ktap introduction

Edited by:

ktap project
www.ktap.org

September 16, 2013

Contents

- Introduction
- Architecture
- Building and Running
- Examples
- Performance
- More information

ktap

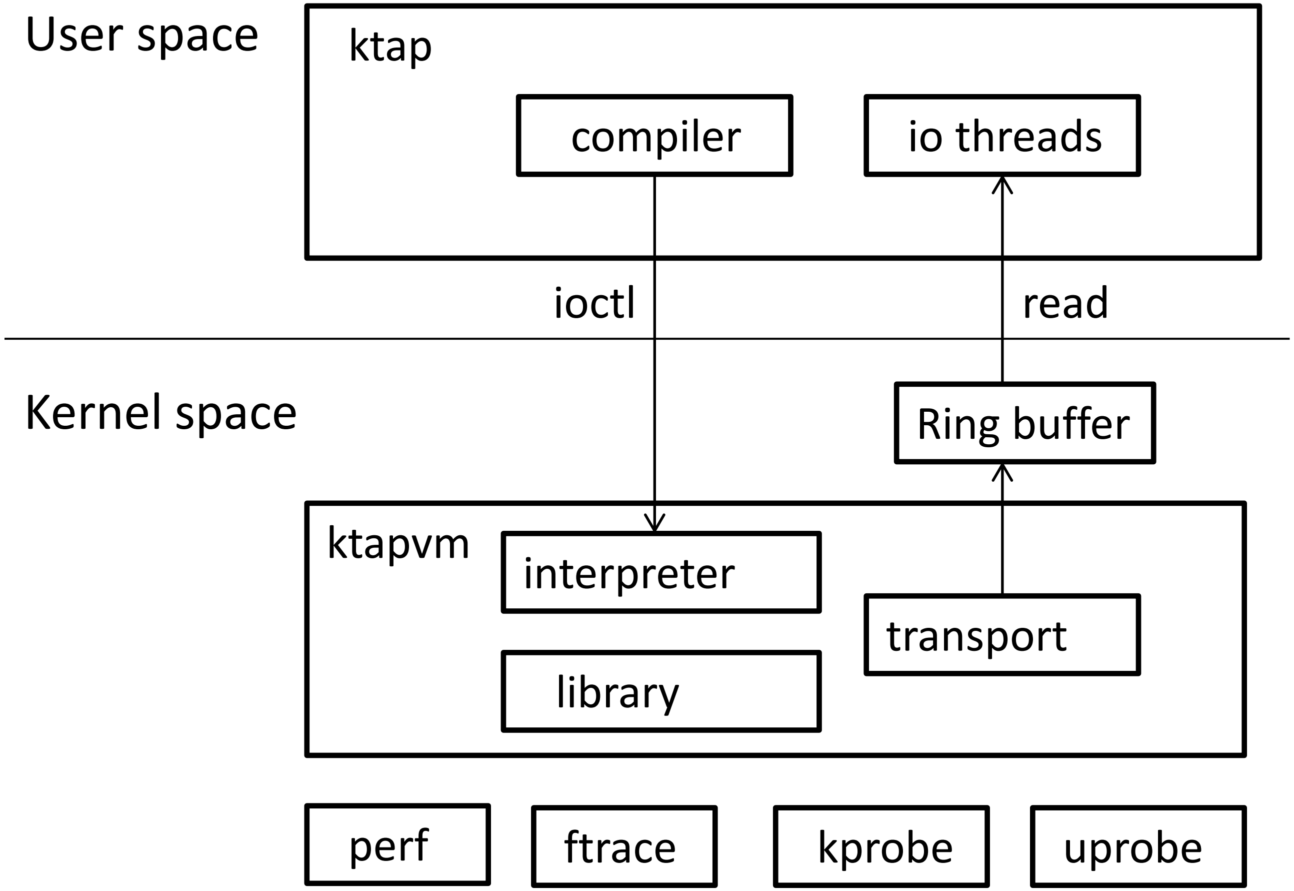
ktap is a new scripting dynamic tracing tool for Linux,
it is designed to give operational insights that allow users
to tune, troubleshoot and extend kernel and application.

ktap is similar with Systemtap and Dtrace, but it’s not a clone of them.

