Instantly share code, notes, and snippets.

jboner / latency.txt

Last active yesterday

☆ Star

<> Code   Revisions 18   ☆ Stars 5,000+   Forks 2,145

Latency Numbers Every Programmer Should Know

latency.txt

```
Latency Comparison Numbers (~2012)
----------------------------------
L1 cache reference                           0.5 ns
Branch mispredict                            5   ns
L2 cache reference                           7   ns                      14x L1 cache
Mutex lock/unlock                           25   ns
Main memory reference                      100   ns                      20x L2 cache, 200x L1 cac
Compress 1K bytes with Zippy             3,000   ns        3 us
Send 1K bytes over 1 Gbps network       10,000   ns       10 us
Read 4K randomly from SSD*             150,000   ns      150 us          ~1GB/sec SSD
Read 1 MB sequentially from memory     250,000   ns      250 us
Round trip within same datacenter      500,000   ns      500 us
Read 1 MB sequentially from SSD*     1,000,000   ns    1,000 us    1 ms  ~1GB/sec SSD, 4X memory
Disk seek                           10,000,000   ns   10,000 us   10 ms  20x datacenter roundtrip
Read 1 MB sequentially from disk    20,000,000   ns   20,000 us   20 ms  80x memory, 20X SSD
Send packet CA->Netherlands->CA    150,000,000   ns  150,000 us  150 ms

Notes
-----
1 ns = 10^-9 seconds
1 us = 10^-6 seconds = 1,000 ns
1 ms = 10^-3 seconds = 1,000 us = 1,000,000 ns

Credit
------
By Jeff Dean:               http://research.google.com/people/jeff/
Originally by Peter Norvig: http://norvig.com/21-days.html#answers

Contributions
-------------
'Humanized' comparison:  https://gist.github.com/hellerbarde/2843375
Visual comparison chart: http://i.imgur.com/k0t1e.png
```

Load earlier comments...

haai commented on Sep 4, 2019

interesting when you see in a glance. but would't it be good to use one unit in the comparison e.g. memory page 4k?

acuariano commented on Sep 11, 2019

> Nanoseconds

It's an excellent explanation. I had to search the video because the account was closed. Here's the result I got: https://www.youtube.com/watch?v=9eyFDBPk4Yw

KevinZhou92 commented on Jan 30, 2020

`Send 1K bytes over 1 Gbps network       10,000   ns       10 us`
This doesn't look right to me. 1 Gbps = 125, 000 KB/s, the time should be 1 / 125,000 = 8 * 10^-6 seconds which is 8000ns

andaru commented on Apr 4, 2020

> `Send 1K bytes over 1 Gbps network 10,000 ns 10 us`
> This doesn't look right to me. 1 Gbps = 125, 000 KB/s, the time should be 1 / 125,000 = 8 * 10^-6 seconds which is 8000ns

For a direct host-to-host connection with 1000BaseT interfaces, a wire latency of 8μs is correct.

However, if the hosts are connected using SGMII, the Serial Gigabit Media Independent Interface, data is 8b10b encoded, meaning 10 bits are sent for every 8 bits of data, leading to a latency of 10μs.

Jeff may also have been referring to the fact that in a large cluster you'll have a few switches between the hosts, so even where 1000BaseT is in use, the added switching latency (even for switches operating in cut-through mode) for, say, 2 switches can approach 2μs.

In any event, the main thing to take away from these numbers are the orders of magnitude differences between latency for various methods of I/O.

arunkumaras10 commented on May 20, 2020

Are these numbers still relevant in 2020? Or this needs an update?

maning711 commented on Jun 9, 2020

> Are these numbers still relevant in 2020? Or this needs an update?

I think hardwares are so expensive that can't update them~

vladimirvs commented on Jul 21, 2020

One thing that is misleading is that different units are used for send over 1Gbps versus read 1 MB from RAM. RAM is at least x20 times faster, but it ranks below send over network which is misleading. They should have used the same 1MB for network and RAM.

amresht commented on Aug 6, 2020 • edited

> need a solar system type visualization for this, so we can really appreciate the change of scale.

Hi
I liked your request and made an comparison. One unit is Mass of earth not radius.

