POLITECNICO DI MILANO





Black-box Performance Models for Virtualized Web Service Applications

Danilo Ardagna, Mara Tanelli, Marco Lovera, Li Zhang
ardagna@elet.polimi.it

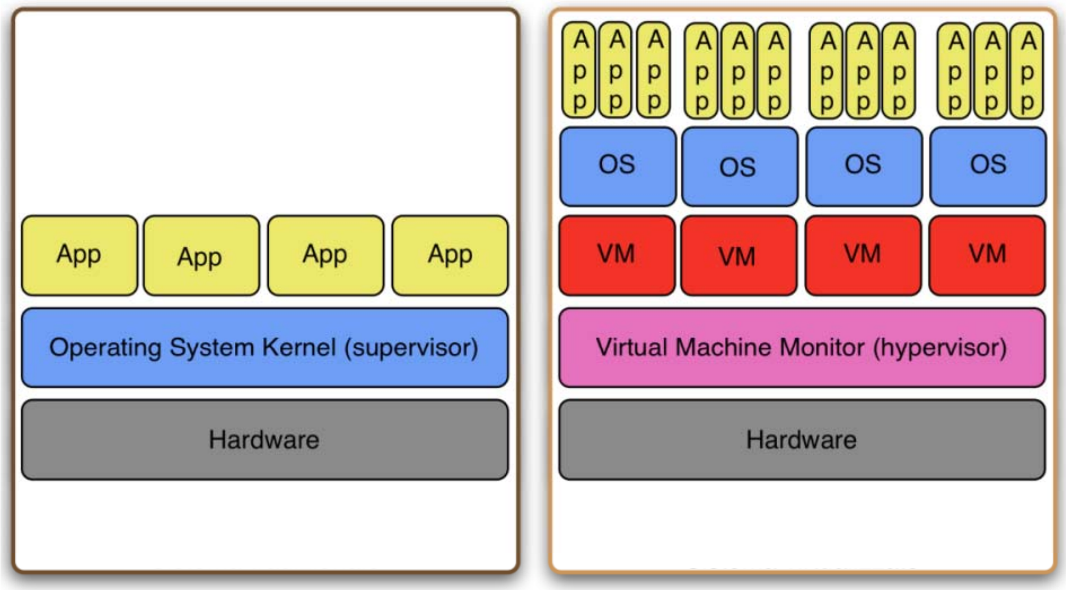
Reference scenario

- Virtualization, proposed in early '70s, is driving again the interest both of industry and academia
- Advantages:
  - Physical resources are partitioned among competing running VMs, improved security and reliability, performance isolation
  - Resource allocation parameters can be updated by in few milliseconds without introducing any system overhead
- Problems:
  - Performance modelling of virtualized environments is challenging
  - Traditional queueing network models are inadequate to model virtualized systems performance at a very fine-grained time scale

Reference scenario

Hardware resources (CPU, RAM, ecc...) are partitioned and shared among multiple **virtual machines** (VMs)

The virtual machine monitor (VMM) governs the access to the physical resources among running VMs



Reference scenario

- Virtualization, proposed in early ’70s, is driving again the interest both of industry and academia
- Advantages:
  - Physical resources are partitioned among competing running VMs, improved security and reliability, performance isolation
  - Resource allocation parameters can be updated by in few milliseconds without introducing any system overhead
- Problems:
  - Performance modelling of virtualized environments is challenging
  - Traditional queueing network models are inadequate to model virtualized systems performance at a very fine-grained time scale

Black-box models for run-time modelling of virtualized systems

- Use experimental data to construct dynamical models for performance control of virtualized Web systems
  - Short time frame (minutes, seconds)
  - System identification used to develop models for:
    - Capturing system transients
    - Taking into account workload variability

- In the literature: only SISO controllers for DVFS and admission control available

- Present work: preliminary analysis of time-varying MIMO models for VMs resource provisioning

Problem statement

- Performance modelling of CPU bounded Web service applications running on a single core
- Each VM hosts a single application
- VMM configured to support work-conserving mode

Notation

- Δt: sampling time interval
- k: discrete time index
- R^i_k: average application *i* response time in the *k*-th time interval
- ϕ^i_k: fraction of capacity devoted for executing the VM which hosts application *i* in the *k*-th time interval
- n: number of running VMs

Generalized Processor Sharing (GPS) Scheduling:

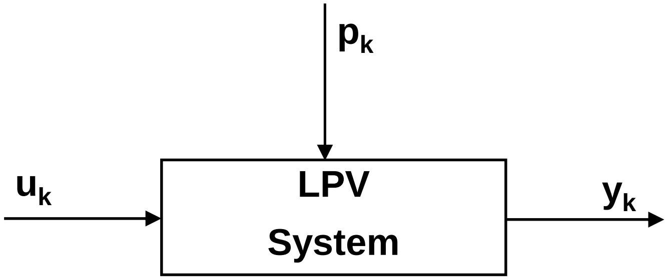
$$\frac{\phi_k^{i}}{\sum_{i' \in K(t)} \phi_k^{i'}}$$

- Identification problem:
  - Derive a mathematical representation for the behaviour of a physical system on the basis of input-output data
  - Select a class of models and a suitable algorithm for the estimation of the model parameters

