Unikernels & Memory Deduplication

Gaulthier Gain, Laurent Mathy

gaulthier.gain@uliege.be, laurent.mathy@uliege.be

Montefiore Institute – University of Liège – Belgium

Introduction

The software world faces a dilemma:

Virtual Machines	Containers
✓ Strong isolation	✓ Lightweight
✗ Heavyweight	✗ Poor isolation

Solution → **Unikernels**

Unikernels are specialised, lightweight virtual machines built using microlibs which are libraries that include only the essential components.

Memory Deduplication

1. Memory deduplication is a technique to reduce memory consumption by merging identical pages to a same frame.
2. In Linux, the Kernel Samepage Merging (KSM)[1] runs in the background to scan and merge identical pages.

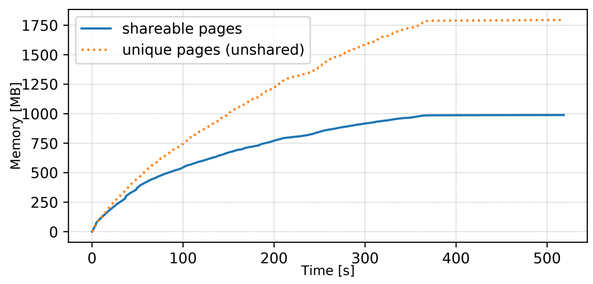

Figure: Evolution of unshared and shareable pages when running 1000 different FaaS unikernels. Unique pages are much more frequent than shared ones.

Unikernels issues

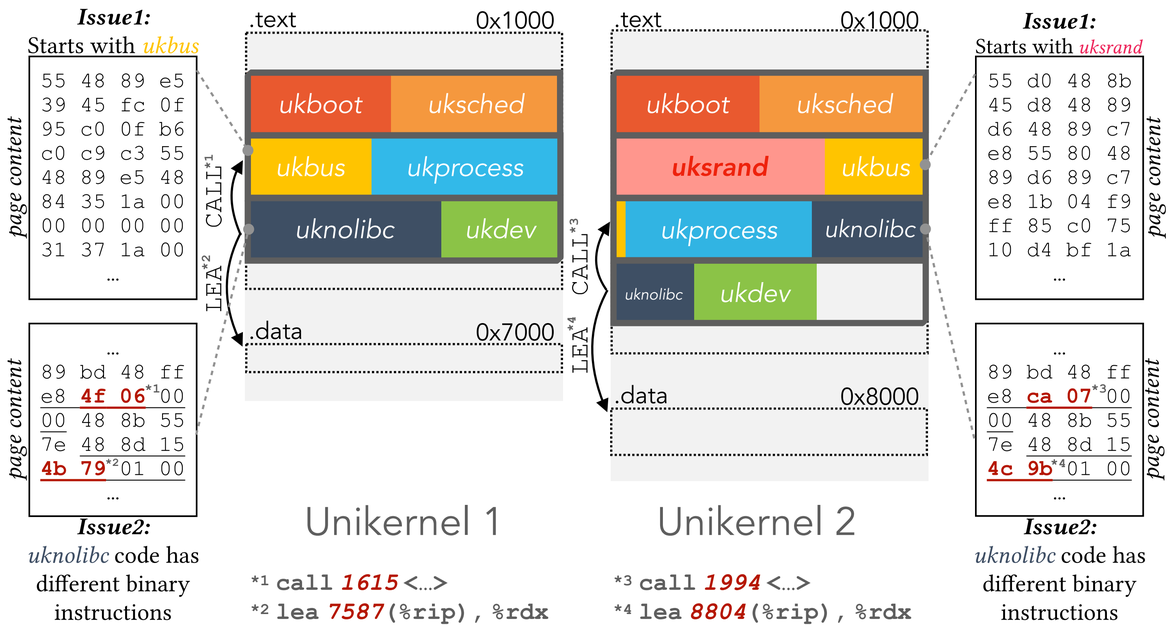

Figure: Unikernels are too compact. The insertion of library `uksrand` (a) shifts following libraries in memory; (b) makes the cross-reference addresses different in instructions such as `call` or `lea`. Both mechanisms concur to prevent memory deduplication.

Solution

We introduce a new methodology based on page alignment[2]:

1. Aligning sections and libraries at the same absolute addresses in the virtual address space of all instances.
2. Keeping a global map of all libraries used and mapping them to a specific address between instances.

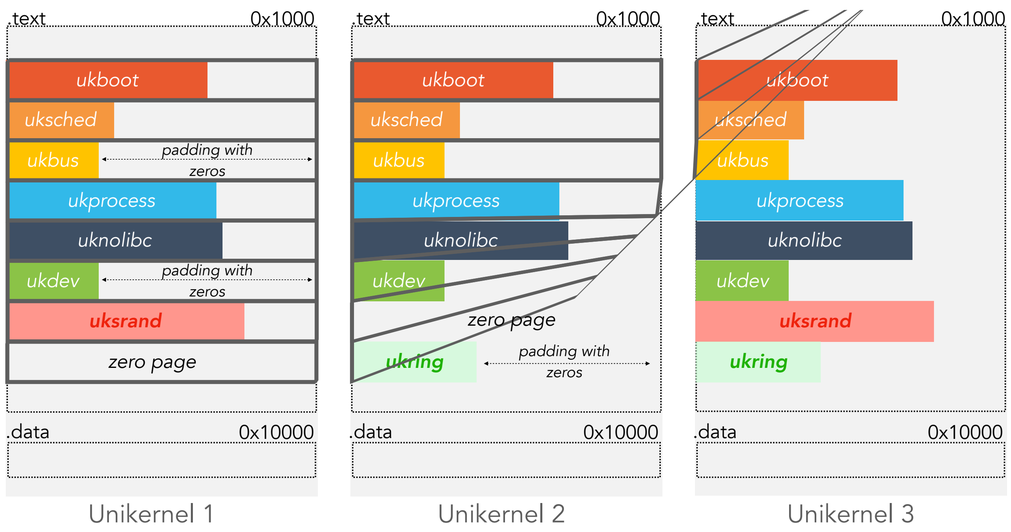

Figure: Aligning libraries across several instances.

