
ORACLE®


GLOC
Great Lakes Oracle Conference
Best Practices for Scaling and Speeding Java Applications
MAY 16 & 17, 2018
CLEVELAND PUBLIC AUDITORIUM, CLEVELAND, OHIO
WWW.NEOOUG.ORG/GLOC

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.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Program Agenda

1. What's New?
2. Speeding Up Java applications
3. Scaling the Java workloads
4. Questions

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Program Agenda with Highlight

1. What's New?
2. Speeding Up Java applications
3. Scaling the Java workloads
4. Questions

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

What's New?

Java Cloud Connectivity

- Wallet Support in JDBC for DB Cloud services & Key Store Service (KSS)
  - Better support for SSO wallets (auto-login)
  - Auto-load and facilitates SSO wallets as trusted security provider.
  - Key Store Service for Web Logic Server and other Java containers.
- JDBC support for a new `ojdbc.properties` configuration file
- JDBC Connection URL enhancements
  - Support of MY_WALLET_LOCATION in the connect string
  - Setting TNS_ADMIN and the JDBC properties file in the connect string

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

What's New?

NEW IN 18c

Java Support for HA & App Continuity

- Continuous Availability - Support for Auto-AC and Auto-drain
- DRCP + AC support in driver and UCP
- Support for concrete classes with Application Continuity

Java Support for Sharding

- UCP Support for RAC Affinity using Sharding Infrastructure
- New APIs to expose sharding routing to mid-tier (WLS)
- Sharding performance optimizations

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

What's New?

Java Security

- JDBC support for HTTPS_PROXY and Websockets
- UCP Support for Secure Authentication

Performance Features

- Lightweight connection validation in driver and UCP
- UCP Performance optimizations

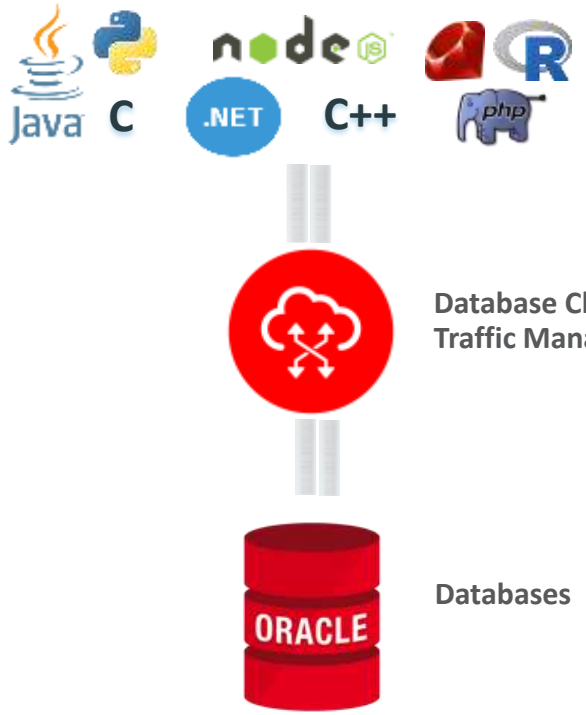
Datatypes

- JDBC support for RefCursor as In-bind for PL/SQL
- JDBC support for Object elements and the Index with index-by table
- `OracleResultSetMetaData.isColumnJSON(int index)` that returns true if the column is a JSON column

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Oracle Database Cloud Traffic Manager (DCTM)

NEW IN 18c



- Fully transparent to applications
- Clustered to avoid SPOF
- Supports zero application downtime
- Optimizes database session usage
- Routes database traffic
- Enhances database security

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Program Agenda with Highlight

1. What's New?
2. Speeding Up Java applications
3. Scaling Java applications
4. Questions

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Speeding Up Java Applications

- Use Connection URL with connection descriptors
- Use Universal Connection Pool (UCP)
- Tune excessive hard parsing
- Tune excessive soft parsing
- Enforce result set caching
- Use LOB Best Practices
- Use Array operations (Fetch, DML)
- Use Java in the Database
- Use Network compression over WAN
- Use Memory management best practices
- Lightweight Connection Validation

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Connection URL with Connection Descriptors

