When Your Database becomes 100% Available

Pieter Van Puymbroeck - Laurent Léturgez
We will do this presentation in English to avoid this ....

https://www.youtube.com/watch?v=Kh1Y4hHCszc
About Pieter

- Sailor
- Musician —> Earplugs provided ;-)
- Engineered systems Oracle DBA
- dad

working for Exitas Oracle Platinum Partner in BELGIUM

And ....

http://vanpupi.stepi.net
likes to be in control
Dreams at night ;}
About Laurent

• Oracle Consultant since 2001
• Former developer (C, Java, perl, PL/SQL)
• Hadoop aficionado
• Owner@Premiseo: Data Management on Premises and in the Cloud
• Blogger since 2004: http://laurent-leturgez.com
• Dad of 2, guitar player
• Twitter: @lleturgez
About Laurent & Pieter

• Neighbours
• Passionate by
  • Beers
  • Flemish food
  • ... and Oracle
3 Membership Tiers
- Oracle ACE Director
- Oracle ACE
- Oracle ACE Associate

Nominate yourself or someone you know: acenomination.oracle.com
Disclaimer

This presentation covers the day to day common failures and does not cover planned interventions.
Agenda

• What is Application Continuity
• Data source types usable with AC
• How does it work? ... Nothing is worth a demo :)
• What about performances?
Part 1: What is Application Continuity
What is Application Continuity

• We all know Oracle RAC

• RAC main features are:
  • Scalability
  • High availability
  • Oh, I mean ... database High Availability (and related resources: network, storage, network services etc.)

• But, what about application availability when resources are relocated?
What is Application Continuity

• Existing tools to improve applications’ availability
  • Transparent Application Failover
  • Fast Application Notification
  • Fast Connection Failover
• Transparent application failover (TAF)
  • Feature of the OCI driver (And JDBC thick driver)
  • Provides
    • Automatic re-connection
    • Session migration
    • Resumable queries (SELECT statements)
What is Application Continuity

• Fast Application Notification (FAN)
  • Up/down events sent to clients through Oracle Notification Service (ONS)
  • ONS subscription
  • Applications have to process each notification and react as required (reconnection, transaction rollback and replay etc.)
  • Works with JDBC thin, OCI, ODP.Net
  • Fully integrated with UCP pools
What is Application Continuity

• Fast Connection Failover (FCF)
  • FCF is a FAN Client integrated in the JDBC connection pool
  • Feature not enabled by default
    • setONSConfiguration to setup ONS config for the pooled datasource
    • setFastConnectionFailoverEnable(true) to enable FCF

• Main limitations of TAF, FAN and FCF
  • Don’t work for transactions (DML). A simple rollback is done
  • Don’t work when server process fails
  • Handle exception and connection state control is mandatory in the code
What is Application Continuity

• Application Continuity (AC)
  • Application-independent solution that enables recovery of work from an application perspective (planned and unplanned outages)
  • Introduced with Oracle Database 12c Release 1
  • Support for XA data sources introduced with Oracle Database 12c Release 2
  • AC is a feature of
    • Oracle JDBC thin driver (not supported for JDBC thick driver)
    • Oracle Database 12c
What is Application Continuity

• Application Continuity (AC) licensing model
  • Licensing AC has to be done only on the database site (JDBC drivers are free)
  • On premises
    • Enterprise Edition and Exadata with Active Dataguard option or RAC option
    • According to Oracle Database Licensing guide, it’s not available for OracleSE2 RAC databases.
  • Oracle Cloud
    • Enterprise Edition Extreme Performance option
    • Exadata Cloud Service

<table>
<thead>
<tr>
<th>Feature/Option/Pack</th>
<th>SE2</th>
<th>EE</th>
<th>EE-Exa</th>
<th>DBCS SE</th>
<th>DBCS EE</th>
<th>DBCS EE-HP</th>
<th>DBCS EE-EP</th>
<th>ExaCS</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Continuity</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

EE and EE-Exa: Requires the Active Data Guard option or the Real Application Clusters option
Part 2: Data source types usable with AC
Data source types usable with AC

