PAPI

PAPI Support for AI Architectures, Code Annotation, & PC Sampling

Daniel Barry

Scalable Tools Workshop

July 26-30, 2026

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Outline

1. Updates to PAPI's **`rocp_sdk`** Component:
   a. Honoring ROCTx Pause and Resume
   b. Delayed Initialization & Sync-before-Read
   c. Thread Safety
   d. Discussion: Integrating PC Sampling into PAPI

2. PAPI's **`amd_smi`** Component & Power Monitoring

3. Support for AI Architectures & PAPI's **`gaudi2`** Component

Thank you to AMD

- Thank you to the AMD Team (Sadik Armagan, Anthony Danalis, and Benjamin Welton) for their support explaining the features of the ROCProfiler-SDK.

- Thank you to University of Oregon's Performance Research Lab for allowing to use their OACISS testbed, without which we could not test PAPI components with a wide variety of hardware architectures.

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

Code Segment	Count
`PAPI_start()`	0

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

Code Segment	Count

`PAPI_start()`
 	
`gemm_on_gpu()`
	0

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

Code Segment	Count
`PAPI_start()`	0
`gemm_on_gpu()`	
`PAPI_read()`	32

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

Code Segment	Count
`PAPI_start()`	0
`gemm_on_gpu()`	
`PAPI_read()`	32
`gemm_on_gpu()`	

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

Code Segment	Count
`PAPI_start()`	0
`gemm_on_gpu()`	
`PAPI_read()`	32
`gemm_on_gpu()`	
`PAPI_stop()`	64

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

Code Segment	Count
`PAPI_start()`	0
`gemm_on_gpu()`	
`PAPI_read()`	32
`gemm_on_gpu()`	
`PAPI_stop()`	64

ROCTx Pause/Resume Support

- Monitoring event “rocp_sdk:::SQ_INSTS_VALU_FMA_F64”

Code Segment	Count
`PAPI_start()`	0
`roctxProfilerPause()`	
`gemm_on_gpu()`	
`roctxProfilerResume()`	
`PAPI_read()`	0
`gemm_on_gpu()`	
`PAPI_stop()`	32

How does this work internally?

- At ROCTx pause(), PAPI saves event counts from the active profiling context on the calling thread; and then it pauses the that thread's profiling context.

- At ROCTx resume(), PAPI resumes the profiling context for the thread that made the ROCTx call; and then restores the saved event counts for further profiling on that thread/context.

Delayed Initialization in `rocp_sdk`

- Tools that depend on PAPI need certain components (*e.g.*, `cuda`, `rocm`, `rocp_sdk`, *etc.*) to not be initialized until a user adds events to an event set:

1. Tools also try to initialize underlying APIs.
2. Prevent automatically initializing unused components.
3. The user actually intends to monitor the events of a given component.

Sync before PAPI_read()

- We provide a record_callback() to the ROCProfiler to get the counter values.

Challenge:

- There is no guarantee that record_callback() has completed before the array of counter values is accessed in PAPI_read().

Solution:

- We lock at the beginning of the dispatch_callback(), but don't unlock until the end of record_callback().
- PAPI_read() will not attempt to access the counter values until it can access the lock.

Thread Safety in `rocp_sdk`

- Multiple threads are now permissible.
  - Frontier@OLCF (AMD MI250X, ROCm 7.2.0)
  - Odyssey@Oregon (AMD MI300A, ROCm 7.2.4)

Challenge:

- Previously, two threads attempting to start counting events would result in zero-counts for both threads.

Solution:

- Atomically set active profiling context to operate with the first thread to call PAPI_start(), and release the context when that thread calls PAPI_stop().

PC Sample Format and Examples

From rocm-systems/projects/rocprofiler-sdk/samples:

- (code_obj_id, offset): (8, 0x2b08), timestamp: 897264657547l755, exec: 1, workgroup_id_(x= 276, y= 380, z= 0), wave_in_group: 0, chiplet: 1, dispatch_id: 711, correlation: {internal= 711, external= 0}, wave issued ROCPROFILER_PC_SAMPLING_INSTRUCTION_TYPE_SCALAR instruction, two VALU instructions issued: 0, arbiter state: {pipe issued: (VALU: 0, MATRIX: 0, LDS: 0, LDS_DIRECT: 0, SCALAR: 1, TEX: 0, FLAT: 0, EXPORT: 0, MISC: 0), pipe stalled: (VALU: 0, MATRIX: 0, LDS: 0, LDS_DIRECT: 0, SCALAR: 0, TEX: 0, FLAT: 0, EXPORT: 0, MISC: 0)}

