Software Access to Precise Time for Industrial, Automotive, A/V[, Etc.]

Requirements and Approach

Kevin B. Stanton, Ph.D.
Sr. Principal Engineer
Intel Corporation

kevin.b.stanton@intel.com

Abstract: Synchronization needs of Industrial, Automotive, and A/V differ substantially from those of telecommunications. Herein we describe some of their characteristic challenges and approaches, and describe their relationship to computation.

Need for Precision Timekeeping is Broad

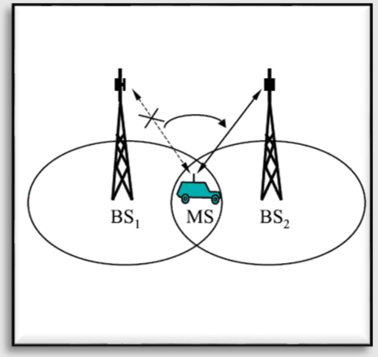
Automotive

Conferencing

Realtime A/V

Industrial/ Energy

Hard Deadlines



Cellular/Telco

Financial

Cloud/HPC

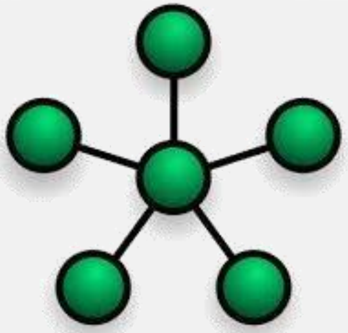
Time Sync is enough

Some Apps Require UTC Traceability
…Some Do Not

TSN (Time Sensitive Networking)
TCC (Time Coordinated Computing)

Time Sync

Timeliness



TSN

Between Systems

PTP

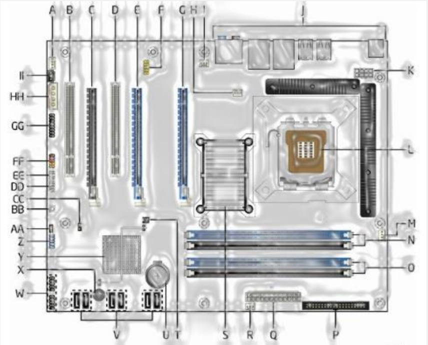
IEEE 802.1AS profile of IEEE 1588

"TSN"

IEEE 802.1Q

IETF DetNet

Latency Guarantees, Seamless Redundancy, Ingress Policing, Ethernet Preemption, etc.



