

PAPI: Performance API

Introduction & Overview

Anara Kozhokanova – RWTH Aachen University, Frank Winkler,
Heike Jagode, Anthony Danalis - University of Tennessee

Presented by: Radita Liem (RWTH Aachen University)

What are Hardware Performance Counters?

- For many years, hardware engineers have designed in specialized registers to measure the performance of various aspects of a microprocessor.
- HW performance counters provide application developers with valuable information about code sections that can be improved.
- **Hardware performance counters can provide insight into:**
 - Whole program timing
 - Cache behaviors
 - Branch behaviors
 - Memory and resource contention and access patterns
 - Pipeline stalls
 - Floating point efficiency
 - Instructions per cycle
 - Subroutine resolution
 - Process or thread attribution

What is PAPI?

- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i. e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O system, File System, Energy/Power, etc.
- PAPI (**P**erformance **A**pplication **P**rogramming **I**nterface) enables software engineers to see, in near real time, the relation between SW performance and HW events across the entire compute system

PAPI: Supported Architectures

- AMD **up to Zeppelin Zen, power for Fam17h**
- AMD **GPUs Vega, power, temperature, fan**
- ARM Cortex A8, A9, A15, ARM64
- CRAY: Gemini and Aries interconnects, power/energy
- IBM Blue Gene Series, Q: 5D-Torus, I/O system, EMON power/energy
- IBM Power Series, **PCP for POWER9-uncore**
- Intel Sandy|Ivy Bridge, Haswell, Broadwell, Skylake, **Kabylake, Cascadelake**, KNC, KNL, KNM
- Intel RAPL (power/energy), **power capping**
- InfiniBand
- Lustre FS
- NVIDIA Tesla, Kepler, Maxwell, Pascal, **Volta**: support for multiple GPUs
- **NVIDIA: support for NVLink**
- NVIDIA NVML (power/energy); **power capping**

PAPI Hardware Events

Preset Events (only CPU related)

- Standard set of over 100 events for application performance tuning
- No standardization of the exact definition
- Mapped to either single or linear combinations of native events on each platform
- Use **papi_avail** to see what preset events are available on a given platform

Native Events

- Any event countable by the CPU, GPU, network card, parallel file system or others
- Same interface as for preset events
- Use **papi_native_avail** utility to see all available native events

Use **papi_event_chooser** utility to select a compatible set of events

PAPI Framework: 1999 - 2009

PAPI provides two interfaces to the underlying counter hardware.

A **Low-Level API** manages hardware events (preset and native) in user defined groups called *EventSets*. Meant for experienced application programmers wanting fine-grained measurements.

Applications / 3rd Party Tools

Low-Level API

High-Level API

Graphical and end-user tools provide facile data collection and visualization.

**PAPI
PORTABLE LAYER**

A **High-Level API** provides the ability to record hardware events of instrumented code sections. Meant for programmers wanting simple event measurements.

