Unikraft: Fast, Specialized Unikernels the Easy Way

Simon Kuenzer, Vlad-Andrei Bădoiu, Hugo Lefeuvre, Sharan Santhanam, Alexander Jung, Gaulthier Gain, Cyril Soldani, Costin Lupu, Ştefan Teodorescu, Costi Răducanu, Cristian Banu, Laurent Mathy, Răzvan Deaconescu, Costin Raiciu, and Felipe Huici

EuroSys'21

August 5th, 2022

Presented by Yejin Han

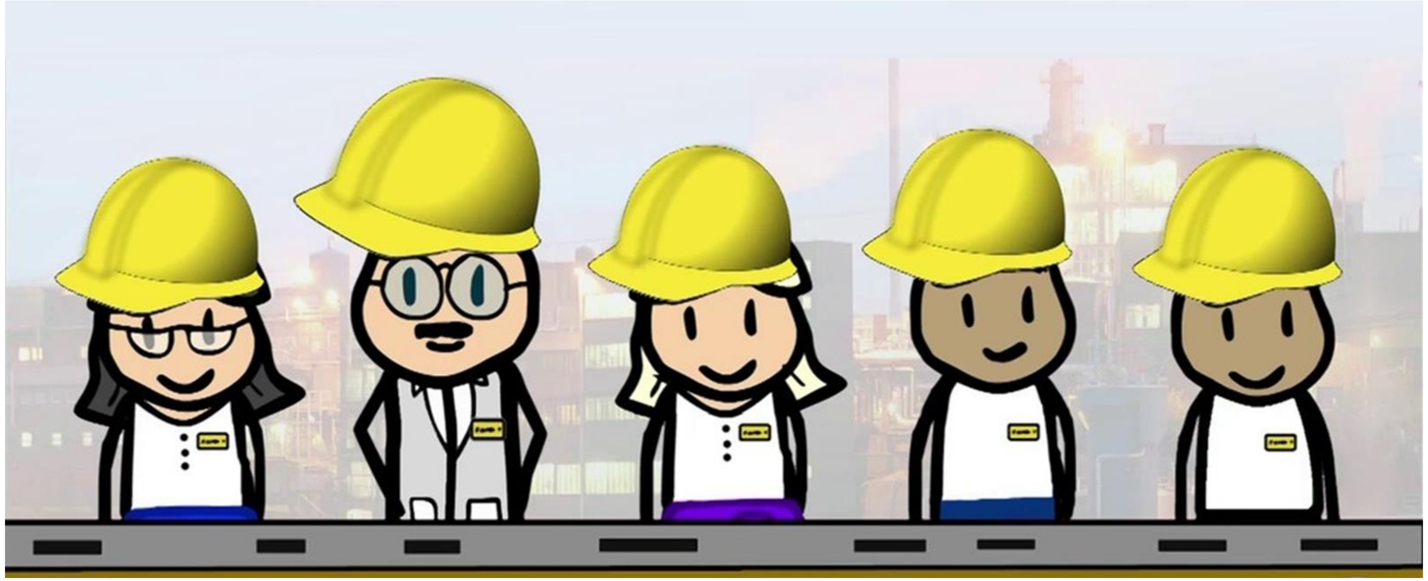
yj0225@dankook.ac.kr

DANKOOK UNIVERSITY Syracuse University

Dankook University System Software Laboratory

Introduction

Specialization = High performance

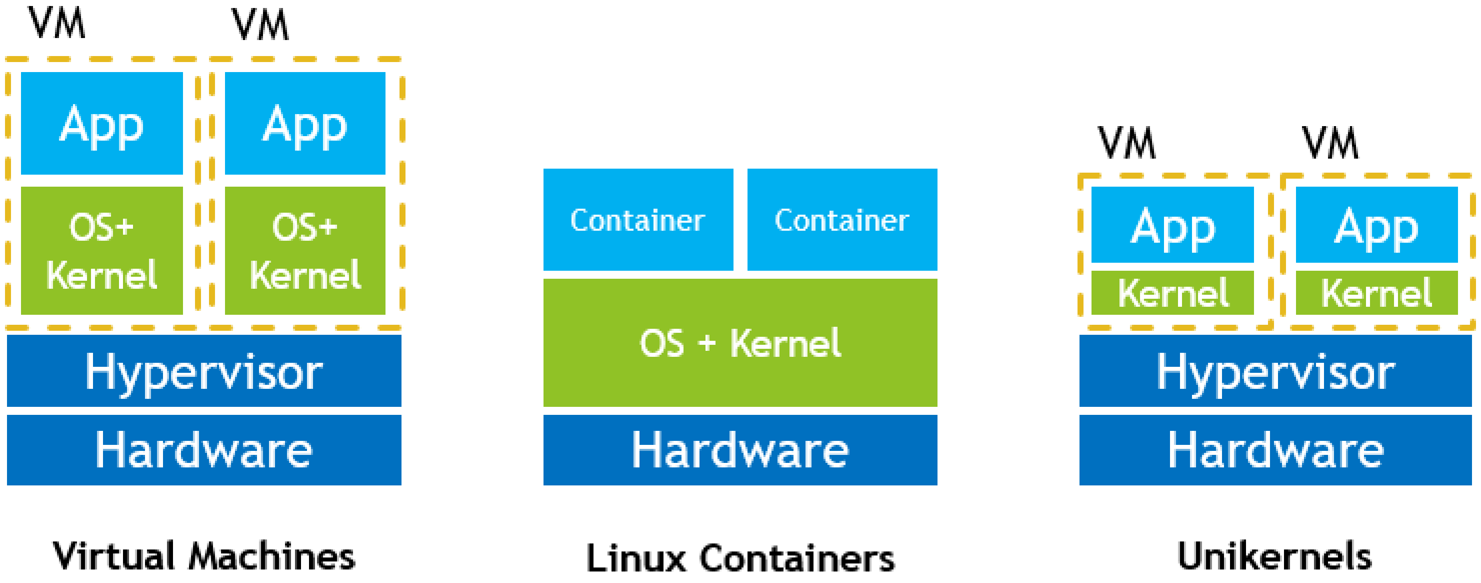


OCaml JS DPD intel Movidius BlueField Google Tensor Processing Unit

(Source: https://study.com/academy/lesson/work-specialization-in-organizations.html)

What is Unikernel?

- Unikernels are specialized virtual machines
  - Easy to build and run
  - Easy or no app porting
  - Great performance



(Source: https://github.com/cetic/unikernels)

