213

MVS update

June 2004

In this issue

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.

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.

Printed in England.

Checking a steplib concatenation for authorized datasets

The following ISPF edit macro can be used to check a steplib concatenation for authorized datasets. The EXEC will highlight any datasets that do not exist, are migrated, or are not in the APF list. The EXEC assumes that the APF list is in the dynamic format.

```
/* Rexx */
/* an edit macro to check that a STEPLIB concatenation  */
/* is APF authorized                                    */
address isredit
"macro"
"reset special"
true = 1; false = 0
"find '//STEPLIB ' first 1"
if rc ¬= 0 then
   do
      say 'no //STEPLIB found'
      exit
   end
call build_apflist
finished = false
"(lnum) = LINENUM .zcsr"
"(ldata) = LINE "lnum
"(max) = LINENUM .zlast"
do until (finished)
   call process_jcl_card
   lnum = lnum + 1
   if lnum > max then
      exit
   "(ldata) = LINE "lnum
   finished = end_of_steplib( )
end
infomsg = '=== end of APFCHECK ==='
"line_before "lnum "= INFOLINE (infomsg)"
exit
process_jcl_card :
   if substr(ldata,1,3) = '//*' then
      return
   startpos = pos('DSN=',ldata)
   if startpos = 0 then
      return
   startpos = startpos + 4
```

```
   endpos = pos(',',ldata,startpos)
   if endpos = 0 then
      endpos = pos(' ',ldata,startpos)
   dsname = substr(ldata,startpos,endpos-startpos)
   if pos('&',dsname) ¬= 0 then  /* symbols in name ? */
      do
     infomsg = copies(' ',startpos-1) || dsname' === can''t resolve ==='
         "line_after "lnum "= INFOLINE (infomsg)"
         return
      end
   x = listdsi("'"dsname"' smsinfo norecall")
   if  sysreason = 5 then  /* doesn't exist ? */
      do
     infomsg = copies(' ',startpos-1) || dsname' === doesn''t exist ==='
         "line_after "lnum "= INFOLINE (infomsg)"
         return
      end
   if  sysreason = 9 then  /* migrated ? */
      do
       infomsg = copies(' ',startpos-1) || dsname' === is migrated ==='
         "line_after "lnum "= INFOLINE (infomsg)"
         return
      end
   do j = 1 to apflist.0      /* search the list of APF datasets */
      if dsname ¬= substr(apflist.j,8) then
         iterate
      if sysvolume = substr(apflist.j,1,6) then /* specific volume */
         return
      if sysstorclass ¬= ' ' & substr(apflist.j,1,3) = 'SMS' then
         return
   end
   infomsg = copies(' ',startpos-1) || dsname' === not authorised ==='
   "line_after "lnum "= INFOLINE (infomsg)"
return
end_of_steplib : procedure expose true false ldata
   if substr(ldata,1,2) ¬= '//' then
      return (true)
   if strip(ldata) = '//*' | strip(ldata) = '//' then
      return (true)
   if substr(ldata,1,3) = '//*' & substr(ldata,4,1) ¬= ' ' then
      return (true)
   if substr(ldata,3,1) ¬= '*' & substr(ldata,3,1) ¬= ' ' then
      return (true)
return (false)
build_apflist :
   /* this assumes a dynamic APF list */
   numeric digits 20
   @cvt  = c2d(storage(10,4))
   @ecvt = c2d(storage(d2x(@cvt+140),4))        /* cvtecvt  */
   @csvt = c2d(storage(d2x(@ecvt+228),4))       /* ecvtcsvt */
```

```
   @apfa = c2d(storage(d2x(@csvt+12),4))
   apht = c2d(storage(d2x(@apfa+8),4))
   i = 0
   do while (apht ¬= 0)
      dsn = storage(d2x(apht+24),44)
      if storage(d2x(apht+4),1) = '80'x then
         volser = 'SMS   '
      else
         volser = storage(d2x(apht+68),6)
      if substr(dsn,1,1) ¬= '00'x then
         do
            i = i + 1
            apflist.i = volser','strip(dsn)
         end
      apht = c2d(storage(d2x(apht+8),4))
   end
   apflist.0 = i
return
```


Quick HFS free space report

Every now and then it happens that a storage administrator or system programmer encounters an HFS file system flagging an out-of-space condition. This can happen when installing a new application, a product that is not part of the standard ServerPac order, or simply because of an application's high write activity. By taking a proactive approach to planning and monitoring HFS free space, one can avoid the situation where the file/volume becomes so full that there is a risk of not writing out data from the buffer when it is time to unmount the HFS dataset (in order to add candidate volumes). Thus, when installing a new application or a product that installs into the HFS, consider doing the following:

- Creating new directories where the files associated with the new application/product will be installed.
- If possible, creating a new HFS dataset and mounting it for the new directory. After installation of the product/application, all the files will reside in the new HFS dataset.

- Keeping new applications/products in different HFS datasets, which offers better file system management while maintaining a more stable root file system. This also ensures easier maintenance when applying service.

Before proceeding any further, it might be helpful to remember that in order to update an HFS file, DFSMS 1.5 uses a shadow write technique that maintains duplicate pages in the dataset until the update is completed. This technique improves the integrity of the data, but on the other hand it also requires that there always must be a certain number of free pages within the dataset. In order to shadow write, it needs to make a copy of the attribute page(s) it is updating along with any index pages that need to change because of pointers to different pages. Usually the tree depth is small and it needs only a few pages. HFS keeps a 30-60 byte block in reserve for this purpose in case the file system runs out of user space. Please note that some free space is needed even if only reading data! It is good to know that a new parameter has been added to the PARM keyword on the MOUNT statement to control the number of pages HFS should reserve for Sync processing of the file system metadata. When this parameter is specified on the MOUNT statement, it will override HFS's internal reserved page estimation algorithm. This parameter should be used only if it you find that the internal algorithm is not providing the desired results. The new parameter is SYNCRESERVE(*nn*), where *nn* represents the percentage of the file system space to be reserved for the Sync shadow write mechanism. Valid values for *nn* are between 1 and 50. The trade-off is that less space will be available for user file data in the HFS (see APAR OW43771). How to recover from an out-of-space error during sync on the HFS root file system or /etc directory is described in great detail by info APAR II13537.

As already stated, it is necessary to monitor the utilized space within each dataset regularly, as well as to take preventative action when you find a dataset that is close to exhausting its available space. The storage administrator's attention should be especially focused on high I/O activity datasets, since one may see a performance benefit by allocating particularly active

application files across a number of different HFS datasets as well as predicting and thus preventing out-of-space errors. Unfortunately, there do not seem to be many tools available that help you easily to identify your most active files. There is some activity data recorded in SMF type 92 records that may prove helpful, but it will take some manipulation on your part.

To help alleviate the burden of monitoring HFS free space, a simple yet easy-to-use REXX procedure was written. It uses the USS command df, which displays the amount of free space in the file system. By default the df measures space in units of 512-byte disk sectors. One can specify a particular file system by naming any filename on that file system. If you do not give an argument, df reports on space for all mounted file systems known to the system. The total space reported is the space in the already allocated extents (primary and any already allocated secondary extents) of the HFS dataset that holds this file system. Therefore, the total space may increase as new extents are allocated. An additional option, –S, was used because it provides SMF I/O accounting for mounted files. A detailed description of the df command and the output it produces can be obtained from *z/OS Unix System Services Command Reference* (SA22-7802). The report this procedure produces (HFS free space and I/O activity) can shed some light on your HFS file system status and activity, thus helping you to prevent the out-of-space problems:

```
                                                                  - Dir
I/O blocks -  Total bytes
Filesystem          Allocated   Used   Free Free %  reads writes total
read write   read written
-------------------------------------------------------------------------
--------------------------
OMVS.DB2710.DSNHFS       3600   1051   2549  70.80    70     0    2348
63     0   154550     0
OMVS.DB2710.HFS.DB2TX   10800  10573    227   2.10   144     0    3322
148     0   305247     0
OMVS.DB2710.HFS.DB2EXT  10800   5899   4901  45.37    54     0    2498
74     0   215262     0
OMVS.DB2.HFSWHSE         5760    112   5648  98.05    43     0    1092
181     0   640123     0
OMVS.DTW.SDTWHFS         4680   2233   2447  52.28   174     0    4158
165     0   371080     0
OMVS.WAS.CONFIG.HFS     32400  14942  17458  53.88  4008     0  567710
```

```
5227     Ø  3635399     Ø
OMVS.JV39Ø              81900Ø   3631  78269  95.56  1Ø34    Ø   86470
11Ø4     Ø  2723247     Ø
OMVS.AS39Ø             135ØØØ  39771  95229  7Ø.54   736    Ø   57282
14Ø2     Ø  4192596     Ø
OMVS.ILMSPDM             36ØØ   3578     22   Ø.61     6    Ø      1Ø
5     Ø     437Ø     Ø
OMVS.ILMUC               36ØØ   3578     22   Ø.61     6    Ø      1Ø
5     Ø     437Ø     Ø
OMVS.VAR                 36ØØ   3475    125   3.47    71    2     8Ø7
53     4    95267  9969
OMVS.ETC                1Ø8ØØ  1Ø171    629   5.82  274Ø  131   21Ø13
3168   131  9Ø4Ø493  6727
OMVS.HFS.ROOT           882ØØØ  18ØØ3 863997  97.95 32473  243 1534188
79883   243  63Ø2481 34581
```

Detecting out-of-space conditions was greatly facilitated by applying APAR OW44631, which has provided an early warning capability to let users know when a file system is becoming full. This APAR shipped a new function for HFS called HFS MONITOR.

It provides support to monitor how full an HFS is and will issue operator message IGW023A when the HFS exceeds a user-specified full threshold. The fullness of an HFS is based on the number of pages currently in use versus the currently allocated HFS file system size. The user can specify threshold and increment values via the parmlib member BPXPRM*x* to set default values to be used for all HFS file systems. The values can also be specified on the Mount command to set values for a specific file system. Parameters on the Mount command will override parmlib values. If no values are specified in either place, no threshold checking will be done. Message IGW023A will be automatically removed if the HFS is extended to bring the HFS below its threshold, if files are deleted to bring the HFS below its threshold, or if the HFS is unmounted.

The new parameter syntax is:

```
FSFULL(threshold,increment)
```

where *threshold* means that when the HFS exceeds threshold% full, operator messages will start to be generated (default is 100%). *increment* means that with each increment% increase/decrease in file system fullness beyond the threshold, the message will be updated (default is 5%).

HFSFREE

```
/* REXX ***************************************************************
   Procedure: HFSfree
   Description: Get information on HFS free space and I/O activity
   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||'.hfs.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(150) RECFM(F B)"
/*-------------------------------------------------------------------*/
/* Allocate BPXWUNIX load library:                                   */
/* supply the name of received REXX function load library            */
/*-------------------------------------------------------------------*/
 arg hlq
 if hlq = "" then HLQ = 'uid.REXXFUNC.LOAD'
Address ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"hlq") STACK"
call syscalls 'ON'
/*-------------------------------------------------------------------*/
/* Return USS information                                            */
/*-------------------------------------------------------------------*/
Address SYSCALL
'uname sys.'
/*-------------------------------------------------------------------*/
/* Print headers and labels                                          */
/*-------------------------------------------------------------------*/
sis.1 = left('HFS Quick report - produced on:',32,),
        ||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(' ',112,' ') left('- Dir I/O blocks -',21),
```

```
        left('Total bytes',12)
hf.1  = left('Filesystem',1Ø) left(' ',13)   left('Allocated',13),
        left('Used',8)        left('Free',5) left('Free %',6),
        left('Mounted on',1Ø) left(' ',22,' ') left('reads',7),
        left('writes',7)      left('total',8)  left('read',6),
        left('write',6)       left('read',6)   left('written',7)
hf.2  = left('-',15Ø,'-')
/*-----------------------------------------------------------------*/
/* Get HFS data                                                    */
/*-----------------------------------------------------------------*/
Address SH
  call BPXWUNIX "df -S",,out.
 dfrc = rc
  If dfrc  <>Ø Then Do
    Say "Return Code         =" rc
    Say "OMVS Return Value   =" retval
    Say "OMVS Return Code    =" errno
    Say "OMVS Reason Code    =" errnojr
  End
  j = 1
  Do i = 2 to OUT.Ø                 /* Process each entry returned*/
  parse var out.i  mount . '(' HFS ')' rawdata .
  parse var  rawdata  ava  '/' tot
  fre  =  tot   - ava
  pct = trunc((1-(ava/tot))*1ØØ, 2); i = i + 1;
  read = word(out.i,5);   i =i + 1;
  write= word(out.i,5);   i =i + 1;
  ioblk= word(out.i,6);   i =i + 1;
  reblk= word(out.i,6);   i =i + 1;
  wrblk= word(out.i,6);   i =i + 1;
  byread = word(out.i,7); i =i + 1;
  bywrite= word(out.i,7)
  HFS = left(HFS,25)
rrw.j=left(HFS,25),                 /* Filesystem                */
      right(tot/8,8),               /* Total 4K pages allocated  */
      right(ava/8,8),               /* Available pages (4K)      */
      right(fre/8,8),               /* Free pages (4K)           */
      format(pct,3,2),              /* Percent free              */
      left(mount,32),               /* Mounted on                */
      right(read,7) ,               /* Number of reads           */
      right(write,7) ,              /* Number of writes          */
      right(ioblk,7) ,              /* Number directory I/O block*/
      right(reblk,7) ,              /* Number read I/O blocks    */
      right(wrblk,7) ,              /* Number write I/O blocks   */
      right(byread,7) ,             /* Total number bytes read   */
      right(bywrite,7) ; j = j + 1  /* Total number bytes writte */
    End
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 hf. )"
"EXECIO * DISKW PRC (STEM rrw. )"
 /*-------------------------------------------------------------*/
 /* Close & free allocated report file; then display result     */
 /*-------------------------------------------------------------*/
 "EXECIO Ø DISKW PRC (FINIS "
  "free FILE(PRC)"
   Address ISPEXEC
  "ISPEXEC BROWSE DATASET('"outds"')"
 exit Ø
 /*-------------------------------------------------------------*/
 /* Error exit routine                                          */
 /*-------------------------------------------------------------*/
 ERROR: say 'The following command produced non-zero RC =' RC
        say SOURCELINE(SIGL)
        exit
```

Mile Pekic
Systems Programmer (Serbia)

High resource users – accumulated statistics suite based on SMF records

INTRODUCTION

There are various products to monitor the performance of system components. However, not all of these products analyse the components to establish which ones should then be selected for monitoring. At our site we have encountered this problem.

We have a super tool to monitor on-line and batch processes. For on-line it is quite easy to set up the monitoring because we have only two relevant production CICS to worry about. For both of these CICS systems we automatically trigger measurements twice a day via a simple batch job controlled by our scheduling system.

Batch, on the other hand, is quite different because we must specify which, from several hundred, jobs/steps/programs we want to measure. We could measure them all, but this would produce an overload of information that somebody would have to wade through.

To make our life easier we could buy add-on products from other vendors, but in this case we have developed something ourselves to identify the greatest resource users and therefore the potential candidates for performance tuning.

At a later date we intend to improve the selection process (which jobs/steps/programs should be measured) by providing user interfaces in ISPF. The first step, however, is to establish a method of comparison to be able to select the greatest resource users for subsequent monitoring and measurement tasks.

To satisfy the needs of this first phase and to produce something of use in the interim, I have developed a suite to accumulate, analyse, and report resource usage information per batch jobstep.

EXTRACTING THE DATA

As you know, MVS (OS/390, z/OS) has a background 'tool' called System Management Facilities (known simply as SMF), which collects and records system- and job-related information that you can use in:

- Billing users
- Reporting reliability
- Analysing the configuration
- Scheduling jobs
- Summarizing direct access volume activity
- Evaluating dataset activity
- Profiling system resource use
- Maintaining system security.

The type 30 SMF record provides accounting information, and subtype 4 provides 'step' total information. This will be our source of information for determining which 'steps' are the greatest system resource users.

For further information please see the IBM manual *MVS System Management Facilities (SMF)*, http://publibz.boulder.ibm.com/epubs/pdf/iea2g233.pdf.

