Sustaining Planned/Unplanned Database Outages: Best Practices for DBAs & Developers

Nirmala Sundarappa
Principal Product Manager,
Oracle JDBC & UCP
&
Kuassi Mensah
Director, Product Management
Oracle Database Development

@kmensah – db360.blogspot.com

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.
Program Agenda

1. Maximum Availability planned/unplanned downtimes
2. Building Blocks which Solved the Issues
3. DBAs & Developers Steps for Planned Maintenance
4. DBAs & Developers Steps for Unplanned Downtime
5. Questions
Achieving Maximum Application Uptime

Avoid this
Objectives for planned/unplanned downtimes

Planned Maintenance

– Detect “DOWN” event triggered by DBA on instances targeted for maintenance
– Drain sessions away from that (those) instance(s)
– Perform transparent maintenance
– Resume operations on upgraded instance(s)

Unplanned Downtime

– Detect “DOWN” event
– Reliable determination of the outcome of In-flight work
– Continuation of In-flight work
– Recovery with response time targets
Program Agenda

1. Maximum Availability with planned/unplanned downtimes
2. Building Blocks which Solved the Issues
3. DBAs & Developers Steps for Planned Maintenance
4. DBAs & Developers Steps for Unplanned Downtime
5. Questions
Building Blocks of High Availability (HA): Planned & Unplanned

- Universal Connection Pool (UCP)
- Fast Application Notification (FAN)
- Oracle Notification System (ONS)
- Fast Connection Failover (FCF)
- Transaction Guard (TG)
- Application Continuity (AC)

- Database request (unit of work)
- Logical Transaction ID (LTXID)
- Recoverable Errors
- Mutable Functions
Universal Connection Pool (UCP)

- Standalone jar (ucp.jar) and simple to configure in any application servers
- Designed to work with RAC, ADG, and GDS
- An Example

```java
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin:@//localhost:1521/orcl");
pds.setUser("<user>"); pds.setPassword("<password>");
Connection conn = pds.getConnection();
```
UCP Configuration in Tomcat
Context.xml

```xml
<Context docBase="UCPTomcat" path="/UCPTomcat"
   reloadable="true" source="org.eclipse.jst.jee.server:UCPTomcat">

<Resource name="tomcat/UCPPool" auth="Container"
   factory="oracle.ucp.jdbc.PoolDataSourceImpl"
   type="oracle.ucp.jdbc.PoolDataSource"
   description="UCP Pool in Tomcat"
   connectionFactoryClassName="oracle.jdbc.pool.OracleDataSource"
   minPoolSize="40" maxPoolSize="60" initialPoolSize="50" autoCommit="false"
   user="<user>" password="<password>
   url="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=racscan)(PORT=1521)(SERVICE_NAME=orcl)))" />

</Context>
```
UCP Configuration in WebSphere

WAS Admin Console

Java Database Connectivity (JDBC) providers in the WebSphere Admin Console: The JDBC provider object encapsulates the specific JDBC driver implementation class for access to the specific vendor database of your environment.

Configuration

General Properties

- Scope
  - cell:cl03/nodem01/CElf
- Name
  - Oracle JDBC Driver UCP
- Description
  - Oracle JDBC Driver UCP
- Class path
  - /$ORACLE_JDBC_DRIVER_PATH/jdbc-1.0.jar
  - /$ORACLE_JDBC_DRIVER_PATH/oracle-jdbc-x.x.jar
  - /$ORACLE_JDBC_DRIVER_PATH/oracle-x.x.jar

Native library path

- Isolate this resource provider
- Implementation class name
  - oracle.jdbc.pool.PoolDataSourceImpl
Fast Application Notification (FAN)

• Oracle FAN APIs provide event notification for developing more responsive applications to take advantage of Oracle Database HA features.

• Supported FAN events are mentioned below
  – **Up** – Relocates work when services resume
  – **Service Down** – Re-allocates sessions when services resume
  – **Node Down** – Re-allocates sessions when services resume
  – **RLB %** - Hint to balance sessions locally/globally
  – **Affinity** - Hint to maintain conversation locality
Oracle Notification Service (ONS)

- Simple publish/subscribe method to produce and deliver event messages for both local and remote consumption.
- ons.jar: Needs to be included in the classpath
- Database Side Configuration
  
  ```
  $ORACLE_HOME/opmn/conf/ons.config or $ORACLE_CONFIG_HOME/opmn/conf/ons.config.<hostid>
  
  # First three values are required
  localport=4100
  remoteport=4200
  nodes=racnode1:4200, racnode2:4200, racnode3:4200
  
  $srvctl status nodeapps |grep ONS
  ```

- Application Side Configuration
  
  ```
  <!-- Same string as ons.config; not needed in 12c Auto-ONS-- >
  onsConfiguration="nodes=racnode1:4200,racnode2:4200, racnode3:4200"
  ```
Fast Connection Failover (FCF)

