Section 3: Week 6: Clock Synchronization

Nate Bachmeier

TIM-8120: Distributed Systems

October 27th, 2019

North Central University

# Clock Synchronization

A common challenge of presenting a single system image across multiple nodes is ensuring that the sequence of events processed, is correct. Many strategies to address this issue focus on timestamps, though this approach has challenges as most clock implementations are imperfect. Clock skew causes these imperfections and needs Clock Synchronization Protocols to mitigate.

# Physical Clocks

Most mechanical devices have timers, not clocks. These timers rely on the oscillation of quartz crystal as a means to decrement a counter, signifying the duration until the clock tick system interrupt occurs (Ubolkosold, Knedlik, & Loffeld, 2005). These crystals “suffer from large frequency shifts due [to] the high sensitivity to external or internal factors (Zhang, et al., 2019).” According to Zhang et al., atomic clocks rely on energy transitions to achieve 100,000 times better precision than quartz after ten days. These more precise instruments are difficult to deploy more broadly due to both their physical size and power consumption. Researchers are continuing to explore methods to reduce these limitations.

As these timers raise clock ticks, the more extensive system needs to report that it is relative to some universal reference point. Universal Coordinated Time (UTC) is the de-facto solution as alternatives, such as regional time zones, are impacted by political cross-cutting concerns. For example, Eastern Time in America changes at different points of the year for daylight savings. Another solution is to rely on the local time zone of the device. This design introduces additional challenges for systems spanning multiple locations as the example value “2019-10-27 10:03:00” occurs numerous times.

# Clock Synchronization Algorithms

|  |  |  |  |
| --- | --- | --- | --- |
| Protocol | Description | Structure | Benefit |
| Network Time Protocol (NTP) | A protocol for replicating timestamps across the Internet. | Hierarchical with parent servers using more precise hardware. | Measurements from expensive hardware repeat economically. |
| Berkeley Algorithm (Gusella & Zatti, 1989) | A group of peers performing an election process to choose a master. The master periodically polls the slaves for their local time, then adjusts itself based on the average value after accounting for round trip time (RTT). | Centralized group of peers. | Easy to deploy for a cluster that only needs to be precise to itself. Works in labs without Internet access. |
| Reference Broadcast Synch (Akbar, Ichsan, & Darmawan, 2019) | Relies on a beacon to periodically broadcast reference packets to receivers, which compares the arrival time against the local time. The beacon message contains a sequence number, not a timestamp. The receiver then compares the reference message timestamp against its peers. | A broadcast model for decentralized peers to gain a shared synchronization object. | It removes the distortion caused by network latency between wireless sensors. |
| Logical Clocks | A message counting protocol. Alice sends a message (with counter=0) to Bob, then Bob forwards then the message (with counter=1) to Charlie. Afterward, an observer of the transactions can efficiently determine the *happened-before* flow using only the counter. | A simple message counter that increments at each hop. | It addresses scenarios that need local relative offsets, not UTC timestamps. |
| Total Ordered Multicasting (Defago, Schiper, & Urban, 2004) | A collection of strategies for broadcasting in-order events to multiple consumers. | A multicasting solution of sequenced events. | Improves the performance of scenarios, such as database replication, that avoids reordering on the consumer side. |
| Vector Clocks | A similar strategy to Logical Clocks, except an array of multiple process’s local counters, is used, instead of a single scalar. | Each process updates a unique event counter on the local message. | It allows for more sophisticated modeling of causality relationships. Perhaps a multicast began a scatter/gather operation, making a single counter erroneous. |
| Token-Based Solutions | A strategy where a process that possesses a *token* is allowed to enter a critical section. This solution differs from non-token systems that use *timestamps* to sequence afterward. | A process requests the token from a store and must wait until it is received. Afterward, it can exclusively use a resource. | Removes the challenges of resolving conflicts, at the cost of performance due to increased lock contention. |

# Using Clock Synchronization

Contoso’s Retail eCommerce Platform (see Figure 1) needs to use a variety of event synchronization strategies. Multiple solutions are necessary as each focuses on a different aspect of the holistic system.

Systems within the data center periodically poll from a centralized network time services node, which in turn is regularly querying public government NTP services. This strategy ensures that clocks are generally aligned, but can still fall victim to clock drift. For instance, the shopping cart service runs on virtualized hardware, and the multi-tenant computation environment does not guarantee every system interrupt is timely delivered. Similar, the authentication services run in a part of the data center that abnormally hot, and causes the quartz to oscillate slightly faster.

