CONCURRENCY IN RUBY 3

SF RUBY MEETUP — 08/03/2024

By Konstantin Gredeskoul

Staff Software Engineer, academia.edu

WHO AM I?

STUDIED MATHEMATICS & STATISTICS
RUBYIST SINCE 2007
50 GEMS WITH 140M DOWNLOADS
MAKING WEB APPS SINCE '1997

EXCTO @ WANELO (WAS A SOCIAL NETWORK FOR SHOPPING)
CURRENTLY @ ACADEMIA.EDU
AND WE ARE HIRING

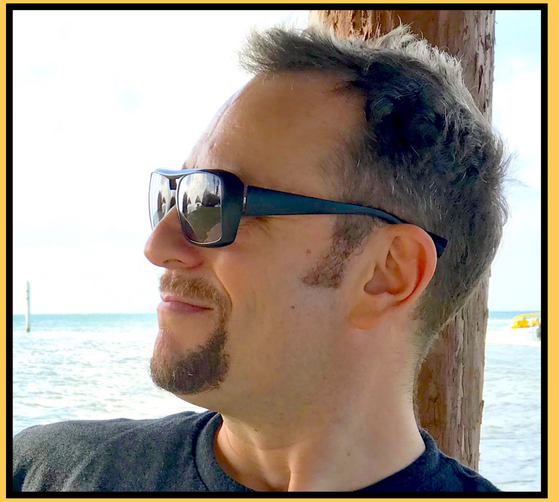
https://github.com/kigster

https://twitter.com/kig

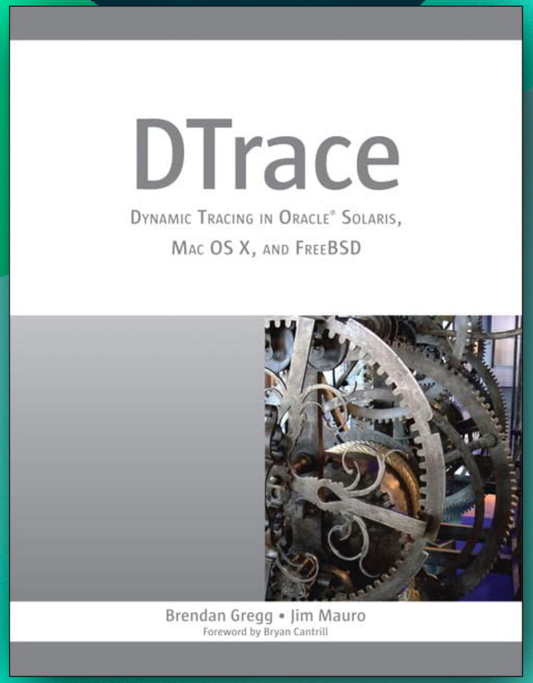
https://kig.re (blog)

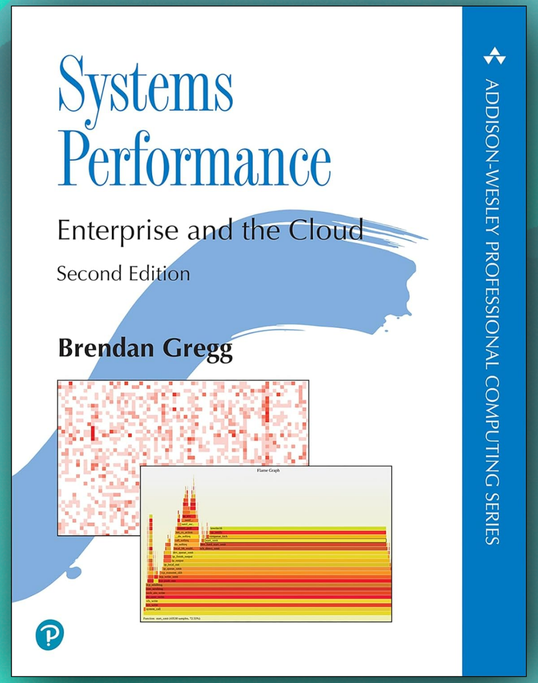
https://reinvent.one/ (consulting link)

