Practice 15: Using Resource Manager with CDB and PDBs

Practice Overview

In this practice, you will create two CDB Resource Manager plans and associated directives to limit the CPU time used by two PDBs.

Following are the resource manager plans that you will created in this practice:

Resource Manager Plan	PDB	Shares Directive
BALANCED_PLAN	**PDB1**	1
	PDB2	1
BIASED_PLAN	**PDB1**	4
	PDB2	1

You will follow the following testing procedure:

- Enable the `BALANCED_PLAN`
- Apply load on the CPU in `PDB1` and `PDB2` in the same time.
- Enable the `BIASED_PLAN`
- Apply load on the CPU in both PDBs.
- Compare the results that you obtained when enabling those plans.

Practice Assumptions

- `CDB1` (in `srv1`) database is up and running.

Create and Examine CDB Resource Manager Plans and Directives

A. Create CDB Resource Manager Plans and Directives

In this section of the practice, you will create the resource manager plans and directives as shown in the table in the preview section.

1. Create a Pluggable Database named as `PDB1` from the seed.

```
sqlplus / as sysdba
CREATE PLUGGABLE DATABASE pdb1 ADMIN USER pdb1admin IDENTIFIED BY oracle;
ALTER PLUGGABLE DATABASE pdb1 OPEN;
ALTER PLUGGABLE DATABASE pdb1 SAVE STATE;
```

2. Verify that `PDB1` and `PDB2` databases are available and opened in read/write mode.

```
col name format a10
SELECT NAME, OPEN_MODE FROM V$PDBS;
```

3. Connect to `PDB1` as `SYSTEM` user and create the following PL/SQL procedure.

This procedure simply consumes purely the CPU because it only executes arithmetic operation. When you execute it, you just pass the number of loops on which you want to apply the load on the CPU. At the end of its execution, it prints out the elapsed execution time.

```
conn system/oracle@pdb1

CREATE OR REPLACE PROCEDURE SYSTEM.HIT_CPU ( P_LOOP NUMBER )
IS
 V_BEGIN DATE;
 V NUMBER;
BEGIN
 V_BEGIN := SYSDATE;
 FOR I IN 1..P_LOOP*10000 LOOP
  V := SQRT(I);
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('ELAPSED TIME: ' || ROUND((SYSDATE-V_BEGIN)*24*60*60,2 )||
' seconds');
END HIT_CPU;
/
```

4. Connect to `PDB2` as `SYSTEM` user and create the same PL/SQL procedure.

```
conn system/oracle@pdb2
-- copy the procedure code from the previous step.
```

5. Create the resource plan `BALANACED_PLAN` and configure the plan directives in it.

This plan gives one share to both `PDB1` and `PDB2`. This means that they both have the same priority to receive the CPU time resource.

```
ALTER SESSION SET CONTAINER = CDB$ROOT;
EXEC DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
EXEC DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
EXEC DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN('BALANACED_PLAN', 'One share to PDB1
and PDB2');
EXEC DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE('BALANACED_PLAN', 'PDB1',
shares => 1);
EXEC DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE('BALANACED_PLAN', 'PDB2',
shares => 1);
```

6. Create the resource plan `BIASED_PLAN` and configure the plan directives in it.

This plan gives four shares to `PDB1` and one share to `PDB2`.

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN('BIASED_PLAN', 'PDB1 is given high
priority');
EXEC DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE('BIASED_PLAN', 'PDB1',
shares => 3);
EXEC DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE('BIASED_PLAN', 'PDB2',
shares => 1);
EXEC DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
EXEC DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
```

7. Make sure the plans and their associated directives were successfully created.

```
col plan format a20
SELECT Plan from CDB_CDB_Rsrc_Plans
WHERE CON_ID = 1 AND PLAN IN ('BALANACED_PLAN','BIASED_PLAN')
ORDER BY 1;

col pluggable_database format a30
SELECT PLAN, PLUGGABLE_DATABASE, SHARES
FROM  CDB_CDB_RSRC_PLAN_DIRECTIVES
WHERE CON_ID = 1 AND PLAN IN ('BALANACED_PLAN','BIASED_PLAN')
ORDER BY 1, 2;
```

B. Test the Created Resource Manager plans

In this section of the practice, you will test the created resource manager plans. You will create four sessions connection to the database. Two sessions connected to `PDB1` and the other two sessions connected to `PDB2`. You will then enable the plans that you created and apply load on the CPU from all the sessions.

8. Enable the CDB plan `BALANACED_PLAN`.