[1] https://docs.kernel.org/admin-guide/mm/ksm
[2] https://people.montefiore.uliege.be/gain/spacer
[3] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options
[4] https://firecracker-microvm.github.io

Towards ASLR support

Using fixed absolute addresses leads to security issues (no ALSR).

Solution: Create a trampoline table per library which contains problematic instruction. Such instructions are replaced by a relative call/jump to their new position.

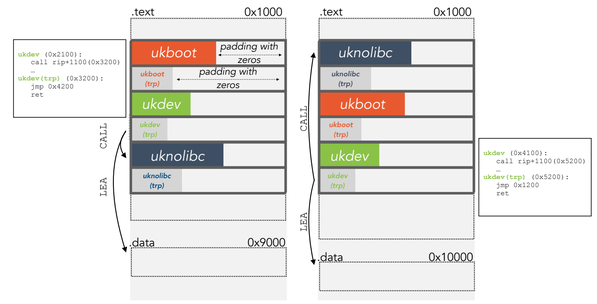

Figure: To support ASLR while sharing code pages, we add one instance-specific trampoline table per library.

A first improvement?

- Aligning unikernel libraries may lead up to a **3x** reduction in memory consumption, even when compared to unikernels built with DCE[3].
- Furthermore, the alignment **does not** introduce significant overhead in terms of ELF size, nor does it impairs application performance.
- Can we do better?

→ Yes, but we must get rid of KSM.

Memory deduplication at load-time

Merging memory at load-time requires (1) a toolchain, (2) a pool of libraries and (3) a custom loader.

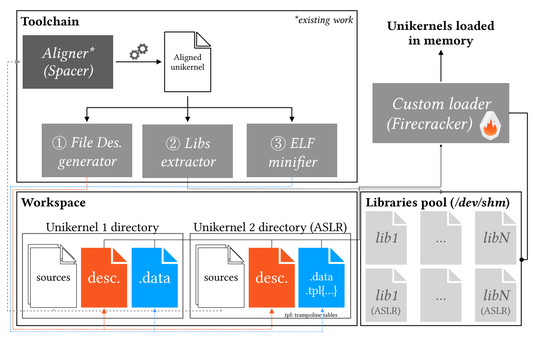

Figure: High-level overview of our architecture.

- It minimises unikernels by extracting their libraries and placing them in a library pool.
- A description file with libraries and application data is used as input to a custom loader that runs multiple unikernels, sharing common libraries.

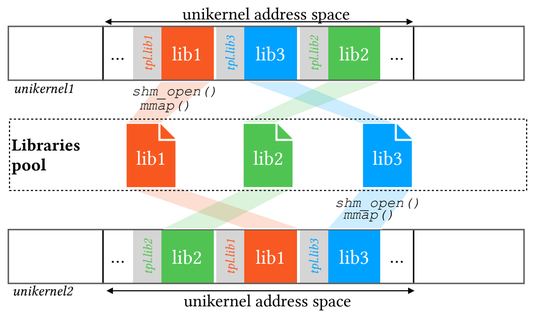

Figure: Libraries are shared via a pool of libraries.

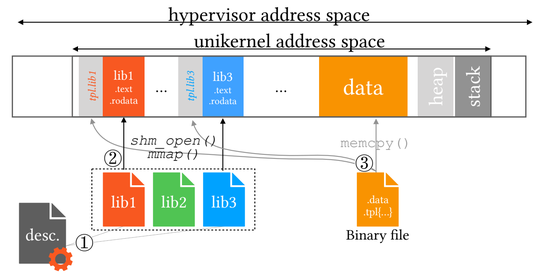

Figure: Our custom loader based on Firecracker[4] performs a well-defined set of operations.

Evaluation

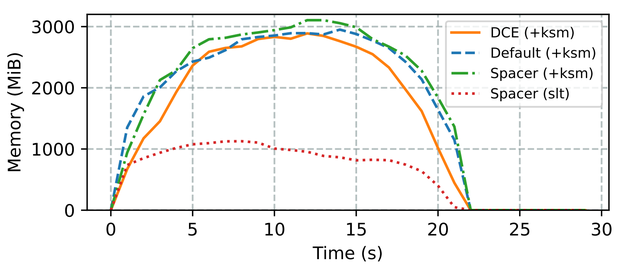

Figure: Evolution of the physical memory used by 1000 FaaS unikernels (+hypervisor) being benchmarked. Our approach achieves the lowest memory consumption.

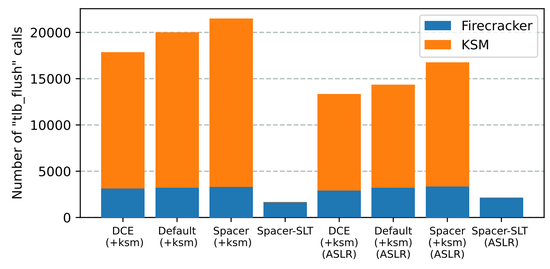

Figure: Number of TLB flushes issued by 20 heterogeneous unikernels (monitored at hypervisor and KSM level).

Conclusion

- Although unikernels are small and have impressive performance measurements, they show few opportunities for VM page sharing.
- We brought a new methodology that rearranges and inflates unikernel images memory layout by using libraries alignment.
- We also designed a unikernel loader capable of performing read-only memory deduplication at load time.