

Unikernels?

Thomas Gazagnaire

@samoht [GitHub]

@eriangazag [Twitter]

<http://gazagnaire.org/pub/2015.12.loops.pdf>

About me...

- PhD at INRIA in Distributed Systems
- Citrix on Xen/Xenserver
- OCamlPro on Opam and OCaml tooling
- University of Cambridge on MirageOS and Irmin
- recently Unikernel Systems


Unikernels

- based on **library OS**
- contains only what is needed
- single process
- single address space
- runs everywhere


Unikernels

MIRAGE OS
HaLVM
Rumprun
...

Hypervisor
(optional)

KVM
Xen™

Hardware



Traditional OS

- Multiple users
- Multiple process
- Multiple purposes

Configuration

Application code

Language runtime

System libraries

OS Kernel

Hypervisor
(optional)



Hardware



Traditional OS

Traditional OS

The Kernel

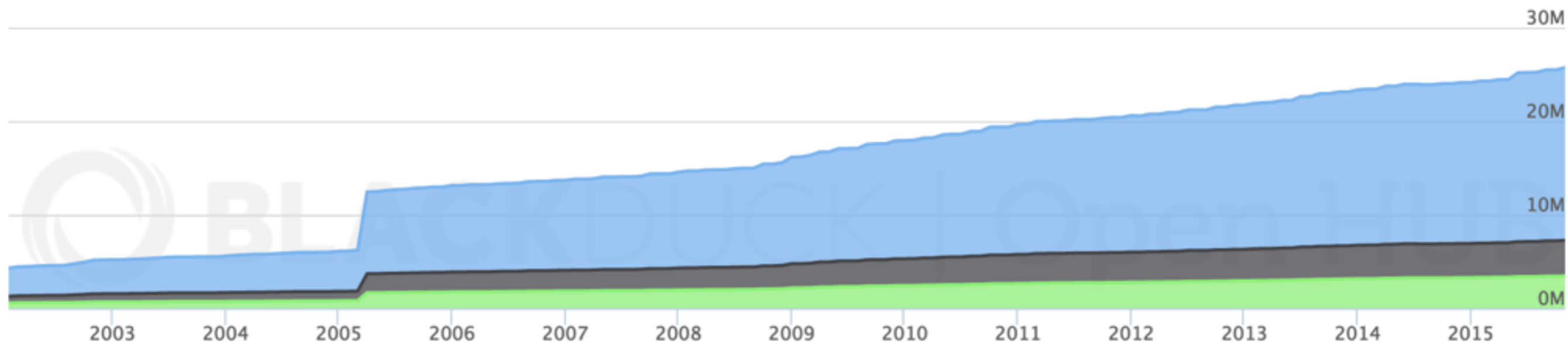
True, linux is monolithic, and I agree that microkernels are nicer... As has been noted (not only by me), the linux kernel is a minuscule part of a complete system: Full sources for linux currently runs to about 200kB compressed. And all of that source is portable, except for this tiny kernel that you can (provably: I did it) re-write totally from scratch in less than a year without having /any/ prior knowledge.

– Linus Torvalds, 1992

Traditional OS

The Kernel

Currently Linux has over 25 million lines of code...



... and Windows has 50 million.

Traditional OS

The System libraries

OS distributions contain a lot of code!

