OpenVMS Performance & Availability Updates

Rob Eulenstein – HP Enterprise
CSS ERT (Converged Systems & Solutions Engineering Resolution Team)

Agenda/Outline

- V8.4 - DISMOUNT Keyword in HBMM Policies
- Loss of S0/S1 Space due to many KPBs
- Eliminate Unused NICs; Better for PEDRIVER and for S0/S1 Space
- Use Paged Dynamic Memory Lookaside Lists (PAGED_LAL_SIZE)
- An Interesting Tidbit from Console Output during Boot
- How does AUTOGEN handle Large Memory Systems?
- A Couple of DVE (Dynamic Volume Expansion) Quirks
- Remember to use Host-Based Mini-Copy when possible
- Caution When Setting EFIFCScanLevel
- A Forced Crash due to a CPU Bottleneck
- Halt Testing on Integrity Servers

Agenda/Outline - continued

- Adding RMS Global Buffers to a file should help performance, right?
- The RMS Directory Path Cache and deleting .DIR files
- Locating Multinet Code in S0/S1 Space in a Crash Dump
- A Multinet issue when using DECnet over IP
- Shadow Sets with Zero Working Members
- Unusual SYS$EXAMPLES:RAD.COM Output
- Root RSB Lock Activity Counters: Are those Locking Rates Correct?
- CPUSPINWAIT crashes during Lock Directory Rebuild Operations
- A Troubleshooting Tip; Always Consider Everything that has Changed
- A V8.4-1H1 Installation Log
- i2 blade versus i4 blade Performance Tests

V8.4 - DISMOUNT Keyword in HBMM Policies

In V8.4 a new keyword was available in HBMM (host-based mini-merge) policies. This new keyword was DISMOUNT. Use of the DISMOUNT keyword enabled multiple HBMM bitmaps to be converted into mini-copy bitmaps when a shadow set member was dismounted, thus removing the single point of failure that existed prior to V8.4 when using DISMOUNT/POLICY=MINICOPY.

Earlier this year a problem was found when using the DISMOUNT keyword, which under certain circumstances, could lead to data inconsistencies between shadow set members. The problem was reproduced in-house and OpenVMS engineering has made the VMS84x_SHADOWING-V0300 patch kits available to correct this behavior.

Note: The VMS84x_SHADOWING-V0300 patch kits require VMS84x_PCSI-V0400 and VMS84x_UPDATE-V1000 or later as prerequisites.

Note: Traditional mini-copy bitmaps do NOT have this problem.

Loss of S0/S1 Space due to many KPBs

From engineering analysis it was found that by default, for DK/DG devices a new KPB (kernel process block) for unit initialization was being allocated irrespective of whether there was a reusable KPB in the RAD cache or not. While allocating a KPB for any device, a comparison happens between the input stack size parameter and the global stack size value. If the input parameter is less, then a new KPB is directly getting allocated. For DK/DG devices, the stack size was not sent as an input parameter so OpenVMS was using the default stack size value which is less than global stack size which in turn resulted in the allocation of a new KPB for every DK/DG device.

Engineering has fixed the issue by making necessary changes in the KPB allocation code to consider the presence of KPBs in the RAD cache and thereby limiting the KPB allocation.

Customers with several hundred (or several thousand) DGA devices and multiple paths to each, were losing hundreds of MBs of S0/S1 space due to this behavior. New SYSTEM_PRIMITIVES are now available to correct this behavior.

Loss of S0/S1 Space due to many KPBs

For a couple of customers, a small increase in the SYSGEN parameter KSTACKPAGES rendered the system unbootable due to the kernel stack that is associated with each KPB. Each of these kernel stacks (along with two additional 8K pages that are used as guard pages) are located in allocatable S0/S1 space. The KPBs come from non-paged pool, but the associated stacks are allocated from S0/S1 space.

One site increased KSTACKPAGES from the default of 5 pages to 10 pages. That is only an additional five pages (five 8K pages) per stack, but if we do some math, 5 times 8K per additional page times a bunch of KPBs (say 3,300 KPBs as was seen in one boot-time INCONSTATE bugcheck) yields about 135 MB of additional S0/S1 space. If the system was tight on S0/S1 space prior to the change (as was the case at this site), a modest increase in KSTACKPAGES could very well (and did) push the system over the edge.

The following boot-time INCONSTATE bugcheck is one such example.

Loss of S0/S1 Space due to many KPBs

```
SDA> clue crash

Crashdump Summary Information:
------------------------------
Crash Time:         13-DEC-2014 20:31:36.91
Bugcheck Type:      INCONSTATE, Inconsistent I/O data base
Node:               XXXX    (Cluster)
CPU Type:           HP BL890c i2  (1.73GHz/6.0MB)
VMS Version:        V8.4
Current Process:    STACONFIG
Current Image:      STACONFIG.EXE
Failing PC:         FFFFFFFF.80FC96B0    DK_UNIT_INIT_C+007F0
Failing PS:         00000000.00000800
Module:             SYS$DKDRIVER    (Link Date/Time:  5-FEB-2013 15:34:39.62)
Offset:             00014EB0

Boot Time:          13-DEC-2014 20:29:55.00
System Uptime:                0 00:01:41.91
...
    R7                =  00000000.0000003A
    R8                =  00000000.00002044
    R9                =  00000000.00000324
...

SDA> eval/cond 2044
%SYSTEM-F-INSFSPTS, insufficient SPTEs available
```

Loss of S0/S1 Space due to many KPBs

```
SDA> show rad/all

Resource Affinity Domains
-------------------------

    RAD information header address: FFFFFFFF.CD06CE00
    Maximum RAD count:                        0000000A
    Base RAD:                                 00000008
    Alternate base RAD:                       00000009
.
.
.
Resource Affinity Domain 0008
-----------------------------

RAD data:

  Database address:      FFFFFFFF.D7FA6000   Length:                   00001088
  Count of cached PCBs:           00000001   Count of cached KPBs:     00000CDD
  Zeroed list count:              0001000A   Zeroed list limit:        00000000



SDA> eval cdd
Hex = 00000000.00000CDD   Decimal = 3293
```

Loss of S0/S1 Space due to many KPBs

```
SDA> clue memory/layout

System Virtual Address Space Layout:
------------------------------------
   Item                                  Base                End           Length
System Virtual Base Address        FFFFF802.00000000
EFI/PAL/SAL Memory                 FFFFF802.00000000  FFFFF802.37400000  37400000
PFN Database                       FFFFF802.37400000  FFFFF802.BF400000  88000000
Kernel Promote VA                  FFFFF802.BF400000  FFFFF802.BF416000  00016000
.
.
.
Executive Mode Data Page           FFFFFFFF.D7FB2000  FFFFFFFF.D7FB4000  00002000
System Space Expansion Region      FFFFFFFF.FFDF0000  FFFFFFFF.FFDF0000  00000000  <-- Zero Free
System Page Table Window           FFFFFFFF.FFDF0000  FFFFFFFF.FFFF0000  00200000
N/A Space                          FFFFFFFF.FFFF0000  FFFFFFFF.FFFFFFFF  00010000


SDA> show param kstackpages

   Parameter                 Variable                 Address    Value    (decimal
---------------    --------------------------------  --------  --------  --------
KSTACKPAGES        SGN$GL_KSTACKPAG                  CD020C98  0000000A        10
```

The system manager lowered KSTACKPAGES from 10 back down to 5 and the system booted without a problem.

Eliminate Unused NICs; Better for PEDRIVER and Better for S0/S1 Space

There is a savings in S0/S1 space when you exclude unused LAN devices from the configuration via SYSMAN> IO SET EXCLUDE. For those sites tight on S0/S1 space, every NIC excluded from the configuration saves about 10 MB of S0/S1 space. If a site could drop from 26 NICs down to say 6 NICs, they could save around 200 MB of S0/S1 space. An example command to exclude EWX, EWY and EWZ would be:

SYSMAN> io set exclude =(ewx*,ewy*,ewz*)

In cluster environments, in addition to the savings of S0/S1 space, PEDRIVER has an easier time managing fewer NICs. Accordingly to OpenVMS engineering 4 to 5 paths for cluster communication is optimal. I have seen CLUEXIT bugchecks and forced crashes in which the only “smoking gun” is 26 NICs, all running SCA . If we do some math, 26 times 26 times the number of other cluster nodes can very quickly result in a large number of potential channels that PEDRIVER must track for the ECS (equivalent channel set).

Use Paged Dynamic Memory Lookaside Lists (PAGED_LAL_SIZE)

Paged dynamic memory lookaside lists were introduced in V8.4 and made available in earlier OpenVMS versions through patch kits. In the VMS84x_SYS-V0400 kits and later SYS kits for V8.4, the default for the PAGED_LAL_SIZE SYSGEN parameter became 512 thus enabling paged dynamic memory lookaside lists by default, but still we see systems with this feature disabled.

These sites may not run AUTOGEN or they may have PAGED_LAL_SIZE hardcoded to zero in MODPARAMS.DAT. In addition, we have heard responses from customers that indicate paged dynamic memory (PAGEDYN) is not used nearly as much as non-paged dynamic memory (NPAGEDYN / NPAGEVIR), why should I bother enabling this feature?

On the next slide I show data from a crash dump in which the rx2800 server had been up for just 4.5 days. Notice the number of allocations from non-paged pool compared to the number of allocations from paged pool. Our recommendation is to enable this feature on all systems, or at a minimum, check the allocation rates on the running system via SDA> CLUE MEMORY/STATISTICS.

Use Paged Dynamic Memory Lookaside Lists (PAGED_LAL_SIZE)

```
SDA> clue mem/stat
Memory Management Statistics:
-----------------------------
Pagefaults:                              Non-Paged Pool:
Total Page Faults          230697580     Successful Expansions              0
Total Page Reads            42558994     Unsuccessful Expansions            0
I/O's to read Pages         26101809     Failed Pages Accumulator           0
Modified Pages Written             0     Total Alloc Requests          419399
I/O's to write Mod Pages           0     Failed Alloc Requests              0
Demand Zero Faults          87333962
Global Valid Faults         70561285     Paged Pool:
Modified Faults             10832325     Total Failures                     0
Read Faults                        0     Failed Pages Accumulator           0
Execute Faults              74082524     Total Alloc Requests       123392503
                                         Failed Alloc Requests              0

SDA> show param paged_lal_size

   Parameter                 Variable          Address    Value    (decimal)
---------------    --------------------------  --------  --------  ---------
PAGED_LAL_SIZE     EXE$GL_PAGED_LAL_SIZE       AB023650  00000000          0
```

An Interesting Tidbit from Console Output during Boot

```
7,0,0,0 5418006301E10000 0000000000000000 EVN_BOOT_START
******************************************************
* ROM Version : 01.95
* ROM Date    : Fri Feb 01 00:39:56 PST 2013
******************************************************
7,0,0,0 3418083701E10000 000000000002000C EVN_BOOT_CELL_JOINED_PD
7,0,0,0 341800B101E10000 0000001C0205000C EVN_MEM_DISCOVERY
7,0,0,0 Start memory test ......   0/100
........
7,0,0,0 Memory test progress....  33/100
........
7,0,0,0 Memory test progress....  66/100
........
7,0,0,0 Memory test progress.... 100/100
7,0,0,0 1418002601E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,1,0 1418002605E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,2,0 1418002609E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,3,0 141800260DE10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,1,1 1418002607E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,3,1 141800260FE10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,2,1 141800260BE10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,0,1 1418002603E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,0,0 5418020701E10000 000000000011000C EVN_EFI_START
```

An Interesting Tidbit from Console Output during Boot

Questions for you, from the console output on the previous slide, can we determine if these are dual-core or quad core CPUs, if hyper-threads are enabled, what slot number (or slot numbers) this server occupies in the c7000 enclosure, how many CPU sockets are populated, and finally, can we determine if this is a BL860, a BL870 or a BL890 server?

