Concurrency In The Erlang VM

Erlang is often referred to as the "*concurrency oriented programming language*".

How did it get this name? How can a language created in the 80s for the telecom industry help us now?

Swaathi Kakarla
@imswaathik

SKCRIPT

The Plan

OS

How does the OS process instructions?

Execution

How is a program executed?

Models

How can 2 programs be executed?

The Plan

Demo

Getting our hands dirty with Elixir.

So much more to learn! So little time to do it.

Extended Topics

Coronavirus will end.... maybe?

SKCRIPT

whoami

Swaathi Kakarla

CTO, Skcript
Guest Author
ROR Contributor
Yoga Practitioner

SKCRIPT

SAUCELABS

RAILS

zendesk

TOMTOM

Twistlock

VictorOps

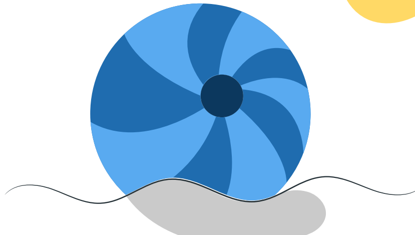
ActiveState

How does the OS process instructions?

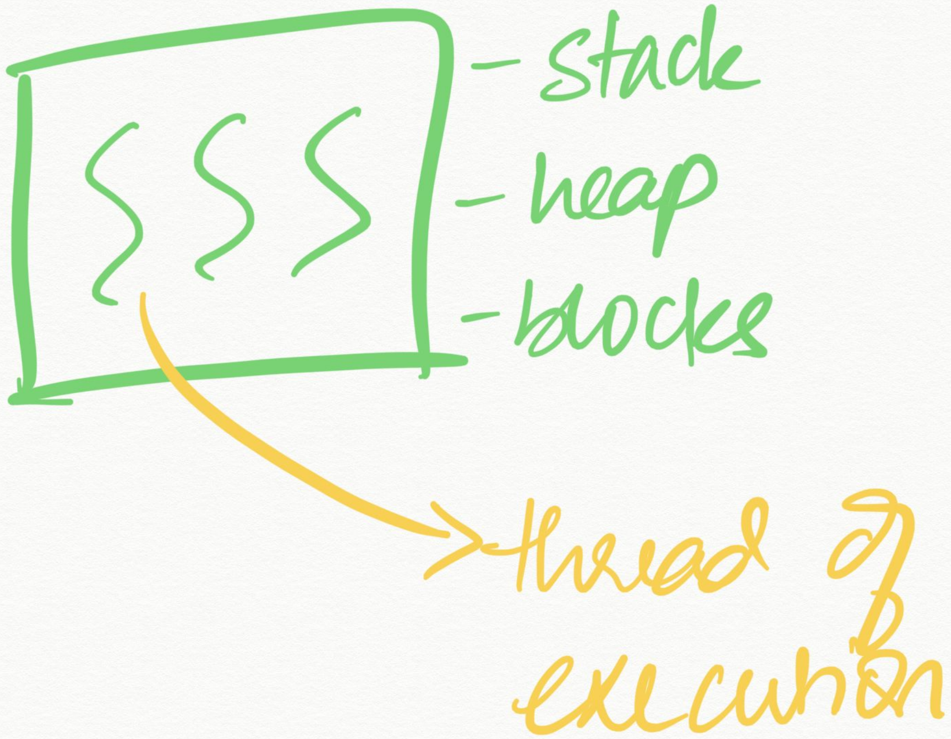
1. Obviously uses the **CPU**.
2. The CPU executes processes.
3. Processes are isolated blocks of execution. It occupies memory, it has a stack and heap, it is able to context switch.
4. **Threads** are “lightweight” processes.

1. The CPU does not process multiple processes at a time. It processes small bits of multiple processes sequentially, switching over so fast that it “looks” like it is processing in parallel.
2. **Context switching** is expensive!

1. **Moore’s law**: The number of transistors on an affordable CPU would double every two years.
2. Unfortunately we’ve hit a bump in the road, we’ve reached the upper limit of the Moore’s law.
3. Now it makes sense to scale horizontally, instead of vertically. Hence “cores”.
4. This happened when we realized going much higher than 4GHz is very difficult and futile. *Speed of light actually became a constraint*.



An OS Process



SKCRIPT

Execution

Sequential

Start executing process B, only after process A is complete.

