Unikraft: Fast, Specialized Unikernels the Easy Way

Simon Kuenzer
Sharan Santhanam
Cyril Soldani
Costi Răducanu
Răzvan Deaconescu

Vlad-Andrei Bădoiu
Alexander Jung
Costin Lupu
Cristian Banu
Costin Raiciu

Hugo Lefeuvre
Gaulthier Gain
Stefan Teodorescu
Laurent Mathy
Felipe Huici

Eurosys 2021, April 26th-28th

Specialization = High Performance

software

OCaml

JS

DPDK
DATA PLANE DEVELOPMENT KIT

hardware

intel
Movidius™

BlueField

Google
Tensor Processing Unit

Unikernels = Specialized Virtual Machines

GOALS

- **Easy to build and run**
- **Easy or no app porting**
- **Great performance**

Design Principles

1. Fully **modular** kernel

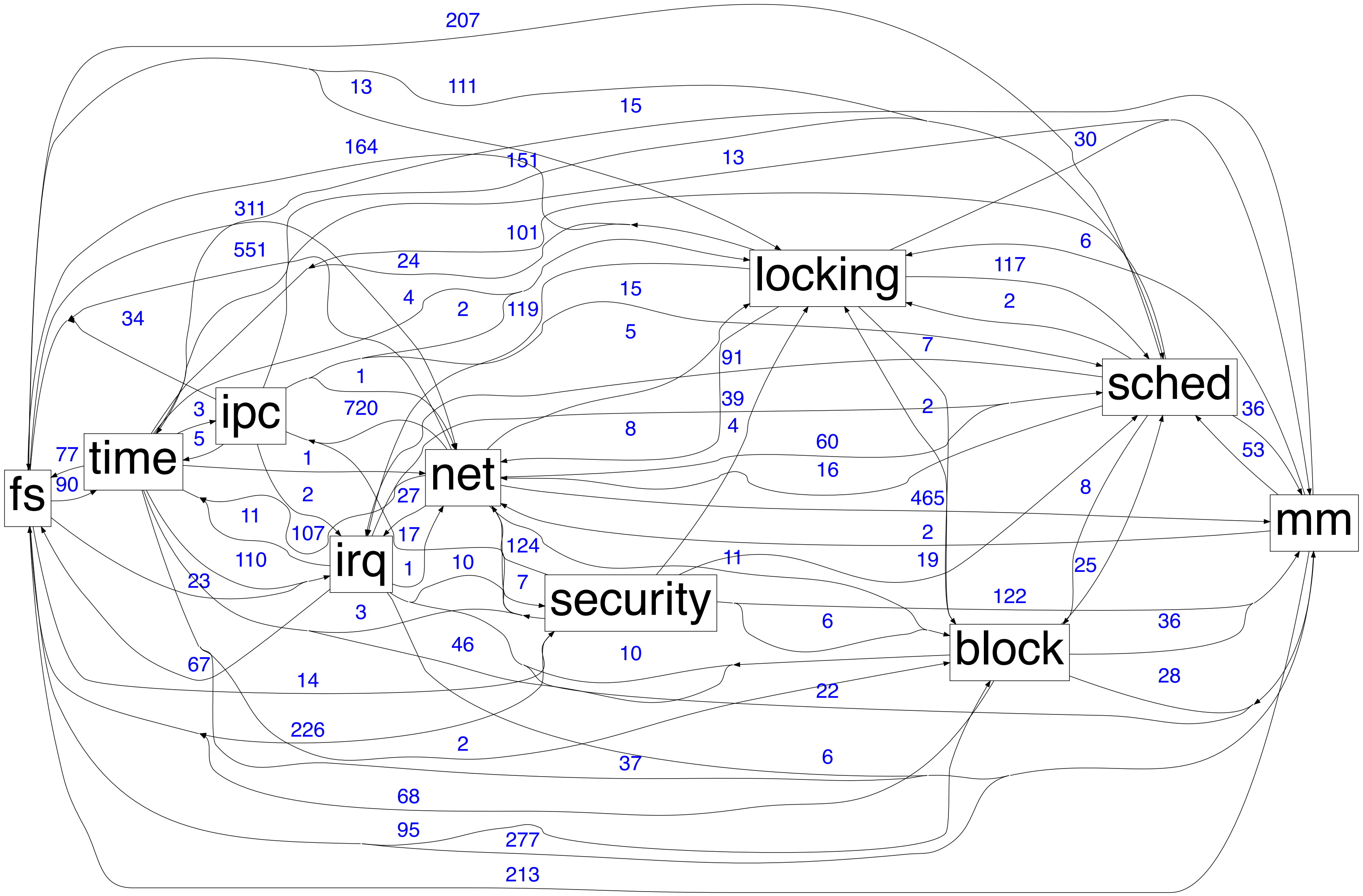
2. Provide high performance **specialized APIs**

Design Principles

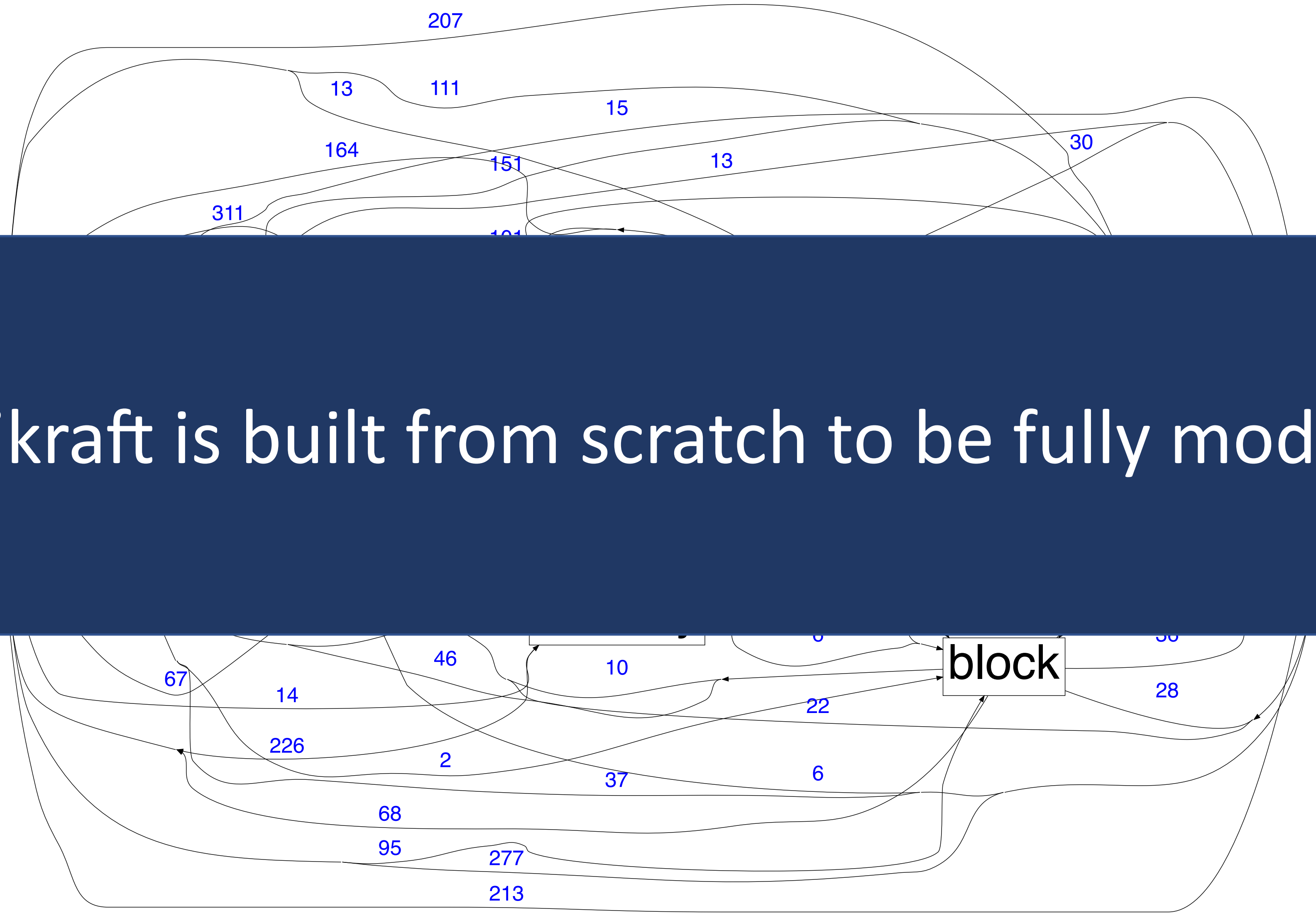
1. Fully **modular** kernel

2. Provide high performance **specialized APIs**

Why not Linux?

