Managing Web server performance with AutoTune agents

by Y. Diao, J.L. Hellerstein, S. Parekh, J.P. Bigus

IBM SYSTEM JOURNAL, VOL 42, NO 1, 2003

Seoul National University
Computer Science and Engineering
DCSLAB
Sophal HONG
2016-12-08

Outline

- Introduction
- Apache web server and performance tuning
- Server self-tuning with AutoTune agents
    - Modeling Agent
    - Run-Time Control Agent
    - Controller Design Agent
- Experimental assessment
- Conclusions

Introduction

- Managing the performance of e-commerce sites is challenging
  - Site content changes frequently
  - Dynamically varying workloads
  - Some applications of control theory to computing systems include
    - flow and congestion control, differentiated caching and web service, multimedia streaming, web server performance, e-mail server control

- To maintain good performance
  - System administrators must tune their information technology environment
  - Manual effort can be time consuming and error-prone, and requires highly skilled, making it costly

Introduction

- All applications provide a degree of autonomic behavior by providing algorithms
  - to automatically control some aspect of a computing system’s operation

- In this paper…
  - proposing an **agent-based** solution
    - Automates the ongoing system tuning
    - Automatically designs an appropriate tuning mechanism for the target system

Apache Web server and performance tuning

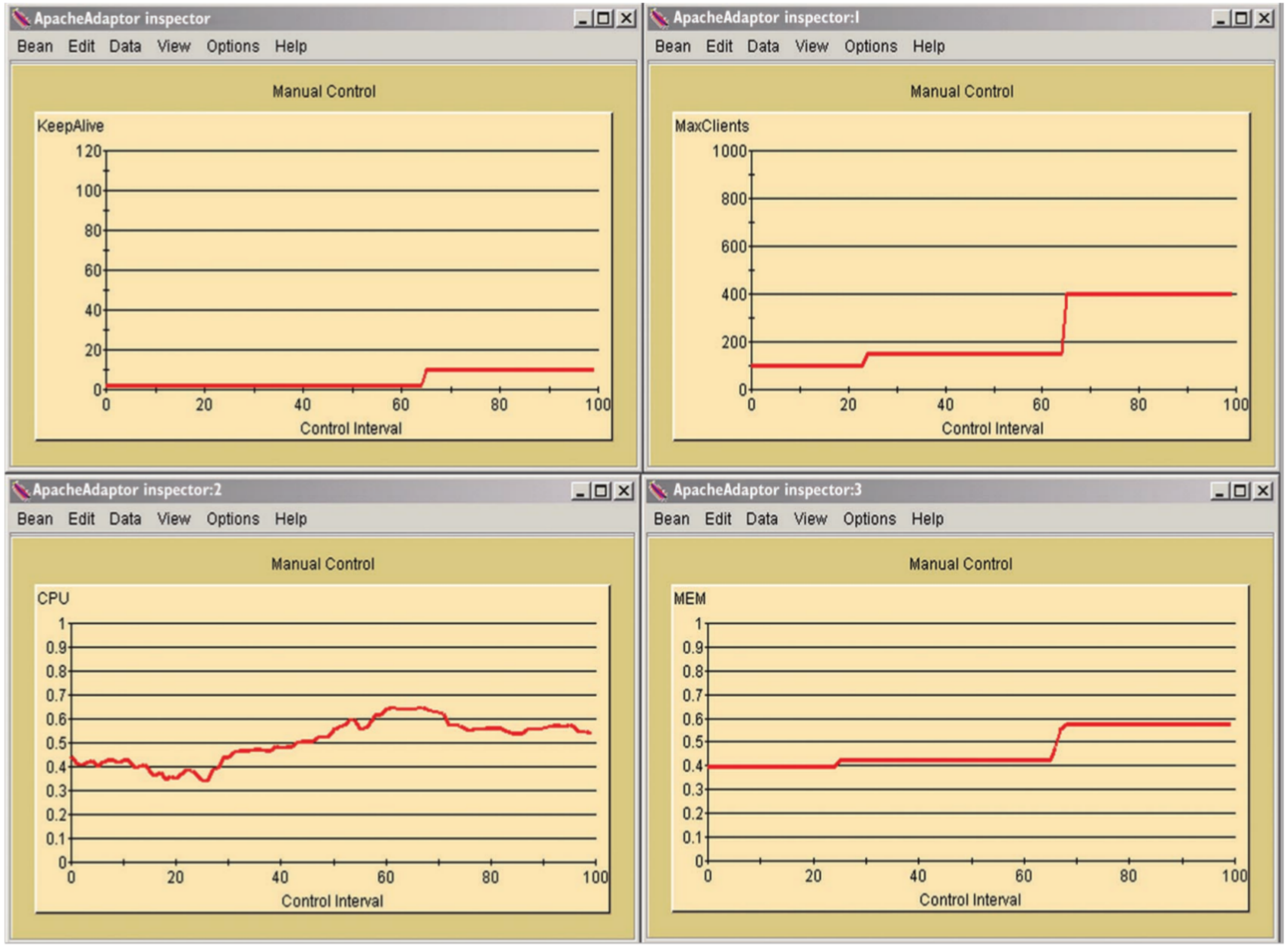
- Apache v1.3.x of the server on UNIX is structured as
  - One master process: monitors the health of the worker processes and manages their creation and destruction.
  - A pool of worker processes: responsible for communicating with Web clients and generating responses.
    - One worker process can handle at most one connection at a time.
    - Worker processes cycle through three states: idle, waiting, busy

