Go Is Boring

Jon Bodner

November 13th, 2020

Go Is Boring

AND THAT’S FANTASTIC

Prolog

C++

Snobol

Java

Rexx

Ruby

HyperTalk

AppleScript

Python

Objective-C

Visual Basic

FORTRAN

Pascal

Lisp

Logo

JavaScript

6502 Assembly

C

Groovy

x86 Assembly

Go

SQL

DBase IV

Commodore BASIC

Scala


commodore
2001 series
Personal Electronic Transactor


ANKER
PD

C and Pascal

- Structured Programming
- The “joys” of manual memory management

C++

- Object-Oriented Programming
  - Inheritance
  - Abstract Data Types
- Public/Private/Friends
- Operator Overloading
- Templates

C++
But Wait, There's More

- Type Inference
- For/Range Loops
- Lambda Functions
- New Function Syntax
- Strongly Typed Enums
- Unicode Strings
- Threadlocals

Java

- Virtual Machine for portability
- OO can actually make sense
- Garbage collectors are great
  - The more, the merrier!
- Threads
- Exceptions

Java 2: The Javaning

- Generics
- Enumerations
- For/Range Loop
- Annotations
- Aspect-Oriented Programming
- Try with resources

Java 3: Please Make It Stop

- Type Inference for Generics (<>)
- Modules
- Lambdas
- java.util.Timer, javax.swing.Timer, java.util.concurrent.ScheduledThreadPoolExecutor
- Type Inference, take 2
- Functional Switch
- Records
- Value types?

JavaScript

- Classes
- Modules
- Iterators
- Generators
- Constants
- New Scoping Rules for Let
- Promises
- Reflection
- Async/Await
- Spread Operators

Python

- Formatted string literals
- Underscores in numbers
- Data classes
- Type annotations
- async/await
- := operator

Rust

```
use std::slice;

fn split_at_mut(slice: &mut [i32], mid: usize) -> (&mut [i32], &mut [i32]) {
    let len = slice.len();
    let ptr = slice.as_mut_ptr();

    assert!(mid <= len);

    unsafe {
        (slice::from_raw_parts_mut(ptr, mid),
         slice::from_raw_parts_mut(ptr.offset(mid as isize), len - mid))
    }
}
```

Swift

- Automatic Reference Counting
- Inheritance
- Exceptions
- Generics
- guard
- subscripts
- lazy properties
- computed properties

Programming Languages Grow Features


Virtual Machine


Exceptions


Inheritance


Overloading


AVANCIS


Immutables


Enumerations

*yet

Garbage Collector

- One algorithm
- Two parameters

Control Structures

- if
- for
- switch
- goto


M
Style 3652
White
1
N
Style 5339 White
M
N
Plaidstallions

Access Controls

- Exported from Package
- Not Exported from Package

Concurrency

- goroutines
- channels
- select


commodore
2001 series
Personal Electronic Transactor

Why is everyone interested in this boring language?

MECHANIZATION

OF

CONTRACT

ADMINISTRATION

SERVICES

What is MOCAS?

Integrated system supporting post-award administration and contract payment

Who uses MOCAS?

DCMA

DFAS

Procurement Offices

DCAA

DTRA

WHS

Why MOCAS?