```
-- you need to be connected to the root:
connect / AS SYSDBA

ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = BALANACED_PLAN;

col name format a20
SELECT NAME FROM V$RSRC_PLAN WHERE CON_ID = 1;
```

9. Connect as the `SYSTEM` user in `PDB1` and set `SERVEROUTPUT` variable to `ON`

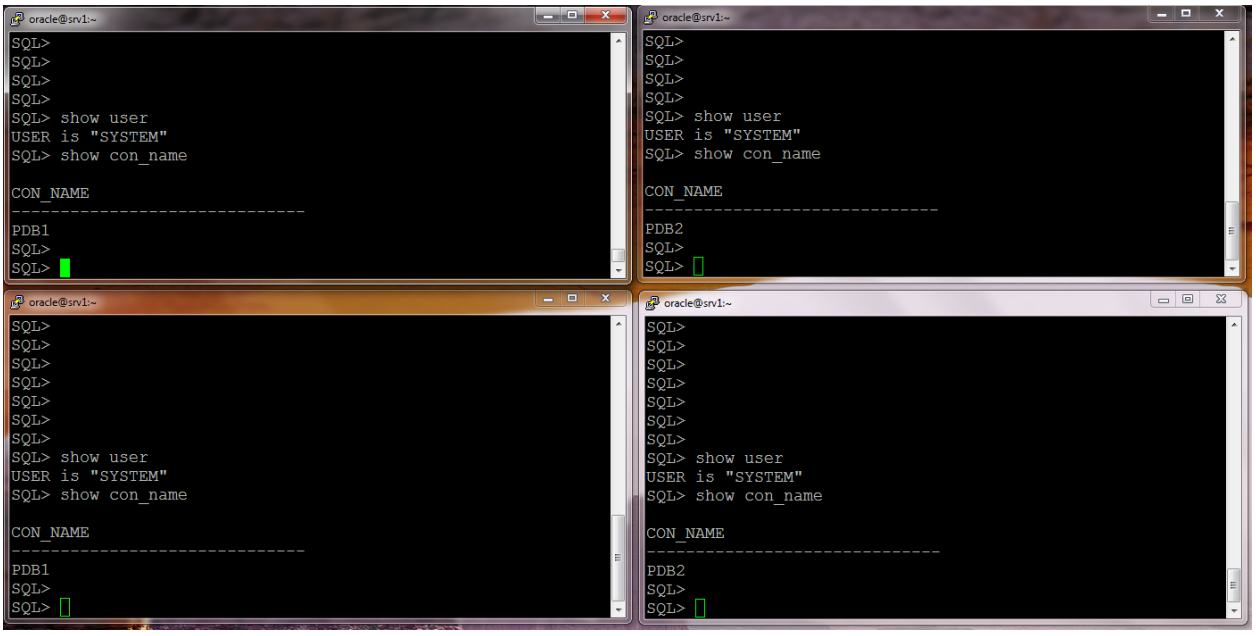
```
connect system/oracle@pdb1
SET SERVEROUTPUT ON
```

10. Open a new Putty session window and connect as the `SYSTEM` user to `PDB1` and set `SERVEROUTPUT` variable to `ON`.

11. Open another **two** Putty sessions, connect as the `SYSTEM` user to `PDB2`, and set `SERVEROUTPUT` variable to `ON`.

```
sqlplus system/oracle@pdb2
SET SERVEROUTPUT ON
```

12. Arrange the Putty windows so that they appear nearly the same as in the following screenshot:



13. Copy the following command and paste it into all the Putty windows without pressing the ENTER button.

```
EXEC HIT_CPU(10000)
```

14. Click on every Putty window and press ENTER. Try to do it as soon as possible.

15. When the procedure finishes execution in all the windows, compare the elapsed execution time in all the windows.

You will observe that they finish executing the procedure in nearly the same time period. This is the expected behavior because each PDB is receiving one share of CPU.

16. In one of the windows that is connected to `PDB1`, connect to the root and change the resource manager plan to `BIASED_PLAN`. Then connect back to `PDB1` as `SYSTEM` user.

```
connect / AS SYSDBA

ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = BIASED_PLAN;
SELECT NAME FROM V$RSRC_PLAN WHERE CON_ID = 1;


connect system/oracle@pdb1
SET SERVEROUTPUT ON
```

17. Perform the same test again: copy the following command, paste it into all the Putty windows, then execute it in all the windows in nearly the same time.

```
EXEC HIT_CPU(10000)
```

18. Wait for the procedure to finish its execution, and compare its elapsed execution time in all the windows.

The procedure is expected to run faster in `PDB1` because it has been allocated higher shares than `PDB2`. However, the difference is not five times slower simply because once the procedure is executed in `PDB1`, all CPU time will be allocated to `PDB2`.

Note: if in your case the procedure finishes execution in nearly the same time in all the windows, this could be because the hosting machine has so high speed CPU that the procedure execution in all the Putty windows could not hit the CPU to its maximum power. One method to handle this situation is to create additional Putty sessions connected to `PDB1` and `PDB2`.

19. Clean up procedure:

a. Set the CDB plan back to its default.

```
conn / as sysdba


ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = '';


col name format a30
SELECT NAME FROM V$RSRC_PLAN WHERE CON_ID = 1;
```

b. Delete the resource plans.

```
begin
 DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
 DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
 DBMS_RESOURCE_MANAGER.DELETE_CDB_PLAN('BALANACED_PLAN');
 DBMS_RESOURCE_MANAGER.DELETE_CDB_PLAN('BIASED_PLAN');
 DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
 DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
end;
/
```

c. Exit from all the Putty sessions.

Summary

Using the CDB resource manager, you can control how the CPU power is distributed among the PDBs.