OpenVMS Technical Journal V14

OpenVMS – Paged Dynamic Memory Fragmentation Causing Performance Problems

John Hockett, HP Services Technology Consultant

Introduction

When customers added additional users to their environment, they started receiving complaints related to response time performance problems. The number of complaints increased as the time since the last reboot increased. After a reboot, the performance was acceptable for part of a normal production day. The problems were seen only by users who were running a large custom application. Typically, users running other applications and DCL commands did not see any performance problems. The users were experiencing random pauses of 3 to 5 seconds, and often multiple processes were seen in a MUTEX wait state for short intervals as shown in Figure 1.

```
$ SHOW SYSTEM
OpenVMS V7.3-2 on node L 26-OCT-2005 13:46:02.97
Pid   Process Name State Pri I/O     CPU   Page flts  Pages
:
2892F00C _TNA6942: MUTEX 6 58122 0 00:00:31.32 36814  814
2885C06C _TNA8017: MUTEX 6   491 0 00:00:03.33   592  714
2890C9D2 146811    MUTEX 6    87 0 00:00:00.83   123  112
288F7203 _TNA8023: MUTEX 5    81 0 00:00:01.96   125  116
2890BA06 _TNA8024: MUTEX 6    87 0 00:00:00.72   129  116
2890920F RA_127    MUTEX 3     0 0 00:00:00.00    20   30 S
288F6211 MS_66     MUTEX 6    15 0 00:00:00.00   123  134 S
2891BA13 MAINT_CAR MUTEX 6     0 0 00:00:00.00    20   30 S
2891AB04 _TNA7911: MUTEX 6 10016 0 00:00:10.04  7191 1412
28932E42 _TNA8009: MUTEX 6 32810 0 00:00:14.61  1231  763
:
```

Figure 1: SHOW SYSTEM

Analysis

After reviewing the collected T4 data from node L, it was determined that each time an MWAIT spike (Blue) occurred, there was also a similar Kernel Mode spike (Red) just before and throughout the MWAIT spike.

Note: A process in a Mutex wait state is considered an MWAIT process. See Figure 2 below.

