Monitoring Cassandra Database

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

Table of Figures

Chapter 1: Introduction

Apache Cassandra™ is a massively scalable open source NoSQL database. Cassandra is perfect for managing large amounts of structured, semi-structured, and unstructured data across multiple datacenters and the cloud. Cassandra delivers continuous availability, linear scalability, and operational simplicity across many commodity servers with no single point of failure, along with a powerful dynamic data model designed for maximum flexibility and fast response times.

Cassandra's built-for-scale architecture means that it is capable of handling petabytes of information and thousands of concurrent users/operations per second. Cassandra is designed from the ground up as a distributed database with peer-to-peer communication.

For ensuring high availability, performance, and security, a database server includes a wealth of data storage, caching, and retrieval functions. To ensure peak performance, a database server needs to be continuously monitored and tuned. Sometimes, there may be a sudden change in workload to the database, resulting in an increase in the number of simultaneously processed transactions. This scenario could result in a performance bottleneck at the database server. Continuous monitoring and optimization of the database server is essential for ensuring that the database server operates at its peak.

The eG Enterprise suite is programmed with a variety of tests that are designed to monitor the critical parameters of the Cassandra Database servers. This document describes how the eG Enterprise suite monitors the Cassandra Database servers.

Chapter 2: How Does eG Enterprise Monitor Cassandra Database?

eG Enterprise can monitor the Cassandra Database in an agent-based or an agentless manner. In case of the agentless approach, the remote agent used to monitor the Cassandra Database should be deployed on a remote Windows host in the environment.

2.1 Pre-Requisites for Monitoring the Cassandra Database

- Regardless of the approach (agent-based or agentless), the eG agent should be configured to connect to the JRE used by the Cassandra Database so that the eG agent can pull out the required metrics, using JMX methodology.
- By default, JMX requires no authentication or security (SSL). In this case therefore, to use JMX for pulling out metrics from the Cassandra Database, the following needs to be done:
  - Login to the Cassandra Database host.
  - The **<CASSANDRA_HOME>\conf** folder used by the target application will typically contain the following files:
    - cassandra-env.ps1 or cassandra-env.sh
    - jmxremote.password
    - jmxremote.access
  - Edit the cassandra-env.ps1 or cassandra-env.sh file and set the port number at which the JMX listens to. If the JMX listens to port 7199, then set the port number as 7199.
  - Then save the file.
  - Next, during test configuration, do the following:
    - Set JMX as the mode;
    - Set the port that you defined in step 3 above (in the cassandra-env.ps1 or cassandra-env.sh file) as the jmx remote port;
    - Set the user and password parameters to none.
    - Update the test configuration.

2.2 Enabling JMX Support for JRE

In older versions of Java (i.e., JDK/JRE 1.1, 1.2, and 1.3), very little instrumentation was built in, and custom-developed byte-code instrumentation had to be used to perform monitoring. From JRE/JDK 1.5 and above however, support for Java Management Extensions (JMX) were pre-built into JRE/JDK. JMX enables external programs like the eG agent to connect to the JRE of an application and pull out metrics in real-time.

By default, JMX requires no authentication or security (SSL). In this case therefore, to use JMX for pulling out metrics from a target application, the following needs to be done:

1. Login to the application host.
2. The **<CASSANDRA_HOME>\conf** folder used by the target Cassandra Database node will typically contain the following file:
   - cassandra-env.sh if the target Cassandra Database node is installed on a Windows host or cassandra-env.ps1 if the target Cassandra Database node is installed on a Unix host
3. If the Cassandra Database node is installed on a Windows host, edit the cassandra-env.ps1 file, and append the following lines to it:

   ```
   $env:JVM_OPTS="$env:JVM_OPTS -Djava.rmi.server.hostname=<IP_ADDRESS>"
   $JMX_PORT="<Port No>"
   ```

   For instance, if the JMX listens on port 7199, then the second line of the above specification would be:

   ```
   $JMX_PORT="7199"
   ```

   If the Cassandra Database node is installed on a Unix host, edit the cassandra-env.sh file, and append the following lines to it:

   ```
   JVM_OPTS="$JVM_OPTS -Djava.rmi.server.hostname=<IP_ADDRESS>"
   JMX_PORT="<Port No>"
   ```

   For instance, if the JMX listens on port 7199, then the second line of the above specification would be:

   ```
   JMX_PORT="7199"
   ```

4. Then, save the file.
5. In the first line, set the <IP Address> to the IP address using which the Cassandra Database

node has been managed in the eG Enterprise system. Alternatively, you can add the following line to the startup script:

```
-Djava.rmi.server.hostname=localhost
```

6. Save this script file too.
7. Next, during test configuration, do the following:
   - Set the port that you defined in step 3 above (in the cassandra-env.ps1 file or cassandra-env.sh) as the JMX Remote Port;
   - Set the user and password parameters to *none*.
   - Update the test configuration.

On the other hand, if JMX requires only authentication (and no security), then the following steps will apply:

1. Login to the application host. If the target Cassandra Database node is installed on a Windows host, then, login to the host as a local/domain administrator.
2. The <JAVA_HOME>\jre\lib\management folder used by the target Cassandra Database node will typically contain the following files:
   - management.properties
   - jmxremote.access
   - jmxremote.password.template
3. First, copy the jmxremote.password.template file to any other location on the host, rename it as jmxremote.password, and then, copy it back to the **<JAVA_HOME>\jre\lib\management** folder. Ensure that the jmxremote.password and the jmxremote.access files are copied to the **<CASSANDRA_HOME>**folder.
4. Next, edit the jmxremote.password file and the jmxremote.access file copied to the <CASSANDRA_HOME> to create a user with read-write access to the JMX. To know how to create such a user, refer to Section **2.2**.
5. Then, proceed to make the jmxremote.password file secure by granting a single user “full access” to that file. For monitoring applications executing on Windows in particular, only the Owner of the jmxremote.password file should have full control of that file. To know how to grant this privilege to the Owner of the file, refer to Section **2.2**.
6. In case of applications executing on Linux hosts on the other hand, any user can be granted full

access to the jmxremote.password file, by following the steps below:

- Login to the host as the user who is to be granted full control of the jmxremote.password file.
- Issue the following command:

  chmod 600 jmxremote.password

- This will automatically grant the login user full access to the jmxremote.password file.

7. The **<CASSANDRA_HOME>\conf** folder used by the target Cassandra Database node will typically contain the following file:

   - cassandra-env.sh if the target Cassandra Database node is installed on a Windows host or cassandra-env.ps1 if the target Cassandra Database node is installed on a Unix host

8. If the Cassandra Database node is installed on a Windows host, edit the cassandra-env.ps1 file, and append the following lines to it:

```
$env:JVM_OPTS="$env:JVM_OPTS -Djava.rmi.server.hostname=<IP_ADDRESS>"
$JMX_PORT="<Port No>"
```

   For instance, if the JMX listens on port 7199, then the second line of the above specification would be:

```
$JMX_PORT="7199"
```

   If the Cassandra Database node is installed on a Unix host, edit the cassandra-env.sh file, and append the following lines to it:

```
JVM_OPTS="$JVM_OPTS -Djava.rmi.server.hostname=<IP_ADDRESS>"
JMX_PORT="<Port No>"
```

   For instance, if the JMX listens on port 7199, then the second line of the above specification would be:

```
JMX_PORT="7199"
```

9. In the first line of the script mentioned in step 8, set the <IP Address> to the IP address using which the Cassandra Database node has been managed in the eG Enterprise system. Alternatively, you can add the following line to the startup script of the target Cassandra Database node:

```
-Djava.rmi.server.hostname=localhost
```

10. Next, edit the cassandra-env.ps1 file, and append the following lines to it:

```
$env:JVM_OPTS="$env:JVM_OPTS -Dcom.sun.management.jmxremote.authenticate=true"
```

```
$env:JVM_OPTS="$env:JVM_OPTS -Dcom.sun.management.jmxremote.password.file=<Path of jmxremote.access>"
```

```
$env:JVM_OPTS="$env:JVM_OPTS -Dcom.sun.management.jmxremote.access.file=<Path of jmxremote.password>"
```

For instance, assume that the the jmxremote.access and jmxremote.password files exist in the following directory on a Windows host: <CASSANDRA_HOME>. The specification above will then read as follows:

```
$env:JVM_OPTS="$env:JVM_OPTS -Dcom.sun.management.jmxremote.authenticate=true"
```

```
$env:JVM_OPTS="$env:JVM_OPTS -Dcom.sun.management.jmxremote.password.file=<CASSANDRA_HOME>\jmxremote.password"
```

```
$env:JVM_OPTS="$env:JVM_OPTS -Dcom.sun.management.jmxremote.access.file=<CASSANDRA_HOME>\jmxremote.access"
```

11. If the application in question is executing on a Unix/Linux host, and the jmxremote.access and jmxremote.password files reside in the **<CASSANDRA_HOME>** folder of the host, then the specification will be:

```
JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.authenticate=true"
```

```
JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.password.file=<CASSANDRA_HOME>\jmxremote.password"
```

```
JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.access.file=<CASSANDRA_HOME>\jmxremote.access"
```

12. Finally, save the file.
13. Next, during test configuration, do the following:
    - Ensure that the port number configured at step 8 above is set as the JMX Remote Port;
    - Make sure that the user and password parameters of the test are that of a user with readwrite rights to JMX. To know how to create a new user and assign the required rights to him/her, refer to Section **2.3**.

2.3 Configuring the eG Agent to Support JMX Authentication

By default, the eG agent uses JMX for monitoring the Cassandra Database, and this JMX requires **authentication only** (and not security). Therefore, every test to be executed by the eG agent should be configured with the credentials of a valid user to JMX, with read-write rights. The steps for creating such a user are detailed below:

1. Login to the application host. If the application being monitored is on a Windows host, then login as a local/domain administrator to the host.
2. Go to the **<CASSANDRA_HOME>\conf** folder used by the target application to view the following files:
   - cassandra-env.sh or cassandra-env.ps1
   - jmxremote.access
   - jmxremote.password.template
3. Copy the jmxremote.password.template file to a different location, rename it as jmxremote.password, and copy it back to the **<CASSANDRA_HOME>\conf** folder.
4. Open the jmxremote.password file and scroll down to the end of the file. By default, you will find the commented entries indicated by Figure 2.1 below:

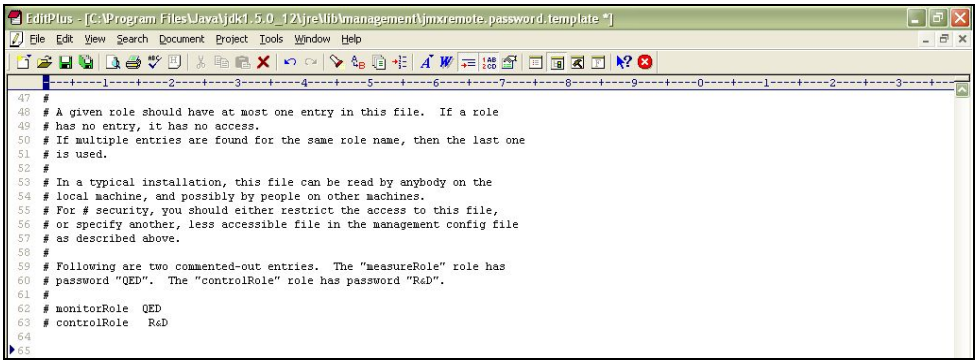


Figure 2.1: Scrolling down the jmxremote.password file to view 2 commented entries

5. The two entries indicated by Figure 2.1 are sample username password pairs with access to JMX. For instance, in the first sample entry of Figure 2.1, monitorRole is the username and QED is the password corresponding to monitorRole. Likewise, in the second line, the controlRole user takes the password R&D.
6. If you want to use one of these pre-defined username password pairs during test configuration, then simply uncomment the corresponding entry by removing the # symbol preceding that entry. However, prior to that, you need to determine what privileges have been granted to both these users. For that, open the jmxremote.access file in the editor.

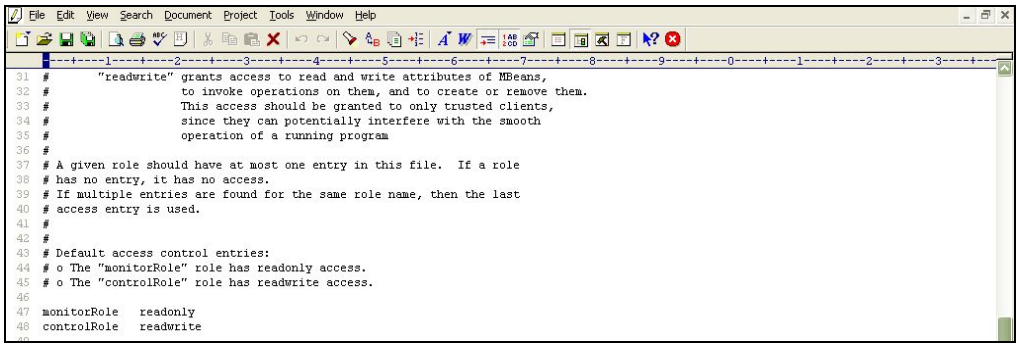


Figure 2.2: The jmxremote.access file

7. Scrolling down the file (as indicated by Figure 2.2) will reveal 2 lines, each corresponding to the sample username available in the jmxremote.password file. Each line denotes the access rights of the corresponding user. As is evident from Figure 2.2, the user monitorRole has only readonly rights, while user controlRole has readwrite rights. Since the eG agent requires readwrite rights to be able to pull out key JVM-related statistics using JMX, we will have to configure the test with the credentials of the user controlRole.