An Interesting Tidbit from Console Output during Boot

Questions for you, from the console output on the previous slide, can we determine if these are dual-core or quad core CPUs, if hyper-threads are enabled, what slot number (or slot numbers) this server occupies in the c7000 enclosure, how many CPU sockets are populated and finally, can we determine if this is a BL860, a BL870 or a BL890 server?

Hint: In the console output, the n,n,n,n on the left hand side represents (moving from left to right) the blade slot number, the socket on that blade, the core in that socket and the thread on that core.

An Interesting Tidbit from Console Output during Boot

```
7,0,0,0 5418006301E10000 0000000000000000 EVN_BOOT_START
*****************************************************
* ROM Version : 01.95
* ROM Date    : Fri Feb 01 00:39:56 PST 2013
*****************************************************
7,0,0,0 3418083701E10000 000000000002000C EVN_BOOT_CELL_JOINED_PD
7,0,0,0 341800B101E10000 0000001C0205000C EVN_MEM_DISCOVERY
7,0,0,0 Start memory test ......   0/100
........
7,0,0,0 Memory test progress....  33/100
........
7,0,0,0 Memory test progress....  66/100
........
7,0,0,0 Memory test progress.... 100/100
7,0,0,0 1418002601E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,1,0 1418002605E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,2,0 1418002609E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,3,0 141800260DE10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,1,1 1418002607E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,3,1 141800260FE10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,2,1 141800260BE10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,0,1 1418002603E10000 000000000006000C EVN_BOOT_CPU_LATE_TEST_START
7,0,0,0 5418020701E10000 000000000011000C EVN_EFI_START
```

An Interesting Tidbit from Console Output during Boot

Therefore the answers to our questions are: this server is a BL860 in slot 7 in the c7000 enclosure, only one CPU socket on the blade is populated, the CPUs are quad-core CPUs and hyper-threads are enabled.

How does AUTOGEN handle Large Memory Systems?

Anyone ever heard of AUTOGEN_LM.COM? I hadn't either before last Fall. On systems with more than 1 TB of physical memory, the regular AUTOGEN.COM command procedure invokes AUTOGEN_LM.COM. (The LM stands for large memory.)

Although we all have our own opinions about AUTOGEN, generally speaking AUTOGEN works well on systems with less than 1 TB of physical memory. Recent testing on systems with exactly 1 TB of physical memory did uncover a bug. The AUTOGEN code was using memory size in pagelets and adding 8192 to this value. On systems with 1 TB of physical memory, this lit the sign bit in the DCL symbol and changed a very large positive number into a negative number and that led to the calculation of a negative value for PQL_DWSDEFAULT.

Please see the details in the Word document named "1 TB AUTOGEN Bug.docx".

How does AUTOGEN handle Large Memory Systems?

Anyone ever heard of AUTOGEN_LM.COM? I hadn't either before last Fall. On systems with more than 1 TB of physical memory, the regular AUTOGEN.COM command procedure invokes AUTOGEN_LM.COM. (The LM stands for large memory.)

Although we all have our own opinions about AUTOGEN, generally speaking AUTOGEN works well on systems with less than 1 TB of physical memory. Recent testing on systems with exactly 1 TB of physical memory did uncover a bug. The AUTOGEN code was using memory size in pagelets and adding 8192 to this value. On systems with 1 TB of physical memory, this lit the sign bit in the DCL symbol and changed a very large positive number into a negative number and that led to the calculation of a negative value for PQL_DWSDEFAULT.

Please see the details in the Word document named "1 TB AUTOGEN Bug.docx".

Official fixes to these issues in AUTOGEN.COM and AUTOGEN_LM.COM are in the works.

A Couple of DVE (Dynamic Volume Expansion) Quirks

DVE (Dynamic Volume Expansion) has been around since the OpenVMS Alpha V7.3-2 days. I worked a couple of interesting cases this past year and wanted to share the results with you. Actually one of these cases was a repeat of an item I presented at Boot Camp last year (September 2014).

The first case (the new issue) was logged in March 2015. SET VOLUME/SIZE was returning the following error on an OpenVMS V8.4 system:

```
$ set volume/size dsa55:
%SET-E-NOTSET, error modifying _DSA55:
-SYSTEM-F-ILLBLKNUM, illegal logical block number
```

Ideas?

A Couple of DVE (Dynamic Volume Expansion) Quirks

DVE (Dynamic Volume Expansion) has been around since the OpenVMS Alpha V7.3-2 days. I worked a couple of interesting cases this past year and wanted to share the results with you. Actually one of these cases was a repeat of an item I presented at Boot Camp last year (September 2014).

The first case (the new issue) was logged in March 2015. SET VOLUME/SIZE was returning the following error on an OpenVMS V8.4 system:

```
$ set volume/size dsa55:
%SET-E-NOTSET, error modifying _DSA55:
-SYSTEM-F-ILLBLKNUM, illegal logical block number
```

Ideas?

Although the above command was issued against a shadow set, the problem had nothing to do with shadowing. The shadow set in question was 600 GB. The system manager was attempting to expand the volume to 1 TB; exactly 1 TB (80000000 hex blocks). Anyone suspect a boundary condition here?

A Couple of DVE (Dynamic Volume Expansion) Quirks

Further testing showed that an expansion to .99 TB or to 1.01 TB worked as expected. A diagnostic F11BXQP image was built that showed for volumes of exactly 1 TB, a boundary condition was encountered. An INCR instruction in the BLISS source code should have been a INCRU, the U standing for unsigned.

What I find interesting here is that V8.4 added 2 TB volume support. Didn't V8.3 and earlier OpenVMS versions support 1 TB volumes? Well not quite, V8.3 and earlier versions supported volumes up to 2,147,475,456 decimal or 7FFFE000 hex disk blocks, or just under 1 TB. V8.4 is the first version that supported volumes of exactly 1 TB.

The second case was actually a forced crash dump due to an apparent system hang. In the dump over a hundred processes were waiting for the F11B$b lock (the blocking lock) on the system disk. The process which owned this lock in EX-mode also held PW-mode F11B$s locks on BITMAP.SYS, BADBLK.SYS and GPT.SYS, and held a PW-mode lock on the F11B$v resource for the system disk.

A Couple of DVE (Dynamic Volume Expansion) Quirks

Needless to say, file system operations on the system disk were stalled.

Checking the process' recall buffer I found a SET VOLUME/SIZE DSA0: command, DSA0: being the system disk. The system disk was being expanded from 150 GB to 210 GB. Ideas?

A Couple of DVE (Dynamic Volume Expansion) Quirks

Needless to say, file system operations on the system disk were stalled.

Checking the process' recall buffer I found a SET VOLUME/SIZE DSA0: command, DSA0: being the system disk. The system disk was being expanded from 150 GB to 210 GB. Ideas?

```
SDA> show dev/addr=@exe$gl_sysucb

I/O data structures
-------------------
DSA0                                          3PARdata VV         UCB: 903A7900
.
.
.
 --- Volume Control Block (VCB) 906C3E40 ---

Volume: PROD_SYS          Lock name: PROD_SYS
Status:         A1 write_if,extfid,system
Status2:        0D writethru,mountver,erase  <-- Erase On Delete Is Set
Status3: 00000013 subsystem,struc_ods5,special_files
Shadow status: 01 shadmast
```

A Couple of DVE (Dynamic Volume Expansion) Quirks

At last year's Boot Camp I showed a bit of source code from [F11X.LIS]ACPCNTRL.LIS. From reading the comments in this code, it turns out that the volume characteristic "erase-on-delete" is used to determine if all of the new blocks must be zeroed or not. "Erase on delete" is on by default if the volume was initialized with /ERASE and no values were specified for the /ERASE qualifier. "Erase-on-delete" can also be enabled via the SET VOLUME/ERASE_ON_DELETE command.

When erase on delete is set the volume lock is acquired (among other locks) and all of the new blocks must be zeroed. This takes a long time on a regular volume, but I would imagine it takes even longer on a thinly provisioned volume. If this customer would have waited it out, the SET VOLUME/SIZE command would have eventually completed and normal system operation would have resumed.

Remember to use Host-Based Mini-Copy when possible

When temporarily removing a shadow set member to perform backups or removing a member for some other reason, if that member will not be written to, remember to use /POLICY=MINICOPY on the DISMOUNT command. Depending on the percent populated in the mini-copy bitmap, a member can be returned to a shadow set in just a few seconds.

```
$ show device dsa1002  <-- A 256 GB Shadow Set

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA1002:                Mounted              0  EVASMLSHADOW 454016768     1   8
$1$DGA1002:       ()    ShadowSetMember      2  (member of DSA1002:)
$1$DGA2002:       ()    ShadowSetMember      1  (member of DSA1002:)
$1$DGA4002:       ()    ShadowSetMember      0  (member of DSA1002:)

$ dismount/policy=minicopy $1$DGA4002:
%%%%%%%%%%  OPCOM  25-JUN-2015 10:33:50.77  %%%%%%%%%%
$1$DGA4002: () has been removed from shadow set.

%%%%%%%%%%  OPCOM  25-JUN-2015 10:33:50.77  %%%%%%%%%%
DSA1002: shadow set has been reduced.
```

Remember to use Host-Based Mini-Copy when possible

```
$ show device dsa1002:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA1002:                Mounted              0  EVASMLSHADOW 454016768     1   8
$1$DGA1002:       ()    ShadowSetMember      2  (member of DSA1002:)
$1$DGA2002:       ()    ShadowSetMember      1  (member of DSA1002:)


$ mount/system dsa1002:/shad=$1$DGA4002:/policy=minicopy=required
_Label: EVASMLSHADOW
_Log name:
%MOUNT-I-MOUNTED, EVASMLSHADOW mounted on _DSA1002:
%MOUNT-I-SHDWMEMMCPY, _$1$DGA4002: () added to the shadow set with a minicopy operation
%MOUNT-I-ISAMBR, _$1$DGA1002: () is a member of the shadow set
%MOUNT-I-ISAMBR, _$1$DGA2002: () is a member of the shadow set

%%%%%%%%%%  OPCOM  25-JUN-2015 10:34:58.57  %%%%%%%%%%
Message from user SYSTEM on
%SHADOW_SERVER-I-SSRVBEGCPY, BEGINNING copy operation on _DSA1002: at LBN: 0, I/O size:
127 blocks, ID number: 39001363

%%%%%%%%%%  OPCOM  25-JUN-2015 10:34:59.20  %%%%%%%%%%
Message from user SYSTEM on
%SHADOW_SERVER-I-SSRVNORMAL, successful completion of copy operation on device _DSA1002:
at LBN: 536870912, ID number: 39001363
```

Caution When Setting EFIFCScanLevel

From Dave Sullivan's "60 Minutes that can save you from Disaster" talk:

"Used with Boot Order List Login. The variable EFIFCScanLevel is maintained by EFI in the system NVRAM. If the variable is not defined or set to 0, then only devices in the Boot Order List will be logged in and reported to EFI. If the variable is set to any non zero value, then all devices found on the SAN will be logged in and reported to EFI. The Driver Configuration protocol allows this variable to be created if it does not exist. If the variable does not exist, the menu will display a message asking if it should be created."

