The Exa-PAPI project is developing new performance counter monitoring capabilities as well as power management support for novel and advanced ECP hardware, and software technologies. Exa-PAPI builds upon classic-PAPI functionality and strengthens its path to exascale with 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.

**ECP SCOPE**

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

- **GOAL 1** Widening its applicability and providing robust support for exascale hardware resources.
- **GOAL 2** Supporting finer-grain measurement and control of power, thus offering software developers a basic building block for dynamic application optimization under power constraints.
- **GOAL 3** Extending PAPI to support Software-Defined Events that originate from the ECP software stack and are treated as black boxes (e.g. communication and math libraries, runtime systems, etc.).
- **GOAL 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.

In summary, the Exa-PAPI team is channeling the monitoring capabilities of hardware counters, power usage, software-defined events into a robust PAPI software package for exascale-level systems.

**PERFORMANCE COUNTER MONITORING CAPABILITIES**

**SUPPORTED ARCHITECTURES**

- **AMD**
  - Power for Intel Xeon, GPU: MI50/60/100, power, temp
  - Power monitoring support (NVML), NVLink
  - KNM, Sky/Kaby/Cascade/Ice Lake
  - Power 5,6,7,8,9

- **arm**
  - Cortex, Aarch64

- **Cray**
  -Center, Cascade, Thunder, SEA, ARM, Power

- **IBM**
  - ELC Power, IBM power, time

- **Power9**
  - NEST event support via

- **SUN**
  - Tools, V86, x86, Power, IBM

- **NVidia**
  - Virtual Environment, Aries

- **VMWARE**
  - Virtual Environment

**SUPPORT FOR GPUs: INTEL (AURORA EARLY ACCESS)**

Added PAPI capabilities for monitoring Intel GPUs:

- GPUs hardware events, and
- Memory performance metrics (bytes read / written / transferred from / to L3).
- Monitoring of GPU and memory performance counters aids developers in producing more efficient code by profiling the utilization of the latest GPU resources and diagnosing performance bottlenecks.

Scope and Objectives:

- A mechanism to collect Intel GPU performance metrics via PAPI's well known API

Approach:

- Enable through PAPI component framework
- Access Intel GPU performance metrics via Intel One API Level-Zero Interface
- Tricky part: Two different collection modes via one PAPI component

**ECP PROJECTS AND 3RD PARTY TOOLS APPLYING PAPI**

**SOFTWARE-DEFINED EVENTS IN PAPI**

**KEY POINTS ABOUT SDEs**

- New measurement possibilities:
  - Tasks stolen, matrix residuals, partial results reached, arguments passed to functions
- Any tool can read PAPI SDEs:
  - SDEs from a library can be read with PAPI_start()/PAPI_stop()/PAPI_read().
- Low overhead:
  - Performance critical codes can implement SDEs with zero overhead.
- Easy to use, with rich feature set:
  - Pull-mode & push-mode, read-write counters, sampling/overflowing, counters, groups, recordings, statistics, thread safety, custom callbacks

**SUPPORT FOR GPUs: AMD (FRONTIER EARLY ACCESS)**

Extended PAPI capabilities for monitoring AMD GPUs:

- Previous AMD GPU monitoring was limited to “Sampling mode” only.
- Latest PAPI room component supports both “Sampling” and “Intercept Mode”.
- Features and approach are equivalent to PAPI’s Support for Intel GPUs (see details on the left)

**Example:**

```cpp
/* launch work on GPU */
hipSetDevice(omp_get_thread_num());

#pragma omp parallel num_threads(3)
PAPI_thread_init
PAPI_library_init
setenv("ROCP_HSA_INTERCEPT", "0", 0);

PAPI_start(&eventset);

// launch work on GPU /*

PAPI_stop(eventset);

// wait for work on GPU

PAPI_add_event(eventset, PAPI_L2_DATA_RD, PAPI_L2_DATA_WR, PAPI_L2_DATA_TRANS);

PAPI_create_eventset(&eventset);

接入()

// read the counters

PAPI_read(eventset);

// end the context

PAPI_thread_exit;
```

**SOFTWARE AVAILABLE AT**

http://icl.utk.edu/exa-papi/