Managing Web Server Performance with AutoTune Agents

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

Presented by
Sunghwan Jung
Contents

1 Introduction

2 Apache Web Server and Performance Tuning

3 Server Self-Tuning with AutoTune Agents

4 Experimental Assessment
Introduction

- Managing the Performance of E-Commerce Sites Is Challenging
  - Site content changes frequently
    - Dynamically varying workloads

- System Administrators Must Tune Their Information Technology Environment
  - Manual effort can be time consuming and error-prone, and requires highly skilled
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
In This Paper…

- They propose an agent-based solution
  - Automates the ongoing system tuning
  - Automatically designs an appropriate tuning mechanism for the target system
- They illustrated as the context of managing a web server
  - Modeling the behavior of an Apache Web Server
  - Designing the feedback control law
  - Adjusting the server parameters in response to workload variations
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

Above are significant parameters to effect CPU and memory utilization.
- 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)
Apache Web Server and Performance Tuning (Cont’d)

- Results of manually tuning the Apache web server
Effects of Dynamic Workloads on Manually Tuned Apache Server
Server Self-Tuning with AutoTune Agents

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

- System Designer
- System Administrator
- Modeling Agent
- Experiment Parameters
- Exciting Signal Generator
- Modeling Data
- Apache Adaptor
- Apache Web Server
- Design Criteria
- Controller Parameters
- Log Adaptor
- Control Design Agent
- Run-Time Control Agent
- Feedback Controller
- Desired Utilization
- Control Data
Standard control-theoretic methodology for the design process: System modeling, controller design

- The modeling and design phases: In “testing” mode
- The run-time control: In “live” mode

Modeling agent

\[
\begin{bmatrix}
\text{CPU}_{k+1} \\
\text{MEM}_{k+1}
\end{bmatrix} = A \cdot \begin{bmatrix}
\text{CPU}_k \\
\text{MEM}_k
\end{bmatrix} + B \cdot \begin{bmatrix}
\text{KeepAlive}_k \\
\text{MaxClients}_k
\end{bmatrix}
\]

- A, B: 2 x 2 matrices
  - Include modeling parameters
  - Can be identified using the least squares method
Run-time control agent

\[
\begin{bmatrix}
\text{KeepAlive}_k \\
\text{MaxClients}_k
\end{bmatrix} = K_P \cdot \begin{bmatrix}
\text{CPU}_k \\
\text{MEM}_k
\end{bmatrix} + K_I \cdot \sum_{j=1}^{k-1} \begin{bmatrix}
\text{CPU}_j \\
\text{MEM}_j
\end{bmatrix}
\]

- **K_P**: Proportional control gain for fast response
- **K_I**: Integral control gain for removing steady-state error
Controller design agent

- In particular, the controller design agent chooses the controller parameters based on minimizing the following quadratic cost function:

\[
J(K_p, K_I) = \sum_{k=1}^{\infty} \left[ e_{CPU,k} e_{MEM,k} v_{CPU,k} v_{MEM,k} \right] \cdot Q \cdot \begin{bmatrix} e_{CPU,k} \\ e_{MEM,k} \\ v_{CPU,k} \\ v_{MEM,k} \end{bmatrix} + \left[ \text{KeepAlive}_k \text{MaxClients}_k \right] \cdot R \cdot \begin{bmatrix} \text{KeepAlive}_k \\ \text{MaxClients}_k \end{bmatrix}
\]
Riccati equation is allow us to compute the optimal $K_P$ and $K_I$ that minimize the above cost function.

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

- Environment: Linux 2.2.16, Apache 1.3.19
  - Workload generator: WAGON (Web trAffic GeneratOr and beNchmark)
  - File access distribution: Web Stone

- Dynamic workload
  - Web pages generated through CGI
  - The session following a Poisson distribution
  - A rate of 10 sessions per second
Experimental Assessment (Cont’d)

- Results of autonomically tuning the Apache Web server
Experimental Assessment (Cont’d)

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