# Observe-Analyze-Act Paradigm for Storage System Resource Arbitration

***Li Yin*[1]**
***Email: yinli@eecs.berkeley.edu***

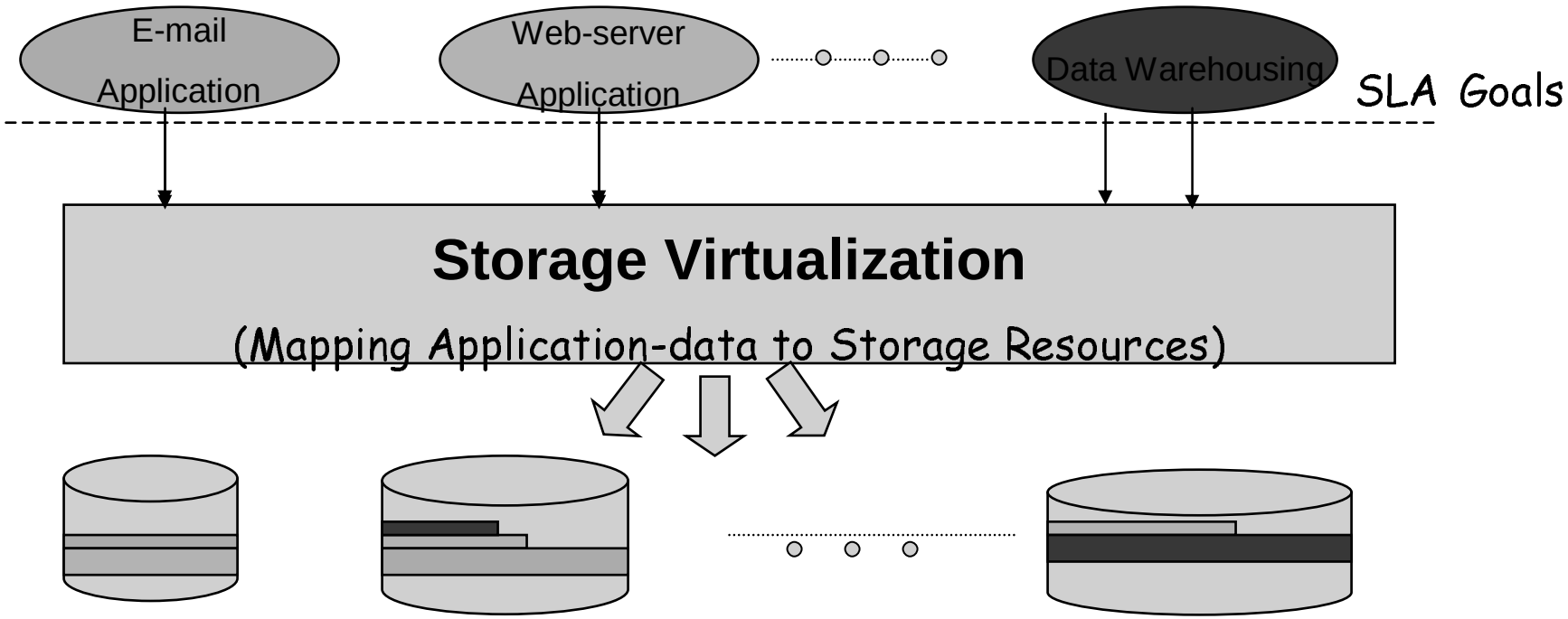
Joint work with: *Sandeep Uttamchandani*[2]
*Guillermo Alvarez*[2]
*John Palmer*[2]
*Randy Katz*[1]

*1:University of California, Berkeley*
*2: IBM Almaden Research Center*

# Outline

- Observe-analyze-act in storage system: CHAMELEON
  - Motivation
  - System model and architecture
  - Design details
  - Experimental results
- Observe-analyze-act in other scenarios
  - Example: network applications
- Future challenges

# Need for Run-time System Management

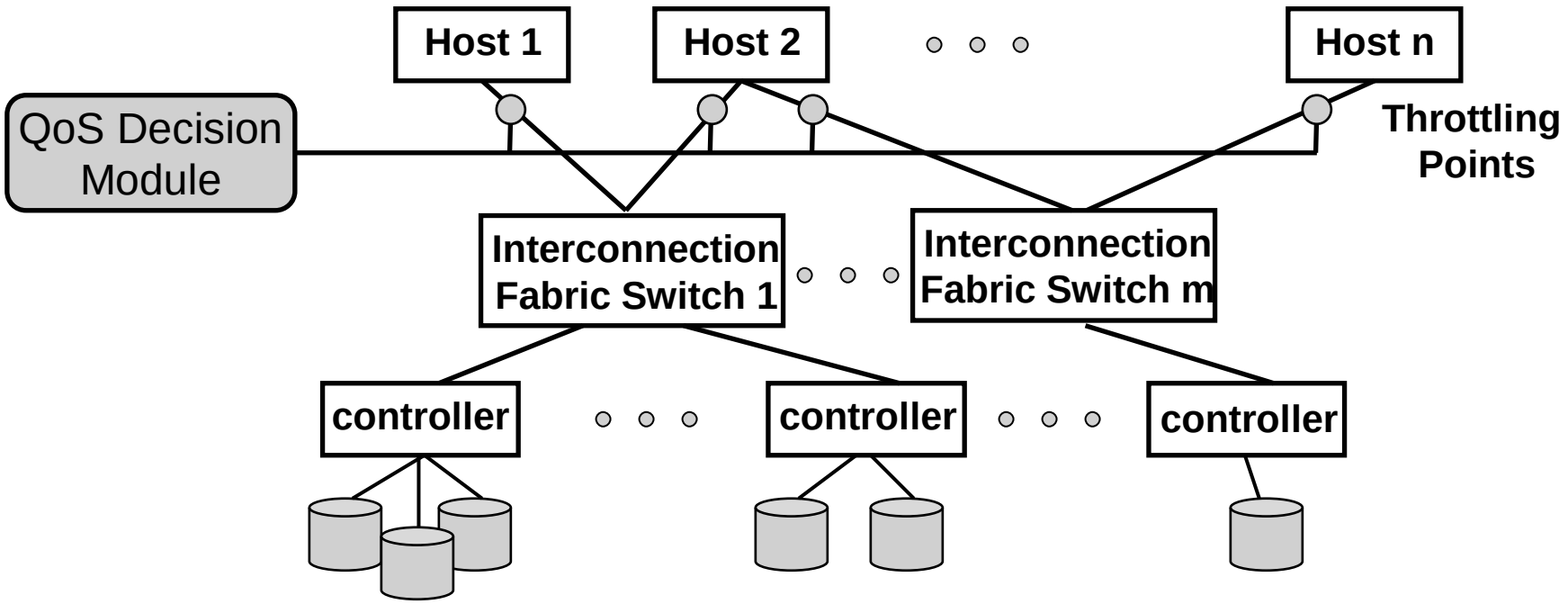



- Static resource allocation is not enough
  - Incomplete information of the access characteristics: workload variations; change of goals
  - Exception scenarios: hardware failures; load surges.

