Oracle Global Data Services with Active Data Guard

Hands-On Lab

Introduction

Oracle Global Data Services (GDS) is an automated workload management feature of Oracle Database 12*c* that provides workload routing, load balancing and inter-database service failover, replication lag based routing, role based global services and centralized workload management for a set of replicated databases that are globally distributed or located within the same data center.

GDS technology provides the following salient capabilities:

Region-based Workload Routing: With GDS, customers can choose to configure client connections to be always routed among a set of replicated databases in a local region. This capability allows customers to maximize their application performance (avoiding the network latency overhead accessing databases in remote regions).

Replication Lag-based Workload Routing: Replicas may lag behind the master/primary database. A global Service allows customers to choose the lag tolerance that is acceptable for a given application. GDS routes requests to replicas whose replication lag is below the limit. If the replication lag exceeds the lag limit, the service is relocated to another available database that lags below the threshold. New requests are routed to a database that satisfies the lag limit. If there is no available database, then the global service is shutdown. Once the lag is resolved or comes within the limit, GDS automatically brings up the service.

Connect-Time Load Balancing: Global Service Managers (GSM) use the load statistics from all databases in the GDS pool, inter-region network latency, and the configured connect-time load balancing goal to route the incoming connections to the best database in a GDS pool.

Run-Time Load Balancing: GDS also enables run-time load balancing across replicated databases by publishing a real-time load balancing advisory for connection pool based clients (for example – OCI, JDBC, ODP.NET, WebLogic etc.,). The connection pool based clients subscribe to this load balancing advisory and route database requests in real-time across already established connections. With the run-time connection load balancing feature of GDS, application client work requests are dynamically routed to the database that offers the best performance. In addition, GDS also supports the ability to dynamically re-distribute connections when the database performance changes.

Inter-database Service Failover: If a database running a global service crashes, GDS, taking into account the service placement attributes, automatically performs an inter-database service failover to another available database in the pool. GDS sends Fast Application Notification (FAN) events so that the client connection pools can reconnect to the new database where the global service has been newly started.

Role-based Global Services: Upon a database role transition via Data Guard broker, GDS can automatically relocate the global service to the new Primary and the new Standby if the role assigned to the service matches the role of the database.

Centralized Workload Management for Replicas: GDS allows easier configuration and management of the resources of the replicated databases that are located anywhere with a single unified framework.

For the details on Oracle Global Data Services, refer to the white paper on Global Data Services and the GDS Presentation shown in the Resources section of this document. This lab covers the GDS setup for an Active Data Guard environment. There will be a separate hands-on lab for GDS with Oracle GoldenGate.

GDS Components and Architecture

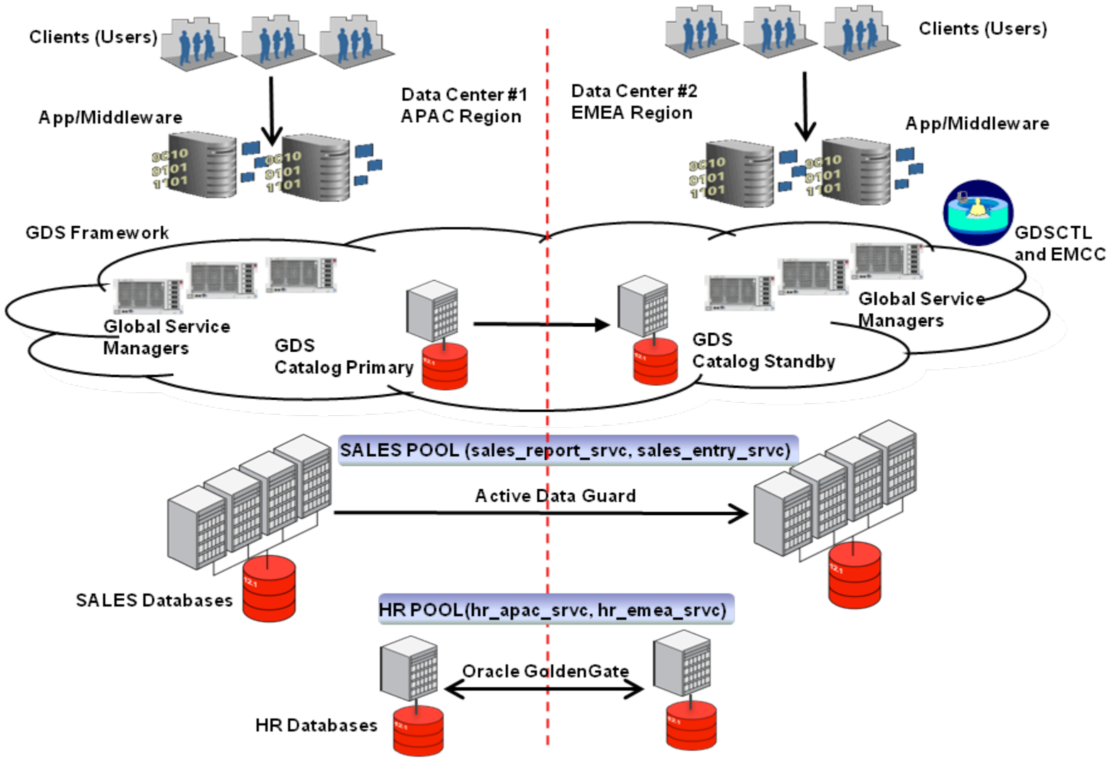


Figure 1: GDS Architecture

Figure 1 illustrates a typical GDS deployment which has two data centers (APAC, EMEA) and two sets of replicated databases (SALES, HR).

Global Service

Distributed workload management for replicated databases relies on the use of global services. Global services hide the complexity of a set of replicated databases by providing a single system image to manage the workload. The set of databases may include clustered or non-clustered Oracle databases (hosted on homogeneous or heterogeneous server platforms) which are synchronized with some form of replication technologies such as Oracle Data Guard, Oracle GoldenGate etc. A client request for a global service can be forwarded to any of the set of replicated databases.

GDS Configuration

A GDS configuration is a set of databases integrated by the GDS framework into a single virtual server offering global services. The set of databases in a GDS configuration can be located in any data center. Within a GDS configuration, there can be various sets of replicated databases belonging to different domains that do not share anything besides GDS framework components that manage them.
Clients connect by specifying a global service name and need not know the architectural topology of the GDS configuration.

GDS Pool

A set of replicated databases within a GDS configuration providing a unique set of global services and that belong to a certain administrative domain is termed as a GDS pool. Examples of a GDS pool can be a SALES pool or HR pool etc.

GDS Region
GDS region typically corresponds to a data center. Multiple data centers and clients that are in close geographic proximity to the data center can be mapped to be in the same GDS region as well. Examples of GDS regions are APAC region or EMEA region etc.

Global Service Manger (GSM)
GSM is the “brain” of the GDS technology and is the central component of the GDS configuration. At a minimum, there must be one GSM per GDS region. GSMs can be installed on any commodity server similar to the ones used for the application tier. The recommended approach is to deploy three GSMs per region.
Global Service Manager provides the following set of functions:

- Acts as a regional listener that the clients use to connect to global services. Any GSM can forward a connection request to any database (in any GDS region) that provides a given global service.
- Manages GDS configuration by making changes to the configuration data in all GDS components
- Gathers connection load balancing and performance metrics from all instances of the databases in the GDS configuration.
- Measures network latency between its own region and all other GDS regions and relays this information to all GSMs in other regions.
- Performs connection load balancing (CLB), routing client connections to the best database instance servicing a given global service in the GDS configuration – based on CLB metrics , network latency and the region affinity of the global service
- Generates FAN runtime load balancing (RLB) advisory and publishes it to client connection pools – based on performance metrics and integrating them with estimated network latency
- Manages failover of global services
- Monitors availability of database instances and global services notifies clients via FAN HA events upon failure incidents

GDS Catalog
The GDS catalog is a repository that keeps track of the configuration data and the run-time status of a given GDS configuration. Basically it contains information pertinent to global services, their attributes, GDS pools, regions, GSMs and database instances that are in the GDS configuration.

GDSCTL
GDS configurations can be administered either by the GDSCTL command-line user interface or Oracle Enterprise Manager Cloud Control 12c graphical user interface. Administrators who have used SRVCTL will be quite familiar with the look and feel of GDSCTL. GDS is also supported by EMCC DB plug-in starting from release 12.1.0.5.

GDS ONS Network
Each GSM process contains an Oracle Notification Server (ONS). A GSM employs the ONS to publish FAN HA events and RLB metrics subscribed by the clients.
Note: In Oracle Database 12.1.0.1, GDS supports Single Instance databases and Policy Managed RAC databases. The Data Guard configurations must be broker enabled.