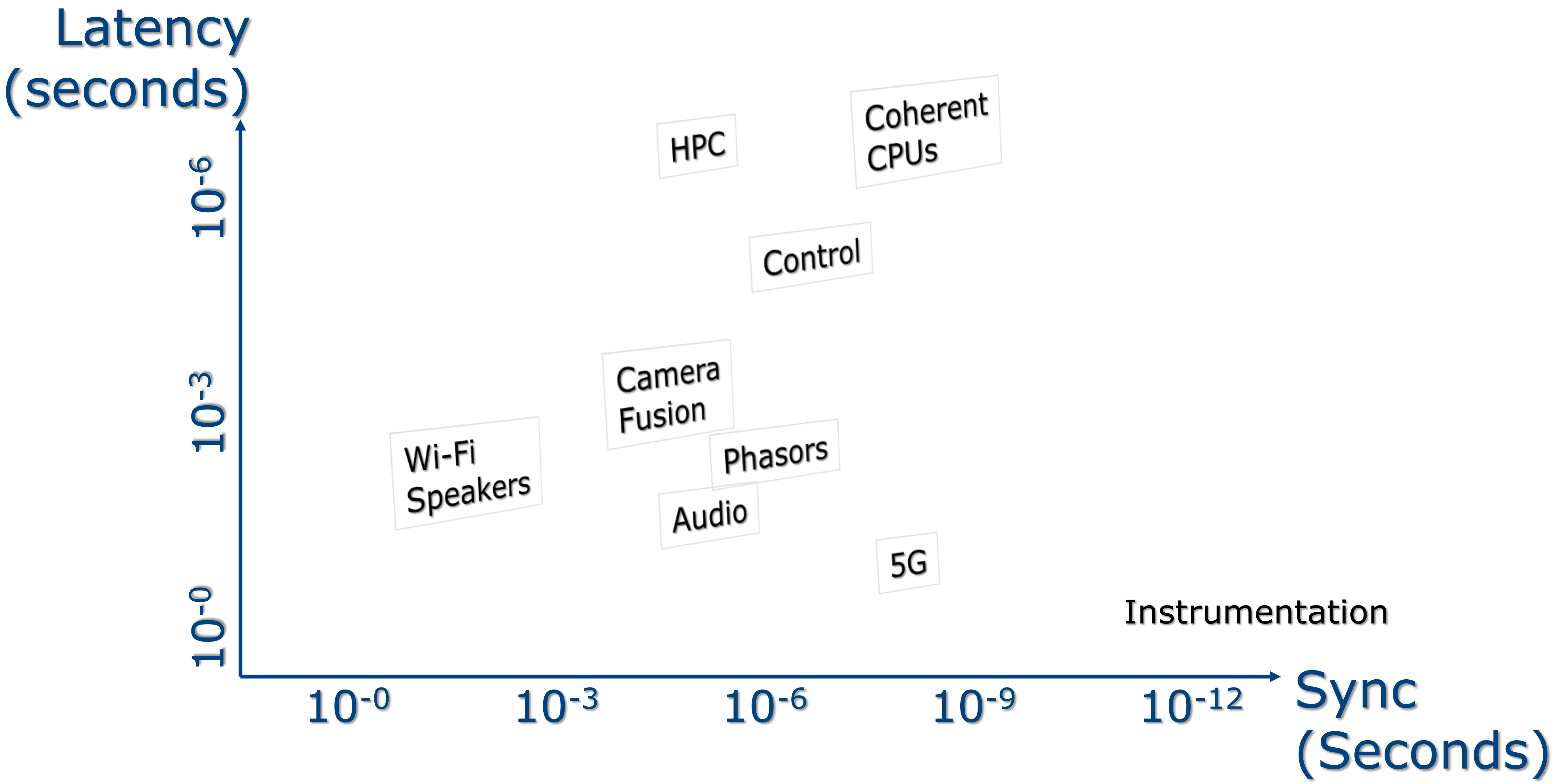
TCC

Within The System

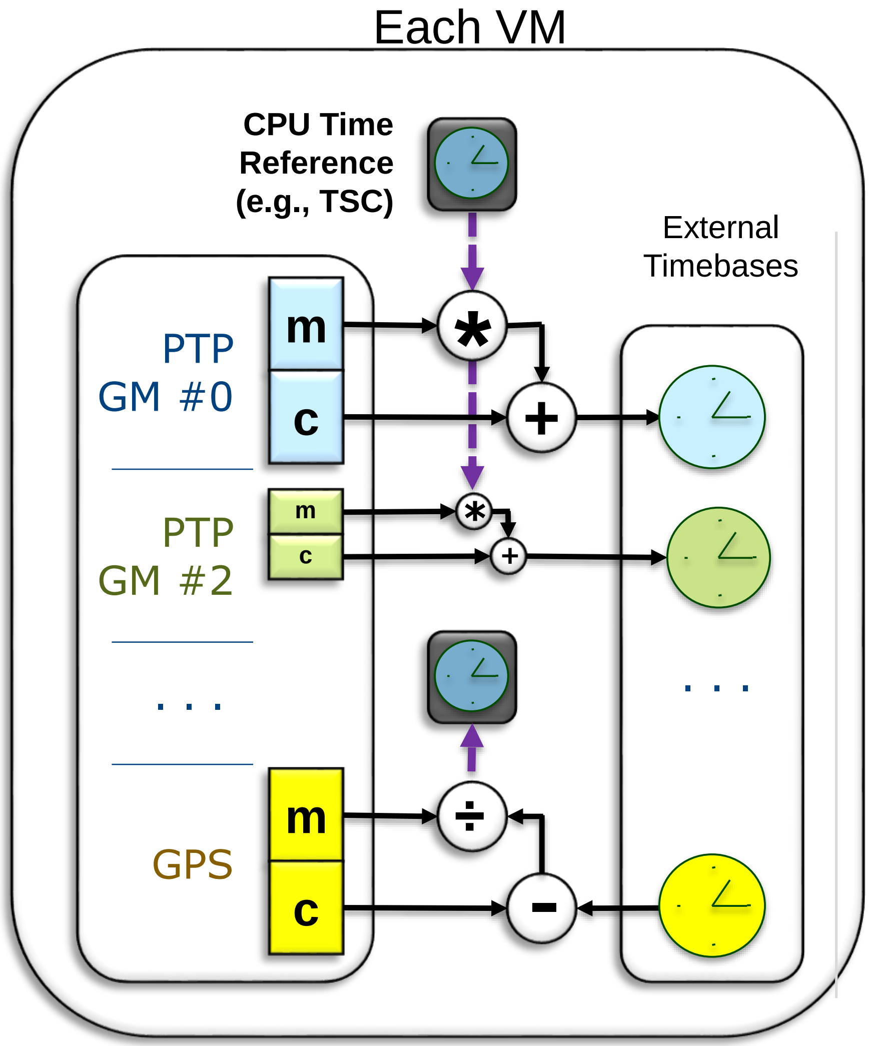
System Timekeeping

Traditional Realtime

Application Requirements



Scalable Timebase Representation

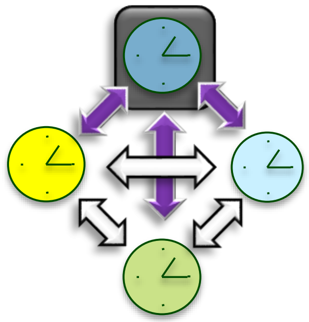


Linear transformation between CPU time and other arbitrary time via y=mx+c

Here's what's needed:

1. A Stable HW Reference
2. Fast * and + Ops
3. Precise estimate of m and c

➔ Any Timebase to/from Any Timebase



Timescale Translation Scales Well

Immediate Software access to "Now" in Linux

Time "now"

(1) `clock_gettime(CLOCK_MONOTONIC_RAW, &now);`