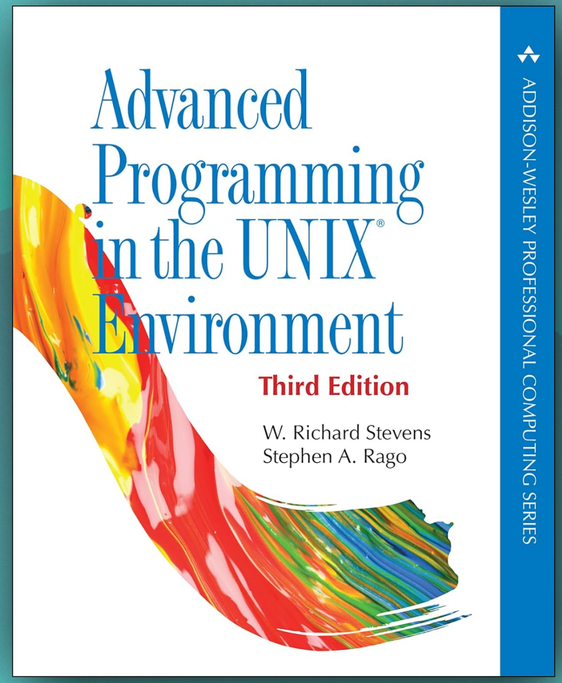
https://youtube.com/@reinvent-one



BEDESIDE READING







WHAT IS THIS TALK ABOUT?

RUBY & CONCURRENCY

1. We will look at why multi-core programming is an eventual necessity
2. We'll compare Ruby 2 & 3 concurrency primitives

- Threads
- Fibers
- Processes
- EventMachine
- Ractors* (ruby 3)

If time permits....

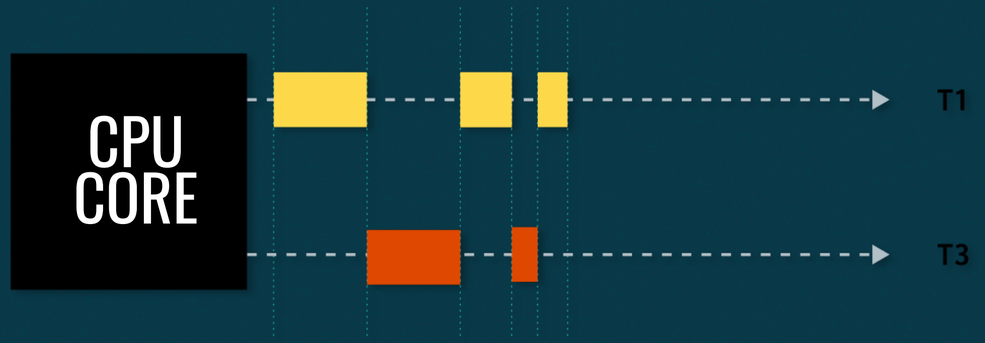
3. When to use what...

CONCURRENCY VS PARALLELISM

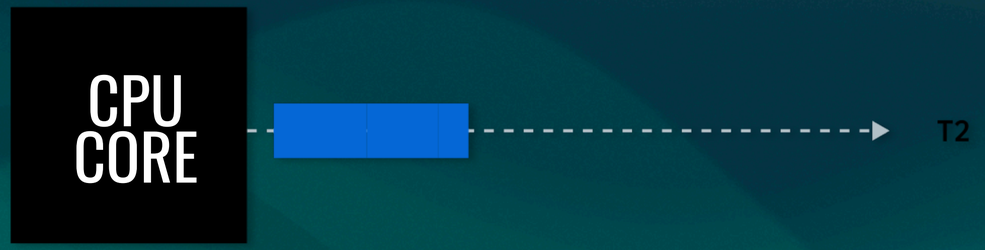
CONCURRENCY VIA CONTEXT SWITCHING VS PARALLELISM

Context Switching:

1 x CPU — 1 x CPU Task



Single Thread, No Context Switching



Between two cores — true parallelism.

