Practice 10

Using Application Continuity

Practice Overview

In this practice, you will demonstrate how Application Continuity can be used in an application to recover from a RAC instance outage.

Practice Assumptions

- The practice assumes that you have the Oracle RAC database up and running in the virtual machines `srv1` and `srv2`.
- You downloaded the filed "**ac_demo.zip**" from the downloadable resources of this lecture. This compressed file has a simple Java program that connects to an Oracle database using the known database user `scott`.
  `Scott` user is installed automatically in Oracle databases when you install the Examples schemas in it.

Using Application Continuity on a Java Application

In the following steps, you will run a Java application that does not have the application continuity enabled in it. You will notice its reaction when the RAC instance the application is connected to crashes. You will then run the same application with the application continuity enabled and perform the same test on it.

1. Using Winscp, login to `srv1` as oracle and copy the compressed file `ac_demo.zip` to the directory

```
/home/oracle/scripts
```

2. Open a Putty session connected to `srv1` as oracle. This Putty session window will be referred to in this practice as the **client window**.

3. Unzip the compressed file then change the current directory to the decompressed directory

4. In the extracted directory, you will see two scripts: `runnoreplay` and `runreplay`. They both execute the same application (`actest.jar`). The only difference between them is that each script references a different properties file.

```
/home/oracle/scripts/
unzip ac_demo.zip
cd ac_demo
ls -al
cat runnoreplay
cat runreplay
```

5. Examine the properties files. Observe that the only difference between the two properties files is the data source specification. The NoReplay file uses the standard 12.1 data source (`OracleDataSource`) and the Replay file uses the replay data source (`OracleDataSourceImpl`).

In other words, the first file does not take advantage of the application continuity. The second has the application continuity enabled.

```
cat actest_noreplay.properties
cat actest_replay.properties
```

6. Grant execution access privilege to `oracle` on the script files.

```
chmod u+x runnoreplay
chmod u+x runreplay
```

7. Create and start a database service for use in conjunction with application continuity.

```
srvctl add service -db rac -service acsrv -preferred rac1 -available rac2 -
failovertype TRANSACTION -commit_outcome TRUE -failoverretry 30 -failoverdelay 10
-retention 86400 -replay_init_time 1800 -notification TRUE

srvctl start service -db rac -service acsrv
```

8. Using SQL*Plus, connect to `rac` as `system` and unlock `scott` account.

```
sqlplus system/oracle@rac
alter user scott identified by tiger account unlock;

# test:
conn scott/tiger@//srv1:1521/acsrv.localdomain
```

9. Open another Putty session connected to srv1 and connect to it as `oracle`. This `window will be` referred to in this practice as **admin window**.

10. Configure the prompt in the admin window as follows. This helps you to distinguish between the admin and client Putty session windows.

```
export PS1='[ADMIN $]'
```

11. In the **client** window, execute the `noreplay` script.

Observe that when the application is running, it displays a periodic status messages.

```
./runnoreplay
```

12. In the **admin** window, verify that the sessions created by the application are connected to `rac1`.

```
sqlplus sys/oracle@rac as sysdba
SELECT DISTINCT INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';
```

13. Crash `rac1`.

```
pkill -9 -f ora_pmon_rac1
```

14. Observe how the application reacts.

The application should report errors as a result of the crash. This result is expected as application continuity is disabled.

15. In the **client window**, press **[Ctrl] + [C]** to abort the application.

Test the Java when Application Continuity is enabled

Now, you will follow the same procedure to test the reaction of the application when the application continuity is enabled.

16. In the **admin** window, verify that the crashed instance is up again.

```
srvctl status database -d rac
```

17. Verify that `acsrv` service is running in `rac2` now.

```
srvctl status service -d rac -s acsrv
```

18. In the **client** window, run the script where application continuity is enabled (`runreplay`).

The application executes exactly the same way as the application that has the application continuity disabled.

```
./runreplay
```

19. In the **admin window**, verify that the sessions created by the application are connected to `rac2`.

```
sqlplus sys/oracle@rac as sysdba

SELECT DISTINCT INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';
```

20. In the **admin window**, connect to `srv2` and crash `rac2` instance.

```
ssh srv2
pkill -9 -f ora_pmon_rac2
```

21. Observe the reaction of the application execution.

You should notice that the application continues its operation as normal without any noticeable interruption or error. This is the expected behavior when the application continuity is on.

22. In the admin window, exit from the session connected to `srv2`.

```
exit
```

23. Verify that the sessions created by the application have migrated to `rac1`.

```
sqlplus sys/oracle@rac as sysdba
SELECT DISTINCT INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';

srvctl status service -d rac -s acsrv
```

24. Perform the following cleanup steps:

a) Abort the application; by pressing on [Ctl] + [c]

b) Stop and delete the services.

```
srvctl stop service -d rac -s acsrv
srvctl remove service -d rac -s acsrv
```

c) Close all windows opened for this practice.

Summary

Application Continuity enhances the application availability response when a database instance in a RAC database crashes.