Oracle RAC with ADG

```
alias =(DESCRIPTION =
   (CONNECT_TIMEOUT=90)  (RETRY_COUNT=20)(RETRY_DELAY=3)
        (TRANSPORT_CONNECT_TIMEOUT=3)
    (ADDRESS_LIST =
          (LOAD_BALANCE=on)
          ( ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
     (ADDRESS_LIST =
          (LOAD_BALANCE=on)
          ( ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

Automatic Retries

No reliance on DNS

ALWAYS use a SERVICE that is NOT DB/PDB name

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Connection URL – Best Practices

- Always use the latest version of JDBC driver and UCP against current or earlier database version
- Do not mix up the versions of jars
  - Use **ojdbc8.jar, ucp.jar, and ons.jar** from 12.2.0.1 version
- Always return connections to the pool for stable connection usage, RLB, and planned draining
- Connection URL best practices
  - Use one DESCRIPTION and more can cause long delays
  - Set LOAD_BALANCE on per ADDRESS_LIST to balance SCANs
  - DO NOT USE RETRY_COUNT without RETRY_DELAY
  - DO not use Easy Connect URL – it has no HA capabilities

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Universal Connection Pool (UCP)

The best configured connection pools result in higher throughput

UCP Properties	Description
setInitialPoolSize(), setMinPoolSize(), setMaxPoolSize()	Set pool size based on database server resources (number of cores) Eg: MaxPoolSize = *(rdbms-cores * n) / sum (pools-per-mid-tier)*
setTimoutCheckInterval(int) **setTimeToLiveConnectionTimeout() ,** **setAbandonConnectionTimeout(),** **setConnectionWaitTimeout(int)** **setInactiveConnectionTimeout(int)**	Set the Connection Timeout sufficiently high enough to suite the application profile.
setMaxStatements()	Enable Statement Caching – By default it is OFF
setFastConnectionFailoverEnabled(), setONSConfiguration()	For High Availability Features. Enabled by default in 12.2 UCP

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Excessive Hard Statement Parsing?

Hard parse: all steps involved in creating a SQL statement parse tree
Monitor hard parses with AWR/ADDM

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	10.4	43.1	0.00	0.00
DB CPU(s):	1.7	6.9	0.00	0.00
Redo size:	11,793.8	49,001.1		
Logical reads:	6,588.8	27,375.1		
Block changes:	30.7	127.4		
Physical reads:	444.9	1,848.6		
Physical writes:	28.6	118.9		
User calls:	11,032.4	45,837.5		
Parses:	5,988.3	24,880.3		
Hard parses:	920.2	3,823.4		
W/A MB processed:	279,318.6	1,160,517.8		
Logons:	0.4	1.6		
Executes:	6,003.7	24,944.3		
Rollbacks:	0.0	0.0		
Transactions:	0.2			

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Tune excessive hard parsing

- Avoid Hard Parsing using **Prepared Statement & Bind Variables**

Instead of:

