Experiences and Lessons Learned with a Portable Interface to Hardware Performance Counters

Jack Dongarra, Kevin London, Shirley Moore, Philip Mucci, Daniel Terpstra, Haihang You, and Zhou Min

INNOVATIVE COMPUTING LABORATORY

Tools for Performance Evaluation

- Timing and performance evaluation has been an art
  - Resolution of the clock
  - Issues about cache effects
  - Different systems
  - Can be cumbersome and inefficient with traditional tools
- Situation about to change
- Almost all high performance processors include hardware performance counters.
  - Some are easy to access, others not available to users.
- On most platforms the APIs, if they exist, are not appropriate for the end user or well documented.

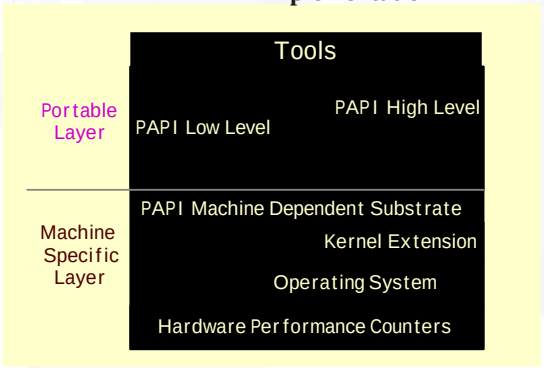
PAPI PERFORMANCE APPLICATION PROGRAMMING INTERFACE

- PAPI is a proposed "standard" cross-platform interface to hardware performance counters.
- PAPI provides two APIs to access the underlying performance counter hardware:
  - A low-level interface designed for tool developers and expert users, and
  - A high-level interface for application engineers.

Hardware Counters

- Small number of registers dedicated for performance monitoring functions
  - AMD Athlon, 4 counters
  - Pentium <= III, 2 counters
  - Pentium IV, 18 counters
  - IA64, 4 counters
  - Alpha 21x64, 2 counters
  - Power 3, 8 counters
  - Power 4, 8 counters
  - UltraSparc II, 2 counters
  - MIPS R14K, 2 counters

PAPI Implementation



PAPI Preset Events

- Proposed standard set of event names deemed most relevant for application performance tuning
- Exact standardization of the semantics not possible
  - eg IBM's FMA
- PAPI supports approximately 100 preset events.
- Mapped to native events on a given platform
  - Preset events are mappings from symbolic names to machine specific definitions for a particular hardware event.
    - Example: **`PAPI_TOT_CYC`**
  - PAPI also supports presets that may be derived from multiple underlying hardware metrics.
    - Example: **`PAPI_L1_DCM`**

Sample Preset Listing

```
.tests/avail
Test case 8: Available events and hardware information.
-------------------------------------------------------------------------
Vendor string and code   : GenuineIntel (-1)
Model string and code    : Celeron (Mendocino) (6)
CPU revision             : 10.000000
CPU Megahertz            : 366.504944
-------------------------------------------------------------------------
Name            Code                    Avail     Deriv     Description (Note)
PAPI_L1_DCM     0x80000000              Yes       No        Level 1 data cache misses
PAPI_L1_ICM     0x80000001              Yes       No        Level 1 instruction cache
                                                            misses
PAPI_L2_DCM     0x80000002              No        No        Level 2 data cache misses
PAPI_L2_ICM     0x80000003              No        No        Level 2 instruction cache
                                                            misses
PAPI_L3_DCM     0x80000004              No        No        Level 3 data cache misses
PAPI_L3_ICM     0x80000005              No        No        Level 3 instruction cache
                                                            misses
PAPI_L1_TCM     0x80000006              Yes       Yes       Level 1 cache misses
PAPI_L2_TCM     0x80000007              Yes       No        Level 2 cache misses
PAPI_L3_TCM     0x80000008              No        No        Level 3 cache misses
PAPI_CA_SNP     0x80000009              No        No        Requests for a snoop
PAPI_CA_SHR     0x8000000a              No        No        Requests for shared cache line
PAPI_CA_CLN     0x8000000b              No        No        Requests for clean cache line
PAPI_CA_INV     0x8000000c              No        No        Requests for cache line inv.
.
.
```

http://icl.cs.utk.edu/projects/papi/files/html_man/papi_presets.html

Support for Native Events

- PAPI supports native events:
  - An event countable by the CPU can be counted even if there is no matching preset PAPI event.
  - The developer uses the same API as when setting up a preset event, but a CPU-specific bit pattern is used instead of the PAPI event definition.

