ERLANG

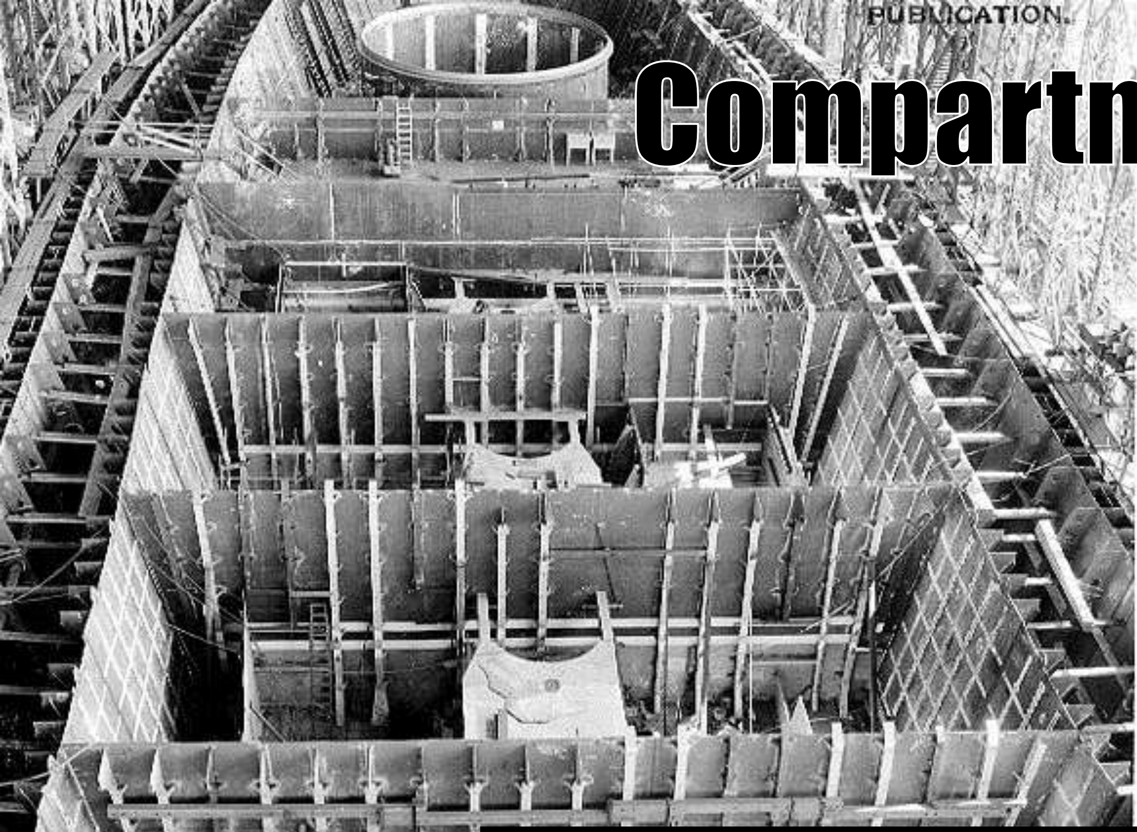
on

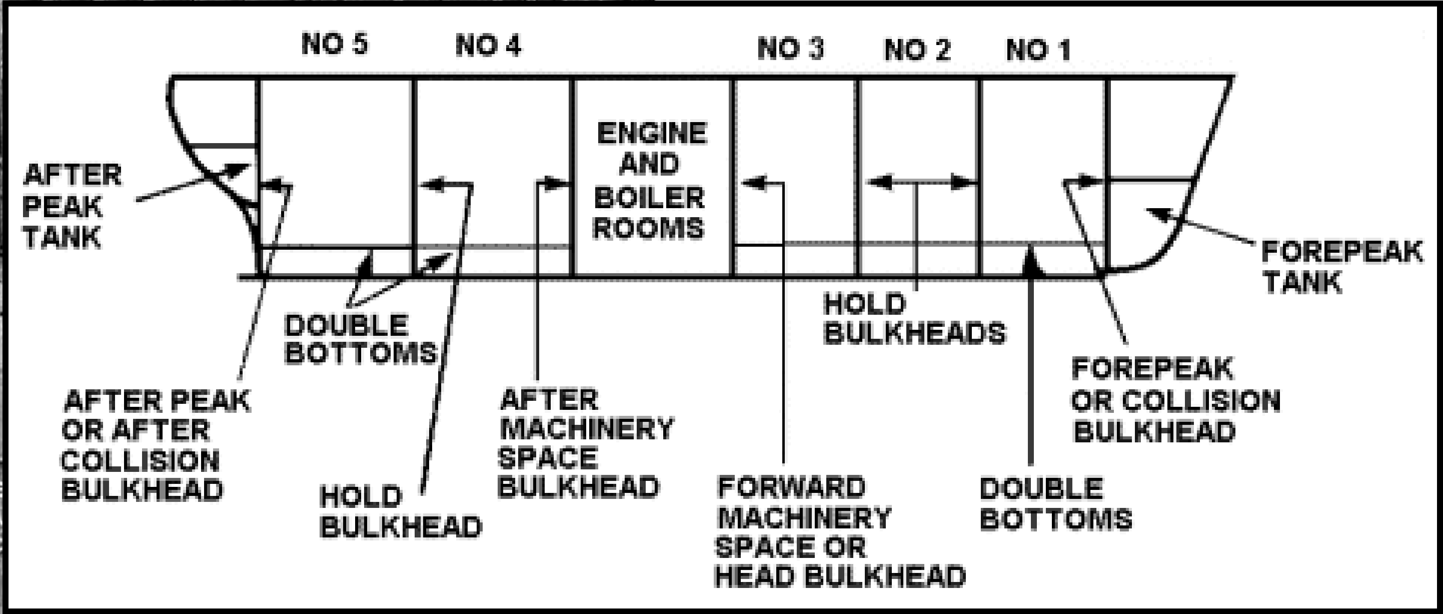
OSv

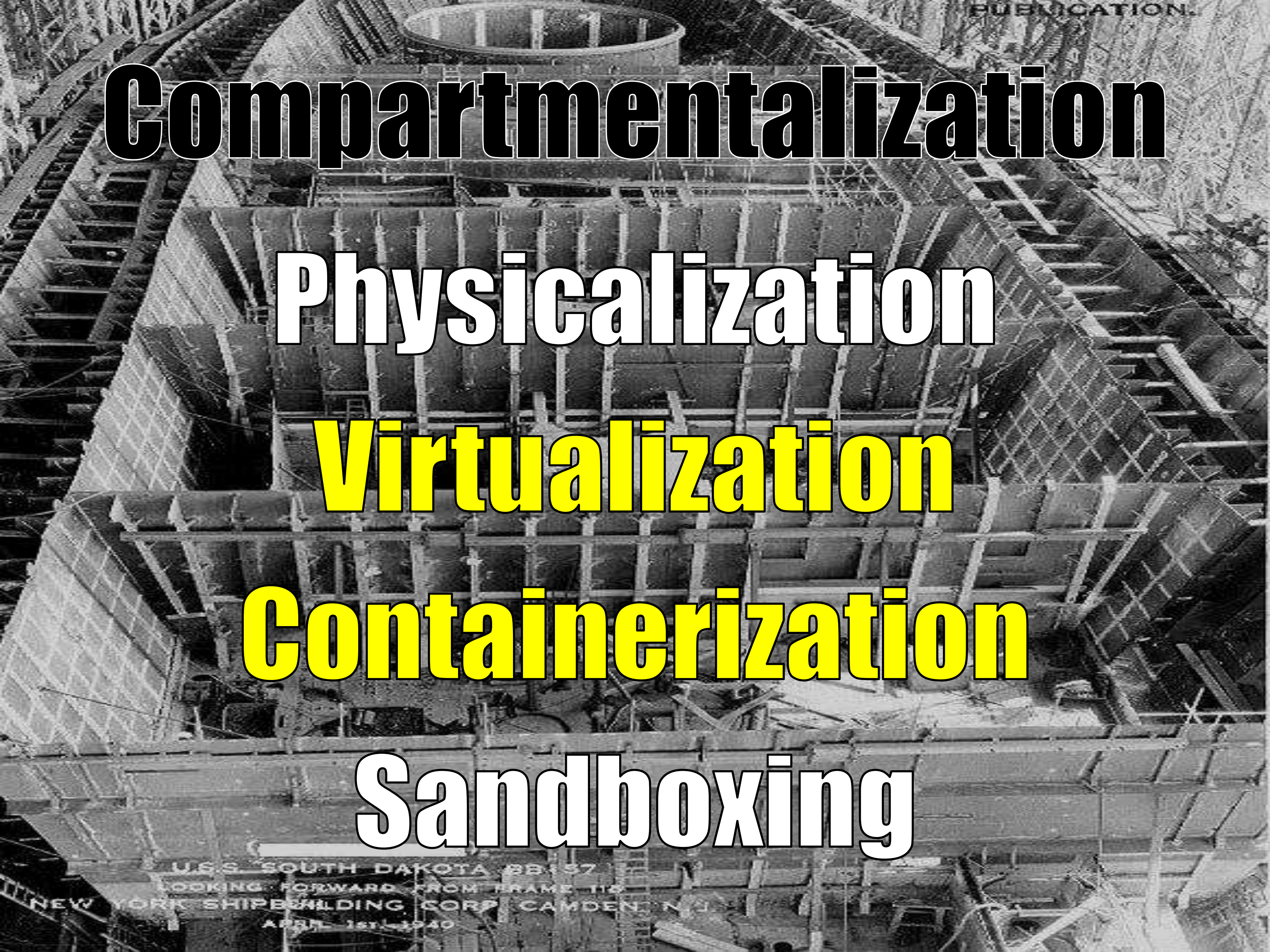
```
$ whoami

Name:    Zvi Avraham
E-mail:  zvi@nivertech.com
```

Compartmentalization

/ˈkɒm.
pɑː(ɹ)t.
mɛnţl̩.
aɪˌzeɪ.
ʃən/




NO 5
NO 4
NO 3
NO 2
NO 1
ENGINE AND BOILER ROOMS
AFTER PEAK TANK
FOREPEAK TANK
DOUBLE BOTTOMS
HOLD BULKHEADS
FOREPEAK OR COLLISION BULKHEAD
AFTER PEAK OR AFTER COLLISION BULKHEAD
HOLD BULKHEAD
AFTER MACHINERY SPACE BULKHEAD
FORWARD MACHINERY SPACE OR HEAD BULKHEAD
DOUBLE BOTTOMS


PUBLICATION
U.S.S. "SOUTH DAKOTA" BB-57
LOOKING FORWARD FROM FRAME 118
NEW YORK SHIPBUILDING CORP. CAMDEN, N.J.
APRIL 1st, 1940

Compartmentalization

Physicalization

Virtualization

Containerization

Sandboxing

Physicalization

- The opposite of Virtualization
- dedicated machines
- no virtualization overhead
- no noisy neighbors
  - nobody stealing your CPU cycles, IOPS or bandwidth
  - your EC2 instance may have a Netflix “roommate” ;)
- Mostly used by ARM-based public clouds
- also called *Bare Metal* or HPC clouds

Sandbox –
a virtual container in which untrusted code can be safely run

Sandbox examples: ZeroVM & AWS Lambda

based on Google Native Client: A Sandbox for Portable, Untrusted x86 Native Code

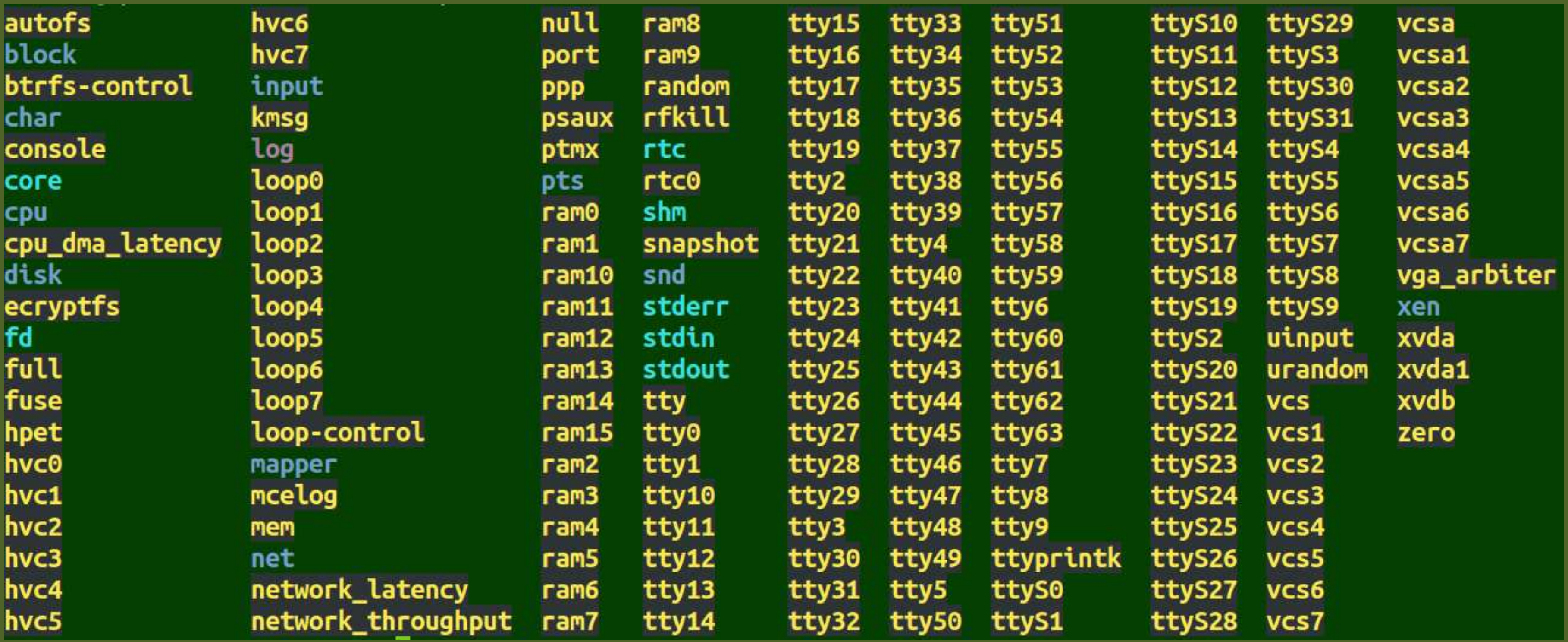
Compartmentalization in terms of Virtualization

Physicalization	No Virtualization
Virtualization	HW-level Virtualization
Containerization	OS-level Virtualization
Sandboxing	Userspace-level Virtualization*

Cloud runs on virtual HW

Does the OS on your Cloud instance still supports floppy drive?

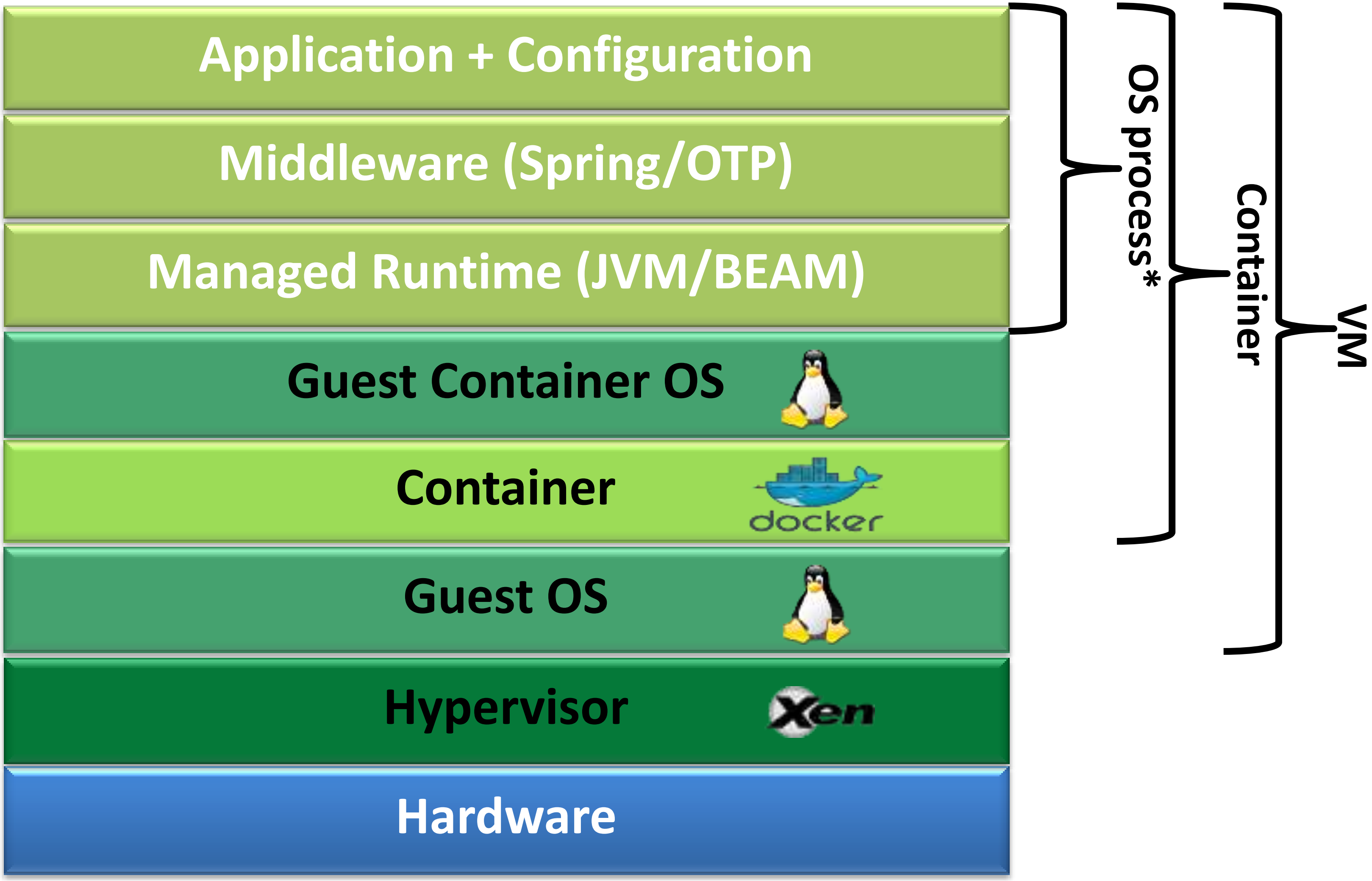
$ ls /dev
on Ubuntu 14.04 AWS EC2 instance



```
autofs           hvc6                null   ram8      tty15  tty33  tty51      ttyS10  ttyS29   vcsa
block            hvc7                port   ram9      tty16  tty34  tty52      ttyS11  ttyS3    vcsa1
btrfs-control    input               ppp    random    tty17  tty35  tty53      ttyS12  ttyS30   vcsa2
char             kmsg                psaux  rfkill    tty18  tty36  tty54      ttyS13  ttyS31   vcsa3
console          log                 ptmx   rtc       tty19  tty37  tty55      ttyS14  ttyS4    vcsa4
core             loop0               pts    rtc0      tty2   tty38  tty56      ttyS15  ttyS5    vcsa5
cpu              loop1               ram0   shm       tty20  tty39  tty57      ttyS16  ttyS6    vcsa6
cpu_dma_latency  loop2               ram1   snapshot  tty21  tty4   tty58      ttyS17  ttyS7    vcsa7
disk             loop3               ram10  snd       tty22  tty40  tty59      ttyS18  ttyS8    vga_arbiter
ecryptfs         loop4               ram11  stderr    tty23  tty41  tty6       ttyS19  ttyS9    xen
fd               loop5               ram12  stdin     tty24  tty42  tty60      ttyS2   uinput   xvda
full             loop6               ram13  stdout    tty25  tty43  tty61      ttyS20  urandom  xvda1
fuse             loop7               ram14  tty       tty26  tty44  tty62      ttyS21  vcs      xvdb
hpet             loop-control        ram15  tty0      tty27  tty45  tty63      ttyS22  vcs1     zero
hvc0             mapper              ram2   tty1      tty28  tty46  tty7       ttyS23  vcs2
hvc1             mcelog              ram3   tty10     tty29  tty47  tty8       ttyS24  vcs3
hvc2             mem                 ram4   tty11     tty3   tty48  tty9       ttyS25  vcs4
hvc3             net                 ram5   tty12     tty30  tty49  ttyprintk  ttyS26  vcs5
hvc4             network_latency     ram6   tty13     tty31  tty5   ttyS0      ttyS27  vcs6
hvc5             network_throughput  ram7   tty14     tty32  tty50  ttyS1      ttyS28  vcs7
```

- 64 teletype devices?
- 32 serial ports?
- Sound?
- VGA?

