Linux Instrumentation

Ian Munsie

June 24, 2010

Outline

Outline

What major players are in this area? DTrace

What major players are in this area? DTrace

DTrace

- All purpose kernel and userspace tracing
- "Better than Linux"
- Originated from Solaris
- Mainstream ports for FreeBSD, NetBSD, Mac OS X
- Licencing Issues with Linux
- No performance impact when probes disabled
- Minimal probe effect

Definition

Probe Effect: The phenomena where observing a system will change the behaviour of that system

DTrace

- All purpose kernel and userspace tracing
- "Better than Linux"
- Originated from Solaris
- Mainstream ports for FreeBSD, NetBSD, Mac OS X
- Licencing Issues with Linux
- No performance impact when probes disabled
- Minimal probe effect

Definition

Probe Effect: The phenomena where observing a system will change the behaviour of that system

DTrace

- All purpose kernel and userspace tracing
- “Better than Linux”
- Originated from Solaris
- Mainstream ports for FreeBSD, NetBSD, Mac OS X
- Licencing Issues with Linux
- No performance impact when probes disabled
- Minimal probe effect

Definition

Probe Effect: The phenomena where observing a system will change the behaviour of that system

DTrace

- All purpose kernel and userspace tracing
- “Better than Linux”
- Originated from Solaris
- Mainstream ports for FreeBSD, NetBSD, Mac OS X
- Licencing Issues with Linux
- No performance impact when probes disabled
- Minimal probe effect

Definition

Probe Effect: The phenomena where observing a system will change the behaviour of that system

DTrace Implementation

- Well documented probes

Definition

Probe: A location or action which can be hooked into in order to perform some arbitrary action

- Probes placed statically in source code
- Probe location well considered
- Probes in Solaris kernel, postgreSQL, x.org, ...
- Can create new probes dynamically
- “Dynamic tracing framework”
- DTrace D programming language

DTrace Implementation

- Well documented probes

Definition

Probe: A location or action which can be hooked into in order to perform some arbitrary action

- Probes placed statically in source code
- Probe location well considered
- Probes in Solaris kernel, postgreSQL, x.org, . . .
- Can create new probes dynamically
- “Dynamic tracing framework”
- DTrace D programming language

DTrace Implementation

- Well documented probes

Definition

Probe: A location or action which can be hooked into in order to perform some arbitrary action

- Probes placed statically in source code
- Probe location well considered
- Probes in Solaris kernel, postgreSQL, x.org, . . .
- Can create new probes dynamically
- “Dynamic tracing framework”
- DTrace D programming language

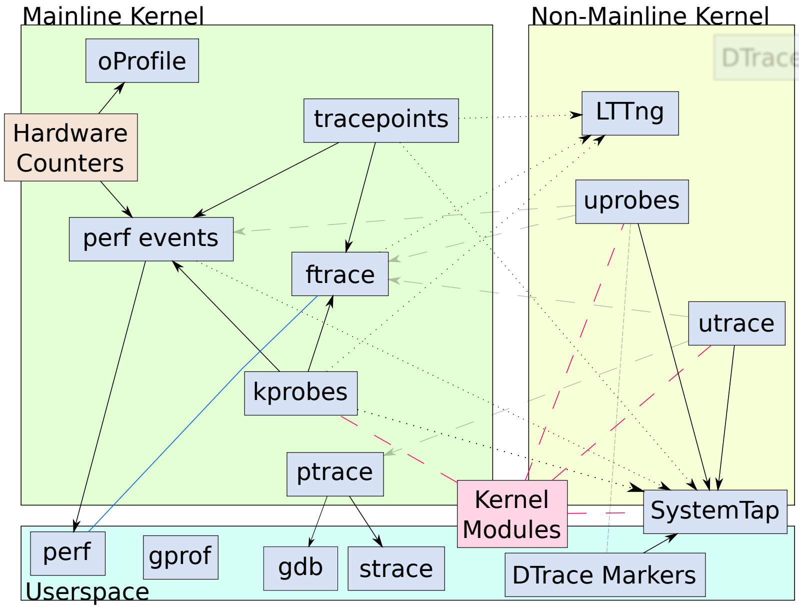
DTrace on Linux?

- DTrace approach generally unsuitable for Linux kernel
- Linux kernel evolves rapidly
- Resistance to placement of static probes
- Unofficial Linux port does exist
- SystemTap can use DTrace markers

DTrace on Linux?

- DTrace approach generally unsuitable for Linux kernel
- Linux kernel evolves rapidly
- Resistance to placement of static probes
- Unofficial Linux port does exist
- SystemTap can use DTrace markers

Enough DTrace envy, what about Linux?


Mainline Kernel
Non-Mainline Kernel
DTrace
oProfile
tracepoints
LTTng
Hardware Counters
uprobes
perf events
ftrace
utrace
kprobes
ptrace
Kernel Modules
SystemTap
perf
gprof
gdb
strace
DTrace Markers
Userspace

Static probes? tracepoints

Static probes? tracepoints

Tracepoints

- Static probe points in the kernel source
- Replace old Kernel Markers
- Infrastructure mainline
- Low performance impact
- TRACE_EVENT() macro

Kernel dynamic probes? kprobes

Kernel dynamic probes? kprobes

Kprobes

- Insert a new probe into kernel at runtime
- Does not do userspace probes
- Heavily architecture specific
- Can be used by loadable kernel modules
- Used by other instrumentation tools
- Provides three types of probes
    - kprobes—Probe instruction
    - jprobes—Probe function call
    - kretprobes—Probe function return
- Mainline

Userspace dynamic probes? uprobes
Not mainline yet.

Userspace dynamic probes? uprobes

Not mainline yet.

Userspace dynamic probes? uprobes
Not mainline yet.

Uprobes

- Kprobes for userspace
- Not mainline yet—predict that may soon change
- Used by SystemTap
- Handlers run in task context
- Background page replacement mechanism
- Probed instruction single stepped in XOL area
- No longer depends on utrace
- Handlers implemented as kernel module
- perf interface

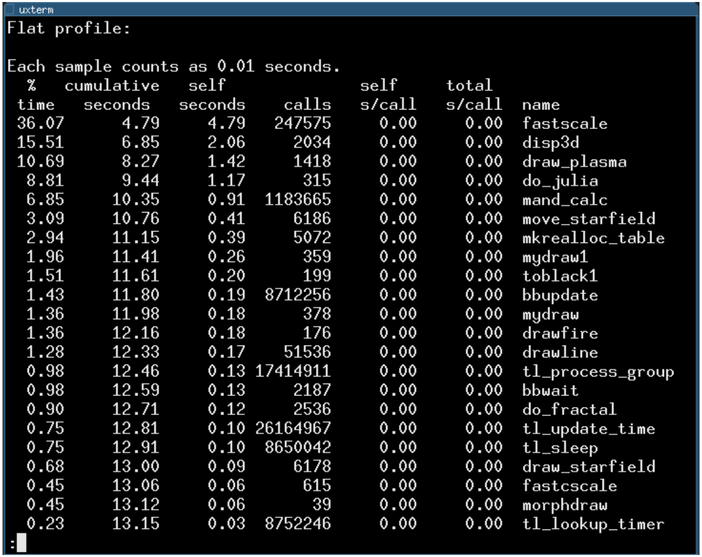
Userspace profiling? gprof

Userspace profiling? gprof

gprof

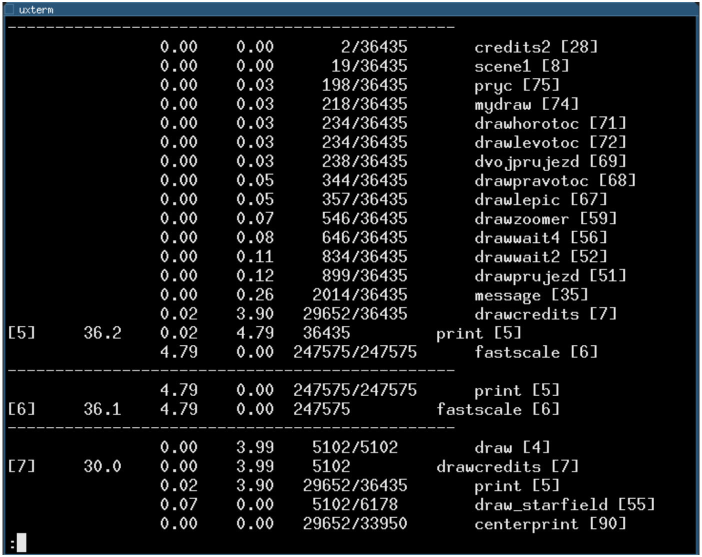
- Legacy, but deserves a mention
- Counts and profiles calls to functions
- Function granuality
- Compile executable with -pg flag
- Produce function call graph
- Does NOT profile libraries or kernel
- Kprof provides GUI to visualise call graph

gprof Flat Profile



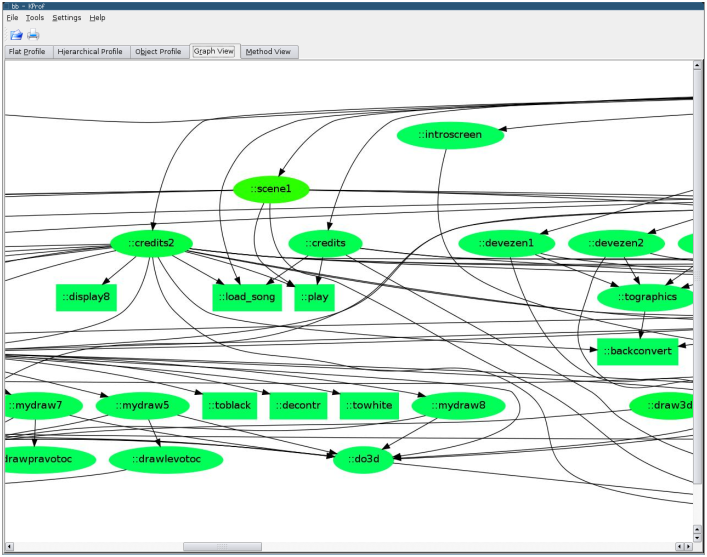
```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 36.07      4.79     4.79   247575     0.00     0.00  fastscale
 15.51      6.85     2.06     2034     0.00     0.00  disp3d
 10.69      8.27     1.42     1418     0.00     0.00  draw_plasma
  8.81      9.44     1.17      315     0.00     0.00  do_julia
  6.85     10.35     0.91  1183665     0.00     0.00  mand_calc
  3.09     10.76     0.41     6186     0.00     0.00  move_starfield
  2.94     11.15     0.39     5072     0.00     0.00  mkrealloc_table
  1.96     11.41     0.26      359     0.00     0.00  mydraw1
  1.51     11.61     0.20      199     0.00     0.00  toblack1
  1.43     11.80     0.19  8712256     0.00     0.00  bbupdate
  1.36     11.98     0.18      378     0.00     0.00  mydraw
  1.36     12.16     0.18      176     0.00     0.00  drawfire
  1.28     12.33     0.17    51536     0.00     0.00  drawline
  0.98     12.46     0.13 17414911     0.00     0.00  tl_process_group
  0.98     12.59     0.13     2187     0.00     0.00  bbwait
  0.90     12.71     0.12     2536     0.00     0.00  do_fractal
  0.75     12.81     0.10 26164967     0.00     0.00  tl_update_time
  0.75     12.91     0.10  8650042     0.00     0.00  tl_sleep
  0.68     13.00     0.09     6178     0.00     0.00  draw_starfield
  0.45     13.06     0.06      615     0.00     0.00  fastcscale
  0.45     13.12     0.06       39     0.00     0.00  morphdraw
  0.23     13.15     0.03  8752246     0.00     0.00  tl_lookup_timer
:
```

gprof Call Graph



```
-----------------------------------------------
                0.00    0.00       2/36435       credits2 [28]
                0.00    0.00      19/36435       scene1 [8]
                0.00    0.03     198/36435       pryc [75]
                0.00    0.03     218/36435       mydraw [74]
                0.00    0.03     234/36435       drawhorotoc [71]
                0.00    0.03     234/36435       drawlevotoc [72]
                0.00    0.03     238/36435       dvojprujezd [69]
                0.00    0.05     344/36435       drawpravotoc [68]
                0.00    0.05     357/36435       drawlepic [67]
                0.00    0.07     546/36435       drawzoomer [59]
                0.00    0.08     646/36435       drawwait4 [56]
                0.00    0.11     834/36435       drawwait2 [52]
                0.00    0.12     899/36435       drawprujezd [51]
                0.00    0.26    2014/36435       message [35]
                0.02    3.90   29652/36435       drawcredits [7]
[5]     36.2    0.02    4.79   36435         print [5]
                4.79    0.00  247575/247575      fastscale [6]
-----------------------------------------------
                4.79    0.00  247575/247575      print [5]
[6]     36.1    4.79    0.00  247575         fastscale [6]
-----------------------------------------------
                0.00    3.99    5102/5102        draw [4]
[7]     30.0    0.00    3.99    5102         drawcredits [7]
                0.02    3.90   29652/36435       print [5]
                0.07    0.00    5102/6178        draw_starfield [55]
                0.00    0.00   29652/33950       centerprint [90]
:
```

kprof Profile Visualisation Tool



Userspace tracing? ptrace . . .
A bit ugly.

