Component PAPI: Performance Measurement Beyond the CPU

Presented by

Jack Dongarra
Heike Jagode
Shirley Moore
Dan Terpstra

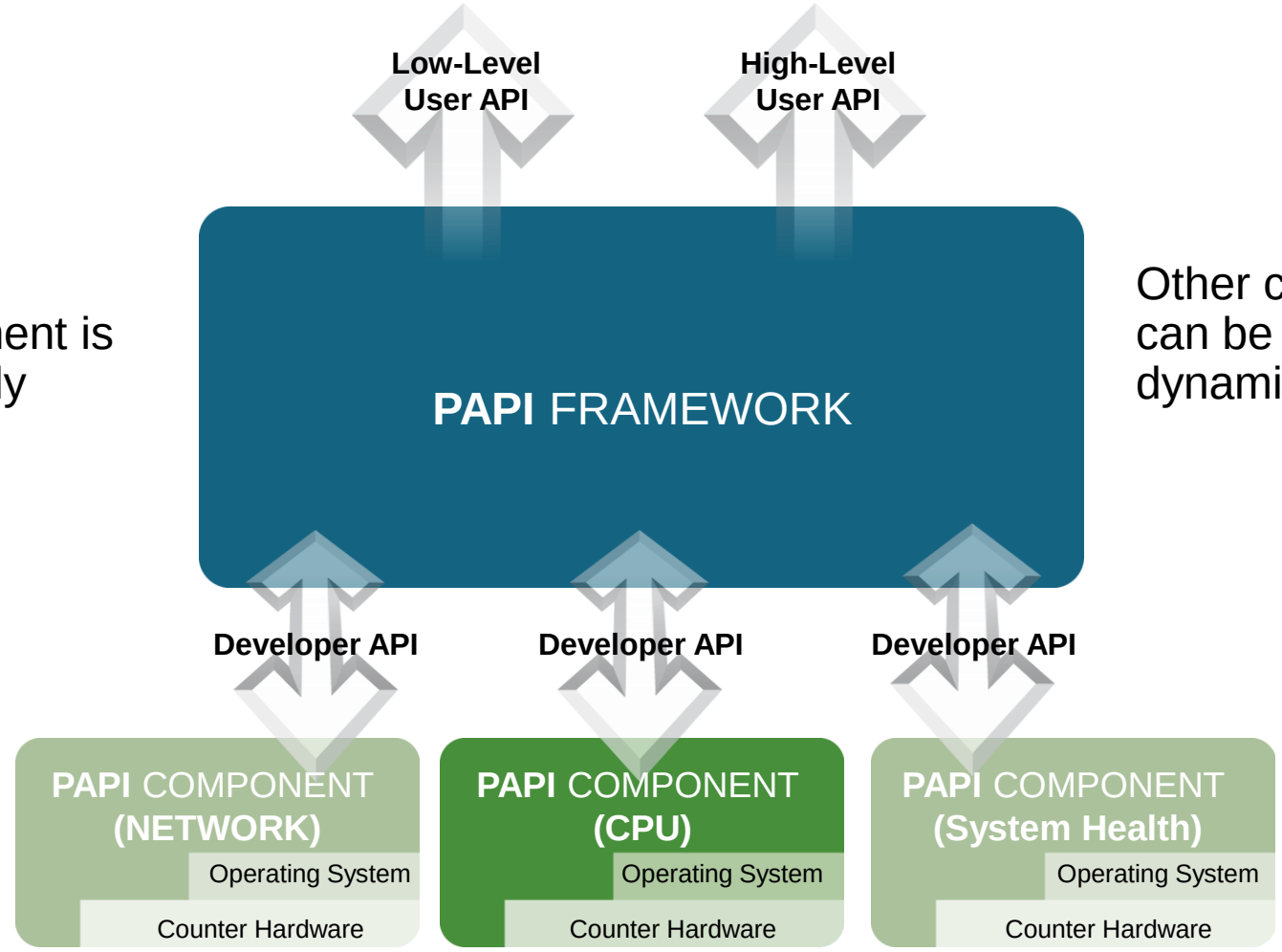
University of Tennessee
Oak Ridge National Laboratory

