

Analyzing Non-Shared Resource (NSR) Files

October 15, 2004

By: Eugene S. Hudders

CICS/TS Performance Tuning Using C\TREK Course

This seminar covers a lot of tuning material and will require 8 hours a day. The total time for the seminar is 40 hours.

The followings topics included in this course are:

- Introduction to CICS Performance Tuning
- Using Operating System Information to Tune CICS/TS
- Tuning CICS/TS Processor Cycles
- Tuning Real Storage in CICS/TS
- Tuning Virtual Storage in CICS/TS
- Tuning CICS/TS Transaction Controls
- Tuning On Line VSAM Files
- Tuning NSR Files in CICS/TS
- Tuning CICS/TS LSR Buffer
- Tuning the Index CISZ
- Reviewing VSAM Free Space
- Reviewing the DB2 Interface

If you wish to attend this course, please send us an email at ctrek@actpr.com

In a previous article we explored the tuning options available for files residing in a Local Shared Resource (LSR) pool. This article explores the tuning associated with Non-Shared Resource (NSR) files. As a starting point, an important question to ask is why is this file in NSR? There has to be a very strong justification as to why a particular file has been assigned to use NSR buffering instead of LSR because of the better performance that can be achieved using LSR's more superior look-aside algorithm. This article explores NSR and the reasons to use this type of buffering technique in CICS. We will concentrate on performance issues associated with VSAM KSDS files in our review of NSR. One of the first tasks that must be done is to determine which files are designated as NSR (Figure 1).

```

FNSR ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      0A03C168  DATE 10/15/2004
APPLID CICSTS23                   APPLICATION DOMAIN  TIME 15:41:30
VERSION 6.3                       NON SHARED RESOURCES FILES  TERM 0206
                                     NORMAL
DDNAME  TOTAL I/O  NUM STRN  STRN  STRN  BFND  BFNI  BUFR  BUFR  STRN  BFND  BFNI  A
CTREKHLP      0    5    4    0    0    32    5    0    0    5    32    7  +
DFHDBFK      CLO DIS R U A D B VSAM KSDS
    
```

ENTER REFRESH ENTER+CURSOR DETAILS PF1 HELP PF2 SEARCH PF3 PREV PAGE
PF5 MEMORY PF7 BACKWARD PF8 FORWARD PF10+CURSOR LISTCAT CLR MAIN MENU

4B █ :00.4 07/54

Figure 1. NSR BUFNI Recommendation Display

NSR is a means of processing VSAM files in which the resources are dedicated to the file. Specifically, dedicated means that the control block structures including the strings and buffers are owned by the VSAM file to which they were defined. NSR is the default access for batch jobs and can be optionally selected for on-line CICS files by placing a NONE when defining the LSR pool id in the file definition under RDO. In this case, all strings and buffers defined are

allocated for exclusive use by the file. To properly tune an NSR file, you would need to know how the file is being accessed because the buffer allocations are different. In general, when accessing the file sequentially, you want to allocate additional data buffers (BUFND) to be able to overlap data I/O operations such as reading ahead. When accessing the file directly, you would want to have additional index buffers (BUFNI) to be able to get better index look-aside hit ratios. In the case where the file is dynamically accessed, you would probably want to increase both the data and index buffers.

In a CICS on-line environment, sequential processing occurs when you browse the file (e.g., STARTBR, READNEXT and ENDBR) or whenever a CA split occurs, the shifting of CIs from one CA to another is a sequential process. In these cases, having defined additional BUFNDs can help improve the time it takes to perform the sequential operation. On-line direct processing of VSAM files (e.g., READ) occurs whenever you issue commands requesting a specific record. In this case, additional BUFNIs can improve direct access. The challenge would be to determine how many buffers are required for optimum performance. Let us first review how a VSAM file uses the resources allocated. We will review direct processing first.

An NSR file that has one string assigned requires at least two data buffers (BUFND) and one index buffer (BUFNI). The extra BUFND is used to process splits. Each string requires one data and one index buffer. The data buffer is used to read the appropriate CI while the index buffer is used to read the entire index CIs associated with locating the proper data CI. In other words, a file that has three index levels requires three index CIs to be read directly before the location of the data CI is identified. Thus, there are potentially four I/O operations possible, three for the index that use the index buffer and one for the data that uses the data buffer. The major problem is that there is only one index buffer. So, subsequent file requests will require that the three index I/O operations be done again even if the same CI is requested. This is due to the fact that each index read overlays the previous index CI. By the time you read the last index record, all higher-level index CIs would be overlaid and unavailable. Therefore, additional index buffers would help improve this condition. Additional index buffers are used to house the index set records (2nd and higher level index). So, if additional BUFNIs were specified, then you could avoid reading these CIs in future requests. The extra BUFNI buffers provide additional look-aside capacity to an NSR file.

