Erlang on Rumprun Unikernel

An Erlang/Elixir platform enabling the microservices architecture.

Neeraj Sharma
neeraj.sharma@alumni.iitg.ernet.in

Special Thanks

- Antti Kantee
  - Author of Rumprun Unikernel
- Fred Hebert
  - for cool rebar3 CRUD project plugin used in demo


{Mission} Platform for the microservices architecture

Guiding Principles

- Stable & Lightweight Core
- Maintainable
- Multi-Platform Support
- Release friendly
- Simple Workflow

Where it’s at?

- Erlang/OTP
- Elixir
- Single Node and Clustered
- KVM, Qemu, Virtualbox*
- x86, x86_64, ARM
- Xen PV*

Tested Erlang/OTP Releases

- R17
- R18

What does it look like?

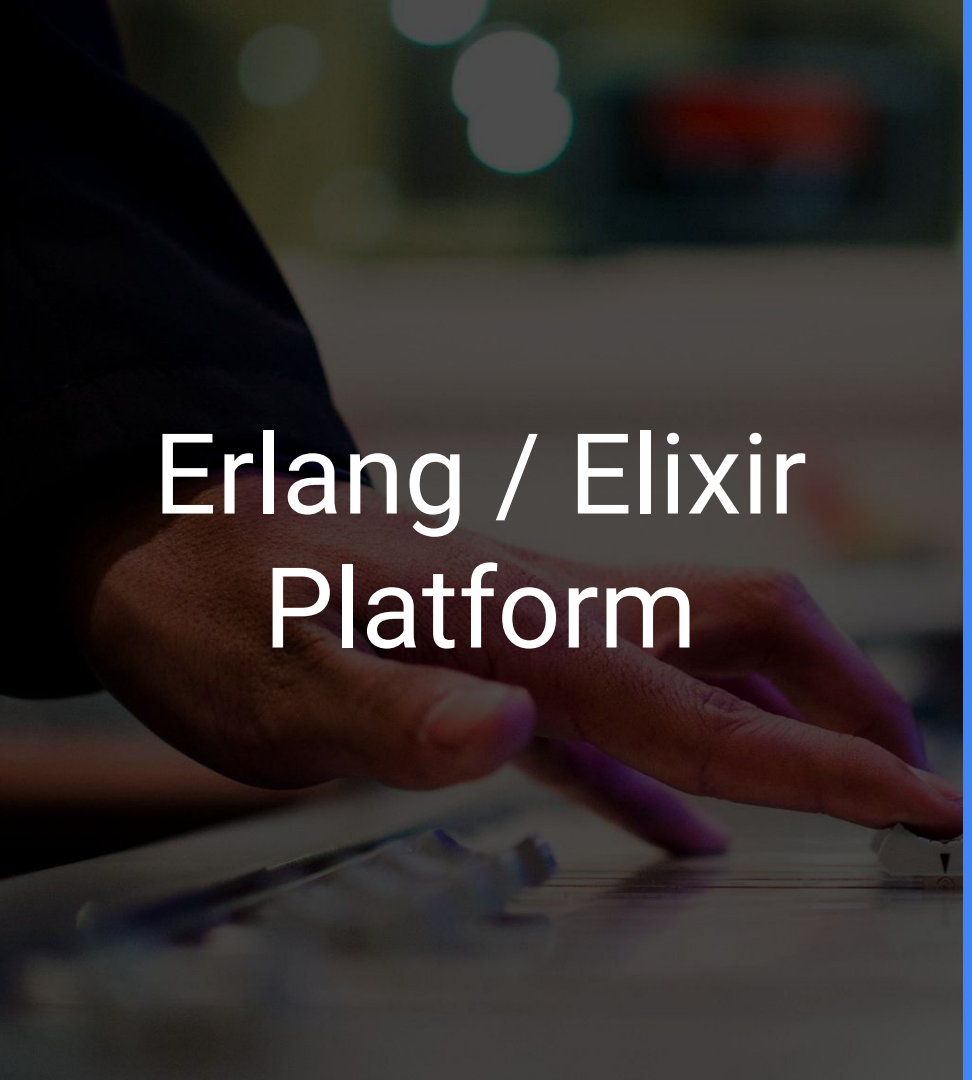
- Erlang/OTP BEAM VM builds to 6.3MB (stipped)
- Custom Cowboy Websocket demo builds to ~8MB
  - Boots in KVM under 2 seconds
- Hello Phoenix Elixir builds to ~19MB
  - Boots in KVM under 3 seconds
- The euc2016-cool-demo builds to ~12MB

