Teamcenter Oracle Database Maintenance Best Practices White Paper

SIEMENS

White Paper

A Technical White paper describing Teamcenter's best practices for use in an Oracle DB environment.

How well the database server instance is tuned, as well as how memory, disk, and operating system components are configured has a direct impact on the Database Tier's performance. Monitoring and tuning of the database along with vendor configuration guidelines (best practices) are captured in the Teamcenter Deployment Guide. Oracle includes a number of built-in performance reports accessible via Oracle Enterprise Manager.

SIEMENS

Software Copyright and Trademark Notices

Note: The pages of this document are numbered consecutively 1…N without the usual Roman numeral numbering of the front matter. This makes the document's page numbers consistent with the numbers displayed by the Adobe® Acrobat® viewer and simplifies printing of page ranges from Acrobat.

Printed in the United States of America. 6/22/18

Disclaimer

This document is intended to provide information on Performance recommendations for Teamcenter database and application support work done for Teamcenter databases. Siemens PLM Software is providing this information as is, without warranty of any kind. **SIEMENS PLM SOFTWARE hereby disclaims and assumes no responsibility or liability for any results that occur due to the use of the information contained in this document**.

DOCUMENT HISTORY

Version	*Date*	*Author*	*Description*
1.0	03/06/2018	Sachin Vaidya	Initial version for release
1.5	06/07/2018	Tim Goerdt	Review and Update
2.0	6/22/2018	Tim Goerdt	Review and Update

CONTRIBUTORS

Sachin Vaidya	Infrastructure Architect, Systems Architecture and Performance Team, Americas Technology Office, Siemens PLM
Tim Goerdt	Principal Infrastructure Architect , Systems Architecture and Performance Team, Americas Technology Office, Siemens PLM

REVIEWERS

Scott Schweyher	Manager, Infrastructure Architect, Systems Architecture and Performance Team, Americas Technology Office, Siemens PLM
Tim Goerdt	Principal Infrastructure Architect , Systems Architecture and Performance Team, Americas Technology Office, Siemens PLM

Table of Contents

1 Introduction

This document provides information on Teamcenter Oracle Database Maintenance Best Practices. These practices should be followed along with regular maintenance of the Oracle database such as space and alert log monitoring, to improve and maintain performance of the Teamcenter database.

2 Assumptions

2.1 INFODBA schema

The Teamcenter schema is assumed to have been named as INFODBA which is the default setting used by TC schema Installation scripts. It is possible that this name might have been changed during initial installation and you may have different Teamcenter schema name. Please make schema name changes in the commands and scripts below to reflect TC schema name at your site.

2.2 Infrastructure knowledge

The reader has a reasonable understanding of the factors that affect the performance of the database and is familiar with terms defined later in this document.

3 Best Practices

3.1 Use dedicated Physical server box for running the Teamcenter database

The Teamcenter application consumes server resources e.g. CPU, memory and disk I/O extensively and thus it is recommended that Teamcenter Oracle database be set up on its own physical server and that server should not be used to set up or run any other databases or applications. The Teamcenter Oracle database should not be run on a virtual machine of any kind as the demands for the resources made by the TC application on the database cannot be met efficiently due to changing load conditions. The Teamcenter database should not be used to run or install schemas of other applications as TC Oracle database needs to be tuned to efficiently run the Teamcenter application and tuning it to run other applications may conflict with TC applications operations. The CPU utilization on the database server should be kept below 80%. Above that level, the response times of the application degrade significantly. The memory utilization by SGA and PGA combined should not exceed 80% of the total memory of the server. At least 10% of total memory should be left for consumption by the Operating system and other Oracle processes such as listeners. In no situation, should the swapping of memory be allowed on the Oracle database. Check swap space utilization using vmstat and swap –l commands and set swap usage parameters such as swapiness on Linux to 10 or lower value for the database server. The huge pages should be set up for managing memory on Linux platforms for the database SGA and PGA. The huge pages need to be set up in Kernel and server needs to be rebooted before starting the database instance on the server so SGA and PGA can grab huge pages from the

memory map to efficiently use memory pages. The I/O storage subsystem should be set up to protect the data from disk failures and to provide throughput of less than 5 ms/block for the Oracle read and write operations.