QUICK QUIZ ON RUBY 2

EXAMPLE 1: CHECKSUM CALCULATION

- Let's say we we need to compute a checksum on each object in an array.
- We know that computing checksum is an operation on CPU and it requires no IO.

```ruby
# EXAMPLE 1: COMPUTE CHECKSUMS

require 'digest/md5'

10.times.map do
  Thread.new do
    Digest::MD5.hexdigest(rand)
  end
end.each(&:value)
```

- Question:
  Should our implementation use threads (or a thread pool) to parallelize our computation and complete the task faster?
  - Answer:
    Not on MRI Ruby 2: if each checksum computation requires only the CPU, there is no benefit in using threads in MRI Ruby.
  - Answer:
    Yes on JRuby
  - And what about Ruby 3?

EXAMPLE 2: CRAWLING THE WEB

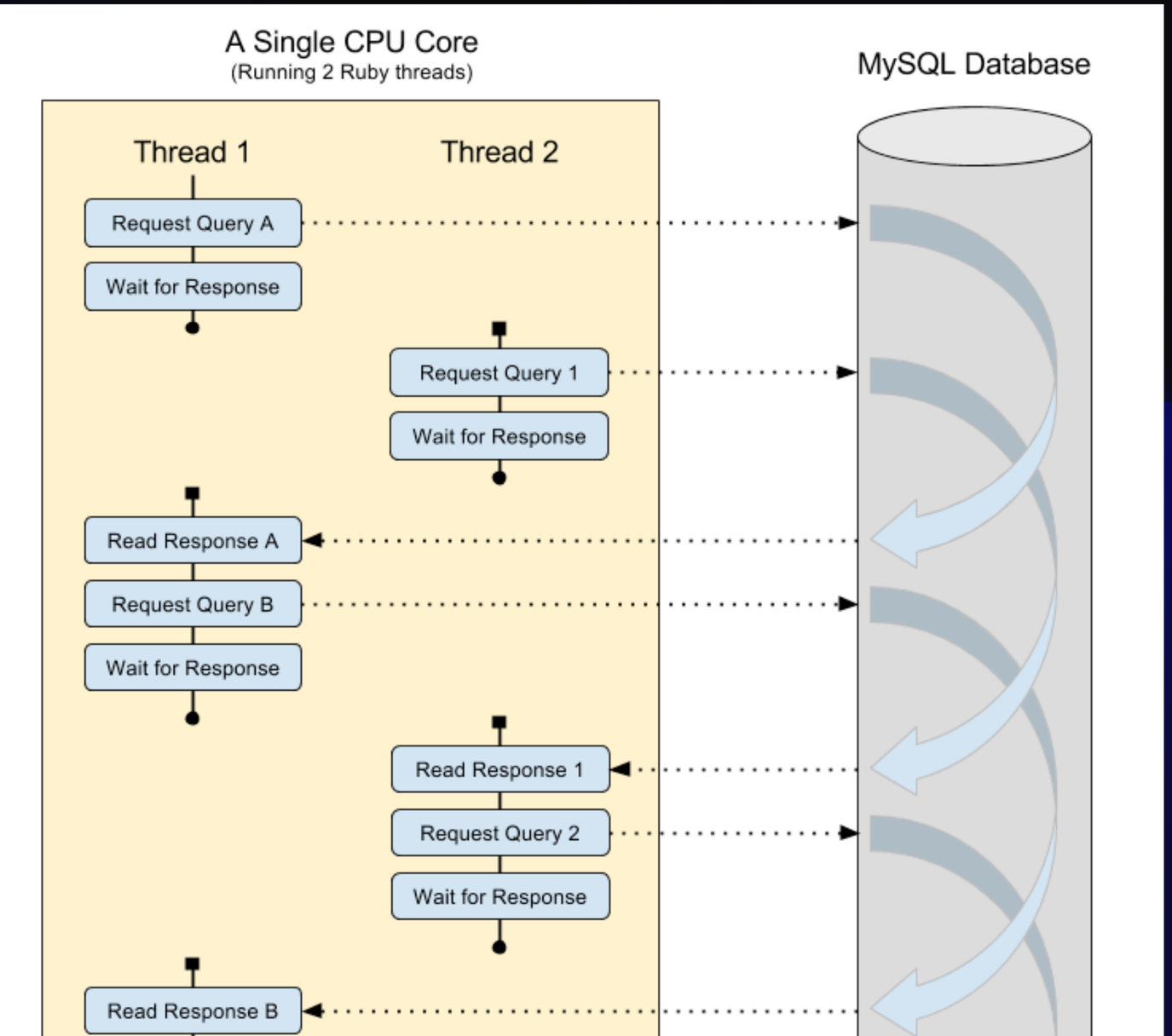
- Let's say we want download remote content given an array of URLs.
- We know that reading from a remote URL over the network is an operation on IO

```ruby
# EXAMPLE 2: LOAD REMOTE URL
require 'open-uri'

10.times.map do
  Thread.new do
    open('http://zombo.com')
  end
end.each(&:value)
```