Background

Rumprun Unikernel

The details

- A **“full stack”** solution
- Github dates back to 2013
- Builds on Rump kernels
  - early beginnings 2002 source usenix*
- Cooperative Threading
- Statically Linked
- **NO** fork / exec
- Unmodified NetBSD drivers
- **NO** virtual memory
- **Minimal** mmap support
- **NO** SMP

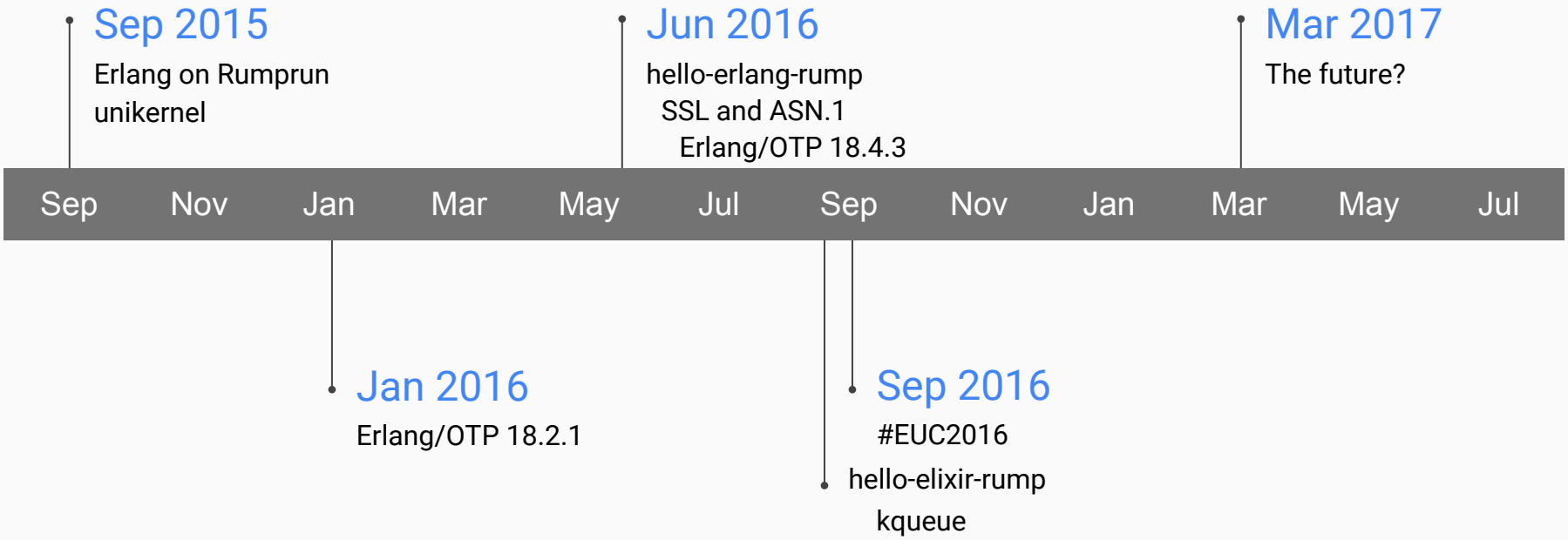


Erlang / Elixir Platform

Building for the next generation of services

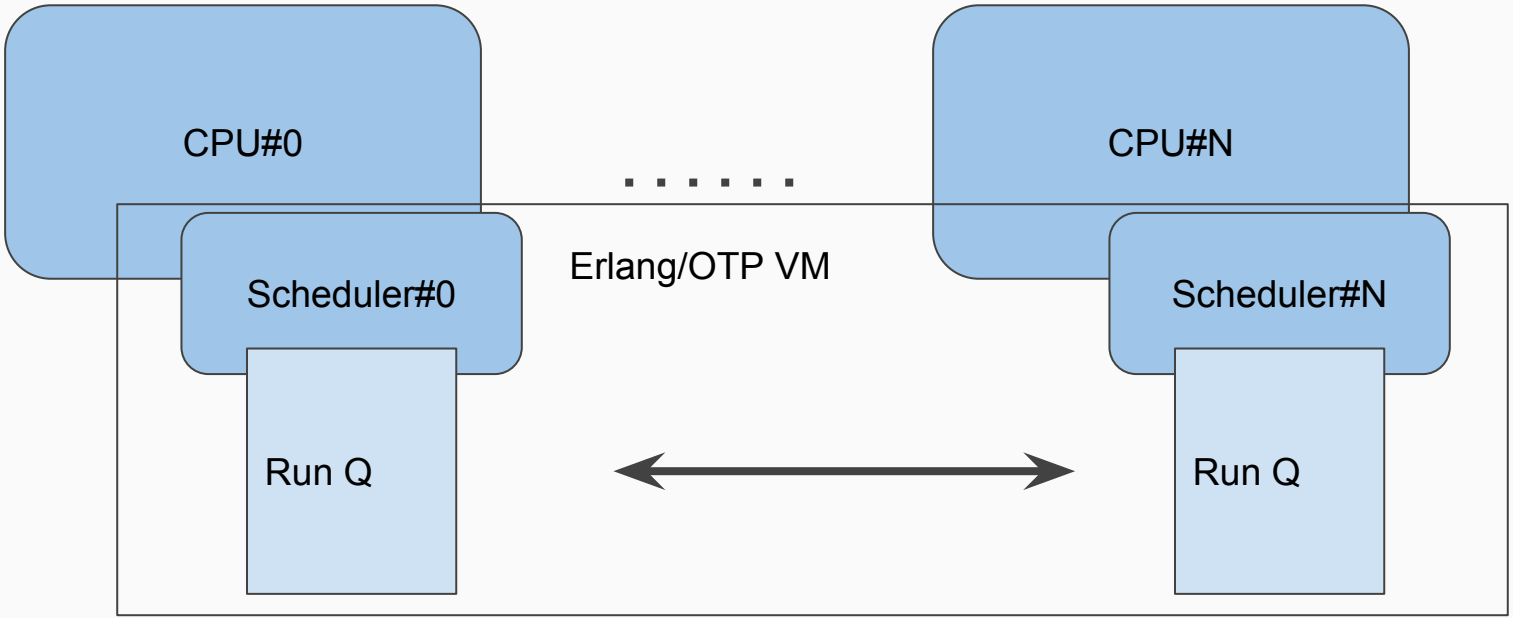
Milestones

Where are we?



