# Performance API (PAPI)

14th Scalable Tools Workshop

Anthony Danalis, Heike Jagode, Giuseppe Congiu, Jack Dongarra

Tahoe City, CA

June 19-23, 2022

INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

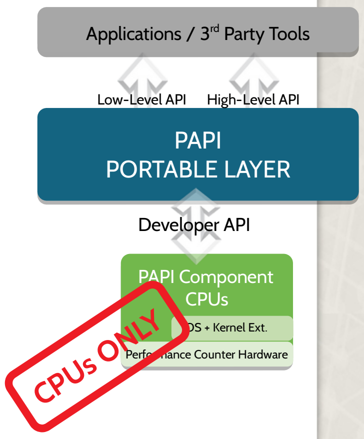
# PAPI

- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i.e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O, FS, Energy/Power.

- PAPI enables SW engineers to see, in near real time, the relation between **SW performance** and **HW events across the entire compute system.**

# PAPI

1999 - 2009



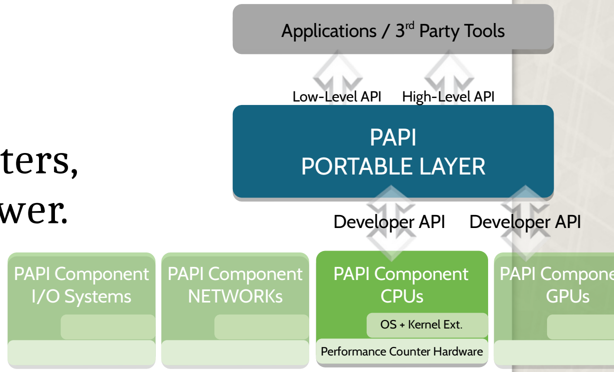

- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i.e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O, FS, Energy/Power.
- PAPI enables SW engineers to see, in near real time, the relation between **SW performance** and **HW events across the entire compute system.**

**SUPPORTED ARCHITECTURES:**

- AMD **up to Zen3**
- ARM Cortex A8, A9, A15, ARM64, **ARM uncore-support**
- 
- IBM Blue Gene Series
- IBM Power Series, **PCP for POWER9-nest**
- Intel Sandy|Ivy Bridge, Haswell, Broadwell, Skylake, **Kaby-**, **Cascade-, Ice-lake**, KNC, KNL, KNM

# PAPI

2009 - 2018




- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i.e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O, FS, Energy/Power.
- PAPI enables SW engineers to see, in near real time, the relation between **SW performance** and **HW events across the entire compute system.**

**SUPPORTED ARCHITECTURES:**

- AMD **up to Zen3**
- AMD **GPUs MI50, MI60, MI100**
- ARM Cortex A8, A9, A15, ARM64, **ARM uncore-support**
- CRAY: Gemini and Aries interconnects, power/energy
- IBM Blue Gene Series, Q: 5D-Torus, I/O system
- IBM Power Series, **PCP for POWER9-nest**
- Intel Sandy|Ivy Bridge, Haswell, Broadwell, Skylake, **Kaby-**, **Cascade-, Ice-lake**, KNC, KNL, KNM

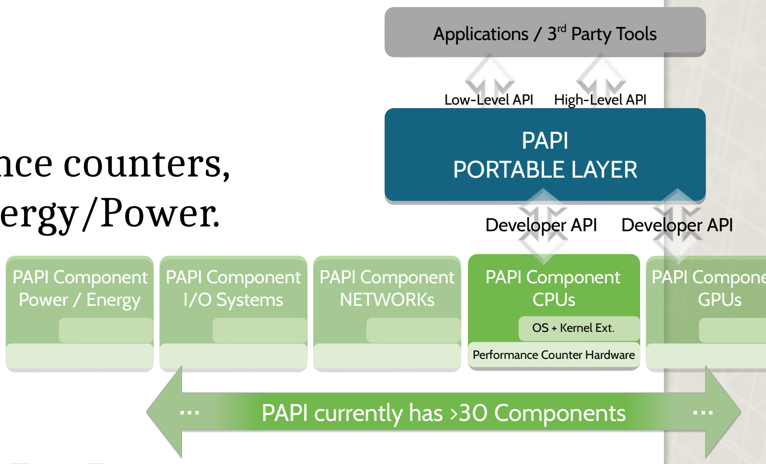
- **Intel GPUs**
- InfiniBand
- Lustre FS
- NVIDIA Tesla, Kepler, Maxwell, Pascal, Volta, **Turing, Ampere**: support for multiple GPUs
- **NVIDIA: support for NVLink**

# PAPI

- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i.e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O, FS, Energy/Power.

- PAPI enables SW engineers to see, in near real time, the relation between **SW performance** and **HW events across the entire compute system.**

2009 - 2018

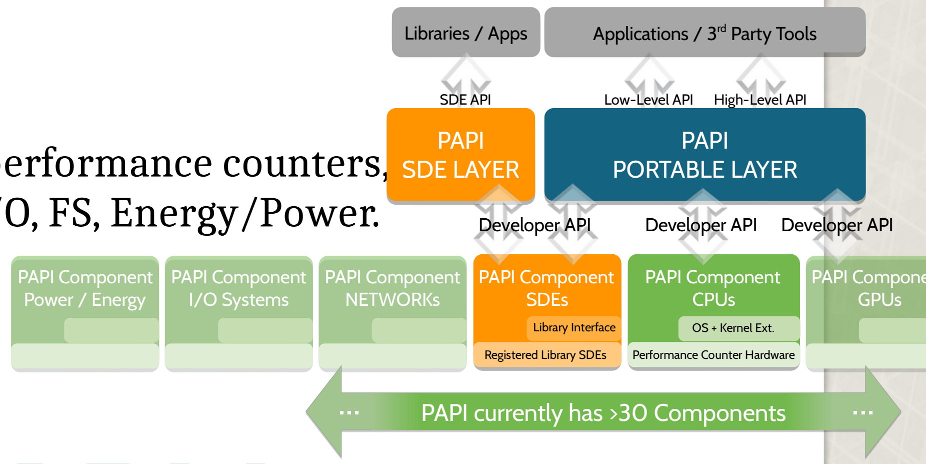



**SUPPORTED ARCHITECTURES:**

- AMD **up to Zen3**, **power for Fam17h**
- AMD **GPUs MI50, MI60, MI100**, **power, temperature, fan**
- ARM Cortex A8, A9, A15, ARM64, **ARM uncore-support**
- CRAY: Gemini and Aries interconnects, power/energy
- IBM Blue Gene Series, Q: 5D-Torus, I/O system, EMON power/energy
- IBM Power Series, **PCP for POWER9-nest, power monitoring & capping on POWER9**
- Intel Sandy|Ivy Bridge, Haswell, Broadwell, Skylake, **Kaby-**, **Cascade-, Ice-lake**, KNC, KNL, KNM
- Intel RAPL (power/energy), **power capping**
- **Intel GPUs**
- InfiniBand
- Lustre FS
- NVIDIA Tesla, Kepler, Maxwell, Pascal, Volta, **Turing, Ampere**: support for multiple GPUs
- **NVIDIA: support for NVLink**
- NVIDIA NVML (power/energy); **power capping**
- Virtual Environments: VMware, KVM

# PAPI

2018 - now




- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i.e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O, FS, Energy/Power.

- PAPI enables SW engineers to see, in near real time, the relation between **SW performance** and ~~**HW**~~ **events across the entire compute system.**

**SUPPORTED ARCHITECTURES:**

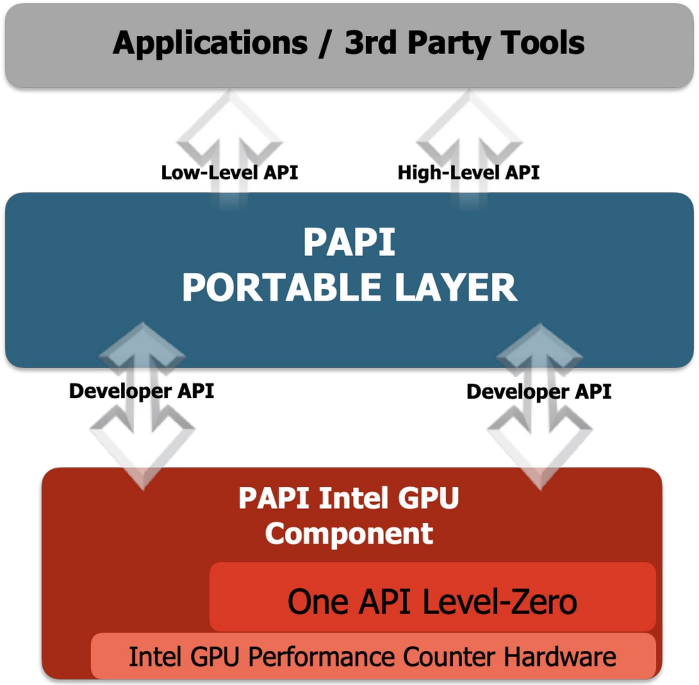
- AMD **up to Zen3**, **power for Fam17h**
- AMD **GPUs MI50, MI60, MI100**, **power, temperature, fan**
- ARM Cortex A8, A9, A15, ARM64, **ARM uncore-support**
- CRAY: Gemini and Aries interconnects, power/energy
- IBM Blue Gene Series, Q: 5D-Torus, I/O system, EMON power/energy
- IBM Power Series, **PCP for POWER9-nest, power monitoring & capping on POWER9**
- Intel Sandy|Ivy Bridge, Haswell, Broadwell, Skylake, **Kaby-**, **Cascade-, Ice-lake**, KNC, KNL, KNM
- Intel RAPL (power/energy), **power capping**
- **Intel GPUs**
- InfiniBand
- Lustre FS
- NVIDIA Tesla, Kepler, Maxwell, Pascal, Volta, **Turing, Ampere**: support for multiple GPUs
- **NVIDIA: support for NVLink**
- NVIDIA NVML (power/energy); **power capping**
- Virtual Environments: VMware, KVM
- **Software-defined Event (SDE) Support**

# Intel GPU Support

# Intel GPUs

Support for monitoring Intel GPUs on Aurora Early Access (Iris & Florentia).

- GPU hardware events
- Memory performance metrics (bytes read/written/transferred from/to LLC)

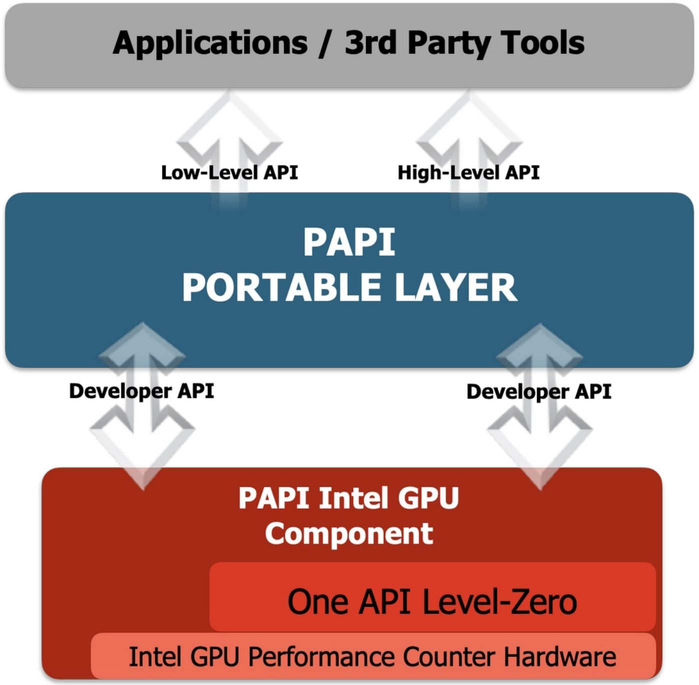



Thanks to Peinan Zhang & Rashawn Knapp

# Intel GPUs

Support for monitoring Intel GPUs on Aurora Early Access (Iris & Florentia).

- GPU hardware events
- Memory performance metrics (bytes read/written/transferred from/to LLC)

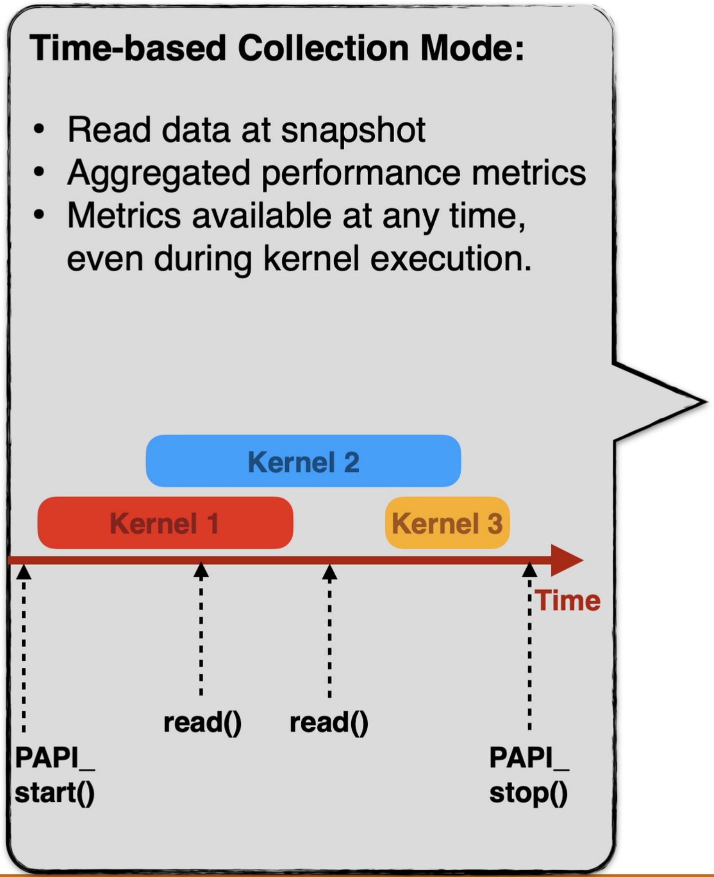


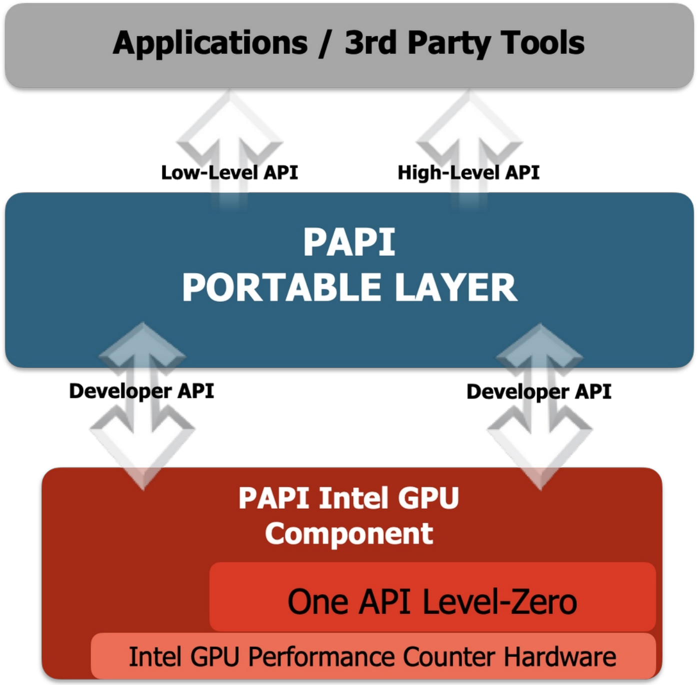

Two different collection modes supported by PAPI component

# Intel GPUs

Support for monitoring Intel GPUs on Aurora Early Access (Iris & Florentia).

