Cheryl Watson's

January 1991

# TUNING Letter

A PRACTICAL JOURNAL OF z/OS TUNING AND MEASUREMENT ADVICE

*Editor's note: This January 1991 issue is the first newsletter we ever published. Back then, it was a monthly newsletter. Today the TUNING Letter is published six times a year. Although they vary in size, the average issue is now 44 pages.*

*This issue has been reformatted for Web publication because the information is still relevant - eleven years later! We hope you find it useful.*

## *THIS ISSUE*

FOCUS: TUNING TSO
- Reducing ISPF Response Time 2
- Eliminating TSO STEPLIBs 4
- Reducing Output Wait Swaps 6
- Reducing TSO Period Swaps 9
- Invoking CLISTs Faster 12
- Reducing CLIST Response Time 13
- Providing Consistent Response 14

MVS REVIEW
- MVS Program Fetch 16

MVS MEASUREMENTS
- Measuring TSO Response Time 17
- Measuring TSO Command Use 18

Q & A 19

MISCELLANY 19

## *WELCOME!*

Welcome to the first issue of **TUNING LETTER**! This monthly newsletter is a natural outgrowth of my MVS classes, articles, and over twenty four years in IBM mainframe software and tuning. People *do* want to know how to tune MVS. Not everyone (especially me) concurs with vendors who say "It's simply too expensive to tune. It's cheaper to upgrade and buy more hardware". In reality, it's seldom cheaper to buy more hardware!

This newsletter is written for the systems programmer or performance analyst who wants to **1) extend the life of their current system, 2) provide better service to their users and/or 3) provide room for growth without the cost of upgrading their hardware.** Or, if they've already ordered an upgrade, it will show how to tune their systems to last until the upgrade is installed. Each tuning tip will provide one of these four benefits, saving money and/or increasing system responsiveness. Capacity planners and charge back analysts will benefit from the emphasis on MVS measurement theory and practice.

Each issue will focus on one or two aspects of MVS by providing several practical techniques to improve performance. For example, this month's issue provides seven different techniques for tuning TSO subsystems. Most articles will be applicable to both MVS/XA and MVS/ESA systems, and I will indicate if there's a difference between the two.

Each tuning tip is described in multiple sections: 1) an overview describes the intent of the change and its applicability. 2) WHAT TO DO is a step-by-step checklist of how to make the change. 3) BENEFITS describes the positive sides of the change. 4) COSTS tells you what resource(s) you're trading off! (Very few items come free: for example, you might give up some storage to reduce CPU time or give up a little CPU time to reduce I/Os). 5) HOW TO MEASURE provides techniques for determining whether you've made an improvement. Naturally, you'll always want to measure before and after a change to gauge its effectiveness for you. All in all, this will be a very practical guide to tuning your system and saving money.

# FOCUS: TUNING TSO

Most installations see at least a 30% increase in TSO CPU use each year. This rate of growth appears to be continuing or even increasing as more TSO applications and 4GL products are installed. Tuning TSO subsystems can often provide a reduction in TSO response time of 50% or more.

This month's focus provides seven ways to improve or adjust TSO response time and reduce its resource consumption. The impact of any change will depend on the installation. The techniques are described in descending order of potential savings for a typical TSO site.

## Reducing ISPF Response Time

You can reduce ISPF response time, average TSO swap size, and central storage use by moving ISPF modules to LPA (Link Pack Area) from their default location in link list. The majority of TSO commands are ISPF commands, and would be found much faster in LPA than in link list. See MVS PROGRAM FETCH in this newsletter. Consequently, moving ISPF modules to LPA reduces response time.

In addition, modules found in link list are loaded into the user private region and are liable for swap out and swap in. Because they're in the private region and not common storage, you might actually have multiple copies of the same module in central storage. Consequently, leaving ISPF modules in link list results in additional CPU time and I/O for loading, and larger working set sizes (WSS) for TSO users. Larger WSS result in larger swap sets and therefore more pages are moved during a swap. If logical swap is in effect, this larger WSS uses more central storage. Even if logical swap is not used, the resulting physical swap to auxiliary storage requires more I/O. Hence, moving ISPF modules to PLPA is an excellent tuning step for any ISPF shop.

### WHAT TO DO

The solution is simple - place reentrant ISPF modules into link pack area. Here's a recommended checklist:

1. Test the change (under an LPAR or stand-alone environment):

- ☐ Measure TSO response time, average swap size, LPA size in virtual storage, and activity on the load ISPF libraries. See the HOW TO MEASURE below.
- ☐ Make a backup of members LPALSTxx and LNKLSTxx.
- ☐ Place all reentrant ISPF modules into a separate library.
- ☐ Add the library name to LPALSTxx.
- ☐ IPL with CLPA (Clear Link Pack Area). This causes the new LPALSTxx to be used during IPL.
- ☐ Measure the same indicators as above.

2. Install in production:

- ☐ Move the changes to the production system.
- ☐ Remove the reentrant modules from the link list library (or simply remove the library from the LNKLSTxx member.

## BENEFITS

This change will result in shorter response times, less CPU time, fewer I/Os to link list, less use of central storage, and a smaller TSO swap size.

## COSTS

This change will require more virtual storage, primarily above the 16MB line, although some modules might be below the 16MB line. It requires an IPL. It will slightly increase LPA search time for all other users since the LPA incore directory is slightly longer. (This is minimal).

## HOW TO MEASURE

This is a tough one to measure since measurements will have to done before and after an IPL and the differing workloads may alter the results. You should analyze TSO response time before and after the change. See MEASURING TSO RESPONSE TIME at the end of this newsletter. This could be done for one TSO user issuing a variety of ISPF commands or by simply looking at the average response time for all TSO users. You'll also want to look at virtual storage usage, swap sizes and activity on the ISPF libraries. Figure 1 shows a portion of an RMF Virtual Storage Activity report. Before and after the change, look at the amount of LPA above (EPLPA) and below (PLPA) the line, and the PRIVATE region below the line. You can expect to see the EPLPA and PLPA increase, while the PRIVATE area might decrease.

```
      STATIC STORAGE MAP
  AREA          ADDRESS   SIZE
 EPVT           2E00000  2002M
 ECSA           1EB5000  15.3M
 EMLPA          1EAA000    44K
 EFLPA                0     0K
 EPLPA          1AB8000  4040K
 ESQA           1268000  8512K
 ENUC           1000000  2462K
 ----- 16 MEG BOUNDARY ------
 NUCLEUS         F96000   423K
 SQA             EA6000   960K
 PLPA            CEC000  1768K
 FLPA                 0     0K
 MLPA            CEA000     8K
 CSA             600000  7080K
 PRIVATE           1000  6140K
 PSA                  0     4K
```

