redhat.

Walking the Linux Kernel

Stanislav Kozina
Associate Manager

April 2016

AGENDA

Linux Kernel in general

Debugging kernel issues without crash

Prepare the system for crashing

Crash it with systemtap

See what we can get from the crash dump

Linux Kernel in general – boring

- Just software…
- Written in C & asm
- List of expected features
  - Boot and initialization process
  - Memory and process management
  - Hardware abstraction
    - Files, directories, sockets, …
  - Resources abstraction
    - CPU, memory, …
  - POSIX

Linux Kernel in general – BUT!

- Quite big (20mil LOC)
- No libc (many other standards functions instead)
- Special environment
  - Preemptive, shared memory space
- Early boot code is tricky
  - No dynamic allocator, AP, scheduler, even locks!
- Special security requirements
  - Kernel should not just die and/or leak anything

Linux Kernel in general – development system

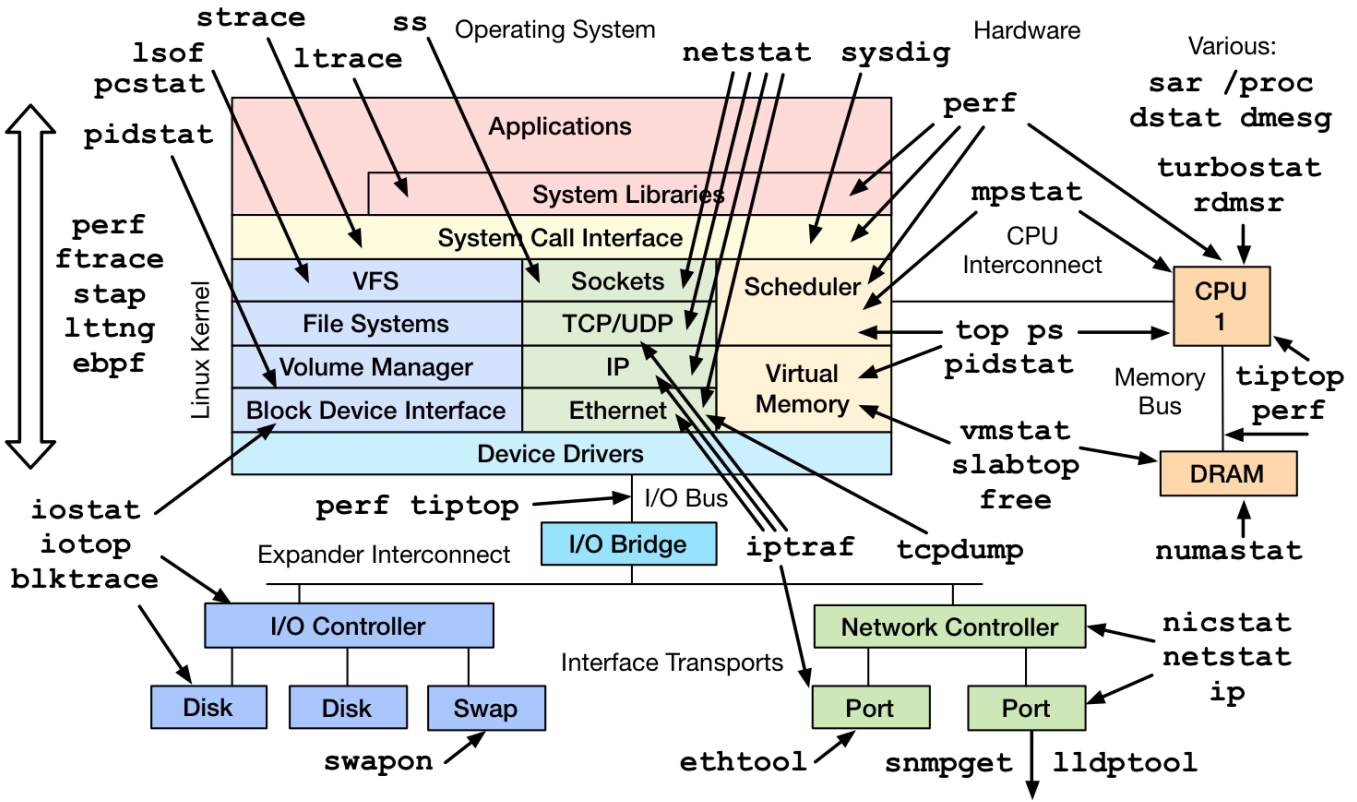
- Open source
- List of maintainers in MAINTAINERS
- Patches posted via email
  - Documentation/SubmittingPatches
- LKML
- Usually companies care about support of their stuff
  - Hardware vendors…
- “We don't break userland”


DIGGING IN

Observing kernel is not trivial

- Hard to get a consistent picture
  - If we stop it, how we observe it?
  - Using Vms?
  - printk()
  - Statistics, /proc, perf, strace, ftrace, ...

