Living the cutting edge

Bitcoin & Elixir

Yurii Rashkovskii yrashk@bex.io

This presentation lacks coherency and features racing thoughts, unpaid advertisements, random facts and violence against sanity.

Viewer discretion is advised.

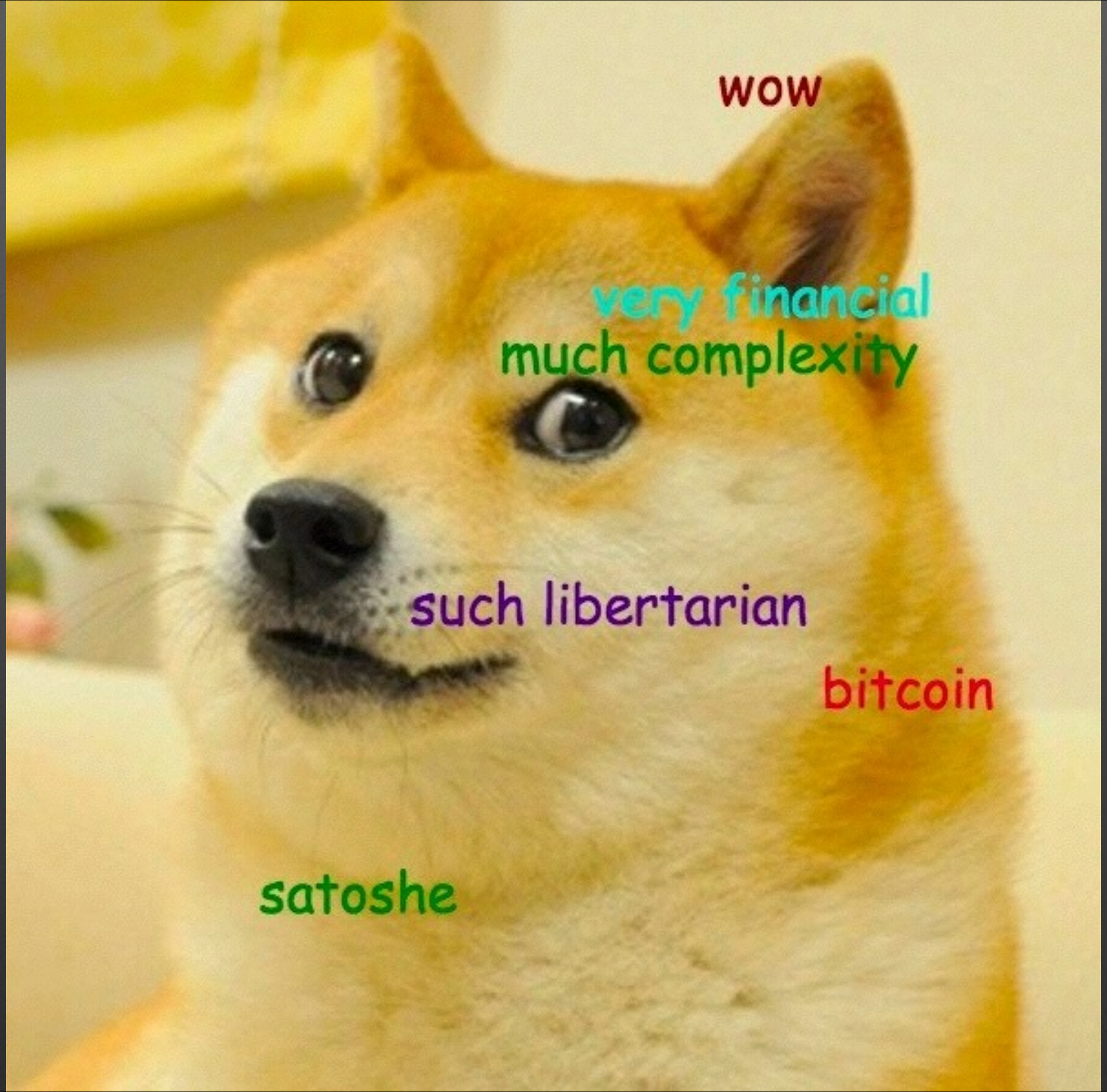
Our Story

- Small but passionate Erlang team
- Various open source and proprietary projects
- Had a look at Elixir at 0.5

⊙‿⊙

- Practiced it on a few more projects
- **and then…**

The bitcoin spring of 2013



The bitcoin spring of 2013

Shameless plug

Our mission is to proliferate Bitcoin globally

http://bex.io

Why does Bitcoin matter?

