50 Shades of

Concurrency

About me

About me

@michaelklishin

About me

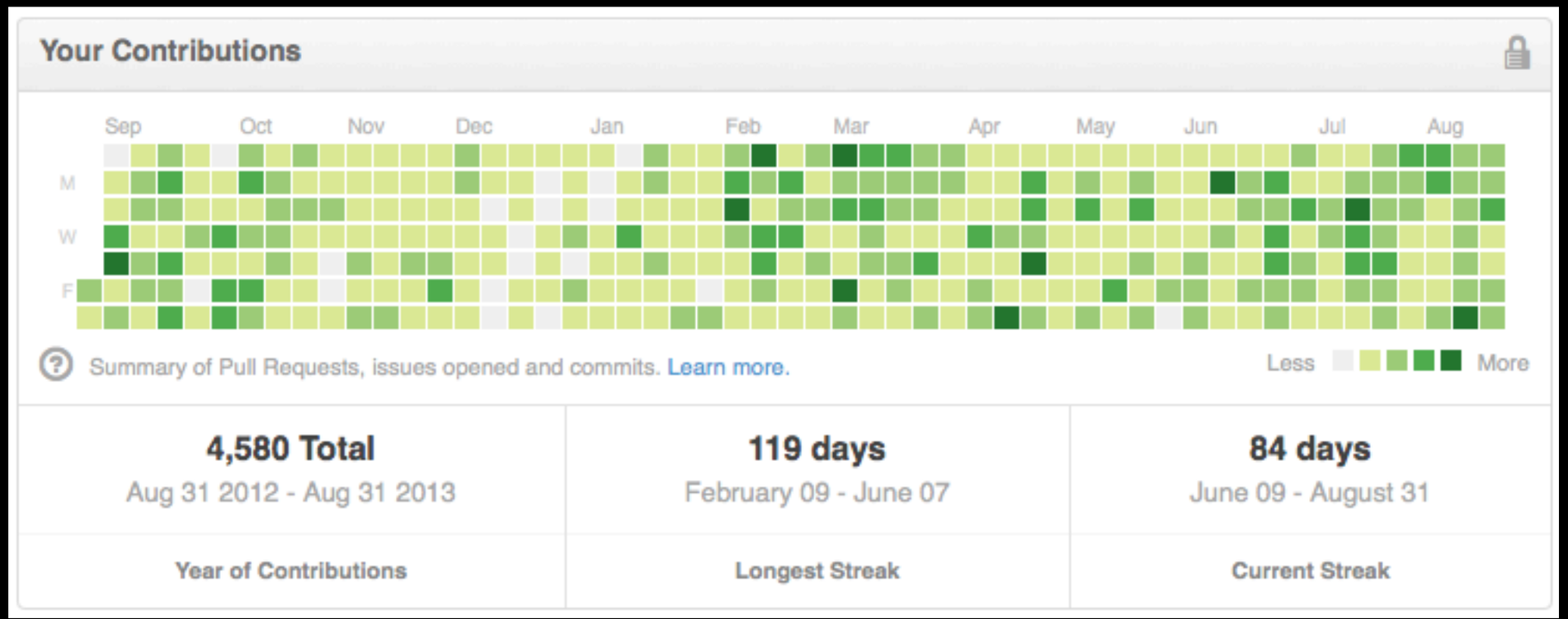
About me

clojurewerkz.org

About me

Ex-Travis CI core

About me


Your Contributions
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
M
W
F
Summary of Pull Requests, issues opened and commits. Learn more.
Less
More
4,580 Total
Aug 31 2012 - Aug 31 2013
Year of Contributions
119 days
February 09 - June 07
Longest Streak
84 days
June 09 - August 31
Current Streak

About me

data processing and analytics

About me

dozen of programming languages, <3 λ

About me

recognized animated gifs expert

About this talk

About this talk

N lightning talks combined

About this talk

Multiple related topics

About this talk

Observations

About this talk

Rants

Do we really need concurrency?

The shit is just too damn hard


Thank you
but no
ICANHASCHEEZBURGER.COM

Do we need concurrency?

«In 2013 alone, IP traffic is expected to grow by around 14 exabytes per month…» — ITU

Exabyte? LOLWUT?

Exabyte

- 10^18 bytes

Exabyte

- 10^18 bytes
- 1,000 petabytes

Exabyte

- 10^18 bytes
- 1,000 petabytes
- 1,000,000,000 gigabytes


San Ysidro
Passenger
Pedestrian

If your project is data-driven, concurrency is no longer a nice-to-have

Do we need concurrency?