Introduction

- **PAPI** has provided a consistent programming interface for performance counter hardware
- **PAPI-C** extends that interface to multiple performance counter domains
- Interesting performance phenomenon can be measured throughout high performance computing systems:
  - File systems
  - Network fabrics
  - GPGPUs
- **PAPI-C** can provide the interface between user level performance tools and low level performance measurements
- Third parties can develop **PAPI-C** components for specialized hardware

Component PAPI

A CPU component is generally present



Other components can be added dynamically

File system components: Lustre

- **Measures data collected in: `/proc/…/stats` and: `/proc/…/read_ahead_stats`**

```
Hits                                    631592284
misses                                  9467662
readpage not consecutive                931757
miss inside window                      81301
failed grab_cache_page                  5621647
failed lock match                       2135855
read but discarded                      2089608
zero size window                        6136494
read-ahead to EOF                       160554
hit max r-a issue                       25610
```

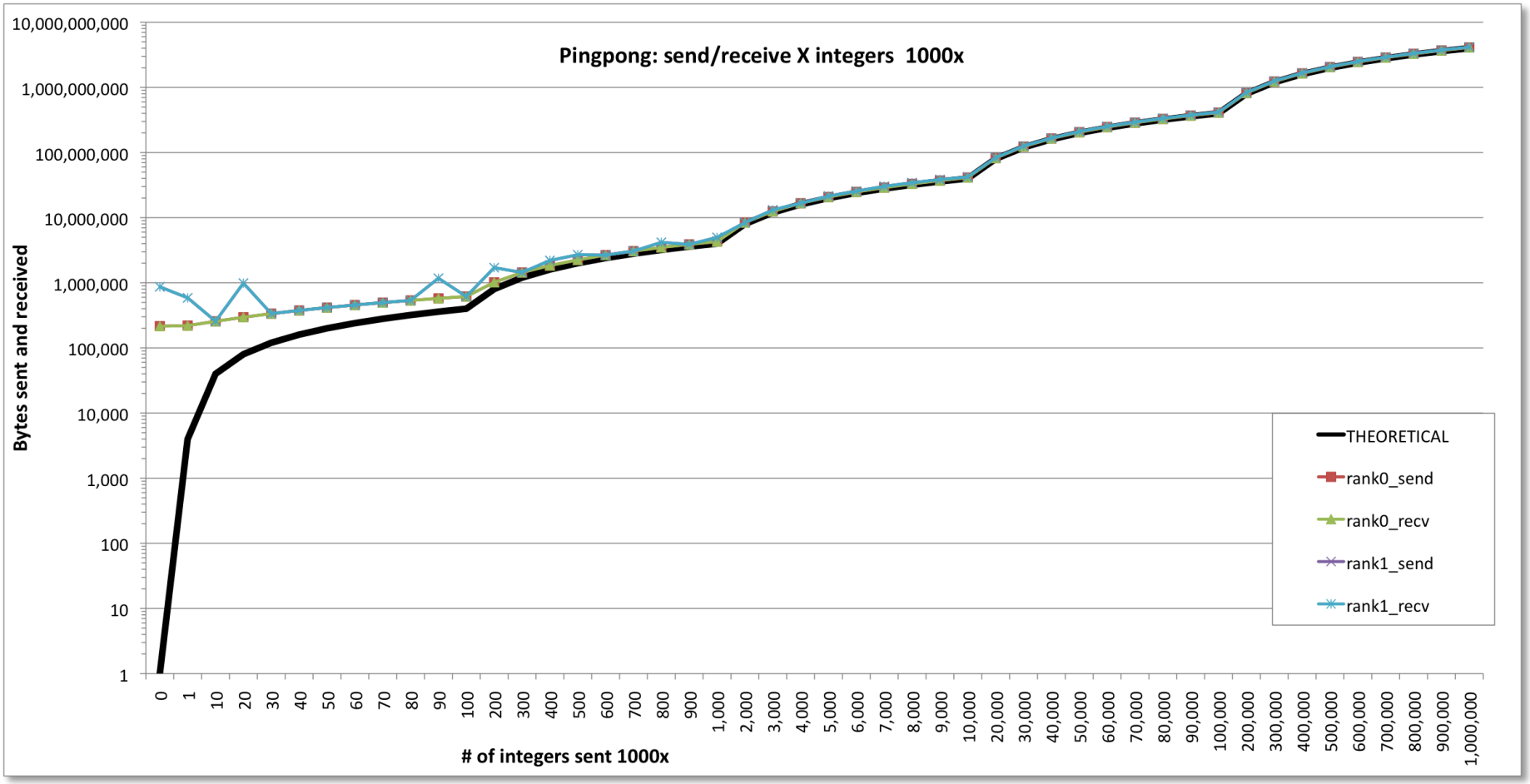
- **Snippet of available native events for Lustre:**

```
0x44000002  fastfs_llread           | bytes read on this lustre client
0x44000003  fastfs_llwrite          | bytes written on this lustre client
0x44000004  fastfs_wrong_readahead  | bytes read but discarded due to
  readahead
0x44000005  work_llread             | bytes read on this lustre client
0x44000006  work_llwrite            | bytes written on this lustre client
0x44000007  work_wrong_readahead    | bytes read but discarded due to
  readahead
```

