Linux perf_events updates

Stephane Eranian
Scalable Tools Workshop 2019
Lake Tahoe, CA

Agenda

- Quick updates on perf_events
- Intel IceLake support

SKX90 Erratum

- Intel Skylake (all models) with RTM enabled may lose access generic counter 3
  - Content of `EVTSEL` and counter for is unreliable
  - Counter 3 may not be available **regardless** of Hyperthreading mode
- Two options:
  - PMU priority: all RTM transactions abort, PMC3 available for monitoring
  - RTM priority: RTM transactions operate normally, PMC3 unavailable for monitoring
  - Controlled by new MSR `TSX_FORCE_ABORT`
- Linux support (v5.2): TFA
  - New sysctl `/sys/devices/cpu/allow_tsx_force_abort`
  - PMU priority: `echo 1 > /sys/devices/cpu/allow_tsx_force_abort` (boot default)
  - RTM priority: `echo 0 > /sys/devices/cpu/allow_tsx_force_abort`
  - Guarantee all events are out of counter 3 on return from `echo`

Intel Xeon Processor Scalable Family specification update June 2019

SK90 Erratum effects

1. with HT enabled and RTM priority, only 3 generic counters

```
$ echo 1 >/sys/devices/cpu/allow_tsx_force_abort

$ perf stat -a -C 0 --no-merge -I 1000  -e branches,branches,branches,branches sleep 4
#           time             counts unit events
     1.000543512      3,181,592,253      branches
     1.000543512      3,182,862,453      branches
     1.000543512      3,182,864,125      branches
     1.000543512      3,182,865,320      branches

$ echo 0 >/sys/devices/cpu/allow_tsx_force_abort

$ perf stat -a -C 0 --no-merge -I 1000  -e branches,branches,branches,branches sleep 4
#           time             counts unit events
     1.000549218      3,173,776,230      branches ( 75.00%)
     1.000549218      3,174,894,652      branches ( 75.01%)
     1.000549218      3,173,908,733      branches ( 75.02%)
     1.000549218      3,174,442,578      branches ( 74.96%)
```

¾ counters available

Group of 4 generic counter events will fail:

```
$ perf stat -a -C 0 --no-merge -I 1000 -e '{branches,branches,branches,branches}'
```

Intel CascadeLake : Persistent Memory Module support

- core/uncore PMUs identical to SkylakeX
- PMU support for Persistent Memory Module (PMM) AppDirect or Memory Mode
  - Core PMU: `MEM_LOAD_RETIRED.LOCAL_PMM, MEM_LOAD_RETIRED.REMOTE_PMM`
  - Core PMU: `OFFCORE_RESPONSE`
  - Lack of writeback coverage
  - Uncore PMU IMC: `UNC_M_PMM_RPQ_OCCUPANCY.ALL, UNC_M_PMM_RPQ_INSERTS, UNC_M_PMM_CMD1.*, UNC_M_PMM_WPQ_OCCUPANCY.ALL, UNC_M_PMM_WPQ_INSERTS`
  - Uncore PMU M2M: `UNC_M2M_IMC_READS.TO_PMM, UNC_M2M_IMC_WRITES.TO_PMM`
  - Possible to measure PMM read/write bandwidth and access latencies for reads/writes

Intel Icelake support

- Architectural Perfmon v5:
  - Anythread deprecated
  - Enumeration of fixed counters
- PERF_METRICS
- PEBS v4

Intel Icelake PMU: Anythread bit recap

- Per-counter filter (incl. fixed counters)
- Increment counter when event occurs in **either** thread

```
$ perf stat -e cpu/event=0x3c,any/,cpu/event=0x3c/ -I 1000 -a -C 10,66
#           time CPU                counts unit events
     1.000097760 CPU10           83,156,388      cpu/event=0x3c,any/u
     1.000097760 CPU66           83,156,367      cpu/event=0x3c,any/u
     1.000097760 CPU10           43,494,158      cycles:u
     1.000097760 CPU66           39,665,495      cycles:u
```

(CPU10, CPU66 siblings)

- Removed over concerns of security and complexity
  - Can be used to snoop on sibling which may be running program from distinct user

Intel Icelake PMU: Anythread bit alternative

- New `CPU_CLK_UNHALTED.DISTRIBUTED` core event
  - Increment by 1 in each cycle where **only** the current thread executes uops
  - Alternative **only** for `cycles`
- Example: ping-pong between 2 threads sharing same core

```
$ perf stat -e cpu/event=0xec,umask=0x2,name=cycles_distributed/,cycles pong
```

CYCLE	1	2	3	4	Total
THREAD0 executes	✓		✓		2
THREAD1 executes		✓		✓	2
THREAD0 **cycles:any**	1	1	1	1	4
THREAD1 **cycles:any**	1	1	1	1	4

4/2 = 2

CYCLE	1	2	3	4	Total
THREAD0 Executes	✓		✓		2
THREAD1 Executes		✓		✓	2
THREAD0 **cycles_distributed**	1		1		2
THREAD1 **cycles_distributed**		1		1	2

Intel IceLake core PMU: more counters!