Linux Performance Observability Tools



http://www.brendangregg.com/linuxperf.html 2015

Debugging kernel issues

- Oops messages
  - Message buffer, registers, stack/backtrace
- Oops leaves the system running, but unstable!
  - Current task is killed
- Printk()
- Systemtap, ftrace
- crash
- Sysrq triggers

Kernel oops

```
[   32.580355] SysRq : Trigger a crash
[   32.581331] BUG: unable to handle kernel NULL pointer dereference at
[   32.582703] IP: [<ffffffff813b9716>] sysrq_handle_crash+0x16/0x20
[   32.583781] PGD 3b503067 PUD 3b502067 PMD 0
[   32.584609] Oops: 0002 [#1] SMP
[   32.585210] Modules linked in: ip6t_rpfilter (... )
[   32.598062] CPU: 1 PID: 2370 Comm: bash Not tainted 3.10.0-
327.el7.x86_64 #1
[   32.598923] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996),
BIOS 1.8.2-20150714_191134- 04/01/2014
[   32.600020] task: ffff88003b544500 ti: ffff88003b518000 task.ti:
ffff88003b518000
```

Kernel oops 2

```
[   32.600875] RIP: 0010:[<ffffffff813b9716>]  [<ffffffff813b9716>]
sysrq_handle_crash+0x16/0x20
[   32.601864] RSP: 0018:ffff88003b51be80  EFLAGS: 00010046
[   32.602469] RAX: 000000000000000f RBX: ffffffff81a06220 RCX:
0000000000000000
[   32.603281] RDX: 0000000000000000 RSI: ffff88003fd0d6c8 RDI:
0000000000000063
[   32.604092] RBP: ffff88003b51be80 R08: 0000000000000092 R09:
0000000000000268
[   32.606518] FS:  00007fa978f3b740(0000) GS:ffff88003fd00000(0000)
knlGS:0000000000000000
[   32.607431] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
```

Kernel oops 3

```
[   32.613426] Call Trace:
[   32.613724]  [<ffffffff813b9ed2>] __handle_sysrq+0xa2/0x170
[   32.614359]  [<ffffffff813ba3af>] write_sysrq_trigger+0x2f/0x40
[   32.615049]  [<ffffffff812492ad>] proc_reg_write+0x3d/0x80
[   32.615685]  [<ffffffff811de5cd>] vfs_write+0xbd/0x1e0
[   32.616286]  [<ffffffff816411b3>] ? trace_do_page_fault+0x43/0x110
[   32.616999]  [<ffffffff811df06f>] SyS_write+0x7f/0xe0
[   32.617581]  [<ffffffff81645909>] system_call_fastpath+0x16/0x1b
```

Kernel oops

- Good enough when
  - The problem is in a direct callpath
- Not sufficient when
  - We need to see more context
  - Other tasks
  - Structure content
- Be aware of -fomit-frame-pointer
  - Reliable address only if %rsp == %rbp + sizeof (long)
  - CONFIG_FRAME_POINTER

Getting crash dump

- Need to configure kdump
  - Failsafe crash kernel loaded in a memory
  - Switch to the new kernel via kexec
  - Run kdump to save to content of /proc/vmcore

```
# echo c > /proc/sysrq-trigger
```

Opening a crash dump

- Dump of all kernel memory
  - Zero pages, caches, buffers and userland are not dumped by default
- Can be opened in gdb
  - But we need DWARF debugging symbols
  - Kernel-debuginfo packages in Fedora/CentOS/RHEL
- Crash(8) tool
  - Can run on live system through /proc/kcore as well


DEMO

Using crash(8)

- mod -S
- set hex
- sys
- log
- mount
- swap
- mach [-m | -c]
- net

Linux tasks

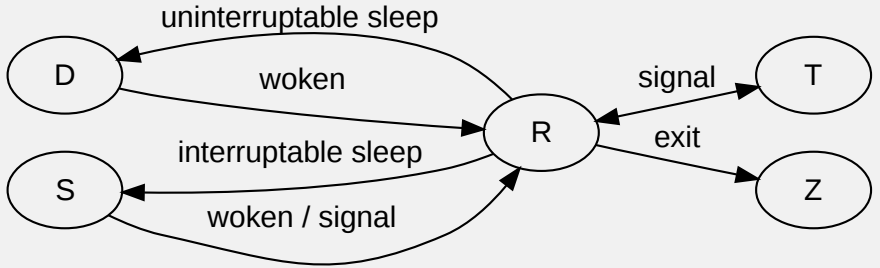
- include/linux/sched.h: struct task_struct {}
- clone(2)
- CLONE_THREAD:
  - “Since Linux 2.4, calls to getpid(2) return the TGID of the caller.”

