SAP® MaxDB™ Expert Session

SAP® MaxDB™ Performance Optimization

Christiane Hienger
IMS MaxDB/liveCache Development Support
2015
News - General Remarks

• Some functionality might have been changed or expanded over the years.
• Recommendations might have changed.

➔ For each expert session an update with the most important changes is published.

• Check the additional remarks after having heard or read the expert session.
• Feedback to the responsible author is welcome if you recognize outdated information within the session.
Agenda

1. Performance Analysis: Parameter Setting
2. News: Command Monitor
EnableIOTimeStatistics
Setting this parameter to 'YES' starts collection of timing information used for monitoring I/O throughput. Since it is online changeable, it is possible to enable or disable collection of time information during runtime. Measuring time of I/O operations adds only a minimal overhead, therefore it is turned on by default.

UseExtendedTimeMeasurement
This parameter allows to enforce extended time measurement with database start. The default value for this database parameter is YES (enable time measurement).

Time measurement:
- is necessary to provide useful statistics regarding command monitoring
- allows the Database Analyzer to collect more detailed statistics

Note that command and resource monitoring is not possible when time measurement is switched off.

Time measurement can be switched off with the console command TIME DISABLE or by setting this database parameter to NO, which you can do in ONLINE mode.
Performance Analysis: Parameter Setting (2)

**UseSharedSQL**
Enables the use of SharedSQL

'YES': SharedSQL is used for caching SQLCommands
'NO': SharedSQL will not be used

**Note that command and resource monitoring is not possible if Shared SQL is switched off.**

**SharedSQLCommandCacheSize**
Specifies the maximum size (in KB) for SharedSQL statement caching

The lower and upper limits:
32768 (=32MByte) <= UseSharedSQLCommandCacheSize < 8388608 (=8GByte)

DEFAULT = 262144
Agenda

1. Performance Analysis: Parameter Setting

2. News: Command Monitor
As of SAP MaxDB Version 7.9 the image of the Command Monitor has been changed.

Use the Command Monitor if the analysis of the database bottlenecks (Database Analyzer) reports inefficient database accesses. With this tool, you can identify long running SQL statements in a systematic way.

The tool should be used for short-term analysis. You can filter the SQL statements by first specifying logging criteria. This enables you to concentrate on long-running SQL statements.

Enter the desired recording criteria in the Change value display. Now you can determine which SQL statements are to be logged in the command monitor tables. If one of the set criteria is fulfilled the SQL statement is logged in the command monitor.

Number of Pages Read: An SQL statement is logged if the number of specified page accesses is exceeded.
Runtime of SQL Statement in ms: An SQL statement is logged if the specified runtime is exceeded.
Selectivity in %: An SQL statement is logged in the command monitor tables if the ratio of qualified records to read records falls below the specified percentage.

SAP provides default values for these thresholds. Use Enter Default Values and then adapt or confirm these values.

You have to explicitly activate the monitoring criteria with Activate Values.
With the new Command Monitor a more detailed analysis is possible.

You will now get a detailed measurement where the runtime of a command was spent:

- **CPU-Time**: % of CPU usage
- **IO Time**: wait for IO operations in %
- **Wait-Time**: Wait situations in status: Sleep/Suspend/Wait in %

These new statistic values are displayed by default in the monitoring screen. There are a lot more detailed statistical values which can be selected and displayed by choosing *Change Layout*. 
Command Monitor: More Statistics in Change Layout