```
String query = "SELECT EMPLOYEE_ID, LAST_NAME, SALARY FROM EMPLOYEES
WHERE EMPLOYEE_ID = " +  generateNumber();
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);
prepStmt.setInt(1, generateNumber());
resultSet = pstmt.executeQuery();
```

Fallback if application cannot be changed to use binds

init.ora parameter: **CURSOR_SHARING={FORCE|SIMILAR|EXACT}**

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Tune excessive soft parsing

- Enable Statement caching
  - `oracleDataSource.setImplicitCachingEnabled(true)`
- Choose the right cache size to best utilize the memory
  - `connection.setStatementCacheSize(10)`
  - Try to be closer to the number of most used statements
  - Default statement cache size is 10
- Fallback if you cannot change the application to use statement caching
  - **session_cached_cursors = 50**

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Enforce result set caching

Server side result set caching

- Query not re-executed (until cache invalidated)

Client side result-set caching

- Shared across multiple connections from the same Data Source in the middle tier
- Faster access to frequently queried data. Enable with a SQL hint
  - String query = "select /** result_cache */ first_name, last-name from employees where employee_id < : 1”;
- Transparent invalidation through Query Change Notification
- Server side configuration (**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

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

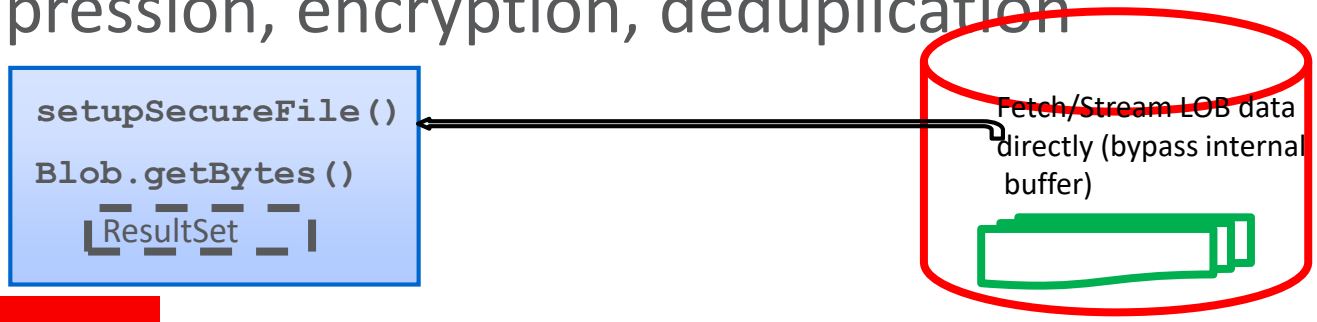
Use LOB best practices

- Convert LOB to LONG using **`defineColumnType(index,type,size)`**
- Tune Session Data Unit (SDU) for large LOBs, XML, large Result sets
  - Max: **2MB (12c)**, 64K (11.2), 32K (pre-11.2)
  - Set on both server and client side (SQLNet.ORA, TNSNAMES.ORA or URL)
  - jdbc:oracle:thin:@(DESCRIPTION=**(SDU=11280)** (ADDRESS=(PROTOCOL=tcp)(HOST=myhost-vip)(PORT=1521)) (CONNECT_DATA=(SERVICE_NAME=myorcldbservicename)))
  - Improves performance, network utilization, and memory consumption
- Use PreFetching for small LOBs **`setLobPrefetchsize()`**
- Data Interface
  - Streamlined mechanism for writing and reading the entire LOB contents using the standard JDBC methods **`getString()`** and **`setBytes()`**

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Use SecureFiles LOBs

- Large reads/writes
  - BASIC LOBs: internal buffer copy are expensive
  - SECUREFILE LOBS: ”oracle.net.useZeroCopyIO” or “Vectored i/o mechanism”
- Transparent to existing APIs
  - Use **`public boolean isSecureFile() throws SQLException`** to check whether or not your BLOB or CLOB data uses Oracle SecureFile storage
- Compression, encryption, deduplication



Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Array DML/Fetching/Prefetching

Best Practices

- Use array operations instead of single row operations
  - Single row DMLs/fetches incur excessive roundtrips
  - Default fetch size is 10
- Check if array size is large enough
- Some drivers support prefetching instead of array fetching

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

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());
 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));

 }
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Array DML in Java

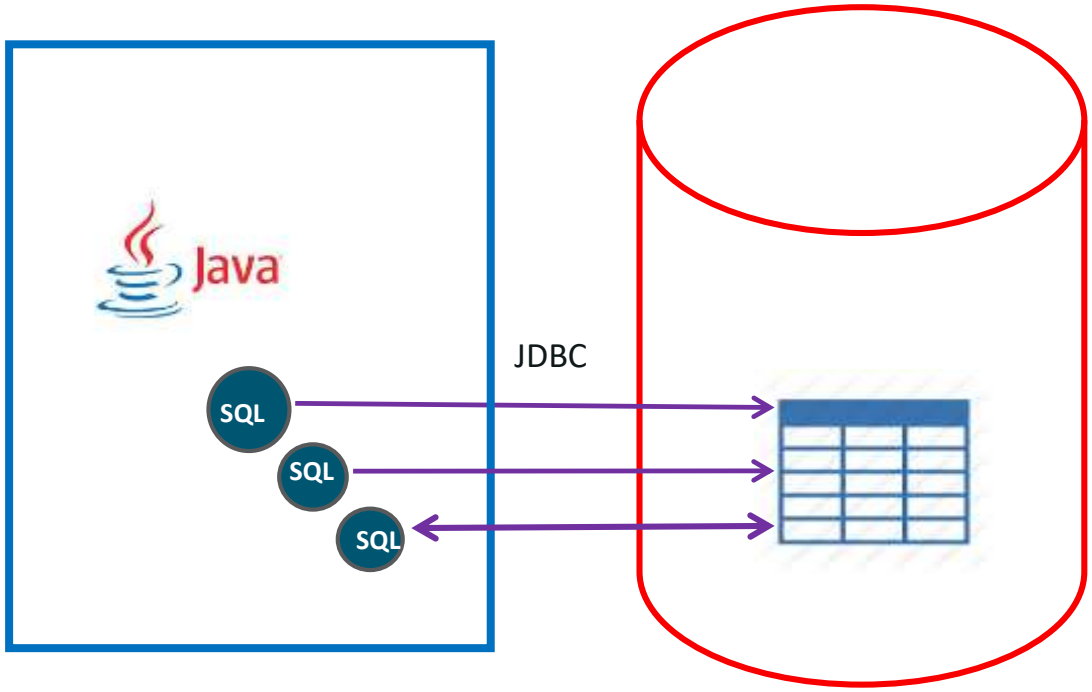
```
String dml = "UPDATE EMPLOYEES SET SALARY = ?"
             +" WHERE EMPLOYEE_ID = ?";
pstmt = connection.prepareStatement(dml);
for(int i = 0; i < NUM_ROWS_TO_INSERT; ++i) {
  int empId = getEmployeeId(employeeIdList);
  pstmt.setInt(1, getNewSalary(empId));
  pstmt.setInt(2, empId);
  pstmt.addBatch();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

pstmt.sendBatch();
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Client-side Java Database Access

