Monitoring Oracle Cluster

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

Oracle Real Application Clusters (RAC) allows Oracle Database to run any packaged or custom application, unchanged across a server pool. This provides the highest levels of availability and the most flexible scalability. If a server in the pool fails, the Oracle database continues to run on the remaining servers. When you need more processing power, simply add another server to the pool without taking users offline. With Oracle Real Application Clusters, Oracle de-couples the Oracle Instance (the processes and memory structures running on a server to allow access to the data) from the Oracle database (the physical structures residing on the storage storing the data, commonly referred to as the datafiles).

An Oracle RAC database is a clustered database. A cluster can be described as a pool of independent servers that co-operate as a single system. A clustered database is a single database that can be accessed by multiple instances, where each instance runs on a separate server in the server pool. A server pool is made up of 1 or more servers, each having a public LAN connection, an interconnect connection, and must be connected to a shared pool of storage. Server Pools provide improved fault resilience and modular incremental system growth over single symmetric multi-processor (SMP) systems. In the event of a system failure, clustering ensures high availability to users. Also, when additional resources are required, additional servers and instances can easily be added to the server pool with no downtime. Each server in a server pool is called a cluster node.

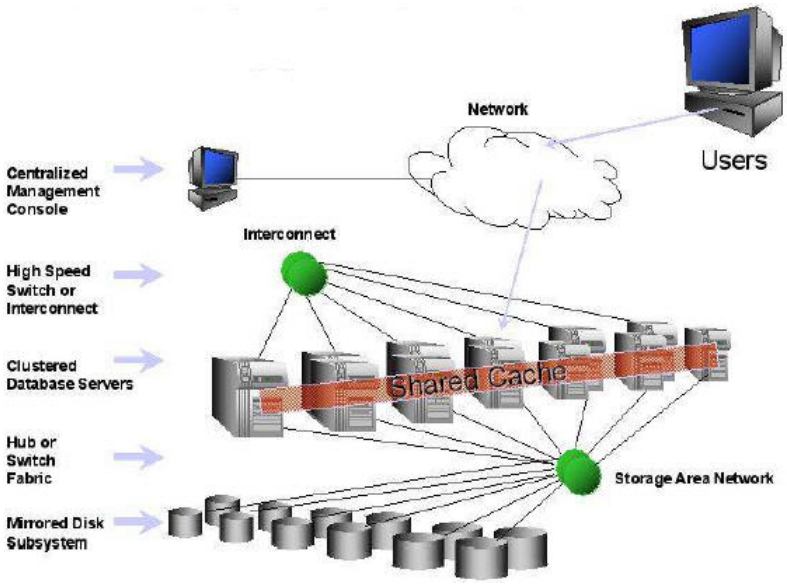


Figure 1.1: The Oracle RAC architecture

Besides cluster nodes, an Oracle RAC database requires Oracle clusterware and shared pool storage. Oracle Clusterware provides a complete clustering solution and supports any application. It is a prerequisite for all Oracle RAC implementations, and monitors and manages Oracle Real Application Cluster databases. When a server in the server pool is started, all instances, listeners and services are automatically started. If an instance fails, the Oracle Clusterware will automatically restart the instance so that the service is often restored before the administrator notices it was down.

Oracle Real Application Clusters is a shared everything architecture. All nodes in the cluster share all storage used for an Oracle RAC database. The type of storage pool used can be network attached storage (NAS), storage area network (SAN), or SCSI disks. Your storage choice is dictated by the server hardware choice and the hardware supported by your hardware vendor. The key to choosing an appropriate storage pool is choosing a storage system that will provide scaleable I/O for your application and an I/O system that will scale as additional servers are added to the pool.

To connect to an Oracle RAC database, applications use a virtual IP address. This IP address is assigned to each server in the cluster so that, if a node fails, the virtual IP is failed over to another node in the cluster to provide an immediate “node down”-response to incoming connection requests. This increases the availability for applications.

Owing to the high availability and load-balancing capabilities, clustered databases are often the preferred backend in many mission-critical IT infrastructures delivering key end-user services. This is why, the non-availability of a clustered database to an application, or a tablespace contention experienced by the clustered database, or an unusually high locking activity on the database, may cause the dependent end-user services to suffer. To avoid this, the Oracle RAC database should be continuously monitored. For this purpose, the eG Enterprise provides a specialized monitoring model.

Chapter 2: Administering the eG Manager to Monitor an Oracle Cluster

1. Log into the eG administrative interface.
2. eG Enterprise cannot automatically discover the Oracle Cluster component. You need to manually add the server using the **COMPONENTS** page (see Figure 2.1) that appears when the Infrastructure -> Components -> Add/Modify menu sequence is followed. Remember that components manually added are managed automatically.

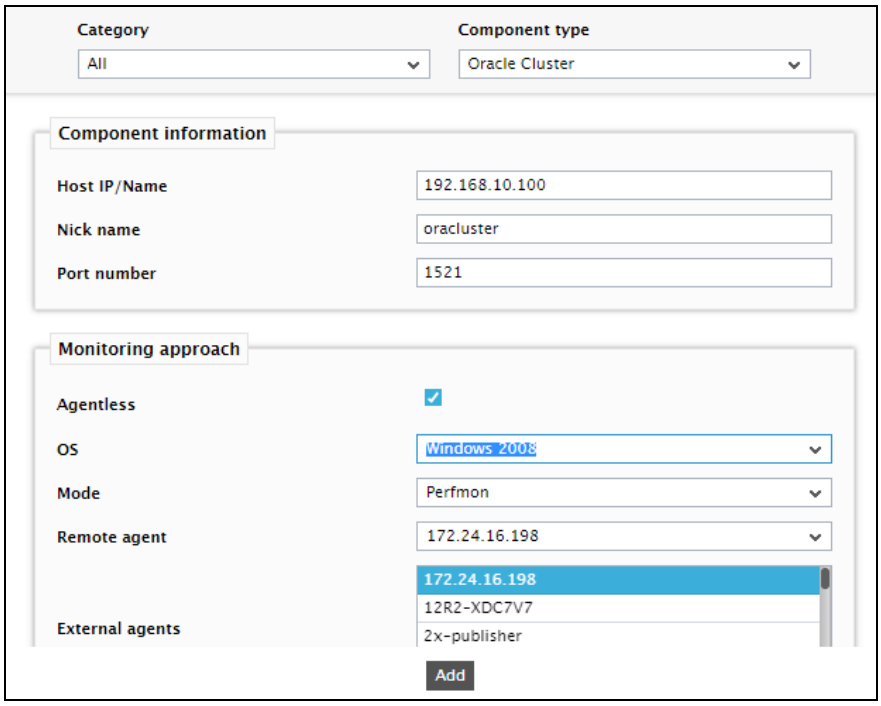
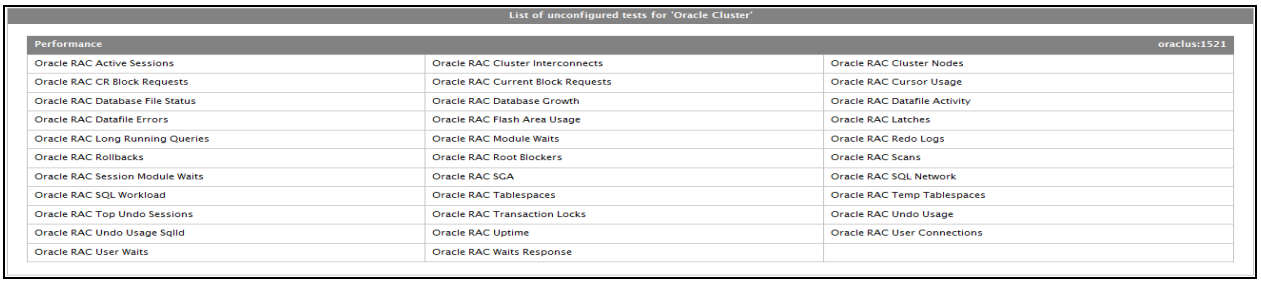


Figure 2.1: Adding the Oracle Cluster

3. To manage the cluster, specify the VIP or the SCAN IP address of any node in that cluster against **Host IP** in Figure 2.1. VIP or virtual IP address is the IP that the Oracle Clusterware assigns to each node in the cluster, upon cluster startup. SCAN provides a single domain name via DNS, allowing end-users to address a RAC cluster as-if it were a single IP address.
4. Then, provide a **Nick name** for the cluster.
5. eG Enterprise recommends that the Oracle cluster be monitored in an 'agentless' manner. Therefore, select the **Agentless** check box in Figure 1 to enable agentless monitoring of the cluster.

6. Next, pick the **OS** on which the cluster node is running. Then, from the Mode drop-down, select the agentless monitoring mechanism you want the remote agent to use. If the **OS** you have chosen is Windows, then pick *Perfmon* as the **Mode**. If the **OS** you have chosen is a flavor of Unix or Solaris, then pick *Rexec* as the Mode.
7. Then, select the **Remote agent** and **External agent** that will be monitoring the cluster.
8. Finally, click the **Add** button to register the changes.
9. When you attempt to sign out, a list of unconfigured tests appears (see Figure 2.2).


List of unconfigured tests for 'Oracle Cluster'

Performance		oraclus:1521
Oracle RAC Active Sessions	Oracle RAC Cluster Interconnects	Oracle RAC Cluster Nodes
Oracle RAC CR Block Requests	Oracle RAC Current Block Requests	Oracle RAC Cursor Usage
Oracle RAC Database File Status	Oracle RAC Database Growth	Oracle RAC Datafile Activity
Oracle RAC Datafile Errors	Oracle RAC Flash Area Usage	Oracle RAC Latches
Oracle RAC Long Running Queries	Oracle RAC Module Waits	Oracle RAC Redo Logs
Oracle RAC Rollbacks	Oracle RAC Root Blockers	Oracle RAC Scans
Oracle RAC Session Module Waits	Oracle RAC SGA	Oracle RAC SQL Network
Oracle RAC SQL Workload	Oracle RAC Tablespaces	Oracle RAC Temp Tablespaces
Oracle RAC Top Undo Sessions	Oracle RAC Transaction Locks	Oracle RAC Undo Usage
Oracle RAC Undo Usage Sqlid	Oracle RAC Uptime	Oracle RAC User Connections
Oracle RAC User Waits	Oracle RAC Waits Response	

Figure 2.2: List of tests to be configured for the Oracle Cluster

10. Click on the **Oracle Active Sessions** test to configure it. To know how to configure the test, refer to**Monitoring the Oracle Cluster** chapter.
11. Once all the tests are configured, signout of the eG administrative interface.

Chapter 3: Monitoring the Oracle Cluster

eG Enterprise provides a dedicated Oracle Cluster model for monitoring the Oracle RAC. This is an 'agentless' model that requires the eG agent to be installed on any remote Windows host in the environment . This agent should be configured to access the cluster via the VIP or SCAN IP of any node in the cluster, so that it can report the availability of the cluster service, how load is balanced across the nodes of the service, the wait events that occur on each instance of the service, tablespace usage, lock behavior, and more!

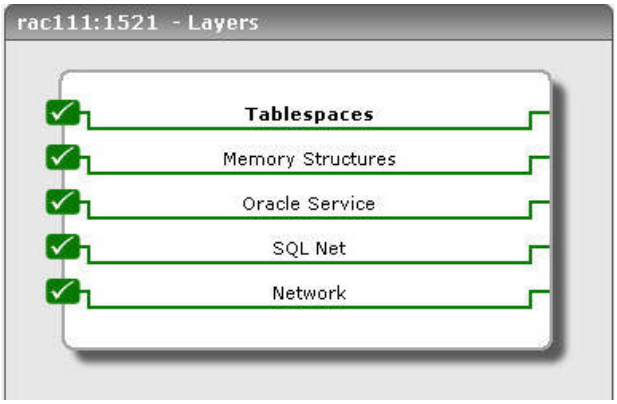


Figure 3.1: The layer model of the Oracle Cluster service

Each layer of Figure 3.1 above is mapped to tests that report a wealth of metrics that will help cluster administrators find quick and accurate answers for the following performance queries:

- Is the cluster service available? If so, how quickly does it respond to user requests?
- Is session load uniformly balanced across all the instances of the Oracle RAC?
- Are too many wait events occurring on any instance? What type of wait events are these - checkpoint events, log file syn waits, log file parallel write events, Db file parallel write events, or Db file sequential read events?
- Is any session/session module on an instance frequently waiting for CPU, cluster resources, I/O, locks, or latches? If so, which module/session is this and which instance is it associated with?
- Has the eG agent captured on a session/session module any wait event that should typically not occur on any system?
- What is the target and estimated MTTR (Mean Time to Recovery) of each of the instances managed by the RAC?

- Is the estimate of recovery I/O and the redo blocks that must be processed by an instance during recovery too high? Will it affect MTTR?
- Are the redo log files adequately sized to avoid unnecessary checkpointing?
- Is the checkpoint auto-tuning mechanism functioning effectively?
- Is any instance unncessarily holding many transaction locks for long periods of time?
- Is any tablespace experiencing a space drain? What type of objects (tables, indexes, partitions, LOB segments, etc.) are consuming too much space in this tablespace?
- Are temporary tablespaces adequately sized?
- Is the undo tablespace of any instance taking too long to execute queries? Is it because one/more queries are inefficient? Which queries are these?

How long does an instance take to perform undo retention? Were any bottlenecks detected in this process?

The sections that follow will discuss each of the top 3 layers of Figure 3.1 elaborately, as the remaining layers have been dealt with in the *Monitoring Oracle Database Server* document.

3.1 The Oracle Service Layer

Besides monitoring the session load on the cluster service and pinpointing issues in load-balancing across cluster instances, this layer monitors the following:

- The availability and responsiveness of the cluster service;
- The number and nature of wait events that occur on each instance of the cluster, and the sessions/session modules affected by these waits;
- The target and estimated MTTR (Mean Time to Recovery) for every instance;
- Accesses to the undo tablespace of an instance.

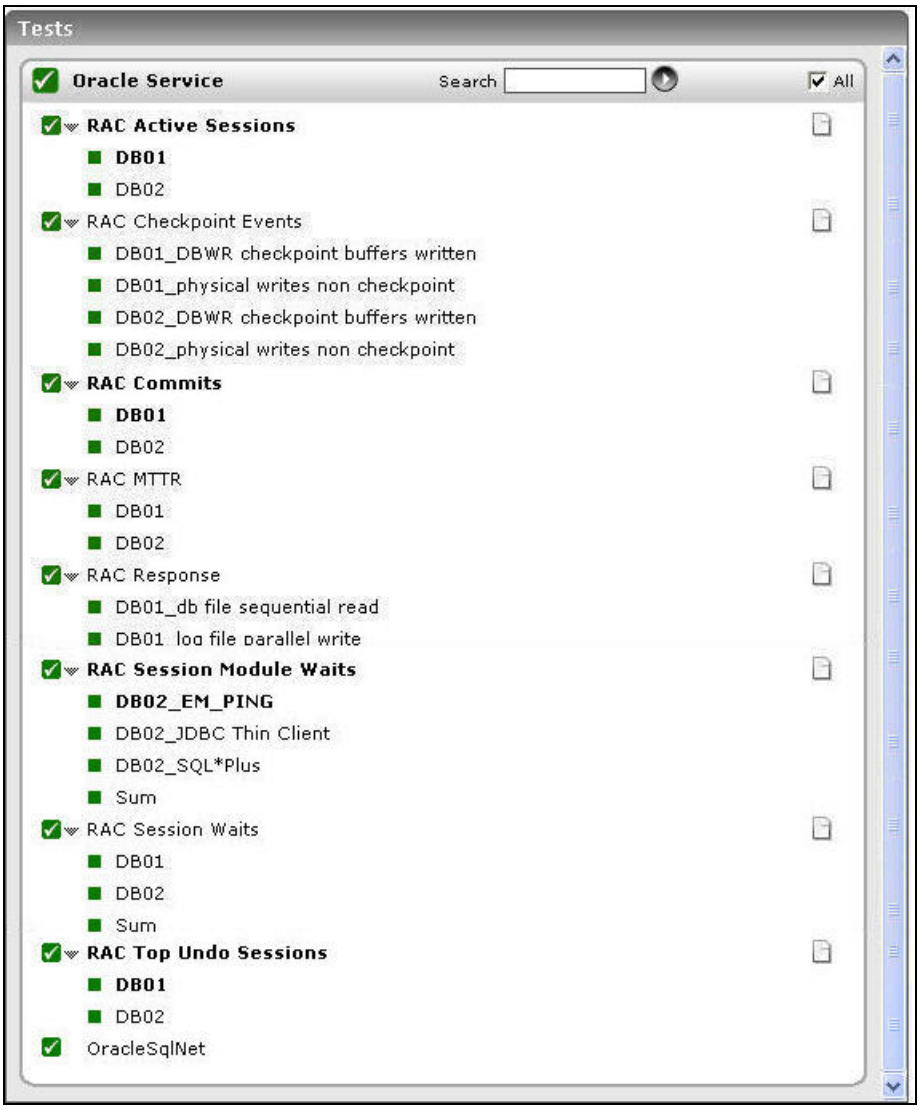


Figure 3.2: The Oracle Service Layer

3.1.1 Oracle RAC Active Sessions Test

This test reports the number of sessions currently active on each database in the Oracle RAC, and thus points you to overloaded instances.

