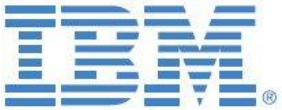


IBM Planning Analytics: Parallel Processing with Multi-Threaded Queries (MTQ) - Configuration Options, Scenarios & Recommendations

Andreas Kugelmeier
Executive Consultant, FOPM
Planning Analytics Architect
IBM Data and AI Expert Labs
Mobile Phone: +1-215-384-7302
Email: kugelmeier@us.ibm.com

Notices & Disclaimers

Information in these presentations and papers (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS document is distributed "AS IS" without any warranty, either express or implied. In no event shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.
Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.
Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose.
The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services ®, Global Technology Services ®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli®, Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.
- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.
- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Document Version History

Date	Version	Author	Description
06/11/2015	1.0	Andreas Kugelmeier	
07/13/2015	1.1	Andreas Kugelmeier	Add information on parameters MTQ.OperationProgressCheckSkipLoopSize & MTQ.ImmediateCheckForSplit
07/24/2015	1.2	Andreas Kugelmeier	Additional information on caveats when using MTQ to speed up View generation via ViewConstruct
7/28/2015	1.3	Andreas Kugelmeier	Additional information on MTQ Logging Corrected/Improved section on MTQ Behaviour & Configuration Options as well as Recommendations on MTQ Configuration based on latest tests
7/29/2015	1.31	Andreas Kugelmeier	Add recommendation to increase/re-optimize VMM when employing MTQ
7/30/2015	1.32	Andreas Kugelmeier	Corrected VMM default as per http://www-01.ibm.com/support/docview.wss?uid=swg27024354
12/1/2015	1.34	Andreas Kugelmeier	Add footnote on VMM setting
12/9/2015	1.35	Andreas Kugelmeier	Minor edits
8/31/2016	1.36	Andreas Kugelmeier	Additions to section on Caching with MTQ (VMM and VMT configuration)
9/23/2016	1.4	Andreas Kugelmeier	Corrections on VMM, add section 1.5
10/20/2016	1.45	Andreas Kugelmeier	Addl. information on leveraging MTQ TI-processing of views
11/4/2016	1.5	Andreas Kugelmeier	Misc. edits and additions to section 1.5
01/03/2019	2.0	Andreas Kugelmeier	Misc. corrections and additional information on MTQ behavior
2/27/2019	2.1	Andreas Kugelmeier	Add Caveat re adequate HW-sizing
11/25/2019	2.2	Andreas Kugelmeier	Remove "*In cases where MTQ.CTreeWorkUnitMerge is enabled (= set to TRUE), the additional parameter MTQ.CTreeRedundancyReducer=TRUE (also a dynamic parameter) may be used to reduce query redundancies if applicable.*", because MTQ.CTreeRedundancyReducer=TRUE should not be used

Table of Contents

1. Multi-Threaded Queries (MTQ)

1.1 MTQ: what is it?

As of TM1 version 10.2, a new feature known as "multi-threaded queries", or MTQ, is available. MTQ allows TM1 to take advantage of multiple processor cores to significantly increase the speed at which TM1 can perform calculations and queries by allowing queries to split into multiple processing threads, effectively using a parallel processing regime to resolve queries significantly faster.

The performance improvement of Multi-Threaded-Queries is in an approximately linear relationship to the # of CPU cores that the TM1 query engine is allowed to utilize: Approximate TM1 10.2 Query Time = TM1 <10.2 Query Time / # of CPU Cores utilized for Multi-Threaded Queries.

