SHARE

SHARE SESSION REPORT

SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE
61	M512/B189	User Experiences with VSCR	520

PROJECT	SESSION CHAIRMAN	INST. CODE
CME/MVSP	Robert H. Johnson	BCG

Boeing Computer Services, 7990 Gallows Court, Vienna, VA 22180

SESSION CHAIRMAN'S COMPANY, ADDRESS, AND PHONE NUMBER

The following presenters gave their solutions to virtual storage constraint relief (VSCR):

Robert Johnson (BCG)
Clinton A. Shaw (JHC)
Clark Morris (WET)
Steve Nathan (BTP)

All of these solutions were done in MVS/370 environments in attempts to survive to MVS/XA. MVS/XA will provide temporary relief but these techniques will be needed for the foreseeable future.

3/K/rcg/1

Virtual Storage Constraint Relief (VSCR)

Bob Johnson

Boeing Computer Services
7990 Gallows Court
Mail stop CV-55
Vienna, Virginia, 22180
(703) 821-6140

Installation Code: BCG

MVS Performance / Computer Management & Evaluation Projects

B190/M512
Thursday 8/25/82 2:45 P.M.
SHARE 61
New York, N.Y.

ABSTRACT

This presentation is based on the virtual storage problems and the solutions that were implemented at BCS. Many of these ideas were created and sent to me by SHARE attendees at SHARE 60. Thank you for your assistance. The presentation covers how we got constrained, the symptoms of constraint, how we removed the constraints and how we plan to manage virtual and real storage in the future.

CONTENTS

VIRTUAL STORAGE CONSTRAINT RELIEF (VSCR) USER EXPERIENCES

1.1 *INTRODUCTION*

1.1.1 *What is Virtual Storage Constraint (VSC)*

MVS/370 or MVS SP 1.3 has a virtual storage limit of 16 megabytes. Virtual storage constraint exists when your installation:

1. Can not add a new function.
2. Spends systems programmer time to manage virtual storage by moving modules from one library to another or changing members of SYS1.PARMLIB.
3. De-tunes the system by moving modules from pageable or fixed areas of virtual storage to SYS1.LINKLIB where the module is loaded each time it is used.
4. Can no longer run SMP.
5. Finds JES2 problems such as x0A abends (out of region) or can not add remotes.

1.1.2 *How we got there*

In the beginning we tuned SYSRES. We moved command modules from SYS1.CMDLIB to SYS1.LPALIB. We added neat stuff from IBM: VTAM, IMS, SPF, ISPF/PDF. We added neat stuff from other vendors: IDMS, INQUIRE, UCC1-999, etc., etc.

Maintenance costs. My rule of thumb is that each years maintenance costs 50-100K of virtual storage. This is a combination of new code and new/larger control blocks.

1.2 *VSC SYMPTOMS: ARE YOU THERE?*

1.2.1 *One if by land...*

How do you know if VSC is coming? We found several indications. Most of them appeared shortly after maintenance or new function was added. JES2 would not initialize or would initialize and then abend with various x0A abends. JES2 issues GETMAIN macros during normal processing. These abends indicate JES2 was out of V=V storage.

Another symptom was that operator commands would not return information.

A good rule of thumb is to look at the size of PLPA. *If PLPA is over 4.3 to* 5 megabytes you probably have a virtual storage *problem.*

1.2.2 *Tools, tools and more tools.*

OMEGAMON (Candle Corporation) has an outstanding display of virtual and real storage.

IEHLIST can be used to list the LPA and scan for module names that do not conform to IBM conventions. Look for SVC modules that have two CSECTs. This may mean that a new version of the SVC was added with a different CSECT name. One is used and one takes up space.

Virtual Storage Monitor 5798-DEL. This requires you to install in an APF library and track virtual storage with an SMF record. Watch out as it requires linking in RMF code and you may back level modules.

ACF/TCAM operator command **D TP,POOL,ALL** will display TCAM virtual storage usage. If you have your performance statistics and TSOMON in a MICS data base you can track the minimum TCAM buffers over a period of time. Keep 50 or whatever your IKJPRMxx members states is the minimum and you can remove the rest.

VTAM operator command **D NET,BFRUSE** will list the ACF/VTAM buffer statistics. Check during peak time and decrease the number allocated until you start to get buffer pool expansion. This is better than giving ACV/VTAM extra

buffers to prevent expansion. Pick an IOBUF size to make the best use of MVS page size (4K). The number of IOBUFs per page is different between Ver 1 Rel 2 and Rel 3. Chapter 12 of ACF/VTAM planning and installation has a table listing buffers per page for various bufsize values.

Virtual Storage Assessment Methodology (ZZ20-4151) is a very good tutorial on virtual storage and contains most of the information needed for a Roll-your-own monitor.

AMDPRDMP MVS Service Aid with two exits from Field Engineering Division that allow you to specify "ALLOCCSA" and "CSAMAP". Ask your IBM SE or PSR to search RETAIN.

MVS LPA OPTIMIZER 5798-DKP will help you build and maintain a pack list or will help you identify modules that can be moved from SYS1.LPALIB to SYS1.LINKLIB.

MVS BLDL Optimizer 5798-DKQ will generate a new BLDL list and create the IEBCOPY/SMP cards to move modules from LPALIB to LINKLIB.

1.3 *HOW DO YOU SPELL RELIEF?*

1.3.1 *Introduction - get a map*

It is important to get a benchmark of where you are at the time you begin. Draw out the Virtual storage of your MVS/370 system. This is important because if you do not, you can not quantify your good works and write a SHARE paper!

The following is a list of actions that I believe will reduce virtual storage usage. Where I have been successful at reducing virtual storage, I will note the amount of virtual storage in parenthesis (i.e. (xK)).

1.3.2 *PARMLIB manipulation*

PARMLIB is very easy to manipulate so it is usually the first to be tried. One short term solution was to reduce the number of JES2 buffers (&NUMBUF). The 4008 byte buffers could be sacrificed for a quick fix. JES2 issues message $HASP050 when he is out of buffers. This was notification that we have squeezed JES as hard as we should.

We reduced SQA and let it overflow into CSA. (192K)

We reduced the number of TCAM TIOC buffers SYS1.PARMLIB(IKJPRMxx) TIOC BUFFERS=nnn BUFSIZE=nnn. Be sure you have enough space to allow full screen SPF edit to write a full screen of data to the terminal. (50K)

Remove IEAPAK00. Some claim 450K savings but we were only able to get 4K. I made IEAPAK00 smaller but retained the member to minimize paging.

Reorder MLPA by descending module size. There is no packing done for MLPA module. By loading the largest module first, we minimize the holes. (20K)

The loading of the fixed LPA not only does not try to pack the modules, but does not even keep track of the holes created by module sizes. The fixed LPA is located in the nucleus and the nucleus is rounded up to a 64K boundary. Careful packing may not create any virtual savings.

1.3.3 *System usage of storage*

1.3.3.1 Follow instructions

Run the MVS SP 1.3.0 job (see cover letter) to remove unused modules from the system after bringing your system up to MVS SP 1.3.0. (88K)

1.3.3.2 TCAM

TCAM can be re-gened to remove unused line groups.

1.3.3.3 MVS Control blocks

Remove unneeded devices from your I/O gen. Extra 2305-2 drums (1408 bytes) and extra strings of DASD (1024 bytes for 32 devices) can waste real and virtual storage. Nucleus pages are fixed real storage.

Stop unneeded initiators (2K per initiator)

1.3.3.4 MVS paging supervisor

Reduce the size of each page data set. The first reason is that there is a bit for each page slot allocated. The second reason is that the paging supervisor wanders all over the device to allocate pages. Smaller page data sets (65 cylinders) work best.

1.3.3.5 JES2

Be sure that your JES2 &BUFSIZE is set to 4008 bytes for all versions of JES2 and on all types of DASD. Be sure the SJB size is less than or equal to 1024 if you are running JES2 at a version below MVS SP 1.3.3.