- GPU hardware events
- Memory performance metrics (bytes read/written/transferred from/to LLC)

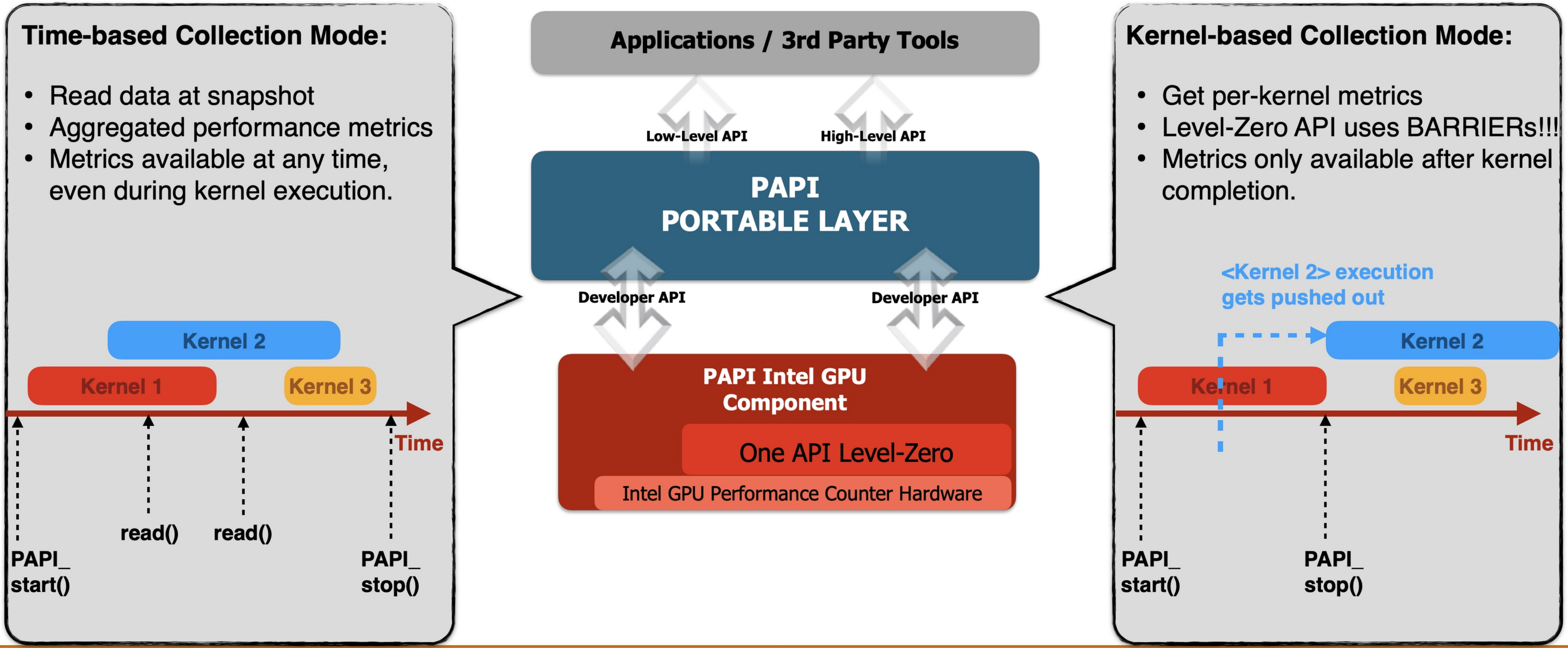


# Intel GPUs

Support for monitoring Intel GPUs on Aurora Early Access (Iris & Florentia).

- GPU hardware events
- Memory performance metrics (bytes read/written/transferred from/to LLC)



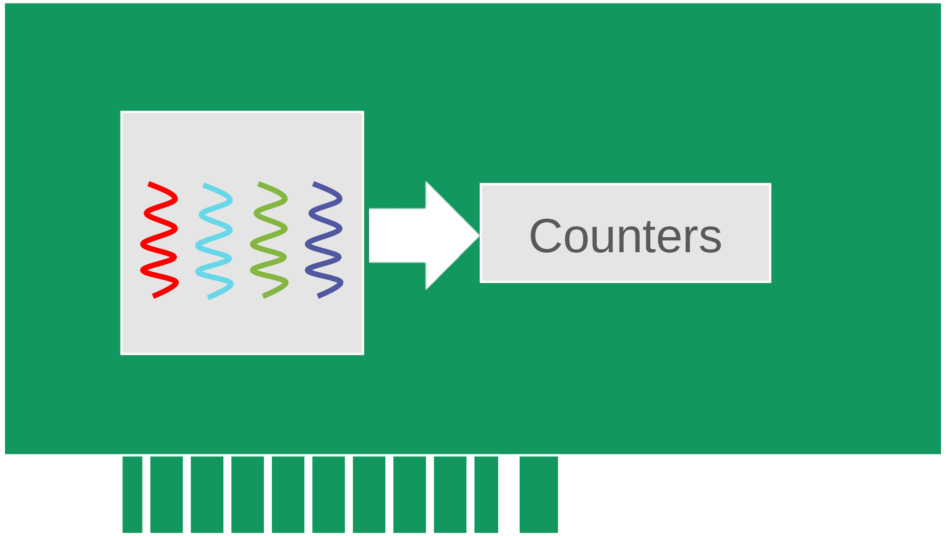
# AMD GPU Support

INNOVATIVE COMPUTING LABORATORY

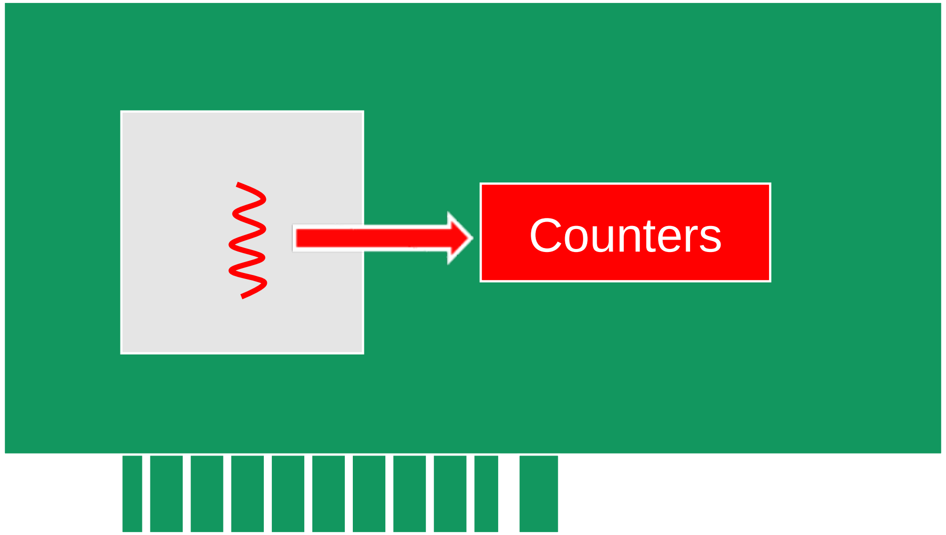
THE UNIVERSITY OF TENNESSEE KNOXVILLE

# ROC Profiler Counter Semantics (Profiling Modes)

- ROC Profiler supports two profiling modes: sampling and intercept
- Sampling: GPU-wide hardware performance counter monitoring
- Intercept: per-kernel hardware performance counter monitoring



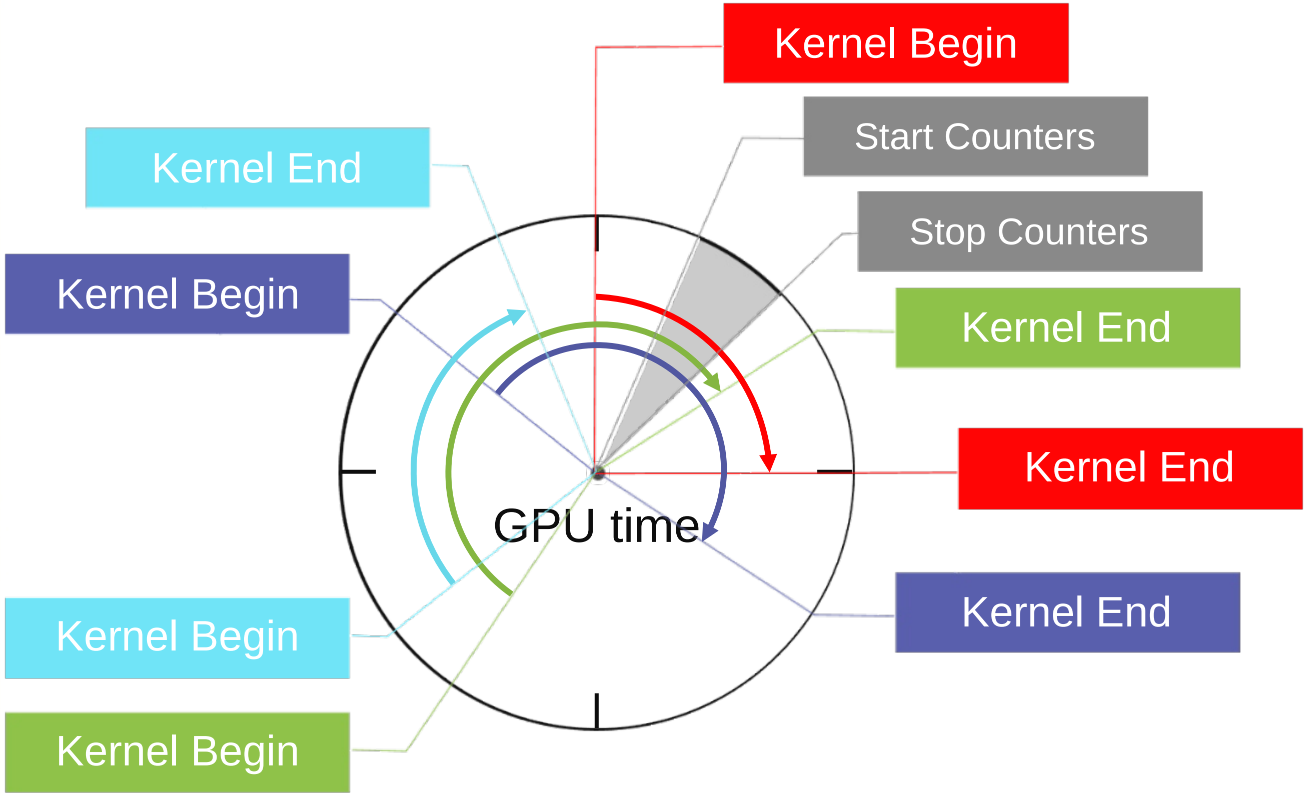

ROC Profiler **Sampling** Mode



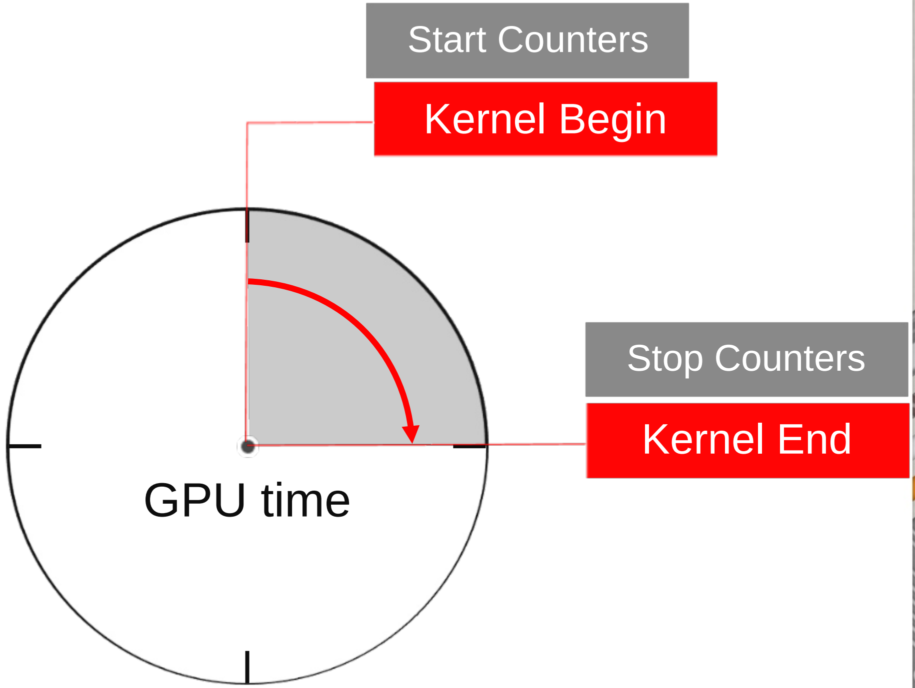

ROC Profiler **Intercept** Mode**:**
Kernels are **serialized** by the GPU runtime