	Skylake	Icelake
Generic counters per CPU HT on	4	**8**
Generic counters per CPU HT off	8	8
Fixed counters	3	4
Fixed metrics counters	0	1

- Largest generic counter increase since Core 2
- More event constraints: many events **can only be counted on lower 4 counters**
- New kind of fixed counters: precomputed metrics instead of raw event counts
- More counters = less multiplexing = less overhead
- More counters = more state = more expensive ctxsw, but RDPMC latency is improved 4x (15 cycles)

Intel Icelake PMU: `PERF_METRICS` topdown counter

- New fixed counter: `PERF_METRICS` (fixed ctr4)
- Topdown level 1 (unit: % of issue slots):
  - `frontend bound, backend bound, bad speculation, retiring`
- Fixed counter 4: `PERF_METRICS`
  - 4x 8-bit fields interpreted as fraction of 255
  - Example: frontend bound (bits[16:23])=120 => 120/255 = 47%
  - Computes 4 topdown level 1 metrics **since last reset**
-

Intel IceLake PMU: PERF_METRICS

- Pros of PERF_METRICS:
  - 1 counter read = 4 pre-computed metrics vs. Skylake needed 5 events = 5 counter reads
  - Computes metrics **per-thread** : huge improvement vs. Skylake which is per-core only
  - Topdown can be measured with up to 8 + 3 other events (HT on)
- Cons of PERF_METRICS:
  - Accuracy: 1 / 255 = ±0.4%
  - Needs reset on read to understand point of reference
  - Must be reset regularly to maintain precision
  - No sampling support
  - Linux support is not trivial due to single counter multiple metrics, not incrementing metrics

Intel IceLake PMU: `TOPDOWN.SLOTS` event

- New fixed counter: `TOPDOWN.SLOTS` (fixed ctr3)
  - Architectural event enumerated by CPUID `0xa` leaf function
  - Supported on generic counters as event `0x01a4`
- Counts number of available issue slots per logical CPU
  - Counts unhalted_cores_cycles * machine_width
  - Distributes slots between unhalted logical CPU (HT threads)

Intel Icelake PMU: `PERF_METRICS` Linux support

- Intel (Andi Kleen, Kan Liang) LKML patches from 07/24/2019 under review
- Exports new pseudo-events (`/sys/devices/cpu/events`)
  - `topdown-retiring, topdown-fe-bound, topdown-be-bound, topdown-bad-spec`
  - `Event code 0x00, umask=0x10, 0x11, 0x12, 0x13`
  - `RDPMC` supported though tricky when computing deltas
- Exported as individual pseudo events but must be in same event group
  - Cause single RDPMC in kernel and automatic reset on read
- Perf stat support

```
$ perf stat -I 1000 --topdown -a
#           time             counts unit events
     1.000373951      8,460,978,609      topdown-retiring # 22.9% retiring
     1.000373951      3,445,383,303      topdown-bad-spec #  9.3% bad speculation
     1.000373951     15,886,483,355      topdown-fe-bound # 43.0% frontend bound
     1.000373951      9,163,488,720      topdown-be-bound # 24.8% backend bound
```

Multiplying `topdown.slots` to scale counts

Actual Topdown breakdown

Intel Icelake PMU: PEBS recap

- **Processor** (used to be Precise!) Event-Based Sampling
  - Hardware buffer in virtual memory to save samples for limited set of at-retirement events
  - PMU interrupt only when buffer threshold reached
  - Record machine state at each sample
  - Provide precise IP pointing to instruction causing event, i.e., skidless
- Pros of PEBS
  - Precise IP
  - Lower sampling overhead: 1 PMU Intr for N samples (Linux: 341 samples/PMU intr)
  - Sampling data addresses for load/store events
- Cons of PEBS
  - Constraint on sampling mode: cannot use Linux frequency mode or certain PERF_SAMPLE_*
  - Not all events supported
  - Does not virtualize
  - PEBS record fixed size (24 x 8 = **192 bytes**). Too big for most measurements, wasted space+cycles

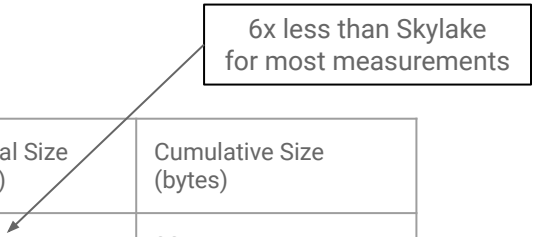
Intel Icelake PMU: Extended PEBS support

- Adds Extended PEBS
  - All counters support PEBS, including Fixed counters 0-3 (Skylake only generic counters 0-3)
  - All events can generate PEBS records
  - Only events supporting precise sampling will generate precise IP, others will include variable skid
- Pros of Extended PEBS
  - Amortize cost of sampling with PEBS buffer for ALL events
  - For non-precise event, skid on IP likely smaller (better) than without PEBS
  - Sample on more PEBS events simultaneously
- Cons of Extended PEBS
  - Does not make all events precise
- Linux support in v5.2
  - Nothing specific to do in tools, can just enable with precise_ip > 0 on **any** events

```
$ perf stat -e cpu/event=0x08,umask=0x2,name=dtlb_load_misses_walk_completed_4k/pp
```

Intel Icelake PMU: Adaptive PEBS support

- Adds Adaptive PEBS
  - PEBS record are configurable by groups of fields
  - PEBS can record full `LBR`
  - PEBS can record `XMM` registers
- PEBS groups



Name	Required	Regs	Size (bytes)	Minimal Size (bytes)	Cumulative Size (bytes)
Basic	Yes	eventing_IP, TSC, OVFL, REC FMT	4 * 8 = 32	32	32
Memory	No	Data addr, ldlat, latency, tsx_info	4 * 8 = 32	32 + 32 = 64	64
GPR	No	EFLAGS, RIP, EAX, EBX, ECX, EDX, RDI, RSI, R8-R15	18 * 8 = 144	32 + 144 = 176	240
XMM	No	XMM0-XMM15	16 * 8 = 128	32 + 128 = 160	400
LBR	No	32x LBR_TO/LBR_FROM/LBR_INFO	32 * 24 = 768	32 + 768 = 800	1200

Intel Icelake PMU: Adaptive PEBS Linux support

- Supported in Linux v5.2
- Kernel automatically selects field groups
  - Based on `attr.sample_type` and `attr.regs_smpl_usr/intr`

Perf command line	PEBS field groups	PEBS record size (bytes)
`perf record -e cpu/event=0xd0,umask=0x81/pp`	basic	32
perf record **-d** -e cpu/event=0xd0,umask=0x81/pp	Basic + memory	64
perf record **-intr-regs=AX** -e cpu/event=0xd0,umask=0x81/pp	Basic + GPR	176
perf record **-b** -e cpu/event=0xd0,umask=0x81/pp	Basic + LBR	800
perf record **-intr-regs=XMM0-XMM7** -e cpu/event=0xd0,umask=0x81/pp	Basic + XMM	128
perf record **-b -d --intr-regs=DI,SI** -e cpu/event=0xd0,umask=0x81/pp	Basic + GPR + Memory + LBR	976

Google `perf_events` current challenges

- **Scalability** of the interface and implementation is the biggest challenge
  - Worried about single machine scalability
- Machines are getting bigger with 200+ CPUs
- `perf_events` model: individual event abstraction
  - 1 file descriptor per event, 1 event instance per CPU (system-wide mode)
  - Cgroup mode: 1 event instance per cgroup per CPU
- Examples on 256 CPU machine and 200 cgroups:

`$ perf stat -a -e a,b,c,d,e,f` : 6 * 256 = 1536 fds

`$ perf stat -a a,b,c,d,e,f -G grp0,grp2,grp3, …,grp199` : 6 * 256 * 200 = 307,200 fds, 51,200 events/CPU

- Cost of multiplexing
- Google is working on improving scalability: patches posted on LKML

Conclusion

- Intel Icelake PMU packs the most PMU improvements
- PEBS improvements allows lowering sampling cost by a lot
- `PERF_METRICS` introduce a new counter concept here to stay

References

- Intel SDM Vol 3b May 2019 edition