Michael Klishin

Repositories | Profile

Flick the switches below to turn on the Travis service hook for your projects, then push to GitHub.

Last synchronized from GitHub: about 16 hours ago — Sync now

michaelklishin/acits	OFF
michaelklishin/adventures_with_ssl_talk	OFF
michaelklishin/amqp_broker_stress_tests	OFF
michaelklishin/apache-jackrabbit-chef-cookbook	OFF
michaelklishin/cassandra-chef-cookbook	OFF
michaelklishin/chash	OFF
michaelklishin/clang-chef-cookbook	OFF
michaelklishin/clj1062	OFF
michaelklishin/clojure-cookbook	OFF
michaelklishin/community	OFF
michaelklishin/crawlista	ON
michaelklishin/cucumber.el	OFF
michaelklishin/cyclist	ON

Timely updates from N sources

is a UI feature

Products compete on

user experience

Reasonably responsive

UI or GTFO

Do we need concurrency?

The world around us is concurrent

Pull iPhone headphones jack when the track is being switched

Why it happens?

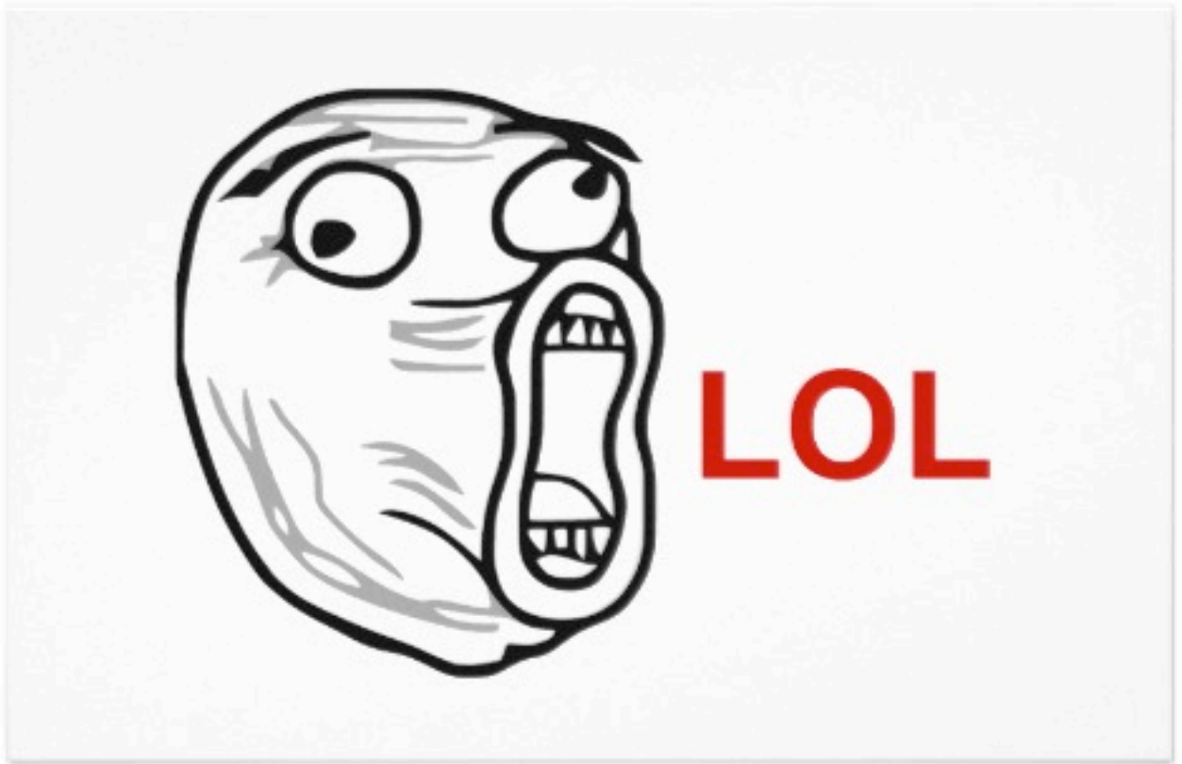
Race condition

Race conditions, explained

Do we need concurrency?

Do we need to
understand
concurrency?

«my framework will solve this problem for me!»