Target of the test : Oracle RAC

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance monitored.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of sessions:	Indicates the number of sessions currently active on this instance.	Number	This measure is a good indicator of the session load on an instance. Comparing the value of this measure across instances will help you identify the overloaded

Measurement	Description	Measurement Unit	Interpretation
			instance and also shed light on load-balancing irregularities. You can use the detailed diagnosis of this measure to view the details of the active sessions. With the help of these details, you can figure out whether a session has remained open for too long a time due to prolonged wait events.

3.1.2 Oracle RAC Checkpoint Events Test

The checkpoint process is responsible for updating file headers in the database datafiles. A checkpoint occurs when Oracle moves new or updated blocks (called dirty blocks) from the RAM buffer cache to the database datafiles. A checkpoint keeps the database buffer cache and the database datafiles synchronized. This synchronization is part of the mechanism that Oracle uses to ensure that your database can always be recovered.

Check-pointing is an important Oracle activity which records the highest system change number (SCN), so that all data blocks less than or equal to the SCN are known to be written out to the data files. If there is a failure and then subsequent cache recovery, only the redo records containing changes at SCN(s) higher than the checkpoint need to be applied during recovery.

Key checkpoint-related activities may generate wait events. For instance, SQL statements may have to wait for processing until the DBWR (database writer) finishes writing dirty blocks in the buffer cache to the datafiles. If too many such wait events occur on an instance, it may cause the performance of the Oracle cluster to deteriorate. It is hence essential to keep close tabs on the checkpoint-related wait events and the activity responsible for them.

The **RAC Checkpoint Events** test auto-discovers the wait event types related to the checkpoint process, and reports the number of events of each type that have occured in each instance of an Oracle RAC.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the *Oracle RAC* as desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS**

list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle RAC

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every wait event type discovered on each instance of the monitored RAC

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Event Count:	Indicates the number of wait events of this type that have occurred on this instance during the last measurement period.	Number	Ideally, the value of this measure should be low. A consistent increase in this value is a cause of concern, as it indicates that a checkpoint- related activity is not getting completed, resulting in the generation of numerous wait events and degrading the overall performance of the Oracle RAC. Compare the value of this measure across the event types to determine which type of wait event has occurred most frequently on an instance.

3.1.3 Oracle RAC Commits Test

A wait class is a grouping of wait events, and every wait event belongs to a wait class. The main wait classes of the Oracle database server are:

- Administrative
- Application
- Cluster
- Commit
- Concurrency
- idle
- Network
- Other
- Scheduler

- System I/O
- User I/O

The Commit wait class comprises of only one wait event - wait for redo log write confirmation after a commit (that is, 'log file sync' event). Commit is not complete until LGWR (log writer) writes log buffers including commit redo records to log files. In a nutshell, after posting LGWR to write, user or background processes waits for LGWR to signal back with 1 sec timeout. User process charges this wait time as 'log file sync' event.

This test reports the number of sessions to each instance, which are waiting for a redo log write confirmation after a commit. This way, the test sheds light on the open sessions to a instance, and the reason for the sessions remaining.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the *Oracle RAC* as desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle RAC

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every instance in the monitored Oracle RAC.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive

server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of Sessions:	Indicates the number of sessions to this instance that are waiting for redo log confirmation after commit.	Number	A steady increase in the value of this measure is a cause of concern, as it indicates the following: a. Many sessions are forced to stay open owing to the commit wait events, and this may cause a session overload; b. Many "log file sync" wait events are occuring, causing the performance of the Oracle RAC to deteriorate. The root cause for 'log file sync' waits are as follows: • LGWR is unable to complete writes fast enough - this could be because, the disk I/O performance to log files is not good enough or, the LGWR is

Measurement	Description	Measurement Unit	Interpretation
			starving for CPU resources or, the LGWR paged out due to memory starvation issues or, due to file system or unix buffer cache limitations • LGWR is unable to post the processes fast enough, due to excessive commits. • IMU undo/redo threads • LGWR is suffering from other database contention such as enqueue waits or latch contention • Various bugs Use the detailed diagnosis of this measure to view the details of the sessions affected by log file sync waits.

3.1.4 Oracle RAC MTTR Test

Instance recovery, which is the process of recovering the redo thread from the failed instance, is a critical component affecting availability. When using Oracle RAC, the SMON process in one surviving instance performs instance recovery of the failed instance. The sooner this happens and lesser the I/O that is consumed during recovery, the better will be the user experience with the Oracle RAC.

Mean time to recovery (MTTR) is the average time that the Oracle server will take to recover from any failure. In order to limit recovery I/O and optimize cluster performance, you need to understand the MTTR target your system is currently achieving and what your potential MTTR target could be, given the I/O capacity. This test provides you with this understanding by reporting the target and estimated MTTR, and by monitoring the key factors affecting MTTR such as the redo log size and the number of redo blocks to be processed.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the *Oracle RAC* as desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for every instance in the monitored Oracle RAC.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'
6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg ;

create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Target MTTR:	Indicates the effective mean time to recover (MTTR) this instance.	Secs	Usually, the value of this measure should be equal to the value of the *FAST_ START_ MTTR_ TARGET* initialization parameter. *FAST_ START_ MTTR_ TARGET* specifies a target for the expected

Measurement	Description	Measurement Unit	Interpretation
			mean time to recover (MTTR), that is, the time (in seconds) that it should take to start up the instance and perform cache recovery. After *FAST_ START_ MTTR_ TARGET* is set, the database manages incremental checkpoint writes in an attempt to meet that target. If *FAST_ START_ MTTR_ TARGET* is set to such a small value that it is impossible to do a recovery within its time frame, then the the value of this measure will be larger than *FAST_ START_ MTTR_ TARGET* . If *FAST_ START_MTTR_TARGET* is set to such a high value that even in the worst- case (the whole buffer cache is dirty) recovery would not take that long, then the value of this measure will be the same as the Estimated MTTR. If *FAST_ START_ MTTR_ TARGET* is not specified, then again, the value of this measure will be the same as the value of the Estimated MTTR measure.
Estimated MTTR:	Indicates the current estimated mean time to recover (MTTR).	Secs	This measure is calculated based on the number of dirty buffers and log blocks (0 if *FAST_ START_ MTTR_ TARGET* is not specified). Basically, this value tells you how

Measurement	Description	Measurement Unit	Interpretation
			long you could expect recovery of the instance to take place based on the work your system is doing at the time of testing. This measure reports the estimated mean time to recovery based on the current state of the running database. If the database has just opened, the system may contain only a few dirty buffers, so not much cache recovery would be required if the instance failed at this moment. That is why the value of this measure can, for the moment, be lower than the minimum possible Target MTTR.
Recovery Estimated IOs:	Indicates the estimated number of dirty buffers in the buffer cache of this instance.	Number	
Target Redo blocks:	Indicates the target number of redo blocks that must be processed while recovering this instance.	Number	Instance recovery is nothing more than using the contents of the online log files to rebuild the database buffer cache to the state it was in before the crash. This will replay all changes extracted from the redo logs that refer to blocks that had not been written to disk at the time of the crash. Though instance recovery guarantees no corruption, it may take a considerable time to do its roll forward before the database can be opened. This time is dependent

Measurement	Description	Measurement Unit	Interpretation
			on two factors: how much redo has to be read and how many read/write operations will be needed on the datafiles as the redo is applied. The values of these measures serve as good indicators of the amount of redo reading work that needs to be performed as part of the recovery process, and are hence useful while determining the MTTR.
Actual Redo blocks:	Indicates the actual number of redo blocks that are required by this Oracle instance to recover.	Number	
Writes logfile size:	Indicates the number of writes driven by the smallest redo log file size for each oracle instance.	Number	This measure is used to drive the checkpoint process, if your redo log file size is under sized. Since the *FAST_ START_ MTTR_ TARGET* parameter is set to limit the instance recovery time, Oracle automatically tries to checkpoint as frequently as necessary. Under such a condition, the size of the log files should be large enough to avoid additional checkpoint due to under sized log files.
Writes auto tune:	Indicates the number of writes due to auto-tune checkpointing.	Number	The checkpoint auto- tuning mechanism inspects statistics on machine utilization, such as the rate of disk I/O and CPU usage, and if it appears that there is spare capacity, it will use this capacity to write out additional dirty buffers from the database buffer cache, thus pushing the checkpoint position forward. The result is that even if the *FAST_START_MTTR_ TARGET* parameter is set to a

Measurement	Description	Measurement Unit	Interpretation
			high value (the highest possible is 3600 seconds— anything above that will be rounded down), actual recovery time may well be much less. Enabling checkpoint auto- tuning with a high target should result in your instance always having the fastest possible recovery time that is consistent with maximum performance.

3.1.5 Oracle RAC Waits Response Test

This test reports the key performance statistics pertaining to the following wait events in each Oracle instance:

- **log file parallel write**: This event occurs when writing redo records to the redo log files from the log buffer. Writing redo records to the redo log files from the log buffer.
- **Db file parallel write**: This event occurs in the DBWR. It indicates that the DBWR is performing a parallel write to files and blocks. When the last I/O has gone to disk, the wait ends.
- **log file sync**: When a user session commits, the session's redo information needs to be flushed to the redo logfile. The user session will post the LGWR to write the log buffer to the redo log file. When the LGWR has finished writing, it will post the user session.
- **Db file sequential read**: The session waits while a sequential read from the database is performed. This event is also used for rebuilding the control file, dumping datafile headers, and getting the database file headers.

Effective wait analysis helps determine on which wait event the instance spends most of its time, and which current connections are responsible for the above-mentioned wait events.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for every wait event type captured on every instance in the monitored Oracle RAC.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total waits:	Indicates the total number of times this wait event has occurred since	Number	If the value of this measure is very high, then you can drill down

Measurement	Description	Measurement Unit	Interpretation
	the last measurement period.		further using the detailed diagnosis capability (if enabled) of the eG Enterprise suite to figure out which current connections may be responsible for this. The detailed diagnosis of this measure reveals the session IDs of the sessions that caused the wait events to occur, the users who initiated the sessions, and the total number of waits, wait time, and the maximum wait time for every session.
Time waited:	Indicates the total time for which the events of this type were in existence on this instance.	Seconds	Ideally, the value of this measure should be low.
Average wait time:	Indicates the average duration of time in which this wait event was persistent since the last measurement period.	Seconds	Ideally, the value of this measure should be low. A very high value or a consistent increase in this value is indicative of a problem condition which requires further investigation. Use the detailed diagnosis capability to zoom into the session that has contributed to the abnormal increase in wait time.

3.1.6 Oracle RAC Session Module Waits Test

For each session module on a monitored instance, this test reports the number and nature of wait events that occurred during the last measurement period. In addition, the test also reports the total number of events (of a type) that occurred across all modules and instances. With the help of these metrics, administrators can figure out how much time an instance has spent waiting and what it was waiting for; a high value is a cause for concern, as it indicates that the instance has waited too long, and could have consequently suffered significant processing delays.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the *Oracle RAC* as desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for every session module on each instance of the monitored Oracle RAC.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg ;

create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cpu waits:	Indicates the number of times since the last measurement period this session module on this instance has waited for CPU.	Number	Ideally, the value of this measure should be low.
User IO waits:	Indicates the total number of waits that have occurred in this session module on this instance for user I/O, since the last measurement period.	Number	Some of the user I/O wait events include: BFILE read, buffer read retry, db file parallel read, db file scattered read, db file sequential read, and db file single write. Ideally, the value of this measure should be low.
Cluster waits:	Indicates the total number of waits pertaining to cluster resources that have occurred in this session module on this instance since the last measurement period.	Number	A low value is desired for this measure.
Committed waits:	Indicates the total number of commit waits that have occurred in this session module on this instance since the last measurement period.	Number	A low value is desired for this measure. If this measure shows a high value, refer to the detailed diagnosis capability of this measure to identify the sessions that are affected by committed waits.
Transaction lock waits:	Indicates the number of times this session module on this instance	Number	A low value is desired for this measure. If this measure shows a high value, use the detailed

Measurement	Description	Measurement Unit	Interpretation
	waited for transaction locks, during the last measurement period.		diagnosis of the RAC Transaction Locks test to obtain the detailed report on all the sessions that are affected by the lock waits.
Latch waits:	Indicates the total number of latch waits that have occurred in this session module since the last measurement period.	Number	A low value is desired for this measure.
Other waits:	Indicates the total number of times since the last measurement period waits that should typically not occur on a system (eg., 'wait for EMON to spawn') occurred in this session module on this instance.	Number	Ideally, the value of this measure should be low.

3.1.7 Oracle RAC Session Waits Test

This test reports the number and nature of session wait events that occurred on each instance of the monitored RAC during the last measurement period. In addition, the test also reports the total number of events (of a type) that occurred across all instances. With the help of these metrics, administrators can figure out how much time an instance has spent waiting and what it was waiting for; a high value is a cause for concern, as it indicates that the instance has waited too long, and could have consequently suffered significant slowdowns.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the Oracle Cluster as the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each instance of the monitored Oracle RAC.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'
6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cpu waits:	Indicates the number of times since the last measurement period the	Number	Ideally, the value of this measure should be low.

Measurement	Description	Measurement Unit	Interpretation
	sessions on this instance waited for CPU.		
IO waits:	Indicates the total number of times the sessions on this instance waited for user I/O, since the last measurement period.	Number	Some of the user I/O wait events include: BFILE read, buffer read retry, db file parallel read, db file scattered read, db file sequential read, and db file single write. Ideally, the value of this measure should be low.
Cluster waits:	Indicates the total number of times the sessions on this instance waited for cluster resources since the last measurement period.	Number	A low value is desired for this measure.
Committed waits:	Indicates the total number of commit waits that have occurred in the sessions of this instance since the last measurement period.	Number	A low value is desired for this measure. If this measure shows a high value, refer to the detailed diagnosis capability of this measure to identify the sessions that are affected by committed waits.
Transaction lock waits:	Indicates the number of times the sessions on this instance waited for transaction locks, during the last measurement period.	Number	A low value is desired for this measure. If this measure shows a high value, use the detailed diagnosis of the RAC Transaction Locks test to obtain the detailed report on all the sessions that are affected by the lock waits.
Latch waits:	Indicates the total number of latch waits that have occurred in the sessions of this instance	Number	A low value is desired for this measure.

Measurement	Description	Measurement Unit	Interpretation
	since the last measurement period.		
Configuration waits:	Indicates the total number of configuration wait events that occurred in the sessions of this instance during the last measurement period.	Number	Configuration waits are waits caused by inadequate configuration of database or instance resources (for example, undersized log file sizes, shared pool size). Ideally, the value of this measure should be low.
System IO waits:	Indicates the total number of system I/O wait events that occurred in the sessions of this instance during the last measurement period.	Number	System I/O waits are waits for background process I/O (for example, DBWR wait for 'db file parallel write'). Ideally, the value of this measure should be low.
Network waits:	Indicates the total number of times since the last measurement period waits related to network messaging (for example, 'SQL*Net more data to dblink', occurred in the sessions of this instance.	Number	A low value is desired for this measure.
Other waits:	Indicates the total number of times since the last measurement period waits that should typically not occur on a system (eg., 'wait for EMON to spawn') occurred in the sessions	Number	Ideally, the value of this measure should be low.

Measurement	Description	Measurement Unit	Interpretation
	of this instance.		
Subtotal waits:	Indicates the overall number of waits that have occurred since the last measurement period in the sessions of this Oracle instance.	Number	Ideally, the value of this measure should be low.