1.2 MTQ Configuration

1.2.1 MTQ Behavior & Configuration Options

- MTQ is configured and enabled via corresponding entries in the tm1s.cfg file (MTQ=N, with N=#of processing Threads)
- The MTQ parameters in the tm1s.cfg are dynamic, i.e. the TM1 Server process does not have to be restarted after a change in the MTQ settings.
- The MTQ value specifies the total # of threads that TM1 can leverage for MTQ.
- If MTQ=1 or MTQ=0, MTQ is disabled
- TM1 10.2: If there is no MTQ entry in the tm1s.cfg file, MTQ is disabled.
- TM1 11 (Planning Analytics): If there is no MTQ entry in the tm1s.cfg file MTQ is set to the # of processors on the machine (MTQ=ALL)
- To set the value to the maximum number of cores available on a server, the setting MTQ=ALL can be used.
- Setting MTQ to a negative number will (MTQ=-N) will result in the # of MTQ threads being determined as follows: T=M-N+1 (where T= of threads to be used by MTQ, M= # of CPU cores available to TM1). For example, if your computer has 64 cores and you set MTQ=-10, the # of MTQ Cores will be T = 64-10+1 = 55
- **The MTQ value (MTQ=X, like MTQ=4) is the <u>total</u> thread pool for all users that the TM1 Server makes available for Multi-Threaded-Queries.**
  - ⇒ Example: If you set MTQ=4 on an 8-CPU machine, only 4 threads will be used in Total for MTQ. If two users run a query that leverages 4 MTQ cores, one user will get 1 non-MTQ thread, and the other user will get 1 non-MTQ Thread plus 4 MTQ Worker Threads.
  - ⇒ MTQ is not the MTQ pool per user, but for the TM1 Database server in total.
- An MTQ pool thread becomes 'available' to other users once it has finished.
  - ⇒ Example: If a query initially gets assigned 4 MTQ worker threads, this does not mean it will keep all those threads. One thread may finish and the query will continue with 3 worker threads. Or 2 threads will finish, but the TM1 MTQ engine will determine that the next 'part' of the query is best served with an additional 8 threads, resulting in 10 threads being used…
  - ⇒ As a particular MTQ worker threads is finished (right after 'commit' in TM1Top), the thread becomes available again, and it may be used by the same query (for a new worker thread), or by a different user query.

- MTQ very effectively leverages Hyper-Threaded Cores. If the CPU and OS supports Hyper-Threading, we strongly recommend to enable Hyper-Threading (some hardware requires Hyper-Threading to be enabled via a Bios setting) and include the hyper-threaded cores in the MTQ configuration consideration.
- MTQ is NOT dependent/reliant on the # of CPUs on the machine!
  - ⇒ MTQ will leverage multi-core processing capabilities by splitting and resolving queries via parallel worker threads
  - ⇒ If you set MTQ to ALL or leave it at the default (ALL), TM1 will set an MTQ thread pool equal to the number of cores on the machine.
  - ⇒ You can set MTQ to a value higher than the number of cores on the machine. You will see more worker threads than CPU cores. This is acceptable, even though on most hardware, it is typically not going to result in performance gains, yet at lower concurrency typically will also not lead to performance degradation.
- The CPU time and resources given to a MTQ worker thread are entirely handled by the Operating System.
- An MTQ worker thread will ONLY occupy an entire CPU core if the core is not busy otherwise.
- TM1 simply initiates a parallel thread. Where and how this thread is being processed is handled by the Operating System.

1.2.2 MTQ Configuration Scenarios, Considerations & Practices

a) Generally, the best practice is to set the MTQ value such that the maximum available processor cores are used, i.e. MTQ=All or MTQ=-1 or MTQ=M (with M= # CPU cores incl. hyper-threading cores).
   - ⇒ it is typically best to leave MTQ at its MTQ=ALL (default for PA) and hence have it leverage as many worker threads as there are CPU cores. This ensures that hardware is leveraged at its maximum.
   - ⇒ If you have two or more TM1 databases on the same machine, it is still recommended to set MTQ to ALL, because: An MTQ worker thread does not occupy a CPU core. MTQ worker threads will share CPU cores where needed and where applicable. It is the Operating System that handles CPU time and resources.
     - ⇒ If you have two databases on an 8 core machine, each set to MTQ=8, and on each a user runs a query leveraging 8 MTQ threads, the 16 MTQ threads are balanced among the 8 cores on the machine, utilizing the HW in an optimal way.
     - ⇒ If only one user runs a query leveraging 8 MTQ threads, the HW is still used in an optimal way.
     - ⇒ But if you set MTQ=4 and just one user runs a query, the HW is utilized at only 50%.
     - ⇒ If we extend these simple examples to many users and dozens to hundreds of threads (like in a typical, large TM1 environment), we can argue that every TM1 database should be configured to run queries as efficiently as possible, leveraging the available HW as much as possible.
     - ⇒ Balancing HW utilization is the job of the Operating System