LOL

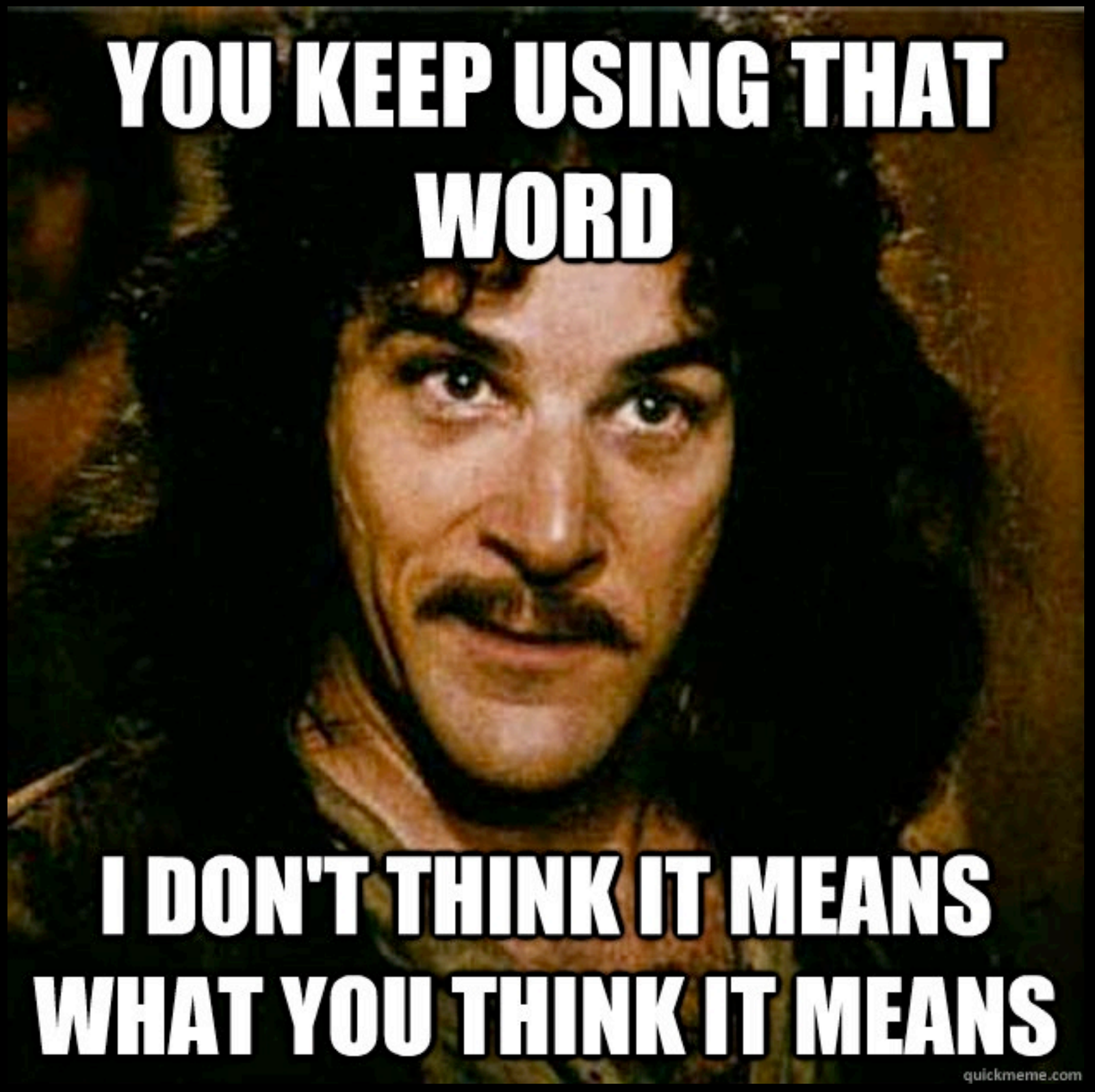
```ruby
require "mutex"

m = Mutex.new

m.synchronize do
  m.synchronize do
    # ...
  end
end
```

```
ThreadError: deadlock; recursive
locking
  from (irb):4:in `synchronize'
  from (irb):4:in `block in
irb_binding'
  from (irb):4:in `synchronize'
```


O RLY?


YOU KEEP USING THAT WORD
I DON'T THINK IT MEANS WHAT YOU THINK IT MEANS
quickmeme.com

«Deadlock describes a situation when one or more threads are blocked waiting on each other…»

```ruby
require "mutex"

m = Mutex.new

m.synchronize do
  m.synchronize do
    # ...
  end
end
```

Unfortunately, Ruby standard library has more poor design decisions w.r.t. concurrency

Solvable?

require “monitor”

5-10% throughput decrease

Backwards compatiblity

Mutexes, explained

Mutexes, explained

«I know, Java is incredibly solid and stable!»

```
Map<String, Integer> m =
new HashMap<>();
```

```
m.put("michael", 3);
```

results in...