Linux vs. Unikraft

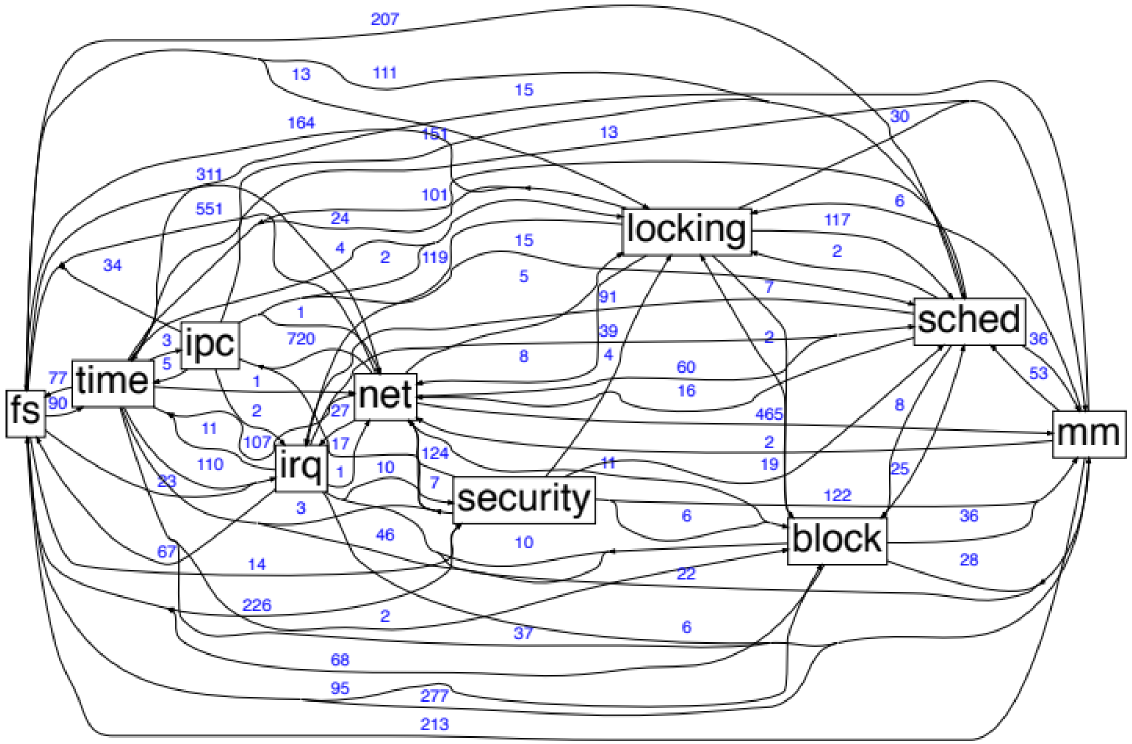


Figure 1. Linux kernel components have strong inter-dependencies, making it difficult to remove or replace them.

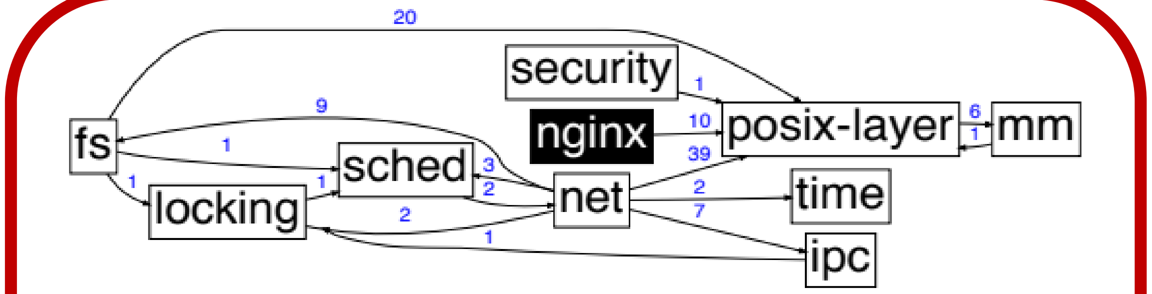


Figure 2. Nginx Unikraft dependency graph

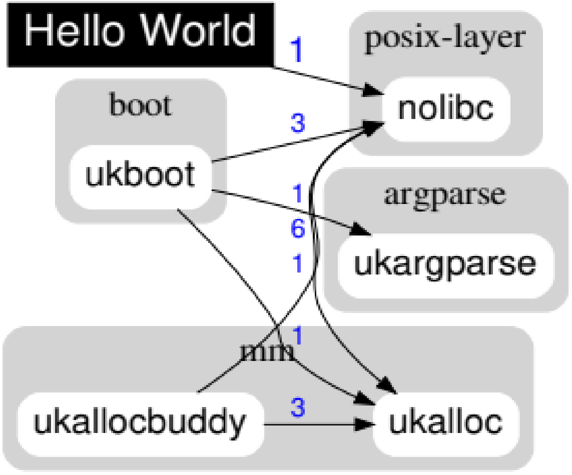


Figure 3. Helloworld Unikraft dependency graph

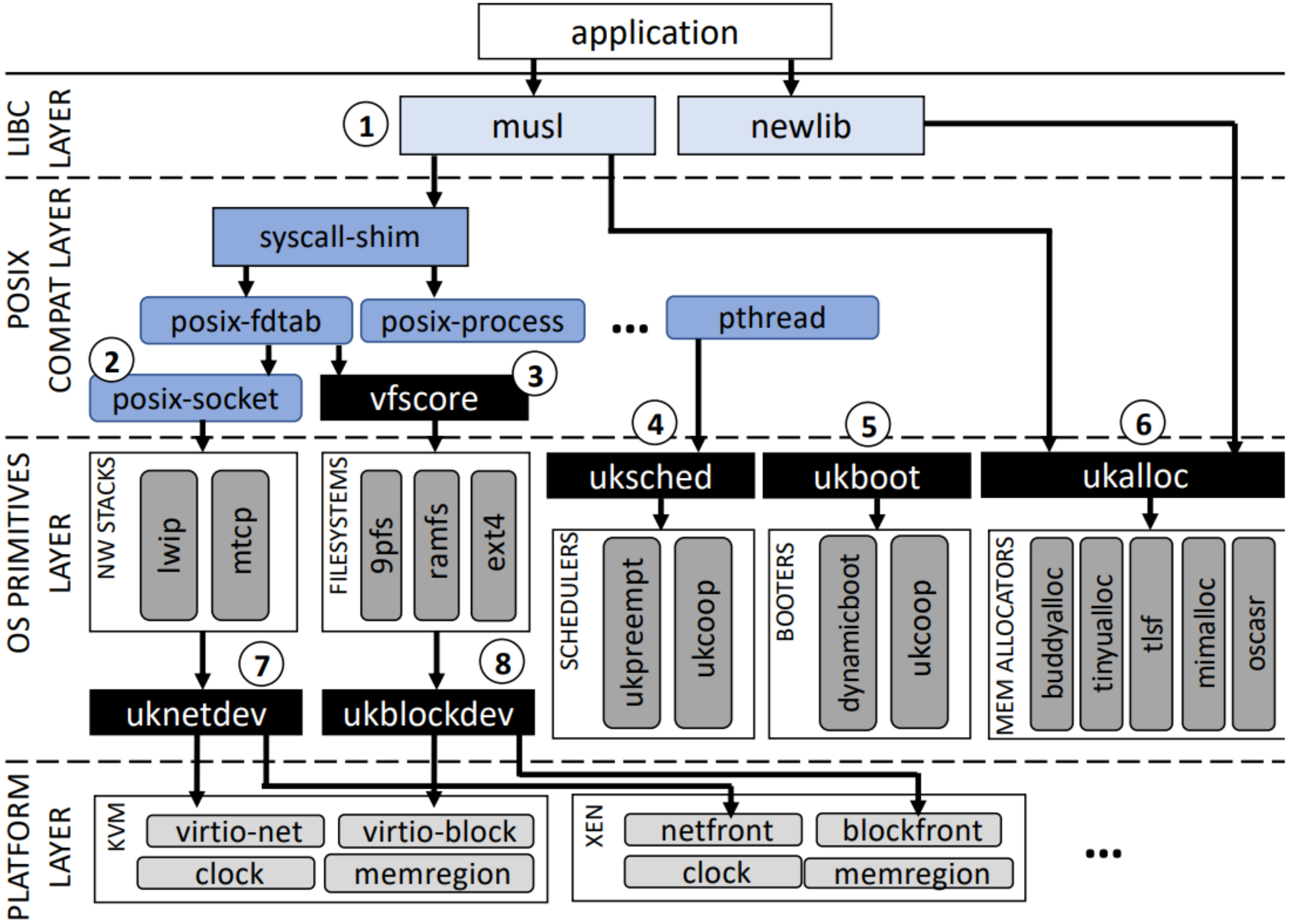
How about existing unikernels?

