SYSTEM-LEVEL VIEW OF
END-TO-END TIME SYNCHRONIZATION

Kevin B. Stanton, Ph.D.
Today: Independent Consultant (and Adjunct Prof.)
Formerly: Sr. Principal Engineer, Intel Corporation

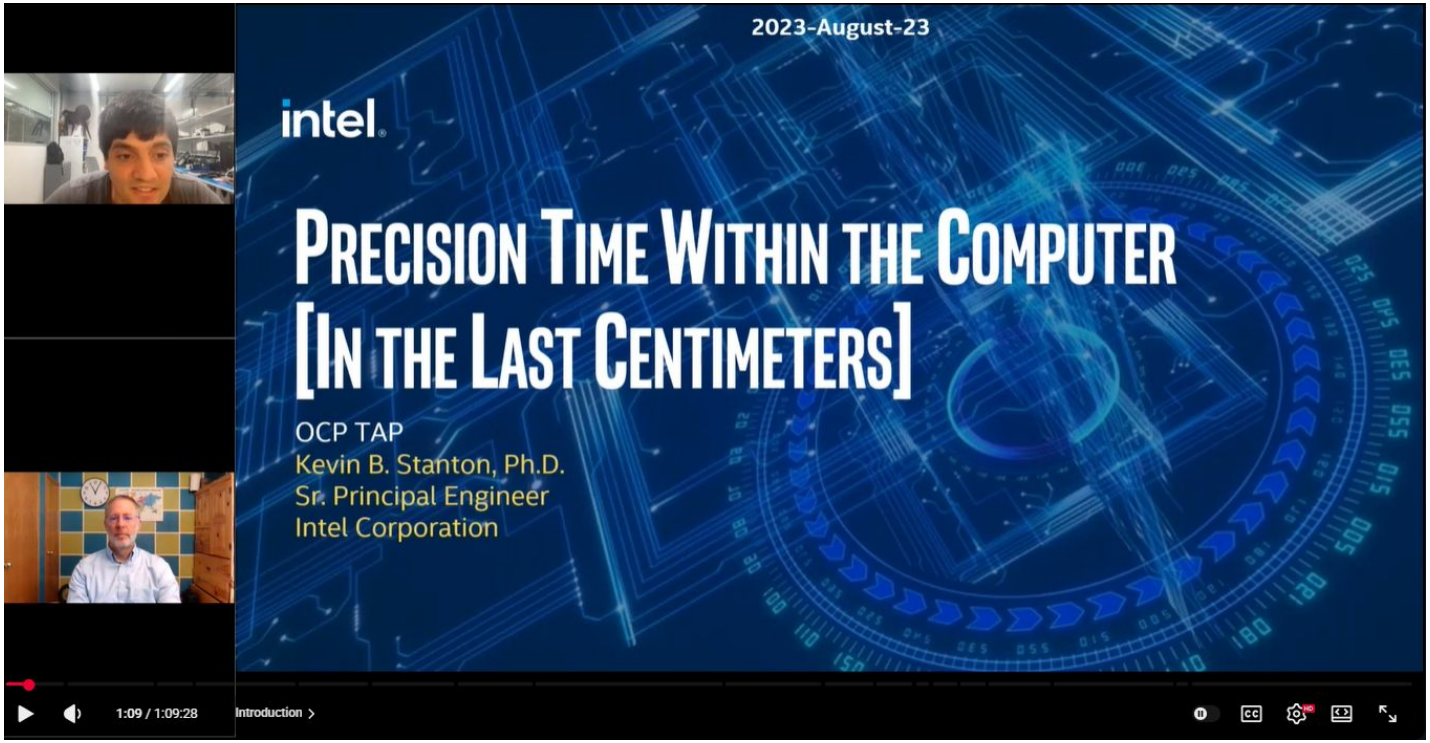
TUTORIAL FOR: WSTS Conference 2026

Agenda

System-level View of End-to-end Time Synchronization

- **Hardware Endpoint Clocks**
  - X86, ARM, RISC-V
- **Software Access to Clock(s)**
- **Time Transfer Fundamentals**
- **Measuring Clock Agreement**
  - Read/Read versus TGPIO
- **PTM: Time Transfer within the System**
- **Software:**
  - For 1588/PTP: PTP4L, PHC2SYS, CHRONY
  - For CPU clock management: Drivers, OS, POSIX

OCP TAP CAll: Precision Time in the Last Centimeters with PCIe PTM: A Deeper Dive, August 23, 2023



