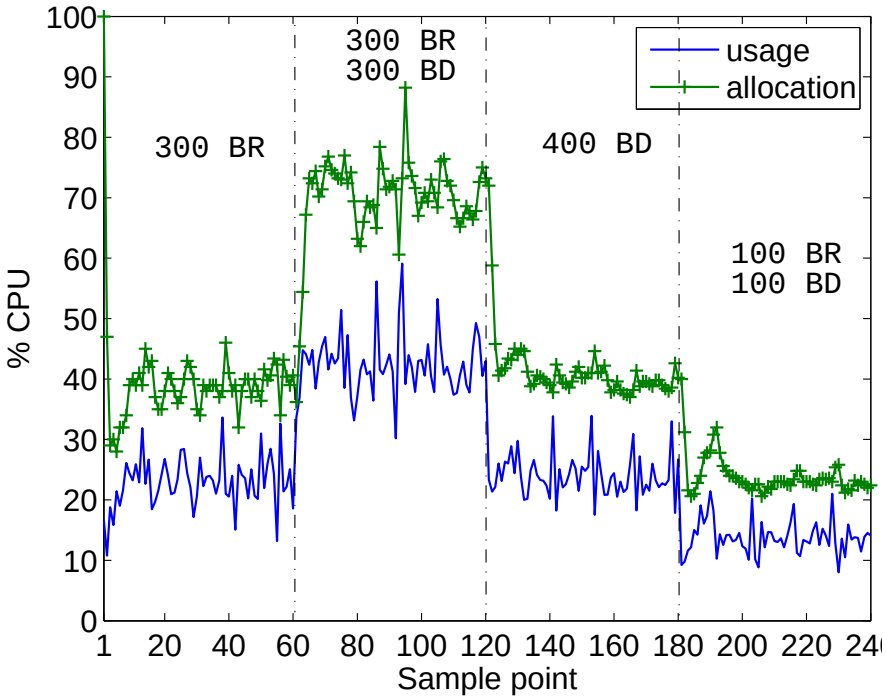
Adaptive PNCC - APNCC



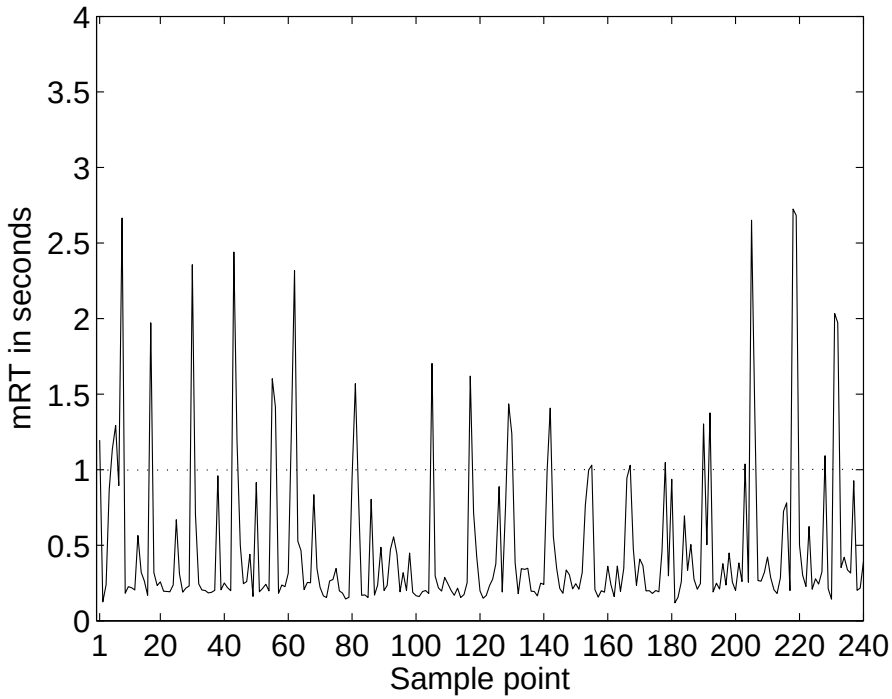
Kalman gain K_k is updated

APNCC Results 1/3

Tomcat CPU



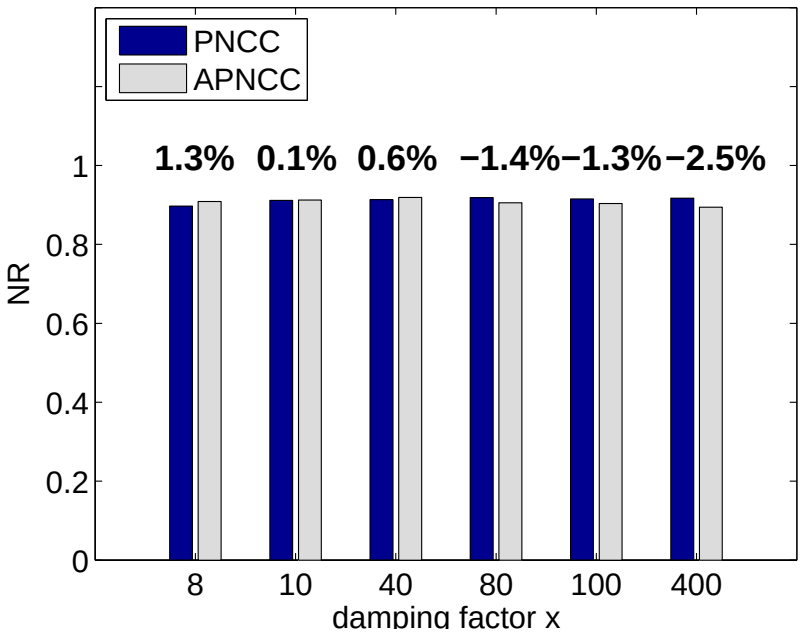
Response



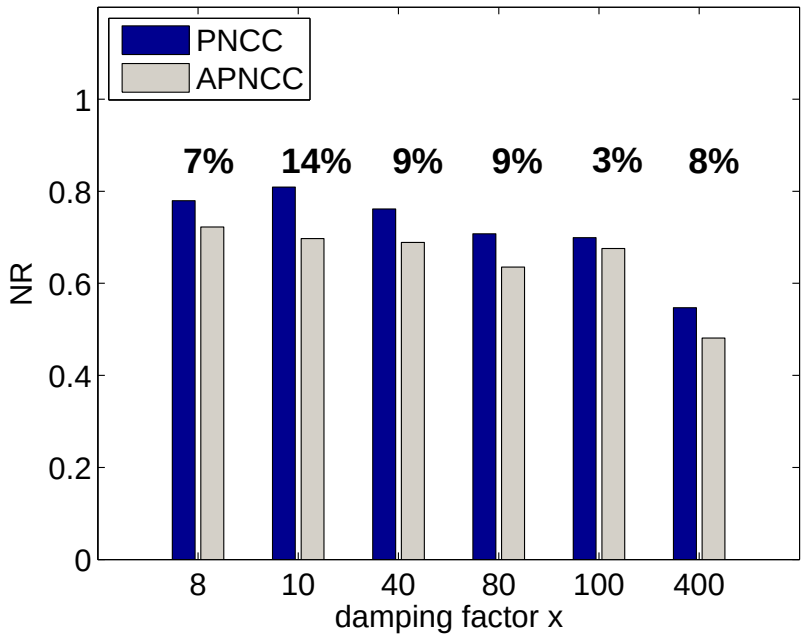
- mixed, varying, unknown workloads
- APNCC adapts to utilisations across workloads
- 89.58% of requests have mean response time $\leq$ 1sec