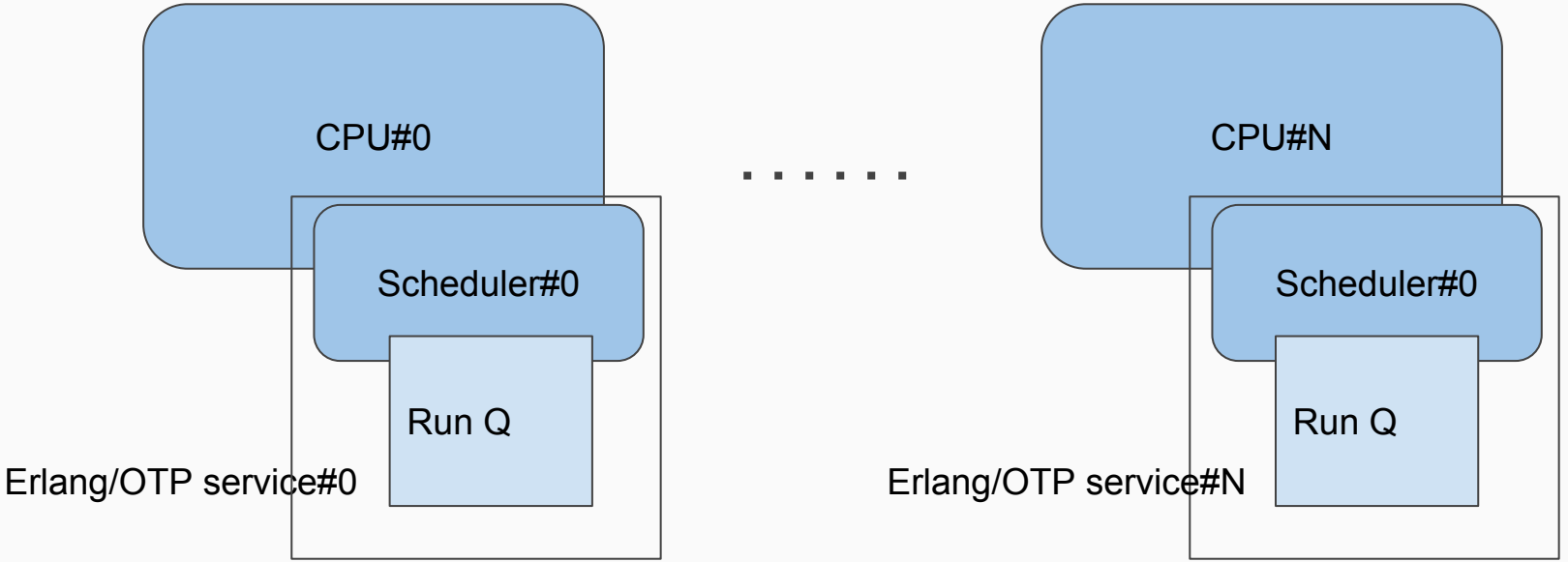
Erlang/OTP Multicore Architecture

for Erlang/OTP VM running on many cores



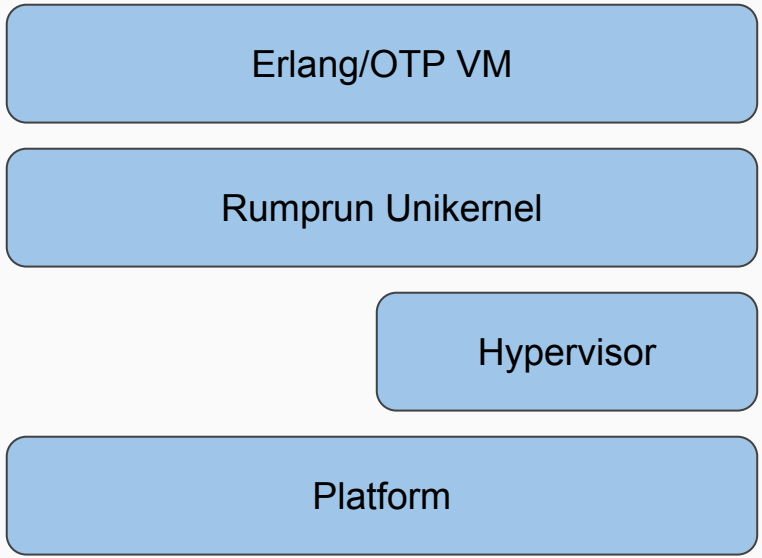
Erlang/OTP Microservice Architecture

(each Erlang/OTP VM without SMP)