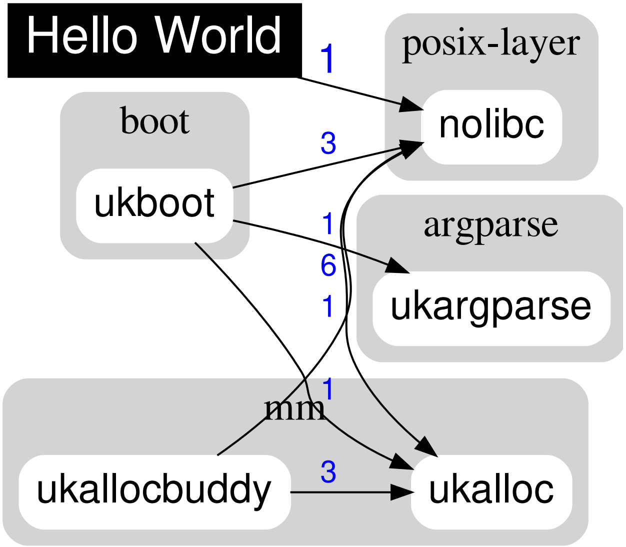


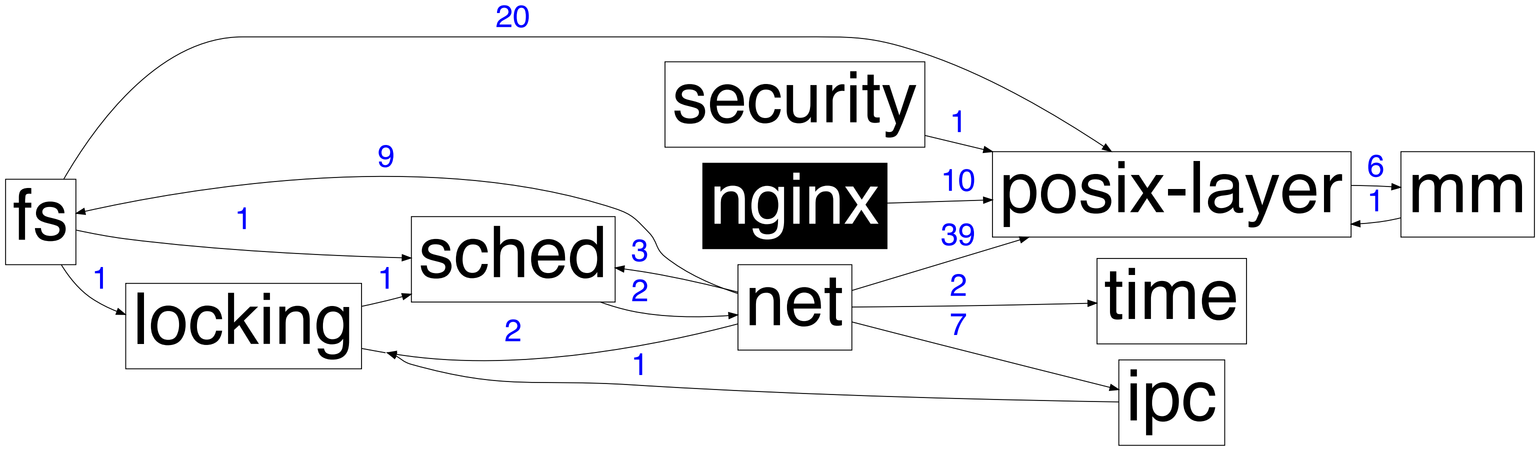
Why not Linux?



Unikraft is built from scratch to be fully modular

With Unikraft





Doing it with existing unikernels?

(1) Require significant expert work to build

(2) They are often non-POSIX compliant

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

Doing it with existing unikernels?

(1) Require significant expert work to build

Unikraft is built from scratch (with borrowing)

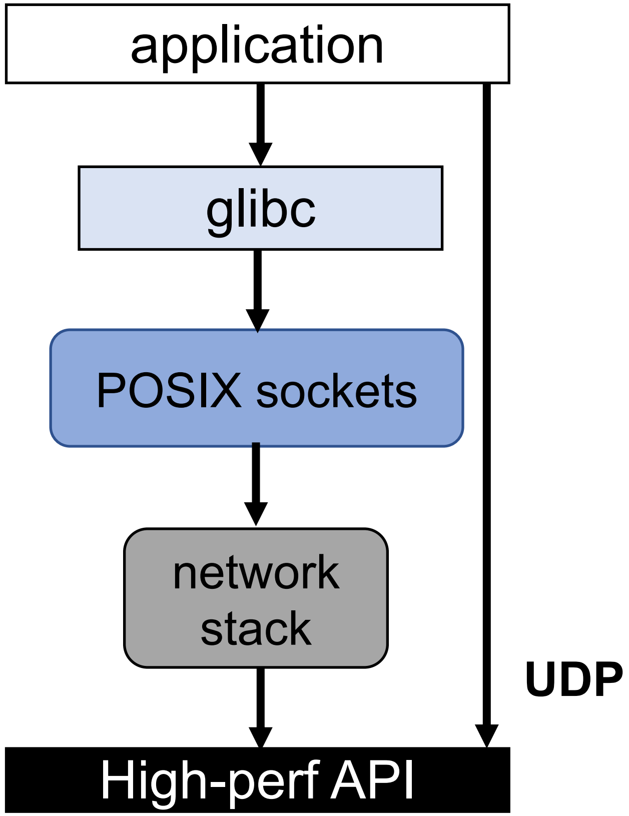
(3) The (uni)kernels are *still* monolithic

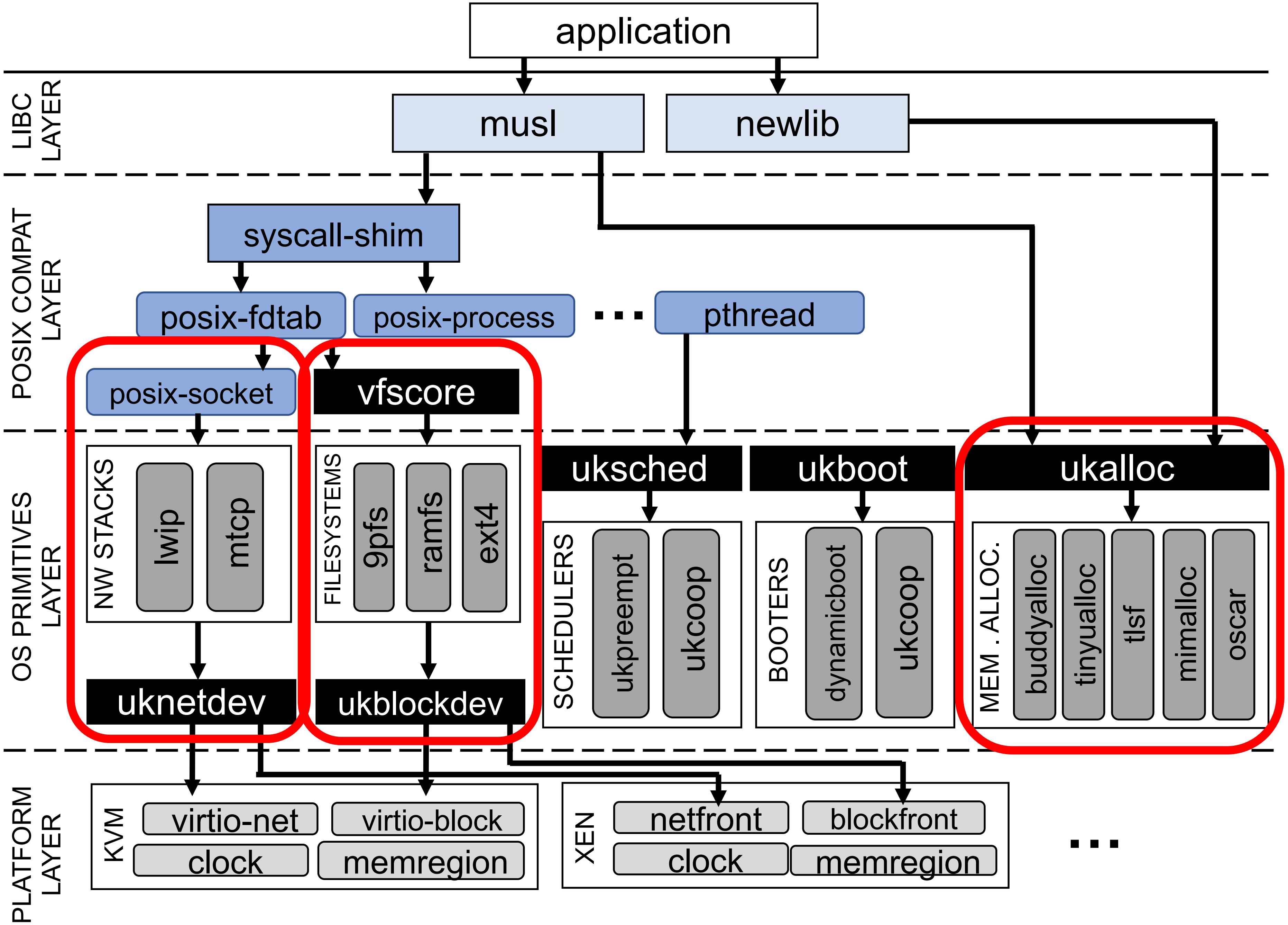
Design Principles

1. Fully **modular** kernel

2. Provide high performance **specialized APIs**

Specialized API Example




application
LIBC LAYER
musl
newlib
POSIX COMPAT LAYER
syscall-shim
posix-fdtab
posix-process
...
pthread
posix-socket
vfscore
OS PRIMITIVES LAYER
NW STACKS
lwip
mtcp
FILESYSTEMS
9pfs
ramfs
ext4
uksched
ukboot
ukalloc
SCHEDULERS
ukpreempt
ukcoop
BOOTERS
dynamicboot
ukcoop
MEM . ALLOC.
buddyalloc
tinyualloc
tlsf
mimalloc
oscar
uknetdev
ukblockdev
PLATFORM LAYER
KVM
virtio-net
virtio-block
clock
memregion
XEN
netfront
blockfront
clock
memregion
...

