PAPI: Performance API
Introduction & Overview

Giuseppe Congiu, Heike Jagode, Anthony Danalis
University of Tennessee

arm · FAU Friedrich-Alexander Universität Erlangen-Nürnberg · Lawrence Livermore National Laboratory · RWTH Aachen University · Technische Universität Dresden · Université de Versailles St-Quentin-en-Yvelines · Universität Stuttgart · Barcelona Supercomputing Center Centro Nacional de Supercomputación · Jülich Forschungszentrum · lrz Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities · Technische Universität Darmstadt · TUM Technische Universität München · University of Oregon · The University of Tennessee

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**

- ARM Cortex A8, A9, A15, ARM64

- IBM Blue Gene Series
- IBM Power Series, **PCP for POWER9-uncore**
- Intel Sandy|Ivy Bridge, Haswell, Broadwell, Skylake, **Kabylake**, **Cascadelake,** KNC, KNL, KNM

PAPI: Supported Architectures

- AMD **up to Zeppelin Zen**
- AMD **GPUs Vega**
- 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

- InfiniBand
- Lustre FS
- NVIDIA Tesla, Kepler, Maxwell, Pascal, **Volta**: support for multiple GPUs
- **NVIDIA: support for NVLink**

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

- Countable events are defined in two ways:
  - Platform-neutral **Preset Events** (e.g., PAPI_TOT_INS)
  - Platform-dependent **Native Events** (e.g., L3_CACHE_MISS)

- Preset Events can be **derived** from multiple Native Events (e.g. PAPI_L1_TCM might be the sum of L1 Data Misses and L1 Instruction Misses on a given platform)

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

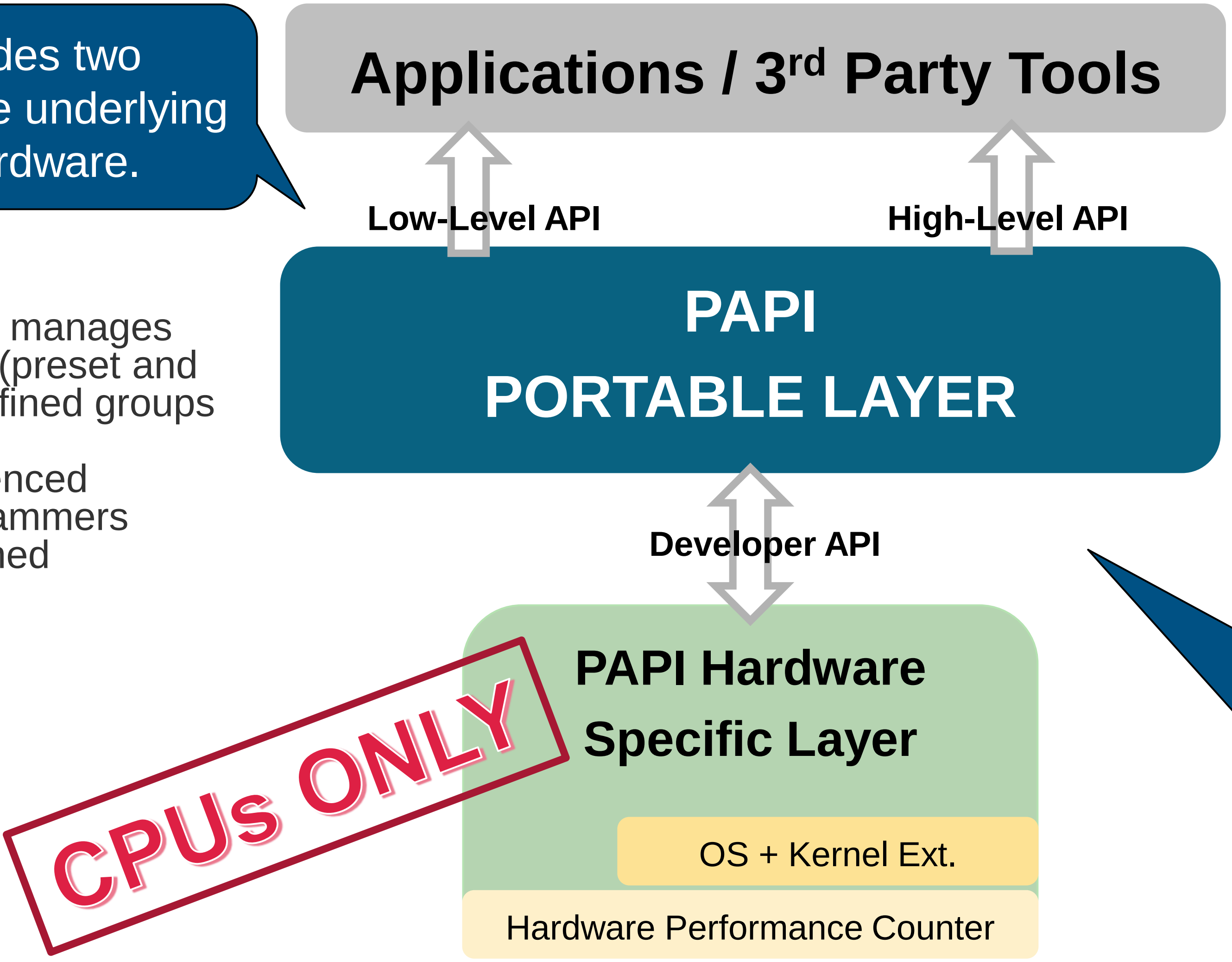
Native Events

- 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.