# ROC Profiler Counter Semantics (Granularity)

ROC Profiler **Sampling** Mode

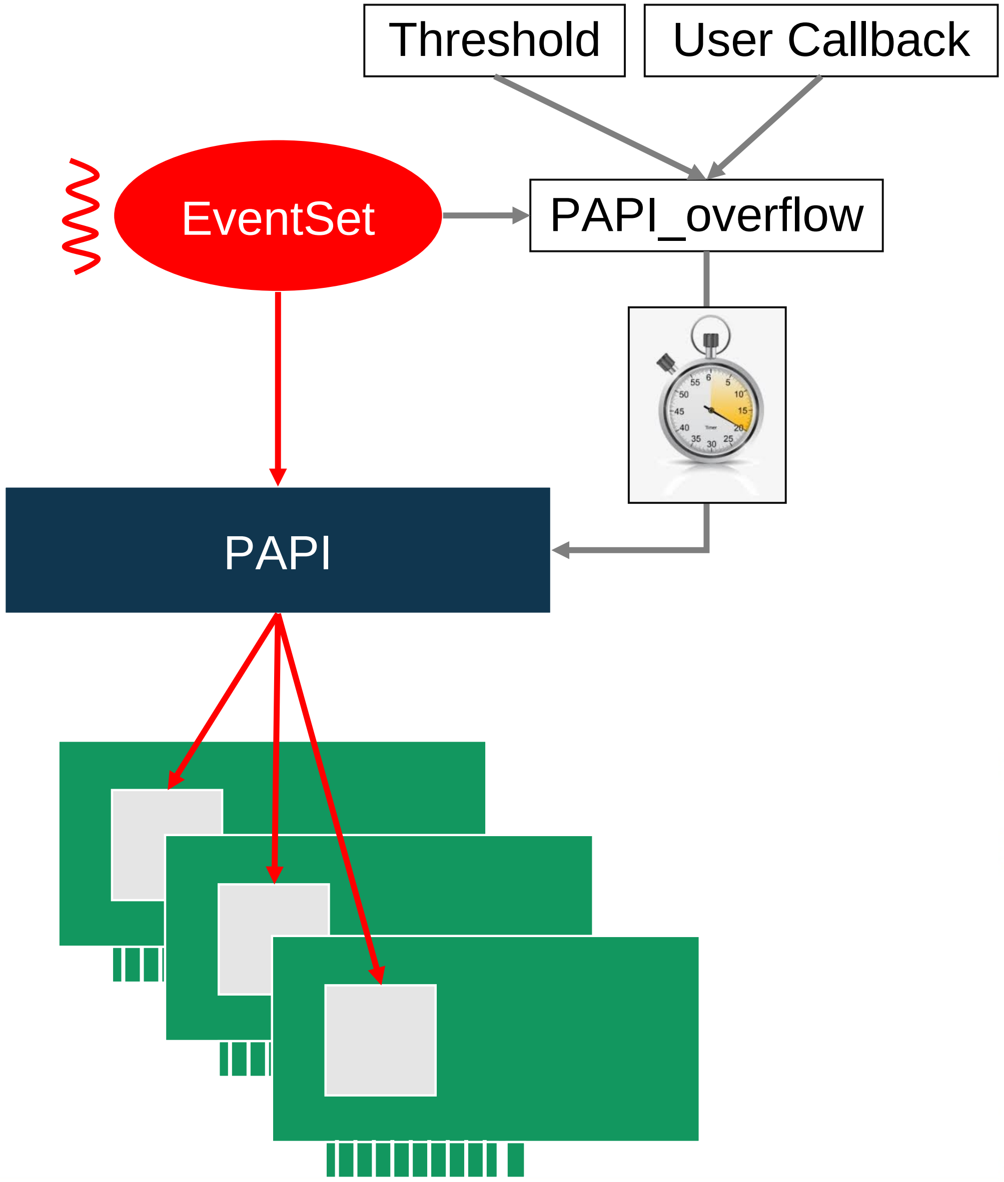



ROC Profiler **Intercept** Mode:
Kernels are **serialized** by the GPU runtime



# Counter Sampling

- The PAPI ROCm component also supports counter sampling
- Tools can register a callback, which gets invoked when a counter overflows, using **PAPI_overflow**
- ROC Profiler does not support counter overflow in hardware, thus PAPI emulates overflow using timers
- Only makes sense when ROC Profiler is configured in **ROC profiler** “sampling mode”

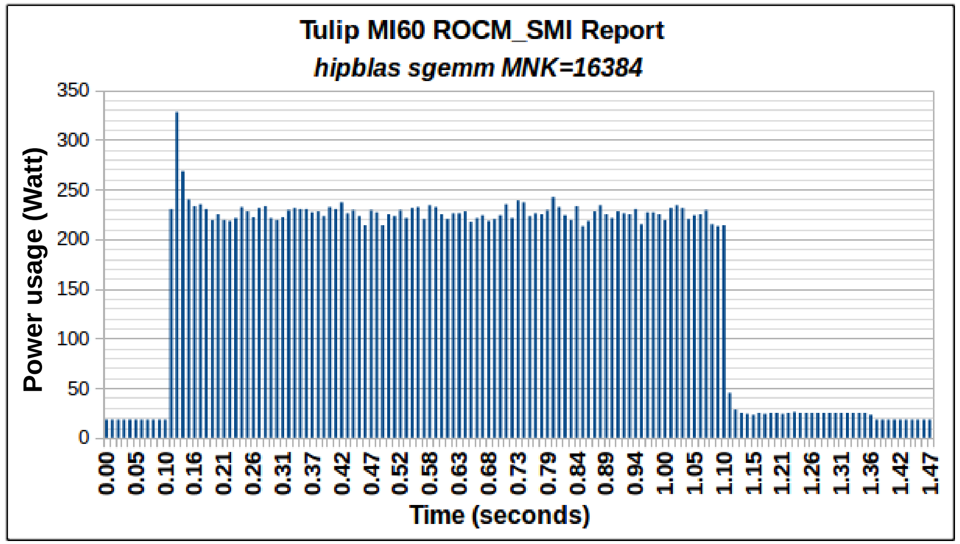


# AMD GPU power monitoring & capping

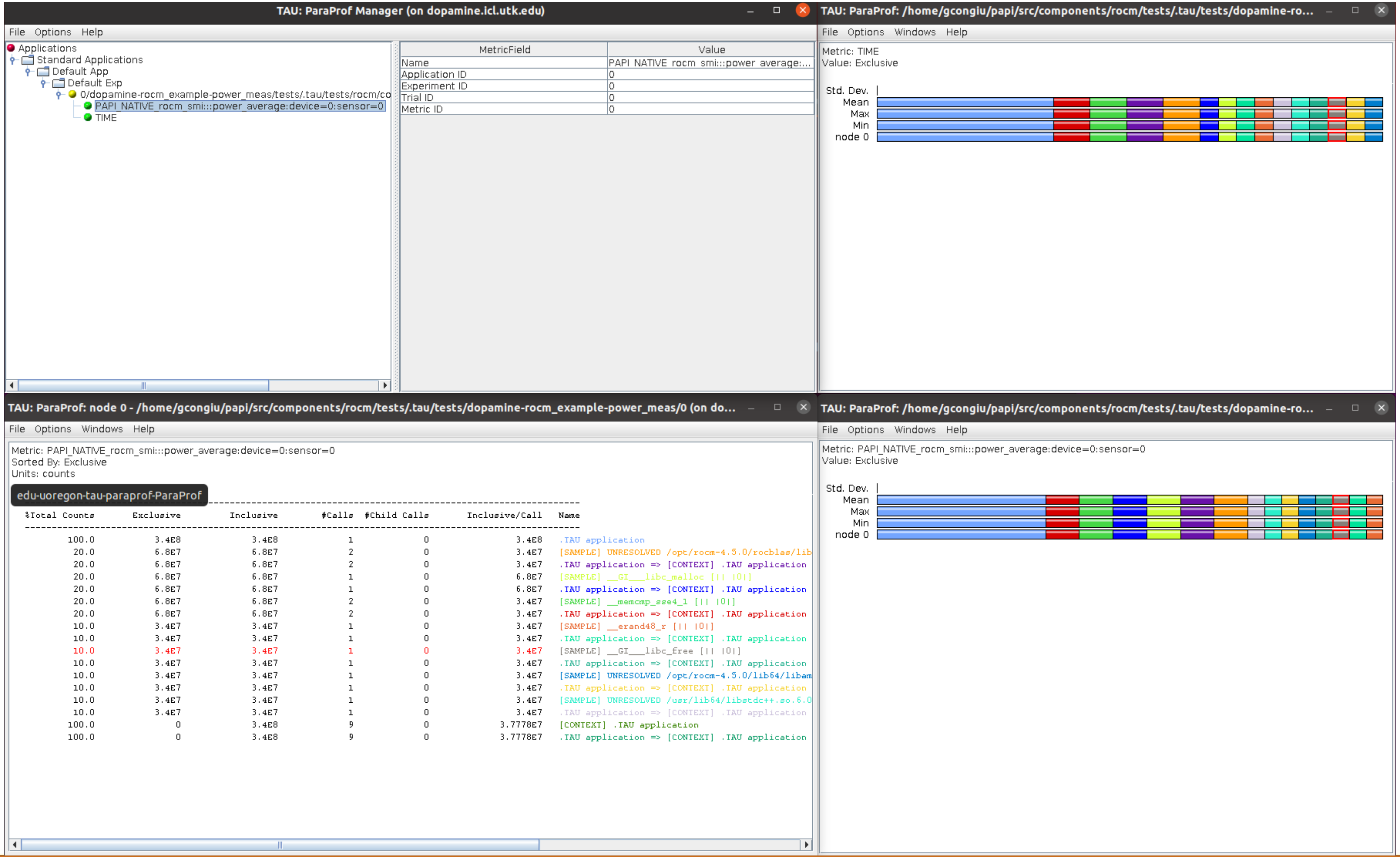
Support for **AMD GPUs power manipulation** for GPUs on **Frontier** EAS

PAPI **ROCm-smi component** enables developers to change run profiles to reduce energy cost

- Power: monitoring and power capping.
- Temperature: current temp., max critical value, temporary emergency temperature.
- Fan: fan speed in RPM, max speed, read / write speed.
- Memory: Total VRAM, Visible VRAM, GTT usage of VRAM, usage of VIS VRAM.
- PCI: Throughput sent, received, max packet size.
- Busy percent: % of time device is busy doing any processing.



# AMD power using PAPI through TAU



# Software Defined Events (SDE)

INNOVATIVE
COMPUTING LABORATORY

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

# PAPI Software Defined Events (SDEs)

Support for Events that originate in Software Layers

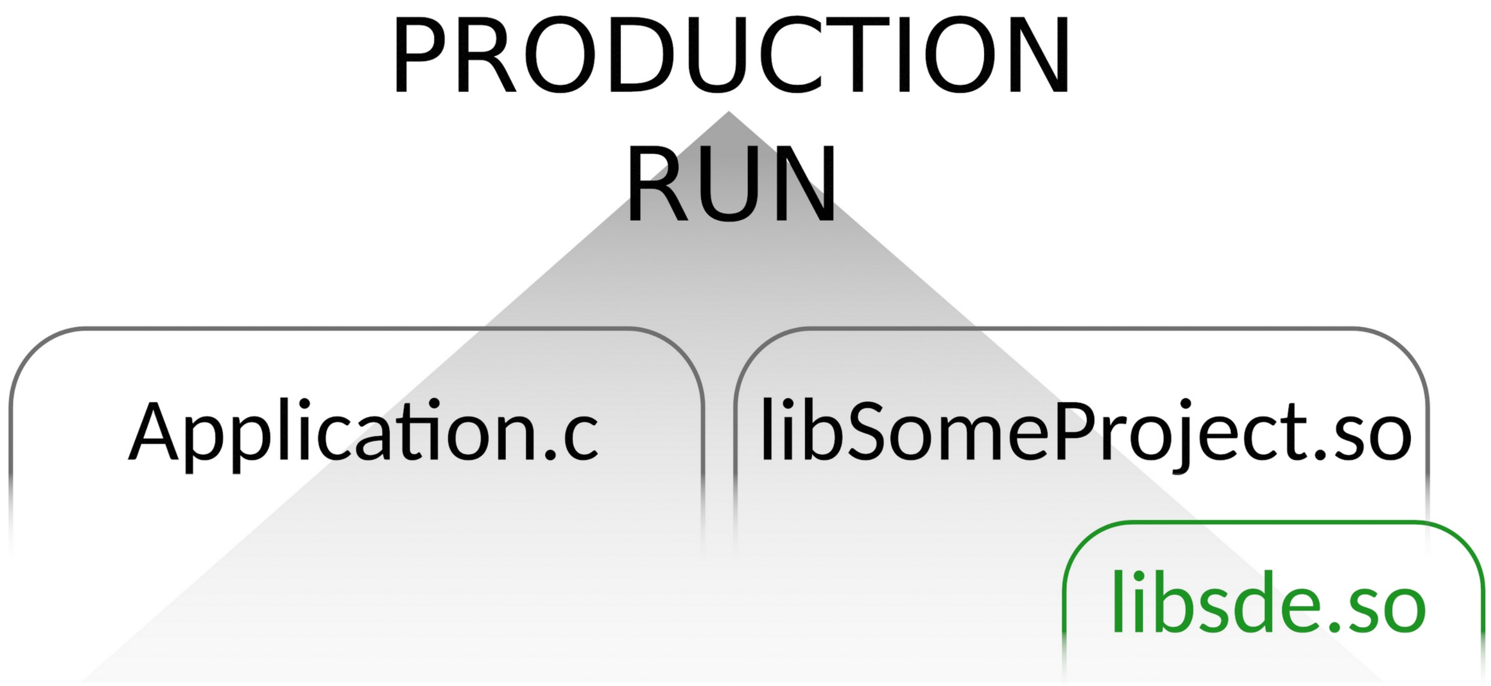
SDEs enable **software** layers to export **arbitrary information** as if it came from hardware counters

Arguments passed to functions, residuals, tasks stolen, hash-table collisions, messages sent, memory consumption, size of internal data structures, ...

