PAPI 7.0 and beyond

15th Scalable Tools Workshop

Anthony Danalis, Heike Jagode, Giuseppe Congiu, Daniel Barry

Tahoe City, CA

June 18-22, 2023

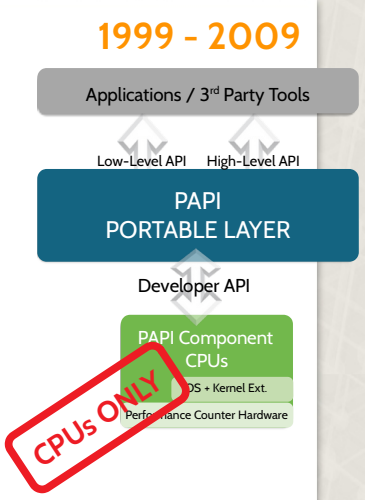
ICL INNOVATIVE COMPUTING LABORATORY

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



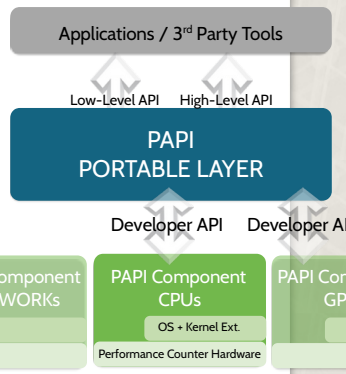
- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i.e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O, FS, Energy/Power.
- PAPI enables SW engineers to see, in near real time, the relation between **SW performance**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

SUPPORTED ARCHITECTURES:

- AMD **up to Zen4**
- ARM Cortex A8, A9, A15, ARM64, **ARM uncore-support**
-
- IBM Blue Gene Series
- IBM Power Series, **PCP for POWER9-nest**
- Intel KNC, KNL, KNM, up to **Sapphire Rapids**

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.**

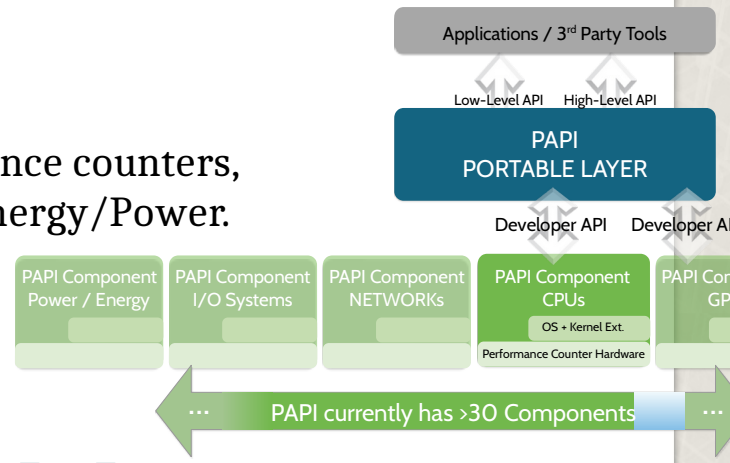
SUPPORTED ARCHITECTURES:

- AMD **up to Zen4**
- AMD **GPUs up to MI2x0**
- 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 KNC, KNL, KNM, up to **Sapphire Rapids**

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

PAPI

2009 - 2018



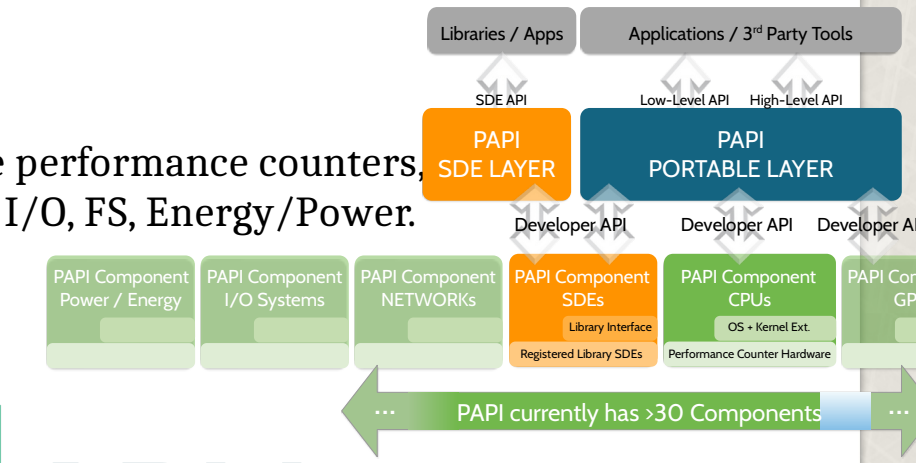
- Library that provides a **consistent interface** (and methodology) for hardware performance counters, found across the system: i.e., CPUs, GPUs, on-/off-chip Memory, Interconnects, I/O, FS, Energy/Power.
- PAPI enables SW engineers to see, in near real time, the relation between **SW performance**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

SUPPORTED ARCHITECTURES:

- AMD **up to Zen4**, **power for Fam17h**
- AMD **GPUs up to MI2x0**, **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 KNC, KNL, KNM, up to **Sapphire Rapids**
- 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:

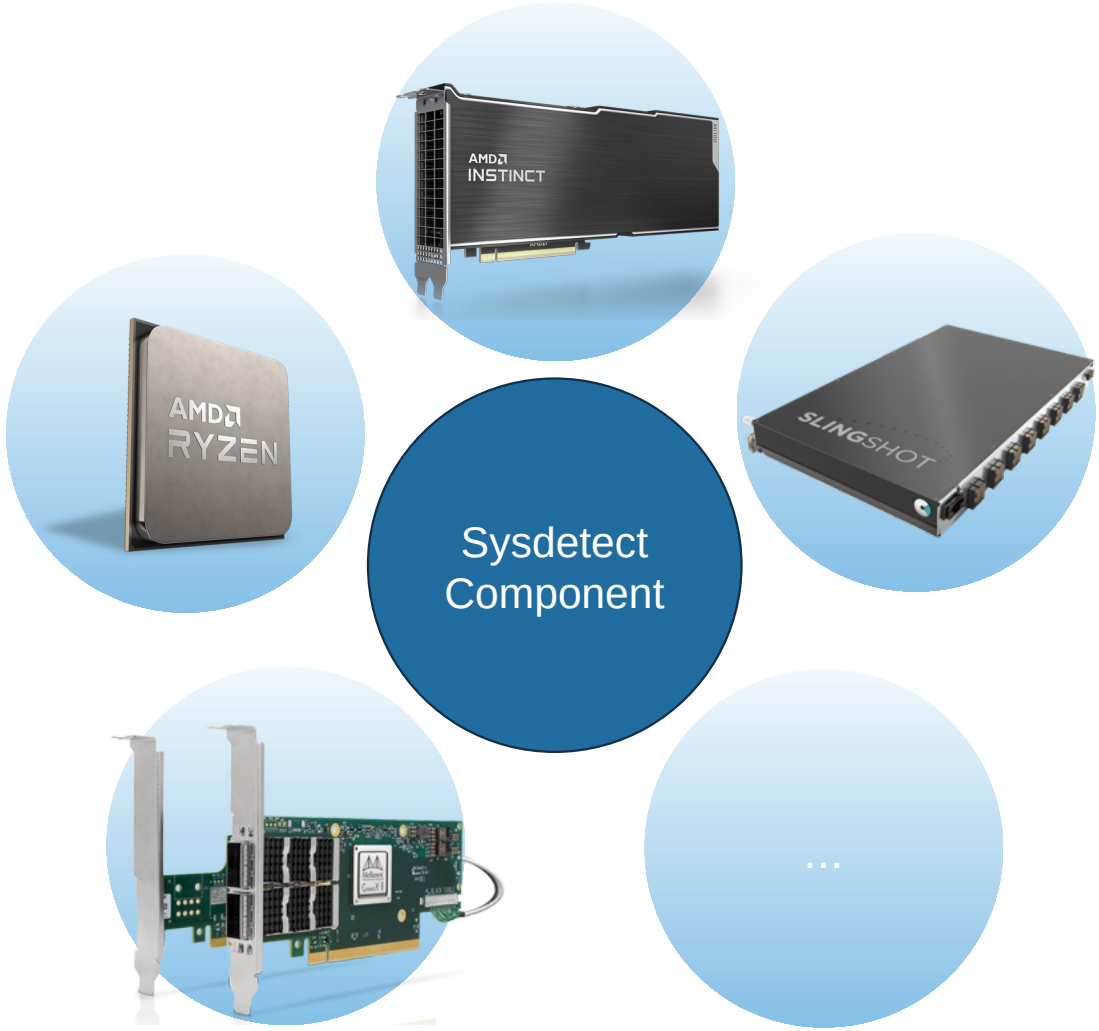
- AMD **up to Zen4**, **power for Fam17h**
- AMD **GPUs up to MI2x0**, **power, temperature, fan**
- ARM Cortex A8, A9, A15, ARM64, **ARM uncore-support**
- CRAY: Gemini and Aries interconnects, power/energy
- IBM Blue Gene Series, Q: 5D-Torus, I/O system, EMON power/energy
- IBM Power Series, **PCP for POWER9-nest, power monitoring & capping on POWER9**
- Intel KNC, KNL, KNM, up to **Sapphire Rapids**
- 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**