1.3.3.6 VTAM

If you do not have someone watching VTAM CSA usage all the time you are committing suicide.

1.3.3.7 RACF

RACF can have unused in-core index blocks. Reorganization of the RACF data sets gained us 8K.

1.3.3.8 IMS 1.2

The use of Local Storage Option (LSO) and Cross Memory LSO (XLSO) greatly reduces the use of CSA. On non 3081 or 3033 without the Cross memory emulation, you pay a price of a program check every cross memory request. The emulation is done in the Program Check First Level Interrupt Handler (PC FLIH). (256-400K)

1.3.3.9 CICS

To reduce high paging rate within CICS, change the following instruction in DFHSCP:

```
BAL SCLINK,SCBESTFT       18700000
to:
BAL SCLINK,SCFRSTFT       18700000
```

This will cause CICS to use a first fit rather than best fit for shared and control subpools.

All CICS/VS COBOL application programs should be coded with RES,NODYNAM option. This will allow sharing code of ILBxxxxx modules (savings of 2K per program).

Alternate Index files (AIX) in CICS can be allowed to use the LSR pool by NOPing the following instructions in module DFHSIE1:

```
BE  VFCTNSRR              26400000
BE  VFCTNSRR              30560000
```

Make sure enough strings are specified in the LSR pool via the SHRCTL macro in the FCT.

Users of CICS and VTAM only can specify TCP=NO in the System Initialization Table (savings of 60K).

1.3.3.10 VSAM

Keep the number of VSAM catalogs to a minimum. Each open VSAM user catalog requires 20K of CSA. Recoverable catalogs

require an additional 8.5 to 10K of CSA. Convert completely to DF/EF VSAM catalogs. This should be done as quickly as possible. While you are running both VSAM and DF/EF catalogs you will have several hundred thousand bytes of modules duplicated in PLPA.

1.3.4 *PTFs and APARs*

This is the hard part. It is one thing to get a PTF or APAR number to fix a particular problem. It is an entirely different thing to follow all the PRE and REQ and PE chains to gather all the things needed to apply the one PTF. The webs we weave...

1. **AZ64868** Reduce storage shortages, bypass steal of pages (and growth of SQA) if there are not PCCW's (and therefore PCB's)
2. **UZ60519** Fix logical swap to pre MVS SP 1.3 days.
3. **UZ62648 plus OZ69158** IDA0192M has problems with page boundaries.
4. **OZ57760** JES2 C/I storage
5. **OZ67913** JES2 C/I storage
6. **OZ73256** JES2 users hung in wait until next IPL if you cancel JES2 and restart JES2.
7. **AZ56154** JES2 out of buffer problems: recovery
8. **AZ65152** JES2 out of buffer problems: prevention
9. **AZ66397** JES2 out of buffer problems: prevention
10. **AZ69283** JES2 out of buffer problems: recovery

1.3.5 *Houseclean LPALIB*

Modules in SYS1.LPALIB that are included in SYS1.PARMLIB members IEALPAxx and IEAFIXxx will have two copies in storage. Remove these from SYS1.LPALIB and place them in SYS1.LINKLIB. Then use IEALPAxx or IEAFIXxx to get them into storage. Do not forget to use SMP UCLIN/LMOD statement to tell SMP where to do maintenance.

1.3.6 *Time to get mad*

1.3.6.1 I have seen the Enemy and it is US

Use VMAP or OMEGAMON to get a map of virtual storage. Use IEHLIST to get a list of SYS1.LPALIB. Then scan the listing. Remember if your PLPA is larger than 4.3-5 megabytes, your should know why.

Eliminate backup copies of modules in SYS1.LPALIB. They often can be identified by names of OLDxxxxx or XXXnnnnn. (129K)

Eliminate old versions of SVC's for vendor packages. You may need two copies of a SVC (test and production) but no more. Do not forget SVC ROUTER members (IGX000nn). (10K)

User SVC modules may have two CSECTS (1K or better) and only one is in use. INQUIRE SVC's seem to have this problem more than other vendors. SMP includes the old module when it is linking in a new module. If there was a CSECT in an old version of SVC 255 for example, it will be carried forward. It will not be used, only taking virtual storage. (4K)

1.3.6.2 Remove SORT modules

I found the IBM SORT modules in LPA. Your installation could have SYNCSORT in LPALIB. I removed these modules: (300K)

IEBCOPY cards	module length
S M=ICEAM1	304
S M=ICECKP	4096

S M=ICECRE	4096
S M=ICECREM	5368
S M=ICECRO	4096
S M=ICEDEC	11912
S M=ICEDED	10920
S M=ICEDEF	8192
S M=ICEDEG	12808
S M=ICEDEGM	13664
S M=ICEDEV	12760
S M=ICEDEVM	1896
S M=ICEERR	1552
S M=ICEEXIT	3992
S M=ICEEXOLD	184
S M=ICEIOTSK	8192
S M=ICEIPUT	20512
S M=ICEKPUS	3800
S M=ICEKPUT	20560
S M=ICELIM	4096
S M=ICELIMM	4672
S M=ICELIV	4096
S M=ICEMAN	27384
S M=IERRC000	ALIAS
S M=IGHRC000	ALIAS
S M=SORT	ALIAS
S M=ICEMANM	4920
S M=ICEMESI	2720
S M=ICEMESO *	3992
S M=ICEMON	8192
S M=ICEOBS	5448
S M=ICEOPUT	20608
S M=ICEPAR	8192
S M=ICEPARM	5352
S M=ICERED	4096
S M=ICEREDM	4456
S M=ICEVED	4096
S M=ICEVEE	4096
S M=ICEVIM	4096
S M=ICEVIMM	4856
S M=ICEVIP	4096
S M=ICEVRE	8192
S M=ICEVREM	6184
S M=ICEVRN	6144
S M=ICEVRO	4096
S M=ICEXCP	5688

1.3.6.3 Remove modules identified by Dallas SC

Remove the modules marked as removeable without damage. This included zapping AMDPRDMP AMDPRSEG to ignore the 10K limit for loading dump modules. *Watch to be sure you get all aliases.* (*150K*)

```
      IEBCOPY cards  module length
      -------------- -----

       S M=IEEVCPU       BCPVARY CPU
       S M=IEEVCPR         ALIAS
       S M=IEEVCH          ALIAS N
       S M=IEEVDSS         ALIAS N
       S M=IEEVWKUP        ALIAS N
       S M=IEEMPVST BCPVARY STORAGE
       S M=IEEVSTEL      ALIAS
       S M=IEEVSTGP      ALIAS
       S M=IEEVSTOR      ALIAS
       S M=IGG0190A GAM--2260 ETC.
       S M=IGG0190E
       S M=IGG0193Y 3
       S M=IGG0203Y 3
       S M=IGC0007D 3
       S M=IGC0107D 3
       S M=IGC0007C 3
       S M=IGC0007E 3
       S M=IGC0107C 3
       S M=IGG0190B 3
       S M=IGG0190J 3
       S M=IGG0190C 3
       S M=IGG086AE 3      IEHATLAS
       S M=IGC0008F 3
       S M=IGG0860A 3
       S M=IGG0860B 3
       S M=IGG0860D 3
       S M=IGX00013 3      MV/1
       S M=IGX00014 3
       S M=IRBMFECH 3
       S M=IRBMFEDV 3
       S M=IRBMFEVT 3
       S M=IRBMFEVE 3 ALIAS
       S M=IRBMFEVL 3 ALIAS
       S M=IRBMFTCH 3
       S M=AMDSYS01 3        PRDMP REQUIRES ZAP TO AMDPRSEG
       S M=AMDSYS02 3
       S M=AMDSYS03 3
       S M=AMDSYS04 3
```



```
       S M=AMDSYS05 3
       S M=AMDSYS06 3
       S M=AMDSYS0S 3
       S M=AMDUSRF8 3
       S M=HMDUSRF8 3 ALIAS
       S M=IMDUSRF8 3 ALIAS
       S M=AMDUSRF9 3
       S M=IMDUSRF9 3 ALIAS
       S M=AMDUSRFD 3
       S M=AMDUSRFE 3
       S M=AMDUSRFF 3
       S M=HMDUSRFF 3 ALIAS
       S M=IMDUSRFF 3 ALIAS
       S M=AMDPRFMT 3
```

Note that there is one exception to the above list. *When I removed IGG0203Z (alias IGG0213Z), my sysres pack increased by 10-15 I/O's per second. It was loading that module for everyone*! Looking at the microfiche, it looks as if some VTAM or other terminal program has taken this name and used it for a terminal handling program.