- Debian 5.0: 65 million lines of code
- OSX 10.4: 85 million lines of code

Traditional OS

The System Libraries

Application

openGL

iconv

gtk

libz

libgmp

libtls

libc

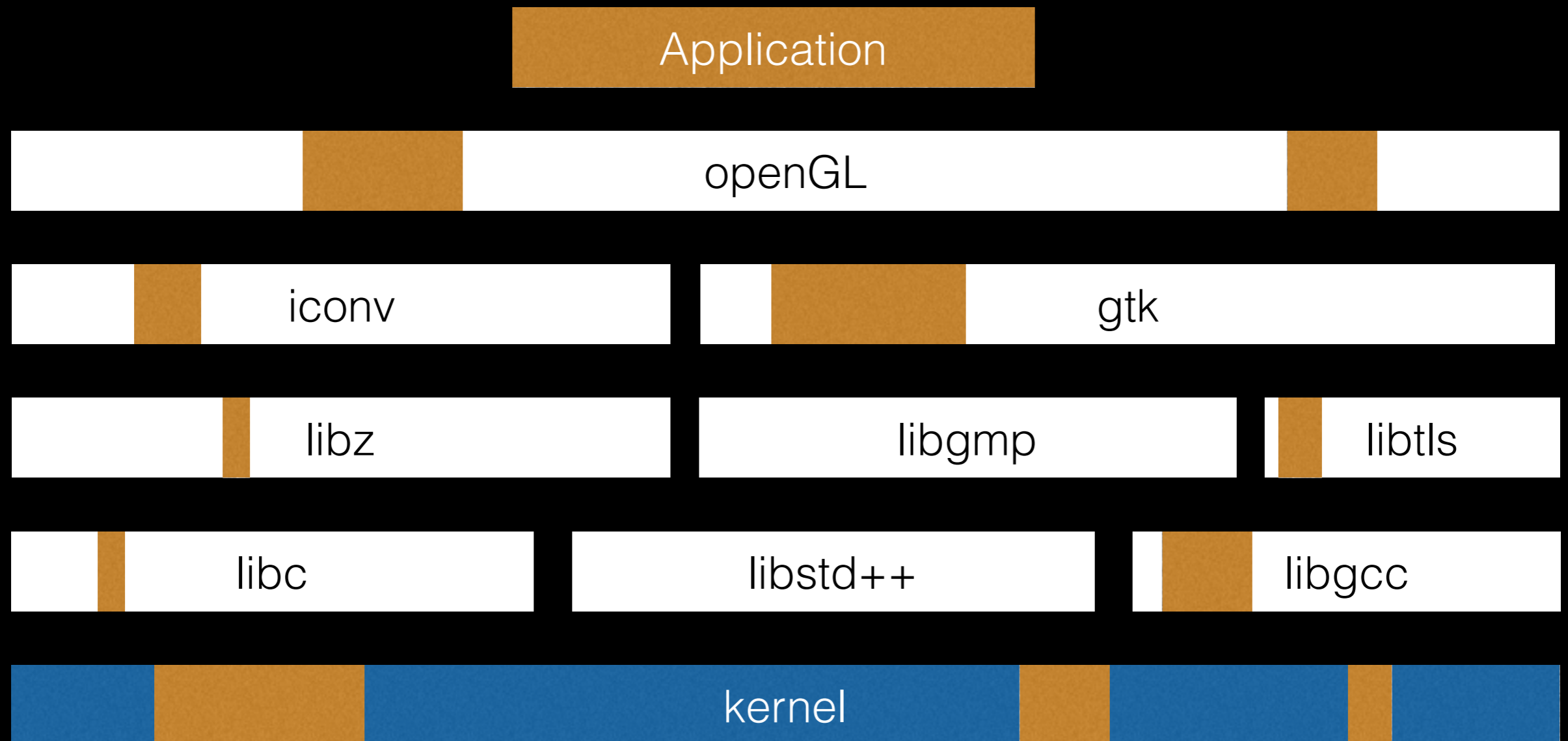
libstd++

libgcc

kernel

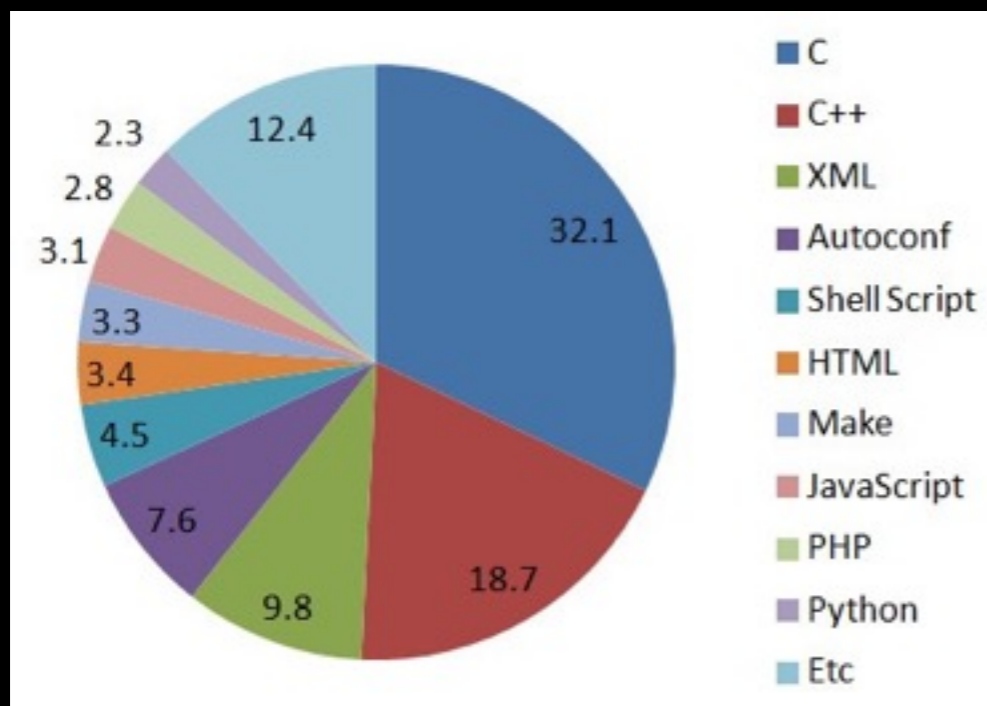
Traditional OS

The System Libraries



Traditional OS

(lack?) of Language Runtime



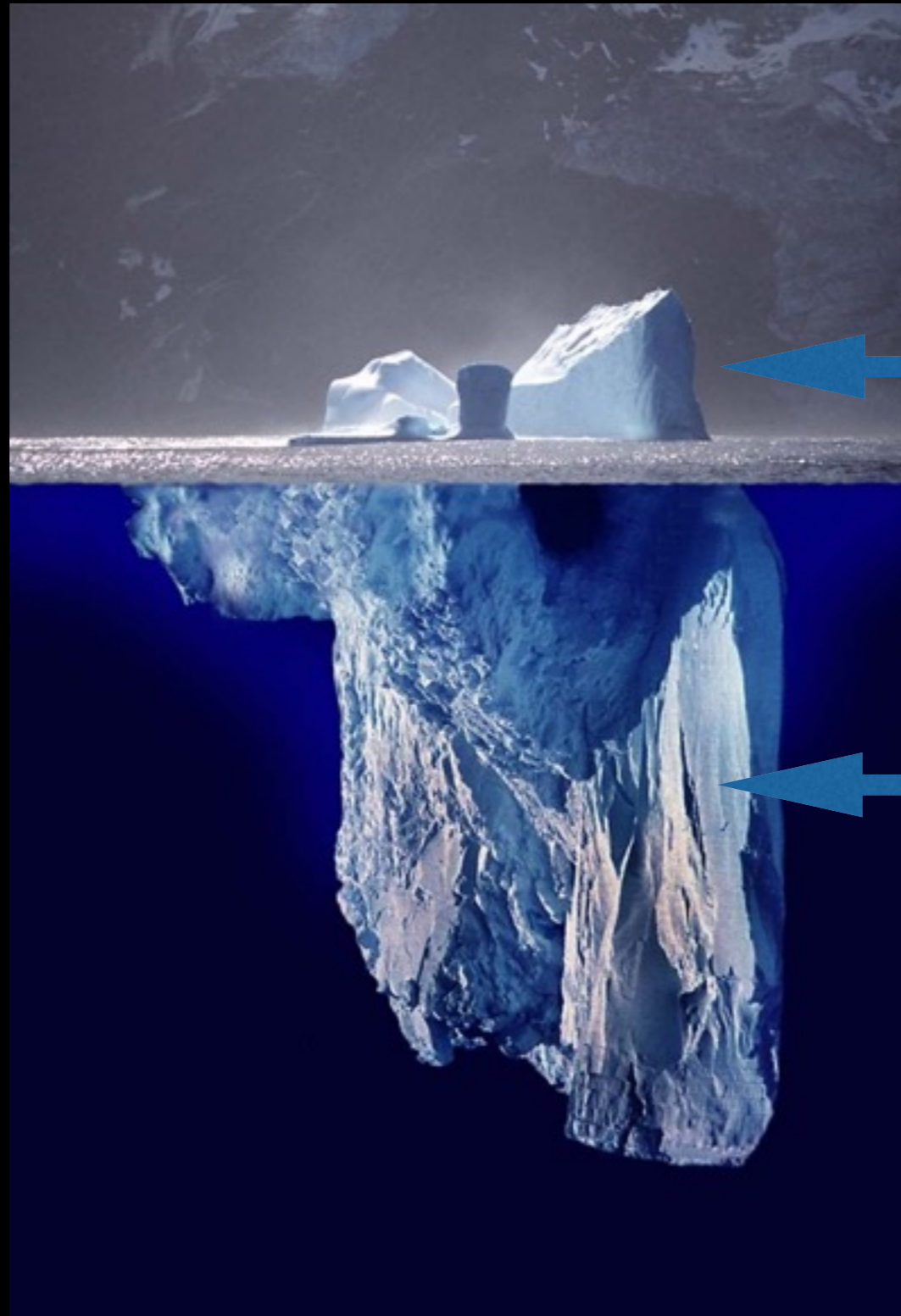
Debian, in 2013

openssl:

- 500k loc
- used by 2/3 of web servers
- 29 CVE in 2015



Traditional OS



Your code

???

Traditional OS

Summary

- current software stack rely 100M+ lines of code
- most of which is written in C
- and a long time ago
- hard to re-use in different contexts

*It's a credit to our civilisation that we managed to build "working" systems, but **can we do better?***

Unikernels

What?


Unikernels

- based on **library OS**
- contains only what is needed
- single process
- single address space
- runs everywhere


Unikernels

MIRAGE OS
HaLVM
Rumprun
...

Hypervisor
(optional)

KVM
Xen™

Hardware



Library OS

the basis

- Long line of research on library OS (since the 60s')
- Consider kernel modules as normal libraries. Use same language and tools than user-space libraries.
