9. Configuring the Shared Server

Abstract: As the number of users connecting to Oracle services in the enterprise grows, the system requirements of the servers increase—particularly the memory and process requirements. This lesson discusses the Oracle Shared Server option and its benefits. In addition, you will learn the steps of the connection process, how to configure the Shared Server option, and how user requests are processed in a shared server configuration. You will also learn how to manage the shared server environment using various Oracle data dictionary views and how to configure clients to request dedicated connections when you are connecting to an Oracle database configured with Shared Server. Finally, you will learn where to find information to assist you in tuning the Shared Server. It is important for the database administrator to understand the Oracle Shared Server architecture and how and when it is appropriate to utilize this feature. When used properly, this feature can help alleviate problems with servers that are being taxed in terms of the number of processes running on the server. When the Shared Server is configured properly, it can help stave off system upgrades that may have seemed imminent, thus saving the organization time and money.

Contents

Objectives:

- Identify the components of the Oracle Shared Server.
- Describe the Oracle Shared Server architecture.
- Configure the Oracle Shared Server.
- Identify and explain usefulness of related dictionary views.

1. Shared Server Configuration

Shared Server is an optional configuration of the Oracle server that allows the server to support a larger number of concurrent connections without increasing physical resource requirements. This is accomplished by sharing resources among groups of users.

1.1. Dedicated Server versus Shared Server

If you have ever gone to a very upscale restaurant, you may have had your own personal waiter. The waiter is there to greet you and escort you to your seat. The waiter will take your order for food and drinks and even help in the preparation of your order. No matter how many other patrons enter the restaurant, your waiter is responsible for serving only your requests. Therefore, your service is very consistent—if the person is a good waiter.

A *dedicated server* environment works in much the same way. Every client connection has an associated dedicated server process on the machine where the Oracle server exists. No matter how many other connections are made to the server, you always have the same dedicated server responsible for processing only your requests. You utilize the services of that server process until you disconnect from the Oracle server.

Most restaurants tend to be more like shared servers. When you walk in, you are assigned a waiter or waitress, but they may have many other tables they are responsible for serving. This is good for the restaurant because they can serve more customers without increasing the staff. It may be fine for you as well, if the restaurant is not too busy and the waiter or waitress is not responsible for too many tables. Also, if most of the orders are small, the staff can keep up with the requests and the service will be as good as if you had your own personal waiter.

In the diner, things work slightly differently; here, the waitress takes your order and places it on a turnstile. If the diner has multiple cooks, the order is picked up from the turnstile and prepared by one of the available cooks. When the cook completes the preparation of the dinner, it is placed in a location where the waitress can pick it up and bring it back to your table.

This is how a shared server environment works. In a shared server environment, clients share processes on the Oracle server. These shared processes are called *dispatchers*. Dispatchers are like the waiter or waitress in the diner. A dispatcher can be responsible for taking the orders of many clients. When you request something from the server, it is the dispatcher's responsibility to take your request and place it in a location called a *request queue*.

The request queue functions like the turnstile in the diner analogy. There is one request queue where all of the dispatcher processes place their client requests. The request queue is a structure contained in the System Global Area (SGA).

Shared server processes, like cooks in a diner, are responsible for fulfilling the client requests. The shared server process executes the request and places the result into an area of the SGA called a *response queue*. Every dispatcher has its own response queue. The dispatcher picks up the completed request from the response queue and returns the results back to the client. Figure 1 depicts the processing steps for a Shared Server request shown here:

1. The client passes a request to the dispatcher serving it.
2. The dispatcher places the request on a request queue in the SGA.
3. One of the shared server processes executes the request.
4. The shared server places the completed request on the dispatchers' response queue of the SGA.
5. The dispatcher picks up the completed request from the response queue.
6. The completed request is passed back to the client.