The important performance issue associated with processing direct KSDS files is determining the number of index buffers to be allocated to the file to ensure the best look-aside for the high level index that we mentioned were the 2nd level and higher indices. These indices are called the Index Set (IS) in order to differentiate them from the lowest index level called the Sequence Set Index (SSI). A VSAM LISTCAT provides us with the necessary information so that we could determine how many IS indices are on a file (Figures 2, 3 and 4). Before entering into the formula, you should look at the number of index levels in the file to avoid unneeded computations. The number of index levels can be found on the second LISTCAT page at the bottom right hand side (Figure 3). If the file has two index levels, then there is only one Index

Set record. Therefore, only one additional BUFNI is required. If the index levels is equal to one, then there are no IS records and no additional BUFNIs are required. In this case there is only one SSI record.

The information required from the LISTCAT is as follows:

- The total number of index records (Figure 2)
- The data CISZ (Figure 2)
- The number of data CIs per CA (Figure 2)
- The High Used RBA (HURBA) (Figure 3)
- The total number of strings assigned (Figure 4)

```

LCAT ADVANCED COMPUTER TECHNOL C\TREK ON-LINE 0A081648 DATE 10/15/2004
APPLID CICSTS23 APPLICATION DOMAIN TIME 15:36:12
VERSION 6.3 DATA SET LIST CATALOG TERM 0206

DSN CTREKHP ACT.CTREK.HELP.TEXT.ALL BASE CLUSTER
CL DATA NAME ACT.CTREK.HELP.TEXT.ALL.DATA LOCAL
CL INDX NAME ACT.CTREK.HELP.TEXT.ALL.INDX VARIABLE BLOCKED
VSAM KSDS OPEN ENABLED READ UPDATE ADD DELETE BROWSE
DATA INDEX DATA INDEX
FRSPC BYTES 2,783,232 36,864 BYTES USED/TRACK 55,296 55,296
CA SIZE (TRKS) 15 1 % TRACK USED 97.58 97.58
BYTES PER CA 829,440 55,296 PHYSICAL REC SIZE 18,432 18,432
CA SPLITS 1 0 PHYS RECS/TRACK 3 3
% CA FREE SPACE 2 0 REC LEN (EST AVG) 275 18,425
FREE CI PER CA 0 0 REC LENGTH (MAX) 1,952 18,425
CI SIZE 18,432 18,432 KEY LENGTH - 8
CIS IN CNTRL AREA 45 3 RELATIVE KEY POS - 0
AVAIL BYTES/CI 18,227 18,401 TOTAL RECS 4,697 4
CI SPLITS 14 1 INSERTS 6,573 0
% CI FREE SPACE 0 0 UPDATES 412 16
FREE BYTES PER CI 0 0 DELETES 6,444 0
NUMB OF EXTENTS 1 1 RETRIEVES 6098938 0
ENTER REFRESH PF1 HELP EXCPS 29498 34061
PF3 PREV PAGE PF5 MEMORY PF6 RECOMM PF11 SCROLL RIGHT CLR MAIN MENU MORE... R

