Practice

Configuring Databases for Shared Server

Practice Target

In this practice you will configure shared server processes in the database in `srv1` for the `PDB1` service.

Specifically, you will learn:

- Test the default shared server configuration in `srv1`
- Configure the shared server processes for `PDB1` service
- Monitor shared server processes

Practice Assumptions

This practice assumes that you have the virtual machine `srv1` up and running from the **CDB**.

Testing the Default Shared Server Configuration in `srv1`

In this section of the practice, you will try connecting to the shared server processes in the database in `srv1`.

1. Open a Putty session to `srv1` as `oracle`

2. Connect to the root container as `sys`

```
sqlplus / as sysdba
```

3. Check out the parameters that enable the shared server processes in the database.

   The database can accept connections to the Shared Server processes only to the service `oradbXDB`. This configuration was added by the `dbca`.

```
col NAME for a20
col VALUE for a35
SELECT NAME, VALUE FROM V$PARAMETER
WHERE NAME IN ('shared_servers', 'dispatchers');
```

4. Exit from SQL*Plus

```
quit
```

Let's try connecting to the database using shared server configuration and check what would be the database behavior.

5. Open the `tnsnames.ora` file in the `vi` editor. Add the following connection configuration in it.

```
vi /u01/app/oracle/product/19.0.0/db_1/network/admin/tnsnames.ora
```

```
PDB1_shared =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = srv1)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = pdb1.localdomain)
      (SERVER = SHARED)
    )
  )
```

6. Try connecting to the configured shared connection.

   The connection attempt failed with the following error because there is no shared server process configured for the database service provided in the connection name.

   `ORA-12520: TNS:listener could not find available handler for requested type of server`

```
sqlplus soe/ABcd##1234@pdb1_shared
```

Configuring Shared Server Processes for PDB1 Service

In this section of the practice, you will configure shared server processes for PDB1 service in the database in srv1.

7. Open SQL Developer and connect to the root container (oradb) in srv1 as SYS. This session will be used for monitoring the shared server processes.

8. In SQL Developer window, submit the following query to retrieve information about the dispatcher configurations existing in the database.

You should see only one dispatcher configuration. It is linked to the oradbXDB service.

```
SELECT * FROM V$DISPATCHER_CONFIG ;
```

9. In the Putty session, connect to the root container as sys

```
sqlplus / as sysdba
```

10. Submit the following statement to configure another dispatcher configuration in the database for the PDB1 service. Configure only one dispatcher in this configuration.

By setting the INDEX to 1, this statement will add a new dispatcher configuration and keep the existing one because the existing one INDEX is zero.

Observe that the parameter is dynamic. No need to restart the instance after setting the parameter.

```
ALTER SYSTEM SET DISPATCHERS
='(INDEX=1)(PROTOCOL=TCP)(DISPATCHERS=1)(SERVICE=pdb1.localdomain)' SCOPE = BOTH;
```

11. In SQL Developer session, verify that the second dispatcher configuration is added.

You have a new dispatcher configuration for the protocol TCP and the service PDB1. Check out the dispatcher configuration attributes.

Observe that we have two dispatcher processes. One for the oradbXDB and one for the PDB1.

The parameter VALUE in V$PARAMETER is not literally the same as the value passed to the statement above, still it matches the result.

```
SELECT * FROM V$DISPATCHER_CONFIG ;
SELECT NAME, STATUS, MESSAGES,IDLE,BUSY FROM V$DISPATCHER;
SELECT VALUE FROM V$PARAMETER WHERE NAME='dispatchers';
```

12. In Putty session, exit from SQL*Plus session, wait for a couple of minutes then check if the dispatcher process registers itself in the Listener.

Observe that a port number is assigned to the process. If we need to set a specific port number to the process, we need to set the ADDRESS attribute to the DISPATCHERS value together with the PORT attribute.

```
exit

lsnrctl service
```

13. Try connecting to the shared server process. It should succeed this time.

We have configured the shared processes in the database by only setting the parameter DISPATCHERS. We did not need to set the parameter SHARED_SERVERS because it is already set to a non-zero value. However, you might consider setting this parameter to a value higher than one.

Note: In some rare cases, the listener returns the following error. In my experience, restarting the listener resolved the ssue.

ORA-12520: TNS:listener could not find available handler for requested type of server

```
sqlplus soe/ABcd##1234@pdb1_shared
```

14. Try connecting to the shared server process using easy connection format. It should succeed.

We still use the listener port number in the connection string, not the dispatcher port.

```
conn soe/ABcd##1234@"(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=srv1) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=pdb1.localdomain) (SERVER=SHARED)))"
```

15. In the SQL Developer session, check the SERVER column value in V$SESSION for the SOE user.

SERVER equals to NONE. Which means it was connected to a shared process but it is idle. The next step is to verify this statement.

```
SELECT DISTINCT SERVER FROM V$SESSION WHERE USERNAME='SOE';
```

16. In the Putty session, run the following code. The code keeps the session busy running some CPU-intense operation.

```
DECLARE
 n number;
begin
 while (true) loop
  n := dbms_random.value(1,100);
 end loop;
END;
/
```

17. In the SQL Developer session, check the SERVER column value in V$SESSION for the SOE user.

SERVER equals to SHARED.

```
SELECT DISTINCT SERVER FROM V$SESSION WHERE USERNAME='SOE';
```

18. In the Putty session, cancel the running operation by pressing on [Ctl]+[c]

Monitoring Shared Server Processes

In this section of the practice, you will monitor the Shared Server processes.

