Online Scaling in a Highly Available DBMS

Svein Erik Bratsberg and Rune Humborstad
Clustra AS
Motivation

- **High availability**
  Max 2 min unavailability per year
- **Soft real-time response times**
- **Online scaling of data volume and transaction capacity**
- **Online schema evolution**
- **Commodity HW and open SW**
Characteristics of Clustra

- Function shipping (vs. data shipping)
- Shared nothing architecture
  - Multi-node
  - Messaging intensive
- Critical path entirely in memory
- Main-memory logging
- 2-phase commit for transaction decisions
- Storage using B-tree access
Online schema changes

- Must be online for availability
- Should have little impact on response times
- Online scaling of tables
- Online index creation
- SQL’s alter table
- Logical changes to tables, e.g. change type of attribute, table split and union, change primary key
Hardware architecture

Disaster unit A

Disaster unit B
Data declustering
Software architecture

- All nodes have the same software
- Makes management simpler
- Makes load balancing simpler

![Diagram of software architecture with nodes TCON, DICT, KERN, SQLC, and SQLX.]
Transaction processing

- 2PC protocol including a hot standby controller
- Prepare-to-commit may be piggybacked on slave commands
Transaction processing, cont'd
Logical logging

- Logical logging for replication
- LSNs attached to records, not to physical blocks
- Makes replication independent on physical storage and location
- Allows for non-mirrored declustering and redistribution of data

Log record:

<table>
<thead>
<tr>
<th>LSN</th>
<th>ReplId</th>
<th>Key</th>
<th>Changes</th>
</tr>
</thead>
</table>

Record:

| Key | LSN | Fields |
Data copy

Source

Target

Data stream
Log copy

Source

Target

Slow log stream

Realtime log stream
Fuzzy copy and redo

Node 0

RedoThread

LOG

Node 1

LOG SHIPPING (realtime)

LOG SHIPPING (slowlog)

DATA SHIPPING

Node 0

LOG

Node 1

LOG SHIPPING (realtime)

LOG SHIPPING (slowlog)

DATA SHIPPING

Node 0

LOG

Node 1

LOG SHIPPING (realtime)

LOG SHIPPING (slowlog)

DATA SHIPPING
Takeover after scaling

- Takeover in 2 milliseconds
- Takeover operations lock the old primaries and are replicated to the new fragment replicas
- When all takeover operations are received, the new distribution scheme may commit
- Access to old fragment replicas is rejected
Measurements - priorities

- Eager scaling: priority on scaling
- Medium scaling: equal priority
- Lazy scaling: priority on user transactions
<table>
<thead>
<tr>
<th>Client</th>
<th>Eager</th>
<th>Medium</th>
<th>Lazy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-read (high load)</td>
<td>5.26</td>
<td>2.51</td>
<td>1.98</td>
</tr>
<tr>
<td>1-upd (high load)</td>
<td>4.35</td>
<td>1.52</td>
<td>1.32</td>
</tr>
<tr>
<td>1-read (low load)</td>
<td>8.02</td>
<td>3.43</td>
<td>3.12</td>
</tr>
<tr>
<td>1-upd (low load)</td>
<td>5.46</td>
<td>2.63</td>
<td>2.47</td>
</tr>
<tr>
<td>Client</td>
<td>Eager</td>
<td>Medium</td>
<td>Lazy</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>1-read (high load)</td>
<td>0.40</td>
<td>0.59</td>
<td>0.59</td>
</tr>
<tr>
<td>1-upd (high load)</td>
<td>0.25</td>
<td>0.37</td>
<td>0.16</td>
</tr>
<tr>
<td>1-read (low load)</td>
<td>1.00</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>1-upd (low load)</td>
<td>0.95</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>