PAPI Users Group Meeting SC2003

Philip Mucci, mucci@cs.utk.edu
Felix Wolf, fwolf@cs.utk.edu
Nils Smeds, smeds@pdc.kth.se

Tuesday, November 18th
Phoenix, AZ

Agenda

- CVS Web Structure
- 2.3.4 Bugs
- 2.3.5 Release
- PAPI 3.0, 3.1
- Status of Platforms
- Feature Requests

CVS Web Presence

- New web site format
  - Better tools section, now includes 3rd party
  - Up to date reference information
  - Up to date FAQ
- Bugzilla
  - http://icl.cs.utk.edu/projects/papi/bugz

2.3.4 Bugs

- Bad exe info when built as shared library.
  - Undefined _data_start in libpapi.so in certain configurations. (Rice)
- Very, very rare, multiplexing errors causing:
  - Assert() on IA64, IA32 platforms
  - Hang of POE applications on AIX 5
    - Re-enters pm_child_sighandler, hangs aquiring lock

2.3.4 Bugs 2

- Pentium IV
  - L1DCM/L2DCM Metrics return same numbers.
- Alpha EVx
  - Returning all zeroes even with patched kernel.
- AIX 5 and Threads
  - Pmapi was returning “ Context already created” from pm_set_program_mythread() depending on mix of thread calls used.
  - This was because pmapi was causing all new threads to inherit the parents context.
  - Fix was to call pm_delete_program_mythread() and then call the above.

2.3.5 Release

- Latest PerfCtr 2.4 and PerfCtr 2.6 for:
  - Opteron
  - Xeon
- Additional Xeon events
- Latest pfmon/perfmon for IA64
- Bug fixes
- X1 port
- Build from external PerfCtr installation, not just external source tree.

PAPI 3 Design Goals

- Using lessons learned from earlier releases:
  - Eliminate unused features
  - Add functionality where needed
  - Respond to user requests
- Redesign for:
  - Robustness
  - Feature set flexibility
  - Simplicity and speed
  - Portability to new platforms

New PAPI 3 Functionality

- New Substrates: AMD Opteron and Cray X1
- Streamlined overhead on calls to start/ stop and read counters.
- Better native event support:
  - This:

```
PAPI_event_name_to_code("PM_FPU0_FDIV", &native);
PAPI_add_event(EventSet, native);
```

  - Instead of this:

```
native =  0 | 10 << 8 | 0; /* PM_FPU0_FDIV */
PAPI_add_event(EventSet,native);
```

- Bipartite counter allocation. If a mapping exists, PAPI will find it.
- Multiple simultaneous overflow and profile.
- Version Macros to obtain Major, Minor and Revision level of the PAPI library.

PAPI 3 High Level Changes

- The PAPI High-level interface is now thread safe.
- High-level calls maintain better state for mixing calls.
- Now supports three rate-based calls:
  - **`PAPI_flops`** – Floating point Operations per Second*
  - **`PAPI_flips`** – Floating point Instructions per Second*
  - **`PAPI_ipc`** – Instructions per Cycle

* Instructions typically measure what goes through the floating point execution unit, while Operations measure the theoretically expected number of floating point arithmetic operations.

PAPI 3 API Changes

- Deprecated calls:
  - **PAPI_add_pevent, PAPI_rem_pevent**
  - **PAPI_query_(all)_events_verbose**
  - **PAPI_describe_event, PAPI_label_event**
  - **PAPI_get_overflow_address**
  - **PAPI_save, PAPI_restore**
- Modified calls:
  - **PAPI_add_event(s)**
  - **PAPI_lock, PAPI_unlock**
  - **PAPI_cleanup_eventset**
  - **PAPI_initialized PAPI_is_initialized**
  - **PAPI_rem_event(s) PAPI_remove_event(s)**
  - **PAPI_get_mem_info PAPI_get_hardware_info**