https://youtu.be/9OILFLV-Sfc?si=ouLMBpJMILOV0bZN

Hardware Endpoint Clocks

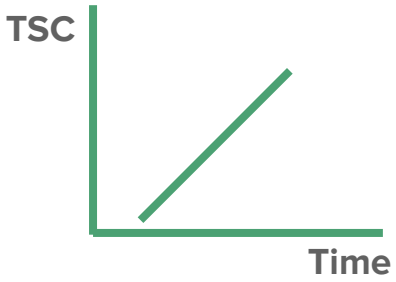
x86, ARM, RISC-V

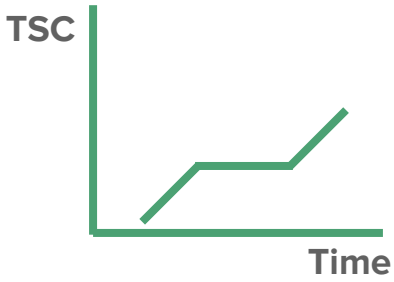
CPU Hardware Clocks

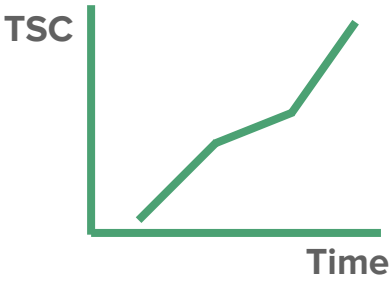
x86	ARM	RISC-V
TSC (Time Stamp Counter) RDTSC instruction Invariant TSC (constant rate) Synchronized across cores Max Non-Turbo Frequency	Generic Timer CNTPCT_EL0 register Per-core counters Local access via MRS inst. Runs at 1 GHz (v8.6+)	RDTIME instruction MTIME register Spec-defined interface Platform-dependent freq

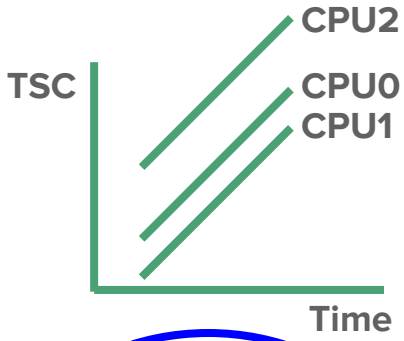
Modern CPUs provide: monotonic, high-resolution counters for timekeeping

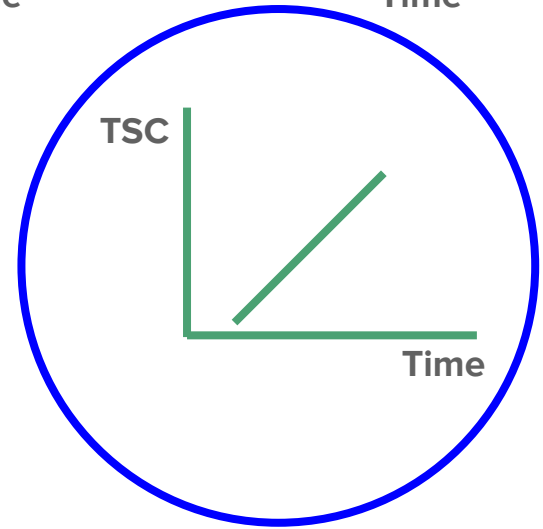
Modern x86 TSC Behavior — Which is it?



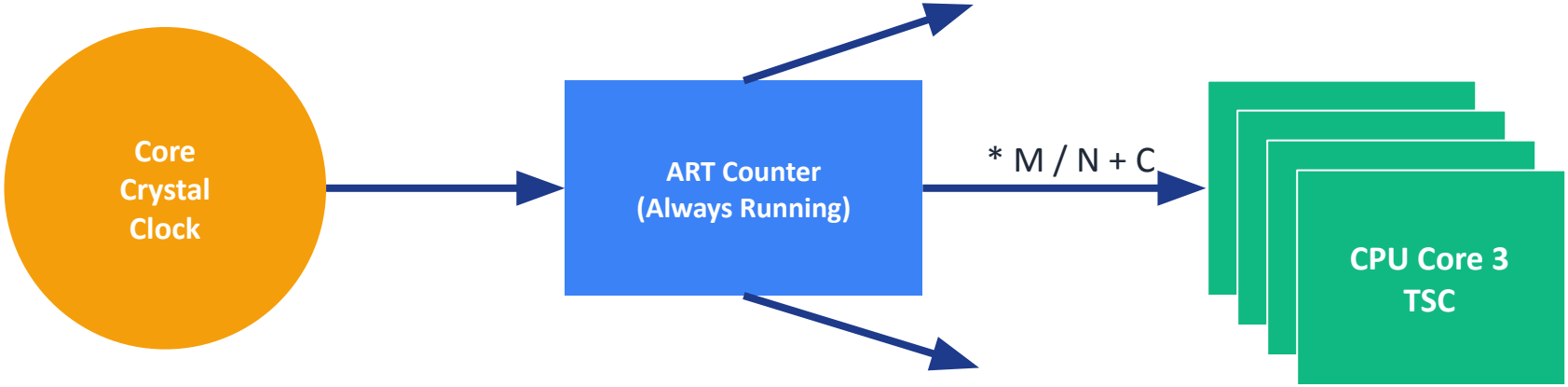








[Intel] x86: TSC and ART Relationship

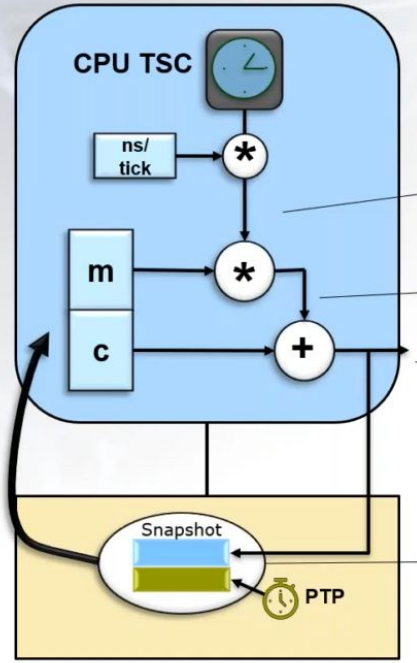


- TSC = Time Stamp Counter (RDTSC instruction) — counts at the max non-turbo frequency
- ART = Always Running Timer (hardware foundation)
- `Conversion: TSC = (ART × CPUID[15H].EBX) ÷ CPUID[15H].EAX + Constant`
- Nominal Core Crystal Clock frequency is also available, in `CPUID[15H].ECX`

