Parallel Data Processing with IBM Planning Analytics:

Methods, design options,
and a plug&play parallel processing utility

V3.0

Last updated:
February 2020

By:
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

Document Version History

Date	Version	Author	Description
06/12/2015	1.0	Sameer Gujar & Andreas Kugelmeier	Version 1.0
8/18/2015	1.1	Andreas Kugelmeier	Add trigger file functionality to allow use of the template via RunTI or via Command Center or similar (using the trigger file functionality, double execution of thread is avoided).
11/19/2015	1.2	Andreas Kugelmeier	Add references & copy on RunTI.exe, Cognos Command Center, TI Scheduling Utility
4/29/2016	1.21	Andreas Kugelmeier	Fix links
11/9/2016	1.22	Andreas Kugelmeier	Minor updates to section 2
7/2018	2.0	Andreas Kugelmeier	Major update to V2 of the parallel processing utility
11/19/2019	2.1	Andreas Kugelmeier	Misc. updates
January 2020	3.0	Andreas Kugelmeier	Alternate use of RunTI or RunProcess
February 2020	3.1	Andreas Kugelmeier	Fix typos

Table of Contents

1. Introduction to Parallel data processing

In a parallel data-processing regime, data is loaded or processed into one or multiple cubes via separate & parallel processing threads. Using a parallel data processing framework, a cube can be loaded using as many CPU threads as available and applicable, hence providing significant performance gains of data-load operations. Due to the very fast general load speeds of TM1 Turbo integrator (TI) processes, Parallel data processing can be particularly useful when updating in scenarios where very large cubes are to be updated at a high frequency (for example, billions of facts to be updated daily).

For very large cubes, we recommend applying a TI framework methodology that will allow loading one or more cubes using parallel data loads, allowing the reload of cubes (with adjusted historical data for example) using parallel load threads and hence significantly speeding up load time. For example, if the load of one month of data takes 10 minutes and one wants to load/update data for an entire year, one can leverage parallel data processing to load 12 data sets (one for each month) in parallel, hence still resulting in a total load time of only 10 minutes instead of 120 minutes.

Using a parallel data processing regime/framework, TM1 can load upwards of 50.000 records per second per CPU core. In a 16 CPU core context, this can mean an overall data-load/update speed of roughly 800.000 records per second or 1 Billion records in 21 minutes!

2. Parallel Data Load: Restrictions & Design Options

Methods to initiate and perform a parallel data load include:

a) Launching multiple TM1 Architect/Perspectives or Performance Modeler sessions manually and starting a data load process in each session,

b) Using TM1RunTI or RunProcess to trigger multiple data load processes built to allow parallel execution and data load,

c) Setting up chores (one for each thread) that will automatically trigger a data load process based on an execution flag/indicator.

d) Leveraging the scheduling & orchestration capabilities of IBM Cognos Command Center in combination with TM1,

e) Using Python or equivalent, and TM1 Rest API calls (Note: packages like TM1py make using Python with the TM1 Ret API easy. TM1py is a Python package that wraps the TM1 REST API in a simple to use library)

All the above methods require that applicable TI process(es) or other procedures (Python code) used to load the data (the processes that are initiated in parallel) will adhere to the following guidelines, conditions & restrictions:

- When performing a parallel data load, the zero-out / clear data operation should also be parallelized where applicable. Note that zero-out operations against large data volumes are also performance intensive. If a complete refresh/load is required, the cube should be cleared using the CubeClearData() Command (CubeClearData() is significantly faster than ZeroOutView() in clearing ALL data from a cube).

- The load processes may not update meta- & master-data that is in a dependency-relationship to the cube(s) that are updated. Any data-update-related master- & metadata updates shall occur
  - in a single thread per dimension and
  - prior to initiating the parallel fact data load.

  In other words: master-data for one dimension can at this point not be updated in parallel. A dimension update will lead to a lock of the dimension and through the lock will prevent a parallel data load/update from occurring. Dimension updates are hence to occur prior to the fact data update.

- Any subsets and views that are created in the context of the data-load TI processes need to be unique subsets and views (unique within the TM1 instance) to avoid contention during subset and view creation and deletion.

- Cube-Dependencies need to be (re)established where applicable prior to parallel data load. Please refer to Understanding Cube Dependency for more information on identifying and handling dependency locking.