- Require significant expert work to build
- They are often non-POSIX compliant
- The (uni)kernels are still monolitic

unikernel

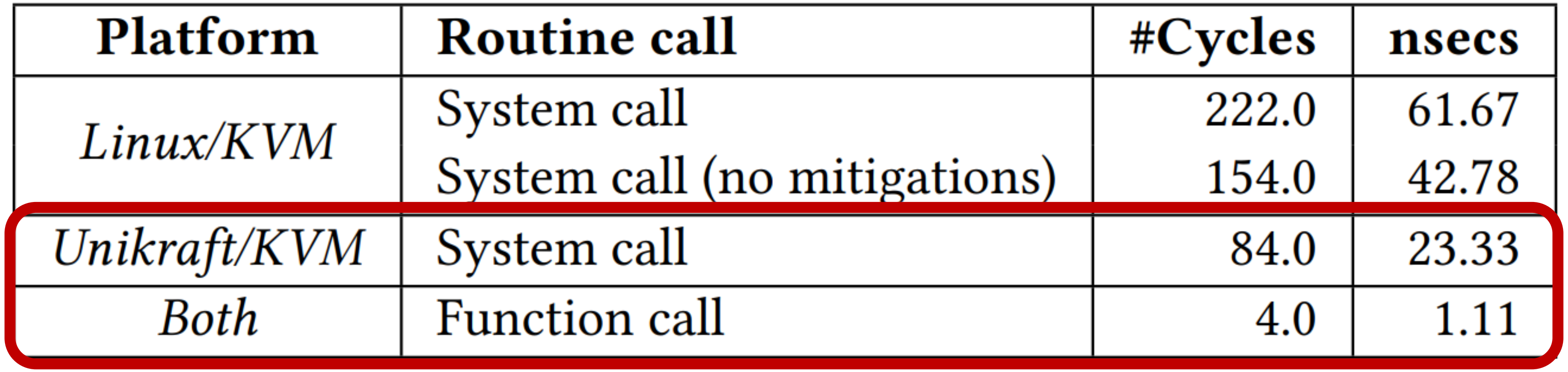
Unikernel + craft =

Unikraft architecture



Application Support and Porting

- Binary Compatibility
  - System calls have a significant performance cost compared to function calls



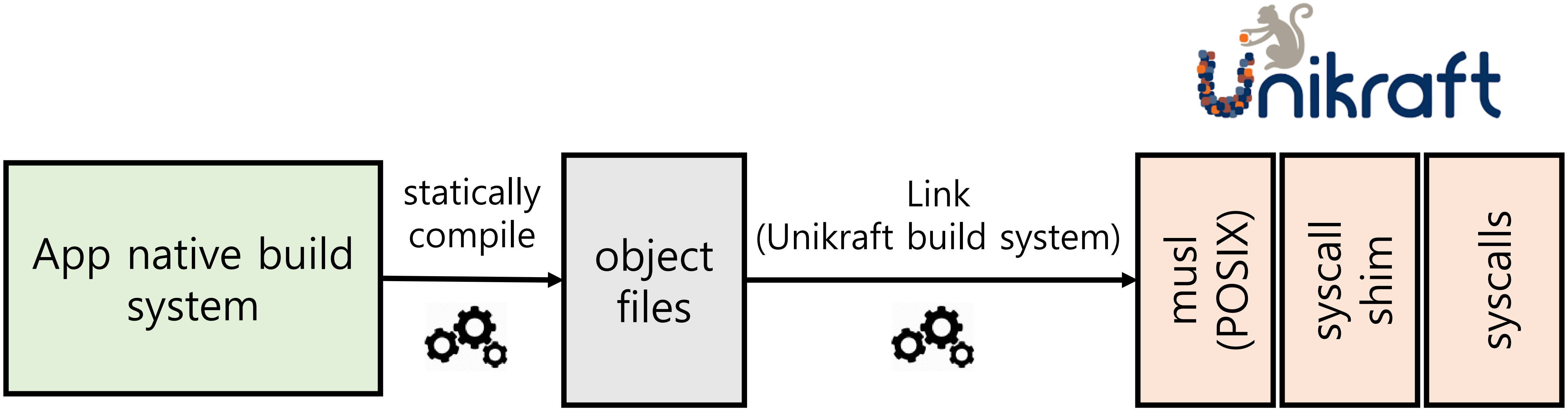
Platform	Routine call	#Cycles	nsecs
Linux/KVM	System call	222.0	61.67
	System call (no mitigations)	154.0	42.78
Unikraft/KVM	System call	84.0	23.33
Both	Function call	4.0	1.11

Table 1. Cost of binary compatibility/syscalls with and without security mitigations.

Binary compatibility is expensive!!

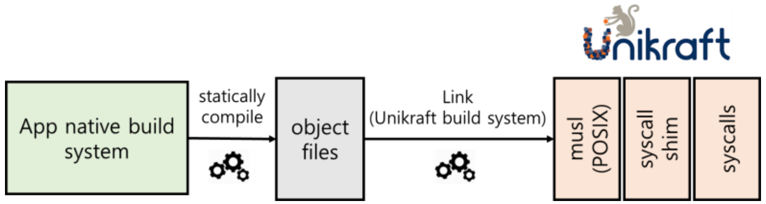
Application Support and Porting

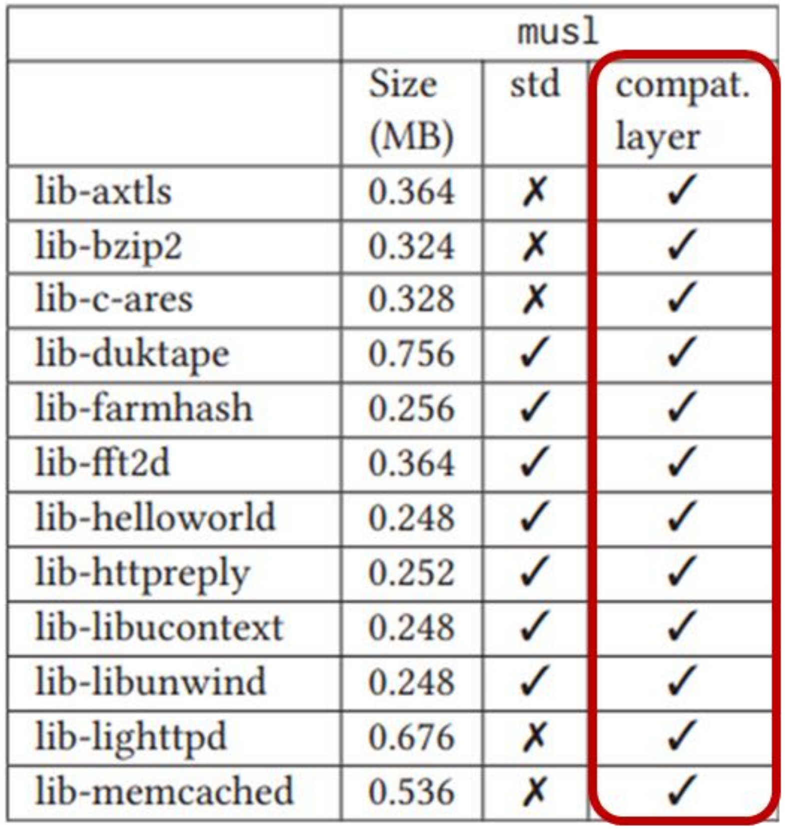
- Auto-porting from Source
  - Rely on the target application's native build system