High-level Interface

- Meant for application programmers wanting coarse-grained measurements
- As easy to use as SGI IRIX prefex calls
  - a command-line interface to the R10000 hardware performance counters
- Requires no setup code
- Restrictions:
  - Allows only PAPI presets
  - Not thread safe
  - Only aggregate counters

High-level API Calls

- `PAPI_flops(float *rtime, float *ptime, long_long *flpins, float *mflops)`
  - Wallclock time, process time, FP ins since start,
  - Mflop/s since last call
- `PAPI_num_counters()`
  - Returns the number of available counters
- `PAPI_start_counters(int *cntrs, int alen)`
  - Start counters
- `PAPI_stop_counters(long_long *vals, int alen)`
  - Stop counters and put counter values in array
- `PAPI_accum_counters(long_long *vals, int alen)`
  - Accumulate counters into array and reset
- `PAPI_read_counters(long_long *vals, int alen)`
  - Copy counter values into array and reset counters

Low-level Interface

- Increased efficiency and functionality over the high level PAPI interface
- Approximately 60 functions
- Thread-safe (SMP, OpenMP, Pthreads)
- Supports both preset and native events

Low-level Functionality

- API Calls for:
  - Counter multiplexing
  - SVR4 compatible profiling
  - Processor information
  - Address space information
  - Accurate and low latency timing functions
  - Hardware event inquiry functions
  - Eventset management functions
  - Static and dynamic memory information
  - Simple locking operations
  - Callbacks on user defined overflow threshold

PAPI 2.3.4 Release
April 14, 2003

Platforms
- IBM PPC604, 604e, Power 3, Power4, AIX 5
- Intel x86/Linux, Windows, including Pentium IV
- Sun UltraSparc I/II/III
- SGI MIPS R10K/R12K/R14K
- Compaq Alpha 21164/21264 with DADD/DCPI
- Itanium/Itanium2 Linux
- Cray T3E

- Enhancements
  - Static/dynamic memory info
  - IA64 hardware profiling and sampling
  - Misc bug fixes
- Sample Tools
  - Perfometer
  - Trapper
  - Dynaprof

Design and Implementation Experiences

- Success of community-based open source development effort
  - Parallel Tools Consortium http://www.ptools.org/
- Tradeoffs between ease-of-use and increased functionality and features
- Operating system support
- Interfacing to third-party tools
- Data interpretation and accuracy issues
- Efficiency and scalability issues

Operating System Support

- Perfctr kernel patch by Mikael Pettersson required for Linux/x86
  - Kernel modification has met resistance from some system administrators
  - Effort underway to get perfctr into mainstream Linux release
- Vendor cooperation has been good (in most cases)
  - Register level operations code provided by Cray
  - IBM pmtoolkit included in AIX 5
  - Perfmon library from Hewlett-Packard for Itanium/Itanium2 Linux
  - DADD (Dynamic Access to DCPI Data) extension to DCPI from Hewlett-Packard for Alpha Tru64 UNIX

Tools

- Tools developed by the PAPI project
  - Dynaprof
  - Perfometer
- Third-party tools
  - HPCView (Rice University)
  - SvPablo (University of Illinois)
  - TAU (University of Oregon)
  - Vampir 3.x (Pallas)
  - VProf (Sandia National Lab)
  - Others (see PAPI home page)

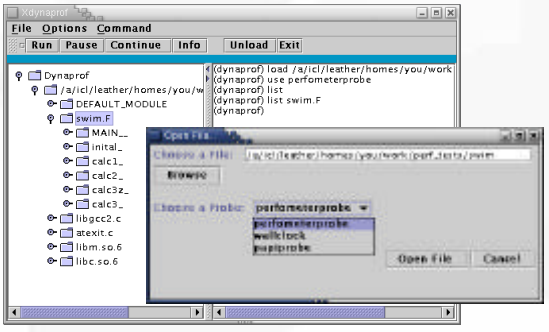
Dynaprof

- A portable tool to dynamically instrument serial and parallel programs for the purpose of performance analysis
- Simple and intuitive command line interface like GDB
- Java/Swing GUI
- Instrumentation is done through the run-time insertion of function calls to specially developed performance probes.

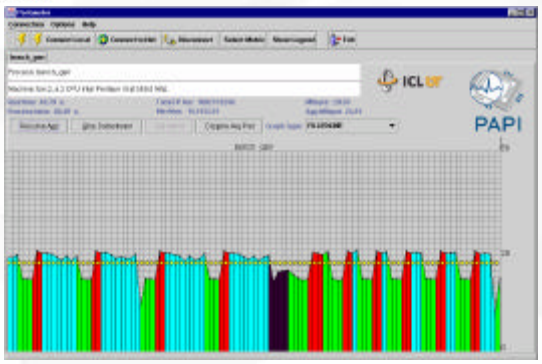
- Avoiding source-code instrumentation and recompilation
- Avoiding perturbation of compiler optimizations
- Providing complete language independence
- Built on DynInst and DPCL
  - IBM and Maryland

No source code required!