8. For that, first, edit the jmxremote.password file and uncomment the controlRole <password> line as depicted by Figure 2.3.

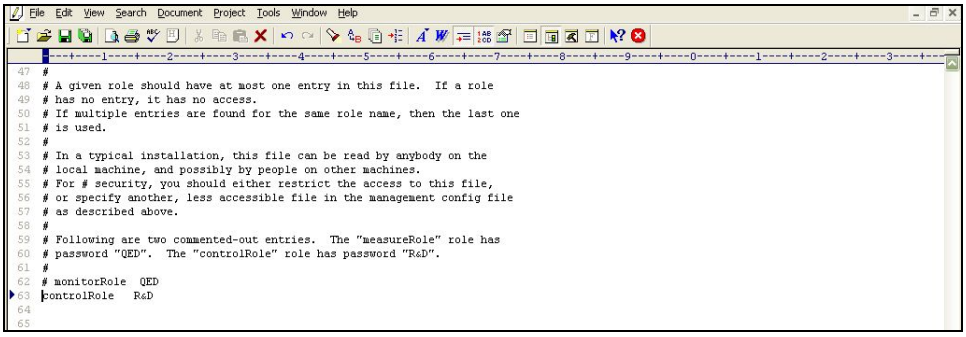


Figure 2.3: Uncommending the 'controlRole' line

9. Then, save the file. You can now proceed to configure the tests with the user name controlRole and password R&D.

10. Alternatively, instead of going with these default credentials, you can create a new username password pair in the jmxremote.password file, assign readwrite rights to this user in the jmxremote.access file, and then configure the eG tests with the credentials of this new user. For instance, let us create a user john with password john and assign readwrite rights to john.

11. For this purpose, first, edit the *jmxremote.password* file, and append the following line (see

Figure 2.4) to it:

john john

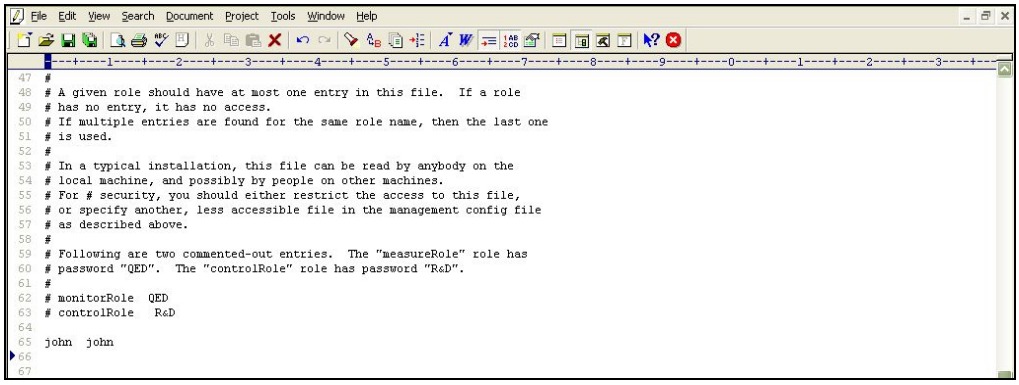


Figure 2.4: Appending a new username and password pair

12. Save the jmxremote.password file.
13. Then, edit the jmxremote.access file, and append the following line (see Figure 2.5) to it:

    john readwrite

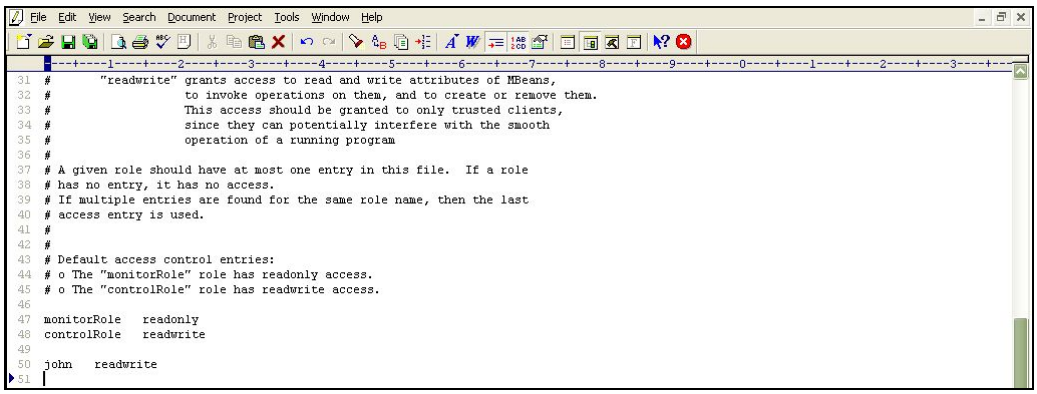


Figure 2.5: Assigning rights to the new user in the jmxremote.access file

14. Then, save the jmxremote.access file.
15. Finally, proceed to configure the tests with the user name and password, john and john, respectively.

2.3.1 Securing the 'jmxremote.password' file

To enable the eG agent to use JMX (that requires **authentication only**) for monitoring the Cassandra Database, you need to ensure that the jmxremote.password file in the **<CASSANDRA_**

HOME>\conf folder used by the target application is accessible **only by the Owner of that file**. To achieve this, do the following:

1. Login to the Windows host as a local/domain administrator.
2. Browse to the location of the jmxremote.password file using Windows Explorer.
3. Next, right-click on the jmxremote.password file and select the **Properties** option (see Figure 2.6).

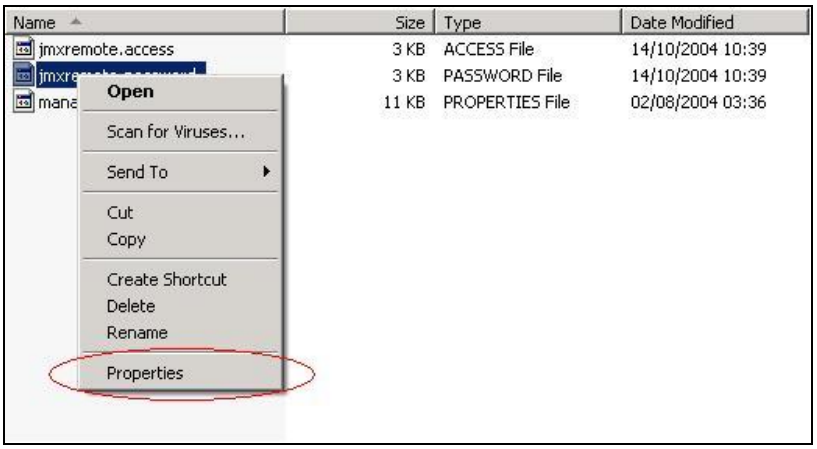


Figure 2.6: Selecting the Properties option

4. From Figure 2.7 that appears next, select the **Security** tab.

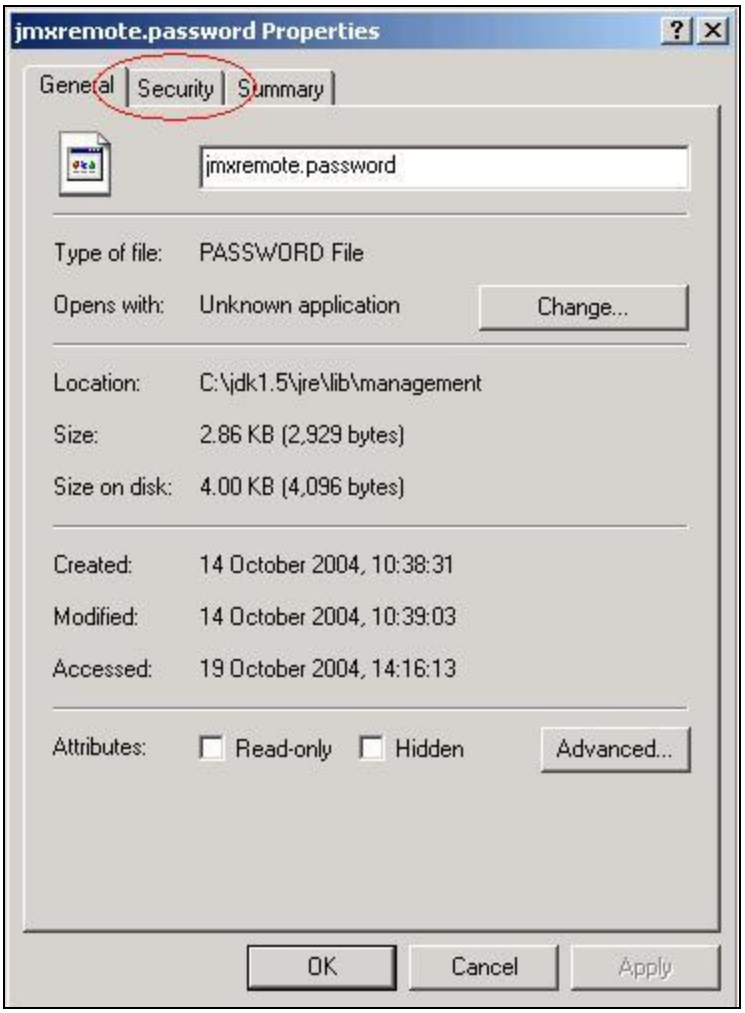


Figure 2.7: The Properties dialog box

However, if you are on a Windows computer that is not part of a domain, then the **Security** tab may be missing. To reveal the **Security** tab, do the following:

- Open Windows Explorer, and choose **Folder Options** from the **Tools** menu.
- Select the **View** tab, scroll to the bottom of the **Advanced Settings**section, and clear the check box

next to **Use Simple File Sharing**.

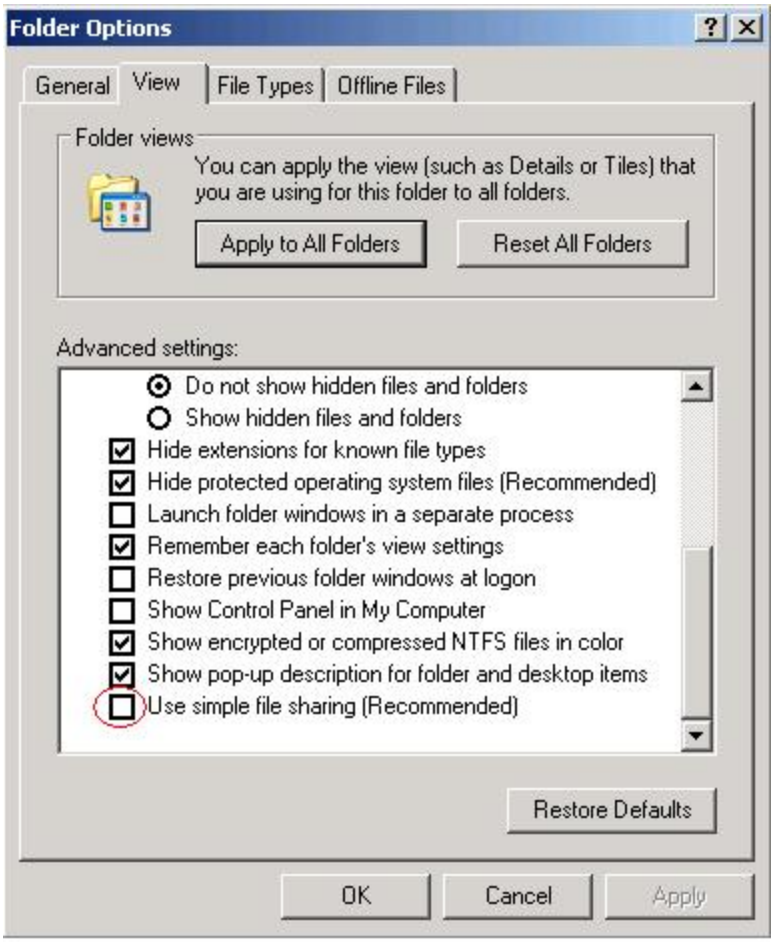


Figure 2.8: Deselecting the 'Use simple file sharing' option

- Click **OK** to apply the change
- When you restart Windows Explorer, the **Security** tab would be visible.