1.4 *FUTURE VSCR PROJECTS*

1.4.1 *JES2 for MVS SP 1.3.3*

JES2 for MVS SP 1.3.3 should give us more relief (64K?).

1.4.2 *Remove page boundary from IGC0005A (28K)*

By relinking and removing the page boundaries from the abend module IGC0005A we can save 28890 bytes of PLPA. This will be tried when time allows.

1.5 *PREVENTION OF VSC*

1.5.1 *Keep what you gain--IEABJ*

I wrote a module which I called IEABJ. The module is an empty module of 600k, 400K or 200K. The module is included in LPA with and entry in SYS1.PARMLIB member IEALPAxx. This way I can reserve virtual storage until the next major requirement rears its ugly head.

1.5.2 *Track the growth*

I suggest that each Monday morning you print a virtual storage map to compare with last week's map. If it has grown, find out why. You can not prevent it, but you can surely put the blame or responsibility where it belongs.

1.5.3 *Project what is coming*

We estimate that DFEF will cost us 220K (132K in modules alone.) of virtual storage. All the advisors say to get VSAM catalogs out as soon as possible, but we do not project that this is possible due to production jobs.

ISPF/PDF costs an additional 61K even after we moved the old SPF module out of LPA. The base module increased in size and an additional module was added.

1.6 *CONCLUSION: THE GOOD NEWS AND THE CHALLENGING NEWS*

The good news: **IF I CAN DO IT, YOU CAN DO IT.** By lots of hard work you can lower your virtual storage requirements to a point where you have breathing room. You must not quit.

Movement to MVS/XA will help in the battle for virtual storage constraint relief. It will not do it all. Virtual storage must be managed from now on.

The challenging news: As we overcome the problem with virtual storage we force our systems into a real storage

problem. We moved modules to PLPA to save each address space the overhead of its "own" copy. As we go to disk to get the modules more and more real storage is fixed for the duration of the I/O operation (milliseconds is a long time).

Reevaluate your IEASYSxx statement on V=R storage. This makes pages not available for certain requests and makes the paging supervisor work harder.

The same can be said for reconfigurable storage. I found that the IBM service FE's do not take one side of the AP and some storage and run diagnostics. Therefore reconfigurable storage is a waste.

IMS USE OF VIRTUAL STORAGE

STEVE NATHAN
BELL LABORATORIES (BTP)

SHARE 61

SESSION M512

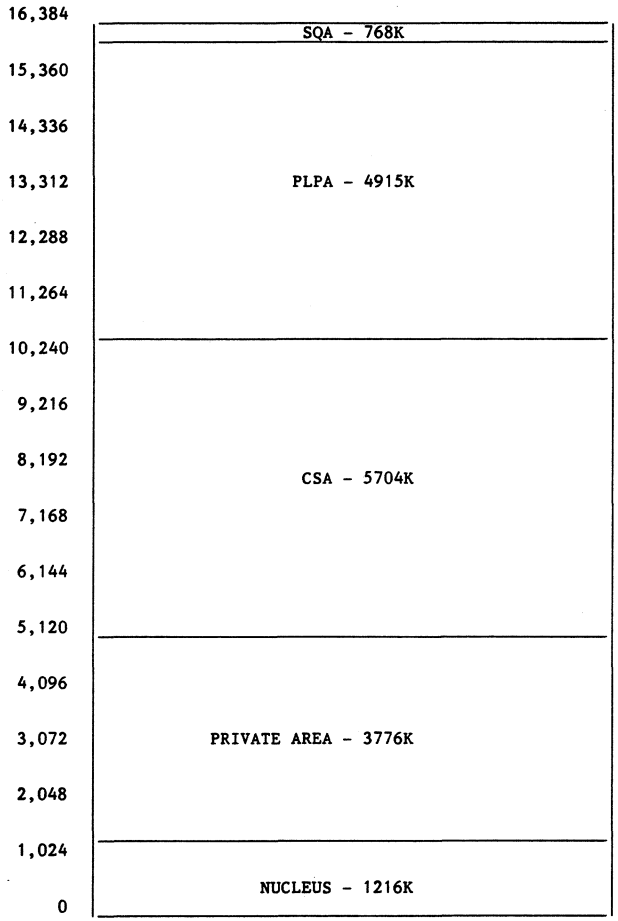
AUGUST 25, 1983

COMPUTER ENVIRONMENT

SYSTEM 'A'	SYSTEM 'T'
3081K - 32MB	3081K - 32MB
MVS/SP 1.3.1	MVS/SP 2.1
4 IMS	4 IMS
120 TSO	25 TSO
14 BATCH	14 BATCH

SYSTEM 'D'	SYSTEM 'E'
4341-12 16MB	4341-12 16MB
MVS/SP 1.3.1	MVS/SP 1.3.1
3 IMS	3 IMS
10 TSO	10 TSO
11 BATCH	11 BATCH

VIRTUAL STORAGE - SYSTEM A



FOIL 3

IMS MODULES IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSAGTOx	SECURITY TABLE FROM MATRIX DATA SET ONLY IF RACF SECURITY USED	VARIABLE	XLIC0
DFSFDTR0	DISK LOG TRACE MODULE - ALIAS DFSFDTRT ONLY IF DISK LOG TRACE IS ACTIVE	7088 BYTES	XDGI0
DFSIDSP0	ONLINE SYSTEM DISPATCHER	5608 BYTES	PLPP0 IIN10
DFSISMN0	POOL STORAGE MANAGER	4328 BYTES	PLPP0 IIN10
DFSLRH00	LOCK REQUEST HANDLER	16032 BYTES	PLPP0 IIN10
DFSMRTR0	WTO MODULE	528 BYTES	PLPP0 IIN10
DFSPTCH0	PATCH AREA	256 BYTES	PLPP0 IIN10
DFSRDSH0	/DBD, /DBR, /STA DB HANDLER	4552 BYTES	XLIC0

TOTAL = 37.5K + DFSAGTOx

FOIL 4

REENTRANT IMS MODULES IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSASK00	INTER-REGION COMMUNICATION ASK SVC	7256 BYTES	VCI00 VCI10
DFSBCK00	INTER-REGION COMMUNICATION BCK SVC (BACKOUT)	2264 BYTES	VCI00 VCI10
DFSCONU0	CONVERSATIONAL SPA UNPACK ROUTINE	232 BYTES	PLPP0 IIN10
DFSCPY00	INTER-REGION COMMUNICATION CPY SVC (COPY)	3224 BYTES	VCI00 VCI10
DFSDFLS0	DL/I FIELD LEVEL SENSITIVITY MAPPING	1760 BYTES	PLPP0 IIN10
DFSDLA00	DL/I CALL ANALYZER	13224 BYTES	PLPP0 IIN10
DFSDLTR0	DL/I CALL IMAGE CAPTURE (/TRA SET ON PSB)	4760 BYTES	PLPP0 IIN10
DFSFDLL0	DISK LOGICAL LOGGER	1848 BYTES	PLPP0 IIN10
DFSFLLG0	LOGICAL LOGGER	3544 BYTES	PLPP0 IIN10
DFSFTIM0	DATE AND TIME SERVICES	408 BYTES	PLPP0 IIN10
DFSFUNL0	LATCH MANAGER	456 BYTES	PLPP0 IIN10
DFSISI00	INTER-SUBSYSTEM INTERFACE	2080 BYTES	VCI00
DFSISIS0	USER SECURITY MODULE ONLY IF USER SECURITY SELECTED	VARIABLE	XLIC0

