

# IBM Tivoli Netcool/OMNIbus 7.3.1

## ObjectServer key performance indicators



© 2013 IBM Corporation

In this training module, you learn how to use ObjectServer key performance indicator measurements to monitor system performance. You also learn how to limit the impact of alert table size on system performance.

## Objectives

When you complete this module, you can perform these tasks:

- Name the ObjectServer key performance indicators used to create a system baseline measurement
- Enable and use the stats\_trigger trigger group to monitor connections
- Monitor the volume of events and deduplications with the alerts.status table KPI
- Monitor the amount of processing time and memory of the ObjectServer process
- Measure memstore usage in the catalog.memstores table
- Analyze the profiler report to determine both the comprehensive and individual impact of triggers on granularity

When you complete this module, you can perform these tasks:

- Name the seven ObjectServer key performance indicators
- Enable and use the stats\_trigger trigger group to monitor connections
- Monitor the volume of events and deduplications with the alerts.status table KPI
- Monitor the amount of processing time and memory of the ObjectServer process
- Measure memstore usage in the catalog.memstores table
- Analyze the profiler report to determine both the comprehensive and individual impact of triggers on granularity

## Using ObjectServer KPIs to measure system resource utilization

Key performance indicators (KPIs) are used to determine the root causes of system performance issues:

- Key performance indicators are used to check on objectserver performance
- Establish KPI baseline measurements of the system under its normal operational load
- KPI measurement level increases indicate system resource utilization increases
- Workload increases can impact system performance

Comparing the baseline KPI measurements to current KPI measurements determines whether the system is experiencing a higher-than-normal resource utilization workload

To ensure that IBM Tivoli® Netcool/OMNibus is performing well, you can monitor several key performance indicators. You can also monitor key performance indicators to check on ObjectServer performance. When using the key performance indicators, first establish a baseline on the system. The system baseline is the KPI measurements of system resources being used under their normal operational load. These KPIs are used to indicate the possibility of system performance changes as a direct result of changes in the system resource utilization workload. Comparing the baseline KPI measurements to current KPI measurements determines whether the system is experiencing a higher-than-normal resource utilization workload.

## Limiting alert table row quantity to boost system performance

The number of rows in these tables can impact system performance:

- alerts.status
- alerts.journal
- alerts.details

The number of rows in alerts.status, alerts.journal, and particularly, the alerts.details tables can affect ObjectServer performance. Only use the alerts.details table when alerts.status is not large enough to hold all information for a specific alarm or during rules file development. Try to keep the alerts.details table below 5,000 rows. Details statements in probe rules files are used to generate records into the alerts.details table. The statement details (\$\*) record each token as one row into alerts.details. If you have details(\$\*) set in your rules file, for each event in alerts.status table, there can be approximately 10 to 50 rows in the alerts.details table. Details can be disabled by commenting out any details (\$\*) statements in all your probe rules files, restarting all probes, and clearing the current records in the details table (delete from the alerts.details table).

## Monitor the volume of events in the alerts.status

The throughput volume of events on the alerts.status table is a KPI:

- Includes new inserts
- Includes deduplications
  - Deduplication triggers prevent the number of rows from increasing
  - Deduplication still takes up processing time in the ObjectServer
  - As events are deduplicated, the alertTally field value increases

Another key performance indicator is the volume of events in alerts.status, including new inserts and deduplications. Deduplication triggers prevent the number of rows from increasing, but still take processing time in the ObjectServer. The ObjectServer Tally field indicates the number of deduplications.