Software Access to Local Clocks

Software: POSIX Clock Access

CPU Counter → Synchronized Time



Time "now" (from a Linux application)

clock_gettime(CLOCK_MONOTONIC_RAW, &now);
- Returns TSC scaled to nominal nanoseconds

clock_gettime(CLOCK_MONOTONIC, &now);
- Returns TSC scaled to track TAI, in nanoseconds

clock_gettime(CLOCK_REALTIME, &now);
- Returns CLOCK_MONOTONIC + (now-1/1/1970) [incl. leap seconds]

Cross-Timestamp "Snapshot"

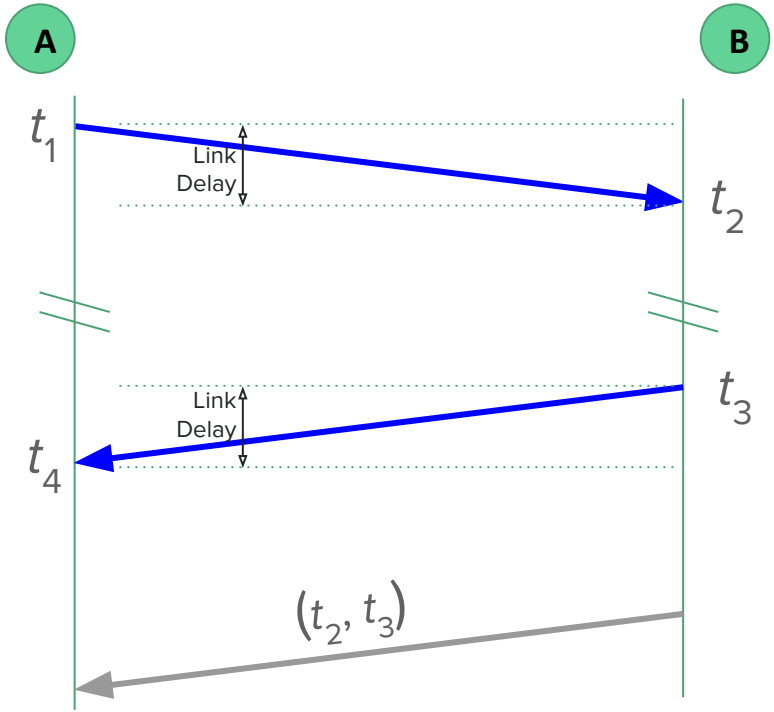
ioctl(phc_fd, PTP_SYS_OFFSET[**_PRECISE**])
- Returns (CLOCK_REALTIME, PTP_TIME)
- Used to nudge **m** up or down over time

