6.5810: Nsight

Adam Belay <abelay@mit.edu>

MIT CSAIL

Motivation

- Many developers are building systems that require low latency
- But no tools exist to systematically diagnose latency problems
- Existing statistical profilers (e.g., perf-tool) are great at attributing CPU use, but can't measure latency

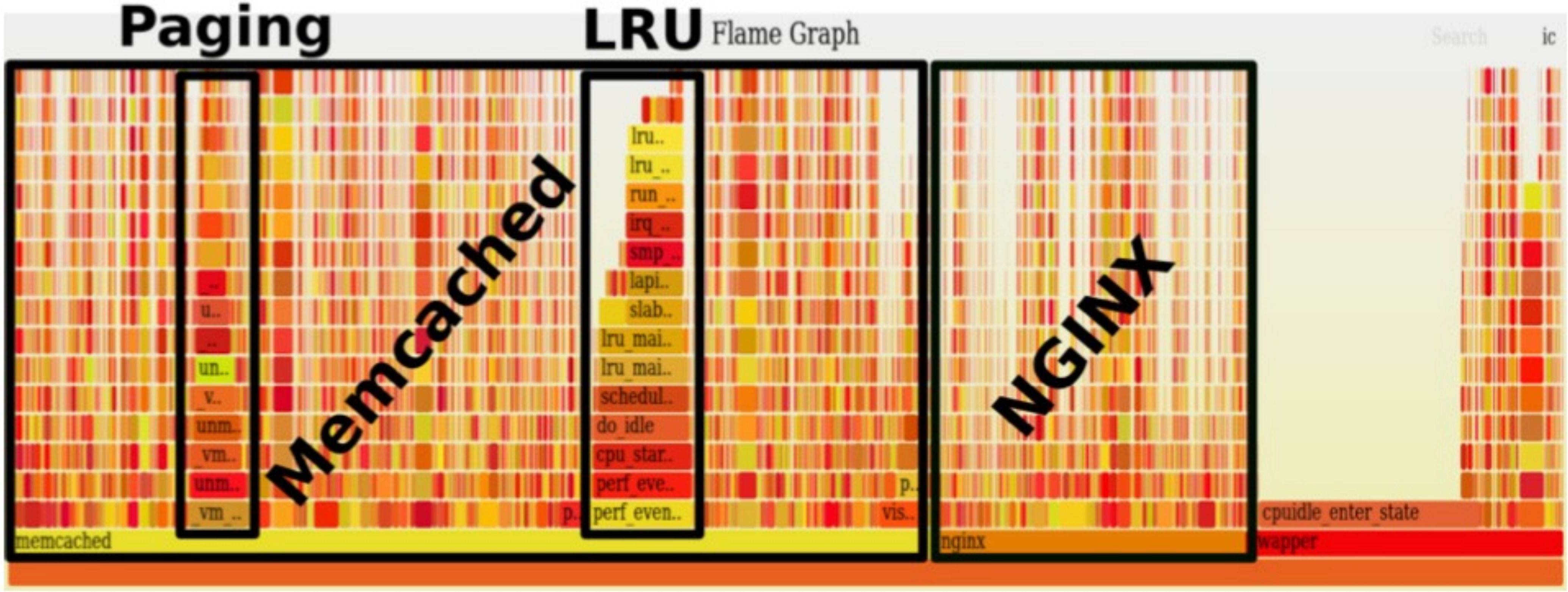
Problems with existing tools

1. Some latency deviations are caused by the NIC; not in software
2. Existing tools have high overhead (often called time dilation)
3. Difficult to trace everything; tools focus on specific parts

What is Nsight?