19. In the hosting PC, open a command-line window and start Swingbench.

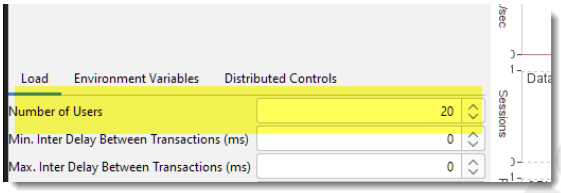
```
cd C:\swingbench2.6\winbin
swingbench.bat
```

20. In Swingbench, change the connection string to the following. Test the connection to it to make sure that this connection works. Enter the `srv1` IP address in its placeholder.

We could configure a shared connection in tnsnames.ora file.

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=<srv1 ip>) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=pdb1.localdomain) (SERVER=SHARED)))
```

21. Change the number of users to 20. This is the number of to-be-connected client sessions. They connect concurrently.



22. Run the Swingbench sessions.

23. In SQL Developer session, submit the following query to retrieve information about the dispatchers running in the system. Keep running the query multiple times.

Observe that we have two dispatchers. One dispatcher process for each dispatcher configuration we have in the system.

Observe that the `MESSAGES` for the `D001` keeps progressing as the requests are being sent from Swingbench sessions.

```
SELECT NAME, STATUS, MESSAGES, IDLE, BUSY FROM V$DISPATCHER;
```

24. Submit the following query multiple times to retrieve the value of `SERVER` column in `V$SESSION` for the client sessions connected to shared server processes.

The `SERVER` values for the sessions connected to shared processes are either `SHARED` or `NONE`

```
SELECT SERVER, COUNT(*) FROM V$SESSION WHERE USERNAME='SOE' GROUP BY SERVER;
```

25. Submit the following query multiple times to retrieve the information about the queue.

Observe that one of the dispatcher queue keeps progressing while the other is idle. The progressing one is the one that receives the request from Swingbench.

```
SELECT TYPE, QUEUED, WAIT, TOTALQ, CON_ID FROM V$QUEUE;
```

26. Submit the following query multiple times to retrieve the information about the dispatcher processes.

The `V$PROCESS` view provides information about the processes running in the database at the OS level. It also provides performance information like the PGA allocated for the processes and CPU time consumed by the processes.

Dispatcher process names always has the format `Dnnn`, where *nnn* is a three-digit integer.

In our case, we have `D000` for the XML DB service dispatcher and `D001` for the `PDB1` dispatcher.

```
SELECT PID, SOSID, SPID, STID, EXECUTION_TYPE, PNAME, USERNAME, PGA_USED_MEM,
PGA_ALLOC_MEM, CPU_USED
FROM V$PROCESS WHERE PNAME LIKE 'D0%';
```

27. Submit the following query multiple times to retrieve the information about the shared server processes.

Shared server process names always has the format `Snnn`, where *nnn* is a three-digit integer.

In our case, the database assigns more than 10 processes to server the 20 Swingbench sessions. The database may increase the number of shared server processes as the load goes on.

```
SELECT PID, SOSID, SPID, STID, EXECUTION_TYPE, PNAME, USERNAME, PGA_USED_MEM,
PGA_ALLOC_MEM, CPU_USED
FROM V$PROCESS WHERE PNAME LIKE 'S0%';
```

The database may increase the number of shared server processes as the load goes higher. That would be fine if all the users connected to the database use the same dispatcher. But, if we have other user types that might connect to the database using an alternative connection (for example batch processing session overnight), the shared server processes may compete with those sessions. In such scenarios, we prefer to set a limit to the shared server processes spawned by the database. We might allow the shared processes to go high in the daytime and low in the nighttime.

The best method to control the maximum number of shared processes in a database server is by setting the parameter `MAX_SHARED_SERVERS`. In the following steps, you will set this parameter and study its impact on the database.

28. Stop the Swingbench sessions.

29. Set the `MAX_SHARED_SERVERS` to 5.

```
conn / as sysdba
ALTER SYSTEM SET MAX_SHARED_SERVERS = 5 SCOPE=BOTH;
```

30. Start Swingbench sessions.

31. Monitor the server processes by running the following query multiple times.

The number of shared processes keep going down till they reach to five processes.

```
SELECT PID, SOSID, SPID, STID, EXECUTION_TYPE, PNAME, USERNAME, PGA_USED_MEM,
PGA_ALLOC_MEM, CPU_USED
FROM V$PROCESS WHERE PNAME LIKE 'S0%';
```

Cleanup

32. Stop Swingbench sessions and exit from Swingbench.

33. Shutdown `srv1`

34. In Oracle VirtualBox restore `srv1` from its **CDB** snapshot to undo the changes made in this practice.

Summary

- Configuring shared server processes in Oracle databases allows us to pre-allocate server processes to serve all the incoming client requests of specific attributes (for example connecting to specific database service). With this feature, we would have fewer server processes serving higher number of client sessions.

- With the parameter `SHARED_SERVERS` we set the initial number of server processes starting to serve the requests. With the parameter `MAX_SHARED_SERVERS` we can set the maximum number of processes the database may spawn.