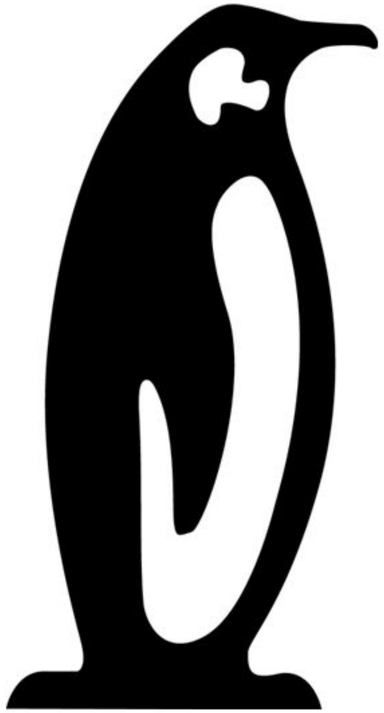


Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

Improving the Perf event subsystem after 15 years

Ian Rogers <irogers@google.com>

LINUX PLUMBERS CONFERENCE | Vienna, Austria
Sept. 18-20, 2024

https://en.m.wikipedia.org/wiki/File:Intel_Pentium_MMX_166_PGA_Front.jpg

Linux profiling 26 years ago…
or how am I qualified to talk on this subject

On being qualified, I'm probably not but I'll give it a go.. there are plenty of others in the room to correct me.

26 years ago I was building binary translators and embarking on a PhD. We were using gprof but the overheads were too high. Pentium MMX introduced performance counters and we could get access to global counters through tools like "rabbit".

Global meant whole system and so to run benchmarks we ran them at init runlevel 1.

Being a poor student this wasn't great as the benchmark machine was also the workstation, but we had access to other poor students. This led to John Levon's work on OProfile.

OProfile's initial mission, let's save and restore the performance counters on context switches. Focus is on user space profiling.

Later OProfile had sampling, kernel profiling and JIT support. It was a much loved tool.

Linux profiling 15 years ago…

OProfile stopped seeing as much active development. Ingo Molnar and many others started perf events:

https://lkml.org/lkml/2009/6/6/149 - Performance Counters for Linux, v8

Some of the advantages of perf events were:

- Deeper kernel integration - not a module
- Extensibility and support for things like tracepoints
- Performance
- Active development

Machines of the day had moved forward from Pentium Pros to Core 2 duos and Athlon IIs.

https://en.wikipedia.org/wiki/Athlon_II#/media/File:AMD_Athlon_II_X4_640_%22Propus%22(1).jpg

https://en.wikipedia.org/wiki/Intel_Core_2#/media/File:Core_2_Duo_E7500_2.93GHz.jpg

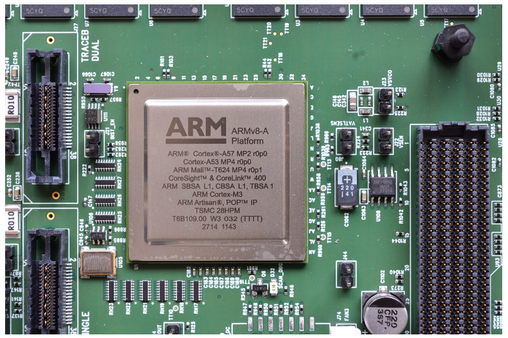
Linux profiling 8 years ago…

Perf events are the de facto profiling API.

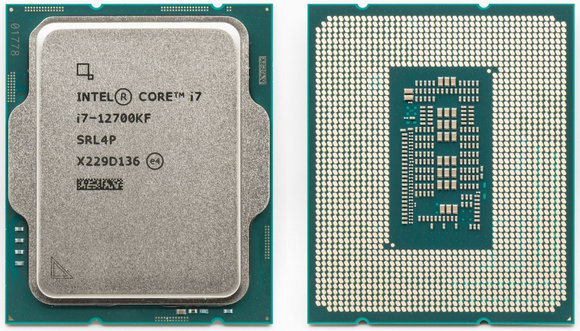
Intel i915 graphics try to use perf events: drivers/gpu/drm/i915/i915_perf.c

```
In the end we didn't see a clear benefit to making perf's implementation and
interface more complex by changing design assumptions while we knew we still
wouldn't be able to use any existing perf based userspace tools.
```

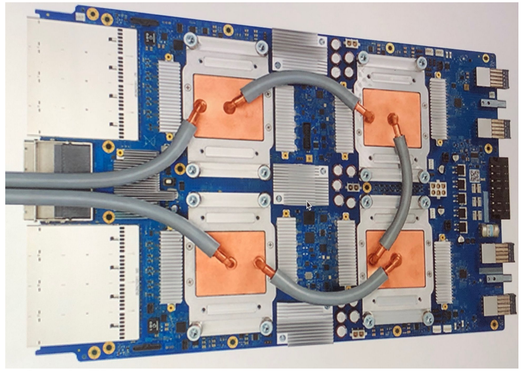
Increasingly we see the use of accelerators and heterogeneity.