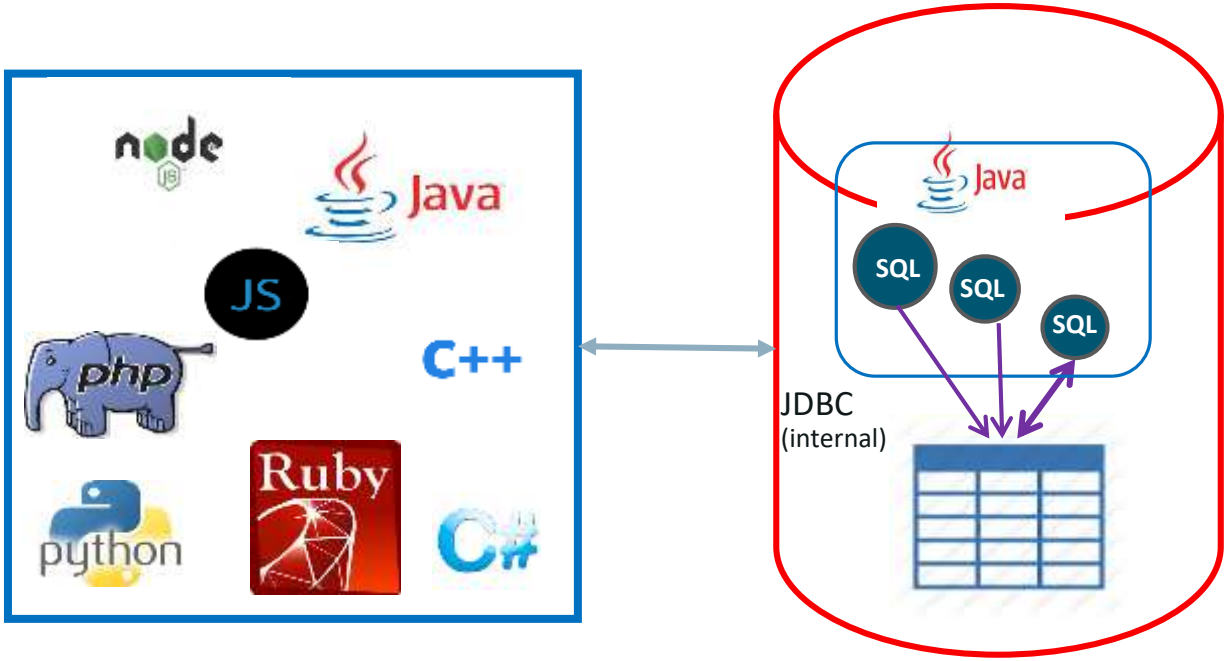
- Multiple Network exchange for processing each SQL statements
- Could lead to high latency in WAN or Cloud environments



Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Java in the Database

- One call for the entire procedure
- Low network latency in WAN or Cloud environments
- Can be consumed by SQL, PL/SQL or any client

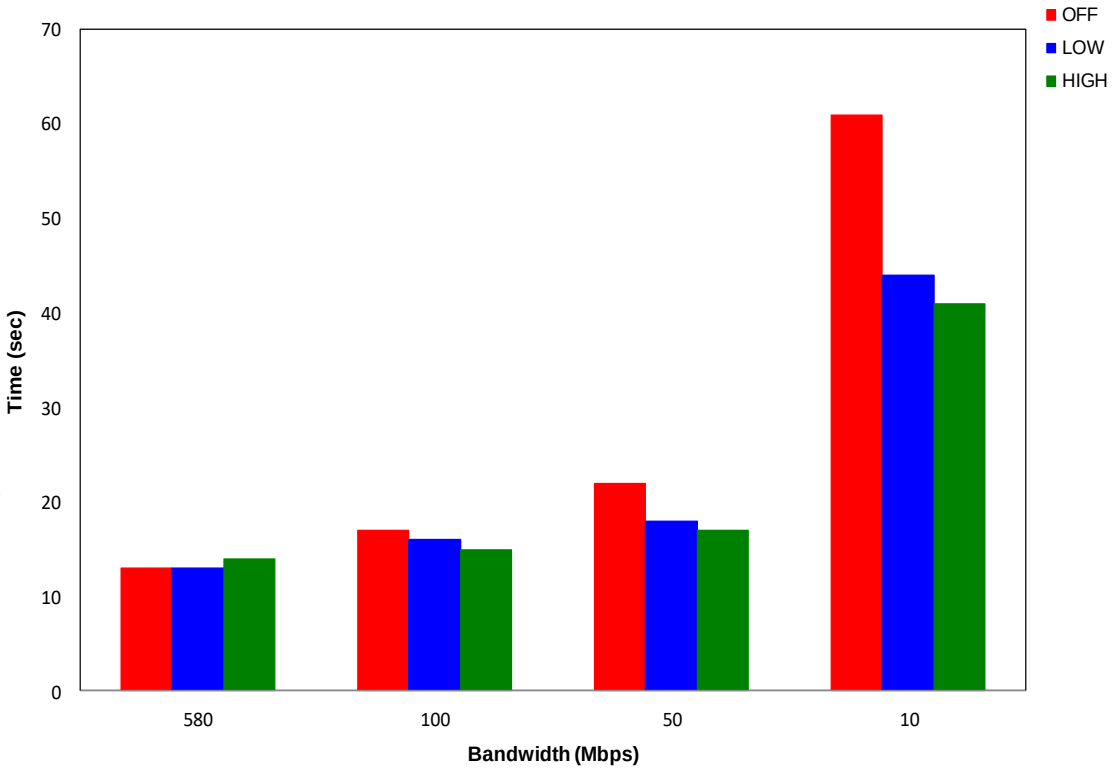


Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Network Compression Over WAN

