Power and Performance Management of Virtualized Computing Environments via Lookahead Control

Dara Kusic[1], Jeffrey O. Kephart[2], James E. Hanson[2], Nagarajan Kandasamy[1], and Guofei Jiang[3]

1- Drexel University, Philadelphia, PA 19104

2- IBM T.J. Watson Research Center, Hawthorne, NY 10532

3- NEC Labs America, Princeton, NJ 08540

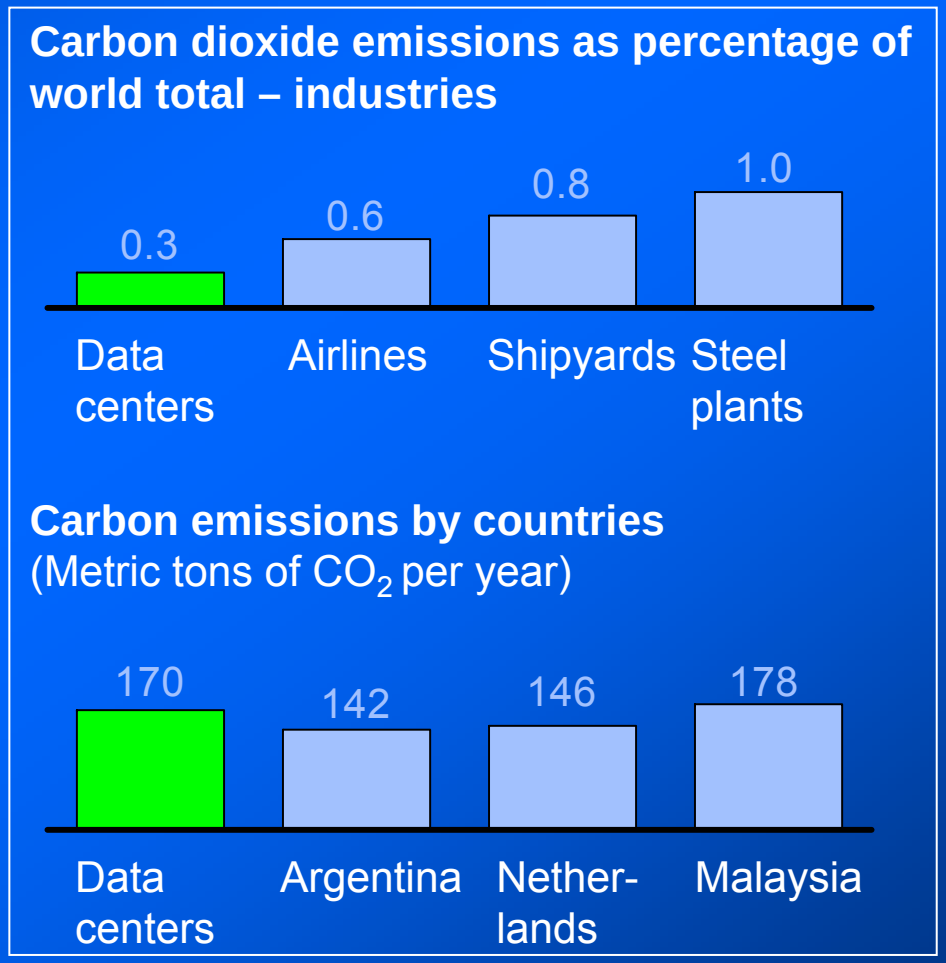
IEEE International Conference on Autonomic Computing, 2008

OUTLINE

- Motivation and problem statement
- Description of the experimental testbed
- Problem formulation and controller design
- Performance results
- Conclusions

DATA-CENTER ENERGY COSTS

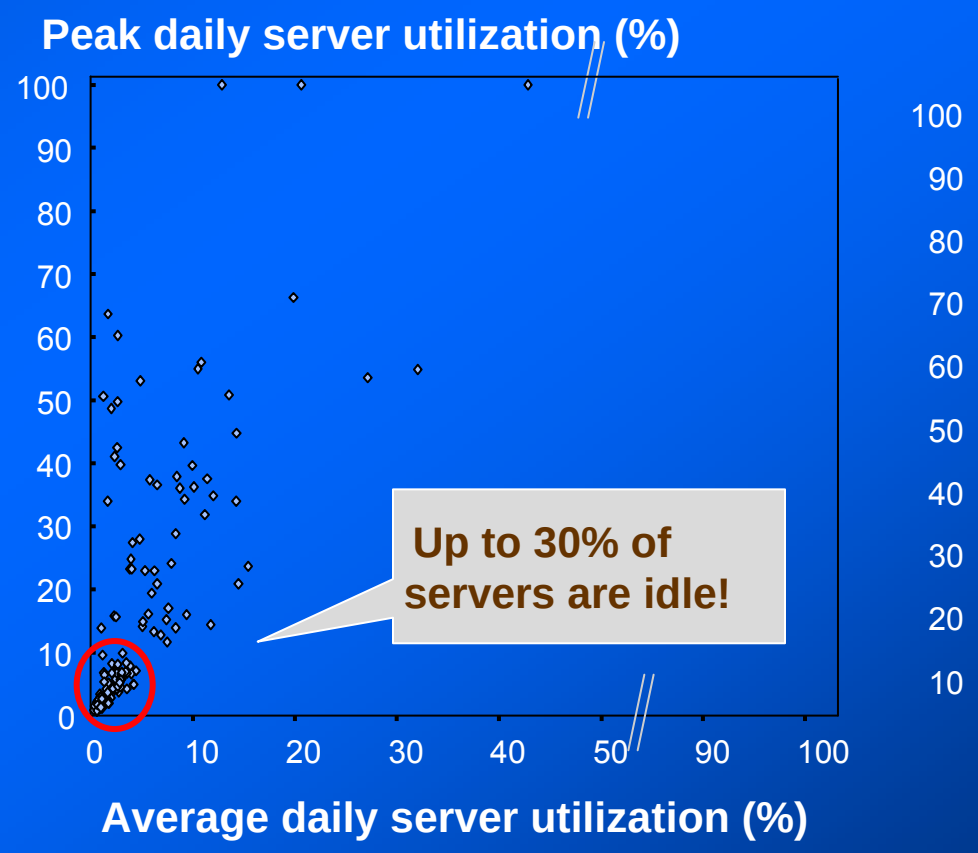
- Server energy consumption is growing at 9% per year
- Data centers are projected to surpass the airline industry in CO_2 emissions by 2020



McKinsey & Co. Report:
http://uptimeinstitute.org/content/view/168/57

SERVER UTILIZATION IN DATA CENTERS

- Server utilization averages about 6%, accounting for idle servers



McKinsey & Co. Report:
http://uptimeinstitute.org/content/view/168/57

VIRTUALIZATION AS THE ANSWER

- Performance-isolated platforms, called **virtual machines** (VMs), allow resources (e.g., CPU, memory) to be shared on a single server
- Enables consolidation of online services onto fewer servers
- Increases per-server utilization and mitigates "server sprawl"
- Enables **on-demand computing**, a provisioning model where resources are dynamically provisioned as per workload demand

Technique	Efficiency Impact
Selectively turn off core components to increase remaining unit efficiency	• 3-5%
Deploy virtualization for existing and new demand	• 25-30%
Implement free cooling	• 0-15%
Introduce greener and more power efficient servers	• 10-20%

