TADDM 7.2.1 Streaming Server Performance

Field Experience

A practical guide to tuning TADDM 7.2.1 streaming server discovery performance from a field user's perspective.

Jennifer Lee – TADDM L2 Software Engineer
11/11/2013 V1R1

Cloud & Smarter Infrastructure
Lab Services and Support
Dedicated to Client Success
IBM

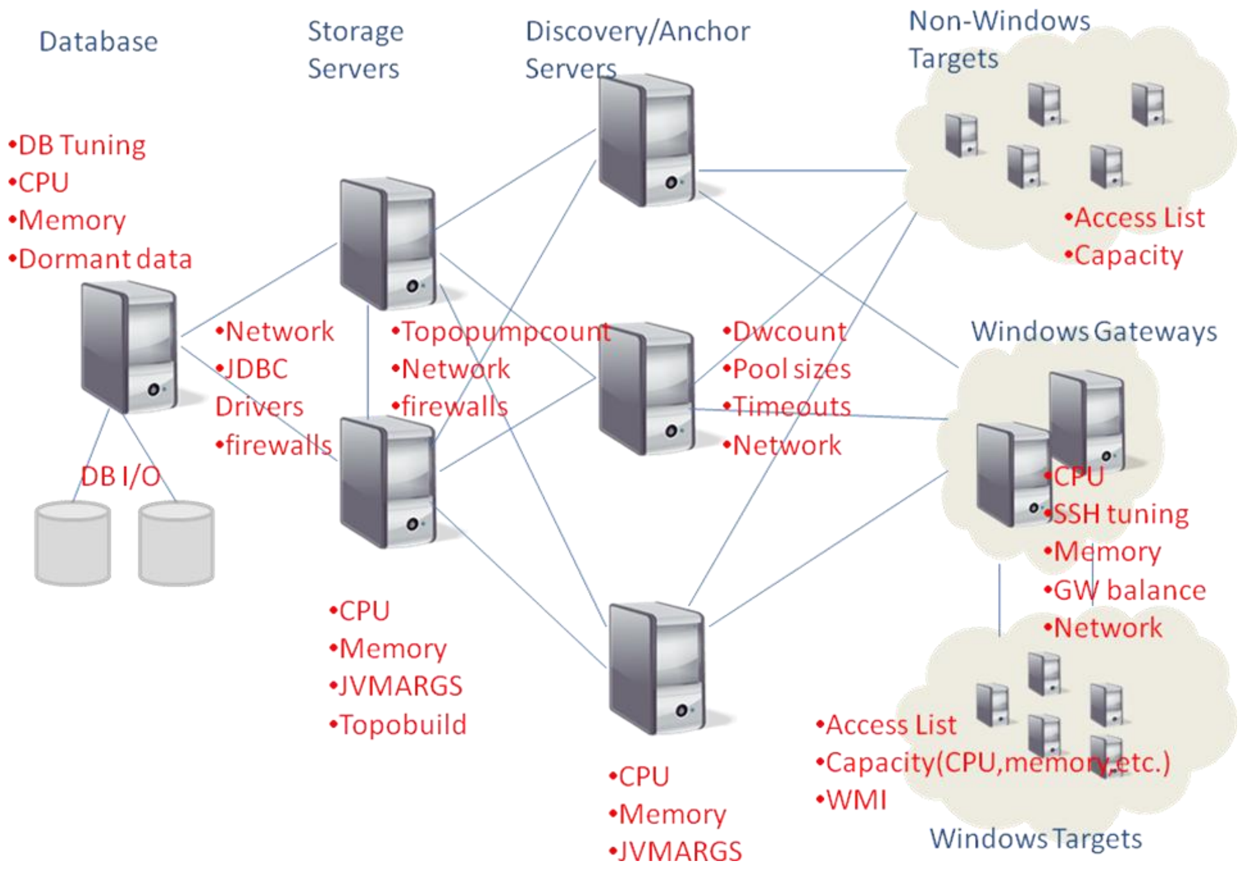
Contents

Overview

TADDM performance tuning is much more complex than most users envision. There are many factors and tuning points that interrelate and changing one area can have cascading and unexpected effects elsewhere.

The intent of this whitepaper is to provide an overview of all the areas in a TADDM 721 streaming environment that could potentially impact discovery performance so that users can tune their environment without risking unexpected results.

The following diagram includes the topics that will be covered in this document in red. All of these items together impact discovery performance. Note that this paper covers sensor discovery. While DLA discoveries are also impacted by the below and much of the information is relevant to bulk load, there are more parameters for this process which will not be covered at this time.



Understanding the limitations

Performance is often a factor of the environmental conditions in which TADDM resides. A properly tuned database and properly sized hardware are a must to even begin making adjustments to the tuning parameters to increase speed. It is assumed that the hardware and software in use in the TADDM 7.2.1 environment is at a supported level as documented in the TADDM Install Guide here:

http://pic.dhe.ibm.com/infocenter/tivihelp/v46r1/index.jsp?topic=%2Fcom.ibm.taddm.doc_721fp4%2FInstallGuide%2Fc_cmdb_server_req.html

Additionally, this whitepaper is intended to supplement the TADDM7.2.1 Server Sizing Guidelines for a Streaming Server Deployment. That documentation available here:

TADDM7.2.1 Server Sizing Guidelines for a Streaming Server Deployment

Both documents should be reviewed to ensure proper hardware sizing based on the expected count of Computer Systems stored in TADDM.

Most of this document is based on field experience related to TADDM 7.2.1 in streaming mode. However, it is also applicable to 7.2.2 with the caveat that as a new release, there is minimal field experience yet to determine what other factors may apply. Where changes between the releases are known they are noted within this document.

Some of the documentation can apply to TADDM 7.2.1's domain mode. In domain mode, however, everything runs on one server so most of the Storage Server and Discovery server topics are combined and in some cases are not relevant in this mode.

We will start from the left side of the below diagram and discuss each areas impact upon overall performance. Please note that if you are running TADDM in a virtual environment, you should work with your system administrator for CPU and memory monitoring. Operating system tools such as 'top' in Linux do not accurately reflect utilization due to the way hypervisors allocate resources.

The TADDM Database Server

The TADDM Database server is obviously an integral part of the performance equation. No matter what changes you make to the right of the diagram, if the database performance is not up to par, TADDM will be slow.