As the customer adds and removes items into the shopping cart, these operations need to be sequenced. These requests flow across multiple service instances into an eventually consistent data store. To sequence these operations after the fact, the system designers can use event counters that are managed by the web-portal. After deploying the solution, they discover that this strategy does not scale, as multiple web portal instances need to use token-based locking to manage the central counter. They replace the counter management with a total ordered multi-casting solution. An additional benefit of this change is that the recommendation services can remove much of the event reordering logic. Total ordered multicasting also exists within the datastores as they perform replication.

A business requirement of the payment service states that a user can only proceed through one checkout process at a time. This check exists to ensure the third-party partner does not create partially redundant customer orders. The system enforces the distributed mutual exclusion through the use of a token that represents the user context. If a customer attempts to checkout from a different shopping cart, they cannot acquire the token, and the web portal guides them through a workflow to either cancel or wait.

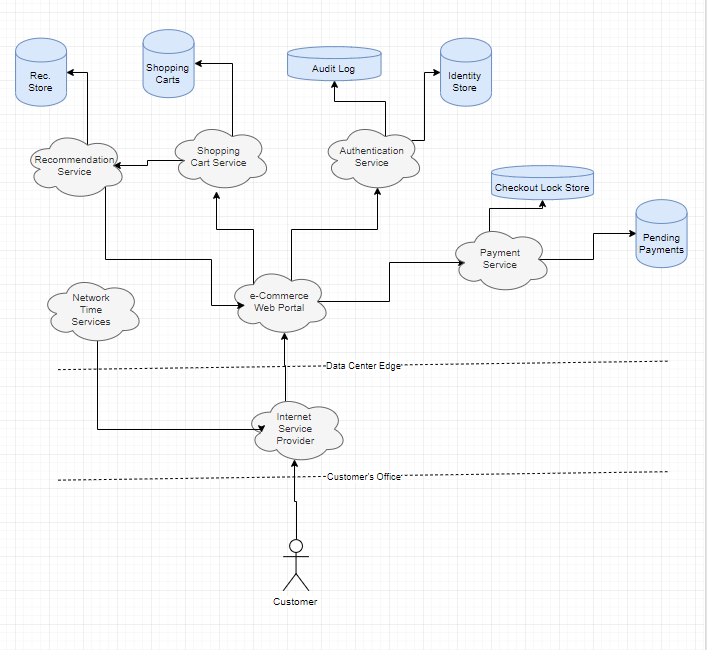


Figure 1: High-Level SoA

# Processors

## Chart AMD versus Intel versus ARM Pipelining

Instructions into a CPU processor need decoding, fetching, transforming, and then writing back to memory. This process, called a pipeline, is performed on both Reduced and Complex Instruction (RISC/CISC) set processors. Modern Intel and AMD 64bit processors, use a combination of the two with many CISC instructions implemented as a series of RISC execute operations (Patterson, 2017). Both manufacturers support the execution of two instructions per core with consumer-grade silicon supporting up to 32 cores per chip. Each physical core supports Simultaneous Multithreading (SMT) and can weave out of order instructions into the gaps (WikiChip, 2019). Where the two manufactures differ, is architectural decisions around which objects are shared or decoupled. For instance, Intel chose a single instruction scheduler, or AMD coupled instructions and data into the same cache (AdoredTV, 2016). There have also been deviations with the memory management units and strategies for branch predictions (Fog, 2015).