Client Connectivity in GDS
In a GDS environment, the clients connect to the GSM listeners instead of the database listeners. GSM forwards the connection to the local listener (bypassing the SCAN listeners).
The TNS entry will include the specification of the endpoints of the GSM listeners. Client will load balance among local GSMs and use the remote GSMs if all the local GSMs are unavailable. It is possible for a client to

connect to a GSM in another region. Therefore, it is not possible to infer the clients region by the GSM it connects to. Clients specify global service name and which region they originate from.

```
sales_reporting_srvc =
  (DESCRIPTION =
   (FAILOVER=ON)
    (ADDRESS_LIST =                                     ← APAC's GSMs
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = gsm-host1a)(PORT = 1571))
     (ADDRESS = (PROTOCOL = TCP)(HOST = gsm-host2a)(PORT = 1571))
     (ADDRESS = (PROTOCOL = TCP)(HOST = gms-host3a)(PORT = 1571))

    )
    (ADDRESS_LIST =                                     ← EMEA's GSMs
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = gsm-host1e)(PORT = 1572))
     (ADDRESS = (PROTOCOL = TCP)(HOST = gsm-host2e)(PORT = 1572))
     (ADDRESS = (PROTOCOL = TCP)(HOST = gsm-host3e)(PORT = 1572))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = sales_reporting_srvc.sales.oradbcloud) (REGION=WEST)
    )
  )
```

Figure 2: Example of TNS connect descriptor

Figure 2 shows an example connect descriptor mapped to a TNS alias using GSM listeners in the tnsnames.ora file.

Overview of the Exercises

The objectives of the hands-on GDS session are to:

- Install Global Service Managers
- Deploy GDS framework for your data cloud
- Configure global services for the following GDS features
  - Inter-database global service failover
  - Role based global services
  - Lag based routing
  - Locality based workload routing
  - Connect-time load balancing
  - Run-time load balancing
- Obtain familiarity with the GDSCTL interface.

The exercises are designed from the top down so that each step builds upon the previous exercise. **It is essential that you follow the exercises in order**.

Prerequisites

1. All databases must be in Oracle Database Enterprise Edition 12.1.0.1
2. For Single Instance (Non-RAC) databases, Oracle Clusterware must not be installed on the database nodes.
   a. This restriction has been removed in 12.1.0.2

3. An application database (e.g. SALES) must already exist in a broker-enabled Active Data Guard configuration where the current Primary is SFO and Standby is CHI. The Data Guard broker is fully functional for role transitions. The user SYSDG has to be unlocked and have a password of 'oracle'
4. A separate catalog database (GDSCAT) must be created. GDS catalog schema will be loaded in this database.

GDS Test Bed Layout

This lab walks us through the creation of GDS configuration that offers the global services failover/load balancing capabilities. We will create a GDS configuration that contains the following components:

- GDS catalog hosted in "gdscat" database
    - Note: Best practice is to deploy HA/DR for gdscat.
- Two GDS Regions : West region and East region
- Two Global Services Managers (GSMs): one GSM per region (gsmwest, gsmeast)
    - Note: Best practice is to have 3 GSMs per region (For scalability & Availability)
- One GDS pool : Sales pool
- Two GDS pool databases in an Active Data Guard configuration, with one database in each of the datacenters. The "SFO" database is in the West region and the "CHI" database is in the East region.

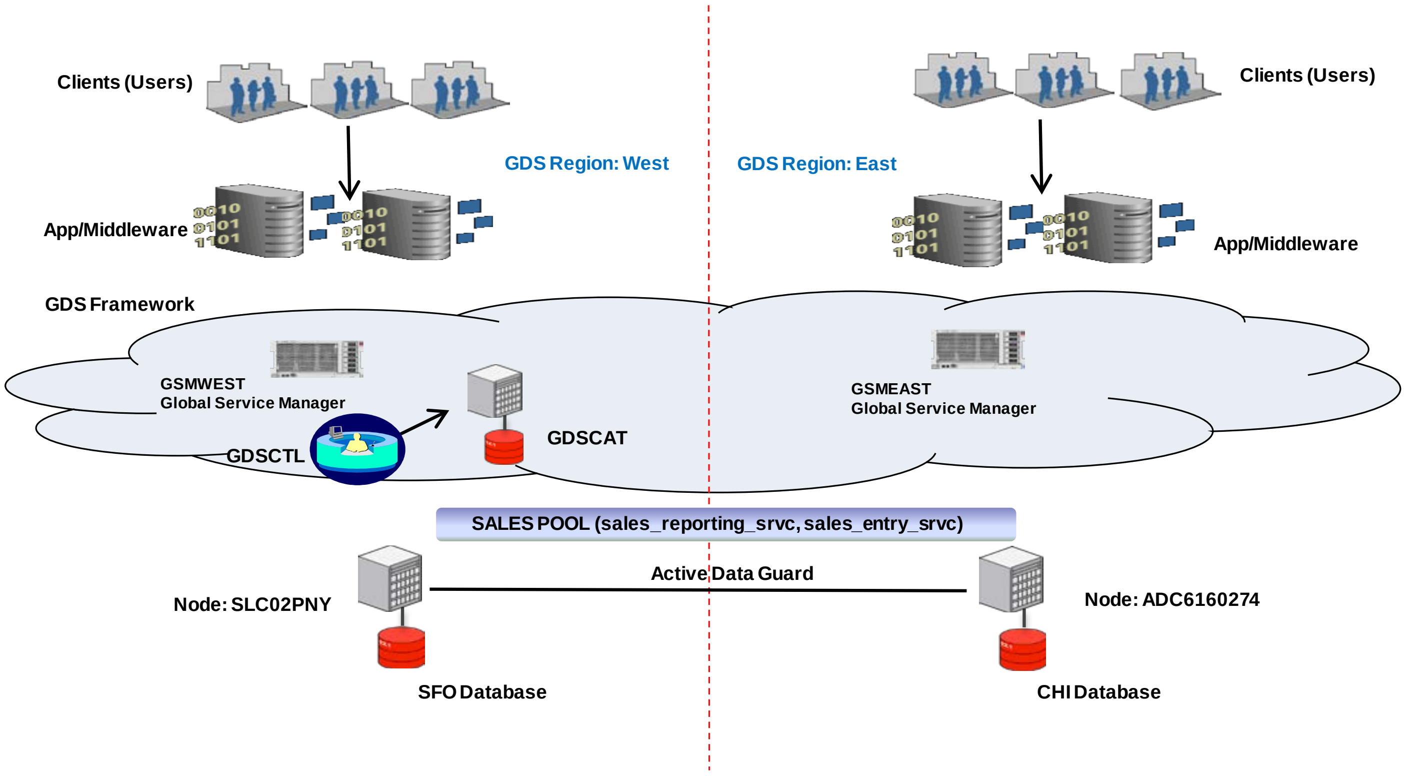


Figure 1: GDS Hands-On Lab Test Bed Layout

Note: In this lab, GSMWEST Global Service Manager, GDSCAT and SFO Database are hosted on SLC02PNY server. GSMEAST Global Service Manager and CHI Database are hosted on ADC6160274 server.

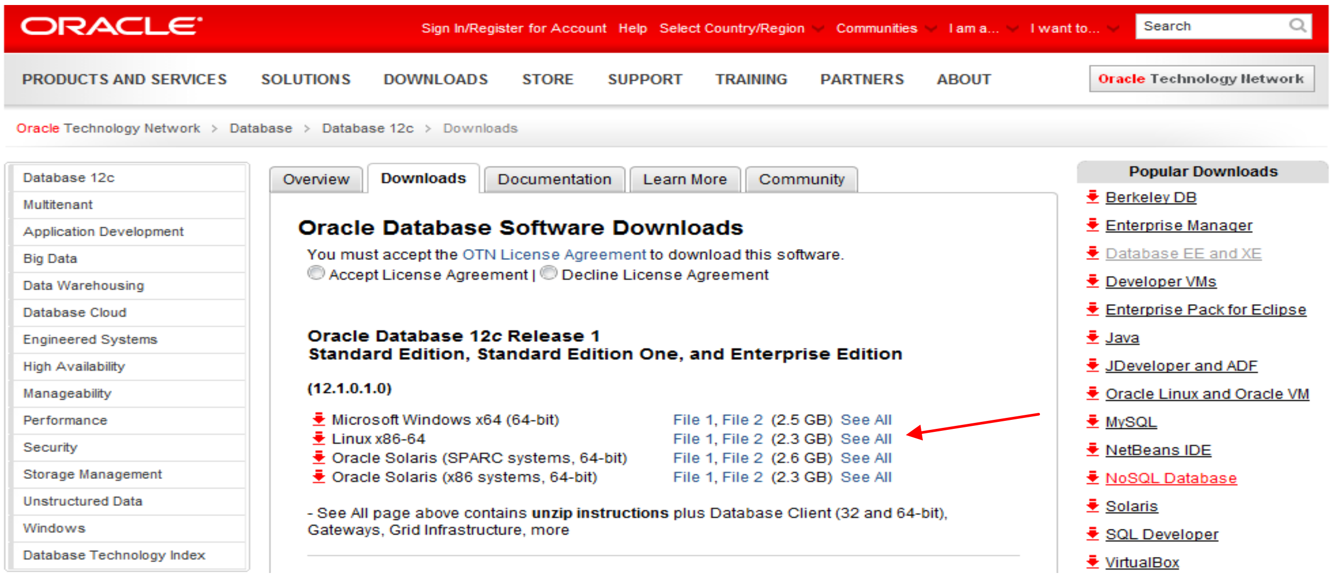
The high-level steps of GDS deployment are:

1. Install GSM software on GSM Nodes
2. Setup and Configure GDS
    a. Setup GDS Administrator accounts & privileges
    b. Configure GDS (Create GDS catalog, Add GSMs, regions, pools, brokerconfig, global services)
3. Setup client connectivity

Let’s begin the lab.

