Deltek

Deltek Costpoint® 8.0

Performance Optimization

September 1, 2020

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This edition published September 2020.

Contents

Overview

Welcome to the Deltek Costpoint Version 8.0 Performance Optimization document. This document contains a summary of the following:

- Costpoint Performance Tips and Tricks: This section is more theoretical and will help you better understand the concepts, technologies, and techniques that you can take advantage of to either troubleshoot or optimize your deployment.
- Deltek Performance Lab: This section provides specific information on the configuration of the Deltek performance lab, which is used for tuning Costpoint applications. Although the lab does not employ the top of the line hardware available on the market today, Deltek observed same or better performance in the Deltek lab as observed in environments of large clients with whom Deltek compared results.

  Deltek recommends that you consider Oracle configuration used in Deltek's lab and described in this document as a starting point when optimizing your Costpoint deployments on the Oracle platform. Of course each company's deployment is unique in terms of its hardware, data volume, data distribution, and the Costpoint business features used. Therefore, a single approach would not work for all Costpoint deployment optimization. Costpoint system and database administrators need to adjust provided configuration recommendations as necessary to find the configuration that works best for the specific attributes of your firm's deployment.
- Optimized Applications: This section lists the Costpoint applications that have been optimized in 6.1 for processing **All** projects, as well as to support the Oracle parallel execution feature.

Target Audience

This document is designed for database administrators, system administrators, and other technical staff responsible for the configuration and tuning of the Oracle database that supports the Deltek Costpoint application suite.

Importance of the Database Tier in Performance Tuning

Although many factors can impact Costpoint's performance—such as the hardware used on the client workstation, network performance, or the robustness of the hardware used on the middle tier for web deployment—at least 80 percent of performance challenges in deployments are associated with the database tier.

The purpose of this document is to provide Costpoint and database administrators with information that, in combination with Oracle performance guides, will help you to understand the performance characteristics of the Costpoint database tier. This improved understanding should lead to better tuning and performance of your Costpoint deployment.

Costpoint Performance Tips and Tricks

Oracle Optimizer

The component that affects the performance of your Oracle database the most is the Oracle optimizer. The optimizer is a key part of any relational database engine and is responsible for figuring out how to execute every SQL statement sent to the Oracle database. There are a number of mathematical algorithms built into the Oracle engine that can be used to sort and group data, join tables, and move data from one table to another. Each algorithm works well under certain circumstances and performs poorly in other cases. The optimizer is responsible for picking the most appropriate algorithm. If an inappropriate algorithm is selected, an SQL statement that otherwise takes a few seconds can literally take many hours to complete, regardless of how good your hardware is.

Rule-Based Optimizer

As the Oracle database engine itself evolved and changed over the years, so did the optimizer. In earlier versions of its database, Oracle used a rule-based optimizer, which determined how to execute a given SQL statement (order of table joins, join methods, usage of indices, and so on) based on the fixed number of rules applied to the text of the SQL statement. In other words, the actual data in the database had no impact on Oracle's decision on how to execute a given SQL statement.

Costpoint was originally written for this rule-based optimizer. Developers wrote and tested their SQL statements to make sure that they performed well on the volumes of data considered representative of Deltek customers. Of course, each customer is different and neither Deltek nor Oracle could guarantee the optimal performance based on a given data volume. In spite of its disadvantages, the rule-based optimizer had one important characteristic—100% predictability. If Deltek engineers found a way to write an SQL statement that worked well for a given setup with large volumes of data, then Deltek engineers and customers could be certain that they would see similar performance in all comparable deployments. At the same time, performance for smaller data volumes could be less than optimal.

Cost-Based Optimizer

To address this issue and provide optimal performance on all data volumes, Oracle switched to a cost-based optimizer, which makes decisions based on the current volume of data rather than just the text of the SQL statement.