Applications / 3rd Party Tools
Low-Level API
High-Level API
PAPI
PORTABLE LAYER
Developer API
PAPI Hardware Specific Layer
OS + Kernel Ext.
Hardware Performance Counter
CPUs ONLY

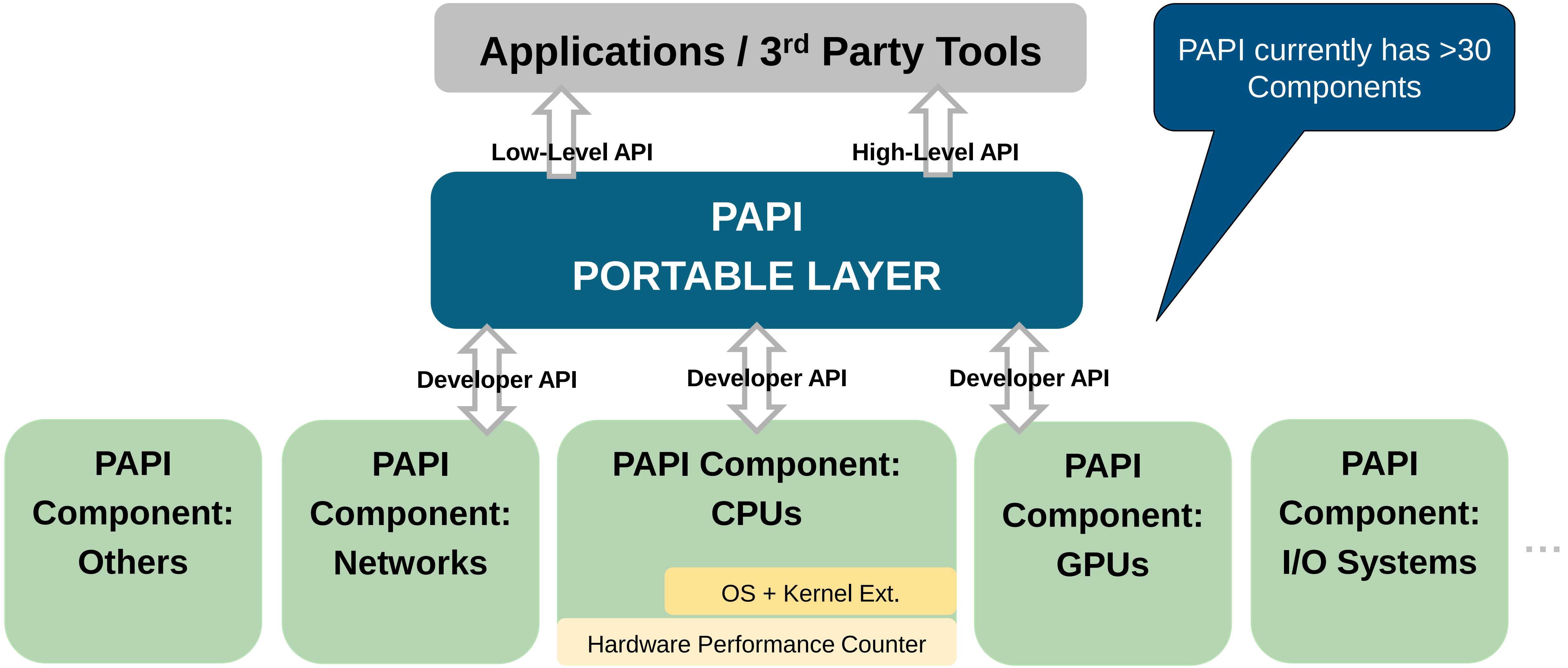
Graphical and end-user tools provide facile data collection and visualization.

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.