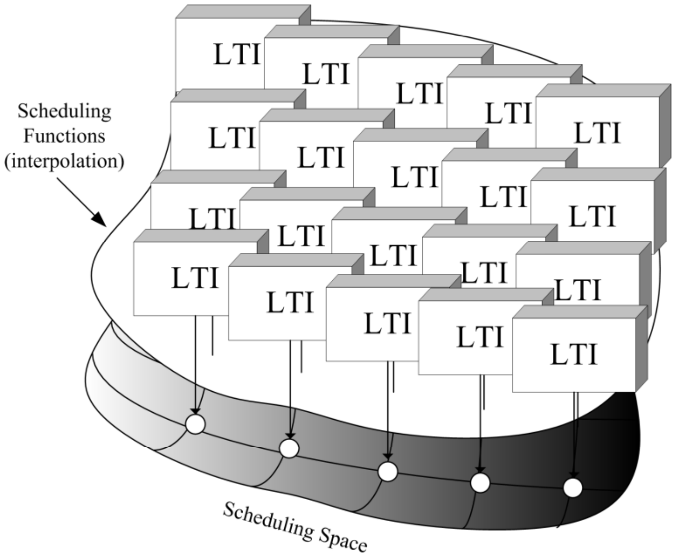
LPV State Space Models and Identification Algorithm

- Linear Parameter Varying (LPV) systems are a class of time-varying systems
- In discrete-time state space form:

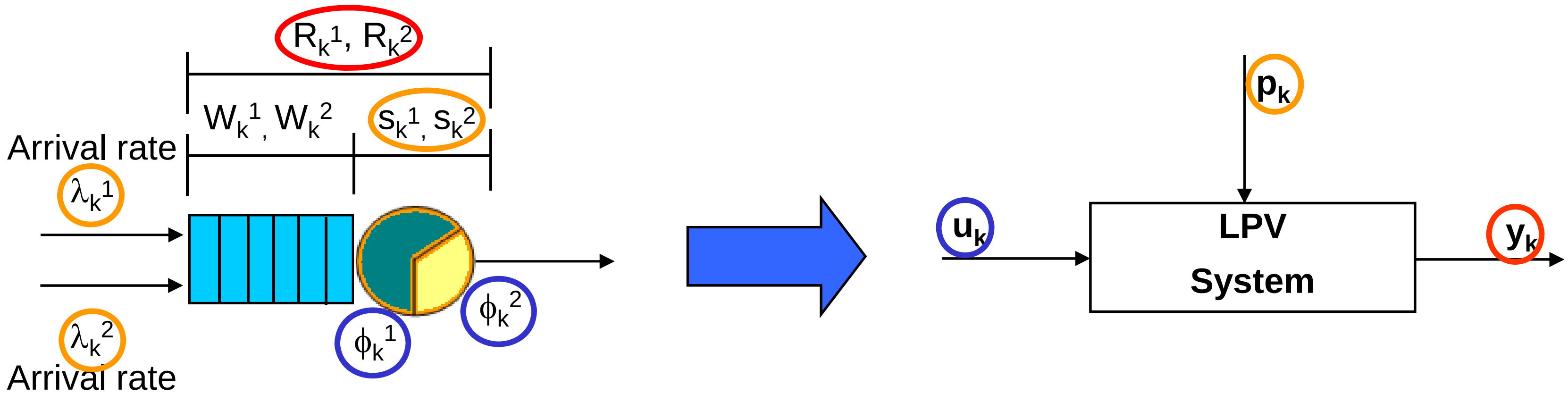
$$x_{k+1} = A(p_k)x_k + B(p_k)u_k$$
$$y_k = C(p_k)x_k + D(p_k)u_k$$



- "Time varying systems, the dynamics of which are functions of a measurable, time varying parameter vector p."



LPV state-space models



- Affine parameter dependence (LPV-A):

$$A(p_k) = A_0 + A_1 p_{1,k} + \ldots + A_s p_{s,k}$$

where $p_{i,k}$, i=1,…,s denotes the *i*-th component of vector p_k

- Input-affine parameter dependence (LPV-IA):
  - B and D matrices are parametrically-varying
  - $A=A_0$, $C=C_0$

State-space LPV identification algorithms

- LPV-IA models: obtained using LTI subspace methods – we consider the MOESP class (Verhaegen and Dewilde 1992, Verhaegen 1994)
- System parameterised by θ, identification perfomed minimising

$$V_N(\theta) := \sum_{k=1}^{N} ||y_k - \hat{y}_k(\theta)||_2^2 = E_N^T(\theta) E_N(\theta)$$

- Levenberg-Marquardt gradient search:

$$\theta^{(i+1)} = \theta^{(i)} - \alpha^{(i)} \left(\beta^{(i)} I + \Psi_N^T(\theta^{(i)}) \Psi_N(\theta^{(i)})\right)^{-1} \Psi_N^T(\theta^{(i)}) E_N(\theta^{(i)})$$

where

$$\Psi_N(\theta) := \frac{\partial E_N(\theta)}{\partial \theta^T}$$

- Choice of θ: full parameterisation

$$\theta = \mathrm{vec}(\Theta), \qquad \Theta = \begin{bmatrix} A_0 & A_1 & \cdots & A_s & B_0 & B_1 & \cdots & B_s \\ C_0 & C_1 & \cdots & C_s & D_0 & D_1 & \cdots & D_s \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