“It’s DUPLICATED on so many LAYERS”



We run Single App per VM

We run in Single User mode

So, why do we still use Operating Systems designed for the desktops in the Cloud?

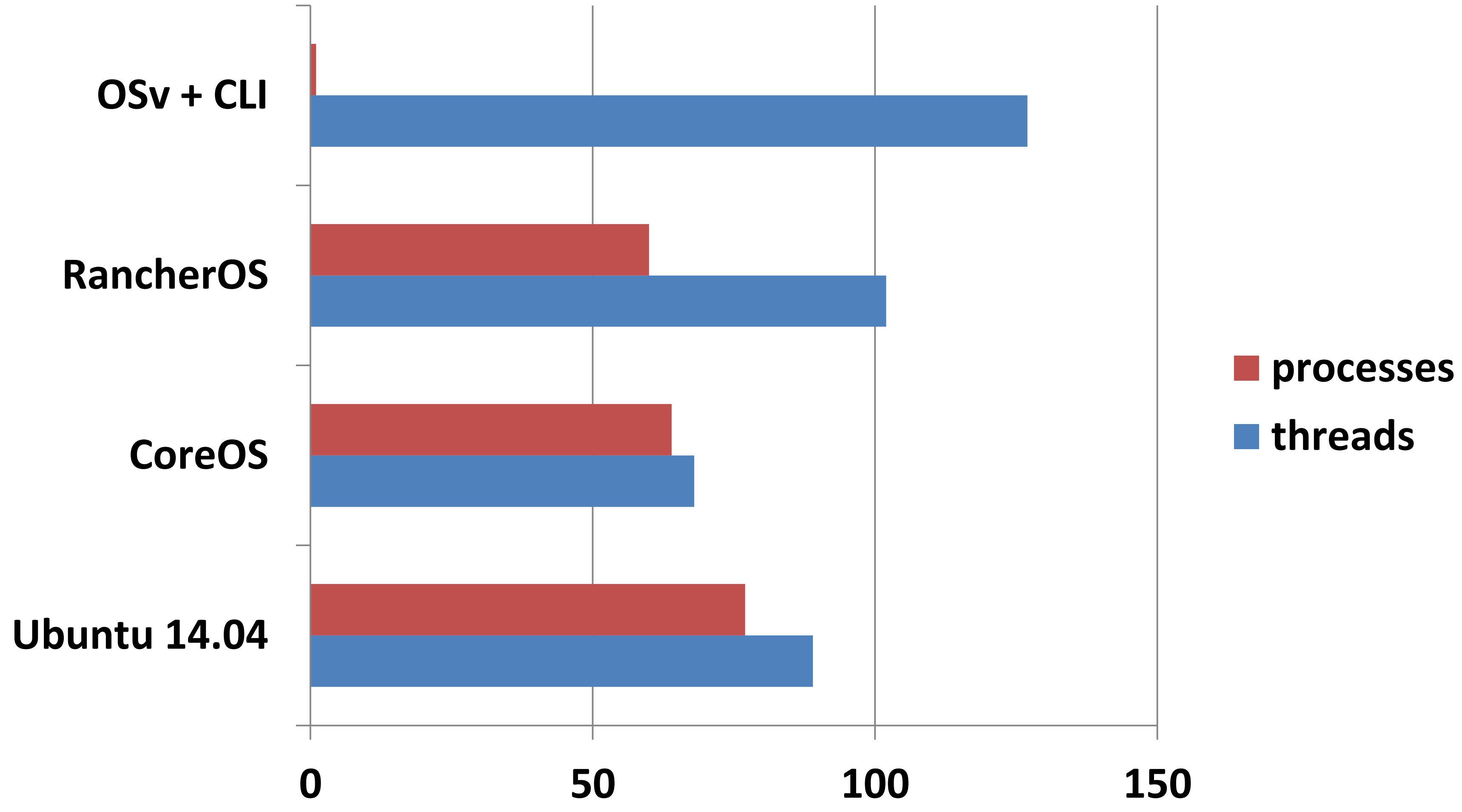
Minimalistic Linux OSes

- Embedded Linux versions
- DamnSmall Linux
- Linux with BusyBox

Min. Linux OSes for Containers
JeOS – "Just Enough OS"

- CoreOS
- RancherOS
- RedHat Project Atomic
- VMware Photon
- Intel Clear Linux
- Hyper

# of Processes and Threads per OS



What’s Unikernel?

- A “library” operating system
- kernel library linked with your application
- A kernel that can only support one process
- Single Address Space
- No kernel / userspace separation
- *“Unikernels: Library Operating Systems for the Cloud”*
    - **http://anil.recoil.org/papers/2013-asplos-mirage.pdf**

What’s Anykernel?

- Programming discipline for kernel code reuse
- Capabilities
  - NetBSD filesystems as Linux processes
  - User-space TCP/IP stack
- *“The Design and Implementation of the Anykernel and Rump Kernels”, Antti Kantee*
  - **http://book.rumpkernel.org/**

Unikernel + Anykernel

- Unikernels originally were designed to run on top of Hypervisor
- Now combined with Anykernel / Rump kernel ideas, some unikernels, like ***MirageOS*** or ***LING VM*** can run on ***Bare Metal***

A new Erlang platform – LING – runs directly on Xen.

You get less administrative headaches,
better security and performance.

LING is highly-compatible with Erlang/OTP.

LING understands .beam files.

Develop on Erlang/OTP –
deploy using LING.

http://erlangonxen.org/

LING: NOT JUST ERLANG ON XEN PORTING LING TO ARM/MIPS MICROCONTROLLERS

Viktor Sovietov

Embedded Erlang is Real

LING: NOT JUST ERLANG ON XEN PORTING LING TO ARM/MIPS MICROCONTROLLERS

LING is an Erlang platform with minimal requirements with respect to its software environment. Until now LING
virtualised x86. We managed to port LING to ARM (Raspberry Pi). The port to PIC32 (MIPS) microcontrollers is in the
discusses the challenges of running Erlang on bare metal and potential benefits of LING/Erlang as a basis for embedde

Talk objectives:

- Describes of how Erlang is applicable to develop IoT and embedded applications;

Target audience:

- Erlang developers, technology entrepreneurs interested in IoT;

OSv

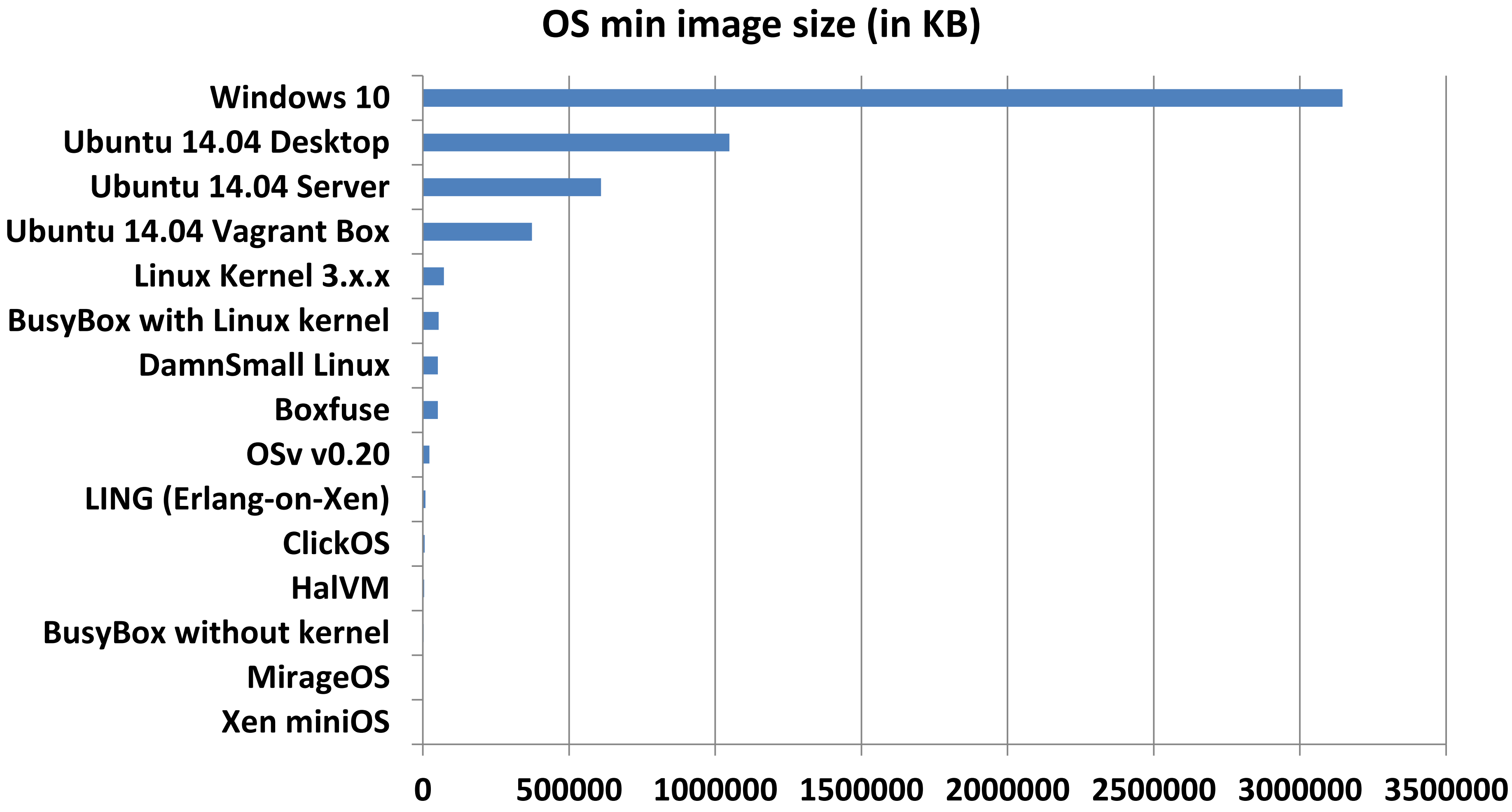
designed for the cloud

http://osv.io

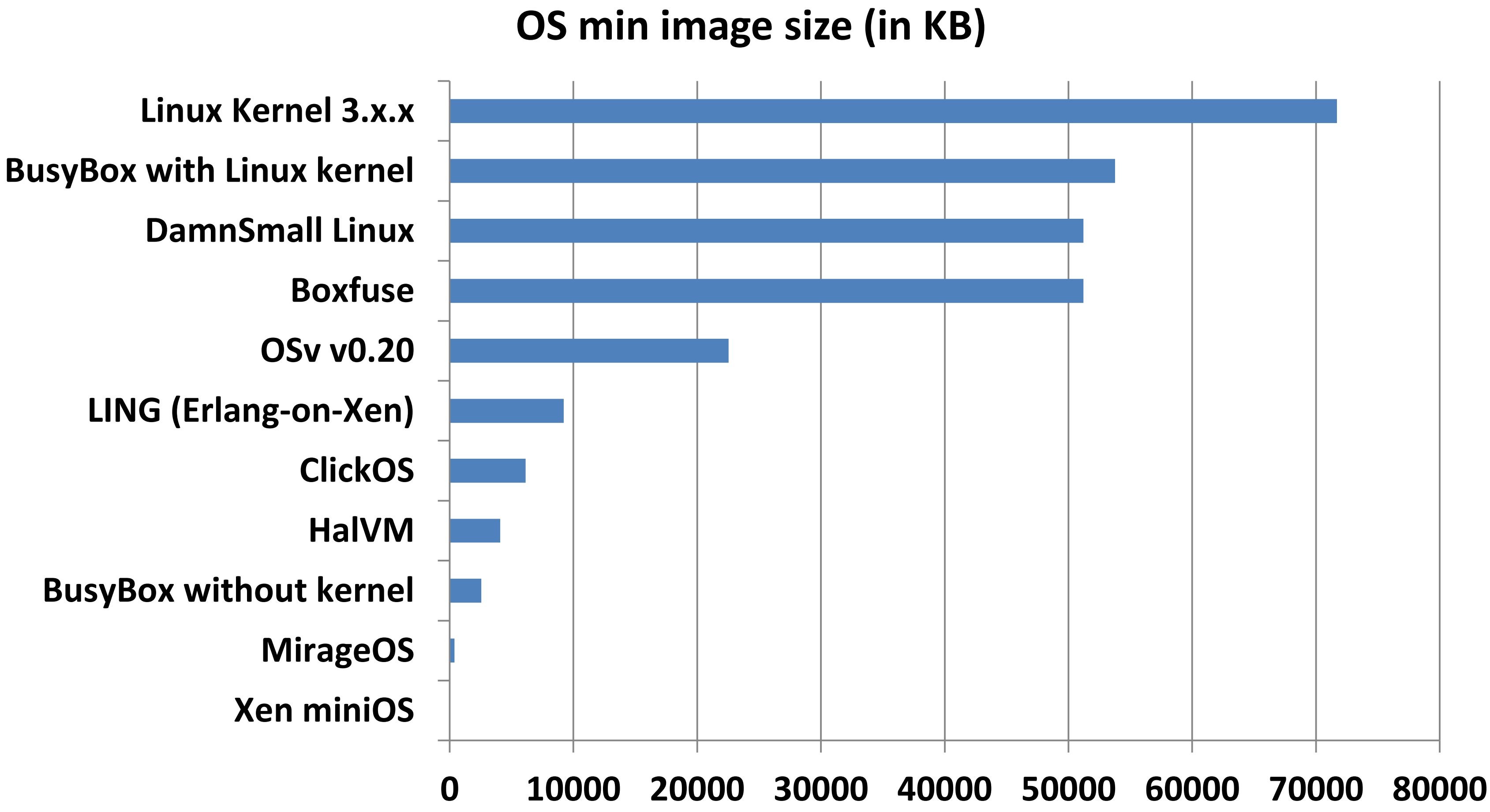
Unikernel Projects

Name	Target
MirageOS	OCaml
HalVM	Haskell
ClickOS	Click DSL
Clive	Go (inpired by Plan 9 & CSP)
Boxfuse	JVM
LING VM (Erlang-on-Xen)	Erlang
OSv	POSIX*, JVM, High Performance non-POSIX API