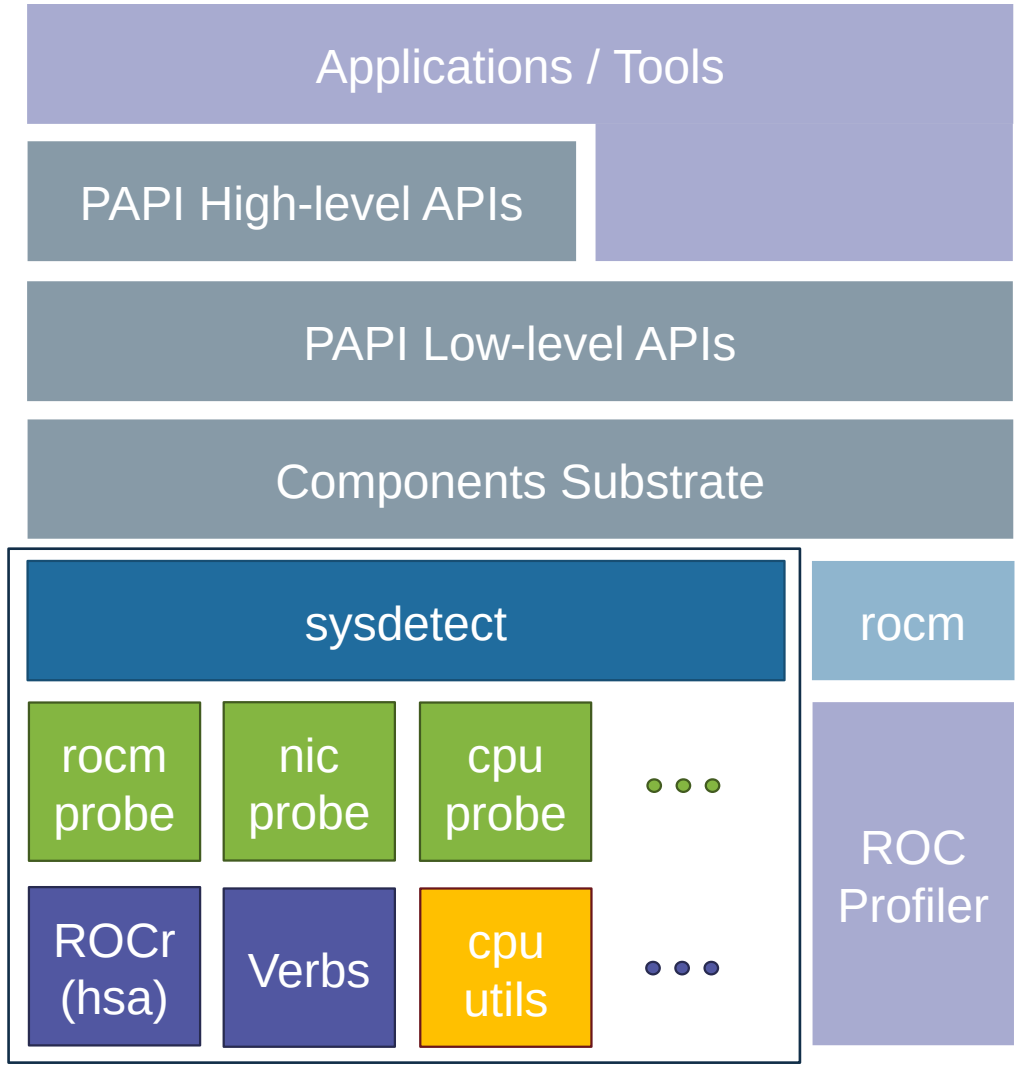
The PAPI repo has moved:

https://github.com/icl-utk-edu/papi

Sysdetect component

System Detection Capabilities





System Detection Component

```
$ papi_hardware_avail

Device Summary -------------------------------------------------------
Vendor           DevCount
AuthenticAMD       (1)
 \-> Status: Device Initialized
AMD                (1)
 \-> Status: MPI not configured

Device Information ---------------------------------------------------
Vendor                                  : AuthenticAMD
Id                                      : 0
Name                                    : AMD EPYC 7413 24-Core Processor
CPUID                                   : Family/Model/Stepping 25/1/1
0x19/0x01/0x01
Sockets                                 : 2
Numa regions                            : 2
Cores per socket                        : 24
Cores per NUMA region                   : 48
SMT threads per core                    : 2
L1i Cache                               : Size/LineSize/Lines/Assoc 32KB/64B/512/8
L1d Cache                               : Size/LineSize/Lines/Assoc 32KB/64B/512/8
...
Numa Node 0 Memory                      : 264023436 KB
Numa Node 1 Memory                      : 264199004 KB
Numa Node 0 Threads                     : 0 1 2 3 4 5 6 7 8 9 10 11 ...
Numa Node 1 Threads                     : 24 25 26 27 28 29 30 31 32 ...

Vendor                                  : AMD
Id                                      : 0
Name                                    : gfx908
```

Sysdetect low-level calls

- `PAPI_enum_dev_type`
  - `In  - modifier  (e.g. PAPI_DEV_TYPE_ENUM__ROCM)`
  - `Out - handle    (opaque)`
- `PAPI_get_dev_type_attr`
  - `In  - handle`
  - `In  - attribute  (e.g. PAPI_DEV_TYPE_ATTR__INT_PAPI_ID)`
  - `Out - value`
- `PAPI_get_dev_attr`
  - `In  - handle`
  - `In  - identifier (e.g. numa/gpu/... number)`
  - `In  - attribute  (e.g. PAPI_DEV_ATTR__ROCM_ULONG_UID)`
  - `Out - value`

device type attribute

```
Wavefront size                          : 64
SIMD per compute unit                   : 4
Max threads per workgroup               : 1024
Max waves per compute unit              : 40
Max shared memory per workgroup         : 65536
Max workgroup dim x                     : 1024
Max workgroup dim y                     : 1024
Max workgroup dim z                     : 1024
Max grid dim x                          : 4294967295
Max grid dim y                          : 4294967295
Max grid dim z                          : 4294967295
Compute unit count                      : 120
Compute capability                      : 1.1
```

Counter Analysis Toolkit (CAT)

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

0% Mispredicted

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

50% Mispredicted

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

150% Taken

```
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);
```

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);
```

CAT kernel example for Branch Events

0% Mispredicted

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

50% Mispredicted

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

Taken