3.1.8 Oracle RAC Top Undo Sessions Test

Every Oracle Database must have a method of maintaining information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo.

Undo records are used to:

- Roll back transactions when a ROLLBACK statement is issued
- Recover the database
- Provide read consistency
- Analyze data as of an earlier point in time by using Oracle Flashback Query
- Recover from logical corruptions using Oracle Flashback features

When a **ROLLBACK** statement is issued, undo records are used to undo changes that were made to the database by the uncommitted transaction. During database recovery, undo records are used to undo any uncommitted changes applied from the redo log to the datafiles. Undo records provide read consistency by maintaining the before image of the data for users who are accessing the data at the same time that another user is changing it.

Oracle provides a fully automated mechanism, referred to as automatic undo management, for managing undo information and space. In this management mode, you create an undo tablespace, and the server automatically manages undo segments and space among the various active sessions.

Each instance in the RAC system can only use one undo tablespace at a time. In other words, instances cannot share undo tablespaces. Each instance in the cluster, being an independent transaction-processing environment, maintains its own UNDO area for undo management. The RAC

system allows the creation and use of several undo tablespaces. When the instance is started, it uses the first available undo tablespace. A second instance will use another undo tablespace. Thus, each instance in a RAC system will have exclusive access to a particular undo tablespace at a given time. The undo tablespace cannot be shared among the instances at the same time. Only once an undo tablespace is released by an instance, it can be assigned to another instance. However, all instances can read blocks from any or all undo tablespaces for the purpose of constructing read-consistency images.

You need to closely observe how the sessions to each RAC instance use the undo tablespaces; this will enable you to proactively detect unusually high/long usage conditions. The **RAC Top Undo Sessions** test brings such anomalies to light. This test reports the number of sessions (per instance) accessing the undo tablespace and the duration of usage of these sessions, thus indicating excessive usage (if any) of the undo tablespace. The detailed diagnosis capability of the test turns the spotlight on those sessions that are the leading users of the undo tablespace, and provides pointers to the query executed by these sessions. With the help of this information you can identify inefficient queries and fine-tune them, so that potential processing delays and consequent instance slowdowns/crashes can be averted.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each instance of the monitored Oracle RAC.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Average duration:	Indicates the average time taken by the sessions to this instance, to execute queries on the undo tablespace.	Secs	Ideally, the value of this measure should be low. An unusually high value for this measure could indicate that one/more sessions are using the undo tablespace for too long a time. Use the detailed diagnosis of this measure to identify the top sessions in terms of duration of usage of the undo tablespace, and determine the SQL ID of the query executed by each session on that tablespace. Inefficient queries can thus be isolated. Fine-tuning these queries will enable the optimal usage of the undo tablespace. Query processing will also be delayed if the undo tablespace is improperly sized. If the undo tablespace has insufficient space, many transactions to the

Measurement	Description	Measurement Unit	Interpretation
			tablespace may terminate before completion, and many more transactions may even hang; this will result in a long line of long-running queries. Also, since Oracle automatically tunes the undo retention period based on undo tablespace size and system activity, if the undo tablespace runs out of space, it will grossly affect the auto retention capability of Oracle, once again causing query failures. When available space for new transactions becomes short, the database begins to overwrite expired undo. If the undo tablespace has no space for new transactions after all expired undo is overwritten, the database may begin overwriting unexpired undo information. If any of this overwritten undo information is required for consistent read in a current long- running query, the query could fail with the snapshot too old error message.
Number of sessions:	Indicates the number of sessions to this instance that are utilizing the undo tablespace.	Number	This serves as a good indicator of the load on the undo tablespace.

3.1.9 Oracle RAC SQL Network Test

This test executes an external test that emulates a query to the cluster to determine its availability and responsiveness. The test sends the emulated request to the virtual cluster server (i.e., the Oracle Cluster), which will promptly forward the request to that node in the cluster that currently owns the cluster server. If at least one node in the cluster is currently active, then the query will successfully execute on that node, and report the good health of the cluster. On the other hand, if none of the nodes in the cluster are active, then the query will be unable to execute, and the test will hence report the non-availability of the cluster.

Target of the test : Oracle RAC

Agent deploying the test : An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Oracle cluster is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Oracle cluster is listening.

Outputs of the test : One set of results for every SID (instance) monitored..

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **TIMEOUT** - Specify the duration (in seconds) beyond which the test will timeout if no response is received from the server. The default value is 30 seconds.
10. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

11. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Oracle server availability:	Whether the cluster is responding to requests.	Percent	The availability is 100% when the cluster is responding to a request and 0% when it is not. Availability problems may be caused if none of the nodes of the cluster are currently operational.
Total response time:	The time taken by the cluster to respond to a user query.	Secs	A sudden increase in response time is indicative of a bottleneck at the cluster.

3.1.10 Oracle RAC Cluster Interconnects Test

A cluster database comprises two or more nodes that are linked by an interconnect. The interconnect serves as the communication path between the nodes in the cluster database. Each Oracle instance uses the interconnect for the messaging that synchronizes each instance's use of shared resources. Oracle also uses the interconnect to transmit data blocks that the multiple instances share.

The non-availability of the interconnect on any cluster node can impair that node's communication with other nodes in the cluster. As a result, fail-over operations will be hampered and the cluster service will be forced to distribute session/request load across the remaining clusters in the node;

this in turn may overload the other nodes in the cluster. In the aftermath of this, mission-critical business services using the clustered resources may experience prolonged outages or slowdowns, resulting in considerable loss of revenue and reputation.

To avoid this, administrators need to continuously monitor the availability of the cluster interconnect on each node, analyze how session/process load is distributed across the nodes via the interconnect, and proactively detect the following:

- The sudden unavailability of the interconnect on a node;
- How the unavailability of an interconnect affects the load on the other nodes in the cluster;

For this purpose, you can use the **Oracle Cluster Interconnects** test. This test periodically verifies whether the nodes in the cluster are able to communicate via the cluster interconnect, and promptly reports the non-availability of the interconnect. In addition, the test also keeps tabs on the session and process load on each node in the cluster, thus promptly revealing the impact of the unavailability of a cluster interconnect on the load and performance of other nodes in the cluster.

Target of the test : Oracle RAC

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *clusternodeID_ <IP_ address_ used_ for_ internode_communication>* in the Oracle cluster.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

10. **HIDE IP** – This test reports a set of metrics for that IP address on each cluster node using which a node communicates with other nodes in the cluster. The descriptors of this test therefore will be of the following format by default: *clusternodeID_ <IP_ used_ for_ internode_ communication*>. Accordingly, the **HIDE IP** parameter is set to **No** by default. High security environments however may not want to expose the IP address that cluster nodes use for internal communication. In such environments, you can set the **HIDE IP** flag to **No**, so that the descriptors of this test do not include the <*IP_used_for_internode_communication*>. In such cases therefore, only the *clusternodeID* will be displayed as the descriptors of this test.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cluster interconnect percentage:	Indicates whether the cluster interconnect isavailable on this node or not.	Percent	The value 0 for this measure indicates that this node is unable to communicate with other nodes in the cluster via the cluster interconnect. The value 100 indicates that the interconnect is available and is enabling this node to communicate with the other cluster nodes.
Logon rate :	Indicates the rate at which user logons occurred on this node.	Logons/Sec	
Processes utilization :	Indicates the number of processes currently running on this cluster node.	Number	As long as the value of this measure is much lower than the value of the processes setting in the database parameter file, the node will be able to handle the process load.
Processes utilization percentage :	Of the maximum number of processes this node can handle, what percentage is currently active on this cluster node.	Percent	Ideally, the value of this measure should be low. If this measure value is close to 100%, it could mean that the node is about to exhaust its processing limit and

Measurement	Description	Measurement Unit	Interpretation
			may not be able to handle any more processes. On the other hand, if the value of this measure is consistently high for a cluster node, then check the processes setting in the database parameter file to figure out whether/not the node has been configured with adequate processing capability. If this check reveals that the node has been configured with a limited number of processes than it can handle, you may want to increase the processes setting to suit the node's capacity.
Session utilization :	Indicates the number of sessions that are currently active on this node.	Number	As long as the value of this measure is much lower than the value of the sessions setting in the database parameter file, the node will be able to handle the session load. If the value of this measure is unusually high for any cluster node, then compare the value of this measure across nodes to figure out whether/not load is uniformly distributed across all cluster nodes. If session load on most of the cluster nodes is high, then the sudden increase in session load could be attributed to an unavailable cluster interconnect. Because of the unavailability, the cluster service may not have been unable to contact the affected cluster node and may have been

Measurement	Description	Measurement Unit	Interpretation
			compelled to distribute the load amongst the remaining cluster nodes. This may have caused load on the other nodes to suddenly increase. To confirm this, check the value of the Interconnect availability percentage measure of all nodes. On the other hand, if no interconnect is unavailable, and if Session utilization is abnormally high on a particular node only, it could mean that that node is indeed overloaded.
Session utilization percentage :	Of the maximum number of sessions this node can handle, what percentage is currently active on this cluster node.	Percent	Ideally, the value of this measure should be low. If this measure value is close to 100%, it could mean that the node may not be able to handle any more sessions. On the other hand, if the value of this measure is consistently high for a cluster node, then check the sessions setting in the database parameter file to figure out whether/not the node has been configured with adequate session-handling capability. If this check reveals that the node has been configured with a limited number of sessions than it can handle, you may want to increase the sessions setting to suit the node's true capacity.

3.1.11 Oracle RAC CR Block Requests Test

Data blocks requested from the Global Cache are of two types: current and consistent-read (CR) blocks. When you update data in the database, Oracle Database must locate the most recent version of the data block that contains the data, which is called the current block. If you perform a query, only data committed before the query began is visible to the query. Data blocks that were changed after the start of the query are reconstructed from data in the undo segments, and the reconstructed data is made available to the query in the form of a consistent-read block.

Whenever a session requests for a CR block, Oracle first checks whether it has that block in its local cache. If the block does not exist in the local cache but is available in the remote cache, then it is transferred from the remote to local cache via the interconnect. The time that elapses between when a CR block is requested and when the session receives it should be tracked continuously, so that global cache block access latencies (if any) are detected proactively and resolved promptly. Use the **Oracle CR Block Requests** test to perform this tracking.

This test, at configured intervals, monitors requests for CR blocks and reports how long it took for the requests to be serviced by the buffer caches. This sheds light on request processing delays (if any).

Target of the test : Oracle RAC

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle cluster.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive

server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Average cr block request time:	Indicates the time taken to service requests for cr blocks.	Centisecs	Ideally the value of this measure should be low. A high value for this measure is indicative of high latencies while accessing global cache blocks. This can be caused by any of the following: • A high number of requests caused by SQL statements that are not tuned. • A large number of processes in the queue waiting for CPU, or scheduling delays. • Slow, busy, or faulty interconnects. In these cases, check your network connection for dropped packets,

Measurement	Description	Measurement Unit	Interpretation
			retransmittals, or cyclic redundancy check (CRC) errors. When global cache requests cause a performance problem, optimizing SQL plans and the schema to improve the rate at which data blocks are located in the local buffer cache, and minimizing I/O is a successful strategy for performance tuning.

3.1.12 Oracle RAC Current Block Requests Test

Data blocks requested from the Global Cache are of two types: current and consistent-read (CR) blocks. When you update data in the database, Oracle Database must locate the most recent version of the data block that contains the data, which is called the current block. If you perform a query, only data committed before the query began is visible to the query. Data blocks that were changed after the start of the query are reconstructed from data in the undo segments, and the reconstructed data is made available to the query in the form of a consistent-read block.

Whenever a session requests for a current block, Oracle first checks whether the block is present in the local cache. If not, it looks for the same in the remote cache. If the block is available in the remote cache,it pins the block in the exclusive mode, flushes it from the cache, and sends it across over the interconnect. The time that elapses between when a current block is requested and when the session receives it should be tracked continuously, so that global cache block access latencies (if any) are detected proactively and resolved promptly. Use the **Oracle Current Block Requests** test to perform this tracking.

This test, at configured intervals, monitors requests for current blocks and reports how long it took for the requests to be serviced by the buffer caches. This sheds light on request processing delays (if any).

Target of the test : Oracle RAC

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle cluster.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'
6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Average current block request time:	Indicates the time taken to service requests for current blocks.	Centisecs	Ideally the value of this measure should be low. A high value for this measure is indicative of high latencies while accessing the global cache. This

Measurement	Description	Measurement Unit	Interpretation
			can be caused by any of the following: • A high number of requests caused by SQL statements that are not tuned. • A large number of processes in the queue waiting for CPU, or scheduling delays. • Slow, busy, or faulty interconnects. In these cases, check your network connection for dropped packets, retransmittals, or cyclic redundancy check (CRC) errors. When global cache requests cause a performance problem, optimizing SQL plans and the schema to improve the rate at which data blocks are located in the local buffer cache, and minimizing I/O is a successful strategy for performance tuning.

3.1.13 Oracle RAC Global Cache Corrupt Blocks Test

This test reports the number of blocks that were corrupted while being transferred between Oracle instances through the private interconnect in this Oracle RAC.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle cluster.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'
6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Block corrupted count:	Indicates the number of blocks that were corrupted during transfer through the private interconnect.	Number	Ideally the value of this measure should be zero. A high value indicates the possibility of an IPC, network or hardware problem.

3.1.14 Oracle RAC Global Cache Lost Blocks Test

This test reports the number of blocks that were lost during transfer from one Oracle instance to another through private interconnects in this Oracle RAC.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the *Oracle RAC* as desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Oracle cluster.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so,

ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg ;

create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;***

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Block lost count:		Number	Ideally the value of this measure

Measurement	Description	Measurement Unit	Interpretation
	Indicates the number of blocks that were lost during transfer from one Oracle instance to another.		should be zero. A high value is indicative of network problems. The use of an unreliable IPC protocol such as UDP may result in the value for global cache blocks lost being non- zero. This ratio should be as small as possible. Many times, a non-zero value for global cache blocks lost does not indicate a problem because Oracle will retry the block transfer operation until it is successful.

3.1.15 Oracle RAC Scans Test

Full table scans on a database instance can degrade the performance of the database. This test monitors the extent of full table scans happening on each database in the Oracle RAC.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the *Oracle RAC* as desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on an Oracle cluster.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.

5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Percent long table scans:	Indicates the percentage of long table scans happening on this database.	Percent	Ideally, this value should be lower than 10%. If more than 20% of scans are happening on long tables, the database/accesses to the database may need to be tuned. Full table scans may happen due to several reasons. For instance, the indexes of a table may not be used properly in queries. By tuning the queries, the full table scans can be reduced and the database performance significantly improved.

Measurement	Description	Measurement Unit	Interpretation
Long table scans:	Indicates the number of long table scans that happened on each database in this Oracle RAC during the last measurement period.	Number	
Short table scans:	Indicates the number of short table scans that happened on each database in this Oracle RAC during the last measurement period.	Number	

3.1.16 Oracle RAC Cluster Nodes Test

This test reports the count of nodes in the Oracle cluster and indicates the number and names of those nodes that are currently accessible. This way, the nodes that are inaccessible/unavailable can be identified. The test also reports the percentage of available nodes, and thus indicates if only very few nodes in the cluster are able to service the client requests to the cluster. This signals a potential overload.

Target of the test : Oracle RAC

Agent deploying the test : An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Oracle cluster is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Oracle cluster is listening.

Outputs of the test : One set of results for every SID (instance) monitored..

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.

4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'
6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. **TIMEOUT** - Specify the duration (in seconds) beyond which the test will timeout if no response is received from the server. The default value is 30 seconds.
11. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total nodes:	Indicates the number of nodes in the cluster.	Number	
Available nodes:	Indicates the number of nodes in the cluster that are currently accessible.	Number	Use the detailed diagnosis of this measure to know which nodes in the cluster are currently accessible.
Unavailable nodes:	Indicates the number of nodes in the cluster that are currently	Number	

Measurement	Description	Measurement Unit	Interpretation
	unavailable.		
Percentage of available nodes:	Indicates the percentage of nodes in the cluster that are available.	Percent	A high value is desired for this measure.

3.2 The Memory Structures Layer

A transaction lock is acquired when a transaction initiates its first change and is held until the transaction does a COMMIT or ROLLBACK. It is used mainly as a queuing mechanism so that other sessions can wait for the transaction to complete.