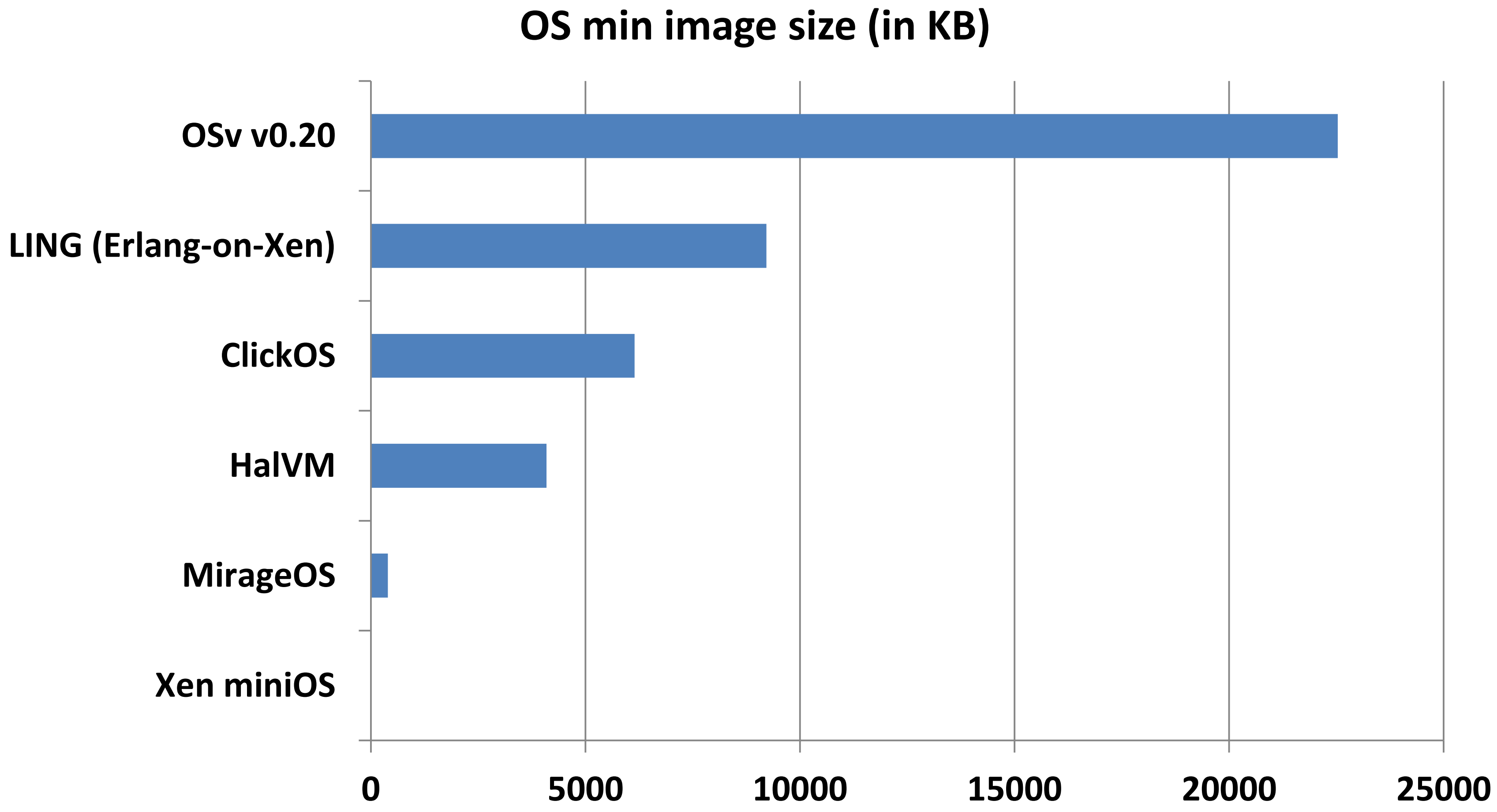
Windows 10 & Ubuntu – are outliers



OSv is half-way between Unikernels & Linux Kernel



OSv is “fat” Unikernel or Anykernel-ish



Specialized Unikernels

- Compile to very small images
- Boot very fast
- requires writing all code in Higher Level statically typed language, like OCaml or Haskell
- Hard/impossible to port existing code
- Very secure: tiny attack surface

Anykernel-ish or “fat unikernels”

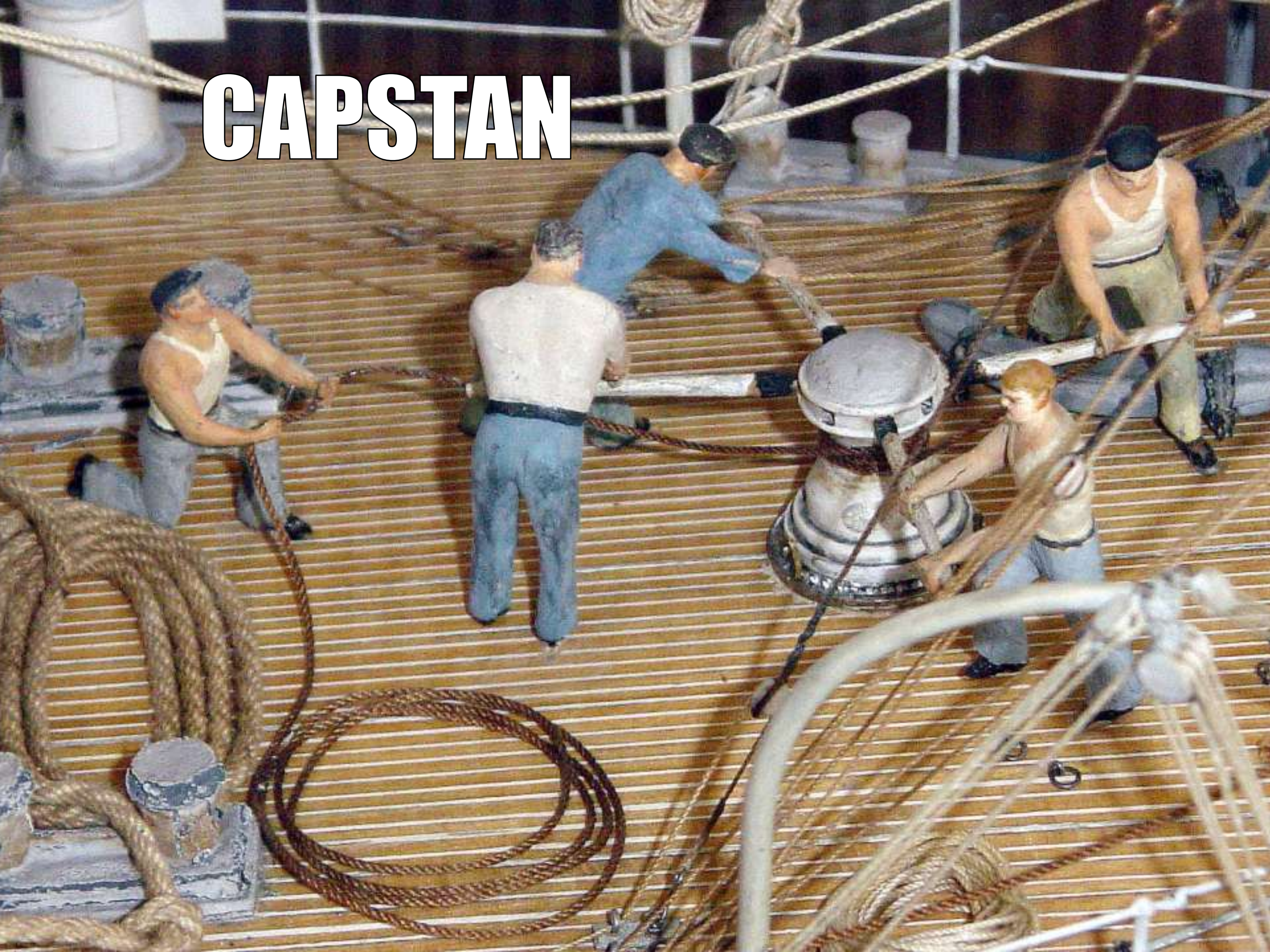
- Larger images (still much smaller that Linux)
- Longer boot times (much faster that Linux)
- Larger attack surface (much smaller that Linux)
- Easier to port existing code
- More pragmatic solution overall

YO UNIKERNEL SO FAT
IT SEMI-LINUX ABI COMPATIBLE
imgflip.com

OSv is “Fat” Unikernel / Anykernel-ish

- OSv sometimes called ***“fat” unikernel***
- since OSv images are a little bit larger than for other unikernels
- It also called ***anykernel***-ish, since it
  - run on top of multiple hypervisors
  - provide TCP/IP stack and filesystem
- Small price to pay for LINUX ABI compatibility

CAPSTAN



Capstan – Docker-like CLI for OSv

```
$ capstan
NAME:   capstan - pack, ship, and run applications in light-weight VMs
USAGE: capstan [global options] command [command options] [args...]
VERSION: v0.1.8
COMMANDS:
info           show disk image information
import         import an image to the local repository
pull           pull an image from a repository
rmi            delete an image from a repository
run            launch a VM. You may pass the image name as the first arg
build          build an image
images, i      list images
search         search a remote images
instances, I   list instances
stop           stop an instance
delete         delete an instance
help, h        Shows a list of commands or help for one command
```

Download & Run OSv+CLI image

```
$ capstan run cloudius/osv
Downloading cloudius/osv/index.yaml...
170 B / 170 B [====================] 100.00 %
Downloading cloudius/osv/osv.qemu.gz...
21.78 MB / 21.78 MB [==============] 100.00 %
Created instance: cloudius-osv
OSv v0.20
eth0: 192.168.122.15
pthread_setcancelstate() stubbed
/#
```

ERLANG + ☁ =

Match
Made
in
Heaven

OSv

designed for the cloud

ERLANG

on

OSv

Containers add new layers
vs
Unikernels remove layers

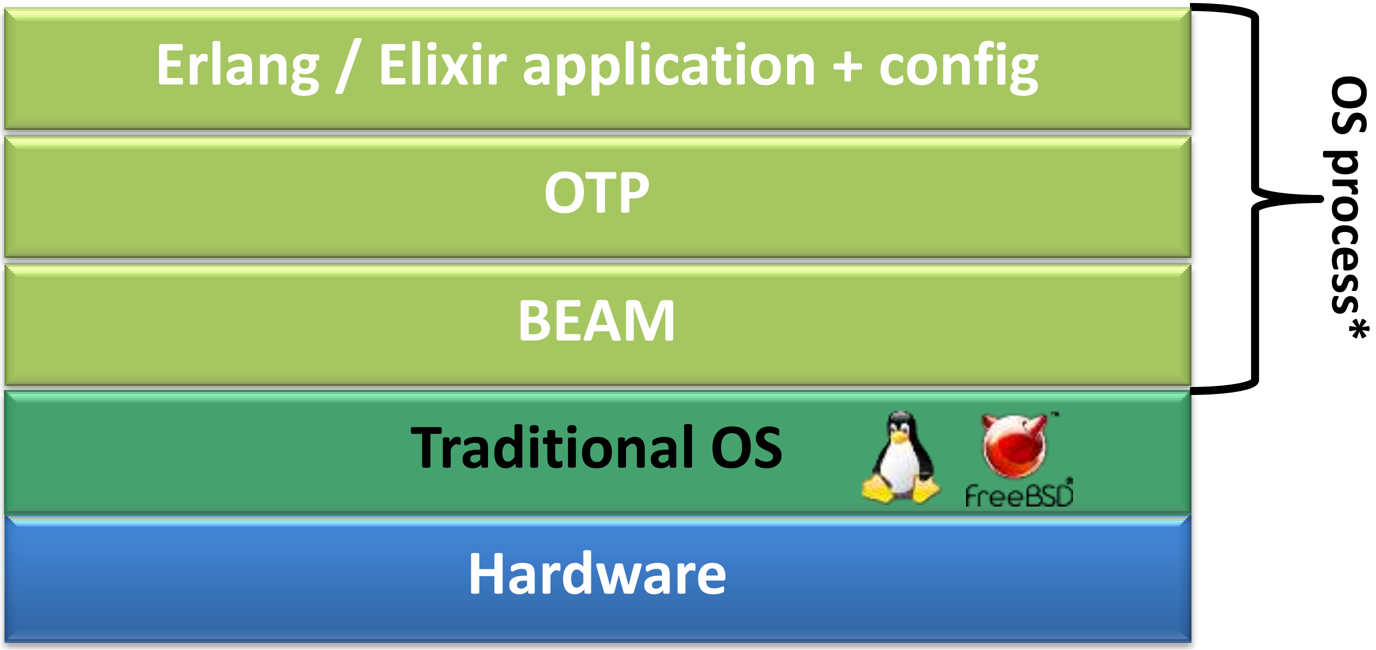
"All problems in computer science can be solved by another level of indirection“

– David Wheeler

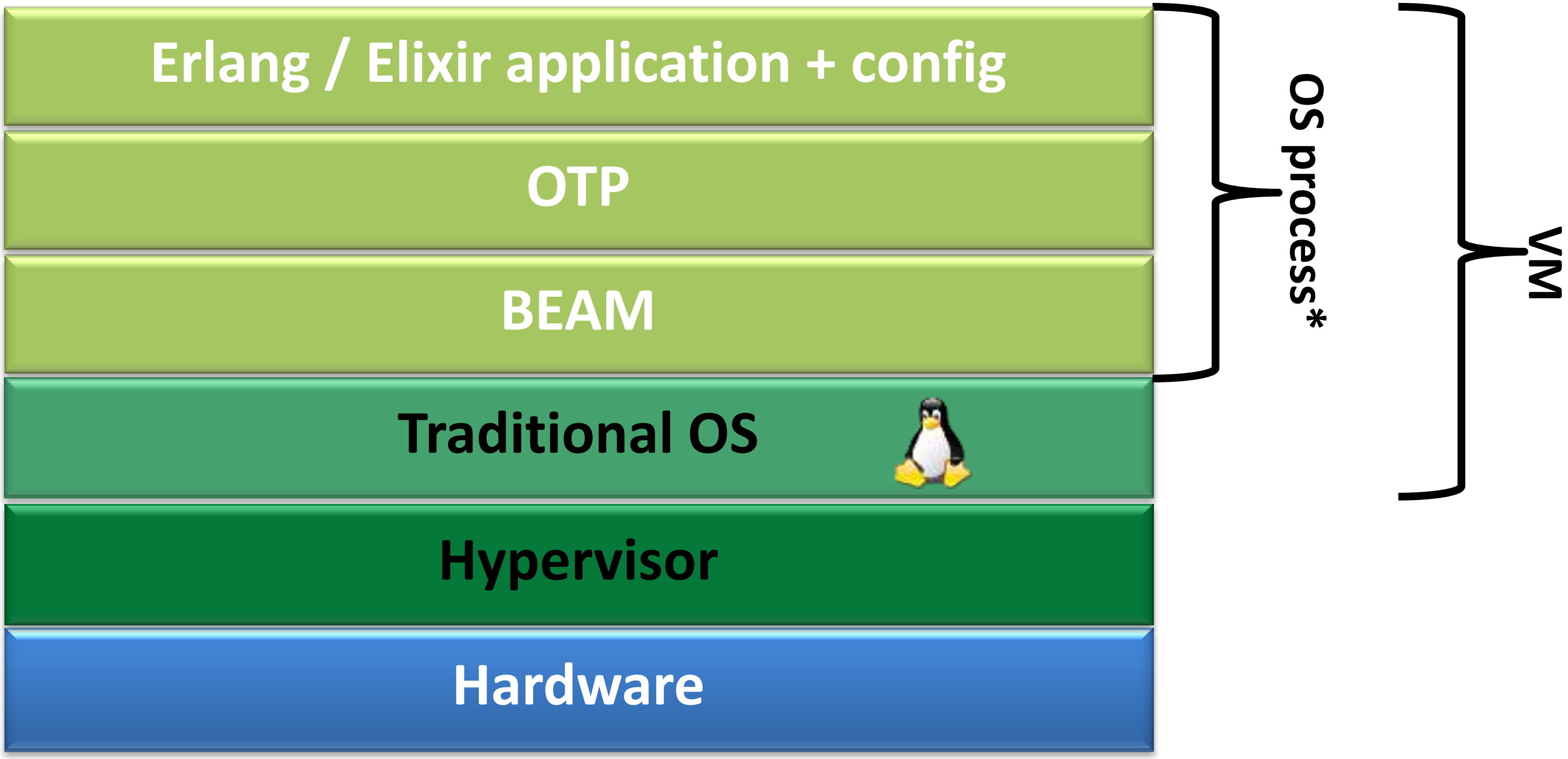
"...except for the problem of too many layers of indirection.“

– Kevlin Henney

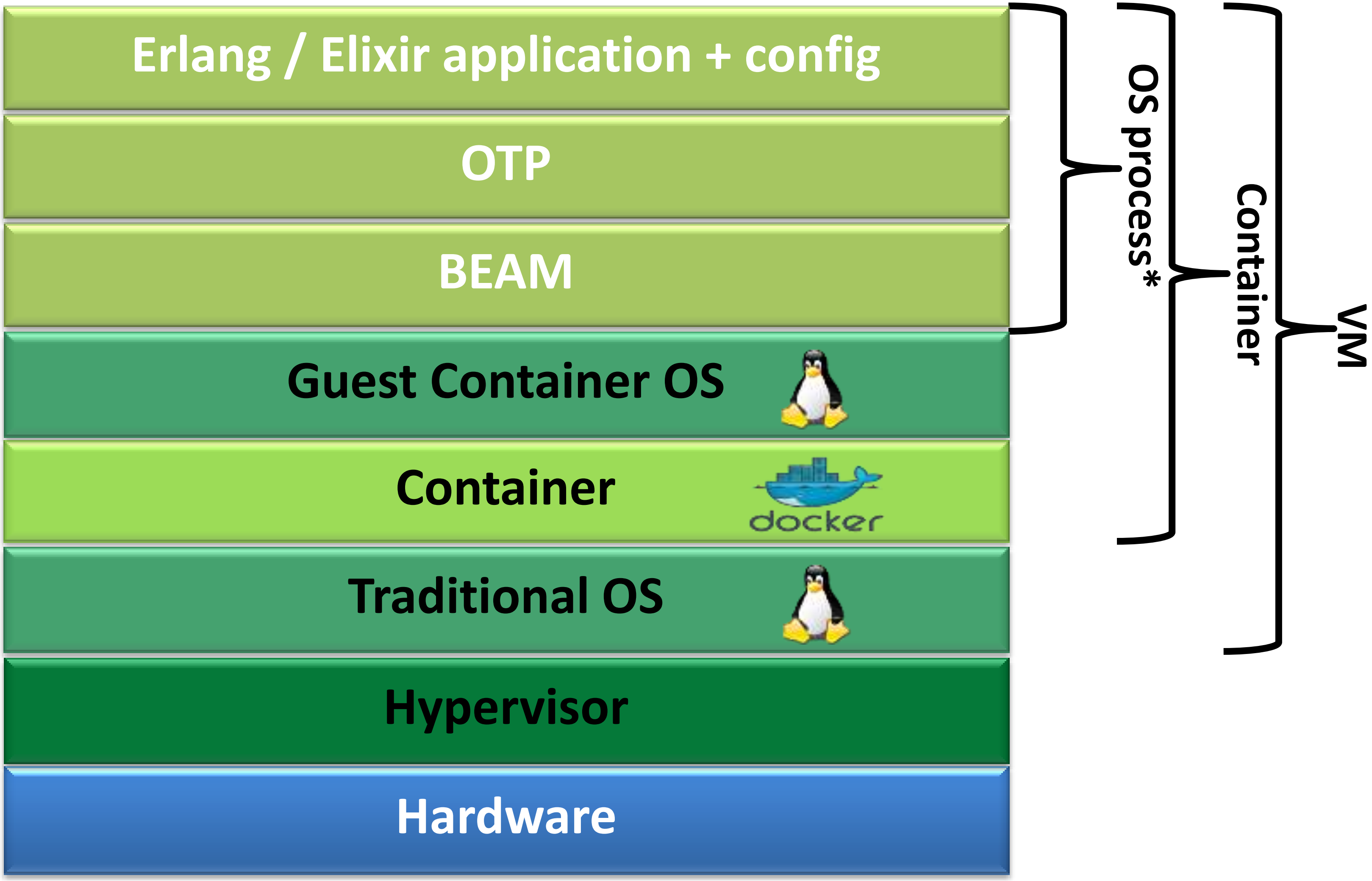
Erlang on Physical Machine



Erlang in VM


Erlang / Elixir application + config
OTP
BEAM
Traditional OS
Hypervisor
Hardware
OS process*
VM