Userspace tracing? ptrace . . .

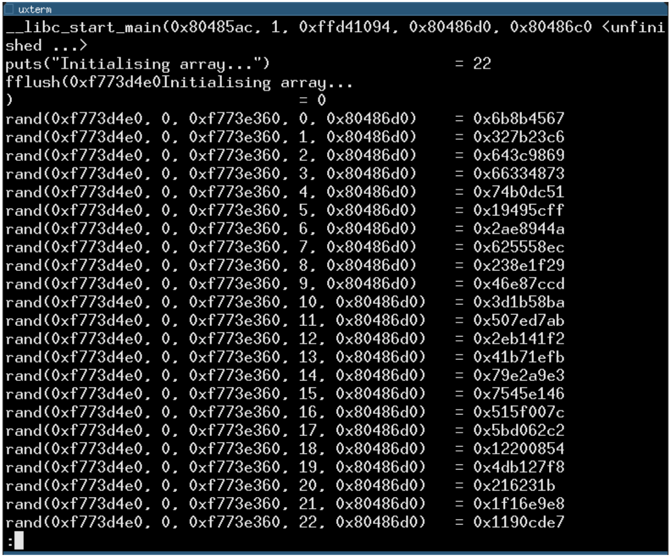
A bit ugly.

Userspace tracing? ptrace . . .
A bit ugly.

ptrace

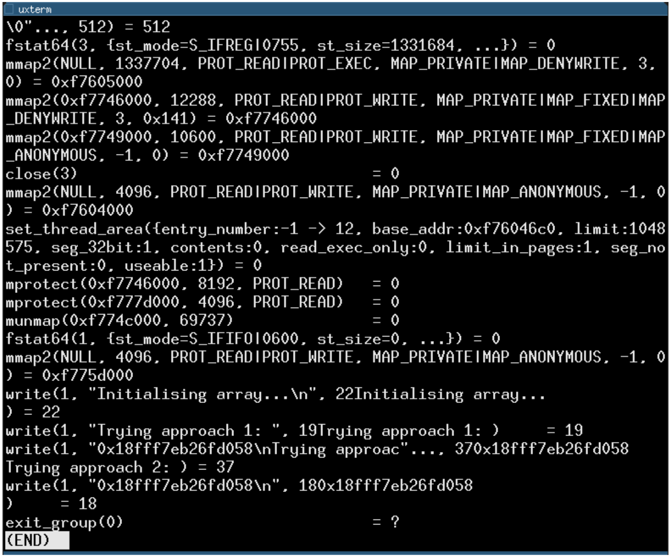
- Means of one process observing/controlling another
- Prominently used by gdb, strace, ltrace
- Ugly, limited interface
- Processes cannot be traced by multiple processes
- Signal oriented architecture
- Large overhead

ptrace—ltrace



```
__libc_start_main(0x80485ac, 1, 0xffd41094, 0x80486d0, 0x80486c0 <unfini
shed ...>
puts("Initialising array...")                         = 22
fflush(0xf773d4e0Initialising array...
)                                = 0
rand(0xf773d4e0, 0, 0xf773e360, 0, 0x80486d0)      = 0x6b8b4567
rand(0xf773d4e0, 0, 0xf773e360, 1, 0x80486d0)      = 0x327b23c6
rand(0xf773d4e0, 0, 0xf773e360, 2, 0x80486d0)      = 0x643c9869
rand(0xf773d4e0, 0, 0xf773e360, 3, 0x80486d0)      = 0x66334873
rand(0xf773d4e0, 0, 0xf773e360, 4, 0x80486d0)      = 0x74b0dc51
rand(0xf773d4e0, 0, 0xf773e360, 5, 0x80486d0)      = 0x19495cff
rand(0xf773d4e0, 0, 0xf773e360, 6, 0x80486d0)      = 0x2ae8944a
rand(0xf773d4e0, 0, 0xf773e360, 7, 0x80486d0)      = 0x625558ec
rand(0xf773d4e0, 0, 0xf773e360, 8, 0x80486d0)      = 0x238e1f29
rand(0xf773d4e0, 0, 0xf773e360, 9, 0x80486d0)      = 0x46e87ccd
rand(0xf773d4e0, 0, 0xf773e360, 10, 0x80486d0)     = 0x3d1b58ba
rand(0xf773d4e0, 0, 0xf773e360, 11, 0x80486d0)     = 0x507ed7ab
rand(0xf773d4e0, 0, 0xf773e360, 12, 0x80486d0)     = 0x2eb141f2
rand(0xf773d4e0, 0, 0xf773e360, 13, 0x80486d0)     = 0x41b71efb
rand(0xf773d4e0, 0, 0xf773e360, 14, 0x80486d0)     = 0x79e2a9e3
rand(0xf773d4e0, 0, 0xf773e360, 15, 0x80486d0)     = 0x7545e146
rand(0xf773d4e0, 0, 0xf773e360, 16, 0x80486d0)     = 0x515f007c
rand(0xf773d4e0, 0, 0xf773e360, 17, 0x80486d0)     = 0x5bd062c2
rand(0xf773d4e0, 0, 0xf773e360, 18, 0x80486d0)     = 0x12200854
rand(0xf773d4e0, 0, 0xf773e360, 19, 0x80486d0)     = 0x4db127f8
rand(0xf773d4e0, 0, 0xf773e360, 20, 0x80486d0)     = 0x216231b
rand(0xf773d4e0, 0, 0xf773e360, 21, 0x80486d0)     = 0x1f16e9e8
rand(0xf773d4e0, 0, 0xf773e360, 22, 0x80486d0)     = 0x1190cde7
:
```

ptrace—strace



```
\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1331684, ...}) = 0
mmap2(NULL, 1337704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0xf7605000
mmap2(0xf7746000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP
_DENYWRITE, 3, 0x141) = 0xf7746000
mmap2(0xf7749000, 10600, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP
_ANONYMOUS, -1, 0) = 0xf7749000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
) = 0xf7604000
set_thread_area({entry_number:-1 -> 12, base_addr:0xf76046c0, limit:1048
575, seg_32bit:1, contents:0, read_exec_only:0, limit_in_pages:1, seg_no
t_present:0, useable:1}) = 0
mprotect(0xf7746000, 8192, PROT_READ)   = 0
mprotect(0xf777d000, 4096, PROT_READ)   = 0
munmap(0xf774c000, 69737)               = 0
fstat64(1, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
) = 0xf775d000
write(1, "Initialising array...\n", 22Initialising array...
) = 22
write(1, "Trying approach 1: ", 19Trying approach 1: )      = 19
write(1, "0x18fff7eb26fd058\nTrying approac"..., 370x18fff7eb26fd058
Trying approach 2: ) = 37
write(1, "0x18fff7eb26fd058\n", 180x18fff7eb26fd058
)       = 18
exit_group(0)                           = ?
(END)
```

Improved userspace tracing? utrace
Not mainline yet.

Improved userspace tracing? utrace

Not mainline yet.

Improved userspace tracing? utrace
Not mainline yet.

utrace

- Improved userspace tracing
- Not mainline
- Prominently used by SystemTap
- Utrace clients run in kernel, typically in module
- Ptrace reimplemented as a utrace client
- Infrastructure to monitor threads
- Establish “engine” for each monitored thread
  - Event reporting
  - Thread control
  - Thread machine state access

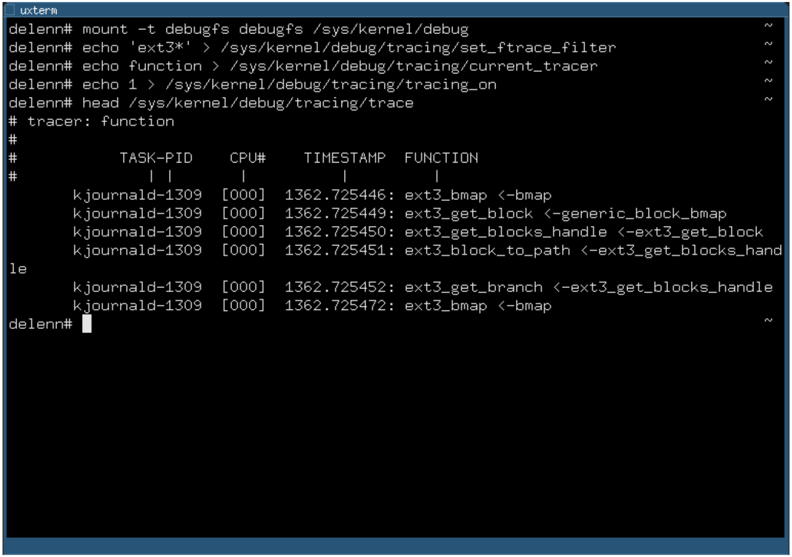
Kernel tracing? ftrace

Kernel tracing? ftrace

ftrace

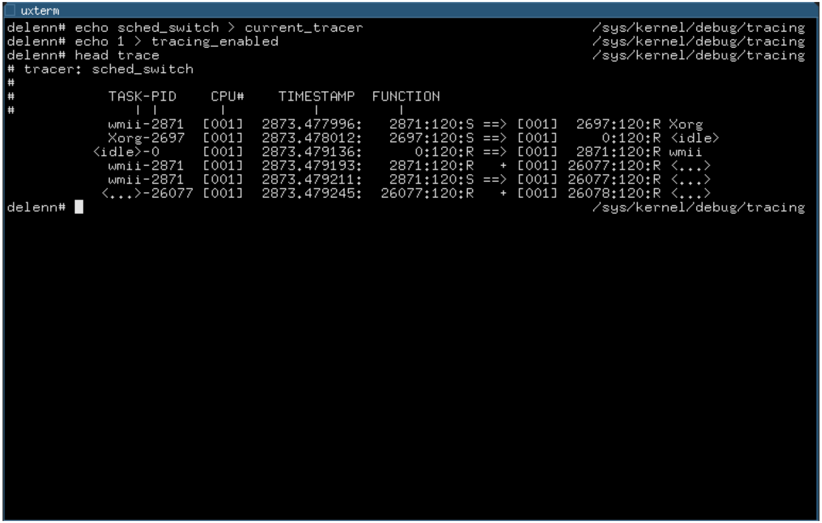
- Mainline
- Kernel tracing infrastructure
- Originated from the realtime efforts
- Hijacks mcount (from gprof) for own purposes
- Low performance impact
- Tracing Plugins
    - Function
    - Funtion graph
    - Context switches
    - Time interrupts disabled
    - Time preemtion disabled
    - Delay for high priority tasks
    - Power state transitions
    - Branch prediction
    - . . .
- Exposed through debugfs—Manipulate with echo & cat
- Gained event support—perf events favoured for this