The key field issues that have been encountered in this area are below. **Before** proceeding with any TADDM specific tuning, all of these should be reviewed. If you are using DB2, please also review the tech article at this link for more details on DB2 performance;

http://www.ibm.com/developerworks/data/library/techarticle/anshum/0107anshum.html

1. Databases statistics - Old statistics can cause dramatic performance degradation; with poor statistics it could take 2 minutes to complete an operation that would normally take 2 milliseconds. If there is any major change to the database, such as thousands of new or deleted objects, update the statistics to ensure proper database performance in the future. Additionally the database should be reorged regularly as documented in the TADDM publications here:

   http://pic.dhe.ibm.com/infocenter/tivihelp/v46r1/index.jsp?topic=%2Fcom.ibm.taddm.doc_721fp4%2FAdminGuide%2Fr_cmdb_tuningdatabase.html

   Note – there is a recent APAR IV47906 regarding the use of the 'distribution all' clause on the DB2 database runstats command. This problem can cause performance issues and a fix is available. See the following link for more information;

   http://www-01.ibm.com/support/docview.wss?uid=swg21655664

2. The database software needs to be a supported level and a current fix pack. Running an older GA version of database software with no fix packs is known to cause performance problems.

3. When using Oracle, ensure you have supplied the correct JDBC driver to TADDM upon installation or upgrade. Mismatched drivers can cause performance issues as well as other issues with the TADDM software. If using Oracle 11 copy ojdbc5.jar to your TADDM installation and overlay or link all copies of oracle-jdbc-9.2.jar to it. If using Oracle 10 copy ojdbc14.jar.

4. Database I/O is critical. For instance, if the database is writing to a SAN and the fiber is over utilized (more then 80%) then disk reads and writes will be impacted which in turn slows down TADDM performance. Database logs should be placed on separate disk drives (RAID) / disk arms (SAN) from the TADDM database tables. This is an extremely critical point; increasing TADDM parameters such as topopump and dwcount will have a negative affect if the I/O to the database is already saturated.

5. The network speed between the database and the storage server should be 1 GB. Ensure no errors on the installed interfaces using the appropriate tool for your OS. (For example, netstat on AIX or ifconfig or ethtool on Linux.)

6. If there is a firewall between the database server and the TADDM servers, it is critical that it does not drop connections that are idle. This will cause TADDM to fail silently and can manifest as a performance issue. If possible, do not allow the firewall to timeout any connections between the TADDM servers and the database. If using an Oracle database you can set sqlnet.expire_time to a number less than the firewall timeout to ensure that the connections never appear idle to the firewall.

7. While CPU and Memory are typically not an issue on the database server if the database is appropriately sized per the TADDM documentation, it is wise to monitor this data during the discovery window to ensure no issues as lack of either can severely degrade performance.

8. Monitor and tune database configuration parameters. It is important that the database resources are properly tuned for the discovery workload. Typically a DBA monitors and ensures this activity occurs. Some resources to perform this task include:
   For both DB2 and Oracle, we recommend the IBM Performance Analysis Suite. This tool can be downloaded at this link:

   https://www.ibm.com/developerworks/community/groups/community/perfanalyst

   Refer to the usage guide for details on how to use it. This tool can analyze both DB2 and Oracle data inputs and point out potential issues.

   For Oracle – refer to the TADDM documentation here for information on tuning the buffer cache (SGA):

   http://pic.dhe.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_721fp4/AdminGuide/r_cmdb_tuningonlyoracle.html

   For DB2:

   The tech article noted earlier:

   http://www.ibm.com/developerworks/data/library/techarticle/anshum/0107anshum.html

   The TADDM publications:

   http://pic.dhe.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_721fp4/InstallGuide/r_cmdb_upgrade_guidelines.html

   Specifically:
   The following DB2® settings provide good results in the lab environment for enterprise size database that consists of 10,000,000 CIs:

   ```
   db2 update db config using SORTHEAP 8000
   db2 update db config using DBHEAP 8000
   db2 update db config using APPLHEAPSZ 2000
   db2 update db config using UTIL_HEAP_SZ 8000

   db2 update dbm config using UTIL_IMPACT_LIM 95
   db2 update dbm config using SHEAPTHRES 16000
   ```

   Additionally, you can change the buffer pools to:

```
db2 alter bufferpool IBMDEFAULTBP size 240000
db2 alter bufferpool BUF8K size 40000
db2 alter bufferpool BUF32K size 8000
```

9. Managing dormant data and clearing the change_history_table on a regular interval is important. The publications document clearing change history here: http://pic.dhe.ibm.com/infocenter/tivihelp/v46r1/index.jsp?topic=%2Fcom.ibm.taddm.doc_721fp4%2FAdminGuide%2Fr_cmdb_deleteolddata.html

NOTE: the aliases_jn cleanup referenced on this page is for 7.2.1 fix pack 3 only, in fix pack 4 and later a Topology builder agent clears this table. It is recommended to clear dormant CIs to ensure TADDM data is as accurate as possible and not to waste DB space which can decrease performance over time. It is wise to clear dormant data on a regular interval as it is less resource consuming to do this over time rather than occasionally. For more information on this topic please see:

https://www.ibm.com/developerworks/community/blogs/TADDM/entry/dormant_data_cleanup_in_taddm_a_customer_story?lang=en

10. It is helpful to keep statistics over time on database information. The easiest way to do this is to run the healthcheck script in the dist/bin directory of your primary storage server at least once a month and save the results after reviewing them for any issues. If there is a future performance issue, this data can show database growth over time.

How do I know which of the Storage Server or the Discovery Server is bottlenecked?

These rules from the TADDM7.2.1 Server Sizing Guidelines help you determine where you should focus your tuning: the Storage Servers or the Discover Servers.

While monitoring a large discovery through the UI:

1. If the number of sensors running is much greater than the dwcount (com.collation.discover.dwcount in collation.properties on the discovery server) then TADDM is waiting to store data. If the bottleneck still occurs after working through the database and Storage Server sections of this whitepaper then you will need to increase the capacity of your Storage Servers by adding another Storage Server.

2. If the number of sensors running is about the same as your dwcount (com.collation.discover.dwcount in collation.properties on the discovery server), then TADDM is waiting for sensors to run. The storage servers are able to store the results from the sensors

without a bottleneck. You should refer to the Discovery Server section of this document to ensure that your configuration for discovery is not leading to slow sensors. If the bottleneck still occurs after making the changes recommended by the Discovery Server section, you will need to increase the capacity of your Discovery Servers (add another Discovery Server, gateway, or increase the dwcount count).

The TADDM Storage Servers

The great part about TADDM 7.2.1 is its scalability. If storage is too slow, add another storage server. If discovery is too slow, add more discovery servers. But before doing this, you should ensure that your current servers are fully utilized.

If you are familiar with TADDM prior to 7.2.1 you might recall that it was always a very memory intensive application. What you may not realize is TADDM 7.2.1 was greatly optimized and is now much more CPU intensive. While TADDM still needs plenty of memory, CPU is often driven to 100% before it runs out of memory. So the first thing to do is to ensure your TADDM storage servers are not spiking the CPU. For storage servers you want to have as many CPUs as possible with the fastest processors.

