Towards Model-based Management of Database Fragmentation

Asim Ali, Abdelkarim Erradi, Rashid Hadjidj
Qatar University

Rui Jia, Sherif Abdelwahed
Mississippi State University
Outline

- Introduction
- Model-based Control Framework
- Control Approach
- Case Study
- Conclusion
Introduction
Database Fragmentation

- Databases are integral parts of systems such as:
  - Social Networks
  - Online trading
  - E-Banking
Database Fragmentation

- Clustered indexes
- Cause: large number of write, delete and update operations
- Internal and external
Database Fragmentation

Internal and External Fragmentation

Index leaf level after random inserts/deletes

Red arrow is the allocation order
Black arrows are following the logical order

Image courtesy: SQLSkills.com
Fragmentation Impacts

- **Fragmented Database**
  - Internal
  - Extent
  - Disk

- **Increased Resource Usage**
  - CPU usage
  - Memory Usage
  - Disk IO

- **Degraded QoS Parameters**
  - Response time
  - Throughput
  - Availability

- **Causes Financial Overhead**
  - Financial Penalties due to SLA violations

**Example**: Decreasing throughput with increasing fragmentation levels
Defragmentation Approaches

Conventional
- Human expertise
- Rules of thumb decide the time and indexes to defragment
- Does not consider impact during defrag.
- Potentially suboptimal decision
- No model is required

Model-based
- Autonomous controller
- Model-based decision
- Considers impact on system performance
- Decision is optimal based on model
- System models are required
Contribution

- A fragmentation management framework
  - Model-based
  - Considers dynamic SLAs
  - Fragmentation detection
  - Performance optimization

- Autonomic controller
Model-based Control Framework
Model-based Control Framework

Key components

- Environment (workload)
- Managed System
- Fault Diagnosis
- Performance Controller
Model-based Control Framework

Overview of the Framework
Model-based Control Framework

- Workload variations
  - Arrival rate (TPM)
- Fault variations
  - Fragmentation levels
- Monitored Parameters
  - CPU usage, memory usage, disk IO, response time, throughput, etc.
- Data analysis: curve fitting tools
Model-based Control Framework

Model Development Process

- Analyzing historic traces
- Simulations
- Relationships
- Benchmark workload
  - Based on OLTP workload configurations
  - Performance measured based on read operations
- Fragmentation injection script
Model-based Control Framework

- Model-based Fault Detection
  - Model based fault detection
    - Monitoring system -> Current performance
    - System model -> Expected performance
    - Threshold difference
  - Fault signature (parameters)
    - Vector of system and application parameters
    - Magnitude of deviation -> Fault intensity
Control Approach
Control Input

Available Control Options

<table>
<thead>
<tr>
<th>Management Actions</th>
<th>Recovery Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing Database Buffer Sizes (e.g. 128MB, 256MB, and etc.)</td>
<td>Offline Database Rebuilding</td>
</tr>
<tr>
<td></td>
<td>Online Database Reorganization</td>
</tr>
</tbody>
</table>

Criteria to Select the Most Optimal Control Option?
## Selection Criteria

<table>
<thead>
<tr>
<th>Management Actions</th>
<th>Benefits</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing Database Buffer Sizes (e.g. 128MB, 256MB, and etc.)</td>
<td>Bigger Buffer = Less I/O Delay</td>
<td>Bigger Buffer = Potential Resource Shortage</td>
</tr>
</tbody>
</table>

Each control option has both benefits and drawbacks: a tradeoff
## Control Input

### Selection Criteria

<table>
<thead>
<tr>
<th>Recovery Actions</th>
<th>Benefits</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline Database Rebuilding</td>
<td>Fast Recovery</td>
<td>Service Blackout</td>
</tr>
<tr>
<td>Online Database Reorganization</td>
<td>Service Available</td>
<td>Slow Recovery</td>
</tr>
</tbody>
</table>

Each control option has both benefits and drawbacks: a tradeoff.
Controller Structure

Controller Design

Controller Structure Diagram:
- Performance Optimizer
- Utility function
- Cost Model
- System Model
- SLA Manager
- Fault Diagnosis
- ARIMA Model
- Predicted System Parameters
- Diagnosis Information
- Expected Workload
- Control Options
- Environment Input
- Control Input
- Utility Value
- Target QoS Parameters

