Monitoring 32-Bit Shared Memory and Implementing Memory Windows on HP-UX

David Carmichael

Response Center Engineer
HP Atlanta Response Center

HP WORLD 2003
Solutions and Technology Conference & Expo

Objectives

- Understand when memory windows would be helpful
- Learn the basics of how memory windows work
- Demonstrate memory windows with a simple example
- Provide additional resources for memory windows information

Why do we need Memory Windows?

- All 32 bit applications in the system are limited to a total of 1.75 gigabytes of shared memory, 2.75 gigabytes if compiled as **SHMEM_MAGIC**.
- In a system with 16 gigabytes of physical memory, only 1.75 can be used for 32 bit shared resources. Imagine trying to place multiple distinct database instances on a single machine. The SGA (System Global Area) requirements for all those databases may very well exceed the 1.75 gigabyte limitation.

Why do we need Memory Windows?

Why only 1.75g for shared memory?

- The process address space is divided into four quadrants. On a 32 GB system, these quadrants are each 1 GB. They are 4 TB on a 64 bit system.

- The size of the quadrants limits the amount of address space available to processes. Each quadrant is used to store different types of information for the process.

Why do we need Memory Windows?

The default usage for the quadrants is:

Quadrant 1 Process text
Quadrant 2 Process data
Quadrant 3 Shared libraries
Shared memory
Memory mapped files
Quadrant 4 Shared libraries
Shared memory
Memory mapped files

The last part of quadrant 4 is reserved for system I/O.

Why do we need Memory Windows?

- For 64 bit processes the quadrant sizes are 4tb, and therefore the maximum sizes for text, data, shared memory, etc, are quite large.

- On 32 bit systems each quadrant is 1g. Since q3 and q4 are used for shared memory processes are limited to about 1.75g of shared memory.

What does the Memory Windows feature do?

- A 32-bit process can create a unique memory window for shared objects like shared memory. Other processes can then use this window for shared objects as well.
- The ability to create a unique memory window removes the current system wide 1.75 gigabyte limitation. If a server had 7 Oracle instances each one could potentially have a 1gigabyte SGA.

What does the Memory Windows feature do?

Enabling memory windows does change the kernels default allocation policy.

With only the global memory window configured, the kernel allocates shared addresses from the 4th quadrant first and then the 3rd quadrant.

When **max_mem_window** is set to a non-zero value (enabling memory windows) , the kernel changes the default allocation policy such that the 3rd quadrant is tried first followed by the 4th.

What does the Memory Windows feature do?

- Note that memory windows allows the creation of more than 1.75 gigabytes of total system wide shared memory, but it does not extend how much shared memory a single process can create.

- **SHARED_MAGIC** executables are still limited to 1.75 gigabytes.

Magic number review

- There are 3 magic numbers that can be used for an executable at 11.00/11i. They are **SHARE_MAGIC** (**DEMAND_MAGIC**), **EXEC_MAGIC**, and **SHMEM_MAGIC**.
- For 64 bit 11.00/11i executables there is currently no need to have different memory maps available as the standard one allows up to 4TB for the program text, another 4TB for it's private data and a total of 8TB for shared areas.

SHARE_MAGIC

SHARE_MAGIC is the default at 11.00/11i. **SHARE_MAGIC** is also called **DEMAND_MAGIC**. With **SHARE_MAGIC**, quadrant 1 is used for program text, quadrant 2 is used for program data, and quadrants 3 and 4 are for shared items.

EXEC_MAGIC

EXEC_MAGIC allows a larger process data space by allowing text and data to share quadrant 1. Quadrant 2 is still solely used for data, and quadrants 3 and 4 are also the same as with **SHARE_MAGIC** executables. **EXEC_MAGIC** applications are created by linking the application with the **-N** option

SHMEM_MAGIC

SHMEM_MAGIC makes 2.75 GB of shared memory available to a process. With **SHMEM_MAGIC** all of the text and data is in quadrant 1 freeing up quadrant 2 for shared items. The **SHMEM_MAGIC** processes on the system will share quadrant 2 for shared memory, as well as sharing quadrants 3 and 4 with other processes on the system.

SHMEM_MAGIC

Note:

- Quadrant 2 is only available for system V shared memory segments, while in quadrants 3 and 4, shared libraries and shared memory mapped files compete with shared memory for the 1.75 GB of space.
- Even with **shmem_magic** executables, a single shared memory segment must be contained completely in one quadrant, so 1 GB is still the maximum size of a single shared memory segment.

What is needed to run Memory Windows?

Memory Windows will run on either 32 or 64 bit HP-UX.

Patches to install for 11.00:

- PHKL_28180 Probe,IDDS,PM,VM,PA-8700,AIO,T600,FS,PDC,CLK
- PHCO_23651 fsck_vxfs(1M) cumulative patch
- PHKL_18543 PM/VM/UFS/async/scsi/io/DMAPI/JFS/perf patch

Continued on next slide…

What is needed to run Memory Windows?

- Patches to install: (continued)

- PHCO_25902 cumulative SAM/ObAM patch
- PHKL_27980 VxFS 3.1 cumulative patch: CR_EIEM
- PHCO_23705 11.00 memory windows cumulative patch
- PHCO_20179 Release Notes and Release Notes Addendum

- No patches needed for 11i.

What is needed to run Memory Windows?

To enable the use of memory windows the kernel tunable **max_mem_window** must be set to the desired amount of memory windows. The disabled value is 0. A good default is 256.

Note: This tunable is not dynamic.

What is needed to run Memory Windows?

To enable 256 memory windows in the kernel add the following line to the kernel configuration file (**/stand/system**):

max_mem_window 256

Build a new kernel and reboot.