5. Next, select the **Advanced** button in the **Security** tab of Figure 2.9.

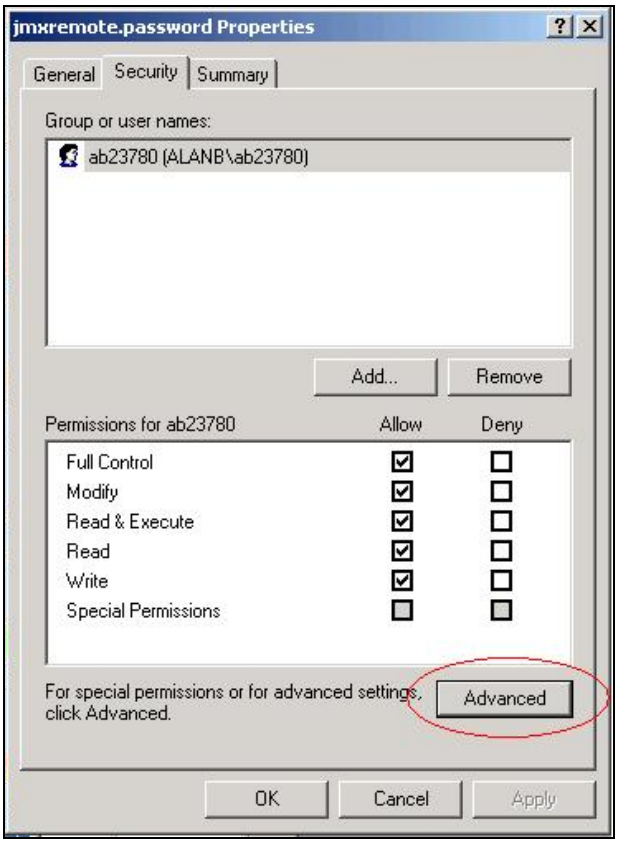


Figure 2.9: Clicking the Advanced button

6. Select the **Owner** tab to see who the owner of the file is.

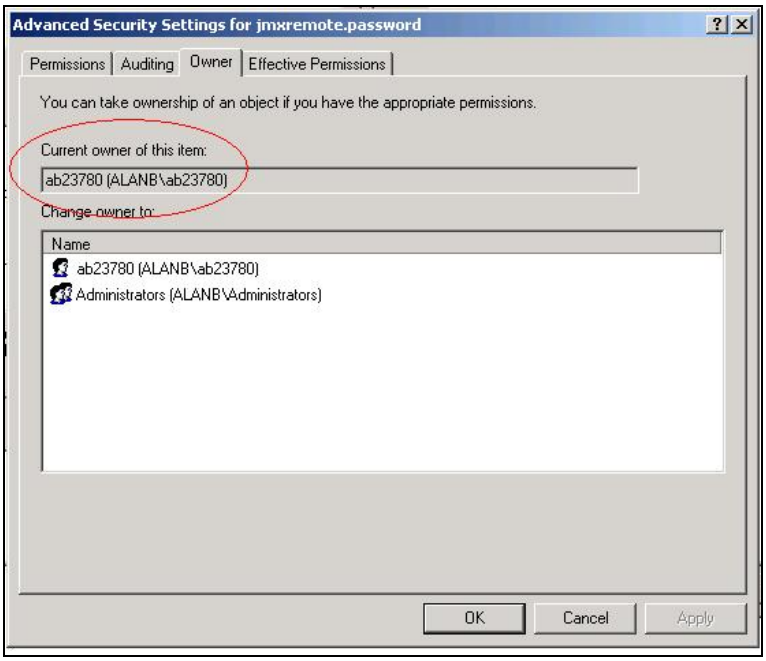


Figure 2.10: Verifying whether the Owner of the file is the same as the application Owner

7. Then, proceed to select the **Permissions** tab in Figure 2.10 to set the permissions. If the jmxremote.password file has inherited its permissions from a parent directory that allows users or groups other than the **Owner** to access the file, then clear the **Inherit from parent the permission entries that apply to child objects** check box in Figure 2.11.

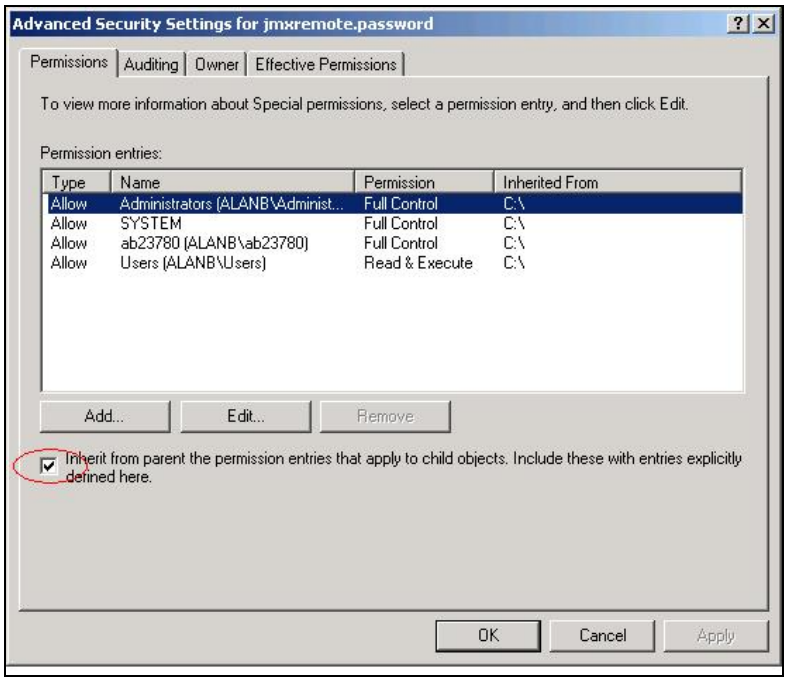


Figure 2.11: Disinheriting permissions borrowed from a parent directory

8. At this point, you will be prompted to confirm whether the inherited permissions should be copied from the parent or removed. Press the **Copy** button in Figure 2.12.

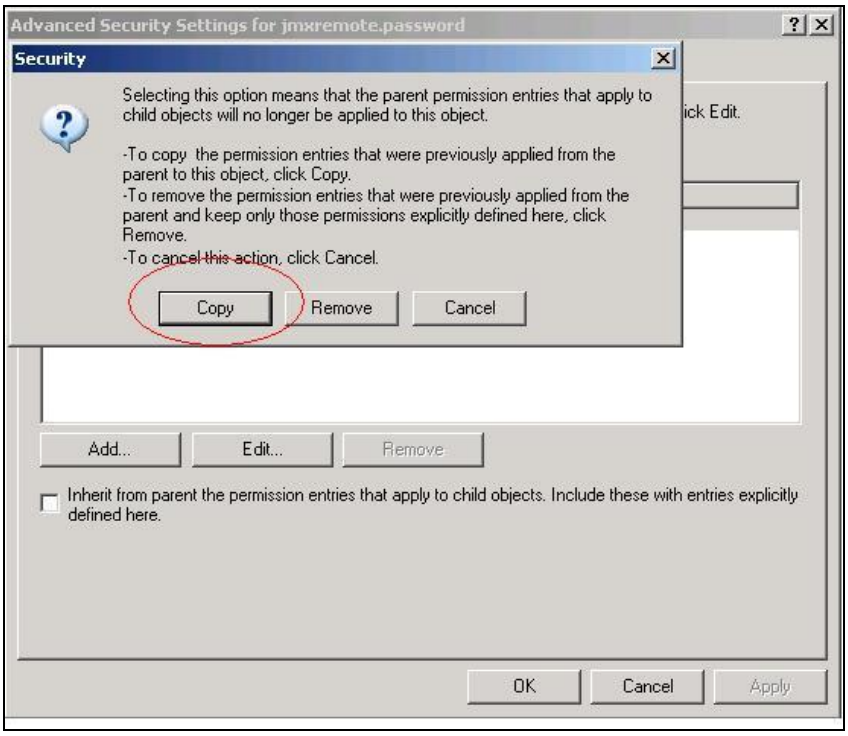


Figure 2.12: Copying the inherited permissions

9. Next, remove all permission entries that allow the jmxremote.password file to be accessed by users or groups other than the file **Owner**. For this, click the user or group and press the **Remove** button in Figure 2.13. At the end of this exercise, only a single permission entry granting **Full Control** to the owner should remain in Figure 2.13.

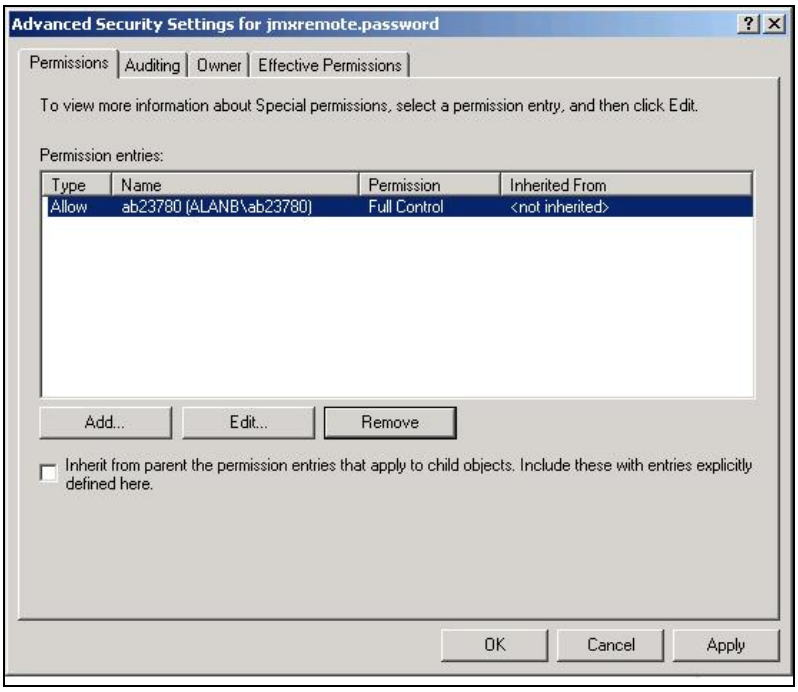


Figure 2.13: Granting full control to the file owner

10. Finally, click the **Apply** and **OK** buttons to register the changes. The password file is now secure, and can only be accessed by the file owner.

Note:

If you are trying to enable JMX on a Linux host, you might encounter issues with the way hostnames are resolved.

To solve it you might have to set the **-Djava.rmi.server.hostname=<hostname or localhost or ip>** property in the startup script of the target server.

If you are in local, simply try with **- Djava.rmi.server.hostname=localhost** or **-Djava.rmi.server.hostname=127.0.0.1**.

2.4 Managing the Cassandra Database

The Cassandra Database server cannot be automatically discovered by eG Enterprise. This implies that you need to manually add the database into the eG Enterprise system to manage it. Follow the steps below to achieve the same:

1. Follow the Components -> Add/Modify menu sequence in the **Infrastructure** tile of the **Admin** menu.

2. Next, select *Cassandra Database* from the **Component type** drop-down and then click the **Add New Component** button.
3. When Figure 2.14 appears, provide the **Host IP/Name** of the Cassandra Database that you want to manage.

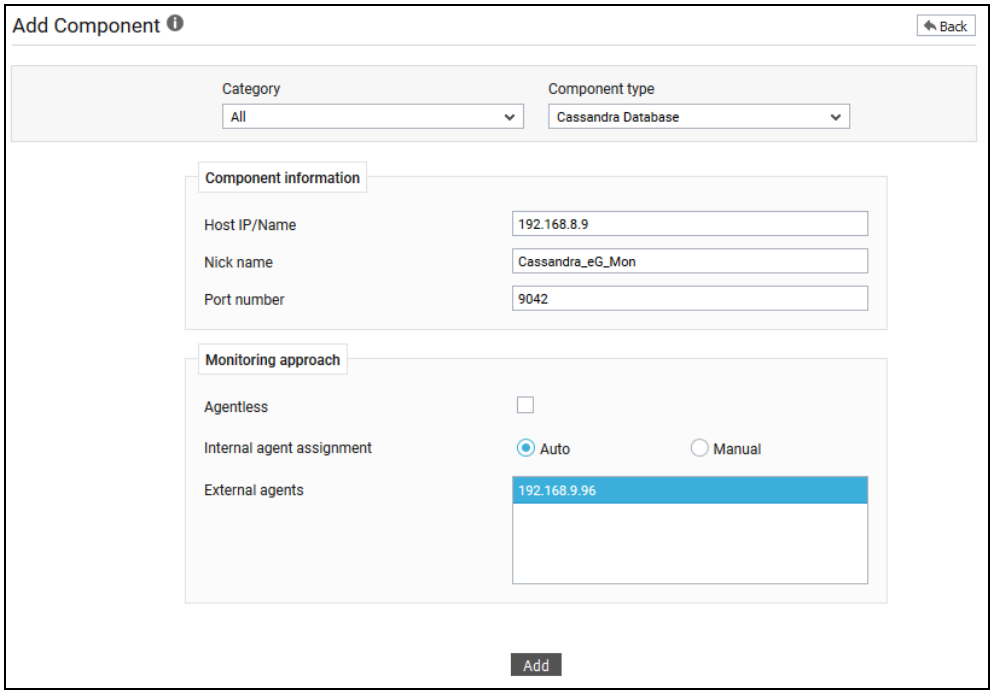


Figure 2.14: Managing a Cassandra Database server in an agent-based manner

4. Then, provide a **Nick name** for the server.
5. The **Port number** will be set as 9042 by default. If the Cassandra Database server is listening on a different port in your environment, then override this default setting.
6. In case you are monitoring a Cassandra Database server in an agent-based manner, just pick an external agent from the **External agents** list box and click the **Add** button to add the component for monitoring.
7. On the other hand, if you are monitoring a Cassandra Database server in an agentless manner, then do the following:

   - Select the **Agentless** check box.
   - Pick the **OS** on which the Cassandra Database server is running.
   - Set the **Mode** to **Other**.