## Managing the number of ObjectServer connections

The maximum permitted number of connections is determined by the ObjectServer Connections property:

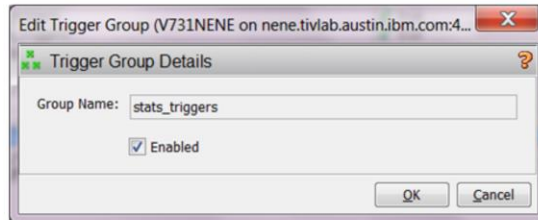
- The default is 30 connections
- The number of connections is always increased from the default value in production environments
- A finite number of connections can be made to the ObjectServer
  - The maximum number of connections is 1024
  - An error is logged in the ObjectServer log when connections are exceeded

Connections can be monitored through the stats\_triggers in the master.stats table

The number of ObjectServer connections is another key performance indicator. The maximum permitted number of connections is determined by the ObjectServer Connections property, with a default of 30. The Connections property of the ObjectServer should always be increased when the server is first brought into a production environment. Only a finite number of connections can be made to an ObjectServer. The maximum number of connections is 1024. An error is logged in the ObjectServer log when the allowed number of connections is exceeded. Connections can be monitored through stats\_triggers in the master.stats table.

## Monitoring row count, volume size, and connections quantity

Enable the stats\_triggers trigger group



Row count on alerts.status, alerts.journal, alerts.details, volume of alerts.status and the number of connections can be monitored through enabling the stats\_triggers trigger group. Ensure that stats\_triggers is enabled. The stats\_triggers group uses triggers to gather several statistics and metrics in the default ObjectServer configuration. To enable stats\_triggers log in to the OMNibus Administrator desktop client. Edit the trigger group and ensure the **Enabled** check box is checked.

## Monitoring row count, volume size, and connection quantity

The triggers in the stats\_triggers trigger group count these:

- Current event counts on alerts.status, alerts.details and alerts.journal
- Number of inserts to the alerts.status, alerts.details and alerts.journal
- Number of deduplications and new inserts to alerts.status
- Number of client connections

|   | DetailCount | StatusInserts | StatusNewInse... | StatusDedups | JournalInserts | DetailsInserts |
|---|-------------|---------------|------------------|--------------|----------------|----------------|
| 0 | 0           | 0             | 0                | 0            | 0              | 0              |
| 0 | 19          | 19            | 0                | 0            | 0              | 0              |
| 0 | 22          | 22            | 0                | 0            | 0              | 0              |
| 0 | 26          | 25            | 1                | 0            | 0              | 0              |
| 0 | 28          | 27            | 1                | 0            | 0              | 0              |
| 0 | 0           | 0             | 0                | 0            | 0              | 0              |
| 0 | 19          | 19            | 0                | 0            | 0              | 0              |
| 0 | 22          | 22            | 0                | 0            | 0              | 0              |
| 0 | 26          | 25            | 1                | 0            | 0              | 0              |
| 0 | 28          | 27            | 1                | 0            | 0              | 0              |
| 0 | 0           | 0             | 0                | 0            | 0              | 0              |
| 0 | 19          | 19            | 0                | 0            | 0              | 0              |
| 0 | 22          | 22            | 0                | 0            | 0              | 0              |
| 0 | 26          | 25            | 1                | 0            | 0              | 0              |
| 0 | 28          | 27            | 1                | 0            | 0              | 0              |
| 0 | 0           | 0             | 0                | 0            | 0              | 0              |

8

ObjectServer key performance indicators

© 2013 IBM Corporation

Triggers in the stats\_triggers group count the current events, inserts, deduplications, and client connections. A new row is logged to the master\_stats table with this information by default every five minutes.



## Monitoring ObjectServer microprocessor usage

Monitor the microprocessor usage of the nco\_objserv process:

- If the ObjectServer is under heavy load, it is reflected in microprocessor usage
- The source of the heavy load is reflected in the profiler report and trigger statistics logs

The microprocessor usage of the ObjectServer process is a key performance indicator. If the ObjectServer is under heavy load, this is reflected in microprocessor usage. Microprocessor usage might spike as the ObjectServer processes a particular SQL statement, but the average usage over time should remain stable. If microprocessor usage increases, the profiler report and trigger statistics logs show the source of the heavy load.

