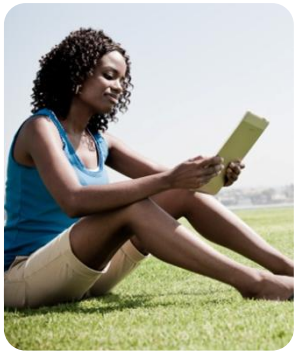
Open Source. Open Possibilities.



Multiple Performance Monitoring Units in Perfevents

Presented by: Ashwin Chaugule

Presentation Date: August 19, 2011

QuIC
QUALCOMM INNOVATION CENTER, INC.

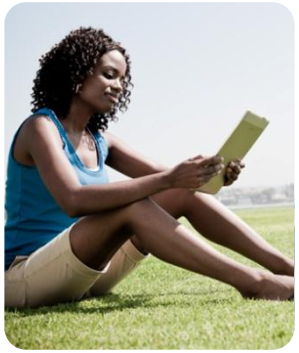
That gravity defying dive!

Photocred: Dominator Fridays http://www.facebook.com/TheUltimatePage
http://www.facebook.com/photo.php?fbid=10150123348217273&set=pu.300060247272&type=1&theater

Agenda

- Perfevents overview
- Current hardware PMU support in perfevents
- The missing parts
- Where we are in the ARM world
- Multiple PMU support added in ARM perfevents
- Where we are now
- What’s coming up in the near future

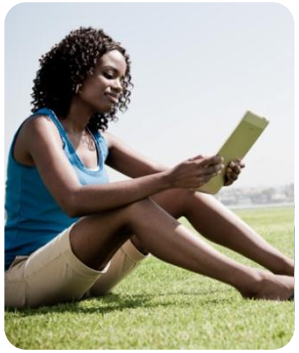
Perfevents



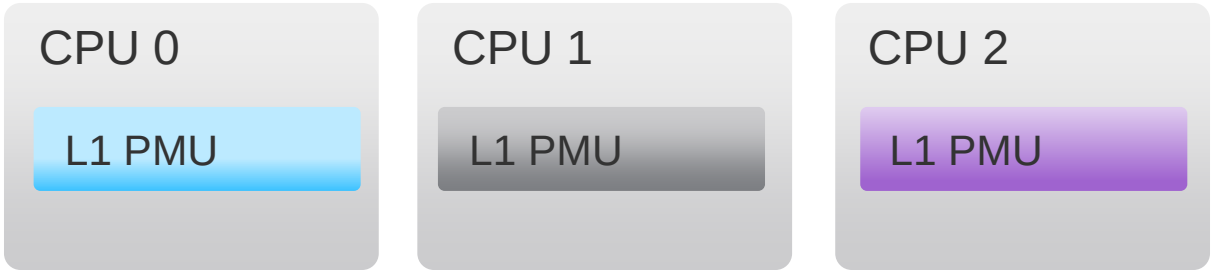
Perfevents

- Framework for monitoring the system
  - Software events
    - Context switches, migrations, page faults…
  - Hardware events
    - Cycles, instructions, cache stats…
  - And much, much more…
- Userspace support
  - Sys_perf_event_open(), IOCTL's EVENT_{DISABLE, ENABLE}
  - <kernel src>/tools/perf/
  - *Struct perf_event_attr*
  - perf binary includes a ton of sub-tools
    - perf stat
    - perf record → report
    - perf top
    - New stuff added almost every month!

Perfevents: Hardware PMU support



CPU side PMUs



- Primarily supported only CPU-side PMUs
- Easier to support using per-cpu data structures.
- Easier to sample per task / per thread / per CPU