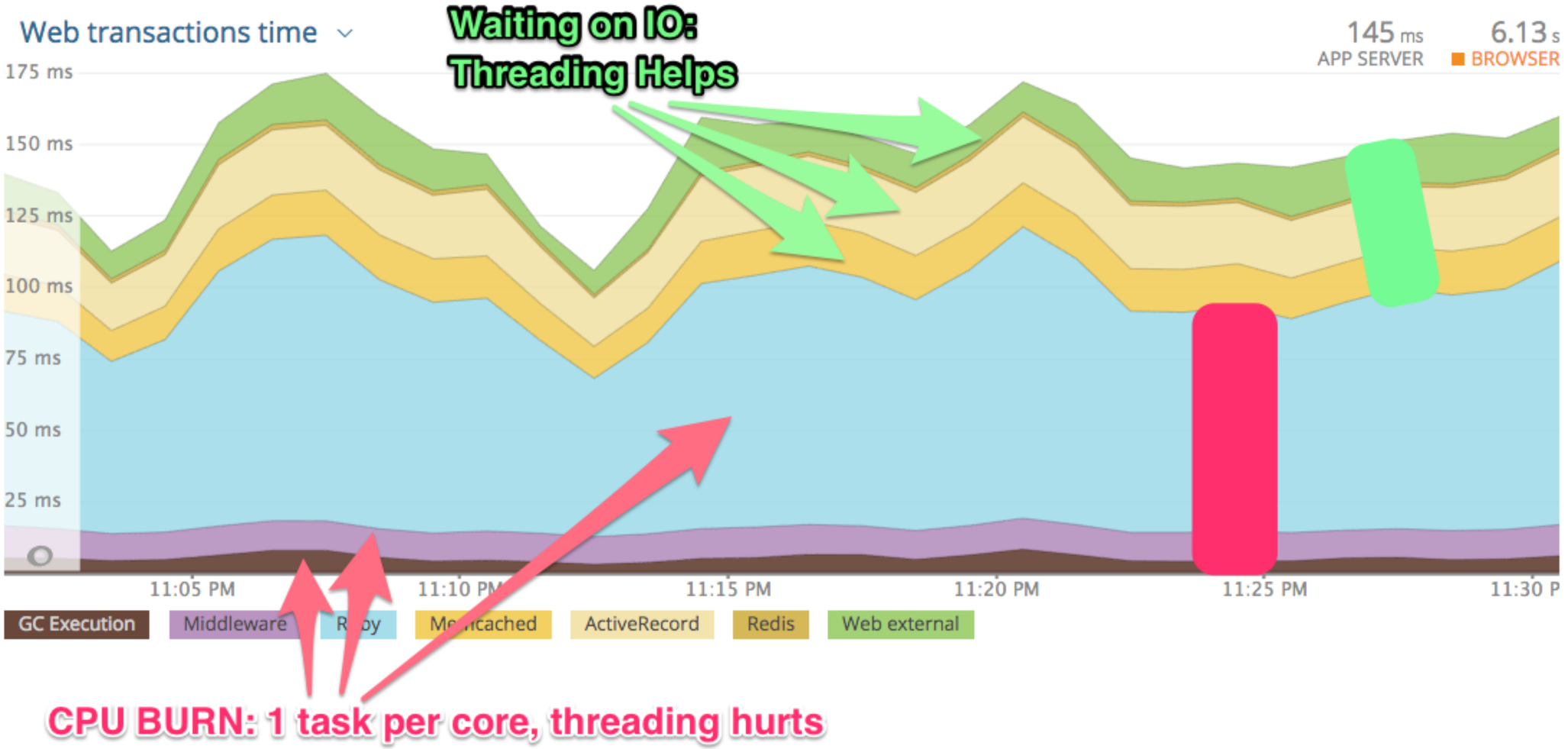
- Question:
  Should our solution create threads to parallelize our computation so that we can fetch URLs a lot faster?
  - Yes on MRI Ruby 2+, JRuby, etc. ✔
    Use a Thread Pool to prevent creating too many threads Each thread will spend some time waiting on IO: network is slow.