- Issues associated with non uniqueness:

$$\overline{\Theta} = \begin{bmatrix} \overline{A} & \overline{B} \\ \overline{C} & \overline{D} \end{bmatrix} = \begin{bmatrix} \mathcal{T}^{-1} & 0 \\ 0 & I_\ell \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \mathcal{T}_{s+1} & 0 \\ 0 & I_{m(s+1)} \end{bmatrix}$$

is such that $V_N(\theta) = V_N(\overline{\theta}), \quad \overline{\theta} = \mathrm{vec}(\overline{\Theta}).$

- This must be taken into account in the update rule, see also (Verdult, Lovera, Verhaegen 2004) and (Lee, Poolla 1999)

- Restriction of the update to the directions that change the cost function:

$$\theta^{(i+1)} = \theta^{(i)} - \alpha^{(i)} U_2(\theta^{(i)}) \left(\beta^{(i)} I + U_2^T(\theta^{(i)}) \Psi_N^T(\theta^{(i)}) \Psi_N(\theta^{(i)}) U_2(\theta^{(i)})\right)^{-1} \times U_2^T(\theta^{(i)}) \Psi_N^T(\theta^{(i)}) E_N(\theta^{(i)})$$

- where $M_\Theta = \begin{bmatrix} U_1(\theta) & U_2(\theta) \end{bmatrix} \begin{bmatrix} \Sigma(\theta) \\ 0 \end{bmatrix} V^T(\theta),$

$$M_\Theta := \sum_{i=1}^{s+1} \begin{bmatrix} \Pi_i^T \\ 0_{m(s+1)\times n} \end{bmatrix} \otimes \begin{bmatrix} A\Pi_i^T \\ C\Pi_i^T \end{bmatrix} - \begin{bmatrix} A^T \\ B^T \end{bmatrix} \otimes \begin{bmatrix} I_n \\ 0_{\ell \times n} \end{bmatrix},$$

$$\Pi_i := \begin{bmatrix} 0_{n\times(i-1)n} & I_n & 0_{n\times(s+1-i)n} \end{bmatrix}.$$

Experimental setting

- Two reference scenarios:
  - A Micro benchmarking instrumented Web application where the CPU service time was generated according to a lognormal distribution
  - SPECweb2005 industrial benchmark suite and the data required for the parametrization (mainly the VMs' utilization) has been gathered from the hypervisor without any code instrumentation

- VMM monitor: Xen 3.0 and Xen 3.3

- Validation: Synthetic workload inspired by a real-world. Log trace from a large financial system. Experiments last 24 hours

- Variance accounted for (VAF)

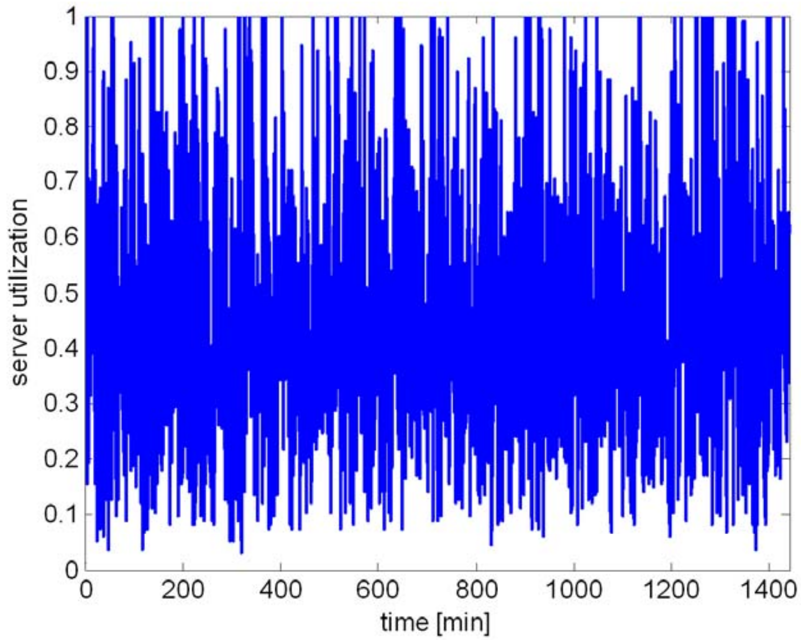
$$VAF = 100\left(1 - \frac{Var[y_k - y_{sim,k}]}{Var[y(k)]}\right)$$

