PMU Performance Measurement

Participants: Alfredo Gimenez, Ben Woodard, Rashawn Knapp, Stephane Eranian, Xu Liu, Yukon Maruyama

- Data-Centric Profiling, where are we?
- Can high-level HWC abstractions help users?
- Q/A with Stephane

Data-Centric Profiling, where are we?

- Survey of strategies for **attribution of samples to data**

  a. **Self -profiling that captures of data reference addresses**. Captures PC and data address for each sample (Intel PEBS, AMD IBS, IBM Marked Events, SPARC...)
  Metric attribution to callstack and source:

foo.c, line 3: `foo.myInteger++;`	10000234 reads

  Plus metric attribution to memory address hierarchy (VAddress, cache line, etc.):

data address: `0x12342515`	5023 reads

  (May include data type information for Global variables)

  b. Self-profiling to capture data reference addresses plus **Tracing all heap allocations (via interposition)**. Example: Xu Liu's presentation. Associates data access with original allocation stack, and location within allocation. Can show which region of allocation is accessed where (call site, thread, etc). Does not attribute metrics to data type. (See Xu's slides.)

data address: `0x12342515`	**callpath**: `foo->bar->malloc(512)`

  c. Self-profiling to capture data reference addresses plus **compiling binaries to include compiler-generated descriptions of data structures accessed at each PC**. Example: Oracle Studio compiler. Adds ability to attribute metrics to **Data Structure fields**. (Also see Hot-cold structure splitting.) Example:

`data structure`	read count
`struct foo{`	1000010 (total for struct)
`  int myInteger;`	1000000
`  char myChar;`	10
`};`	

  d. **Static analysis** based on binary and DWARF to determine data types. Examples Napa (Matt Legendre).

`double *d =`
`  (double*)malloc(sizeof(double)*4096);`
	Trace return address of malloc to symbol **d**, type **double ptr**

  e. **Source-Level Annotation** Example Mitos (Alfredo Gimenez)

`double *d =`
`  (double*)malloc(sizeof(double)*4096);`	

`Mitos_add_symbol(“d”,d,sizeof(double),4096);`	Data symbol, data type, access index (*i* in array[*i*])

- Wish-list for improving Data-Space Profiling
    a. gcc: update compiler to emit memory reference’s data structure and offset
        - Should we refresh last year’s DWARF wish-list?
        - Ben Woodard is interested in hearing specific recommendations for DWARF
        - Will enough users benefit to justify the work?

    b. Handle workloads with frequent allocations
        - May rule out tracing

    c. “Attach” to a running process
        - May rule out tracing

    d. Attribute metric to data structure fields even if field is accessed via an intermediate pointer or wrapper.

Example 1:

```
int wrapper(int* ptr){
  return *ptr;
}
int x = foo(&mystruct.myInteger);
```

Want to attribute access to “mystruct.myInteger”, not int*

Example 2:

```
class q {
    int *a;
    double *b;
    q(size_t asz, size_t bsz) {
        a = malloc(asz);
        b = malloc(bsz);
    }
}
load q->b[4]
```

Want to attribute access to “class q”, not “b”

- May require compiler support *and* tracing of allocations.

Can high-level HWC abstractions help users?

- Common problems
    a. Which raw hardware counters are needed to answer high-level questions?
        - What kinds of abstractions might help?

    b. Some high-level metrics cannot be measured directly (require derived metrics)
        - Could abstractions prod HW vendors support more direct measurement?
        - John M-C suggested organizing a meeting between Vendors and Tools Developers
            1. Rob Fowler might be a good contact.
- Analyzing at the process level
    a. Intel “Top Down” decision tree may be an OK abstraction. (See Intel website for diagram with ‘UopAllocate?’ at root, and ‘BackEndStall?’ and ‘UopEverRetire?’ as children)

    b. Pros:
        - Generic enough at the top level to apply to other vendors?
        - Intel provides specific counters
        - Hopefully encourages Intel to provide direct counters that support the methodology

    c. Cons:
        - Some measurements are derived from multiple counters so can’t be used for direct profiling
        - Not every decision box deals with the same unit (uops vs cycles…)
        - May undercount/overcount some stuff (D. Levinthal)
        - Portions of tree and HWCs are not cross-architecture

- Analyzing at the ‘uncore’ level
    a. Example motivating scenarios
        - See if close to Peak Utilization for some metric
        - Check for NUMA-like imbalances

    b. High-level abstractions of interest (not complete…)
        - I/O bandwidth
        - intra-socket bandwidth (of what?)
        - power
        - local vs remote (memory?) traffic
        - PCIe traffic
        - Indications of saturation (queue wait times, queue depths, high-water marks)
        - C-state
        - Distinguishing I/O vs. Memory Traffic.
        - Any others?

    c. Implementing the abstraction layer: who should do it?
        - OS (kernel-space)
            1. pros
                a. user needs no architecture-specific knowledge
                b. maintained list of metrics
            2. cons
                a. opaque mapping == not trustworthy? Example: ‘cycles’. Does this mean unhalted cycles? unscaled cycles?
                b. adoption of OS update not always possible (backporting chaos)
        - Tools (user-space)
            1. pros
                a. quicker support
            2. cons

a. tool developer may not have broad enough focus

Q/A with Stephane

- Q: Intel HWC bugs we should know about?
    a. Hyperthreading causes corrupted counts
        - Fixed in Broadwell
        - Workaround for Pre-Broadwell in Linux 4.1
        - On affected systems, might be ok if activate only one counter
        - PEBS Load Latency w/ threshold IS not be affected by this bug
    b. PEBS “shadow”: wrong instruction pointer tagged
        - Affects SandyBridge, IvyBridge, improved with actual “precise ip” in Haswell
    c. **Broadwell** has erratum on instructions_retired sampling period, bottom 6 bits must be zero, kernel handles it by masking off those bits
- Q: How does Linux extend HWC 40-bit counters to 64-bits when there is no interrupt capability?
    a. A: polling the counter with timer
- Q: Who’s working on Linux HWCs these days?
    a. <answer offline>