IBM Enterprise2013
pOS585 - Using SVMON to Understand AIX Memory Usage

Grover Davidson – grover@us.ibm.com

Objectives are to identify and understand:

- Segmented memory architecture
- System VMM segment types
- Virtual address space
- Effective address space
- Private and Shared memory segments
- File system cache

Segmented memory architecture

- Allows every process to function like it is the only process on the system
- Removes developer concerns about what is running on the system outside of their individual program
- Saves memory by allowing ‘holes’ in physical memory assigned
- 32 bit processes can have 16 segments: 0-F
- 64 bit processes can have up to 2^36 possible segments (note: this requires 64 bit hardware)
- AIX 5L supports multiple page sizes: 4k/64k/16M/16G
- 4k pages are supported on all platforms
- All other pages sizes are platform dependant
- A single segment references 256MB of address space

VMM Page sizes

- Physical memory pages are called Memory Frames.
- A frame consists of a specific amount of physically contiguous memory (the page size) aligned on a physical address boundary.
- The ability to support different page sizes is directly tied to the hardware memory management hardware.
- AIX maintains a free list of both 4k, 64k and 16M pages to be assigned as needed.
- All files are backed by 4k pages.
- Working storage can use all page sizes.

VMM page size selection

- AIX 6.1 supports dynamic page promotion. When 16 (4k) frames are *virtually* contiguous the VMM will attempt to migrate the pages to a single 64k page.
- The balance of 4k/64k pages is maintained by psmd (Page Size Management Damon).
- Page sizes can be explicitly chosen for a process by using the LDR_CNTRL environment variable or using the ldedit command on an executable. LDR_CNTRL is preferred in most cases.
- Using 64k pages CAN result in a larger memory footprint for an application due to memory holes.
- Using 64k pages reduces page faults and is more efficient for addressing memory.
- Do not set LDR_CNTRL to use all 64k pages in /etc/environment! There are many programs that will never need 64k pages and will waste memory!

Segment characteristics

- Addresses have a segment, page and page offset
  - Starting from the least significant digit of an address:
  - Typical Address: F10006000 35A6C00

    Segment   Page/offset

    - 7 hex digits for page/page offset:

      4k pages: 4 hex digits for page number/3 hex digits for offset

      64k pages: 3 hex digits for page number/4 hex digits for offset

      16M pages: 1 hex digit for page number/6 hex digits for offset

      16G pages: Handled completely different
    - As of AIX 6.1 on POWER6 segments can have multiple page sizes
    - Remaining digits are the effective segment id (ESID) –

      1 hex digit for 32 bit applications

      9 hex digits for 64 bit applications
- The process address uses the ESID
- The system wide segment id is the VSID
- Translation is performed by hardware, VMM and the hypervisor
- The use of ESID/VSID depends on if the address is being referenced from the process space or kernel space

Assigning physical pages to a segment

- Segments are created as address space is needed by a process or kernel
- Initially have no physical pages in them
- Addressable pages are malloced/allocated
- As each address is referenced by the program, it is checked to be a valid address
- If the address is valid, the page frame table is checked for a reference to the frame
- If there is no reference, a page is assigned from the free list
- eXternal Page Tables (XPT) provide the tracking of pages on paging space devices
- File system pages are tracked though file system specific routines

Page Frame Table

- Used to translate between the virtual addresses and memory frames.
- Used to be part of AIX before virtualization.
- Now part of the hypervisor.
- Prevents an LPAR from changing it’s own memory translations to access memory from another LPAR.
- Created based on the maximum amount of memory the LPAR can have assigned.
- These pages are part of the LPAR memory but not directly accessible by AIX.
- Part of the ‘other’ pages in svmon output.