ftrace—Function Tracer



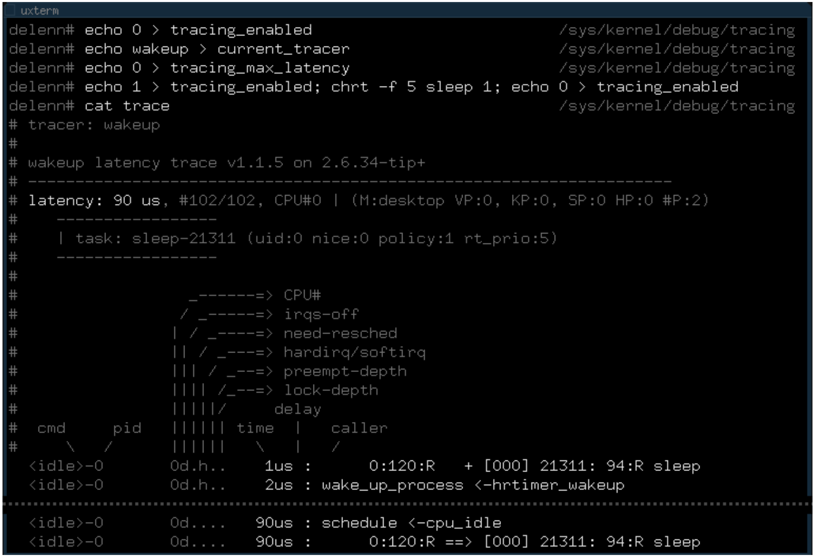
```
delenn# mount -t debugfs debugfs /sys/kernel/debug
delenn# echo 'ext3*' > /sys/kernel/debug/tracing/set_ftrace_filter
delenn# echo function > /sys/kernel/debug/tracing/current_tracer
delenn# echo 1 > /sys/kernel/debug/tracing/tracing_on
delenn# head /sys/kernel/debug/tracing/trace
# tracer: function
#
#           TASK-PID    CPU#    TIMESTAMP  FUNCTION
#              | |       |          |         |
       kjournald-1309  [000]  1362.725446: ext3_bmap <-bmap
       kjournald-1309  [000]  1362.725449: ext3_get_block <-generic_block_bmap
       kjournald-1309  [000]  1362.725450: ext3_get_blocks_handle <-ext3_get_block
       kjournald-1309  [000]  1362.725451: ext3_block_to_path <-ext3_get_blocks_hand
le
       kjournald-1309  [000]  1362.725452: ext3_get_branch <-ext3_get_blocks_handle
       kjournald-1309  [000]  1362.725472: ext3_bmap <-bmap
delenn#
```

ftrace—Task scheduling


uxterm

```
delenn# echo sched_switch > current_tracer                                 /sys/kernel/debug/tracing
delenn# echo 1 > tracing_enabled                                           /sys/kernel/debug/tracing
delenn# head trace                                                         /sys/kernel/debug/tracing
# tracer: sched_switch
#
#           TASK-PID    CPU#    TIMESTAMP  FUNCTION
#              | |       |          |         |
            wmii-2871  [001]  2873.477996:   2871:120:S ==> [001]  2697:120:R Xorg
            Xorg-2697  [001]  2873.478012:   2697:120:S ==> [001]     0:120:R <idle>
          <idle>-0     [001]  2873.479136:      0:120:R ==> [001]  2871:120:R wmii
            wmii-2871  [001]  2873.479193:   2871:120:R   + [001] 26077:120:R <...>
            wmii-2871  [001]  2873.479211:   2871:120:S ==> [001] 26077:120:R <...>
           <...>-26077 [001]  2873.479245:  26077:120:R   + [001] 26078:120:R <...>
delenn#                                                                    /sys/kernel/debug/tracing
```

ftrace—Measuring task wakeup latency


uxterm

```
delenn# echo 0 > tracing_enabled                          /sys/kernel/debug/tracing
delenn# echo wakeup > current_tracer                      /sys/kernel/debug/tracing
delenn# echo 0 > tracing_max_latency                      /sys/kernel/debug/tracing
delenn# echo 1 > tracing_enabled; chrt -f 5 sleep 1; echo 0 > tracing_enabled
delenn# cat trace                                         /sys/kernel/debug/tracing
# tracer: wakeup
#
# wakeup latency trace v1.1.5 on 2.6.34-tip+
# --------------------------------------------------------------------
# latency: 90 us, #102/102, CPU#0 | (M:desktop VP:0, KP:0, SP:0 HP:0 #P:2)
#    -----------------
#    | task: sleep-21311 (uid:0 nice:0 policy:1 rt_prio:5)
#    -----------------
#
#                  _------=> CPU#
#                 / _-----=> irqs-off
#                | / _----=> need-resched
#                || / _---=> hardirq/softirq
#                ||| / _--=> preempt-depth
#                |||| /_--=> lock-depth
#                |||||/     delay
#  cmd     pid   |||||| time  |   caller
#     \   /      ||||||   \   |   /
  <idle>-0       0d.h..    1us :      0:120:R   + [000] 21311: 94:R sleep
  <idle>-0       0d.h..    2us : wake_up_process <-hrtimer_wakeup
..........................................................................
  <idle>-0       0d....   90us : schedule <-cpu_idle
  <idle>-0       0d....   90us :      0:120:R ==> [000] 21311: 94:R sleep
```

Kernel and Userspace tracing? LTTng
Not mainline.

Kernel and Userspace tracing? LTTng

Not mainline.

Kernel and Userspace tracing? LTTng
Not mainline.

LTTng

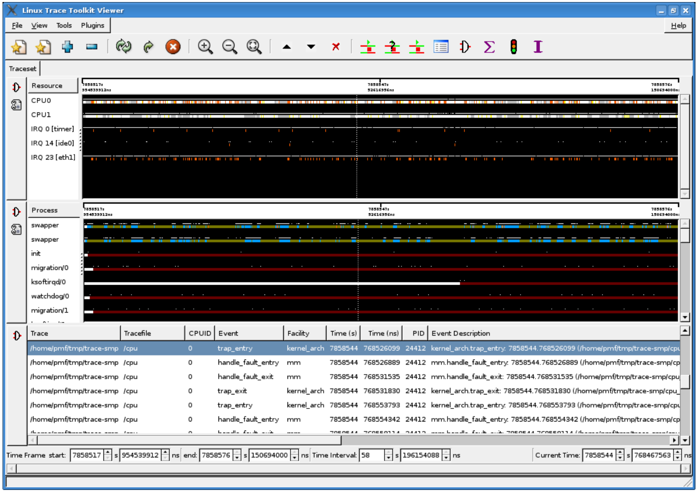
- Kernel & Userspace tracing package
- Not mainline
- Adds instrumentation points into kernel
- Uses tracepoints, ftrace, kprobes
- Can plot traces
- Completely failed to work for me in every way

LTTng

- Kernel & Userspace tracing package
- Not mainline
- Adds instrumentation points into kernel
- Uses tracepoints, ftrace, kprobes
- Can plot traces
- Completely failed to work for me in every way

LTTng Screenshot

LTTng did not work for me, but here's what it's supposed to do:



Trace	Tracefile	CPUID	Event	Facility	Time (s)	Time (ns)	PID	Event Description
/home/pmf/tmp/trace-smp	/cpu	0	trap_entry	kernel_arch	7858544	768526099	24412	kernel_arch.trap_entry: 7858544.768526099 (/home/pmf/tmp/trace-smp/cpu
/home/pmf/tmp/trace-smp	/cpu	0	handle_fault_entry	mm	7858544	768526889	24412	mm.handle_fault_entry: 7858544.768526889 (/home/pmf/tmp/trace-smp/cp
/home/pmf/tmp/trace-smp	/cpu	0	handle_fault_exit	mm	7858544	768531535	24412	mm.handle_fault_exit: 7858544.768531535 (/home/pmf/tmp/trace-smp/cpu
/home/pmf/tmp/trace-smp	/cpu	0	trap_exit	kernel_arch	7858544	768531830	24412	kernel_arch.trap_exit: 7858544.768531830 (/home/pmf/tmp/trace-smp/cpu_
/home/pmf/tmp/trace-smp	/cpu	0	trap_entry	kernel_arch	7858544	768553793	24412	kernel_arch.trap_entry: 7858544.768553793 (/home/pmf/tmp/trace-smp/cpu
/home/pmf/tmp/trace-smp	/cpu	0	handle_fault_entry	mm	7858544	768554342	24412	mm.handle_fault_entry: 7858544.768554342 (/home/pmf/tmp/trace-smp/cp

Time Frame start: 7858517 s 954539912 ns end: 7858576 s 150694000 ns Time Interval: 58 s 196154088 ns Current Time: 7858544 s 768467563 ns

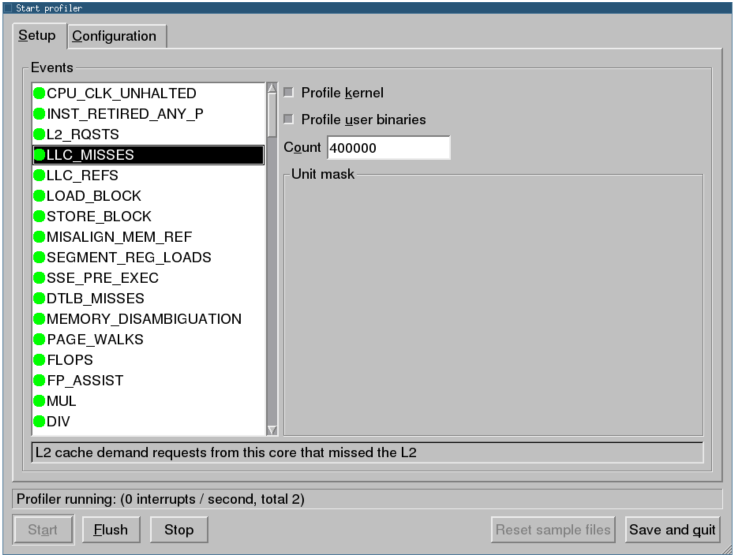
Hardware performance counters? oProfile

Hardware performance counters? oProfile

oProfile

- System wide sampling profiler
- Mainline
- Interface to Performance Measurement Unit
- GUI eases selection of performance counters
- Can generate gprof style call graph

oProfile GUI



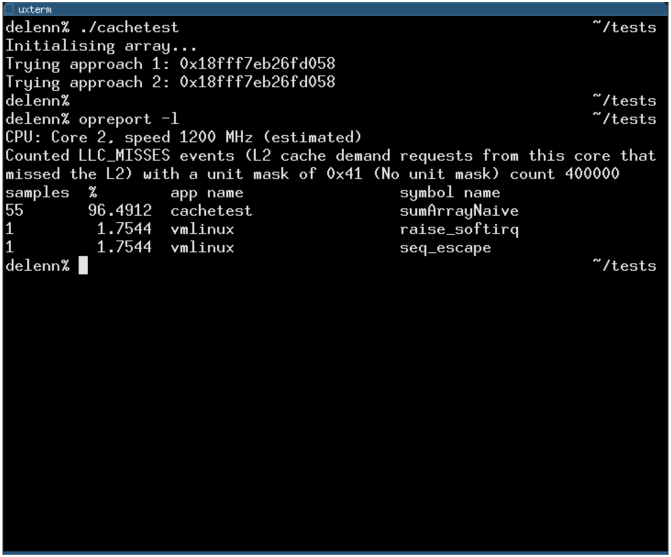
● CPU_CLK_UNHALTED
● INST_RETIRED_ANY_P
● L2_RQSTS
● LLC_MISSES
● LLC_REFS
● LOAD_BLOCK
● STORE_BLOCK
● MISALIGN_MEM_REF
● SEGMENT_REG_LOADS
● SSE_PRE_EXEC
● DTLB_MISSES
● MEMORY_DISAMBIGUATION
● PAGE_WALKS
● FLOPS
● FP_ASSIST
● MUL
● DIV