PC Sample as a Struct

There are many fields…

- ID of code obj containing PC
- Offset within the code obj
- Timestamp of sample
- Dispatch ID
- Workgroup ID (X, Y, Z)
- Workgroup wave index
- Shader engine ID
- Compute unit ID
- SIMD ID
- Instruction type
- Stall reason
- Execution mask
- Logical GPU device ID
- Wave issued flag
- Chiplet ID

Discussion: PC Sampling

- For the tools community at large, what is a useful way for PAPI to expose PC sampling?
- Consider a PC sampling API in PAPI.
- Is it meaningful or helpful to expose this type of sample through a PC-sampling API in PAPI?

PAPI's `amd_smi` Component

- Thank you to Dong Jun Woun for his development of the `amd_smi` component.

AMD SMI & ROCm SMI

- Merged into PAPI in PR #481.
- Status & performance metrics for various GPU/APU models.
  - Power, Temperature, and Compute Unit Utilization, *etc.*
- AMD transitioned from ROCm SMI to AMD SMI
- AMD SMI: ROCm 6.4.0 ~ 7.2.4
- ROCm SMI went into Maintenance Mode since ROCm 7.0.0

Three Challenges with AMD SMI

1. Portability
2. Metric Availability
3. Low Overhead & Safety

AMD SMI: Portability

Challenge:

- **Frequent API deprecations** (ROCm-SMI), **symbol changes (ROCm 6.4–7.2)**, and **varying hardware capabilities** (MI210/MI250X/MI300A/MI300X) break tool compatibility.

Solution:

- Continuously **track ROCm updates**, **dynamically load library, Primary/Fallback symbol bind,** and **probe for metric availability** at initialization.
- AMD SMI component works seamlessly across different AMD GPU architectures and ROCm versions.
- PAPI requires no code changes when moving from one AMD GPU system to another.

AMD SMI: Metric Availability

Challenge:

- AMD SMI **lacks a metric enumeration API to list available metrics.**

Solution:

- **Probe-based discovery mechanism to verify available metrics** at runtime.
- Ensures **only metrics supported by the specific device and driver version are exposed** to the user.
- Expose a discoverable list of supported metrics via event names.

```
| amd_smi:::temp_current_sensor=1
|             Device 0 temp_current for sensor 1
|     :device=0
|             Mandatory device qualifier [0,1,2,3]
| amd_smi:::temp_critical_sensor=1
|             Device 0 temp_critical for sensor 1
|     :device=0
|             Mandatory device qualifier [0,1,2,3]
| amd_smi:::temp_emergency_sensor=1
|             Device 0 temp_emergency for sensor 1
|     :device=0
|             Mandatory device qualifier [0,1,2,3]
```