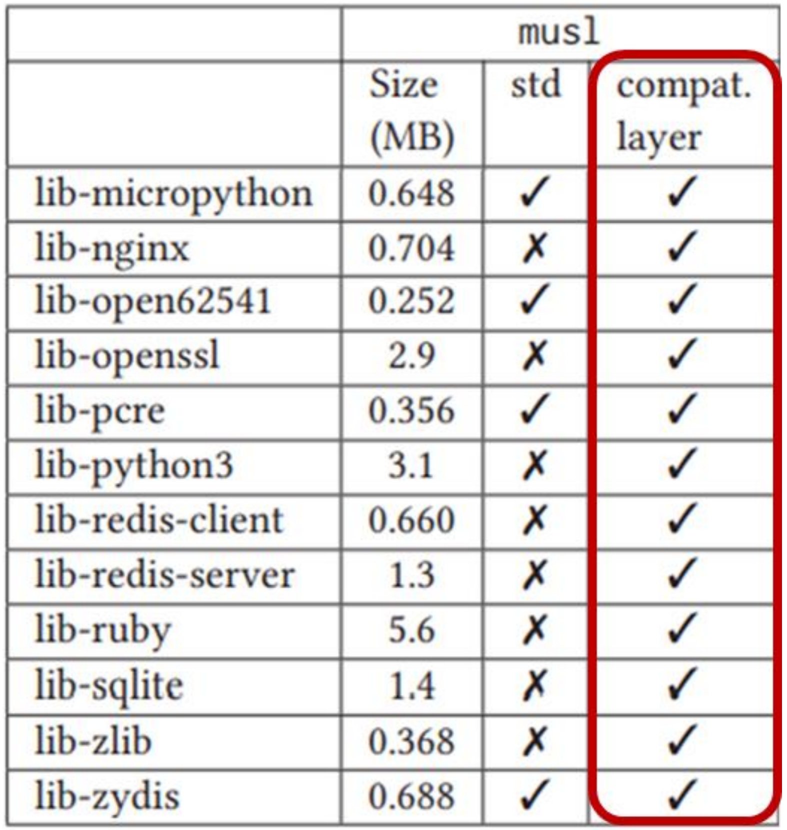
Application Support and Porting

- Auto-porting from Source





	musl		
	Size (MB)	std	compat. layer
lib-axtls	0.364	✗	✓
lib-bzip2	0.324	✗	✓
lib-c-ares	0.328	✗	✓
lib-duktape	0.756	✓	✓
lib-farmhash	0.256	✓	✓
lib-fft2d	0.364	✓	✓
lib-helloworld	0.248	✓	✓
lib-httpreply	0.252	✓	✓
lib-libucontext	0.248	✓	✓
lib-libunwind	0.248	✓	✓
lib-lighttpd	0.676	✗	✓
lib-memcached	0.536	✗	✓



	musl		
	Size (MB)	std	compat. layer
lib-micropython	0.648	✓	✓
lib-nginx	0.704	✗	✓
lib-open62541	0.252	✓	✓
lib-openssl	2.9	✗	✓
lib-pcre	0.356	✓	✓
lib-python3	3.1	✗	✓
lib-redis-client	0.660	✗	✓
lib-redis-server	1.3	✗	✓
lib-ruby	5.6	✗	✓
lib-sqlite	1.4	✗	✓
lib-zlib	0.368	✗	✓
lib-zydis	0.688	✓	✓

Syscall Support

- If all else fails, manual porting is another option

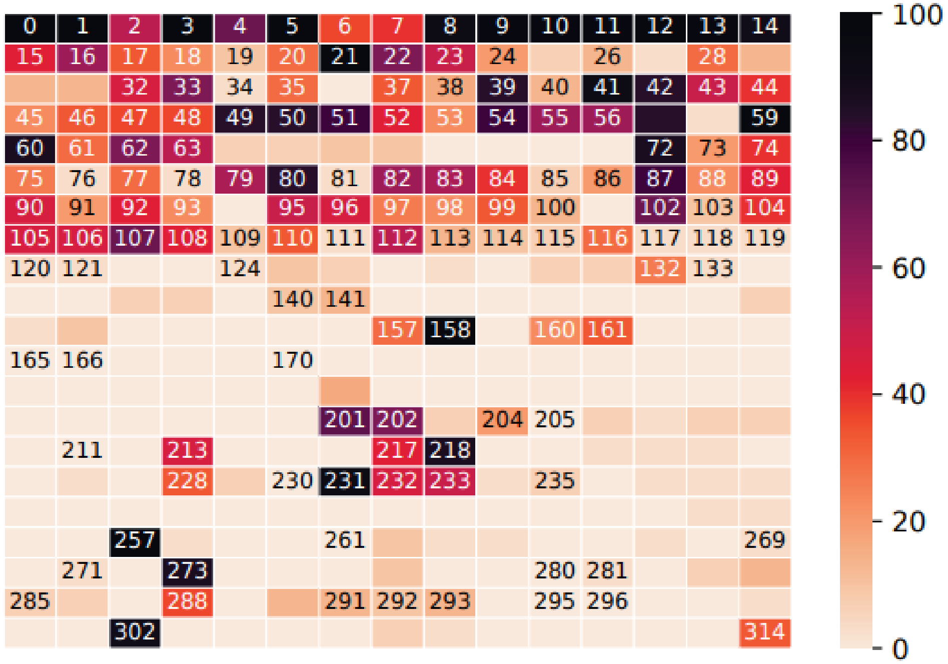


Figure 5. Syscalls required by 30 server apps vs syscalls supported by Unikraft.

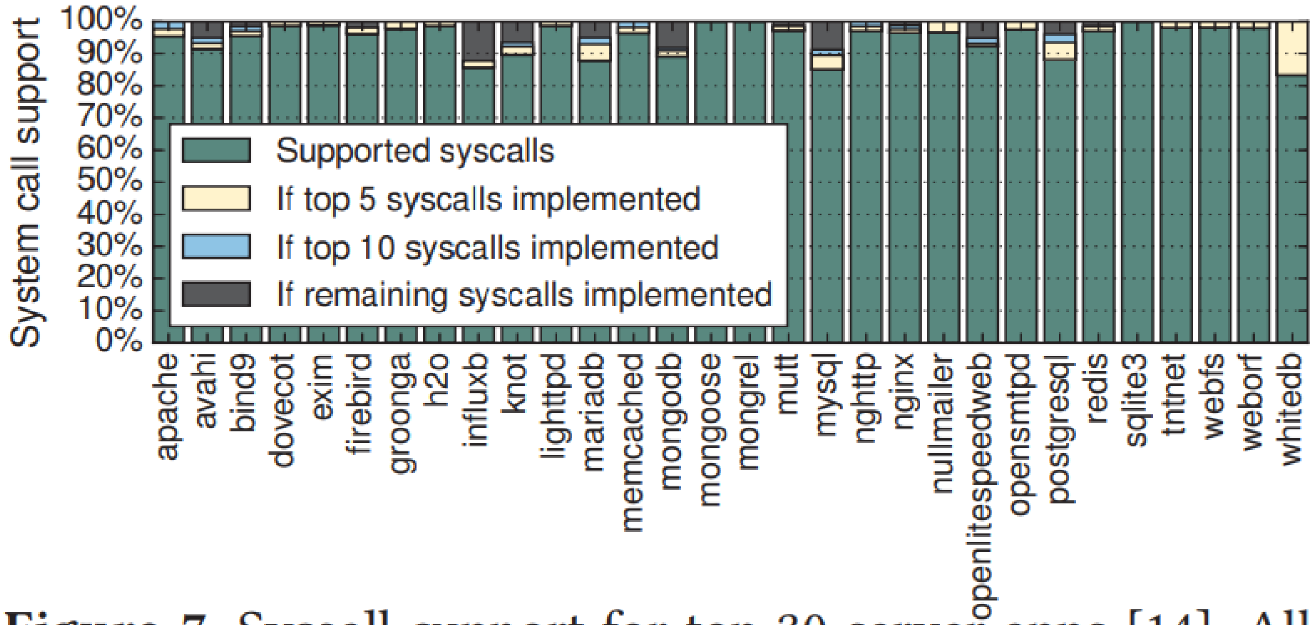


Figure 7. Syscall support for top 30 server apps [14]. All apps are close to being supported, and several already work even if some syscalls are stubbed (SQLite, nginx).

