Sharing Resources in a Share Nothing System

By Brian Mitchell

brian@strmpnk.co

How did we get here?

The passing of Joe Armstrong has given me a chance to reflect on how Erlang has influenced my life.

How did ~~we~~ I get here?

How did ~~we~~ I get here?

- Was working with Ruby for many years (lucky enough to have picked the language up before the Rails wave), Vibrant community and plenty of jobs (post-Rails at least)
- Hit a wall in what Ruby offered (remember DRb and Rinda?)
- Enter RabbitMQ (~2007) and CouchDB (~2005, Apache ~2008)
- What is this Erlang thing and why are these projects using it? (hint: ejabberd was started ~2002)

2008’s Erlang was not like 2018’s Erlang

- Slower interpreter, memory management issues, far easier to crash and burn the node (!!)
- SMP support was still being ironed out
- NIFs didn’t exist (linked in port drivers anyone?)
- No maps (now you know why keyword lists are so pervasive)
- Exception messages were … cryptic lists of tuples rather than pretty printed traces
- Tooling was focused on supporting complex release processes and thus had complex interfaces

Erlang developed a reputation

Erlang developed a reputation

Then

- *Ewww.* Syntax
- Tools
- Libraries

Is it time to stop repeating the same silly lines over and over again?

Now

- It’s simple and consistent
- rebar3 rocks
- There many on hex.pm now as well as lots of Erlang on sites like GitHub

Yes

Why Elixir

Let’s Compare Code

Not All That Different

It's easy to work with Erlang code from Elixir

Elixir APIs can be designed to be more easily used from Erlang

Erlang code is easy to use with mix

ERTS: always evolving

Counting Things

What options do we have?

- Agent
- ETS
- New: `counters`
  - write concurrency vs atomic

More On Atomics

Concurrency primitives:

- Compare and swap
- Add and get
- Swap

Tip: `erts_debug`

persistent_term

- Read heavy data like:
  - Configuration
  - Lookup tables
- Still incurs global GC
- Allows non-literal terms to be shared

To Be Continued?

Thank you!

Questions?