task_struct

```
struct task_struct {
        long state;
        void *stack;
        struct sched_entity se;
        struct mm_struct *mm;
        int exit_code;
        pid_t pid, tgid;
        struct task_struct *parent;
        const struct cred __rcu *cred;
        char comm[TASK_COMM_LEN];
        struct files_struct *files;
}
```

Process states in Linux

- ps
  - Walk task_struct in the kernel
  - -u | -k | -G -- filter only user|kernel|thread group leader tasks
  - ps -p | -c -- walk parent/child relationship
  - ps -S -- summary of task states



https://idea.popcount.org/2012-12-11-linux-process-states/

Process tree

- ps -p

```
crash> ps -p 6864
PID: 0      TASK: ffffffff81c124c0  CPU: 0   COMMAND: "swapper/0"
 PID: 1      TASK: ffff88007c888000  CPU: 0   COMMAND: "systemd"
  PID: 784    TASK: ffff8800369fbc00  CPU: 0   COMMAND: "login"
   PID: 2884   TASK: ffff880036a19e00  CPU: 0   COMMAND: "bash"
    PID: 6864   TASK: ffff880036850000  CPU: 0   COMMAND: "cat"
```


DEMO

Backtrace

- bt -s

```
 #8 [ffff88007abefd88] ping_v4_seq_show+0x5 at ffffffff816f6a15
 #9 [ffff88007abefd98] seq_read+0xec at ffffffff81246b7c
#10 [ffff88007abefe28] proc_reg_read+0x42 at ffffffff81290132
#11 [ffff88007abefe48] __vfs_read+0x37 at ffffffff81223387
#12 [ffff88007abefed0] vfs_read+0x83 at ffffffff81223e43
#13 [ffff88007abeff08] sys_read+0x55 at ffffffff81224b95
#14 [ffff88007abeff50] entry_SYSCALL_64_fastpath+0x12 at
ffffffff81781e2e
```

Full stack dump

- bt -fs

```
    ffff88007abefe10: 0000000000020000 0000000000020000
    ffff88007abefe20: ffff88007abefe40 ffffffff81290132
#10 [ffff88007abefe28] proc_reg_read+0x42 at ffffffff81290132
    ffff88007abefe30: ffff88007c015b00 ffff88007abeff18
    ffff88007abefe40: ffff88007abefec8 ffffffff81223387
#11 [ffff88007abefe48] __vfs_read+0x37 at ffffffff81223387
    ffff88007abefe50: 0000000000020000 ffff88007c015b00
    ffff88007abefe60: ffff88007c015b10 ffff880036575148
```

Full stack dump

- bt -fs

```
    ffff88007abefe10: 0000000000020000 0000000000020000
    ffff88007abefe20: ffff88007abefe40 ffffffff81290132
#10 [ffff88007abefe28] proc_reg_read+0x42 at ffffffff81290132
    ffff88007abefe30: ffff88007c015b00 ffff88007abeff18
    ffff88007abefe40: ffff88007abefec8 ffffffff81223387
#11 [ffff88007abefe48] __vfs_read+0x37 at ffffffff81223387
    ffff88007abefe50: 0000000000020000 ffff88007c015b00
    ffff88007abefe60: ffff88007c015b10 ffff880036575148
```

Disassemble

```
crash> dis proc_reg_read
0xffffffff812900f0 <proc_reg_read>:     nopl   0x0(%rax,%rax,1) [FTRACE
NOP]
0xffffffff812900f5 <proc_reg_read+0x5>: push   %rbp
0xffffffff812900f6 <proc_reg_read+0x6>: xor    %r8d,%r8d
0xffffffff812900f9 <proc_reg_read+0x9>: mov    %rsp,%rbp
0xffffffff812900fc <proc_reg_read+0xc>: push   %r12
0xffffffff812900fe <proc_reg_read+0xe>: push   %rbx
```