- Modern compilers/type-systems can do magic:
 - whole system optimisation: reduce size and increase speed
 - static analysis: remove bugs

Library OS

the basis

- Extreme **specialisation**: embedded systems techniques for general applications
- few things made that easier in the last few years:
 - pervasive use of Hypervisors
 - upstream efforts to turn OS into libraries

Hypervisors

the enablers

- Use hardware isolation extensions (when available)
- Something that creates a virtual machine
- Can run on bare-metal (Xen) or hosted (VMware, VBox, KVM)
- Power the modern "Cloud" (EC2)
- Initial industry pull (mid-00s') was to optimise hardware utilisation

Hypervisors

the enablers

The 1st generation of unikernels (10s') **MirageOS** and **HaLVM** are built on:

- A minimal OS for Xen: mini-OS
- A stable device driver interface

They have now reached more platforms since then: bare-metal, browsers, ...

OS distributions

are turning into unikernels

Recent efforts by OS distros:

- The rump kernel project turns FreeBSD kernel modules into libraries. This has been integrated upstream and is used by the **Rumprun unikernels**.
- LKL is trying to turn the Linux kernel into a set of libraries. Very recent project, not sure where this is going (possible "issues" with the GPL)

Unikernels

Benefits?

Unikernels

Benefits

Improved security

- Small attack surface: static linking and dead-code elimination removes unnecessary services. *No shellshock!*
- Less exposure to general attack
- High-level languages, with static and runtime analysis

Unikernels

Benefits

Increased speed

- Fast boots (see Jitsu: "Just-In-Time Summoning of Unikernels): can boot in less than a TCP packet RTT
- More predictable performance (fewer scheduler layers): lower latencies

Unikernels

Benefits

Efficient resource usage

- Reduced memory footprint, cheap to host on the "Cloud": typical stateless MirageOS app: ~10/20MB of RAM
- Small disk-footprint: DNS server in MirageOS is ~100KB.
- Reduced need for disk-space

Unikernels

Benefits

Immutable Infrastructure

- Can statically link data in your application
- Small enough to be stored in Git
- Enable a new model for updates and upgrade
- Can be sealed: once built, can enable hardware memory-protection so it is **really** immutable

Unikernels

Benefits

Small

Secure

Fast

Pick all three!