Apache Web server and performance tuning

- The application-level tuning parameters in Apache Web server
  - MaxClients: The number of simultaneous requests that will be served
  - KeepAlive: Whether or not to allow persistent connections

- Administrator must operate indirectly by adjusting tuning parameters
  - Increasing MaxClients: Increasing both CPU and Memory utilizations
  - Decreasing keepAlive: Allows worker process to be more active.
    - Directly results in higher CPU utilization
    - Indirectly increases memory utilization (more clients can connect).

Results of manually tuning the Apache Web server

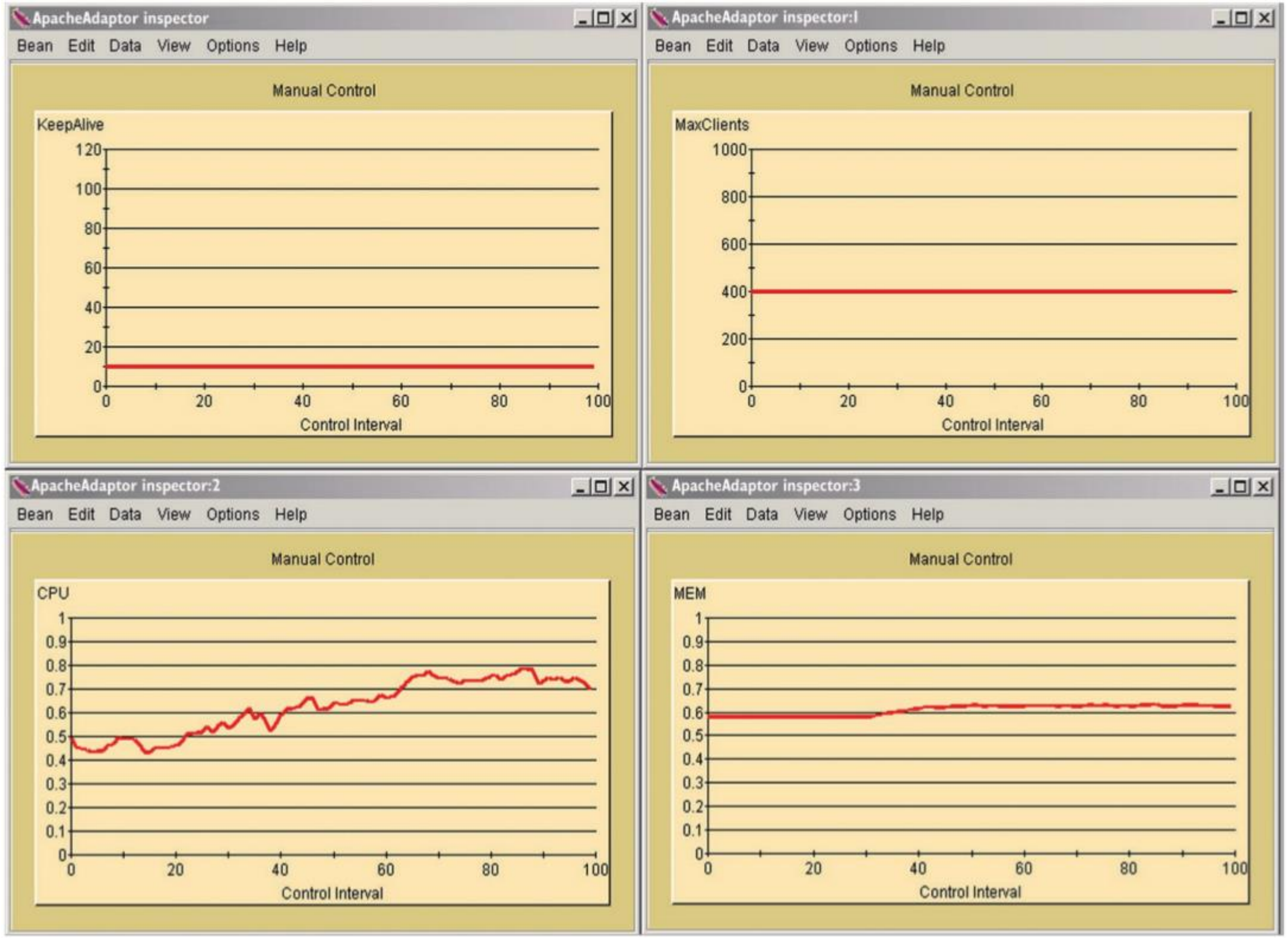


Suppose the desired
CPU level = 0.5
Memory = 0.6

Y-axis: measured values
X-axis: time (second)

