What Does It Take to Keep PAPI Instrumental for the HPC Community?

Heike Jagode, Anthony Danalis, Jack Dongarra
{jagode | adanalis | dongarra}@icl.utk.edu

Innovative Computing Laboratory
University of Tennessee
Knoxville, TN, USA

White paper submission for the
2019 Collegeville Workshop on Sustainable Scientific Software (CW3S19).

Scientific application communities, who are invested in high-performance computing (HPC), have relied on the Performance Application Programming Interface (PAPI) to track low-level hardware operations for the past two decades. *The goal of this white paper is to identify and articulate opportunities and possible solutions for PAPI to remain sustainable and useful for the next two decades—and beyond.*

*A. What does the **classical PAPI library** stand for?*

PAPI provides tool designers and application engineers with a consistent interface and standardization layer for the use of low-level performance counter hardware found across the entire compute system (i.e., CPUs, GPUs, on/off-chip memory, interconnects, I/O system, energy/power). It enables users to see, in near real time, the relationships between software performance and hardware events across a system. Prior to the availability of PAPI, there was no standard mechanism for collecting detailed performance counter data.

B. How important is community involvement for PAPI?

PAPI has seen widespread, active use at National Science Foundation (NSF) and US Department of Energy (DOE) HPC centers, and its use spans groups working in exascale system design, scientific simulation, numerical methods, site management, and user support. Each group uses the data collected by PAPI to improve decision-making in their respective processes. Software and hardware designers make use of PAPI data to work around the bottlenecks found in today's hardware architectures. Simulation and application scientists use PAPI-based tools to understand, optimize, and scale their codes. Site directors leverage PAPI data to understand their workloads, optimize resource allocation, and plan for future procurements. In a nutshell, the scientific application and HPC communities have been instrumental to PAPI's growth, both in terms of usage and by soliciting vendors to make it available for their various hardware components and systems.

C. Why a new PAPI++ effort?

One of the many challenges ahead for programming in the exascale era is providing support for multiple hardware architectures from the same code base. The main programming problems to solve are portability and performance of codes, which are increasingly difficult to achieve as hardware architectures are becoming more and more diverse. According to several research teams that are part of the HPC community, capitalizing on the momentum behind C++ as well as achieving a standard, higher-level abstraction and programming model for parallelism in the language for heterogeneous environments is the best path toward meeting exascale requirements.

Our *Performance Application Programming Interface with Modern C++ (PAPI++)* effort develops a new software package from the ground up. With PAPI++, we converge and consolidate over 20 years of technology innovation in cross-platform performance metric monitoring and modeling software—which the PAPI team pioneered, implemented, and disseminated—into a performance counter library that integrates seamlessly into the HPC ecosystem and enables a wide range of applications to fully exploit the power of near-exascale and exascale systems as soon as they come online.

To put the new PAPI++ plan into perspective, the first PAPI version that offered a standardized, easy-to-use interface for accessing hardware performance counters was released in 1999. The past two decades witnessed tectonic shifts in hardware technology followed by paradigm shifts in software technology. During that time, PAPI has been repeatedly:

- "extended" with performance counter support for newly released CPUs,
- "redesigned" to enable hardware monitoring information that became available in other sub-systems throughout modern computer architectures (e.g., counters found in GPUs, on/off-chip memory, interconnects, I/O systems), and
- "upgraded" to extend PAPI's role further for monitoring power consumption as well as performance events that originate from other software layers.

No viable replacement for PAPI has emerged and established itself as the de facto standard for monitoring hardware counters, power usage, software-defined events, and channeling this technological progress into a robust software package. The PAPI++ package is meant to be this replacement—with a more flexible and sustainable software design.

*D. What does the **new PAPI++ library** stand for?*

The new PAPI++ software package will build on classic-PAPI functionality and strengthen its path to exascale with a more efficient and flexible software design—one which takes advantage of C++'s object-oriented nature, but still preserves the low-overhead monitoring of performance counters, and adds a vast testing suite. In addition to developing a new C++ Performance API from the ground up, PAPI++ will be extended with performance counter monitoring capabilities for new and advanced exascale hardware; fine-grained power management support; and Software-Defined Event (SDE) support for exposing performance-critical events that originate from different software layers.

The PAPI++ plan can be classified into three main efforts:

Effort 1: The core, and most elaborate, effort is the *development of a modular, object-oriented design for a new PAPI software package from the ground up, called PAPI++, that will leverage modern C++* as an essential element of scientific computing. This effort involves a managed transition away from our legacy PAPI software while continuing to add support for new hardware and software until the official release of PAPI++.