   Caveat: While MTQ=ALL ensures that hardware is leveraged at its maximum, it is only a recommended setting provided that the HW-sizing is adequately matched with the demands of the TM1 Database(s) that are running on the machine. Operating TM1 databases with MTQ=ALL yet on machines with insufficient HW capabilities may over-tax the HW and lead to performance degradation.

b) On very powerful HW (high # of CPU cores) on smaller to mid-sized databases, it may not make much of a difference to set MTQ to a value that is lower than the total number of cores. I.e. on a 36

core machine, an MTQ=32 or MTQ=36 may not make not much of a difference if the data volume is not sufficiently high to allow performance gains via additional processing threads.

c) On very large Databases, a high MTQ # typically does make a big difference
   ⇒ It is a good practice to start with MTQ=ALL (= default).
   ⇒ To evaluate if different MTQ settings improve performance, adopt a holistic testing approach:
      - ***MTQ Worker Thread ≠ CPU Core*** => An MTQ Worker Thread is 'just' a TM1 Thread.
      - What is the peak load on each database?
      - What is the Avg CPU utilization at Peak?
      - Do we currently max-out our HW capabilities? (HW should be used as much as possible; HW idle time is not good, because it means 'things' could be running faster)
      - How do the # of cores affect our end-user queries? This depends on the database/cubes: are they very large, do they leverage rules, …

      ⇒ Analyze!

      ⇒ Is there an MTQ setting where using additional worker threads do not matter much anymore? If Yes, then this should be your max MTQ setting (i.e. do not go higher than where you see a tangible difference)

d) Some additional considerations when evaluating MTQ Settings:
   - One could argue that setting MTQ to a slightly lower value than the # of cores will allow more CPU time to be available for TI-processing (for example). **But:** This is **only** true IF the total concurrency on the database(s) is sufficiently low (compared to the # of cores). Example: The Operating System is balancing 200 concurrent TM1 users (and their TM1 threads) among its 36 cores. In such a context, setting MTQ to 36 or 34 or 30 will not really make a difference, because the CPU will be operating at maximum capacity regardless of the MTQ setting.

   ⇒ When considering to lower MTQ, take into account the overall concurrency on the database at peak time and the data volume that you are querying.

   ⇒ If concurrency is high and the data volume high, a higher MTQ value typically results in better performance in that end-user queries will be faster.

   ⇒ When you operate multiple Databases on one server and each database has a high concurrency, lowering MTQ will not necessarily free up CPU time for processing etc., because the overall load on the environment is already high.

   ⇒ Lowering MTQ in environments where multiple TM1 Database Instances share the same hardware will only have a positive impact on performance if the concurrency on each database is sufficiently low such that HW resources are kept available for other TM1 databases (and if those other databases need the free CPU time.

   ⇒ When setting MTQ, do not optimize it for low concurrency times. Optimize it for when the database is used most.

e) By employing a 'give and take' approach to assigning and re-assigning processing units, TM1 will balance and re-balance MTQ thread pool utilization between query requests: if all MTQ pool threads are available and a particular query can leverage all threads, TM1 may assign the max # of worker threads available and hence start MTQ processing with all threads. If additional queries are run during that time, hence 'requesting' query time, and once one or more MTQ worker threads have finished (committed), TM1 will assign the now free threads to other queries.

f) Keep in mind that you can change MTQ settings while the server is running and while users are running queries. The MTQ cores will be re-balanced according to your new MTQ configuration.

g) The use of MTQ will increase memory (RAM) usage. Subsequently, the use of MTQ typically will require an increase in VMM size. If VMM cache is set too low, even queries that were cached without MTQ use may not be cached anymore once MTQ is enabled. In such cases – to avoid unnecessary re-execution of MTQs - increase the VMM value until repeated query execution will not trigger MTQ activity anymore (indicating the cache is used). See <Caching of MTQ results> below.

