

Erlang

my name is Julian Doherty, and I use Erlang because it's the only sane way I've found to do concurrent programming

madlep@rawblock.com
<http://www.rawblock.com>

Main Design Goal: Reliability

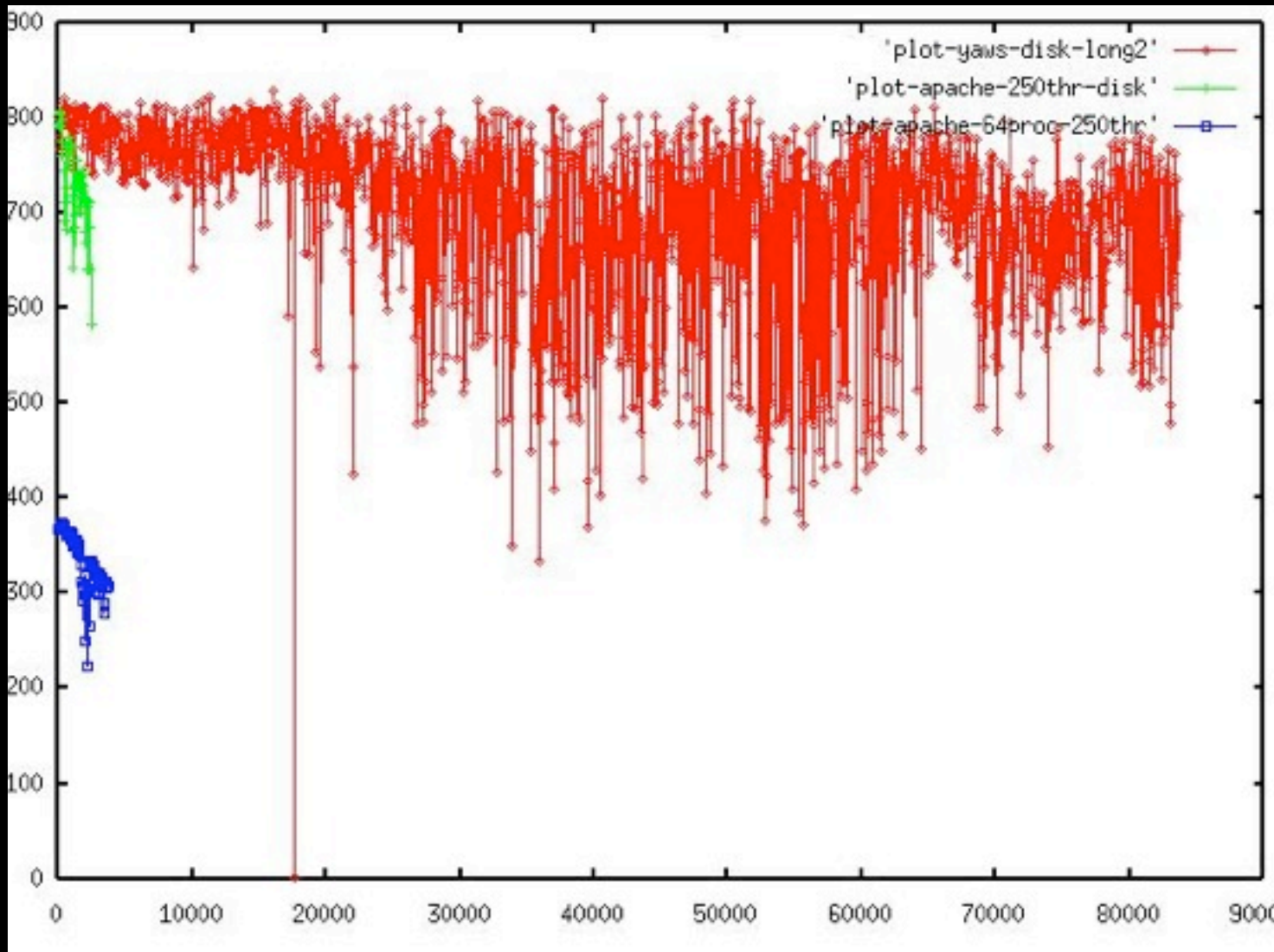
- Designed for real world use
- Concurrent
- Distributed
- Fault tolerant
- CouchDB, Mochiweb, Rabbitmq, Ejabberd

Concurrency

- Asynchronous message passing
- Can fake sync concurrency
- NO shared state (can cheat though)
- Each process has own “inbox”
- `spawn(...)` == `Object.new`

Light Weight Processes

- OS Process > OS Thread > Erlang Process
- Green (software) threads, non blocking IO
- Built in scheduler
- Can start 100,000s - 1,000,000s of processes
- Cheap to create / stop / switch
- E.g. <http://ihatevans.com> - 3000 req/sec
- 1,000,000 user comet app



meaningless micro benchmark:
Yaws (Erlang web server) vs Apache

Distributed

- Built in
- Same syntax and semantics as local concurrency
- Slower though, so can't just ignore
- All or nothing security (annoying)

Fault Tolerance

- NO THREADS (been burnt too many times)
- “Nine nines” (99.9999999%) uptime
- Linked processes
- Supervisors
- Let processes die, then restart
- OTP - framework/best practices for servers

Language Syntax

- Immutable, single assignment
- Pattern matching
- Strictly functional (no OO in there)
- Compiled, Dynamic typed
- Tail recursion, actor pattern used for state
- Sometimes clunky (Strings, Records)

The code!

(talk to me about the Erlang / Ruby Rack handler I hacked together later on if you're still interested over a couple of beers)