Result:
MaxClients: 400
KeepAlive: 10

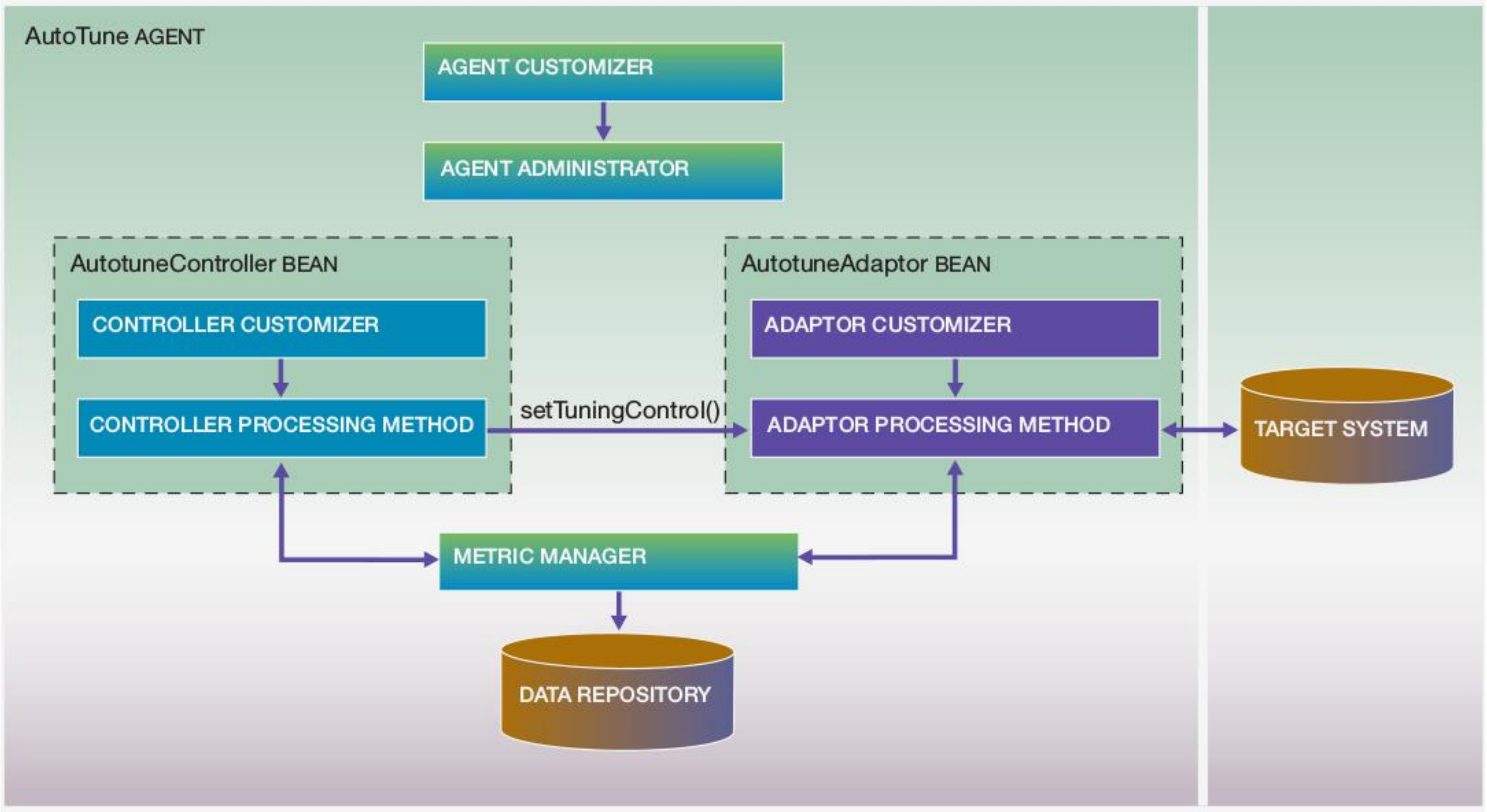
Effects of Dynamics Workloads



- A change of Web site contents also affect the CPU and memory usage per request and
- also require different MaxClients and KeepAlive setting.