I have run into trouble when attempting to change the setting for EFIFCScanLevel from OpenVMS when using the $ MCR EFI$SET command and when using the $ MCR EFI$BCFG command. In addition, I have run into trouble when using the EFI vms_bcfg command. My results have been unpredictable. After executing one of the above commands, I have seen both of the following from the EFI drvcfg command:

Caution When Setting EFIFCScanLevel

```
6. EFI Variable EFIFCScanLevel [72057594037927936]
```

or

```
6. EFI Variable EFIFCScanLevel [?]
```

The expected output is:

```
6. EFI Variable EFIFCScanLevel [0]
```

or

```
6. EFI Variable EFIFCScanLevel [1]
```

The lesson I learned, for consistent results, was to always use the EFI drvcfg command to set or change the value for EFIFCScanLevel.

A Forced Crash due to a CPU Bottleneck

I saw the following in a forced crash:

```
SDA> show summary

Current process summary
-----------------------

 Extended Indx Process name    Username     State   Pri PCB/KTB    PHD    Wkset
-- PID -- ---- --------------- ------------ ------- --- -------- -------- ------
 23400401 0001 SWAPPER         SYSTEM       HIB      16 98933F48 98933200      4
 23401002 0002 RDM_RB72_00050  SYSTEM       LEF      15 9E74C340 B49B6000    874
 23401004 0004 RDM_RB72_00052  SYSTEM       LEF      15 9FDD2400 B49F2000    874
 23401005 0005 RDM_RB72_00053  SYSTEM       LEF      15 9E76CA80 B49FE000    644
 23400406 0006 LCKMGR_SERVER   SYSTEM       CUR 003  63 9C3965C0 B48BA000    371
 23400408 0008 CLUSTER_SERVER  SYSTEM       COM      14 9E28F2C0 B48C6000    236
 23400409 0009 SHADOW_SERVER   SYSTEM       COM      14 9E2904C0 B48CA000    244
 2340040A 000A CONFIGURE       SYSTEM       HIB      14 9C6D3B80 B48C2000    164
 2340040B 000B USB$UCM_SERVER  SYSTEM       HIB      14 9E355A80 B48CE000    478
 2340040C 000C LANACP          SYSTEM       HIB      14 9E356E00 B48D2000    317
 2340040E 000E FASTPATH_SERVER SYSTEM       HIB      14 9E35A540 B48DA000    207
 2340040F 000F IPCACP          SYSTEM       HIB      14 9E357F80 B48D6000    179
 23401010 0010 RDM_RB72_00056  SYSTEM       LEF      15 9EB716C0 B4A0A000    644
 23400411 0011 CACHE_SERVER    SYSTEM       HIB      16 9E3D16C0 B48E2000    163
 23400412 0012 OPCOM           SYSTEM       HIB      14 9E3D2240 B48E6000    217
```

A Forced Crash due to a CPU Bottleneck

```
23400413 0013 AUDIT_SERVER    AUDIT$SERVER LEF      14 9E3D2F80 B48EA000    316
23400414 0014 JOB_CONTROL     SYSTEM       LEF      14 9E3D39C0 B48EE000    258
23401016 0016 RDM_RB72_00057  SYSTEM       LEF      15 9E772A40 B4A12000    644
23401017 0017 RDM_RB72_00058  SYSTEM       LEF      15 9E756FC0 B4A16000    644
23400818 0018 RDM_RB72_00059  SYSTEM       LEF      15 9E763D00 B4A1A000    644
23400419 0019 SECURITY_SERVER SYSTEM       COM      14 9E3FFDC0 B48FE000    682
2340041A 001A ACME_SERVER     SYSTEM       HIB      14 9E400C80 B4902000    616
2340041D 001D LES$ACP_V30     SYSTEM       HIB      14 9E3D55C0 B48F6000    221
2340041E 001E NET$ACP         DNA$SessCtrl HIB      14 9E43CB40 B490A000    319
2340041F 001F REMACP          SYSTEM       HIB      14 9E444140 B490E000    150
23400421 0021 SMISERVER       SYSTEM       HIB      14 9E451E40 B4916000    364
23400422 0022 TP_SERVER       SYSTEM       HIB      14 9E454FC0 B491A000    233
23401023 0023 RDM_RB72_0005C  SYSTEM       LEF      15 9E7546C0 B4A26000    644
23401024 0024 RDM_RB72_0005D  SYSTEM       LEF      15 9E7958C0 B4A2A000    644
23400C25 0025 RDM_RB72_0005E  SYSTEM       LEF      15 9E5BC980 B4A2E000    644
23401026 0026 RDM_RB72_0005F  SYSTEM       LEF      15 9E77DC40 B4A32000    644
23401028 0028 RDM_RB72_00061  SYSTEM       LEF      15 9FC4E000 B4A46000    644
23401029 0029 RDM_RB72_00062  SYSTEM       LEF      15 9E40B000 B4A4A000    644
23400C2A 002A RDM_RB72_00063  SYSTEM       LEF      15 9E77F4C0 B4A4E000    644
23400C2B 002B RDM_RB72_00064  SYSTEM       LEF      15 9E76BB00 B4A52000    644
2340102C 002C RDM_RB72_00065  SYSTEM       LEF      15 9FC5FF80 B4A56000    644
23400C2D 002D RDM_RB72_00066  SYSTEM       LEF      15 9E798BC0 B4A5A000    644
2340042E 002E TIMER           SYSTEM       HIB      14 9E46C600 B4926000    164
23400C2F 002F RDM_RB72_00067  SYSTEM       LEF      15 9E5B3500 B4A5E000    644
```

A Forced Crash due to a CPU Bottleneck

```
.
.
.
23401067 0067 RDM_RB72_00089  SYSTEM       LEF      15 9FC4D080 B4AA2000    470
23401068 0068 RDM_RB72_0008A  SYSTEM       CUR 000  15 A0103300 B4AAA000    470
23401069 0069 APACHE$SWS0004  APACHE$WWW   LEF      14 9E557480 B4AAE000    289
2340046A 006A APACHE$SWS      APACHE$WWW   LEF      14 9C39C500 B48BE000   2783
2340106B 006B RDM_RB72_0008B  SYSTEM       CUR 002  15 9FCC3900 B4AC2000    470
2340106C 006C RDM_RB72_0008C  SYSTEM       CUR 001  15 9E8FEC40 B4B06000    470
2340106D 006D FXDB005000D73   SYSTEM       LEF      14 9E75A540 B4B16000    235
2340106E 006E FXDB005000E73   SYSTEM       LEF      14 9E5E1640 B49BE000    235
2340106F 006F FXDB005000F73   SYSTEM       LEF      14 9E713E40 B4B1A000    235
.
.
.
```

What is unusual about the process priorities? Could PIXSCAN priority boosts have occurred?

A Forced Crash due to a CPU Bottleneck

```
.
.
.
23401067 0067 RDM_RB72_00089  SYSTEM       LEF      15 9FC4D080 B4AA2000    470
23401068 0068 RDM_RB72_0008A  SYSTEM       CUR 000  15 A0103300 B4AAA000    470
23401069 0069 APACHE$SWS0004  APACHE$WWW   LEF      14 9E557480 B4AAE000    289
2340046A 006A APACHE$SWS      APACHE$WWW   LEF      14 9C39C500 B48BE000   2783
2340106B 006B RDM_RB72_0008B  SYSTEM       CUR 002  15 9FCC3900 B4AC2000    470
2340106C 006C RDM_RB72_0008C  SYSTEM       CUR 001  15 9E8FEC40 B4B06000    470
2340106D 006D FXDB005000D73   SYSTEM       LEF      14 9E75A540 B4B16000    235
2340106E 006E FXDB005000E73   SYSTEM       LEF      14 9E5E1640 B49BE000    235
2340106F 006F FXDB005000F73   SYSTEM       LEF      14 9E713E40 B4B1A000    235
.
.
.
```

What is unusual about the process priorities? Could PIXSCAN priority boosts have occurred?

Clearly a CPU bottleneck was present on this system and the PIXSCAN mechanism raised process priorities to 15. Once these processes reached quantum end at priority 15, they waited in COM again at priority 14.

A Forced Crash due to a CPU Bottleneck

The system was not hung, but extremely slow due to the four CPUs being overwhelmed. Many RDB processes were accumulating large quantities of CPU time and were running at priority 15. 8 processes (including several system process) were in COM at priority 14. One process holding a key lock had not reached quantum end for 566 seconds. The process at the front of the ACP request queue was CUR on CPU 00, but it also had not reached quantum end for 13.57 seconds.

One oddity was the dedicated CPU lock manager running on a system with only 4 CPUs. I was not sure if this was wise because on a system with only 4 CPUs, the dedicated CPU lock manager will consume 25% of the total CPU cycles.

Recommendations: Disable the dedicated CPU lock manager and check into those RDB processes that were consuming large quantities of CPU time.

Halt Testing on Integrity Servers

On Alpha systems we have a console environmental variable named AUTO_ACTION. This console environmental variable could be set to either BOOT, HALT or RESTART. For certain bugcheck types, AUTO_ACTION had to be set to RESTART in order to generate a crash dump.

On Integrity servers there is no AUTO_ACTION console environmental variable. There are EFI bcfg boot commands to set “auto-action type behavior” to either BOOT or HALT, but there is nothing for RESTART. What happens on an Integrity server if a CPU encounters a halt instruction? A curious mind wanted to know.