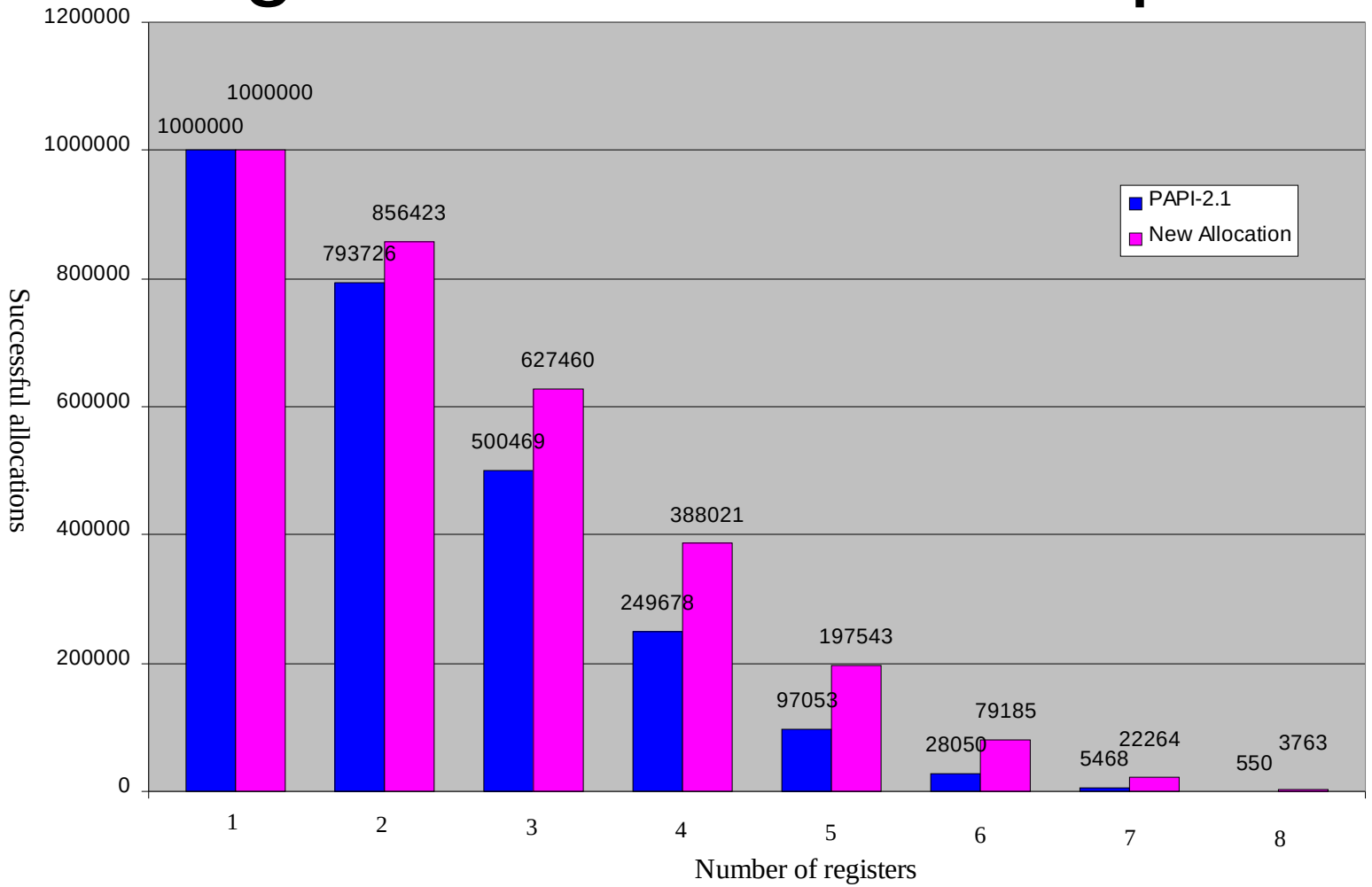
New PAPI 3 API Calls

- New event description calls work on both native and preset events:
  - **`PAPI_enum_event`**
  - **`PAPI_get_event_info`**
- New thread storage and registration events:
  - **`PAPI_get_thr_specific`**
  - **`PAPI_set_thr_specific`**
  - **`PAPI_register_thread`**
- Other new events:
  - **`PAPI_num_events`**
  - **`PAPI_get_shared_lib_info`**

