Formulation of Requirements for new PAPI++ Software Package

Part I: Survey Results

Heike Jagode
Anthony Danalis
Jack Dongarra

Innovative Computing Laboratory (ICL)

January 31, 2020

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Revision	Notes
01-2020	first publication

```
@techreport{2020-01-exa-papi,
  author={Jagode, Heike and Danalis, Anthony and Dongarra, Jack},
  title={Formulation of Requirements for new {PAPI++} Software Package, {PDN} No. 1},
  institution={Innovative Computing Laboratory, University of Tennessee Knoxville},
  year={2020},
  month={January},
  number={ICL-UT-20-02},
}
```

Contents

CHAPTER 1

Introduction

The Exascale Performance Application Programming Interface (Exa-PAPI) project [1] is developing a new C++ Performance API (PAPI++) software package from the ground up that offers a standard interface and methodology for using low-level performance counters in CPUs, GPUs, on/off-chip memory, interconnects, and the I/O system, including energy/power management. PAPI++ is building upon classic-PAPI functionality and strengthening its path to exascale with a more efficient and flexible software design, one that takes advantage of C++ object-oriented nature but preserves the low-overhead monitoring of performance counters and adds a vast testing suite.

Scientific application communities, who are invested in high-performance computing (HPC), have relied on the "classic" Performance Application Programming Interface (PAPI) to track low-level hardware operations for the past two decades (PAPI was first released in 1999). The goal of the Exa-PAPI effort—starting with a PAPI survey and its results, summarized in 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.

[1] https://icl.utk.edu/exa-papi/

CHAPTER 2

Background

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.

2.1 Why a new PAPI++ effort?

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 and capping 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.

2.2 Going Beyond Hardware Counters

In addition to providing hardware counter-based information through PAPI++, a standardizing layer for monitoring software-defined events (SDE) is being incorporated into the "Performance API" to enable easy exposure of the internal behavior of runtime systems and libraries, such as communication and math libraries, to the applications. As a result, the notion of performance events is broadened from strictly hardware-related events to include software-based information. Enabling monitoring of both hardware and software events provides more flexibility to developers when capturing performance information.

2.3 Summary

In summary, the Exa-PAPI team is preparing PAPI support to stand up to the challenges posed by exascale systems by:

(1) widening its applicability and providing robust support for exascale hardware resources;

(2) supporting finer-grain measurement and control of power, thus offering software developers a basic building block for dynamic application optimization under power constraints;

(3) extending PAPI functionality to support software-defined events; and

(4) applying semantic analysis to hardware counters so that the application developer can better make sense of the ever-growing list of raw hardware performance events that can be measured during execution.

The team will be channeling the monitoring capabilities of hardware counters, power usage, software-defined events into a robust PAPI++ software package. PAPI++ is meant to be PAPI's replacement—with a more flexible and sustainable software design.

CHAPTER 3

ECP Applications and Software Technologies Survey

In January 2020, the Exa-PAPI team circulated a survey to the Exascale Computing Project (ECP) applications (AD) and software technology (ST) teams to assess their needs and requirements for hardware and software performance counter functionality. Twenty responses were collected. Here, the responses to the most important questions are summarized.

3.1 Results

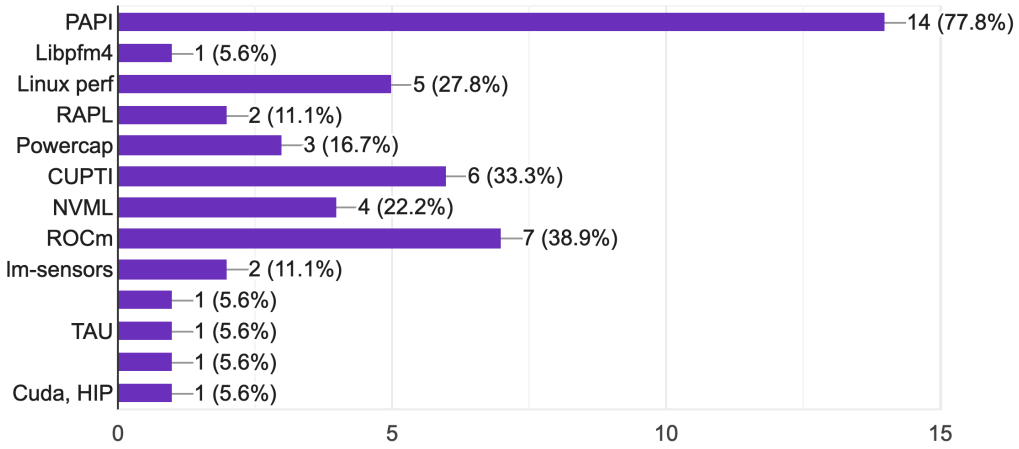

Figure 3.1: Is your application / tool / software directly calling any of the following packages? Mark all that apply.