**Figure 1 - RMF Virtual Storage Activity Report**

```
                                   W O R K L O A D   A C T I V I T Y

          OS/VS2                    SYSTEM ID SYSA             START 08/16/89-10.00.00  INTERVAL 001.00.00
          SP2.2.0                   RPT VERSION 3.5.0          END   08/16/89-11.00.00  IPS = IEAIPS00

       OPT = IEAOPT00                              REPORT BY PERFORMANCE GROUP    SERVICE DEFINITION COEFFICIENTS     SU/SEC= 700.0
       ICS = IEAICS00                                         PERIOD              IOC =   5.0  CPU = 10.0  MSO =   3.0  SRB = 10.0

**** PERFORMANCE ***** DOMAIN  TIME   INTERVAL SERVICE  AVERAGE ABSORPTION,    PAGE     STORAGE   AVERAGE   ENDED     AVG TRANS
GROUP    GROUP  OBJTVE NUMBER  SLICE  (TOTAL, BY TYPE   AVG TRX SERV RATE,      IN      AVERAGE,  TRANS,    TRANS,    TIME, STD DEV
NUMBER   PERIOD NUMBER         GROUP  AND PER SECOND)   WORKLOAD LEVEL         RATE     TOTAL     MPL       #SWAPS    HHH.MM.SS.TTT

SUBSYS = TSO          TRXCLASS =             ACCTINFO = NO
USERID =               TRXNAME =

0002          1    11     003     **    IOC=   605,069   ABSRPTN =   1,445      5.80      44856.        1.49    18230    000.00.00.156
                                        CPU= 4,292,360   TRX SERV=   1,247
                                        MSO=   185,918   WKLD LEV=    3.21                57968.        1.29    18063    000.00.01.720
                                        SRB=   639,476
                                        TOT= 5,722,823
                                        PER SEC=   1,590

0002          2    11     023     **    IOC= 1,209,703   ABSRPTN =   1,433      2.94      8501.0        1.48     3148    000.00.01.235
                                        CPU= 5,833,925   TRX SERV=   1,419
                                        MSO=   230,108   WKLD LEV=     .98                12538.        1.47     3366    000.00.02.277
                                        SRB=   334,691
                                        TOT= 7,608,427
                                        PER SEC=   2,113

0002          3    11     004     **    IOC= 7,949,566   ABSRPTN =   2,370       .52      253.32        6.61     2269    000.00.12.480
                                        CPU= 42.190407M  TRX SERV=   2,286
                                        MSO= 2,727,881   WKLD LEV=   64.78                1616.8        6.38     5064    000.00.56.810
                                        SRB= 1,569,228
                                        TOT= 54.437082M
                                        PER SEC=  15,121

0002          ALL  ALL    ALL     ALL   IOC= 9,764,338   ABSRPTN =   2,088      1.65      7882.5        9.60    23647    000.00.01.482
                                        CPU= 53.316692M  TRX SERV=   1,989
                                        MSO= 2,143,907   WKLD LEV=   51.71                72123.        9.14    26493    000.00.18.040
                                        SRB= 2,543,395
                                        TOT= 67.768332M
                                        PER SEC=  18,825
```

**Figure 2 - RMF Workload Activity Report**

```
F                         MIG=319 CPU= 97 UIC= 38 PDT=  63 DPR= 39   ASD      T

09:03:41       P P C  R  DP  RS  ESF  ESF   TAR   WS   TX  WRK  CPU  I/O STM
JOBNAME  DMN   G P L  LS PR  F         +RS   WSS   IN   SC  RV   RV   RV  RV
TSOUS13    3   2 1 WT TI FF    0  234  234     0   31    0   15   +0   +0   +0
TSOUS27    4   2 3 IN    68  115  141  256     0   47    0   10   +0   +0   +0
TSOUS05    4   2 3 IN    62   95  406  501     0   38    0   10   +0   +0   +0
```

**Figure 3 - RMF Monitor II ASD Screen**

The average swap size can be found in two places. The average swap size for all swaps (TSO as well as non-TSO) can be found in the RMF Monitor I Swap Placement Activity report shown in Figure 6. The second line from the bottom shows the average swap size, "AVERAGE PAGES PER SWAP IN - 102". Moving ISPF modules to LPA should reduce this number, particularly where the largest percent of swaps is for TSO.

The RMF Monitor I Workload Activity report in Figure 2 shows the average number of central and expanded storage frames for users within a performance group (under the column STORAGE AVERAGE). This figure should also decrease. You can get detailed information on each TSO user with RMF Monitor II data (or any other online monitor) as shown in Figure 3. The ASD (Address Space State data) includes the WS IN field which shows the average working set size (in pages) for swapping. Implementation of this change should reduce both the WS IN, as well as the number of real storage frames (RS F) and expanded storage frames (ESF). In MVS/ESA, real storage is referred to as central storage and the column is titled CS F.

Don't forget to look at the device activity to the ISPF load library devices (RMF type 74 - see Figure 4). You will see a decrease in the activity rate (SSCHs per second) and the average DASD response time for the volume(s).

This sounds like a lot of trouble, but it's only looking at four reports and you should always know the impact of any change. You'll definitely see improvement in at least one of them and probably more.

```
                   DEVICE    AVG   AVG   AVG   AVG    AVG   AVG   AVG   ...  AVG
DEV  VOLUME  LCU   ACTIVITY  RESP  IOSQ  CUB   DB     PEND  DISC  CONN  ...  DS
NUM  SERIAL        RATE      TIME  TIME  DELAY DELAY  TIME  TIME  TIME  ...  OPEN

100  DSK010  00D    0.013     11     0   0.0    0.1    0.1   8.9   1.9  ...   2.6
101  DSK102  00D    4.200     18     2   0.0    0.1    0.1  12.2   3.7  ...  23.4
102  DSK203  00D   21.507     73    39   0.0    0.1    0.2  25.6   7.6  ...   124
103  DSK024  00D   23.146     85    51   0.0    0.1    0.2  25.3   8.6  ...   124
```

**Figure 4 - RMF Monitor I DASD Device Activity Report**

## Eliminating TSO STEPLIBS

TSO STEPLIBs cause a performance problem that can easily be eliminated. TSO STEPLIBs use CPU cycles and I/O to search the STEPLIB directories for every command. This can cause high contention on the STEPLIB DASD volumes. While there may be some very good occasions for using STEPLIBs, you can reduce TSO response time and DASD response time by eliminating them wherever possible. (Don't worry, there are some alternatives.) This change is applicable to TSO sites where most user LOGON procs contain STEPLIBs. To see if STEPLIBs are used in your current LOGON proc, you can issue a TSO command, "LISTA ST", which displays the DDnames allocated. If you see the DDname STEPLIB, you're using them!

To appreciate the performance problem with TSO STEPLIBs, first review MVS PROGRAM FETCH in this issue, which describes the order of search for programs and TSO commands. Since program fetch is invoked each time you enter a TSO command, it will search the STEPLIB libraries every time you hit the enter key! That's several thousand times a day. If you don't believe me, take a look at the activity on the DASD device that contains the TSO STEPLIBs. They will have very high activity and typically long response times (well over 30ms) due to high IOSQ times and high connect times. You'll also see high disconnect times if there are several directories on a single volume because of seek time. Figure 4 shows a sample RMF Device Activity report. Devices 102 and 103 in Figure 4 are examples of DASD devices that contain all of the TSO STEPLIBs in one site. Notice the high DASD response time (twice the desired 20 or 30ms). **In fact, TSO STEPLIBs create some of the worst queueing (and therefore poorest response time) that you'll see!** In a shared DASD system with TSO users on different systems, the RESERVE on the volumes is extremely high.

But you say that you can't live without STEPLIBs? Oh yes you can. Most TSO users don't need STEPLIBs. The majority of commands are found in LPA (Link Pack Area) or link list. See the previous article for another tip on the use of LPA. Some sysprogs use STEPLIBs for testing new products or modules. That's just fine, but you don't need it all the time.

## WHAT TO DO

Here's a checklist to test and implement this change. Two alternatives are reviewed.