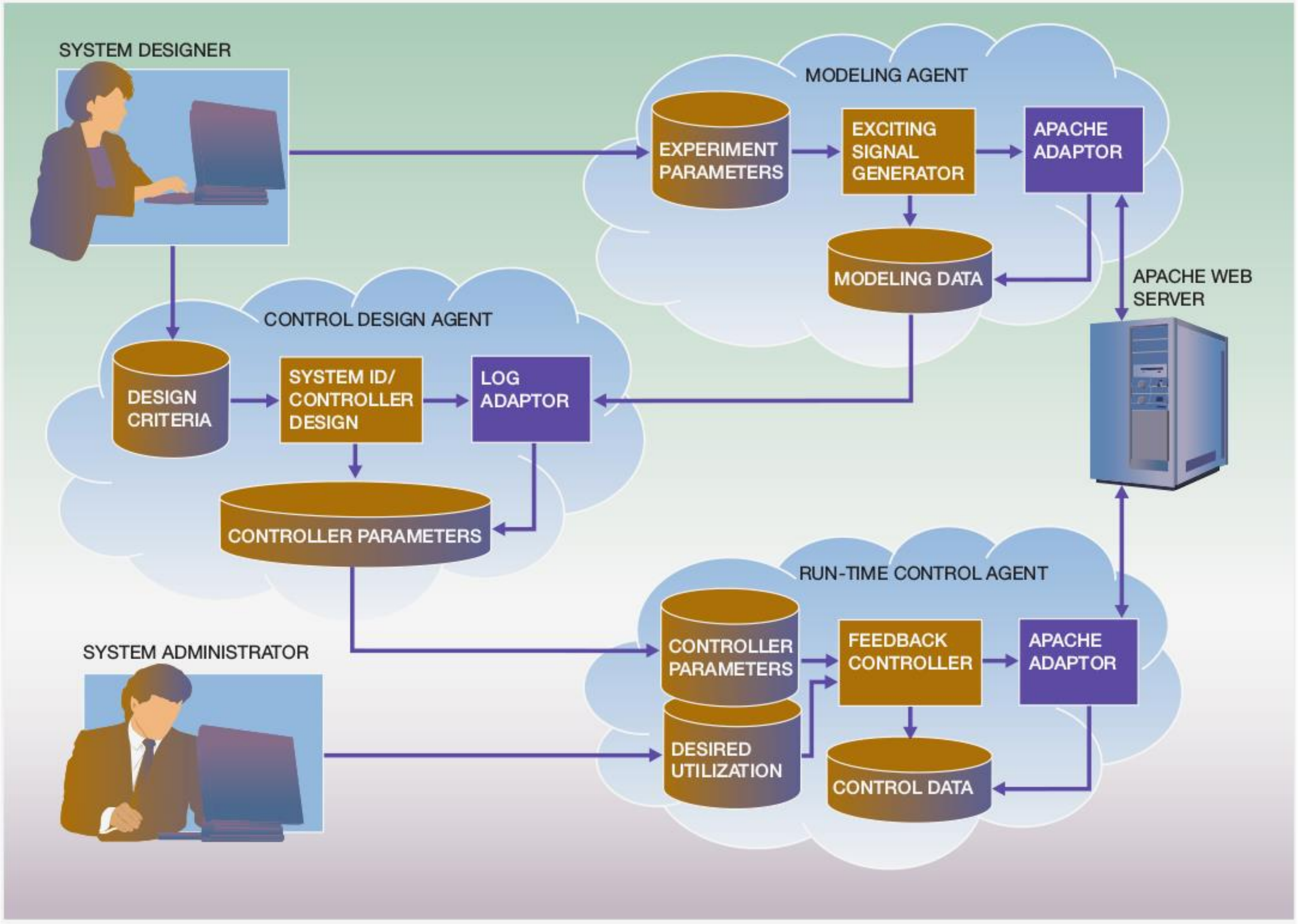
- Need AutoTune agents: to automate the adjustment of the MaxClients and KeepAlive values
- Both at system start-up and on an on going basis in response to changing workload