```
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);
```

100% Direct

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

Notice: No Memory Accesses

CAT kernel example for Branch Events

0% Mispredicted

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

50% Mispredicted

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

100% Taken

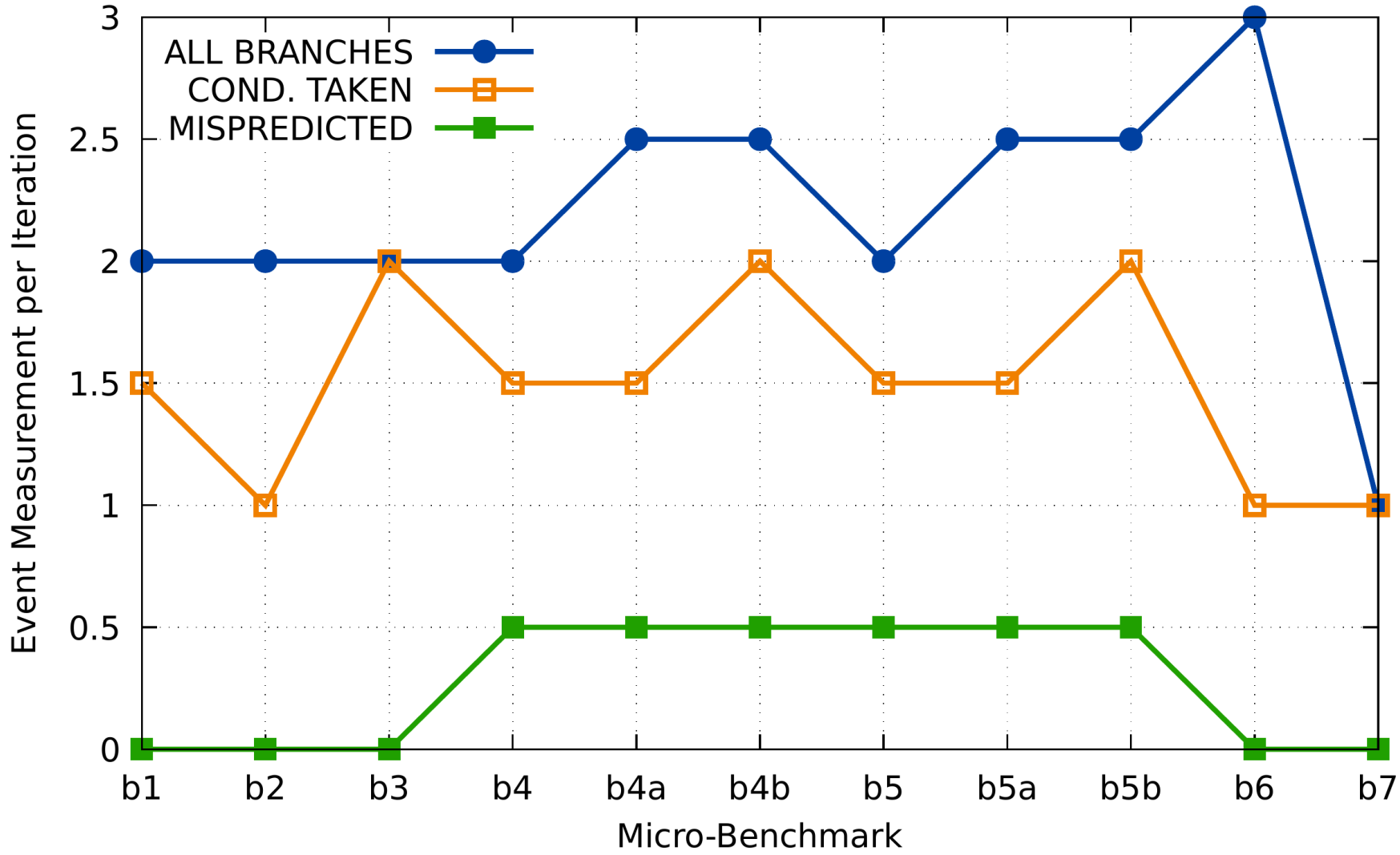
```
do{
    iter_c
    BUSY_W
    BRNG(
    if (
        if(
            global_var
        }
        global_var1+=2;
        BUSY_WORK();
    }
}while(iter_count<size);
```

100% Direct

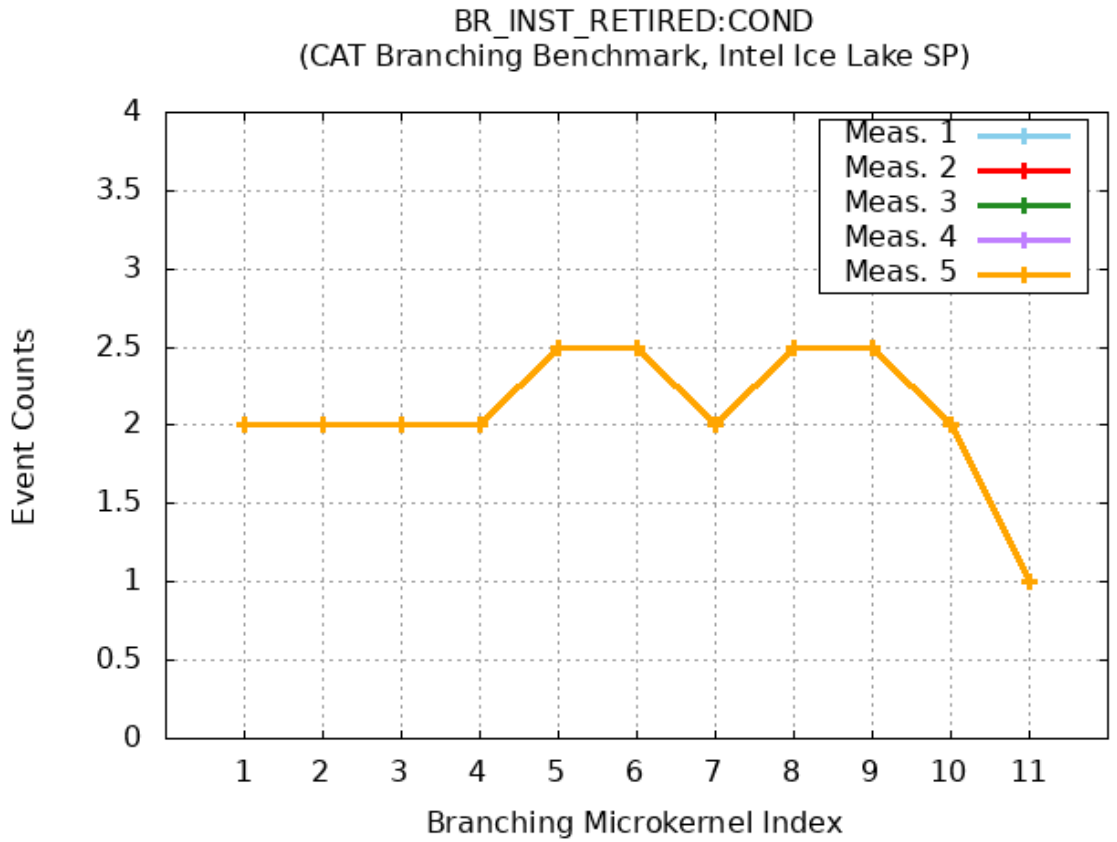
```
var2 ){
zzz:
    BRNG();
}while(iter_count<size);
```

Notice: No Memory Accesses
No Large Code base
No Floating point

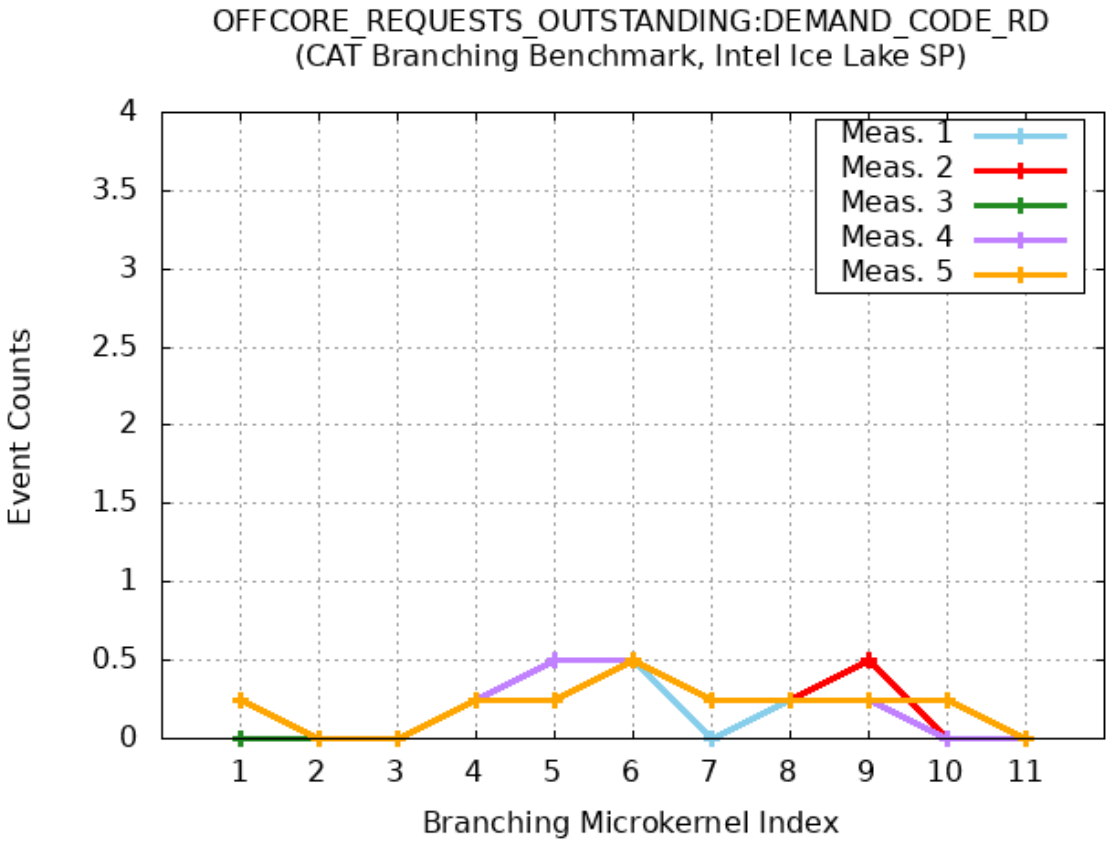
Unique Responses Reveal Mapping to Preset Events

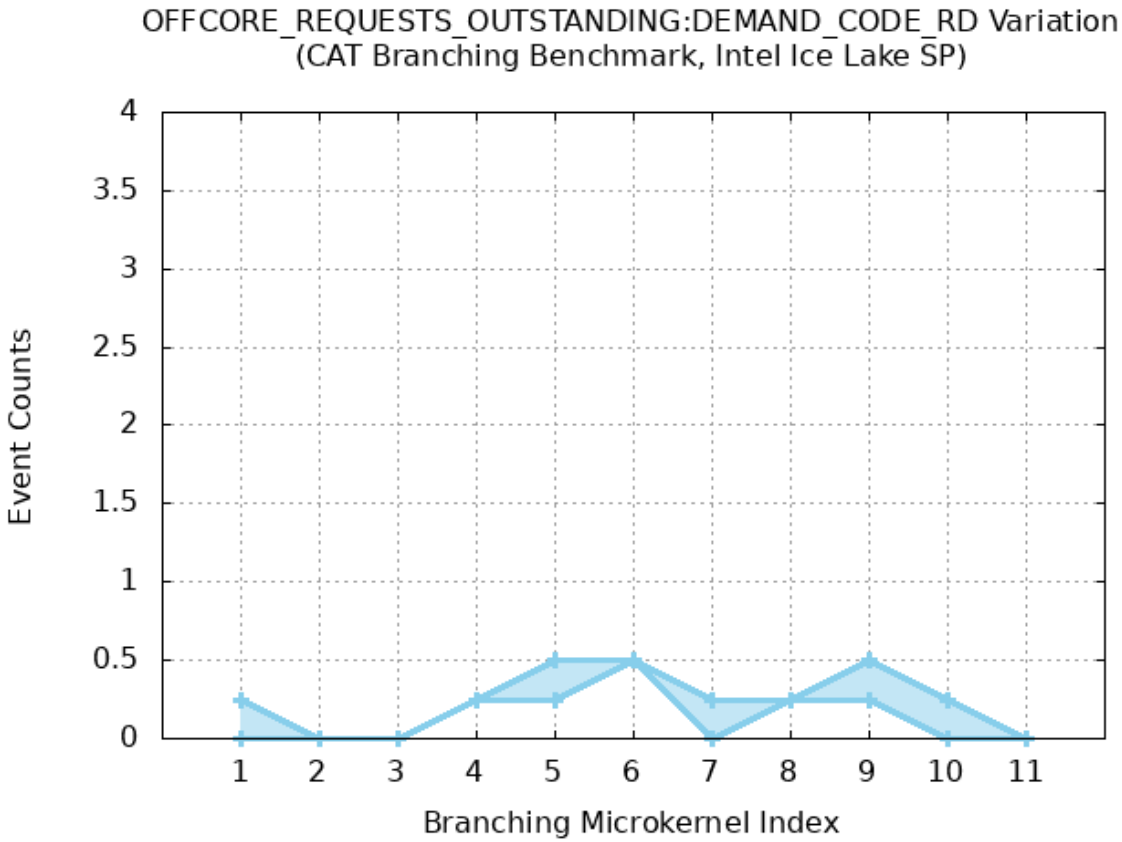


Noise in Measurements: Relevant Event

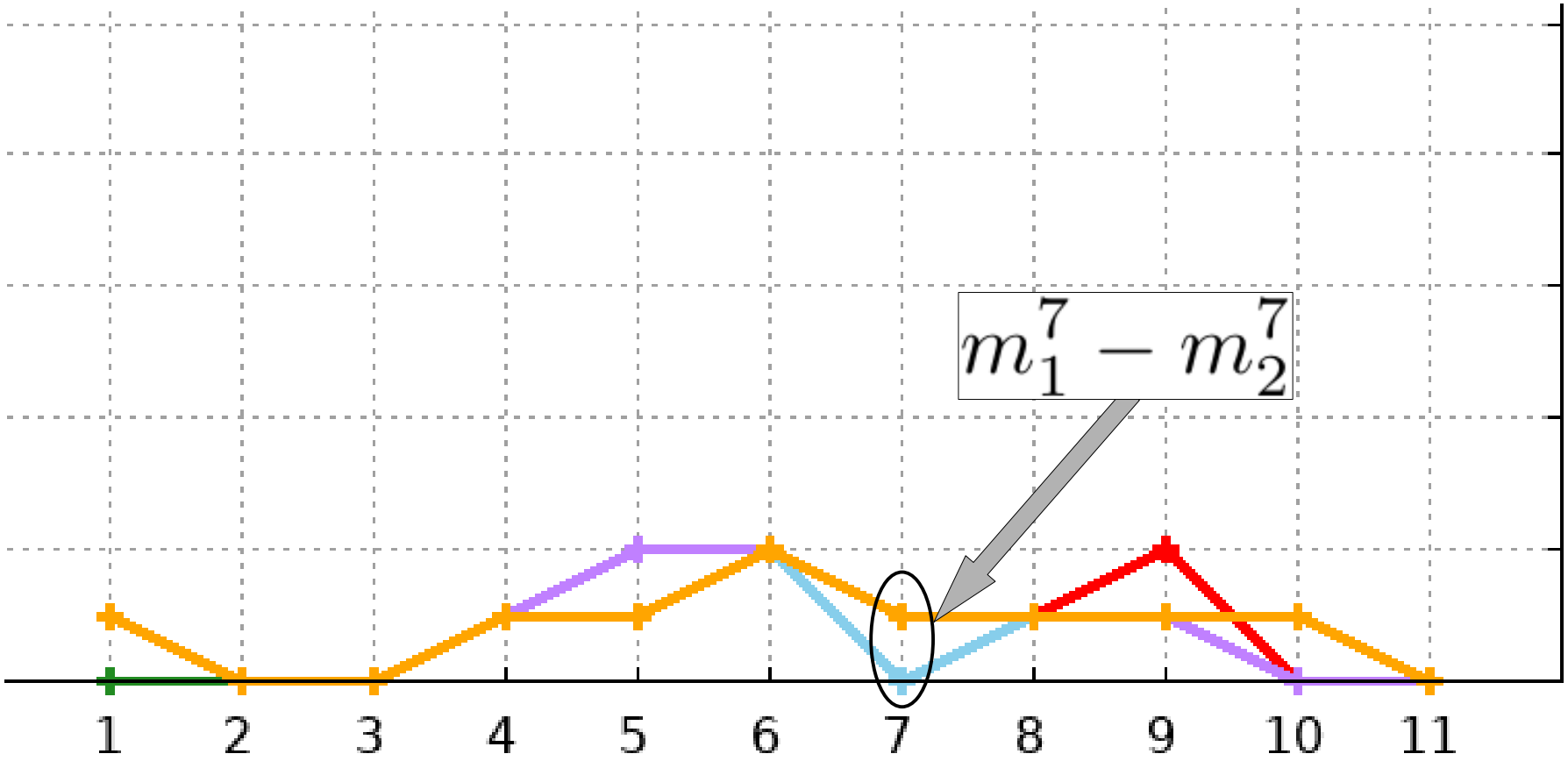


Noise in Measurements: Irrelevant Event