Manual Porting

- If all else fails, manual porting is another option

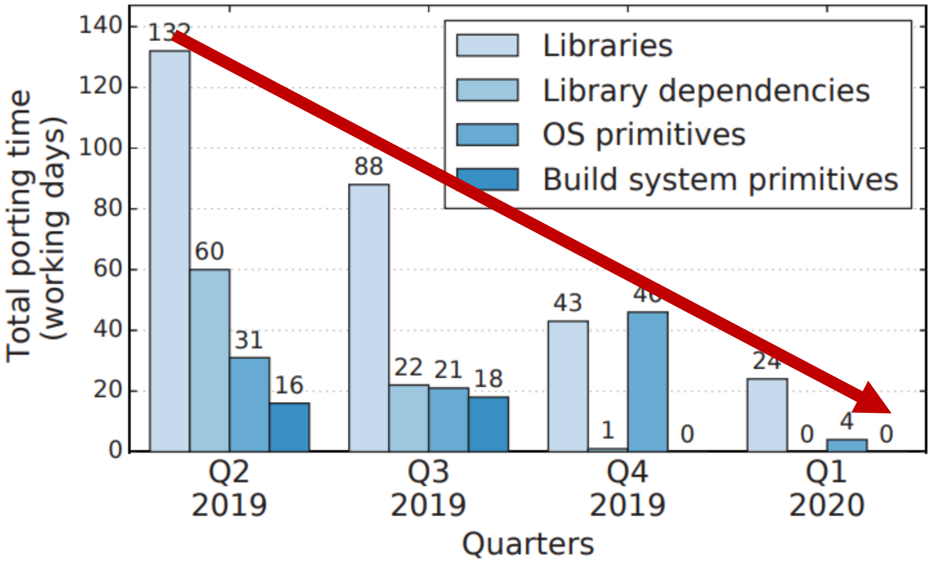
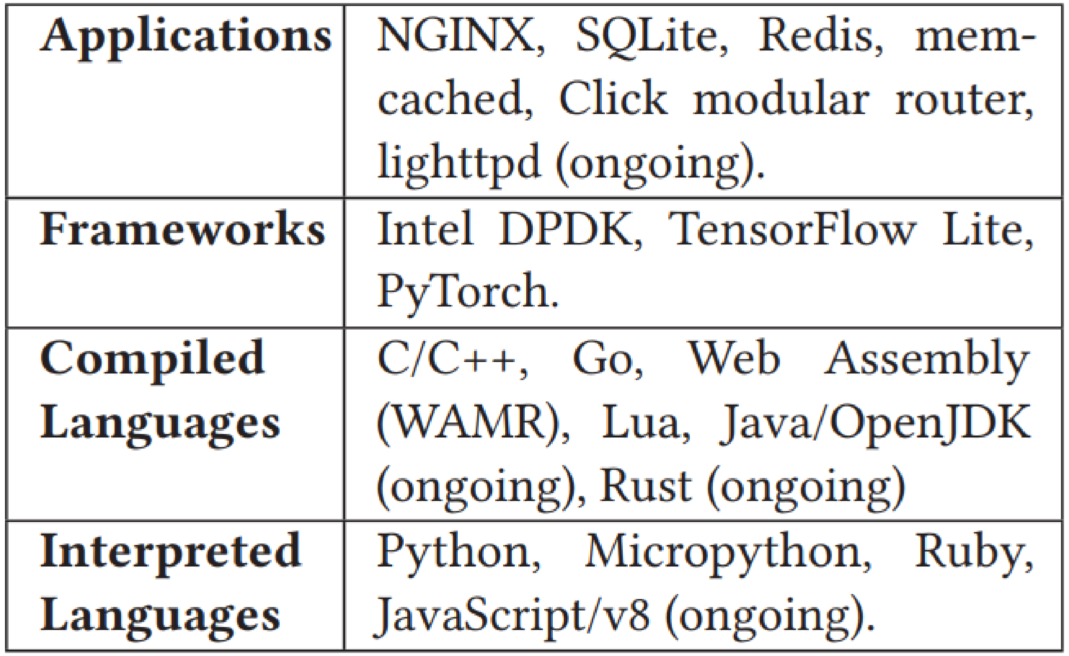


Figure 6. Devel survey of total effort to port a library, including dependencies, missing OS and build system primitives.

Porting functionality has gotten increasingly easier

What Unikraft supports



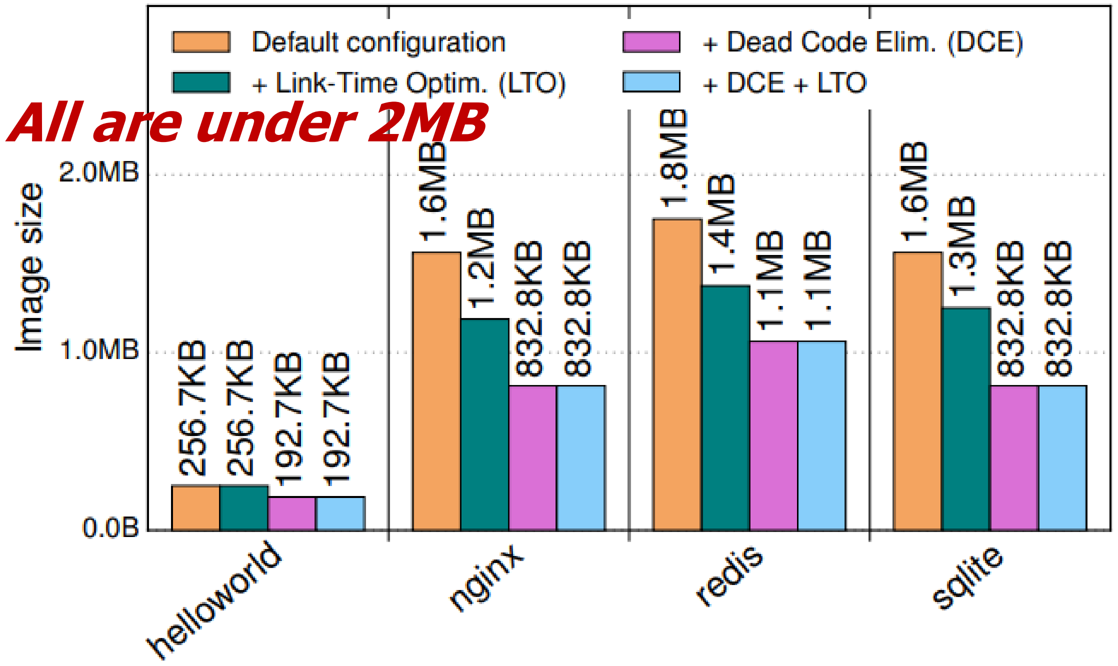
Applications	NGINX, SQLite, Redis, memcached, Click modular router, lighttpd (ongoing).
Frameworks	Intel DPDK, TensorFlow Lite, PyTorch.
Compiled Languages	C/C++, Go, Web Assembly (WAMR), Lua, Java/OpenJDK (ongoing), Rust (ongoing)
Interpreted Languages	Python, Micropython, Ruby, JavaScript/v8 (ongoing).

