PAPI - C
What Can Performance Components Do for You?

Dan Terpstra
with a little help from:
Heike Jagode,
Brian Sheely,
Vince Weaver,
& James Ralph

Center for Scalable Application Development Software
Performance Tools Aug 2 – 5, 2010

ICL UT
INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE

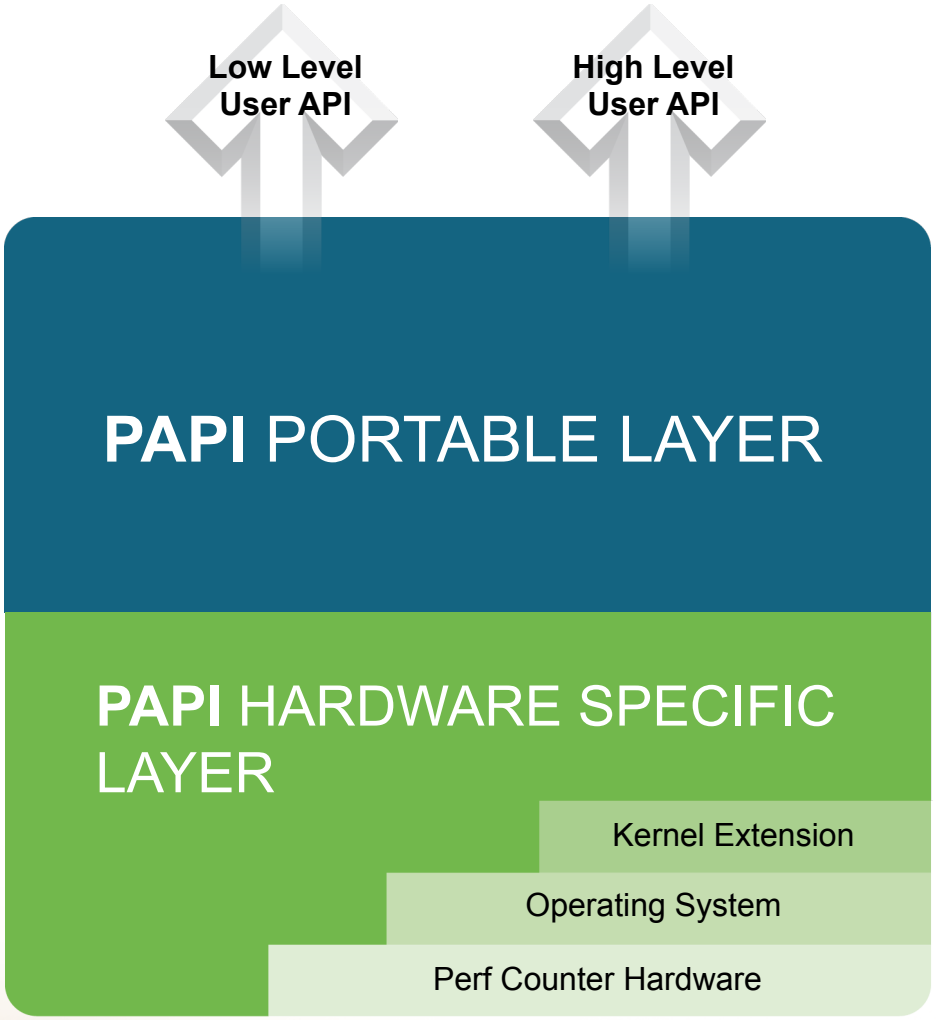
What's PAPI? What's PAPI-C?

(component??)

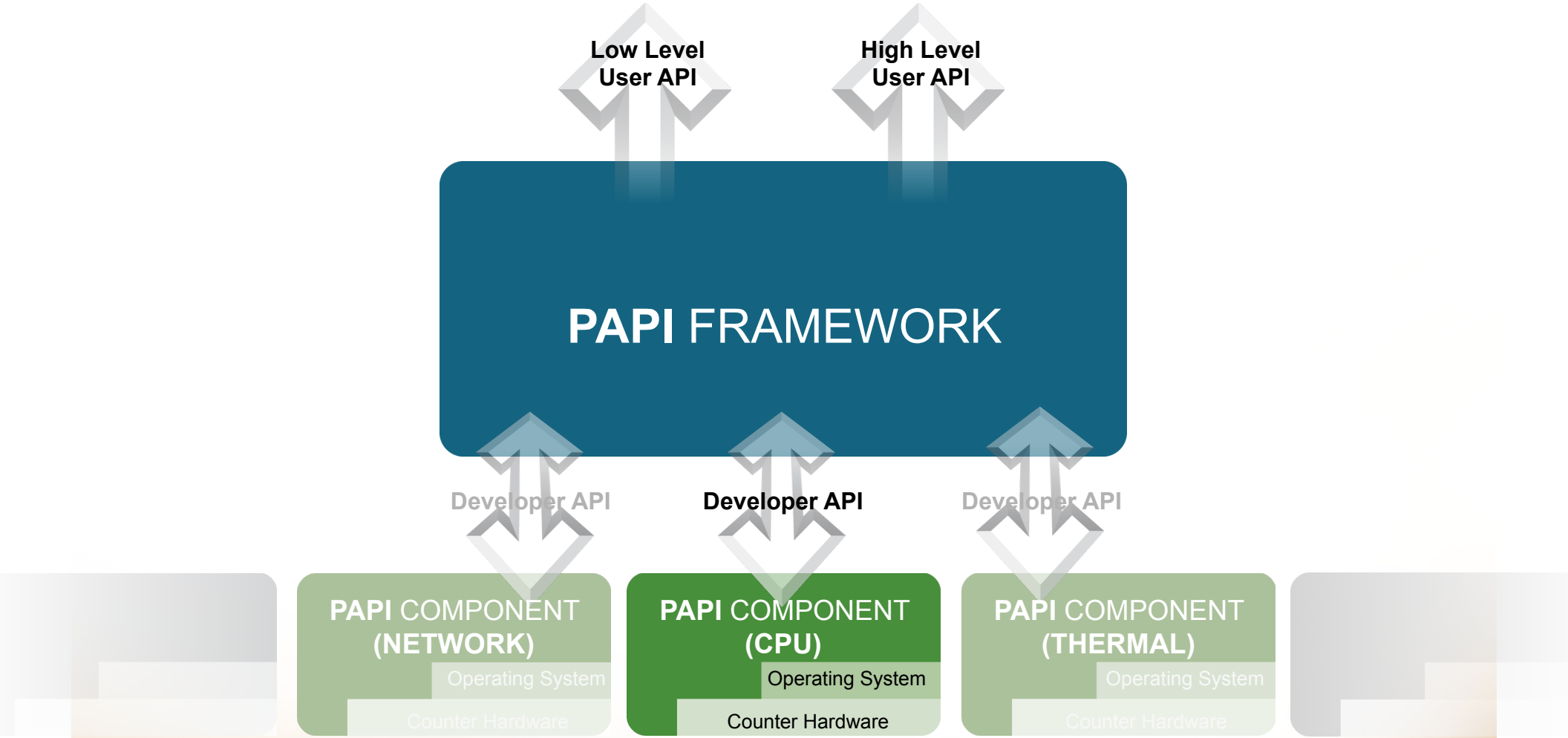
- A software layer (library) designed to provide the tool developer and application engineer with a consistent interface and methodology for use of the performance counter hardware found in most major micro-processors.
  - Platform-neutral Preset Events
  - Platform-dependent Native Events
- All events referenced by name and collected in EventSets for sampling
- Events can be multiplexed if counters are limited
- Statistical sampling on timeout or overflow

