

Xen Crash Dumps on Solaris

Nils Nieuwejaar

Senior Staff Engineer

Solaris Kernel Technology

Debugging Principles

- The most important bugs happen:
 - > at customer sites
 - > on mission-critical production systems
 - > are not easily reproducible
- So, we must be able to:
 - > reliably get a crash dump on each failure
 - > analyze that dump on a different system
 - > be able to root-cause and fix the problem from the information in that one dump

Solaris Panic Handling

- Capture state of panicking CPU
- Stop all other CPUs
- Dump crash information to raw disk partition
 - > Crash header
 - > Symbol table
 - > Per-page va->pfm mappings
 - > Pagetables
 - > All mapped kernel pages
 - > Optional – one or more processes
- Reboot

After Reboot

- Extract dump from raw partition
- Post-process and store as a core file
- Analyze using mdb or other tools

Xen Panic Handling

- Capture state of panicking CPU
- Stop all other CPUs
- Print panic message and register state
- Jump into dom0 – still in protected mode
 - > Jump target registered at dom0 boot using a platform op
- Overlaps with kexec functionality
- Patch size: ~900 lines

High-level Dump Strategy

- Dump full dom0 image
- Add Xen symbol information to the dump
- Identify the physical pages mapped by Xen
- Add them to the crash dump
- Reboot

Memory-Related Complications

- Dom0 pagetables are read-only
 - > Disable cr0.WP during pagetable updates
- Finding Xen's pages
 - > Don't want Solaris to grok Xen's VM structures
 - > Extract mappings from pagetables
 - > Ugly – but stable
- Xen's pages have no dom0 PFNs
 - > Can't use MFNs – might collide with Solaris PFNs
 - > Invent PFNs for Xen pages
 - > PFN is just an index into core data

Other Complications

- Interrupts/exceptions still go to Xen
 - > Install new IDT with simple dom0 panic handlers
- Device usage
 - > Reclaim/re-initialize the console
 - > Use polled I/O – no interrupts
- Time Management
 - > Reprogram APIC to get timer interrupts
 - > Use TSC to track time
 - > All time is relative, so 'time of day' doesn't matter
 - > Trust Xen's `cpu_freq` info to convert ticks to time

Xen Symbol Information

- Solaris build post-processes binaries
 - > Converts some DWARF debug information into 'Compact Type Format'
 - > Added to new .SUNW_ctf ELF section
 - > Removes STABS section(s)
- Apply same post-processing to xen-syms
 - > 32-bit PAE xen-syms
 - > As built: 2916356 bytes
 - > No sym info: 559088 bytes
 - > CTF: 813784 bytes

Accessing Symbols

- Currently installing both xen.gz and xen-syms
 - > Risk of a mis-match, so we might merge them
- xen-syms loaded at dom0 boot
 - > Text/data is already loaded by GRUB
 - > Just load symbol, string, and CTF information
 - > Stored in standard Solaris 'struct module'
 - > Maps VAs not accessible to OS, so not linked unless/until Xen panics
- In dump, Xen looks like any other kernel module

Example - panic!

```
panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
```

```
gs:          161 fs:          0 es:          e010 ds:          e010
edi: f4b63000 esi: ff192f62 ebp: ffbf3f38 esp: ffbf3f1c
ebx: ffbf3f44 edx:          3 ecx:          1efc eax:          3
trp:          0 err:          0 eip: ff13b83f cs:          e008
efl:          282 usp: ffbf3fe0 ss:          e010
cr0: 8005003b cr2: c5a08fd4 cr3:  be1d00 cr4:          6f0
```

```
Xen panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
```

```
ffbf3f38 0xff11b058 (in Xen)
ffbf3f48 0xff1043f0 (in Xen)
ffbf3f68 0xff10435c (in Xen)
ffbf3f88 0xff10438b (in Xen)
ffbf3fa8 0xff17daf1 (in Xen)
```

```
dumping to /dev/dsk/c0t0d0s1, offset 860356608, content: kernel
```

```
100% done: 52473 pages dumped, compression ratio 3.51, dump succeeded
```