```

Figure 2. File LISTCAT Information (Part 1)

```

LCAT ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      0A081648  DATE 10/15/2004
APPLID CICSTS23                   APPLICATION DOMAIN    TIME 15:37:30
VERSION 6.3                       DATA SET LIST CATALOG  TERM 0206

DSN CTREKHLP ACT.CTREK.HELP.TEXT.ALL              BASE CLUSTER
CL DATA NAME ACT.CTREK.HELP.TEXT.ALL.DATA        LOCAL
CL INDX NAME ACT.CTREK.HELP.TEXT.ALL.INDX          VARIABLE BLOCKED
  VSAM KSDS OPEN  ENABLED  READ UPDATE ADD DELETE BROWSE
      DATA      INDEX
ALLOCATION          CYL      TRK  SPEED/RECOVERY  SPEED  RECOVERY
  PRIMARY          5        2  ROTATION POS SENSING YES      YES
  SECONDARY        1        1  REPLICATE          NO      NO
VOLUME ID          OSW005   OSW005  IMBED              NO      NO
HI ALLOC RBA      4147200   110592  ORDERED            NO      NO
HI USED RBA       2488320   73728  KEYRANGES          NO      NO
% UTILIZATION     60.00     66.66  SPANNED RECORDS   NO      NO
TRACKS PER CYL    15        15     WRITECHECK         NO      NO
BYTES PER TRACK   56,664     -      TME STAMP DATE    10/15/2004 10/15/2004
SHR OPTIONS       3,3        3,3    TME STAMP TIME    09:50:07.6 09:50:07.6
LSR POOL          NS          -      EDB ADDRESS       7F4EE050 7F4EE1C8
NUMBER OF BUFFERS 32          5      LPMB ADDRESS      7F4EE078 7F4EE1F0
% LOOK ASIDES     .00         .00    PLH ADDRESS       2242A25C 2242A25C
INDEX LEVELS      -          2      AMDSB ADDRESS     093D85E0 093D8418
PF1 HELP PF3 PREV PF5 MEM PF6 RECOMM PF10 LEFT PF11 RIGHT CLR MENU MORE...L/R
  
```

Figure 3. File LISTCAT Information (Part 2)

```

LCAT ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      0A081648  DATE 10/15/2004
APPLID CICSTS23                   APPLICATION DOMAIN    TIME 15:52:09
VERSION 6.3                       DATA SET LIST CATALOG  TERM 0206

DSN CTREKHLP ACT.CTREK.HELP.TEXT.ALL              BASE CLUSTER
CL DATA NAME ACT.CTREK.HELP.TEXT.ALL.DATA        LOCAL
CL INDX NAME ACT.CTREK.HELP.TEXT.ALL.INDX          VARIABLE BLOCKED
  VSAM KSDS OPEN  ENABLED  READ UPDATE ADD DELETE BROWSE
      DATA      INDEX
NUMBER OF BUFFERS 32 AFCT ADDRESS          BUFFER POOL      0A0742E0
TOTAL WAIT BUFFER 0  FCTE ADDRESS          0A081648 VSWA ADDRESS    00000000
HI WAIT ON BUFFER 0  FRTE ADDRESS          00000000
NUMB STRINGS      5  DSNB ADDRESS          0A03FE00 ASCB ADDRESS    00F9BB80
# UPD/ADD STRINGS 5  ACB ADDRESS           0A07D080 DSAB ADDRESS    008C9E88
TOTAL WAIT STRING 0  AMBL ADDRESS          08AAE680 TIOT DISPLACEMENT 0000
CURR WAIT STRING  0  DATA AMB ADDRESS     22401B90
HI WAIT STRING    0  INDX AMB ADDRESS     2242B1E8
AT MAX STRINGS    40  BASE CLUSTER         093B3948
AT PSUDO MAX STRNG 40  TIME FILE OPEN       06/28/2003 02:05:42
DISPOSITION       SHR
LOGGING           NO
LOG ID            00
ENTER REFRESH PF1 HELP PF3 PREV PAGE
PF5 MEMORY PF6 RECOMM PF10 SCROLL LEFT PF11 SCROLL RIGHT MORE...L/R
  
```

Figure 4. File LISTCAT Information (Part 3)

In order to compute the number of records in the IS you need to obtain the information from a LISTCAT and follow the detailed steps:

1. Compute the number of Control Areas (CA) in the file. The number of CAs is equal to the number of SSI records (lowest level index):

$$\frac{\text{High Used Data RBA}}{\text{Data CISZ} * \text{CI/CA}} = \text{Total Number of CAs in the File}$$

2. Locate the number of index records in the file and using the above result compute:

$$\# \text{ Index Records} - \# \text{ of CAs in the File} = \text{Total \# of IS Records}$$

3. The total number of IS records from the above formula represents the minimum additional BUFNIs required for the file. Using the above figure compute:

$$\# \text{ File Strings} + \# \text{ of IS Records} = \text{Total \# of Required BUFNIs}$$

Once you have determined the number of IS records in the file (Step 2), then you need to add this result to the total number of strings (Figure 4) to determine the number of BUFNIs required to hold the IS and the associated SSI record. Note that each string requires one index buffer. So, the number of strings indicates the number of index buffers required to cover the string allocation. This figure has to be adjusted by the formula results (Step 3) in order to reflect all the buffers required.

