Enhancing PAPI with Low-Overhead `rdpmc` Reads

Yan Liu and **Vince Weaver**

{yan.liu,vincent.weaver}@maine.edu

ESPT Workshop 2017 — 12 November 2017

PAPI Background

- PAPI, the Performance API – widely used cross-platform performance library
- Extreme Scale? Finding where performance is going
- We reduced counter read latency in PAPI by 3-10x

Hardware Performance Counters

- Counters built into CPU that measure useful performance info:
  - Cycles, Instructions
  - Cache hits/misses
  - Branch predictor
  - etc.

Self-Monitoring

- Most other tools provide
  - Aggregate total count – total for entire program run
  - Statistical sampling – periodically read counters, extrapolate hot spots based on where interrupted
- PAPI also provides self-monitoring
  - Putting “calipers” around code of interest, giving exact counts
  - Does require inserting code into program, disrupting results

PAPI caliper

```
PAPI_create_eventset(&eventset);
PAPI_add_named_event(eventset,"PAPI_TOT_CYC");
PAPI_start();
...
PAPI_read(&value_before);

CODE OF INTEREST

PAPI_read(&value_after);
...
PAPI_stop();
```

`PAPI_read()` **is the key**

- On Linux perf_event, by default, uses `read()` syscall
- This calls into the kernel via syscall (slow) and disrupts execution
- Is there a better way?

`rdpmc` **instruction**

- x86 processors support `rdpmc` instruction which can read performance counters directly from userspace
- Operating system has to set bit in CR4 to enable this
- NOTE: this only works on the core CPU counters
  No Uncore counters, no RAPL counters

Add `rdpmc` to PAPI

- Not a new idea, perfctr (out-of-tree patch dating to 1999) did this and PAPI once supported it
- perf_event didn't originally until we complained. It's been there for years but no one had hooked it up

`rdpmc` **difficulties**

- Dropping rdpmc instruction into code is easy and fast
- If perf_event is running things though, problems
  - Kernel can re-schedule which event in which slot
  - If multiplexing is going on, events can be swapped out and counts might not reflect full time running
- Solution is kernel provides a page (per event) that can be `mmap()`ed that provides enough info
- Slower than just a `rdpmc`, but faster than `read()`

`rdpmc` **Pseudocode**

```
do {
        seq=pc->lock; barrier();
        calculate multiplex;
        get counter slot to read from;
        get previous count from kernel;
        rdpmc()
        adjust, scale, handle multiplex;
} while (pc->lock!=seq);
```