Mixed Data

- **Query:**
  Select * from Table;
- **Data:**
  Five columns
  50 bytes per row
  1 million rows
- **Array Size:**
  5000
- **Connection Property:**
  oracle.net.networkCompression=”on”
- Reduces the size of SDU transmitted across the network



Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Memory Management Tips

- Memory issue is caused due to the buffers used to store query results
  - **More dynamic buffer mechanism** and return buffers to the cache when the ResultSet is closed
  - CREATE TABLE TAB (ID NUMBER(10), NAME VARCHAR2(40), DOB DATE)
    ResultSet r = stmt.executeQuery("SELECT ID, NAME, DOB FROM TAB");
- Manage the memory used by buffers
  - Form Queries carefully (Retrieve columns explicitly and avoid select * from employees)
  - Set the fetchsize carefully (avoid setting higher setFectchSize(10000))
- Use PreparedStatements and enable Implicit Statement Caches
- Control the memory heap of JVM using –Xmx or –Xms

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Lightweight Connection Validation

- Checks only the underlying socket health
- Performs a faster validation by sending an empty data packet to DB and doesn't wait to receive it back. Successful send indicates that socket is alive.
- Two ways to enable lightweight connection validation
  - Enable this with a system property `oracle.jdbc.defaultConnectionValidation="SOCKET"`
  - Use a new overloaded method `OracleConnection.isValid(ConnectionValidation.SOCKET, timeout)`

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Program Agenda with Highlight

1. What's New?
2. Speeding Up Java Applications
3. Scaling Java Applications
4. Questions

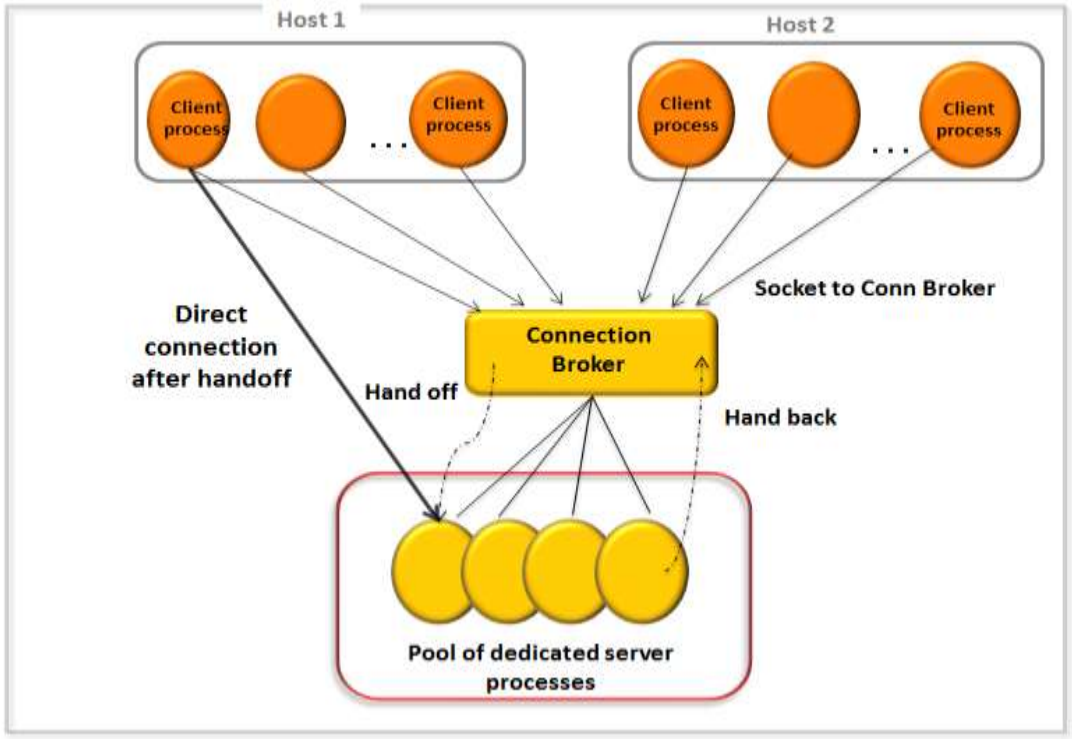
Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Scaling Java Applications

- Using Server Side Connection Pool (DRCP)
- Horizontally Scaling Java Workload
- Using Runtime Load Balancing (RLB)

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Database Resident Connection Pool (DRCP)



