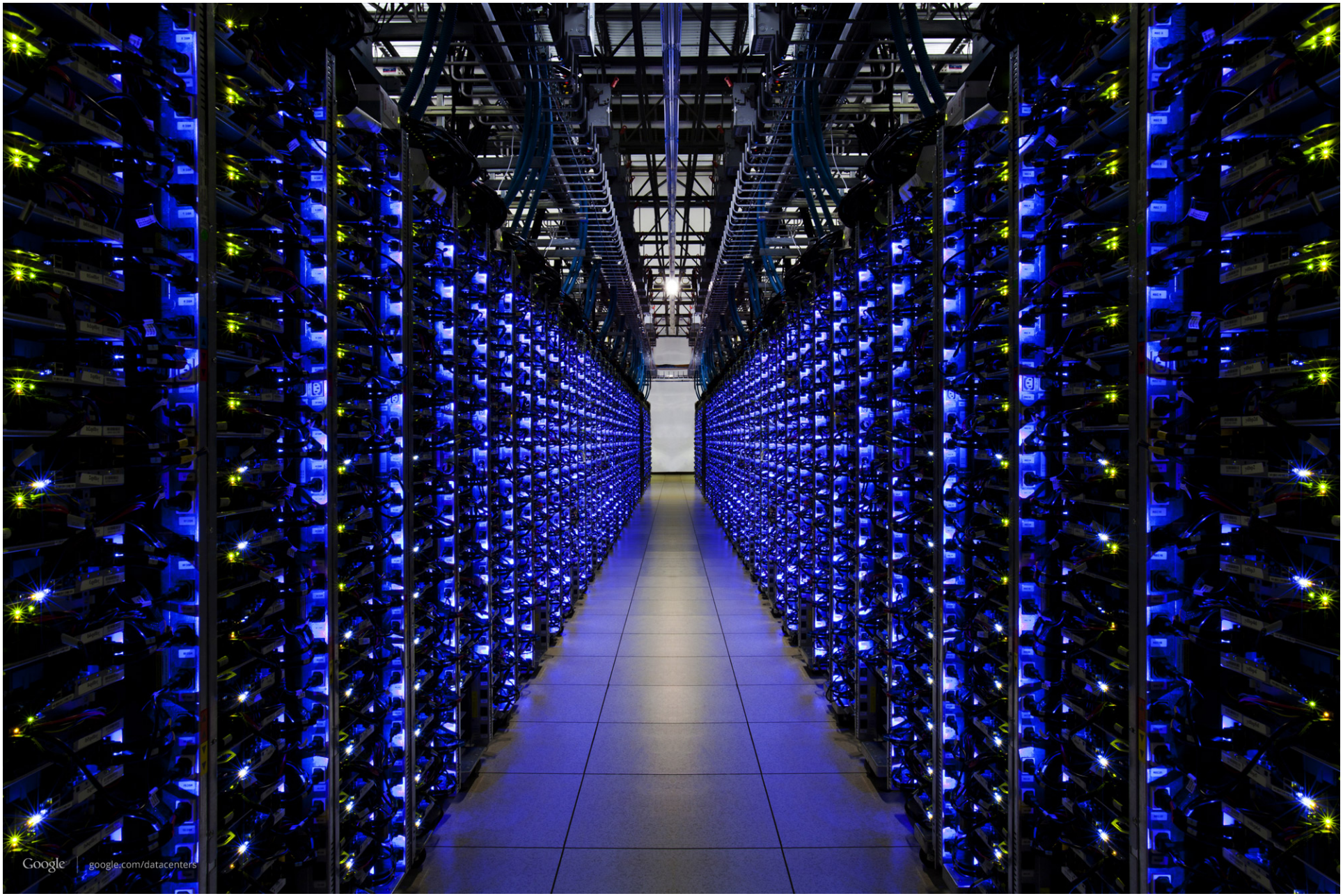
Autonomic Computing: a new design principle for complex systems

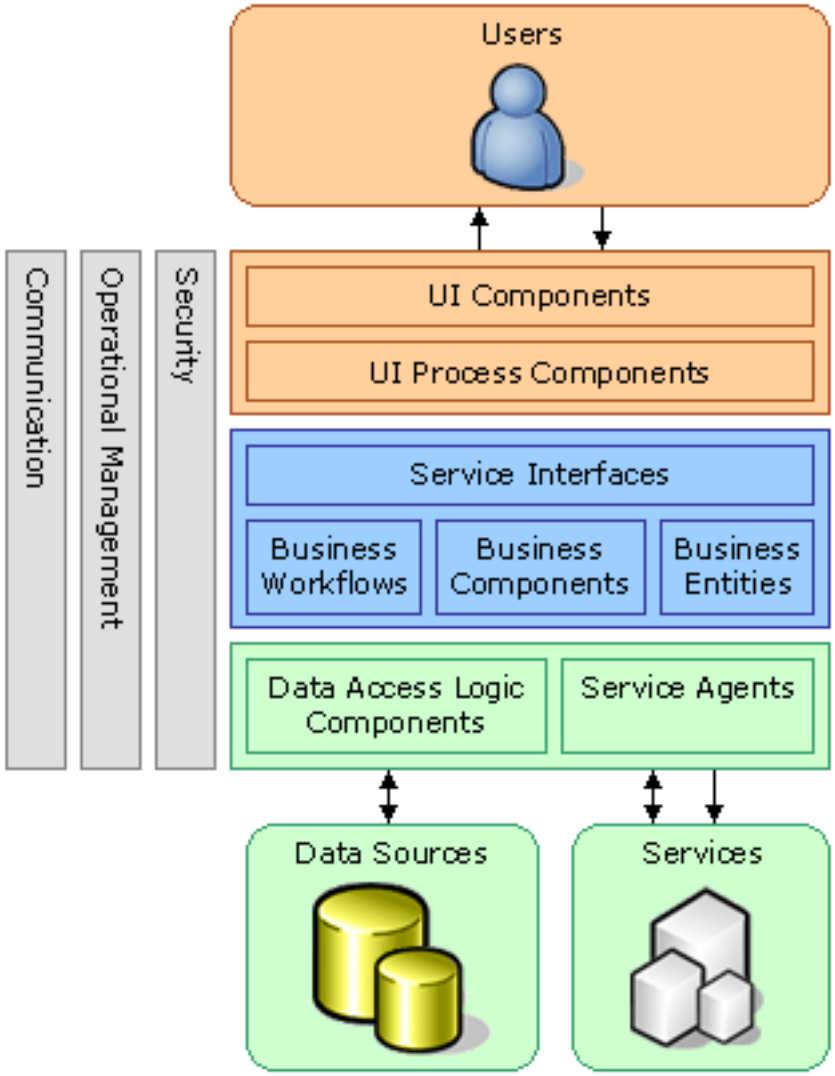
Danny Menascé
Department of Computer Science
George Mason University
www.cs.gmu.edu/faculty/menasce.html

Amazon EC2

The New York Times



Layered Software Architecture



Outline

- Motivation for Autonomic Computing
- Techniques used in AC
  - Model-driven
    - Performance model
    - Control theory
  - Model-free
    - Machine learning (e.g., reinforcement learning)
    - Statistical learning
- Applications
  - Internet data centers, virtual machine management, e-commerce and Web-systems, service oriented computing, cloud computing resource management, databases, adaptive software systems, and emergency departments
- Concluding Remarks

Outline

- Motivation for Autonomic Computing
- Techniques used in AC
  - Model-driven
    - Performance model
    - Control theory
  - Model-free
    - Machine learning (e.g., reinforcement learning)
    - Statistical learning
- Applications
  - Internet data centers, virtual machine management, e-commerce and Web-systems, service oriented computing, cloud computing resource management, databases, adaptive software systems, and emergency departments
- Concluding Remarks

Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code

Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code
  - Skilled IT professionals required to install, configure, tune, and maintain.

Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code
  - Skilled IT professionals required to install, configure, tune, and maintain.
  - Need to integrate many heterogeneous systems

Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code
  - Skilled IT professionals required to install, configure, tune, and maintain.
  - Need to integrate many heterogeneous systems
  - Limit of human capacity being achieved

Motivation for AC (cont'd)

- Harder to anticipate interactions between components at design time:
  - Need to defer decisions to run time

Motivation for AC (cont'd)

- Harder to anticipate interactions between components at design time:
  - Need to defer decisions to run time
- Computer systems are becoming too massive, complex, to be managed even by the most skilled IT professionals

Motivation for AC (cont'd)

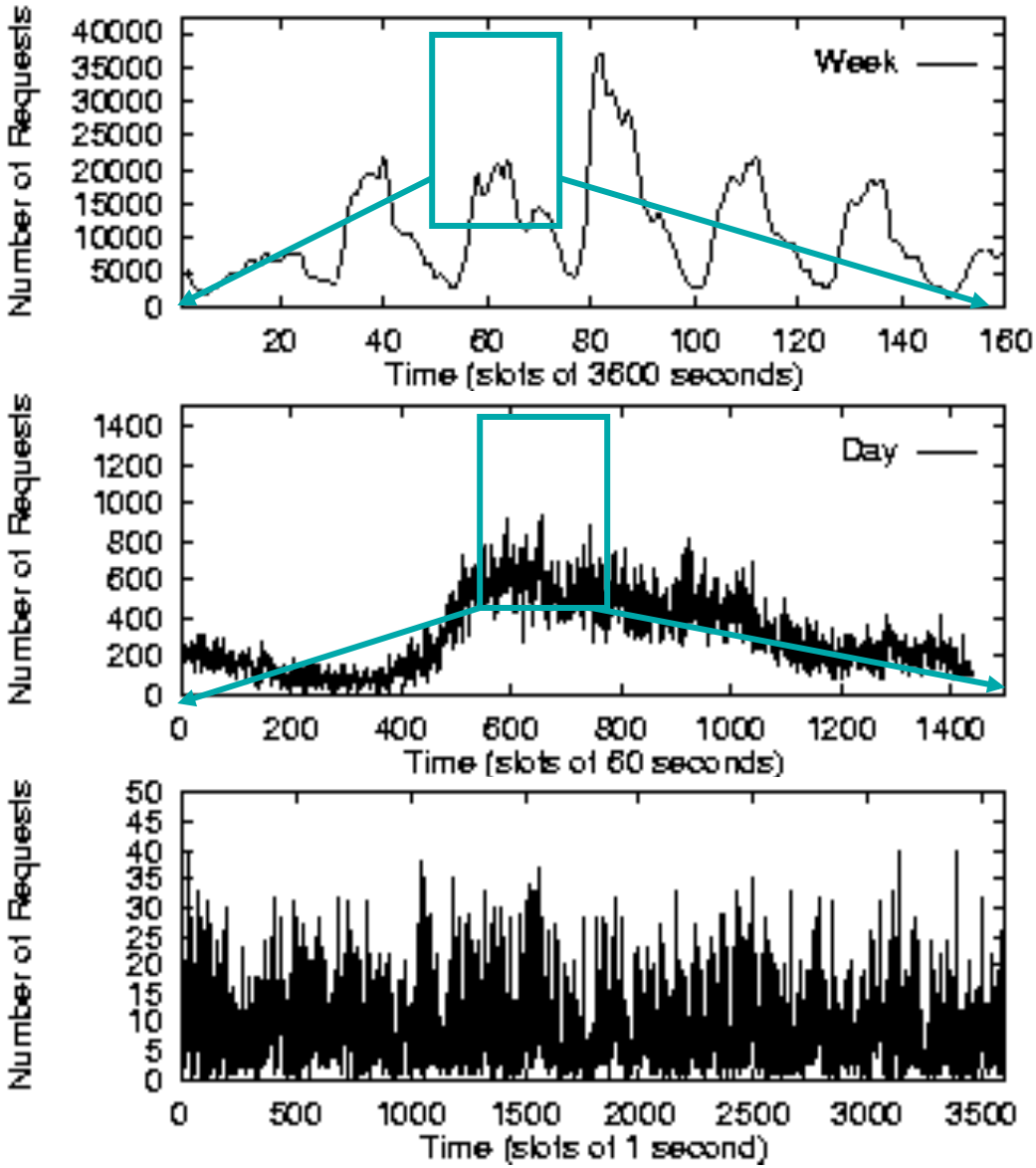
- Harder to anticipate interactions between components at design time:
  - Need to defer decisions to run time
- Computer systems are becoming too massive, complex, to be managed even by the most skilled IT professionals
- The workload and environment conditions tend to change very rapidly with time

Multi-scale time workload variation of a Web Server

3600 sec

60 sec

1 sec



Large Number of Configurations

- Complex middleware and database systems have a very large number of configurable parameters.

