Performance/Scalability with JDBC, UCP & Oracle Database 12c

Nirmala Sundarappa
Principal Product Manager
Oracle JDBC & UCP
October 2nd, 2014
Safe Harbor Statement

The following 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.
Program Agenda

1. Connections
2. Hard and Soft Parses
3. Result Set Cache
4. LOB Operations
5. Array Operations/ Stored Procedures
6. Questions
Program Agenda

1 Connections

Hard and Soft Parses
Result Set Cache
LOB Operations
Array Operations/Stored Procedures
Questions
Connections

• Creating a DB connection is an **expensive** database operation
  – Spawn O/S process, network connection, several roundtrips
  – Associated database authentication and session creation
• Database Connections are expensive to tear down!
• Repeated Connection/Disconnection can be a huge scaling issue

**Calls for a Connection Strategy**
Connection Management Strategy

• Frequent short term connections
  – Use Universal Connection Pool (UCP)
• Very large middle tier deployment
  – Use Database Resident Connection Pool (DRCP)
• Single Instance and multiple databases
  – Use Multi-tenant Architecture
• Multiple instances, single database
  – RAC Architecture
Connections
Universal Connection Pool (UCP)

**Problem to Solve**
- Connection creation/thrashing in a database intensive applications requires a proper solution
- Need for a Client Side Connection Pool

**Solution**
- Universal Connection Pool (UCP) caches database connection objects
- Most convenient and works with most of the application containers such as Weblogic, Tomcat, Websphere, JBOSS etc.,
Connections
Universal Connection Pool (UCP) – Best Practices

- Built-in Failover mechanisms which makes it easier to use with RAC, RAC one etc.,
- The best configured connection pools result in higher throughput
- Set the below UCP properties for better performance

<table>
<thead>
<tr>
<th>UCP Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setInitialPoolSize(), setMinPoolSize(), setMaxPoolSize()</td>
<td>Set pool size based on database server resources (number of cores)</td>
</tr>
<tr>
<td>setTimeToLiveConnectionTimeout(), setAbandonConnectionTimeout()</td>
<td>Set the Connection Timeout appropriately</td>
</tr>
<tr>
<td>setMaxStatements()</td>
<td>Enable Statement Caching</td>
</tr>
</tbody>
</table>
## Connections

### Universal Connection Pool (UCP) – Other UCP Properties

<table>
<thead>
<tr>
<th>UCP Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setValidateConnectionOnBorrow(boolean)</code></td>
<td>Specifies whether or not connections are validated when the connection is borrowed from the connection pool</td>
</tr>
<tr>
<td><code>setMaxConnectionReuseTime(int)</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setMaxConnectionReuseCount(int)</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setInvalid()</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setConnectionWaitTimeout(int)</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setInactiveConnectionTimeout(int)</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setTimeoutCheckInterval(int)</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setConnectionHarvestable(boolean)</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setConnectionHarvestTriggerCount(int)</code></td>
<td>Connection related properties</td>
</tr>
<tr>
<td><code>setConnectionHarvestMaxCount(int)</code></td>
<td>Connection related properties</td>
</tr>
</tbody>
</table>
Connections
Database Resident Connection Pool (DRCP)

- **Problem to Solve**
  - eCommerce-style deployment
  - Many mid-tiers and multiple databases where each mid-tier with client side pool, each database hit by a large number of connections, most of which are idle
  - More Idle connections

- **Solution**
  - Database Resident Connection Pool (DRCP) is the solution
  - Supports only homogeneous connections. i.e., username/password should be same for all connections
Connections
Database Resident Connection Pool (DRCP)

• Enabling DRCP on the client side
  ---Specify (SERVER=POOLED) in short URL to enable DRCP

• Enabling DRCP on the server side
  – Enable with `dbms_connection_pool.start_pool`

• Use `ConnectionPoolDataSource` for other DataSources
  – Use `OracleConnectionPoolDataSource` as connection factory
  – Modify URL to add POOLED as above
  – Add Connection Class connection property
Connections
Multitenant Architecture

• Single instance for managing multiple databases – Multitenant
  – Container Database (CDB)
  – Pluggable Database (PDB) or Tenant

• Multitenant Container Database (CDB)
  – One set of background processes, one System global Area (SGA), many PDBs

• Manage Many as One
  – Easier resource management

• Simple Operations on PDBs as entities.
  – Unplug/plug, clone, create, drop
Connections
Multitenant Architecture

• **Problem to Solve**
  – Multiple database connections cause overhead

• **Solution**
  – Optimize connections by allowing sharing a pool across PDBs

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.
Connections
RAC Architecture

- Oracle RAC provides High Availability and Scalability for all application types
- RAC provides automatic work load management
  - High Availability Framework
  - Load Balancing Advisory
  - Services
  - Connection Load Balancing
Connections
RAC for scalability
alias =

(DESCRIPTION=(CONNECT_TIMEOUT=90)(TRANSPORT_CONNECT_TIMEOUT=10)(RETRY_COUNT=10)(RETRY_DELAY=3))

(ADDRESS_LIST=

(LOAD_BALANCE=on)

(ADDRESS=(PROTOCOL=TCP)(HOST=10.1.1.1)(PORT=10240))
(ADDRESS=(PROTOCOL=TCP)(HOST=10.1.1.2)(PORT=10240))
(ADDRESS=(PROTOCOL=TCP)(HOST=10.1.1.3)(PORT=10240))
(CONNECT_DATA=(SERVICE_NAME=sales)))
Program Agenda

