Using Transaction Statistics to Subdivide CA IDMS Task Statistics

Dick Weiland

CA Technologies

Principal Software Engineer

The use of CA IDMS DC task statistics is a common method for determining the resource utilization of a single CA IDMS-DC task, while transaction statistics were designed to accumulate resource information across logically related tasks. However it may be desirable at times to be able to isolate the resource utilization of a portion of a single DC task. The portion of a task being monitored might be a called subroutine or just a segment of code within a single program. This can be accomplished by initiating the use of transaction statistics before the invocation of the monitored code, terminating these statistics at the conclusion of this code, and writing the statistics to some medium for subsequent review.

What are Transaction Statistics?

Transactions statistics are a facility available within a CA IDMS-DC (CV) address space that allows a user to collect statistics across multiple CV tasks that make up a logical transaction. For example, a user may wish to determine the total resources that it takes to store a new order onto their database but the entire process requires 3 separate tasks within the CV. Trying to identify the 3 logically related tasks within statistical reports may be cumbersome but by enabling transaction statistics the user has the ability to consolidate the statistics created by the 3 tasks and report on them as a single entity.

The statistics provided are too extensive to attempt to describe them in detail in this document, but include the standard statistical categories of DC, Extended DC, DB, Extended DB, and SQL. Transaction statistics are made available to a user through a control block returned to the application program or written to the CA IDMS DCLOG. The description of this control block can be found in dsect #TSBDS (TSB) in the CA IDMS DSECT Reference Guide. Their description can also be copied into a COBOL program by using the dictionary record named TRANSACTION-STATISTICS.

To be able to collect transaction statistics both task and transaction statistics must be enabled for the CV. This is accomplished by including the 'STATISTICS TASK TRANSACTION 'clause on the SYSTEM statement of the CV's sysgen. If not specified on the SYSTEM statement, transaction statistics can be enabled while the CV is active by executing task 'DCMT VARY STATISTICS TRANSACTION ON'.

The use of transaction statistics also requires the insertion of code into the application programs at the points where statistical collection is to begin and where the collection process is to be terminated. These functions are provided in a COBOL program by use of the BIND TRANSACTION STATISTICS and END TRANSACTION STATISTICS commands respectively. It is also possible for a program to access the current values contained within the TSB by executing an ACCEPT TRANSACTION STATISTICS command. It

should be noted that transaction statistics are never reset between a BIND and END command so multiple serial ACCEPT commands will contain cumulative values.

A user has options as to how the transaction statistics will be made available for viewing. The ACCEPT TRANSACTION STATISTICS and END TRANSACTION STATISTICS commands have options that will direct CA IDMS to write the TSB to the DCLOG. The statistics can then be reported on using statistics reports SREPORT 010, SREPORT 011, or SREPORT 021using an archived DCLOG file as input. Each of these commands also contains options to return the TSB to an area of an application program's storage. This area could be snapped to the DCLOG using a SNAP command for viewing. The TSB will be displayed in a dump format when viewed through an OLP task or by running the PRINT LOG utility. Sites have also been known to write this block of storage to the CA IDMS journal files using a WRITE JOURNAL command and then writing in-house routines to report on the numbers extracted from archived journal files.

Transaction statistics can also be invoked from Assembler and PL/I programs. In an Assembler environment these statistics are controlled through the use of the #TRNSTAT macro which has variants for the BIND, ACCEPT, and END functions. It should also be noted that these statistics are only available within a CA IDMS-DC environment as the TSB is anchored off of the LTERM control block associated with a terminal defined within the CV.

Subdividing a DC Task

In a complex online application it would be unusual for any one program to handle all aspects of a transaction's functions. Frequently a program may call other subroutines to perform various functions from mainline processing for some specific end-user requests to accessing portions of the database 'common' to a wide range of the applications transactions. When performance problems occur it can be difficult to identify which program of all the possible routines being invoked is at fault. The inclusion of transaction statistics commands at strategic locations within the application code can provide statistical information about resource usage by each program. This makes it easier to locate the problem program instead of attempting to use the entire task's statistics which will combine the numbers from all programs involved into a single set of values.