Feature highlights

- simple but powerful scripting language
- register based interpreter (heavily optimized) in kernel
- small and lightweight (6KLOC of interpreter)
- not depend on gcc for each script running
- easy to use in embedded system without debug info
- support for tracepoint, kprobe, uprobe, ftrace, timer, etc
- supported in x86, arm, ppc, mips
- good extendibility
- safety in sandbox

Architecture



Building and Running

1) Clone ktap from github

```
# git clone http://github.com/ktap/ktap.git
```

2) Compiling ktap

```
# cd ktap
# make       #generate ktapvm kernel module and ktap binary
```

3) Load ktapvm kernel module(make sure debugfs mounted)

```
# make load  #need to be root or have sudo access
```

4) Running ktap

```
# ./ktap scripts/helloworld.kp
```

Examples

1) simplest one-liner command to enable all tracepoints

```
ktap -e "trace *:* { print(argevent) }“
```

2) syscall tracing on target process

```
ktap -e "trace syscalls:* { print(argevent) }" – ls
```

3) function tracing

```
ktap -e "trace ftrace:function { print(argevent) }"
ktap -e "trace ftrace:function /ip == mutex*/ { print(argevent) }"
```

Examples (Cont.)

4) simple syscall tracing

```
#scripts/syscalls/syscalls.kp
trace syscalls:* {
	print(cpu(), pid(), execname(), argevent)
}
```

5) syscall tracing in histogram style

```
#scripts/syscalls/syscalls_count.kp
hist = {}

trace syscalls:sys_enter_* {
	hist[argname] += 1
}

trace_end {
	histogram(hist)
}
```

Examples (Cont.)

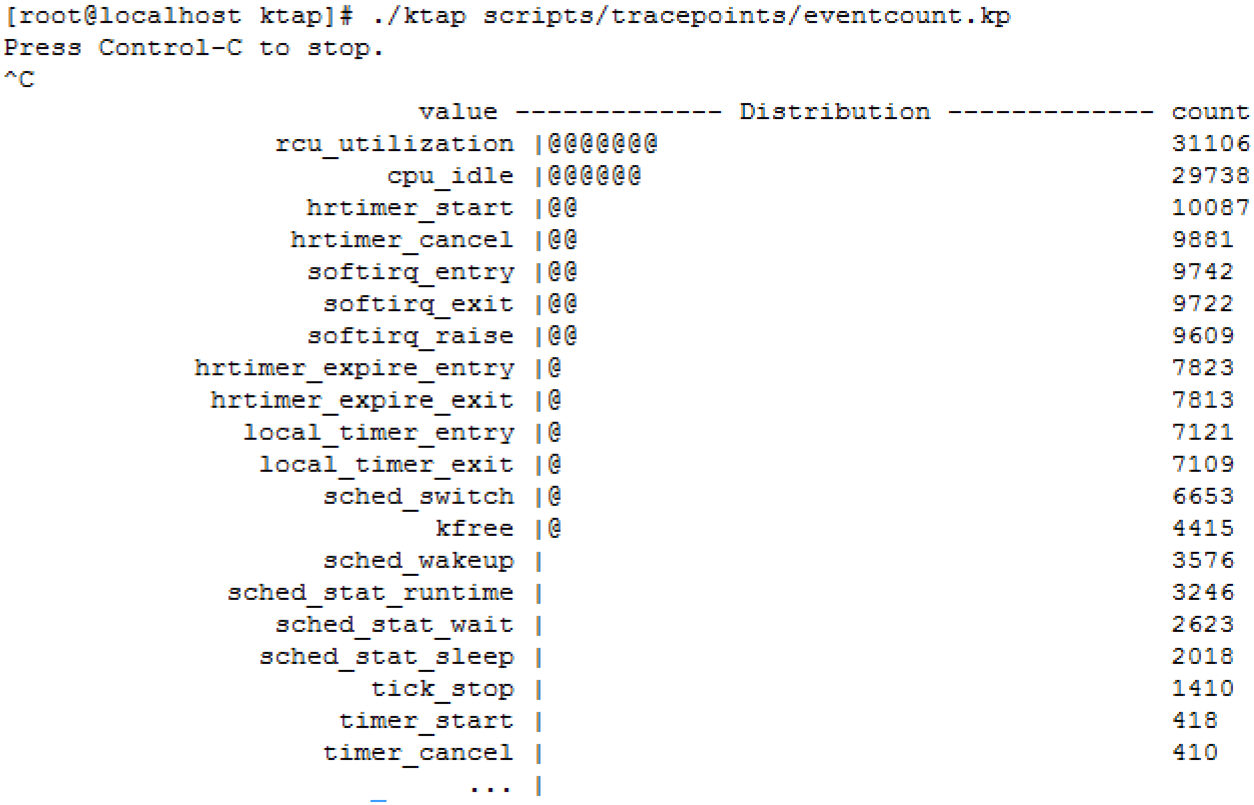
6) kprobe tracing

```
#scripts/io/kprobes-do-sys-open.kp
trace probe:do_sys_open dfd=%di fname=%dx flags=%cx mode=+4($stack) {
    print("entry:", execname(), argevent)
}

trace probe:do_sys_open%return fd=$retval {
    print("exit:", execname(), argevent)
}
```