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

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”
```

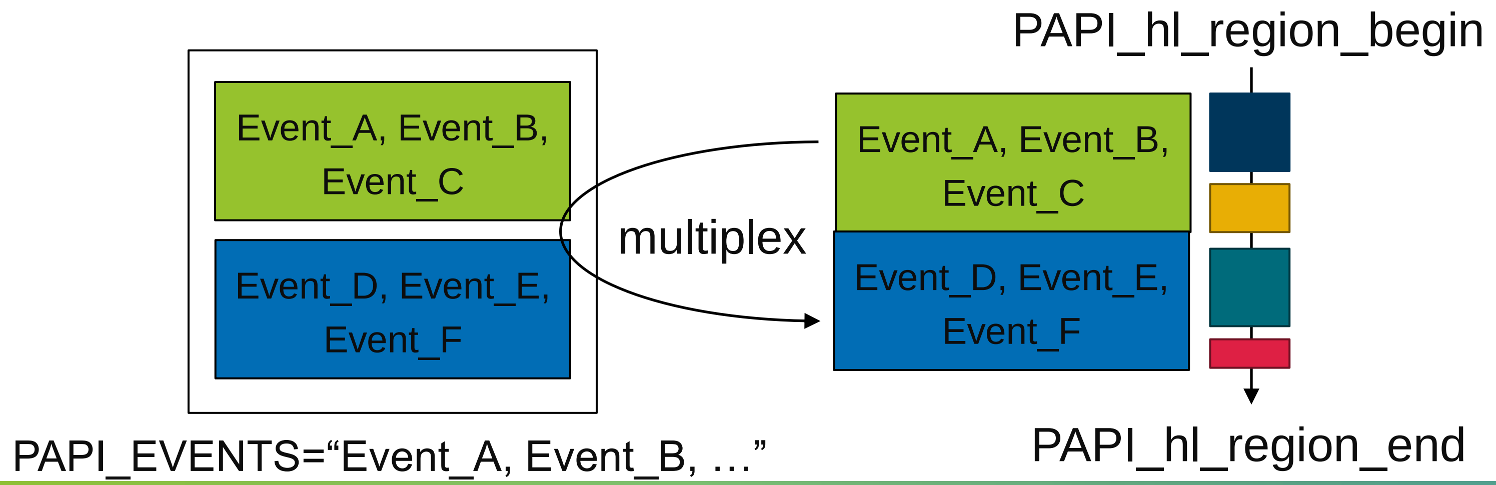
```
#include "papi.h"

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_hl_region_begin
Event_A, Event_B, Event_C
Event_D, Event_E, Event_F
multiplex
Event_A, Event_B, Event_C
Event_D, Event_E, Event_F
PAPI_EVENTS="Event_A, Event_B, …"
PAPI_hl_region_end

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/dd/dbc/group__low__api.html

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

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

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

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
- **Vampir** (GWT-TUD) http://www.vamir.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) http://www.ompp-tool.com

PAPI Tools

- Available components
  - **papi_component_avail**
- Available hardware
  - **papi_hardware_avail**
- Memory hierachy
  - **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**
- Revised High-Level performance report
  - **papi_hl_output_writer.py**

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

PAPI Tools

- **Hint: Check the tools on the compute node and not on the login node!**
  - Compute nodes can have both different architectures and different paranoid levels than the login nodes
- **Use an interactive job for PAPI's utility tools!**

```
# interactive job on Goethe-HLR
% srun -A training2123 -p booster --gres=gpu:4 -t 01:00:00 --pty /bin/bash -i

# do not load papi from the system modules; we will clone and build papi ourselves
```

PAPI Tools - papi_component_avail

```
% papi_component_avail
Available components and hardware information.
--------------------------------------------------------------------------------
PAPI version             : 6.0.0.1
…
--------------------------------------------------------------------------------
Compiled-in components:
Name:   perf_event                Linux perf_event CPU counters
Name:   perf_event_uncore         Linux perf_event CPU uncore and northbridge
Name:   io                        A component to read /proc/self/io
Name:   infiniband                Linux Infiniband statistics using the sysfs interface
Name:   net                       Linux network driver statistics
Name:   coretemp                  Linux hwmon temperature and other info
Name:   powercap                  Linux powercap energy measurements

Active components:
Name:   perf_event                Linux perf_event CPU counters
                                  Native: 173, Preset: 59, Counters: 10
                                  PMUs supported: ix86arch, perf, perf_raw, skx

Name:   perf_event_uncore         Linux perf_event CPU uncore and northbridge
                                  Native: 3065, Preset: 0, Counters: 176
[…]
```

PAPI Tools - papi_hardware_avail

```
% papi_hardware_avail

Device Summary -----------------------------------------------------------------
Vendor           DevCount
AuthenticAMD       (1)
 \-> Status: Device Initialized
NVIDIA             (4)
 \-> Status: libmpi.so: cannot open shared object file: No such file or directory
AMD/ATI            (0)
 \-> Status: ROCm not configured, no ROCm device available

Device Information -------------------------------------------------------------
Vendor                                  : AuthenticAMD
Id                                      : 0
Name                                    : AMD EPYC 7402 24-Core Processor
CPUID                                   : Family/Model/Stepping 23/49/0 0x17/0x31/0x00
Sockets                                 : 2
Numa regions                            : 8

Vendor                                  : NVIDIA
Id                                      : 0
Name                                    : NVIDIA A100-SXM4-40GB
Warp size                               : 32
Max threads per block                   : 1024
...
```

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 Data TLB:
  Page Size:              4 KB
  Number of Entries:     64
  Associativity:          4

Cache Information.