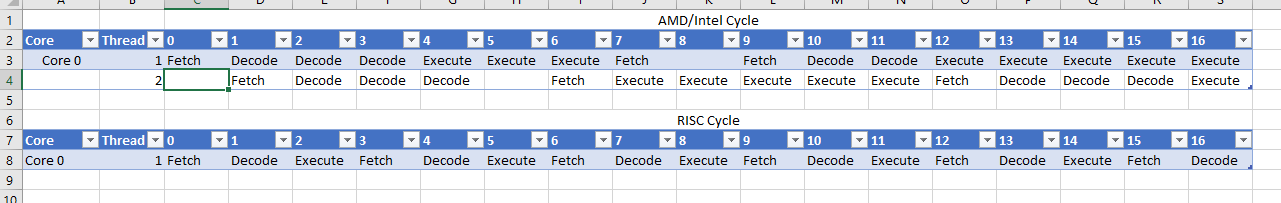


Figure 2: Pipelining

## Choosing RISC versus CISC for Distributed Systems

“One of the major differences between RISC and CISC is that RISC emphasizes efficiency in cycles per instruction, and CISC emphasizes efficiency in instructions per program (Thornton, 2018).” RISC has the potential for greater *portability*, as there are fewer design requirements. On the other hand, CISC has a higher potential for *compatibility,* as it implements a broader specification. The mainstream market has decided that RISC is the de-facto solution for mobile, and CISC for desktop and server scenarios. However, since the early 2000s, the software landscape has drastically changed, but not this dividing line in the sand (Blem, Menom, & Sankaralingam, 2015). Blem et al. reviewed seven processors across twenty-six workloads to collect 20,000 data points, and conclude the line should not exist as the Instruction Set Architecture (ISA) is irrelevant-- CISC can copy and optimization that RISC introduces.

Patterson states that the challenges between CISC versus RISC are not the ISA itself, but come from the bloat that is expensive and unused (Patterson, 2017). He provides an example of DEC’s VAX instructions, which account for 60% of the microcode and 0.2% of the execution time. Perhaps something in between these two extremes needs to exist, allowing device manufacturers to create a “RISC++” that focuses on a modular design for a Domain Specific Architectures (DSAs), similar as Google’s Tensor Processing Unit (Muzaffar & Elfadel, 2019) or Amazon’s Nitro ASICs (Sharwood, 2017). As the conversation moves away from CISC versus RISC and toward DSA, distributed systems can gain efficiencies by removing waste, adding hardware acceleration, and gaining expertise.

Figure 3 contains a distributed system with labels for the potential processor types. It starts with users continuing to use either CISC based desktops or RISC based mobile devices. ASIC edge routers will service their requests by forwarding into legacy traffic management systems on CISC. Web Frontend components will transition towards lighter instruction sets, and rely on dedicated ASIC or FPGA to accelerate their hardware needs. Backend services are the least incentivized to move off of CISC, as these are the target audience of many instruction set extensions. Storage networks and stores can leverage RISC, FPGA, and other DSA strategies to gain the most performance at the lowest price point.

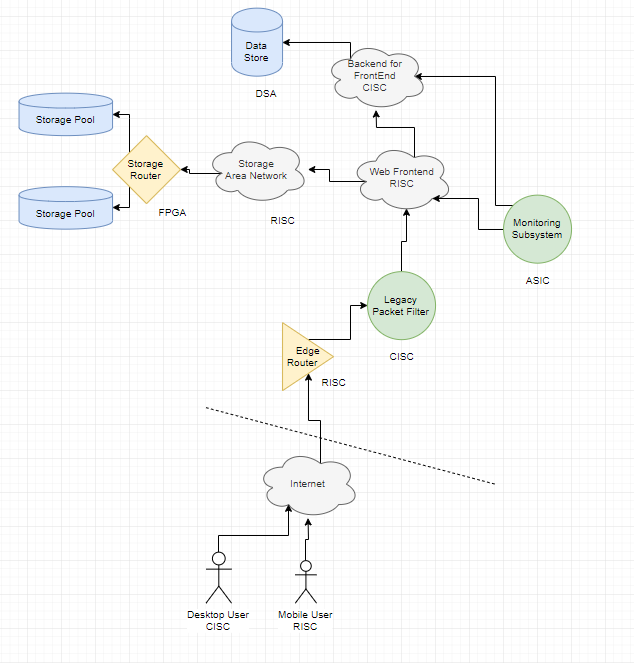


Figure 3: Processor Architecture of Future

## Importance of Processor Concurrency

The critical difference between High-Performance Computing (HPC) versus High Throughput Computing (HTC) is faster versus more (see Figure 4) (Fan, et al., 2018). Under HTC scenarios, such as serving hard real-time web content, the system needs the ability to schedule more logical threads. In contrast, an HPC can decompose the job into queued tasks and distribute them as resources are ready to pull them. Ideally, both scenarios could dedicate a physical core to each request, however that is not economical. Instead, the system needs to understand the workload and strike the right balance. Fan et al.’s experiments across large-scale many-core architectures show that balance differs for HPC and HTC concurrency. As the pendulum swings further towards the HTC scenario, it can be tempting to provision a server with hundreds of physical cores. However, managing the concurrency across a single system becomes highly complex, and performance degrades. Instead, an external scheduler can be used to orchestrate hundreds of individual virtual core servers. Using this horizontal scaling strategy reduces the contention of one virtual node impacting another.

