CA IDMS Environmental Database Tuning

Dick Weiland

CA Technologies
Principal Software Engineer

CA IDMS environmental database tuning is concerned with those aspects of a Central Version (CV) environment that impact application performance relative to the actual accessing of the database. This document will cover the major types of bottlenecks related to the accessing of an application that can impede efficient processing and some of the factors that contribute to their creation. General guidelines, independent of the database design, for avoiding these situations will also be discussed. Although this material will not consider specific database design scenarios it is important to remember that no amount of environmental tuning will negate the effect of poor database or application design.

Identifying the Problem Environment

Although it may seem obvious, the first step is to isolate the specific characteristics of the CV environment experiencing the problem. Are the periods of poor performance during the processing of batch jobs or online tasks? Does the problem occur when there is a mix of both types of processing? It may be that the problems only occur at specific times of day which can have implications related to the volume of processing within the CV. In other words, it is necessary to isolate the specific mix of programs that seem to consistently produce problem performance and when those periods can be expected to occur.

CA IDMS provides numerous reports that can be used to identify the processing that is occurring within the CV. All of the reports mentioned below have some facility to limit the reported data by a desired time interval. If a site has CA IDMS Performance Monitor installed a good report with which to start the investigation is PMARPT02 (Task Summary). This report will list the executed tasks and the number of times each task was executed within a specified interval. If there were CA ADS tasks involved another valuable report would be PMARPT04 (Dialog Summary). PMARPT04 will allow you to further break down any CA ADS tasks to identify the dialogs that were executed. Details concerning the usage of CA Performance Monitor reports can be found in the CA IDMS Performance Monitor System Administration Guide.

Those sites without CA Performance Monitor may wish to use SREPORT12 (Task Summary) and SREPORT13 (Program Summary) to identify similar information. These reports typically require that TASK statistics are enabled via the STATISTICS parameter of the CV's sysgen SYSTEM statement. Details for running the various SREPORTs can be found in the CA IDMS Reports Guide.

The goal of using these reports is to identify those programs or dialogs that are having the greatest negative impact on the performance of the CV. You may discover that during the entire interval exhibiting poor performance there are one or two programs that are always executing which perform large amounts of database activity. In other cases it may be a situation where there is simply a significantly higher concurrent volume of all programs then during periods of acceptable performance.

JREPORT04 (Program Summary) can be run to provide initial database activity for the entire interval or just for those programs suspected of being the major cause of the performance problems. This report uses journal archive files for input and these files will not contain any information for retrieval-only programs if the CV's SYSTEM statement contains the parameter RETRIEVAL NOLOCK. If you suspect any retrieval-only programs as being a potential cause of the performance problem you may want to consider running the CV with the RETRIEVAL LOCK parameter for a day to generate this information. However this parameter may increase the potential for additional wait time or deadlocks to occur between these retrieval-only transactions and concurrent update processing.

Those programs identified as being a major factor in the performance problem should also be reviewed to determine the type of database access they are performing. The general types of processing that should be identified are as follows:

- Area sweeps
- Random access (CALC retrieval)
- Walking sets, especially non-VIA sets
- Index processing
- Retrieval-only or update processing

This knowledge will provide the basic information to determine the types of stresses that can be expected on buffer pools and potential conflicts between concurrent transactions that may lead to processing bottlenecks.

Once there is an understanding of the types of processing occurring during intervals of poor performance it is possible to focus on the three major types of contributors to this condition which are:

- I/O waits (specifically read waits)
- Journal activity
- DBKEY contention

I/O Wait Analysis

Perhaps the single most common negative impact to database performance is the I/O activity generated by an application and in particular the amount of time needed to wait for the I/O activity to complete. When I/O waits are a problem they can typically be attributed to the following reasons:

- Buffer contention

- Inappropriate use of PREFETCH
- Too many I/Os for the amount of data being processed

Buffer contention is the conflict that arises for the availability of individual pages within a buffer pool during concurrent usage of the pool by multiple transactions. Analysis to identify the level of buffer contention is best performed at the transaction level and is typically measured by using a ratio of PAGES-REQUESTED/PAGES-READ. This ratio is frequently referred to as the 'buffer utilization' ratio. The type of processing being performed by the transaction as determined by your earlier analysis is needed to establish a standard to identify when buffer contention may exist for the suspected transaction during the timeframe when poor performance has been experienced.