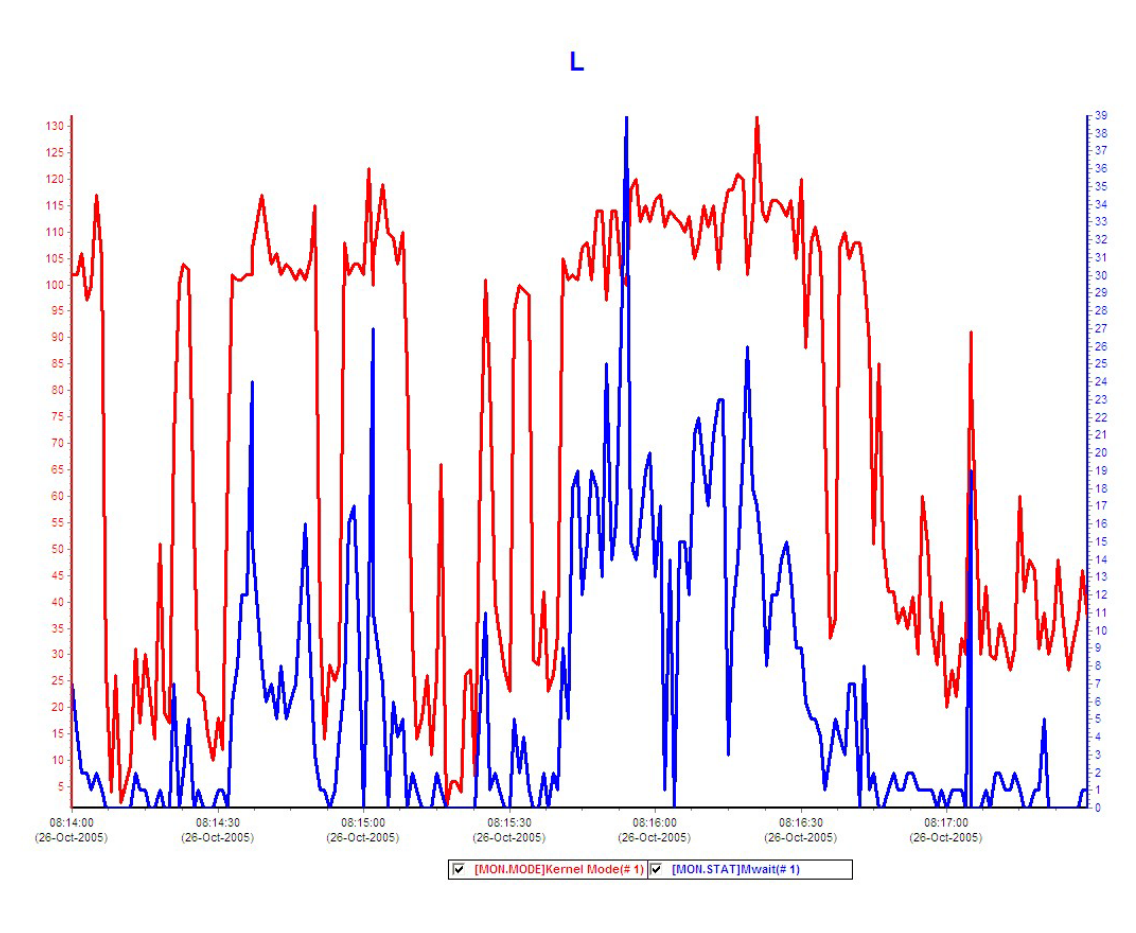

Figure 2: KERNEL mode and processes in MUTEX WAIT

A process running in Kernel Mode owns a MUTEX causing the number of other processes waiting for the same MUTEX to spike. Normally, the time spent in Kernel Mode holding a MUTEX is very short, but there is evidence to show that some Kernel Mode operations are taking longer than expected, and thus causing the other processes, waiting for the same MUTEX, to spike at times.

In the following example (Figure 3), process 288E3D5A ran continually from 10:11:54.174601 to 10:11:58.000763 with most PC samples in EXE$DEALLOCATE_C+0001C or EXE$DEALLOCATE_C+00020, and an occasional LOGICAL_NAMES+00288. During this time, other processes waiting for the same MUTEX are placed in a MUTEX wait state.

```
$ ANALYZE/SYSTEM
SDA> PCS LOAD
SDA> PCS START TRACE
      wait 10 minutes
SDA> PCS STOP TRACE
SDA> SET OUTPUT PC.DAT
SDA> PCS SHOW TRACE
SDA> PCS UNLOAD
SDA> EXIT
$ EDIT PC.DAT
      :

Timestamp              CPU PC       IPL Pid      Routine
Module
---------------------- --- -------- --- -------- ------------------------
-------- ------
31-OCT 10:11:54.000216  03 80124F74  3  00000000
SCH$CALC_CPU_LOAD_C+00464        PROCESS_MANAGEMENT+00000F74
      :
31-OCT 10:11:54.174601  03 8003FCFC  2  288E3D5A EXE$DEALLOCATE_C+0001C
SYSTEM_PRIMITIVES_MIN+00013CFC
      :  1866 total PC samples in this collection from the same PID
         that contained:
           EXE$DEALLOCATE_C 1808 samples
           LOGICAL_NAMES 58 samples
31-OCT 10:11:58.000763  03 8003FD10  2  288E3D5A EXE$DEALLOCATE_C+00030
SYSTEM_PRIMITIVES_MIN+00013D10
      :
```