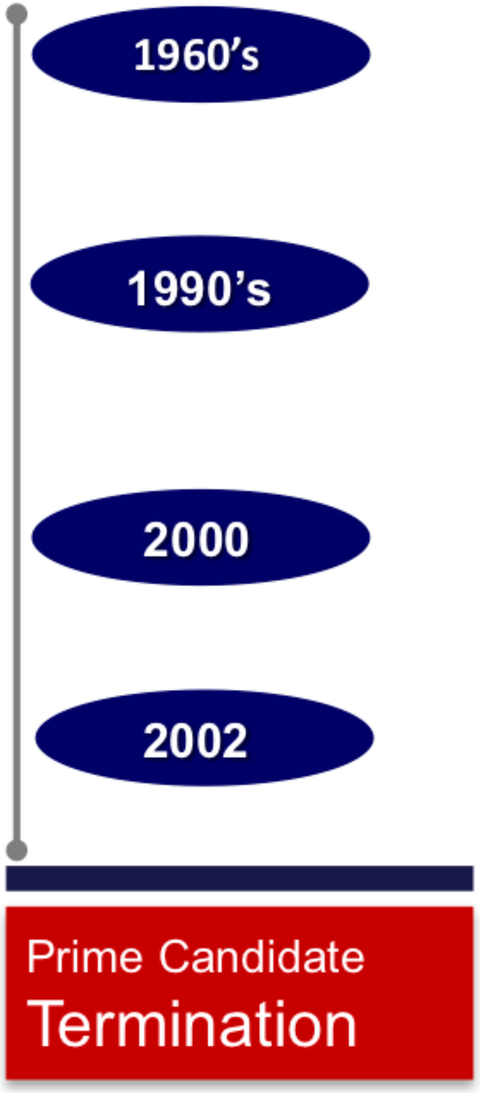
Pays More Complex Contracts

- ✓ Mixed-Type
- ✓ Multi-Service
- ✓ Multi-Year
- ✓ Multi-Deliverables
- ✓ Foreign Military Sales
- ✓ Foreign Currencies

Makes Financing Payments

- ✓ Progress Payments
- ✓ Performance-Based Payments
- ✓ Commercial Item Financing
- ✓ Interim Cost Payments

MOCAS – System Timeline



- In 1960s, MOCAS designed as a joint service single accounting and contract system.
- By 1990s, MOCAS was more than 30 years old and had become costly to maintain.
- March 23, 2000 memo announced that October 1, 2002 scheduled retirement date for MOCAS.
- August 29, 2000, a conversion/closeout integrated process team (IPT) was assembled.
- January 01, 2001 began the interim period known as the MOCAS "brownout" (Legacy System).
- **October 1, 2002 scheduled retirement date for MOCAS.**

https://xkcd.com/2030/


Google

Photo by Daniel McCullough on Unsplash

Photo by Cody Hiscox on Unsplash

Photo by Alex Azabache on Unsplash

Photo by Silas Baisch on Unsplash

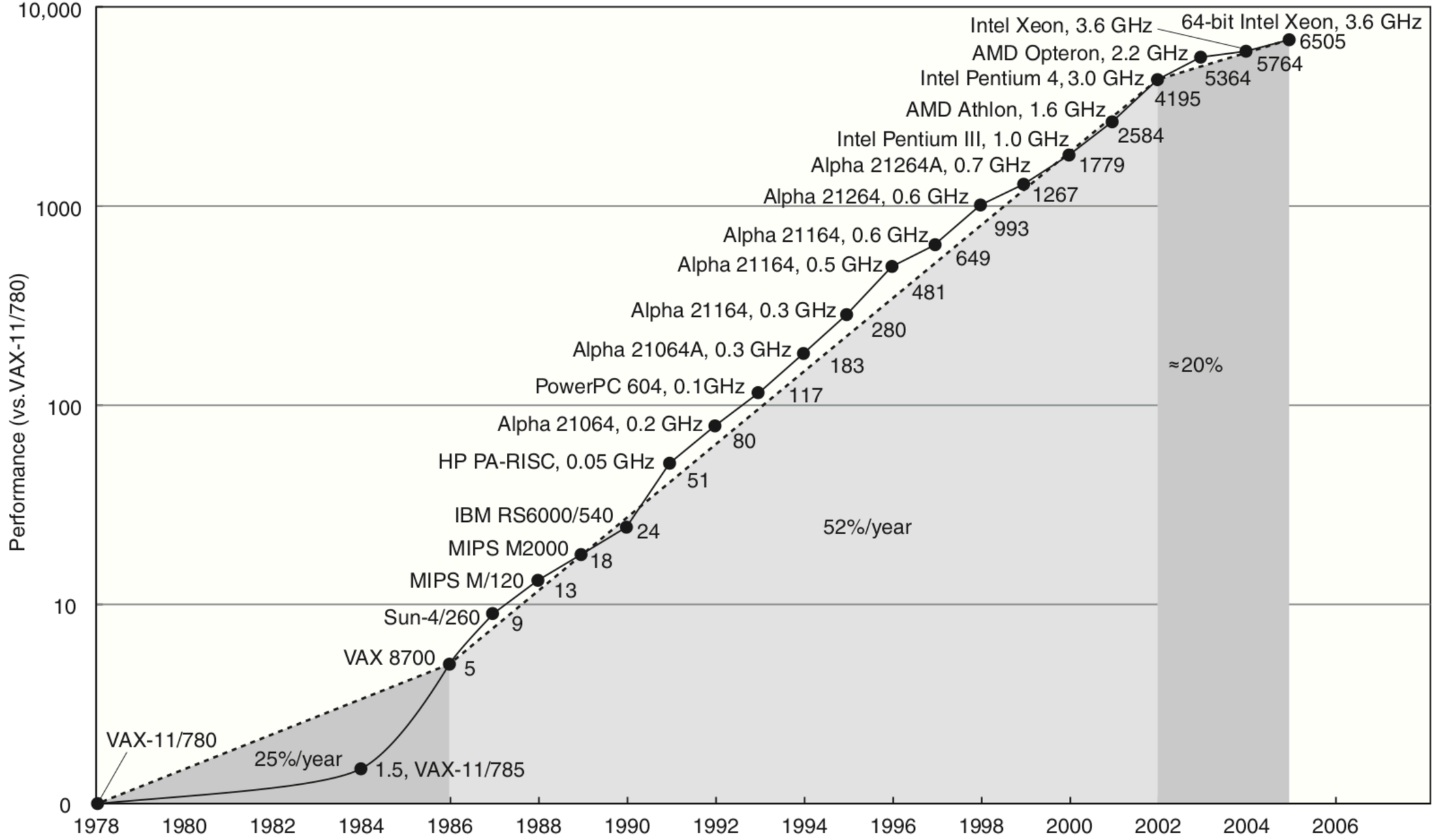
What do we know about writing software?