7) timer

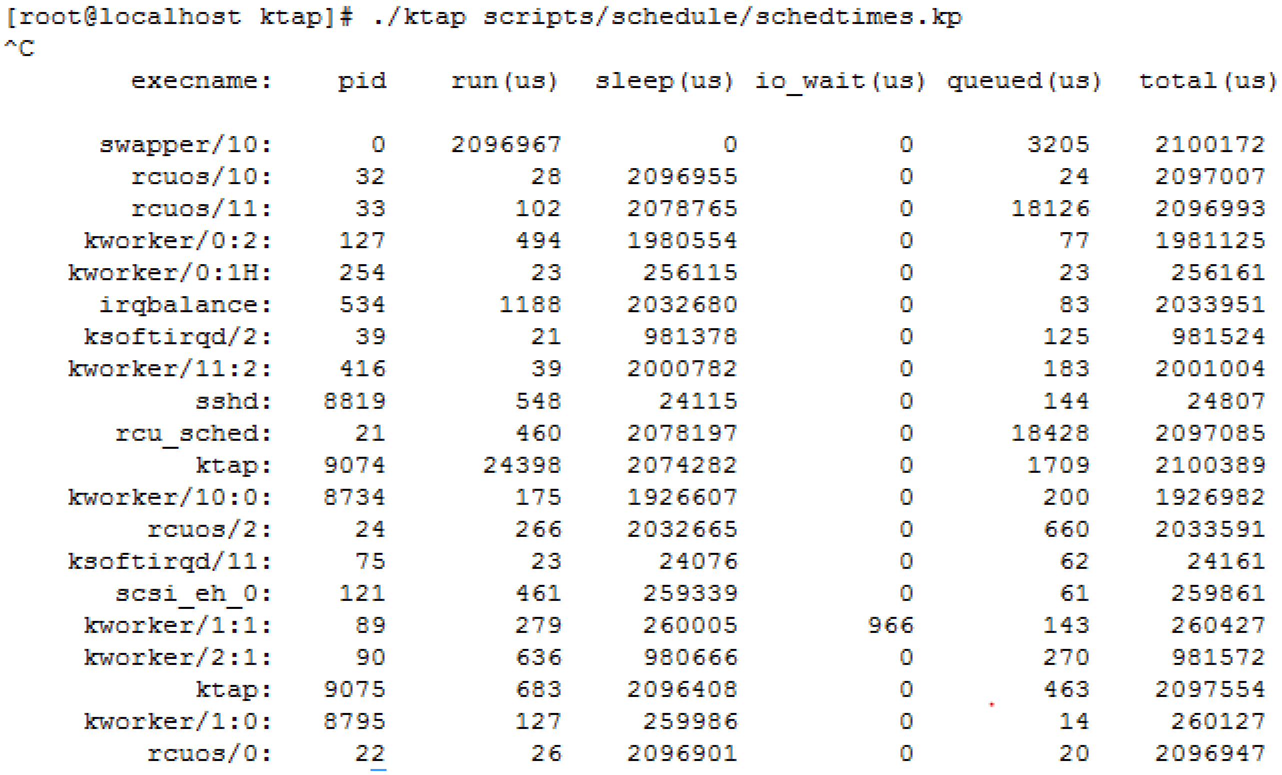
```
tick-1ms {
    printf("time fired on one cpu\n");
}

profile-2s {
    printf("time fired on every cpu\n");
}
```