McKinsey & Co. Report: http://uptimeinstitute.org/content/view/168/57

PROBLEM STATEMENT

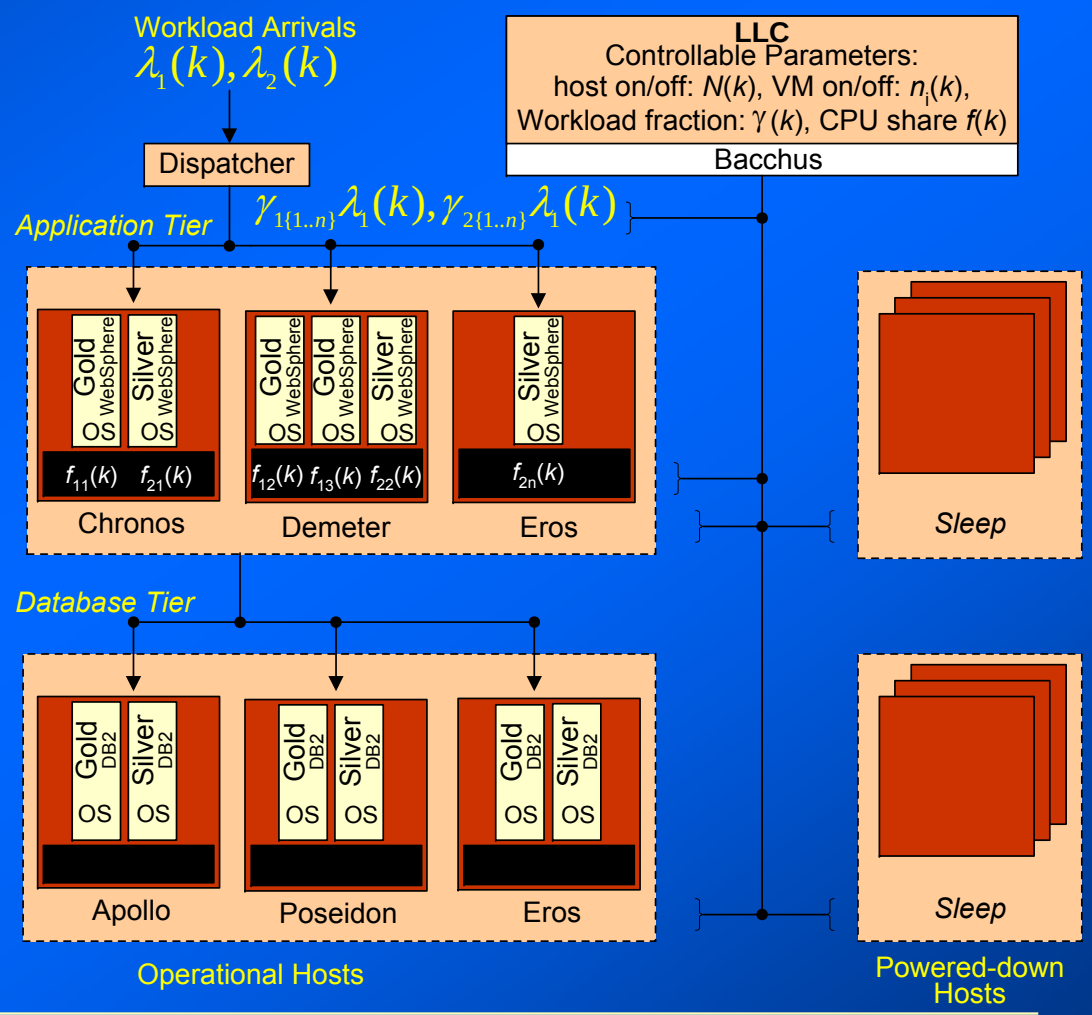
- We address combined power and performance management in a virtualized computing environment
  - The problem is posed as one of sequential optimization under uncertainty and solved using limited look-ahead control (LLC)
  - The notion of risk is encoded explicitly in the problem formulation
- Summary of main results
  - A server cluster managed using LLC **saves 26% in power-consumption costs** over a 24 hour period when compared to an uncontrolled system
  - Power savings are achieved with very few SLA violations (1.6% of the total number of requests)

OUTLINE

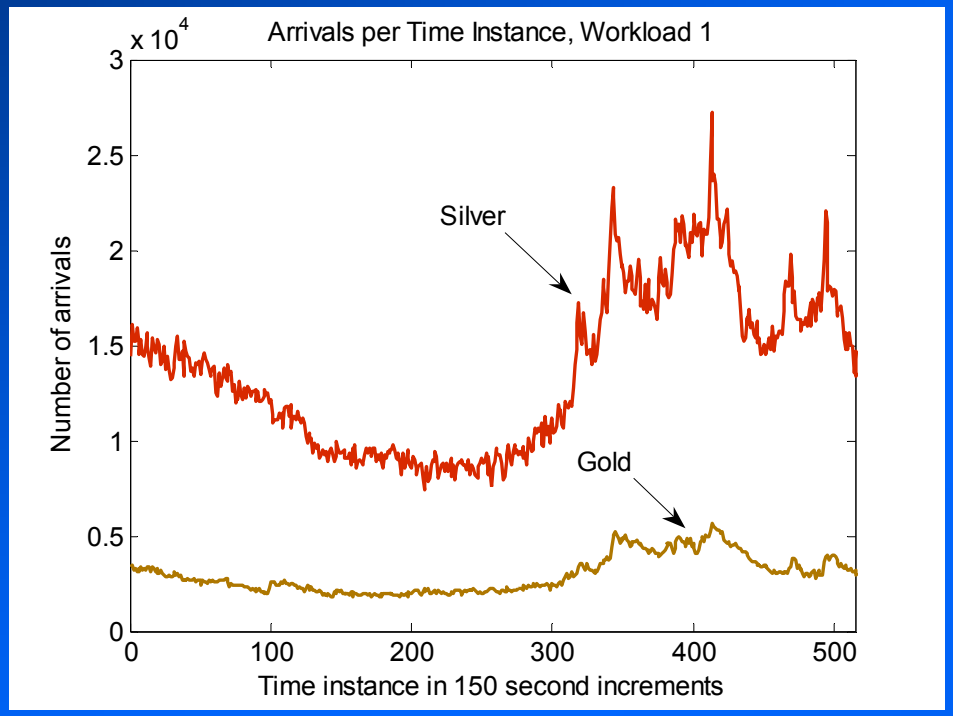
- Motivation and problem statement
- Description of the experimental testbed
- Problem formulation and controller design
- Performance results
- Conclusions

THE EXPERIMENTAL TESTBED

- The testbed is a two-tier architecture with front-end application servers and back-end databases
- It hosts two online services (**Gold** and **Silver**)
- Servers are virtualized
- Performance goals
  - Minimize power consumption
  - Minimize SLA violations
- We target the application and the database tiers



CHARACTERISTICS OF THE INCOMING WORKLOAD

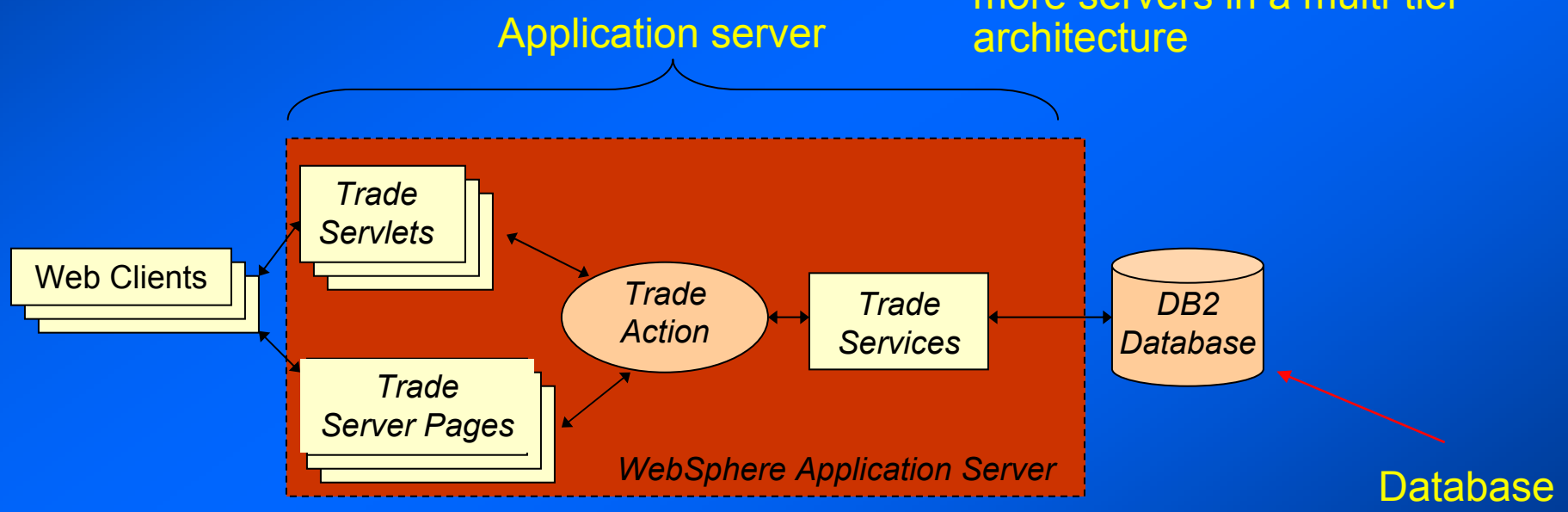


- We assume a session-less workload, i.e., incoming requests are independent of each other
- The transaction mixed is fixed to a constant proportion of browse/buy requests