- Note that TM1 and Planning Analytics will create cube read-locks or write-locks after dimension edits. Such locks will have to be 'removed' prior to initiating parallel processing:

  Avoiding Locks: Handling of Relationship between Dimension Updates and Cube Locking

  When a dimension is edited, all cubes that use this dimension and that leverage cube rules need to be locked (a short lock) to re-evaluate rule validity and consistency. By default, this resulting lock is an IX lock (just as with TM1 10.2.2) but the behavior in PA can be changed for ReduceCubeLockingOnDimensionUpdate=T, such that first an RO lock is issues and then evaluated for necessity of an IX lock).
  But while ReduceCubeLockingOnDimensionUpdate will reduce cube locking, it does not eliminate locking: After a dimension has been edited, processing or reading/writing against a cube with that same dimension will - the first time the cube is accessed after a dimension update - lead to a lock and prevent parallel processing (all threads except one in WAIT mode) and/or end-user write-back to this cube during processing (users locked during process commit) and/or prevent reading the cube (in case the RO lock was converted to an IX lock).
  A proven procedure to eliminate contention due to RO or even IX locks after dimension updates is as follows: Implement a functionality that will create (and destroy) applicable target cube views right after dimension update processing (after the dimension update was committed, i.e. NOT in the same TI process chain, but in a separate thread following the dimension updates). This procedure will establish and immediately release possible post-dimension processing locks. As a result, subsequent cube queries, commits and write-backs can all occur in parallel, hence significantly improving user experience and performance.

  Example Scenario: Two large cubes are to be updated. Affected dimensions: Customer & Product =>
  - Update Meta-& Master-data for (a) customer dimension & (b) product dimension
    via two parallel processes.
  - Update both cubes via N-parallel processes & with each process updating a bucket of data & running in parallel.

3. A Plug&Play Utility for Parallel data processing using TI processes and Chores

3.1 Introduction

The objects referenced in this utility may be downloaded at the following location:

Parallel Processing Utility

Parallel Processing Utility with Examples

Note: compared to V1 of this paper which used chores in the sample to simplify the illustration and use case of parallel data load, the sample utility presented in this paper uses a proven practice for launching and managing parallel Turbo-Integrator processing.

3.2 Components

SYS_IBM_Control.cub: Initial configuration of RunTI environment (configure where RunTI executable is found, how RunTI is to authenticate & initiate, how and where the thread governor TI is to manage and govern thread execution).

SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration.cub: Configuration of parallel processing jobs, by thread, including one pre-parallel thread and one post-parallel threads.

SYS_IBM_TI_-_Process_Parallel_Job.pro: process to run a parallel processing job configured in SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration.cub

SYS_IBM_TI_-_Create_Batch_File.pro: Creates a batch file for TI-process launch as per RunTI (.bat or .sh or other as per SYS_IBM_Contorl.cub entry). Is a sub-process to SYS_IBM_TI_-_Process_Parallel_Job.pro

SYS_IBM_TI_-_Launch_Batch_File.pro: Launches a batch file if 'Run Scripts Directly' in SYS_IBM_Control.cub is not set to Y. Is a sub-process to SYS_IBM_TI_-_Process_Parallel_Job.pro

3.3 Configuration

3.3.1 Initial configuration of the RunTI environment: SYS_IBM_Control.cub

'SYS_IBM_Control' is used for 'global' settings/configurations:

- **Process Log Directory Path**: The ProcessLog Directory is used as the location for all Debug Logs, Trigger Files, & Batch files for Parallel Processing. **Before running parallel jobs, please ensure that the 'Process Log Directory Path' in cube 'SYS_IBM_Control' points to a valid directory on your hard drive. For Linux: Please ensure the Process Log Directory Path uses only lower case characters.**
- **SystemObjectsPrefix**: Prefix used for Subset and View Names generated in the context of View and Subset Create processing and for passing subset and view names to the parallel processing threads.
- **Use SubsetMDXSet** (setting applies to the included processes SYS_IBM_View_Create.pro & SYS_IBM_Subset_Create.pro. The processes are used in the example, but are NOT needed for the parallel processing utility per se): Set to Y to use the newer TM1 Function SubsetMDXSet to convert MDX subsets to static subsets. This setting should be set to Y in general, and particularly for parallel

processing scenarios (to limit contention and locking due to MDX subset retrieval and conversion of MDX subsets to static subsets).

- **TemporaryVsPermanentSubsets_Default, TemporaryVsPermanentViews_Default** (setting applies to the included processes SYS_IBM_View_Create.pro & SYS_IBM_Subset_Create.pro. The processes are used in the example, but are NOT needed for the parallel processing utility per se): empty => permanent Subsets/Views, Permanent => Permanent Subsets/Views, Temporary => Temporary Subsets/Views. Use of 'Temporary' setting requires PA 2.0.5 (TM1 11.3.00000.27) or higher. If subsets set to 'Temporary', views should also be set to 'Temporary'
- **Misc**. parameters for Version Control, Time Period Management, etc.