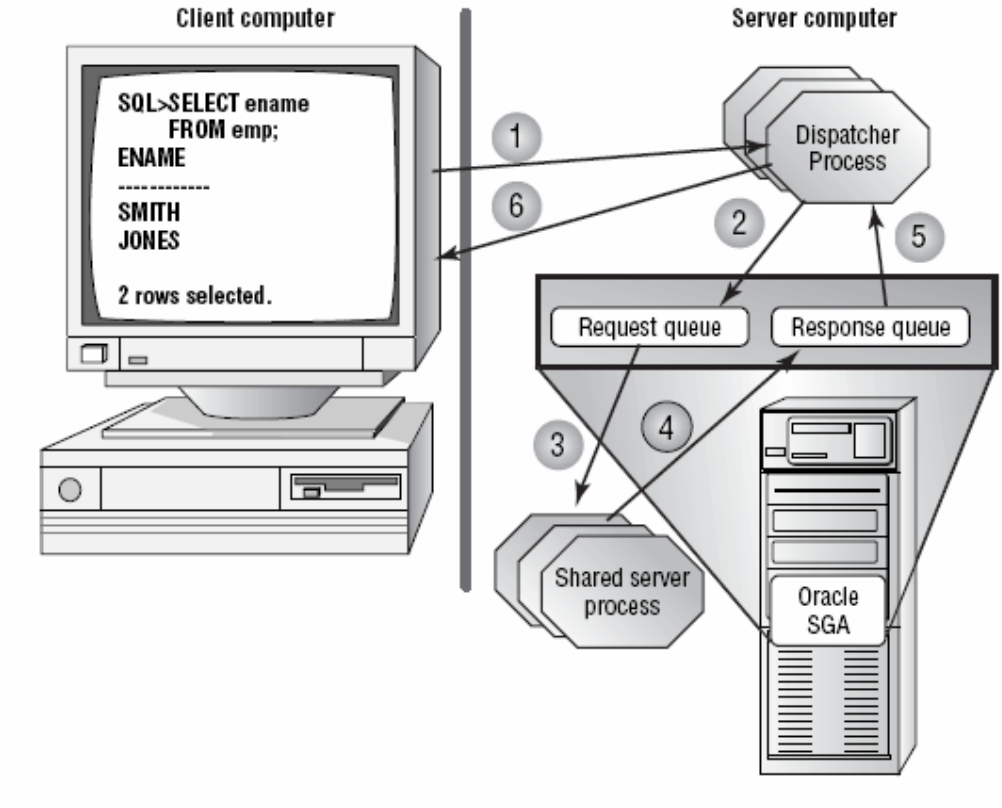

FIGURE 1. Request processing in Shared Server.

1.2. Advantages of Shared Server

A shared server is used in situations where server resources, such as memory and active processes, become constrained. People tend to throw more hardware at problems like these; this will likely remedy the problem, but it may be an unnecessary expense.

If your system is experiencing these problems, Shared Server allows you to support the same or greater number of connections without additional hardware requirements. As a result, Shared Server tends to decrease the overall memory and process requirements on the server.

An average dedicated connection takes roughly two to four megabytes of memory. A shared server connection takes about two megabytes of memory. As you can see, there will be some overall memory reduction. Also, because clients are sharing processes, the total number of processes is reduced. These translate into resource savings on the server.

Shared Server is also required to take advantage of certain network options, such as connection concentration when using Oracle Connection Manager.

1.3. Applications Suited to Shared Server

Shared Server is suitable for "high think" applications. High think applications are comprised of small transactions with natural pauses in the transaction patterns. These types of applications are good candidates for shared server connections. An example of a high think application would be an order entry system. Order entry systems tend to have small transactions with natural pauses in the work pattern of entering the information.

1.4. Drawbacks of Shared Server

Applications that generate a significant amount of network traffic or result in large result sets are not good candidates for shared server connections. Think of the diner analogy from the previous discussion. Your service is fine until two parties of 12 people show up. All of a sudden, the waitress is overwhelmed with work from these two other tables and your service begins to suffer. The same thing would happen in a shared server environment. If requests for large quantities of information start going to the dispatchers, the dispatchers can become overwhelmed by these large requests, and you may see performance suffer for the other clients connected to the dispatcher. This, in turn, will increase your response times. Dedicated processes better serve these types of applications.

There are some functions that are not allowed when you are using a shared server connection. You cannot start up, shut down, or perform certain kinds of recovery of an Oracle server when you are connected via a shared server.

1.5. Scalability versus Performance

Shared Server is a scalability enhancement option, not a performance enhancement option. If you are looking for an increase of performance, Shared Server is not what you should be configuring. Only use Shared Server if you are experiencing the system constraint problems discussed earlier. You will always have equal or better performance in a dedicated server environment.

1.6. Oracle Server Changes in a Shared Server Environment

When Shared Server is configured, Oracle adds two new types of structures to the SGA: request queues and response queues. These structures do not exist in a dedicated server environment. There is one request queue for all dispatchers but each dispatcher has its own response queue. So if you have four dispatchers, there would be one request queue and four response queues. The request queue is a location in the SGA where the dispatcher places client requests. A shared server process executes each request, and then it places the completed request in the dispatchers' response queue.

You have to configure the number of dispatcher and shared server processes that you want to start with when the instance starts; you also have to configure the maximum number of each of these structures. Later, you will see how to determine the starting number of dispatchers and shared server processes.

In a dedicated environment, each dedicated server has a memory segment called a *Program Global Area* (PGA). The PGA is an area of memory where information about each client session is maintained. This information includes bind variables, cursor information, and the client's sort area. In a shared server environment, this information is moved from the PGA to an area of the SGA called the *User Global Area* (UGA). You can configure a special area of the SGA called the *Large Pool* to accommodate the bulk of the UGA. In older releases of Oracle, the entire UGA was stored in the Shared Pool. As of Oracle8, the majority of the UGA can be stored in the Large Pool. You will see how to configure the Large Pool later. Figure 2 shows how the SGA and PGA structures differ between a dedicated and a shared server environment.