1.2.3 Advanced MTQ configuration

- Per IBM support flash, Customers who are applying MTQ to models with reasonably complex rules (i.e. to rules that have cross-cube references with recursiveness depth more than 2) who are on an early TM1 10.2 release (prior to 10.2 FP1) could intermittently suffer from incorrect calculations unless they have disabled MTQ for single cell consolidation via parameter MTQ.SingleCellConsolidation=false. This parameter should only be used in such Pre 10.2 FP1 environments, once an upgrade to the fixpack or higher version has occurred, the parameter should be removed.

- The dynamic TM1s.cfg parameter MTQ.MultithreadStargateCreationUsesMerge=TRUE can speed up the creation of large stargate views of >100MB. The default value of this parameter in TM1 10.2.2 is FALSE.

- The dynamic TM1s.cfg parameter MTQ.CTreeWorkUnitMerge=TRUE speeds up concurrent population of calculation cache kept in the CTree, thereby improving performance of queries that cause a very large cache (re-)population. The default value of this parameter in TM1 10.2.2 is FALSE. Note that MTQ.CTreeWorkUnitMerge=TRUE could lead to redundant work in MTQ threads when the same calculation for exactly the same cell is re-computed per MTQ thread that needs it, whereas with MTQ.CTreeWorkUnitMerge=FALSE a computed cell would be published faster into a global CTree cache and then re-used by all MTQ threads.

- In TM1 10.2 Version prior to TM1 10.2.2. FP2 HF9, the (default) MTQ.CTreeWorkUnitMerge=FALSE setting could lead to rules not being calculated in certain scenarios where TI-processes were used to (re-)populate/refresh data. We therefore recommend to upgrade to TM1 10.2.2 FP2 HF9 or higher (10.2.2 FP3 and 10.3) when using MTQ. If an upgrade is not feasible at the given time and issues with rule calculations are encountered after TI-processing, enabling MTQ.CTreeWorkUnitMerge (MTQ.CTreeWorkUnitMerge=TRUE) will also solve the issue.

- The dynamic TM1s.cfg parameter MTQ.OperationProgressCheckSkipLoopSize=<N> specifies the number of cells to be processed before checking whether multi-threaded splits are needed. The default value is 10000. Is overridden by MTQ.ImmediateCheckForSplit=T which will effectively set MTQ.OperationProgressCheckSkipLoopSize to 1.

1.3 MTQ Logging and Monitoring

1.3.1 MTQ Monitoring

MTQ activity is best monitored via use of the TM1 Operations Console. The parent thread and each MTQ worker thread will all be visible as separate threads in TM1 Operations Console.

In the following screenshot, thread ID 14744 ran a query (a view) that subsequently spawned 8 worker unit threads:

ID	User	Context	State	Function	Type	Object	Info	Time (s)
6840	Th:Pseudo	-	Idle	-	-	-	-	0
6112	Th:Dynamic Config	-	Idle	-	-	-	-	0
7484	Admin	PM Hub	Idle	-	-	-	-	0
14744	Admin	Architect	Run:R	GetViewByHandle	Rule	FX	-	57
13260	> Work unit for 14744	-	-	-	-	-	0	
3136	> Work unit for 14744	-	-	-	-	-	0	
13656	> Work unit for 14744	-	-	-	-	-	0	
13748	> Work unit for 14744	-	-	-	-	-	0	
13632	> Work unit for 14744	-	-	-	-	-	0	
12816	> Work unit for 14744	-	-	-	-	-	0	

1.3.2 MTQ Logging

To generate logging information on multi-threaded queries, the following entries can be made in the tm1s-log.properties file (located in the same location as your tm1s.cfg file):