1. Test STEPLIB Elimination:

☐ Create a backup of the current LNKLSTxx member in SYS1.PARMLIB.
☐ Measure TSO response time and swap sizes. See HOW TO MEASURE below.
☐ Add the STEPLIB dataset names to LNKLSTxx. This adds to the SYS1.LINKLIB concatenation and will only take effect after an IPL.
☐ IPL.
☐ Create a second LOGON proc for one user without a STEPLIB.
☐ Update SYS1.UADS to allow a second LOGON proc for that user.
☐ Let that one user LOGON to TSO using the second LOGON proc. Test. Try all common functions, features and applications that are normally used by your TSO community.
☐ Measure TSO response time and average swap size for the one user to see the result of the change.

2. Install in Production:

☐ If all is well, notify the other TSO users that you are making a change and give them a contact number.
☐ Start updating the TSO LOGON procs by removing the STEPLIBs. You can do this gradually if you'd prefer.
☐ Measure TSO response time and average swap size after the change.

3. Alternative #1: Multiple LOGON procs.

For the very few sysprogs who do need a STEPLIB for testing, create two LOGON procs, one with STEPLIBs and one without STEPLIBs. When you need STEPLIBs, LOGON with the STEPLIB proc. Easy as pie! Of course, this does require maintaining two LOGON procs. See the testing checklist above.

4. Alternative #2: Dynamic STEPLIBs.

Use a program to dynamically allocate STEPLIBs or TASKLIBs as needed. There are several tools available to do this on the CBT MODs tape [REF001]. For testing this, use the same basic procedure described above.

*[Editor's Note: An excellent example of such a program can be found in our TUNING Letter 2000, No. 6. Our focus article is by* ***Tom Conley****, Pinnacle Consulting Group, on his solution for Dynamic ISPF.]*

### BENEFITS

Removing STEPLIBs will result in shorter response times, less I/Os, and less contention on the STEPLIB volumes, which reduce delays to other users of the volume.

### COSTS

1) Some users may see higher CPU times, although most users will see less. 2) Maintaining two LOGON procs can be confusing - changes made to one might be forgotten in the other. 3) It takes more effort to logoff and logon to get to a special STEPLIB.

### HOW TO MEASURE

When measuring the change to a single user, SMF type 30, subtype 5 records are extremely useful. Compare the following fields before and after the STEPLIB change: SMF30CPT (TCB CPU time), SMF30CPS (SRB CPU time), SMF30TEP (total EXCPs), SMF30PSI (number of pages swapped in), SMF30PRV (virtual storage at the bottom of the region), and SMF30SYS (virtual storage at the top of the region). If most modules are found in PLPA, you'll see a reduction in all of the fields. If the modules are found in link list, then you'll mainly see a change in the amount of CPU time, EXCPs and response time.

Review the DEVICE ACTIVITY RATE and response time (AVG RESP TIME) on the device(s) referenced by the STEPLIB. See Figure 4 for an example. This data is recorded in the RMF type 74 record. If all STEPLIBs are removed, compare the device activity rate on the STEPLIB volumes from RMF data and you'll see significant reductions and, of course, a corresponding improvement in DASD response time.

### MVS/ESA

MVS/ESA provides considerable benefits through the use of VLF (Virtual Lookaside Facility) and LLA (Library Lookaside). Through these facilities, you can direct the STEPLIB directories to be maintained in virtual storage in a data space. LLA defines which libraries are controlled and VLF provides the service to search them. You can specify that the libraries are "frozen" (infrequently changed), and if so, all directory searches are done in virtual storage instead of direct access. Obviously, access to virtual storage could result in paging I/O. If all libraries are in an LLA data space, you'll see no DASD activity due to STEPLIB directory search, but there will be CPU time to search the directories maintained in the data space by LLA. Use of LLA does not change the location of the modules. Use of the STEPLIB even with LLA will result in modules being loaded into the user's private region, thus using more central storage. The primary benefit of VLF is the reduction of I/Os to read the directory. Use of VLF will be addressed in a future issue.

## Reducing Output Wait Swaps

Increasing the buffersize for TSO screen buffers can eliminate unnecessary swapping due to TSO output waits. This article describes an IBM default that's at least ten years old and now results in unnecessary swaps. Defaults are tricky. They may have been valid at one time, but often don't apply to the current technology.

Look at SYS1.PARMLIB(TSOKEY00) in Figure 5.

```
USERMAX=40,RECONLIM=3,
BUFRSIZE=132,
HIBFREXT=48000,
LOBFREXT=24000,
CHNLEN=4,SCRSIZE=480,MODE=NOBREAK,
MODESW=NO,CONFTXT=YES
```

**Figure 5 - Default TSOKEY00 Parameters**

This member contains the start up parameters for TSO each day and defines the buffersize and bufferspace for TSO to use when transferring data between the host and the terminal screen. The parameters are documented in the IBM "Init & Tuning Guide" [REF003]. The defaults were designed for TSO "line mode". You can use line mode by coding EDIT (or E) at TSO READY or ISPF Option 6, but few people still use this technique. Most users operate in a full screen mode, such as in ISPF.

Here's how TSOKEY00 works. TSO provides permanent buffers of the size specified in the BUFRSIZE parameter. The default size is 132 byte buffers. If the screen size is greater than 132 bytes, TSO will issue a getmain to obtain space for the larger buffer. It will continue to dynamically getmain virtual storage until it has obtained a maximum, as specified by parameter HIBFREXT (default 48,000 bytes). When it reaches this maximum, TSO will swap the user out until the dynamically allocated amount drops to the amount specified in LOBFREXT (default 24,000 bytes). This is counted as a "Terminal Output Wait Swap" by SRM (Systems Resources Manager). The user is then swapped back in when LOBFREXT is reached and the transaction completes. SRM treats this swap in as a new transaction and records it as a new completed transaction.

The major problem is in the default BUFRSIZE of 132. Most installations run ISPF or some other full screen application and the average screen size is closer to 2048 bytes than 132. Therefore, the majority of transactions are incurring the overhead of unnecessary getmains and some are incurring the delay of an output wait swap.

You can find the number of swaps by looking at any RMF Swap Placement Activity report (look at page 2 of an MVS/XA report or page 3 of an MVS/ESA report). Figure 6 shows a sample. The total number of TSO input *and* output wait swaps is shown in the upper left, 381,599. Terminal input wait swaps occur when TSO has sent a screen to a user and the address space is swapped out waiting for input from the terminal. Terminal output waits occur when TSO cannot find enough bufferspace and the address space is swapped out. The total number of Terminal Output Wait swaps is buried in the report in a single line at the bottom of the page, 35703. A good value to see there is zero. After all, the only cost is in virtual storage.

## WHAT TO DO

This is a fairly simple and risk-free change. Here's a suggested checklist:

- ☐ Determine the parameter member. It's possible that you may not be using TSOKEY00. Look at your start of TSO after IPL ("S TSO"). If it's started with a MEMBER or MBR parameter (or has a member coded on the PARMLIB DD), you may be using a different member name.
- ☐ After determining the correct member, make a backup!
- ☐ Collect measurement data on CPU time and TSO output wait swaps. See HOW TO MEASURE below.
- ☐ Change BUFRSIZE to 2048 which is the maximum for Model 2 terminals. You might want to use a larger value if you have applications that use extended data streams or windowing services. A larger value can also be used if you have many Model 3, Model 4 or Model 5 terminals. A Model 5 terminal has a screen size of 27x132 (or 3564 bytes). If you only have a few terminals with the larger sizes, you should at least increase the HIBFREXT to 96000. The next time that TSO is started, the changes will take effect.
- ☐ If no TSO users are on the system, simply stop TSO and restart it.
- ☐ Measure the key indicators once more.

## BENEFITS

This change reduces unnecessary CPU time for getmains for any screen over 132 bytes. It also eliminates unnecessary swaps (some installations have 5,000 terminal output swaps per hour), which in turn reduces TSO average response time, CPU time for swapping and I/O time for swapping. This reduction will not occur in all installations. Exceptions will be covered next month.