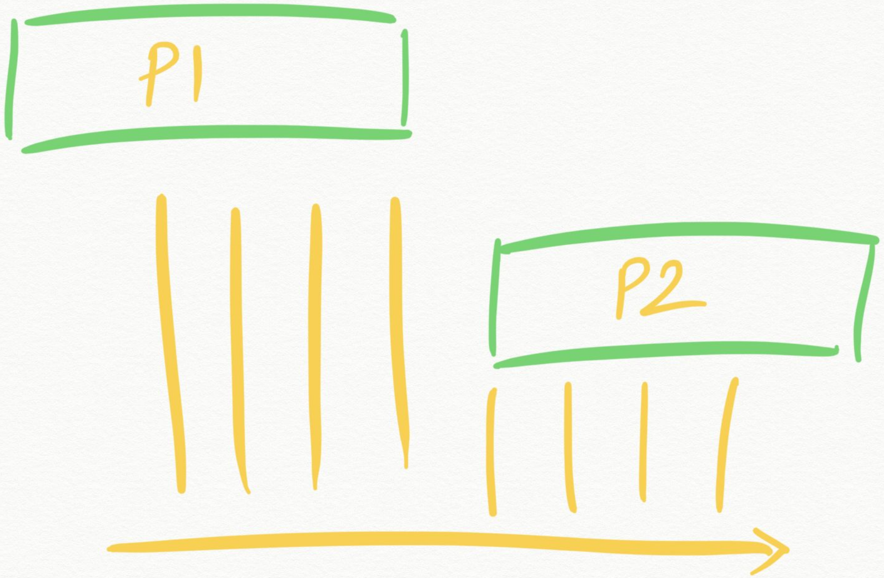
Concurrency

Break up process A and B, switch between them really fast.

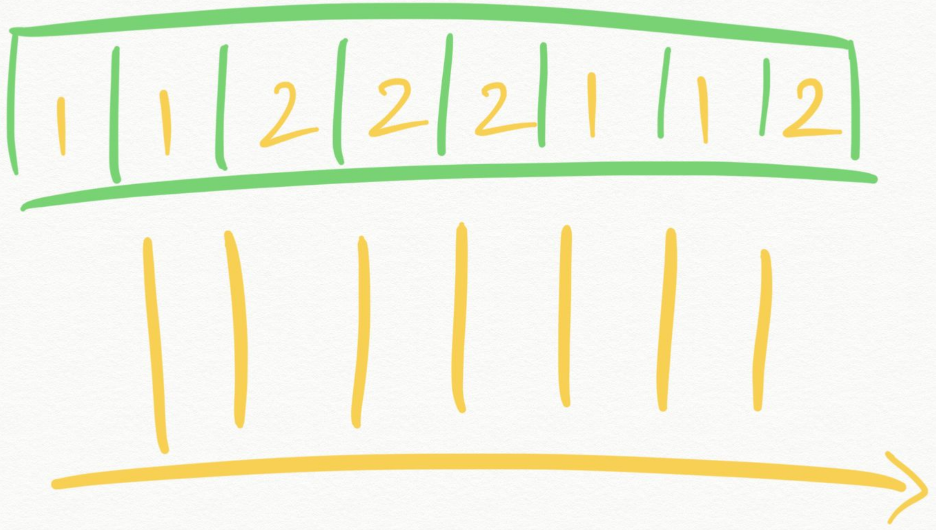
Parallelism

Execute both process A and process B at the same time.