`rdpmc` **Linux Bugs Found**

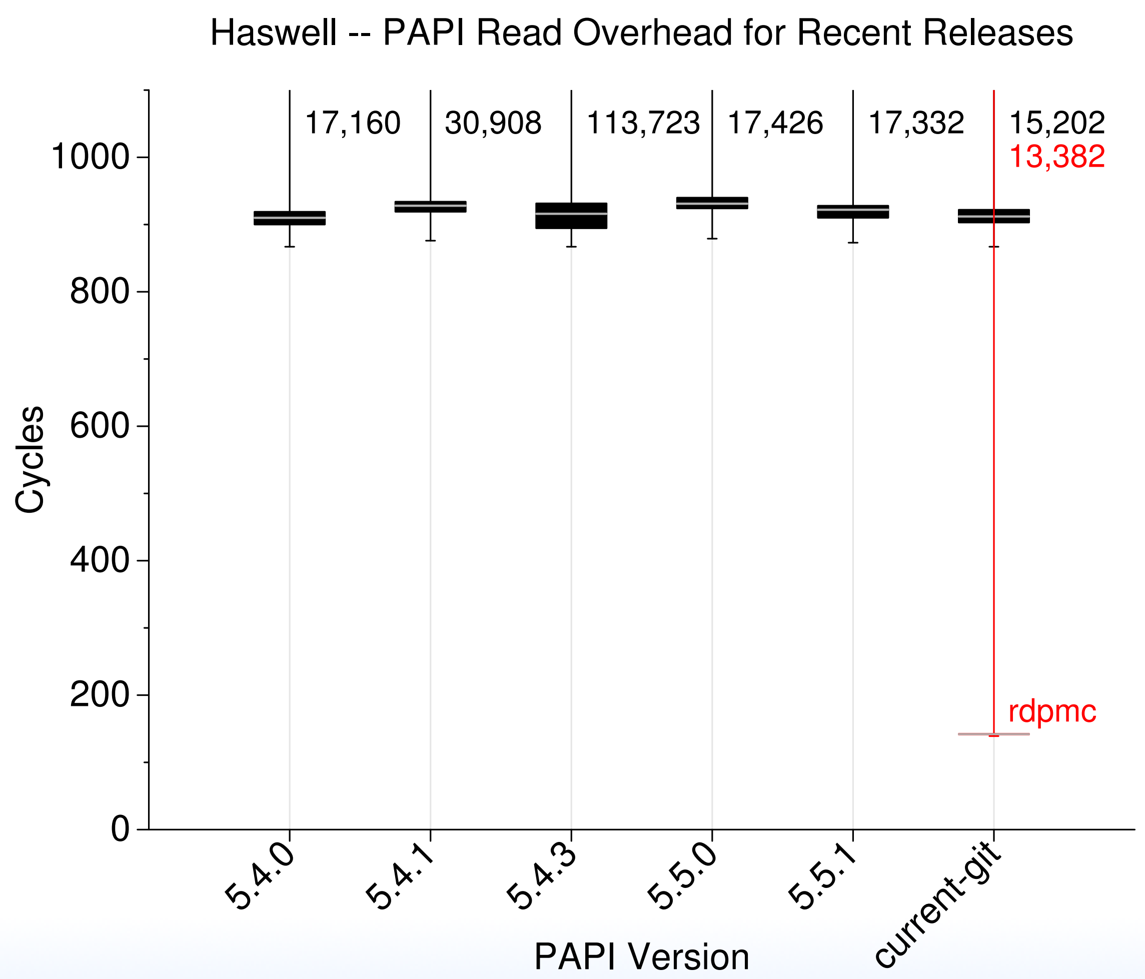
- Putting rdpmc in PAPI made various PAPI unit tests fail
  - CR4 GPF when using multiple threads, rdpmc ref count was wrong
  - calling exec() without munmapping also get rdpmc ref count wrong
  - when measuring attached process, time accounting wrong, causing PAPI to scale by hugely wrong number
- All of these were fixed by Linux 4.13

`rdpmc` PAPI results

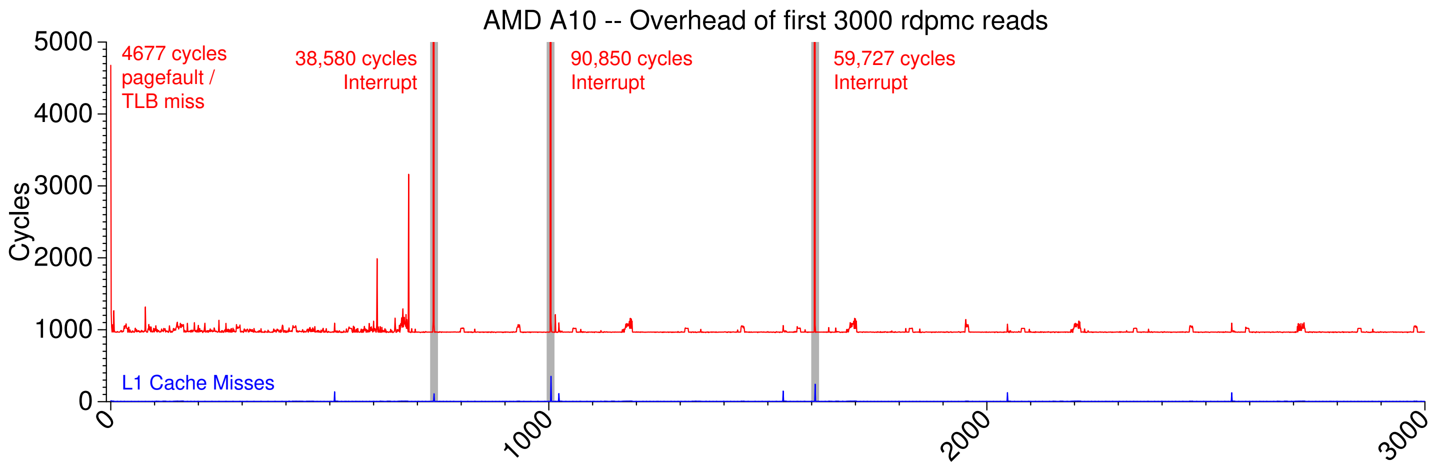
- PAPI_cost, runs million `PAPI_read()`s, these are median results