- Select the **Remote agent** that will be monitoring the Cassandra Database server. **Note that the Remote agent you choose should run on a Windows host.**
- Choose an external agent for the server by picking an option from the **External agents** list box.
- Finally, click the **Add** button to add the Cassandra Database server for monitoring.

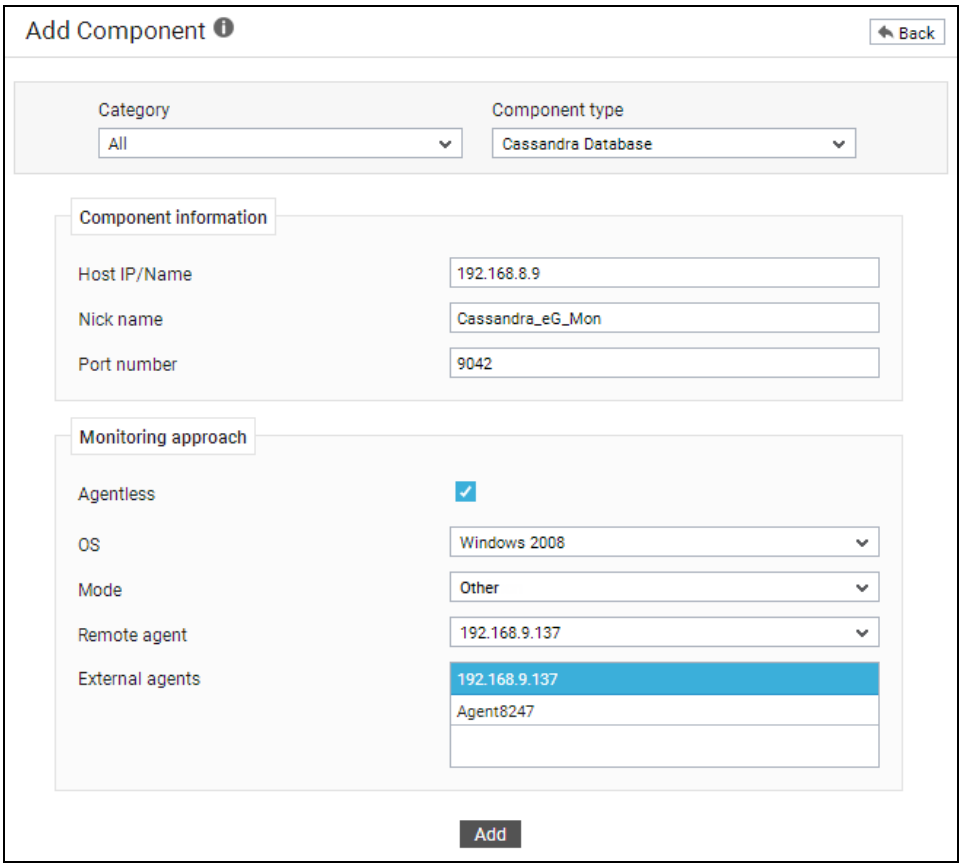
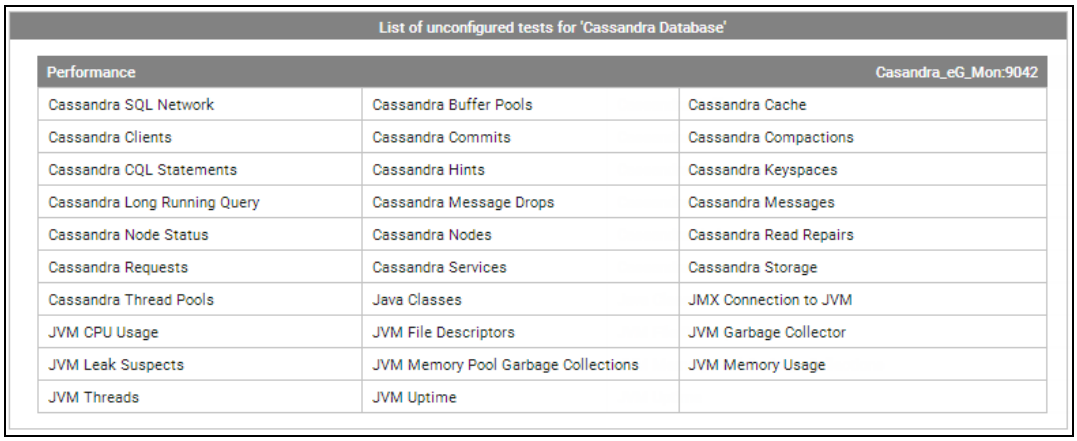


Figure 2.15: Managing a Cassandra Database server in an agentless manner

8. Now, try to sign out of the eG admin interface. Doing so, will reveal 2.4, which prompts you to configure a list of unconfigured tests for the newly added Cassandra Database.



List of unconfigured tests for 'Cassandra Database'

Performance		Casandra_eG_Mon:9042
Cassandra SQL Network	Cassandra Buffer Pools	Cassandra Cache
Cassandra Clients	Cassandra Commits	Cassandra Compactions
Cassandra CQL Statements	Cassandra Hints	Cassandra Keyspaces
Cassandra Long Running Query	Cassandra Message Drops	Cassandra Messages
Cassandra Node Status	Cassandra Nodes	Cassandra Read Repairs
Cassandra Requests	Cassandra Services	Cassandra Storage
Cassandra Thread Pools	Java Classes	JMX Connection to JVM
JVM CPU Usage	JVM File Descriptors	JVM Garbage Collector
JVM Leak Suspects	JVM Memory Pool Garbage Collections	JVM Memory Usage
JVM Threads	JVM Uptime	

Figure 2.16: The list of unconfigured tests for Cassandra Database

9. Click on any test in the list of unconfigured tests. For instance, click on the **Cassandra Buffer Pools** test to configure it. 2.4 then appears.

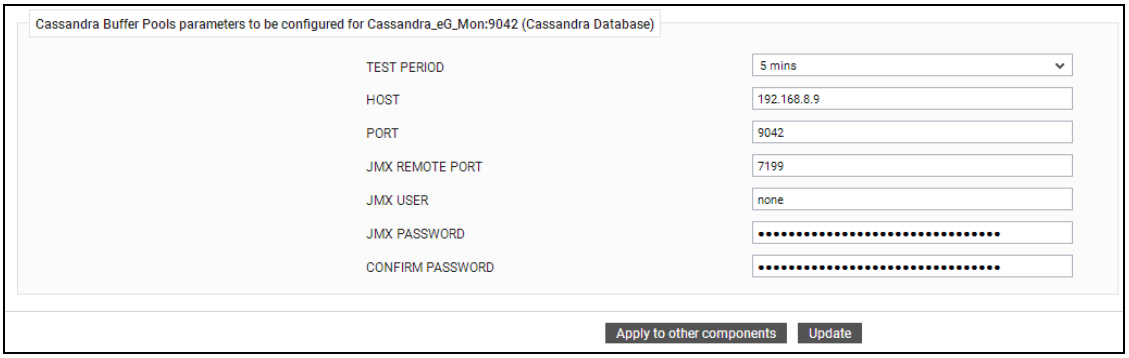


Figure 2.17: Configuring the Cassandra Buffer Pools test

10. To know how to configure the test, refer to **Monitoring the Cassandra Database**
11. Finally, click the **Signout** button at the right, top corner of the eG admin interface to sign out.

Chapter 3: Monitoring the Cassandra Database

eG Enterprise provides an exclusive Cassandra Database monitoring model that runs quick health checks on the Cassandra Database at configured intervals, and proactively alerts administrators to potential bottlenecks to the performance of the server.

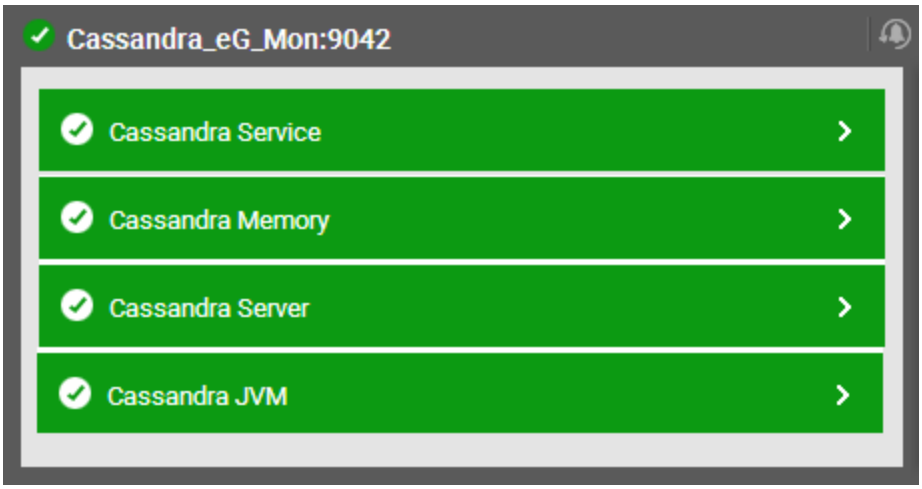


Figure 3.1: The layer model of the Cassandra Database

Using the model depicted by Chapter 3, administrators can determine the following:

- What is the availability and responsiveness of the target database node?
- What is the size of the buffer pool?
- How many times user requests were not serviced from the buffer pool?
- What is the total size of each cache?
- Is the cache over-utilized?
- How well requests are serviced by the cache?
- What is the ratio of requests serviced by the cache without looking into the disks?
- How many clients are connected to the target database node through native protocol and thrift protocol?
- What protocol is preferred by the clients to connect to the target database node - is it native protocol or thrift protocol?
- What is the rate at which messages were written to the commit log?

- What is the growth rate of the commit log?
- What is the wait time spent for the data to be fsnyced?
- What is the size of the commit log?
- How many compactions were performed per second?
- How many compactions are pending per second?
- What is the amount of data compacted per second?
- How many prepared statements were cached per second?
- How many prepared statements were executed per second?
- How many prepared statements were evicted per second from the prepared statement cache?
- Is the *Hinted Handoff* feature enabled on the database node?
- How many hints are available in the target database node?
- How many hints are currently active for replay?
- How many bloom filter false positives were detected on each keyspace?
- What is the amount of space utilized in the disk by the bloom space for each keyspace?
- What is the average time taken by each keyspace to read the data for the requests?
- What is the average time taken by each keyspace to write the data corresponding to the requests?
- What is the average time taken by each keyspace to respond to a range of requests?
- What is the current size of the log files on the target database node?
- How many fatal errors were logged in the log file?
- How many warning messages were logged in the log file?
- What is the growth rate of the log file?
- How many queries are executing on the database node beyond the configured time?
- How many messages of each type are dropped by the target database node per second?
- How long did it take for the messages of each type to be dropped from the target database node?
- How many large messages/small messages/gossip messages were completely transferred to each node from the target database node?

- How many large messages/small messages/gossip messages were dropped during transfer to each node from the target database node?
- How many messages were timed out during transfer to each node?
- What is the current status of each node?
- What is the amount of data handled by each node?
- How many nodes are available in the cluster?
- How many nodes joined the cluster?
- How many nodes are unreachable in the cluster?
- How many background read repairs were coordinated by the target database node?
- How many read repairs were attempted by the target database node?
- How many data digests were coordinated by the target database node?
- What is the time taken for servicing each request type?
- What is the rate at which requests of each type were unavailable?
- How many requests of each type failed per second?
- How may requests of each type timed out per second?
- Is the gossip protocol service running?
- Is the native protocol service running?
- What is the current data size of the target database node?
- What is the rate at which data is growing on the target database node?
- How many unhandled exceptions occurred on the target database node per second?
- How many tasks were active in each thread pool?
- What is the rate at which tasks were completed in each thread pool?
- How many tasks were pending in each thread pool?
- What is the rate at which tasks were blocked per second in each thread pool?

The sections that will follow discuss each of the layers of Figure 3.1 in great detail.

3.1 The Cassandra JVM Layer

This layer collectively reports the resource usage and overall health of the target Cassandra Database Node.

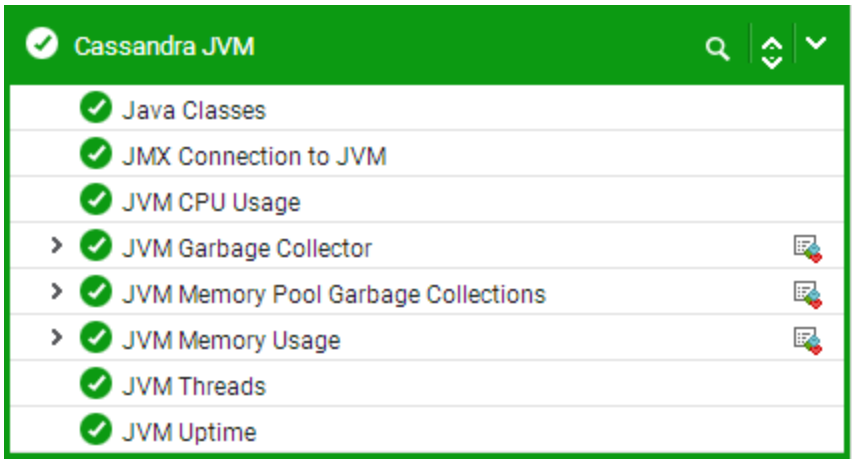


Figure 3.2: The tests mapped to the Cassandra JVM layer

All the tests displayed in Figure 3.2 have been dealt with in the *Monitoring Java Applications* document. Therefore, let us discuss the remaining layers in detail in the forthcoming sections.