- Largest **untamperable**, massively replicated ledger
- Removes the middleman
- Enabling mistrusting peer-to-peer trust
- True digital age “programmable money”

Dangers associated with Bitcoin

- Irreversibility of transactions
- Theft (private keys, cold storage draining, etc.)
- Overall complexity
- Somewhat insufficient bitcoind API

Our bitcoin processing architecture

- Trusted “proxy” bitcoind
- All-elixir bitcoin protocol parser
- Pros: greater control and dynamism
- Cons: complexity of blocks and confirmations handling

Order book

- Our first product doesn’t feature one for both market and MVP reasons, **but…**
- we did two experimental implementations
  - linear
  - parallel

Linear

- Bids & Asks ordered_set ETS tables
- Match against existing orders, keep residual order
- ~300K orders in 2 minutes
- ~200 LOC

Parallel
(a lot more experimental!)

- A process per price level, listing bids & asks in ordered_set ETS tables
- On demand price level process creation
- Approximate initial level targeting, let levels move the order away if necessary
- ~300K orders in under 10 seconds
- ~200 LOC

We weren’t the first, but…

- We took the FIX XML specification and got it automatically parsed & compiled to:
  - decoder
  - encoder
  - (partially) validating “composer”

General architecture

- Amazon ELB HTTPS termination
- SockJS messaging “proxy” between client & backend
  (cowboy & sockjs)
- “Big rocket” application server with PostgreSQL
- Everything **dockerized**

Docker

- We got really tired of maintaining the same dependencies & configurations across our dev machines
- Tried out docker + vagrant. Hint: don’t use VirtualBox, go straight to VMWare! (but don’t expect it to be flawless either!)
- Obviously, we use docker containers in production

However,

- fundamentally, docker's strength is its weakness. Expect to have to transport large files.

- docker's registry is still a nightmare (randomly failing downloads, etc.)

- don't Dockerfile build every Erlang release (we made this mistake and we're likely to fix it)

Some hindsight thoughts

Things break & change
Embrace change!

- **HEAD** is your enemy but

  *“hold your friends close, but your **enemies closer**”*

  (also, make sure **your head** is **not** your enemy…)

- Incremental upgrades

  don’t delay them while in heavy development
  don’t do that before shipping releases,
  but **right after**

- **Expect to write your own libraries! (BYOL)**

Our experience so far

- Bex.io platform went from 0.9.4-dev to 0.12.0 (~300 lines CHANGELOG)
    - We'll go for R17 and 0.13.0 after full production deployment
- We are no longer afraid of using pre-release software

The sky is the limit

once you're on this path,
you're inspired to stay creative

- ExLogger
- Hypnotoad
- exdbi
- and many other things

ExLogger

Logging sucks and yet it has to be done...
ideally, the right way

My main thought was

“I like lager, but I am sorry, I don't want to format the data I log in the same process... or ever”

...I also got sidetracked by some other thoughts

https://github.com/ElixirWerkz/exlogger

ExLogger: What's in there?

```
L.info "User ${user} ordered an ${size}-size t-shirt", user: current_user, size: tshirt[:size]
```

- Clear use of arguments
- Caching gen_event lookup
- String rendering is done by handlers (if necessary)
- Compile-time exclusion (debug, verbose or whatever)
- Configurable default attributes
  (so that you can keep including your session data with every message!)
- Configurable backends (I/O, Splunk, etc.)

Hypnotoad

Everybody Loves Server Orchestration

https://github.com/elhypnotoad/hypnotoad

Hypnotoad

Recipe for a disaster
or a fine list of ingredients to stay awake for days

- Being furiously unhappy with existing major players
  That is, Chef, Puppet, SaltStack, Ansible... you name it
- Having a list of wants and wishes
- Being inspired by some smaller tools
- Being acutely aware of THE DEADLINE
- And, of course, "why is this all not Erlang?"