CPU usage spikes

HashMap can create infinite loops internally if you write to it concurrently, and a write causes a resize.

When I find myself in times of trouble
Prof. Doug Lea comes to me
Coding lines of wisdom
j.u.c.

java.util.concurrent

«My VM will solve the problem for me!»

Very GC/allocations heavy algorithm implemented in Clojure

«abysmal multicore performance, especially on AMD processors…»

https://groups.google.com/d/topic/clojure/48W2eff3caU/discussion

~100 replies and 20 participants later…

The bottleneck is…

OpenJDK/Oracle JDK memory allocator

LOLWUT?

«It's C++ and a few billions of $ worth of engineering man-hours!»

Not every algorithm has concurrent alternatives

Efficient, concurrent writes is a hard problem

Lock contention

Lock contention, exlained

Lock contention, exlained

Memory models

«My compiler will solve the problem for me!»

Compilers are very sophisticated tools and can do a lot for you. But they still need your help from time to time.

False sharing

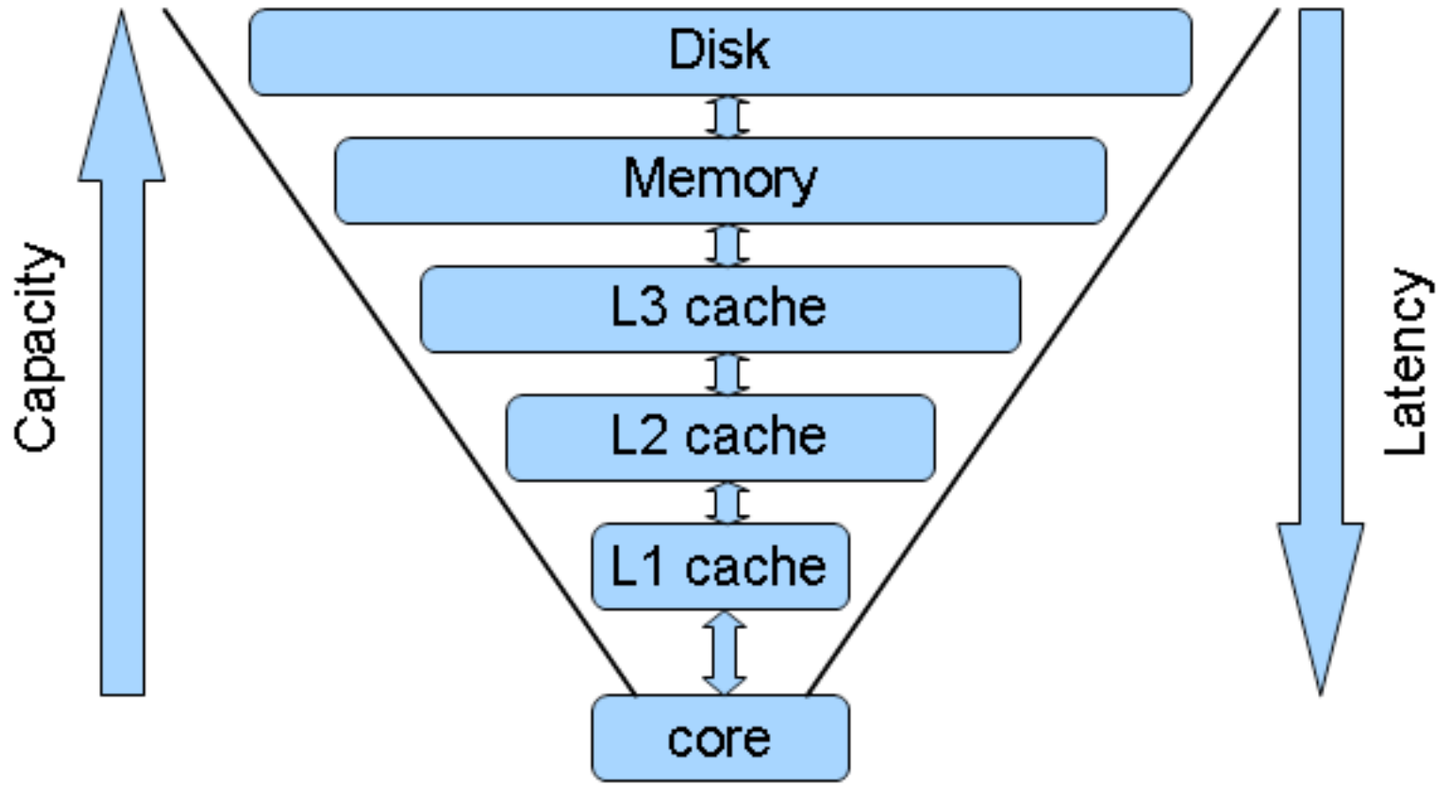
False sharing

```
class Metrics
  attr_accessor :signin_requests
  attr_accessor :signout_requests
end
```