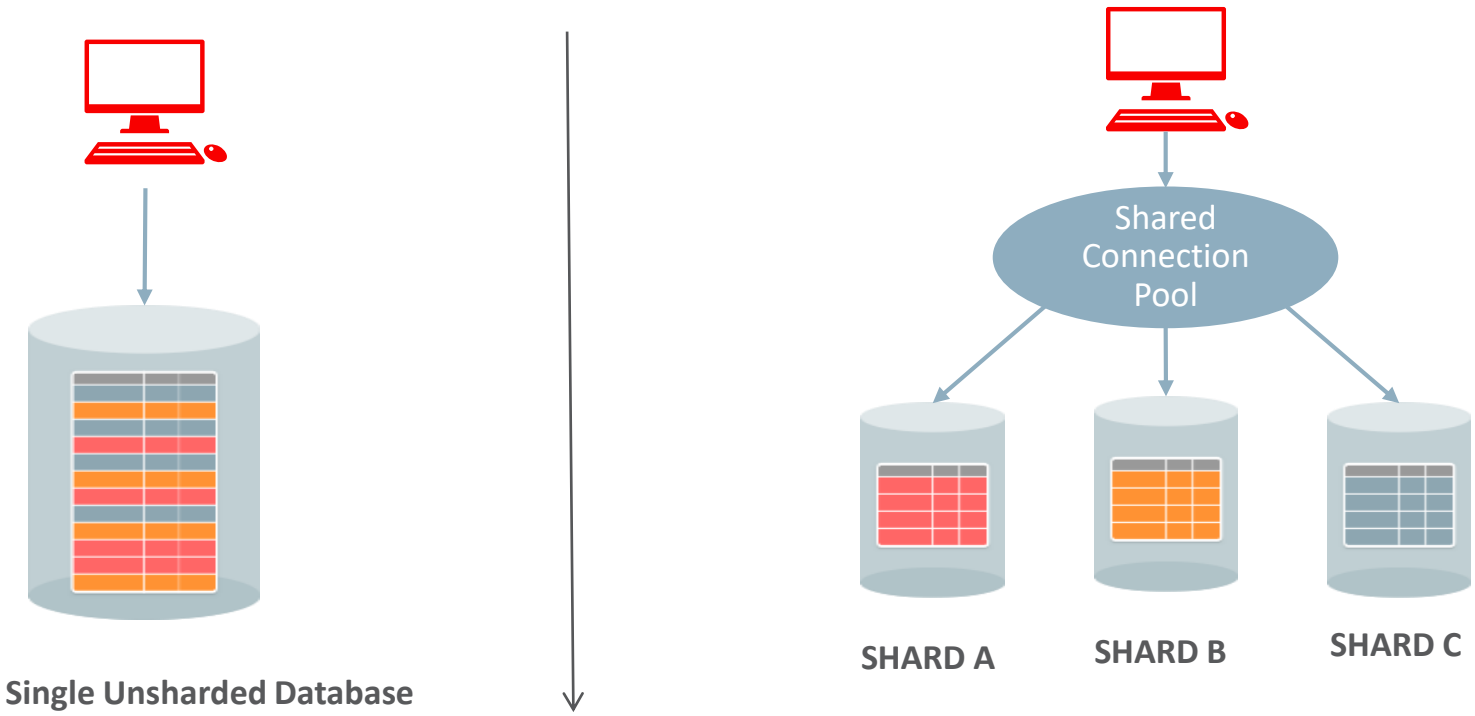
Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Using Server side connection Pool (DRCP)

- DRCP is the server side connection pool shared across many mid-tiers
- Enabling DRCP on the server side
  - dbms_connection_pool.start_pool();
- Change the connection URL to enable DRCP
  - Eg:,jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(HOST=myhost)(PORT=1521)(PROTOCOL=tcp))(CONNECT_DATA=(SERVICE_NAME=myorcldbservicename))(SERVER=POOLED)))";
- DRCP Tagging – Associate a connection to a mid-tier with a particular tag name to retrieve a specific session easily.
  - Enable using the property `oracle.jdbc.UseDRCPMultipletag = true`

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Horizontal Scaling Java Workloads



Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Scaling Java applications

- Session based Sharding via Sharding Key
- Sharding APIs – JDBC/UCP
- UCP as a Shard Director