So, what does hypnotoad do?

(when not filming the next episode)

- Opens SSH connections to multiple hosts
  (and keeps 'em open)

- Lets you write modules that do just this:

test → run → test

- Resolves dependencies between modules

- Runs it all in parallel, reporting on everything **instantaneously**

and this is how it looks like

Hypnotoad

Bexng.Deployment.Docker.DockerLogin	{}	standby	success	Log
Apt.Source	{"name":"docker","url":"http://get.docker.io/ubuntu","distribution":"docker"}	standby	running	Log
Package	{"name":"lxc-docker-0.6.6","apt_source": {"name":"docker","url":"http://get.docker.io/ubuntu","distribution":"docker"}}	standby	pending	
Bexng.Deployment.Docker.Base	{}	standby	pending	
Bexng.Deployment.File	{"path":"/etc/init/bexng-bitcoind.conf","content":"description \"bexng bitcoind\"\nauthor \"Bex.io (Spawngrid, Inc.)\"\nstart on filesystem and started lxc-net and started docker\nstop on runlevel [!2345]\nrespawn\npre-start script\n docker rm bitcoind || true\nend script\npost-stop script\n docker stop bitcoind\n docker rm bitcoind\nend script\nscript\n docker run -t -v /volumes/bitcoind:/var/lib/bitcoin -name bitcoind quay.io/bexio/bitcoind\nend script","assert":true}	standby	success	Log
Bexng.Deployment.File	{"path":"/etc/init/bexng-pgsql.conf","content":"description \"bexng postgresql\"\nauthor \"Bex.io (Spawngrid, Inc.)\"\nstart on filesystem and started lxc-net and started docker\nstop on runlevel [!2345]\nrespawn\npre-start script\n docker rm postgresql || true\nend script\npost-stop script\n docker stop postgresql\n docker rm postgresql\nend script\nscript\n docker run -t -v /volumes/postgresql:/var/lib/postgresql -name postgresql quay.io/bexio/postgresql\nend script","assert":true}	standby	success	Log

```
## Bexng.Deployment.File.before_filter
## Bexng.Deployment.File.test
# (( sudo
test -f /etc/init/bexng-pgsql.conf && md5sum /etc/init/bexng-pgsql.conf
# ))
# EXIT CODE 1
## Bexng.Deployment.File.run
# (( sudo
test -f /etc/init/bexng-pgsql.conf
# ))
# EXIT CODE 1
# Uploading /etc/init/bexng-pgsql.conf
# Uploading /etc/init/bexng-pgsql.conf, permission denied, re-trying
# (( sudo
sudo -u $SUDO_USER sh -c 'echo $HOME'
# ))
/home/vagrant
# EXIT CODE 0
# Uploading /etc/init/bexng-pgsql.conf to /home/vagrant/.hypnotoad.37411224 (temporarily)
# Uploading /etc/init/bexng-pgsql.conf, moving from /home/vagrant/.hypnotoad.37411224
# (( sudo
mv -f /home/vagrant/.hypnotoad.37411224 /etc/init/bexng-pgsql.conf
```

Features

- No server software required beyond sshd
- Highly concurrent (especially if you tweak sshd's MaxSessions)
- Beautiful in-progress introspection with a built-in, first class **web UI**
- Built-in, flexible lock management
- SSH port forwarding
- Lets you write your scenarios in Elixir!

The Enablers

- **ssh** application from Erlang/OTP
- **gproc** — Extended process registry
- **cowboy** — Small, fast modular HTTP server
- the inspirational power of Elixir
- a few sleepless nights aka "*I never heard of NIH!*"

Somewhat interesting discoveries

- Erlang's SSH library is not good at letting you know about whether the channel was closed yet
  (There goes a reference to sshd's MaxSession). I might have just saved you a few hours...

- Apparently there is (was?) no good generic mutex/semaphore library for Erlang
  (Hint: you know what to do!)

- Insane things are possible in just a few days!

(here I am supposed to hydrate myself)

Bottom line:
Was it necessary to get crazy?

YES

NO

If you want to be a part of this insanity…

talk to me, we're hiring

Q&A SESSION

AMA!