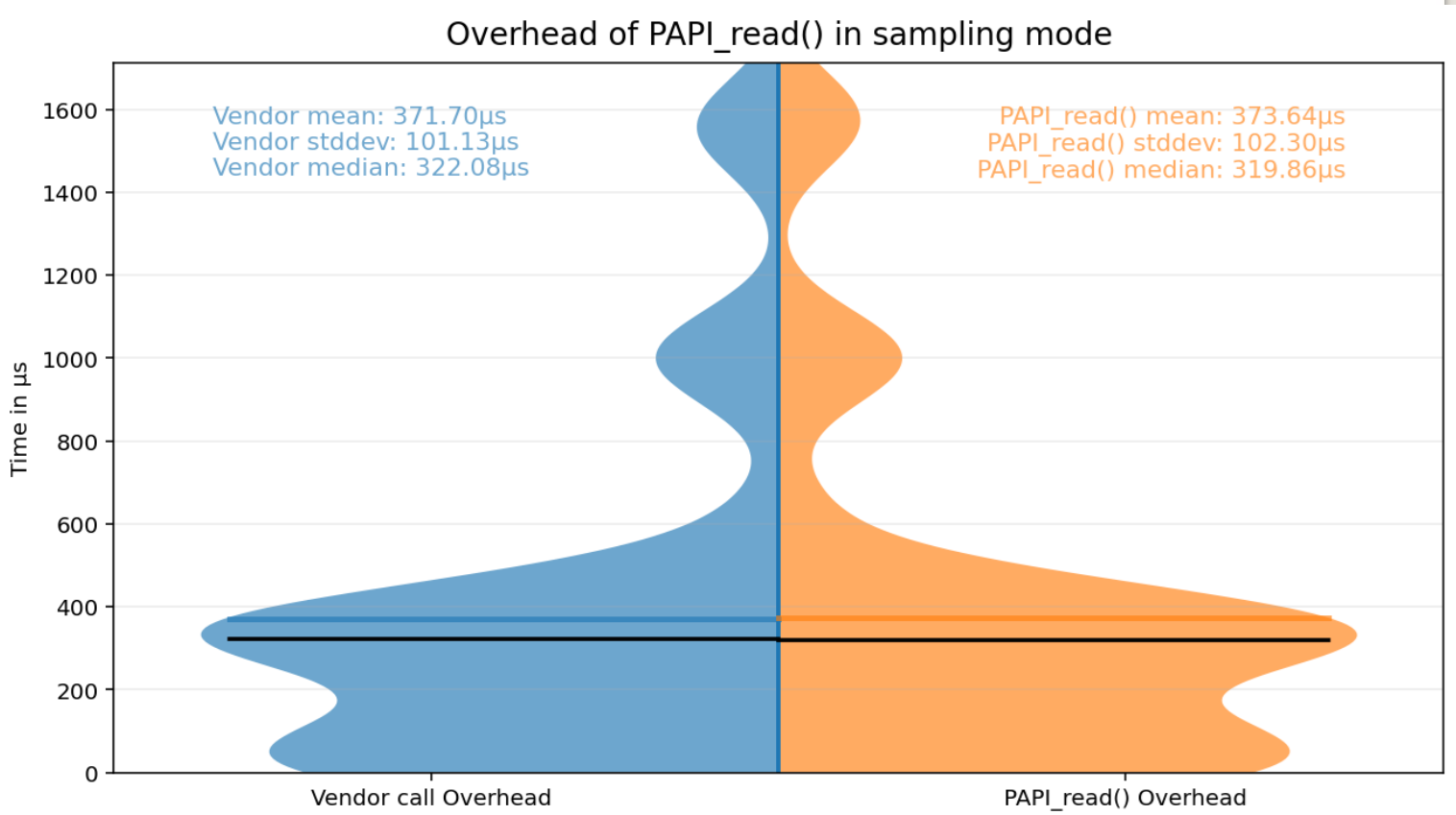
AMD SMI: Low Overhead & Safety

Challenge:

- Adding a layer **could potentially add measurement overhead** and **unsafe library interactions could risk instability due to uninitialized memory or out-of-bounds access.**

Solution:

- **Strictly bounded, zero-initialized accessors** with a minimized **function-pointer read path.**