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

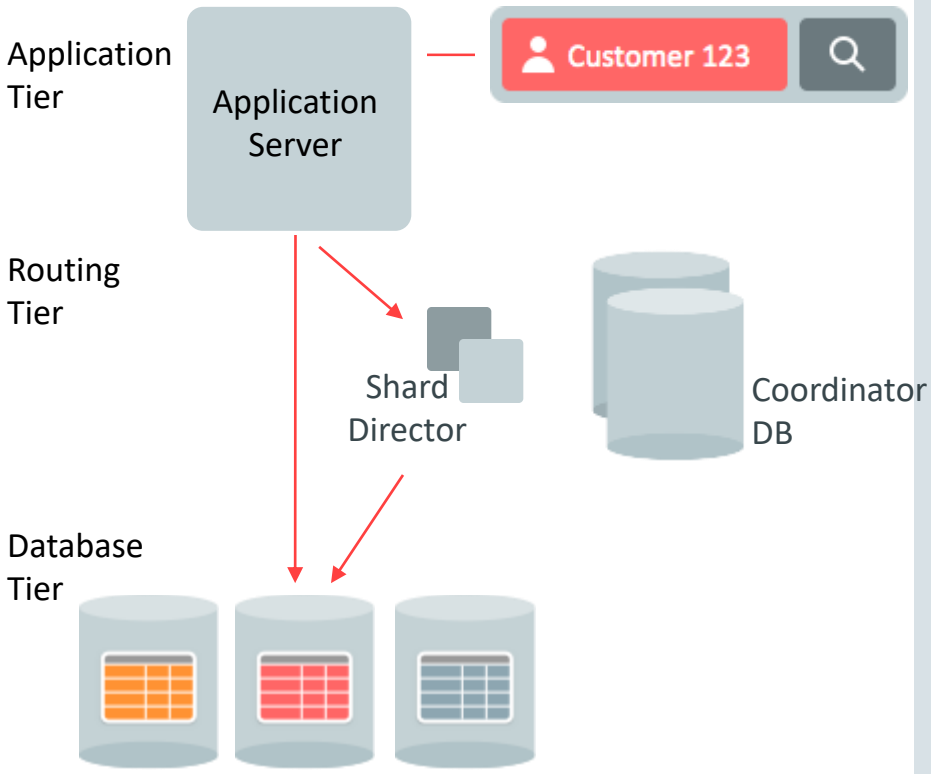
Session-Based Routing via Sharding Key

Sharding Key: A partitioning key for a sharded table.

- E.g: custid- shard key used for partitioning the table 'customers'.

Super Sharding Key: Required in case of composite sharding.

Shard Director: Looks up sharding key and redirects to right shard containing the data.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Sharding APIs : JDBC/UCP

Building Sharding Key and Super Sharding Key

```
// Sharding Key is a compound key of 2 sub-keys
 OracleShardingKey shardingKey =
           dataSource.createShardingKeyBuilder()
                .subkey("Customer_EMAIL", oracle.jdbc.OracleType.VARCHAR2)
                     .subkey("1234",
oracle.jdbc.OracleTypes.NUMBER)
                .build();

 // Super Sharding Key with only one sub-key
 OracleShardingKey superShardingKey =
          dataSource.createShardingKeyBuilder()
                .subkey("Customer_Location_US",oracle.jdbc.OracleType.VARCHAR2
)
                .build();
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

JDBC/UCP connection with a Sharding Key

Oracle JDBC Driver

```
OracleDataSource ods =
    new OracleDataSource();

// Set connection properties
ods.setURL(DB_URL);
ods.setUser("hr");
ods.setPassword("****");
// Get an Oracle JDBC connection for a shard
Connection conn =
ods.createConnectionBuilder()
      .shardingKey(shardingKey)
      .superShardingKey(superShardingKey)
      .build();
```

Oracle Universal Connection Pool (UCP)

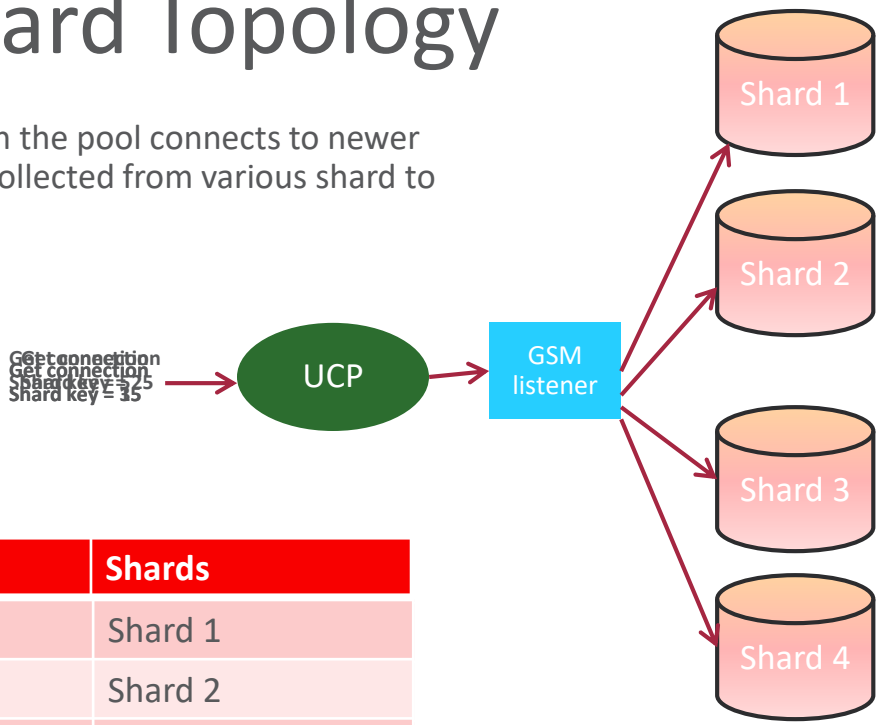
```
PoolDataSource pds =
 PoolDataSourceFactory.getPoolDataSource();