The test mapped to this layer reports the number and duration of transaction locks held by each instance of the cluster.

Figure 3.3: The tests mapped to the Memory Structures layer

3.2.1 Oracle RAC Transaction Locks Test

A transaction lock held for too long a time will prevent other sessions from accessing the database object, thereby stalling critical database operations. It is hence imperative to monitor the transaction locks to each database instance in an Oracle RAC. Using the **RAC Transaction Locks** test, you can determine the number of transaction locks held by each database instance and the duration of these locks, so that you can quickly identify the instance holding a large number of transaction locks and that which is holding locks for an unreasonably long time.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each instance of the monitored Oracle RAC.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'
6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG

monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of Locks:	Indicates the number of locks held by this instance.	Number	A high value may indicate one of the following: • Too many transactions happening • Locked resources not being released properly • Locks are being held unnecessarily.
Average wait time:	Indicates the time for which the locks were held by this instance.	Seconds	A high value may indicate one of the following: • Too many transactions happening

Measurement	Description	Measurement Unit	Interpretation
			• Locked resources not being released properly • Locks are being held unnecessarily.

3.3 The Tablespaces Layer

The tests mapped to this layer proactively alert administrators to potential space constraints in the tablespaces and temporary tablespaces of the cluster, and reveals long-running queies to the undo tablespaces of the cluster.

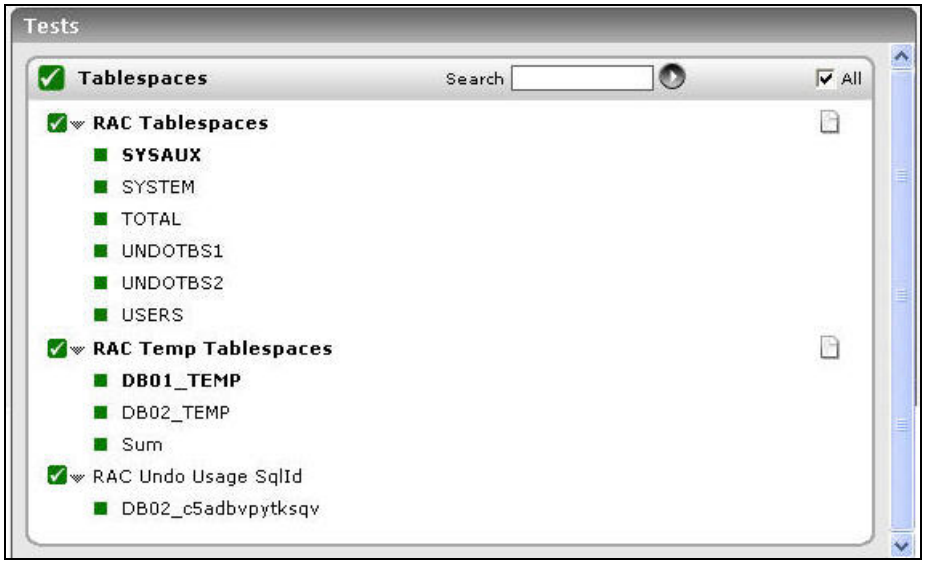


Figure 3.4: The tests mapped to the Tablespaces layer

3.3.1 Oracle RAC Tablespaces Test

Tablespaces should be adequately sized. If not, then the tablespaces may frequently run very low on free space, causing all statements that attempt to acquire new space in the tablespace to fail. This in turn will result in serious performance issues ranging from slowdowns to shutdowns. Continuous monitoring of tablespace size and usage is hence important.

The **RAC Tablespaces** test auto-discovers the tablespaces managed by the Oracle RAC, and reports how well every tablespace has been utilized. In the process, the test also reveals the type of database objects (tables, indexes, partitions, LOB segments, etc.) that are occupying space in the

tablespace. This way, you will be able to instantly identify the tablespace left with very little free space, and also zero-in on those objects that could be eroding space in that tablespace.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for each tablespace managed by the monitored Oracle RAC

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=<Oracle_service_name>;

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Table size:	Indicates the size of the tables present in this tablespace.	GB	A high value for these measures indicates that a tables and indexes consume a large chunk of space in the tablespace.
Index size	Indicates the space consumed by the indexes on the tables present in this tablespace.	GB	
Table partition size:	Indicates the size of all	GB	Partitioning addresses key issues in

Measurement	Description	Measurement Unit	Interpretation
	partitions of tables present in this tablespace.		supporting very large tables and indexes by letting you decompose them into smaller and more manageable pieces called partitions. Each partition of a table or index must have the same logical attributes, such as column names, datatypes, and constraints, but each partition can have separate physical attributes such as *pctfree, pctused*, and *tablespaces*. A high value for these measures indicates that table and index partitions consume a large chunk of space in the tablespace.
Index partiton size:	Indicates the size of all partitions of indexes present in this tablespace.	GB	
Lob segment size:	Indicates the size of LOB segments in this tablespace.	GB	LOBs (Large Object) are Oracle's data structures designed to store and retrieve large amounts of unstructured data such as video, audio, photo images, etc within the database. Whenever a table containing a LOB column is created, two segments are created to hold the specified LOB column. These segments are of type **LOBSEGMENT** and **LOBINDEX**. The **LOBINDEX** segment is used to access LOB chunks/pages stored in the LOBSEGMENT segment. The values of these measures report the size of the **LOBSEGMENT** sements and the **LOBINDEX** segments (respectively). A high value for these measures, quiet naturally,

Measurement	Description	Measurement Unit	Interpretation
Lob index size:	Indicates the size of LOB indexes in this tablespace.	GB	indicates that too many large objects are stored in the tablespace.
Lob partition size:	Indicates the size of table partitions with LOBs in this tablespace.	GB	You can partition tables with LOBs. As a result, LOBs can take advantage of all of the benefits of partitioning. For example, LOB segments can be spread between several tablespaces to balance I/O load and to make backup and recovery more manageable. LOBs in a partitioned table also become easier to maintain. A high value for this measure indicates that table partitions containing LOB columns are consuming a large amount of space in this tablespace.
Maximum size:	Indicates the maximum extent upto which a tablespace can grow.	GB	
Used size:	Indicates the size upto which this tablespace has been utilized.	GB	If this value is very high, it indicates that the tablespace memory is almost full.
Free size:	Indicates the amount of unused space available in this tablespace.	GB	If this value is very low, then it indicates over- utilization of the tablespace.
Free space usage:	Indicates the space available for overall growth expressed as a ratio of Free size with	Percent	If this value is very low, then it indicates over- utilization of the tablespace. Also, if the value of this measure is below 80 %, then

Measurement	Description	Measurement Unit	Interpretation
	respect to the Maximum size of the tablespace.		sufficient space must be allocated to the tablespace.

3.3.2 Oracle RAC Temp Tablespaces Test

A temporary tablespace, contrary to what the name might indicate, does exist on a permanent basis as do other tablespaces, such as the System and Sysaux tablespaces. However the data in a temporary tablespace is of a temporary nature, which persists only for the length of a user session. Oracle uses temporary tablespaces as work areas for tasks such as sort operations for users and sorting during index creation. Oracle does not allow users to create objects in a temporary tablespace. By definintion, the temporary tablespace holds data only for the duration of the user's session, and the data can be shared by all users.

Sufficient free space should be available in the temporary tablespace, as critical operations such as sorting and execution of hash-intensive queries may otherwise fail. Periodically checking the space usage in the temporary tablespaces will provide you with early warning signals of potential space contentions. The **RAC Temp Tablespaces** test monitors the usage of the temporary tablespace in each instance of the Oracle RAC, and proactively reports which temporary tablespace is running dangerously low on free space. Moreover, the test also reports the usage of the temporary tablespace across instances.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the temporary tablespace of each instance managed by the monitored Oracle RAC

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any

available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used space:	Indicates the space used by this temporary tablespace.	MB	If this value is very high, then it indicates that the memory of this tablespace is almost full.
Free space:	Indicates the amount of unused space in this temporary tablespace.	MB	If this value is very low, then it indicates over- utilization of the tablespace.
Total space:	Indicates the total amount of space allocated for this temporary tablespace.	MB	
Max space:	Indicates the maximum extent upto which this temporary tablespace can grow.	MB	
Used percentage:	Indicates the percentage of space used in this temporary tablespace.	Percent	If this value is very high, then it indicates over- utilization of the tablespace.
Free percentage:	Indicates the space available for overall growth expressed as a ratio of Free space with respect to the Max space of this temporary tablespace. The formula	Percent	If this value is very low, it indicates that more space needs to be allotted to this tablespace to ensure that critical operations do not fail.

Measurement	Description	Measurement Unit	Interpretation
	used is: Free_space/Max_bytes*100		

3.3.3 Oracle RAC Undo Usage SqlId Test

Every Oracle Database must have a method of maintaining information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo. Undo records are used to:

- Roll back transactions when a *ROLLBACK* statement is issued
- Recover the database
- Provide read consistency
- Analyze data as of an earlier point in time by using Oracle Flashback Query
- Recover from logical corruptions using Oracle Flashback features

When a *ROLLBACK* statement is issued, undo records are used to undo changes that were made to the database by the uncommitted transaction. During database recovery, undo records are used to undo any uncommitted changes applied from the redo log to the datafiles. Undo records provide read consistency by maintaining the before image of the data for users who are accessing the data at the same time that another user is changing it.

Oracle provides a fully automated mechanism, referred to as automatic undo management, for managing undo information and space. In this management mode, you create an undo tablespace, and the server automatically manages undo segments and space among the various active sessions.

Each instance in the RAC system can only use one undo tablespace at a time. In other words, instances cannot share undo tablespaces. Each instance in the cluster, being an independent transaction-processing environment, maintains its own UNDO area for undo management. The RAC system allows the creation and use of several undo tablespaces. When the instance is started, it uses the first available undo tablespace. A second instance will use another undo tablespace. Thus, each instance in a RAC system will have exclusive access to a particular undo tablespace at a given time. The undo tablespace cannot be shared among the instances at the same time. Only once an undo tablespace is released by an instance, it can be assigned to another instance. However, all instances can read blocks from any or all undo tablespaces for the purpose of constructing read-consistency images.

If instances take too long a time to read from or write to a undo tablespace, it will unnecessarily delay the recovery/rollback process and sometimes even abnormally terminate it, causing serious data inconsistencies. To avoid such adversities, it is imperative that you monitor how long the undo tablespace takes to execute queries, and promptly detect latencies in query execution. The **Oracle RAC Undo Usage SqlID** test enables you to achieve the same. The test also points you to time-consuming SQL queries to the undo tablespace, so that you can fine-tune them. In the process, the test also monitors the contents of the undo tablespace used by each instance and their undo retention period.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for the undo tablespace of each instance managed by the monitored Oracle RAC

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so,

ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg ;

create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;***

create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Maximum query time:	Indicates the time taken	Secs	Ideally, the value of this measure

Measurement	Description	Measurement Unit	Interpretation
	for executing a query on this tablespace in this instance.		should be low. An unusually high value for this measure could indicate that one/more queries are taking too long to execute on the undo tablespace. Use the detailed diagnosis of this measure to identify the the SQL IDs of teh top queries to the undo tablespace in terms of duration of execution. Inefficient queries can thus be isolated. Fine-tuning these queries will enable the optimal usage of the undo tablespace. Query processing will also be delayed if the undo tablespace is improperly sized. If the undo tablespace has insufficient space, many transactions to the tablespace may terminate before completion, and many more transactions may even hang; this will result in a long line of long-running queries. Also, since Oracle automatically tunes the undo retention period based on undo tablespace size and system activity, if the undo tablespace runs out of space, it will grossly affect the auto retention capability of Oracle, once again causing query failures. When available space for new transactions becomes short, the database begins to overwrite

Measurement	Description	Measurement Unit	Interpretation
			expired undo. If the undo tablespace has no space for new transactions after all expired undo is overwritten, the database may begin overwriting unexpired undo information. If any of this overwritten undo information is required for consistent read in a current long- running query, the query could fail with the snapshot too old error message.
Active blocks:	Indicates the number of active blocks available in this undo tablespace.	Number	These blocks consist of undo data that supports active transactions and are required in the event of rollback.
Unexpired blocks:	Indicates the number of unexpired blocks available in this undo tablespace.	Number	After a transaction is committed, undo data is no longer needed for rollback or transaction recovery purposes. However, for consistent read purposes, long- running queries may require this old undo information for producing older images of data blocks. Such type of blocks that are needed for supporting the **UNDO_ RETENTION** parameter is called as unexpired blocks.
Expired blocks:	Indicates the number of expired blocks available in this undo tablespace.	Number	The undo information that is no longer needed for rollback is stored in these expired blocks. These type of blocks will be over written by fresh information as and when required.
Tuned undo	Indicates the time taken	Secs	After a transaction is committed,

Measurement	Description	Measurement Unit	Interpretation
retention:	for undo retention of data blocks in this undo tablespace.		undo data is no longer needed for rollback or transaction recovery purposes. However, for consistent read purposes, long- running queries may require this old undo information for producing older images of data blocks. Furthermore, the success of several Oracle Flashback features can also depend upon the availability of older undo information. For these reasons, it is desirable to retain the old undo information for as long as possible. When automatic undo management is enabled, there is always a current undo retention period, which is the minimum amount of time that Oracle Database attempts to retain old undo information before overwriting it. Old (committed) undo information that is older than the current undo retention period is said to be expired. Old undo information with an age that is less than the current undo retention period is said to be unexpired. Oracle Database automatically tunes the undo retention period based on undo tablespace size and system activity. You can specify a minimum undo retention period (in seconds) by setting the **UNDO_**

Measurement	Description	Measurement Unit	Interpretation
			RETENTION initialization parameter. The database makes its best effort to honor the specified minimum undo retention period, provided that the undo tablespace has space available for new transactions. When available space for new transactions becomes short, the database begins to overwrite expired undo. If the undo tablespace has no space for new transactions after all expired undo is overwritten, the database may begin overwriting unexpired undo information. If any of this overwritten undo information is required for consistent read in a current long- running query, the query could fail with the snapshot too old error message. The following points explain the exact impact of the **UNDO_ RETENTION** parameter on undo retention: The **UNDO_RETENTION** parameter is ignored for a fixed size undo tablespace. The database may overwrite unexpired undo information when tablespace space becomes low. For an undo tablespace with the **AUTOEXTEND** option enabled, the database attempts to honor the

Measurement	Description	Measurement Unit	Interpretation
			minimum retention period specified by **UNDO_RETENTION**. When space is low, instead of overwriting unexpired undo information, the tablespace auto- extends. If the **MAXSIZE** clause is specified for an auto- extending undo tablespace, when the maximum size is reached, the database may begin to overwrite unexpired undo information.

3.3.4 Oracle RAC Flash Area Usage Test

The Flash Recovery Area is a specific area of disk storage that is set aside exclusively for retention of backup components such as datafile image copies, archived redo logs, and control file autobackup copies. These features include:

- **Unified Backup Files Storage**. All backup components can be stored in one consolidated spot. The Flash Recovery Area is managed via Oracle Managed Files (OMF), and it can utilize disk resources managed by Oracle Automated Storage Management (ASM). In addition, the Flash Recovery Area can be configured for use by multiple database instances if so desired.
- **Automated Disk- Based Backup and Recovery**. Once the Flash Recovery Area is configured, all backup components (datafile image copies, archived redo logs, and so on) are managed automatically by Oracle.
- **Automatic Deletion of Backup Components**. Once backup components have been successfully created, RMAN can be configured to automatically clean up files that are no longer needed (thus reducing risk of insufficient disk space for backups).
- **Disk Cache for Tape Copies**. Finally, if your disaster recovery plan involves backing up to alternate media, the Flash Recovery Area can act as a disk cache area for those backup components that are eventually copied to tape.
- **Flashback Logs**. The Flash Recovery Area is also used to store and manage flashback logs, which are used during Flashback Backup operations to quickly restore a database to a prior desired state.

Oracle recommends that the Flash Recovery Area should be sized large enough to include all files required for backup and recovery. Using this test, administrators can figure out whether the Flash Recovery Area is adequately sized or not, and accordingly make sizing recommendations.

Note:

This test is applicable only to clusters based on Oracle database server 10g (and above).