Container in VM



Erlang / Elixir application + config

OTP

BEAM

Guest Container OS

Container

Traditional OS

Hypervisor

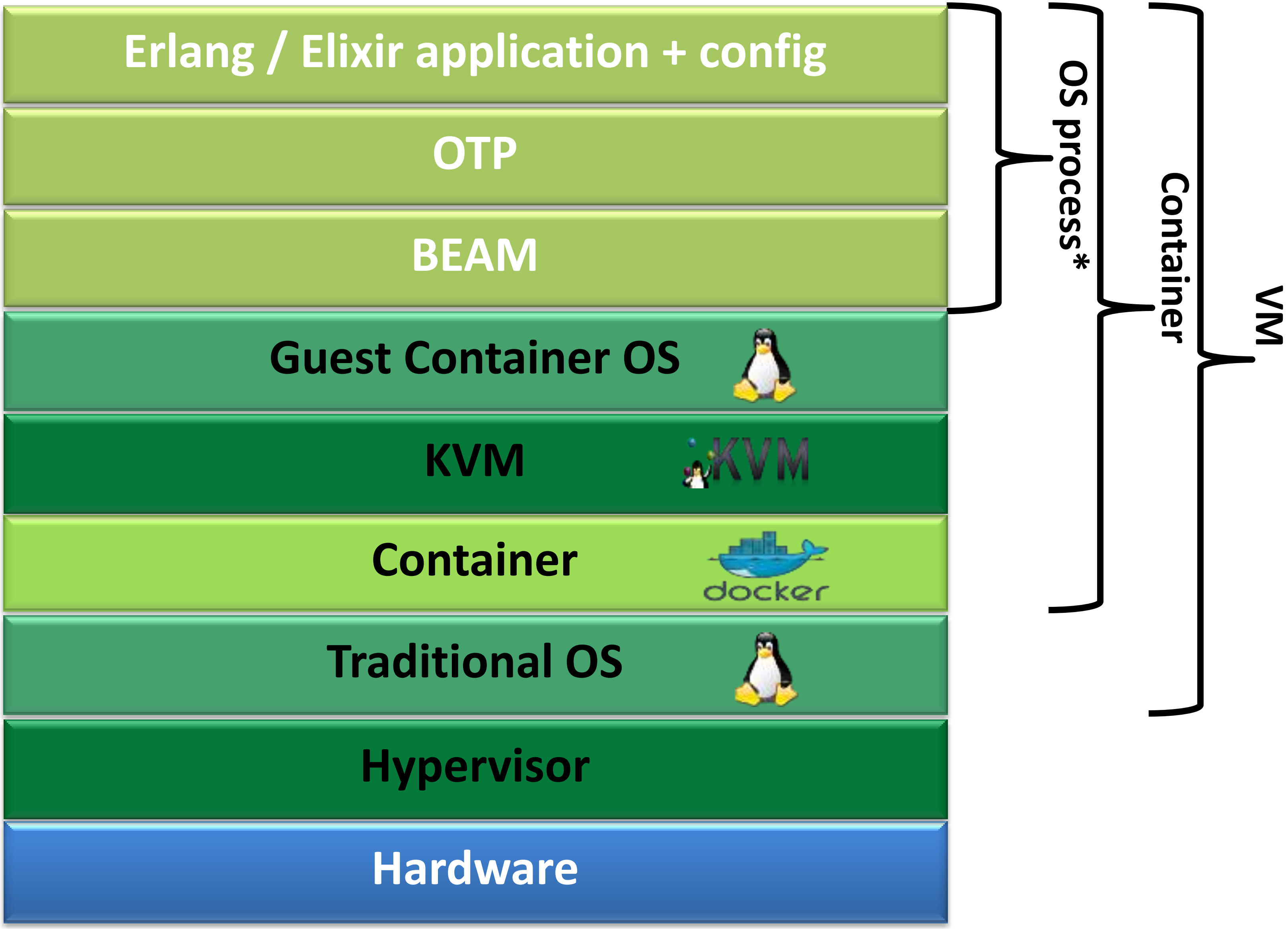
Hardware

OS process*

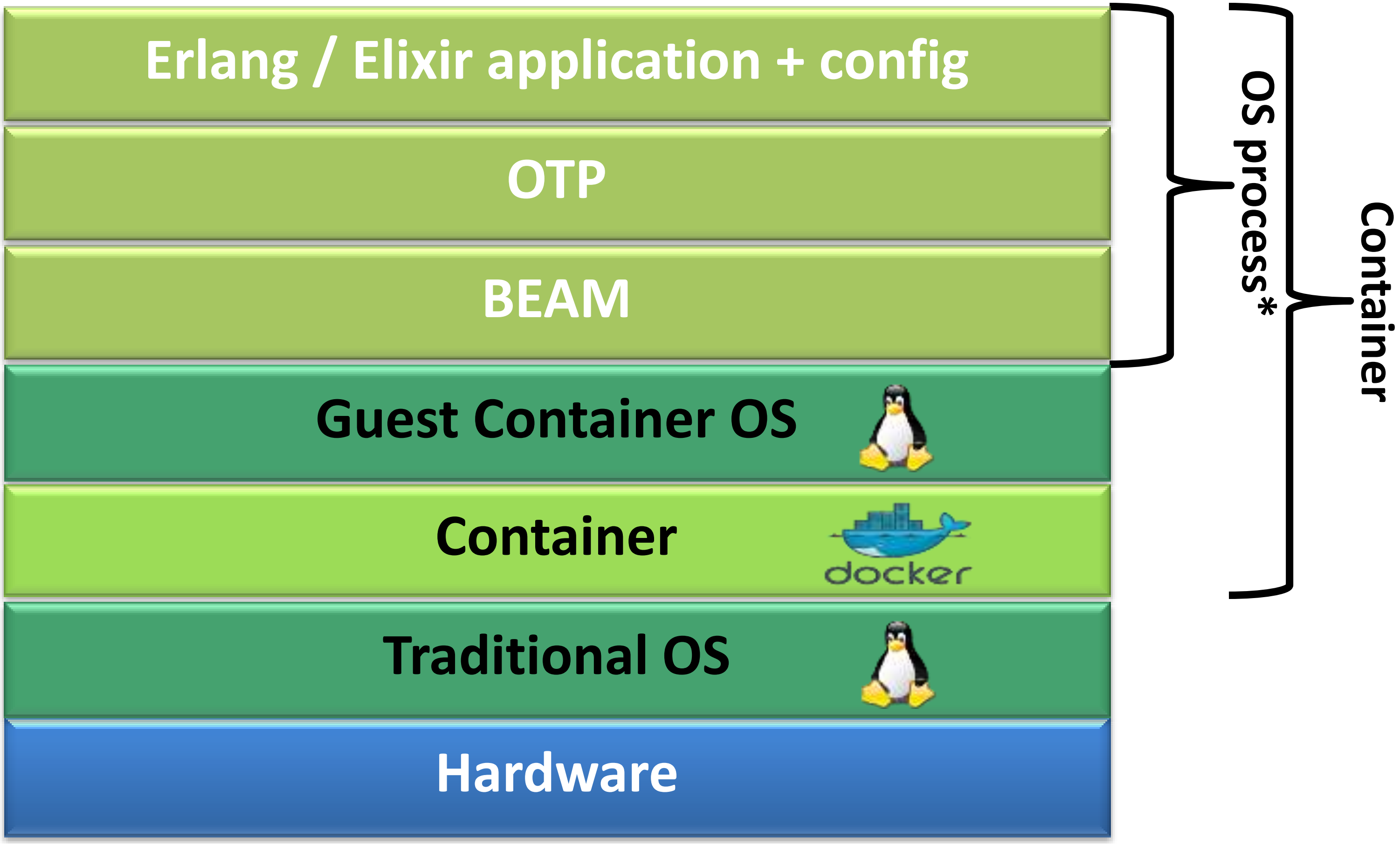
Container

VM

Multitenancy - VM in Container in VM



Container on Physical Machine



Erlang / Elixir application + config

OTP

BEAM

Guest Container OS

Container

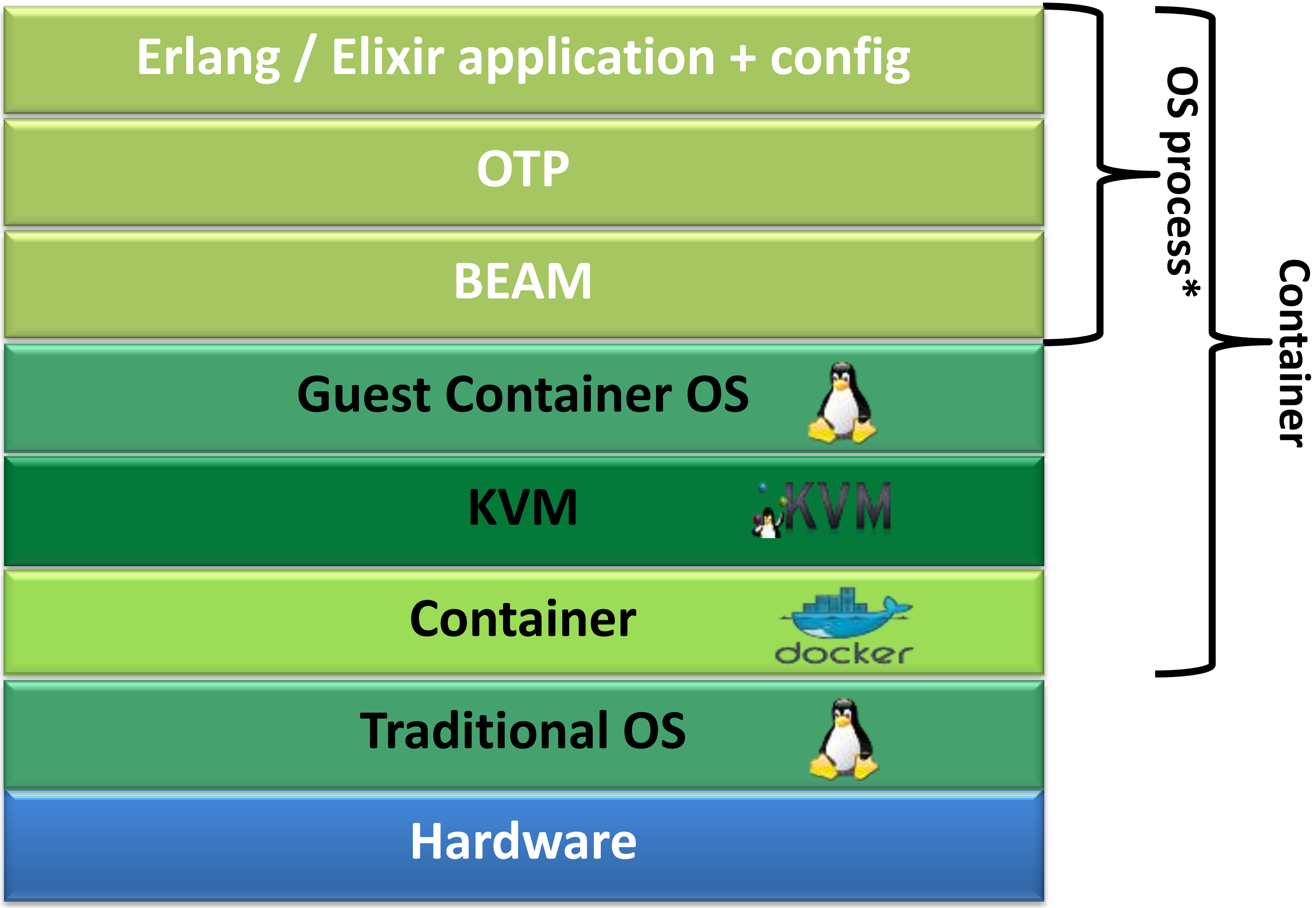
Traditional OS

Hardware

OS process*

Container

VM in Container on Physical Machine for Multitenancy



Erlang / Elixir application + config

OTP

BEAM

Guest Container OS

KVM

Container

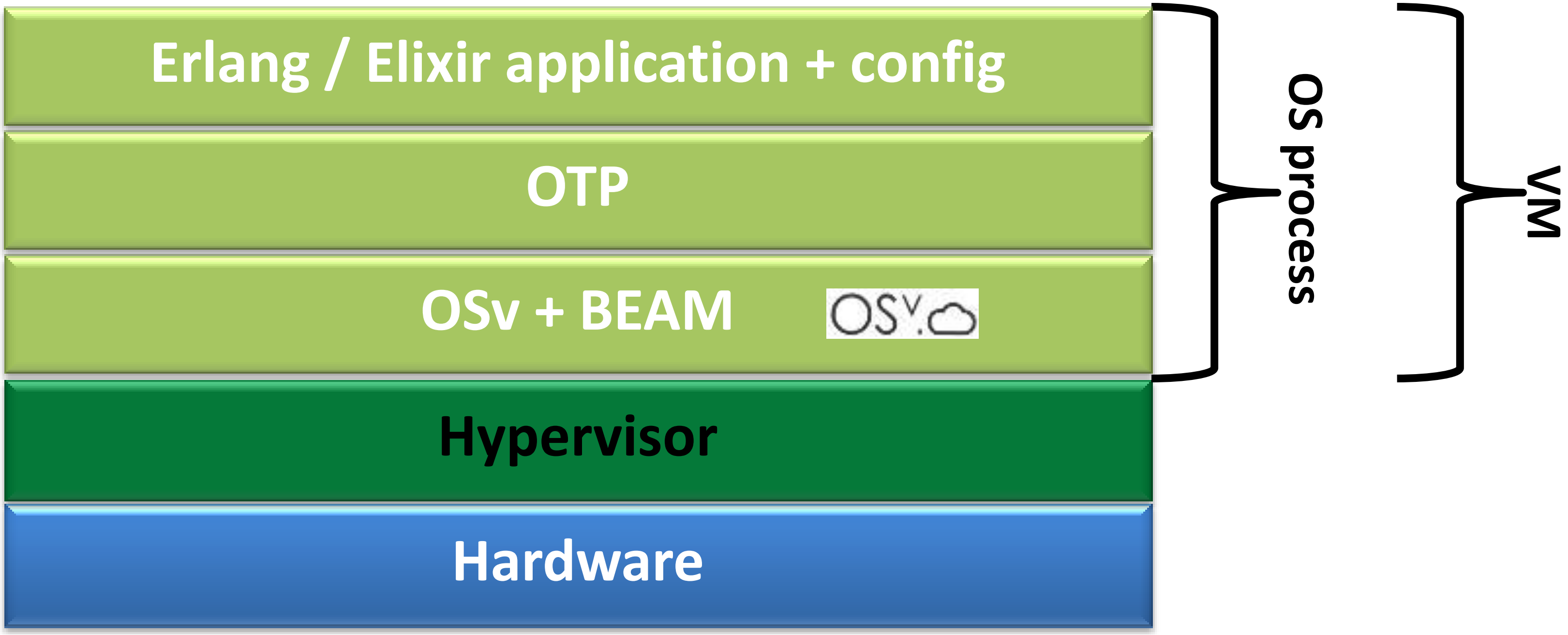
Traditional OS

Hardware

OS process*

Container

Erlang on OSv



Erlang / Elixir application + config

OTP

OSv + BEAM

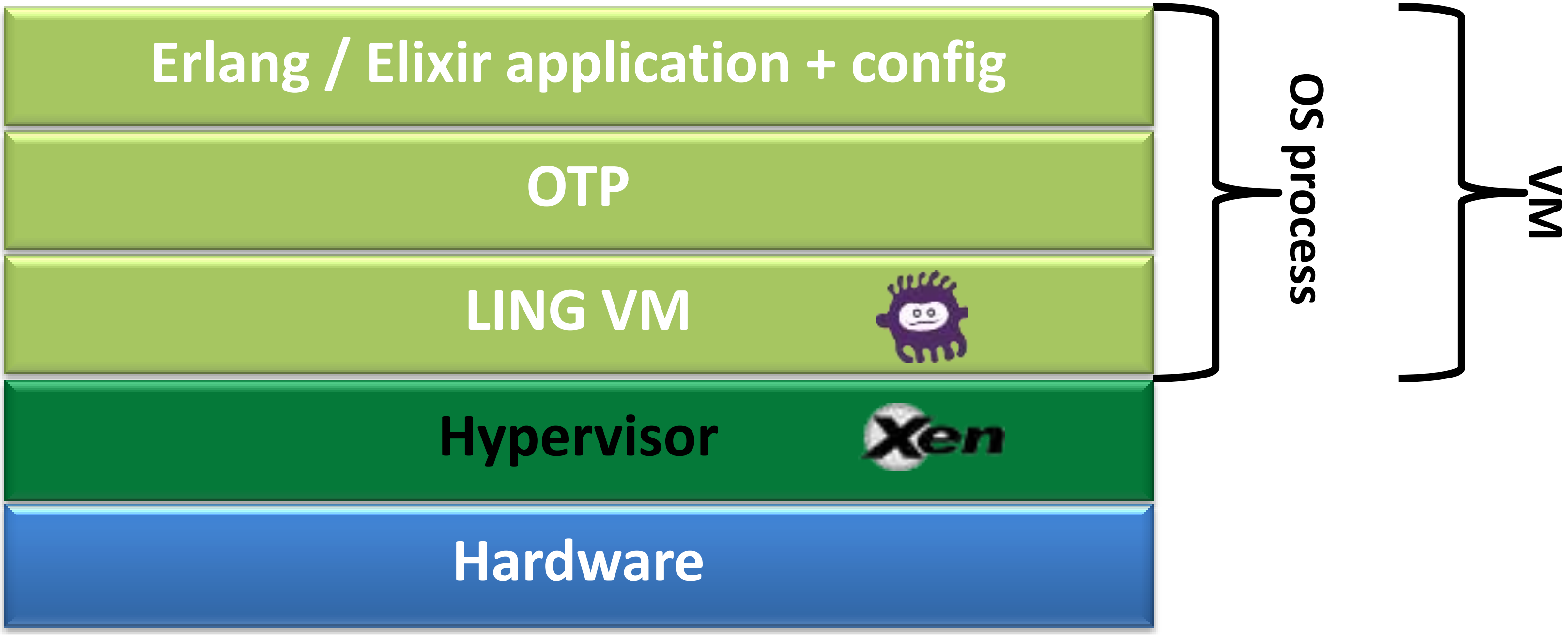
Hypervisor

Hardware

OS process

VM

LING VM (Erlang-on-Xen)