Server self-tuning with AutoTune Agents



- Solution:
  - Multiple agents
    - Automate the entire methodology of controller design
    - Perform the on-line system control
- These agents are implemented using the ABLE (Agent Building and Learning Environment)
  - Java**based toolkit
  - ABLE: provides a comprehensive library of intelligent reasoning and learning components

Architecture of the AutoTune agent



- Modeling and design: performed in a “testing” (or nonproduction) mode
- Run-time control: active when the system is “live” (Production mode)

Modeling Agent

- Modeling agent: A good design for the feedback controller relies on a mathematical model of the target system.
- Quantifying the relationship between the tuning parameters and performance metrics

- 2 x 2 matrics A and B
- Include modeling parameters
- Can be identified using the least squares method

$$\begin{bmatrix} \mathrm{CPU}_{k+1} \\ \mathrm{MEM}_{k+1} \end{bmatrix} = A \cdot \begin{bmatrix} \mathrm{CPU}_{k} \\ \mathrm{MEM}_{k} \end{bmatrix} + B \cdot \begin{bmatrix} \mathrm{KeepAlive}_{k} \\ \mathrm{MaxClients}_{k} \end{bmatrix}$$

Controller Design Agent

- To design the parameters
- Choosing the controller parameters based on minimizing the following quadratic cost function:

$$J(K_P, K_I) = \sum_{k=1}^{\infty} [e_{\mathrm{CPU},k}\ e_{\mathrm{MEM},k}\ v_{\mathrm{CPU},k}\ v_{\mathrm{MEM},k}] \cdot Q \cdot \begin{bmatrix} e_{\mathrm{CPU},k} \\ e_{\mathrm{MEM},k} \\ v_{\mathrm{CPU},k} \\ v_{\mathrm{MEM},k} \end{bmatrix} + [\mathrm{KeepAlive}_k\ \mathrm{MaxClients}_k] \cdot R \cdot \begin{bmatrix} \mathrm{KeepAlive}_k \\ \mathrm{MaxClients}_k \end{bmatrix}$$

$$v_{\mathrm{CPU},k} = \sum_{j=1}^{k-1} e_{\mathrm{CPU},j}$$

$$R = diag(r_1, r_2)$$

$$Q = diag(q_1, q_2, q_3, q_4)$$

- Q and R perform some scaling functions in addition to determining a trade-off between control error and control variability

Run-time Control Agent

- Implements a state feedback controller
  - To make control decisions based on feedback of errors

$$\begin{bmatrix} \mathrm{KeepAlive}_k \\ \mathrm{MaxClients}_k \end{bmatrix} = K_P \cdot \begin{bmatrix} e_{\mathrm{CPU},k} \\ e_{\mathrm{MEM},k} \end{bmatrix} + K_I \cdot \sum_{j=1}^{k-1} \begin{bmatrix} e_{\mathrm{CPU},j} \\ e_{\mathrm{MEM},j} \end{bmatrix}$$

  - K_P: Proportional control gain for fast response
  - K_I: Integral control gain for removing steading-state error