Web Server (IIS 5.0)	**Application Server (Tomcat 4.1)**	**Database Server (SQL Server 7.0)**
HTTP KeepAlive	acceptCount	Cursor Threshold
Application Protection Level	minProcessors	Fill Factor
Connection Timeout	maxProcessors	Locks
Number of Connections		Max Worker Threads
Logging Location		Min Memory Per Query
Resource Indexing		Network Packet Size
Performance Tuning Level		Priority Boost
Application Optimization		Recovery Interval
MemCacheSize		Set Working Set Size
MaxCachedFileSize		Max Server Memory
ListenBacklog		Min Server Memory
MaxPoolThreads		User Connections
worker.ajp13.cachesize		

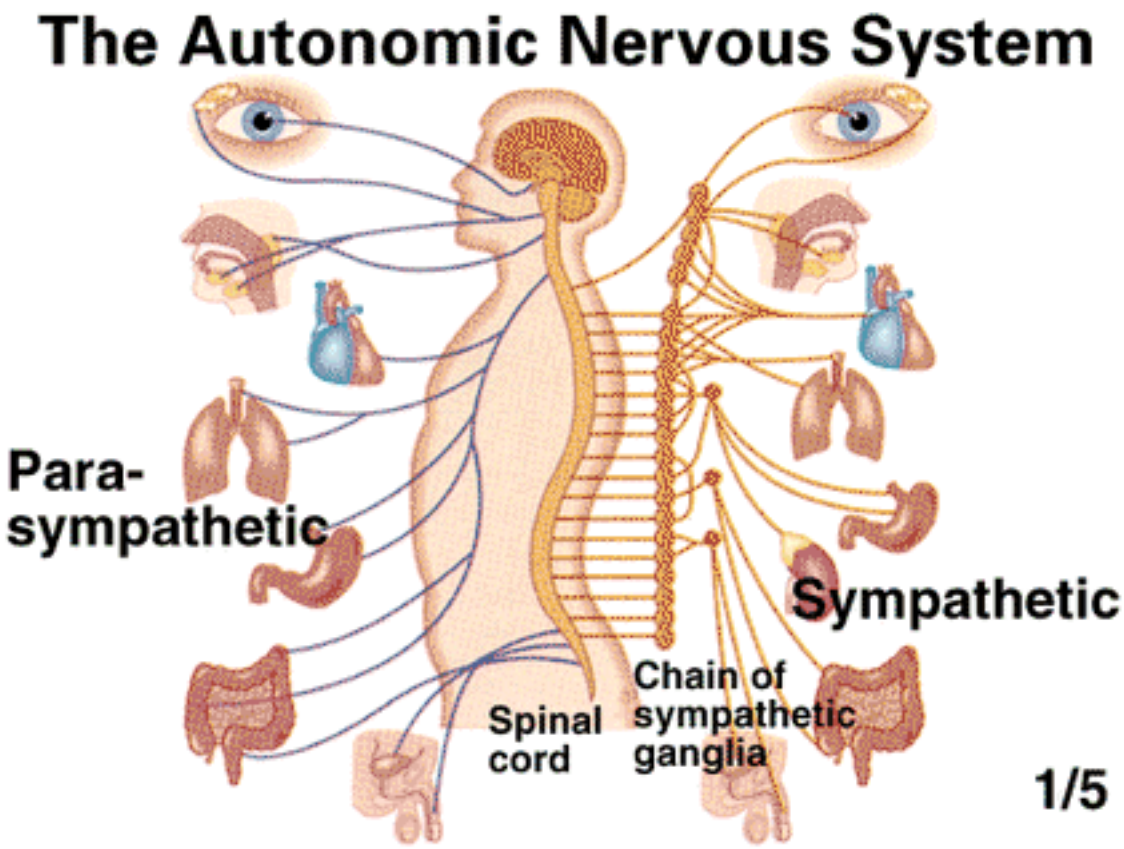
Autonomic Computing

- Systems that can manage themselves given high-level objectives expressed in term of service-level objectives or utility functions.
  - Average response time < 1.0 sec
  - Response time of 95% of transactions ≤ 0.5 sec
  - Search engine throughput ≥ 4600 queries/sec
  - Availability of the e-mail portal ≥ 99.978%.
  - Percentage of phishing e-mails filtered by the e-mail portal ≥ 90%

Autonomic Computing

- Autonomic computing: inspired in the human autonomic nervous system:
  - Sensory and motor neurons that run between the central nervous system and various internal organs.
  - Monitors conditions in the internal environment and effects changes in them.
    - E.g., contraction of both smooth and cardiac muscles is controlled by motor neurons of the autonomic system.
  - Functions in an involuntary and reflexive manner.

Autonomic Computing



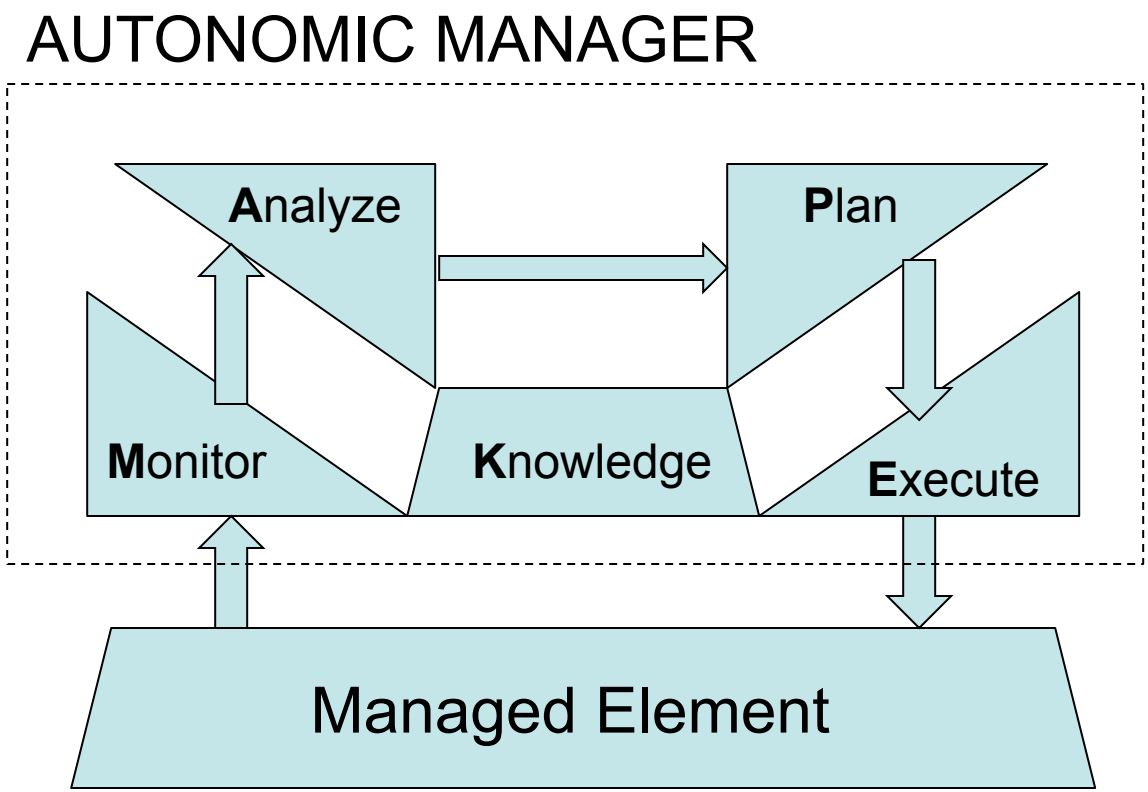
Autonomic Systems

- Self-managing
  - Self-configuring
  - Self-optimizing
  - Self-healing
  - Self-protecting
- Self-* systems

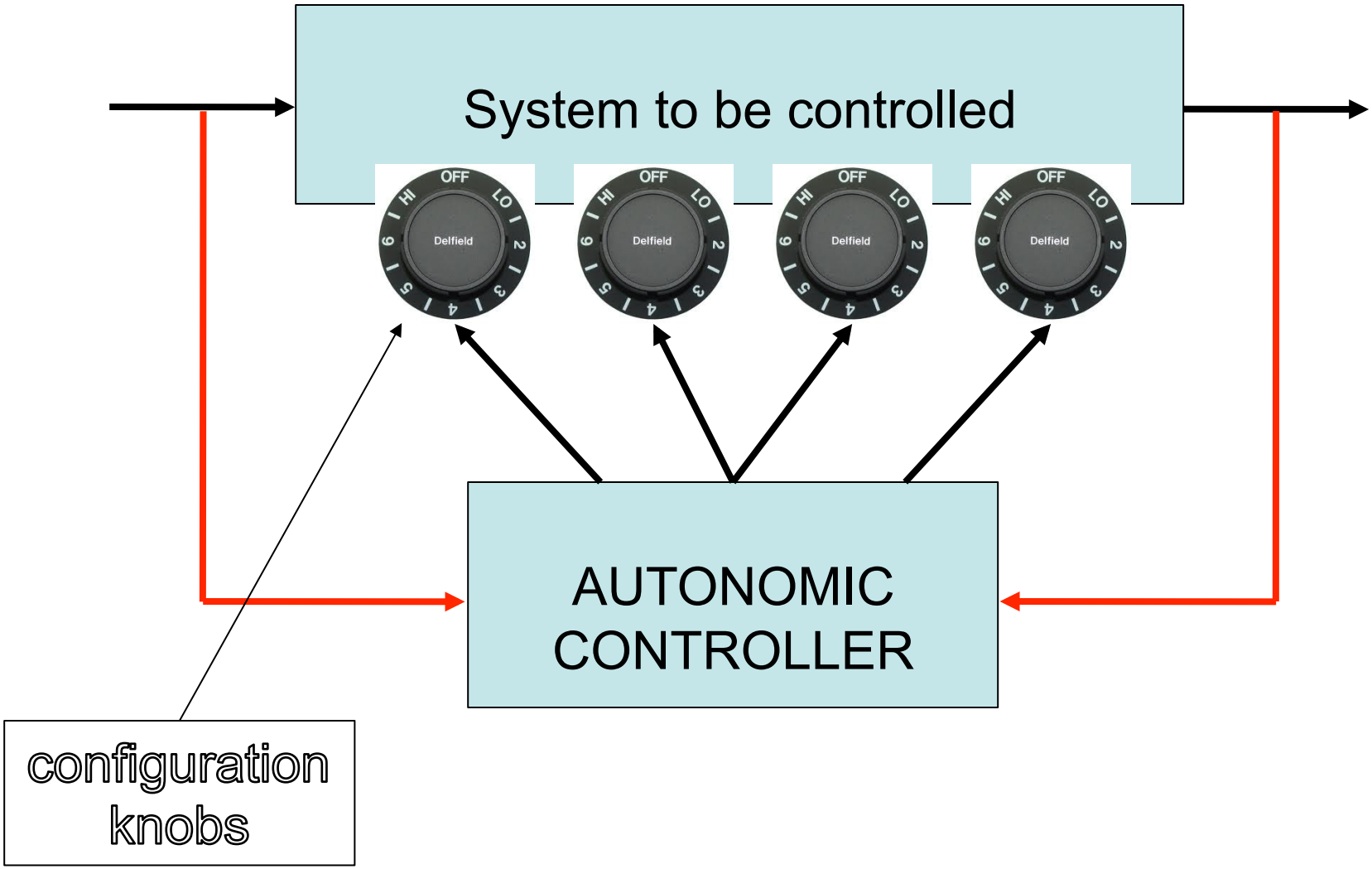
Autonomic Systems

- Self-managing
  - Self-configuring
  - Self-optimizing
  - Self-healing
  - Self-protecting
- Self-* systems

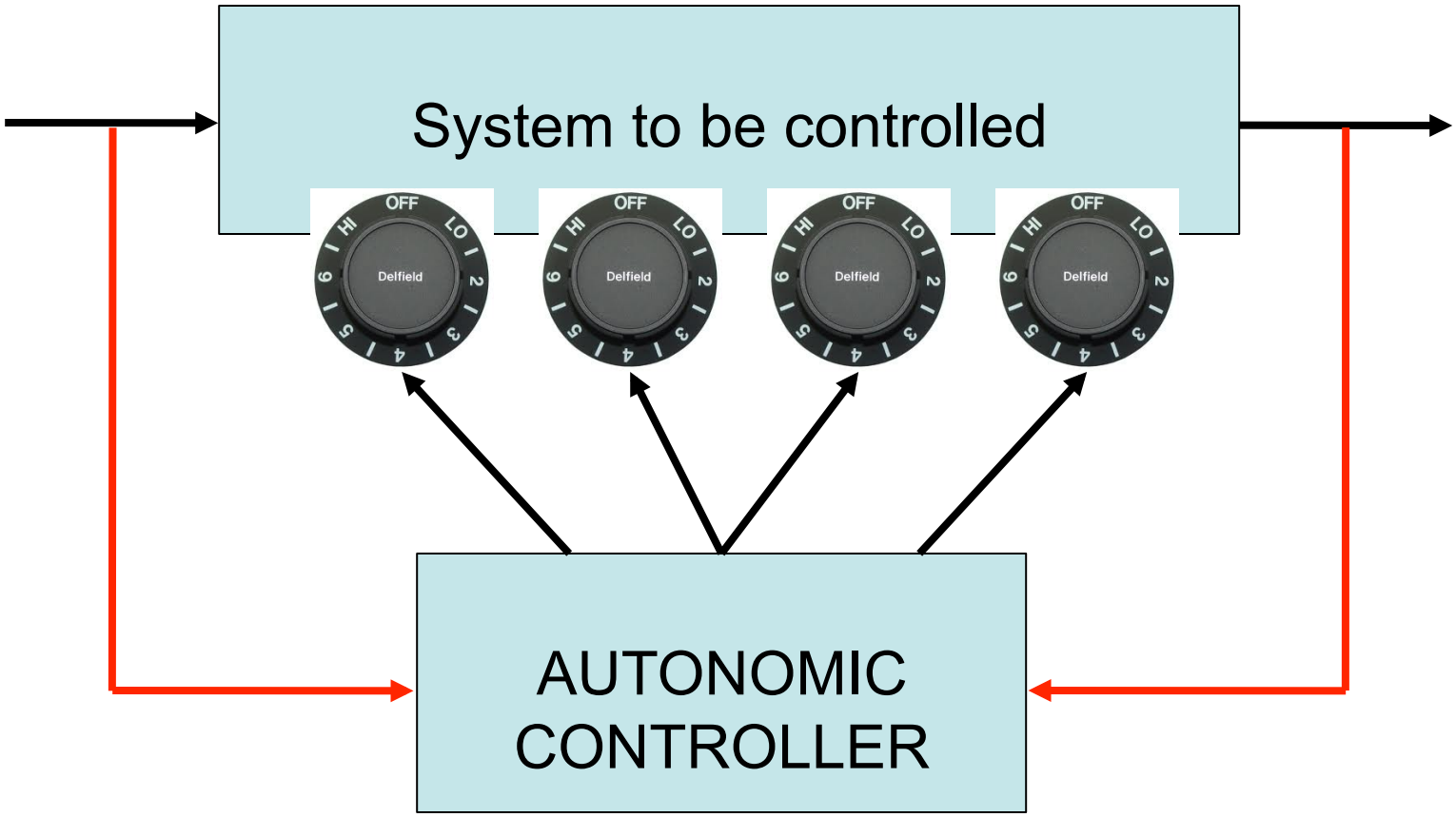
IBM's MAPE-K Model for AC



Autonomic Controller

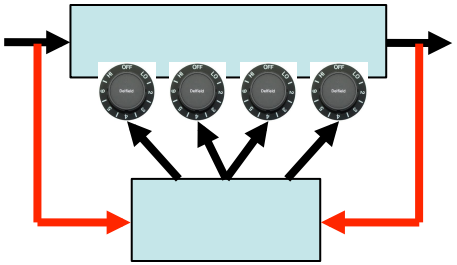


Autonomic Controller



How does the AC know the output of the system for a given combination of the knobs?