Target of the test : An Oracle database server 10g

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the flash recovery area on each node in the Oracle cluster

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;***

create user **<user_name>***identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used flash area:	Indicates the space currently occupied by the flash recovery files on this node.	MB	

Measurement	Description	Measurement Unit	Interpretation
Maximum flash area size:	Indicates the maximum space allocated for flash recovery files on this node.	MB	
Flash area usage:	Indicates the percentage of space occupied by the flash recovery files on this node.	Percent	Oracle recommends that the Flash Recovery Area should be sized large enough to include all files required for backup and recovery. Therefore, ideally, the value of this measure should be very low. A value close to 100% indicates excessive usage of the recovery area; this implies that the flash recovery area could soon run out of space. in such a case you can resize the flash recovery area by reconfiguring the parameter "*db_ recovery_ file_ dest_ size* " in database parameter file, provided enough disk space is available. If not, then Oracle recommends that the flash area be sized at least large enough to contain any archived redo logs that have not yet been backed up to alternate media. Alternatively, you can remove the old files from the flash recovery area to create space for the new recovery files.
Free flash area:	Indicates the free space currently available for recovery files on this node.	Percentage	

3.3.5 Oracle RAC ASM Disk I/O Test

ASM is a volume manager and a file system for Oracle database files that supports single-instance Oracle Database and Oracle Real Application Cluster (Oracle RAC) configuration. ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

ASM uses disk groups to store datafiles; an ASM disk group is a collection of disks that ASM manages as a unit. Within a disk group, ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group are evenly distributed, or striped, to eliminate hot spots and to provide uniform performance across the disks.

You need to periodically monitor the read-write activity on each disk in a disk group to make sure that I/O load is uniformly balanced across all disks in a group. The **ASM Disk I/O** test helps you do just that. At pre-configured intervals, this test monitors the I/O activity on each disk in every disk group of an Oracle cluster, reveals I/O-intensive and error-prone disks, and brings irregularities in load balancing to the fore.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle RAC* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *DiskGroup:Disk pair* in the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to

any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=<**Oracle_service_name**>*;*

   create user <**user_name**>*identified by* <**user_password**> *container=current default tablespace* <**name_of_default_tablespace**> *temporary tablespace* <**name_of_temporary_tablespace**>*;*

   Grant create session to <user_name>;

   Grant select_catalog_role to <**user_name**>*;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user

8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Reads:	Indicates the rate at which reads occur on this disk.	Reads/Sec	Compare the values of each of these measures across the disks in a disk group to identify the I/O-intensive disks in that group. In the process, you can also determine whether/not I/O load is equally balanced across all the disks in the group. If any irregularities are noticed in load-balancing are noticed, you may want to consider adding more disks to the group.
Writes:	Indicates the rate at which writes occur on this disk.	Writes/Sec	
Read errors:	Indicates the number of errors that occur per second while reading from this disk.	ReadErrors/Sec	The value 0 is desired for both these measures. A non- zero value is indicative of I/O errors. By comparing the values of each of these measures across disks and across disk groups, you can not only point to the error-prone disks and groups, but can also figure out when most of the errors occurred on the disk/group - when reading? or when writing?
Write errors:	Indicates the number of errors that occur per second when writing to this disk on this cluster node.	WriteErrors/Sec	

3.3.6 Oracle RAC ASM Disk Space Test

ASM is a volume manager and a file system for Oracle database files that supports single-instance Oracle Database and Oracle Real Application Cluster (Oracle RAC) configuration. ASM is Oracle's

recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

ASM uses disk groups to store datafiles; an ASM disk group is a collection of disks that ASM manages as a unit. Within a disk group, ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group are evenly distributed, or striped, to eliminate hot spots and to provide uniform performance across the disks.

To ensure that a disk group always has sufficient space to store the critical organizational data, you will have to continuously track the space usage of the disk group. This will provide you with early pointers to potential space contentions and help you swiftly provide more space to the group by adding more disks. The **ASM Disk Space** test enables you to achieve this end. This test closely monitors how each disk in a disk group uses the space available to it, points you to the disks that are running out of space, and thus holds a mirror to space contentions on a disk group.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page. To access this page, follow the Tests -> Enable/Disable menu sequence in the **Agents** tile of the **Admin** tile menu. In the **ENABLE/DISABLE TESTS** page, pick *Oracle Cluster* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle RAC

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each DiskGroup:Disk pair on the Oracle server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed.
2. **HOST** – The host for which the test is to be configured.
3. **PORT** - The port on which the server is listening.
4. **ORASID** - The variable name of the oracle instance.
5. **ORACLE HOME** - By default, this test auto-discovers the full path to the Oracle installation directory. This is why, the **ORACLE HOME** parameter is set to *none* by default.
6. **USE ASM INSTANCE** - By default, this flag is set to **No** indicating that the eG agent is capable of extracting the metrics related to the space utilization of the ASM disks from the **ORASID**. In some environments, the **ORASID** may provide outdated metrics. In such cases, set this flag to

Yes so that the eG agent can extract the metrics using the name of the ASM instance instead of the **ORASID**.

7. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By specifying a valid **SERVICE NAME**, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

8. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg ;

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_ name>** *identified by* **<user_ password>** *container=current default tablespace* **<name_ of_ default_ tablespace>** *temporary tablespace* **<name_ of_ temporary_tablespace>***;*

   Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

If the **ORACLE HOME** is set to *none* and the **USE ASM INSTANCE** flag is set to **Yes**, then, the user monitoring the Oracle database server should be vested with an additional *SYSDBA* privilege. To provide this privilege, do the following:

grant sysdba to user <user_name>;

The name of this user has to be specified here.

9. **PASSWORD** – Password of the specified database user
10. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Used space:	Indicates the amount of space currently used in this disk.	MB	Ideally, the value of this measure should be low. A consistent increase in this value is a cause for concern.
Free space:	Indicates the amount of space in this disk that is currently free - i.e., available for use.	MB	Ideally, the value of this measure should be high. A consistent decrease in this value is a cause for concern.
Space availability:	Indicates the percentage of space in this disk that is currently unused.	Percent	A high value is typically desired for this measure. By comparing the value of this measure across disks and across disk groups, you can quickly isolate the disks/groups that are running short of space. If the free space is alarmingly low for all disks in a group, it indicates that the group requires more space. You can then consider making

Measurement	Description	Measurement Unit	Interpretation
			space by adding more disks to the group.
Space usage:	Indicates the percentage of space in this disk that is currently used.	Percent	A low value is typically desired for this measure. By comparing the value of this measure across disks and across disk groups, you can quickly isolate the disks/groups that are utilizing space excessively. If the used space is alarmingly high for all disks in a group, it indicates that the group is rapidly running out of space. You can then consider making space by adding more disks to the group.
Used space growth:	Indicates the growth in space usage of this disk since the last measurement period.	MB/Sec	If you observe the variations to this measure over time, you will be able to detect early whether the space in the disk is being steadily eroded or not. This way, you can initiate measures to conserve space much before the disk exhausts all the space available to it.

If you observe the variations to this measure over time, you will be able to detect early whether the space in the disk is being steadily eroded or not. This way, you can initiate measures to conserve space much before the disk exhausts all the space available to it.

3.3.7 Oracle RAC Root Blockers Test

One common problem encountered with databases is blocking. Suppose that process A is modifying data that process B wants to use. Process B will be blocked until process A has completed what it is doing. This is only one type of blocking situation; others exist and are common. What matters to a database administrator is identifying when blocking is a problem and how to deal with it effectively. When blocking is bad enough, users will notice slowdowns and complain about it. With a large number of users, it is common for tens or hundreds of processes to be blocked when slowdowns are

noticed. Killing these processes may or may not solve the problem because 10 processes may be blocked by process B, while process B itself is blocked by process A. Issuing 10 kill statements for the processes blocked by B probably will not help, as new processes will simply become blocked by B. Killing process B may or may not help, because then the next process that was blocked by B, which is given execution time, may get blocked by process A and become the process that is blocking the other 9 remaining processes. When you have lots of blocking that is not resolving in a reasonable amount of time you need to identify the root blocker, or the process at the top of the tree of blocked processes. Imagine again that you have 10 processes blocked by process B, and process B is blocked by process A. If A is not blocked by anything, but is itself responsible for lots of blocking (B and the 10 processes waiting on B), then A would be the root blocker. (Think of it as a traffic jam. Figure 3.5 will help) Killing A (via kill) is likely to unblock B, and once B completes, the 10 processes waiting on B are also likely to complete successfully.

The Oracle RAC Root Blockers test reports the number of root blocker processes in the clustered database. The detailed diagnosis of this test, provides the details of each of these blocker processes, thereby enabling you to identify the root blocker.

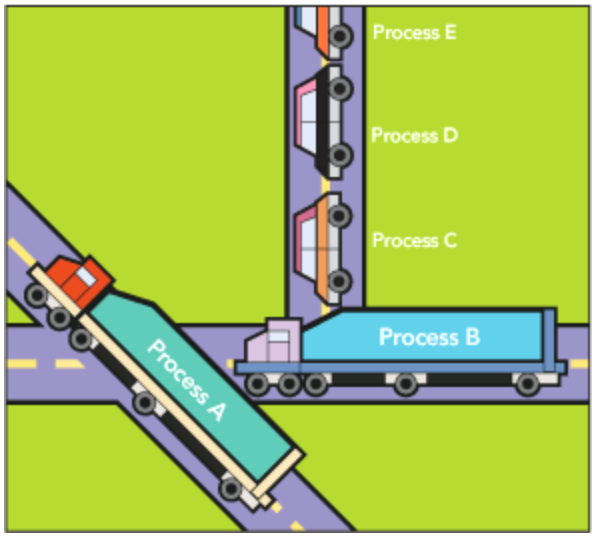


Figure 3.5: The traffic jam analogy representing blocking

Target of the test : Oracle Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle cluster monitored.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed

2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_name>***identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Root blockers:	Indicates the number of root blocker processes.	Number	If this value increases suddenly, this is a cause for concern. Likewise, if a process has been blocking other processes for a long time, it is a reason for further investigation. The detailed diagnosis for this test, if enabled, will indicate which process is blocking which other processes. Killing a blocker process that has been running for a long while may get the database running well again. Also, by carefully observing the details of the blocker processes, you can quickly identify the root blocker, and investigate the reason why it is blocking other processes.
Max blocking time	Indicates the maximum time for which a process blocked one/more processes.	Seconds	eG Enterprise isolates processes that have been blocking other processes for a duration greater than the configured **MAX BLOCKING**

Measurement	Description	Measurement Unit	Interpretation
			TIME. The blocking time of these processes is then compared and the maximum blocking time is identified and reported as the value of this measure. If this time is abnormally high, it indicates that a process been blocking resource access to other process(es) for a very long time. Prolonged blocking can significantly degrade database performance. Under such circumstances therefore, you can use the detailed diagnosis of the *Blocked sessions* measure to know which process was blocked for the maximum time and by which process.

3.3.8 Oracle RAC User Connections Test

This test reports the number and state of sessions of each user who is currently connected to the Oracle cluster. Using the metrics reported by this test, administrators can promptly isolate idle sessions, which are a drain on a cluster's resources.

Target of the test : Oracle RAC

Agent deploying the test : An internal/remote agent

Outputs of the test : One set of results for every user who is currently connected to the Oracle cluster

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured

3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   *alter session set container=***<Oracle_service_name>***;*

   create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

   Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
10. **EXCLUDEUSER** - In the **EXCLUDEUSER** text box, specify a comma-separated list of user names that need to be excluded from monitoring. By default, *none* is displayed here indicating that this test monitors connections initiated by all current users to the MS SQL server, by default.
11. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated. The default is *1:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. If you want the detailed diagnosis of this test to be generated at a different frequency, set a different **DD FREQUENCY** here. To disable the detailed diagnosis capability for a test, you can set this parameter to 0:0.
12. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total connections:	Indicates the total number of connections currently established by	Number	

Measurement	Description	Measurement Unit	Interpretation
	this user on the cluster.		
Active connections:	Indicates the number of connections of this user that are currently active.	Number	The detailed diagnosis of this measure, if enabled, will provide the complete details of the active sessions of a particular user. Using this information, you can understand how each of the connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.
Inactive connections:	Indicates the number of sessions initiated by this user that are currently idle.	Number	Ideally, the value of this measure should be low. A high value is indicative of a large number of idle sessions, which in turn causes the unnecessary consumption of critical server resources. Idle sessions also unnecessarily lock connections from the connection pool, thereby denying other users access to the server for performing important tasks. The detailed diagnosis of this measure, if enabled, will provide the complete details of the idle sessions of a particular user. Using this information, you can understand how each of the idle connections were made - i.e., using which program - from where - i.e., from which host - and to which cluster node.
Background connections:	Indicates the number of background processes	Number	Ideally, the value of this measure should be low.

Measurement	Description	Measurement Unit	Interpretation
	that were started when sessions are initiated by this user.		The detailed diagnosis of this measure, if enabled, will provide the complete details of the background sessions of a particular user. Using this information, you can understand how each of the background connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.
Blocked connections:	Indicates the number of sessions initiated by this user were blocked.	Number	Blocking occurs when one session holds a lock on a resource that another session is requesting. As a result, the requesting session will be blocked - it will hang until the holding session gives up the locked resource. In almost every case, blocking is avoidable. In fact, if you find that your session is blocked in an interactive application, then you have probably been suffering from the lost update bug as well, perhaps without realizing it. That is, your application logic is flawed and that is the cause of blocking. The five common DML statements that will block in the database are *INSERT, UPDATE, DELETE, MERGE* and *SELECT FOR UPDATE*. Ideally, the value of this measure should be low. A high value may cause unnecessary consumption of critical server resources thereby

Measurement	Description	Measurement Unit	Interpretation
			blocking access to potential active sessions. The detailed diagnosis of this measure, if enabled, will provide the complete details of the blocked sessions of a particular user. Using this information, you can understand how each of the blocked connections were made - i.e., using which program - and from where - i.e., from which host.
Cached connections:	Indicates the number of sessions of this user that were cached for future use.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the cached sessions of a particular user. Using this information, you can understand how each of the cached connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.
Killed connections:	Indicates the number of sessions of this user that were terminated due to inactivity.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the killed sessions of a particular user. Using this information, you can understand how each of the killed

Measurement	Description	Measurement Unit	Interpretation
			connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.
Sniped connections:	Indicates the number of sessions of this user that were idle for a period more than the profile's maximum idle time while waiting for a client's response.	Number	Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the sniped sessions of a particular user. Using this information, you can understand how each of the sniped connections were made - i.e., using which program - from where - i.e., from which host – and to which cluster node.

3.3.9 Oracle RAC Cursor Usage Test

This test monitors the number of open cursors for every node of an Oracle cluster.

Target of the test : Oracle Server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every cluster node in the Oracle cluster monitored.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients

connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   *alter session set container=***<Oracle_service_name>***;*

   create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Current open cursors:	The number of cursors currently opened by this node on the shared cluster database.	Number	Many open cursors can exist if any application does not properly close the ResultSets before closing a connection. Alternatively, many simultaneous queries to the database can also result in many open cursors. A continuous increase in open cursors is an indicator of a problem in an application's use of the database.
Percent open cursors:	This metric reports the average percentage of open cursors with respect to the total allowed limit.	Percent	If the percentage of open cursors nears 100%, then this could invoke the "maximum open cursors exceeded" error message. If the percentage is consistently near 100%, consider increasing the value of the 'open_ cursors' parameter in the init file.

3.3.10 Oracle RAC Datafile Activity Test

This test indicates the level of read/write activity on each datafile in the shared cluster storage.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every datafile in the shared cluster storage monitored.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>**;

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. **SHOW DATAFILE PATH** - This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the **SHOW DATAFILE PATH** flag is set to **No** by default. If you want the data file names to be prefixed by the full path to the data files, then, set the **SHOW DATAFILE PATH** flag to **Yes**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Physical block read rate:	Indicates the rate at which disk blocks are being read from this datafile.	Blocks/Sec	A scenario in which more than 50% of blocks are being read from a single datafile could signify a problem.
Physical block write rate:	Indicates the rate at which disk blocks are being written to this datafile.	Blocks/Sec	A scenario in which more than 50% of blocks are being written to a single datafile could signify a problem. Too much activity to a specific datafile can result in reduced database performance. To improve performance, consider balancing I/O across disks, and reorganize tables across

Measurement	Description	Measurement Unit	Interpretation
			tablespaces to reduce activity to a specific datafile.
Percent total I/O:	Indicates the percentage of total I/O operations on the database server that were handled by this data file.	Percent	Disk reads and writes are expensive operations and all I/Os should be balanced across the different data files of an Oracle database for optimal performance. This metric reports the percentage of all I/O of an Oracle database that are happening on each of the data files of the Oracle database. This metric allows an Oracle administrator to determine which is/are the hot data file (s) (e.g., which data file is handling 80% of the total I/O).