Outside of CPU, here are other items to look at for Storage Server performance:

1) JVMARGS – your server only has so much memory; do not increase your TADDM jvm arguments such that the sum of them is more than your available real memory minus OS overhead. If the TADDM server runs out of real memory and pages to disk, performance is atrocious and TADDM can silently fail. Monitor paging and memory usage to ensure no problems in this area. If you run other applications on this server remember that they consume memory as well. 8G of memory can be consumed quickly by TADDM with the default settings; if other apps are consuming memory then 8G will not be enough memory. By default, TADDM 7.2.1 Storage Servers set the following JVM memory sizes;

- StorageService – this is set by default in dist/deploy-tomcat/ROOT/WEB-INF/ storage-server-context.xml. The default setting is "-Xms768M|-Xmx1512M|-DTaddm.xmx64=4g" which means the JVM will start with 768M of memory and be able to expand to 1.5G if a 32 bit OS and 4G if a 64 bit OS. The collation.properties setting com.collation.StorageService.jvmargs is used to customize this setting and will override the value in storage-server-context.xml.

- Gigaspaces (also known as JSpaceServer) – is set in the collation.properties setting com.collation.GigaSpaces.jvmargs as "Xms128M -Xmx768M" by default

- Tomcat (Apache) – is set in the dist/bin/control script as "Xms64M -Xmx1024M" and can be over ridden by the collation.properties setting com.collation.Tomcat.jvmargs.

- If you also run bulkloads on this host, that can consume another 1G, or whatever Xmx value is in dist/etc/bulkload.properties for com.ibm.cdb.bulk.allocpoolsize.

  As you can see, the sum of the maximum settings above is 4G + 768M + 1G (without bulkload), so out of the box a storage server can use 5.7G of memory. If you only have 8G of memory available, with OS overhead and other applications, this may be exceeding the memory available.

  Just because you have a lot of memory does not mean you should increase the maximum memory available to the StorageService JVM. In most cases, the default 4G Xmx is enough. You can increase it to 6G if there is a need for it, but only increase it higher if the problem has been analyzed and no other solution found. If you increase it much larger than 6G there are two potential issues:

    a. If the JVM does run out of memory, Support will not be able to read the extremely large heap dump and you will need to re-create the issue with a smaller amount of memory.

    b. Depending on the problem this could leave behind a much larger set of objects to have to clean up manually. This is not typical, but it has happened. If your TADDM storage server is hitting an out of memory condition, it is always best to open a PMR with Support to review it and ensure there is not a code problem rather than just increasing the memory.

2) Topopumpcount – many customers believe they can increase their performance simply by changing this one parameter in collation.properties on the discovery server:

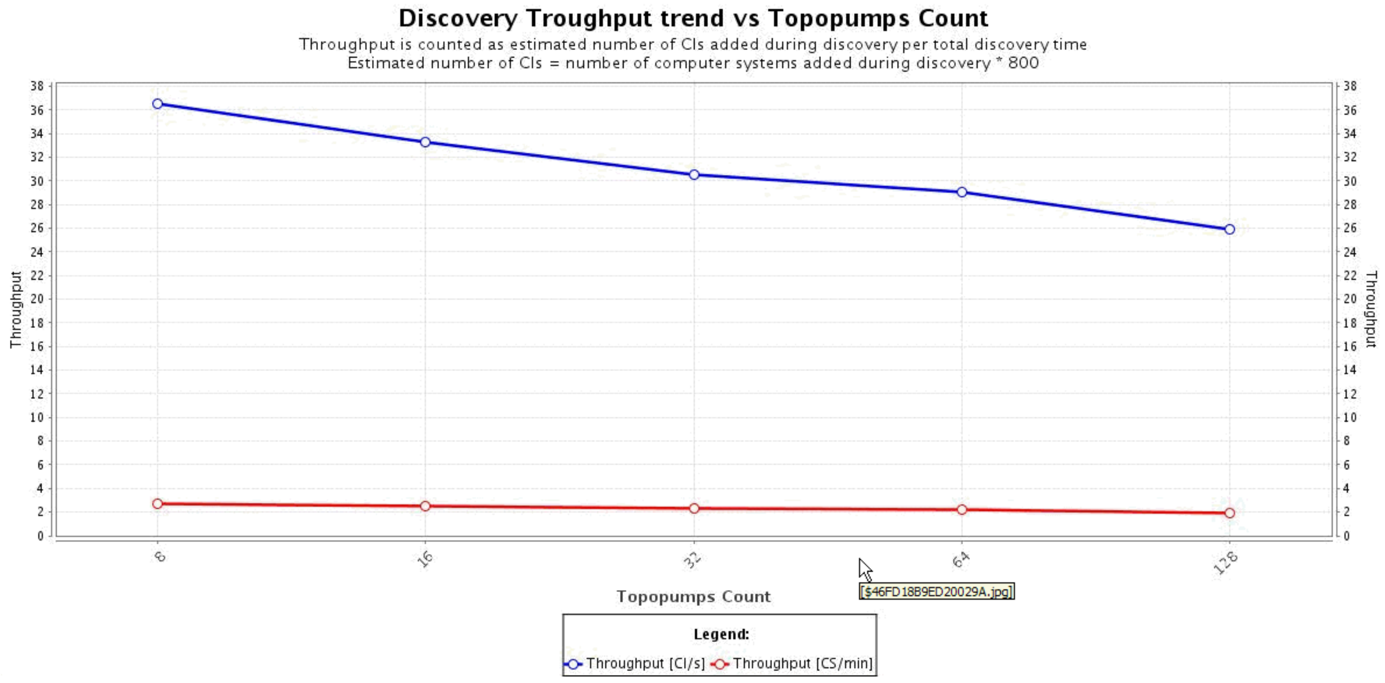
   com.collation.discover.observer.topopumpcount

   The default value of this property is 16. Please review the following technote for further documentation on this parameter: http://www-01.ibm.com/support/docview.wss?uid=swg21634124

   The short story here is increasing the value of this property tends to have a negative effect on performance. In the lab and in the field, a value of 8 actually performed slightly better than 16.

   If despite the above you still increase topopumpcount you should realize that you are adding load to the storage server and you may have to increase the memory (including JVMARGs) and CPU. This property should not be changed without careful monitoring of system resources before and after the change.

