

Performance Counters on Linux

The New Tools

Linux Plumbers Conference, September, 2009

Arnaldo Carvalho de Melo
acme@redhat.com

How did I get involved?

- . I am no specialist on performance counters
- . pahole & the dwarves
- . ELF, DWARF, symtabs, dynsyms, relocations, etc
- . ftrace
- . Presentation focus on the tools
- . But some kernel details will be mentioned

pahole - naked struct

```
struct cheese {  
    char name[27];  
    short age;  
    int  calories;  
    short price;  
    char type;  
};
```

pahole - data structure layout

```
$ pahole --quiet swiss_cheese
```

```
struct cheese {
```

```
    char    name[27]; /* 0 27 */
```

```
    short int age; /* 28 2 */
```

```
    int     calories; /* 32 4 */
```

```
    short int price; /* 36 2 */
```

```
    char    type; /* 38 1 */
```

```
};
```

pahole - the holes!

```
$ pahole swiss_cheese
struct cheese {
    char          name[27]; /* 0 27 */
    /* XXX 1 byte hole, try to pack */
    short int     age;      /* 28 2 */
    /* XXX 2 bytes hole, try to pack */
    int          calories; /* 32 4 */
    short int     price;    /* 36 2 */
    char          type;     /* 38 1 */
    /* size: 40, cachelines: 1 */
    /* sum members: 36, holes: 2, sum holes: 3 */
    /* padding: 1, last cacheline: 40 bytes */
};
```

How did I get involved? Part two

- . Part of the Red Hat Real Time team
- . We need to discover why 100us deadlines are not being met
- . Why this is slower on your RT kernel than on your RHEL one?
- . Observability tooling!
- . Huge educational value, use it!

OK, Back to the future!

Where is my bottleneck?!

later, at the pub...

Performance Counters

Performance counters are special hardware registers available on most modern CPUs. These registers count the number of certain types of hw events: such as instructions executed, cache-misses suffered, or branches mispredicted - without slowing down the kernel or applications. These registers can also trigger interrupts when a threshold number of events have passed - and can thus be used to profile the code that runs on that CPU.

From <http://perf.wiki.kernel.org/>

Limited resource:

Processor:

UltraSparc	2
Pentium III	2
Athlon	4
IA-64	4
POWER4	8
Pentium IV	18

Some are programmable, some are for specific events.

The oprofile development problem

- . Disconnected kernel & userspace development
- . Linus problem with Atom and Nehalem support
- . Less of the "two broken pieces" approach -> One working piece
- . <http://lwn.net/Articles/339406/>

How LPE started?

- . perfmon/perfmon2/perfmon3
- . Dozens of new syscalls || multiplexer
- . Userspace knows about PMU details
- . Kernel doesn't abstracts those details
- . Endless discussions, low buyin from core kernel developers
- . Ingo Molnar + Thomas Gleixner presents how they think it should be done

The perf user interface approach

- . git like
- . Many subcommands
- . Per thread/per workload/per CPU/system wide
- . No daemons

The perf development approach

- . Tools hosted in the kernel sources: tools/perf/
- . Subcommands Can be developed largely independently
- . Developers expected to touch both sides (kernel/user)
- . Written in the C idiom used in the kernel
- . Shares code with the kernel (rbtree, list, more to come)

The new implementation approach

- . Just one new syscall: `sys_perf_counter_open`
- . Returns a file descriptor
- . `read/write/mmap/close/fcntl/ioctl/poll` work as usual
- . Per thread/cpu/whole system
- . Transparent inheritance support
 - Full workloads can be measured
 - Without having to use `ptrace` methods to follow forks & clones
- . Events Mapped to closest per arch hw counter
- . Possible to use raw events
- . Supports tracepoints
- . Software counters (hrtimer based or not)
- . Support for hw breakpoints being developed

sys_perf_counter_open - The syscall

- event type attributes for monitoring/sampling
- target pid
- target cpu
- group_fd
- flags

sys_perf_counter_open - event type

- PERF_TYPE_HARDWARE
- PERF_TYPE_SOFTWARE
- PERF_TYPE_TRACEPOINT
- PERF_TYPE_HW_CACHE
- PERF_TYPE_RAW (for raw tracepoint data)

sys_perf_counter_open - attr.sample_type

- bitmask
- PERF_SAMPLE_IP
- PERF_SAMPLE_TID
- PERF_SAMPLE_TIME
- PERF_SAMPLE_CALLCHAIN
- PERF_SAMPLE_ID
- PERF_SAMPLE_CPU

sys_perf_counter_open - attr config bitfield

- disabled: off by default
- inherit: children inherit it
- exclude_{user,kernel,hv,idle}: don't count these
- mmap: include mmap data
- comm: include comm data
- inherit_stat: per task counts
- enable_on_exec: next exec enables