Profile kernel
Profile user binaries
Count 400000

L2 cache demand requests from this core that missed the L2

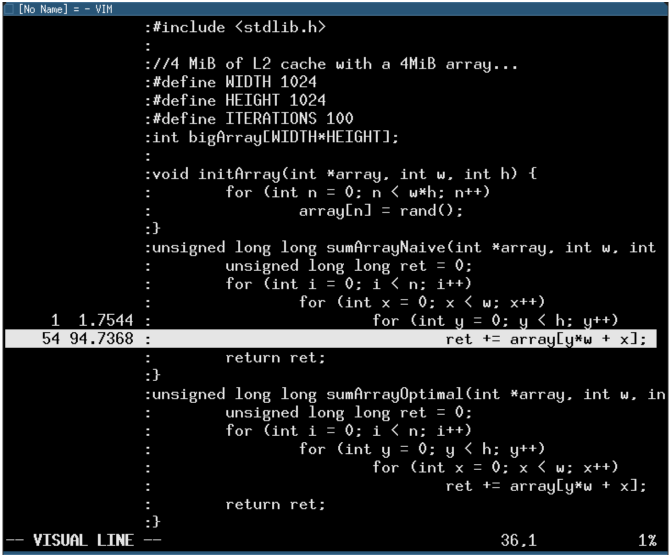
Profiler running: (0 interrupts / second, total 2)

oProfile Report



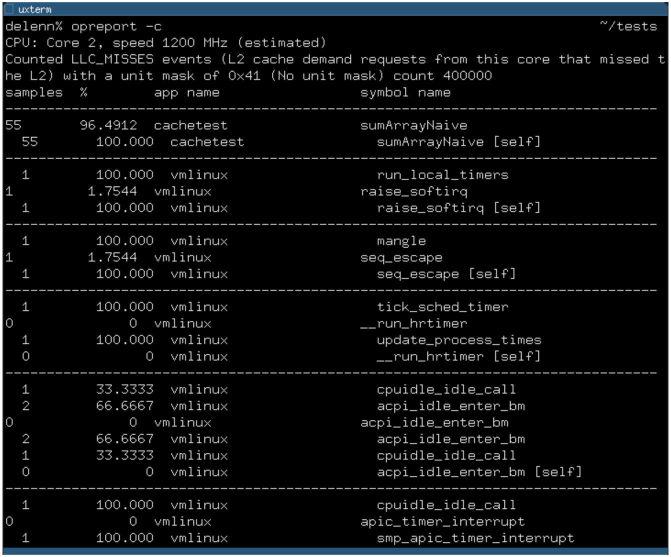
```
delenn% ./cachetest                                                           ~/tests
Initialising array...
Trying approach 1: 0x18fff7eb26fd058
Trying approach 2: 0x18fff7eb26fd058
delenn%                                                                       ~/tests
delenn% opreport -l                                                           ~/tests
CPU: Core 2, speed 1200 MHz (estimated)
Counted LLC_MISSES events (L2 cache demand requests from this core that
missed the L2) with a unit mask of 0x41 (No unit mask) count 400000
samples  %        app name                 symbol name
55       96.4912  cachetest                sumArrayNaive
1         1.7544  vmlinux                  raise_softirq
1         1.7544  vmlinux                  seq_escape
delenn%                                                                       ~/tests
```

oProfile Annotate



```
[No Name] = - VIM
               :#include <stdlib.h>
               :
               ://4 MiB of L2 cache with a 4MiB array...
               :#define WIDTH 1024
               :#define HEIGHT 1024
               :#define ITERATIONS 100
               :int bigArray[WIDTH*HEIGHT];
               :
               :void initArray(int *array, int w, int h) {
               :        for (int n = 0; n < w*h; n++)
               :                array[n] = rand();
               :}
               :unsigned long long sumArrayNaive(int *array, int w, int
               :        unsigned long long ret = 0;
               :        for (int i = 0; i < n; i++)
               :                for (int x = 0; x < w; x++)
     1  1.7544 :                        for (int y = 0; y < h; y++)
    54 94.7368 :                                ret += array[y*w + x];
               :        return ret;
               :}
               :unsigned long long sumArrayOptimal(int *array, int w, in
               :        unsigned long long ret = 0;
               :        for (int i = 0; i < n; i++)
               :                for (int y = 0; y < h; y++)
               :                        for (int x = 0; x < w; x++)
               :                                ret += array[y*w + x];
               :        return ret;
               :}
-- VISUAL LINE --                                    36,1          1%
```

oProfile Call Graph


uxterm

```
delenn% opreport -c                                                   ~/tests
CPU: Core 2, speed 1200 MHz (estimated)
Counted LLC_MISSES events (L2 cache demand requests from this core that missed t
he L2) with a unit mask of 0x41 (No unit mask) count 400000
samples  %        app name                 symbol name
-------------------------------------------------------------------------------
55       96.4912  cachetest                sumArrayNaive
  55       100.000  cachetest                sumArrayNaive [self]
-------------------------------------------------------------------------------
  1        100.000  vmlinux                  run_local_timers
1         1.7544  vmlinux                  raise_softirq
  1        100.000  vmlinux                  raise_softirq [self]
-------------------------------------------------------------------------------
  1        100.000  vmlinux                  mangle
1         1.7544  vmlinux                  seq_escape
  1        100.000  vmlinux                  seq_escape [self]
-------------------------------------------------------------------------------
  1        100.000  vmlinux                  tick_sched_timer
0              0  vmlinux                  __run_hrtimer
  1        100.000  vmlinux                  update_process_times
  0              0  vmlinux                  __run_hrtimer [self]
-------------------------------------------------------------------------------
  1        33.3333  vmlinux                  cpuidle_idle_call
  2        66.6667  vmlinux                  acpi_idle_enter_bm
0              0  vmlinux                  acpi_idle_enter_bm
  2        66.6667  vmlinux                  acpi_idle_enter_bm
  1        33.3333  vmlinux                  cpuidle_idle_call
  0              0  vmlinux                  acpi_idle_enter_bm [self]
-------------------------------------------------------------------------------
  1        100.000  vmlinux                  cpuidle_idle_call
0              0  vmlinux                  apic_timer_interrupt
  1        100.000  vmlinux                  smp_apic_timer_interrupt
```

Something to pull everything together? SystemTap
Not mainline.

Something to pull everything together? SystemTap
Not mainline.

Something to pull everything together? SystemTap
Not mainline.

SystemTap

- Powerful kernel & userspace analysis suite
- Scripting language
- Generates, Compiles and loads kernel modules
- Primarily uses kprobes
- Utrace for userspace tracing
- Uprobes support
- Many userspace dependencies
- Hard to use
- More of interest to enterprise distros
- Version 1.2 recently released
  - Prototypical support for perf events
  - Hardware breakpoint probe support
- Not mainline and . . .

SystemTap

Let's face it, system tap isn't going to be merged, so why even bring it up? Every kernel developer I have _ever_ seen agrees that all the new tracing is a million times superior. I'm sure there are system tap people who disagree, but quite frankly, I don't see it being merged considering how little the system tap people ever did for the kernel.

So if things like system tap and "security models that go behind the kernel by tying into utrace" are the reasons for utrace, color me utterly uninterested. In fact, color me actively hostile. I think that's the worst possible situation that we'd ever be in as kernel people (namely exactly the "do things in kernel space by hiding behind utrace without having kernel people involved")

Linus

Something else to pull everything together? perf events
The future of Linux Instrumentation.

Something else to pull everything together? perf events

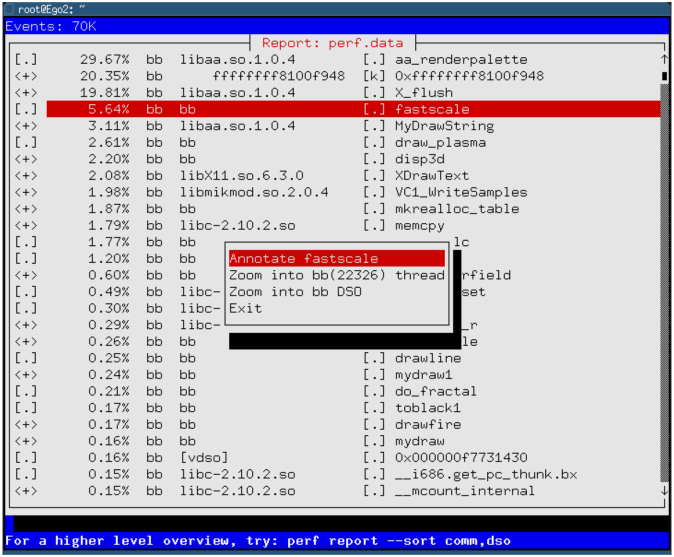
The future of Linux Instrumentation.

Something else to pull everything together? perf events
The future of Linux Instrumentation.

perf events

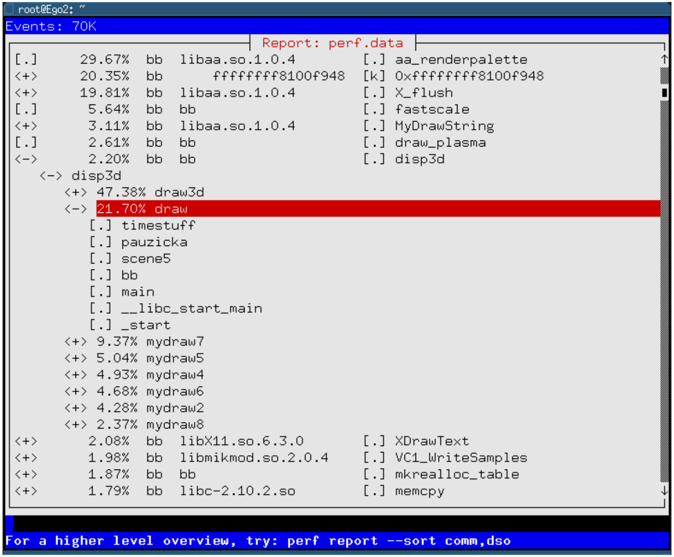
- All in one Performance tool
- Newcomer, but already mainline
- Interface to Performance Measurement Unit
- Virtual counters
- Integrates with tracepoints
- Kprobe support
- Scripting support
- SLAB subsystem profiling
- Lock profiling
- Scheduler analyser
- Timechart