L1 Data Cache:
  Total size:            32 KB
  Line size:             64 B
  Number of Lines:      512
  Associativity:          8

L1 Instruction Cache:
  Total size:            32 KB
  Line size:             64 B
  Number of Lines:      512
  Associativity:          8
[…]
```

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   : 14
max cycles   : 39630
mean cycles  : 19.724646
std deviation: 59.824999

Performing start/stop test...

Total cost for PAPI_start/stop (2 counters) over 1000000 iterations
min cycles   : 14024
max cycles   : 113416
mean cycles  : 14220.991932
std deviation: 689.008573

[…]
```

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
--ins             Display instruction related PAPI preset events
--idl             Display Stalled or Idle PAPI preset events
--l1              Display level 1 cache related PAPI preset events
--l2              Display level 2 cache related PAPI preset events
--l3              Display level 3 cache related PAPI preset events
--mem             Display memory related PAPI preset events
[…]
```

PAPI Tools - papi_avail

Display Floating Point related PAPI preset events.

```
% papi_avail --avail --fp
Available PAPI preset and user defined events plus hardware information.
--------------------------------------------------------------------------------
PAPI version             : 6.0.0.1
Operating system         : Linux 3.10.0-1127.13.1.el7.x86_64
Vendor string and code   : GenuineIntel (1, 0x1)
Model string and code    : Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz (85, 0x55)
CPU revision             : 4.000000
CPUID                    : Family/Model/Stepping 6/85/4, 0x06/0x55/0x04
CPU Max MHz              : 3700
CPU Min MHz              : 1000
Total cores              : 80
…
--------------------------------------------------------------------------------


================================================================================
  PAPI Preset Events
================================================================================
    Name        Code    Deriv Description (Note)
PAPI_SP_OPS  0x80000067  Yes   Floating point operations; optimized to count scaled single precision vector operations
PAPI_DP_OPS  0x80000068  Yes   Floating point operations; optimized to count scaled double precision vector operations
PAPI_VEC_SP  0x80000069  Yes   Single precision vector/SIMD instructions
PAPI_VEC_DP  0x8000006a  Yes   Double precision vector/SIMD instructions
--------------------------------------------------------------------------------
Of 4 available events, 4 are derived.
```

PAPI Tools - papi_native_avail

```
% papi_native_avail
Available native events and hardware information.
--------------------------------------------------------------------------------
[…]
================================================================================
 Native Events in Component: perf_event
================================================================================
| ix86arch::UNHALTED_CORE_CYCLES                                               |
|            count core clock cycles whenever the clock signal on the specific |
|            core is running (not halted)
[…]
================================================================================
 Native Events in Component: infiniband
================================================================================
| infiniband:::mlx5_0_1_ext:req_remote_access_errors                           |
|            Req remote access errors (free-running 64bit counter).            |
[…]
================================================================================
 Native Events in Component: powercap
================================================================================
| powercap:::ENERGY_UJ:ZONE0                                                   |
|                                                                              |
[…]
Total events reported: 3426
```

Get an overview of all native events from each installed component.

PAPI Tools - papi_event_chooser

Check which events can be measured concurrently.

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


% papi_event_chooser PRESET PAPI_DP_OPS
Event Chooser: Available events which can be added with given events.
--------------------------------------------------------------------------------
…
    Name         Code     Deriv Description (Note)
PAPI_TOT_INS 0x80000032  No    Instructions completed
PAPI_VEC_DP  0x8000006a  Yes   Double precision vector/SIMD instructions
PAPI_REF_CYC 0x8000006b  No    Reference clock cycles
-------------------------------------------------------------------------
Total events reported: 3


% papi_event_chooser PRESET PAPI_DP_OPS PAPI_TOT_INS
Event Chooser: Available events which can be added with given events.
--------------------------------------------------------------------------------
…
PAPI_VEC_DP  0x8000006a  Yes   Double precision vector/SIMD instructions
PAPI_REF_CYC 0x8000006b  No    Reference clock cycles
-------------------------------------------------------------------------
Total events reported: 2
```

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
-x          Display output values as hexadecimal
-h          Print this help message
EVENTNAMEs  Specify one or more preset or native events

