It's Time to Debloat the Cloud with Unikraft

Felipe Huici, Simon Kuenzer, Sharan Santhanam – NEC Laboratories Europe GmbH
Alexander Jung – Lancaster University
The Private/Public Cloud

200

Cloud computing with AWS

Amazon Web Services (AWS) is the world’s most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally.

Multitude of services

Bad for the environment
We need to debloat our VMs
We need to stop idle VMs
The Problem Part 1: Size

Wasted Resources

Your virtual machine

Made of...

Your application

Unused

Unused

Unused

$
The Problem Part 2: Time

Wasted Resources

idle active idle active idle

$t=0$
Specialization = High Efficiency
Unikernels = Specialized Virtual Machines

GOALS
• Easy to build and run
• Easy or no app porting
• Great performance
Unikernel Power

- **Fast start/stop/migration times**
  - 10s of milliseconds or less (and as little as 2.3ms)
  - **REFS:** LightVM [Manco SOSP 2017], Jitsu [Madhvapeddy, NSDI 2015]

- **Low memory footprint**
  - Few MBs of RAM or less
  - **REFS:** ClickOS [Martins NSDI 2014], Unikraft [Kuenzer, Eurosys 2021. Best Paper Award]

- **High density**
  - 8k guests on a single x86 server
  - **REFS:** LightVM [Manco SOSP 2017]

- **High Performance**
  - ~300K reqs/sec nginx with a single guest CPU
  - **REFS:** Unikraft [Kuenzer, Eurosys 2021. Best Paper Award], Elastic CDNs [Kuenzer VEE 2017]

- **Security Features**
  - Small trusted compute base
  - Strong isolation by hypervisor
  - Per-library isolation
  - **REFS:** FlexOS [Lefeuvre HotOS 2021], CubicleOS [Sartakov ASPLOS 2021]
Unikernels in One Slide

General Purpose OS (e.g., Linux)

3rd party libraries

OS libraries

OS kernel

Web server

Unikernel

Lots of devel time!
How do we *transparently* build efficient and POSIX-compliant unikernels?
Design Principles

1. Fully **modular** kernel

2. Provide high performance **specialized** APIs
Design Principles

1. Fully **modular** kernel

2. Provide high performance specialized APIs
Why not Linux?

Unikraft is built from scratch to be fully modular
With Unikraft
Doing it with existing unikernels?

(1) Require significant expert work to build

Unikraft is built from scratch (with borrowing)

(3) The (uni)kernels are *still* monolithic
Design Principles

1. Fully modular kernel

2. Provide high performance specialized APIs
Specialized API Example

- Application
- glibc
- POSIX sockets
- Network stack
- UDP
- High-perf API
GOALS

• Easy to build and run
• Easy or no app porting
• Great performance
GOALS

• Easy to build and run
• Easy or no app porting
• Great performance
POSIX Compatibility – Two Approaches

AUTO PORTING

Application Sources (object files, native build)

open sources

musl

Application Sources

link obj files

libc (musl)

BINARY COMPATIBILITY

Application Binary (unmodified ELF)

closed source

ELF reader/loader

trap syscall

syscall shim layer

syscalls

open sources

closed source

musl

link obj files

libc (musl)

ELF reader/loader

syscall shim layer

syscalls
What about syscall support?
Syscall Support

146 syscalls currently supported

Eurosys 2016

146 syscalls currently supported

Linux: ~350 syscalls
Syscall Support
Top 30 Debian Popcon Apps

146 syscalls currently supported
What Unikraft Supports (sample)
GOALS

• Easy to build and run
• Easy or no app porting
• Great performance
Does autoporting sacrifice performance?
Autoporting doesn’t negatively affect performance

time for 60K insertions
Transparent Benefits – Boot, Memory, Size, Throughput
Image Sizes vs. other Projects

- Unikraft
- Hermitux
- Linux User
- Lupine
- Mirage
- OSv
- Rumprun

Image size:
- 0B
- 2MB
- 4MB
- 6MB
- 8MB
- 10MB
Unikraft Boot Times

QEMU
QEMU (1NIC)
QEMU (MicroVM)
Solo5
Firecracker

Total Boot Time (ms)
38.4ms
42.7ms
9.1ms
3.1ms
3.1ms
npx Thro</p>oughput

![Graph](image-url)