Erlang / Elixir application + config

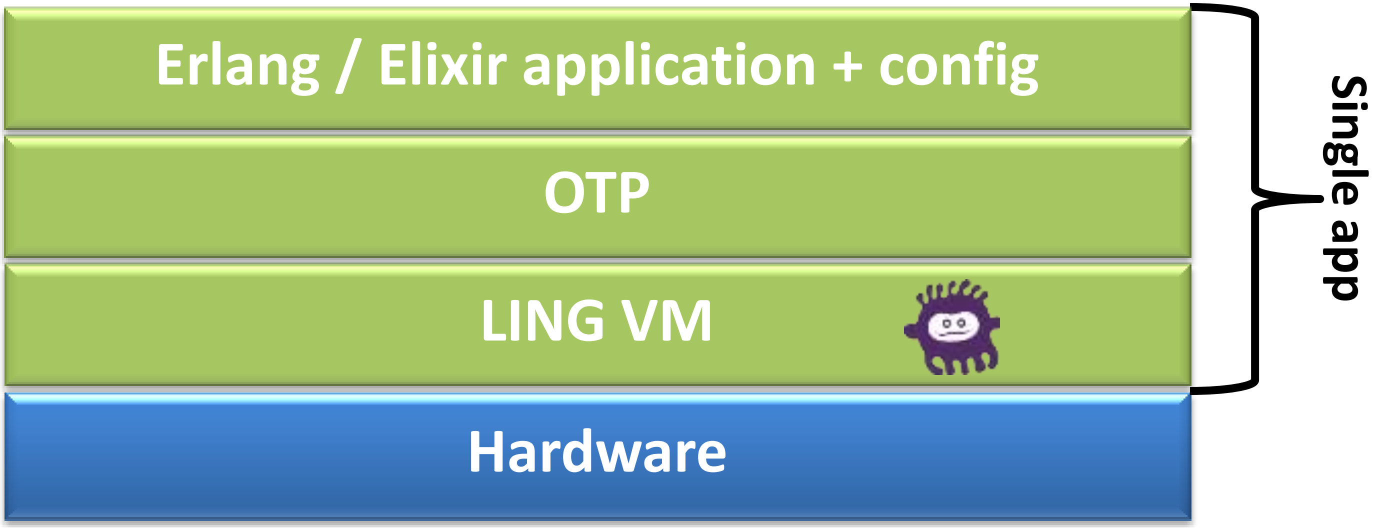
OTP

LING VM

Hypervisor

Hardware

LING VM on Bare Metal


Erlang / Elixir application + config

OTP

LING VM

Hardware

Single app

Erjang – Erlang for JVM

OSv has built-in JVM support, so Erjang runs w/o porting on OSv

on

OSv

```
$ git clone https://github.com/cloudius-systems/osv-apps
$ cd osv-apps/erjang/
$ cat Capstanfile
base:
    cloudius/osv-openjdk
cmdline: >
    /java.so -jar erjang.jar
build:
    ./GET
files:
    /erjang.jar: erjang.jar
```

Build OSv+JVM+Erjang image

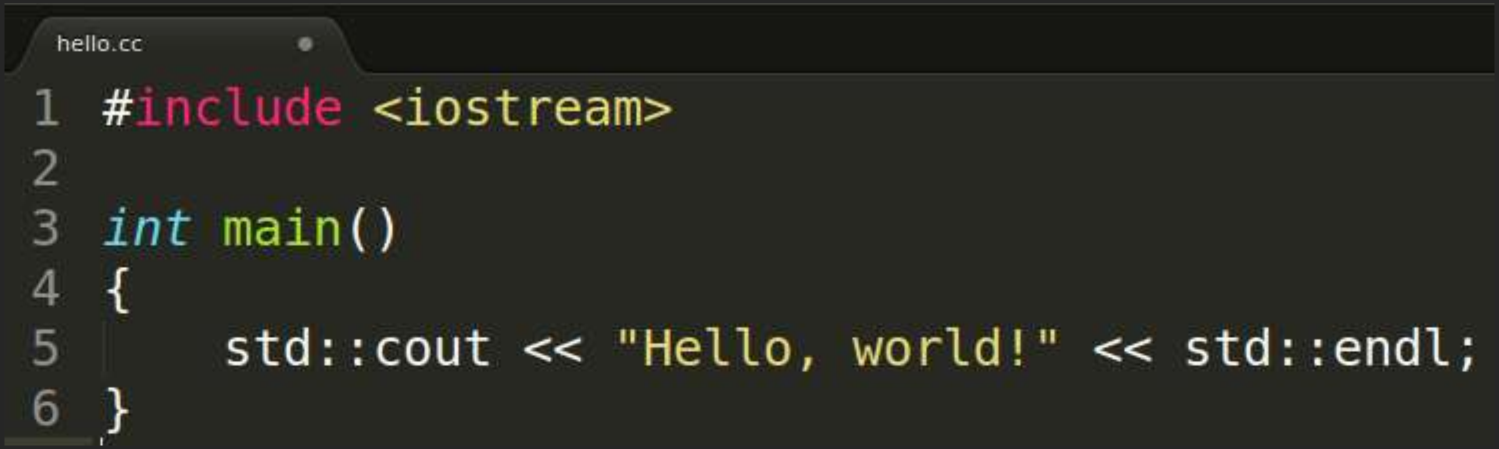
```
$ cd osv-apps/erjang
$ capstan build
Building erjang...
Downloading cloudius/osv-openjdk/index.yaml...
169 B / 169 B [==================] 100.00 %
Downloading cloudius/osv-openjdk/osv-openjdk.qemu.gz...
74.26 MB / 74.26 MB [============] 100.00 %
Uploading files...
1 / 1 [==========================] 100.00 %
```

```
$ capstan run
Created instance: erjang
OSv v0.20
eth0: 192.168.122.15
** Erjang R16B01 **  [root:/~resource]
[erts:5.10.2] [smp S:1 A:10]
[java:1.7.0_55] [unicode]
WARNING: fcntl(F_SETLK) stubbed
Eshell V5.10.2  (abort with ^G)
1> lists:reverse("Hello, World!").
"!dlroW ,olleH“
2> q().
ok
```

Porting a C/C++ application to OSv

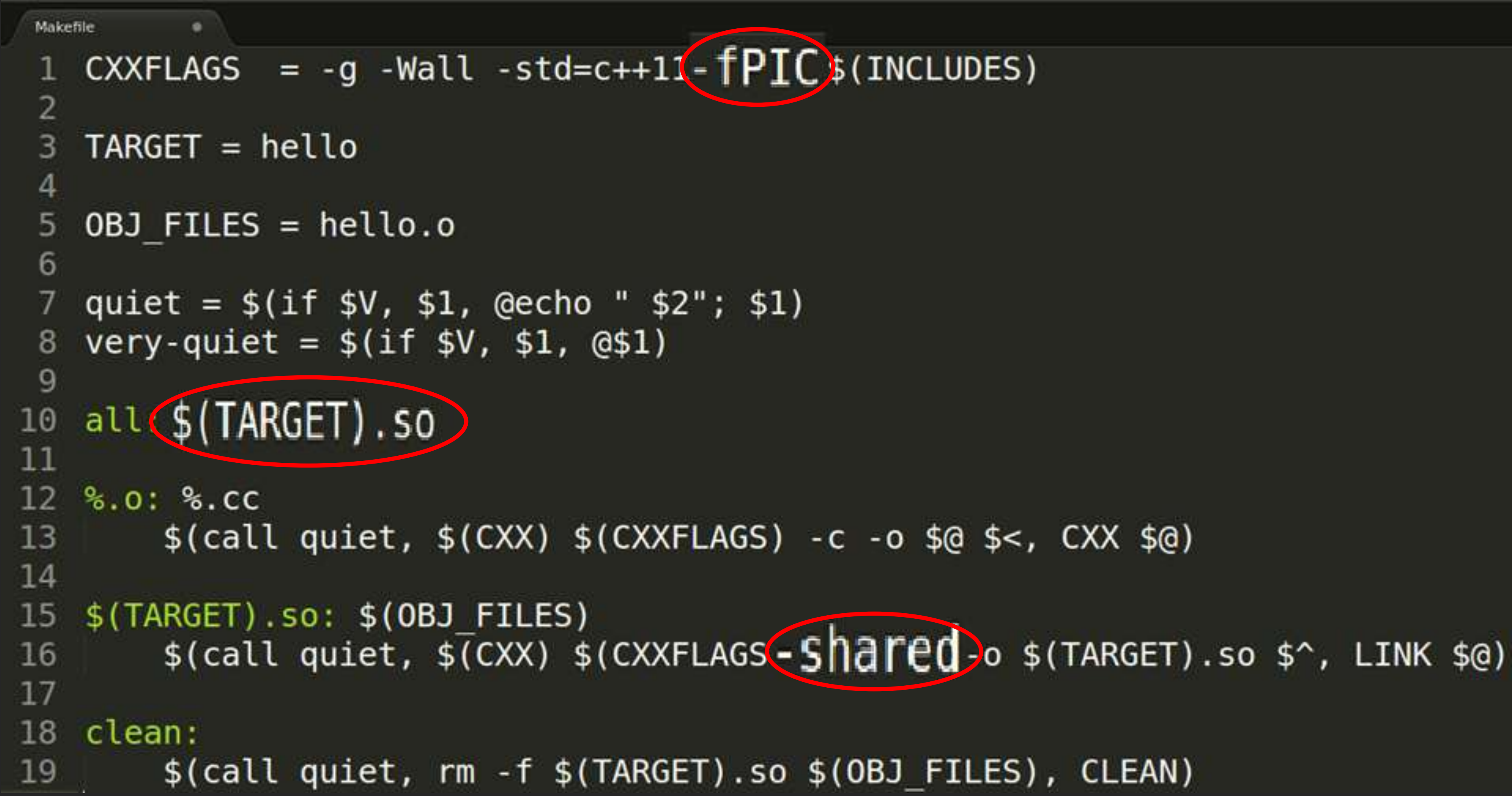
- A single-process application
  - may not ***fork()*** or ***exec()***
- ***Position Independent Code***
  - recompile with ***–fPIC*** flag
- Need to rebuild as a shared object (***.so***)
  - link with ***–shared*** flag
- or as ***Position Independent Exec.*** (***– fpie***)
  - can run the same executable in Linux and OSv

Build OSv image with native code



```
#include <iostream>

int main()
{
    std::cout << "Hello, world!" << std::endl;
}
```

Makefile



```
CXXFLAGS  = -g -Wall -std=c++11 -fPIC $(INCLUDES)

TARGET = hello

OBJ_FILES = hello.o

quiet = $(if $V, $1, @echo " $2"; $1)
very-quiet = $(if $V, $1, @$1)

all: $(TARGET).so

%.o: %.cc
	$(call quiet, $(CXX) $(CXXFLAGS) -c -o $@ $<, CXX $@)

$(TARGET).so: $(OBJ_FILES)
	$(call quiet, $(CXX) $(CXXFLAGS) -shared -o $(TARGET).so $^, LINK $@)

clean:
	$(call quiet, rm -f $(TARGET).so $(OBJ_FILES), CLEAN)
```

```
$ git clone https://github.com/cloudius-systems/capstan-example
$ cd capstan-example
$ cat Capstanfile
base:
    cloudius/osv-base
cmdline:
    /tools/hello.so
build:
    make
files:
    /tools/hello.so: hello.so
```

Build OSv + Hello World image

```
$ capstan build
Building capstan-example...
Downloading cloudius/osv-base/index.yaml...
154 B / 154 B [==================] 100.00 %
Downloading cloudius/osv-base/osv-
base.qemu.gz...
20.13 MB / 20.13 MB [============] 100.00 %
Uploading files...
1 / 1 [==========================] 100.00 %
```

Run OSv + Hello World image

```
$ capstan run
Created instance: capstan-example
OSv v0.20
eth0: 192.168.122.15
Hello, world!

$
```

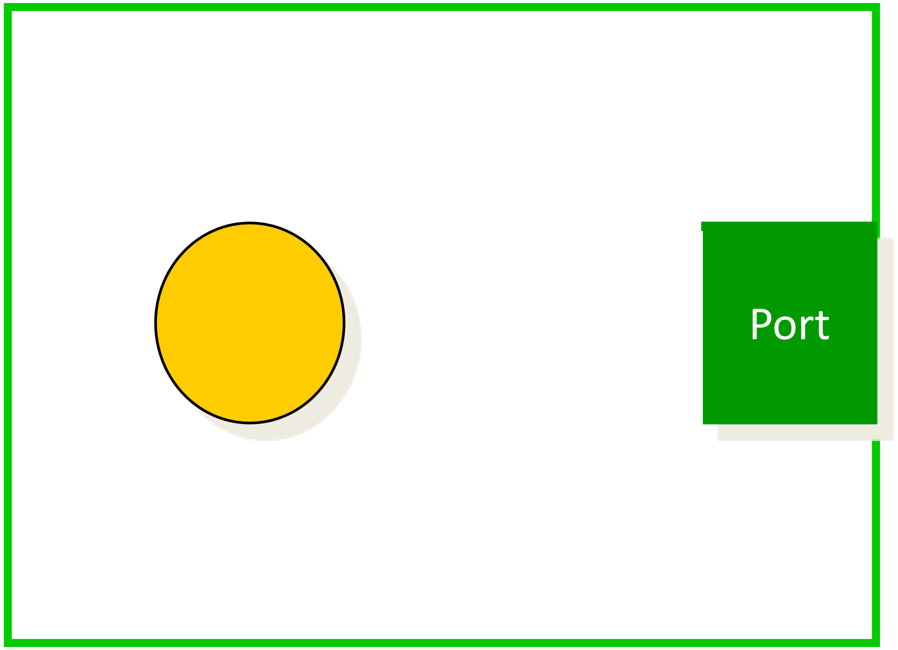
PORTING ERLANG/OTP TO OSV

No fork() or exec() in OSv

BUT ERLANG ♥ FORKS

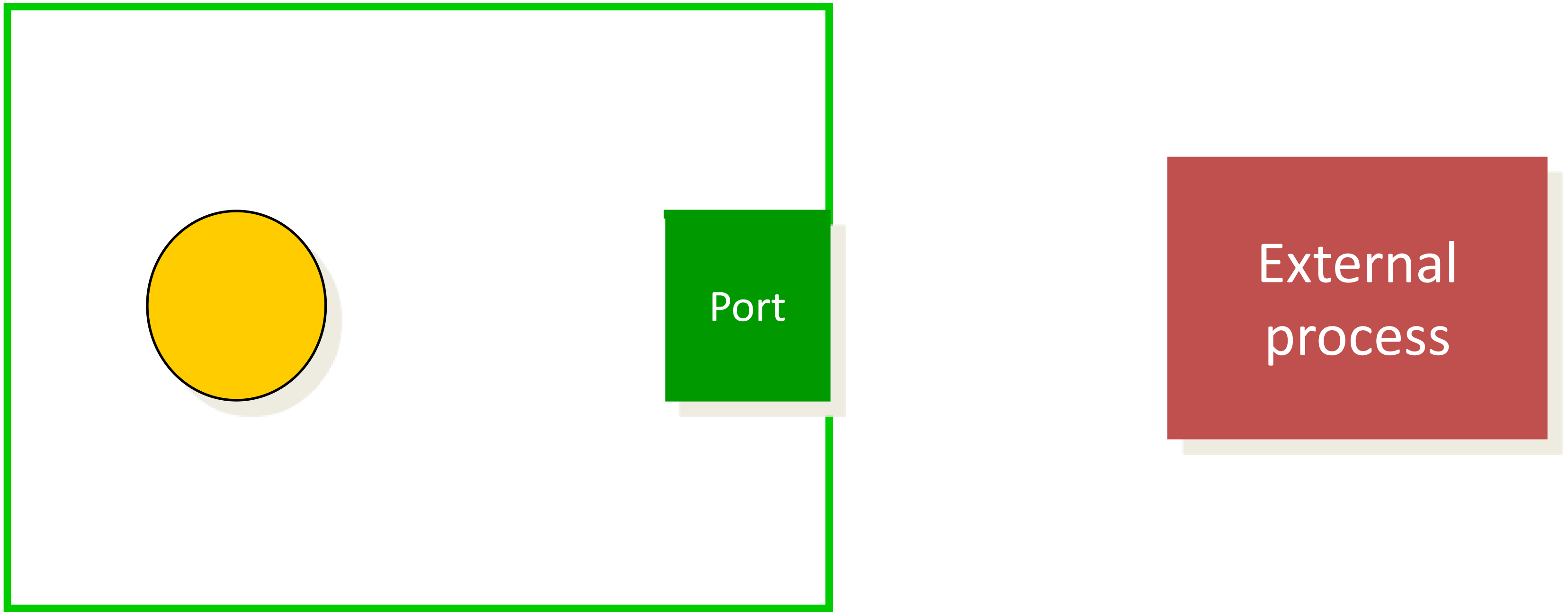
ERLANG ♥ PORTS
Interfaces to the outside world

