OpenVMS Technical Journal V5

Removing the 32-Bit Limit from System Space (OpenVMS V6.1 -- V7.3-2)

Removing the 32-Bit Limit from System Space (OpenVMS V6.1 -- V7.3-2)

By Dan Buckley, Consultant, HP Services

Overview

With OpenVMS VAX Version 6.0 and earlier systems, 32-bit VAX address space was divided into four 1-GB sections called S0, S1, P0, and P1. S0 space was designated as system space. This 1-GB limit on system space could prevent a system from having a large number of users while also having large working sets, virtual address space, memory disks, nonpaged pool, and LBN caches. If your usage attempted to exceed the amount of system space available, your system would not boot.

As systems grew larger with more CPUs and huge amounts of memory (16 GB is common now, whereas a MicroVAX II was limited to 16 MB), the 32-bit limit prevented systems from effectively using all their memory.

This article tracks the changes made to OpenVMS over the years to expand system space, remove large users of system space, and reduce the size of the remaining users of system space.

Background

With OpenVMS VAX Version 6.0, system space was limited to the 1 GB S0 space, and S1 space was not used by OpenVMS. The largest users of system space were balance slots, nonpaged pool, memory disks, and VIOC cache. Each of these system space consumers is described in the following paragraphs.

Balance Slots

Balance slots use system space to describe all the possible addresses that a process can use. Each balance slot includes space for a working set list of WSMAX entries to describe the maximum physical pages, and page tables of VIRTUALPAGECNT entries to describe the maximum virtual address space. Each balance slot can hold the maximum size process header (PHD). Since large numbers of working set list and virtual page table entries are needed to run a large program in memory without incurring paging overhead, the space required to hold all of these entries can be very large.

On a VAX, the maximum values for the system parameters WSMAX and VIRTUALPAGECNT are 1048576 and 4194304, respectively. With 4 bytes per entry on a VAX, the largest possible balance slot is 4*(1,048,576 + 4,194,304) = 20,971,520 or 20 MB. This means that even in the hypothetical instance where there are no other users of system space, a VAX running Version 6.0 could have only 50 balance slots of the maximum size.

If the number of processes on the system (MAXPROCESSCNT) exceeds the number of processes in memory (BALSETCNT), swapping can occur no matter how much free memory is on the system. OpenVMS VAX Version 6.0 did attempt to reduce this swapping by using virtual balance slots (enabled by the VBSS_ENABLE system parameter), but system slowdowns and hangs were still possible with virtual balance slots enabled if all real balance slots were in use.

Nonpaged Pool

Nonpaged pool is used to hold many OpenVMS and application data structures. These include, but are not limited to, locks, resources, unit control blocks, I/O request packets (IRP), timer queue entries, window control blocks, access control lists, and VAX communication request packets (VCRP). Nonpaged pool is very closely tied to OpenVMS and most likely will never move out of system space.

The system parameter NPAGEDYN specifies the initial allocation of pool on boot and NPAGEVIR specifies the maximum to which nonpaged pool can expand. System space usage for nonpaged pool is one page for each page in NPAGEVIR, so you can gain back some system space on systems where nonpaged pool does not grow by reducing the value of NPAGEVIR to equal the value of NPAGEDYN (or 2*NPAGEDYN). As shown in **Example 1**, you can monitor nonpaged pool usage and expansion by executing the DCL command SHOW MEMORY/POOL/FULL and comparing the current value to the initial size and maximum size. If these values are equal, look at the free space.

```
$ SHOW MEMORY/POOL/FULL
System Memory Resources on 13-AUG-2004 10:15:10.02
Nonpaged Dynamic Memory      (Lists + Variable)
    Current Size (bytes)       2999808    Current Total Size (pages)    5859
    Initial Size (NPAGEDYN)    2999808    Initial Size (pages)          5859
    Maximum Size (NPAGEVIR)   11999744    Maximum Size (pages)         23437
    Free Space (bytes)          553600    Space in Use (bytes)       2446208
    Size of Largest Block        77568    Size of Smallest Block          64
    Number of Free Blocks          426    Free Blocks LEQU 64 Bytes       12
    Free Blocks on Lookasides      331    Lookaside Space (bytes)     128960
```