3.3.11 Oracle RAC Data File Errors Test

The most common reasons for data file errors are corrupted blocks and invalid blocks. Both these can cause damage to portions of the database or the whole database, and can thus result in minimal to heavy loss of data. This is why, you should waste no time in identifying the error-prone data files and in doing all that is necessary to clear the errors and salvage the data. The **Oracle Data File Errors** test can play a key role in this exercise.

This test combs all the data files in the shared cluster storage for errors and reports the number of errors (if any). The test also provides the complete details of every error, thus enabling a speedy and effective resolution.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>**;

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Error files count:	Indicates the number of data file errors that have occurred.	Number	Ideally the value of this measure should be zero. The detailed diagnosis of this measure indicates the File number, Status, Error, Recover and the Tablespace name for each error that has occurred in the datafiles.

3.3.12 Oracle RAC Database File Status Test

This test reports the status of each datafile in the shared cluster storage and the current access mode of every datafile.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every datafile in the shared storage of the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=<Oracle_service_name>;

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>;**

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>;**

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
10. **SHOW DATAFILE PATH**- This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the **SHOW DATAFILE PATH** flag is set to **No** by default. If you want the data file names to be prefixed by the full path to the data files, then, set the **SHOW DATAFILE PATH** flag to **Yes**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
File status:	Indicates the current status of this datafile.		The table below indicates the values that this measure can report and their

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>corresponding numeric equivalents:
<table><tr><th>Numeric Value</th><th>Measure Value</th></tr><tr><td>1</td><td>System</td></tr><tr><td>2</td><td>Online</td></tr><tr><td>3</td><td>Recover</td></tr><tr><td>4</td><td>Unknown</td></tr></table>
If a datafile is part of the SYSTEM tablespace, its status is SYSTEM (unless it requires recovery).

If a datafile in a non-SYSTEM tablespace is online, its status is ONLINE. If a datafile in a non-SYSTEM tablespace is offline, its status can be either OFFLINE or RECOVER.

Note:

By default, this measure reports the above- mentioned Measure Values while indicating the current status of a datafile. However, in the graph of this measure, data file states will be represented using the corresponding numeric equivalents only - i.e., 1 to 4.</td></tr>
<tr><td>File access mode:</td><td>Indicates the current access mode of this datafile.</td><td></td><td>The table below indicates the values that this measure can report and their corresponding numeric equivalents:</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			<table><tr><th>Numeric Value</th><th>Measure Value</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Read Only</td></tr><tr><td>2</td><td>Read Write</td></tr><tr><td>3</td><td>Unknown</td></tr></table> **Note:** By default, this measure reports the above- mentioned **Measure Value** s while indicating the mode through which this datafile can be accessed. However, the graph of this measure will be represented using the corresponding numeric equivalents i.e., 0 to 3.

3.3.13 Oracle RAC Database Growth Test

Periodic monitoring of the usage of the shared cluster storage is essential to ensure that the cluster is always adequately sized to handle current and future loads. The Ora

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle cluster monitored.

Configurable parameters for the test

cle RAC Database Growth test monitors the usage of a shared storage, and indicates if it requires resizing.

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening

4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'
6. **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg

   create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is::

   alter session set container=**<Oracle_service_name>***;*

   create user **<user_name>***identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

   Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user

   This login information is required to query Oracle's internal dynamic views, so as to fetch the current status / health of the various database components.

8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.

9. **ALTERNATE VIEW** – In large environments, where the volume of transactions to the Oracle database server is generally very high, this test may take time to execute and retrieve the desired results. To ensure that the test is faster and is resource-efficient, administrators of such environments can create an alternate 'view' on the target Oracle database server, and grant *select* privileges to the view to the special database **USER** mentioned above. Once the view is created, the test should be configured to use the alternate view for metrics collection; to achieve this, specify the name of the view in the **ALTERNATE VIEW** text box. By default, this text box is set to *none*, which implies that the alternate view is not used by default.

   This alternate 'view' should be created with the following structure:

```
CREATE OR REPLACE VIEW <VIEW_NAME> (
TABLESPACE_NAME,
FILE_ID,
BLOCK_ID,
BYTES,
BLOCKS,
RELATIVE_FNO
) AS
select /*+ use_hash (tsfi, fet2) */ tsfi.tablespace_name,
       tsfi.file_id,
       fet2.block_id,
       tsfi.blocksize * fet2.blocks,
       fet2.blocks,
       tsfi.relfile#
 from   (select /*+ use_hash (ts, fi) */ ts.name tablespace_name,
               fi.file# file_id,
               ts.BLOCKSIZE,
               fi.relfile#,
               ts.ts#
        from   sys.ts$ ts,
               sys.file$ fi
        where  ts.ts# = fi.ts#
        and    ts.online$ in (1,4)) Tsfi,
       (select f.block# block_id,
               f.length blocks,
               f.file# file_id,
               f.ts#
        from   sys.fet$ f
        union all
        select f.ktfbfebno block_id,
               f.ktfbfeblks blocks,
               f.ktfbfefno,
               ktfbfetsn
        from   sys.x$ktfbfe f) Fet2
where  fet2.file_id = tsfi.relfile#
and    fet2.ts# = tsfi.ts# ;
```

10. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable’ by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Allocated size of database:	Indicates the amount of space currently allocated to the datafiles for use.	GB	
Used space in allocated:	Indicates the amount of allocated space currently used by the datafiles.	GB	
Free space in allocated:	Indicates the amount of allocated space that is still unused by the datafiles.	GB	
Space usage in allocated:	Indicates the percentage of allocated space that has been utilized by the datafiles.	Percent	If the use max size parameter of this test has been set to No, then the value of this measure will be computed using the following formula: Used space/Total size of database * 100 If the use max size parameter of this test has been set to Yes, then the value of this measure will be computed using the following formula: Used space/Maximum size upto which the database can grow * 100 Ideally, this value should be low. A value close to 100% is a cause for concern.
Free percentage in	Indicates the	Percent	

Measurement	Description	Measurement Unit	Interpretation
allocated size:	percentage of allocated space that is unused.		
Max size of database:	Indicates the maximum size upto which the shared cluster storage can grow.	GB	
Used space in max size:	Indicates the amount of space used by datafiles.	GB	
Free space in max size:	Indicates the amount of space that is still unused by datafiles.	GB	
Space usage in max size:	Indicates the percentage of max space that is currently used by datafiles.	Percent	
Free percentage in max size:	Indicates the percentage of max space that is available for use.	Percent	
Space free:	Indicates the percentage of free space in this database instance.	Percent	If the use max size parameter of this test has been set to No, then the value of this measure will be computed using the following formula: Free space/Total size of database * 100 If the use max size parameter of this test has been set to Yes, then the value of this measure will be computed using the following formula: Free space/Maximum size upto

Measurement	Description	Measurement Unit	Interpretation
			which the database can grow * 100 Ideally, this value should be high. A sudden/consistent decrease in the value of this measure could indicate excessive utilization of the database caused by a sporadic/steady increase in database activity. Very low free space in a database instance could significantly deteriorate its performance. Under such circumstances therefore, you might want to check the measures reported by the Oracle Datafile GrowthTest to figure out which datafile is consuming too much space. You might then want to resize the datafile.

3.3.14 Oracle RAC DB Wait Time Test

Oracle's response time for an operation is composed of time executing (=CPU time) and time spent waiting (=Waiting time). An increase in either or both the above-mentioned factors will adversely impact the responsiveness of the Oracle cluster service.

When Oracle executes an SQL statement, it is not constantly executing. Sometimes it has to wait for a specific event to happen before it can proceed. For example, if Oracle (or the SQL statement) wants to modify data, and the corresponding database block is not currently in the SGA, Oracle waits for this block to be available for modification. The Waiting time refers to the time spent by the Oracle server waiting for such events to complete. Oracle has a bunch of events that it can wait for - eg., buffer busy waits, db file scattered read, db file sequential read.

Whenever users complaint of a slowdown while accessing databases in a cluster, it would be helpful to know which node is experiencing a slowdown and where it is spending too much time - is the time executing more than the time spent waiting, or vice-versa? To determine this, you should monitor both the CPU time and the Waiting time of each node of the cluster. This test enables you to perform 'half' this analysis. In other words, this test reports the percentage of time that every node spent on

waiting for one/more events to complete. This way, the test helps you understand whether/not the waiting time is contributing to the poor responsiveness of the cluster service and which node has been waiting too long.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the Oracle Cluster as the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each node in the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the

select_catalog_role and *create session* privileges.
The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=<Oracle_service_name>;

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
DB time spent waiting:	Indicates the percentage of time this node spent on waiting for one/more events to complete.	Percent	A high value is indicative of the following cases: • An increase in load (either more users, more calls, or larger

Measurement	Description	Measurement Unit	Interpretation
			transactions) • I/O performance degradation (I/O time increases and wait time increases, so DB time increases) • Application performance degradation • CPU- bound host (foregrounds accumulate active run- queue time, wait event times are artificially inflated)

3.3.15 Oracle RAC Defer Transactions Test

Oracle uses deferred transactions to propagate data-level changes asynchronously among master sites in an advanced replication system as well as from an updatable snapshot to its master table.

This test reports the number of deferred transactions in the shared cluster storage.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the *Oracle Cluster* as the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance

5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   *alter session set container=***<Oracle_service_name>***;*

   create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Defer transaction count:	Indicates the number of deferred transactions in the shared cluster storage.	Number	When the advanced replication facility pushes a deferred transaction to a remote site, it uses a distributed transaction to ensure that the transaction has been properly committed at the remote site before the transaction is removed for the queue at the local site. If transactions are not being pushed to a given remote site, verify that the destination for the transaction was correctly specified. If you specify a destination database when calling *DBMS_ DEFER_ SYS.SCHEDULE_ EXECUTION* using the DBLINK parameter, or *DBMS_ DEFER_ SYS.EXECUTE* using the *DESTINATION* parameter, make sure the full database link is provided.

3.3.16 Oracle RAC Index Fragmentation Test

Indexes are Oracle database objects that provide a fast, efficient method of retrieving data from database tables. The physical addresses of required rows can be retrieved from indexes much more

efficiently than by reading the entire table. Effective indexing usually results in significant improvements to SQL performance.

Oracle's default index structure is B*-tree, which stands for "Balanced tree." It has a hierarchical tree structure. At the top is the header. This block contains pointers to the appropriate branch block for any given range of key values. The branch block points either to another branch block, if the index is big, or to an appropriate leaf block. Finally, the leaf block contains a list of key values and physical addresses (ROWIDs) of rows in the database. Theoretically, any row in a table, even a big one, could be retrieved in a maximum of three or four I/Os (input/output operations): one header block, one or two branch block(s), and one leaf block.

The advantages of indexing do not come without a cost. As database objects, indexes are created for tables only and they must be in sync with them: indexes must be updated by the database with every data manipulation language (q) operation - *INSERT, DELETE*, or *UPDATE*. Where there are a large number of tables with dynamic data, too many *INSERTs, DELETEs*, and *UPDATEs* on the tables can over time, fragment the index. When indexes are fragmented, queries take longer to pull out rows from tables, thereby significantly increasing disk I/O. This adversely impacts overall SQL performance.

The first step to resolving the performance threat posed by fragmented indexes is to identify which indexes are fragmented. The **Oracle RAC Index Fragmentation** test helps in this regard. This test scans a pre-configured index sample for high and very high levels of fragmentation, and reports the count of fragmented indexes. Using the detailed diagnosis capability of the test, you can also quickly drill down to the specific indexes that have been fragmented. You can thus proceed to rebuild the fragmented indexes to reduce disk I/O.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the Oracle Cluster as the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every *DisplayName* configured for the **OBJECT NAME** parameter of this test

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
10. **OBJECT NAME** - Specify a comma-separated list of tables, the indexes of which need to be checked for fragmentation. Every table name should be specified in the following format: *<DisplayName>:<schema_name>.<table_name>*, where *schema_name* refers to the name of the table owner, and *table_name* refers to the name of the table. The *DisplayName* in your specification will appear as the descriptor of this test. For instance, to monitor the indexes of the *alarm* and *history* tables owned by user *admin* , your specification would be: *AlarmMon1:admin.alarm,AlarmMon2:admin.history*. To monitor all tables in a schema, the specification would be of the following format: *<DisplayName>:<schema_ name>.**. For example, to monitor all the tables in the *admin* schema, your specification would be: *AlarmMon:admin.**.

    You can also configure the **OBJECT NAME** to indicate what percentage of records in a table are to be considered by this test for running index fragmentation checks. To achieve this, your **OBJECT NAME** specification should be of the following format: *<DisplayName>:<schema_ name>.<table_name>@<Percentage_of_records_in_the table>*. For instance, say that you want to configure this test to monitor the indexes that correspond to **20%** of the *alarm* table and **30%** of the *history* table. The **OBJECT NAME** specification in this case will be: *AlarmMon:admin.alarm@20,AlarmMon1:admin.history@30*. **It is recommended that you keep this 'percentage value' small, as higher values will make this test that much more resource-intensive**.

    Note:

    Make sure that you configure the **OBJECT NAME** parameter with only table names and not view names. This is because, indexes are available for tables alone and not views.

11. **QUERYTIMEOUT** - Specify the time period upto which a query has to wait to obtain the required result set from the database in the **QUERYTIMEOUT** text box. If the query is not successful or if the query waits for a time period exceeding the specified time limit, the test will automatically kill the query.
12. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Highly fragmented Oracle indexes:	Indicates the number of highly fragmented indexes.		If 30% - 49% of an index is found to be fragmented, then such an index is counted as a highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates high index fragmentation. Fragmentation is characterized by splitting and spawning. Splitting happens when an index node becomes full with keys and a new index node is created at the same level as a full node. This widens the B*-tree horizontally. Spawning is the process of adding a new level to an index. As a new

Measurement	Description	Measurement Unit	Interpretation
			index is populated, it begins life as a single- level index. As keys are added, a spawning takes place and the first- level node reconfigures itself to have pointers to lower-level nodes. Both these phenomenon are key performance degraders. This is why, a high value of this measure, if left unchecked, can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify highly fragmented indexes, and proceed to rebuild them.
Very highly fragmented Oracle indexes:	Indicates the number of indexes that are very highly fragmented.	Number	If 50% or more of an index is found to be fragmented, then such an index is counted as a very highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates very high index fragmentation. Fragmentation is characterized by splitting and spawning. Splitting happens when an index node becomes full with keys and a new index node is created at the same level as a full node. This widens the B*-tree horizontally.

Measurement	Description	Measurement Unit	Interpretation
			Spawning is the process of adding a new level to an index. As a new index is populated, it begins life as a single- level index. As keys are added, a spawning takes place and the first- level node reconfigures itself to have pointers to lower-level nodes. Both these phenomenon are key performance degraders. This is why, a high value of this measure, if left unchecked, can cause disk I/O to mount, queries to run for long periods, and all performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify very highly fragmented indexes, and proceed to rebuild them.

3.3.17 Oracle RAC Jobs Test

This test monitors Oracle jobs and reports the number of jobs that have failed and those that are broken. The detailed diagnosis capability offered by this test enables administrators perform further diagnosis on failed/broken jobs, by additionally revealing the complete details of the failed and broken jobs.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick the Oracle Cluster as the desired **Component type**, set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list, and click on the **>>** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Oracle cluster

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>**;

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>**;

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>**;

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
10. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Failed Oracle jobs:	Indicates the number of jobs that failed.	Number	Ideally, the value of this measure should be 0. Any value greater than zero, is a cause of concern, as it indicates the existence of a failed job. To know which job(s) has failed, use the detailed diagnosis capability of this measure.