The first phase is to extract the relevant information from the SMF records dataset and format it into smaller records for further analysis. The SMF records dataset is huge, with a VBS (variable block spanned) record of length 32,760 bytes (at our site a daily dataset uses about 1,500 cylinders), and we need only a small portion of this.

In our case we select SMF 30 records (SMF30RTY = X'1E') of subtype 04 (SMF30STP = 04), which meet the criterion 'batch job', eg 'JES2' (SMF30WID = 'JES 2'), and write them to an interim dataset with VB records of length 32,756 and space of about 30 cylinders.

We then extract the relevant information for our use from this dataset.

The conversion from VBS to VB reduces the length of the records and also means that this needs to be taken into account when determining the offsets to the various sections that we will need.

The main section (header/self-defining section) has offset fields referring to the following sections. In our case, because the first 3 bytes have been cut off, all the offsets will be shorter by a factor of 3 bytes. Our selection is performed with a sort (SORT01).

FORMATTING

The next stage is to format the records into new records containing the information we will need. We reduce here from VB 32756 to FB 240. Fields of interest are as follows:

- Header/self-defining section:

  - system identification – SMF30SID
  - end date – SMF30DTE
  - end time – SMF30TME.
- Identification section:
  - job name – SMF30JBN
  - job number – SMF30JNM
  - step name – SMF30STM
  - program name – SMF30PGM (PGM=xxxxxxxx)
  - step number – SMF30STN
  - RACF user – SMF30RUD
  - initiator start date – SMF30STD
  - initiator start time – SMF30SIT.
- Processor accounting section:
  - step CPU time – SMF30CPT and SMF30CPS.
- Performance section:
  - total service units – SMF30SRV
  - CPU service units – SMF30CSU
  - SRB service units – SMF30SRB
  - I/O service units – SMF30IO
  - MSO service units – SMF30MSO.
- Completion section:
  - step completion code – SMF30SCC.

ACCUMULATION, ANALYSIS, AND EXCLUSION

The first exclusion is to use a sort to include only records with

CPU usage and *Total Service Units* greater than zero, and return codes of 04 and lower. This then excludes records with insignificant usage and those with abnormal runs. The abnormal runs show that something was likely to be in error and the job must be rerun, allowing corruption of the 'normal' run statistics; therefore we will also exclude these records.

In the next stage, BTCHSDB1, we reformat the data one last time for our record accumulation dataset. We have two lots of JCL: one for daily runs and another for initialization. The daily run JCL produces a mini version of the accumulation dataset, which is then merged with the accumulation dataset in the next sort step. The initializing JCL is constructed to create the accumulation dataset and fill it with the data from several days' worth of SMF records.

A second exclusion is performed to remove all records with a run-time under 1 second.

Now that we have our accumulated records dataset we can perform statistical analysis.

The analysis is pretty simple and can be thought of as a bit of basic number crunching followed by sorting to show the greater resource users.

The routine BTCHSDB2 is used to do this number crunching. It updates the accumulation dataset and produces a statistical dataset. The statistical dataset can be used as input for ISPF selection routines (something we have planned) and for input for a daily report.

The method of statistical selection and preparation is as follows.

We pass through the accumulation dataset, which has been previously sorted on job name, step number, step name, program name, and run date (descending). All records for common job, step, and program information are selected into a group (a change in any of these fields causes a new group to be accumulated). Per group, the latest records up to and including the twentieth record are then summed, average values calculated, and the frequency established.

Excess records, count greater than 20, will be discarded to keep the running statistics up-to-date and also to keep the size of the accumulated records dataset within reason.

The calculated frequency per common job step is then used in connection with the average values to establish estimated yearly usage values.

Limitations

At first, the job steps that will contribute to a better level accuracy for the yearly estimations are those jobs which run more often. This is because the frequency will be more accurately calculated. Job steps that have been recorded as having been run only once in the period of record accumulation will be assumed to run only once a year. In reality they may run more often, so the estimated yearly usage calculated will be significantly less than the actual yearly usage.

OPTIONS

REXX BTCHSDB2 contains a portion of code that is commented out. It was my intention to discard the older records when a job name and job step number remained the same, but either program name or step name changed. The logic behind this is that the JCL had been changed and the older statistics would then be irrelevant. If the job names are unique then this assumption is true (normally the case for production jobs). However, we noticed that there were several generated jobs with a common job name on our system, and until we introduce a method of producing unique names for these generated jobs I have commented out this portion of the code.

If the routines produce excessive output, the limits of exclusion can be adjusted to reduce the throughput. The current limits I have set are: CPU > 0, Total service Units > 0, and runtime > 0.

REPORTING

As a method of control, and as an interim measure until we

incorporate the statistical dataset into our ISPF routines, we produce two reports within the daily run, which are written out to a separate datasets. These reports could be printed out, say once a month, and could be used as a control to show the effects of performance tuning and/or the negative effects of changed programs. One report is sorted on estimated yearly total service units, the other on average total service units.

FUTURE PLANNING

As mentioned earlier, we are planning to incorporate the generated statistics within an ISPF suite to allow user-friendly selection of job steps to monitor.

A further extension that we can visualize is that we will write a routine to compare the run statistics of the latest runs against our accumulated averages. When a large apparent discrepancy (outside of set percentage limits) is recorded, we would then report the culprit job steps for further observation, either manually or by automatic monitoring.

SUMMARY

The suite produces daily updated statistics for up to the latest 20 runs of a jobstep and reports the greatest resource (total service units) users sorted on estimated yearly usage and average usage per run. The input for the suite is the IBM standard SMF records.

The code supplied contains:

- BTCHSTA0 – initialization of the accumulated records dataset JCL.
- BTCHSTAT – daily run with statistics updates and reporting JCL.
- BTCHSFOR – initial formatting and record shrinking REXX
- BTCHSDB1 – final formatting for accumulated records dataset REXX.

- BTCHSDB2 – statistical functions and discard of old records REXX.

BTCHSTA0

```
//jobnamex JOB (SRØØ,SRØ16882,SRØ16882),'BTCHSTAØ SMFØ4 JES2.',
//             CLASS=Ø,MSGCLASS=H,REGION=ØM,
//             NOTIFY=useridx,MSGLEVEL=(1,1)
/*JOBPARM L=Ø1ØØ,G=99999
//*  -------------------------------------------------------------------
//* change 'hlq' to site high level qualifier
//* change 'smf.dataset.day(-n)' to site SMF datsets
//*  -------------------------------------------------------------------
//*  ZOS. SMF3Ø subtyp Ø4 selection sorting and reporting ..
//*  DELØ1                                                ..
//*  Clear datasets prior to Creation                     ..
//*                                                       ..
//*  SORTØ1                                               ..
//*  1st Selection SMF3Ø subtype Ø4 (SORT)                ..
//*                                                       ..
//*  REXXØ1                                               ..
//*  Formatting to summary records (BTCHSFOR)             ..
//*                                                       ..
//*  SORTØ2                                               ..
//*  Sorting on Jobname Stepname                          ..
//*                                                       ..
//*  REXXØ2                                               ..
//*  Database initialization                              ..
//*                                                       ..
//*  Initial Database to contain all steps with CPU usage ..
//*  and total service units greater than Ø               ..
//*  -------------------------------------------------------------------
//DELØ1   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 DELETE hlq.BTCHSTAT.FORMAT
 DELETE hlq.BTCHSTAT.INPUT
 DELETE hlq.BTCHSTAT.DB
 DELETE hlq.BTCHSTAT.EXTRACT
 SET MAXCC = Ø
/*
//IEFBR14 EXEC PGM=IEFBR14
//AUSGABE  DD  DISP=(,CATLG,DELETE),
//             DSN=hlq.BTCHSTAT.FORMAT,
//             DCB=(RECFM=FB,LRECL=24Ø,BLKSIZE=2784Ø),
//             SPACE=(CYL,(5Ø,5Ø)),UNIT=WORK,
//             VOL=SER=WRKØØ1
/*
```

```
//*  SORTØ1                                                  ..
//*  Conditions:
//*  ---------------------------------------------------------------
//*  (6,1,BI,EQ,X'1E') record type 3Ø
//*  (19,4,CH,EQ,C'JES2') SMF3ØWID = 'JES2' (Batch)
//*  (23,2,BI,EQ,X'ØØØ4') Record Subtype = Ø4
//*  ---------------------------------------------------------------
//SORTØ1  EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=smf.dataset.day(-1)
//         DD  DISP=SHR,DSN=smf.dataset.day(-2)
//         DD  DISP=SHR,DSN=smf.dataset.day(-3)
//         DD  DISP=SHR,DSN=smf.dataset.day(-4)
//         DD  DISP=SHR,DSN=smf.dataset.day(-5)
//         DD  DISP=SHR,DSN=smf.dataset.day(-6)
//         DD  DISP=SHR,DSN=smf.dataset.day(-7)
//         DD  DISP=SHR,DSN=smf.dataset.day(-8)
//         DD  DISP=SHR,DSN=smf.dataset.day(-9)
//         DD  DISP=SHR,DSN=.....................etc.
//SORTOUT  DD  DISP=(,CATLG,DELETE),DSN=hlq.BTCHSTAT.EXTRACT,
//             SPACE=(CYL,(25Ø,1ØØ),RLSE),UNIT=WORK,
//             DCB=(BLKSIZE=3276Ø,LRECL=32756,RECFM=VB),
//             VOL=SER=WRKØØ1
//SORTWKØ1 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ2 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ3 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ4 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ5 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD    *
 OPTION   COPY,VLSHRT
 INCLUDE  COND=(6,1,BI,EQ,X'1E',AND,19,4,CH,EQ,C'JES2',AND,            C
               (23,2,BI,EQ,X'ØØØ4'))
/*
//REXXØ1  EXEC PGM=IRXJCL,PARM='BTCHSFOR'
//SYSEXEC  DD  DISP=SHR,DSN=hlq.ISPF.EXEC
//SYSTSPRT DD  SYSOUT=*
//SMFEXTR  DD  DISP=SHR,DSN=hlq.BTCHSTAT.EXTRACT
//SMFFORM  DD  DISP=SHR,DSN=hlq.BTCHSTAT.FORMAT
/*
//*  SORTØ2                                                  ..
//*  Sort Fields:
//*  ---------------------------------------------------------------
//*  (1,8,CH,D) Jobname
//*  (19,8,CH,D) Stepname
//*  ---------------------------------------------------------------
//*  Conditions:
//*  ---------------------------------------------------------------
//*  (59,1Ø,CH,GT,C'ØØØØØØØØØØ) Total service Units
//*  (5Ø,8,CH,GT,C'ØØØØØØØØØ)  CPU Usage
```

```
//*  (114,2,CH,LE,C'Ø4')  job Completion Code
//*  -------------------------------------------------------------
//SORTØ2  EXEC PGM=SORT
//SORTIN    DD  DISP=SHR,DSN=hlq.BTCHSTAT.FORMAT
//SORTOUT   DD  DISP=(,CATLG,DELETE),DSN=hlq.BTCHSTAT.INPUT,
//              SPACE=(CYL,(5Ø,5Ø),RLSE),UNIT=WORK,
//              DCB=(RECFM=FB,LRECL=24Ø,BLKSIZE=2784Ø),
//              VOL=SER=WRKØØ1
//SORTWKØ1 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ2 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ3 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ4 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ5 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD    *
 SORT FIELDS=(1,8,CH,D,19,8,CH,D)
 INCLUDE  COND=(59,1Ø,CH,GT,C'ØØØØØØØØØØ',AND,5Ø,8,CH,GT,C'ØØØØØØØØ',  C
               AND,114,2,CH,LE,C'Ø4')
/*
//REXXØ2  EXEC PGM=IRXJCL,PARM='BTCHSDB1'
//SYSEXEC  DD  DISP=SHR,DSN=hlq.ISPF.EXEC
//SYSTSPRT DD  SYSOUT=*
//SMFINPUT DD  DISP=SHR,DSN=hlq.BTCHSTAT.INPUT
//SMFDB    DD  DISP=(,CATLG,DELETE),
//              DSN=hlq.BTCHSTAT.DB,
//              SPACE=(CYL,(25Ø,1ØØ),RLSE),
//              DCB=(RECFM=FB,LRECL=172,BLKSIZE=28ØØ36),
//              VOL=SER=ALUØØ2
/*
```

