Instantly share code, notes, and snippets.

hellerbarde / latency.markdown

Forked from jboner/latency.txt
Created May 31, 2012

Code · Revisions 16 · Stars 3,942 · Forks 678

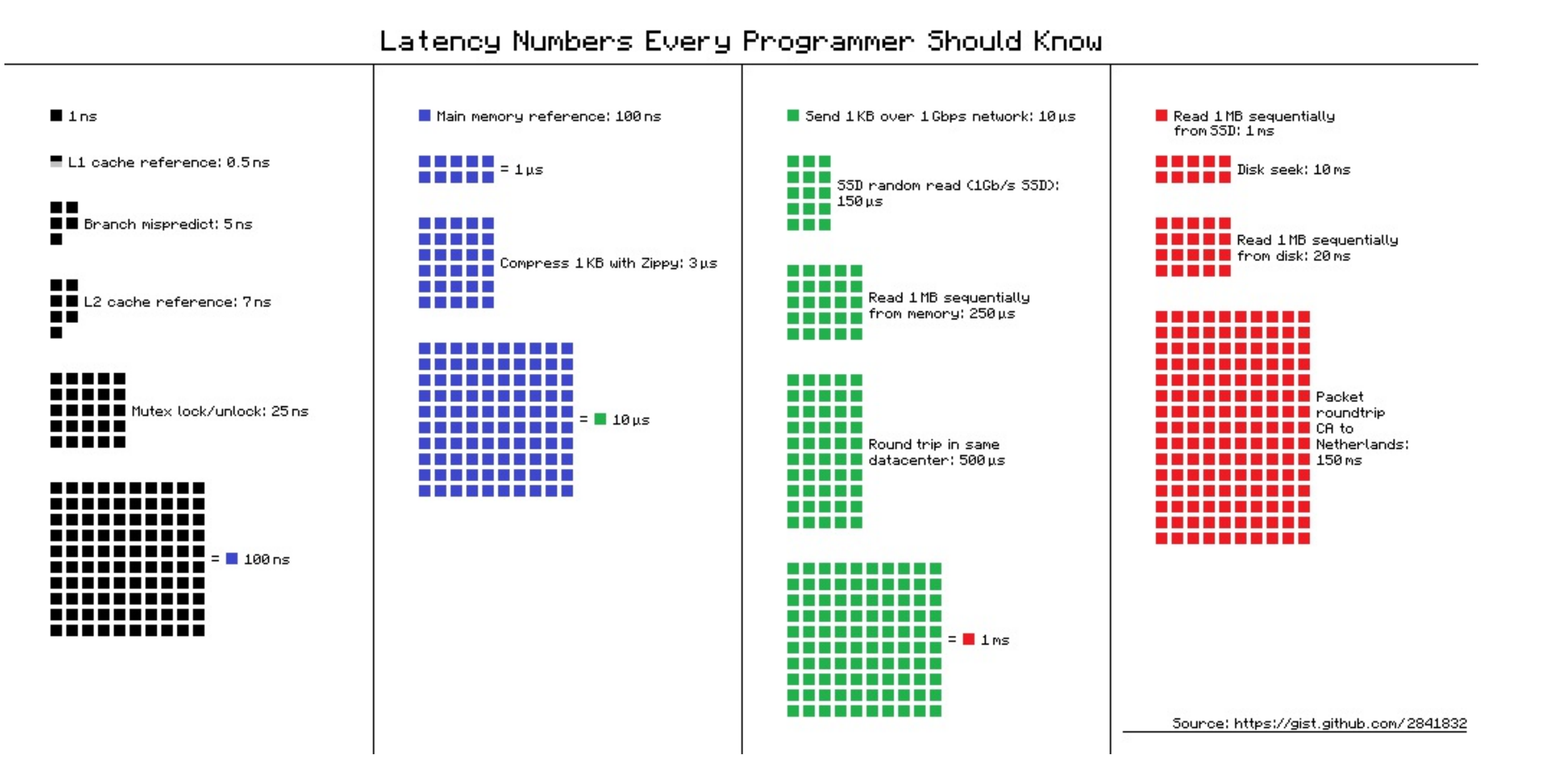
Latency numbers every programmer should know

latency.markdown

Latency numbers every programmer should know

```
L1 cache reference ......................... 0.5 ns
Branch mispredict ............................ 5 ns
L2 cache reference ........................... 7 ns
Mutex lock/unlock ........................... 25 ns
Main memory reference ...................... 100 ns
Compress 1K bytes with Zippy ............. 3,000 ns  =   3 µs
Send 2K bytes over 1 Gbps network ....... 20,000 ns  =  20 µs
SSD random read ........................ 150,000 ns  = 150 µs
Read 1 MB sequentially from memory ..... 250,000 ns  = 250 µs
Round trip within same datacenter ...... 500,000 ns  = 0.5 ms
Read 1 MB sequentially from SSD* ..... 1,000,000 ns  =   1 ms
Disk seek ........................... 10,000,000 ns  =  10 ms
Read 1 MB sequentially from disk .... 20,000,000 ns  =  20 ms
Send packet CA->Netherlands->CA .... 150,000,000 ns  = 150 ms
```