   The chart below illustrates the degradation of discovery throughput as topopumpcount is increased in the lab using 3 servers (1 Database, 1 Storage Server and 1 Discovery Server);



3) Topology Builder – The Primary Storage Server runs topology builder agents on regular intervals. When these are running, this storage server is marked 'busy' if there are other storage servers available. For this reason, we generally recommend at least two storage servers so that discovery storage does not compete with the topology builder agents. If you only have one storage server, consider a second. You can read more about the topology builder agents on the TADDM blog here:

   https://www.ibm.com/developerworks/community/blogs/TADDM/entry/all_you_ever_wanted_to_know_about_topology_builder_agents_issue_1_the_basics?lang=en

   In general we do not recommend changing the intervals that the topology builder agents run at unless there is a definitive need for improved performance. Changing the intervals often does not improve performance much; it just delays the work and causes more work to be done at the longer interval. For the cleanup agents, most process a set row count each time, so if you extend their interval they may never finish processing all the rows. However, it is OK to slightly increase the dependency group interval to allow the primary storage server to have less 'busy' time and participate more fully in sensor storage. By default this group runs every 30 minutes; you could change this by adding the following collation.properties entry on your Primary Storage server:

```
com.collation.topobuilder.agent.DependencyAgentGroup=Dependency@2.0
```

This example sets the group to run every 2 hours rather than the default 30 minutes.
Then monitor the group during and after discovery to determine if it is running much longer then it was at the smaller interval. You can monitor the amount of time the group takes to run in the TADDM.log file (TADDM-StorageService.log for TADDM 7.2.2).

The other groups can also be set as follows, but generally the default 4 hours is sufficient and we do not recommend changing these without thorough testing of the impact;

```
com.collation.topobuilder.agent.BackgroundAgentGroup=Background@4.0
com.collation.topobuilder.agent.CleanupAgentGroup=Cleanup@4.0
```

If you only run bulkload and no sensors, then it is appropriate to change the topology builder agents to run outside the bulkload window, after it completes.

NOTE – in TADDM 7.2.2 the format of the above properties has changed to the following;

```
# Dependency group agents run period (in hours e.g. 1.0)
com.ibm.cdb.topobuilder.groupinterval.dependency=1.0
# Background group agents run period (in hours e.g. 4.0)
com.ibm.cdb.topobuilder.groupinterval.background=4.0
# Cleanup group agents run period (in hours e.g. 4.0)
com.ibm.cdb.topobuilder.groupinterval.cleanup=4.0
# BizApps group agents run period (in hours e.g. 4.0)
com.ibm.cdb.topobuilder.groupinterval.bizapps=4.0
# Integration group agents run period (in hours e.g. 3.0)
com.ibm.cdb.topobuilder.groupinterval.integration=6.0
```

4) Topology Builder agents – typically each topology agent runs in a matter of minutes. After a large discovery some agents may run an hour or two but all the agents should run within a half day or less. See the diagnosis section of this document for more information on monitoring topology builder agents.

The TADDM Discovery Servers

As noted earlier, a discovery server could be a source of a bottleneck if the discovery status shows the number of sensors running close to the number set in the com.collation.discover.dwcount property. There are a variety of issues that could slow down the Discovery Server:

1. Sensor timeouts have been increased, causing sensors to run much longer. There is a training module on this topic; the bottom line is timeouts should be changed with care and do not change the default agent timeout (com.collation.discover.DefaultAgentTimeout) from its initial value of 10 minutes. Please review the timeout training here:

   http://publib.boulder.ibm.com/infocenter/ieduasst/tivv1r0/index.jsp?topic=/com.ibm.iea.taddm/taddm/7.2.1/troubleshooting/diagnose_sensor_timeouts_player.html

2. If running Windows discovery, review the TADDM Windows gateway section of this document below. A gateway can easily slow down discovery if not properly configured.

3. Both statements about JVMARGs in the Storage Server section of this document also apply to discovery servers. Please review those on the discovery server as well. For a discovery server, the following jvm memory settings apply;

- DiscoveryService – this is set by default in dist/deploy-tomcat/ROOT/WEB-INF/ discovery-server-context.xml. The default setting is "-Xms256M|-Xmx1512M" which means the JVM will start with 256M of memory and be able to expand to 1.5G. This can be over ridden by collation.properties setting com.collation.DiscoveryService.jvmargs if customized.

- Discover – this is set by default in dist/deploy-tomcat/ROOT/WEB-INF/ discovery-server-context.xml. The default setting is "-Xms128M|-Xmx1024M. The collation.properties setting com.collation.Discover.jvmargs is used to customize this setting and will override the value in discovery-server-context.xml

- JSpaceServer (Gigaspaces) – is set in the collation.properties setting "com.collation.GigaSpaces.jvmargs" by default as "Xms128M -Xmx768M".

- Tomcat (Apache) – is set in the control script with a default value of "Xms64M -Xmx1024M". It can be over ridden by the collation.properties setting com.collation.Tomcat.jvmargs.

- If running discovery of applications that use local anchors, such as Websphere, the local anchor can also use .5G of memory as shown in this collation.properties setting: com.ibm.discover.localanchor.jvmargs=-Xmx512M

  As you can see, the sum of the maximum settings above is 1.5G+1G+768M+1G+512M, so out of the box a discovery server can use 4.7G of memory. If you only have 8G of memory available, with OS overhead and other applications, you may be exceeding the memory available.

4. CPU – while not typically an issue on a discovery server, this should be monitored to ensure no problems, especially if running at a low specification.

5. Network – the discovery server has a lot of activity to the targets and should have the fastest network speeds and no errors. Monitor the interfaces to ensure no errors or dropped packets.

6. Sensors –do not run redundant sensors. For example, never run both the HostStorageSensor and the StorageSensor for the same IP in one discovery. This can cause large performance degradation because both sensors collect and store almost the same data. If you want to collect fiber channel data, run the host storage sensor, otherwise run only the storage sensor.

7. Sensors - run only what you need to run and disable sensors for data you do not intend to use. For example, if your scopes include Windows systems, but you do not want any Windows system data stored in TADDM, you can disable scanning of ports 135 and 445 (WMI/DCOM and SMB) in the Port Sensor configuration in your discovery profile to prevent Session Sensors being started for Windows computers. You can also disable sensors for applications when you have no need for that data. Additionally, if you only want low level discovery of an application, you could create a custom server template to gather basic data and disable the out of the box sensor. This often requires only OS credentials and not application credentials which will ease initial setup. Custom sensors provide basic information necessary for the application, while running much quicker as less data needs to be collected and stored. It is important to note that if you find out later on that you need deeper data, the custom applications may not merge with the out of the box sensor applications. In this case you will need to delete the applications and re-discover.

8. Concurrent discovery – running discoveries in parallel improves performance as often in a specific scope there will be some sensors that take a long time to run and some that do not. As discovery gets closer to the end, there may only be 2-3 sensors still running and they might run for another 2 hours. For example, Virtual Center discovery takes far longer than a smaller application such as a database server or a single computer system sensor as Virtual Center sensor may be storing hundreds of computer systems. If you run 2-3 Virtual Center sensors in context of a larger scope chances are the last two or three hours only those Virtual Center sensors are running leaving many resources to spare. Concurrent discovery fills those gaps and ensures all discovery worker and topology storage threads are in use as much as possible.