Sample perf report



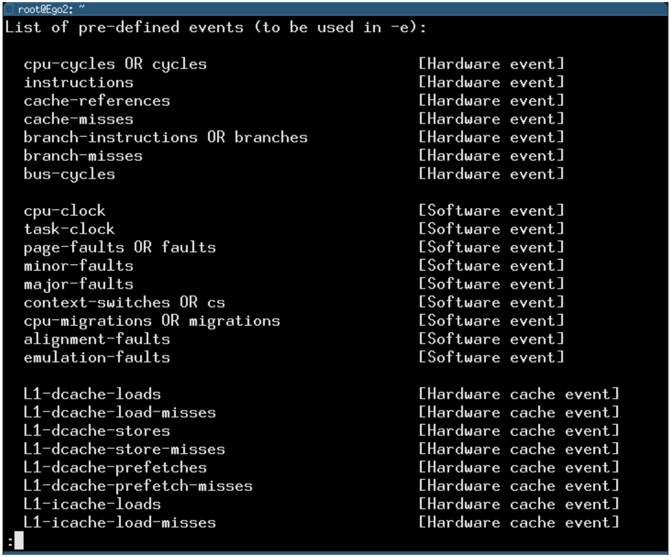
```
root@Ego2: ~
Events: 70K
                                     Report: perf.data
[.]      29.67%  bb  libaa.so.1.0.4        [.] aa_renderpalette
<+>      20.35%  bb      ffffffff8100f948  [k] 0xffffffff8100f948
<+>      19.81%  bb  libaa.so.1.0.4        [.] X_flush
[.]       5.64%  bb  bb                    [.] fastscale
<+>       3.11%  bb  libaa.so.1.0.4        [.] MyDrawString
[.]       2.61%  bb  bb                    [.] draw_plasma
<+>       2.20%  bb  bb                    [.] disp3d
<+>       2.08%  bb  libX11.so.6.3.0       [.] XDrawText
<+>       1.98%  bb  libmikmod.so.2.0.4    [.] VC1_WriteSamples
<+>       1.87%  bb  bb                    [.] mkrealloc_table
<+>       1.79%  bb  libc-2.10.2.so        [.] memcpy
[.]       1.77%  bb  bb                               lc
[.]       1.20%  bb  bb    Annotate fastscale
<+>       0.60%  bb  bb    Zoom into bb(22326) thread rfield
[.]       0.49%  bb  libc- Zoom into bb DSO           set
[.]       0.30%  bb  libc- Exit
<+>       0.29%  bb  libc-                            _r
<+>       0.26%  bb  bb                               le
[.]       0.25%  bb  bb                    [.] drawline
<+>       0.24%  bb  bb                    [.] mydraw1
[.]       0.21%  bb  bb                    [.] do_fractal
[.]       0.17%  bb  bb                    [.] toblack1
<+>       0.17%  bb  bb                    [.] drawfire
<+>       0.16%  bb  bb                    [.] mydraw
[.]       0.16%  bb  [vdso]                [.] 0x000000f7731430
[.]       0.15%  bb  libc-2.10.2.so        [.] __i686.get_pc_thunk.bx
<+>       0.15%  bb  libc-2.10.2.so        [.] __mcount_internal

For a higher level overview, try: perf report --sort comm,dso
```

Sample perf call graph



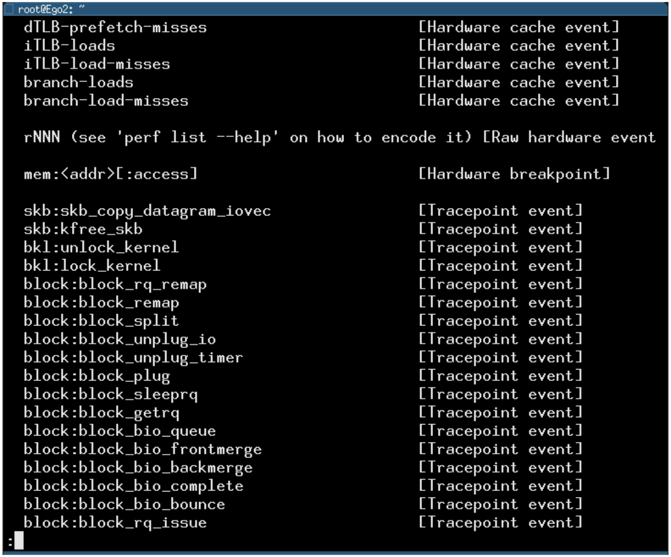
```
root@Ego2: ~
Events: 70K
                                   Report: perf.data
[.]      29.67%  bb  libaa.so.1.0.4      [.] aa_renderpalette
<+>      20.35%  bb      ffffffff8100f948 [k] 0xffffffff8100f948
<+>      19.81%  bb  libaa.so.1.0.4      [.] X_flush
[.]       5.64%  bb  bb                  [.] fastscale
<+>       3.11%  bb  libaa.so.1.0.4      [.] MyDrawString
[.]       2.61%  bb  bb                  [.] draw_plasma
<->       2.20%  bb  bb                  [.] disp3d
   <-> disp3d
      <+> 47.38% draw3d
      <-> 21.70% draw
         [.] timestuff
         [.] pauzicka
         [.] scene5
         [.] bb
         [.] main
         [.] __libc_start_main
         [.] _start
      <+> 9.37% mydraw7
      <+> 5.04% mydraw5
      <+> 4.93% mydraw4
      <+> 4.68% mydraw6
      <+> 4.28% mydraw2
      <+> 2.37% mydraw8
<+>       2.08%  bb  libX11.so.6.3.0     [.] XDrawText
<+>       1.98%  bb  libmikmod.so.2.0.4  [.] VC1_WriteSamples
<+>       1.87%  bb  bb                  [.] mkrealloc_table
<+>       1.79%  bb  libc-2.10.2.so      [.] memcpy

For a higher level overview, try: perf report --sort comm,dso
```

perf—Available events



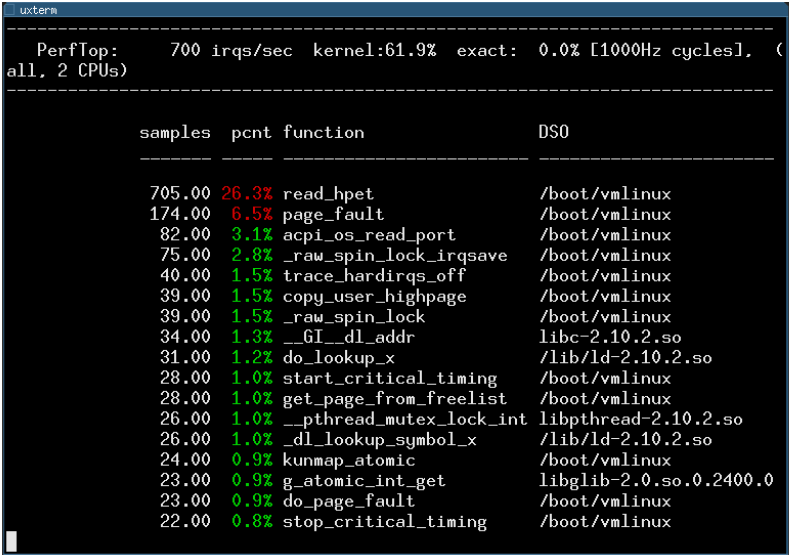
```
root@Ego2: ~
List of pre-defined events (to be used in -e):

  cpu-cycles OR cycles                           [Hardware event]
  instructions                                   [Hardware event]
  cache-references                               [Hardware event]
  cache-misses                                   [Hardware event]
  branch-instructions OR branches                [Hardware event]
  branch-misses                                  [Hardware event]
  bus-cycles                                     [Hardware event]

  cpu-clock                                      [Software event]
  task-clock                                     [Software event]
  page-faults OR faults                          [Software event]
  minor-faults                                   [Software event]
  major-faults                                   [Software event]
  context-switches OR cs                         [Software event]
  cpu-migrations OR migrations                   [Software event]
  alignment-faults                               [Software event]
  emulation-faults                               [Software event]

  L1-dcache-loads                                [Hardware cache event]
  L1-dcache-load-misses                          [Hardware cache event]
  L1-dcache-stores                               [Hardware cache event]
  L1-dcache-store-misses                         [Hardware cache event]
  L1-dcache-prefetches                           [Hardware cache event]
  L1-dcache-prefetch-misses                      [Hardware cache event]
  L1-icache-loads                                [Hardware cache event]
  L1-icache-load-misses                          [Hardware cache event]
:
```

perf—Available events



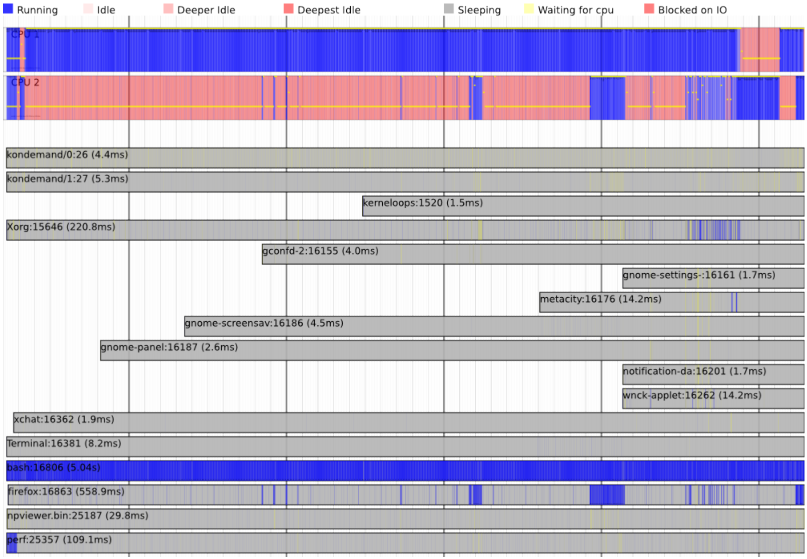
```
root@Ego2: ~
  dTLB-prefetch-misses                       [Hardware cache event]
  iTLB-loads                                 [Hardware cache event]
  iTLB-load-misses                           [Hardware cache event]
  branch-loads                               [Hardware cache event]
  branch-load-misses                         [Hardware cache event]

  rNNN (see 'perf list --help' on how to encode it) [Raw hardware event

  mem:<addr>[:access]                        [Hardware breakpoint]

  skb:skb_copy_datagram_iovec                [Tracepoint event]
  skb:kfree_skb                              [Tracepoint event]
  bkl:unlock_kernel                          [Tracepoint event]
  bkl:lock_kernel                            [Tracepoint event]
  block:block_rq_remap                       [Tracepoint event]
  block:block_remap                          [Tracepoint event]
  block:block_split                          [Tracepoint event]
  block:block_unplug_io                      [Tracepoint event]
  block:block_unplug_timer                   [Tracepoint event]
  block:block_plug                           [Tracepoint event]
  block:block_sleeprq                        [Tracepoint event]
  block:block_getrq                          [Tracepoint event]
  block:block_bio_queue                      [Tracepoint event]
  block:block_bio_frontmerge                 [Tracepoint event]
  block:block_bio_backmerge                  [Tracepoint event]
  block:block_bio_complete                   [Tracepoint event]
  block:block_bio_bounce                     [Tracepoint event]
  block:block_rq_issue                       [Tracepoint event]
:
```

perf top

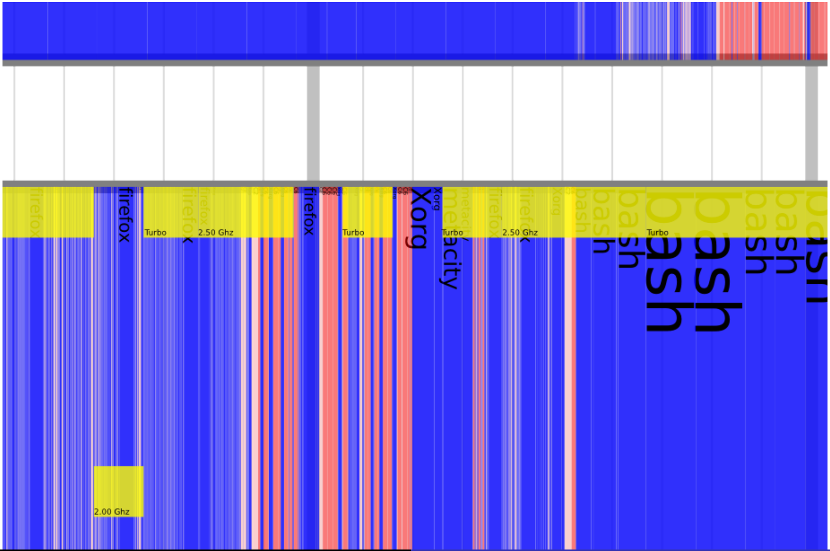

uxterm

```
------------------------------------------------------------------------------
   PerfTop:     700 irqs/sec  kernel:61.9%  exact:  0.0% [1000Hz cycles],  (
all, 2 CPUs)
------------------------------------------------------------------------------

             samples  pcnt function                  DSO
             _______ _____ ________________________ ________________________

              705.00 26.3% read_hpet                 /boot/vmlinux
              174.00  6.5% page_fault                /boot/vmlinux
               82.00  3.1% acpi_os_read_port         /boot/vmlinux
               75.00  2.8% _raw_spin_lock_irqsave    /boot/vmlinux
               40.00  1.5% trace_hardirqs_off        /boot/vmlinux
               39.00  1.5% copy_user_highpage        /boot/vmlinux
               39.00  1.5% _raw_spin_lock            /boot/vmlinux
               34.00  1.3% __GI__dl_addr             libc-2.10.2.so
               31.00  1.2% do_lookup_x               /lib/ld-2.10.2.so
               28.00  1.0% start_critical_timing     /boot/vmlinux
               28.00  1.0% get_page_from_freelist    /boot/vmlinux
               26.00  1.0% __pthread_mutex_lock_int libpthread-2.10.2.so
               26.00  1.0% _dl_lookup_symbol_x       /lib/ld-2.10.2.so
               24.00  0.9% kunmap_atomic             /boot/vmlinux
               23.00  0.9% g_atomic_int_get          libglib-2.0.so.0.2400.0
               23.00  0.9% do_page_fault             /boot/vmlinux
               22.00  0.8% stop_critical_timing      /boot/vmlinux
```