We mentioned that the result from the formula was the minimum buffers required for processing the NSR file. Why is this the minimum? Well, if we have a volatile file, then there is the exposure that there be splits within the file. Splits not only occur in the data portion but also occur in the index portion. An index split can result in a CA split in the index component. This extra index CI can be another IS record that was not included in the formula. So, if the file exhibits CA split capacity, then it is best to add a few more BUFNIs to account for the extra IS CIs that may occur during the life of the file. Usually, two or three extra BUFNIs are sufficient unless the file is infrequently reorganized. In this case, you would have to compute the number of IS records using the above formula just before reorganizing the file. C\TREK performs the above analysis and provides the BUFNI recommendation for the NSR file (Figure 1).

In the case of a three level index file, adding sufficient index buffers will ensure a look-aside hit ratio of at least 50% once the extra IS buffers are full. Is there any way of increasing this hit ratio? Yes, there is a look-aside done at the string level buffers, that is, the string index buffer is searched to see if the desired index SSI CI is present and use it if so. In addition, the data buffer is also searched when the index SSI was found to see if the data CI is also present. As you add more strings to the file, then you wind up with less opportunity to be assigned the

correct string. However, there is a chance that you could get better than 50% look-aside hit ratio.

NSR also provides the capability of improving sequential operations such as browses and CA splits. Sequential activity can be improved by adding additional data buffers in excess of the ones needed by the strings and the extra one used by splits. However, it is important to note that all extra buffers available are assigned to the first person desiring these extra buffers. Therefore, one cannot give more importance to CA split processing over a normal browse operation because the first one takes all the available buffers. However, the problem lies in how many extra data buffers are needed for definition. For example, you would want a browse transaction not to have to wait for the records in a browse operation by reading ahead.

However, in order to determine how many buffers you would want to read ahead, you would have to know the applications running in the computer system. The first major problem is how do you determine how many I/O operations are normally requested for a sequential browse. The application programmers probably don't know the answer to that question either. So, you may decide to add ten additional buffers above the ones allocated to the strings and the extra one for splits. When the program begins its first browse operation, CICS assigns all of the available extra buffers (10) to the operation. So, the I/O would read eleven CIs into the allocated buffers, the one normally assigned to the string plus the ten additional ones. Reading eleven buffers elongates the response time because you need to wait until eleven buffers are read and the I/O operation completed before the application program can begin to process the data. What happens if the application program ends the browse after the second or third buffer is processed? Then the remaining buffers are not used and you added additional overhead to the transaction by reading extra data that were not used.

Also, what happens to a second browse request that may occur while the first browse is still active? The second request would get one buffer, the one assigned to the string for the request. Another important question to ask is how much data should the transaction process? The answer is as much as it can display on one screen. Most browse transactions can only display one screen of information or around sixteen to eighteen lines of detail information. So why read more than can be displayed. Reading more than the capacity of the display would mean that the program would have to store the data (e.g., Temporary Storage) and have to re-access it when needed, adding to the overhead. In addition, a transaction that finds the desired record in the buffer tends to monopolize the system affecting other transactions running concurrently. Therefore, adding more buffers to benefit one transaction may have a negative effect on the overall system response and may waste I/O operations.

A better alternative to heavily browsed files is to ensure the proper CISZ rather than adding buffers. The idea is to determine the average number of browse requests made by the user to the file. As we previously mentioned, this information is not readily available but C\TREK does provide an estimated number of READNEXT operations issued to the file (Figure 5). Once this