Table 3. Applications, frameworks and languages currently supported by Unikraft.

• • •

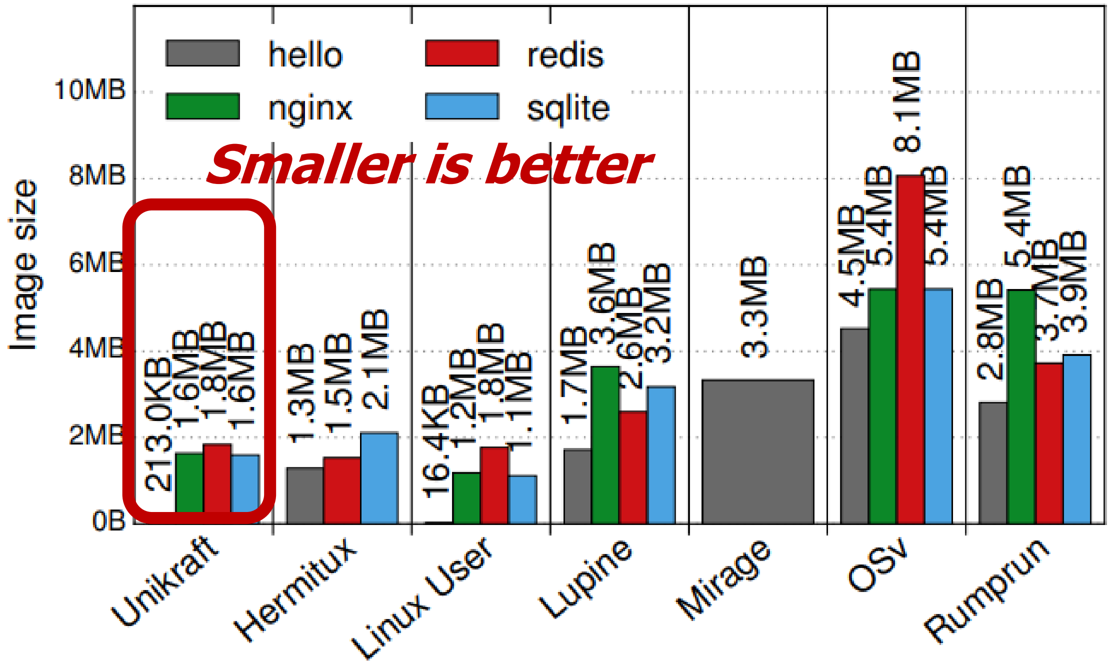
Evaluation

- Low resource consumption: small image size



All are under 2MB

Figure 8. Image sizes of Unikraft applications with and without LTO and DCE.



Smaller is better

Figure 9. Image sizes for Unikraft and other OSes, stripped, w/o LTO and DCE.

Evaluation

- Low resource consumption: quick boot time

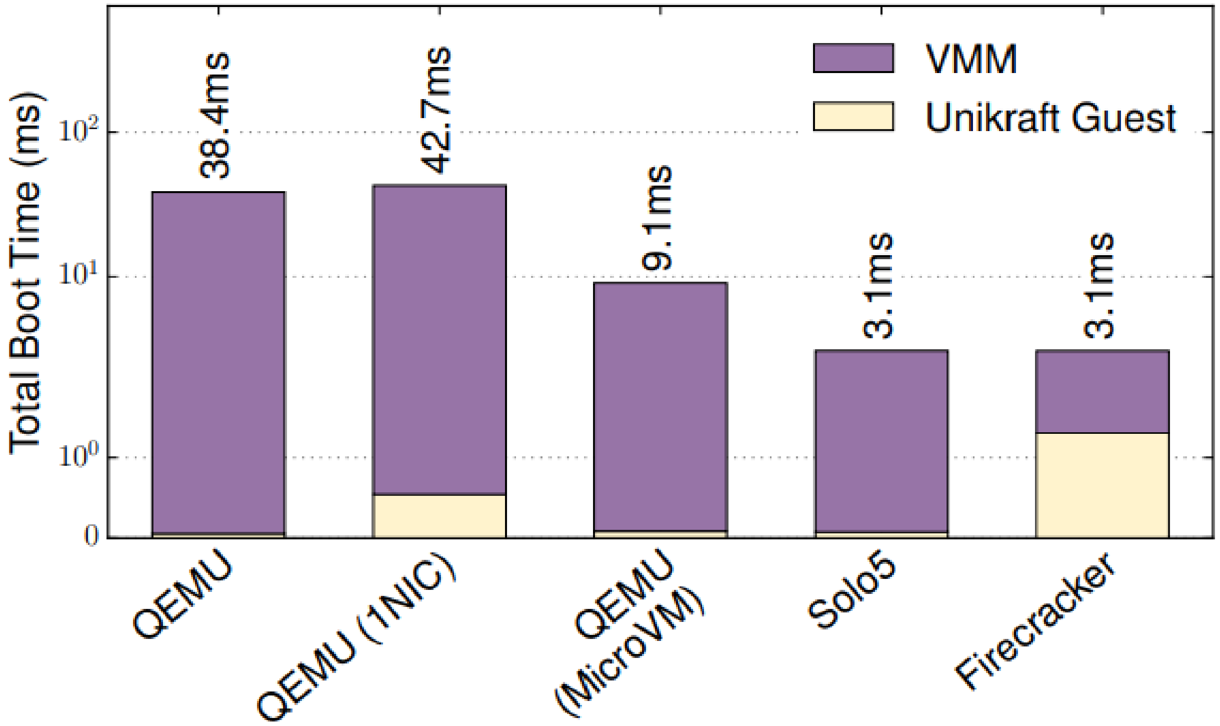


Figure 10. Boot time for Unikraft images with different virtual machine monitors.