9. Scope size – If you are using concurrent discovery then scope size is not relevant to performance other then it is best to stay under 64K IP addresses to avoid memory issues. If you are running only one discovery at a time, there will be better performance running larger scopes then small ones as you will be taking advantage of all available threads for as long possible. For manageability purposes, a scope size of 'pingable' IPs in the low to mid 1000's generally work the best. This size will be able to take advantage of the thread count for several hours.

10. When choosing scopes to discover, try not to scope by application or computer system type. Similar sensors store to the same database tables, which introduces more database locking. For example, if you choose to discover all Windows at once, there will be a lot more contention on

Windows based database tables which can degrade performance. If performance is critical, it is better to discover on a more random basis with a good mix of applications and OS types. Many customers discover on a subnet basis using a master exclude scope for targets they do not want included.

11. Access Lists - TADDM will try each applicable access list entry from top to bottom to access a resource. It will repeat this 5 times by default (com.collation.SshWeirdErrorTryCount=5). If we do not have access to a target with any access list entry, the process of running thru all the entries 5 times each can take a long time -- in the field this can take 25 minutes or more if there are several access list entries in scope. That is a sensor thread being wasted for no results; it is best to remove that IP from scope using a scope exclusion so the 25 minutes can be used by a working sensor. Many customers maintain a 'permanent exclude' scope, which is essentially a non-existing range or subnet, followed by all IPs to exclude. This scope is then used in conjunction with all discoveries to ensure TADDM does not try to discover these non-accessible targets that answered the initial ping. The other possible scenario is having 10 non scope restricted access list entries, and the one the target uses is the last one. That means we will spend time trying 9 non-working ones before we get to the working one. This can be avoided by properly limiting the scopes on access list entries.

12. Pool sizes – these are discussed in detail in the timeout training at this link:

    http://publib.boulder.ibm.com/infocenter/ieduasst/tivv1r0/index.jsp?topic=/com.ibm.iea.taddm/taddm/7.2.1/troubleshooting/diagnose_sensor_timeouts_player.html

    However it is good to specifically note the following here as they do play a role in discovery performance;

    - TADDM by default limits the number of sessions to a single UNIX target to three:

      com.collation.platform.session.PoolSize=3

    - TADDM by default limits the number of sessions the discovery server can have with each gateway to 20:

      com.collation.platform.session.GatewayPoolSize=20

    Keep in mind that increasing pool sizes does increase the load on the targets and gateway, so while increasing these parameters can improve performance, they can also degrade performance if the target and/or gateway cannot handle the additional load.

    You can also monitor how long your sensors are waiting for a pool resource by setting collation.properties value com.collation.platform.session.ExtraDebugging=true on the discovery server and checking the logs for 'pool lock'. See the following link for more information on this setting;

http://pic.dhe.ibm.com/infocenter/tivihelp/v46r1/index.jsp?topic=%2Fcom.ibm.taddm.doc_721fp4%2FAdminGuide%2Fc_cmdb_tuningdiscovery_drate.html

13. Discover Worker Thread Count -- com.collation.discover.dwcount – this property is discussed almost last in the list because all of the above needs to be considered first. This property controls how many sensors can run at once on the discovery server. Increasing it will increase the load on the discovery server, gateways and any anchors. After making any changes to the dwcount, monitor the CPU and memory usage on those servers and ensure they are properly tuned. Gateways must also have all of items listed in the The TADDM Windows gateways section below reviewed as increasing dwcount can degrade performance if the gateways are not properly configured.

    In the field, several customers have found the optimal dwcount to be 96 with a topopumpcount of 8 or 16. This is with two storage servers, Primary and Secondary, and 2 discovery servers running discovery at the same time. If you have more than two discovery servers running discovery at any given time you should adjust the value lower such that a backlog does not build up during storage, e.g. Many more sensors 'running' (in the GUI) then the dwcount value would indicate a storage backlog. If that occurs lower dwcount or add storage servers to accommodate the increased speed of incoming sensor result storage requests. If increasing dwcount to 96 (or similarly high) you must also increase the Xmx for both Discover and DiscoveryService with the following changes or additions to the collation.properties file:

    com.collation.Discover.jvmargs=-Xms128M –Xmx2048M

    com.collation.DiscoveryService.jvmargs=-Xms256M –Xmx2048M

    IMPORTANT – when increasing memory, always be sure that much memory is available on the TADDM server. The above represents a 1.5G increase in memory from the default maximum settings of 1024M and 1512M respectively. The additional allocated memory must be available otherwise such a change will degrade performance.

    NOTE –The jvmarg properties above are the proper syntax for TADDM 7.2.1. In TADDM 7.2.2 the jvmarg properties require a vendor specific code at the end of the property. Using the settings above in TADDM 7.2.2 over write the xml properties and will cause some TADDM functions, such as VMM integration, to fail. See this technote for more information: http://www-01.ibm.com/support/docview.wss?uid=swg21648976

    Additionally, TADDM 7.2.2 has slightly improved performance. In the lab, with a one Primary Storage Server and one Secondary Storage server with **a total** dwcount of 256 (number of discovery servers * dwcount; 2 * 128) provided optimal performance with the default topopumpcount of 16. In this test case, the discovery service Xmx JVM argument was increased an additional gigabyte to 3G.

14. Anchors – when using anchors, the TADDM discovery server has to deploy much of the TADDM code to the anchor during initial deployment which can be very time consuming. Setting the collation.property; com.collation.discover.anchor.lazyDeployment to true limits some of the data which is sent to the anchor and can speed up anchor deployment. However, if you are discovering Websphere, Weblogic or Oracle thru the anchor, you should leave this property false until the anchor has been successfully deployed once. Refer to the charts at the following link for more details on the bandwidth savings of lazy anchor deployment; http://pic.dhe.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_7.2.2/InstallGuide/r_cmdb_sizing_hw_anchor_taddm_transfer.html

The TADDM Windows gateways

The following topics apply to Windows gateways:

1. Proper CPU and memory must be allocated. Gateways can be CPU intensive. Ensure fast processors and enough of them to support the load. Monitor your gateways to ensure no bottlenecks.

2. Check both the Windows and ssh logs periodically for any indication of error conditions. Persistent errors can slow down the discovery as TADDM does retry on certain failure conditions which take additional processing time.

3. If you have more than one unscoped gateway and also have scoped gateways on the same discovery server, ensure you have properly balanced them. See the following technote for details:

   http://www-01.ibm.com/support/docview.wss?uid=swg21596380

   Essentially you want to evenly balance the gateway order such that scoped and unscoped gateways are spread equally through the list.