- A software layer (library) designed to provide the tool developer and application engineer with a consistent interface and methodology for measurement of performance events found at any level of the computing hierarchy.
  - CPU core events
  - CPU chip level events
  - Networks
  - System Health
  - Peripheral subsystems
  - Etc…

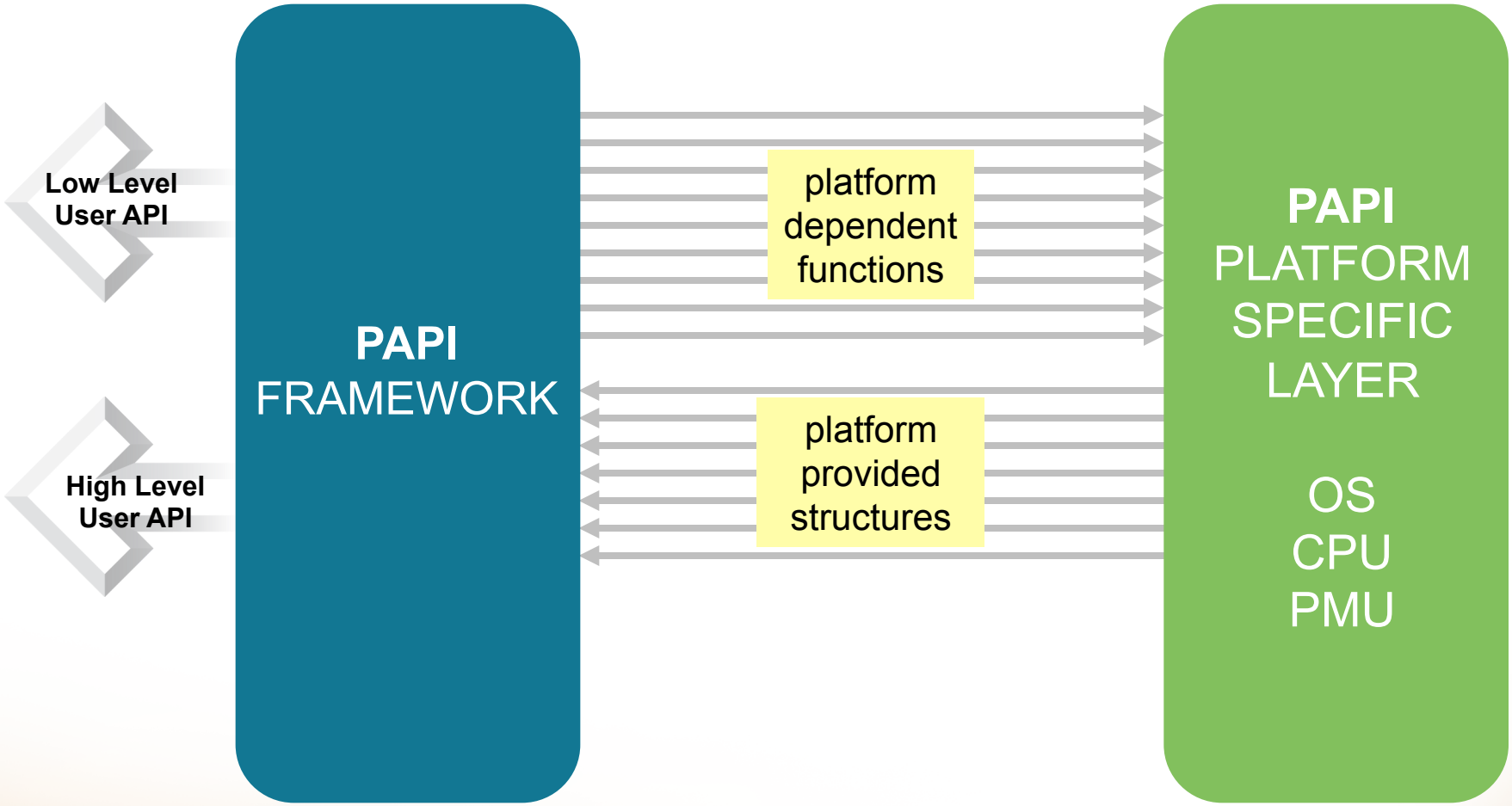
‘PAPI Classic’



