Archive - tuning and debugging

Archive - performance tuning

This section contains tips for debugging archive performance.

- The SV._System_ArchivPerformance system structure is intended for monitoring the performance of the archive - see the description of individual columns. We recommend archiving the PendingDbRequest, PendingStatReqest, PerformedDbRequest, and PerformedCalcRequest columns (with an appropriate filter) and possibly creating statistical historical values on top of them (eg 1-minute and hourly summary).

- The PENDING_REQUESTS command copies the archive request queue to a text file. It can then be analyzed for anomalies.

- The STATISTICS command allows you to find out the number of values that have been stored in the database for each archive object in the last N hours. The output of the command is a text file, which can then be imported, e.g. to Excel, and sorted by the last column. Subsequently, it is possible to optimize the archives with the largest number of values (setting filters, or limiting the number of calculations for the calculated archives).

- The STATISTICS command does not capture if one value has been changed multiple times (for example, inserting values from a script). Therefore, there are *DBG.ARCHIV.MANUAL_INSERT_VALUE* and *DBG.ARCHIV.MANUAL_UPDATE_VALUE* debug categories that can be set by the D2000 System Console process in the Debug info window. These categories turn on the display of INSERTARCHARR or UPDATEARCHVAL actions (and about manual modifications of values from D2000 HI). The IDs of the archive objects and the inserted values are displayed (in the case of INSERTARCHARR action the first and last value from the inserted list).

- Another problem may be that repeatedly inserted values from the script cause a large number of subsequent recalculations. In this case, it is advisable to first read (GETARCHVAL) and test whether the value is not already stored in the archive before inserting the value from the script. There is also a debug category *DBG.ARCHIV.MANUAL_READ_BEFORE_IU*, which enables this functionality directly in the handling of INSERTARCHARR or UPDATEARCHVAL actions and debug category *DBG.ARCHIV.MANUAL_READ_BEFORE_IU.DBG*, which activates listings when the value is already found in the archive. Using these debug listings, it is possible to find out for which objects (inserted from the script) it makes sense to implement read-before-write. The listings have the form
Insert value $XXX (NAME) Count= N First value=VALUE_DETAILS Last value=VALUE_DETAILS
Update value $XXX Action=MANARCHINSERT VALUE

- Debug category *DBG.ARCHIV.MONITOR.MV* activates the listings of the oldest values, which are inserted by the actions INSERTARCHARR or UPDATEARCHVAL actions. The listings have the form
ManualValues delay (Update) XXX sec for ID $YYY: VALUE_DETAILS
resp.
ManualValues delay (Insert) XXX sec for ID $YYY: VALUE_DETAILS
Using this category (activation, deactivation) it is possible to find out whether old values from the script are inserted into the archive, which subsequently causes recalculations of dependent archive objects.

- Debug category *DBG.ARCHIV.MONITOR.NV* activates the listings of the oldest values that come to the archive as values of objects that go to the primary archives. The listings have the form
NewValues delay XXX sec for ID YYY
With the help of this category (activation, deactivation) it is possible to find out whether values with the old time stamp go to the archive (e.g. from communication), which subsequently causes recalculations of dependent archive objects.
Note: the longest recorded delay is displayed after the debug category is turned off.

- Debug category *DBG.ARCHIV.MONITOR.OV* activates the listings of the oldest values that come to the archive as old values (from the communication, from the gateway) that go to the primary archives. The listings have the form
OldValues delay XXX sec for ID $YYY, source object $ZZZ
With the help of this category (activation, deactivation) it is possible to find out whether old values go to the archive, which subsequently causes recalculations of dependent archive objects.
Note: the longest recorded delay is displayed after the debug category is turned off.

- Debug category *DBG.ARCHIV.OLDVAL.COMMIT* is related to *DBG.ARCHIV.MONITOR.OV*. It activates listings for application "commits" (OLDVALUE COMMIT), which follow the insertion of old values (read from communication or from the gateway) that go to the primary archives. The listings have the form
Old value commit TransactId XXX <time_from, time_to>
Using this category, it is possible to find out for which period the old values go to the archive, which subsequently causes recalculations of dependent archive objects (also for this period).

- Archive performance on the Linux platform (especially for computed and statistical archives) can be significantly increased by deploying the optimized unixODBC library libodbc.so.2.0.0. In addition, we recommend deploying archive patches from October 2021 and later that have implemented ODBC cursor recycling, which addresses the unixODBC library slowness caused by using data structures (linear lists) unsuitable for applications with tens of thousands of precompiled cursors.