- FCF is a feature implemented using FAN APIs in the connection pool
- FCF is enabled in the application as follows.

```java
import PoolDataSourceFactory;

PoolDataSource pds = new PoolDataSourceFactory.getPoolDataSource();

// Not required with auto-ONS in 12c
pds.setONSConfiguration(ONS_CONFIG);

pds.setFastConnectionFailoverEnabled(true);
```
Enabling FCF in Tomcat

Context.xml

```xml
<Context docBase="UCPTomcat" path="/UCPTomcat"
        reloadable="true" source="org.eclipse.jst.jee.server:UCPTomcat">
    <Resource name="tomcat/UCPPool" auth="Container"
              factory="oracle.ucp.jdbc.PoolDataSourceImpl"
              type="oracle.ucp.jdbc.PoolDataSource"
              description="UCP Pool in Tomcat"
              connectionFactoryClassName="oracle.jdbc.pool.OracleDataSource"
              minPoolSize="40" maxPoolSize="60" initialPoolSize="50" autoCommit="false"
              user="<user>" password="<password>" fastConnectionFailoverEnabled="true"
              url="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=racscan)(PORT=1521)) (CONNECT_DATA=(SERVICE_NAME=orcl)))" />
</Context>
```
Recoverable Errors

• Oracle DB 12c error messages have a new `OracleException.IsRecoverable` property
• JDBC throws a `SQLRecoverableException`; your application checks connection validity using JDBC 4.0 `isValid()`

```java
try {
    conn = pds.getConnection(); ...
} catch (SQLRecoverableException rea) {
    if (!((ValidConnection)conn).isValid()) {
        ea.printStackTrace();
        conn.close();
        System.out.println("Retry to get a new connection" + ea.getMessage());
    }
}
```
Database Request & Logical Transaction ID (LTXID)

• **Database Request (Unit of Work)**
  – Demarcates a replay-able unit of work

    ```java
    Connection conn = pooldatasource.getConnection();
    // 3rd party conn pool must use beginRequest()
    PreparedStatement pstmt = ...
    ...
    SQL, PL/SQL, local calls, RPC
    ...
    conn.commit();
    // 3rd conn pool must use endRequest()
    conn.close();
    ...
    ```

• **Logical Transaction ID (LTXID)**
  – Each transaction is associated with a LTXID
    issued at the beginning of the Tx
  – LTXIDs are changed when a transaction is committed or rolled back
Transaction Guard
At most Once COMMIT Execution

1. Application starts a TX; RDBMS associates LTXID (Logical Transaction ID)
2. Application performs a transaction (DML) & issues a COMMIT
3. COMMIT fails due to unplanned outage
   The RDBMS aborts the session and associated connection dies
4. UCP receives a FAN DOWN event and performs the following
   • receives a recoverable SQL Exception
   • gets the LTXID from the dead connection
   • gets a new connection and checks the outcome of the COMMIT using the LTXID
Application Continuity (AC)

1- Normal Operation
• Marks database requests
• Builds proxy objects
• Decides what can & cannot be replayed
• Holds original calls with binds and validation

2 - Reconnect
• Checks Replay is enabled
• Creates a new session
• Validates target database
• Uses Transaction Guard to check outcome of in-flight COMMIT

3 - Replay
• Replays held calls
• If user visible results match, based on validations then application proceeds as if nothing happened
AC - Mutable Functions

• **SYSDATE, SYSTIMESTAMP, SEQUENCES and SYS_GUID**, and so on which results change on each invocation
  
  – Replaying mutable functions will not give the same result as the initial call.
  
  – (Web) Applications may or may not be sensitive to mutable functions during the resubmission of the same unit of work
  
  – To keep the same values, the user schema must be granted the right to do so, by DBA
Program Agenda

- Maximum Availability with planned/unplanned downtimes
- Building Blocks which solved the Issues
- **3** DBAs & Developers Steps for Planned Maintenance
- DBAs & Developers Steps for Unplanned Downtime
- Questions
DBAs & Developers Steps for Planned Maintenance

“UCP at work”
Planned Maintenance
Development Steps

• Two Simple steps as shown below.

  (1) Enable Fast Connection Failover (FCF) as shown below.

  ```java
  PoolDataSource pds = new PoolDataSourceFactory.getPoolDataSource();
pds.setONSConfiguration(ONS_CONFIG); // not required with auto-ONS in 12c
pds.setFastConnectionFailoverEnabled(true);
  ```

  Make sure to have ons.jar file in the classpath
  ‘ONSConfiguration’ is required only in pre 12c database version

  (2) UCP introduces a new system property to gracefully drain the sessions

  ```
  -Doracle.ucp.PlannedDrainingPeriod=30
  ```

  Smoothens the session draining
  Avoids logon storms on the active instance
## Planned Maintenance
### DBA Steps

