215

MVS update

August 2004

In this issue

MVS Update

Published by
Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor
Trevor Eddolls
E-mail: trevore@xephon.com

Publisher
Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs $505.00 in the USA and Canada; £340.00 in the UK; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 2000 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon .com/mvs; you will need to supply a word from the printed issue.

Disclaimer

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

USS processes at a glance

PROBLEM ADDRESSED

In an earlier article I have provided a brief introduction to monitoring USS performance from the z/OS perspective that can be used as a starting point in building a set of USS performance yardsticks (see 'Monitoring USS performance from z/OS – an introduction', *MVS Update*, issues 213 and 214, June and July 2004). This sample process reporter was written for those who are somewhat new to the USS world, and specifically not familiar with commands that may be used to 'see' what is going on there. However, I have to note that only after one has become familiar with the general concepts of USS processes, threads, programs, commands, and cross-correlation of these to an address space, is one ready to proceed to the next step of extracting information about USS process activity.

When it comes to on-line performance monitoring tools for Unix System Services, there are currently two well-known options available within the standard IBM toolkit. The first one is RMF Monitor III OPD report, while the second is SDSF's PS command. However, what I myself really needed was a simple, flexible and yet fast reporting tool that would quickly provide a compact view on processes running, commands associated with these processes, the parent/child relationship of the processes, the thread structure of a process, programs running on these threads, address spaces representing the USS workload as well as the memory, and other resource usage for active processes.

SOLUTION PROPOSED

In a search for a solution I asked myself whether there is any REXX callable service that would allow me to get information about all active processes and threads. It came as no surprise that actually there is such a service. In general, the syscall commands invoke the z/OS Unix callable service that corresponds to the command verb (the first word of the command). The parameters that follow the command verb are specified in the same order as in POSIX.1 and the z/OS Unix callable services, where applicable. For complete information about the processing of a particular syscall command, read about the callable service that it invokes as described in *z/OS Unix System Services Programming: Assembler Callable Services Reference* (SA22-7803).

In this particular case, I have found that getpsent is a callable service that provides data describing the status for only those processes for which RACF allows the user access based on effective user ID, real user ID, or saved set user ID. The data returned is formatted in a stem – stem.0 contains the number of processes for which information is returned; stem.1 to stem.*n* (where *n* is the number of entries returned) each contain process information for the *n*th process. One can use the predefined variables that begin with PS_ or their equivalent numeric values (see Appendix A of a *z/OS Using REXX and z/OS Unix System Services*, SA22-7806 manual) to access the information. It should be noted that besides some 25 variables described in the book mentioned above, there are three additional predefined variables that were added to z/OS V1.2, a description of which appears to be missing from the book. These three variables are: ps_jobname, ps_asid, and ps_waittime.

While getpsent is good at providing process-related information, it does not know anything about the threads making a process. In fact, there is not any other callable service (syscall command) that is thread related. However, there is a function called BPXWUnix available from the USS Tools and Toys page, www-1.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty2.html.

In fact, the BPXWUnix is a part of *REXX Function Package for REXX in OpenEdition*, which contains the seven REXX I/O functions and several other useful functions. The REXX function package is located in an unloaded PDS called REXXFUNC.UNLOAD. Download this in BINARY into an FB 80 dataset. It is restored using the TSO/E receive inda() command.

Complete install and usage information can be found at the URL www.s390.ibm.com/ftp/os390/oe/toys/rexx/rexxfunc.html.

The two important features of BPXWUnix function were exploited in this case – it is the only function in this package that can also be run outside of a USS REXX environment (ie in TSO), and it allows you to run any shell command. It should be noted that when this function is used outside of the Unix REXX environment, stdin, stdout, stderr, and environment variables will not be inherited from the current process environment.

CODE

The REXX EXEC provided here (called PSDSP) is in fact a combination of several syscall commands, each one reporting on specific USS resources. The first used is uname, which returns information about the system you are running on. The getrlimit syscall was used to obtain the maximum and current resource limits for the calling process (part 1 of the report). The next syscall, getpsent, was used here as a main source of information on processes currently running (see part 2 of the report). In order to obtain information about threads making a process, a shell command ps -m –ppid was passed to BPXWUnix function (see part 3 of the report). Of course, one can replace this command with a more elaborate shell command that would display thread wait times and semaphore wait information, as well as the last five syscalls. Note that the last five syscalls (USS kernel calls) are valid only if CTRACE for OMVS syscalls is enabled. The single last syscall is valid all the time. (The CTRACE statement provides tracing while the kernel is starting and to avoid having to issue a TRACE operator command to set tracing options. The only way to change any CTRACE value is with the TRACE command. You cannot use the SETOMVS or SET OMVS commands to change the value.)

As it is set by USS design, the processes on the same system can exchange information or communicate data to each other and synchronize their activities by using three different types of inter-process communication resources – message queues, shared memory segment, and semaphores. The shell command ipcs - y was used to obtain inter-process communication settings (part 4). If you do not specify any flags, ipcs writes information in a short form about currently active message queues, shared memory segments, and semaphores (part 5).

USS PROCESS REPORT

Part 2 information about processes currently running:

```
Display USS processes - summary:
-Jobname- --Asid-- -Userid-- -Process-- --PPID-- --Started at---- ------
--State-------- --Size(KB)-- Wait   Sys Cpu  User Cpu Total Cpu
BPXOINIT     251     BPXROOT         1         0 01.02.2004 18:53:22 File
system kernel wait          88 38:58:87 00:18:36 00:55:10 01:13:46
SYSLOGD1      28     BPXROOT         5         1 01.02.2004 18:55:08 File
system kernel wait         556 00:19:20 00:21:59 01:04:77 01:26:36
TCPIP         32     BPXROOT         7         1 01.02.2004 18:55:11
Running                      11160 00:00:00 28:49:01 26:27:03 55:16:04
TCPIP         32     BPXROOT         8         1 01.02.2004 18:55:11
Running                      11160 00:00:00 28:49:01 26:27:03 55:16:04
TCPIP         32     BPXROOT         9         1 01.02.2004 18:55:11
Running                      11160 00:00:00 28:49:01 26:27:03 55:16:04
TCPIP         32     BPXROOT        10         1 01.02.2004 18:55:11 File
system kernel wait      11160 00:49:32 28:49:01 26:27:03 55:16:04
TCPIP         32     BPXROOT  16777219         1 01.02.2004 18:55:07
Running                      11160 00:00:00 28:49:01 26:27:03 55:16:04
TCPIP         32     BPXROOT  16777220         1 01.02.2004 18:55:14
Running                      11160 00:00:00 28:49:01 26:27:03 55:16:04
SYSTM05       52     BPXROOT  33554456         1 26.03.2004 10:34:36
Running                       3412 00:00:00 00:00:60 00:01:81 00:02:41
DBLDIST       49     BPXROOT  67108991         1 05.03.2004 06:37:26 File
system kernel wait        9768 57:00:86 07:09:80 21:29:42 28:39:22
DBDCCICS      50     BPXROOT  33554560         1 05.03.2004 07:42:14
Running                     147648 00:00:00 01:45:02 05:15:09 07:00:11
FTPD1         53     BPXROOT       130         1 05.03.2004 09:45:28 File
system kernel wait        1636 48:34:98 00:00:02 00:00:08 00:00:10
```

Part 3 information about threads making a process:

```
Currently there are  12 processes running
Display USS processes & threads:
-Jobname- -Asid- -Userid-- Process- Threads --------State-------- Total
Cpu --Size(KB)-- Path----   -- Process started by CMD ---
BPXOINIT   251   BPXROOT         1      5    File system kernel wait
00:01:13        88    BPXPINPR       BPXPINPR
                          00000000          X
0:00              Last syscall:     1FRK
```

```
                        00000001          W
0:45                Last syscall:     1WAT
                        00000002          Y
0:00                Last syscall:     -
                        00000003          K
0:00                Last syscall:     1KIN
                        00000004          F
0:00                Last syscall:     1ACP

-Jobname- -Asid- -Userid-- Process- Threads --------State--------- Total
Cpu --Size(KB)-- Path----   -- Process started by CMD ---
TCPIP       32   BPXROOT  16777220     5    Running
01:55:16     11160    EZBTTMST       EZBTTMST
                        00000000          Y
0:00                Last syscall:     -
                        00000001          Y
0:00                Last syscall:     -
                        00000002          Y
0:00                Last syscall:     -
                        00000003          Y
0:00                Last syscall:     -
                        00000004          R
0:00                Last syscall:     -

-Jobname- -Asid- -Userid-- Process- Threads --------State-------- Total
Cpu --Size(KB)-- Path----   -- Process started by CMD ---
DBLDIST     49   BPXROOT  67108991     3    File system kernel wait
00:28:39      9768    DSNVEUS3       DSNVEUS3
                        00000000          Y
0:00                Last syscall:     -
                        00000001          F
0:00                Last syscall:     1AIO
                        00000004          F
0:00                Last syscall:     1AIO

-Jobname- -Asid- -Userid-- Process- Threads --------State-------- Total
Cpu --Size(KB)-- Path----   -- Process started by CMD ---
DBDCCICS    50   BPXROOT  33554560     2    Running
00:07:00    147648    DFHKETCB       DFHKETCB
                        00000000          Y
0:00                Last syscall:     -
                        00000001          R
0:00                Last syscall:     1KLM
```