APNCC Results 2/3

NR, stable worklaod



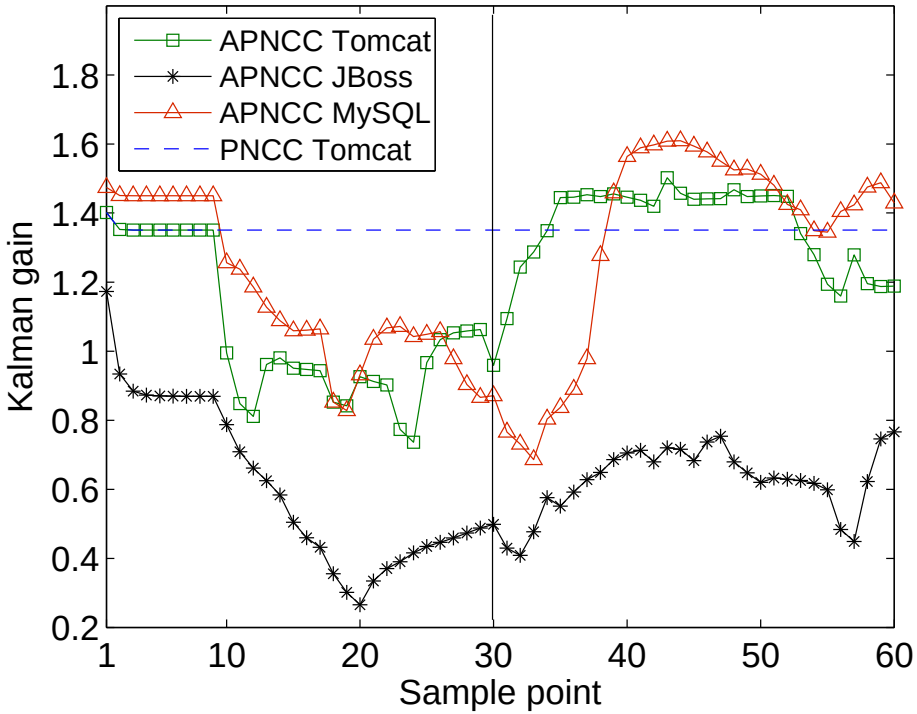
NR, workload increase



- NR: % of completed requests with mean response time $\leq$ 1sec
- $\uparrow$ damping factor $\rightarrow$ smoother filter
- APNCC's performance slightly decreases from PNCC

APNCC Results 3/3

APNCC Gains



- focus on workload increase
- APNCC has a short delay to the workload increase
- APNCC adapts its gain to any workload

Controllers

controller	filtering	setting of parameters	resource coupling	parameter adaptation
KBC	✓	server information	**x**	**x**
PNCC	✓	server information	✓	**x**
APNCC	✓	server information	✓	✓

Conclusions

A self-managing framework with Kalman-based feedback controllers to update the CPU allocations of virtualized applications.

Characteristics:

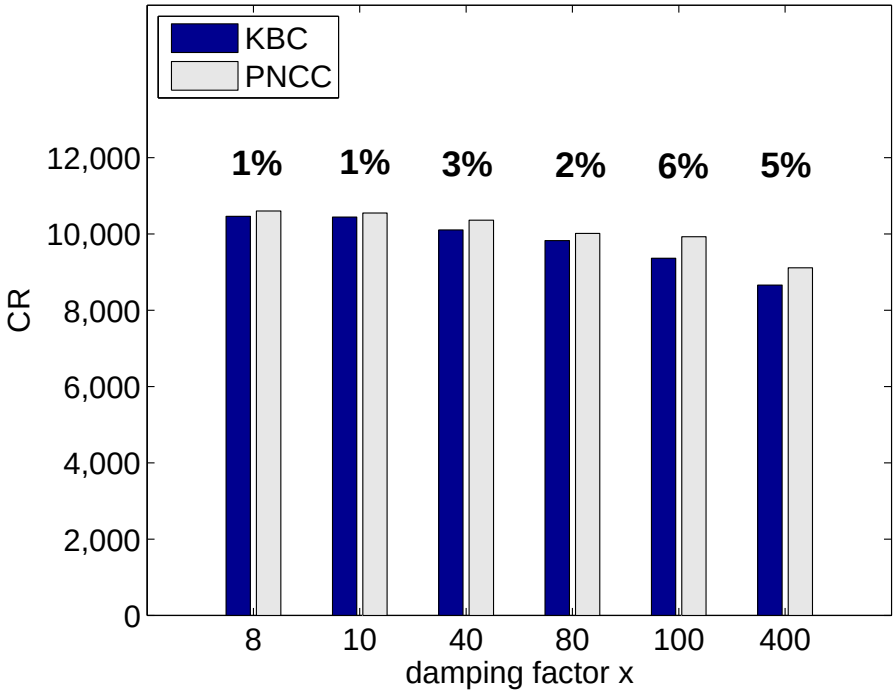
- use a simple and general model application
- operate on different filtering modes
- self-configured input parameters

Thank You!

