SCALE 13X

The Next Generation Cloud: The Rise of the Unikernel

Russell Pavlicek
Xen Project Evangelist
Russell.Pavlicek@XenProject.org

About the Old, Fat Geek Up Front

- Linux user since 1995; became a Linux advocate immediately
- Delivered many early talks on Open Source Advocacy
- Former Open Source columnist for Infoworld, Processor magazines
- Former weekly panelist on “The Linux Show”
- Wrote one of the first books on Open Source: Embracing Insanity: Open Source Software Development
- 30 years in the industry; 20+ years in software services consulting
- Currently Evangelist for the Xen Project (employed by Citrix)
- Over 75 FOSS talks delivered; over 150 FOSS pieces published

Why Am I Talking About This?

- I am not a unikernel implementer
- I am Evangelist for Xen Project, which is at the forefront of unikernel development
- There are a number of people implementing unikernels and discussing what they've done, but relatively few discussing the big picture
- This talk will attempt to examine both the forest and the trees:
  - We will discuss the value of the unikernel movement
  - We will examine several prominent unikernels and their uses

Topic 1: The Forest

The importance of unikernels

The Cloud We Know

- Field of innovation is in the orchestration
  - The Cloud Engine is paramount (OpenStack, CloudStack, etc.)
  - Workloads adapted to the cloud strongly resemble their non-cloud predecessors
    - Some basic adaptations to facilitate life in the cloud, but basically the same stuff that was used before the cloud
    - Applications with full stacks (operating system, utilities, languages, and apps) which could basically run on hardware, but are run on VMs instead.
    - VMs are beefy; large memory footprint, slow to start up
    - It all works, but its not overly efficient
    - 10s of VMs per physical host

The Next Generation Cloud

- Turning the scrutiny to the workloads
  - Should be easier to deploy and manage
  - Smaller footprint, removing unnecessary duplication
  - Faster startup
  - Higher levels of security
  - 1000s of VMs per host

The New Stuff: Docker & Containers

- Makes deployment easier
- Smaller footprint by leveraging kernel of host
- Less memory needed to replicate shared kernel space
- Less disk needed to replicate shared executables
- Really fast startup times
- Higher number of VMs per host

Docker Downsides

- Improvements, yes; but not without issues
  - Can't run any payload that can't use host kernel
  - Potential limits to scaleability
    - Linux not really optimized for 1000s of processes
  - Security
    - Security is a HUGE issue in clouds
    - Still working on real security; someday...
    - At LinuxCon North America 2014, Docker CEO doesn't even identify security as one of the top priorities
    - Google & others run Docker in VMs when they need security

The Unikernel: A Real Cloud Concept

- Very small
- Very efficient
- Very quick to boot
- And very, VERY secure!
- It's a Green (energy) technology which saves you green (cash)
- Many unikernels already exist, including Mirage OS, a Xen Project Incubator Project