Autonomic Controller

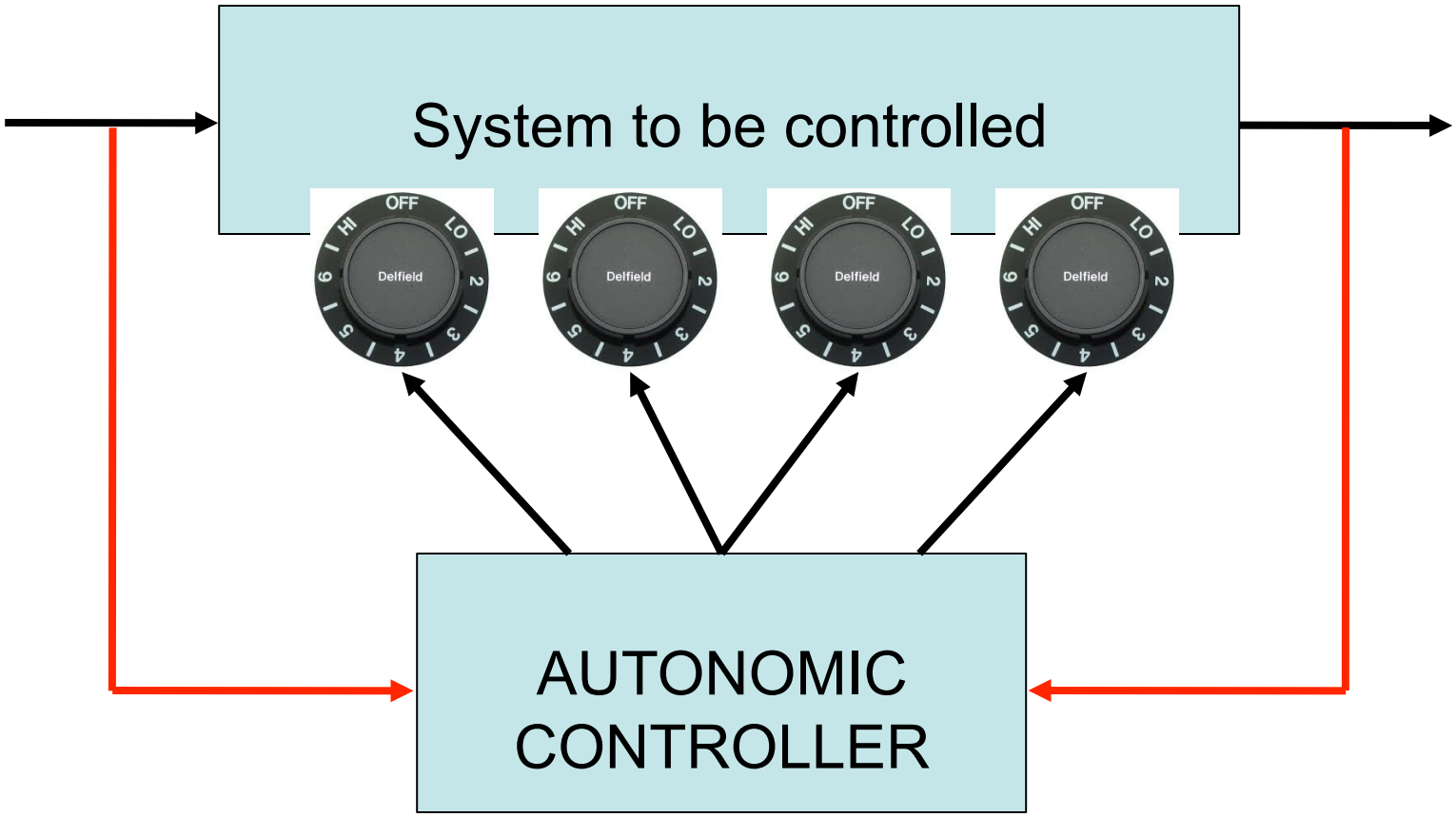


How does the AC know the output of the system for a given combination of the knobs?

$$S_{out} = f(k_1, k_2, \ldots, k_n, S_{input})$$

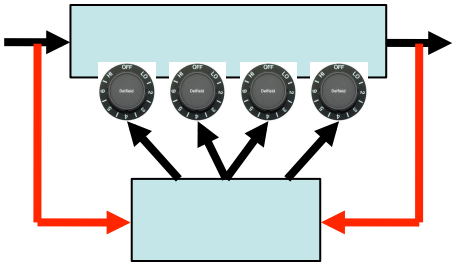
The function *f* can be obtained by a model or can be learned by the AC controller by observing system inputs and outputs.

Autonomic Controller



What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

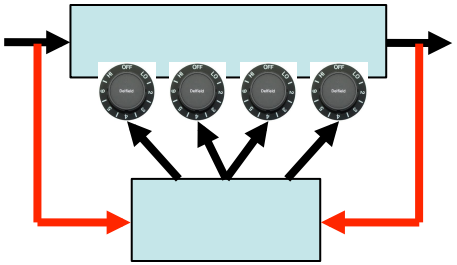
Autonomic Controller



What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

- The AC may want to maximize/minimize a performance metric:
  - Minimize response time
  - Maximize availability
  - Maximize throughput
  - Minimize energy consumption

Autonomic Controller

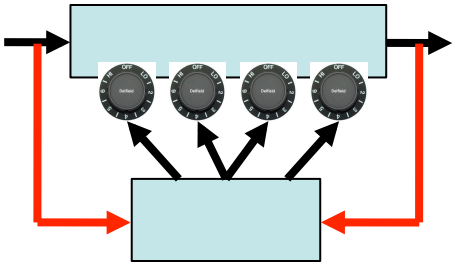


What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

Minimize ResponseTime = f (k1, …, kn)

Subject to
EnergyConsumed = g1 (k1, …, kn) ≤ MaxEnergy
Throughput = g2 (k1, …, kn) ≥ MinThroughput
Availability = g3 (k1, …, kn) ≥ MinAvailability

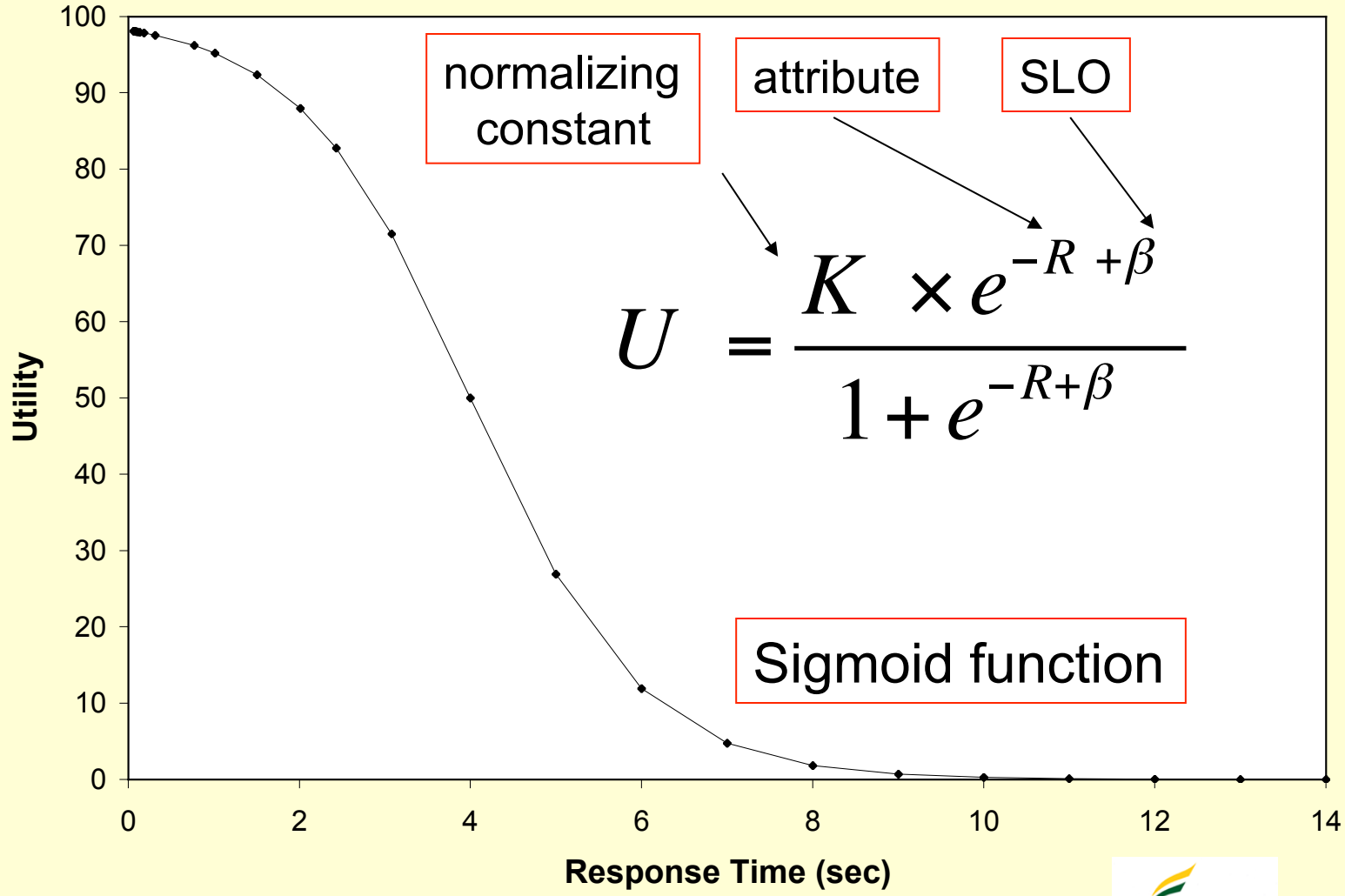
Utility Functions and the AC



What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

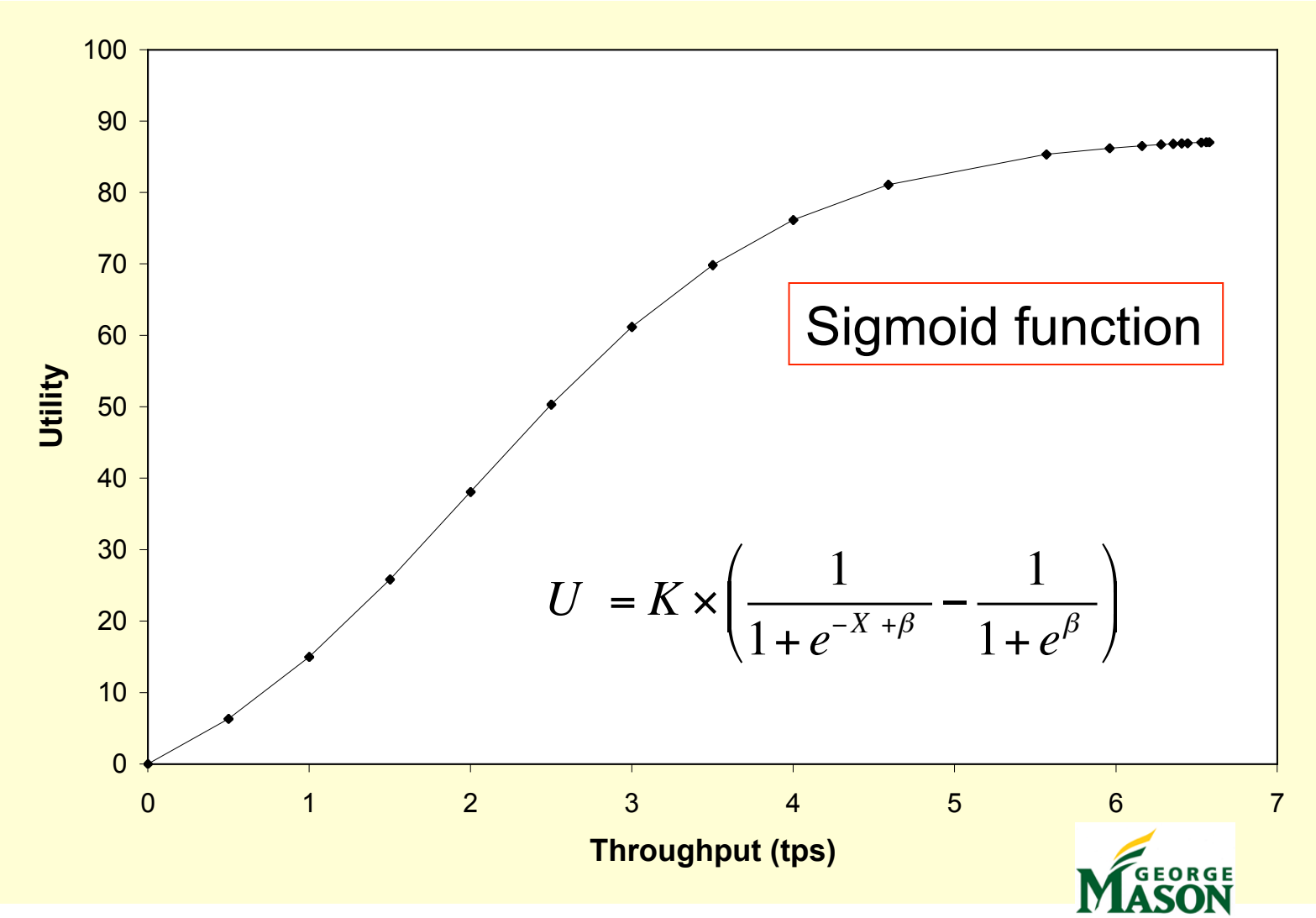
- The AC may want to consider trade-offs between performance metrics.
- Use utility function.
  - A utility function of an attribute ***a*** indicates the usefulness of a system as a function of the value of the attribute ***a***.