Sequential Processing

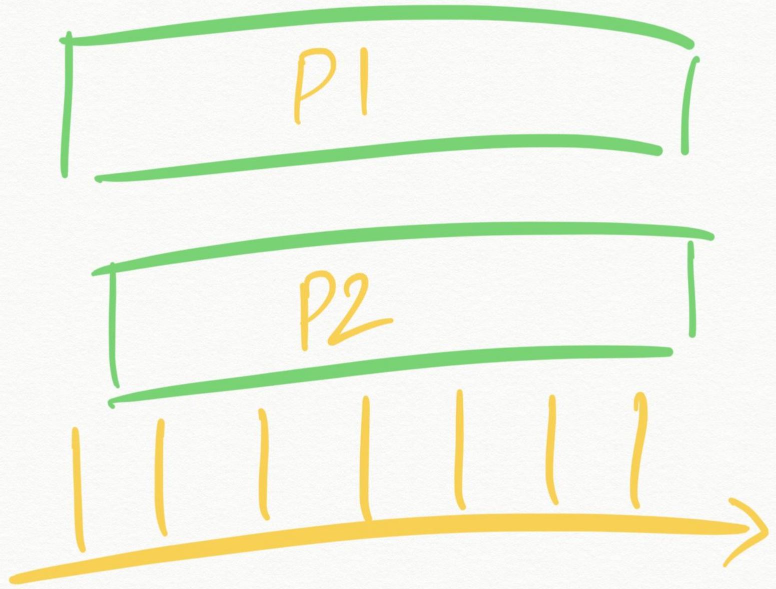


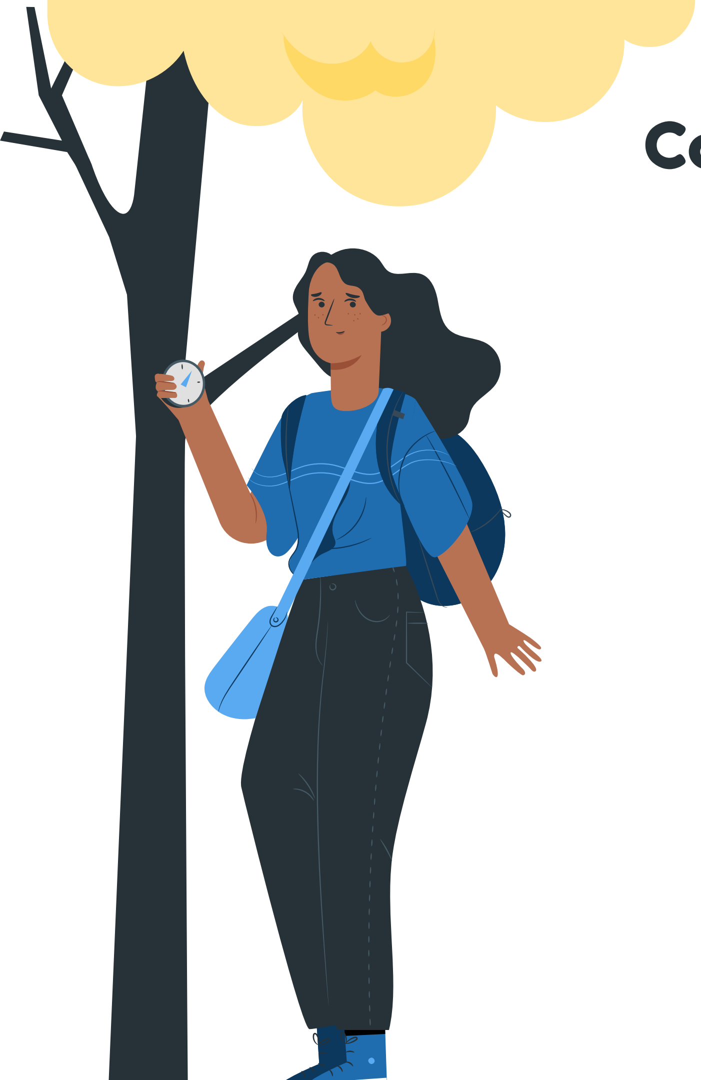
Concurrency



SKCRIPT

Parallelism





Concurrency Models

SM

Shared Memory model used by Java and C#.

Actor

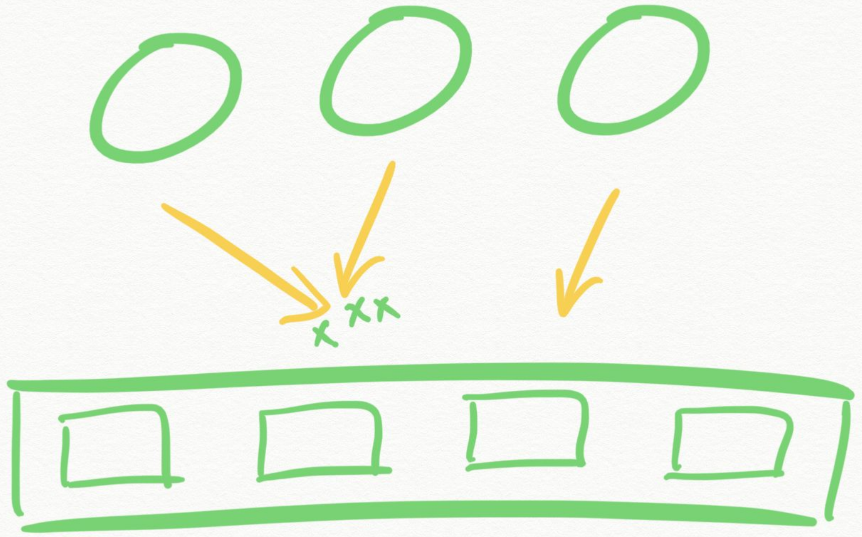
Actor model used by Erlang and Rust.