Architectures already supported

- . x86: p6, core+, k7+
- . ppc64
- . sparc
- . Others supporting just software/ftrace events
 - frv
 - parisc
 - s390
 - sh

Tools

- . git like: subcomands
- . list
- . stat
- . top
- . record
- . report
- . annotate
- . trace
- . sched

perf list

\$ perf list

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]

perf list - continued

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]

perf list - continued

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]
L1-icache-prefetches	[Hardware cache event]
L1-icache-prefetch-misses	[Hardware cache event]

perf list - continued

LLC-loads	[Hardware cache event]
LLC-load-misses	[Hardware cache event]
LLC-stores	[Hardware cache event]
LLC-store-misses	[Hardware cache event]
LLC-prefetches	[Hardware cache event]
LLC-prefetch-misses	[Hardware cache event]

perf list - continued

dTLB-loads	[Hardware cache event]
dTLB-load-misses	[Hardware cache event]
dTLB-stores	[Hardware cache event]
dTLB-store-misses	[Hardware cache event]
dTLB-prefetches	[Hardware cache event]
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	[raw hardware event descriptor]

perf list - example of tracepoints

block:block_rq_insert	[Tracepoint event]
jbd2:jbd2_start_commit	[Tracepoint event]
ext4:ext4_allocate_inode	[Tracepoint event]
kmem:kmalloc	[Tracepoint event]
module:module_load	[Tracepoint event]
workqueue:workqueue_execution	[Tracepoint event]
timer:timer_expire_{entry,exit}	[Tracepoint event]
timer:hrtimer_start	[Tracepoint event]
irq:irq_handler_{entry,exit}	[Tracepoint event]
irq:softirq_{entry,exit}	[Tracepoint event]
sched:sched_{wakeup,switch}	[Tracepoint event]
syscalls:sys_{enter,exit}_epoll_wait	[Tracepoint event]

perf list - plumbers wanted!

- . Too many events, allow filtering
- . Ex.: `--tracepoint sched:wakeup*`

perf stat

\$ perf stat ls Makefile
Makefile

Performance counter stats for 'ls Makefile':

2.204554	task-clock-msecs	#	0.842 CPUs
0	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
240	page-faults	#	0.109 M/sec
2176584	cycles	#	987.313 M/sec
1224357	instructions	#	0.563 IPC
60577	cache-references	#	27.478 M/sec
1788	cache-misses	#	0.811 M/sec

0.002618700 seconds time elapsed

\$

perf stat - statistic

\$ perf stat -r 5 sleep 5

Performance counter stats for 'sleep 5' (5 runs):

1.411021 task-clock-msecs	#	0.000 CPUs	(+- 0.829%)
1 context-switches	#	0.001 M/sec	(+- 0.000%)
0 CPU-migrations	#	0.000 M/sec	(+- nan%)
176 page-faults	#	0.125 M/sec	(+- 0.000%)
1378625 cycles	#	977.041 M/sec	(+- 0.796%)
752343 instructions	#	0.546 IPC	(+- 0.362%)
30534 cache-references	#	21.639 M/sec	(+- 0.763%)
2074 cache-misses	#	1.470 M/sec	(+- 4.879%)

5.001883846 seconds time elapsed (+- 0.000%)

\$

perf top - loading firefox

PerfTop: 10068 irqs/sec kernel:34.0% [100000 cycles], (all, 2 CPUs)

samples	pcnt	kernel function
3125.00	- 15.3%	: read_hpet
2230.00	- 10.9%	: mwait_idle_with_hints
780.00	- 3.8%	: drm_clflush_pages [drm]
698.00	- 3.4%	: thinkpad_acpi_module_init [thinkpad_acpi]
665.00	- 3.3%	: i915_gem_detach_phys_object [i915]
475.00	- 2.3%	: _spin_lock_irqsave
434.00	- 2.1%	: copy_user_generic_string
433.00	- 2.1%	: acpi_os_read_port
404.00	- 2.0%	: clear_page_c
350.00	- 1.7%	: _spin_lock
338.00	- 1.7%	: hpet_next_event
293.00	- 1.4%	: i915_gem_attach_phys_object [i915]
267.00	- 1.3%	: lock_kernel
257.00	- 1.3%	: page_fault
236.00	- 1.2%	: i8042_interrupt

^C
\$

perf top - plumbers wanted!