- **Principles**
  - JDBC Thin client mandatory
  - Service correctly defined on the server side

- **Java Data Source to get specific JDBC connections**
  - `oracle.jdbc.replay.OracleDataSourceImpl`
  - `oracle.jdbc.replay.OracleConnectionPoolDataSourceImpl`
  - `oracle.jdbc.replay.driver.OracleXADataSourceImpl`

- Failover type.
  - `failoverttype TRANSACTION`
  - `commit_outcome TRUE`
  - `retention 86400`
  - `replay_init_time 900`
  - `failoverretry 20`
  - `failoverdelay 2`
  - `notification TRUE`

Note: If you set `-failoverttype` to `TRANSACTION`, then you must set `-commit_outcome` to `TRUE`. To enable Application Continuity for Java, set this parameter to `TRANSACTION`. To enable TAF for OCI, set this parameter to `SELECT` or `SESSION`. 
Data source types usable with AC

• Usages
  • Standalone JDBC application
  • Standalone JDBC application with UCP Pooled Connections
  • Weblogic Application
    • With UCP Data source
    • With GridLink Data source
Data source types usable with AC

• Standalone JDBC application
  • Need to define a « replayable » zone in the code
    • Cast connection to oracle.jdbc.replay.ReplayableConnection
    • Delimit replayable zone with beginRequest() and endRequest()
  • Disable autocommit

```java
.../
((oracle.jdbc.replay.ReplayableConnection)c).beginRequest();
    pstmt=c.prepareStatement(updsq1);
    c.setAutoCommit(false);
    for (int i=0;i<numValue;i++) {
        pstmt.setInt(1,i);
        pstmt.executeUpdate();
    }
    c.commit();
    // End of the Callback.
((oracle.jdbc.replay.ReplayableConnection)c).endRequest();
.../
```
Data source types usable with AC

• Standalone JDBC application with UCP Pooled Connections
  • Set class name for the Connection Factory to
    • oracle.jdbc.replay.OracleDataSourceImpl
    • oracle.jdbc.replay.OracleConnectionPoolDataSourceImpl
  • Replayable zone is implicitly defined between when you get connection and until the connection ends
import oracle.ucp.jdbc.PoolDataSourceFactory;
import oracle.ucp.jdbc.PoolDataSource;

String CONN_FACTORY_CLASS_NAME = "oracle.jdbc.replay.OracleConnectionPoolDataSourceImpl";
String ONS_CONFIG = "nodes=ocm-rac1:6200,ocm-rac2:6200";

PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName(CONN_FACTORY_CLASS_NAME);
pds.setURL(DB_URL);
pds.setUser(DB_USER);
pds.setPassword(DB_PASSWORD);
pds.setInitialPoolSize(5); pds.setMinPoolSize(5); pds.setMaxPoolSize(20);
pds.setONSConfiguration (ONS_CONFIG);
pds.setFastConnectionFailoverEnabled(true); // Enables FCF for the DataSource

Connection conn;
conn=pds.getConnection(); // implicit start of the replayable zone

Statement stmt=conn.createStatement();

// WORK

conn.close(); // End of replayable zone
Data source types usable with AC

• Weblogic Application With UCP Data source
  • Create a UCP DataSource in Weblogic Server
Data source types usable with AC

- Webloglic Application With UCP Data source
  - Don’t forget to configure ONS and enable FCF at the driver level (not enabled by default)
    - ONSConfiguration
    - FastConnectionFailoverEnabled

Scan name in the JDBC URL

ONS Configuration to subscribe ONS Service

Fast Connection Failover activation

Connection class delivered by PoolDataSource
Data source types usable with AC

• Basic JSP example

```jsp
<%@ page import="javax.naming.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<%@ page import="oracle.ucp.jdbc.PoolDataSource" %>

<body>
try {
  Context ctx = new InitialContext();
  Hashtable ht = new Hashtable();
  ht.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.WLInitialContextFactory");
  ht.put(Context.PROVIDER_URL, "t3://wls12.localdomain:7003");
  ctx = new InitialContext(ht);
  PoolDataSource pds=(PoolDataSource)ctx.lookup("jdbc/ucp_rac_ac");
  Connection conn=pds.getConnection(); // implicit start of the replayable zone
  Statement stmt=conn.createStatement();
  // WORK
  conn.close(); // End of replayable zone
}
catch(Exception e) {e.printStackTrace();}%>
</body>
```
Data source types usable with AC