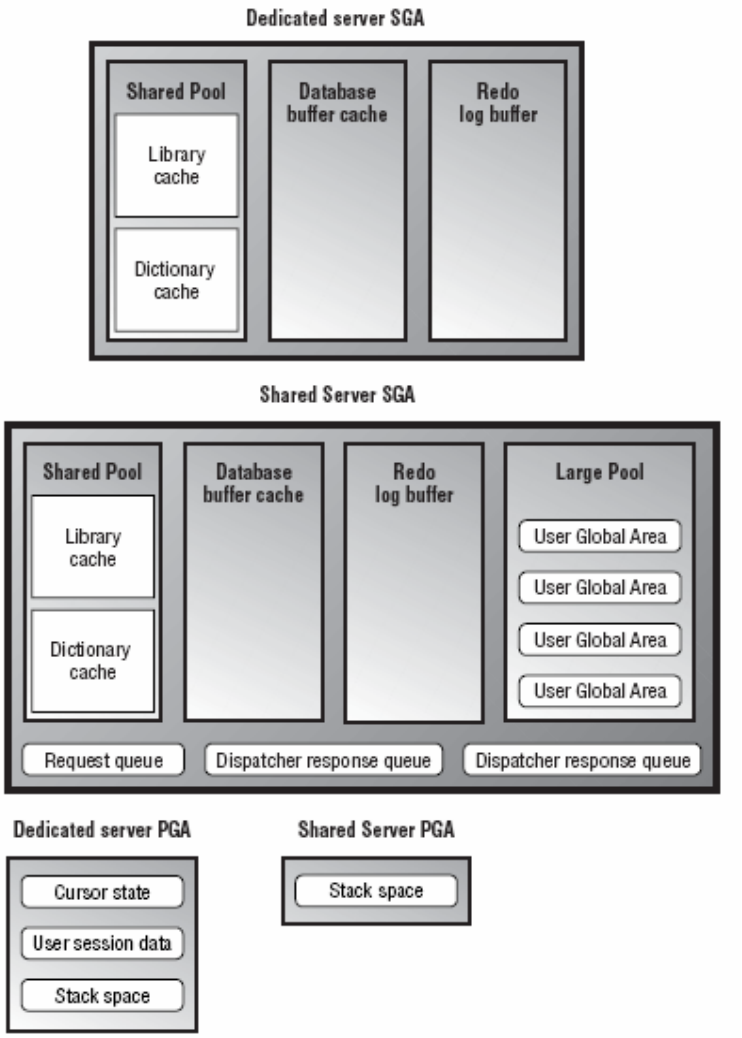

FIGURE 2. SGA dedicated server versus Shared Server.

Each connection being serviced by a dispatcher is bound to a shared memory segment and forms a *virtual circuit*. The shared memory segment is utilized by the dispatcher to manage

communications between the client and the Oracle server. The shared server processes use the virtual circuits to send and receive information to the appropriate dispatcher process.

1.7. The Role of the Listener in a Shared Server Environment

The listener plays an important role in the shared server environment. It is the listener that is responsible for supplying the client with the address of the dispatcher to connect to when a user requests connections to a shared server. The Oracle background process PMON notifies the listener as to which dispatcher is responsible for servicing this virtual circuit. The listener is then aware of the number of connections the dispatcher is managing. This information allows the listener to take advantage of dispatcher load balancing. Dispatcher load balancing was introduced in Oracle8i.

Load balancing allows the listener to make intelligent decisions about which dispatcher to redirect client connections to so that no one dispatcher becomes overburdened. When the listener receives a connection request, it looks at the current connection load for each dispatcher and redirects the client connection request to the least loaded dispatcher. By doing so, the listener ensures that connections are evenly distributed across dispatchers. When a client connection terminates, the listener is updated to reflect the change in the number of connections the dispatcher is handling. Figure 3 depicts the steps of the shared server connection process shown here:

1. The dispatcher processes are spawned when an instance is started.
2. The client contacts the Oracle server after resolving the service name.
3. The server redirects the client connection to the least busy dispatcher.
4. The dispatcher process manages the client server request.
5. PMON registers connection information with the listener.

