Unikernels and Docker: From Revolution to Evolution

Richard Mortier, @mort
Anil Madhavapeddy, @avsm
The Next Trillion CPUs
…Or The Next Trillion Problems?

- Inherit existing system issues
  - Security
  - Speed
  - Footprint
- Connected to the network
  - But disconnected from each other
- No professional management
  - Who will apply security updates?

- Must be intrinsically robust
New Devices, Old Software

Examples of generational bugs:

- **Heartbleed**
  - 17% of all Internet secure web servers vulnerable to a single bug, described as "catastrophic" by Bruce Schneier
- **ShellShock**
  - CGI, Web, DHCP all vulnerable to code execution
- Millions of sites vulnerable

- Will never be solved as we cannot physically find all affected IoT devices!
Traditional Software

Application

- openGL
- iconv
- gtk
- libz
- libgmp
- libtls
- libc
- libstd++
- libgcc

KERNEL
Traditional Software: Only A Small Portion Used

Application

 KERNEL

openGL

iconv
gtk

libz
libgmp
libtls

libc
libstd++
libgcc
Microservices: Tip of the Iceberg

No matter how simple the API endpoint, the OS kernel is always a huge dependency:

- Debian 5.0 is 65 million lines
- ...but the Linux Kernel is over 25 million lines ($\frac{1}{3}$) of that!
Revolution: Unikernels!

Unikernels compile your source code into a custom operating system that includes only the functionality required to support the application logic.
MirageOS: A Type-Safe Unikernel Framework

- Written in OCaml, an efficient systems language
- Distributed as many (100+) small libraries
- Self-hosted at mirage.io since 2009
- From-scratch implementations of many protocols including TCP/IP, TLS
- Benefits include:
  - Reduced attack surface
  - Reduced resource use
  - Predictable scheduling
- Many others — see unikernel.org
- Typically language-specific e.g. HaLVM for Haskell or IncludeOS for C++

[Diagram of MirageOS architecture]

Benefits:
- Reduced attack surface
- Reduced resource use
- Predictable scheduling

Typically language-specific e.g. HaLVM for Haskell or IncludeOS for C++

See [Unikernel](https://en.wikipedia.org/wiki/Unikernel) for more information.
Not Reinventing the General Purpose OS!

- Not trying to remake Windows or Linux
- The model is “just enough” systems software to run your application code
- Specialisation all the way down the stack
- Follow devops philosophy:
  - Application developer in complete control
  - Open source library base with liberal licenses
Example: Rebuilding TLS

- MirageOS wanted to avoid legacy, Heartbleed-vulnerable OpenSSL
- New TLS stack written in OCaml from scratch specifically for unikernels
- Coding started in Jan 2015, released in June 2015, see nqsb.io
- Code size: < 10 kloc
  - OpenSSL roughly 350 kloc
  - BoringSSL/LibreSSL 200 kloc
  - PolarSSL 50 kloc
- Most of the protocol written in safe code
  - Not vulnerable to Heartbleed-style bugs

But how to see if this new code works in the real world?
The Bitcoin Piñata

- Put 10 Bitcoins into a unikernel service online built using MirageOS
- Invite the Internet to connect and try to break it
- If anyone breaks in, they can obtain the Bitcoin wallet and win!

Results?
- Hundreds of thousands of attacks against the Bitcoin Piñata
- It was crashed — but rebooted in just a few milliseconds
- Nobody got the Bitcoins out of the Piñata, though some donated…

Bitcoin services is just a library, so easy to do this in a unikernel
Imagine trying to build a Bitcoin Linux kernel module!
Unikernels: Benefits of Specialisation

Performance and Size
• Simple unikernels can be as small as kilobytes in size
• Similar to the “network appliances” that powered early Internet growth

Multi-target
• Recompile application logic with different OS libraries for (e.g.,) IoT, JavaScript
• Portability as cloud evolves: don’t tie your application down needlessly

Security and Correctness
• Type-safety can now infiltrate existing micro-service deployments
• Not limited to just application logic — system services too
Unikernels: Steps Remain

Management
• Unikernel lifecycle management that is application-centric
• Clustering support

Deployment
• Cloud APIs are not geared towards dust clouds
• Image management of unikernel libraries

Ease of build
• Remove need for cross-compilation and integrate with IDE ecosystem
• Much functionality is exposed as libraries, but not all

All of these are solved with Docker!
A new Reason for OCaml

From the OCaml Labs wiki

< Blog:News

Contents [hide]

1 What's the Reason?
2 Reason, OCaml Labs and wider collaboration
   2.1 Build system
   2.2 Opam local
   2.3 Merlin
   2.4 Rtop and OCaml 4.03 support
   2.5 Reason Documentation
3 What's Next?

