PAPI 5: Measuring Power, Energy, and the Cloud

Vincent M. Weaver*, Dan Terpstra†, Heike McCraw†, Matt Johnson†, Kiran Kasichayanula†, James Ralph†,
John Nelson†, Phil Mucci†, Tushar Mohan§, and Shirley Moore‡
*Electrical and Computer Engineering, University of Maine
†Innovative Computing Lab, University of Tennessee
‡Computer and Computational Sciences, University of Texas at El Paso
§Minimal Metrics

I. Introduction

The PAPI library [1] was originally developed to provide portable access to the hardware performance counters found on a diverse collection of modern microprocessors. Rather than learning and writing to a new performance infrastructure each time code is moved to a new machine, measurement code can be written to the PAPI API which abstracts away the underlying interface.

Over time, other system components besides the processor have gained performance interfaces (for example, GPUs and network interfaces). PAPI was redesigned to have a component architecture to allow modular access to these new sources of performance data [2].

In addition to incremental changes in processor support, the recent PAPI 5 release adds support for two emerging concerns in the high-performance landscape: energy consumption and cloud computing.

As processor densities climb, the thermal properties and energy usage of high performance systems are becoming increasingly important. We have extended the PAPI interface to simultaneously monitor processor metrics, thermal sensors, and power meters to provide clues for correlating algorithmic activity with thermal response and energy consumption.

We have also extended PAPI to provide support for running inside of Virtual Machines (VMs). This ongoing work will enable developers to use PAPI to engage in performance analysis in a virtualized cloud environment.

II. New Features

The recent PAPI 5.0 and 5.1 releases have added many new features over PAPI 4.4.

A. Improved CPU Support

PAPI 5 provides better support for Intel SandyBridge, Ivy Bridge, Cedarview Atom, and Xeon Phi architectures. Support has been added for Intel Offcore Response and Uncore Events.

PAPI now supports Blue Gene/Q (BG/Q), the third generation in the IBM Blue Gene line of massively parallel, energy efficient supercomputers. The BG/Q predecessor, Blue Gene/P, suffered from incompletely implemented hardware performance monitoring tools. To address these limitations, an industry/academic collaboration was established to extend PAPI with five new components that allow hardware performance counter monitoring of the 5D-Torus network, the I/O system and the Compute Node Kernel in addition to the processing cores on BG/Q.

B. New Interfaces

The previous limit of 16 components has been lifted, allowing for a much richer collection of measurement options. PAPI can now report results other than unsigned 64-bit integer, such as signed integer, fixed point, and ratios. Support is also included for reporting units, which becomes necessary with events that report power and energy. A document is available that describes all of the interface changes [3].

C. Power and Energy Components

Energy and power have become increasingly important components of overall system behavior in high-performance computing (HPC). Now that HPC machines have hundreds of thousands of cores [4], the ability to reduce consumption by just a few Watts per CPU quickly adds up to major power, cooling, and monetary savings.

There are some limitations when measuring power and energy using PAPI. Typically these readings are system-wide: it is not possible to exactly map the results to the user's code, especially on multi-core systems.

1) Intel RAPL: Recent Intel SandyBridge chips include the "Running Average Power Limit" (RAPL) interface. Internal circuitry can estimate current energy usage based on a model driven by hardware counters, temperature, and leakage models. The results of this model are available to the user via a model specific register (MSR), with an update frequency on the order of milliseconds. Linux has no RAPL driver, so we must use the "MSR driver" that exports MSR access to userspace. If the MSR driver is given proper read-only permissions then PAPI can access these registers without needing kernel support.

2) NVIDIA Management Library: Recent NVIDIA GPUs can report power usage via the NVIDIA Management Library (NVML) [5]. The `nvmlDeviceGetPowerUsage()` routine exports the current power; on Fermi C2075 GPUs it is updated at 60Hz with milliwatt precision and ± 5 Watt absolute accuracy. PAPI can use this interface to report power for the entire board, including GPU and memory.

3) Xeon Phi / MIC: The PAPI MIC power component exposes instantaneous voltage and current data collected by an onboard system management controller (SMC); the measurements are from an analog sensor, not a model. The SMC also provides an averaged total power utilization over two time windows.

D. Virtualization

Cloud computing involves use of a hosted computational environment that can provide elastic compute and storage services on demand. Virtualization is a technology that allows multiple virtual machines (VMs) to run on a single physical machine and share its resources. Virtualization is increasingly being used in cloud computing to provide economies of scale, customized environments, fault isolation, and reliability.

PAPI 5 addresses various aspects of measuring performance in the cloud.