Unikernels

At a cost

- Maybe you really want to use all the stack
- Maybe you don't want to rewrite all the libraries in your favorite language
- Maybe you don't know which tool to use to manage the build, ship and run your 100s of unikernels

Unikernels

At a cost

- *Maybe you really want to use all the stack*
 - fair enough
 - you might still want to use similar techniques on desktop, e.g. QubeS+MirageOS

Unikernels

At a cost

- *Maybe you don't want to rewrite all the libraries in your favorite language*
 - why not, it's fun!
 - upstream OS distribution efforts help, i.e. rump kernels

Unikernels

At a cost

- *Maybe you don't know which tool to use to manage the build, ship and run your 100s of unikernels*
 - the tooling is indeed lacking
 - there is hope...

Unikernels

and tooling

Unikernels are at the stage that Linux containers were three years ago before Docker

- Few users
- Hard to build
- Hard to ship
- Hard to run

Clearly this needs to be fixed for widespread use...

Unikernels

and tooling



Solomon Hykes
@solomonstre



Follow

"Wouldn't it be cool if we could deploy and manage unikernels with Docker?" Yes, yes it would! [#dockercon](#)



RETWEETS
58

LIKES
49



Unikernels

Panorama

Panorama

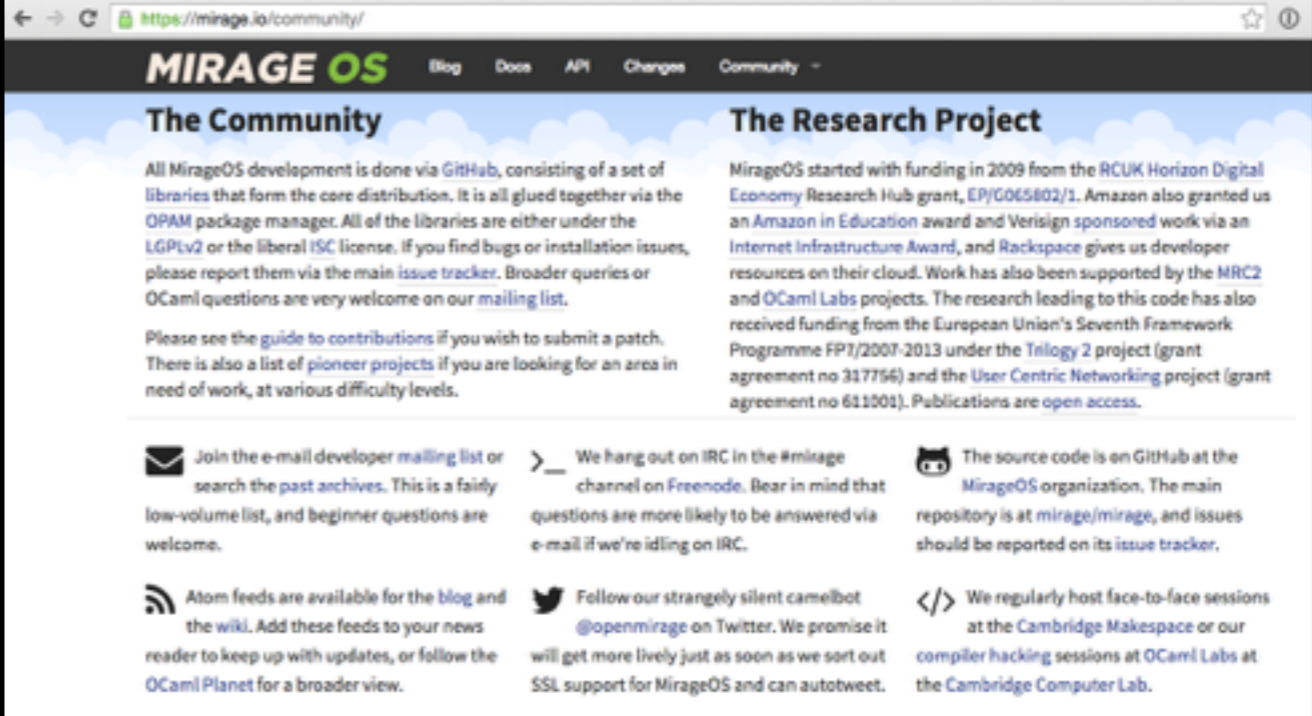
<http://unikernel.org/projects/>

Project	Language	Hypervisor
ClickOS	C/C++	Xen
Clive	Go	bare-metal
Drawbridge	.NET	Microsoft picoprocess
HaLVM	Haskell	Xen
IncludeOS	C++	Xen
LING	Erlang	Xen
MirageOS	OCaml	Xen/Unix
OSv	Java	Xen
Rumprun	POSIX C	bare-metal, Xen, KVM
Runtime.js	Javascript	KVM

MirageOS

- Use OCaml
- Project started in 2007 with Anil Madhavapeddy's PhD
- memory and type-safe network and storage stack from device drivers to TLS
- ~100 OCaml libraries