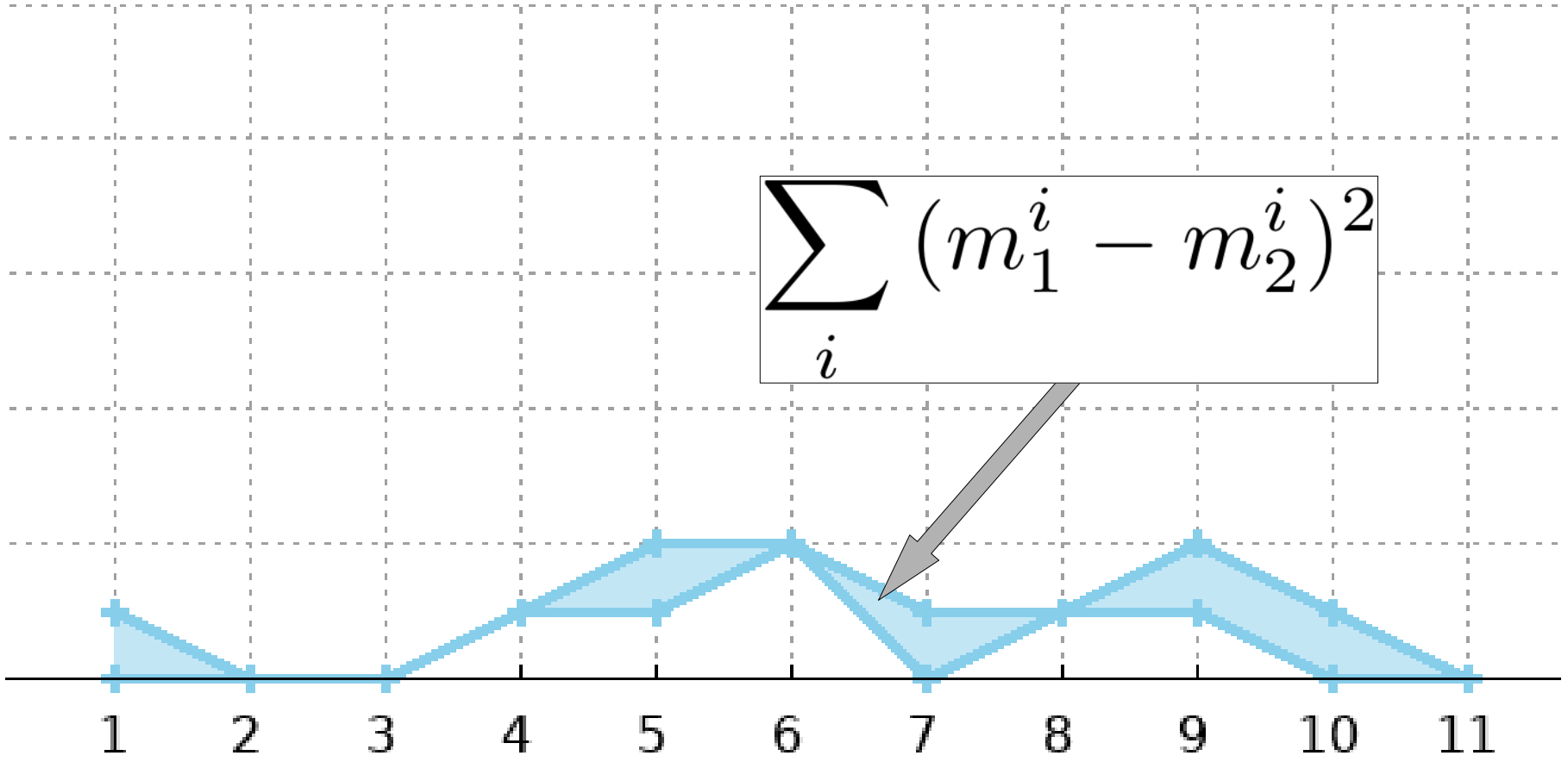




Distance Between Two Measurements



Distance Between Two Curves



RNMSE: Root Normalized Mean Square Error

$$RNMSE(1,2) = \sqrt{\frac{1}{N}\sum_i \frac{(m_1^i - m_2^i)^2}{\overline{m_1}\cdot\overline{m_2}}}$$

$$\overline{m_1} = \frac{1}{N}\sum_i m_1^i$$

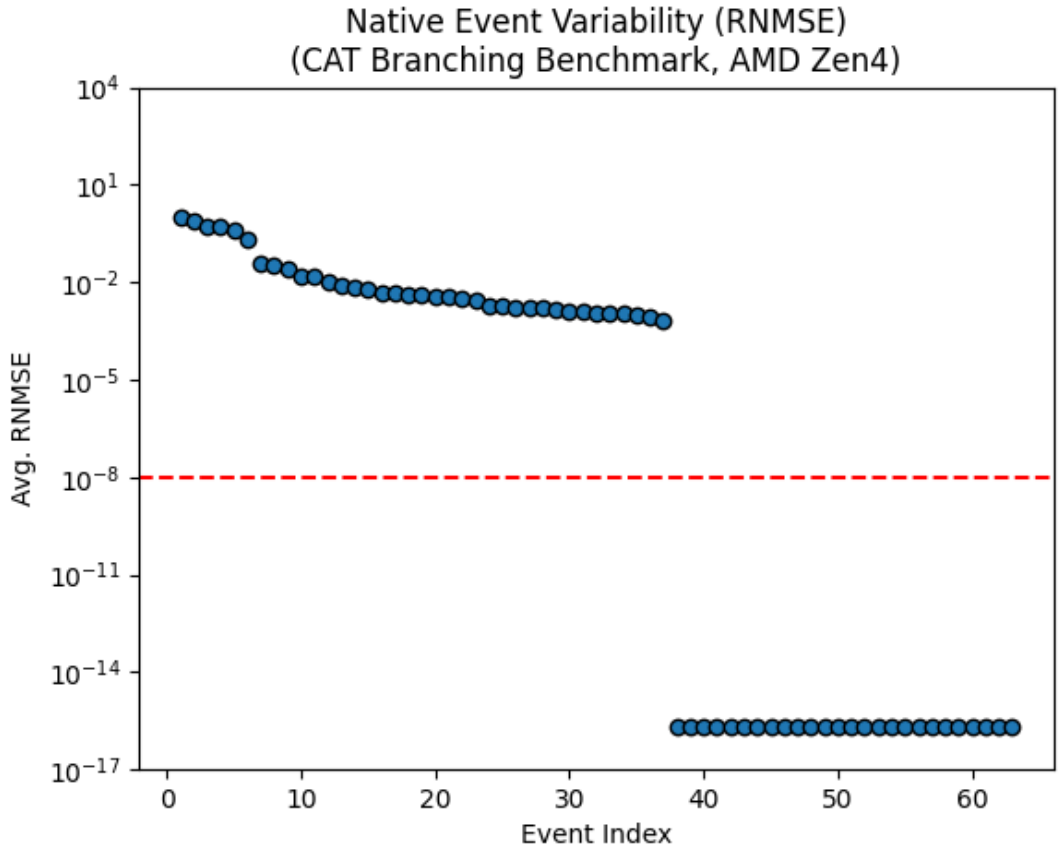
$$\overline{m_2} = \frac{1}{N}\sum_i m_2^i$$

RNMSE: Root Normalized Mean Square Error

$$RNMSE(1,2) = \sqrt{\frac{1}{N}\sum_i \frac{(m_1^i - m_2^i)^2}{\overline{m_1}\cdot\overline{m_2}}}$$

$$\overline{m_1} = \frac{1}{N}\sum_i m_1^i$$

$$\overline{m_2} = \frac{1}{N}\sum_i m_2^i$$



Events with Low Noise Measurements (Zen4)

CYCLES_NO_RETIRE:EMPTY
OPS_SOURCE_DISPATCHED_FROM_DECODER:DECODER
RETIRED_BRANCH_INSTRUCTIONS
RETIRED_BRANCH_INSTRUCTIONS_MISPREDICTED
RETIRED_BRANCH_MISPREDICTED_DIRECTION_MISMATCH
RETIRED_CONDITIONAL_BRANCH_INSTRUCTIONS
RETIRED_FUSED_INSTRUCTIONS
RETIRED_INSTRUCTIONS
RETIRED_OPS
RETIRED_TAKEN_BRANCH_INSTRUCTIONS
RETIRED_TAKEN_BRANCH_INSTRUCTIONS_MISPREDICTED