## COSTS

This takes additional virtual storage in each TSO address space.

## HOW TO MEASURE

Just look at the RMF Swap Placement Activity report (Figure 6) before the change and after the change. You should see an elimination or major reduction of the number of terminal output wait swaps. A reasonable goal is to eliminate all output wait swaps. If the number of output wait swaps was fairly large, you'll also see a decrease in the number of ended transactions because SRM counts a transaction with an output wait as two transactions. This would lead to a reported increase in response time simply because the average response time is based on the number of transactions. In reality the improved response time for the end users will definitely be improved. You'll also see decreased CPU time per transaction for any transaction with screen sizes over 132 bytes due to the elimination of getmains.

```
 OS/VS2                    SYSTEM ID SYSA              START 08/16/88-08.14.02  INTERVAL 007.59.50
 SP2.2.0                   RPT VERSION 3.5.0           END   08/16/88-15.59.01  CYCLE 1.000 SECONDS

 OPT = IEAOPT01                             S W A P   P L A C E M E N T   A C T I V I T Y

                             *------ AUX STORAGE -------* *---LOGICAL SWAP--* *--EXPANDED STORAGE--*

                             AUX               AUX STOR
                             STOR     AUX STOR   VIA                 LOG SWAP
                    TOTAL    TOTAL    DIRECT   TRANSITION LOG SWAP  EFFECTIVE

TERMINAL      CT  381,599   13,011     7,518      5,493  366,584    326,598
INPUT/OUTPUT  RT    13.25     0.45      0.26       0.19    12.73      11.34
WAIT           %    86.0%     3.4%     57.8%      42.2%    96.1%      89.1%

LONG          CT    7,194       18         2         16    6,498      5,432
WAIT          RT     0.25     0.00      0.00       0.00     0.23       0.19
               %     1.6%     0.3%     11.1%      88.9%    90.3%      83.6%

DETECTED      CT   10,726      420        22        398   10,058      7,572
WAIT          RT     0.37     0.01      0.00       0.01     0.35       0.26
               %     2.4%     3.9%      5.2%      94.8%    93.8%      75.3%

UNILATERAL    CT   41,958      370       142        228   38,541     34,013
              RT     1.46     0.01      0.00       0.01     1.34       1.18
               %     9.5%     0.9%     38.4%      61.6%    91.9%      88.3%

EXCHANGE ON   CT    1,975       18         1         17    1,875      1,492
RECOMMENDA-   RT     0.07     0.00      0.00       0.00     0.07       0.05
TION VALUE     %     0.4%     0.9%      5.6%      94.4%    94.9%      79.6%

ENQUEUE       CT      275        4         3          1      243        201
EXCHANGE      RT     0.01     0.00      0.00       0.00     0.01       0.01
               %     0.1%     1.5%     75.0%      25.0%    88.4%      82.7%

TRANSITION    CT      120      119       119          0        1          1
TO NON-       RT     0.00     0.00      0.00       0.00     0.00       0.00
SWAPPABLE      %     0.0%    99.2%    100.0%       0.0%     0.8%     100.0%

TOTAL         CT  443,847   13,960     7,807      6,153  423,800    375,309
              RT    15.42     0.48      0.27       0.21    14.72      13.04
               %   100.0%     3.1%     55.9%      44.1%    95.5%      88.6%

AUXILIARY STORAGE - AVERAGE PAGES PER SWAP OUT - 65    AVERAGE PAGES PER SWAP IN - 102
OCCURRANCES OF TERMINAL OUTPUT WAIT -    35,703
```

**Figure 6 - RMF Monitor I Swap Placement Activity**

*[Editor's Note - The following notes on this topic were included in an update sheet after the issue was printed:*

*Most of the calls I received were about reducing TSO terminal output wait swaps. There have been dozens of calls from people who said that they eliminated or greatly reduced the number of output wait swaps. One site saved 18,000 output wait swaps per hour! But there have been a few calls from people who saw little or no reduction.*

*Applications that transfer data between the host and a PC through TSO can exceed the buffer length. Another possibility might be graphic applications, such as GDDM or SAS/GRAPH. An IBM document indicated that a terminal output wait swap occurred whenever you see the three asterisks at the bottom of your screen indicating more data to be displayed. This doesn't appear to be true, since in tests these users all appeared to be in a terminal input wait. The document indicated that this will occur with line-mode commands such as LISTCAT. Another items mentioned by some users is that their average TSO working set size increased by 4 to 12K. Others saw no increase.*

*There are two ways to track this down if you don't see a reduction. Sites that have TSO/MON (now from CA) can look at the message sizes from TSO sessions and might be able to identify transactions that result in extremely large buffers. If you don't have TSO/MON, you can try this, but realize that it takes lots of patience (!): turn on RMF Monitor II (RMFMON), enter the ASD screen with "ASD T,A,A" which will display swapped in and swapped out TSO users, look for "TO" in the R LS column (Reason for Last Swap), identify those users, and go ask them what they're doing! Any online monitor that shows the reason for the swap out can be used. It's very, very, difficult to find anyone in an output wait swap, but you might be lucky enough to find one. Keep the interval examined as small as possible, such as one second.*

*If you specify a BUFRSIZE of greater than 3016, TSO will default to its original value of 132. See APARs* ***OZ95253***, ***OZ95252***, ***OZ93234***, *and* ***OY16367*** *that addressed some abends and LOGON-failed problems. The solution of the APARs was to set the default to 132 if a BUFRSIZE greater than 3016 was specified. The manuals will be changed to reflect the new maximum.*

*I've been trying to collect information on sites that aren't successful in reducing output wait swaps. A very large proportion of these sites are running FOCUS or ADABAS applications.*

***Guy Albertelli*** *from B.F. Goodrich and* ***Dan Gillis*** *from Ameritech Services have identified more instances of output wait swaps. A LOGON to TSO causes an O/W swap, as does entering and leaving ISPF. A TSO SUBMIT produces an O/W swap. Any application that uses a TPUT HOLD fullscreen will cause an O/W swap.]*

## Reducing TSO Period Swaps

By removing the ISV parameter from all but the last period of TSO performance groups, you can eliminate unnecessary exchange swaps due to inappropriate use of this parameter by SRM. ISV (Interval Service Value) was originally created to reduce swapping. To fully understand this change requires more details on SRM internals than I can cover in one issue. But the change is so simple that you can easily take it on faith and test to see if it works.

First, see if this suggestion applies to you by looking at your SYS1.PARMLIB members. Look at Figure 2 which is a sample Workload Activity report. It contains the names of the current ICS and IPS members, at the upper left and upper right, respectively.

```
SUBSYS=TSO,PGN=2
  USERID=DXX(1),PGN=21
  USERID=TXX(1),PGN=22
  TRXNAME=EXEC,RPGN=127
```

**Figure 7 - SRM ICS Parameters**

You first need to identify the performance groups used by TSO. You might be using the TSO default group of 2 or you may have one or more TSO performance groups. To find your installation's defined TSO performance groups, look at the ICS member (IEAICSxx). A sample ICS is shown in Figure 7. Look for the PGN parameters on the SUBSYS=TSO statement and all statements below it down to the next SUBSYS statement. The sample in Figure 7 shows that TSO users are controlled in performance groups 2, 21 and 22. For this example, you can ignore all references to RPGN.

```
Old Use of ISV:

  PGN=2,(...DUR=500,ISV=500...)
        (...DUR=1000,ISV=1500...)
        (...ISV=50K...)

Recommended Use of ISV:

  PGN=2,(...DUR=500...)
        (...DUR=1000...)
        (...ISV=50K...)
```

**Figure 8 - Sample TSO ISV Parameters**

Now look at the sample IPS in Figure 8, as well as your own IPS. An old suggestion to reduce swaps was to code an ISV equal to the sum of the durations of the current and prior periods. This is a little hard to explain without a ten-page discussion on how SRM works, but here goes!