JREPORT04 can be used to obtain the average buffer utilization ratio for a transaction across the timeframe in question. It is also important to calculate this ratio for timeframes when performance is at acceptable levels for comparison purposes. Similar values between good timeframes and those experiencing problems may rule out buffer contention as the cause of the poor performance for the transactions being examined.

Identifying an unacceptable buffer utilization number requires you to analyze the number based on the transaction's processing characteristics. Within a CV environment each time that a DML command makes a request for a record on a page the PAGES-REQUESTED value is incremented. If that request results in the page being physically read because it was not in a buffer results in the PAGES-READ field to be incremented. The types of commands issued and the record types accessed will allow you to estimate what may be considered to be the acceptable buffer utilization for a transaction.

If a program issues only OBTAIN CALC commands it is performing random accesses and it can be assumed that each DML command will cause a physical read I/O to be performed. In this case it may be acceptable that the transaction's buffer utilization ratio will be around a value of 1.0. However if a program issues a OBTAIN CALC command followed by 5 OBTAIN NEXT WITHIN SET commands where the set's members are stored VIA the record type of the OBTAIN CALC command, it would be hoped that the set owner and all 5 members would be accessed with a single read I/O. In this case the ideal buffer utilization would be a value around 5.0. A value significantly less than 5.0 may indicate that buffer contention is occurring and by the time each OBTAIN NEXT command is issued another transaction has caused the desired database page to have been flushed from the CV's buffers.

In the above example a smaller buffer utilization value may also be a clue that the anticipated VIA cluster has been spread across multiple database pages due to overflow processing. We will discuss ways to identify and correct this particular problem a little later in this document.

To minimize buffer contention the number of buffer pools and the number of pages assigned to each buffer should be determined by the type of processing performed against the areas mapped to each buffer. Areas whose transaction processing is primarily random in nature may all be mapped to a single buffer containing a large number of pages. Online transaction processing generally falls into this

category. However areas whose processing includes frequent area or index sweeps might be mapped to buffer pools of their own with an appropriate number of pages. This category might include batch processing or large SQL queries. In some cases the number of pages within a buffer pool may be altered during the course of a day's processing using DCMT VARY BUFFER commands to accommodate varying processing requirements.

Another feature that can negatively affect buffer utilization is the inappropriate use of PREFETCH. PREFETCH causes CA IDMS to read multiple pages into the buffer with a single I/O command in anticipation of those pages being accessed by the transaction. If PREFETCH is enabled it will take control if the associated buffer pool contains 500 or more pages for non-area sweep processing or greater than 255 pages if an area sweep is being performed. If enough pages are available in the buffer PREFETCH will read up to 2 tracks of data in a CV environment with a single I/O command.

The impact of PREFETCH on a buffer is that it will use pages at a much faster rate than single page I/O operations. This can increase buffer contention and increase the number of page reads for concurrent transactions utilizing that buffer pool. In addition due to the significant amount of data being transferred the wait time for the I/O to complete is longer. Transactions whose processing does not fit the PREFETCH model may actually run slower. Therefore it is recommended that larger buffer pools that support transactions doing a high amount of random retrievals should have PREFETCH turned off. Applications that may benefit from PREFETCH process transactions will tend to sweep all or a portion of a database area. It is a good idea to map the database areas for these applications to their own buffer pool where PREFETCH can be enabled without a negative impact to other database processing. DCMT VARY BUFFER commands can also be used to enable and disable the use of PREFETCH within a specific buffer as needed.