To generate logging information on multi-threaded queries, the following entries can be made in the tm1s-log.properties file (located in the same location as your tm1s.cfg file):

- To capture Stargate creation times: log4j.logger.TM1.Cube.Stargate=DEBUG

  => look for tm1server.log entries like

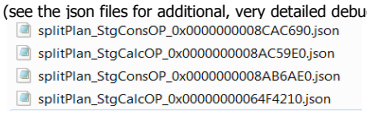
```
10148  [3]  DEBUG  2015-07-27 20:42:05.666  TM1.Cube.Stargate.Reference  Stargate 0x00000000E6967210 referenced : RefCount=1->2
10148  [3]  DEBUG  2015-07-27 20:42:05.666  TM1.Cube.Stargate.Reference  Stargate 0x00000000E6967210 released : RefCount=2->1
10148  [3]  DEBUG  2015-07-27 20:42:05.689  TM1.Cube.Stargate.Catalog  Adding stargate: 0x00000000E6969410 to catalog: 0x00000000E004BF58 of cube:
0x0000000006993010.
10148  [3]  DEBUG  2015-07-27 20:42:05.689  TM1.Cube.Stargate.Reference  Stargate 0x00000000E6969410 referenced : RefCount=0->1
10148  [3]  DEBUG  2015-07-27 20:42:05.689  TM1.Cube.Stargate.Reference  Stargate 0x00000000E6969410 referenced : RefCount=1->2
10148  [3]  DEBUG  2015-07-27 20:42:29.298  TM1.Cube.Stargate  dbstgCreate: dbstgCalculate (0x00000000E6969410) in 23609ms SGSize=24701Kb
workPoolSize=0Kb hPoolSize=128Kb.
10148  [3]  DEBUG  2015-07-27 20:42:29.299  TM1.Cube.Stargate.Definition  Axes Consolidation Subsets:
   Dimension: <Dimension Elements as on columns and rows>
10148  [3]  DEBUG  2015-07-27 20:42:30.059  TM1.Cube.Stargate  dbstgCreate: dbstgCalculateConsolidationLevel_UseTree (0x00000000E6969410) in 760ms
SGSize=24701Kb workPoolSize=1055Kb hPoolSize=128Kb.
10148  [3]  DEBUG  2015-07-27 20:42:30.063  TM1.Cube.Stargate  dbstgCreate: dbstgStoreConsolidationsBack_UseTree (0x00000000E6969410) in 4ms
SGSize=24701Kb workPoolSize=1055Kb hPoolSize=128Kb.
10148  [3]  DEBUG  2015-07-27 20:42:30.063  TM1.Cube.Stargate  dbstgCreate: CalculateAxesConsolidations Stargate (0x00000000E6969410) in 765ms
10148  [3]  DEBUG  2015-07-27 20:42:30.063  TM1.Cube.Stargate  New Stargate Created (0x00000000E6969410) in 24374ms SGSize=24701Kb workPoolSize=0Kb
hPoolSize=128Kb.
10148  [3]  DEBUG  2015-07-27 20:42:30.063  TM1.Cube.Stargate.Reference  Stargate 0x00000000E6969410 released : RefCount=2->1
10148  [3]  DEBUG  2015-07-27 20:42:30.357  TM1.Cube.Stargate.Reference  Stargate 0x00000000E6967210 released : RefCount=1->0
10148  [3]  DEBUG  2015-07-27 20:42:30.357  TM1.Cube.Stargate  Destroying Stargate 0x00000000E6967210
```

- To capture work unit splitting: log4j.logger.TM1.Parallel=DEBUG

  => look for tm1server.log entries like

```
10148  [3]  DEBUG  2015-07-27 20:30:52.187  TM1.Parallel.Splits ...
10148  [3]  DEBUG  2015-07-27 20:35:20.175  TM1.Parallel.SplitPlan  Wrote split plan for StgConsOP[<Cube>:( [<DimensionA>].[<DimensionAElement>],
[<DimensionB>].[<DimensionBElement>],...)] to file '<LogFileDirectory>\<jsonfilename>.json'
10148  [3]  DEBUG  2015-07-27 20:35:20.178  TM1.Parallel.Operation  ...OP:... TM1ParallelOperation::MergeWorkUnits in 3ms
```