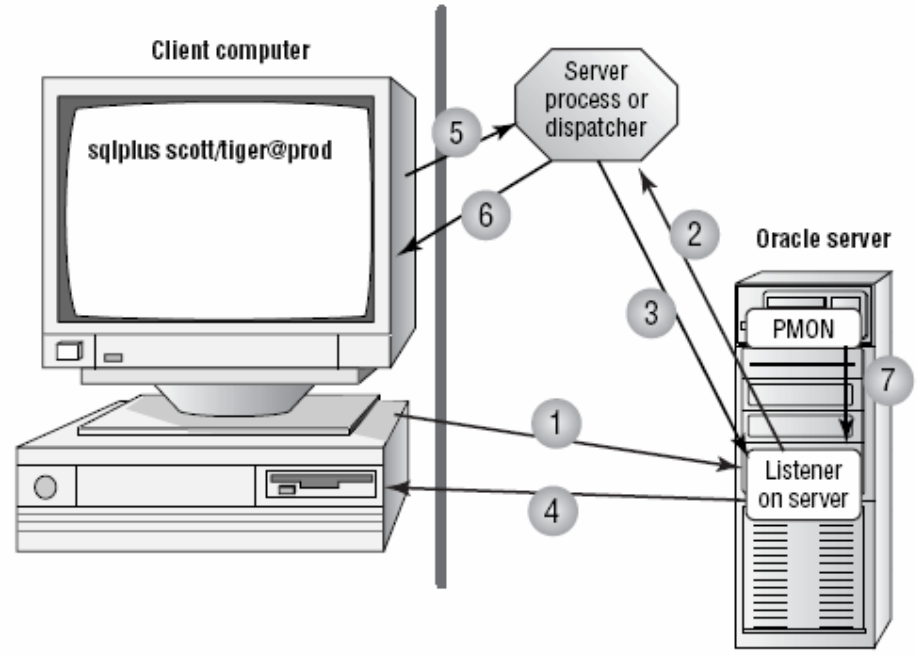

FIGURE 3. Shared server connection process.

2. Configuring the Shared Server Option

Shared Server requires additional parameters in the init.ora file. These parameters identify the number and type of dispatchers, the number of shared servers, and the name of the database that you want to associate with the Shared Server.

2.1. DISPATCHERS Parameter

This parameter configures the number of dispatchers that should start when the instance is started. The DISPATCHERS parameter specifies the number of dispatchers and the type of protocol that the dispatchers can respond to.

You can add additional dispatchers dynamically using the ALTER SYSTEM command. This command allows you to increase the number of dispatchers without bringing the instance down. In addition, this command has a number of optional attributes. The two main attributes are the number of dispatchers and the protocol the dispatcher will listen for. For example, say you wanted to configure three TCP/IP dispatchers and two IPC dispatchers. You would set the parameter as follows:

```
DISPATCHERS = "(PRO=TCP)(DIS=3)(PRO=IPC)(DIS=2)"
```

All of the attributes for this parameter can be abbreviated. Table 1 shows the other attributes you can set with the DISPATCHERS parameter. Of the three attributes, ADDRESS,

DESCRIPTION, or PROTOCOL, only one needs to be specified for a DISPATCHERS definition.

TABLE 1. Summary of DISPATCHER Attributes.

Attribute	**Abbreviations**	**Description**
ADDRESS	ADD or ADDR	Specifies the network protocol address of the endpoint on which the dispatchers listen.
CONNECTIONS	CON or CONN	The maximum number of network connections per dispatcher. The default value varies by operating system.
DESCRIPTION	DES or DESC	The network description of the endpoint where the dispatcher is listening, including the protocol being listened for.
DISPATCHERS	DIS or DISP	The number of dispatchers to start when the instance is started. The default is 1.
LISTENER	LIS or LIST	The address of the listener that PMON sends connection information to.
PROTOCOL	PRO or PROT	The network protocol for the dispatcher to listen for.
SESSIONS	SES or SESS	The maximum number of network sessions allowable for this dispatcher.

Determining the Number of Dispatchers to Start

The number of dispatchers you start will vary depending on your particular configuration. Your operating system may place a limit on the number of connections that one dispatcher can handle. Consult the operating system documentation to obtain this information.

Use the following formula as a guide when you are deciding how many dispatchers to initially configure:

```
Number of Dispatchers = CEIL (maximum number of concurrent sessions / connections per dispatcher)
```

For example, if you have 200 concurrent TCP/IP connections, and you want each dispatcher to manage 20 concurrent connections, you would need 10 dispatchers. You would set your DISPATCHERS parameter as follows:

```
DISPATCHERS = "(PRO=TCP)(DIS=10)"
```


You can determine the number of concurrent connections by querying the V$SESSION view. This view shows you the number of clients currently connected to the Oracle server. Here is an example of the query:

```
SQL> select sid,serial#,username,server,program from v$session
  2*  where sid > 6
SID    SERIAL#    USERNAME  SERVER      PROGRAM
------- ---------      ----------      ---------      ---------------
7       13             SCOTT       DEDICATED  SQLPLUS.EXE
8       12             SCOTT       DEDICATED  SQLPLUS.EXE
9       4              SYSTEM      DEDICATED  SQLPLUS.EXE
```

In this example, you see three users connected to the server, and their SIDs begin with the number 7. This is because in the SQL statement, you can ignore the first six sessions. These entries refer to the background processes such as PMON and SMON. If you took a sampling of this view over a typical work period, you would get an idea of the average number of concurrent connections for your system. You would then use this number as a guide when you established the starting number of dispatchers.