Example: eventcount.kp



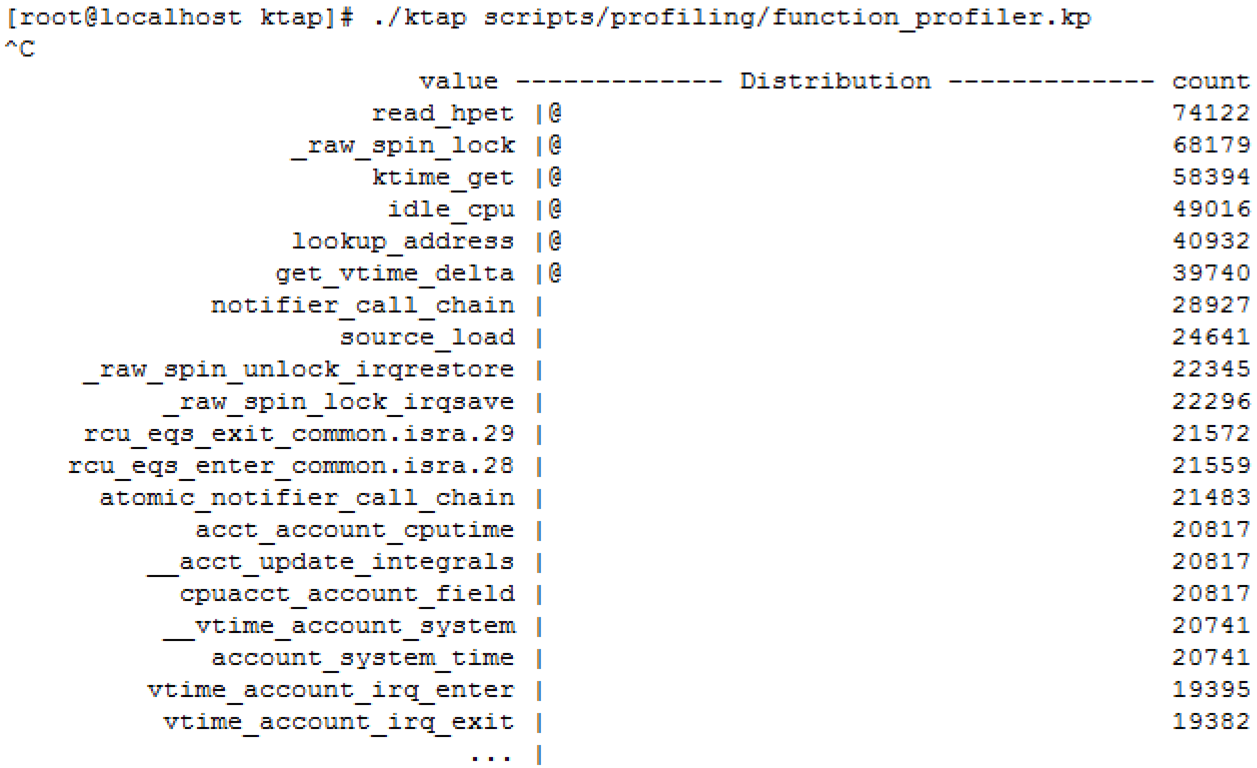
```
[root@localhost ktap]# ./ktap scripts/tracepoints/eventcount.kp
Press Control-C to stop.
^C
                          value ------------- Distribution ------------- count
                 rcu_utilization |@@@@@@@                                 31106
                        cpu_idle |@@@@@@                                  29738
                   hrtimer_start |@@                                      10087
                  hrtimer_cancel |@@                                      9881
                   softirq_entry |@@                                      9742
                    softirq_exit |@@                                      9722
                   softirq_raise |@@                                      9609
            hrtimer_expire_entry |@                                       7823
             hrtimer_expire_exit |@                                       7813
               local_timer_entry |@                                       7121
                local_timer_exit |@                                       7109
                    sched_switch |@                                       6653
                           kfree |@                                       4415
                    sched_wakeup |                                        3576
              sched_stat_runtime |                                        3246
                 sched_stat_wait |                                        2623
                sched_stat_sleep |                                        2018
                       tick_stop |                                        1410
                     timer_start |                                        418
                    timer_cancel |                                        410
                             ... |
```

