

## Unikernel-powered transient microservices: Changing the face of software architecture

Russell Pavlicek  
rcpavlicek@yahoo.com

# 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
- Formerly Evangelist for the Xen Project, now looking for new opportunities
- Over 100 FOSS talks delivered; over 200 FOSS pieces published

# Why Am I Talking About This?

- I am not a unikernel implementer
- I had been 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, and almost no one talking about the changes to S/W architecture
- 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
- The existence of these unikernels will alter the architecture of the cloud. Microservices will become smaller, faster, and more transient than today.

# The Current Basis for Architecture

- Workloads are essentially Persistent, even in clouds
  - Clouds may move the the workload around, but the workload itself is virtually identical to workloads in the pre-cloud era
- Someone has to authorize their birth, check their health, and authorize their shutdown
- Services tend to be long-lived, because it takes so long to start one up or shut one down
  - Ever get stuck at a Point-Of-Sale terminal run off a modem?
- Services will often sit idle for long periods waiting for a client to make a request
  - Resources needed by the service are, therefore, underutilized

# A New Basis for Architecture

- Imagine workloads which are as nimble as the clouds which deploy them
- Instant startup and shutdown
  - Able to be deployed the instant the need appears
- Lasting only as long as the immediate need
- And much more secure than what we have now
- Key capability to implement the Internet Of Things (IoT)
- This is a Transient workload

# How Did We Get Here?

- The history of computing tells us why we have a love with Persistent machines
- Once upon a time, machines were *really expensive*
  - When I was in college, our student computer center consisted of one machine – a DEC PDP 11/34a – for the entire campus
  - 248kb addressable memory
  - 25 MB total disk storage (not including 8” floppies)
  - No virtualization available
  - One machine had to do everything for everyone, so we loaded it down with thousands of programs
  - This beast cost over US\$100K fully configured
  - Startup and shutdown was slow – minutes. So Persistent service was a necessity

# How Does This Match Today's Reality?

- Hardware today is dirt cheap!
- A smart phone is *several orders of magnitude more powerful* than my college's old PDP-11; we frequently send PCs to a landfill which would have been supercomputers by comparison
- Memory and storage are cheap and plentiful
- Virtualization is stable and available
- Startup and shutdown is much quicker
- **So why are we still focused on building Persistent machines with 1000s of programs like we used to when hardware was expensive?**

# Let's Upgrade Our Architecture

- Much modern solution architecture is focused on leveraging physical components designed in 1980
- Let's rethink software architecture to best use the current truths, and let go of the assumptions created in a bygone era
- Let's start with the Cloud and ask, “What should your machine images look like to be optimal today?”

# A View From the Clouds

What does a Really Good Cloud look like?

# The Cloud So Far

- The major field of innovation is in the orchestration
  - The Cloud Engine is paramount (OpenStack, CloudStack, etc.)
  - But the workloads adapted to the cloud strongly resemble their Persistent 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

- Let's turn our scrutiny to the workloads
  - Should be easier to deploy and manage
  - Smaller footprint, removing unnecessary duplication
  - Faster startup
  - Mixture of Persistent and Transient microservices
  - 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 scalability
    - Linux not really optimized for 1000s of processes
  - Security
    - Security is a HUGE issue in clouds
    - Still working on security which brings containers up to the level of current solutions
      - Cumbersome add-ons won't work; security needs to be baked in
      - We need to raise the bar *higher* in the cloud; status quo is not enough
    - Containers are still often run *within VMs* when security is needed

# Security: 800 Ton Gorilla in the Room

- For too long, Security is the Red-Headed Step Child of Software Architecture
  - We've used bolt-on solutions
  - “The sysadmin will handle that!”
    - Which quickly becomes “It's the sysadmin's fault that they DIDN'T handle that!”
- You only need to glance at the news to know the state of security in our industry today: ***IT SUCKS!***

# Security By Design is a Requirement

- We MUST raise the defacto level of security in our cloud and web applications!
- “Good enough” is no longer good enough
- Some enhanced level of security needs to be baked in so that even a totally incompetent idiot can deploy applications which are more secure than today's status quo

# 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); extremely important to foster adoption
- Many unikernels already exist, and the number is growing steadily