CSP

Communicating Sequential Process model used by Go.

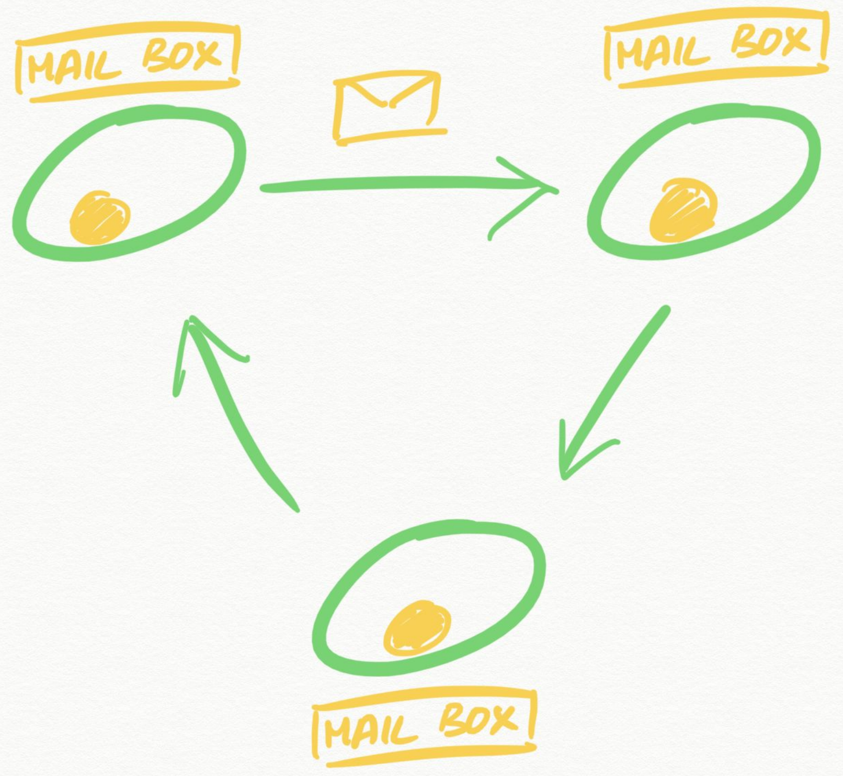
SKCRIPT

Shared Memory

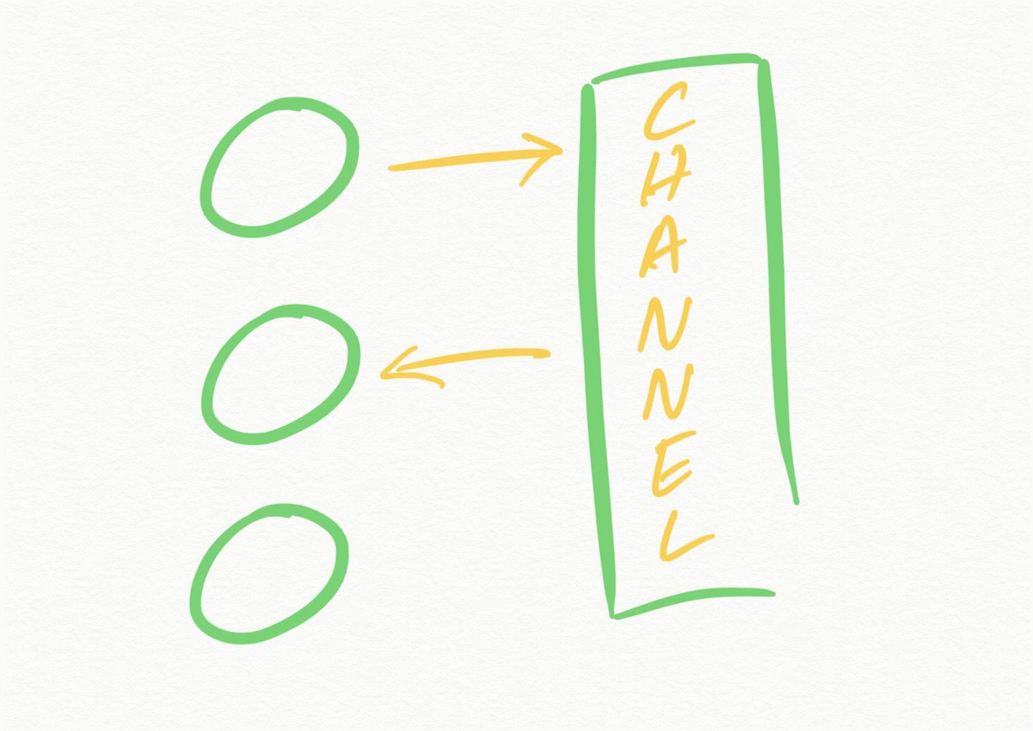


SKCRIPT

Actors



Communicating Sequential Processing



Actor Model

Create

Create more actors (these are not child processes).