Managing the Number of Dispatchers

You can start additional dispatchers or remove dispatchers dynamically using the ALTER SYSTEM command. You can start any number of dispatchers up to the MAX_DISPATCHERS setting. Here is an example of adding three TCP/IP dispatchers to a system configured with two TCP/IP dispatchers:

```
ALTER SYSTEM SET DISPATCHERS="(PRO=TCP)(DIS=5)";
```

Notice that you set the number to the total number of dispatchers you want, not the number of dispatchers you want to add.

2.2. MAX_DISPATCHERS Parameter

Set this parameter to the maximum number of dispatchers you anticipate needing for the Oracle server. This number cannot be set dynamically. The maximum number of processes a dispatcher could run concurrently is operating system dependent. Use the following formula to set this parameter:

```
MAX_DISPATCHERS = (maximum number of concurrent sessions/ connections
per dispatcher)
```

Here is an example of the parameter:

```
MAX_DISPATCHERS = 5
```


2.3. SHARED_SERVERS Parameter

This parameter specifies the number of shared servers to start when the Oracle instance is started. This parameter can be altered dynamically with the ALTER SYSTEM command. You must set this parameter to at least 1 for Oracle to use shared server connections. A setting of 0 or no setting means shared servers will not be used. This parameter can be changed dynamically, so even if shared servers are not configured when the instance starts, they can be configured without bringing the Oracle instance down and restarting it.

The number of servers necessary will vary depending on the type of activities your users are performing. Generally, for the types of high think applications that will be using shared server connections, 15 to 20 connections per shared server should be adequate. If your users are going to require larger result sets or are doing more intensive processing, then you will want to reduce this ratio.

Here is an example of the parameter:

```
SERVERS = 3
```

2.4. SHARED_SERVER_SESSIONS Parameter

This parameter specifies the total number of shared server sessions that are allowable for the Oracle instance. This parameter is derived from the lesser of CIRCUITS or SESSIONS – 5. For example if CIRCUITS is set to 10 and SESSIONS is set to 20, then SHARED_SERVER_SESSIONS would be equal to 10 because this is the lesser of the 2 values. This is a derived value and would not be set in the init.ora file manually.

Managing the Number of Shared Servers

You can start additional shared servers or reduce the number of shared servers dynamically using the ALTER SYSTEM command. You can start any number of shared servers up to the MAX_SERVERS setting. Here is an example of adding three additional shared servers to a system initially configured with two shared servers:

```
ALTER SYSTEM SET SHARED_SERVERS = 5;
```

Notice that you set the number to the total number of shared servers you want, not the number of shared servers you want to add.

2.5. MAX_SERVERS Parameter

Set this parameter to the maximum number of shared servers you anticipate needing for the Oracle server. This number cannot be set dynamically. Generally, you should set this parameter to accommodate your heaviest work times.

Here is an example of the parameter:

```
MAX_SERVERS = 5
```

2.6. CIRCUITS Parameter

This parameter manages the total number of virtual circuits allowed for all incoming and outgoing network sessions. This is a static parameter and defaults to the value of the SESSIONS init.ora parameter. This parameter does influence the total size of the SGA at system startup.

Here is an example of the parameter:

```
CIRCUITS = 200
```

3. Starting the Instance and Managing Shared Server

After you have completed the configuration work in the init.ora file, you will need to restart your database. Once you have restarted your database, you should query the V$ views to see if the instance has started the shared servers and the dispatcher processes. Later in this lesson, we will discuss the various data dictionary views used to manage Shared Server.

3.1. Registering Dispatcher Information with the Listener

When Shared Server is started, PMON registers dispatcher information with the listener. You can query the listener to see this registered information. The listener will keep track of the current connection load across all of the dispatchers. This information is necessary so that the listener can take advantage of dispatcher load balancing.

REAL WORLD SCENARIO **Troubleshooting Shared Server Startup Problems**
If you have problems starting the instance, chances are that you have a problem with one of the Oracle Net files. If you receive an error that states the DISPATCHERS parameter is incorrect, make a backup of all of your Oracle Net files, such as tnsnames.ora and listener.ora, and re-create these files using the Oracle Net Manager. Most problems we have experienced with the Shared Server can be traced back to modifications made to these files without using Oracle Net Manager. Even cutting and pasting service information to create new service definitions can cause problems. After you have re-created your files, try to restart your instance.