GOALS

- Easy to build and run
- Easy or no app porting
- Great performance

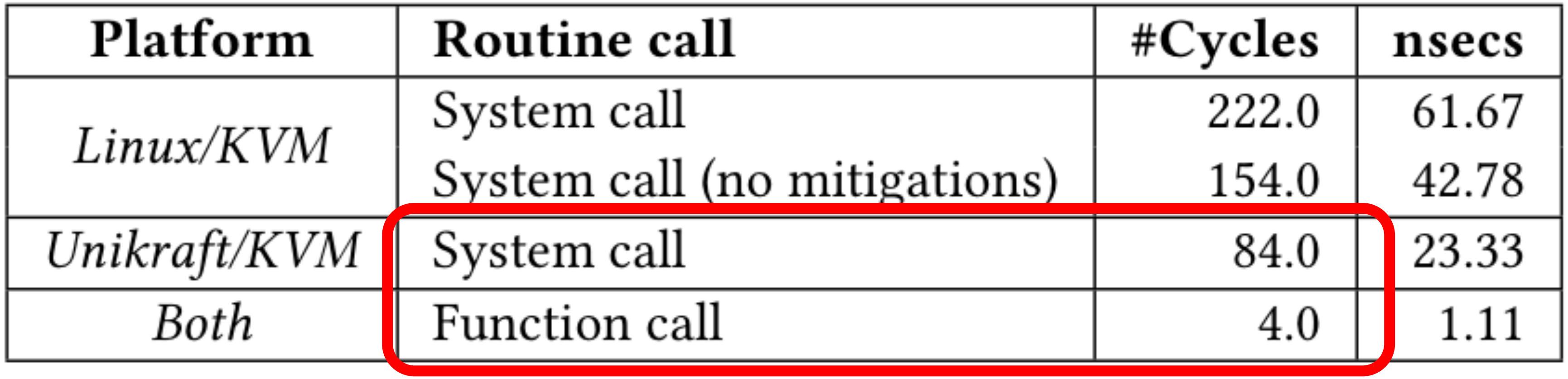
```
root@kraft:~#
```

```
[0] 0:bash*                                                              kraft NGINX demo
```

GOALS

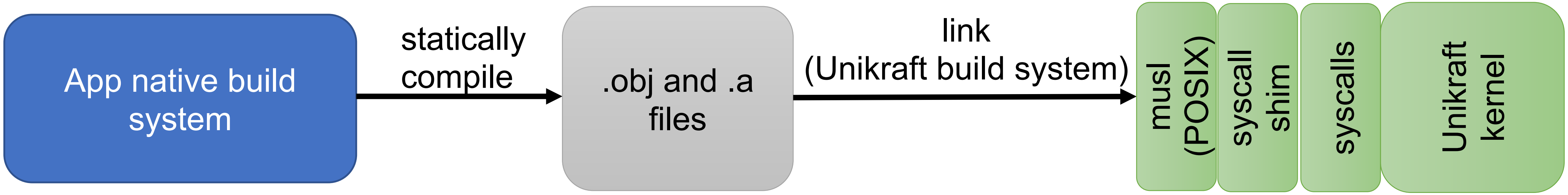
- Easy to build and run
- **Easy or no app porting**
- Great performance

Binary Compatibility?

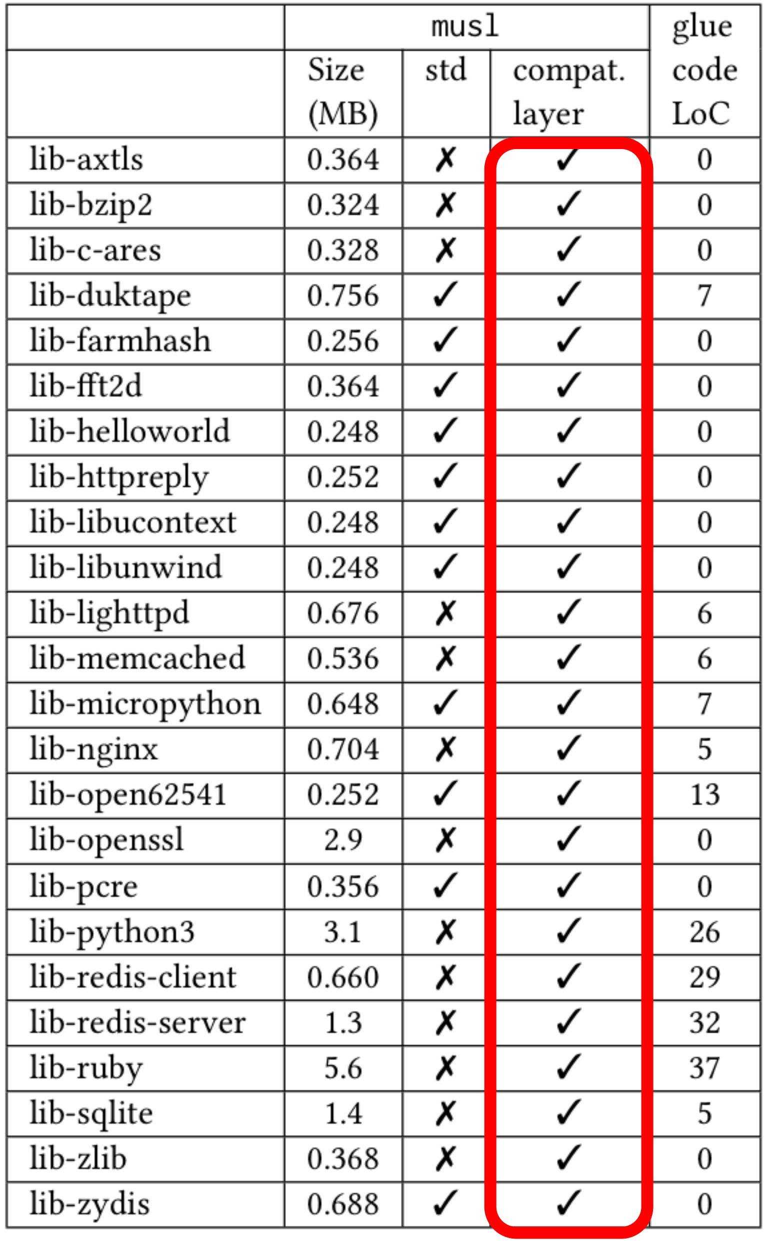


Platform	Routine call	#Cycles	nsecs
Linux/KVM	System call	222.0	61.67
	System call (no mitigations)	154.0	42.78
Unikraft/KVM	System call	84.0	23.33
Both	Function call	4.0	1.11

Auto-porting from Source



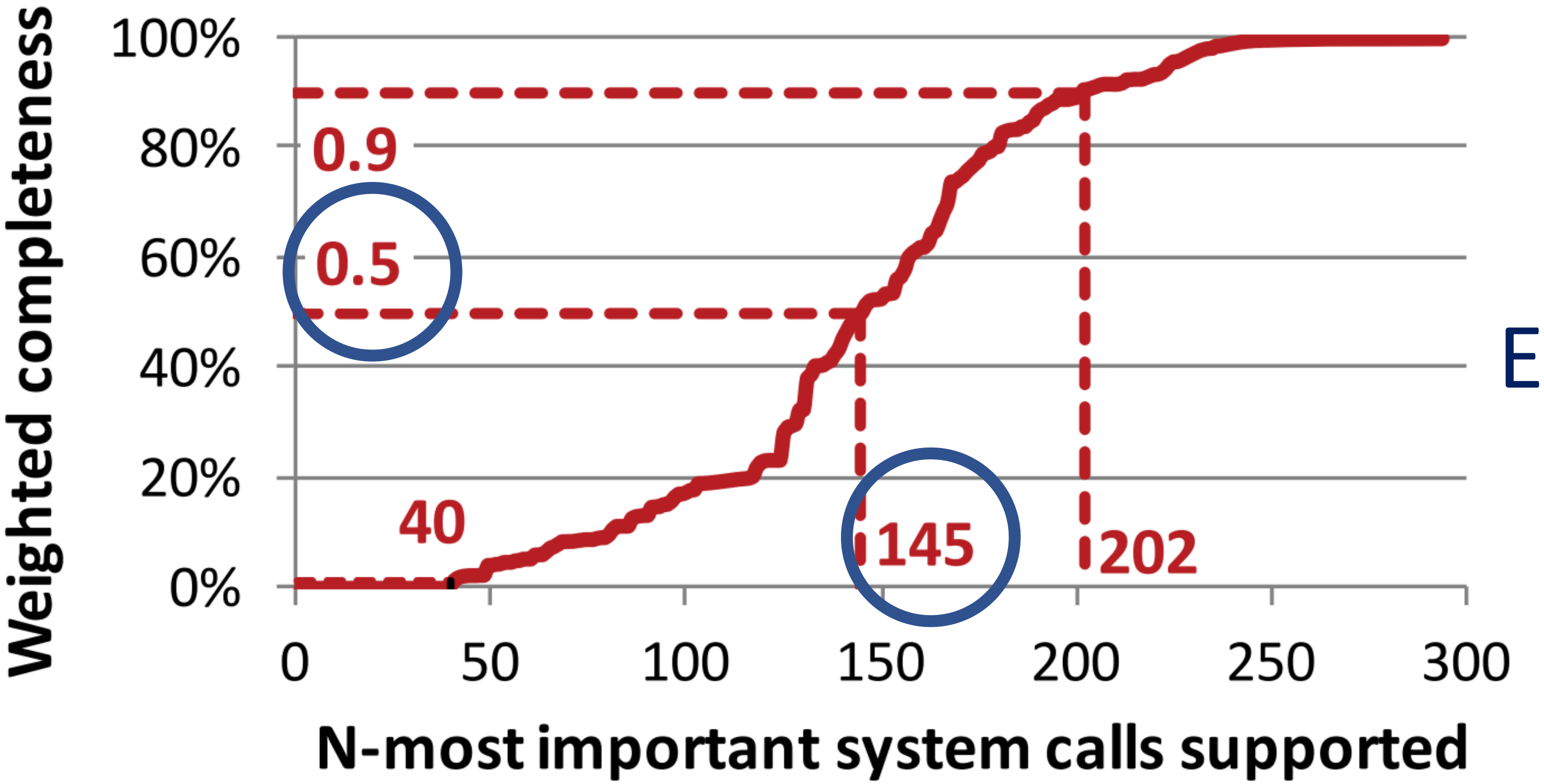
Compile Time



	musl			glue code LoC
	Size (MB)	std	compat. layer	
lib-axtls	0.364	✗	✓	0
lib-bzip2	0.324	✗	✓	0
lib-c-ares	0.328	✗	✓	0
lib-duktape	0.756	✓	✓	7
lib-farmhash	0.256	✓	✓	0
lib-fft2d	0.364	✓	✓	0
lib-helloworld	0.248	✓	✓	0
lib-httpreply	0.252	✓	✓	0
lib-libucontext	0.248	✓	✓	0
lib-libunwind	0.248	✓	✓	0
lib-lighttpd	0.676	✗	✓	6
lib-memcached	0.536	✗	✓	6
lib-micropython	0.648	✓	✓	7
lib-nginx	0.704	✗	✓	5
lib-open62541	0.252	✓	✓	13
lib-openssl	2.9	✗	✓	0
lib-pcre	0.356	✓	✓	0
lib-python3	3.1	✗	✓	26
lib-redis-client	0.660	✗	✓	29
lib-redis-server	1.3	✗	✓	32
lib-ruby	5.6	✗	✓	37
lib-sqlite	1.4	✗	✓	5
lib-zlib	0.368	✗	✓	0
lib-zydis	0.688	✓	✓	0

What about syscall support?