Although a cost-based optimizer was a superior concept compared to the rule-based approach, database vendors quickly recognized that finding the right way to execute a SQL statement is not an easy task. The cost of incorrect optimizer decisions can translate into many hours in executing a single SQL statement. To make better decisions, an optimizer needs to consider more and more information. For example, in Oracle 9i the cost-based optimizer is making a decision entirely based on the cost of IO associated with a given execution plan. In Oracle 10g, the optimizer is considering both IO and CPU cost. It also collects and evaluates run-time performance metrics from your infrastructure, such as the speed of IO. As the optimizer becomes more complex and intelligent, it is becoming difficult to predict the ideal optimizer decisions in a given environment. If you have run into a situation in which your system was working well and then suddenly—without any substantial data volume or infrastructure changes—a particular application started to take significantly longer to execute, there is a good chance that Oracle picked a different execution plan for one of the statements.

Collecting TKProf Traces and AWR Reports

To alleviate the issue of unpredictability, Oracle developed tools in Oracle 11g to help database administrators record, monitor, and approve any changes to the execution plan. In the absence of such

tools, there is a relatively simple approach that a DBA can use on any version of Oracle to help monitor and troubleshoot optimizer-related issues.

The key is to determine the set of performance-critical applications and periodically collect TKProf traces on them, together with the AWR (Automatic Workload Repository) reports.

> **Attention:** Costpoint makes it very easy to generate a TKProf trace for an application. See the Generate an Oracle Trace for Specific Functions section for details.

Setting a Baseline

If your Costpoint system is currently running well, make sure that you collect traces and save these traces as a baseline.

If You Have a Performance Issue

Anytime you have a performance issue with an application, collect an AWR report and a trace for the application and compare it with the good AWR/trace from your baseline. Comparing AWR reports will help you discover if performance degraded due to other applications being executed at the same time and colliding with the Costpoint application in question.

If this is not the cause, then comparing traces will help you find out which SQL statements slowed down and if their execution plans have changed or not.

- If the execution plan did not change but the same statement is taking longer, TKProf trace will help you verify whether the statement is processing the same volume of data or the volume significantly increased.
- If neither the execution plan nor the data volume processed changed, then the trace will show you whether the IO or CPU portion of the execution slowed down. This may give you an important insight. For example, you may recall that the storage system was recently reconfigured and determine that was the cause of the change in performance.
- If the execution plan for a statement has changed compared with your baseline and performance has degraded, you can try to make Oracle go back to the original plan. A good starting point is to make sure that you have updated statistics on all Costpoint base tables and at the same time do not have any statistics on any Costpoint temporary tables (Z_ , X_).
- The Deltek Performance Lab section contains a script on how to update statistics for Costpoint. Oracle documentation contains information on how to manually set statistics to manipulate Oracle execution plan. If you set statistics manually, you must use great care in doing so.

> **Note:** If you need to report your case to Deltek technical support, your ability to provide before and after AWR reports and TKProf traces will greatly improve Deltek's ability to understand the problem in your environment and help with the resolution.

Update an Application by Including Oracle Hints

In some cases, when Deltek engineers see that Oracle has a tendency to overlook a more efficient execution plan for a given statement, they can update an application to include an Oracle hint that forces Oracle to use a particular technique when executing a statement. This approach is used carefully as its results depend on specific settings. For example, a good execution plan when computing for **All** projects will not necessarily result in optimal performance when computing for **One** project. Typically, use of hints is limited to **All** compute/print options.

Be sure to review the Deltek Performance Lab section for specific parameters affecting optimizer behavior and to learn how they are set in the Deltek lab. Keep in mind that the Oracle optimizer in 10g considers many different factors when choosing an execution plan. For example, a low setting for PGA memory can easily discourage Oracle from using Hash methods for grouping and joining tables which will likely have a substantial impact on Costpoint performance if you are running a large database. Hashing techniques (such as Hash Group By) are new in Oracle 10g and work particularly well for grouping and joining large tables.

Should I Process for All Projects or for a Range?

Processing large volumes of data, or the majority of data in the table, typically requires different execution plans to be used for many SQL statements in the application compared to processing for **One** project (or a small range). Also, the cost of inefficient execution plans on large volumes of data usually translates into many minutes or hours of extra processing time.

Deltek has specifically optimized a set of Costpoint applications when they are used with the **All** option on the parameter screen. After you install the optimized version of these applications, make sure that if you intend to process **All** projects (for example, as part of your closing batch) or a large subset of projects, that you run these applications for **All** projects rather than executing an application several times for different ranges. This step is critical to realize the gains from the performed optimization.