Send

Send messages to other actors.

SKCRIPT

Actor Model

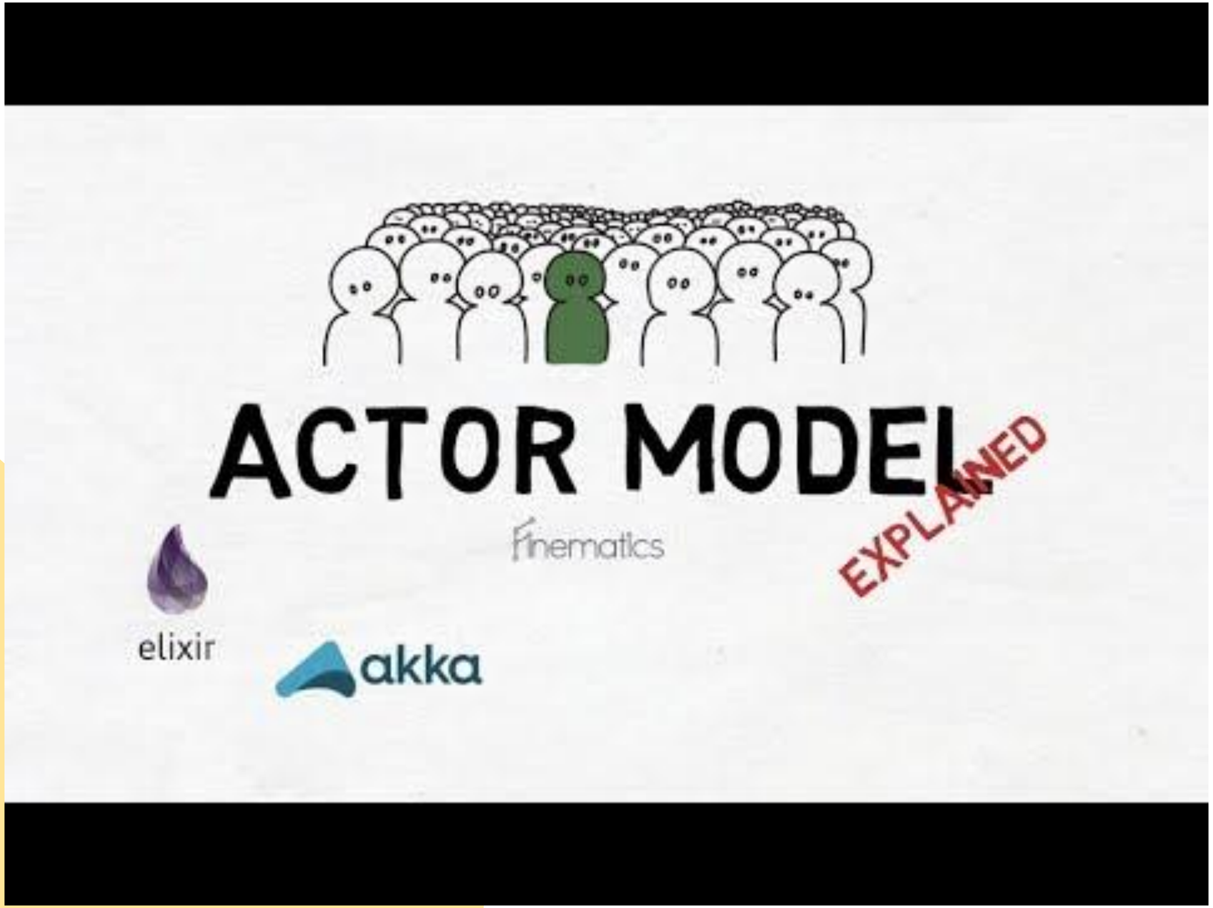
Designate

Designate what to do with the next message. It basically means defining how this state will look like for the next message it receives. Or, more clearly, it's how actors *mutate state*.