Figure 3: System PC samples summarized

The System Dump Analyzer (SDA) MTX tool is used to determine which MUTEX has the highest usage. In Figure 4, the MUTEX with the highest usage is the LNM MUTEX (logical name MUTEX).

```
$ ANALYZE/SYSTEM
SDA> MTX LOAD
SDA> MTX START TRACE
SDA> MTX SHOW TRACE/SUMMARY

Mutex Trace Information:
--------------------------------------------------------
           Read  Read Lock Write  Write Lock
           Locks   Waits   Locks    Waits  Wait  Unlocks
Mutex      /sec    /sec    /sec     /sec    %     /sec
---------- ------ -------  ----- -------  ------  ------
LNM        1664.7 602.6    145.2   188.6   43.7%  1810.0
IODB        382.4   4.1    324.0    50.8    7.8%   706.4
CEB           0.0   0.0     13.6     0.0    0.0%    13.6
PGDYN         0.0   0.0    159.6     0.3    0.2%   159.5
GSD           0.0   0.0     52.4     2.2    4.2%    52.4
CIA           0.0   0.0      0.1     0.0    0.0%     0.1
ORB           0.0   0.0      0.0     0.0    0.0%     0.0
VOL          74.0   0.0      0.0     0.0    0.0%    74.0
OBJCLS       17.2   0.0      0.0     0.0    0.0%    17.2
RSDM          0.0   0.0      1.7     0.0    0.0%     1.7
???          53.7   0.0      0.2     0.0    0.0%    53.9
           ------ -------  -----  ------  ------  ------
           2192.1 606.8    696.8   241.9   29.4%  2888.8
...
Mutex         Rate/sec  Operation         Callers PC
Module                      Offset
----------  ----------  ---------------   ------------------------------------  -
---------------------------------

LNM           2157.3  Lock Read         801FFFAC  LOGICAL_NAMES+07FAC
LOGICAL_NAMES           00007FAC
LNM            289.9  Lock Write        801FD7BC  LOGICAL_NAMES+057BC
LOGICAL_NAMES           000057BC
LNM            109.2  Lock Read         801F8FA8  LNM$SEARCH_ONE_C+00068
LOGICAL_NAMES           00000FA8
LNM             23.4  Lock Write        801F9314  LNM$LOCKW_C+00024
LOGICAL_NAMES           00001314
LNM             20.6  Lock Write        801FF284  LOGICAL_NAMES+07284
LOGICAL_NAMES           00007284

IODB           144.5  Lock Read         800F5EF8  IO_ROUTINES+23EF8
IO_ROUTINES             00023EF8
IODB           127.1  Lock Read         800F6920  EXE$DVI_FREEBLOCKS_C+00650
IO_ROUTINES             00024920
IODB           126.3  Lock Write        800ED164  IO_ROUTINES+1B164
IO_ROUTINES             0001B164
IODB           125.7  Lock Write        800EAEC8  IOC_STD$CREATE_UCB_C+00F68
IO_ROUTINES             00018EC8
IODB           111.4  Lock Read         800F5E68  IO_ROUTINES+23E68
IO_ROUTINES             00023E68

CEB             13.6  Lock Write        80156B34  EXE_STD$CHKWAIT2_C+00DF4
PROCESS_MANAGEMENT      00032B34

PGDYN          109.6  Lock Write        8003F604  EXE$ALOPAGED_C+00074
SYSTEM_PRIMITIVES_MIN   00013604
PGDYN           50.3  Lock Write        8003FB54  EXE$DEAPAGED_C+00094
SYSTEM_PRIMITIVES_MIN   00013B54
```

Figure 4: MUTEX usage

After observing the system following a reboot, it was evident that the Paged Dynamic Memory (PAGEDYN) free list fragmentation increased the longer the system was up. As the fragmentation increased, the Kernel Mode time and the number of processes in MUTEX wait state increased. Once the PAGEDYN freelist exceeded 10,000 free blocks, the pauses started becoming noticeable to the users.

The high LNM MUTEX utilization was presented to the customer and they were able to identify that their applications used the Job Logical Name Tables quite extensively. Working with the customer, we determined that their production applications frequently execute thousands of Logical Name creations and deletions in the Job Logical Name Table. The data structures that support the Job Logical Name Tables (LNMx) are allocated from PAGEDYN.

PAGEDYN has always been organized as a singly linked list.
Each time new LNMx structures are to be allocated from PAGEDYN, the LOGICAL_ NAMES code acquires the LNM MUTEX and also requests the memory allocation code to traverse the PAGEDYN linked list, one data structure at a time until the proper sized LNMx structure can be allocated from PAGEDYN. The LOGICAL_ NAMES code then adds the LNMx to an existing Job Logical Name Table or creates a new Job Logical Name Table.

When a LNMx is to be returned, the LOGICAL_NAMES code must also acquire the LNM MUTEX and remove the LNMx structure from the Job Logical Name Table and call the memory deallocation code to traverse the PAGEDYN linked list one data structure at a time until the proper place is determined to insert the returned LNMx structure.

If PAGEDYN has become fragmented into several pieces, this can cause users to experience performance problems while the singly linked list is being traversed. During the time that it takes to allocate or return a LNMx, other processes that want to allocate or delete logical names or logical name tables go into a MUTEX wait state.

After reviewing the customer's system, it was determined that PAGEDYN was very fragmented. In the example shown in Figure 5, there are 34,826 fragments. Most of the fragments are less than 64 decimal bytes in length:

```
$ SHOW MEMORY/POOL/FULL
      :
Paged Dynamic Memory
Current Size (MB)          143.04 Current Size (Pagelets)     292960
Free Space (MB)             84.24 Space in Use (MB)            58.79
Largest Var Block (MB)      81.30 Smallest Var Block (bytes)   16.00
Number of Free Blocks       34826 Free Blocks LEQU 64 bytes    33533
```

Figure 5: SHOW MEMORY output

Therefore, a potential allocation or deallocation of an LNMx (or some other packet from PAGEDYN) could require, in a worst case scenario, "walking" a list of 34,000 packets before the allocation or deallocation request could be completed.