# PAPI Software Defined Events (SDEs)

Support for Events that originate in Software Layers

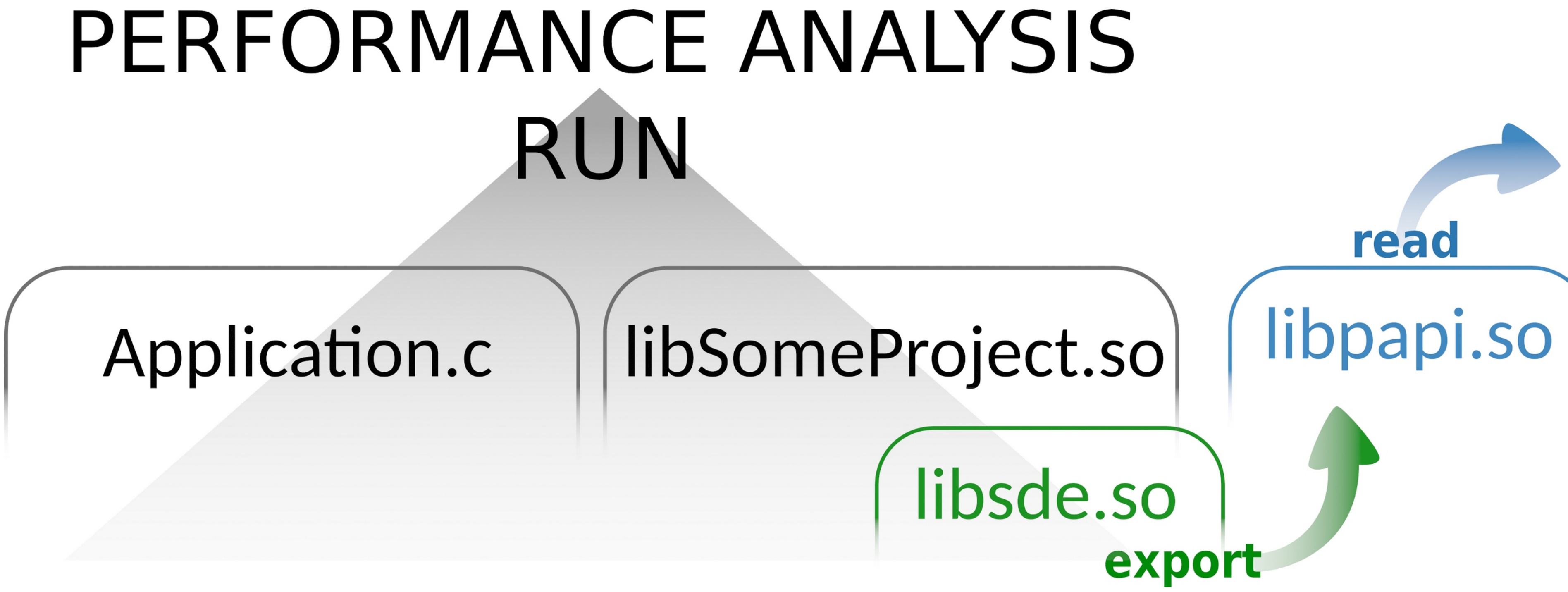
SDEs enable **software** layers to export **arbitrary information** as if it came from hardware counters



# PAPI Software Defined Events (SDEs)

Support for Events that originate in Software Layers

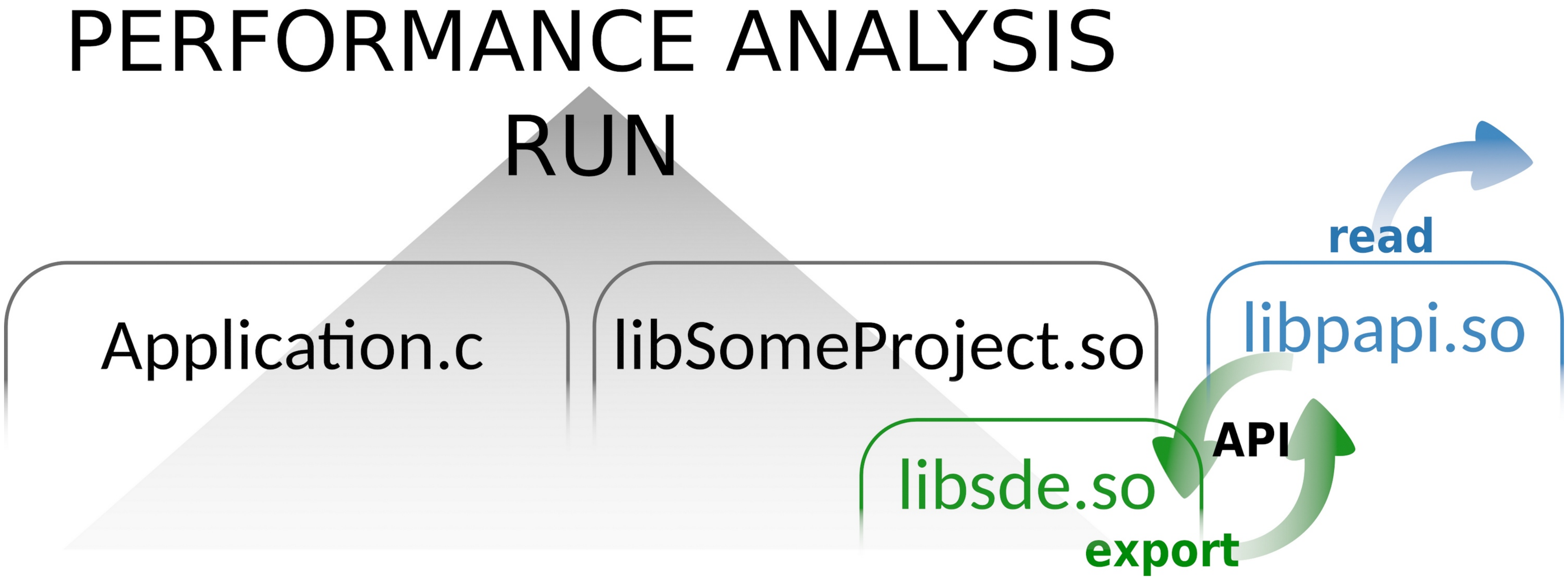
SDEs enable **software** layers to export **arbitrary information** as if it came from hardware counters



# PAPI Software Defined Events (SDEs)

Support for Events that originate in Software Layers

SDEs enable **software** layers to export **arbitrary information** as if it came from hardware counters



# Counter Analysis Toolkit (CAT)

INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

# Key Concepts

- Goal:

  Create a set of micro-benchmarks for illustrating details in hardware events and how they relate to the behavior of the micro-architecture

- Target audience:
  - Performance conscious application developers
  - PAPI developers working on new architectures (think preset events)
  - Developers interested in validating hardware event counters

# CAT kernel example for Branch Events

50% Taken
0% Mispredicted

```
do{
    if ( iter_count < (size/2) ){
        global_var2 += 2;
    }
    BRNG();
    iter_count++;
}while(iter_count<size);
```

100% Taken
0% Mispredicted

50% Taken
50% Mispredicted

```
do{
    iter_count++;
    BUSY_WORK();
    BRNG();
    if ( (result % 2) == 0 ){
        global_var1+=2;
    }
}while(iter_count<size);
```

# CAT kernel example for Branch Events

```
do{
    if ( iter_count < (size/2) ){
        global_var2 += 2;
    }
    BRNG();
    iter_count++;
}while(iter_count<size);
```

1.5 Branches

```
do{
    iter_count++;
    BUSY_WORK();
    BRNG();
    if ( (result % 2) == 0 ){
        if( (global_var1 % 2) != 0 ){
            global_var2++;
        }
        global_var1+=2;
        BUSY_WORK();
    }
}while(iter_count<size);
```

```
do{
    iter_count++;
    BUSY_WORK();
    BRNG();
    if ( (result % 2) == 0 ){
        global_var1+=2;
    }
}while(iter_count<size);
```

100% Direct

```
 do{
    BRNG();
    global_var2+=2;
    if ( iter_count < global_var2 ){
        global_var1+=2;
        goto zzz;
    }
    BRNG();
zzz: iter_count++;
    BRNG();
 }while(iter_count<size);
```

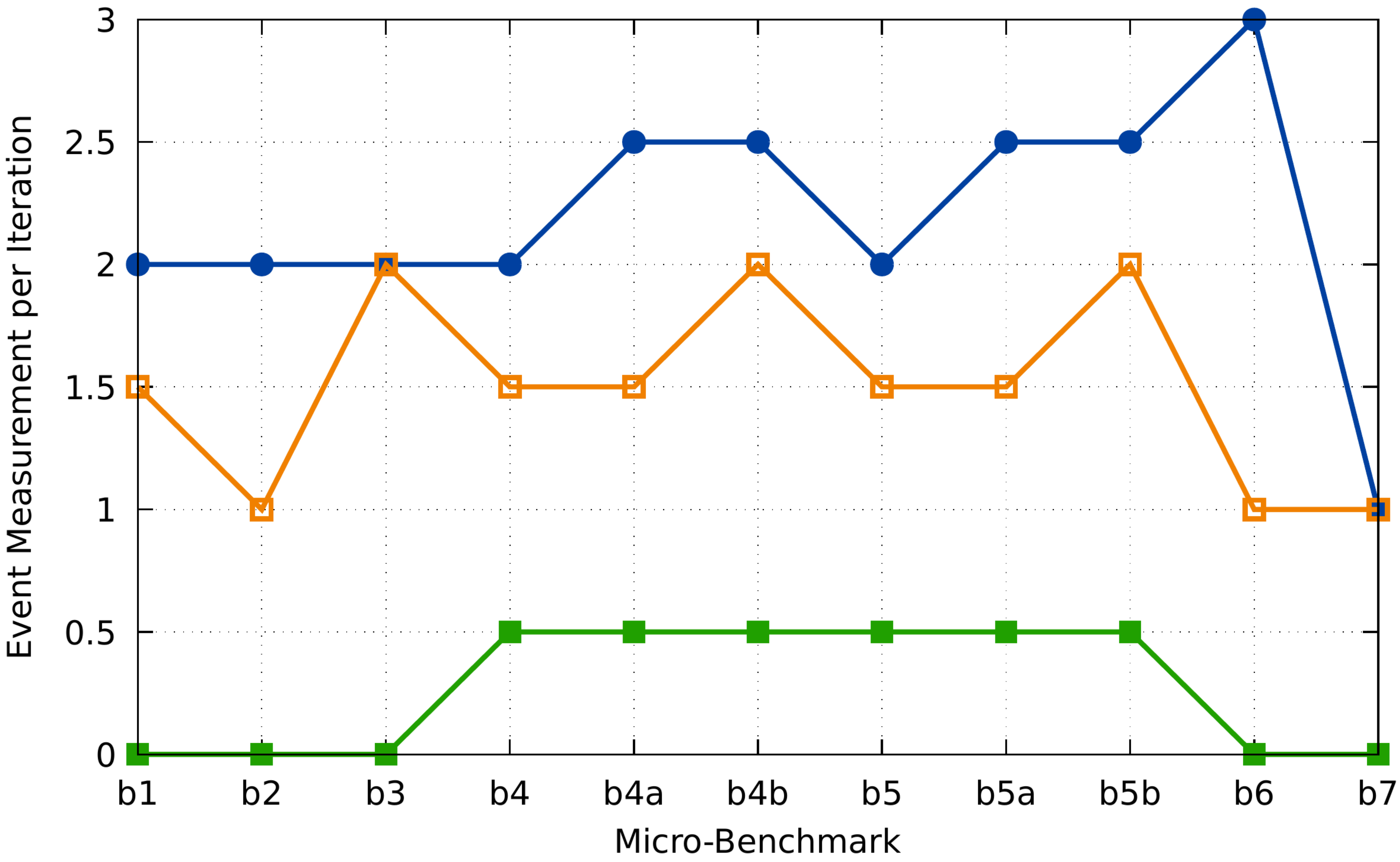
# Expected Behavior Table

| | b1 | b2 | b3 | b4 | b4a | b4b | b5 | b5a | b5b | b6 | b7 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ALL BR | 2 | 2 | 2 | 2 | 2.5 | 2.5 | 2 | 2.5 | 2.5 | 3 | 1 |

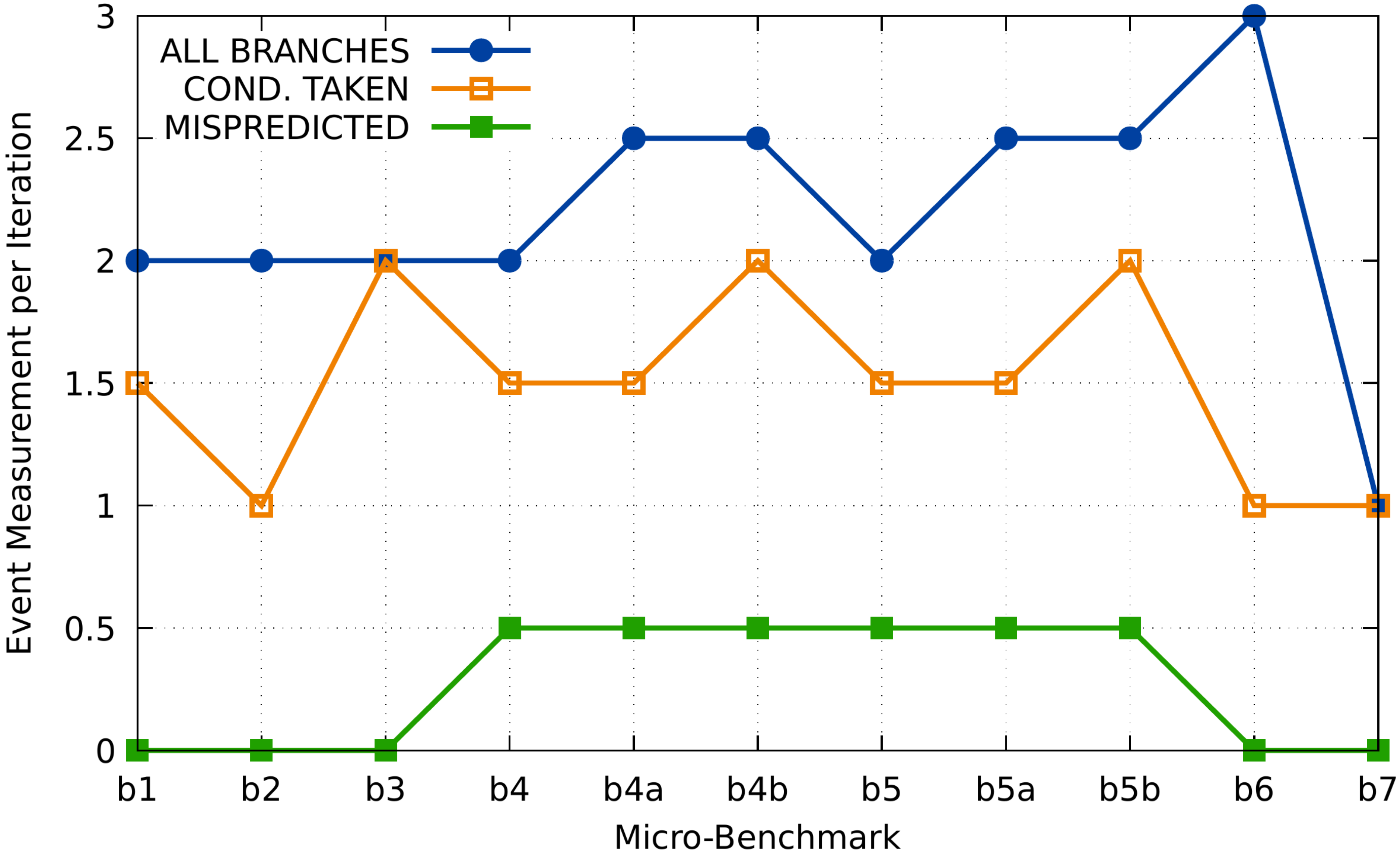
# Expected Behavior Table

| | b1 | b2 | b3 | b4 | b4a | b4b | b5 | b5a | b5b | b6 | b7 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ALL BR | 2 | 2 | 2 | 2 | 2.5 | 2.5 | 2 | 2.5 | 2.5 | 3 | 1 |
| MISP | 0 | 0 | 0 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0 | 0 |