- Weblogic Application With GridLink Data source
  - What is a Gridlink Data Source?
    - Replacement for old multiple Data Sources
    - Integrated for Weblogic and RAC Clusters

Dynamic load balancing of requests to RAC Nodes

Web session and transaction affinity for data locality

Application Continuity enabled
Data source types usable with AC

• Grid Link Data Source features
  • Configurability / Manageability
    • Single Data Source per Weblogic cluster
    • Use SCAN Addresses: automatically detects new nodes in the RAC cluster, no weblogic configuration change

• Availability
  • Failure notification with ONS
  • New connections to surviving nodes
  • Replayable connections
Data source types usable with AC

• Grid Link Data Source features
  • Performance
    • Runtime Connection load Balancing distributes connection to the least busy nodes
    • Integration with Database Resident Connection Pooling (DRCP)
    • Web session affinity reduces cluster wait time
Data source types usable with AC

• Grid Link Data Source features

AC support for XA-capable data sources is limited to local transactions and local transactions that are promotable to global/XA transactions. (=> Only from Oracle RAC 12.2)
Data source types usable with AC

• Grid Link Data Source features

RCLB (Runtime Connection Load-Balancing) is configured at the RAC service level!

```
-rlbgoal {NONE|SERVICE_TIME|THROUGHPUT}
```
Data source types usable with AC

• Basic JSP example

```jsp
<%@ page import="javax.naming.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="javax.sql.*" %>
<%@ page import="java.util.*" %>

<body>
<%
try {
    Context ctx = new InitialContext();
    Hashtable ht = new Hashtable();
    ht.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.WLInitialContextFactory");
    ht.put(Context.PROVIDER_URL, "t3://wls12.localdomain:7003");
    ctx = new InitialContext(ht);
    DataSource ds=(DataSource)ctx.lookup("jdbc/gridlink_rac_ac");
    Connection conn=ds.getConnection(); // implicit start of the replayable zone
    // WORK
    conn.close(); // implicit end of the replayable zone
} catch(Exception e) {e.printStackTrace();}%>

</body>
```
Part 3: How does it work? ...
Nothing is worth a demo :)}
Demo Time: Agenda

• Environment description
  • Code
  • Lab environment

• Tests:
  • Crash one rac-node at a time + recover it
  • Switchover to standby database
Demo Time: Code

```java
try {
    // Here we have to use OracleDataSourceImpl in replacement of OracleDataSource.
    OracleDataSourceImpl ocpds = new OracleDataSourceImpl();
    ocpds.setURL("jdbc:oracle:thin:@(DESCRIPTION_LIST = "
        "(LOAD_BALANCE = off) "+
        "(FAILOVER = on)" +
        "(DESCRIPTION = "+
        "(CONNECT_TIMEOUT = 5) (TRANSPORT_CONNECT_TIMEOUT = 3) (RETRY_COUNT = 3)"+
        "(ADDRESS_LIST = "+
        "(LOAD_BALANCE = on)" +
        "(ADDRESS = (PROTOCOL = TCP) (HOST = labrac01scan.rac.labo.internal.stepi.net) (PORT = 1521))" +
        "(CONNECT_DATA = (SERVICE_NAME = raclab_pdb_ha_test)) "+
        ")" +
        "(DESCRIPTION = "+
        "(CONNECT_TIMEOUT = 5) (TRANSPORT_CONNECT_TIMEOUT = 3) (RETRY_COUNT = 3)"+
        "(ADDRESS_LIST = "+
        "(LOAD_BALANCE = on) (ADDRESS = (PROTOCOL = TCP) (HOST = labvms01n01.labo.internal.stepi.net) (PORT = 1521))"+
        "(CONNECT_DATA = (SERVICE_NAME = raclab_pdb_ha_test))"+
        ")" +
        ")
    ");
    ocpds.setUser("vanpupi");
    ocpds.setPassword("******");
    vanpupi_ha_test self = new vanpupi_ha_test();
    // Get a connection from the data source and prepare the statement
    conn = ocpds.getConnection();
    // Get instance name from the data source and prepare the statement
    System.out.println("Connected to Instance Name = "+self.getInstanceName(conn));
    // Retrieve the max seq value from the rac_test table
    sequence=self.getSequence(conn);
    System.out.println("Max sequence id in rac_test table is:"+sequence);
    System.out.println(" ");
    // test also with alter session. Eg put date format and insert that as a string to see if session context moves too
    // Prepare the sql statements
    String inssql = "insert into transtest (select ?,?,?,a.host_name,a.instance_name from (select host_name,instance_name from v$instance) a)";
    String updsql = "UPDATE RAC_TEST SET SEQ_VALUE=?, LAST_UPDATE_DATE=SYSDATE WHERE SEQ_VALUE=?";
```
Demo Time: Code