Better Register Allocation

- On most CPUs, counter registers are scarce
- Often, not all events can be counted on all registers
- As the number of simultaneously counted events increases, effective mapping of registers to events becomes increasingly important
- PAPI 2 used a ‘ greedy’ or opportunistic allocation scheme: many theoretical mappings failed
- PAPI 3 implements a bipartite maximal

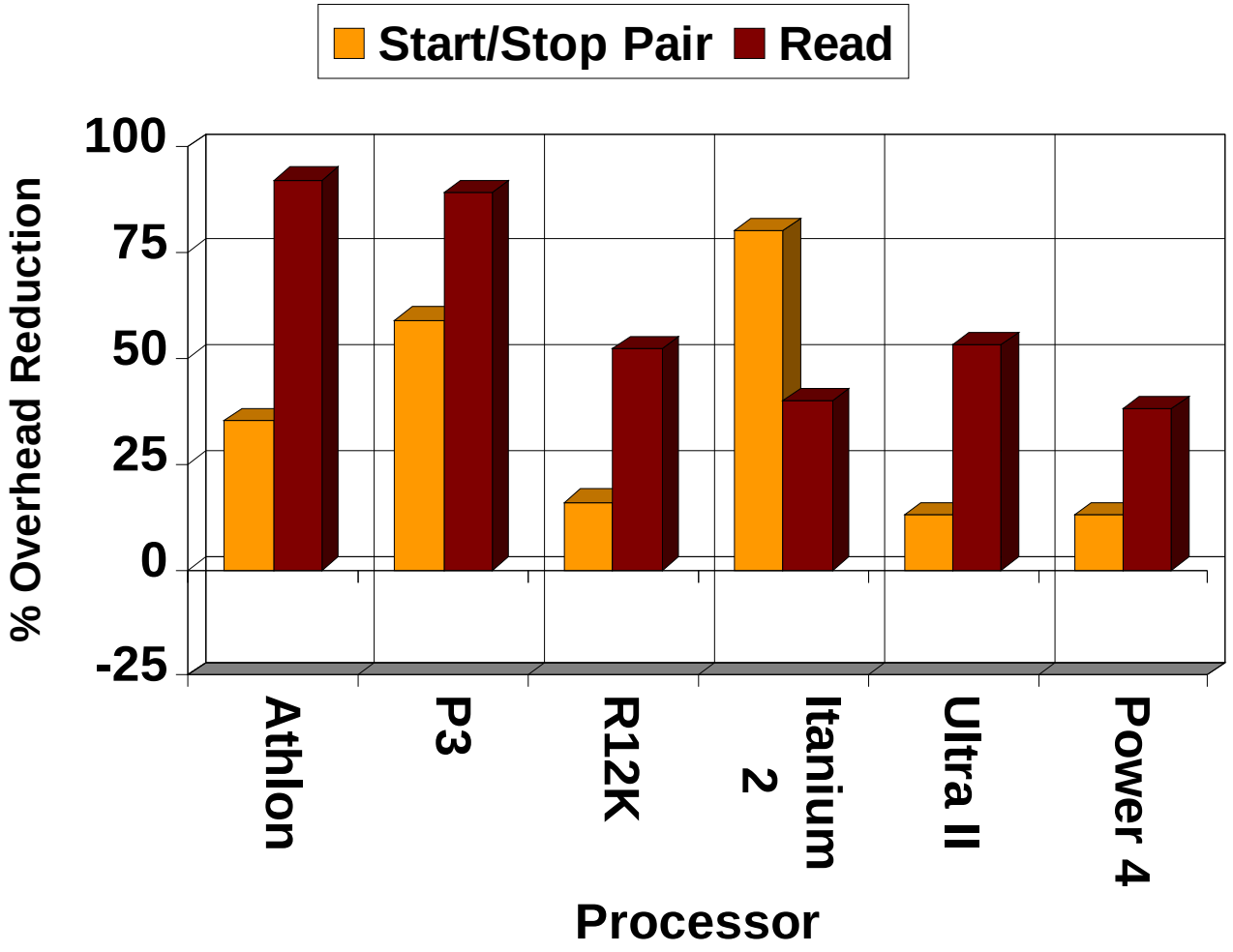
Register Allocation Comparison



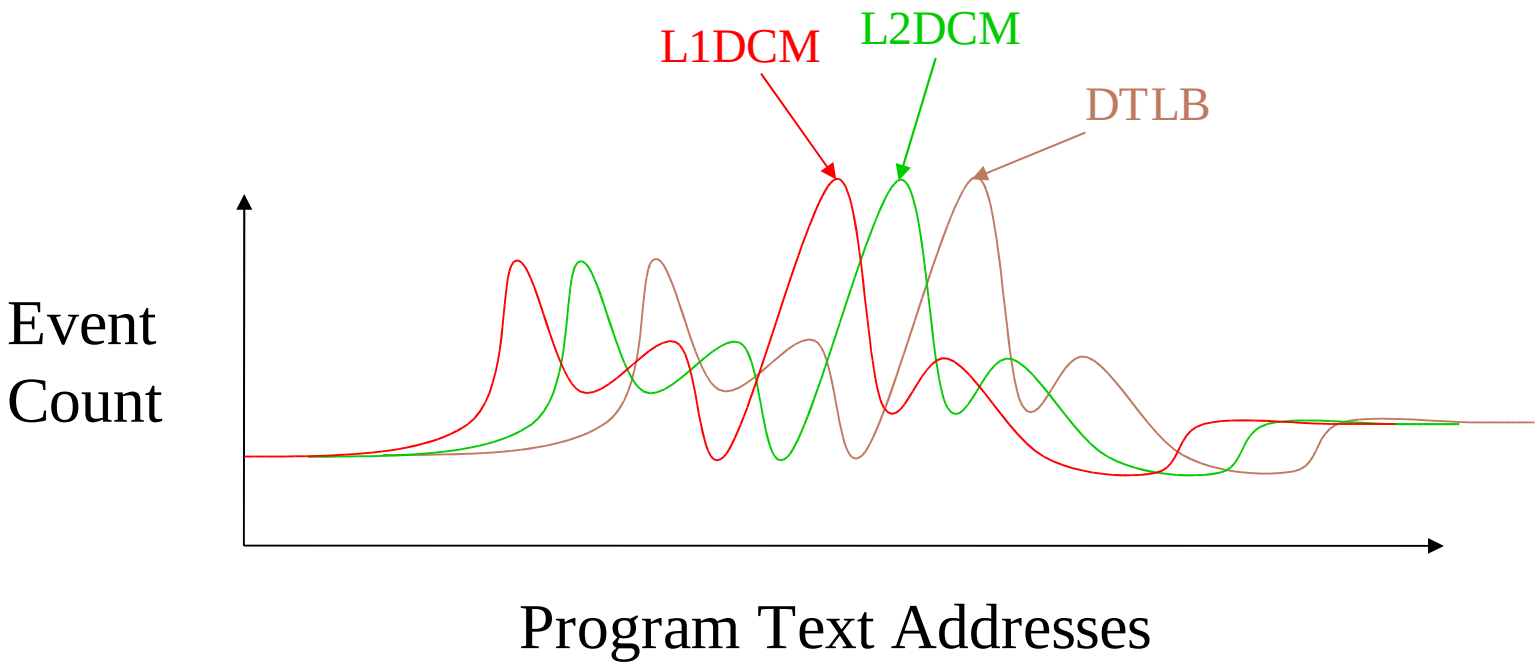
The new bipartite allocation scheme maps many more events for larger event sets than the old opportunistic scheme.

Overheads: PAPI 3 vs PAPI 2.3.4

((PAPI 2.3.4 overhead – PAPI 3.0 overhead)/PAPI 2.3.4 overhead)*100%



N Counter Statistical Profiling



Feature Requests

- Efficient timestamp with PAPI_read()
  - PAPI_read_ts() maybe
- P4/SSE events
  - LLNL, UIUC
- Opteron Memory reference events
  - Loads/Stores
- Variance metrics in ctests/cost

Latest RedHat IA64

- Red Hat Enterprise Linux 3.0 broke kernel support for the hardware counter infrastructure.
  - First update of RHEL will include a fix

Message to developers

- Improve and coalesce documentation
- Per Platform Installation Guide on Web
- Power 4 event map
- Pentium IV event map
- Opteron Loads/Stores