> **Attention:** See the Optimized Applications section for more information.

Is a Full Table Scan a Bad Idea?

Deltek engineering often receives the following feedback from customers' DBAs analyzing performance of their systems:

- "We suspect that your system is missing an index because it is doing a full table scan on table XYZ. Please, add an index to avoid a full table scan."
- "We see that your system is generating high IO due to a full table scan performed on table XYZ. Full table scans are inefficient and can be avoided if you add an appropriate index".

These comments might have been accurate when analyzing earlier versions of Oracle with older, slow hard drives. Currently, the situation is often the opposite of this, especially when processing large volumes of data such as processing or printing for **All** projects.

In the majority of cases, use of table access techniques when a row is found through an index diminishes performance. It can take hours to process all rows in a table this way, while a single full table scan (for example, for a **Hash join** technique) would take a few seconds, even for a large table. The following analogy might help you to understand this point. If you are reading a book and intend to read only one small chapter, it makes sense to read the table of contents and scan to find the chapter so that you can jump directly to it. The time that you spent reading the table of contents is certainly worth the time you gained by jumping directly to the right page. On the other hand, if you intend to read the entire book or even most of it, you will start reading from the very beginning and will not spend time going back to read the table of contents after each page. The same logic applies when you run a process for **All projects** and need to read/update the entire table or majority of the rows—it is faster for Oracle to read the entire table once then to read all the rows and spend time on finding them through an index.

The full table scan is often the most efficient technique for processing large volumes of data. Deltek recommends that you do not create your own indices just for the sake of eliminating full scans.

Parallel Execution

The most powerful server in your entire IT infrastructure is most likely your database server, and it is likely to be a multi-CPU box. This single machine has to service, in parallel, many requests for executing SQL statements coming from different Costpoint applications/users. But what about your nightly or weekend closing batches, which you most likely schedule through the Costpoint process server? How well do the massive compute and print jobs that process millions of rows of data—and which often run as the only process overnight—take advantage of your hardware with multiple CPUs?

The likely answer is not so well or not at all. By default, an Oracle database engine is using only one CPU to process a single SQL statement. So if you run a Costpoint process (such as Compute Burden or Update PSR) that needs to execute an SQL statement to process data from one or several large tables, the application may be held up on this single statement for a while, waiting for the Oracle database to execute it using a single CPU while the rest of the available CPUs stay idle.

CPU Utilization: Seeking Granular Statistics

Do not be confused by looking at average CPU utilization statistics, which could show that CPUs are no more than 25% utilized. For example, what that means is that out of four CPUs in your server, one is 100% busy while the other three are idle.

Finding the Bottleneck

A single CPU may very well be the bottleneck in your process and you need to look at single CPU stats to see if this is the case. However, you should also consider the implication that a single CPU processing has on IO performance. IT and DBA administrators often look at the overall IO bus bandwidth and average statistics and conclude that their IO system is more than adequate for the Costpoint application in question. According to these average statistics, Costpoint applications hardly utilize the bandwidth available. This creates a logical paradox, as all IT hardware resources—CPU, IO, network, and so on—work at a fraction of their capacity and yet the application is processing slowly. How could that be possible? The software itself does not actually do anything other than asking hardware to execute commands. If the hardware is not 100 percent busy, then what can the software be doing?

The answer is that there is a hardware bottleneck somewhere. However, to find it, you need to look at more granular statistics.

Going back to the implications of single CPU processing on IO, the reality is that regardless of the overall IO bandwidth a single CPU can typically utilize only a fraction of this bandwidth. As a result, even though IO may be the bottleneck for executing a given SQL statement, you do not see it if you only analyze overall (average) IO statistics.

To address both CPU and IO limitations in processing single SQL statements, Oracle (Enterprise edition only) supports a feature called **Parallel SQL execution**. Although the actual technology is subject to a number of rules and requirements, it is easy to understand the idea and potential of this technology on a simple example.

Example of Parallel Execution Improving Performance

Assume that you execute the Compute Burden application for All projects and the application needs to update all rows in the PROJ_SUM table for a given period (for example, 500,000 rows), based on the data retrieved from another work table. By default, Oracle uses a single CPU that loops through 500K rows, updating them one row at a time.