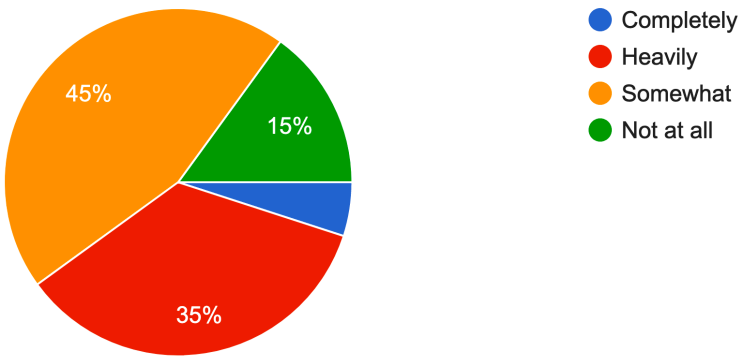

Figure 3.2: To what extent does your application / tool / software rely on performance counter monitoring?

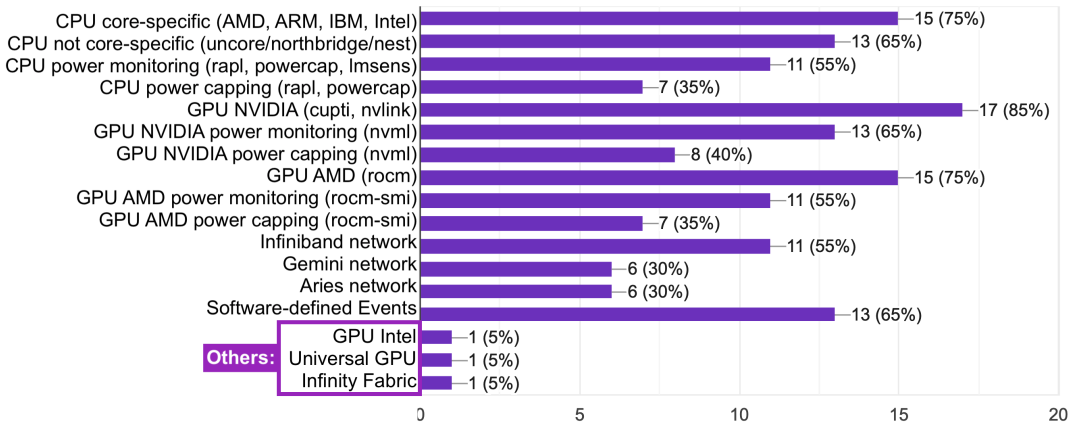

Figure 3.3: What performance counters are you using / interested in? Mark all that apply.

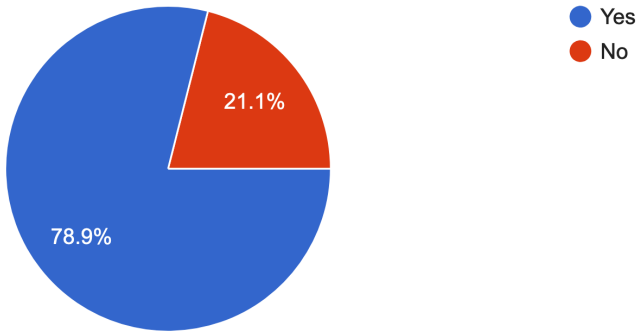

Figure 3.4: Are you interested in combining performance counter monitoring from different architectures (CPUs, GPUs, networks, power, etc.)?

Figure 3.5: What type of PAPI events are you using / interested in? Mark all that apply.