VMM Segment types

- Segments are classified by how they are being used
- The classification can change as the segment is used in different ways
- Processes that have the same VSID mapped into their address space are using the same memory
- Segments are classified to allow for memory management, security, and usage information

Basic Segment types

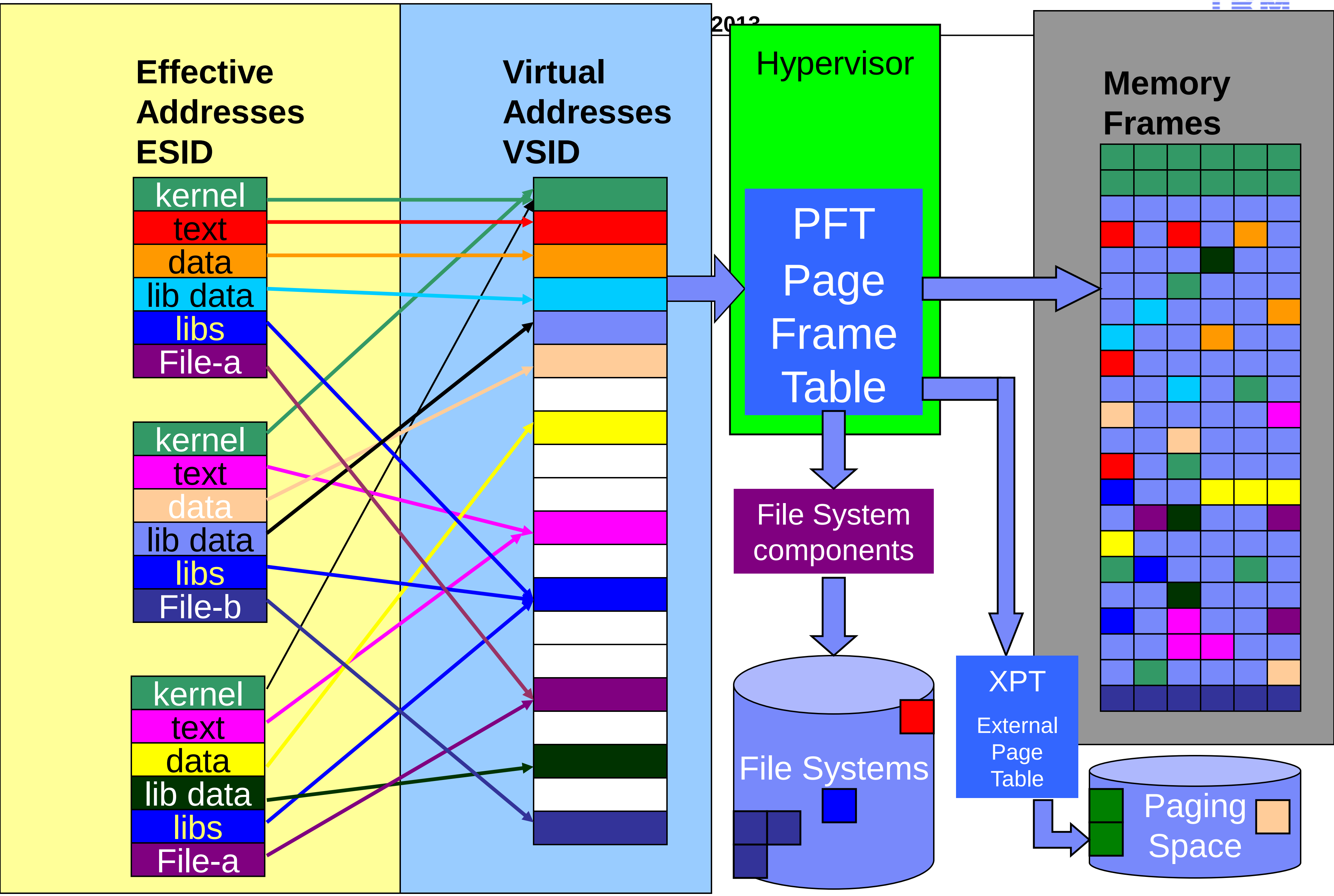
- Working
  - – Kernel – segment is used in kernel space and is addressable by all processes – kernel and application
  - – Text/shared library text – Contain executable code
    - Not modifiable except through debuggers
    - Normally shared by multiple processes
  - – Data – typically not shared and private to a single process
  - – mbuf – Network memory segments
- Mmap/shmat’ed – segments specifically setup to be mapped by 0 or more processes
  - – Some applications use shmat segments to hold various data and are not normally attached to any process

Basic Segment types (cont'd)

- Persistent – JFS file segment
  - Tightly integrated with the kernel
- Client – Non-JFS file segment – JFS2/NFS/Veritas/etc
  - Externalized from the kernel
- Multiple processes can map the segment by opening the same file
- All processes that map a segment have access to all pages referenced by that segment