# Approaches for Run-time Storage System Management

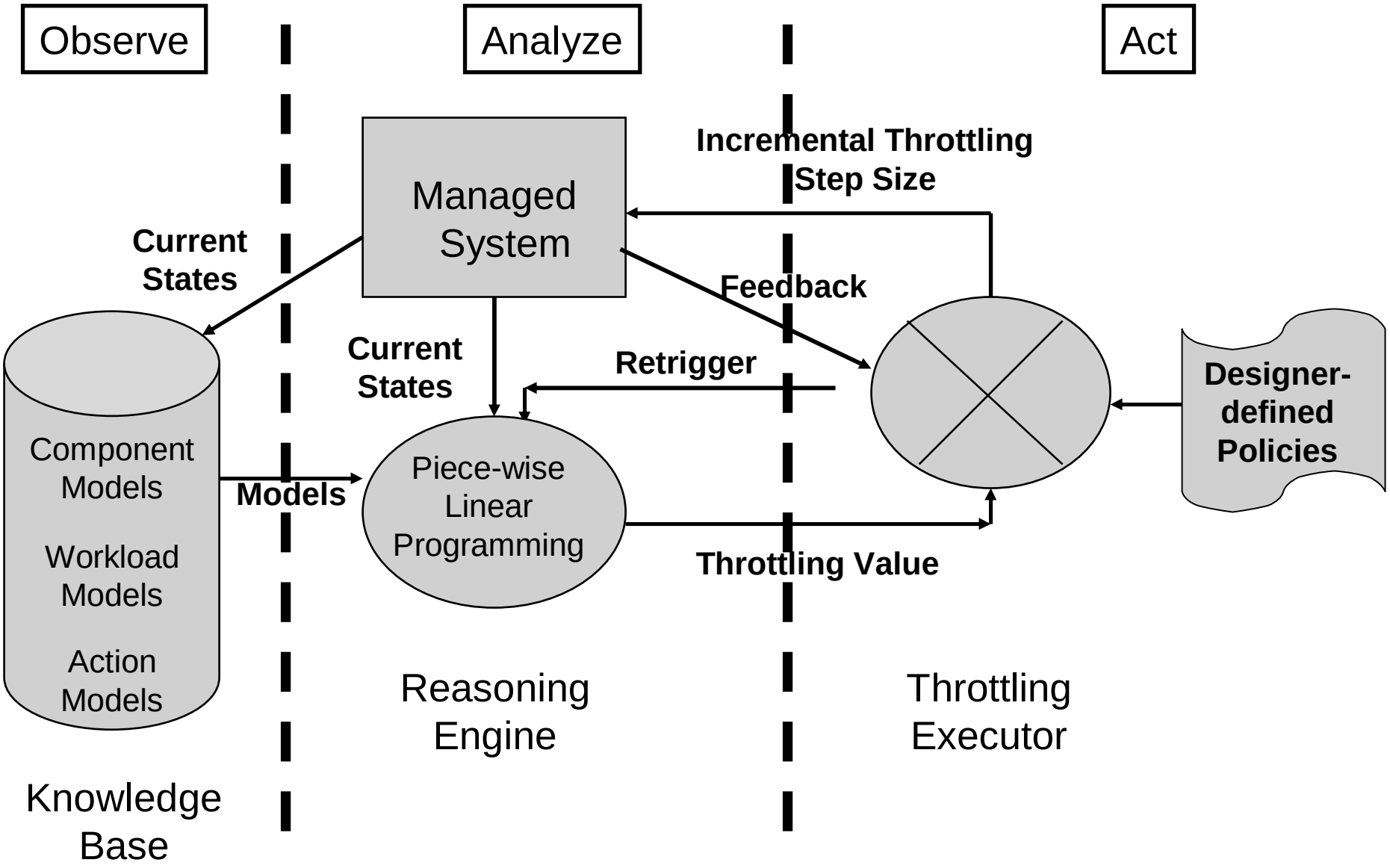
- Today: Administrator observe-analyze-act
- Automate the observe-analyze-act:
  - Rule-based system
    - Complexity
    - Brittleness
  - Pure feedback-based system
    - Infeasible for real-world multi-parameter tuning
  - Model-based approaches
    - Challenges:
      - How to **represent** system details as models?
      - How to **create/evolve** models?
      - How to **use** models for decision making?

# System Model for Resource Arbitration




- Input:
  - SLAs for workloads
  - Current system status (performance)
- Output:
  - Resource reallocation action (Throttling decisions)

# Our Solution: CHAMELEON

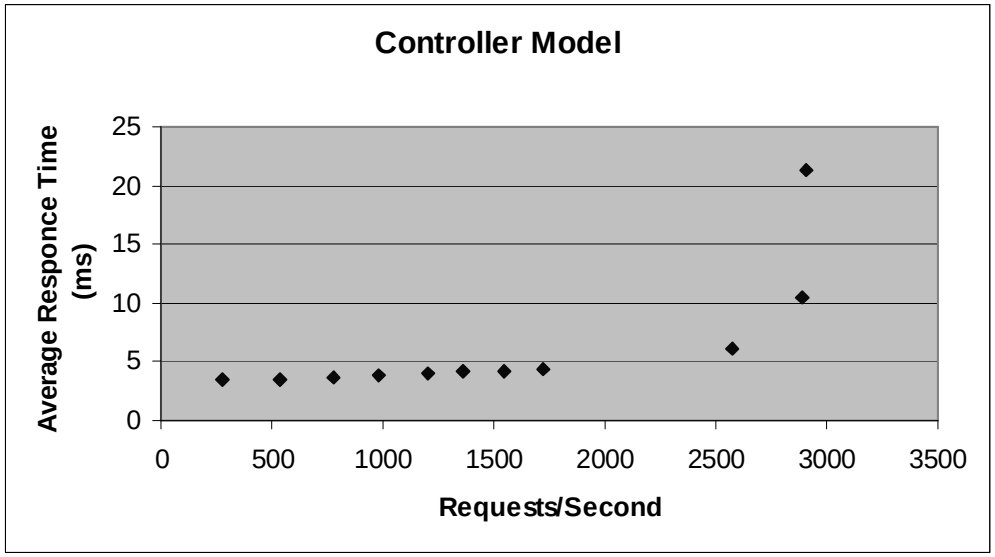


# Knowledge Base: Component Model

- Objective: Predict service time for a given load at a component (For example: storage controller).