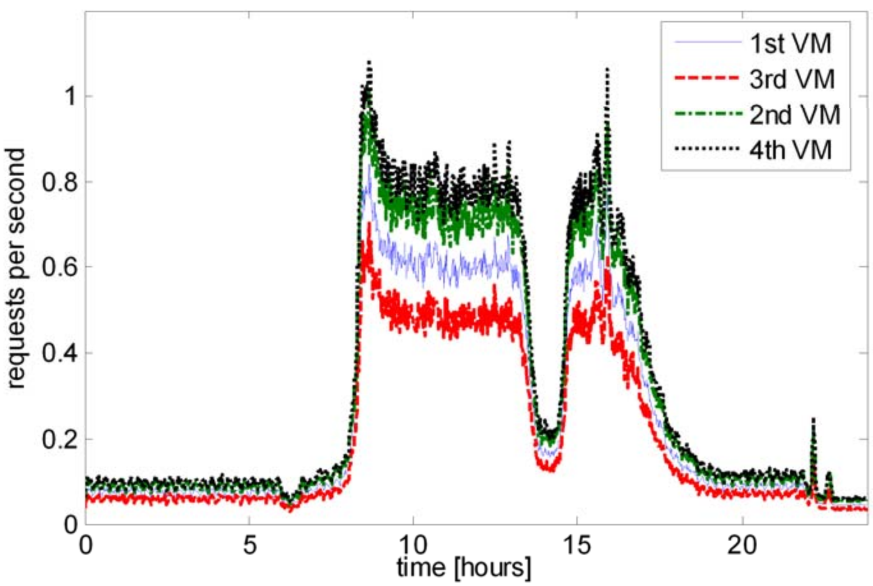
- Average simulation error (e_{avg})

$$e_{avg} = 100\left(\frac{E\left[|y_k - y_{sim,k}|\right]}{E\left[|y_k|\right]}\right)$$

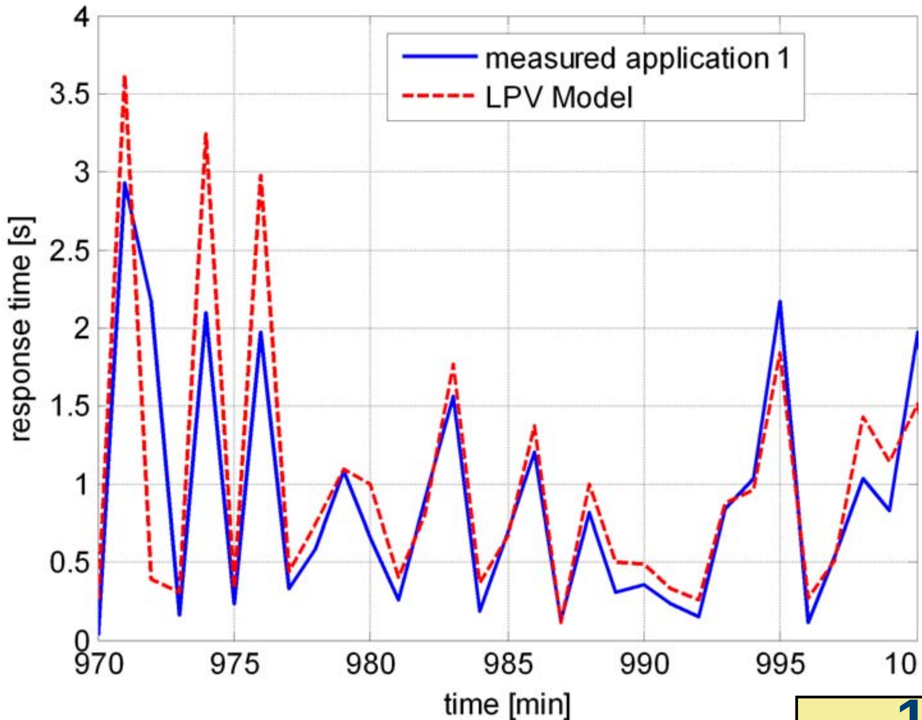
Micro-benchmarking Web Service Application Experiments

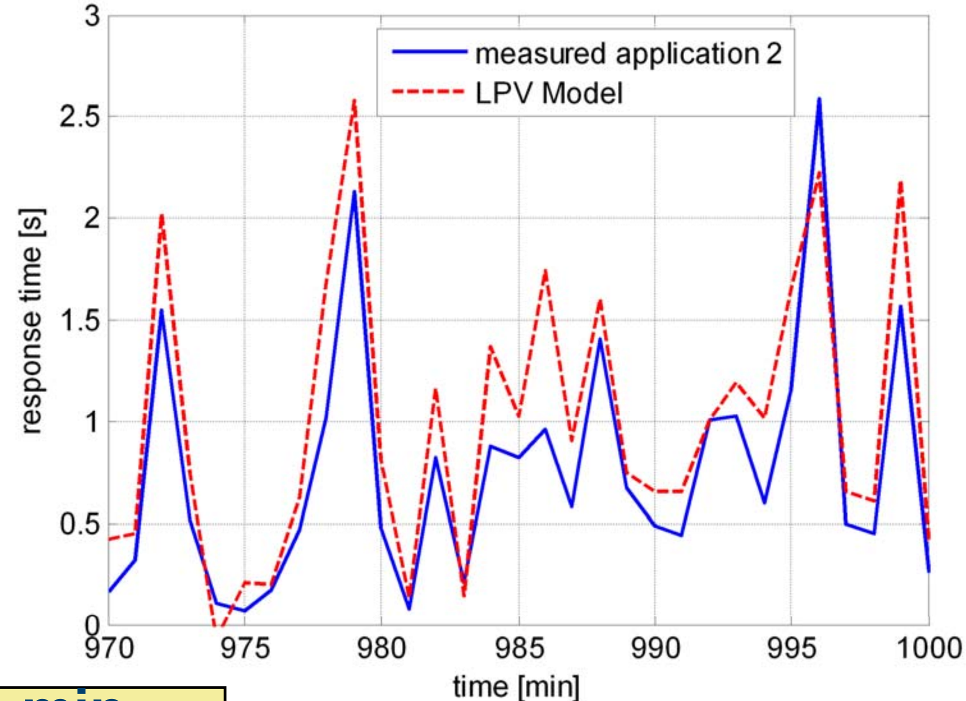
- Number of VMs varied between 2 and 4
- For system identification purposes λ_k^i accessing each VM vary stepwise every 1 minute, between 0.15 req/s and 1.5 ,req/s, according to a Poisson process
- Each request consumes s^i_k CPU time varied between 0.06 s and 1.1 s.
- 1,440 intervals (24 hours)
- ϕ^i_k has been selected as a realization of a uniform random variable with values between 0.1 and 0.9
- Parametrization $[s_k^i \ \rho_k^i]$