Developer API

**PAPI Hardware
Specific Layer**

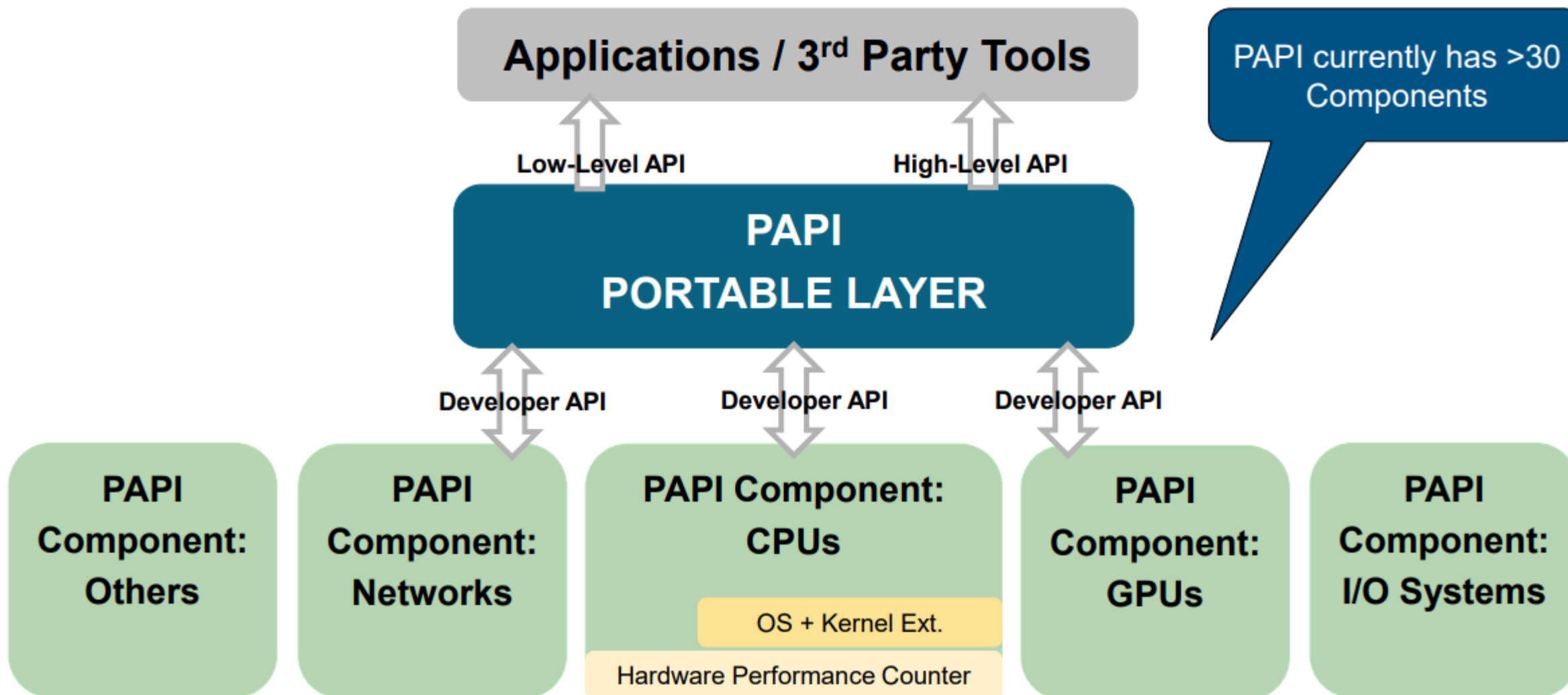
CPUs ONLY

OS + Kernel Ext.

Hardware Performance Counter

Goal: Accessing hardware counters, found on a diverse collection of modern microprocessors, in a portable manner.

PAPI Framework: 2009 - present



PAPI – High-Level Calls

- **PAPI_hl_region_begin (const char *region)**
 - Read events at the beginning of a region (also start counting the events)
- **PAPI_hl_region_end (const char *region)**
 - Read events at the end of a region and store the difference from the beginning
- **PAPI_hl_read (const char *region)**
 - Read events inside a region and store the difference from the beginning
- **PAPI_hl_stop ()**
 - Stop a running high-level event set (optional)

Some events, like temperature or power, must be specified as instantaneous values. In this case, only the value of the read or end region call is stored.

```
% export PAPI_EVENTS="PAPI_TOT_INS,PAPI_TOT_CYC,coretemp::hwmon0:temp2_input=instant"  
% ./<PAPI instrumented binary>
```

<https://bitbucket.org/icl/papi/wiki/PAPI-HL.md>

PAPI – Example High-Level API

```
% export PAPI_EVENTS="PAPI_TOT_INS,PAPI_TOT_CYC"
```

```
#include "papi.h"

int main()
{
    int retval;
    retval = PAPI_hl_region_begin("computation");
    if ( retval != PAPI_OK )
        handle_error(1);

    /* Do some computation here */
    retval = PAPI_hl_region_end("computation");
    if ( retval != PAPI_OK )
        handle_error(1);
}
```