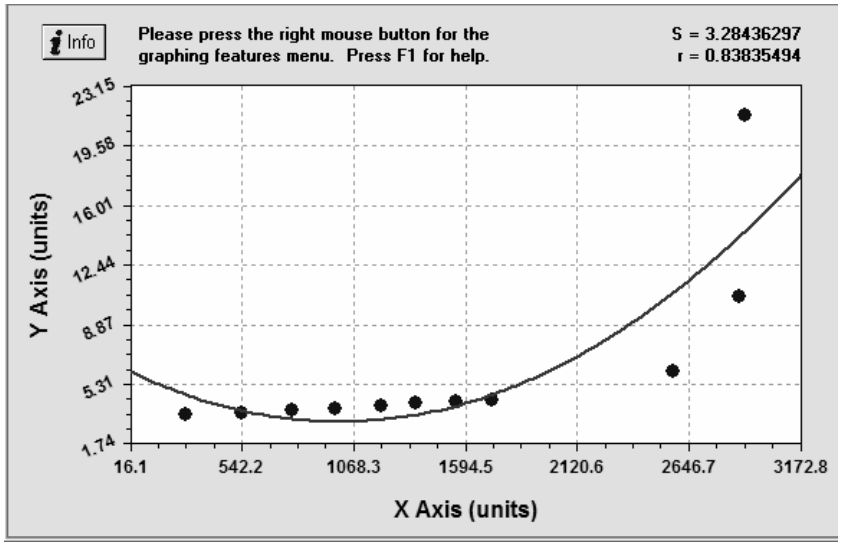
  $Service\_time_{controller} = L($ *request size, read write ratio, random sequential ratio, request rate)*

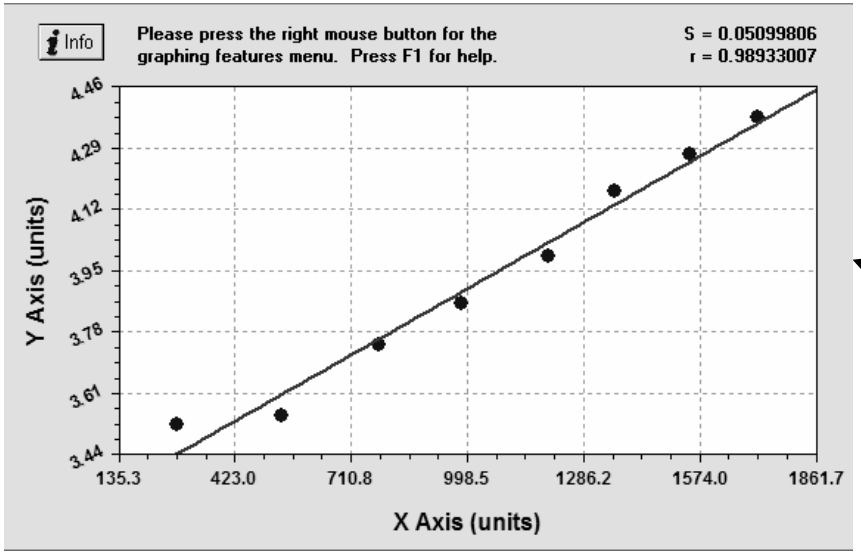
- An example of component model
  - FAStT900, 30 disks, RAID0
  - Request Size 10KB, Read/Write Ratio 0.8, Random Access

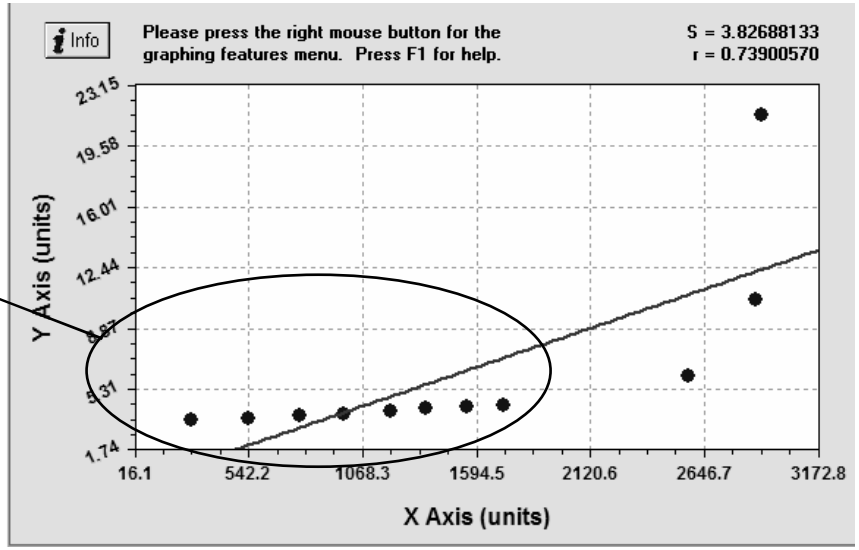


# Component Model (cont.)

- Quadratic Fit
  - S = 3.284, r = 0.838
- Linear Fit
  - S = 3.8268, r = 0.739
- Non-saturated case: Linear Fit
  - S = 0.0509, r = 0.989



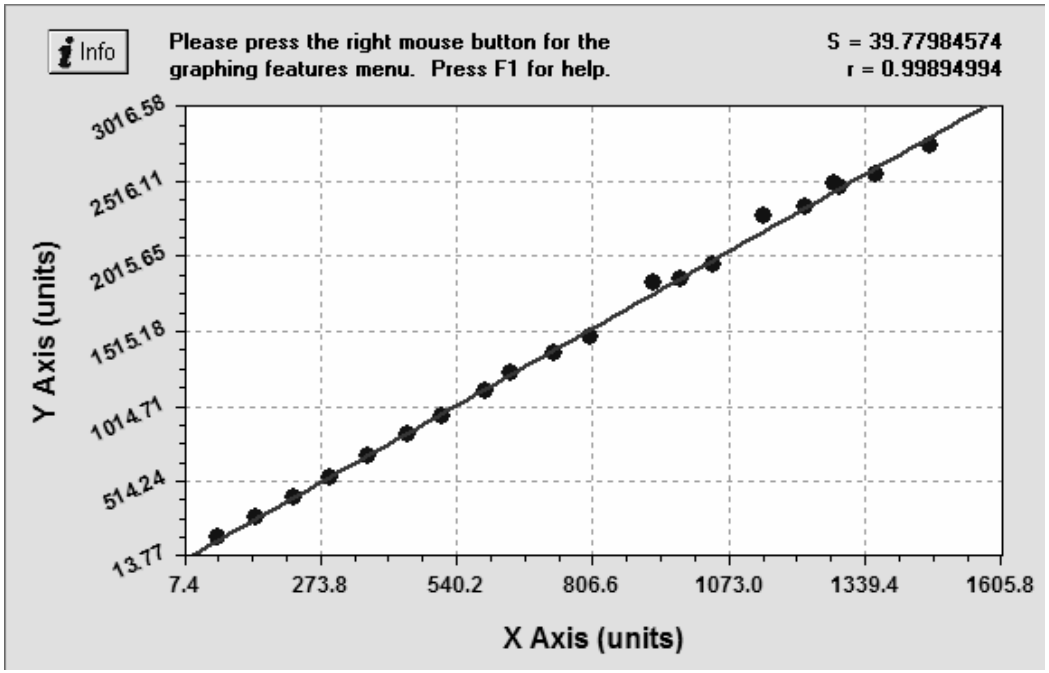



# Knowledge Base: Workload Model

- Objective: Predict the load on component i as a function of the request rate j

  $Component\_load_{i,j} = W_{i,j}(\ workload\ j\ request\ rate)$

- Example:
  - Workload with 20KB request size,
    0.642 read/write ratio and 0.026 sequential access ratio