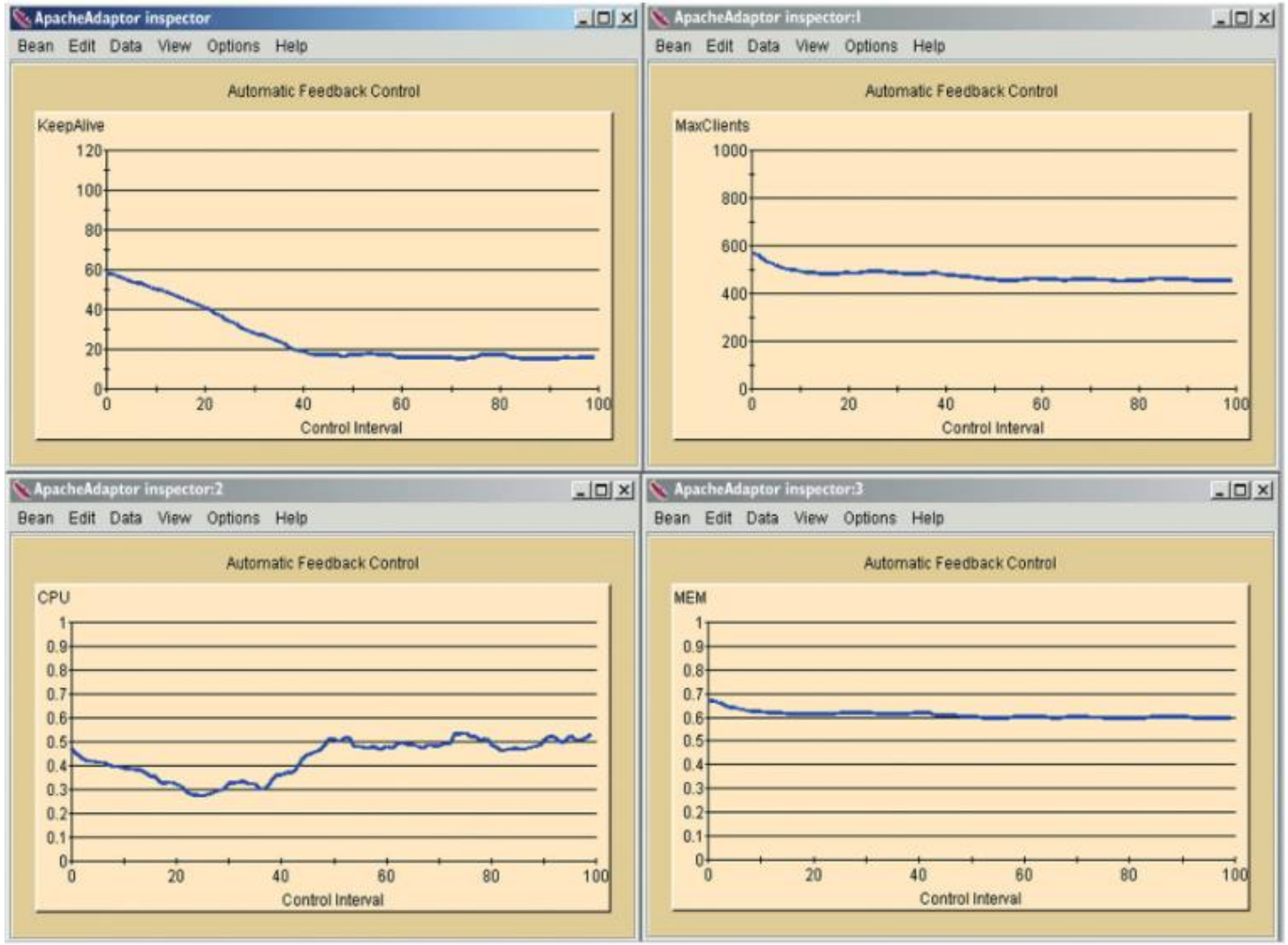
Experimental Environment

- Sever machine: Linux 2.2.16, Apache HTTP server v1.3.19
- One or more client machines:
  - Workload generator: WAGON (Web trAffic Generator and beNchmark)
  - File access distribution: Web Stone

- Dynamic workload
  - Web pages generated through CGI (Common Gateway Interface)
  - The session following a Poisson distribution
  - A rate of 10 sessions per second