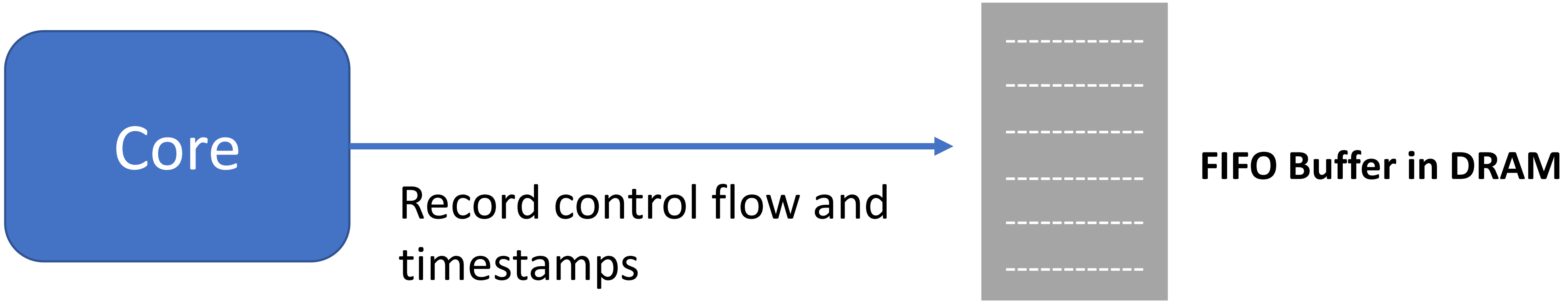
- Track the entire lifetime of network requests and identify the precise reasons for latency deviations
- Networking side: NIC timestamps
- CPU side: CPU profiling (Intel-PT)

Problem with perf: Can’t see deviations….



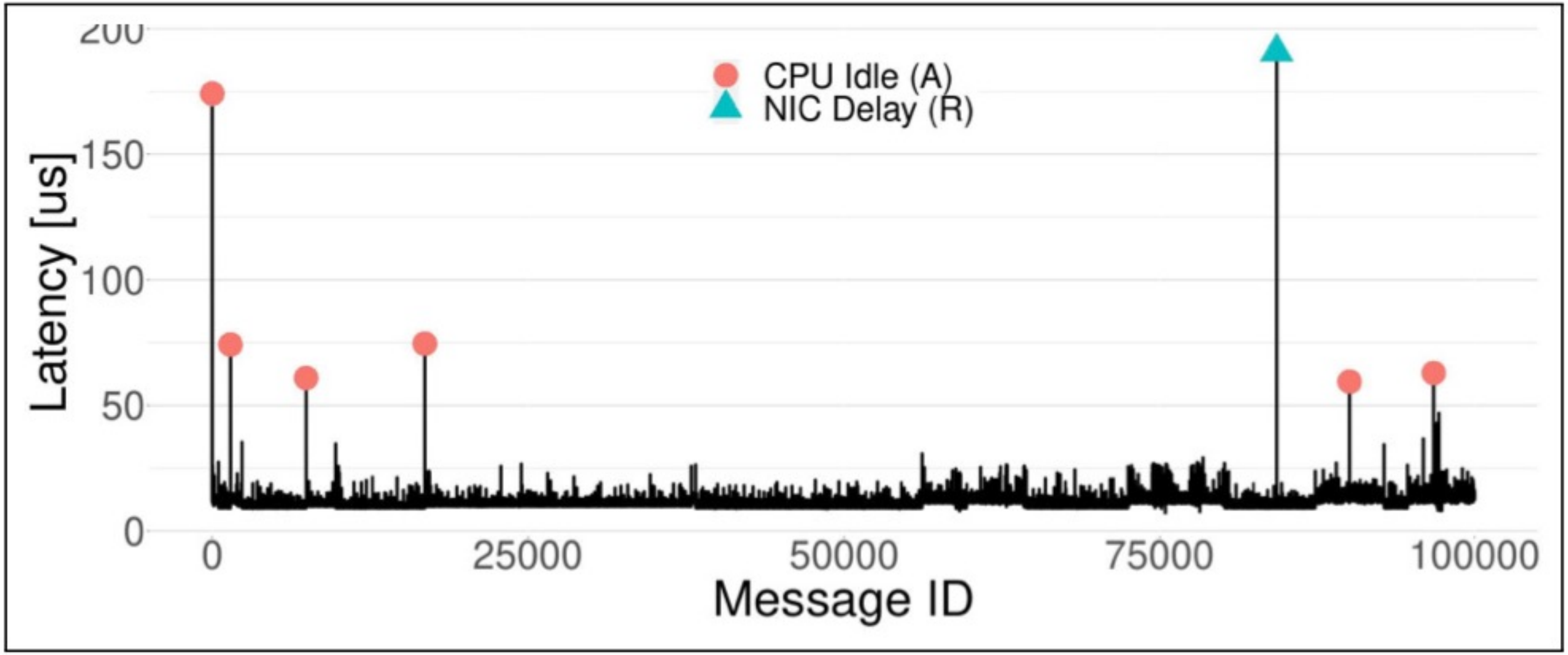
Quick aside: What is Intel PT?