Example: schedtimes.kp



```
[root@localhost ktap]# ./ktap scripts/schedule/schedtimes.kp
^C
        execname:    pid    run(us)  sleep(us) io_wait(us) queued(us)  total(us)

      swapper/10:      0    2096967          0          0       3205    2100172
        rcuos/10:     32         28    2096955          0         24    2097007
        rcuos/11:     33        102    2078765          0      18126    2096993
     kworker/0:2:    127        494    1980554          0         77    1981125
    kworker/0:1H:    254         23     256115          0         23     256161
      irqbalance:    534       1188    2032680          0         83    2033951
     ksoftirqd/2:     39         21     981378          0        125     981524
    kworker/11:2:    416         39    2000782          0        183    2001004
            sshd:   8819        548      24115          0        144      24807
       rcu_sched:     21        460    2078197          0      18428    2097085
            ktap:   9074      24398    2074282          0       1709    2100389
    kworker/10:0:   8734        175    1926607          0        200    1926982
         rcuos/2:     24        266    2032665          0        660    2033591
    ksoftirqd/11:     75         23      24076          0         62      24161
       scsi_eh_0:    121        461     259339          0         61     259861
     kworker/1:1:     89        279     260005        966        143     260427
     kworker/2:1:     90        636     980666          0        270     981572
            ktap:   9075        683    2096408          0        463    2097554
     kworker/1:0:   8795        127     259986          0         14     260127
         rcuos/0:     22         26    2096901          0         20    2096947
```

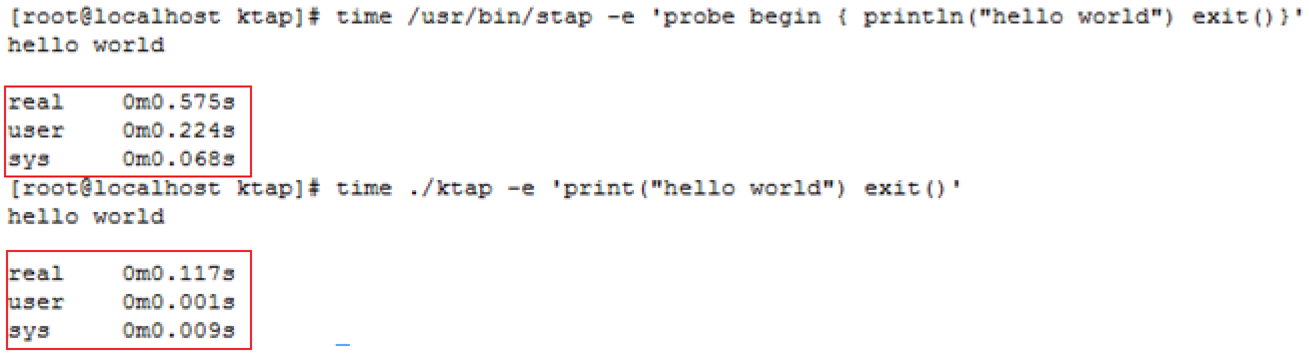
Example: function_profiler.kp



```
[root@localhost ktap]# ./ktap scripts/profiling/function_profiler.kp
^C
                          value ------------- Distribution ------------- count
                       read_hpet |@                                      74122
                  _raw_spin_lock |@                                      68179
                       ktime_get |@                                      58394
                        idle_cpu |@                                      49016
                  lookup_address |@                                      40932
                 get_vtime_delta |@                                      39740
             notifier_call_chain |                                       28927
                     source_load |                                       24641
     _raw_spin_unlock_irqrestore |                                       22345
          _raw_spin_lock_irqsave |                                       22296
     rcu_eqs_exit_common.isra.29 |                                       21572
    rcu_eqs_enter_common.isra.28 |                                       21559
      atomic_notifier_call_chain |                                       21483
            acct_account_cputime |                                       20817
         __acct_update_integrals |                                       20817
           cpuacct_account_field |                                       20817
          __vtime_account_system |                                       20741
             account_system_time |                                       20741
         vtime_account_irq_enter |                                       19395
          vtime_account_irq_exit |                                       19382
                             ... |
```