Assuming ~1GB/sec SSD



Visual chart provided by ayshen

Data by Jeff Dean

Originally by Peter Norvig

latency_humanized.markdown

Lets multiply all these durations by a billion:

Magnitudes:

Minute:

```
L1 cache reference                  0.5 s         One heart beat (0.5 s)
Branch mispredict                   5 s           Yawn
L2 cache reference                  7 s           Long yawn
Mutex lock/unlock                   25 s          Making a coffee
```

Hour:

```
Main memory reference               100 s         Brushing your teeth
Compress 1K bytes with Zippy        50 min        One episode of a TV show (including ad breaks)
```

Day:

```
Send 2K bytes over 1 Gbps network   5.5 hr        From lunch to end of work day
```

Week

```
SSD random read                     1.7 days      A normal weekend
Read 1 MB sequentially from memory  2.9 days      A long weekend
Round trip within same datacenter   5.8 days      A medium vacation
Read 1 MB sequentially from SSD    11.6 days      Waiting for almost 2 weeks for a delivery
```

Year

```
Disk seek                           16.5 weeks    A semester in university
Read 1 MB sequentially from disk    7.8 months    Almost producing a new human being
The above 2 together                1 year
```

Decade

```
Send packet CA->Netherlands->CA     4.8 years     Average time it takes to complete a bachelor's degree
```

Load earlier comments...

milesrout commented Jun 19, 2014

@legrady it depends on a lot of factors. Is it a virtual function call?

coolearn commented Aug 18, 2014

大神

AdamBSteele commented Dec 8, 2014

If reading 1MB from an SSD costs 1ms, what would the cost be to read 10MB sequentially from an SSD?

b1nary commented Dec 8, 2014

This is a great collection. I just dont get where or how i am able to make coffee in just 25s.

stultus commented Dec 8, 2014

Agree @b1nary . if someone knows how to do that, please share the source code 😄

jeveloper commented Dec 8, 2014

That would be a shocker if devops status page turned into humanized numbers one day (sometime in april).
We should all start working harder to improve our numbers ! and enjoy more Round trip within same datacenter 😃

benibela commented Dec 8, 2014

Do not forget:

3ms: Time till a wrongly configured sendmail timeouts and fails to deliver a mail. Roughly corresponds to mail servers in a 500km (3 millilightseconds) radius

6h: Time to send a mail across those 500km via RFC 1149

caimaoy commented Jan 5, 2015

cool

hellerbarde commented Apr 24, 2015

@stultus @b1nary we have a coffee machine that makes coffee. Ta-Dah! 😄

GreatmanBill commented Apr 28, 2015

good, it's cool!

villadora commented Apr 15, 2016

cool! great summary

susingha commented Oct 9, 2016

this is awesome. Thank you

marianposaceanu commented Oct 9, 2016 • edited

hmm:

> branch misprediction penalty on Haswell ~ 1500 ns vs 5 ns in the gist. That's three orders of magnitude of error

EDIT:

I used the ticks from Windows (are 10K in a ms) which is incorrect related to the gist.

If the Haswell CPU is running 3.6Ghz - one cycle would equal to 0.27ns that would mean a branch miss would be 4.05ns - seems about right now.

rr-paras-patel commented Oct 11, 2016

cool..... thank you

Kevin-Hamilton commented Oct 11, 2016

Multiplying by a billion stretches the timescales out too much for my taste. So I came up with an alternate list based on multiplying by only 22,000:

```
L1 cache reference .................  0.000011 seconds (SR-71 travels 1cm)
Branch mispredict ..................  0.000110 sec (Bullet travels 4cm)
L2 cache reference .................  0.000154 sec (Boeing 777 travels 4cm)
Mutex lock/unlock ..................  0.00055 sec (Time before you hear a fingersnap made in front of your face [speed of sound across 19cm])
Main memory reference ..............  0.0022 sec (Camera shutter on a sunny day [1/400 - 1/500 shutter speed])
Compress 1K bytes with Zippy ........  0.066 sec (Lightning bolt travels 4km from cloud to ground)
Send 2K bytes over 1 Gbps network ...  0.44 sec (Fastball from pitcher to home plate)
SSD random read ....................  3.3 sec (SR-71 travels 3.1km)
Read 1 MB sequentially from memory ..  5.5 sec (Yawn)
Round trip within same datacenter ... 11.0 sec (A Cheetah runs 200m)
Read 1 MB sequentially from SSD* .... 22.0 sec (Usain Bolt runs 200m)
Disk seek ..........................  3.6 minutes (Brewing coffee in a French Press)
Read 1 MB sequentially from disk ....  7.3 min (A performance of the first movement of Beethoven's 5th Symphony)
Send packet CA->Netherlands->CA ..... 55.0 min (Going for a brisk 5km walk)
```

