The Upcoming PAPI 5.0 Release

Vince Weaver

`vweaver1@eecs.utk.edu`

CScADS Performance Tools for Extreme-scale Computing

27 June 2012

PAPI 5.0 has many changes

Most of the changes involve component support

- Some break the ABI (expected)
- Some break the CDI (component interface)
- Some break the API (we try our best to avoid this)

Overhaul of the CDI — `.cmp_info`

- OS-specific fields separated out (itimer, os_version, etc.)
- `_papi_hwd[0]->cmp_info` turned into `_papi_os_info`
- CPU specific fields (such as `cntr_IEAR_events`, `cntr_DEAR_events`) removed
- Names standardized away from CVS (“linux-net” not

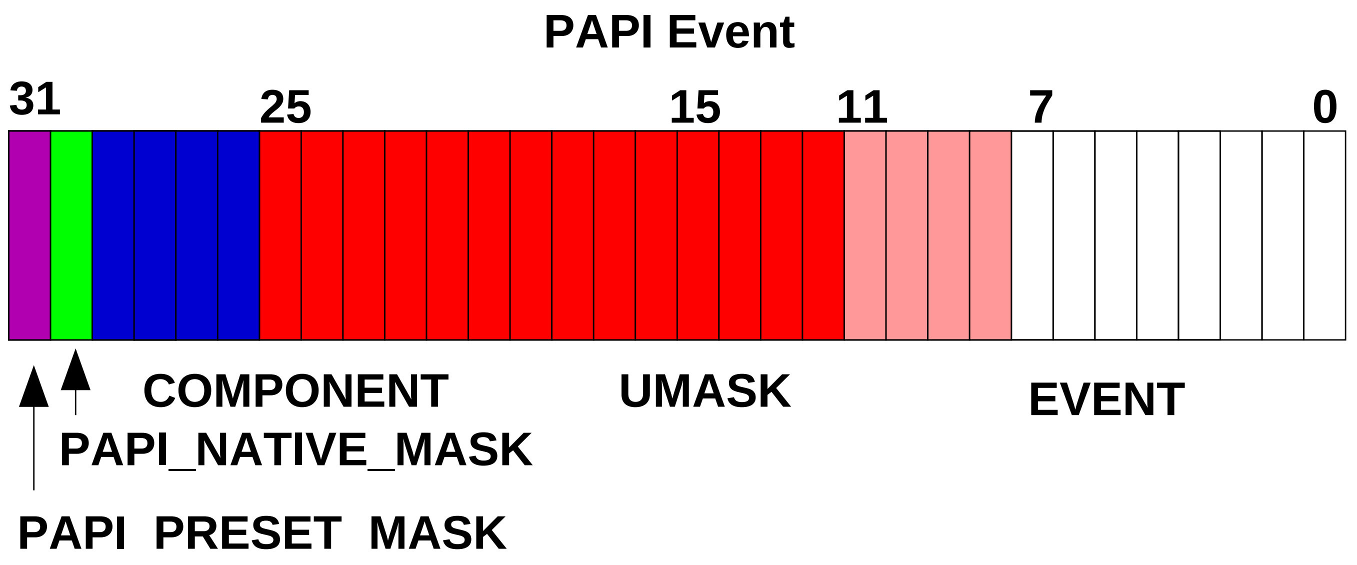
  `$Id: linux-net.c,v 1.11 2012/02/13 17:09:51 ter`
- Added short_name, description, disabled, disabled_reason, component_type

Overhaul of the CDI — `papi_vector_t`

- Move OS specific `papi_vector_t` functions to `papi_os_vector_t`: `get_real_cycles`, `get_real_usec`, `get_virt_cycles`, `get_virt_usec`, `update_shlib_info`, `get_system_info`, `get_memory_info`, and `get_dmem_info`.
- Add `get_real_nsec`, `get_virt_nsec` to allow ns for archs that support it (before it was just calc from usec)
- Remove Bipartite Map Functions
- Add `ntv_code_to_info`

