Perfmon2: status update

Stéphane Eranian <eranian@google.com>
Agenda

• What changed since last year?

• Status

• Mainline merge challenges
what changed?
perfmon2 interface changes

• No new features, quite the contrary!

• No struct arg modified by programming syscalls
  ▪ used to be for pfarg_setdesc_t (timeout), pfarg_pm*_t (error)
  ▪ possible to share parameters between sessions: system-wide

• Set switch timeout must be multiple of clock granularity
  ▪ kernel does not round it up anymore
  ▪ avoid mistakes later on
  ▪ granularity via clock_getres() (I know, it's not in libc!)
perfmon2 implementation changes

• Use hrtimers for timeout-based event set switching
  ▪ avoid problems with tickless kernel in system-wide mode
  ▪ less overhead

• More modular code
  ▪ group features by module: rw, intr, sets, pmu, init, sysfs, debugfs, syscalls, smpl,...
  ▪ clear separation between kernel and user headers
  ▪ make headers_install

• Rewritten x86 support
  ▪ all PMUs-specific code embedded into PMU description module (code+data)

• Vastly updated sysfs code

• New optional debugfs interface to report perfmon2 statistics
other perfmon2 changes

• Kernel source code managed with GIT on kernel.org
  ▪ improves manageability, visibility
  ▪ improves tracking of code contributions
  ▪ easier to track mainline kernel

• One release for each new mainline kernel
  ▪ yes, libpfm, pfmon version numbers are increasing again!

• All user code + mailing list hosted on SourceForge.net
perfmon2 experimental code

• Utrace-ready (internal tracing engine)
  ▪ automatic support for Roland McGraph's utrace interface
  ▪ key advantage: no ptrace() to stop/resume thread, can invoke syscalls directly
  ▪ need utrace-enabled kernel

• Intel PEBS/BTS/DS management interface (Markus Metzger, Intel)
  ▪ coordinate access to resources via internal API
  ▪ provides DS_AREA, IA32_DEBUGCTL context switch support
  ▪ simplify PEBS code for perfmon2
  ▪ code not yet released, waiting for full ds.h interface in mainline
libpfm changes

• IBM Power4, Power5, Power6 support (IBM)
• IBM Cell support (Toshiba)
• Sicortex Nodechip support (Phil Mucci)
• AMD Barcelona support (AMD)
  ▪ including full support for Instruction Based Sampling (IBS)
• Sun Sparc support (Ultra*, Niagara*) (Dave Miller)
• documentation cleanups (Cray)
• Python bindings (Google)
• dynamic system call numbers detection (2.6.24 and up)
pfmon changes

- Old Pentium II, III, Pro support (Vince Weaver, Cornell U.)
- SUN Sparc support (Ultra*, Niagara*) (Dave Miller)
- IBM Power4, Power5, Power6 support (IBM)
- IBM Cell support (Sony)
- Symbol correlations across dlopen/dlclose (CERN)
- Process attribution in system-wide (Phil Mucci)
- Intel PEBS support on Core 2
- Lots of bug fixes
status
Perfmon2 architecture summary

- **PMU Hardware**

**user level**
- sysfs
- syscalls
- perfmon core
- pmu
- res
- sets
- ctxsw
- intr
- smpl
- fmt
- perfmon arch-specific

**kernel level**
- file
- default
- two-way
- kernel-call-stack
- OProfile
- PEBS
## Supported processors

<table>
<thead>
<tr>
<th>HW Vendors</th>
<th>Model</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD</td>
<td>AMD64 family 6</td>
<td>Cornell U.</td>
</tr>
<tr>
<td>AMD</td>
<td>AMD64 family 15</td>
<td>Hewlett-Packard Laboratories</td>
</tr>
<tr>
<td>AMD</td>
<td>AMD64 family 16</td>
<td>AMD</td>
</tr>
<tr>
<td>Intel</td>
<td>Itanium (all models)</td>
<td>Hewlett-Packard Laboratories</td>
</tr>
<tr>
<td>Intel</td>
<td>Pentium II, Pentium Pro</td>
<td>Cornell U.</td>
</tr>
<tr>
<td>Intel</td>
<td>Pentium III, Pentium M</td>
<td>Hewlett-Packard Laboratories</td>
</tr>
<tr>
<td>Intel</td>
<td>Core Duo/Core Solo</td>
<td>Hewlett-Packard Laboratories</td>
</tr>
<tr>
<td>Intel</td>
<td>Pentium 4 (Neburst)</td>
<td>Intel</td>
</tr>
<tr>
<td>Intel</td>
<td>Core 2 Duo</td>
<td>Hewlett-Packard Laboratories</td>
</tr>
<tr>
<td>MIPS</td>
<td>many</td>
<td>Phil Mucci, SiCortex, Broadcom, Cornell U.</td>
</tr>
<tr>
<td>IBM</td>
<td>Power4, Power5, PPC970</td>
<td>IBM</td>
</tr>
<tr>
<td>IBM</td>
<td>Power 6</td>
<td>IBM</td>
</tr>
<tr>
<td>IBM</td>
<td>Cell</td>
<td>IBM, Sony, Toshiba</td>
</tr>
<tr>
<td>Cray</td>
<td>X2, XT</td>
<td>Cray</td>
</tr>
<tr>
<td>Sun</td>
<td>Ultra12, Ultra3*, Ultra4+</td>
<td>David S. Miller</td>
</tr>
<tr>
<td>Sun</td>
<td>Niagara1, Niagara2</td>
<td>David S. Miller</td>
</tr>
</tbody>
</table>