IT'S BETTER TO FIND PROBLEMS EARLY

Manual memory management doesn't work

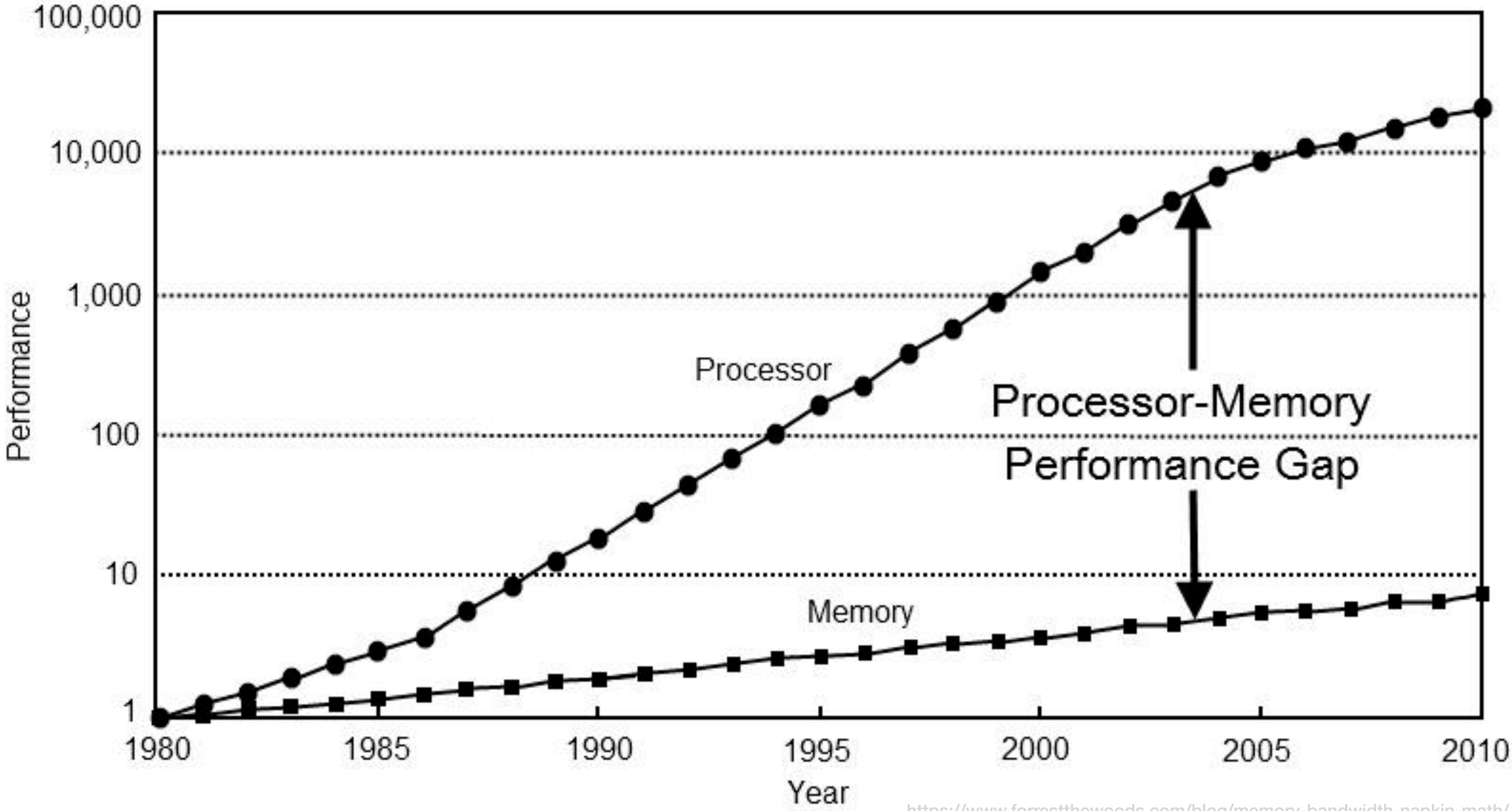
CODE REVIEWS HELP

COMMUNICATION COSTS DOMINATE

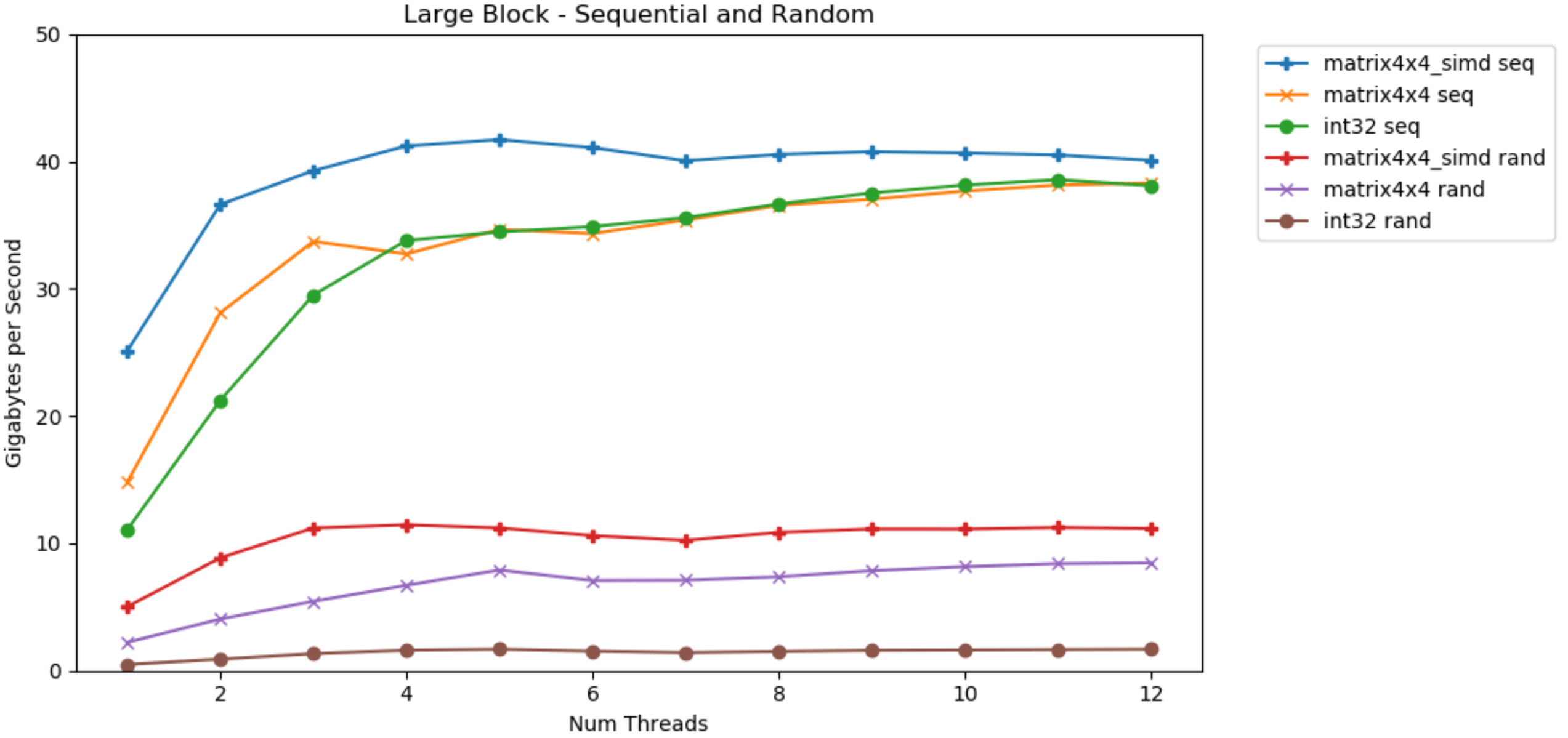


Not So Fast…

Intel Core i7-3960X	Intel Core i9-9900KS
Released Q4 2011	**Released Q4 2019**
3.3 GHz (6 cores)	4.0 GHz (8 cores)
814 Single Core	1414 Single Core
4808 Multi Core	9498 Multi Core



https://www.forrestthewoods.com/blog/memory-bandwidth-napkin-math/



1 Thread	matrix4x4	unique_ptr	diff
Large Block - Seq	14.8 GB/s	0.8 GB/s	19x
16 KB - Seq	31.6 GB/s	2.2 GB/s	14x
256 KB - Seq	22.2 GB/s	1.9 GB/s	12x
Large Block - Rand	2.2 GB/s	0.1 GB/s	22x
16 KB - Rand	23.2 GB/s	1.7 GB/s	14x
256 KB - Rand	15.2 GB/s	0.8 GB/s	19x

6 Threads	matrix4x4	unique_ptr	diff
Large Block - Seq	34.4 GB/s	2.5 GB/s	14x