Syscall Support

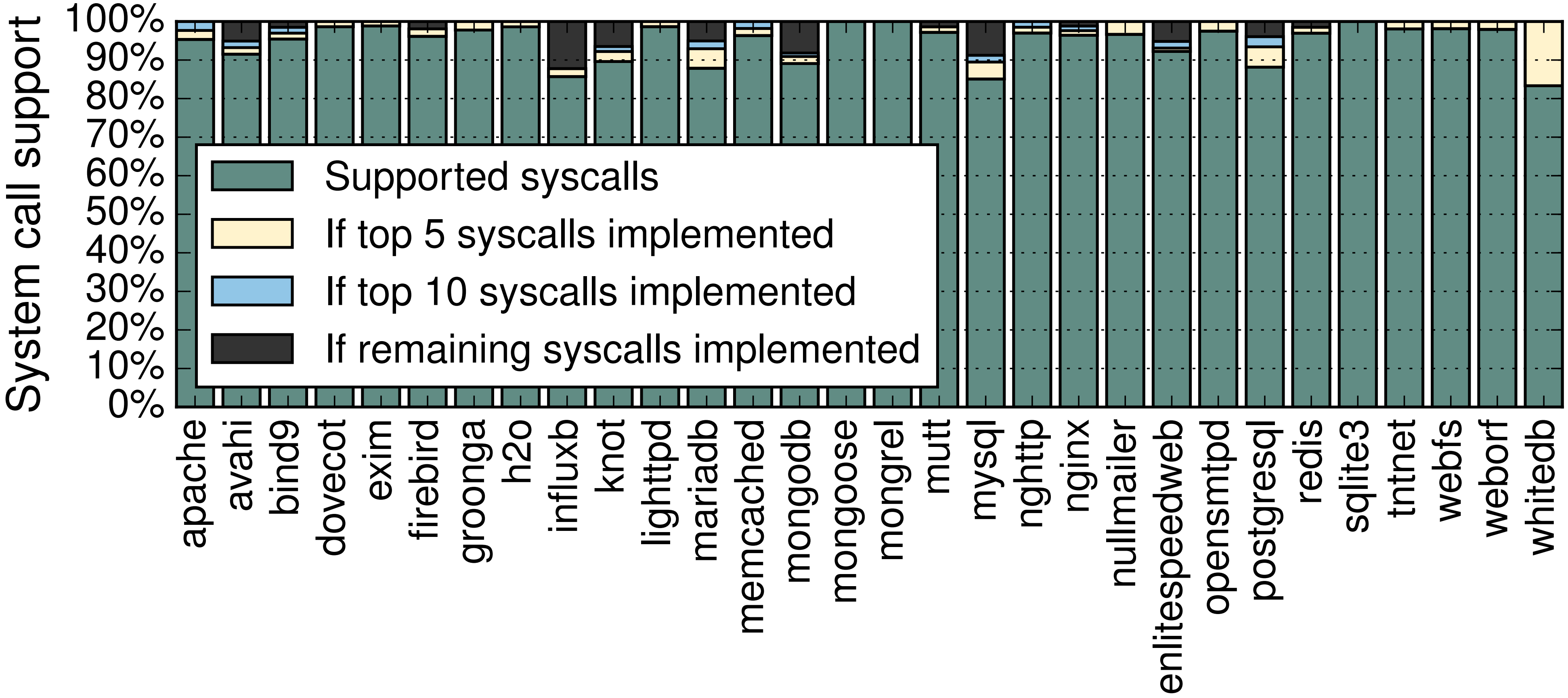


Eurosys 2016

146 syscalls currently supported

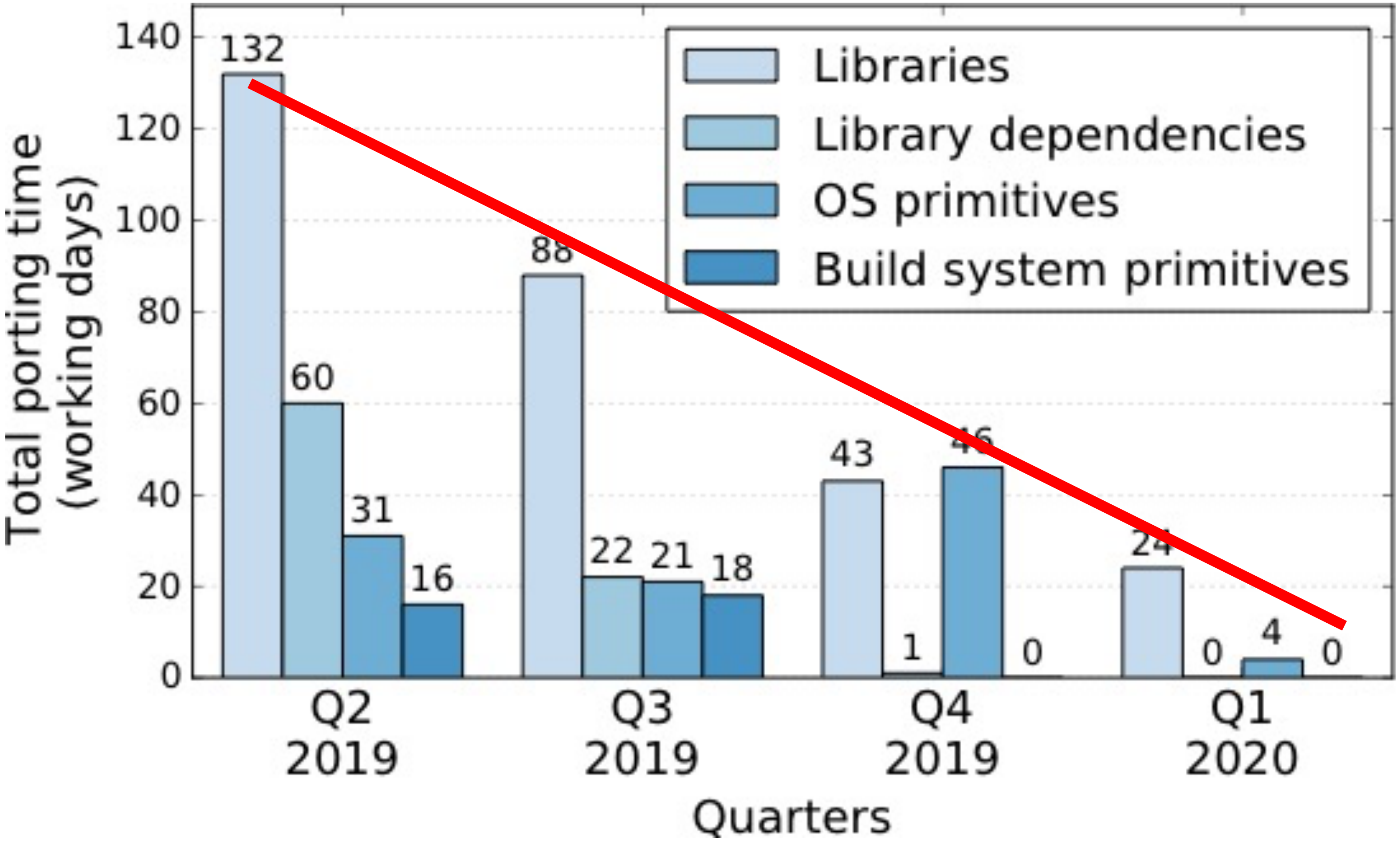
Linux: ~350 syscalls

Syscall Support
Top 30 Debian Popcon Apps

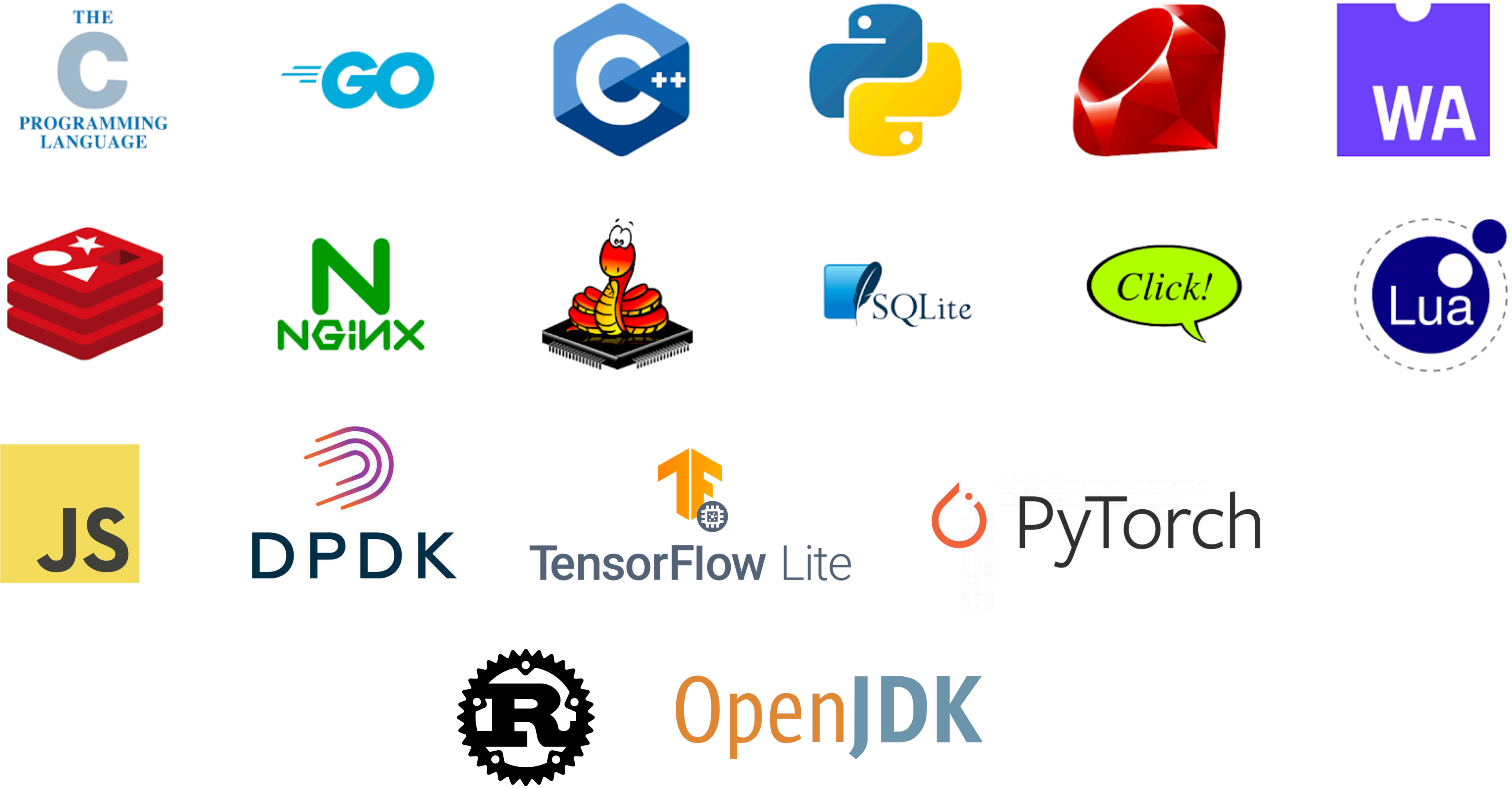


146 syscalls currently supported