- Mirage Solo
- Linux FC
- Lupine FC
- Linux KVM
- Rump KVM
- Docker Native
- Linux Native
- Lupine KVM
- OSv KVM
- Unikraft KVM

Average Throughput (x1000 req/s):

- 25.9
- 60.1
- 71.6
- 104.5
- 152.6
- 160.3
- 175.6
- 189.0
- 232.7
- 291.8
Boot Times - Different Allocators

<table>
<thead>
<tr>
<th>Allocator</th>
<th>Total Boot Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary buddy</td>
<td>3.07</td>
</tr>
<tr>
<td>Mimalloc</td>
<td>0.94</td>
</tr>
<tr>
<td>Bootalloc</td>
<td>0.49</td>
</tr>
<tr>
<td>tinyalloc</td>
<td>0.87</td>
</tr>
<tr>
<td>TLSF</td>
<td>0.51</td>
</tr>
</tbody>
</table>
Redis Throughput
Different Allocators

Aver. Throughput (Million req/s)

<table>
<thead>
<tr>
<th>Allocator</th>
<th>GET</th>
<th>SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mimalloc</td>
<td>2.72</td>
<td>2.22</td>
</tr>
<tr>
<td>TLSF</td>
<td>2.47</td>
<td>1.97</td>
</tr>
<tr>
<td>Binary buddy</td>
<td>2.32</td>
<td>1.89</td>
</tr>
<tr>
<td>tinyalloc</td>
<td>1.01</td>
<td>0.78</td>
</tr>
</tbody>
</table>
Specialized APIs
Filesystem Specialization – SHFS

Unikraft

Average # TSC

Linux

FILE EXISTS

NO FILE
Unikraft on the cloud
Popular Categories

Operating Systems  Security  Networking  Storage

Data Analytics  DevOps  Machine Learning  Data Products

View all categories

Explore solutions by topic area and use case

Business Applications
Find, buy, and deploy SaaS solutions to drive your business forward.
LEARN MORE

Data & analytics
Gain the decision-making power and insights to act in real time.
LEARN MORE

DevOps
Design and deploy reliable, secure modern applications and infrastructure.
LEARN MORE

Infrastructure software
Move, modernize, and manage your IT infrastructure.
LEARN MORE
<table>
<thead>
<tr>
<th>AMI Name</th>
<th>AMI ID</th>
<th>Owner</th>
<th>Visibility</th>
<th>Status</th>
<th>Creation Date</th>
<th>Platform</th>
<th>Root Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>nginx-xenp-busypoll-1606255273</td>
<td>ami-04dbb339b6f603df</td>
<td>88500117030</td>
<td>Private</td>
<td>available</td>
<td>November 24, 2020 at 11:02:31 PM UTC+1</td>
<td>Other Linux</td>
<td>instance-store</td>
</tr>
</tbody>
</table>

Image: ami-04dbb339b6f603df

Details

<table>
<thead>
<tr>
<th>AMI ID</th>
<th>Owner</th>
<th>Status</th>
<th>Creation Date</th>
<th>Architecture</th>
<th>Image Type</th>
<th>Root Device Type</th>
<th>Kernel ID</th>
<th>Block Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami-04dbb339b6f603df</td>
<td>88500117030</td>
<td>available</td>
<td>November 24, 2020 at 11:02:31 PM UTC+1</td>
<td>x86_64</td>
<td>machine</td>
<td>instance-store</td>
<td>aki-93f1e3fc</td>
<td>-</td>
</tr>
</tbody>
</table>
M3.large

2X throughput (reqs/sec)

M3.large

50% more efficient

M3.large

M3.medium
Our bill in dollars:

- $87.70 Linux/Debian
- $43.80 Unikraft

= 50% savings
Xen-based

KVM-based

KVM-based

Hyper-V (upcoming)

Unikraft on GCP

SeaBIOS (version 1.8.2-20190620_183534-google)
Total RAM Size = 0x0000000026000000 = 614 MB
CPUs found: 1    Max CPUs supported: 1
found virtio-scsi at 0x03350000

Booting from Hard Disk 0...