Operation	Time in Nano Seconds	Astronomical Unit of Weight
L1 cache reference	0.5 ns	1/2 Earth or Five times Mars
Branch mispredict	5 ns	5 Earths
L2 cache reference	7 ns	7 Earths
Mutex lock/unlock	25 ns	Roughly [Uranus +Neptune]
Main memory reference	100 ns	Roughly Saturn + 5 Earths
Compress 1K bytes with Zippy	3,000 ns	10 Jupiters
Send 1K bytes over 1 Gbps network	10,000 ns	20 Times All the Planets of the Solar System
Read 4K randomly from SSD*	150,000 ns	1.6 times Red Dwarf Wolf 359
Read 1 MB sequentially from memory	250,000 ns	Quarter of the Sun
Round trip within same datacenter	500,000 ns	Half of the Mass of Sun
Read 1 MB sequentially from SSD*	1,000,000 ns	Sun
Disk seek	10,000,000 ns	10 Suns

Operation	Time in Nano Seconds	Astronomical Unit of Weight
Read 1 MB sequentially from disk	20,000,000 ns	Red Giant R136a2
Send packet CA->Netherlands->CA	150,000,000 ns	An Intermediate Sized Black Hole

https://docs.google.com/spreadsheets/d/13R6JWSUry3-TcCyWPbBhD2PhCeAD4ZSFqDJYS1SxDyc/edit?usp=sharing

asimilon commented on Oct 4, 2020

> > need a solar system type visualization for this, so we can really appreciate the change of scale.
>
> Hi
> I liked your request and made an comparison. One unit is Mass of earth not radius.

For me the best way of making this "more human relatable" would be to treat nanoseconds as seconds and then convert the large values.

eg. 150,000,000s = ~4.75 years

sirupsen commented on Jan 9, 2021

I've been doing some more work inspired by this, surfacing more numbers, and adding throughput:

https://github.com/sirupsen/napkin-math

sachin-j-joshi commented on Mar 28, 2021

Is there a 2021 updated edition?

ellingtonjp commented on Apr 16, 2021 • edited

@sirupsen I love your project and I'm signed up for the newsletter. Currently making Anki flashcards :)

There are some large discrepancies between your numbers and the ones found here (not sure where these numbers came from):
https://colin-scott.github.io/personal_website/research/interactive_latency.html

I'm curious what's causing them. Specifically, 1MB sequential memory read: **100us** vs **3us**.

sirupsen commented on Apr 16, 2021

@ellingtonjp My program is getting ~100 us, and this one says 250 us (from 2012). Lines up to me with some increases in performance since :) Not sure how you got 3 us

ellingtonjp commented on Apr 16, 2021 • edited ▾

@sirupsen I was referring to the numbers here https://colin-scott.github.io/personal_website/research/interactive_latency.html

The 2020 version of "Read 1,000,000 bytes sequentially from memory" shows 3us. Not sure where that comes from though. Yours seems more realistic to me

sirupsen commented on Apr 17, 2021 • edited ▾

Ahh, sorry I read your message too quick. Yeah, unclear to me how someone would get 3us. The code I use for this is very simple. It took reading the x86 a few times to ensure that the compiler didn't optimize it out. I do summing, which is one of the lightest workloads you could do in a loop like that. So I think it's quite realistic. Maybe that person's script it was optimized out? 🤷

ellingtonjp commented on Apr 17, 2021

To everyone interested in numbers like this:

@sirupsen 's project is *really* good. He gave an excellent talk on the "napkin math" skill and has a newsletter with monthly challenges for practicing putting these numbers to use.

Newsletter: https://sirupsen.com/napkin/
Github: https://github.com/sirupsen/napkin-math
Talk: https://www.youtube.com/watch?v=IxkSlnrRFqc

awsles commented on Jun 9, 2021

:)

```
Light to reach the moon      2,510,000,000  ns  2,510,000 us  2,510 ms 2.51 s
```

invisiblethings commented on Nov 24, 2021 • edited ▾

Heh, imagine this transposed into human distances.

1ns = 1 step, or 2 feet.