Paging activity

- Used mainly for working storage pages, there are some rare exceptions
- Pages can be stolen and their contents are copied to paging space on a page by page basis
- If the page is referenced, a page fault occurs and the page is copied back into memory from paging space
- The page can be allocated just in real memory, just on paging space OR both
- These details are handled by the XPT
- Applications have no direct control over paging



Effective Addresses ESID

kernel
text
data
lib data
libs
File-a

kernel
text
data
lib data
libs
File-b

kernel
text
data
lib data
libs
File-a

Virtual Addresses VSID

Hypervisor

PFT Page Frame Table

File System components

File Systems

XPT External Page Table

Memory Frames

Paging Space

“svmon –S” – A system wide view

```
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   2c6d6         - pers /dev/testcase2_lv:20         s  84243     0    -       -
    cb26         - pers /dev/hd1:2079                s  38274     0    -       -
   1e00f         - work kernel heap                  s  30218 12928    0   30218
   341ba         - work                              s  22068    26    0   22068
   24d12         - pers /dev/testcase2_lv:19         s  20975     0    -       -
    8c44         - clnt /dev/j2_test2:4              s  20671     0    -       -
   1c00e         - work misc kernel tables           s  17322     0    0   17322
   12c29         - clnt /dev/j2_test2:5              s  14667     0    -       -
```

Vsid – unique system segment ID.

Esid – effective segment – only applies from process view

Type – Segment type: work/persistent/client

Desc – Any additional information like device:inode or details of how the segment is being used.

PSize – Page size: s=4k m=64k L=16M S=16G

Inuse – actual number a memory frames in use

Pin – pages that cannot be stolen by lrud

Pgsp – paging space usage

Virtual – total number of virtual pages created. This indicates how many pages would be needed for all the accessed pages to be in memory.

“svmon –P” – View from the process

```
# svmon -P 43818
    Pid Command              Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
  43818 less                 25520      9293       0    25478      N     N     N
   Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
  3601b         d work shared library text          s  13594     0    0 13594
      0         0 work kernel segment               s  11833  9291    0 11833
  11349         1 clnt code                         s     33     0    -     -
  1936d         2 work process private              s     26     2    0    26
  29415         f work shared library data          s     25     0    0    25
  393dd         - clnt /dev/j2_lv:4108              s      8     0    -     -
   46a2         - pers /dev/hd2:221186              s      1     0    -     -
```

The first summary line of page counts for the process is always in units of 4k pages.

Vsid – System wide segment number. 2 processes with the same Vsid mapped are sharing the same memory

Esid – Effective segment based on the address the process addresses the segment at. Esid’s are different in 64 and 32 bit *processes*.

Esid’s of ‘-’ here indicate a file the is open by the process. The file is found with: ‘find /usr -inum 221186 -xdev’. This segment is shared by all processes accessing this file.

Note that Vsid 0 is mapped to all processes. This is how the processes access the kernel. Vsid 3601b is also shared by all non-kernel processes on the system. This is the shared library code. Any other instances of the ‘less’ command will use the same VSID. The exact segment numbers will vary from system to system.

All other segments here are private to this process.

svmon with multiple page sizes

```
   Pid Command            Inuse      Pin      Pgsp  Virtual 64-bit Mthrd  16MB
446880 db2sysc           797610      339         1   707095      Y     N     N

   PageSize          Inuse         Pin         Pgsp      Virtual
   s    4 KB        794325         320            0       703809
   m   64 KB          3269           3            1         3270

  Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
170ab6         - clnt /dev/fslv13:12328            s  90504     0    -       -
     0         0 work kernel (lgpg_vsid=0)         L     16    16    0      16
1070b0  90000000 work shared library text          m   3262     0    1    3263
 70ac6 f00000002 work process private              m      7     3    0       7

Many lines have been deleted to save space.
```

- Note here that the kernel segment is using 16MB pages (PSize=L). To compute the pages added to in use multiply by 4096 (16 * 1024 * 1024 / 4096).
- The medium pages add 3269 * 16 = 52304 (4k) pages.
- Remember that the summary in use is in units of 4k pages. The Inuse for each SEGMENT is in the page size for that segment.