Micro-benchmarking Web Service Application Experiments

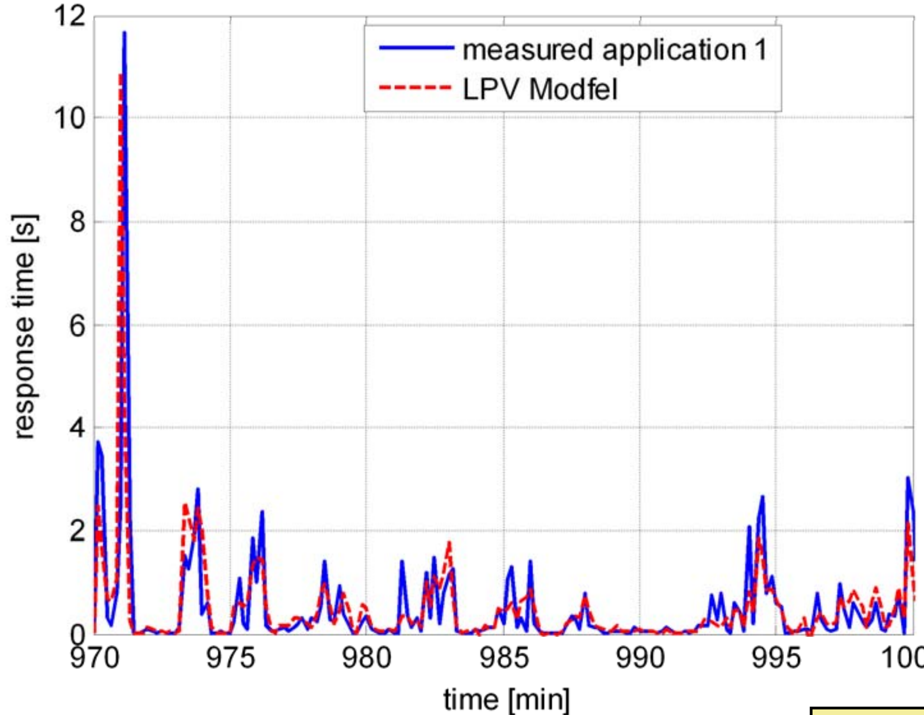


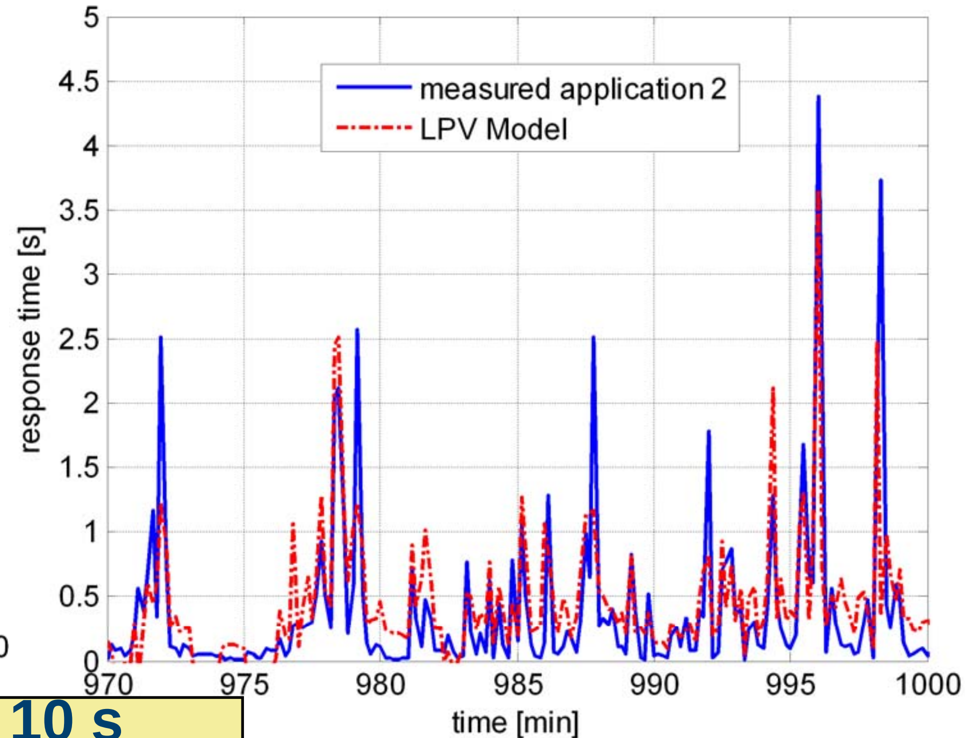


1 min

		VAF on 24h	VAF light-load	VAF heavy-load	e_{avg} on 24h	e_{avg} light-load	e_{avg} heavy-load
2 VMs	VM1	82.21%	82.8%	82.48%	5.46%	1.43%	13.4%
	VM2	73.41%	72.49%	82.22%	2.63%	6.42%	13.18%
3 VMs	VM1	81.59%	78.33%	88.26%	4.61%	4.13%	5.38%
	VM2	81.1%	79.45%	86.17%	1.16%	1.59%	6.08%
	VM3	72.97%	73.48%	61.62%	11.07%	12.84%	7.63%
4 VMs	VM1	78.66%	72.45%	90.09%	8.12%	8.73%	7.18%
	VM2	73.12%	70.69%	86%	11.36%	14.98%	4.79%
	VM3	73.57%	77.67%	67.88%	7.74%	7.53%	8.02%
	VM4	56.27%	52.52%	85.67%	15.06%	19.04%	7.15%