figure is known, then you would try to make the CISZ sufficiently big so that the number of records to be read fit into one CI. For example, suppose you determine that the average number of browse requests was fifteen. The objective would be to make the CISZ sufficiently large enough to accommodate fifteen or more records. If the maximum CISZ cannot accommodate the number of records or generates a bad CISZ that results in poor track utilization, then select a CISZ that would reduce the number of I/O operations required to process the browse request. Naturally, the browse request could start in the middle of a CI and require an additional I/O to complete the browse. However, you would limit the number of physical reads required to service the browse request to a few external operations. Larger CIs also provide better free space alternatives that will be useful in the next topic regarding CA splits.

```

FCTX ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE  0A081648  DATE 10/15/2004
APPLID CICSTS23                APPLICATION DOMAIN  TIME 16:32:16
VERSION 6.3                    DATA SET DESCRIPTION  TERM 0208

DSN CTREKHLP ACT.CTREK.HELP.TEXT.ALL          LOCAL
  VSAM KSDS OPEN  ENABLED  READ UPDATE ADD DELETE BROWSE
----- FCTE STATS -----  --- REMOTE FILE ---  --- DATA TABLES ---
READ REQUESTS                0 DDNAME                TAB ADDRESS
ADD RECORDS                  0 SYSTEM                TAB NAME
UPDATE REQUEST                0 REC SIZE              PATH ADDR
GET FOR UPDATE                0 KEY LENGTH            OPEN FL CHN
BROWSE REQUEST                398 LAST CONN            TAB SIZE
BROWSE UPDATE                 0 STATUS                REC HWM
DELETE REQUEST                0                      READS
                                ADDS                   VSAM
DATA EXCPS                    7 DELETES              REWRITES
INDEX EXCPS                   0 READS                 DELETES
                                GET UPDT              ADDS TOT
EST AVG OF BRW REQ           56.85 UPDATES            TAB FULL
AVG # OF RECS/CI             66.00 BROWSE              LOADED
RATIO                          .86 BROWSE UPDATE        RECOVERABLE

PF3 PREV PAGE PF7 PREV FILE PF8 NEXT FILE PF12 MORE OPTIONS CLR MAIN MENU
01/54
  
```

Figure 5. Estimated Number of READNEXT Requests

What about having extra buffers to improve CA split processing? The first problem is that if there is any browse activity, there is no way to predict what activity occurs first and acquires the extra buffers. Second problem and probably more important, the handling of CA splits under NSR ties up the TCB on which the split occurs for the duration of the split processing. This affects the response time of all transactions using this TCB. In the case where the SUBTSKS parameter has not been specified, then the QR TCB will be locked out for the duration of the split. In the case that the SUBTSKS parameter has been specified, then the CO TCB is locked up for the duration of the split. It is important to activate the CO TCB in CICS systems that have files on which splits can occur. Unfortunately, the CO TCB is only recommended for multiprocessing systems that have more than one CPU available for processing.

Based on the above information, we have to return to our original question – why is this file in NSR? NSR does not provide a good look-aside hit ratio when compared to LSR and in the case of splits, can lock out the TCB processing the split. LSR uses a VSAM exit to process splits and does not tie up the processing TCB. In addition, adding additional buffers can actually have a negative effect if the buffers read-ahead are not processed because the operation completed resulting in slower processing and wasted resources. What type of file is suited for NSR? Basically, any file that has Share Options 4 specified. This type of file should not be in LSR because it has a negative effect on the buffers and the hit ratio. Another type of file that could be in NSR is a file being processed by a program that does not follow Command Level guidelines. For example, a program that in the middle of a READNEXT sequence, issues a read for update. If this file is in LSR, the transaction will hang. However, if this file is in NSR, it will work as long as an additional string is available because the read for update requires a new string to operate. However, you wind up re-reading the CI into virtual storage and have the record twice in memory. This type of processing led to hung transaction even in NSR because of lack of strings. So, the system programmer generally over-allocated strings to the file to avoid string lockouts. This results in wasted resources, as each string needed a data and an index buffer assigned.

NSR was made for batch processing. Its use should be limited in an on-line CICS environment to handle the two cases mentioned above. Otherwise, the file should be allocated into LSR. There is one final problem that should be reviewed with placing files into NSR. One of the new storage protection features available in CICS since CICS/ESA 4.1 is Transaction Isolation. This feature is not widely used but represents an important step towards controlling errant storage violations that can affect system integrity. We recommend its use in spite of the overhead associated with Transaction Isolation. Transaction Isolation does not support NSR files. Therefore, if you plan to use Transaction Isolation, then the files have to be defined into LSR. Review all your files and determine the need for any NSR file. If there is no justifiable reason, then convert the file to LSR and receive better performance due to the better look-aside algorithm used by LSR.

If you have any questions, comments or request for more information, please contact us:

Address:	C\TREK Corporation PO BOX 560069 MONTVERDE FL 34756	Phone:	(407) 469-3600 (787) 756-5620	C\TREK Corporation Advanced Computer Technology
E-mail:	ctrek@actpr.com	Fax:	(787) 756-5150	
Website:	www.ctrekcorp.com	Support::	(787) 397-4150 (321) 297-5838 (787) 462-0406	