RETIRED_UNCONDITIONAL_BRANCH_INSTRUCTIONS

perf::BRANCH-INSTRUCTIONS
perf::BRANCH-LOAD-MISSES
perf::BRANCH-LOADS
perf::BRANCH-MISSES
perf::BRANCHES
perf::INSTRUCTIONS
perf::L1-ICACHE-LOADS
perf::PERF_COUNT_HW_BRANCH_INSTRUCTIONS
perf::PERF_COUNT_HW_BRANCH_MISSES
perf::PERF_COUNT_HW_CACHE_BPU:ACCESS
perf::PERF_COUNT_HW_CACHE_BPU:MISS
perf::PERF_COUNT_HW_CACHE_BPU:READ
perf::PERF_COUNT_HW_CACHE_L1I:ACCESS
perf::PERF_COUNT_HW_INSTRUCTIONS

Software Defined Events (SDE)

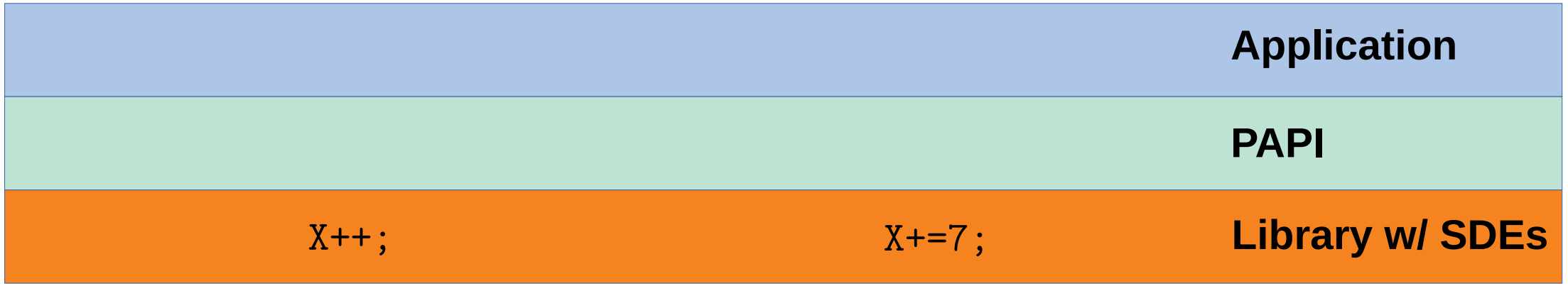
PAPI Software Defined Events (SDEs)

Support for Events that originate in Software Layers

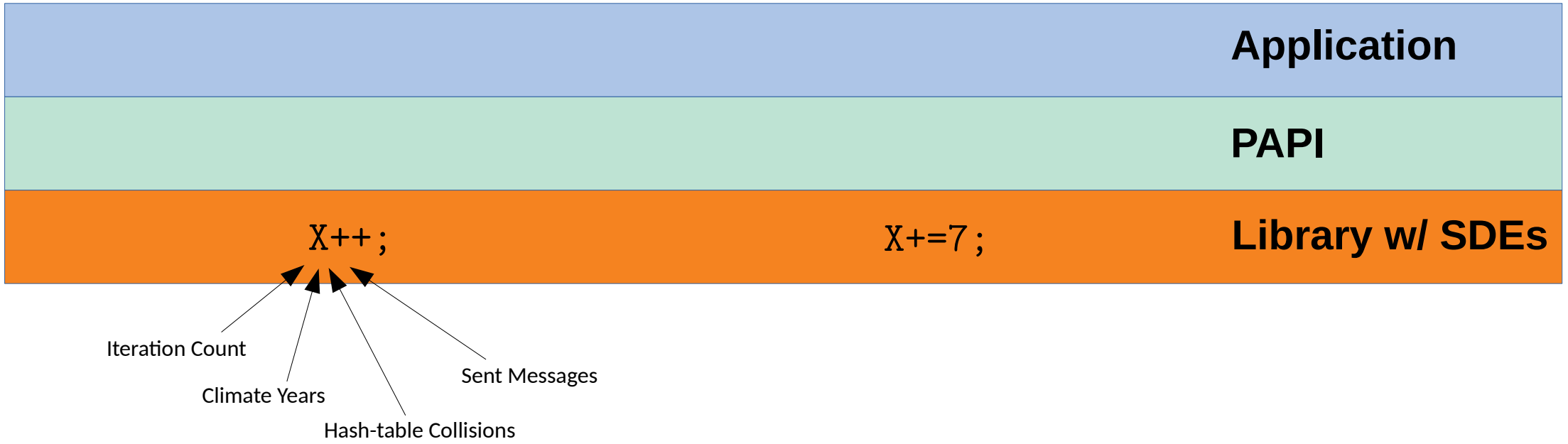
SDEs enable **software** layers to export **arbitrary information** as if it came from hardware counters

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

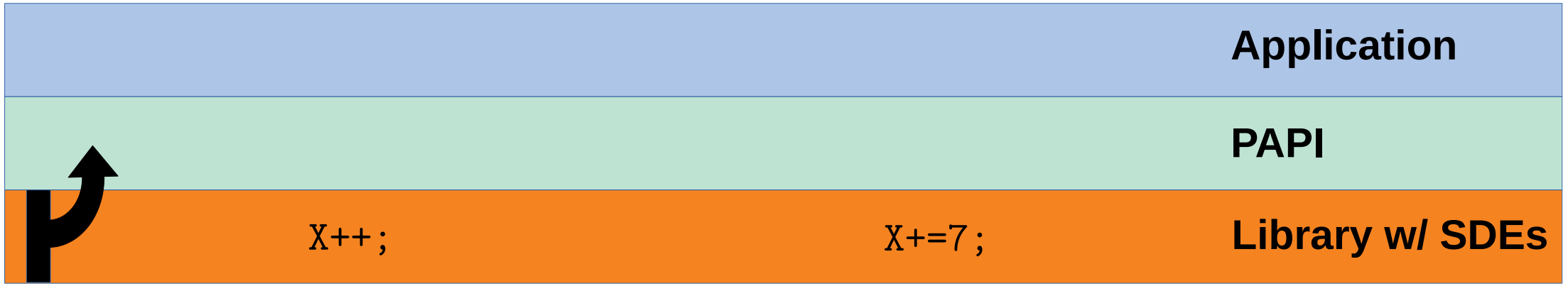
Pull mode: Low overhead (down to zero)



Pull mode: Low overhead (down to zero)



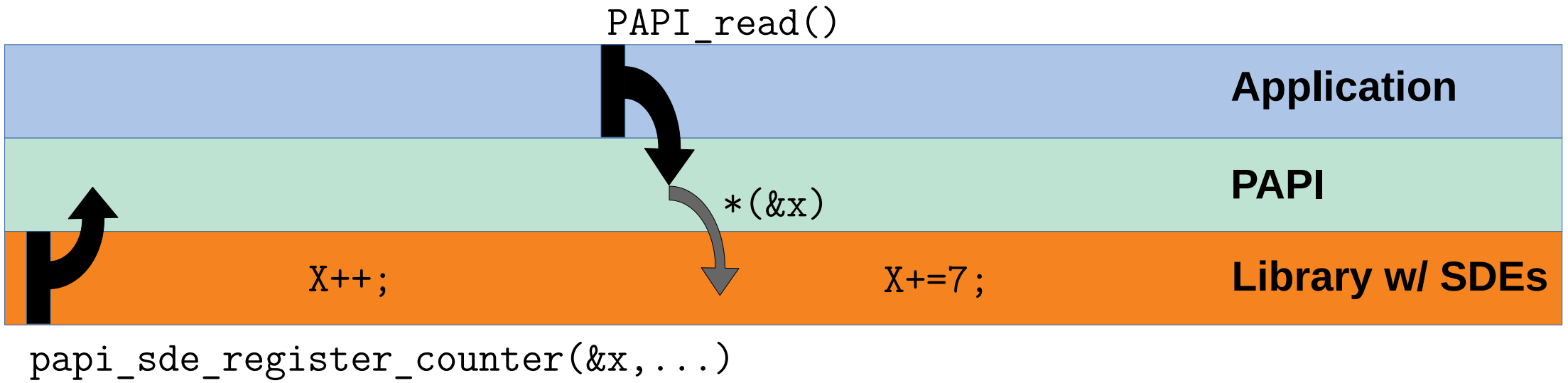
Pull mode: Low overhead (down to zero)




