Self-Tuning Memory Management of A Database System

Yixin Diao
diao@us.ibm.com

IM 2009 Tutorial: Recent Advances in the Application of Control Theory to Network and Service Management
DB2 Self-Tuning Memory Management

- **Technical problems**
  - Large systems with varying workloads and many configuration parameters
  - Autonomic computing: systems self-management

- **Challenges from systems aspects**
  - Heterogeneous memory pools
  - Dissimilar usage characteristics

- **Challenges from control aspects**
  - Adaptation and self-design
  - Reliability and robustness
Load Balancing for Database Memory

Load Balancing
- Fairness $\rightarrow$ optimal?
- Common measured output?

OLTP

\[
x_i = p_i \left(1 - e^{-q_i u_i}\right)
\]

\[
y_i = \frac{dx_i}{du_i} = p_i q_i e^{-q_i u_i}
\]
Constrained Optimization and Regulatory Control

**Constrained Optimization**

\[ J = f(u_1, u_2, \ldots, u_N) \]

\[ g(u_1, u_2, \ldots, u_N) = \sum_{i=1}^{N} u_i - U = 0 \]

\[ h(u_1, u_2, \ldots, u_N) = u_i - b_i \geq 0 \]

**Karush-Kuhn-Tucker conditions**

\[ L = f(u_1, u_2, \ldots, u_N) + \lambda g(u_1, u_2, \ldots, u_N) \]

\[ + \mu h(u_1, u_2, \ldots, u_N) \]

\[ \frac{\partial L}{\partial u_i} = \frac{\partial f}{\partial u_i} + \lambda + \mu_i = 0 \]

\[ \mu_i = 0 \text{ if } u_i > b_i \; ; \; \mu_i > 0 \text{ if } u_i = b_i \]

**Regulatory Control**

\[ \frac{\partial f}{\partial u_i} - \frac{1}{N} \sum_{j=1}^{N} \frac{\partial f}{\partial u_j} = 0 \]
Dynamic State Feedback Controller

- State space model
  \[ y(k + 1) = Ay(k) + B(u(k) + d^I(k)) \]

- Control error
  \[ e(k) = \left( \frac{1}{N} 1_{N,N} - I \right) (y(k) + d^O(k)) \]

- Integral control error
  \[ e_I(k + 1) = e_I(k) + e(k) \]

- Feedback control law
  \[ u(k) = K_p e(k) + K_I e_I(k) \]
Incorporating Const of Control into Controller Design

Major cost: write dirty, move memory, victimize hot

Linear quadratic regulation (LQR)

\[ J = \sum [e^T(k) e^T_1(k)] Q [e^T(k) e^T_1(k)]^T + u^T(k) R u(k) \]

Define Q and R regarding to performance

- Cost of transient load imbalances
- Cost of changing resource allocations
Adaptive Controller Design

Local linear model

\[ y_i(k+1) = b_i(k)u_i(k) \]

Decentralized integral control

\[ u_i(k+1) = u_i(k) - \frac{1-p}{b_i(k)} \left( y_i(k) - \frac{1}{N} \sum_{j=1}^{N} y_j(k) \right) \]
Experimental Assessment

- **OLTP workload: multiple (20) buffer pools**

- **DSS workload: various query lengths**

- **DSS workload: index drop**

  - **Execution time for Query 21 (10 stream avg)**

  - **ConfigAdvisor settings**
    - Ts = 26342s
  - **STMM tuning**
    - Ts = 10680s

  - > 2x improvement

  - Reduce 63%

  - avg = 959
  - avg = 2285
  - avg = 6206

- Increase TP from ~100 to ~250
## Comparing Control and Optimization Techniques

### Control-based approach

**Local linear model**

\[ y_i(k+1) = b_i(k)u_i(k) \]

**Decentralized integral control**

\[ u_i(k+1) = u_i(k) - \frac{1-p}{b_i(k)} \left( y_i(k) - \frac{1}{N} \sum_{j=1}^{N} y_j(k) \right) \]

**Constraint enforcement (projection method)**

\[ d_i(k) = \frac{u_i(k)}{\sum_{j=1}^{N} u_j(k)} \left( U - \sum_{j=1}^{N} u_j \right) + u_i \]

- Less dependence on the model

### Optimization-based approach

**Gradient method**

\[ u(k+1) = u(k) + \lambda(k)p(k) \]

**Projected gradient (quasi-Newton)**

\[
\begin{align*}
p(k) &= -(H(k) - H(k)A^T(k)(A(k)H(k)A^T(k))^{-1}A(k)H(k))y(k) \\
\end{align*}
\]

**Step length (modified Armijo rule)**

\[ \nabla f(u(k) + \lambda(k)p(k)) + p(k) \geq c_1 \nabla f(u(k))^T p(k) \]

- Strictly applies constrained optimization

### Similarity in a simplified scenario

**A(k)**

\[
A(k) = \begin{bmatrix}
1 & 1 & \cdots & 1
\end{bmatrix}
\]

**u(k)**

\[
u(k) = \begin{bmatrix}
\frac{1}{N} & 0 & \cdots & 0 \\
0 & \frac{1}{N} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{1}{N}
\end{bmatrix}
\]

**\(u_i(k+1) = u_i(k) - \frac{\lambda(k)}{s_i(k)} \left( y_i(k) - \frac{1}{\sum_{j=1}^{N} s_j(k)} \sum_{j=1}^{N} y_j(k) \right) \)**

### Differences in design considerations

- “Pure” average vs. convex sum
- Pole location vs. Armijo rule
- Steady-state gain vs. Hessian matrix
Simulation Study: Comparison with Optimization Approach

<table>
<thead>
<tr>
<th>Control-based approach</th>
<th>Optimization-based approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory size</strong></td>
<td><img src="image" alt="Graph" /></td>
</tr>
<tr>
<td><strong>Total saved time</strong></td>
<td><img src="image" alt="Graph" /></td>
</tr>
<tr>
<td><strong>Control intervals</strong></td>
<td><img src="image" alt="Graph" /></td>
</tr>
</tbody>
</table>

**Without noise (single run):**
- More robust and better uncertainty management

**Effect of noise (multiple runs):**
- Faster convergence, but more sensitive to noise
Summary

- **DB2 self-tuning memory management**
  - Interconnection, heterogeneity, adaptation and robustness, cost of control
  - Constrained optimization with a linear feedback controller
  - Experimental assessment for OLTP and DSS workloads