big.LITTLE
https://en.wikipedia.org/wiki/ARM_big.LITTLE#/media/File:ARMCortexA57A53.jpg



Intel hybrid
https://en.wikipedia.org/wikihttps://en.wikipedia.org/wiki/Alder_Lake#/media/File:2023_Intel_Core_i7_12700KF_(5).jpg/Alder_Lake#/media/File:2023_Intel_Core_i7_12700KF_(5).jpg

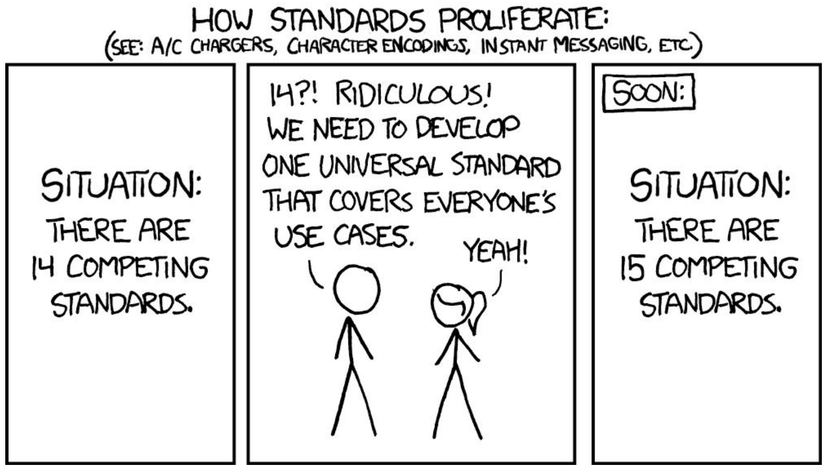


Google TPU
https://en.wikipedia.org/wiki/Tensor_Processing_Unit#/media/File:Tensor_Processing_Unit_3.0.jpg

So what needs fixing?

Accelerators

- Your interesting compute is no longer just on your CPU
- Many competing standards



https://xkcd.com/927/

Uncore events and sampling

- Perf events are associated with threads and CPUs.
- Samples are associated with a process's virtual addresses.

oneAPI NVIDIA CUDA AMD ROCm

Modern and future topologies

- Heterogeneity, chiplets, workloads that span machines.

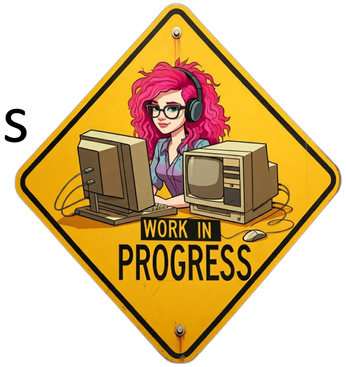
Can we meet these APIs where they are?

Expanding the perf tool to support non-perf_event_open APIs sounds alien but is already done for certain “tool” events.

Patch series showing this being done for hwmon “events”:
https://lore.kernel.org/lkml/20240907050830.6752-1-irogers@google.com/

```
$ perf stat -e temp_cpu,fan1,hwmon_thinkpad/fan2/,tool/num_cpus_online/ -M UNCORE_FREQ -I 1000
     1.001153138                52.00 'C   temp_cpu
     1.001153138                2,588 rpm  fan1
     1.001153138                2,482 rpm  hwmon_thinkpad/fan2/
     1.001153138                    8      tool/num_cpus_online/
     1.001153138      1,077,101,397      UNC_CLOCK.SOCKET                    #     1.08 UNCORE_FREQ
     1.001153138      1,012,773,595      duration_time
```

Scalability of perf events on heavily consolidated server platforms



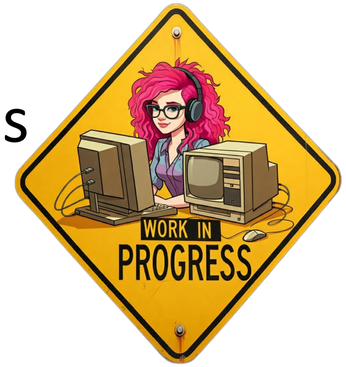
- Perf samples are virtual addresses, so 8-bytes
- BPF style BuildID+offset structs are 32-bytes

Virtual addresses require a memory map to convert the virtual address to a file (aka BuildID) and offset.

An mmap event is 72-bytes, or the increase in size of 3 BuildID+offset samples.

So BuildID+offset doesn’t pay off?

Scalability of perf events on heavily consolidated server platforms



Well hang on there..

- perf events support BPF filtering,
- opening an event twice, once in BuildID+offset mode and once in virtual address, means we can toggle the kind of samples,
- initially use BuildID+offset to avoid mmap event synthesis,
- once we know (via the ring buffer or BPF maps) the processes with samples, synthesize the mmaps and toggle to virtual address mode.

BuildIDs are good in production environments but developers may lack them. Possibly need BuildID to file path events.

More of what needs improving…

BPF and perf event implementation convergence.

- Hopefully improved by the BuildID+offset work.