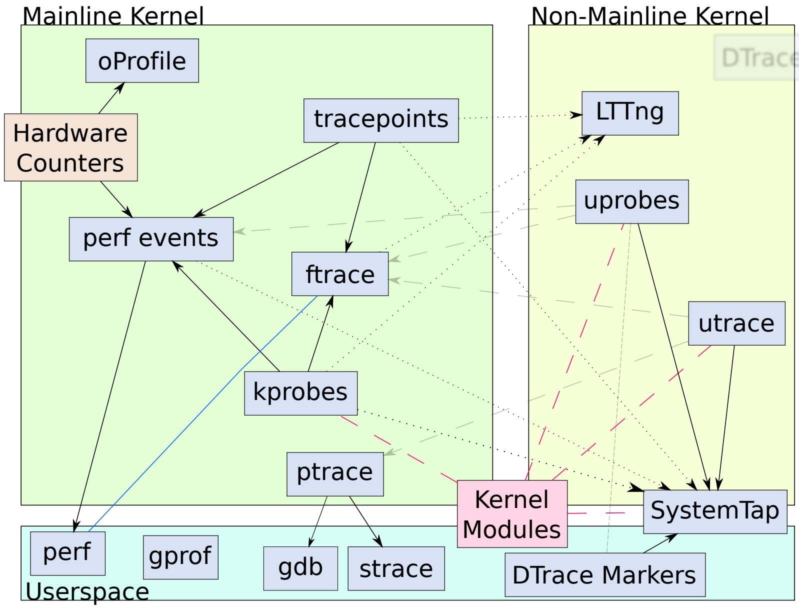
perf timechart



perf timechart



How does it all fit together?


Mainline Kernel
Non-Mainline Kernel
DTrace
oProfile
Hardware Counters
tracepoints
LTTng
uprobes
perf events
ftrace
utrace
kprobes
ptrace
Kernel Modules
SystemTap
perf
gprof
gdb
strace
DTrace Markers
Userspace

Interesting Technologies for the Future

- DTrace
- SystemTap
- LTTng
- gprof
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap
- LTTng
- gprof
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap
- LTTng
- gprof
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng
- gprof
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng
- gprof
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile—Not going anywhere
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile—Not going anywhere
- uprobes
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile—Not going anywhere
- uprobes—Not mainline yet, likely soon. . .
- kprobes
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile—Not going anywhere
- uprobes—Not mainline yet, likely soon. . .
- kprobes—Mainline, Used by others
- tracepoints
- ftrace
- perf events

Interesting Technologies for the Future

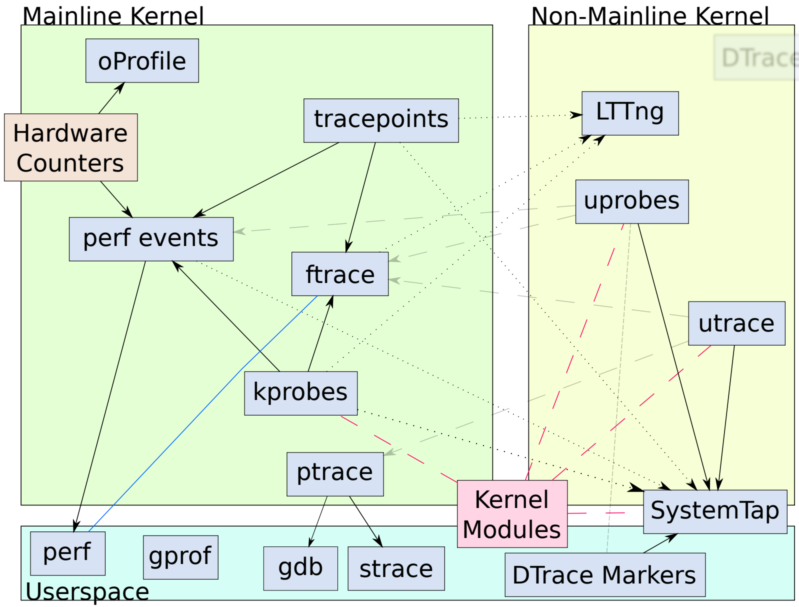
- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile—Not going anywhere
- uprobes—Not mainline yet, likely soon…
- kprobes—Mainline, Used by others
- tracepoints—Mainline, Used by others
- ftrace
- perf events

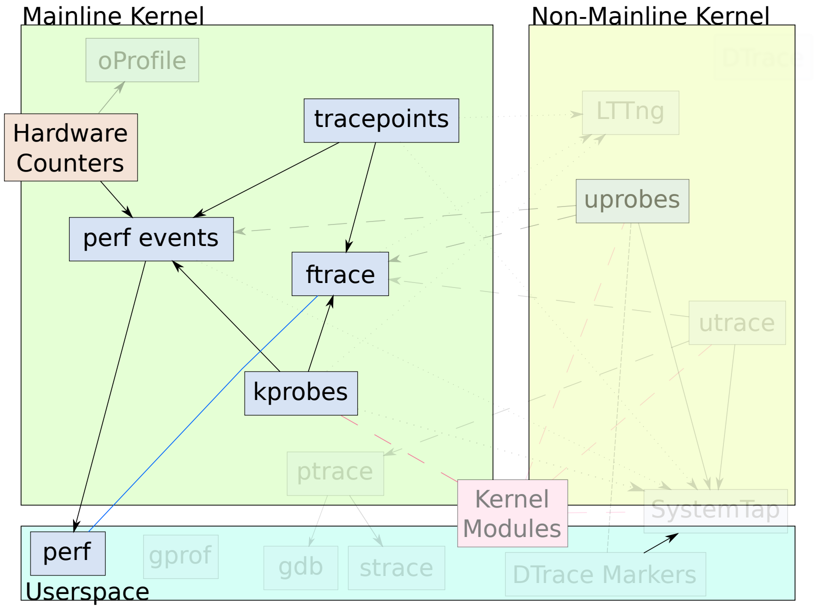
Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile—Not going anywhere
- uprobes—Not mainline yet, likely soon. . .
- kprobes—Mainline, Used by others
- tracepoints—Mainline, Used by others
- ftrace—Mainline, Active
- perf events

Interesting Technologies for the Future

- DTrace—Licence issues, not suited to Linux
- SystemTap—Not mainline, enterprise focus
- LTTng—Not mainline, enterprise focus
- gprof—Limitations, not going anywhere
- ptrace—Ugly, set in stone
- utrace—Trouble going mainline
- oProfile—Not going anywhere
- uprobes—Not mainline yet, likely soon. . .
- kprobes—Mainline, Used by others
- tracepoints—Mainline, Used by others
- ftrace—Mainline, Active
- perf events—Mainline, Very active


Mainline Kernel
Non-Mainline Kernel
DTrace
oProfile
tracepoints
LTTng
Hardware Counters
uprobes
perf events
ftrace
utrace
kprobes
ptrace
Kernel Modules
SystemTap
perf
gprof
gdb
strace
DTrace Markers
Userspace


Mainline Kernel
Non-Mainline Kernel
oProfile
DTrace
Hardware Counters
tracepoints
LTTng
uprobes
perf events
ftrace
utrace
kprobes
ptrace
Kernel Modules
SystemTap
perf
gprof
gdb
strace
DTrace Markers
Userspace

Outline

2 Demonstrations
- Finding cache misses with perf
- Locate sources of block I/O with tracepoints
- Using kprobes to analyse a running kernel

Finding cache misses with perf

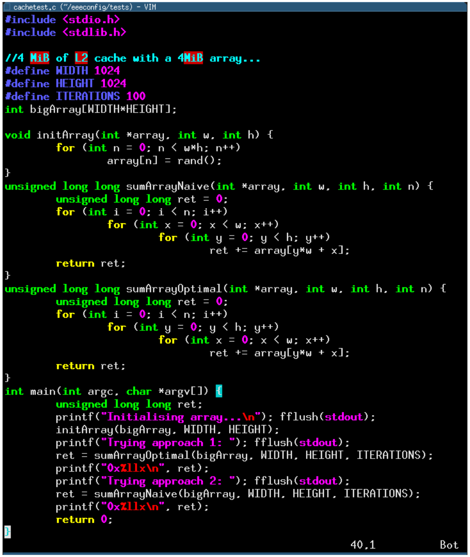
Finding cache misses with perf

- Fairly common scenario
- High cache misses is bad for performance
- Use a very simple C program as an example
- Task: sum every element in a large array
- Complication: array & program too big for L2 cache
- Two approaches with different array access order
- Use perf to observe cache misses

Finding cache misses with perf

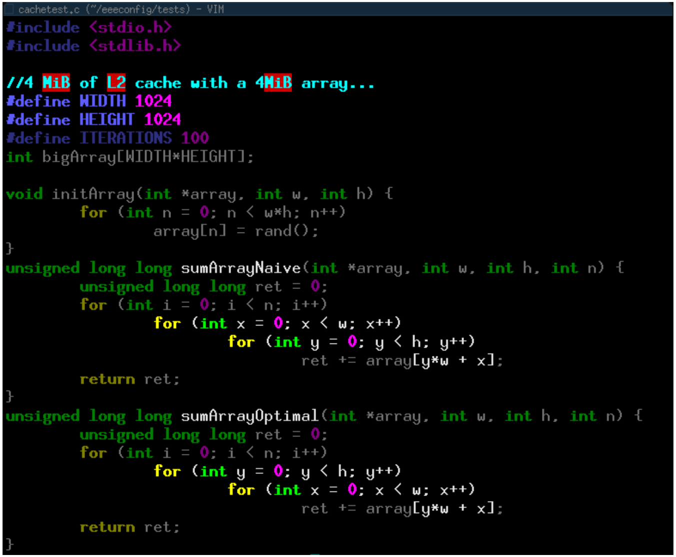
- Fairly common scenario
- High cache misses is bad for performance
- Use a very simple C program as an example
- Task: sum every element in a large array
- Complication: array & program too big for L2 cache
- Two approaches with different array access order
- Use perf to observe cache misses