The network factor that most affects database performance for Teamcenter is *latency*. For many operations, Teamcenter makes multiple queries to the database server to make the user experience more interactive with more real-time feedback.
Each query and result must traverse the distance between client and server. If 100 queries are made across a 200ms latency WAN network link, then the overall operation requires 20 seconds to complete. That same operation on a 1 ms LAN requires only 1/10th of a second, which is imperceptible to the user. The Performance of the TC application is noticeably slower if the network latency exceeds 5–10 ms between tcserver process and the database server.

Ref: TC 11.2 Deployment Guide – section 5.1

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-memory-tunables

https://docs.oracle.com/cd/E11882_01/install.112/e41961/memry.htm#CWLIN382

3.2 Collect Teamcenter schema statistics on daily basis

The Teamcenter schema is an active schema and it updates several tables and indexes continuously during normal operation of the application. This results in significant amount of data changes caused by DML statements issued by the application. In order to generate efficient execution plans for queries issued by the application, the Oracle query optimizer needs the latest and current statistics that are 100% accurate. To generate accurate statistics, run following command while connected to the database as sysdba.

```
SQL > exec dbms_stats.gather_schema_stats(ownname =>'INFODBA', estimate_percent =>
100, method_opt => 'FOR ALL COLUMNS SIZE AUTO', degree=>DBMS_STATS.AUTO_DEGREE,
cascade=>true, no_invalidate=>FALSE);
```

This command needs to be set up to run on a daily basis during time when application activity is known to be minimal in the day (off hours for the users). During this activity, the data from all tables owned by INFODBA schema is read by the server process that is running the statistics command and it uses parallel query servers with default degree or number to run full scan of the data to collect statistics. The time taken for collecting statistics depends upon size of the INFODBA schema, buffer cache or size of the SGA, available CPU to run the activity and I/O throughput of the I/O subsystem.

The statistics collection job needs to be set up in either the crontab of oracle user on UNIX or Linux or in the Windows scheduler or in Oracle database scheduler. This job needs to be run on a daily basis to generate accurate statistics for the optimizer to generate efficient execution plans for all queries that are executed by the application.

Ref: TC 11.2 Deployment Guide – section 5.2

3.3 Verifying Teamcenter indexes

To verify whether all the default Teamcenter indexes exist in the database, use the index_verifier utility. The verifier ensures that all indexes created during installation/upgrade, including custom indexes created using install utility are present in the database. Run index_verifier utility from the application server as below to generate an output file that shows missing indexes, if there are any.

```
$TC_BIN/index_verifier -u=username -p=password -g=group
-o=DRYRUN > index_verifier_dryrun.txt
```

If missing indexes are found, then add them back using index_verifier utility as given below

```
$TC_BIN/index_verifier -u=username -p=password -g=group -o=DO_IT
```

Do not create missing indexes reported by index_verifier using install utility because they are already Teamcenter POM data dictionary and index_verifier recreates them using known definition using DO_IT option.

New indexes, if they are deemed necessary can be added to the Teamcenter INFODBA schema as below.

Install –add_index or –add_func_index infodba <password> dba index_name unique_option class attr1 attr2…

Do not add any index using sqlplus directly in the INFODBA schema as it will not be registered in the POM data dictionary and may get dropped during upgrades or deployments.

3.4 Set up database parameters for performance efficiencies

Following parameters should be set in Teamcenter Oracle database which are based on best practices to achieve performance efficiencies of the database in running queries and DML statements.

1. Commit_logging: Recommended value – BATCH – It controls how commits are grouped together to be written to Redo logs. Writing them in batch from log buffer to redo logs improves performance of commits but it can result in inconsistent database if a crash occurs. Set with caution only if commit waits are one of the top waits in the database.

2. Commit_wait: Recommended value – NOWAIT – it controls if the transaction waits for the commit to be written to redo logs before handing control back to the transaction. With NOWAIT setting transaction does not wait for commit to be written for redo logs and thus commit performance improves. It also introduces risk of committed data not being written to redo logs and causing corruption in case of a crash. Set with caution only if commit waits are one of the top waits in the database.