Utility Function as a Function of Response Time

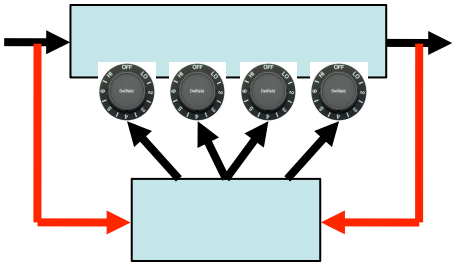


$$U = \frac{K \times e^{-R+\beta}}{1+e^{-R+\beta}}$$

Utility Function as a Function of Throughput



Utility Functions and the AC



What if there is more than one relevant attribute?

- Specify a global utility function that is a function of the utility functions of each attribute:

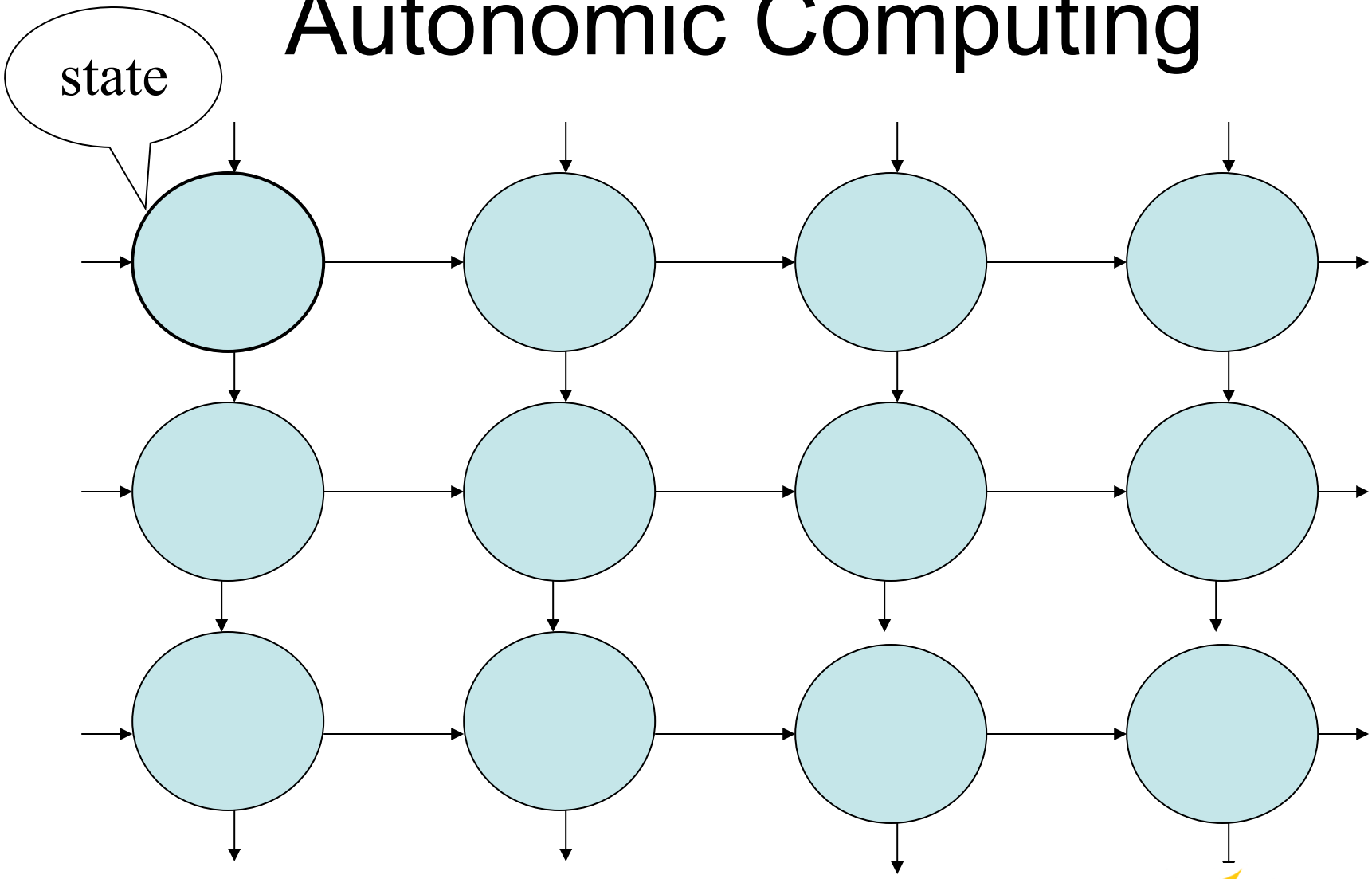
$$U_{global} = f(U_1(a_1),\ldots,U_n(a_n))$$

e.g.,

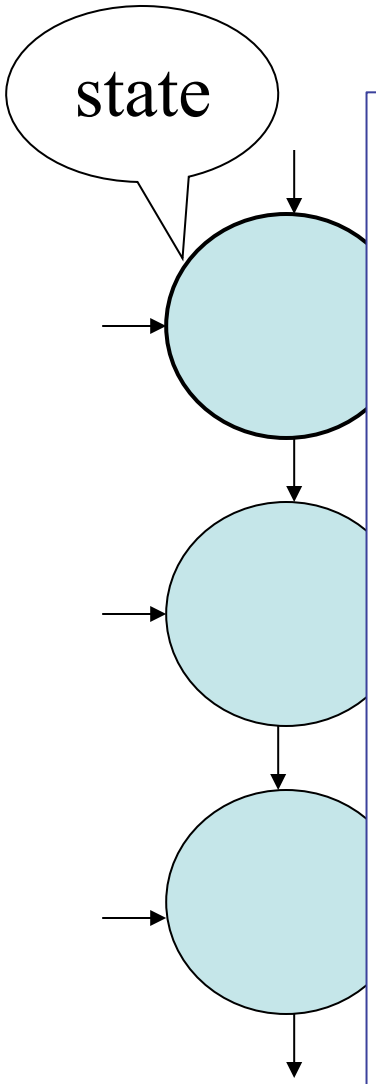
$$U_{global} = w_r U_r(R) + w_x U_x(X) + w_a U_a(a)$$

$$w_r + w_x + w_a = 1$$

Performance Model-Based Autonomic Computing



Performance Model-Based Autonomic Computing



State: e.g., set of configuration parameters

Value: e.g., QoS metric, utility function value

Goal: find state that optimizes the value subject to constraints

- State space is typically large
- Objective function does not have a closed form

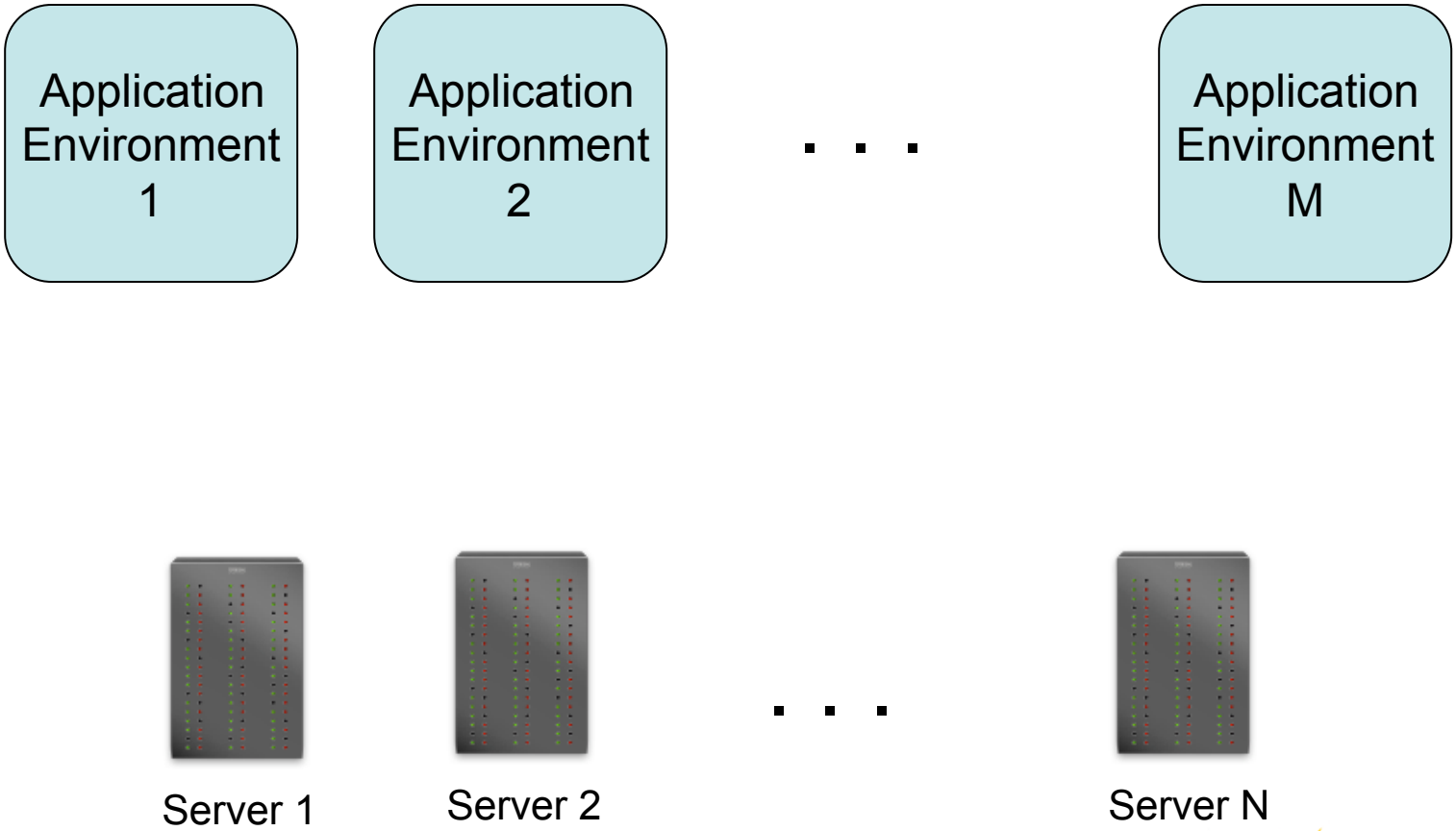
Use performance models to compute value at each state.

Use combinatorial search techniques to find near-optimal solution.

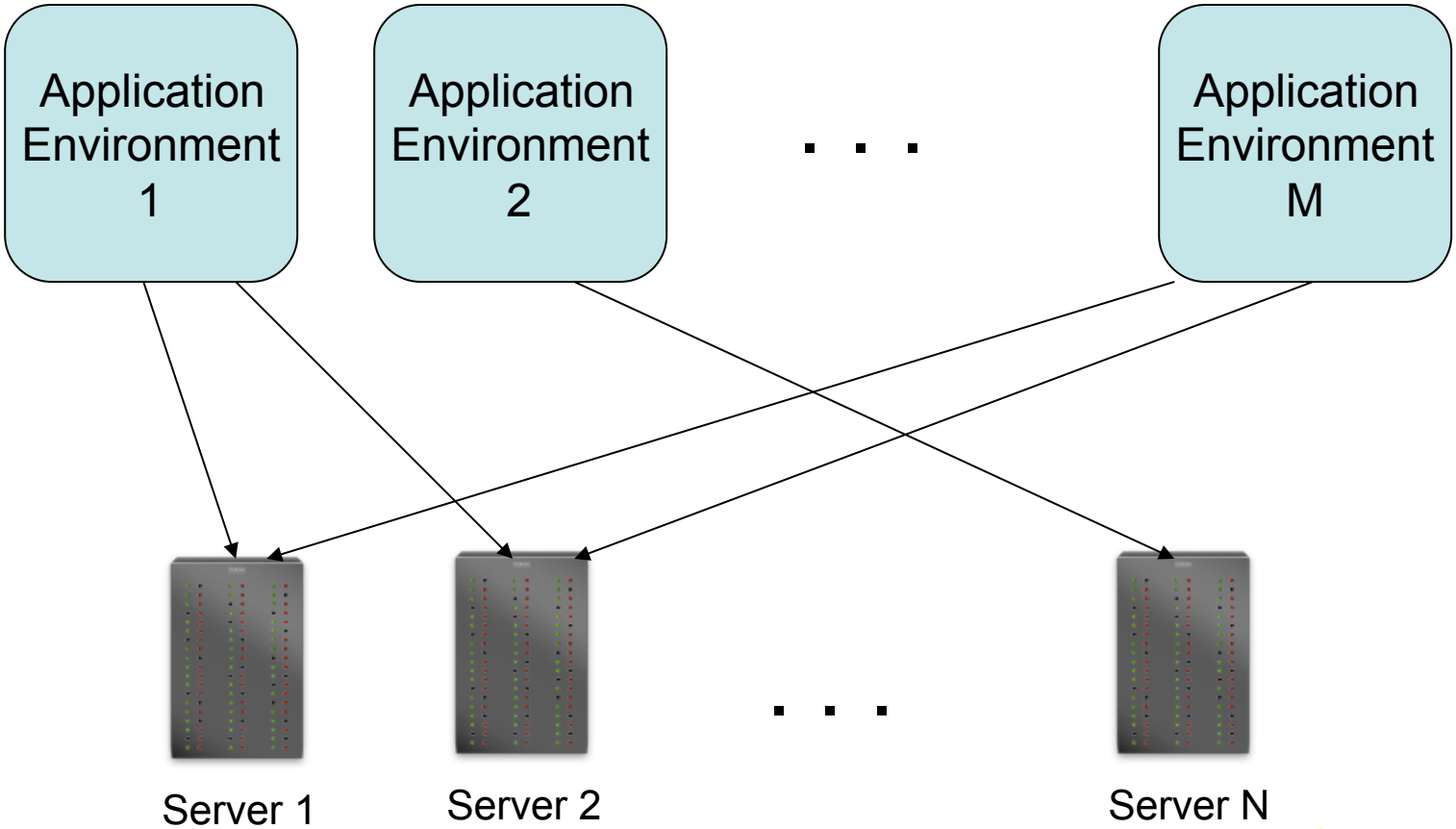
Outline

- Motivation for Autonomic Computing
- Techniques used in AC
  - Model-driven
    - Performance model
    - Control theory
  - Model-less
    - Machine learning
    - Statistical learning
- Applications
  - Internet data centers, virtual machine management, e-commerce and Web-systems, service oriented computing, cloud computing resource management, databases, adaptive software systems, and emergency departments
- Concluding Remarks