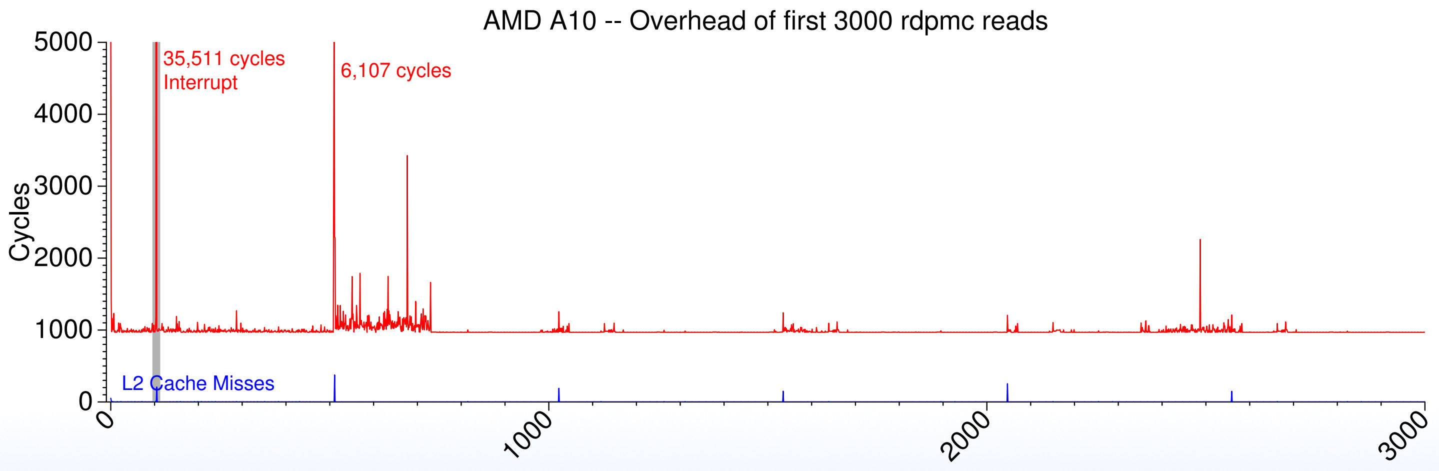
Vendor	Machine	read() cycles	rdpmc cycles	Speedup
Intel	Pentium II	2533	384	6.6x
Intel	Pentium 4	3728	704	5.3x
Intel	Core 2	1634	199	8.2x
Intel	Atom	3906	392	10.0x
Intel	Ivybridge	885	149	5.9x
Intel	Haswell	913	142	6.4x
Intel	Haswell-EP	820	125	6.6x
Intel	Broadwell	1030	145	7.1x
Intel	Broadwell-EP	750	118	6.4x
Intel	Skylake	942	144	6.5x
AMD	fam10h Phenom II	1252	205	6.1x
AMD	fam15h A10	2457	951	2.6x
AMD	fam15h Opteron	2186	644	3.4x
AMD	fam16h A8	1632	205	8.0x