Thread A updates :signin_requests

Thread B updates :signout_requests

CPU caches


Capacity
Disk
Memory
L3 cache
L2 cache
L1 cache
core
Latency

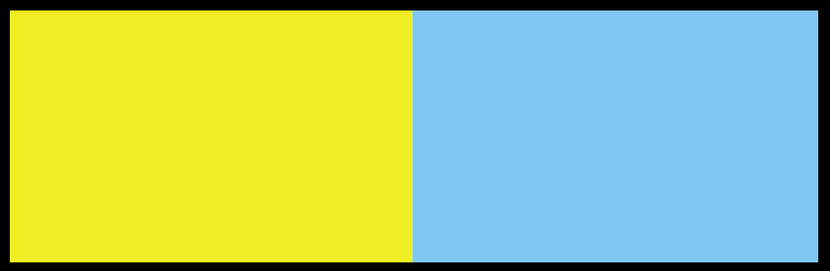
Memory is stored in CPU caches in chunks

The chunks are known as cache lines


VISA

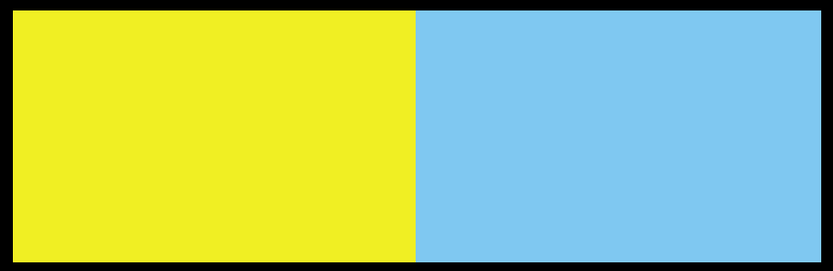
Cache line

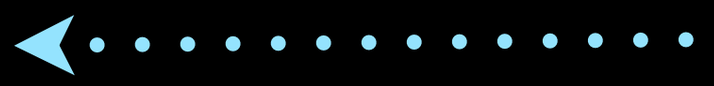
Cache line



Thread A updates

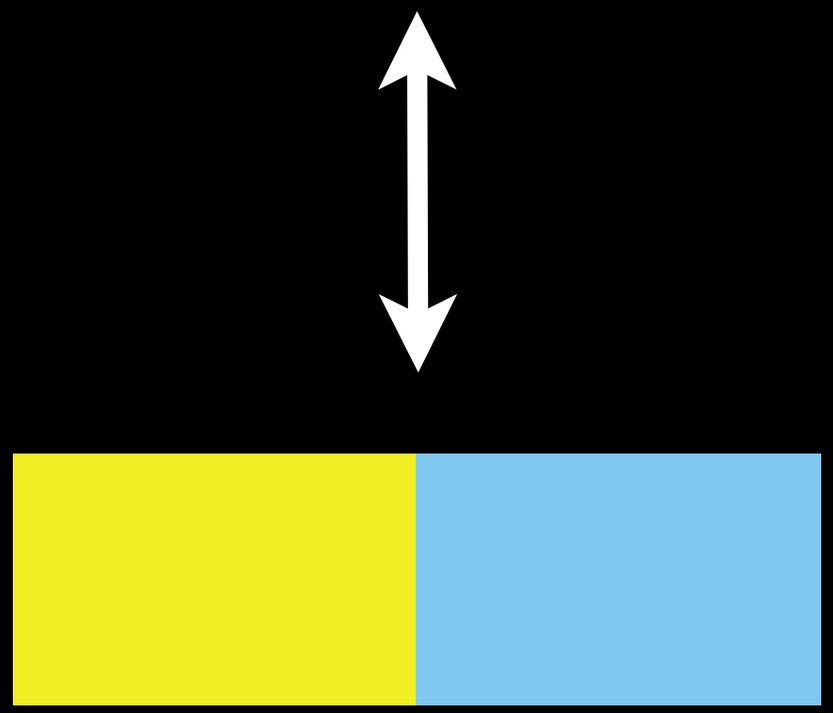
Cache line





Thread B updates

RAM

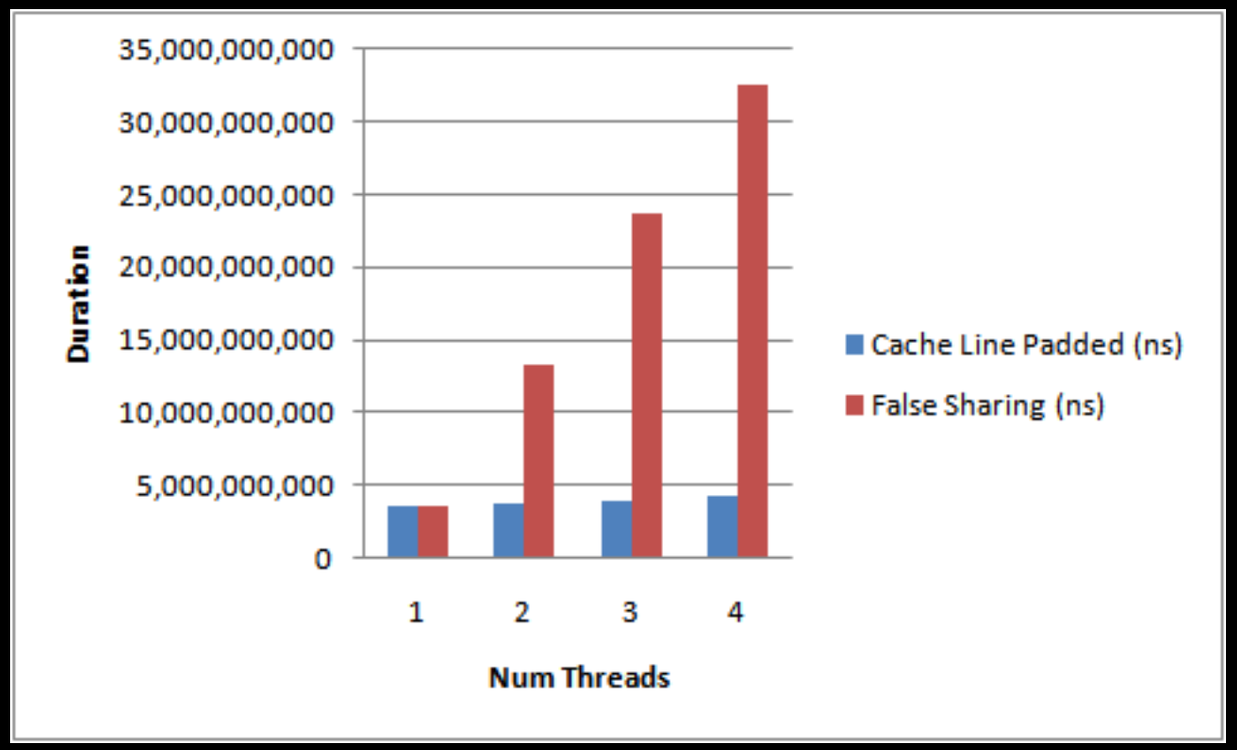


Cache line

Threads update independent variables,
but the entire cache line is
has to be read from RAM

Solvable?

```
public final class VolatileLong {
   public volatile long value = 0L;
   public long p1, p2, p3, p4, p5, p6;
}
```


35,000,000,000
30,000,000,000
25,000,000,000
20,000,000,000
15,000,000,000
10,000,000,000
5,000,000,000
0
Duration
1
2
3
4
Num Threads
Cache Line Padded (ns)
False Sharing (ns)

«Threads suck! UNIX processes will solve my problem!»

fork(2)

fork(2) vs forkall(2)

fork(2) and parent process that holds mutex(es)

async signal safe

«The child inherits copies of the parent's set of open file descriptors…»

«All APIs, including global data symbols, in any framework or library should be assumed to be unsafe after a fork() unless explicitly documented to be safe or async-signal safe…»