FOIL 5.1

REENTRANT IMS MODULES IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSKLSPT	LSO MODULE NAME TABLE	280 BYTES	VCI00 VCI10
DFSMNTR0	DC MONITOR RECORD FORMATTING ROUTINE	4072 BYTES	XLGI0
DFSMODU0	MODULE MANAGER	4688 BYTES	VCI00 VCI10
DFSMPOS0	MVS CROSS MEMORY EXECUTOR	152 BYTES	PLPP0 IIN10
DFSPIEX0	PI ENQUEUE/DEQUEUE INTERFACE	584 BYTES	PLPP0 IIN10
DFSREP00	IMS DISPATCHER SCP DEPENDENT CODE SUBPOOL 228	6080 BYTES	PLPP IIN10
DFSSCHR0	RACF SECURITY MODULE ONLY IF RACF SECURITY SELECTED	520 BYTES	XLIC0
DFSTRAC0	COMMON TRACE TABLE ROUTINE	672 BYTES	DVBI0
DFSUICC0	ON-LINE IMAGE COPY GET ROUTINE	1624 BYTES	VCI00 VCI10
DFSUICO0	ON-LINE IMAGE COPY INITIALIZATION	2240 BYTES	VCI00 VCI10
DFSV4100	SYSTEM RECOVERY SUPERVISOR ROUTINE	384 BYTES	VCI00 VCI10
DFSV4200	SCP AUTHORIZED SERVICES	3712 BYTES	VCI00 VCI10

TOTAL = 64.5K + DFSISIS0

FOIL 5.2

NAME	DESCRIPTION	SIZE	WHERE
DFSAOS10	OSAM OPEN/CLOSE/EOV	2800 BYTES REENTRANT	PLPP0 IIN10
DFSAOS60	OSAM READ/WRITE	14240 BYTES REENTRANT	PLPP0 IIN10
DFSAOS70	OSAM I/O DRIVER FRONT END	10016 BYTES REENTRANT	VCI00 VCI10
DFSAOS80	OSAM CHECK I/O COMPLETION	2160 BYTES REENTRANT	PLPP0 IIN10
DFSDBH00	DL/I ISAM/OSAM BUFFER HANDLER	14696 BYTES REENTRANT	PLPP0 IIN10
DFSDBUFF	DISK LOG BUFFERS	# LOG BUFFERS * BUFFER SIZE	XDGI0
DFSDDLE0	DL/I LOAD/INSERT	20600 BYTES REENTRANT	PLPP0 IIN10
DFSDHDS0	DL/I SPACE MANAGER	15184 BYTES REENTRANT	PLPP0 IIN10
DFSDISM0	DL/I ISAM SIMULATOR	3928 BYTES REENTRANT	PLPP0 IIN10
DFSDLD00	DL/I DELETE/REPLACE	32064 BYTES REENTRANT	PLPP0 IIN10
DFSDLOC0	DL/I OPEN/CLOSE	18280 BYTES REENTRANT	PLPP0 IIN10
DFSDLOGB	DYNAMIC LOG BUFFERS	# DBL BUFFERS * DBL BLKSIZE	XLGI0
DFSDLR00	DL/I RETRIEVE	37920 BYTES REENTRANT	PLPP0 IIN10
DFSDPIT0	DL/I PI TRACE ROUTINE	4096 BYTES	PLPP IIN10

NAME	DESCRIPTION	SIZE	WHERE
DFSDSEH0	DL/I REORGANIZATION/LOAD WORK DATA SET GENERATOR	4016 BYTES	PLPP0 IIN10
DFSDVBH0	DL/I VSAM BUFFER HANDLER (ALIAS DFSDVBH2 FOR STATISTICS ENTRY)	5416 BYTES REENTRANT	DVBI0
DFSDVSM0	DL/I VSAM INTERFACE (ALIAS DFSEXLST FOR JRNAD EXIT LIST)	21600 BYTES REENTRANT	DVBI0
DFSDXMT0	DL/I INDEX MAINTENANCE	7304 BYTES REENTRANT	PLPP0 IIN10
DFSDYLEB	DYNAMIC LOG WORK AREA	468 BYTES	XLGI0
DFSERA20	DL/I SNAP PROCESSOR	6672 BYTES REENTRANT	PLPP0 IIN10
DFSFXC10	DL/I ENQUEUE/DEQUEUE PROCESSOR	5072 BYTES REENTRANT	PLPP0 IIN10
DFSFXC50	DL/I DATA BASE SYNCHPOINT PROCESSOR	1560 BYTES REENTRANT	PLPP0 IIN10
DFSLCD	LOG CONTROL DIRECTORY	LONG CALCULATION 11000 BYTES	XLGI0
DFSOBFPL	ISAM/OSAM BUFFER POOL	USER SPECIFIED (DFSVSAMP)	DOBI0
DFSPWKAP	GENERAL POOL (WKAP)	USER SPECIFIED (WKAP=)	IINS0
DFSRDBL0	DL/I DATA BASE CHANGE LOGGER	2472 BYTES REENTRANT	PLPP0 IIN10
DFSRDLG0	DL/I DYNAMIC LOGGER	2976 BYTES REENTRANT	PLPP0 IIN10
DFSXCWxx	ENQUEUE/DEQUEUE QCB STORAGE SUBPOOL 241	USER SPECIFIED (IMSCTF CORE=)	FXC10

REENTRANT TOTAL = 220K

TOTAL = 239K + + +

PST RELATED CONTROL BLOCKS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSABSxx	DIAGNOSTIC WORK AREA	2976 BYTES/PST	MINIO
DFSDLWxx	DL/I RETRIEVE WORK AREA	1280 BYTES/PST	SMICO
DFSDSPxx	DISPATCHER WORK AREA	304 BYTES/PST	MINIO
DFSLOGxx	LOG WORK AREA	1028 BYTES/PST	SMICO
DFSPSTxx	SAP WORK AREA FOR QUEUE MANAGER	256 BYTES/PST	SMICO
DFSISIT	DIRCA AND SSOB FOR EACH PST	236 BYTES/PST	MINIO
DFSPSTP	PST POOL	2304 BYTES/PST + BGWRT, COMMON	XLICO
DFSRSTEB	RESTART PST	2304 BYTES	XLICO
DFSXPST	PST IPOST/ISWITCH WORK AREA	236 BYTES/PST + BGWRT, COMMON, RESTART	XLICO

TOTAL = 8.4K/PST + 2.2K

FOIL 7

BUFFERS AND CONTROL BLOCKS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSBCPTB	CHECKPOINT ID TABLE	RDS BLKSIZE + DECB FOR RDS	RBCPO
DFSDKLWA	DISK LOG WORK AREA	1024 BYTES	XDGIO
DFSFCTP	FUNCTION CONTROL TABLE (FCT) POOL	32 BYTES/PST + 7 OTHER TASKS	XLICO
DFSHSAMB	BUFFERS FOR HSAM BSAM DATA BASES ONLY IF OPENING HSAM BSAM DATA BASES	DATA SET BLKSIZE	DLOCO
DFSLCDST	LOG CONTROL DIRECTORY FOR DC MONITOR	LONG CALCULATION 9300 BYTES	XLGIO
DFSMTCLB	MESSAGE TASK CLB FOR DFSCMTIO	132 BYTES (CLB LENGTH)	IINBO
DFSSAP	SAVE AREA PREFIX TABLE	LONG CALCULATION	XLICO
DFSSLX	SCD LATCH EXTENSION	312 BYTES	XLICO
DFSWEQEL	WRITE ERROR QUEUE ELEMENT	2048 BYTES	DVBIO
DFSXMCTL	DISPATCHER WORK AREA AND SRB'S SUBPOOL 228	BASED ON #PSTS, #ITASKS, # IPOST SRBS	MINIO
DFSZIB00	MAIN STORAGE MANAGEMENT POOL	BASED ON #CONV, #PSTS, #LINES, #DBDS, #PSBS	IINSO ISMNO
OLICAREA	CSA WINDOW TO MOVE IMAGE COPY RECORDS ONLY IF LOCAL VSAM OR LSO OSAM	IMAGE COPY RECORD SIZE	UICCO
OLICDSGA	DATA BASE DATA SET GROUP CONTROL BLOCK ONLY IF PCB NOT SENSITIVE TO THIS DSG	316 BYTES	UICOO
RTVTR	RETRIEVE REGISTER STACK	1280 BYTES/PST	DVBIO