- Returns current TSC value scaled to nominal nanoseconds

(2) `clock_gettime(CLOCK_MONOTONIC,     &now);`

- Returns current TSC value scaled to track TAI, in nanoseconds

(3) `clock_gettime(CLOCK_REALTIME,      &now);`

- Returns CLOCK_MONOTONIC + (now-1/1/1970) [incl. leap seconds]

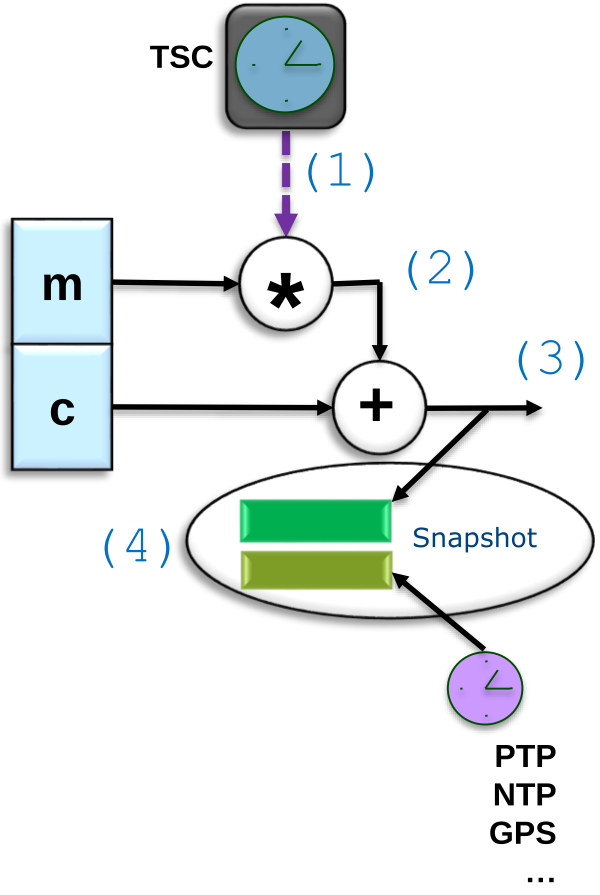
Cross-Timestamp

(4) `ioctl(phc_fd,PTP_SYS_OFFSET[_PRECISE], &offset )`

- returns the triple:
  - `eth_ptp_time; realtime; monotonic_raw`

Event Timestamp / Event Scheduling

- Device Specific



POSIX: Piecewise-Linear Clock Model: Y[n]=mx[n]+c[n]