My first tests were on a single-CPU Integrity server, an rx2660 (and it is extremely rare to encounter a single-CPU Integrity server).

```
$ run halt

**** Primary HALTED with code HWRPB_HALT$K_NO_ACTION

****   Hit any key to cold reboot   ****
P00>>>
```

Halt Testing on Integrity Servers

```
......<reset>......
*******************************************************
* ROM Version : 04.11
* ROM Date    : 10/20/2008
* BMC Version :  05.25
*******************************************************
0 0 0x0015B2 0x0000000012932813 boot time event
1 0 0x0000A4 0x0000000000000000 start memory configuration
2 0 0x001CBB 0x0000000000000000 System set to insecure mode
0 0 0x0015B2 0x0000000024532997 boot time event
1 0 0x000014 0x0000000000000000 CPU0 starting
.
.
.
```

As soon as I hit a key, the system did a cold restart and no crash dump was captured. How do you capture a crash dump on a single CPU Integrity server when that single CPU encounters a HALT instruction?

The answer is, use ToC, the Transfer of Control mechanism.

Halt Testing on Integrity Servers

```
$ run halt      <-- From OPA0:

**** Primary HALTED with code HWRPB_HALT$K_NO_ACTION

****   Hit any key to cold reboot   ****

P00>>>              <-- I entered Ctrl-B

   MP MAIN MENU:

         CO: Console
        VFP: Virtual Front Panel
         CM: Command Menu
      SMCLP: Server Management Command Line Protocol
         CL: Console Log
         SL: Show Event Logs
         HE: Main Help Menu
          X: Exit Connection

[vc4mp] MP> cm      <-- I entered cm
```

Halt Testing on Integrity Servers

```
(Use Ctrl-B to return to MP main menu.)

[vc4mp] MP:CM> tc  <-- I entered tc

TC

Execution of this command irrecoverably halts all system processing and
I/O activity and restarts the computer system.

Type Y to confirm your intention to restart the system: (Y/[N]) y  <-- I
entered y
y
   -> SPU hardware was successfully issued a TOC.

[vc4mp] MP:CM>     <-- I entered Ctrl-B

   MP MAIN MENU:

         CO: Console
        VFP: Virtual Front Panel
         CM: Command Menu
      SMCLP: Server Management Command Line Protocol
```

Halt Testing on Integrity Servers

```
          CL: Console Log
          SL: Show Event Logs
          HE: Main Help Menu
           X: Exit Connection

[vc4mp] MP> co     <-- I entered co

        [Use Ctrl-B or ESC-( to return to MP main menu.]

- - - - - - - - - - Prior Console Output - - - - - - - - - -

**** OpenVMS I64 Operating System V8.4     - BUGCHECK ****

** Bugcheck code = 00000AFC: CPUINT_INIT, Hardware INIT interrupt received
** Crash CPU: 00000000    Primary CPU: 00000000    Node Name: VC4
** Highest CPU number:    00000000
** Active CPUs:           00000000.00000001
** Current Process:       _OPA0:
** Current PSB ID:        00000001
** Image Name:            DSA14:[SYS0.][SYSMGR]HALT.EXE;1
.
.
```

Halt Testing on Integrity Servers

After the system rebooted, the crash dump looked good.

```
SDA> show crash

System crash information
------------------------

Time of system crash:                19-MAY-2015 12:44:23.14
Version of system:                   OpenVMS I64 Operating System, Version V8.4

System Version Major ID/Minor ID:  3/0
VMScluster node:                     XXXXXX
System type:                         HP rx2660  (1.59GHz/12.0MB)
.
.
.
CPU 000 reason for Bugcheck: CPUINT_INIT, Hardware INIT interrupt received
Process currently executing on this CPU: _OPA0:
Current image file: DSA14:[SYS0.][SYSMGR]HALT.EXE;1
Current IPL: 31  (decimal)
.
.
```

Halt Testing on Integrity Servers

```
Exception Frame Summary:

Exception Frame    Type              Stack          IIP / Ret_Addr
Trap_Type / Service_Number
-----------------  ----              -----          -----------------   -----
--------------------
FFFFFFFF.A6057CA0   ORIGINAL_INTSTK   MCHK (Init)    FFFFFFFF.80044E20
00000066  System Machinecheck Abort

00000000.7FF43770   INTSTK           Kernel         00000000.00000100
00000060  Interval Clock Interrupt

00000000.7FF43A90   STALE_SSENTRY    Kernel         FFFFFFFF.00000008
01000170  SYS$CRETVA_64

00000000.7FF43BB0   INTSTK           Kernel         00000000.00010110
0000008A  BREAK_HALT

00000000.7FF43F40   SSENTRY          Kernel         00000000.00010080
01000019  SYS$CMKRNL
```

Halt Testing on Integrity Servers

```
SDA> show exception 00000000.7FF43BB0

Exception Frame at 00000000.7FF43BB0
------------------------------------

    IPL                 =                   00
    TRAP_TYPE           =             0000008A      BREAK_HALT
    IVT_OFFSET          =             00002C00      Break Instruction
    IIP                 =   00000000.00010110      SYS$K_VERSION_16+000D0
    IIPA                =   00000000.00010100      SYS$K_VERSION_16+000C0
    IFA                 =   000007FD.FFFEC000

SDA> exam/inst 00010110-20;20
                          {  .mii
SYS$K_VERSION_16+000B0:                 alloc       r41 = ar.pfs, 0D, 00, 00
                                        mov         r43 = r4
                                        mov         r42 = r12
                          }
                          {  .mmi
SYS$K_VERSION_16+000C0:                 mov         r29 = r12 ;;
                                        mov         r44 = r6
                                        nop.i       000000
                          }
```

Halt Testing on Integrity Servers

```
                          {  .mfi
SYS$K_VERSION_16+000D0:                 break.m     100003  <-- HALT Instruction
                                        nop.f       000000
                                        nop.i       000000 ;;
                          }
```

On multiple CPU Integrity servers (and the vast majority of Integrity servers have more than one CPU) there is not an issue. If one of the CPUs encounters a HALT instruction, within a few seconds (SMP_SANITY_CNT in SYSGEN) the system will crash with a CPUSANITY bugcheck.

```
$ run halt

**** Primary HALTED with code HWRPB_HALT$K_NO_ACTION

****   Hit any key to cold reboot   ****
P00>>>

**** OpenVMS I64 Operating System V8.4     - BUGCHECK ****

** Bugcheck code = 00000774: CPUSANITY, CPU sanity timer expired
```

Halt Testing on Integrity Servers

```
** Crash CPU: 0000000F    Primary CPU: 00000000    Node Name: XXXXXX
** Highest CPU number:    0000000F
** Active CPUs:           00000000.0000FFFF
** Current Process:       NULL
** Current PSB ID:        00000001
** Image Name:

**** Starting compressed selective memory dump at 20-MAY-2015 05:01...
.....................................................................
.....................................................................
.....................................................
** System space, key processes, and key global pages have been dumped.
** Now dumping remaining processes and global pages...
.............................
...Complete ****

        SYSTEM SHUTDOWN COMPLETE

**** Primary HALTED with code HWRPB_HALT$K_REMAIN_HALTED

****   Hit any key to cold reboot   ****
P00>>>
```

Adding RMS Global Buffers to a file should help performance, right?

Working a call earlier this Spring, the system manager reported that as he continued to add (or increase the number of) global buffers on their RMS indexed files, overall system performance declined and direct I/O rates increased.

To make the changes, the system manager was editing FDL files and running CONVERT operations. A “before” and “after” view of one of the FDL files showed the following differences:

```
File $1$DGA22:[ZZZZ.YYYY]XXXX.FDL;23
   24           GLOBAL_BUFFER_COUNT     68
   25           NAME                    "XXXX.ISM"
   26           ORGANIZATION            indexed
   27

******

File $1$DGA22:[ZZZZ.YYYY]XXXX.FDL;24
   26           NAME                    "XXXX.ISM"
   27           ORGANIZATION            indexed
   28           GLBUFF_CNT_V83          160
   29           GLBUFF_FLAGS_V83        none
```

Ideas?

Adding RMS Global Buffers to a file should help performance, right?

We can see in the “old” FDL file, that they were using the pre-V8.3 mechanism of specifying the global buffer count for the file similar to a SET FILE/GLOBAL=nnnn command. The pre-V8.3 mechanism uses a 16-bit field to store the number of global buffers, which limits the number of global buffers to a maximum of 32,767.

In the “new” FDL file, they were using the V8.3+ mechanism of specifying the global buffer count for the file similar to a SET FILE/GLOBAL=COUNT=nnnn command. The V8.3+ mechanism uses a 32-bit field to store the number of global buffers, which allows for a much larger value.

Shouldn't both methods work equally well? How can a system manager determine the number of global buffers (pre-V8.3 and V8.3+) on a file?

Research showed that the DUMP/HEADER command only displays the pre-V8.3 setting (if there is one) and completely ignores any V8.3+ global buffer setting. The DIRECTORY command displays both global buffer settings, if both the pre-V8.3 setting and the V8.3+ setting are nonzero, but if only one setting is nonzero, only one number is displayed without an indication if it is pre-V8.3 or V8.3+.

Adding RMS Global Buffers to a file should help performance, right?

From DUMP/HEADER output:

```
Default extension size:           0
Global buffer count:             15
Directory version limit:          0
```

From DIR/FULL output when both pre-V8.3 and V8.3+ settings are nonzero:

```
File attributes:    Allocation: 16, Extend: 0, Global buffer count pre-V8.3: 15,
                    Global buffer count V8.3+: 30, No version limit
```

From DIR/FULL output when only one setting is nonzero:

```
File attributes:    Allocation: 16, Extend: 0, Global buffer count: 30  <-- pre-V8.3 or V8.3+ ?????
                    No version limit
```

This made troubleshooting a bit more difficult. One mechanism that we did find to accurately determine the global buffer settings was the output from ANAL/RMS.

```
Global Buffer Count  pre-V8.3:          0
Global Buffer Count post-V8.3:         30
Global Buffer Flags post-V8.3:       none
```

Adding RMS Global Buffers to a file should help performance, right?

OK, now that we have some background, what was the customer's problem? Using the SDA> SHOW PROC/RMS=(GBH,GBDSUM) command we determined that after the RMS indexed files had been CONVERTed, OpenVMS didn't seem to use global buffers on the files for some reason. Further debugging narrowed it down to just RMS indexed files opened by the customer's application, and that led us to the Synergy DIBOL RTL.

The Synergy DIBOL RTL does not appear to recognize or utilize the V8.3+ RMS global buffer settings. For this case, the customer did not need to use the V8.3+ specifications because the number of global buffers they were using were much less than the pre-V8.3 limit of 32,767. Therefore, the fix was simple: revert back to pre-V8.3 specifications for global buffers in their FDL files.

