A Unikernel Toolkit

Simon Kuenzer <simon.kuenzer@neclab.eu>
Lead Maintainer and Senior Researcher
NEC Laboratories Europe GmbH

FOSDEM 2020

This work has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreements no. 675806 ("5G CITY") and no. 825377 ("UNICORE"). This work reflects only the author’s views and the European Commission is not responsible for any use that may be made of the information it contains.
Traditional OS vs. Specialized Unikernel

One application → Flat and single address space
- Concept: Multiple apps => multiple Unikernels, isolated by Hypervisor

Thin kernel layer, only what application needs
- Single monolithic binary that contains OS and application

Further advantages from specialization
- Performance and efficiency; reduced attack vector; small memory footprint
The Potential of Specialization

- Fast instantiation, destruction and migration time
  - 10s of milliseconds or less (and as little as 2.3ms) ([LigthVM](https://sosp2017.com/papers/14-MANCO.pdf), [Jitsu](https://nsdi2015.org/papers/138.pdf))

- Low memory footprint
  - Few MBs of RAM or less ([ClickOS](https://nsdi2014.org/papers/84-MARTINS.pdf))

- High density
  - 8k guests on a single x86 server ([LigthVM](https://sosp2017.com/papers/14-MANCO.pdf))

- High Performance
  - 10-40Gbit/s throughput with a single guest CPU ([ClickOS](https://nsdi2014.org/papers/84-MARTINS.pdf), Elastic CDNs ([Kuenzer](https://www.vee2017.org/))

- Reduced attack surface
  - Small trusted compute base
  - Strong isolation by hypervisor
Application Domains

**Minimal SW Stack**
- Fast boot, migration destroy
- Resource efficient

**Minimal SW Stack**
- Serverless, (Per-customer) vNFs, IoT, MEC, etc.

**Specialization**
- High performance
- Mission critical

**Small code base**
- Low attack surface
- Cheaper verification

- Automotive, (Industrial) IoT, etc.

- Reactive vNFs, Serverless, Lambda functions, IoT, etc.
- NFV, MEC, etc.
The Unikraft Way: Library Pool

- Everything is a (micro-)library
  - Decomposed OS functionality
    - Schedulers, memory allocators, VFS, filesystems
  - Architectures, platform support, drivers
    - Virtualization environments, bare-metal
  - Application interfaces
    - POSIX, Linux system call ABI

- Specialization: Highly configurable
  - Compile-in only features that your application and environment needs

- Most common libraries are in Unikraft repository
- Applications and additional features can be hosted off-tree

- (Micro-)Libraries pool shared across unikernel projects
The Unikraft Way: Building

make-based build system
- Builds each library and links them

KConfig-driven configuration
- Linux style: make menuconfig
- Menu for selecting and configuring libraries
- Save and restore configurations

kraft
- Companion tool
  - Further improves user experience
- Supports:
  - Defining, configuring, building, and running Unikraft unikernel applications
  - `kraft update`
  - `kraft init -a APPNAME`
  - `kraft build`
Community Status and Achievements
Timeline

**Early 2017: NEC-Internal project launch; 0.1**
- Build system
- Initial port from Mini-OS and Solo5/KVM

**Dec/2017: Public Launch; RELEASE-0.2 Titan**
- As Xen Incubator project
- Arm32 Xen, x86 Xen, x86 KVM, x86 Linux
- Binary buddy allocator (heap)
- Cooperative scheduling

**Feb/2019: RELEASE-0.3 Iapetus**
- Arm64 support for KVM
- Networking (uknetdev, lwip, virtio-net)
- Initial VFS with in-RAM filesystem
- newlib

**Feb/2020: RELEASE-0.4 Rhea**
- Support for External platforms, starting with Solo5
- Language support: C++, Python, Go, Lua, JavaScript, WebAssembly, Ruby
- Tracepoint subsystem
- 9pfs filesystem support (Xen, KVM)
- Libraries: musl (initial) intel-intrinsics, libunwind, libuuid, pthread-embedded, compiler-rt, eigen, fp16, fxdiv, pthreadpool, etc.
Contributions by Affiliation (since 0.3)