# What is a Unikernel? From MirageOS

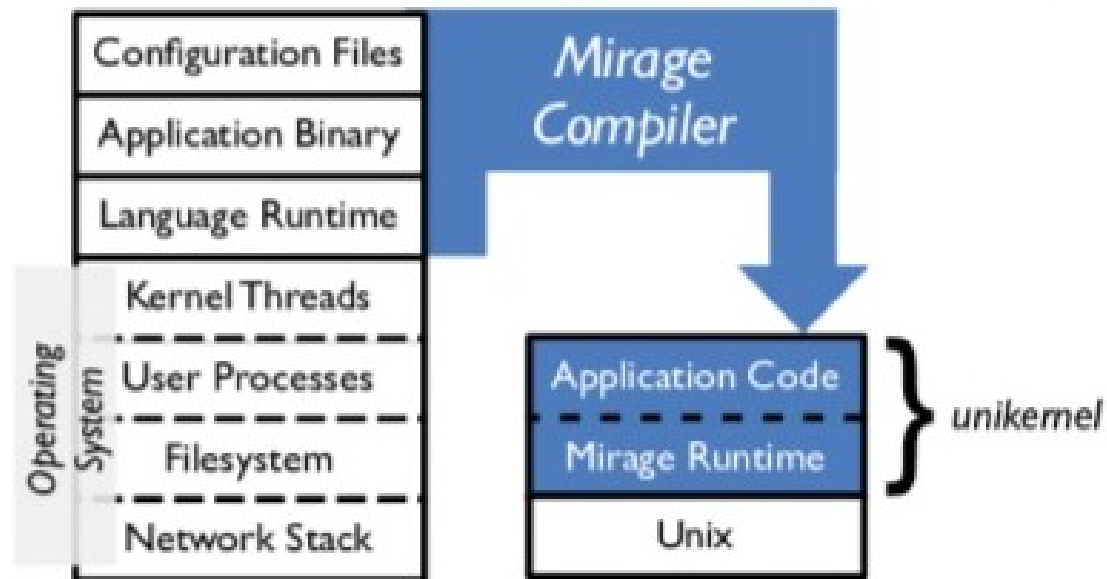


## THE UNIKERNEL APPROACH

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

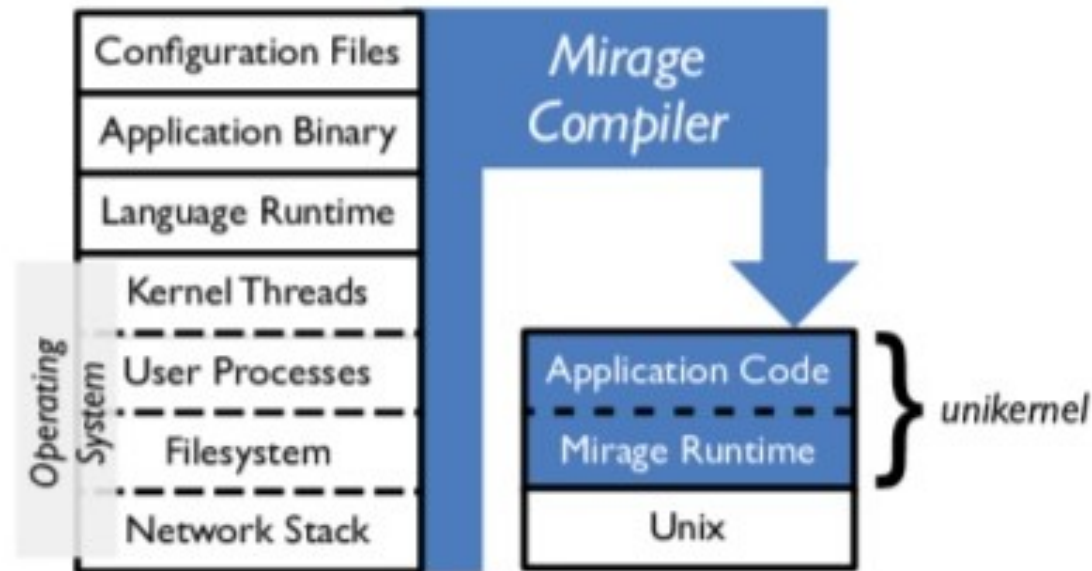
# Unikernel Approach: MirageOS

Swap system libraries to target different platforms:  
**develop application logic using native Unix.**