- . Show userspace symbols too
- . --pid N
- . callchains

perf record

- . No daemons
- . Callchains
- . Output to different files
- . Feed to other tools
- . Outputs just into the regular filesystem
- . No separate 'oprofile repository' of sample files
- . Files are next to the project you are working on
- . Can record events on a task, on a CPU or on the whole system

perf record example

```
$ cd firefox.data
```

```
$ perf record --pid $(pidof firefox)
```

```
^C[ perf record: Captured and wrote 1.215 MB perf.data (~53065 samples) ]
```

```
$ ls -sh perf.data
```

```
1,3M perf.data
```

perf record - plumbers wanted!

- . Should get the symtabs at the end of the session, include in perf.data
- . To allow for analysis on a different machine
- . Would be good to have -symtab packages, not just -debuginfo
- . Perhaps use debuginfofs
- . <http://fedoraproject.org/wiki/Features/DebuginfoFS>

perf report

- . Lazy/Late symbol resolution
- . Picks what is available
- . -debuginfo packages, .symtab, .dynsym
- . --fractal, --graph
- . Supports JATO generated symbol tables for JAVA JIT profiling
- . Automatically pick them from the dso name

perf report example

```
$ perf report -C firefox --sort comm,dso
# Samples: 52628
# Overhead   Shared Object
# .....
36.37% /usr/lib64/xulrunner-1.9.1/libxul.so
30.29% /usr/lib64/xulrunner-1.9.1/libmozjs.so
19.39% [kernel]
 3.69% /usr/lib64/firefox-3.5/firefox
 2.48% /lib64/libpthread-2.10.1.so
 1.78% /lib64/libnspr4.so
 0.98% /usr/lib64/libjpeg.so.62.0.0
 0.87% /lib64/libglib-2.0.so.0.2000.3
 0.68% /lib64/libc-2.10.1.so
 0.55% /usr/lib64/libsqlite3.so.0.8.6
$
```

\$ perf report example 2

```
$ perf report
# Samples: 52628
# Overhead          Shared Object  Symbol
# .....
13.17% [kernel]          vread_hpet
7.51% /lib64/xulrunner/libxul.so SelectorMatches(RuleProcessorData&, nsCSSSelecto
5.82% /lib64/xulrunner/libmozjs.so js_Interpret
2.90% /lib64/firefox-3.5/firefox 0x0000000000dd26
1.68% /lib64/xulrunner/libxul.so SelectorMatchesTree(RuleProcessorData&, nsCSSSel
1.50% /lib64/xulrunner/libmozjs.so js_Invoke
1.46% /lib64/xulrunner/libmozjs.so js_InternalInvoke
1.42% /lib64/xulrunner/libmozjs.so js_LookupPropertyWithFlags
1.31% /lib64/xulrunner/libxul.so nsAttrValue::Contains(nsIAtom*, nsCaseTreatment)
1.27% /lib64/libpthread-2.10.1.so __pthread_mutex_lock_internal
1.22% /lib64/xulrunner/libmozjs.so js_GetPropertyHelper
1.12% /lib64/xulrunner/libmozjs.so js_ExecuteRegExp
1.10% /lib64/xulrunner/libmozjs.so js_SearchScope
```

\$

perf report -g

- Callchains
- Needs -fno-omit-frame-pointer
- Register pressure on IA32
- Study ongoing to enable it on upcoming distros at least on x86_64

perf report -g

```
# Samples: 216342
# Overhead Command          Shared Object Symbol
# .....
15.82% pahole /usr/lib64/libdw-0.141.so  [.] __libdw_find_attr
    |--1.85%-- __libdw_findabbrev
    |--1.78%-- __die__process_tag
    |         cus__load_module
    |         cus__process_dwflmod
    |         __dwfl_getmodules_internal
    |--1.25%-- Dwarf_Abbrev_Hash_find
    |--1.14%-- die__process_function
    |         |--63.33%-- die__create_new_lexblock
    |         |         |--57.89%-- die__process_function
    |         |         |         |--63.64%-- __die__process_tag
    |         |         |         |         cus__load_module
    |         |         |         |         cus__process_dwflmod
    |         |         |         |         __dwfl_getmodules_internal
    |         |         |         |--36.36%-- die__create_new_lexblock
```

<SNIP>

perf report - plumbers wanted!