Performance: boot time

The execution time of helloworld script is nearly **5x** than Systemtap



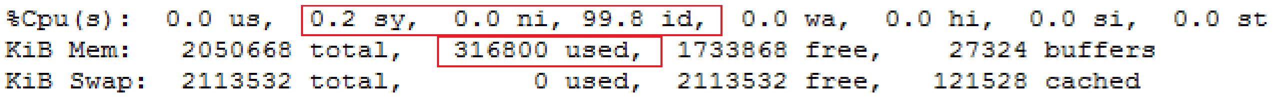
```
[root@localhost ktap]# time /usr/bin/stap -e 'probe begin { println("hello world") exit()}'
hello world

real    0m0.575s
user    0m0.224s
sys     0m0.068s
[root@localhost ktap]# time ./ktap -e 'print("hello world") exit()'
hello world

real    0m0.117s
user    0m0.001s
sys     0m0.009s
```

In Systemtap, normally you will need to wait **10+ seconds** before start to run script, for some complicated scripts, 20+ seconds maybe need to wait.

In ktap, you just need to wait **10+ms**(6ms to compile, 4ms to boot) to starting most scripts.

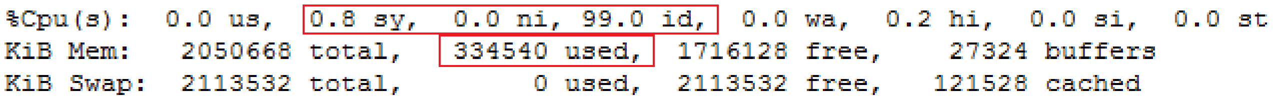
Performance: function_profiler

Idle:



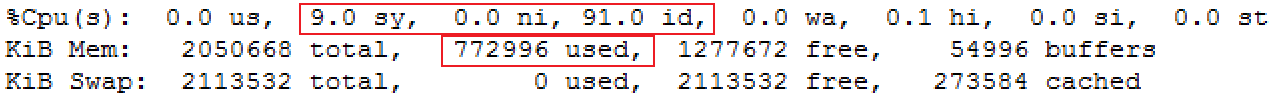
```
%Cpu(s):  0.0 us,  0.2 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2050668 total,  316800 used, 1733868 free,    27324 buffers
KiB Swap:  2113532 total,        0 used, 2113532 free,   121528 cached
```

Enable all kernel function profiling in ktap:



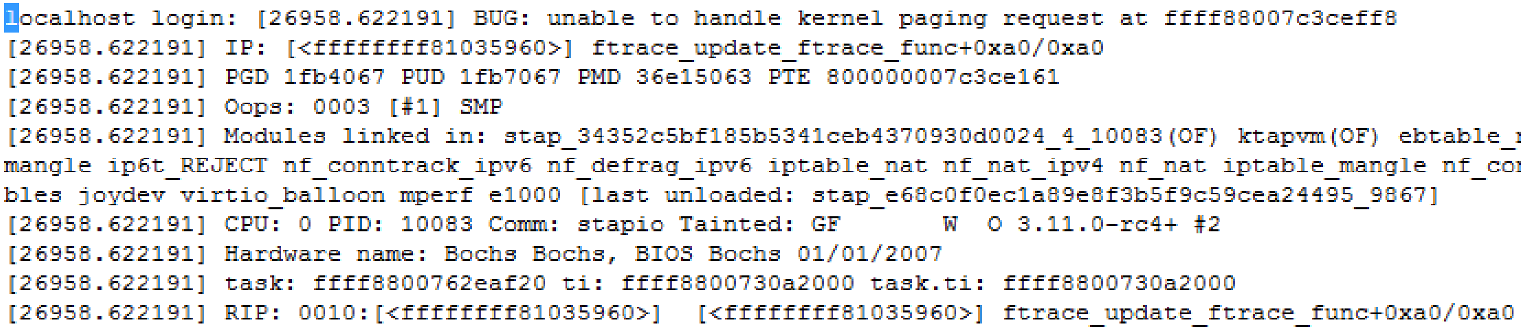
```
%Cpu(s):  0.0 us,  0.8 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.2 hi,  0.0 si,  0.0 st
KiB Mem:   2050668 total,  334540 used, 1716128 free,    27324 buffers
KiB Swap:  2113532 total,        0 used, 2113532 free,   121528 cached
```