## Monitoring ObjectServer memory usage

Memory usage of nco\_objserv process:

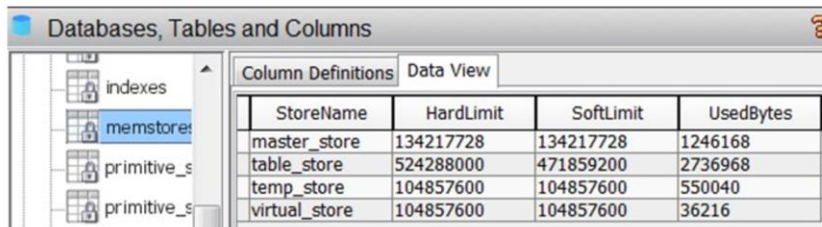
- The memory usage of the process increases proportionally to increases in the number of rows in the alerts.status table, alerts.details table, and the alerts.journal table (or any additional tables you have defined), to increases in the number of connections, and increased usage by clients
- The memory usage should remain stable over time
- Any increases should correspond to increases in the numbers of table rows or additional clients.

Memory usage of the ObjectServer process is another key performance indicator. The memory usage of the process increases proportionally to increases in the number of rows in the alerts.status table, alerts.details table, and the alerts.journal table (or any additional tables you have defined), increases in the number of connections, and increased usage by clients. Memory usage should remain stable over time. Any increases should correspond to increases in the number of table rows or additional client SQL commands.

## ObjectServer memstore usage

To monitor the memstore, inspect the content of the catalog.memstores table:

- For each row, compare the value of the UsedBytes column with the values of the SoftLimit column and the HardLimit column
- Once a memstore soft limit is exceeded, an error is logged to the ObjectServer log file
- Once the hard limit is exceeded, the ObjectServer shuts down



| StoreName     | HardLimit | SoftLimit | UsedBytes |
|---------------|-----------|-----------|-----------|
| master_store  | 134217728 | 134217728 | 1246168   |
| table_store   | 524288000 | 471859200 | 2736968   |
| temp_store    | 104857600 | 104857600 | 550040    |
| virtual_store | 104857600 | 104857600 | 36216     |

The last ObjectServer key performance indicator is memstore usage for the ObjectServer database. To monitor the memstore, inspect the content of the catalog.memstores table. For each row, compare the value of the UsedBytes column with the values of the SoftLimit column and the HardLimit column. Memstores are containers that are maintained by the ObjectServer. They contain ObjectServer data and tables in memory. Memstores have a finite size. You must ensure that the memstores do not become full, otherwise they do not allow any more data to be inserted. Once a memstore soft limit is exceeded, an error is logged to the ObjectServer log file. The ObjectServer shuts down once the hard limit is exceeded.

## ObjectServer Granularity

ObjectServer property Granularity:

- The default granularity period is 60 seconds
- The ObjectServer needs to service all client and trigger requests within the granularity period

The ObjectServer property Granularity specifies the update interval for Insert, Delete, Update, and Control (IDUC) notifications to clients. The default granularity period is 60 seconds. It is not recommended to change the default. The ObjectServer is designed to service all client and trigger requests within the granularity period.

## ObjectServer profiler report

- Profile statistics are also logged to a profile log file  
\$NCHOME/omnibus/log/ObjectServername\_profiler\_report.log#
- The profiler report shows a breakdown of the time spent for each client connection and the total time spent by client type, for each granularity period
- You can use the profile log file to analyze how the ObjectServer time is spent during each granularity period and calculate the percentage of time used
- For example, if the granularity period is set to 60 seconds and the total time spent for all the connections during a particular period was 30 seconds, you can calculate that the ObjectServer spent 50% of its available time on running SQL commands from client connections

Profile statistics are also logged to the file shown under the first bullet in this slide. The profiler report shows a breakdown of the time spent for each client connection and the total time spent by client type, for each granularity period. You can use the profile log file to analyze how the ObjectServer time is spent during each granularity period and calculate the percentage of time used. For example, if the granularity period is set to sixty seconds and the total time spent for all the connections during a particular period was 30 seconds, you can calculate that the ObjectServer spent fifty percent of its available time running SQL commands from client connections. Profiler report logging is enabled by default in IBM Tivoli Netcool/OMNibus V7.3.1. In earlier versions of IBM Tivoli Netcool/OMNibus the ObjectServer property Profile had to be manually set to TRUE in order to create the profiler report log.