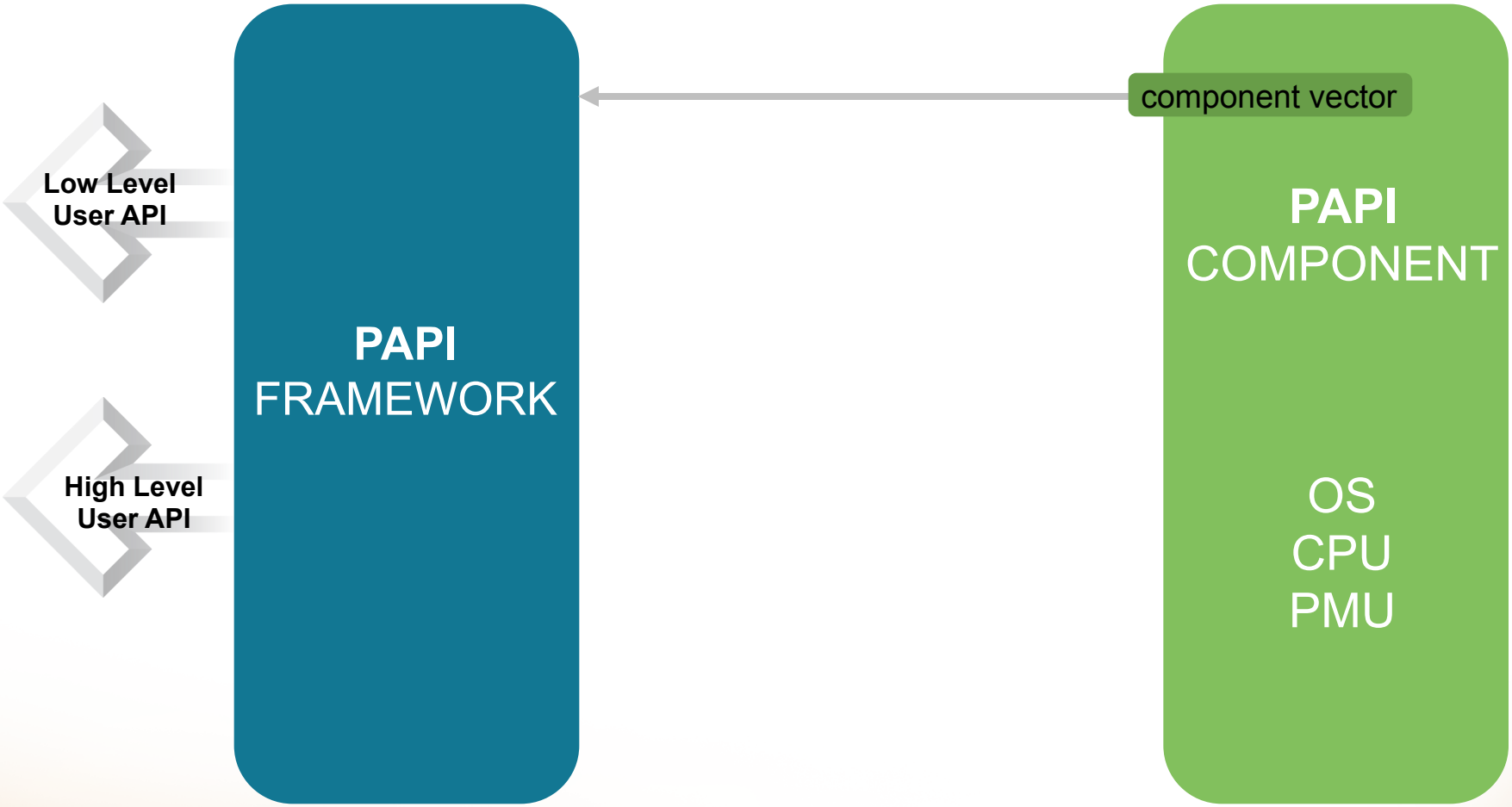
Component **PAPI**



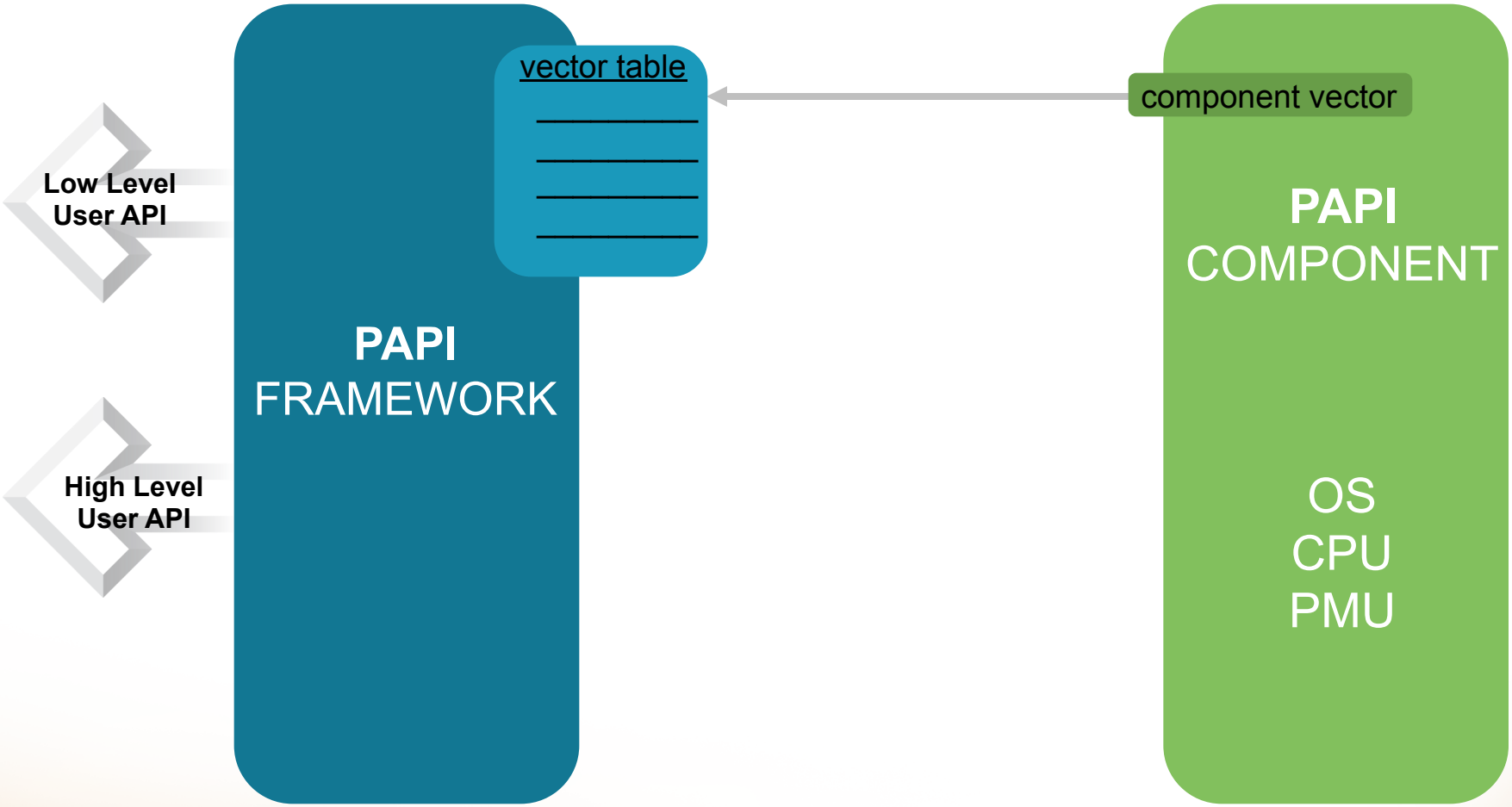
PAPI Classic Direct Linking



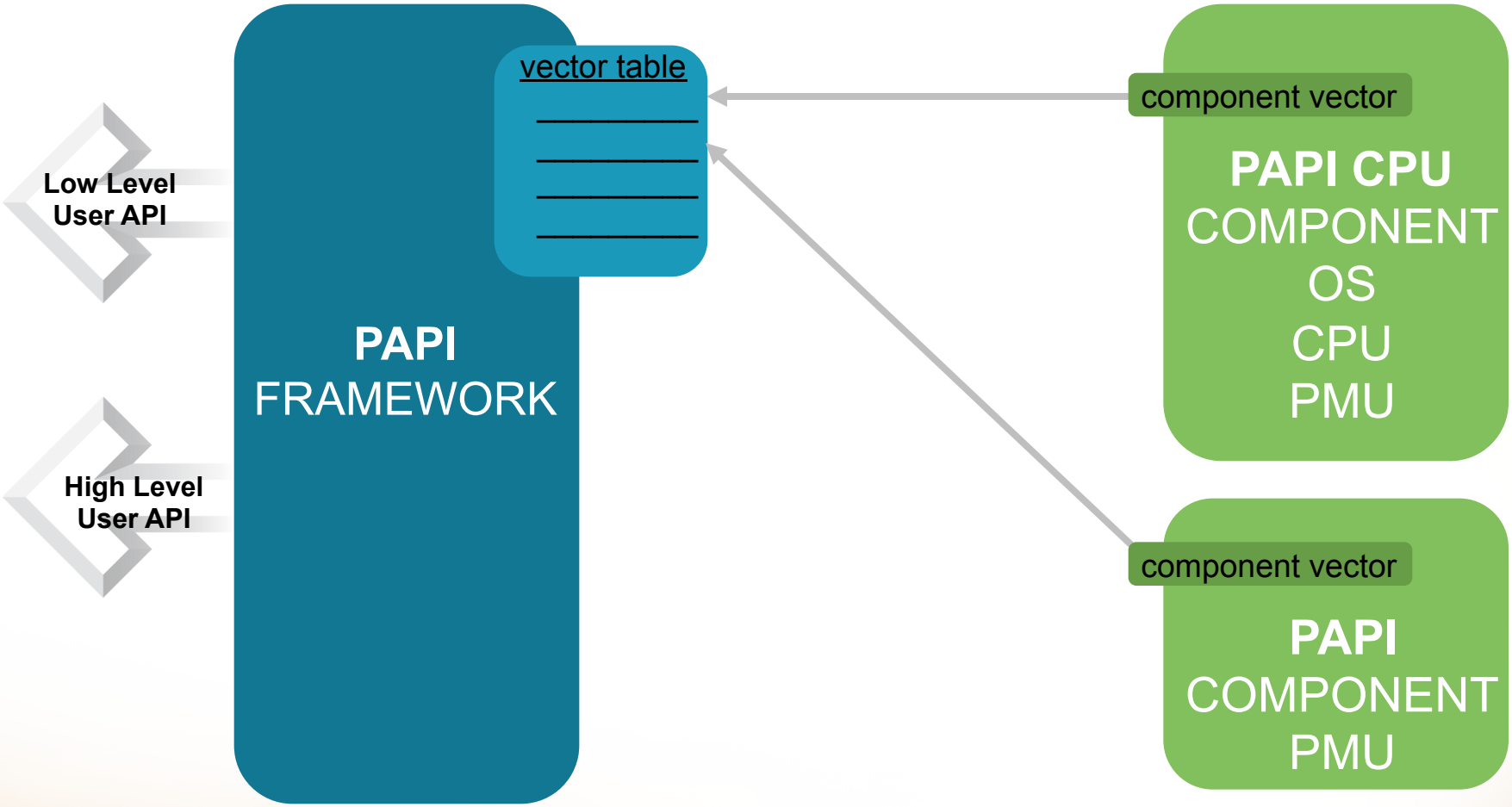
Component **PAPI** vectors



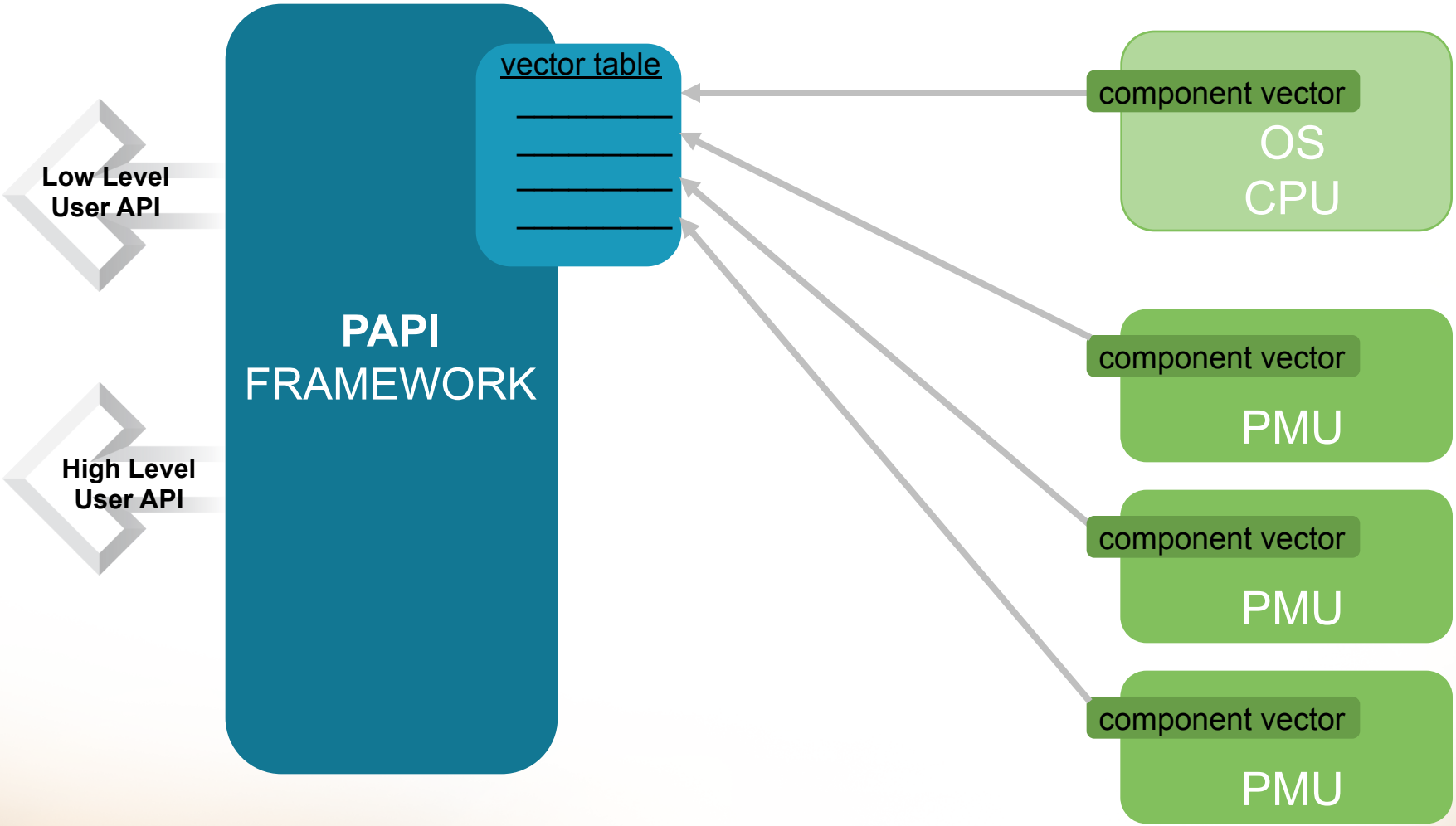
Vector table: Indirect Linking



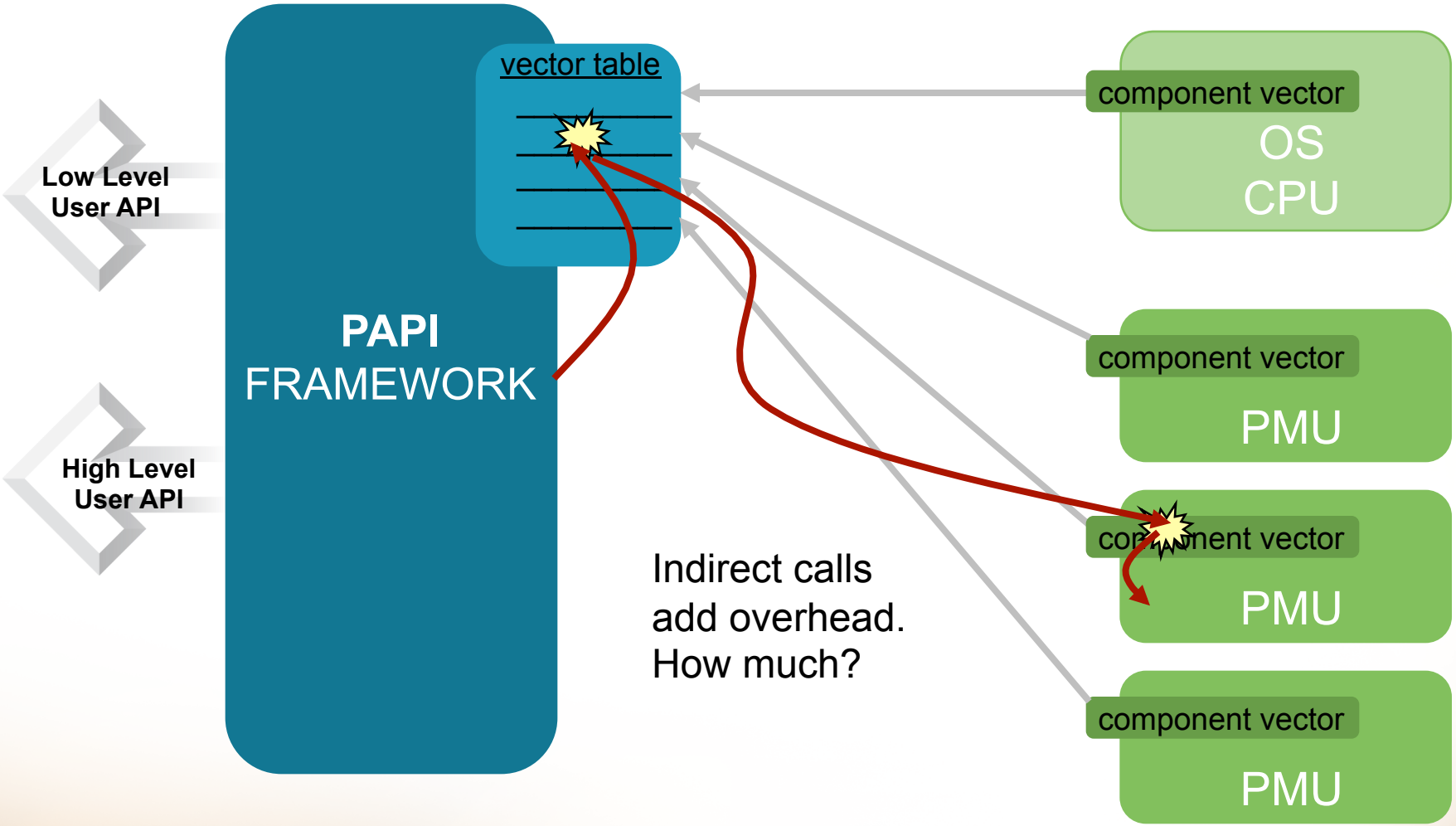
Special CPU Component



Null CPU Component



Indirect Calls



Indirect Call costs

- How much does an indirect call cost?
- Test on various platforms
- 1M iterations of 10 calls
  - To empty functions
  - To PAPI functions

	Pentium4	Core2	Nehalem	Opteron	POWER6
direct cycles/call	13.8	8.4	5.8	9.6	106.3
indirect cycles/call	17.8	10.3	6.2	11	155.2
% slowdown	29.00%	22.60%	6.90%	14.60%	46.00%
PAPI start/stop slowdown	0.66%	0.52%	0.13%	0.39%	1.36%
PAPI 2 counter read slowdown	9.76%	6.40%	2.47%	11.30%	1.26%

API Changes: EventSets

- Events are encapsulated in EventSets
- An EventSet can contain multiple events
- Multiple EventSets can co-exist