How Erlang Port Works



Source: Ulf Wiger, Erlang Open Telecom Platform

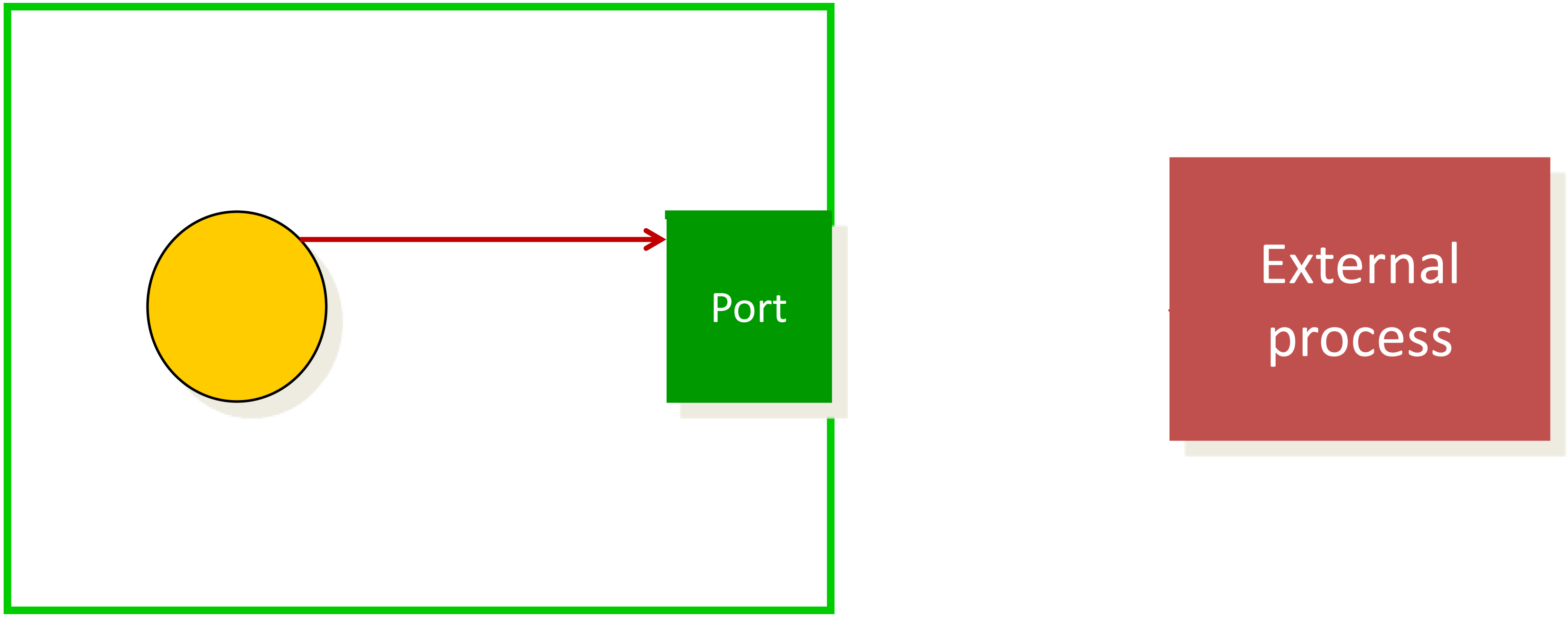
fork/exec the Port executable



- Erlang Port is a middleman/proxy
- allow External Process to pretend to be another Erlang process
- allow communication using regular Erlang message passing

Source: Ulf Wiger, Erlang Open Telecom Platform

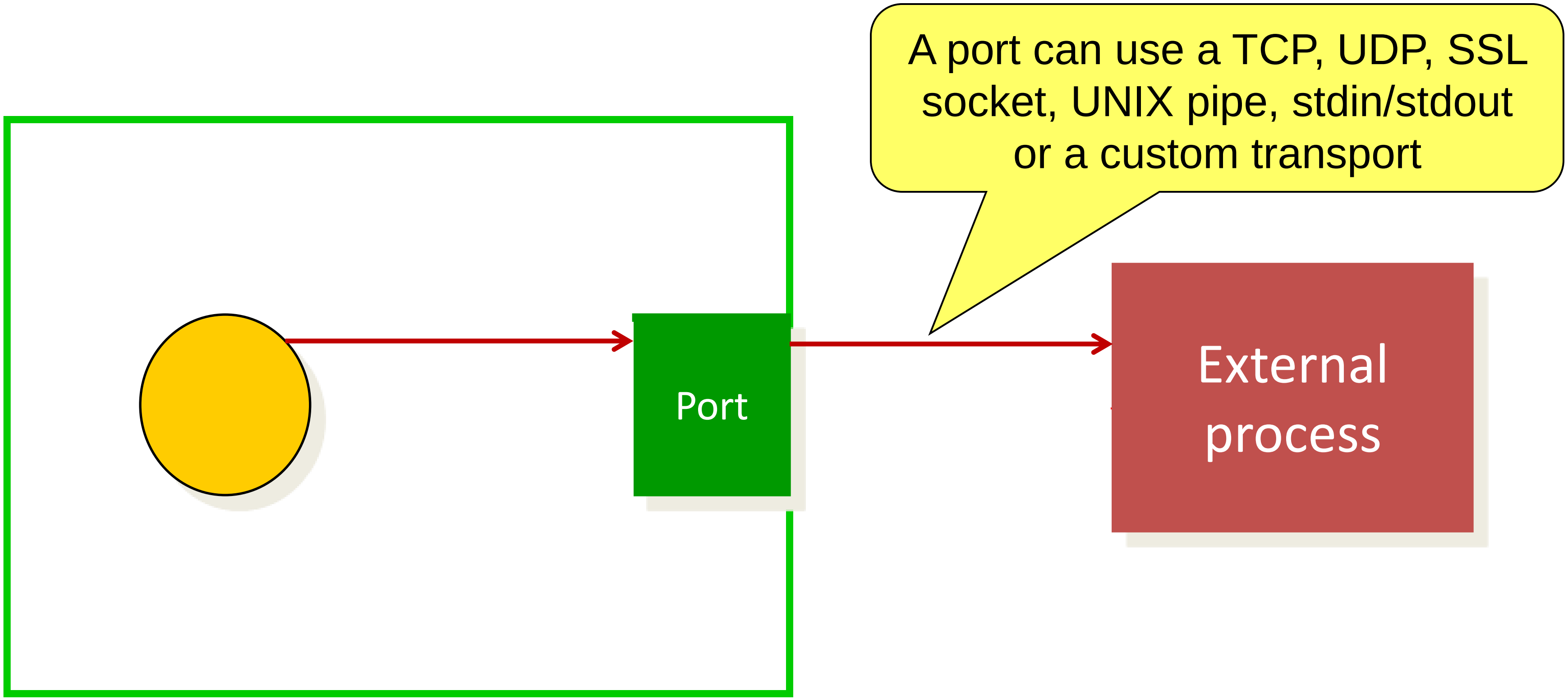
Erlang process sends msg to Port



```
Port ! {self(), {command, [1,2,3]}}
```

Source: Ulf Wiger, Erlang Open Telecom Platform

Port send msg to External Process

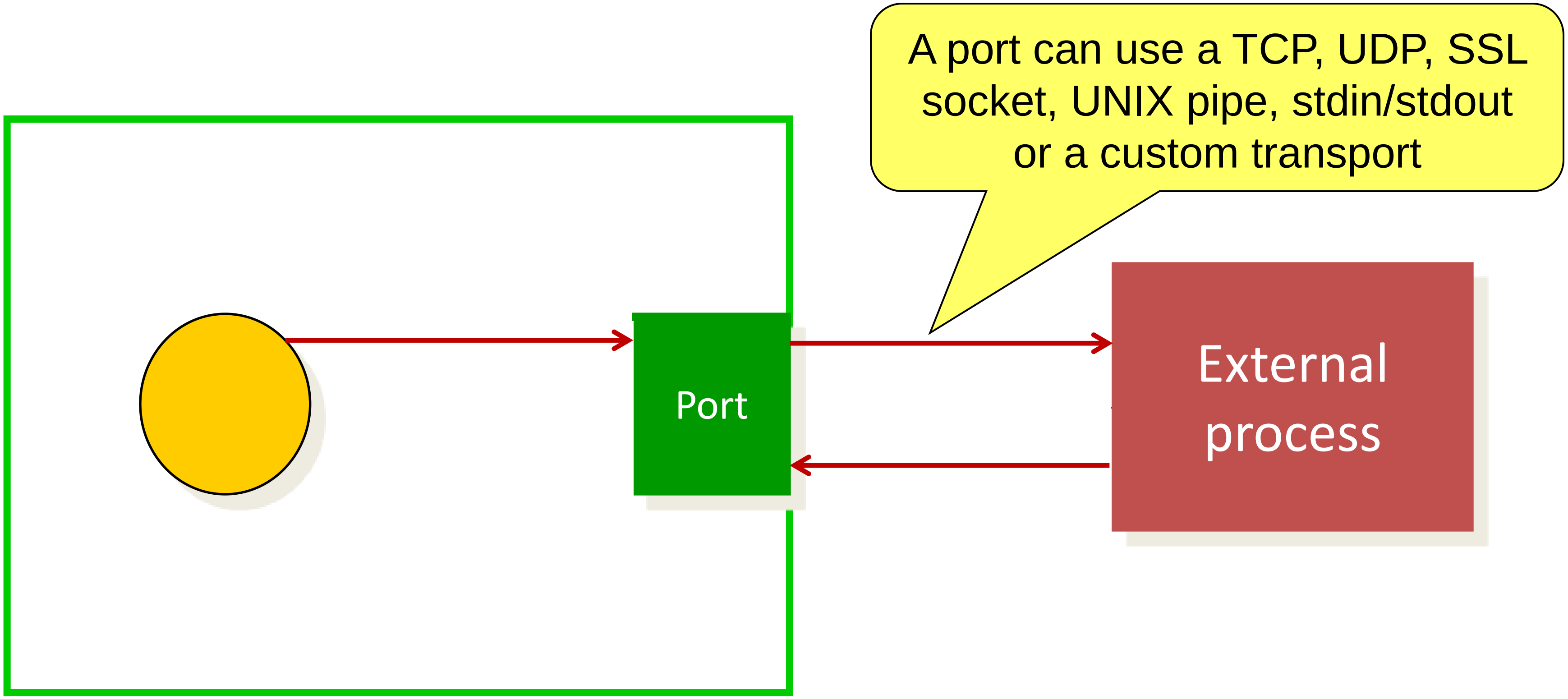


A port can use a TCP, UDP, SSL socket, UNIX pipe, stdin/stdout or a custom transport

```
Port ! {self(), {command, [1,2,3]}}
```

Source: Ulf Wiger, Erlang Open Telecom Platform

Port rcv msg from External Process

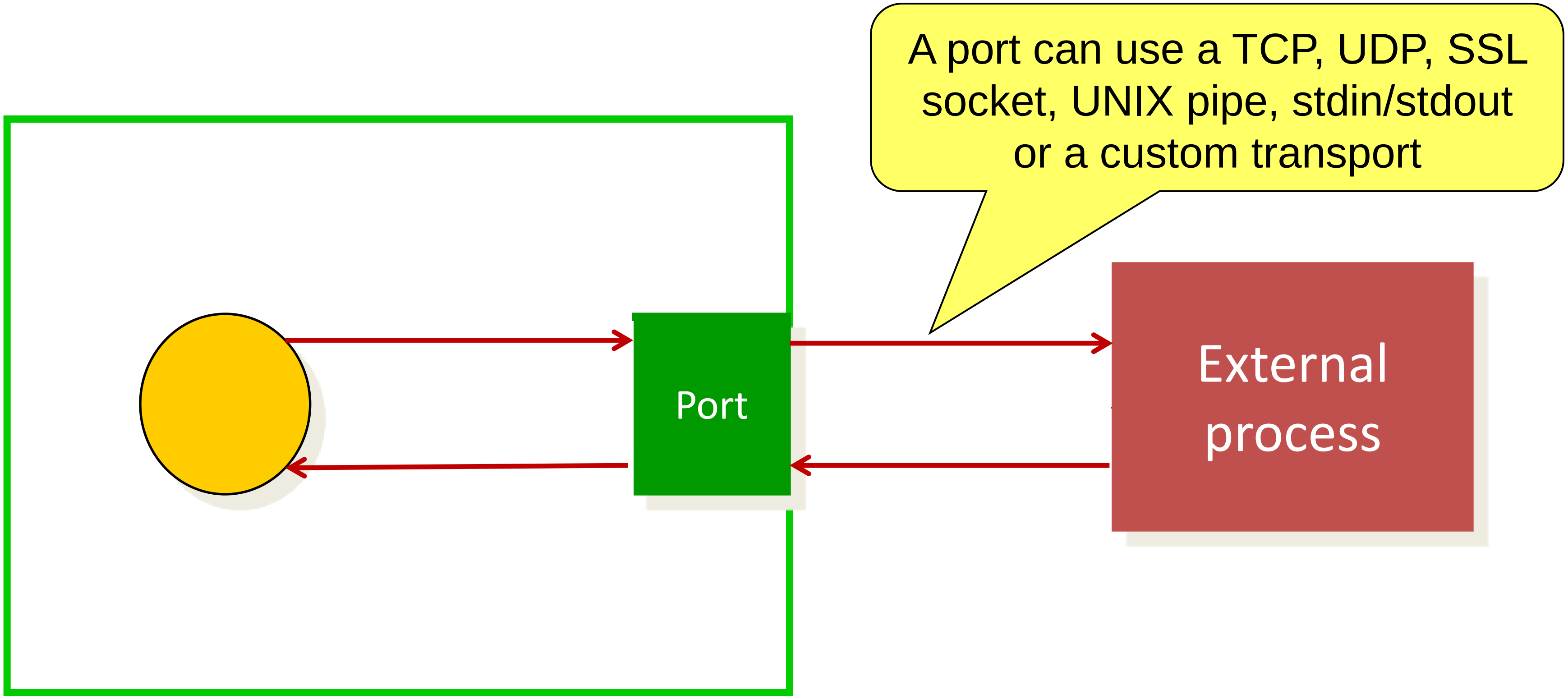


A port can use a TCP, UDP, SSL socket, UNIX pipe, stdin/stdout or a custom transport

```
receive
    {Port, {data, Info}} -> ...
end
```

Source: Ulf Wiger, Erlang Open Telecom Platform

Port send msg to Erlang Process



A port can use a TCP, UDP, SSL socket, UNIX pipe, stdin/stdout or a custom transport

```
receive
    {Port, {data, Info}} -> ...
end
```

Source: Ulf Wiger, Erlang Open Telecom Platform

Communicating with External world

	In-proc	Out-of-proc
Custom Protocol	Linked-in Drivers (.so)	~~Ports (executable)~~
Generic Protocol	NIF (.so)	C-Node

How to adapt existing Ports to OSv?

- the solution was to rewrite ***ports using OS processes*** as ***linked-in drivers using POSIX threads***
- With some exceptions

Community Port of Erlang/OTP to OSv

- I “ported” Erjang a year ago
- I tried to port official Erlang/OTP – but quickly lost myself in the Erlang build system
- Also reliance on lots of ports and shell scripts made me less optimistic
- Cloudius is focused on POSIX & JVM, Erlang is not a priority
- OSv mailing list discussed porting Erlang, but there was no progress

Community Port of Erlang/OTP to OSv

- People who helped to port Erlang to OSv:
  - ***Yao Zheng*** (https://github.com/bhuztez)
  - ***Bas Wegh***
  - ***Zika L'Etrange***

- ***Yao Zheng*** also ported following projects:
  - Elixir
  - LFE
  - yaws

The most important executables

- **erl** – wrapper to ***erlexec***
- ***epmd*** – Erlang Port Mapping Daemon
- ***heart*** – watchdog for BEAM (Erlang VM)
- ***inet_gethost*** – native DNS resolver
- ***osmon*** – mem_sup, disk_sup, cpu_sup
- ***odbserver*** – ODBC connectivity

Not everything ported yet.

erl

- ***erlang.so*** – OSv shared object executable wrapper to ***erlexec***
- pass ***vm.args*** and other CLI parameters to ***erlexec***

```
base:
  cloudius/osv-base
cmdline: >
   /usr/lib64/erlang.so -env HOME / \
   /etc/erlang/vm.args \
   /etc/default/erlang/vm.args
build:
  make
files:
  /usr/lib64/erlang/: ROOTFS/usr/lib64/erlang/
  /usr/lib64/erlang.so: erlang.so
  /etc/default/erlang/: default/
  /etc/erlang/vm.args: default/vm.args
```

```
$ git clone git@github.com:cloudius-systems/osv-apps.git
$ cd osv-apps/erlang
$ capstan run
Created instance: erlang
OSv v0.20
eth0: 192.168.122.15
sched_getaffinity() stubbed
Eshell V6.2  (abort with ^G)
1> lists:reverse("Hello, World!").
"!dlroW ,olleH“
2>
```

epmd – Erlang Port Mapper Daemon

- Option #1: **use pure Erlang implementation of epmd**:
  - https://github.com/bhuztez/noepmd
  - https://github.com/lemenkov/erlpmd
- Option #2: **run unmodified epmd as a POSIX thread, instead of a separate OS process**
- Option #2 was selected
- Q: How epmd implemented in Erlang on RTOS like VxWorks or OSE?

inet_gethost

```
1> httpc:request("http://google.com").

=ERROR REPORT==== 6-May-2015::13:43:58 ===
Error in process <0.79.0> with exit value:
{unknown,[{erlang,open_port,[
  {spawn,"inet_gethost 4"}, ...
```

inet_gethost

- Currently DNS resolving doesn’t work yet
- Force to use pure-Erlang inet DNS resolver
- It’s a default in Erlang on VxWorks & ENEA OSE RTOS

erl -sname mynode -kernel inetrc "'./erl_inetrc'“

{edns, 0}

http://www.erlang.org/doc/apps/erts/inet_cfg.html

osmon

- ***osmon*** ports where modified from standalone OS processes to linked-in drivers on POSIX thread
  - ***disk_sup***
  - ***mem_sup***
  - ***cpu_sup*** is disabled since OSv lacks required API

OTP apps + BEAM + Unikernel OS

- Erlang releases already may include Erlang VM
- Adding a Unikernel OS to releases will make them truly self-contained!