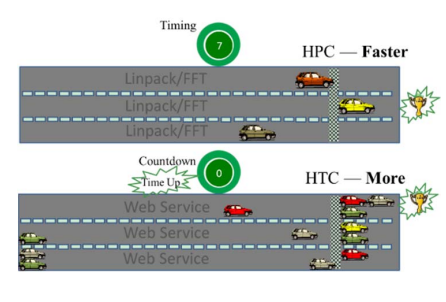


Figure 4: HTC versus HPC (Fan, et al., 2018)

# Conclusions

Distributed systems need mechanisms for synchronizing events into a time-ordered single system image. If the order cannot be reliably determined, then race conditions will appear in the user experience and cause undesirable behaviors. When determining the sequence, protocols exist for both absolute timestamps and relative counters. For many production environments, like Contoso’s eCommerce Portal, a combination of both types will be required.

Modern Processors are some of the most complex distributed systems available, and researchers need to understand their designs. They come in two flavors CISC and RISC, with CISC presently the defacto solution for desktops and service, while RISC continues as the market leader of mobile and embedded platforms. However, in the future, a transition more toward the center is likely to unfold, as the necessity to offload complexity into dedicated modules becomes more commonplace.

# References

AdoredTV. (2016, August 26). *Zen - AMD's New CPU Architecture*. Retrieved from YouTube: https://www.youtube.com/watch?v=vBXk-VSJ7UI

Akbar, S., Ichsan, M., & Darmawan, A. (2019). Reference broadcast synchronization and time division multiple access implementations on WSN. *TELKOMNIKA, Vol.17, No.1, February*, 291-298.

Blem, E., Menom, J. V., & Sankaralingam, K. (2015). ISA Wars: Understanding the Relevance of ISA being RISC or CISC to Performance, Power, and Energy on Modern Architectures. *ACM Transactions on Computer Systems. March, Vol. 33, Issue 1*, 1-34.

Defago, X., Schiper, A., & Urban, P. (2004). Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey. *ACM Computing Surveys. Dec 2004, Vol. 36, Issue 4*, 372.

Fan, D., Li, W., Yi, X., Wang, D., Zhang, H., Tang, Z., & Sun, N. (2018). SmarCo: An Efficient Many-Core Processor for High-Throughput Applications in Datacenters. *2018 IEEE International Symposium on High-Performance Computer Architecture (HPCA) HPCA High-Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on.*, 596-607.

Fog, A. (2015). The microarchitecture of Intel, AMD, and VIA CPUs: An optimization guide for assembly programmers and compiler makers. The *Technical University of Denmark*.

Gusella, R., & Zatti, S. (1989). The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. VOL. 15, NO. 7, JULY*, 847-855.

Muzaffar, S., & Elfadel, I. (2019). A Domain-Specific Processor Microarchitecture for Energy-Efficient, Dynamic IoT Communication. *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 27, NO. 9, SEPTEMBER* , 2074-2088.

Patterson, D. (2017). Reduced Instruction Set Computers Then and Now. *COMPUTER PUBLISHED BY THE IEEE COMPUTER SOCIETY*, 10-13.

Sharwood, S. (2017, November 29). *Amazon reveals 'Nitro'... Custom ASICs and boxes that do grunt work so EC2 hosts can just run instances*. Retrieved from The Register: https://www.theregister.co.uk/2017/11/29/aws\_reveals\_nitro\_architecture\_bare\_metal\_ec2\_guard\_duty\_security\_tool/

Thornton, S. (2018, January 9). *RISC vs. CISC Architectures: Which one is Better*. Retrieved from Microcontroller TIPS: https://www.microcontrollertips.com/risc-vs-cisc-architectures-one-better/

Ubolkosold, P., Knedlik, S., & Loffeld, O. (2005). Clock Synchronization Protocol for Distributed Satellite Networks. *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Geoscience and remote sensing Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings. 2005 IEEE International*.

WikiChip. (2019). *Zen - Microarchitectures - AMD*. Retrieved from Wikichip: https://en.wikichip.org/wiki/amd/microarchitectures/zen#Pipeline

Zhang, H, Herdian, H., Narayanan, A., Shirane, A., Suzuki, M., . . . Okada, K. (2019). ULPAC: A Miniaturized Ultralow-Power Atomic Clock. *IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 54, NO. 11, NOVEMBER, 2019*, 3135-3149.