Example – starting postmortem

```
fsh-trout# ls -l
total 1316400
-rw-r--r--    1 root      root      1196459 Apr 10 11:29 unix.0
-rw-r--r--    1 root      root      672440320 Apr 10 11:59 vmcore.0
fsh-trout# mdb *0
Loading modules: [ unix genunix specfs dtrace xpv_psm scsi_vhci ufs ip
hook neti sctp arp usba fctl nca lofs zfs random sPPP ptm nfs md logindmux
xpv ]
> $<utsname
{
    sysname = [ "SunOS" ]
    nodename = [ "goldcloth" ]
    release = [ "5.11" ]
    version = [ "matrix-build-2007-04-02" ]
    machine = [ "i86pc" ]
}
```

Example – continuing postmortem

```
> xenver::print
{
  xv_ver = 0xccad8fc0 "3.0.4-1-sun"
  xv_chgset = 0xccad1f80 "Fri Jan 05 16:11:49 2007 +0000 13187:e2466414acc3"
  xv_compiler = 0xccacffc0 "gcc version 3.4.3 (csl-sol210-3_4-20050802)"
  xv_compile_date = 0xccad6f08 "Mon Apr  2 06:59:37 PDT 2007"
  xv_compile_by = 0xccad8fa0 "xen-discuss"
  xv_compile_domain = 0xccad8f80 "opensolaris.org"
  xv_caps = 0xccad8f60 "xen-3.0-x86_32p"
}
> ::xpvbuf
Xen panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
      gs:      161  fs:      0  es:      e010  ds:      e010
      edi: f4b63000 esi: ff192f62 ebp: ffbf3f38 esp: ffbf3f1c
      ebx: ffbf3f44 edx:      3  ecx:      1efc  eax:      3
      trp:      0  err:      0  eip: ff13b83f  cs:      e008
      efl:      282  usp: ffbf3fe0  ss:      e010
      cr0: 8005003b cr2: c5a08fd4 cr3:  be1d00  cr4:      6f0
ffbf3f38 0xff11b058 (in Xen)
ffbf3f48 0xff1043f0 (in Xen)
ffbf3f68 0xff10435c (in Xen)
[...]
```

Example - stack

```

> $C
ffbf3f38 xpv`panic+0x33()
ffbf3f48 xpv`dom0_shutdown+0x43()
ffbf3f68 xpv`domain_shutdown+0x22()
ffbf3f88 xpv`__domain_crash+0x9d()
ffbf3fa8 xpv`__domain_crash_synchronous+0x25()
ffbf3fc8 0xff17daf1() ← Exception frame. Needs work.
c5a0904c page_ctr_sub+0x47(0, 0, f3328700, 1)
c5a090e4 page_get_mnode_freelist+0x3c1(0, 10, 0, 0, 20009)
c5a09164 page_get_freelist+0x16f(f502ed14, e5406000, 0, c5a09224, e5406000,
[...])
c5a0a154 lofi_map_file+0xfc(2400000, c5a0a3e8, 1, c5a0a800, cd326e78, 80100403)
c5a0a360 lofi_ioctl+0x13e(2400000, 4c4601, c5a0a3e8, 80100403, cd326e78,
c5a0a38c cdev_ioctl+0x2e(2400000, 4c4601, c5a0a3e8, 80100403, cd326e78, c5a0a800
c5a0a3b4 ldi_ioctl+0xa4(dfb15310, 4c4601, c5a0a3e8, 80100403, cd326e78, c5a0a800
c5a0a804 xdb_setup_node+0xbc(e52e1000, c5a0a84c)
c5a0ac58 xdb_open_device+0xc9(e52e1000)
c5a0ac88 xdb_start_connect+0x59(e52e1000)
c5a0acbc xdb_oe_state_change+0x70(c66c9088, c57e5c40, 0, c5a0ad70)
c5a0acf8 ndi_event_run_callbacks+0x87(c54f0d80, c66c9088, c57e5c40, c5a0ad
[...])

```

Example – follow the clues

```

> ffbf3fc8,4/X
0xffbf3fc8:    c5a0901c    0          e0002

```

Exception frame pointer →

%eip → f4c76e9d


- Extract the faulting instruction pointer from the exception frame
 - > Yes. This should be cleaner.