- Only one EventSet can be active per component
- An EventSet is bound to a single component
  - When the first event is added
    - Late binding insures backward compatibility
  - By use of a new API call:
    - PAPI_assign_eventset_component()
- Old code can run with no source modification
  - (except some instances of multiplexing)

API Changes: Function Calls

- 3 calls augmented with a component index
  - PAPI_get_opt → PAPI_get_cmp_opt
  - PAPI_set_domain → PAPI_set_cmp_domain
  - PAPI_num_hwctrs → PAPI_num_cmp_hwctrs
- Old syntax preserved in wrapper functions for backward compatibility
  - CPU component is assumed to be component 0
- New entry points for new functionality:
  - PAPI_num_components
  - PAPI_get_component_info

- Old code can run with no source modifications

Building PAPI with Components

```
UNIX> configure --with-components="lustre net acpi”
UNIX> cat components_config.h

/* Automatically generated by configure */
extern papi_vector_t MY_VECTOR;
extern papi_vector_t _lustre_vector;
extern papi_vector_t _net_vector;
extern papi_vector_t _acpi_vector;

papi_vector_t *_papi_hwd[] = {
   &MY_VECTOR,
   &_lustre_vector,
   &_net_vector,
   &_acpi_vector,
   NULL
};

UNIX> make
```

Ask not…

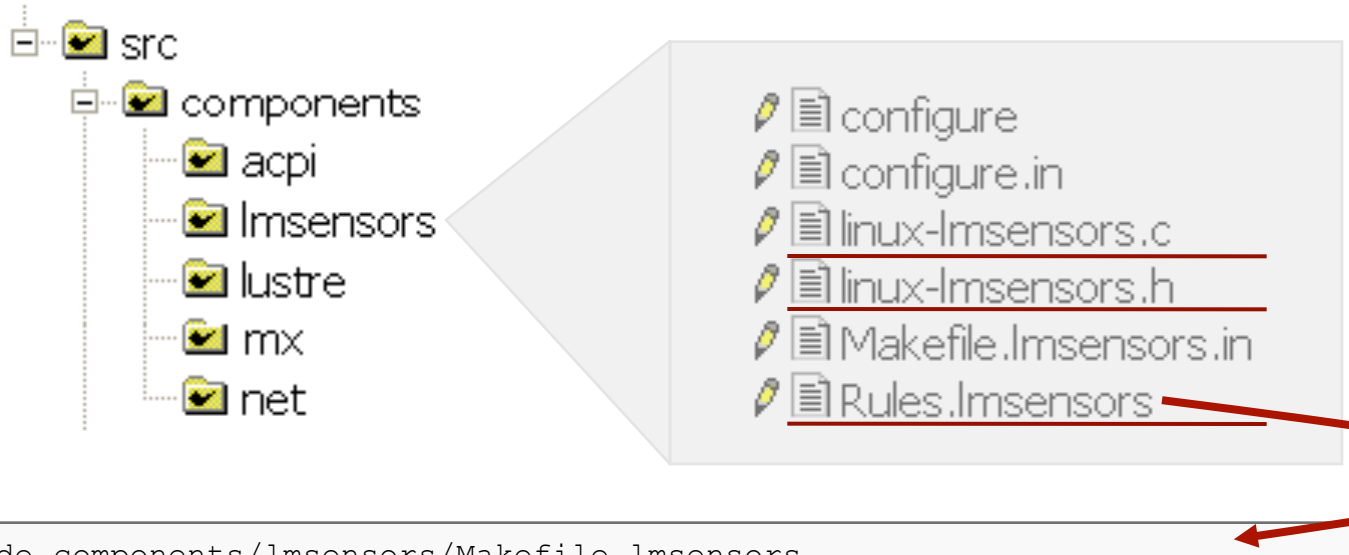
"What Can Performance Components Do for You? "

…Ask what you can do for Performance Components

Dan Terpstra
with a little help from:
Heike Jagode,
Brian Sheely,
Vince Weaver,
& James Ralph

ICL UT

Adding a Component




```
include components/lmsensors/Makefile.lmsensors

COMPSRCS += components/lmsensors/linux-lmsensors.c
COMPOBJS += linux-lmsensors.o
CFLAGS += -I$(SENSORS_INCDIR)
LDFLAGS += -L$(SENSORS_LIBDIR) -lsensors
LINKLIB += $(SENSORS_LIBDIR)/libsensors.a -lm

linux-lmsensors.o: components/lmsensors/linux-lmsensors.c components/lmsensors/linux-lmsensors.h
        $(HEADERS) $(CC) $(LIBCFLAGS) $(OPTFLAGS) -c components/lmsensors/linux-lmsensors.c
        -o linux-lmsensors.o
```