Unikraft can be used where just-in-time instantiation of VMs is needed

Evaluation

- Low resource consumption: minimum memory requirement

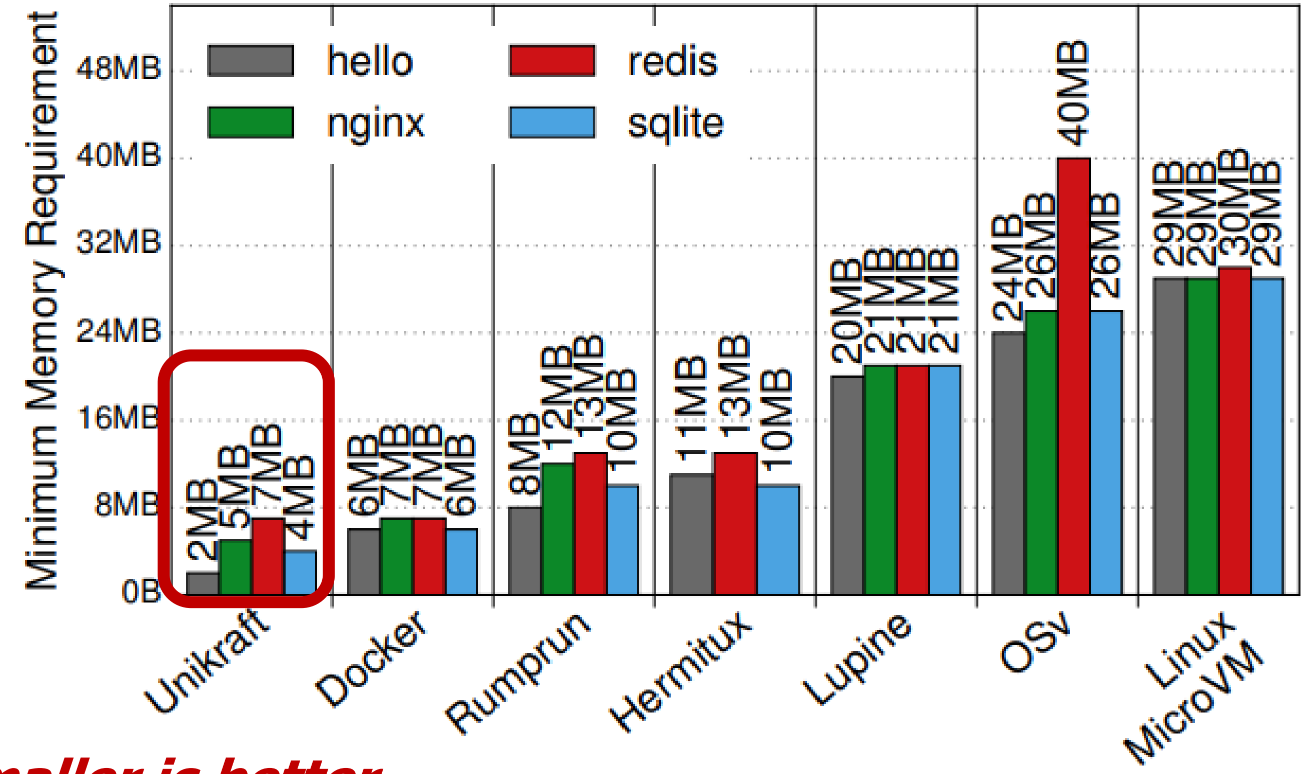


Figure 11. Minimum memory needed to run different applications using different OSes, including Unikraft.

Smaller is better

Evaluation

- Application throughput

higher is better

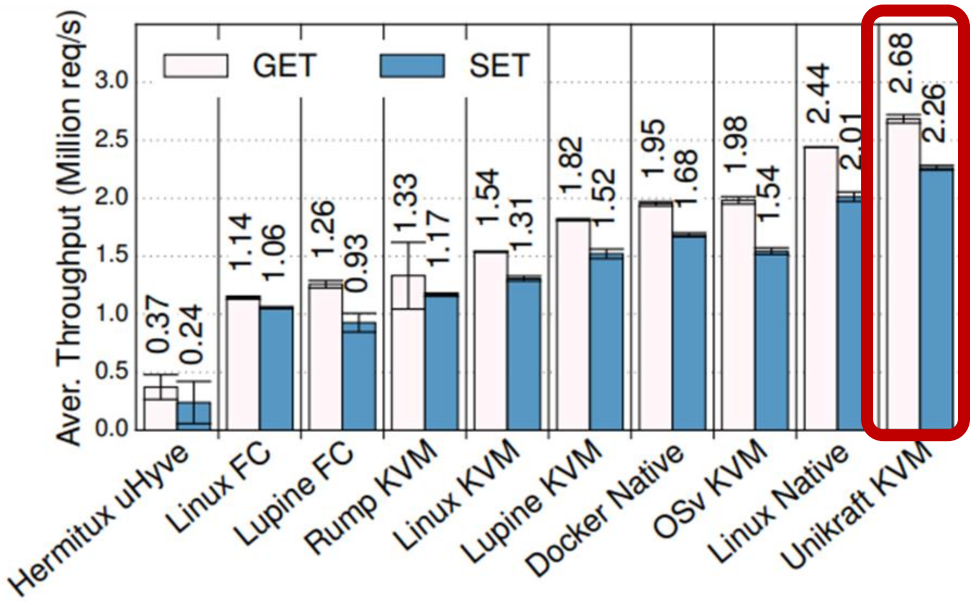


Figure 12. Redis perf (30 conns, 100k reqs, pipelining 16) with QEMU/KVM and Firecracker (FC).

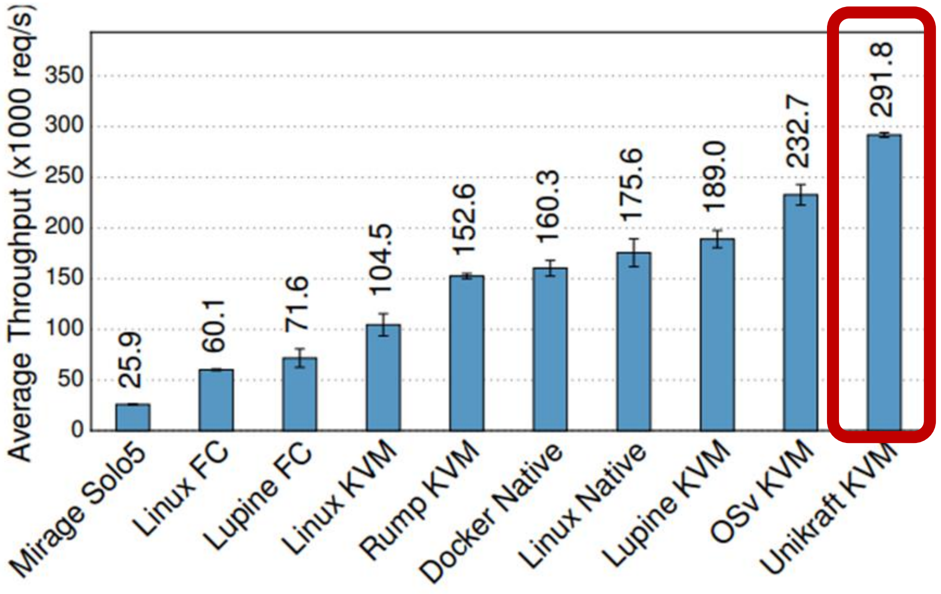
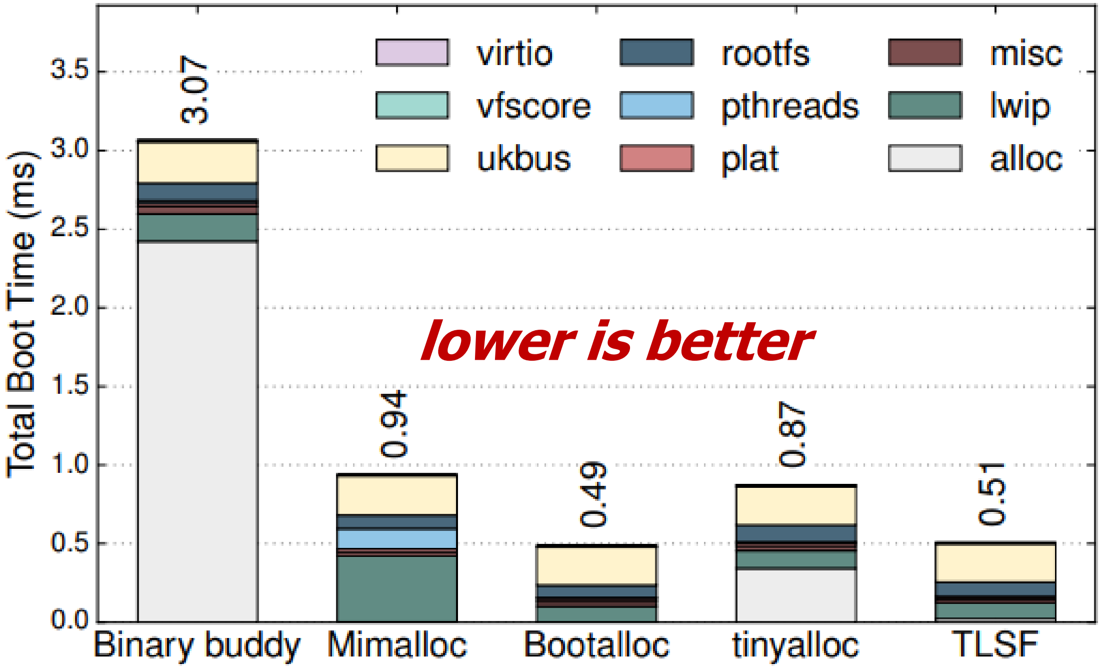


Figure 13. NGINX (and Mirage HTTP-reply) performance with wrk (1 minute, 14 threads, 30 conns, static 612B page).