SKCRIPT

The Actor Model in 5 Min

Because a picture is worth a 1000 words, and a video so much more.



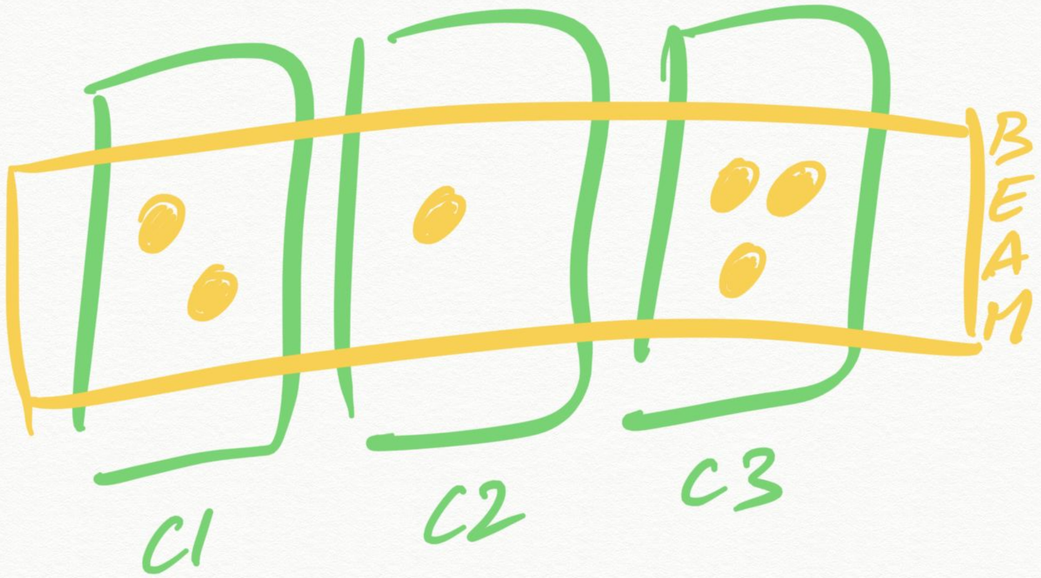
So.. Erlang?

Erlang is so great at concurrency because of BEAM.

Beam me up Scotty....? No!

BEAM is the Erlang VM. It schedules lightweight Erlang processes. Erlang processes, not OS processes.

BEAM



01
BEAM
Lightweight Erlang threads and scheduling.

02
Actor Model
Message passing and isolated processes.

03
Distributed
Scale horizontally and make use of all cores.

04
Fault Tolerance
Failure at one node does not affect other nodes.

Also...

No GIL

Global Interpreter Lock

Immutability

Not mutable?

Compiled

Hence the VM!

Supervisor

Just like a real life supervisor.

GenServer

Used to keep state, execute code asynchronously.

...

I wish I knew this one to make this slide balanced.

Demo time!

Let’s have a look at how Erlang processes communicate with each other, are fault tolerant, distributed and so much more!

Resources

- https://www.knowthen.com/elixir-and-phoenix-for-beginners
- https://stackoverflow.com/questions/2708033/technically-why-are-processes-in-erlang-more-efficient-than-os-threads
- http://dockyard.com/blog/2020/05/28/scaling-up-with-elixir
- http://ablogaboutcode.com/2012/02/06/the-ruby-global-interpreter-lock
- https://tsh.io/blog/simple-guide-concurrency-node-js/
- https://www.poeticoding.com/spawning-processes-in-elixir-a-gentle-introduction-to-concurrency/
- https://www.poeticoding.com/hey-process-there-is-a-message-for-you/
- https://www.brianstorti.com/the-actor-model/
- http://blog.plataformatec.com.br/2018/04/elixir-processes-and-this-thing-called-otp/

- Zen of Erlang
- Gary from Android Authority: Process and Threads
- Concurrency in a Go Coffee Shop
- Hewitt, Meijer and Szyperski: The Actor Model

Thanks!

@imswaathik

www.skcript.com

www.swaathi.com/talks

SKCRIPT