The use of transaction statistics can be implemented in a number of ways based on the scope of the performance testing to be provided. If faced with a single DC task that is performing poorly you might decide to use transactions statistics in an ad hoc manner and simply snap the TSB to the DCLOG for a quick review using the OLP task or printing the appropriate time range of an archived log using PRINT LOG. In other sites the number of routines to be reviewed may be large enough to justify the creation of a standard procedure for inclusion into each new program being created. In such a case, the various TSBs might be written to the journal file after including additional identifying information and a suite of reporting programs developed to report on the user records extracted from the journals.

COBOL Implementation

All COBOL implementations of transaction statistics begin with the following BIND command at the point at which you wish to start collecting statistics. The command causes CA IDMS to allocate the TSB internally and assigns an identifier composed of the 32 byte user identification, if any, and the 8 byte logical terminal id. Within a COBOL implementation statistic collection, always begins from the point in time when the BIND command is issued.

```
BIND TRANSACTION STATISTICS.
```

At the completion of the segment of code to be monitored the program must issue a variant of the END TRANSACTION STATISTIC command. If it is acceptable for the TSB to be written to the DCLOG the following command will terminate statistics collection, write the TSB to the DCLOG, and free the internally allocated TSB.

```
END TRANSACTION STATISTICS WRITE.
```

When subdividing a task's statistics using transaction statistics the problem with the above command is that there is nothing in the data written to the DCLOG to further identify what code had been measured. To provide this information the contents of the TSB can be copied back to the calling program. The following command can be used to copy record TRANSACTION-STATISTICS from the dictionary into the COBOL program's WORKING-STORAGE or LINKAGE SECTIONS.

```
01      COPY IDMS RECORD TRANSACTION-STATISTICS.
```

Within the TRANSACTIONS-STATISTICS record is data field USER-SUPP-ID. This field is an 8 character field into which further identifying information can be moved. The following pseudo-code shows how transaction statistics could be implemented in a routine designed to call COBOL subroutines and outputting each subroutine's statistics by snapping the program's copy of the TSB to the DCLOG. Remember that the SNAP command will write the storage to the DCLOG in a text format similar to a dump that can only be viewed by using the OLP task code or printing a section of the DCLOG using the PRINT LOG utility. It is the user's responsibility to determine the location of the desired statistics in the snap output using the TRANSACTION-STATISTICS record layout or the #TSBDS dsect as a guide.

```
BIND TRANSACTION STATISTICS.
TRANSFER CONTROL TO SUBROUTINE-NAME LINK.
END TRANSACTION STATISTICS NOWRITE INTO TRANSACTION-STATISTICS.
MOVE SUBROUTINE-NAME TO USER-SUPP-ID.
```

```
SNAP FROM TRANSACTION-STATISTICS TO TSB-STATS-DCX-FILL01.
```

An implementation that would write the program's copy of the TSB to a user journal record for subsequent processing by a user-written report program is as follows.

```
BIND TRANSACTION STATISTICS.
TRANSFER CONTROL TO SUBROUTINE-NAME LINK.
END TRANSACTION STATISTICS NOWRITE INTO TRANSACTION-STATISTICS.
MOVE SUBROUTINE-NAME TO USER-SUPP-ID.
WRITE JOURNAL NOWRITE FROM TRANSACTION-STATISTICS.
```

Descriptions for all of the CA IDMS DML commands in this section can be found in the DML Reference Guide for COBOL.

Assembler Implementation

Although more application code is written in COBOL, it is worth the effort to examine the Assembler implementation of transaction statistics which is controlled by the use of the #TRNSTAT macro. The #TRNSTAT has 3 major variants that are controlled by the TYPE= parameter. Each of the functions discussed earlier for the COBOL commands BIND TRANSACTION STATISTICS, END TRANSACTION STATISTICS, and ACCEPT TRANSACTION STATISTICS are respectively performed using the TYPE=BIND, TYPE=END, and TYPE=ACCEPT versions of the #TRNSTAT macro. The #TRNSTAT macro is described in the CA IDMS DML Reference Guide for Assembler.