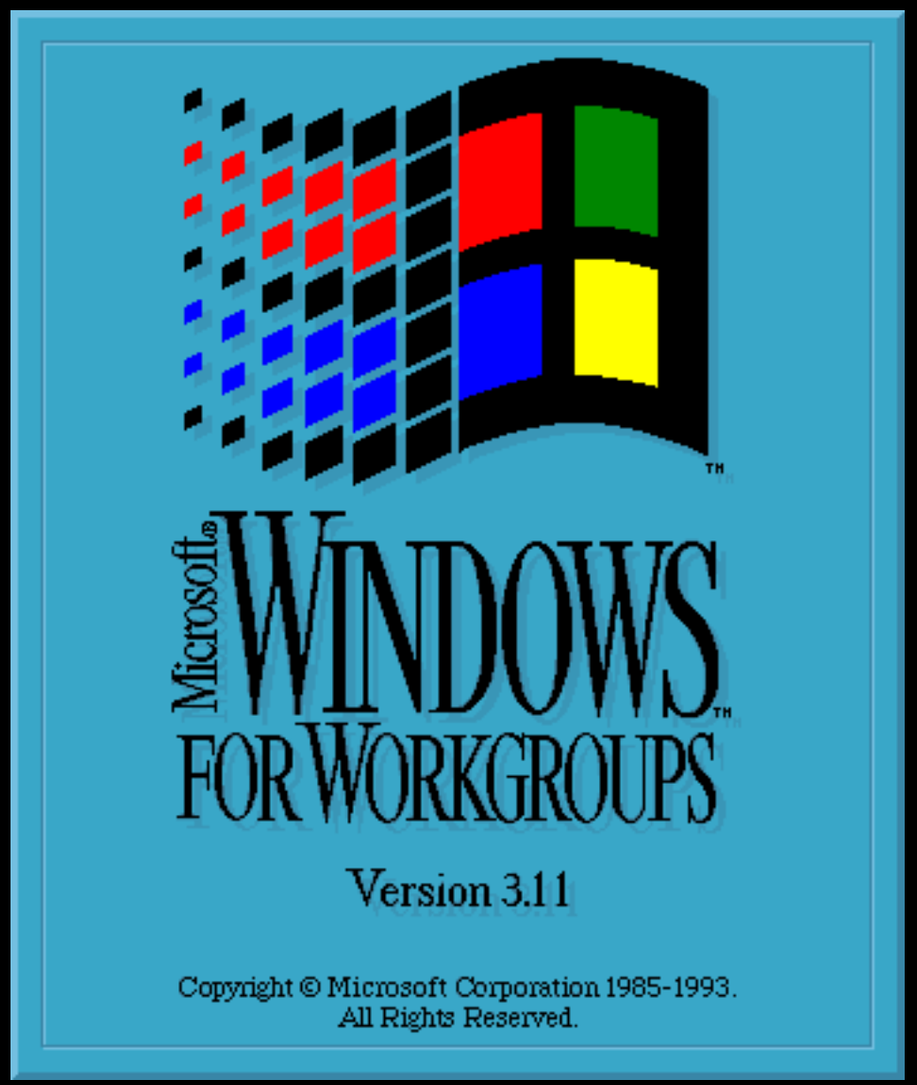
man 2 fork, CAVEATS

man 2 vfork

Process race conditions

Not that many concurrency libraries for processes

«My OS will solve the problem for me!»


Microsoft
WINDOWS
FOR WORKGROUPS
Version 3.11
Copyright © Microsoft Corporation 1985-1993.
All Rights Reserved.


© 2001 Apple Computer, Inc. All rights reserved. Apple, the Apple logo, Mac, the Mac logo, and Sherlock are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.
Mac OS 9
Featuring Sherlock 2
To start up from this CD, hold down the C key as the computer starts up.
Version 9.2.1
691-3334-A
Mac

Cooperative concurrency, explained

Cooperative concurrency, explained

Preemptive scheduling

Scheduling

C10K

C100M

Lightweight processes

Erlang, Clojure's
core.async, Go

Lightweight processes: the Holy Grail?