BTCHSTAT

```
//jobnamex JOB (SRØØ,SRØ16882,SRØ16882),'BTCHSTAT SMFØ4 JES2.',
//              CLASS=Ø,MSGCLASS=H,REGION=ØM,
//              NOTIFY=useridx,MSGLEVEL=(1,1)
/*JOBPARM L=Ø1ØØ,G=99999
//*  -------------------------------------------------------------
//* change 'hlq' to site high-level qualifier
//* change 'smf.dataset.day' to site SMF datasets for today
//*  eg     smf.dataset.D2ØØ4Ø63.SMFDATA
//* change date D2ØØ4Ø63 to today's date (also for reports)
//*  -------------------------------------------------------------
//*  ZOS. SMF3Ø subtyp Ø4 selection sorting statistics and..
//*  reporting Daily run.
//*  Update of Database for Step resource usage with the
//*  days SMF records. Followed by update and report from
//*  statistics file
//*  Database always reduced to last 2Ø entries per jobstep
```

```
//*
//*  DELØ1                                                  ..
//*  Clear datasets prior to Creation                       ..
//*                                                         ..
//*  SORTØ1                                                 ..
//*  Selection SMF3Ø subtype Ø4 (SORT)                      ..
//*                                                         ..
//*  REXXØ1                                                 ..
//*  Formatting to summary records (BTCHSFOR)               ..
//*                                                         ..
//*  SORTØ2                                                 ..
//*  Sorting on Jobname Stepname                            ..
//*  remove step records with CPU/SERVICE units = Ø         ..
//*  and records with return code > Ø4                      ..
//*                                                         ..
//*  REXXØ2                                                 ..
//*  Create update database (BTCHSDB1)                      ..
//*                                                         ..
//*  SORTØ3                                                 ..
//*  Merge existing database with update database           ..
//*  sort = jobname stepnum pgm stepname startdate cpu      ..
//*                                                         ..
//*  REXXØ3                                                 ..
//*  Creation of new updated database and new statistics    ..
//*  file (BTCHSDB2)                                        ..
//*                                                         ..
//*  SORTØ4                                                 ..
//*  Sort statistics file on average Total Service Units    ..
//*                                                         ..
//*  ICETØ1                                                 ..
//*  Report on above                                        ..
//*                                                         ..
//*  SORTØ5                                                 ..
//*  Sort statistics file on yearly est. Total Service Units
//*                                                         ..
//*  ICETØ2                                                 ..
//*  Report on above
//*                                                         ..
//*  RENØ1                                                  ..
//*  Delete old datbase and rename new database to oldname..
//*                                                         ..
//*  ---------------------------------------------------------------
//*  Note: Change the Date Qualifier to relevant date       ..
//*                                                         ..
//*  Daily DSN  : smf.dataset.D2ØØ4Ø63.SMFDATA  ..
//*  Saved for 1Ø years in HSM                              ..
//*
//SET1     SET SMFAL1Ø=smf.dataset.D2ØØ4Ø63.SMFDATA
//*
//*  ---------------------------------------------------------------
```

```
//DELØ1   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 DELETE hlq.BTCHSTAT.EXTRACT
 DELETE hlq.BTCHSTAT.FORMAT
 DELETE hlq.BTCHSTAT.INPUT
 DELETE hlq.BTCHSTAT.DBØ
 DELETE hlq.BTCHSTAT.DB2
 DELETE hlq.BTCHSTAT.DBST
 SET MAXCC = Ø
/*
//IEFBR14 EXEC PGM=IEFBR14
//FORMAT   DD  DISP=(,CATLG,DELETE),
//             DSN=hlq.BTCHSTAT.FORMAT,
//             DCB=(RECFM=FB,LRECL=24Ø,BLKSIZE=2784Ø),
//             SPACE=(CYL,(1Ø,1Ø)),UNIT=WORK,
//             VOL=SER=WRKØØ1
/*
//*  SORTØ1                                                  ..
//*  Conditions:
//*  ------------------------------------------------------------------
//*  (6,1,BI,EQ,X'1E') record type 3Ø
//*  (19,4,CH,EQ,C'JES2') SMF3ØWID = 'JES2' (Batch)
//*  (23,2,BI,EQ,X'ØØØ4') Record Subtype = Ø4
//*  ------------------------------------------------------------------
//SORTØ1  EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=&SMFAL1Ø
//SORTOUT  DD  DISP=(,CATLG,DELETE),DSN=hlq.BTCHSTAT.EXTRACT,
//             SPACE=(CYL,(25Ø,1ØØ),RLSE),UNIT=WORK,
//             DCB=(BLKSIZE=3276Ø,LRECL=32756,RECFM=VB),
//             VOL=SER=WRKØØ1
//SORTWKØ1 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ2 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ3 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ4 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ5 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD    *
 OPTION   COPY,VLSHRT
 INCLUDE  COND=(6,1,BI,EQ,X'1E',AND,19,4,CH,EQ,C'JES2',AND,            C
               (23,2,BI,EQ,X'ØØØ4'))
/*
//REXXØ1  EXEC PGM=IRXJCL,PARM='BTCHSFOR'
//SYSEXEC  DD  DISP=SHR,DSN=hlq.ISPF.EXEC
//SYSTSPRT DD  SYSOUT=*
//SMFEXTR  DD  DISP=SHR,DSN=hlq.BTCHSTAT.EXTRACT
//SMFFORM  DD  DISP=SHR,DSN=hlq.BTCHSTAT.FORMAT
/*
//*  SORTØ2                                                  ..
```

```
//*  Sort Fields:
//*  ------------------------------------------------------------------
//*  (1,8,CH,D) Jobname
//*  (19,8,CH,D) Stepname
//*  ------------------------------------------------------------------
//*  Conditions:
//*  ------------------------------------------------------------------
//*  (59,1Ø,CH,GT,C'ØØØØØØØØØØ) Total service Units
//*  (5Ø,8,CH,GT,C'ØØØØØØØØØØ)  CPU Usage
//*  (114,2,CH,LE,C'Ø4')  job Completion Code
//*  ------------------------------------------------------------------
//SORTØ2  EXEC PGM=SORT
//SORTIN    DD  DISP=SHR,DSN=hlq.BTCHSTAT.FORMAT
//SORTOUT   DD  DISP=(,CATLG,DELETE),DSN=hlq.BTCHSTAT.INPUT,
//              SPACE=(CYL,(1ØØ,1ØØ),RLSE),UNIT=WORK,
//              DCB=(RECFM=FB,LRECL=24Ø,BLKSIZE=2784Ø),
//              VOL=SER=WRKØØ1
//SORTWKØ1 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ2 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ3 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ4 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SORTWKØ5 DD  UNIT=339Ø,SPACE=(CYL,(15,15),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSOUT    DD  SYSOUT=*
//SYSIN     DD  *
 SORT FIELDS=(1,8,CH,D,19,8,CH,D)
 INCLUDE  COND=(59,1Ø,CH,GT,C'ØØØØØØØØØØ',AND,5Ø,8,CH,GT,C'ØØØØØØØØ',  C
               AND,114,2,CH,LE,C'Ø4')
/*
//REXXØ2  EXEC PGM=IRXJCL,PARM='BTCHSDB1'
//SYSEXEC  DD  DISP=SHR,DSN=hlq.ISPF.EXEC
//SYSTSPRT DD  SYSOUT=*
//SMFINPUT DD  DISP=SHR,DSN=hlq.BTCHSTAT.INPUT
//SMFDB     DD  DISP=(,CATLG,DELETE),
//              DSN=hlq.BTCHSTAT.DBØ,
//              SPACE=(CYL,(1ØØ,1ØØ),RLSE),UNIT=WORK,
//              DCB=(RECFM=FB,LRECL=172,BLKSIZE=28Ø36),
//              VOL=SER=WRKØØ1
/*
//*  SORTØ3                                                  ..
//*  Sort Fields:
//*  ------------------------------------------------------------------
//*  (1,8,A) Jobname
//*  (28,2,A) Stepnumber
//*  (19,8,A) pgmname
//*  (1Ø,8,A) stepname
//*  (133,8,D) start date
//*  ------------------------------------------------------------------
//*  Conditions:
//*  ------------------------------------------------------------------
```

```
//*  (53,12,CH,GT,C'000000000000') runtime (seconds)
//*  -------------------------------------------------------------
//SORT03  EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=hlq.BTCHSTAT.DB0
//         DD  DISP=SHR,DSN=hlq.BTCHSTAT.DB
//SORTOUT  DD  DISP=SHR,DSN=hlq.BTCHSTAT.DB
//SORTWK01 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK02 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK03 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK04 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK05 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
 SORT FIELDS=(1,8,A,28,2,A,19,8,A,10,8,A,133,8,D),FORMAT=CH
 INCLUDE  COND=(53,12,CH,GT,C'000000000000')
/*
//REXX03  EXEC PGM=IRXJCL,PARM='BTCHSDB2'
//SYSEXEC  DD  DISP=SHR,DSN=hlq.ISPF.EXEC
//SYSTSPRT DD  SYSOUT=*
//BTCHSDBI DD  DISP=SHR,DSN=hlq.BTCHSTAT.DB
//BTCHSDBO DD  DISP=(,CATLG,DELETE),
//             DSN=hlq.BTCHSTAT.DB2,
//             SPACE=(CYL,(250,100),RLSE),
//             DCB=(RECFM=FB,LRECL=172,BLKSIZE=28036),
//             VOL=SER=ALU003
//BTCHSTAT DD  DISP=(,CATLG,DELETE),
//             DSN=hlq.BTCHSTAT.DBST,
//             SPACE=(CYL,(50,25),RLSE),
//             DCB=(RECFM=FB,LRECL=265,BLKSIZE=28090),
//             VOL=SER=ALU003
/*
//SORT04  EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=hlq.BTCHSTAT.DBST
//SORTOUT  DD  DISP=SHR,DSN=hlq.BTCHSTAT.DBST
//SORTWK01 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK02 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK03 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK04 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK05 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
 SORT FIELDS=(119,10,CH,D)
/*
//ICET01  EXEC PGM=ICETOOL
//ICEIN    DD  DISP=SHR,DSN=hlq.BTCHSTAT.DBST
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//TOOLMSG  DD  SYSOUT=*
```

```
//DFSMSG   DD  SYSOUT=*
//LIST1    DD  DISP=(,CATLG,DELETE),
//             DSN=hlq.BTCHSTAT.D2004063.STATLST1,
//             SPACE=(CYL,(5,5),RLSE),
//             DCB=(RECFM=FB,LRECL=289,BLKSIZE=28033),
//             VOL=SER=ALU001
//TOOLIN   DD  *
 DISPLAY FROM(ICEIN) LIST(LIST1)  -
   TITLE('BATCH JOB - SERVICE UNITS USAGE - JOBSTEP SUMMARY') -
   PAGE DATE TIME -
   HEADER('JOBNAME') ON(1,8,CH)  -
   HEADER('STEPNAME') ON(10,8,CH)  -
   HEADER('PGMNAME') ON(19,8,CH)  -
   HEADER('STEPNUM') ON(28,2,CH)  -
   HEADER('AVG-Total-SRV-Units') ON(229,6,PD,A2)  -
   HEADER('YY-Total-SRV-Units') ON(177,7,PD,A2)  -
   HEADER('AVG-CPU-SRV-UNITS') ON(236,6,PD,A2)  -
   HEADER('YY-CPU-SRV-UNITS') ON(185,7,PD,A2)  -
   HEADER('AVG-SRB-SRV-UNITS') ON(243,6,PD,A2)  -
   HEADER('YY-SRB-SRV-UNITS') ON(193,7,PD,A2)  -
   HEADER('AVG-IO-SRV-UNITS') ON(250,6,PD,A2)  -
   HEADER('YY-IO-SRV-UNITS') ON(201,7,PD,A2)  -
   HEADER('AVG-MSO-SRV-UNITS') ON(257,6,PD,A2)  -
   HEADER('YY-MSO-SRV-UNITS') ON(209,7,PD,A2)  -
   HEADER('AVG-Runtime.SECS') ON(223,5,PD,A2)  -
   HEADER('YY-Runtime.SECS') ON(217,5,PD,A2)  -
   HEADER('INC. Records Count') ON(264,2,PD,A0)  -
   LINES(38) -
   BETWEEN(2) -
   AVERAGE('AVERAGE')
/*
//SORT05  EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=hlq.BTCHSTAT.DBST
//SORTOUT  DD  DISP=SHR,DSN=hlq.BTCHSTAT.DBST
//SORTWK01 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK02 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK03 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK04 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SORTWK05 DD  UNIT=3390,SPACE=(CYL,(15,15),RLSE)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
 SORT FIELDS=(31,13,CH,D)
/*
//ICET02  EXEC PGM=ICETOOL
//ICEIN    DD  DISP=SHR,DSN=hlq.BTCHSTAT.DBST
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//TOOLMSG  DD  SYSOUT=*
//DFSMSG   DD  SYSOUT=*
```

```
//LIST2    DD  DISP=(,CATLG,DELETE),
//             DSN=hlq.BTCHSTAT.D2004063.STATLST2,
//             SPACE=(CYL,(5,5),RLSE),
//             DCB=(RECFM=FB,LRECL=289,BLKSIZE=28033),
//             VOL=SER=ALU001
//TOOLIN   DD  *
 DISPLAY FROM(ICEIN) LIST(LIST2)  -
   TITLE('BATCH JOB - SERVICE UNITS USAGE - JOBSTEP SUMMARY') -
   PAGE DATE TIME -
   HEADER('JOBNAME') ON(1,8,CH)  -
   HEADER('STEPNAME') ON(10,8,CH)  -
   HEADER('PGMNAME') ON(19,8,CH)  -
   HEADER('STEPNUM') ON(28,2,CH)  -
   HEADER('YY-Total-SRV-Units') ON(177,7,PD,A2)  -
   HEADER('AVG-Total-SRV-Units') ON(229,6,PD,A2)  -
   HEADER('YY-CPU-SRV-UNITS') ON(185,7,PD,A2)  -
   HEADER('AVG-CPU-SRV-UNITS') ON(236,6,PD,A2)  -
   HEADER('YY-SRB-SRV-UNITS') ON(193,7,PD,A2)  -
   HEADER('AVG-SRB-SRV-UNITS') ON(243,6,PD,A2)  -
   HEADER('YY-IO-SRV-UNITS') ON(201,7,PD,A2)  -
   HEADER('AVG-IO-SRV-UNITS') ON(250,6,PD,A2)  -
   HEADER('YY-MSO-SRV-UNITS') ON(209,7,PD,A2)  -
   HEADER('AVG-MSO-SRV-UNITS') ON(257,6,PD,A2)  -
   HEADER('YY-Runtime.SECS') ON(217,5,PD,A2)  -
   HEADER('AVG-Runtime.SECS') ON(223,5,PD,A2)  -
   HEADER('INC. Records Count') ON(264,2,PD,A0)  -
   LINES(38) -
   BETWEEN(2) -
   AVERAGE('AVERAGE')
/*
//REN01   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 DELETE hlq.BTCHSTAT.DB
 ALTER hlq.BTCHSTAT.DB2 NEWNAME(hlq.BTCHSTAT.DB)
 SET MAXCC = 0
/*
```

BTCHSFOR

```
/* REXX *  BTCHSFOR ************************************************** */
/*Description: SMF records of subtype 04 to show resource usage of    */
/*               batch jobs. Per total service units.                 */
/* ---------------------------------------------------------------- */
/* Created ....: 03.02.2004      Rolf Parker                        */
/* ---------------------------------------------------------------- */
/* Updates ....: 03.02.2004 RP Creation                             */
/* **************************************************************** */
/* **************************************************************** */
```

```
/* The records after the selection/sort are now 3 bytes shorter due   */
/* to the conversion from VBA to VB. This will be taken into account  */
/* with the adjustment variable 'adj'. ie. the offsets will naturally */
/* be 3 bytes shorter                                                 */
/* ****************************************************************** */
 adj = 3    /* offset adjustment */
/* ****************************************************************** */
 numeric digits 12                   /* increase num. digits 9 --> 12 */
/* read input ---------------------------------------------------- */
"execio * diskr SMFEXTR (stem ESMF. finis)"

 do i          = 1 to ESMF.Ø
    /* Header/Self-defining Section ---------------------------------- */
    SMF3ØRTY   = x2d(c2x(SUBSTR(ESMF.i,2,1)))       /* SMF Type       */
    SMF3ØSTP   = x2d(c2x(SUBSTR(ESMF.i,19,2)))      /* SMF Subtype    */
    SMF3ØWID   = SUBSTR(ESMF.i,15,4)           /* Work type indicator */
    SMF3ØSID   = SUBSTR(ESMF.i,11,4)                /* SID            */
    SMF3ØION   = x2d(c2x(SUBSTR(Record,35,2)))      /* Identification */
    SMF3ØCON   = x2d(c2x(SUBSTR(ESMF.i,59,2)))   /* Processor Sections*/
    SMF3ØPON   = x2d(c2x(SUBSTR(ESMF.i,83,2)))  /*Performance Sections*/
    /*                                                                */
    select
     when SMF3ØION = Ø then iterate             /* no Identification  */
     when SMF3ØCON = Ø then iterate             /* no Processor       */
     when SMF3ØPON = Ø then iterate             /* no Performance     */
     otherwise              nop
    end
    /* Identification Section ---------------------------------------- */
    SMF3ØIOF   = x2d(c2x(SUBSTR(ESMF.i,32-adj,4)))-adj
                                                /* Identification */
    SMF3ØJBN   = SUBSTR(ESMF.i,SMF3ØIOF,8)          /* Job Name       */
    SMF3ØJNM   = SUBSTR(ESMF.i,SMF3ØIOF+32,8)       /* Job Number     */
    SMF3ØPGM   = SUBSTR(ESMF.i,SMF3ØIOF+8,8)        /* PGM=xxxxxxxx   */
    SMF3ØSTM   = SUBSTR(ESMF.i,SMF3ØIOF+16,8)       /* Step Name      */
    SMF3ØSTN   = C2D(SUBSTR(ESMF.i,SMF3ØIOF+4Ø,2))  /* Steb Number    */
    SMF3ØUSR   = SUBSTR(ESMF.i,SMF3ØIOF+8Ø,2Ø)      /* Programmer Name*/
    SMF3ØGRP   = SUBSTR(ESMF.i,SMF3ØIOF+1ØØ,8)      /* RACF User      */
    SMF3ØRUD   = SUBSTR(ESMF.i,SMF3ØIOF+1Ø8,8)      /* RACF Group     */
    /* -- Start Job Time and Day ---------------------------------- */
    SMF3ØSIT   = c2x(SUBSTR(ESMF.i,SMF3ØIOF+56,4))
    SMF3ØSIT   = x2d(SMF3ØSIT)
    SMF3ØSTD   =
Date("S",substr(c2x(SUBSTR(ESMF.i,SMF3ØIOF+6Ø,4)),3,5),"J")
    /* -- End Job Time and Day ------------------------------------ */
    SMF3ØTME   = c2x(SUBSTR(ESMF.i,6-adj,4))
    SMF3ØTME   = x2d(SMF3ØTME)
    SMF3ØDTE   = Date("S",substr(c2x(SUBSTR(ESMF.i,1Ø-adj,4)),3,5),"J")
    /* Completion Section -------------------------------------------- */
    SMF3ØTOF   = x2d(c2x(SUBSTR(ESMF.i,48-adj,4)))-adj
                                              /* Identification    */
```

```
    SMF30SCC   = RIGHT(C2D(SUBSTR(ESMF.i,SMF30TOF,2)),2,'0') /* JOB CC*/
    /* Processor Accounting Section ------------------------------- */
    SMF30COF   = x2d(c2x(SUBSTR(ESMF.i,56-adj,4)))-adj /* Processor   */
    SMF30CPT   = x2d(c2x(SUBSTR(ESMF.i,SMF30COF+4,4))) /* CPU Time TCB*/
    SMF30CPS   = x2d(c2x(SUBSTR(ESMF.i,SMF30COF+8,4))) /* CPU Time SRB*/
    /* CPU Time total --------------------------------------------- */
    SMF30CPU   = SMF30CPT + SMF30CPS             /* CPU Time tcb+SRB  */
    /* Performance Section ---------------------------------------- */
    SMF30POF   = x2d(c2x(SUBSTR(ESMF.i,80-adj,4)))-adj
                                                 /* Perform. Sections */
    SMF30SRV   = x2d(c2x(SUBSTR(ESMF.i,SMF30POF,4)))
                                                 /* Total Serv. Units */
    SMF30CSU   = x2d(c2x(SUBSTR(ESMF.i,SMF30POF+4,4)))
                                                 /* CPU Serv. Units   */
    SMF30SRB   = x2d(c2x(SUBSTR(ESMF.i,SMF30POF+8,4)))
                                                 /* SRB Serv. Units   */
    SMF30IO    = x2d(c2x(SUBSTR(ESMF.i,SMF30POF+12,4)))
                                                 /* I/O Serv. Units   */
    SMF30MSO   = x2d(c2x(SUBSTR(ESMF.i,SMF30POF+16,4)))
                                                 /* MSO Serv. Units   */
    /* ---------------------------------------------------------------- */
    Queue ' '||,
        Left(SMF30JBN,8,' '),        /* Job Name                      */
        Left(SMF30JNM,8,' '),        /* Job Number                    */
        Left(SMF30STM,8,' '),        /* Step Name                     */
        Left(SMF30PGM,8,' '),        /* PGM=xxxxxxxx (Name)           */
        Right(SMF30STN,2,' '),       /* Step number                   */
        Left(SMF30RUD,8,' '),        /* Racf User                     */
        Right(SMF30CPU,8,'0'),       /* Step CPU Time                 */
        Right(SMF30SRV,10,'0'),      /* Total Service Units           */
        Right(SMF30CSU,10,'0'),      /* CPU Service Units             */
        Right(SMF30SRB,10,'0'),      /* SRB Service Units             */
        Right(SMF30IO,10,'0'),       /* I/O Service Units             */
        Right(SMF30MSO,10,'0'),      /* MSO Service Units             */
        Left(SMF30SCC,2,'0'),        /* Step completion code          */
        Left(SMF30STD,8,' '),        /* Initiator start date          */
        Right(SMF30SIT,8,'0'),       /* Initiator start time          */
        Left(SMF30DTE,8,' '),        /* End Date                      */
        Right(SMF30TME,8,'0'),       /* End Time                      */
        Right(SMF30SID,4,' ')        /* SubSystemID                   */
 End
/* write output ------------------------------------------------------ */
"execio "queued()" diskw SMFFORM (FINIS"
Return
```