Finding cache misses with perf



```
#include <stdio.h>
#include <stdlib.h>

//4 MiB of L2 cache with a 4MiB array...
#define WIDTH 1024
#define HEIGHT 1024
#define ITERATIONS 100
int bigArray[WIDTH*HEIGHT];

void initArray(int *array, int w, int h) {
        for (int n = 0; n < w*h; n++)
                array[n] = rand();
}
unsigned long long sumArrayNaive(int *array, int w, int h, int n) {
        unsigned long long ret = 0;
        for (int i = 0; i < n; i++)
                for (int x = 0; x < w; x++)
                        for (int y = 0; y < h; y++)
                                ret += array[y*w + x];
        return ret;
}
unsigned long long sumArrayOptimal(int *array, int w, int h, int n) {
        unsigned long long ret = 0;
        for (int i = 0; i < n; i++)
                for (int y = 0; y < h; y++)
                        for (int x = 0; x < w; x++)
                                ret += array[y*w + x];
        return ret;
}
int main(int argc, char *argv[]) {
        unsigned long long ret;
        printf("Initialising array...\n"); fflush(stdout);
        initArray(bigArray, WIDTH, HEIGHT);
        printf("Trying approach 1: "); fflush(stdout);
        ret = sumArrayOptimal(bigArray, WIDTH, HEIGHT, ITERATIONS);
        printf("0x%llx\n", ret);
        printf("Trying approach 2: "); fflush(stdout);
        ret = sumArrayNaive(bigArray, WIDTH, HEIGHT, ITERATIONS);
        printf("0x%llx\n", ret);
        return 0;
}
                                                      40,1          Bot
```

Finding cache misses with perf



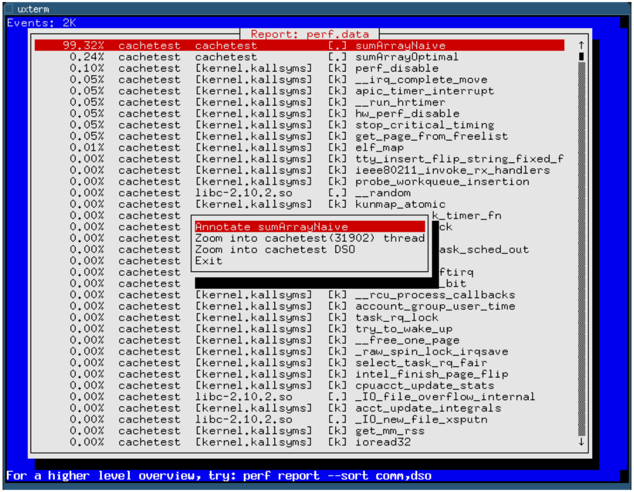
cachetest.c (~/eeeconfig/tests) - VIM

```c
#include <stdio.h>
#include <stdlib.h>

//4 MiB of L2 cache with a 4MiB array...
#define WIDTH 1024
#define HEIGHT 1024
#define ITERATIONS 100
int bigArray[WIDTH*HEIGHT];

void initArray(int *array, int w, int h) {
        for (int n = 0; n < w*h; n++)
                array[n] = rand();
}
unsigned long long sumArrayNaive(int *array, int w, int h, int n) {
        unsigned long long ret = 0;
        for (int i = 0; i < n; i++)
                for (int x = 0; x < w; x++)
                        for (int y = 0; y < h; y++)
                                ret += array[y*w + x];
        return ret;
}
unsigned long long sumArrayOptimal(int *array, int w, int h, int n) {
        unsigned long long ret = 0;
        for (int i = 0; i < n; i++)
                for (int y = 0; y < h; y++)
                        for (int x = 0; x < w; x++)
                                ret += array[y*w + x];
        return ret;
}
```

Finding cache misses with perf

$ perf record -e cache-misses ./cachetest

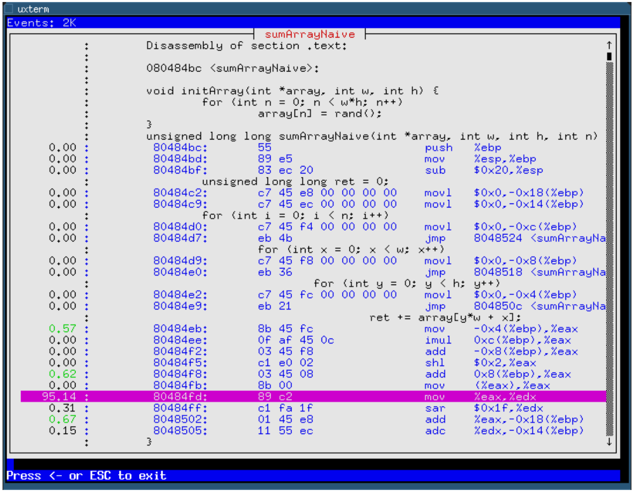
$ perf report



```
uxterm
Events: 2K
                              Report: perf.data
   99.32%  cachetest  cachetest          [.] sumArrayNaive
    0.24%  cachetest  cachetest          [.] sumArrayOptimal
    0.10%  cachetest  [kernel.kallsyms]  [k] perf_disable
    0.05%  cachetest  [kernel.kallsyms]  [k] __irq_complete_move
    0.05%  cachetest  [kernel.kallsyms]  [k] apic_timer_interrupt
    0.05%  cachetest  [kernel.kallsyms]  [k] __run_hrtimer
    0.05%  cachetest  [kernel.kallsyms]  [k] hw_perf_disable
    0.05%  cachetest  [kernel.kallsyms]  [k] stop_critical_timing
    0.05%  cachetest  [kernel.kallsyms]  [k] get_page_from_freelist
    0.01%  cachetest  [kernel.kallsyms]  [k] elf_map
    0.00%  cachetest  [kernel.kallsyms]  [k] tty_insert_flip_string_fixed_f
    0.00%  cachetest  [kernel.kallsyms]  [k] ieee80211_invoke_rx_handlers
    0.00%  cachetest  [kernel.kallsyms]  [k] probe_workqueue_insertion
    0.00%  cachetest  libc-2.10.2.so     [.] __random
    0.00%  cachetest  [kernel.kallsyms]  [k] kunmap_atomic
    0.00%  cachetest                            k_timer_fn
    0.00%  cachetest  Annotate sumArrayNaive     ck
    0.00%  cachetest  Zoom into cachetest(31902) thread
    0.00%  cachetest  Zoom into cachetest DSO    ask_sched_out
    0.00%  cachetest  Exit
    0.00%  cachetest                             ftirq
    0.00%  cachetest                             _bit
    0.00%  cachetest  [kernel.kallsyms]  [k] __rcu_process_callbacks
    0.00%  cachetest  [kernel.kallsyms]  [k] account_group_user_time
    0.00%  cachetest  [kernel.kallsyms]  [k] task_rq_lock
    0.00%  cachetest  [kernel.kallsyms]  [k] try_to_wake_up
    0.00%  cachetest  [kernel.kallsyms]  [k] __free_one_page
    0.00%  cachetest  [kernel.kallsyms]  [k] _raw_spin_lock_irqsave
    0.00%  cachetest  [kernel.kallsyms]  [k] select_task_rq_fair
    0.00%  cachetest  [kernel.kallsyms]  [k] intel_finish_page_flip
    0.00%  cachetest  [kernel.kallsyms]  [k] cpuacct_update_stats
    0.00%  cachetest  libc-2.10.2.so     [.] _IO_file_overflow_internal
    0.00%  cachetest  [kernel.kallsyms]  [k] acct_update_integrals
    0.00%  cachetest  libc-2.10.2.so     [.] _IO_new_file_xsputn
    0.00%  cachetest  [kernel.kallsyms]  [k] get_mm_rss
    0.00%  cachetest  [kernel.kallsyms]  [k] ioread32
For a higher level overview, try: perf report --sort comm,dso
```

Finding cache misses with perf

```
$ perf record -e cache-misses ./cachetest
$ perf report
```



```
uxterm
Events: 2K
                                  sumArrayNaive
         :         Disassembly of section .text:
         :
         :         080484bc <sumArrayNaive>:
         :
         :         void initArray(int *array, int w, int h) {
         :                 for (int n = 0; n < w*h; n++)
         :                         array[n] = rand();
         :         }
         :         unsigned long long sumArrayNaive(int *array, int w, int h, int n)
    0.00 :          80484bc:       55                      push   %ebp
    0.00 :          80484bd:       89 e5                   mov    %esp,%ebp
    0.00 :          80484bf:       83 ec 20                sub    $0x20,%esp
         :                 unsigned long long ret = 0;
    0.00 :          80484c2:       c7 45 e8 00 00 00 00    movl   $0x0,-0x18(%ebp)
    0.00 :          80484c9:       c7 45 ec 00 00 00 00    movl   $0x0,-0x14(%ebp)
         :                 for (int i = 0; i < n; i++)
    0.00 :          80484d0:       c7 45 f4 00 00 00 00    movl   $0x0,-0xc(%ebp)
    0.00 :          80484d7:       eb 4b                   jmp    8048524 <sumArrayNa
         :                         for (int x = 0; x < w; x++)
    0.00 :          80484d9:       c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%ebp)
    0.00 :          80484e0:       eb 36                   jmp    8048518 <sumArrayNa
         :                                 for (int y = 0; y < h; y++)
    0.00 :          80484e2:       c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%ebp)
    0.00 :          80484e9:       eb 21                   jmp    804850c <sumArrayNa
         :                                         ret += array[y*w + x];
    0.57 :          80484eb:       8b 45 fc                mov    -0x4(%ebp),%eax
    0.00 :          80484ee:       0f af 45 0c             imul   0xc(%ebp),%eax
    0.00 :          80484f2:       03 45 f8                add    -0x8(%ebp),%eax
    0.00 :          80484f5:       c1 e0 02                shl    $0x2,%eax
    0.62 :          80484f8:       03 45 08                add    0x8(%ebp),%eax
    0.00 :          80484fb:       8b 00                   mov    (%eax),%eax
   95.14 :          80484fd:       89 c2                   mov    %eax,%edx
    0.31 :          80484ff:       c1 fa 1f                sar    $0x1f,%edx
    0.67 :          8048502:       01 45 e8                add    %eax,-0x18(%ebp)
    0.15 :          8048505:       11 55 ec                adc    %edx,-0x14(%ebp)
         :         }
Press <- or ESC to exit
```