Contributing to poorer than expected buffer contention values is the possibility that more pages than anticipated must be read due to the configuration of the data within the area. A major cause of this type of condition is excessive overflow for CALC records or VIA clusters which can be monitored on a program basis by using the ratio RECORDS-REQUESTED/PAGES-READ. This ratio is also available on JREPORT04 under the heading of Space Management.

The value of this ratio should be as large as possible but knowledge of the processing of the various programs is required to make a valid interpretation. Once again programs that do extensive CALC retrievals will tend to have lower values often in the range of 2.0. Consistent values below 2.0 may indicate excessive CALC overflow conditions. When evaluating programs that walk VIA sets you need to have an idea of how many record occurrences to expect in the average cluster and how many member records may fit on a given page. For example if the average VIA set is expected to contain 100 records but only 50 records can fit on a page the entire set would require 2 pages. A Space Management ratio of around 50.0 would be the ideal value for that program's processing.

The Space Management ratio may degrade over time due to the manner in which an application stores and retains data or it can occur due to larger cluster sizes than originally anticipated when the area's page size was selected. The IDMSDBAN utility can be used to review the clustering characteristics of any

VIA sets suspected of having excessive overflow. For each chain set reported the utility will produce two histograms. The Chain Length Histogram will break down the number of set occurrences based on the number of members within each owner's chain. This information can be used to determine an average length of the set and will also alert you to whether there are any occurrences seriously deviating from that average. The Page Change Histogram results from IDMSDBAN walking the set and determining how many times the page number for the current member changed from the previous member occurrence. This histogram in conjunction with the Chain Length Histogram can provide insight as to whether the sets being reported on may have overflow problems.

Corrective action would require the execution of the UNLOAD/RELOAD, REORG, or DB-REORG utilities to reorganize the problem areas resulting in the consolidation of the VIA clusters. During this processing it may be desirable to re-evaluate the page size assigned to the affected area and a possible increase in the total number of pages assigned to the database area.

Higher than expected rates of I/O when index processing is involved may be an indication of inefficient index structures. Processing that displays high I/O activity during STORE or CONNECT commands or during retrieval processing while the index area is readied in a retrieval mode may be due to an excessive number of index levels. Whereas high I/O activity during ERASE or DISCONNECT commands or commands that walk the index while the index area is readied in an update mode might be due to excessive orphan occurrences within the index structure.

Starting with CA IDMS Release 17.0 the PRINT INDEX utility was enhanced to include the SUMMARY parameter. This option allows you to get an overview of the condition of an index structure including the number of members within the index, the number of levels of SR8 records making up the index, and the overall number of orphans present. A general guideline for an index is that it is made up of 3 or 4 levels of SR8 records. Given the number of data records indexed by the structure you might want to review the calculations of the page size, Index Block Count (IBC), and Level-0 SR8 displacement to see if they are still valid. Formulas for these calculations can be found in the CA IDMS Database Design Guide. If the new values calculated are different than what is currently used you should consider adjusting the index's physical implementation and rebuild the index using the MAINTAIN INDEX utility.

Indexes that have a high volume of insertions and deletions are especially prone to developing high numbers of orphaned records. You should consider rebuilding these indexes on a regular interval.

Journal Activity

For update transactions journal activity can be a hidden yet significant bottleneck. The following are the journal considerations that may have the greatest impact on a transaction's performance:

- Journal page size
- Journal buffer
- Journal transaction level

When CA IDMS needs to update a database page all journal blocks containing images associated with that page must be successfully written before any attempt is made to issue the write for the page. Large numbers of waits for journal writes may be an indication that the journal page size is improper. You should insure that the journal page size is at least large enough to contain both the BFOR and AFTR image of the largest database record to be processed within the CV. Too small a page size could generate multiple journal writes for each database record being updated.

To allow for journal images to be created while earlier blocks are written a pool of buffer pages is maintained by a CA IDMS CV. Frequent waits on the JBC ECB may indicate that the number of pages within this pool is insufficient for the volume of data being written. This pool defined in the CV's DMCL should be increased in size, especially if most journal blocks are not full when written as reported by the ARCHIVE JOURNAL utility. If the majority of the journal blocks are full when written and the JOURNAL TRANSACTION LEVEL feature is not being used consider increasing the journal block size to accommodate more data per block and reduce the number of required journal writes.