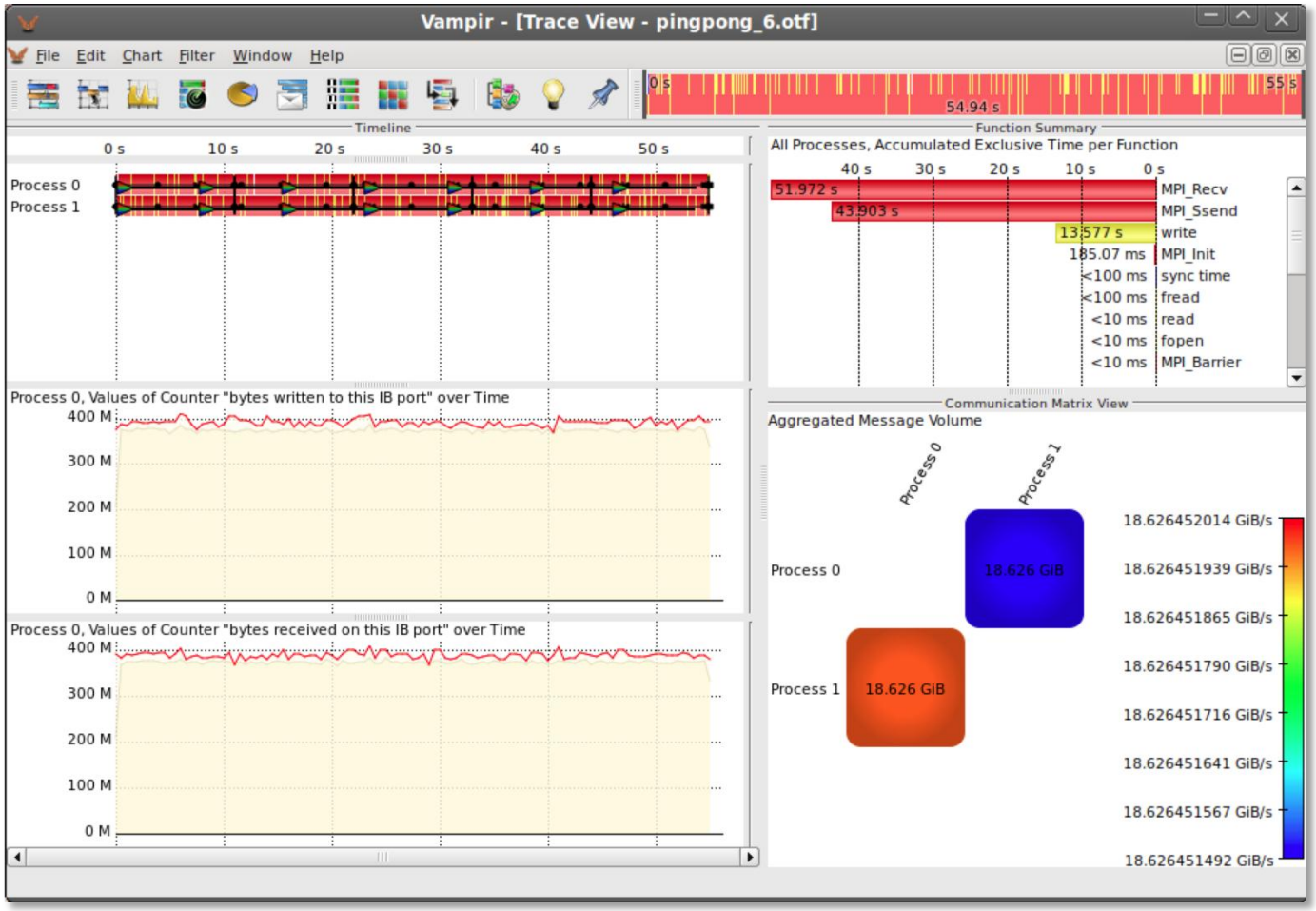
Network components: InfiniBand

- Measures everything that is provided by the libibmad:
- Errors, bytes, packets, local IDs (LID), global IDs (GID), etc.
- ibmad library provides low-layer IB functions for use by the IB diagnostic and management programs, including MAD, SA, SMP, and other basic IB functions
- Snippet of available native events on a machine with 2 IB devices, mthca0 and mthca1:
- …
- 0x44000000 mthca0_1_recv| bytes received on this IB port
- 0x44000001 mthca0_1_send | bytes written to this IB port
- 0x44000002 mthca1_1_recv| bytes received on this IB port
- 0x44000003 mthca1_1_send | bytes written to this IB port
- …

InfiniBand component results



InfiniBand events measured over time (via Vampir linked with PAPI)



Run Pingpong 5x: send 1,000,000 integers 1000x (theor: ~19 GB)