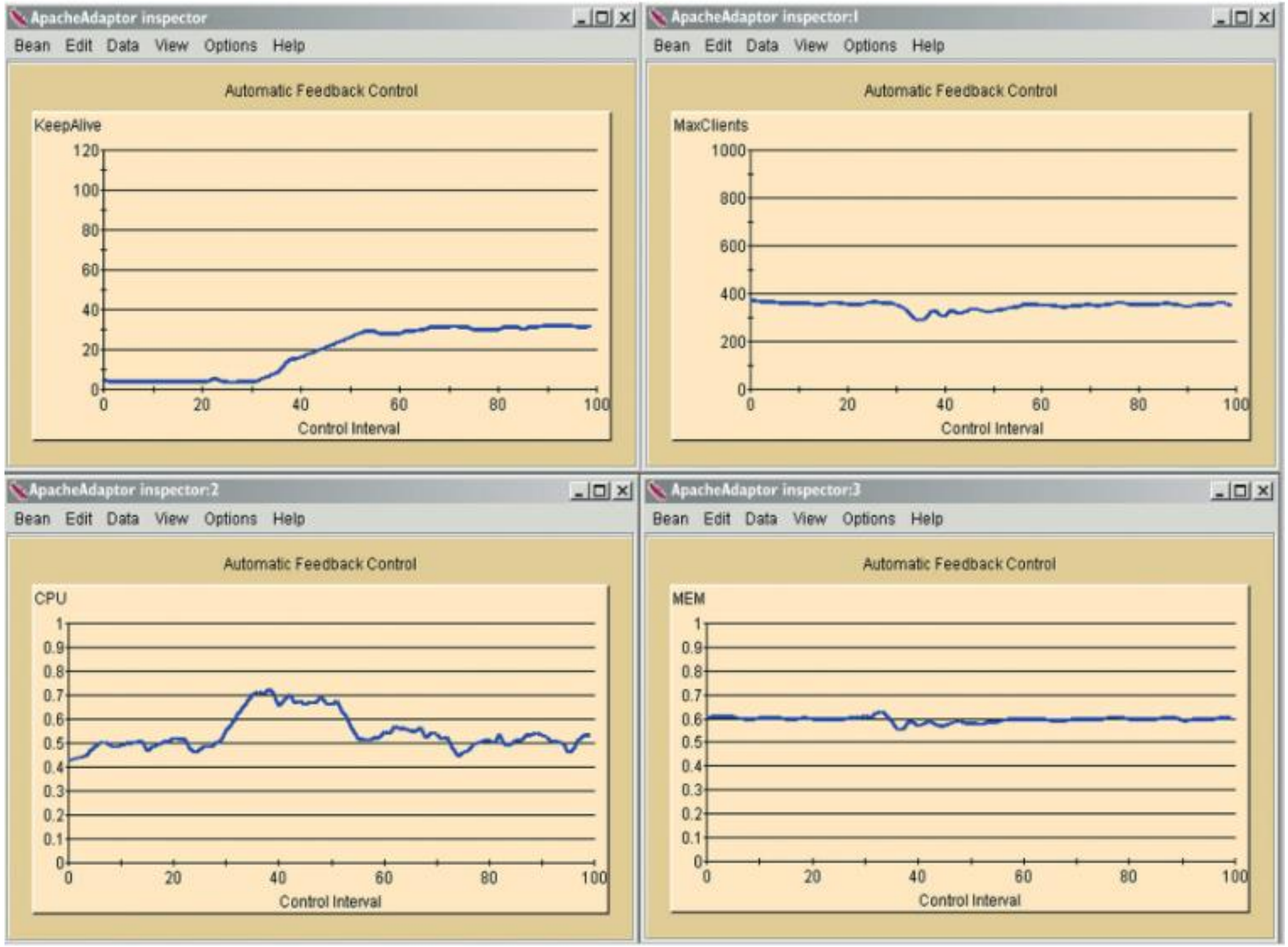
Experimental Assessment

- Results of automatically tuning the Apache Web server



Experimental Assessment

- Performance of the AutoTune controller for the Apache Web server under dynamic workload



Conclusions

- Proposing an agent-based solution
  - Automating the ongoing system tuning
  - Automatically designing an appropriate tuning mechanism for the target system

- Experiments showing
  - The feedback-driven controller to be robust and adaptable to situations other than the one for which it was designed