Example 1 - Using SHOW MEMORY to Monitor Nonpaged Pool Usage

A system's usage of nonpaged pool depends on how the system is set up and how the applications are being run. If you run out of nonpaged pool (expand to NPAGEVIR), the system will crash or hang while trying to allocate nonpaged pool.

Memory Disks

Memory disks (also known as DECram disks) use system memory as a pseudo-disk to greatly increase disk throughputs and decrease file access times. The cost of a memory disk is about 1 page (pagelet on Alpha or I64) of memory per disk block and 1 system page table entry (SPTREQ) for each 16 blocks of disk space under DECram V2.1. Each SPTREQ requires 4 bytes of system space on a VAX, so a large memory disk can consume significant amounts of system space. Putting frequently accessed files onto a memory disk can significantly improve application performance, but large DECram disks can run into system space limits. The use of system space was reduced with DECram V2.2, which uses only 1 SPTREQ for each 16K DECram disk blocks.

Virtual I/O Cache (VIOC)

Virtual I/O cache (VIOC) or logical block number (LBN) cache is a cache of blocks recently read from disk. If another read is made to an LBN that is already in cache, the system does not have to go to disk to get the data, and the associated delay to perform the I/O operation is avoided. The larger the VIOC, the more likely a read hit will occur (some applications do not benefit from read caches), so large caches can produce significant performance improvements. Each VIOC page consumes one page of nonpaged pool and is shown as a VCC packet in the output display from the SDA command SHOW POOL/SUMMARY.

Changes Made to OpenVMS to Address System Space Limitations

OpenVMS Version 6.1 and Later

In OpenVMS VAX Version 6.1 and on all versions of OpenVMS Alpha and OpenVMS I64, the unused S1 address space was combined with S0 space to create a 2-GB system address space called S0/S1 space. VAX machines that do not support extended addressing (XA) remain limited to the 1-GB S0 system space.

OpenVMS Version 7.0

OpenVMS Version 7.0 introduced 64-bit addressing and a new, very large, S2 address space. S2 space defaults to 6 GB, but can be increased by tweaking the SYSGEN parameter S2_SIZE. For more information about the virtual address space layout, refer to Section 1.6 of *OpenVMS Alpha Internals and Data Structures*, Memory Management volume, by Ruth Goldenberg.

Most changes in OpenVMS Alpha (and subsequently in OpenVMS I64) to reduce the use of system space take advantage of these new features by moving data structures from S0/S1 space to S2 space or by moving large users of nonpaged pool to S2 space.

As noted in the following OpenVMS Version 7.0 release note, virtual page tables were removed from balance slots, reducing their size in system space.

Starting with OpenVMS Alpha Version 7.0, VIRTUALPAGECNT ceases to be a tunable parameter on Alpha systems and is no longer used to specify the virtual size of a process. The process page tables have migrated from system space to a dedicated page table space that is guaranteed to be large enough to accommodate the virtual address space provided by the system. This migration has rendered the parameter obsolete, and OpenVMS Alpha ignores its contents entirely.

OpenVMS Version 7.1

Memory resident global sections allow an entire global section to be mapped at boot time, and the pages in the section are not charged against a process's working set. This allows for programs with very large global sections to run on machines with a reduced value for WSMAX, which in turn reduces the amount of system space required for balance slots in system space.

OpenVMS Version 7.2

Lock manager data was moved from nonpaged pool to pool zones in S2 space. Two of the largest users of nonpaged pool prior to Version 7.2 were lock blocks (LKB) and resource blocks (RSB). Moving the data used by the distributed lock manager out of nonpaged pool greatly reduced the amount of NPAGEDYN used on most AlphaServer systems. This reduced pool usage could then be used to lower the value of NPAGEVIR and reduce system space usage.