RUBY 2 CONCURRENCY MODEL



- In Ruby 2 we have a Global Interpreter Lock that protects internal data structures in Ruby from race conditions.
- Yet, even Ruby 2 can take advantage of the pauses caused by system calls that are waiting on IO: typically writing/reading to/from disk, network (waiting on databases, caches, API services)

READING NEW RELIC: WAITING ON IO VERSUS CPU BURN



PARALLEL NECESSITY

WHY IS CONCURRENCY SO DAMN IMPORTANT?

Concurrency means multiple computations are happening at the same time.
Concurrency is everywhere in modern programming, whether we like it or not:

- Multiple computers in a network (network concurrency)
- Multiple users are typically accessing your software at the same time (online concurrency)
- Multiple applications running on one computer (OS-level multi-tasking)
- Multiple tasks running within each application (multiple threads, fibers, etc)
- Multiple CPU Cores working on different tasks at the same time (processes, Ractor?)

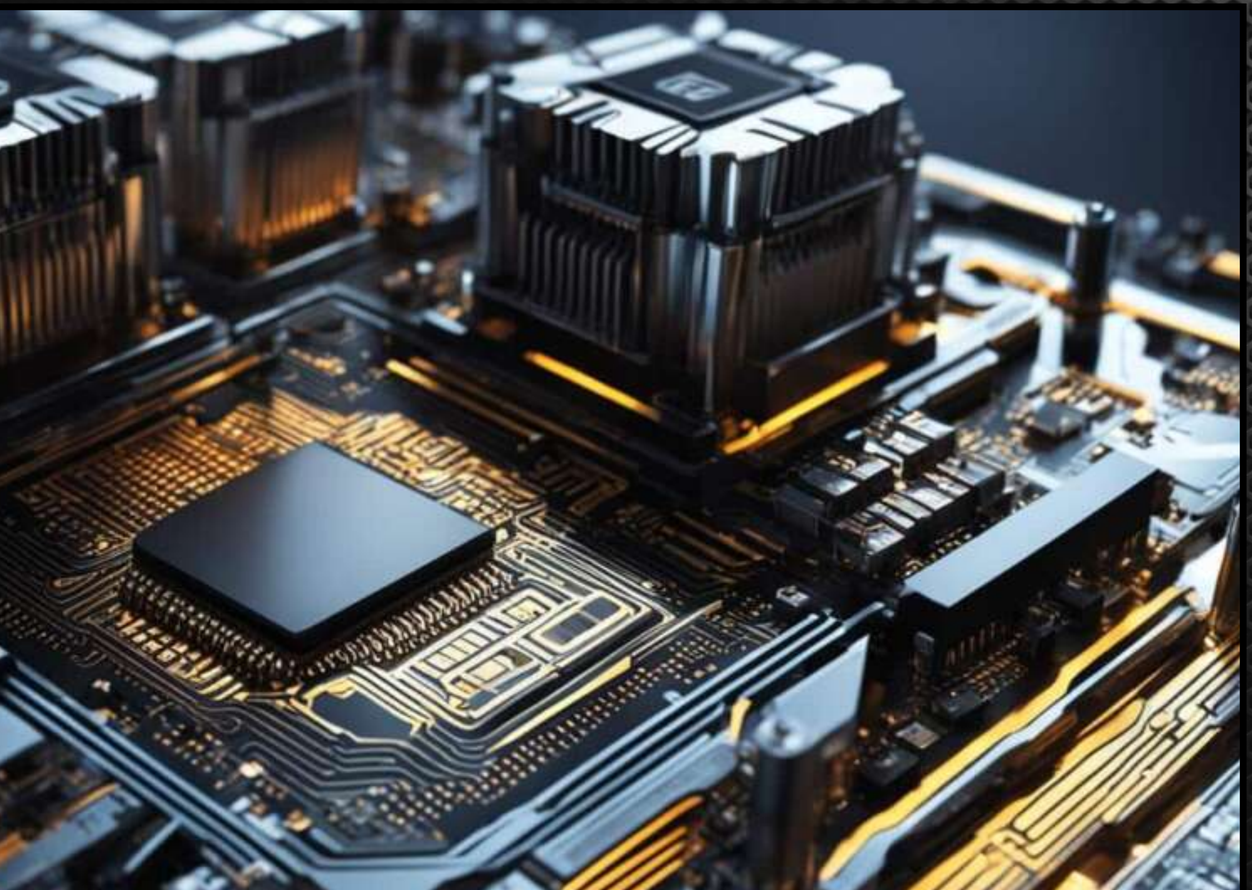
HOW'S MOORE'S LAW IN 2024?