The 8th International Workshop on Feedback Computing
San Jose, CA, USA
June 25th, 2013
Utility Cost Model

- Quantify the benefits and drawbacks of a control option
- Evaluate the system performance in terms of QoS and SLA violation
- Treat management and recovery actions separately
Utility Function

For management actions

\[ J_M(i) = \left[ \frac{r^M_x(i) - r^*}{r^*} + \frac{\theta^* - \theta^M_x(i)}{\theta^*} \right] \times T_x^M(i) \]

\[ + \left[ \frac{\hat{r}^M(i) - r^M(i-1)}{r^M(i-1)} + \frac{\hat{\theta}^M(i) - \theta^M(i-1)}{\theta^M(i-1)} \right] \times T \]
Utility Function

For recovery actions

\[ J_R(i) = \left[ \frac{\hat{r}^R(i) - r^*}{r^*} + \frac{\theta^* - \hat{\theta}^R(i)}{\theta^*} \right] \times T \]
Predictive Control

Predictive Control: Select the Minimum-cost Control Plan

Optimization Objective:

1. Minimize $\sum_{i=k}^{k+m-1} [W_M(i)J_M(i)]$

2. Minimize $W_R(j) \sum_{i=k}^{k+n-1} J_R(i) / n$
Predictive Control Algorithm

- Start
- Input $x(k)$, $\lambda(k)$, $U_R$, $U_M$, $f(k)$
- Is Fault Detected? No → Use $H_M$ and Traditional LLC to Find $J_{normal}$
- Yes → Enumerate $U_R$, Estimate $x(k+1)$ and $\lambda(k+1)$
- Calculate and store $J_{fault}$
- Is Recovery Horizon $H_R$ Met? No → Select the Recovery Action with minimum $J_{fault}$
- Yes → Select the Management Action with minimum $J_{normal}$
- End
Case Study
Case Study

Experimental Setup
Case Study

Example Simulation Results
## System Parameters

- Fragmentation detection using average response times

<table>
<thead>
<tr>
<th>FL</th>
<th>Average query response time (ms)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std. Dev.</td>
<td>% variation</td>
</tr>
<tr>
<td>0</td>
<td>189</td>
<td>8</td>
<td>0.00</td>
</tr>
<tr>
<td>14</td>
<td>192</td>
<td>5</td>
<td>1.59</td>
</tr>
<tr>
<td>21</td>
<td>202</td>
<td>8</td>
<td>6.88</td>
</tr>
<tr>
<td>33</td>
<td>238</td>
<td>16</td>
<td>25.93</td>
</tr>
<tr>
<td>42</td>
<td>275</td>
<td>19</td>
<td>45.50</td>
</tr>
<tr>
<td>51</td>
<td>380</td>
<td>30</td>
<td>101.06</td>
</tr>
<tr>
<td>60</td>
<td>425</td>
<td>31</td>
<td>124.87</td>
</tr>
<tr>
<td>70</td>
<td>466</td>
<td>36</td>
<td>146.56</td>
</tr>
<tr>
<td>80</td>
<td>501</td>
<td>75</td>
<td>165.08</td>
</tr>
<tr>
<td>90</td>
<td>551</td>
<td>50</td>
<td>191.53</td>
</tr>
<tr>
<td>100</td>
<td>552</td>
<td>46</td>
<td>192.06</td>
</tr>
<tr>
<td>110</td>
<td>655</td>
<td>56</td>
<td>246.56</td>
</tr>
</tbody>
</table>
# System Parameters

<table>
<thead>
<tr>
<th>Arrival rate</th>
<th>Throughput</th>
<th>Avg. Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda &lt; 350$</td>
<td>$0.9 \times \lambda + 8.9$</td>
<td>224</td>
</tr>
<tr>
<td>$\lambda \geq 350$</td>
<td>$0.01 \times \lambda + 318.1$</td>
<td>189</td>
</tr>
</tbody>
</table>

System model for fragmentation level 0
Conclusion

- The system model is developed to facilitate the fault management
- The detection method can provide the level of fragmentation
- The model-based fault management framework has been applied to solve the database fragmentation problem
Future Work

- Workload: add composition variation
- A fully autonomic control process, with more
  - Recovery actions
  - Management actions
- Applied on a distributed database system
Thank you!