BTCHSDB1

```
/* REXX *  BTCHSDB1 ************************************************** */
/* Description:  Reporter for Output from BTCHSFOR (SORT02)           */
```

```
/*                                                                   */
/*                                                                   */
/* ----------------------------------------------------------------- */
/* Created ....: Ø3.Ø2.2ØØ4      Rolf Parker                         */
/* ----------------------------------------------------------------- */
/* Updates ....: Ø3.Ø2.2ØØ4 RP Creation                              */
/* ***************************************************************** */
/* ***************************************************************** */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/* ***************************************************************** */
 numeric digits 12                  /* numeric increase from 9 to 12  */
   dd = 'SMFINPUT'
   "EXECIO * DISKR "dd" (STEM SMFIN. FINIS"
   do rec = 1 to smfin.Ø
     job = word(smfin.rec,1)
     jobnum = word(smfin.rec,2)
     stepname = word(smfin.rec,3)
     pgmname = word(smfin.rec,4)
     stepnum = word(smfin.rec,5)
     racfuser = word(smfin.rec,6)
     cputime = word(smfin.rec,7)
     days = Ø
     if word(smfin.rec,14) /= word(smfin.rec,16) then
       do
         datestr = word(smfin.rec,14)
         dateend = word(smfin.rec,16)
         daysstr = date('B',datestr,'S')
         daysend = date('B',dateend,'S')
         days = daysend - daysstr
       end
     dayshs = days * 8640ØØØ
     runtime = (word(smfin.rec,17)+dayshs) - word(smfin.rec,15)
     runtime = trunc(runtime/1ØØ)
     totsrv = word(smfin.rec,8)
     cpusrv = word(smfin.rec,9)
     srbsrv = word(smfin.rec,1Ø)
     iosrv = word(smfin.rec,11)
     msosrv = word(smfin.rec,12)
     cc = word(smfin.rec,13)
     runfromd = word(smfin.rec,14)
     runfromt = word(smfin.rec,15)
     runfromt = hhmmss(runfromt)
     runtod = word(smfin.rec,16)
     runtot = word(smfin.rec,17)
     runtot = hhmmss(runtot)
     sid = word(smfin.rec,18)
```

```
     queue,
          left(job,8,' '),
          left(stepname,8,' '),
          left(pgmname,8,' '),
          left(stepnum,2,' '),
          left(racfuser,8,' '),
          right(cputime,12,' '),
          right(runtime,12,'0'),
          right(totsrv,10,' '),
          right(cpusrv,10,' '),
          right(srbsrv,10,' '),
          right(iosrv,10,' '),
          right(msosrv,10,' '),
          left(jobnum,8,' '),
          right(cc,2,'0'),
          right(runfromd,8,' '),
          right(runfromt,8,' '),
          right(runtod,8,' '),
          right(runtot,8,' '),
          left(sid,4,' ')

   end

"execio "queued()" diskw SMFDB (FINIS"
return

hhmmss:
arg timein
  splt1 = ':'
  splt2 = ':'
    hh      = right(timein%100%3600,2,'0')
    mm      = right(timein%100//3600%60,2,'0')
    ss      = right(timein%100//60,2,'0')
    hs      = right(timein-(trunc(timein/100)*100),2,'0')
    if hs > 50 then ss = right(ss+1,2,'0')
    timeout    = hh||splt1||mm||splt2||ss
Return timeout
```

BTCHSDB2

```
/* REXX *  BTCHSDB2 ************************************************** */
/* Description:  Analyser for BTCHSTAT DB                             */
/*                                                                    */
/*                                                                    */
/* ------------------------------------------------------------------ */
/* Created ....: 05.02.2004      Rolf Parker                          */
/* ------------------------------------------------------------------ */
/* Updates ....: 05.02.2004 RP Creation                               */
/* ****************************************************************** */
```

```
/* ***************************************************************** */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/* ***************************************************************** */
 numeric digits 12                  /* numeric increase from 9 to 12  */
 underline = '-----------------------------------------------------',
 || ,-----------------------------------------------------------',
 || ,-------------------------'
 dd1 = ,BTCHSDBI'
 dd2 = ,BTCHSDBO'
 dd3 = ,BTCHSTAT'
 "EXECIO * DISKR "dd1" (STEM dbin. FINIS"
 outpos = Ø
 count = Ø
 btchdbout. = ''
 inpos = 1
 /* set initial select keys */
 key_new = word(dbin.2,1) word(dbin.2,2) word(dbin.2,3) word(dbin.2,4)
 key_cur = word(dbin.1,1) word(dbin.1,2) word(dbin.1,3) word(dbin.1,4)
 key_old = 'DUMMY'
 /* set current record */
 cur_rec = dbin.1
 count = 1
 do forever
   do forever /* count loop */
     select
       when ((key_old<>key_cur)&(key_cur<>key_new)&, /* single record */
            (key_new<>key_old)) then
         do
           dbwork.count = cur_rec
           key_old = key_cur
           key_cur = key_new
           cur_rec = dbin.inpos
           inpos = inpos + 1
           key_new = word(dbin.inpos,1) word(dbin.inpos,2),
                     word(dbin.inpos,3) word(dbin.inpos,4)
           call calc_stat                        /* process block   */
           count = 1
           if inpos > (dbin.Ø+1) then leave
         end
       when (key_old='DUMMY') then
         do                                       /* 1st record     */
           dbwork.count = cur_rec
           cur_rec = dbin.inpos
           inpos = inpos + 1
           key_old = key_cur
           key_cur = key_new
```

```
            key_new = word(dbin.inpos,1) word(dbin.inpos,2),
                      word(dbin.inpos,3) word(dbin.inpos,4)
          end
        when (key_cur=key_new) then
          do                                          /* record in block */
            dbwork.count = cur_rec
            cur_rec = dbin.inpos
            inpos = inpos + 1
            key_old = key_cur
            key_cur = key_new
            key_new = word(dbin.inpos,1) word(dbin.inpos,2),
                      word(dbin.inpos,3) word(dbin.inpos,4)
            count = count + 1
          end
        when key_cur <> key_new then                /* change of key    */
          do
            dbwork.count = cur_rec
            cur_rec = dbin.inpos
            key_old = key_cur
            key_cur = key_new
            inpos = inpos + 1
            key_new = word(dbin.inpos,1) word(dbin.inpos,2),
                      word(dbin.inpos,3) word(dbin.inpos,4)
            call calc_stat                          /* process block    */
            count = 1
            if inpos > (dbin.0+1) then leave
          end
        otherwise nop
      end
    end
    if inpos > dbin.0 then leave
  end
  /* write statistic records */
  "execio "queued()" diskw "dd3" (FINIS"
  /* write reduced db records */
  "execio "outpos" diskw "dd2" (stem dbout. FINIS"
  exit

calc_stat:
  /* sift out the old info after jcl changes
       step_pgm_changed = 0
       key_todo =  word(dbwork.1,1)||word(dbwork.1,4)
       if key_todo = key_done then step_pgm_changed = 1
       if step_pgm_changed then return
       key_done =  word(dbwork.1,1)||word(dbwork.1,4)
  */
  /* limit statistical sample to newest 20 SMF records */
  if count > 20 then count = 20
  /* initialise */
  stat_rec = ''
```

```
jobname  = word(dbwork.1,1)
stepname = word(dbwork.1,2)
pgmname  = word(dbwork.1,3)
stepnum  = word(dbwork.1,4)
cputot = Ø
runtimetot = Ø
totsrvtot = Ø
cpusrvtot = Ø
srbsrvtot = Ø
iosrvtot  = Ø
msosrvtot = Ø

/* process block */
do stat = 1 to count
  outpos = outpos + 1
  /* output of db record */
  dbout.outpos = dbwork.stat
  /* sum action */
  runtimetot = runtimetot + word(dbwork.stat,7)
  totsrvtot = totsrvtot + word(dbwork.stat,8)
  cpusrvtot = cpusrvtot + word(dbwork.stat,9)
  srbsrvtot = srbsrvtot + word(dbwork.stat,1Ø)
  iosrvtot  = iosrvtot + word(dbwork.stat,11)
  msosrvtot = msosrvtot + word(dbwork.stat,12)
end
/* resource averages */
runtimeav = trunc(runtimetot / count)
totsrvav = trunc(totsrvtot / count)
cpusrvav = trunc(cpusrvtot / count)
srbsrvav = trunc(srbsrvtot / count)
iosrvav  = trunc(iosrvtot / count)
msosrvav = trunc(msosrvtot / count)
/* sample period in days */
dayfirst = word(dbwork.count,15)
daylast  = word(dbwork.1,15)
dayfirst = date('B',dayfirst,'S')
daylast = date('B',daylast,'S')
days = (daylast - dayfirst) + 1
/* frequency of job/jobstep per year */
if count = 1 then
  yearfactor = 1
else
  do
    frequency = trunc(count/days,8)
    yearfactor = frequency * 365
  end
/* calculated yearly resource use */
runtimeyy = trunc(runtimeav * yearfactor)
totsrvyy = trunc(totsrvav * yearfactor)
cpusrvyy = trunc(cpusrvav * yearfactor)
```

```
srbsrvyy = trunc(srbsrvav * yearfactor)
iosrvyy  = trunc(iosrvav * yearfactor)
msosrvyy = trunc(msosrvav * yearfactor)
/* PD (packed decimal) conversion */
pdtotsrvyy  = totsrvyy || 'C'
pdtotsrvyy  = right(pdtotsrvyy,14,'0')
pdtotsrvyy  = x2c(pdtotsrvyy)
pdcpusrvyy  = cpusrvyy || 'C'
pdcpusrvyy  = right(pdcpusrvyy,14,'0')
pdcpusrvyy  = x2c(pdcpusrvyy)
pdsrbsrvyy  = srbsrvyy || 'C'
pdsrbsrvyy  = right(pdsrbsrvyy,14,'0')
pdsrbsrvyy  = x2c(pdsrbsrvyy)
pdiosrvyy   = iosrvyy  || 'C'
pdiosrvyy   = right(pdiosrvyy,14,'0')
pdiosrvyy   = x2c(pdiosrvyy)
pdmsosrvyy  = msosrvyy || 'C'
pdmsosrvyy  = right(pdmsosrvyy,14,'0')
pdmsosrvyy  = x2c(pdmsosrvyy)
pdruntimeyy = runtimeyy|| 'C'
pdruntimeyy = right(pdruntimeyy,10,'0')
pdruntimeyy = x2c(pdruntimeyy)
pdruntimeav = runtimeav|| 'C'
pdruntimeav = right(pdruntimeav,10,'0')
pdruntimeav = x2c(pdruntimeav)
pdtotsrvav  = totsrvav|| 'C'
pdtotsrvav  = right(pdtotsrvav,12,'0')
pdtotsrvav  = x2c(pdtotsrvav)
pdcpusrvav  = cpusrvav || 'C'
pdcpusrvav  = right(pdcpusrvav,12,'0')
pdcpusrvav  = x2c(pdcpusrvav)
pdsrbsrvav  = srbsrvav || 'C'
pdsrbsrvav  = right(pdsrbsrvav,12,'0')
pdsrbsrvav  = x2c(pdsrbsrvav)
pdiosrvav   = iosrvav  || 'C'
pdiosrvav   = right(pdiosrvav,12,'0')
pdiosrvav   = x2c(pdiosrvav)
pdmsosrvav  = msosrvav || 'C'
pdmsosrvav  = right(pdmsosrvav,12,'0')
pdmsosrvav  = x2c(pdmsosrvav)
pdcount     = count    || 'C'
pdcount     = right(pdcount,4,'0')
pdcount     = x2c(pdcount)
/* output statistical records */
queue       ,
  left(jobname,8,' '),
  left(stepname,8,' '),
  left(pgmname,8,' '),
 right(stepnum,2,' '),
 right(totsrvyy,13,' '),
```

```
   right(cpusrvyy,13,' '),
   right(srbsrvyy,13,' '),
   right(iosrvyy,13,' '),
   right(msosrvyy,13,' '),
   right(runtimeyy,8,' '),
   right(runtimeav,8,' '),
   right(totsrvav,10,' '),
   right(cpusrvav,10,' '),
   right(srbsrvav,10,' '),
   right(iosrvav,10,' '),
   right(msosrvav,10,' '),
   right(count,2,' '),
   pdtotsrvyy,
   pdcpusrvyy,
   pdsrbsrvyy,
   pdiosrvyy,
   pdmsosrvyy,
   pdruntimeyy,
   pdruntimeav,
   pdtotsrvav,
   pdcpusrvav,
   pdsrbsrvav,
   pdiosrvav,
   pdmsosrvav,
   pdcount
return
```

Rolf Parker
Systems Programmer (Germany)

Analysing legacy applications

Analysis is an art perfected over a period of time. In most shops, legacy applications lack proper documentation. Most of the time code has comments here and there that help in understanding the underlying logic to a certain extent, but not completely. So as the need arises, programmers/analysts are asked to document the application in some form of analysis document. Most of the time there are no clear guidelines on the structure of this document, and different shops come up with their own versions. We'll focus here on the actual analysis part and leave the

structure of the document to the respective shops and programmers.

COBOL has been one of the most widely-used languages in legacy systems. In COBOL programs, the function-oriented structured programming approach is followed. The code is developed in a top-down way using a modular programming technique. But still there are many applications using the old GOTO approach to programming, which must be avoided when writing newer programs or adding new code to existing programs.

While analysing, we've got to keep in mind that this is not a stand-alone program and therefore try to find out at what point the subroutines are being called and the purpose of the subroutine. We can put the purpose in the analysis document. Generally subroutines are used with a 'write once call again' philosophy so the same subroutine can be called in different places from a program or from different programs.

Some shops use Data Manager or Data Dictionary, which certainly make life easier for programmers. These are generally based on SSAD (Structured System Analysis and Design) methodology. They use elements, data flows, and processes that correspond to individual data items, group items, and functions of the program. Based on this, Data Flow Diagrams (DFDs) can be created and put in the document to understand better the functionalities.

