Scalability and Reliability Features of MySQL Connector/J

Mark Matthews
Principal Software Developer
MySQL Enterprise Tools

Todd Farmer
Senior Manager
MySQL Support
Load-balancing/Failover Use Cases

• Directly (jdbc:mysql:loadbalance:// URL prefix):
  – Clustered (NDB) or Multi-Master Replication deployment where both read and write operations are distributed across all hosts.

• Indirectly:
  – Replication deployments where read-only load can be distributed to slaves (jdbc:mysql:replication://)
  – Deployments requiring strong server affinity for specific server, failing over only when primary host is unavailable (jdbc:mysql://primary,failover-1,failover-2...)

ORACLE
Quick Load Balancer/Failover History Lesson

jdbc:mysql://primary,failover - since 3.0.2 (2002!)
   (upcoming in 5.1.13, this is a special case of
    jdbc:mysql://loadbalance under the hood)

   (since 5.1.11, the slaves are a jdbc:mysql:loadbalance:// under
    the hood)

Things to Consider

Requires understanding of your application and your production environment
Understanding Your Application

• How transparent should failure be to end users?
• How transparent can failure be to your codebase?
  • No framework can determine when it’s safe to retry failed operations
  • Easier to add this to existing frameworks than scatter throughout codebase
• The JDBC driver makes no attempt to make failure transparent
• It can make recovery transparent
Understanding Your Production Environment

• Key to making faults transparent to your users
• Key driver to setting many configuration values
• The more data you have about your application and your environment the better (and it changes over time!)
• Using DNS? Make sure you know everywhere it’s being cached!
Two Key Data Points
95th Percentile Connection Setup Time

- Red line: Not Cached
- Blue line: Cached
Worst Case Query Time

**Execution Time Statistics**

<table>
<thead>
<tr>
<th>Max Time</th>
<th>Min Time</th>
<th>Avg Time</th>
<th>Total Time</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.141</td>
<td>0.141</td>
<td>0.141</td>
<td>0.141</td>
<td>0.055</td>
</tr>
</tbody>
</table>

**Row Statistics**

<table>
<thead>
<tr>
<th>Max Rows</th>
<th>Min Rows</th>
<th>Avg Rows</th>
<th>Total Rows</th>
<th>Standard Deviation</th>
<th>Total Size</th>
<th>Max Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8 B</td>
<td>8 B</td>
</tr>
</tbody>
</table>

**Execution Summary**

<table>
<thead>
<tr>
<th>Executions</th>
<th>Errors</th>
<th>Warnings</th>
<th>Table Scans</th>
<th>Bad Index Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Time Span**

From Apr 8, 2010 10:18:22 AM to Apr 8, 2010 10:48:22 AM.

**First Seen**

Apr 8, 2010 10:30:07 AM
Drive the following configuration parameters...

connectTimeout - safety \times 95{\text{th}} \text{ %ile connect time}

socketTimeout - \text{safety} \times (\text{mean}_{\text{worst query}} + (3 \times \text{stddev}_{\text{worst query}}))

“safety” value? What are you comfortable with?

Remember, the defaults on most OS’s are in \textit{minutes}
Don’t Forget!

Set “loadBalanceBlacklistTimeout” to your best case outage window (in milliseconds)

(setting it to zero means hosts won’t be automatically blacklisted)
Additional Flexibility Coming in 5.1.13...
Controlling load-balance fail-over

- **Standard component**
  - Communication exceptions
  - SQLState starting with “08”
  - User-defined SQLState list match
  - User-defined Class list match

- **Custom component**
  - Implement LoadBalanceExceptionChecker interface
StandardLoadBalanceExceptionChecker properties

- **LoadBalanceSQLExceptionFailover**
  - Comma-delimited list of SQLState values
  - Will match with trailing wildcard
    - "08" will match "08000" and "08S01"

- **loadBalanceSQLExceptionSubclassFailover**
  - Comma-delimited list of fully-qualified class/interface names
  - Comparison using Class.isInstance(Exception)
Custom Exception Checker

- Must implement LoadBalanceExceptionChecker
  - shouldExceptionTriggerFailover(SQLException ex) method
  - NDBLoadBalanceExceptionChecker example:

```java
public class NdbLoadBalanceExceptionChecker extends StandardLoadBalanceExceptionChecker {

  public boolean shouldExceptionTriggerFailover(SQLException ex) {
    return super.shouldExceptionTriggerFailover(ex) || checkNdbException(ex);
  }

  private boolean checkNdbException(SQLException ex) {
    // Have to parse the message since most NDB errors are mapped to the same DEMC, sadly.
    return (ex.getMessage().startsWith("Lock wait timeout exceeded") ||
            (ex.getMessage().startsWith("Got temporary error")
            && ex.getMessage().endsWith("from NDB")));
  }
}
```
Managing load-balanced deployments

• Monitoring statistics
  – Active/total configured hosts
  – Active/total logical connection counts
  – Active/total physical connection counts
  – Total transaction count

• Manipulation of hosts
  – Introduce new host to running load-balanced deployment
  – Remove existing host gracefully
Introducing “ConnectionGroup”

- Logical named group of Connections within a single class loader
- Defined by new loadBalanceConnectionGroup property
- Must be defined in order to manage load-balanced deployment
- ConnectionGroups span Connection creation/allocation mechanism
Using ConnectionGroup to manage deployment

• Obtained via ConnectionGroupManager
  – getConnectionGroup(String groupName)
  – Comparable static methods directly on ConnectionGroupManager can be used with group name

• Statistics:
  – getActiveHostCount()
  – getActiveLogicalConnections()
  – getTotalLogicalConnections()
  – getActivePhysicalConnections()
  – getTotalPhysicalConnections()
  – getTotalTransactionCount()

• Manipulation
  – addHost(String hostPort, boolean forExistingConns)
  – removeHost(String hostPort, boolean killExistingConns)
Live demonstration

- JMX-based implementation
  - com.mysql.jdbc.jmx.LoadBalanceConnectionGroupManager
  - Requires Java 1.5+
  - Use -Dcom.sun.management.jmxremote JVM flag
Example - Homegrown horizontal partitioning with parallel query...

• Did we tell you that the load balancing algorithm is extensible?
• Use a ThreadLocal to hold the partition key
• Install your algorithm - “loadBalanceStrategy=…”
• Select the right host in pickNewConnection()
• For those icky cross-partition queries - StatementInterceptors to the rescue!
  • “Parallel Query”
Shard Querying Demo
Questions?
Resources

http://dev.mysql.com/usingmysql/java/
http://forums.mysql.com/

mark.matthews@oracle.com
todd.farmer@oracle.com
Thanks!
The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.