- The workload to the computing system is time varying and shows significant variability over short time periods

APPLICATION ENVIRONMENT

- Online services are enabled by enterprise applications
- Trade6 is an example
  - It is transaction-based stock trading application from IBM
  - It can be hosted across one or more servers in a multi-tier architecture



OUTLINE

- Motivation and problem statement
- Description of the experimental testbed
- Problem formulation and controller design
- Performance results
- Conclusions

PROBLEM FORMULATION

- The power/performance management problem is posed as a dynamic resource provisioning problem under dynamic operating constraints
- Objectives
  - Maximize the profit generated by the system (i.e., minimize SLA violations and the power consumption cost)
- Control or decision variables to be optimized
  - Number of VMs to provision to each service
  - The CPU share given to each VM
  - Number of servers to turn on or off
  - Fraction of incoming workload to distribute to servers hosting the service

PROBLEM FORMULATION (Contd.)

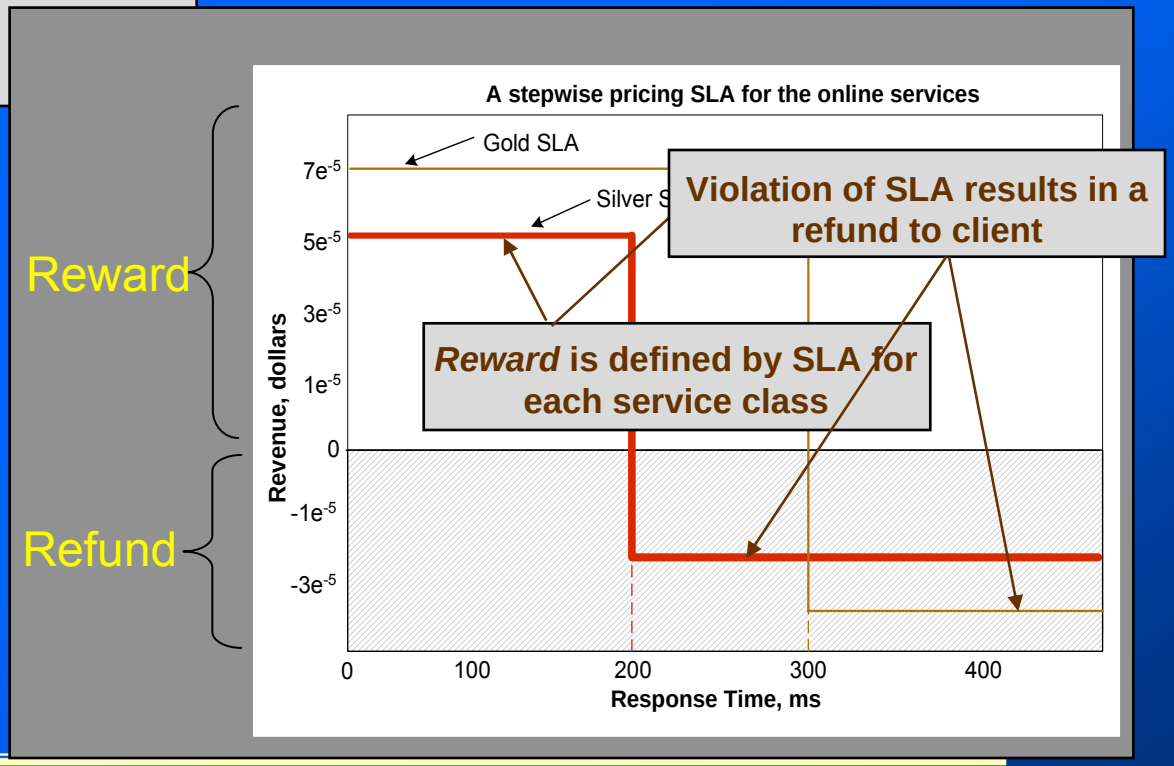
$$\underset{u}{\text{Maximize}} \sum_{k} [R(x(k),u(k)) - \psi(u(k)) - S(\Delta u(k))]$$

$$\text{Subject to:}$$

$$x(k+1) = F(x(k),u(k),\omega(k))$$

$$H(x(k)) \le 0, u(k) \in U(x(k))$$

- Dollars generated
  - Obtained as per a (nonlinear) reward-refund curve specified by the SLA



PROBLEM FORMULATION (Contd.)

$$\underset{u}{\text{Maximize}} \sum_{k} [R(x(k),u(k)) - O(u(k)) - S(\Delta u(k))]$$

$$\text{Subject to:}$$

$$x(k+1) = F(x(k),u(k),\omega(k))$$

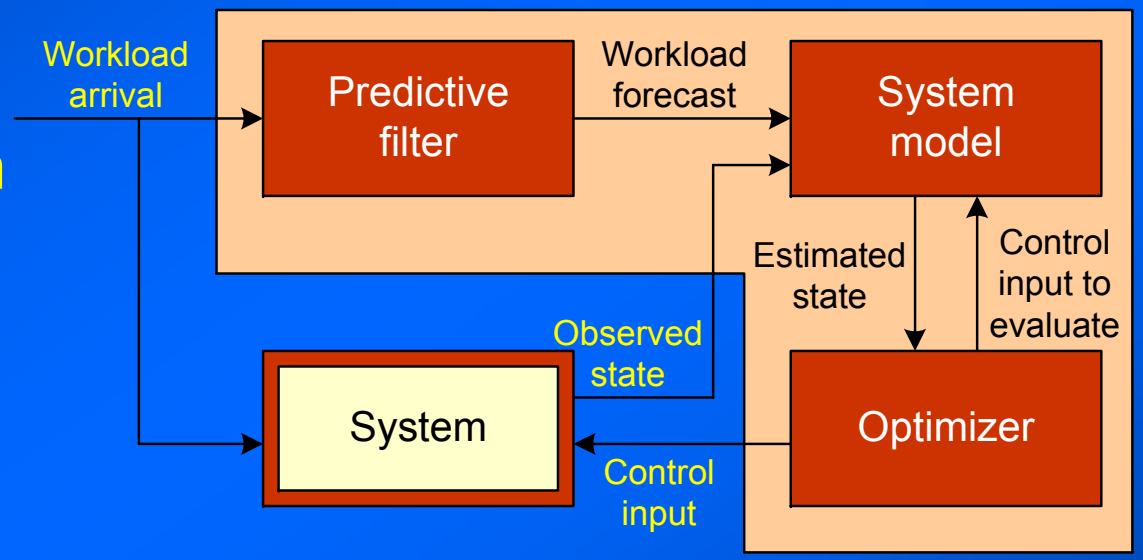
$$H(x(k)) \le 0, u(k) \in U(x(k))$$

- Power consumption cost of operating servers
- Switching costs
  - Opportunity cost lost due to the unavailability of servers/VMs involved in provisioning decisions
  - Transient power consumption costs

- Key characteristics of the control problem
  - Some control actions have (long) *dead times*; e.g., switching on a server, instantiating VMs, migrating VMs
  - Decision variables must be optimized over a *discrete domain*
  - Optimization must be performed under *explicit constraints*
- We use a **limited look-ahead control** (**LLC**) concept