Segments with multiple page sizes

```
--------------------------------------------------------------------------------
     Pid Command          Inuse      Pin      Pgsp  Virtual 64-bit Mthrd  16MB
36372852 kdb              42164    14080         0    30887      Y     N     N

     PageSize                 Inuse        Pin       Pgsp    Virtual
     s    4 KB                16788          0          0       5511
     m   64 KB                 1586        880          0       1586

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   20002         0 work kernel segment                m    937   877    0     937
  a10021  90000000 work shared library text           m    644     0    0     644
  85a604         - clnt remote                        s   9079     0    -       -
   50005  9ffffffd work shared library               sm   2899     0    0    2899
  f1ce71        11 work text data BSS heap           sm   1772     0    0    1772
```

- Page size column showing ‘sm’ for the page size indicates multiple pages sizes are being used in this segment.
- Unable to determine details of how many of each pages sizes is in use.
- The Inuse/Pin/Pgsp/Virtual page counts are adjusted to display all in 4k units.

Segments with multiple page size details

```
 svmon -O pgsz=on,segment=on,mpss=on  -P 36372852
Unit: page
-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual
36372852 kdb              42164    14080        0    30887
     PageSize                Inuse        Pin       Pgsp    Virtual
     s    4 KB               16788          0          0       5511
     m   64 KB                1586        880          0       1586
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   20002         0 work kernel segment               m    937   877    0     937
  a10021  90000000 work shared library text          m    644     0    0     644
  85a604         - clnt remote                       s   9079     0    -       -
   50005  9ffffffd work shared library               s   2019     0    0    2899
                                                     m     55     0    0       0
  f1ce71        11 work text data BSS heap           s    636     0    0    1772
                                                     m     71     0    0       0
```

- Each page size is lised and the number of pages in each catagory.

▪When using ’svmon –S’ use ’svmon –O mpss=on –S’ to see the segement page size details.

▪For segment 50005 we have 2019 (4k) pages + 55 (64k) pages = 2019 + 55(16 4k pages per 64k page) = 2899 (64k) pages. This correctly matches the output from the previous slide.

Differences between svmon –S and -P

- -S
  - No segment is listed more than 1 time
  - Segments not associated with a process are listed
  - Unable to tell how many processes are referencing a segment
- -P
  - Segment is listed for every process using it
  - Segments not associated with a process are not listed
  - Most kernel segments are not shown
  - Files not opened by a process but that are cached are not listed

“svmon –G” – a global summary

```
/usr/bin/svmon -G
----------
                size       inuse        free         pin     virtual   mmode
memory      18284544    18234208       50336     2495215     4369991     Ded
pg space     4325376       15337
                work        pers        clnt       other
pin          1950719           0           0      544496
in use       4369991           0    13864217
PageSize    PoolSize       inuse        pgsp         pin     virtual
s    4 KB          -    15312896       15337      729887     1448679
m   64 KB          -      182582           0      110333      182582
```

- All segments are counted and classified.
- ‘Other pages’ summarize pages in use but do not have a VMM segment, like the page frame table in the hypervisor.
- Memory mode is listed, in this case it is traditional dedicated mode.
- Paging space usage is summarized.
- The total number of memory frames needed for working segments (everything except file cache) is the value under ‘virtual’. This is the number of 4k pages needed for working storage. This should be approximately equal to AVM column of vmstat.
- File system cache in use is determined by adding pers and clnt in use values.
- Memory page usage by page size is also summarized here.

2 instances of the same executable

```
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
   37224 more             25644     9293        0    25522      N     N     N
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   3601b         d work text or shared-lib code seg  s  13622     0    0 13622
       0         0 work kernel segment               s  11833  9291    0 11833
    53c3         - pers /dev/hd3:80                  s    105     0    -     -
   29355         f work working storage              s     31     0    0    31
   133e8         3 work working storage              s     19     0    0    19
   193ed         2 work process private              s     17     2    0    17
    6723         1 pers code,/dev/hd2:108702         s     16     0    -     -
    46a2         - pers /dev/hd2:221186              s      1     0    -     -
-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
   40074 more             25542     9293        0    25523      N     N     N
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   3601b         d work text or shared-lib code seg  s  13622     0    0 13622
       0         0 work kernel segment               s  11833  9291    0 11833
    f486         f work working storage              s     30     0    0    30
   10fa9         3 work working storage              s     21     0    0    21
   13488         2 work process private              s     17     2    0    17
    6723         1 pers code,/dev/hd2:108702         s     16     0    -     -
   3871c         - pers /dev/hd2:117193              s      1     0    -     -
    8024         - pers /dev/hd4:8469                s      1     0    -     -
    46a2         - pers /dev/hd2:221186              s      1     0    -     -
```

```
# find /usr  -xdev -inum 108702
/usr/bin/more
```