If all Else Fails – Manual Porting



What Unikraft Supports (sample)



(ongoing)

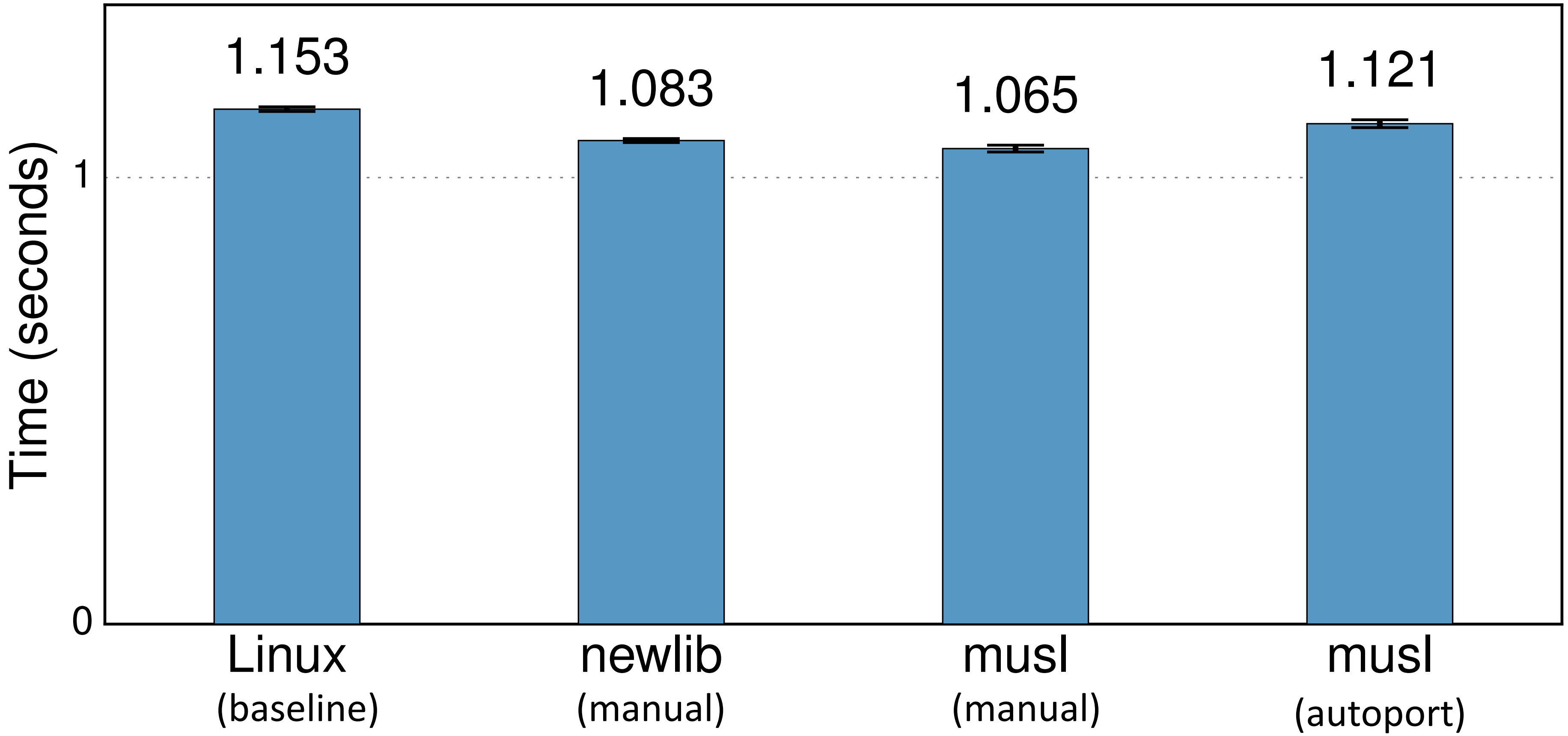
(ongoing)

GOALS

- Easy to build and run
- Easy or no app porting
- **Great performance**

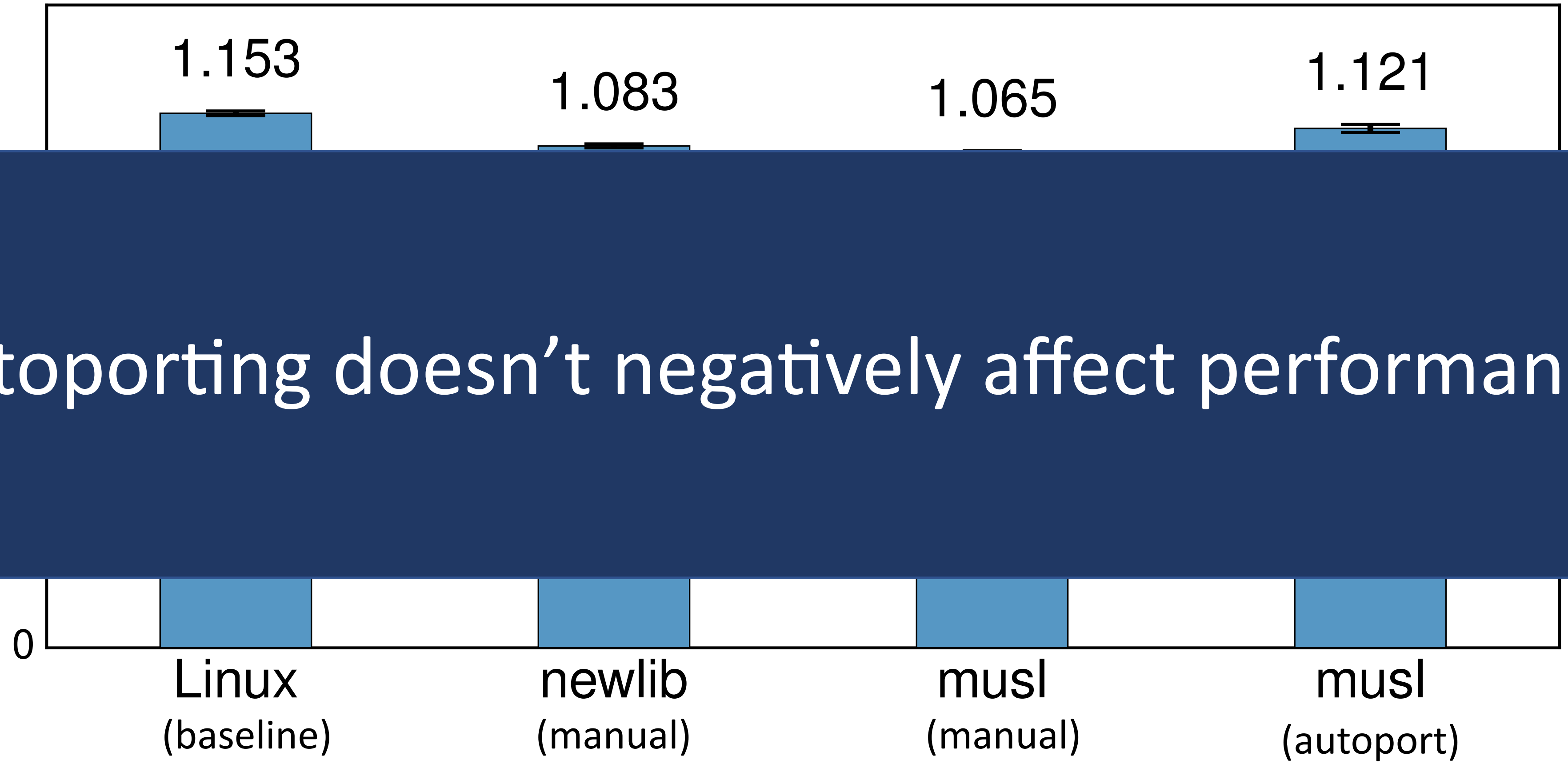
Does autoporting sacrifice performance?