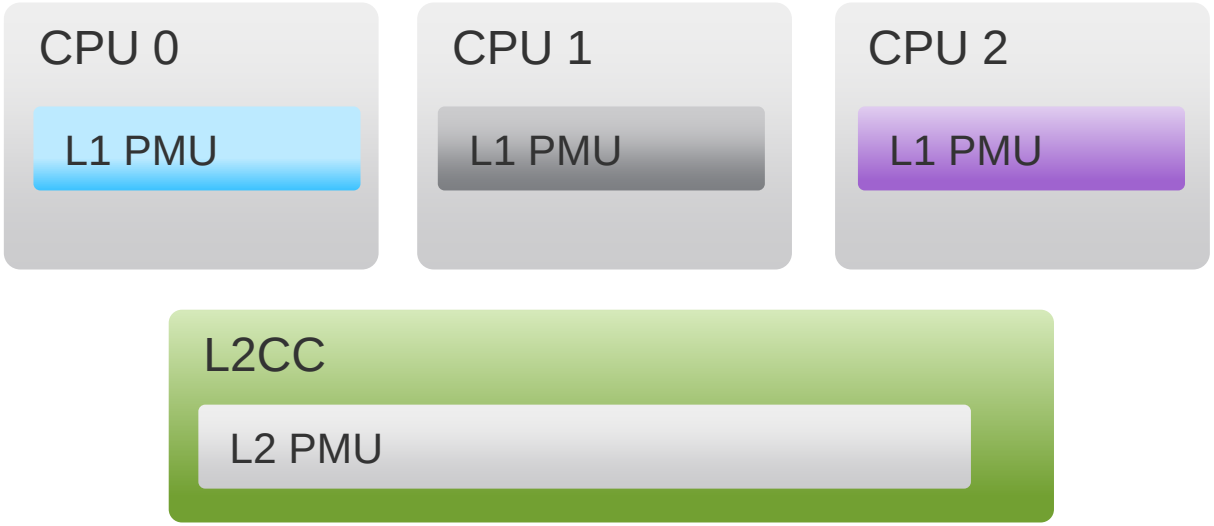
```
perf stat ls
Performance counter stats for 'ls':

        4938636  cycles                #   1180.822 M/sec
        1124192  instructions          #      0.228 IPC
         149797  branches              #     35.816 M/sec
          51796  branch-misses         #     34.577 %
    <not counted>  cache-references
    <not counted>  cache-misses

      0.005561630  seconds time elapsed
```