16 KB - Seq	154.8 GB/s	8.0 GB/s	19x
256 KB - Seq	111.6 GB/s	5.7 GB/s	20x
Large Block - Rand	7.1 GB/s	0.4 GB/s	18x
16 KB - Rand	95.0 GB/s	7.8 GB/s	12x
256 KB - Rand	58.3 GB/s	1.6 GB/s	36x

Let's take another look at Go

Photo by Ashim D'Silva on Unsplash


Photo by Gary Chan on Unsplash

Photo by Kelly Sikkema on Unsplash

Photo by Kelly Sikkema on Unsplash


QUACK
QUACK QUACK


QUACK
QUACK QUACK

	Python	Ruby
Regex-redux	3659.78%	219.83%
mandelbrot	48.11%	6133.87%
fannkuch-redux	3312.66%	2060.82%
fasta	3015.64%	2438.86%
n-body	4191.53%	1770.56%
spectral-norm	4128.64%	3753.64%
pidigits	117.16%	157.35%
reverse-compliment	439.95%	852.01%
k-nucleotide	572.85%	837.17%
binary-trees	320.84%	177.33%

	Python	Ruby
Regex-redux	3659.78%	219.83%
mandelbrot	48.11%	6133.87%
fannkuch-redux	3312.66%	2060.82%
fasta	3015.64%	2438.86%
n-body	4191.53%	1770.56%
spectral-norm	4128.64%	3753.64%
pidigits	117.16%	157.35%
reverse-compliment	439.95%	852.01%
k-nucleotide	572.85%	837.17%
binary-trees	320.84%	177.33%

	Java	JavaScript
Regex-redux	142.44%	132.18%
mandelbrot	123.24%	123.24%
fannkuch-redux	107.97%	130.48%
fasta	105.21%	171.56%
n-body	102.78%	123.61%
spectral-norm	100.73%	106.80%
pidigits	89.71%	126.47%
reverse-compliment	87.67%	109.92%
k-nucleotide	72.14%	207.73%
binary-trees	32.87%	76.30%

	Java	JavaScript
Regex-redux	142.44%	132.18%
mandelbrot	123.24%	123.24%
fannkuch-redux	107.97%	130.48%
fasta	105.21%	171.56%
n-body	102.78%	123.61%
spectral-norm	100.73%	106.80%
pidigits	89.71%	126.47%
reverse-compliment	87.67%	109.92%
k-nucleotide	72.14%	207.73%
binary-trees	32.87%	76.30%