<table>
<thead>
<tr>
<th>Steps</th>
<th>Additional Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stop the service without ‘-force’ or Relocate the service</strong></td>
<td><code>$srvctl stop service -db &lt;db_name&gt; -service &lt;service_name&gt; -instance &lt;instance_name&gt;</code>&lt;br&gt;or&lt;br&gt;<code>$srvctl relocate service -db &lt;db_name&gt; -service &lt;service_name&gt; -oldinst &lt;oldins&gt; -newinst &lt;newinst&gt;</code></td>
</tr>
<tr>
<td><strong>Disable the service &amp; Allow sessions to drain Eg.2-30mins</strong></td>
<td><code>$srvctl disable service -db &lt;db_name&gt; -service &lt;service_name&gt; -instance &lt;instance_name&gt;</code></td>
</tr>
<tr>
<td><strong>Wait to allow sessions to drain</strong></td>
<td>Wait for 10-30 minutes</td>
</tr>
<tr>
<td><strong>Check for long-running sessions.</strong></td>
<td>SQL&gt; select count(*) from ( select 1 from v$session where service_name in upper('&lt;service_name&gt;') union all select 1 from v$transaction where status = 'ACTIVE' )&lt;br&gt;SQL&gt; exec dbms_service.disconnect_session ('&lt;service_name&gt;', DBMS_SERVICE.POST_TRANSACTION);</td>
</tr>
</tbody>
</table>
Planned Maintenance  
**DBA Steps-Contd...**

<table>
<thead>
<tr>
<th>Steps</th>
<th>Additional Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeat the steps above</td>
<td>Repeat for all services targeted for planned maintenance</td>
</tr>
<tr>
<td>Stop the database <strong>instance</strong></td>
<td><code>$srvctl stop instance -db &lt;db_name&gt; -instance &lt;instance_name&gt; -stopoption immediate</code></td>
</tr>
<tr>
<td>Disable the <strong>instance</strong></td>
<td><code>$srvctl disable instance -db &lt;db_name&gt; -instance &lt;instance_name&gt;</code></td>
</tr>
<tr>
<td>Maintenance</td>
<td>Apply patch or carry out the scheduled maintenance work</td>
</tr>
<tr>
<td>Enable &amp; Start  the <strong>instance</strong></td>
<td><code>$srvctl enable instance -db &lt;db_name&gt; -instance &lt;instance_name&gt;</code></td>
</tr>
<tr>
<td>Enable &amp; start the <strong>service</strong> back</td>
<td><code>$srvctl start instance -db &lt;db_name&gt; -instance &lt;instance_name&gt;</code></td>
</tr>
<tr>
<td>Check if the <strong>service</strong> is up and running</td>
<td><code>$srvctl enable service -db &lt;db_name&gt; -service &lt;service_name&gt; -instance &lt;instance_name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>$srvctl start service -db &lt;db_name&gt; -service &lt;service_name&gt; -instance &lt;instance_name&gt;</code></td>
</tr>
</tbody>
</table>
Planned Maintenance
Graphical representation

Fig 1: Planned Downtime

Time (Mins)

Number of Connections

RAC.Inst.1
RAC.Inst.2
Program Agenda

Maximum Availability with planned/unplanned downtimes

Building Blocks which solved the Issues

DBAs & Developers Steps for Planned Maintenance

4 DBAs & Developers Steps for Unplanned Downtime

Questions
DBAs & Developers Steps for Unplanned Downtime

Application Continuity (AC) can fix this
Unplanned Downtime

Application changes

• Few Simple steps as shown below; JDBC & UCP take care of the rest

<table>
<thead>
<tr>
<th>Building Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Fast Connection Failover (FCF)</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Design for Application Continuity (AC)</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Design for Transaction Guard (TG)</td>
<td>Optional, can be used independently</td>
</tr>
</tbody>
</table>
Unplanned Downtime
Application Continuity – DBA Steps

• Specify the Replay Data Source
  – connectionFactoryClassName="oracle.jdbc.replay.OracleDataSourceImpl"

• Some calls can be excluded from replay by disabling them with disableReplay() method of ReplayableConnection interface

• DBA must configure a service with the following properties. Default database service is not recommended
  – FAILOVER_TYPE='TRANSACTION' -- For Application Continuity
  – REPLAY_INITIATION_TIMEOUT=1800 -- After which replay is cancelled
  – FAILOVER_DELAY=10 -- Delay in seconds between connection retries
  – FAILOVER_RETRIES=30 -- Number of connection retries per replay
Unplanned Downtime
Enabling Transaction Guard – DBA Steps

• Configure the service for TG
  – $srvctl modify service –db <db_name> –service tgservice –commit_outcome true
  – $GDSCTL modify service –db <db_name> -service tgservice –commit_outcome true

• Grant execute privilege to the user schema (e.g., SCOTT)
  – SQL> GRANT EXECUTE ON DBMS_APP_CONT TO SCOTT;
White Papers

“Java Programming with Oracle Database 12c RAC and Active DataGuard”
http://www.oracle.com/technetwork/database/application-development/12c-ha-concepts-2408080.pdf

“Planned/Unplanned downtime & Runtime Load Balancing with UCP in WebSphere”

“Planned/Unplanned downtime & Runtime Load balancing with UCP in Tomcat”
Program Agenda

- Maximum Availability with planned/unplanned downtimes
- Building Blocks which solved the Issues
- DBAs & Developers Steps for Planned Maintenance
- DBAs & Developers Steps for Unplanned Downtime

Questions
Hardware and Software
Engineered to Work Together