TOTAL = VARIABLE

FOIL 8

OSAM RELATED CONTROL BLOCKS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSEOVOS	OSAM DCB EXTENSION	68 BYTES/OSAM (+ 7 IMS OSAM)*	AOS10
DFSIOSBG	GLOBAL IOSB/IOMA POOL SUBPOOL 228	LONG CALCULATION 12288 BYTES	AOSF0
DFSOSDEB	FAKE OS DEB AND EXTENSION SUBPOOL 241 - KEY 5 - PAGE FIXED	216 BYTES/OSAM (+ 7 IMS OSAM)*	AOSF0

* 7 IMS CONTROL REGION OSAM DATA SETS ARE:
LOG TAPE (IF TOSAM), DYNAMIC LOG, SPA,
SHORT MESSAGE, LONG MESSAGE, QBLKS, AND RDS

TOTAL = 284 BYTES/OSAM + 12K

FOIL 9

VSAM RELATED CONTROL BLOCKS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSBFSPP	VSAM CONTROL BLOCKS AND TRACE TABLES (EVEN IF LOCAL VSAM SUBPOOLS)	VSAM RPL'S + PREFIX (120) + TRACE TABLES*	DVBI0
--------	VSAM GLOBAL BUFFER POOLS ONLY IF GLOBAL BUFFER POOLS	USER SPECIFIED (DFSVSAMP)	DVBI0
--------	VSAM GLOBAL CONTROL BLOCKS** ONLY IF GLOBAL POOLS - SUBPOOL 241	16384 BYTES	IDA 0192Y

* TRACE TABLES ARE LOCK ACTIVITY, DL/I, DISPATCHER, AND SCHEDULER
DEFAULT IS 4096 BYTES/TRACE TABLE

** VSAM CONTROL BLOCKS IN SUBPOOL 241 IF GLOBAL SHARED RESOURCES ARE:
CPA HEADER, IOMB, IOMB EXTENSION, PLH,
SUBPOOL HEADER, VIOT, VSRT, AND WSHD HEADER

TOTAL = VARIABLE

FOIL 10

DMB AND PSB POOLS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSPDMB	DMB POOL	USER SPECIFIED	IINS0
DFSPDBWP	DATA BASE WORK POOL	USER SPECIFIED	IINS0
DFSPPSBW	PSB POOL AND PSB WORK POOL	USER SPECIFIED	IINS0
DFSDMBRS	RESIDENT DMB POOL	USER SPECIFIED	IIND0
DFSPSBRS	RESIDENT PSB POOL	USER SPECIFIED	IIND0

TOTAL = VARIABLE

FOIL 11

DFSBLK0x

NAME	DESCRIPTION	SIZE	WHERE
DFSFXC00	ENQUEUE/DEQUEUE WORK AREA, CONSTANTS, AND STORAGE	3552 BYTES	XDBI0
DFSIDIR0	DATA BASE CONTROL BLOCKS (DDIR) PROGRAM CONTROL BLOCKS (PDIR) TRANSACTION CONTROL BLOCKS (SMB)	48 BYTES/DB 44 BYTES/PROGRAM 76 BYTES/TRAN	XDBI0
DFSIOS50	DCB'S FOR QBLKS, SHMSG, LGMSG, ACBLIB, RDS, AND DBLLOG	1116 BYTES	XDBI0
DFSISCD	SYSTEM CONTENTS DIRECTORY (SCD)	2456 BYTES	XDBI0

TOTAL = 7.0K + USER CONTROL BLOCKS

FOIL 12

USER ROUTINES IN CSA

NAME	DESCRIPTION	SIZE	WHERE
--------	HDAM RANDOMIZERS*	VARIABLE	DLOC0
--------	SEGMENT EDIT/COMPRESSION ROUTINES*	VARIABLE	DLOC0
--------	SECONDARY INDEX MAINTENANCE ROUTINES*	VARIABLE	DLOC0 DXMT0

* WILL BE LOADED INTO LOCAL STORAGE IF LSO IS ACTIVE AND PTF UP27843 IS APPLIED

TOTAL = VARIABLE

FOIL 13

DATA SHARING MODULES AND CONTROL BLOCKS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSANOT	IRLM NOTIFY SRB'S SUBPOOL 228	(#PST * 3) * 88 + 132 BYTES	XLIC0* V4200
DFSIRLM	IRLM PARM AREA AND SAVE SETS SUBPOOL 228	584 BYTES/PST + BGWRT, RESTART, DATA SHARING	XLIC0* V4200
DFSLMGR0	GLOBAL LOCK MANAGER IRLM INTERFACE (HAS ALIAS OF DFSLMBI0 FOR BATCH)	8592 BYTES	XLIC0

* LOADED BY DFSXLIC0 IN AN ONLINE ENVIRONMENT
LOADED BY DFSV4200 IN A BATCH ENVIRONMENT

TOTAL = 8.5K + 848 BYTES/PST

FOIL 14

MSC BUFFERS AND CONTROL BLOCKS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
DFSLXBC	CTC LXB CONTROL BLOCK, IOSB, AND SRB PAGE FIXED	688 BYTES/LINK	IIMS0
DFSLXBM	MTM LXB CONTROL BLOCK AND BUFFERS	INPUT BUFFER + OUTPUT BUFFER + LXB	IIMS0
DFSLXBV	VTAM LXB CONTROL BLOCKS	MAX SESSIONS * VTAM LXB (64)	IIMS0

TOTAL = VARIABLE

FOIL 15

VENDOR MODULES AND CONTROL BLOCKS IN CSA

NAME	DESCRIPTION	SIZE	WHERE
AMExxxxx	CONTROL/IMS	56112 BYTES	
WEMxxxxx	CONTROL/IMS REALTIME	5320 BYTES	

TOTAL = 60K

FOIL 16

IMS USE OF LPA

NAME	DESCRIPTION	SIZE	WHERE
DFSAAP20 IGG019xx	7770 CHANNEL-END APPENDAGE ONLY IF 7770'S DEFINED	552 BYTES	N/A
DFSAFMD0 IGC0905A	FORMATTED DUMP	568 BYTES	N/A
DFSCMC10 IGG019yy	MSC CTC CHANNEL-END APPENDAGE ONLY IF MSC CTC DEFINED	1080 BYTES	N/A
DFSMRCL0	RESOURCE CLEAN-UP	2272 BYTES	N/A