## Analyzing the profiler report

Goal: Profiler Report "Total time in the report period" + Trigger Statistics "Time for all triggers" < ObjectServer Granularity period of 60 seconds

- Current Total time in the report period of **29.980000s** indicates ~50% of ObjectServer Granularity period is taken services client connections.
- Identify the highest clients in the report period:

Mon Oct 12 17:39:46 2009: 'e@c0B4D@c0142:11.0' (uid = 0) time on **omnihost1.ibm.com: 10.010000s**

Mon Oct 12 17:39:46 2009: 'e@c0B4D@c0142:11.0' (uid = 45) time on **omnihost1.ibm.com: 9.870000s**

- High probes can indicate an event flood or large number of status or details inserts
- High event lists or WebGUI clients can indicate inefficient filters
- High ObjectServer gateway clients can indicate transfer of many events or a resync

The goal is that the profiler report "total time in the report period" plus the trigger statistics "time for all triggers" is less than the ObjectServer granularity period of sixty seconds. Current total time in the report period of **29.980000s** indicates that approximately fifty percent of the ObjectServer Granularity period is taken to service client connections. That does not include the current time for trigger load. To troubleshoot, you must identify the highest-used clients in the report period. In this example, an event list client with two connections was taking 19 seconds of the period. High probe clients can indicate an event flood or large number of either status or details inserts. High event lists or WebGUI clients can indicate inefficient filters. High usage of ObjectServer gateway clients can indicate either a transfer of many events or a resync.

## ObjectServer trigger statistics

- Trigger statistics are also logged to the file  
\$NCHOME/omnibus/log/ObjectServername\_trigger\_stats.log#
- The trigger statistics log file shows the amount of time that each trigger has used in the last profiling period
- You can use this log file for automation debugging, and to determine which triggers are slow due to slow-running SQL queries

Trigger statistics are also logged to the file shown under the first bullet in this slide. The trigger statistics log file shows the amount of time that each trigger has used in the last profiling period. You can use this log file for either automation debugging or to determine which triggers are slow due to slow-running SQL queries.

## Example trigger statistics

```
Mon Oct 12 18:03:56 2009: Trigger Profile Report
....
Mon Oct 12 18:03:56 2009: Trigger Group 'primary_only'
Mon Oct 12 18:03:56 2009: Trigger time for 'generic_clear': 5.879707s
Mon Oct 12 18:03:56 2009: Trigger time for 'expire': 0.008233s
Mon Oct 12 18:03:56 2009: Trigger time for 'delete_clears': 0.007219s
Mon Oct 12 18:03:56 2009: Trigger time for 'enrich_and_correlate': 23.007219s
...
Mon Oct 12 18:03:56 2009: Trigger Group 'iduc_triggers'
Mon Oct 12 18:03:56 2009: Trigger time for 'disconnect_iduc_missed': 0.000000s
Mon Oct 12 18:03:56 2009: Trigger time for 'iduc_stats_update': 0.000949s
Mon Oct 12 18:03:56 2009: Trigger time for 'iduc_messages_tbclean': 0.000089s
Mon Oct 12 18:03:56 2009: Trigger time for 'deduplicate_iduc_stats': 0.000000s
Mon Oct 12 18:03:56 2009: Trigger time for 'iduc_stats_insert': 0.000000s
Mon Oct 12 18:03:56 2009: Trigger Group 'automatic_backup_system'
Mon Oct 12 18:03:56 2009: Trigger time for 'backup_succeeded': 0.000000s
Mon Oct 12 18:03:56 2009: Trigger time for 'backup_failed': 0.000000s
Mon Oct 12 18:03:56 2009: Trigger time for 'backup_state_integrity': 0.000000s
Mon Oct 12 18:03:56 2009: Trigger Group 'gateway_triggers'
Mon Oct 12 18:03:56 2009: Trigger time for 'resync_finished': 0.000000s
Mon Oct 12 18:03:56 2009: Time for all triggers in report period (60s): 29.789663s
```

Here is an example trigger statistics log. Individual trigger statistics are logged for each trigger. Trigger time should normally be in the subseconds. The time for all triggers is logged every 60 seconds. In this example, all of the triggers combined consume 29.78 seconds of the 60-second granularity period.



