Unikernels - The next big little thing ?

Snowcamp.io, Grenoble, France, Feb 2017

Mike Bright, @mjbright

Unikernels

Overview

- What are Unikernels?
  - Why do we need them?
  - Domains of application
- Unikernel implementations
  - Clean-slate or legacy
  - Tooling
- Containers and Unikernels
- Demo
- Conclusions

Why this talk?

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker …)
- Who the players are

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker ...)
- Who the players are

1990's: First unikernels - Exokernel and Nemesis (Univ. Cambr)

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker ...)
- Who the players are

1990's: First unikernels - Exokernel and Nemesis (Univ. Cambr)

Jan 2014: ACM - "Unikernels: Rise of the Virtual Library Operating System", Anil Madhavapeddy and David J. Scott

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker ...)
- Who the players are

1990's: First unikernels - Exokernel and Nemesis (Univ. Cambr)

Jan 2014: ACM - "Unikernels: Rise of the Virtual Library Operating System", Anil Madhavapeddy and David J. Scott

Mar 2015: Linux.com - 7 Unikernel Projects to Take On Docker in 2015, Lars Kurth

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker ...)
- Who the players are

1990's: First unikernels - Exokernel and Nemesis (Univ. Cambr)

Jan 2014: ACM - "Unikernels: Rise of the Virtual Library Operating System", Anil Madhavapeddy and David J. Scott

Mar 2015: Linux.com - 7 Unikernel Projects to Take On Docker in 2015, Lars Kurth

Nov 2015: Docker Con Europe demo Cool Hack "Unikernels, Meet Docker!"

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker ...)
- Who the players are

1990's: First unikernels - Exokernel and Nemesis (Univ. Cambr)

Jan 2014: ACM - "Unikernels: Rise of the Virtual Library Operating System", Anil Madhavapeddy and David J. Scott

Mar 2015: Linux.com - 7 Unikernel Projects to Take On Docker in 2015, Lars Kurth

Nov 2015: Docker Con Europe demo Cool Hack "Unikernels, Meet Docker!"

Jan 2016: Docker Blog - Unikernel Systems Joins Docker

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker ...)
- Who the players are

1990's: First unikernels - Exokernel and Nemesis (Univ. Cambr)

Jan 2014: ACM - "Unikernels: Rise of the Virtual Library Operating System", Anil Madhavapeddy and David J. Scott

Mar 2015: Linux.com - 7 Unikernel Projects to Take On Docker in 2015, Lars Kurth

Nov 2015: Docker Con Europe demo Cool Hack "Unikernels, Meet Docker!"

Jan 2016: Docker Blog - Unikernel Systems Joins Docker

Oct 2016: The NewStack - Debunking Unikernels, Idit Levine Dell-EMC

Why this talk?

Curiosity about

- What we can expect to see from Unikernels (and Docker ...)
- Who the players are

1990's: First unikernels - Exokernel and Nemesis (Univ. Cambr)

Jan 2014: ACM - "Unikernels: Rise of the Virtual Library Operating System", Anil Madhavapeddy and David J. Scott

Mar 2015: Linux.com - 7 Unikernel Projects to Take On Docker in 2015, Lars Kurth

Nov 2015: Docker Con Europe demo Cool Hack "Unikernels, Meet Docker!"

Jan 2016: Docker Blog - Unikernel Systems Joins Docker

Oct 2016: The NewStack - Debunking Unikernels, Idit Levine Dell-EMC

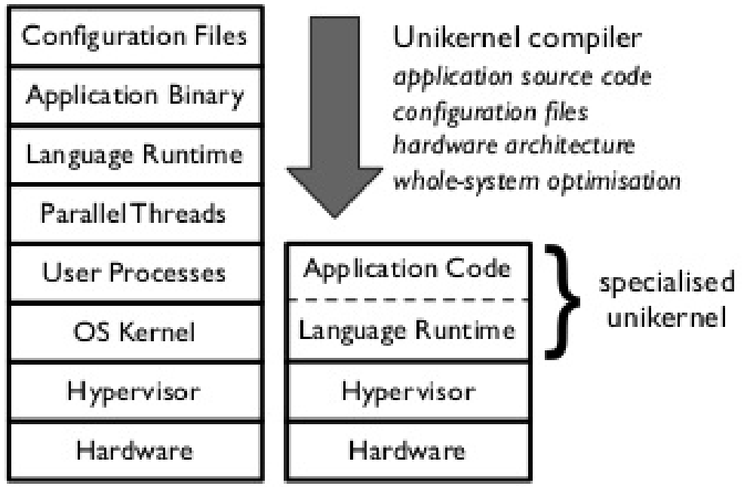
In 2017 ?

- MirageOS 3 will be released
- DockerCon US and EU

What are Unikernels? "Library OS"

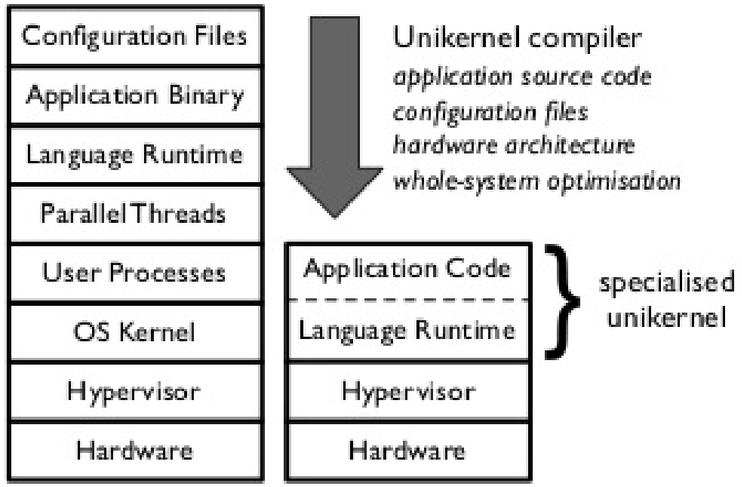
What are Unikernels? "Library OS"

Applications built with only the OS components they actually require, e.g. TCP Stack, DNS, DHCP, NAT, F/w, Disk access.



What are Unikernels? "Library OS"

Applications built with only the OS components they actually require, e.g. TCP Stack, DNS, DHCP, NAT, F/w, Disk access.



Single process(*) applications (no threads, forking or multi-user) (*)

Small size (few lines of code) and very fast to boot

Small attack surface (potentially secure)

High performance - no context switches!

No shell

Why are Unikernels needed?

Think for a moment - What OS do you run?

On what hardware?

Modern OS provide amazing backwards compatibility and features

Why are Unikernels needed?

Think for a moment - What OS do you run?

On what hardware?

Modern OS provide amazing backwards compatibility and features

But an app uses a tiny fraction of those features consuming resources, increasing the attack surface (linux kernel ~ 25 M LOC)

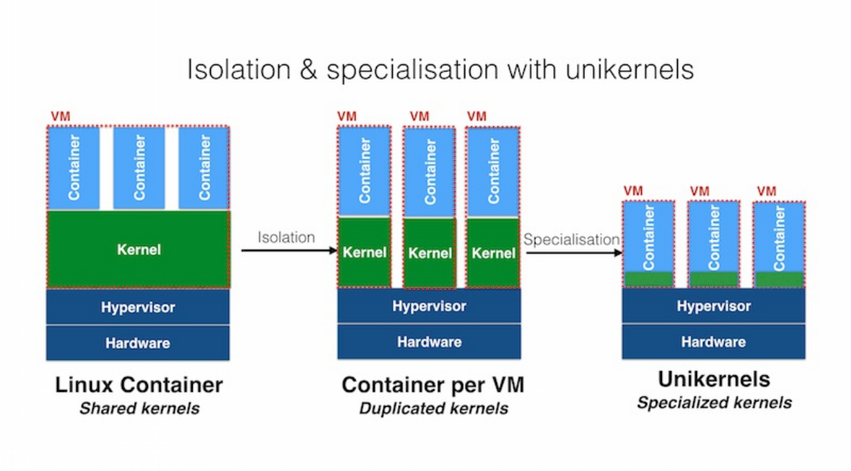
Why are Unikernels needed?

Think for a moment - What OS do you run?

On what hardware?

Modern OS provide amazing backwards compatibility and features

But an app uses a tiny fraction of those features consuming resources, increasing the attack surface (linux kernel ~ 25 M LOC)



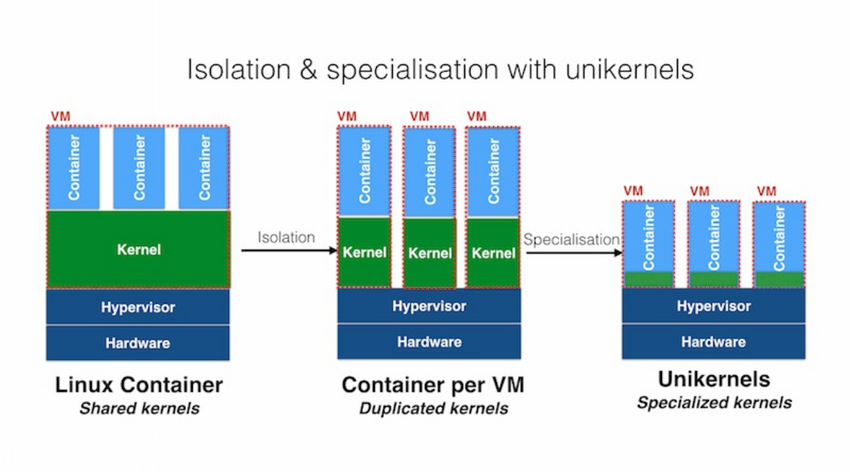
Why are Unikernels needed?

Think for a moment - What OS do you run?

On what hardware?

Modern OS provide amazing backwards compatibility and features

But an app uses a tiny fraction of those features consuming resources, increasing the attack surface (linux kernel ~ 25 M LOC)



Unikernels provide an alternative

But are they a panacea?

The 2 families of Unikernels

Unikernel Implementations - 2 families

There are 2 main classes of Unikernels

Unikernel Implementations - 2 families

There are 2 main classes of Unikernels

Clean-Slate

The Clean-Slate approach emphasizes safety and security.
Same language for application and "Library OS" components.

One example of this approach is MirageOS (written in Ocaml)

Unikernel Implementations - 2 families

There are 2 main classes of Unikernels

Clean-Slate

The Clean-Slate approach emphasizes safety and security.
Same language for application and "Library OS" components.

One example of this approach is MirageOS (written in Ocaml)

Legacy

The Legacy approach favours backward compatibility of existing applications based on POSIX-compatibilities.

One example of this approach is OSv for which there are implementatons of Tomcat, Jetty, Cassandra, OpenJDK

We will see more Unikernel implementations later ...

Application domains for Unikernels

In what domains might they be used?

Cloud Computing

- Small (kB/MB) immutable entities with fast boot times (100's ms).
- Possibility of on-demand servers, µ-services
- Potentially greater security (< LOC)

In what domains might they be used?

Cloud Computing

- Small (kB/MB) immutable entities with fast boot times (100's ms).
- Possibility of on-demand servers, µ-services
- Potentially greater security (< LOC)

NFV (Network Functional Virtualization)

- Cloud but stricter requirements on response times
- Decouple software from the hardware, decompose/chain functions
- Ericsson, NEC, Cisco are active in this domain

In what domains might they be used?

Cloud Computing

- Small (kB/MB) immutable entities with fast boot times (100's ms).
- Possibility of on-demand servers, µ-services
- Potentially greater security (< LOC)

NFV (Network Functional Virtualization)

- Cloud but stricter requirements on response times
- Decouple software from the hardware, decompose/chain functions
- Ericsson, NEC, Cisco are active in this domain

IoT / Embedded / Network Switches

- For low-resource, potentially secure elements (baremetal or µ-vmm ?)
- Build up the "app" instead of stripping down the "OS"

In what domains might they be used?

Cloud Computing

- Small (kB/MB) immutable entities with fast boot times (100's ms).
- Possibility of on-demand servers, µ-services
- Potentially greater security (< LOC)

NFV (Network Functional Virtualization)

- Cloud but stricter requirements on response times
- Decouple software from the hardware, decompose/chain functions
- Ericsson, NEC, Cisco are active in this domain

IoT / Embedded / Network Switches

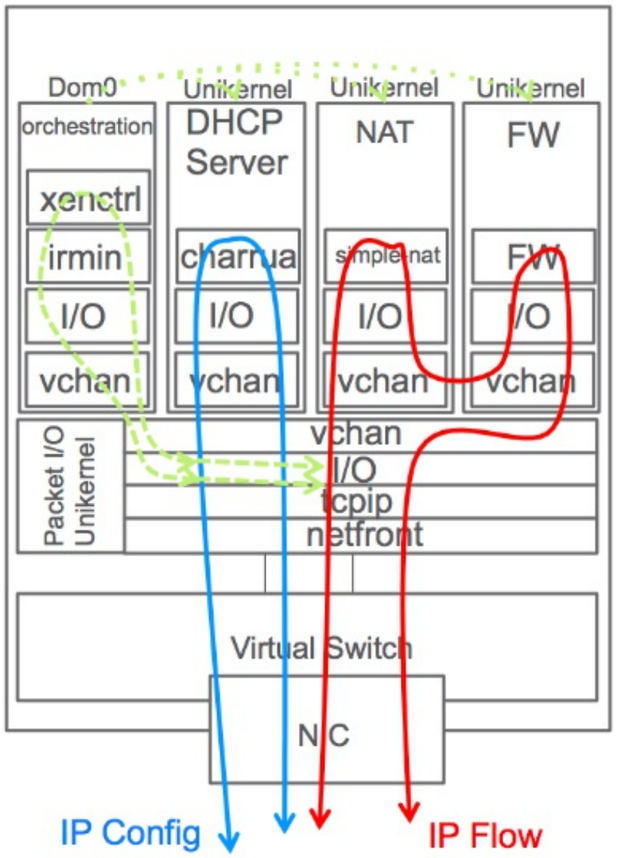
- For low-resource, potentially secure elements (baremetal or µ-vmm ?)
- Build up the "app" instead of stripping down the "OS"

HPC

- Greater performance possible (but may be hard work)

In what domains might they be used? - NFV/SDN

Nano-services boot up in 10-20 msec on demand and are removed when the request completes.



Presented by Ericsson Research, Jan 2016 at SCALE 14x.

Unikernels meet NFV

Ericsson Research Blog

Unikernels.org Blog

IETF draft on Containers for NFV expired Jan 2017

Taken from: draft-natarajan-nfvrg-containers-for-nfv-03.txt

4.2. Instantiation Times

Measurement of time to boot image, up to the 1st RST packet (to a SYN flood).

Technology Type	Time (msecs)
standardvm.xen	6500
standardvm.kvm	2988
Container	1711
tinyx.kvm	1081
tinyx.xen	431
unikernel.osv.kvm	330
unikernels.minios.xen	** 31 **

Note:

- These unikernels include just one application - iperf.
- Tinyx is "Tinyfied Linux" running 4.4.1 kernel - busybox+sshd+iperf
- Standard VM is Debian running 4.4.1 kernel + iperf
- Docker container including iperf

IETF draft on Containers for NFV expired Jan 2017

4.3. Throughput

TCP/IP throughput was measured using iperf from guest to host (to avoid physical medium limitations)

Technology Type	Throughput (Gb/s) Tx	Throughput (Gb/s) Rx
standardvm.xen	23.1	24.5
standardvm.kvm	20.1	38.9
Container	45.1	43.8
tinyx.kvm	21.5	37.9
tinyx.xen	28.6	24.9
unikernel.osv.kvm	** 47.9 **	** 47.7 **
unikernels.minios.xen	** 49.5 **	32.6

Note:

- Throughput depends not just on guest efficiency
- Xen is optimized for Tx but not Rx (similar to ClickOS experience)

IETF draft on Containers for NFV expired Jan 2017

4.4. RTT

Average round-trip time (RTT) measured from an external server using a ping flood.

Technology Type	Time (msecs)
standardvm.xen	34
standardvm.kvm	18
Container	** 4 **
tinyx.kvm	19
tinyx.xen	15
unikernel.osv.kvm	9
unikernels.minios.xen	** 5 **

IETF draft on Containers for NFV expired Jan 2017

4.5. Image Size

We measure image size using the standard "ls" tool.

Technology Type	Size (MBs)
standardvm.xen	913
standardvm.kvm	913
Container	61
tinyx.kvm	3.5
tinyx.xen	3.7
unikernel.osv.kvm	12
unikernels.minios.xen	** 2 **

IETF draft on Containers for NFV expired Jan 2017

4.6. Memory Usage

"top" and "xl" (on Xen) used to measure memory usage:

Technology Type	Usage (MBs)
standardvm.xen	112
standardvm.kvm	82
Container	** 3.8 **
tinyx.kvm	30
tinyx.xen	31
unikernel.osv.kvm	52
unikernels.minios.xen	8

Note:

- OSv pre-allocates memory, e.g for buffers
- Best result is Docker as it has no OS function

IETF draft on Containers for NFV expired Jan 2017

So what conclusions can we draw in the case of NFV?

IETF draft on Containers for NFV expired Jan 2017

So what conclusions can we draw in the case of NFV?

Well it depends of course !!

IETF draft on Containers for NFV expired Jan 2017

So what conclusions can we draw in the case of NFV?

Well it depends of course !!

It depends upon your applications', your organizations' criteria:

- Service agility/elasticity: spin up/down times
- Memory consumption
- Security/Isolation
- Management frameworks
- Compatibility with applications

IETF draft on Containers for NFV expired Jan 2017

So what conclusions can we draw in the case of NFV?

Well it depends of course !!

It depends upon your applications', your organizations' criteria:

- Service agility/elasticity: spin up/down times
- Memory consumption
- Security/Isolation
- Management frameworks
- Compatibility with applications

These are still early days for Unikernels for Cloud Computing.

Hybrid approaches may be appropriate.

Unikernel Implementations ... in more detail

Unikernel Implementations - 2 families

There are 2 main classes of Unikernels

Unikernel Implementations - 2 families

There are 2 main classes of Unikernels

The Clean-Slate approach emphasizes safety and security.
Same language for application and Library OS components.

- MirageOS (Ocaml)
- HalVM (Haskell)
- LING (Erlang)

Unikernel Implementations - 2 families

There are 2 main classes of Unikernels

The Clean-Slate approach emphasizes safety and security.
Same language for application and Library OS components.

- MirageOS (Ocaml)
- HalVM (Haskell)
- LING (Erlang)

The Legacy approach favours backward compatibility of existing applications based on POSIX-compatibilities.

Many applications have been ported

- OSv (Tomcat, Jetty, Cassandra, OpenJDK, ...)
- Rumprun (MySQL, PHP, Nginx)
- Runtimejs
- Clive (Go)

Unikernel Implementations

Technology	Description
ClickOS cnp.neclab.eu	For embedded network h/w. ~5MB images, boots <20ms, 45 µs delay, 100 VMs => 10Gbps
Clive lsub.org	Written in Go. For distributed and cloud.
Drawbridge MS	Research prototype. Picoprocess/container with minimal kernel API surface, and Windows library OS.
Graphene graphene	Securing "multi-process" legacy apps - adds IPC.
HaLVM galois.com	Port of GHC (Glasgow Haskell Compiler) suite. Write apps in Haskell to run on Xen.
IncludeOS includeos.org	Research project for C++ code on virtual hardware.
LING erlangonxen.org	Erlang/OTP runs on Xen.
MirageOS mirage.io	Clean-slate library OS for secure, high-perf network apps. More than 100 MirageOS libraries plus OCaml ecosystem.
OSv osv.io Cloudius	Run Linux binaries (w. limitations), supports C/C++, JVM, Ruby, Node.js
Rumprun	FreeBSD - Runs POSIX s/w on BM or VM (Xen)

Unikernel implementations - MirageOS/Ocaml

Clean-Slate

MIRAGE OS

https://mirage.io/

OCaml-Based

MirageOS "Library OS" components are written in Ocaml .

ML-derived languages are best known for their static type systems and type-inferring compilers.

OCaml unifies functional, imperative, and object-oriented programming under an ML-like type system.

OCaml has extensive libraries available

(Unison sync utility)

Unikernel implementations - MirageOS-2

Clean-Slate

MIRAGE OS

https://mirage.io/

OCaml-Based

MirageOS Unikernels are based on the Mirage-OS Unikernel base (OS library).

The mirage tool is used to build Unikernels for various backends:

- Xen Hypervisor (PV)
- Unix (Linux or OS/X binaries)
- Browser (via Ocaml->JS compiler !!)
- MirageOS 3 (/Solo5) will support kvm (/ukvm)
- Even an experimental BM backend for Raspberry Pi

Unikernel implementations - MirageOS-2

Clean-Slate

MIRAGE OS

https://mirage.io/

OCaml-Based

MirageOS Unikernels are based on the Mirage-OS Unikernel base (OS library).

The mirage tool is used to build Unikernels for various backends:

- Xen Hypervisor (PV)
- Unix (Linux or OS/X binaries)
- Browser (via Ocaml->JS compiler !!)
- MirageOS 3 (/Solo5) will support kvm (/ukvm)
- Even an experimental BM backend for Raspberry Pi

Building applications for unix or xen

```
mirage configure -t unix
make
./mir-console

mirage configure -t xen
make
****xen create ./mir-console.xen
```

Unikernel implementations - MirageOS - Use Cases

Clean-Slate

MIRAGE OS

https://mirage.io/

- BNC Pinata: http://ownme.ipredator.se/
- Networking applications
  - e.g. CyberChaff "false network hosts"
- PayGarden, Sean Grove

  "Baby steps to unikernels in production"
  - Too painful to create/configure AMI images on AWS
  - Solo5 allows to create KVM images deployable on GCE

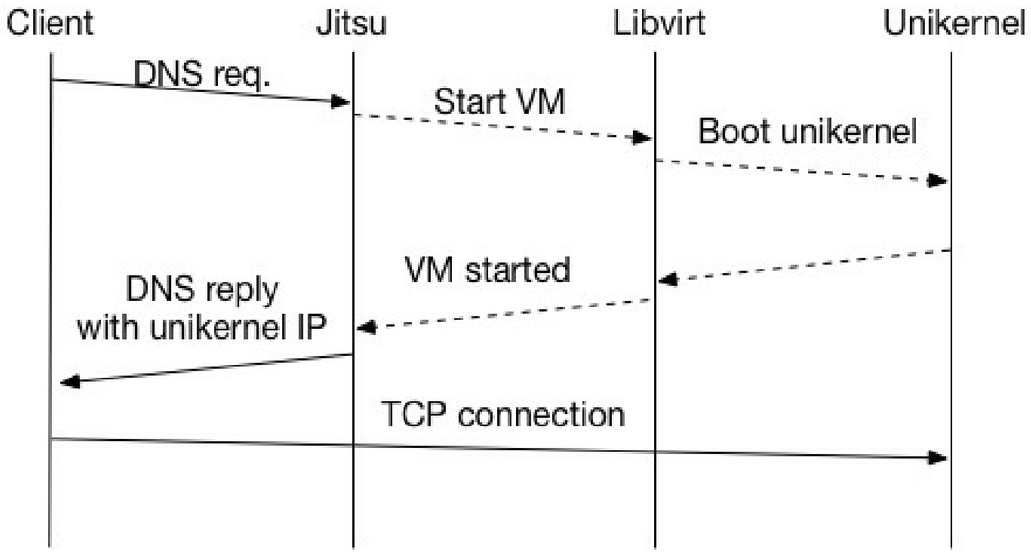
Unikernel Tooling

- Unik [EMC-Dell]: "The Unikernel Compilation and Deployment Platform" (+ image hub)
  - rumprun: Python, Node.js and Go
  - OSv: Java, Node.js, C and C++
  - IncludeOS: C++
  - MirageOS: OCaml
- Solo5 [IBM] : An alternative unikernel-base for MirageOS
  - Provides qemu/KVM support for MirageOS
  - Is currently being integrated into MirageOS 3 beta
- ukvm [IBM]: An alternative VM Monitor
  - a "library hypervisor"
- capstan : OSv build tool (+ image hub)

Unikernel Tooling

MirageOS jitsu : "Just-In-Time Summoning of Unikernels"

A DNS server that starts unikernels on demand.



Tested with MirageOS and Rumprun unikernels.

https://github.com/mirage/jitsu

Unikernels and Containers : My guess ...

So what about Containers? ... and why did Docker buy Unikernel Systems?

Unikernels and Containers : My guess ...

So what about Containers? ... and why did Docker buy Unikernel Systems?

- Unikernel Systems are involved in MirageOS/Xen
- Unikernels already used as specific functions in "Docker for Mac"

Unikernels and Containers : My guess ...

So what about Containers? ... and why did Docker buy Unikernel Systems?

- Unikernel Systems are involved in MirageOS/Xen
- Unikernels already used as specific functions in "Docker for Mac"
- No-brainer: Provide build/ship/run tools for Unikernels
    - build:
        - tools to facilitate building Unikernels
        - test: run Unikernels in containers to faciltate testing https://github.com/mato/docker-unikernel-runner
    - ship: Docker registry extended to provide Unikernel images
    - run: Docker Swarm orchestrates tasks incl. Unikernels

Unikernels and Containers : My guess ...

So what about Containers? ... and why did Docker buy Unikernel Systems?

- Unikernel Systems are involved in MirageOS/Xen
- Unikernels already used as specific functions in "Docker for Mac"
- No-brainer: Provide build/ship/run tools for Unikernels
  - build:
    - tools to facilitate building Unikernels
    - test: run Unikernels in containers to faciltate testing https://github.com/mato/docker-unikernel-runner
  - ship: Docker registry extended to provide Unikernel images
  - run: Docker Swarm orchestrates tasks incl. Unikernels
- Secure Container deployments through hybrid solutions
  - Secure front-ends made of unikernels
    - e.g. for OCaml MediaWiki (http2https, tlstunnel, ...)
  - Containers for backend
- Surprises? ...

Demo

DeferPanic - Unikernel IaaS - https://deferpanic.com/

runtime.js - Node.js Unikernel - https://github.com/runtimejs/example-web-server

4 unikernel demos - Look Ma, no OS! - https://github.com/technolo-g/lookma

Conclusions

- Much work needs to be done
  - to make them easy to build, deploy, debug
  - We will see easier to use solutions
    - Whatever Docker plan to surprise us with
    - Unik will facilitate building, deploying multiple technologies
    - Solo5 will allow mixing of technologies
- Several disparate technologies today
  - but some efforts to synergize
- Unikernels are an interesting complimentary technology to containers
  - We can expect hybrid solutions
- 2017 will be an interesting year for Unikernels

Resources

Scoop.it Unikernels	www.scoop.it/t/unikernels
Youtube Playlist	youtube.com/.../unikernels
Wikipedia	en.wikipedia.org/wiki/Unikernel
unikernels.org	unikernels.org
mirageos.io	mirageos.io mirage.io/docs/papers
OReilly "Unikernels"	Free download
@unikernel	@unikernel
github.com/ocamllabs	ocamllabs
github.com/mirage	MirageOS

@mjbright

Thank you

Q&A