• to come: Intel Nehalem, Intel Tukwila
• still missing: ARM
general status and future work

• up to perfmon v2.8

• soon to be released
  ▪ Linux v2.6.26 patch, libpfm-3.5, pfmon-3.5

• future work:
  ▪ focus on merge (cleanup, simplify, explain)
  ▪ variable width counters
  ▪ drop MASKED state
  ▪ add PMD -> PMC dependency information for all PMUs
Gpfmon: pfmon GUI front-end

- Python-based, open-source front-end from CERN
what's up with the merge?
introduction

• To have impact, perfmon2 must be in mainline
  ▪ available in off-the-shelf distros

• Perfmon2 code base is big:
  ▪ spans 5 processor architectures
  ▪ touches context switch, syscalls, fork, exit, kernel exit, interrupt
  ▪ 1.1MB patch, 10,000 lines of C (patch)

• LKML review: cannot be merged as is
  ▪ too big, over-engineered
  ▪ concerns about extensibility (syscalls never disappear)
  ▪ feedback can be constructive: debugfs, sysfs, Kbuild,

• Must start from scratch: perfmon3
Our solution: minimal perfmon

• Start from scratch: perfmon3
  ▪ concentrate first on basic value-add: per-thread counting
  ▪ don't be afraid to break backward compatibility with perfmon2
  ▪ use their tool: quilt

• Quilt: what's that?
  ▪ collection of scripts to manage patches as a stack: push, pop
  ▪ force features (code) to be attributed to a specific patch
  ▪ easy to mail patch stack to LKML

• Minimal perfmon2 quilt series:
  ▪ track linux-next and mainline GIT trees
  ▪ import ONLY code to support per-thread counting
  ▪ drop everything else: sampling, event-sets, PMU descriptions, sampling formats
  ▪ only supports: Intel architectural, AMD AMD64 PMUs
  ▪ 200KB, 2000 lines of C (patch)
Minimal perfmon2 architecture summary

- PMU Hardware
  - user level
    - sysfs
    - syscall
  - kernel level
    - pmu
    - res
    - perfmon core
      - AMD64
      - intel x86 arch
      - ctxsw
      - intr
      - perfmon arch-specific
system calls and extensibility

• How to extend a syscall-based interface?
  1. add new system calls
  2. make the syscall parameters extensible

• Parameter extensibility:
  ▪ add new flag for new parameter, then kernel checks:
    • `pfm_func(int fd, int flags, pfarg_p_t *p);`
    • `pfm_func(int fd, int flags, pfarg_p_t *p, pfarg_q_t *q);`
    • `pfm_func(fd, XTRA_FEATURE, &p, &q)`

• Struct extensibility:
  ▪ add new flags for each new feature, reserved no need to be zero
    • `struct pfarg_p_t { int flags; uint64_t reserved[8]; }`
    • `struct pfarg_p_t { int flags; uint64_t new; uint64_t reserved[7]; }`
    • `struct pfarg_p_t p = { NEW_P, 0xf0, };`
syscall interface proposal

- Use flags for extensibility
- Merge pfm_start() and pfm_restart()
- Do we need pfm_delete_evtsets()?
- Drop pmd/pmc distinction? pfm_read_reg(), pfm_write_reg()
• All major processors supported now!

• Feature set is complete

• Merging with minimal patch series

• Strong community of users and contributors, thanks!
Thank You!

Q&A
How we got there?

- incremental work over several years
- most features were requested by advanced users (like you guys!)
  - huge gap between casual and advanced users
- over-engineered to provide maximum flexibility:
  - no need for yet another interface to support new HW
  - successful so far: IBM Power 6, Intel PEBS, AMD IBS, Itanium
- not a big fan of pushing complexity down to kernel
  - no event knowledge
  - system-wide is per-cpu
- development remained outside mainline too long
  - needed to validate interface on all major processors
  - difficult to change APIs once integrated