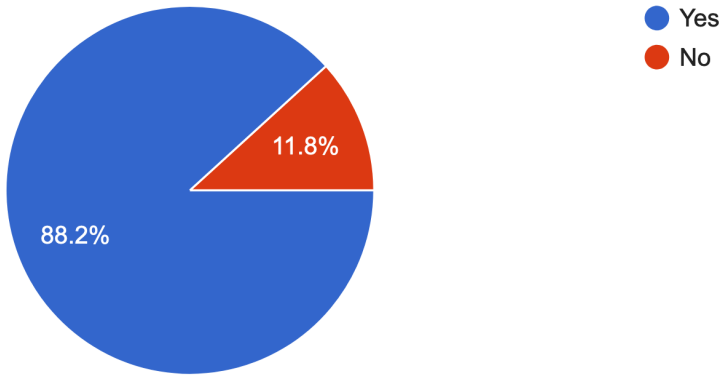

Figure 3.6: Currently PAPI supports presets events for CPUs only. Are you interested in similar presets for non-CPU components (e.g. AMD and NVIDIA GPUs)?

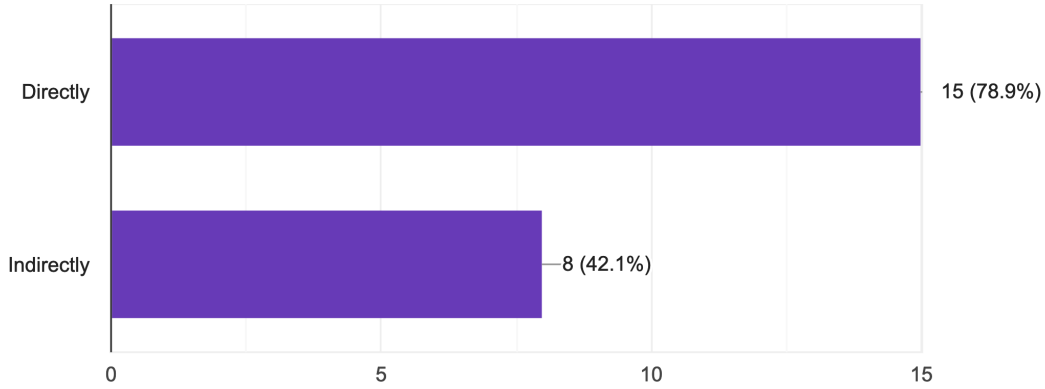

Figure 3.7: Do you prefer to use PAPI directly or indirectly via 3rd-party performance tools (e.g. TAU, VAMPIR, Scalasca, Paraver)?.

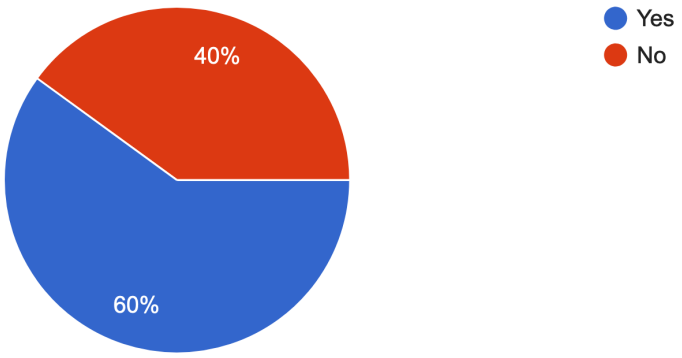

Figure 3.8: If you answered "directly", do you want PAPI to generate a JSON output file with measurement results?

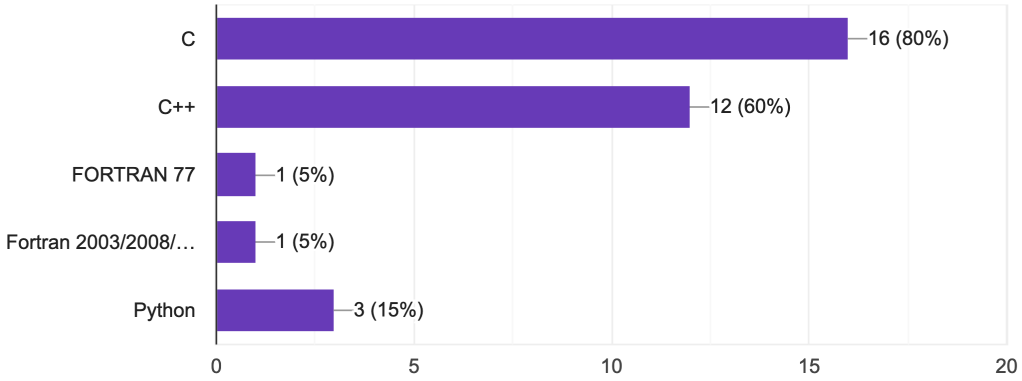

Figure 3.9: Which API do you need? Mark all that apply.