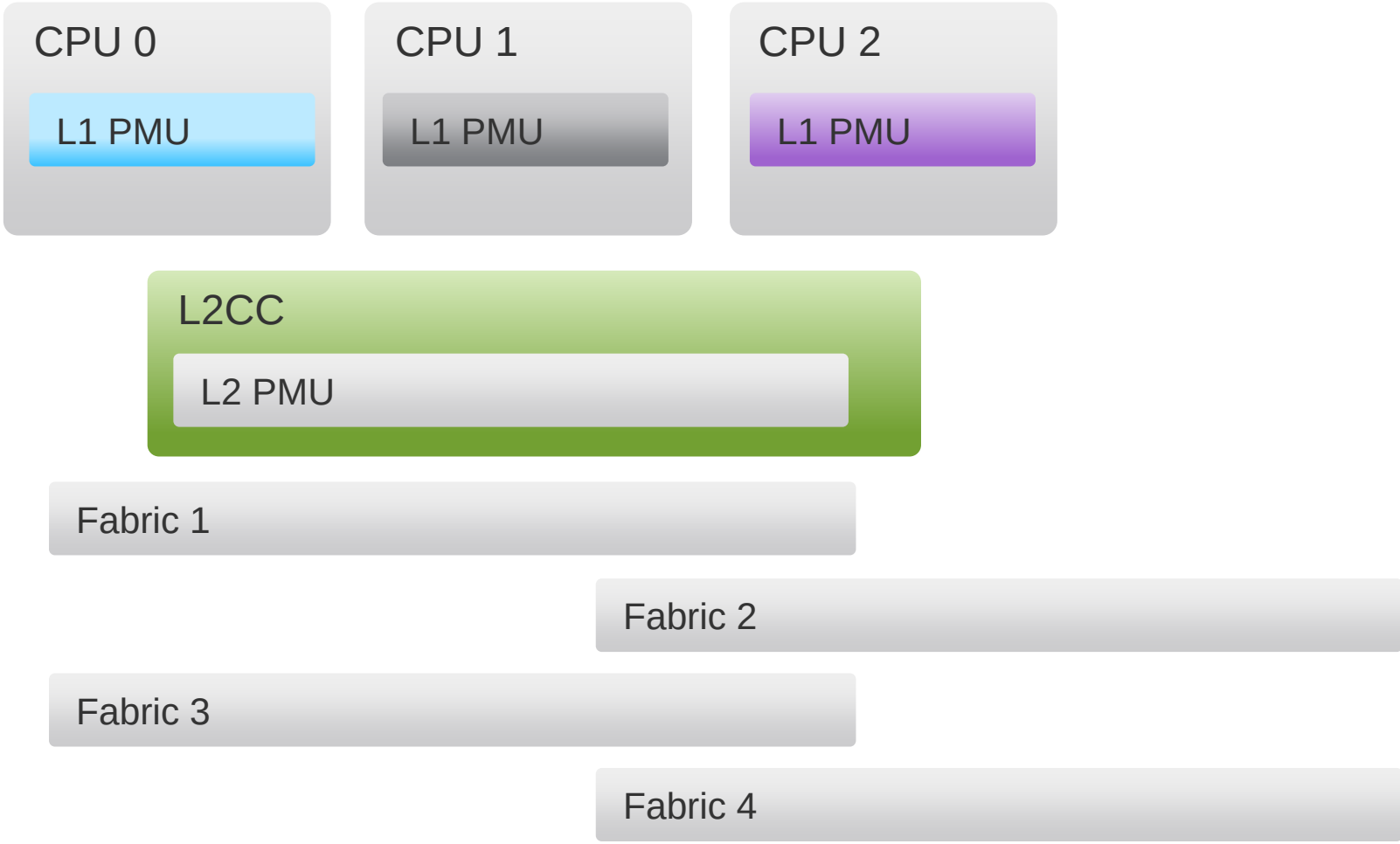
Multiple PMUs

- But there are more of these

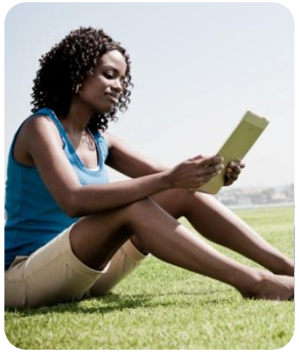


Multiple PMUs

- And then some more



Current State of Perfevents in ARM



ARM Perfevents

- Currently supporting ARM
  - v6, v6mp
  - v7
    - Cortex A8,
    - Cortex A9
  - v11, v11mp
  - xscale, xscalemp
  - Cortex A15 patches in RFC stage
- All above support is for CPU-side PMUs; L1CC stuff
  - Fits well with the design of perf-core code.
- Upstream code only supports one PMU at a time
  - Makes it easy to unify such PMU code.

ARM Perfevents

- Code is nicely organized for L1CCs
- Perf-core requires PMU registration via:
  - perf_pmu_register(&pmu, "cpu", PERF_TYPE_RAW);
  - Perf stat –e **r**XXX
- Only one *struct pmu* defined for all ARM variants
  - Only one of the ARM variants active at a time

```
static struct pmu pmu = {
        .pmu_enable     = armpmu_enable,
        .pmu_disable    = armpmu_disable,
        .event_init     = armpmu_event_init,
        .add            = armpmu_add,
        .del            = armpmu_del,
        .start          = armpmu_start,
        .stop           = armpmu_stop,
        .read           = armpmu_read,
};
```

- Each ARM variant has its own way of configuring the PMU, reading, writing counters, and interrupts

ARM Perfevents

- arm_pmu defines lower level plumbing of PMUs

```
static struct arm_pmu armv7pmu = {
        .handle_irq             = armv7pmu_handle_irq,
        .enable                 = armv7pmu_enable_event,
        .disable                = armv7pmu_disable_event,
        .read_counter           = armv7pmu_read_counter,
        .write_counter          = armv7pmu_write_counter,
        .get_event_idx          = armv7pmu_get_event_idx,
        .start                  = armv7pmu_start,
        .stop                   = armv7pmu_stop,
        .raw_event_mask         = 0xFF,
        .max_period             = (1LLU << 32) - 1,
};
```

- Similarly for armv6, v11, etc.
- At init, depending on cpuinfo
  - Global instance of struct arm_pmu points to one of the above

ARM Perfevents

- CPU-side PMUs have PERCPU data structs that hold info of events currently running on that CPU

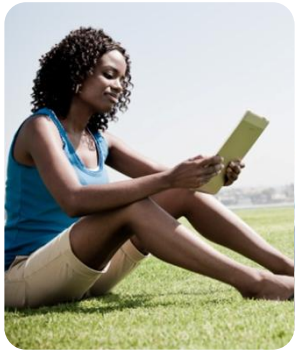
```
struct cpu_hw_events {
    struct perf_event  *events[ARMPMU_MAX_HWEVENTS];

    unsigned long    used_mask[BITS_TO_LONGS(ARMPMU_MAX_HWEVENTS)];

    unsigned long   active_mask[BITS_TO_LONGS(ARMPMU_MAX_HWEVENTS)];
};
static DEFINE_PER_CPU(struct cpu_hw_events, cpu_hw_events);
```