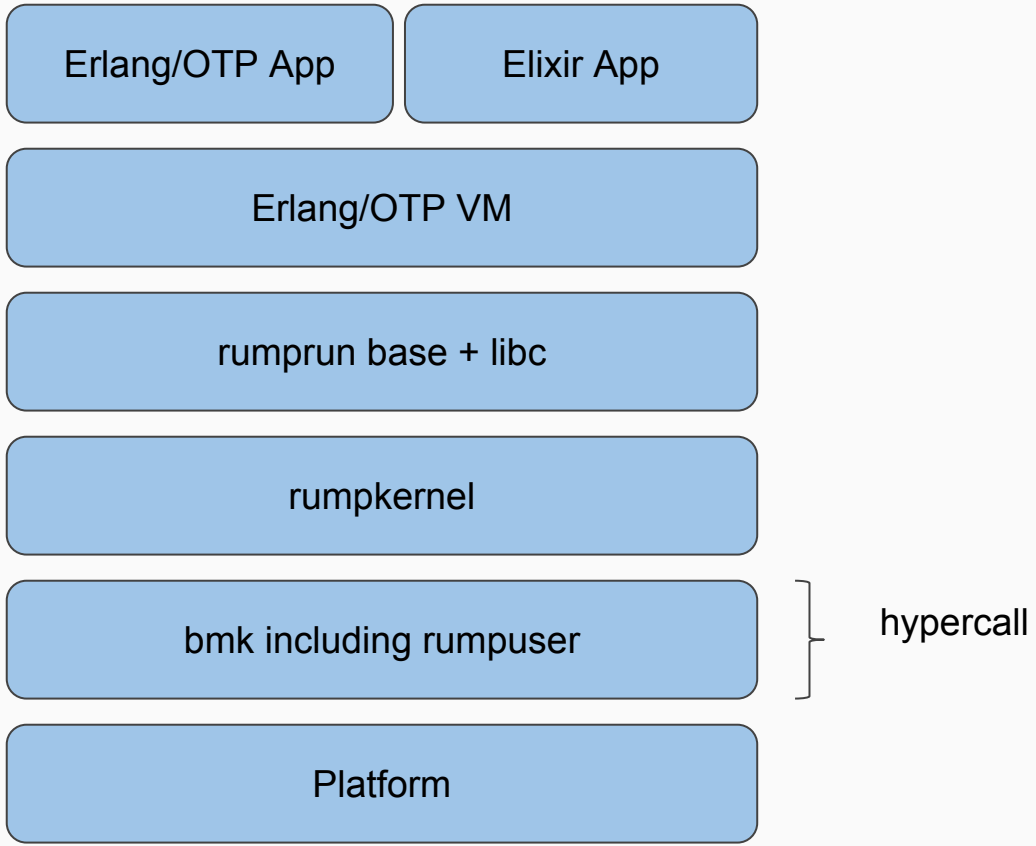
Erlang/OTP or Elixir Stack

The platform stack



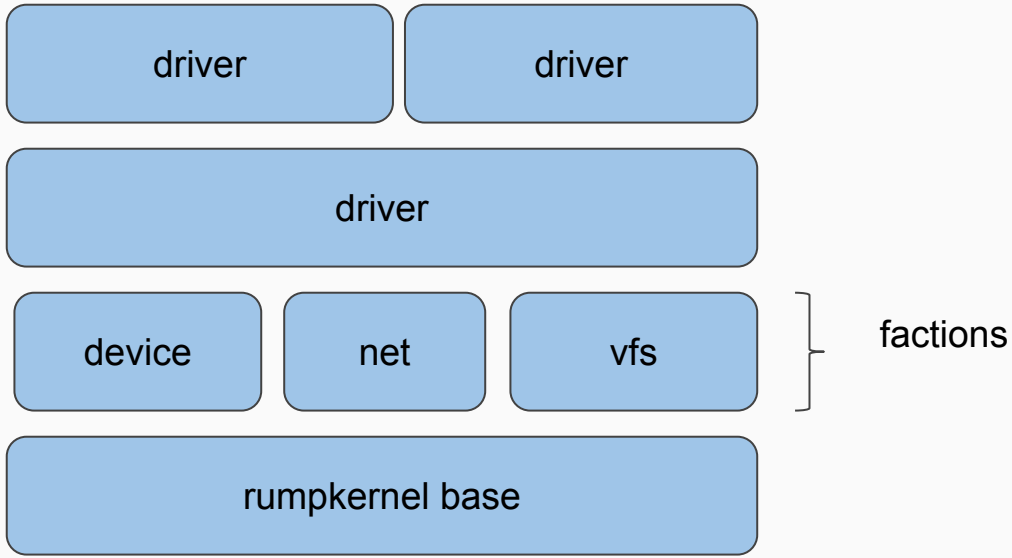
Erlang/OTP or Elixir Stack

The platform stack of Erlang/OTP or Elixir



Rump Kernel

A very high level view.



How it’s done?

The way to get started is to quit talking and begin doing.

- *Walt Disney*

The Build Process {how it works}

Step 1

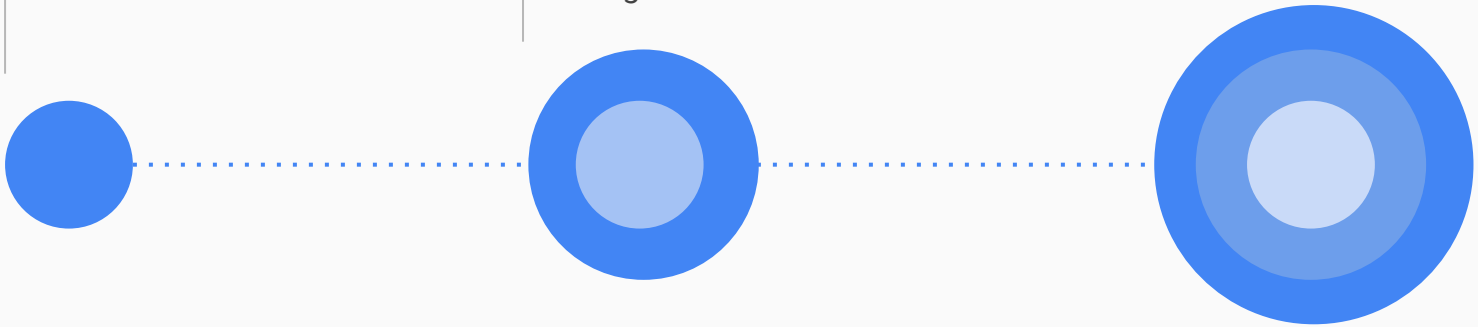
Build rumprun unikernel base

Step 2

Build rumprun-packages Erlang/OTP base

Step 3

Build Erlang/Elixir application and package everything together



Let’s Build Something - Dependencies

```
$ sudo apt-get install -y g++ libssl-dev
$ sudo apt-get install -y libncurses5-dev
$ sudo apt-get install -y genisoimage wget
```

Let’s Build Something - hello-elixir-rump

```
$ git clone https://github.com/neeraj9/hello-elixir-rump
$ cd hello-elixir-rump
$ make -f Makefile.rumprun .rumprun_packages_built
$ make -f Makefile.elixir .elixir_built
```

hello-elixir-rump - Setup Env, Hex and Rebar3

```
$ source setenv.sh
$ mix local.hex --force
$ mix local.rebar --force
```

hello-elixir-rump - Install Phoenix

```
$ mix archive.install
https://github.com/phoenixframework/archives/raw/master/phoenix_new.ez --force
```

hello-elixir-rump - Build App

$ cd hello_phoenix

$ MIX_ENV=prod mix do deps.get, compile, phoenix.digest, release

$ cd ..