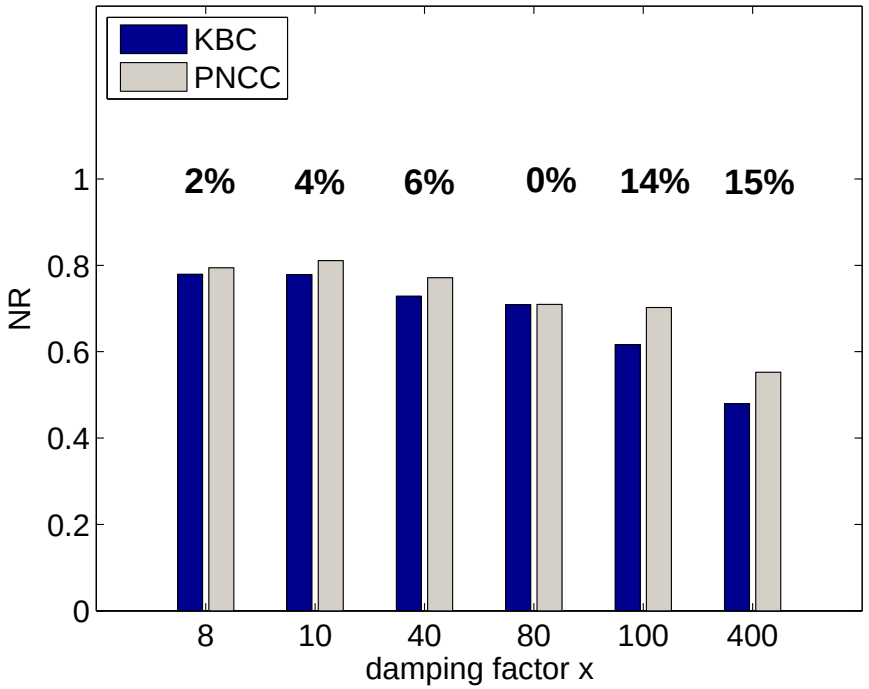
Questions?