- Though Moore’s Law has largely held true for over 50 years, some signs indicate that its exponential growth may be reaching an end in the 2020s decade.
- Largely due to physical constraints of nanometer technology.
- Since early 2010s hardware manufacturers answered with multi-core systems, some servers boasting 256 real CPU cores

MANY CORES = PARALLELISM

- CPU Manufactures went "wide" (adding more cores) instead of going "up" (increasing CPU clock) due to nano-meter constraints
- Software community responded with several new languages designed for fast multi-core applications:
  - Before: Java & C++ (pthreads, etc)
  - Now: Go & Rust
  - And maybe Ruby 3?

PROCESS PER CORE MODEL



- How do you write your program to take advantage of multi-core systems?
- In Ruby 2 we could use Threads, Fibers, and multi-processes (like Puma or Unicorn, or Sidekiq)
- But we'd still need as many Ruby processes as the cores to fully saturate the available CPU.

RUBY GEM • PARALLEL

- Supports a thread pool, as well as a process pool, as shown in the following example:

```ruby
Parallel.each(User.all, in_threads: 8) do |user|
  ActiveRecord::Base.connection_pool.with_connection do
    user.update_attribute(:some_attribute, some_value)
  end
end

# maybe helps: reconnect once inside every fork
Parallel.each(User.all, in_processes: 8) do |user|
  @reconnected ||= User.connection.reconnect! || true
  user.update_attribute(:some_attribute, some_value)
end
```

ENTER RUBY 3

QUESTION:
TO FULLY SATURATE ALL AVAILABLE CORES ON A MULTI-CORE SERVER, WE SHOULD...

A) RUN ONE RUBY PROCESS WITH MANY THREADS

B) RUN SEVERAL SINGLE-THREADED RUBY PROCESSES

C) RUN AS MANY RUBY PROCESSES AS THE NUMBER OF CPU CORES AVAILABLE

D) RUN ONE RUBY PROCESS THAT FORKS THOUSANDS OF WORKERS

QUESTION:
TO FULLY SATURATE ALL AVAILABLE CORES ON A MULTI-CORE SERVER, WE SHOULD...

A) RUN ONE RUBY PROCESS WITH MANY THREADS

B) RUN SEVERAL SINGLE-THREADED RUBY PROCESSES

C) **RUN AS MANY RUBY PROCESSES AS THE NUMBER OF CPU CORES AVAILABLE**

D) RUN ONE RUBY PROCESS THAT FORKS THOUSANDS OF WORKERS

THANK YOU!

KONSTANTIN GREDESKOUL

https://github.com/kigster

https://twitter.com/kig

https://kig.re

https://reinvent.one/