3.2. Displaying Information about Shared Server Connections Using lsnrctl

You can use the lsnrctl command line utility to see information about the dispatcher processes. Use the lsnrctl services query to view information about dispatchers. The example below shows a listener listening for two TCP/IP dispatchers. Notice that the listing displays how many connections each dispatcher is managing, the listening location of the dispatcher, and the process ID of the dispatcher. This example has three active connections, two dispatcher D000s and one dispatcher D001.

```
D:\>lsnrctl services

LSNRCTL for 32-bit Windows: Version 9.0.1.1.1 - Production on 03-OCT-2001
20:50:35

Copyright (c) 1991, 2001, Oracle Corporation. All rights reserved.

Connecting to

(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=mprntw507953)

(PORT=1521)))

Services Summary...


Service "MJW" has 2 instance(s).

        Instance "MJW", status UNKNOWN, has 1 handler(s) for this service...

        Handler(s):

                "DEDICATED" established:0 refused:0

                LOCAL SERVER

        Instance "MJW", status READY, has 3 handler(s) for this service...

        Handler(s):

                "DEDICATED" established:0 refused:0 state:ready

                LOCAL SERVER

        "D001" established:11 refused:1 current:1 max:1002

        state:ready

        DISPATCHER <machine: MPRNTW507953, pid: 352>

        (ADDRESS=(PROTOCOL=tcp)(HOST=mprntw507953.cmg.com)

        (PORT=1038))

        "D000" established:15 refused:3 current:2 max:1002

        state:ready
```



```
DISPATCHER <machine: MPRNTW507953, pid: 117>
(ADDRESS=(PROTOCOL=tcp)(HOST=mprntw507953.cmg.com)
(PORT=1036))
The command completed successfully
```

3.3. Data Dictionary Views for Shared Server

The data dictionary provides views you can query to gather information about the Shared Server environment. These views provide information about the number of dispatchers and shared servers configured, the activity among the shared servers and dispatchers, the activity in the request and response queue, as well as the clients that are connected with shared server connections. The data dictionary views are described in the following sections..

V$DISPATCHER Dictionary View

The V$DISPATCHER view contains information about the dispatchers. You can collect information about the dispatchers' activity, the number of connections the dispatchers are currently handling, and the total number of connections each dispatcher has handled since instance startup. Here is a sample output from the V$DISPATCHER view:

```
SQL> select name,status,messages,idle,busy,bytes,breaks from
  2  v$dispatcher
NAME  STATUS   MESSAGES   IDLE    BUSY  BYTES   BREAKS
----  ------   ---------  -------  ----  ------  ------
D000  WAIT     168        389645 108     12435   0
D001  WAIT     94         389668 48      6940    0
```

V$DISPATCHER_RATE Dictionary View

The V$DISPATCHER_RATE view shows statistics for the dispatchers, such as the average number of bytes processed, the maximum number of inbound and outbound connections, and the average rate of bytes processed per client connection. The columns in the table that begin with CUR show current statistics. Columns that begin with AVG or MAX show historical statistics taken at some time interval. The time interval is typically measured in hundredths of a second.

The scale measurement periods used for each of the column types is contained in the columns that begin with SCALE. This information can be useful when you are taking load measurements for the dispatchers. Here is a sample of the output from this view.

```
SQL>select name,cur_event_rate,cur_msg_rate, cur_svr_byte_rate from
v$dispatcher_rate
```



```
NAME  CUR_EVENT_RATE  CUR_MSG_RATE  CUR_SVR_BYTE_RATE
----  --------------  ------------  -----------------
D000  12              0             0
D001  14              0             1
```

V$QUEUE Dictionary View

The V$QUEUE dictionary view contains information about the request and response queues. The information deals with how long requests are waiting in the queues. This information is valuable when you are trying to determine if more shared servers are needed. The following example shows the COMMON request queue and two response queues:

```
SQL> select * from v$queue;
PADDR     TYPE        QUEUED   WAIT     TOTALQ
--------  ----------  -------  -------  -------
00        COMMON      0        0        152
03C6C244  DISPATCHER  0        0        91
03C6C534  DISPATCHER  0        0        71
```

V$CIRCUIT Dictionary View

V$CIRCUIT displays information about Shared Server virtual circuits, such as the volume of information that has passed between the client and the dispatcher and the current status of the client connection. The SADDR column displays the session address for the connected session. This can be joined to the V$SESSION view to display information about the user to whom this connection belongs. Here is a sample output from this view:

```
SQL> select circuit,dispatcher,server,waiter WTR,
  2  status,queue,bytes from v$circuit;
CIRCUIT   DISPATCH  SERVER    WTR  STATUS  QUEUE   BYTES SADDR
--------  --------  --------  ---  ------  ------  ----- ------
03E2A624  03C6C244  00        00   NORMAL  NONE    47330 3C7AB68
03E2A724  03C6C534  03C6BC64  00   NORMAL  SERVER  43572 03C79BE8
```

V$SHARED_SERVER Dictionary View

This view contains information about the shared server processes. It displays information about the number of requests and the amount of information processed by the shared servers. It also indicates the status of the shared server (i.e., whether it is active or idle).

```
SQL> select name,status,messages,bytes,idle,busy, requests from v$shared_server;

NAME  STATUS  MESSAGES  BYTES  IDLE  BUSY  REQUESTS
----  ------  --------  -----  -----  ----  --------
S000  EXEC    372       86939  98472  300   175
S001  EXEC    26        9851   98703  38    13
```