OpenVMS engineering will be making changes to DUMP/HEADER output to display both the pre-V8.3 and V8.3+ global buffer settings. Engineering was reluctant to change DIR/FULL output due to backward compatibility issues.

The RMS Directory Path Cache and deleting .DIR files

When locking bottlenecks occur on a mounted volume, often file serialization locks (F11B$s locks) on the MFD (000000.DIR) are one of the contributing factors. One reason a file serialization lock would be taken on the MFD would be to parse a directory path.

For existing processes, RMS caches the directory paths in a per-process, per-volume RMS directory path cache. File system locks are not required when traversing this cache. However, this RMS directory path cache gets invalidated when any directory (any .DIR file) on that volume is deleted by any process on any node in the cluster. Once the RMS directory path cache gets invalidated for a volume, every process, cluster wide, needs to parse the directory path again, and for this operation the process needs to acquire a PW-mode file serialization lock on the MFD on that volume.

Newly created processes need to build their RMS directory path cache; therefore, newly created processes need to acquire a file serialization lock on the MFD as well.

The RMS Directory Path Cache and deleting .DIR files

Testing was done with some batch jobs accessing files on a volume. Initially the F11B$s locks on the MFD were all in NL-mode.

```
Resource Database
-----------------
RSB:          FFFFFFFF.7D3BD840  GGMODE:     NL  Status: VALID
Parent RSB:   FFFFFFFF.7E00A300  CGMODE:     NL
Sub-RSB count:      0            FGMODE:     NL
Lock Count:         6            RQSEQNM:  0531
BLKAST count:       0            CSID: 00000000  (XXXXXX)

Resource:          00047324 42313146  F11B$s..  Valblk: 00000005 00000002
Length   10        00000000 00000000  ........          00000000 00000000
Kernel mode        00000000 00000000  ........          ................
System             00000000 00000000  ........  Seqnum: 000047DA

Granted queue (Lock ID / Gr mode / Range):
03001D87  NL 00000000-FFFFFFFF         7100120F  NL 00000000-FFFFFFFF
6E001795  NL 00000000-FFFFFFFF         7A004311  NL 00000000-FFFFFFFF
480076D0  NL 00000000-FFFFFFFF         0700342F  NL 00000000-FFFFFFFF

Conversion queue (Lock ID / Gr mode / Range -> Rq mode / Range):
     *** EMPTY QUEUE ***

Waiting queue (Lock ID / Rq mode / Range):
     *** EMPTY QUEUE ***
SDA>
```

The RMS Directory Path Cache and deleting .DIR files

As soon as a directory file (a .DIR file) was deleted on that volume, there was a spike in PW-mode lock requests on the MFD.

```
Resource Database
-----------------
RSB:          FFFFFFFF.7D3BD840  GGMODE:      PW  Status: VALID   WTFULRG
Parent RSB:   FFFFFFFF.7E00A300  CGMODE:      PW
Sub-RSB count:       0           FGMODE:      PW
Lock Count:         31           RQSEQNM:   050C
BLKAST count:        0           CSID: 00000000  (XXXXXX)

Resource:          00047324 42313146  F11B$s..  Valblk: 00000005 00000002
Length   10        00000000 00000000  ........          00000000 00000000
Kernel mode        00000000 00000000  ........          ................
System             00000000 00000000  ........  Seqnum: 00004703

Granted queue (Lock ID / Gr mode / Range):
1D0070BD  PW 00000000-FFFFFFFF         03001D87  NL 00000000-FFFFFFFF
7100120F  NL 00000000-FFFFFFFF         6E001795  NL 00000000-FFFFFFFF
7A004311  NL 00000000-FFFFFFFF         480076D0  NL 00000000-FFFFFFFF
0700342F  NL 00000000-FFFFFFFF

Conversion queue (Lock ID / Gr mode / Range -> Rq mode / Range):
     *** EMPTY QUEUE ***

Waiting queue (Lock ID / Rq mode / Range):
0C0065AE  PW 00000000-FFFFFFFF         05003802  PW 00000000-FFFFFFFF
2B002019  PW 00000000-FFFFFFFF         58001687  PW 00000000-FFFFFFFF
340035E8  PW 00000000-FFFFFFFF         29003ACE  PW 00000000-FFFFFFFF
3E0012EA  PW 00000000-FFFFFFFF         520073DD  PW 00000000-FFFFFFFF
```

The RMS Directory Path Cache and deleting .DIR files

```
1B00117A  PW 00000000-FFFFFFFF        1C001A29  PW 00000000-FFFFFFFF
07006999  PW 00000000-FFFFFFFF        15002CD7  PW 00000000-FFFFFFFF
4F000704  PW 00000000-FFFFFFFF        2700740C  PW 00000000-FFFFFFFF
4A003599  PW 00000000-FFFFFFFF        47003D04  PW 00000000-FFFFFFFF
58000552  PW 00000000-FFFFFFFF        7D000AAF  PW 00000000-FFFFFFFF
3E0045F9  PW 00000000-FFFFFFFF        2D002644  PW 00000000-FFFFFFFF
5F00777D  PW 00000000-FFFFFFFF        18003165  PW 00000000-FFFFFFFF
13001338  PW 00000000-FFFFFFFF        06007627  PW 00000000-FFFFFFFF
```

Recommendation: Avoid deleting .DIR files on active volumes. Perform these delete operations off-hours if possible.

Note: These tests were done by OpenVMS engineering on a volume that was full (zero free blocks). This was in response to a case logged in which a volume was full and on which unusually high locking activity was observed. As files were deleted on this volume in an attempt to free space, directories were deleted as well. This directory file deletion potentially aggravated the locking situation by invalidating the per-process RMS directory path cache on this volume.

Locating Multinet Code in S0/S1 Space in a Crash Dump

This next topic is likely only applicable to folks in support. When working a crash dump, if the failure occurs in code that SDA has no symbols for, how can you determine whose code it is? This “mystery” code could be an in-house application, a user-written system service or a 3rd party product, etc..

This can be especially challenging when a crash dump file has been sent to a Customer Support Center for analysis. Many techniques are available if access to the system that crashed is available, but if you only have access to the crash dump file, what are your options?

For example, for systems running Multinet from Process Software, the kind folks at Process Software have made logical names available, and these logical names are available in paged dynamic memory (PAGEDYN) in the crash dump. The logical name of interest is MULTINET_KERNEL_BASE_ADDRESS.

```
!
! Searching for MULTINET logicals in PAGEDYN
!
```

Locating Multinet Code in S0/S1 Space in a Crash Dump

```
SDA> clue memory/layout
System Virtual Address Space Layout:
------------------------------------
   Item                                  Base                End          Length
System Virtual Base Address        FFFFFEFE.00000000
PFN Database                       FFFFFEFE.00000000  FFFFFEFE.0A000000  0A000000
.
.
.
Paged Pool                         FFFFFFFF.861B8000  FFFFFFFF.86DD4000  00C1C000
.
.
.

SDA> search/step=byte/length=quad 861B8000:86DD4000 415F455341425F4C  <-- L_BASE_A
Searching from FFFFFFFF.861B8000 to FFFFFFFF.86DD4000 in BYTE steps for 415F4553.41425F4C...
Match at FFFFFFFF.8642D45E    415F4553.41425F4C


SDA> exam FFFFFFFF.8642D45E-5e;80
5F4E4F4D 4D4F435F 54454E49 544C554D  MULTINET_COMMON_     FFFFFFFF.8642D400
4D5D5445 4E49544C 554D5B3A 544F4F52  ROOT:[MULTINET]M     FFFFFFFF.8642D410
017A3735 3B455845 2E54454E 49544C55  ULTINET.EXE;57z.     FFFFFFFF.8642D420
00000000 00400080 861BA84C 8656A790  .§V.L¨....@.....     FFFFFFFF.8642D430  <-- LNM Structure
0000001C 00000000 8642D470 859E7EC8  È~..pÔB.........     FFFFFFFF.8642D440
5F4C454E 52454B5F 54454E49 544C554D  MULTINET_KERNEL_     FFFFFFFF.8642D450
00000000 53534552 4444415F 45534142  BASE_ADDRESS....     FFFFFFFF.8642D460
00000000 00000000 00000000 00000000  ................     FFFFFFFF.8642D470
```

Locating Multinet Code in S0/S1 Space in a Crash Dump

```
SDA> form FFFFFFFF.8642D430
%SDA-E-NOSYMBOLS, no "LNM" symbols found to format this block


SDA> exam FFFFFFFF.8642D430;80
00000000 00400080 861BA84C 8656A790  .§V.L¨....@.....     FFFFFFFF.8642D430
0000001C 00000000 8642D470 859E7EC8  È~..pÔB.........     FFFFFFFF.8642D440
5F4C454E 52454B5F 54454E49 544C554D  MULTINET_KERNEL_     FFFFFFFF.8642D450
00000000 53534552 4444415F 45534142  BASE_ADDRESS....     FFFFFFFF.8642D460
00000000 00000000 00000000 00000000  ................     FFFFFFFF.8642D470
30303045 31463738 00000022 00000000  ....".87F1E000       FFFFFFFF.8642D480
30303833 36314220 36333539 38303120   1089536 B163800     FFFFFFFF.8642D490
00000000 00003043 33354146 31422030  0 B1FA53C0......     FFFFFFFF.8642D4A0
```

The 87F1E000 and the 1089536 in red above are the base address and the length for the Multinet Kernel in S0/S1 space.

A similar technique could possibly be used to identify other products.

A Multinet issue when using DECnet over IP

While we are talking about Multinet, we saw a couple of cases this past year where non-paged pool had filled up with DECnet VCRP data structures. Once non-paged pool is exhausted, the system may hang or even crash with a variety of bugcheck types including: CPUSPINWAIT, CLUEXIT, RESEXH, etc.

Researching these events showed that in each case a large file was being copied between OpenVMS nodes using DECnet over IP; the IP stack being Multinet. The sites were running Multinet V5.3 and V5.4. Per the following release note this issue has been corrected via the introduction of a new kernel variable named PWIP_IO_LIMIT in Multinet V5.4. The recommended setting for PWIP_IO_LIMIT is 200.

```
DECNET-over-IP (Phase V) or other operations that use PWIPDRIVER
may freeze, with the processes involved hanging in LEF state.
Nonpaged pool expansion may or may not accompany the process hangs.
Other ancillary problems, such as pool allocation failures in other
processes, or system crashes, may also occur on rare occasions.

The KERNEL-UPDATE dependency provides a means of controlling the
memory and I/O usage while using DECNET-over-IP (Phase V).
A new kernel variable PWIP_IO_LIMIT has been provided.
```

Shadow Sets with Zero Working Members

A shadow set (a DSAnnnn: device) with zero working members is very likely not a good thing. In a recent case, the topic for discussion was; how does a shadow set end up with zero working members and does the SYSGEN parameter MVTIMEOUT play a role?