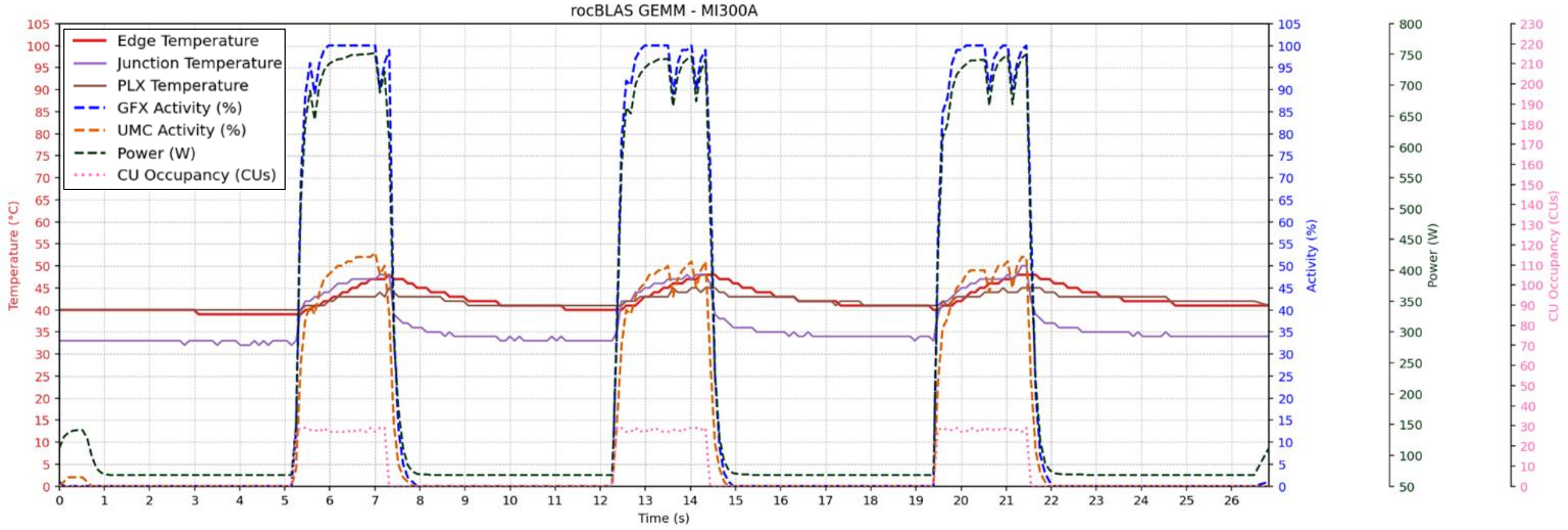
AMD SMI: Metrics Covered

- **Granular Utilization:**
  - Graphics, Memory Controller, and Multimedia Engine
- **Interconnect Health:**
  - XGMI: Topology weights, hops, and link types.
  - PCIe: Physical link health via Replay and NAK counters
- **Process-Level Monitoring:**
  - VRAM usage, engine utilization, and Compute Unit (CU) occupancy to specific Process IDs (PIDs).
- **Reliability, Availability, and Serviceability (RAS):**
  - Provides block-level error counters (Correctable vs. Uncorrectable) for specific components (*e.g.*, System Direct Memory Access vs. GFX) to predict failures.

AMD SMI: Metrics Covered

- **Device & Topology:**
  - Surfaces static hardware limits (cache sizes, PCIe caps), driver and library versions, and topology IDs (NUMA nodes, KFD).
- **Power & Frequency:**
  - Reports instantaneous power, cumulative energy, and granular clock frequencies (SCLK, MCLK, FCLK).
- **Thermal:**
  - Tracks specific thermal sensors (Edge/Junction/VRAM).

Example: rocBLAS GEMM on MI300A



Problem Size: (M=14592, K=131072, N=131072)

PAPI's `gaudi2` Component

- Thank you to Tokey Tahmid for his development of the `gaudi2` component.

Challenges with Specialized AI Chips

Per last year's workshop:

- "Some are less accessible to the broad tools community, *e.g.*, TPU."
- "Some require very different models of measurement and analysis."

"Positioning Tools and the Tools Community to Support AI/ML." Scalable Tools Workshop, 2025.

Challenges with Specialized AI Chips

Intel Gaudi

- **Trace Buffer**
- **Gaudi Profiler APIs**

Source: https://pdsw.org/pdsw25/papers/wips/ws_pdswwip105s2-file1.pdf

Challenges with Specialized AI Chips

Intel Gaudi
- Trace Buffer
- Gaudi Profiler APIs

Cerebras
- Proprietary Metrics
- Software-managed Scheduling

Source: https://pdsw.org/pdsw25/papers/wips/ws_pdswwip105s2-file1.pdf

Challenges with Specialized AI Chips

Intel Gaudi
- Trace Buffer
- Gaudi Profiler APIs

Cerebras
- Proprietary Metrics
- Software-managed Scheduling

SambaNova
- SambaFlow
- Graph-level Profiling

Source: https://pdsw.org/pdsw25/papers/wips/ws_pdswwip105s2-file1.pdf

Challenges with Specialized AI Chips

Intel Gaudi
- Trace Buffer
- Gaudi Profiler APIs

Cerebras
- Proprietary Metrics
- Software-managed Scheduling

SambaNova
- SambaFlow
- Graph-level Profiling

Groq
- GroqView Profiler
- Compile-time Performance Modeling

Source: https://pdsw.org/pdsw25/papers/wips/ws_pdswwip105s2-file1.pdf

Alternative Approach with PAPI SDEs

PAPI Software Defined Events (SDEs)