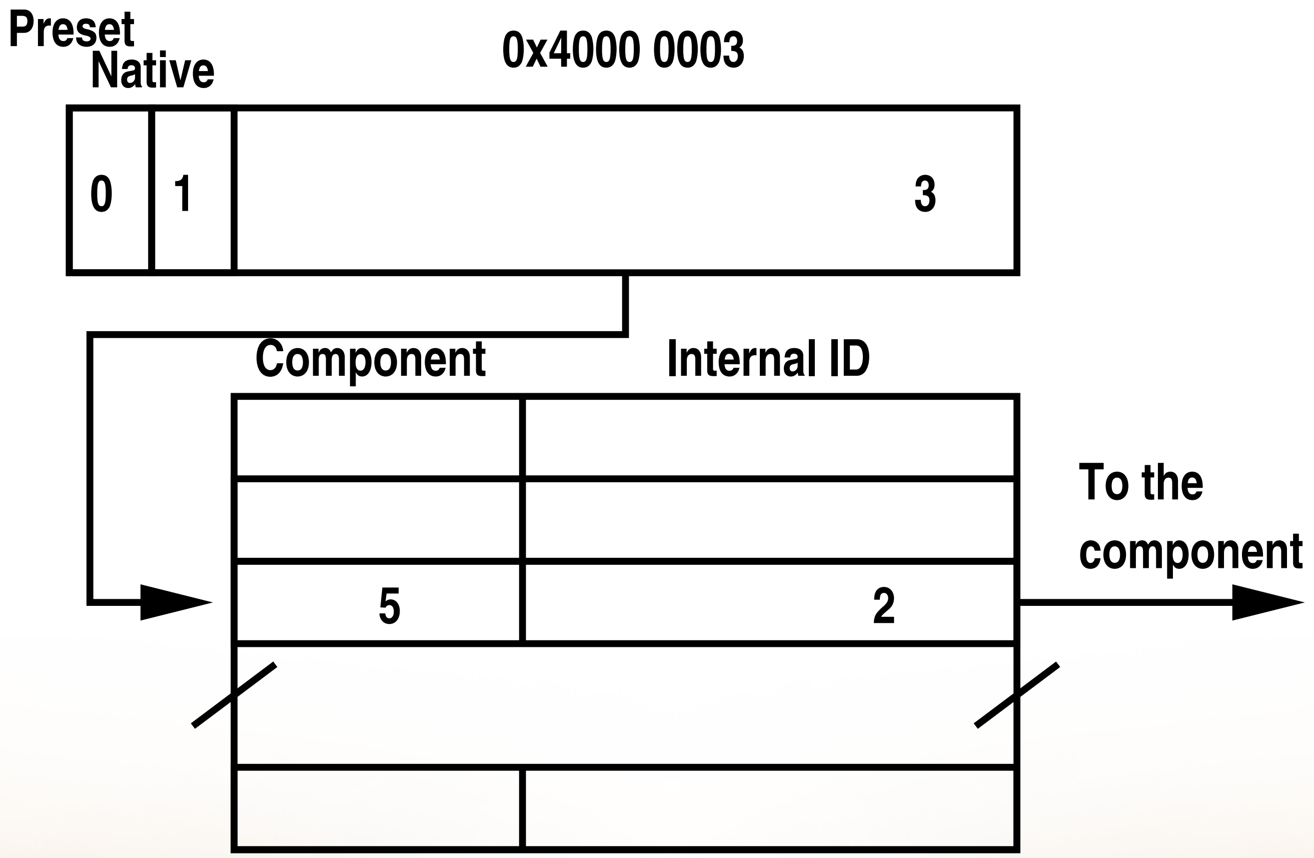
Removal of the 16-Component limit

How PAPI 4.x worked:



Removal of the 16-Component limit

New lookup method:



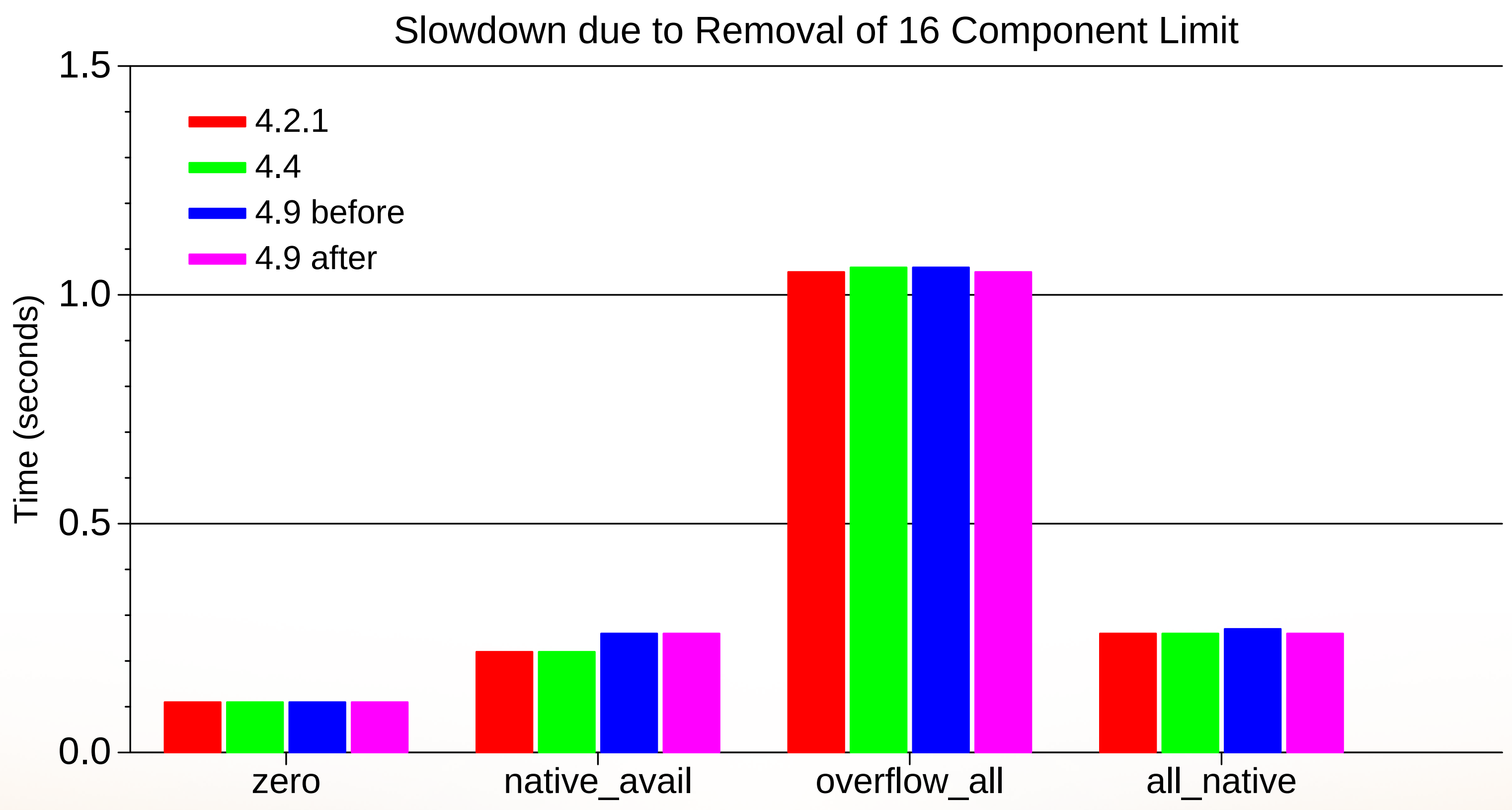
Removal of the 16-Component limit

New interfaces:

```
int PAPI_get_event_component(int EventCode);
PAPI_COMPONENT_INDEX() /* compatibility macro */
int PAPI_enum_cmp_event(int *EventCode, int cid, int modif
int PAPI_get_component_index(char *name);
```

Removal of the 16-Component limit

Slowdown due to new code:



New Interfaces — Named Events

Allow operating on events without having to lookup eventcode first:

```
int PAPI_add_named_event(int EventSet, char *EventName);
int PAPI_remove_named_event(int EventSet, char *EventName)
int PAPI_query_named_event(char *EventName);
```