Setting **max_mem_window** to 256 results in creating 256 memory windows, plus the global memory window. Setting **max_mem_window** to 1 would create 1 memory window, plus the global window, for a total of 2.

What is needed to run Memory Windows?

Modify **/etc/services.window**

.

.

database1 20

database2 30

database3 40

Start a process in a memory window.

- **getmemwindow** is the command used to extract window ids of user processes from the **/etc/services.window** file.
- **setmemwindow** is the command that changes the window id of a running process or starts a specified program in a particular memory window.

Start a process in a memory window.

- An application cannot dynamically switch between one window and another.
- Applications are only allowed to change their window at exec time.
- Applications are not required to change their source code, but instead use the memory window command **setmemwindow** to start an application in a specified memory window.

Start a process in a memory window.

For example:

setmemwindow -i 10 myprog arg1 arg2 arg3

This would start the executable "myprog" with 3 arguments in the memory window corresponding to user key 10. There are other options to setmemwindow to control how memory windows are created.

Start a process in a memory window.

- **getmemwindow** is used to extract the user id's/keys from the **/etc/services.window** file given a particular string.
- This avoids having to embed the keys in the scripts themselves.
- syntax:

getmemwindow <name>

Start a process in a memory window.

Putting it all together in a real example:

```
# cat startDB1.sh
WinId=$(getmemwindow database1)
setmemwindow -i $WinId
   /home/dave/memwinbb/startDB1 "DB1 says hello!"
```

Start a process in a memory window.


```
# ./startDB1.sh
writing to segment: "DB1 says hello!"
Key is 1377312562
```

Viewing the segments

```
# ipcs -mob
IPC status from /dev/kmem as of Thu Jul 10 08:52:34 2003
T     ID     KEY        MODE       OWNER    GROUP NATTCH  SEGSZ
Shared Memory:
m      0 0x4124288b --rw-rw-rw-      root     root          0        348
m      1 0x4e000002 --rw-rw-rw-      root     root          1      61760
m      2 0x41280050 --rw-rw-rw-      root     root          1       8192
…
m   5215 0x52181f32 --rw-r--r--      root      sys          0       1024
```

Our key of 1377312562 converted to hex is 52181f32

Viewing the segments

- **memwin_stats** is the unsupported command to display information about shared memory segments, processes and the memory windows themselves.
- **memwin_stats** is available at:

ftp://contrib:9unsupp8@hprc.external.hp.com/sysadmin/memwin/

Viewing the segments

```
# ./memwin_stats -m -w
Entry USER_KEY KERN_KEY  QUAD2_AVAIL  QUAD3_AVAIL    PID    REFCNT
Memory Windows:
  0    Global          0          262144          243449          0         309
  1   Private          1               0               0          0           1
  2        20          2          262144          262143       7161           1

...
```

Viewing the segments


```
# ./startDB2.sh
writing to segment: "DB2 says Bon Jour!"
Key is 1377312563

# ./startDB3.sh
writing to segment: "DB3 says Ola!"
Key is 1377042734
```

Viewing the segments

```
# ./memwin_stats -m –w
Entry        USER_KEY KERN_KEY  QUAD2_AVAIL  QUAD3_AVAIL    PID
   REFCNT
Memory Windows:
  0   Global      0        262144      243449          0          309
  1   Private     1          0           0             0          1
…
  2      20       2        262144      262143         7161        1
  3      40       3        262144      262143         7251        1
  4      30       4        262144      262143         7270        1
…
```

Viewing the segments

```
# more stopDB1.sh
WinId=$(getmemwindow database1)
setmemwindow -i $WinId /home/dave/memwinbb/stopDB1
# ./stopDB1.sh
# ./stopDB2.sh
# ./stopDB3.sh
```

Viewing the segments

```
# ./memwin_stats -m
Shared Memory:
T     ID     KEY        MODE       OWNER    GROUP   UserKey   KernId
m      0 0x2f100002 --rw-------      root      sys          Global      0
m      1 0x411c36c1 --rw-rw-rw-      root     root          Global      0
m      2 0x4e0c0002 --rw-rw-rw-      root     root          Global      0
m      3 0x41203058 --rw-rw-rw-      root     root          Global      0
m      4 0x0c6629c9 --rw-r-----      root     root          Global      0
m      5 0x06347849 --rw-rw-rw-      root     root          Global      0
```

Viewing the segments

shminfo is an unsupported HP utility which provides information about shared memory segments, processes and memory windows.

shminfo is available only by request from the HP Response Center

Common **shminfo** Uses

- Examine shared memory quadrant fragmentation.
- Examine memory window usage information.
- Isolate what processes are attached to a given identifier. Equate shared memory id to memory quadrant and virtual memory address.

Viewing the segments

A small portion of **shminfo** output:

Shared space from Window id 20:

```
        Space      Start        End  Kbytes Usage
Q2 0x0e0c6000.0x40000000-0x7fffffff 1048576 FREE
Q3 0x0eea2800.0x80000000-0x80000fff       4 SHMEM
   id=5215
Q3 0x0eea2800.0x80001000-0xbfffffff 1048572 FREE
```

References

- Memory Windows White Paper
  http://docs.hp.com/hpux/onlinedocs/os/memwn1_4.pdf

  /usr/share/doc/mem_wndws.txt on 11.00 systems
- PHKL_18543 patch text
- ITRC doc rcfaxmemory001 (magic numbers)
- Man pages: getmemwindow(1), services.window(4), setmemwindow(1), README_memwin_stats

Thanks for attending!

Feel free to contact me with any questions.

david.carmichael@hp.com

Interex, Encompass and HP bring you a powerful new HP World.