3. Compatible: Recommended value – your RDBMS version – set it to your 3 digit RDBMS version e.g. 11.2.0. This parameter specifies the release with which Oracle must maintain compatibility. Allows database to fully use features of the new release and to maintain backward compatibility.

4. Cursor_sharing: Recommended value – EXACT – Since TC engine generates same SQL statements with many different values for bind variables, setting cursor_sharing to EXACT reduces amount of work done by database during parsing of the statement and helps in reducing parsing time of the SQL statement.

5. Db_block_checksum: Recommended value – FALSE – Used to detect block corruption by Oracle background processes such as db writer and log writer. Setting it to TYPICAL causes 1 to 2 % CPU overhead but if resources are at high utilization then set it to FALSE.

6. Db_block_size: Recommended value – 8192 – standard value for hybrid databases who have OLTP and long transaction mix which TC has. Also matches standard OS block size on most UNIX and Linux operating systems making I/O efficient.

7. Db_cache_size: Recommended value – 0 – No need to set db_cache_size when SGA is set to manage buffer pool automatically using sga_target and sga_max_size parameters.

8. Db_file_multiblock_read_count: Recommended value – 0 – Setting it to 0 sets it to default value of underlying OS default read block count. Makes I/O efficient by translating one Oracle read call into one OS read call. This parameter is used only FTS in query which should be minimal for large tables in efficiently running queries. *In some cases setting it to 16 may be beneficial.

9. Db_keep_cache_size – Recommended value – (calculate the size) – This pool in SGA is used to pin tables into SGA for fast access. The TC deployment guide recommends certain tables to be pinned in SGA for fast data access. To pin these tables this memory is reserved and used by the SGA. It can also be used for pinning other small tables that may improve performance of the frequently executed queries.

10. Dml_locks – Recommended value – 1024 – Since TC database has large number of transactions happening parallel by many sessions, dml_locks ensures that sufficient DML locks are available for tables to be locked during transactions.

11. Filesystemio_options – Recommended value – SETALL – Asynchronous I/O is enabled for database that uses file systems for datafiles. Not required to be set for RAC databases that use ASM.

12. Java_pool_size – Recommended value – 0 – Set to 0 for SGA automatic management of java pool.

13. Large_pool_size – Recommended value – 0 - Set to 0 for SGA automatic management of large pool.

14. Log_buffer – Recommended value – 1MB – Used by log writer to write committed and uncommitted transaction data for redo operations. Circular buffer and it is overwritten as redo data is written to redo logs. If redo logs are set up on fastest available devices and have I/O rate that does not cause log sync waits then log_buffer is set up right. Per Oracle no benefit is gained by setting value greater than 1MB. Monitor REDO BUFFER ALLOCATION RETRIES statistic over a period of time and it should remain 0 or to a small number if log_buffer is sized correctly.

15. Log_checkpoint_interval – Recommended value – 0 – it is set up in number of redo blocks after which checkpoint occurs. By setting it to 0 the checkpoints occur only when logs are switched and thus avoids expensive checkpoints in middle of redo operations.

16. Log_checkpoint_timeout – Recommended value – 1800 – value is in seconds and used to force a checkpoint if none have occurred in last 30 minutes.

17. Memory_max_target and memory_target – used for automatic management of full memory used by db – SGA + PGA – Siemens recommends to use SGA and PGA to be managed automatically but separately using sga_target and pga_aggregate_target parameters.

18. Open_cursors – Recommended value – 1024 – Open cursors as required by each session. Not having cursors to run queries causes application connection to fail and it writes a message in the alert log. Setting high value for this parameter does not cause resources to be consumed but if this limit is reached then the query fails.

19. Optimizer_dynamic_sampling – Recommended value – 2 – Generates statistics if at least one table in SQL statement has no statistics. This is the default value for dynamic sampling behavior.

20. Optimizer_index_caching – Recommended value – 95 – influences behavior of the optimizer when it considers cost of using index in queries. Setting it close to 100 cause's optimizer to choose index more likely over full table scans. Teamcenter queries have been found to execute efficiently using available indexes with this setting.

21. Optimizer_index_cost_adj – Recommended value – 10 – this parameter adjusts cost of using index and makes index use more likely for optimizer's decision to use it in an execution plan of the query. Setting it to 10 has been found to cause most TC queries to use index when available.