Piecewise-Linear Clock Model: y=mx+c

https://youtu.be/9OILFLV-Sfc?si=ohnZ9U07JT7eoYN6&t=1211

Time Transfer Fundamentals

Two-Way Transfer: Detail



$$\text{Link Delay} = \frac{[(t_4 - t_1) - (t_3 - t_2)]}{2}$$

$$\text{Clock Offset} = \frac{[(t_2 - t_1) - (t_4 - t_3)]}{2}$$

$$\text{Clock Offset} = t_2 - (t_1 + \text{Link Delay})$$

Assumes symmetric path/link delay

Hardware or Software Packet Timestamps

Why it matters for PTP accuracy

Software Timestamping

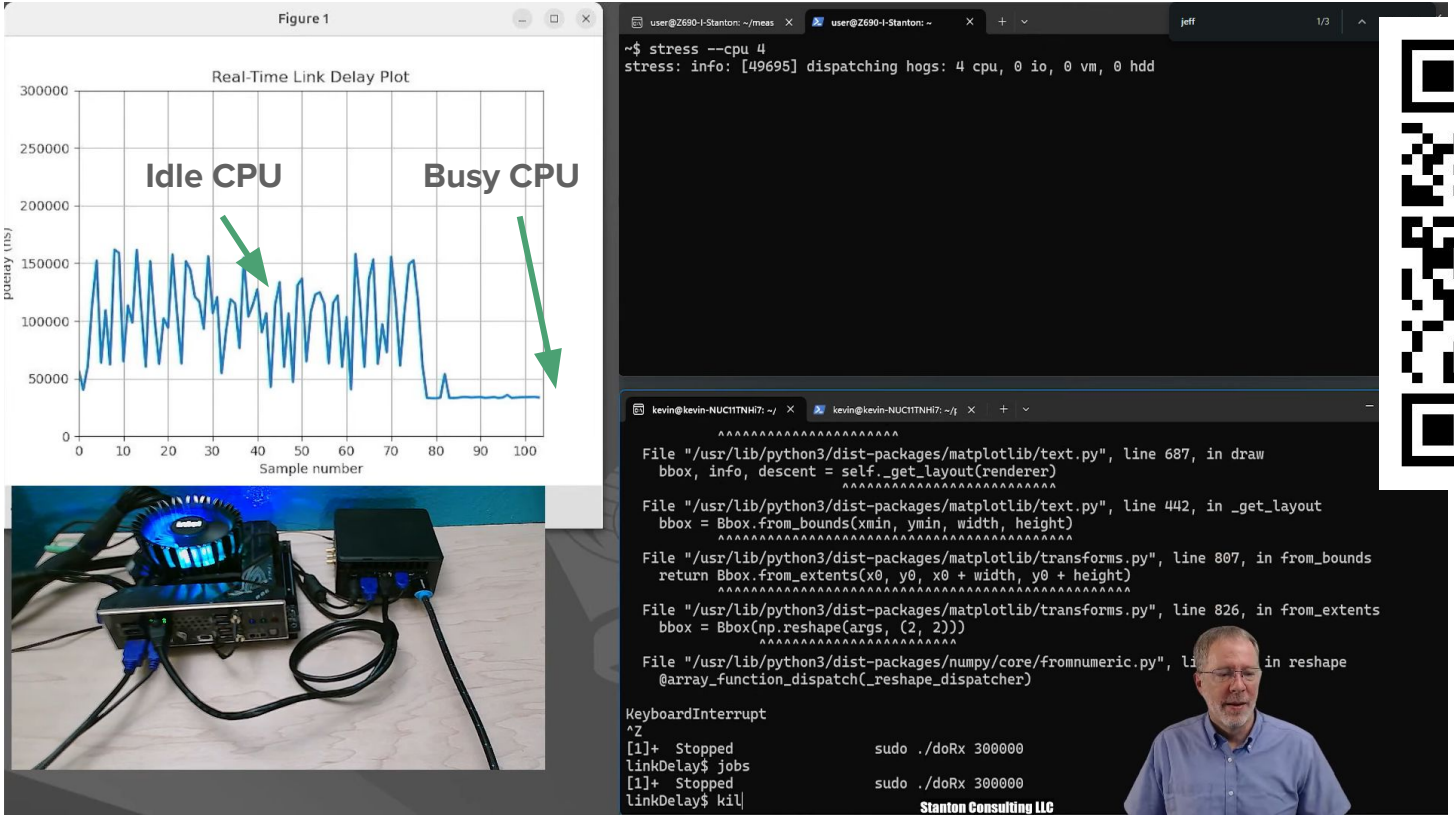
- Timestamp in network stack
- Variable kernel latency
- Interrupt delays
- Microsecond-level jitter
- Accuracy: ~1,000-100,000 ns

Hardware Timestamping

- Timestamp at PHY layer
- Deterministic timing
- No software overhead
- Nanosecond precision
- Accuracy: ~10 ns typical