- compile to Unix process, Xen VM, in-browser JavaScript

<https://mirage.io>



The screenshot shows the website <https://mirage.io/community/>. The page has a dark header with the "MIRAGE OS" logo and navigation links for "Blog", "Docs", "API", "Changes", and "Community". The main content is divided into two columns: "The Community" and "The Research Project".

The Community

All MirageOS development is done via [GitHub](#), consisting of a set of [libraries](#) that form the core distribution. It is all glued together via the [OPAM](#) package manager. All of the libraries are either under the [LGPLv2](#) or the liberal [ISC](#) license. If you find bugs or installation issues, please report them via the [main issue tracker](#). Broader queries or OCaml questions are very welcome on our [mailing list](#).

Please see the [guide to contributions](#) if you wish to submit a patch. There is also a list of [pioneer projects](#) if you are looking for an area in need of work, at various [difficulty](#) levels.

The Research Project

MirageOS started with funding in 2009 from the [RCUK Horizon Digital Economy Research Hub](#) grant, [EP/G065802/1](#). Amazon also granted us an [Amazon in Education](#) award and Verisign sponsored work via an [Internet Infrastructure Award](#), and [Rackspace](#) gives us developer resources on their cloud. Work has also been supported by the [MRC2](#) and [OCaml Labs](#) projects. The research leading to this code has also received funding from the European Union's Seventh Framework Programme FP7/2007-2013 under the [Trilogy 2](#) project (grant agreement no 317756) and the [User Centric Networking](#) project (grant agreement no 611001). Publications are [open access](#).