While solving maintenance issues, analysis of the problem is vital. But there is nothing like simulation of the problem. If you can simulate the problem with different sets of data, you get an insight into the problem. On the other hand, in the case of production abends, we generally go with the offset of the instruction in the dump. Tools like Abend-Aid help in getting the offset as well as the instruction where the abend occurred.

Coming back to analysis actually to understand the business logic of the program, not only do we have to understand the purpose of each section or paragraph but also we have to keep track of some variables. We need to remember the context in which this variable is being called. Variable names in COBOL

programs are generally indicative of the purpose, so pay close attention to this. A Notepad document can be opened along with the mainframe programs where the programmer can put these variables or a few important sections or paragraphs of the program. It avoids losing sight of the problem or the logic being analysed.

Analysis is not just about having facts – we need to make them understandable to others. Put yourself in the shoes of a novice/ other programmers and then read what you have put in the analysis document. Is this understandable? Bear in mind that what you are writing in the document is not only for your use later but others may refer to it while doing maintenance or enhancing the application. It could also be used for review or reference. Also, is it language-independent? Try as much as possible to make it so.

Some automated tools are available in the market that help in application analysis to a certain extent but not completely.

One thing to take care to avoid while preparing this analysis document is information overload or over-simplification of the logic. After a self-review of this document, submit it for a review to one of your peer programmers or manager. This is a good practice, giving him/her the opportunity to point out what is not clear, needs more explanation, or even is incorrect.

Aseem Anand
Programmer Analyst
Cognizant Technology Solutions (India)

z/OS Dynamic Channel path Management (DCM)

The Intelligent Resource Director (IRD) is a new feature of z/OS V1R1. IRD extends the concept of goal-oriented resource management by allowing users to group MVS images that reside on the same physical server running in LPAR mode, and in the same sysplex, into an LPAR cluster.

This gives Workload Manager (WLM) the ability to manage resources, both processor and DASD I/O, not just in one single image but across the entire cluster of system images.

The three functions that make up IRD are:

- LPAR CPU management, which lets WLM distribute processor resource across an LPAR cluster by dynamically adjusting the LPAR weights in response to changes in the workload requirements. When important work is not meeting its goals, WLM will raise the weight of the partition where that work is running, thereby giving it more processing power. As part of LPAR CPU management, WLM will also optimize the number of on-line logical CPUs configured on-line to each partition.
- Dynamic channel path management, which lets WLM dynamically move managed channel paths through one ESCON Director from one I/O control unit to another, in response to changes in the workload requirements. DCM exploits the dynamic I/O reconfiguration capability to add and remove physical paths to control units.
- Channel subsystem priority queueing, which is an extension o f I/O priority queueing, a concept that has been evolving in MVS over the past few years. If important work is missing its goals because of I/O contention on channels shared with other work, it will be given a higher channel subsystem I/O priority than the less important work.

In order to implement these IRD functions, you should use a zSeries server (Z/9*xx*) running in z/Architecture mode (64 bits).

This article will describe in detail how to implement Dynamic Channel path Management (DCM).

DCM BENEFITS

Improved and maximized I/O bandwidth

DCM can provide improved performance by dynamically moving

the available channel bandwidth to where it is most needed. Prior to DCM, you had manually to balance your available channels across your I/O devices, trying to provide sufficient paths to handle the average load on every controller. This means that at any one time, some controllers probably have more I/O paths available than they need, while other controllers possibly have too few.

DCM attempts to balance the responsiveness of the available channels by moving channels to the controllers that require additional bandwidth.

Simplified I/O configuration

Dynamic Channel path Management also simplifies the task of defining your I/O configuration. Without DCM, you have to identify the bandwidth required for each LCU to provide acceptable performance at peak times. Because DCM can dynamically move I/O paths to the LCUs that are experiencing channel delays, you can reduce the CU/channel-level capacity planning and balancing activity that was necessary prior to DCM.

Using DCM, you are required to define only a minimum of one non-managed path and up to seven managed paths to each controller (although a realistic minimum of at least two non-managed paths are recommended), with DCM taking responsibility for adding more paths as required, and ensuring that the paths meet the objectives of availability and performance.

DCM PREREQUISITES

Processor requirements

In order to implement DCM you must be running on an IBM zSeries 900 or later CPC.

Operating system requirements

The only operating system that can use a managed path is z/OS 1.1 or higher (in z/Architecture mode).

Some support is contained in OS/390 V2R10 that allows you, for example, to use some of the new DCM-related display commands; however, the OS/390 V2R10 system itself cannot use a managed path.

DCM consists of code in both the IOS and WLM components of z/OS. You should review DCM's latest PTFs for both of these components.

Supported control units

The only control units supported by DCM are those used to attach the more recent DASD devices.

The IBM DASD control units that are supported by and have been successfully tested with DCM are:

- 2105 Enterprise Storage Subsystem with microcode level SC01208.
- 9393 RAMAC Virtual Array.

9032 microcode level

DCM supports all 9032 models. The following list shows the required microcode levels for each of the ESCON director models:

- 9032-2 – Version 4.1
- 9032-3 – LIC 04.03.00
- 9032-5 – LIC 05.04.00.

IMPLEMENTING DCM

Sample configuration

The sample configuration that will be used to implement IRD-DCM is shown below:

- A Z/900 server (type: 2064/serial number 9150).

CHPID	*Connected to ESCD #*	*ESCD port #*	*CHPID type*
34	02	96	Static
35	02	9E	Managed
36	02	82	Managed
37	02	87	Managed
04	04	49	Static
05	04	4A	Managed
06	04	34	Managed
07	04	35	Managed

Figure 1: Channel connections

- One 2105 control unit (CNTL 1200/Devices 1200-12FF).
- Two ESCON directors (ESCD 02 – CNTL/DEV: 9002 and ESCD 04 – CNTL/DEV: 9004).

Port	*Connected to ESCD #*	*ESCD port #*
0	02	C2
1	02	81
2	02	9D
3	02	9F
4	04	32
5	04	33
6	04	06
7	04	10

Figure 2: Control unit configuration

The Z/900 CHPIDs configuration

In order for DCM to be able to add or remove paths to a control unit, the control unit must be attached to a switch and that switch, in turn, must be attached to managed channels.

Eight channels are used to connect to the 2105 control unit. These channels are connected to two ESCON directors.

On each ESCON director, one channel connection will be static, the three others will be managed by IRD. This is illustrated in Figure 1.

The 2105 control unit configuration

The 2105 control unit has eight ports which are connected to the ESCON directors.

The 2105 control unit number is 1200 and it serves 256 DASD devices (1200-12FF). This is illustrated in Figure 2.

The ESCON directors' configuration

The ESCON directors' configuration is:

- ESCD 02 is a 9032 mod 3. Its device number is 9002 (CNTL 9002).
- ESCD 04 is a 9032 mod 5. Its device number is 9004 (CNTL 9004).

HCD definitions

In order to implement IRD-DCM, three sets of changes need to be made in HCD:

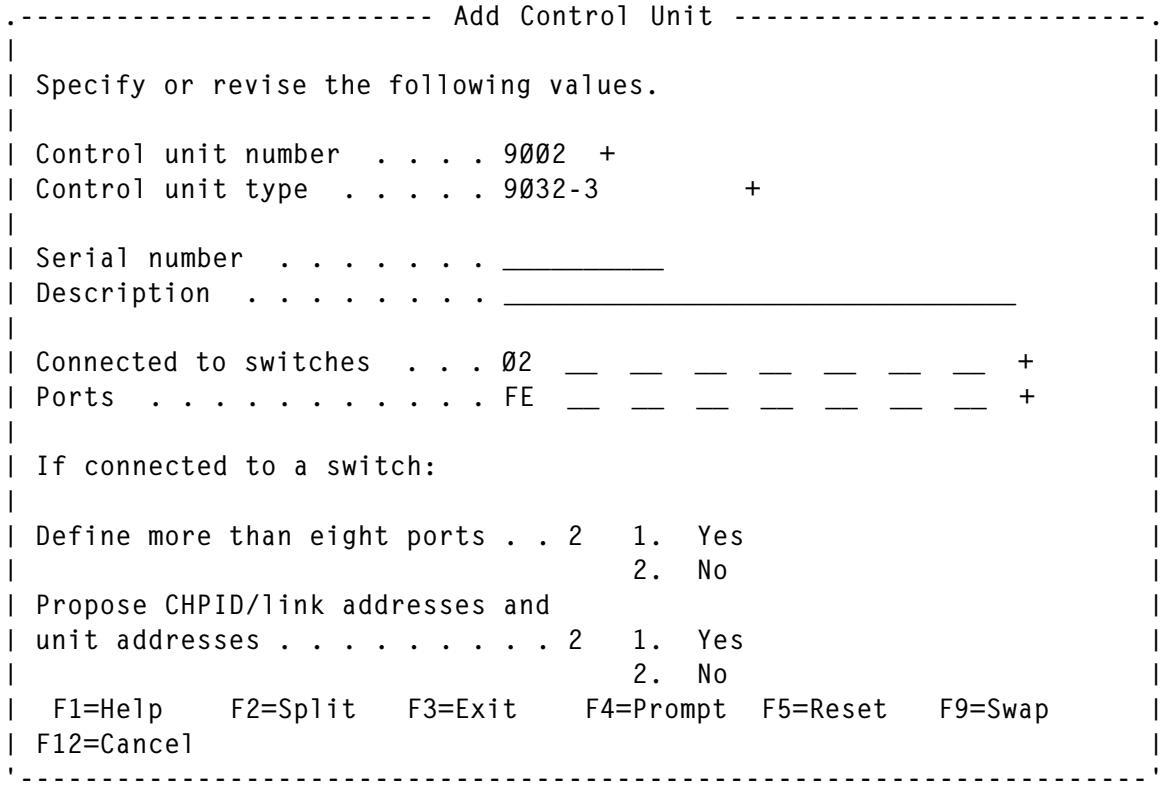
- Switch definitions.
- Managed channels definitions.
- Control unit definitions.

Switch definitions

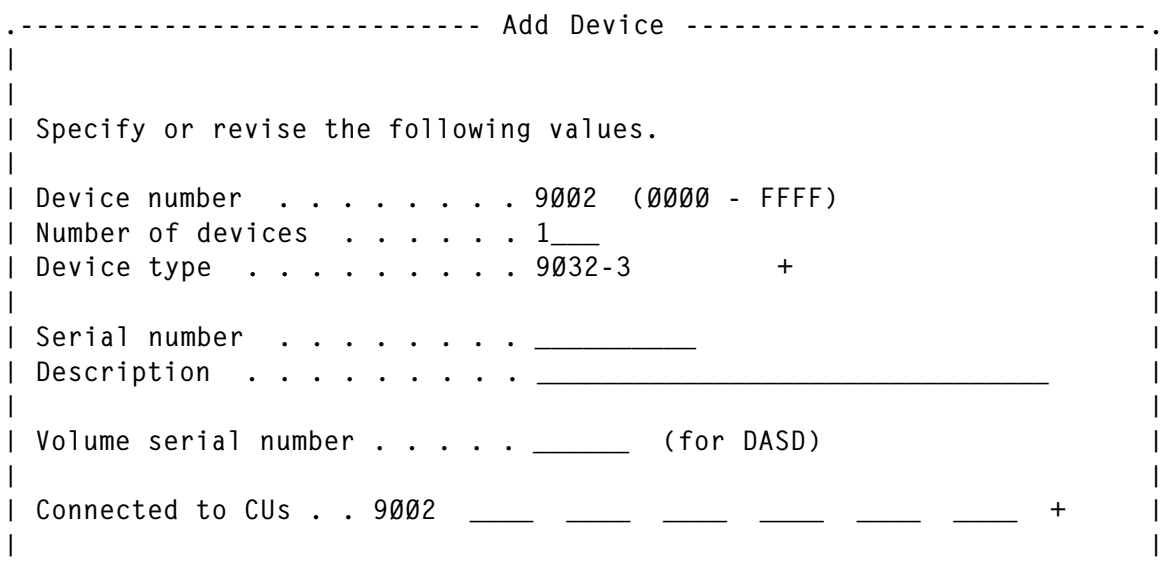
In order to allow communication between IRD and ESCON directors, switches have to be defined in HCD.

Because DCM uses the control unit port on the switch to get information about the configuration of the switch, and all the devices attached to it, you must define the control unit port in HCD.

First, you should define a 9032-x control unit:



And a 9032-x device:



```
|                                                                          |
|  F1=Help     F2=Split    F3=Exit     F4=Prompt   F5=Reset  F9=Swap     |
| F12=Cancel                                                               |
'--------------------------------------------------------------------------'
```

At this point, you can use HCD Option 2 (switches) to define your ESCON directors' configurations:

```
------------------------------------------------------------------------------
                           Switch List          Row 1 of 2 More:       >
 Command ===> ____________________________________________ Scroll ===> PAGE

 Select one or more switches, then press Enter. To add, use F11.

                                                                  CU   Dev
 / ID Type +        Ad Serial-# + Description                     Num. Num.
 P Ø2 9Ø32-3        __ __________ ______________________________ 9ØØ2 9ØØ2
 _ Ø4 9Ø32-5        __ __________ ______________________________ 9ØØ4 9ØØ4
 ************************** Bottom of data ***************************
```

Using options P (Work with ports), you can manage your switch attachments:

```
   Goto  Filter  Backup  Query  Help
 ------------------------------------------------------------------------------
                           Port List                        Row 1 of 153
 Command ===> ____________________________________________ Scroll ===> PAGE

 Select one or more ports, then press Enter.

 Switch ID  . . . . : Ø2   Address :

                                  --------------Connection--------------
 / Port H Name +                      Unit ID        Unit Type          O
 _ 8Ø   N ____________________________                                   _
 _ 81   Y ____________________________ CU 12ØØ           21Ø5             N
 _ 82   N ____________________________                                   _
 _ 83   N ____________________________                                   _
 _ 84   N ____________________________                                   _
 _ 85   N ____________________________                                   _
 _ 86   N ____________________________                                   _
 _ 87   Y ____________________________ PR CPCØC81    CHP 37 2Ø64-2C4    N
 _ 88   N ____________________________                                   _
…
```

Managed channel definitions

Managed channels must have the following characteristics in HCD:

- The channel cannot be in the path list for any defined control units. If you are converting an existing channel to become a managed channel, you must first remove it from the definition of any control units that it is currently attached to.
- The channel must be defined as *Shared*.
- The *Managed field* must be set to *YES*.
- The *I/O Cluster field* must be set to the *sysplex name* of the LPARs that can share this managed channel
- The switch characteristics (entry switch ID, entry port) of the channel connection must be defined.

The following panel shows the definition of the managed channel 35:

```
.------------------ Change Channel Path Definition --------------------.
|                                                                        |
|                                                                        |
| Specify or revise the following values.                                |
|                                                                        |
| Processor ID . . . . : CPC0C01                                         |
| Configuration mode . : LPAR                                            |
| Channel Subsystem ID :                                                 |
|                                                                        |
| Channel path ID  . . . . 35    +            PCHID . . . ___            |
| Channel path type  . . . CNC   +                                       |
| Operation mode . . . . . SHR   +                                       |
| Managed  . . . . . . . . Yes  (Yes or No)   I/O Cluster AXCF      +   |
| Description  . . . . . . . ____________________________________        |
|                                                                        |
| Specify the following values only if connected to a switch:            |
|                                                                        |
| Dynamic entry switch ID  02  + (00 - FF)                               |
| Entry switch ID  . . . . 02  +                                         |
| Entry port . . . . . . . 9E  +                                         |
|                                                                        |
|                                                                        |
'------------------------------------------------------------------------'
```

Control unit definitions

Having defined the managed channels, the next step is to define the control unit.