- University of Liège; 10
- University Politehnica of Bucharest; 506
- Arm; 127
- NEC; 402
- Individual; 2

Signed-off-by’s
Total: 1047

Number of contributors
Total: 28
Ongoing and upcoming Projects
Binary Compatible Unikernels

Even with complete library pool, manual porting is non-trivial

- Existing build system need to be ported or instrumented (e.g., cross-compilation)
- Pre-compiled binaries cannot be executed (e.g., proprietary executables)

ELF binary compatibility, Linux ABI

- Same executable for Linux should run on Unikraft without recompilation
- ELF loader
- System call emulation

[1] Pierre et. al, A Binary-Compatible Unikernel, VEE’19
[2] Kiviti et. al, OSv—Optimizing the Operating System for Virtual Machines, USENIX ATC ’14
Language Runtimes

- Support applications written in higher-level language as Unikernel
  - C++, Rust, Go, Ruby, Javascript (v8), Python, Lua WebAssembly

- Example: WebAssembly
  - Seamless programming experience from browser, to cloud
  - Trying with Mozilla and WebAssembly community

<table>
<thead>
<tr>
<th>Unikernel Image sizes (uncompressed, Xen)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micropython</td>
</tr>
<tr>
<td>Python 2</td>
</tr>
<tr>
<td>Go runtime</td>
</tr>
</tbody>
</table>

Browser ➔ Cloud ➔ IoT
NFV

Virtualized Network Functions
- Package vNF directly as VM with Unikraft
- Remove maintenance effort of hosting OS
- Minimal OS overhead
- Minimal OS noise
- High networking performance & throughput

Click
- Programmable vNF

Intel DPDK
- Dataplane development Kit
- SDK for building high-performance VNFs
- Directly build Unikernel instead of kernel-bypassing application

eBPF
Hardening

Already small attack vector due to specialization

Common attack prevention features need to be implemented\(^1\), for instance:
- ASLR (via boot loader or toolstack)
- Stack canaries
- Page protection bits
- Heap integrity checks

Enable enhanced preventions with lower performance costs in an unikernel
- Make direct use privileged functionality
- E.g., secure memory allocators based on page permissions\(^2\)

---


\(^2\) Oscar: A Practical Page-Permissions-Based Scheme for Thwarting Dangling Pointers [https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/dang](https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/dang)
Application Porting

- Initial set of ported application
  - Typical for cloud deployments
  - Learn about missing components in Unikraft

- Webservers
  - Nginx, lighttpd

- Databases
  - Memcached, Redis, sqlite

- Machine Learning
  - Pytorch

SQLite Unikernel

Image size (uncompressed, Xen) 844 KB

Welcome to

```
Welcome to Iapetus 0.3.1-5fa13f63
-- warning: cannot find home directory; cannot read ~/.slicerc
SQLite version 3.25.2 2018-09-25 19:08:10
Enter "help" for usage hints.
Connected to a transient in-memory database.
Use "open FILENAME" to reopen on a persistent database.
```
Demo Time
Join us!

- Project page
  - [www.unikraft.org](http://www.unikraft.org)

- Documentation
  - [docs.unikraft.org](http://docs.unikraft.org)

- Sources (GIT)
  - [xenbits.xen.org/gitweb/](http://xenbits.xen.org/gitweb/) (Namespace: Unikraft)
  - [github.com/unikraft](http://github.com/unikraft)

- Contributing
  - [minios-devel@lists.xen.org](mailto:minios-devel@lists.xen.org) (Shared mailing list)
  - [https://patchwork.unikraft.org](http://https://patchwork.unikraft.org)

- IRC Channel on Freenode
  - #unikraft
Orchestrating a brighter world

NEC