What's the Reason?  [edit]

Reason is a collaborative open source project released by Facebook today - and we are incredibly excited to be part of it! Reason is a new approachable interface to the OCaml language, with the long-term goal of improving the developer experience by providing a functional syntax and toolchain for writing, building and sharing code quickly and easily.

OCaml is used widely in large-scale projects by a wide range of users; within Facebook to build scalable infrastructure (Hack, Flow, and Infer); providing performance-sensitive applications for Jane Street; and a modular system as part of MirageOS. We are excited to be collaborating on the Reason project which provides the opportunity to employ the benefits of OCaml (such as type inference, fast runtime and bare metal compilation) whilst removing the blockers to progression that have previously existed.
### OCL-Mediation

**Running**

- a month ago

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql</td>
<td>Running</td>
<td>mysql:latest</td>
</tr>
<tr>
<td>mediawiki</td>
<td>Running</td>
<td>oamllabs/infrastructure:mediawiki</td>
</tr>
<tr>
<td>tlstunnel</td>
<td>Running</td>
<td>oamllabs/infrastructure:tlstunnel</td>
</tr>
<tr>
<td>http2https</td>
<td>Running</td>
<td>avsm/http2https:latest</td>
</tr>
</tbody>
</table>
OCamlLabs MediaWiki: What’s Going On?

Apache

Mysql

MediaWiki

Tlstunnel

http2https
Docker is Moving into Diverse Architectures

Operating System

- Linux Containers
- Windows Containers
- Windows HyperV Containers
- WIP: Solaris, FreeBSD, ...

Recent Developments

- Docker on Windows Server TP5 and the Docker Hub
- Docker on the Raspberry Pi
- Build and Use Docker on the IBM POWER Linux
- Docker comes to FreeBSD
Build, Ship, Run

- Linux Containers, Windows Containers and, soon, Unikernels
- No physical infrastructure limitations
- Workloads in the datacenter are on a spectrum from physical machine to container hypervisor
- Only the Docker platform covers this ever-widening scope
- More flexibility for orchestrating hybrid applications
From Revolution to Evolution

How can we run Linux containers seamlessly on OSX and Windows?

• To date:
  VirtualBox and a Linux virtual machine
• Docker for Mac & Windows:
  Semantic OS translation layers convert filesystem and network calls from library hypervisor into Mac/Windows

Results in Windows and Mac applications that “just run” Docker containers

Now in beta! Signup at http://beta.docker.com
Join Us: Become a Pioneer!

We just open sourced three components from Docker for Mac and Windows!

• **HyperKit**, a lightweight virtualization toolkit on OSX
• **VPNKit**, a library toolkit for embedding virtual networking
• **DataKit**, a modern UNIX pipeline framework for distributed components
Join Us: Become a Pioneer!

We just open sourced three components from Docker for Mac and Windows!

- **HyperKit**, a lightweight virtualization toolkit on OSX
- **VPNKit**, a library toolkit for embedding virtual networking
- **DataKit**, a modern UNIX pipeline framework for distributed components

Come along to the Contribute Docker session to hack!
Tomorrow, 3—6pm, Meeting Room 6
THANK YOU

Particular thanks to the team, and to Hannes Mehnert and David Kaloper for leading the Bitcoin Piñata effort!
MirageOS: Some Numbers

<table>
<thead>
<tr>
<th>Appliance</th>
<th>Default</th>
<th>Dead Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>0.449 MB</td>
<td>0.184 MB</td>
</tr>
<tr>
<td>Web Server</td>
<td>0.674 MB</td>
<td>0.172 MB</td>
</tr>
<tr>
<td>Openflow switch</td>
<td>0.393 MB</td>
<td>0.164 MB</td>
</tr>
<tr>
<td>Openflow controller</td>
<td>0.392 MB</td>
<td>0.168 MB</td>
</tr>
</tbody>
</table>

![Graphs and charts showing performance metrics for Mirage and Linux.](image-url)