2 processes accessing the same file

```
-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
   42436 more             26267     9294        0    25521      N     N     N
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   3601b         d work text or shared-lib code seg  s  13623     0    0 13623
       0         0 work kernel segment               s  11833  9292    0 11833
   12ce9         - pers /dev/hd3:300                 s    730     0    -     -
   191ad         f work working storage              s     31     0    0    31
    f486         3 work working storage              s     19     0    0    19
    6723         1 pers code,/dev/hd2:108702         s     16     0    -     -
    eb27         2 work process private              s     15     2    0    15
-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
   31562 pg               26247     9294        0    25509      N     N     N
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   3601b         d work text or shared-lib code seg  s  13623     0    0 13623
       0         0 work kernel segment               s  11833  9292    0 11833
   12ce9         - pers /dev/hd3:300                 s    730     0    -     -
   29355         f work working storage              s     30     0    0    30
   1d48f         2 work process private              s     16     2    0    16
   16c4b         1 pers code,/dev/hd2:108726         s      8     0    -     -
   133e8         3 work working storage              s      7     0    0     7
```

```
# find /tmp -inum 300
/tmp/errorlog.grover
```

The pages used by segment 12ce9 are SHARED by both processes. It does not matter which process causes the pages to be read, they are available to all processes accessing the segment.

NOTE: Not all file systems provide this level of detail! Encrypted file systems may add additional complications.

Shared Memory Segments

```
svmon -Pns
-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
  442782 db2sysc         853887      339        1   707265      Y     N     N

     PageSize      Inuse        Pin       Pgsp    Virtual
     s   4 KB     850602        320          0     703979
     m  64 KB       3269          3          1       3270

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
  170ab6         - clnt /dev/fslv13:12328            s  90504     0    -     -
  170856  78000003 work default shmat/mmap           s  65536     0    0 65536
   80989  78000004 work default shmat/mmap           s  65536     0    0 65536
       0         0 work kernel (lgpg_vsid=0)         L     16    16    0    16
   70986  78000008 work default shmat/mmap           s  65536     0    0 65536
   20983  78000009 work default shmat/mmap           s  65536     0    0 65536
   50984  78000007 work default shmat/mmap           s  65536     0    0 65536
   e09af  78000006 work default shmat/mmap           s  65536   312    0 65536
   d09ac  7800000a work default shmat/mmap           s  65536     0    0 65536
```

Shared segments are identified as shmat'ed or mmap’ed. Any processes that have these segments mapped are in fact accessing the same memory.

Please note that these segments are fully populated with pages! 65536 * 4k = 256MB.

Shmat’ed memory segments are not deleted when they are not mapped to a process. These segments must be explicitly deleted. Segments marked for deletion will not be deleted until no processes are attached to them.

Correlating ipcs with svmon

```
# ipcs -aS
IPC status from /dev/mem as of Mon Aug 28 16:21:17 MST 2006
T        ID     KEY        MODE       OWNER    GROUP  CREATOR   CGROUP NATTCH     SEGSZ
Shared Memory:
m   1048576 0x00004dbe --rw-rw-rw-     root   sapsys     root   sapsys      1   1845702
SID :
0xe084f
m   5242881 0xffffffff --rw-------   rctadm   sapsys   rctadm   sapsys      0 2147483648
SID :
0x90548 0x1b05da 0x601 0x107e0 0x1807b9 0x1605b7 0x1c05dd
```

‘svmon –U’ will NOT add these segments in because they are not attached to a process. Note that this information looks more like file information.

NATTCH indicates how many process have the segment mapped into their address space.

The SID listed is the VSID that can be examined with svmon

The second shared memory segment listed here is larger than a single memory segment and as a result, multiple segments are allocated.