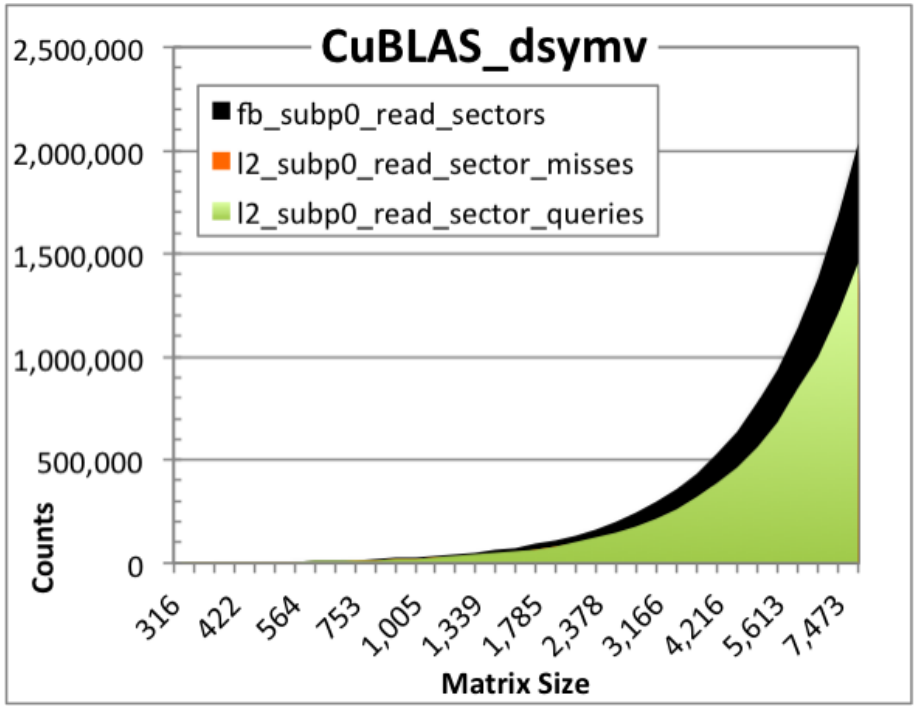
PAPI CUDA Component

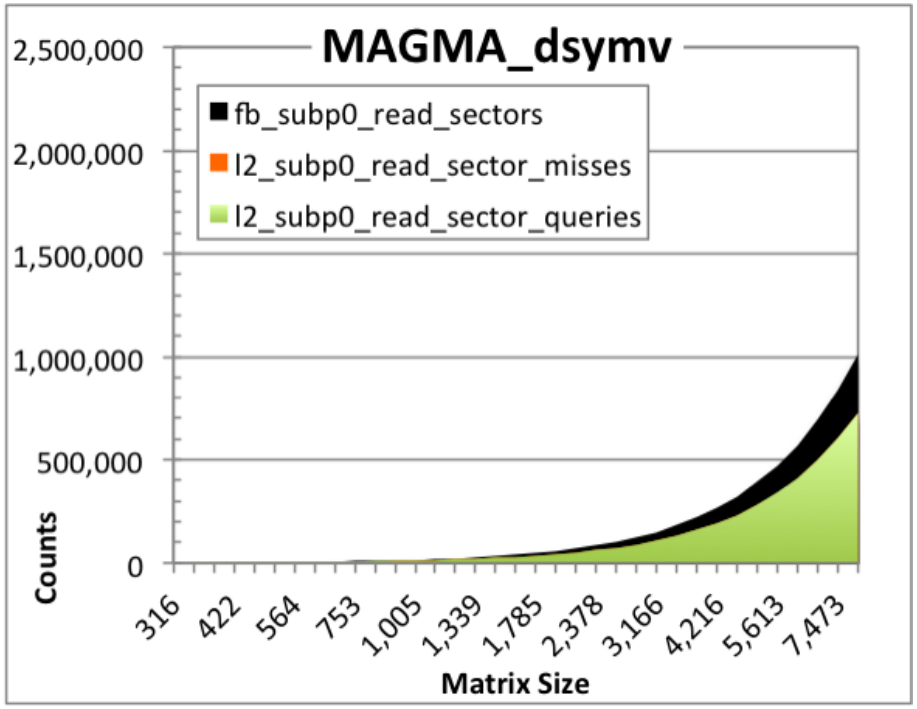
- HW performance counter measurement technology for NVIDIA CUDA platform
- Access to HW counters inside the GPUs
- Based on CUPTI (CUDA Performance Tool Interface) in CUDA 4.0
- In any environment with CUPTI, PAPI CUDA component can provide detailed performance counter info regarding execution of GPU kernel
- Initialization, device management and context management is enabled by CUDA driver API
- Domain and event management is enabled by CUPTI
- Name of events is established by the following hierarchy: Component.Device.Domain.Event

MAGMA versus CUBLAS: SYMV

- Symmetry exploitation more challenging