Enable all kernel function profiling in Systemtap:



```
%Cpu(s):  0.0 us,  9.0 sy,  0.0 ni, 91.0 id,  0.0 wa,  0.1 hi,  0.0 si,  0.0 st
KiB Mem:   2050668 total,  772996 used, 1277672 free,    54996 buffers
KiB Swap:  2113532 total,        0 used, 2113532 free,   273584 cached
```

kernel crashed after 5 seconds:

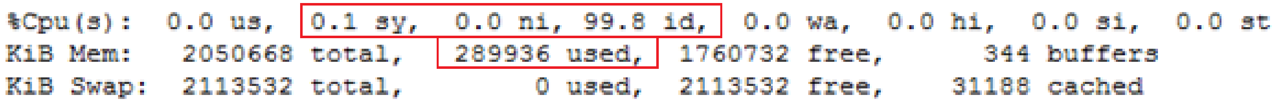


```
localhost login: [26958.622191] BUG: unable to handle kernel paging request at ffff88007c3ceff8
[26958.622191] IP: [<ffffffff81035960>] ftrace_update_ftrace_func+0xa0/0xa0
[26958.622191] PGD 1fb4067 PUD 1fb7067 PMD 36e15063 PTE 800000007c3ce161
[26958.622191] Oops: 0003 [#1] SMP
[26958.622191] Modules linked in: stap_34352c5bf185b5341ceb4370930d0024_4_10083(OF) ktapvm(OF) ebtable_
mangle ip6t_REJECT nf_conntrack_ipv6 nf_defrag_ipv6 iptable_nat nf_nat_ipv4 nf_nat iptable_mangle nf_co
bles joydev virtio_balloon mperf e1000 [last unloaded: stap_e68c0f0ec1a89e8f3b5f9c59cea24495_9867]
[26958.622191] CPU: 0 PID: 10083 Comm: stapio Tainted: GF       W  O 3.11.0-rc4+ #2
[26958.622191] Hardware name: Bochs Bochs, BIOS Bochs 01/01/2007
[26958.622191] task: ffff8800762eaf20 ti: ffff8800730a2000 task.ti: ffff8800730a2000
[26958.622191] RIP: 0010:[<ffffffff81035960>]  [<ffffffff81035960>] ftrace_update_ftrace_func+0xa0/0xa0
```

Performance: stack_profiler

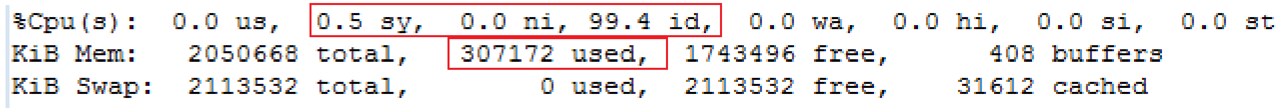
System stack sample with 1ms period

Idle:



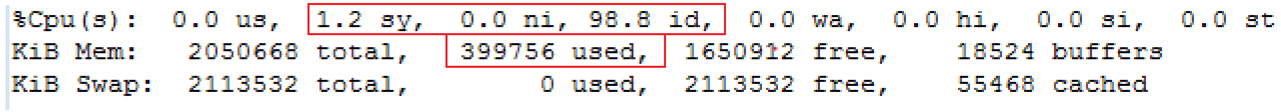
```
%Cpu(s):  0.0 us,  0.1 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2050668 total,  289936 used,  1760732 free,      344 buffers
KiB Swap:  2113532 total,        0 used,  2113532 free,    31188 cached
```

Enable stack profiling in ktap:



```
%Cpu(s):  0.0 us,  0.5 sy,  0.0 ni, 99.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2050668 total,  307172 used,  1743496 free,      408 buffers
KiB Swap:  2113532 total,        0 used,  2113532 free,    31612 cached
```

Enable stack profiling in Systemtap:



```
%Cpu(s):  0.0 us,  1.2 sy,  0.0 ni, 98.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2050668 total,  399756 used,  1650912 free,    18524 buffers
KiB Swap:  2113532 total,        0 used,  2113532 free,    55468 cached
```

More information

- Project status:
  - Released 0.2
  - only support kernel newer than 3.1
- Project home: www.ktap.org
- Code location: https://github.com/ktap/ktap.git
- Mailing list: ktap@freelists.org
- Documentation: ktap/doc/
- Sample scripts: ktap/scripts/