```
papi_sde_register_counter(&x,...)
```

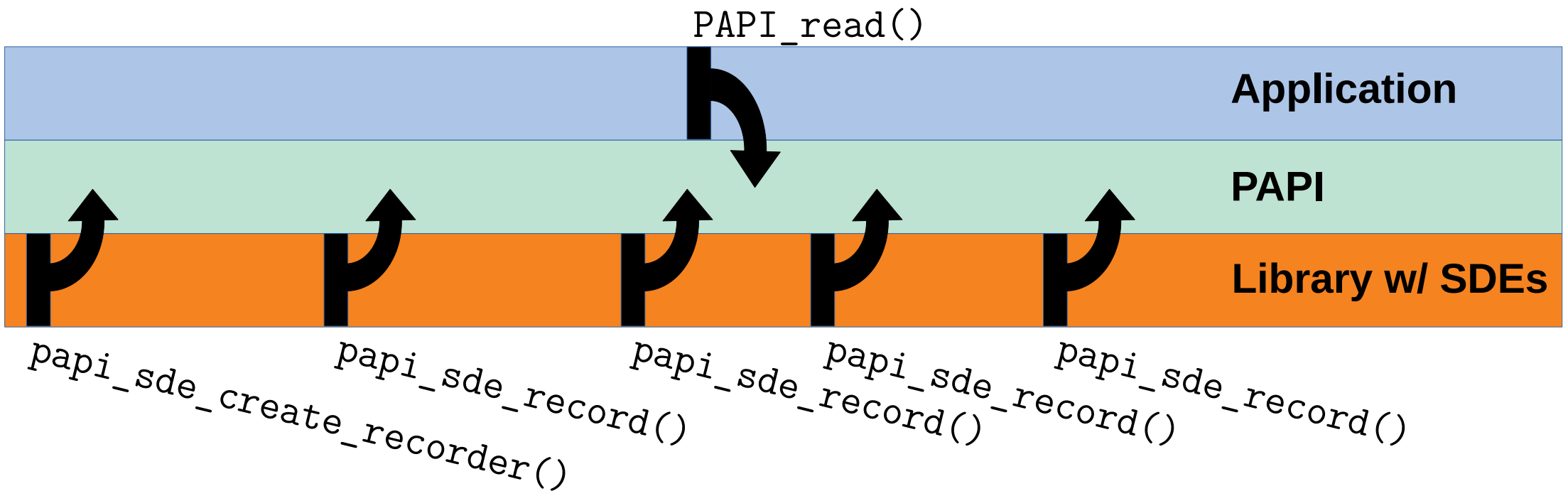
Pull mode: Low overhead (down to zero)

The application reads whenever it deems necessary.

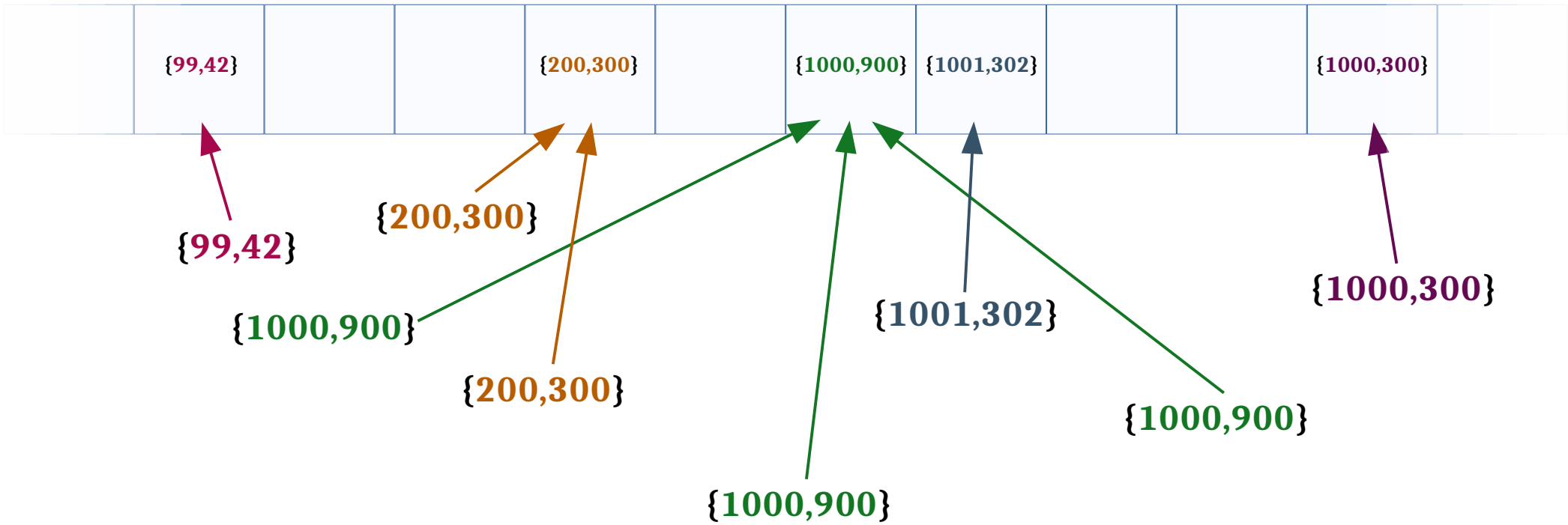


Recorders (Multi-Value SDEs)

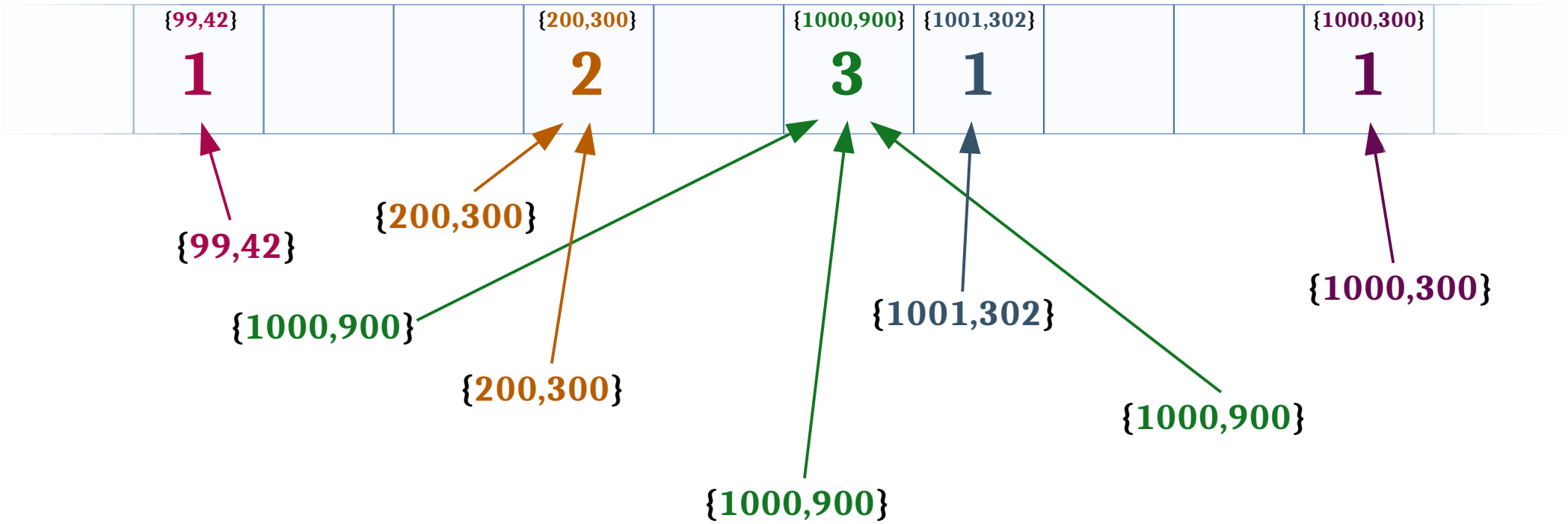
The library keeps track of a series of events.



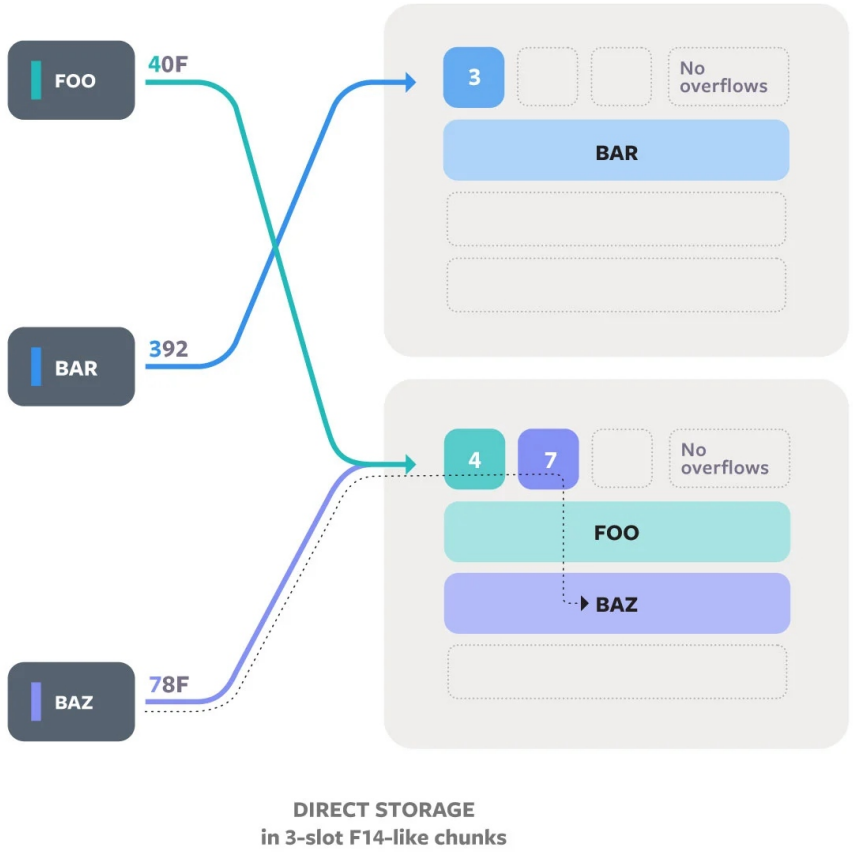
Set data-structure



CountingSet: set that counts duplicates



Hash table implementation inspired by FB's F14



https://engineering.fb.com/2019/04/25/developer-tools/f14/

Birthday paradox on buckets

- Add 85 elements in hash table with capacity 5110:
- P(collision) = 1-(5109/5110)^(85*84/2) = 0.502

Birthday paradox on buckets

- Add 85 elements in hash table with capacity 5110:
- P(collision) = 1-(5109/5110)^(85*84/2) = 0.502
- HT: 365 buckets with capacity of 14, how many insertions to get 15 collisions?

365*14 = 5110

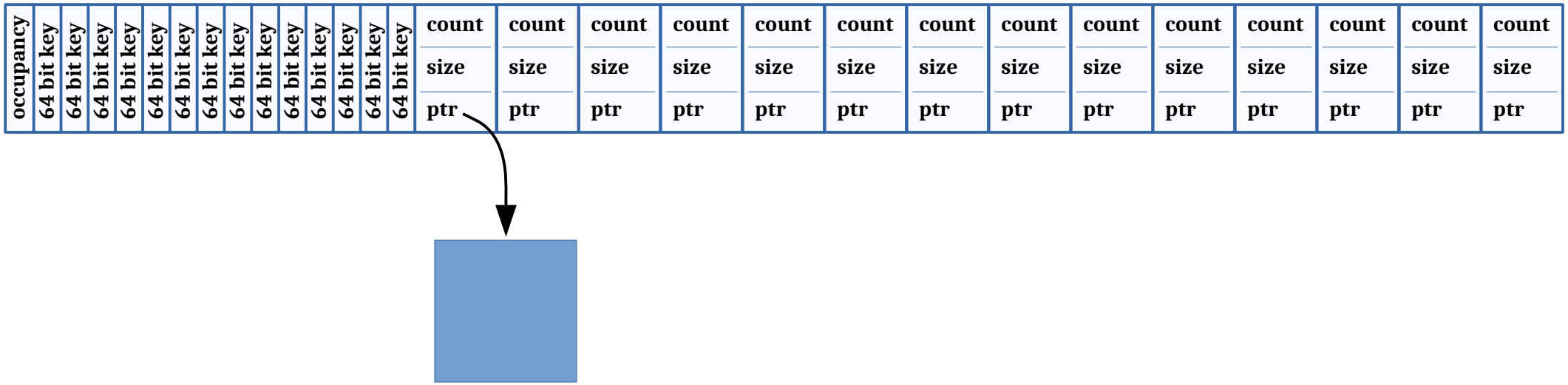
Birthday paradox on buckets

- Add 85 elements in hash table with capacity 5110:
- P(collision) = 1-(5109/5110)^(85*84/2) = 0.502
- HT: 365 buckets with capacity of 14, how many insertions to get 15 collisions?
- N = 2263 for P>50%

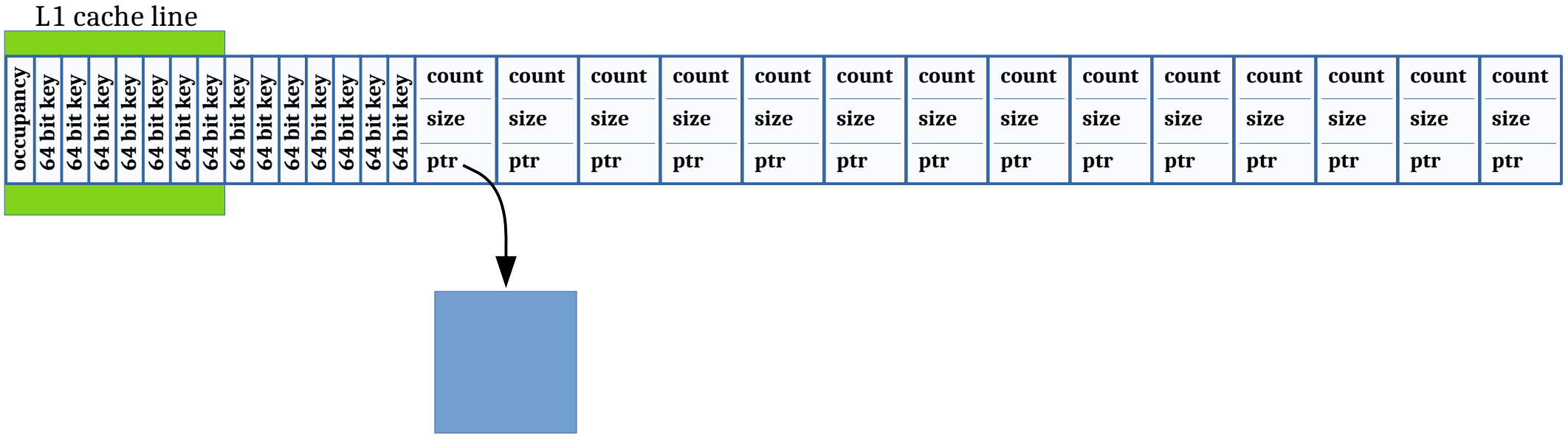
Birthday paradox on buckets

- Add 85 elements in hash table with capacity 5110:
- P(collision) = 1-(5109/5110)^(85*84/2) = 0.502