Installation of Global Service Managers

Locate the GSM media for your platform from
http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html


ORACLE
Sign In/Register for Account Help Select Country/Region Communities I am a... I want to... Search
PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT TRAINING PARTNERS ABOUT Oracle Technology Network
Oracle Technology Network > Database > Database 12c > Downloads
Database 12c
Multitenant
Application Development
Big Data
Data Warehousing
Database Cloud
Engineered Systems
High Availability
Manageability
Performance
Security
Storage Management
Unstructured Data
Windows
Database Technology Index
Overview Downloads Documentation Learn More Community
Popular Downloads
Berkeley DB
Enterprise Manager
Database EE and XE
Developer VMs
Enterprise Pack for Eclipse
Java
JDeveloper and ADF
Oracle Linux and Oracle VM
MySQL
NetBeans IDE
NoSQL Database
Solaris
SQL Developer
VirtualBox

Oracle Database Software Downloads

You must accept the OTN License Agreement to download this software.
Accept License Agreement | Decline License Agreement

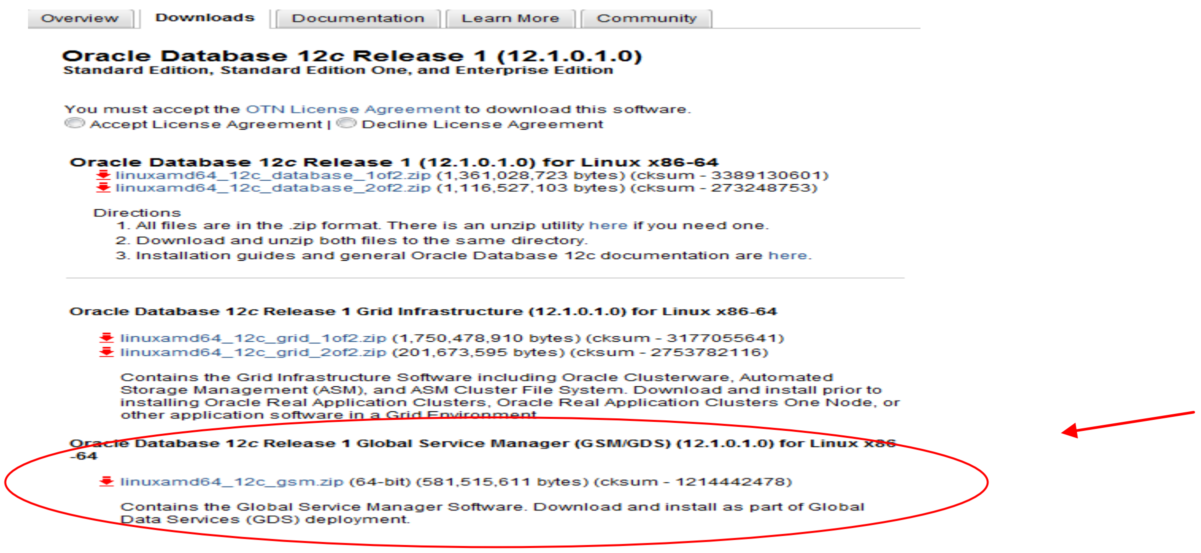
Oracle Database 12c Release 1
Standard Edition, Standard Edition One, and Enterprise Edition

(12.1.0.1.0)

Microsoft Windows x64 (64-bit)	File 1, File 2 (2.5 GB) See All
Linux x86-64	File 1, File 2 (2.3 GB) See All
Oracle Solaris (SPARC systems, 64-bit)	File 1, File 2 (2.6 GB) See All
Oracle Solaris (x86 systems, 64-bit)	File 1, File 2 (2.3 GB) See All

- See All page above contains **unzip instructions** plus Database Client (32 and 64-bit), Gateways, Grid Infrastructure, more

Note: This HOL uses the GSM software for the Linux x86-64



Oracle Database 12c Release 1 (12.1.0.1.0)
Standard Edition, Standard Edition One, and Enterprise Edition

You must accept the OTN License Agreement to download this software.
Accept License Agreement | Decline License Agreement

Oracle Database 12c Release 1 (12.1.0.1.0) for Linux x86-64
linuxamd64_12c_database_1of2.zip (1,361,028,723 bytes) (cksum - 3389130601)
linuxamd64_12c_database_2of2.zip (1,116,527,103 bytes) (cksum - 273248753)

Directions
1. All files are in the .zip format. There is an unzip utility here if you need one.
2. Download and unzip both files to the same directory.
3. Installation guides and general Oracle Database 12c documentation are here.

Oracle Database 12c Release 1 Grid Infrastructure (12.1.0.1.0) for Linux x86-64

linuxamd64_12c_grid_1of2.zip (1,750,478,910 bytes) (cksum - 3177055641)
linuxamd64_12c_grid_2of2.zip (201,673,595 bytes) (cksum - 2753782116)

Contains the Grid Infrastructure Software including Oracle Clusterware, Automated Storage Management (ASM), and ASM Cluster File System. Download and install prior to installing Oracle Real Application Clusters, Oracle Real Application Clusters One Node, or other application software in a Grid Environment.

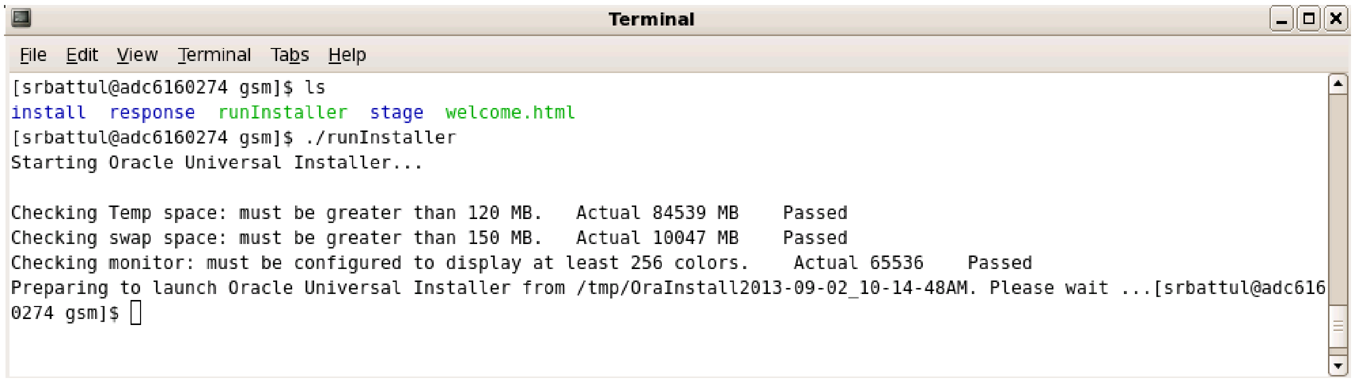
Oracle Database 12c Release 1 Global Service Manager (GSM/GDS) (12.1.0.1.0) for Linux x86-64

linuxamd64_12c_gsm.zip (64-bit) (581,515,611 bytes) (cksum - 1214442478)

Contains the Global Service Manager Software. Download and install as part of Global Data Services (GDS) deployment.

Download the GSM/GDS media and extract the zip file

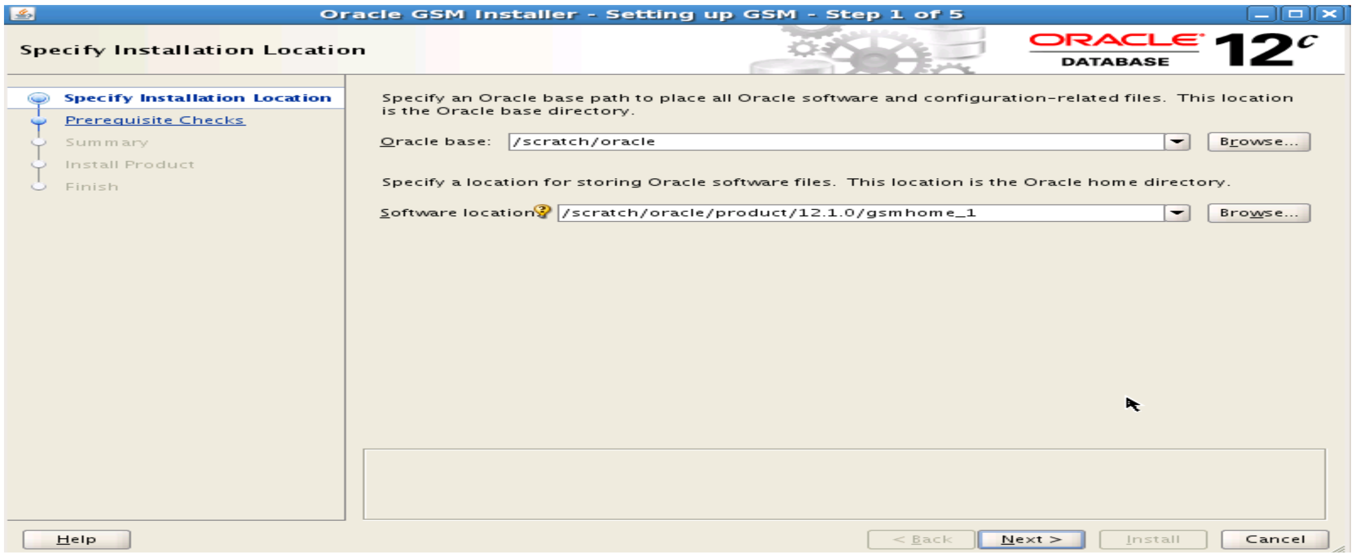
Run the Installer from the GSM media as shown in the screens below and follow the steps.



```
[srbattul@adc6160274 gsm]$ ls
install  response  runInstaller  stage  welcome.html
[srbattul@adc6160274 gsm]$ ./runInstaller
Starting Oracle Universal Installer...

Checking Temp space: must be greater than 120 MB.   Actual 84539 MB    Passed
Checking swap space: must be greater than 150 MB.   Actual 10047 MB    Passed
Checking monitor: must be configured to display at least 256 colors.    Actual 65536    Passed
Preparing to launch Oracle Universal Installer from /tmp/OraInstall2013-09-02_10-14-48AM. Please wait ...[srbattul@adc616
0274 gsm]$ 
```

