Basics of Kernel Crash Dump Analysis

Agenda:

- Basics of kernel crash dump analysis
- Initial analysis of kernel crash dump
- Initial analysis of memory sub-system

What are the tools/data required to perform a kernel crash dump analysis ?

a) **crash utility** (/usr/bin/crash)

- Provided by "crash" package.
  Eg: crash-6.0.4-2.el6.x86_64.rpm

- No cross platform support.
  o Use 32 bit version of crash for vmcore capture from 32 bit kernel.
  o Use 64 bit version of crash for vmcore captured from 64 bit kernel.
  o Similarly for s390, s390x and PPC.

How to install ?

- Install it using yum or rpm command.

```
# yum install crash
```

OR

```
# rpm -ivh crash-6.0.4-2.el6.x86_64.rpm
```

- Compile from upstream source code (not supported by Red Hat Support):

```
# git clone https://github.com/crash-utility/crash.git
# cd crash/
# make
# make install
```

b) Kernel symbol file (vmlinux) of the crashed kernel

- Provided by "kernel-debuginfo" package.
  Eg: kernel-debuginfo-2.6.32-431.el6.x86_64.rpm

- The version and arch of kernel-debuginfo package must match with the version of kernel from which vmcore was captured.

How to install ?

- Subscribe to "rhel-6-server-debug-rpms" channel.

```
# yum-config-manager --enable rhel-6-server-debug-rpms
# subscription-manager repos --enable rhel-6-server-debug-rpms
```

- Install it using yum or debuginfo-install command:

```
# yum install kernel-debuginfo-2.6.32-431.el6
# debuginfo-install kernel-debuginfo-2.6.32-431.el6
```

OR

- Download kernel-debuginfo-common and kernel-debuginfo rpm packages from Red Hat Customer Portal and install it using rpm command.

```
# rpm -ivh kernel-debuginfo-common-2.6.32-431.el6.x86_64.rpm
# rpm -ivh kernel-debuginfo-2.6.32-431.el6.x86_64.rpm
```

How to use (vmlinux) without installing kernel-debuginfo package on the system ?

- Extract the kernel-debuginfo rpm package using rpm2cpio command.

```
# rpm2cpio kernel-debuginfo-2.6.32-431.el6.x86_64.rpm | cpio -idv
```

c) A machine of same architecture as of kernel from which vmcore was captured.

- A kernel crash dump (vmcore) captured from a x86_64 machine can only be viewed on a x86_64 machine.

- Similarly, a kernel crash dump (vmcore) captured from a s390 machine can only be viewed on a s390 machine.

d) kernel crash dump file (vmcore)

- Captured using kdump/diskdump/netdump/xendump/LKCD/vmss2core mechanisms.

e) Source code of the crashed kernel (optional)

- Provided by "src" rpm package of kernel.
  Eg: kernel-2.6.32-431.el6.src.rpm

- The version of "src" rpm package of kernel must match with the version of kernel from which vmcore was captured.

How to install ?

- Install it using yum or rpm command.

```
# yum install kernel-src
# rpm -ivh kernel-2.6.32-431.el6.src.rpm
```

OR

- Extract the kernel-src rpm package using rpm2cpio command.

```
# rpm2cpio kernel-2.6.32-431.el6.src.rpm | cpio -idv
```

How to open a kernel crash dump (vmcore) for analysis ?

a) **Typical postmortem debugging:** [Offline mode]

Syntax:

```
# crash /path/to/vmlinux /path/to/vmcore
```

- Kernel object file and memory image are supplied, respectively.

```
# crash --osrelease vmcore
2.6.32-431.el6.x86_64

# crash -d 1 vmcore | grep RELEASE
OSRELEASE=2.6.32-431.el6.x86_64

# crash /var/crash/vmcore /usr/lib/debug/lib/modules/2.6.32-431.el6.x86_64/vmlinux
```

b) **Live memory debugging:** [Online mode]

Syntax:

```
# crash /path/to/vmlinux
```

- /dev/crash used by default for live memory image.

```
# crash  /usr/lib/debug/lib/modules/2.6.32-431.el6.x86_64/vmlinux
```

c) **Live memory debugging** (with vmlinux search): [Online mode]

Syntax:

```
# crash
```

- Predefined directories are searched for proper vmlinux
- Version string matched to the running kernel (/proc/version)

What are the basic commands of crash utility ?