Now assume that you have 16 CPU boxes and are using parallel execution. In this case, Oracle can logically split PROJ_SUM table into, say, 10 sections with 50,000 rows each, and have one CPU allocated to each section. In this scenario, not only are CPUs processing the rows concurrently but multiple CPUs

can take advantage of the overall IO bandwidth of your storage system. The same SQL statement can easily run several times faster.

Costpoint Job Server and Parallel Execution

In order to take advantage of this powerful Oracle feature, the Costpoint Job (Process) Server was enhanced to support queues designated for parallel execution. If you submit an application to such a queue, it runs with the parallel execution option enabled.

Parallel execution is not a solution for all performance issues. In fact, different applications (depending on the SQL statements used in them) react to parallel execution in different ways. An application that has SQL statements processing large volumes of data at once can benefit significantly from parallel execution. Examples include massive Update/Delete, Insert, or Select statements.

On the other hand, applications that have a loop and execute separate SQL statements to process small subsets of data typically would slow down if using the parallel execution option.

Deltek has specifically optimized a set of Costpoint applications (listed in the "Optimized Applications" section) to support and take advantage of this Oracle feature when they are executed through the Job Server using a queue designated for parallel execution. If you want to use parallel execution for an application other than those listed in the "Optimized Applications" section, Deltek recommends that you first test how well the particular application reacts to processing in this way.

Should My Company Use Partitioning? And if so, How?

Partitioning is another powerful feature of the Oracle database (Enterprise edition required) that can greatly improve performance of processing and reporting with Costpoint applications. Historically, partitioning was something that Deltek customers were performing themselves. The purpose of this section is to explain out-of-the-box partitioning as well as to provide guidance on additional partitioning options that you can benefit from using.

Partitioning and Performance

Partitioning is often used by Oracle DBAs to improve database maintainability and help with backup schedules. However, these aspects of partitioning are outside of the scope of this paper, which is focusing on performance. Also, the benefits of partitioning related to database management are typically much better understood among the Oracle DBA community.

How Does Partitioning Help with Performance?

The most typical answer is that partitioning makes database queries faster because the table is divided into a number of logical partitions and the Oracle database engine may need to scan only one partition to find the needed records, as opposed to scanning the entire table. This is certainly true in many cases, though not always, as required records may come from more than one partition. In that case, it might actually take longer to scan a partitioned table than a non-partitioned one. This brings us to an important point: how you divide the table into partitions is absolutely critical for improving performance. Whoever makes this decision needs to have some idea about the table usage patterns and SQL statements executed against the table to determine if a given partitioning model makes sense and would actually improve performance.

Local Indices on Each Partition

At the same time, the improvement on data query operations is often not the area that brings the most performance gains. Another important aspect of partitioning a table is ability to have local indices on each partition, rather than a global index on the entire table. You can use local indices instead of global indices

when you know that the majority of the SQL statements access data from only one partition. Not only do local indices work faster, due to their smaller size, but they are also faster to update when inserting or deleting data in the table. It is the improvement in the speed of Insert and Delete operations because of faster local index management which often results in the most noticeable performance gains.

Out-of-the-Box Partitioning and Costpoint Patch 1904

Patch 1904 is the first time that Deltek offers out-of-the-box partitioning for Costpoint. This patch is optional. Customers who do not have the Enterprise edition of Oracle cannot deploy this patch.

Although all Costpoint applications will continue to work with or without this patch being applied, Deltek strongly recommends that you take advantage of partitioning and deploy this patch if improving performance for the applications listed in the "Optimized Applications" section is important to your business.

The following tables are partitioned as part of patch 1904:

- POOL_SIE_SUPPORT
- PSR_FINAL_DATA
- PSR_HDR
- PROJ_SUM
- PROJ_BURD_SUM
- RPT_REVENUE_WKS
- RPT_PROJ_LAB_SUM
- RPT_REV_SUM
- RPT_PROJ_TMA

Range Partitioning

It is important to understand that Deltek is limited in its ability to provide out-of-the-box partitioning due to the nature of this Oracle feature. Although Oracle supports various partitioning options, the most common one is **range partitioning**. This option allows a DBA to define ranges of data values that determine what table records would fit into each partition. For example, we can say that all records with a **Fiscal Year** column value less than 2009 will be in partition A, while the rest of the records will be in partition B.