Unikraft is 30-80% faster than running the same app in a container, 70-170% faster than in a Linux VM, 10-60% faster than Native Linux

Evaluation

- Memory allocators



lower is better

Figure 14. Unikraft Boot time for Nginx with different allocators.

higher is better

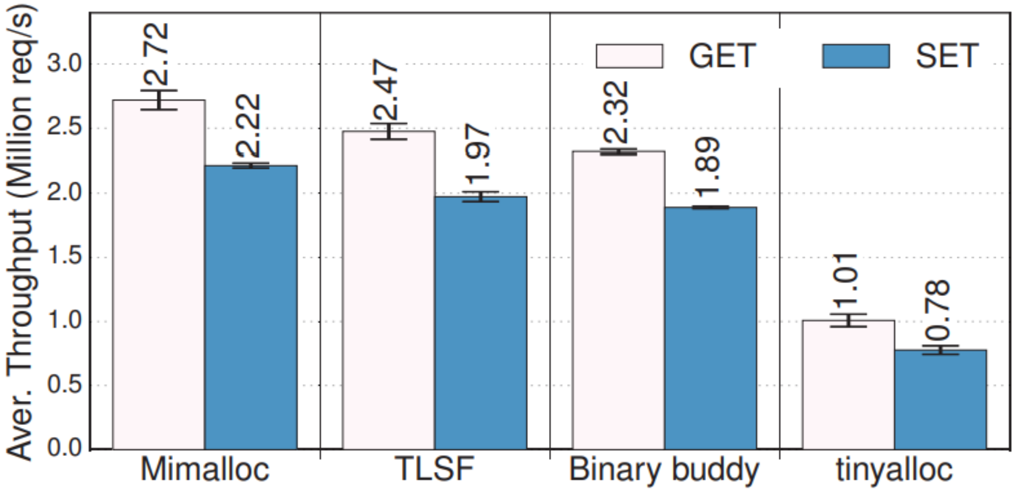


Figure 18. Redis throughput on Unikraft for different allocators (`redis-benchmark`, 30 conns, 100k requests, pipelining level of 16.)

Allocators affect bood time and throughput and we can choose one of them

Evaluation

- Filesystem specialization: SHFS

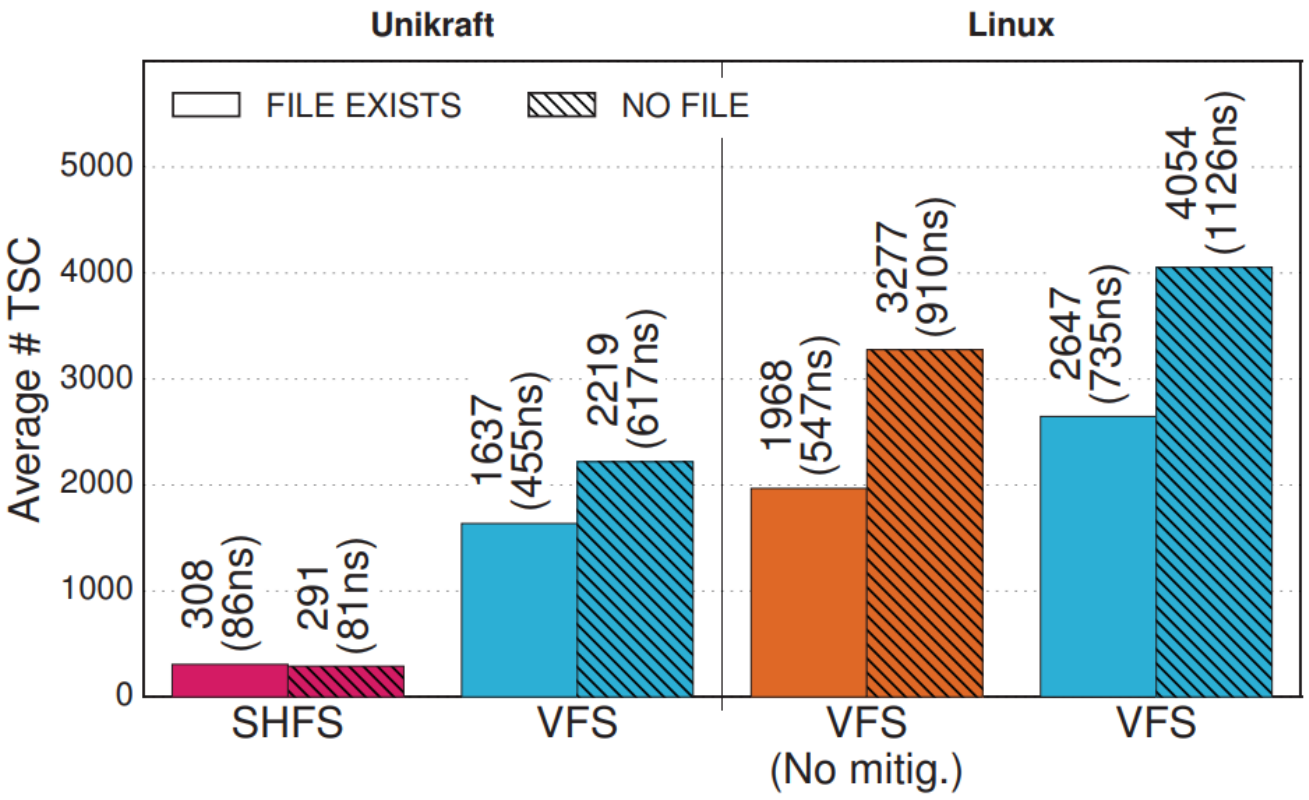


Figure 22. Perf. with a specialized filesystem and removing the VFS layer.

Conclusion

- Unikernels are infamous for the difficulty to port existing applications
- Unikraft is a novel micro-library OS that makes customize the unikernel easy
- Unikraft helps developers quickly and easily create resource-efficient, high-performance unikernels

https://github.com/unikraft/eurosys21-artifacts

Unikraft: Fast, Specialized Unikernels the Easy Way

Simon Kuenzer, Vlad-Andrei Bădoiu, Hugo Lefeuvre, Sharan Santhanam, Alexander Jung, Gaulthier Gain, Cyril Soldani, Costin Lupu, Ştefan Teodorescu, Costi Răducanu, Cristian Banu, Laurent Mathy, Răzvan Deaconescu, Costin Raiciu, and Felipe Huici

EuroSys'21

Thank You !

August 5th, 2022

Presented by Yejin Han

yj0225@dankook.ac.kr