- **sys** : Display essential system information.
- **log** : Display the kernel ring buffer log.
- **bt** : Display a kernel stack backtrace.
- **bt -f** : Display all stack data contained in a frame.
- **dis** : Disassemble a function or symbol.
- **kmem -i** : Displays information about the use of kernel memory.
- **kmem -s** : Displays basic kmalloc() slab data.
- **ipcs** : System V IPC facilities.
- **ps** : Displays processes in the system.
- **runq** : Displays the tasks on the run queues of each cpu.
- **mount** : Displays information about the mounted filesystems.
- **files** : Display the open files of the current context.
- **dev** : Display character and block device data.
- **dev -d** : Display disk I/O statistics.
- **net** : Display the system's network device list.
- **irq** : Display IRQ data.
- **mod** : Display the currently-installed modules.
- **mod -t** : Display modules that are "tainted".
- **exit** : Exit the crash session.
- **extend** : Dynamically loads or unloads crash extension shared object.
- **help** : Get help.

Initial Analysis of Kernel Crash Dump

How to check basic system information ?

- "sys" command displays essential system information.

```
crash> sys
      KERNEL: /usr/lib/debug/lib/modules/2.6.32-431.el6.x86_64/vmlinux
    DUMPFILE: /var/crash/127.0.0.1-2016-01-21-17:45:34/vmcore
        CPUS: 4
        DATE: Thu Jan 21 17:45:30 2016
      UPTIME: 00:02:01
LOAD AVERAGE: 0.49, 0.30, 0.11
       TASKS: 264
    NODENAME: hvyas.example.com
     RELEASE: 2.6.32-431.el6.x86_64
     VERSION: #1 SMP Sun Nov 10 22:19:54 EST 2013
     MACHINE: x86_64  (2790 Mhz)
      MEMORY: 5.8 GB
       PANIC: "SysRq : Trigger a crash"
```

How to check hardware information ?

- The option (-i) of "sys" command dumps the Desktop Management Interface (DMI) string data if available in the kernel.

```
crash> sys -i
        DMI_BIOS_VENDOR: Seabios
       DMI_BIOS_VERSION: 0.5.1
          DMI_BIOS_DATE: 01/01/2007
        DMI_SYS_VENDOR: Red Hat
       DMI_PRODUCT_NAME: KVM
    DMI_PRODUCT_VERSION: RHEL 6.5.0 PC
     DMI_PRODUCT_SERIAL:
       DMI_PRODUCT_UUID: 3D51C070-C307-BD3A-281D-CA1A689C22C6
     DMI_CHASSIS_VENDOR: Red Hat
       DMI_CHASSIS_TYPE: 1
    DMI_CHASSIS_VERSION:
     DMI_CHASSIS_SERIAL:
  DMI_CHASSIS_ASSET_TAG:
     DMI_SMBIOS_VERSION: 2.4


OR

crash> log | grep DMI:
DMI: Red Hat KVM, BIOS 0.5.1 01/01/2007
```

How to check kernel ring buffer (dmesg) ?

- "log" or "dmesg" command dumps the kernel log_buf contents.

```
crash> log | tail -n 38
SysRq : Trigger a crash
BUG: unable to handle kernel NULL pointer dereference at (null)
IP: [<ffffffff8134b6c6>] sysrq_handle_crash+0x16/0x20
PGD 140aff067 PUD 140a9f067 PMD 0
Oops: 0002 [#1] SMP
last sysfs file: /sys/devices/pci0000:00/0000:00:01.2/usb1/1-1/speed
CPU 2
Modules linked in: nfsd lockd nfs_acl auth_rpcgss exportfs autofs4 sunrpc bnx2fc
cnic uio fcoe 8021q libfcoe garp stp libfc llc scsi_transport_fc scsi_tgt
xt_NFQUEUE iptable_filter ip_tables ip6t_REJECT nf_conntrack_ipv6 nf_defrag_ipv6
xt_state nf_conntrack ip6table_filter ip6_tables ipv6 uinput microcode
virtio_balloon virtio_net snd_hda_intel snd_hda_codec snd_hwdep snd_seq
snd_seq_device snd_pcm snd_timer snd soundcore snd_page_alloc i2c_piix4 i2c_core
ext4 jbd2 mbcache virtio_blk pata_acpi ata_generic ata_piix virtio_pci virtio_ring
virtio dm_mirror dm_region_hash dm_log dm_mod [last unloaded: speedstep_lib]

Pid: 2187, comm: bash Not tainted 2.6.32-431.el6.x86_64 #1 Red Hat KVM
RIP: 0010:[<ffffffff8134b6c6>]  [<ffffffff8134b6c6>] sysrq_handle_crash+0x16/0x20
RSP: 0018:ffff8801422bde18  EFLAGS: 00010096
RAX: 0000000000000010 RBX: 0000000000000063 RCX: 0000000000000000
RDX: 0000000000000000 RSI: 0000000000000000 RDI: 0000000000000063
RBP: ffff8801422bde18 R08: 0000000000000000 R09: 203a207152737953
R10: 0000000000000000 R11: 0000000000000000 R12: 0000000000000000
R13: ffffffff81b01a40 R14: 0000000000000286 R15: 0000000000000007
FS:  00007ffdf778f700(0000) GS:ffff880028280000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
CR2: 0000000000000000 CR3: 0000000140815000 CR4: 00000000000006e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process bash (pid: 2187, threadinfo ffff8801422bc000, task ffff88018d4e8ae0)
Stack:
 ffff8801422bde68 ffffffff8134b982 ffff88018d4e8ae0 ffff880100000000
<d> 0000000000000300 0000000000000002 ffff880140ace3c0 00007ffdf7799000
<d> 0000000000000002 fffffffffffffffb ffff8801422bde98 ffffffff8134ba3e
Call Trace:
 [<ffffffff8134b982>] __handle_sysrq+0x132/0x1a0
 [<ffffffff8134ba3e>] write_sysrq_trigger+0x4e/0x50
 [<ffffffff811f328e>] proc_reg_write+0x7e/0xc0
 [<ffffffff81188f78>] vfs_write+0xb8/0x1a0
 [<ffffffff81189871>] sys_write+0x51/0x90
 [<ffffffff8100b072>] system_call_fastpath+0x16/0x1b
Code: d0 88 81 a3 1c fe 81 c9 c3 66 66 66 2e 0f 1f 84 00 00 00 00 00 55 48 89 e5 0f
1f 44 00 00 c7 05 0d 07 75 00 01 00 00 00 0f ae f8 <c6> 04 25 00 00 00 00 01 c9 c3
55 48 89 e5 0f 1f 44 00 00 8d 47
RIP  [<ffffffff8134b6c6>] sysrq_handle_crash+0x16/0x20
 RSP <ffff8801422bde18>
CR2: 0000000000000000
```

How to determine the panic task ?

- The option (-p) of "set" command sets the context to the panic task, or back to the crash task on a live system.

```
crash> set -p
    PID: 2187
COMMAND: "bash"
   TASK: ffff88018d4e8ae0  [THREAD_INFO: ffff8801422bc000]
    CPU: 2
  STATE: TASK_RUNNING (SYSRQ)
```

How to check the backtrace of panic task ?

- "bt" command displays a kernel stack backtrace. If no arguments are given, the stack trace of the current context will be displayed.

```
crash> bt
PID: 2187   TASK: ffff88018d4e8ae0  CPU: 2   COMMAND: "bash"
 #0 [ffff8801422bd9e0] machine_kexec at ffffffff81038f3b
 #1 [ffff8801422bda40] crash_kexec at ffffffff810c5d92
 #2 [ffff8801422bdb10] oops_end at ffffffff8152b510
 #3 [ffff8801422bdb40] no_context at ffffffff8104a00b
 #4 [ffff8801422bdb90] __bad_area_nosemaphore at ffffffff8104a295
 #5 [ffff8801422bdbe0] bad_area at ffffffff8104a3be
 #6 [ffff8801422bdc10] __do_page_fault at ffffffff8104ab6f
 #7 [ffff8801422bdd30] do_page_fault at ffffffff8152d45e
 #8 [ffff8801422bdd60] page_fault at ffffffff8152a815
    [exception RIP: sysrq_handle_crash+22]
    RIP: ffffffff8134b6c6  RSP: ffff8801422bde18  RFLAGS: 00010096
    RAX: 0000000000000010  RBX: 0000000000000063  RCX: 0000000000000000
    RDX: 0000000000000000  RSI: 0000000000000000  RDI: 0000000000000063
    RBP: ffff8801422bde18   R8: 0000000000000000   R9: 203a207152737953
    R10: 0000000000000000  R11: 0000000000000000  R12: 0000000000000000
    R13: ffffffff81b01a40  R14: 0000000000000286  R15: 0000000000000007
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
 #9 [ffff8801422bde20] __handle_sysrq at ffffffff8134b982
#10 [ffff8801422bde70] write_sysrq_trigger at ffffffff8134ba3e
#11 [ffff8801422bdea0] proc_reg_write at ffffffff811f328e
#12 [ffff8801422bdef0] vfs_write at ffffffff81188f78
#13 [ffff8801422bdf30] sys_write at ffffffff81189871
#14 [ffff8801422bdf80] system_call_fastpath at ffffffff8100b072
    RIP: 0000003e2c2db560  RSP: 00007fff425f6548  RFLAGS: 00010202
    RAX: 0000000000000001  RBX: ffffffff8100b072  RCX: 00000000004b5e34
    RDX: 0000000000000002  RSI: 00007ffdf7799000  RDI: 0000000000000001
    RBP: 00007ffdf7799000   R8: 000000000000000a   R9: 00007ffdf778f700
    R10: 0000000000000000  R11: 0000000000000246  R12: 0000000000000002
    R13: 0000003e2c58e7a0  R14: 0000000000000002  R15: 0000003e2c58e7a0
```

```
    ORIG_RAX: 0000000000000001  CS: 0033  SS: 002b
```

How to display parental hierarchy of a process ?

- The (-p) option of "ps" command displays the parental hierarchy of selected, or all, tasks.

```
crash> ps -p 2187
PID: 0      TASK: ffffffff81a8d020  CPU: 0   COMMAND: "swapper"
 PID: 1      TASK: ffff880192d39500  CPU: 2   COMMAND: "init"
  PID: 1865   TASK: ffff88014cf00040  CPU: 1   COMMAND: "sshd"
   PID: 1877   TASK: ffff88018d606040  CPU: 2   COMMAND: "sshd"
    PID: 2161   TASK: ffff8801420beaa0  CPU: 1   COMMAND: "bash"
     PID: 2187   TASK: ffff88018d4e8ae0  CPU: 2   COMMAND: "bash"
```

How to display the child task of a process ?

- The (-c) option of "ps" command displays the children of selected, or all, tasks.

```
crash> ps -c 2187
PID: 2187   TASK: ffff88018d4e8ae0  CPU: 2   COMMAND: "bash"
  (no children)

crash> ps -c 2161
PID: 2161   TASK: ffff8801420beaa0  CPU: 1   COMMAND: "bash"
  PID: 2187   TASK: ffff88018d4e8ae0  CPU: 2   COMMAND: "bash"
```

How to check list of open files by panic task ?

- "files" command displays the open files of the current context.

```
crash> files 2187
PID: 2187   TASK: ffff88018d4e8ae0  CPU: 2   COMMAND: "bash"
ROOT: /    CWD: /root
 FD       FILE            DENTRY           INODE       TYPE PATH
  0 ffff880140ac6d40 ffff880141057500 ffff880141037d58 CHR  /dev/pts/0
  1 ffff880140ace3c0 ffff8801410738c0 ffff880141043078 REG  /proc/sysrq-trigger
  2 ffff880140ac6d40 ffff880141057500 ffff880141037d58 CHR  /dev/pts/0
 10 ffff880140ac6d40 ffff880141057500 ffff880141037d58 CHR  /dev/pts/0
254 ffff88018cc71500 ffff8801434ef240 ffff88014104c4b8 REG  /root/system_crash.sh
255 ffff880140ac6d40 ffff880141057500 ffff880141037d58 CHR  /dev/pts/0
```

How to check task priority and policy ?

- "task" command can used to determine the task priority and policy.
- "task" command displays the contents of a task's task_struct and thread_info structures.

```
crash> set -p
    PID: 2187
COMMAND: "bash"
   TASK: ffff88018d4e8ae0  [THREAD_INFO: ffff8801422bc000]
    CPU: 2
  STATE: TASK_RUNNING (SYSRQ)

crash> task 2187 -R policy, prio, rt_priority
PID: 2187   TASK: ffff88018d4e8ae0  CPU: 2   COMMAND: "bash"
  policy = 0,
  prio = 120,
  rt_priority = 0,
```

Scheduling policies:

```
#define SCHED_NORMAL        0
#define SCHED_FIFO          1
#define SCHED_RR            2
#define SCHED_BATCH         3
#define SCHED_IDLE          5
```

OR

```
crash> set -p
    PID: 2187
COMMAND: "bash"
   TASK: ffff88018d4e8ae0  [THREAD_INFO: ffff8801422bc000]
    CPU: 2
  STATE: TASK_RUNNING (SYSRQ)

crash> task_struct.policy,prio,rt_priority 0xffff88018d4e8ae0
  policy = 0
  prio = 120
  rt_priority = 0
```

How to check the command line arguments and environment strings of task ?

- The option (-a) of "ps" command displays the argument and environment data for the task.

```
crash> ps -a automount
PID: 3948   TASK: f722ee30  CPU: 0   COMMAND: "automount"
ARG: /usr/sbin/automount --timeout=60 /net program /etc/auto.net
ENV: SELINUX_INIT=YES
     CONSOLE=/dev/console
     TERM=linux
     INIT_VERSION=sysvinit-2.85
     PATH=/sbin:/usr/sbin:/bin:/usr/bin
     LC_MESSAGES=en_US
     RUNLEVEL=3
     runlevel=3
     PWD=/
     LANG=ja_JP.UTF-8
     PREVLEVEL=N
     previous=N
     HOME=/
     SHLVL=2
     _=/usr/sbin/automount
```

Note: This is information is only available if user-space contents are not filtered from kernel crash dump.

How to determine resource limits (rlimits) of a process ?

- The option (-r) of "ps" command displays resource limits (rlimits) of selected, or all, tasks.

```
crash> set 1
    PID: 1
COMMAND: "init"
   TASK: ffff880192d39500  [THREAD_INFO: ffff880192d3a000]
    CPU: 2
  STATE: TASK_INTERRUPTIBLE

crash> ps -r 1
PID: 1      TASK: ffff880192d39500  CPU: 2   COMMAND: "init"
      RLIMIT     CURRENT       MAXIMUM
         CPU   (unlimited)   (unlimited)
       FSIZE   (unlimited)   (unlimited)
        DATA   (unlimited)   (unlimited)
       STACK    10485760     (unlimited)
        CORE         0       (unlimited)
         RSS   (unlimited)   (unlimited)
       NPROC       45331         45331
      NOFILE       1024          4096
     MEMLOCK       65536         65536
          AS   (unlimited)   (unlimited)
       LOCKS   (unlimited)   (unlimited)
  SIGPENDING       45331         45331
    MSGQUEUE      819200        819200
        NICE         0             0
      RTPRIO         0             0
      RTTIME   (unlimited)   (unlimited)
```

How to determine total number of tasks in different state ?

- The option (-S) of "ps" command displays a summary consisting of the number of tasks in a task state.

```
crash> ps -S
  RU: 5
  IN: 259
```

How to display only user space process ?

- The option (-u) of "ps" command displays only user tasks.

```
crash> ps -u | head
   PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
     1       0   2  ffff880192d39500  IN   0.0   19364   1500  init
   483       1   1  ffff88018d76a040  IN   0.0   11240   1320  udevd
  1336       1   3  ffff88014cf00aa0  IN   0.0   27640    864  auditd
  1337       1   0  ffff88018e46f540  IN   0.0   27640    864  auditd
  1361       1   0  ffff88018cde6080  IN   0.0  249092   1628  rsyslogd
  1362       1   3  ffff880191be5540  IN   0.0  249092   1628  rs:main Q:Reg
  1363       1   2  ffff88018d49c080  IN   0.0  249092   1628  rsyslogd
  1364       1   3  ffff88018e796aa0  IN   0.0  249092   1628  rsyslogd
  1396       1   2  ffff88018d4d4080  IN   0.0   18976    924  rpcbind
```

How to display only kernel threads ?

- The option (-k) of "ps" command displays only kernel threads.

```
crash> ps -k | head
   PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
>     0       0   0  ffffffff81a8d020  RU   0.0       0      0  [swapper]
>     0       0   1  ffff880192d81540  RU   0.0       0      0  [swapper]
      0       0   2  ffff880192d8a040  RU   0.0       0      0  [swapper]
>     0       0   3  ffff880192dc2aa0  RU   0.0       0      0  [swapper]
      2       0   1  ffff880192d38aa0  IN   0.0       0      0  [kthreadd]
      3       2   0  ffff880192d38040  IN   0.0       0      0  [migration/0]
      4       2   0  ffff880192d67540  IN   0.0       0      0  [ksoftirqd/0]
      5       2   0  ffff880192d66ae0  IN   0.0       0      0  [migration/0]
      6       2   0  ffff880192d66080  IN   0.0       0      0  [watchdog/0]
```

How to check the total time of a process in a specific state ?

- The option (-m) of "ps" command displays the timestamp into days, hours, minutes, seconds, and milliseconds since the task was last run on a cpu.

```
crash> ps -m | grep RU
[0 00:00:00.000] [RU]  PID: 2187   TASK: ffff88018d4e8ae0  CPU: 2   COMMAND: "bash"
[0 00:02:01.177] [RU]  PID: 0      TASK: ffffffff81a8d020  CPU: 0   COMMAND: "swapper"
[0 00:02:01.409] [RU]  PID: 0      TASK: ffff880192d81540  CPU: 1   COMMAND: "swapper"
[0 00:02:01.408] [RU]  PID: 0      TASK: ffff880192d8a040  CPU: 2   COMMAND: "swapper"
[0 00:02:01.408] [RU]  PID: 0      TASK: ffff880192dc2aa0  CPU: 3   COMMAND: "swapper"
```

How to check run queue of each CPU ?

- "runq" command displays the tasks on a CFS run queue:

```
crash>  runq
CPU 0 RUNQUEUE: ffff880028216840
  CURRENT: PID: 0      TASK: ffffffff81a8d020  COMMAND: "swapper"
  RT PRIO_ARRAY: ffff8800282169c8
     [no tasks queued]
  CFS RB_ROOT: ffff8800282168d8
     [no tasks queued]

CPU 1 RUNQUEUE: ffff880028256840
  CURRENT: PID: 0      TASK: ffff880192d81540  COMMAND: "swapper"
  RT PRIO_ARRAY: ffff8800282569c8
     [no tasks queued]
  CFS RB_ROOT: ffff8800282568d8
     [no tasks queued]

CPU 2 RUNQUEUE: ffff880028296840
  CURRENT: PID: 2187   TASK: ffff88018d4e8ae0  COMMAND: "bash"
  RT PRIO_ARRAY: ffff8800282969c8
     [no tasks queued]
  CFS RB_ROOT: ffff8800282968d8
     [no tasks queued]

CPU 3 RUNQUEUE: ffff8800282d6840
  CURRENT: PID: 0      TASK: ffff880192dc2aa0  COMMAND: "swapper"
  RT PRIO_ARRAY: ffff8800282d69c8
     [no tasks queued]
  CFS RB_ROOT: ffff8800282d68d8
     [no tasks queued]
```

How to determine the values of sysctl parameter from vmcore ?

- Check the ".data" field of desired sysctl parameter in kernel socurce sysctl.c file.

Source File: kernel-2.6.32-431.el6/kernel/sysctl.c

```
{
        .ctl_name       = VM_SWAPPINESS,
        .procname       = "swappiness",
        .data           = &vm_swappiness,
        .maxlen         = sizeof(vm_swappiness),
        .mode           = 0644,
        .proc_handler   = &proc_dointvec_minmax,
        .strategy       = &sysctl_intvec,
        .extra1         = &zero,
        .extra2         = &one_hundred,
    },
```

```
crash> vm_swappiness
vm_swappiness = $1 = 60
```

Source File: kernel-2.6.32-431.el6/kernel/sysctl.c

```
{
        .ctl_name       = VM_PANIC_ON_OOM,
        .procname       = "panic_on_oom",
        .data           = &sysctl_panic_on_oom,
        .maxlen         = sizeof(sysctl_panic_on_oom),
        .mode           = 0644,
        .proc_handler   = &proc_dointvec,
    },
```

```
crash> sysctl_panic_on_oom
sysctl_panic_on_oom = $2 = 1
```

Initial Analysis of Memory Subsystem

How to check overall memory usage on the system ?

- "kmem -i" command displays general memory usage information:

```
crash> kmem -i
                 PAGES        TOTAL      PERCENTAGE
    TOTAL MEM  1205838       4.6 GB         ----
         FREE  1102506       4.2 GB   91% of TOTAL MEM
         USED   103332     403.6 MB    8% of TOTAL MEM
       SHARED    14484      56.6 MB    1% of TOTAL MEM
      BUFFERS     6377      24.9 MB    0% of TOTAL MEM
       CACHED    39960     156.1 MB    3% of TOTAL MEM
         SLAB    17725      69.2 MB    1% of TOTAL MEM

   TOTAL SWAP    16382        64 MB         ----
    SWAP USED        0            0    0% of TOTAL SWAP
    SWAP FREE    16382        64 MB  100% of TOTAL SWAP

 COMMIT LIMIT   616741       2.4 GB         ----
    COMMITTED    59755     233.4 MB    9% of TOTAL LIMIT
```

How to check per-zone memory statistics ?

- "kmem -z" command displays per-zone memory statistics.

```
crash> kmem -z
NODE: 0  ZONE: 0  ADDR: ffff880000010000  NAME: "DMA"
  SIZE: 4095  PRESENT: 3831  MIN/LOW/HIGH: 42/52/63
  VM_STAT:
                NR_FREE_PAGES: 3930
             NR_INACTIVE_ANON: 0
               NR_ACTIVE_ANON: 0
             NR_INACTIVE_FILE: 0
               NR_ACTIVE_FILE: 0
               NR_UNEVICTABLE: 0
                     NR_MLOCK: 0
              NR_ANON_PAGES: 0
             NR_FILE_MAPPED: 0
              NR_FILE_PAGES: 0
              NR_FILE_DIRTY: 0
               NR_WRITEBACK: 0
         NR_SLAB_RECLAIMABLE: 0
       NR_SLAB_UNRECLAIMABLE: 0
                NR_PAGETABLE: 0
             NR_KERNEL_STACK: 0
             NR_UNSTABLE_NFS: 0
```

How to determine memory usage in user-space ?