Since community involvement is a central part of our software development philosophy, as a first and important step, we are going to circulate a survey to the HPC applications and software technology teams to assess their needs for hardware and software performance counter functionality. Based on the survey results, we will develop a roadmap for the development of PAPI++, perform a software requirement analysis, and start the PAPI++ design process for a modular framework that includes a new C++ PAPI interface, in addition to the traditional PAPI C and Fortran interfaces to preserve backward-compatibility.

Effort 2: *Developing performance counter monitoring capabilities for new and advanced HPC hardware and software technologies that will be released in the near future.* This involves close collaboration with hardware vendors (early in the design cycle), and pursuing "near-exascale" efforts that focus on cooperative hardware and software advancements.

Effort 3: *The deployment and extension of "Software-Defined Events":* With software complexity being one of the fundamental issues the community will face at exascale, it is one of our central goals to close the gap between software-based event monitoring and hardware performance counter monitoring. This effort is our most recent endeavor to extend PAPI's usability for the HPC community. PAPI++ integrates a standardizing layer for defining and monitoring software-based events that expose the internal behavior of runtime systems and libraries to the applications. Addressing the gap of SDE monitoring—and enabling monitoring of both types of performance events through PAPI—stands to offer a transformative impact on performance analysis and application development as a whole.

Redesigning PAPI now—having 20 years worth of experience and a fair understanding of the future roadmap of HPC software and hardware—gives us the opportunity to ensure that PAPI will remain sustainable and useful for the next two decades, and beyond. To name a few examples that span the three efforts mentioned above, PAPI++ will be designed to seamlessly support features that are currently out of scope, such as:

- counters that are not of a fixed type (currently PAPI is limited to "`long long int`"),
- SDEs that go beyond the notion of a simple counter by allowing arbitrary information to be exported by software layers,
- preset events for non-CPU hardware, so that events from other important hardware platforms, such as GPUs, can also be abstracted into simpler, representative categories,
- modern hardware behaviors that do not map well to CPU behaviors, such as the multiple replay passes that are required to calculate some GPU-derived metrics.

E. How about an official "red team" of independent software reviewers?

As mentioned above, an important aspect of PAPI's sustainability is our interaction with the community. The PAPI team has a solid reputation for openness, is willing to listen to stakeholder ideas and concerns, and eager to receive feedback and code contributions. As part of the PAPI++ effort, we have planned and budgeted to hire an official "red team" of experts who (a) have been using PAPI, and (b) are willing to give us feedback on our new design decisions early in the design cycle and long before any code release. The goal of having a *red team of independent consultants and reviewers* for a new open-source software project is to make sure we are developing what the community needs—precisely by involving community experts—and that new functionalities and design decisions are implemented and coded proficiently from the beginning.

F. Who will benefit from the new PAPI++ library?

The sustained deployment of PAPI over the years confirms the validity of having one middleware interface (PAPI) that provides a consistent platform, as well as operating system–independent access to hardware performance counters within CPUs, GPUs, interconnects, and the system as a whole. This means that third-party tools and applications have only had to handle a single hook to PAPI in order to access all performance metrics in a system. It has been PAPI's responsibility to efficiently and correctly handle all necessary details for each platform and system component.

With the ground-up development of a new PAPI++ software package, by leveraging modern C++ and extending the functionality of PAPI's abstraction and unification layer into the realm of a more sustainable software design, this project stands to strengthen the ability of the high-level performance toolkits that utilize PAPI. Although PAPI can be used independently as a performance monitoring library and tool for application analysis, it has found its greatest utility as a middleware component for a number of third-party profiling, tracing, and sampling toolkits, such as Caliper, CrayPat, HPCToolkit, Scalasca, Score-P, TAU, Vampir. Ultimately, all users of PAPI++, regardless of direct use or use through end-user tools, will benefit from the set of innovations we will make available through this effort.

Without the PAPI++ effort, the HPC community would lack a consistent, standard interface that offers the ability to not only monitor performance events for next-generation hardware, but also to manage power/energy and export software-critical events from HPC libraries—all in a uniform way. Without PAPI++, software developers are destined to use multiple APIs to access hardware counters from across the system, which, ultimately, damages productivity. As a result, performance assessment and improvement for multiple vendor platforms would become exceedingly difficult.