3.2 The Cassandra Server Layer

The tests associated with this layer indicate the level of activity on the Maria Database server by monitoring the keyspaces, nodes, prepared statements executed on it. In addition, the tests associated with this layer also throws insights into the requests that failed/time out often and also helps administrators determine the performance degradation of the database node if any, by analyzing the status of the hinted handoffs and the count of hints.

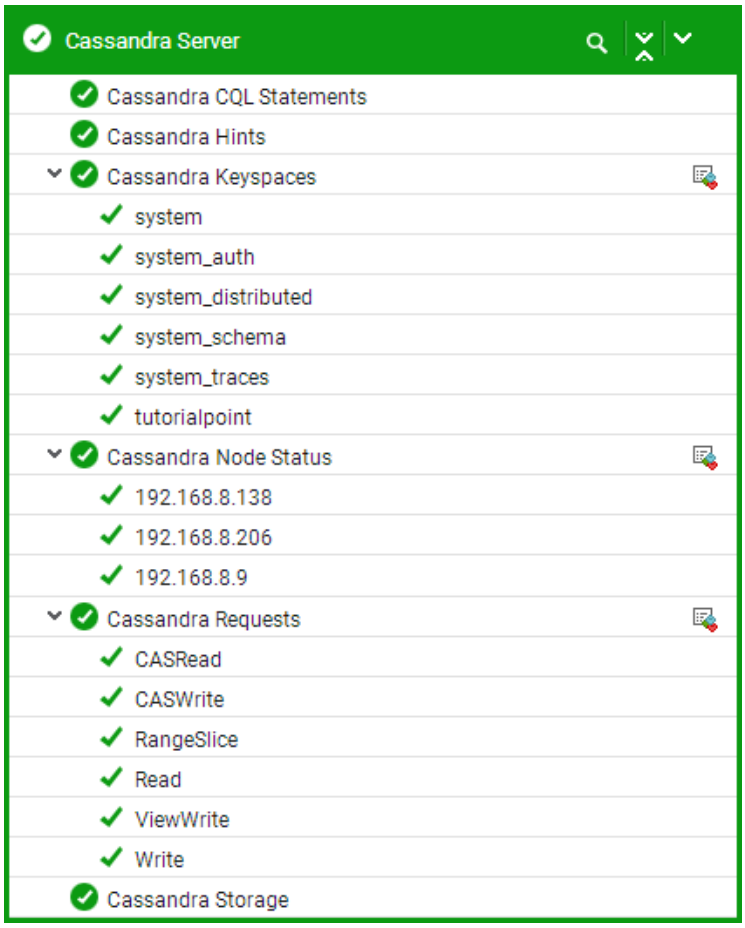


Figure 3.3: The tests associated with the Cassandra Server layer

3.2.1 Cassandra CQL Statements Test

Cassandra Query Language (CQL) is a query language for the Cassandra database. The Cassandra Query Language (CQL) is the primary language for communicating with the Cassandra database. The most basic way to interact with Cassandra is using the CQL shell, cqlsh. Using cqlsh, you can create keyspaces and tables, insert and query tables etc. When using the JPA annotations for queries, the result is prepared statements that will be executed against the database. Prepared statements have two phases for execution: preparation of the statement, and execution of the statement. These prepared statements will be stored in the prepared statement cache. If the prepared statements are not executed as and when the statements are prepared, then, it may be owing to reasons such as processing bottlenecks on the database server or inadequate resources (CPU, memory etc) allocated for execution of the statements or the database node in itself may be poorly configured. In addition, if more statements are prepared and are stored in the prepared statement cache, the statements may be evicted from the cache if the cache is not configured appropriately. To constantly maintain a steady configuration of the database node and its

components, it is important to keep a vigil on the execution of the prepared statements. The **Cassandra CQL Statements** test helps administrators in this regard!

This test reports how well the prepared statements are cached and executed per second. In addition, this test also reports the rate at which the prepared statements are evicted from the cache and how well the regular statements are executed. Using this test, administrators can be alerted to inadequate sizing of the prepared statement cache and processing bottlenecks on the target database node.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Prepared statements	Indicates the number of prepared statements	Statements/sec	CQL supports prepared statements. Prepared statements

Measurement	Description	Measurement Unit	Interpretation
	cached per second during the last measurement period.		are an optimization that allows to parse a query only once but execute it multiple times with different concrete values.
Executed prepared statements	Indicates the number of prepared statements executed per second during the last measurement period.	Statements/sec	Any statement that uses at least one bind marker will need to be prepared. After which the statement can be executed by providing concrete values for each of its marker. A low value for this measure is a cause of concern. This may be due to the processing bottlenecks on the target database node or inadequate resource allocated to the target database node or the inadequate resource allocated to the execution of the statement.
Evicted prepared statements	Indicates the rate at which the prepared statements were evicted from the prepared statement cache during the last measurement period.	Statements/sec	A high value for this measure indicates that too many statements are prepared and most of them could not be stored in the prepared statement cache. This is mainly due to the inadequate configuration of the cache size. Administrators may therefore increase the size of the prepared statement cache to keep the value of this measure to a minimum.
Regular statements	Indicates the rate at which regular statements i.e., non prepared statements were executed during the last measurement period.	Statements/sec	

3.2.2 Cassandra Hints Test

Over time, data in a Cassandra replica can become inconsistent with other replicas due to the distributed nature of the database. Node repair corrects the inconsistencies so that eventually all nodes have the same and most upto-date data. It is important part of regular maintenance for every Cassandra cluster.

Cassandra provides the following repair processes:

- Hinted Handoff
- Read Repair
- Anti-Entropy Repair

Occasionally, a node may become unresponsive while data is being written. This unresponsiveness may be due to hardware problems, network issues, or overloaded nodes that experience long garbage collection (GC) pauses. If a node is unable to receive a particular write, the write's coordinator node preserves the data to be written as a set of hints. When the node comes back online, the coordinator effects repair by handing off hints so that the node can catch up with the required writes. This type of repair process is termed as Hinted Handoff. The handing off hints will be happening for a period given by the *max_hint_window_ms* setting in *cassandra.yaml*. Once this window expires, nodes will stop saving hints.

Hinted Handoff is an optional part of writes whose primary purpose is to provide extreme write availability when consistency is not required. Secondarily, Hinted Handoff can reduce the time required for a temporarily failed node to become consistent again with live ones. This is especially useful when a flakey network causes false-positive failures. If the hinted handoff is not enabled, then, the node may contain outdated data for a longer duration which may result in users using stale data which may result in a dip in user experience. It is therefore necessary to monitor the status of the hinted handoff round the clock. The **Cassandra Hints** test helps administrators in this regard!

By closely monitoring the Casssandra Database node, this test helps administrators to figure out if the hinted handoff is enabled or not. In addition, this test reports the total number of hints that the node needs to be updated with and the number of hints that are active for replay. If there is an abnormal increase in the count of hints, administrators may infer that there is a potential database performance degradation.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the management.properties file in the <JAVA_HOME>\jre\lib\management folder used by the target application.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Is hinted handoff enabled?	Indicates whether/not the hinted handoff is enabled.		The values that this measure reports and its corresponding numeric values are mentioned in the table below: Measure Value / Numeric Value No / 0 Yes / 1 **Note:** By default, this measure reports the above-mentioned **Measure Value**s to indicate whether/not the hinted handoff is enabled. However, in the graph of this measure, the same will be represented using the numeric equivalents - 0 or 1 only.
Hints in progress	Indicates the number of hints that are currently	Number	Ideally, the value of this measure should be 0.

Measurement	Description	Measurement Unit	Interpretation
	active to replay.		
Total hints	Indicates the total number of hints.	Number	Ideally, the value of this measure should be 0. Though hints are part of Cassandra's failure management, a lot of hints logged in within a short span may indicate a problem that is being ignored. A suddden/gradual increase in the value of this measure indicates network or availability issues.

3.2.3 Cassandra Keyspaces Test

A keyspace in Cassandra is a namespace that defines data replication on nodes. A cluster contains one keyspace per node. CQL stores data in tables (SSTables, memtable), whose schema defines the layout of said data in the table, and those tables are grouped in keyspaces. A keyspace defines a number of options that applies to all the tables it contains, most prominently of which is the replication strategy used by the keyspace. It is generally encouraged to use one keyspace by application, and thus many cluster may define only one keyspace.

The keyspace is the top-level database object that controls the replication for the object it contains at each datacenter in the cluster. Keyspaces contain tables, materialized views and user-defined types, functions and aggregates.

In the read path, Cassandra merges data on disk (in SSTables) with data in RAM (in memtables). To avoid checking every SSTable data file for the partition being requested, Cassandra employs a data structure known as a bloom filter. Bloom filters are maintained per SSTable, i.e. each SSTable on disk gets a corresponding bloom filter in memory.

Bloom filters are a probabilistic data structure that allows Cassandra to determine one of two possible states: - The data definitely does not exist in the given file, or - The data probably exists in the given file. While bloom filters can not guarantee that the data exists in a given SSTable, bloom filters can be made more accurate by allowing them to consume more RAM. As accuracy improves (as the *bloom_filter_fp_chance* (bloom filter false positive) gets closer to 0), memory usage increases non-linearly i.e., the bloom filter with a *bloom_filter_fp_chance* = *0.01* requires about three

times as much memory as the same table with *bloom_filter_fp_chance = 0.1*. If the bloom filter false positives increases rapidly, the memory usage may decrease and the disk overhead increase manifold. Therefore, it is essential to contain the bloom filter false positives before the disk is bombarded with requests. Similarly, the read requests and write requests in each keyspace also should be monitored at a closer pace so that administrators can ensure that the data is available in the keyspace. This will ensure a reduced disk overhead for the requests received. The **Cassandra Keyspaces** test helps administrators in monitoring the keyspace and containing the bloom filter false positives!

This test auto-discovers the keyspaces in the target Cassandra Database node and for each keyspace, this test reports the count of SSTables and memory tables available. In addition, this test reveals the count of bloom filter false positives on each keyspace and the space utilization of the bloom filters in depth. The test also provides insights into the read and write latency of each keyspace so that administrators can get an idea of the keyspace that is lagging behind in catering the requests.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Parameters	Description
Detailed Diagnosis	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
Bloom filter false positives	Indicates the number of bloom filter false positives in this keyspace.	Number	Typical values for *bloom_ filter_ fp_ chance* are usually between 0.01 (1%) to 0.1 (10%) false-positive chance, where Cassandra may scan an SSTable for a row, only to find that it does not exist on the disk. The parameter should be tuned by use case: • Users with more RAM and slower disks may benefit from setting the *bloom_ filter_ fp_ chance* to a numerically lower number (such as 0.01) to avoid excess IO operations. • Users with less RAM, more dense nodes, or very fast disks may tolerate a higher *bloom_ filter_ fp_ chance* in

Measurement	Description	Measurement Unit	Interpretation
			order to save RAM at the expense of excess IO operations • In workloads that rarely read, or that only perform reads by scanning the entire data set (such as analytics workloads), setting the *bloom_filter_fp_chance* to a much higher number is acceptable.
Bloom filter false positive rate	Indicates the bloom filter false positive ratio in this keyspace.	Percent	A low value is desired for this measure.
Bloom filter space used	Indicates the disk space used by the bloom filter in this keyspace.	MB	A high value indicates that the data is available in the keyspace.
Live SS tables	Indicates the number of SSTables that are currently live/active in this keyspace.	Number	Compare the value of this measure across the keyspaces to figure out the keyspace on which there are too many SSTables that are active/live.
Disk space used by live SSTables	Indicates the disk space utilized by the SSTables that are live/active in this keyspace.	MB	A continuously increasing value of this measure indicates that the SSTables are upto-date with the data.
Memory table column count	Indicates the number of columns present in the memory table available in this keyspace.	Number	
Memory table switch count	Indicates the number of flushes in memory table per second that resulted in the switch out of the memory table available in this keyspace.	Switches/second	

Measurement	Description	Measurement Unit	Interpretation
Memory table live data size	Indicates the size of the data stored in the memory table available in this keyspace.	MB	A continuously increasing value of this measure indicates that the memory tables are not updating the data to the SSTables. Administrators should therefore check if adequate space is allocated to the SSTables.
Memory table off-heap size	Indicates the off-heap memory size of the memory table available in this keyspace.	MB	
Memory table on-heap size	Indicates the on-heap memory size of the memory table available in this keyspace.	MB	
Recent Bloom filter false positives	Indicates the recent number of bloom filter positives negotiated in this keyspace.	Number	
Recent Bloom filter false positive rate	Indicates the recent bloom filter false positive ratio negotiated in this keyspace.	Percent	
Avg read latency	Indicates the average time taken by this keyspace to respond to read requests.	Milliseconds/request	Compare the value of this measure across the keyspaces to determine the keyspace that is taking too long to respond to read requests.
Read latency 99thpercentile	Indicates the average 99th percentile time taken by this keyspace to respond to user requests.	Milliseconds	
Avg write latency	Indicates the average time taken by this keyspace to write the data for the requests.	Millseconds/request	Compare the value of this measure across keyspaces to figure out the keyspace that is taking too long to write the data for the requests received.