4. Cygwin and Bitvise are configured out of the box to reject ssh sessions when too many arrive at once. Such a scenario is considered a denial of service attack. This can happen even with a discovery of only 10 Windows targets. See the following technote for details on how to configure your ssh software to allow a higher number of concurrent logins:

   http://www-01.ibm.com/support/docview.wss?uid=swg21570126

5. If your gateway is running Windows 2008 SP2, ensure the fix noted at the following technote has been applied:

   http://www-01.ibm.com/support/docview.wss?uid=swg21508691

6. Gateway pool size – as noted earlier, only 20 sessions are allowed at once to a given gateway. If all the above is completed, consider slowly increasing the com.collation.platform.session.GatewayPoolSize property. Try 40 sessions and monitor the gateway resources, Windows logs and the sshd logs to ensure no issues. If you see more Windows sensor timeouts or unexpected errors with the higher setting then the gateway is not handling the increased workload.

TADDM Maintenance

TADDM is constantly evolving; performance is typically improved with each TADDM release or fixpack. For this reason, running the latest stable release or fix pack is always recommended. See the TADDM Support site for more information:

http://www-947.ibm.com/support/entry/portal/Overview/Software/Tivoli/Tivoli_Application_Dependency_Discovery_Manager

Subscribe under the notifications topic to get notified of the latest news and releases for TADDM so that you can plan ahead to keep your maintenance up to date.

Other collation.properties settings that impact performance

The following additional collation.properties settings can impact TADDM performance:

1. DEBUG logging – often necessary for any kind of problem diagnoses but can add a slight overhead due to the additional disk IO.

2. com.collation.discover.observer.topopumpdeadlockstrategy=avoid
   This property single threads each IP's sensor storage so there is no DB contention. This is useful if you have a lot of large applications running on one server. For example, 5 instances of Websphere on one server storing at the same time can cause contention and deadlocks. This property should only be used when running larger scale discoveries so that performance is not degraded. If the deadlock strategy is set to avoid during the discovery of a small number of IPs, each IP that is discovered will store sensor results sequentially (single threaded storage per IP) and will wait for the first sensor to complete before it stores the next one. In a discovery of a single IP with many sensors or a substantial amount of discovered data this can significantly increase storage time.

3. com.collation.discover.supressruntimeprocs – A runtime process is a process running on an operating system. It has PID and it may listen on some ports but it does not have to. In TADDM terms, an Application Server is more than just a runtime process. It is a running application which listens on some ports and can be a single process or multiple processes. The GenericServerSensor discovers all such processes. Setting com.collation.discover.supressruntimeprocs=true keeps the runtime processes from being stored in the DB unless a customer server template (CST) is already defined for it. If you are not currently configuring CSTs for unknown processes, setting this to true can save the overhead of storing them. This will improve the performance of storing the GenericServerSensor result and also the performance of the topology builder cleanup agents.

4. Jdo fetch batch size, com.ibm.JdoQuery.FetchBatchSize=1000, determine how many rows to fetch from the database at a time. In general you should not change the default as too low of a setting will degrade performance and too high of a setting can cause an OutOfMemory.

5. com.collation.topomgr.generateExplicitRelationship – this property was removed in TADDM 7.2.0 and no longer used by TBSM. It should not exist at all; if it does it must be false. A setting of true can cause up to 50% performance degradation of discovery. There are post procedures to remove data stored due to this setting. If you had this property set to true for more than a short period of time, contact Support for assistance.

6. com.collation.security.enabledatalevelsecurity – this property should be enabled with care, the granular performance checking needed for data level security has some overhead, especially in the user interface.

Discovery Targets

Discovery targets can cause performance issues when they do not respond to the sensor driven requests in a timely manner. If you use the default timeouts then the delay will not be more than 10 minutes. If you increase the timeout and the sensors actually use it, then discovery will take that much longer. If you have a lot of targets with performance issues an attempt should be made to resolve the issue on the targets. This topic is covered in more detail in the training module here:

http://publib.boulder.ibm.com/infocenter/ieduasst/tivv1r0/index.jsp?topic=/com.ibm.iea.taddm/taddm/7.2.1/troubleshooting/diagnose_sensor_timeouts_player.html

Some other target based issues that can cause performance problems are:

1. Capacity of the target – By default TADDM can run 3 simultaneous commands on a given non-Windows IP as noted in the pool size topic under the Discovery Server section earlier in this document. Typically this is not an issue, but occasionally older systems will be at full capacity and will not be able to respond to the commands. Such targets should be fixed or removed from scope. Watch this particularly when increasing poolsize because this increases the target workload.

2. For Windows based targets, WMI must be working for discovery to be successful. WMI problems can slow down Windows discovery and should be corrected or the target removed from scope.

3. Access issues – both incorrect logins and missing permissions can force retries that delay sensors resulting in slower performance.

Other Issues that can cause performance degradation

Errors – The TADDM administrator should keep an eye on the error.log, especially on the storage servers. Errors can certainly degrade performance as retries cost time. There will always be errors, but watch for larger than normal amounts. If the error.log looks like its looping thru 20 logs in a day, there is something to be concerned about.

How to diagnose a TADDM performance issue

The following 4 steps can help you diagnose performance issues;

Step 1: Set a goal

Performance issues are very arbitrary. Some users are happy to discover 10,000 targets in a week while others want the same in hours. TADDM 7.2.1 has great capability to scale to meet each user's goals. In the end, however, it is the hardware that plays the key role in the outcome, but the settings noted above can help. Before embarking on diagnosing a performance problem, you will need a specific goal. For example, 'my discovery currently takes 2 days but I only have 1, what do I need to do to get there'? Setting a goal will help determine what steps are worth taking. For example, more hardware might buy you 10 hours; is it is worth the cost?

Step 2: Isolate Bottlenecks

Once you know your goal, the first step to getting there is to isolate the bottlenecks in your current performance. Setup monitoring of the CPU and memory, including paging, on all servers including the gateways and anchors. For virtual machines ensure the monitoring is being done from the parent perspective as virtual machines do not always get what is believed to be allocated. Monitor I/O, particularly the database I/O, as well. This is often an area of bottlenecks in TADDM 7.2.1 as the database storage is rapid and any slowness during the database writes can substantially impact performance. Have your DBA setup database monitoring such as DB2 snapshots, or Oracle Automatic Workload Repository (AWR) reporting. See the Appendix for more information on database monitoring. Once this is setup, run your weekly discovery then review the data to ensure there are no resource issues. If there are any, address those first.

Step 3: Check sensor execution

Once you have determined that CPU, memory, database and I/O are fine, the next place to look would be for obvious sensor issues. Sensors have two phases of execution;

Discovery time – the time the sensor spends gathering information from the target.

Storage time – the time it takes to store the data collected into the database.

Each phase needs to be looked at individually. As I mentioned earlier, you can tell from the GUI by the number of sensors 'running' if we are waiting to store or if we are waiting to run sensors. There are other ways to do this, including using logs or the discevnt database table, but I found it easier to use grep on the DEBUG logs to quickly sort out performance issues with specific sensors. If you are running on Windows, you can install a unix toolkit such as Microsoft's free SFU or Cygwin to have access to unix commands such as grep.