- PMU has PPIs (Private Peripheral Interrupts)
- L1CC PMU has four event counters and one cycle counter PER CPU
- Easy to profile by task

Multiple PMU support in ARM Perfevents



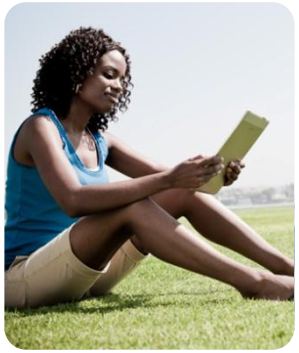
PMU Categories

- CPU-aware PMUs
  - Typically per-cpu, accessed via co-proc instructions
  - PPIs (private peripheral interrupts)
  - Counter outputs attributable to a task and CPU
  - e.g., L1CC, VeNum unit PMU
- Shared PMUs
  - Shared across CPUs; masters can only be amongst set of CPUs
  - Accessible via co-proc instructions
  - SPIs (shared peripheral interrupts)
  - Counter outputs ***may or may not*** be attributable to a task or CPU
  - e.g., L2CC PMU
- Peripheral PMUs
  - Typically monitor traffic from a master to a slave or have various combinations
  - Accessible via mem mapped I/O
  - Need at least one CPU to handle interrupts, program the PMU
  - e.g., Fabric PMUs

CPU-Aware PMUs

- Qualcomm's 8x50, 7x30
  - ARMv7-based CPUs; L1CC PMUs compatible with PMUv1
  - ARM architected 19 events so far
    - Codes 0x0 through 0x12 defined
    - 0x13 - 0x3f RESERVED
  - Qualcomm L1CC PMUs extend event space in the 0x40-0xfe space
    - 0xff is the cycle counter
  - Piggy back on armv7 pmu fops
    - Define own .enable .disable functions of struct arm_pmu
    - Access mechanism changes for event codes >= 0x40
    - Can reuse a lot of armv7 PMU code
- 8x60 and 8x90 L1CC
  - MP CPUs
  - Similarly define own .enable and .disable functions of arm_pmu
  - Have VeNum PMU (also CPU-aware)
    - But counting of VeNum events happen using L1CC counters
- One cycle counter + four event counters PERCPU

L2CC PMUs



L2CC PMUs

- Shared PMU category
- Qualcomm's L2CC PMU has
  - One cycle counter + four event counters
  - Shared across all CPUs
- Overflow interrupt is an SPI
- Started off getting this to work on 2.6.35
  - No multiple PMU support in perfevents, which came in 2.6.38
  - Patches up on Codeaurora

L2CC PMUs (2.6.35)

- Somehow needed to tag an event (perf_event) to the right PMU
- ARM registered only one PMU with perf-core
- Made own register_arm_pmu() function
- Define multiple PMUs of type arm_pmu
  - Embed struct pmu fops inside *struct arm_pmu*

```
struct arm_pmu foo = {
          . pmu = {  .pmu_enable = bar_enable_event,
                     .pmu_disable = bar_disable_event,
                      ...
                  }
          . read_counter =
          . write_counter = ..
};
```

  - Register embedded .pmu with perf-core
  - Access arm_pmu with
    - struct arm_pmu *armpmu = container_of(event->pmu, struct arm_pmu, pmu)

L2CC PMUs (2.6.38)

- Add new perf_type_id: PERF_TYPE_SHARED
  - Avoid collision with PERF_TYPE_RAW
  - Change perf userspace tool to parse differently
    - Perf stat –e r**s**XXX
    - attr::type changed to PERF_TYPE_SHARED if "**s**" exists
    - Separates event namespace from L1 events which have attr::type == PERF_TYPE_RAW
- perf_pmu_register(&l2_pmu, "L2", PERF_TYPE_SHARED);
- Skip struct cpu_hw_events completely, since this is not a PERCPU PMU
- Define