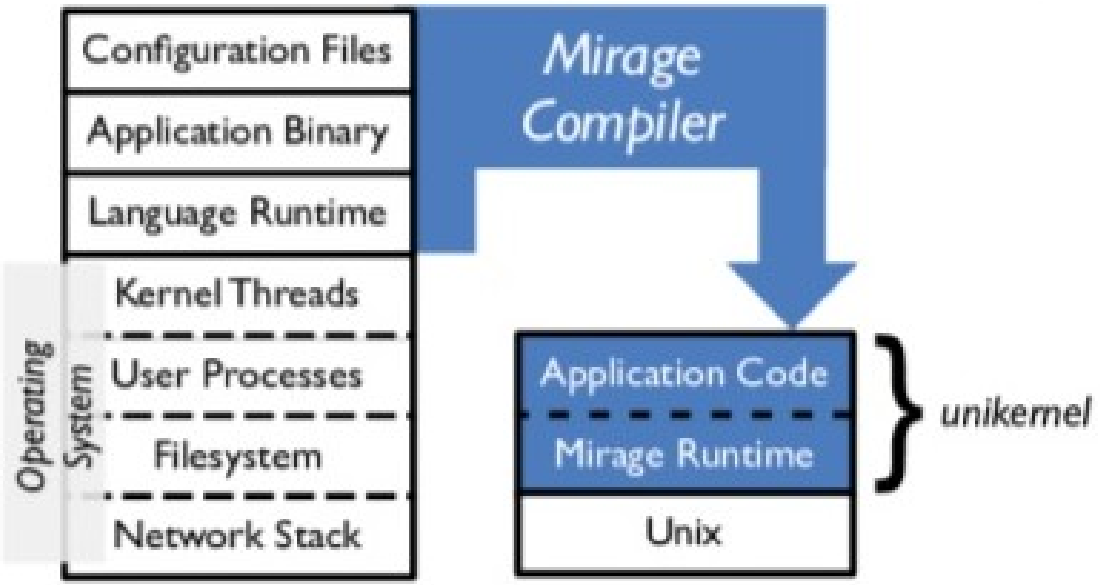
What is a Unikernel? From Mirage OS

Unikernels *are specialised virtual machine images compiled from the modular stack of application code, system libraries and configuration*

Unikernel Approach: Mirage OS

Swap system libraries to target different platforms:

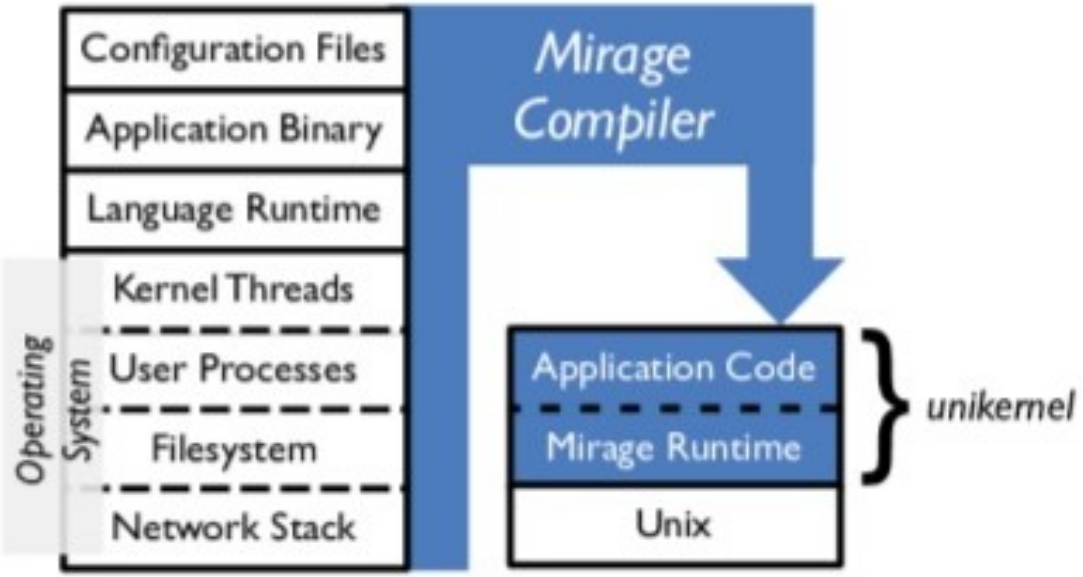
develop application logic using native Unix.