THE LLC FRAMEWORK

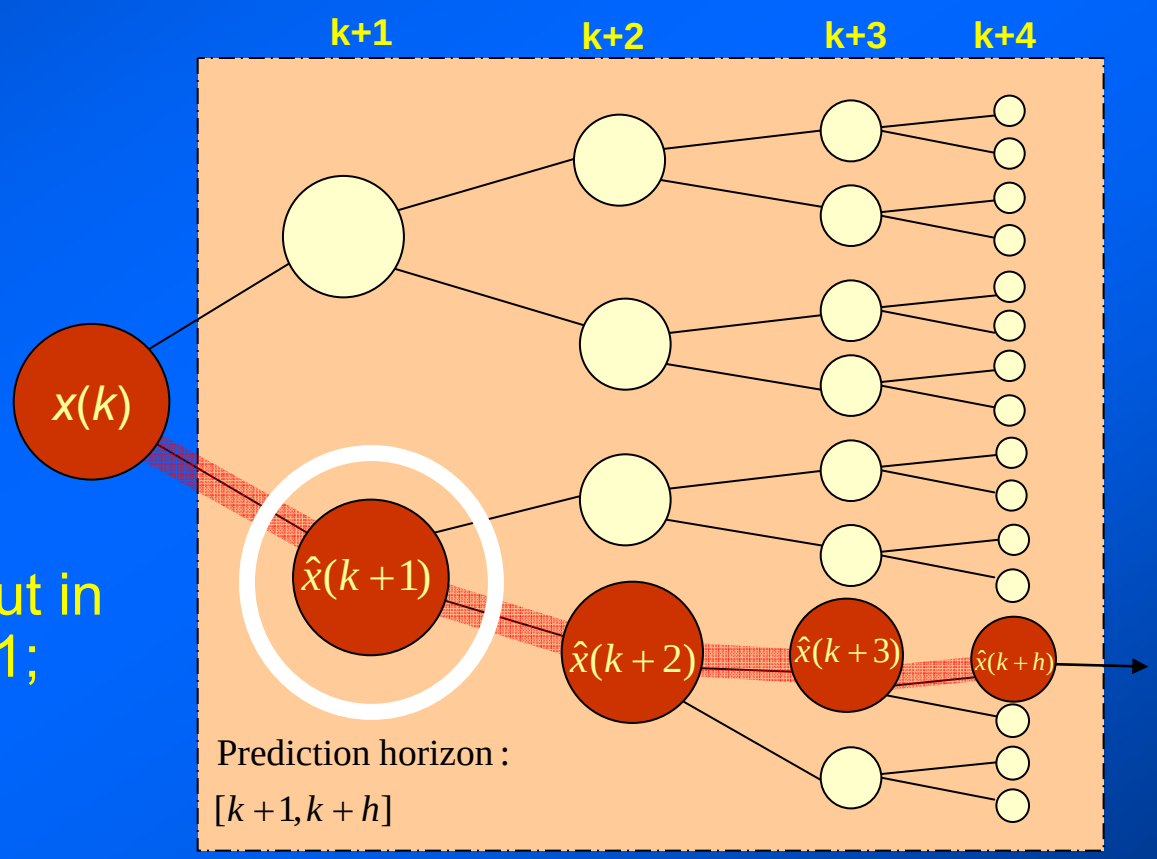
- LLC uses concepts from model predictive or receding horizon control



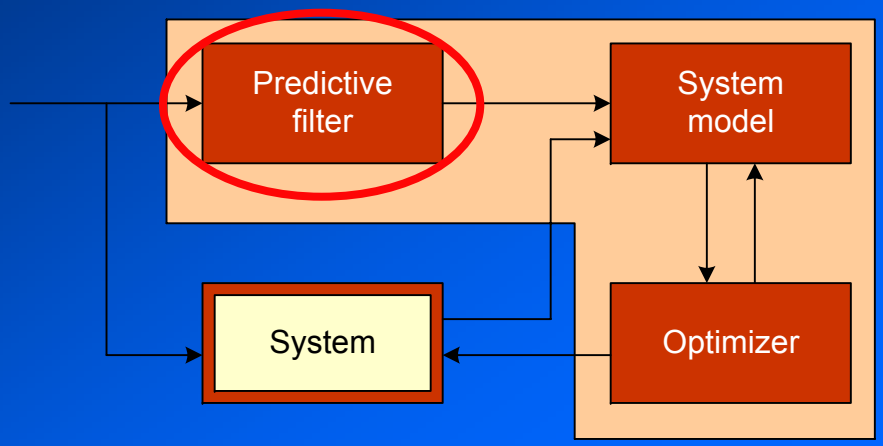
- Advantages
  - Systematic use of predictions to improve control performance
  - Robust operation in dynamic operating conditions
  - Inherent compensation for dead times
  - Multi-objective and non-linear optimization in the discrete domain under explicit constraints

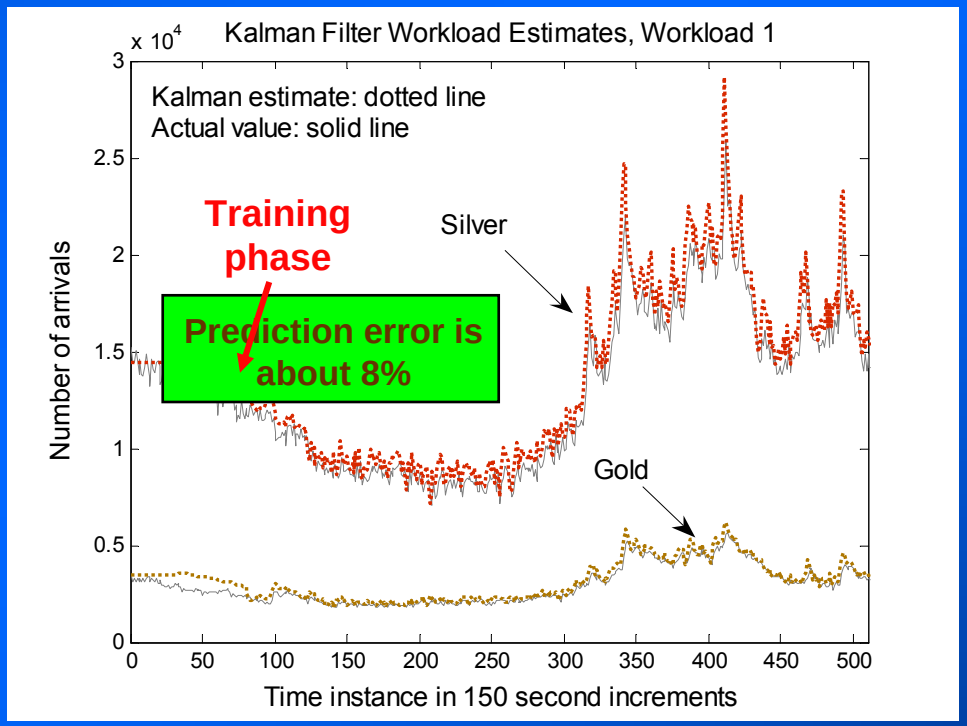
THE LLC FRAMEWORK (Contd.)

- Use a system model to estimate future system states over a prediction horizon
- Obtain an "optimal" sequence of control inputs
- Apply the first control input in the sequence at time k +1; discard the rest



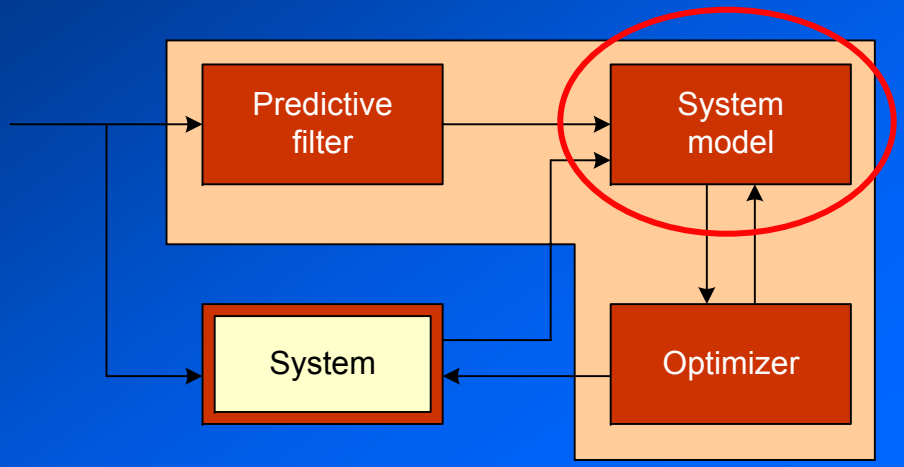
WORKLOAD ESTIMATION USING A PREDICTIVE FILTER