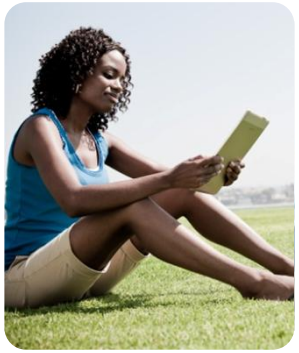
```
struct hw_l2_pmu {
        struct perf_event *events[MAX_L2_CTRS];
        unsigned long active_mask[BITS_TO_LONGS(MAX_L2_CTRS)];
        raw_spinlock_t lock;
};
```

- Add new arm_pmu_type :: ARM_PMU_DEVICE_L2
  - Treat L2 PMU as a separate platform driver

L2CC PMUs

- Qualcomm L2CC PMU can filter according to origin
  - Each counter has origin filter
  - Makes task-based filtering possible
- Perf core calls:
  - SYSCALL perf_event_open() - > Event init (called once)
  - pmu_disable
  - event_add (filter here)
    - event_start
  - pmu_enable
- Only one cycle counter
  - First CPU to “init” L2 cycle counting wins access
  - In perf stat “-a” mode, deny event “allocation” if cycle counter already active

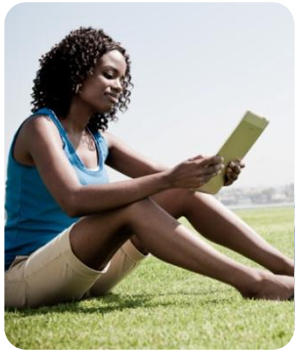
Fabric PMUs



Fabric PMUs

- WIP
- Challenges:
  - Multiple masters, multiple slaves, multiple fabrics
  - 64 bits of event attr:: config_base not enough
  - perf sampling modes “-a” (systemwide), task-based may not apply to all fabrics
    - But still need a CPU to config fabric PMU
    - Experimenting with task = -1 and cpu = -1 in perf tools
  - Typically start multiple counters at once
    - perf reads only one per “event”

Event Naming

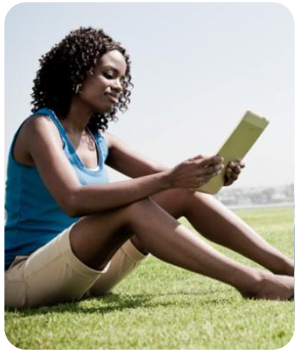


QuIC
QUALCOMM INNOVATION CENTER, INC.

Event Naming

- perf stat –e rXXX
- Need to define most commonly used events
  - e.g., perf stat –e cycles
  - A lot of these are esoteric
- Keep raw event encoding
  - Useful for controlling distribution of events
- Pfmlib4
  - Event string to raw encoding
  - Does pmu detection
  - Sets up perf attr:: members

Coming Up!



QuIC
QUALCOMM INNOVATION CENTER, INC.

Next Steps

- Wait for ARM code re-org to settle
- A9 L2CC PL310 patches from Will Daecon
- Scrap PERF_TYPE_XX stuff, dynamic PMU id detection
- Use sysfs hierarchy to list common events
- Unify SHARED PMU code
- Fabric type PMUs require some more thinking

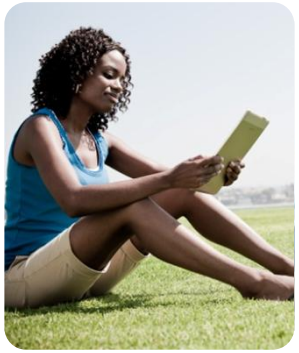
Next Steps

- Qualcomm L1CC and L2CC code for 7x30, 8x50, 8x60, 8x90 code available on Codeaurora.org
  - https://www.codeaurora.org/gitweb/quic/le/?p=kernel/msm.git;a=shortlog;h=refs/heads/msm-2.6.38
- 2.6.35 stuff is at:
  - https://www.codeaurora.org/patches/quic/qsd/
    - PATCH_M8260AAABQNLZA3055_6842_ScorpionMP-L2-cache-perfevents_20110616.tar.gz
- Re-org according to latest perf framework and RFC to LKAML

Disclaimer

Nothing in these materials is an offer to sell any of the components or devices referenced herein. Certain components for use in the U.S. are available only through licensed suppliers. Some components are not available for use in the U.S.

Thank You



QuIC
QUALCOMM INNOVATION CENTER, INC.