Unikernel Approach: Mirage OS

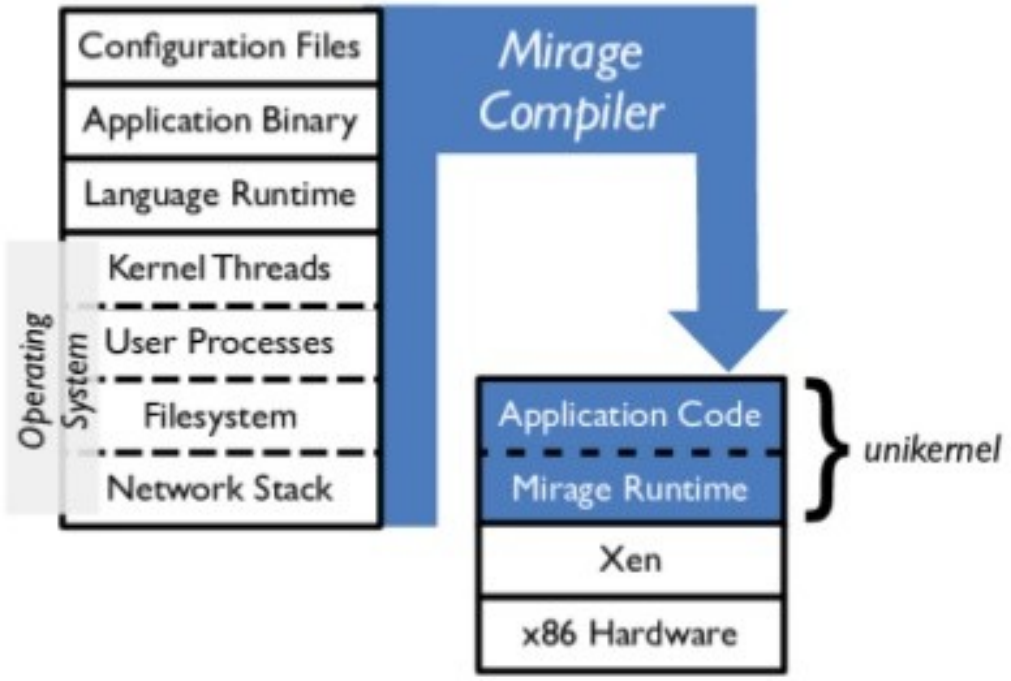
Swap system libraries to target different platforms:

test unikernel using Mirage system libraries.



Unikernel Approach: Mirage OS

Swap system libraries to target different platforms:
deploy by specialising unikernel to Xen.



Unikernel Concepts

- Use just enough to do the job
  - No need for multiple users; one VM per user
  - No need for a general purpose operating system
  - No need for utilities
  - No need for a full set of operating system functions
- Lean and mean
  - Minimal waste
  - Tiny size

Unikernel Concepts

- Similar to an embedded application development environment
  - Limited debugging available for deployed production system
  - Instead, system failures are reproduced and analyzed on a full operating system stack and then encapsulated into a new image to deploy
  - Tradeoff is required for ultralight images

What Do the Results Look Like?

- Mirage OS examples:
  - DNS Server: 449 KB
  - Web Server: 674 KB
  - OpenFlow Learning Switch: 393 KB
- LING metrics:
  - Boot time to shell in under 100ms
  - Erlangonxen.org memory usage: 8.7 MB
- ClickOS:
  - Network devices processing >5 million pkt/sec
  - 6 MB memory with 30 ms boot time

What About Security?

- Type-Safe Solution Stack
  - Can be certified
  - Certification is crucial for certain highly critical tasks, like airplane fly-by-wire control systems
- Image footprints are unique to the image
  - Intruders cannot rely on always finding certain libraries
  - No utilities to exploit, no shell to manipulate

Topic 2: The Trees

Examples of unikernels

What's Out There Right Now?

- Mirage OS, from the Xen Project Incubator
- HaLVM, from Galois
- LING, from Erlang-on-Xen
- ClickOS, from NEC Europe Labs
- OSv, from Cloudius Systems
- And that's just the beginning...