Measurement	Description	Measurement Unit	Interpretation
			Typically, if a job fails, Oracle attempts to run the job again 16 times, at fixed time intervals. You are advised to investigate the reason for the failure and fix it, by the time Oracle completes its 16th attempt. This is because, if the 16th attempt too fails, Oracle flags the job as a 'broken job', which can then be executed only manually.
Broken Oracle jobs:	Indicates the number of jobs broken.	Number	Ideally, the value of this measure should be 0. Any value greater than 0 is a problem, as it indicates the existence of one/more broken jobs. A job is considered broken, only if the 16th attempt made by Oracle to run the job fails. To know which jobs have broken, use the detailed diagnosis capability of this measure. Once the jobs are identified, you can proceed to manually run the broken jobs through the DBMS_JOB.RUN procedure after logging in as the owner of that job.

3.3.18 Oracle RAC Latches Test

Latches are mechanisms for protecting and managing SGA data structures and database objects being accessed concurrently. Unlike locks, latches provide exclusive access to protected data structures. Requests for latches are not queued. So, if a request fails, the requesting process may try later. Typically, latches are used to protect resources that are briefly needed.

An Oracle process can request a latch in one of the following two modes:

- Willing-to-Wait Mode: If the requested latch is not immediately available, the process will wait. When an attempt to get a latch in a willing-to-wait mode fails, the process will spin and try again. If the number of attempts reaches the value of the SPIN_COUNT parameter, the process

sleeps. Sleeping is more expensive than spinning.

- Immediate Mode (no-wait mode): In this case, the process will not wait if the requested latch is not available and it continues its processing.

Latch contention has a significant impact on performance when:

a. Enough latches are not available

b. A latch is held for a relatively long time

Latch mechanisms most likely to suffer from contention involve requests to write data into the redo log buffer. To serve the intended purpose, writes to the redo log buffer must be serialized. There are four different groupings applicable to redo buffer latches: redo allocation latches and redo copy latches, each with immediate and willing-to-wait priorities.

The **Oracle RAC Latches** test is used to monitor latches in the shared cluster storage.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every node in the Oracle cluster monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   *alter session set container=***<Oracle_service_name>***;*

   create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

   Grant create session to <user_name>;

   Grant select_catalog_role to **<user_name>***;*

   The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Willing- to- wait misses:	This measures the latch contention for requests that were willing to wait to acquire a latch. The value of this metric represents the ratio of the number of requests that could not acquire a latch, to those that could acquire a latch.	Percent	Both the above metrics should be 1% or less. For redo allocation latches, if the Willing_ to_ wait_ misses is high, consider decreasing the *LOG_ SMALL_ ENTRY_ MAX_ SIZE* parameter in the *INIT.ORA* file. By making the max size for a redo allocation latch smaller, more redo log buffer writes qualify for a redo copy latch instead, thus better utilizing multiple CPU's for the redo log buffer writes. Even though memory structure manipulation times are measured in nanoseconds, a larger write still takes longer than a smaller write. If the size for remaining writes done via redo allocation latches is small enough, they can be completed with little or no redo allocation latch contention. On a single CPU node, all log buffer writes are done via redo allocation latches. If log buffer latches are a significant bottleneck, performance can benefit from additional CPU's (thus enabling redo copy latches) even if the CPU utilization is not an operating system level bottleneck. If the values for redo copy latches is > 1%, consider increasing the LOG_ SIMULTANEOUS_ COPIES

Measurement	Description	Measurement Unit	Interpretation
			parameter in the INIT.ORA file. This initialization parameter is the number of redo copy latches available. It defaults to the number of CPU's (assuming a multiple CPU node). Oracle recommends setting it as large as 2 times the number of CPU's on the particular node, although quite a bit of experimentation may be required to get the value adjusted in a suitable manner for any particular instance's workload. Depending on CPU capability and utilization, it may be beneficial to set this initialization parameter smaller or larger than 2 X #CPU's. Note that the *LOG_ SIMULTANEOUS_ COPIES* parameter obsolete from Oracle 8i onwards. Hence, if you are monitoring Oracle 8i (or higher), use the hidden parameter *_ LOG_ SIMULTAENOUS_ COPIES* instead. Recall that the assignment of log buffer writes to either redo allocation latches or redo copy latches is controlled by the maximum log buffer write size allowed for a redo allocation latch, and is specified in the *LOG_ SMALL_ ENTRY_ MAX_ SIZE* initialization parameter. Recall also that redo copy latches apply only to multiple CPU hosts. Note that the *LOG_ SMALL_ ENTRY_*

Measurement	Description	Measurement Unit	Interpretation
			MAX- SIZE parameter is not supported from Oracle 9i onwards.
Immediate misses:	This metric measures the latch contention for requests that were not willing to wait to acquire a latch. The value of this metric represents the percentage of "not willing to wait" latch requests that failed. In other words: **the number of "not willing to wait" request misses / the total number of "not willing to wait" requests**	Percent	
Immediate misses:	This metric measures the latch contention for requests that were not willing to wait to acquire a latch. The value of this metric represents the percentage of "not willing to wait" latch requests that failed. In other words: **the number of "not willing to wait" request misses / the total number of "not willing to wait" requests**	Percent	

3.3.19 Oracle RAC Long Running Queries Test

This test tracks the currently executing queries on each node of an Oracle cluster and determines the number of queries that have been running for a long time and on which node.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each node in the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=<**Oracle_service_name**>*;*

create user <**user_name**>*identified by* <**user_password**> *container=current default tablespace* <**name_of_default_tablespace**> *temporary tablespace* <**name_of_temporary_tablespace**>*;*

Grant create session to <user_name>;

Grant select_catalog_role to <**user_name**>*;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
10. **ELAPSED TIME** - In the **ELAPSED TIME** text box, specify the duration (in seconds) for which a query should have executed for it to be regarded as a long running query. The default value is 10.
11. **DISPLAYQUERY FULLTEXT** - The detailed diagnosis of this test lists the queries that have been running for a long time. In the **DETAILED DIAGNOSIS** page by default, query strings that are very long are truncated to display the first 1000 characters of the query alone. This is why, the **DISPLAYQUERY FULLTEXT** flag is set to **No** by default. To view the full query in the detailed diagnosis page, set this flag to **Yes**. **Note that setting this flag to 'Yes' may increase the size of your eG database**.
12. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off**

option.

The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

- The eG manager license should allow the detailed diagnosis capability
- Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Long running queries:	Indicates the number of queries currently executing on the this cluster node that have been running for more time than the configured **ELAPSED TIME**.	Number	The detailed diagnosis for this measure indicates the exact queries and which user is executing the queries. This information can be very useful in identifying queries that may be candidates for optimization.

3.3.20 Oracle RAC Redo Logs Test

Redo logs are applied during the roll forward phase of the recovery process. These logs hold information about the changes made to the database and whether they were committed. Each change is recorded in a redo record, which has information like the SCN of the change, changed data, commit flag, and information about which data block is changed. The Oracle RAC Redo Logs test monitors key performance metrics pertaining to the redo log buffer in each node of an Oracle cluster.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every node in a cluster

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured

3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=<**Oracle_service_name**>*;*

   create user <**user_name**>*identified by* <**user_password**> *container=current default tablespace* <**name_of_default_tablespace**> *temporary tablespace* <**name_of_temporary_tablespace**>*;*

   Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Redo buffer entries:	This indicates the number of attempts to allocate space in the redo buffer of this node.	Number	A value other then 0 indicates that the redo writer is falling behind. This could be caused by log switches or checkpoints. By adjusting the *LOG_ CHECKPOINT_ INTERVAL* and *LOG_ CHECKPOINT_ TIMEOUT* parameters in the init.ora, you will be able to minimize the number of checkpoints. From Oracle 9i onwards however, the *LOG_ CHECKPOINT_ INTERVAL* parameter is supported only for ensuring backward compatability with previous versions of Oracle. The recommended equivalent in case of Oracle 9i therefore is *FAST_START_MTTR_TARGET*. You can also increase the number of LGWR writers. These parameters are new in Oracle 8 and are defined in the init.ora

Measurement	Description	Measurement Unit	Interpretation
			parameters *LGWR_ IO_ SLAVES and ARCH_ IO_ SLAVES* . However, note that both these parameters are obsolete from Oracle 8i onwards.
Redo log space requests:	The active log file is full and Oracle is waiting for disk space to be allocated for the redo log entries on this cluster node. Space is created by performing a log switch.	Number	Small Log files in relation to the size of the SGA or the commit rate of the work load can cause problems. When the log switch occurs, Oracle must ensure that all committed dirty buffers are written to disk before switching to a new log file. If you have a large SGA full of dirty buffers and small redo log files, a log switch must wait for DBWR to write dirty buffers to disk before continuing.
Redo entries:	This statistic increments each time redo entries are copied into the redo log buffer on this cluster node. (ie. The number of attempts to allocate space in the redo)	Number	
Log space requests:	This indicates the percentage of log space requests on this cluster node.	Percentage	If the number is greater than 1%, you should increase the size of the Redo Log buffer. I would also check the checkpoint and size of the online redo log file.
Log space waits:	This measure indicates the number of times wait has happened to acquire a log buffer on this node.	Number	If the Log Buffer space waits exist, consider increasing the size of the redo log. Also I would check the speed of the disk that the Online Redo Log files are in.

3.3.21 Oracle RAC SGA Test

The System Global Area (SGA) is the most important memory structure in Oracle. The SGA stores several different components of memory usage that are designed to execute processes to obtain data for user queries as quickly as possible while also maximizing the number of concurrent users that can access the Oracle instance. The main components of the SGA are

- The buffer cache: This area of memory allows for selective performance gains on obtaining and changing data. The buffer cache stores data blocks that contain row data that has been selected or updated recently. When the user wants to select data from a table, Oracle looks in the buffer cache to see if the data block that contains the row has already been loaded. If it has, then the buffer cache has achieved its selective performance improvement by not having to look for the data block on disk. If not, then Oracle must locate the data block that contains the row, load it into memory, and present the selected output to the user.
- The shared pool: The two main components of the shared pool are the shared SQL library cache and the data dictionary cache. The shared SQL library cache is designed to store parse information for SQL statements executing against the database. Parse information includes the set of database operations that the SQL execution mechanism will perform in order to obtain data requested by the user processes. This information is treated as a shared resource in the library cache. If another user process comes along wanting to run the same query that Oracle has already parsed for another user, the database will recognize the opportunity for reuse and let the user process utilize the parse information already available in the shared pool. The other component of the shared pool is the data dictionary cache, also referred to by many DBAs as the “row” cache. This memory structure is designed to store the data from the Oracle data dictionary in order to improve response time on data dictionary queries. Since all user processes and the Oracle database internal processes use the data dictionary, the database as a whole benefits in terms of performance from the presence of cached dictionary data in memory.

An Oracle database server brings in data into the SGA before doing any operation on it. So it is critical to monitor the various structures inside the SGA to ensure optimal database performance. The Oracle RAC SGA test collects a variety of statistics relating to the various SGA components on each node in an Oracle cluster.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every node in the Oracle cluster

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=**<Oracle_service_name>***;*

create user **<user_name>***identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Library cache hit ratio:	The library cache is a buffer that contains the shared SQL and PL/SQL areas. The library cache hit ratio indicates the percentage of shared SQL statements being reparsed by this cluster node.	Percent	For a well-tuned database, this ratio is 90% or more. A lower hit ratio may indicate that the memory allocation to the library cache is insufficient. A low value can significantly degrade the database performance. Increasing the value of the *SHARED_ POOL_ SIZE* initialization parameter will help in improving the hit ratio.
Data buffer cache hit ratio:	Indicates the percentage of time that this cluster node is able to satisfy a request with information that is already available in the memory.	Percent	Physical I/O takes a significant amount of time, and also increases the CPU resources required. The database configuration should be tuned to ensure that a required block will most likely be in memory. The extent to which this is achieved is measured using the buffer cache hit ratio. For a well-tuned database,

Measurement	Description	Measurement Unit	Interpretation
			this ratio should be 80% or higher. A lower value indicates insufficient memory allocation to the database buffer cache. Increasing the value of the *DB_ BLOCK_ BUFFERS* initialization parameter will help in improving the hit ratio. If you are monitoring Oracle 9i or higher, then, note that the *DB_ BLOCK_ BUFFERS* parameter is not supported in Oracle 9i or above. It is therefore recommended that you use the equivalent *DB_ CACHE_ SIZE* parameter instead.
Dictionary cache hit ratio:	Indicates the percentage of data dictionary information pertaining to the database, file space availability and object privileges being readily available in this cluster node's memory.	Percent	As with the case of the library cache, the dictionary cache hit ratio should be at least 90%. A lower value may be due to the insufficient memory allocation to the dictionary cache. Increasing the value of the *SHARED_POOL_SIZE* parameter will help in improving the hit ratio.
Redo log buffer misses:	Indicates the percentage of requests to this cluster node that had to wait before the redolog buffer is allocated to it.	Percent	Before any transaction could occur, the before image of the data will be stored in the redo log buffer. It is crucial to make the redo log buffer available immediately to the transactions without any wait. The above is crucial to improve the overall performance. This measure indicates how many percentage of times it had to wait for a redo log buffer to be allocated. This can be

Measurement	Description	Measurement Unit	Interpretation
			improved by increasing the *LOG_ BUFFER* parameter.
Sorts on disk:	Indicates the percentage of sorts that is happening on the secondary storage disk of this cluster node.	Percent	For best performance, most sorts should occur in memory; sorts written to disk adversely affect performance. If more than 10% of sorts happen on disk, the database performance could degrade. To improve the sorting performance of a database, consider tuning the parameters *SORT_ AREA_ SIZE* and *SORT_ AREA_ RETAINED_ SIZE*. The dynamically modifiable initialization parameter called *SORT_ AREA_ SIZE* specifies the maximum amount of memory to use for each sort. If a significant number of sorts require disk I/O to temporary segments, an application's performance may benefit from increasing the size of the sort area. Oracle 9i (or above) supports the *SORT_ AREA_ SIZE* and the *SORT_ AREA_ RETAINED_ SIZE* parameters only to ensure backward compatibility with previous versions of Oracle. Therefore, while monitoring Oracle 9i or higher, it is recommended that you use the equivalent *PGA_ AGGREGATE_ TARGET* parameter instead.
Current size:	Indicates the amount of space allocated to the SGA that is currently in	MB	A consistent and significant increase in the value of this measure is a cause for concern, as

Measurement	Description	Measurement Unit	Interpretation
	use.		it indicates that SGA components are over- utilizing the available memory resources. In such a scenario, you can use the detailed diagnosis of the *Current size* measure to know the memory usage of the individual SGA components. In the process, you can identify the exact SGA component that is over-utilizing the memory resources.
Buffer nowait:	Indicates the percentage of requests a server process makes for a specific buffer where the buffer was available immediately.	Percent	If this ratio falls below 90%, it indicates that the server process has to wait for something before obtaining the buffer. In this case, determine which type of block is being contended for by examining the Buffer Waits Section of Statspack/ AWR report.
Soft parse:	Indicates the percentage of parse requests where the cursor was already in the cursor cache compared to the number of total parses.	Percent	A soft parse is recorded when the Oracle Server checks the shared pool for a SQL statement and finds a version of the statement that it can reuse. If the value of this measure falls below 90%, it indicates that very often server processes are unable to find SQL statements in the shared pool and are forced to perform hard parses for these statements. Soft parses consume less resources than hard parses, so the

Measurement	Description	Measurement Unit	Interpretation
			larger the value for this item, the better.
Execute to parse:	Is a measure of how many times you execute a sql statement versus parse it.	Percent	If this value is too low, it indicates that an application is parsing statements highly, but not executing properly. This could result in excessive CPU usage, increased shared pool latches, and serious performance degradations in the Oracle database server. The execute to parse ratio takes a hit when an application does not use shareable SQL or if the database has sub- optimal parameters that are reducing the effectiveness of **cursor sharing**. A problem like excessive parsing is likely to manifest itself as additional network traffic between the application server and clients. The additional parse activity may also show up as a marked increase in CPU consumption on the database server.
Parse CPU to parse elapsed:	Indicates the percentage of CPU time used when parsing.	Percent	Parse CPU time means amount of CPU time used for parsing. Parse Elapsed time means amount of clock time used for parsing – this is actually the sum of Parse CPU time and Parse Wait time. The *Parse CPU* to parse elapsed ratio is calculated using the formula: *(Parse CPU time /Parse elapsed*

Measurement	Description	Measurement Unit	Interpretation
			*time)*100* Ideally, Parse elsapsed must be equal to *Parse CPU* - i.e., only CPU time should be used for parsing. In that case the ratio will be 100%. However, if wait time is more then this ratio will be less. A low value for this ratio is an indicator of latching problems. Investigate the latch sections in AWR and Statspack report for contention on library cache and shared pool latches.
CPU to non-parse:	Indicates the percentage of CPU time spent for activities other than parsing the SQLs.	Percent	The closer the value of this measure is to 100, better will be the performance of the server. This is because, such a value means that your CPU works on executing your queries instead of parsing them.
Hard parse ratio:	Indicates the percentage of hard parses.	Percent	Hard parsing happens when the oracle server parses a query and cannot find an exact match for the query in the library cache. A hard parse is a very expensive operation both in terms of CPU used and in the number of latches that gets performed. This is why, the value of this measure should be very low. One of the common reasons for high hard parse ratio is the inefficient sharing of SQL statements.
SGA usage:	Indicates the	Percent	The *SGA_ TARGET_ SIZE* is the

Measurement	Description	Measurement Unit	Interpretation
	percentage of the target SGA size that is in use currently.		total size of all SGA components. You can use this measure to know how much of the configured target SGA size is being used. If this value is close to 100%, it is a cause for concern, as it indicates that the SGA is about to run out of memory. This in turn can slow down user accesses and query execution. In such a scenario, you can use the detailed diagnosis of the Current size measure to know the memory usage of the individual SGA components. In the process, you can identify the exact SGA component that is over-utilizing the memory resources.

