On the Use of Performance Models to Design Self-Managing Computer Systems

Daniel A. Menascé

and

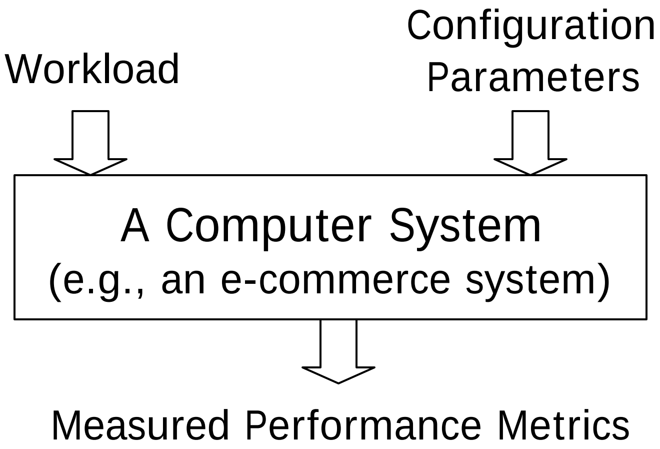
Mohamed Bennani

Dept. of Computer Science

George Mason University

www.cs.gmu.edu/faculty/menasce.html

A Computer System



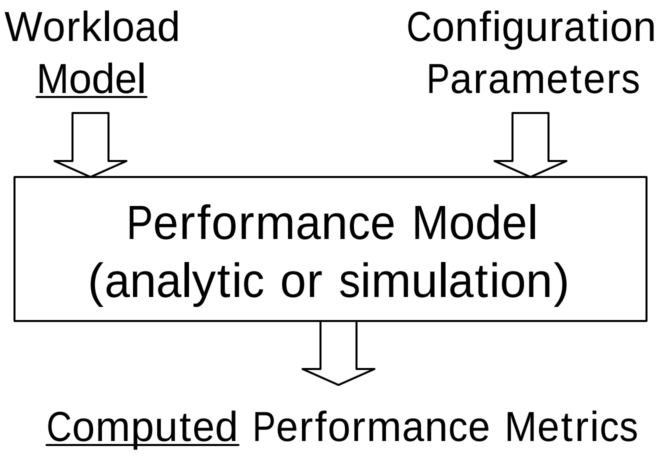
Workload

Configuration Parameters

A Computer System
(e.g., an e-commerce system)

Measured Performance Metrics

A Performance Model



Workload Model

Configuration Parameters

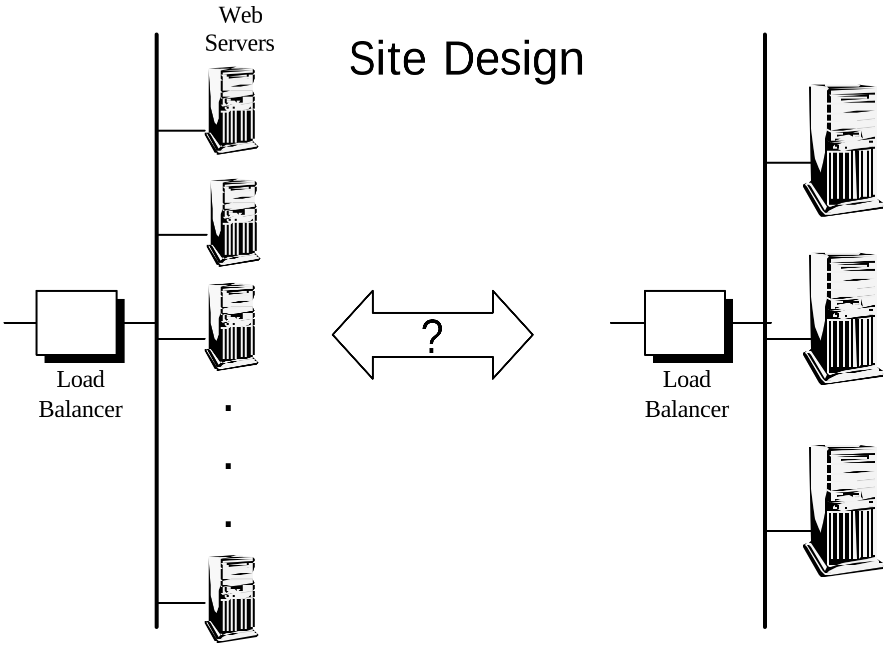
Performance Model (analytic or simulation)

Computed Performance Metrics

What are performance models good for?

- At the design stage:
  - Compare competing design alternatives.
    - A large number of low capacity servers vs. a small number of large capacity servers?

Site Design



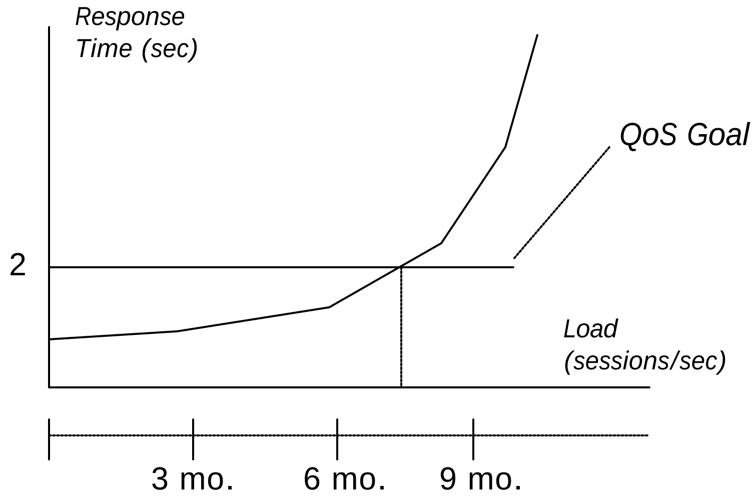
5

What are performance models good for?

- At the design stage:
  - Compare competing design alternatives.
    - A large number of low capacity servers vs. a small number of large capacity servers?
- During production:
  - Medium and long-term (weeks and months):
    - Capacity planning.

6

Capacity Planning



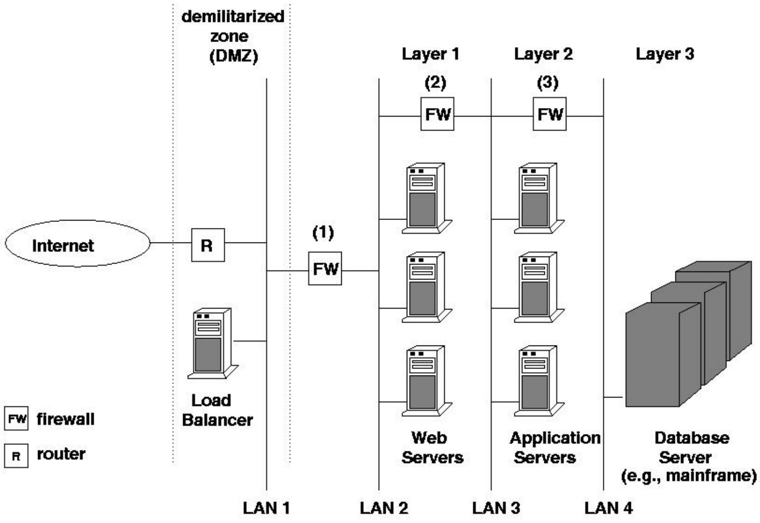
What are performance models good for?

- At the design stage:
  - Compare competing design alternatives.
    - A large number of low capacity servers vs. a small number of large capacity servers?
- During production:
  - Medium and long-term (weeks and months):
    - Capacity planning.
  - Short-term (minutes):
    - Dynamic reconfiguration.

Dynamic QoS Control: Motivation

- Computer systems are becoming very complex and composed of multiple tiers.

Multi-tier Architecture



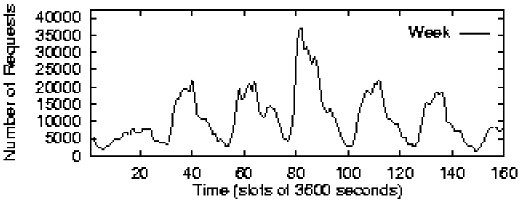
Dynamic QoS Control: Motivation

- Computer systems are becoming very complex and composed of multiple tiers.
- The workload presents short-term variations with high peak-to-average ratios.

11

Multi-scale workload variation

3600 sec

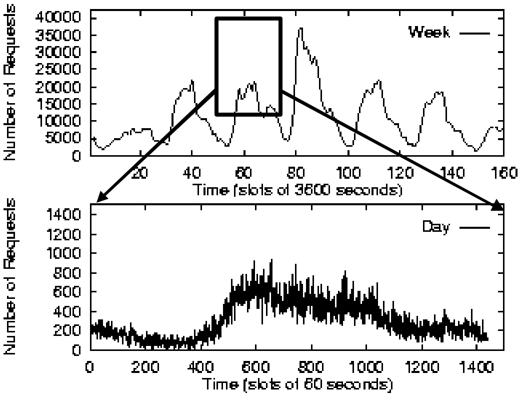


12

Multi-scale workload variation

3600 sec

60 sec

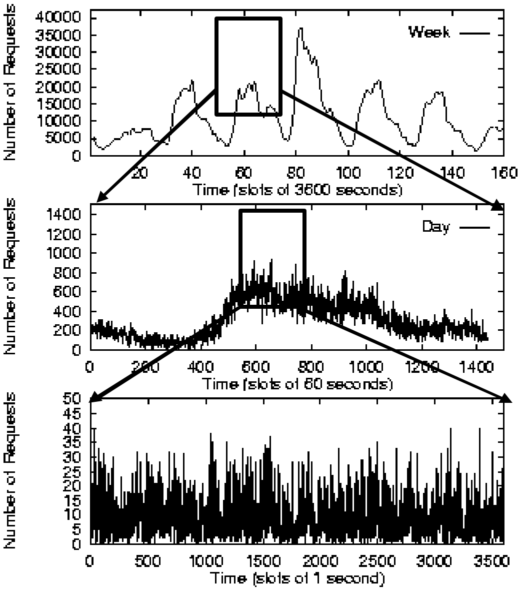


Multi-scale workload variation

3600 sec

60 sec

1 sec



Dynamic QoS Control: Motivation

- E-commerce sites are complex and composed of multiple tiers.
- The workload presents short-term variations with high peak-to-average ratios.
- Many software and hardware parameters influence the performance of computer systems.

⇒ Manual reconfiguration is not an option.

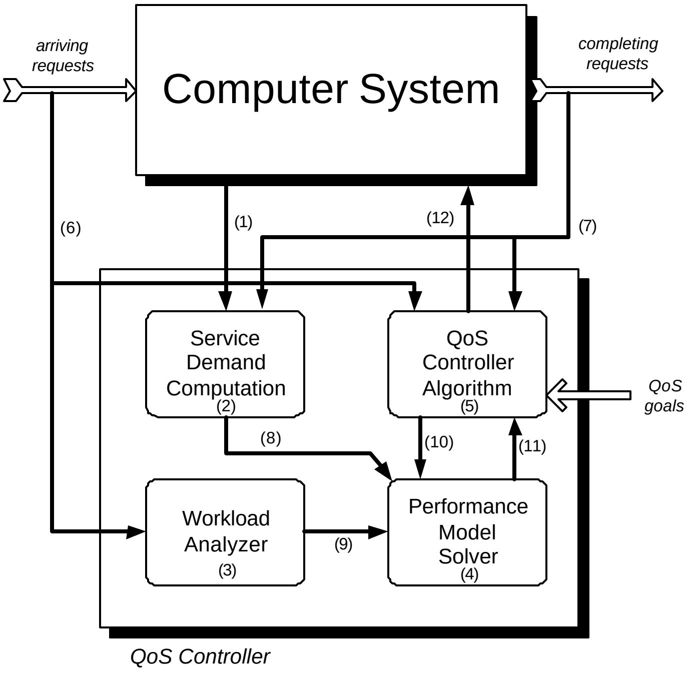
Need self-managing systems!

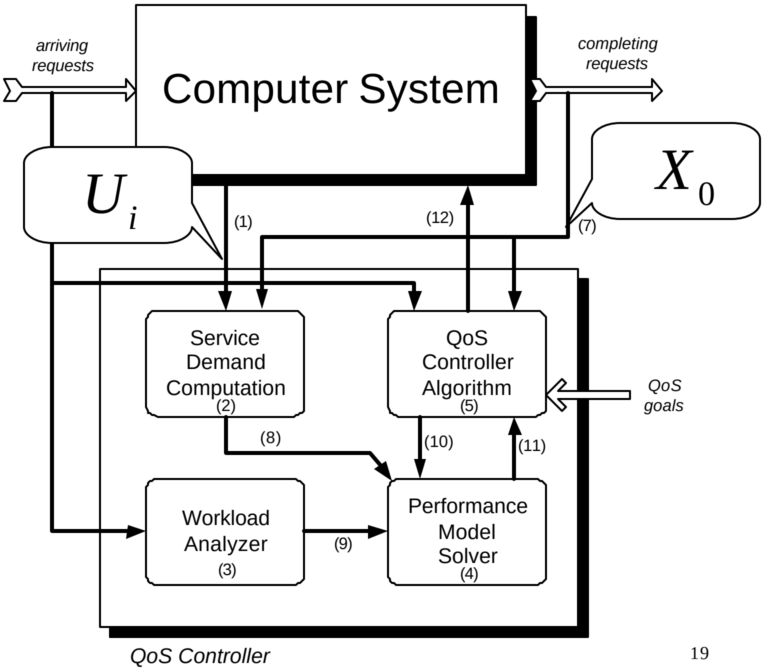
Outline of the rest of the talk

- Basic Approach to Self-managing Systems
- QoS metrics
- A Multi-Threaded Server Example
- Simulation Experiments
- Experiments on an Actual Web Server
- Concluding Remarks

Outline of the rest of the talk

- Basic Approach to Self-managing Systems
- QoS metrics
- A Multi-Threaded Server Example
- Simulation Experiments
- Experiments on an Actual Web Server
- Concluding Remarks





arriving requests

Computer System

completing requests

U_i

X_0

(1)

(12)

(7)

Service Demand Computation (2)

QoS Controller Algorithm (5)

QoS goals

(8)

(10)

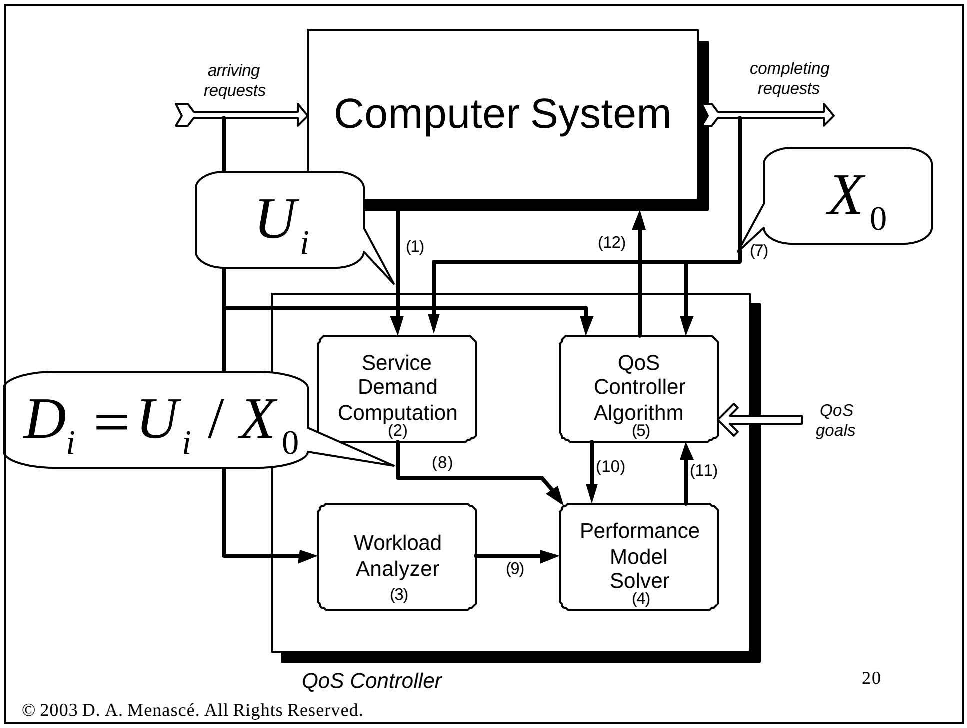
(11)

Workload Analyzer (3)

(9)

Performance Model Solver (4)

QoS Controller



arriving requests

Computer System

completing requests

U_i

X_0

(1)

(12)

(7)

Service Demand Computation (2)

QoS Controller Algorithm (5)

QoS goals

$D_i = U_i / X_0$

(8)

(10)

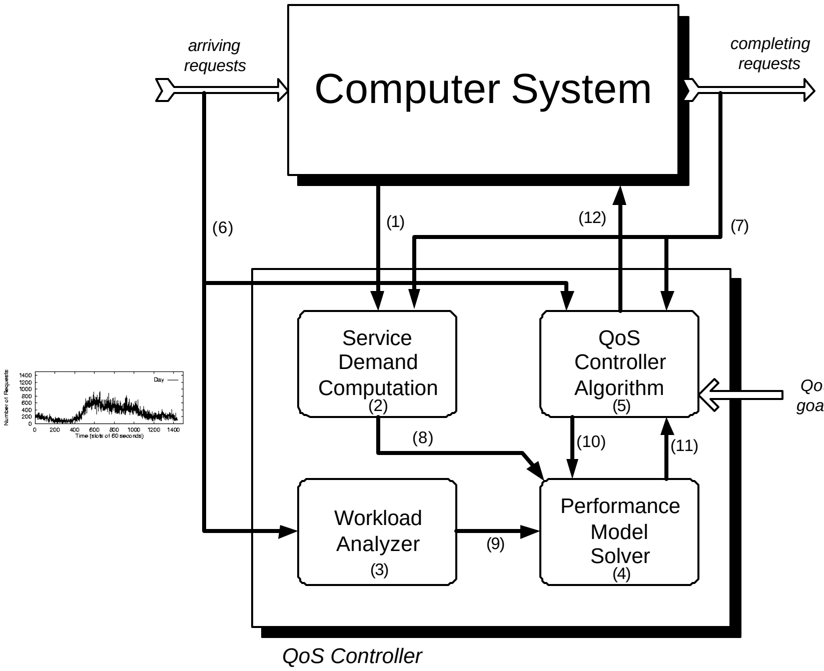
(11)

Workload Analyzer (3)

(9)

Performance Model Solver (4)

QoS Controller



Computer System

arriving requests

completing requests

(1) (6) (7) (12)

Service Demand Computation (2)

QoS Controller Algorithm (5)

QoS goals

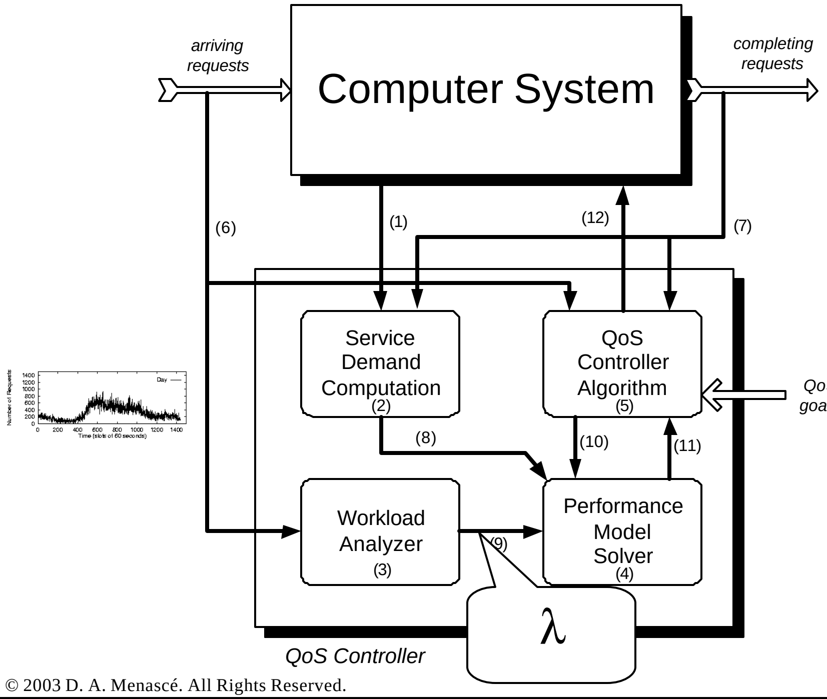
(8) (10) (11)

Workload Analyzer (3)

(9)

Performance Model Solver (4)

QoS Controller



Computer System

arriving requests

completing requests

(1) (6) (7) (12)

Service Demand Computation (2)

QoS Controller Algorithm (5)

QoS goals

(8) (10) (11)

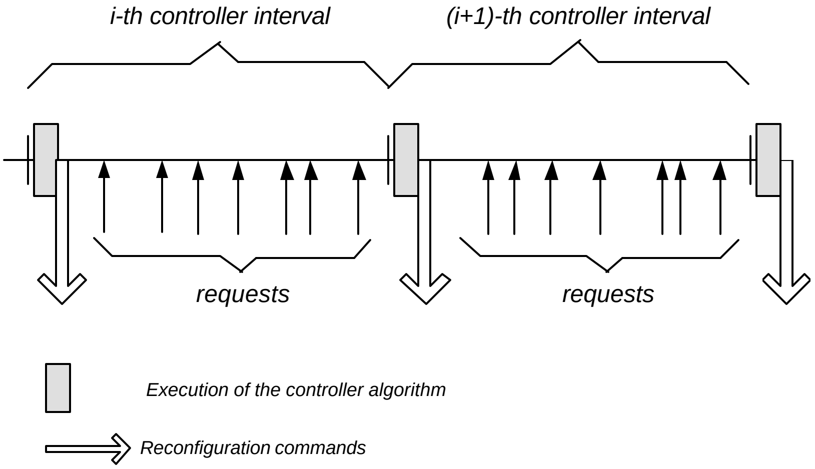
Workload Analyzer (3)

(9)

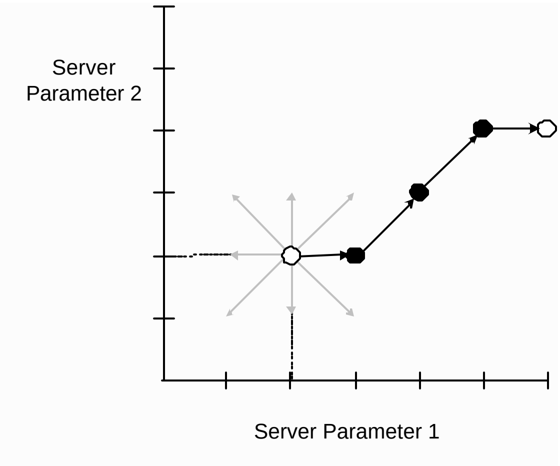
Performance Model Solver (4)