SQLite: Manual vs. Auto Port



time for 60K insertions

SQLite: Manual vs. Auto Port



Autoporting doesn’t negatively affect performance

time for 60K insertions

Transparent Benefits – Boot, Memory, Size, Throughput

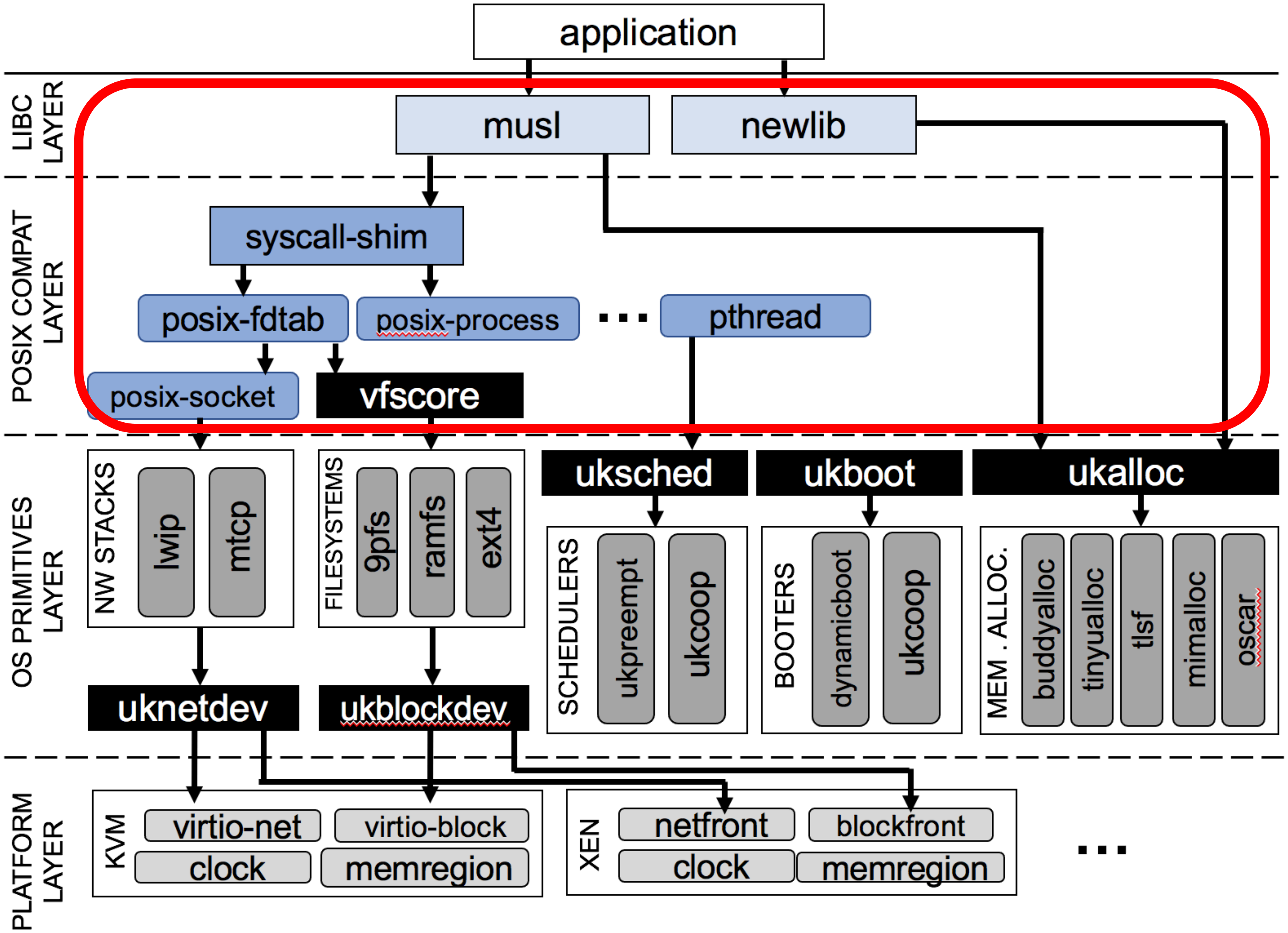
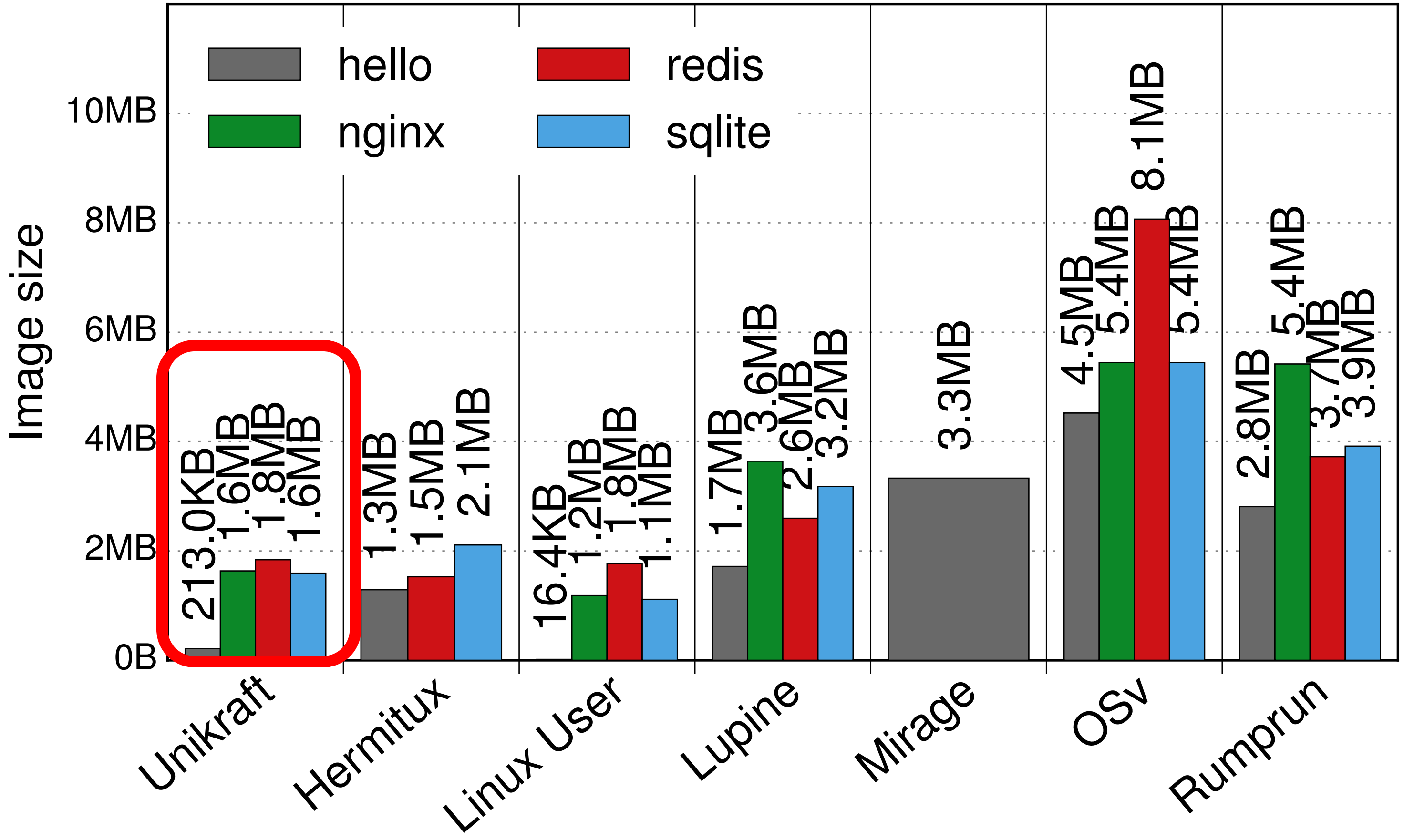
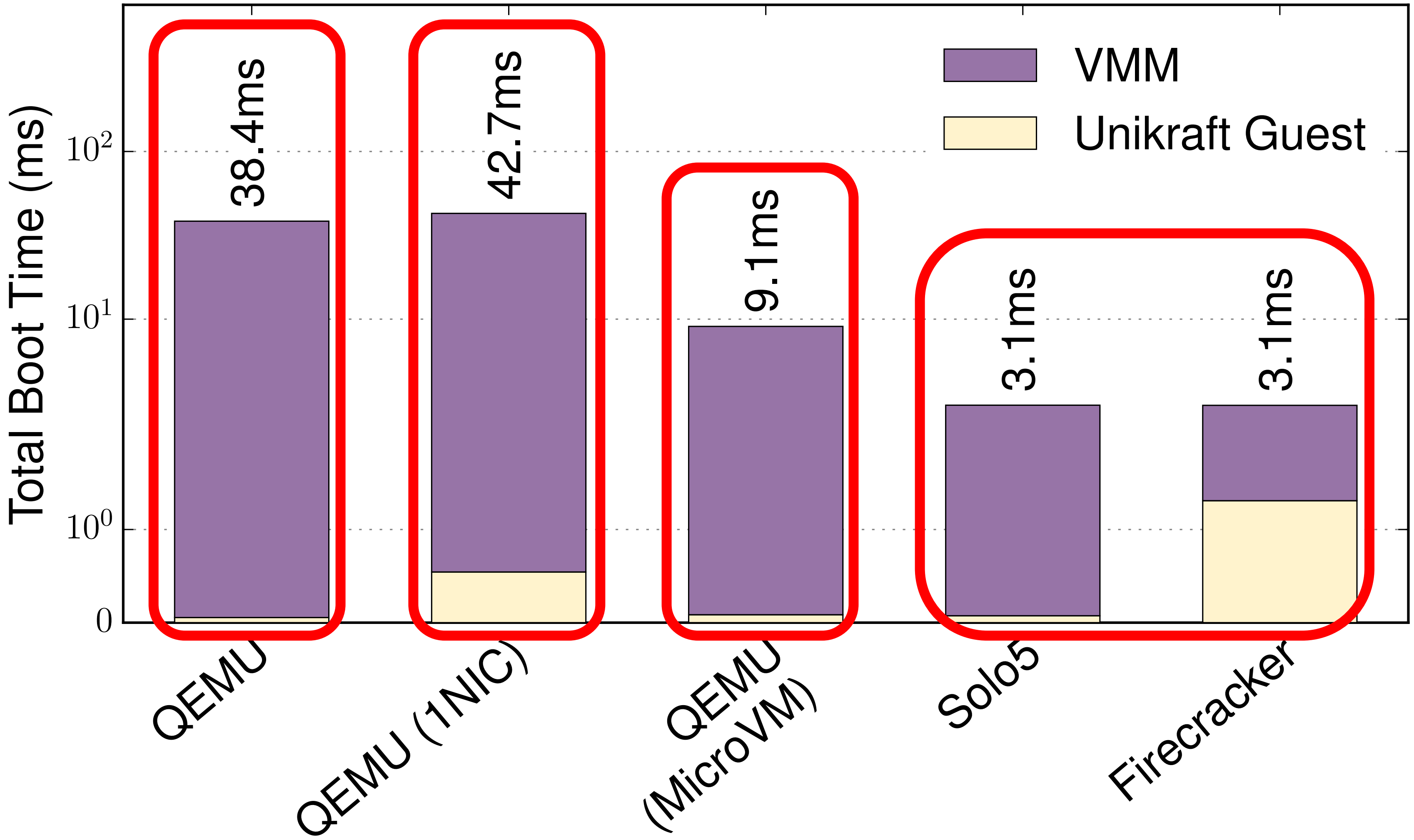