The DUR (duration) parameter on a performance group defines the length of time that a transaction lives in one period. Typically, the first period or two of a TSO transaction's life are given a high dispatch priority in order to get short transactions through the system quickly.

As the transaction continues, it moves to the next period where the priority is slightly lower. The length of time specified by the DUR is actually given in total service units (a combination of CPU time, I/Os, and storage use adjusted by the machine speed). More about service units in a later issue.

ISV is the number of service units since the last swap in during which you will be protected against a swap out. Until you have accumulated the amount of service units specified by ISV, swapping is inhibited, except in extreme emergencies. The old theory was to keep ISV at the sum of the durations, so you did not swap during first and second period TSO. The reason for this is that it takes less service units to complete a period than it takes to swap a user - just get the job done!.

Here's where the theory went awry. Let's say that the TSO user in the performance group in Figure 8 had accumulated 450 service units when SRM pops in to check on its status. SRM checks to see if it's time to move to the next period every SRM second (about 20 times a real second on a 3090). Since the user had not accumulated 500 service units (specified by DUR=500), SRM leaves the user in first period. During the next second, it would be possible for that user to consume another 150 service units. Now when SRM checks it on the next pass, the transaction has accumulated 600 service units and the user gets moved to second period. The DUR on period 2 indicates that the user will stay in period 2 for at least 1000 service units. But look at what happens to ISV. After consuming another 910 service units during second period, it's consumed a total of 1510 service units since swap in. ISV no longer protects it since the sum since swap in is greater than the ISV value (ISV=1500). The transaction becomes available for a swap out. This may cause an exchange swap. This will cause no problem unless you have other users who are swapped out and ready.

TSO users should never be swapped out in first or second period (since it takes more time to swap than complete). To avoid unnecessary swaps, simply remove the ISV parameter from all but the last period of TSO. The default ISV is 100K. The DUR will really control how long the user is in each period, so there is no need to have an ISV.

## WHAT TO DO

Here's a checklist for implementing this change:

☐ Collect measurements for TSO response time, number of swaps and total number of exchange swaps. See HOW TO MEASURE below.
☐ Create a test version of the IPS (e.g. IEAIPS99).
☐ Remove the ISV from all but the last TSO performance group.
☐ Reset the IPS. You can do this with an operator command of "T IPS=xx" or simply wait for the next IPL. If you change it in the middle of the day, RMF will cut records for the interval and you'll get multiple RMF records. SRM will also reset many of its counters at this time. Many sites try to avoid resetting the IPS in the middle of the day.
☐ Measure the indicators again.

This same technique should be applied to all performance groups, not just TSO. In fact, I seldom code the ISV parameter. The default of 100K tends to reduce the number of exchange swaps.

## BENEFITS

This change reduces the number of exchange swaps in second period (and third period, if duration is coded for third period) if you have other users who are swapped out and ready. Therefore it could reduce response time, and CPU and I/O due to swapping.

## COSTS

Just the time to change it. Isn't that neat?

## HOW TO MEASURE

Monitor your RMF Swap Placement Activity report, Figure 6, to see if you are doing exchange swaps. If not, then this change will not have an affect on your system. If you have exchange swaps, there's no way to determine if these are TSO users or batch jobs. You can review the RMF Workload report in Figure 2 to get the average number of swaps in second and third period TSO (such as 3366 swaps in period 2). If you have been doing unnecessary swapping, you would expect to see a reduction in the number of total exchange swaps and the number of swaps in TSO second and third period (or whichever periods other than the first one have an ISV).

*[Editor's Note - The following item was added as an update to this newsletter issue:*

*One reader called with this warning. He tried increasing the ISV on third period TSO and ended up with worse response time! This may happen in other shops, so I want to explain what might occur. Here's an example with a 'before' and 'after' set of IPS parameters:*

```
BEFORE:    PGN=2,(...DUR=800,ISV=800...)
             (...DUR=1000,ISV=1800...)
             (...ISV=10K...)

AFTER:     PGN=2,(...DUR=800...)
             (...DUR=1000...)
             (...ISV=50K...)
```

*Let's assume that there are some very large resource consumers in third period that might take 100,000 service unit (I'll call them BIGGIES). Also assume that some of the transactions will only take 5,000 service units (they're SHORTIES). Remember that ISV only comes into play if you are doing exchange or unilateral swapping. Now as-*

*sume that a third period BIGGIE is swapped in. In the AFTER example, it will stay swapped in for 50,000 service units before getting a lower workload level and allowing a SHORTIE to get swapped in, execute for 5,000 service units, and complete. The response time for the SHORTIE is increased by the time it takes the BIGGIE to consume 50,000 service units. If the ISV had been 10K, SHORTIE would have been delayed for a shorter period of time, resulting in shorter response times. The moral of this story is that for third (or the last) period TSO users, you'll want some ISV value that will allow most of your 'shorter' third period users to complete without causing too much delay to the other users. You can determine the average number of service units consumed in third period by dividing the TOTAL service units consumed in third period by the number of ENDED TRANSACTIONS in third period. In Figure 1, the average number of service units in third period is: 67,768,332 / 23,647 = 2866 service units. For this site, you could set the third period ISV to 5K or 10K to reduce the effect of a very large transaction delaying the others. This will only have an effect in sites that are experiencing exchange or unilateral swaps. The advice to remove the ISV from the first two periods still holds true.]*

## Invoking CLISTs Faster

Here's a quick way to save both CPU cycles and I/O while reducing response time with no cost except to publish a memo to TSO users. What a deal! Simply tell users to use a percent sign, "%", on the front of any TSO CLIST that they enter. It will reduce the search time for the CLIST and provide better response.

To better understand this concept, please review MVS PROGRAM FETCH in this issue.

### WHAT TO DO

Here's the trick! If you enter the CLIST name preceded by a percent sign, such as "%myclist", program fetch will immediately skip to step 8 in Figure 11 to search SYSPROC. Look at the processing that you will have saved! By the way, CLISTs executed within CLISTs can also be improved with this technique. Simply use a "%" in front of the CLIST name each time you use it.

### BENEFITS

This technique is a "win-win" proposition! It saves CPU cycles, I/O accesses and reduces response time.

### COSTS

The main problem is that you have to write a MEMO (the bane of sysprogs everywhere)! One additional problem may occur in the future. At some point you may want to rewrite a CLIST as a command processor (see MEASURING TSO COMMAND USE later in this issue). If you do this, the users will have to remove the "%" sign when accessing the CLIST.

### HOW TO MEASURE

So how much will you save? It depends on your environment. Do you have LLA installed? How big is link list? How big is link pack? Do you use STEPLIBs? Here's one technique for determining how much you can save. Have a CLIST ready to try. Get into TSO ready mode. Enter the TIME command. It displays the total service units during your TSO session. Enter your CLIST (without "%") followed by the TIME command again. The difference in service units is the amount of service to execute the CLIST and the TIME command. Do this several times to get an average amount of service units for the CLIST. Now follow the same procedure, but use "%" in front of the CLIST name. Again, do this several times to get a good average number of service units for the CLIST. The difference between the two averages is the average service saved by using "%".