PSDSP EXEC

```
/* REXX ************************************************************/
/*    Procedure: psdsp                                              */
/*  Description: Get information about: active processes & threads*/
/*               USS resource limits and IPC                       */
/*****************************************************************/
/* trace ?R                                                        */
Address TSO
userid=SYSVAR(SYSUID)
outds  =userid||'.proc.out'               /*  change dataset name */
x = MSG('ON')                             /* to fit your standards*/
if SYSDSN(outds) = 'OK'
Then "DELETE "outds" PURGE"
"ALLOC FILE(PRC) DA("outds")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
   " REUSE LRECL(145) RECFM(F B) BLKSIZE(1450)"
 arg hlq
/*---------------------------------------------------------------*/
/* Supply the name of received REXX function load library         */
/*---------------------------------------------------------------*/
 if hlq = "" then HLQ = 'uid.REXXFUNC.LOAD'
PARSE SOURCE . . . . . . Envir .
If Envir ¬= 'SH' then ,
 IF SYSCALLS('ON')>3 THEN
   Do
    Say 'Unable to establish the SYSCALL environment'
    Say 'Return code was 'RC
    Return
   End
/*---------------------------------------------------------------*/
/* Allocate  BPXWUnix load library                                */
/*---------------------------------------------------------------*/
Address ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"hlq".LOAD') STACK"
Address SYSCALL
/*---------------------------------------------------------------*/
/* Return USS information                                         */
/*---------------------------------------------------------------*/
'uname sys.'
sis.1 = left('USS process report - produced on',32,),
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' '),
        ||left(time(),10)
sis.2 = left('System identification:',22)
sis.3 = left('Sysname: ',11)||sys.U_SYSNAME
sis.4 = left('Version: ',11)||sys.U_VERSION
sis.5 = left('Release: ',11)||left(sys.U_RELEASE,10)
sis.6 = left('Node    : ',11)||left(sys.U_NODENAME,10)
sis.7 = left('Hardware:',11)||left(sys.U_MACHINE,10)
sis.8 = left(' ',1)
/*---------------------------------------------------------------*/
/* Display current limits for USS resources (process related)     */
/*---------------------------------------------------------------*/
proc.1 = left('Part 1:',20)
proc.2 = left('Display current system limits for USS processes:',5Ø)
proc.3 = left('-',5Ø,'-')
proc.4=  left('Amount of CPU time (in seconds) used by a process:',5Ø)
CALL findlimit 'rlimit_cpu'
proc.5 = left('Current:',9)||rvalue.1
proc.6 = left('Maximum:',9)||rvalue.2
proc.7 = left(' ',1)
proc.8 = left('Files size (in bytes) created by a process:',5Ø)
CALL findlimit 'rlimit_fsize'
proc.9 = left('Current:',9)||rvalue.1
proc.1Ø = left('Maximum:',9)||rvalue.2
proc.11 = left(' ',1)
proc.12 = left('Number of open file descriptors for a process:',5Ø)
CALL findlimit 'rlimit_nofile'
proc.13 = left('Current:',9)||rvalue.1
proc.14 = left('Maximum:',9)||rvalue.2
proc.15 = left(' ',1)
proc.16 = left('Address space size for a process:',5Ø)
CALL findlimit 'rlimit_as'
proc.17 = left('Current:',9)||rvalue.1
proc.18 = left('Maximum:',9)||rvalue.2
proc.19 = left(' ',1)
proc.2Ø = left('Size (in bytes) of a core dump created by a
process:',6Ø)
CALL findlimit 'rlimit_core'
proc.21 = left('Current:',9)||rvalue.1
proc.22 = left('Maximum:',9)||rvalue.2
proc.23 = left(' ',1)
/*-----------------------------------------------------------------*/
/* Print headers and labels                                        */
/*-----------------------------------------------------------------*/
raw.1 = left('Part 2:',5Ø)
raw.2 = left('Display USS processes - summary:',5Ø)
raw.3 = Centre("Jobname",9,"-"),
        Centre("Asid",9,"-"),
        Centre("Userid",9,"-"),
        Centre("Process",1Ø,"-"),
        Centre("PPID",8,"-"),
        Centre("Started at",17,"-"),
        Centre("State",23,"-"),
        Centre("Size(KB)",12,"-"),
        Centre("Wait",5),
        right("Sys Cpu",8),
        right("User Cpu",9),
        right("Total Cpu",9)
w=1
ps.Ø=Ø
/*-----------------------------------------------------------------*/
/* Get process data                                                */
/*-----------------------------------------------------------------*/
```

```
'getpsent ps.'
 Do i=1 to ps.0                     /* Process each entry returned */
 cmd =left('ps -m -p',8)||ps.i.ps_pid
 start = start(ps.i.ps_starttime)
 wait  = TIM(ps.i.ps_waittime )
 syst  = TIM(ps.i.ps_SYSTIME  )
 usert = TIM(ps.i.ps_USERTIME )
 toto  = TIM(ps.i.ps_USERTIME+ps.i.ps_SYSTIME)
/*-------------------------------------------------------------------*/
/* Obtain information about a user                                   */
/*-------------------------------------------------------------------*/
    'getpwuid 'ps.i.ps_euid' pw.'
    'getgrgid 'ps.i.ps_egid' gr.'
 ksiz = ps.i.ps_SIZE / 1024
/*-------------------------------------------------------------------*/
/* Print process-related data                                        */
/*-------------------------------------------------------------------*/
 rww.i =Translate(ps.i.ps_Jobname,"40"x,"00"x), /* Jobname         */
        right(ps.i.ps_asid,7),                  /* Asid (dec.)     */
        left(' ',3),
        right(pw.pw_name,8),                    /* Display user    */
        right(ps.i.ps_PID,8),                   /* Process ID      */
        right(ps.i.ps_PPID,8),                  /* Parent PID      */
        right(sinf,19),                         /* Start date/time*/
        Left(Ps_Status(ps.i.ps_state),27),      /* Proc.status     */
        right(Ksiz,6),                          /* Total size      */
        right(wait,8),                          /* Waittime        */
        right(syst,8),                          /* System CPU      */
        right(usert,8),                         /* User CPU        */
        right(toto,8)                           /* System + User   */
/*-------------------------------------------------------------------*/
/* Print headers and labels                                          */
/*-------------------------------------------------------------------*/
 kww.1 = left(' ',1)
 kww.2 = left('Part 3:',10)
 kww.3 = left('Currently there are',20)||right(ps.0,3),
         left('processes running',19)
 kww.4 = left('Display USS processes & threads:',50)
call SHCMD cmd                                 /* Obtain thread info*/
/*-------------------------------------------------------------------*/
/* Print header and labels                                           */
/*-------------------------------------------------------------------*/
th.w   = Centre("Jobname",9,"-"),
         Centre("Asid",6,"-"),
         Centre("Userid",9,"-"),
         Centre("Process",8,"-"),
         Centre("Threads",7,"-"),
         Centre("State",23,"-"),
         right("Total Cpu",9),
         Centre("Size(KB)",12,"-"),
        left("Path",14,"-"),
        left("Process started by CMD",44,"-"); w = w +1
/*------------------------------------------------------------------*/
/* Print process & thread related data                              */
/*------------------------------------------------------------------*/
th.w   =Translate(ps.i.ps_Jobname,"40"x,"00"x), /* Jobname         */
        right(ps.i.ps_asid,5),                  /* Asid (dec.)     */
        left(' ',1),
        right(pw.pw_name,8),                    /* Display user    */
        right(ps.i.ps_PID,8),                   /* Process ID      */
        right(thread,5)  left(' ',2),           /* # threads       */
        Left(Ps_Status(ps.i.ps_state),24),      /* Proc/thread     */
        left(toto,8) left(' ',2),               /* Total Cpu       */
        right(Ksiz,6) left(' ',2),              /* Storage size    */
        left(ps.i.ps_PATH,14),                  /* Command path    */
        left(ps.i.ps_CMD,44)                    /* Command         */
/* The following variables are also available but                   */
/* were NOT used by this USS process reporter:                      */
/* Controlling tty                         : ps.i.ps_CONTTY         */
/* Effective group ID                      : ps.i.ps_EGID           */
/* Effective user ID                       : ps.i.ps_EUID           */
/* Foreground process group ID             : ps.i.ps_FGPID          */
/* Process Group ID                        : ps.i.ps_PGPID          */
/* Real group ID                           : ps.i.ps_RGID           */
/* Real user ID                            : ps.i.ps_RUID           */
/* Saved set group ID                      : ps.i.ps_SGID           */
/* Saved set user ID                       : ps.i.ps_SUID           */
/* Session ID (leader)                     : ps.i.ps_SID            */
/* Server flags                            : ps.i.ps_SERVERFLAGS    */
/* Server name supplied on registration    : ps.i.ps_SERVERNAME     */
/* Server type (File=1; Lock=2)            : ps.i.ps_SERVERTYPE     */
/* Maximum number of vnode tokens allowed : ps.i.ps_MAXVNODES      */
/* Current number of vnode tokens          : ps.i.ps_VNODECOUNT     */
/*                                                                  */
 do l = 1 to k
   w = w + 1
  th.w   = left(' ',25) left(stid.l,10) left(' ',6), /* thread id */
           left(tstat.l,10) left(' ',17),       /* thread status  */
           left(ttime.l,10) left(' ',6),        /* thread Cpu     */
           left('Last syscall:',17),
           left(tsysc.l,10)                     /* syscall        */
 end
   w = w +1
th.w  = left(' ',1); w = w +1
End                                              /* main loop end */
/*------------------------------------------------------------------*/
/* Print header and labels then issue shell command                 */
/* to obtain inter process communication settings (ipcs -y)         */
/*------------------------------------------------------------------*/
it.1 = left('Part 4:',50)
it.2 = left('Display current IPCS settings:',40)
it.3 = left('-',50,'-')
g=0
com = 'ipcs -y'
 call SHCMM com
    do z = 1 to n
      g = g +1
      ip.g =left(x.z,90)
 end
 drop ip.g
/*----------------------------------------------------------------*/
/* Print header and labels then issue shell command               */
/* to obtain inter process communication usage (ipcs)             */
/*----------------------------------------------------------------*/
ix.1 = left(' ',1)
ix.2 = left('Part 5:',50)
ix.3 = left('Display current IPCS usage:',40)
ix.4 = left('-',50,'-')
h=0
com =  'ipcs'
 call SHCMM com
    do z = 1 to n
      h = h +1
      ic.h =left(x.z,90)
 end
 drop ic.h
call syscalls 'OFF'
Address TSO
 nt.1  =left(' ',1)
 nt.2  =left('-',45,'-')
 nt.3  =left('A Short Explanation of Current Thread state:',77)
 nt.4  =left('-',45,'-')
 nt.5  =left(' ',1)
 nt.6  =left('A - Message queue receive wait',77)
 nt.7  =left('B - Message queue send wait',77)
 nt.8  =left('C - Communication system kernel wait',77)
 nt.9  =left('D - Semaphore operation wait',77)
 nt.10 =left('E - Quiesce frozen ',77)
 nt.11 =left('F - File system kernel wait',77)
 nt.12 =left('G - MVS Pause wait',77)
 nt.13 =left('K - Other kernel wait (for example, pause or
sigsuspend)',77)
 nt.14 =left('J - The thread was pthread created rather than dubbed',77
)
 nt.15 =left('N - The thread is medium weight',77)
 nt.16 =left('O - The thread is asynchronous and medium weight',77)
 nt.17 =left('P - Ptrace kernel wait',77)
 nt.18 =left('Q - Quiesce termination wait',77)
 nt.19 =left('R - Running (not kernel wait)',77)
 nt.20 =left('S - Sleeping',77)
 nt.21 =left('U - Initial process thread (heavy weight and
synchronous)',77)
 nt.22 =left('V - Thread is detached',77)
 nt.23 =left('W - Waiting for child (wait or waitpid callable
service)',77)
 nt.24 =left('X - Creating new process (fork callable service is
running)',77)
 nt.25 =left('Y - Thread is in an MVS wait',77)
/*---------------------------------------------------------------*/
/* Write out USS info, settings, process, thread, and IPCS data   */
/*---------------------------------------------------------------*/
"EXECIO * DISKW PRC (STEM sis.)"
"EXECIO * DISKW PRC (STEM proc.)"
"EXECIO * DISKW PRC (STEM raw.)"
"EXECIO * DISKW PRC (STEM rww.)"
"EXECIO * DISKW PRC (STEM kww.)"
"EXECIO * DISKW PRC (STEM th.)"
"EXECIO * DISKW PRC (STEM it.)"
"EXECIO * DISKW PRC (STEM ip.)"
"EXECIO * DISKW PRC (STEM ix.)"
"EXECIO * DISKW PRC (STEM ic.)"
"EXECIO * DISKW PRC (STEM nt.)"
/*---------------------------------------------------------------*/
/* Close & free allocated report file; then display result        */
/*---------------------------------------------------------------*/
"EXECIO Ø DISKW PRC (FINIS "
 "free FILE(PRC)"
  Address ISPEXEC
 "ISPEXEC BROWSE DATASET('"outds"')"
exit Ø
FINDLIMIT:
/* Return current limits for process                              */
 arg value
 num=value(value)
 "getrlimit "num" rvalue."
 return rvalue.1 rvalue.2
SHCMD:
/* REXX - BPXWUnix: list thread info                              */
  parse arg usscmd
  call BPXWUnix usscmd,,out.
  If result<>Ø Then Say "Rc("||result||")"
  thread = out.Ø - 2
 Do r = 3 to out.Ø
 k = r - 2
      stid.k = word(out.r,4)
      tstat.k= word(out.r,5)
      ttime.k= word(out.r,6)
      tsysc.k= word(out.r,7)
 End r
 return thread k
```

```
SHCMM:
/* REXX - BPXWUnix: call ipcs commands                              */
  parse arg usscmd
  call BPXWUnix usscmd,,out.
  If result<>Ø Then Say "Rc("||result||")"
   Do n = 1 to OUT.Ø
   x.n=OUT.n
      End n
 return n
TIM:
/* REXX - convert duration vars to hh:mm:ss:hd format               */
arg time
    time1    = time % 1ØØ
    hh       = time1 % 36ØØ
    hh       = RIGHT("Ø"||hh,2)
    mm       = (time1 % 6Ø) - (hh * 6Ø)
    mm       = RIGHT("Ø"||mm,2)
    ss       = time1 - (hh * 36ØØ) - (mm * 6Ø)
    ss       = RIGHT("Ø"||ss,2)
    fr       = time //  1ØØØ
    fr       = RIGHT("Ø"||fr,2)
    rtime  = hh||":"||mm||":"||ss||":"||fr
    return rtime
START:
/*  Convert starting date & time to display                          */
arg strtime
  "gmtime (strtime) tm."
    If rc>=Ø Then Do
      tmdd = Right(tm.tm_mday,2,Ø)||"."
      tmmm = Right(tm.tm_mon,2,Ø)||"."
      tmyy = tm.tm_year
      tmhh = Right(tm.tm_hour,2,Ø)||":"
      tmmn = Right(tm.tm_min,2,Ø)||":"
      tmss = Right(tm.tm_sec,2,Ø)
      sinfo = tmyy||tmmm||tmdd tmhh||tmmn||tmss
      sinf  = tmdd||tmmm||tmyy tmhh||tmmn||tmss
    End
 return sinf
Ps_Status:
  Parse Arg ps_status
  Select
    when ps_status=ps_child   Then
       ps_status ="Waiting for a child process"
    when ps_status=ps_fork    Then
       ps_status ="fork() a new process"
    when ps_status=ps_freeze  Then
       ps_status ="QUIESCEFREEZE"
    when ps_status=ps_msgrcv  Then
       ps_status ="IPC Msgrcv wait"
    when ps_status=ps_msgsnd  Then
```

```
       ps_status ="IPC Msgsnd wait"
    when ps_status=ps_pause   Then
       ps_status ="MVS PAUSE"
    when ps_status=ps_quiesce Then
       ps_status ="Quiesce termination wait"
    when ps_status=ps_run     Then
       ps_status ="Running"
    when ps_status=ps_semwt   Then
       ps_status ="IPC Semop wait"
    when ps_status=ps_sleep   Then
       ps_status ="Sleep() issued"
    when ps_status=ps_waitc   Then
       ps_status ="Communication kernel wait"
    when ps_status=ps_waitf   Then
       ps_status ="File system kernel wait"
    when ps_status=ps_waito   Then
       ps_status ="Other kernel wait"
    when ps_status=ps_zombie  Then
       ps_status ="Process cancelled"
    when ps_status=ps_zombie2 Then
       ps_status ="PS_ZOMBIE2"
    Otherwise ps_status = "PS_UNKNOWN"
  End
Return ps_status
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

VSAM System Managed Buffering

INTRODUCTION

Everyone knows that, just like DB2, VSAM performance can be improved by buffering when the control interval data is transferred from DASD to a virtual storage area.

Typically when a VSAM application performance degrades, a tuning exercise is carried out where the BUFND (data buffers) and the BUFNI (index buffers) values are arrived at religiously, based on the LISTCAT output. The problem with this approach is that these values are sensitive to changes in dataset definition, eg CI size, CA size, record size, and freespace. Even the normal data growth of the dataset requires that this exercise be carried out periodically.

This article aims at explaining VSAM System Managed Buffering and how allowing the system to optimize the buffer size and the algorithm used would be simple and appropriate in most situations.

VSAM BUFFERING

VSAM, by default, provides enough space for only three buffers, and use of these buffers depends on the type of VSAM:

- For KSDS or variable-length RRDS, two are allocated for data CIs and one for an index CI. One of the data buffers is used only for formatting CAs and during CI/CA splits.
- For ESDS, fixed-length RRDS, or LDS, only data buffers are needed.

It is obvious that the default buffers (surprisingly unchanged over the years) are not sufficient to provide any meaningful performance benefits.

By choosing the appropriate buffering technique and VSAM buffer length, the number of I/Os and the I/O response time can be considerably reduced. The alternative approaches by which this buffering can be controlled are:

1. Specifying BUFFERSPACE in the DEFINE command. This approach is no longer recommended.
2. Specifying BUFSP, BUFNI, and BUFND in the VSAM ACB macro.
3. Specifying BUFSP, BUFNI, and BUFND in the JCL DD AMP parameter.
4. Specifying ACCBIAS in the JCL DD AMP parameter.
5. Specifying SYSTEM or USER in the data class record access bias parameter.

Specifying BUFSP is also not preferred, for the following reasons:

- If BUFSP is specified and the amount is smaller than the minimum amount of storage required to process the dataset, VSAM cannot open the dataset.
- A valid BUFSP amount takes precedence over the amount called for by BUFND and BUFNI.

While explicitly specifying the number of buffers, it is not uncommon to set a number of buffers and simultaneously reduce the CI size (possibly to improve random performance), nullifying the impact. Also specifying more buffers (either data or index) than necessary might cause excessive paging or excessive internal processing.

VSAM BUFFERING TECHNIQUES

VSAM provides four buffering techniques:

1 *Non-Shared Resources* (NSR), which is the default. In NSR, the buffers are located in the private area and are not shared among VSAM datasets. As a sequential algorithm manages the buffers, this is most suited to sequential processing.

2 *Local Shared Resources* (LSR), where the buffers are shared among VSAM datasets in the same address space, thereby utilizing the virtual storage efficiently. This is exploited by OLTPs like CICS and IMS where a large number of datasets are open in one address space.

3 *Global Shared Resources* (GSR), which is an extension of LSR. The buffers are located in common storage and can be shared by datasets in multiple address spaces. Because of constraints like requiring the program to be in supervisor state and the inability to do workload balancing, GSR is not recommended.

4 *Record Level Sharing* (RLS) is what CICS exploits effectively.

For more details on each of these techniques, refer to the latest version of IBM Redbook *VSAM Demystified*, dated September 2003.

SYSTEM MANAGED BUFFERING (SMB)

A VSAM batch program's performance can be greatly improved by using buffering. Although you could control the buffering technique and the number of buffers used by each program, SMB is an option that lets the system do it.

The following are the factors to be considered for using SMB:

1 SMB introduced in DFSMS V1R4 is, in effect, only for VSAM datasets open for NSR processing.

2 SMB requires that the system-managed VSAM datasets are defined in extended format (use data class parameter DSNTYPE=EXT).

3 SMB originally supported KSDS only. With DFSMS V1R5, SMB is now available for all other VSAM dataset types, namely ESDS, RRDS (fixed or variable), and LDS.

USING SYSTEM MANAGED BUFFERING

A VSAM dataset can be made to use SMB simply by defining the dataset through an SMS data class with RECORD_ACCESS_BIAS=SYSTEM.

Typically a change in data class affects only new allocations. But in the case of SMB, the data class attributes are retrieved and used when the dataset is opened. Therefore, existing VSAM datasets can also be made to use SMB just by changing the RECORD_ACCESS_BIAS parameter of the associated data class.

Another approach to using SMB is to specify an appropriate ACCBIAS value for the AMP parameter in the JCL. ACCBIAS in the JCL overrides the record access bias value specified in the data class.

SMB PROCESSING TECHNIQUES

SMB will weight or optimize buffer handling towards either sequential or direct processing. ACCBIAS value in the JCL can be any of the following four SMB processing techniques:

- SO – SMB with sequential optimization. It uses the NSR buffering technique with large read-aheads. It is appropriate for applications that read a large percentage of a dataset in sequential order. A back-up of a dataset would be one such application.
- SW – SMB weighted for sequential processing (the majority is sequential and some is direct). It uses the NSR buffering technique and uses read-ahead buffers for sequential requests. The read-ahead will not be as large as the amount of data transferred with SO. It provides additional index buffers for direct requests.

  This technique is preferred if the application accesses the entire dataset predominantly in sequential order.
- DO – SMB with direct optimization. It uses the LSR buffering technique with no read-ahead. This is suitable for applications accessing records in a completely random order.
- DW – SMB weighted for direct processing (the majority is direct and some is sequential). It uses the LSR buffering technique. It provides minimum read-ahead buffers for sequential retrieval and maximum index buffers for direct requests. This technique is preferred if the application accesses the dataset predominantly in random order.

FORCING OR OVERRIDING SMB

Instead of you choosing one of the above four techniques, you can let the system choose the buffering technique by specifying an ACCBIAS value of ‘SYSTEM’. The system will choose the technique based on the processing options that are specified when the dataset is opened, and on the following storage class attributes:

1 BIAS values (direct or sequential).

2 Associated millisecond response (MSR) times given as the performance objective.

An ACCBIAS value of USER (the default) allows the SMB specified in the data class to be overridden. If the value is specified as USER, the buffers are obtained in the same way that the system would have behaved without SMB. This option is quite useful to handle exceptional tasks where using SMB is not appropriate, whereas other tasks using the same dataset can benefit from SMB.

It is interesting to note that USER and SYSTEM are the only values you may use to specify record access bias in the data class. Allowing other values at the data class level doesn't make sense because they impact all the datasets pertaining to a data class and even a single dataset will be accessed differently (sequential, random, or dynamic) by different programs.

HOW SMB WORKS

When a dataset is opened, SMB chooses the buffering option based on dataset and application specifications. After the initial decision on buffering technique and number of buffers has been made, SMB has no further involvement.

If the requested System Managed Buffering request could not be completed during open, the request can fail with a message IEC161I.

STORAGE REQUIREMENTS OF SMB

The storage requirements for SMB buffers are based on the following factors:

1 Number of VSAM datasets opened for SMB within a single application program.

2 Technique chosen or specified.

3 Dataset size (applicable for some techniques).

Using buffers above 16MB virtual storage ensures that the buffer size is allocated independently of the region size. While the default VSAM allocation for a resource pool is below the 16MB line, for SMB the default is above the 16MB line.

By specifying the following values in the RMODE31 value for the AMP parameter, you can control the virtual storage location of buffers and control blocks:

1 BUFF – buffers above the 16MB line.

2 CB – control blocks above the 16MB line.

3 ALL – control blocks and buffers above the 16MB line.

4 NONE – control blocks and buffers below the 16MB line.

If the application runs as AMODE=RMODE=24 and issues locate-mode requests (RPL OPTCD=(,LOC)), it is required that RMODE31=NONE is specified on the AMP= parameter for the datasets using SMB.

SMB AND DIRECT OPTIMIZATION (DO)

In the case of direct optimization, additional subparameters SMBVSP, SMBDFR, and SMBHWT can be used to further control VSAM performance. These can be stated either as AMP subparameters in the JCL or in conjunction with the dataclass record access bias parameter.

DEFERRING WRITE REQUESTS

VSAM performance can further be improved by deferring writes that save I/O operations. Deferred writes provide performance benefits in the case of multiple reads, where the request is satisfied by the data in the buffer pool as well as multiple updates to a control interval where the updated CI is written only once.

VSAM automatically defers writes for sequential PUT requests. For direct PUTs, by default, the deferred write processing depends on the SHAREOPTIONS specified for the dataset:

1 Deferred write is used with SHAREOPTIONS (1,3) and (2,3).

2 Non-deferred write is used with SHAREOPTIONS (3,3), (4,3), and (x,4).

The default behaviour can be overridden by specifying the value of Y or N for SMBDFR. An SMBDFR value of Y instructs VSAM to defer the writing of the I/O buffers to the medium until either the dataset is closed or the buffer is required for handling a different request. Deferred write is not allowed for SHAREOPTIONS 4 where the dataset can be fully shared.

CONTROLLING VIRTUAL STORAGE REQUIREMENT

Direct optimization uses the Local Shared Resources (LSR) buffering technique, thereby creating an LSR pool for each dataset opened. A separate pool is built for both data and index components, if applicable, for each dataset.

The default buffer space to be obtained is calculated assuming that 20% of the data accounts for 80% of the accesses. This could result in an increased virtual storage requirement for buffering and the size increases with the size of the dataset involved. If the required virtual storage is unavailable, the SMB technique changes to DW, requiring less virtual storage and an IEC161I message is issued.

The SMBVSP parameter can be used to restrict the size of the LSR pool for the data component. SMPVSP can be specified in KB or MB, and the value can range from 1MB to 2048MB. This specifies the amount of virtual storage to be obtained for buffering while opening the dataset.

It is to be noted that there is no possibility of overriding the size of the pool for the index component.

USING HIPERSPACES

Hiperspace can be used to supplement the use of virtual buffer space for the data component, which reduces the virtual memory requirement. Because data needs to be moved to the virtual buffer from the hiperspace buffer before it can be accessed, it involves greater use of the CPU compared with the use of virtual buffers.

The SMBHWT parameter is used to specify the use of hiperspace for buffering. SMBHWT has the format nn, where nn can take any integer value between 0 and 99. The value of 0, which is the default, implies that the system doesn't get any hiperspace.

A non-zero value for this parameter is used as the multiplier of the virtual buffer space for hiperspace buffers. It is to be noted that this value doesn't act as a direct multiplier but rather as a weighting factor, based on which the number of hiperspace buffers is established.

Similar to virtual space, the hiperspace size buffer is also in multiples of 4KB, and hence the CI size of the data component is recommended to be a multiple of 4KB, avoiding wastage of virtual space and hiperspace.

SMB AND LOAD PERFORMANCE

There are two additional internal techniques that cannot be specified by the user, but are used internally by the system to support dataset creation and load-mode processing:

1 Create Optimized (CO) is the most efficient technique for loading a dataset and is used when an ACCBIAS of SYSTEM is given. This requires that the dataset be in initial load state (HURBA=0) and SPEED be specified at dataset definition.

   The SPEED parameter does not pre-format the data control areas and writes an end-of-file indicator only after the last record is loaded. Hence when an error occurs while loading, it would be required to reload the records from the beginning. For a KSDS, the SPEED parameter affects only the data component.

2 Create Recovery Optimized (CR) uses the maximum number

of buffers to optimize load performance if RECOVERY is specified at dataset definition.

SMB monitors the current state of the dataset and adjusts to the load mode requirements by regulating the number of buffers it uses, thereby minimizing the number of I/O operations required.

USING SMB

- SMB is obviously preferable to the VSAM defaults.
- Performance gains provided by SMB are comparable to and usually better than what can be achieved by specifying buffers explicitly.
- SMB safeguards the performance tuning measures carried out from being adversely impacted because of changes in dataset definition or data growth.
- SMB provides dramatic performance gains in the case of random access because it switches from NSR to exploit LSR buffering techniques.
- Now that SMB supports all types of VSAM (and not just KSDS), the use of Batch LSR can be completely done away with. This, in addition to simplifying the process of buffer exploitation for direct access, provides relief from various restrictions of BLSR.
- Most importantly, SMB allows the system to take better advantage of current and future hardware technology.

The following are points to be noted while considering the use of SMB:

- SMB is available only for datasets defined for Non-Shared Resources (NSR), which excludes RLS access.
- SMB's choice of buffering technique and number of buffers is not dependent on the application design and hence not necessarily the optimum in all cases. In cases where dataset access is predictable by the application designer, it would be

better to let the application designer define the size of the buffers for each pool and the number for each.

- SMB typically allocates more buffers than without SMB. So the installation should allow larger virtual storage.
- To use SMB with linear datasets, CI access must be set in the ACB.

The IBM Redbook *VSAM Demystified* provides a sample program that extracts information from SMF record 64 and helps in identifying VSAM datasets with heavy access and good candidates for SMB.

CONCLUSION

The benefits of System Managed Buffering (SMB) can now be achieved just by defining the RECORD_ACCESS_BIAS parameter in the dataclass. It is observed that the key reason why SMB is not heavily used is a reluctance to convert to extended format. The use of extended format is strongly recommended, because it is a pre-requisite for many other useful features like data striping, compression, and extended addressibility. For most batch applications, use of SMB is recommended and can give performance improvements as high as 60%. When required, the buffering algorithms determined by VSAM can easily be overridden via JCL.

Sasirekha Cota
Tata Consultancy Services (India)

Read spool data from a C/C++ program – part 2

This month we conclude the code for this article.