Automatic performance report

```
{
  "computation":{
    "region_count":"1",
    "cycles":"2080863768",
    "PAPI_TOT_INS":"2917520595",
    "PAPI_TOT_CYC":"2064112930" }
}
```

PAPI – High-Level API: Optional Environment Variables

Environment Variable	Description	Type
PAPI_EVENTS	PAPI events to measure	String
PAPI_OUTPUT_DIRECTORY	Path of the measurement directory	Path
PAPI_REPORT	Print report to stdout	-
PAPI_MULTIPLEX	Enable Multiplexing	-
PAPI_HL_VERBOSE	Suppress warnings and info	-
PAPI_DEBUG=HIGHLEVEL	Enable debugging of high-level routines	String

PAPI – Low-Level Calls

- **PAPI_accum** - accumulate and reset hardware events from an event set
- **PAPI_add_event** - add single PAPI preset or native hardware event to an event set
- **PAPI_add_events** - add array of PAPI preset or native hardware events to an event set
- **PAPI_attach** - attach specified event set to a specific process or thread id
- **PAPI_cleanup_eventset** - remove all PAPI events from an event set
- **PAPI_create_eventset** - create a new empty PAPI event set
- **PAPI_destroy_eventset** - deallocates memory associated with an empty PAPI event set
- [...]

Total of **81** functions covering the whole functionality of the PAPI Low-Level interface.

<http://icl.cs.utk.edu/papi/docs/>

PAPI – Example Low-Level API

```
#include "papi.h"

#define NUM_EVENTS 2
int Events[NUM_EVENTS]={ PAPI_FP_OPS, PAPI_TOT_CYC };
int EventSet = PAPI_NULL;
long long values[NUM_EVENTS];

/* Initialize the Library */
retval = PAPI_library_init (PAPI_VER_CURRENT);
/* Allocate space for the new eventset and do setup */
retval = PAPI_create_eventset (&EventSet);
/* Add Flops and total cycles to the eventset */
retval = PAPI_add_events (EventSet, Events, NUM_EVENTS);

/* Start the counters */
retval = PAPI_start (EventSet);

do_work(); /* What we want to monitor*/

/*Stop counters and store results in values */
retval = PAPI_stop (EventSet, values);
```



User has to implement the performance report of the measured values.

PAPI – Rate Calls

- **PAPI_flops_rate**
 - Get Mflops/s (floating point operation rate), real and processor time
- **PAPI_flips_rate**
 - Get Mflips/s (floating point instruction rate), real and processor time
- **PAPI_ipc**
 - Get instructions per cycle, real and processor time
- **PAPI_epc**
 - Get arbitrary events per cycle, real and processor time
- **PAPI_rate_stop**
 - Stop a running event set of a rate function

<https://bitbucket.org/icl/papi/wiki/PAPI-Rates.md>

The first call of a rate function will initialize the PAPI interface (threadsafe), set up the counters to monitor and start the counters. Subsequent calls will read the counters and return values since the latest matching call.

PAPI - Low-Level vs. High-Level API

▪ **Low-Level API**

- Aimed at experienced application programmers and tool developers who require fine-grained measurement and control of the PAPI interface
- Requires to create the necessary event sets for each component