- **Parallel Processing Method**: RunTI or RunProcess (if empty, then = RunTI)

- **'RunTI' Parameters in 'SYS_IBM_Control' are used for initial configuration of Processing via RunTI:**

  - **Enable Parallel Processing via RunTI**: Set to Y to enable Parallel Processing (for RunTI and/or RunProcess)
  - **RunTI Path**: specify the path where the TM1RunTI can be found. The directory also needs to contain all other files needed to allow TMRunTI to connect to TM1 (such as the API DLLs in sub-directories for example). If the RunTI path is specified, the RunTI batch file will execute a CD <RunTI Path' before launching TM1Runti in a second row of the batch file. If the RunTI path is not specified, it can also be specified together with the TM1RunTIExecutable parameter (see below),
  - **WAIT in Milliseconds between Checks for Parallel Thread Processing Completion**: For processing via process <Processing Logic: SYS_IBM_TI_-_Process_Parallel_Job.pro>; the time in milliseconds that the Parallel Thread Management Algorithm will wait in between checking if the launched parallel threads for parallel processing job have completed. It is recommended to set this interval to a few seconds duration at least to keep the thread manager from checking continuously. 10 seconds (10000) or more is a recommended default.
  - **WAIT in Milliseconds before Thread Launch**: For processing via process <Processing Logic: SYS_IBM_TI_-_Process_Parallel_Job.pro>; wait time in milliseconds in between the writing and the launching of the batch files with the TMRunTI command. Recommended Setting: Depending on Disk Access Speed, it can be good to specify a wait time of half a second to a second (500-100 milliseconds) to ensure the batch file is accessible in the file system before launch.
  - **WAIT in Milliseconds after completion of Pre- or Post-Parallel Threads**: For processing via process <Processing Logic: SYS_IBM_TI_-_Process_Parallel_Job.pro>; wait time in milliseconds after Pre-Parallel, Parallel, and Post-Parallel single threads have completed via RunTI and/or RunProcess. Recommended setting: 1000 milliseconds or higher. As of V3.0, this setting is also applied to Parallel Threads. The WAIT time is to be set such that after completion, there is sufficient time for the Cube Commit to occur prior to continuing with processing.
  - **TM1RunTI Start prefix**: prefix to start the TM1Run TI executable. Use: <Prefix><TM1RunTIExecutable> <Parameters>. Example: ./ for Linux, none for Windows
  - **TM1RunTI Batch File Extension**: Extension for TM1RunTI executable. Example: .bat for Windows, .sh for Linux
  - **TM1RunTI Executable**: TM1RunTI Executable. Example TM1RunTI.exe or (with path) C:\ibm\tm1_64\bin64\TM1RunTI.exe or in quotes if path contains spaces. If the path is included, the parameter 'RunTI Path' shall be left empty.
  - **Launch Script Directly:** Set to Y to not launch the RunTI sessions via .bat or .sh file, but directly via Command Line. Note that the *.bat and *.sh files will still be created as a means to log the script that was launched. If Launch Script directly is set to Y, the RunTI path needs to be

part of the TM1 RunTI Executable field (i.e. the path shall not be entered separately under 'RUNTI Path').

TM1 Run TI 'Connect' Parameters:

- **Adminhost** (optional): name or IP of the TM1 Admin host machine. If empty/not specified, then 'localhost' will be used implicitly.
- **TM1 Server** (server): the name of the TM1 DB instance
- **User**: user name. Note: Instead of using the 'user' and 'pwd' entries, a secured 'Configuration File' may be used along with an encrypted 'passwordfile' and 'passwordkeyfile', resulting in fully secured logon credentials for the RunTI process. Please refer to the TM1RunTI Documentation for further information. See 'password parameters' below for corresponding RunTI Parallel Configuration parameters.
- **AdminSvrSSLCertAuthority** (optional): The full path of the certificate authority file that issued the ICAS Admin Server's certificate.
- **AdminSvrSSLCertID** (optional) (API Default is : tm1adminserver). The name of the principal to whom the ICAS Admin Server's certificate is issued. The value of this parameter should be identical to the SSLCertificateID parameter in the Tm1admsrv.ini file.
- **AdminSvrSSLCertRevList** (optional): The full path of the certificate revocation file issued by the certificate authority that originally issued the ICAS Admin Server's certificate. A certificate revocation file will only exist in the event a certificate had been revoked.
- **ExportAdminSvrSSLCert** (optional) (T/F, Default = F): Specifies whether you want the certificate authority certificate which originally issued the ICAS Admin Server's certificate to be exported from the Microsoft Windows certificate store at runtime. When this option is selected, you must also set a value for AdminSvrSSLEx-portKeyID as described here. Refer to IBM Cognos TM1® Installation and Configuration Guide for appropriate TM1Server configuration.
- **AdminSvrSSLExportKeyId** (optional): The identity key used to export the certificate authority certificate, which originally issued the ICAS Admin Server's certificate, from the certificate store. This parameter is required only if you choose to use the certificate store by setting ExportAdminSvrSSLCert=T. Refer to IBM Cognos TM1 Installation and Configuration Guide for appropriate TM1Server configuration.
- **CAMNamespace** (optional): CAM namespace id (not the CAM namespace name) in case CAM authentication is used.

Password Parameters: Passwords are either provided in cleartext (not recommended) using the pwd parameter, or using an encrypted file provided by the passwordfile parameter.

- **Pwd**: password for user. Note: Instead of using the 'user' and 'pwd' entries, a secured 'Configuration File' may be used along with an encrypted 'passwordfile' and 'passwordkeyfile', resulting in fully secured logon credentials for the RunTI process. Please refer to the TM1RunTI Documentation for further information. See below for using such configuration and password files for parallel processing threads.
- **Passwordfile**: The full path of the file containing the encrypted password for the specified user. If no path is specified, the ICAS server directory is assumed. When this option is used, you cannot use -pwd.
- **Passwordkeyfile**: If passwordfile is set, the full path to the key file is also required in order to decrypt the password. The password file and key file can be created using TM1Crypt tool. Refer to IBM Cognos TM1 Installation and Configuration Guide.
- **Configuration File**: TM1 Run TI Configuration file.
- **SecurityMode**: Security Mode
- **Certversion**: Certificate Version

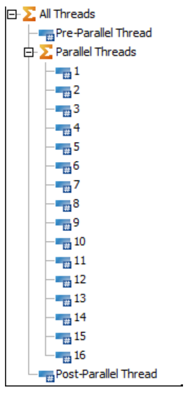
3.3.2 Parallel Processing Job Configuration: SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration.cub

Parallel Processing allows the configuration and execution of multi-threaded processing jobs consisting of:
a) a single process with parameters to be run prior to initiating parallel processing
b) parallel processing threads, involving one or more Ti processes & their parameters
c) a single process with parameters to be run after the completion of parallel processing.

Use Case Examples:
I) Parallel Update of a reporting cube with data from a large source cube
   a) zero-out or CubeClearData of a reporting target cube
   b) multiple parallel processing threads using the same TI process with different parameters, effectively splitting the source data into pieces that each are processed into the target cube in parallel.
   c) zero-out or CubeClearData of the source cube
II) Parallel Update of a reporting cube with data from different source cubes
   a) zero-out or CubeClearData of a reporting target cube
   b) multiple parallel processing threads using the same or different TI processes with different parameters, effectively splitting the source data into pieces that each are processed into the target cube in parallel.

Using the configuration cube 'SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration', single (pre- and post-parallel) and parallel threads can be configured per processing 'Job' and then processed accordingly. Processing of a job (using the Job ID or an Alias) will occur using process '**SYS_IBM_TI_-_Process_Parallel_Job.pro**', which will leverage the information in 'SYS_IBM_Control' to use RunTI to invoke serial and parallel processing threads, for a particular job & as per configuration in cube 'SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration':

- Dimension '**SYS_IBM_TI_-_Processing_Job_No**.dim': Processing Job ID (or Alias Name)
- Dimension '**SYS_IBM_TI_-_Parallel_Thread**.dim': Thread Types and Thread IDs
  - Pre-Parallel Thread: a single process with parameters to be run prior to initiating parallel processing
  - Parallel Threads 1-N: parallel processing threads, involving one or more Ti processes & their parameters
  - Post-Parallel Thread: a single process with parameters to be run after the completion of parallel processing.



- Dimension '**SYS_IBM_TI_-_Parallel_Thread_Configuration_Measure**.dim', with Elements:
  - **Counter**: Counter//Flag indicating a configured thread. Once the 'Process' value is populated, the thread is considered 'configured' and will be initiated.
  - **Parallel Thread Counter**: Counter which at runtime of 'SYS_IBM_TI_-_Process_Parallel_Job.pro' will indicate running (started) threads
  - **Parallel Thread Status**: status of a thread (status will be set back to 'Completed' by 'SYS_IBM_TI_-_Process_Parallel_Job.pro' once trigger file was found)
  - **Parallel Thread Completion Trigger File**: name of trigger file to look for
  - **Process**: name of the TI process to run
  - **Parameter A-AB**: name of parameters A-Z,AA,AB to call when running the TI process
  - **Parameter A-AB Value**: string value for parameters A-Z,AA,AB to call when running the TI process

Note that 'SYS_IBM_TI_-_Process_Parallel_Job.pro' may leverage and manage parallel processing for <u>any</u> TI process, for as long as the following conditions are met and parameters and TI-script added for processes that run in parallel (not needed for pre- and post-parallel threads):

- The TI process needs to allow for parallel processing in that it shall not include TI-processing actions that may cause contention/locking). If permanent views and subsets are created and used by a process that is run multiple times in parallel, the subset and view names should also be made unique per thread. For this purpose, the parallel process agent will pass the Thread Number (as a number string) to each process that is spawned. The Thread No should then be used as part of the subset and view names. It is therefore recommended to include the parameter ***pThread*** to each process that may be running in parallel mode, and to use the Thread number as part of the subset and view names.
- **String Parameter *pCompletionTriggerFile.*** The completion trigger file parameter value is determined by 'SYS_IBM_TI_-_Process_Parallel_Job.pro' at runtime and passed to the parallel process. The completion trigger file name is used by the process to write a 'completion flag' to the Process Logs directory, via:
- The following script at the **end of the TI-process epilog**:

```
IF ( pCompletionTriggerFile @<> '' );
  ASCIIOUTPUT ( pCompletionTriggerFile, 'thread has finished');
ENDIF;
```

3.4 Processing Logic: SYS_IBM_TI_-_Process_Parallel_Job.pro

1. Launch Pre-Parallel Thread and wait until finished. Once Trigger file can be found, continue processing. The trigger file is created by process 'SYS_IBM_TI_-_Process_Parallel_Job.pro' and passed to the target process via the pCompletionTriggerFile parameter. 'SYS_IBM_TI_-_Process_Parallel_Job.pro' will wait until the thread has completed, and then flag thread as 'completed'. The completion trigger file and - upon completion of the thread - the flag 'completed' are written to 'SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration'.

2. Launch ALL Parallel Threads. Parallel Threads need to be configured starting with Thread 1, without gaps (i.e. a 5-Thread processing configuration needs Threads 1-5 to be populated, and not Threads 1-4, and then Thread 6). Process 'SYS_IBM_TI_-_Process_Parallel_Job.pro' assigns a unique trigger file to each thread. The trigger file is passed to the target process via the pCompletionTriggerFile parameter. The completion trigger file is also written to 'SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration'.

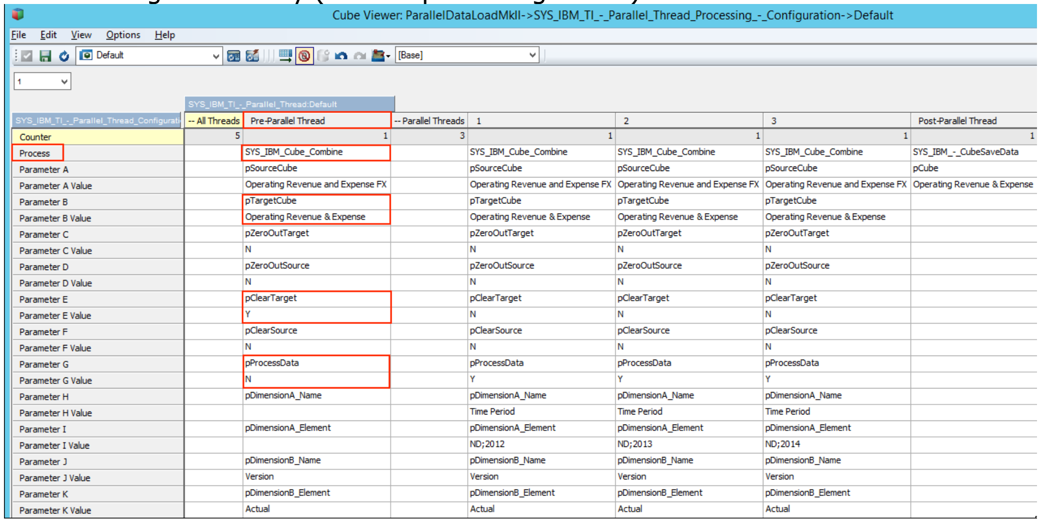
3. After process 'SYS_IBM_TI_-_Process_Parallel_Job.pro' has launched all parallel threads, it will enter a WAIT loop and keep checking if the parallel threads have completed (= Trigger file exists in ProcessLog directory). Once 'SYS_IBM_TI_-_Process_Parallel_Job.pro' finds the completion trigger file of a processing thread, the thread is flagged as 'completed'. For each completed thread, the flag 'completed' is written to 'SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration'. Once all threads have completed (all trigger files exist), processing of the job will continue:

4. Launch Post-Parallel Thread and wait until finished. 'SYS_IBM_TI_-_Process_Parallel_Job.pro' will wait until the thread has completed, and then flag thread as 'completed'. The completion trigger file and - upon completion of the thread - the flag 'completed' are written to 'SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration'.

3.5 Example Configuration

Job 1 in cube 'SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration' is configured for parallel processing of data from 'Operating Revenue and Expense FX' to 'Operating Revenue and Expense', using one generic process for parallel processing.
Note that the utility is process-agnostic (see above for instructions on how to use any process for parallel processing) and that any number of (different) TI processes can be run in parallel. Our example uses one generic process only for illustration purposes (and to make working with the example and adjusting the example a little easier). Also, please note that the generic process imposes a runtime overhead. When processing very large data volumes, a tailored process is typically better suited for achieving optimal runtimes. Our example configuration uses

i. One pre-parallel thread where the parameters of process 'SYS_IBM_Cube_Combine' are configured to clear the target cube only (no data processing occurs):


Cube Viewer: ParallelDataLoadMkII->SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration->Default

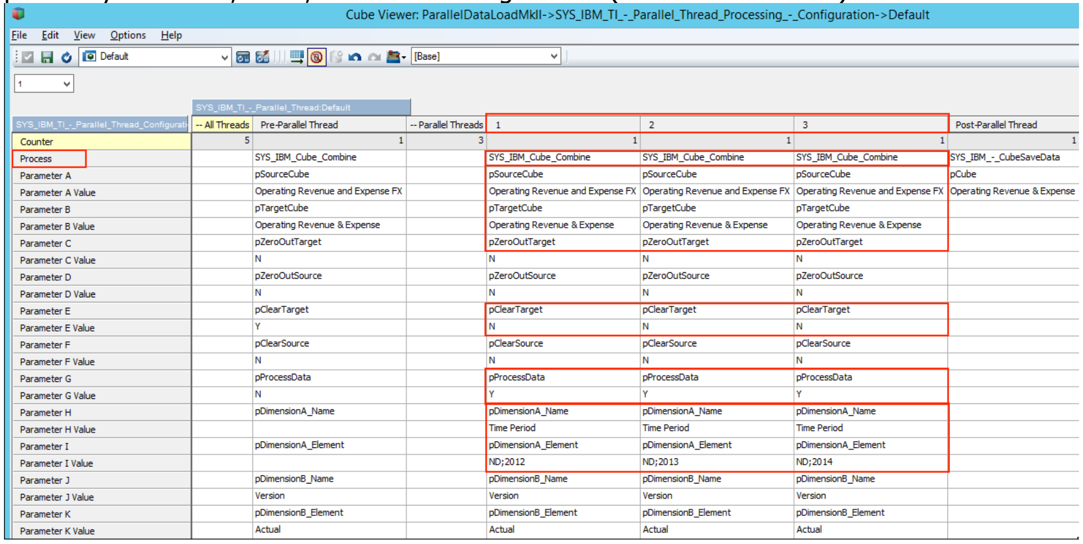
File Edit View Options Help

Default [Base] 1

SYS_IBM_TI_-_Parallel_Thread:Default

SYS_IBM_TI_-_Parallel_Thread_Configurati	-- All Threads	Pre-Parallel Thread	-- Parallel Threads	1	2	3	Post-Parallel Thread
Counter	5	1	3	1	1	1	1
Process		SYS_IBM_Cube_Combine		SYS_IBM_Cube_Combine	SYS_IBM_Cube_Combine	SYS_IBM_Cube_Combine	SYS_IBM_-_CubeSaveData
Parameter A		pSourceCube		pSourceCube	pSourceCube	pSourceCube	pCube
Parameter A Value		Operating Revenue and Expense FX		Operating Revenue and Expense FX	Operating Revenue and Expense FX	Operating Revenue and Expense FX	Operating Revenue & Expense
Parameter B		pTargetCube		pTargetCube	pTargetCube	pTargetCube	
Parameter B Value		Operating Revenue & Expense		Operating Revenue & Expense	Operating Revenue & Expense	Operating Revenue & Expense	
Parameter C		pZeroOutTarget		pZeroOutTarget	pZeroOutTarget	pZeroOutTarget	
Parameter C Value		N		N	N	N	
Parameter D		pZeroOutSource		pZeroOutSource	pZeroOutSource	pZeroOutSource	
Parameter D Value		N		N	N	N	
Parameter E		pClearTarget		pClearTarget	pClearTarget	pClearTarget	
Parameter E Value		Y		N	N	N	
Parameter F		pClearSource		pClearSource	pClearSource	pClearSource	
Parameter F Value		N		N	N	N	
Parameter G		pProcessData		pProcessData	pProcessData	pProcessData	
Parameter G Value		N		Y	Y	Y	
Parameter H		pDimensionA_Name		pDimensionA_Name	pDimensionA_Name	pDimensionA_Name	
Parameter H Value				Time Period	Time Period	Time Period	
Parameter I		pDimensionA_Element		pDimensionA_Element	pDimensionA_Element	pDimensionA_Element	
Parameter I Value				ND;2012	ND;2013	ND;2014	
Parameter J		pDimensionB_Name		pDimensionB_Name	pDimensionB_Name	pDimensionB_Name	
Parameter J Value		Version		Version	Version	Version	
Parameter K		pDimensionB_Element		pDimensionB_Element	pDimensionB_Element	pDimensionB_Element	
Parameter K Value		Actual		Actual	Actual	Actual	

ii. Three parallel threads where the parameters of process 'SYS_IBM_Cube_Combine' are configured to process years 2013, 2014, 2015 to the target cube (no zero-out occurs):


Cube Viewer: ParallelDataLoadMkII->SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration->Default