The first segment will appear in both the process and system output for svmon since it is mapped by a process.

Since the second segment is not mapped, it will not appear as a shared memory segment in svmon!

Identifying shared segments

Execute several instances of the program and select 1 instance of the program and examine each segment to see how many processes are using it:

```
# svmon -P 29180 -l
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   3601b         d work shared library text          s  13538     0    0 13538
                   Shared library text segment
       0         0 work kernel segment               s  11833  9293    0 11833
                   System segment
   164ab         1 pers code,/dev/hd2:32977          s     18     0    -      -
                       pid(s)=30730, 30468, 29700, 29434, 29180, 26144, 25574
                       pid(s)=24790, 24534, 24282, 24014, 23768, 23246, 23020,
                       pid(s)=22742, 22516, 17372, 16342, 13764
   344fa         2 work process private              s     18     2    0    18
                   parent=264f3
                   pid(s)=29180
   384fc         f work shared library data          s      5     0    0     5
                   parent=2a4f5
                   pid(s)=29180
```

Here we see that segment 164ab is shared because it is mapped to multiple processes. Shared library text segments do not list each process they are mapped into.

File system cache

- The file system cache will grow to use all free pages by default
- All processes accessing a file share access to all the pages of the file in the cache
- Pages for a file remain in memory even after the file is closed until they are stolen or the file is unlinked or the file system is unmounted
- File system segments (pers or clnt) that are marked as unused by svmon are files that are not currently open but have pages in the file cache
- Executables are reclassified from file pages to working pages in most cases

How much memory does the file cache need?

- Depends on the type of application:
  - Databases that have internal caches like db2 can use CIO to disable AIX caching – use application statistics to determine the cache hit ratio to determine the effectiveness of the application cache
  - Databases without caches and other applications that rely on the operating system to cache will need more memory that can be used for caching
- Every system needs some file system cache
- Applications that access file systems will fill available memory with cached files
- Many backup utilities can be configured to use CIO/DIO to avoid flushing the active cache
- This is all normal and expected

What does AVM size really mean?

- AVM (Active Virtual Memory) column of vmstat or the Virtual value of svmon represents how much memory in 4k pages are needed for non-filesystem cache. This is all of the memory needed to execute the programs on the system.
- If Virtual/AVM exceeds the real memory on the system, paging cannot be prevented.

How much memory is a process using?

- Identify:
  - Private data
  - Code
  - Shared library
  - Process Shared memory
  - System segments (these are generally not included)
- Total memory = (N * private_data_pages) + Code + shared_libraries + process_shared_memory

N= number of instances of the process

A real example

- Find all the instances of the program:

```
# ps -ef | grep kdb_64
  grover  692314 1073586   0 14:53:06 pts/128  0:02 ./kdb_64 ./dump ./unix
   sivar  950408  966850   0 15:12:19 pts/80   0:04 ./kdb_64 dump unix
raghavan 1589418  716976   0 15:15:46 pts/99   0:08 ./kdb_64 ./dump ./unix
   asulu  917900 1851674   0   Dec 27 pts/108  0:03 ./kdb_64 dump unix
```

- Note that just because the names are the same it does not mean the executables are the same
- Select 1 instance and collect ‘svmon –P XXX –l’ and catagorize the segment based on the number of PIDs is it mapped to:

```
# svmon -P 692314 -l
-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
  692314 kdb_64           78815    65558      544    71989      Y     N     N
    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
       0         0 work kernel (lgpg_vsid=0)         L     16    16    0    16
                   System segment
  12cbd0         - clnt /dev/lvsivar:90946           s   6702     0    -     -
                   pid(s)=897244, 692314
   f2f8c        11 work text data BSS heap           s   3214     0    0  3214
                   pid(s)=692314
   10001  9ffffffd work shared library               s   1532     0  495  2018
                   System segment
    30a0  90000000 work shared library text          s    862     0   14   888
                   Shared library text segment
  1aee9b         - clnt /dev/lvsivar:91006           s    349     0    -     -
                   pid(s)=897244, 692314
  1ea55c        10 clnt text data BSS heap,          s    298     0    -     -
                        /dev/lvsivar:90949
                   pid(s)=897244, 692314
   a01ab  90020014 work shared library               s    190     0   35   213
                   pid(s)=917900, 282906, 147872, 1093808, 897244, 692314
   f022e  9001000a work shared library data          s     43     0    0    43
                   pid(s)=692314
   b0248 f00000002 work process private              s     28    22    0    28
                   pid(s)=692314
   c2faf  80020014 work USLA heap                    s     24     0    0    24
                   pid(s)=692314
   40024  9ffffffe work shared library               s     12     0    0    12
                   System segment
  1b01ba  9fffffff clnt USLA text,/dev/hd2:12355     s     12     0    -     -
                   pid(s)=917900, 282906, 1093808, 897244, 692314
  1a3079  ffffffff work application stack            s      9     0    0     9
                   pid(s)=692314
  1b2f78  8fffffff work private load data            s      4     0    0     4
                   pid(s)=692314
```