```
/* Values for field "SSS2REAS" */
#define SSS2RENI 4    /* SSS2JEST zero, but SSS2DSN not null  */
#define SSS2REIP 8    /* SSS2SIPA and SSS2SIPN are mutually   */
#define SSS2RALO 12   /* Prior dataset still allocated        */
```

```
#define SSS2RDUP 16   /* SSS2SDUP on in SSS2SEL1 and wild      */
#define SSS2RJBI 20   /* SSS2JBIH < SSS2JBIL & SSS2SJBI on     */
#define SSS2RCRE 24   /* SSS2CREA has error & SSS2SCRE on      */
#define SSS2RLEN 28   /* SSS2LEN is less than SSS2SIZE         */
#define SSS2RTYP 32   /* SSS2TYPE is not valid                 */
#define SSS2RDES 36   /* SSS2DEST has error & SSS2SDST on      */
#define SSS2RJNM 40   /* SSS2JOBN has error & SSS2SJBN on      */
#define SSS2RFRM 44   /* SSS2FORM has error & SSS2SFRM on      */
#define SSS2RPGM 48   /* SSS2PGMN has error & SSS2SPGM on      */
#define SSS2RPRM 52   /* SSS2PRMO has error & SSS2SPRM on      */
#define SSS2RCLS 56   /* SSS2CLSL has error & SSS2SCLS on      */
#define SSS2RFCB 60   /* SSS2FCB  has error & SSS2SFCB on      */
#define SSS2RUCS 64   /* SSS2UCS  has error & SSS2SUCS on      */
#define SSS2RCHR 68   /* SSS2CHAR has error & SSS2SCHR on      */
#define SSS2RMO  72   /* SSS2MOD  has error & SSS2SMOD on      */
#define SSS2RFL  76   /* SSS2FLSH has error & SSS2SFLS on      */
#define SSS2RLPM 80   /* SSS2LMIN or SSS2LMAX is negative      */
#define SSS2RLPG 84   /* SSS2LMIN > SSS2LMAX & SSS2SLIN on     */
#define SSS2RDE2 88   /* SSS2DES2 has error & SSS2TYPE is      */
#define SSS2RVOL 92   /* SSS2VOL  has error & SSS2SVOL on      */
#define SSS2REYE 96   /* SSS2EYE does not have "SSS2"          */
#define SSS2RCTK 100  /* SSS2CTKN bad & SSS2SCTK on  @R05LOPI */
#define SSS2RBRO 104  /* SSS2SBRO on and SSS2TYPE is           */
#define SSS2RECJ 108  /* SSS2SCTK & SSS2SJBI are     @R05LOPI */
#define SSS2RODS 112  /* SSS2ODST has error &        @OW29707 */
#define SSS2RDCL 184  /* SSS2DCLS has error                    */
#define SSS2RDFR 188  /* SSS2DFOR has error                    */
#define SSS2RDPG 192  /* SSS2DPGM has error                    */
#define SSS2RDDS 196  /* SSS2DDES has error                    */
#define SSS2RDHR 200  /* Both SSS2DHLD & SSS2DRLS              */
#define SSS2RENM 240  /* No matching output            @R10AE */
#define SSS2RENS 244  /* Matching output not           @R10AE */
/* Values for field "SSS2TYPE" */
#define SSS2PUGE 1    /* Request type of Put/Get               */
#define SSS2COUN 2    /* Request type of Count.                */
#define SSS2BULK 3    /* Bulk modify request.                  */
/* Values for field "SSS2UFLG" */
#define SSS2SETC 0x80 /* Use SSS2CLAS as the new class         */
#define SSS2DELC 0x40 /* Delete selected dataset(s)            */
#define SSS2ROUT 0x20 /* Use SSS2DES2 as the new dataset       */
#define SSS2RLSE 0x10 /* Release selected datasets             */
/* Values for field "SSS2SEL1" */
#define SSS2SHLD 0x80 /* Select "HOLD/LEAVE" output (JES2);    */
#define SSS2SXWH 0x40 /* Select "hold for XWTR".  In a         */
#define SSS2SHOL 0xC0 /* Select from the hold queue.           */
#define SSS2SWTR 0x20 /* Select "WRITE/KEEP" output (JES2);    */
#define SSS2SAWT 0xE0 /* Select from all the above.            */
#define SSS2SCLS 0x10 /* Use SSS2CLSL as the class             */
#define SSS2SDST 0x08 /* Use SSS2DEST as a filter              */
#define SSS2SJBN 0x04 /* Use SSS2JOBN as a filter              */
```

```
#define SSS2SDUP 0x06 /* Use SSS2JOBN as a filter, but        */
#define SSS2SDU2 0x02 /* Give RC of SSS2DUPJ if duplicate     */
#define SSS2SJBI 0x01 /* Use SSS2JBIL and SSS2JBIH as         */
/* Values for field "SSS2SEL2" */
#define SSS2SPGM 0x80 /* Use SSS2PGMN as a filter             */
#define SSS2SFRM 0x40 /* Use SSS2FORM as a filter             */
#define SSS2SCRE 0x20 /* Use SSS2CREA as a filter             */
#define SSS2SPRM 0x10 /* Use SSS2PRMO as a filter             */
#define SSS2SIPA 0x08 /* Only select output which has an      */
#define SSS2SIPN 0x04 /* Only select output which has no      */
#define SSS2SFCB 0x02 /* Use SSS2FCB as a filter              */
#define SSS2SUCS 0x01 /* Use SSS2UCS as a filter              */
/* Values for field "SSS2SEL3" */
#define SSS2SSTC 0x80 /* Include Started Tasks (STCs)         */
#define SSS2STSU 0x40 /* Include Time Sharing Users (TSUs)    */
#define SSS2SJOB 0x20 /* Include batch jobs (JOBs)            */
#define SSS2SAPC 0x10 /* Include APPC output                  */
#define SSS2STYP 0xFF /* If none of these bits is on, then    */
/* Values for field "SSS2SEL4" */
#define SSS2SMOD 0x80 /* Use SSS2MOD as a filter              */
#define SSS2SFLS 0x40 /* Use SSS2FLSH as a filter             */
#define SSS2SAGE 0x20 /* Datasets selected must be at         */
#define SSS2SLIN 0x10 /* Use minimum and maximum line         */
#define SSS2SPAG 0x08 /* Use minimum and maximum page         */
#define SSS2SPRI 0x04 /* Select output based on priority      */
#define SSS2SVOL 0x02 /* Select output based on the volume    */
#define SSS2SCHR 0x01 /* Use Printer translation tables in    */
/* Values for field "SSS2SEL5" */
#define SSS2SCPN 0x80 /* Only select output which is not      */
#define SSS2SCTK 0x40 /* Use SSS2CTKN as a filter  @R05LOPI   */
#define SSS2SBRO 0x20 /* Application intent is to  @R05LOPI   */
#define SSS2SODS 0x10 /* Use SSS2ODST as a filter  @OW29707   */
/* Values for field "SSS2MSC1" */
#define SSS2CTRL 0x80 /* On - Processing complete             */
#define SSS2FSWB 0x60 /* Return token for SJFREQ calls in     */
#define SSS2FSWT 0x20 /* Return address of SWBTUREQ buffer    */
#define SSS2NJEH 0x10 /* Return address of NJE dataset        */
/* Values for field "SSS2DSP1" */
#define SSS2DKPE 0x80 /* Keep the dataset                     */
#define SSS2RHLD 0x40 /* Keep the dataset and make it         */
#define SSS2RNPR 0x20 /* Keep the dataset and leave it        */
#define SSS2DHLD 0x10 /* Hold the dataset                     */
#define SSS2DRLS 0x08 /* Release the dataset                  */
#define SSS2CHKP 0x04 /* Use SSS2RBA to checkpoint the        */
#define SSS2DNWR 0x02 /* Set writer name to a null value      */
#define SSS2RNPT 0x01 /* Leave the dataset       @OW36019     */
/* Values for field "SSS2WRTN" */
#define SSS2WOK  0    /* Processing successful                */
#define SSS2WERR 4    /* Processing failed                    */
/* Values for field "SSS2RET1" */
```

```
#define SSS2GNVA 0x80 /* An output group name has been        */
#define SSS2DSCL 0x40 /* Line count, page count, byte         */
#define SSS2DSF  0x20 /* First dataset in output group        */
#define SSS2DSC  0x30 /* Output group being continued         */
#define SSS2DSL  0x08 /* Last dataset in output group         */
#define SSS2IP   0x04 /* An Internet Protocol (IP)            */
#define SSS2BRST 0x02 /* BURST=YES specified                  */
#define SSS2OPTJ 0x01 /* OPTCD=J specified                    */
/* Values for field "SSS2RET2" */
#define SSS2NCHR 0x80 /* Selection using printer              */
#define SSS2NVOL 0x40 /* Selecting output based on a          */
#define SSS2NNHD 0x20 /* Returning addresses of NJE           */
#define SSS2NMOD 0x10 /* Selecting output based on Copy       */
#define SSS2NPRI 0x08 /* Selecting output in priority         */
#define SSS2NIPA 0x04 /* IP Address selection not             */
/* Values for field "SSS2RET3" */
#define SSS2RSTC 0x80 /* Dataset created by a started         */
#define SSS2RTSU 0x40 /* Dataset created by a time            */
#define SSS2RJOB 0x20 /* Dataset created by a batch job       */
/* Values for field "SSS2RET4" */
#define SSS2CPDS 0x80 /* Dataset has page mode data           */
#define SSS2SPUN 0x40 /* Dataset was spun at close            */
#define SSS2DSH  0x20 /* All datasets in the current          */
/* Values for field "SSS2RET5" */
#define SSS2RHLV 0x80 /* Dataset on "HOLD/LEAVE"  @OW32461    */
#define SSS2RXWH 0x40 /* Dataset on  "hold for    @OW32461    */
#define SSS2RHOL 0xC0 /* Dataset on one of the    @OW32461    */
#define SSS2RWTR 0x20 /* Dataset on "WRITE/KEEP"  @OW32461    */
#pragma pack(reset)
/*  Indicate to the compiler that standard OS linkage will be used.  */
#ifdef __cplusplus
 extern "OS" int GETSSS2(struct SSS2*, char*);
#else
 #pragma linkage (GETSSS2, OS)
#endif
#define MASK 0x80000000
#define MASK2 0x7FFFFFFF
#define BLANKS "                                                  \
                                                                      \
            "
 /*
  *  Create text unit values equivalent to:
  *  x'0055',x'0001',x'0008',c'        '
  *  x'0002',x'0001',x'002c',cl44' '
  *  x'005c',x'0001',x'0004',c'    '
  *  x'0'
  */
#define TU1 "\0\x55\0\x01\0\x08""        "
#define \
  TU2 "\0\x02\0\x01\0\x2c""                                            "
```

```
#define TU3 "\0\x5c\0\x01\0\x04""    "
main(int argc, char *argv[])
{
 int i,j,k,rc;
 char dsname[45];
 char ssname[9];
 char jobname[9];
 char ddname[9];
 char jobid[9];
 char jobid_save[9];
 char jobname_save[9];
 char stepname_save[9];
 char ddname_save[9];
 char creator_userid[9];
 char stepname[9];
 int  ddmatch;
 struct __S99struc parmlist;
 char *s[4];
 char return_ddname[9];
 char *m;
 char *c;
 FILE *stream;
 char filename[12];
 char input_record[1024];
 char output_buffer[1024];
 struct SSS2 sss2;
 char *strstrPos;
 char argv_buffer[1024];
 if (argc == 1)
 {
   printf("Program parm required.  Format is PARM=\
'JOBNAME=jobname{,JOBNO=JOBnnnnn}{,STEPNAME=stepname}{,\
DDNAME=ddname}\n");
   return 4;
 }
 strncpy(jobname,"        ",8);
 jobname[8] = 0;
 strncpy(jobid,"        ",8);
 jobid[8] = 0;
 strncpy(stepname,"        ",8);
 stepname[8] = 0;
 strncpy(ddname,"        ",8);
 ddname[8] = 0;
// Set the incoming PARM data to upper case
  memcpy(argv_buffer,argv[1],strlen(argv[1]));
 for (i = 0; i < strlen(argv[1]); i++)
 {
   argv_buffer[i] = argv_buffer[i] ] 64;
 }
 argv_buffer[i] = 0;
```