TOTAL = 2840 BYTES + 7770 + MSC

FOIL 17

IMS USE OF SQA
SUBPOOL 245

NAME	DESCRIPTION	SIZE	WHERE
SVCTABLE	IMS MODULE ADDRESSES FOR SVC ROUTING	160 BYTES	VCI00
SSCT	SUB-SYSTEM COMMUNICATION VECTOR TABLE (ALSO CALLED SSCVT IN IMS)	36 BYTES * 2 (2 SSCT'S/IMS)	V4200
SSCD	SECONDARY SCD (CONCATENATED WITH SSCT)	52 BYTES	V4200
SCTEXTN	SSCT EXTENSION	32 BYTES	V4200
SSVT	SUB-SYSTEM VECTOR TABLE	304 BYTES	VCI00
MAJORCDE	LINK PACK AREA CDE + EXTENT LIST	48 BYTES/CDE	MODU0
MINORCDE	LINK PACK AREA MINOR CDE	32 BYTES/CDE	MODU0
MTMWHDR	MSC MTM WINDOW QUEUE HEADER	120 BYTES	V4200
MTMWINDO	MSC MTM WINDOW	212 BYTES/MTM	MTMA0
MWBSTRAC	MSC MTM TRACE TABLE	9616 BYTES	MTMA0
V4100SRB	SRB TO TERMINATE DEPENDENT REGIONS	72 BYTES	V4100

TOTAL = 620 BYTES + CDE'S + MSC

FOIL 18

IMS USE OF PRIVATE AREA

NAME	DESCRIPTION	SIZE	WHERE
DFSVNUCx DFSICCB	CONVERSATIONAL CONTROL BLOCK (CCB)	52 BYTES/ CONV TRAN	
DFSVNUCx DFSICLL0	COMMUNICATION INTERFACE BLOCK (CIB)	96 BYTES/ MFS TERMINAL	
DFSVNUCx DFSICLL0	COMMUNICATION LINE BLOCK (CLB)	132 BYTES/ BTAM LINE, VTAM NODE	
DFSVNUCx DFSICLL0	COMMUNICATION NAME TABLE (CNT)	56 BYTES/ LOGICAL TERMINAL	
DFSVNUCx DFSICLL0	COMMUNICATION RESTART BLOCK (CRB)	56 BYTES/ VTAM NODE	
DFSVNUCx DFSICLL0	COMMUNICATION TERMINAL BLOCK (CTB)	124 BYTES/ TERMINAL	
DFSVNUCx DFSICTT0	COMMUNICATION TERMINAL TABLE (CTT)	40 BYTES/ DEVICE TYPE	
DFSVNUCx DFSICVB0	COMMUNICATION VERB BLOCK (CVB)	16 BYTES/ COMMAND	
DFSPCWAP	COMMUNICATION WORK AREA POOL (CWAP)	USER SPECIFIED (CWAP=)	IINS0
DFSPFBP	MESSAGE FORMAT SERVICE BUFFER POOL (FBP)	USER SPECIFIED (FBP=)	IINS0
DFSPQBUF	MESSAGE QUEUE BUFFER POOL (QBUF)	USER SPECIFIED (QBUF=)	IINS0
DFSPTPDP	COMMUNICATION INPUT/OUTPUT POOL (TPDP) (CIOP)	USER SPECIFIED (TPDP=)	IINS0

FOIL 19

IMS USE

OF

VIRTUAL STORAGE

IS

BETTER

WITH

MVS/XA

FOIL 20

Foil 1. Introduction

The topic I will be discussing today is IMS use of virtual storage. This talk is based on IMS 1.2, at put level 8301. It includes Multiple Systems Coupling and Data Sharing, but it does not include Fast Path. It has been reported that Fast Path does use large amounts of virtual storage in CSA.

Foil 2. Environment

First, a word about the computer environment we are running. One of the primary functions of Bell Laboratories is to develop application software which runs in the Bell Operating Telephone Companies. Our data center is devoted exclusively to one of these very large IMS applications. System 'A' is the primary development host with 4 copies of IMS, 120 TSO users, 14 batch initiators, and assorted started tasks. Most of the 700 terminals are connected in a local VTAM SNA network. System 'A' is a 32MB 3081K, running under MVS/SP 1.3.1, with MVS/SP 2.1.1 scheduled for October of this year. System 'T' is used for Change Management, Integration Testing, and Performance Testing, and also has 4 copies of IMS. This is also a 32MB 3081K which is already running MVS/SP 2.1. Systems 'D' and 'E' are used for System Testing and Training. They are both 16MB 4341-12 CPU's, each with 3 copies of IMS. Another 3081 is scheduled to arrive in November with an unknown number of IMS copies. Our other data center has 8 MVS systems running another 12 copies of IMS and 1 copy of CICS. So you can see why, with 26 copies of IMS, we are concerned about virtual storage.

Foil 3. Virtual Storage

The main concern we have is maintaining enough CSA to run 4 copies of IMS and a large VTAM network, and still have enough private area for JES, IMS, and application development. The Bell System also has a Standard Operating Environment which specifies that a 4MB private area is to be maintained. In order to tune virtual storage I did a detailed study of IMS use of CSA and SQA which I will share with you now. Particular attention will be paid to those areas which can be controlled and tuned.

Tracking what IMS puts in CSA is not difficult in itself. Every module, control block, and buffer which IMS has in CSA is given a name and is put on the IMS CDE chain. These names are shown on the foils which detail IMS CSA usage. The difficulty lies in determining how and why IMS uses the CSA. Almost all of the data which IMS has in CSA is in subpool 231, protect key 7, but there are exceptions which will be noted. Modules and data areas in subpool 228 are fixed in real storage.

Foil 4. IMS Modules in CSA

This first group of modules will always be in CSA, one set per IMS Control Region. The only item which can be controlled is the RACF security table, by limiting the number of entities defined in the IMS Security Generation.

The column labeled "WHERE" contains the last 5 characters of the IMS module that initiates the request for storage. The two modules which load most of the IMS modules in CSA are DFSPLPP0, the Proclib Processor, and DFSVCI00, the SVC Initialization/Termination Routine. They determine which modules to load, and where in virtual storage to load them, based on tables DFSIIN10 and DFSVCI10 respectively. So if you want to change the way IMS handles any of these modules, you can just alter the appropriate table.

Foil 5. Reentrant IMS Modules in CSA

The next group of modules are also normally loaded into CSA by IMS. They are listed separately because they are all reentrant. If you are running more than one IMS Control Region in an MVS system these modules should be put in the Pageable Link Pack Area (PLPA), and removed from the IMS RESLIB. This will save 65K of virtual storage for the second and each subsequent copy of IMS. We did this at Bell Laboratories, and there were absolutely no problems in IMS processing. Putting these modules in PLPA in multiple IMS environments also improves real storage usage and paging performance.

A word of warning. If these modules are in CSA, IMS loads them into subpool 241, protect key 0. So if you determine how much CSA IMS is using by mapping CSA use by subpool and protect key and just looking at protect key 7, as most people do, you will miss these modules. They must be in protect key 0 so they can be executed under the TCB of a dependent region which is running in protect key 8.

Foil 6. LSO Modules, Control Blocks, and Buffers

The IMS Local Storage Option (LSO) can provide instant relief when tuning IMS Virtual Storage. If LSO is used, IMS moves well over 240K of modules, control blocks and buffers from CSA to the Control Region private area. There are some areas which increase their CSA use if LSO is active, mostly when calculating space based on the number of PST's, but these are very small. Module DFSKLSPT is a table which contains a list of all the areas which IMS moves from CSA to the private area when LSO is used. It has been documented that there is a 5 - 10% performance degradation when using LSO. This degradation can be eliminated by using cross-memory LSO (XLSO) if you are running IMS 1.2 under MVS/SP 1.3 or MVS/SP 2.1. As of the time I wrote this (8-1-83), there were some open problems running XLSO under MVS/SP 1.3 and this should be investigated carefully. We are currently running XLSO under MVS/SP 2.1 with no problems.

Foil 6. LSO Modules, Control Blocks, and Buffers (continued)

If you are running multiple copies of IMS, without LSO, in a single MVS, then the reentrant LSO modules noted here should be put into PLPA and removed from the IMS RESLIB. This will save 220K of virtual storage for the second and each subsequent copy of IMS. Remember, if you do not use LSO and you do not put these reentrant modules in PLPA, they will be loaded into subpool 241, protect key 0.