Two tests were performed. In the first test a single-member shadow set was created, then the single physical member of that shadow sets was unpresented from the EVA. Commands to that DSAnnnn: device hung and after MVTIMEOUT number of seconds the shadow set exited from mount verification, went into a mount verification timeout state and contained zero working members.

Note: Even though the SYSGEN parameter MVTIMEOUT is dynamic, the value of MVTIMEOUT at the time of the shadow set mount appears to be used to determine when mount verification will timeout.

Shadow Sets with Zero Working Members

```
$ show dev dsa4444

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA4444:                Mounted              0  SHADOWMVTEST *********     1   1
$1$DGA4403:      (XXX)  ShadowSetMember      0  (member of DSA4444:)


$ mcr sysgen show mvt
Parameter Name            Current    Default     Min.       Max.   Unit  Dynamic
--------------            -------    -------   -------    -------  ----  -------
MVTIMEOUT                    3600       3600         1      64000 Seconds    D


$ !
$ ! Unpresent $1$DGA4403: From The EVA
$ !


$ set def DSA4444:[000000]

$ dir
%%%%%%%%%%  OPCOM  17-APR-2015 08:30:33.42  %%%%%%%%%%
Device DSA4444: is offline.
Mount verification is in progress.

XXX::_TNA17: 08:48:58 DIRECTORY CPU=00:00:00.14 PF=1966 IO=3851 MEM=301
XXX::_TNA17: 08:48:59 DIRECTORY CPU=00:00:00.14 PF=1966 IO=3852 MEM=301
XXX::_TNA17: 08:49:03 DIRECTORY CPU=00:00:00.14 PF=1966 IO=3853 MEM=301
```

Shadow Sets with Zero Working Members

```
!
! From Another Session We See DSA4444: Is In Mount Verification
!

$ show dev dsa4444

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA4444:                MountVerify          0  SHADOWMVTEST *********     3   1
                        mounted
$1$DGA4403:      (VC5)  ShadowSetMember      0  (member of DSA4444:)


!
! From The Console, I Didn't Want To Wait An Hour For Mount Verification To Timeout
!

Interrupt Priority C

Commands:
  C device   Cancel Mount Verification
  Q          Adjust Quorum
  CTRL-P     Prompt for Crash
  CTRL-Z     Exit IPC

IPC> C DSA4444
IPC> ^Z
```

Shadow Sets with Zero Working Members

```
!
! Back To The Original Session Where The DIRECTORY Command Was Hung
!

%%%%%%%%%%  OPCOM  17-APR-2015 08:57:27.12  %%%%%%%%%%
$1$DGA4403: (XXX FGG) has been removed from shadow set.

%%%%%%%%%%  OPCOM  17-APR-2015 08:57:27.12  %%%%%%%%%%
Mount verification has aborted for device DSA4444:

%%%%%%%%%%  OPCOM  17-APR-2015 08:57:27.12  %%%%%%%%%%
DSA4444: contains zero working members.


$ show dev dsa4444:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA4444:                MntVerifyTimeout     1  SHADOWMVTEST *********     1   1


$ dism DSA4444:  <-- We Can DISMOUNT The Device Because The Transaction Count Is 1


$ show dev dsa4444:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA4444:                Online               1
```

Shadow Sets with Zero Working Members

In the second test a single-member shadow set was created, then the single physical member of that shadow sets was presented to an OpenVMS node outside of the cluster. This outside of the cluster OpenVMS node also mounted the single physical member of that shadow set using /OVER=(ID,SHAD).

Test results: In this case of an outside of the cluster mount of the only physical member of a single member shadow set, as soon as a path switch occurs (or other event that triggers mount verification), the shadow set will immediately enter a mount verification timeout state and contain zero working members on the cluster nodes.

Note: The setting for the SYSGEN parameter MVTIMEOUT did NOT matter in this scenario.

Shadow Sets with Zero Working Members

```
$ show dev dsa4444

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA4444:                Mounted              0  SHADOWMVTEST *********     1   1
$1$DGA4403:      (XXX)  ShadowSetMember      0  (member of DSA4444:)


!
! From A Node Outside Of The cluster
!

$ mount/over=(id,shad) $1$DGA4403:
%MOUNT-I-MOUNTED, SHADOWMVTEST mounted on _$1$DGA4403: (XXX)
%MOUNT-I-REBUILD, volume was improperly dismounted; rebuild in progress
$


!
! Back To The Original Cluster Node, Cause A Path Switch To Trigger Mount Verification
!


$ set device/switch/path=FGG0.5001-4380-0119-58AD $1$DGA4403:
```

Shadow Sets with Zero Working Members

```
%%%%%%%%%%  OPCOM  17-APR-2015 13:08:11.46  %%%%%%%%%%
13:08:11.46 Multipath access to device $1$DGA4403: has been manually switched from
path FGG0.5001-4380-0119-58A9 (XXX) to path FGG0.5001-4380-0119-58AD (XXX)

%%%%%%%%%%  OPCOM  17-APR-2015 13:08:11.46  %%%%%%%%%%
$1$DGA4403: (XXX FGG, YYY) is an incorrect shadow set member volume.

%%%%%%%%%%  OPCOM  17-APR-2015 13:08:11.69  %%%%%%%%%%
$1$DGA4403: (XXX FGG, YYY) has been removed from shadow set.

%%%%%%%%%%  OPCOM  17-APR-2015 13:08:11.69  %%%%%%%%%%
Mount verification has aborted for device DSA4444:

%%%%%%%%%%  OPCOM  17-APR-2015 13:08:11.69  %%%%%%%%%%
DSA4444: contains zero working members.


$ show dev dsa4444:

Device                  Device           Error    Volume         Free  Trans Mnt
 Name                   Status           Count     Label        Blocks Count Cnt
DSA4444:                MntVerifyTimeout     1   SHADOWMVTEST *********     1   1


$ dismount dsa4444:
```

Unusual SYS$EXAMPLES:RAD.COM Output

Running the SYS$EXAMPLES:RAD.COM script can provide valuable information about the i2 Integrity blade server that OpenVMS is running on. From the output you can infer whether or not hyper-threads are enabled, the number of CPU sockets which are populated, total system memory, the EFI memconfig option selected and to some extent the layout of memory on the server.

That said, lately we have seen some unusual output from RAD.COM when executed on i2 blade servers. The following slides depict several such examples.

Note: The following problem was corrected in the VMS84I_SYS-V0500 patch kit.

```
For a particular RAD memory configuration, RAD.COM can
show incorrect memory distribution where the memory
belonging to multiple RADs is added up to one specific
RAD and the other RADs are showing no memory.
```

Note: The unusual output on the following slides is NOT due to this problem.

Unusual SYS$EXAMPLES:RAD.COM Output

```
Node: XXXXXX Version: V8.4      System: HP BL890c i2  (1.73GHz/6.0MB)
RAD   Memory (GB)   CPUs
===   ===========   ===============
  0       14.00     0-3
  1       14.00     4-7
  2       14.00     8-11
  3       14.00     12-15
  4       16.00     16-19
  5       16.00     20-23
  6        7.99     0-15
```

In the BL890 server above, total physical memory is 96 GB. Only 6 out of the 8 CPU sockets are populated. The memconfig option is set to MostlyNUMA (1/8 ILM, 7/8 SLM). We know hyper-threads are disabled because we do not see the 24-27 to the right of the 0-3, 28-31 to the right of the 4-7, etc. in the CPU column.

There is only one ILM RAD (RAD 6) and the memory for this RAD is taken from RADs 0 through 3. We cannot have a second ILM RAD because only one of the two CPU sockets is populated in the 3rd and 4th blades.

Unusual SYS$EXAMPLES:RAD.COM Output

```
Node: XXXXXX Version: V8.4      System: HP BL890c i2  (1.73GHz/6.0MB)
 RAD    Memory (GB)   CPUs
===    ===========   ==============
  0        96.00     0-3
  1        32.00     4-7
  2        96.00     8-11
  3        32.00     12-15
  4        96.00     16-19
  5        32.00     20-23
  6        96.00     24-27
  7        32.00     28-31
  8       127.99     0-15
  9       128.00     16-31
```

In the BL890 server above, total physical memory is 768 GB. All CPU sockets are populated. The memconfig option is set to Balanced (1/2 ILM, 1/2 SLM). Hyper-threads are disabled. We see two ILM RADs; RADs 8 and 9.

The unusual pattern of 96GB/32GB in the SLM RADs is due to the use of 16 GB memory DIMMs in this server and the required loading of these DIMMs in quads.

Unusual SYS$EXAMPLES:RAD.COM Output

```
Node: XXXXXX Version: V8.4      System: HP BL890c i2  (1.73GHz/6.0MB)
 RAD    Memory (GB)   CPUs
===    ===========   ==============
  0        32.00     0-3
  1        32.00     4-7
  2        32.00     8-11
  3        32.00     12-15
  4        32.00     16-19
  5        32.00     20-23
  6        32.00     24-27
  7        32.00     28-31
  8       383.99     0-15
  9       384.00     16-31
```

In the BL890 server above, total physical memory is 1 TB. All CPU sockets are populated. The memconfig option is set to MostlyUMA (3/4 ILM, 1/4 SLM). Hyper-threads are disabled. We see two ILM RADs; RADs 8 and 9.

Question: What is the maximum possible physical memory on a BL890 i2 blade server?

Unusual SYS$EXAMPLES:RAD.COM Output

```
Node: XXXXXX Version: V8.4      System: HP BL890c i2  (1.73GHz/6.0MB)
 RAD    Memory (GB)   CPUs
===    ===========   ==============
  0        32.00     0-3
  1        32.00     4-7
  2        32.00     8-11
  3        32.00     12-15
  4        32.00     16-19
  5        32.00     20-23
  6        32.00     24-27
  7        32.00     28-31
  8       383.99     0-15
  9       384.00     16-31
```

In the BL890 server above, total physical memory is 1 TB. All CPU sockets are populated. The memconfig option is set to MostlyUMA (3/4 ILM, 1/4 SLM). Hyper-threads are disabled. We see two ILM RADs; RADs 8 and 9.

Question: What is the maximum possible physical memory on a BL890 i2 blade server? Answer: 1.5 TB.

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

Distributed lock manager performance in an OpenVMS cluster is often critical to overall cluster performance. Some tuning knobs are available via SYSGEN parameters such as LOCKIDTBL, RESHASHTBL, LOCKDIRWT, LOCKRMWT and PE1. Additional tuning is possible via fast path CPU assignments for PEDRIVER and the SCS NICs. With the VMS84x_SYSLOA-V0200 kits, the distributed locking work performed by SYS$CLUSTER code can now be moved from the PEDRIVER CPU to another CPU via the RSVD_CLU_1 SYSGEN parameter.