Measurement	Description	Measurement Unit	Interpretation
Write latency 99thpercentile	Indicates the average 9th percentile time taken by this keyspace to respond to each write request.	Milliseconds	
Avg range latency	Indicates the average time taken by this keyspace to respond to a range of requests.	Milliseconds/request	
Range latency 99thpercentile	Indicates the average 99th percentile time taken by this keyspace to respond to a range of user requests.	Milliseconds	

3.2.4 Cassandra Node Status Test

This test auto-disovers the nodes of the target Cassandra Database cluster and for each node, reports the current state and the load handled. Using this test , administrators can figure out the nodes that are down and the nodes that are handling the maximum amount of load.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for each node clustered to the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1**

Parameters	Description
	file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Status	Indicates the current status of this node.		The values reported by this measure and its numeric equivalents are mentioned in the table below: State / Numeric Value Down: 0 Up: 1 Off cooling: 10 Off connector rating: 11 On but inline power failure: 12 **Note:** By default, this measure reports the Measure Values listed in the table above to indicate the current state of this node. The graph of this measure however, represents the status of the node using the numeric equivalents only.
Load	Indicates the amount of data handled by this node. In other words, this indicates amount of file	MB	By comparing the value of this measure across nodes, administrators can figure out the node that is handling the maximum amount of load.

Measurement	Description	Measurement Unit	Interpretation
	system data under the Cassandra data directory after excluding all content in the snapshots sub directories of this node.		

3.2.5 Cassandra Requests Test

All nodes in Cassandra are peers. A client read or write request can go to any node in the cluster. When a client connects to a node and issues a read or write request, that node serves as the coordinator for that particular client operation. The job of the coordinator is to act as a proxy between the client application and the nodes (or replicas) that own the data being requested. The coordinator determines which nodes in the ring should get the request based on the cluster configured partitioner and replica placement strategy.

In environments where multiple nodes are deployed, the nodes may receive thousands of read and write requests at a single time. To cater to the requests, it is always important for the nodes to be active. If multiple nodes are not available, then the requests may take too long to be serviced or at the worst case, the requests may fail. Therefore, it becomes important to keep track on the time taken by the nodes to service the requests and the count of requests that failed or timed out. The **Cassandra Requests** test helps administrators in this regard!

For each type of requests received by the target Cassandra Database server, this test reports the time taken to service the requests, the count of the requests that were unavailable, timed out and failed. By closely monitoring the measures reported by this test, administrators can further investigate the reason on why the requests were failed/timed out and take remedial measures to ensure that the requests are serviced at a faster pace!

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for each Request type on the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total latency	Indicates the total time taken for servicing the requests of this type during the last measurement period.	Milliseconds	A low value is desired for this measure. If the value of this measure increases all of a sudden or gradually, then, it indicates that some peer nodes of the target Casandra database server are not servicing the requests or most of the requests are failing due to unavailability of the requested data etc.
Unavailable requests	Indicates the rate at which requests of this type were unavailable during the last measurement period.	Requests/sec	An unavailable request is the only request that will cause a write to fail, so any occurrences are serious. Cassandra's inability to meet consistency requirements can mean that several nodes are

Measurement	Description	Measurement Unit	Interpretation
			down or otherwise unreachable, or that stringent consistency settings are limiting the availability of the node.
Timed-out requests	Indicates the rate at which requests of this type were timed out during the last measurement period.	Timeouts/sec	A low value is desired for this measure.
Failed requests	Indicates the rate at which requests of this type failed during the last measurement period.	Failures/sec	Ideally, the value of this measure should be zero.

3.2.6 Cassandra Read Repairs Test

Over time, data in a replica can become inconsistent with other replicas due to the distributed nature of the Cassandra database. Node repair corrects the inconsistencies so that eventually all nodes have the same and most up-to-date data. It is important part of regular maintenance for every Cassandra cluster. Read repair improves consistency in a Cassandra cluster with every read request.

In a read, the coordinator node sends a data request to one replica node and digest requests to others for consistency level (CL) greater than ONE. If all nodes return consistent data, the coordinator returns it to the client.

In read repair, Cassandra sends a digest request to each replica not directly involved in the read. Cassandra compares all replicas and writes the most recent version to any replica node that does not have it. If the query's consistency level is above ONE, Cassandra performs this process on all replica nodes in the foreground before the data is returned to the client. Read repair repairs any node queried by the read. This means that for a consistency level of ONE, no data is repaired because no comparison takes place. For QUORUM, only the nodes that the query touches are repaired, not all nodes.

There are three types of read requests that a coordinator can send to a replica:

- A direct read request
- A digest request

- A background read repair request

In a direct read request, the coordinator node contacts one replica node. Then the coordinator sends a digest request to a number of replicas determined by the consistency level specified by the client. The digest request checks the data in the replica node to make sure it is up to date. Then the coordinator sends a digest request to all remaining replicas. If any replica nodes have out of date data, a background read repair request is sent. Read repair requests ensure that the requested row is made consistent on all replicas involved in a read query.

In some environments, at times, due to network issues or due to failure of multiple nodes, the data may not be replicated to all nodes. If suppose a node becomes available after a short hiatus, there may be a sudden influx of read repair requests to the node so that the outdated data in the node can be replaced with the data that is up to date. This sudden influx of read repairs is a cause of concern when the nodes are not updated at regular intervals. Therefore, it is essential to monitor the read repairs frequently. To identify such erratic behavior in read repair requests and the background repairs performed in the nodes, administrators can use the **Cassandra Read Repairs** test.

Using this test, administrators can figure out the count of background read repairs coordinated by the node and the read repairs attempted by the node. In addition, this test also reveals the full data digests coordinated by the node. By analyzing the count of read repairs at regular intervals, administrators can figure out the erratic pattern of read repairs performed on the node and figure out the exact cause of such erratic behavior in updating the node with the latest data information.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To

Parameters	Description
	know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Background repaired	Indicates the number of background read repairs coordinated by the node.	Number	A high value for this measure indicates that the data requested is out of date and is being updated. This may indicate that there was a delay in update process or a sudden unavailability of the node due to network issues etc.
Read repair attempts	Indicates the number of read repairs attempted by the node.	Number	A high value indicates that the read repair is carried out to update the replica nodes that are out-of-date. This may also indicate that there was a delay in update process or a sudden unavailability of the node due to network issues etc.
Repaired blocking	Indicates the number of full data digests coordinated by the node.	Number	On any consistency level that involves more than one node (i.e., all except ANY and ONE), if the read digests do not match up, read repair is done in a blocking fashion before returning results. This means that if the repair does not complete on time, the read requests fail.

3.2.7 Cassandra Storage Test

This test reports the current size of the target Cassandra Database node and the rate at which data is growing. This test also reveals the rate of unhandled exceptions on the database node. Using this test, administrators can figure out how well the database node is growing as well as capture and rectify the unhandled exceptions before the data becomes stale!

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent

Outputs of the test : One set of results for the target Cassandra Database node that is being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Current data size	Indicates the current data size of the target node.	MB	

Measurement	Description	Measurement Unit	Interpretation
Data growth rate	Indicates the rate at which data is growing on the target node.	MB/sec	
Exceptions	Indicates the rate at which unhandled exceptions occurred on the node during the last measurement period.	Exceptions/sec	Ideally the value of this measure should be 0. A sudden/gradual increase of this measure indicates that too many unhandled exceptions have occurred. Administrators may need to troubleshoot the unhandled exceptions by checking the *Errors* and *Fatal* measures reported by the Cassandra Logs test.

3.2.8 Cassandra Logs Test

The Cassandra database logs are a vast source of information related to errors and warnings that a Cassandra database server encounters. Administrators use these log files not only to spot problem conditions, but also to troubleshoot them. By periodically scanning the Cassandra error log for errors/warnings, the **Cassandra Logs** test promptly notifies administrators as soon as a new error, warning, or fatal error is logged in the file.

As messages keep getting logged in the log files, these log files grow large in size. If log file growth is left unchecked, it can consume all available space in the database. Administrators can effectively track the log file growth and initiate measures to control it using the **Cassandra Logs** test. The test reports the errors logged in the log files, the current size of the log files and their growth rate, and proactively alerts administrators if the rate of growth is abnormal.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
File Path	Specify the full path to the log file that is to be monitored. For example, the log file path can be: *C:\tmp\system.log*.
ISUTF16	If the error log file to be monitored is encoded with UTF-16, then, set this flag to **Yes**. By default, this flag is set to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Errors	Indicates the number of error messages logged in the log files during the last measurement period.	Number	Ideally, the value of this measure should be 0.
Fatal errors	Indicates the number of fatal errors captured by the log files during the last measurement period.	Number	Ideally, the value of this measure should be 0.
Warning messages	Indicates the number of warning messages logged in the log files during the last measurement period.	Number	Ideally, the value of this measure should be 0.
File size	Indicates the current size of the log file.	MB	
Growth rate	Indicates the rate at which the log file is growing.	MB/sec	A high value for this measure or a consistent increase in its value indicates that the Cassandra log is rapidly growing and may end up occupying too much space on the disk.

3.3 The Cassandra Memory Layer

Using the tests mapped to this layer, administrators can proactively detect:

- irregularities in the sizing of the buffer pool
- Poor responsiveness of the cache
- Poor cache usage
- Delays in flushing writes to disk due to over growth of the commit logs
- Potential disk space crunch due to irregularities in the compaction process

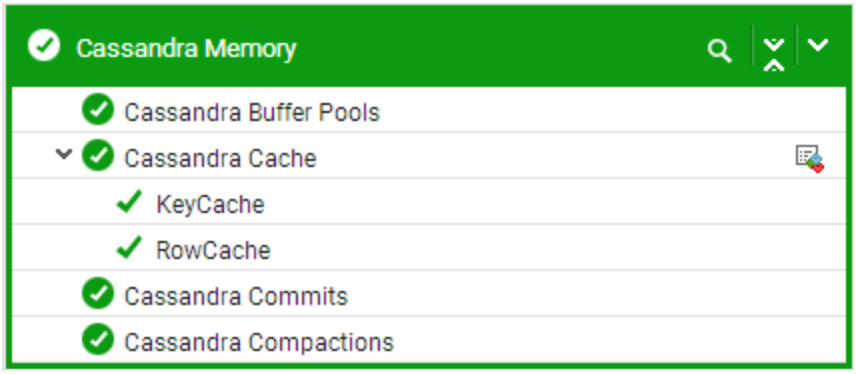


Figure 3.4: The tests mapped to the Cassandra Memory layer

3.3.1 Cassandra Buffer Pools Test

The Cassandra database node manages an internal buffer pool system. This buffer pool is meant to keep allocations and GC lower by recycling on and off heap buffers. If the buffer pool size is inadequate, then, the requests may not be serviced by the buffer pool more often that not. To avoid such frequent buffer pool misses, it is necessary to keep a vigil on the buffer pool of the Cassandra database node. The **Cassandra Buffer Pools** test helps administrators achieve this!

This test reports the size of the buffer pool and also reports the count of service requests that were not serviced by the buffer pool. Using this test, administrators can figure out how often the buffer pool failed to service requests and figure out if it is attributed to the poor sizing of the buffer pool.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node that is being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	By default, the JMX User parameter is set to *none*. If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Misses	Indicates the number of times requests were not serviced from the buffer pool during the last measurement period.	Number	A low value is desired for this measure.
Miss rate	Indicates the rate at which requests were not serviced from the buffer pool during the last measurement period.	Misses/sec	A high value for this measure indicates that the buffer pool needs to be allocated with additional resources.
Size	Indicates the size of the buffer pool.	MB	

3.3.2 Cassandra Cache Test

Cassandra includes integrated caching and distributes cache data around the cluster. When a node goes down, the client can read from another cached replica of the data. The integrated architecture also facilitates troubleshooting because there is no separate caching tier, and cached data matches what is in the database exactly. The integrated cache alleviates the cold start problem by saving the cache to disk periodically. Cassandra reads contents back into the cache and distributes the data when it restarts. The cluster does not start with a cold cache.

The partition key cache is a cache of the partition index for a Cassandra table. Using the key cache instead of relying on the OS page cache decreases seek times. Enabling just the key cache results in disk (or OS page cache) activity to actually read the requested data rows, but not enabling the key cache results in more reads from disk.

To cache rows, if the row key is not already in the cache, Cassandra reads the first portion of the partition, and puts the data in the cache. If the newly cached data does not include all cells configured by user, Cassandra performs another read. The actual size of the row-cache depends on the workload. You should properly benchmark your application to get ”the best” row cache size to configure.

There are two row cache options, the old serializing cache provider and a new off-heap cache (OHC) provider. The new OHC provider has been benchmarked as performing about 15% better than the older option.

Typically, you enable either the partition key or row cache for a table.