New Interfaces — Disabling Components

```
int PAPI_disable_component( int cidx );
```

```
int cidx, result;

cidx = PAPI_get_component_index("example");

if (cidx>=0) {
   result = PAPI_disable_component(cidx);
   if (result==PAPI_OK)
      printf("The example component is disabled\n");
}
/* ... */
PAPI_library_init();
```

Enhanced Event Support

Kluge, Hackenberg, Nagel. *Collecting Distributed Performance Data with Dataheap*

- `data_type` — UINT64, INT64, FP64
- `units` — string
- `location` — core, cpu, package, uncore
- `timescope` — since start, since last, until next, point
- `update_type`, `update_frequency`

Enhanced Attribute Enumeration

```
| UNHALTED_CORE_CYCLES
|    Count core clock cycles whenever the
|    clock signal on the specific core is
|    running (not halted)
|    :e=0    edge level (may require c>=1)
|    :i=0    invert
|    :c=0    counter-mask in range [0-255]
|    :u=0    monitor at user level
|    :k=0    monitor at kernel level
```

PAPI-V

- `virtualized` and `virtual_vendor_string` fields in `hw_info_t`
- VMware component — pseudo-perf counters and vmGuestLib
- Stealtime component — lets you know if your VM was scheduled out
- Virtualized counters work in new KVM and new VMware

CPU frequency support

- PAPI makes various assumptions about MHz (specifically, in `PAPI_get_real_cycles()`)
- Get it from `/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_mi` rather than `/proc/cpuinfo`
- Add `minimum_mhz` and `maximum_mhz` to `hw_info_t`
- Still no turbo-boost support

Locking

- Can enable POSIX pthread mutexes
- Can use Valgrind race-detection tools

New Architectures/Platforms

Most of the hard work is done by libpfm4.

- FreeBSD support improved
- BlueGene/Q support (really 4.4)
- Intel SandyBridge, Interlagos support still new
- ARM Cortex A8, A9 also still new
- Intel Ivy Bridge
- Intel Cedar View Atoms

Updated Components

- coretemp — temperatures, voltages, fan readings
- cuda — NVIDIA GPU events
- example — simple test events
- infiniband — infiniband stats
- lmsensors — temperature/fan readings
- lustre — info from the lustre filesystem
- mx — myrinet stats
- net — generic Linux network stats

New Components

- appio — I/O stats
- bgpm — various BG/Q modules
- nvml — power readings for NVIDIA cards
- rapl — power/energy estimates for Intel SandyBridge
- stealtime — stealtime from inside of KVM
- vmware — VM stats, pseudo-performance counters

New Components for BG/Q

1. Processor Unit — A2 Core: 24 counters / 269 events
2. L2 Unit — 6 counters per L2 Memory Slice / 32 events
3. I/O Unit — 43 counters
4. Network Unit — 5D-Torus network, 66 counters / 31 events
5. Compute Node Kernel Unit — software counters collected by the kernel / 29 events

RAPL Component

- **R**unning **A**verage **P**ower **L**imit
- Part of an infrastructure to allow setting custom per-package hardware enforced power limits
- User Accessible Energy/Power readings are a bonus feature of the interface

How RAPL Works

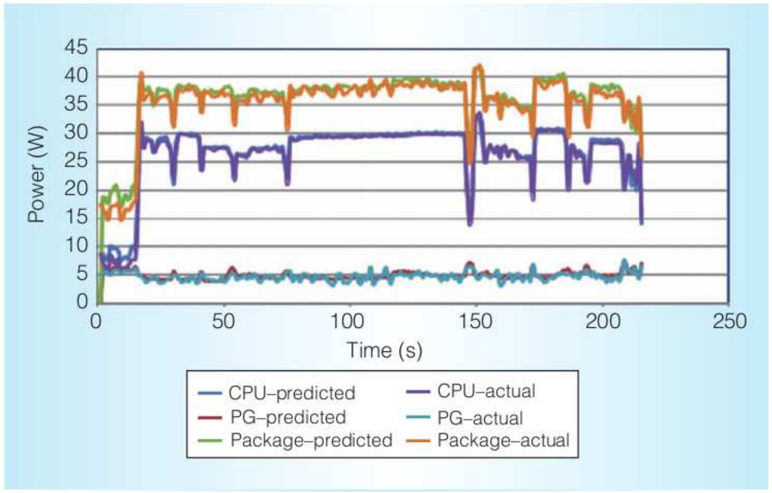
- RAPL is *not* an analog power meter
- RAPL uses a software power model, running on a helper controller on the main chip package
- Energy is estimated using various hardware performance counters, temperature, leakage models and I/O models
- The model is used for CPU throttling and turbo-boost, but the values are also exposed to users via a model-specific register (MSR)