- Intel Processor Trace; a brand-new hardware feature this year
- Does not require any source code modification
- Monitors instruction stream of a core…
  - Whenever a branch instruction is encountered (e.g., call, ret, jmp, je, etc.)…
  - Records a record to a FIFO in DRAM; highly compressed format

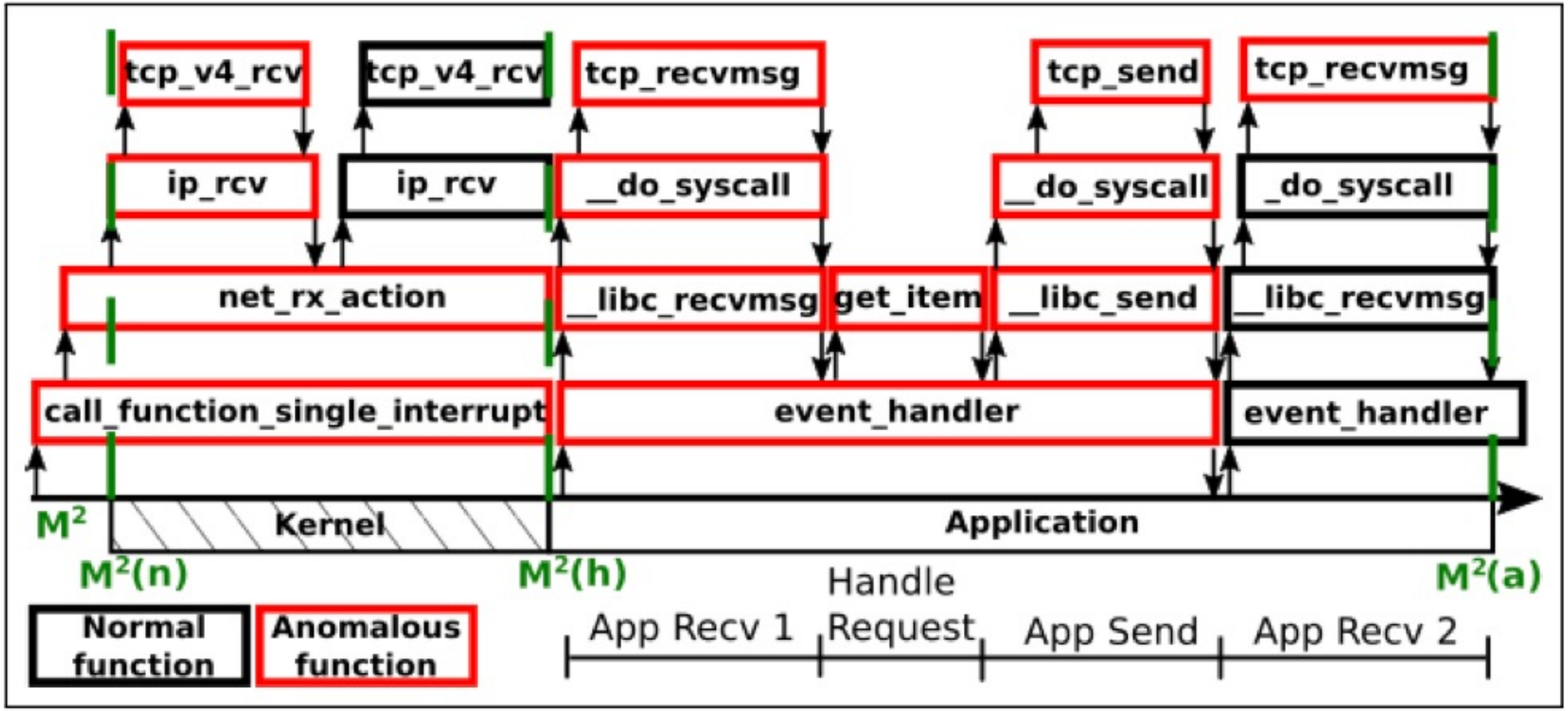


Q: Does Intel-PT have overhead?