- The "ps -Gu" command can be used to determine the RSS of user-space tasks.

```
crash> ps -Gu | sed 's/>//g' | awk '{ total += $8 } END { printf "Total RSS
of user-mode: %.02f GiB\n", total/2^20 }'
Total RSS of user-mode: 30.74 GiB
```

- Per process memory usage:

```
crash> ps -G | sort -k 8,8 -n -r | head
   PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS     COMM
> 28882  20109   1  ffff880831e0e040  RU  92.8 65272428 32120772  rsession
>  6985   6984   2  ffff88082e934040  RU   0.0   641664   4680     coda
>  6392      1   3  ffff88082e934ab0  RU   0.0   167336   6324     perfd
   6123      1   2  ffff88082c194ab0  IN   0.1   238480  45732     splunkd
   6946   6945   1  ffff88082f804ab0  RU   0.0   833744   9188     opcmona
  27387      1   0  ffff88082d288040  RU   0.0   129952   6472     bgsagent
   6275      1   0  ffff88082f7eaab0  UN   0.0    36284   5184     scopeux
   2098   2097   3  ffff88082dbe7520  IN   0.0  1454328   5172     python
   6981   6980   1  ffff880831017520  RU   0.0   903548   3236     opcmsga
   2151      1   3  ffff88082daed520  IN   0.0   706320   2748     EracentEUAServi
```

- Specific process's memory usage:

```
crash> ps -Gu rsession | tail -n +2 | cut -b2- | gawk '{mem += $8} END
{print "Total " mem/1048576 " GB"}'
Total 30.6334 GB
```

How to display information for each configured swap device ?

- "swap" command displays information for each configured swap device.

```
crash> swap
SWAP_INFO_STRUCT    TYPE        SIZE        USED      PCT   PRI   FILENAME
ffff88014be681c0    FILE       65528k       0k         0%    -1   /swapfile
```

How to determine memory usage in kernel-space ?

- "kmem -s" command displays basic kmalloc() slab data.

```
crash> kmem -s | grep -e NAME -e anon_vma_chain
CACHE            NAME                 OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE
ffff88043bf20b00 anon_vma_chain            48  302042720  302042818 3922634     4k
```

Formula: (OBJSIZE * ALLOCATED) / 2^30

```
crash> !bc -q
scale =2
(48*302042720)/2^30
13.50
```

- Memory allocated to "anon_vma_chain" slab object is ~13.50 GiB.

How to check memory allocated for hugepages ?

RHEL-6:

- "kmem -h" command displays the address of hugepage hstate array entries, along with their hugepage size, total and free counts, and name.

```
crash> kmem -h
     HSTATE         SIZE     FREE    TOTAL  NAME
ffffffff81fbb8c0     2MB       10       10  hugepages-2048kB
```

RHEL-5:

a) Total Number of Huge pages.

```
crash> p -d nr_huge_pages
nr_huge_pages = $1 = 35845

crash> p -d nr_huge_pages*(1<<21)
$2 = 75172413440

crash> !bc -q
scale=2
75172413440/2^30
70.00
```

b) The number of huge pages in the pool that are not yet allocated.

```
crash> p -d free_huge_pages
free_huge_pages = $2 = 0
```

How to check memory allocated to VMware ballooning driver (RHEL 6+) ?

- Determine the address of symbol balloon.

```
crash> sym balloon
ffffffffa002b600 (b) balloon [vmware_balloon]
```

- Determine the value of "size" variable using the address of symbol balloon.

```
crash> struct vmballoon.size 0xffffffffa002b600
$4 = 2721049

crash> !bc -q
scale=2
2721049*4/2^20
10.37
```

2721049 pages = 2721049 x 4 = 10884196 KiB = 10.37 GiB

Note: The value of size variable is the amount of memory allocated by VMware Ballooning driver in pages.

- Determine the value of "target" variable using the address of symbol balloon.

```
crash> vmballoon.target 0xffffffffa002b600
$5 = 3177311

crash> !bc -q
scale=2
3177311*4/2^20
12.12
```

3177311 pages = 3177311 x 4 = 12709244 KiB = 12.12 GiB

Note: The value of target variable is the amount of memory needed by VMware Ballooning driver in pages.

Questions ?