Available RAPL Readings

- `PACKAGE_ENERGY`: total energy used by entire package
- `PP0_ENERGY`: energy used by "power plane 0" which includes all cores and caches
- `PP1_ENERGY`: on original Sandybridge this includes the on-chip Intel GPU
- `DRAM_ENERGY`: on Sandybridge EP this measures DRAM energy usage. It is unclear whether this is just the interface or if it includes all power used by all the DIMMs too

RAPL Measurement Accuracy

- Intel Documentation indicates Energy readings are updated roughly every millisecond (1kHz)
- Rotem at al. show results match actual hardware



Rotem et al. (IEEE Micro, Mar/Apr 2012)

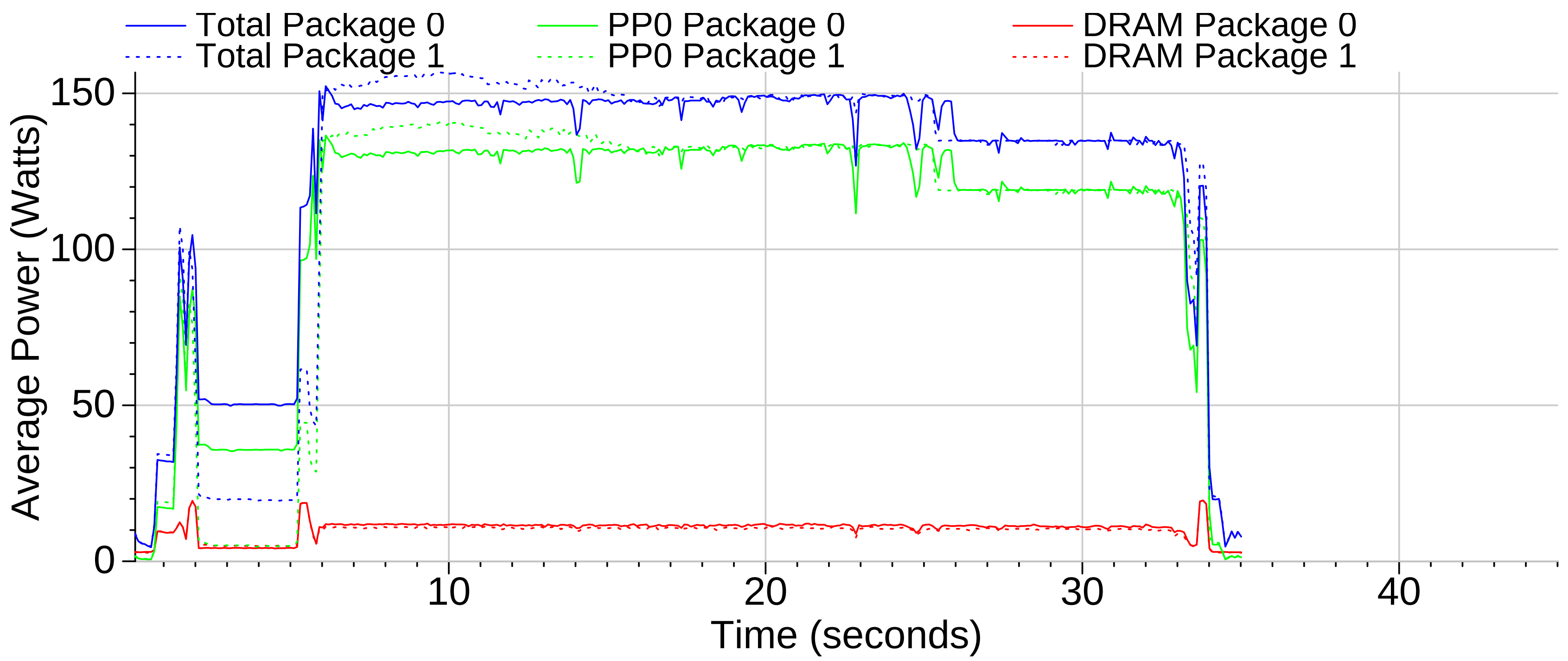
RAPL Accuracy, Continued

- The hardware also reports minimum measurement quanta. This can vary among processor releases. On our Sandybridge EP machine all Energy measurements are in multiples of 15.2nJ
- Power and Energy can vary between identical packages on a system, even when running identical workloads. It is unclear whether this is due to process variation during manufacturing or else a calibration issue.

RAPL PAPI Interface

- Access to RAPL data requires reading a CPU MSR register. This requires operating system support
- Linux currently has no driver and likely won't for the near future
- Linux does support an "MSR" driver. Given proper read permissions, MSRs can be accessed via `/dev/cpu/*/msr`
- PAPI uses the "MSR" driver to gather RAPL values

