Latency Numbers

Key Points

- It seems likely that the video discusses standard latency numbers programmers should know, based on its title and related online content.
- Research suggests these numbers include times for operations like L1 cache access (0.5 ns) and network round trips (up to 150 ms), varying by hardware.
- The evidence leans toward these numbers being approximate, with updates reflecting technological advancements, especially in SSDs and networks.

Introduction

The video “Latency Numbers Programmer Should Know: Crash Course System Design #1”likely covers essential latency figures for computer operations, crucial for system design. These numbers help programmers understand performance impacts and optimize systems.

Latency Numbers and Their Importance

Latency is the delay between initiating and completing an operation, like accessing memory or sending data over a network. The video probably lists typical latencies, such as: - L1 cache reference at 0.5 nanoseconds (ns), the fastest memory access. - A round trip within the same datacenter at 500 microseconds (us) or 0.5 milliseconds (ms), affecting distributed systems.

These figures, while approximate, guide decisions in system design, like choosing between memory and disk storage.

Context in System Design

Understanding these latencies helps in optimizing code, making trade-offs, and enhancing user experience. For instance, knowing a disk seek takes 10 ms can influence database design to minimize such operations.

Unexpected Detail

An interesting aspect is how these numbers, like SSD read times, have improved with technology, yet core CPU latencies like L1 cache access remain stable, showing hardware evolution’s uneven impact.

Survey Note: Detailed Analysis of Latency Numbers from the Video

This note provides a comprehensive exploration of the latency numbers likely discussed in the video “Latency Numbers Programmer Should Know: Crash Course System Design #1,”based on available online content and related resources. The analysis aims to synthesize information for programmers and system designers, offering both a summary and detailed insights into the significance of these numbers.

Background and Context The video, accessible at YouTube, is part of a series on system design, focusing on latency numbers critical for programmers. Latency, defined as the time delay between the initiation and completion of an operation, is pivotal in understanding system performance. Given the video’s title and related searches, it seems to cover standard latency figures popularized by figures like Jeff Dean from Google, often referenced in programming communities.

Online searches revealed several resources discussing these numbers, including a GitHub Gist titled “Latency Numbers Every Programmer Should Know”(GitHub Gist) and a Medium article from 2023 (Medium Article). These sources, along with a High Scalability post from 2013 (High Scalability), provided a foundation for compiling the likely content of the video.

Compilation of Latency Numbers Based on the gathered information, the following table summarizes the standard latency numbers, likely discussed in the video, with explanations for each operation:

Operation	Latency (ns)	Latency (us)	Latency (ms)	Explanation
L1 cache reference	0.5	-	-	Accessing data in the Level 1 cache, the fastest memory near the CPU.
Branch mispredict	5	-	-	Penalty when the CPU incorrectly predicts a conditional branch.
L2 cache reference	7	-	-	Accessing data in Level 2 cache, larger than L1 but slower.
Mutex lock/unlock	25	-	-	Time to acquire and release a mutex in multithreaded programs.
Main memory reference	100	-	-	Accessing data from main Random Access Memory (RAM).
Compress 1K bytes with Zippy	10,000	10	-	Time to compress 1 kilobyte using the Zippy algorithm.
Send 1 KB bytes over 1 Gbps network	10,000	10	-	Time to transmit 1 kilobyte over a 1 Gigabit per second network.
Read 4 KB randomly from SSD	150,000	150	-	Random read of 4 kilobytes from a solid-state drive.

Operation	Latency (ns)	Latency (us)	Latency (ms)	Explanation
Read 1 MB sequentially from memory	250,000	250	-	Sequential read of 1 megabyte from main memory.
Round trip within same datacenter	500,000	500	0.5	Network round trip time within the same datacenter.
Read 1 MB sequentially from SSD	1,000,000	1,000	1	Sequential read of 1 megabyte from an SSD.
HDD seek	10,000,000	10,000	10	Time for a hard disk drive to seek to a new position.
Read 1 MB sequentially from disk	20,000,000	20,000	20	Sequential read of 1 megabyte from an HDD.
Send packet CA->Netherlands->CA	150,000,000	150,000	150	Round trip time for a network packet from California to Netherlands.

These numbers, primarily from 2012 with some updates, reflect typical hardware performance, with variations noted in recent discussions, especially for SSDs and networks due to technological advancements.

Analysis and Implications The latency numbers are not fixed and can vary based on specific hardware and configurations. For instance, a 2020 blog post by Ivan Pesin (Pesin Space) noted that disk and network latencies have improved thanks to better SSDs (NVMe) and faster networks (10/100Gb), but core CPU latencies like L1 cache access remain stable. This uneven evolution highlights the importance of context in system design.

In practice, these numbers guide several aspects: - **Performance Optimization**: Minimizing operations with high latency, like disk seeks (10 ms), can significantly improve application speed. For example, caching frequently accessed data in memory (250 us for 1 MB read) rather than disk can reduce wait times. - **Trade-off Decisions**: System designers often face choices, such as using in-memory caches versus databases. Knowing that a main memory reference (100 ns) is 200 times faster than an L1 cache reference (0.5 ns) can inform such decisions. - **User Experience**: In web applications, network latencies, like a datacenter round trip (500 us), can affect page load times, impacting user satisfaction. A Vercel blog post from 2024 (Vercel Blog) emphasized this for frontend development, noting how network waterfalls can compound latency.

Historical Context and Updates The original numbers, attributed to Jeff Dean and popularized by Peter Norvig, date back to around 2010, with updates by researchers like Colin Scott (Interactive Latencies). A 2019 Medium post by Dan Hon (Dan Hon Medium) added humorous but relevant latencies, like rebooting a MacBook Pro (90 seconds), illustrating broader tech-related delays. However, core latency numbers have seen minimal changes, with the GitHub Gist suggesting they remain “quite similar”as of 2023, based on physical limitations.

Conclusion and Recommendations For programmers and system designers, memorizing these latency numbers provides a mental model for performance tuning. They should be treated as guidelines, with actual benchmarks conducted for specific hardware. Keeping abreast of updates, especially in emerging technologies like quantum computing or 5G networks, will be crucial. Resources like the GitHub Gist and Medium article offer starting points for further exploration.

This analysis, grounded in the video's likely content and supplemented by extensive online research, underscores the enduring relevance of latency numbers in computing, with a call to adapt to technological shifts for optimal system design.

Key Citations

- Latency Numbers Every Programmer Should Know GitHub Gist
- Latency Numbers Programmer Should Know YouTube Video
- Updated Latency Numbers Medium Article
- More Numbers Every Awesome Programmer Must Know High Scalability
- Latency Numbers Every Web Developer Should Know Vercel Blog
- Latency Numbers Every Engineer Should Know Pesin Space Blog
- More Latency Numbers Every Programmer Should Know Dan Hon Medium
- Numbers Every Programmer Should Know By Year Interactive Latencies