After the above tuning is complete, or just to monitor an existing cluster node, one of the more popular commands that can be used to view locking activity is SDA> LCK SHOW ACTIVE. For each root resource, this command displays locking statistics for that resource tree, sorted by locking activity. The activity displayed (the Act column on the next slide) is derived from local locking rates against that entire resource tree. It is calculated by taking 1/8 of the locking rate observed during the previous 8 seconds and then combining this value with 7/8 of the previous locking rate. The rate displayed is for an 8 second period.

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

```
SDA> lck show active
Active Resource Tree Information (Node XXXXX1)
----------------------------------------------
                        Total     Local
RSB Address             Locks     Locks    SubRSB     Act   Node     Resource Name
-----------------    -------   -------   ------   -----   ------   ----------------------------
FFFFFFBE.A2EC3340    6102008   6102008     3897   65528   XXXXX1   Ý...D....DISK999.............
                                                             DISK999:[TEST.DB.YYY]YYY.RDB;1
FFFFFFBE.8FA3EE40        695       695      678   26216   XXXXX0   RMS$R{×....AXPVMSSYS.........
                                                             _$1$DGA9:[VMS$COMMON.SYSEXE]SYSUAF.DAT
FFFFFFBE.8AC46300       3808      3808       14   20657   XXXXX1   SYS$SYS_ID.P....
FFFFFFBE.8AC45D40       5042      5042     5040   20548   XXXXX5   F11B$vI64VMSSYS
FFFFFFBE.8ACDE6C0         61        61       32   15665   XXXXX0   F11B$vAXPVMSSYS
FFFFFFBE.8AC93340       8025      8025     8023   11525   XXXXX5   F11B$vDISK888
...
```

Note: On a cluster node, the lock activity can reach a maximum 65,528. Occasionally a value of 65,535 may be seen; however, this is a transient value and implies the resource tree is in the process of being re-mastered.

Note: On a standalone system the lock activity is not calculated via the 1/8 - 7/8 algorithm. The activity counters just increment. Eventually the activity for all resources pegs at 65,535. Workaround: Configure the system as a 1-node cluster.

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

OK, why is this important? As OpenVMS systems/clusters continue to scale, the current limits/maximums often become problematic. For example, the lock activity is stored in word-length fields in the root RSB. The distributed lock manager uses these word-length values to make resource tree remastering decisions. What if one cluster node is generating 100,000 lock requests per 8 seconds against a resource tree, and another cluster node is generating 200,000 lock requests per 8 seconds against the same resource tree. Currently the distributed lock manager sees activity of 65,528 for both cluster nodes and therefore the resource tree does not move.

In addition, what if the database administrator is tuning a database. If a change results in a drop in locking rates from 150,000 per 8 seconds down to 100,000 per 8 seconds, this drop is currently invisible. The database administrator often cannot determine if a recent change has had a positive or negative effect.

Load is difficult to determine as well. Why is the cluster slower today? Maybe locking rates are higher (300,000 versus 100,000) but all we see from SDA> LCK SHOW ACTIVE is a maximum activity of 65,528.

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

There is another issue too. The activity values are calculated once every 8 seconds via a TQE (a time queue entry). This TQE fires every second, but only performs the lock activity calculations every 8th execution. While performing these activity calculations, the code (the LCK$RRSCAN routine) holds the LCKMGR spinlock. An interesting comment from the source code:

```
390 ;   Every 8 seconds all root RSBs are scanned to adjust
391 ;   the counters, and every minute all root RSBs are scanned
392 ;   to check for trees holding locks from a single node
393 ;   only. On a system with a large number of root RSBs
394 ;   (we have seen customers with more than 500K root RSBs)
395 ;   there can be a noticeable periodic stall for the end
396 ;   users.
397 ;   If we exceed a time slice (15 milliseconds has been proven
398 ;   to be a good value), store the required hash information away
399 ;   and attempt to continue the scan from here the next time.
400 ;   If the resource is gone, start from the beginning of the
401 ;   queue.

SDA> set fetch quad
SDA> validate queue/quad @LCK$GQ_RRSFL  <-- The Root Resource Queue Header
Queue is complete, total of 1262 elements in the queue
```

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

The root resource queue shown on the previous page is tiny at just 1262 elements. As the source code comments indicated, root resource queues can be large:

```
SDA> set fetch quad
SDA> validate queue/quad @LCK$GQ_RRSFL  <-- The Root Resource Queue Header
Queue is complete, total of 450176 elements in the queue
```

The next logical question becomes, how many root resources can be scanned in 15 milliseconds? The answer depends primarily on the clock rate of the CPUs. Via some testing with a diagnostic SYS$CLUSTER image, numbers in the 55,000 to 65,000 range were achieved. Therefore, on the system shown above, it might take 7 (maybe 8) passes to scan the entire root resource queue, and that assumes the root resource scan can successfully restart at the saved checkpoint. So rather than accumulating locking rates over an 8 second period, on systems with very large (and dynamic) root resource queues, the locking rates calculated may represent much longer periods of time. These systems may become resource masters when they should not be due to these artificially high locking rates.

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

Fortunately, OpenVMS engineering is currently investigating solutions for these issues. Can the word-length fields in the RSB structures be promoted to longwords? Can we scale the values in current word-length fields to represent locking rates over a one second period rather than over an eight second period? Can the root resource scan code be modified to use a longer time slice? Can the root resource scan code be modified to use a more robust check pointing technique? I guess we will have to stay tuned.

In the short term, I believe it is prudent to be aware of this situation. If one cluster node has an unusually large root resource queue, expect locking rates reported by that node to be artificially high. Possibly adjust LOCKDIRWT to balance the length of the root resource queues on each cluster node. Maybe lower LOCKRMWT on nodes with large root resource queues.

The next slide shows data captured from a 6-node cluster. What can we infer from this data? What tuning suggestions could we provide to this site?

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

```
SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 186946 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 62468 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 36552 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 13390 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 48302 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 450176 elements in the queue
```

Root RSB Lock Activity Counters: Are those Locking Rates Correct?

```
SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 186946 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 62468 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 36552 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 13390 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 48302 elements in the queue

SDA> validate queue/quad @LCK$GQ_RRSFL
Queue is complete, total of 450176 elements in the queue
```

The first and last node listed above will have artificially high locking rates reported. Check LOCKDIRWT settings on all cluster nodes. Possibly lower LOCKRMWT settings on the first and last node listed to “level the playing field” a bit.

CPUSPINWAIT crashes during Lock Directory Rebuild Operations

What, another issue with distributed locking? Yes, but fortunately this issue is extremely, extremely rare. On systems with very large numbers of locks and resources (say 15 million locks and 10 million resources), CPUSPINWAIT bugchecks have been reported during lock directory rebuild operations. The LCK$INIT_REBUILD code that performs this work must hold the LCKMGR and the IOLOCK8/SCS spinlocks as it walks down the lock ID table and the resource hash table. This work is performed on the PEDRIVER CPU (the CPU to which PEA0 is fast pathed to). If this code is preempted by higher IPL activity on the PEDRIVER CPU, these CPUSPINWAIT bugchecks become more likely.

On systems with very large numbers of locks and resources, OpenVMS engineering has made the following recommendations:

1. Increase SMP_LNGSPINWAIT from 30 to 40 seconds; maybe even to the maximum of 83 seconds.

2. Investigate what other devices (most likely SCS NICs) are fast pathed to the PEDRIVER CPU. Move some or all of these devices away from this CPU.

CPUSPINWAIT crashes during Lock Directory Rebuild Operations

3. Keep the number of network devices used for cluster communication to an optimal level (4 to 5 paths).

4. If changing SMP_LNGSPINWAIT, evaluate the RECNXINTERVAL setting.

To help identify systems that might be at risk, OpenVMS engineering will be making changes to the lock directory rebuild code. The code will now store three performance metrics in new global data cells: 1) the duration of the previous lock directory rebuild operation; 2) the maximum duration of any lock directory rebuild operation (since boot); and 3) the number of lock directory rebuilds (since boot). The names and interpretations of these three new global data cells are as follows:

1. LCK$GQ_RBDTIM - the duration of the last lock directory rebuild in OpenVMS quadword delta time format (SDA> exam/time LCK$GQ_RBDTIM, ignoring the 17-NOV-1858).

CPUSPINWAIT crashes during Lock Directory Rebuild Operations

2. LCK$GQ_MAX_RBDTIM - the maximum lock directory rebuild duration since boot in OpenVMS quadword delta time format (SDA> exam/time LCK$GQ_MAX_RBDTIM, again ignoring the 17-NOV-1858).

3. LCK$GL_NUM_RBDS - the number of lock directory rebuilds since boot.

Note: The maximum number of locks supported by any OpenVMS cluster node is 16,777,215 (00FFFFFF hex). Reaching this limit results in a “no lock ID available” error.

A Troubleshooting Tip; Always Consider Everything that has Changed

While working a recent case, troubleshooting technique played a pivotal role. In a 5-node OpenVMS V8.3/V8.4 cluster (two Alpha ES47s, two BL860 i2 Integrity servers and one rx2800 Integrity server), the rx2800 began crashing. The first crash was totally unexpected. Upon rebooting, the rx2800 would stay up until TCPIP was started. Shortly after TCPIP was started, the rx2800 would crash.

The crash types were variable: SSRVEXCEPTs, INVEXCEPTNs, NOTFCBFCBs, etc. What was common in all of the crashes was corruption in non-paged pool. Following a 2040 hex byte VCRP data structure, about 2240 hex bytes of non-paged pool appeared to have been overwritten by null characters. Depending on what data structure(s) had been overwritten determined the crash type.

Since this was a production cluster, the rx2800 was removed from the cluster and replaced with a BL860 i2 Integrity server. The crashes stopped immediately. The "suspect" rx2800 was moved into a test cluster where it continued to crash.

Is this likely a hardware problem or a software problem?

A Troubleshooting Tip; Always Consider Everything that has Changed

Swapped hardware, the crashes stopped, got to be a hardware problem, right? That is what I thought initially and we even went as far as swapping the NICs and replacing the system board in the “suspect” rx2800; however, the crashes continued. Could this be a software problem?

What else could have changed when a rx2800 was replaced with a BL860 i2? Isn’t a rx2800 a BL860 i2 in a box? Well, not exactly. Upon closer examination, the NICs were different and these different NICs used different device drivers. In the rx2800, the NICs were EIA, EIB, EIC, etc., and used SYS$EI1000 as their device driver. In the BL860 i2 the NICs were EWA, EWB, EWC, etc. and used SYS$EW57711 as their device driver. Could SYS$EI1000 be vulnerable to a problem/condition that SYS$EW57711 is NOT vulnerable to?