Component Developers Interface

- About 40 calls in the complete CDI
- About 15 needed for a useful component
- About 1000 lines of code
- CDI Documentation
- Component Function List
- Component Cookbook

A PAPI Component Repository

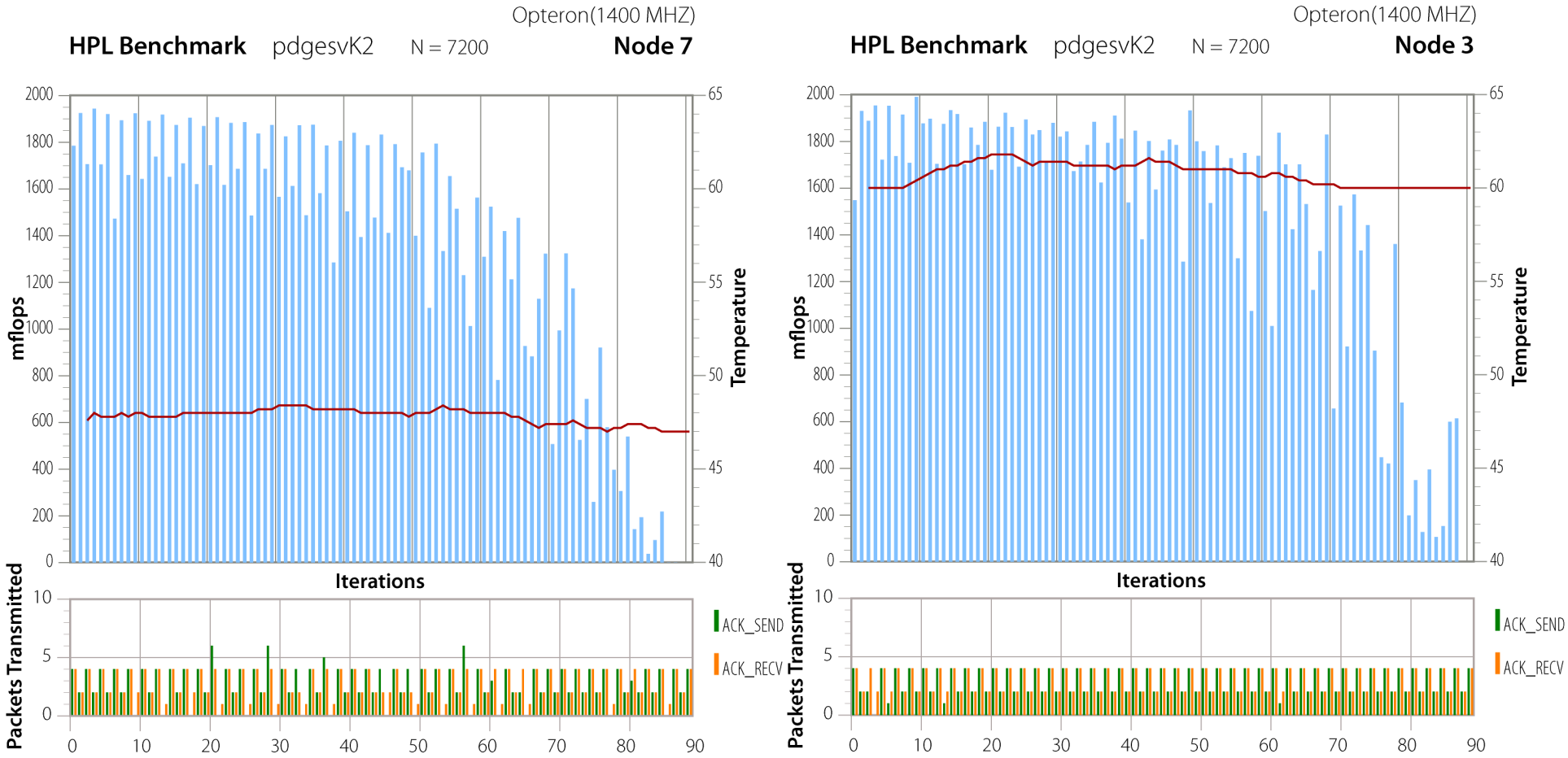
- We want user contributions
  - We **don't** want to maintain them
- Users want to know what's available
  - And often want to contribute
- Why not a web-based Repository?
  - Registration form to submit and track components
  - Link to a tarball or RCS repository
    - Sourceforge, GitHub, Google code, private repository
  - Public page to view current components & descriptions
  - Private page for author updates
  - Admin page to monitor / control submissions

PAPI Alphabet Soup (Part 1)

- **PAPI-G**: GPUs
  - Radically different performance models
  - In conversation with nVidia
- **PAPI-N**: Networks
  - Myrinet
  - InfiniBand
  - Gig-E
  - Gemini??
- **PAPI-D**: Disks
  - User suggestions and implementations
  - Lustre in development
- **PAPI-H**: Health
  - Temperature, Voltages, Power, Fans and Fan Speed…
  - Lm-Sensors
  - coretemp linux driver
  - IPMI (Intelligent Platform Management Interface)

A Component PAPI Example

- HPCC HPL benchmark on Opteron with 3 performance metrics:
  - FLOPS; Temperature; Network Sends/Receives
    - Temperature is from an on-chip thermal diode



Lustre Component

(Deimos: Dual AMD Opteron x86_64 Cluster)