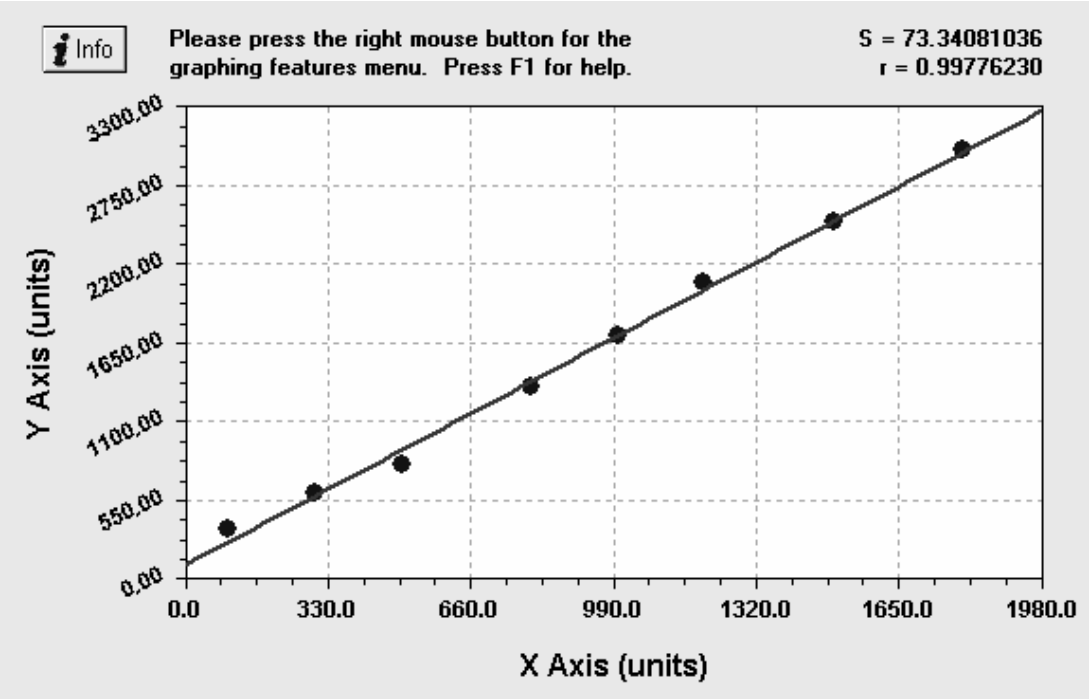


# Knowledge Base: Action Model

- Objective: Predict the effect of corrective actions on workload requirements

  *Workload J request Rate = $A_j$(Token Issue Rate for Workload J)*

- Example:

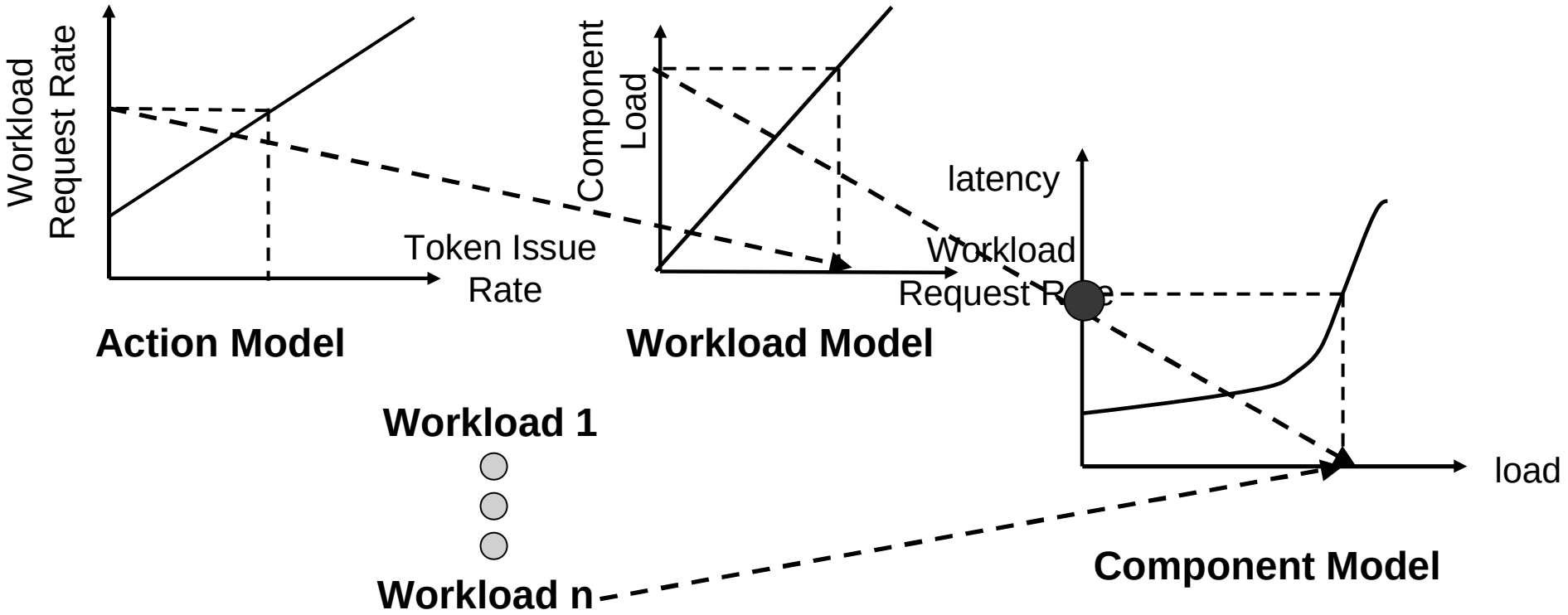


# Analyze Module: Reasoning Engine

- Formulated as a constraint solving problem
  - Part 1: Predict Action Behavior:
    For each candidate throttling decision, predict its performance result based on knowledge base
  - Part 2: Constraint Solving:
    Use linear programming technique to scan all feasible solutions and choose the optimal one