Haswell Boxplot

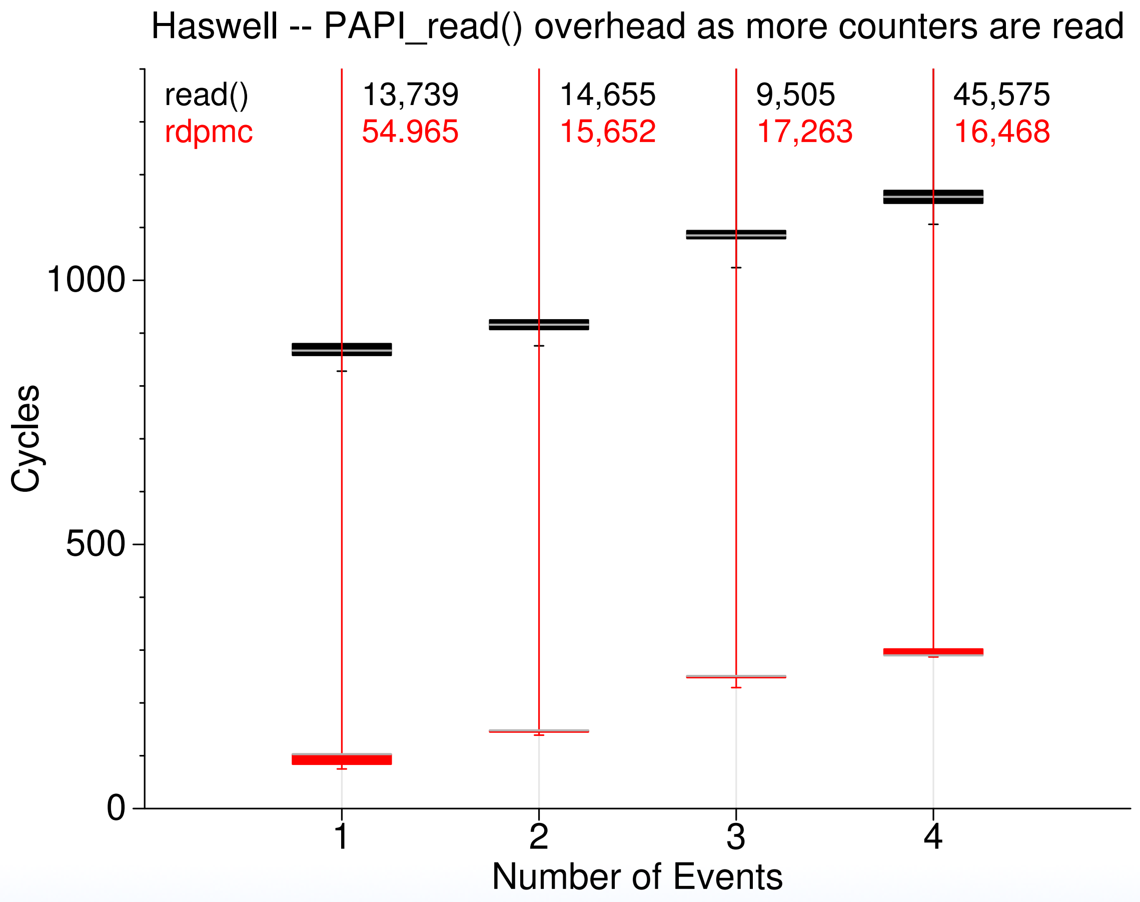


Source of Outliers (AMD a10)