Measures data collected in: /proc/fs/lustre/llite/…/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 papi_native_avail 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
```

lm-sensors Component

Access computer health monitoring sensors, exposed by lm_sensors library

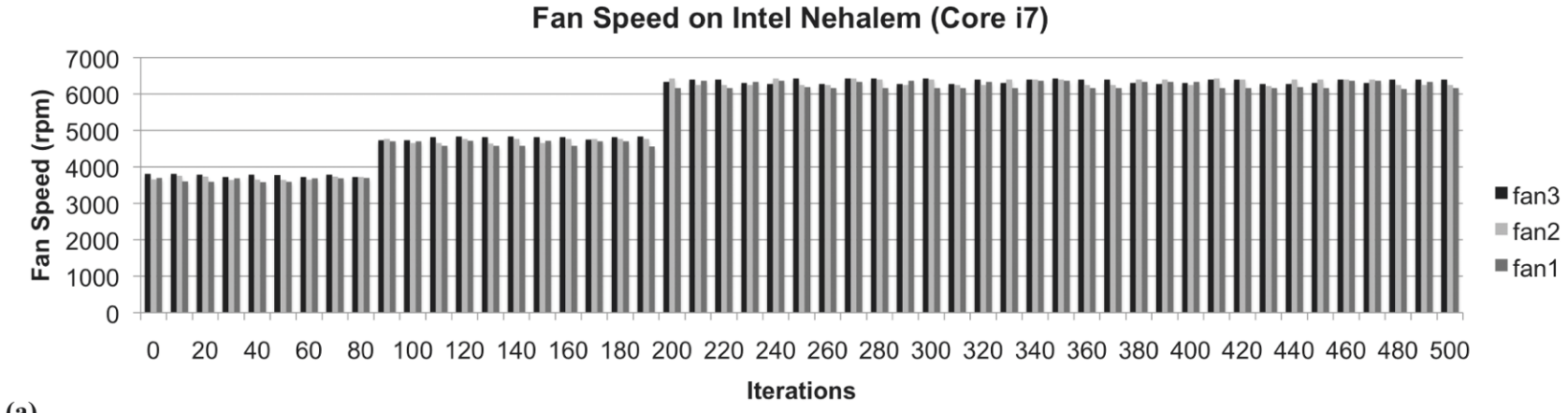
- user is able to closely monitor the system's hardware health
  - observe feedback between performance and environmental conditions
- Available features and monitored events depend on hardware setup

E.g. snippet of papi_native_avail on Gonzo (Intel Nehalem @ ICL):

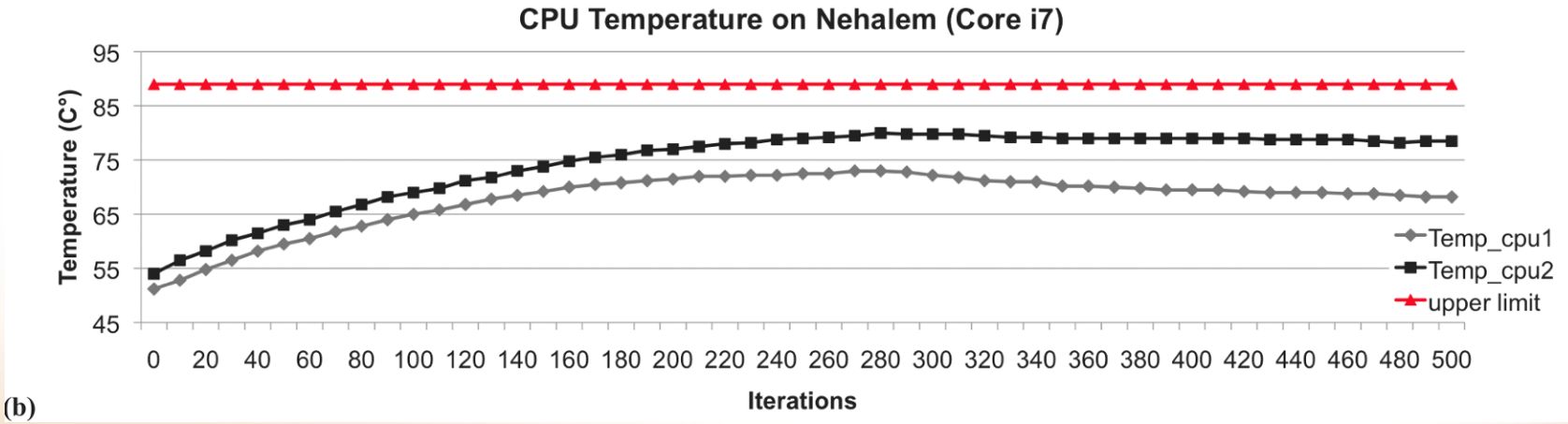
```
…
0x4c000000    LM_SENSORS.max1617-i2c-0-18.temp1.temp1_input
0x4c000001    LM_SENSORS.max1617-i2c-0-18.temp1.temp1_max
0x4c000002    LM_SENSORS.max1617-i2c-0-18.temp1.temp1_min
…
0x4c000049    LM_SENSORS.w83793-i2c-0-2f.fan1.fan1_input
0x4c00004a    LM_SENSORS.w83793-i2c-0-2f.fan1.fan1_min
0x4c00004b    LM_SENSORS.w83793-i2c-0-2f.fan1.fan1_alarm
…
```

lm-sensors Component Example

libsensors version 3.1.1



(a)



(b)

InfiniBand Component

Measures everything that is provided by the libibmad:

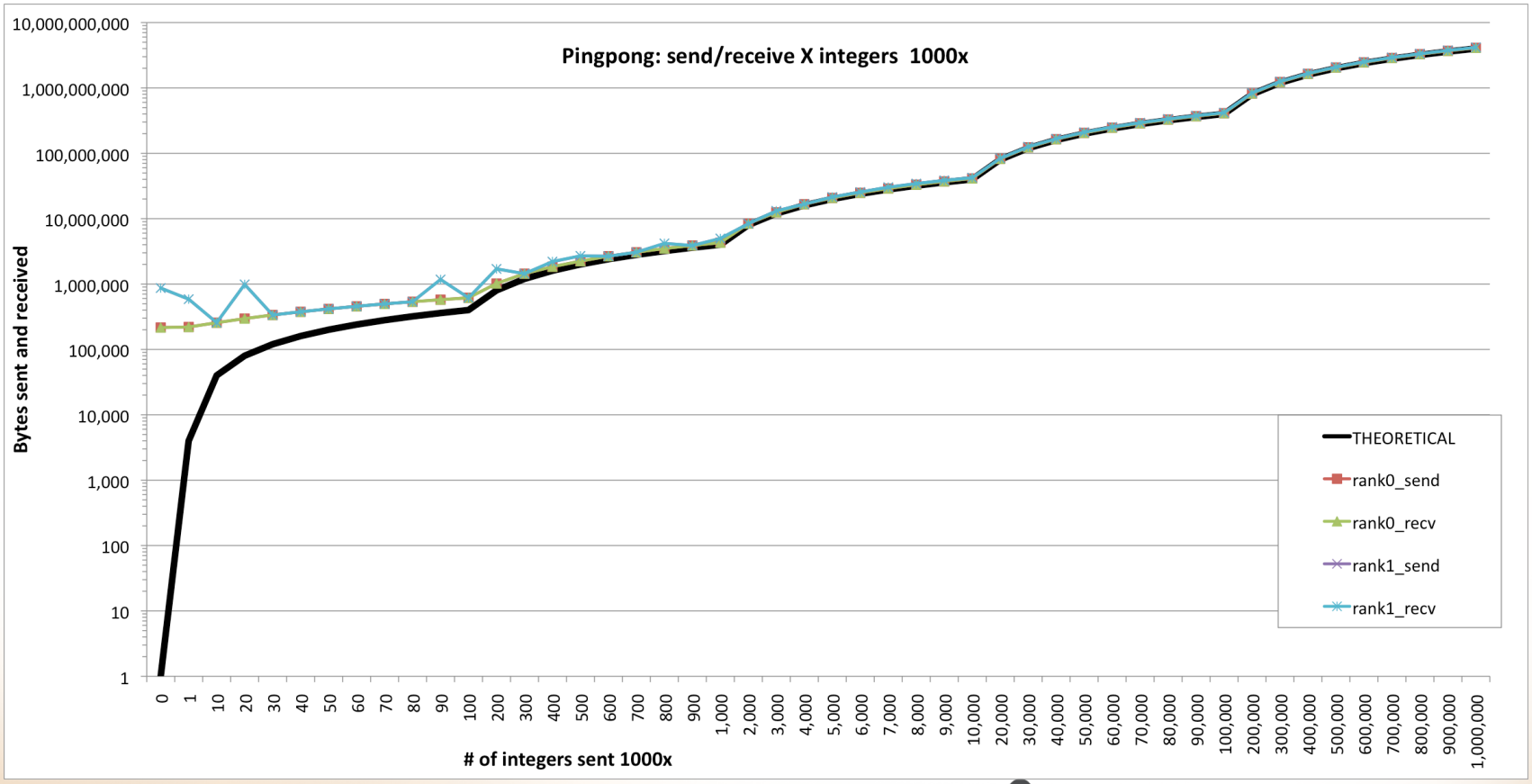
- Errors, Bytes, Packets, local IDs (LID), global IDs (GID), etc.
- **ibmad library** provides low-layer IB functions for use by the IB diagnostic and management programs, including MAD, SA, SMP, and other basic IB functions

E.g. snippet of papi_native_avail on Moria (2 IB devices: mthca0 and mthca1):

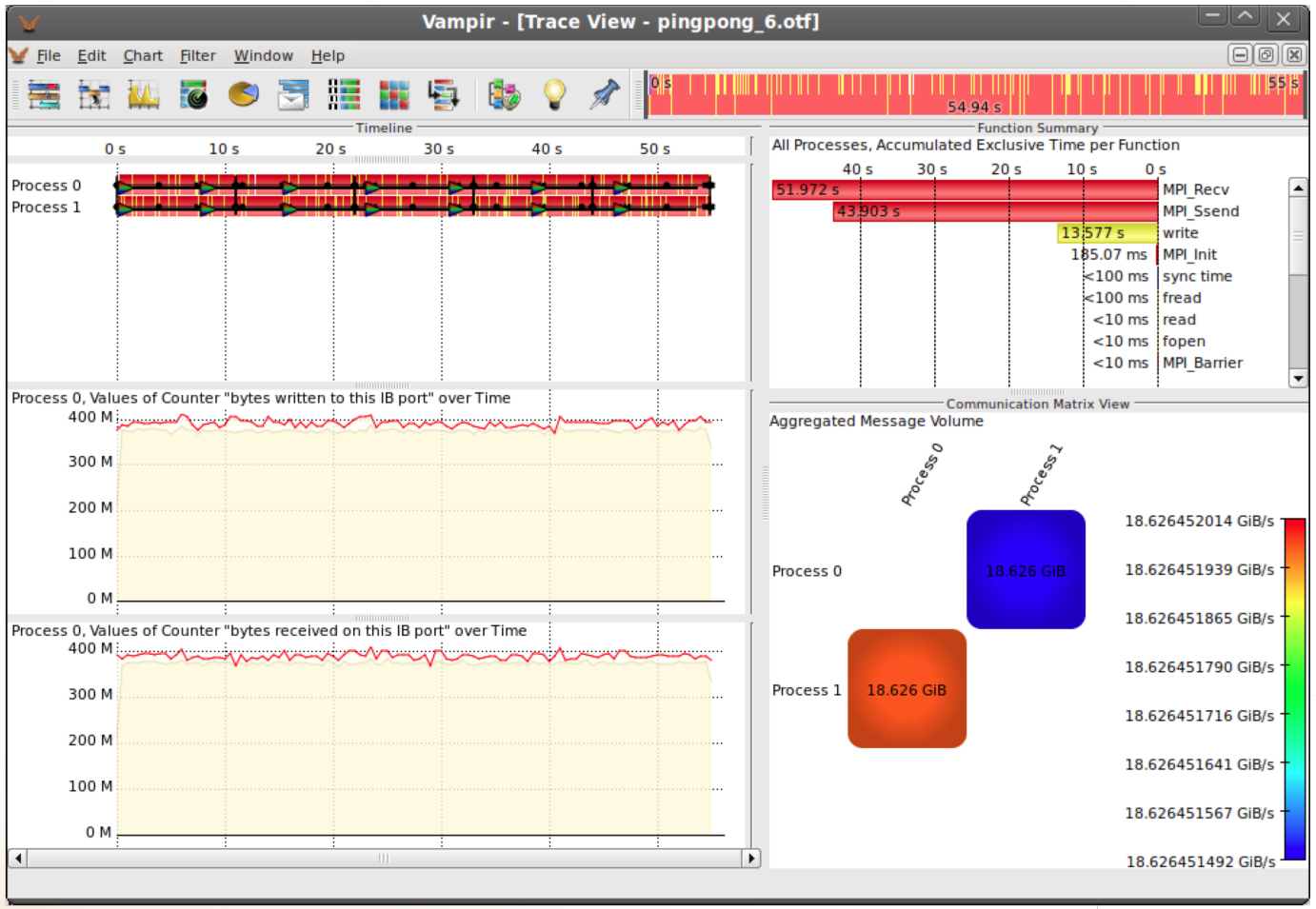
```
…
0x44000000    mthca0_1_recv  | bytes received on this IB port
0x44000001    mthca0_1_send  | bytes written to this IB port
0x44000002    mthca1_1_recv  | bytes received on this IB port
0x44000003    mthca1_1_send  | bytes written to this IB port
…
```

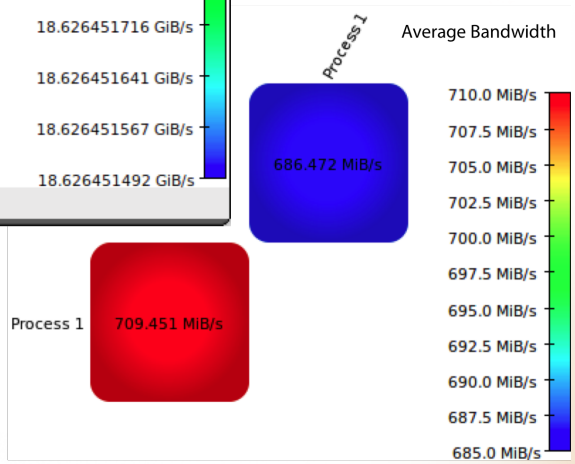
InfiniBand Component Results

(on a Dual Core AMD Opteron x86_64 Cluster)