  → computation would involve irregular data access
- How well is symmetry exploited?
  What about bank conflicts and branching?
- SYMV implementation: Access each element of lower (or upper) triangular part of the matrix only once → N2/2 element reads (vs. N2)
- Since SYMV is memory-bound, exploiting symmetry is expected to be twice as fast
- To accomplish this, additional global memory workspace is used to store intermediate results
- We ran experiments using CUBLAS_dsymv (general) and MAGMA_dsymv (exploits symmetry) to observe the effects of cache behavior on Tesla S2050 (Fermi) GPU

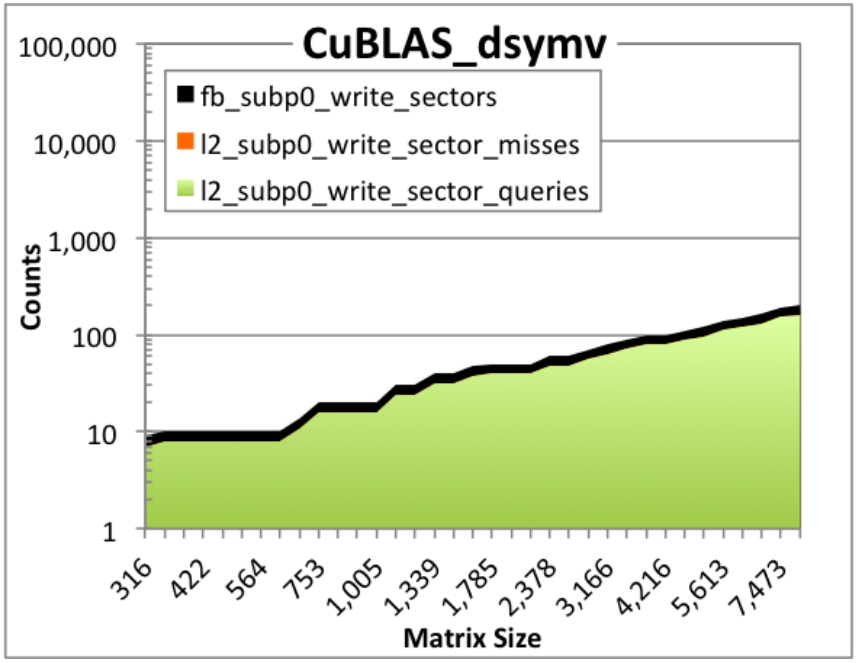
CUDA performance counters for read behavior as measured by PAPI