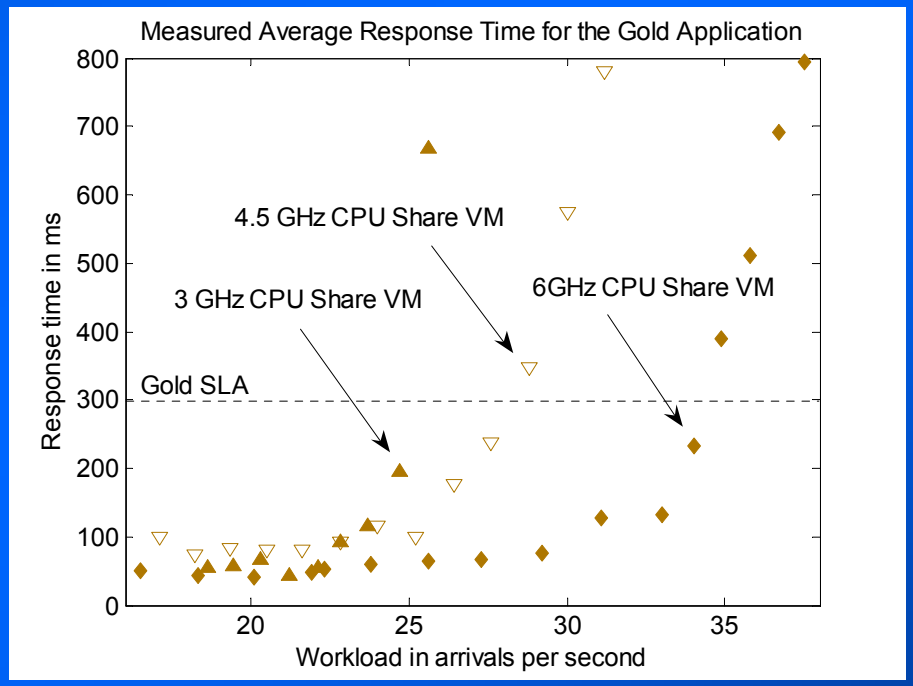




- A Kalman filter is used to estimate the workload over the prediction horizon

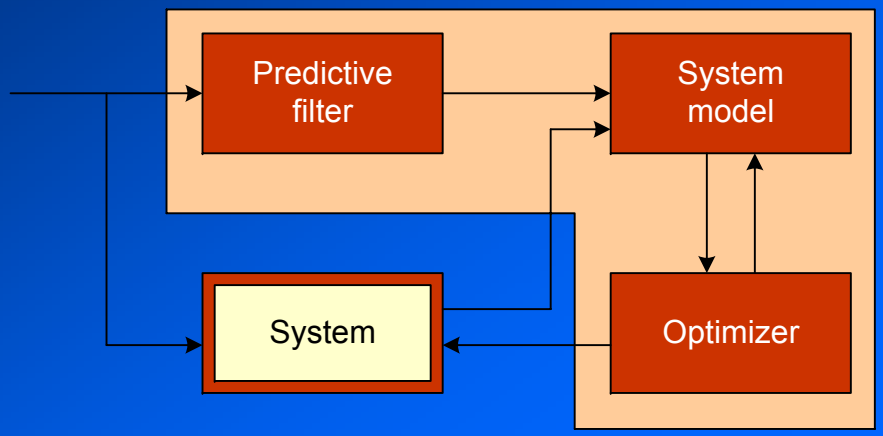
CONSTRUCTING THE SYSTEM MODEL

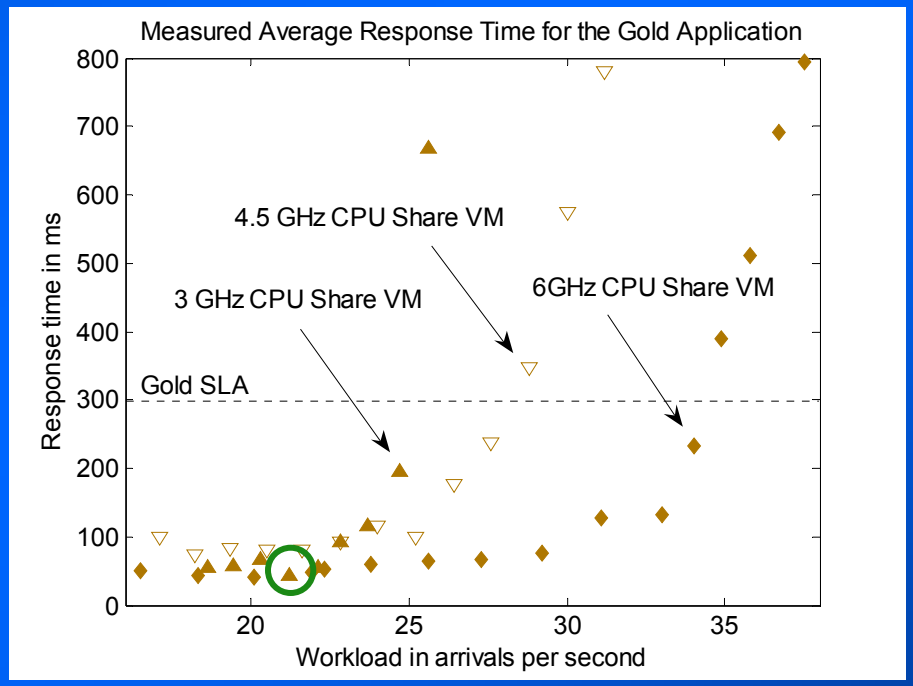




- The behavior of each application is captured using simulation-based learning and stored in an approximation structure (e.g., lookup table, neural network)

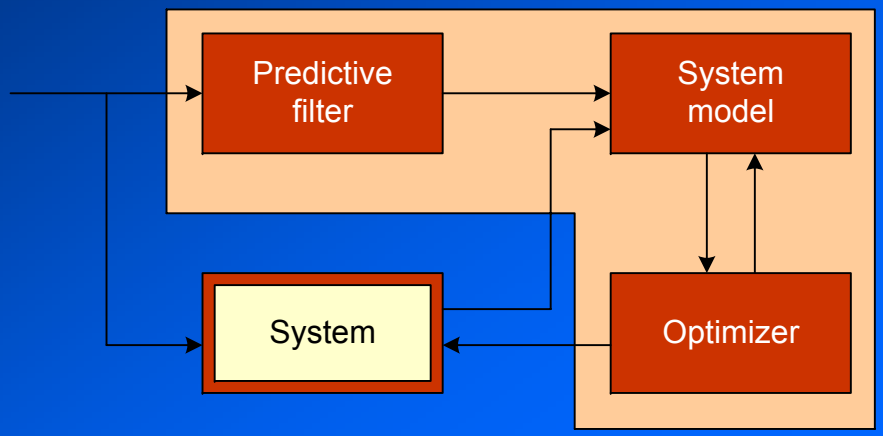
CONSTRUCTING THE SYSTEM MODEL

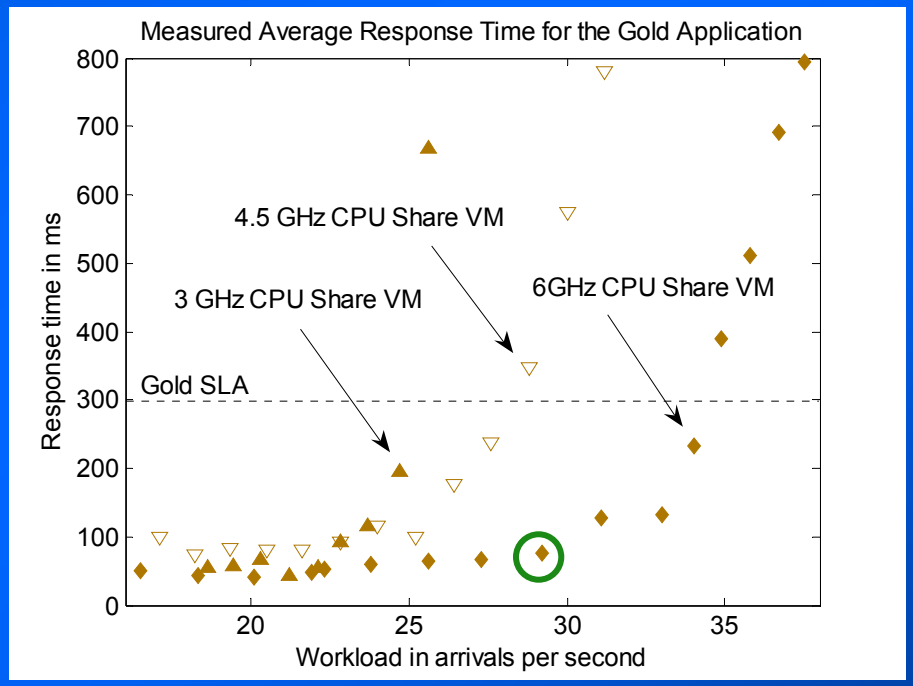




- Example 1: Given a 3 GHz CPU share and 1 GB of memory, how many requests can a Gold VM handle before incurring SLA violations?