```
// Check for JOBNAME=
 strstrPos = strstr(argv_buffer,"JOBNAME=");
 if (strstrPos == NULL)
 {
   printf("Required JOBNAME= not specified in program PARM %s\n",
          argv[1]);
   return 8;
 }
 else
 {
   i = Ø;
   while (i < 8 && strstrPos[8+i] != 64 && strstrPos[8+i] != Ø &&
          strstrPos[8+i] != 1Ø7)
   {
     jobname[i] = strstrPos[8+i];
     i++;
   }
   if (strstrPos[8+i] != 64 && strstrPos[8+i] != Ø &&
       strstrPos[8+i] != 1Ø7)
   {
     printf("Invalid jobname specified\n");
     return 8;
   }
   else
   {
     printf("Specified jobname is %s\n",jobname);
   }
 }
/*
 * If we get to here, we have determined that the PARM data has
 * included a valid JOBNAME= specification.  We'll now scan the
 * remaining PARM data looking for JOBNO=, STEPNAME=, and DDNAME=
 * selection specifications.
 */
// Check for JOBNO=
 strstrPos = strstr(argv_buffer,"JOBNO=");
 if (strstrPos != NULL)
 {
   if (memcmp(strstrPos+6,"JOB",3) != Ø)
   {
     printf("Invalid JOB specification for JOBNO= in PARM %s\n",
            argv[1]);
     return 8;
   }
   else
   {
     i = Ø;
     while (i < 8 && strstrPos[6+i] != 64 && strstrPos[6+i] != Ø &&
            strstrPos[6+i] != 1Ø7)
     {
       jobid[i] = strstrPos[6+i];
       if (i >= 3)
       {
         if (jobid[i] < 240 ]] jobid[i] > 249)
         {
           printf("Invalid numeric specification for jobno in parm
%s\n",
                  argv[1]);
           return 8;
         }
       }
       i++;
     }
     if (strstrPos[6+i] != 64 && strstrPos[6+i] != 0 &&
         strstrPos[6+i] != 107)
     {
       printf("Invalid jobno specified\n");
       return 8;
     }
     else
     {
       printf("Specified jobno is %s\n",jobid);
     }
   }
 }
// Check for STEPNAME=
 strstrPos = strstr(argv_buffer,"STEPNAME=");
 if (strstrPos != NULL)
 {
   if (strstrPos[9] == 91 ]]
       strstrPos[9] == 123 ]]
       strstrPos[9] == 124 ]]
       (strstrPos[9] >= 193 &&
       strstrPos[9] <= 233))
   {
     i = 0;
     while (i < 8 && strstrPos[9+i] != 64 && strstrPos[9+i] != 0 &&
            strstrPos[9+i] != 107)
     {
       stepname[i] = strstrPos[9+i];
       i++;
     }
     if (strstrPos[9+i] != 64 && strstrPos[9+i] != 0 &&
         strstrPos[9+i] != 107)
     {
       printf("Invalid stepname specified\n");
       return 8;
     }
     else
     {
       printf("Specified stepname is %s\n",stepname);
     }
   }
   else
   {
     printf("Invalid stepname value for STEPNAME= in PARM %s\n",
            argv[1]);
     return 8;
   }
 }
// Check for DDNAME=
 strstrPos = strstr(argv_buffer,"DDNAME=");
 if (strstrPos != NULL)
 {
   if (strstrPos[7] == 91 ]]
       strstrPos[7] == 123 ]]
       strstrPos[7] == 124 ]]
       (strstrPos[7] >= 193 &&
       strstrPos[7] <= 233))
   {
     i = Ø;
     while (i < 8 && strstrPos[7+i] != 64 && strstrPos[7+i] != Ø &&
            strstrPos[7+i] != 1Ø7)
     {
       ddname[i] = strstrPos[7+i];
       i++;
     }
     if (strstrPos[7+i] != 64 && strstrPos[7+i] != Ø &&
         strstrPos[7+i] != 1Ø7)
     {
       printf("Invalid ddname specified\n");
       return 8;
     }
     else
     {
       printf("Specified ddname is %s\n",ddname);
     }
   }
   else
   {
     printf("Invalid ddname value for DDNAME= in PARM %s\n",
            argv[1]);
     return 8;
   }
 }
/*
 * Initialize the SSS2 control block to prepare it for the GETSSS2()
 * function call.
 */
 memset(&sss2, Ø, sizeof(sss2));
 memcpy(sss2.SSS2EYE, "SSS2", 4);
 sss2.SSS2VER =  SSS2CVER;
 sss2.SSS2LEN = (short int)(sizeof(sss2));
 sss2.SSS2SEL1 = Ø;
 sss2.SSS2SEL3 = Ø;
 sss2.SSS2TYPE = Ø;
 sss2.SSS2DSP1 = Ø;
 sss2.SSS2SEL3 = sss2.SSS2SEL3 ] SSS2SJOB ] SSS2SSTC ] SSS2STSU;
 sss2.SSS2TYPE = sss2.SSS2TYPE ] SSS2PUGE;
 sss2.SSS2DSP1 = sss2.SSS2DSP1 ] SSS2RNPT ] SSS2DKPE;
 memcpy(sss2.SSS2JOBN,jobname,8);
 if (memcmp(jobid,"        ",8) == Ø)
 {
/*
 * Indicate that selection should occur from all queues (SSS2SAWT)
 * and that the value in SSS2JOBN should be uses as a filter
 * (SSS2SJBN).
 */
   sss2.SSS2SEL1 = sss2.SSS2SEL1 ] SSS2SAWT ] SSS2SJBN;
 }
 else
 {
   memcpy(sss2.SSS2JBIL,jobid,8);
   memcpy(sss2.SSS2JBIH,jobid,8);
/*
 * Indicate that selection should occur from all queues (SSS2SAWT),
 * that the value in SSS2JOBN should be used as a filter (SSS2SJBN),
 * and that SSS2JBIL and SSS2JBIH should be used as filters
 * (SSS2SJBI).
 */
   sss2.SSS2SEL1 = sss2.SSS2SEL1 ] SSS2SAWT ] SSS2SJBN ] SSS2SJBI;
 }
 strncpy(jobname_save,"        ",8);
 jobname_save[8] = Ø;
 strncpy(jobid_save,"        ",8);
 jobid_save[8] = Ø;
 strncpy(stepname_save,"        ",8);
 stepname_save[8] = Ø;
 strncpy(ddname_save,"        ",8);
 ddname_save[8] = Ø;
 i = GETSSS2(&sss2,ssname);
 if (i == Ø)
 {
   printf("Subsystem name is %s\n",ssname);
 }
/*
 * Loop until GETSSS2() does not return a new SSS2.
 */
 while (i == Ø)
 {
   jobname_save[8] = 0;
   jobid_save[8] = 0;
   ddname_save[8] = 0;
   dsname[44] = 0;
   memcpy(jobname_save, sss2.SSS2JOBR, 8);
   memcpy(jobid_save, sss2.SSS2JBIR, 8);
   memcpy(ddname_save, sss2.SSS2DDND, 8);
   memcpy(stepname_save, sss2.SSS2STPD, 8);
   memcpy(dsname, sss2.SSS2DSN, 44);
   ddmatch = 0;
   if (memcmp(ddname,"        ",8) != 0 &&
       memcmp(ddname,sss2.SSS2DDND,8) == 0 &&
       memcmp(stepname,"        ",8) != 0 &&
       memcmp(stepname,sss2.SSS2STPD,8) == 0)
   {
     printf(" \n");
     printf("Jobname: %s  Job#: %s  Stepname: %s  DDname: %s  DSname:
%s\n",
            jobname_save, jobid_save, stepname_save, ddname_save,
dsname);
     ddmatch = 1;
   }
   else if (memcmp(ddname,"        ",8) != 0 &&
            memcmp(ddname,sss2.SSS2DDND,8) == 0 &&
            memcmp(stepname,"        ",8) == 0)
   {
     printf(" \n");
     printf("Jobname: %s  Job#: %s  Stepname: %s  DDname: %s  DSname:
%s\n",
            jobname_save, jobid_save, stepname_save, ddname_save,
dsname);
     ddmatch = 1;
   }
   else if (memcmp(stepname,"        ",8) != 0 &&
            memcmp(stepname,sss2.SSS2STPD,8) == 0 &&
            memcmp(ddname,"        ",8) == 0)
   {
     printf(" \n");
     printf("Jobname: %s  Job#: %s  Stepname: %s  DDname: %s  DSname:
%s\n",
            jobname_save, jobid_save, stepname_save, ddname_save,
dsname);
     ddmatch = 1;
   }
   else if (memcmp(stepname,"        ",8) == 0 &&
            memcmp(ddname,"        ",8) == 0)
   {
     printf(" \n");
     printf("Jobname: %s  Job#: %s  Stepname: %s  DDname: %s  DSname:
%s\n",
            jobname_save, jobid_save, stepname_save, ddname_save,
dsname);
     ddmatch = 1;
   }
/*
 * If ddmatch != 1, the returned SSS2 data does not match with the
 * selection criteria specified on the PARM data.  Bypass this entry.
 */
   if (ddmatch != 1)
   {
     goto RD_NEXT;
   }
/*
 *  If we get to here, we have a JES spool dataset that matches the
 *  PARM data selection criteria.  We can dynmically allocate the
 *  dataset and read its contents.
 */
   creator_userid[8] = Ø;
   memcpy(creator_userid, sss2.SSS2CRER, 8);
   j = (int)sss2.SSS2MLRL;
// printf("  Creator: %s  Max LRECL: %d\n",creator_userid,j);
   printf(" \n");
/*
 *  Build the dynamic allocation parameter data and allocate the
 *  JES spool dataset.
 */
   memset(&parmlist, Ø, sizeof(parmlist));
   parmlist.__S99RBLN = 2Ø;
   parmlist.__S99VERB = 1;          // verb for dsname allocation
   parmlist.__S99FLAG1 = ØxØØØØ;
   parmlist.__S99TXTPP = s;         // pointer to pointer to text unit
// Create the return ddname text unit.
   m = (char *)calloc(1,sizeof(TU1));
   memcpy(m,TU1,sizeof(TU1));
   s[Ø] = m;
// Create the dsname text unit.
   m = (char *)calloc(1,sizeof(TU2));
   memcpy(m,TU2,sizeof(TU2));
   memcpy(m+6, dsname, 44);
   s[1] = m;
// Create the subsystem name text unit.
   m = (char *)calloc(1,sizeof(TU3));
   memcpy(m,TU3,sizeof(TU3));
   memcpy(m+6, ssname, 4);
   s[2] = m;
// Copy the address of the spool dataset browse token text unit.
   s[3] = (char *)(sss2.SSS2BTOK);
   s[3] = (char *)((long unsigned) (s[3]) | MASK);
   rc = svc99(&parmlist);
/*
```

```
 *  If rc == Ø, the spool dataset has been successfully allocated.
 *  Let's open it and read its contents.
 */
   if (rc == Ø)
   {
     return_ddname[8] = Ø;
     memcpy(return_ddname, s[Ø]+6, 8);
//   printf("SYSOUT dataset allocated to DDname: %s\n",return_ddname);
     filename[11] = Ø;
     m = &return_ddname[Ø];
     c = &filename[Ø];
     strncpy(c,"DD:",3);
     strncpy(c+3,m,8);
     stream = fopen(filename,"rb, type=record");
     if (stream == NULL)
     {
       printf("Unable to open %s %s\n",filename,dsname);
     }
     else
     {
       strncpy(input_record,BLANKS,132);
       i = fread(input_record, 1, 1Ø24, stream);
       while (!(ferror(stream)) && !(feof(stream)))
       {
//       printf("%s",input_record);
         strncpy(output_buffer,BLANKS,132);
         output_buffer[132] = Ø;
         strncpy(output_buffer, input_record, 132);
         printf("%s\n",output_buffer);
         strncpy(input_record,BLANKS,132);
         i = fread(input_record, 1, 1Ø24, stream);
       }
       fclose(stream);
     }
     parmlist.__S99RBLN = 2Ø;
     parmlist.__S99VERB = 2;           // verb for unallocation
     parmlist.__S99FLAG1 = ØxØØØØ;
     parmlist.__S99TXTPP = s;          // pointer to pointer to text unit
     s[Ø] = (char *)((long unsigned) (s[Ø]) | MASK);
     s[Ø][1] = 1;                      // set text unit verb to ddname
     rc = svc99(&parmlist);
     if (rc != Ø)
     {
       printf(" DEALLOC Error code = %d   Information code = %d\n",
              parmlist.__S99ERROR, parmlist.__S99INFO);
     }
     s[Ø] = (char *)((long unsigned) (s[Ø]) & MASK2);
   }
   else
   {
     printf(" ALLOC Error code = %d   Information code = %d\n",
            parmlist.__S99ERROR, parmlist.__S99INFO);
   }
   free(s[Ø]);
   free(s[1]);
   free(s[2]);
RD_NEXT:
   i = GETSSS2(&sss2,ssname);
 }
 return(Ø);
}
```

Rudy Douglas
System Programmer (Canada)

Monitoring HFS performance

This article will focus on monitoring the performance of HFS and is a sequel to a previous article (see 'Monitoring USS performance from z/OS – an introduction', *MVS Update*, issues 213 and 214, June and July 2004). The primary focus is on understanding HFS performance metrics as well as on monitoring and managing critical HFS file systems. The sample technique for collecting and analysing HFS performance data will be demonstrated. Tuning recommendations will be briefly discussed too.

INTRODUCTION

Starting with earlier releases, but primarily in OS/390 R5, many of the base MVS components began using OS/390 Unix System Services (USS) and, therefore, Unix file services as supported by IBM's Hierarchical File System (HFS). These components and facilities include TCP/IP, Notes Domino, WebServer, ERP applications, and many others. As already known, you will not be able to avoid HFS any longer because it has now become an integral part of OS/390. It is a fact that each new release of DFSMS continues to build on the previous version to provide enhanced storage management, data access, and device support.

For example, in OS/390 V2R7 and DFSMS 1.5 there were dramatic changes to the way OS/390 HFS file systems work. The most notable of these changes are the addition of HFS global buffers, the ability to perform HFS I/O asynchronously, and the ability to mount multi-volume file systems. These enhancements, and others, resulted in dramatic HFS performance improvements. Thus, the proper tuning of OS/390 HFS is becoming a critical and non-trivial component of overall OS/390 performance. Many of OS/390's performance problems can be traced to poor decisions related to HFS datasets. The good news is that DFSMS 1.5 vastly improved HFS performance, and added some new controls for tuning these important datasets. However, the bad news is that most of these new controls have not been well documented and explained. This article will attempt to provide some practical suggestions for monitoring and improving HFS performance. It provides you with the information you need to understand and evaluate the performance of the HFS file system, along with practical hints and tips. It provides sufficient information for you to start monitoring HFS and evaluate its performance in your DFSMS environment.

ONLINE MONITORING OF HFS PERFORMANCE

When it comes to online performance monitoring tools for HFS file systems there is currently only one option available within the standard IBM's toolkit – RMF Monitor II HFS report. It is invoked by specifying 5 on the I/O Report Selection menu of the RMF Monitor II panel. The HFS File System statistics report it produces provides data for basic performance analysis of HFS, which enables you to identify potential problems and bottlenecks within the HFS component and to take corrective actions. The contents of the report as well as its field descriptions are described in the *RMF: Report Analysis* (SC33-7991) manual. A systems programmer is well aware of the fact that when setting report options (RO) only one file system name can be selected, and, if you try to select several files, RMF will complain and not a single file system name will be selected at all! On the other hand, it was found that RMF Monitor II HFS report is not as informative as it should be because it does not help you easily to identify your most I/O active files. This limitation was the initial impetus that prompted me to look for a tool/procedure that would allow me to get information about all mounted HFS file systems in a single run. To help alleviate the burden of monitoring HFS performance, as well as overcoming RMF's shortcomings, a simple yet easy to use REXX procedure was constructed. The USS confighfs query HFS global statistics command was used to display a system snapshot for all HFS datasets. In addition, the USS command df with option –S was used because it provides SMF I/O accounting for mounted files. It was also used to obtain the mount point for an HFS, which was needed for the confighfs command in order to get complete statistics for the dataset. The full meaning of the command and its invoking argument list can be obtained from *Unix System Services Command Reference* (SA22-7802).

HFSPM EXEC

```
/* REXX ***********************************************************
   Procedure: HFSPM
   Description: Get current information on HFS performance
   Install: - Download BPXWUnix function (a part of "REXX Function
              Package for REXX in OpenEdition") from the IBM's "USS
              Tools and Toys page"
            - Restore it using the TSO/E receive inda() command.
            - Place this where REXX EXECs can be found.
  *****************************************************************/
signal ON ERROR
Address TSO
userid=SYSVAR(SYSUID)
outds  =userid||'.cnf.out'               /* Change dataset name   */
x = MSG('ON')                            /* to fit your standards */
if SYSDSN(outds) = 'OK'
Then "DELETE "outds" PURGE"
"ALLOC FILE(PRC) DA("outds")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
   " REUSE LRECL(7Ø) RECFM(F B)"
 arg hlq
 if hlq = "" then HLQ = 'SYSTMØ5.USER'
Address ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"hlq".LOAD') STACK"
call syscalls 'ON'
/*---------------------------------------------------------------*/
```

```
/* Return USS information                                            */
/*-------------------------------------------------------------------*/
Address SYSCALL
'uname sys.'
/*-------------------------------------------------------------------*/
/* Print headers and labels                                          */
/*-------------------------------------------------------------------*/
sis.1 = left('HFS Performance report - produced on:',37,),
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' '),
        ||left(time(),10)
sis.2 = left(' ',1)
sis.3 = left('System identification:',22)
sis.4 = left('Sysname: ',11)||sys.U_SYSNAME
sis.5 = left('Version: ',11)||sys.U_VERSION
sis.6 = left('Release: ',11)||left(sys.U_RELEASE,10)
sis.7 = left('Node   : ',11)||left(sys.U_NODENAME,10)
sis.8 = left('Hardware:',11)||left(sys.U_MACHINE,10)
sis.9 = left(' ',1,' ')
/*-------------------------------------------------------------------*/
/* Get HFS data                                                      */
/*-------------------------------------------------------------------*/
Address SH
/*-------------------------------------------------------------------*/
/* Construct confighfs command (fixed part of it). NOTE:             */
/* Unlike most z/OS Unix commands, which reside in /bin,             */
/* confighfs is found in the /usr/lpp/dfsms/bin directory            */
/*-------------------------------------------------------------------*/
 fix  ='cd /usr/lpp/dfsms/bin;./confighfs -q '
/*-------------------------------------------------------------------*/
/* Call display file command (df) to get file mount point            */
/* and  I/O activity since mounted                                   */
/*-------------------------------------------------------------------*/
  call BPXWUnix "df -S",,out.
  s = 1
  dfrc = rc
   If dfrc  <>0 Then Do
     Say "Return Code         =" rc
     Say "OMVS Return Value   =" retval
     Say "OMVS Return Code    =" errno
     Say "OMVS Reason Code    =" errnojr
   End
  Do i = 2 to OUT.0                 /* Process each entry returned*/
  parse var out.i  mount . '(' HFS ')' rawdata .
  i = i + 1;
  read      =word(out.i,5); i =i + 1;       /* Read count          */
  write     =word(out.i,5); i =i + 1;       /* Write count         */
  ioblk     =word(out.i,6); i =i + 1;       /* Dir I/O bk.total    */
  reblk     =word(out.i,6); i =i + 1;       /* Dir I/O bkread      */
  wrblk     =word(out.i,6); i =i + 1;       /* Dir I/O bkwritten   */
```