# Reasoning Engine: Predict Result

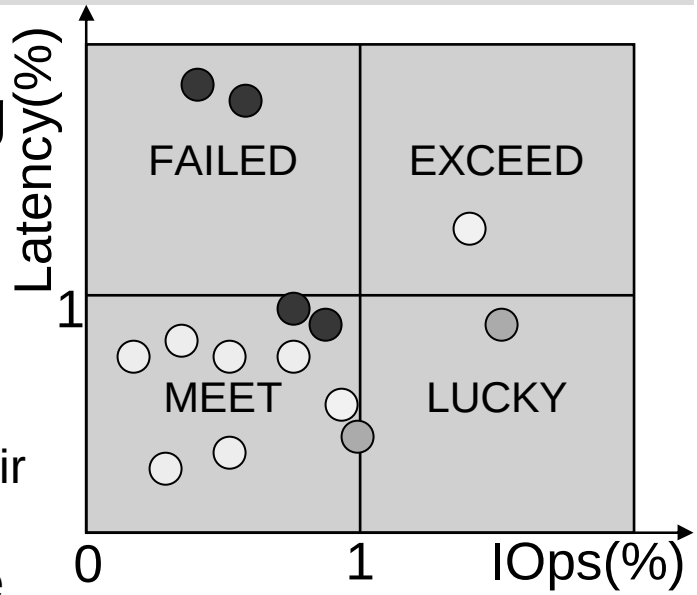
- Chain all models together to predict action result
- Input: Token issue rate for each workloads
- Output: Expected latency



# Reasoning Engine: Constraint Solving

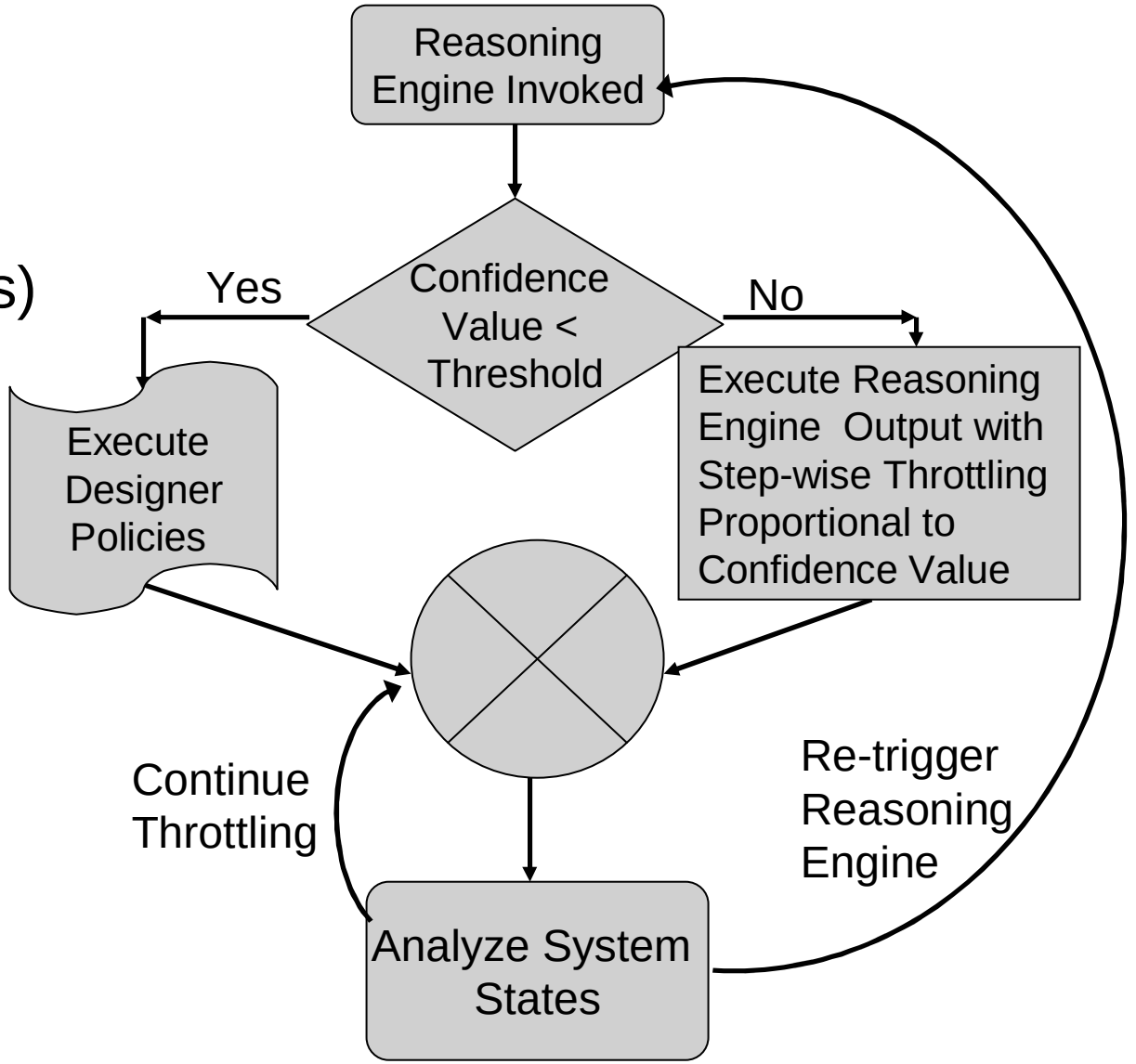
- Formulated using Linear Programming
- Formulation:
  - Variable: Token issue rate for each workload
  - Objective Function:
    - Minimize number of workloads violating their SLA goals
    - Workloads are as close to their SLA IO rate as possible
    - Example: Minimize $\sum \frac{p_{ai}p_{bi}[\ SLA_i - T(current\_throughput_i, t_i)]}{SLA_i}$

      where $p_{ai}$= Workload priority $p_{bi}$ = Quadrant priority
  - Constraints:
    - Workloads should meet their SLA latency goals



# Act Module: Throttling Executor

- Hybrid of feedback and prediction
- Ability to switch to rule-based (policies) when confidence value is low
- Ability to re-trigger reasoning engine