Using Nsight: Request-level view



Using Nsight: Message lifetime



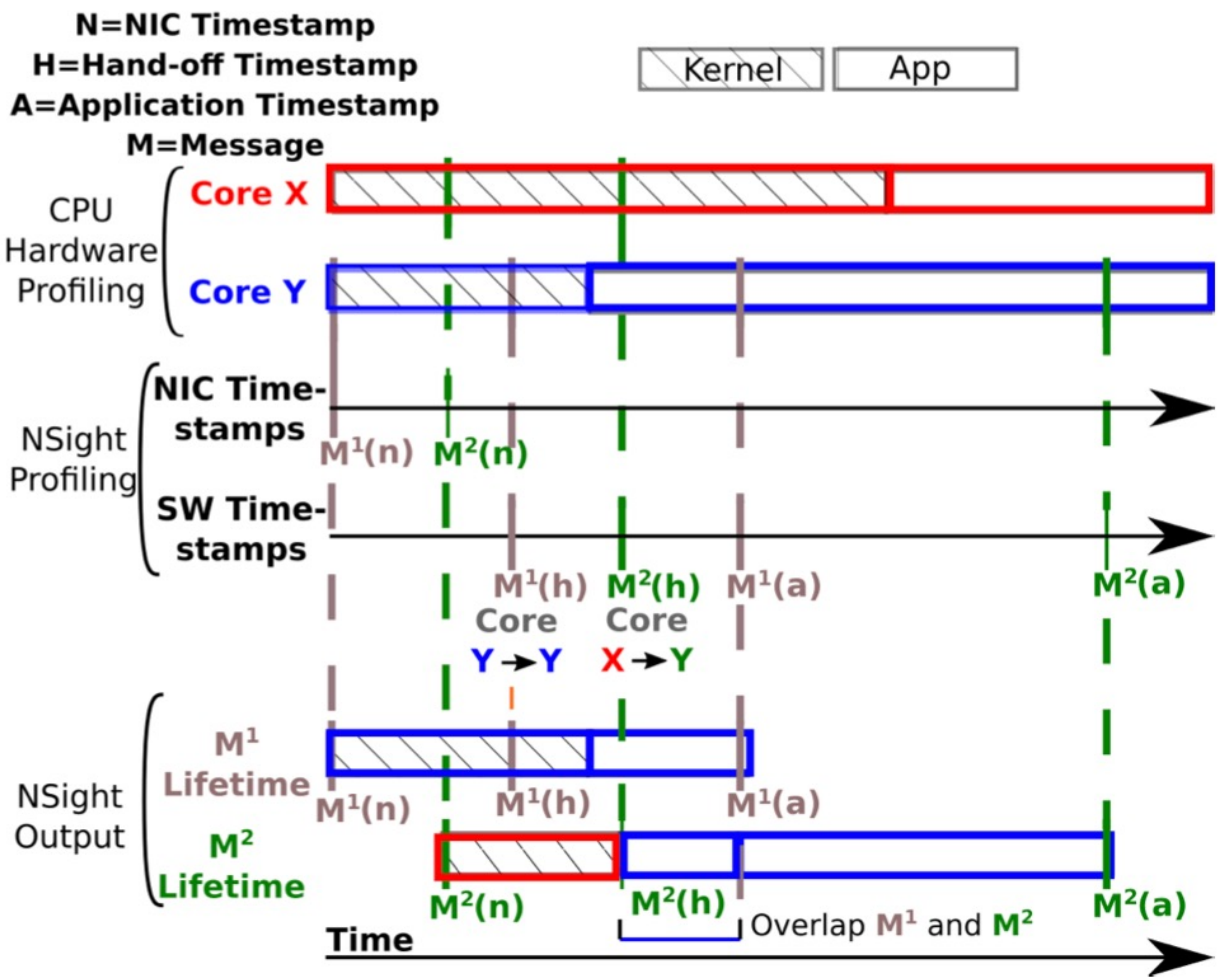
Challenge #1: Clocks can be adjusted

- Synchronizing clocks on different devices is hard in general
- Nsight needs to respond to clock adjustments from NTP
- Solution: Nsight recallibrates whenever time is adjusted
- An entire core is burned to poll for changes to the clock
- Q: Why not use CLOCK_MONOTONIC_RAW instead?

Challenge #2: How to track messages?

- Easy to time packets but hard to integrate timing with host stack
- Solution: NSight records timings of various entry/exit points
- NIC timestamp (N); core handoff (H); application handoff (A)
- Sufficient to produce a single timeline of each message
- Relies on a kernel boundary (i.e., a function) that is called when processing is done and handoff to application will happen

Challenge #2: How to track messages



Q: What is a message?

- Are packets messages?
- How are messages tagged?

