EXA-PAPI

Development of 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, I/O system, and energy/power management. PAPI++ is building upon classic-PAPI functionality and strengthens its path to exascale with a more efficient and flexible software design, one that takes advantage of C++'s object-oriented nature but preserves the low-overhead monitoring of performance counters and adds a vast testing suite.

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.

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.

ECP PROJECTS AND 3RD PARTY TOOLS APPLYING PAPI

ECP DTE (PaRSEC) UTK **http://icl.utk.edu/parsec/**	**ECP LLNL-ATDM (Caliper)** LLVM github.com/LLNL/caliper-compiler	**ECP SNL-ATDM (Kokkos)** SNL **https://github.com/kokkos**	**ECP Proteas (TAU)** University of Oregon **http://tau.uoregon.edu/**
ECP HPCToolkit (HPCToolkit) Rice University **http://hpctoolkit.org**	**Score-P** **http://score-p.org**	**Vampir** TU Dresden **http://www.vampir.eu/**	**Scalasca** FZ Juelich, TU Darmstadt **http://scalasca.org/**
PerfSuite NCSA **http://perfsuite.ncsa.uiuc.edu/**	**Open|Speedshop** Open|SpeedShop **https://openspeedshop.org/**	**SvPablo** RENCI at UNC **www.renci.org/research/pablo**	**ompP** LMU Munich **http://www.ompp-tool.com/**

PERFORMANCE COUNTER MONITORING CAPABILITIES

SUPPORTED ARCHITECTURES

AMD **CPU: up to Fam17 Zeppelin Zen GPU: ROCm, ROCm-smi**	arm **Cortex, Cavium ThunderX, ARM64**	CRAY THE SUPERCOMPUTER COMPANY **Gemini and Aries interconnect, power**	IBM **Blue Gene Series, Q: 5-D Torus, I/O System, EMON power, energy**	IBM **Power 5,6,7,8,9 Power monitoring support**
IBM **Power9 NEST event support via Performance Co-Pilot (PCP) PAPI component**	intel **Westmore, Sandy/Ivy Bridge, Haswell, Broadwell, Skylake(-X), Kaby Lake, Cascadelake**	intel **KNC, KNL, Knights Mill including power/energy**	intel **RAPL (power/energy), powercap**	InfiniBand TRADE ASSOCIATION
lustre **Tesla, Kepler, Maxwell, Pascal, Volta**	NVIDIA **Tesla, Kepler, Maxwell, Pascal, Volta**	NVIDIA **Power monitoring and capping support (NVML), NVLink**	KVM **Virtual Environment**	vmware **Virtual Environment**

SUPPORT FOR GPUs: AMD and NVIDIA

Activity 1: Performance counter monitoring

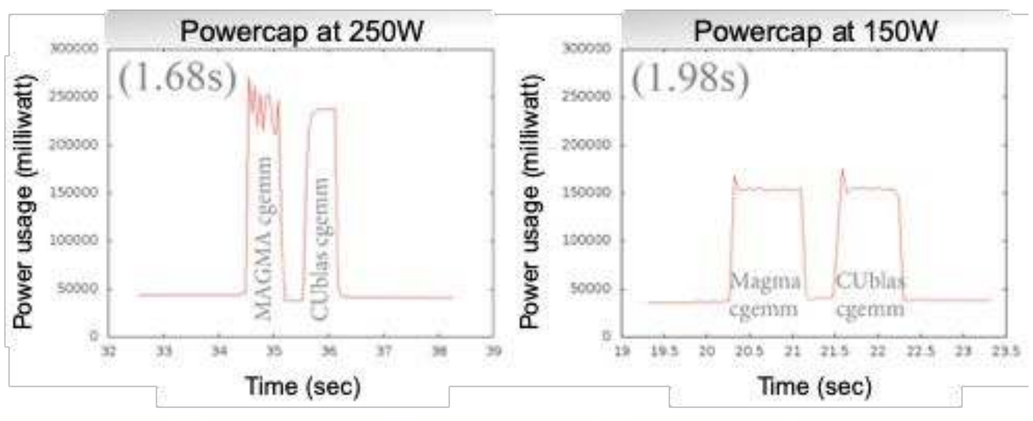
- Develop support for **NVIDIA** monitoring capabilities for GPUs on Summit.
  - Added PAPI capabilities for monitoring TESLA V100 + NVLINK
- Develop support for **AMD GPUs** Monitoring Capabilities:
  - Development of a new PAPI ROCm component