JOURNAL TRANSACTION LEVEL is a feature of CA IDMS that helps to maximize the space within a journal block. When specified within a CV's sysgen it provides a means to defer the writing of journal blocks when some type of journal checkpoint record has been generated. When the number of update transactions active within the CV meets or exceeds the number specified as the JOURNAL TRANSACTION LEVEL the writing of a journal block will be deferred until that block becomes full. This can significantly reduce the amount of journal writes occurring within a CV that performs a high volume of update transactions.

If used the JOURNAL TRANSACTION LEVEL value should never be smaller than 3 to avoid the possibility that the deferment of the journal writes can noticeably delay the completion of update transactions. Frequent JBEE ECB waits or long lasting journal write waits when using this feature can be an indication that there is not enough journal activity at the specified level to quickly fill the journal blocks. If this should occur increase the transaction count so there are more concurrently active update transactions before deferment of the journal writes occur.

More detail on the definition of a CV's journals can be found in a separate paper entitled CA IDMS CV Journal Sizing which was originally published in the March 2009 edition of the IUA Connections newsletter.

DBKEY Contention

Waits on the DBKEY ECB can indicate contention for access to database records which can result in slower run-times and ultimately lead to deadlock conditions. Reductions in the amount of record lock contention can also reduce CPU usage and storage overhead. Sites that have the CA IDMS Performance Monitor have various reports that may be examined to determine the volume of dbkey waits being experienced within their environment. The Interval Monitor provides reports PMIRPT01 (Management Summary) and PMIRPT05 (Dbkey/Area Detail) that allow you to get an overview of the volume of dbkey

waits that are encountered during various time periods. Application Monitor report PMARPT36 (Task Wait Detail) is available to identify the number of dbkey waits experienced by individual tasks.

The most effective steps to reduce dbkey contention typically require changes to the application logic or the database design. However there are some steps that can be taken to minimize contention on a CV level.

The first thing to do is to insure that the CV is not generating needless record locks. Unless specifically needed the CV's sysgen should specify parameters RETRIEVAL NOLOCK and UPDATE NOLOCK. The RETRIEVAL NOLOCK parameter suppresses record locks for retrieval-only transactions. This does open a small window for what are referred to as dirty reads. These are reads where the retrieval transaction accesses records that are being concurrently updated by another transaction. Most sites find this level of concurrency acceptable in favor of reducing lock contention within the CV. UPDATE NOLOCK tells the DBMS to suppress record locks for transactions accessing their database areas in a PROTECTED UPDATE mode. Since only one transaction can update an area in PROTECTED UPDATE at a given time the only reason to use UPDATE LOCK would be in conjunction with RETRIEVAL LOCK to eliminate the possibility of dirty reads by retrieval transactions during the protected update. Each of these options should be reviewed based on the potential impact of a dirty read on the applications involved.

Proper scheduling of the application's processing is another factor in reducing dbkey waits. When possible any program that generates large numbers of record locks such as batch jobs should not be scheduled during periods of high volume online tasks when response time is critical. Even if the large update process does periodic commits the possibility of waits due to dbkey lock contention can impact the overall environment due to additional CPU and storage usage. Also avoid maintenance operations such as index tuning during critical processing timeframes.

Summary

Tuning of the database aspects of a CV environment is a good place to start when performance problems occur. Typically these problems may be caused by one of the following factors:

- I/O waits
- Journal activity
- DBKEY contention

First identify those programs processing during the timeframe when the performance problems are experienced and become familiar with the type of database access they are performing. Next obtain performance related statistics from timeframes when performance is acceptable as well as those times when a problem exist for the purposes of comparison. Finally after identifying problem areas take corrective action.

However always be aware of the fact that no level of environmental tuning can overcome all of the effects of poor database or application design and always be open to the possibility of making changes in those areas.