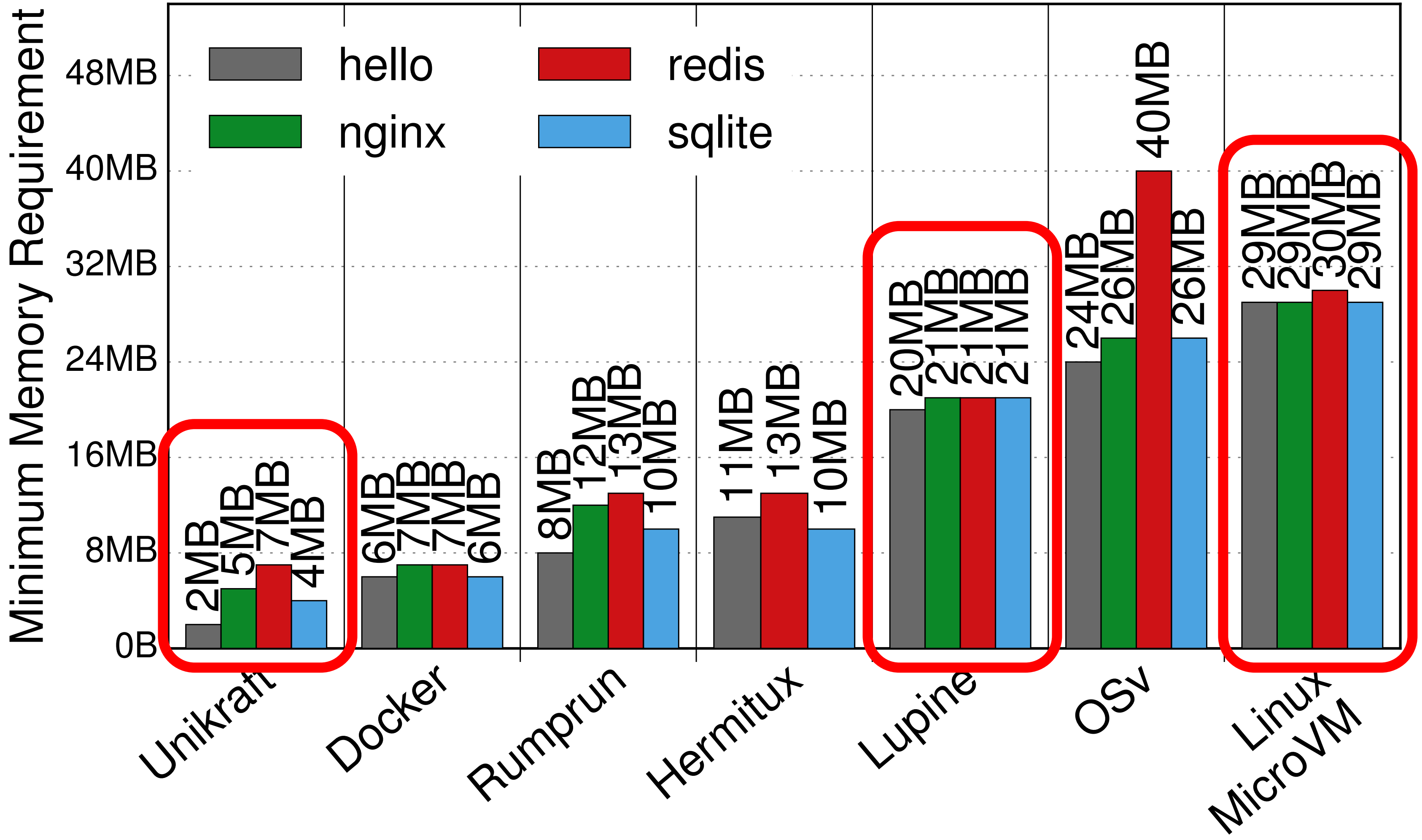
Image Sizes vs. other Projects



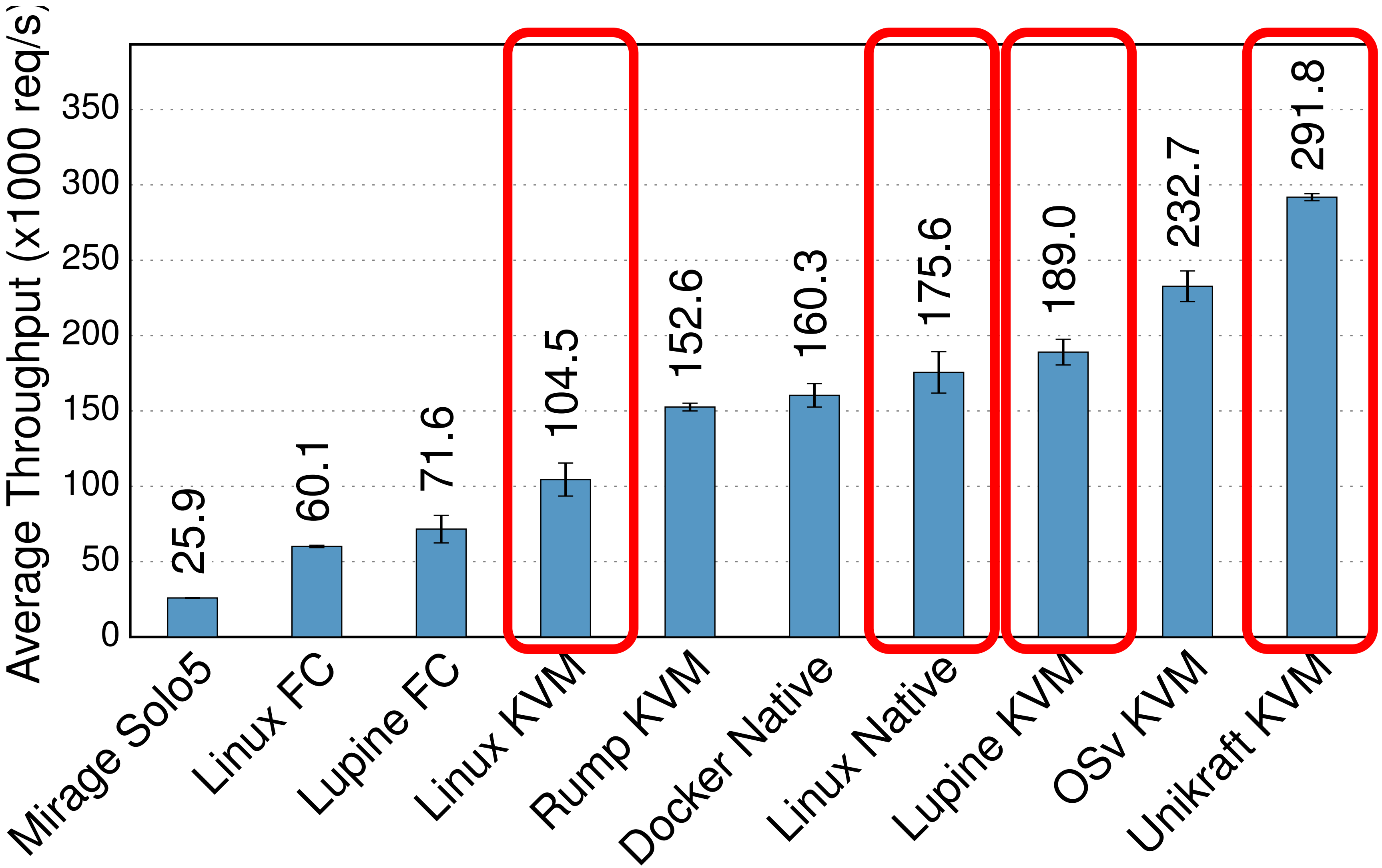
Unikraft Boot Times



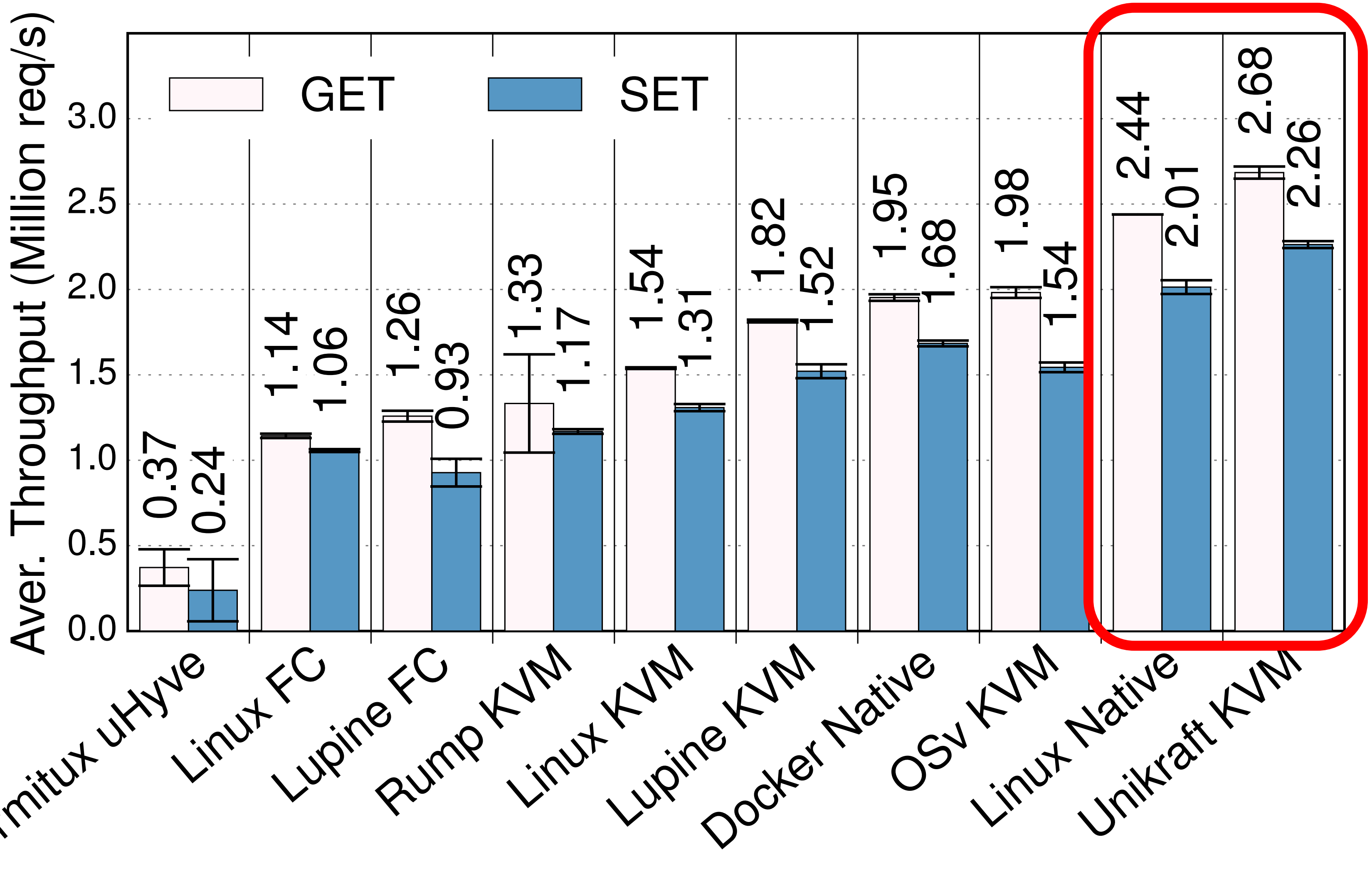