Solution

To eliminate the performance problem, OpenVMS engineering designed and provided new versions of SYSTEM_PRIMITIVES*.EXE,*.STB

This new code implements PAGEDYN lookaside lists similar to the lookaside lists that have been implemented in Nonpaged Dynamic Memory for years. This new code is implemented in the MEMORYALC routine of the SYSTEM_PRIMITIVES image. The MEMORYALC routine does the actual allocation and deallocation of Paged Dynamic Pool from the new lookaside lists.

Like non-paged pool, the PAGEDYN lookaside lists start out empty. As packets are deallocated, the size is checked and the packet is returned to the appropriate list. If the packet is larger than RSVD_EXEC_1 (prior to VMS 8.4) or PAGED_LAL_SIZE (VMS 8.4 and later), the packet is returned to the variable paged pool. When an attempt is made to allocate a packet less than or equal to RSVD_EXEC_1 or PAGED_LAL_SIZE, the appropriate PAGEDYN lookaside list containing packets of that size is checked. If a packet is found, it is removed and returned to the caller without having to acquire the PGDYN MUTEX. If a packet is not found, the PGDYN MUTEX will be acquired and the packet will be allocated from the variable free list as had been traditionally done.

When RSVD_EXEC_1 or PAGED_LAL_SIZE is non-zero, PAGEDYN lookaside lists are enabled and the maximum packet size to use on the paged pool lookaside list is established. It can range between 1 - 2560 decimal bytes (the packets themselves have 16 byte granularity, 16, 32, 48, 64, etc). For most systems, setting RSVD_EXEC_1 or PAGED_LAL_SIZE to 512 bytes is more than adequate. The most utilized PAGEDYN lookaside lists are 80 to 208 bytes. Also, as part of further performance enhancements, the PGDYN MUTEX is not used if a packet is found on or returned to one of the PAGEDYN lookaside lists.

If the requested PAGEDYN lookaside lists is empty or the packet request is larger than the SYSGEN parameter setting, the PGDYN MUTEX is acquired and memory is allocated from the PAGEDYN singly linked variable size free list as it was done in past implementations.

If there is a PAGEDYN shortage, the new code will reclaim memory from the PAGEDYN lookaside lists and return it to the singly linked variable size freelist.
This new code can be enabled in SYSTEM_PRIMITIVES images dated after 14-NOV-2008 by setting the SYSGEN parameter RSVD_EXEC_1 to a nonzero value. When RSVD_EXEC_1 is 0, the default value, this feature is disabled. When RSVD_EXEC_1 is nonzero (1 - 2650), it establishes the size of the largest paged pool lookaside list to use. For example, setting RSVD_EXEC_1 to 512 would create multiple lookaside lists for packets sized up to 512 decimal bytes and should be adequate for most systems.

Remedial kits for VMS 7.3-2, V8.2, V8.2-1, V8.3, and V8.3-1H1 containing a SYSTEM_PRIMITIVES.EXE dated on or after 14-NOV-2008 will contain this new functionality.
The following remedial kits or a later version of these kits contain this new functionally:

- VMS831H1I_SYS-V0400
- VMS83I_SYS-V0900
- VMS821I_SYS-V0900
- VMS83A_SYS-V1100
- VMS82A_SYS-V1200
- VMS732_SYS-V1800

The Paged Dynamic Lookaside Lists are expected to be fully implemented with the 8.4 release of OpenVMS. It will be enabled by setting the SYSGEN parameter PAGED_LAL_SIZE to a nonzero value. The default PAGED_LAL_SIZE setting of zero sets the default behavior as it has been for decades, a singly linked variable size free list.

For more information, refer to the release notes for each of the above mentioned kits. The kits are available at the following websites:
http://www.itrc.hp.com or ftp://ftp.itrc.hp.com/

Acknowledgement

Thanks to Mark Morris for the design, implementation, and description of the changes to the SYSTEM_PRIMITIVES images. Thanks to Kevin Jenkins, Tom Cafarella, and Jean Pierre Corre for their assistance with the initial problem definition. Finally, thanks to Mark Morris, Rob Eulenstein, Ted Saul, and John Fisher for their review of this article.