Development process

- Currently Erlang OSv port is developed as a set of patch files in Osv apps repo:
  - **https://github.com/cloudius-systems/osv-apps**
- Better way would be, to make it part of Erlang/OTP official repository (just like other OSes):
  - **https://github.com/erlang/otp**

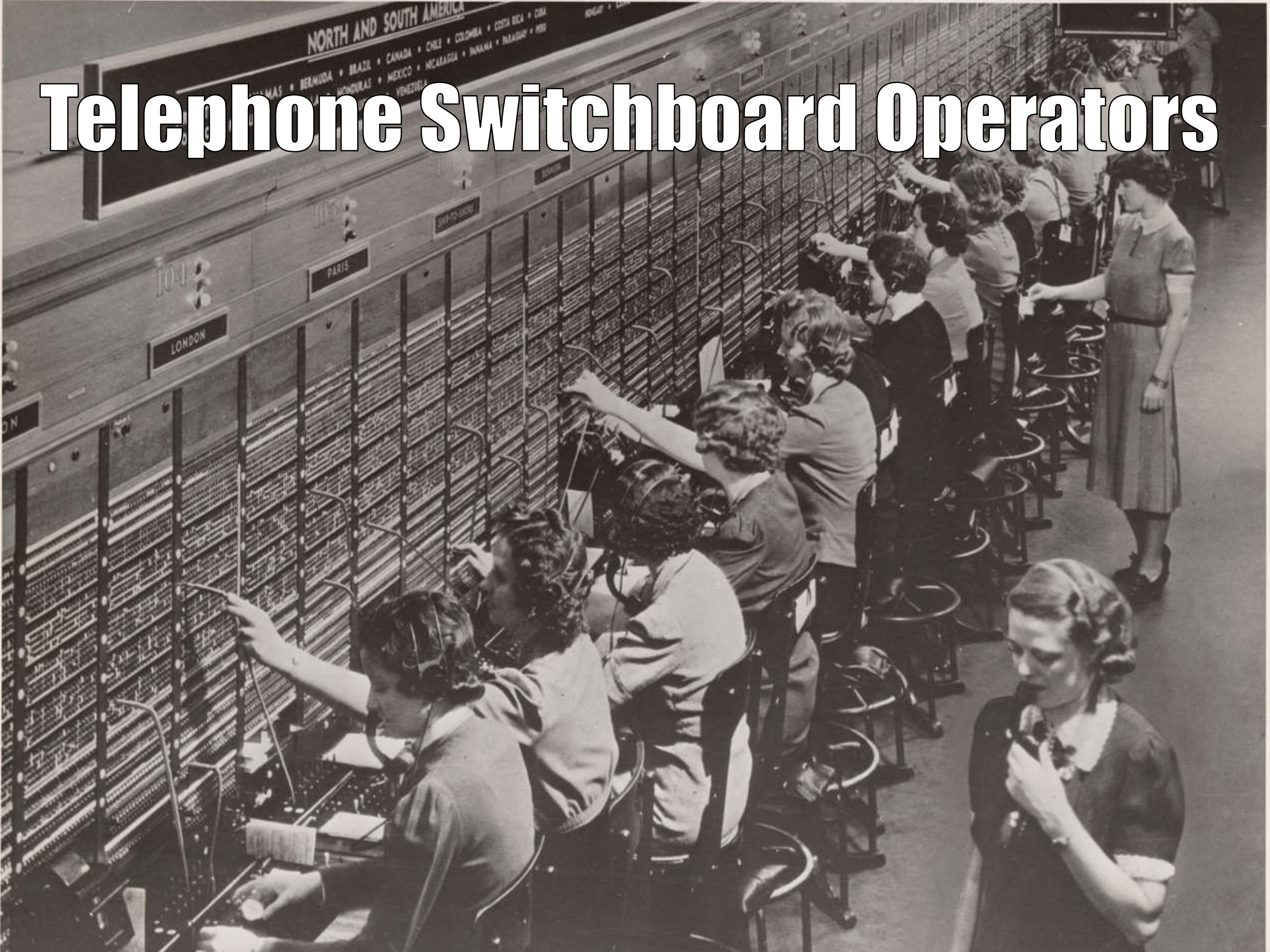
Why Use Unikernels?

HIGH PERFORMANCE NETWORKING & I/O

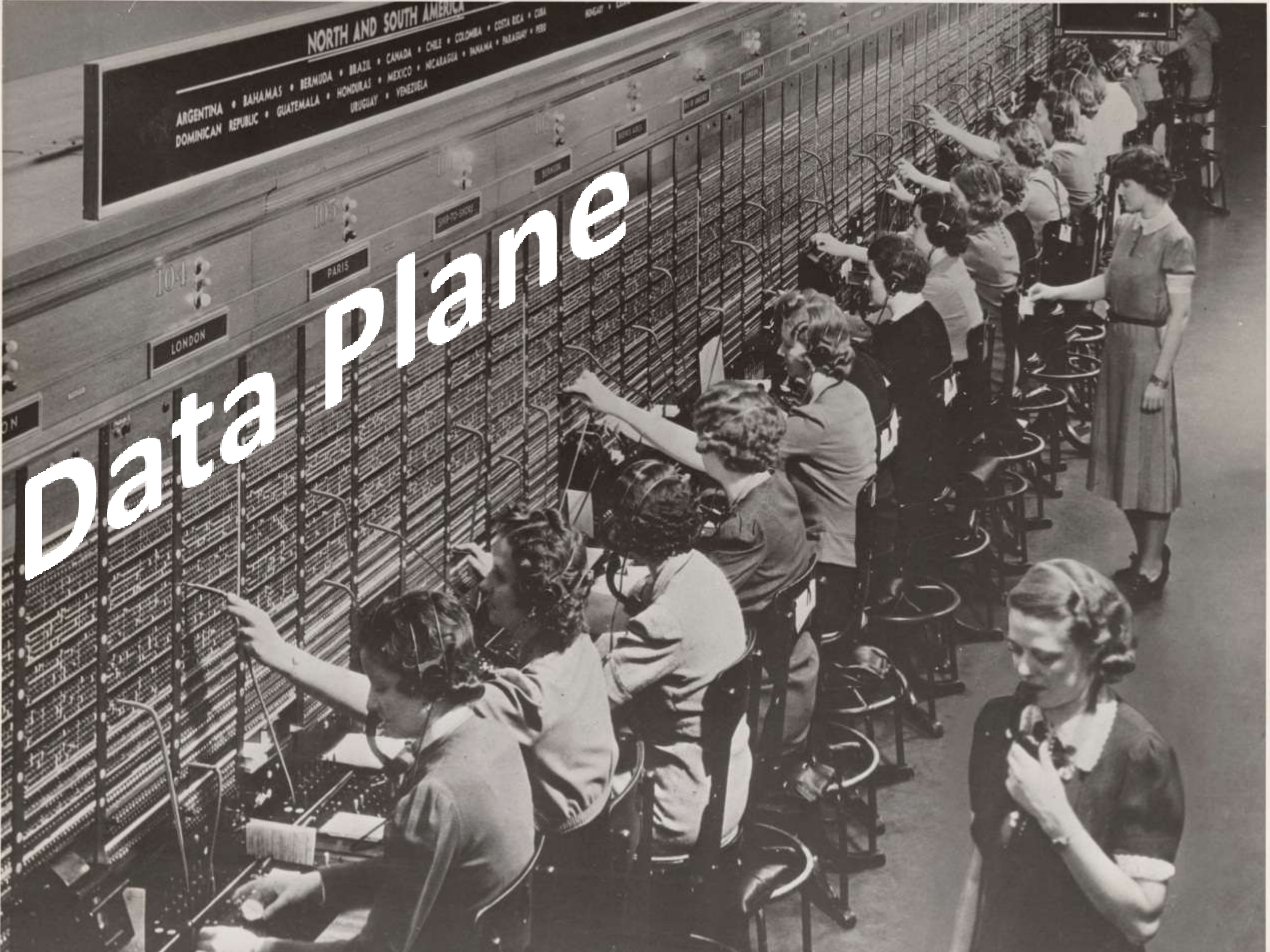
Long time before “Erlang The Movie”. Telephone Switches worked like …

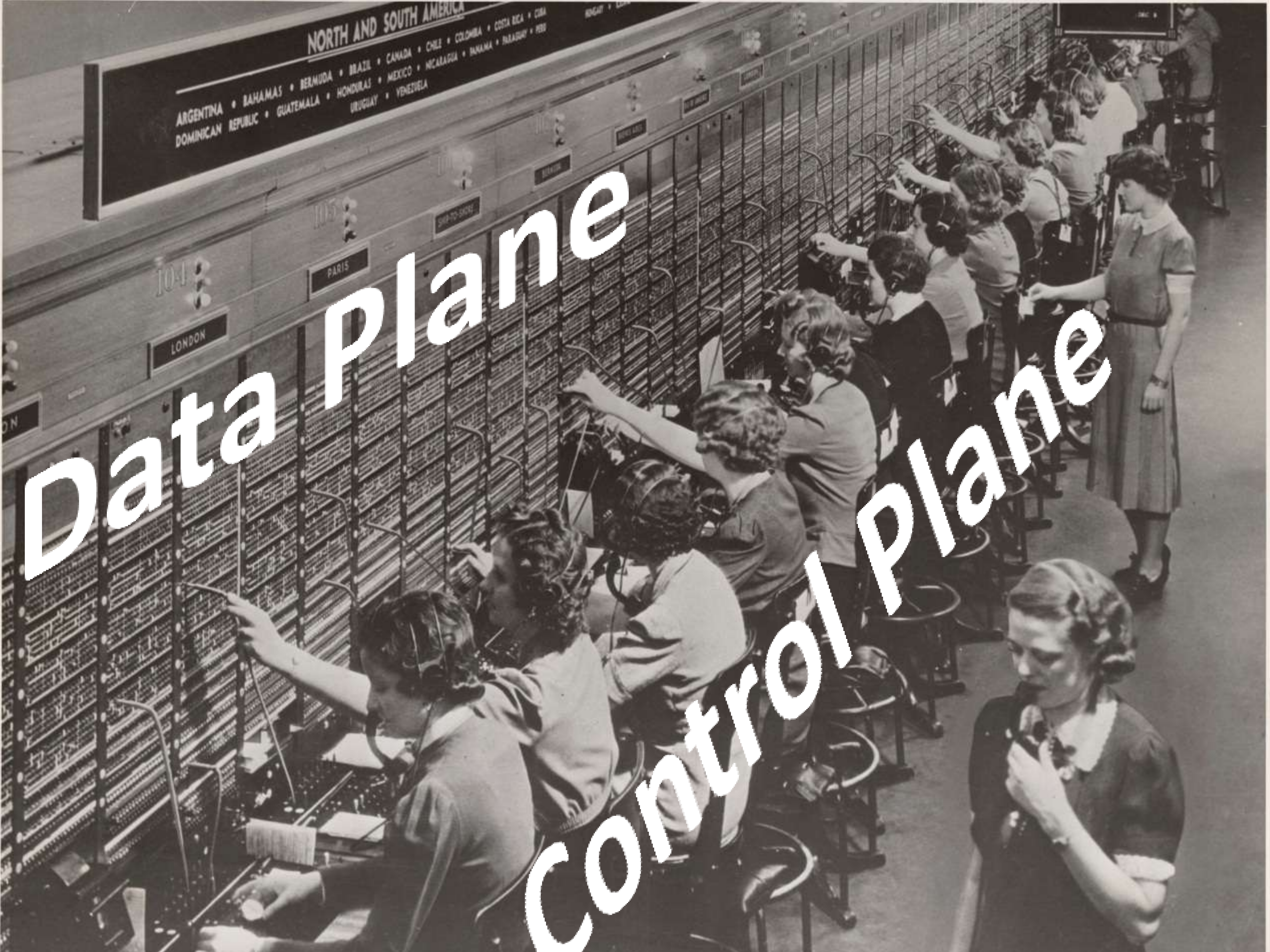
Source: Erlang The Movie

Telephone Switchboard Operators


NORTH AND SOUTH AMERICA
BERMUDA • BRAZIL • CANADA • CHILE • COLOMBIA • COSTA RICA • CUBA
HONDURAS • MEXICO • NICARAGUA • PANAMA • PARAGUAY • PERU
VENEZUELA
105
SHIP-TO-SHORE
104
PARIS
LONDON

Data Plane

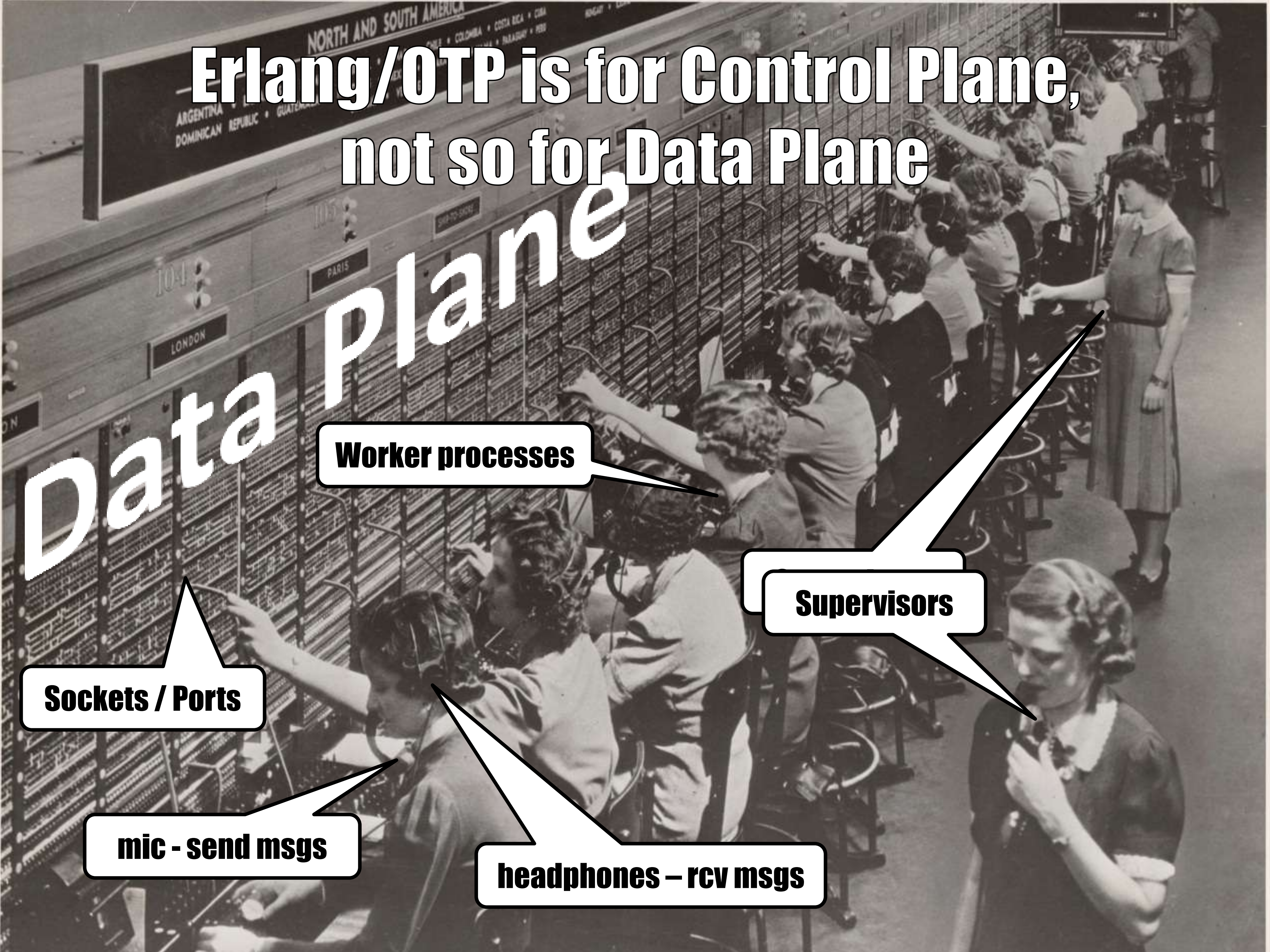

NORTH AND SOUTH AMERICA
ARGENTINA • BAHAMAS • BERMUDA • BRAZIL • CANADA • CHILE • COLOMBIA • COSTA RICA • CUBA
DOMINICAN REPUBLIC • GUATEMALA • HONDURAS • MEXICO • NICARAGUA • PANAMA • PARAGUAY • PERU
URUGUAY • VENEZUELA
104
LONDON
105
PARIS
SHIP-TO-SHORE


NORTH AND SOUTH AMERICA
ARGENTINA • BAHAMAS • BERMUDA • BRAZIL • CANADA • CHILE • COLOMBIA • COSTA RICA • CUBA
DOMINICAN REPUBLIC • GUATEMALA • HONDURAS • MEXICO • NICARAGUA • PANAMA • PARAGUAY • PERU
URUGUAY • VENEZUELA
104
105
LONDON
PARIS
SHIP-TO-SHORE

Data Plane

Control Plane

Erlang/OTP is for Control Plane, not so for Data Plane



Data Plane was handled by specialized HW or RTOS

Source: Erlang The Movie

Erlang for both Control Plane + Data Plane

- In traditional OSes like Linux, IO & TCP/IP implemented via system calls, which has kernel overhead
- Many unikernels have user-space TCP/IP stack
- OSv implements Van Jacobsen’s Net Channels optimization
- Optimized for NFV & SDN applications
- It’s already possible with LING VM
- And with non-POSIX APIs in OSv

Comparing LING VM vs Erlang on OSv

LING VM (ERLANG-ON-XEN) & ERLANG/OTP ON OSV

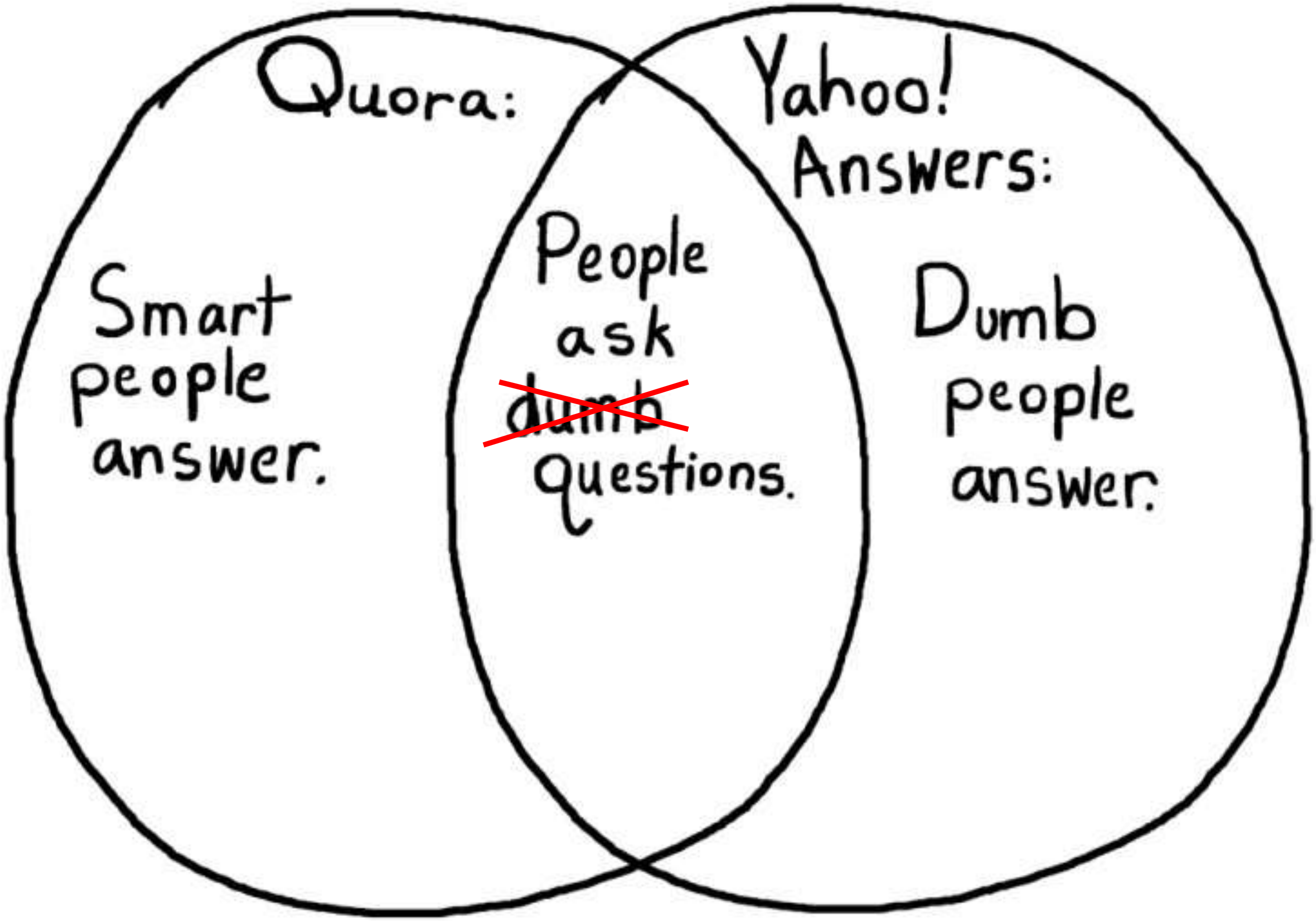
	LING VM	Erlang/OTP on OSv
Erlang VM	Custom VM (latest update from OTP – Dec 2014)	Open-source Ericsson’s Erlang/OTP 17.3
State	Production-grade (3.5 years)	Experimental (6 months)
License	Proprietary (Sleepycat)	BSD 3-clause
SMP (threads)	-	+
Ports	-	-
NIFs	-	+ (possible)
HiPE	-	- (possible)
Disable GC	+	-

	LING VM	Erlang/OTP on OSv
Hypervisors	Xen only	KVM, Xen, VMware, VirtualBox
Bare Metal	Bare Metal RPi	-
Cloud support	AWS	AWS, Google Cloud
Machine arch	X86-64,ARM-32,MIPS	X86-64, ARM-64
Min img size	9 MB	21 MB
Boot time	< 100 ms	< 1 sec
“Dockerfiles”	railing	Capstan
Filesystem	goofs (Plan9)	ZFS
Data Plane, NFV/SDN	+	possible via non-POSIX API?