# Native Branch Events Have Unique Responses

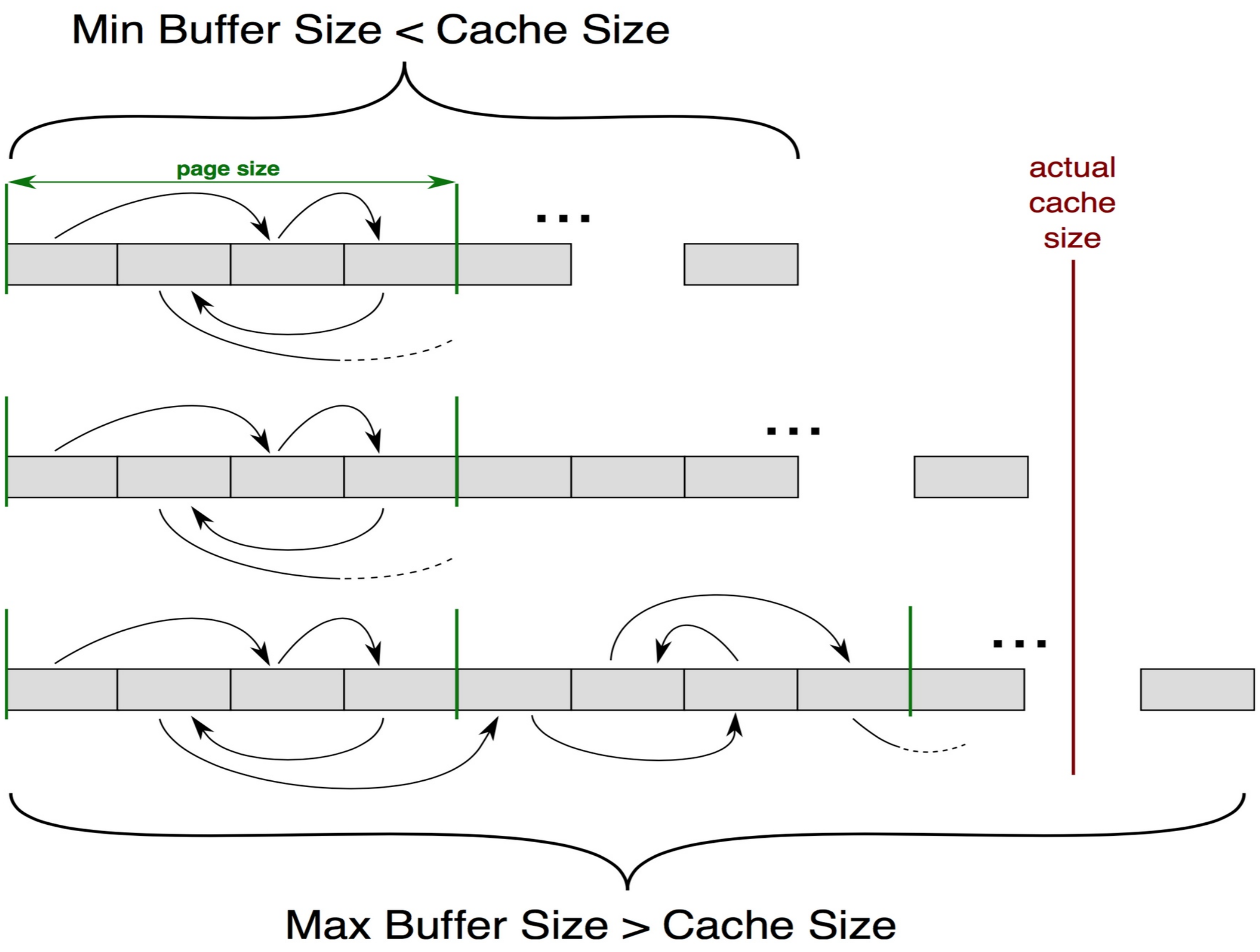


# Unique Responses Reveal Mapping to Preset Events



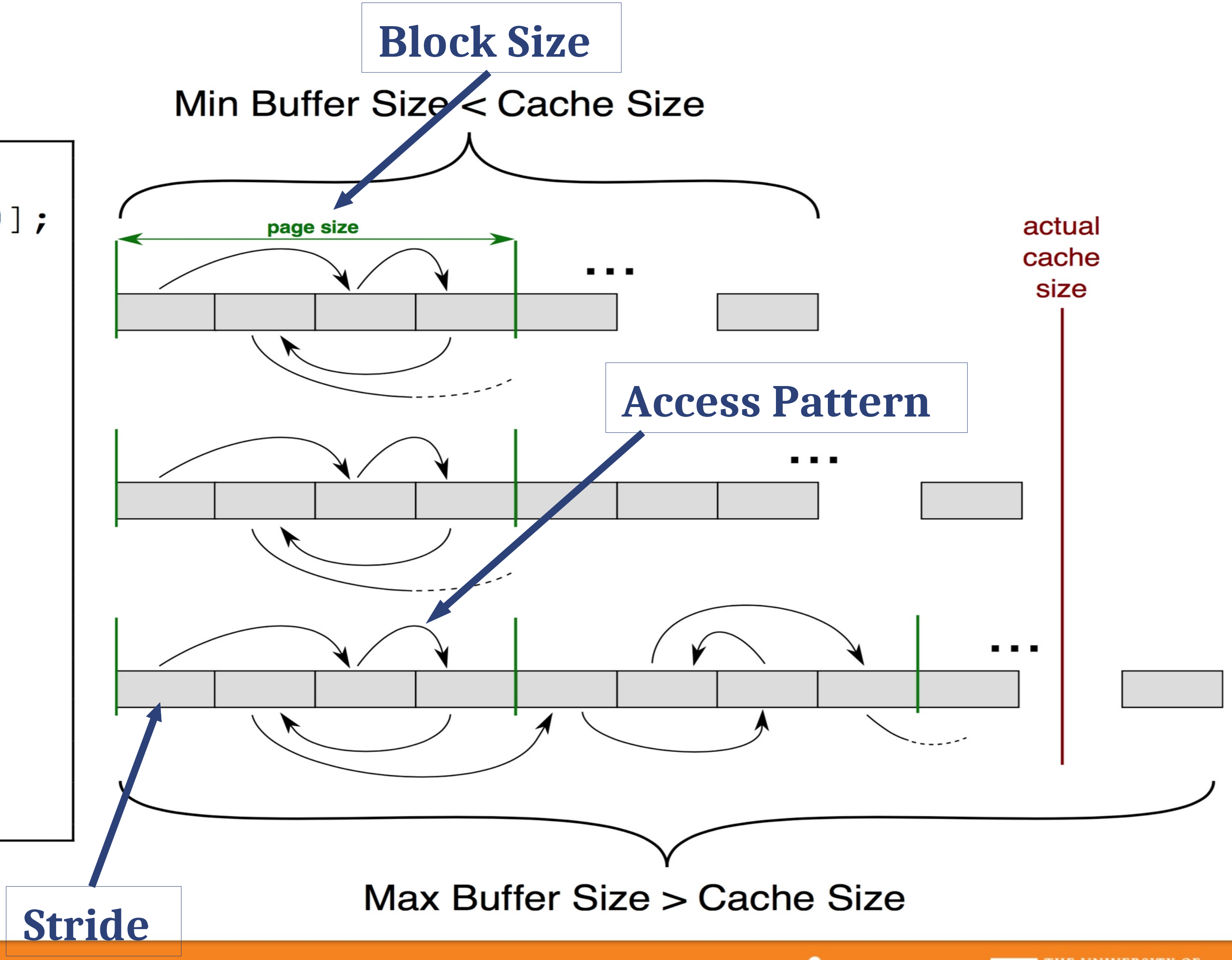
# Pointer Chasing

```
SETUP( ) {
  p = (uintptr_t **) &array[0];
  for (i = random( ) ) {
      next = &array[i];
      *p = next;
      p = (uintptr_t **) next;
  }
}

MEASURE( ) {
  start_measurement();
  for (...) {
      p = (uintptr_t **) *p;
  }
  stop_measurement();
}
```

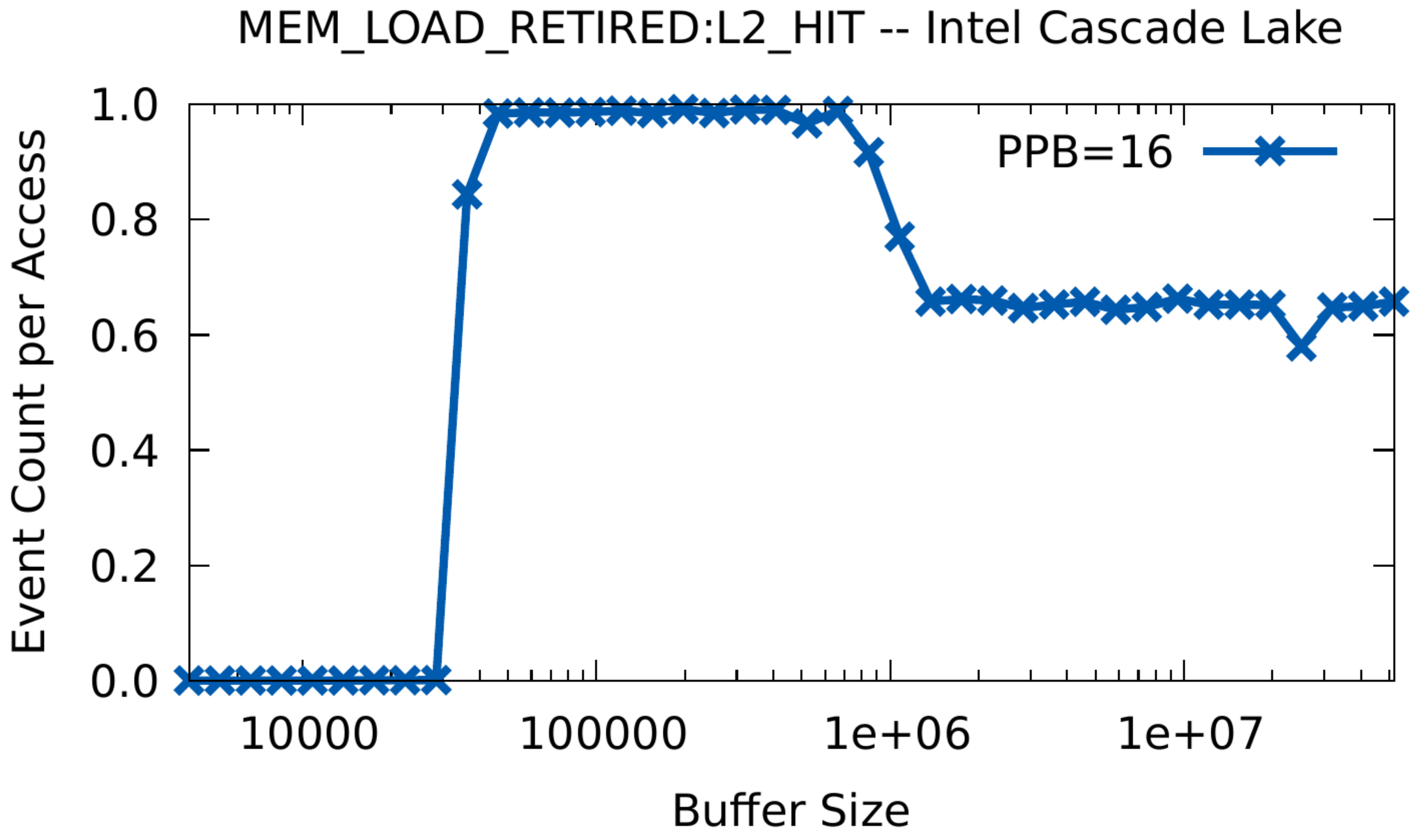


# Pointer Chasing

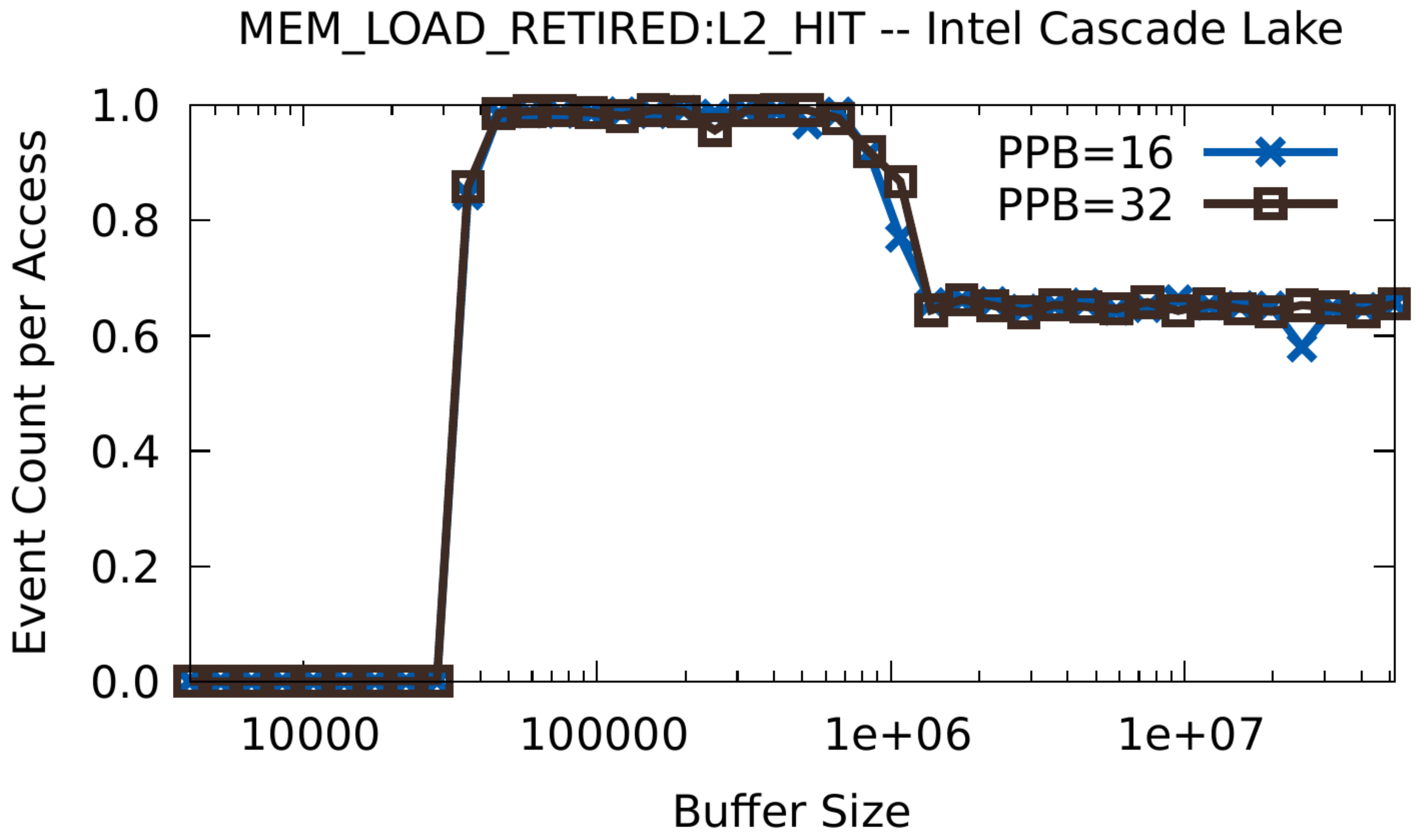
```
SETUP( ) {
  p = (uintptr_t **) &array[0];
  for (i = random( ) ) {
      next = &array[i];
      *p = next;
      p = (uintptr_t **) next;
  }
}

MEASURE( ) {
  start_measurement();
  for (...) {
      p = (uintptr_t **) *p;
  }
  stop_measurement();
}
```