```
   byread    =word(out.i,7); i =i + 1;       /* Total bytes read   */
   bywrite   =word(out.i,7)                  /* Total bytes written*/
   HFS      = left(HFS,25)                   /* Filesystem name    */
  if (HFS ¬= "/tmp") & (HFS ¬= "/dev")  then
   do
  Rw.s  =left('PERFORMANCE DATA FOR FILE:',27)||left(HFS,25); s=s+1;
  Rw.s  =left('Part 1',6);  s=s+1;
 /*-----------------------------------------------------------------*/
 /* Construct confighfs command (variable part of it is added)      */
 /*-----------------------------------------------------------------*/
  cmd =fix||mount
 /*-----------------------------------------------------------------*/
 /* Call confighfs command and process each entry returned          */
 /*-----------------------------------------------------------------*/
  call BPXWUnix cmd,,ott.
   cfgc = rc
    If cfgc  <>Ø Then Do
      Say "Return Code          =" rc
      Say "OMVS Return Value    =" retval
      Say "OMVS Return Code     =" errno
      Say "OMVS Reason Code     =" errnojr
    End
    Do k = 3 to OTT.Ø - 2
  VSTOR =strip(word(ott.k,3),L,'_'); k=k+2;   /* Virtual Storage   */
  FSTOR =strip(word(ott.k,3),L,'_'); k=k+2;   /* Fixed   Storage   */
  Lch   =strip(word(ott.k,4),L,'_'); k=k+1;   /* Lookup cache hit  */
  Lcm   =strip(word(ott.k,4),L,'_'); k=k+1;   /* Lookup cache miss */
  h6   =hitr(Lch Lcm)
  Fdph  =strip(word(ott.k,5),L,'_'); k=k+1;   /* 1st data page hit */
  Fdpm  =strip(word(ott.k,5),L,'_'); k=k+2;   /* 1st data page miss*/
  h5   =hitr(Fdph Fdpm)
  titl  = ott.k; k = k + 1;
 /*-----------------------------------------------------------------*/
 /* Get Storage allocated and buffer pool statistics                */
 /*-----------------------------------------------------------------*/
  Rw.s  =left('Current Buffer pool use:',4Ø); s=s+1;
  Rw.s  =left('Virtual Storage:',16)||right(vstor,5); s=s+1;
  Rw.s  =left('Fixed   Storage:',16)||right(fstor,5); s=s+1;
  Rw.s  =left(' ',3,' ' ,)||left(' ',29,' '),
         left('Already',7)  left('Not al.',7); s=s+1;
  Rw.s  =left('Pool',6)||left('Size',8),
         ||left('#DS',4)||left('BP_pages',9)||left('Fixed',7),
         ||left('fixed',8)||left('fixed',8); s=s+1;
  Rw.s  =left('-',5Ø,'-'); s=s+1;
     do f = 1 to 4
     p.f   = word(ott.k,1)
     ps.f  = strip(word(ott.k,2),L,'_')
     pds.f = strip(word(ott.k,3),L,'_')
     pbp.f = strip(word(ott.k,4),L,'_')
     pf.f  = strip(word(ott.k,5),L,'_')
     paf.f = strip(word(ott.k,6),L,'_')
     pnf.f = strip(word(ott.k,7),L,'_')
  Rw.s =  right(p.f,3),                /* Pool number            */
     right(ps.f,6),                    /* Pool size              */
     right(pds.f,6),                   /* Data spaces in pool    */
     right(pbp.f,6),                   /* # pages in pool - in use*/
     right(pf.f,6),                    /* Permanently fixed pages */
     right(paf.f,7),                   /* Already fixed pages    */
     right(pnf.f,7); s=s+1;            /* Not already fixed      */
     k = k+1;
      end
    k = k+3
  FSsz =strip(substr(ott.k,19,11),L,'_')    /* File system         */
  k= k+2
  Used =strip(word(ott.k,3),L,'_'); k=k+2;  /* Used pages          */
  Apgs =strip(word(ott.k,3),L,'_'); k=k+2;  /* Attribute pages     */
  Cpgs =strip(word(ott.k,3),L,'_'); k=k+2;  /* Cached pages        */
  Sio  =strip(word(ott.k,4),L,'_'); k=k+1;  /* Seq I/O reqs        */
  Rio  =strip(word(ott.k,4),L,'_'); k=k+1;  /* Random I/O reqs     */
  Lh   =strip(word(ott.k,3),L,'_'); k=k+1;  /* Look-up hit         */
  Lm   =strip(word(ott.k,3),L,'_'); k=k+1;  /* Look-up miss        */
  h2   =hitr(Lh Lm)
  Fph  =strip(word(ott.k,4),L,'_'); k=k+1;  /* 1st page hit:       */
  Fpm  =strip(word(ott.k,4),L,'_'); k=k+1;  /* 1st page miss       */
  h1   =hitr(Fph Fpm)
  Ixnt =strip(word(ott.k,4),L,'_'); k=k+1;  /* Index new tops      */
  Ixsp =strip(word(ott.k,3),L,'_'); k=k+1;  /* Index splits        */
  Ixjo =strip(word(ott.k,3),L,'_'); k=k+1;  /* Index joins         */
  Ixrh =strip(word(ott.k,4),L,'_'); k=k+1;  /* Index read hit      */
  Ixrm =strip(word(ott.k,4),L,'_'); k=k+1;  /* Index read miss     */
  h3   =hitr(Ixrh Ixrm)
  Ixwh = strip(word(ott.k,4),L,'_'); k=k+1; /* Index write hit     */
  Ixwm = strip(substr(ott.k,19,20),L,'_');  /*  Index write miss   */
  h4   =hitr(Ixwh Ixwm); k=k+1;
  Rflg = strip(substr(ott.k,19,20),L,'_'); k=k+1; /* RFS flags     */
  Rerr = strip(substr(ott.k,19,20),L,'_'); k=k+2; /* RFS errors    */
  Memc = strip(word(ott.k,3),L,'_'); k=k+1  /* Member count        */
  Sync = strip(word(ott.k,3),L,'_'); k=k+1  /* Sync interval       */
 /*---------------------------------------------------------------*/
 /* Process File attributes (formatted to match RMF display)      */
 /*---------------------------------------------------------------*/
  Rw.s   =left(' ',3,' ');  s=s+1;
  Rw.s   =left('Part 2',6); s=s+1;
  Rw.s   =left('File attributes:',30); s=s+1;
  Rw.s   =left(' ',3,' ' ,); s=s+1;
  Rw.s   =left('---- File allocation (pages): --',34)||left(' ',3,' '),
          ||left('---- Index Events --',21); s=s+1;
  Rw.s   =left('System',9)    left(' ',1,' ') right(FSsz,7),
          ||left(' Used ',8)||left(' ',1,' ')||right(Used,6),
          ||left(' ',10,' '),
          ||left('New tops',9)||right(ixnt,4); s=s+1;
 Rw.s  = left('Attr.dir',9)||left(' ',3,' ')||right(Apgs,6),
          ||left('  Cached',8)||left(' ',1,' ')||right(Cpgs,7),
          ||left(' ',1Ø,' '),
          ||left('Splits',9)||right(ixsp,4); s=s+1;
 Rw.s  = left('Members ',9)||left(' ',3,' ')||right(Memc,6),
          ||left('  Sync.int',1Ø)||Left(' ',1,' ')||right(sync,12),
          ||left(' ',3,' ')||left('Joins',9)||right(ixjo,4); s=s+1;
 Rw.s  = left('RFS flag',9)||left(' ',3,' ')||right(rflg,6),
          ||left(' ',2,' ')||left('RFS error',9),
          ||right(rerr,5); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
/*------------------------------------------------------------------*/
/* Process File's current I/Os (formatted to match RMF display)    */
/*------------------------------------------------------------------*/
 Rw.s   =left('Part 3',6); s=s+1;
 Rw.s   =left('Current I/O activity count:',4Ø); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
 Rw.s   =left(' ',9,' ')||left('-- File --',12),
          ||left('-- Metadata --',16)||left(' ',3,' '),
          ||left('-- Index --',13); s=s+1;
 Rw.s   =left('Cache',1Ø)||right(Fph,5)||left(' ',8,' '),
          ||right(Lh,6)||left(' ',8,' '),
          ||left('read:',5)||right(ixrh,5)||left(' ',2,' '),
          ||left('write:',7)||right(ixwh,3); s=s+1;
 Rw.s   =left('DASD',1Ø)||right(Fpm,5)||left(' ',8,' '),
          ||right(Lm,6)||left(' ',8,' '),
          ||left('read:',5)||right(ixrm,5)||left(' ',2,' '),
          ||left('write:',7)||right(ixwm,3); s=s+1;
 Rw.s   =left('Hit Ratio ',1Ø)||right(h1,5)||left(' ',8,' '),
          ||right(h2,6)||left(' ',13,' ')||right(h3,5)||left(' ',6,' '),
          ||right(h4,6); s=s+1;
 Rw.s   =left('Seq.I/O',1Ø)||right(sio,5); s=s+1;
 Rw.s   =left('Random',1Ø)||right(rio,5); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
/*------------------------------------------------------------------*/
/* File I/O activity since mounted - also found in SMF 92 rec.     */
/*------------------------------------------------------------------*/
 Rw.s   =left('Part 4',6); s=s+1;
 Rw.s   =left('File I/O activity since mounted:',4Ø); s=s+1;
 Rw.s   =left(' ',3,' ' ,); s=s+1;
 Rw.s  = left(' ',18,' ') left('- Dir I/O blocks -',25),
          left('Total bytes',12); s=s+1;
 Rw.s  = left('Reads',8) left('Writes',8),
          left('Total',8) left('Read',6)  left('Write',8),
          left('read',6)  left('written',7); s=s+1;
 Rw.s  = left('-',6Ø,'-'); s=s+1;
 Rw.s = right(read,5) ,                /* Number of reads          */
        right(write,8) ,               /* Number of writes         */
        right(ioblk,8) ,               /* Number dir. I/O block    */
        right(reblk,7) ,                /* Number read I/O blocks  */
        right(wrblk,7) ,                /* Number write I/O blocks */
        right(byread,8) ,               /* Total number bytes read */
        right(bywrite,8); s=s+1;        /* Total num.bytes written */
/*-----------------------------------------------------------------*/
/* Cache usage since mounted                                       */
/*-----------------------------------------------------------------*/
 Rw.s   =left(' ',3,' '); s=s+1;
 Rw.s   =left('Part 5',6); s=s+1;
 Rw.s   =left('Cache usage since mounted:',40); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
 Rw.s   =left(' ',7,' ')||left('File I/O count',16),
         ||left('Metadata I/O count',18); s=s+1;
 Rw.s   =left('-',41,'-'); s=s+1;
 Rw.s   =left('Cache',10)||right(Fdph,5)||left(' ',12,' '),
         ||right(Lch,6); s=s+1;
 Rw.s   =left('DASD',10)||right(Fdpm,5)||left(' ',12,' '),
         ||right(Lcm,6); s=s+1;
 Rw.s   =left('Hit Ratio ',10)||right(h5,5)||left(' ',12,' '),
         ||right(h6,6); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
     End
     End
    End /* main*/
call syscalls 'OFF'
/*----------------------------------------------------------------*/
/* Write out USS System info and HFS info data                    */
/*----------------------------------------------------------------*/
Address ISPEXEC "LIBDEF ISPLLIB";
Address TSO
"EXECIO * DISKW PRC (STEM sis.)"
"EXECIO * DISKW PRC (STEM Rw.)"
/*----------------------------------------------------------------*/
/* Close & free allocated report file; then display result        */
/*----------------------------------------------------------------*/
"EXECIO 0 DISKW PRC (FINIS "
  "free FILE(PRC)"
    Address ISPEXEC
   "ISPEXEC BROWSE DATASET('"outds"')"
  exit 0
  /*---------------------------------------------------------------*/
 /* Error exit routine                                             */
  /*---------------------------------------------------------------*/
 ERROR: say 'The following command produced non-zero RC =' RC
        say SOURCELINE(SIGL)
        exit
HITR:
/* REXX - calculate Hit ratio */
arg a b
   SELECT
```

```
     when a ¬=  Ø Then do
        t  = a  + b
        hr = trunc((a/t)*1ØØ, 2)
       end
      otherwise  hr= Ø
    END
  return hr
```

The data returned by this procedure pertains to each HFS file system mounted and is grouped into five parts.

PERFORMANCE DATA FOR FILE OMVS.ETC

```
Part 1
Current Buffer pool use:
Virtual Storage: 4683
Fixed   Storage:    Ø
                                 Already Not al.
Pool  Size    #DS BP_pages Fixed  fixed   fixed
--------------------------------------------------
  1      1      1   4Ø83      Ø       Ø  686335
  2      4      1      8      Ø       Ø     2Ø4
  3     16      1     8Ø      Ø       Ø     328
  4     64      1    512      Ø       Ø     616

Part 2
File attributes:

---- File allocation (pages): --    ---- Index Events ---
System       1Ø8ØØ  Used       6Ø1          New tops    Ø
Attr.dir        3Ø  Cached       6          Splits      Ø
Members        295  Sync.int  6Ø(seconds)   Joins       Ø
RFS flag        43  RFS error    Ø

Part 3
Current I/O activity count:

         -- File --  -- Metadata --      -- Index --
Cache      1124          3667       read: 8346  write: 74Ø
DASD        161          1557       read:  49Ø  write:   Ø
Hit Ratio 87.47         7Ø.19            94.45      1ØØ.ØØ
Seq.I/O       Ø
Random      158

Part 4
File I/O activity since mounted:

                   - Dir I/O blocks -      Total bytes
Reads     Writes   Total     Read    Write    read   written
```

```
-------------------------------------------------------------
 5Ø9Ø      159    33533    714Ø     159 22369373    1Ø423

Part 5
Cache usage since mounted:

       File I/O count  Metadata I/O count
------------------------------------------
Cache     15572           1777Ø3
DASD       1356           126692
Hit Ratio 91.98            58.37
```

Current buffer pool use (part 1) displays the number of virtual and permanently-fixed storage pages assigned to all four HFS I/O buffer pools. Comparing these actual usage numbers with the VIRTUAL and FIXED values may help you to determine when to adjust the storage thresholds. The table that follows is buffer pool assignment and it shows statistics for each of four buffer pools. This information is not provided by the RMF Monitor II HFS report. The following data is returned:

- *Pool* is the buffer pool ID. It designates one of the four HFS buffer pools. Under the current implementation this number is always in the range 1 to 4.
- *Size* is the buffer size for this pool (in pages 4KB, 16KB, 64K, and 256KB).
- *#DS* is the number of data spaces allocated to support the buffers in this buffer pool – usually set to 1 (one for each pool) except in the case of a very active system. HFS initially allocates to OMVS kernel four 2GB data spaces for four different buffer pools:
  - 4KB pool for small files (files that are less than or equal to 4KB in size), metadata, and most random requests.
  - 16KB and 64KB pool for intermediate sizes, for sequential file I/O if the system determines that this is an optimal buffer size, and for random file I/O if the block size of the file best fits in the buffer.
  - 256KB for large files, sequential file I/O if the system determines that this is an optimal buffer size, and for

random file I/O if the block size of the file best fits in the buffer.

    – additional data spaces are allocated as needed.

- *BP_pages* is the number of virtual pages in this buffer pool currently in use.
- *Fixed* is the number of permanently fixed pages in this buffer pool.

  The last two columns are the measures of effectiveness of the page-fixed storage assigned to the buffer pools. For each buffer pool, the sum of these two columns will show the total number of buffer read and write requests – this is not a physical I/O count.
- *Already_fixed* is the number of times a buffer was already fixed prior to an I/O request in this buffer pool. This is a counter that is never decremented. By dividing this column by *sum* we will get the hit rate percentage for the fixed storage assigned to each pool.
- *Not_already_fixed* is the number of times a buffer was not already fixed prior to an I/O request in this buffer pool. This is a counter that is never decremented.

File attributes (part 2) displays the allocation data (in pages) for each mounted HFS file system and index events. The first three space items (system, used, and attribute directory) may be used to make capacity-related decisions regarding your HFS datasets:

- *System* is the amount of storage allocated to this HFS.
- *Used* is the amount of storage pages internally used within HFS for data files, directories, and HFS internal structures (like the attribute directory).
- *Attr. Dir* is the amount of storage used for the attribute directory (AD). This number is included in the *Used* field. The attribute directory is the internal HFS structure (index) containing attribute information about individual file system objects as well as attributes of the file system itself.

- *Cached* is the amount of data within the HFS that has been moved into the virtual storage cache. This information may be used as a measure of activity of the file within the HFS. If enough virtual storage is made available, more pages of an HFS will be moved into cache as the files become more active.

  The index statistics are relative to all of the indices in the HFS dataset. The attribute directory (AD) is one index (the largest) but each directory (including the root) is also an index. The activity of index pages within the dataset is represented by three items:

  – *New tops* number shows how often HFS added a new level to its index structure, that is when the index is growing.

  – *Splits* number shows how often an index page was split into two pages because new records were inserted. This gives an idea of how much insertion activity there has been for the index structure.

  – *Joins*, contrary to a *new tops* and *splits* (both indicative of index growth), represents a shrinking of the index page. It shows how often HFS was able to combine two index pages into one, because enough index records had been deleted in the two pages.

The next four items in this part of report are not available if you use RMF Monitor II HFS report:

- *Members* is the number of nodes (entries) in the file system that have been used to represent files and directories. In fact, it is a crude approximation of the number of logical files contained within a dataset. Please refer to APAR OW39886 (USS confighfs command shows an incorrect member count) if your member count is incorrect (too high).

- *RFS flag* is HFS internal information and it shows the attributes of the file system at mount time.

- *RFS error* is HFS internal information that reports the errors that may have occurred while sync daemon was trying to harden data for this dataset. Watch for any non-zero value – it should be investigated before you lose more data.

- *Sync interval* is the interval used by the sync daemon for hardening all file data for a file system. Please remember that sync daemon uses vfs_sync operation in order to write to disk (or otherwise stabilizes) all changed data in a buffer cache for files in a mounted file system whenever all the HFS buffers of the file are filled. One should be very cautious when setting this parameter: if you specify SYNCDEFAULT=0 you will degrade your system performance by turning off deferred writes, but you will ensure data integrity of the application using that particular filesystem. (Remember that syncs are done at the HFS level, not file level, ie the sync process of the sync daemon will occur independently on each HFS file system. Even if sync intervals of all the HFS file systems are the same, the sync point of each HFS file system will be different.) On the other hand, be aware that if the system crashes before sync interval completes, the user or metadata written since the last sync interval is lost.

Current I/O activity count (part 3) displays a snapshot of I/O accounting data for all mounted files, thus providing the data for understanding the throughput achieved by HFS, which, in turn, allows you to optimally use system resources. The reporting is done on three levels: file, metadata, and index.

File I/O count items:

- *Cache* is the number of times the first page of a data file was requested and found in virtual storage (cache).
- *DASD* is the number of times the first page of a data file was requested and was not found in virtual storage (cache), thus, I/O was required.
- *Hit Ratio* is the percentage of cache-found requests based on the total number of requests.
- *Seq I/O reqs* is the number of sequential file data I/O

requests that have been issued. A sequential I/O is one of a series of I/Os to read or write a data file, where the first I/O started at the first byte of the file and each subsequent I/O was for the next sequential set of bytes. This is not meant to imply that actual disk I/O was required; the data may have resided in cache.

- *Random I/O reqs* is the number of random file data I/O requests that have been issued. A random I/O is an I/O that does not read or write the start of a file, and was not preceded by an I/O that read or wrote the immediately-preceding set of bytes. This is not meant to imply that actual disk I/O was required; the data may have resided in cache.

Metadata I/O count:

- *Cache* is the number of times the metadata for a file was found in virtual storage (cache) during file look-up (look-up hit).
- *DASD* is the number of times the metadata for the file was not found in virtual storage (cache) during file look-up and an index call was necessary, which may have resulted in I/O (look-up miss).
- *Hit Ratio* is the percentage of cache-found requests based on the total number of requests.

Index I/O count:

- *Cache* is the number of index page read or write hits.
- *DASD* is the number of index page read or write misses.
- *Hit Ratio* is the percentage of cache-found requests based on the total number of requests.

Note: these index hit/miss items report how efficient virtual storage cacheing was in providing the data needed without performing physical access to the data. When combining these indicators with cache items for the first page of a data file and metadata, one can get an idea about the total activity at the dataset level. I have noticed that the RMF Monitor II HFS report does not make any distinction between read and write activity. This is a serious shortcoming since any non-zero value in the *write miss* column should be taken very seriously because it indicates that the virtual storage cache is so heavily overloaded that output buffers cannot be provided.

The next two parts of this report are totally absent from the RMF Monitor II HFS report as well as from post-processor's HFS report.

The file I/O activity since mounted (part 4) report provides summary information on HFS I/O activity (number of reads/ writes, directory activity, and number of bytes read/written) since the time the file system was mounted. It was found to be very useful because it can help you to identify high I/O activity files quickly.

The cache usage since mounted (part 5) report provides the data that can be used to analyse whether storage and buffer pool definitions are correct, or whether some adjustments should be performed to improve the performance of I/O activities for HFS files. This report is similar to the part 3 report except that it does not provide data on index activity.

COLLECTING THE HFS PERFORMANCE DATA

It is quite common to see a performance assessment being performed only after we have seen applications experiencing performance problems or when analysts needs to know whether there is enough capacity to support growth or new USS workloads. When it comes to monitoring HFS performance we already know that data gathering for HFS file statistics will be performed in the Monitor III gatherer session. When enabled, the Monitor III gathers SMF record 74 subtype 6. If you want to get information about specific hierarchical file systems, you have to activate the Monitor III gatherer option HFSNAME(ADD(hfsname)). One can dynamically enable or disable this option by using the OS/390 operator commands:

```
F RMF,F III,HFSNAME(ADD(your.hfs.filename))
```

```
F RMF,F III,HFSNAME(DEL(your.hfs.filename))
```

After data gathering for HFS file statistics was performed in the Monitor III gatherer session, we can process collected SMF records either by invoking the RMF HFS postprocessor report or by running the code that I have provided here.

CODE

The code is a four-part stream (called HFSJOB). In the first step (DEL) auxiliary files are deleted, while in the second step (EXT746) the selected HFS-related SMF records are extracted from the SMF weekly/daily dataset and copied to a file that can be used as a base of archived records. It is worth noting that data related to granularity and quality of performance is very important. Too much data will slow the process and increase the resource consumption without providing additional benefit. Intervals for performance analysis should be chosen carefully: seven days of performance data is sufficient to ensure consistency and repeatability. To limit the amount of data collected, one may use the DATE and TIME filtering options the SMF dump program (IFASMFDP) provides. In the next step (SORT746) the extracted records are further filtered. In the fourth step (HFSREXX) the relevant records are formatted by invoking a corresponding REXX EXEC. The EXEC in this step, HFSREXX, is the EXEC that handles the 74.6 records.

SMF record 74 subtype 6 is a repository for HFS global activity, buffer pool statistics, and HFS file system statistics. These subtypes are generated only if HFS dataset names are included. This EXEC may seem to be unnecessary since there is a postprocessor for HFS reports. The main reason for writing this EXEC is this: the HFS postprocessor report is interval based and each report it produces is in fact a collection of several reports each reporting on resource being monitored. This makes each single interval report very dense – one has to be quite skilful in finding out what to look for and where to look. Contrary to this kind of dense reporting, the HFSREXX EXEC is indicator oriented: each performance indicator is separately reported on, thus making it easier to notice the peaks. This is not meant to be a replacement for the postprocessor's report, but rather a supplement to it: once we have spotted a peak value or an exception, we can turn to the postprocessor's report and analyse all the interval reports.

There is a set of three reports produced by this stream, each providing in-depth information on a certain aspect or domain of HFS performance, which allows you to tune your system and make better use of HFS resources. The first one is HFS dataset I/O-related (buffer pool statistics), the second one is on HFS global statistics, and the last one is on HFS file system statistics.

Generally speaking, tuning HFS's I/O is not that different from what a performance analyst normally does to tune any kind of I/O subsystem, and therefore the same general rules apply: avoid unnecessary I/Os, complete most of the I/Os in memory, and complete the real I/Os as fast as possible. It was noted a long time ago that in most applications, I/O activity inversely correlates to performance. Therefore avoiding unnecessary I/Os is primarily an application issue: sometimes it is possible to do something from outside an application but normally we have to understand and change the application's behaviour. What we can really do depends on the application itself. What a performance analyst can recommend nevertheless is this: since USS stores its HFS files on the z/OS side and an I/O for USS may begin in the kernel address space, nonetheless it will use systems services that will also include processing serviced by z/OS. This means that minimizing the amount of data sharing that occurs between USS and z/OS-based applications will yield a performance gain.

In other words, try to isolate the HFS datasets as much as possible, so that USS I/O requests contend only with each other. One way to achieve this I/O separation is to place HFS datasets on DASD that contain infrequently-referenced z/OS datasets. It is also a good idea is to spread high-activity HFS datasets across multiple volumes to keep user HFS datasets separate from system HFS datasets. Remember too that an HFS dataset is a standard OS/390 PDSE structure only in content. The PDSE access method is used only to open the file. After it is opened, it is managed by Unix services. Therefore, the HFS datasets do not benefit from enhanced internal processing for PDSE searches. DFSMS will not use expanded storage for hiperspaces staging.

The second guideline tells us to complete most of the I/Os in memory and at this stage there are essentially three techniques available – TFS, FILECACHE, and HFS global buffering.

Temporary File System (TFS) is an in-storage-only file system (similar to VIO) and can be used for read/write files. The main benefit of using a TFS is a dramatic improvement in file I/O performance since the
I/O is as fast as a memory-to-memory-only access, and it does not incur the overhead of the HFS global buffers. The main issue to consider is that a TFS mounted file system is not backed up by physical disks. So if the system crashes, all data in TFS will be lost. It is for this reason that TFS is normally useful for temporary data only. A tuning tip: according to USS manuals, the storage assigned to TFS is accounted to the OMVS address space, so if there is a need to intensively use TFS one may like to consider the option of setting up a ‘colony address space’ so as to isolate TFS from the kernel. This would prevent virtual storage constraint in the kernel address space caused by the TFS. This means that one has to find the balance between memory one is willing to devote to TFS for improved performance and dedication of storage to TFS.

Filecache is a command that allows one to cache commonly-used read-only files in a data space belonging to the OMVS kernel address space. The amount of storage occupied by the filecache is equal to the sum of the size of each file – thus care should be taken that large files do not put a strain on processor storage (in a storage-constrained environment this could cause paging). Filecache improves system and end user read response times since just a memory-to-memory copy is done and some 33% to 50% reductions in CPU time to access data have being reported. Good candidates for file cache include commonly-executed shell scripts, commonly-executed binaries, and commonly-referenced read-only files. Files cached using the filecache command could be read/write files, but consider that if cached data is being modified, that data is deleted from the cache and any further data access will be from disk. The only way you have to refresh the cache is by issuing the filecache -r command. This command will refresh all the cached files.

HFS global buffering is the latest option available and, despite its current limitations, the most interesting one. The underlying idea is to provide a cache for all HFS data and metadata. However, there is very little one can do to influence the system behaviour in utilizing HFS global buffers.

The buffer pool statistics report provides information about activities and storage usage within your z/OS Unix environment. This data can be used to analyse whether definitions are correct, or whether some adjustments should be performed to improve the performance of I/O activity for HFS files. The following parameters control the HFS buffer usage:

- *VlRTUAL(max)* specifies the maximum amount of virtual storage (in megabytes) that HFS data and metadata buffers can use. HFS may temporarily exceed the *max* limit to avoid failure of a file read or write request, but the amount of space used is reduced to the *max* specification more or less as soon as possible. As in many other cases, the amount of storage needed depends on the workload and system configuration. If you find your system is using more buffers because of heavy I/O to the HFS, and processor storage contention exists, setting a lower value on the *VIRTUAL* parameter may relieve this situation.
- *FIXED(min)* specifies the minimum amount of virtual storage (in megabytes) that is fixed at HFS initialization and permanently remains fixed even if HFS activity drops to zero. Basically, *FIXED* is used to ensure that storage is there when needed. The specified value of *min* must be less than or equal to *VIRTUAL(max)*. The benefit of *FIXED* is to avoid the overhead of page fixing and unfixing needed for I/O, and thus it minimizes CPU utilization and reduces RSM lock contention.

HFS will continue to temporarily fix additional buffers, as needed, during I/O requests. In addition, HFS will unfix storage and go below *FIXED(min)* if there is a pageable storage shortage in the system. By maintaining a pool of buffers already fixed, the system will shorten the path length of I/O operations, avoiding having to page fix and page free the buffers for every I/O operation out of the HFS global buffer pool. Obviously a fixed buffer uses real storage frames so you have to find the right trade-off to guarantee good HFS performance without degrading your system.

From a performance point of view it is interesting to observe that once assigned to the pools the fixed buffers will not be redistributed: we can have a lot of unused fixed buffers in one pool and none in the others. In order to avoid this, one may need to specify FIXED(0) in parmlib and use the confighfs command after the system start to raise the fixed minimum to the target value. By using this technique, fixed buffers will be allocated to the four pools, depending on workload demands. The choice of 'a right buffer pool' depends on a rather complex algorithm based on the following factors: the amount of virtual storage dedicated to HFS global buffers, the amount of fixed real storage dedicated to HFS global buffers, the distribution of storage to four buffer pools, the I/O pattern (random/sequential), the operation type (read/write), the I/O block size, the file size, and the file type. In brief, when an application needs to acquire an HFS buffer, HFS chooses a buffer based on a preference scheme. Since the purpose of allocating a new buffer is generally to do I/O, which requires that the buffer be page-fixed, the first preference is to use a permanently fixed buffer. (This statement is obvious for reads, but even a deferred write will cause an I/O in the near future.) If no permanently fixed buffers are available, the second preference is to use a pageable buffer that is already backed by real storage if one is available. If all such buffers are in-use, then a new buffer is acquired, and the first time that the buffer is touched, the Real Storage Manager (RSM) will allocate some real storage.

Finally, when we consider the third guideline (to complete the 'real' I/Os as fast as possible) we have to be aware of the fact that a file system is usually a single dataset only from the z/OS side, while in fact the underlying structure of directories and files inside a file system can be quite complex and often includes most of the files of a single application.

It is highly recommended that you review the potential for increased HFS buffers and how the increased virtual storage usage may affect the current system. Installations may be especially susceptible to the default specification if they do their migration testing on a system image with less available central storage configured than the target production system. Production systems will generally have more central storage configured, and/or have higher contention for the central storage. Additional information is available on how to set and measure the HFS buffer definition in *Hierarchial File System Usage Guide* (SG24-5482-00).

Two additional reports that are useful when monitoring HFS dataset activity and performance are HFS global statistics report and HFS file system statistics report.

HFS global statistics report provides overall data about I/O activities of HFS files. Fields in this report include total amount of virtual storage assigned to I/O buffers (virtual used), total amount of permanently fixed storage assigned to I/O buffers (fixed used), file I/O statistics (cache/DASD), and metadata I/O (cache/DASD).

The HFS file system statistics report includes data gathered about I/O activity and the internal structure (index) of the HFS datasets. Some key indicators to observe include: mount point, space (allocated/used/ cached/index), I/O activity (data/ metadata), index activity (read/write/ split/join/create), and cache effectiveness (data/metadata/index). The meaning of these fields can be obtained from the *RMF Report Analysis* (SC33-7991) manual.

HFSJOB

```
//DEL       EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
```

```
//SYSIN     DD *
    DELETE hlq.U746.DATA
    DELETE hlq.SMFCOPY.OUT
    SET MAXCC=Ø
/*
//EXT746   EXEC PGM=IFASMFDP,REGION=5M
//INDA1 DD DSN=your.smf.dataset,DISP=SHR
//OUTDA DD DSN=hlq.SMFCOPY.OUT,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(5Ø,2Ø),RLSE),
//         DCB=(your.smf.dataset)
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
       INDD(INDA1,OPTIONS(DUMP))
       OUTDD(OUTDA,TYPE(74))
/*
//SORT746  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=hlq.SMFCOPY.OUT,DISP=SHR
//SMF746   DD DSN=hlq.U746.DATA,
//         SPACE=(CYL,(15)),UNIT=SYSDA,DISP=(NEW,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=3276Ø)
//TOOLIN   DD *
  SORT FROM(RAWSMF) TO(SMF746) USING(SMF6)
//SMF6CNTL DD *
* Eliminate Header and Trailer records
* Get SMF 74.6 with valid HFS data.
* Sort by date and time
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,74,AND,23,2,BI,EQ,6,
                AND,51,2,BI,GT,Ø,AND,59,2,BI,GT,Ø)
  SORT FIELDS=(11,4,PD,A,7,4,BI,A)
/*
//HFSREXX  EXEC PGM=IKJEFTØ1,REGION=ØM
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.lib
//SMF746   DD DISP=SHR,DSN=hlq.U746.DATA
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
prof nopref
%HFSRMF
/*
```

HFSRMF EXEC

```
/* REXX EXEC to read and format RMF record 74
   subtype 6: HFS Global Activity
            : Buffer pool statistics
            : HFS File system statistics                           */
ADDRESS TSO
```