hello-elixir-rump - App release artifact

```
hello_phoenix/
    rel/
        hello_phoenix/
            releases/
                0.0.1/
                    hello_phoenix.tar.gz
```

hello-elixir-rump - Create ukernel image

```
$ ./create-ukernel.sh
```

Artifact: hello_phoenix-0.0.1.iso

hello-elixir-rump - Setup Network Interface

```
$ sudo ip tuntap add tap0 mode tap
$ sudo ip addr add 10.0.120.100/24 dev tap0
$ sudo ip link set dev tap0 up
```

hello-elixir-rump - Run with KVM

```
$ PATH=build/rumprun/rumprun/bin:$PATH \
    ./run-elixir-vm --virt=kvm --iso=hello_phoenix-0.0.1.iso
```

Monitor the console output in parallel:

```
$ tail -f serial.log
```

Erlang/OTP Details

It's the little details that are vital. Little things make big things happen.

- *John Wooden*

erl_inetrc

Inet configuration

```
%% -- ERLANG INET CONFIGURATION FILE --
{file, hosts, "/opt/erlang/hosts"}.
%% do not monitor the hosts file
{hosts_file, ""}.
{file, resolv, "/opt/erlang/resolv.conf"}.
%% set resolv_conf as well, otherwise things don't work
{resolv_conf, "/opt/erlang/resolv.conf"}.
%% enable EDNS version 0
{edns,0}.
%% cache_size default value is 100
{cache_size, 50}.
%% lookup method is both file and dns
{lookup, [file, dns]}.
```

Patches

Deviating from the Erlang/OTP standard release

- 815.patch
  - by Peter Lemenkov
  - EPMD in Erlang
- erlonrump-18.0.patch
  - Don’t build shared objs in crypto
- have_SCTP.patch
  - source: NetBSD changeset
  - Merged by Igor Galić
  - No sctp delayed ack time support
- Pcre-build-avoidance.patch
  - Ensures Erlang use its own pcre library

Erlang/OTP Build

Building Erlang/OTP for host and Rumprun unikernel (cross compilation)

Configured without

- odbc, wx, debugger, et, javac, hipe, ic, orber, pman, toolbar, tv, webtool, typer, observer, cos*

Cross Compiled with

- Static NIF, drivers
- SSL (LibreSSL default)
- Kqueue

Cross Compiled without

- termcap

Erlang/OTP Built

Artifacts of the Erlang/OTP build

Artifacts:

- build/host_erlangdist
  - Usable on Host OS
- build/erlangdist
  - Cross compiled for Rumprun target
- beam.hw.bin
  - Erlang VM Kernel Image

The Erlang VM has prefix /opt/erlang

Rumprun Unikernel + Erlang / Elixir

Rumprun Unikernel + Erlang / Elixir = Microservices Architecture

Rumprun Unikernel gives “micro”, while Erlang / Elixir complements with “services” to complete the Microservice story.

Why Rumprun Unikernel?

- Stable - uses NetBSD anykernel architecture
- Cache friendly
- Avoids double preemptive scheduler (cooperative threading) - Erlang VM to the rescue
- No context switching
- No virtual memory - lesser complexity when you don’t need one
- Keep what you need - configurable to include/exclude components
- No fork/exec - Erlang VM provides processes

Why Rumprun Unikernel? ...

- Small codebase* - minimal attack surface
- Frowns upon shared objects - Forces static linking and loading
- Devoid of bloat - Erlang/OTP provides bulk of tooling
- Sysproxy* - remote syscalls for the debugging/detailed tooling
- License permissive of commercial use

The Future?

- Docker Integration*
- Rebar3 Integration*
- Revisit EPMD*
- Tune Memory Architecture - Can we get rid of mmap?
- Async Thread Pool - Should it work?
- Test Coverage - It has to begin sometime
- Build Integration - A better integration with rebar3
- Clean Clustering - A better way to enable clustering


{Demo} inevitably fails, but then I am running stateless* !
github.com/neeraj9/euc2016-cool-demo

Thanks

- Neeraj Sharma