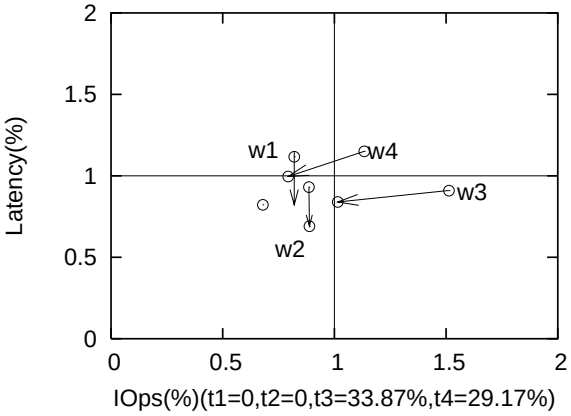


# Experimental Results

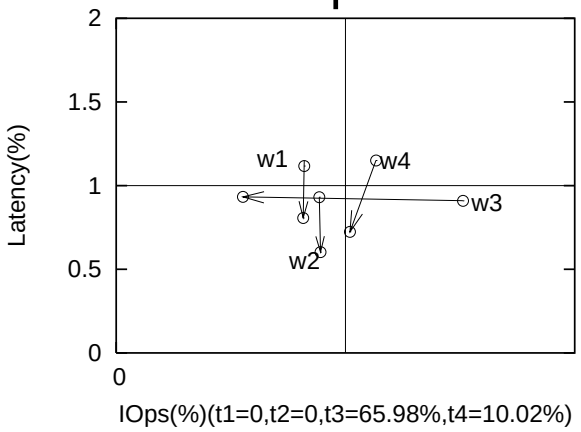
- Test-bed configuration:
  - IBM x-series 440 server (2.4GHz 4-way with 4GB memory, redhat server 2.1 kernel)
  - FAStT 900 controller
  - 24 drives (RAID0)
  - 2Gbps FibreChannel Link
- Tests consist of:
  - Synthetic workloads
  - Real-world trace replay (HP traces and SPC traces)

# Experimental Results: Synthetic Workloads

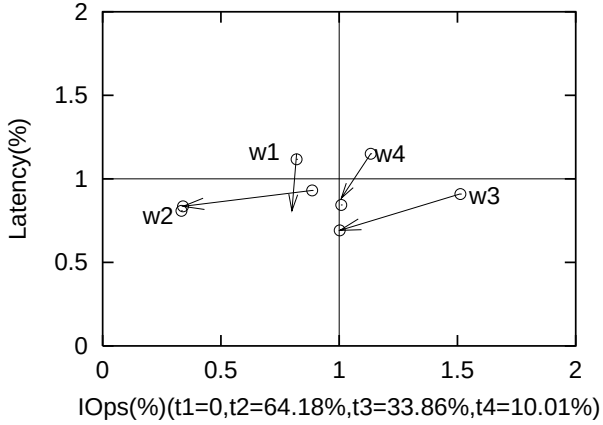
- Effect of priority values on the output of constraint solver




**(a) Equal priority**

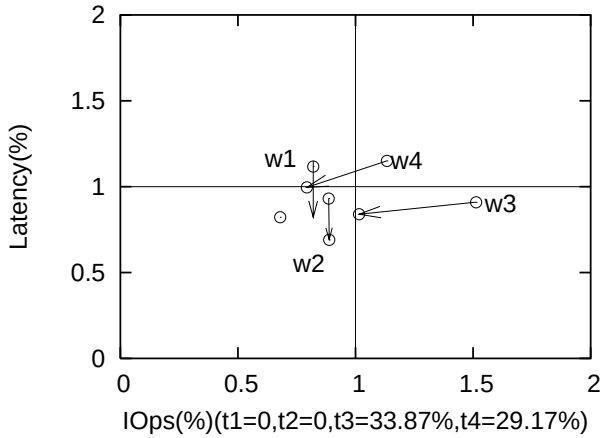



**(b) Workload priorities**

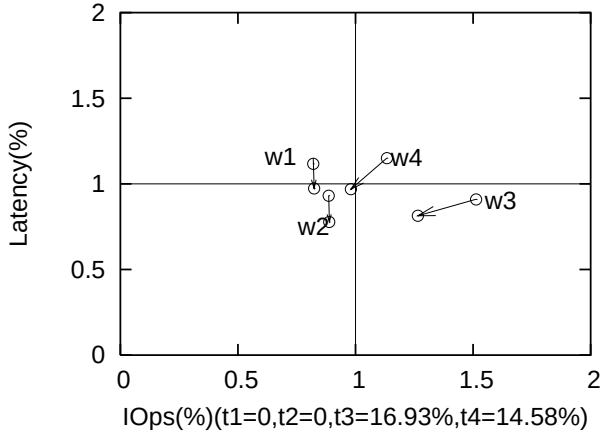



**(c ) Quadrant priorities**

- Effect of model errors on output of the constraint solver




**(a) Without feedback**



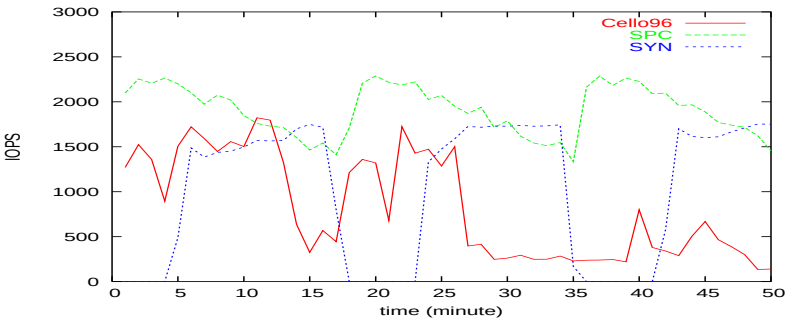

**(b) with feedback**

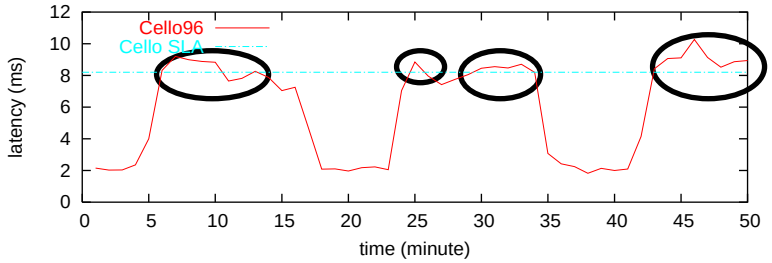
# Experiment Result: Real-world Trace Replay

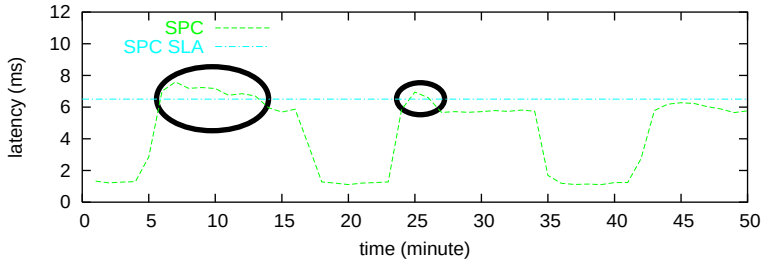
- Real-world block-level traces from HP (cello96 trace) and SPC (web server)
- A phased synthetic workload acts as the third flow
- Test goals:
  - Do they converge to SLAs?
  - How reactive the system is?
  - How does CHAMELEON handle unpredictable variations?