(see the json files for additional, very detailed debugging information on the split plans, for example:



- To capture the event of operation threads picking work units: log4j.logger.TM1.OperationThread=DEBUG

  => look for tm1server.log entries like

```
13064   []   DEBUG   2015-07-27 21:08:53.279   TM1.OperationThread   Executing work unit:StgBuildWU:0x0000000008C318D0 [1/(52):(80)]OP:0x0000000006626CD0
3360   []   DEBUG   2015-07-27 21:08:53.279   TM1.OperationThread   Executing work unit:StgBuildWU:0x0000000008C2F9C0 [1/(17):(36)]OP:0x0000000006626CD0
14636   []   DEBUG   2015-07-27 21:08:53.279   TM1.OperationThread   Executing work unit:StgBuildWU:0x0000000008C356F0 [1/(37):(51)]OP:0x0000000006626CD0
13064   []   DEBUG   2015-07-27 21:09:32.580   TM1.OperationThread   Finished execution of work unit:StgBuildWU:0x0000000008C318D0 [1/(52):(80)]OP:0x0000000006626CD0
13064   []   DEBUG   2015-07-27 21:09:32.601   TM1.OperationThread   Executing work unit:StgBuildWU:0x0000000008C318D0 [1/(33):(36)]OP:0x0000000006626CD0
14636   []   DEBUG   2015-07-27 21:11:46.369   TM1.OperationThread   Finished execution of work unit:StgBuildWU:0x0000000008C356F0 [1/(37):(51)]OP:0x0000000006626CD0
14636   []   DEBUG   2015-07-27 21:11:46.373   TM1.OperationThread   Executing work unit:StgBuildWU:0x0000000008C356F0 [1/(14):(16)]OP:0x0000000006626CD0
```

Note that those logging flags typically are only useful for advanced debugging & troubleshooting. We recommend to only enable MTQ logging temporarily when/if needed or if prompted by IBM technical support.

1.4 Caching of MTQ results

Query caching behaviour is configured per cube via the VMM[1] value in the }CubeProperties cube, where the VMM value defines the maximum amount of memory to be used for caching per cube (i.e. the memory pool in RAM that is made available for caching). In many cases it is therefore a good practice to optimize/increase memory reserved for caching Stargate views by increasing the VMM value in the }CubeProperties Cube to a significantly higher value than the default of 128kb.

The use of MTQ will increase memory (RAM) usage. Subsequently, the use of MTQ typically will require an increase in VMM size. If VMM cache is set too low, even queries that were cached without MTQ use may not be cached anymore once MTQ is enabled. In such cases – to avoid unnecessary re-execution of MTQs - increase the VMM value until repeated query execution will not trigger MTQ activity anymore (indicating the cache is used).[2]

For very large environments/cubes (in the double digit GB range or with hundreds of GB), an increase of the VMM cache to many MBs or even a few GBs may be useful, particularly for environments that are not highly volatile. For larger environments & where MTQ is leveraged heavily & with high concurrency, reducing the VMT threshold may be useful to reduce unnecessary (high) CPU utilization: With MTQ, a query that may a minute to run using one logical processor may run within let's say 3 seconds when using 24 cores. Particularly in high concurrency environments, such a query should be cached to not re-engage the 24 cores again (even if just for 3 seconds), hence freeing up computing resources for new queries or longer/larger queries

[1] TM1 setting thresholds and maximum cache memory for Stargate views. A Stargate view is a calculated and stored subsection of a TM1 cube that TM1 creates when you browse a cube with the Cube Viewer, Web-Sheet or In-Spreadsheet Browser. The purpose of a Stargate view is to allow quicker access to the cube data. A Stargate view is different from a TM1 view object. The Stargate view contains only the data for a defined section of a cube, and does not contain the formatting information and browser settings that are in a view object. A Stargate view that TM1 creates when you access a cube contains only the data defined by the current title elements and row and column subsets. TM1 stores a Stargate view when you access a view that takes longer to retrieve than the threshold defined by the VMT property in the control cube }CubeProperties. A Stargate view persists in memory only as long as the browser-view from which it originates remains unchanged. When you recalculate the browser view, TM1 creates a new Stargate view based on the recalculated view and replaces the existing Stargate view in memory. When you close the browser view, TM1 removes the Stargate view from memory. Stargate View caching behavior/thresholds (by cube) can be configured via changing the VMT and VMM values in the }CubeProperties cube: For each cube, the VMM property determines the amount of RAM reserved on the server for the storage of Stargate views. The more memory made available for Stargate views, the better performance will be. You must, however, make sure sufficient memory is available for the TM1 server to load all cubes. If no VMM value is specified the default value is 128KB. T The valid range is If no VMM value is specified the default value is 128KB. The valid range is 16 - 42934943296 kb (16-2^32 kb).

[2] For larger cubes, increasing the MTQ value to a value of between 1MB (1000000) and 5MB (5000000) is often a good start.

1.5 Use of MTQ for TI-processing

For TM1 10.2.2 <FP6 the following workaround can be used to leverage MTQ:

a) Build the view in a separate process & use the ViewConstruct() function to 'query'/'construct' the view. This will trigger MTQ.

b) Then, in a separate process, you may leverage the cached stargate view for processing. Note that this separate process may also build/rebuild this view and could do so using a different view name. What matters is that the views (the cached one from (a) and the new one from (b)) have the same content, i.e. that the corresponding stargate is the same.

Notes on this pre- FP6 workaround:

- The ViewConstruct and the data-processing TI-operations need to be separate. Using a trigger/master process to call two TI's will not work.
- You may use a chore to trigger both processes, but you have to use multi-commit mode
- The result of the ViewConstruct will need to go into the TM1 cache in order for the 2nd processes to leverage it => For large views, a significant increase in the VMM value may be required
- Only the visible part of the view will be cached when using ViewConstruct. It follows that if you are using views in TI that do not have all dimensions assigned as rows (ViewRowDimensionSet), the view needs to be re-build accordingly for leveraging MTQ
- Some views may be too large to warrant caching via ViewConstruct (VMM too high or not enough memory).
- Processing Leaf-level views may not always provide a performance gain (unless the leaf levels are driven by rule-based consolidations for example)

As of TM1 10.2.2 FP6, the above workaround (via ViewConstruct) is no longer required in order to employ MTQ for TI-processing

(see underlying hyperlink for more information on conditions & limitations).

- As of FP6, MTQ will be triggered in the context of the TI data tab itself.
- For individual TI processes, this auto-MTQ behaviour can be disabled by placing the function DisableMTQViewConstruct( ) in the prolog of the TI.
- Performance with this approach is slightly faster than if employing the pre FP6 workaround.
- MTQ for TI-views is only used for C-level views (Leaf-level views will not leverage MTQ unless a leaf value is derived per rule pointing to a consolidation).
- If the stargate was cached previously, MTQ will not be leveraged. In other words: The FP6 enhancement is cache-aware.
- The new TM1s.cfg parameter MTQQUERY can be used to enable/disable the use of MTQ for Views in TI processing
- The Planning Analytics Default is MTQQUERY=T
- MTQQuery will use MTQ to query and 'pre-'cache C-level views used by TI-processes. While this typically leads to performance improvements, it **can also significantly increase RAM consumption for larger views**. That is because TI with MTQQuewry=F / without MTQQuery will not load a view into RAM, but instead 'cycle' through a view in smaller blocks, not really

increasing RAM usage much. Yet with MTQQuery=T, TM1 will attempt to query and load the entire C-level view into RAM prior to processing it via TI

⇒ MTQQuery=T (default) can increase RAM use significantly
⇒ It is recommended to only use IF (i) size of views is managed (known to be small enough) and (ii) sufficient RAM is available, otherwise, use MTQQuery=F