Software-level metrics:
- I/O BYTEs
- FLOPs

Instrumented HPL-MxP:
- sde_io_read/write_bytes
- sde_float32/sde_float16

Register and track events:
papi_sde_register_counters()

Track counters:
- PAPI_start()
- PAPI_stop()

Source: https://pdsw.org/pdsw25/papers/wips/ws_pdswwip105s2-file1.pdf

Validation Results

Experiments:

- HPL-MxP for FLOPs
- Read/Write I/O example for BYTEs

Architectures:

- ARM Neoverse V2
- Sapphire Rapids (SPR)
- AMD MI300
- Intel Habana Gaudi

New PAPI Component: `gaudi2`

- Merged into PAPI in PR #545.
- Utilizes hlthunk_debug() ioctl interface.
- Tested on HL-225 (Voyager@SDSC: Gaudi2 architecture)
- Exposes the SPMU event group.

* The PAPI `gaudi2` component was developed and tested in a containerized environment on the Voyager system using Kubernetes.

What does the user need to do?

Standard build process while enabling particular components:

```
export PAPI_GAUDI2_ROOT=/path/to/hl-thunk

./configure --prefix=/install/dir --with-components="gaudi2"

make && make install
```

papi_component_avail Utility

```
----------------------------------------------------------------------------

Compiled-in components:
Name:   perf_event          Linux perf_event CPU counters
Name:   gaudi2              Intel Gaudi2 AI Accelerator hardware counters
Name:   sysdetect           System info detection component

Active components:
Name:   perf_event          Linux perf_event CPU counters
                            Native: 166, Preset: 42, Counters: 19
                            PMUs supported: ix86arch, perf, perf_raw, icx

Name:   gaudi2              Intel Gaudi2 AI Accelerator hardware counters
                            Native: 189, Preset: 0, Counters: 6

Name:   sysdetect           System info detection component
                            Native: 0, Preset: 0, Counters: 0

----------------------------------------------------------------------------
```

Challenges with the `gaudi2` Component

- Lack of documentation for performance counters, hl-thunk libraries and debug APIs
- Lack of Gaudi profiler's source code or underlying APIs for events enumeration
- Minimal availability of Gaudi2 hardware

Acknowledgements

This research was supported in part by the U.S. National Science Foundation under award No. 2346423 “CIRC: ENS: Monitoring Infrastructure for Network and Computing Environment Research (MINCER).” This work used Voyager at the San Diego Supercomputer Center through allocation CIS250345 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants No. 2138259, 2138286, 2138307, 2137603, and 2138296.

References

[1] (2025). “Positioning Tools and the Tools Community to Support AI/ML.” In *Scalable Tools Workshop 2025: Working Group Outbrief.* https://dyninst.github.io/scalable_tools_workshop/petascale2025/assets/slides/outbriefs/Sustainability.pdf.

[2] Tahmid, T. and Jagode, H. (2025). “PAPI Support for Specialized AI Architectures.” In *10th International Parallel Data Systems Workshop (PDSW)*.

[3] Barry, D., Danalis, A., and Dongarra, J. (2024). “Automated Data Analysis for Defining Performance Metrics from Raw Hardware Events.” In *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 716-725, https://doi.org/10.1109/IPDPSW63119.2024.00134.

[4] Jagode, H., Danalis, A., Congiu, G., Barry, D., Castaldo, A., and Dongarra, J. (2025). “Advancements of PAPI for the exascale generation.” *The International Journal of High Performance Computing Applications* 39, 2 (2025), 251–268. https://doi.org/10.1177/10943420241303884.

[5] Amaro, R.E., Chen, J.Y., Duarte, J.M., Hutton, T.E., Irving, C., Kandes, M.C., Majumdar, A., Mishin, D.Y., Nguyen, M.H., Rodriguez, P. and Silva, F. (2023). “Voyager–an innovative computational resource for artificial intelligence & machine learning applications in science and engineering.” In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, pp. 278-282. https://dl.acm.org/doi/10.1145/3569951.3597597.

[6] Boerner, T.J., Deems, S., Furlani, T.R., Knuth, S.L., and Towns, J. (2023). “ACCESS: Advancing Innovation: NSF’s Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support.” In *Practice and Experience in Advanced Research Computing (PEARC ’23)*, July 23-27, 2023, Portland, OR, USA. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3569951.3597559.