Specify the Oracle Base and Oracle Home for the GSM.



Oracle GSM Installer - Setting up GSM - Step 1 of 5

Specify Installation Location

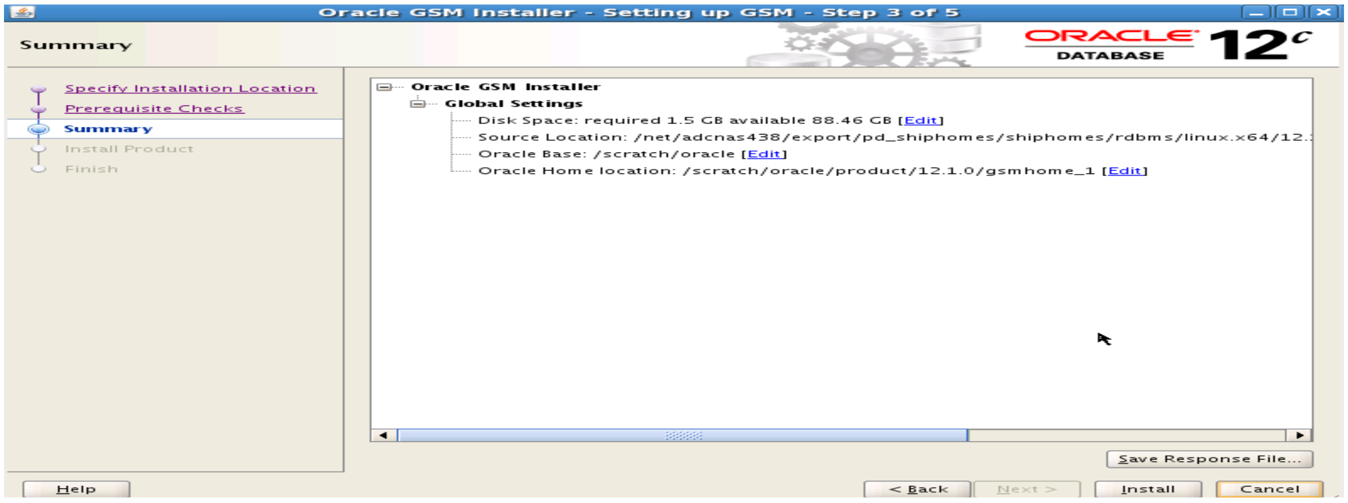
Specify an Oracle base path to place all Oracle software and configuration-related files. This location is the Oracle base directory.

Oracle base: /scratch/oracle

Specify a location for storing Oracle software files. This location is the Oracle home directory.

Software location: /scratch/oracle/product/12.1.0/gsmhome_1

Click Next.

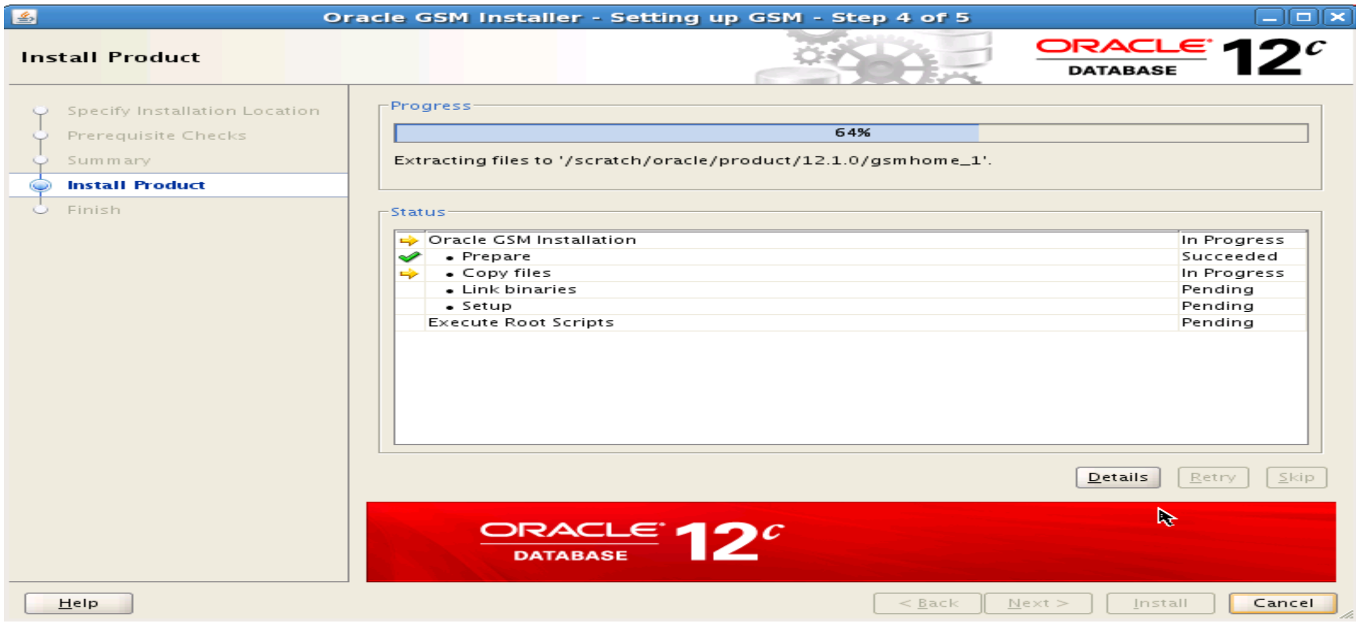


Oracle GSM Installer - Setting up GSM - Step 3 of 5

Summary

- Oracle GSM Installer
  - Global Settings
    - Disk Space: required 1.5 GB available 88.46 GB [Edit]
    - Source Location: /net/adcnas438/export/pd_shiphomes/shiphomes/rdbms/linux.x64/12.
    - Oracle Base: /scratch/oracle [Edit]
    - Oracle Home location: /scratch/oracle/product/12.1.0/gsmhome_1 [Edit]

Click Install.



Oracle GSM Installer - Setting up GSM - Step 4 of 5

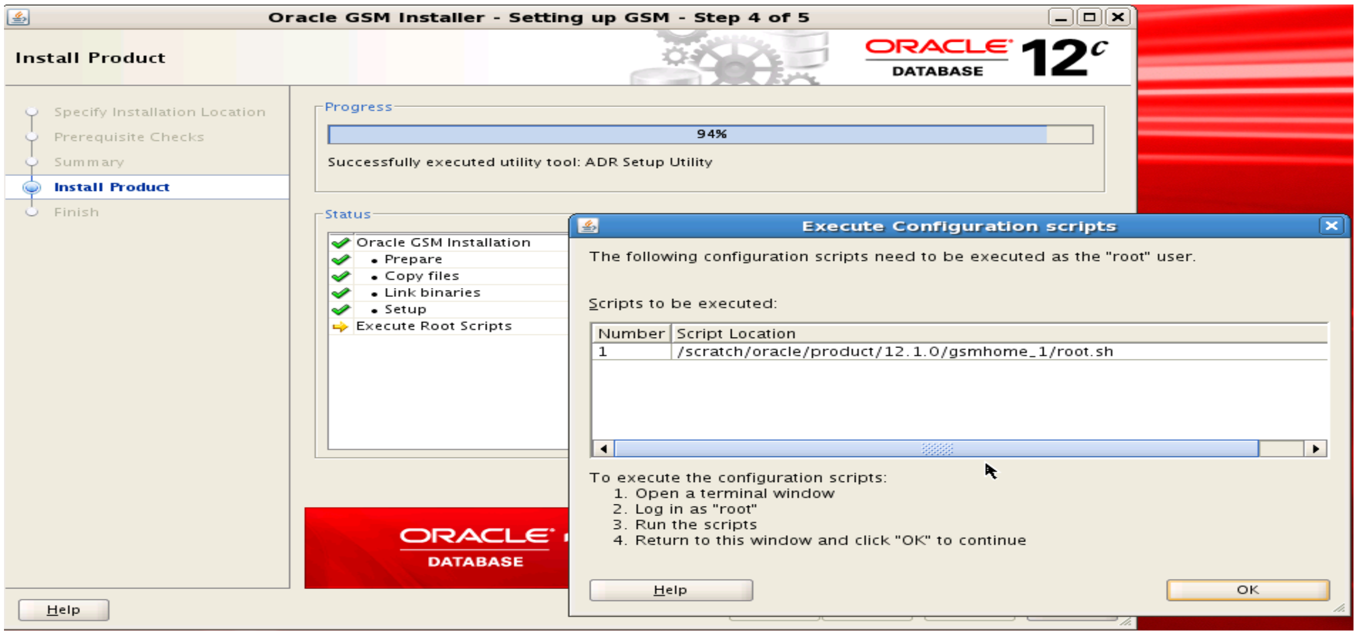
Install Product

Progress: 64%

Extracting files to '/scratch/oracle/product/12.1.0/gsmhome_1'.