Activity 2: Power monitoring and capping support

- Develop support for **NVIDIA power management capabilities** for GPUs on Summit.
  - Added PAPI capabilities for monitoring TESLA V100:
    - Power consumption and power capping support
    - Fan speed, temperature
- Develop support for **AMD GPUs power monitoring**:
  - Development of a new PAPI ROCm-smi component

POWER AWARENESS EXAMPLE

- Power Reading and Capping with PAPI on TESLA V100 GPUs
- Enables developers to change run profiles to reduce energy cost



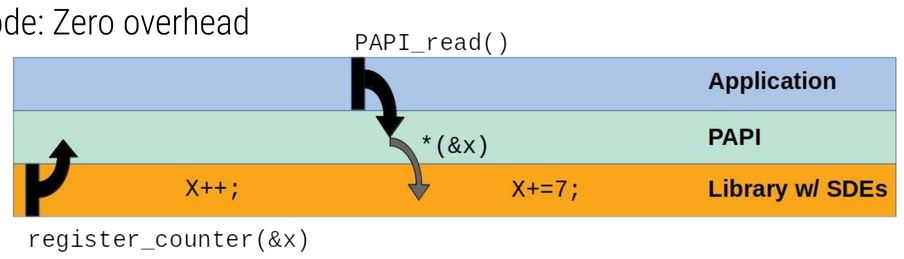
SOFTWARE-DEFINED EVENTS IN PAPI

KEY POINTS ABOUT SDEs

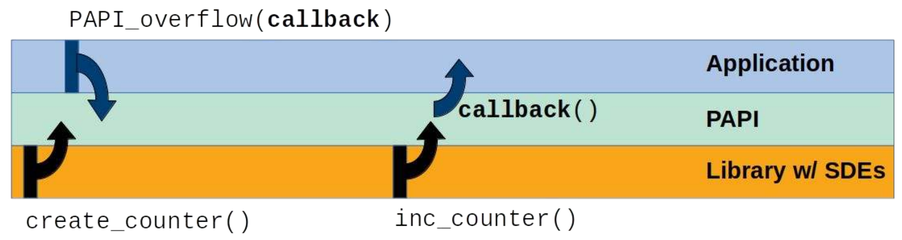
- New measurement possibilities:
  Tasks stolen, matrix residuals, partial results reached, arguments passed to functions
- Any tool can read PAPI SDEs:
  SDEs from a library can be read with PAPI_start()/PAPI_stop()/PAPI_read().
- Low overhead:
  Performance critical codes can implement SDEs with zero 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

DIFFERENT ACCESS MODELS

Pull mode: Zero overhead



Push mode: Determinism and precision



PAPI'S NEW SDE API

- API for reading SDEs remains the same as the API for reading hardware events, i.e. PAPI_start(), etc.
- SDE API calls are only meant to be used inside libraries to export SDEs from within those libraries.
- All API functions are available in C and FORTRAN'08.

```
void   *papi_sde_init(char *lib_name, int event_count);

void   papi_sde_register_counter(void *handle, char *event_name, int type,
       int mode, void *counter);

void   papi_sde_describe_counter(void *handle, char *event_name, char
*event_description);
```


ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY OF TENNESSEE KNOXVILLE
U.S. DEPARTMENT OF ENERGY | Office of Science
SPONSORED BY ECP EXASCALE COMPUTING PROJECT
NNSA National Nuclear Security Administration

SOFTWARE AVAILABLE AT
http://icl.utk.edu/exa-papi/