Backup Slides — PNCC Results

CR



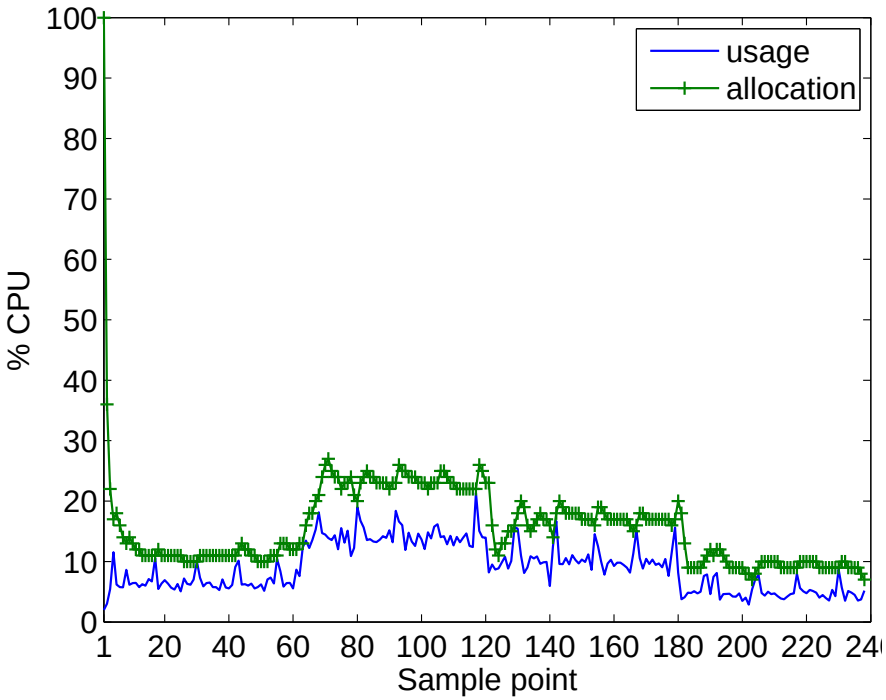
NR



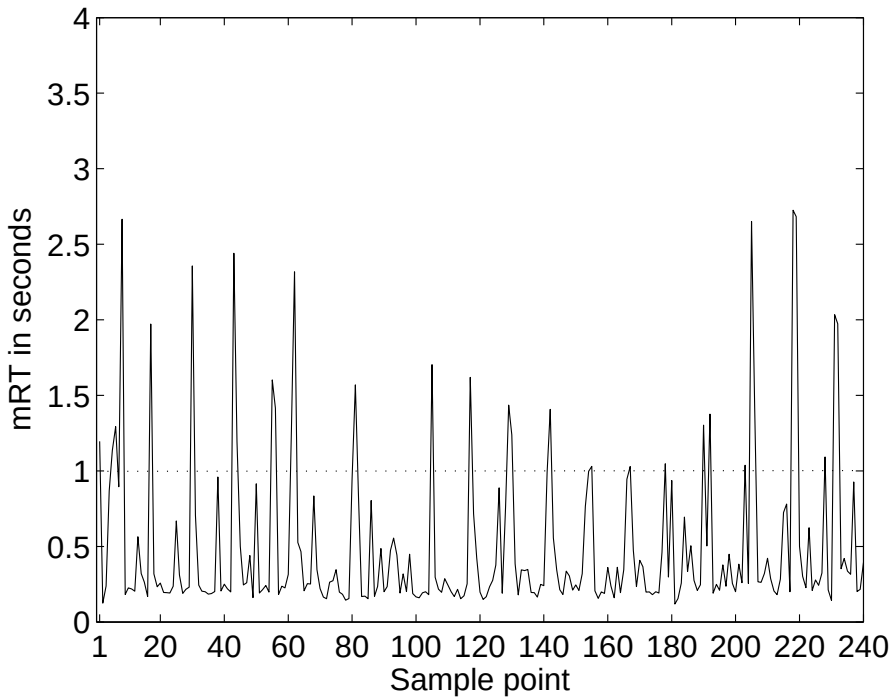
- CR: completed requests
- NR: % of CR with mean response time $\leq$ 1sec
- $\uparrow$ damping factor $\rightarrow$ smoother filter
- PNCC allocates resources faster and has better performance