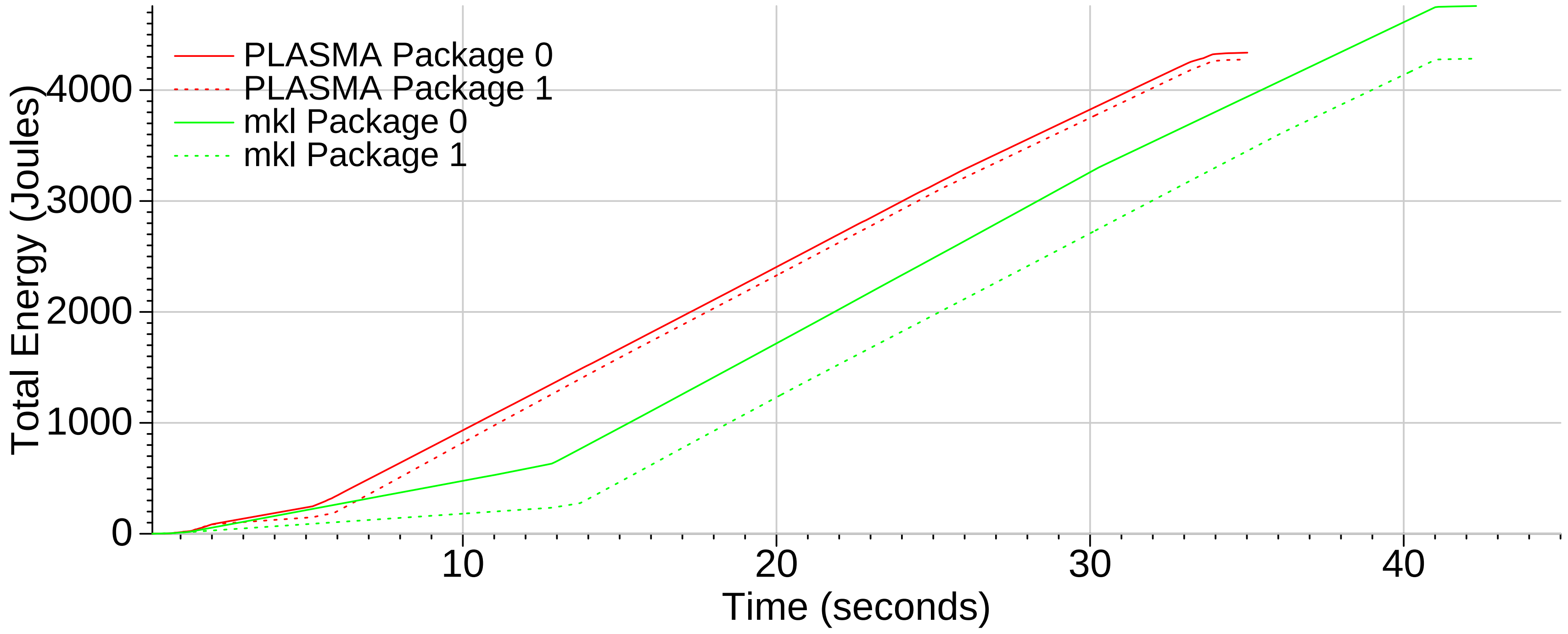
RAPL Power Plot



PLASMA Cholesky Factorization N=30,000 threads=16

Measured on SandyBridge EP

RAPL Energy Plot



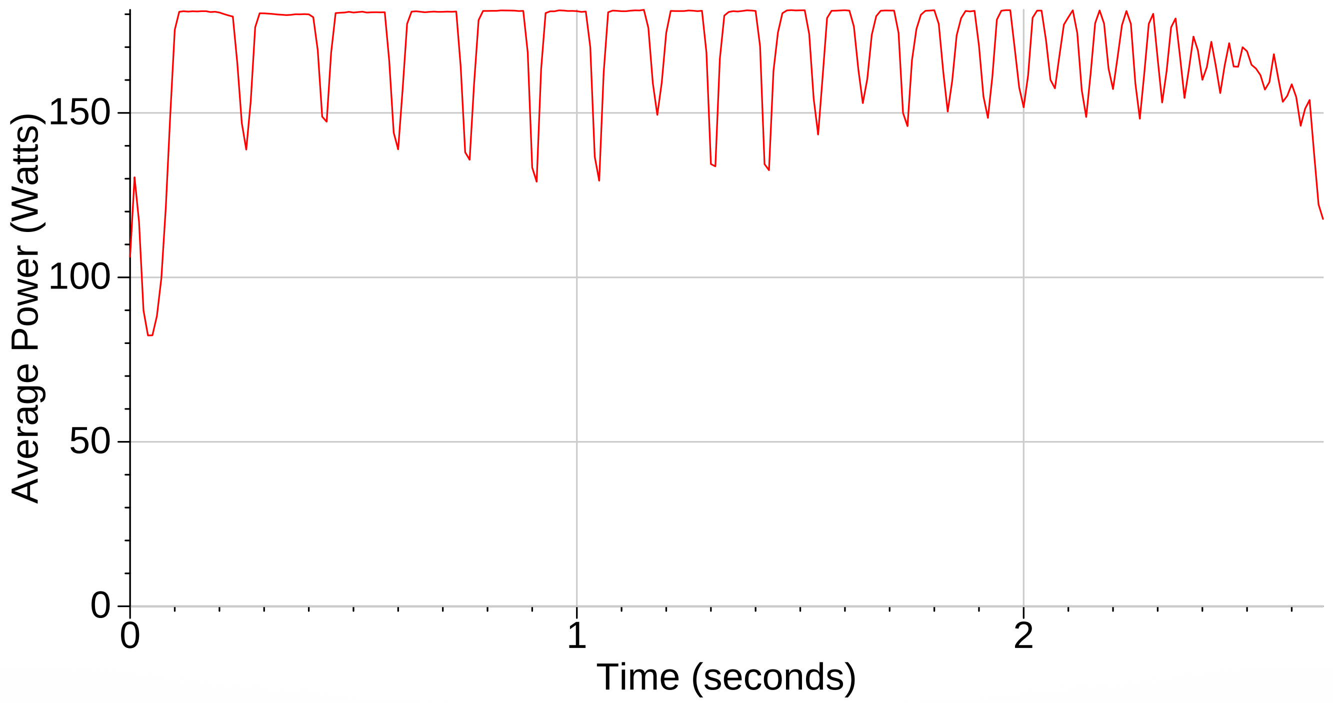
Cholesky Factorization N=30,000 threads=16

Measured on SandyBridge EP

NVML

- Recent NVIDIA GPUs support reading power via the NVIDIA Management Library (**NVML**)
- On Fermi C2075 GPUs it has milliwatt resolution within ±5W and is updated at roughly 60Hz
- The power reported is that for the entire board, including GPU and memory

NVML Power Graph



MAGMA LU 10,000, Nvidia Fermi C2075

Future Work

- New perf_event features: rdpmc, uncore
- Finer-grained user/kernel support
- Per-process vs system-wide measurements
- AMD fam15h Application Power Management
- SNB/IvyBridge FP_OPS events
- Improved attribute enumeration
- Better error message if privileged event
- Enhanced sampling interfaces

Questions