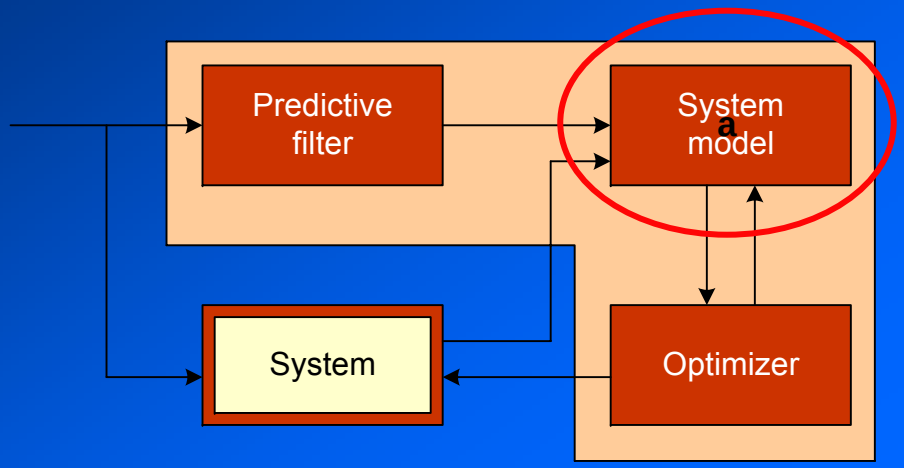
CONSTRUCTING THE SYSTEM MODEL

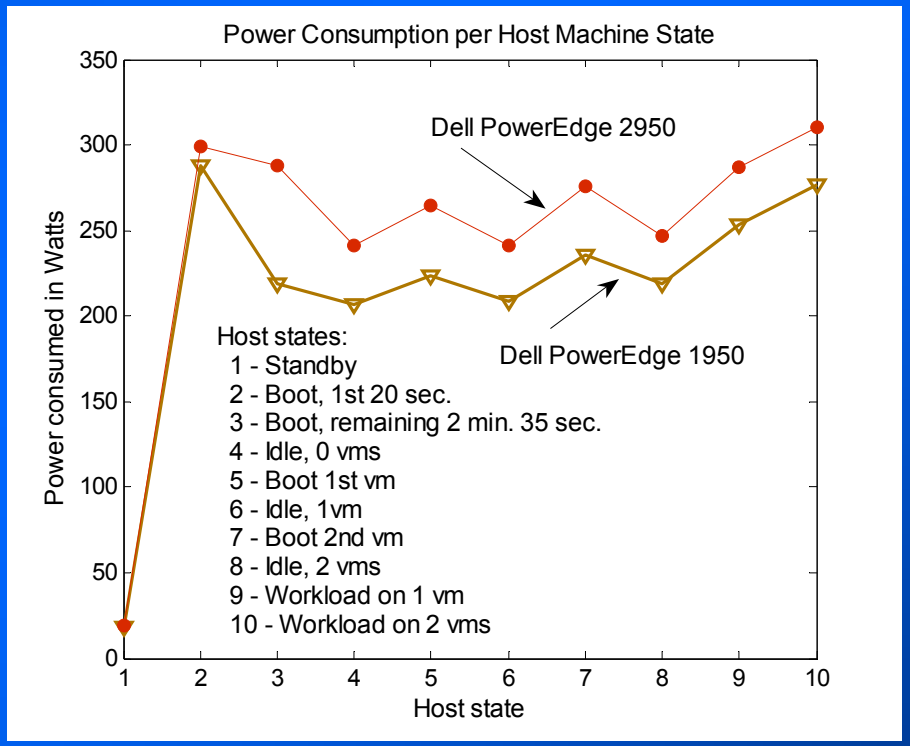




- Example 2: Given a 6 GHz CPU share and 1 GB of memory, how many requests can a Gold VM handle before incurring SLA violations?

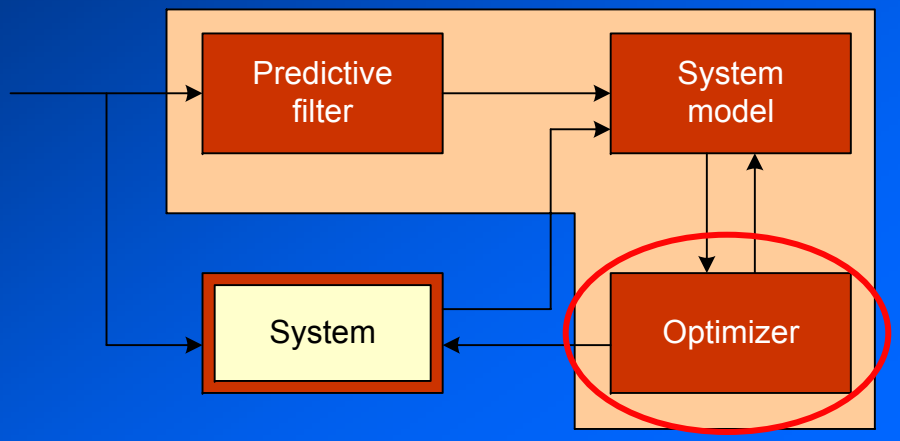
CONSTRUCTING THE SYSTEM MODEL (Contd.)





- Using a clamp-style ammeter, we measured the current drawn by the Dell servers in different operating states and multiplied it by the rated wall-supply voltage

DEVELOPING THE OPTIMIZER



- Issue 1: Risk-aware control
  - Due to the energy and opportunity costs incurred when switching hosts and VMs on/off, excessive switching caused by workload variability may actually reduce profits
  - We need to encode a notion of risk in the cost function

Cost that accumulates during the time a server is being turned on but is unavailable to perform any useful service

RISK-AWARE CONTROL

- Environment-input estimates will have prediction errors
- We encode a notion of risk in the optimization problem
  - Generate a set of expected next states for a range of the predicted environment inputs

Construct an uncertainty band for environment input of interest:

$$\left(\hat{\lambda}_i(j) - \varepsilon_i(j)\right) \leq \hat{\lambda}_i(j) \leq \left(\hat{\lambda}_i(j) + \varepsilon_i(j)\right)$$

Workload prediction and the error envelope

Generate possible next states

$$\hat{x}_i(j) \in \hat{X}_i(j)$$

Estimated states

RISK-AWARE CONTROL (Contd.)

- A utility function encodes risk into the objective function

Apply a mean-square variance model of utility

$$\text{Utility}(R_i) = A \cdot \text{mean}\left(\hat{X}_i(j)\right) - \beta \cdot \left(\text{var}\left(\hat{X}_i(j)\right) + \text{mean}\left(\hat{X}_i(j)\right)^2 \right) \quad \forall i$$

Utility model with tunable risk-preference parameter, β

Uncertainty as variance

Tunable risk-preference parameter, β

$\beta < 0$: risk-seeking

$\beta > 0$: risk-averse

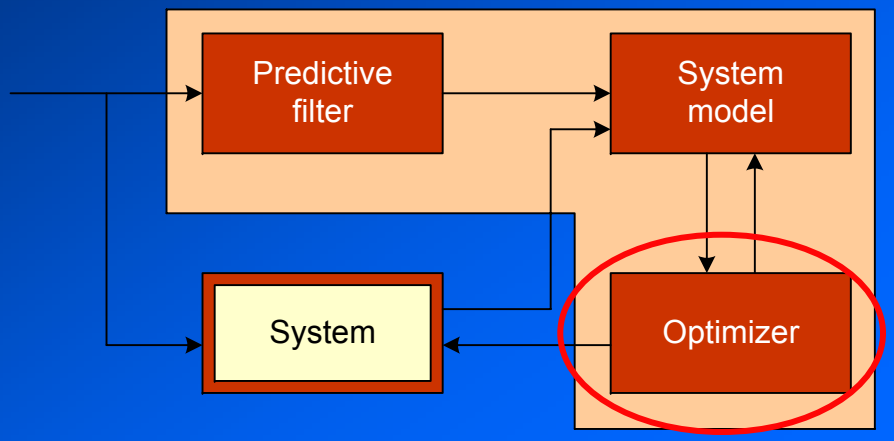
$\beta = 0$: risk-neutral

Formulate a utility maximization problem

$$\max_{\{u\}} \sum_{j=k+1}^{k+h} \sum_{i=i}^{2} U_i\left(\hat{X}_i(j), u_i(j)\right)$$

Maximize utility over horizon and client classes

DEVELOPING THE OPTIMIZER (Contd.)