Backup Slides — APNCC Results

JBoss CPU



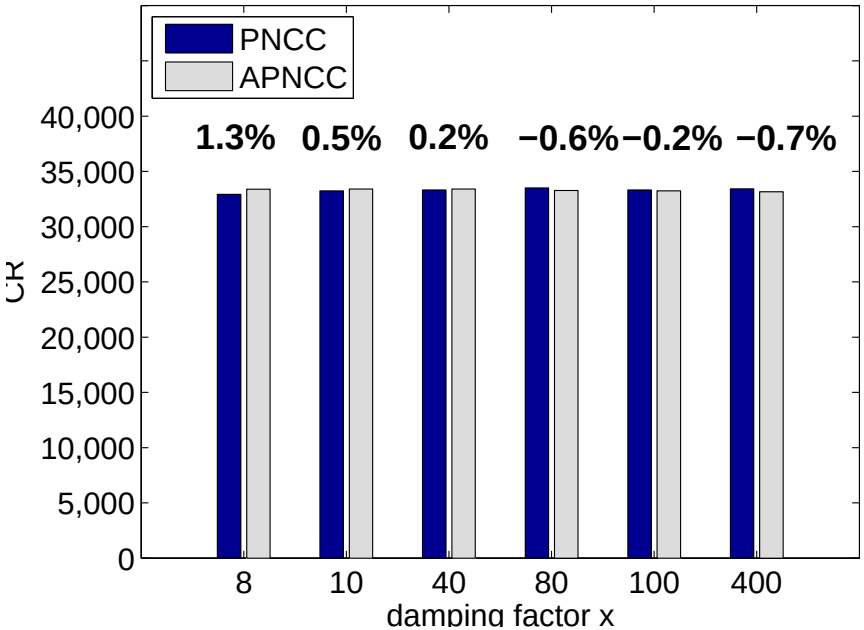
Response



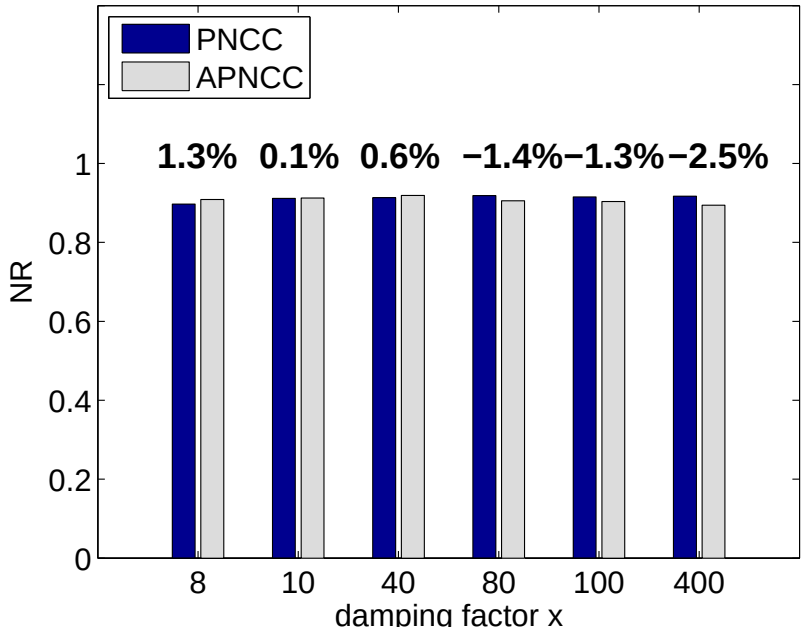
- mixed, varying, unknown workloads
- APNCC adapts to utilisations across workloads
- 89.58% of requests have mean response time $\leq$ 1sec