• Oracle JDBC-driver
• TNS-entry
• Nothing fancy / Standard DB connection code
Demo Time: Code

```java
/* Beginning of the callback */
* AutoCommit has to be disabled in the callback part.*
*/

((oracle.jdbc.replay.ReplayableConnection)conn).beginRequest();
conn.setAutoCommit(false);

PreparedStatement pstmtins = null;
PreparedStatement pstmtupd = null;
pstmtins=conn.prepareStatement(inssql);
pstmtupd=conn.prepareStatement(upssql);

while (!interrupted) {
    // insert ps
    pstmtins.setInt(1, i + 1);
pstmtins.setDate(2, new Date(System.currentTimeMillis()));
pstmtins.setString(3, connectionMethod);
pstmtins.executeUpdate();
    System.out.println("Inserted record: "+i);

    // update ps
    pstmtupd.setInt(1, sequence + 1);
pstmtupd.setInt(2, sequence);
pstmtupd.executeUpdate();
    System.out.println("Updated rac_test record: "+sequence + " to "+(sequence+1));
    i++;
    sequence++;
    try {
        Thread.sleep(interval);
    } catch (InterruptedException e) {} 
}
conn.commit();
```
Demo Time: Code

• Disable Autocommit
• While loop is meant for demo
• All surrounded in a try - catch block
Demo Time: Lab

- cdbrac
  - Cluster db
  - on nodes 1 and 2
- oraclrac
  - Cluster db
  - on nodes 3 and 4 (not used in this demo)
- cdbsi runs on another vm and is a single instance db
- Cdbsi is a standby database from cdbrac
Demo Time: Lab

User Creation:

SQL> grant connect, resource to vanpupi;
Grant succeeded.

SQL> GRANT EXECUTE ON DBMS_APP_CONT TO vanpupi;
Grant succeeded.

SQL> grant keep date time to vanpupi;
Grant succeeded.

SQL> grant keep sysguid to vanpupi;
Grant succeeded.

SQL>
Demo Time: Lab

Service Creation:

• Rac

```bash
srvctl add service -db cdbrac -service raclab_pdb_ha_test
    -serverpool poola
    -cardinality uniform -pdb pdb
    -failovertime TRANSACTION -commit_outcome TRUE -replay_init_time 10
    -retention 86400 -notification TRUE -failoverretry 50
```

• Single instance

```sql
DBMS_SERVICE.CREATE_SERVICE('raclab_pdb_ha_test','raclab_pdb_ha_test');
```
Demo Time: Lab

• Service modification

```sql
DECLARE
  params dbms_service.svc_parameter_array;
BEGIN
  params('FAILOVER_TYPE')            := 'TRANSACTION';
  params('REPLAY_INITIATION_TIMEOUT') := 1800;
  params('RETENTION_TIMEOUT')        := 86400;
  params('FAILOVER_DELAY')           := 10;
  params('FAILOVER_RETRIES')         := 50;
  params('commit_outcome')           := 'true';
  params('aq_ha_notifications')      := 'true';
  DBMS_SERVICE.MODIFY_SERVICE('raclab_pdb_ha_test',params);
END;
/
```