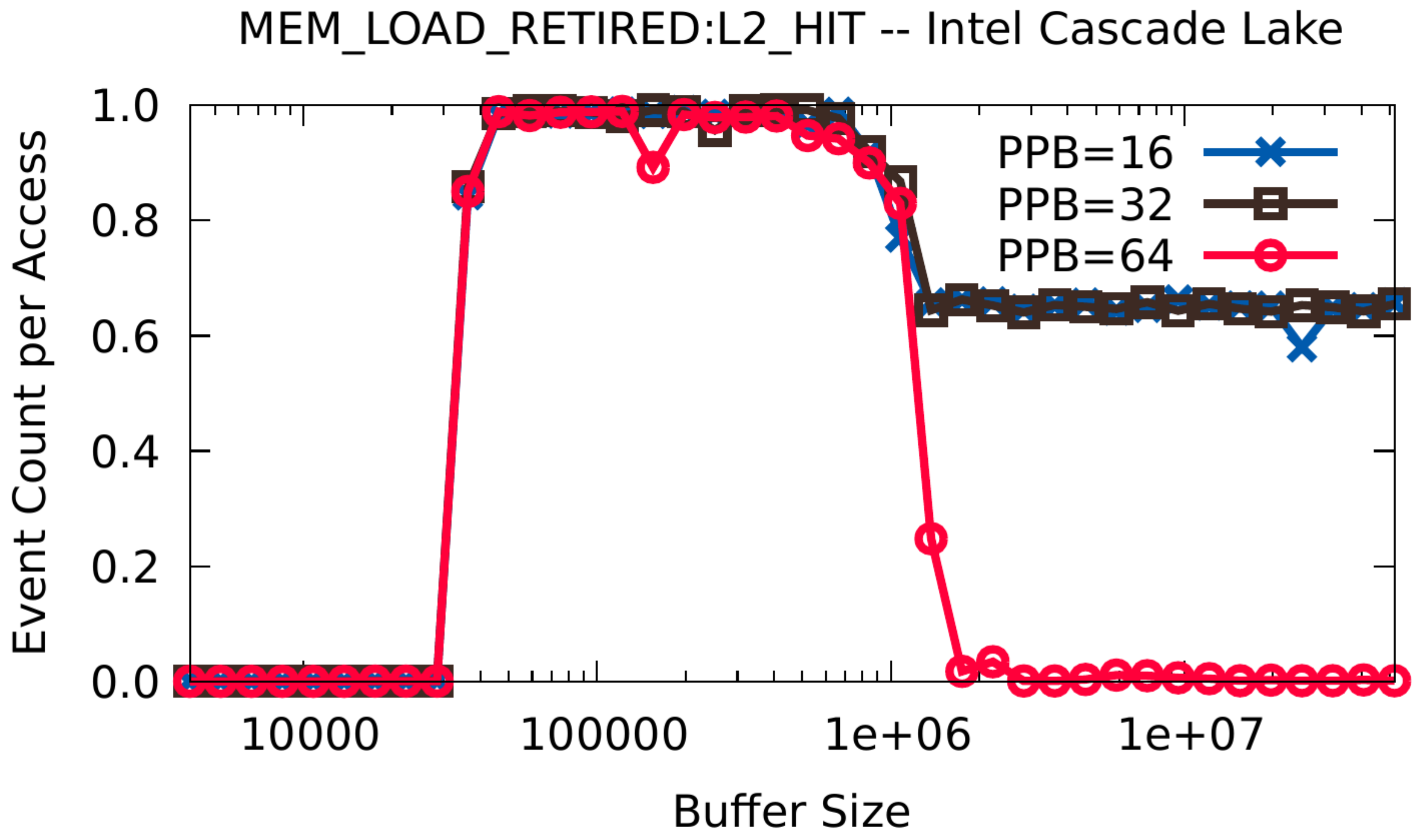
# L2 Hits, Intel Cascade Lake



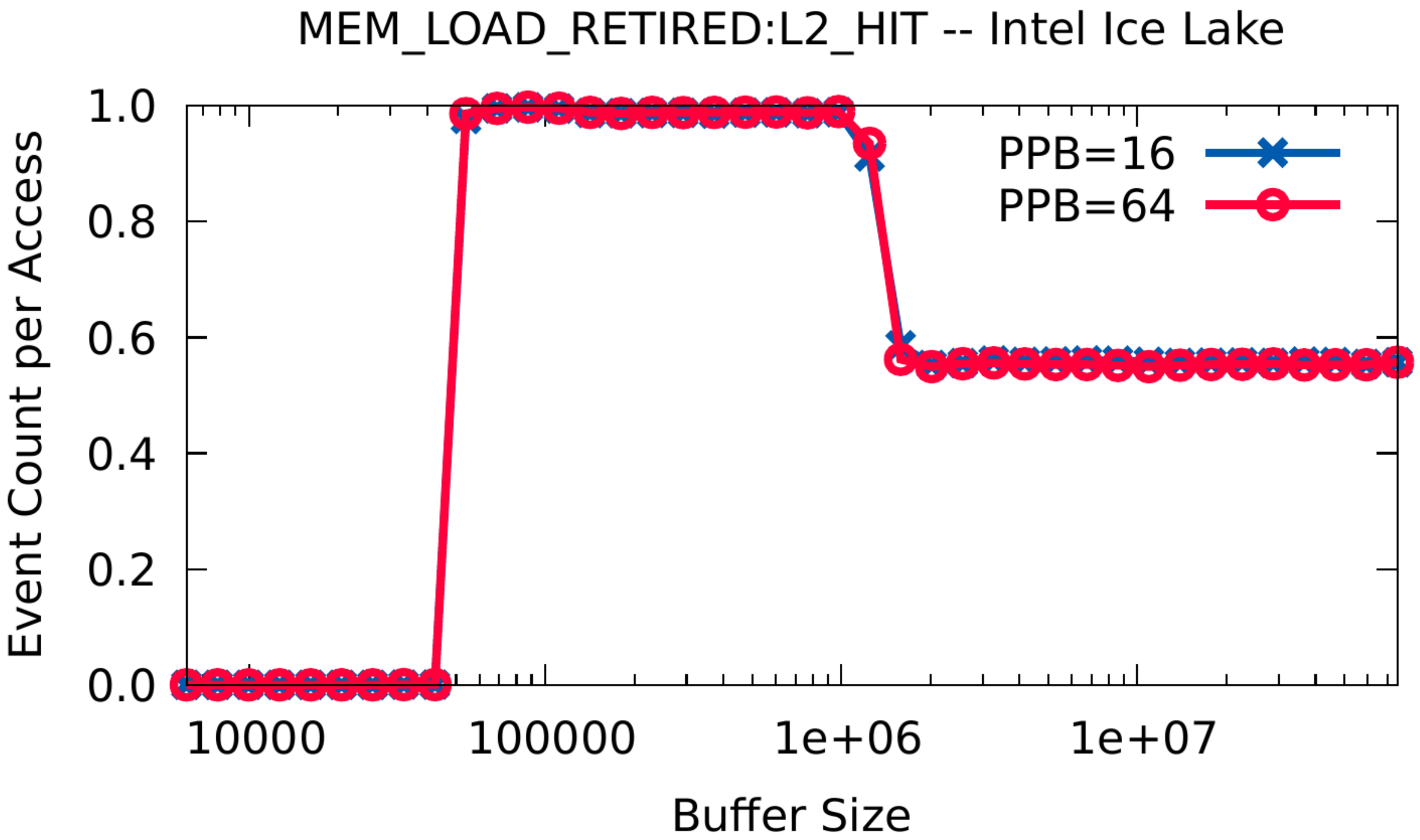
# L2 Hits, Intel Cascade Lake



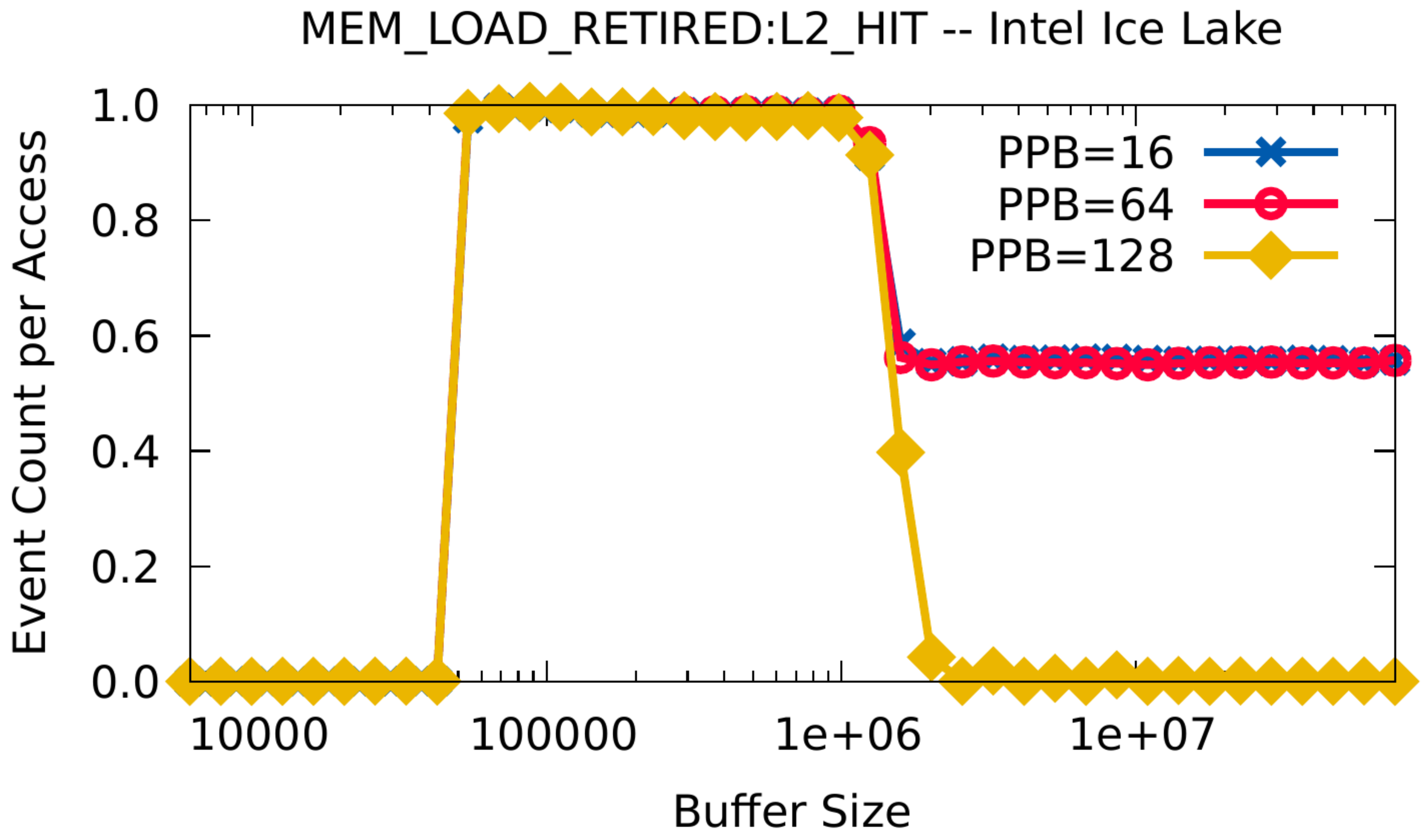
# L2 Hits, Intel Cascade Lake



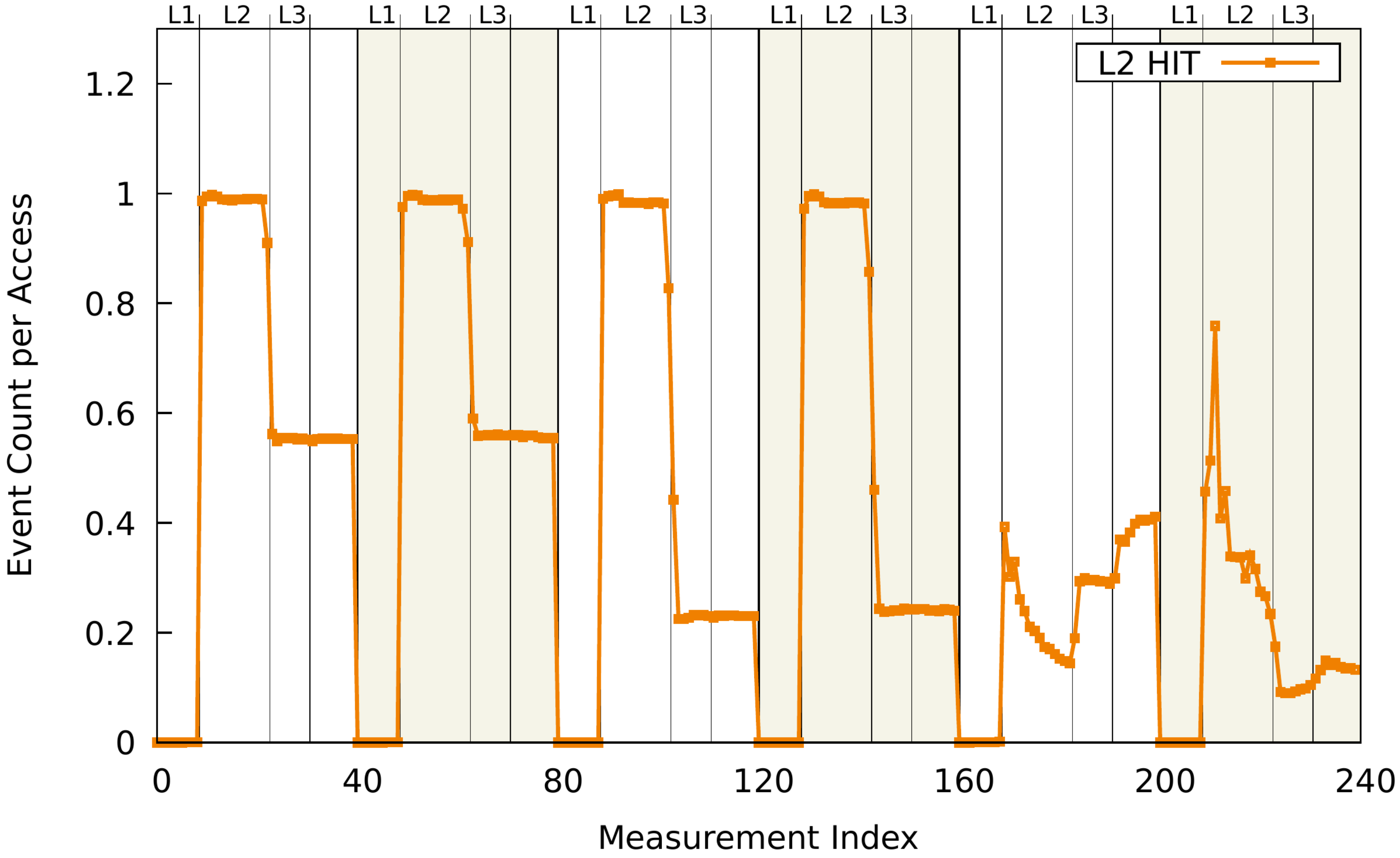