Mirage OS

- From the Xen Project Incubator
- Language support: Ocaml
- Hypervisor support: Xen Project
- V2.0 released in 2014
- General purpose devices
- Can be run on Amazon EC2
- http://www.openmirage.org/

HaLVM

- Galois, Inc.
- Language support: Haskell
- Hypervisor support: Xen Project
- Originally designed to prototype operating system components
- Now suitable for creating network devices
- https://galois.com/project/halvm/

LING

- Erlang-on-Xen project
- Language support: Erlang
- Hypervisor support: Xen Project
- Use cases include Zero-Footprint Cloud
- http://erlangonxen.org/

ErlangOnXen.org Website

System info:

```
os_type:     ling
compat_rel:  17
wordsize:    4
smp_support: false
heap_type:   private
schedulers:  1
hipe_architecture: none
machine:     LING
```

Modules:

```
init
global_group
hipe_unified_loader
group
cow_http_hd
beam_dtl
clustering_dtl
iops_handler
contact_dtl
inet
 (and 140 more)
```

Statistics:

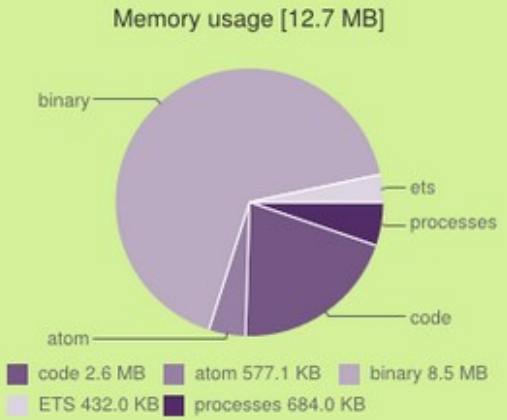
```
context_swithes: 1714004
GC, runs:        3267409
GC, reclaimed:   2.0 GB
reductions:      126941023
run_queue:       0
runtime:    134575
wall_clock: 518565825
```

Processes:

```
<0.73.0>
<0.74.0>
<0.57.0>
<0.52.0>
kernel_safe_sup
user_drv
user
<0.50.0>
<0.51.0>
standard_error_sup
 (and 50 more)
```

Applications:

```
crypto
webling
kernel
stdlib
cowlib
cowboy
ranch
```



ErlangOnXen Zerg Demo

300 sec is how long it takes to launch a Linux instance to availability on Amazon EC2.

50 sec is the time between power on and the lock screen for an Android phone.

0.4 sec ago is when we received your request. Within this time we managed to create a new Xen instance, boot it, and run the application that rendered the page you are viewing. By the time you are done reading this, the instance will be gone.

Why is this important? The fast startup is the cornerstone of the super-elastic clouds of the future. Think personal Facebook, personal Gmail, personal bank at the new level of privacy and security... more

This demo was made possible by Erlang on Xen.

ClickOS

- NEC Europe Labs
- Language support: C, C++, Python
- Hypervisor support: Xen Project
- V0.2 released in 2014
- Suited for Network Function Virtualization (NFV) devices
- http://cnp.neclab.eu/clickos/

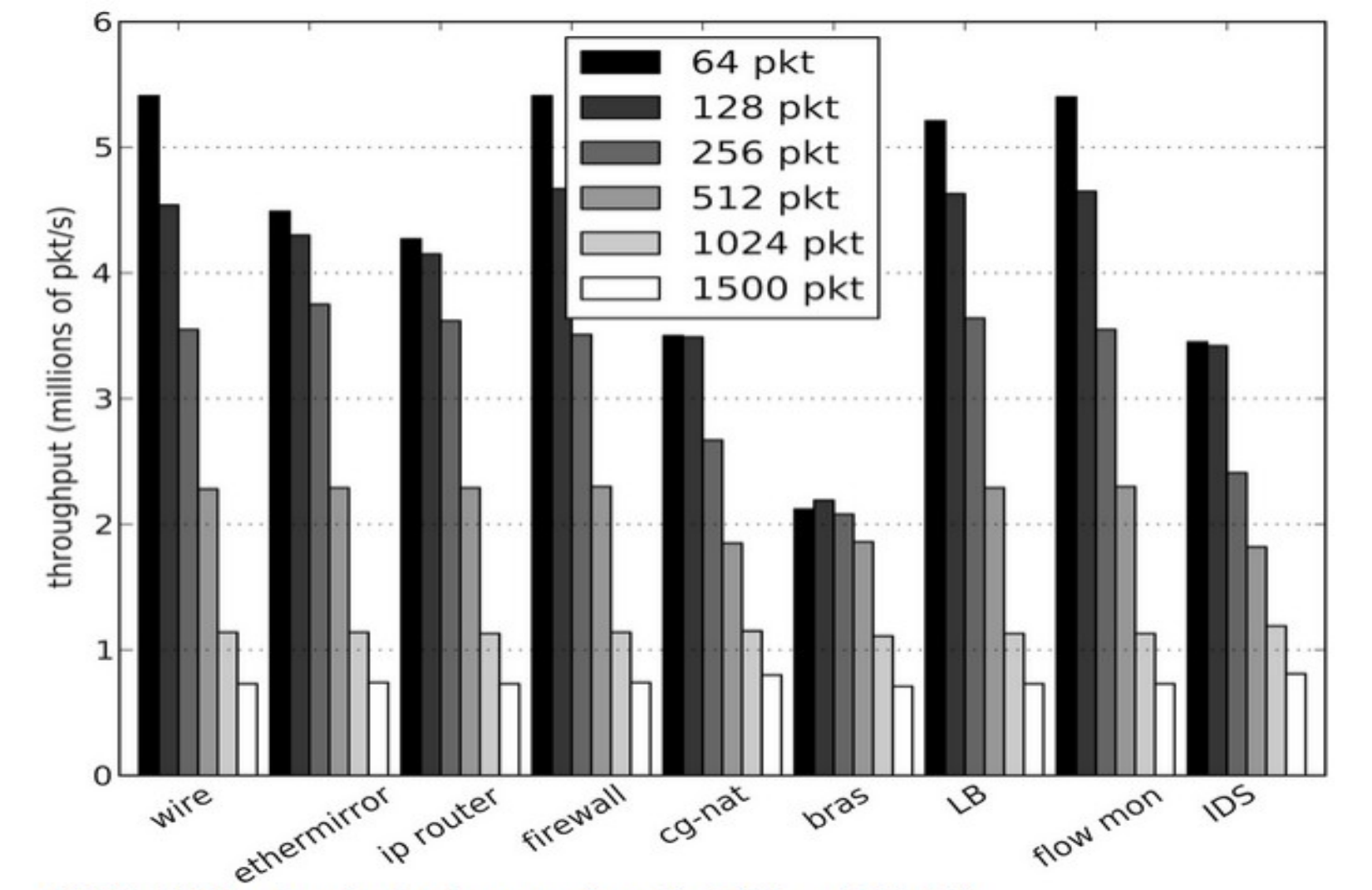
ClickOS Website

The ClickOS website says:

A minimalistic, tailor-made, virtualized operating system to run Click-based middleboxes.

The recent trend towards network function virtualization (NFV) proposes to shift middlebox processing from hardware-based appliances to software running on inexpensive, commodity hardware (e.g., x86 servers with 10Gb NICs). Towards this goal we developed ClickOS, a high-performance, virtualized software middlebox platform. It consists of the Click modular router software running on top of MiniOS (a minimalistic OS available with the Xen sources), plus optimizations to network I/O in order to drive 10 Gb/s throughput for almost all packet sizes. These virtual machines are small (6MB), boot quickly (in about 30 milliseconds) and add little delay (45 microseconds).

ClickOS Throughput



ClickOS middlebox throughput performance using a single CPU core for the VM.

OSv

- Cloudius Systems
- Language support: C, C++, Java
- Hypervisor support: Xen Project, KVM, VMware
- Slightly different from “standard” unikernels
  - Kind of “fat”
  - Full Java JVM stack, minus multi-processes (threads yes, forks no)
  - Can run almost any JAR file
- NFV optimized
- http://osv.io/

OSv Website

OSv website lists some use cases:

Network functions virtualization

Virtualizing network devices requires extreme low latency and high network throughput. OSv, with its Network Channels-based network stack, removes bottlenecks at the guest OS level.

Java application server

The user can upload an application WAR file, via a REST API, and the application runs without further configuration. The deployment process can be connected to a continuous integration system or IDE.

C and C++-based applications

Several C and C++ applications have been ported by OSV developers or third parties. Porting additional applications often requires only a two-line Makefile change. OSv can use unmodified shared libraries built on and for Linux. The images containing these applications can be downloaded and deployed as needed, with lower overhead than on legacy guest OSs.

Horizontal scaling

OSv's sub-second boot time makes it ideal for NoSQL and other applications requiring horizontal scaling or failover. It is often faster to boot a fresh OSv guest than to fail over to an already-running guest.

What About the Unikernel Ecosystem?

- If this is more than just a few isolated experiments in unikernel concepts, we'd expect to see some advances in the general ecosystem
- The unikernel ecosystem is forming:
  - Jitsu (https://github.com/MagnusS/jitsu)
  - Mini-OS (http://wiki.xenproject.org/wiki/Mini-OS)
  - Rump Kernels (http://rumpkernels.org/)
  - Xen Project itself

Jitsu

The Jitsu Website says:

Just-In-Time Summoning of Unikernels

- *Jitsu is a forwarding DNS server that automatically starts virtual machines (VMs) on demand. When a DNS query is received, jitsu first checks for a local VM that is mapped to the requested domain. If a VM is found, the VM is started and its IP is returned to the client. Otherwise, the request is forwarded to the next DNS server. If no DNS requests are received for the VM within a given timeout period it is automatically stopped.*
- *Although Jitsu can be used with any VM that can be controlled with libvirt, it is mainly intended for use with unikernels that can be started quickly and be able to respond to the client request within the time it takes to send the DNS response.*

Mini-OS

- Small basic unikernel
- Distributed with Xen Project source
- Base for others to build their unikernels
  - ClickOS, for example

Rump Kernels

- Come out of the NetBSD camp
- Uses the Anykernel architecture
- Supports Xen Project, bare metal, userspace environments

Xen Project as Ecosystem Enabler

- Work proceeds on support for 1000s of VMs per host
  - Recent redesign of Event Channels removes obstacles to uncap VM growth (theoretically, into millions of VMs)
  - Currently, performance is strong up to around 300 VMs per host
  - Other areas identified and targeted to enable 2000-3000 VMs per host
- Paravirtualization makes creation of a unikernel much simpler
  - Simpler PV interfaces remove need for complex H/W drivers

And More To Come...

- Arrakis (http://arrakis.cs.washington.edu/)
- Clive (http://lsub.org/ls/clive.html)

Are Unikernels a Panacea?

- Nope!
  - But it doesn't have to be a panacea to return value
  - There will always be large databases and beefy apps which won't fit in this mold
  - The truth is that different problems are likely to require different optimal solutions for the foreseeable future
  - It is likely that the solution spectrum of the next few years will include a blend of unikernels, containers, and standard virtualization
  - But the arrival of unikernels means that the bar to efficiency has been raised to new heights

Open Source Leading the Way

- This is an example of how Open Source is working to expand horizons of the cloud
  - The closed source cloud just isn't the way to go
  - The real innovation in cloud is in Open Source
  - Xen Project is at the forefront of new cloud thinking, incubating and facilitating new technologies, including unikernels
  - Friends don't let friends go closed source in the cloud!

The Xen Project Difference

- The Cloud is too critical to leave to hypervisors which are not working to create the future
  - If your hypervisor is just focused on yesterday's payloads, it won't help you get to the next generation cloud
  - Select a hypervisor which is innovating – and Open Source
  - Xen Project is busy moving the cloud forward

Questions?

Russell.Pavlicek@XenProject.org
Twitter: @RCPavlicek

Thanks to the Mirage OS team for the use of their images. Thanks to NEC Europe Ltd (ClickOS) and ErlangOnXen (LING) for the use of images from their respective websites.

This presentation is available in the Presentations Section of XenProject.org