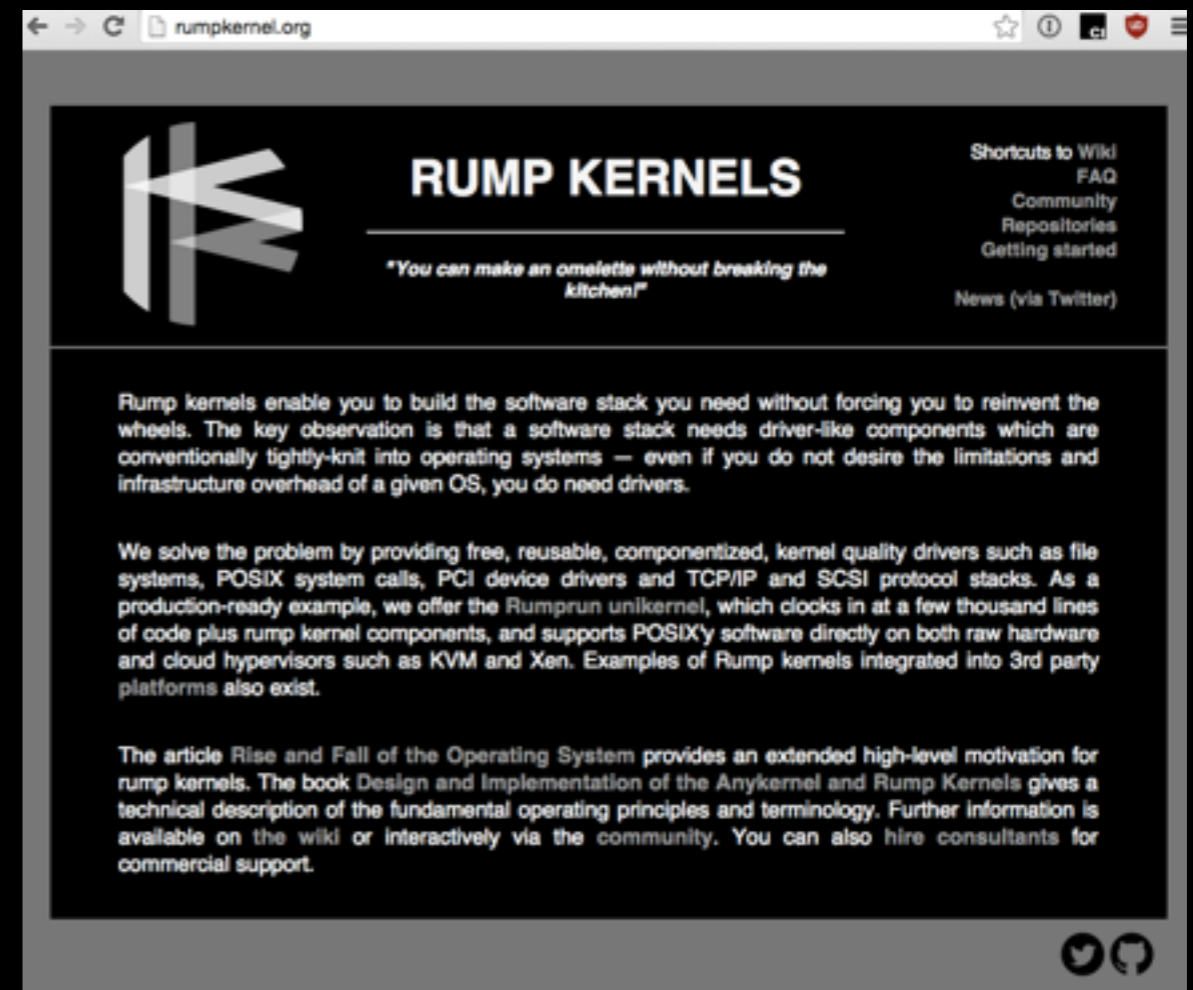
Community Links:

- Join the e-mail developer [mailing list](#) or search the [past archives](#). This is a fairly low-volume list, and beginner questions are welcome.
- We hang out on IRC in the [#mirage](#) channel on [Freenode](#). Bear in mind that questions are more likely to be answered via e-mail if we're idling on IRC.
- The source code is on [GitHub](#) at the [MirageOS](#) organization. The main repository is at [mirage/mirage](#), and issues should be reported on its [issue tracker](#).
- Atom feeds are available for the [blog](#) and the [wiki](#). Add these feeds to your news reader to keep up with updates, or follow the [OCaml Planet](#) for a broader view.
- Follow our strangely silent camelbot [@openmirage](#) on Twitter. We promise it will get more lively just as soon as we sort out SSL support for MirageOS and can autotweet.
- We regularly host face-to-face sessions at the [Cambridge Makerspace](#) or our [compiler hacking sessions](#) at [OCaml Labs](#) at the [Cambridge Computer Lab](#).

Rumprun

- use rump kernels, the FreeBSD library OS
- Project started in 2012 with Antti Kantee's PhD
- compile POSIX applications: useful to port legacy code
- limitations:
 - no forks
 - build system scripts should support cross-compilation

<http://rumpkernel.org>



opam-rumprun

<https://github.com/mato/opam-rumprun>

Experiment to use MirageOS and rumprun

- opam repository with patched packages
- can run the modified packages (41) on any rump kernel support platform, including:
 - bare metal (including ARM boards)
 - KVM
- including the pure-OCaml TLS stack

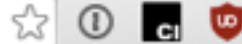
in the process of being upstreamed.

see: <https://github.com/mirage/mirage/issues/479>

MirageOS + TLS + Bitcoins

<https://owme.ipredator.se>

ownme.ipredator.se



You have reached the BTC Piñata.

BTC Piñata knows the private key to the bitcoin address `183XuXTTgnfyfKcHbJ4sZeF46a49Fnihdh`. If you break the Piñata, you get to keep what's inside.