# L2 Hits, Intel Ice Lake



# L2 Hits, Intel Ice Lake

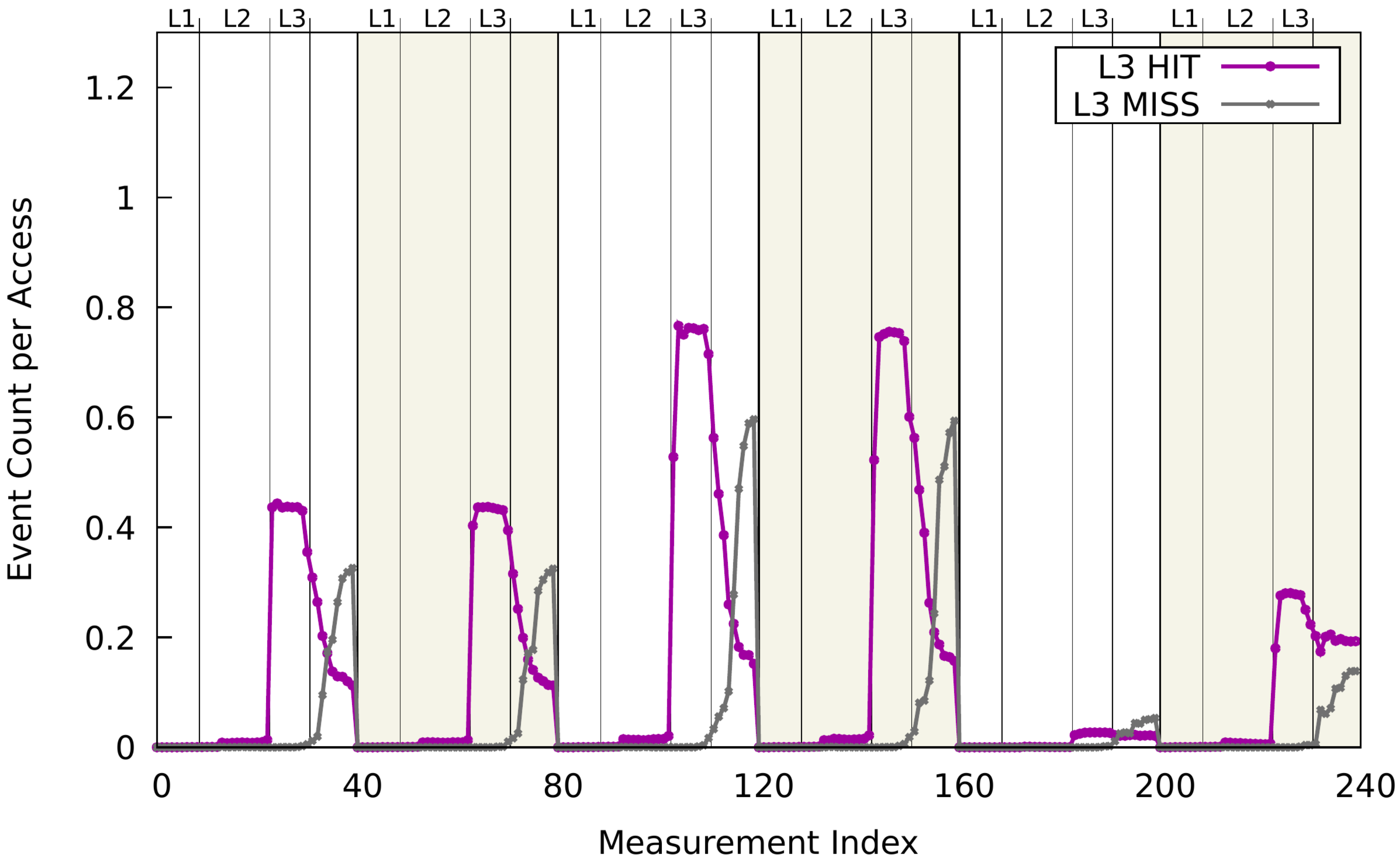


# L2 Hits

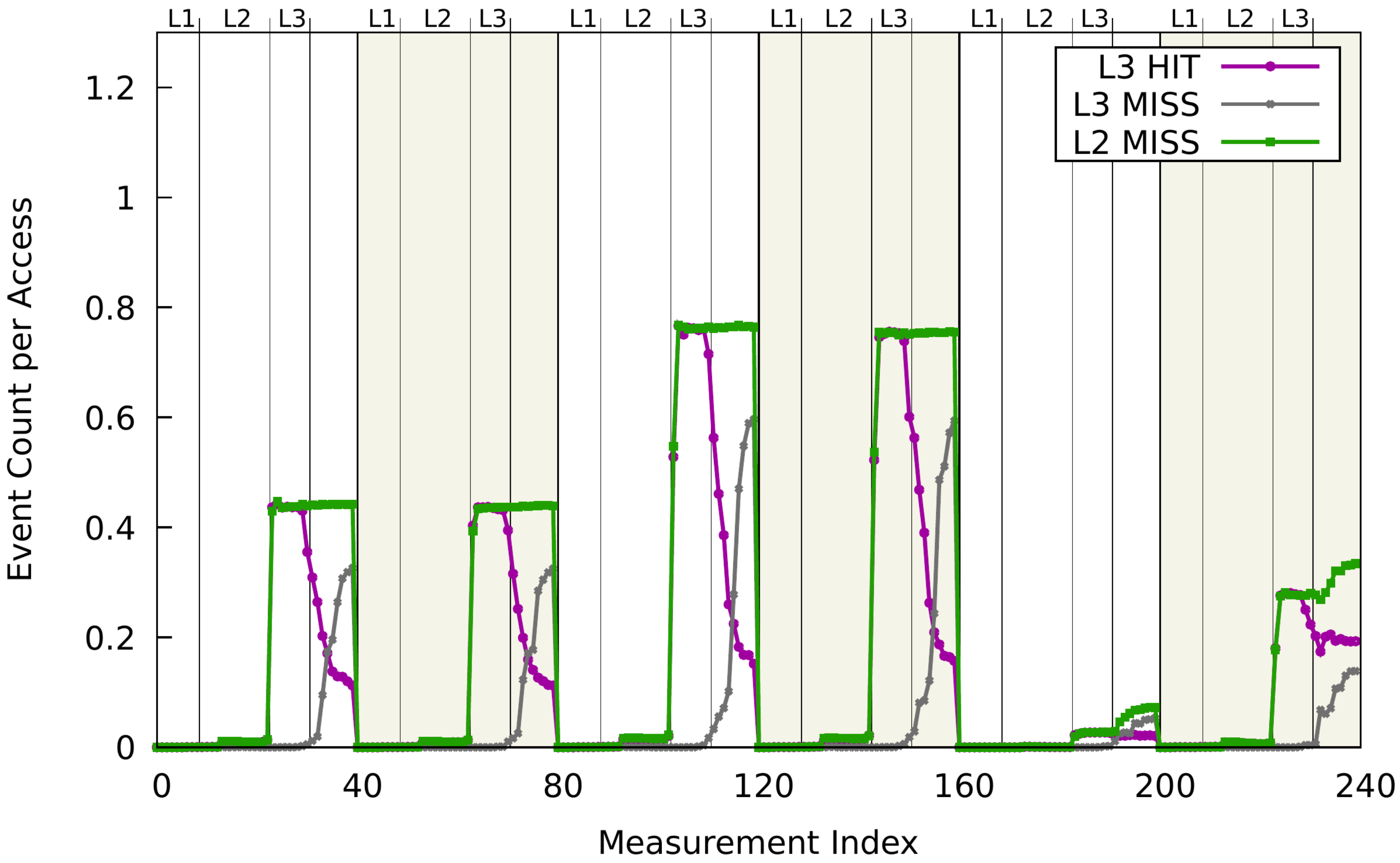


L1
L2
L3
L1
L2
L3
L1
L2
L3
L1
L2
L3
L1
L2
L3
L1
L2
L3
L2 HIT
1.2
1
0.8
0.6
0.4
0.2
0
0
40
80
120
160
200
240
Event Count per Access
Measurement Index