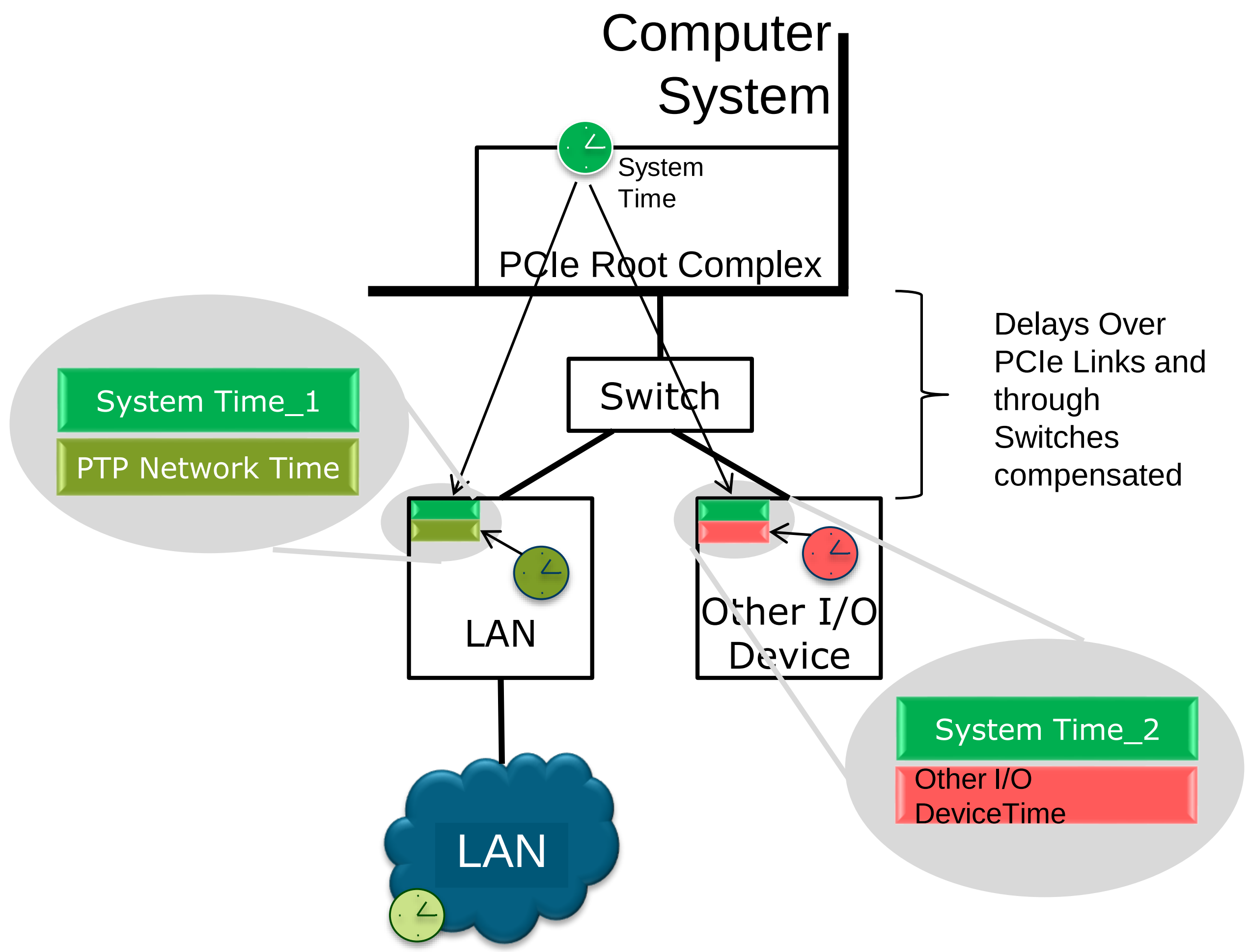
Measuring PTP vs. System Time using PCIe PTM

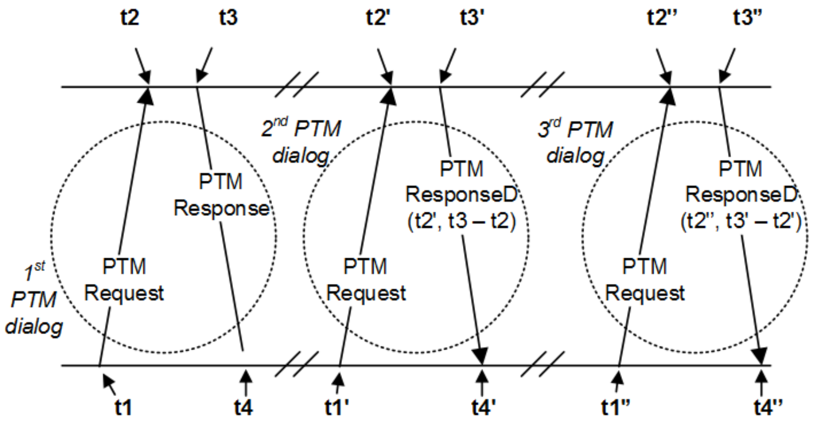
(Precision Time Measurement)