- Issue 2: Execution-time overhead of the controller
  - The utility optimization problem will show an exponential increase in worst-case complexity with an increasing number of control options and longer prediction horizons

- We use a control hierarchy to reduce execution-time overhead
  - An L0 controller for each service decides the CPU share to assign to VMs
  - An L1 controller with a global view of the system decides the number of VMs for each service and the number of servers to keep powered on
  - The average execution time of the L1 controller is about 10 seconds

OUTLINE

- Motivation and problem statement
- Description of the experimental testbed
- Problem formulation and controller design
- Performance results
- Conclusions

EXPERIMENTAL SYSTEM

- Six Dell servers (models 2950 and 1950) comprise the experimental testbed
- Virtualization of the CPU and memory is enabled by VMware ESX Server 3.0
- Virtual machines run SUSE Enterprise Linux Server Edition 10
- Control directives use the VMware API, Linux shell commands, and IPMI
- Silver application is Trade6 only; Gold application is Trade6 + extra CPU load

Host name	CPU speed	# of CPU cores	Memory
Apollo	2.3 GHz	8	8 GB
Bacchus	2.3 GHz	2	8 GB
Chronos	1.6 GHz	8	4 GB
Demeter	1.6 GHz	8	4 GB
Eros	1.6 GHz	8	4 GB
Poseidon	2.3 GHz	8	8 GB

EXPERIMENTAL PARAMETERS

Parameter	Value
Cost per KiloWatt hour	$0.3
Time delay to power on a VM	1 min. 45 sec.
Time delay to power on a host	2 min. 55 sec.
Prediction horizon	L1: 3 steps, L0: 1 step
Control sampling period	L1: 150 sec, L0: 30 sec
Initial configuration for Gold service (application tier)	3 VMs
Initial configuration for Silver Service (application tier)	3 VMs

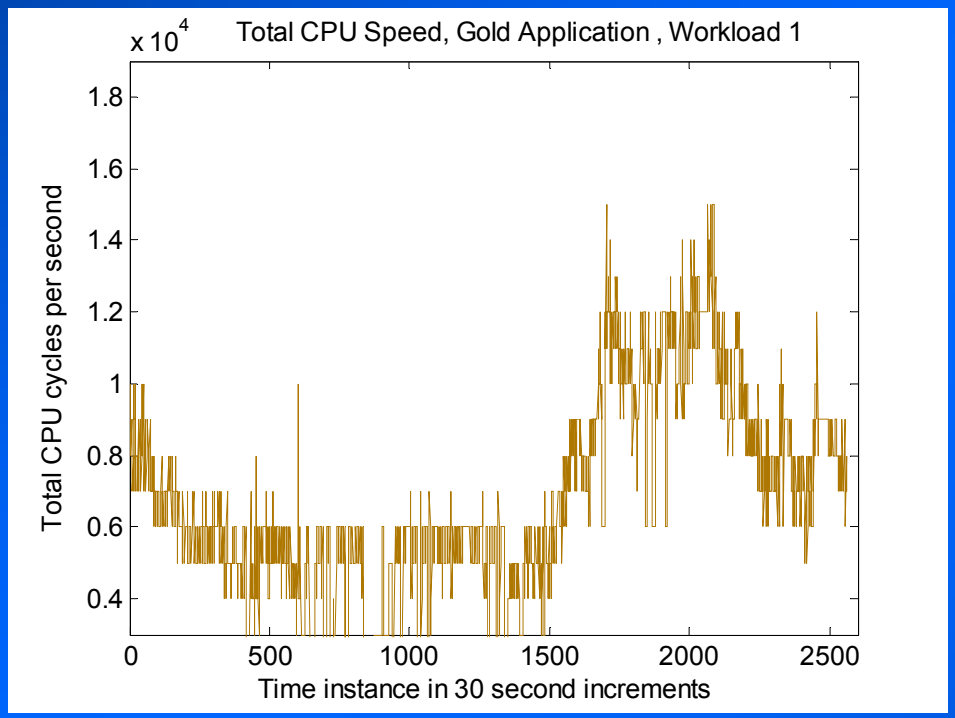
MAIN RESULTS

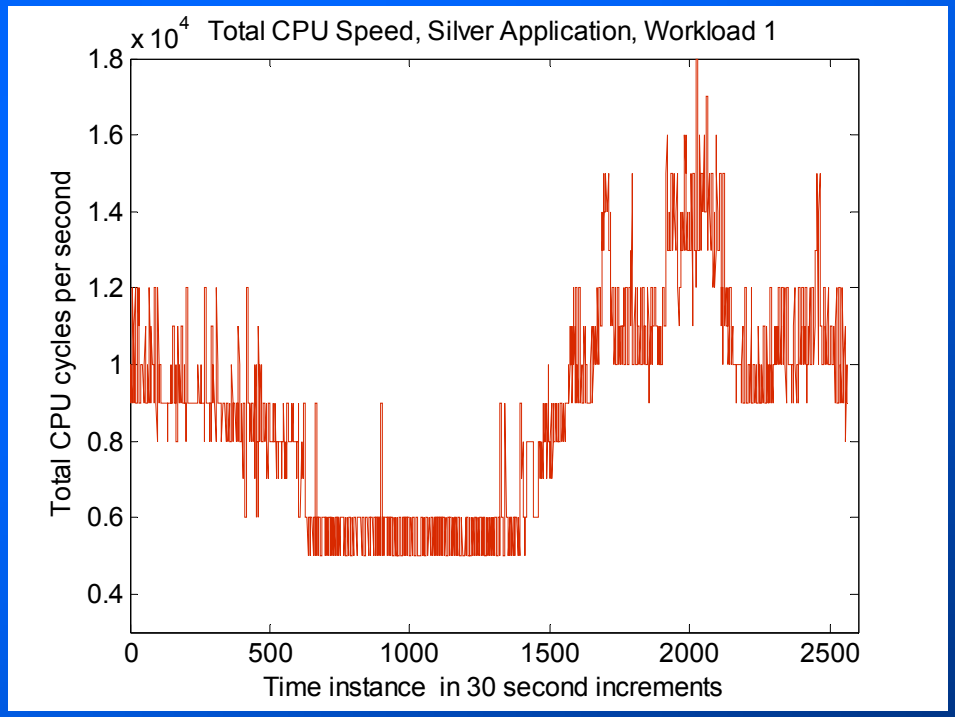
- A risk-neutral controller conserves, on average, 26% more energy than a system without dynamic control with very few SLA violations

Workload	Energy savings	% of SLA violations (Silver)	% of SLA violations (Gold)
Workload 1	18%	3.2%	2.3%
Workload 2	17%	1.2%	0.5%
Workload 3	17%	1.4%	0.4%
Workload 4	45%	1.1%	0.2%
Workload 5	32%	3.5%	1.8%

RESULTS (Contd.)