A managed control unit must have the following HCD characteristics:

- The control unit must support DCM.
- The control unit must be attached to a switch that is in turn attached to managed channels.
- The switch characteristics (switch ID, entry port) of the control unit interfaces must be defined.

```
.------------------------- Add Control Unit -------------------------.
|                                                                     |
|                                                                     |
| Specify or revise the following values.                             |
|                                                                     |
| Control unit number  . . . . 1200  +                                |
| Control unit type  . . . . . 2105          +                        |
|                                                                     |
| Serial number  . . . . . . . __________                             |
| Description  . . . . . . . . ____________________________           |
|                                                                     |
| Connected to switches  . . . 02  02  02  02  04  04  04  04  +     |
| Ports  . . . . . . . . . . . 81  9D  9F  C2  06  10  32  33  +     |
|                                                                     |
| If connected to a switch:                                           |
|                                                                     |
| Define more than eight ports . . 2   1.  Yes                       |
|                                      2.  No                         |
| Propose CHPID/link addresses and                                    |
| unit addresses . . . . . . . . . 2   1.  Yes                        |
|                                      2.  No                         |
|  F1=Help    F2=Split   F3=Exit    F4=Prompt  F5=Reset   F9=Swap     |
| F12=Cancel                                                          |
'---------------------------------------------------------------------'
```

On the second panel:

- In order to allow managed chpid usage, there must be at least one path that is specified with an asterisk (*) rather than a CHPID and Link Address. The control unit definition is:

```
                     Select Processor / CU     Row 1 of 2 More:      >
Command ===> ____________________________________________ Scroll ===> PAGE

Select processors to change CU/processor parameters, then press Enter.

Control unit number . . : 1200     Control unit type . . . : 2105
```

```
          --------------Channel Path ID . Link Address + --------------
 / Proc.CSSID 1------ 2------ 3------ 4------ 5------ 6------ 7------ 8-
 _ CPCØAØ1    Ø4.32   Ø5.33   34.C2   35.81   _______ _______ _______ __
 _ CPCØCØ1    Ø4.32   34.C2   *       *       *       *       *       *
 ***************************** Bottom of data ****************************
```

WLM definitions

There is no switch to turn DCM on or off.

So if you are running in basic mode, or the system that will be using DCM is in XCFLOCAL or monoplex mode, no WLM-related changes are required.

If the system that will be exploiting DCM will be running in an LPAR and is in multisystem sysplex mode, you must have a WLM structure defined. If the structure is not defined or is not available, DCM will not function in this environment.

You should modify your CFRM policy to define a structure named SYSZWLM _ssssttt – where *ssss* is the last four digits of the CPC serial number and *tttt* is the CPC type.

```
//STEP2Ø   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
     DATA TYPE(CFRM) REPORT(YES)
     DEFINE POLICY NAME(POLICYØ) REPLACE(YES)
…
        STRUCTURE NAME(SYSZWLM_91502064)    /* WLM IRD            */
               SIZE(ØØØØ8192)
               INITSIZE(ØØØØ6144)
               PREFLIST(CF1,CF2)
               REBUILDPERCENT(Ø1)
               ENFORCEORDER(YES)
/*
```

HMC definitions

There are no changes to the HMC specifically for DCM. However, in order for an LPAR to use the Dynamic I/O Reconfiguration capability, you must have enabled several functions in the CPC Reset Profile.

You must enable:

- The *Allow dynamic changes to the channel subsystem input/output (I/O) definition* option.
- The *automatic input/output (I/O) interface reset* option.

IOSTmmm module

Whenever Dynamic Channel path Management investigates adding or removing a path to an LCU, it tries to select the path with the best availability characteristics that delivers the required performance. It does this by comparing the points of failure of all potential paths with the points of failure of all existing paths to that LCU.

A control unit is characterized to MVS by a node descriptor, which can be displayed using the D M=SWITCH command:

```
D M=SWITCH(9004,32)
IEE174I 18.35.51 DISPLAY M 821
SWITCH 9004, PORT 32, DCM STATUS=ONLINE
ATTACHED NODE = 002105.F20.HTC.00.000000050031     - CU Node Descriptor
```

DCM uses information from the tag field of the node descriptor and combines this with information from a new load module (IOSTmmm) to identify common points of failure within the device.

The IOSTmmm load module is built using the IOSCUMOD macro, which:

- Identifies manufacturer (mmm).
- Identifies device type (for example 2105).
- Identifies model.
- Describes the meaning of four masks.

The IOST*mmm* load modules must reside in a LNKLST library.

The IOSTIBM module, the one that maps the IBM control units, is shipped in SYS1.LINKLIB as part of the system, and is updated as part of the changes provided with every new IBM control unit.

For non-IBM control units, you will need to obtain a copy of the IOSTmmm load module from each DASD vendor that you use.

To compile the IOST*mmm* load module:

```
//ASMCUMOD EXEC PGM=ASMA90,PARM=(NODECK,OBJECT)
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//         DD  DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD  DSN=&SYSUT1,SPACE=(1024,(120,120),,,ROUND),UNIT=SYSDA
//SYSLIN   DD  DSN=&OBJ,SPACE=(3040,(40,40),,,ROUND),
//             DISP=(MOD,PASS),UNIT=SYSDA,
//             DCB=(BLKSIZE=3040,LRECL=80,RECFM=FBS,BUFNO=1)
//SYSPUNCH DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
*********************************************************************
**  IOSTHTC - CONTROL UNIT MODEL TABLE FOR HITACHI PRODUCTS        **
**  THE FOLLOWING CONTROL UNITS ARE DEFINED                        **
**  ALL HTC 3990/6, 2105 EMULATION DEVICES.                        **
**  CHANGE ACTIVITY:                                               **
**    DATE  IDENT  DESCRIPTION                            AUTHOR   **
**  04/29/02 @HTC  ORIGINAL VERSION                       HTC/SEC  **
*********************************************************************
*
HTC        IOSCUMOD MANF=HTC,DEVT=,MODN=,MASK1=0010,MASK2=001E,
*
               MASK3=001F,MASK4=0000,DCM_SUPPORTED=NO
HTC399061  IOSCUMOD  MANF=HTC,DEVT=003990,MODN=006,MASK1=0010,MASK2=001E,
*
               MASK3=001F,MASK4=0000,DCM_SUPPORTED=NO
HTC399062  IOSCUMOD  MANF=HTC,DEVT=3990,MODN=006,MASK1=0010,MASK2=001E,
*
               MASK3=001F,MASK4=0000,DCM_SUPPORTED=NO
HTC21051   IOSCUMOD  MANF=HTC,DEVT=002105,MODN=,MASK1=0010,MASK2=001E,
*
               MASK3=001F,MASK4=0000,DCM_SUPPORTED=YES
HTC21052   IOSCUMOD  MANF=HTC,DEVT=2105,MODN=,MASK1=0010,MASK2=001E,
*
               MASK3=001F,MASK4=0000,DCM_SUPPORTED=YES
/*
//LINKIT   EXEC PGM=IEWL,COND=(5,LT,ASMCUMOD),
//             PARM='LET,LIST,NCAL,RENT,SCTR,XREF'
//SYSLMOD   DD  DSN=SYS1.LINKLIB,DISP=SHR
//SYSUT1    DD  DSN=&SYSUT1,SPACE=(1024,(120,120),,,ROUND),UNIT=SYSDA
//SYSPRINT DD  SYSOUT=*
//SYSLIN    DD  DSN=&OBJ,DISP=(OLD,DELETE)
//          DD  *
    NAME IOSTHTC(R)
/*
```

Updates to the IOSTmmm load modules can be activated dynamically using the SETIOS DCM,REFRESH command:

```
SETIOS DCM=REFRESH
IOS361I CONTROL UNIT MODEL TABLE REFRESH REQUEST COMPLETE
```

OPERATING DCM

Having completed all these changes, DCM is now ready to manage LCU.

You can use several operator commands to display information about the managed channels and control units.

In normal operation, there should be no need to interact with DCM. If the installation has defined managed channels and DASD subsystems with managed paths, DCM will automatically start managing the paths following an IPL. All actions that are taken by DCM are automatic and should be transparent, from an operator's point of view.

However, if you wish to display information about DCM or exert some control over the actions that it can take, you must be aware of some new commands and enhancements to existing commands.

DCM messages at IPL time

When the system is IPLed, you will receive messages from WLM informing you that it has connected to the WLM LPAR cluster structure. These messages will be followed by IOS351I and IOS356I messages indicating the status of DCM.

DCM messages at IPL time look like:

```
IOS351I DYNAMIC CHANNEL PATH MANAGEMENT ACTIVE
IOS356I DYNAMIC CHANNEL PATH MANAGEMENT NOT MANAGING ON SYSTEM = PROD
IWM050I STRUCTURE(SYSZWLM_91502064), CONNECTED
IWM061I WLM CPU MANAGEMENT AVAILABLE ON PROD
```

Display commands

D IOS command

The D IOS,DCM command shows the status of DCM:

```
D IOS,DCM
IOS353I 17.53.19 DCM STATUS Ø38
DYNAMIC CHANNEL PATH MANAGEMENT IS ACTIVE IN GOAL MODE
```

D WLM,IRD

The D WLM,IRD command provides the status of each of the three IRD functions (WLM CPU management, DCM, and channel subsystem I/O priority queueing) in the system on which the command is issued:

```
D WLM,IRD
IWMØ59I  16.49.29  WLM DISPLAY 198
  OPTIONS
    VARYCPU ENABLED:                    YES
    CPU MANAGEMENT ENABLED:             YES
    CHANNEL SUBSYSTEM PRIORITY ENABLED: YES
  WLM CPU MANAGEMENT STATUS
    CPU MANAGEMENT ACTIVE
  DCM STATUS
    DCM ACTIVE
  WLM LPAR CLUSTER DATA
    SYSPLEX NAME:               AXCF
    WLM LPAR CLUSTER STRUCTURE: SYSZWLM_91502064
    SYSTEM    PARTITION   MVS    CAPABILITY   CONNECT    CHANNEL
    NAME      IDENTIFIER  LEVEL  LEVEL        STATUS     SUBS.ID
    PROD      ØØ2         Ø15    Ø14          CONNECTED  ØØØ
```

D M=DEV command

The DISPLAY M=DEV command has been expanded to include an indicator of whether a given path is managed or not. It also lists the maximum number of managed paths that can be used by the control unit to which the device is attached:

```
D M=DEV(12ØØ)
IEE174I 14.41.39 DISPLAY M 529
DEVICE 12ØØ   STATUS=ONLINE
CHP                   Ø4   34   35
DEST LINK ADDRESS     32   C2   81
ENTRY LINK ADDRESS    49   96   9E
PATH ONLINE           Y    Y    Y
```

```
CHP PHYSICALLY ONLINE Y     Y     Y
PATH OPERATIONAL      Y     Y     Y
MANAGED               N     N     Y
MAXIMUM MANAGED CHPID(S) ALLOWED:  6
DESTINATION CU LOGICAL ADDRESS = ØØ
CU ND       = ØØ2105.ØØØ.HTC.14.ØØØØØØØ50ØØ31
DEVICE NED = ØØ2105.ØØØ.HTC.14.ØØØØØØØ50ØØ31
PAV BASE AND ALIASES  5
```

D M=CHP command

The DISPLAY M=CHP command has been updated to provide additional information related to DCM:

```
D M=CHP(Ø5)
IEE174I 18.41.13 DISPLAY M 322
CHPID Ø5:  TYPE=Ø4, DESC=MANAGED ESCON SWITCHED OR PT-PT, ONLINE
DEVICE STATUS FOR CHANNEL PATH Ø5
NO DEVICES ARE ACCESSIBLE THROUGH THIS CHANNEL PATH
SWITCH DEVICE NUMBER = 9ØØ4
************************ SYMBOL EXPLANATIONS ************************
+ ONLINE    @ PATH NOT VALIDATED   - OFFLINE    . DOES NOT EXIST
* PHYSICALLY ONLINE   $ PATH NOT OPERATIONAL
```

D M=SWITCH command

The command allows you to display information about the selected switch (indicated by specifying the switch device number), and the components that are attached to it (CPCs, control units, or other switches):

```
D M=SWITCH(9ØØ4)
IEE174I 18.26.56 DISPLAY M 74Ø
SWITCH 9ØØ4, PORT STATUS
     Ø  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 Ø   .  .  .  .  p  +  +  c  p  +  c  c  c  c  c  c
 1   +  c  c  +  x  c  p  +  c  c  p  c  x  c  c  c
 2   c  c  +  c  x  +  +  +  +  c  c  c  u  c  c  +
 3   c  c  +  +  c  c  c  u  c  c  +  +  c  c  c  u
 4   u  x  x  x  x  c  c  c  c  c  p  p  .  .  .  .
 5   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 6   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 7   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 8   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 9   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 A   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 B   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 C   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

```
 D   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 E   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
 F   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
***************** SYMBOL EXPLANATION *****************
 + DCM ALLOWED        - DCM NOT ALLOWED BY OPERATOR
 x NOT DCM ELIGIBLE   p DCM NOT ALLOWED DUE TO PORT STATE
 c CHANNEL ATTACHED   $ UNABLE TO DETERMINE CURRENT ATTACHMENT
 u NOT ATTACHED       . DOES NOT EXIST
```

The information provided in the output from the D M=SWITCH command is a combination of actual hardware information about the switch, discovered when DCM gets the node descriptors for all attached devices, and software information, which indicates whether DCM in this LPAR cluster can use the related port.

If you specify a port number on the display command (D M=SWITCH(ssss,pp)), the response includes the node descriptor of the device that is connected to the port, as well as the DCM status of the port:

```
D M=SWITCH(9004,32)
IEE174I 18.35.51 DISPLAY M 821
SWITCH 9004, PORT 32, DCM STATUS=ONLINE
ATTACHED NODE = 002105.F20.HTC.00.000000050031        - control unit
```

or:

```
D M=SWITCH(9004,4A)
IEE174I 18.42.05 DISPLAY M 474
SWITCH 9004, PORT 4A, DCM STATUS=CHANNEL ATTACHED
ATTACHED NODE = 002064.2C4.IBM.51.000000069150        - managed chpid
```

D XCF,STR command

The D XCF,STR command allows you to display information about the WLM structure used by IRD:

```
D XCF,STR,STRNAME=SYSZWLM_91502064
IXC360I  17.02.46  DISPLAY XCF 401
STRNAME: SYSZWLM_91502064
 STATUS: ALLOCATED
 POLICY INFORMATION:
  POLICY SIZE    : 8192 K
  POLICY INITSIZE: 6144 K
  POLICY MINSIZE : 0 K
  FULLTHRESHOLD  : 80
  ALLOWAUTOALT   : NO
  REBUILD PERCENT: 1
```

```
 DUPLEX          : DISABLED
 PREFERENCE LIST: ACF1   ACF2
 ENFORCEORDER    : YES
 EXCLUSION LIST IS EMPTY

ACTIVE STRUCTURE
----------------
 ALLOCATION TIME: 08/28/2003 21:13:41
 CFNAME          : ACF1
 COUPLING FACILITY: 002064.IBM.51.000000069150
                    PARTITION: 01   CPCID: 00
 ACTUAL SIZE     : 6144 K
 STORAGE INCREMENT SIZE: 256 K
 PHYSICAL VERSION: B9F09400 DE78A642
 LOGICAL  VERSION: B9F09400 DE78A642
 SYSTEM-MANAGED PROCESS LEVEL: 9
 DISPOSITION     : DELETE
 ACCESS TIME     : NOLIMIT
 MAX CONNECTIONS: 32
 # CONNECTIONS   : 1

CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
---------------- -- -------- -------- -------- ---- ----------------
#PROD            01 00010003 PROD     WLM      000B ACTIVE
```

Alter commands

V SWITCH command

The new VARY SWITCH command is used to tell DCM on all systems in the LPAR cluster whether it is allowed to use the specified port on the switch or not. It is only necessary to issue the command on one system in the cluster.

Specifying DCM=ONLINE tells DCM that it can use that switch port to set up a path between a managed channel and the attached control unit.

Specifying DCM=OFFLINE will stop DCM from setting up a managed path using this port. If it is already using this port for an existing managed path, the associated path will be varied off-line and removed from the configuration for the affected LCU:

```
V SWITCH(9004,32),DCM=OFFLINE
IEE633I SWITCH 9004, PORT 32, DCM STATUS=OFFLINE
ATTACHED NODE = 002105.F20.HTC.00.000000050031
```

SETIOS command.

The SETIOS command with the DCM= option can be used to turn DCM on or off.

DCM will automatically be started after the IPL.

If you wish to change the status of DCM after the IPL, use the SETIOS command. Don't forget that this command has an LPAR cluster-wide scope, so turning it on or off on one system will produce the same effect on all the other systems in the LPAR cluster:

```
SETIOS DCM=OFF
IOS358I DYNAMIC CHANNEL PATH MANAGEMENT HAS BEEN TURNED OFF
```

The effect of this command is to stop any further changes. It does not reset the configuration to its initial state:

```
SETIOS DCM=ON
IOS353I 17.55.54 DCM STATUS 139
DYNAMIC CHANNEL PATH MANAGEMENT IS ACTIVE IN GOAL MODE
```

VARY PATH command

Because DCM is solely responsible for the managed paths, it is not possible for the VARY PATH command to either vary on or vary off a managed path.

If you try to use this command with a managed path, the command will be rejected as in the following example:

```
VARY PATH(1200,35),OFFLINE
IEE777I VARY PATH REJECTED, CHPID 35 DEFINED AS MANAGED
```

CONFIG CHP command

To take a channel off-line, whether it is a managed or a non-managed channel, issue the CONFIG CHP(xx),OFFLINE command. It is not necessary to issue a V PATH or a V SWITCH command before taking the channel off-line.

Part of the CONFIG CHP processing is to take the path off-line, regardless of whether it is a managed or a non-managed path.

Monitoring USS performance from z/OS – an introduction

This article will focus on monitoring resource utilization and performance issues of Unix System Services (USS). The primary focus is on understand USS performance metrics as well as on monitoring and managing critical z/OS USS resources. The sample technique for collecting and analysing USS performance data will be demonstrated. Tuning recommendations will be briefly discussed too.

INTRODUCTION

For many years MVS (now called z/OS) has been the repository for business-critical applications and data while Unix played a fundamental role in the deployment of low-end server-based applications. This kind of application separation does not exist any more. Today's business environment requires multi-tiered applications that span multiple platforms, including the IBM eServer with z/OS native applications like DB2 and CICS as well as USS-based applications, such as WebSphere Application Server, Domino, ERP, and SAP. These workloads exploit the latest features of the z/OS runtime environment, including Parallel Sysplex, WLM and Unix System Services.

Thus, the merging of two platforms, especially two as disparate as MVS and Unix, is bound to cause confusion in those who have to monitor, maintain, debug, and write applications for the merged system. USS under z/OS poses a challenge to systems programmers in understanding the impact of USS workloads and resource consumption, especially since Unix concepts and commands can be unfamiliar territory for z/OS-oriented systems staff.

On the other hand, users generally have experience in either MVS or Unix and neither side is quite sure what to make of the other. The team who installs, manages, and monitors z/OS

workloads and e-business products must be equally comfortable in both environments. It is interesting to note that OMVS (Unix System Services) has been part of OS/390 for many releases. Many installations have been ignoring OMVS configuration even though the address space tries to start at IPL time. With OS/390 2.5, it became impossible to ignore because it is required in order to run TCP/IP and when applying software maintenance.

The challenges in using a z/OS system as a USS application server are the additional complexities associated with providing and maintaining acceptable performance levels for the new Unix applications, existing legacy applications, and the production batch processes.

THE MAIN FEATURES OF USS

The main feature of USS is that it is not a complete operating system: it is neither a virtual machine within z/OS nor an emulation, a subsystem, or an LPAR; it does not sit on top of z/OS. It is a set of system calls implemented directly into the z/OS system as USS services that perform functions that are similar to the functions other Unixes provide. In addition to that, USS does not have its own resources – memory, CPU, and DASD are shared with other z/OS work, and z/OS is used for all basic services such as CPU dispatching, virtual and real memory allocation, and paging. The resources USS uses are not pre-allocated from z/OS – USS dynamically expands and uses CPU, storage, and I/O capacity at will. This is a job done by the OMVS system address space (kernel), which provides USS services. The kernel is an integral part of the BCP element of z/OS; it sends instructions to the processor, schedules work, manages I/O, and tracks processes, open files, and shared memory, among other things. No work gets done in USS without involving the kernel.

Resource control and HFS file system statements, as well as system processing of USS, occur through a standard parmlib member (BPXPRM*xx*), which USS uses when it initializes during the IPL of a z/OS system. There are over 50 parameters that can be assigned in the BPXPRM*xx* file, each one demanding an

understanding of the resources being controlled. To dynamically reconfigure kernel services, you might use the SETOMVS or SET OMVS operator command. The SETOMVS command allows you to dynamically reconfigure one or more of the system characteristics. The SET OMVS command lets you specify one or more BPXPRMxx parmlib members to dynamically reconfigure the entire setting of kernel services. IBM recommends that you have two BPXPRM*xx* members – one that specifies system limits and processing and one that specifies HFS file system set-up. A detailed description of each parameter that controls the z/OS Unix System Services (z/OS Unix) environment and the file systems can be found in Chapter 9 of *MVS Initialization and Tuning Reference* (SA22-7592-02).

Before proceeding any further and diving into monitoring the performance of USS it might be helpful to review briefly some key concepts of USS.

In the world of the Unix program management model, a 'program' is either an executable program (called a 'binary' sometimes) or a shell script.

A 'process' is a program running one or more threads, along with the resources the threads are using (memory/storage, external files, control blocks and environment variables, and options).

There are three types of processes – user processes (associated with a program or shell user), daemon processes (which perform continuous or periodic system-wide functions), and kernel processes (which perform system-wide functions for the kernel, such as the init process or cleaning up zombie processes). A process maps 'roughly' into a z/OS address space; 'roughly' because, in certain circumstances, one can request that processes share an address space. The reason for this is that address space creation is expensive in z/OS (in terms of processing time and resources) so some performance benefits could be gained by sharing an address space among two or more processes.

Whether a process will try to create a new process in the current

address space is determined by the setting of the environment variable _BPX_SHAREAS. By setting the _BPX_SHAREAS environment variable, the parent process can control whether a spawn() call will result in a process being started in another address space or as a task within the same address space as the parent process itself. If _BPX_SHAREAS is set to YES before the spawn() call is initiated, the child process starts within the Fsame address space as the parent process and WLM is not involved in the creation of the new process. The _BPX_SHAREAS environment variable can be set to YES, NO, MUST, and REUSE.

There are some exceptions where, despite _BPX_SHAREAS=YES, a non-local spawn() (ie a child process starts in another address space) is done. The ability to create multiple processes in a single address space is probably unique to the z/OS versions of Unix. For each process a Task Control Block (TCB) and related control blocks are created. When an address space begins using z/OS USS services, we say the address space is a z/OS USS process. Many of the control blocks for Unix-type entities (directories, process groups, etc) are maintained in the Unix kernel address space. The following control blocks are maintained in the program's address space, off the secondary TCB (STCB): OTCB (Open systems Task Control Block, which holds UID and GID, and points to kernel services and lots of other places, including the THLI control block), THLI (THread Level Information, which holds flags and pointers to Unix thread-related information including the address of the PRLI control block), and PRLI (PRocess Level Information, which contains PID, flags, process return code, and other pointers).

A z/OS Unix program can create new processes and enable multi-threading within a process itself. A child process is created when the parent process issues a fork() system call. The creating process is called a parent process and the newly-created process is called a child process. A parent process can have many child processes, the number of which is controlled through the MAXPROCUSER statement contained in member BPXPRM*xx*.

To be a bit more precise, the MAXPROCUSER specifies the maximum number of processes that a single user (that is, with the same UID) is allowed to have concurrently active regardless of origin.

The MAXPROCSYS statement specifies the maximum number o f z/OS USS processes that the system allows. Most processes use an entire address space. A tuning tip: increasing the MAXPROCSYS value increases the usage of ECSA and ESQA storage on the system. The recommendation from IBM is to specify a higher value for MAXPROCSYS than your system environment can support. If you set the value too high, failures (EAGAIN) for fork or spawn might occur because WLM cannot provide enough fork initiators. However, there are no means to prohibit the creation of new processes by an application programmer. The maximum number of unique users (UIDs) that can use z/OS Unix services at a given time is controlled by the MAXUIDS parameter. This number includes all simultaneous TCP/IP users, as well as z/OS Unix users. FTP usage requires one user per connection, but TN3270 and a Web server do not.

z/OS USS assigns a unique identifier called the Process ID (PID) to the process. Each process also knows the ID of its parent through the Parent Process (PPID). All processes are related to each other through the PIDs and PPIDs. The originator of all processes is called the INIT process (PID = 1). Every process belongs to a process group, which also has a unique identifying number, the process group ID (PGID). A performance analyst will detect that processes are sharing the same address space and process grouping. A process group is a collection of address spaces.

A stream of instructions that is in control of a process is called a thread. A multi-threaded process begins with one stream of instructions (one thread) and may later create other instruction streams to perform tasks. To create a new thread, an application program will use the pthread_create() function. This function allows multiple tasks to run in a single process, thus allowing

concurrent and asynchronous processing without the additional overhead of creating a new address space. The z/OS Unix design for creating a thread is to attach a TCB within a process (address space). The pthread_create system call is used to create a thread and each thread is given its own thread ID.

The maximum number of threads initiated by pthread_create() that a single process can concurrently have active is controlled by the MAXTHREADS parameter.

Transparent to the application running in a USS environment is the fact that the HFS files it uses are treated internally by the system as a subsystem dataset, or, to put it another way, an HFS file is an object that exists within a mountable file system. An HFS dataset is a hierarchical file system that is used to store, and is essentially identified with, a mountable file system.

There are four types of file: regular file (consisting of text or binary), character special file (defining a terminal, a null file, and a descriptor file), special file (used to send data from one process to another, so that the receiving process can read the data), and pipe file. A pipe can be used to send data from one process to another. The pipes are like files where one process can write into a pipe and the other can read from the pipe.

Use MAXFILEPROC to set the maximum number of files that a single process can have open concurrently. This includes all open files, directories, sockets, pipes, etc. By limiting the number of open files a process can have, you limit the amount of system resources a single process can use at one time. The MAXFILEPROC setting affects storage in the OMVS address space. The BPXPRM*xx* parmlib member contains the parameters that control processing and the file system. The system uses these values when initializing the OMVS address space. The file systems are defined at OMVS initialization by FILESYSTYPE (defines the type of file system), ROOT (defines and mounts the root file system), MOUNT (defines the file systems to be mounted at initialization and where in the hierarchy they are to be mounted), and NETWORK (defines information needed for the socket physical file system to run).

ON-LINE MONITORING OF USS PERFORMANCE

As indicated earlier, USS address spaces can consist of several processes, which in turn might run one or more threads.

Each process is typically associated with a Unix command, consumes a certain amount of CPU, and provides state information. After having understood the current set of control parameters and the options that USS uses, the next step would be to find out what work is running in your USS environment.

An on-line USS performance monitoring tool should 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, and address spaces representing the USS workload. On the other hand, for problem determination, a performance analyst needs to find answers to the questions: what are the delayed processes? what is the status of each of the processes? or which processes are high CPU consumers?

When it comes to on-line performance monitoring tools for USS there are currently two options available within the standard IBM's toolkit. The first one is RMF Monitor III OPD report, while the second is SDSF's PS command.

RMF Monitor III OPD report

Currently, RMF Monitor III shows only the USS address spaces with their jobname and *using* or *delay* information collected in a similar way as for normal started tasks, batch jobs, or TSO users. The new Monitor III USS Process Data (OPD) report presents details about workloads running on z/OS under USS. A compact view with process state information and CPU consumption is provided in the base report, while a pop-up panel displays additional data (eg command name) on process granularity. The pop-up can be invoked via cursor-sensitivity (from all columns except Jobname, which goes to the JOB report, and PPID, which locates the parent process). The OPD report allows users to filter by jobname, by user, by process ID, or by address space ID to

give you the highest flexibility to reduce the data. In addition to the OPD report, one enhancement within RMF Monitor III is that the DELAY, PROC, and JOB reports have been enhanced to identify all address spaces that are associated with USS in some way. An 'O' is appended to the letter representing the job class, eg 'SO' is a started task having USS processes. Via cursor sensitivity, one can navigate between those reports and the new OPD report.

SDSF process display

The process display panel invoked by the PS command (new with z/OS R2) allows authorized users to display information about z/OS USS processes and provides the ability to cancel the process. The parameters of the PS command let you show all processes or just active processes (ALL|ACTIVE).

The process panel includes some or all of the following fields. The order and titles of these fields may be different depending on installation and user options. The PS display allows you to filter and arrange (by using FILTER and ARRANGE commands) by jobname, status, state, user, process ID, parent process ID, PID on which the process is waiting, command that created the process, time/date the process was started, number of active files, or by address space ID to give you the highest flexibility to reduce the data. The D OMVS,PID=pid command can be invoked by action character D, which displays thread-level information for any thread. You can use this command to display address space information for a user who has a process that is hung. You can also use the information returned from this command to determine whether an address space is using too many resources, and whether a user's process is waiting for a z/OS Unix kernel function to complete.

MXI PID command

A quick and simple way to display the USS process and thread information is available from MXI (MVS eXtended Information). MXI is free and available from Scott Enterprise Consultancy Ltd

(www.secltd.co.uk). The PID command provides a compact report with process-related information and CPU consumption, while a drill-down panel displays additional data on process granularity, including the unique features of displaying the last five syscalls of each thread as well as the full path name of the executing command.

The pop-up can be invoked via cursor-sensitivity from the Jobname and ASID columns. When selecting the ASID column, it will display address space-related information at the same time providing WLM's Service Class cursor-sensitivity field, which can be used to review its rule definition.

COLLECTING THE USS PERFORMANCE DATA

It is quite common to see a performance assessment being carried out only after we have seen applications experiencing performance problems or when analysts need to know whether there is enough capacity to support growth or new USS workloads. In addition to that, it has been noted that most of the performance tools available seem to relate to only the specific function they are measuring and do not offer a viewpoint of how well the total system is performing. Thus, it was deemed essential to establish a standard method for performance analysis of z/OS USS workload that would provide a consistent and effective process for monitoring and managing USS system resources.

As we have become familiar with the processes, threads, programs, commands, and cross-correlation of these to an address space, we are ready to proceed to the next step – that of choosing the types of data to collect. Information about z/OS Unix activity can be logged in SMF records if these records are specified in the SMFPRM*xx* parmlib member. In general, the following SMF records will have information about z/OS USS:

- Address space-level information is available in the common address space work record (type 30).
- HFS dataset cacheing information is available in the DFSMS statistics and configuration record (type 42, subtype 6).

- RMF support for USS is reflected in SMF record types 70 to 79.
- RACF processing record has information that can be used for security auditing (type 80).
- HFS file system activity data is available in SMF record type 92.

The single most important factor for any useful and meaningful USS performance analysis is a proper understanding of z/OS accounting of USS workload, a very detailed description of which can be found in Chapter 8, *OpenMVS Accounting of MVS System Management Facilities (SMF)* (SA22-7630-03). Based on recommendations gleaned from it as well as from descriptions of relevant SMF records and subtypes, a sample USS performance report writer was constructed in order to provide a starting point from which one can begin to gather performance information about the USS.

CODE

The code is a six-part stream (called USSDRIVE here). In the first step (DEL) auxiliary files are deleted while in the second step (SMFEXTR) the selected USS-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: the 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 that the SMF dump program (IFASMFDP) provides.

In the next step (COPYSMF) the extracted records are further filtered. The reasons for doing this are that the SMF dump program does not know anything about subtypes and, importantly,

we wanted only those records that reflect USS activity, ie the records with valid USS segments.

In the fourth step (REXX13) the relevant records are formatted by invoking a corresponding REXX EXEC. Three EXECs are invoked in this step – USSASID, USSFILE, and RMFUSS.

USSASID is the EXEC that handles the common address space work record (type 30) and it provides information related to USS process accounting data for the address space and special accounting for the EXEC family of functions. For every USS process that creates a new address space, SMF will create at least four type 30 records – job start, step termination, interval, and job termination.

If there are subprocesses that are started with forks or other USS constructs that build a new process (specifically processes that will use a new address space), you will get the same four records for that process. If you are running a shell script (any kind), every 'command' with the shell script will generate at least four type 30 records. Be careful here – if a lot of USS processes are running (such as a continuously running shell script), hundreds of thousands of type 30 records will be created in a 24-hour period. It should be possible to reduce the number of type 30 records by using the environment variable BPXSHAREAS=YES, which tells USS to make commands and forks subtasks of the parent task. The REUSE setting is treated as a YES, except some of the system structures built when the subtask is created are retained and used again during the next fork/spawn. Both the YES and REUSE 'hide' the identity of child processes, while accumulating resource usage into the SMF records. One will also probably see step completion messages in batch job output with *OMVSEX, which reflect a new USS process initiated in the same address space as the original batch job.