Photo by Ninety Eyes on Unsplash


Photo by Ninety Eyes on Unsplash

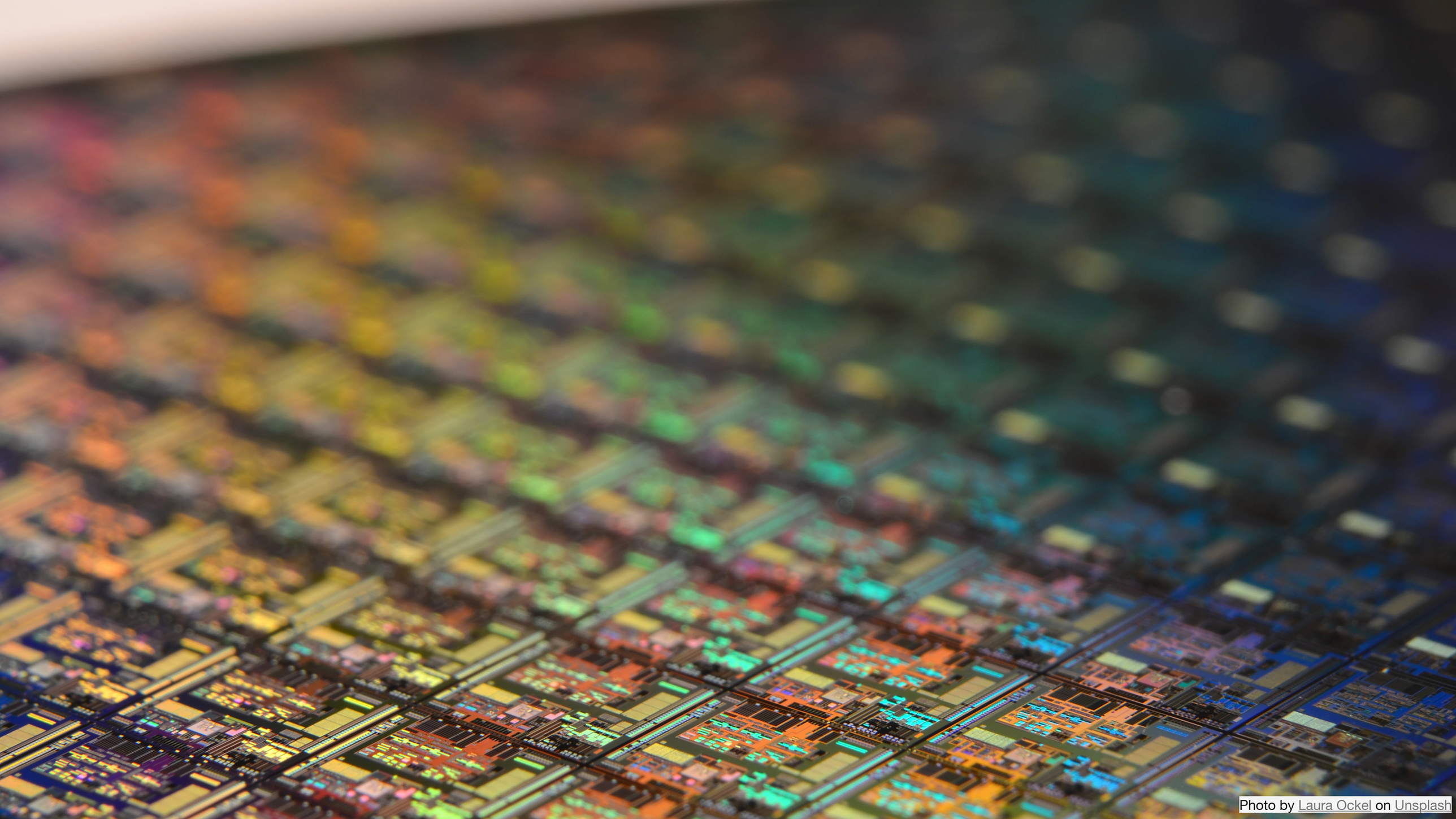


Photo by Laura Ockel on Unsplash

What about the features?