- On all platforms (operating systems), we recommend using the PostgreSQL database for archiving purposes and paying attention to the correct setting of its parameters.

- Acceleration of the archive reading (and thus also when recalculating statistical and computed archives) can be achieved by turning on the isochro nous cache.
- Acceleration of the archive writing can be achieved by increasing the number of writing tasks - parameter WriteThreadsCount. This modification requires High Performance Archive license option.
- Acceleration of the archive reading can be achieved by increasing the number of reading tasks - ReadThreadsCount parameter. This modification requires High Performance Archive license option.
- We recommend the activation of time slices at least for structured archives (DataTableSlices=2). Time slices speed up and simplify the maintenance of data tables and indexes, especially in the case of structured archives.
- The ReportLongRecalc parameter can be used to detect interval recalculations over a period longer than the specified value (in seconds). Such recalculations may be due to data with too old time (or the existence of data that has time in the future).
- The FREEZE command allows you to stop the processing of data in the archive for a defined number of seconds and then process the accumulated requests. In this way, it is possible to perform an archive performance test (in the case of a D2000 system with redundant archives, ideally on a passive archive, so that the "freezing" of archiving does not affect the functionality).

Archive - debugging

This section contains tips for debugging individual archive objects.

- SHOW_DYN_INFO command is intended for the one-time display of information about a specific archive object.
- The DI ON command is used to activate debugging information about a specific archive object. Information on inserted values will be displayed, in the case of calculated and statistical archives also information about recalculation of values and recalculation of intervals. Activating the *DBG. ARCHIV.DATA* debug category for such archive objects turns on the display of calculated values as well as the values of source objects going to the calculation.
- Debug category *DBG.ARCHIV.GET_VALUE_FOR_TIME* activates the listings when reading a value for a specific time from the database.
- Debug category *DBG.ARCHIV.READ* activates the listings about external read requests from the archive (e.g. from the D2000 HI process or as a result of GETARCH* actions). It is thus possible to monitor the effect of reading (e.g. from a script) on the performance of the D2000 Archive.
- Debug category *DBG.ARCHIV.READ.TREZOR* activates the listings about the read requests from the depository databases. Since many depository databases can be connected, reading from the depository can significantly contribute to the load of the D2000 Archive, the database, and the I/O subsystem.
- Debug category *DBG.ARCHIV.READ.EMPTY_EXTERNAL* activates the listings about the read requests from the archive (e.g. from the D2000 HI process or as a result of GETARCH* actions) that did not return any data (i.e. concerned empty archives). Due to optimization in the archive, these requests will not cause a read from the archive database.
- Debug category *DBG.ARCHIV.READ.EMPTY_INTERNAL* activates the listings about the read requests generated directly by the archive (for calculations of calculated/statistical archives) that did not return any data (i.e. concerned empty archives). Due to optimization in the archive, these requests will not cause a read from the archive database.

Archive - debugging for developers

The following debugging categories are for D2000 developers.

- Debug category *DBG.ARCHIV.CACHE* activates the listings related to working with the archive cache (insertion, deleting).
- Debug category *DBG.ARCHIV.CACHE.HIT* activates the listings related to a failed read from the archive cache.
- Debug category *DBG.ARCHIV.COMPRESS.TREZOR* activates the listings related to depository data compression.
- Debug category *DBG.ARCHIV.DISK_SPACE* activates the listings about querying an occupied space in the archive database.
- Debug category *DBG.ARCHIV.SLICES* activates the listings regarding the creation of time slices.
- Debug category *DBG.ARCHIV.TRANSTREE* activates the listings related to the transaction tree. Due to the existence of several write tasks (see the WriteThreadsCount parameter), it is necessary to synchronize the recalculations of the calculated and statistical archive objects so that they take place in the correct order. The transaction tree is used for this.
- Debug category *DBG.ARCHIV.WARNTRANS* activates the listings related to the pending transactions in the transaction tree. In a properly functioning archive, all transactions are terminated correctly.
- Debug categories *DBG.ARCHIV.WRITECACHE* and *DBG.ARCHIV.WRITECACHE.DETAILS* activate the listings related to the "write cache". This is a short-term cache for storing values that have been written to the database but have not yet been committed (and are therefore not yet visible to other tasks).

- Debug category DBG.ARCHIV.WARN_ZERO activates the listings by comparing values if they are almost identical (see AlmostZero parameter).