Challenge #3: Nested function calls

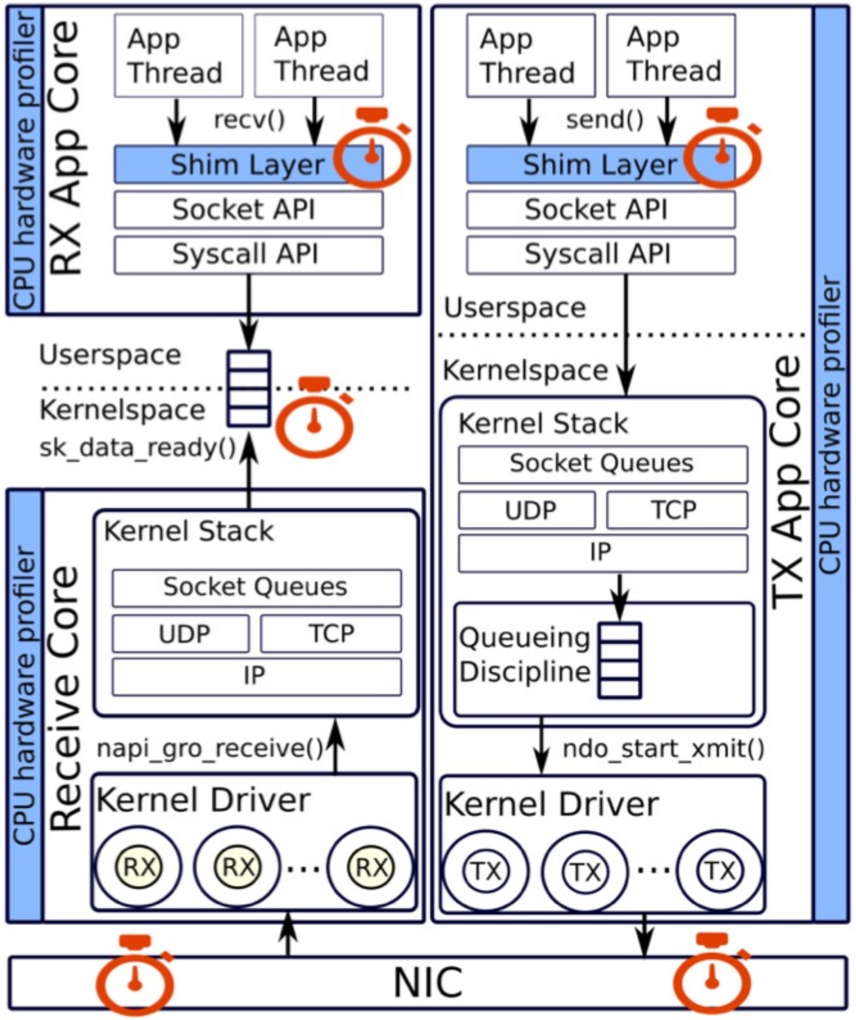
- Need to identify functions as the root cause (on the CPU side)
- If the *callee* takes longer than normal, so will the *caller*
- Solution: If nested calls take more than 80% of the time of the parent, don’t flag the parent as an anomaly, only the children

Nsight components

1. CPU hardware profiler
   - Use Intel-PT to record and timestamp all branch instructions
2. Shim layer
   - LD_PRELOAD all socket system calls to collect entry to app timestamp
3. NIC timestamps
   - Record packet arrivals into the system