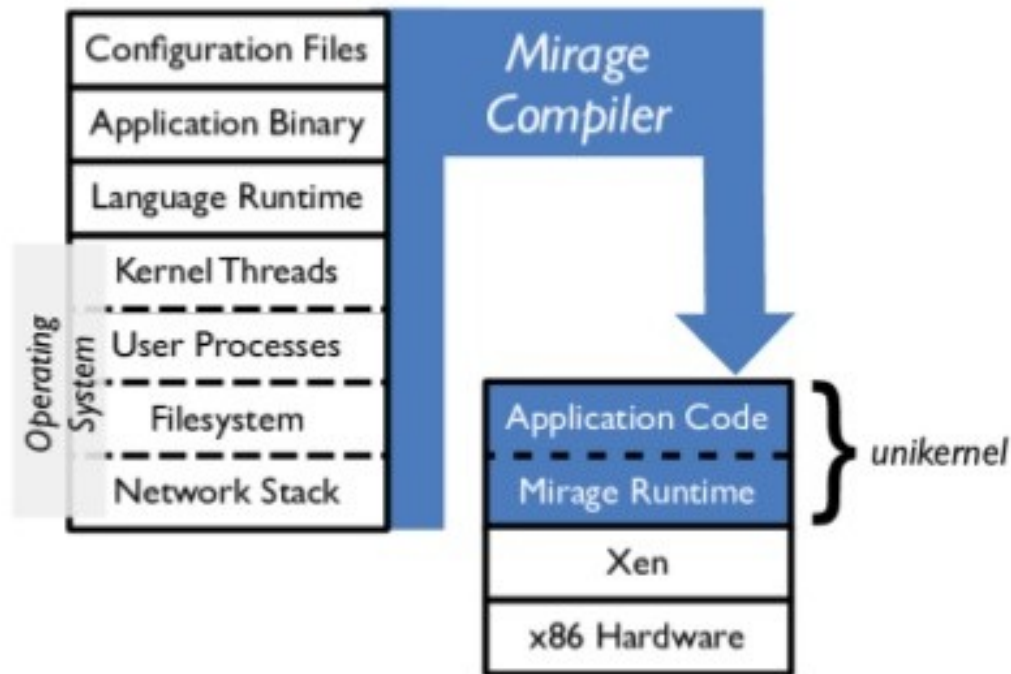
# Unikernel Approach: MirageOS

Swap system libraries to target different platforms:  
**test unikernel using Mirage system libraries.**



# Unikernel Approach: MirageOS

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
    - You have exactly the tools you built into the stack
  - 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
  - Make the attack surface much smaller and much less consistent

# Do Unikernels Really Exist?

Some of the current leading unikernels

# What's Out There Right Now?

- MirageOS, from the Xen Project Incubator
- HaLVM, from Galois
- LING, from Erlang-on-Xen
- ClickOS, from NEC Europe Labs
- OSv, from Cloudeus Systems
- Rumprun, from the Rump Kernel Project
- And that's just the beginning...

# MirageOS

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

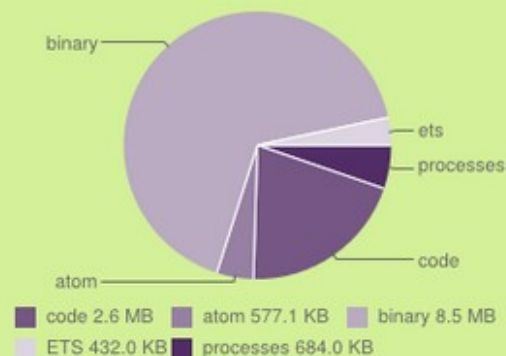
## Processes:

```
<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
```

Memory usage [12.7 MB]