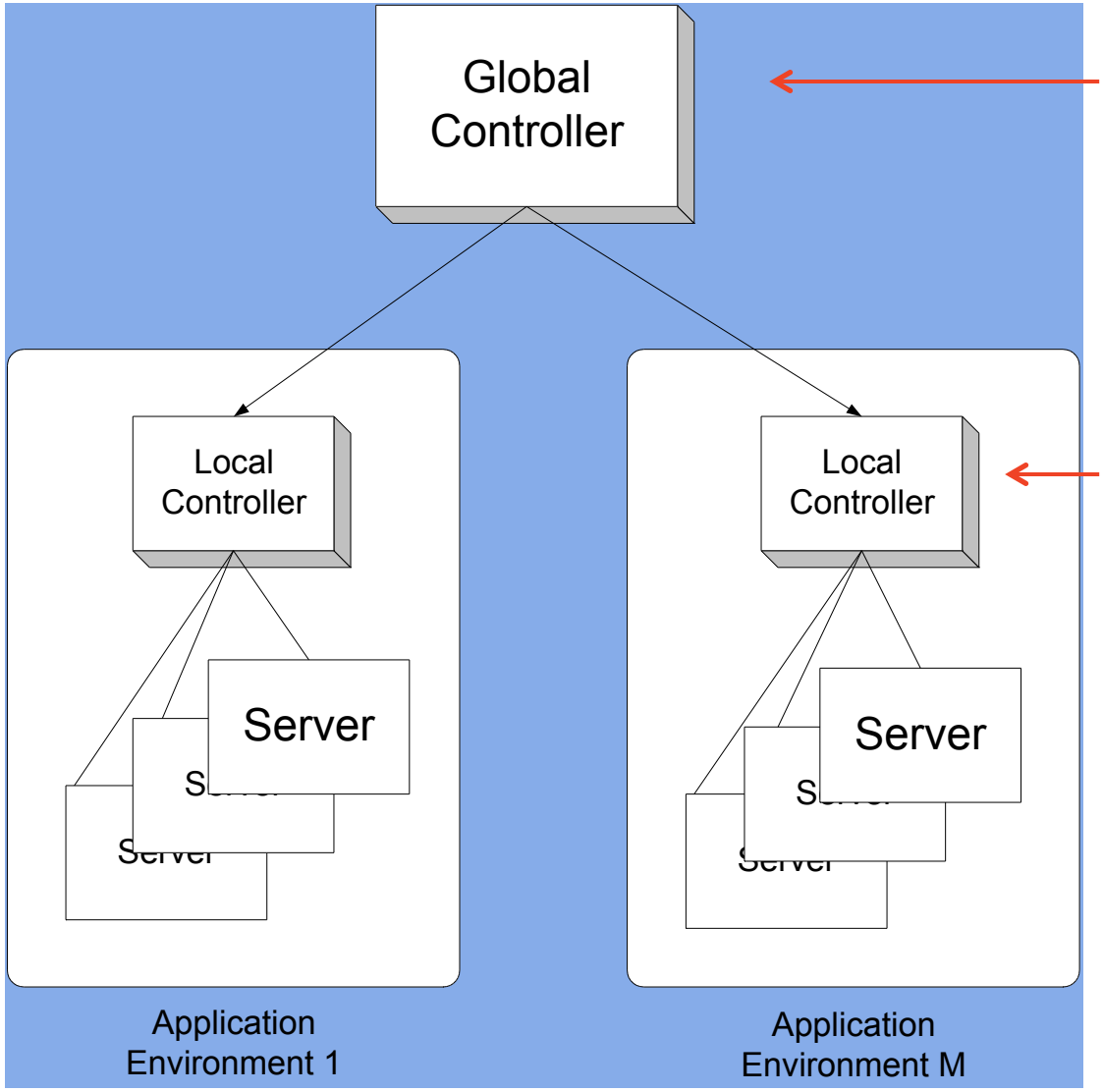
Dynamic Resource Allocation in Internet Data Centers



Dynamic Resource Allocation Problem



Dynamic Resource Allocation
Two-level Controllers



Decides how many servers to assign to each AE.

Implements Global Controller's decisions.

Dynamic Resource Allocation Utility Function

- The global controller uses a global utility function, Ug, to assess the adherence of the overall data center performance to desired service levels objectives (SLOs)

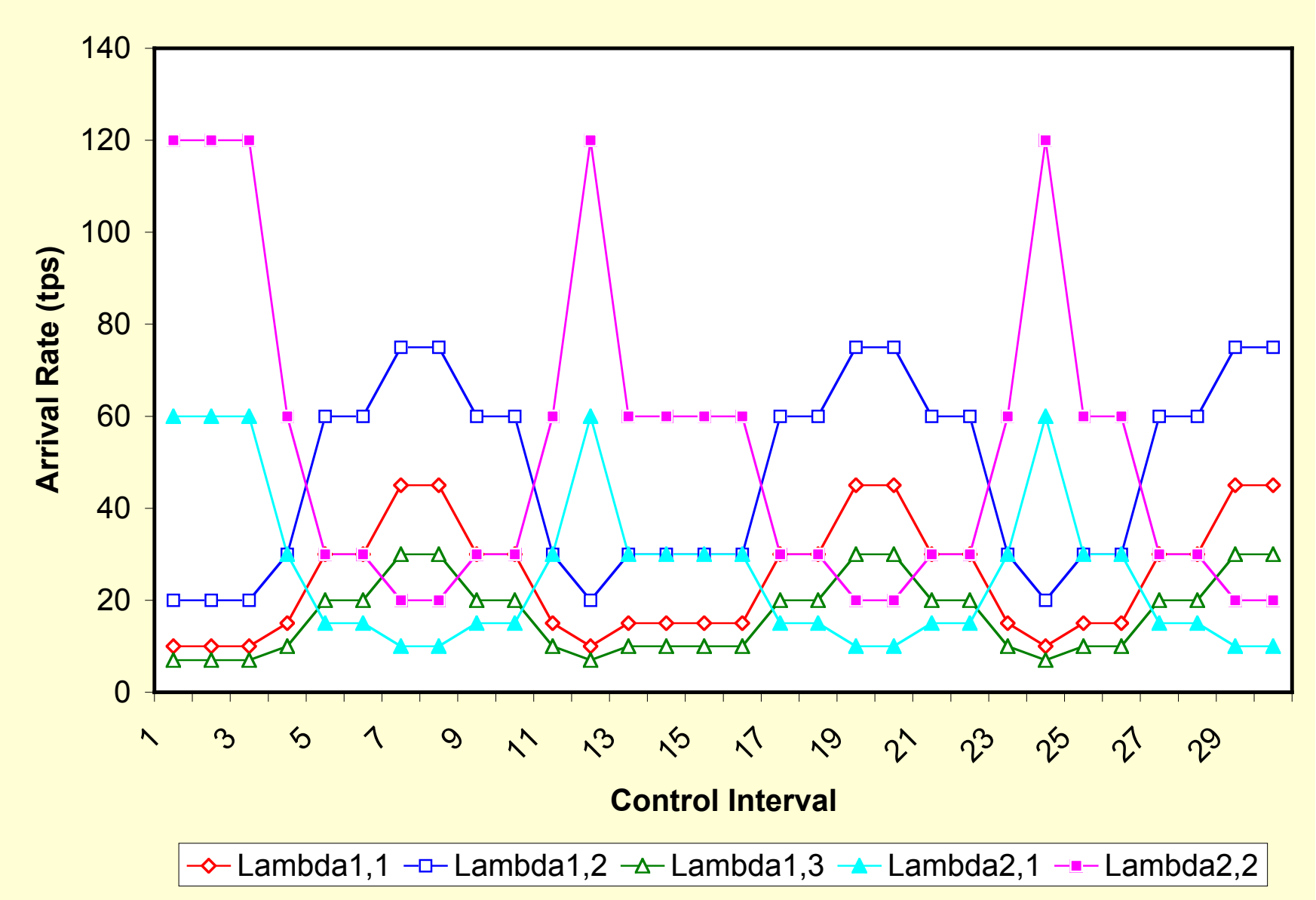
$$U_g = h(U_1, \ldots, U_M)$$

$$U_i = \sum_{s=1}^{S_i} a_{i,s} \times U_{i,s}$$

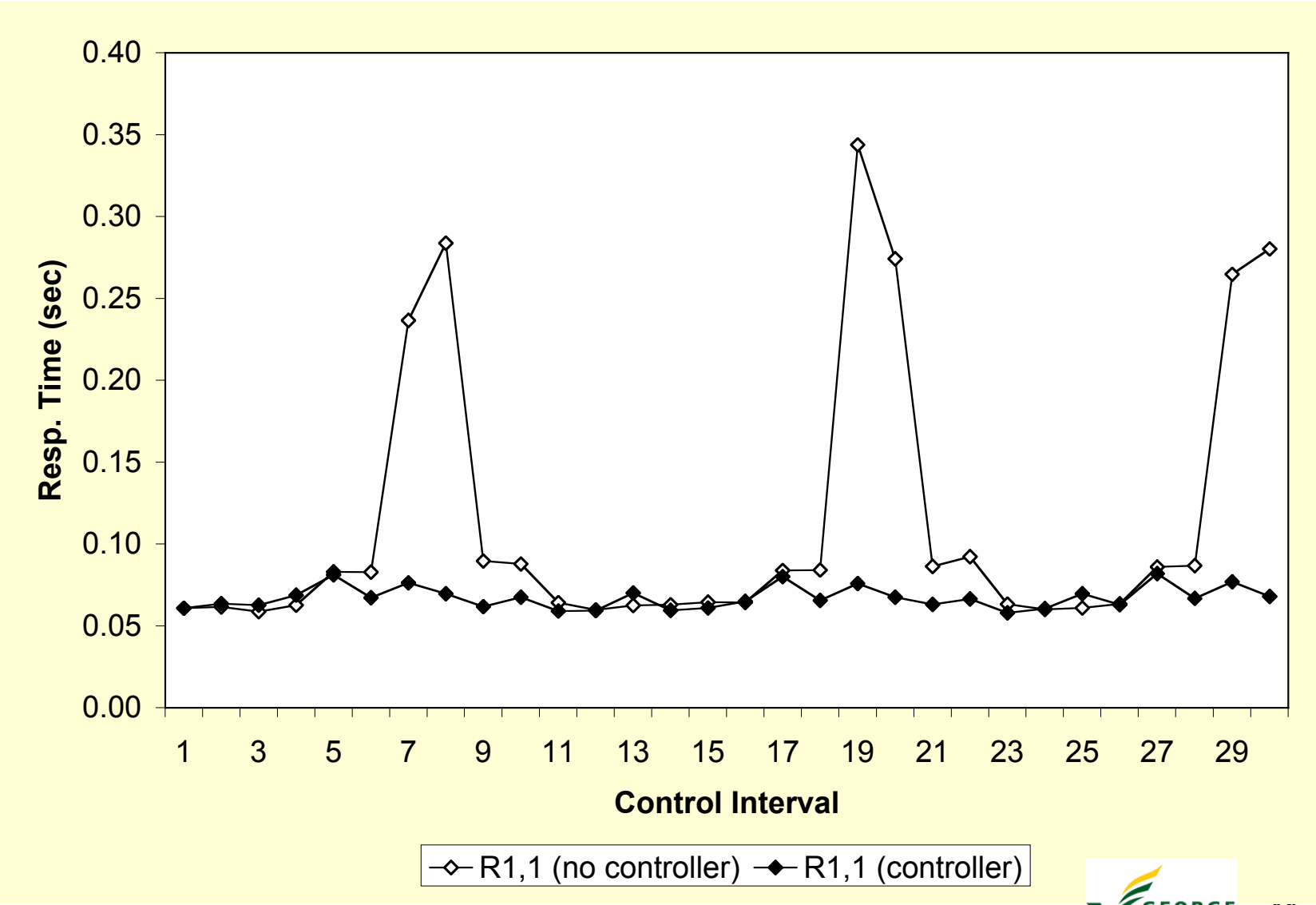
$$0 < a_{i,s} < 1$$

$$\sum_{s=1}^{S_i} a_{i,s} = 1$$

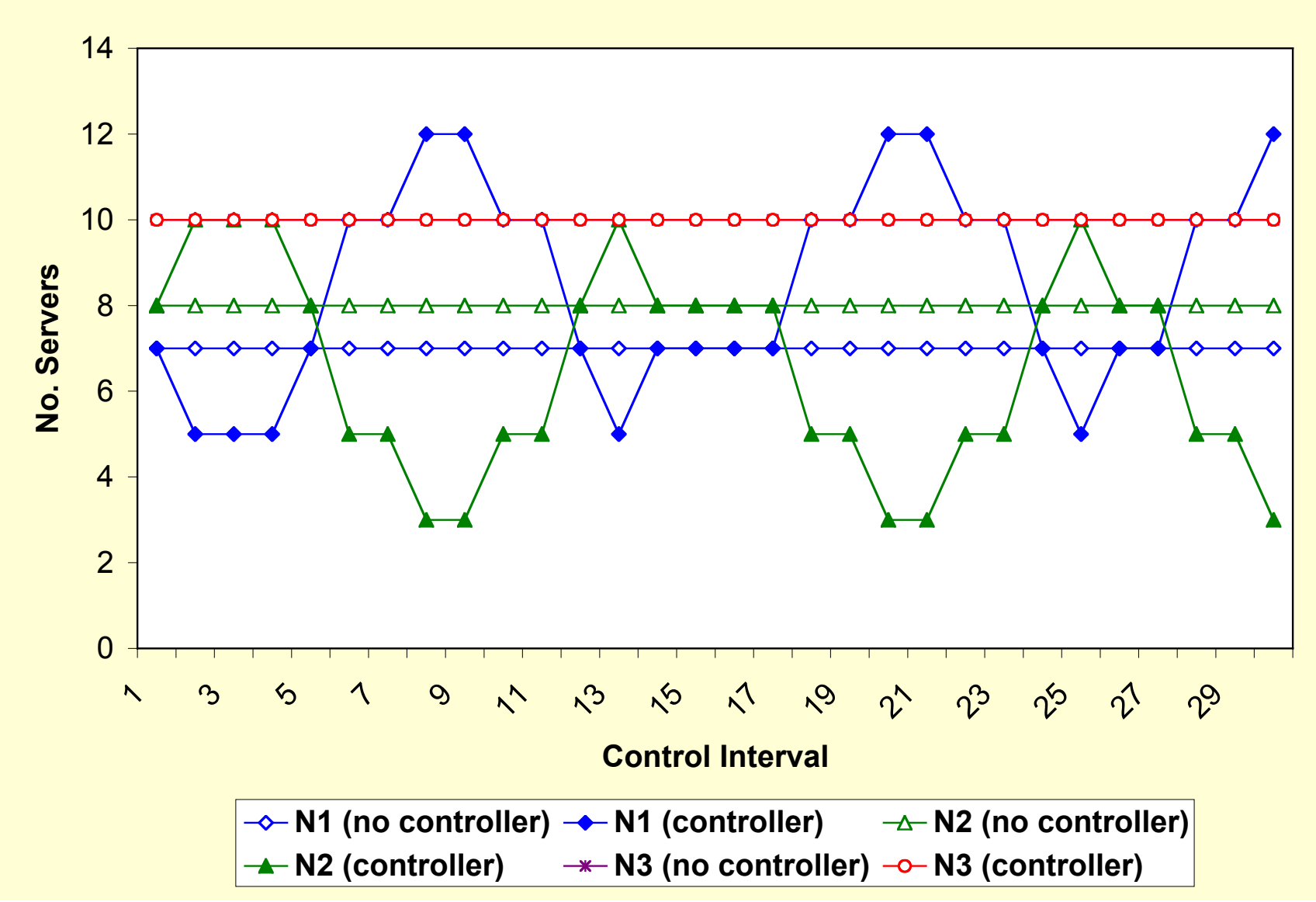
Workload Variation for Online AEs



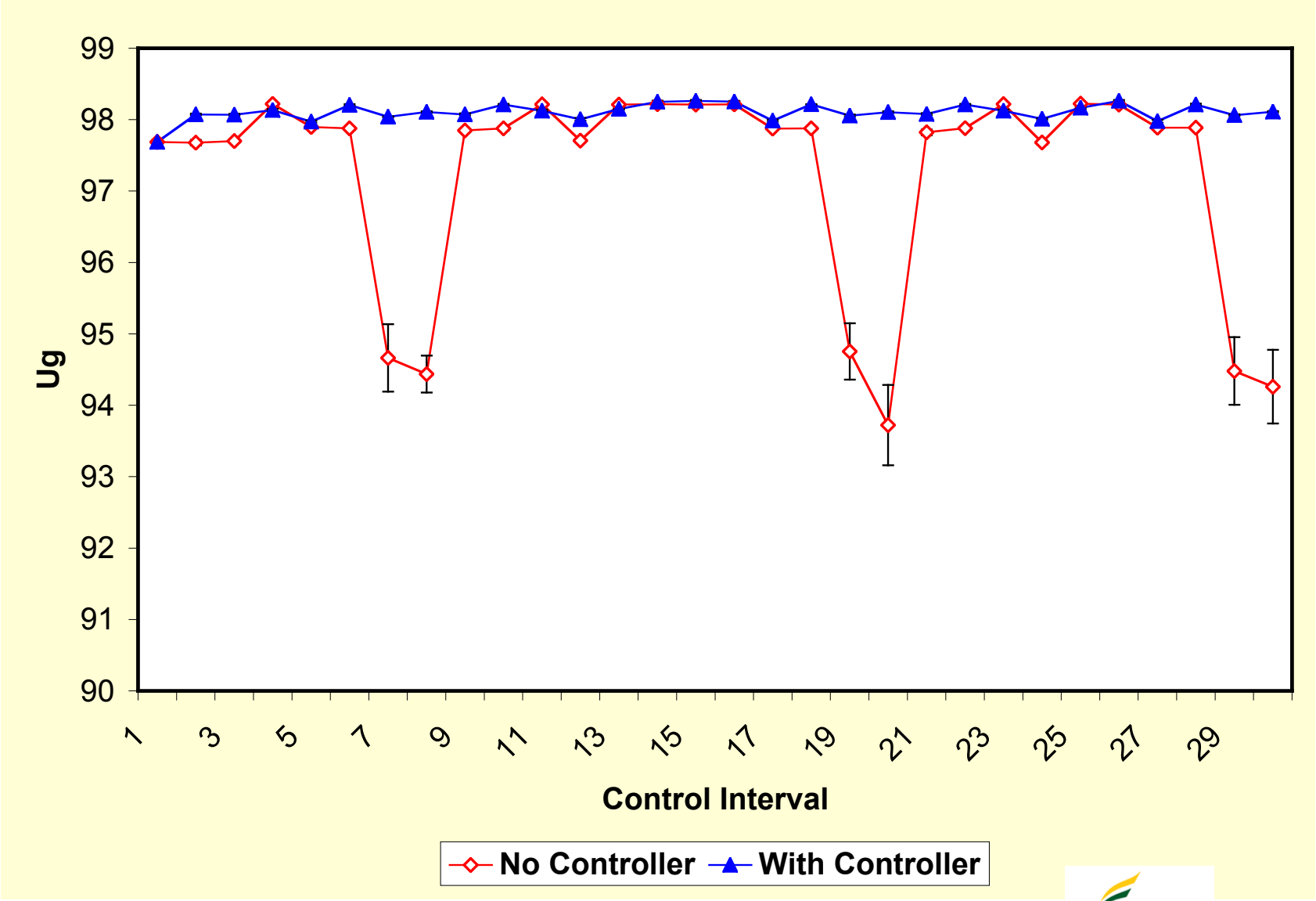
Response Times for Class 1 of AE 1



Variation of the Number of Servers



Variation of Global Utility



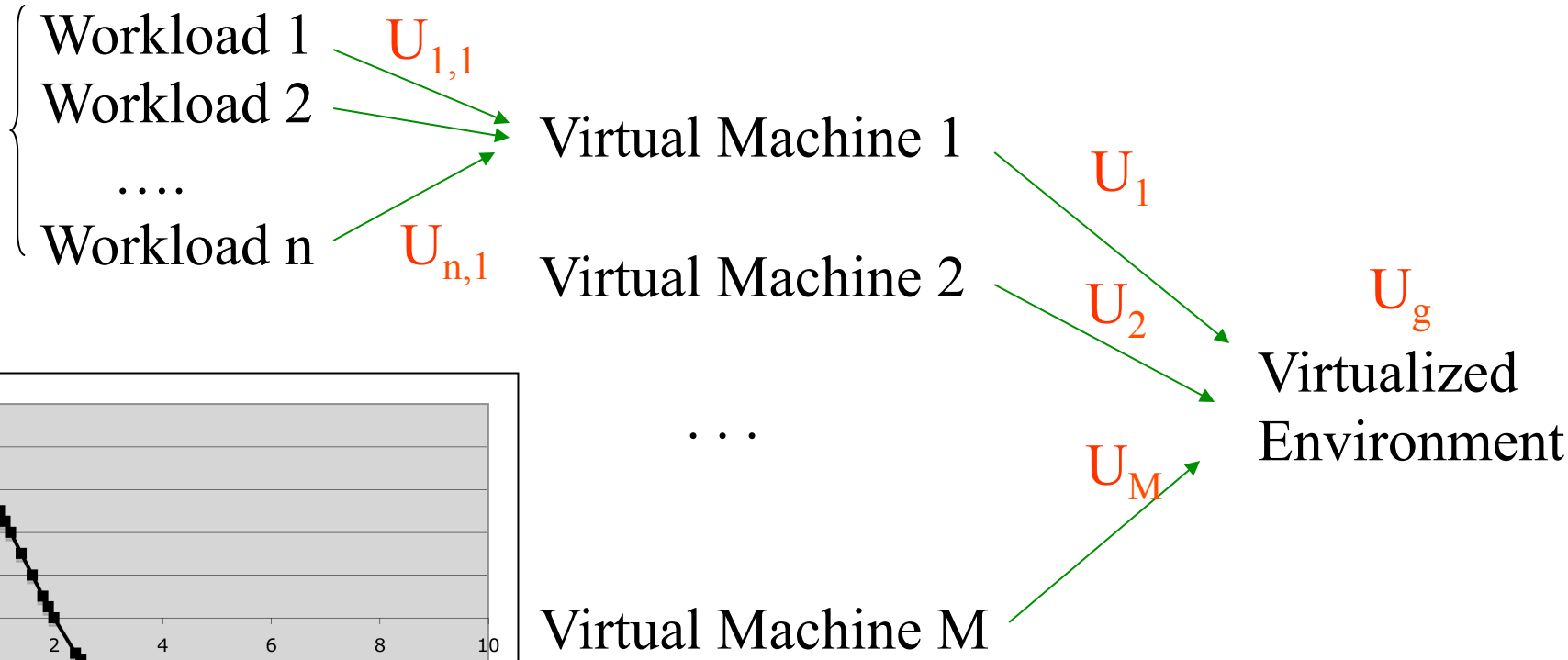
Outline

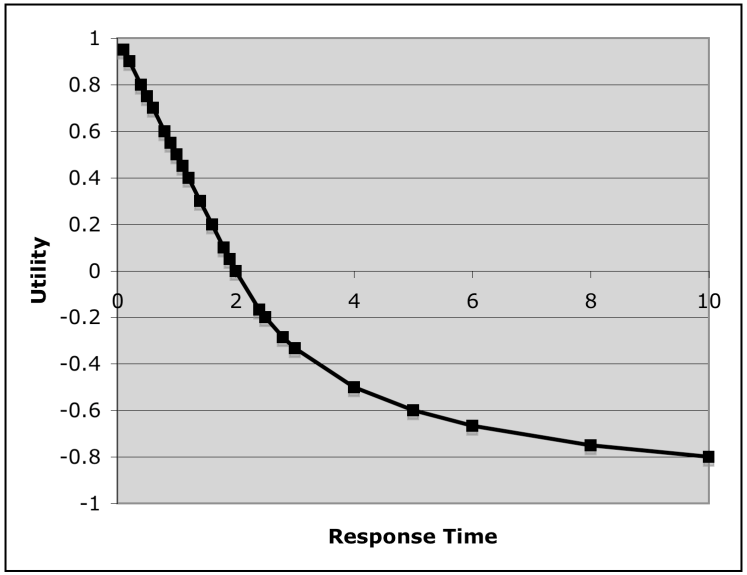
- Motivation for Autonomic Computing
- Techniques used in AC
  - Model-driven
    - Performance model
    - Control theory
  - Model-less
    - Machine learning
    - Statistical learning
- Applications
  - Internet data centers, virtual machine management, e-commerce and Web-systems, service oriented computing, cloud computing resource management, databases, adaptive software systems, and emergency departments
- Concluding Remarks

CPU Allocation Problem for Autonomic Virtualized Environments

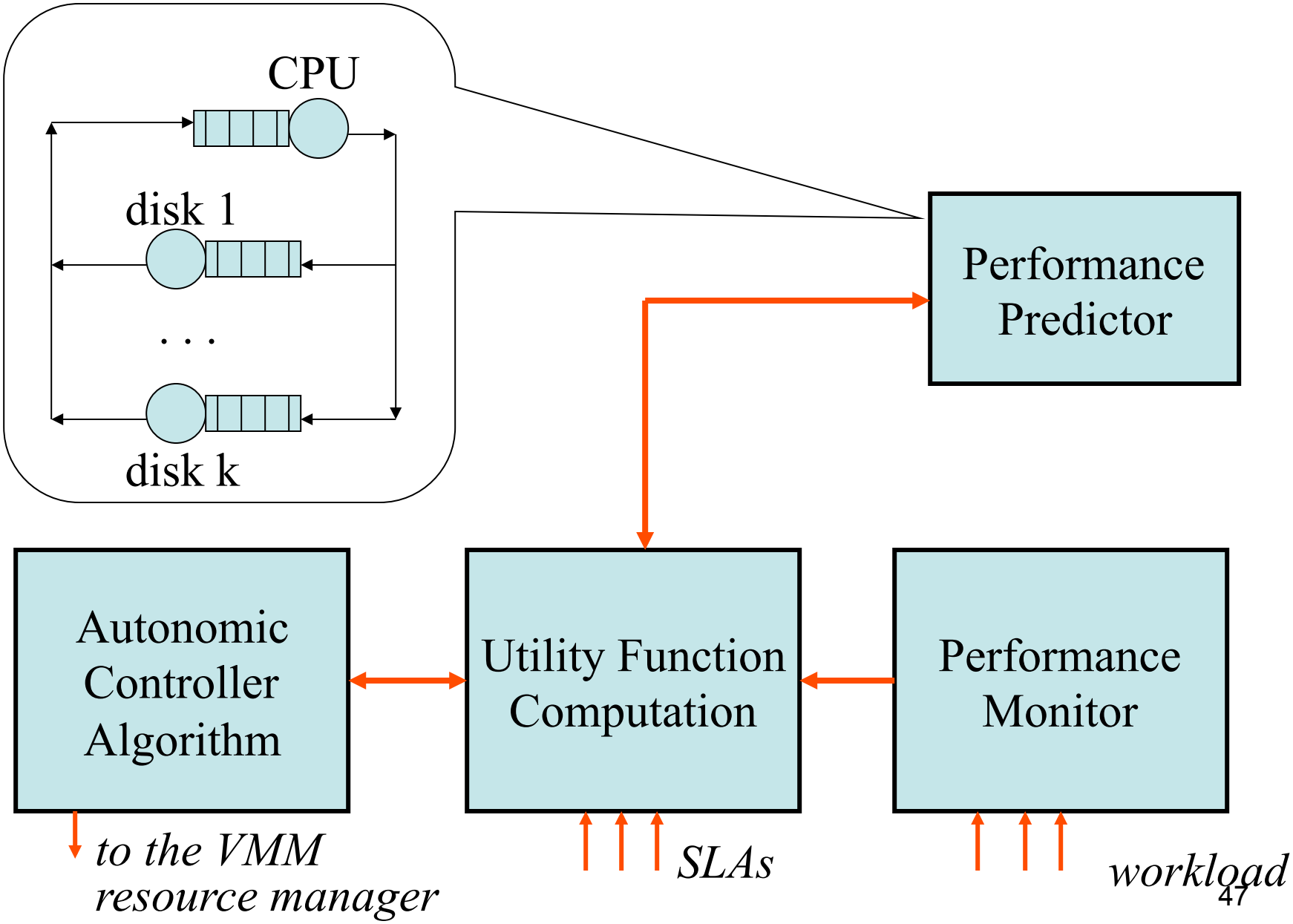
- Existing systems allow for manual allocation of CPU resources to VMs using *CPU priorities* or *CPU shares*.
- Need automated mechanism for the adjustment of CPU shares of the virtual machines in order to maximize the global utility of the entire virtualized environment

CPU Allocation Problem for Autonomic Virtualized Environments (Cont'd)

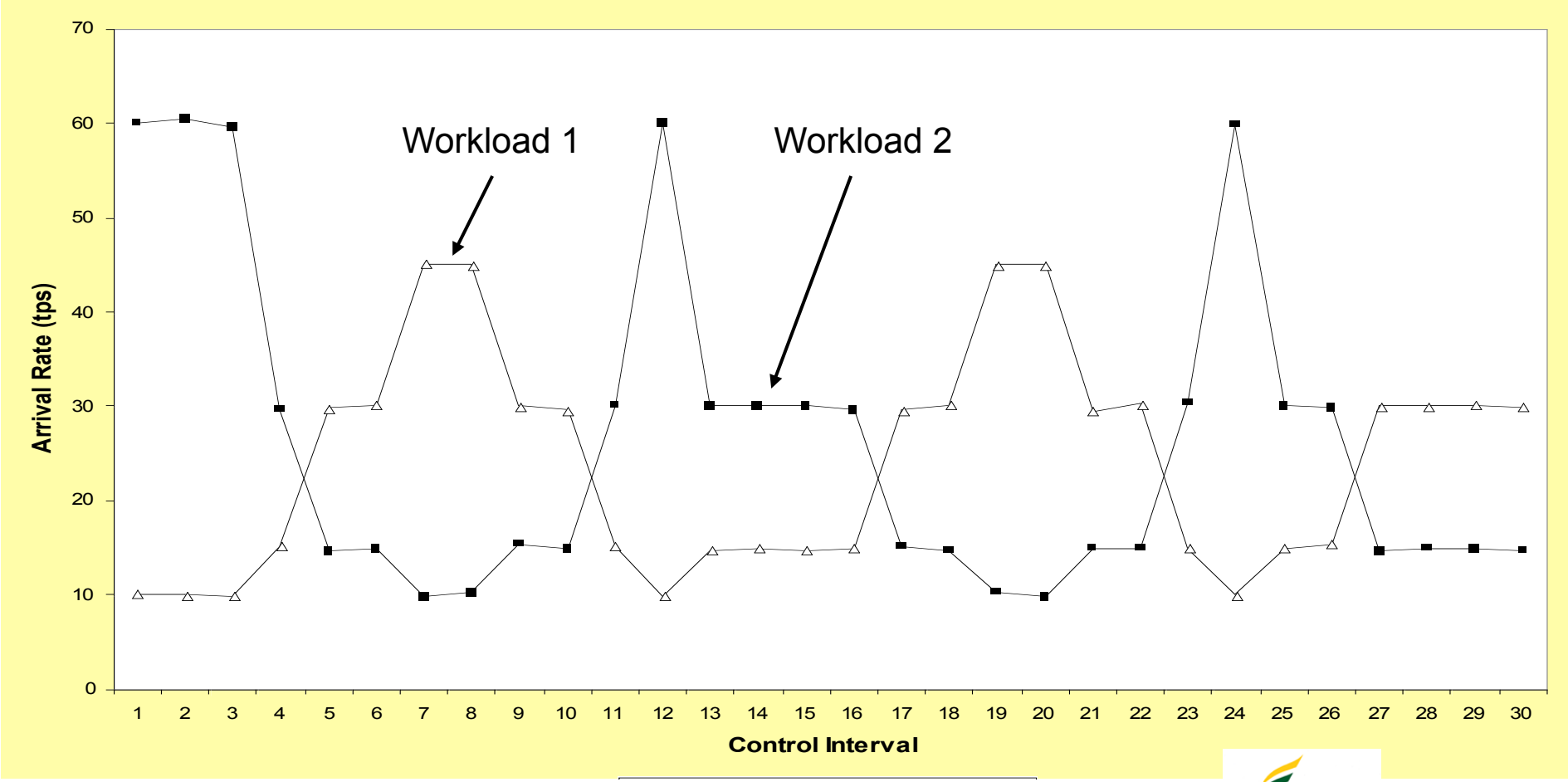




Virtualization Controller Architecture

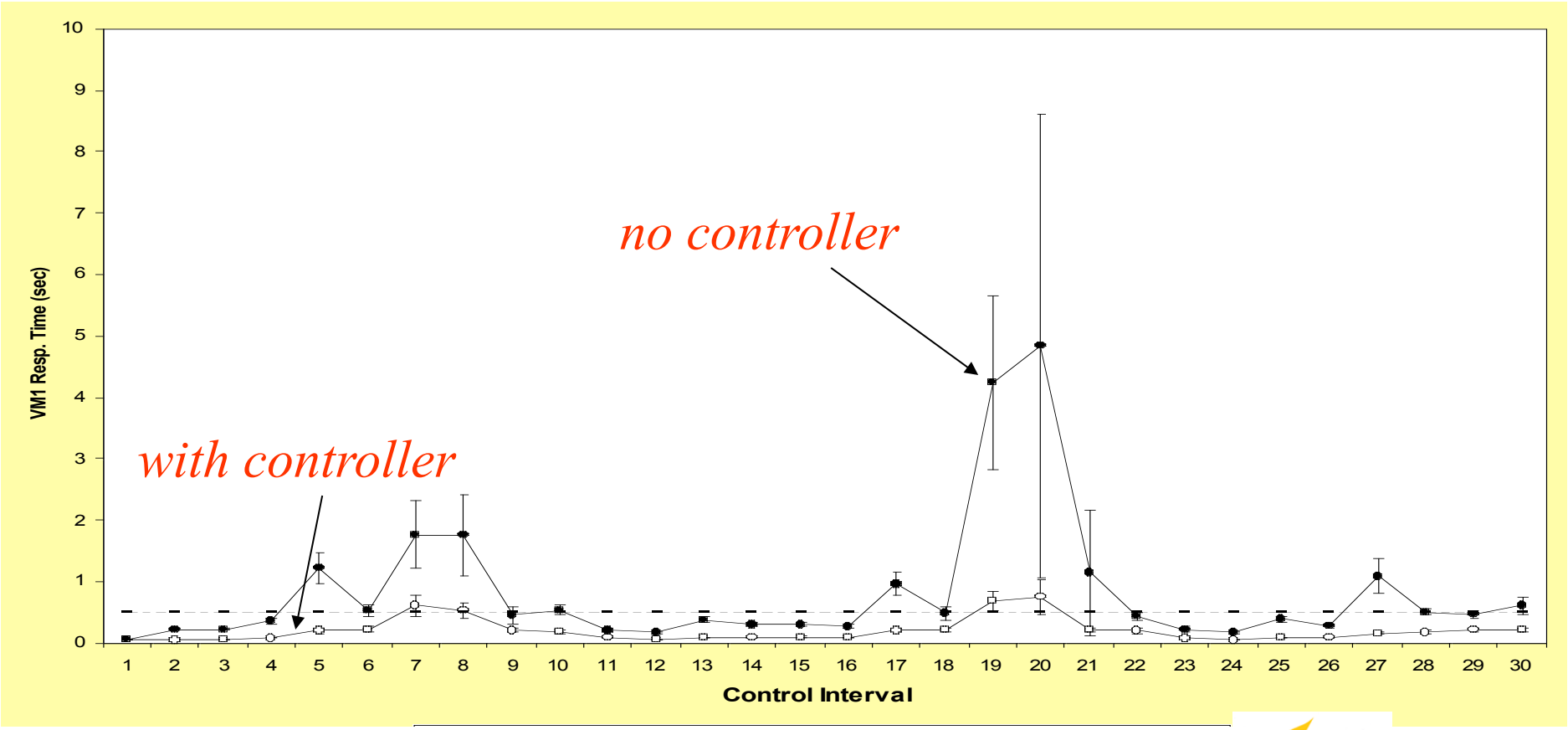


CPU Shares Based Allocation: Workload Variation

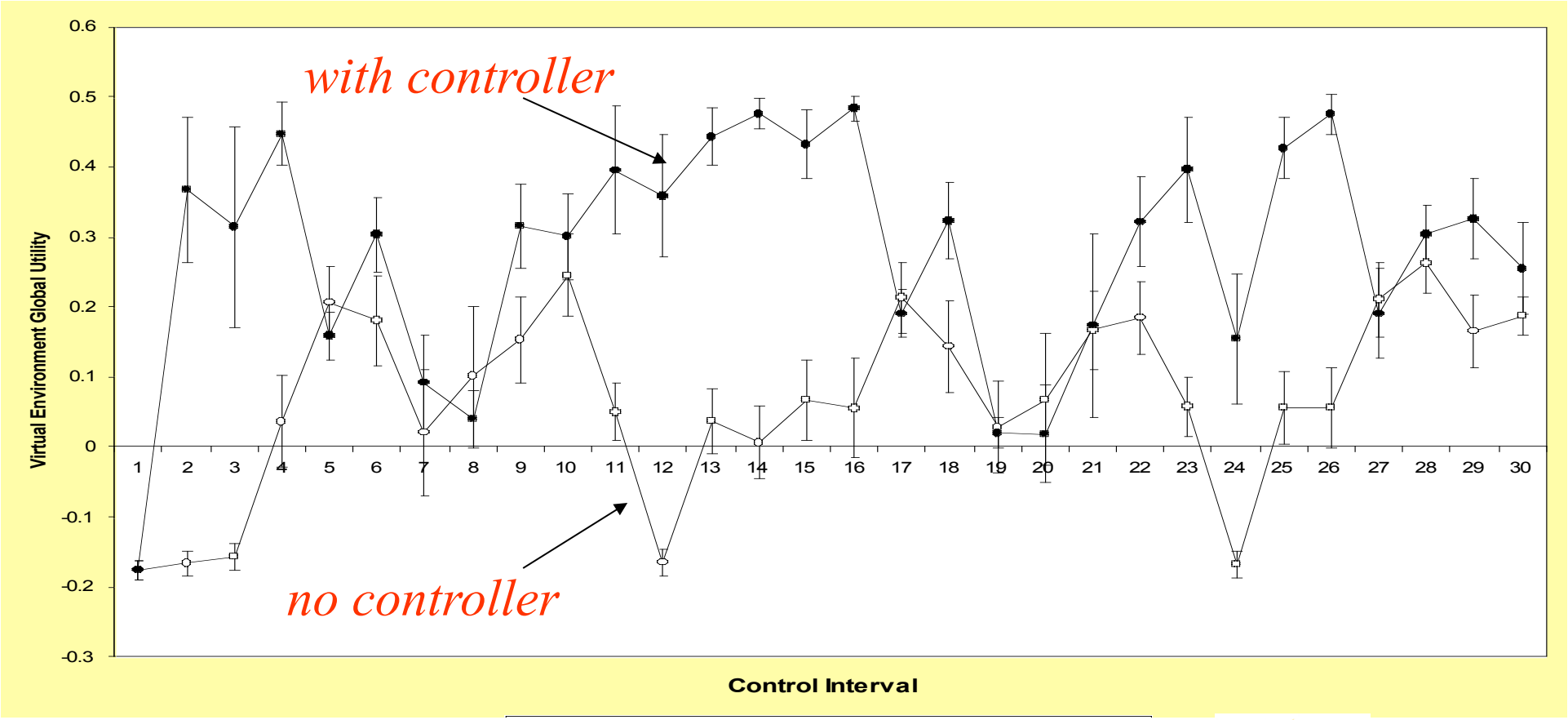


CPU Shares Based Allocation: CPU Shares Variation

CPU Shares Based Allocation: Response Time for VM1



CPU Shares Based Allocation: Global Utility



Outline

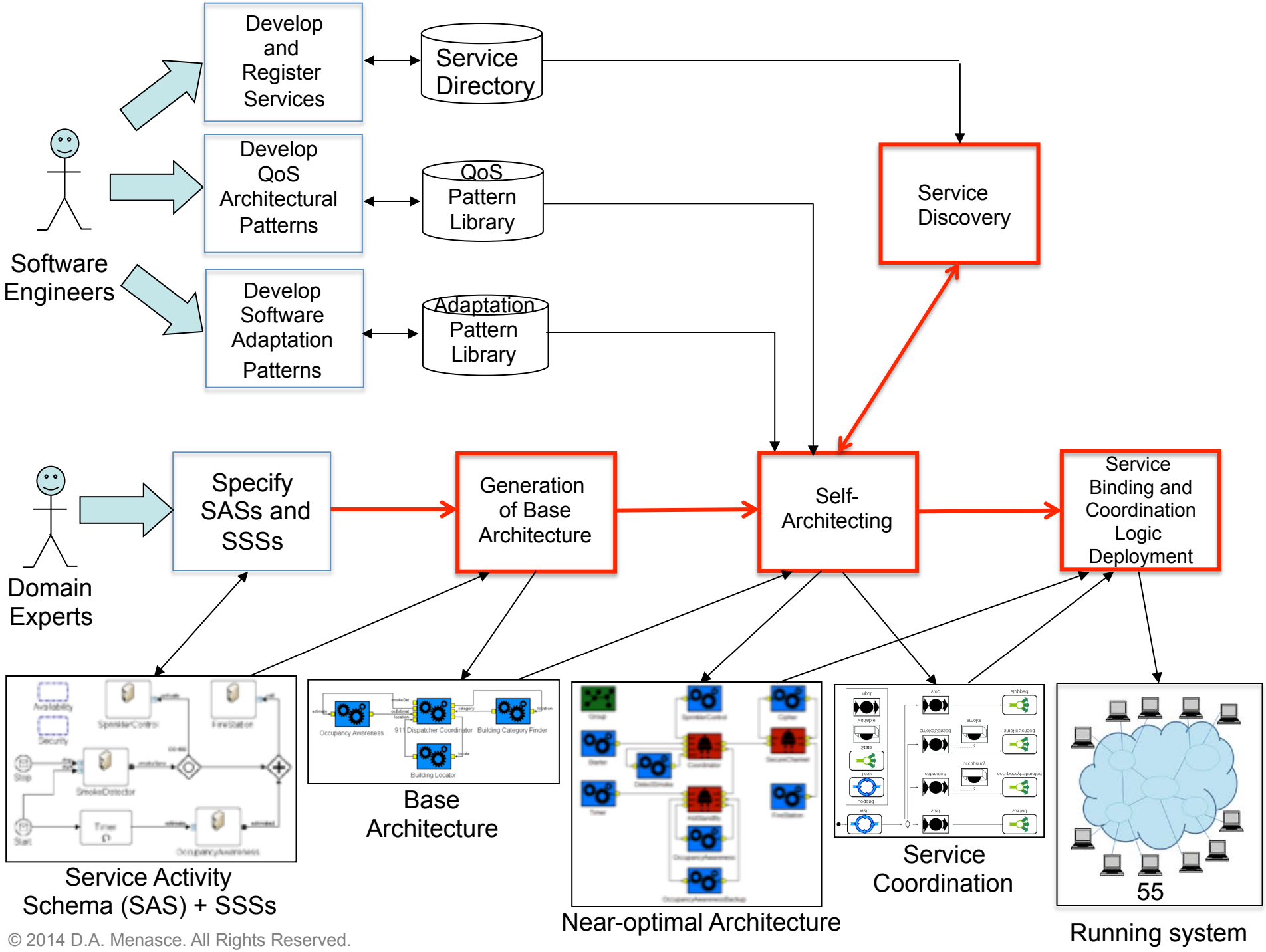
- Motivation for Autonomic Computing
- Techniques used in AC
  - Model-driven
    - Performance model
    - Control theory
  - Model-less
    - Machine learning
    - Statistical learning
- Applications
  - Internet data centers, virtual machine management, e-commerce and Web-systems, service oriented computing, cloud computing resource management, databases, adaptive software systems, and emergency departments
- Concluding Remarks

Outline

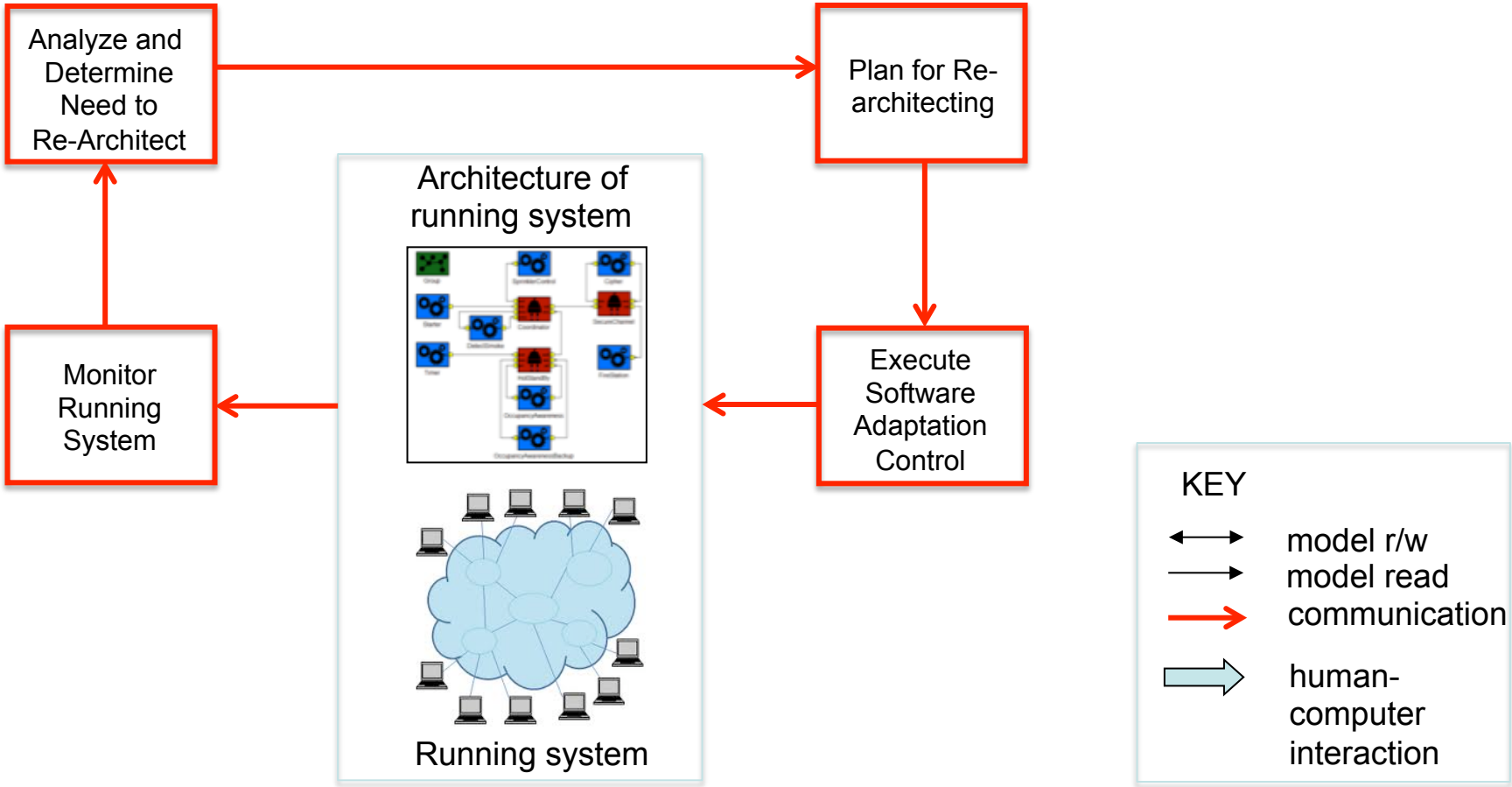
- Motivation for Autonomic Computing
- Techniques used in AC
  - Model-driven
    - Performance model
    - Control theory
  - Model-free
    - Machine learning (e.g., reinforcement learning)
    - Statistical learning
- Applications
  - Internet data centers, virtual machine management, e-commerce and Web-systems, service oriented computing, cloud computing resource management, databases, adaptive software systems, and emergency departments
- Concluding Remarks

Self-Architecting SOA Software Systems (SASSY)

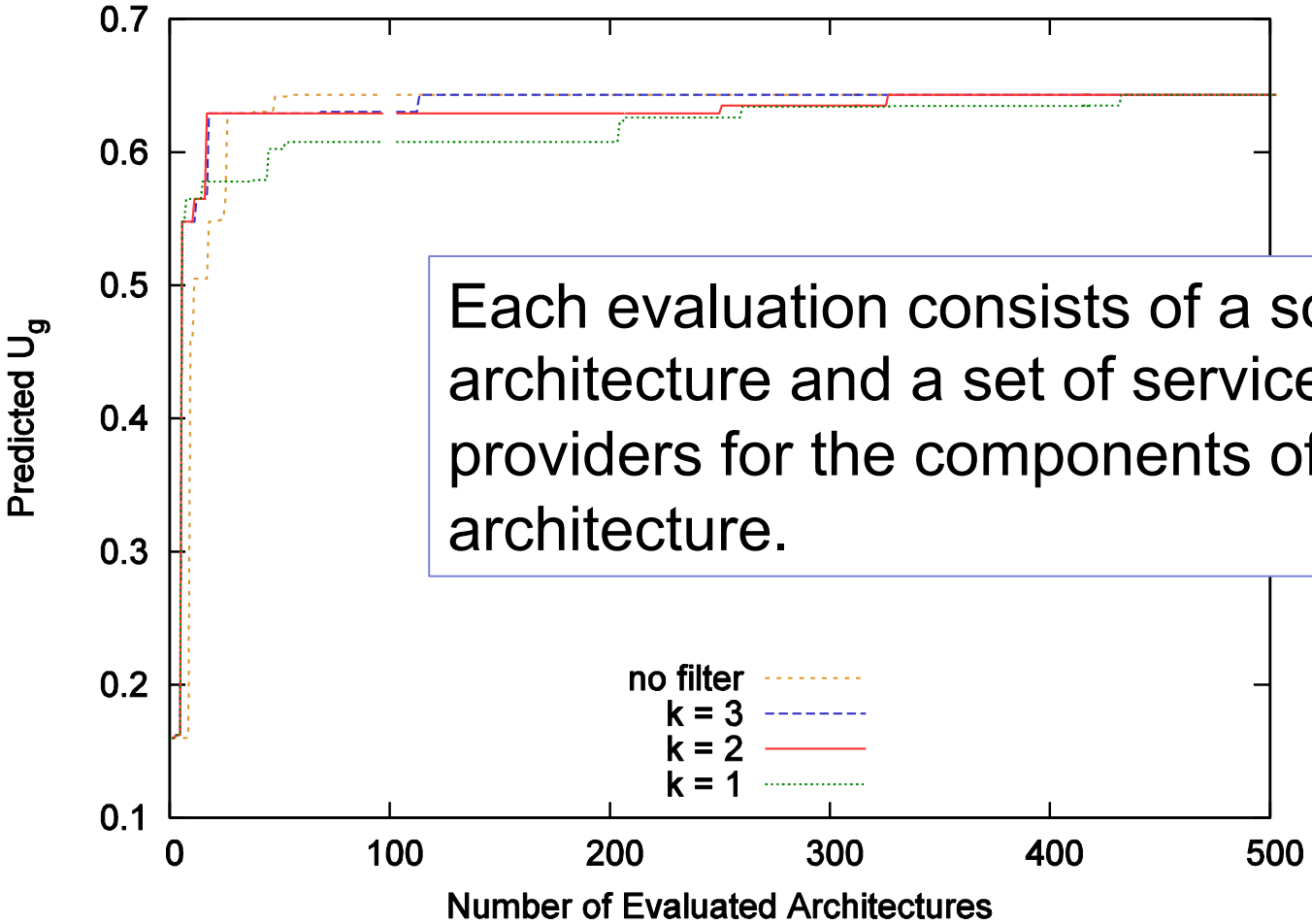
- SASSY allows domain experts to specify requirements using a high-level visual activity language.
- SASSY generates the initial software architecture optimized to maximize a utility function.
- The running system is constantly monitored and SASSY automatically re-architects the system when needed.



SASSY: Run-time Adaptation



Architecture Adaptation Search Trajectories



Each evaluation consists of a software architecture and a set of service providers for the components of the architecture.

Concluding Remarks

- Autonomic computing is a key design discipline for large, complex, and dynamic computer systems
- AC uses models (queuing and/or control models) and optimization techniques (exact and/or approximate).
- AC may learn models of the controlled system.
- Utility functions useful in dealing with tradeoffs.

CREDITS

- Arwa Aldhalaan
- Serene Al-Momen
- Mahmoud Awad
- Firas Alomari
- Daniel Barbará
- Shouvik Bardhan
- Mohamed Bennani
- Alex Brodsky
- Ernesto Casalicchio
- Ronald Dodge
- Vinod Dubey
- Naeem Esfahani
- John Ewing
- Hassan Gomaa
- Gus Jabbour
- Mohan Krishnamoorthy
- Sam Malek
- Faisal Sibai
- João P. Sousa

NOT COVERED TODAY

- Emergency Departments
- Smart Manufacturing
- Energy Management
- Dynamic allocation of services in SOA architectures
- Cluster support for Big Data
- Load-balancing policies
- Computer networks
- Cloud computing
- Databases