▪ **High-Level API**

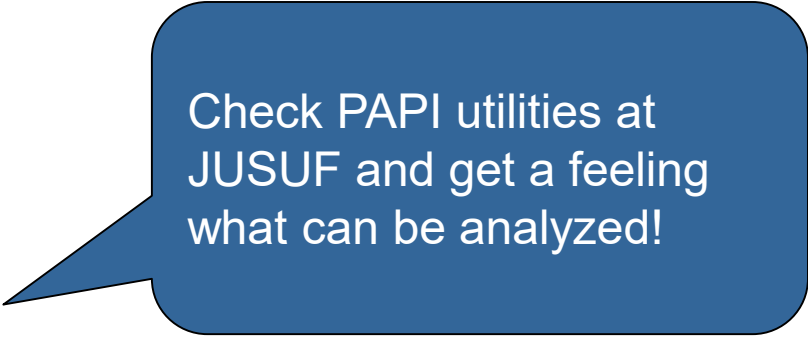
- Simplifies code instrumentation
- Implicit PAPI library initialization (thread-safe)
- No recompilation (set events via env variable)
- Automatic detection of components
- Event checking (availability, combinations)
- Automatic performance report
 - JSON format
 - Derived metrics, e.g. IPC or MFLOPS/s (when using the required raw events)
 - Report Aggregation for parallel programs

3rd Party Tools Applying PAPI

- **Score-P** - <https://www.vi-hps.org/projects/score-p>
- **TAU** (U Oregon) - <http://www.cs.uoregon.edu/research/tau>
- **Scalasca** (FZ Juelich, TU Darmstadt) - <http://scalasca.org>
- **Paraer/Extrae** (BSC) - <https://tools.bsc.es/>
- **Vampir** (GWT-TUD) - <http://www.vampir.eu>
- PaRSEC (UTK) - <http://icl.cs.utk.edu/parsec>
- Caliper (LLNL) - github.com/LLNL/caliper-compiler
- Kokkos (SNL) - <https://github.com/kokkos>
- HPCToolkit (Rice University) - <http://hpctoolkit.org>
- 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)

PAPI Tools

- Available components
 - **papi_component_avail**
- Memory hierarchy
 - **papi_mem_info**
- Costs of PAPI calls
 - **papi_cost**
- Available native/derived/preset counters
 - **papi_avail** , **papi_native_avail**
- Combinable counters
 - **papi_event_chooser**
- Check out to dampen you anticipation
 - **papi_command_line**



Check PAPI utilities at JUSUF and get a feeling what can be analyzed!

```
% source /p/project/training2214/setup.sh  
% module load PAPI/6.0.0.1
```

PAPI Tools - papi_component_avail

```
% papi_component_avail
Available components and hardware information.
-----
PAPI version           : 6.0.0.1
Operating system       : Linux 4.18.0-348.23.1.el8_5.x86_64
Vendor string and code : AuthenticAMD (2, 0x2)
Model string and code  : AMD EPYC 7742 64-Core Processor (49, 0x31)
[...]
Total cores            : 256
SMT threads per core  : 2
[...]
-----
Compiled-in components:
Name:  perf_event           Linux perf_event CPU counters
Name:  perf_event_uncore   Linux perf_event CPU uncore and northbridge
      \-> Disabled: No uncore PMUs or events found
Name:  rapl                 Linux RAPL energy measurements
      \-> Disabled: Can't open fd for cpu0: Permission denied
[...]
```

PAPI Tools - papi_mem_info

```
% papi_mem_info
```

```
Memory Cache and TLB Hierarchy Information.
```

```
-----  
TLB Information.
```

```
There may be multiple descriptors for each level of TLB  
if multiple page sizes are supported.
```

```
L1 Instruction TLB:
```

```
Page Size:          4096 KB  
Number of Entries:   32  
Associativity:       Full
```

```
L1 Data TLB:
```

```
Page Size:          4096 KB  
Number of Entries:   32  
Associativity:       Full
```

```
L1 Instruction TLB:
```

```
Page Size:          2048 KB
```

```
[..]
```


PAPI Tools - papi_cost

```
% papi_cost -h
```

```
This is the PAPI cost program.
```

```
It computes min / max / mean / std. deviation for PAPI start/stop pairs; for PAPI reads, and for PAPI_accums. Usage:
```

```
cost [options] [parameters]  
cost TESTS_QUIET
```

```
Options:
```

```
-b BINS      set the number of bins for the graphical distribution of costs. Default: 100  
-d           show a graphical distribution of costs  
-h           print this help message  
-p           print 25/50/75th percentile results for making boxplots  
-s           show number of iterations above the first 10 std deviations  
-t THRESHOLD set the threshold for the number of iterations. Default: 1,000,000
```