Takeaways

- It seems that after Docker, ***Unikernels*** are going to be ***"The Next Big Thing"*** in the Cloud
- Erlang community now has 2 options to choose from - try both
- Try to port your Erlang app
- Try to port your favorite open-source Erlang app
- But only when it makes sense!
- Download OSv and help porting standard Ericsson's Erlang/OTP to Osv
- **https://github.com/erlang-on-osv**

Thank You! Now Q&A



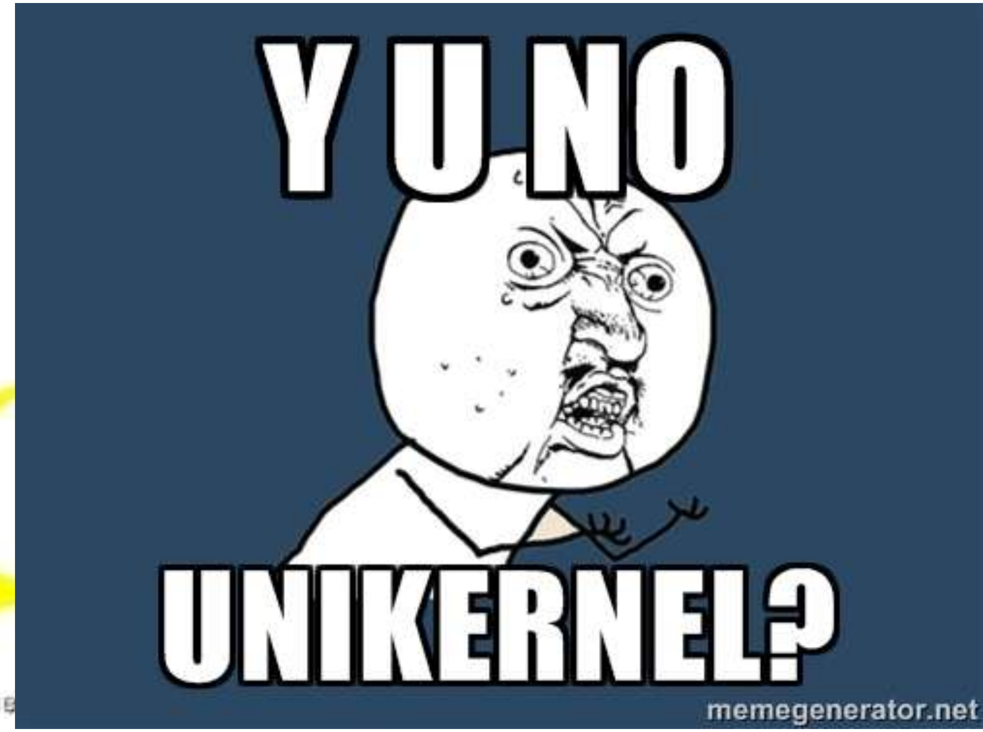
All images are taken from Google Image search and various other places on the Internet

Thanks!

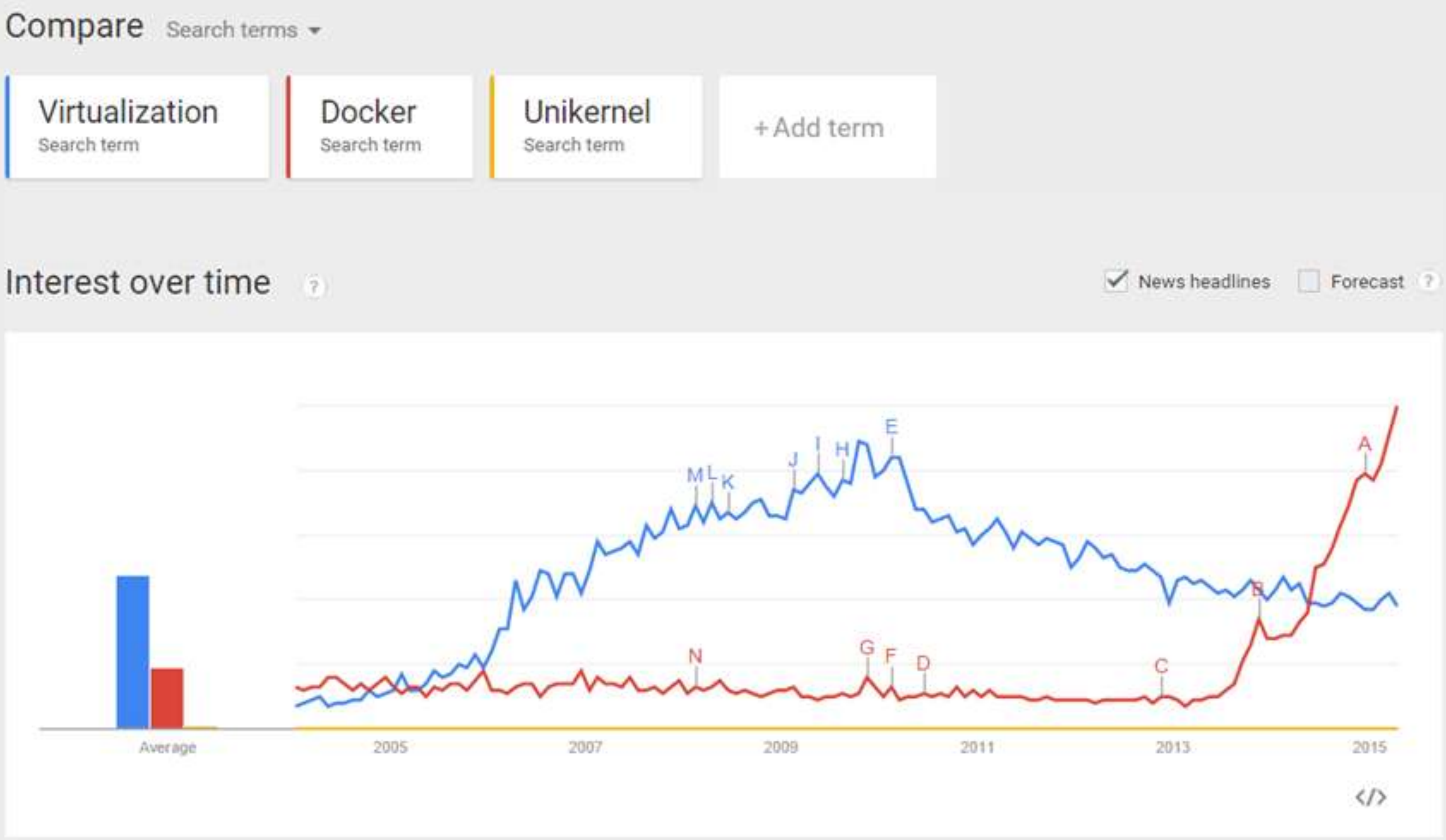
- Thanks to everybody who gave me the feedback on this presentation:
  - Dor Laor
  - Tzach Livyatan
  - Yao Zheng
  - Dave Cottlehuber

BACKUP SLIDES

Mandatory Meme slide ;)



Old-style VMs peaked in 2010
Containers on the rise since 2013



Compare Search terms ▾

Virtualization
Search term

Docker
Search term

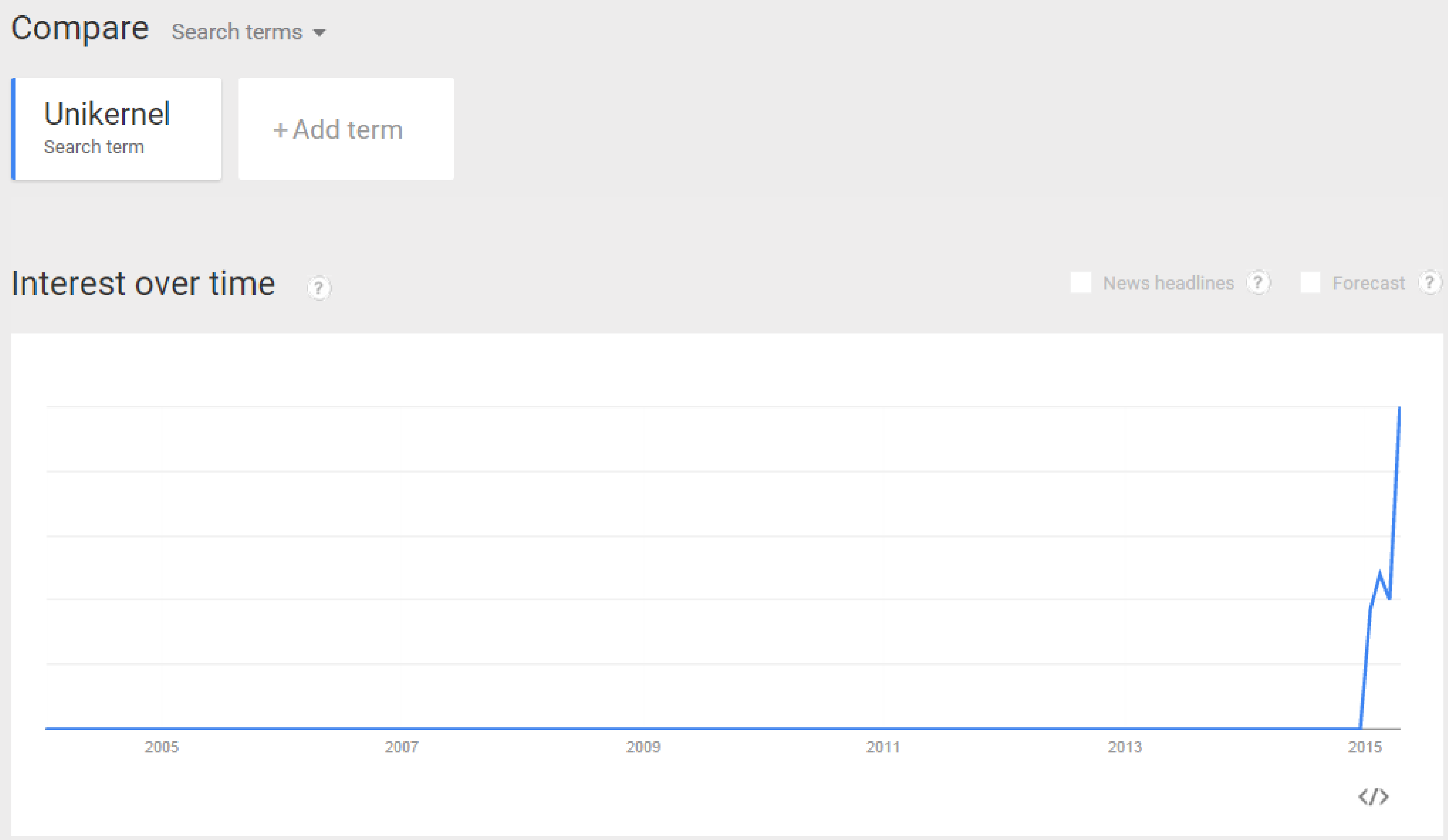
Unikernel
Search term

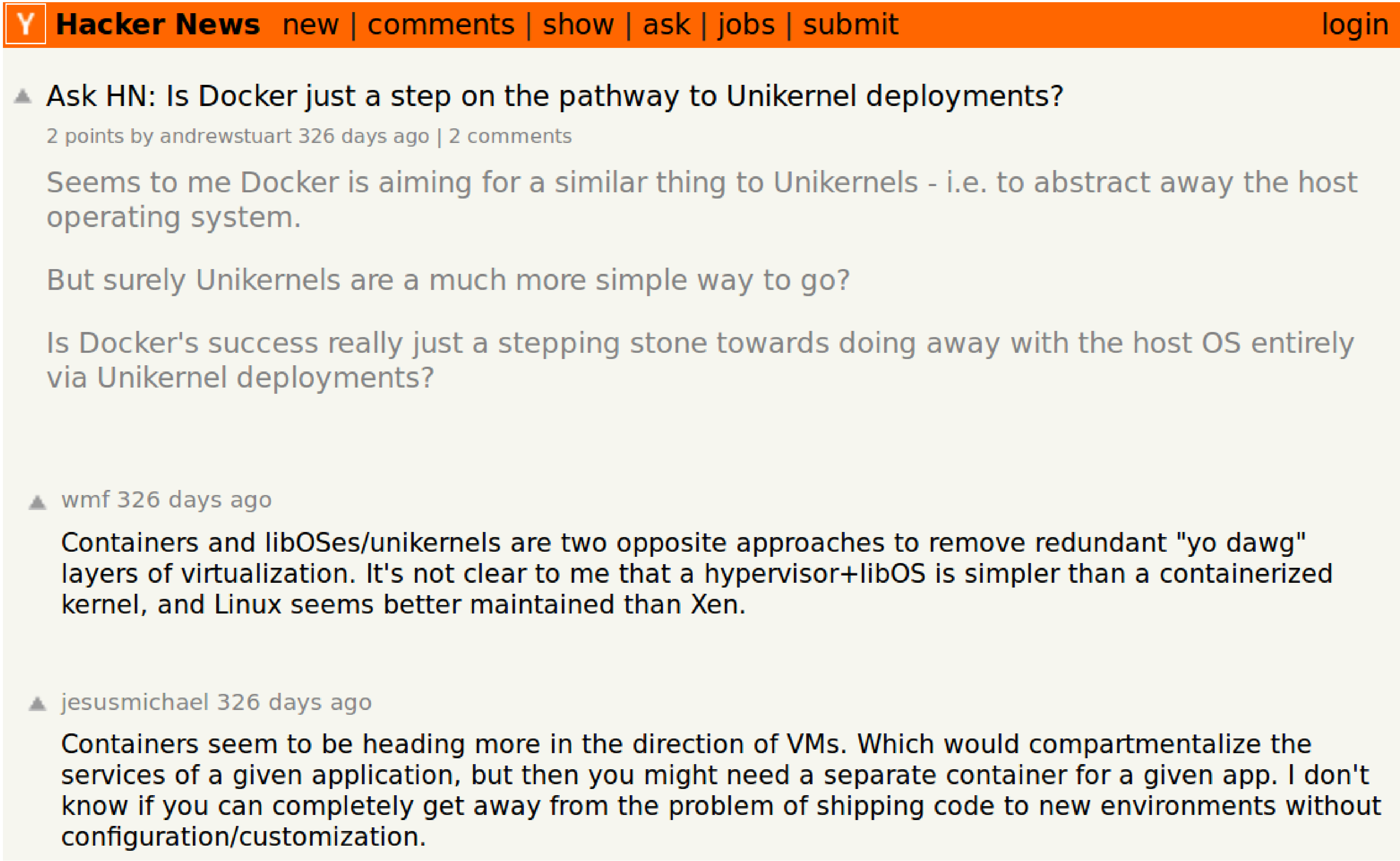
+Add term

Interest over time ?

☑ News headlines □ Forecast ?

Unikernels spiking in 2015





Y **Hacker News** new | comments | show | ask | jobs | submit login

Ask HN: Is Docker just a step on the pathway to Unikernel deployments?

2 points by andrewstuart 326 days ago | 2 comments

Seems to me Docker is aiming for a similar thing to Unikernels - i.e. to abstract away the host operating system.

But surely Unikernels are a much more simple way to go?

Is Docker's success really just a stepping stone towards doing away with the host OS entirely via Unikernel deployments?

wmf 326 days ago

Containers and libOSes/unikernels are two opposite approaches to remove redundant "yo dawg" layers of virtualization. It's not clear to me that a hypervisor+libOS is simpler than a containerized kernel, and Linux seems better maintained than Xen.

jesusmichael 326 days ago

Containers seem to be heading more in the direction of VMs. Which would compartmentalize the services of a given application, but then you might need a separate container for a given app. I don't know if you can completely get away from the problem of shipping code to new environments without configuration/customization.