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_DP_OPS PAPI_TOT_INS

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

Successfully added: PAPI_DP_OPS
Successfully added: PAPI_TOT_INS

PAPI_DP_OPS :       40000000
PAPI_TOT_INS :      200622968

----------------------------------
```

PAPI Tools - papi_hl_output_writer.py

```
% papi_hl_output_writer.py -h
usage: papi_hl_output_writer.py [-h] [--source SOURCE] [--format FORMAT]
                                [--type TYPE] [--notation NOTATION]

optional arguments:
  -h, --help           show this help message and exit
  --source SOURCE      Measurement directory of raw data.
  --format FORMAT      Output format, e.g. json.
  --type TYPE          Output type: detail or summary.
  --notation NOTATION  Output notation: raw or derived.
```

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

Hands on - Code Optimization on CPU

- Measure instruction level parallelism (IPC)
  - An indicator of code efficiency
- Cache miss: a failed attempt to read or write a piece of data in the cache
  - Results in main memory access with much longer latency
  - Important to keep data as close as possible to CPU
- Example: Matrix multiplication

```
void matmul(A, B, C, N)
{
  int i, j, k;
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      float sum = 0.0;
      for (k = 0; k < N; k++) {
        sum += A[i][k] * B[k][j];
      }
      C[i][j] = sum;
    }
  }
}
```

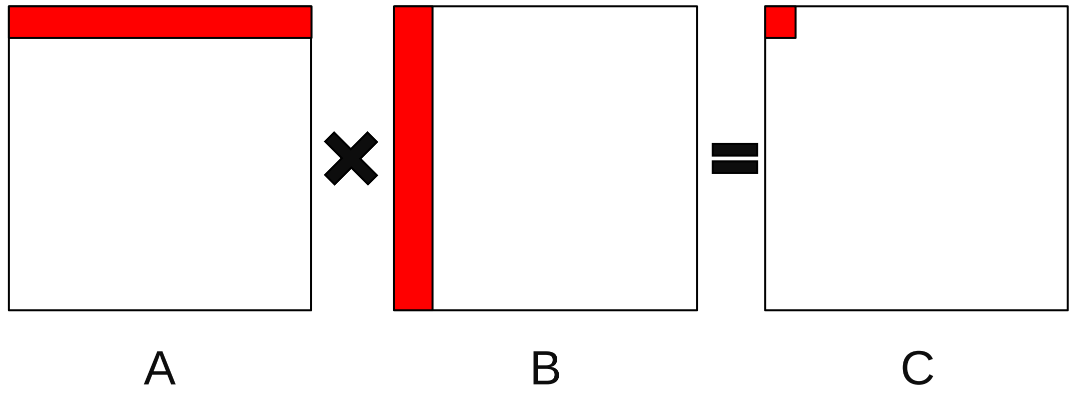
```
void matmul_reord(A, B, C, N)
{
  int i, j, k;
  for (i = 0; i < N; i++) {
    for (k = 0; k < N; k++) {
      for (j = 0; j < N; j++) {
        C[i][j] += A[i][k] * B[k][j];
      }
    }
  }
}
```

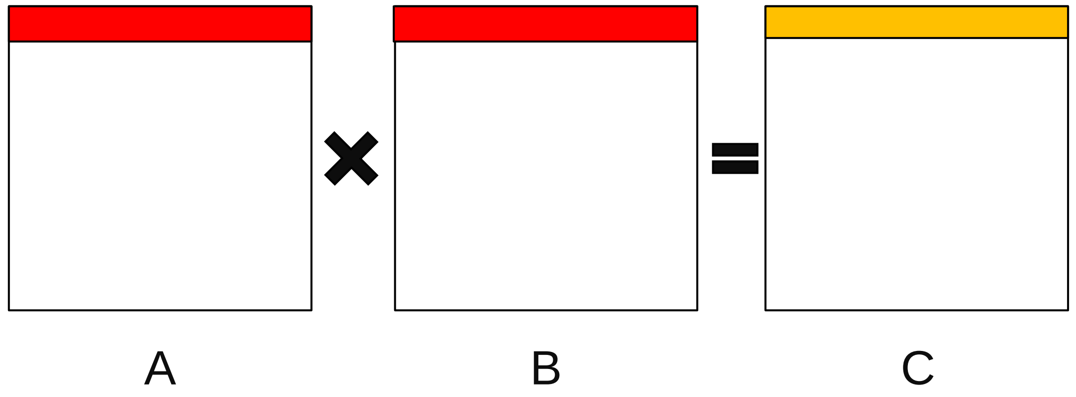
Loop rearranging leads to faster data access along two cache lines.

Hands on - Code Optimization on CPU

```
void matmul(A, B, C, N)
{
  int i, j, k;
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      float sum = 0.0;
      for (k = 0; k < N; k++) {
        sum += A[i][k] * B[k][j];
      }
      C[i][j] = sum;
    }
  }
}
```