Context switch performance

- Uncore events tend to be associated with CPU 0.
- With 100s of PMUs a cgroup context switch can end up swapping in and out 100s of events.

The kernel PMU abstraction, challenges in its specification, use and implementation.

- Drivers tend to cargo cult from one another.
- Intel’s driver is kind of the reference but it still does things badly
  - fixed, generic and MSR counters are all treated the same and exhaustion leads to multiplexing
  - we iterate all events in a context anyway for software events
- Multiplexing is based on insertion order, other orders could exist and be beneficial - like least scheduled
- PERF_EF_ flag use is inconsistent and not well defined.
- lack of hot plug support.

Yet more of what needs improving…

Overhauling points of pain

- mmap events give the VMA after the mmap, this means the size isn't the user requested size as VMAs may merge.
- The lack of mprotect, munmap, mremap events means the virtual address layout is only partially defined.
- There are special JIT compiler workarounds in the tool.

PMU precise event abstractions on non-Intel

- cpu/caps/max_precise only has meaning on Intel, consequently perf_event_open fallbacks rewind the precision requested for an event
- Can we do a better job integrating AMD IBS and ARM SPE events?

Getting richer information at low cost

- Owner of a locks during contention
- "Context" information in stack traces, such as when function is the python interpreter running

Some themes of what needs to be done

Bulk APIs

- Rather than 1 event at a time accelerators want multiple events programmed as a group
- The perf event grouping logic is quirky and requires many syscalls - it is already a point of pain for Intel topdown events on fixed counters on performance cores
- Streams vs events

Asynchronous

- Events aren’t on a CPU so we may receive many at once
- Need for core vs uncore throttling

More types of location, sample identifiers, ..

- Memory locations beyond virtual addresses, to cover memory on accelerators
- DRM job identifiers rather than sample identifiers

Extra thoughts

Being able to identify categories of events easily. For example, all types of power event

Virtual PMUS

- Perf used to map guest events to perf events on the host
- New “pass-through” mode - Guest OS “owns” the performance counters when running

Perf tool evolution, performance, dependency and distribution challenges.

- Should more of the UI tooling be written in Python?
- Should more of the core code be rewritten in Rust?
- Complaints of too many perf tool library dependencies
- Should kernel tools run on Windows?
- Distributions tie the perf tool to a kernel release, etc.

Kernel vs tool separation and licensing challenges

- The tool adopts kernel patterns and its GPLv2 license,
- Event encoding and perf_event_open are complex and worth sharing as a library (libperf),
- GPLv2 is a barrier to linking against the library.

Summary

Performance counters have been a kernel subsystem one way or another for 26 years.

The subsystem isn’t perfect at capturing 26 year old machines, or scaling on ones of today. It is missing an entire category of events for accelerators.

Lots of challenges to support new heterogeneous silicon with complex topologies.

Thanks to all working and contributing in this area.

Work on our team!

Google is hiring senior Linux kernel developers:

https://www.google.com/about/careers/applications/jobs/results/129305906700526278

Search for jobs on:

https://www.google.com/about/careers/applications/

LPC 2023 - Overview

Conference Details

The Linux Plumbers Conference is the premier event for developers working at all levels of the plumbing layer and beyond.

Taking place on Wednesday 18th, Thursday 19th and Friday 20th of September, this year we will be both in person and remote (hybrid). However to minimize technical issues, we'd appreciate most of the content presenters being in-person.

The in-person venue is the Austria Center, Vienna, Austria.

Bruno-Kreisky-Platz 1, 1220 Wien, Austria

Unless specified otherwise, the conference information will be shared in Central European Summer Time (CEST, UTC+02:00, Europe/Vienna timezone).

Sponsorship opportunities

Linux Plumbers Conference would not be possible without our sponsors. Many thanks to all the great organizations that have supported Linux Plumbers Conference over the years.

New sponsorship opportunities are available for 2024! We hope that your organization will consider joining our growing list of amazing sponsors this year. Find out more here.

LPC 2023 - Overview

Conference Details

The Linux Plumbers Conference is the premier event for developers working at all levels of the plumbing layer and beyond. Taking place on Wednesday 18th, Thursday 19th and Friday 20th of September, this year we will be both in person and remote (hybrid). However to minimize technical issues, we'd appreciate most of the content presenters being in-person.Taking place on Wednesday 18th,

Taking place on Wednesday 18th, Thursday 19th and Friday 20th of September, this year we will be both in person and remote (hybrid). However to minimize technical issues, we'd appreciate most of the content presenters being in-person.

The in-person venue is the Austria Center, Vienna, Austria.

Bruno-Kreisky-Platz 1, 1220 Wien, Austria

Unless specified otherwise, the conference information will be shared in Central European Summer Time (CEST, UTC+02:00, Unless specified otherwise, the conference information will be shared in Central European Summer Time (CEST, UTC+02:00, Europe/Vienna timezone).

Conference Details

The Linux Plumbers Conference is the premier event for developers working at all levels of the plumbing layer and beyond.

Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024