While I was studying these LSO modules, I noticed that DFSDSEH0, DL/I Reorganization/Load Work Data Set Generator, was being loaded into the IMS Control Region. Since load mode is not permitted online, and the reorganization utilities only run in batch mode, there did not seem to be any reason to load this module. IBM agreed, and an APAR is being opened to remove it. The APAR number has not yet been assigned. I have put on a test version of this fix which ran with no problems.

Foil 7. PST Related Control Blocks in CSA

This list shows the control blocks in CSA which are related to the number of PST's (dependent regions) defined for a control region. For some of them, IMS gets a separate control block for each PST. For others, the maximum number of PST's is used in size calculations. You can control the maximum number of PST's via a JCL parameter in the IMS Control Region. If this is overspecified you will be wasting virtual storage, approximately 10K per PST.

There are some PST's which are not associated with dependent regions. They include background write, common services, restart, and data sharing. The number of PST's is also used in other size calculations as described in some of the foils which follow. In some cases, the number of PST's for dependent regions is doubled in the calculation if LSO is active.

Foil 8. Buffers and Control Blocks in CSA

This next list is miscellaneous buffers and control blocks that IMS puts into CSA, which you can study at your convenience. Particular attention should be paid to DFSZIB00, the main IMS storage pool. The size calculation for this pool is based on, among other things, the number of lines, programs, and databases defined in the IMS system generation. The main storage pool is mapped by Free/Allocated Queue Elements (FAQE). Each FAQE is 12 bytes. IMS allocates one FAQE for each data base and program defined in the IMS gen, and two FAQE's for each line. Our main storage pool is 60K per IMS copy because of this calculation, but 55K is unused. This appears to be an area where user modification to IMS code could save virtual storage. This pool is originally allocated by module DFSIINS0 but module DFSISMN0 will obtain additional storage if needed.

Please note the names of the last three items in this list. They are the names of IMS control blocks which are on the IMS CDE chain, but do not start with the characters "DFS".

Foil 9. OSAM Related Control Blocks in CSA

If you are using OSAM data bases IMS builds two control blocks, totaling 284 bytes, in CSA, for each OSAM data set. Even if you do not use OSAM data bases, IMS does. The dynamic log, short message queue, long message queue, queue blocks, SPA, and restart disk data sets are all OSAM. If Tape OSAM (TOSAM) is being used for logging this counts too. It should be noted that each copy of DFSOSDEB is in subpool 241, protect key 5, the Data Management protect key. This is to enable OSAM I/O to be initiated from any address space.

Foil 10. VSAM Related Control Blocks in CSA

If you use global VSAM buffer pools, VSAM will obtain them for IMS in subpool 231, protect key 7. However there will not be a module name for these buffers on the IMS CDE chain. In addition, VSAM will put some control blocks in CSA (subpool 241, protect key 7), that would have been in the control region private area if local VSAM pools were used. These control blocks are also not on the IMS CDE chain.

You can reduce the size of your OSAM and VSAM buffer pools to save virtual storage without necessarily hurting performance. One of the criteria used in tuning these pools is that it may be better to have smaller pools and do database I/O rather than page fault through larger pools. Performance can actually improve if this is done, both for IMS and the system as a whole.

IMS also uses CSA for VSAM data bases even if you use local VSAM buffer pools. The control block DFSBFSPP contains VSAM RPL's and is always in CSA.

This control block also contains storage for the Lock Activity, DL/I, Dispatcher, and Scheduler trace tables, even if they are not being turned on at IMS initialization by control cards in DFSVSAMP. If you do not specify any size for these trace tables in the DFSVSAMP control cards, DFSDVBI0 will allocate 4096 bytes for each of them, for a total of 16,384 bytes. If you try to allocate the minimum number of table entries (200) for each trace table with DFSVSAMP, 19,232 bytes will be allocated. Each entry in the DL/I and Lock Activity trace tables is 32 bytes, and each entry in the Dispatcher and Scheduler trace tables is 16 bytes. Thus if you specifiy the minimum number of entries in DFSVSAMP for the latter, and use the default for the former, IMS will allocate 3200 + 3200 + 4096 + 4096, or 14,592 bytes. This will save 1,792 bytes of CSA per IMS copy. If you do not use these traces, or only activate them via the DFSVSAMP control cards, you can modify IMS code to allocate default tables of minimal sizes.

Foil 11. DMB and PSB Pools in CSA

The storage pools used for data base and program control blocks are always in CSA, even if LSO is being used. Much literature has been written on the sizing of these pools. I strongly recommend that they be monitored and tuned on a regular basis.

Care should also be taken when designating DMB's and PSB's as resident. These control blocks will always be in CSA, even if they are not being used. I heard of one account where the applications had coded 1MB worth of DMB's and PSB's as resident. Also, if the program is eligible for parallel scheduling, (executing in more than one dependent region at a time), and is designated as resident, IMS will not use the copy of the PSB that is in the resident PSB pool. IMS makes a copy of the PSB in the normal PSB pool and uses that one.

Foil 12. DFSBLK0x

The module DFSBKL0x exists mainly to hold the data base, program, and transaction definitions for IMS. These definitions, though small individually, can consume significant amounts of CSA. As as example, an IMS system with 500 data bases, 750 programs, and 1,000 transactions will use 130K of CSA for these definitions. I have found this to be an area where large amounts of virtual storage can be reclaimed. Applications should be made aware of the amount of CSA they are using, and should be conscientious about removing unneeded definitions. This will also save storage in the main storage pool (DFSZIB00) as was previously shown. There are fields in the System Contents Directory (SCD) which contain the total number of each of these definitions.

The other three CSECTS in this load module contain the SCD, enqueue/dequeue work areas, and the DCB's for various control region data sets.

Foil 13. User Routines in CSA

IMS loads HDAM randomizers, segment edit/compression routines, and secondary index maintenance routines into CSA if LSO is not being used. The number and size of these routines is, of course, very installation dependent. These routines were being loaded into CSA unnecessarily when LSO was active. PTF UP27843 fixes this, but beware, it has a very long prerequisite chain.

Foil 14. Data Base Sharing Modules and Control Blocks in CSA

If data base sharing is used, IMS loads additional modules and control blocks into CSA, although they are not very large. I do not know if the IRLM address space itself uses any CSA or SQA, but I imagine it does.

Foil 15. MSC Buffers and Control Blocks in CSA

Multiple Systems Coupling (MSC) will also add IMS code to CSA. It is very dependent on what type of links are used, and how many of them there are. They are additive. If an IMS copy has both CTC and VTAM MSC definitions, for example, then storage will be obtained for both DFSLXBC and DFSLXBV. Also remember that the amount of storage used is based on the number of links in the IMS system generation, not the number active. If unneeded links are defined, storage will be wasted.

Foil 16. Other Vendor Modules and Control Blocks in CSA

IBM is not the only organization which can use storage for IMS in CSA. The one example I know of is Control/IMS and Control/IMS Realtime, which together use 60K of CSA per IMS copy. There may be other examples, but they are not installed on our IMS copies.

Foil 17. IMS Use of LPA

There are some IMS modules which must reside in LPA. They do not consume a significant amount of virtual storage, and of course, there is only one copy of each per MVS system.

Foil 18. IMS Use of SQA

To complete the picture of IMS use of common storage, here is a look at IMS use of SQA. This amounts to very little unless MSC memory-to-memory is being used. The names on this foil are only descriptive. These control blocks are not on the IMS CDE chain. The only way to find IMS use of SQA is to read IMS code.

Foil 19. IMS Private Area

I have not had the time to do a detailed study of the IMS private area, but some obvious tuning ideas come to mind after looking at IMS use of common storage.

First, pay close attention to the IMS data communications definitions. These definitions always seem to be added to the IMS system generation, but unused ones are never deleted. Just like the data base, program, and transaction definitions, these are small individually, but when multiplied by the number of entities, grow rapidly.