3.3.22 Oracle RAC Rollbacks Test

The immediate availability of rollback segments for the various activities that occur in a database server is very critical. Contention for rollback segments can adversely impact the performance of a database server and hence, needs to be detected and reported immediately. To detect contention for rollback segments on each node in an Oracle cluster, the Oracle RAC Rollbacks test monitors every cluster node for the degree of contention for buffers that contain rollback segment blocks.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each node in the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured

3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

   alter session set container=<**Oracle_service_name**>*;*

   create user <**user_name**>*identified by* <**user_password**> *container=current default tablespace* <**name_of_default_tablespace**> *temporary tablespace* <**name_of_temporary_tablespace**>*;*

   Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
System segment waits:	Denotes the ratio of the number of waits for acquiring a header block or a block of the SYSTEM rollback segment to the total number of requests for data to this cluster node, measured over a period of time.	Percent	If the number of waits for any class of block exceeds 1% of the total number of requests, the size of the SYSTEM rollback segment needs to be increased.
Non- system segment waits:	Denotes the ratio of the number of waits for acquiring a header block or any other block of a non- SYSTEM rollback segment to the total number of requests for data to this cluster node, measured over a period of time.	Percent	If the number of waits for any class of block exceeds 1% of the total number of requests, the sizes of the existing rollback segments may need to be increased. Alternatively, additional rollback segments to may be created to reduce contention.

3.3.23 Oracle RAC SQL Network Test

Using the JDBC API, this test reports the availability and responsiveness of each node in the cluster, and collects statistics pertaining to the traffic into and out of every node.

Target of the test : Oracle Cluster

Agent deploying the test : An external agent; if you are running this test using the external agent on the eG manager box, then make sure that this external agent is able to communicate with the port on which the target Oracle server is listening. Alternatively, you can deploy the external agent that will be running this test on a host that can access the port on which the target Oracle server is listening.

Outputs of the test : One set of results for each node in the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the

select_catalog_role and *create session* privileges.

The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;*

create user **<user_name>***identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
10. **INDIVIDUAL NODE**–By default, this flag is set to **Yes**, indicating that this test will report metrics for every node in the cluster by default. You can set this flag to **No** to ensure that the test reports the availability and responsiveness of the cluster service as a whole, and not the individual cluster nodes.
11. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

- The eG manager license should allow the detailed diagnosis capability
- Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Oracle cluster node availability:	Whether the cluster node is responding to requests.	Percent	The availability is 100% when a cluster node is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the node, or because the node is using an invalid user account. Besides the above, this measure will report that the server is unavailable even if a connection to the node is unavailable, or if a query to the node fails. In this case, you can check the values of the DB connection availability and Query processor availability measures to know what is exactly causing the node to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?
Total response time:	The time taken by this node to respond to a user query. This is the sum total of the connection time and query execution time.	Secs	A sudden increase in response time is indicative of a bottleneck at the node. This could even be owing to a connection delay and/or long running queries to the node. Whenever the value of this measure is high, it would be good practice to compare the values of the Connection time and

Measurement	Description	Measurement Unit	Interpretation
			Query execution time measures for a node to zero-in on the root-cause of the poor responsiveness of the server - is it because of connectivity issues? or is it because of inefficient queries?
Data transmit rate:	The rate of data being transmitted by this node in response to client requests.	KB/Sec	The data transmission rate reflects the workload on the server.
Data receive rate:	The rate of data received by this node from clients over SQL*Net.	KB/Sec	This measure also characterizes the workload on a node. As the data rate to a node increases, consider tuning the Service Layer Data Buffer (SDU) and the Transport Layer Data Buffer (BDU) in the TNSNames.ora and Listener.ora files to optimize packet transfers across the network.
Cluster node connection availability:	Indicates whether the database connection to this node is available or not.	Percent	If this measure reports the value 100, it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the Oracle server availability measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of a connection to the server.
Connection time	Indicates the time	Secs	A high value could indicate a

Measurement	Description	Measurement Unit	Interpretation
to cluster node:	taken to connect to the cluster node.		connection bottleneck. Whenever the Total response time of the measure soars, you may want to check the value of this measure to determine whether a connection latency is causing the poor responsiveness of the node.
Query processor availability:	Indicates whether the query to this node is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the Oracle server availability measure reports the value 0, check the value of this measure to figure out whether the failed query is the reason why that measure reported a node unavailability.
Query execution time:	Indicates the time taken for query execution.	Secs	A high value could indicate that one/more queries to the node are taking too long to execute. Inefficient/badly designed queries to the database often take too long to execute. If the value of this measure is higher than that of the Connection time measure, you can be rest assured that long running queries are causing the node to respond slowly to requests.

3.3.24 Oracle RAC User Waits Test

When Oracle executes an SQL statement, it is not constantly executing. Sometimes it has to wait for a specific event to happen before it can proceed. For example, if Oracle (or the SQL statement) wants to modify data, and the corresponding database block is not currently in the SGA, Oracle waits for this block to be available for modification. Every such wait event belongs to a class of wait events. The following list describes each of the wait classes.

Wait Class	Description
Administrative	Waits resulting from DBA commands that cause users to wait (for example, an index rebuild)
Application	Waits resulting from user application code (for example, lock waits caused by row level locking or explicit lock commands)
Cluster	Waits related to Real Application Cluster resources (for example, global cache resources such as 'gc cr block busy'
Commit	This wait class only comprises one wait event - wait for redo log write confirmation after a commit (that is, 'log file sync')
Concurrency	Waits for internal database resources (for example, latches)
Configuration	Waits caused by inadequate configuration of database or instance resources (for example, undersized log file sizes, shared pool size)
Idle	Waits that signify the session is inactive, waiting for work (for example, 'SQL*Net message from client')
Network	Waits related to network messaging (for example, 'SQL*Net more data to dblink')
Other	Waits which should not typically occur on a system (for example, 'wait for EMON to spawn')
Scheduler	Resource Manager related waits (for example, 'resmgr: become active')
System I/O	Waits for background process IO (for example, DBWR wait for 'db file parallel write')
User I/O	Waits for user IO (for example 'db file sequential read')

Since wait events are resource-drains and serious performance degraders, administrators need to keep a close eye on these wait classes, figure out how much time the Oracle cluster actually spends waiting for each class, and rapidly decipher why, so that measures can be initiated to minimize these events. To achieve this, you can use the **Oracle RAC User Waits** test. This test reports the time spent by the nodes in the cluster waiting for events of each wait class, helps identify those wait classes with wait events that have remained active for a long time, and also reveals the number of sessions that have been impacted by the waiting. With the help of the detailed diagnostics of this test, you can also zoom into these sessions and identify the queries that they executed that may have caused wait events to occur; this way, inefficient queries can be isolated.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each wait class active on the cluster nodes of the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

   grant select_catalog_role to oratest;

   grant oratest to oraeg;

   The sample script we recommend for user creation (in Oracle database server 12c) for eG

monitoring is:

alter session set container=**<Oracle_service_name>**;

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>**;

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>**;

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active sessions:	Indicates the current number of sessions in which events of this wait class are currently active.	Number	A high value indicates that too many sessions are waiting owing to the events of a particular wait class. To know more about these sessions, the wait events that each session triggered, and which query triggered the events, use the detailed diagnosis of this measure. With the help of the detailed metrics, you can quickly isolate the queries that require optimization.
Max wait time:	Indicates the maximum time for which the Oracle server has waited for events of this wait class.	Secs	A high value is indicative of the following: • An increase in load (either more users, more calls, or larger

Measurement	Description	Measurement Unit	Interpretation
			transactions) • I/O performance degradation (I/O time increases and wait time increases, so DB time increases) • Application performance degradation • CPU- bound host (foregrounds accumulate active run- queue time, wait event times are artificially inflated) Compare the value of this measure across wait classes to identify which wait class has caused the Oracle database server to wait for the maximum time. You can then use the detailed diagnostics reported by the Active sessions measure to identify which sessions were impacted, and what queries were executed by those sessions to increase wait time. Inefficient queries can thus be identified and optimized to ensure that waiting is eliminated or at least minimized.

3.3.25 Oracle RAC SQL Workload Test

Nothing can degrade the performance of an Oracle cluster like a resource-hungry or a long-running query! When such queries execute on the cluster, they either hog almost all the available CPU, memory, and disk resources of the cluster or keep the resources locked for long time periods, thus leaving little to no resources for carrying out other critical cluster operations. This can significantly slowdown the cluster and adversely impact user experience with the cluster. To ensure peak performance of the Oracle cluster at all times, such queries should be rapidly identified and quickly optimized to minimize resource usage. This is where the **Oracle SQL Workload** test helps.

At configured intervals, this test compares the usage levels and execution times of all queries that started running on the cluster in the last measurement period and identifies a 'top query' in each of the following categories - CPU usage, memory usage, disk activity, and execution time. The test then reports the resource usage and execution time of the top queries and promptly alerts administrators if any query consumes more resources or takes more time to execute than it should. In such a scenario, administrators can use the detailed diagnosis of this test to view the inefficient queries and proceed to optimize them to enhance cluster performance.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each node in the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.

The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

create user oraeg identified by oraeg create role oratest;

grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

alter session set container=**<Oracle_service_name>***;***

create user **<user_name>***identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
10. **DDCOUNT** – By default, the detailed diagnosis of this test reports the top-5 queries in resource usage and execution time. This is why, the **DDCOUNT** parameter is set to 5 by default. If you want detailed diagnosis to display less or more number of top queries, then change the **DDCOUNT**.
11. To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the

following conditions are fulfilled:

- The eG manager license should allow the detailed diagnosis capability
- Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Top most query physical reads:	Indicates the number of physical disk reads performed by the top query per execution.	Reads/execution	If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries generating maximum physical disk activity. From this, you can identify the top query in terms of number of physical disk reads. You may then want to optimize the query to reduce the disk reads.
Top most buffer gets:	Indicates the number of memory buffers used by the top query per execution.	Memorybuffergets/execution	If the value of this measure is abnormally high, you can use the detailed diagnosis of this measure to view the top-5 (by default) queries consuming memory excessively. From this, you can easily pick that query which is consuming the maximum memory. You may then want to optimize the query to minimize memory usage.
	Indicates the	Secs/execution	If the value of this measure

Measurement	Description	Measurement Unit	Interpretation
Top most query CPU time:	duration for which each execution of the top query was hogging the CPU resources.		is over 30 seconds, you can use the detailed diagnosis of this measure to the top-5 (by default) queries hogging the CPU resources. From this, you can easily pick that query which is consuming the maximum CPU. You may then want to optimize the query to minimize CPU usage.
Top most query elapsed time:	Indicates the running time of each execution of the top query.	Secs/execution	If the value of this measure crosses 10 seconds, you can use the detailed diagnosis of this measure to view the top- 5 (by default) queries that are taking too long to execute. . From this, you can easily pick that query with the maximum execution time. You may then want to optimize the query to minimize execution time.

3.3.26 Oracle RAC Uptime Test

In most production environments, it is essential to monitor the uptime of critical servers in the infrastructure. By tracking the uptime of each of the servers, administrators can determine what percentage of time a server has been up. Comparing this value with service level targets, administrators can determine the most trouble-prone areas of the infrastructure.

In some environments, administrators may schedule periodic reboots of their servers. By knowing that a specific server has been up for an unusually long time, an administrator may come to know that the scheduled reboot task is not working on a server.

The **Oracle RAC Uptime** Test monitors the uptime of every node in an Oracle cluster.

Target of the test : Oracle Cluster

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each node in the Oracle cluster being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **ORASID** - The variable name of the oracle instance
5. **SERVICE NAME** - A **ServiceName** exists for the entire Oracle RAC system. When clients connect to an Oracle cluster using the **ServiceName**, then the cluster routes the request to any available database instance in the cluster. By default, the **SERVICE NAME** is set to *none*. In this case, the test connects to the cluster using the **ORASID** and pulls out the metrics from that database instance which corresponds to that **ORASID**. If a valid **SERVICE NAME** is specified instead, then, the test will connect to the cluster using that **SERVICE NAME**, and will be able to pull out metrics from any available database instance in the cluster.

   To know the **ServiceName** of a cluster, execute the following query on any node in the target cluster:

   select name, value from v$parameter where name ='service_names'

6. **USER** – In order to monitor an Oracle RAC, a special database user account has to be **USER** – In order to monitor an Oracle database server, a special database user account has to be created in every Oracle database instance that requires monitoring. A **Click here** hyperlink is available in the test configuration page, using which a new oracle database user can be created. Alternatively, you can manually create the special database user. When doing so, ensure that this user is vested with the *select_catalog_role* and *create session* privileges.
   The sample script we recommend for user creation (in Oracle database server versions before 12c) for eG monitoring is:

   create user oraeg identified by oraeg create role oratest;

   grant create session to oratest;

grant select_catalog_role to oratest;

grant oratest to oraeg;

The sample script we recommend for user creation (in Oracle database server 12c) for eG monitoring is:

*alter session set container=***<Oracle_service_name>***;*

create user **<user_name>** *identified by* **<user_password>** *container=current default tablespace* **<name_of_default_tablespace>** *temporary tablespace* **<name_of_temporary_tablespace>***;*

Grant create session to <user_name>;

Grant select_catalog_role to **<user_name>***;*

The name of this user has to be specified here.

7. **PASSWORD** – Password of the specified database user
8. **CONFIRM PASSWORD** – Confirm the **PASSWORD** by retyping it here.
9. **ISPASSIVE** – If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
10. **REPORTMANAGERTIME** – By default, this flag is set to **Yes**, indicating that, by default, the detailed diagnosis of this test, if enabled, will report the shutdown and reboot times of the device in the manager's time zone. If this flag is set to **No**, then the shutdown and reboot times are shown in the time zone of the system where the agent is running (i.e., the system being managed for agent-based monitoring, and the system on which the remote agent is running - for agentless monitoring).

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Has Oracle server been restarted?:	Indicates whether this node has been rebooted during the last measurement period or not.	Boolean	If this measure shows 1, it means that the node was rebooted during the last measurement period. By checking the time periods when this metric changes from 0 to 1, an administrator can determine the

Measurement	Description	Measurement Unit	Interpretation
			times when this node was rebooted. The detailed diagnosis of this measure, if enabled, indicates the date/time at which the node was shutdown, the date on which it was restarted, the duration of the shutdown, and whether the node was shutdown as part of a maintenance outine.
Uptime since the last measurement:	Indicates the time period that this node has been up since the last time this test ran.	Secs	If the node has not been rebooted during the last measurement period and the agent has been running continuously, this value will be equal to the measurement period. If the node was rebooted during the last measurement period, this value will be less than the measurement period of the test. For example, if the measurement period is 300 secs, and if the node was rebooted 120 secs back, this metric will report a value of 120 seconds. The accuracy of this metric is dependent on the measurement period – the smaller the measurement period, greater the accuracy.
Uptime:	Indicates the total time that the node has been up since its last reboot.	Mins	Administrators may wish to be alerted if a node has been running without a reboot for a very long period. Setting a threshold for this metric allows administrators to determine such conditions.

About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.