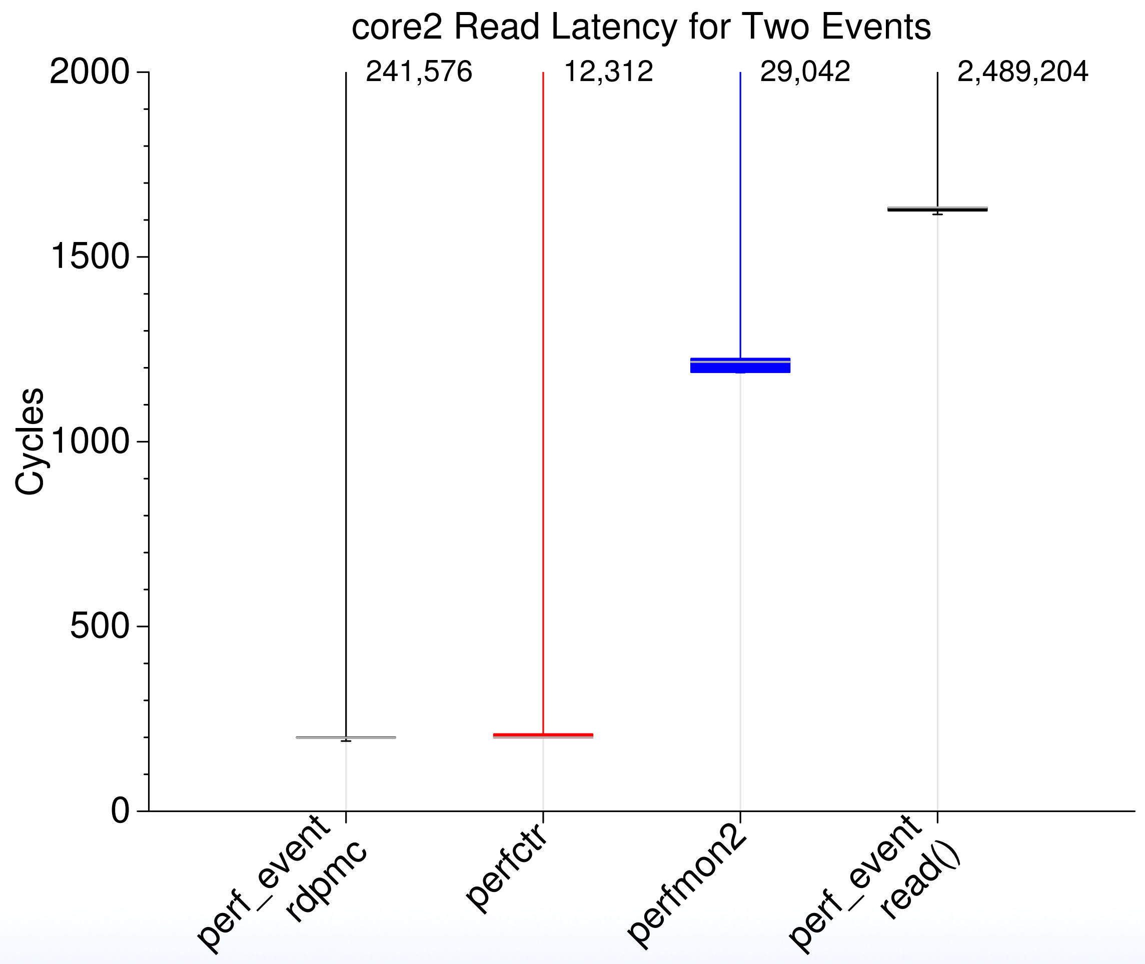




Reading Multiple



Historical Comparison (Core2)



Real-World Results / hpl — Haswell

Caliper around one function, results here are the second `PAPI_read()` call itself measured using `rdpmc`

Note: the cycle counter cycles aren't necessarily the same as rdtsc cycles

Routine	Type	Cycles		L1 DMiss		DTLB Miss	
		User	Kernel	User	Kernel	User	Kernel
HPL_pdpanel_init (low mem pressure)	rdpmc	512	0	5	0	0	0
	read()	461	1755	7	20	0	0
HPL_pdfact (high mem pressure)	rdpmc	4019	0	39	0	11	0
	read()	4551	13,545	43	123	16	16

TLB Impact of Multiple Events

Routine	Type	2 Events		3 Events		4 Events	
		User	Kernel	User	Kernel	User	Kernel
HPL_pdpanel_init (low mem pressure)	rdpmc	0	0	0	0	0	0
	read()	0	0	0	0	0	0
HPL_pdfact (high mem pressure)	rdpmc	11	0	14	0	16	0
	read()	16	16	15	17	16	18

Now Available in PAPI 5.6 Release

- Enabled by default. Need Linux 4.13 or newer

```
./papi_avail | grep rdpmc
Fast counter read (rdpmc): yes
```

Future Work

- ARM64 support – should be possible, someone developed patches but left before contributing

Questions?

http://web.eece.maine.edu/~vweaver/projects/papi-rdpmc/

vincent.weaver@maine.edu