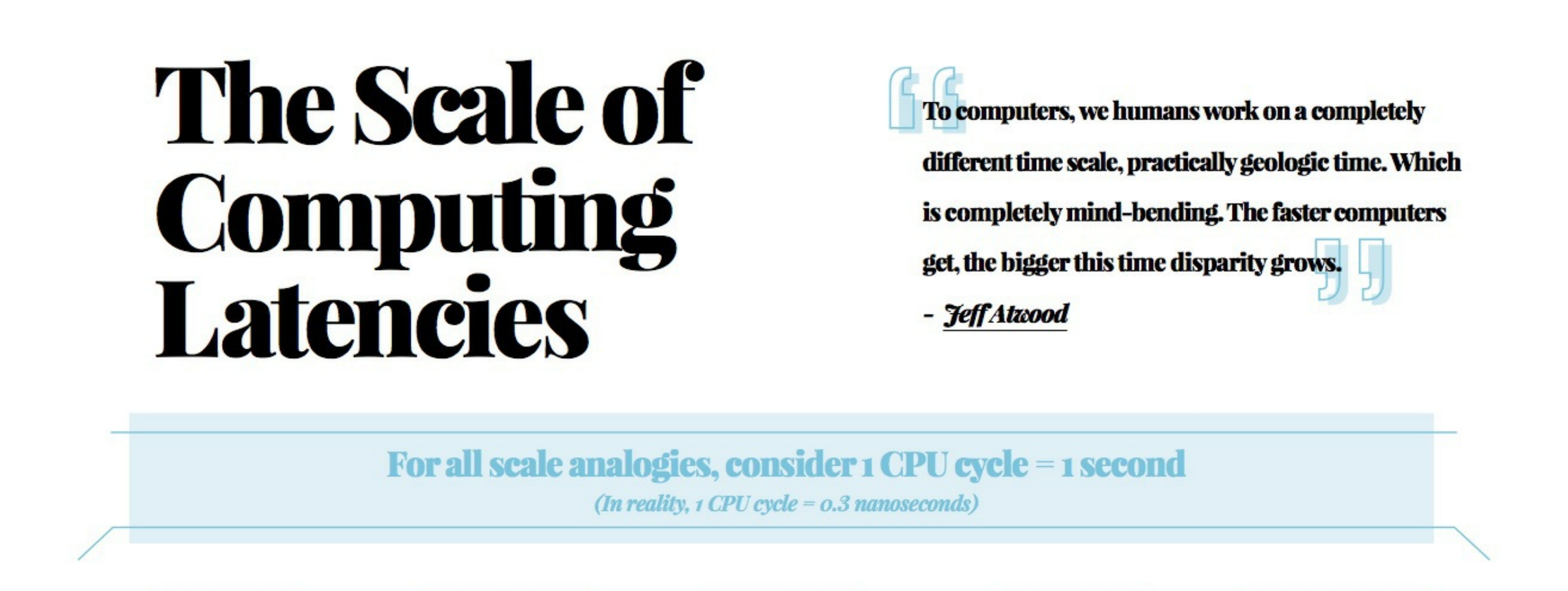
MartyGentillon commented Oct 12, 2016

@Kevin-Hamilton There is a reason to stretch it out that much. From a human perspective, it is really hard to do anything in less than a second. As such, the ridiculously long times give you a better idea of what a computer might be able to do during that disk seek, if it weren't waiting for that disk seek.

Because of this, most of the similar pages I have seen use something like 1 second for a clock cycle (so multiply everything there by 3 or 4). It gives a really good sense of machine sympathy.

mpron commented Oct 12, 2016

Last year, I came up with this concept for an infographic illustrating these latency numbers with time analogies (if 1 CPU cycle = 1 second). Here was the result (attached, and here's a link: http://imgur.com/8LIwV4C)



cth027 commented Nov 19, 2016

Excellent idea ! Great page !
Perhaps an interesting comparison:

- the human eye requires 13 ms to identify an image
- an eye blink is around 100 ms
- the reaction time from eye to mouse is around 215 ms

MAZHARMIK commented Dec 30, 2016

Cool. Loved it.

hhimanshu commented Jan 22, 2017

very interesting!

imonti commented Mar 31, 2017

Excelent Gist.

LeonZhu1981 commented Feb 5, 2019

great!!!

YLD10 commented Jul 9, 2019

Thanks ^o^

vinaypuranik commented Jul 9, 2019

Awesome gist! Thanks

xenowits commented Oct 27, 2019

wowww!!

vapniks commented Nov 14, 2019

Here's a nanosecond: https://www.youtube.com/watch?v=9eyFDBPk4Yw

jiteshk23 commented Nov 14, 2019

These numbers seem old. This page is updated : https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html

Code2Life commented Aug 24, 2020

cool!

eduard93 commented Jan 3, 2022

What about register access timings?

hellerbarde commented Jan 6, 2022 • edited

@eduard93 I think register access happens within one CPU cycle. Which, at 2.4 GHz would be 0.417 nanoseconds, which is very similar to the L1 cache reference. I'm not sure if that's true, because I'm not incredibly familiar with modern CPUs. Feel free to fact check this.