Micro-benchmarking Web Service Application Experiments

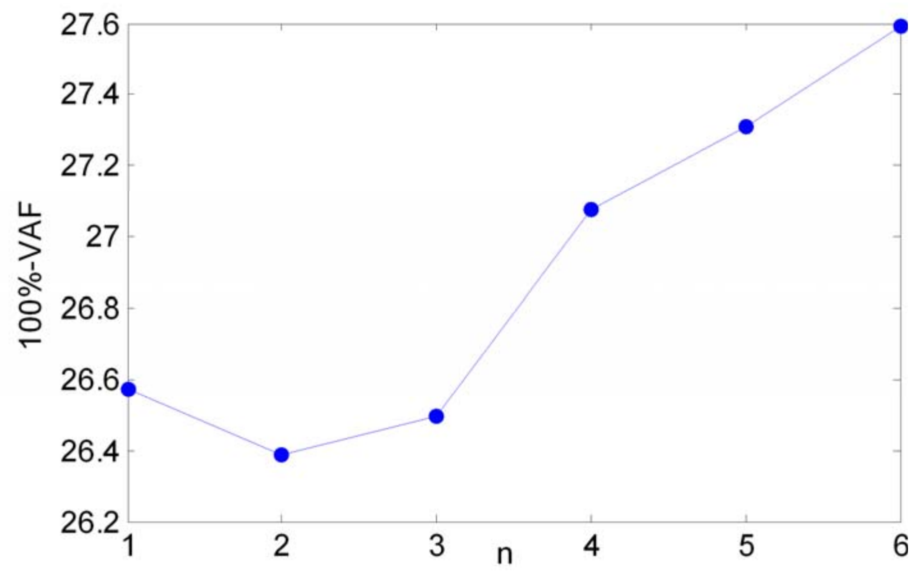


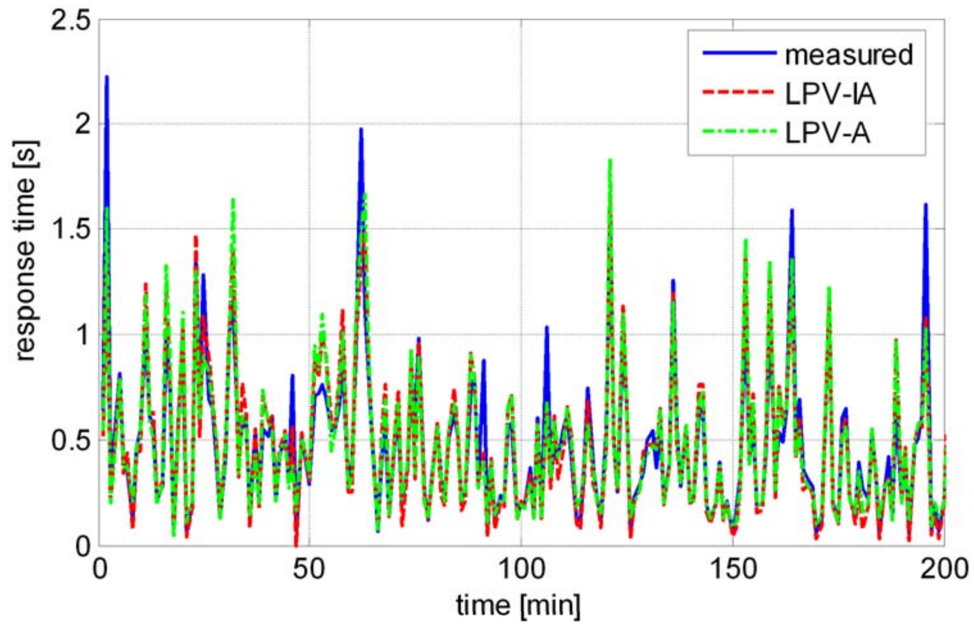


10 s

		VAF on 24h	VAF light-load	VAF heavy-load	e_{avg} on 24h	e_{avg} light-load	e_{avg} heavy-load
2 VMs	VM1	74.85%	74.8%	86.15%	6.5%	8.3%	4.07%
	VM2	70.56%	67.9%	83.94%	3.4%	2.5%	4.17%
3 VMs	VM1	78.5%	75.19%	83.05%	6.9%	1.5%	12.76%
	VM2	78.46%	75.51%	88.78%	6.5%	3.27%	19.58%
	VM3	70.75%	69.14%	83.3%	2.76%	16.55%	18.21%
4 VMs	VM1	75.97%	66.37%	88.97%	6.91%	10.35%	3.16%
	VM2	72.72%	68.1%	87.23%	9.14%	12.23%	5.24%