## Analyzing trigger statistics

- Goal: profiler report “Total time in the report period” + Trigger Statistics “Time for all triggers” < Granularity period of 60 seconds
- Current profiler report total time in the report period of 29.980000s + Trigger Statistics Time for all triggers 29.789663s indicates ObjectServer is at 100% utilization of 60 second granularity period
- Identify the highest triggers:
  - Mon Oct 12 18:03:56 2009: Trigger time for 'enrich\_and\_correlate': 23.007219s
  - Mon Oct 12 18:03:56 2009: Trigger time for 'generic\_clear': 5.879707s
- High generic\_clear or deduplication triggers can indicate high event throughput or high number of resident events
- Ensure best practices are used in creating custom triggers  
[http://publib.boulder.ibm.com/infocenter/tivhelp/v8r1/topic/com.ibm.netool\\_OMNibus.doc\\_7.3.1/omnibus/wip/admin/reference/omn\\_admin\\_per\\_bestpractices/triggers.html](http://publib.boulder.ibm.com/infocenter/tivhelp/v8r1/topic/com.ibm.netool_OMNibus.doc_7.3.1/omnibus/wip/admin/reference/omn_admin_per_bestpractices/triggers.html)
- Ensure trigger execution time is kept to a minimum, no other writes can be performed in the ObjectServer when a trigger is executed

Again, the goal is that the profiler report “total time in the report period” plus the trigger statistics “time for all triggers” is less than the ObjectServer granularity period of sixty seconds.

The current profiler report total time in the report period of **29.980000s** plus trigger statistics time for all triggers of **29.789663s** indicates that the ObjectServer is at one hundred percent utilization of the sixty second granularity period. To troubleshoot performance, identify the highest usage triggers. Typically any single trigger over five seconds is considered a high usage trigger. In this case, the two triggers enrich\_and\_correlate, and generic\_clear which are running high.

High generic\_clear or deduplication triggers can indicate high event throughput or a high number of resident events.

Ensure best practices are used in creating custom triggers.

Ensure trigger execution time is kept to a minimum. No other writes can be performed in the ObjectServer when a trigger is executed.

## Review of ObjectServer key performance indicator monitoring

- Enable Stats triggers
  - Number of rows in alerts.status, alerts.journal and alerts.details
  - Number of inserts in the alerts.status table
  - Number of connections
- System monitors
  - Microprocessor usage of nco\_objserv
  - Memory usage of nco\_objserv
- ObjectServer catalog.memstores table
  - Memstore usage
- ObjectServer profiler report and trigger statistics logs
  - ObjectServer granularity
    - Profiler report
    - Trigger stats

To monitor the ObjectServer key performance indicators, the stats\_trigger trigger group will track the number of rows in the alerts tables, the number of inserts and deduplications on alerts.status, and the number of client connections. Use system monitors to track microprocessor and memory usage of the nco\_objserv process. Use the ObjectServer catalog.memstores table to monitor memstore usage. Use the profiler and trigger statistics logs to gauge ObjectServer granularity.

## Summary

Now that you have finished this training module, you can perform these tasks:

- Name the seven ObjectServer key performance indicators
- Enable and use the stats\_trigger trigger group to monitor connections
- Monitor the volume of events and deduplications with the alerts.status table KPI
- Monitor the amount of microprocessor and memory used by the ObjectServer process
- Measure memstore usage in the catalog.memstores table
- Analyze the profiler report to determine both the comprehensive and individual impact of triggers on granularity

Now that you have finished this training module, you can accomplish these tasks:

- Name the seven ObjectServer key performance indicators
- Enable and use the stats\_trigger trigger group to monitor connections
- Monitor the volume of events and deduplications with the alerts.status table KPI
- Monitor the amount of microprocessor and memory of the ObjectServer process
- Measure memstore usage in the catalog.memstores table
- Analyze the profiler report to determine both the comprehensive and individual impact of triggers on granularity

## Trademarks, disclaimer, and copyright information

IBM, the IBM logo, [ibm.com](http://www.ibm.com), and Tivoli are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2013. All rights reserved.