PAPI Tools - papi_cost

```
% papi_cost
```

```
Cost of execution for PAPI start/stop, read and accum.
```

```
This test takes a while. Please be patient...
```

```
Performing loop latency test...
```

```
Total cost for loop latency over 1000000 iterations
```

```
min cycles   : 22
```

```
max cycles   : 12938
```

```
mean cycles  : 24.873197
```

```
std deviation: 43.318341
```

```
Performing start/stop test...
```

```
Total cost for PAPI_start/stop (2 counters) over 1000000 iterations
```

```
min cycles   : 8662
```

```
max cycles   : 900135
```

```
mean cycles  : 8960.007616
```

```
std deviation: 1501.515969
```

```
[...]
```

PAPI Tools - papi_avail

```
% papi_avail -h
```

```
This is the PAPI avail program.
```

```
It provides availability and details about PAPI Presets and User-defined Events.
```

```
PAPI Preset Event filters can be combined in a logical OR.
```

```
Usage: papi_avail [options]
```

```
Options:
```

```
General command options:
```

```
-h, --help          Print this help message
-a, --avail         Display only available PAPI preset and user defined events
-c, --check         Display only available PAPI preset and user defined events after an availability check
-d, --detail        Display detailed information about events
-e EVENTNAME        Display detail information about specified event
```

```
Event filtering options:
```

```
--br                Display branch related PAPI preset events
--cache             Display cache related PAPI preset events
--cnd               Display conditional PAPI preset events
--fp                Display Floating Point related PAPI preset events
```

```
[..]
```

PAPI Tools - papi_avail

```
% papi_avail
```

```
Available PAPI preset and user defined events plus hardware information.
```

```
-----  
PAPI version           : 6.0.0.1  
Operating system       : Linux 4.18.0-348.23.1.el8_5.x86_64  
Vendor string and code : AuthenticAMD (2, 0x2)  
Model string and code  : AMD EPYC 7742 64-Core Processor (49, 0x31)  
CPU revision           : 0.000000  
CPUID                  : Family/Model/Stepping 23/49/0, 0x17/0x31/0x00  
CPU Max MHz            : 2250  
CPU Min MHz            : 1500  
[..]
```

```
=====  
PAPI Preset Events  
=====
```

Name	Code	Avail	Deriv	Description (Note)
PAPI_L1_DCM	0x80000000	No	No	Level 1 data cache misses
PAPI_L1_ICM	0x80000001	No	No	Level 1 instruction cache misses
PAPI_L2_DCM	0x80000002	No	No	Level 2 data cache misses
[..]				

PAPI Tools - papi_native_avail

```
% papi_native_avail
```

```
Available native events and hardware information.
```

```
-----  
PAPI version           : 6.0.0.1  
Operating system       : Linux 4.18.0-348.23.1.el8_5.x86_64  
Vendor string and code : AuthenticAMD (2, 0x2)  
Model string and code  : AMD EPYC 7742 64-Core Processor (49, 0x31)  
CPU revision           : 0.000000  
CPUID                  : Family/Model/Stepping 23/49/0, 0x17/0x31/0x00  
CPU Max MHz            : 2250  
CPU Min MHz            : 1500  
[..]
```

```
=====  
Native Events in Component: perf_event  
=====
```

```
| perf::PERF_COUNT_HW_CPU_CYCLES |  
|         PERF_COUNT_HW_CPU_CYCLES |  
|         :u=0 |  
|         monitor at user level |  
[..]
```