If the caches are not sized appropriately, then, frequent disk accesses may happen which may cause severe disk overhead. To avoid this, administrators may need to size the caches appropriately and also figure out the cache that is infrequently used. The **Cassandra Cache** test helps administrators in this regard!

This test auto-discovers the caches on the target database server. For each cache discovered, this test reports the maximum cache size and how much of this size is presently occupied by cached data; this reveals, whether/not the cache has enough RAM to hold additional data. Inconsistencies in cache sizing can be detected in the process and their impact on performance analyzed. This test also throws light on how well the cache services requests. Using this test, administrators can figure out the cache from which the least requests have been serviced and analyze the real reason behind such poor responsiveness.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for each cache of the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total size	Indicates the total size of this cache i.e., the total space allocated to this cache.	MB	
Used size	Indicates the amount of space that is already utilized in this cache.	MB	Ideally, this value should be well below the *Total size* measure. A steady and significant rise in this value could mean that cached data is not been written to the disks frequently and/or least-used data is not been evicted properly. You

Measurement	Description	Measurement Unit	Interpretation
			may want to fine- tune these operations and then check to see if it reduces cache memory usage.
Cache usage	Indicates the percentage of space that is already utilized in this cache.	Percent	A value close to 100% is a cause for concern, as it indicates that the cache is consuming RAM excessively. You may want to consider resizing the cache to avoid direct disk reads.
Hit rate	Indicates the rate at which the requests were serviced by this cache during the last measurement period.	Hits/sec	
Hit ratio	Indicates the requests serviced by this cache without having to read from the disk during the last measurement period.	Percent	A value of 85% or more is desired for this measure. Temporary dips below this number are expected directly after a large bulk update, but if you stay here longer term this can be a problem of data modeling issues or configuration problems, such as JNA is not correctly installed, and therefore the keycache is residing on heap. In theory, when combined with heap pressure this can end up over flushing the cache.
Request rate	Indicates the rate at which cache requests were serviced by this cache during the last measurement period.	Requests/sec	A high value is desired for this measure.

3.3.3 Cassandra Commits Test

Cassandra processes data at several stages on the write path, starting with the immediate logging of a write and ending in with a write of data to disk:

- Logging data in the commit log
- Writing data to the memtable
- Flushing data from the memtable
- Storing data on disk in SSTables

When a write occurs, Cassandra database node stores the data in a memory structure called memtable, and to provide configurable durability, it also appends writes to the commit log on disk. Memtables and SSTables are maintained per table. The commit log is shared among tables. The commit log receives every write made to a Cassandra node, and these durable writes survive permanently even if power fails on a node.

The memtable is a write-back cache of data partitions that Cassandra looks up by key. The memtable stores writes in sorted order until reaching a configurable limit, and then is flushed. To flush the data, Cassandra writes the data to disk, in the memtable-sorted order.

When the memtable content exceeds the configurable threshold or the commit log space exceeds the *commitlog_total_space_in_mb*, the memtable is put in a queue that is flushed to disk. If the data to be flushed exceeds the memtable_cleanup_threshold, Cassandra blocks writes until the next flush succeeds. To reduce the commit log replay time, the recommended best practice is to flush the memtable before you restart the nodes. If a node stops working, replaying the commit log restores to the memtable the writes that were there before it stopped. Data in the commit log is purged after its corresponding data in the memtable is flushed to an SSTable on disk.

If the size of the commit log is inadequately configured against the *commitlog_total_space_in_mb* parameter, then, the data should be frequently flushed which in turn may increase the disk I/O manifold. This may cause performance bottlenecks on the target database node. Sometimes the growth of the commit log may also increase steeply. This may be primarily due to the write load on the node exceeding the ability to keep up with flushing the memtables. In the process, the older commit logs may not be deleted on time to accommodate additional data. To address these problems, it is necessary to monitor the commit logs of the target Cassandra Database node periodically. The **Cassandra Commits** test helps administrators in this regard!

This test monitors the commit log on the target database node and reports the current size and the growth rate of the commit log. In addition, this test reports how well the messages were written to the

commit log and the rate at which messages were pending to be written to the disk from the commit log. The time spent to sync the data from the commit log to the disk is also monitored and reported. Using this test, administrators may be able to plan the capacity of the commit log based on the amount of data written to the commit log.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Completed	Indicates the rate at which the messages were written to the commit log since the start/restart of the database node.	Commits/sec	A high value is desired for this measure.
Commit log growth	Indicates the amount of	MB/sec	A sudden/gradual increase in the value

Measurement	Description	Measurement Unit	Interpretation
rate	data written to the commit log per second during the last measurement period.		of this measure is a cause of concern. Sometimes the commit log can grow in an "out of control" fashion. Essentially, this can happen because the write load on the node exceeds Cassandra's ability to keep up with flushing the memtables (and thus, removing old commit log files). If administrators find a node with dozens of commit log files, and the number seems to keep growing, this might be a serious issue which needs to be addressed immediately.
Commit log size	Indicates the current size of the commit log.	MB	Ensure that the size of the commit log is adequately configured against the *commitlog_total_space_in_mb* parameter.
Pending	Indicates the rate at which commit log messages were written but are yet to be fsynced.	Commits/sec	A low value is desired for this measure.
Wait time on commit log fsync	Indicates the time spent for the data to be fsynced in the commit log.	Seconds	
Commit log segment allocation wait time	Indicates the time spent waiting for a segment to be allocated on the commit log.	Seconds	Ideally, the value of this measure should be zero.

3.3.4 Cassandra Compactions Test

The concept of compaction is used for different kinds of operations in Cassandra, the common thing about these operations is that it takes one or more sstables and output new sstables.

The Cassandra write process stores data in files called SSTables. SSTables are immutable. Instead of overwriting existing rows with inserts or updates, Cassandra writes new timestamped versions of

the inserted or updated data in new SSTables. Cassandra does not perform deletes by removing the deleted data: instead, Cassandra marks it with tombstones.

Over time, Cassandra may write many versions of a row in different SSTables. Each version may have a unique set of columns stored with a different timestamp. As SSTables accumulate, the distribution of data can require accessing more and more SSTables to retrieve a complete row.

To keep the database healthy, Cassandra periodically merges SSTables and discards old data. This process is called compaction.

Compaction works on a collection of SSTables. From these SSTables, compaction collects all versions of each unique row and assembles one complete row, using the most up-to-date version (by timestamp) of each of the row's columns. The merge process is performant, because rows are sorted by partition key within each SSTable, and the merge process does not use random I/O. The new versions of each row is written to a new SSTable. The old versions, along with any rows that are ready for deletion, are left in the old SSTables, and are deleted as soon as pending reads are completed.

Compaction causes a temporary spike in disk space usage and disk I/O while old and new SSTables co-exist. As it completes, compaction frees up disk space occupied by old SSTables. It improves read performance by incrementally replacing old SSTables with compacted SSTables. Cassandra can read data directly from the new SSTable even before it finishes writing, instead of waiting for the entire compaction process to finish.

As Cassandra processes writes and reads, it replaces the old SSTables with new SSTables in the page cache. The process of caching the new SSTable, while directing reads away from the old one, is incremental — it does not cause a the dramatic cache miss. Cassandra provides predictable high performance even under heavy load.

If there are too many pending reads that are serviced from the old SSTables, the old SStables cannot be deleted during compaction. This process may delay the number of compactions that need to be performed on the target Cassandra database node. If old SSTables are not deleted periodically, then, administrators may be alerted to potential space crunch in the disk which may impact the addition of new data into the SSTables. Therefore, it is necessary to monitor the compactions and the size of data compacted on the target database node round the clock. The **Cassandra Compactions** test helps administrators in this regard.

This test monitors the target Cassandra database node and reports the rate at which data was compacted. In addition, this test also reveals the count of compactions that are pending and the amount of data pending to be compacted per second. Using this test, administrators can be alerted

to irregularities in compaction process if any, and rectify them at a faster pace before space crunch of the disk reaches abnormal limits.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Compacted size	Indicates the amount of data compacted per second (measured after compaction process) during the last measurement period.	MB/sec	
Completed compactions	Indicates the number of compactions completed	Compactions/sec	A high value is desired for this measure.

Measurement	Description	Measurement Unit	Interpretation
	per second during the last measurement period.		
Pending compactions	Indicates the number of compactions that are currently pending.	Number	Ideally, the value of this measure should be zero.
Total compactions	Indicates the total number of compactions performed per second during the last measurement period.	Compactions/sec	

3.4 The Cassandra Service Layer

The tests associated with this layer monitor the protocol through which clients communicate with the target database node, the nodes that are available/unreachable, the status of the gossip protocol service and the native protocol service, the large messages/small messages/gossip messages pending transfer to the joining node in the cluster. By executing the tests associated with this layer, administrators can monitor the message drop rate on the target database node and the long running queries executing on the database server.

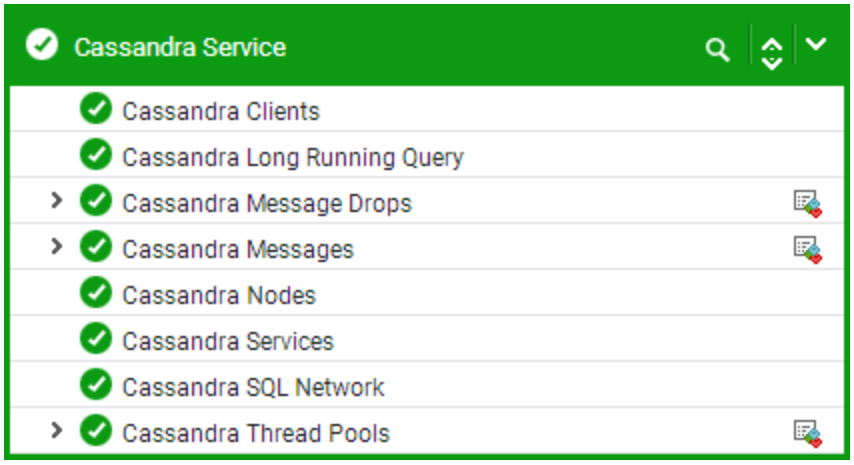


Figure 3.5: The tests associated with the Cassandra Service layer

3.4.1 Cassandra SQL Network Test

This test monitors the availability and responsiveness of the Cassandra Database node by emulating a client connecting and executing queries on the Cassandra Database node.

Target of the test : A Cassandra Database node

Agent deploying the test : An internal agent/remote agent

Outputs of the test : One set of results for the target Cassandra Database node being monitored

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
CQL User	Specify the name of the CQL User with the following privileges in this text box: *SELECT privilege on the system_traces.sessions table* *SELECT privilege on the system.peer_events table*
CQL Password	Specify the password corresponding to the CQL User.
Confirm Password	Confirm the CQL Password by retyping it in this text box.
Keyspace Name	The name of the keyspace to connect to. The default is “system_schema”. To monitor multiple keyspaces, ensure that the keyspaces are provided as a colon-separated list. Alternatively, you can use the semi-colon(;) as the separator for the keyspaces.
Query	The select query to execute. The default is “select * from keyspaces”. If the target Cassandra Database node is installed as case sensitive, then the value of query parameter should also be case sensitive. If multiple keyspaces are specified in the DATABASE text box, then you will have to provide multiple queries here separated by a semi- colon (;) - for e.g., select * from keyspaces;select * from alarm. Every keyspace being monitored, should have a corresponding Query specification.
Detailed Diagnosis	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

Parameters	Description
	• 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
Availability	Indicates the availability of the server.	Percent	The availability is 100% when the server is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database server, or because the server has not been started. The availability is 100% when the instance is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database nodde, 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 database instance is unavailable, or if a query to the database fails. In this case, you can check the values of the *Connection Availability* and *Query Availability* measures to know what is exactly causing the database node to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure? Using the detailed diagnosis of this measure, you can easily find out unavailability of the server.

Measurement	Description	Measurement Unit	Interpretation
Connection response time	The time taken by the database node to respond to a user query. This is the sum total of the response time and query response time.	Seconds	A sudden increase in the connection response time is indicative of a bottleneck at the database node.
Connection availability	Indicates whether the database connection 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 SQL 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.
Query availability	Indicates whether the database query 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 *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 database node unavailability.
Response time	Indicates the time taken by the database	Seconds	A high value could indicate a connection bottleneck. Whenever the

Measurement	Description	Measurement Unit	Interpretation
	connection.		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 response time	Indicates the time taken for query execution.	Seconds	A high value could indicate that one/more queries to the database are taking too long to execute. Inefficient/badly designed queries to the database often run for long periods. If the value of this measure is higher than that of the *Connection response time* measure, you can be rest assured that long running queries are the ones causing the responsiveness of the server to suffer.
No of records	Indicates the number of records fetched from the database node.	Number	The value 0 indicates that no records are fetched from the database node.

3.4.2 Cassandra Clients Test

Cassandra clients can communicate with the server through two protocols: the CQL binary protocol or through an RPC protocol called thrift. The CQL binary protocol is a newer protocol and is preferred over thrift. Cassandra Query Language (CQL) is a language, similar to SQL, that is used by Cassandra to create commands to manipulate its schema structure and data (DDL and DML). While monitoring the Cassandra database node, administrators may often want to analyze which communication protocol is effectively used by the Cassandra clients to access the server. For this purpose, eG Enterprise offers the **Cassandra Clients** test.