To perform this task, do the following before your weekly discovery;

1. Stop the discovery server and clear the log directory
2. Set Discover and DiscoveryService in DEBUG mode
3. If your discovery will last more than a day, increase the com.collation.log.filecount to a value such as 25 or 50 to ensure we have enough logs to pull the metrics from. Ensure the file system has enough space to allow this many more logs.
4. Start TADDM
5. Run your discovery.

When discovery is complete, you can get discovery and storage times with these grep commands:

Storage time (from dist/log/services);

```
grep "activity=Storing" DiscoverO* > sensorstoragetimes.out
```

Discovery time (from dist/log/sensors):

```
grep -r "AgentRunner total time" > sensordiscoverytime.out
```

Import the files into a spreadsheet using space and the '=' sign as delimiters. For the storage times, sort the report on the value of "runtime=", for the discovery time sort on the value of "AgentRunner total time =". These times are all in milliseconds, you may want to use a formula to convert to minutes for an easier read but that is not required.

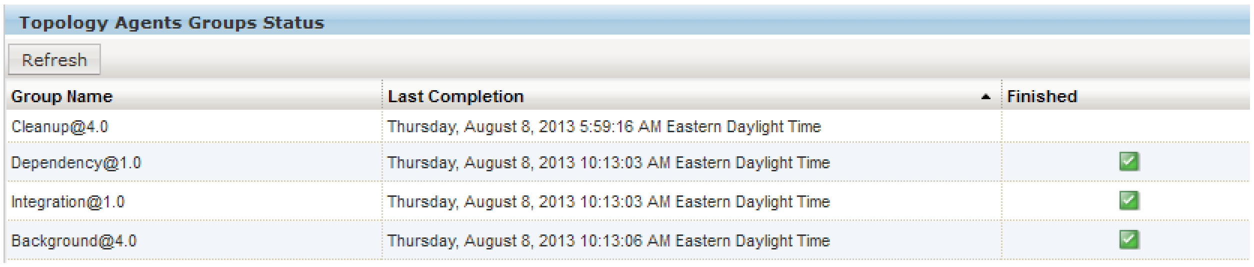
Review the resulting reports.

For discovery, most sensors should take under the default 10 minutes. There will be exceptions, some large applications or Windows servers with a lot of software components can take longer. If running more than 10 minutes is the norm rather than the exception, however, and it is not a Virtual Center or large application, you may want to look deeper into the cause or open a PMR and send the DEBUG sensor logs for the affected sensors.

The same concepts apply to storage times. The VirtualCenter sensor can run for several hours because it is typically storing 100's of computer system objects. Large database and web server discovery can also run for an hour or two. The Generic Server sensor can take a long time on large application servers, especially if you are not suppressing runtime processes. The more applications a server runs, and the larger they are, the longer they take to store. A typical Windows or Linux computer system sensor will take 1-5 minutes to store in a well configured environment, if you are consistently seeing higher than this, look closer at the database section of this documentation. If you need further assistance diagnosing storage times, open a PMR with TADDM Support and include the DEBUG logs from the Discovery Server, as well as TopologyManager logs from the storage servers that cover the time of storage. Database documentation will also be required such as snapshots during the storage time frame and details database config information.

Step 4 – check the performance of the Topology Builder agents.

In addition to specific sensor metrics, TADDM administrators should get an understanding of their topology builder agents and the time they take. As mentioned earlier, these agents should not take days to run. After a discovery they may take a few hours, but typically all groups should be completing at least twice a day when using the default intervals. You can monitor the agents via the TADDM Data Management Portal Discovery → Topology Agents Groups Status pane. For example;


Topology Agents Groups Status

Refresh

Group Name	Last Completion	Finished
Cleanup@4.0	Thursday, August 8, 2013 5:59:16 AM Eastern Daylight Time	
Dependency@1.0	Thursday, August 8, 2013 10:13:03 AM Eastern Daylight Time	✓
Integration@1.0	Thursday, August 8, 2013 10:13:03 AM Eastern Daylight Time	✓
Background@4.0	Thursday, August 8, 2013 10:13:06 AM Eastern Daylight Time	✓

The above indicates that 3 of the 4 groups completed ay 10:13AM on August 8th and the 4th group, the ‘Cleanup’ group is currently running. If it stays in this status for more than 4 hours investigate the agents to see if one is taking an unexpectedly long amount of time. This can be done by reviewing the TADDM.log on the Primary Storage Server(TADDM-StorageService.log if 722) and looking for the latest ‘agent’ messages. For example:

Note the current time:

```
log]# date
Fri Aug  9 10:05:43 CDT 2013
```

Review TADDM.log for the latest activity, for ease of use include a date in the grep to see only the latest data:

```
grep agent TADDM.log | grep "2013-08-0"
```

```
2013-08-08 09:13:06,251 INFO - CTJOT0402I Topology builder agent class
com.ibm.cdb.topomgr.topobuilder.agents.PersobjCleanupAgent is starting.
2013-08-08 20:46:33,751 INFO - CTJOT0403I Topology builder agent class
com.ibm.cdb.topomgr.topobuilder.agents.PersobjCleanupAgent is stopping.
2013-08-08 20:46:33,759 INFO - CTJOT0402I Topology builder agent class
com.ibm.cdb.topomgr.topobuilder.agents.ObjectsWithoutAliasesCleanupAgent is starting.
2013-08-08 20:46:33,762 INFO - CTJOT0403I Topology builder agent class
com.ibm.cdb.topomgr.topobuilder.agents.ObjectsWithoutAliasesCleanupAgent is stopping.
2013-08-08 20:46:33,767 INFO - CTJOT0402I Topology builder agent class
com.ibm.cdb.topomgr.topobuilder.agents.AliasesCleanupAgent is starting.
```

As you can see above, the PersobjCleanupAgent ran for 11+ hours and the AliasesCleanupAgent has been running for almost 12 hours. If database statistics are up to date, then a PMR should be opened to investigate this as long running topology builder agents will cause the Primary Storage Server to be marked busy and unusable for data storage (assuming a secondary storage server) and this can cause performance issues since one server is unable to participate in sensor storage.

Conclusions

As you have now read, TADDM performance tuning can be a complex task as there are so many variables to account for. Proper hardware and database tuning are key elements of a well performing system. There are various tuning parameters, but remember they interact. For example, if you increase discover worker thread counts; remember you may need to increase memory to support the additional threads. If you increase memory settings, do not exceed available memory on the machine as that can cause memory to be paged to disk which really slows down applications. Also if you increase discovery worker thread counts; you may need to increase pool sizes otherwise performance can get worse simply because we are waiting for sessions to the additional targets and timing out.