PAPI Tools - papi_event_chooser

```
% papi_event_chooser
```

```
Usage: papi_event_chooser NATIVE|PRESET evt1 evt2 ...
```

```
% papi_event_chooser PRESET PAPI_TOT_INS
```

```
Event Chooser: Available events which can be added with given events.
```

```
-----  
PAPI version           : 6.0.0.1  
Operating system       : Linux 4.18.0-348.23.1.el8_5.x86_64  
Vendor string and code : AuthenticAMD (2, 0x2)  
Model string and code  : AMD EPYC 7742 64-Core Processor (49, 0x31)  
[..]  
-----
```

Name	Code	Deriv	Description (Note)
PAPI_TLB_DM	0x80000014	No	Data translation lookaside buffer misses
PAPI_TLB_IM	0x80000015	Yes	Instruction translation lookaside buffer misses
PAPI_BR_TKN	0x8000002c	No	Conditional branch instructions taken
PAPI_BR_MSP	0x8000002e	No	Conditional branch instructions mispredicted
PAPI_BR_INS	0x80000037	No	Branch instructions
PAPI_TOT_CYC	0x8000003b	No	Total cycles

```
-----
```

PAPI Tools - papi_command_line

```
% papi_command_line -h
```

```
This utility lets you add events from the command line interface to see if they work.
```

```
Usage: papi_command_line [options] [EVENTNAMEs]
```

```
Options:
```

```
General command options:
```

```
    -u          Display output values as unsigned integers
```

```
[..]
```

```
This utility performs work while measuring the specified events.
```

```
It can be useful for sanity checks on given events and sets of events.
```

```
% papi_command_line PAPI_TOT_INS PAPI_DP_OPS
```

```
This utility lets you add events from the command line interface to see if they work.
```

```
Successfully added: PAPI_TOT_INS
```

```
Failed adding: PAPI_DP_OPS
```

```
because: Event does not exist
```

```
PAPI_TOT_INS : 200557443
```

PAPI – Performance Measurement Categories

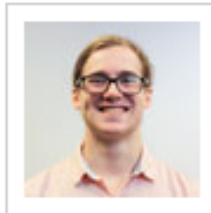
- Efficiency
 - Instructions per cycle (IPC)
 - Floating point operations (# integer ops)
 - Memory bandwidth
- Caches
 - Data cache misses and miss ratio
 - Instruction cache misses and miss ratio
- Translation lookaside buffers (TLB)
 - Data TLB misses and miss ratio
 - Instruction TLB misses and miss ratio
- Control transfers
 - Branch mispredictions
 - Near return mispredictions

PAPI - Conclusions

- PAPI is a library that provides a **consistent interface** for hardware performance counters, found across the system
- It comes with several **components** that allow users to monitor system information about CPUs, network cards, graphics accelerator cards, parallel file systems and more
- Events can be counted through either a simple **High-Level** programming interface or a more complete **Low-Level** interface from either **C or Fortran**
- PAPI can be used either directly by application developers, or indirectly as a **middleware** via 3rd party performance tools like Score-P, Scalasca, Vampir or TAU
- Sources and documentation: <https://bitbucket.org/icl/papi>

If you have any questions, do not hesitate to contact us at ptools-perfapi@icl.utk.edu

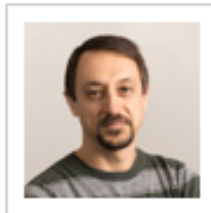
PAPI - Team



Daniel Barry
dbarry@vols.utk.edu
Graduate Research Assistant



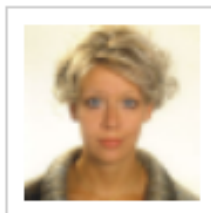
Giuseppe Congiu
gcongiu@icl.utk.edu
Research Scientist I



Anthony Danalis
adanalis@icl.utk.edu
Research Assistant Professor



Jack Dongarra
dongarra@icl.utk.edu
University Distinguished
Professor



Heike Jagode
jagode@icl.utk.edu
Research Assistant Professor



Anustuv Pal
anustuv@gmail.com
Research Scientist I

See a list of all
collaborators and
contributors over the
last 20 years!

<https://icl.utk.edu/papi/people/index.html>