PTP Link Delay with HW and SW Timestamps

SW Timestamps and CPU Low Power States



https://www.youtube.com/playlist?list=PLs8GY_qM2UhYZOym0Sdo0QVF3oHERQZY1

IEEE 1588 / PTP Protocol

…Covered in the other WSTS tutorials

Measuring Clock Agreement

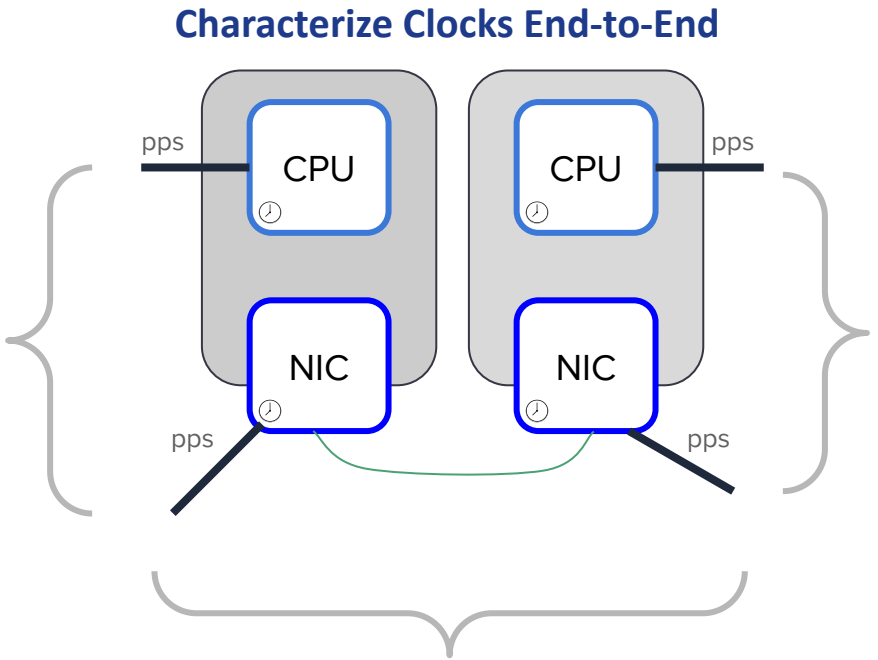
TGPIO

TGPIO: Time-Aware GPIO

Pulse Per Second (PPS) for external verification

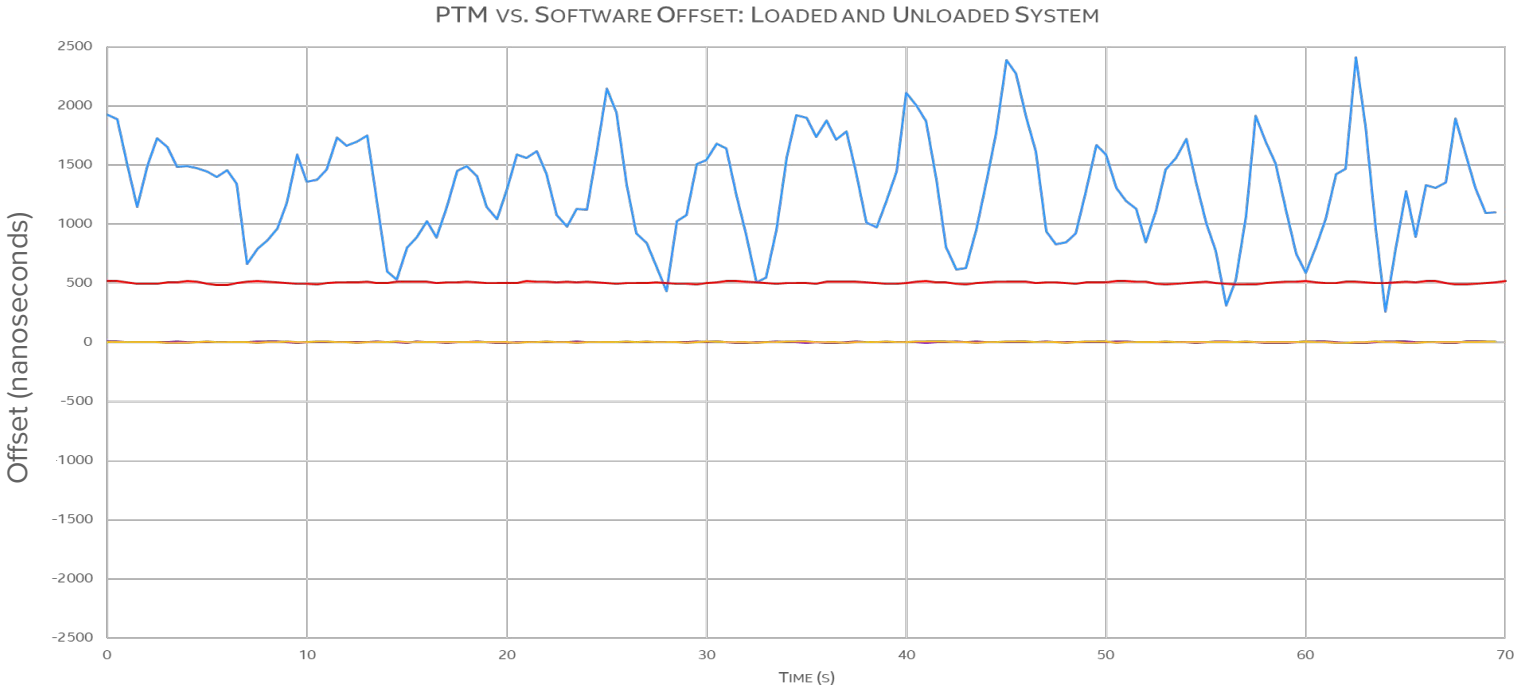
Purpose

- CPU generates PPS output based on its system clock
- Verify time synchronization with oscilloscope or logica analyzer
- Compare local vs. reference time
- Ground Truth of actual clock offsets



PTM: Time Transfer
WITHIN the System

PTM Versus Software Cross-Timestamps



Intel Atom® x6000E / i225 | Credit: Chris Hall @ Intel Corporation

Without PTM:

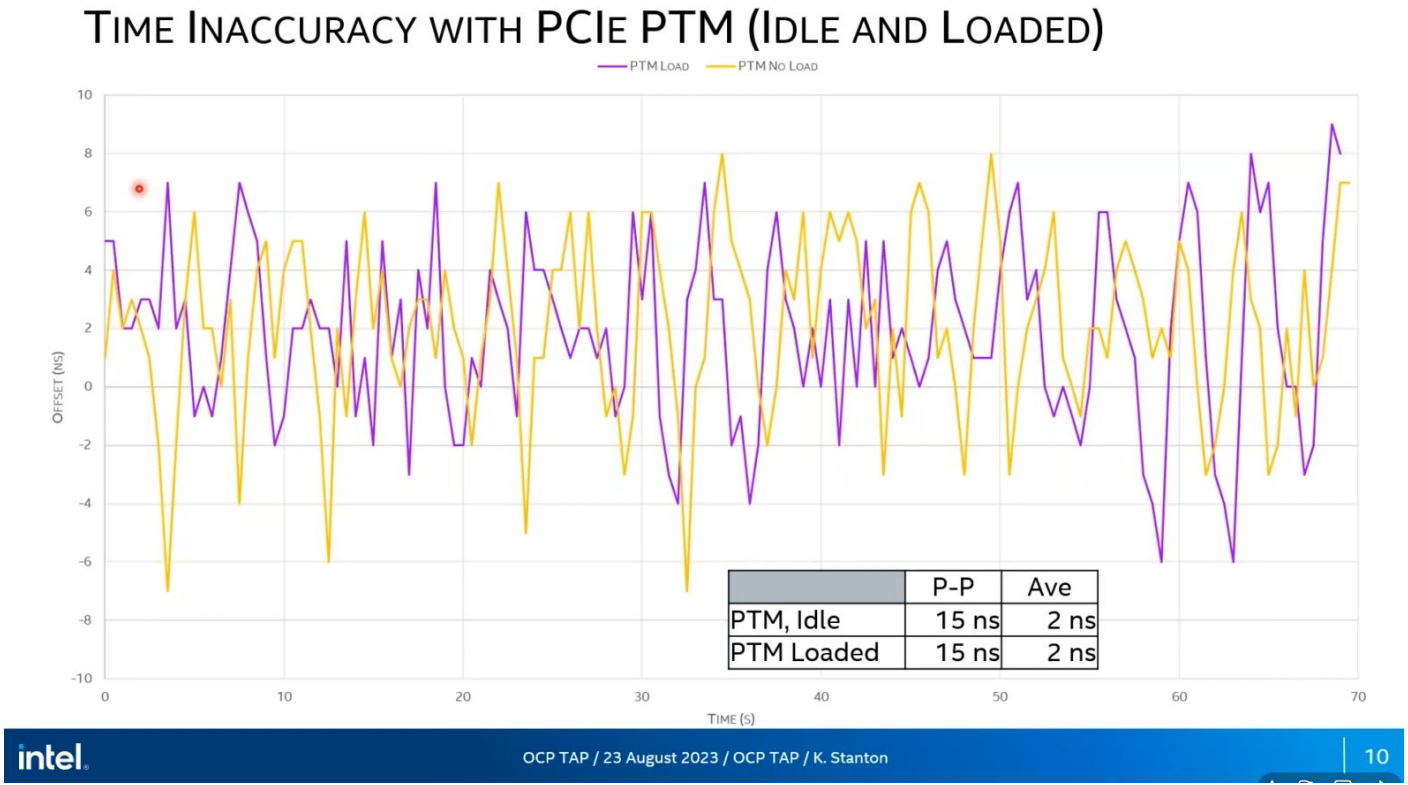
Idle: ~500 ns avg
Load: ~1,300 ns
Peak: 2,148 ns

With PTM:

Always: 2 ns avg
Peak: 15 ns

https://youtu.be/9OILFLV-Sfc?si=ohnZ9U07JT7eoYN6&t=1211

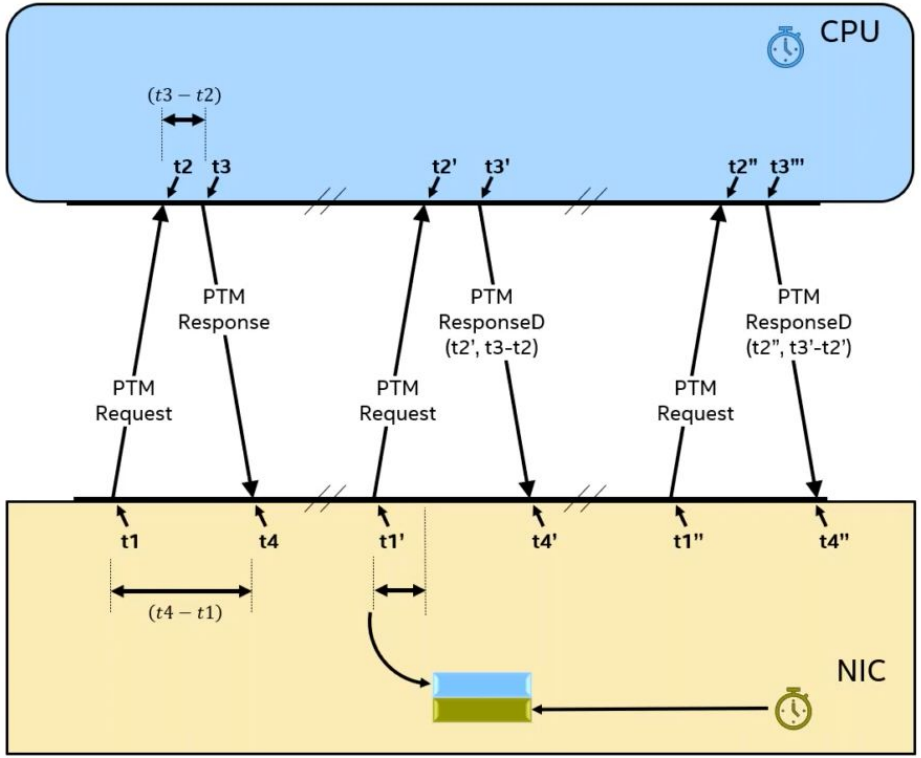
PTM Cross-Timestamp Performance



Intel Atom® x6000E / i225 | Credit: Chris Hall @ Intel Corporation

https://youtu.be/9OILFLV-Sfc?si=ohnZ9U07JT7eoYN6&t=1211

PCIe PTM (Precision Time Measurement)



PCIe PTM In Action

At t1', snapshot **PTP Counter**

Soon after t4', use t1, t4, t2', t3-t2 to compute:

- $LD = LinkDelay = \frac{[(t4-t1)-(t3-t2)]}{2}$
- $PTM\ Root\ time\ @\ t1' = t2' - LinkDelay$

Return the Cross-Timestamp:

- (**PTM Root Time** @t1', **PTP Counter** @t1')

Repeat as requested (e.g. by software)