• 12c services will not start while manually restart pdb (Doc ID 2006021.1)
Demo Time: Lab

• Trigger

CREATE OR REPLACE TRIGGER startDgServices after STARTUP ON PLUGGABLE DATABASE
DECLARE
  db_role VARCHAR(30);
  db_open_mode VARCHAR(30);
  db_hostname_si VARCHAR(50);
BEGIN
  SELECT DATABASE_ROLE, OPEN_MODE INTO db_role, db_open_mode FROM V$DATABASE;
  select host_name into db_hostname_si from v$instance;
  IF db_role = 'PRIMARY' AND db_open_mode LIKE '%READ WRITE%' and lower(db_hostname_si) like '%labvms01n01%' THEN DBMS_SERVICE.START_SERVICE('raclab_pdb_ha_test'); END IF;
END;
/

### Demo Time: Lab

```sql
SQL> desc rac_test
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
LAST_UPDATE_DATE         DATE
SEQ_VALUE                                          NUMBER

SQL> desc transtest
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
ID                                                 NUMBER
INSDATE                                            DATE
CONNECTIONMETHOD                                   VARCHAR2(50)
HOST_NAME                                          VARCHAR2(50)
INSTANCE_NAME                                      VARCHAR2(50)

SQL> desc spy_table_insert
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
D                                                  DATE
INST_NAME                                          VARCHAR2(16)
ID                                                 NUMBER

SQL> desc spy_table_update
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
D                                                  DATE
INST_NAME                                          VARCHAR2(16)
ID                                                 NUMBER
```
Test 1

- Start update + insert transactions
- Crash instance 1

- Expected result: insert and update keep continuing
Test 2

- Keep update + insert transactions running
- Crash instance 2

- Expected result: insert and update keep continuing
Test 3

• Keep update + insert transactions running
• Switchover the primary database to the standby database

• Expected result: insert and update keep continuing
Demo Time!

Be prepared for it to fail, because that's what demo's do
Use Cases

• Transactions are replayed
• Protection!
• Peace of mind
• Easier patching
• Backup-system ready available, even in the cloud
Part 4: What about performances?
Performance Overhead

• What is the performance overhead on :
  • Database Server
    • PGA
    • CPU
  • Client side
    • Memory used by Java application
    • CPU used by Java application
    • Number of classes loaded
Performance Overhead

• Test details
  • Standalone java application (Heap size: 512Mb, -Xms=512M –Xmx=512M)
  • 50000 updates to process through one connection
    • First test: inside a replayable zone
    • Second test: no replayable zone
  • ojdbc7.jar (12.1 jdbc library)
  • No FCF configuration (not relevant for these tests)
• Database Side
  • 2 nodes RAC 12.1.0.2 cluster
    • An AC service configured with RCLB enabled (SERVICE_TIME)
Performance Overhead

• Results
  • Database side

<table>
<thead>
<tr>
<th></th>
<th>Session PGA Max Size</th>
<th>CPU Used by this session</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC Enabled</td>
<td>3961144</td>
<td>34859</td>
</tr>
<tr>
<td>AC not Enabled</td>
<td>3747576</td>
<td>35138</td>
</tr>
</tbody>
</table>
Performance Overhead

• Results
  • Application side: Number of classes loaded

![Diagram showing the number of classes loaded for AC Enabled and AC Disabled scenarios.](image-url)
Performance Overhead

• Results
  • Application side: CPU
Performance Overhead

• Results
  • Application side: Memory

[Graphs showing memory usage with AC Enabled and AC Disabled]
Performance Overhead

• Results summary
  • On the database side
    • No performance overhead (CPU and PGA)
  • On the application side
    • No CPU over consumption
    • No more classes loaded
    • Increased memory on the application side
      → Need to reconsider heap size for existing java servers
Questions?