- HT: 365 buckets with capacity of 14, how many insertions to get 15 collisions?
- N = 2263 for P>50%

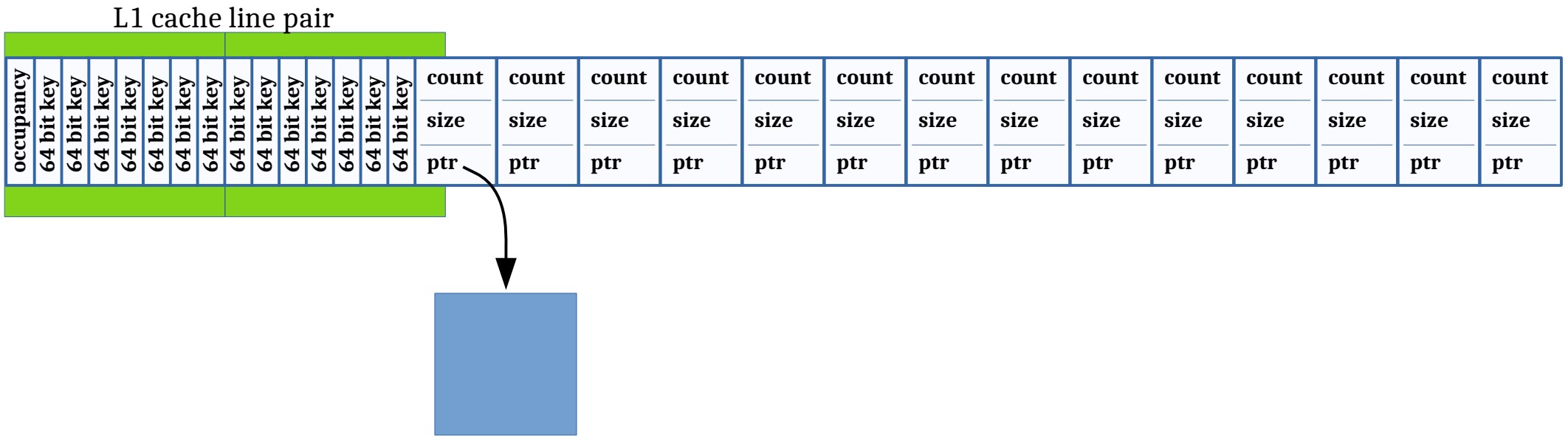
Actual bucket (sde_hash_bucket_t)



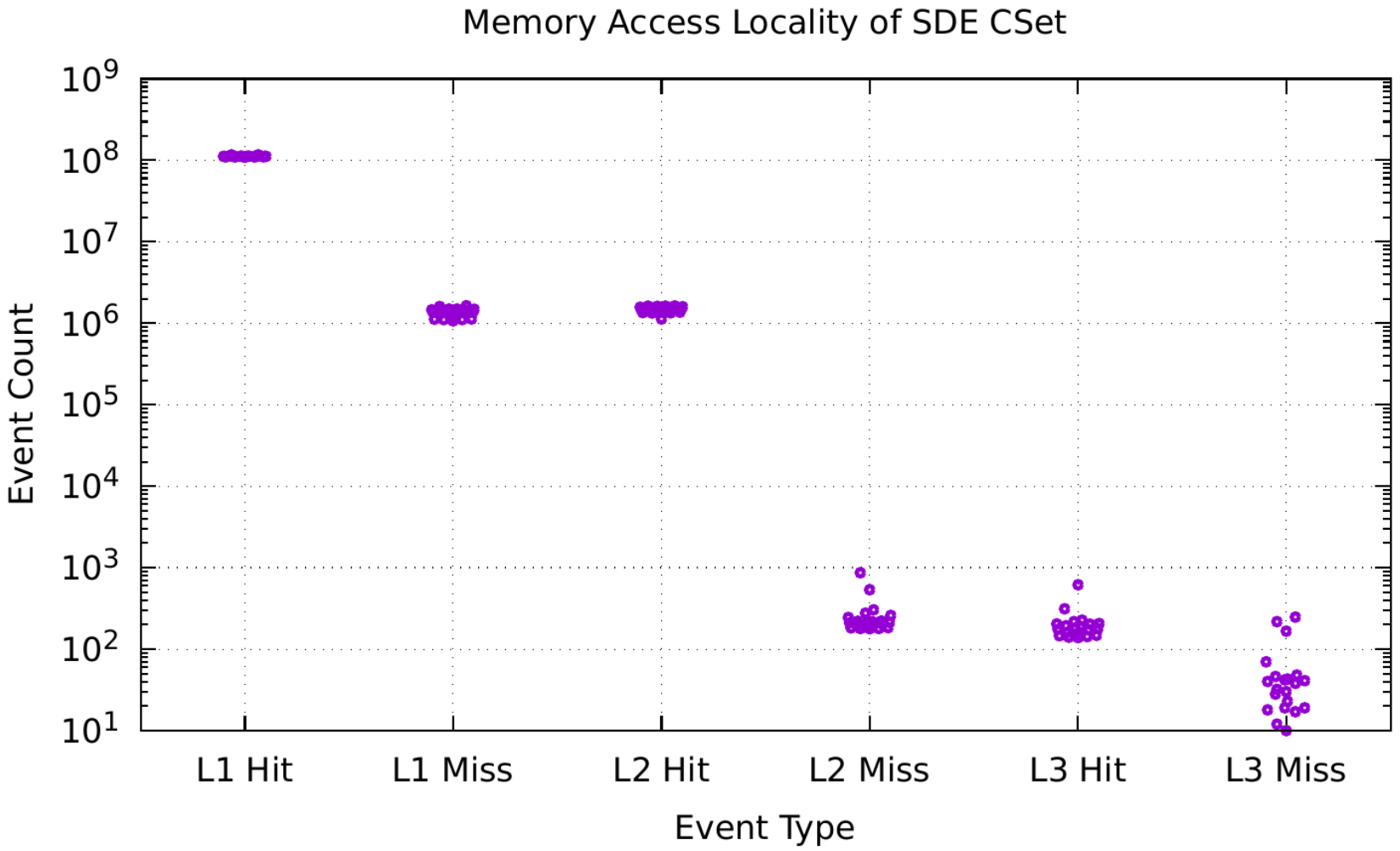
Actual bucket (sde_hash_bucket_t)



Actual bucket (sde_hash_bucket_t)



The proof is in the pudding.



Event Description Ambiguities

What does this event really measure?

LONGEST_LAT_CACHE:REFERENCES

Core-originated cacheable requests that refer to L3 (Except hardware prefetches to the L3).

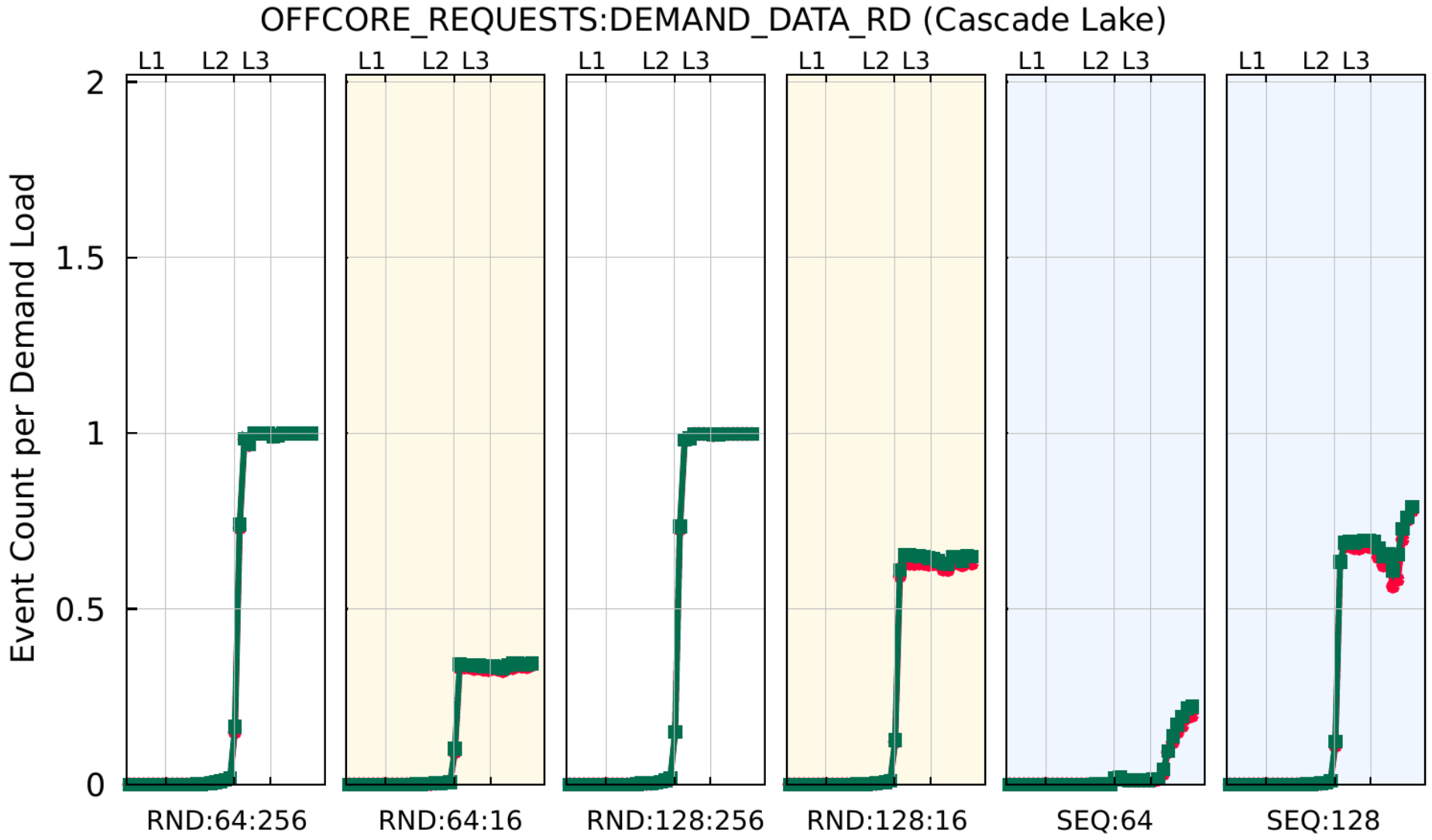
What does this event really measure?

LONGEST_LAT_CACHE:REFERENCES

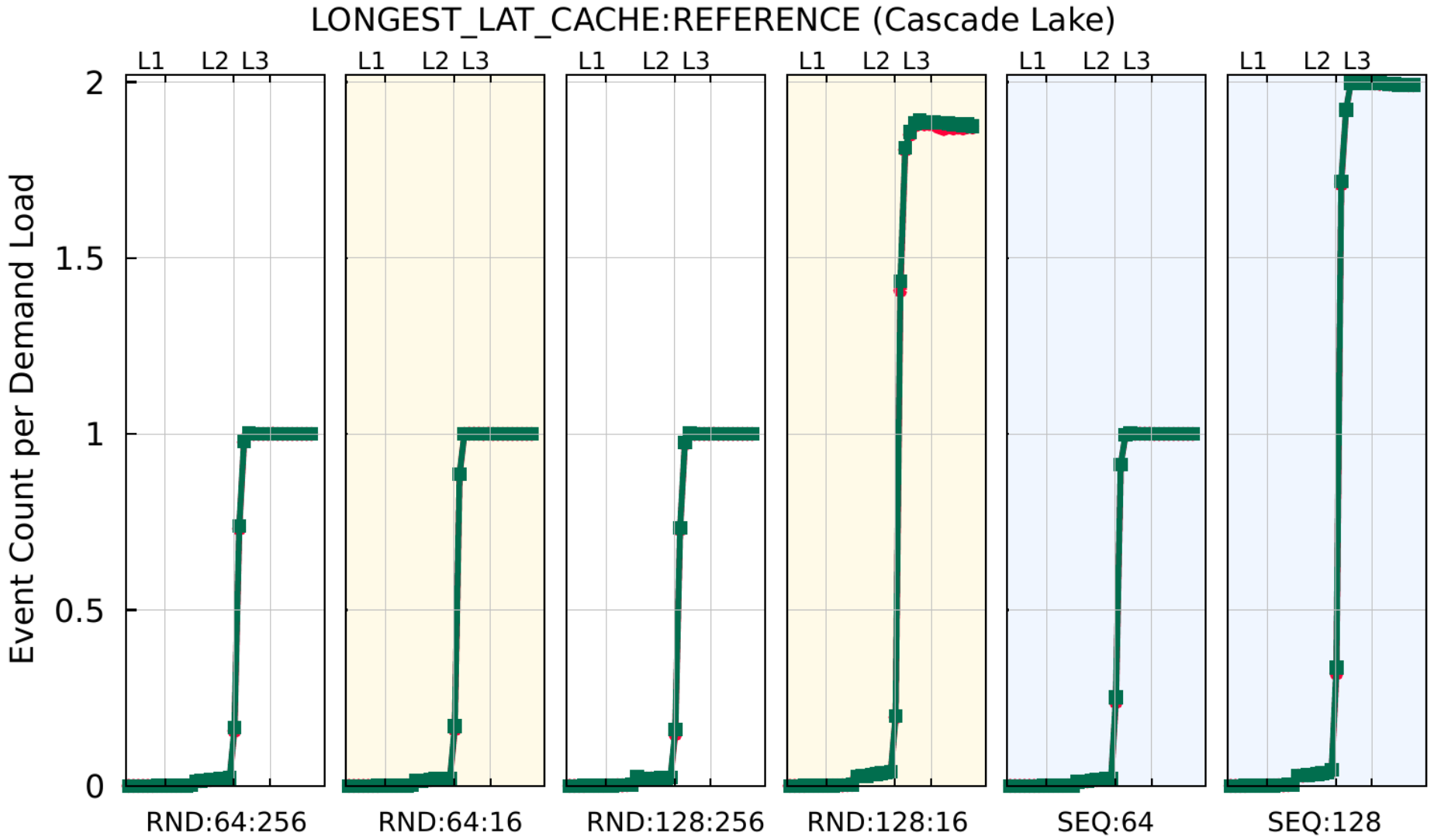
Core-originated cacheable requests that refer to L3 (Except hardware prefetches to the L3).

Is “Except hardware prefetches” the same as “Demand References”?

Demand References on Cascade Lake



LLC References on Cascade Lake



Summary

- PAPI 7.0.1 is out (https://github.com/icl-utk-edu/papi).
- Support for GPU counters/metrics across vendors.
- Support for power management on CPUs & GPUs.
- API & utility for detecting available hardware.
- Software Defined Events as a standalone library (libsde.so).
- Counter Analysis Toolkit provides hardware insights.
- Noise is not due to hardware counters, but system operations.

Backup Slides

Simple Buffer Traversal (C code)