And if you have done everything listed in this document but still do not have the performance you desire, then it is time to scale horizontally by adding more storage servers and/or discovery servers.

Appendix

Tools for monitoring performance on AIX:

Use the link below to download the perfpmr.sh script. This can be used to collect performance statistics while you are having the system performance issue. If the system is a VIO client then please run the perfpmr script on both the VIO client and server at the same time.

You can download the perfpmr script from the following ftp site; choose the one appropriate for your OS level.

ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr/

Database queries for counting resources;

The following database queries can provide some data that is helpful to understand the database size and content:

- Count of how many computer system objects are in the database;

"select count(*) from compsys"

- Computer systems that are specifically CS class (versus routers, bridges, and other devices -- this is a subset of the above):

"select count(*) from compsys where jdoclassx like '%ComputerSystem%'"

- Total CI count for sizing purposes:

"select count(*) * 1.2 from persobj where classname_x NOT IN ('com.collation.platform.model.topology.relation.InstalledOn','com.collation.platform.model.topology.net.BindAddress','com.collation.platform.model.discovery.scope.IpAddressScope','com.collation.platform.model.topology.relation.BindsTo','com.collation.platform.model.topology.relation.BindsAsPrimary','com.collation.platform.model.topology.relation.Uses','com.collation.platform.model.topology.relation.Contains','com.collation.platform.model.topology.relation.AccessedVia','com.collation.platform.model.topology.relation.Performs','com.collation.platform.model.topology.relation.Exports','com.collation.platform.model.topology.relation.MemberOf')"

- CI count by class – a count of all database objects by model object type;

"select classname_x, count(*) as count from persobj group by classname_x order by count desc"

- Distinct sensor (name plus IP address) count in saved discovery runs;

"select count(distinct(EVENT_ATTRIBUTE)) from discevnt"

- All sensor count in saved discovery run

"select count(*) from discevnt"

- Runids available for above counts

"select distinct(EVENT_RUNID) from discevnt order by event_runid"

TADDM performance Wiki

The TADDM Wiki has a performance section that contains additional documentation that may be helpful:

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Application%20Dependency%20Discovery%20Manager/page/Performance%20and%20Tuning

Database Performance documentation

We recommend using the IBM Performance Analysis Suite to assist with database analysis. This tool can be downloaded at this link:

https://www.ibm.com/developerworks/community/groups/community/perfanalyst

Refer to the usage guide for details on how to use it. This tool can analyze both DB2 and Oracle data inputs and point out potential issues.

Outside of this tool, the following data points are also useful for performance analysis:

Oracle

If you suspect problems with the Oracle database performance and have ruled out network, IO and driver issues, you can also do the following:

- Have the DBA gather Oracle AWR reports from the time frame covering any performance issues and review the results.
- Review the Oracle alert log for problems.

- Run the following query and confirm that last_analyzed is recent to ensure statistics are up to date. Usually this date should be within a week if there are many rows in the table, and sooner if there have been a lot of data changes;

```
select table_owner, table_name, index_name, num_rows, UNIQUENESS, last_analyzed from
dba_indexes where table_owner='primary schema owner in caps' order by  table_name,
index_name
```

Be sure to change 'primary schema owner in caps' to the com.collation.db.user from the collation.properties file. Use all upper case values.

Send the above information to IBM Support if a PMR is opened for the performance problem. Include TADDM DEBUG logs from the problem time frame from all storage servers and any discovery and anchor servers involved in the issue. Include CPU, memory and IO statistics from all servers as well, including the database.

DB2

If you suspect problems with the DB2 database performance and have ruled out network and IO issues, the following data points are needed to open a PMR with TADDM support in addition to the TADDM DEBUG logs from all storage servers and any discovery/anchor servers. Include CPU, memory and IO statistics from all servers as well, including the database.

Run the following from a DB2 command window. Replace <dbname> with the name of your database.

1. db2look -d <dbname> -e -o <dbname>.ddl -a -l -f
2. db2 get dbm cfg > dbmcfg.out
3. db2 get db cfg for <dbname> > <dbname>-dbcfg.out
4. connect to <dbname>
5. db2 -tvf table.sql > <dbname>-table.out
   where table.sql contains;

```
SELECT SUBSTR(CREATOR,1,10) AS CREATOR,
    SUBSTR(NAME,1,30) AS NAME, STATS_TIME, type, card, npages, tbspace, pctfree, volatile
FROM SYSIBM.SYStables
where type = 'T'
ORDER BY CREATOR, NAME
with ur
    ;
```

6. db2 -tvf buffer.sql > <dbname>-buffer.out
   where buffer.sql contains;

```
select substr(a.tbspace,1,20) as tbspace, a.datatype,
    substr(b.bpname,1,20) as bpname, b.npages,
    a.pagesize, b.pagesize
```

```
from sysibm.sysbufferpools b
full join sysibm.systablespaces a
  on a.bufferpoolid = b.bufferpoolid
order by 2, a.pagesize
;

select substr(tbspace,1,20) as tbspace, count(*)
from sysibm.systables
where type = 'T'
group by tbspace
;
```

7. db2 -tf index.sql > <dbname>-index.out
   where index.sql contains;

```
select SUBSTR(TBNAME,1,40), SUBSTR(TBCREATOR,1,10),
substr(name,1,30), SUBSTR(CREATOR,1,8),substr(colnames,1,60), firstkeycard, fullkeycard,
sequential_pages, density, iid, uniquerule, stats_time, colnames
from sysibm.sysindexes a
ORDER BY tbcreator, TBNAME, NAME
;
```

8. db2 -tf updmon.sql

where updmon.sql contains;

```
UPDATE MONITOR SWITCHES USING BUFFERPOOL ON ;
UPDATE MONITOR SWITCHES USING LOCK       ON ;
UPDATE MONITOR SWITCHES USING SORT       ON ;
UPDATE MONITOR SWITCHES USING STATEMENT  ON ;
UPDATE MONITOR SWITCHES USING TABLE      ON ;
UPDATE MONITOR SWITCHES USING UOW        ON ;
UPDATE MONITOR SWITCHES USING TIMESTAMP  ON ;
RESET MONITOR ALL
```

9. after this, run db2 get monitor switches
10. they should all say "ON"
11. run the process you are having performance issues with
12. db2 get snapshot for all on <dbname> > <dbname>-dbsnap.out
13. this must be run from the same command window as step 8
14. run snapshots every 5 minutes during the problem interval using a different time stamped output file each time.

Send IBM Support the following files:

1. <dbname>.ddl
2. dbmcfg.out

3. <dbname>-dbcfg.out
4. <dbname>-table.out
5. <dbname>-buffer.out
6. <dbname>-dbsnap.out*
7. <dbname>-index.out