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





IB Counter resolution in Vampir:

1 sec

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

Look Ma! No_cpu!

- PAPI *without* cpu events??
  - Requested by developers in Dresden
  - Debugging components on unpatched kernels
  - Running non-cpu components only
- Requires emulation of basic timing functions
- Available in PAPI 4.1.0
- Invocation:
  - `> configure --with-no-cpu-counters = yes`

PAPI Alphabet Soup (Part 2)

- **PAPI-M**: Multi-core
  - How to measure off-core / on-chip resources?
  - Vendors each have different approaches
  - Requires kernel work
- **PAPI-V**: Virtual
  - What does it mean to measure performance in the cloud?
  - Hypervisor support for performance counters? Not yet…
  - Opportunities for novel Components?

Future Directions for **PAPI-C**

- **Richer Event Naming**
  - PAPI events are 32-bit codes
  - Only 4 bits reserved for (16) components
  - Migrate to a named event model:
    - `[pmu::]event_name[:unit_mask][:attribute][:modifier=val]`
- **Richer Data Types**
  - PAPI data is 64-bit unsigned integer; good for counting stuff
  - Components may need more expressivity:
    - signed, float, fixed point, integer ratios, more?
- **Dynamic Configurability**
  - Build-time configuration
  - Component Repository
  - Run-time discovery? Components as shared objects?
  - Proprietary Components?
- **Time Base Synchronization**
  - Vastly different time scales, from nano- to milli-seconds
  - Skew and drift between components
  - Variable CPU clocks (Speed-Step, Turbo Boost)
- **Per Core Measurements**
- **Uncore Measurements**

PAPI - C
What Can Performance Components Do for You?

Dan Terpstra
with a little help from:
Heike Jagode,
Brian Sheely,
Vince Weaver,
& James Ralph

Center for Scalable Application Development Software
Performance Tools Aug 2 – 5, 2010

ICL UT
INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE