Linux perf_events updates

Stephane Eranian

Scalable Tools Workshop 2018

Solitude, UT

Agenda

- Quick updates on perf_events
- Cache QoS
- Code heat maps
- Branch mispredictions

Google

Perf_events news

- `PERF_SAMPLE_PHYS_ADDR` (v4.14)
  - Extract data physical address, useful for cache simulation
- Spectre-v1 (v4.17) protection
  - Code impact: eliminate array indexing in speculative code
- Intel Skylake uncore IIO PMU free running counters PMUs (v4.17)
  - PCIe bandwidth, utilization, IIO clockticks
  - Ex: `perf stat -a -I 1000 -e uncore_iio_free_running_0/bw_in_port0/`
- PMU capabilities exported in `sysfs` (v4.14)

```
% grep . /sys/bus/event_source/devices/cpu/caps/*
      /sys/bus/event_source/devices/cpu/caps/branches:32
      /sys/bus/event_source/devices/cpu/caps/max_precise:3
      /sys/bus/event_source/devices/cpu/caps/pmu_name:skylake
```

- Event scheduling improvements
- Lots of bugs fixes

Cache QoS introduction

- Monitor cache and memory subsystem utilization per process
  - Track offenders
- Enforce cache and memory subsystem utilization restrictions per process
  - Isolate offenders
- Useful in shared runtime environments (Cloud, Web)
  - Machine is shared by workloads with different memory usage or SLO
- Hardware support introduced with Intel Haswell EP

Cache QoS: cache *occupancy*

- Monitor cache usage (CMT) - Intel Haswell EP
  - Track **new cache line allocations** (loads, rfo, hw/sw prefetch)
  - Supported in L3 (L2 on some processors)
  - Tag process with RMID
  - 4 RMID per core on Haswell, 8 on later processors
  - RMID saved/restored on context switch
- Cache partitioning (CAT) - Intel Haswell EP
  - Enforce cache utilization restrictions
  - Can split Code vs. Data (CDP) on Broadwell EP and later
  - 4 CLOSID on Haswell, 16 on later processors (8 with CDP enabled)
  - Restriction enforced per way (not all combinations possible)
  - CLOSID saved/restored on context switch

Cache QoS : memory bandwidth

- Monitor memory bandwidth usage (MBM) - Intel Broadwell EP
    - Count cache lines read and written back from LLC
    - Count total vs. local socket traffic
    - Tag process with RMID
    - 8 RMID per core
    - RMID saved/restored on context switch
- Memory bandwidth allocations (MBA) - Intel Skylake X
    - Restrict by percentage of total peak bandwidth per core
    - Tag process with CLOSID
    - 8 CLOSID
    - CLOSID saved/restored on context switch
    - Control enforced per core (not per socket), both hyperthread impacted
    - Control applied between L2 -> L3 => L3 hits are also slowed down

Google

Cache Qos: Linux support

- New interface introduced in Linux v4.9 (Intel contribution)
  - Using a new filesystem: `resctrl`
  - No cgroup, perf_events, `perf` tool connections anymore
  - Old interface deprecated
- Retain principles of `cgroup` fs
  - A global default resource group (root of fs)
  - One subdir per resource group
  - Tasks are moved into resource groups with : `echo tid > tasks`
  - Monitoring data read via specific file entry

- Enforcement via a programmable text-based schemata

```
# cat schemata
L3:0=1ff;1=1ff
MB:0=100;1=100
```

Cache QoS: example

- Cache Monitoring: Read cache occupancy on socket0 for `my_grp`:

```
$ mkdir /sys/fs/resctrl/my_grp
$ echo $$ >/sys/fs/resctrl/my_grp/tasks (move my shell into my_grp)
$ do_some_work
$ cat /sys/fs/resctrl/my_grp/mon_data/mon_L3_00/llc_occpuancy
```

- Bandwidth Monitoring: Read local memory bandwidth (read/write)

```
$ mkdir /sys/fs/resctrl/my_grp
$ echo $$ >/sys/fs/resctrl/my_grp/tasks
$ do_some_work &
$ cat /sys/fs/resctrl/my_grp/mon_data/mon_L3_00/mbm_local_bytes
```

- RMID Limit for CMT
    - RMID must be recycled
    - Cannot reset a RMID, must wait for line evictions
    - Kernel keeps list of RMID in limbo

Cache QoS: Triad example on Broadwell EP

```
$ mkdir /sys/fs/resctrl/memtoy;echo $$ >/sys/fs/resctrl/memtoy/tasks
$ triad -i 0 -r 0 -l 3 & (streaming 3 buffers of 256 MiB each)
$ cd /sys/fs/resctrl/memtoy/mon_data/mon_L3_00/
$ cat llc_occupancy
40260672  (40MB BDX L3 cache size)
$ cat mbm_local_bytes; sleep 1; cat mbm_local_bytes
1010472443904
                     (1023447851008-1010472443904)/(1000*1000)=12975 MB/S
1023447851008
$ perf guncore -M mem (uses CHA to compute local vs. remote traffic)
#------------------------------------------
#                 Socket0                 |
#------------------------------------------
#     Local     Local    Remote    Remote|
#        Wr        Rd        Wr        Rd|
#      MB/s      MB/s      MB/s      MB/s|
#------------------------------------------
    3198.76   9594.92      0.12      1.09
```

PMU based code heat map

- Where is the hot code?
- Why?
  - Improve code layout = Minimize ITLB pressure
  - Use large code pages (2MB or larger)
  - Do not exceed L1 ITLB 2MB entries capacity
  - Demote useless 2MB pages (save physical memory)
- How?
  - Using PMU based sampling
  - Intel: `INST_RETIRED:PREC_DIST` + PEBS or LBR
- Why `INST_RETIRED:PREC_DIST` (`0x01c0`)?
  - Introduced in Intel Sandy Bridge
  - HW assist to mitigate bias due to shadow of long latency instructions

Code heat maps

	start	size	%smpl
.plt	0x4099f0	8KB	0.10%
.text	0x40c000	6MB	7.37%
.text.unlikely	0xa8fd20	300MB	2.37%
.text.hot	0x1373e510	3MB	84.14%
.text_startup	0x13a76ea0	10MB	0.00%
.text.other	0x144ce580	8KB	1.89%
libc	0x7f463171a000	1.76MB	1.31%
libm	0x7f4631f3e000	1.02MB	1.40%
kernel	0xffffffff81000000		1.38%
other			0.03%
			99.99%

					4KB Page utilization			64B cacheline utilization		
page address	#TLB 2MB	Total %	Cum %	Page size	pages used	util	left	lines used	util	Source
0x13800000	1	72.93%	72.93%	2MB	471	91.99%	41	12952	39.53%	application
0x13a00000	2	6.87%	79.80%	2MB	109	21.29%	403	2855	8.71%	application
0x13600000	3	4.34%	84.14%	2MB	160	31.25%	352	4298	13.12%	application
0x400000	4	3.55%	87.69%	2MB	507	99.02%	5	15393	46.98%	application
0xa00000	5	2.04%	89.73%	2MB	97	18.95%	415	336	1.03%	application
0x14400000	6	1.89%	91.62%	2MB	3	0.59%	509	36	0.11%	application
0xa600000	7	1.61%	93.24%	2MB	4	0.78%	508	49	0.15%	application
0x600000	8	1.42%	94.66%	2MB	491	95.90%	21	11983	36.57%	application
0xffffffff81000000	9	1.31%	95.97%	2MB	373	72.85%	139	5785	17.65%	kernel
0x7f4631745000		0.83%	96.80%	4KB	1	100.00%	0	10	15.63%	libc.so
0x7f4631f52000		0.70%	97.50%	4KB	1	100.00%	0	31	48.44%	libm.so
0x12a00000	10	0.61%	98.11%	2MB	16	3.13%	496	158	0.48%	application
0x800000	11	0.46%	98.57%	2MB	323	63.09%	189	4273	13.04%	application
0x7f4631f4e000		0.22%	98.78%	4KB	1	100.00%	0	6	9.38%	libm.so
0x7f463173f000		0.15%	98.93%	4KB	1	100.00%	0	7	10.94%	libc.so
0x7f4631f9e000		0.10%	99.03%	4KB	1	100.00%	0	10	15.63%	libm.so

Code heat maps challenges

- Finding page sizes for a code address
  - Many different ways of getting huge pages for text (mremap, THP, tmpfs, ..)
  - Different impact on `/proc/PID/maps`
  - Difficulties for perf tool: must use heuristics
- `/proc/PID/smaps`
  - Shows if VMA is using large pages, but does not tell you the actual address range
- Need `perf_events` support
  - Need new sample format (`perf_event_attr.sample_format`)
  - `PERF_SAMPLE_CODE_PGSZ:` record code page size based on IP
  - `PERF_SAMPLE_DATA_PGSZ:` record data page size based on DLA
  - Must extract information from TLB or vm subsystem, tricky in NMI context

Google

Branch optimizations

- Minimize number of taken branches : FDO or autoFDO
  - Make the code as straight as possible
- Minimize branch misprediction
  - Minimize number of conditional branches
- Minimize branch misprediction cost
  - Mitigate penalty of misprediction
- Topdown branch cost
  - Front-End: Frontend Latency -> Branch Resteers = estimate cycles wasted to fetch correct path
  - Bad Speculation: wasted slots for uops not retiring or lost due to recovery from wrong path
- Large web workload impacted
  - Some large Google apps have 25% wasted uops

PMU support for branches

- Conditional mispredictions always on direction (taken vs. not taken)
  - Target is always known as IP-relative offset
- Need misprediction rate **per branch**
  - `BR_MISP_RETIRED.COND` : **does not tell taken vs. not taken**
- Need rate of taken vs. non-taken **per branch**
  - Use PEBS + `BR_INST_RETIRED.NOT_TAKEN` but missing `BR_INST_RETIRED.COND_TAKEN`
  - Perf_events PEBS issue: return either branch source or target, need both
- New sample format: `PERF_SAMPLE_SKID_IP` (posted on LKML)
  - With PEBS:
    - `PERF_SAMPLE_IP`: return source (precise>1) or target (precise = 1)
    - `PERF_SAMPLE_SKID_IP`: return target
  - Without PEBS:
    - `PERF_SAMPLE_SKID_IP` = `PERF_SAMPLE_IP`