As you can see, partitioning rules are data driven and in many cases are unique to a given Costpoint deployment. For example, each customer can create its own fiscal years—such as 2009A and 2009B—making it impossible for Deltek to partition tables generically by fiscal year. Instead, Deltek identified a way for partitioning the tables that will work for different customers by using the **Period** and **SubPeriod** columns.

Larger Firms: Consider Including the Fiscal Year Column

While out-of-the-box partitioning should work well for a majority of customers, Deltek recommends that larger customers who need to get the most performance out of Costpoint applications should consider modifying the out-of-the-box partitioning model and include the **Fiscal Year** column as an additional partition parameter on top of the columns used by Deltek.

If you use this approach, **Fiscal Year** should be the first column in the range criteria. Fiscal year partitioning is especially beneficial if you store multiple years of data in the above tables.

Do Not Create Global Indices on Partitioned Tables

Make sure that you do not create global indices on partitioned tables.

If You Change the Partitioning Scheme

If you change the partitioning scheme to any way other than that provided by Deltek, it is your responsibility to maintain the scheme with future updates, patches, or hot fixes that involve the partitioned table.

Deltek Performance Lab

Environment

The following set of recommendations are tested on HP-Proliant DL 580 servers, each with four CPU quad-core processors, 32 GB of RAM, and an EVA 8100 SAN with two 4GB fiber channel connections. Each connection has a separate 4GB caching controller.

- The SAN drives are 15k rpm fiber channel drives configured in a 32 drive disk group using a logical RAID 5 configuration.
- The server operating system is Windows Server 2003 R2, SP2 Enterprise 64 bit version.
- The RDBMS is Oracle Enterprise 10g, version 10.2.0.4, 64 bit with the July 2008 CPU patch set applied.
- The application servers used are HP DL-380 servers with a single dual core processor, with 3.5 GB RAM and Windows Server 2003 SP2.

Oracle generates many types of cumulative statistics for the system, sessions, and individual SQL statements that are stored in the Automatic Workload Repository (AWR). The primary tools for monitoring these parameters and the system configuration are the RDA and AWR reports. The following features of Oracle 10g are currently utilized / required in Costpoint. Parameters related to optimizing these features are discussed in the narrative below.

For Costpoint, Deltek uses Automatic Shared Memory Management in Oracle 10g with 10 GB allocated to the SGA and 2 GB to the PGA. Extent Management is local and set to auto-allocate. Segment space management is set to automatic for all tablespaces. Datafiles are set to auto-extend automatically in size.

The following parameters are set as shown:

- SGA_MAX_SIZE = 10G
- SGA_TARGET = 10G
- PGA_AGGREGATE_TARGET = 2G
- STATISTICS_LEVEL = TYPICAL

Before making memory changes, check with your system administrator to make sure that the hardware has sufficient memory and that the operating system is configured to allow applications to make full use of the additional allocated memory.

Deltek strongly recommends that you use the Oracle AWR report—specifically, the memory tuning advisories section. This section gives you an idea about how much memory is needed for different pools in the SGA, including the buffer cache and the shared pool.

The PGA also comes with its own memory advisory section and another section that gives the number of the optimal, one pass, and multi-pass executions for the SQL statements. Another important statistic to consider is the Sorts (disk) that is also part of the AWR report.

A Full Table Scan (FTS) operation is one where Oracle will read all the blocks in a table up to the high water mark value. It will read a number of blocks in a single read operation that is determined by the Oracle parameter DB_FILE_MULTIBLOCK_READ_COUNT. The wait event associated with a FTS is db file scattered read.

In addition, for an Oracle 10g Costpoint database of 430 GB in size, the tablespaces and datafiles are configured as follows:

Tablespace Name	Number of Datafiles	Avg. Size of Datafile
Costpoint_Data	8	10 GB
Costpoint_Index	3	10 GB
Costpoint_Dyn	2	2 GB
Costpoint_Rptdata	8	20 GB
Costpoint_RptIndex	3	15 GB
Costpoint_XZData	5	20 GB
Costpoint_XZIndex	2	2 GB
CPMeta_Data	2	512 MB
CPMeta_Index	2	128 MB

The AWR report shows the read and write latency numbers for each tablespace and corresponding datafiles. A read or write latency below 7 ms is generally considered acceptable.

