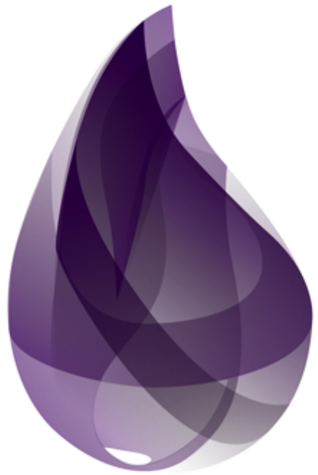

# elixir for the next 10 years

**boyd** multerer

# the basic exercise

know what problems you are trying to solve

know what trends and disruptions are happening around you

**build something relevant when it ships, not when you started**

# problems to solve

# servers

performance per cost

security

data processing

machine learning

communications

robustness

# devices (iot)

performance per cost

security

data processing

machine learning

communications

robustness

# example: game consoles

**perf per cost** // this is why you don’t just buy a PC

**security** // cheating & piracy - no physical security

**scalability** // hard, even though you’d think it would be necessary

**robustness** // keeping all games & consoles up to date is required

# trends and observations

# power drives cost

**servers** // power consumed means heat dissipated
power and heat cost real money

**devices** // power consumed impacts battery life
sensors use lots of power
radios use lots of power

**hw chosen to fit power and problem envelopes**

# security

**servers** // assume your servers are hacked
assume your databases are hacked

**devices** // tremendously tempting targets
both devices and data
no physical security

**this is an arms race and will not go away**

**it’s not paranoia if they are out to get you**

# FPGAs (you'll be friends soon)

think extremely large instructions

very low power for perf

not in addressable space

physically unclonable functions (PUFs)

not as expensive as you think.


Xilinx

# data processing

**servers** // huge amounts of data to process
currently a specialized field

**devices** // sensors!
local aggregation & processing
intermittent / expensive communication

**all the world's a GenStage and we are merely players**

# machine learning

**servers** // core related to data processing
currently a specialized field

**devices** // local decision making
sensor processing
security detection and action

# communication

**servers** // seems like a solved problem (at first)
latency affects datacenter decisions

**devices** // assume global wireless availability
expensive to operate

**satellites will make us rethink many things**

# robustness

**servers** // skating by on speed and reboots
opportunity to reduce cost

**devices** // coordinate multiple subsystems
deal with hardware failures
**hard to physically access devices!**

**danger of becoming the “Internet of Things that Don’t Work”**

# making bets

# beam

**original problems faced by telephony look familiar**

had to be efficient

designed scale to out

robustness - OTP

reduce programmer errors

30 years in production

**beam is your os**

# elixir

**all benefits of beam, erlang libraries plus…**

modern syntax, matching, macros
GenStage, Flow, much more

**separation between language and frameworks**

phoenix
nerves

# low level interop

realtime vs. soft-realtime

perf sensitive code

FPGA access

driver interop

important for security

important for power

# ui

user input

drawing primitives / overall model

GPU

# distribution

nerves boot images

signed/encrypted deployment

server boot images

revocation lists

signing authorities

more…

# example: cars

**robustness** // many sensors (drive train, cameras, proximity…)
many sub-systems (drive, ABS, **self-drive**…)

**security** // track individual's location, destinations, habits…
targeted attack opportunities

**communications** // already shipping with cell modems and connectivity
soon to have full-time satellite communications
servers required for mapping, traffic, more

# fin

**twitter** // @BoydMulterer

**github** // **/boydm**

# developer productivity

**trends** //

moore’s law enabled higher level languages

multi-core means parallel code

complexity is the enemy

only two kinds of code:

- performance sensitive
- everything else

# XNA example

**problem** games required low-level skill

small shops were locked out of the market

cost to build games increasingly prohibitive

# XNA example

**trends**

moore’s law enabled higher level languages

multi-core cpu encouraged parallel code

speed forgave many crimes

# XNA example

**bets** market for independent games would be huge

only two kinds of code:

- performance sensitive
- everything else

build pipelines key to cutting costs

# moore's law

**mobile driving chip design**

perf per watt

many low-powered cores

**tremendous implications on HOW we write code**

single threaded practices obsolete

most of the coding population is out of date