V$SHARED_SERVER_Monitor Dictionary View

This view contains information that can assist in tuning the Shared Server. This includes the maximum number of concurrent connections attained since instance startup and the total number of servers started since instance startup. The query below shows an example of output from the V$SHARED_SERVER view.

```
SQL> select maximum_connections "MAX CONN",maximum_sessions "MAX SESS", servers_started "STARTED" from v$shared_server_monitor;

MAX CONN  MAX SESS  STARTED
--------  --------  --------
115       120       10
```

V$SESSION Dictionary View

This view contains information about the client session. The SERVER column indicates whether this client is using a dedicated session or a dispatcher. The listing below shows an example of the V$SESSION view displaying the server information. This listing ignores any rows that do not have a username to avoid listing information about the background processes. Notice that user Scott has a server value of SHARED. This means Scott is connected to a dispatcher. The SYSTEM user is connected using a local connection because the status is NONE. If a user connected using a dedicated connection, the status would be DEDICATED.

```
SQL> select username,program,server from v$session where username is not null;

USERNAME  PROGRAM      SERVER
--------  -----------  ---------
SYSTEM    SQLPLUS.EXE  NONE
SCOTT     SQLPLUS.EXE  SHARED
```

V$MTS Dictionary View

This view contains information about the configuration of the dispatchers and shared servers. This includes the maximum number of connections for each dispatcher, the number of shared servers that have been started and stopped, and the highest number of shared servers that have been active at the same time. This view gives you an indication of whether more shared server processes should be started. The sample below shows output from this view:

```
SQL> select MAX_CONNECTIONS MAX_CONN, SERVERS_STARTED
SRV_STARTED, SERVERS_TERMINATED SRV_TERM,
SERVERS_HIGHWATER SRV_HW
FROM V$MTS;

MAX_CONN SRV_STARTED   SRV_TERM      SRV_HW
---------- ----------  ----------    ------
60         0           0             2
```

NOTE: The V$MTS view is identical in content to the V$SHARED_SERVER_MONITOR view. V$MTS was the name for this view at Oracle8i release and is still available for reference.

4. Requesting a Dedicated Connection in a Shared Server Environment

You can have Shared Server and dedicated servers connecting to a single Oracle server. This is advantageous in situations where you have a mix of activity on the Oracle server. Some users may be well suited to shared server connections while other types of users may be better suited to use dedicated connections.

By default, if Shared Server is configured, a client is connected to a dispatcher unless the client explicitly requests a dedicated connection. As part of the connection descriptor, the client has to send information requesting a dedicated connection. Configure this option using the Oracle Net Manager.

Clients may request this type of connection if the names resolution method is localnaming. This option cannot be used with hostnaming.

4.1. Configuring Dedicated Connections When Localnaming Is Used

If you are using localnaming, you want to add a parameter to the service name entry in the tnsnames.ora file. The parameter (SERVER=DEDICATED) is added to the DBA net service

name. The SERVER parameter can also be abbreviated as SRVR. Here is an example of the entry in the tnsnames.ora file.

```
# D:\ORACLE\ORA90\NETWORK\ADMIN\TNSNAMES.ORA Configuration
# File:D:\Oracle\Ora90\NETWORK\ADMIN\tnsnames.ora
# Generated by Oracle Net Manager
DBA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = weishan)
        (PORT = 1521))
  )
  (CONNECT_DATA =
    (SERVICE_NAME = DBA)
      (SRVR = DEDICATED) Request a dedicated connection for DBA
  )
)
```

5. Tuning the Shared Server Option

Before tuning the Shared Server, you should examine the performance of the dispatchers and the shared server processes. You want to make sure that you have enough dispatchers so that clients are not waiting for dispatchers to respond to their requests, and you want to have enough shared server processes so that requests are not waiting to be processed. You also want to configure the Large Pool SGA memory area. The Large Pool is used to store the UGA. The UGA takes the place of the PGA that is used for dedicated servers.

The Large Pool is designed to allow the database to request large amounts of memory from a separate area of the SGA. Before the database had a Large Pool design, memory allocations for Shared Server came from the Shared Pool.

This caused Shared Server to compete with other processes updating information in the Shared Pool. The Large Pool alleviates the memory burden on the Shared Pool and enhances performance of the Shared Pool.

5.1. Configure the Large Pool

You can configure the Large Pool by setting the parameter LARGE_POOL_SIZE in the init.ora file. This parameter can be set to a minimum of 300KB and a maximum of at least 2GB; the maximum setting is operating system dependent.

