Towards Highly Specialized, POSIX-Compliant Software Stacks with Unikraft

Felipe Huici, NEC Laboratories Europe
## The Difficult Choice

<table>
<thead>
<tr>
<th>General Purpose OSes</th>
<th>Embedded OSes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support lots of devs and applications</td>
<td>Resource efficient</td>
</tr>
<tr>
<td>Familiar development environment</td>
<td>Smaller attack surface</td>
</tr>
<tr>
<td>POSIX compliant</td>
<td></td>
</tr>
<tr>
<td>Hard to customize</td>
<td></td>
</tr>
<tr>
<td>Large attack surface</td>
<td></td>
</tr>
<tr>
<td>Large memory footprint, image size, memory consumption</td>
<td>Expensive to certify</td>
</tr>
<tr>
<td>Expensive to certify</td>
<td>Unfamiliar development environment</td>
</tr>
</tbody>
</table>

**Is it possible to have the best from both worlds?**
A New OS Paradigm - Requirements

- Easy configuration and customization
- Efficient
- Minimal attack surface
- Low certification costs
- Easy to port/use existing application
Unikraft’s Optimization

Diagram showing the optimization of a system from using third-party libraries, OS libraries, and OS kernel to a more optimized version using Unikraft.
Unikraft is a highly **modular** library pool and build system allowing users to **seamlessly** build extremely specialized and efficient images (VMs, containers, bare metal) targeting particular applications.
A Fully Librarized OS

**Main Unikraft Library Pool**
- Network Stack
  - `liblwip.o`
  - `libdpdk.o`
- Virtual File Systems
  - `libvfs.o`
  - `lib9pfs.o`
- Drivers
  - `libvirtio.o`
  - `libmmio.o`
- Memory Allocators
  - `libtlslf.o`
  - `libmimalloc.o`

**External Library Pool**
- Standard Libraries
  - `libmusl.o`
  - `libnewlib.o`
- Runtimes
  - `libpython.o`
  - `libgolang.o`

**Platform Library Pool**
- `libraspi.o`
- `libultra96t.o`
- `libkvm.o`

**Architecture Library Pool**
- `libx86_64.o`
- `libarm64.o`
- `libarm32.o`

**Build**

**Binaries**
- `raspiarm64`
- `ultra96arm64`
- `kvmx86_64`

---

- **OS primitives as libraries**
  - Schedulers
  - Memory allocators
  - Network stack
- **Customizable via Kconfig**
- **Language support**
  - Compiled: C/C++, Go, Java, Rust
  - Interpreted: Python, Ruby, Lua, WASM
- **Targets POSIX compatibility**
  - Tools for easy porting, musl support
  - Standard app support (e.g., NGINX, SQLite, Redis)
- **Minimal TCB**
- **Multi-platform support (bare metal, virtual machine)**
Unikraft Security Features

- Stack protection
- Page protection
- ASLR
- Security hardening via fuzzing
- Secure memory allocator
- Immutable images
- Minimal attack surface
- Strong isolation (in case of VMs, containers)
Unikraft on the RaspberryPi

Image Size: 1.36927 GB
Memory Use: 110.559 MB
Boot Time: 15.9646 s (+2.99 s GPU boot)

Boot Time: 5472 us (+2.90 s GPU boot)
Code Size: 29 kB (+1668 kB LCD images)
Memory Use: 1104 B (+ Code size)
IRQ delay: 404 cycles
Performance Results
Power Consumption

<table>
<thead>
<tr>
<th></th>
<th>watts</th>
<th>amps</th>
<th>idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpine Linux</td>
<td>2.03</td>
<td>1.446</td>
<td>0.276</td>
</tr>
<tr>
<td>Raspbian (Offshelf)</td>
<td>1.811</td>
<td>0.388</td>
<td>0.255</td>
</tr>
<tr>
<td>Raspbian (Specialized)</td>
<td>1.872</td>
<td>0.359</td>
<td>0.22</td>
</tr>
<tr>
<td>Rhea helloworld</td>
<td>1.767</td>
<td>0.338</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.512</td>
<td>0.113</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.592</td>
<td>0.592</td>
</tr>
</tbody>
</table>
Memory Usage

The bar chart compares the memory usage of different applications under Linux and Unikraft. The applications include Alpine Linux, Raspbian, Raspbian (Specialized), and helloworld under Linux, and Lua, NGINX, NNpack, and SQLite under Unikraft.

- **Linux**:
  - Alpine Linux: 26.8MB
  - Raspbian: 35.5MB
  - Raspbian (Specialized): 29.8MB
  - helloworld: 289.9KB
  - Lua: 289.9KB
  - NGINX: 419.6KB
  - NNpack: 289.9KB
  - SQLite: 305.2KB

- **Unikraft**:
  - Alpine Linux: 26.8MB
  - Raspbian: 35.5MB
  - Raspbian (Specialized): 29.8MB
  - helloworld: 289.9KB
  - Lua: 289.9KB
  - NGINX: 419.6KB
  - NNpack: 289.9KB
  - SQLite: 305.2KB
Tools & Ecosystem
Unikraft is now even easier to use!

**kraft**: companion tool!
- Improves user & developer experience
- Lists and clones available Unikraft libraries from GitHub organization
- Building and initial configuration
- Testing and Benchmarking

- Get [https://github.com/unikraft/kraft/](https://github.com/unikraft/kraft/)
- ..and start building your Unikernel:

```
> kraft update
> kraft list
> kraft init -a APPNAME
> kraft build
> kraft run -p kvm
```
Future Work and Challenges (Bare Metal)

Drivers support
- Provide compatibility layer to be able to re-use existing drivers (e.g., from FreeRTOS)

Device Support
- RPI and Xilinx/Avnet Ultra96-v2 supported
- What other devices should we add? Possibility: support 96 boards
Website

http://unikraft.org

- Blog
- Getting started
- Pointers to repositories
- Pointer to documentation
Join us!

- Project page
  - www.unikraft.org

- Documentation
  - docs.unikraft.org

- Sources
  - github.com/unikraft

- Kraft tool
  - github.com/unikraft/kraft

- Contributing
  - minios-devel@lists.xen.org