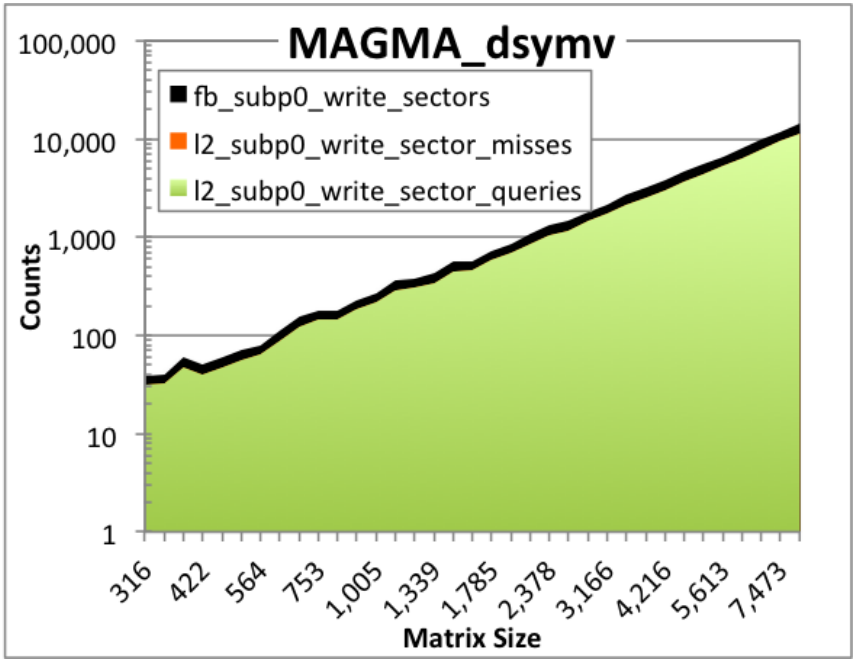




- # of read requests from L1 to L2 (green), which is equal to # of read misses in L2 (orange); number of read requests from L2 to DRAM (black) for CUBLAS_dsymv (left) and MAGMA_dsymv (right)

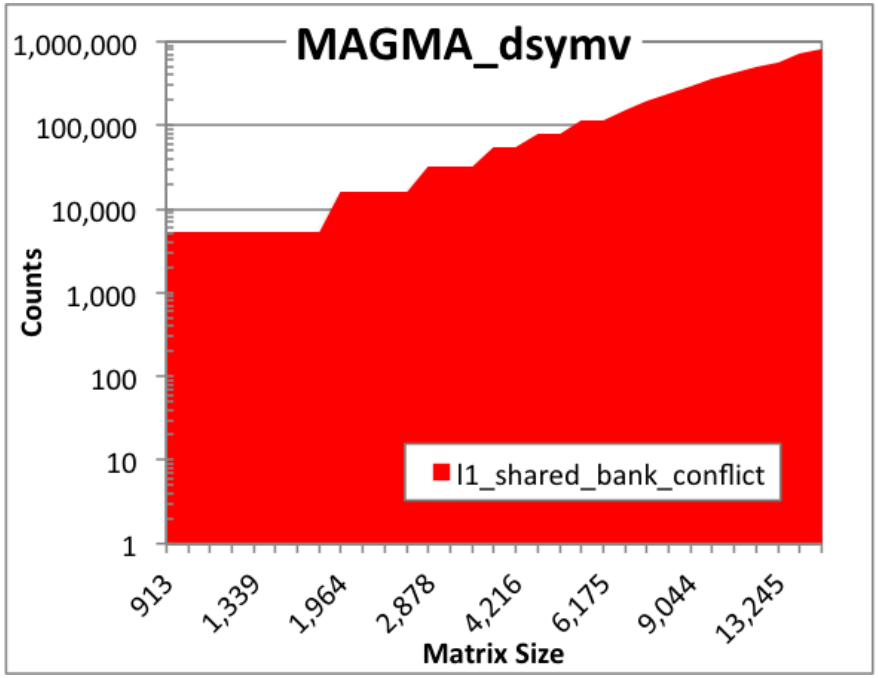
CUDA performance counters for write behavior as measured by PAPI