- . Really big files take long to load
- . Progressive loading, kinda similar to perf top
- . Snapshots updated to the screen every N seconds

perf annotate

- . similar purpose as opannotate
- . colors for hot lines
- . still uses objdump
- . need to make objdump -S use the source in -debuginfo pkgs

Another perf report example

```
$ perf record -g pahole vmlinux > /tmp/vmlinux.pahole
[ perf record: Captured and wrote 13.408 MB perf.data (~585799 samples) ]
$ perf report -g none -C pahole -d libdwarves.so.1.0.0
# dso: ./build/libdwarves.so.1.0.0
# comm: pahole
# Samples: 39486
# Overhead Symbol
# .....
12.57% [.] tag__recode_dwarf_type
10.81% [.] namespace__recode_dwarf_types
10.49% [.] die__process_class
10.20% [.] cu__find_base_type_by_sname_and_size
6.15% [.] strings__compare
4.93% [.] tag__init
4.29% [.] cus__load_module
3.99% [.] list__for_all_tags
3.71% [.] tag__size
2.95% [.] __die__process_tag
2.38% [.] cu__table_add_tag
2.28% [.] class_member__cache_byte_size
1.87% [.] strings__add
1.86% [.] dwarf_attr@plt
1.75% [.] die__create_new_subroutine_type
```

What is happening in tag__recode_dwarf_type?

Percent | Source code & Disassembly of libdwarves.so.1.0.0

```
: Disassembly of section .text:
: 00000000000007ae0 <cu__table_add_tag>:
<SNIP>
: struct dwarf_tag *tpos;
: struct hlist_node *pos;
: uint16_t bucket = hashtags__fn(id);
: const struct hlist_head *head = hashtable + bucket;
:
: hlist_for_each_entry(tpos, pos, head, hash_node) {
27.26 : 11870: 48 89 d0      mov %rdx,%rax
:     if (tpos->id == id)
0.04 : 11873: 75 eb      jne 11860 <tag__recode_dwarf_type+0x4e0>
0.60 : 11875: e9 c7 fe ff jmpq 11741 <tag__recode_dwarf_type+0x3c1>
0.00 : 1187a: 66 0f 1f 44 00 00 nopw 0x0(%rax,%rax,1)
:
:     dtype = dwarf_cu__find_type_by_id(cu->priv, dtag->containing_type)
<SNIP>
```

perf trace

```
# perf record -f -R -e sched:sched_wakeup -e sched:sched_switch -e  
irq:softirq_entry -e irq:softirq_exit -C 0 -F 1 sleep 1
```

```
# perf trace
```

```
version = 0.5
```

```
[init]-0 [0] 5.284716148: sched_switch: task swapper:0 [140] (R) ==> perf:2700 [120]  
[init]-0 [0] 5.284036980: softirq_entry: softirq=1 action=TIMER  
[init]-0 [0] 5.284040256: softirq_entry: softirq=8 action=RCU  
[init]-0 [0] 5.289747091: sched_wakeup: task pulseaudio:2603 [120] success=1 [000]  
sleep-2700 [0] 5.285640822: sched_switch: task sleep:2700 [120] (S) ==> swapper:0 [140]  
[init]-0 [0] 5.289757695: sched_switch: task swapper:0 [140] (R) ==> pulseaudio:2603 [120]  
sleep-2700 [0] 5.285025872: softirq_entry: softirq=1 action=TIMER  
sleep-2700 [0] 5.285027555: softirq_exit: softirq=1 action=TIMER  
sleep-2700 [0] 5.285029358: softirq_exit: softirq=8 action=RCU  
[init]-0 [0] 5.286031283: softirq_exit: softirq=1 action=TIMER  
[init]-0 [0] 5.501252937: sched_switch: task swapper:0 [140] (R) ==> firefox:17259 [120]  
[init]-0 [0] 6.184297736: sched_switch: task swapper:0 [140] (R) ==> pulseaudio:2602 [120]
```

Integration with other tools

- . Systemtap

- In-kernel usage should be designed not just for systemtap
- Study previous design for perfmon usage by wcohen

- . Oprofile

- Keep userspace utilities as-is, use perf kernel bits
- Counter multiplexing added, first seen in perf land
- Reduce the feature gap, future merge

- . sysprof

- Converted to the perf events infrastructure

- . PAPI

- Has support in 3.7.0 version. More tests needed.

More plumbers needed!

- . perf cmp
- . More GUI tools
- . Wiki has a TODO list

Thanks'n'Links

- . Thanks to Ingo, Thomas, PeterZ, Rostedt, Frederic, Mike, Paul
- . And everybody else contributing and testing these new tools

- . `tools/perf/Documentation/examples.txt` (in the kernel tree)
- . http://perf.wiki.kernel.org/index.php/Main_Page
- . <git://git.kernel.org/pub/scm/linux/kernel/git/tip/linux-2.6-tip.git>
- . Performance Counters on Linux: v8: <http://lwn.net/Articles/336542>
- . This presentation: <http://vger.kernel.org/~acme/perf/>

Arnaldo Carvalho de Melo
acme@redhat.com