Shared Files

Shared Text

System Shared

Private work

Adding it up!

- Process private segments are f2f8c, f022e, b0248, c2faf, 1a3079 and 1b2f78. These add up to 3322 pages of memory needed for EACH instance of this executable. Check several instances of this executable and average them.
- Keep in mind that there can be variations in memory usage in private segments depending on the execution path of each instance. Each instance should be examined!
- Shared text segments are 1ea55c, a01ab and 1b01ba. The first instance of this executable also needs 500 pages. Of the 500 pages, 202 are shared with other programs (segments a01ab and 1b01ba).
- The files are using 7051 pages in the file cache for these specific files. Depending on the specific files, they may or may not need to be completely cached.
- Man pages have additional flags for svmon – read them very carefully to understand the flags

AIX 6.1/7.1 updates/changes

- ‘rmss’ pages are also listed on the ‘svmon –G’ output as ‘stolen’ pages.
- New segment type of ‘rmap’ is used to indicate Real Mode Memory Mapping for accessing adapter IO space from memory.

Things that don't work like you may expect

- A number of things go into pinned memory and they are hidden from svmon because of how they are configured. Adding up the pinned column of 'svmon –S' will not match 'svmon –G'.
- LPARs allocate certain structures based on the maximum memory and processors allowed. Adapters may also add/use addressable memory that is dedicated. These systems will have more pinned pages than can be identified in 'svmon –S' output. This can be partly accounted for by the difference between 'memory pages' and 'lruable pages' in 'vmstat –v' output. Not all of the pinned pages will be accounted for. This is normal and expected.
- Both of these issues are resolved by accounting for the pages in the 'other' category. The 'other' category is defined as:
  - Number of frames managed by the operating system that are not attached to a VMM segment.

Finding growing segments

- Growing segments are only important for working segments
- Use ‘svmon –S’ and collect 2 samples separated by some time.
- Now, determine which work segments have changed size with:

```
egrep –v “ clnt | pers “ before after | sort +1 | cut –f2-99 –d: | uniq –c | awk ‘$1==1 {print}’
```

before and after are the 2 svmon files.

Indentifying growing segments

```
Count    Vsid     Esid  Type Description          PSize Inuse  Pin Pgsp Virtual

   1     7466  ffffffff work application stack        s    30    0   13    40
   1     7466  ffffffff work application stack        s    31    0   12    40
   1     7a5a        11 work text data BSS heap       s  5549    0  271  5752
   1     7a5a        11 work text data BSS heap       s  5637    0  217  5816
   1     826f         2 work process private          s    16    3    0    16
   1     865e  ffffffff work application stack        s    74    0   18    89
   1     865e  ffffffff work application stack        s    75    0   18    89
   1     9002         - work kernel heap              s 65450    0 5531 65536
   1     9002         - work kernel heap              s 65482    0 5531 65536
```

- The count should always be 1. Otherwise, this line is not unique.
- Please note that segment 826f is a segment created Or deleted between the before and after samples.

Summary

- By understanding the classification of memory segments in AIX, we can understand how memory is used by the kernel, processes and the file system cache
- svmon shows how AIX has classified a memory segment, the page size of the segment, and the processes using the segment
- Not all segments can be associated with a process
- If AVM exceeds real memory, paging is unavoidable
- Working storage pages can be in memory, on paging space, or in both places at the same time
- Note: svmon in kdb/KDB is different

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.