- All SQL statements are subject to analysis by the Cost Based Optimizer (CBO). The CBO's job is to find an execution mechanism that uses the fewest resources to achieve the required result for a given statement. The following parameters related to the CBO are set as follows:
- OPTIMIZER_MODE = ALL_ROWS
- OPTIMIZER_INDEX_CACHING = 0
- OPTIMIZER_INDEX_COST_ADJ = 100
- DB_FILE_MULTIBLOCK_READ_COUNT = 16
- CURSOR_SHARING = EXACT

Other parameters may be left at their default values for Oracle 10g. One of the key inputs for the CBO in determining the execution plan is optimizer statistics. The CBO determines which execution plan is most efficient by considering all available access paths and by factoring in information based on statistics for the schema objects (tables or indexes) accessed by the SQL statement. The query optimizer may also consider Optimizer Hints and any filter clauses within the SQL statement.

Following is a sample script (Oracle Stored Procedure) to calculate optimizer statistics for Costpoint on Oracle 10g that you should run on a regular basis:

```
CREATE OR REPLACE PROCEDURE ANALYZE_DELTEK
AUTHID DEFINER AS
BEGIN
FOR AA IN (SELECT USERNAME
    FROM ALL_USERS
    WHERE USERNAME IN ('CPMETA', 'CPADMIN','CPSYSTEM', 'DELTEK'))
LOOP
    FOR T IN (SELECT TABLE_NAME FROM ALL_TABLES WHERE OWNER = AA.USERNAME)
    LOOP
        IF SUBSTR(T.TABLE_NAME,1,1) = 'Z'
        OR
        SUBSTR(T.TABLE_NAME,1,1) = 'X'
```

```
        THEN DBMS_STATS.DELETE_TABLE_STATS(
           ownname    => AA.USERNAME,
           tabname    => T.TABLE_NAME);
        ELSE
           DBMS_STATS.GATHER_TABLE_STATS(
           ownname    => AA.USERNAME,
           tabname    => T.TABLE_NAME,
           estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
           method_opt       => 'FOR ALL COLUMNS SIZE AUTO',
           cascade    => true,
           degree     => 4); COMMIT;
        END IF;
        END LOOP;
    END LOOP;
END;
```

The key here is that you should delete statistics on table names starting with X and Z because these are work tables used within Costpoint.

Parallel Execution

To use parallelism, you must own the Oracle Enterprise Edition. In Costpoint, parallel execution is only enabled through the Costpoint Process Server. This makes the Costpoint Process Server jobs capable of the fastest performance in Costpoint. The degree of parallelism (DOP) is set to an Oracle-determined default value, rather than a specific value.

In order to limit the system resources consumed and thus the number of parallel execution servers created, we set the Oracle parameter PARALLEL_ADAPTIVE_MULTI_USER to TRUE and closely monitor and set the parameters PARALLEL_MAX_SERVERS and PARALLEL_MIN_SERVERS.

On our 16-CPU Core system this resulted in the following settings:

- PARALLEL_ADAPTIVE_MULTI_USER = TRUE
- PARALLEL_MAX_SERVERS = 225
- PARALLEL_MIN_SERVERS = 0
- PARALLEL_EXECUTION_MESSAGE_SIZE=8192

The other parameters related to parallelism are left at their default values in Oracle 10g. These parameters need to be evaluated during representative workloads submitted during the Process Server job runs. If the Process Server jobs are not run in a dedicated time window, these settings need to be reduced to leave some CPU resources free for non-Process Server workloads. Other methods of controlling resource allocation, such as Oracle.

> **Note:** The Resource Manager may also be used to control resource allocation but this method has not been tested by Deltek at this time.

The following chart contains the Oracle parameters discussed in this document and list the values used for these parameters in the Deltek Performance Lab during large database testing. This chart represents a starting point for large system configurations. You should adjust the values based on the feedback given by the Oracle AWR reports and on the size and use of your firm's Costpoint database. These are guidelines rather than absolute settings.