When a default value is used, Oracle adds 250KB per session for each shared server if the DISPATCHERS parameter is specified. If you do not configure a Large Pool, Oracle will place the UGA into the Shared Pool. Because of this, you should configure a Large Pool when using Shared Server so that you don't affect the performance of the Shared Pool. Here is an example of setting the LARGE_POOL_SIZE parameter in the init.ora file:

```
LARGE_POOL_SIZE = 50M
```

You can see how much space is being used by the Large Pool by querying the V$SGASTAT view. The free memory row shows the amount available in the Large Pool and the session heap row shows the amount of space used in the Large Pool. Here is a listing that shows an example of the query:

```
SQL> select * from v$sgastat where pool = 'Large Pool';

POOL         NAME                     BYTES
-----------  ------------------------ ---------
large pool   free memory              251640
large pool   session heap              48360
```

5.2. Sizing the Large Pool

The Large Pool should be large enough to hold information for all of your shared server connections. Generally, each connection will need between one and three megabytes of memory, but this depends on that client's type of activity. Clients that do a great deal of sorting or open many cursors will use more memory.

You can gauge how much memory shared server connections are using by querying the V$SESSTAT view. This view contains information about memory utilization per user. The query below shows how to measure the maximum amount of memory for all shared server sessions since the instance was started. You can use this as a guide to determine how much memory you should allocate for the Large Pool. This example shows that the maximum amount of memory used for all shared server sessions is around 240KB:

```
select sum(value) "Max MTS Memory Allocated"from v$sesstat

ss, v$statname st

where name = 'session uga memory max'and ss.statistic#

=st.statistic#;

Max MTS Memory Allocated
```

```
------------------------
                244416
```

5.3. Determine Whether You Have Enough Dispatchers

The dispatcher processes can be monitored by querying the V$DISPATCHER view. This view contains information about how busy the dispatcher processes are. Query this view to determine whether it will be advantageous to start more dispatchers.

The sample query below runs against the V$DISPATCHER view to show what percentage of the time dispatchers are busy:

```
Select name, (busy / (busy + idle))*100
"Dispatcher % busy Rate"
From V$DISPATCHER
Protocol        Dispatcher % Busy Rate
------------ --------------------------
D000            .00070079
D001            .0059
```

These dispatchers show very little busy time. If dispatchers are busy more than 50 percent of the time, you should consider starting more dispatchers. This can be done dynamically with the ALTER SYSTEM command. Add one or two more dispatchers and monitor the busy rates of the dispatchers to see if they fall below 50 percent.

5.4. Determine How Long Users Are Waiting for Dispatchers

To measure how long users are waiting for the dispatchers to execute their request, look at the combined V$QUEUE and V$DISPATCHER views. See the listing below for an example:

```
SELECT decode(sum(totalq),0,'No Responses',
Sum(wait)/sum(totalq)) "Average Wait time"
FROM V$QUEUE q, V$DISPATCHER d
WHERE q.type = 'DISPATCHER'
AND q.paddr = d.paddr;
Average Wait Time
------------------
.0413
```

The average wait time for dispatchers is a little more than four hundredths

of a second. Monitor this measure over time. If the number is consistently

increasing, you should consider adding more dispatchers.

5.5. Determine Whether You Have Enough Shared Servers

You can monitor shared servers by using the V$SHARED_SERVER and V$QUEUE dictionary views. The shared servers are responsible for executing client requests and placing the requests in the appropriate dispatcher response queue.

The measurement you are most interested in is how long client requests are waiting in the request queue. The longer the request remains in the queue, the longer the client will wait for a response. The following statement will tell you how long requests are waiting in the queue:

```
Select decode(totalq,0,'No Requests') "Wait Time",
Wait/totalq || ' hundredths of seconds'
"Average Wait time per request"
from V$QUEUE
where type = 'COMMON'
Wait Time Average Wait time per request
-------- ---------------------------------
.023132      hundredths of a second
```

The average wait time in the request queue is a little more than two hundredths of a second. Monitor this measure over time. If the number is consistently increasing, you should consider adding more shared servers.

6. Summary

The Shared Server is a configuration of the Oracle server that allows you to support a greater number of connections without the need for additional resources. It is important to understand the shared server option because it can stave off potentially unnecessary hardware upgrades when you are faced with the problem of the number of processes your server can manage.

In this configuration, user connections share processes called dispatchers. Dispatchers replace the dedicated server processes in a dedicated server environment. The Oracle server is also configured with shared server processes that can process the requests of many clients.

The Oracle server is configured with a single request queue in which dispatchers place the client requests that the shared servers will take and process. The shared server processes put the completed requests in the appropriate dispatcher's response queue. The dispatcher then sends the completed request back to the client. These request and response queues are structures added to the SGA.

There are a number of parameters that are added to the init.ora file to configure Shared Server. Dispatchers and shared servers can be added dynamically after the Oracle server has been started. You can add more shared servers and dispatchers up to the maximum value specified.

There are several V$ views that are used to monitor Shared Server. The information contained in these views pertains to dispatchers, shared server processes, and the clients that are connected to the dispatcher processes. You can use the V$ views to tune the Shared Server. It is most important to measure how long clients are waiting for dispatchers to process their requests and how long it is taking before a shared server processes the client requests. These factors may lead to increasing the number of shared server and dispatcher processes. You also want to monitor the usage of the Large Pool.

References

[1] Oracle9i DBA Fundamentals II.