A Large Scale Study of Programming Languages and Code Quality in Github

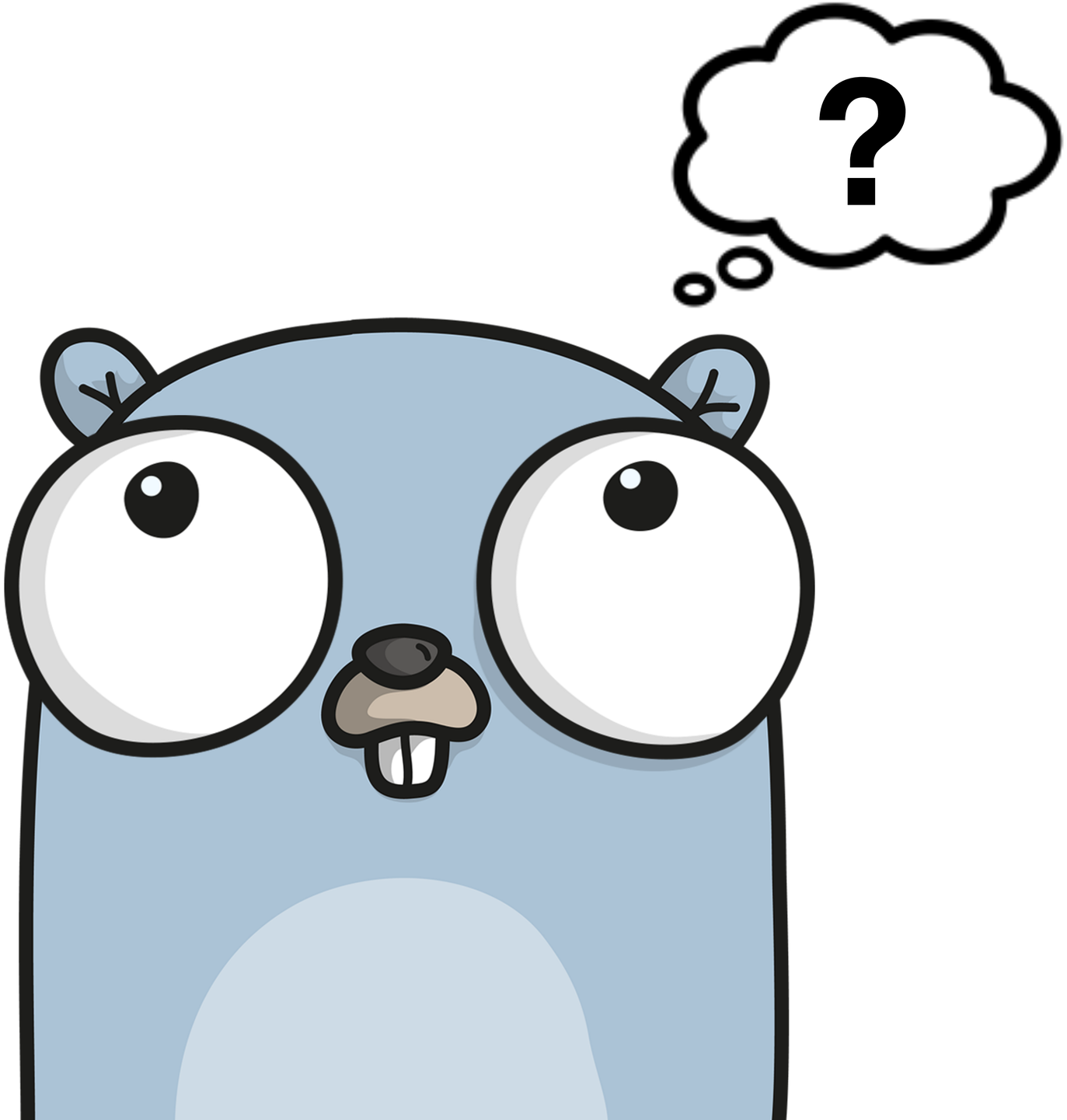
“It is worth noting that these modest effects arising from language design are overwhelmingly dominated by the process factors such as project size, team size, and commit size.”

https://web.cs.ucdavis.edu/~filkov/papers/lang_github.pdf

On the Impact of Programming Languages on Code Quality

“Not only is it not possible to establish a causal link between programming language and code quality based on the data at hand, but even their correlation proves questionable.”

https://arxiv.org/pdf/1901.10220.pdf



The generic dilemma is this: do you want slow programmers, slow compilers and bloated binaries, or slow execution times?

Russ Cox

History of Go Generics Proposals

- April 2016 - Go team publishes paper detailing 4 rejected proposals (https://go.googlesource.com/proposal/+/master/design/15292-generics.md)

History of Go Generics Proposals

- August 2018 - Draft document released describing contracts (https://go.googlesource.com/proposal/+/master/design/go2draft-generics-overview.md)

History of Go Generics Proposals

- July 2019 - Revised draft document on contracts (https://github.com/golang/proposal/blob/master/design/go2draft-contracts.md)

History of Go Generics Proposals

- June 2020 - Revised draft proposal using interfaces (https://go.googlesource.com/proposal/+/refs/heads/master/design/go2draft-type-parameters.md)

Go is boring

and

that's fantastic