```
void matmul_reord(A, B, C, N)
{
  int i, j, k;
  for (i = 0; i < N; i++) {
    for (k = 0; k < N; k++) {
      for (j = 0; j < N; j++) {
        C[i][j] += A[i][k] * B[k][j];
      }
    }
  }
}
```





PAPI – Code Optimization on CPU

- Comparison of classic and reordered matrix multiplication on Juwels Booster
- Matrix-matrix multiplication examples: matmul and matmul_reord
  - Matmul runs a "classic" matrix multiply while matmul_reord runs matrix multiply with the inner loops swapped. The swapped version should have better performance.
  - Find tests in: /p/project/training2123/work/congiu2/hands-on/cpu/{high-level,low-level}

```
# interactive job on juwels booster
% srun -A training2123 -p booster --gres=gpu:1 -t 01:00:00 --pty /bin/bash -i

# setup hands-on papi installation path
% export PATH=/p/project/training2123/work/congiu2/installed/bin:$PATH

% cd /p/project/training2123/work/congiu2/hands-on/cpu/high-level
% make
% ./matmul -N 1024
```

Level 1 data cache accesses (PAPI_L1_DCM) not available on this architecture.

PAPI – Code Optimization on CPU

```
% papi_hl_output_writer.py
{
    "Matrix-Matrix-Mult": {
        "Region count": 1,
        "Real time in s": 7.65,
        "IPC": 0.3,
        "MFLOPS/s": 280.72
    },
    "Reordered Matrix-Matrix-Mult": {
        "Region count": 1,
        "Real time in s": 0.66,
        "IPC": 3.92,
        "MFLOPS/s": 3253.76
    }
}
```

Roughly 13X improvement in reordered code

Hands on - Code Optimization on GPU

- Use the PAPI "cuda" component
  - NVIDIA deprecated CUPTI "Event API" for compute capabilities >= 7.0
  - NVIDIA devices with compute capability >= 7.0 need CUPTI 11 "Profiling API" & "Perfworks API"
  - PAPI supports both "Event API" and "Profiling API", and selects the right one automatically

PAPI on NVIDIA GTX 1060 GPUs (Pascal, Compute Capability 6.1)

```
[bin]$ ./papi_component_avail
----------------------------------------------------------------------
PAPI version             : 6.0.0.1
----------------------------------------------------------------------
...
Active components:
Name:   cuda      CUDA events and metrics via NVIDIA CuPTI interfaces
                  Native: 792, Preset: 0, Counters: 792

Name:   nvml      NVML provides the API for monitoring NVIDIA hardware
                  (power usage, temperature, fan speed, etc.)
                  Native: 72, Preset: 0, Counters: 72
```

PAPI on NVIDIA A100 GPUs (Ampere, Compute Capability 8.0)

```
[bin]$ ./papi_component_avail
----------------------------------------------------------------------
PAPI version             : 6.0.0.1
----------------------------------------------------------------------
...
Active components:
Name:   cuda      CUDA events and metrics via NVIDIA CuPTI interfaces
                  Native: 680888, Preset: 0, Counters: 680888