PEBS and branches

- PEBS captures IP **at retirement** of sampled instruction
  - `PEBS.ip`: next dynamic instruction
  - `PEBS.eventing_ip` (Haswell and later): instruction causing event
- PEBS with perf_events
  - precise=1 => `PEBS.ip`
  - precise=2 or 3 => `PEBS.eventing_ip`

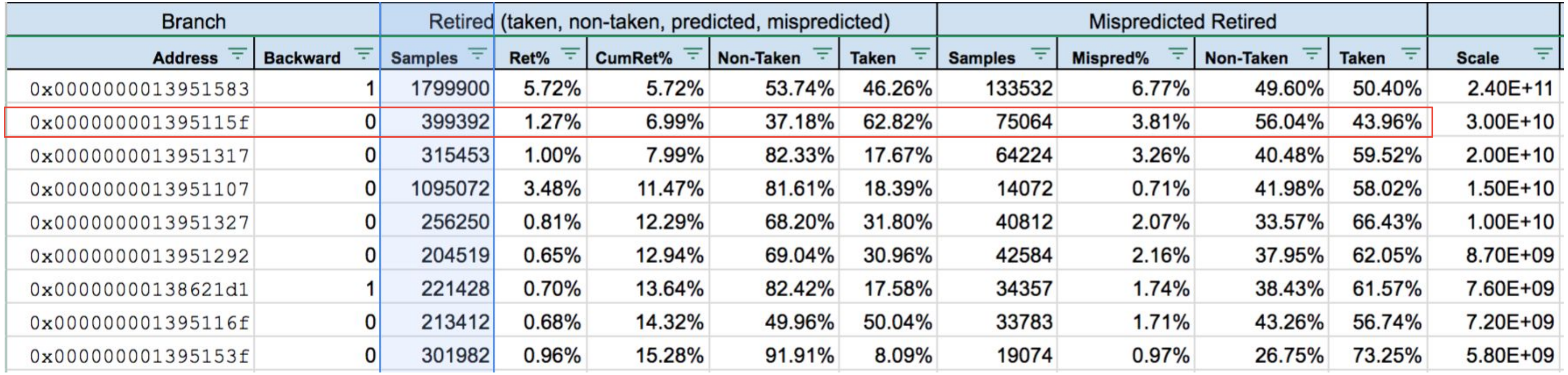
```
$ perf record -e \
  cpu/br_inst_ret.conditional/pp …

%smpl
21.5% 400343: test %rax,%rax  ←— macro-fusion
42.1% 400346: je   400358
      400348: mov  $0x404058,%edi
      ...
      400358: leaveq
```

```
$ perf record -e \
  cpu/br_inst_ret.conditional/p …

%smpl
      400343: test %rax,%rax          not taken
      400346: je   400358
41.5% 400348: mov  $0x404058,%edi
      ...
20.2% 400358: leaveq   ←———————— taken
```

Branch mispredictions example



Branch		Retired (taken, non-taken, predicted, mispredicted)					Mispredicted Retired				
Address	Backward	Samples	Ret%	CumRet%	Non-Taken	Taken	Samples	Mispred%	Non-Taken	Taken	Scale
0x0000000013951583	1	1799900	5.72%	5.72%	53.74%	46.26%	133532	6.77%	49.60%	50.40%	2.40E+11
0x000000001395115f	0	399392	1.27%	6.99%	37.18%	62.82%	75064	3.81%	56.04%	43.96%	3.00E+10
0x0000000013951317	0	315453	1.00%	7.99%	82.33%	17.67%	64224	3.26%	40.48%	59.52%	2.00E+10
0x0000000013951107	0	1095072	3.48%	11.47%	81.61%	18.39%	14072	0.71%	41.98%	58.02%	1.50E+10
0x0000000013951327	0	256250	0.81%	12.29%	68.20%	31.80%	40812	2.07%	33.57%	66.43%	1.00E+10
0x0000000013951292	0	204519	0.65%	12.94%	69.04%	30.96%	42584	2.16%	37.95%	62.05%	8.70E+09
0x00000000138621d1	1	221428	0.70%	13.64%	82.42%	17.58%	34357	1.74%	38.43%	61.57%	7.60E+09
0x000000001395116f	0	213412	0.68%	14.32%	49.96%	50.04%	33783	1.71%	43.26%	56.74%	7.20E+09
0x000000001395153f	0	301982	0.96%	15.28%	91.91%	8.09%	19074	0.97%	26.75%	73.25%	5.80E+09

References

- Cache QoS
  - Intel SDM Vol 3b, Chapter 17.19 Intel Resource Director Technology
  - CAT at Scale: Deploying Cache Isolation in a Mixed Workload Environment - Rohit Jnagal & David Lo, Google
  - Heracles: Improving Resource Efficiency at Scale