L1 cache reference = reaching 1 foot across your desk to pick something up
Datacentre roundtrip = 94 mile hike.
Internet roundtrip (California to Netherlands) = Walk around the entire earth. Wait! You're not done. Then walk from London, to Havana. Oh, and then to Jacksonville, Florida. Then you're done.

apimaker001 commented on Dec 23, 2021

useful information & thanks

eduard93 commented on Jan 4, 2022

What about register access timings?

crazydogen commented on Apr 6, 2022 • edited

> Markdown version :p

Operation	ns	µs	ms	note
L1 cache reference	0.5 ns			
Branch mispredict	5 ns			
L2 cache reference	7 ns			14x L1 cache
Mutex lock/unlock	25 ns			
Main memory reference	100 ns			20x L2 cache, 200x L1 cache
Compress 1K bytes with Zippy	3,000 ns	3 µs		
Send 1K bytes over 1 Gbps network	10,000 ns	10 µs		
Read 4K randomly from SSD*	150,000 ns	150 µs		~1GB/sec SSD
Read 1 MB sequentially from memory	250,000 ns	250 µs		
Round trip within same datacenter	500,000 ns	500 µs		
Read 1 MB sequentially from SSD*	1,000,000 ns	1,000 µs	1 ms	~1GB/sec SSD, 4X memory

Operation	ns	µs	ms	note
Disk seek	10,000,000 ns	10,000 µs	10 ms	20x datacenter roundtrip
Read 1 MB sequentially from disk	20,000,000 ns	20,000 µs	20 ms	80x memory, 20X SSD
Send packet CA -> Netherlands -> CA	150,000,000 ns	150,000 µs	150 ms	

LuisOsta commented on Aug 20, 2022

@jboner What do you think about adding cryptography numbers to the list? I feel like that would be a really valuable addition to the list for comparison. Especially as cryptography usage increases and becomes more common.

We could for instance add Ed25519 latency for cryptographic signing and verification. In a very rudimentary testing I did locally I got:

1. Ed25519 Signing - 254.20µs
2. Ed25519 Verification - 368.20µs

You can replicate the results with the following rust program:

```
fn main() {
    println!("Hello, world!");
    let msg = b"lfasjhfoihjsofh438948hhfklshfosiuf894y98s";
    let sk = ed25519_zebra::SigningKey::new(rand::thread_rng());

    let now = std::time::Instant::now();
    let sig = sk.sign(msg);
    println!("{:?}", sig);
    let elapsed = now.elapsed();
    println!("Elapsed: {:.2?}", elapsed);

    let vk = ed25519_zebra::VerificationKey::from(&sk);
    let now = std::time::Instant::now();
    vk.verify(&sig, msg).unwrap();
    let elapsed = now.elapsed();
    println!("Elapsed: {:.2?}", elapsed);
}
```

bob333 commented on Sep 15, 2022

What is "Zippy"? Is it a google internal compression software?

Yrwein commented on Oct 5, 2022

@bob333 https://en.wikipedia.org/wiki/Snappy_(compression)

milesrichardson commented on Nov 4, 2022

> Send 1K bytes over 1 Gbps network 10,000 ns 10 us

this seems misleading, since in common networking terminology 1 Gbps refers to throughput ("size of the pipe"), but this list is about "latency," which is generally independent of throughput - it takes the same amount of time to send 1K bytes over a 1 Mbps network and a 1 Gbps network

A better description of this measure sounds like "bit rate," or more specifically the "data signaling rate" (DSR) over some communications medium (like fiber). This also avoids the ambiguity of "over" the network (how much distance?) because DSR measures "aggregate rate at which data passes a *point*" instead of a segment.

Using this definition (which I just learned a minute ago), perhaps a better label would be:

```
- Send 1K bytes over 1 Gbps network       10,000   ns       10 us
+ Transfer 1K bytes over a point on a 1 Gbps fiber channel       10,000   ns       10 us
```

🤷 (also, I didn't check if the math is consistent with this labeling, but I did pull "fiber channel" from the table on the DSR wiki page)

nking commented on Jun 9, 2023

Thanks for sharing your updates.

You could consider adding a context switch for threads right under disk seek:
computer context switches: 1e7 ns

VTrngNghia commented on Dec 19, 2023

I see "Read 1 MB sequentially from disk", but how about disk write?

SergeSEA commented on Dec 20, 2023 • edited ▾

the numbers are from Dr. Dean from Google reveals the length of typical computer operations in 2010. I hope someone could update them as it's 2023

VTrngNghia commented on Dec 20, 2023

The numbers should be still quite similar.

These numbers based on Physical limitation only significant technological leap can make a difference.

In any case, these are for estimates, not exact calculation. For example, 1MB read from SSD is different for each SSD, but it should be somewhere around the Millisecond range.

xealits commented last month • edited

it could be useful to add a column with the sizes in the hierarchy. Also, a column of the minimal memory units sizes, the cache line sizes etc. Then you can also divide the sizes by the latencies, which would be some kind of limit for a simple algorithm throughput. Not really sure if this is useful though.