Name:   nvml      NVML provides the API for monitoring NVIDIA hardware
                  (power usage, temperature, fan speed, etc.)
                  Native: 216, Preset: 0, Counters: 216
```

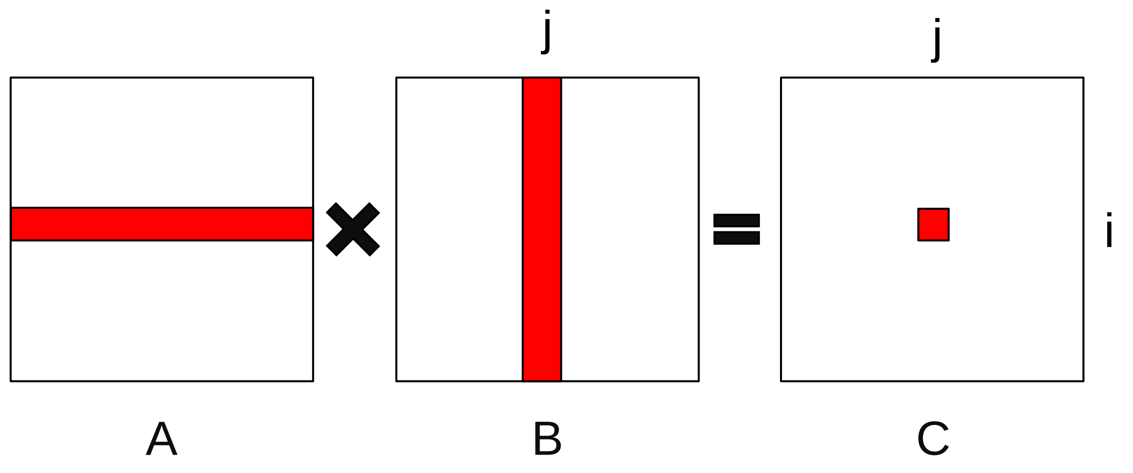
Hands on - Code Optimization on GPU

- Matrix multiplication on single-GPU device
  - Instructions executed by the Streaming Multiprocessor
  - Cycles in which the SM was active
  - Elapsed time

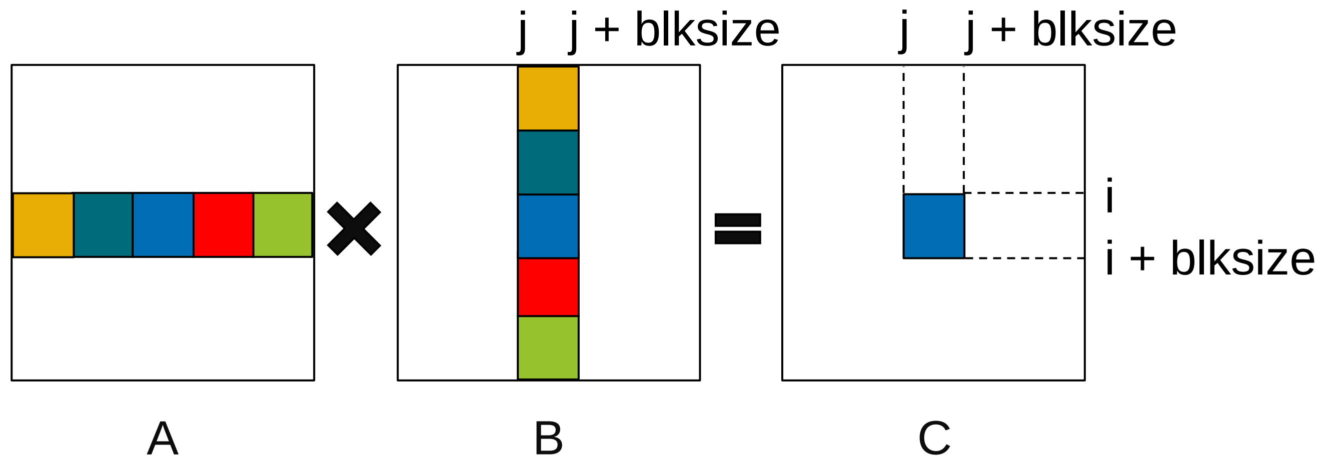
```
//matmul
__global__ void matmul(A, B, C, N, dev, dev_count)
{
    int i = (blockIdx.y * blockDim.y) + threadId.y;
    int j = (blockIdx.x * blockDim.x) + threadId.x;

    if (i < N && j < N) {
        for (int k = 0; k < N; ++k) {
            sum += A[i][k] * B[k][j];
        }
        C[i][j] = sum;
    }
}
```

```
//matmul_blk
__global__ void matmul_blk(A, B, C, N, dev, dev_count)
{
    double *Csub = getSubMatrix(C, N, r_blk, c_blk);
    double val = 0.0;

    for (int k = 0; k < N / BLKSIZE; ++k) {
        double *Asub = getSubMatrix(A, N, r_blk, k);
        double *Bsub = getSubMatrix(B, N, k, c_blk);
        __shared__ double As[BLKSIZE][BLKSIZE];
        __shared__ double Bs[BLKSIZE][BLKSIZE];
        As[i][j] = getElement(Asub, N, i, j);
        Bs[i][j] = getElement(Bsub, N, i, j);
        __syncthreads();
        for (int e = 0; e < BLKSIZE; ++e)
            val += As[i][e] * Bs[e][j];
        __syncthreads();
    }
    setElement(Csub, N, i, j, val);
}
```

Hands-on – Code Optimization on GPU



matmul: map thread (i, j) in CUDA block to elements in C and compute each value Independently.



matmul_blk: map thread (i, j) in CUDA block to elements in a sub-block of matrix C and compute all elements in that block together.

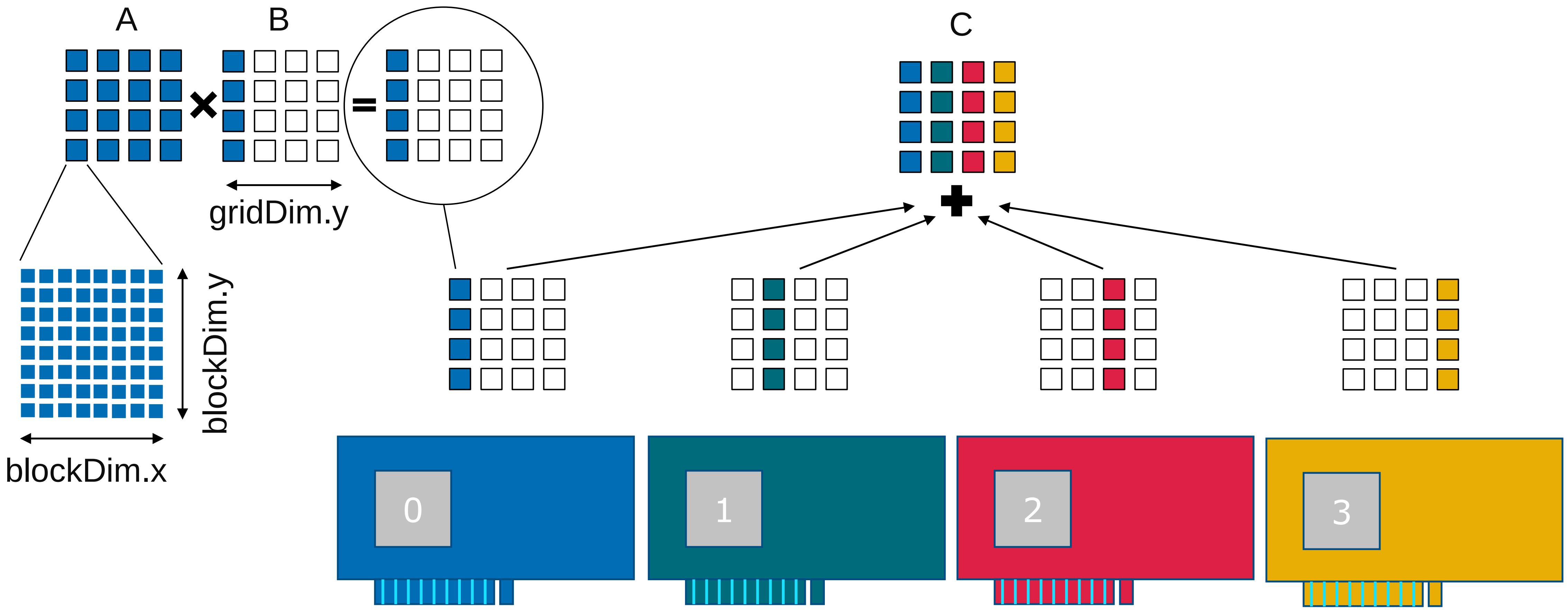
PAPI – Code Optimization on GPU