*Also, a dedicated core polls for time config changes

Nsight components



Limitation: Cannot be used continuously

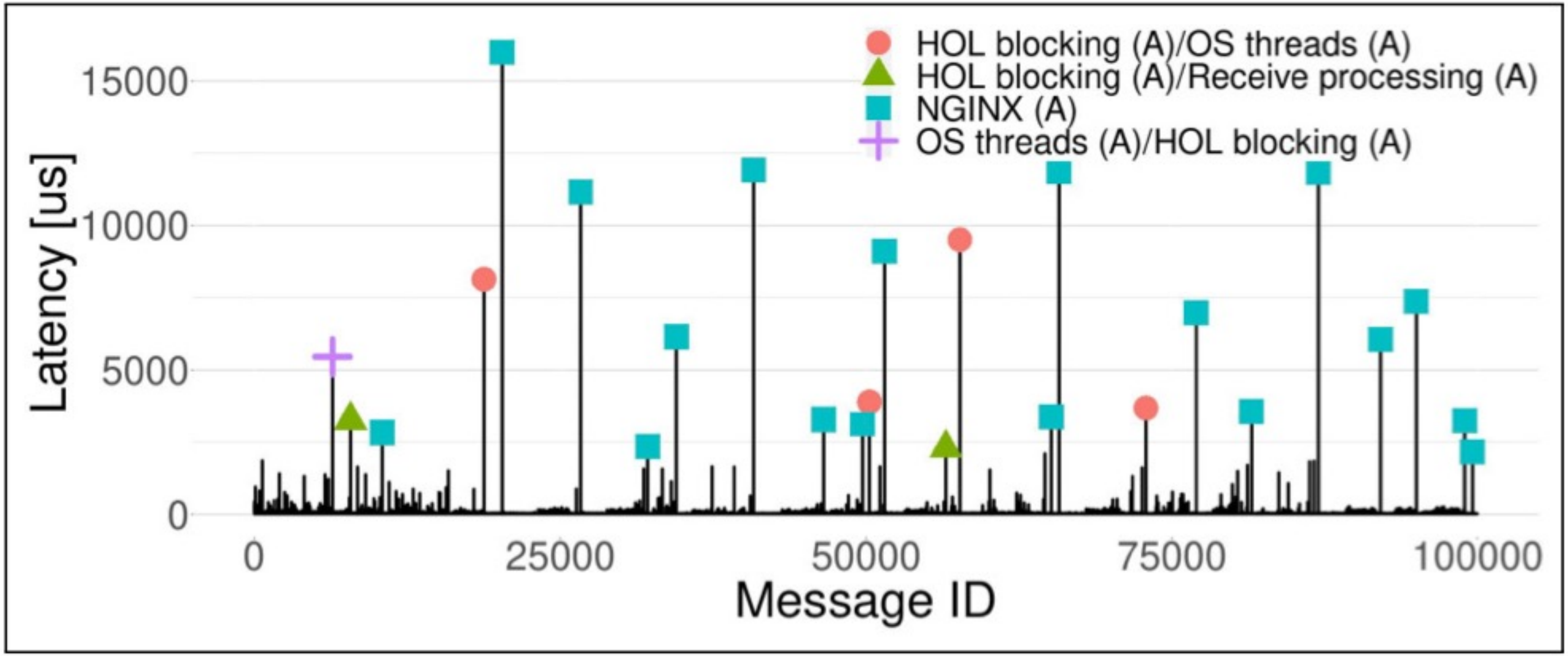
- To capture random events across time, users must turn on NSight repeatedly. We have ambitions of using NSight for continuous profiling, but the current buffering implementation in Intel-PT limits such use.
- Why is this?
- Does it matter?

How high is profiling overhead?

All numbers in μs. h = high load, l = low load				
Tool	median (h)	99.9th (h)	median (l)	99.9th (l)
Baseline	30.3	112.6	10	14.4
Intel-PT	30.8 (2%)	120.4 (7%)	10.6 (5%)	15.3 (6%)
NSight	31.1 (3%)	132.8 (18%)	11 (10%)	16.2 (12%)
eBPF-1	38.6 (27%)	157.6 (40%)	11.8 (18%)	17.3 (20%)
eBPF-2	41.8 (38%)	165 (46%)	13.2 (31%)	18.6 (29%)
eBPF-4	51.9 (71%)	556 (393%)	14.1 (41%)	19.4 (35%)
eBPF-8	59.1 (95%)	565 (402%)	15.5 (54%)	21 (45%)
Ftrace	201.8 (565%)	1060 (841%)	40.1 (298%)	66.4 (359%)

Memcached latency in microseconds

Can Nsight identify root causes?



Discoveries from using Nsight in diff. configs

1. Memcached can introduce high latencies by batching many requests together on each thread
   - Solution: Expose more request parallelism
2. NUMA page migration causes latency spikes
   - Solution: Unclear
3. Connection set up hands of TCP socket to different thread
   - Thread wakeup causes occasionally large delays
   - Solution: Need a better CPU scheduler
4. Core pinning can cause CPU overload
   - Solution: Don’t pin, and need a better CPU scheduler

Limitations

- Dedicated core for polling time configuration changes
- Cannot be used for continuous profiling
- Analysis time is dominated by huge dataset produced by Intel PT
- Cannot capture small (nanosecond) timescale events
  - Does this matter?