```
/*-----------------------------------------------------------------*/
/* Part 1: Handle file allocation & dataset existence and          */
/*         print report header and labels                          */
/*-----------------------------------------------------------------*/
  userid=SYSVAR(SYSUID)
  r74gl  =userid||'.rmfglob.rep'
  r74bf  =userid||'.rmfbuff.rep'
  r74hf  =userid||'.rmfhsfs.rep'
    x = MSG('OFF')
  if SYSDSN(r74gl) = 'OK'             /* check report dsn validity */
  Then "DELETE "r74gl" PURGE"
  if SYSDSN(r74bf) = 'OK'             /* check report dsn validity */
  Then "DELETE "r74bf" PURGE"
  if SYSDSN(r74hf) = 'OK'             /* check report dsn validity */
  Then "DELETE "r74hf" PURGE"
  "ALLOC FILE(global) DA("r74gl")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
  "REUSE LRECL(95) RECFM(F B) BLKSIZE(27930)"
 Gl.1  ='HFS Global statistics'
 Gl.2  ='  '
 Gl.3  ='Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
 Gl.4  ='  '
 Gl.5  =left(' ',28,' ')||left('-- Storage limits (MB) --',33),
       ||left('-- File I/O --',15)||left(' -- Metadata I/O --',19)
 Gl.6  =left('RMF interval TOD',21),
       ||left("VIRT(MAX)",12)||left('Used',8),
       ||left("FIXED(MIN)",12)||left('Used',9),
       ||left('cache',9)||left('dasd',8),
       ||left('cache',9)||left('dasd',4)
 Gl.7  =left('-',95,'-')
    "EXECIO * DISKW global (STEM Gl.)"
  "ALLOC FILE(buffer) DA("r74bf")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
  "REUSE LRECL(95) RECFM(F B) BLKSIZE(27930)"
 Bf.1  ='Buffer pool statistics'
 Bf.2  ='  '
 Bf.3  ='Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
 Bf.4  ='  '
 Bf.5  =left(' ',26,' ')||left('Number',9)||left('Buffer',8),
        ||left(' -- Pool size --',24)||left('Data',6),
        ||left(' -- I/O activity --',19)
 Bf.6  =left('RMF interval TOD',21)||left('Pool',6)||left('buff.',9),
        ||left('size',8)||left('pages',8)||left('MB',5),
        ||left('fixed',9)||left('spaces',7),
        ||left('total',7)||left('fixed',6)||left("not fix.",8)
 Bf.7  =left('-',94,'-')
```

```
   "EXECIO * DISKW buffer (STEM Bf.)"
  "ALLOC FILE(hfsfil) DA("r74hf")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
  "REUSE LRECL(235) RECFM(F B) BLKSIZE(27965)"
 Hf.1  ='HFS File system statistics'
 Hf.2  ='  '
 Hf.3  ='Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
 Hf.4  ='  '
 Hf.5  =left(' ',83,' ')||left('---- Allocation (pages) ----',40),
       ||left('------ File I/O ------',29),
       ||left('- Matadata I/O -',24),
       ||left('-- Index read/write --',32),
       ||left('-- Index Events --',24)
 Hf.6  =left('RMF interval TOD',21)||left('File system name',31),
       ||left('Mount TOD',28)||left('System',10),
       ||left('Data',6)||left('Attr.dir',11)||left('Cached',11),
       ||left('seq.',7)||left('random',10)||left('cache',10),
       ||left('dasd',8)||left('cache',10)||left('dasd',10),
       ||left('hit',8)||left('miss',10)||left('hit',8)||left('miss',7),
       ||left('new level',12)||left('split',9)||left('join',4)
 Hf.7  =left('-',231,'-')
 "EXECIO * DISKW hfsfil (STEM Hf.)"
/*-------------------------------------------------------------------*/
/* Part 2: Read  SMF record type 74;                                 */
/*         process selected sections & write out to report file     */
/*-------------------------------------------------------------------*/
 'EXECIO * DISKR SMF746 (STEM x. FINIS'
   numeric digits 20
    do i = 1  to x.0
   smftype  = c2d(SUBSTR(x.i,2,1))              /* SMF record type */
   smfstype = c2d(SUBSTR(x.i,19,2))             /* Record subtype  */
   smfdate  = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5)  /* Unpack SMF date */
   smftime  = smf(c2d(SUBSTR(x.i,3,4)))         /* Decode SMF time */
   sid      = SUBSTR(x.i,11,4)                  /* SID             */
   ssi      = SUBSTR(x.i,15,4)                  /* Subsystem ID.   */
   pps = c2d(SUBSTR(x.i,25,4))       /*  Offset to RMF product sect.*/
   prl = c2d(SUBSTR(x.i,29,2))       /*  Lenght of RMF product sect.*/
   prn = c2d(SUBSTR(x.i,31,2))       /*  Number of RMF product sect.*/
 IF pps  <> 0 AND prn  <> 0 Then do
    pps =pps -3
    prd =     SUBSTR(x.i,pps+2,8)
    sam = c2d(SUBSTR(x.i,pps+24,4))
    mvs =     SUBSTR(x.i,pps+40,8)
    ptn = c2d(SUBSTR(x.i,pps+50,1))
    srl = c2d(SUBSTR(x.i,pps+51,1))
    oil = c2d(SUBSTR(x.i,pps+76,2))
    syn = c2d(SUBSTR(x.i,pps+78,2))
    gie = st(c2x(SUBSTR(x.i,pps+80,8)))
    xnm =     SUBSTR(x.i,pps+88,8)
    snm =     SUBSTR(x.i,pps+96,8)
  END
  IF smfstype = '6'  Then do
/*-- -----------------------------------------------------------*/
/* Part 1:  Read SMF record type 74.6 (HFS record)              */
/*--------------------------------------------------------------*/
   do = c2d(SUBSTR(x.i,33,4))      /*  Offset to HFS global data sec.*/
   dl = c2d(SUBSTR(x.i,37,2))      /*  Length of HFS global data sec.*/
   dn = c2d(SUBSTR(x.i,39,2))      /*  Number of HFS global data sec.*/
   bo = c2d(SUBSTR(x.i,41,4))      /*       Offset to HFS buffer sec.*/
   bl = c2d(SUBSTR(x.i,45,2))      /*       Length of HFS buffer sec.*/
   bn = c2d(SUBSTR(x.i,47,2))      /*       Number of HFS buffer sec.*/
   fo = c2d(SUBSTR(x.i,49,4))      /*  Offset to HFS file system sec.*/
   fl = c2d(SUBSTR(x.i,53,2))      /*  Length of HFS file system sec.*/
   fn = c2d(SUBSTR(x.i,55,2))      /*  Number of HFS file system sec.*/
/*---------------- ---------------------------------------------*/
/* Part 2.1: Read SMF record type 74.6 (HFS Section)            */
/*--------------------------------------------------------------*/
 IF do  <> Ø AND dn  <> Ø Then do
  do =do -3
  gmxv = c2d(SUBSTR(x.i,do,4))           /*      VIRTUAL(MAX) -   MB)*/
  gusv = c2d(SUBSTR(x.i,do+4,4)) /* pages of VIR assigned to IO buff.*/
  gmnf = c2d(SUBSTR(x.i,do+8,4))         /*         FIXED(MIN) -  MB */
  gusf = c2d(SUBSTR(x.i,do+12,4))        /*   permanently fixed pages*/
  gmc  = flt(c2x(SUBSTR(x.i,do+16,8)))   /*          metadata - cache*/
  gmnc = flt(c2x(SUBSTR(x.i,do+24,8)))   /*           metadata - dasd*/
  g1c  = flt(c2x(SUBSTR(x.i,do+32,8)))   /*        first page - cache*/
  g1nc = flt(c2x(SUBSTR(x.i,do+4Ø,8)))   /*         first page - dasd*/
  glrc = c2d(SUBSTR(x.i,do+48,4))     /*Ret. code DisplayBufferLimits*/
  glrs = c2d(SUBSTR(x.i,do+52,4))     /*Reas.code DisplayBufferLimits*/
  gsrc = c2d(SUBSTR(x.i,do+56,4))     /*Ret. code DisplayGlobalStats */
  gsrs = c2d(SUBSTR(x.i,do+6Ø,4))     /*Reas.code DisplayGlobalStats */
  gsfl =     SUBSTR(x.i,do+64,1)         /* Global data status flags*/
    SELECT
     when gsfl ='1ØØØØØØØ'b then ffl='Kernel not ready    '
     when gsfl ='Ø1ØØØØØØ'b then ffl='No buffer limit data'
     when gsfl ='ØØ1ØØØØØ'b then ffl='No global data      '
     when gsfl ='ØØØ1ØØØØ'b then ffl='Partial global data '
     otherwise  ffl='Reserved'
    END
  mbv  = (gusv * 4) / 1Ø24
  mbuse= trunc(mbv,2)
 END
  glb = right(Date('N',smfdate,'J'),11)  left(smftime,8),
        right(gmxv,8)   right(mbuse,8)   right(gmnf,8),
        right(gusf,8)   right(g1c,8)     right(g1nc,8),
        right(gmc,8)    right(gmnc,8)
      PUSH glb
   "EXECIO 1 DISKW global"
```

```
/*--------------------------------------------------------------------*/
/* Part 2.2: Read SMF record type 74.6 (Buffer section)               */
/*--------------------------------------------------------------------*/
 IF bo  <> Ø AND bn  <> Ø Then do
    bo =bo -3
    do k = Ø to bn  -1
       num = k + 1                         /*          pool number*/
       incr  = (bo + (k*bl))
       gsb.k = c2d(SUBSTR(x.i,incr,2))     /*size of buffers in bp*/
       gnds.k= c2d(SUBSTR(x.i,incr+2,2))   /*   data spaces for bp*/
       gsbp.k= c2d(SUBSTR(x.i,incr+4,4))   /*    poll size - pages*/
       gsbf.k= c2d(SUBSTR(x.i,incr+8,4))   /*  fixed buffers in bp*/
       gbf.k = flt(c2x(SUBSTR(x.i,incr+16,8)))    /* buffer was fixed*/
       gbnf.k= flt(c2x(SUBSTR(x.i,incr+24,8)))    /* buffer not fixed*/
       nb.k= gsbp.k /gsb.k                        /*   no. of buffers*/
       mb.k= (gsbp.k * 4) / 1Ø24                  /*   pool size - MB*/
       mm.k=trunc(mb.k,2)
       tot.k=gbf.k+gbnf.k                         /*  total I/O count*/
   SELECT
    when k  = Ø then
    inter = right(Date('N',smfdate,'J'),11)||left(' ',1,' '),
            ||left(smftime,8)
    otherwise
    inter =left(' ',2Ø,' ')
   END
  buf.k= inter  right(num,4) right(nb.k,5) right(gsb.k,8),
         right(gsbp.k,8)     right(mm.k,6) right(gsbf.k,5),
         right(gnds.k,8)     right(tot.k,6),
         right(gbf.k,6)      right(gbnf.k,6)
      PUSH buf.k
   "EXECIO 1 DISKW buffer"
    end
   line = left(' ',3,' ')
      PUSH line
   "EXECIO 1 DISKW buffer"
 END
/*-------------------------------------------------------------------*/
/* Part 2.3: Read SMF record type 74.6 (HFS file section)            */
/*-------------------------------------------------------------------*/
 IF fo  <> Ø AND fn  <> Ø Then do
  fo =fo -3
   do j = Ø to fn  -1
    inc  = (fo + (j*fl))
    fsnl.j= c2d(SUBSTR(x.i,inc+44,1))
    fsnm.j=     SUBSTR(x.i,inc,fsnl.j)     /*        File system name*/
    fsfØ.j=     SUBSTR(x.i,inc+45,1)       /*       File Status flags*/
     SELECT
      when fsfØ.j ='1ØØØØØØØ'b then fflag='No HFS file system stat.'
      when fsfØ.j ='Ø1ØØØØØØ'b then fflag='Mount time changed'
      when fsfØ.j ='ØØ1ØØØØØ'b then fflag='File system now mounted'
     otherwise  fflag='Reserved'
     END
    fctm.j= st(c2x(SUBSTR(x.i,inc+48,8)))  /* RMF TOD timestamp STCK */
    fmtm.j= st(c2x(SUBSTR(x.i,inc+56,8)))  /*   Mount timestamp STCK */
    fsf.j = c2d(SUBSTR(x.i,inc+64,4))      /*    Size of file system */
    fpf.j = c2d(SUBSTR(x.i,inc+68,4))      /*   Number of data pages */
    fpd.j = c2d(SUBSTR(x.i,inc+72,4))      /*   Attr.directory pages */
    fpc.j = c2d(SUBSTR(x.i,inc+76,4))      /*Data buffer pages cached*/
    fsfi.j= flt(c2x(SUBSTR(x.i,inc+80,8)))   /*       Sequential I/O */
    frfi.j= flt(c2x(SUBSTR(x.i,inc+88,8)))   /*           Random I/O */
    fmc.j = flt(c2x(SUBSTR(x.i,inc+96,8)))   /*  Metadata from cache */
    fmnc.j= flt(c2x(SUBSTR(x.i,inc+104,8)))  /*   Metadata from DASD */
    f1c.j = flt(c2x(SUBSTR(x.i,inc+112,8)))  /*First page from cache */
    f1nc.j= flt(c2x(SUBSTR(x.i,inc+120,8)))  /* First page from DASD */
    fint.j= flt(c2x(SUBSTR(x.i,inc+128,8)))  /*      Index new tops */
    fis.j = flt(c2x(SUBSTR(x.i,inc+136,8)))  /*         Index splits */
    fij.j = flt(c2x(SUBSTR(x.i,inc+144,8)))  /*          Index joins */
    firh.j= flt(c2x(SUBSTR(x.i,inc+152,8)))  /* Index page read hits */
    firm.j= flt(c2x(SUBSTR(x.i,inc+160,8)))  /*  Index page read miss*/
    fiwh.j= flt(c2x(SUBSTR(x.i,inc+168,8)))  /* Index page write hits*/
    fiwm.j= flt(c2x(SUBSTR(x.i,inc+176,8)))  /* Index page write miss*/
    fsrc.j= c2d(SUBSTR(x.i,inc+184,4))       /* Ret.code Disp.FSStats*/
    fsrs.j= c2d(SUBSTR(x.i,inc+184,4))       /* Reas.codeDisp.FSStats*/
   SELECT
    when j  = 0 then
    inter = right(Date('N',smfdate,'J'),11)||left(' ',1,' '),
            ||left(smftime,8)
    otherwise
    inter =left(' ',20,' ')
   END
  fil.j= inter  left(fsnm.j,30) right(fmtm.j,25),
         right(fsf.j,8)  right(fpf.j,8)  right(fpd.j,8),
         right(fpc.j,8)  right(fsfi.j,8) right(frfi.j,8),
         right(f1c.j,8)  right(f1nc.j,8) right(fmc.j,8),
         right(fmnc.j,8) right(firh.j,8) right(firm.j,8),
         right(fiwh.j,8) right(fiwm.j,8) right(fint.j,8),
         right(fis.j,8)  right(fij.j,8)
      PUSH fil.j
   "EXECIO 1 DISKW hfsfil"
     end
   line = left(' ',3,' ')
      PUSH line
   "EXECIO 1 DISKW hfsfil"
 END
     END
   end
    "EXECIO 0 DISKW global(FINIS "
    "EXECIO 0 DISKW buffer(FINIS "
    "EXECIO 0 DISKW hfsfil(FINIS "
 say
 say 'HFS Global statistics report dsn  . . . . . . . .:'r74gl
 say 'Buffer pool statistics report dsn . . . . . . . .:'r74bf
 say 'HFS File system statistics report dsn . . . . . .:'r74hf
 say
   "free FILE(SMF746 global buffer hfsfil)"
 exit
SMF: procedure
/* REXX - convert a SMF time  */
arg time
    time1    = time % 100
    hh       = time1 % 3600
    hh       = RIGHT("0"||hh,2)
    mm       = (time1 % 60) - (hh * 60)
    mm       = RIGHT("0"||mm,2)
    ss       = time1 - (hh * 3600) - (mm * 60)
    ss       = RIGHT("0"||ss,2)
    otime  = hh||":"||mm||":"||ss             /* Compose SMF time*/
    return otime
FLT: procedure
/* This REXX exec computes a floating-point's value */
arg  float
float = X2C(float)     /* convert float to a hexadecimal string */
float_size = LENGTH(float)                    /* size in Bytes  */
Byte_0 = SUBSTR(float, 1, 1)
 select
   when BITAND(Byte_0, '80'x) == '00'x then sign = '+'
   when BITAND(Byte_0, '80'x) == '80'x then sign = '-'
 end
exponent = C2D(BITAND(Byte_0, '7F'x)) - 64
fraction = 0
power    = -1
do i = 2 to float_size
   if i = 9 then iterate              /* skip bits 64-71  */
   Next_Byte   = C2D(SUBSTR(float, i, 1))
   left_Digit  = Next_Byte %  16
   fraction    = fraction + left_Digit * 16**power
   right_Digit = Next_Byte // 16
   power       = power - 1
   fraction    = fraction + right_Digit * 16**power
   power       = power - 1
end
interpret 'value =' sign ( fraction * 16 ** exponent )
val1=trunc(value,0)
return val1
ST: Procedure
/* STCK timestamp format converted
   The BLSUXTOD proc is described in "z/OS
   V1R3 MVS IPCS Customization"                 */
arg todtime
If todtime    <> '00000000000000000000' Then
```

```
  Do
    TOD_Value = X2C(todtime)
    Returned_Date = '-------------------------'
    address LINKPGM "BLSUXTOD TOD_Value Returned_Date"
  End
Else
  Returned_Date = ''
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Implementation of the IBM Flashcopy/Snapshot functions in a z/OS environment

KEY IDEAS

This article provides a business solution to the demands of providing regular offsite DASD volume back-ups in a timely fashion. IBM provides copy service functions such as Flashcopy for the IBM Enterprise Storage Server and Snapshot for the IBM RAMAC Virtual Array. However, these functions alone do not provide a complete solution.

This article uses code written to meet this business need. This code automates the back-up process by generating the JCL upon execution; it also provides key documentation for recovery, which includes information on when the dump was taken, the actual tapes used to dump, and also all the recovery JCL required at the recovery site. Finally, the code provides a level of optimization to reduce tape media and tape handling costs, which equated to a 50% reduction for the Hursley MVS volumes.