Dat single Erlang process bottleneck

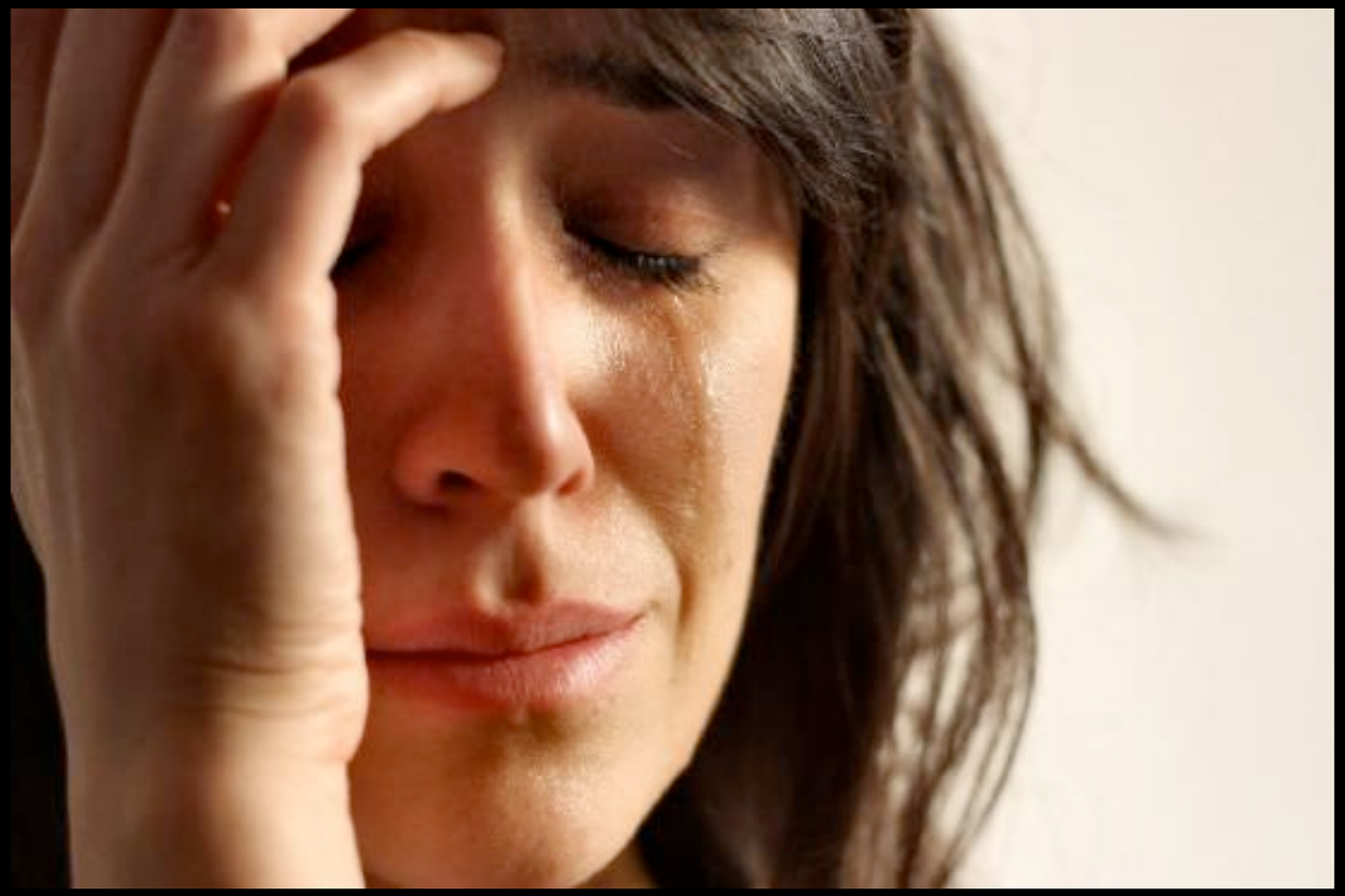
Unbounded mailboxes

and backpressure

Race conditions

Deadlocks in Erlang

Livelocks in Erlang



Concurrency is about

algorithms

Correctness

Race conditions are possible with any language

Concurrency makes it harder to reason about correctness

Most issues arise from

shared mutable state

Solvable?

Solutions

- Eliminate shared state

Solutions

- Eliminate shared state
- Use immutable data structures

«Immutable? But how do I change anything?»

Immutable data structures create copies when updated (mutated)

«But copying is expensive!»

Copying doesn't have to be a naïve "full copy"

Value immutability vs reference immutability

Value immutability:
Haskell, Clojure, Erlang.

Reference immutability: C++, Java, Scala, Rust.

Value immutability: libraries for any language (Ruby, Java, JS)

Immutability and message passing make reasoning much easier

Summary

Concurrency is no longer a nice-to-have

There is no such thing as a “thread safe framework”

Concurrency is about algorithms

Algorithms are different at every level

Problems are different at every level

No substitute

for

understanding

Being able to reason about your program matters

Immutability and message passing make reasoning much easier

Not about event

loops or Web Scale™

Concurrency is the new Cloud Computing

Concurrency is the new Clown Computering

«Blah blah, X blows Y out of the water in concurrency…»

Be specific or GTFO

Start learning a language with immutable data structures today

clojure-doc.org

Thank you

@michaelklishin

clojurewerkz.org