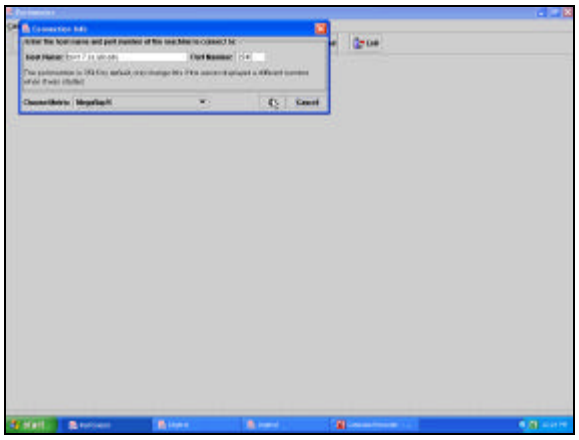
Dynaprof GUI Screenshot



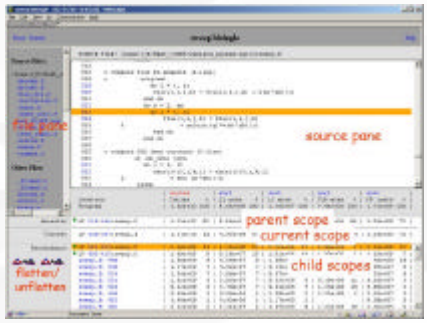
Perfometer Screenshot



April 26, 2003 IPDPS/PADTAD 2003 19

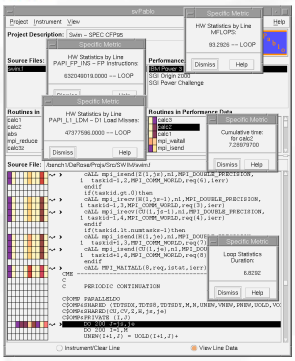


HPCView Screenshot



April 26, 2003 IPDPS/PADTAD 2003 21

SvPablo from UIUC

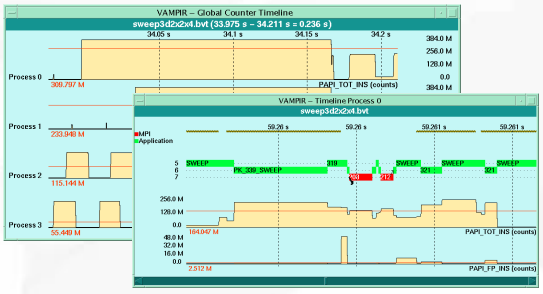


- Source based instrumentation of loops and function calls for Fortran and C
- Profiling statistics based on time and/or hardware counter data
- Supports serial, MPI, and OpenMP programs
- Freely available

April 26, 2003 IPDPS/PADTAD 2003 22

Vampir 3.x from Pallas

http://www.pallas.com/e/products/vampir/index.htm



April 26, 2003 IPDPS/PADTAD 2003 23

Data Accuracy Issues

- Act of measuring perturbs the system being measured
  - Extra instructions
  - Cache pollution
  - Servicing interrupts
- PC sampling can be inaccurate on out-of-order processors with speculative execution.
- Solutions:
  - PAPI is being redesigned to keep its runtime overhead and memory footprint as small as possible.
  - Hardware support for interrupt handling and profiling (e.g., event address registers) is being used where available.
  - Work by Pat Teller at University of Texas -El Paso on validation of hardware counter data using microbenchmarks

April 26, 2003 IPDPS/PADTAD 2003 24

PAPI Version 3 (expected June 2003)

- Using lessons learned from years earlier
- Redesign for:
  - Robustness
  - Feature set
  - Simplicity
  - Portability to new platforms
- New features
  - Multiway multiplexing
    - Use all available counter registers instead of one per time slice. (Just 1 additional register means 2x increase in accuracy)
      - Effective collection of 5 events on 4 counters
  - Improved performance
    - Pentium 4, a PAPI_read() costs 230 cycles.
      - Today can be as much as 3000 cycles
    - Register access alone costs 100 cycles.

PAPI Version 3 (cont.)

- New features (cont.)
  - Programmable events
  - Third-party interface
    - Allow control of counters in other threads of execution
  - Internal timer/signal/thread abstractions
  - Static and dynamic memory utilization information
  - Advanced profiling functions for event address sampling (branch, cache, etc...)
  - System-wide counting
  - High level API made thread safe
  - Optimal counter allocation scheme
  - Papirun utility
- Additional platforms
  - Cray X1
  - AMD Opteron/K8

Conclusions

- PAPI has been widely adopted by application and tool developers.
  - Use of PAPI simplifies collection and interpretation of hardware counter data by application developers.
  - Use of PAPI allows tool developers to focus on tool design rather than expending redundant effort on implementing low-level access to hardware counters.
- Data must be accurate to be useful.
  - Keep perturbation small.
  - Validate results.
- Counter access must be efficient and scalable.
  - Eliminate unnecessary features to streamline the interface (PAPI Version 3)
  - Make use of available hardware support for sampling, interrupt handling, etc.

For More Information

- http://icl.cs.utk.edu/papi/
  - Software and documentation
  - Reference materials
  - Papers and presentations
  - Third-party tools
  - Mailing lists