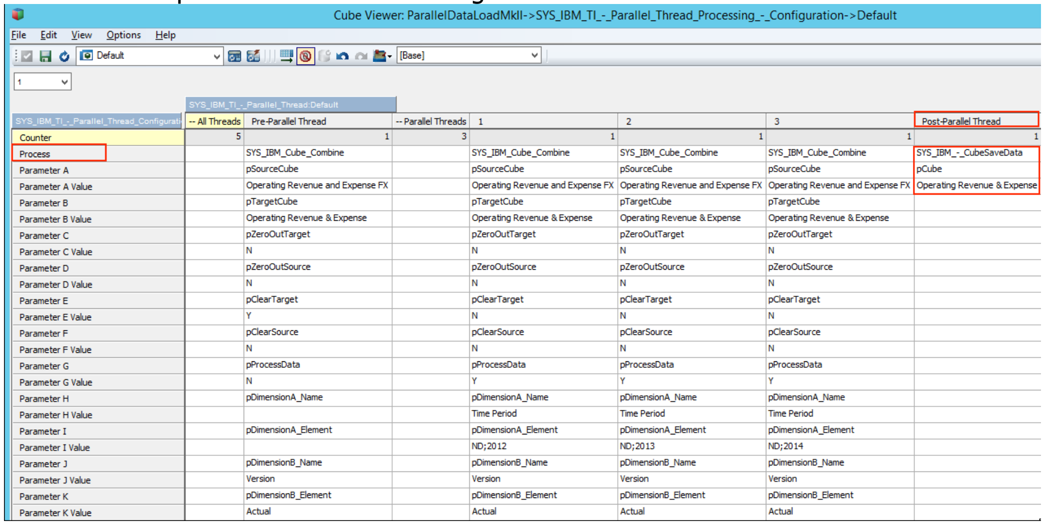
File Edit View Options Help

Default [Base] 1

SYS_IBM_TI_-_Parallel_Thread:Default

SYS_IBM_TI_-_Parallel_Thread_Configurati	-- All Threads	Pre-Parallel Thread	-- Parallel Threads	1	2	3	Post-Parallel Thread
Counter	5	1	3	1	1	1	1
Process		SYS_IBM_Cube_Combine		SYS_IBM_Cube_Combine	SYS_IBM_Cube_Combine	SYS_IBM_Cube_Combine	SYS_IBM_-_CubeSaveData
Parameter A		pSourceCube		pSourceCube	pSourceCube	pSourceCube	pCube
Parameter A Value		Operating Revenue and Expense FX		Operating Revenue and Expense FX	Operating Revenue and Expense FX	Operating Revenue and Expense FX	Operating Revenue & Expense
Parameter B		pTargetCube		pTargetCube	pTargetCube	pTargetCube	
Parameter B Value		Operating Revenue & Expense		Operating Revenue & Expense	Operating Revenue & Expense	Operating Revenue & Expense	
Parameter C		pZeroOutTarget		pZeroOutTarget	pZeroOutTarget	pZeroOutTarget	
Parameter C Value		N		N	N	N	
Parameter D		pZeroOutSource		pZeroOutSource	pZeroOutSource	pZeroOutSource	
Parameter D Value		N		N	N	N	
Parameter E		pClearTarget		pClearTarget	pClearTarget	pClearTarget	
Parameter E Value		Y		N	N	N	
Parameter F		pClearSource		pClearSource	pClearSource	pClearSource	
Parameter F Value		N		N	N	N	
Parameter G		pProcessData		pProcessData	pProcessData	pProcessData	
Parameter G Value		N		Y	Y	Y	
Parameter H		pDimensionA_Name		pDimensionA_Name	pDimensionA_Name	pDimensionA_Name	
Parameter H Value				Time Period	Time Period	Time Period	
Parameter I		pDimensionA_Element		pDimensionA_Element	pDimensionA_Element	pDimensionA_Element	
Parameter I Value				ND;2012	ND;2013	ND;2014	
Parameter J		pDimensionB_Name		pDimensionB_Name	pDimensionB_Name	pDimensionB_Name	
Parameter J Value		Version		Version	Version	Version	
Parameter K		pDimensionB_Element		pDimensionB_Element	pDimensionB_Element	pDimensionB_Element	
Parameter K Value		Actual		Actual	Actual	Actual	

iii. One post-parallel thread where the parameters of process 'SYS_IBM_-_CubeSaveData' is configured to commit the processed data in the target cube to disk:



Cube Viewer: ParallelDataLoadMkII->SYS_IBM_TI_-_Parallel_Thread_Processing_-_Configuration->Default

SYS_IBM_TI_-_Parallel_Thread:Default

SYS_IBM_TI_-_Parallel_Thread_Configurati	-- All Threads	Pre-Parallel Thread	-- Parallel Threads	1	2	3	Post-Parallel Thread
Counter	5	1	3	1	1	1	1
Process		SYS_IBM_Cube_Combine		SYS_IBM_Cube_Combine	SYS_IBM_Cube_Combine	SYS_IBM_Cube_Combine	SYS_IBM_-_CubeSaveData
Parameter A		pSourceCube		pSourceCube	pSourceCube	pSourceCube	pCube
Parameter A Value		Operating Revenue and Expense FX		Operating Revenue and Expense FX	Operating Revenue and Expense FX	Operating Revenue and Expense FX	Operating Revenue & Expense
Parameter B		pTargetCube		pTargetCube	pTargetCube	pTargetCube	
Parameter B Value		Operating Revenue & Expense		Operating Revenue & Expense	Operating Revenue & Expense	Operating Revenue & Expense	
Parameter C		pZeroOutTarget		pZeroOutTarget	pZeroOutTarget	pZeroOutTarget	
Parameter C Value		N		N	N	N	
Parameter D		pZeroOutSource		pZeroOutSource	pZeroOutSource	pZeroOutSource	
Parameter D Value		N		N	N	N	
Parameter E		pClearTarget		pClearTarget	pClearTarget	pClearTarget	
Parameter E Value		Y		N	N	N	
Parameter F		pClearSource		pClearSource	pClearSource	pClearSource	
Parameter F Value		N		N	N	N	
Parameter G		pProcessData		pProcessData	pProcessData	pProcessData	
Parameter G Value		N		Y	Y	Y	
Parameter H		pDimensionA_Name		pDimensionA_Name	pDimensionA_Name	pDimensionA_Name	
Parameter H Value				Time Period	Time Period	Time Period	
Parameter I		pDimensionA_Element		pDimensionA_Element	pDimensionA_Element	pDimensionA_Element	
Parameter I Value				ND;2012	ND;2013	ND;2014	
Parameter J		pDimensionB_Name		pDimensionB_Name	pDimensionB_Name	pDimensionB_Name	
Parameter J Value		Version		Version	Version	Version	
Parameter K		pDimensionB_Element		pDimensionB_Element	pDimensionB_Element	pDimensionB_Element	
Parameter K Value		Actual		Actual	Actual	Actual	