	VM3	74.85%	69.22%	82.94%	5.84%	7.29%	4.38%
	VM4	66.80%	62.38%	89.06%	11.99%	16.92%	4.77%

Model order evaluation and LPV-A vs. LPV-IA





- Identification Execution Time

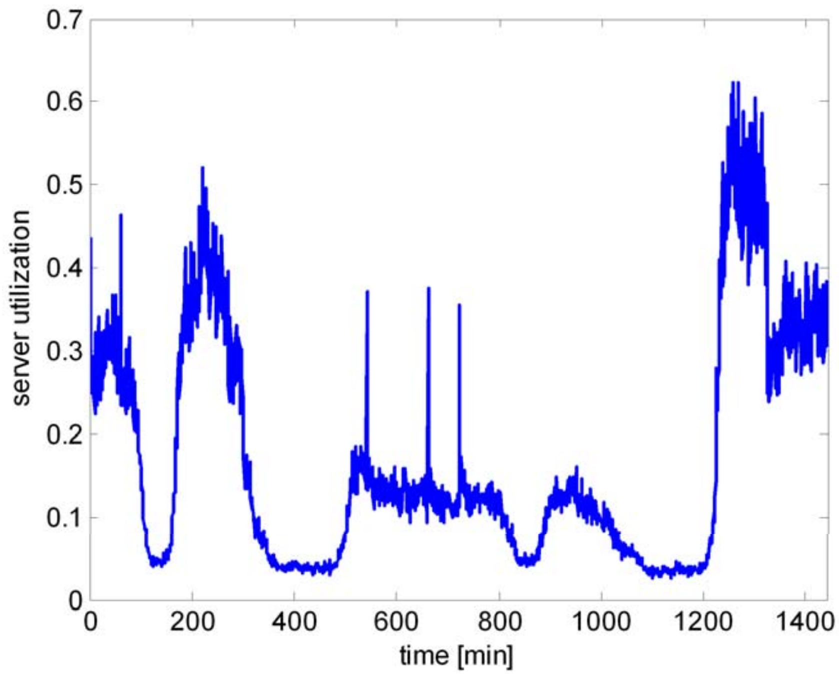
	$\Delta t = 1$ min	$\Delta t = 10$ s
2 VM	0.129 s	1.109 s
3 VM	0.690 s	4.696 s
4 VM	1.899 s	13.691 s

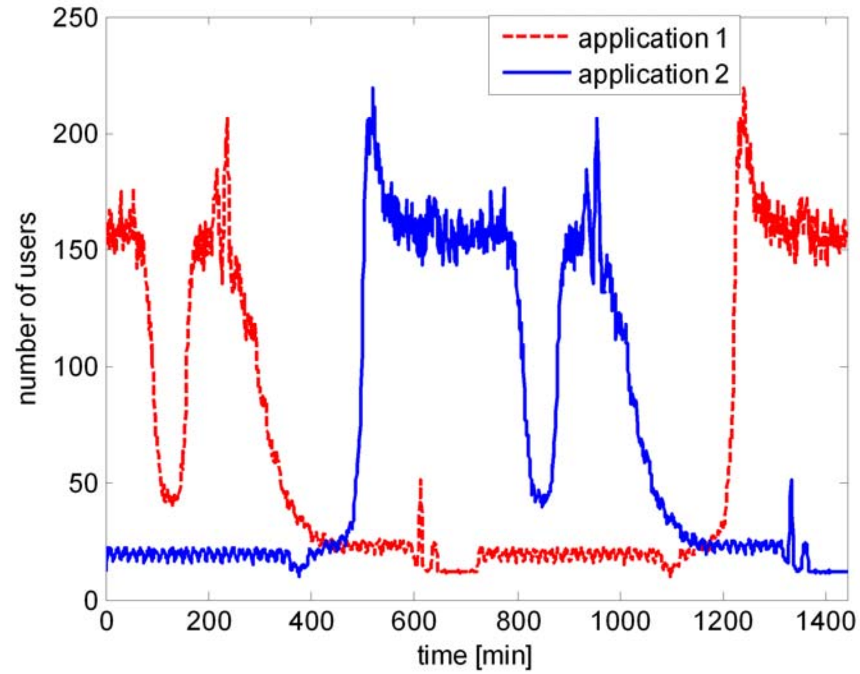
SPECweb2005 Experiment

- Two VMs running the banking and e-commerce loads
- The number of users N^i_k accessing each of the two VMs varied stepwise every 1 minute, with values between 10 and 220
- Proportional assignment scheme:

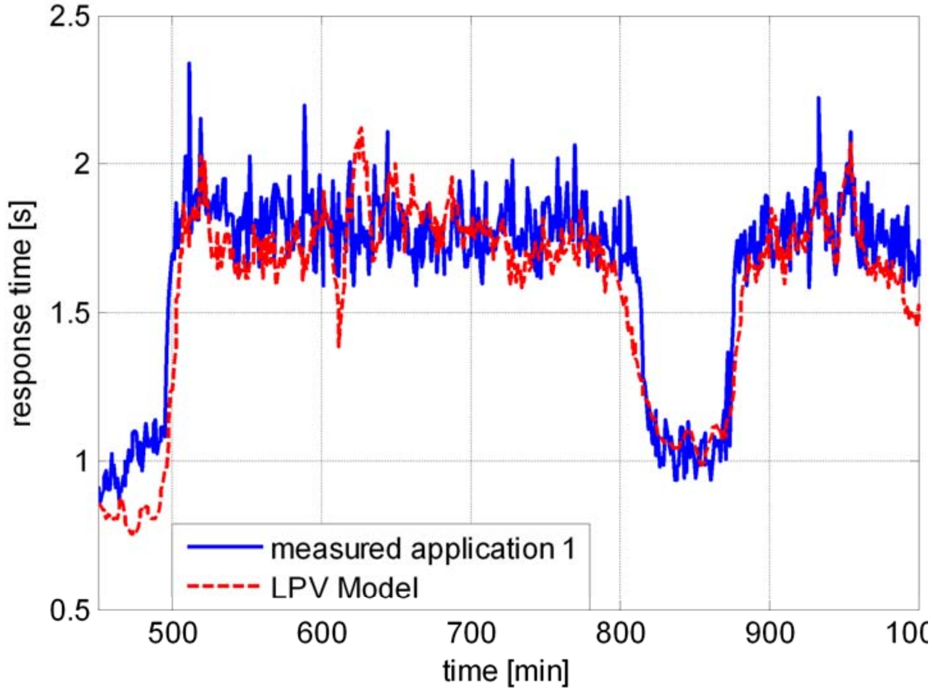
$$\phi_k^i = \max\left(0.1, \frac{N_k^i}{N_k^1 + N_k^2}\right)$$

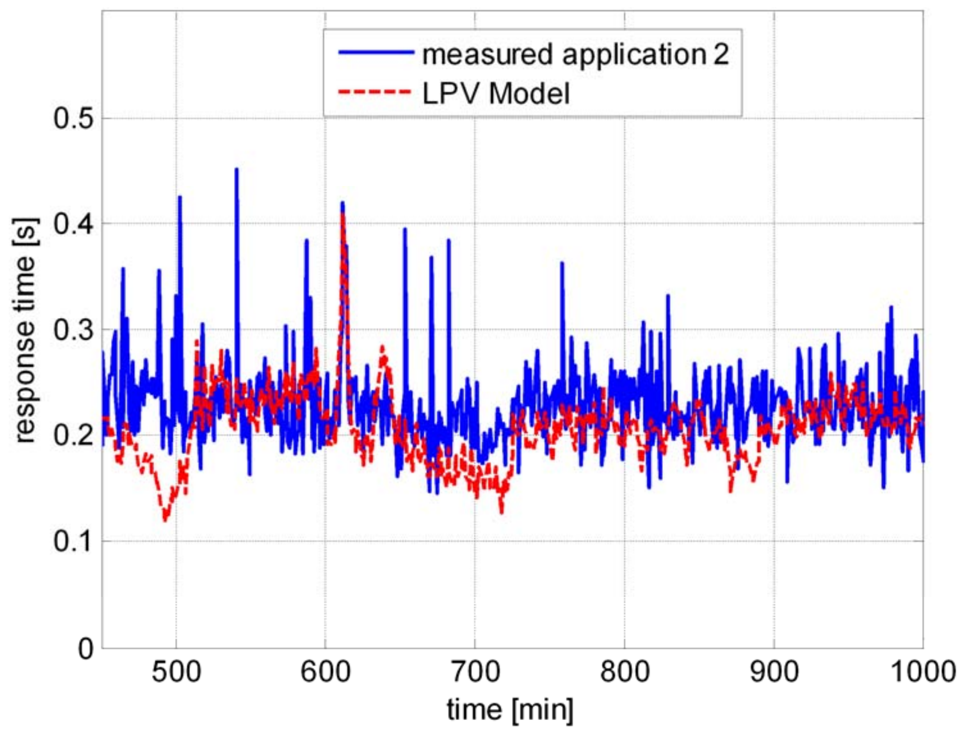
- 1,440 intervals (24 hours)
- Parametrization $[N_k^i \ \rho_k]$





SPECweb2005 Experiment





		VAF on 24h	e_{avg} on 24h
Identification data	VM1	59.85%	6.85%
	VM2	87.20%	6.97%
Validation data	VM1	64.51%	7.34%
	VM2	77.60%	10.63%

Conclusions and future work

- LPV model identification seems suitable to model virtualized systems dynamics
- Current work aims at:
  - Analysis of real applications
  - Controller design