Here are the rules of the game:

- You can connect to port 10000 using TLS. Piñata will send the key and hang up.
- You can connect to port 10001 using TCP. Piñata will immediately close the connection and connect back over TLS to port 40001 on the initiating host, send the key, and hang up.
- You can connect to port 10002 using TCP. Piñata will initiate a TLS handshake over that channel serving as a client, send the key over TLS, and hang up.

And here's the kicker: in both the client and server roles, Piñata requires the other end to present a certificate. Authentication is performed using standard **path validation** with a single certificate as the trust anchor. And no, you can't have the certificate key.

It follows that it should be impossible to successfully establish a TLS connection as long as Piñata is working properly. To get the spoils, you have to smash it.

Before you ask: yes, Piñata will talk to itself and you can enjoy watching it do so.



MirageOS/rumpkernels + DNS

<http://www.skjegstad.com/blog/categories/jitsu/>

www.iitsu.v0.no
forward, hold to see history

Hello World from Jitsu!

Unikernel booted in 0.303204 seconds, 1.227919 seconds ago

Live GC Stats

Allocated Bytes	3m
Head Words	126k
Love Words	124k

Build Manifest

55 packages

Name	Version
unikernel	git
base-bytes	legacy
base-no-ppx	base
base-threads	base
base-unix	base
base64	2.0.0
camp4	4.01+system
channel	1.0.0

MirageOS/Git+browser

<http://roscidus.com/blog/blog/2015/04/28/cuekeeper-gitting-things-done-in-the-browser/>

The screenshot shows the CueKeeper web application interface. At the top, there is a navigation bar with tabs for "Process", "Work", "Contact", "Schedule", and "Review". Below this, there are sub-tabs for "Job" and "Personal". The main content area is divided into two columns. The left column contains a list of tasks, and the right column shows the details for a selected task.

Next actions +

Email +

- Make a Mirage unikernel +
- ✓ ★ *Subscribe to Mirage list*

Reading +

- Learn OCaml +
- ✓ ★ *Read "Real World OCaml"*
- ✓ ★ *Try OCaml tutorials*
- Make a Mirage unikernel +
- ✓ ★ *Follow Mirage tutorial*
- ✓ ★ *Read "My First Unikernel"*
- Start using CueKeeper +
- ✓ ★ *Read wikipedia page on GTD*

Recently completed

- ✓ ★ *Learn-to-use-Git*

Task Details:

- ✓ ★ *Read "My First Unikernel"*
- An action in [Make a Mirage unikernel](#) (show)
- Context: [Reading](#) (show)
- Contact: (no contact)
- Repeats: (never)
- Slower than the official tutorial, but explains what's going on in more detail:
<http://roscidus.com/blog/blog/2014/07/28/my-first-unikernel/>
- (add log entry)
- Created 2015-04-20 19:43 (Mon) (delete)

- ✓ ★ *Follow Mirage tutorial*
- An action in [Make a Mirage unikernel](#) (show)
- Context: [Reading](#) (show)
- Contact: (no contact)
- Repeats: (never)
- The official tutorial works on Linux or OS X: <http://openmirage.org/wiki/install>
- (add log entry)
- Created 2015-04-20 19:40 (Mon) (delete)

- ✓ ★ *Try OCaml tutorials*
- An action in [Learn OCaml](#) (show)
- Context: [Reading](#) (show)
- Contact: (no contact)
- Repeats: (never)
- On-line interactive tutorial: <http://try.ocamlpro.com/>
- Also, lots of useful stuff here: <http://ocaml.org/learn/tutorials/>

Merci pour votre attention!

Credits

Garrett Smith, *Rainbows and Unikernels*

https://www.youtube.com/watch?v=cUvNths_5RA

Adam Wick, *Unikernels: Who, What, Where, When, Why*

<https://www.youtube.com/watch?v=oHcHTFleNtg>

Justin Cormack, *The Road to Unikernels*

<http://roadtounikernels.myriabit.com/>

<https://mirage.io/>

<http://unikernel.org>

Dan Nanni, *Interesting facts about Debian Linux*

<http://xmodulo.com/interesting-facts-about-debian-linux.html>