As a troubleshooting step, a diagnostic SYSTEM_PRIMITIVES image was provided. This diagnostic image added 2240 hex bytes to every allocation request of 2040 hex bytes from non-paged pool, thus allocating a 4280 hex byte packet. The theory was to allow space for the apparent packet overflow. With this diagnostic image in place, the crashes on the rx2800 stopped.

A Troubleshooting Tip; Always Consider Everything that has Changed

Further digging into the SYS$EI1000 device driver source code showed a problem in the way the driver was communicating the receive buffer size to the NIC. The driver was telling the NIC that the receive buffer was 9 KB; however, the actual receive buffer was under 8 KB. This was not a problem for “regular” sized packets, but for “jumbo” sized packets the potential for non-paged pool corruption was present.

From OpenVMS engineering:

```
“When TCPIP is started, it adds FF-FF-FF-FF-FF-FF to the multicast address table.  This
might be the reason why the problem is seen only after TCPIP is started.  This enabled the
reception of broadcast packets.  The “suspect” packet, protocol 89-22 from a VMware
system, was a packet addressed to FF-FF-FF-FF-FF-FF.”
```

What did I learn from this case; when a change is made and a particular problem is resolved, a prudent troubleshooter must consider everything that has changed as a result of that change.

i2 blade versus i4 blade Performance Tests

For the performance testing that I performed (and this was very basic and very limited performance testing), I used the cluster shown below. Each system had 32 GB of physical memory, each system was booted from a local SAS disk in the blade, and each server had a total of either 4 or 8 processor cores. The i2 blade had 1.33 GHz/4.0 MB CPUs and the i4 blades had 2.39 GHz/32.0 MB CPUs.

```
I4VMS1 $ show cluster

View of Cluster from system ID 1026  node: I4VMS1          2-SEP-2015 08:37:24
```

SYSTEMS		MEMBERS
NODE	SOFTWARE	STATUS
I4VMS1	VMS V8.4-1H1	MEMBER
I2VMS	VMS V8.4-1H1	MEMBER
I4VMS2	VMS V8.4-1H1	MEMBER

Overall, I was very impressed by the performance of the i4 blades. In some tests, I got more than double the performance on the i4 blades.

i2 blade versus i4 blade Performance Tests

For the first test I wanted to compare reboot times. Since the i2 server and the i4 server are configured identically the results should be fairly representative: a BL860 i2 blade versus a BL860 i4 blade, both 32 GB, both booting from local SAS drives, P410I controllers, and both reboots including a 30 second timeout at the boot options screen.

Multiple tests were performed. The time captured was the time from when I entered “$ reboot” from DCL at the console until the “SYSTEM job terminated” message was displayed on the console.

The BL860 i2 blade rebooted in an average of 351 seconds. Subtracting the 30 second timeout at the boot options screen results in 321 seconds.

The BL860 i4 blade rebooted in an average of 266 seconds. Subtracting the 30 second timeout at the boot options screen results in 236 seconds.

Using the i2 server reboot time as the baseline, the i4 server reboot time is 26.5% faster.

i2 blade versus i4 blade Performance Tests

For the second test a simple DCL script was used, and as you can see from the script below, I am certainly no DCL expert. Essentially the script loops for one minute incrementing two counters.

```
$ I = 0
$ J = 0
$ START=F$TIME()
$ !
$ TOP:
$ I = I + 1
$ J = J + 1
$ NOW=F$TIME()
$ DELTA = F$DELTA_TIME(START,NOW)
$ MINUTE = F$EXTRACT(9,1,DELTA)
$ IF (MINUTE .EQS. "1") THEN GOTO DONE
$ GOTO TOP
$ !
$ DONE:
$ SHOW SYMBOL START
$ SHOW SYMBOL NOW
$ SHOW SYM I
$ SHOW SYM J
$ EXIT
```

i2 blade versus i4 blade Performance Tests

The DCL script was run multiple times on the i2 and on the i4 blade servers. Hyper-threads were disabled. Typical runs are shown below.

```
I2VMS $ @counter_loop.com
  START = " 6-JUL-2015 16:16:41.08"
  NOW = " 6-JUL-2015 16:17:41.08"
  I = 480211    Hex = 000753D3  Octal = 00001651723
  J = 480211    Hex = 000753D3  Octal = 00001651723
I2VMS $
```

```
I4VMS1 $ @counter_loop.com
  START = " 7-JUL-2015 10:49:45.22"
  NOW = " 7-JUL-2015 10:50:45.22"
  I = 693586    Hex = 000A9552  Octal = 00002512522
  J = 693586    Hex = 000A9552  Octal = 00002512522
I4VMS1 $
```

Using the i2 server data as the baseline, the i4 server is 44.4% faster.

i2 blade versus i4 blade Performance Tests

For the next test I used a C program kindly provided by a highly-skilled system manager. The program allocates an array of 8,000,000 quadword pointers and then allocates and initializes 8,000,000 structures of 2 quadwords. It then walks the list in reverse order initializing the quadwords. Hyper-threads were enabled.

```
I2VMS $ run BE_EXAMPLE_C_LOOP.EXE
Elapsed time to populate array of 8000000 structures: 2.225756 seconds
Elapsed time to walk array and init 16000000 members: 0.452436 seconds
I2VMS $ run BE_EXAMPLE_C_LOOP.EXE
Elapsed time to populate array of 8000000 structures: 2.224068 seconds
Elapsed time to walk array and init 16000000 members: 0.458465 seconds
I2VMS $ run BE_EXAMPLE_C_LOOP.EXE
Elapsed time to populate array of 8000000 structures: 2.245774 seconds
Elapsed time to walk array and init 16000000 members: 0.453250 seconds

I4VMS1 $ run BE_EXAMPLE_C_LOOP.EXE
Elapsed time to populate array of 8000000 structures: 1.547299 seconds
Elapsed time to walk array and init 16000000 members: 0.262601 seconds
I4VMS1 $ run BE_EXAMPLE_C_LOOP.EXE
Elapsed time to populate array of 8000000 structures: 1.507329 seconds
Elapsed time to walk array and init 16000000 members: 0.262541 seconds
I4VMS1 $ run BE_EXAMPLE_C_LOOP.EXE
Elapsed time to populate array of 8000000 structures: 1.315525 seconds
Elapsed time to walk array and init 16000000 members: 0.262431 seconds
```

i2 blade versus i4 blade Performance Tests

I ran the C program multiple times in succession. I found it interesting that successive runs on the i2 blade produces similar run times; however, during successive runs on the i4 blade, the run times improved a bit, by about 15%.

Comparing the initial run time on an i2 blade versus the initial run time on an i4 blade, the i4 blade was about 32.4% faster. Comparing the best run on the i4 blade versus the best run time on the i2 blade, the i4 blade was about 41% faster.

The final test in this series of tests was performed using the SPIN.EXE program provided by OpenVMS engineering. This program loops in kernel mode acquiring and then immediately releasing a particular spinlock as rapidly as possible. For these tests I had four CPUs competing for the IOLOCK8/SCS spinlock. Hyper-threads were disabled. The results showed that the i4 blades were about 108% faster; more than double the performance on the i2 blades. A repeat test showed almost a 200% improvement in performance.

i2 blade versus i4 blade Performance Tests

I2 Blade Test

```
OpenVMS Operating System, Version V8.4-1H1 -- System Dump Analysis   25-JUN-2015
Debug Trace Information:

Timestamp                CPU Buffer
------------------------ --- ------------------
25-JUN 16:36:31.550900    03 Acquires = 3609246
25-JUN 16:36:31.548601    00 Acquires = 3766070
25-JUN 16:36:31.548601    02 Acquires = 3630493
25-JUN 16:36:31.548600    01 Acquires = 3703832
```

I4 Blade Test

```
OpenVMS Operating System, Version V8.4-1H1 -- System Dump Analysis   25-JUN-2015
Debug Trace Information:

Timestamp                CPU Buffer
------------------------ --- ------------------
25-JUN 22:35:30.081070    00 Acquires = 7765056
25-JUN 22:35:30.081046    03 Acquires = 7771477
25-JUN 22:35:30.081046    02 Acquires = 7553359
25-JUN 22:35:30.081046    01 Acquires = 7471990
```

i2 blade versus i4 blade Performance Tests

I2 Blade Test - Repeat

```
OpenVMS Operating System, Version V8.4-1H1 -- System Dump Analysis    1-JUL-2015
Debug Trace Information:

Timestamp                CPU Buffer
------------------------ --- ------------------
 1-JUL 15:05:17.892320   01 Acquires = 3720490
 1-JUL 15:05:17.889719   03 Acquires = 3618733
 1-JUL 15:05:17.889719   00 Acquires = 3752892
 1-JUL 15:05:17.889718   02 Acquires = 3645858
```

I4 Blade Test - Repeat

```
OpenVMS Operating System, Version V8.4-1H1 -- System Dump Analysis    1-JUL-2015
Debug Trace Information:

Timestamp                CPU Buffer
------------------------ --- -------------------
 1-JUL 15:16:35.936724   03 Acquires = 10364656
 1-JUL 15:16:35.936724   01 Acquires =  9889580
 1-JUL 15:16:35.936724   00 Acquires = 10292958
 1-JUL 15:16:35.936724   02 Acquires = 10230450
```

i2 blade versus i4 blade Performance Tests

I have never been a big fan of hyper-threads. I suppose there are those situations were hyper-threads being enabled improves performance, but I have to believe those situations are in the minority.

For the tests on the previous slides, I did repeat the tests with hyper-threads enabled if they were originally disabled and with hyper-threads disabled if they were originally enabled. A quick summary of those results are provided below.

Enabling hyper-threads on an i2 blade costs you about .87% in performance.

Enabling hyper-threads on an i4 blade costs you about .81% in performance.

Running a simple DCL procedure, on co-thread CPUs on an i2 blade costs you about 1.37% in performance.

Running a simple DCL procedure, on co-thread CPUs on an i4 blade costs you about 2.59% in performance.

i2 blade versus i4 blade Performance Tests

Running a C program with heavy memory accesses, on co-thread CPUs on an i2 blade gains you about 5.43% in performance.

Running a C program with heavy memory accesses, on co-thread CPUs on an i4 blade costs you about 8.65% in performance.

As was mentioned at the beginning of this topic, these performance tests were very basic and very limited. Also please note that the i2 blade used in these tests had 1.33GHz/4.0MB CPUs which are not the fastest i2 blade CPUs. i4 blade CPUs vary in clock rate as well.

For additional i4 blade performance data, I would recommend speaking with the folks at VMS Software Inc.

Speaker Contact Information

rob.eulenstein@hp.com (rob.eulenstein@hpe.com)

thilo.lauer@hp.com (thilo.lauer@hpe.com)