OpenVMS Version 7.3

Extended File Cache (XFC) in S2 space replaced VIOC in S0/S1 space for LBN cache. Prior to the release of XFC, the largest LBN cache you could make on an AlphaServer was less than 1.5 GB -- no matter how much memory was on the system. With XFC using S2 space, the size of cache is limited only by the amount of memory in the system. LBN caches of 4 GB and larger are common, and these larger LBN caches require no system space.

OpenVMS Version 7.3-1

Preallocated floating-point registers and execution data (FRED) addresses were removed from system space. Prior to this change, either 16 FRED blocks (in Version 7.2) or 256 FRED blocks (in Version 7.2 and later) were reserved in each balance slot at process creation, even if the process did not become multithreaded. Starting with Version 7.3-1, FRED blocks are allocated when a process calls $INIT_MULTITHREADING. Only the primary kernel thread's FRED is in the PHD, thus reducing the size of the balance slots.

OpenVMS Version 7.3-2

Working set list pages were moved to S2 space, reducing the size of each balance slot to 1 page. **Examples 2 and 3** demonstrate the difference in memory usage and the size of balance slots between OpenVMS Version 7.3 and Version 7.3-2. Both examples show output in response to the SDA command CLUE MEMORY/LAYOUT. In Example 3, note the new location of the working set list (WSL), which has been moved outside S0/S1 space, and the reduced size of the balance slots (now just 1 page each).

```
System Virtual Address Space Layout:
------------------------------------
   Item                                    Base               End           Length
...
287 Balance Slots, 29 pages each  FFFFFFFF.8366A000  FFFFFFFF.87770000  04106000
```

Example 2 – Memory Usage on OpenVMS Version 7.3 (WSMAX = 196608)

```
System Virtual Address Space Layout:
------------------------------------
   Item                                  Base               End            Length
System Virtual Base Address       FFFFFEFE.00000000
PFN Database                      FFFFFEFE.00000000  FFFFFEFE.00280000  00280000
188 WSL Slots, 17 pages each      FFFFFEFE.00280000  FFFFFEFE.01B78000  018F8000
Permanent Mapping of System L1PT  FFFFFEFE.01B78000  FFFFFEFE.01B7A000  00002000
...
to the beginning of system space
Execlet Code Region               FFFFFFFF.80000000  FFFFFFFF.80800000  00800000
...
188 Balance Slots, 1 page each    FFFFFFFF.828A0000  FFFFFFFF.82A18000  00178000
```

Example 3 - Memory Usage on OpenVMS Version 7.3-2 (WSMAX = 262144)

Summary

The changes made to OpenVMS Alpha from Version 7.0 through Version 7.3-2 (and included in later versions of OpenVMS Alpha and OpenVMS I64) have made the S0/S1 system space limitation of 2 GB a nonissue. The biggest user of system space is now nonpaged pool, but most of the large users of nonpaged pool have been removed. By setting SYSGEN parameter S2_SIZE to reflect the needs of your system, you are limited only by the amount of physical memory in the system, which can be as high as 512 GB in a 64-processor AlphaServer GS1280. Of course, twenty years from now, folks will shake their heads and mutter "ONLY 512 GB of memory…!" with the same disbelief we have now when contemplating a 16-MB MicroVAX II.

For More Information

OpenVMS Alpha Internals and Data Structures, Memory Management volume, by Ruth Goldenberg (ISBN:1-55558-159-5).

The *Release Notes* and *New Features* manuals for your version of OpenVMS.

About the Author

Dan Buckley joined Digital in 1980 and started working on OpenVMS in 1983 on a VAX 750 running OpenVMS Version 3.2. He has been supporting OpenVMS in the Customer Support Center since 1995.