En: Added
En: Interface is up
Welcome to

Listening on port 8888...
Ecosystem - kraft
Define, configure, build and run Unikraft unikernels.
• Easily manage multiple libraries from different sources

• Quickly access updates and change between versions

• Automatically download application source dependencies
Retrieve stable and latest versions with pre-built source files

(no need to compile everything from scratch)
Usage: kraft [OPTIONS] COMMAND [ARGS]...

Options:
--version Show the version and exit.
-C, --no-color Do not use colour in output logs.
-T, --timestamps Show timestamps in output logs.
-Y, --yes Assume yes to any binary prompts.
-v, --verbose Enables verbose mode.
-h, --help Show this message and exit.

Commands:
built   Build the application.
clean   Clean the application.
configure  Configure the application.
fetch   Fetch Library dependencies.
init   Initialize a new unikraft application.
lib   Unikraft library commands.
list   List architectures, platforms, libraries or applications.
package Package unikernel image.
prepare Runs preparations steps on libraries.
push   Push OCI image.
run   Run the application.
up   Configure, build and run an application.

Influential Environmental Variables:
UK_WORKDIR The working directory for all unikraft
source code [default: ~/.unikraft]
UK_ROOT  The directory for unikraft's core source
  code [default: $UK_WORKDIR/unikraft]
UK_LIBS The directory of all the external unikraft
  libraries [default: $UK_WORKDIR/libs]
UK_APPS The directory of all the template applications
  [default: $UK_WORKDIR/apps]
KRAFTRC The location of kraft's preferences file
  [default: ~/.kraftrc]

Help:
For help using this tool, please open an issue on Github:
https://github.com/unikraft/kraft
root@demo:/usr/src/kraft#
Towards Seamless Integration & Deployment

**Kubernetes**
- Deploy extremely efficient Unikraft images seamlessly on your Kubernetes cluster
- **ongoing**

**Prometheus**
- Monitor your Unikraft instances through a standard monitoring platform
- **ongoing**

**BOSH**
- Perform your Unikraft builds directly via your Cloud Foundry infrastructure
- **planned**
Open Source Project
Project History

- **2017**: Internal launch
- **2018**: Public launch
- **2019**: Hypervisor and baremetal support
- **2020**: Polyglot support
- **2021**: Framework integration, Transparent Scalability
Unikraft: Fast, Specialized Unikernels the Easy Way

Simon Kuenzer
NEC Laboratories Europe GmbH

Vlad-Andrei Bădoiu*
University Politechnica of Bucharest

Hugo Lefeuvre*
The University of Manchester

Sharan Santhanam*
NEC Laboratories Europe GmbH

Alexander Jung*
Lancaster University

Gaulthier Gain*
University of Liège

Cyril Soldani*
University of Liège

Costin Lupu
University Politechnica of Bucharest

Stefan Teodorescu
University Politechnica of Bucharest

Costi Răducanu
University Politechnica of Bucharest

Cristian Banu
University Politechnica of Bucharest

Laurent Mathy
University of Liège

Răzvan Deaconescu
University Politechnica of Bucharest

Costin Raiciu
University Politechnica of Bucharest

Felipe Huici
NEC Laboratories Europe GmbH

Abstract

Unikernels are famous for providing excellent performance in terms of boot times, throughput and memory consumption, to name a few metrics. However, they are infamous for making it hard and extremely time consuming to extract such performance, and for needing significant engineering effort in order to port applications to them. We introduce Unikraft, a novel micro-library OS that (1) fully modularizes OS primitives so that it is easy to customize the unikernel 65], or providing efficient container environments [62, 76], to give some examples. Even in the hardware domain, and especially with the demise of Moore’s law, manufacturers are increasingly leaning towards hardware specialization to achieve ever better performance; the machine learning field is a primary exponent of this [30, 32, 34].

In the virtualization domain, unikernels are the golden standard for specialization, showing impressive results in terms of throughput, memory consumption, and boot times,
All our experiments are open and reproducible:

[github.com/unikraft/eurosys21-artifacts](https://github.com/unikraft/eurosys21-artifacts)
High performance POSIX unikernels are now a reality!

Info: https://unikraft.org/
Code: https://github.com/unikraft
Reproduce: https://github.com/unikraft/eurosyst21-artifacts