Scenario:

1. Device Driver Triggers Cross-Timestamp
2. Device initiates *PTM Request* TLP to Root Complex
3. System Time is Returned (delays are compensated)
4. (PTM Time, PTP Time) returned to Device Driver
5. Software "disciplines" two variables per clock: m and c

Cross Timestamps,
Captured Simultaneously





In-System Cross Timestamps → Time Translation Coefficients

"What Time Is It" Measurements

Three steps:

- Quickly read free-running CPU counter
- Transform to desired timescale (desired_timescale = m * TSC_value + c)
- Periodically update (m, c) using cross-timestamps

Following preliminary data is from:

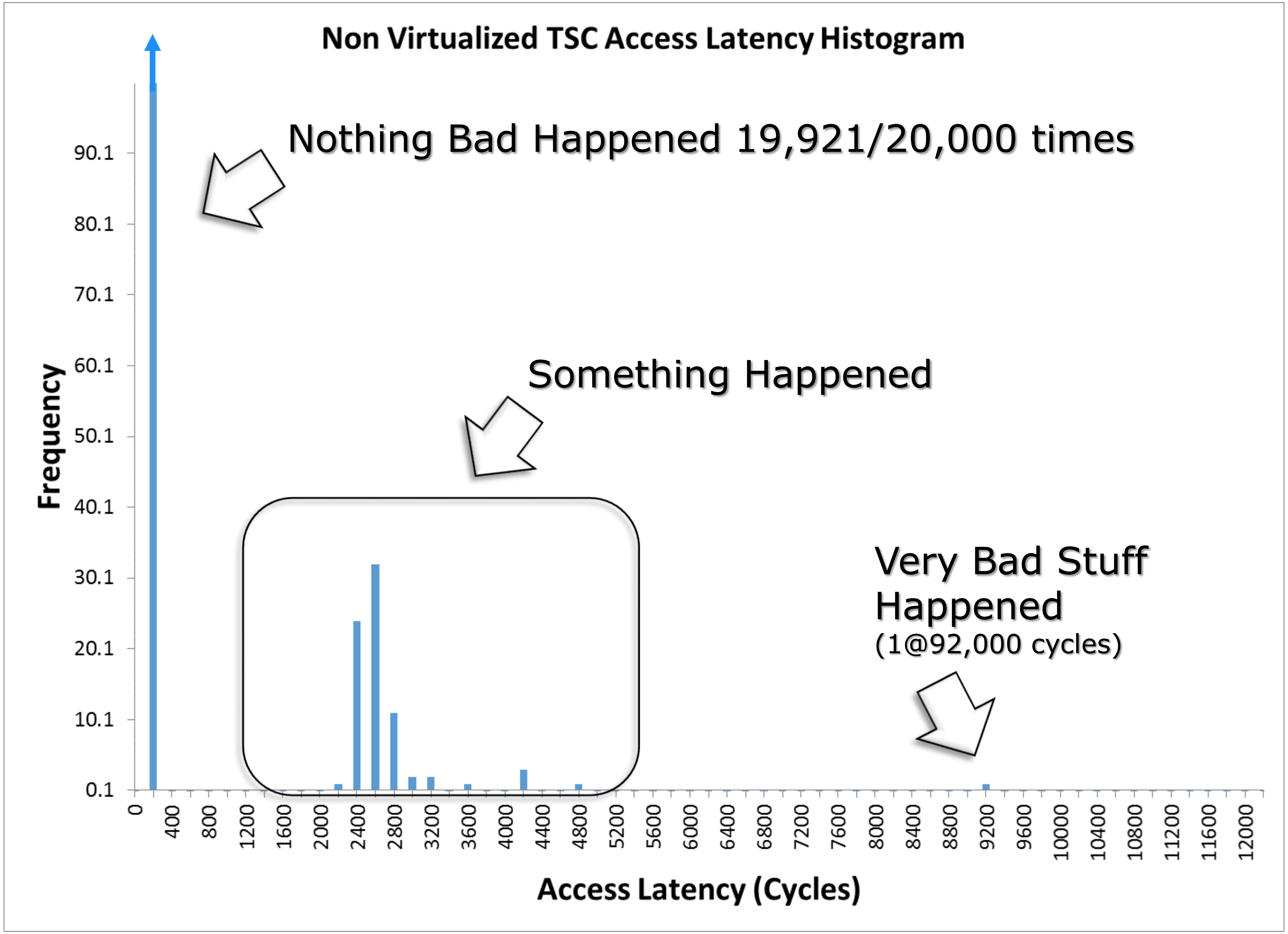
- CPU: Intel® Core™ i5-6600 CPU @ 3.30GHz
- Motherboard: Gigabyte GA-H170M-DH3