22. Pga_aggregate_traget – Recommended value – depends on user activity – Set it sufficiently large so PGA hit percentage is 100 % meaning all user processes get sufficient memory to process queries.

23. Processes – Recommended value – depends on user activity – Set it high enough to accommodate all database user process generated by the application. If this limit on processes is reached then user connection fails to open a session and message is written in the alert log.

24. Query_rewrite_enabled – Recommended value – TRUE – Oracle rewrites query internally to execute it faster than original query without changing result or select clause. Efficient and default behavior in 11gR2 onwards and query execution is improved transparently without modifying code.

25. Query_rewrite_integrity – Recommended value – TRUSTED - Oracle allows rewrites using relationships that have been declared, but that are not enforced by Oracle. Makes use of relationships other than constraints in where clause, resulting in faster execution.

26. Recyclebin – Recommended value – ON – can be used to restore dropped table if FLASHBACK feature is enabled in the database before the drop.

27. Resource_manager_plan – Recommended value – Null or ‘’- It is not required unless resources consumed by one database are affecting other databases and thus the resource use needs to be limited. Usually in production, TC is set up as a single database on a server that does not host any other applications so disabling resource manager by setting null plan removes limits on usage of CPU, memory and I/O and lets the database use full available resources.

28. Session_cached_cursors – Recommended value – 1024 – this parameter allows a session to retain closed cursors by queries run in past and thus helps in reducing parsing of queries who match existing used cursors. Helps in reducing parsing and thus time taken by SQL to execute. It takes memory from PGA so setting high value can consume PGA with large number of sessions opened by Teamcenter.

29. Sga_max_size and sga_target – Recommended value – depends on need of memory by the database according to use by the application. Setting these both parameters enable Oracle to manage SGA automatically for database cache and shared pool allocations. Both parameters need to be set to same value unless there is a process in place to change SGA size for specific operations. If values are not matching for these parameters then memory allocated remains at max size but memory used by database remains at target size and thus memory resources are wasted. The memory allocated to the database – sga + pga – should not exceed 75 % of the total available physical memory as memory outside SGA and PGA is used by operating system and database background processes that work outside of SGA e.g. listener, DLM etc.

30. Shared_pool_reserved_size – Recommended value – depends on need of the database to store large packages and queries in shared pool. Usually it is not necessary to reserve memory for the shared pool if SGA is managed automatically. If the errors reported in alert log for loading of parsed packages or procedures for not having sufficiently large shared pool then this parameter may need to be set but usually it is a rare event in automatically managed SGA which resizes shared pool as per database need from available allocation. The SGA may need to be resized to avoid shared pool and cache issues that affect the performance of the database.

31. Shared_pool_size – Recommended value – 0 – Setting it to 0 allows SGA to be managed automatically by SGA_TARGET parameter.

32. Statistics_level – Recommended value – TYPICAL – default value that allows database to collect required statistics to measure performance metrics of the database.

33. Streams_pool_size – Recommended value – 0 – allows streams pool to be managed by automatics SGA management when required.

34. Timed_statistics – Recommended value – TRUE – default value that allows timing statistics to be collected for various events in the database for waits and performance metrics.

35. Undo_management – Recommended value – AUTO – default value that allows automatic management of the rollback segments and rollback activity of the transactions.

3.5 Coalesce indexes to reduce space consumption and improve performance

In order to reduce the space used by indexes and to reduce fragmentation of indexes, coalesce indexes on regular basis. Coalescing indexes moves keys in indexes to leaf blocks that balance the indexes and releases free blocks for new data in the table. This reduces logical I/O required to access table data via indexes and thus improves performance of the queries. The coalescing of indexes is an online operation and does not cause interruption for the application processing. It should be run during time of low application activity to reduce impact of coalescing I/O activity. Use the script provided in Appendix A to coalesce indexes.