Backup Slides — APNCC Results

CR



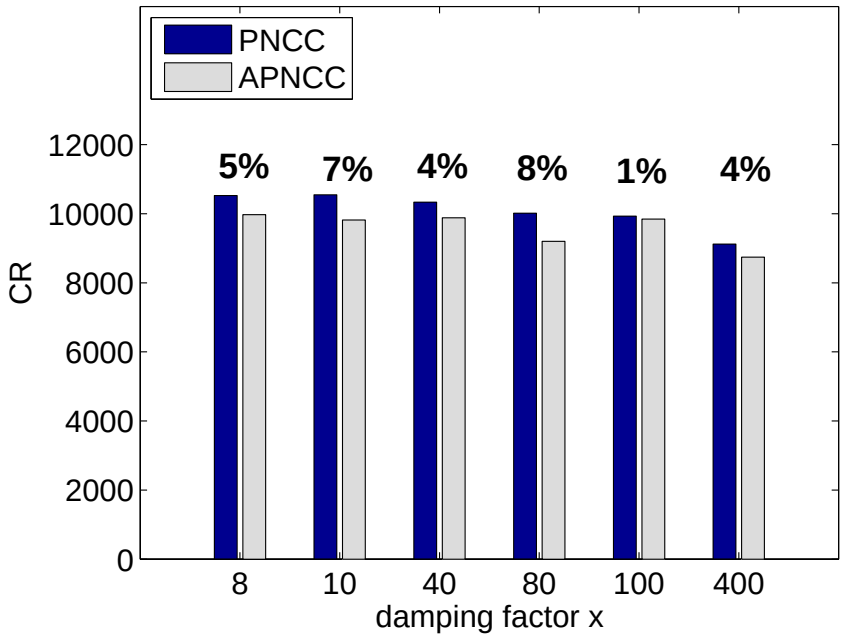
NR



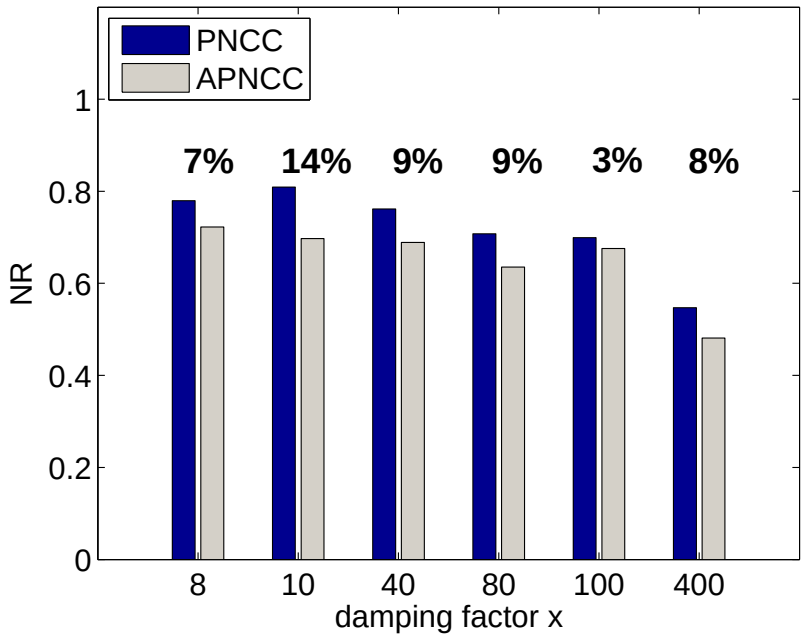
- CR: completed requests
- NR: % of CR with mean response time $\leq$ 1sec
- stable workload
- KBC and PNCC perform equally wells

Backup Slides — APNCC Results

CR



NR



- focus on workload increases
- APNCC's performance slightly decreases from PNCC