// Set Connection Pool properties
 pds.setURL(DB_URL);
 pds.setUser("hr");
 pds.setPassword("****");
 pds.setInitialPoolSize(10);
 pds.setMinPoolSize(20);
pds.setMaxPoolSize(30);
// Get an UCP connection for a shard
Connection conn =
 pds.createConnectionBuilder()
    .shardingKey(shardingKey)
    .superShardingKey (superShardingKey)
    .build();
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

UCP Learning Shard Topology

- During pool initialization (or when the pool connects to newer instances) the shard topology is collected from various shard to create a Shard Routing Table

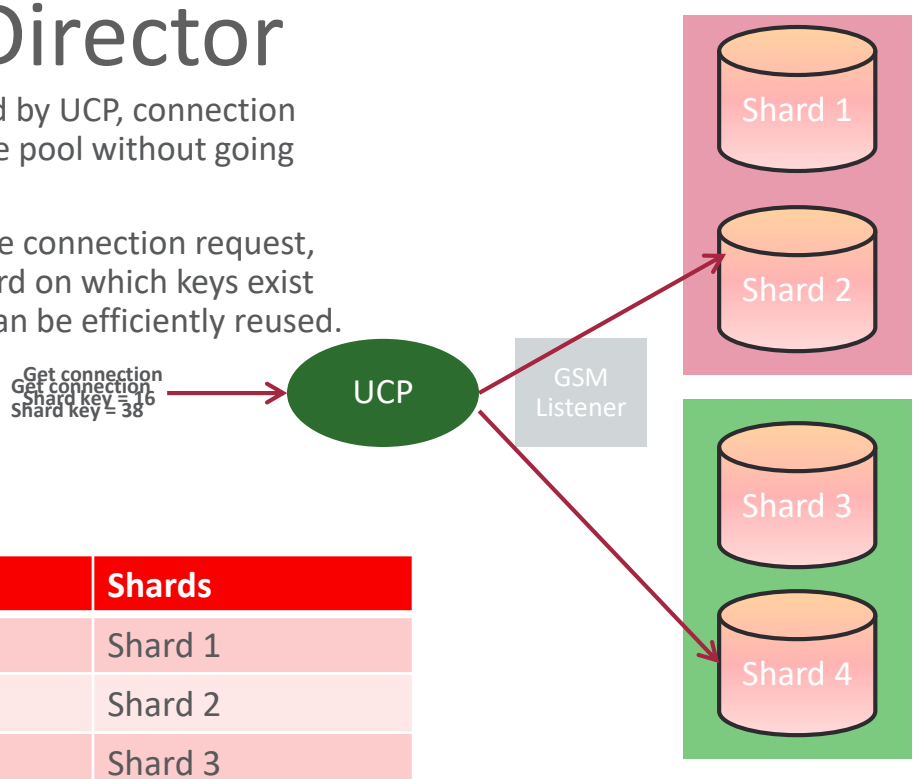


Shard Keys Range	Chunk Name	Shards
1 -- 10	Chunk 1	Shard 1
10 -- 20	Chunk 2	Shard 2
20 -- 30	Chunk 3	Shard 3
30 -- 40	Chunk 4	Shard 4

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

UCP as a Shard Director

- Once the required topology is collected by UCP, connection requests can be directly serviced by the pool without going through the shard director.
- Based on the shard keys provided in the connection request, UCP can lookup the corresponding shard on which keys exist and pooled connections to the shard can be efficiently reused.



Shard Keys Range	Chunk Name	Shards
1 -- 10	Chunk 1	Shard 1
10 -- 20	Chunk 2	Shard 2
20 -- 30	Chunk 3	Shard 3
30 – 40	Chunk 4	Shard 4

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Run time Load Balancing

- Leverage Load Balancing Advisory
  - Used to balance the work across the RAC instances.
  - Determine the instance that offers the best performance
  - Establish connections based on the advisory
- UCP uses RAC Load Balancing Advisory and distributes the load equally among the instances.
- Service goals must be set to SERVICE_TIME or THROUGHPUT
  - srvctl modify service -db <db_name> -service <service_name> **-rlbgoal SERVICE_TIME -clbgoal SHORT**
  - srvctl modify service -db <db_name> -service <service_name> **-rlbgoal THROUGHPUT -clbgoal SHORT**

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Program Agenda with Highlight

1. What's New?
2. Speeding Up Java Applications
3. Scaling Java Applications
4. Questions

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Questions?

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. | Confidential – Oracle Internal

Integrated Cloud

Applications & Platform Services

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |


ORACLE®