Foil 19. IMS Private Area (continued)

Second, monitor and tune the private area pools on a regular basis. As with OSAM and VSAM data base pools, reducing the size of these pools does not necessarily have to hurt performance. A well tuned I/O subsystem and smaller pools can outperform the paging subsystem. Just as important, if IMS takes a page fault the whole control region waits. If IMS does an I/O, only one user waits while the control region continues processing.

Third, be curious. Scan the IMS CDE chain, find anything that looks large, and investigate it. Reading IMS code and logic manuals can be fun, enlightening, and profitable.

Foil 20. Conclusion

I hope that this talk has given you an idea of how IMS uses virtual storage and how it can be tuned. The foils should be in enough detail so that they can be used by IMS systems programmers to apply specific tuning strategies at each installation.

As final note of hope, for those of you who have not yet migrated to MVS/XA, it is wonderful. We realized an immediate gain of 1.6MB of virtual storage with MVS/SP 2.1. We are installing MVS/SP 2.1.1 in October, and I am eagerly awaiting another 400K of virtual storage to give to IMS.

VIRTUAL STORAGE CONTRAINT RELIEF/ the NICKEL-DIME APPROACH

Clark Morris
N. American Philips Lighting
201-429-3607 (WEJ)

My name is Clark Morris and I am a senior systems programmer at North American Philips Lighting Corporation, SHARE code WEJ. For long time attendees, we used to be known as Westinghouse Lamp Divisions. We have a 4341 group 2 with 8 megabytes, 4 actuators of 3380, 14 actuators of 3350 over 2 channels, 8 tape drives, and Comten 3650 plus various remotes and 3278's. We run MVS SP 1.3.2, JES3 SP 1.3.1, ACF-TCAM 2.3 with TSO and message switching, both a production and a test CICS at the 1.5 level with DL1 and SPF with up to 20 logged on users, in short the usual alphanumeric soup.

What I am going to be discussing briefly is the nickel-dime approach to virtual storage constraint relief otherwise known as removing unneeded modules form the LINKPACK area. This will cover some of the reasons for doing so, some of the pitfalls I found in the first steps I took toward removal of unneeded modules, some of the methodology used for determining which modules to remove, and some of the steps that I am looking at taking in the future.

Probably the initial motivation was that I saw my LPA usage growing and being an ex-MVT user one of the first things I understood was that there probably were unnecessary modules. Currently my system has a 95.10 full LPA page data set which is 1430 slots, 11 cylinders. For performance reasons, among others, I really don't want it to grow too much bigger. This figure is a slight increase over 2 weeks ago since we went from MVS SP1.3.1 to MVS SP1.3.2 and from JES3 SU26 to JES3 SP1.3.1. This increase is despite the fact I removed JES2 from our system and the FMID EMI1102. This is troubling since we have not yet installed DF/EF and we probably should put some more CICS modules in the Link Pack Area. I have made some of the COBOL routines resident for performance reasons since we compile using the DYNAM option. We also made the Sort resident for Performance reasons. Thus you can see that even a medium sized shop (an 8 meg 4341 group 2) can be looking at possibly having virtual storage problems in the not too distant future.

Some other reasons you might want to consider for taking this somewhat interesting and hazardous path are that there are a number of things it is useful to have in the PLPA. Access within the data set is basically random so an increase in its size may well mean an increase in page delay time on a heavily used PLPA. Even in XA with all programs in the 31 bit mode, 20 megabytes may be the largest you want the PLPA to grow because that is the recommended maximum size for a 3880-11 page data set. In case this latter figure looks like one which should not be reached remember that the FORTRAN Compiler is now re-entrant and some installations may get a noticeable performance improvement by putting it in the PLPA. This will probably take 500k to a megabyte alone.

Turning to methodology and experience for this excercise. My first step was to do a full sysgen to some target data sets and compare libraries. The results were most interesting. One of the first things I noticed was that CAM modules were not carried forward, a fact which quickly convinced me never to put a full sysgen into production if at all possible. CICS, SPF, and TSO modules also were not in the PLPA but that was to be expected. More interesting was the fact that we were carrying some MICR modules and Mass Storage modules despite the fact that we have neither device class and have no intentions of getting them. I then used GTF to gather information on the modules actually used by collecting the link, load, and XCTL records. By

accident one of the traces was left on for 18 hours and took 17 cylinder of 3350 space. GTF itself seemed to take less than 5 minutes CPU time for that period. I used DYL280 to sort and summarize the records since we don't have SAS and don't have GTFPARS. The results were used to determine which COBOL modules to put into the LPA. I also removed some TSO modules from the LPA library based on the results. The information also suggested that even more TSO modules could be removed. Unfortunately GTF does not pick up SVC calls and certain other module use information when just the load, link and XCTL records are specified. The data gathered also suggested that some reduction of system overhead could be gained by the combining of modules.

The next step was to have our SE verify that I could safely remove JES2 and MICR modules. A couple of weeks later he got the confirmation from the Dallas systems center. I removed the FMID's by applying and accepting a USERMOD which deleted EJE1102 and EMI1102. This removed all modules from the systems and distribution libraries and all related PTF's from the PTS. A side discovery in this process is that when you delete a FMID, you also should delete the PTS entry for the FMID or you will continue to get maintenance. I went through my PTS intending to delete JES2 maintenance and instead ended up deleting TCAM 10 maintenance.

The surprises started coming when I deleted EMI1102 which looks like MICR but might better be called the obscure device FMID. I found we also lost support for OCR, tape cartridge readers, and diskette readers. Fortunately we don't have any intentions of having any of these devices on our system. Then I did a sysgen to apply MVS SP1.3.2 and found that the three sysgen macros owned by EMI1102 were unconditionally required by sysgen. Even more surprising was the fact that for SYSGEN purposes, a 3505 and a 3525 are in this obscure device category because Rochester owned them. We HAVE a 3505 and a 3525. Research with a very helpful person at level 2 sysgen verified that all of the 3505 and 3525 modules were owned by EDM1102 so that I hadn't deleted any of them and that the SYSGEN macros in question had last changed with release 1.3 of VS1. Thus I felt confident in restoring them to a special library from a backup tape and redoing the sysgen pointing to that library in the concatenation. Elimination of EMI1102 bought about 10K and of JES2 35k so that the savings were trivial. However the experience was useful and the data gathered showed that there are potential savings of 300k to 350k. If we remove the capability of running free VTAM2 in emergencies and on weekends, we can save 300k. Removal of some more TSO unused modules is the other area of simple reduction. If I could get rid of MSS from the target, or running libraries without doing a full sysgen and leave it in the DLIB's and continue to get maintenance I would do it. It seems theoretically feasible and clean to just apply a deleting user mod and not accept that mod. If and when we get rid of ISAM in our shop, those modules represent 40-50K of LPA that can be removed. Even now, if we were really tight on space we would experiment with moving them to SYS1.SVCLIB and see if the system would pull them into the address space.

In conclusion, while removing modules is the slow way to virtual storage contraint relief, it does have benefits and the excercise may tell various things that will help your system run better. Because there are a number of modules it is desirable to have in the LPA, it would be useful to easily eliminate unused modules from our running libraries and keep them maintained only in our distribution libraries so that we have them if we ever need them. This study also suggested architectural changes that could give at least some relief and this might be a good topic for follow-on discussions.

SESSION REPORT

SHARE

61	M560	A MANAGEMENT PERSPECTIVE OF ~~PROJECT MANAGEMENT~~	130
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE
ADM		Carolyn Andersen	USF
PROJECT		SESSION CHAIRMAN	INST. CODE

A MANAGEMENT PERSPECTIVE

OF

PROJECT MANAGEMENT

Dr. Doris F. Bernardini

IBM Information Systems Management Institute
909 Third Avenue
New York, New York 10022

Session No. M560

Project ADM