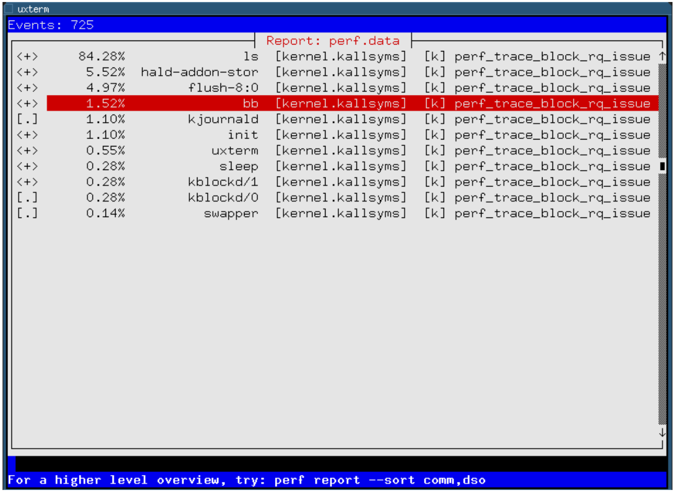
Locate sources of block I/O with perf and tracepoints

perf—Locate sources of block I/O

- System is under heavy disk I/O causing slowdowns
- Traditional top does not reveal offending processes
- Even atop only shows I/O at process granularity
- block:block_rq_issue tracepoint available
- Kernel must be compiled with block I/O tracing
- perf can produce call graphs to track exact origin

perf—Locate sources of block I/O

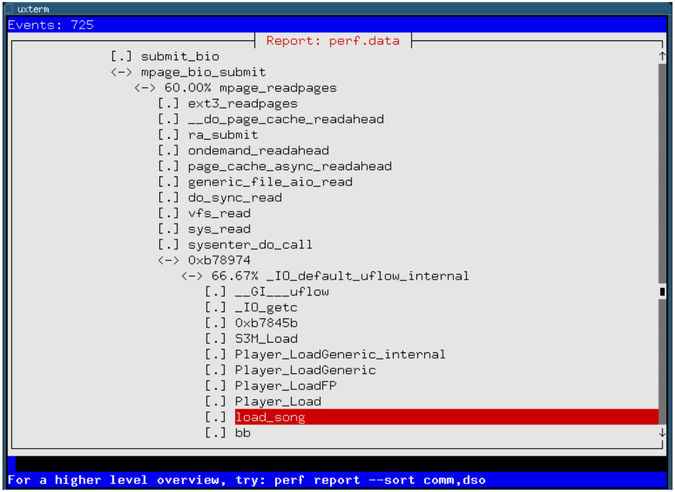
```
# perf record -g -a -e block:block_rq_issue sleep 10
# perf report
```



```
uxterm
Events: 725
                                   Report: perf.data
<+>     84.28%                ls  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
<+>      5.52%  hald-addon-stor  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
<+>      4.97%       flush-8:0  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
<+>      1.52%              bb  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
[.]      1.10%        kjournald  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
<+>      1.10%             init  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
<+>      0.55%           uxterm  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
<+>      0.28%            sleep  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
<+>      0.28%        kblockd/1  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
[.]      0.28%        kblockd/0  [kernel.kallsyms]  [k] perf_trace_block_rq_issue
[.]      0.14%          swapper  [kernel.kallsyms]  [k] perf_trace_block_rq_issue

For a higher level overview, try: perf report --sort comm,dso
```

perf—Locate sources of block I/O

```
# perf record -g -a -e block:block_rq_issue sleep 10
# perf report
```



```
uxterm
Events: 725
                              Report: perf.data
            [.] submit_bio
            <-> mpage_bio_submit
               <-> 60.00% mpage_readpages
                  [.] ext3_readpages
                  [.] __do_page_cache_readahead
                  [.] ra_submit
                  [.] ondemand_readahead
                  [.] page_cache_async_readahead
                  [.] generic_file_aio_read
                  [.] do_sync_read
                  [.] vfs_read
                  [.] sys_read
                  [.] sysenter_do_call
                  <-> 0xb78974
                     <-> 66.67% _IO_default_uflow_internal
                        [.] __GI___uflow
                        [.] _IO_getc
                        [.] 0xb7845b
                        [.] S3M_Load
                        [.] Player_LoadGeneric_internal
                        [.] Player_LoadGeneric
                        [.] Player_LoadFP
                        [.] Player_Load
                        [.] load_song
                        [.] bb

For a higher level overview, try: perf report --sort comm,dso
```

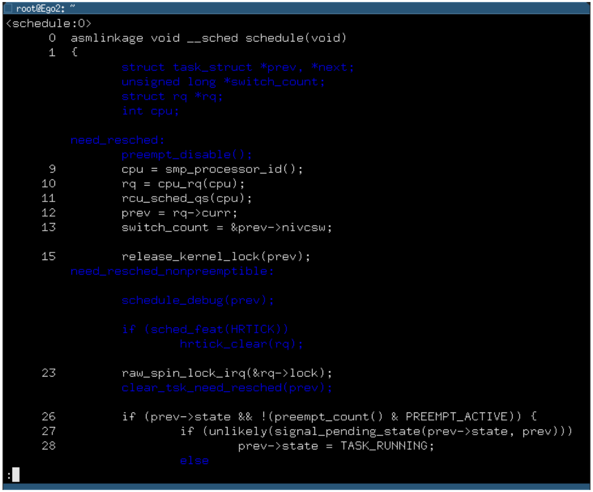
Analyse a running kernel with perf and kprobes

Analyse a running kernel with perf and kprobes

- Want to pull information out of a running kernel
- Information is not already revealed to userspace
- No tracepoints provide the information
- Live system—cannot afford downtime
- Kernel debugging information and source is available
- Let's probe the scheduler as an example. . .

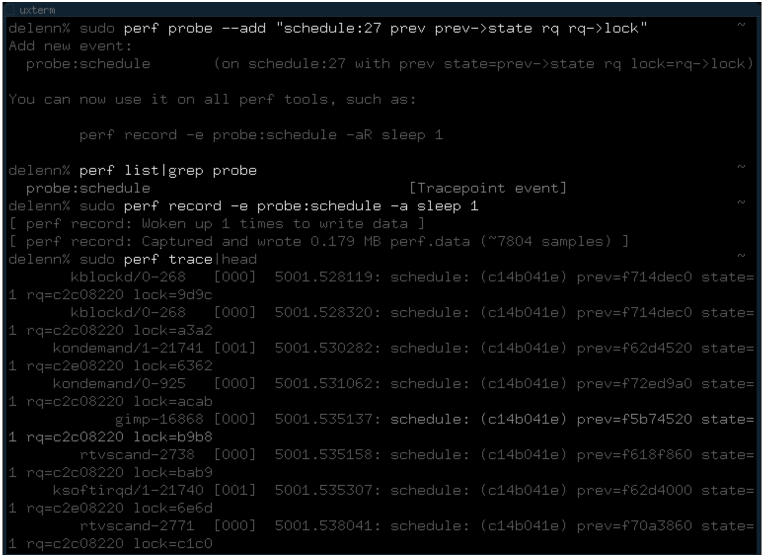
Analysing a running kernel with perf and kprobes

# perf probe –line schedule



```
root@Ego2: ~
<schedule:0>
      0  asmlinkage void __sched schedule(void)
      1  {
                struct task_struct *prev, *next;
                unsigned long *switch_count;
                struct rq *rq;
                int cpu;

         need_resched:
                preempt_disable();
      9         cpu = smp_processor_id();
     10         rq = cpu_rq(cpu);
     11         rcu_sched_qs(cpu);
     12         prev = rq->curr;
     13         switch_count = &prev->nivcsw;

     15         release_kernel_lock(prev);
         need_resched_nonpreemptible:

                schedule_debug(prev);

                if (sched_feat(HRTICK))
                        hrtick_clear(rq);

     23         raw_spin_lock_irq(&rq->lock);
                clear_tsk_need_resched(prev);

     26         if (prev->state && !(preempt_count() & PREEMPT_ACTIVE)) {
     27                 if (unlikely(signal_pending_state(prev->state, prev)))
     28                         prev->state = TASK_RUNNING;
                        else
:
```

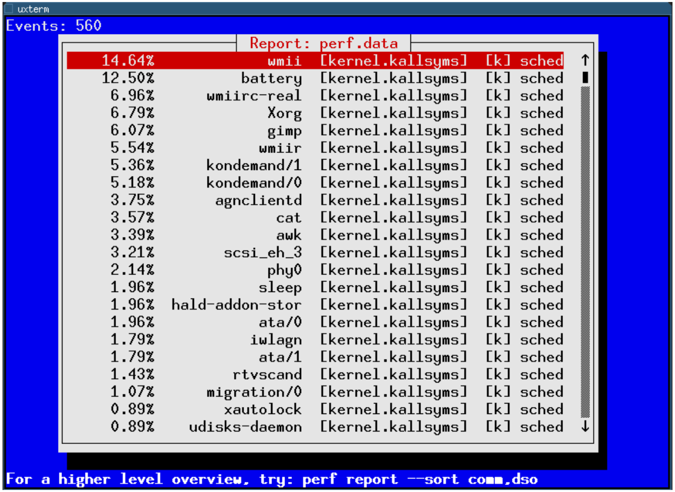
Analysing a running kernel with perf and kprobes



```
uxterm
delenn% sudo perf probe --add "schedule:27 prev prev->state rq rq->lock"
Add new event:
  probe:schedule       (on schedule:27 with prev state=prev->state rq lock=rq->lock)

You can now use it on all perf tools, such as:

        perf record -e probe:schedule -aR sleep 1

delenn% perf list|grep probe
  probe:schedule                                [Tracepoint event]
delenn% sudo perf record -e probe:schedule -a sleep 1
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.179 MB perf.data (~7804 samples) ]
delenn% sudo perf trace|head
       kblockd/0-268    [000]  5001.528119: schedule: (c14b041e) prev=f714dec0 state=
1 rq=c2c08220 lock=9d9c
       kblockd/0-268    [000]  5001.528320: schedule: (c14b041e) prev=f714dec0 state=
1 rq=c2c08220 lock=a3a2
     kondemand/1-21741 [001]  5001.530282: schedule: (c14b041e) prev=f62d4520 state=
1 rq=c2e08220 lock=6362
     kondemand/0-925    [000]  5001.531062: schedule: (c14b041e) prev=f72ed9a0 state=
1 rq=c2c08220 lock=acab
            gimp-16868 [000]  5001.535137: schedule: (c14b041e) prev=f5b74520 state=
1 rq=c2c08220 lock=b9b8
        rtvscand-2738   [000]  5001.535158: schedule: (c14b041e) prev=f618f860 state=
1 rq=c2c08220 lock=bab9
     ksoftirqd/1-21740 [001]  5001.535307: schedule: (c14b041e) prev=f62d4000 state=
1 rq=c2e08220 lock=6e6d
        rtvscand-2771   [000]  5001.538041: schedule: (c14b041e) prev=f70a3860 state=
1 rq=c2c08220 lock=c1c0
```

Analysing a running kernel with perf and kprobes

# perf report



```
Events: 560
                         Report: perf.data
   14.64%            wmii  [kernel.kallsyms]  [k] sched
   12.50%         battery  [kernel.kallsyms]  [k] sched
    6.96%     wmiirc-real  [kernel.kallsyms]  [k] sched
    6.79%            Xorg  [kernel.kallsyms]  [k] sched
    6.07%            gimp  [kernel.kallsyms]  [k] sched
    5.54%           wmiir  [kernel.kallsyms]  [k] sched
    5.36%     kondemand/1  [kernel.kallsyms]  [k] sched
    5.18%     kondemand/0  [kernel.kallsyms]  [k] sched
    3.75%      agnclientd  [kernel.kallsyms]  [k] sched
    3.57%             cat  [kernel.kallsyms]  [k] sched
    3.39%             awk  [kernel.kallsyms]  [k] sched
    3.21%       scsi_eh_3  [kernel.kallsyms]  [k] sched
    2.14%            phy0  [kernel.kallsyms]  [k] sched
    1.96%           sleep  [kernel.kallsyms]  [k] sched
    1.96% hald-addon-stor  [kernel.kallsyms]  [k] sched
    1.96%           ata/0  [kernel.kallsyms]  [k] sched
    1.79%          iwlagn  [kernel.kallsyms]  [k] sched
    1.79%           ata/1  [kernel.kallsyms]  [k] sched
    1.43%        rtvscand  [kernel.kallsyms]  [k] sched
    1.07%     migration/0  [kernel.kallsyms]  [k] sched
    0.89%       xautolock  [kernel.kallsyms]  [k] sched
    0.89%   udisks-daemon  [kernel.kallsyms]  [k] sched

For a higher level overview, try: perf report --sort comm,dso
```

Thankyou all for listening

- perf can be found in the Linux kernel sources under /tools/perf
- perf is being actively developed in the “tip” kernel tree
- Linux Weekly News has some good further reading on ftrace—google *site:lwn.net ftrace*
- perf articles not as easy to find yet, here are some:
  - http://lwn.net/Articles/339361—“Perfcounters added to the mainline”
  - http://lwn.net/Articles/373842—“Scripting support for perf”
  - http://lwn.net/Articles/382554—“A “live mode” for perf”
  - http://lwn.net/Articles/346470—“Fun with tracepoints”
  - http://lkml.org/lkml/2010/3/17/91—Ingo Molnar on database I/O latency