```
% ./matmul -N 8192
cuda:::sm__inst_executed.sum:device=0 : 77403783168
cuda:::sm__cycles_active.sum:device=0 : 79466112027
cuda:::fe__cycles_elapsed.sum:device=0 : 943507344
```

```
% ./matmul_blk -N 8192
cuda:::sm__inst_executed.sum:device=0 : 52690944000
cuda:::sm__cycles_active.sum:device=0 : 61312238598
cuda:::fe__cycles_elapsed.sum:device=0 : 774306379
```

Roughly 32% reduction in instructions executed and 18% reduction in elapsed cycles by using shared memory

Exercise – rewrite matmul kernel for multi-GPU

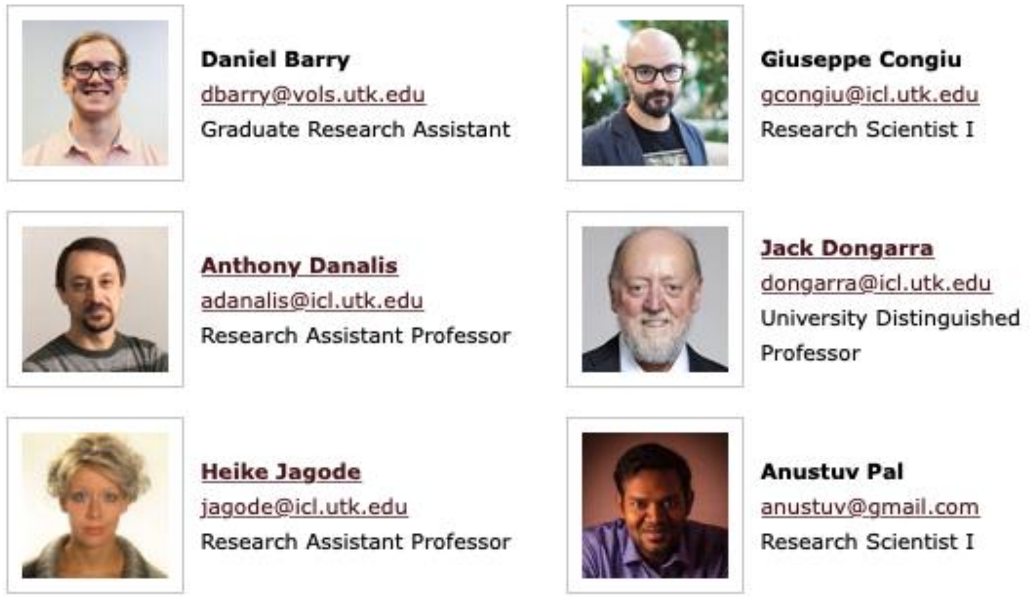


PAPI - Conclusions

- PAPI is a library that provides a **consistent interface** for hardware performance counters, found across the system
- It comes with several **components** that allow to monitor system information of 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

Thank you!