Serialized RDTSC Instructions in tight loop. Pseudocode:

- `for (i=0; i  < 20000; i++) {`
  - `timestampArray[i++] = clock_gettime(CLOCK_REALTIME);`

Graphs show latency for

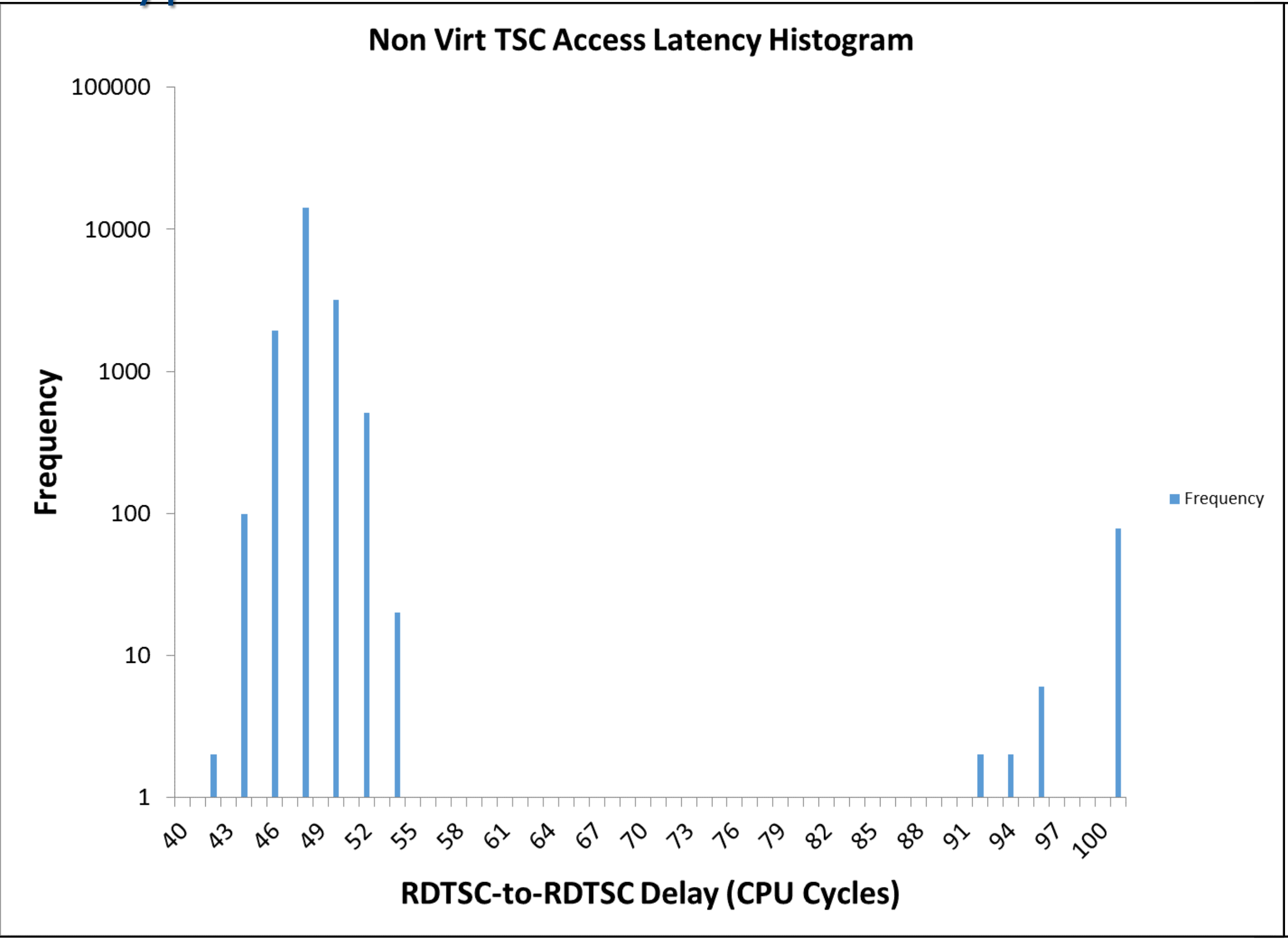
- Serializing RDTSC
- Store
- [Scaling]
- Index++
- Branch



