Erlang
Lecture 21, cs262a

Ion Stoica & Ali Ghodsi
UC Berkeley
April 9, 2018

My personal take on Erlang

- **Language created for a purpose**
  - Build modern mobile phone switches
- What kind of requirements do such switches have?
  - Highly **reliable** and **scalable**!
- In particular
  - Reliable
  - Scale up (multiprocessors)
  - Scale out (distribution)
  - Fault tolerant
  - High available

Achieving the requirements

- Reliable
  - Avoid sharing and be stateless → functional based on message passing
- Scale up (multiprocessors)
  - High concurrency → support millions of threads
- Scale out (distribution)
  - Same model for single-machine and distributed (message passing everywhere)
- Fault tolerant
  - Supervisor model, very powerful
- High available
  - Hot swapping code

Erlang

Making hard things simple, and simple things difficult

```
myfunc([]) -> [];
myfunc([First|Rest]) ->
  myfunc([Front || Front <- Rest, Front < First]) ++
  [First] ++
  myfunc([Back || Back <- Rest, Back >= First]).
```

What is this?

- 100% correct quick sort!

Impact

- Hugely successful at Ericsson
  - 70% of all worldwide calls in early 2000 went through an Erlang switch
  - Almost no downtime over 6 years

My years in grad school

- Erlang was almost dead
  - OO was on a rise, and JVM was a standard
  - Most research was on type theory (static typing)
  - Multi-paradigm languages were popular (Mozart, Scala, …)
  - Erlang had issues (not always tail-recursive, weird k/v store)
- Joe remained bullish
  - “Humans pass messages, they don’t edit each others brains”

Popularity of the language picks up

- Erlang uptick
  - Joe publishes ”Programming Erlang”, big hit
  - Silicon Valley picks up the language:
    WhatsApp, CouchDB, RabbitMQ, Facebook Messenger …
- Why?
  - Erlang was built with a purpose:
    building highly scalable and reliable distributed systems
  - With Moore’s law ending, and web-scale computing, a highly relevant purpose
  - Joe was extremely stubborn and pursued his vision & dream
  - End story