TM1 Server Memory Diagnostic Playbook

Playbook Description

This playbook discusses select scenarios for TM1 Server memory consumption and growth and will provide guidance on how to diagnose causes of TM1 Server memory growth and an understanding of how TM1 Server has allocated memory. The following provides guidance on capturing logs/data/usage and techniques for identifying causes for TM1 Server memory consumption. This type of investigation is very application and usage specific, so guidance is intentionally broad.

Common Causes of Memory Growth

Known operations that can result in changes to the memory profile include:

- Changes to the Database/Application
- Upgrading the version of TM1 and/or utilizing new product features
- Multiple threads with the same user ID. Consider process initiated threads or concurrent TM1runti.exe usage
- Creating subsets/views
- Loading data
- Adding dimension elements
- View Zero Out
- Sandboxes - Large delta file on disk
- Modified VMM cube property (default is 128K)
- Server configuration parameter CalculationThresholdForStorage value set lower than default value of 50. A lower value can result in better performance, but higher memory usage.
- Idle sessions when IdleConnectionTimeOutSeconds is not set
- Dimension reordering
- Pre-calculating cubes / slices
- Feeder statements

Analysis Gathering

Memory Reporting

The first values to check are the system reported memory for tm1sd.exe and what TM1 is reporting as its Total Memory. This can be checked a few ways, but let's stick with perfmon.exe.

In the perfmon.exe log, compare the `Process:tm1sd.exe VirtualBytes` and `IBM Cognos TM1 <server name>: Memory: Total Pool Bytes`.

- These values should be close, within a few hundred MB. If so, skip to Identifying the trigger(s)
- If the difference much larger it could point to memory loss in the way TM1 interacts with a loaded library or the OS. In these cases, diagnosing the problem will be difficult at a customer location.

The following should be assembled and delivered to IBM Support.

- Operating System version and patch level
- All logs (listed below)
- Third-party connections (ODBC, ODBO, LDAP, etc.)

Logging

Enable/capture the following:

- Windows Performance Monitor Counters (log) (minimum of 5 seconds sample rate)
  - All `IBM Cognos TM1*` counters
  - All `Process*` counters
- TM1 Top/OpsConsole logging. (minimum of 5 second sample rate, 1-2 seconds preferred)
  - Start new logs daily, naming the log files appropriately
- Monitor TM1 Server Statistics. Specifically, the `}StatsForServer` and `}StatsByCubes` cubes. Provide export or provide sampling (screen shots/snapshot) every 30 min or so.

Package and send the above log files along with the TM1Server.log and TI Process error logs to Support.

Debugging

Identifying the Trigger(s)

If it looks like the Operating System and TM1 are not reporting the same memory utilization, then it will be necessary to interrogate the log files to identify the

trigger(s).
Continue looking at the perfmon.exe logs to identify upticks or spikes in memory usage (trigger point). Take note of the times these occur.
Look at the other logs at the noted times:

- Are other perfmon.exe counters that follow the same usage pattern?
    - Subset/view creation counts
    - Dimension or cube creation counts
    - Thread creation counts
    - Cache related counters
    - Other counts with same pattern
- Are the }StatsByCube samplings increasing?
    - Same usage pattern exist
- What is the usage pattern in TM1Top/OpsConsole log?
    - Identify activity before and after the time of uptick/spike. If users are in a wait or run state at the trigger point, understand the activity and cause that led to those states.
- What is the usage pattern in the tm1server.log?
    - Where processes being run?
    - Were there errors reported?

Identify User Types and Activities

Once the various log entries have been identified, develop the usage pattern. Different users have different agendas, being able to describe each thread's activity will aid in the investigation of the problem. Based on the TM1Top/OpsConsole logs, group activities in to the following categories:

- TI Processing
    - User initiated
    - Chore
- Client
    - Perspectives/CAFE
    - Architect
    - TM1 Web
        - Worksheets
        - Cube views
    - BI
    - Insight
    - Contributor
    - API/Scripting

Now that you have grouped thread activity, look back at the perfmon.exe log files and try to hone in (or rule out) threads.

Putting It All Together

Here are a few examples on how to use all the information captures to isolate a test case.

Scenario 1

- There are five active threads on the server, one of which is a scheduled chore that saves data.
- According to the perfmon.exe log, memory is increasing along with the subset and view creation counts.
- Based on the activity grouping exercise, three threads are using Websheets for reporting, one chore thread and one admin modifying security.

In this scenario, focus on the Websheet users, the chore and admin thread can probably be ruled out. Identify the Websheet(s) and how each thread is using it. Use that information as the basis for the isolated test case.
Run each Websheet test case to identify the scenario memory increases, compare that to common causes

Scenario 2

- There are five active threads on the server, one of which is a scheduled chore that processes data hourly.
- According to the perfmon.exe log, memory is increasing along with the dimension creation counts.
- The }StatsByCube snapshots also show a continuous increase in base level data, cube/feeder memory.
- Based on the activity grouping exercise, four threads are using BI for reporting and one chore thread.

In this scenario, all of the threads are likely culprits, but the BI threads can probably be ruled out. Investigate the TI Process(es) to determine the actions being performed. Its known that these chore loads data hourly, if the chore also adds/removes elements then this is the cause.
Run the chore in isolation. Verify the dimension creation count and }StatsByCube base level data increases at the same interval. In this case the

memory increase is likely related to the incremental load and the BI reporting being performed.

Scenario 3

Is there a problem?

- During business hours, activity averages 50 users on the server at any given time.
- According to the perfmon.exe log, memory increases over time, but generally plateaus for hours at a time.
- The }StatsByCube snapshots also show a continuous increase in base level data, cube/feeder memory.
- Based on the activity grouping exercise most threads are using Websheets that contain multiple reports, there are incremental data load threads and occasional administration threads.

After investigation there isn't one "spike" or groups of activities that explains the usage. In this case, looking at the garbage memory TM1 might provide a clue. After looking at closer at perfmon.exe logs, the `IBM Cognos TM1 <server name>: Memory: Garbage Pool Bytes` consistently increases by small amounts during "off" hours. After closer inspection you might see that the only thing running off hours is process that checks the ODBC source to see if a new data is available. Running this processes in isolation should be able to prove if this is the problem.