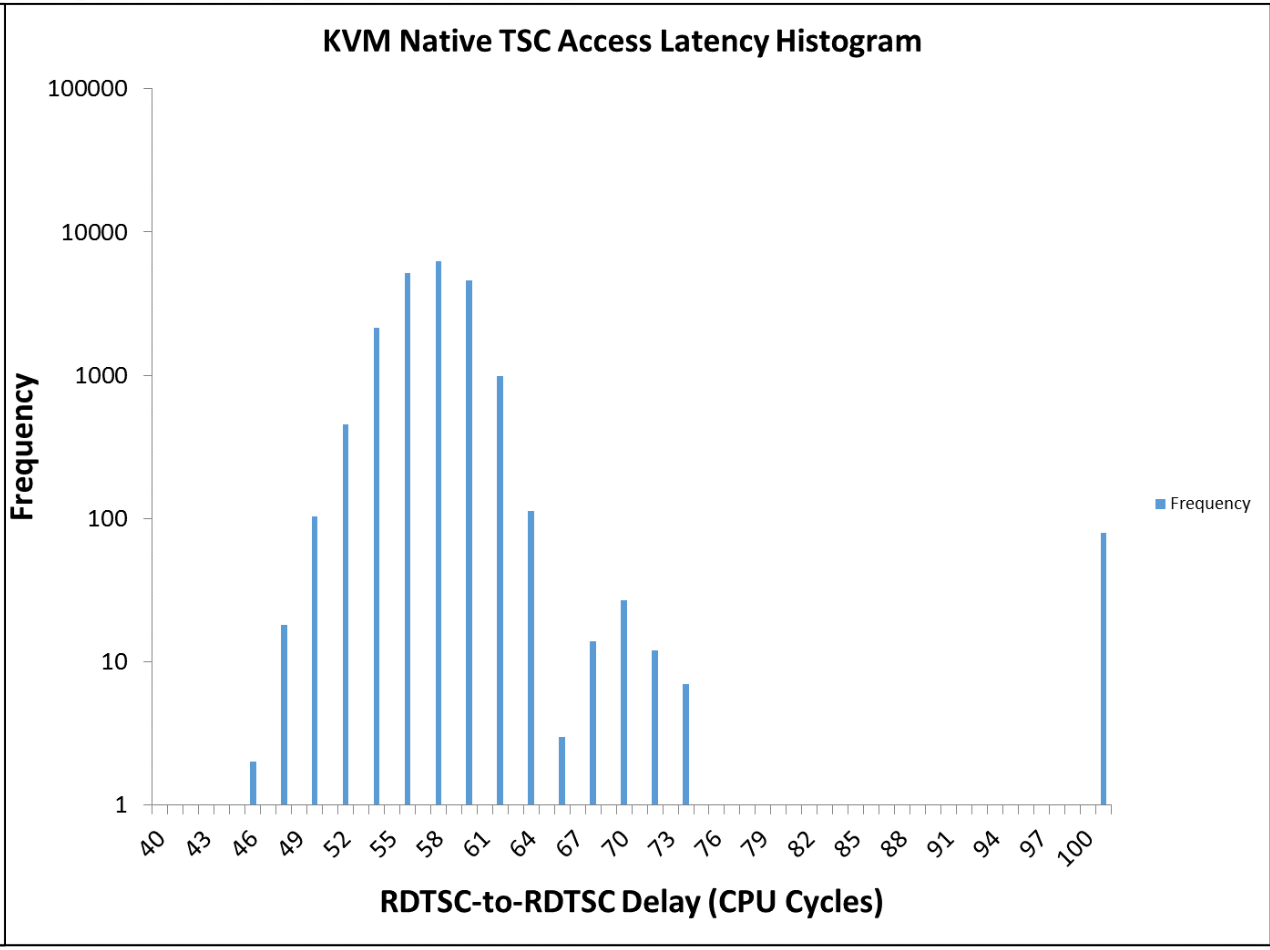
Preliminary Measurements

Direct Read of the TSC (RDTSC) – Log Y Scale

No Hypervisor Present



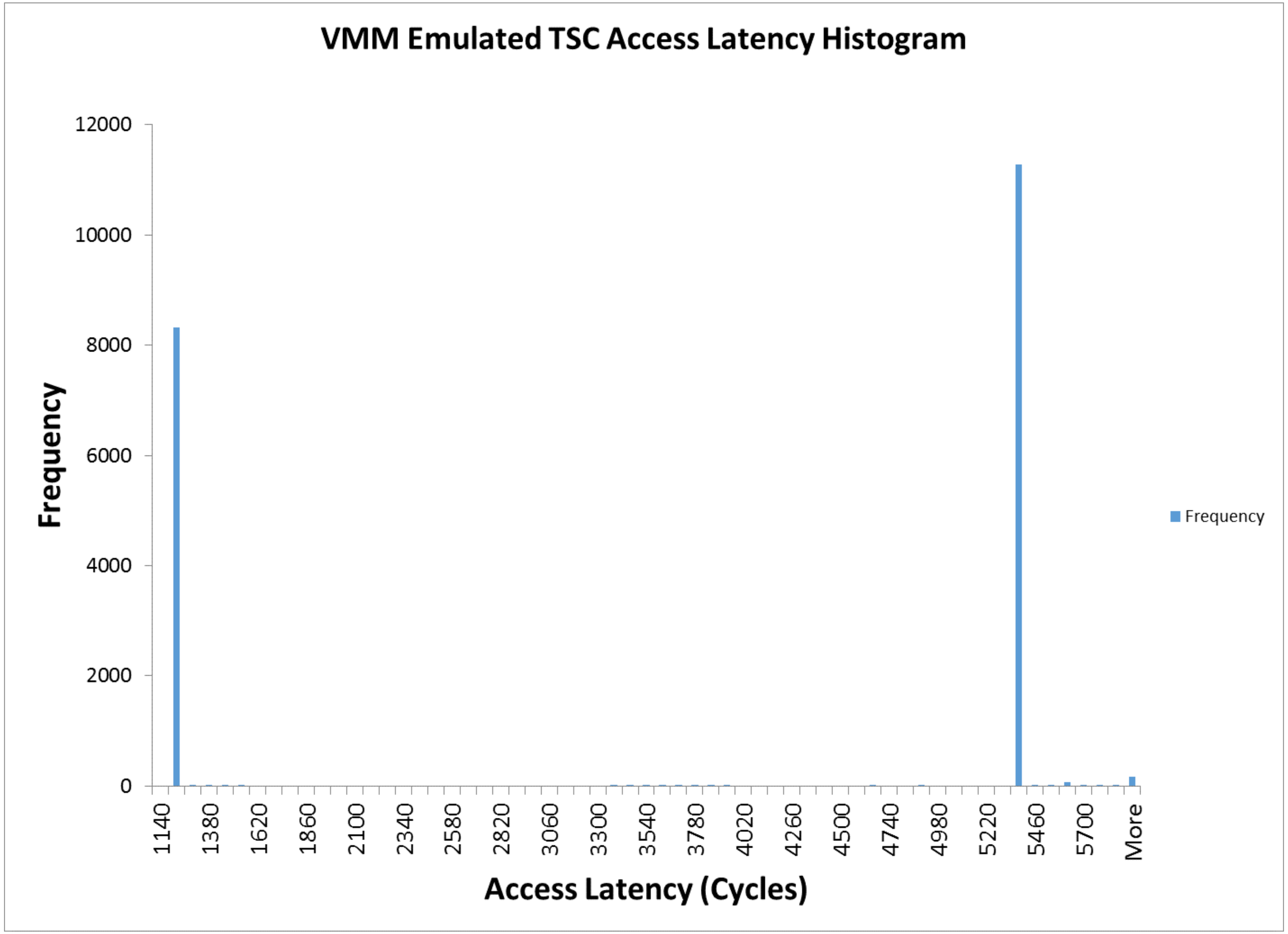
From Virtual Machine



NOTE: For clock_gettime(CLOCK_REALTIME), increases to [67-73] cycles

Preliminary Measurements

Wrong Configuration: Hypervisor Emulates TSC



Preliminary Measurements

intel Time

Conclusions

There are many uses for synchronized time → Software

Virtualization need not preclude immediate software access to accurate synchronized time – but software execution might be interrupted.

Hypervisor can introduce long time-read delays if misconfigured