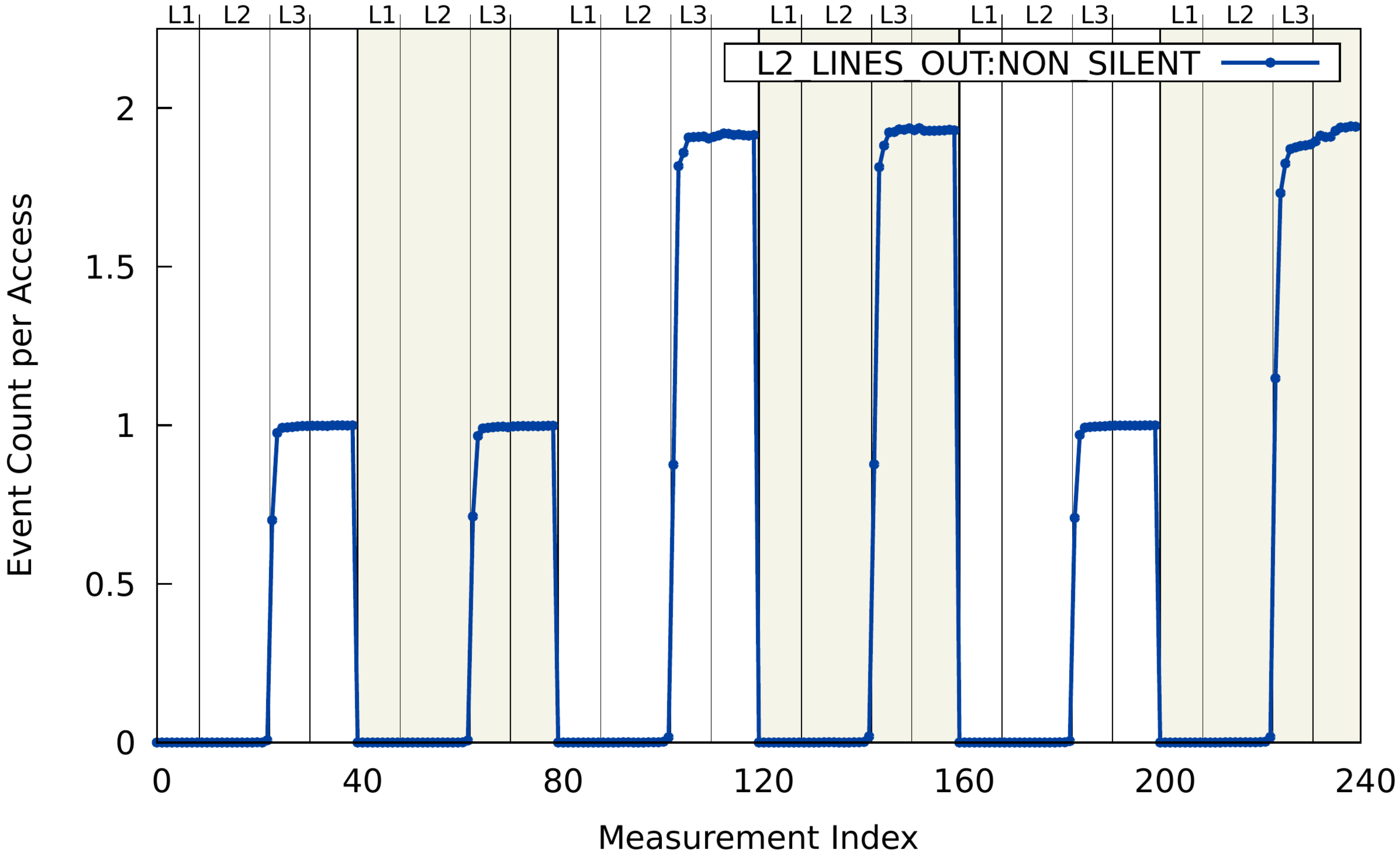
# L3 Hits & Misses



# L3 Hits + L3 Misses = L2 Misses



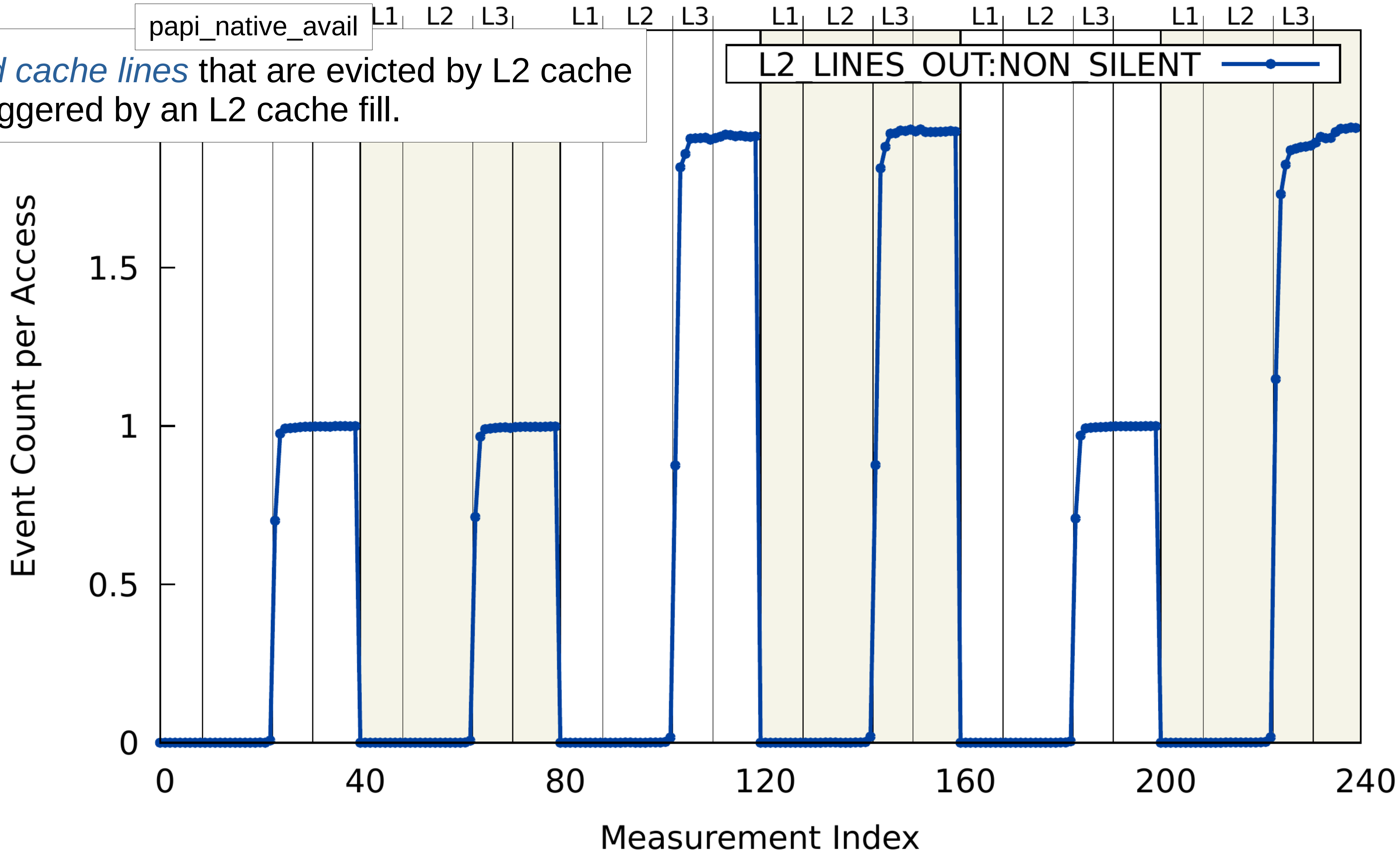
# Non-obvious results/naming



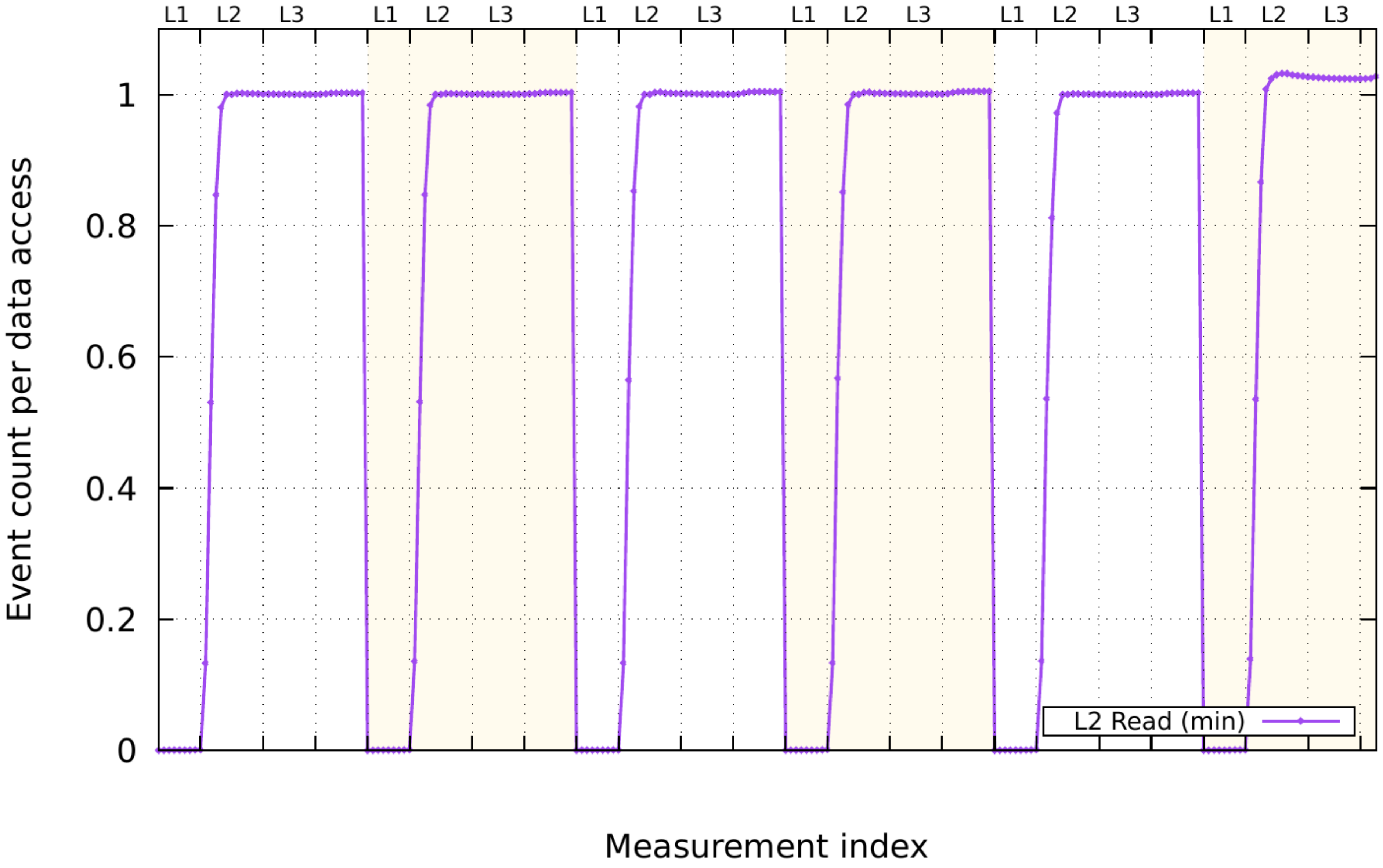
# Non-obvious results/naming

papi_native_avail

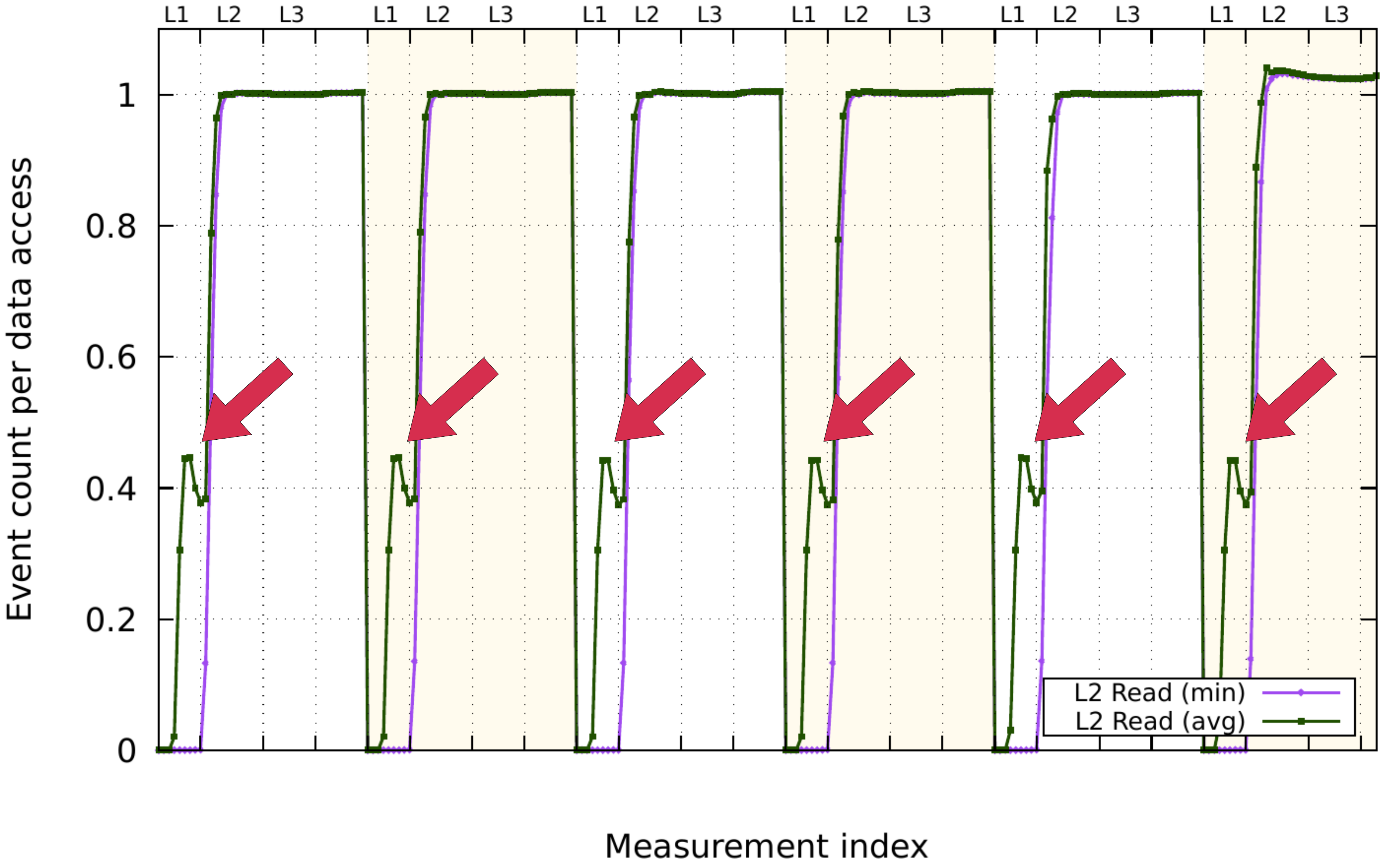
*Modified cache lines* that are evicted by L2 cache when triggered by an L2 cache fill.



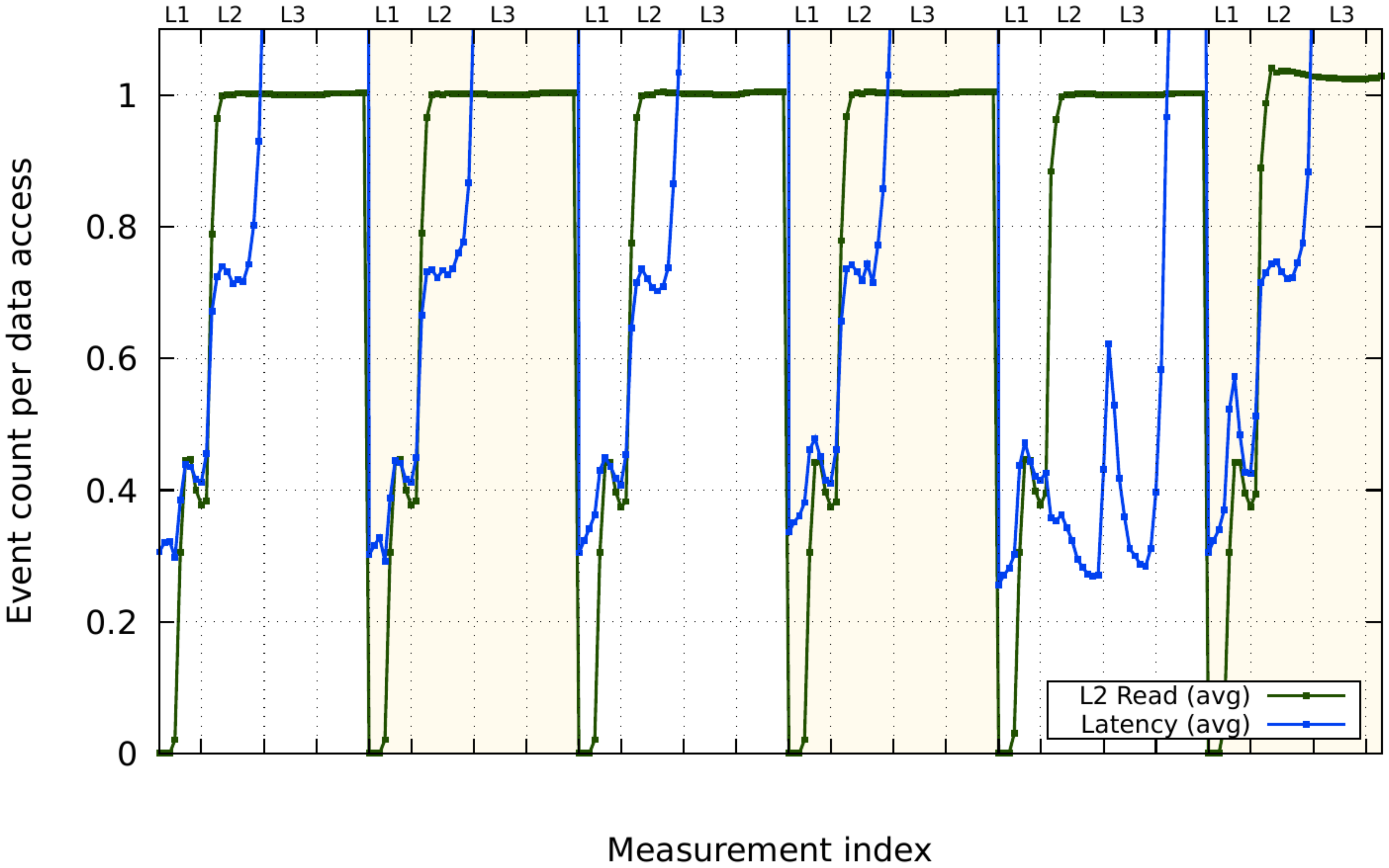
# Surprising results (AMD Zen3: EPYC 7413)



# Surprising results (AMD Zen3: EPYC 7413)



# Surprising results (AMD Zen3: EPYC 7413)



# Sysdetect component

# Available information example

| CPU | NVIDIA GPU | AMD GPU |
|---|---|---|
| ID | ID | ID |
| Name | UID | UID |
| Family/model/stepping | Name | Name |
| Sockets | Warp size | Wavefront size |
| Numas | Max threads per block | Simd per compute unit |
| Cores | Max blocks per multiproc. | Max threads per workgroup |
| Cache Size/Line Size/Lines/Assoc. | Max shm per block | Max waves per compute unit |
| Memory per numa | Max shm per multiproc. | Max shm per workgroup |
| Thread numa affinity | Block dims | Max workgroup dims |
| - | Grid dims | Max grid dims |
| - | Multiprocessor count | Compute unit count |
| - | Multiple kernels per context | Compute capability |
| - | Can map host memory | - |
| - | Can overlap compute and data xfer | - |
| - | Compute capability | - |

# Command line utility: papi_hardware_avail

```
bash~$ utils/papi_hardware_avail

Device Summary -----------------------------------------------------------------
Vendor           DevCount
GenuineIntel       (1)
 \-> Status: Device Initialized
NVIDIA             (2)
 \-> Status: Device Initialized
AMD/ATI            (0)
 \-> Status: ROCm not configured, no ROCm device available


Device Information -------------------------------------------------------------
Vendor                                      : GenuineIntel
Id                                          : 0
Name                                        : Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz
CPUID                                       : Family/Model/Stepping 6/63/2 0x06/0x3f/0x02
Sockets                                     : 2
Numa regions                                : 2
Cores per socket                            : 10
Cores per NUMA region                       : 20
SMT threads per core                        : 2
...

Vendor                                      : NVIDIA
Id                                          : 0
Name                                        : Tesla K80
Warp size                                   : 32
Max threads per block                       : 1024
Max blocks per multiprocessor               : 16
Max shared memory per block                 : 49152
...
```

# Summary

- PAPI 7.0 coming soon!
- Support for GPU counters/metrics across vendors.
- Support for power management on CPUs & GPUs.
- Software Defined Events as a standalone library.
- Counter Analysis Toolkit provides hardware insights.
- API & utility for detecting available hardware