The next EXEC invoked in this step, USSFILE, is the EXEC that handles the USS file system activity record type 92 subtype 11 (close of a file).

All file system activity records contain information needed to

associate them with a particular job and jobstep. These fields can be used to link file system records with related common address space records (type 30), which, in fact, was done in the last step of this stream by MATCH EXEC.

Also included is information needed to associate them with a particular user, when appropriate.

The last EXEC in this step, RMFUSS, handles the SMF 74.3 and 74.6 records. SMF record 74 subtype 3 deals with USS kernel activity while subtype 6 is a repository for HFS global activity, buffer pool statistics, and HFS file system statistics. These subtypes are generated only if the OPD parameter of the RMF Mon III gatherer is set on and HFS dataset names are included therein. This EXEC may seem to be redundant since there is a postprocessor's OMVS kernel activity report.

The main reason for writing this EXEC is that the OMVS kernel activity report is interval based and each one of the reports it produces is in fact a collection of five reports, each reporting on resources 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. In contrast, the RMFUSS EXEC is resource oriented – each resource is separately reported on, thus making it easier to notice the peaks and overruns. Also, each report contains the counted totals per interval, which is the information missing in the postprocessor's report. This is not meant to be a replacement for the postprocessor's kernel activity report but rather a supplement to it. Once we have spotted a peak value, an exception, or an overrun, we can turn to the postprocessor's report and analyse all of the interval report. The very same logic is behind the processing of SMF 74.6 records, which pertain to three interval statistics – HFS global statistics, buffer pool statistics, and HFS file system statistics.

The fifth step (MTOOL) is an auxiliary step that prepares the selected portions of address space records (type 30 subtype 4) and close file records (type 92 subtype 11) for the match/merge EXEC invoked in the last step (REXX4).

A set of 16 reports is produced by this stream, each providing in-depth information on certain aspects or domains of USS performance. Each report consists of two sets of variables. The first set is a fixed one consisting of the variables that uniquely identify the process being monitored. This set is meant to be used across all common address space work reports. The pool of variables in this set contains the generated observation number, process date and time stamp, user id, job id, job name, step name, process (PID), and the name of the program that was run (an MVS load module or HFS executable filename). The second set of printed variables is area specific and pertains only to the performance domain being monitored.

The processor report is an overview of various processor timings (total, TCB, SRB, I/O, and RCT) for hiperspace transfers, pre-emptable SRBs, and enclaves (independent/dependent). This report can be used to determine which processes are the heaviest consumers of CPU. There are several means to control CPU usage within an instance of USS: the MAXPROCSYS parameter can be used to limit the total number of processes within USS; MAXPROCUSER is used to limit the number of concurrent processes that any given user can spawn; and MAXCPUTIME parameter is useful when we want to control the amount of CPU that any given application requires to complete its processing.

An additional note should be made on enclaves: when an application issues a fork or spawn, it is desirable for the performance characteristics of the application to be propagated to the forked or spawned process. This is achieved through the use of enclaves. Some code, which runs in the forked (or spawned) address space before the application code receives control, is usually called set-up code. The service consumed by this set-up code is not charged to the enclave that the application is running in. The service consumed by the set-up code is charged under the OMVS subsystem according to the classification rules you set up. The set-up code counts as a transaction in the service class it is associated with (under the OMVS subsystem), as does the original application transaction

(under the subsystem of the original transaction, for example IWEB). This results in a count of two transactions for one business unit of work. However, so that the service units consumed are not double-counted, those used by the application are charged to the enclave, and those consumed by the set-up code are charged to the OMVS subsystem. The new thread will be joined to the enclave of the caller of pthread_create.

The I/O activity report is a detailed report that shows calculated delays (dataset delay and allocation delay), the count of blocks transferred, DASD I/O connect time, and enclave DASD I/O timing for both independent and dependent ones, including connect, disconnect, and pending time.

The process I/O report provides a summary of the I/O activity done using USS. I/O activity is summarized in the categories directory reads, read and write activity to regular files (this includes all I/O to HFS files), read and write activity to pipe files (although there is no physical I/O associated with pipes this activity is still listed as I/O – this field contains I/O activity to pipe files and to Unix domain sockets), read and write activity to special files (this includes I/O activity to terminals), and read and write activity to network sockets (this field includes I/O activity to network sockets as returned by the TCP/IP physical file system). Path look calls and path generated calls are also included in this report, as well as message queues' byte counts for send and receive. Finally, there is also a count of total sync() calls and the number of USS syscalls requested by each process and the corresponding CPU timing (in seconds) accumulated by these requests.

The CPU timing is not included in the total CPU seconds accumulated for the address space and its value is limited since there is no way of knowing which syscalls are being issued by the processes, how often each is being issued, or how much CPU time each syscall type is consuming. The value, however, could be monitored to help monitor changes to the workload.

A tuning hint: the information related to directory look-ups is quite

useful because high values normally indicate that the accessed files are close to the bottom in the file system hierarchy. To reduce look-ups there are two options – move these files towards the top of the hierarchy (as close to the root directory as possible, keeping in mind that HFS locks all the paths to the file) or change applications to force a change directory before opening the files.

The storage and paging report is a summary of paging/swapping activity and storage mapping of the process. The maximum region size (in bytes) for an address space hosting process/ processes is under the control of the MAXASSIZE parameter. In the context of address space reporting, it applies to a daemon process that forks another, and a daemon that was not manually started via TSO/E or the OMVS shell. Note that the IEFUSI limit for the OMVS work region size overrides MAXASSIZE. Users are strongly discouraged from altering the region size of address spaces in the OMVS subsystem category. Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXASSIZE.

The performance report is a detailed report on service units consumed by each process (by CPU, SRB, I/O, and MSO), transaction timings (how long a transaction was active and how long it was real storage resident), and WLM-related information (workload, service class, resource group, and reporting class). Remember that the most important tool we have with which to tune USS performance from the z/OS side is a proper WLM set-up. Because we normally do it for any z/OS workloads we have also to check service policy to ensure that proper service classes and objectives are specified to manage the USS and the increase in the number of address spaces it uses.

Based on established business requirements, workloads that act as servers should be at a higher dispatching priority and importance than other applications. The USS system address spaces we have to take care of include:

- OMVS – the kernel – the default service class is SYSTEM, no need to change it.

- BPXOINIT – the first parent (process id = 1) – it starts the daemons, inherits orphaned children allowing for a correct clean up of their processes, and it is the parent of the processes inside an existing address space (not created by fork or spawn). WLM classification should be to SYSSTC service class or high-velocity goal. It is quite useful to place OMVS and BPXOINIT in their own report classes.
- BPXAS – a pool of address spaces WLM maintains in order to avoid the delay of address space creation during a fork operation. BPXAS initiators are started by WLM only on demand and are similar in concept to JES2 initiators. It is not advisable to terminate the BPXAS address spaces when a user finishes with them. There is a very high overhead in creating address spaces. BPXAS remains for the next request in order to speed up processing. Because of the nature of Unix, thousands of processes can be started in a matter of seconds, for example a make command against a large set of programs could cause this. Thus, one should be careful here: too many extra address spaces in the pool wastes CPU and storage while too few forces a new address space to be created during a fork(), thus slowing down fork response time. The default service class is SYSSTC and there is no need to change it. A good thing would be to place it in its own report class to monitor average spaces in the pool.
- SYSBMAS is the DFSMS buffer manager that is classified in the default STC service class – change it to SYSSTC.

There are also two performance flags available, and these should be taken care of if the values are not zero. The first one provides insight into address space initiation/restart status and service class (if CPU-critical or storage-critical). The second one is address space related and warns us that there might be some WLM performance issue such as an address space being designated as storage-critical, an address space currently CPU-protected/storage-protected, or an address space that cannot be managed to meet transaction goals.

The summary report is the last one in the group of reports produced by the USSASID EXEC, providing a summary of the processes by areas covered by the reports described above. It was tailored after IBM's job/step termination exit routine (IEEACTRT) and is probably the best place to start when monitoring USS performance. Once an exception or a peak value has been spotted in any key performance value, one can use generated observation identification to drill down across domain-relevant reports.

The USS file system read and write activity report is a product of USSFILE EXEC.

When an HFS file system is mounted, SMF begins collecting accounting data for the file system. When an HFS file is opened, SMF begins collecting accounting data for the file. Partial SMF accounting does not occur – either all the information for a file system or open file is collected, or none is collected.

There are several subtypes of SMF record type 92. Subtype 11 (written when a file is closed) is the most important subtype for performance data.

The report shows how hard a file system (identified by file serial number and pathname used at open time, if known) is being hit by number of calls (read/write), count of I/O blocks (directory, read/write), and number of bytes (read /write). Also included in the report is data on the time the file was opened as well as when it was closed. The calculated difference between these two timestamps tells us how long the file was open. Note that the file serial number changes every time the file is reallocated. Included in the report is information necessary to associate file activity with a particular job and jobstep (jobname, reader start time and date, step name, process ID). In general, it is best not to create these records unless one is trying to diagnose an I/O performance issue or determine the reason for a certain activity. A warning: when enabled, SMF can generate a huge volume of SMF92 data; use the alternative SMFPRM*xx* setting to avoid it.

The composite I/O activity report uses SMF type 30 records to

create a table displaying program name, system resource and file activity.

The SMF process ID (SMF30OPI) field was used with SMF type 92 records to identify HFS dataset names.

The following two sets of reports (eight in total) are a product of RMFUSS EXEC (discussed above): the first one is HFS dataset I/O related (buffer pool statistics, HFS global statistics, and HFS file system statistics) while the second one is a kernel activity report set consisting of five separate reports (USS Inter Process Communication, USS shared library region and queued signals, USS memory map, USS process activity, and USS system call activity).

At the beginning of this article we said that no work gets done in USS without involving the kernel. Thus, monitoring and tuning the kernel is of prime importance and should be the focus of a performance analyst's attention. Nonetheless, it should be observed that if your z/OS system is currently experiencing performance problems, you can be assured that USS will not perform well. The system must have CPU cycles, storage, and I/O capacity available before an attempt is made to fully exploit the USS-based applications. For an overall heath-check of the USS kernel the RMFUSS-produced set of reports has been provided, and these should be carefully examined. These reports provide feedback for installation-specified limits and actual activity for system calls, process activity, inter-process communication, memory map and shared library region, and queued signals. It can be used for verification of parmlib settings as well as for tracking and planning the resource utilization in terms of MPL, CPU, and storage.

The system call activity report provides the same information as included in the SMF 30 record, but aggregated at the system level. The count can be turned on by BPXPRM*xx* parameter SYSCALL_COUNTS(YES/NO). The data presented reports on the total, minimum, average, and maximum number of system calls during an interval and the corresponding total, minimum, average, and maximum CPU time consumed for system calls. It

is good to monitor changes in these numbers, especially in a before/after situation when introducing a new USS application because certain system calls can take up significant CPU time. It not possible at this stage to identify and measure what type of system calls do consume the CPU. (The measurement unit for syscalls is number per second, and for CPU timing it is number per hundredth of a second.)

PROCESS ACTIVITY REPORT

The BPXPRM*xx* parameters corresponding to the defined maximum values as reported under the heading PARM VALUE for *Processes*, *Users*, and *Proc/User* are MAXPROCSYS (maximum processes allowed system wide), MAXUIDS (maximum number of concurrent Unix users), and MAXPROCUSER (maximum number of processes allowed per user). To set initial values for these parameters the following advice might be of some help: estimate the number of users concurrently accessing the system; add 30% and set the value for MAXUIDS; multiply MAXUIDS by 4; add a sufficient number of processes for daemons (5–15); and set the value for MAXPROCSYS. If you don't run applications with specific requirements, set MAXPROCUSER to 25 (default).

The count of *Processes* is the number of Unix processes controlled by USS during this interval (total, minimum, average, and maximum). The count of *Users* is the number of Unix users controlled by USS during this interval (total, minimum, average, and maximum). Monitor the 'AVERAGE' used values to help determine whether the value is set too high (waste of resources).

There is also an 'OVERRUNS' part to the report showing you how many times you had to wait because a specified number was too small. For *Processes*, you get the number of processes that could not be created by USS because the maximum number of processes would have been exceeded; for *Users*, the number of Unix users that could not be created by USS because the maximum number of users would have been exceeded; for *Proc/User*, the number of processes per user that could not be created

by USS because the maximum number of processes per user would have been exceeded. Monitor the 'OVERRUNS' to determine whether you are experiencing slowdowns because a value is set too low. (The measurement unit for overruns is number per second.)

On the other hand, if you get the message, 'FSUM7726 cannot fork: EDC5112I resource temporarily unavailable', the parent process has completed (gone away), leaving the child process abandoned. These child processes cannot terminate and clean up; they are hanging around in a zombie state. Issue a D OMVS,A=ALL operator command to see whether there are any zombies. Zombies will be address spaces flagged with a Z. If no zombies, the specified limits have been exhausted. Issue a D OMVS,O command, which shows the settings for MAXPROCUSER and MAXPROCSYS. Increasing MAXPROCSYS or MAXPROCUSER will fix this.

Sometimes, when a problem has occurred, there may be many zombie processes left around. Each zombie takes up a process slot. If zombies exist something is wrong because zombie processes are not getting cleaned up. This can happen when an incorrectly-coded application does not do wait() calls to clean up child processes or is otherwise functioning incorrectly. If there are many zombies and the parent pid for them is 1, it usually indicates a hang in the PID=1 BPXOINIT process, which is responsible for zombie clean up. Issue D GRS,C to see whether there is a latch hang on BPXPRIT latch #1. If PID(1) is the parent of the zombies, it is likely that the system is having a problem. You may need to cancel the process that is stuck holding this latch. Terminating the parent with a SIGNAL or CANCEL should clean up the problem. Issue a D GRS,C command. This will show whether BPXOINT is stuck waiting for the process latch for PID(1). If it is waiting, then the problem is the owner of that latch. Cancelling the owner of the latch should free up the zombies once BPXOINIT gets control.

Editor's note: this article will be concluded next month.

Mile Pekic
Systems Programmer (Serbia and Montenegro)

BMC Software has announced the availability of CONTROL-M for File Transfer, which delivers unified file transfer management. The product manages the entire file transfer process, providing a graphical interface and real-time progress reporting to ensure successful transfers of files. This automation reduces the risk of transmission delays, data corruption, and poor security.

CONTROL-M for File Transfer allows users to define file transfers through a GUI. It features capabilities for monitoring and managing files in transfer, allowing full control of the file transfer during execution and real-time reporting on the transfer's exact status using various parameters including the amount of data transferred, transfer rate, and the transfer's predicted completion time.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042-2827, USA.
Tel: (713) 918 8800.
URL: http://www.bmc.com.

* * *

Computer Associates has announced general availability of Advantage CA-IDMS R16 for z/OS, a Web-enabled database platform that handles high-volume processing requirements, including online transaction processing, batch job activity, and transactions from diverse distributed-platform applications.

R16 delivers: z/Architecture 64-bit processing that utilizes virtual storage to support massive memory cacheing, reducing real I/O and improving performance; Parallel Access Volume exploitation that allows multiple jobs to simultaneously access the same logical volume and thereby allows much higher I/O rates for increased throughput and reduced response times; full, two-phase commit compatibility between multiple Advantage CA-IDMS/DB systems, DB2, WebSphere MQ and other resource managers, providing fully automatic resynchronization if processing is interrupted during the commit operation; plus other features.

For further information contact:
Computer Associates, One CA Plaza, Islandia, CA 11749, USA.
Tel: (631) 342 5224.
URL: http://www3.ca.com/Solutions/ProductFamily.asp?ID=126.

* * *

Compuware has announced its intention to provide Compuware STROBE support for IBM's zSeries Application Assist Processor (zAAP), planned to be available on the IBM eServer z990 and zSeries 890 servers.

Specifically, Compuware STROBE will provide application performance analysis for Java applications utilizing the new IBM zAAP execution environment on the zSeries platform.

For further information contact:
Compuware Corporation, One Campus Martius, Detroit, MI 48226, USA.
Tel: (313) 227 7300.
URL: http://www.compuware.com/products/strobe/default.htm.

* * *