Status

Oracle GSM Installation	In Progress
• Prepare	Succeeded
• Copy files	In Progress
• Link binaries	Pending
• Setup	Pending
Execute Root Scripts	Pending



Oracle GSM Installer - Setting up GSM - Step 4 of 5

Progress: 94%

Successfully executed utility tool: ADR Setup Utility

Execute Configuration scripts

The following configuration scripts need to be executed as the "root" user.

Scripts to be executed:

Number	Script Location
1	/scratch/oracle/product/12.1.0/gsmhome_1/root.sh

To execute the configuration scripts:
1. Open a terminal window
2. Log in as "root"
3. Run the scripts
4. Return to this window and click "OK" to continue

Open a new Terminal window and execute the root.sh

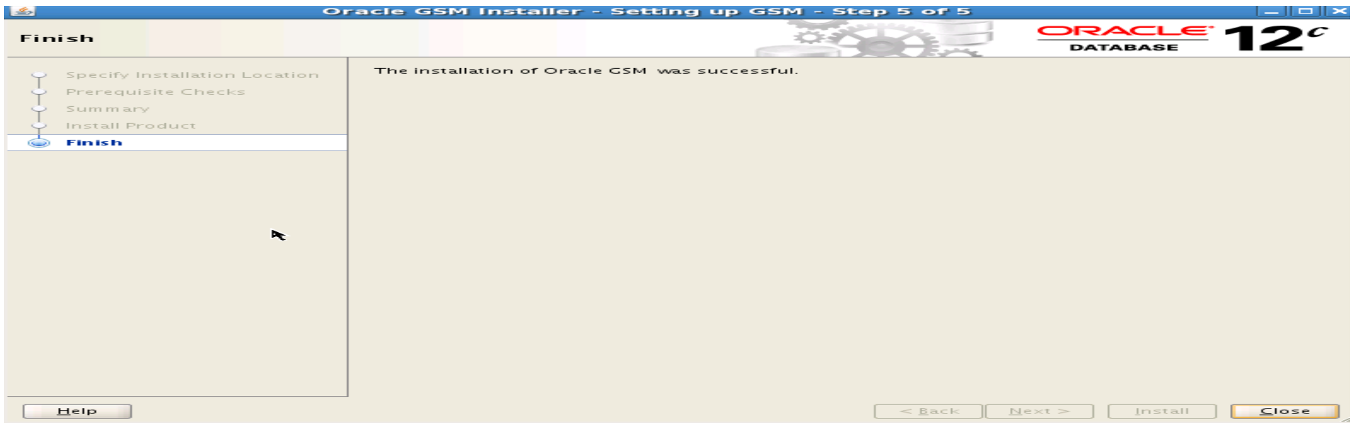
```
[srbattul@adc6160274 gsm]$ sudo /scratch/oracle/product/12.1.0/gsmhome_1/root.sh
Password:
Performing root user operation for Oracle 12c

The following environment variables are set as:
    ORACLE_OWNER= srbattul
    ORACLE_HOME=  /scratch/oracle/product/12.1.0/gsmhome_1

Enter the full pathname of the local bin directory: [/usr/local/bin]: /usr/bin
The contents of "dbhome" have not changed. No need to overwrite.
The contents of "oraenv" have not changed. No need to overwrite.
The contents of "coraenv" have not changed. No need to overwrite.

Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root script.
Now product-specific root actions will be performed.
[srbattul@adc6160274 gsm]$ _
```

Close the installer.



The installation of Oracle GSM was successful.

Note: Follow the GSM installation step for each of the identified GSM Nodes. In our examples, it will be on slc02pny and adc6160274 Nodes.

GDS Setup and Configuration

Add the following TNS entries of GDS catalog and pool databases to the GSM Home's tnsnames.ora.
Note: Repeat this on all GSM Nodes

```
GDSCAT =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = slc02pny.us.oracle.com)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = gdscat.us.oracle.com)
    )
  )

CHI,CHI.US.ORACLE.COM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = adc6160274)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = chi.us.oracle.com)
    )
  )

SFO,SFO.US.ORACLE.COM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = slc02pny)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = sfo.us.oracle.com)
    )
  )
```

Setup GDS Administrator Accounts & Privileges

Make sure that the listeners & databases sfo, chi and gdscat are up.

Connect to the pool database SFO (current Primary) and unlock the GSM user (as shown below).

```
$ cd $HOME
$ oraenv
```

```
 $ sqlplus system/oracle@sfo
 SQL> spool unlock_gdsusers_sfo.lst append
 SQL> show parameter db_unique
 SQL> alter user gsmuser account unlock;
 SQL> alter user gsmuser identified by passwd_gsmuser;
 SQL> spool off;
```

In this lab, we are hosting the GDS catalog in the gdscat database. So, connect to the gdscat (in the terminal window DB) and

- Unlock gsmcatuser
- Grant GSMADMIN_ROLE to an existing user or to a newly created user.

```
 SQL> connect system/oracle@gdscat
 SQL> spool unlock_gdscatuser.lst append
 SQL> show parameter db_unique
 SQL> alter user gsmcatuser account unlock;
 SQL> alter user gsmcatuser identified by passwd_gsmcatuser;
 SQL> create user mygdsadmin identified by passwd_mygdsadmin;
 SQL> grant gsmadmin_role to mygdsadmin;
 SQL> spool off
 SQL> exit
```

Note: Please make a note of the passwords of gsmuser, gsmcatuser and mygdsadmin

Configure GDS

Using the GDSCTL interface, the user with gsmadmin_role should complete the following steps.

Step 1: Create the GDS catalog

From the Terminal window of the West GSM node, set the following env variables

```
$cd $HOME
$setenv ORACLE_HOME /scratch/oracle/product/12.1.0/gsmhome_1
$setenv ORACLE_BASE /scratch/oracle
$setenv PATH $ORACLE_HOME/bin:$PATH
```

```
$ gdsctl

GDSCTL: Version 12.1.0.1.0 - Production on Tue Sep 03 06:04:30 PDT 2013

Copyright (c) 2011, 2012, Oracle.  All rights reserved.

Welcome to GDSCTL, type "help" for information.

Current GSM is set to GSMORA
```

```
GDSCTL> create catalog -database slc02pny:1521:gdscat –user
mygdsadmin/passwd mygdsadmin
```

Step 2: Add Global Service Managers (GSM)

Add the first GSM:

```
GDSCTL>add gsm -gsm gsmwest -listener 1571 -catalog slc02pny:1521:gdscat
"gsmcatuser" password:*****************
Create credential oracle.security.client.connect_string1
GSM successfully added
```

Note: As the –pwd parameter has not been provided, GDSCTL asks for the gsmcatuser password
Port 1571 is specified for the GSM listener of the GSMWEST.