10

IOWAITREADTIME
IOWAITWRITETIME
PHYSICALREADS, PHYSICALWRITES, PHYSICALPAGESREAD, PHYSICALPAGESWRITTEN
FETCHCOUNT, Number of Rows Found Using a FETCH
SUSPENDCOUNT, WAITCOUNT
READPAGECOUNT, WRITEPAGECOUNT
READAHEADREQUESTCOUNT, READAHEADIGNOREDREQUESTCOUNT, READAHEADPHYSICALREADPAGECOUNT
DISPATCHCOUNT, TASKSELF_SUSPENDCOUNT
REGIONACCESSCOUNT
SELFIOREADCOUNT
SELFIOREADPAGECOUNT
AVGSELFIOREADABSOLUTETIME
SELFIOREADABSOLUTETIME
SELFIOWRITECOUNT
SELFIOWRITEPAGECOUNT
AVGSELFIOWRITEABSOLUTETIME
SELFIOWRITEABSOLUTETIME
IOREADCOUNT
IOREADPAGECOUNT
AVGIOREADRELATIVETIME
IOREADRELATIVETIME
AVGIOREADABSOLUTETIME
IOREADABSOLUTETIME
IOWRITECOUNT
IOWRITEPAGECOUNT
AVGIOWRITERELATIVETIME
IOWRITERELATIVETIME
AVGIOWRITEABSOLUTETIME
IOWRITEABSOLUTETIME
AVGSUSPENDRELATIVETIME
SUSPENDRELATIVETIME
AVGSUSPENDABSOLUTETIME
SUSPENDABSOLUTETIME
AVGWAITRELATIVETIME
AVGWAITABSOLUTETIME
WAITABSOLUTETIME
SLEEPCOUNT
AVGSLEEPLATIVETIME
SLEEPRELATIVETIME
AVGSLEEPABSOLUTETIME
SLEEPABSOLUTETIME
RUNNINGCOUNT
AVGRUNNINGABSOLUTETIME
RUNNINGABSOLUTETIME
SCHEDULINGCOUNT
AVG_SCHEDULING_ABSOLUTETIME
SCHEDULINGABSOLUTETIME
The following IO statistic values are not part of the default view. It is possible to select them via Choose Layout.

In this slide the following columns have been additionally selected:

IOWAITREADTIME, IOWAITWRITETIME, PHYSICALREADS, PHYSICALWRITES, PHYSICALPAGESREAD, PHYSICALPAGESWRITTEN

The unit is microseconds.

You can exactly see if the runtime was spent in IO.
In this example we can see the total runtime of the commands in the first column.

The command in the first line ran 1.7 seconds and nearly the total runtime was spent with waiting for IO to read the data: For 225 physical reads, which read 3263 physical pages.

This command did not perform any write IO. The execution time includes 4.38% CPU time but the main part of the execution time (95.76%) was spent with I/O.

=> the high runtime of this command was caused by the time spent for IO

Can we get more details about the time spent for IO? Yes we can – see next slide
If we analyze a long running SQL command whose runtime was caused by IO time mainly, we can get more detailed information about where the time exactly was spent.

*Background information*: SAP MaxDB Kernel is running in one OS process which consists of several threads. Threads which contain user tasks are so called User Kernel Threads (UKT). SQL statements are executed by user tasks. Several user tasks are running in the same User Kernel Thread. Only one task can have the CPU at one time. To avoid a CPU bottleneck in the UKT if a task requests IO, the User Kernel Thread does not execute the IO request by itself but orders special threads, so called IO threads which are executing the IO requests.

If no other task in the same UKT is active e.g. the others are in status Connect Wait, the task is able to do the IO by itself. This action is called self-IO.

If a task has done self-IO the following statistics are of interest:
- SELFIOREADABSOLUTETIME
- SELFIOWRITECOUNT
- SELFIOWRITEPAGECOUNT
- AVGSELFIOWRITEABSOLUTETIME
- SELFIOWRITEABSOLUTETIME

In most cases the tasks are not doing self-IO. The IO orders are done via the IO threads. Then the following additional columns are of interest and should be selected:
- IOWAITREADTIME
- IOREADRELATIVETIME
- IOREADABSOLUTETIME
- AVGIOREADRELATIVETIME
- AVGIOREADABSOLUTETIME

The Command Monitor displays values of relative and absolute Read time of each command.

*Absolute time*: Measurement starts when the task sends an IO request and stops when the IO complete information reaches the user task.

*Relative time*: Measurement starts when the IO workerthread sends the IO request to the IO sub system.
In this example there is a command which has no IO time but 99.6 % CPU time. You can get more detailed information e.g. by listing the column: RUNNINGABSOLUTETIME – showing the time the CPU was in use by this command.
Thank you

Contact information:

Heike Gursch
IMS MaxDB / liveCache
Heike.Gursch@sap.com

Christiane Hienger
IMS MaxDB / liveCache
Christiane.Hienger@sap.com