```c
int main(int argc, char **argv){
    int M, N;
#define BUFFER_SIZE 256
    int buffer[BUFFER_SIZE];

    M = 1024;
    N = 1024;

    for(int i=0; i<BUFFER_SIZE; i++){
        buffer[i] = (argc*1222)/(1223+i);
    }

    int index = buffer[0];
    for(int i=0; i<M; i++){
        for(int j=0; j<N; j++){
            index = buffer[index%BUFFER_SIZE];
        }
    }

    printf("index=%d\n",index);

    return 0;
}
```

Simple Buffer Traversal (C code)

```c
int main(int argc, char **argv){
    int M, N;
#define BUFFER_SIZE 256
    int buffer[BUFFER_SIZE];

    M = 1024;
    N = 1024;

    for(int i=0; i<BUFFER_SIZE; i++){
        buffer[i] = (argc*1222)/(1223+i);
    }

    int index = buffer[0];
    for(int i=0; i<M; i++){
        for(int j=0; j<N; j++){
            index = buffer[index%BUFFER_SIZE];
        }
    }

    printf("index=%d\n",index);

    return 0;
}
```

```c
int main(int argc, char **argv){
    int M, N;
#define BUFFER_SIZE 256
    int buffer[BUFFER_SIZE];

    M = 1024;
    N = 1024;

    for(int i=0; i<BUFFER_SIZE; i++){
        buffer[i] = (argc*1222)/(1223+i);
    }

//    int index = buffer[0];
    uintptr_t index = buffer[0];
    for(int i=0; i<M; i++){
        for(int j=0; j<N; j++){
            index = buffer[index%BUFFER_SIZE];
        }
    }

    printf("index=%d\n",index);

    return 0;
}
```

Simple Buffer Traversal (asm code)

```
int index = buffer[0];
)00000000010cc <+76>:    mov    (%rsp),%edx
)00000000010cf <+79>:    mov    $0x400,%esi
)00000000010d4 <+84>:    nopl   0x0(%rax)

for(int i=0; i<M; i++){
)00000000010fa <+122>:   sub    $0x1,%esi
)00000000010fd <+125>:   jne    0x10d8 <main+88>

    for(int j=0; j<N; j++){
)00000000010f5 <+117>:   sub    $0x1,%ecx
)00000000010f8 <+120>:   jne    0x10e0 <main+96>

        index = buffer[index%BUFFER_SIZE];
)00000000010e0 <+96>:    mov    %edx,%eax
)00000000010e2 <+98>:    sar    $0x1f,%eax
)00000000010e5 <+101>:   shr    $0x18,%eax
)00000000010e8 <+104>:   add    %eax,%edx
)00000000010ea <+106>:   movzbl %dl,%edx
)00000000010ed <+109>:   sub    %eax,%edx
)00000000010ef <+111>:   movslq %edx,%rdx
)00000000010f2 <+114>:   mov    (%rsp,%rdx,4),%edx

    }
}
```

Simple Buffer Traversal (asm code)

```
int index = buffer[0];
0000000010cc <+76>:    mov    (%rsp),%edx
0000000010cf <+79>:    mov    $0x400,%esi
0000000010d4 <+84>:    nopl   0x0(%rax)

for(int i=0; i<M; i++){
0000000010fa <+122>:   sub    $0x1,%esi
0000000010fd <+125>:   jne    0x10d8 <main+88>

    for(int j=0; j<N; j++){
0000000010f5 <+117>:   sub    $0x1,%ecx
0000000010f8 <+120>:   jne    0x10e0 <main+96>

        index = buffer[index%BUFFER_SIZE];
0000000010e0 <+96>:    mov    %edx,%eax
0000000010e2 <+98>:    sar    $0x1f,%eax
0000000010e5 <+101>:   shr    $0x18,%eax
0000000010e8 <+104>:   add    %eax,%edx
0000000010ea <+106>:   movzbl %dl,%edx
0000000010ed <+109>:   sub    %eax,%edx
0000000010ef <+111>:   movslq %edx,%rdx
0000000010f2 <+114>:   mov    (%rsp,%rdx,4),%edx

    }
}
```

```
uintptr_t index = buffer[0];
0000000010cc <+76>:    movslq (%rsp),%rdx
0000000010d0 <+80>:    mov    $0x400,%ecx
0000000010d5 <+85>:    nopl   (%rax)

for(int i=0; i<M; i++){
0000000010ec <+108>:   sub    $0x1,%ecx
0000000010ef <+111>:   jne    0x10d8 <main+88>

    for(int j=0; j<N; j++){
0000000010e7 <+103>:   sub    $0x1,%eax
0000000010ea <+106>:   jne    0x10e0 <main+96>

        index = buffer[index%BUFFER_SIZE];
0000000010e0 <+96>:    movzbl %dl,%edx
0000000010e3 <+99>:    movslq (%rsp,%rdx,4),%rdx

    }
}
```