Parameter	Value	Comments
Optimizer_Mode	ALL_RO WS	The optimizer optimizes statements with the goal of best throughput and minimum use of resources.
Optimizer_Index_Caching	0	This encourages the use of FTS over Nested Loop Joins and IN List iterations.
Optimizer_Index_Cost_Adj	100	This causes the optimizer to assume that Index-based access path is equally expensive as an FTS.
Db_Block_Size	8192	This determines the size of each database block.
Timed_Statistics	TRUE	This is used to collect timed statistics.
Processes	300	This specifies the maximum number of user processes that can simultaneously connect to Oracle.
Sessions	350	This determines the concurrency in the system.
Sga_Max_Size	10G	This is the maximum size of the SGA for the lifetime of the instance.
Sga_Target	10G	This specifies the total size of all SGA pools and enables Automatic Memory Management.
Pga_Aggregate_Target	2G	This is the target PGA memory available to all server processes for SQL operations.
Statistics_Level	TYPICAL	This specifies the level of collection of database and OS statistics.
Parallel_Adaptive_Multi_User	TRUE	Setting this to TRUE enables the use of Oracle adaptive multi-user algorithm for parallel execution.

Parameter	Value	Comments
Parallel_Max_Servers	225	This specifies the maximum number of parallel execution servers for the instance.
Parallel_Min_Servers	0	This specifies the minimum number of parallel execution servers when the instance is started.
Parallel_Execution_Message_Size	8192	This specifies the size of messages for parallel execution.
Cursor_Sharing	EXACT	This determines what kind of SQL statements can share the same cursor.
Db_File_Multiblock_Read_Count	16	This impacts the user of FTS v/s index operations on the system.

Attention: For more information about using parallel execution, refer to the following documents:

- Chapter 25, “Using Parallel Execution,” Oracle 10g Data Warehousing Guide Release 2
- Chapter 18, “Partitioned Tables and Indexes,” Oracle 10g Concepts Release 2

Optimized Applications

The following Costpoint applications have been optimized for processing **All Projects** as well as to support Oracle parallel execution feature. They have also been tested with optional partitioning patch 1904.

For more information about each change, as well as other software fixes, see the descriptions in the following section.

Application ID	Application Name	Knowledge Base Article	Patch	Partitioning Patch
PJPALCST	Compute Burden	40937	1896	1904
PJPCOMPR	Compute Revenue	39664	1897	1904
PJPCRRPT	Create Report Tables	40939	1898	1904
PJPUPPSR	Update PSRs	40307		1904
PJPCOMPP	Compute + Print Pool Rates	40313	1899	
PJPBLDRA	Build Rate Application Table	40941	1900	
PJPBREAL	Redistribute Revenue	40942	1871,1901	
PJPMJALC	Compute Multi-Job Allocation	39637	1902	
PJPCRLNK	Create Links	39891	1903	
PJMSETNG	Project Settings	40934	1903	
BLPGBILL	Calc Bills	39588		
PJPCOGS	Compute COGS	39895		
BLPUNBIL	Create Unbilled Analysis Rpt	40927		
PJPPOSTR	Post Revenue	40928		
PJPINBRD	Compute Indirect Burden	39507		
PJRREVW	Print Revenue Worksheet	40933	1894 1911,1914	

Finding Costpoint Session Information in Oracle

In the cp613_sys_025 system jar file update, there is a feature that will now show both Costpoint application and user IDs using a given connection in the **CLIENT_IDENTIFIER** field of the V$SESSION view. Costpoint clients who use the Oracle parallel execution feature through the Costpoint Job Server will be able to see application and user ID information for each of the parallel threads spawned by the Oracle server.