Connections

2 Hard and Soft Parses

Result Set Cache

LOB Operations

Array Operations/Stored Procedures

Questions
Hard Pares

• **About Hard Parse**
  - Parse structure and Execution plan is created every time
  - Parse structure and Execution Plan is thrown out when the statement is closed and not reused

• **Problem to solve**
  - Causes library cache latch contention
  - Causes shared pool contention
  - Causes scalability issues

• **Solution**
  - Avoid Hard Parsing using **Prepared Statement & Bind Variables**
    - Reduces hard parses on the server
Hard Parses
An Example

• Instead of:
  String query = "SELECT EMPLOYEE_ID, LAST_NAME, SALARY FROM "+
  "EMPLOYEES WHERE EMPLOYEE_ID = " + generateNumber(MIN_EMPLOYEE_ID,
  MAX_EMPLOYEE_ID);
  prepStmt = connection.prepareStatement(query);
  resultSet = pstmt.executeQuery();

• Change to:
  String query = "SELECT EMPLOYEE_ID, LAST_NAME, SALARY FROM "+
  "EMPLOYEES WHERE EMPLOYEE_ID = ?";
  prepStmt = connection.prepareStatement(query);
  int n = generateNumber(MIN_EMPLOYEE_ID, MAX_EMPLOYEE_ID)
  prepStmt.setInt(1, n);
  resultSet = pstmt.executeQuery();
Soft Parses

• Parse structure and Execution Plan are saved and re-used.
• Session executes a statement that exists in shared pool
• Creates session specific cursor context
  – Executing a statement with the cursor context is cheaper
• Metadata processing is not required
  – Parse, Describe, Execute, Fetch → Four operations performed while executing any SQL
  – Soft Parsing eliminates Parse and Describe operations
Soft Parses

**Problems to solve**
- Though light weight, can take some time

**Solution**
- Enable Statement caching
  - `oracleDataSource.setImplicitCachingEnabled(true)`
- Choose the correct cache size to best utilize the memory
  - `connection.setStatementCacheSize(10);`
Program Agenda

Connections
Hard and Soft Parses

Result Set Cache
LOB Operations
Array Operations/Stored Procedures
Questions
Result Set Cache
Client Side Result Set Caching

- Cached data is shared across multiple connections from the same Data Source in the middle tier
- Faster access to frequently queried data
- Some of the benchmarks showed 3x improvement
- Change init.ora to enable Result Set Caching
  - CLIENT_RESULT_CACHE_SIZE – set to client result set cache size
  - CLIENT_RESULT_CACHE_LAG – Specifies client result cache lag in minutes
Program Agenda

Connections

Hard and Soft Pares

Result Set Cache

LOB Operations

Array Operations/Stored Procedures

Questions
Large Objects (LOBs)

• BASIC LOBs
  – LOBS are accessed via LOB locators

• LOBs Optimizations
  – Convert LOB whenever possible
    • LOB to LONG conversion using defineColumnType(index,type,size)
  – Tune Session Data Unit (SDU) for large LOBs
    • SDU can be specified in Connection URL
  – Use PreFetching for small LOBs
    • setLobPrefetchsize() – Set the prefetchsize to LOBsize for better performance
SecureFiles LOBs

- Optimize very large LOBs operations
- Large reads/writes
  - BASIC LOBs: internal buffer copy are expensive
- SECUREFILE LOBS: “Zero-copy IO” or “Vectored i/o mechanism”

```java
setupSecureFile()
Blob.getBytes()

Result Set

Fetch/Stream LOB data
directly (bypass internal
buffer)
Program Agenda

Connections

Hard and Soft Parses

Result Set Cache

LOB Operations

Array Operations/Stored Procedures

Questions
Array DML/Fetching/Prefetching

Best Practices

• Use array operations instead of single row operations
  – Single row DMLs/fetches incur excessive roundtrips

• Use Array pre-fetching

• Use Batch Update for Array DML operations
Array Fetching in Java

String query = "SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES "
+ " WHERE EMPLOYEE_ID > ? "
+ " ORDER BY EMPLOYEE_ID";

pstmt = connection.prepareStatement(query);
pstmt.setInt(1, generateNumber(MIN_EMPLOYEE_ID, MAX_EMPLOYEE_ID));
pstmt.setFetchSize(20);
rs = pstmt.executeQuery();
ResultSetMetaData rsmd = rs.getMetaData();
int columnCount = rsmd.getColumnCount();

while (rs.next()) {
    for (int i = 1; i <= columnCount; ++i)
        System.out.println(rsmd.getColumnName(i) + "[" + rsmd.getColumnTypeName(i) + "]": " + rs.getString(i));
}
Array DML in Java

String dml = "UPDATE EMPLOYEES SET SALARY = ?"
    + " WHERE EMPLOYEE_ID = ?";
pstmt = connection.prepareStatement(dml);
for(int i = 0; i < NUM_ROWS_TO_INSERT; ++i) {
    int empId = getEmployeeId(employeIdList);
    pstmt.setInt(1, getNewSalary(empId));
    pstmt.setInt(2, empId);
    pstmt.addBatch();
}
pstmt.executeBatch();
Stored Procedures
Best Practices

• Bundle multiple SQL statements in one call
  – Use stored procedures
  – Eliminates roundtrips to database
  – Eliminates moving data between database and client

• Can improve performance dramatically

• Monitor roundtrips and bytes transferred stats
  – High values may indicate optimization opportunities

• Oracle furnishes Java and PL/SQL Stored Procedures
Useful Information

• Real World Performance Playlist

http://www.youtube.com/playlist?list=PLKCK3OyNwIzvwEXdaubc6PQXwnQOAE9h2
Questions
Safe Harbor Statement

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.
Hardware and Software
Engineered to Work Together