At this stage, the GSM can be started:

```
GDSCTL>start gsm –gsm gsmwest
GSM is started successfully
GDSCTL>config gsm -gsm gsmwest
Name: gsmwest
Endpoint 1: (ADDRESS=(HOST=slc02pny.us.oracle.com)(PORT=1571)(PROTOCOL=tcp))
Local ONS port: 6123
Remote ONS port: 6234
ORACLE_HOME path: /scratch/oracle/product/12.1.0/gsmhome_1
GSM Host name: slc02pny.us.oracle.com
Region: west
Buddy
------------------------

GDSCTL>status gsm -gsm gsmwest
Alias                     GSMWEST
Version                   12.1.0.1.0
Start Date                21-MAR-2014 20:18:13
Trace Level               off
Listener Log File         /scratch/oracle/diag/gsm/slc02pny/gsmwest/alert/log.xml
Listener Trace File
/scratch/oracle/diag/gsm/slc02pny/gsmwest/trace/ora_7493_47345711117536.trc
Endpoint summary
(ADDRESS=(HOST=slc02pny.us.oracle.com)(PORT=1571)(PROTOCOL=tcp))
GSMOCI Version            0.1.10
Mastership                Y
Connected to GDS catalog  Y
Process Id                7495
Number of reconnections   0
Pending tasks.     Total  0
Tasks in  process. Total  0
Regional Mastership       TRUE
Total messages published  21282
Time Zone                 -07:00
Orphaned Buddy Regions:
     None
GDS region                regionora


GDSCTL>exit
```

Add the second GSM :

From the Terminal window of the East GSM Node, set the following env variables

```
$cd $HOME
$setenv ORACLE_HOME /scratch/oracle/product/12.1.0/gsmhome_1
$setenv ORACLE_BASE /scratch/oracle
$setenv PATH $ORACLE_HOME/bin:$PATH
```

```
$ gdsctl
GDSCTL: Version 12.1.0.1.0 - Production on Tue Sep 03 06:10:39 PDT 2013

Copyright (c) 2011, 2012, Oracle.  All rights reserved.

Welcome to GDSCTL, type "help" for information.

Current GSM is set to GSMORA
GDSCTL>
```

```
GDSCTL>add gsm -gsm gsmeast -listener 1572 -catalog slc02pny:1521:gdscat
"gsmcatuser" password:*****************
Create credential oracle.security.client.connect_string1
GSM successfully added
```

Note: As the –pwd parameter has not been provided, GDSCTL asks for the gsmcatuser password
Port 1572 is specified for the GSM listener of the GSMEAST.

Start the second GSM:

```
GDSCTL>start gsm –gsm gsmeast
GSM is started successfully
GDSCTL>config gsm -gsm gsmeast
Name: gsmeast
Endpoint 1: (ADDRESS=(HOST=adc6160274.us.oracle.com)(PORT=1572)(PROTOCOL=tcp))
Local ONS port: 6123
Remote ONS port: 6234
ORACLE_HOME path: /scratch/oracle/product/12.1.0/gsmhome_1
GSM Host name: adc6160274.us.oracle.com
Region: east
Buddy
------------------------
```

```
GDSCTL>status gsm -gsm gsmeast
Alias                     GSMEAST
Version                   12.1.0.1.0
Start Date                21-MAR-2014 20:23:33
Trace Level               off
Listener Log File         /scratch/oracle/diag/gsm/adc6160274/gsmeast/alert/log.xml
Listener Trace File
/scratch/oracle/diag/gsm/adc6160274/gsmeast/trace/ora_11548_46969260647648.trc
Endpoint summary
(ADDRESS=(HOST=adc6160274.us.oracle.com)(PORT=1572)(PROTOCOL=tcp))
GSMOCI Version            0.1.10
Mastership                N
Connected to GDS catalog  Y
Process Id                11550
Number of reconnections   0
Pending tasks.     Total  0
Tasks in  process. Total  0
Regional Mastership       FALSE
Total messages published  0
Time Zone                 -07:00
Orphaned Buddy Regions:
     None
GDS region                regionora
```

Step 3: Create the GDS regions

We will create two regions, west and east.

```
GDSCTL>add region -region west, east
```

Assign a region to each GSM:

```
GDSCTL>modify gsm –gsm gsmeast -region east
GSM modified
GDSCTL>exit
```

From the Terminal window of the West GSM node:

```
$ gdsctl
GDSCTL>Connect mygdsadmin/passwd_mygdsadmin@gdscat
Catalog connection is established
GDSCTL>modify gsm –gsm gsmwest -region west
GSM modified
```

Step 4: Create the GDS pools

As we need just one gdspool, we can either use the pre-defined one dbpoolora or create a new one called sales. Note: Once the regions are defined and GSM assignment is done, you can stay in any GSM env from here onwards and be able to manage the GDS configuration.

```
GDSCTL>add gdspool -gdspool sales
```

Step 5: Add Data Guard broker configuration to the GDS pool

We will add the Data Guard broker configuration to the sales gdspool.

Using DGMGRL, first let's check how the Data Guard configuration looks like from one of the database nodes.

```
DGMGRL> connect sysdg/oracle
Connected as SYSDG.
DGMGRL> show configuration

Configuration - sfochi

  Protection Mode: MaxAvailability
  Databases:
  sfo - Primary database
    chi - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

Observe that in this configuration, "SFO" is the current Primary and "CHI" is the current Standby.

Note: Always add the broker config connecting to the current PRIMARY db. In this case "sfo" is the PRIMARY.

```
GDSCTL>add brokerconfig -connect slc02pny:1521:sfo -region west -gdspool sales
"gsmuser" password:**************
DB Unique Name: sfo
```

Now, we will assign the database "chi" to the EAST region.

```
GDSCTL>modify database -database chi -region east -gdspool sales
```

To check that the databases are registered to the GSM run:

```
GDSCTL> databases

Database: "sfo" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region:
west
   Registered instances:
     sales%1
Database: "chi" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1 Region:
east
   Registered instances:
     sales%11
```

Step 6: Create and Start Global Service in a pool

In the following example, we will create a service called sales_reporting_service which should always run on the standby database. In the event the standby database is not available, the service can run on the primary database.

```
GDSCTL>add service -service sales_reporting_srvc -gdspool sales –preferred_all –role
PHYSICAL_STANDBY –failover_primary

GDSCTL>start service –service sales_reporting_srvc –gdspool sales
GDSCTL>status service –service sales_reporting_srvc
Service "sales_reporting_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%11", name: "chi", db: "chi", region: "east", status: ready.

Note: You can also start all services in a given pool with one gdsctl command as shown below:
GDSCTL>start service -gdspool sales
```

To observe the configuration info of a given service:

```
GDSCTL>config service -service sales_reporting_srvc
Name: sales_reporting_srvc
Network name: sales_reporting_srvc.sales.oradbcloud
Pool: sales
Started: Yes
Preferred all: Yes
Locality: ANYWHERE
Region Failover: No
Role: PHYSICAL_STANDBY
Primary Failover: Yes
Lag: ANY
Runtime Balance: SERVICE_TIME
Connection Balance: LONG
Notification: Yes
TAF Policy: NONE
Policy: AUTOMATIC
DTP: No
Failover Method: NONE
Failover Type: NONE
Failover Retries:
Failover Delay:
Edition:
PDB:
Commit Outcome:
Retention Timeout:
Replay Initiation Timeout:
Session State Consistency: DYNAMIC
SQL Translation Profile:


Databases
------------------------
Database                     Preferred Status
--------                     --------- ------
chi                          Yes       Enabled
sfo                          Yes       Enabled
```

To display the config info of all the components that are part of a given GDS configuration, the following command can be run in gdsctl:

```
GDSCTL>config

Regions
------------------------
Name                          Buddy
----                          -----
regionora           🡐 Default GDS Region
west
east

GSMs
------------------------
gsmeast
gsmwest

GDS pools
------------------------
dbpoolora           🡐 Default GDS Pool
sales

Databases
------------------------
chi
sfo

Services
------------------------
sales_reporting_srvc

GDSCTL pending requests
------------------------
Command                       Object                        Status
-------                       ------                        ------

Global properties
------------------------
Name: oradbcloud
Master GSM: gsmwest
```

Now that the global service has been created and started, we can query DBA_SERVICES and V$ACTIVE_SERVICES to learn more about the global services. So let’s connect to the standby database where the global service is currently running on:

```
SQL> column name format a30
SQL> column network_name format a40
SQL> column global format a10
SQL> set linesize 120

SQL> select name, network_name, global_service from dba_services;

NAME                           NETWORK_NAME                             GLO
------------------------------ ---------------------------------------- ---
SYS$BACKGROUND                                                          NO
SYS$USERS                                                               NO
sfo_DGB                        sfo_DGB                                  NO
chi_DGB                        chi_DGB                                  NO
sfoXDB                         sfoXDB                                   NO
sfo.us.oracle.com              sfo.us.oracle.com                        NO
chiXDB                         chiXDB                                   NO
chi.us.oracle.com              chi.us.oracle.com                        NO
sales_reporting_srvc           sales_reporting_srvc.sales.oradbcloud    YES

9 rows selected.
```

Note: For the sales_reporting_srvc, the value of the column GLOBAL_SERVICE" is "Yes", denoting that it is a global service**.**

```
SQL> select name, network_name, global from v$active_services;

NAME                           NETWORK_NAME                             GLOBAL
------------------------------ ---------------------------------------- ----------
sales_reporting_srvc           sales_reporting_srvc.sales.oradbcloud    YES
chi_DGB                        chi_DGB                                  NO
chiXDB                         chiXDB                                   NO
chi.us.oracle.com              chi.us.oracle.com                        NO
SYS$BACKGROUND                                                          NO
SYS$USERS                                                               NO

6 rows selected.
```

Note: The query above shows that currently the sales_reporting_srvc global service is listed in the v$active_services because it has been started.

Step 7: Setup TNS entries for clients

Add the TNS entries (based on GSM listeners) to the client's tnsnames.ora. Here is an example TNS entry for the "sales_reporting_srvc" application clients – where SERVICE_NAME is the name of the global service and the REGION is the region that the client is coming from.
Note: The connect descriptor in the tnsnames.ora in a GDS config will be using the GSM listener end points and not the RAC SCAN listeners or local listeners.