1) Timing and Stealtime: PAPI aims to use the best and most accurate timers exposed by each VMM to implement a uniform timing interface that can be used across VMMs. This timer standardization will allow the same timing code to be used from within an application regardless of which native or virtualized environment it is running on. Some VMMs support the notion of virtualized time, for example called "apparent time" by VMware [6], whereby the virtual machine can have its own idea of time.

Most processors have a built-in hardware clock that allows the operating system to measure real and process time. Real time, also called elapsed or wall clock time, is the time according to an external standard since some fixed point such as the start of the life of a process. Process time is the amount of the CPU time used by a process since it was created. Process time is broken down into user CPU time (also called virtual time), which is the amount of time spent executing in user mode; and system CPU time, which is the amount of time spent executing in kernel mode. Measurement of process time can be useful for evaluating the performance of a program, including on a per-process or per-thread basis.

In order to improve the accuracy of CPU time accounting on virtual systems, the mechanism must be able to not only distinguish between real and virtual CPU time but also recognize being in involuntary wait states. These wait states are referred to as "steal time". Linux KVM/Xen supports reporting steal time, and we have written a PAPI component for reporting it.

Steal time is only reported system-wide, so it is not possible to get fine-grained per-process results, thus making it hard for PAPI to auto-adjust results returned by `PAPI_get_virt_usec()`. Stealtime is only an issue if a machine is over-subscribed with VMs, and in most HPC situations only one task is running per node, so this might not be a critical limitation.

2) I/O Performance: Variable I/O performance has been found to significantly impact application performance in virtual environments [7], [8], [9]. We have developed a PAPI component (appio) for measuring IO performance at application level in virtual environments. This component intercepts calls to read, write, fread and fwrite and reports a variety of metrics on the size and amount of I/O taking place.

3) VMware Component: The PAPI VMware component exposes information provided from within VMware to users running inside as a guest. Values are gathered using VMware's Guest SDK [10], as well as what VMware refers to as "pseudo performance" counters [6]. VMware makes pseudo performance counters available through an `rdpmc` instruction to obtain fine-grained time from within the virtual space. These pseudo-performance counters are not enabled by default; they must be enabled and an environment variable (`PAPI_VMWARE_PSEUDOPERFORMANCE`) must be set before use.

4) Virtualized Processor Counters: The performance monitoring unit (PMU) of a processor typically includes a set of performance counter registers that count the frequency or duration of specific processor events, a set of performance event select registers used to specify the events that are tracked by the performance counter registers, a hardware interrupt that can be generated when a counter overflows, and a time stamp counter (TSC) that can be used to count processor clock cycles. On x86 hardware these interfaces are programmed via MSRs; typically these require ring-0 (kernel) levels of permission.

In order to transparently use performance counters (and PAPI) from within a VM the full MSR interface must be trapped and emulated. Recently support for doing this was added to KVM (in Linux 3.3) and Xen (Linux 3.5). Support is still undergoing beta testing at VMware.

III. Conclusion

The PAPI library provides transparent access to new classes of interfaces, including virtualized, power and energy measurements. Existing programs that already support PAPI instrumentation for CPU performance measurements can quickly be adapted to measure these new events with a simple PAPI upgrade.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. 0910899 and 1117058. Additional support for this work was provided through a Sponsored Academic Research Award from VMware, Inc.

References

[1] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A portable programming interface for performance evaluation on modern processors," *International Journal of High Performance Computing Applications*, vol. 14, no. 3, pp. 189–204, 2000.

[2] D. Terpstra, H. Jagode, H. You, and J. Dongarra, "Collecting performance data with PAPI-C," in *3rd Parallel Tools Workshop*, 2009, pp. 157–173.

[3] V. Weaver, "New features in the PAPI 5.0 release," University of Tennessee, Tech. Rep., 2012. [Online]. Available: http://www.eece.maine.edu/~vweaver/papers/papi/papi_v5_changes.pdf

[4] "Top 500 supercomputing sites," http://www.top500.org/.

[5] *NVML Reference Manual*, NVIDIA, 2012.

[6] *Timekeeping in VMware Virtual Machines*, VMware, Inc., 2010.

[7] A. Nanos, G. Goumas, and N. Koziris, "Exploring i/o virtualization data paths for MPI applications in a cluster of VMs: A networking perspective," in *Proc. 5th Workshop on Virtualization in High Performance Cloud Computing*, 2011.

[8] H. Kim, H. Lim, J. Jeong, H. Jo, and J. Lee, "Task-aware virtual machine scheduling for I/O performance," in *Proc. of the ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, 2009, pp. 101–110.

[9] D. Ongaro, A. Cox, and S. Rixner, "Scheduling I/O in virtual machine monitors," in *Proc. of the 4th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, 2008, pp. 1–10.

[10] *vSphere Guest SDK Documentation*, VMware, Inc., 2011.