Example – follow the clues

```

> ffbf3fc8,4/X
0xffbf3fc8:      c5a0901c      0      e0002      f4c76e9d
> f4c76e9d::dis
page_ctr_sub_internal:      pushl   %ebp
page_ctr_sub_internal+1:    movl   %esp,%ebp
page_ctr_sub_internal+3:    subl   $0x30,%esp
page_ctr_sub_internal+6:    andl   $0xffffffff0,%esp
page_ctr_sub_internal+9:    pushl   %ebx
page_ctr_sub_internal+0xa:  pushl   %esi
page_ctr_sub_internal+0xb:  pushl   %edi

```





- Look at the instruction
 - > A simple push. Not many ways for this to go wrong.

Example – follow the clues

```

> ffbf3fc8,4/X
0xffbf3fc8:      c5a0901c      0      e0002      f4c76e9d
> f4c76e9d/ai1
page_ctr_sub_internal+9:
page_ctr_sub_internal+9:      pushl   %ebx
> ::xpvbuf !egrep "(cr2|vcpu)"
    Xen panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
        cr0: 8005003b cr2: c5a08fd4 cr3:  be1d00 cr4:      6f0

```

- Revisit the panic message for some basic information

Example – follow the clues

```

> ffbf3fc8,4/X
0xffbf3fc8:      c5a0901c      0              e0002          f4c76e9d
> f4c76e9d/ai1
page_ctr_sub_internal+9:
page_ctr_sub_internal+9:      pushl   %ebx
> ::xpvbuf !egrep "(cr2|vcpu)"
    Xen panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
        cr0: 8005003b cr2: c5a08fd4 cr3:  beld00 cr4:      6f0
> 0xffbe1100::print struct vcpu vcpu_id
vcpu_id = 0

```

- Identify the faulting vcpu

Example – follow the clues

```

> ffbf3fc8,4/X
0xffbf3fc8:      c5a0901c      0      e0002      f4c76e9d
> f4c76e9d/ai1
page_ctr_sub_internal+9:
page_ctr_sub_internal+9:      pushl   %ebx
> ::xpvbuf !egrep "(cr2|vcpu)"
      Xen panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
      cr0: 8005003b cr2: c5a08fd4 cr3:      beld00 cr4:      6f0
> 0xffbe1100::print struct vcpu vcpu_id
vcpu_id = 0
> ::cpuinfo
ID ADDR      FLG NRUN BSPL PRI RNRN KRNRN SWITCH THREAD  PROC
0 f50454b8  1b    0    0  60   no    no t-0    c5a0ade0 sched
1 c5423080  1b    1    0  59   yes   no t-0    c6178ee0 python2.4
2 c5a71a80  1b    0    0  59   no    no t-0    c617cac0 xenstored
3 c5a70a00  1b    1    0  -1   no    no t-0    c5afdde0 (idle)

```

- Map the Xen vcpu to a Solaris CPU

Example – follow the clues

```

> ffbf3fc8,4/X
0xffbf3fc8:      c5a0901c      0      e0002      f4c76e9d
> f4c76e9d/ai1
page_ctr_sub_internal+9:
page_ctr_sub_internal+9:      pushl   %ebx
> ::xpvbuf !egrep "(cr2|vcpu)"
      Xen panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
      cr0: 8005003b cr2: c5a08fd4 cr3:      beld00 cr4:      6f0
> 0xffbe1100::print struct vcpu vcpu_id
vcpu_id = 0
> ::cpuinfo
ID ADDR      FLG NRUN BSPL PRI RNRN KRNRN SWITCH THREAD  PROC
 0 f50454b8  1b    0   0  60   no   no t-0   c5a0ade0 sched
 1 c5423080  1b    1   0  59  yes  no t-0   c6178ee0 python2.4
 2 c5a71a80  1b    0   0  59   no   no t-0   c617cac0 xenstored
 3 c5a70a00  1b    1   0  -1   no   no t-0   c5afdde0 (idle)
> c5a0ade0::print kthread_t t_stkbase
t_stkbase = 0xc5a09000

