New Developments for PAPI 5.6+

Anthony Danalis, Heike Jagode, and Jack Dongarra
University of Tennessee, Knoxville

Vince Weaver and Yan Liu
University of Maine

INTRODUCTION

- **PAPI provides a consistent interface (and methodology) for hardware performance counters, found across a compute system: i. e., CPUs, GPUs, on- and off-chip memory, interconnects, I/O system, file system, energy/power, etc.**
- **PAPI enables software engineers to see, in near real time, the relationship between software performance and hardware events across the entire compute system.**

SUPPORTED ARCHITECTURES

AMD	ARM Cortex A8, A9, A15, ARM64	CRAY Gemini and Aries interconnect, power	IBM Blue Gene Series, Q: 5-D Torus, I/O System, EMON power, energy	IBM Power Series	intel Westmore, Sandy/Ivy Bridge, Haswell, Broadwell, Skylake(-X), Kaby Lake	intel KNC, Knights Landing including power/energy
intel RAPL (power/energy), power capping	INFINIBAND	Lustre	NVIDIA Tesla, Kepler: CUDA support for multiple GPUs; PC Sampling	NVIDIA NVML	vmware Virtual Environment	KVM Virtual Environment

3RD PARTY TOOLS APPLYING PAPI

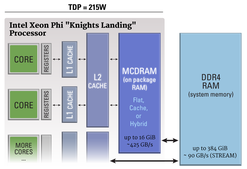
PaRSEC UTK http://icl.utk.edu/parsec/	HPCToolkit Rice University http://hpctoolkit.org/	Score-P http://score-p.org/	TAU University of Oregon http://www.cs.uoregon.edu/research/tau/	Scalasca FZ Juelich, TU Darmstadt http://scalasca.org/	Vampir TU Dresden http://www.vampir.eu/	PerfSuite NCSA http://perfsuite.ncsa.uiuc.edu/	Open|Speedshop SGI http://oss.sgi.com/projects/openspeedshop/	SvPablo RENCI at UNC http://www.renci.org/research/pablo/	ompP UTK http://www.ompp-tool.com/

PART 1: PAPI for Power-Aware Computing

Heike Jagode, Azzam Haidar, Phil Vaccaro, Asim YarKhan, and Stan Tomov

Intel Knights Landing: FLAT mode

- Entire MCDRAM is used as addressable memory → memory allocations are treated similarly to DDR4 memory allocations



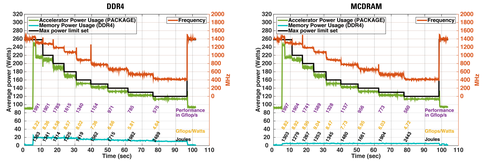
PAPI for power-aware computing

- We use PAPI's latest powercap component for measurement, control, and performance analysis.
  - In the past, PAPI power components supported only reading power information.
  - New component exposes RAPL functionality to allow users to read and write power.
- Study numerical building blocks of varying computational intensity.
- Use PAPI powercap component to detect power optimization opportunities.
- Cap the power on the architecture to reduce power usage while keeping the execution time constant → energy savings.

Power awareness for Level-3 BLAS: DGEMM

68 cores KNL, Peak DP = 2,662 Gflop/s Bandwidth MCDRAM ~425 GB/s DDR4 ~90 GB/s

- DGEMM is run repeatedly for a fixed matrix size of 18K per iteration.
- For each iteration, we dropped the power in 10 or 20 Watts steps.
- Powercap achieves its power limiting though DVFS scaling. See frequency value, which — as it drops — corresponds to a decrease in power.



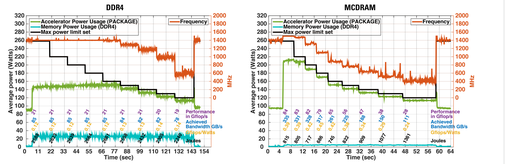
Lesson for DGEMM-type operations (compute intensive):

- Both data allocation schemes provides roughly the same performance. This is caused by the high ratio of FLOPs to bytes in the compute-intensive routines, which allows for as much cache reuse as possible while hiding the data transfer between memory and cache levels.
- Performance drops with decreasing power caps, which is expected for compute-intensive kernels.

Power awareness for Level-3 BLAS: DGEMV

68 cores KNL, Peak DP = 2,662 Gflop/s Bandwidth MCDRAM ~425 GB/s DDR4 ~90 GB/s

- DGEMV is run repeatedly for a fixed matrix size of 18K per iteration.
- For each iteration, we dropped the power in 10 or 20 Watts steps.
- Frequency is not affected until the power caps kick in.



Lesson for DGEMV-type operations (memory bound):

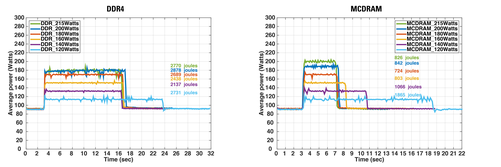
- Performance varies between the two storage options: Data allocations in MCDRAM results in much higher performance for memory-bound kernels (BW for MCDRAM is 425GB/s vs. 90 GB/s for DDR4).
- Dgemv is bound by the reading of matrix A. Depending on its storage location, performance and power may be affected.