```
sales_reporting_srvc =
  (DESCRIPTION =
   (FAILOVER=ON)
    (ADDRESS_LIST =
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = slc02pny)(PORT = 1571))
    )
    (ADDRESS_LIST =
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = adc6160274)(PORT = 1572))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = sales_reporting_srvc.sales.oradbcloud) (REGION=WEST)
    )
  )
```

Note: 1571 is the GSM listener port on slc02pny and 1572 is the GSM listener port on adc6160274.

Client connectivity with GDS

Clients can connect to the pool databases either via the Easy Connect Naming or with the TNSNAMES.

```
Via Easy Connect Naming Method:
SQL> connect system/oracle@//slc02pny:1571/sales_reporting_srvc.sales.oradbcloud

Via TNSNAMES Connect Descriptor:
SQL> connect system/oracle@sales_reporting_srvc
```

The workload management capabilities for a set of replicas can be achieved just by setting the relevant global service attributes (e.g. clbgoal for connect-time load balancing; local_only for region based routing etc.) and employing the global service in the TNS entry of the clients. The following test cases walk you through various capabilities of GDS.

Test Case I: Global Service Failover

To see how the failover of global services happens, execute “shutdown immediate” on one of the databases (“chi”) and observe that the services are automatically started in the other pool database (“sfo”).

With GDSCTL, observe that the sales_reporting_srvc global service is currently running on the standby “chi”as per the Service attributes that we have defined.

```
GDSCTL>services
Service "sales_reporting_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%11", name: "chi", db: "chi", region: "east", status: ready.

GDSCTL>databases
Database: "sfo" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region: west
   Service: "sales_reporting_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
Database: "chi" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1 Region: east
   Service: "sales_reporting_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%11
```

Connect to the pool database “chi” (in the terminal window DB), using sqlplus

```
$ cd $HOME
$ oraenv
setenv ORACLE_SID chi
sqlplus sys/oracle as sysdba
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

With GDSCTL, observe that the sales_reporting_srvc global service is automatically failed over to “sfo” database as per the Service attributes that we have defined.

```
GDSCTL>services
Service "sales_reporting_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%1", name: "sfo", db: "sfo", region: "west", status: ready.

GDSCTL>databases
Database: "sfo" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region: west
   Service: "sales_reporting_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
Database: "chi" Registered: N State: Ok ONS: N. Role: N/A Instances: 0 Region: east
   Service: "sales_reporting_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
```

At this point, you may stop and remove the service that we have created for this test case.

```
GDSCTL>stop service -service sales_reporting_srvc -gdspool sales
GDSCTL>remove service -gdspool sales -service sales_reporting_srvc
```

Now, you can bring the “chi” up

```
SQL> startup
```

The above test case illustrated the automatic global service failover capability of Oracle GDS.

Test Case II: Role based Global Services

When a Data Guard role transition is performed either manually or via Fast-Start Failover, GDS automatically relocates the global services based on the role of the databases. GDS does this without the Oracle Clusterware. To comprehend how the role-based global services function, create two global services. sales_entry_srvc targeted to run on the Primary (“sfo”) and sales_adhoc_srvc targeted to run on the standby(“chi”). And, execute the Data Guard switchover operation and observe that upon the role-change, the sales_entry_srvc global service automatically gets relocated to the new primary (“chi”) and the sales_adhoc_srvc to the new Standby (“sfo”).

With GDSCTL, create the global services.

```
GDSCTL>add service -service sales_entry_srvc -gdspool sales -preferred_all -role PRIMARY
GDSCTL>start service -service sales_entry_srvc -gdspool sales
GDSCTL>add service -service sales_adhoc_srvc -gdspool sales -preferred_all -role
PHYSICAL_STANDBY
GDSCTL>start service -service sales_adhoc_srvc -gdspool sales
GDSCTL>services
Service "sales_adhoc_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%11", name: "chi", db: "chi", region: "east", status: ready.
Service "sales_entry_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%1", name: "sfo", db: "sfo", region: "west", status: ready.

GDSCTL>databases
Database: "sfo" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region: west
   Service: "sales_adhoc_srvc" Globally started: Y Started: N
            Scan: Y Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: Y
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%1
Database: "chi" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1 Region: east
   Service: "sales_adhoc_srvc" Globally started: Y Started: Y
            Scan: Y Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: N
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%11
```

On any of the database nodes, With DGMGRL, perform the Data Guard switchover.

```
[srbattul@slc02pny ~]$ dgmgrl
DGMGRL for Linux: Version 12.1.0.1.0 - 64bit Production
Copyright (c) 2000, 2012, Oracle. All rights reserved.
Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sysdg/oracle
Connected as SYSDG.
DGMGRL> show configuration
Configuration - sfochi
  Protection Mode: MaxAvailability
  Databases:
  sfo - Primary database
    chi - Physical standby database

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

```
DGMGRL> switchover to chi
Performing switchover NOW, please wait...
Operation requires a connection to instance "chi" on database "chi"
Connecting to instance "chi"...
Connected as SYSDBA.
New primary database "chi" is opening...
Operation requires startup of instance "sfo" on database "sfo"
Starting instance "sfo"...
ORACLE instance started.
Database mounted.
Database opened.
Switchover succeeded, new primary is "chi"
DGMGRL> show configuration

Configuration - sfochi

  Protection Mode: MaxAvailability
  Databases:
  chi - Primary database
    sfo - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

With GDSCTL, observe that the sales_entry_srvc global service is automatically relocated to the new Primary (“Chi”) and the sales_adhoc_srvc global service to the new standby (“sfo”)

```
GDSCTL>services
Service "sales_adhoc_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%1", name: "sfo", db: "sfo", region: "west", status: ready.
Service "sales_entry_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%11", name: "chi", db: "chi", region: "east", status: ready.

GDSCTL>databases
Database: "sfo" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1 Region:
west
   Service: "sales_adhoc_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
Database: "chi" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region:
east
   Service: "sales_adhoc_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%11
```

At this point, you may stop and remove the services that we have created for this test case.

```
GDSCTL>stop service -service sales_entry_srvc -gdspool sales
GDSCTL>remove service -gdspool sales -service sales_entry_srvc

GDSCTL>stop service -service sales_adhoc_srvc -gdspool sales
GDSCTL>remove service -gdspool sales -service sales_adhoc_srvc
```

The above test case illustrated the automatic role based global services capability of Oracle GDS.

Test Case III: Replication Lag based Routing

Sometimes the Data Guard standby databases may lag behind the primary database due to various reasons. If the replication lag exceeds the lag limit, the global service is relocated to another available database that lags below the threshold. With the –failover_primary clause, we can even relocate the service to the Primary database.
To understand the Lag tolerance based routing, let's create a global Service **sales_reader_lag15_srvc** and set the -lag attribute to 15 seconds. We will artificially create the lag by turning off the Apply process. We then will observe that once the lag exceeds the 15 seconds threshold, GDS automatically relocates the global service to the Primary (since we used the –failover_primary clause)

```
GDSCTL>add service -service sales_reader_lag15_srvc -gdspool sales -preferred_all -role
PHYSICAL_STANDBY -lag 15 -failover_primary

GDSCTL>start service -service sales_reader_lag15_srvc -gdspool sales

GSCTL>services
Service "sales_reader_lag15_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%1", name: "sfo", db: "sfo", region: "west", status: ready.

GDSCTL>databases
Database: "sfo" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1 Region: west
   Service: "sales_reader_lag15_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
Database: "chi" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region: east
   Service: "sales_reader_lag15_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%11

GDSCTL>config service -service sales_reader_lag15_srvc
Name: sales_reader_lag15_srvc
Network name: sales_reader_lag15_srvc.sales.oradbcloud
Pool: sales
Started: Yes
Preferred all: Yes
Locality: ANYWHERE
Region Failover: No
Role: PHYSICAL_STANDBY
Primary Failover: Yes
Lag: 15
Runtime Balance: SERVICE_TIME
Connection Balance: LONG
Notification: Yes
TAF Policy: NONE
Policy: AUTOMATIC
… < output edited for space>

Databases
------------------------
Database                      Preferred Status
--------                      --------- ------
chi                           Yes       Enabled
sfo                           Yes       Enabled
```

Using DGMGRL on any of the database nodes:

```
DGMGRL> show configuration

Configuration - sfochi
  Protection Mode: MaxAvailability
  Databases:
  chi - Primary database
    sfo - Physical standby database

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS

DGMGRL> show database chi
Database - chi

  Role:                PRIMARY
  Intended State:      TRANSPORT-ON
  Instance(s):
    chi

Database Status:
SUCCESS

DGMGRL> show database sfo
Database - sfo

  Role:                PHYSICAL STANDBY
  Intended State:      APPLY-ON
  Transport Lag:       0 seconds (computed 1 second ago)
  Apply Lag:           0 seconds (computed 1 second ago)
  Apply Rate:          1.10 MByte/s
  Real Time Query:     ON
  Instance(s):
    sfo

Database Status:
SUCCESS

DGMGRL> edit database sfo set state='APPLY-OFF';
Succeeded.

DGMGRL> show database sfo
Database - sfo

  Role:                PHYSICAL STANDBY
  Intended State:      APPLY-OFF
  Transport Lag:       0 seconds (computed 1 second ago)
  Apply Lag:           15 seconds (computed 1 second ago)
  Apply Rate:          (unknown)
  Real Time Query:     OFF
  Instance(s):
    sfo

Database Status:
SUCCESS
```

Wait for 15 seconds and run the services command from GDSCTL and observe that the sales_reader_lag15_srvc global service has failed over to primary.

```
GDSCTL>services
Service "sales_reader_lag15_srvc.sales.oradbcloud" has 1 instance(s). Affinity: ANYWHERE
   Instance "sales%11", name: "chi", db: "chi", region: "east", status: ready.

GDSCTL>databases
Database: "sfo" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1 Region: west
   Service: "sales_reader_lag15_srvc" Globally started: Y Started: N
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%1
Database: "chi" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region: east
   Service: "sales_reader_lag15_srvc" Globally started: Y Started: Y
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%11
```

At this point, you may stop and remove the services that we have created for this test case.

```
GDSCTL>stop service -service sales_reader_lag15_srvc -gdspool sales
GDSCTL>remove service -gdspool sales -service sales_reader_lag15_srvc
```

Also, make sure that the Apply Process is started.