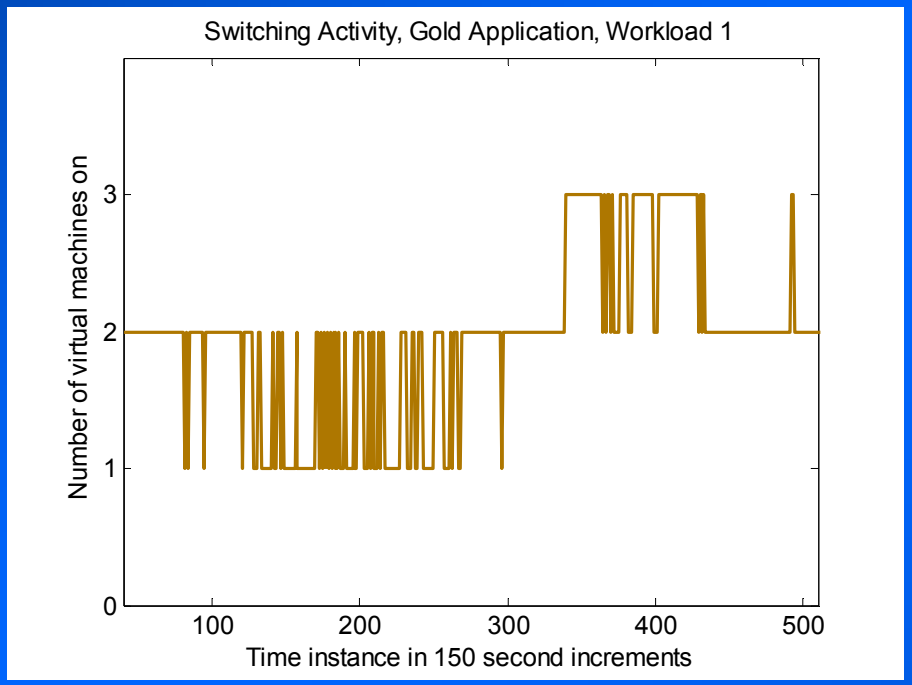
- CPU shares assigned to the Gold and Silver applications over a 24-hour period

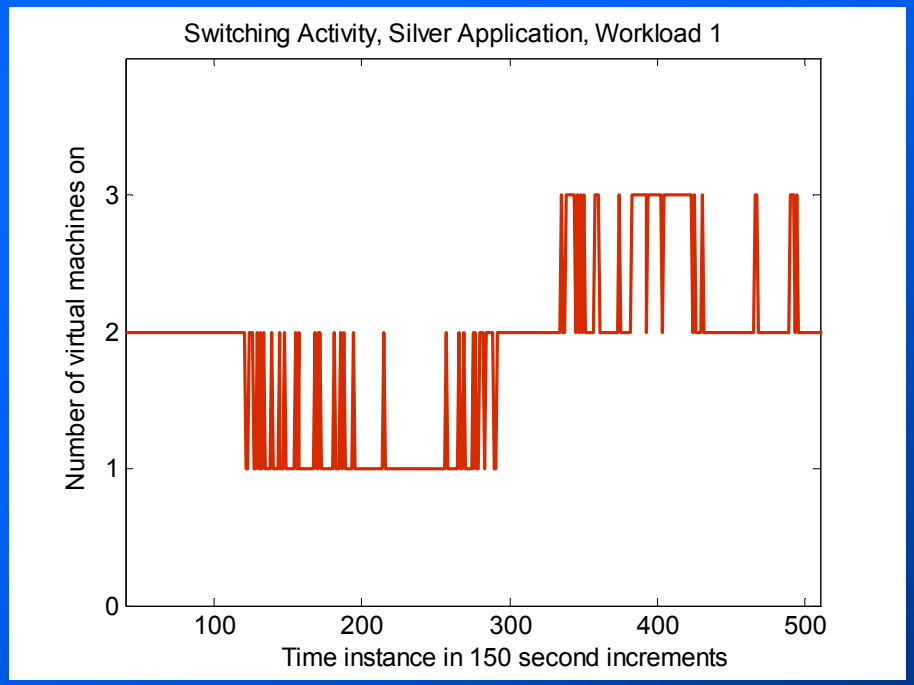




RESULTS (Contd.)

- Number of virtual machines assigned to the Gold and Silver applications over a 24-hour period





EFFECT OF THE RISK PREFERENCE PARAMETER

- A risk-averse controller conserves about the same amount of energy as a risk-neutral controller

Workload	Energy savings (risk-neutral control) ($\beta = 0$)	Energy savings (risk-averse control) ($\beta = 2$)
Workload 6	20.8 %	20.9 %
Workload 7	25.3 %	25.2 %

EFFECT OF THE RISK PREFERENCE PARAMETER (Contd.)

- A risk-averse controller maintains a higher QoS than a risk-neutral controller by reducing switching activity

Workload	SLA violations (risk-neutral control) ($\beta = 0$)	SLA violations (risk-averse control) ($\beta = 2$)	% reduction in SLA violations
Workload 6	28,635 (2.3%)	15,672 (1.7%)	45%
Workload 7	34,201 (2.7%)	25,606 (2.0%)	25%

Workload	Switching activity (risk-neutral control) ($\beta = 0$)	Switching activity (risk-averse control) ($\beta = 2$)	% reduction in switching activity
Workload 6	30	28	7%
Workload 7	40	30	5%

OPTIMALITY CONSIDERATIONS

- The controller cannot achieve optimal performance
  - Limited by errors in workload predictions
  - Limited by constrained control inputs
  - Limited by a finite prediction horizon
- To evaluate optimality, profit gains of a risk-neutral and best-case risk-averse controller were compared against an "oracle" controller with perfect knowledge of the future

Controller	Total Energy Savings	Total SLA violations	Num. times hosts switched
Risk neutral	25.3%	34,201 (2.7%)	40
Risk averse	25.2%	25,606 (2.0%)	38
Oracle	16.3%	14,228 (1.1%)	32

CONCLUSIONS

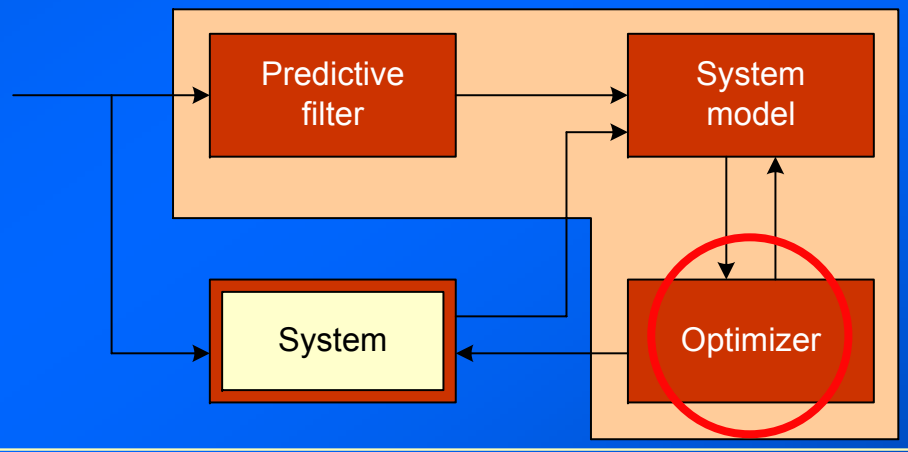
- We have addressed power and performance management in a virtualized computing environment within a LLC framework
- The cost of control and the notion of risk is encoded explicitly in the problem formulation
- A server cluster managed using LLC **saves 26% in power-consumption costs** over a 24 hour period when compared to an uncontrolled system
- Power savings are achieved with very few SLA violations (1.6% of the total number of requests)
- Our recommendation is a risk-averse controller since it reduces SLA violations and switching activity

QUESTIONS

- **Published work:**
  - **D. Kusic and N. Kandasamy, "Risk-Aware Limited Lookahead Control for Dynamic Resource Provisioning in Enterprise Computing Systems," *Proc. IEEE Conference on Autonomic Computing (ICAC)*, June 2006.**
  - **D. Kusic and N. Kandasamy. Approximation modeling for the online performance management of distributed computing systems. In *Proc. of IEEE Intl. Conf. on Autonomic Computing (ICAC)*, June 2007.**
  - **D. Kusic and N. Kandasamy, "Risk-Aware Limited Lookahead Control for Dynamic Resource Provisioning in Enterprise Computing Systems," *Journal of Cluster Computing,* December 2007.**
  - **D. Kusic, N. Kandasamy and G. Jiang, "Approximation Modeling for the Online Performance Management of Distributed Computing Systems," to appear in *IEEE Transactions on Systems, Man and Cybernetics, 2008.***

SCALABILITY

- Execution time of the controller can be reduced through various techniques
  - Approximating control
  - Implementing the controller in hardware
  - Increasing the number of tiers in the control hierarchy
  - Simplifying the iterative search process to “hold” a control input constant over the prediction horizon



- A neural network or regression tree can be trained to learn the decision-making behavior of the optimzer

TRANSACTION MIX ADAPTATION

- The controller can adapt to different transaction mixes by interpolating between models