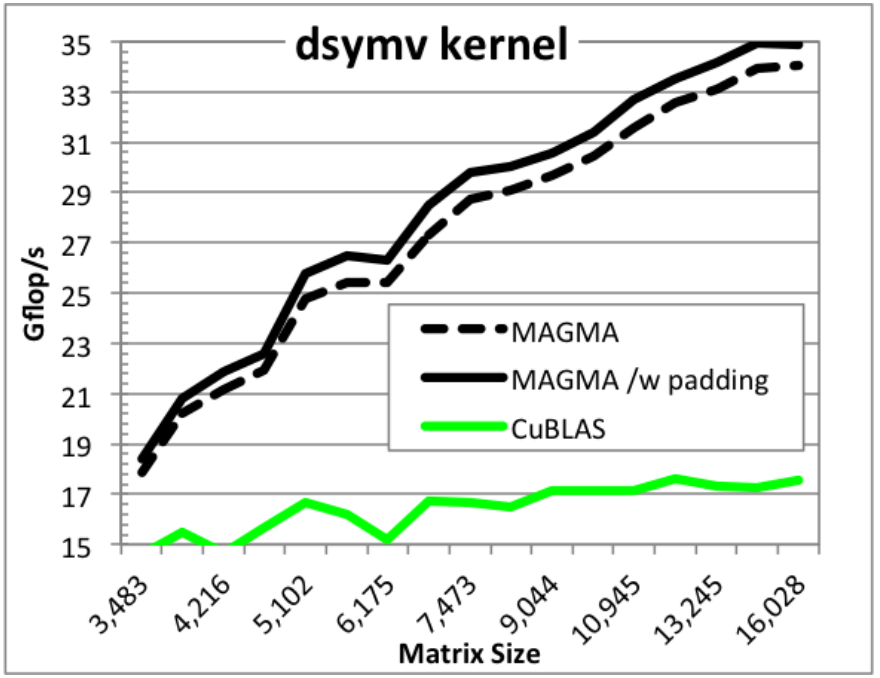




- # of write requests from L1 to L2 (green), which is equal to # of write misses in L2 (orange); # of write requests from L2 to DRAM (black) for CUBLAS_dsymv (left) and MAGMA_dsymv (right)

CUDA performance counter for L1 behavior as measured by PAPI





- **# of L1 shared bank conflicts in the MAGMA_dsymv kernel for medium to large matrix sizes (left); Performance of MAGMA_dsymv kernel with and without shared bank conflicts (right)**

Conclusion and future directions

- Component PAPI provides performance measurement beyond the CPU
- Increasing CPU densities places greater importance on
  - Thermal health and management
  - Power consumption
- Higher processor counts make communication metrics more critical (bandwidth, latency, how many bytes transferred)
- Heterogeneity requires performance measurement in multiple domains
- Third parties can develop and contribute specialized components
- User-level performance tools can access multiple components with a common interface

Contact

Dan Terpstra

UTK Innovative Computing Laboratory
terpstra@eecs.utk.edu

For more information

http://icl.cs.utk.edu/papi/

- **Software and documentation**
- **Component interface details**
- **Reference materials**
- **Papers and presentations**
- **Third-party tools**
- **Mailing lists and User Forum**

Acknowledgments

This work used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.
These resources were made available via the Performance Evaluation and Analysis Consortium End Station, a Department of Energy INCITE project.
This work was also supported in part by the U.S. Department of Energy Office of Science under contract DE-FC02-06ER25761; by the National Science Foundation under Grant No. 0910899, as well as Software Development for Cyberinfrastructure (SDCI) Grant No. NSF OCI-0722072 Subcontract No. 207401; and by the Department of Defense, using resources at the Extreme Scale Systems Center.