This test case illustrated replication lag tolerance based routing for Active Data Guard configuration which allows applications achieve better data quality. Instead of accessing data in the standby database that is lagging behind, we can automatically relocate the service to a database which is not lagging more than the defined threshold.

Test Case IV: Locality based Routing

The Locality based routing capability allows customers to maximize their application performance by avoiding latency overhead accessing databases in remote regions. With GDS, customers can choose to configure client connections to be always routed among a set of replicated databases in a local region. To exhibit this capability, let's create a new global service called sales_local_only_srvc and set the –locality attribute to LOCAL_ONLY.

```
GDSCTL>add service -service sales_reader_local_only_srvc -gdspool sales -preferred_all –
locality LOCAL_ONLY

GDSCTL>start service -service sales_reader_local_only_srvc -gdspool sales
```

Add the following TNS entry to the client's tnsnames.ora file. But, for this exercise, we are appending this entry to the $ORACLE_HOME/network/admin for the database server on the WEST ("slc02pny"). Note that we are specifying the REGION as WEST. And, when we launch a SQLPLUS session from the HOME that has the TNS entry shown below, we will always be routed to a database in the GDS REGION West.

```
sales_reader_local_only_srvc=
  (DESCRIPTION =
   (FAILOVER=ON)
    (ADDRESS_LIST =
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = slc02pny)(PORT = 1571))
    )
    (ADDRESS_LIST =
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = adc6160274)(PORT = 1572))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = sales_reader_local_only_srvc.sales.oradbcloud) (REGION=WEST)
    )
  )

GDSCTL>services
Service "sales_reader_local_only_srvc.sales.oradbcloud" has 2 instance(s). Affinity:
LOCALONLY
   Instance "sales%1", name: "sfo", db: "sfo", region: "west", status: ready.
   Instance "sales%11", name: "chi", db: "chi", region: "east", status: ready.

GDSCTL>databases
Database: "sfo" Registered: Y State: Ok ONS: N. Role: PH_STNDBY Instances: 1 Region: west
   Service: "sales_reader_local_only_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
Database: "chi" Registered: Y State: Ok ONS: N. Role: PRIMARY Instances: 1 Region: east
   Service: "sales_reader_local_only_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%11
```

```
GDSCTL>config service -service sales_reader_local_only_srvc -gdspool sales
Name: sales_reader_local_only_srvc
Network name: sales_reader_local_only_srvc.sales.oradbcloud
Pool: sales
Started: Yes
Preferred all: Yes
Locality: LOCAL_ONLY
Region Failover: No
Role: NONE
Primary Failover: No
Lag: ANY
Runtime Balance: SERVICE_TIME
Connection Balance: LONG
Notification: Yes
TAF Policy: NONE
Policy: AUTOMATIC
DTP: No
Failover Method: NONE
Failover Type: NONE
Failover Retries:
Failover Delay:
Edition:
PDB:
Commit Outcome:
Retention Timeout:
Replay Initiation Timeout:
Session State Consistency: DYNAMIC
SQL Translation Profile:


Databases
------------------------
Database                     Preferred Status
--------                     --------- ------
chi                          Yes       Enabled
sfo                          Yes       Enabled
```

```
[srbattul@slc02pny admin]$ sqlplus sys/oracle@sales_reader_local_only_srvc as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Wed Apr 2 17:15:40 2014

Copyright (c) 1982, 2013, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options


SQL> show parameter db_unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      sfo
SQL> exit
```

In the above example, since the REGION attribute is set to WEST in the client's TNS entry, the client is routed to the SFO database in the GDS region West.

If the TNS entry's REGION attribute is updated to EAST (to mimic a client from the EAST region), the client will be routed to the CHI database in the GDS region East.

```
sales_reader_local_only_srvc=
  (DESCRIPTION =
   (FAILOVER=ON)
    (ADDRESS_LIST =
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = slc02pny)(PORT = 1571))
    )
    (ADDRESS_LIST =
     (LOAD_BALANCE=ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = adc6160274)(PORT = 1572))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = sales_reader_local_only_srvc.sales.oradbcloud) (REGION=EAST)
    )
  )

[srbattul@slc02pny admin]$ sqlplus sys/oracle@sales_reader_local_only_srvc as sysdba

… < output edited for space>

SQL> show parameter db_unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      chi
SQL> exit
```

The above test case illustrated the region affinity based routing capability of Oracle GDS.

Test Case V: Connect-time Load Balancing

The load balancing capabilities over a set of replicas can be achieved by setting the *–clbgoal* global service attribute and employing the global service in the TNS entry of the clients. The *–clbgoal* can be set to either LONG or SHORT. For applications with long lived connections (e.g. connection pools and SQL*Forms sessions), the *–clbgoal* is set to LONG.

Here is an example of setting the global service attributes in order to enable the GDS connect-time load balancing:

```
GDSCTL>add service -service sales_clb_srvc -gdspool sales –preferred_all -clbgoal LONG
```

Note: The following next step regarding the simulator is a suggestion and not a tool provided as part of the HOL.

Once the global services are created and are reflected in the TNS entries, run your application or simulator to spawn database connections and observe (with monitoring tools such as EM Cloud Control or oratop) that the database connections are well balanced based on load, network latency, clbgoal and other global service attributes.

Test Case VI: Run-time Load Balancing

The run-time load balancing capability over a set of replicas can be achieved by setting the *–clbgoal* and *–rlbgoal* global service attributes and employing the global service in the TNS entry of the clients.
Once *–rlbgoal* is set for the global Service, configure the clients to subscribe for FAN events via GSM ONS ports and Enable FCF. (Refer to the GDS Documentation link shown in the Resources Section for details)
The *–rlbgoal* can be set to either SERVICE_TIME or THROUGHPUT.

Here is the example of setting the global service attributes in order to enable the GDS connect-time and run-time load balancing:

```
GDSCTL>add service -service sales_clb_rlb_srvc -gdspool sales –preferred_all -clbgoal
LONG –rlbgoal SERVICE_TIME
```

Note: The following next step regarding the simulator is a suggestion and not a tool provided as part of the HOL.

Run your applications or simulators that use Oracle Integrated clients (e.g. JDBC/UCP, OCI, ODP.NET, and WLS) and observe that the databases are well balanced even under varying load conditions (via external load, backups etc). We can confirm that RLB is working by noting that the response times of the global services are equalized.

Conclusion

This concludes the exercises for the Oracle Database 12c Global Data Services (for Active Data Guard) hands-on lab. In this lab, you have explored and exercised some of the Global Data Services capabilities.

You have performed the following exercises:

- Installed Global Service Managers
- Deployed GDS Framework for your data cloud
- Configured global services for the following GDS features
  - Inter-database global Service failover
  - Role based global services
  - Lag based routing
  - Locality based workload routing
  - Connect-time load balancing
  - Run-time load balancing
- Obtained familiarity with the GDSCTL interface.

Resources

The following documentation is available for Global Data Services

- GDS White Paper
  - o http://www.oracle.com/technetwork/database/availability/global-data-services-12c-wp-1964780.pdf

- GDS OOW Presentations
  - o http://www.oracle.com/technetwork/database/availability/globaldataservices-2030199.pdf
  - o http://www.oracle.com/technetwork/database/availability/amway-2030200.pdf
  - o http://www.oracle.com/technetwork/database/availability/8397-gds-paypal-1966397.pdf

- Oracle® Global Data Services 12c Release 1 (12.1)
  - o http://www.oracle.com/pls/db121/to_toc?pathname=doc.121/e22100/toc.htm

- GDS OTN Page:
  - o http://www.oracle.com/goto/gds

The Maximum Availability Architecture Best Practices papers are on Oracle OTN.

- o Best Practices for High Availability -- Maximum Availability Architecture (MAA)
  - o http://www.oracle.com/technetwork/database/features/availability/maa-096107.html

Oracle Global Data Services
with Active Data Guard
Hands-On Lab

May 2014

Author: Nagesh Battula
Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com