```

- Examine the thread on that CPU

Example – follow the clues

```

> ffbf3fc8,4/X
0xffbf3fc8:      c5a0901c      0      e0002      f4c76e9d
> f4c76e9d/ai1
page_ctr_sub_internal+9:
page_ctr_sub_internal+9:      pushl   %ebx
> ::xpvbuf !egrep "(cr2|vcpu)"
      Xen panic[cpu2]/vcpu=0xffbe1100: Domain 0 crashed:
      cr0: 8005003b cr2: c5a08fd4 cr3:      belld00 cr4:      6f0
> 0xffbe1100::print struct vcpu vcpu_id
vcpu_id = 0
> ::cpuinfo
ID ADDR      FLG NRUN BSPL PRI RNRN KRNRN SWITCH THREAD  PROC
0 f50454b8  1b    0    0  60   no    no t-0    c5a0ade0 sched
1 c5423080  1b    1    0  59   yes   no t-0    c6178ee0 python2.4
2 c5a71a80  1b    0    0  59   no    no t-0    c617cac0 xenstored
3 c5a70a00  1b    1    0  -1   no    no t-0    c5afdde0 (idle)
> c5a0ade0::print kthread_t t_stkbase
t_stkbase = 0xc5a09000

```

- > Faulted just below the stack base
- > Conclusion: we've blown our stack

Example – Assigning Blame

```

> $C
[...]
c5a0a154 lofi_map_file+0xfc(2400000, c5a0a3e8, 1, c5a0a800, cd326e78, 80100403)
c5a0a360 lofi_ioctl+0x13e(2400000, 4c4601, c5a0a3e8, 80100403, cd326e78,
c5a0a38c cdev_ioctl+0x2e(2400000, 4c4601, c5a0a3e8, 80100403, cd326e78, c5a0a800
c5a0a3b4 ldi_ioctl+0xa4(dfb15310, 4c4601, c5a0a3e8, 80100403, cd326e78, c5a0a800
c5a0a804 xdb_setup_node+0xbc(e52e1000, c5a0a84c)
c5a0ac58 xdb_open_device+0xc9(e52e1000)
c5a0ac88 xdb_start_connect+0x59(e52e1000)
c5a0acbc xdb_oe_state_change+0x70(c66c9088, c57e5c40, 0, c5a0ad70)
c5a0acf8 ndi_event_run_callbacks+0x87(c54f0d80, c66c9088, c57e5c40, c5a0ad
[...]

```

- > Problem: Big gaps -> large allocations on the stack
- > Fix: dynamically allocate the MAXPATHLEN arrays in these two routines

Examples - misc.

```
> ::domain
```

ADDR	ID	TPAGES	MPAGES	FLAGS	HVM	VCPU	EVTCHN
ffbe4100	0	76cfd	ffffffff	0	0	ffbe43b0	ffbe4148
ff26a100	3	44afd	44c00	0	0	ff26a3b0	ff26a148

```
> dom0_nrpages/X
```

```
xpv`dom0_nrpages:
```

```
xpv`dom0_nrpages:          76c00
```

```
> ff1e0440::whatis
```

```
ff1e0440 is xpv`domain_list+0 in xpv's data segment
```

```
> ffbe4100::walk vcpu | ::print struct vcpu runstate.state
```

```
runstate.state = 0
```

```
runstate.state = 0
```

```
runstate.state = 0
```

```
runstate.state = 0x2
```

```
> ffbe1100::print struct vcpu arch.guest_context.user_regs
```

```
{
```

```
    arch.guest_context.user_regs.ebx = 0xdeadbeef
```

```
    arch.guest_context.user_regs.ecx = 0xdeadbeef
```

```
    arch.guest_context.user_regs.edx = 0xcd5aee08
```

```
    arch.guest_context.user_regs.esi = 0xf50303d0
```

```
    arch.guest_context.user_regs.edi = 0x1
```

Xen Crash Dumps on Solaris

Nils Nieuwejaar

nils.nieuwejaar@sun.com

<http://blogs.sun.com/nilsn>