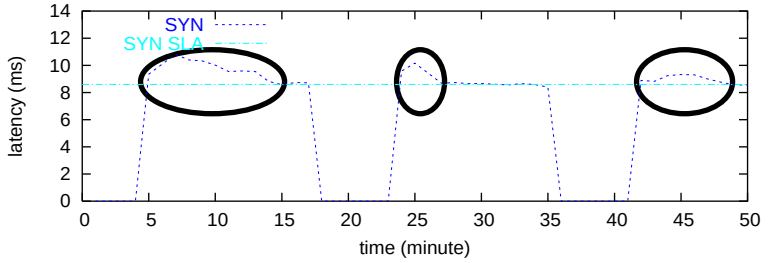
# Real-world Trace Replay

- Without CHAMELEON:

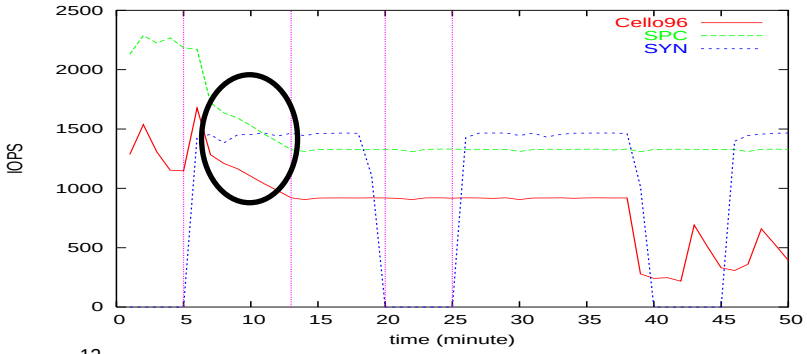


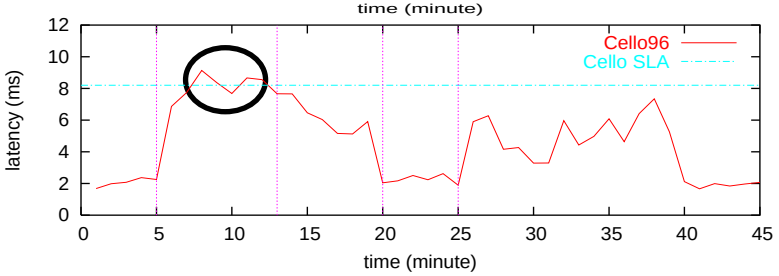



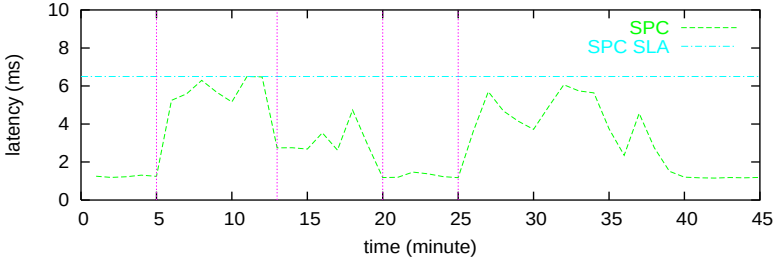



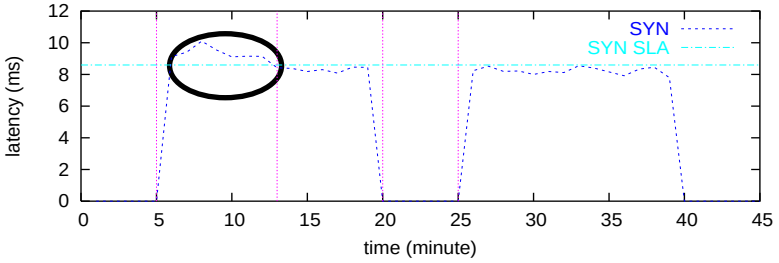




- With throttling



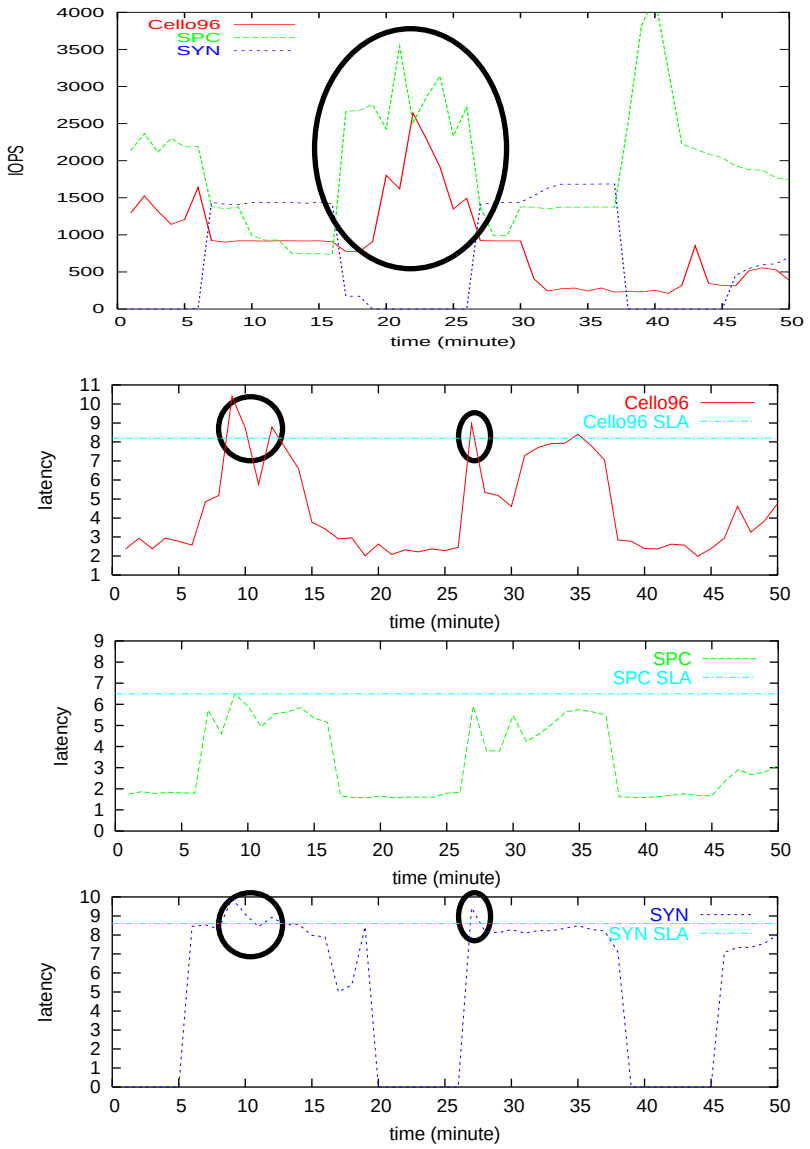



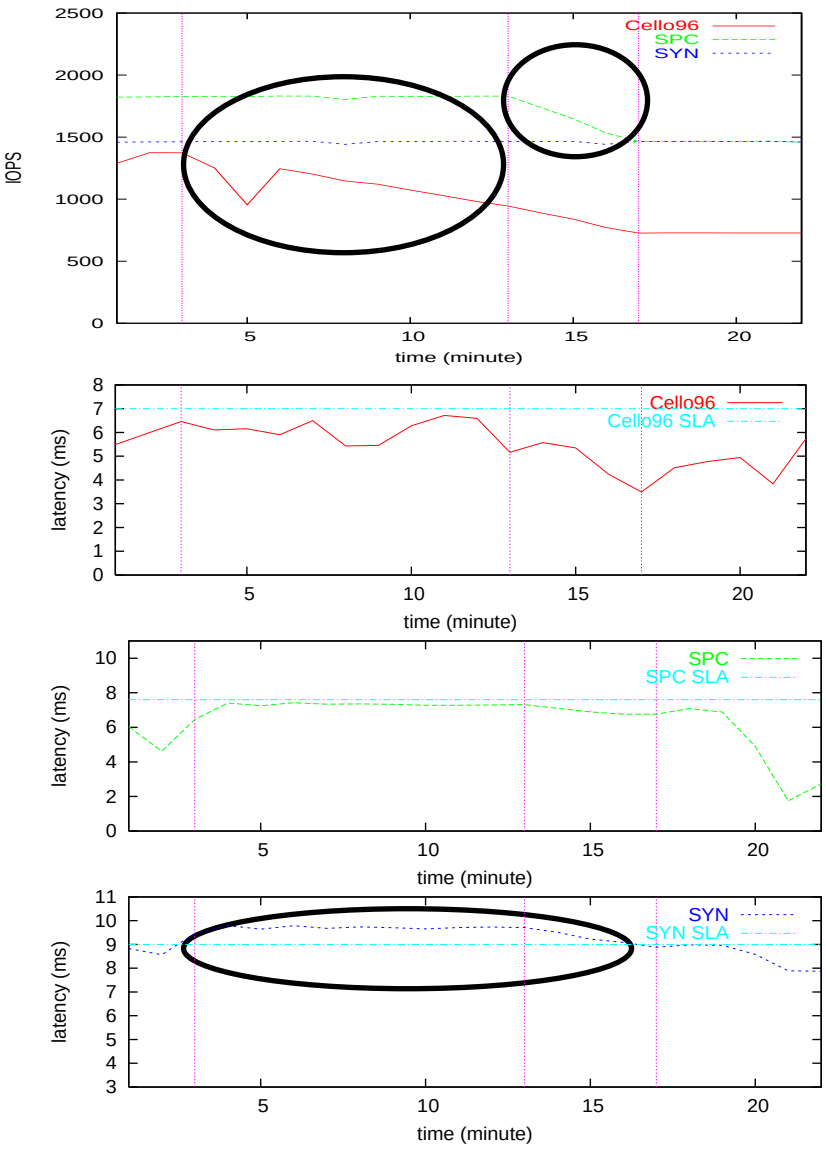



# Real-world Trace Replay

- With periodic un-throttling

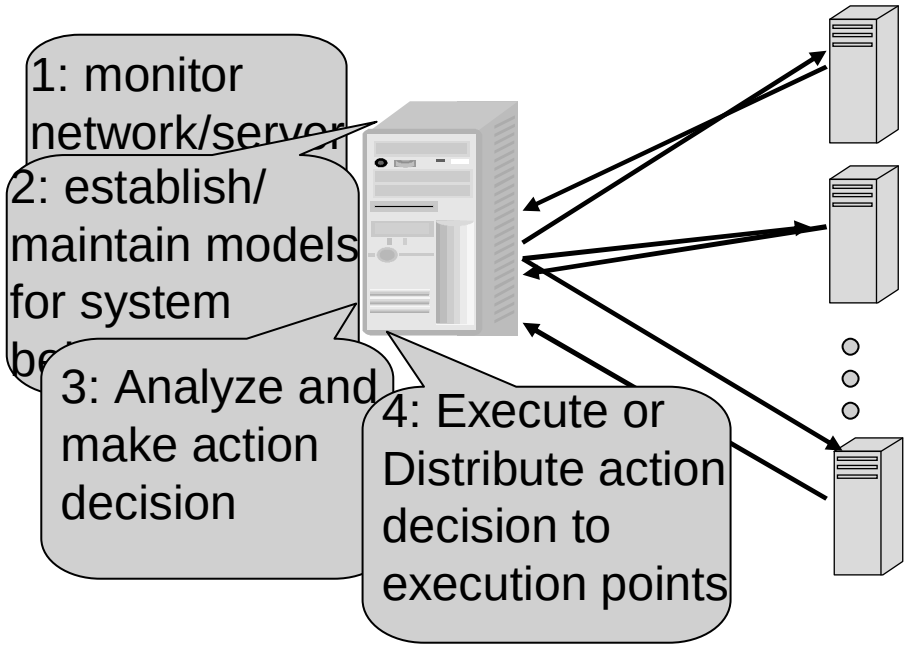



- Handling system changes



# Other system management scenarios?

- Automate the observe-analyze-act loop for other self-management scenarios
- Example: CHAMELEON for network applications
  - Example:  A proxy in front of server farm



# Future Work

- Better methods to improve model accuracy
- More general constraint solver
- Combining with other actions
- CHAMELEON in other scenarios
- CHAMELEON for reliability and failure

# References


- L. Yin, S. Uttamchandani, J. Palmer, R. Katz, G. Agha, “AUTOLOOP: Automated Action Selection in the ``Observe-Analyze-Act’’ Loop for Storage Systems”, submitted for publication, 2005
- S. Uttamchandani, L. Yin, G. Alvarez, J. Palmer, G. Agha, “CHAMELEON: a self-evovling, fully-adaptive resource arbitrator for storage systems”, to appear in USENIX Annual Technical Conference (USENIX’05), 2005
- S. Uttamchandani, K. Voruganti, S. Srinivasan, J. Palmer , D. Pease, “Polus: Growing Storage QoS Management beyond a 4-year old kid”, 3rd USENIX Conference on File and Storage Technologies (FAST’04), 2004

*Questions?*

**Email: yinli@eecs.berkeley.edu**