Ref: Oracle support note - Index Rebuild, the Need vs the Implications (Doc ID 989093.1)

https://asktom.oracle.com/pls/apex/asktom.search?tag=rebuilding-indexes-200202

3.6 Drop temporary tables that are no longer used

The Teamcenter application creates thousands of temporary tables for application use. These tables accumulate in database as empty tables and can cause database dictionary to grow significantly with definitions of temporary tables. These temporary tables need to be cleaned out using Teamcenter install utility on regular interval basis such as weekly or monthly depending on volume of user activity and volume of temporary tables generated.

Run the SQL statement below to find the count of temporary tables in your TC database.

```
select trunc(created) CREATED ,count(*) TEMP_TABLES_COUNT
from dba_objects o,dba_tables t
where  o.object_type='TABLE'
and o.object_name=t.table_name
and t.temporary='Y'
and t.owner=(select owner from dba_segments where segment_name='PPOM_USER')
group by trunc(created)
order by 1;
```

The command below when ran from TC application server, lists all temporary tables in the TC database

$TC_BIN/ install -temp_table -u=infodba -p=$INFOPWD -g=dba list

The command below when ran from TC application server, drops temporary tables in the TC database that are created older than certain date.

$TC_BIN/ install -temp_table -u=infodba -p=$INFOPWD -g=dba drop older_than_date=2018/01/06 00:00:00

Any temporary table that is dropped or purged will get recreated if it is needed for application use so dropping temporary tables older than a month usually helps to keep temporary table count low enough to not affect database performance.

Ref: Teamcenter 11.2 - Utilities Reference Manual

3.7 Detect slow running SQL statements from application server and tune them in Teamcenter database

The TC_SLOW_SQL environment variable can be set in the Teamcenter application server as given below and then SQL statements running slower in the TC database than set threshold are reported in the syslog on the application server.

TC_SLOW_SQL=10 --- Reports SQL statements that ran over 10 seconds in syslog
TC_SLOW_SQL=-1 --- Turns off slow SQL reporting. In production database,

TC_SLOW_SQL should always be set to -1 to reduce I/O impact on database and application server as it affects all sessions generated by the application server. It should only be used to test certain use cases and to capture slow running SQL statements from the run of use cases.

The TC_SQL_DEBUG environment variable can be set in Teamcenter application server to capture detailed run times and data about all SQL statements issued by each session from that application server in their respective syslogs. It is highly resource intensive and should be used with caution in a production environment. Usually both of the above variables are used for use case SQL statement analysis for slow running SQL statements. The TC_KEEP_SYSTEM_LOG=Y should also be set along with these variables to capture required data for the runtime of the use case.

The SQL Statements found in syslogs that are considered to be candidate for tuning need to be found in database AWR reports so their SQL ID could be found and then

further actions regarding execution plans, SQL profiles and SQL baselines can be taken after thorough performance analysis of the SQL statements.

Ref: TC 11.2 – Deployment guide – section 5.1.5

4 Appendix A – Coalesce index script

The following process can be used to create a script to coalesce the indexes for the Teamcenter database.

Step 1: Using sqlplus connect to the database as system.

Step 2: To create the sql script to coalesce all INFODBA indexes copy following commands in a sql script called bc.sql and run it. It will create script to be run in next step.

```
set echo off feed off serveroutput off term off head off lines 120 pages 0
spool coalesce_indexes.sql
select 'set echo on feed on time on' from dual;
select 'spool coalesce_indexes.log' from dual;
set serveroutput on
BEGIN
FOR INDEx_RECORD IN (select owner||'.'||object_name as obj
from dba_objects
where object_type = 'INDEX' and
owner =(select owner from dba_segments where segment_name='PPOM_USER')
order by created desc
)
LOOP
dbms_output.put_line ('ALTER INDEX '||INDEx_RECORD.obj||' COALESCE;');
END LOOP;
END;
/
select 'spool off' from dual;
select 'exit' from dual;
spool off
```

Step 3: Now run coalesce_indexes.sql script created by running above bc.sql script.

Step 4 : Collect full statistics on the INFODBA schema after the indexes are coalesced.

```
SQL>exec dbms_stats.gather_schema_stats(ownname =>'INFODBA', estimate_percent => 100,
method_opt => 'FOR ALL COLUMNS SIZE AUTO', degree=>DBMS_STATS.AUTO_DEGREE,
cascade=>true, no_invalidate=>FALSE);
```