Anyone intending to create an application independent performance monitoring tool employing transaction statistics may wish to develop it using Assembler as there are two undocumented parameters used with the TYPE=BIND option that provide a bit more flexibility. The ID= parameter provides a mechanism to place additional identifying information into the TSB when the transaction statistics monitoring session is initiated. This replaces the need within a COBOL environment to bring the TSB into the program's storage and move this identifying information into field USER-SUPP-ID of the TRANSACTION-STATISTICS record followed by writing the program's TSB storage to some medium from which the end-user can access it. The following code is an example of the use of the ID= parameter where the field labeled COMPNAME is used to hold the name of the component being monitored.

```
LA          R5,COMPNAME
#TRNSTAT    TYPE=BIND,TASK=NO,ID=(R5)
```

The #TRNSTAT macro also provides the TASK= parameter. This parameter allows a user to determine whether the transaction statistics session being initiated will contain statistics since the beginning of the current task or should start collecting from the point at which the #TRNSTAT TYPE=BIND macro is

executed. The default setting for this parameter is TASK=YES which instructs CA IDMS to collect transaction statistics from the start of the current task. However when subdividing a task's statistics to measure a subset of the task it is preferable to specify TASK=NO, which starts statistic collection from the point at which the #TRNSTAT TYPE=BIND macro is executed. This is similar to the way a BIND TRANSACTION STATISTICS command works within a COBOL environment.

At the end of the code to be measured a #TRNSTAT TYPE=END macro must be executed. The TYPE=END variation of the macro will also instruct CA IDMS whether the TSB should be written to the DCLOG or returned to user storage for subsequent processing. The following code example can be used to write the TSB to the DCLOG. Notice that there is also a TASK=NO parameter for this variation of the macro. In this case the TASK=NO parameter instructs CA IDMS to immediately write the TSB instead of waiting until the end of the current task.

```
#TRNSTAT    TYPE=END,WRITE=YES,TASK=NO
```

If it is not desirable to write the TSB to the DCLOG it is possible for the control block to be returned to the user's storage from which the program can output the information to wherever is convenient. As discussed earlier this information might be snapped to the DCLOG in a dump format for a quick review or written to the journals as a user journal record for later extraction from a journal archive for reporting through some user developed program. The following code example will write the user TSB storage to the CA IDMS journal using the #PUTJRNL macro. The address represented by label USERTSB is the start of the user storage where CA IDMS will return the TSB contents.

```
LA          R5,USERTSB
#TRNSTAT    TYPE=END,WRITE=NO,RECORD=(R5)
#PUTJRNL    RECORD=(R5),RECLEN=TSBLEN,OPTIONS=NOWAIT
```

The following code examples illustrate how the BIND and END variants of the #TRNSTAT macro would be used to measure the resources of a subroutine linked to from a calling program.

Writing the TSB to the DCLOG:

```
LA          R5,COMPNAME
#TRNSTAT    TYPE=BIND,TASK=NO,ID=(R5)
#LINK       PGM=(R5)
#TRNSTAT    TYE=END,WRITE=YES,TASK=NO
```

Writing the TSB to the journal:

```
LA          R5,COMPNAME
#TRNSTAT    TYPE=BIND,TASK=NO,ID=(R5)
#LINK       PGM=(R5)
LA          R5,USERTSB
#TRNSTAT    TYPE=END,WRITE=NO,RECORD=(R5)
```

```
#PUTJRNL        RECORD=(R5),RECLEN=TSBLEN,OPTIONS=NOWAIT
```

Conclusion

Although transaction statistics were designed with the idea of grouping together multiple CA IDMS-DC tasks to measure the resource utilization of a logical transaction we have seen that their usage can be modified to measure portions of individual tasks. This subdivision being measured may represent entire called subroutines or could be just a segment of code within a single program. Transaction statistics do require some code to be inserted into the modules to be monitored but they can provide the DBA and application development teams with a powerful tool with which to tune the performance of online applications.

Dick Weiland is a Principal Software Engineer for CA IDMS Level II Support working out of the CA Lisle Office. He began working with CA IDMS in 1977 as a DBA and joined Cullinet in 1981 in the Field Support organization. He moved to Level II Support in 1988 and is responsible for the DBMS engine and various utilities.