Other comments or feature requests for PAPI / PAPI++ are:

- We need working tools on GPUs, all vendors, and preferably the same counters. e.g. To submit a Gordon Bell prize application. CPU performance does not look very relevant for ECP target platforms, but we do still need to need to keep an eye on e.g. single thread performance. A successful tool will allow us to do the same thing on multiple platforms.
- Are there counters for overflow happening, e.g., in FP16 arithmetic.
- We don't require PAPI, but it could be useful to use for performance analysis, and SDEs could be nice to have as an option to report internal measurements such as number of iterations, though this is less of a concern than in sparse linear algebra.
- Number of flops, data communicated between CPU and GPU and via MPI, and power use over time would probably be the measurements of most interest for us.
- Counter reproducibility/stability and verification is very relevant for us, we need to know what the variance is, and be sure that the values are correct.

CHAPTER 4

Consequences for PAPI++

This summary is based on the results of the survey as well as follow-up interactions with various application and software technology teams. Here we summarize the main observations.

Impact: In order to reach exascale performance levels, inevitably, one has to be able to measure if and how efficient the compute power is utilized. Many ECP applications and software technology projects—specifically, 85% of our survey participants—are relying on performance counter monitoring (either completely, heavily, or somewhat) to measure progress toward better performance and resource utilization. 78% of the respondents indicate reliance on the PAPI library for performance counter analysis. Only 15% of the correspondents indicated no need for performance counter monitoring, but were interested in newer features, such as software-defined events (SDEs). In fact, 67% demonstrated interest in SDEs for their software projects to report internal measurements.

Target Hardware: The survey results clearly indicate a universal agreement that all types of architectures should be supported (3.3), including counter support for multi-core CPUs as well as heterogeneous, accelerated architectures, and also counter support for a variety of networks (Infiniband, Gemini, Aries, Infinity Fabric) is desired. Likewise, memory traffic and the data communicated between CPU and GPU was solicited and explicitly mentioned in the comment section for "feature requests". There is also a growing interest in monitoring energy consumption on GPUs. Between 55 and 65% of the respondents indicated interest in power use over time on AMD GPUs and NVIDIA GPUs, respectively.

Desired APIs: In general, APIs for all common HPC programming languages are needed: C, C++, Python, Fortran (legacy as well as modern versions). Nonetheless, the main runners are C++ and C, with 60% indicating a significant need in C++, and 80% in C interfaces.

In summary, the PAPI++ software needs to:

- serve as a fully-functional replacement for PAPI,
- provide a C++ API while maintaining traditional APIs,
- support preset events for non-CPU components (e.g. presets for AMD, Intel, and NVIDIA GPUs). 88% of the respondents were in favor of it,
- support monitoring of memory traffic and data communicated between devices,
- provide a way to combine different types of performance counters from different architectures in the same event set. 79% of the respondents were in favor of it.

4.1 Software engineering with C++

Historically speaking, the PAPI framework has been implemented in C and also provides a Fortran API. The complexity of modern hardware and software systems for HPC, however, necessitates the use of modern programming languages to ease the development process, avoid code repetition, and keep the volume of code that's exposed to changing requirements as minimal as possible. While there is no question of the robustness of procedural programming languages such as C and modern Fortran, when developing a new library from the ground up, modern software engineering demands generic programming, data abstraction and encapsulation, inheritance, to name but a few, all of which can be easily expressed with C++. It is only natural for the development of PAPI++ to adopted C++ as implementation language to leverage its support for object-oriented programming.

As for the language specification, for PAPI++ we are targeting C++11 or newer, as it introduces many new features, such as built-in atomic support, and is completely supported by the GNU, Intel, and LLVM compilers.

4.2 Team of independent reviewers

An important aspect of PAPI's sustainability is our interaction with the HPC 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 appoint 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.

4.3 Stakeholders of new PAPI++

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.