λ

QoS Controller

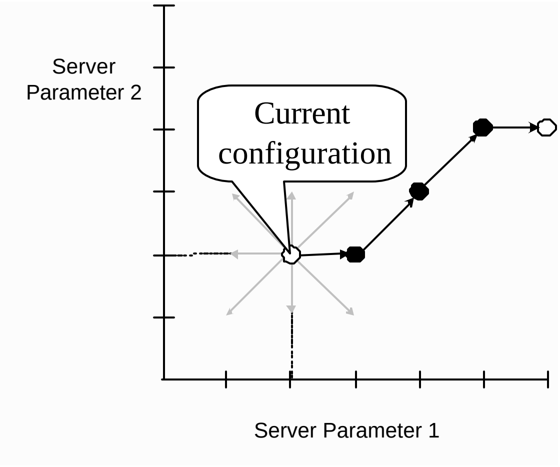

i-th controller interval
(i+1)-th controller interval
requests
requests
Execution of the controller algorithm
Reconfiguration commands

Heuristic Optimization Approach



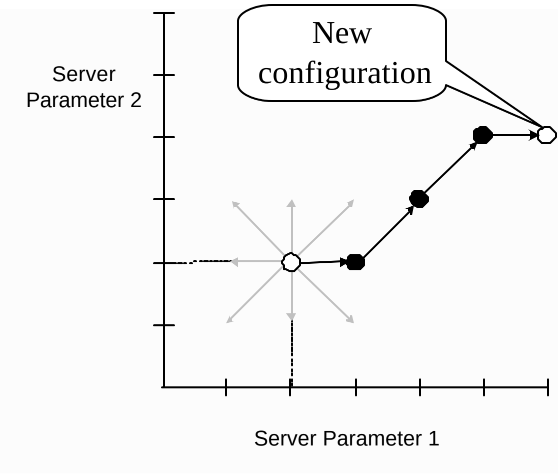
- The space of configuration points is searched using combinatorial search techniques.
- Each point has a QoS value computed through an analytic performance model.

Heuristic Optimization Approach



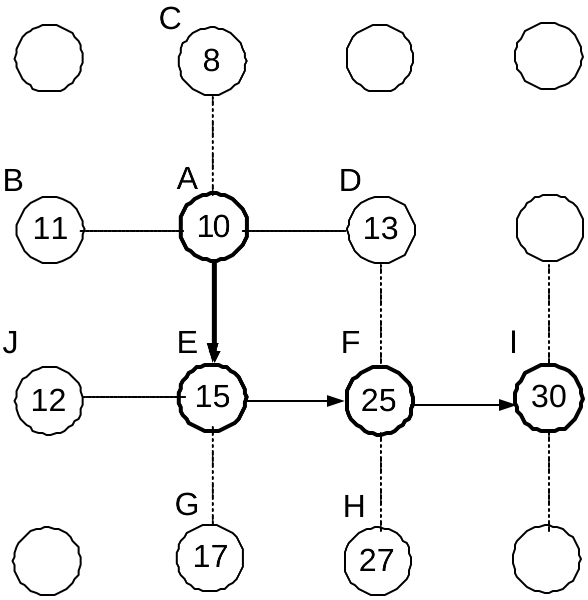
- The space of configuration points is searched using combinatorial search techniques.
- Each point has a QoS value computed through an analytic performance model.

Heuristic Optimization Approach

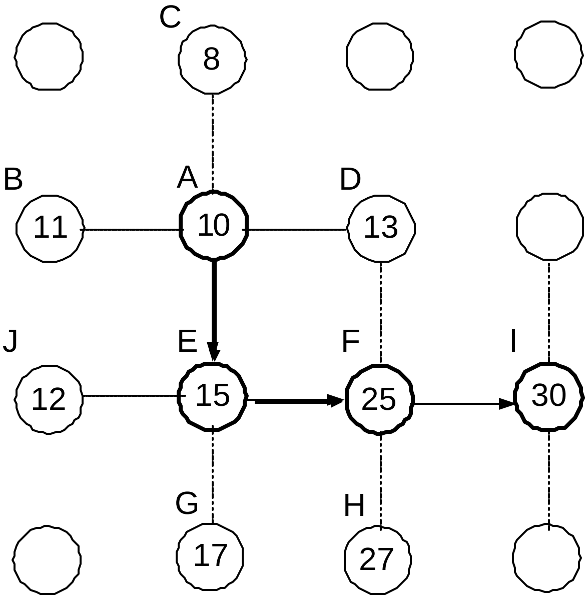


- The space of configuration points is searched using combinatorial search techniques.
- Each point has a QoS value computed through an analytic performance model.

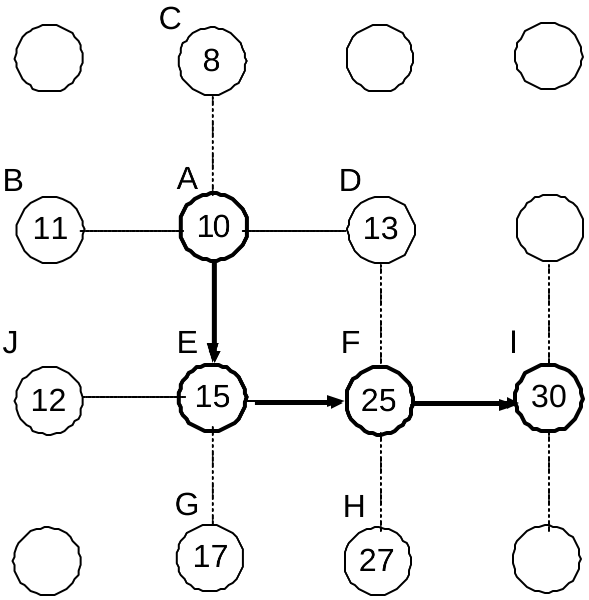
Hill-Climbing Search



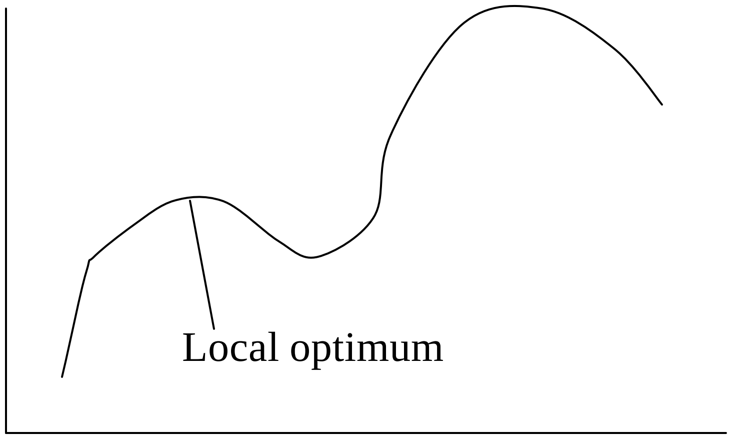
Hill-Climbing Search

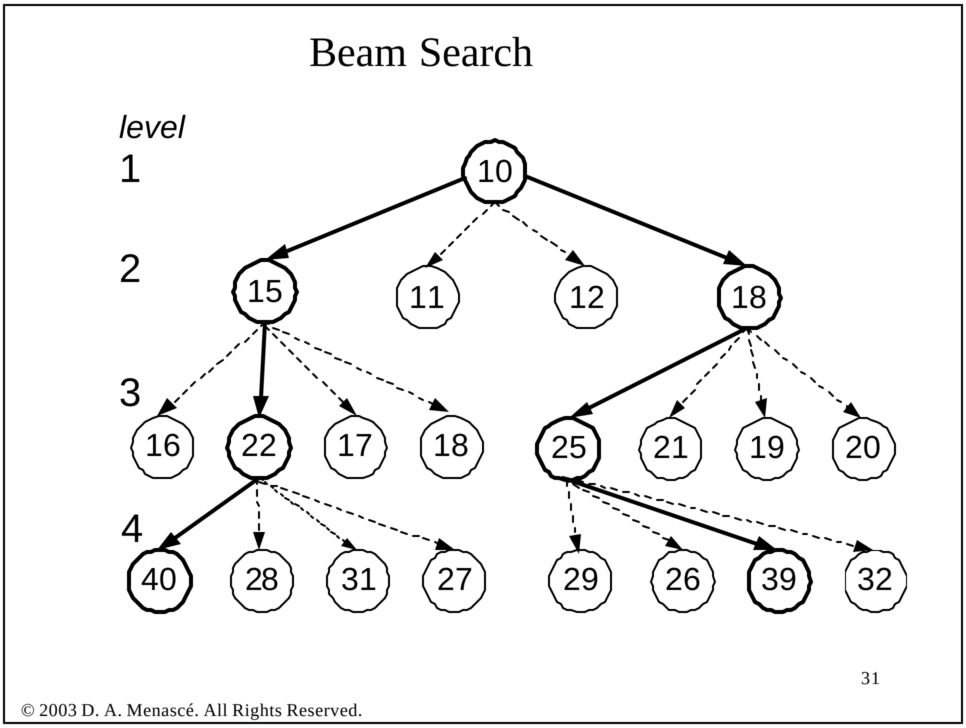


Hill-Climbing Search

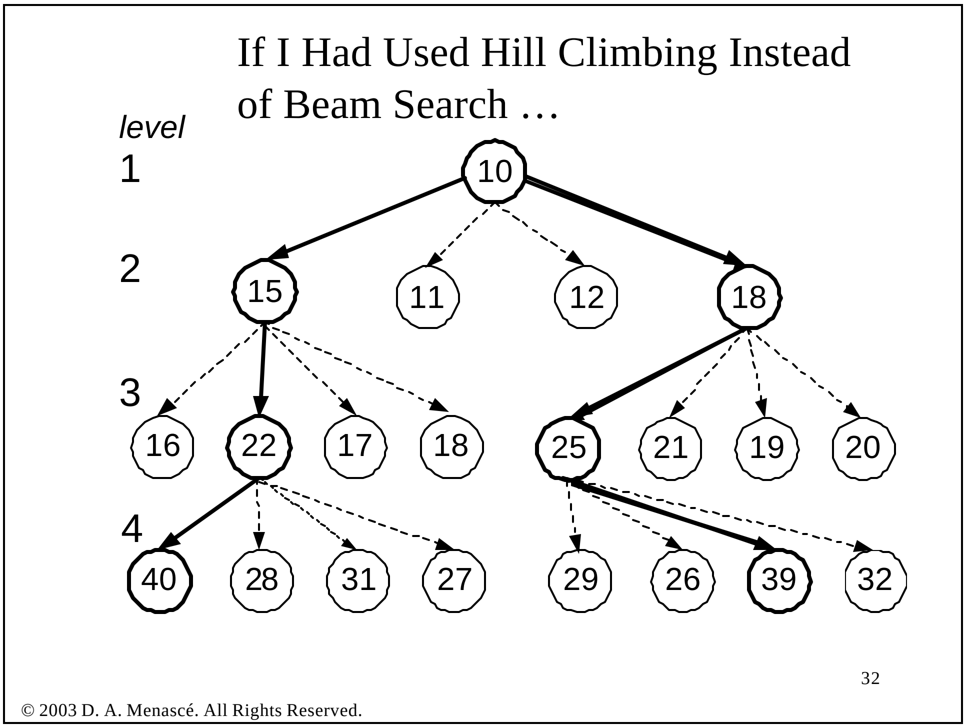


Problems with Hill-Climbing





Beam Search



If I Had Used Hill Climbing Instead of Beam Search …

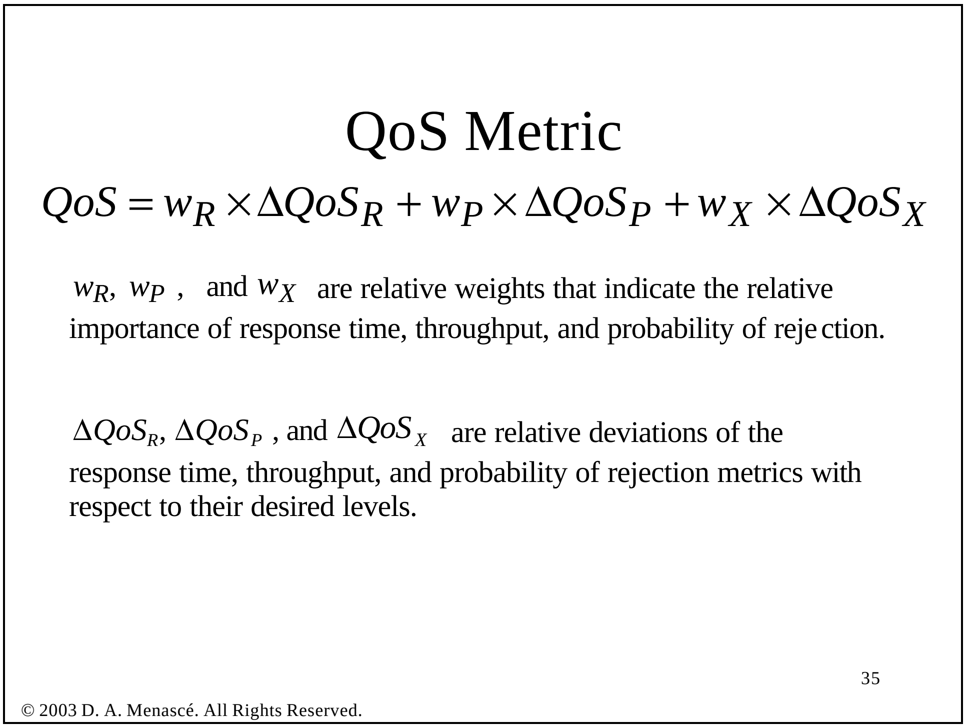
Outline of the rest of the talk

- Basic Approach to Self-managing Systems
- QoS metrics
- A Multi-Threaded Server Example
- Simulation Experiments
- Experiments on an Actual Web Server
- Concluding Remarks

QoS Metric

$$QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X$$

w_R, w_P , and w_X are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

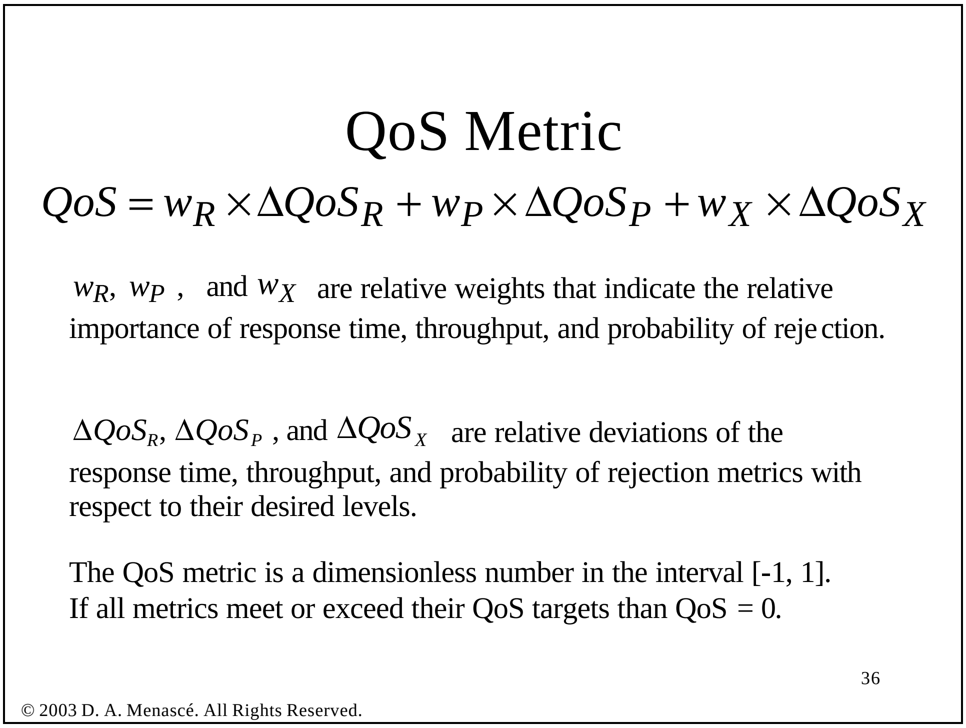


QoS Metric

$$QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X$$

w_R, w_P , and w_X are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

ΔQoS_R, ΔQoS_P , and ΔQoS_X are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.



QoS Metric

$$QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X$$

w_R, w_P , and w_X are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

ΔQoS_R, ΔQoS_P , and ΔQoS_X are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.

The QoS metric is a dimensionless number in the interval [-1, 1].
If all metrics meet or exceed their QoS targets than QoS = 0.

Response Time Deviation

$$\Delta QoS_R = \frac{R_{\max} - R_{measured}}{\max(R_{\max}, R_{measured})}$$

- = 0 if the response time meets its target.
- > 0 if the response time exceeds its target.
- < 0 if the response time does not meet its target.

- $\Delta QoS_R \leq 1 - (\sum_{i=1}^{K} D_i) / R_{\max} < 1$
- $-1 < -(1 - R_{\max} / R_{measured}) \leq \Delta QoS_R$

37

Probability of Rejection Deviation

$$\Delta QoS_P = \frac{P_{\max} - P_{measured}}{\max(P_{\max}, P_{measured})}$$

- = 0 if the probability of rejection meets its target.
- > 0 and = 1 if the probability of rejection exceeds its target.
- < 0 and = -1 if the probability of rejection does not meet its target.

38

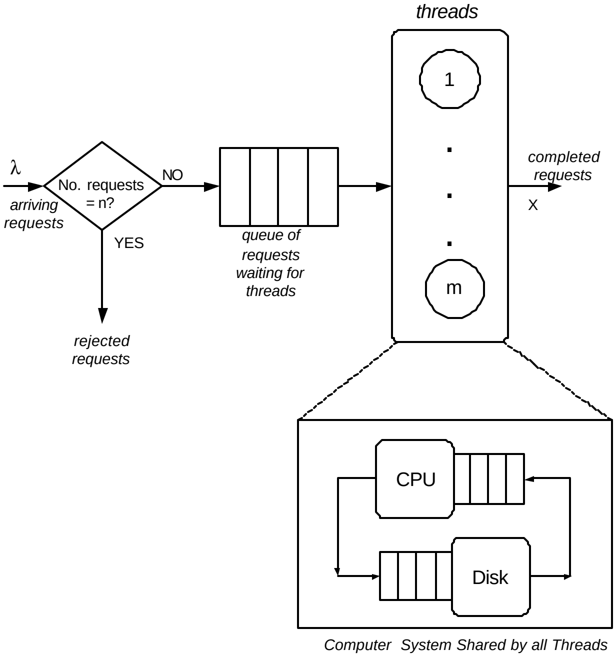
Throughput Deviation

$$\Delta QoS_X = \frac{X_{measured} - X^*_{\min}}{\max(X_{measured}, X^*_{\min})}$$

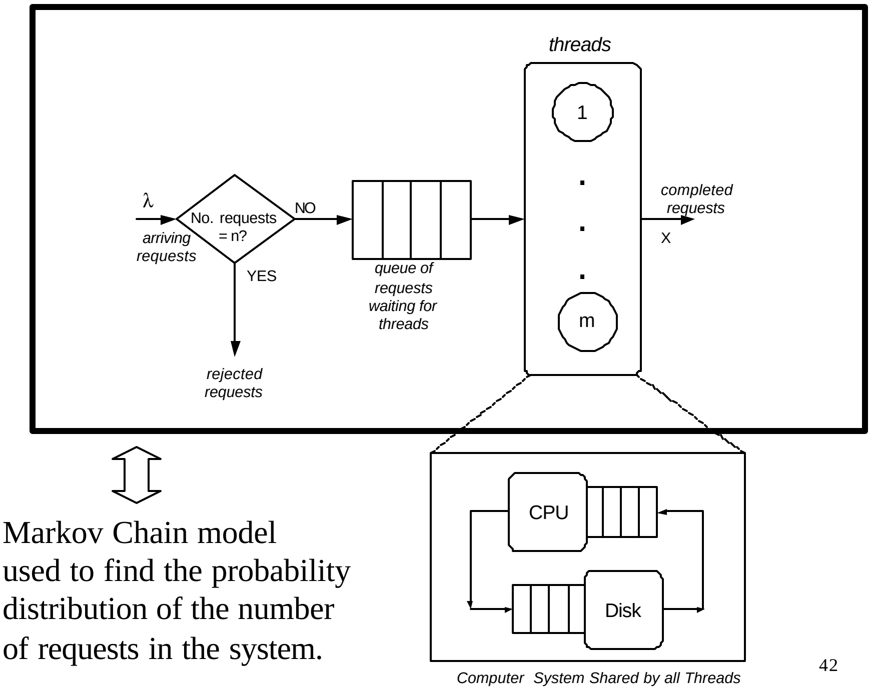
- $X^*_{\min} = \min(\, l\,, X_{\min})$
- = 0 if the throughput meets its target.
- > 0 and < 1 if the throughput exceeds its target.
- < 0 and > -1 if the throughput does not meet its target.

Outline of the rest of the talk

- Basic Approach to Self-managing Systems
- QoS metrics
- A Multi-Threaded Server Example
- Simulation Experiments
- Experiments on an Actual Web Server
- Concluding Remarks



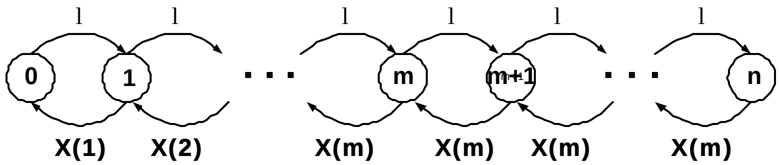
41



Markov Chain model
used to find the probability
distribution of the number
of requests in the system.

42

Markov Chain Model

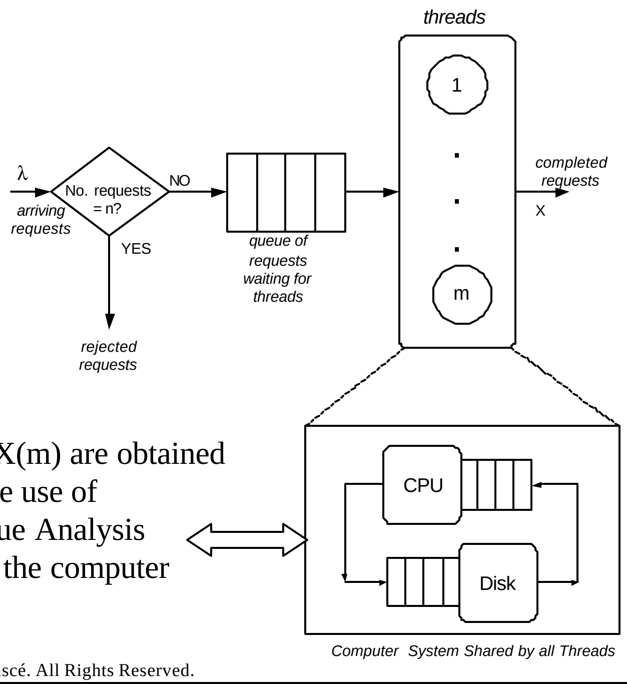


$$P_k = \begin{cases} P_0 \, l^k / b(k) & k = 1, \ldots, m \\ P_0 \, r^k X(m)^m / b(m) & k = m+1, \ldots, n \end{cases}$$

where $b(k) = X(1) \times X(2) \times \ldots \times X(k)$, $r = l / X(m)$, and

$$P_0 = \left[1 + \sum_{k=1}^{m} \frac{l^k}{b(k)} + \frac{r \times l^m (1 - r^{n-m})}{b(m)(1-r)} \right]^{-1}.$$

43



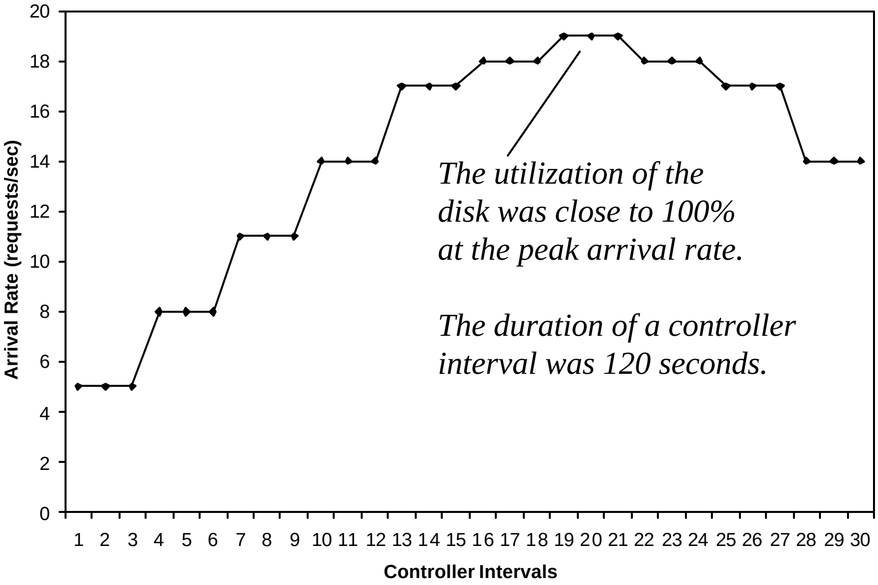
X(1), …, X(m) are obtained through the use of Mean Value Analysis applied to the computer system.

44

Outline of the rest of the talk

- Basic Approach to Self-managing Systems
- QoS metrics
- A Multi-Threaded Server Example
- Simulation Experiments
- Experiments on an Actual Web Server
- Concluding Remarks

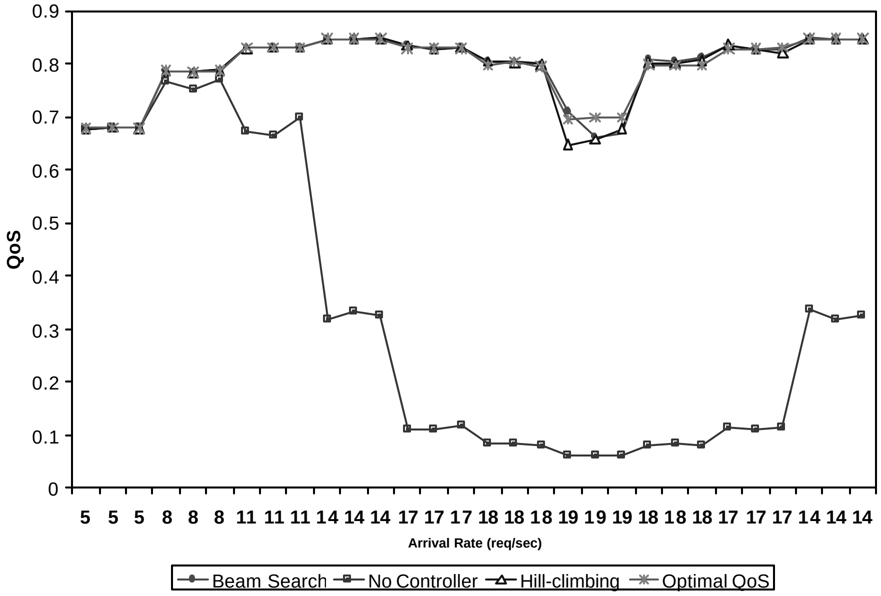
Evolution of Workload Intensity



The utilization of the disk was close to 100% at the peak arrival rate.

The duration of a controller interval was 120 seconds.

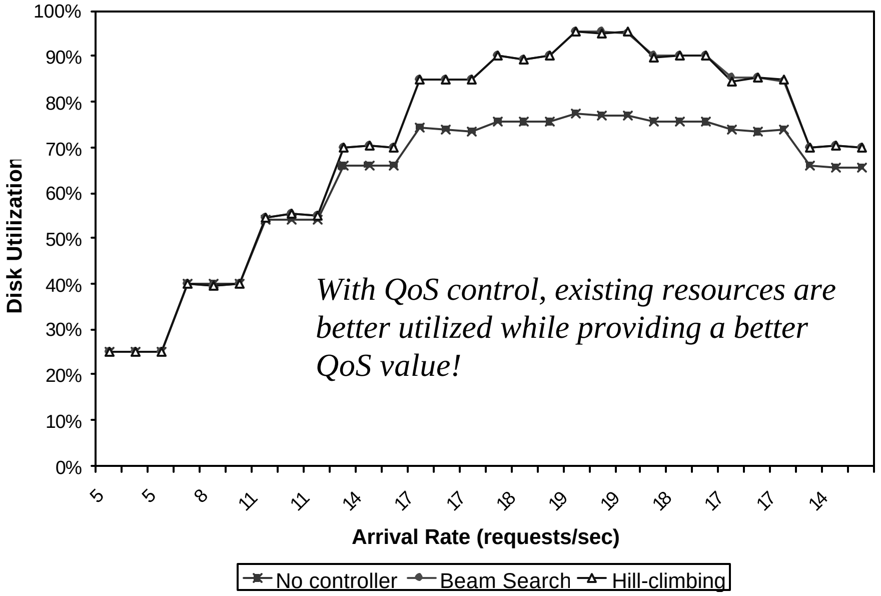
QoS Variation During the Experiment



Some details about this experiment

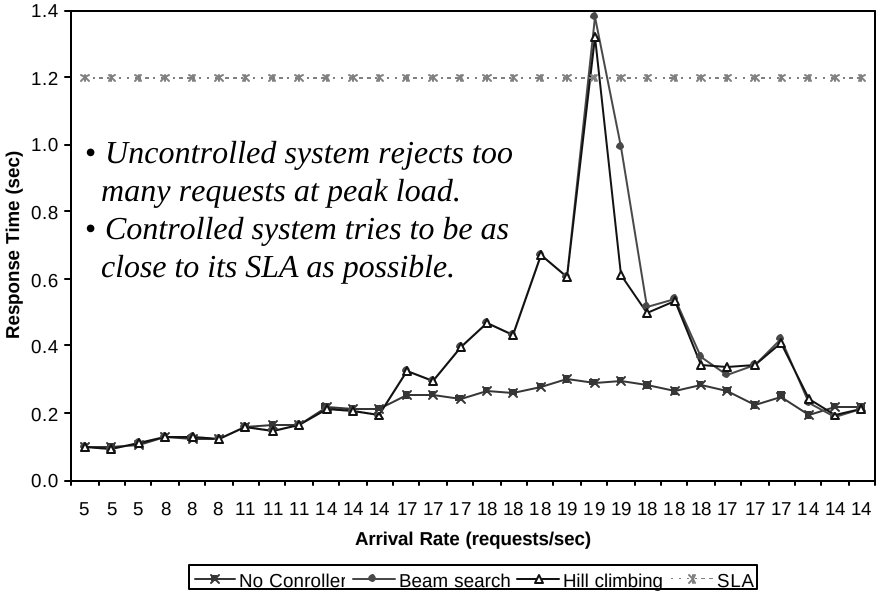
- The “optimal” QoS was computed off-line by considering all possible 9,801 configurations at each controller interval.
- Both heuristics evaluated no more than 120 configurations (i.e, 1.2% of the total) per controller interval

Disk Utilization During the Experiment



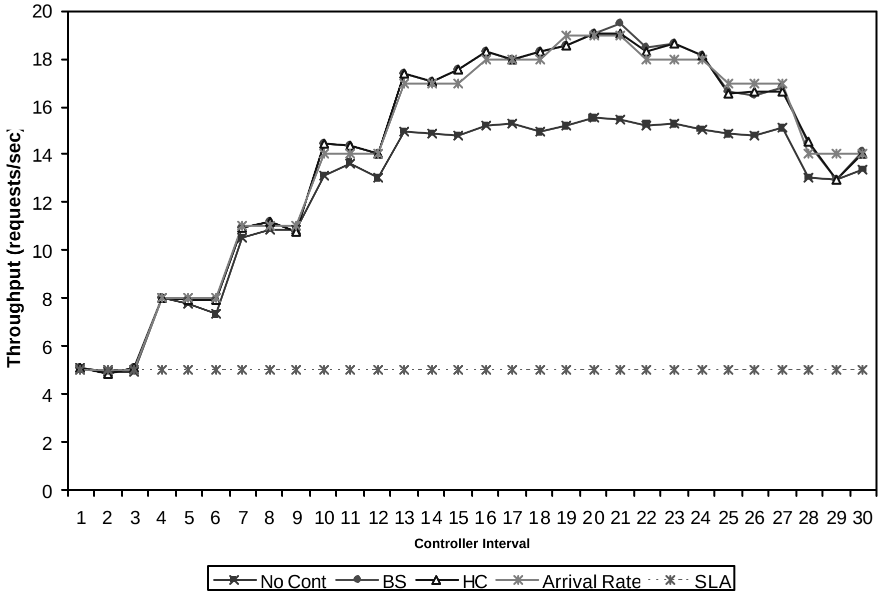
With QoS control, existing resources are better utilized while providing a better QoS value!

Response Time Variation During the Experiment



- *Uncontrolled system rejects too many requests at peak load.*
- *Controlled system tries to be as close to its SLA as possible.*

Throughput Variation During the Experiment



51

Outline of the rest of the talk

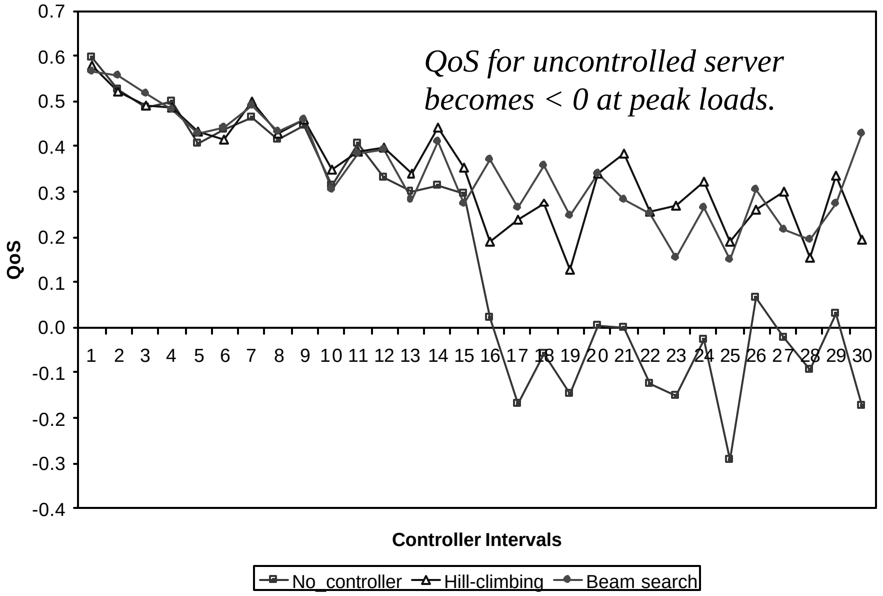
- Basic Approach to Self-managing Systems
- QoS metrics
- A Multi-Threaded Server Example
- Simulation Experiments
- Experiments on an Actual Web Server
- Concluding Remarks

52

Experiments with an Actual Web Server

- Controller code is the same as in the simulation.
- HTTP server: Apache 1.3.12 modified to allow for dynamic number of threads (m) and maximum number of requests (n).
- Workload generated by SURGE (Scalable URL Reference Generator)

Experiments with an Actual Web Server



QoS for uncontrolled server becomes < 0 at peak loads.

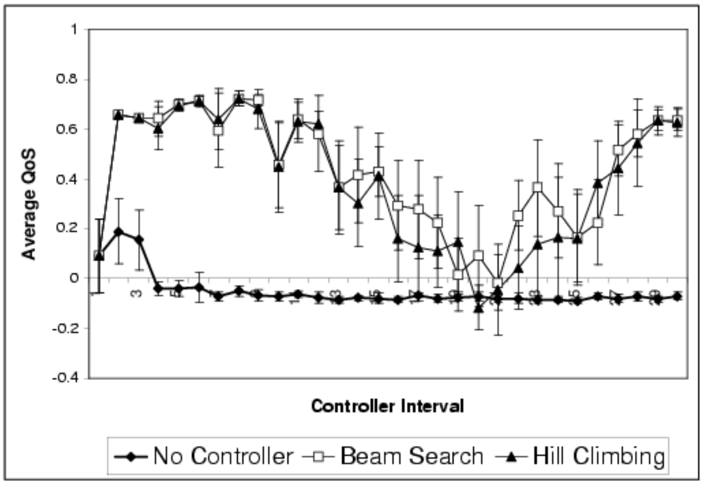
Outline of the rest of the talk

- Basic Approach to Self-managing Systems
- QoS metrics
- A Multi-Threaded Server Example
- Simulation Experiments
- Experiments on an Actual Web Server
- Concluding Remarks

Concluding Remarks

- Analytic models can be used to design self-managing systems due to their ability to track the evolution of the performance metrics as the workload and configuration parameters change.
- Experiments show that this approach is robust even when the coefficients of variation of the inter-arrival times and service times are high (around 3).

Highly Variable Workloads
Ca = 2 and Cs = 4



57

Concluding Remarks

- Self-managing systems provide better resource utilization while meeting the SLAs.
- Self-managing systems allow existing resources to be used to their fullest extent before upgrades are needed. Better ROI!
- Self-managing systems reduce personnel costs.
- Other approaches: control theory.

58

Bibliography

- "Assessing the Robustness of Self-Managing Computer Systems under Highly Variable Workloads," M. Bennani and D.A. Menascé, *Proc. International Conf. Autonomic Computing (ICAC-04)*, New York, NY, May 17-18, 2004.
- "Automatic QoS Control," D.A. Menascé, *IEEE Internet Computing*, January/February 2003, Vol. 7, No. 1.
- Preserving QoS of E-commerce Sites Through Self-Tuning: A Performance Model Approach," D. A. Menascé, R. Dodge and D. Barbara, *Proc. 2001 ACM Conference on E-commerce*, Tampa, FL, October 14-17, 2001.