Here's another measurement technique that takes a little extra work but provides even more information. Have two adjacent TSO terminals. Bring up RMF Monitor II (or any other real-time monitor that displays CPU time or service units). Logon to a second TSO session on the other terminal. For this example, we'll assume the second TSO userid is TSOUSR1. Now you're ready to start measuring. Under RMF Monitor II, enter "ASRMJ TSOUSR1". A sample is shown in Figure 9. Under TSOUSR1, enter a CLIST that can be found in your SYSPROC, such as: "my-clist". Hit enter on the RMF screen. The service units shown on the RMF screen were used to find the CLIST and execute it. Repeat this sequence a few times to collect multiple measurements. Look at the CPU and I/O service units. Now enter "%myclist" and repeat the same measurements. The difference between the two measurements will be service used for the program fetch. Now you have the amount of CPU time and EXCPs saved for each CLIST. To determine your total savings, you'll need to estimate the number of CLISTs executed each day.

## Reducing CLIST Response Time

*[Editor's Note: In the original issue of this newsletter, I described a technique to improve response time at the cost of CPU and detriment of other users. I said that you might want to specify CNTCLST=YES instead of the default and recommended CNTCLST=NO. Even though I included this warning: "This isn't a change that I can recommend," several readers did try it. I'm sorry that I included it, since I strongly think you should keep the default. Because of this, I've eliminated the original article from this republished issue.]*

Here's the disadvantage of changing CNTCLST=YES. It increases the reported number of TSO transactions (since each command in a CLIST is treated as a transaction). If installations have been tracking the number of TSO transactions for the last year, you'll see a dramatic increase in the number of transactions for exactly the same workload. It really changes the definition of a transaction. It also decreases the *reported* average response time. The average response time for normal TSO users is not changed, but this change causes a CLIST to be counted as multiple transactions thereby reducing the reported response time. Beware of consultants whipping into an installation, changing the parameter and announcing that they've increased the TSO volume while reducing the response time!!

### MVS/ESA

MVS/ESA has an even better technique for reducing CLIST response time. You can use the VLF and LLA facilities to load CLISTs into a data space. The added benefit of this is that CLISTs are pre-compiled one time when loaded into a data space, rather than compiled at each execution. More about this in a later issue.

```
                         MIG=264 CPU= 99 UIC= 44 PDT=  59 DPR= 92    ASRM      D
 09:03:55        P P  TRANS   TRANS    TX   TX     TX     TX     TX     TX   SESS
 JOBNAME   DMN   G P  ACTIVE  CUR RES  CT   SC     CPU    MSO    IOC    SRB  TOTAL

 TSOUSR1    1    2 1  00:00:12 00:00:12  1   1     707   1363    123     65   2258
 TSOUSR1    1    2 1  00:00:22 00:00:22  1   1     795   1480    142     72   2489
   .. Now let TSOUSR1 enter myclist ..
 TSOUSR1    1    2 1  00:00:45 00:00:45  2   2    1750   3475    162    150   5537
   .. myclist again and again ..
 TSOUSR1    1    2 1  00:00:55 00:00:45  .................
 TSOUSR1 .............
 TSOUSR1 .............
   .. Now let TSOUSR1 enter %myclist ..
 TSOUSR1 .............
 TSOUSR1 .............
 TSOUSR1 .............


 Notes:  From the above screen, you can estimate the average CPU service units and I/O service
units both with and without the "%".  The difference between the two will show the savings
using "%".
```

**Figure 9 - RMF Monitor II ASRM Activity**

## Providing Consistent Response Time

This is a change that won't improve TSO response time immediately, but it will make life much easier after a hardware upgrade. I wouldn't be without it!

RTO (Response Time Option) is an SRM parameter that allows an installation to specify the desired average response time for first period TSO users. It is used to provide consistent response time across hardware and software upgrades. To install, the system programmer specifies a desired TSO first period response time in seconds and SRM will attempt to provide that response time. The specification is given in the RTO parameter. The sample shown in Figure 10 is a request for 0.7 second response time. The RTO value is specified in seconds from 0.0 to 999.9. Don't set it too high or your TSO users will think the system has crashed!

```
PGN=2,(...DP=F54,DUR=500,RTO=.7...)
      (...DP=F53,DUR=1000...)
      (...DP=M4...)
```

**Figure 10 - RTO (Response Time Option)**

RTO is an SRM option that represents a moral dilemma in any installation. In order to be most effective, it should be kept secret! And in many installations, the users don't have a lot of trust in the system programmers to begin with. If the users discover the secret, they'll never trust the systems group again! So here's a description and some suggestions from someone who has found it extremely effective, even though I suffered pangs of guilt for doing it.

During execution, SRM calculates an average response time. Using the example from Figure 10, if the desired response time is 0.7 seconds and the average response time in a performance group is .5 seconds, SRM will delay the swap in for all TSO users of the performance group for .2 seconds (.7 desired minus .5 actual equals .2, the delay). If a specific transaction actually takes .2 seconds, the response will therefore be .4 seconds. If it actually takes 1.0 seconds, the user will see a response in 1.2 seconds. Notice that the transaction is delayed first and then allowed to execute. I really wish they had implemented it where the delay was at the end of the transaction, so that long transactions wouldn't incur the delay. RTO is only applied for the swap in after a terminal input wait (where TSO is waiting for input from the terminal). This is applicable to both physically swapped and logically swapped users. It is not applied to output terminal waits, subsequent commands in a CLIST, stacked commands and swap-ins of users holding enqueued resources.

What does this mean? If the average response time is equal to or greater than the RTO value, there will be no effect. If the average response time is less than the RTO value, ALL transactions in first period are delayed by this set amount, even those that will pass into periods 2 and 3. Thus every user is delayed by the delay amount. The good news is that it will provide fairly consistent response time to all users. Remember that this is simply a cushion to keep response time from being *too* good.

The most common use of RTO occurs when an installation is installing an upgrade that is needed to handle a future workload. As an example, an installation predicts its TSO workload will increase by 50% by the end of the year, so they install a faster CPU to handle the increased workload. The workload gradually increases during the year. If RTO is NOT used, here's the scenario: TSO response time has been getting worse and worse during the last six months. The service level objectives had been established at .7 seconds and TSO response time has increased from .7 seconds to 2.0 seconds during the last six months. Does this sound familiar to anyone? A CPU upgrade is installed. Without RTO, the users might see response times of .2 seconds and be overjoyed. Then, as the expected workload starts to show up on the machine, response time degrades. First to .4 seconds, then .6 seconds, then .7 seconds. Even though the users had once agreed to .7 seconds, they'll never be satisfied with worse than .2 seconds again.

This brings up more management issues than we can really handle in this issue. Some of the questions to be answered: How does your management view the users, their morale and their productivity? Are users really more productive with .2 second response time than with .7 second response time? Does management want to encourage or discourage TSO interactive use? Is management willing to pay for the increased resources consumed by users who will use the system more because of fast response times? Will users resent the data center if they find out that response time is being kept at a minimum?

So why doesn't everyone use this technique? If users find that their response is artificially degraded, they'll be angry. After all, they'll point out, productivity studies have shown that people are more productive with faster response time. That's true, but the studies normally don't discuss what happens when response time increases again. Some installations use RTO only during the upgrade period and not after. Others have used it continually without noticeable effect.

### WHAT TO DO

The change is relatively easy. The best time to implement this change is just prior to a CPU upgrade. Here's a sample checklist:

- ☐ Monitor TSO response time. See HOW TO MEASURE.
- ☐ Create a test IPS with RTO specified. Set the RTO value on first period TSO users to the desired response time (the service objective), such as .7 seconds. This is probably some value less than they are receiving before the upgrade. If the average response time is over .7 seconds, SRM will cause no delay.
- ☐ Install the new IPS. See WHAT TO DO in REDUCING TSO OUTPUT WAIT SWAPS.
- ☐ Measure the indicators after the change. After the change, the users, who have been getting 2 second response will be happy with the .7 second response. The actual time for a transaction (.2 seconds) isn't seen by the user.
- ☐ Continue to monitor TSO response time. As workload increases, the actual transaction time will increase, but the observed response time will remain at .7. If the actual response time exceeds .7 seconds, RTO is not invoked (that is, there is no additional delay).
- ☐ Implement the IPS in production.

### SAMPLE IMPLEMENTATIONS

Three of my experiences may help explain. In Company A, I met with the users at the start and explained what we wanted to do. They agreed to the artificial delay as long as they saw an improvement with the new upgrade. Most of the time everyone was happy, but if anything increased response time, they blamed RTO (not paging or a CICS loop or DASD contention). At Company B, we implemented RTO without the users being told. By the way, the managers of the users should ALWAYS be told! The upgrade was installed, the response time improved slightly and when new TSO users were added, nobody saw any degradation of response. At Company C, the users weren't told initially and found out about the delayed transactions. The anger could be heard round the world! It was a long time before a dialog could be re-established. I personally think that RTO can be an extremely useful tool. I've even gone as far as implementing the technique in CICS and IMS (back when source code was available), which don't provide such an option. You do have to be careful of the political implications.

## MVS REVIEW

This section is provided to explain basic MVS concepts or techniques that relate to material presented in the tuning or measurement articles. This month, a look at how the MVS Program Fetch operates.

## MVS Program Fetch

MVS program fetch locates and loads programs and TSO commands and is used for program loads, links, XCTLs, attaches, initial executes, TSO commands and CLISTS. Knowledge of its search order will help you understand several tuning tips. Figure 11 shows the search order.

1. **Job pack area**
2. **TASKLIB(S)**
3. **STEPLIB if present**
4. **JOBLIB if present and no STEPLIB**
5. **Link pack area - active modules**
6. **Link pack directory - inactive modules**
7. **SYS1.LINKLIB and LNKLST modules**
8. **If TSO, SYSPROC libraries**

**Figure 11 - Program Fetch Search**

Step 1, the job pack area, refers to programs that have been previously loaded and are still resident in the address space. It will seldom find modules in this step.

Step 2 refers to the dataset (TASKLIB) that is defined during the ATTACH of a task. Applications seldom use this technique, but it's often used by software products, such as CICS with the DFHRPL ddname. TASKLIBs are product dependent and their installation instructions will tell you when to include them in the JCL. Modules from TASKLIB are loaded into the user's private region and increase his working set size (WSS).

Step 3 is to search STEPLIB and all of its concatenations. Batch applications use STEPLIBs to point to test load libraries that are continually updated. Some sites put STEPLIBs in TSO LOGON procs which point to ISPF load libraries or production load libraries. Sysprogs might use TSO STEPLIBs for testing new releases of software products. Remember that STEPLIB is searched for every TSO command. Modules from STEPLIB are loaded into the user's private region and increase his WSS.

If there isn't a STEPLIB, but there is a JOBLIB, then the datasets concatenated under the JOBLIB ddname are searched (Step 4). Started tasks and TSO users never have JOBLIBs. Modules from JOBLIB are loaded into the user's private region and increase his WSS.

Step 5 is a fairly short step. Member LPALSTxx in SYS1.PARMLIB defines one or more libraries containing reentrant modules that are to be loaded at IPL into common virtual storage (called LPA - Link Pack Area). A directory of all LPA modules is also created at that time. As modules are referenced, a pointer is added to the "active CDE (Contents Directory Entry)" list. Some of the LPA modules may be defined in SYS1.PARMLIB member IEAFIXxx as residing in fixed storage (FLPA), while other LPA modules may be defined in IEALPAxx as being modifiable (MLPA). Step 5 is a search through the MLPA, the FLPA and the active CDE list. Most active routines, such as access method routines and high use TSO routines will be found in this list.

Step 6 goes on to search the rest of the link pack directory, an in-core table that is dependent on the number of modules in SYS1.LPALIB and datasets defined in LPA.

Step 7 searches the link list members. This list consists of SYS1.LINKLIB and all datasets defined in LNKLSTxx in SYS1.PARMLIB. Most installations will have all or part of the linklist directory in storage using LLA (Linklist Lookaside) in MVS/XA or LLA (Library Lookaside) in MVS/ESA. Modules from link list are loaded into the user's private region and increase his WSS.

Step 8 (finally!). If program fetch has not found the module, it assumes that it's searching for a TSO CLIST and starts a search of the datasets concatenated under the SYSPROC dd statement of the TSO logon proc. This is where you would expect to find TSO CLISTs.

You'll see in steps 2, 3, 4 and 8 that a physical DASD search of the library directories must be made. The physical directory searches can be fairly costly in response time and also in the amount of contention on a DASD volume. The elimination of the DASD directory search time is one of the benefits of using LPA for modules. Another major benefit is that modules in LPA reside in common storage while other modules reside in the private address space and take more central storage and result in larger swap sizes.

In MVS/ESA, there is an additional aspect of library searches to consider. Any library can be controlled using LLA (Library Lookaside). This includes TASKLIBs, STEPLIBs, JOBLIBs, and of course link list. If a library is controlled by LLA, program fetch will request the directory search of LLA. This results in a cross memory search (using CPU cycles rather than I/O to search the directory). Paging may be involved in place of traditional I/O if the directory is not in central storage at the time of the request. Installations with a lot of expanded storage will tend to find most directories living in central storage or expanded storage. Use of LLA reduces the physical directory search, although it does not reduce the working set size or additional use of central storage.

# MVS MEASUREMENTS

This section provides additional information each month on the meaning of measurements in MVS and techniques for measuring MVS.

## Measuring TSO Response Time

You can measure TSO response time in one of two ways. TSO/MON from LEGENT Corporation [REF004] provides measurements on TSO response time and command usage. RMF Monitor I collects data on TSO response time and records it by performance group in the SMF Type 72 records. I'll use RMF Monitor I data since most installations have access to it.

Figure 2 is a sample RMF Workload Activity report (type 72 data). This example shows a TSO performance group 0002 with three periods. The period is listed in the second column. The rightmost column for each period shows the average response time and the standard deviation of the response time. This is an estimated standard deviation using the "sum of the squares" technique. In our example, we see that first period TSO users are getting subsecond response (0.156 seconds). That looks pretty good to me! You must remember, however, that this is an average. Some transactions are shorter and some are longer. Some statisticians use a calculation derived from "Tchebysheff's Theorem" which states that in a non-normal distribution you can find 75% of the responses within 2 standard deviations of the average.

We can apply this algorithm to our example as follows: we find that 75% of the transactions completed within the average plus two standard deviations or .156 + 2*1.72 = 3.6 seconds. Another way of viewing this is to say that 25% of the transactions are *over 3.6 seconds*! That .156 doesn't look as good now! This is only an estimate, but shows you that large standard deviations could be an indication that not all users are experiencing great response time. To expand this technique further, you can see that 25% of second period users have over 5.7 second response time and 25% of third period users have over 2 minute response time.

The reported response time is based on the number of ended transactions during that period, which are shown in the second column from the right. Read the discussion on REDUCING CLIST RESPONSE TIME for a further discussion of the definition of a transaction.

A common calculation made from this report is the average number of transactions completing in period 1. In our example in Figure 2, the total number of TSO transactions is shown as the fourth group of data (GROUP PERIOD = ALL). Simply determine the percentage of first period transactions by dividing them by the total and multiplying by 100 (18230/23647 * 100 = 77.1%).

These measurements represent internal response time only. The user sees longer response times due to network delay and terminal polling. The RMF response time should be used to simply see the host portion of the user response time. A network monitor or a PC-based application is needed to measure the true user response time.

## Measuring TSO Command Use

In several of these tuning techniques, you'll want to measure the effect of a change on a given TSO application. The simplest method is to use a third party software package, such as TSO/MON from LEGENT [REF004]. I'd like to describe two methods here that you can use with standard RMF and SMF facilities.

But first a word about "commands". RMF and SMF can identify TSO commands or command processors. They cannot identify called programs or CLISTs. Called programs all show up under the command "CALL" and CLISTs show up under the command "EXEC". If you wanted to get some information about a specific CLIST, you could write a command processor to invoke the CLIST. Then let users execute the command processor. For example, I've had several requests asking how to measure SAS usage under TSO. The SAS program counts under the CALL command and the SAS CLIST counts under the EXEC command. You can write a command processor called RUNSAS, ask users to execute RUNSAS, then collect command usage on RUNSAS.

*[Editor's Note - This was contributed by* ***Chuck Hopf*** *regarding collecting SAS usage: "SAS provides a user exit to write an SMF record that contains the CPU time, EXCPs, and memory for each SAS step. Version 5 also includes the number of pages. The exit is documented in SAS Publication Y-103R."]*

### RMF COMMAND MEASUREMENTS

One method of collecting command response and resource usage is through RMF Monitor I Workload Activity data (SMF type 72). On the ICS (IEAICSxx in SYS1.PARMLIB), you can define a new report performance group as shown in Figure 12. Then collect RMF data. Look at the RMF data for the report performance groups. In our example, we would look at report performance group 301 to determine the number of executions of RUNSAS (ENDED TRANSACTIONS), the average response time and the CPU and I/O usage from the service units. A fuller explanation of service units will be given in a later issue.

```
SUBSYS=TSO,PGN=2
   TRXNAME=ALLOC,RPGN=300
   TRXNAME=RUNSAS,RPGN=301
```

**Figure 12 - Use of Report Performance Groups**

### SMF COMMAND MEASUREMENTS

The second technique is to use the SMF type 32, TSO User Work Accounting Record, available with TSO/E. When this record is activated, you can collect resource usage by command. A type 32 record is written when a TSO user logs off and at the end of an interval if interval recording is turned on. It contains information on commands executed by the user during that period. Only commands in CSECT IEEMB846 are collected. The minimum information that you'll get is the number of commands executed for each command listed in the CSECT. Commands not specified in the CSECT fall into a category of ***OTHER. Optionally, you can get more resource information by specifying "DETAIL" in the SUBSYS(TSO) option in SMFPRMxx. This additional data includes TCB time, SRB time, the number of TGETs and TPUTs, transaction count, EXCP count and device connect time. Obviously, DETAIL requires more CPU time and more space in each record (40 bytes per command). Fewer commands take less overhead. The following steps are needed:

- ☐ Create command processors for programs or CLISTs that you want to monitor.
- ☐ Define the commands that you want to monitor in CSECT IEEMB846.
- ☐ Modify SMF to collect type 32 records in member SMFPRMxx in SYS1.PARMLIB. Optionally, add DETAIL to the SUBSYS TSO if it's not specified in the SYS parameter.
- ☐ Collect type 32 data.
- ☐ Process the type 32 records to get average CPU and I/O consumption for the commands.

When do you use one technique versus the other? I use RMF to get response time and resource usage for a command, realizing that it is the average for all users. I use the type 32 records to find out who is using a particular command and how many resources are consumed by that user. For example, I might want to find out who is using SAS and which are the largest users.

## Q & A

Q The system seems to come to a grinding halt whenever we do an SMF switch in the middle of the day. What's causing it? How can I stop it?

A The probable reason is the collection of SMF type 19 records (DASD Volume records). At IPL time and SMF switch time, SMF collects VTOC information on all online DASD devices. It uses this information to create type 19 records, one record per device that contains the VOLSER, the number of available tracks and the largest available extent. In order to collect this information, however, SMF must read each VTOC and place a RESERVE on it. In a shared DASD environment, you can cause all systems to take a coffee break while the VTOCs are being read.

You can avoid this situation in one of two ways. If you want the type 19 records, enlarge your SMF datasets so they can be switched only in the middle of the night. Batch jobs will hardly notice the delay. If you don't need type 19 records, turn them off in SMFPRMxx in SYS1.PARMLIB. You'll also need to turn off type 69 records.

## Miscellany

■ A lot of people keep looking for a way to measure the operator delay in resolving tape mounts. And most of them have the program in house! The MXG software from Merrill Consultants [REF005] contains a program to collect tape mount information and record it to SMF.

■ If you're running RMF under a PR/SM environment or under VM, there's a great manual from the IBM International Technical Support Centers, GG24-3274, called *Operating MVS/XA in Multi-Image Environments*! Don't let the MVS/XA stop you - it's applicable to MVS/ESA as well. It describes the differences in RMF measurements when run under PR/SM and VM. It also explains how to measure CPU busy time under VM.

■ I've mentioned several RMF reports and SMF records in this issue. If you'd like a primer on RMF and an SMF reference card, please call.

## Future Topics

**NEXT MONTH:**

FOCUS: Tuning Central Storage

FOCUS: Tuning PR/SM (or MDF or MLPF)

Including a new technique for determining when an LPAR is out of capacity!

MVS REVIEW: Service Units

**FUTURE ISSUES:**

Tuning DASD
Tuning Expanded Storage
Tuning Your Paging Subsystem
Tuning & Measuring Swapping
Tuning VSAM Files
Tuning Application Programs
MVS/ESA Performance Measurements
Tuning Virtual Storage
Tuning MVS for CICS

## References

REF001 - CBT Mods Tape, a shareware tape containing MVS tools and utilities. Now distributed by NaSPA [REF002].

REF002 - NaSPA, National Systems Programmers Association, Inc., 4811 S. 76th St, Suite 210, Milwaukee, WI 53220, (414) 423-2420.

REF003 - IBM SPL: Initialization & Tuning Guide. GC28-1828(MVS/ESA), GC28-1149 (MVS/XA).

REF004 - LEGENT Corporation, Vienna, VA, (703) 734-9494.

REF005 - Merrill Consultants, Dallas, Texas, (214) 351-1966

Cheryl Watson's
**TUNING Letter**
January 1991 (Vol. 1, No. 1)
Watson & Walker, Inc.
**www.watsonwalker.com**

Publisher: Tom Walker
Executive Editor: Cheryl Watson

PURPOSE: To provide practical MVS tuning knowledge and techniques which achieve savings by reducing costs or improving response times.

SUBSCRIPTION RATES: Electronic version (CD-ROM & email) $695 per year. Printed version $495 per year, $545 outside North America. Payment may be made by a check drawn on a U.S. bank, a money order, any major credit card, or by wire transfer. Issues are mailed via first class mail. All back issues are available. Call for a complete list. Send all correspondence to: Watson & Walker, Inc., 2130 Bispham Road, Sarasota, Florida 34231, USA. Tel: 800-553-4562 or 941-924-6565. Fax: 941-924-4892. For customer service, send email to **admin@watson-walker.com**.