DDR4
- We see lower package-power-usage but higher DDR4-power-usage.
  - Capping at 140 W improves energy efficiency by ~10%.
  - Capping at 120 W results in a 17% energy savings with some performance degradation (8%).

MCDRAM
- Capping at 180 Watts renders in ~15% of energy saving with a negligible performance penalty (of less than 5%).

Power awareness for applications: Jacobi

- Solving Helmholtz equation with *Jacobi Iterative Method* (grid size 12,800 x 12,800, 2D 5-point stencil)
  → requires multiple memory accesses per update
  → high-memory bandwidth and low computational intensity

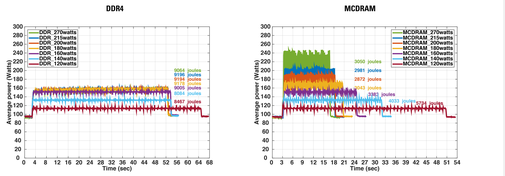


Lesson for Jacobi iteration:

- Computation is about 3.5X faster when the data is allocated in MCDRAM compared to DDR4.
- MCDRAM: Capping at 170 Watts improves energy efficiency by ~14% without any loss in time to solution.
- DDR4: capping at 135 Watts improves energy efficiency by ~25% without any loss in time to solution.

Power awareness for applications: Lattice-Boltzman

- Lattice-Boltzmann simulation of computational fluid dynamics (from SPEC 2006 benchmark)
  → high-memory bandwidth and low computational intensity



Lesson for Lattice-Boltzmann:

- Computation is about 3.6X faster when the data is allocated in MCDRAM compared to DDR4.
- For MCDRAM, capping at 190 results in energy saving of ~6% without loss of time to solution.
- For DDR4, capping at 130 Watts improves energy efficiency by ~12%.

PART 2: PAPI's Counter Inspection Toolkit (CIT)

Anthony Danalis, Heike Jagode, and Hanumanthrayappa

Key Concepts

Goal:

- Create a set of micro-benchmarks for illustrating details in hardware events and how they relate to the behavior of the micro-architecture

Target audience:

- Performance-conscious application developers
- PAPI developers working on new architectures (think preset events)
- Developers interested in validating hardware-event counters

Highlighting non-obvious behavior

QUESTION

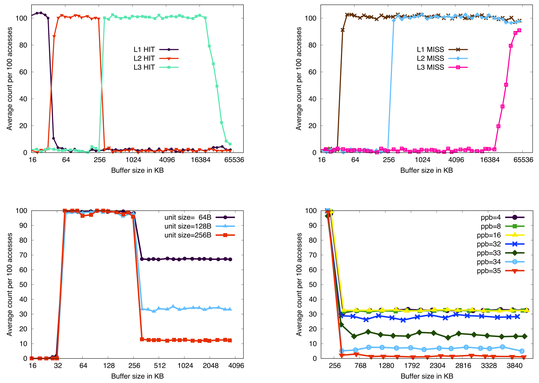
How many branch instructions will these codes execute per iteration?

The expectation is that both codes will execute 2.

The measured answer is 2 for the first and 2.5 for the second!

Can you guess why?

Native Event Characterization



PART 3: Modernizing PAPI Infrastructure

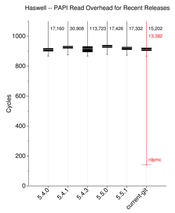
Vince Weaver and Yan Liu

Improved PAPI Test Infrastructure

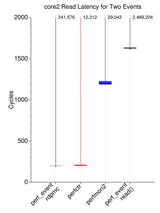
- The existing PAPI testsuite is used to test the correctness of PAPI before release.
- The hardware and operating systems used by PAPI are always changing, and some of the existing tests were outdated or gave false negatives.
- Existing tests were checked to ensure accurate results on modern hardware.
- New counter validation tests were created which should provide a sanity check when bringing up support for a new processor architecture.

Low-Overhead `PAPI_read()` support

- Traditionally `PAPI_read()` counter reads went through the standard Linux `read()` system call, which can be slow (around 1,000 cycles).
- x86 hardware supports a userspace `rdpmc()` instruction that bypasses the kernel and requires 200 cycles (a 5× speedup).
- Various bugs in the Linux kernel around this interface were found and fixed so that `rdpmc()` can be enabled by default.



Boxplot showing read latency for various versions of PAPI and the large improvement by using `rdpmc`



Comparison of historical performance counter interfaces (perfmon2, perfctr) showing that `perf_event rdpmc` matches even the best historical interface.

Enhanced Sampling Interface

- PAPI currently has a limited counter sampling interface that only allows gathering the instruction pointer at regular intervals
- Modern processors support much richer sampling information, including the cause of cache misses, where in the cache hierarchy the miss happened, and the cycles taken.
- We extend the PAPI sampling interface to provide this additional sampling information.

ACKNOWLEDGMENTS This material is based upon work supported in part by the National Science Foundation NSF under awards No. 1450429. A portion of this research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.