# 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](#)

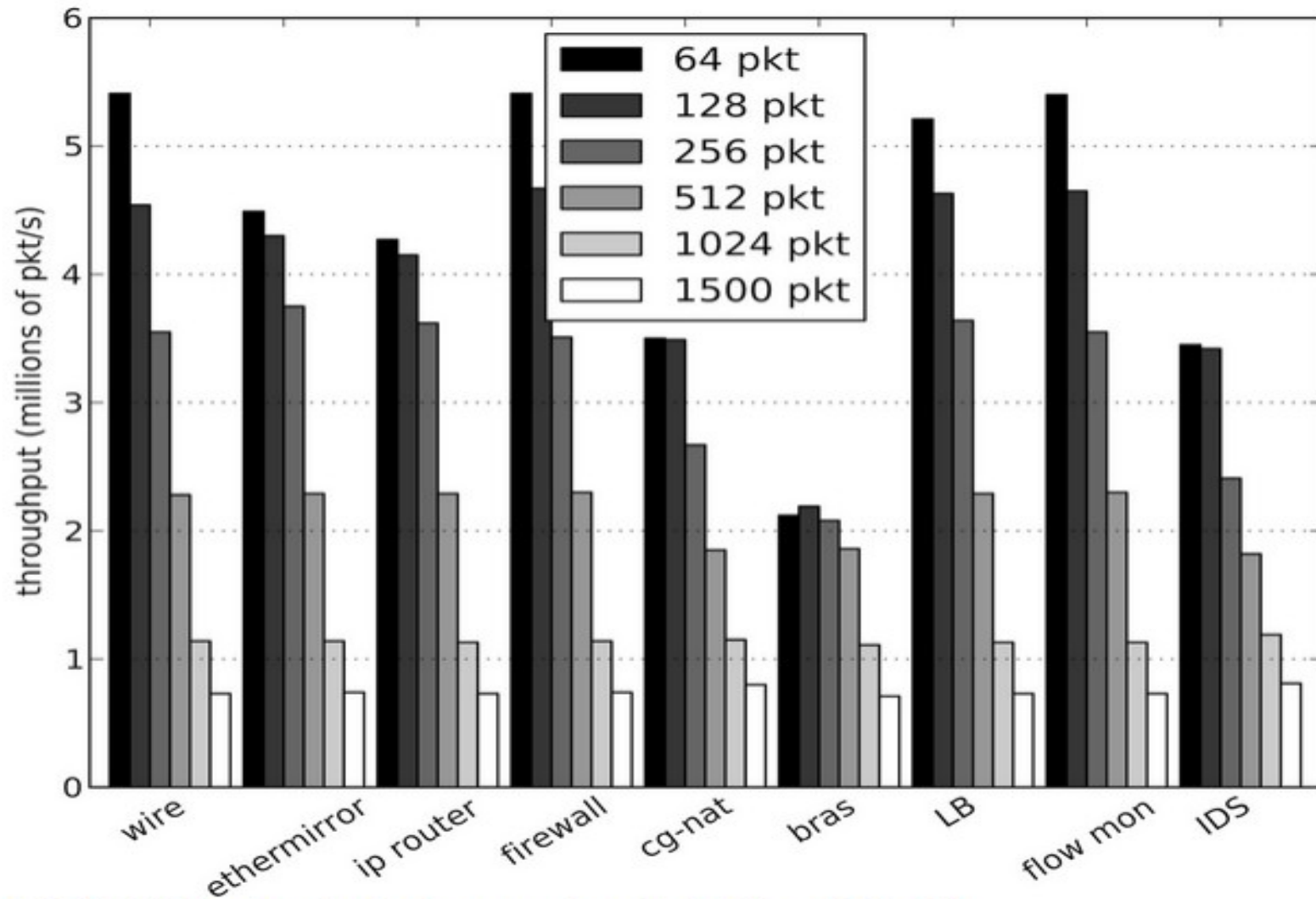
This demo was made possible by [Erlang on Xen](#).



# 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 Throughput



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

# OSv

- Cloudeus Systems (now ScyllaDB)
  - Company may have moved on, but their Open Source project survives
  - Language support: C, C++, Java, Python, Javascript, Node.js, Ruby
- 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/>

# Rumprun

- A working product of the rump kernel ecosystem (which we'll discuss shortly)
- Under active development, rumprun does allow a growing number of programs to run as-is
  - Its goal is to a universal base for most unikernel-appropriate workloads for currently existing real-world POSIX-based applications
  - It has the potential to open the door to a *huge* number of functional unikernels
- <http://repo.rumpkernel.org/rumprun>

# 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>)
  - MiniOS (<http://wiki.xenproject.org/wiki/Mini-OS>)
  - Rump Kernels (<http://rumpkernel.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.*

# MiniOS

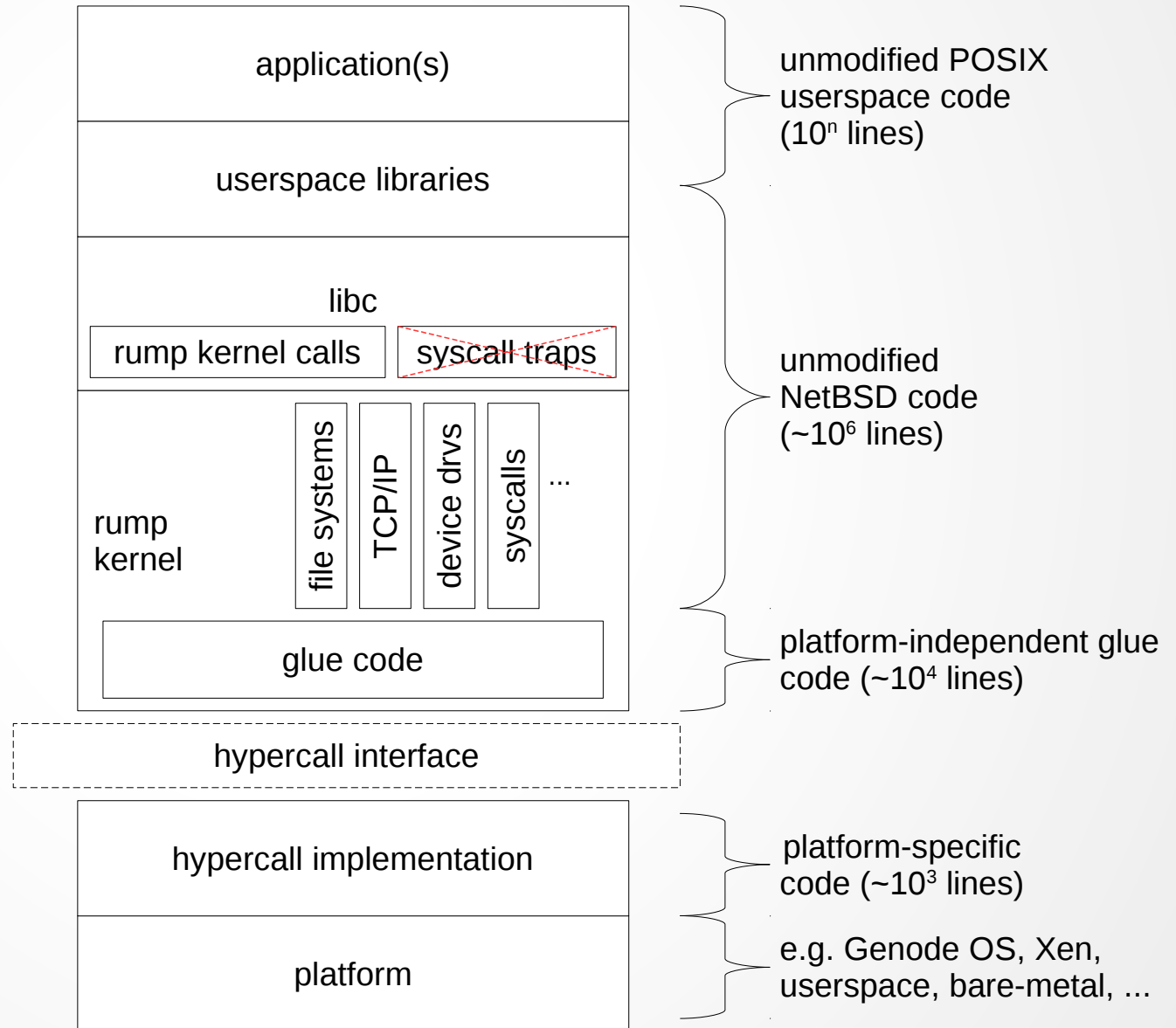
- Small basic unikernel
- Distributed with Xen Project source
- Originally designed for driver disaggregation
- Base for others to build their unikernel projects
  - ClickOS, for example
  - Also the base for the earliest version of rumprun, which has advanced considerably since

# Rump Kernels

- Derived from the work of the NetBSD community
- Employs the notion of a kernel containing just enough code to get real work done
  - Concept is not limited to NetBSD, but existing work leverages NetBSD
- An open-ended framework containing production-quality drivers, currently manifesting itself in the *rumprun* unikernel
- Supports Xen Project, bare metal, userspace environments

# Rump Kernel Architecture

same thread  
throughout entire stack



# THIS JUST IN...

**News Flash:** *The Fat Boy up front was **wrong!***

*You **CAN** do databases as Unikernels!*

# The “RAMP” Stack!

- Major announcement 1 year ago: Nginx, MySQL, and PHP built on Rump Kernels!
- No rearchitecting the application; the work is in getting things to cross compile correctly (Nginx & MySQL)
- Working out usability and configuration kinks still
- Unikernel-compatible unmodified POSIX C and C++ applications “just work” on top of Rump Kernels, provided that they can be cross-compiled
  - Stacks on Rump Kernels are *always* cross-compiled, since the compiler *never* runs directly on the Rump Kernel
- Still in skunkworks stage; watch Twitter @rumpkernel for announcement when it is done

# More Rump Kernel & RAMP Info

- Rump Kernels contain the work of many BSD contributors, all the way back to the 1980s
- Antti Kantee leading the Rump Kernel project
- Martin Lucina leading the RAMP work
- Current Temporary Github repositories (will probably be replaced with a permanent Wiki page):
  - <https://github.com/mato/rump-php>
  - <https://github.com/mato/rump-mysql>
- Rump Kernel Mailing List:
  - <http://www.freelists.org/list/rumpkernel-users>
- Rump Kernel Twitter:
  - @rumpkernel

# 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 600 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 Still More To Come...

- Arrakis (<http://arrakis.cs.washington.edu/>)
  - Derived from the Barrelfish operating system
- Clive (<http://lsub.org/lsub/clive.html>)
  - Using the *go* language
- Solo5 (<https://developer.ibm.com/open/solo5-unikernel/>)
  - A new unikernel base from IBM
- IncludeOS
  - C++ on KVM

# Are Unikernels a Panacea?

- Nope!
  - But it doesn't have to be a panacea to return value
  - There will always be really 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

# What Does This Mean for Architecture?

- We like to talk about (Persistent) Microservices; we are witnessing the birth of *Transient Microservices*
  - Lifetimes possibly measured in fractions of second
  - Populations in the thousands concurrently per host
  - Now these aren't small just from an external standpoint, but internally as well
  - It's much easier manipulating smaller items than bigger ones, so what was once difficult to change becomes easier to change

# How Do You Control a Transient Microservice?

- Excellent question!
- How can we tell if a service was born, executed correctly, and died appropriately?
- Do *you* know the answer?
  - This is where Open Source shines – and why all this is coming from Open Source efforts
  - “Given enough eyeballs, all bugs are shallow” – ESR, CatB
  - Get enough people looking at the problem and the solution will be obvious to someone

# Open Source Leading the Way

- This is an example of how Open Source is working to expand horizons of the cloud
  - Closed source in the next-gen cloud just isn't the way to go
  - The real innovation continues to come from Open Source
  - Friends don't let friends go closed source in the cloud!

# Questions?

rcpavlicek@yahoo.com  
Twitter: @RCPavlicek

*Thanks to the Mirage OS team and Antti Kantee of the Rump Kernel project for the use of their images. Thanks to NEC Europe Ltd (ClickOS) and ErlangOnXen (LING) for the use of images from their respective websites. Rights to same belong to the copyright holders.*

**Look for my upcoming article in BSD Magazine and an upcoming eBook from your favorite publisher!**

*And don't forget my session at 4:35pm here today!*

If anyone is looking for a community guy / evangelist, drop me a line!