Note: All PTM timestamps are in units of nanoseconds
Note: Clock discontinuities are allowed at any time in the Upstream Port except between t1→t4, t1'→t4', etc.

https://youtu.be/9OILFLV-Sfc?si=ohnZ9U07JT7eoYN6&t=1211

Software: POSIX Clock Access

```
clock_gettime(CLOCK_MONOTONIC_RAW, &ts);
 // TSC scaled to nominal nanoseconds

clock_gettime(CLOCK_MONOTONIC, &ts);
 // TSC adjusted to track the frequency of network time

clock_gettime(CLOCK_REALTIME, &ts);
 // MONOTONIC + offset to Unix epoch

clock_gettime(CLOCK_TAI, &ts);
 // Essentially REALTIME but without the leap seconds

ioctl(phc_fd, PTP_SYS_OFFSET[_PRECISE], &pso);
   // Cross-timestamp: (system_time, PHC_time)
```

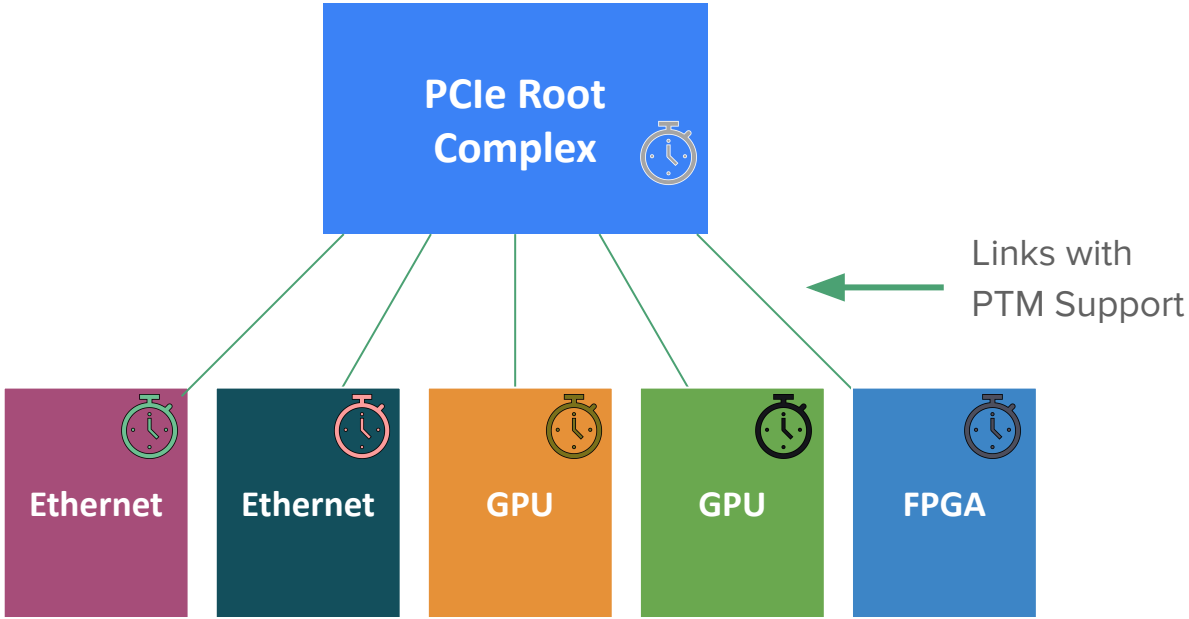
PTM: Is it for Ethernet or GPU or…?

Ethernet, GPU, FPGA, …

PTM **MEASURES** clocks relative to Root Complex

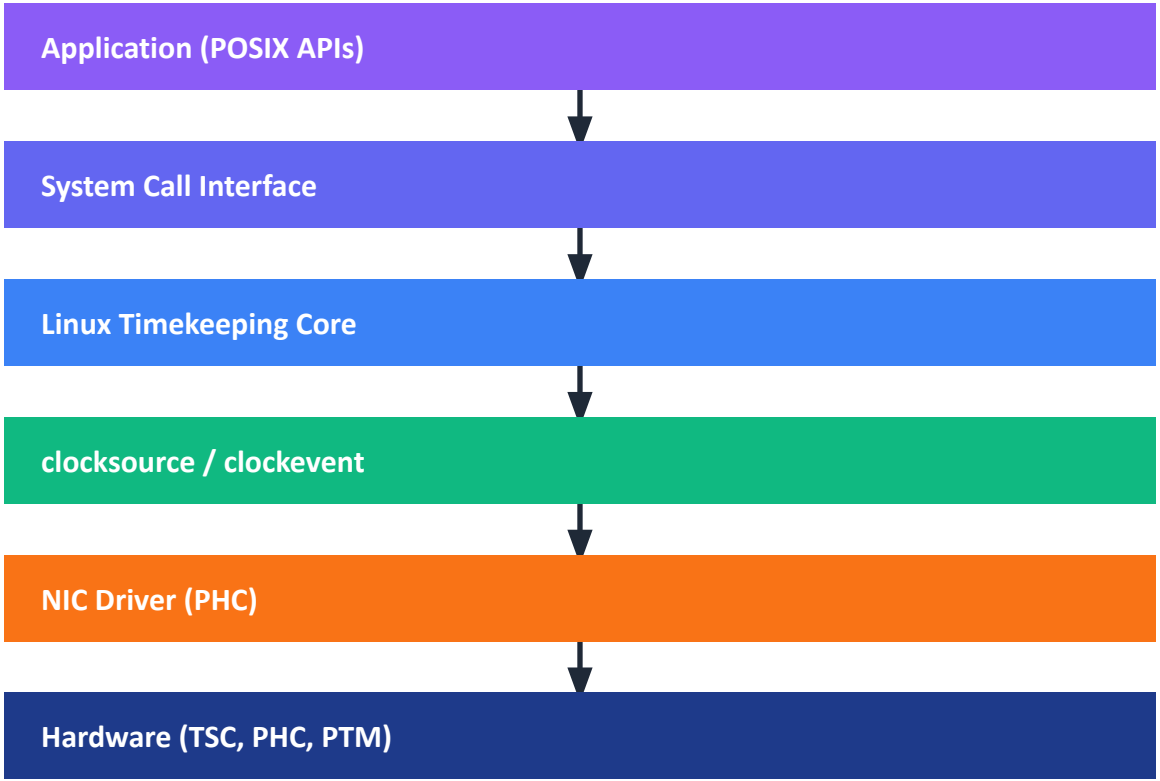
Upper-level protocols then communicate offsets