This test reports the number of clients accessing the target database server through the native protocol and the thrift protocol. By comparing the count of these measures, administrators can figure out the protocol that is used frequently by the clients to communicate with the server.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Connected native clients	Indicates the number of clients connected to the database node through the native protocol.	Number	Compare the value of this measure with the Connected thrift clients measure to figure out if the clients preferred native protocol over thrift protocol.
Connected thrift clients	Indicates the number of clients connected to the database node through the thrift protocol.	Number	Compare the value of this measure with the Connected native clients measure to figure out if the clients preferred thrift protocol over native protocol.

3.4.3 Cassandra Long Running Query Test

This test tracks the currently executing queries on a Cassandra database node and determines the number of queries that have been running for a long time. Using this test, resource-intensive queries to a database can be quickly isolated.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
CQL User	Specify the name of the CQL User with the following privileges in this text box: SELECT privilege on the system_traces.sessions table SELECT privilege on the system.peer_events table
CQL Password	Specify the password corresponding to the CQL User.
Confirm Password	Confirm the CQL Password by retyping it in this text box.
Elapsed Time (seconds)	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.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of queries	Indicates the number of queries that are currently executing on the database beyond 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.

3.4.4 Cassandra Messages Test

When a new Cassandra database node is added to an existing Cassandra database cluster,administrators transfer the data to the new node by executing the *{{netstats}}* command on the node from which data should be sent to the new node. This command helps in data transfer between the nodes when the node is joining the cluster. If the data transfer is not done completely or is partially completed, the data in the new node may be stale or outdated. When users access data from that particular node, the data may either be incorrect or the data may not be available at all. To alleviate such data transfer issues and ensure that the data in the new node joining the cluster is up to date, it is important to monitor the messages that are being transferred to each new node. The **Cassandra Messages** test helps administrators in this regard!

This test auto-discovers the nodes that are joining the target Cassandra database node in a cluster and for each node, this test reports the rate at which large messages, small messages and gossip messages were transferred from the target database node. In addition, this test also reveals the rate at which large/small/gossip messages were pending transfer to the joining node and were dropped during transfer. This test also reveals how many messages timed out during transfer. Using this test, administrators can figure out the joining node that is not up to date with the target database nose.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for each node clustered with the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.

Parameters	Description
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Large messages completed	Indicates the rate at which large messages were completely transferred to this node during the last measurement period.	Messages/sec	A high value is desired for this measure.
Large messages dropped	Indicates the rate at which large messages were dropped while being transferred to this node during the last measurement period.	Messages/sec	Ideally, the value of this measure should be zero.
Large messages pending	Indicates the rate at which large messages were pending to be transferred from this node during the last measurement period.	Messages/sec	A sudden/gradual increase in the value of this measure is a cause of concern. This indicates that the data in the joining node is not up to date with the data available in the target database node.
Small messages completed	Indicates the rate at which small messages were completely transferred to this node during the last measurement period.	Messages/sec	A high value is desired for this measure.
Small messages dropped	Indicates the rate at which small messages were dropped while being transferred to this node during the last measurement period.	Messages/sec	Ideally, the value of this measure should be zero.

Measurement	Description	Measurement Unit	Interpretation
Small messages pending	Indicates the rate at which small messages were pending to be transferred to this node during the last measurement period.	Messages/sec	A sudden/gradual increase in the value of this measure is a cause of concern. This indicates that the data in the joining node is not up to date with the data available in the target database node.
Gossip messages completed	Indicates the rate at which gossip messages were completely transferred to this node during the last measurement period.	Messages/sec	A high value is desired for this measure.
Gossip messages dropped	Indicates the rate at which gossip messages were dropped while being transferred to this node during the last measurement period.	Messages/sec	Ideally, the value of this measure should be zero.
Gossip messages pending	Indicates the rate at which gossip messages were pending to be transferred to this node during the last measurement period.	Messages/sec	A sudden/gradual increase in the value of this measure is a cause of concern. This indicates that the data in the joining node is not up to date with the data available in the target database node.
Timeouts	Indicates the rate at which messages timed out while being transferred to this node during the last measurement period.	Timeouts/sec	A low value is desired for this measure. A sudden/steady increase in the value of this measure reveals that the messages were not transferred to the joining node successfully and the joining node does not contain all the data that needs to be transferred from the target database nodde.

3.4.5 Cassandra Message Drops Test

Cassandra has a concept of back pressure. Back pressure is a technique used in staged event-driven architecture (SEDA) architectures in which, if a stage is already full of requests, it will not accept requests from earlier stages. As a result of back pressure, Cassandra will drop already

timed-out requests without processing, and log an error. Since Cassandra is based on SEDA architecture, a request has to hop between multiple threadpools during processing, thereby increasing the latency. If too many message drops are noticed on the the Cassandra database node frequently, then, administrators have to check for overload conditions or figure out the real reason behind such performance lag at the earliest. To help the administrators, eG Enterprise offers the **Cassandra Message Drops** test.

The test auto-discovers the type of messages on the target Cassandra Database node and for each message type, reports the number of messages dropped and the time duration with which the messages were dropped from within the node and across the node. Using this test, administrators can figure out the messages of the message type that were dropped frequently and measure the maximum latency observed for the messages of each message type within the node and across nodes.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for each message type on the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Dropped messages	Indicates the rate at which messages of this type were dropped during the last measurement period.	Messages/sec	Ideally, the value of this measure should always be 0. This measure is a good indicator of load on the database node. Messages are dropped when the internode messages received by a node are not processed within their proper timeout. Load shedding is part of the Cassandra architecture. If this is a persistent issue, it is generally a sign of an overloaded node or cluster.
Dropped latency across nodes	Indicates the time taken for the messages of this type to be dropped from across the nodes.	Milliseconds	Ideally, the value of this measure should be 0. Compare the value of this measure across message type to figure out the messages of which message type were more prone to be dropped from across the nodes.
Dropped latency within node	Indicates the time taken for the messages of this type to be dropped from within the node.	Milliseconds	Ideally, the value of this measure should be 0. Compare the value of this measure across message type to figure out the messages of which message type were more prone to be dropped from within the node.

3.4.6 Cassandra Nodes Test

Cassandra is a distributed database system using a shared nothing architecture. A single logical database is spread across a cluster of nodes and thus the need to spread data evenly amongst all participating nodes. Data is evenly divided around its cluster of nodes. Each node is responsible for a portion of data.

If a node becomes unreachable, then, that portion of data may become unavailable, provided the data is not replicated to other nodes. Therefore, it becomes mandatory to monitor the nodes that are unavailable and unreachable. The **Cassandra Nodes** test helps administrators to keep track on the nodes constantly.

This test helps administrators to figure out the count of the nodes that are unavailable and unreachable. In addition, this test also reports the number of nodes that joined the cluster, left the cluster and migrated to another cluster. Using this information, administrators can figure out the nodes that are frequently unreachable and initiate further investigation to avoid the nodes from being unreachable too often.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent.

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.
Detailed Diagnosis	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.

Parameters	Description
	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
Available nodes	Indicates the number of nodes available in the cluster.	Number	The detailed diagnosis of this measure lists the IP address of the nodes that are available in the cluster.
Joining nodes	Indicates the number of nodes that joined the cluster.	Number	The detailed diagnosis of this measure lists the IP address of the nodes that joined the cluster.
Leaving nodes	Indicates the number of nodes that left the cluster.	Number	The detailed diagnosis of this measure lists the IP address of the nodes that left the cluster.
Moving nodes	Indicates the number of nodes that were migrating to another cluster from the cluster.	Number	The detailed diagnosis of this measure lists the IP address of the nodes that migrated to another cluster.
Unreachable nodes	Indicates the number of nodes that were unreachable in the cluster.	Number	The detailed diagnosis of this measure lists the IP address of the nodes that are unreachable. By identifying the nodes that are unreachable, administrators can initiate further investigation on why the nodes are unreachable and troubleshoots the issues related to those nodes.

3.4.7 Cassandra Services Test

Gossip is a peer-to-peer communication protocol in which nodes periodically exchange state information about themselves and about other nodes they know about. The gossip process runs every second and exchanges state messages with upto three other nodes in the cluster. The nodes exchange information about themselves and about the other nodes that they have gossiped about, so all nodes quickly learn about all other nodes in the cluster. A gossip message has a version associated with it, so that during a gossip exchange, older information is overwritten with the most current state for a particular node. Likewise, the native transport is the CQL Native Protocol that is the way all modern Cassandra Driver's communicate with the server. The native protocol defines the format of the binary messages exchanged between the driver and Cassandra over TCP. If these protocol services are not running, then, in large environments, a few of the Cassandra Database nodes may not be up to date. This may cause data inconsistencies in production environments. To avoid such data inconsistencies, it is important to keep a constant vigil on the protocol services and figure out if the protocol services are running at all times! The **Cassandra Services** test helps administrators in this regard!

This test monitors the Gossip protocol service and the Native protocol service and reports whether the services are running or not. Using this test, administrators can figure out the protocol service that is not running and rectify the same before data mismatch is noticed by the end users!

Target of the test : A Cassandra Database node

Agent deploying the test : An external/remote agent

Outputs of the test : One set of results for the target Cassandra Database node being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.

Parameters	Description
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Is Gossip running?	Indicates whether/not the Gossip protocol service is running.		The values reported by this measure and its numeric equivalents are mentioned in the table below: **Measure Value** / **Numeric Value**: Yes = 1; No = 0 **Note:** By default, this measure reports the **Measure Value**s listed in the table above to indicate whether/not the protocol service is running. The graph of this measure however, is represented using the numeric equivalents only i.e., 0 or 1.
Is Native transport running?	Indicates whether/not the native transport service is running.		The values reported by this measure and its numeric equivalents are mentioned in the table below: **Measure Value** / **Numeric Value**: Yes = 1; No = 0

Measurement	Description	Measurement Unit	Interpretation
			Note: By default, this measure reports the **Measure Value**s listed in the table above to indicate whether/not the transport service is running. The graph of this measure however, is represented using the numeric equivalents only i.e., 0 or 1.

3.4.8 Cassandra Thread Pools Test

Cassandra is based on a Staged Event Driven Architecture (SEDA). Cassandra separates different tasks into stages connected by a messaging service. Cassandra maintains distinct thread pools for different stages of execution. Each like task is grouped into a stage having a queue and thread pool . Some stages skip the messaging service and queue tasks immediately on a different stage if it exists on the same node. Each of these queues can be backed up if execution at a stage is being over run. This is a common indication of an issue or performance bottleneck. If too many tasks are pending in a thread pool, then administrators should be alerted about those tasks so that they can either further investigate the problem or add additional threads in that thread pool. The **Cassandra Thread Pools** helps administrators in this process!

This test auto-discovers the thread pools of each stage on the target Cassandra Database node and for each thread pool, reports the number of active tasks and the tasks that were completed per second. In addition, this test reveals the tasks that were pending and the rate at which the tasks were blocked per second. Using this test, administrators can determine the thread pool containing the maximum number of active threads.

Target of the test : A Cassandra Database

Agent deploying the test : An external/remote agent

Outputs of the test : One set of results for each *Stage:Thread pool* on the target Cassandra Database node that is being monitored.

Configurable parameters for the test

Parameters	Description
Test Period	How often should the test be executed.
Host	The host for which the test is to be configured.
Port	The port on which the specified host listens. By default, this is 9042.
JMX Remote Port	Here, specify the port at which the JMX listens for requests from remote hosts. Ensure that you specify the same port that you configured in the **cassandra-env.sh** file (if the target Cassandra Database node is installed on a Unix host) or the **cassandra-env.ps1** file (if the target Cassandra Database node is installed on a Windows host) in the **<CASSANDRA_HOME>** directory used by the target Cassandra Database node. To know how to specify the remote port, refer to Section **2.2**.
JMX User and JMX Password	If JMX requires **authentication only (but no security)**, then ensure that the user and password parameters are configured with the credentials of a user with read-write access to JMX. To know how to create this user, refer to Section **2.3**.
Confirm Password	Confirm the Password by retyping it in this text box.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active tasks	Indicates the number of active tasks in this thread pool.	Number	Compare the value of this measure across the thread pools to figure out the thread pool that contains the maximum number of threads that are active.
Completed tasks	Indicates the rate at which tasks were completed in this thread pool.	Tasks/sec	Compare the value of this measure across the thread pools to figure out the thread pool that has completed the maximum number of tasks per second.
Current blocked tasks	Indicates the number of tasks that are currently blocked in this thread pool.	Number	A high value for this measure indicates that there are no more threads in the thread pool to service the tasks. To avoid the tasks from being blocked, administrators can calibrate the thread pool to accommodate enough threads to service the tasks.

Measurement	Description	Measurement Unit	Interpretation
Pending tasks	Indicates the number of tasks that are pending in this thread pool.	Number	Compare the value across the thread pools to identify the thread pool on which the maximum number of tasks are pending. A sudden/gradual increase in the measure is an indication for the administrators to add additional threads to the thread pool.
Total blocked tasks	Indicates the rate at which tasks were blocked in this thread pool during the last measurement period.	Tasks/sec	A low value is desired for this measure.

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.