DEMO

mm_struct

- task_struct -> mm

```
struct mm_struct {
        struct vm_area_struct *mmap;            /* list of VMAs */
        struct rb_root mm_rb;
        pgd_t * pgd;
        unsigned long start_code, end_code, start_data, end_data;
        unsigned long start_brk, brk, start_stack;
}
```

vm_area_struct

- mm_struct -> mmap

```
struct vm_area_struct {
        unsigned long vm_start;
        unsigned long vm_end;
        struct file * vm_file;
        struct anon_vma * anon_vma_chain;
}
```

Walk the memory map of a process

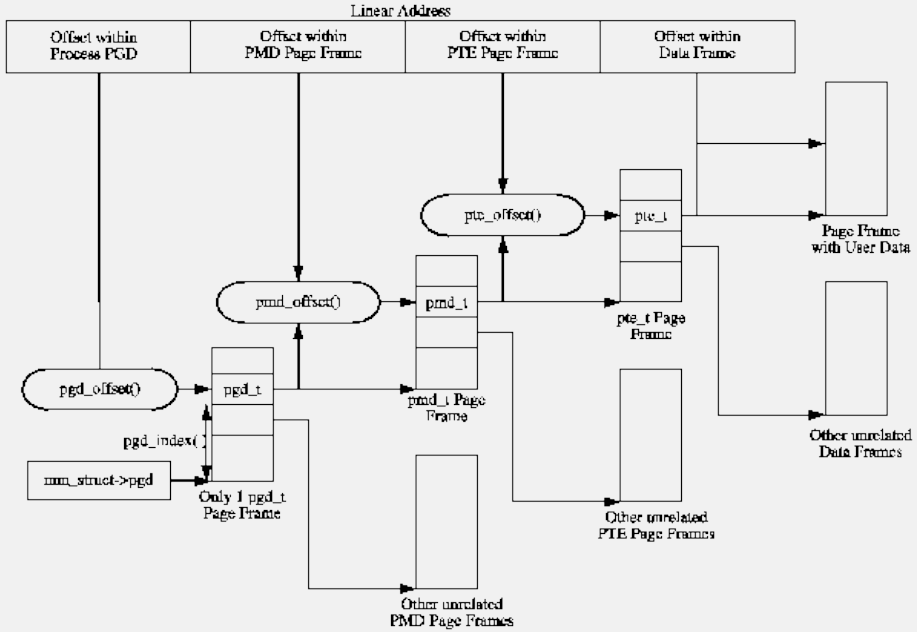
- pmap(1)
- task_struct→mm→mmap→vm_start

```
$ pmap $(pgrep bash)

crash> list -o vm_area_struct.vm_next
```


DEMO

Page tables



https://www.kernel.org/doc/gorman/html/understand/understand006.html

Page tables (2)

- mm_struct->pgd

```
crash> rd -64 0xffff88007bed9000 1024
ffff88007bed9000:  0000000000000000 0000000000000000   ................
ffff88007bed9010:  0000000000000000 0000000000000000   ................
ffff88007bed9020:  0000000000000000 0000000000000000   ................
ffff88007bed9030:  0000000000000000 0000000000000000   ................
```


DEMO

kmem

- [-i] -- overview
- [-p] -- Page info, walks mem_mam
- [-s] -- Slab statistic

```
crash> kmem -s ext4_inode_cache
CACHE            NAME                 OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE
ffff880079fa1000 ext4_inode_cache        1016      48728     48728   6091     8k
crash> struct ext4_inode_info | tail -n 1
SIZE: 0x3f8
```

Slab allocator

```
crash> kmem -s ext4_inode_cache
CACHE            NAME                 OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE
ffff880079fa1000 ext4_inode_cache        1016      48728     48728   6091     8k
```

```
ext4_inode_cachep = kmem_cache_create("ext4_inode_cache",
                                      sizeof(struct ext4_inode_info),
                                      0, (SLAB_RECLAIM_ACCOUNT|
                                        SLAB_MEM_SPREAD),
                                      init_once);
```


DEMO

CPUs

- There's no “cpu_t” structure
- Instead:
  - DECLARE_PER_CPU_SHARED_ALIGNED(struct rq, runqueues);
- O(1) scheduler replaced with CFS
  - Uses RB tree

runq

- runq

```
crash> runq
CPU 0 RUNQUEUE: ffff88007fc16c80
  CURRENT: PID: 6864   TASK: ffff880036850000  COMMAND: "cat"
  RT PRIO_ARRAY: ffff88007fc16e30
     [no tasks queued]
  CFS RB_ROOT: ffff88007fc16d20
     [120] PID: 6464   TASK: ffff880036868000  COMMAND: "kworker/0:1"
```


DEMO

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHatNews


youtube.com/user/RedHatVideos