Consequence ⇒ synchronization can flow in ANY DIRECTION



Open Source Software

Kernel Infrastructure for PTP+PTM



PTP Software Ecosystem

ptp4l

IEEE 1588 PTP daemon

Time Transmitter/Receiver • Software or Hardware Timestamping • Various Profiles • Boundary clock support

phc2sys

Synchronize system clock to PHC

PHC → System clock • Uses PTM cross-timestamps (where supported) • Servo algorithms

chrony

NTP/PTP hybrid

Fast convergence • Handles network gaps • HW timestamping support

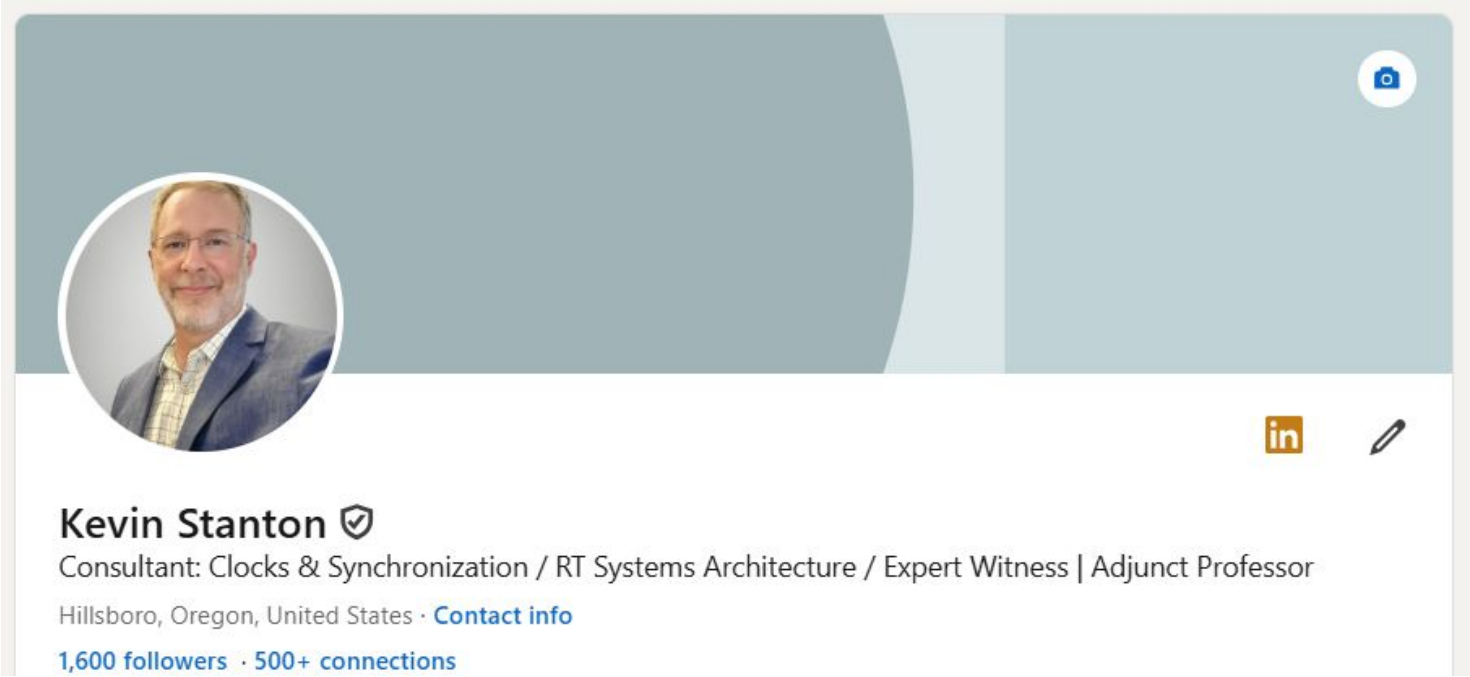
commercial software

from an increasing number of vendors

I am not associated with any. Reach out if you're interested in learning more.

And much more...

Thanks for Listening



https://www.linkedin.com/in/kevin-stanton-53b1641/