Use the following query to get Costpoint session information in Oracle:

```
SELECT SESS.SID, SESS.SERIAL#, PRO.SPID, SESS.USERNAME, SESS.OSUSER,
SESS.MACHINE, SESS.PROGRAM, SESS.MODULE,
SESS.CLIENT_IDENTIFIER, SESS.OWNERID, SESS.STATUS
FROM V$SESSION SESS, V$PROCESS PRO
WHERE SESS.PADDR = PRO.ADDR
AND SESS.PROGRAM NOT LIKE 'ORACLE%';
```

The columns reported are:

- **SID**: User Session Identifier
- **Serial #**: User Session Serial #
- **SPID**: Operating System Process Identifier
- **Username**: Oracle User Name
- **Osuser**: Operating System Client User Name
- **Machine**: Operating System Machine Name
- **Program**: Operating System Program Name (used with Costpoint Client Server)
- **Module**: Currently executing Module (used with Costpoint Client Server)
- **Client_Identifier**: Can be one of the following (used only with Costpoint Web):
  - **USERID@appId**: If the user ID and app ID are available. This should be the case for any SQL executed within an already open application (for example, when long running processes and reports are executed).
  - **N_A@CSTPOINT**: If the connection was requested from system code where the user ID is not defined (for example, login process before a user is authenticated).
  - **USERID@CSTPOINT**: If the connection was requested from system code where the user ID is available for example, system code initializing the session before any application is open).
- **Ownerid**: The column contents are invalid if the value is 2147483644. Else, this column contains the identifier of the user who owns the migratable session. For operations using Parallel Slaves, interpret this value as a 4-byte value. The low-order 2 bytes of which represent the session ID, and the high-order bytes the instance ID of the query coordinator.
- **Status**: Status of a session. This can be one of the following values:
  - **Active**
  - **Inactive**
  - **Killed**
  - **Cached**

- **Sniped**

The user executing this query must have the **Select_Catalog_Role** privilege in Oracle. This query works for Oracle versions 9i, 10g, and 11g.

Generate an Oracle Trace for Specific Functions

One of the key logistical challenges in collecting application traces is often coordination between a business user running Costpoint process and a member of the IT/DBA team who needs to start and stop tracing for an application on the Oracle database. Often, the process may be scheduled as part of a larger batch to be run during the night and exact start/end time for the process are unpredictable. What makes this even more complicated is that a DBA has to properly identify what sessions on the Oracle server need to be traced, which can be tricky in an enterprise environment with many users and some processes using parallel execution.

Costpoint solves this problem by providing a way for a system administrator to specify upfront which Costpoint processes need to be traced. This allows Costpoint to communicate automatically to Oracle which sessions need to be traced and when. As a result, the need for a business user and a DBA to work with each other in real time as the process is running has been eliminated.

The cp613_sys_025.jar update allows you to enable an Oracle trace through the enterprise.properties file. If you are on system jar file version cp613_sys_024 or lower, please download and apply the latest system jar file for Costpoint from Deltek Customer Care.

To enable tracing:

1. Grant the user DELTEK rights to execute the Oracle DBMS_MONITOR package.
2. Open the enterprise.properties file on the WebLogic server (found in \Deltek\Costpoint\61\applications\enterprise\properties).
3. Add the following entry at the bottom of the enterprise.properties file:

   ```
   PRODCP.PJPALCST.enableOracleTrace=true
   ```

   Where

   - RODCP is the instance name.
   - JPALCST (Compute Burden) is an example of a function you want to trace.

   Note: If you want to trace more than one function, you must add one entry like the entry above for each function. For example, the following three entries would be for tracing the Compute Burden (PJPALCST), Compute Revenue (PJPCOMPR), and Print Revenue Worksheet (PJRREVW) functions:

   ```
   PRODCP.PJPALCST.enableOracleTrace=true
   PRODCP.PJPCOMPR.enableOracleTrace=true
   PRODCP.PJRREVW.enableOracleTrace=true
   ```

4. Restart the Costpoint Web service.
5. Execute the function(s) you want to trace in Costpoint.
6. When the function(s) run, a trace will be produced in the default udump directory for each session used by the function. The trace files will have the following naming format:

   instance name + '_ora_' + sessionid + userid + appId + date (in format YYYY_MM_DD).

7. Run TKPROF against the trace files generated, and send them to Costpoint Technical Support for analysis. If the issue is performance related, use the EXEELA sort option.

About Deltek

Better software means better projects. Deltek delivers software and information solutions that enable superior levels of project intelligence, management and collaboration. Our industry-focused expertise makes your projects successful and helps you achieve performance that maximizes productivity and revenue. www.deltek.com