CLAIMS

This article supports the following claims:

1 It automates the back-up and recovery process for DASD volumes.

2 Following a one-time set up, the procedure is repeatable, eg:

```
//DRMVSH  JOB MSGCLASS=H,REGION=6M,CLASS=2,USER=CPSSDMØ
//*********************************************************************
//**  This job will dump DASD volumes and datasets identified by the  **
//**  IBM Hursley IT Department as necessary in supporting Disaster   **
//**  Recovery processes.                                             **
//**  Author:  Martin Hitchman, Senior IT Specialist, IBM Hursley     **
//*********************************************************************
//**  Changes:                                                        **
//*********************************************************************
//** Code changing variables                                          **
//*********************************************************************
// SET PREPARE=N        << Y or N
// SET CYCLE=4          << Ø,1-4 = defined cycle number range
//*********************************************************************
//** Code fairly static variables                                     **
//*********************************************************************
// SET OVERRIDE=N       << ignore check 8 target volser label check
// SET TAPEGB=59        << Capacity of your tapes including compression
// SET VITAL=CBP        << VBP or BRP
// SET SERVICE=MQ       << service group
// SET SYSTEM=MVSH      << system identifier
// SET METHOD=FLASH     << SNAPSHOT or FLASH
// SET DRIVES=ROBOT7    << Manual, Robot7, Robot9
// SET CODE=££REXX££    << where is the local code stored ?
// SET DSN=CPSSDM.HURSLEY.ITDEPT.CODE
//*********************************************************************
//** Generate JCL dynamically                                         **
//*********************************************************************
//BUILDJCL EXEC PGM=IKJEFTØ1,DYNAMNBR=99,
// PARM='%&CODE &PREPARE &CYCLE &TAPEGB &VITAL &SERVICE &SYSTEM        X
//             &METHOD &DRIVES &OVERRIDE'
//SYSEXEC  DD DISP=SHR,DSN=&DSN
//SYSIN    DD DUMMY
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INADDR  DD *
             63ØØ 6317 24
             638Ø 6397 24
//OUTADDR  DD *
             6318 632F 24
             6398 63AF 24
//DUMMY    DD *
             DRDAT1 DRDAT2
//VOLSERS  DD *
//DSNAMES  DD *
//TAPESØ   DD *
             VBTØ21 VBTØ25 5
//TAPES1   DD *
```

```
              VBØØ73 VBØØ77 5
//TAPES2   DD *
              VBØØ78 VBØØ82 5
//TAPES3   DD *
              VBØØ83 VBØØ87 5
//TAPES4   DD *
              VBØØ88 VBØØ92 5
//HTAPESØ   DD *
              VBTØ26 VBTØ26 1
//HTAPES1   DD *
              VBØ6Ø5 VBØ6Ø5 1
//HTAPES2   DD *
              VBØ6Ø6 VBØ6Ø6 1
//HTAPES3   DD *
              VBØ6Ø7 VBØ6Ø7 1
//HTAPES4   DD *
              VBØ6Ø8 VBØ6Ø8 1
//SYSTSIN  DD DUMMY
//*
```

3 It reduces the time to implement a disaster recovery back-up solution and also ensures speedy recovery by automatically providing all back-up and recovery JCL.

4 It reduces the number of tapes to purchase by providing a lower level of tape utilization.

5 It reduces your costs with an offsite service provider by reducing the number of tapes used.

6 It is a scalable solution. Your changing business storage requirements are easily accommodated as the initial JCL file contains only key data. The program generates all the other required JCL.

7 Error checking routines ensure that you have the correct number of tapes, you don't overwrite live DASD volumes (based on volume labels), and the correct number and type of DASD volumes are present.

EMBODIMENTS

In a z/OS MVS environment the following steps are required to dump the DASD volumes to tape to enable them to be sent offsite for a disaster recovery.

1 Identify source volumes to dump.

2 Identify target volumes for Flashcopy/Snapshot.

3 Initiate Flashcopy/Snapshot commands to provide a point-in-time copy of source volumes.

4 Change the DASD volume labels because the replication process duplicates the source volume label and MVS hosts allow only unique volume labels to be on-line.

5 Put target volumes on-line to host.

6 Dump volumes to tape.

This isn't really the complete picture for a workable, repeatable procedure because every step above can raise questions and issues. So my article includes the following.

The user performs a one time set-up by coding the following in the sample JCL file:

1 Source volumes to dump by device address range.

2 Target volumes for Flashcopy/Snapshot by device address range.

3 Tape label ranges to be used per cycle.

This static information is used to derive all the back-up and recovery JCL dynamically and prepare reports for documentation purposes. The JCL that is automatically built is split into a number of stages to provide a clear procedure to follow.

USER SCENARIO

A user has an IBM Enterprise Storage Server for his Sysplex and is using DASD control unit numbers 0–7.

The control units each contain 80 3390 Model 9 DASD volumes and the customer is using the first half of the control unit for the source volumes and reserving the second half for the target volumes. The customer uses the volume convention name of FL*xxxx* for the target volumes and this ensures that the code to validate replication is not targeting live volumes.

The customer executes the following steps:

1 A batch job to calculate DASD occupancy.

2 A batch job to prepare the customized JCL.

3 A batch job to validate and prepare all back-up and recovery JCL together with all documentation required during the restore.

4 A suite of customized batch jobs to automate and complete the back-up process from start to finish

BUSINESS VALUE

This article provides a complete application to perform regular point-in-time back-up copies of DASD volumes in a z/OS MVS environment.

The code is very easy to use and set up, and can quickly be incorporated into your own business environment.

The business reduces its overheads in people resources because to perform these tasks without automation is time consuming and because of the repetition would inevitably result in coding mistakes.

The business will also make savings in its tape management systems. Not only will fewer removable media cartridges be required but also any expenses related to storing the media off-site will be reduced because this is typically charged by the cartridge.

CODING JCL

INADDR – list of MVS DASD addresses.

Format: <start address> <end address> <count>

OUTADDR – list of MVS DASD addresses.

Format: <start address> <end address> <count>

DUMMY – list of MVS DASD volumes that are not dumped but an empty volser will be created at the recovery site.

VOLSERS – list of specific DASD volumes to dump to tape.

DSNAMES – list of specific MVS datasets to dump to tape.

TAPES*x* – list of tape volsers to use for each cycle 0–4.

HTAPES*x* – list of tape volsers to use for each cycle 0–4.

DYNAMICALLY BUILD JCL

1 Submit:

```
"CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(££INFO££)"
```

2 Edit "cpssdm.hursley.itdept.cbp.cntl(££VARS££)".

Increment the CYCLE= for each of the members (note: only four cycles 1, 2, 3, and 4 are defined). Change PREPARE= Y.

3 EXexute 'cpssdm.hursley.itdept.cbp.cntl(££sub££)'.

4 Submit: "CPSSDM.HURSLEY.VARYOFF.SOURCE".

5 Submit: "CPSSDM.HURSLEY.VARYOFF.TARGET".

6 Submit: "CPSSDM.HURSLEY.VARYON.SOURCE".

7 Submit: "CPSSDM.HURSLEY.VARYON.TARGET".

8 Submit: "CPSSDM.CBP.**.STAGE27".

9 Edit "cpssdm.hursley.itdept.cbp.cntl(££VARS££)". Change PREPARE= N.

10 EXexute 'cpssdm.hursley.itdept.cbp.cntl(££sub££)'.

EXECUTE THE JCL

The process sequence is as follows:

1 "CPSSDM.CBP.**.STAGE*xx*"

2 Vary off target devices only for Flash jobs.

3 Replication.

4 DSF Clip target volsers.

5 Vary on-line target volumes: Submit "CPSSDM.HURSLEY.VARYON.TARGET".

6 List VTOC. Submit and once complete release any HELD DRVTOC jobs.

7 Dump volsers.

8 Dump header information.

9 Send reports to storage taskid.

10 F HSM,RELEASE ALL.

11 Vary off target devices – wait. Only for FLASH jobs.

12 Withdraw replication – wait.

13 DSF INIT target volsers.

14 Vary on-line target volumes, or submit "CPSSDM.HURSLEY.VARYON.TARGET".

Produce a tape report for audit purposes. Submit one of the 'CPSSDM.CBP.**.STAGE25' stages.

Eject the tapes from the ATL:

```
EX "CPSSDM.HURSLEY.AUDIT.EJECTS"
```

CNTL_INFO ($$INFO$$)

```
//DCOLLECT  JOB MSGCLASS=H,NOTIFY=CPSSDM2,REGION=6M
//MYLIBS JCLLIB ORDER=(SYS1.SACBCNTL)
//*******************************************************************
//* cleanup first
//*******************************************************************
//DELETE  EXEC PGM=IEFBR14
//DD1      DD   DSN=CPSSDM.HURSLEY.DCOLLECT.OUTPUT,
//         SPACE=(TRK,(1,1)),DISP=(MOD,DELETE,DELETE)
```

```
//DD2      DD   DSN=CPSSDM.HURSLEY.DCOLLECT.REPORT,
//         SPACE=(TRK,(1,1)),DISP=(MOD,DELETE,DELETE)
//DD3      DD   DSN=CPSSDM.HURSLEY.AUDIT.EJECTS,
//         SPACE=(TRK,(1,1)),DISP=(MOD,DELETE,DELETE)
//DD4      DD   DSN=CPSSDM.HURSLEY.VARYON.SOURCE,
//         SPACE=(TRK,(1,1)),DISP=(MOD,DELETE,DELETE)
//DD5      DD   DSN=CPSSDM.HURSLEY.VARYOFF.SOURCE,
//         SPACE=(TRK,(1,1)),DISP=(MOD,DELETE,DELETE)
//DD6      DD   DSN=CPSSDM.HURSLEY.VARYON.TARGET,
//         SPACE=(TRK,(1,1)),DISP=(MOD,DELETE,DELETE)
//DD7      DD   DSN=CPSSDM.HURSLEY.VARYOFF.TARGET,
//         SPACE=(TRK,(1,1)),DISP=(MOD,DELETE,DELETE)
//*
//*********************************************************************
//* dcollect volume information
//* exclude vse volumes +
//*********************************************************************
//STEP1  EXEC  PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//DCOUT     DD  DSN=CPSSDM.HURSLEY.DCOLLECT.OUTPUT,DISP=(NEW,CATLG),
//         SPACE=(TRK,(100,10)),UNIT=3390,
//         DCB=(DSORG=PS,LRECL=264,RECFM=VB,BLKSIZE=0)
//SYSIN     DD  *
 DCOLLECT OUTFILE(DCOUT) VOLUMES(*) NODATAINFO  -
  EXCLUDEVOLUMES(SLSP04,SL*,VSE*,DOSRES,SYWRK1,IAL*,DMTA*,#*,H9*)
//***********************************************************************
//* Generate volume report to include %free space
//***********************************************************************
//REPORT  EXEC  ACBJBAOB,PLIB1=SYS1.DGTPLIB,TABL2=CPSSDM.HURSLEY.ISPTABLE
//DCOLIN   DD  DSN=CPSSDM.HURSLEY.DCOLLECT.OUTPUT,DISP=SHR
//ISPFILE DD  DSN=CPSSDM.HURSLEY.DCOLLECT.REPORT,DISP=(NEW,CATLG),
//         SPACE=(TRK,(100,10)),UNIT=3390,
//         DCB=(DSORG=PS,LRECL=133,RECFM=FBA,BLKSIZE=0)
//SYSTSIN  DD *
     PROFILE PREFIX(CPSSDM2)
     ISPSTART CMD(ACBQBAR6) +
     BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
/*
//SYSIN DD  *
     ADDRESS
     VOLSER
     ALLOCSPC
     DEVTYPE
/*
```

$$SUB££

```
SUBMIT 'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(mvsh)'
SUBMIT 'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVSZ)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVS16)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVS17)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVS18)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVS19)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVS2)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVS4K)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(MVS91)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXC)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXD)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXE)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXF)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXG)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXJ)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXK)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXL)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXN)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXP)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXS)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXW)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEXY)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEX1)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEX2)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEX3)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PLEX5)'
SUBMIT  'CPSSDM.HURSLEY.ITDEPT.CBP.CNTL(PYE)'
```

$$VARS$$

```
// SET PREPARE=N         << Y OR N
// SET CYCLE=4           << Ø,1-4 = defined cycle number range
//*********************************************************************
//** Code fairly static variables                                   **
//*********************************************************************
// SET OVERRIDE=N        << ignore check 8 target volser label check
// SET TAPEGB=56         << Capacity of tapes + compression 2.8)
// SET VITAL=CBP         << CBP
// SET METHOD=FLASH      << SNAPSHOT or FLASH
// SET DRIVES=ROBOT7     << Manual, Robot7, Robot9
// SET CODE=££REXX££     << where is the local code stored ?
//  SET DSN=CPSSDM.HURSLEY.ITDEPT.CODE
//*********************************************************************
//**  This job will dump DASD volumes and datasets identified by the  **
//**  IBM Hursley IT Department as necessary in supporting Disaster   **
//**  Recovery processes.                                             **
//**                                                                  **
//**  Documentation on execution and maintanence can be found in the  **
//**  following Lotus Notes Database:                                 **
//**   http://vØ6dblØ2.hursley.ibm.com/i_dir/itmqseries.nsf           **
//**                                                                  **
```

```
//**  Author:  Martin Hitchman, Senior IT Specialist, IBM Hursley     **
//*********************************************************************
//**  Changes:                                                       **
//*********************************************************************
```

PLEX1

```
//DRPLEX1   JOB MSGCLASS=H,REGION=6M,CLASS=1,USER=CPSSDMØ
//       JCLLIB ORDER=(CPSSDM.HURSLEY.ITDEPT.CBP.CNTL)
//       INCLUDE MEMBER=££VARS££
//*********************************************************************
//** system specific information                                     **
//*********************************************************************
// SET SERVICE=MQ       << service group
// SET SYSTEM=PLEX1     << system identifier
//*********************************************************************
//** Generate JCL dynamically                                        **
//*********************************************************************
//BUILDJCL EXEC PGM=IKJEFTØ1,DYNAMNBR=99,
// PARM='%&CODE &PREPARE &CYCLE &TAPEGB &VITAL &SERVICE &SYSTEM        X
//              &METHOD &DRIVES &OVERRIDE'
//SYSEXEC  DD DISP=SHR,DSN=&DSN
//SYSIN    DD DUMMY
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INADDR  DD *
              618Ø 6197 24
              62ØØ 6217 24
              628Ø 62B3 52
//OUTADDR  DD *
              6198 61AF 24
              6218 622F 24
              62B4 62E7 52
//DUMMY    DD *
              DRDAT1 DRDAT2
//VOLSERS  DD *
//DSNAMES  DD *
//TAPESØ   DD *
              VBTØ27 VBTØ44 18
//TAPES1   DD *
              VBØØ93 VBØ11Ø 18
//TAPES2   DD *
              VBØ111 VBØ128 18
//TAPES3   DD *
              VBØ129 VBØ146 18
//TAPES4   DD *
              VBØ147 VBØ164 18
//HTAPESØ   DD *
              VBTØ45 VBTØ45 1
```

```
//HTAPES1   DD *
              VBØ6Ø9 VBØ6Ø9 1
//HTAPES2   DD *
              VBØ61Ø VBØ61Ø 1
//HTAPES3   DD *
              VBØ611 VBØ611 1
//HTAPES4   DD *
              VBØ612 VBØ612 1
//SYSTSIN  DD DUMMY
//*
```

Editor's note: the rest of the code for this article, which is quite extensive, is available from the Xephon Web site. It's at www.xephon.com/extras/fsfunction.txt, and contains PLEX2 and $$REXX$$.

Martin Hitchman
Senior IT Specialist
IBM (UK)

MVS news

NEON Systems has announced Shadow Interface for Enterprise Applications, which enables organizations with legacy, terminal-based, enterprise applications to reuse their mainframe resources in new composite applications built around a SOA.

The product enables the deployment of 3270 applications using a service-oriented architecture for distributed environments. The product provides tools for Web-enabled application refacing and service-enabled application remodelling of 3270 applications, allowing for composite applications to be sourced from existing mainframe applications.

For further information contact:
NEON Systems, 14100 Southwest Freeway, Suite 500, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200.
URL: http://neonsys.com/Shadow/si-enterprise_applications.asp.

* * *

IBM has announced Version 2.3 of Tivoli System Automation for z/OS, which helps customers who have a single processor z/OS systems and Parallel Sysplex clusters by providing functions that ease systems management. It offers message management with self-configuration and single action automation policy. Users can navigate through a dependency graph, and fill or overwrite start and stop requests with customer chosen defaults. SA z/OS automates I/O, and processor and system operations. It includes out-of-the-box automation for IMS, CICS, Tivoli Workload Scheduler, DB2, mySAP, and WebSphere.

For further information contact your local IBM representative.

* * *

Redwood Software has announced Version 6.0.1 of Cronacle, its event-driven process automation and job scheduling platform.

The product has an enhanced interface with JES2 and JES3 to enable the further integration of z/OS and OS/390 processes. Also CronacleBeans, Cronacle's Java and J2EE scheduling component, can now run on z/OS systems. Complex dependencies between the mainframe and other systems can be managed from a single interface.

For further information contact:
Redwood Software, 3000 Aerial Center, Suite 115, Morrisville, NC 27560, USA.
Tel: (919) 460 5400.
URL: http://www.redwood.com/products/cronacle.htm.

* * *

Blockade Systems has released ManageID Enterprise Suite for Microsoft Identity Integration Server 2003, Enterprise Edition. This suite encompasses Management Agents for z/OS and OS/390 environments (RACF, ACF2, Top Secret), which integrate with MIIS.

The product provides real-time event detection, which includes the ability to both detect and apply account creations or account deletions, and account attribute or state changes on z/OS security environments.

For further information contact:
Blockade Systems, 2200 Yonge Street, Suite 1300, Toronto, Ontario, Canada, M4S 2C6.
Tel: (416) 482 8400.
URL: http://www.blockade.com/products/miis.html.

* * *