Minimum Memory Requirements



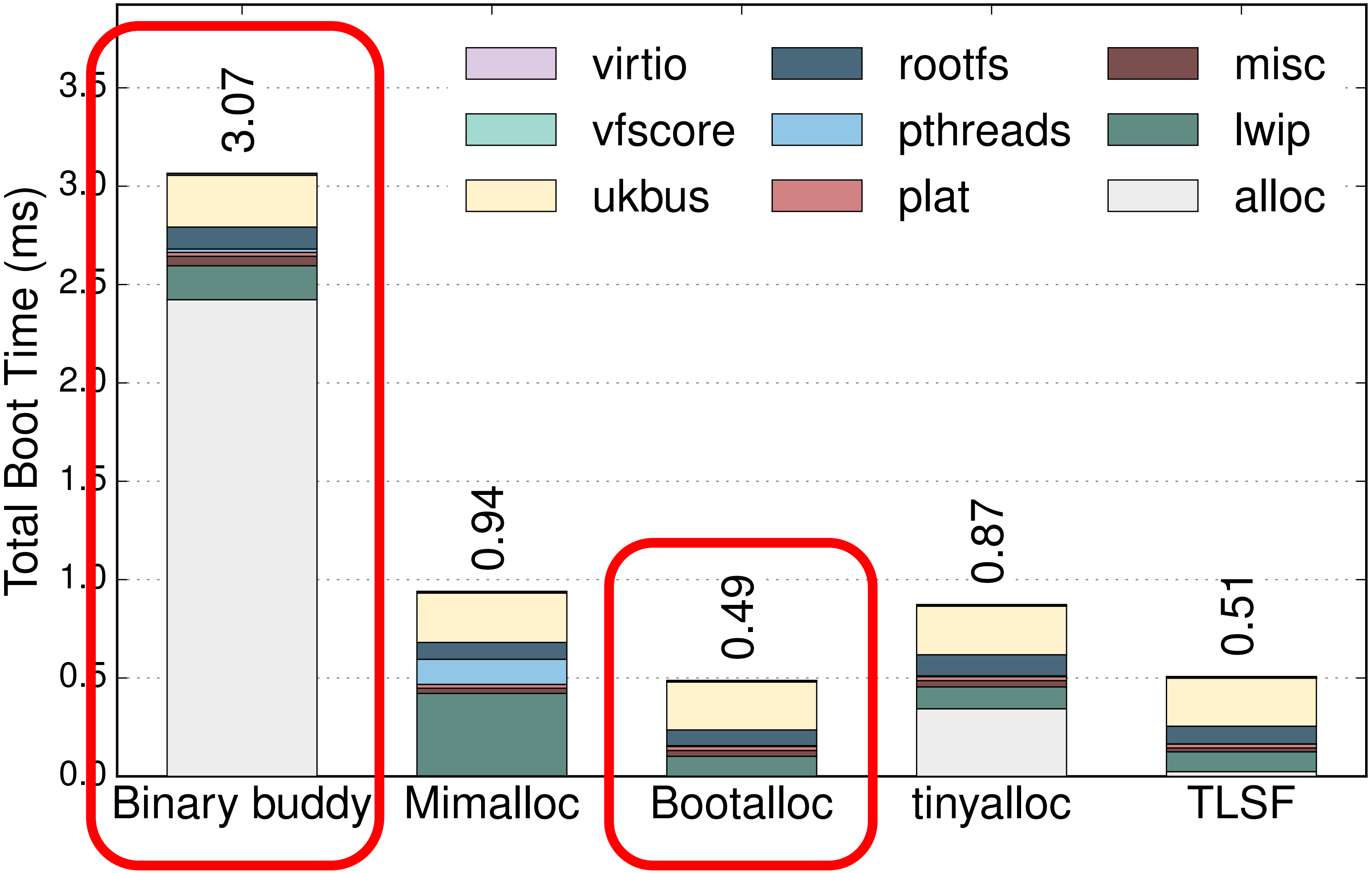
nginx Throughput



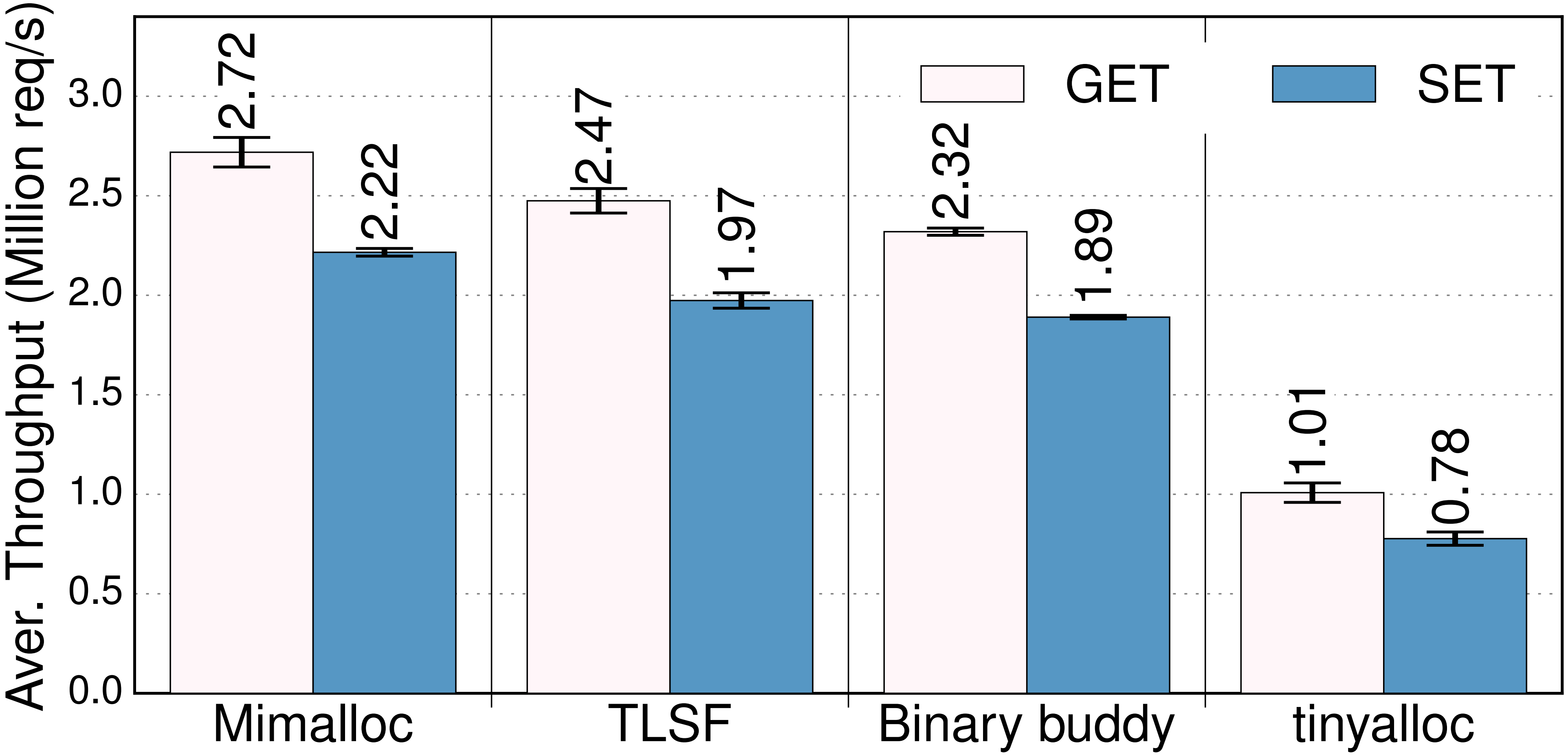
Redis Performance



Boot Times - Different Allocators

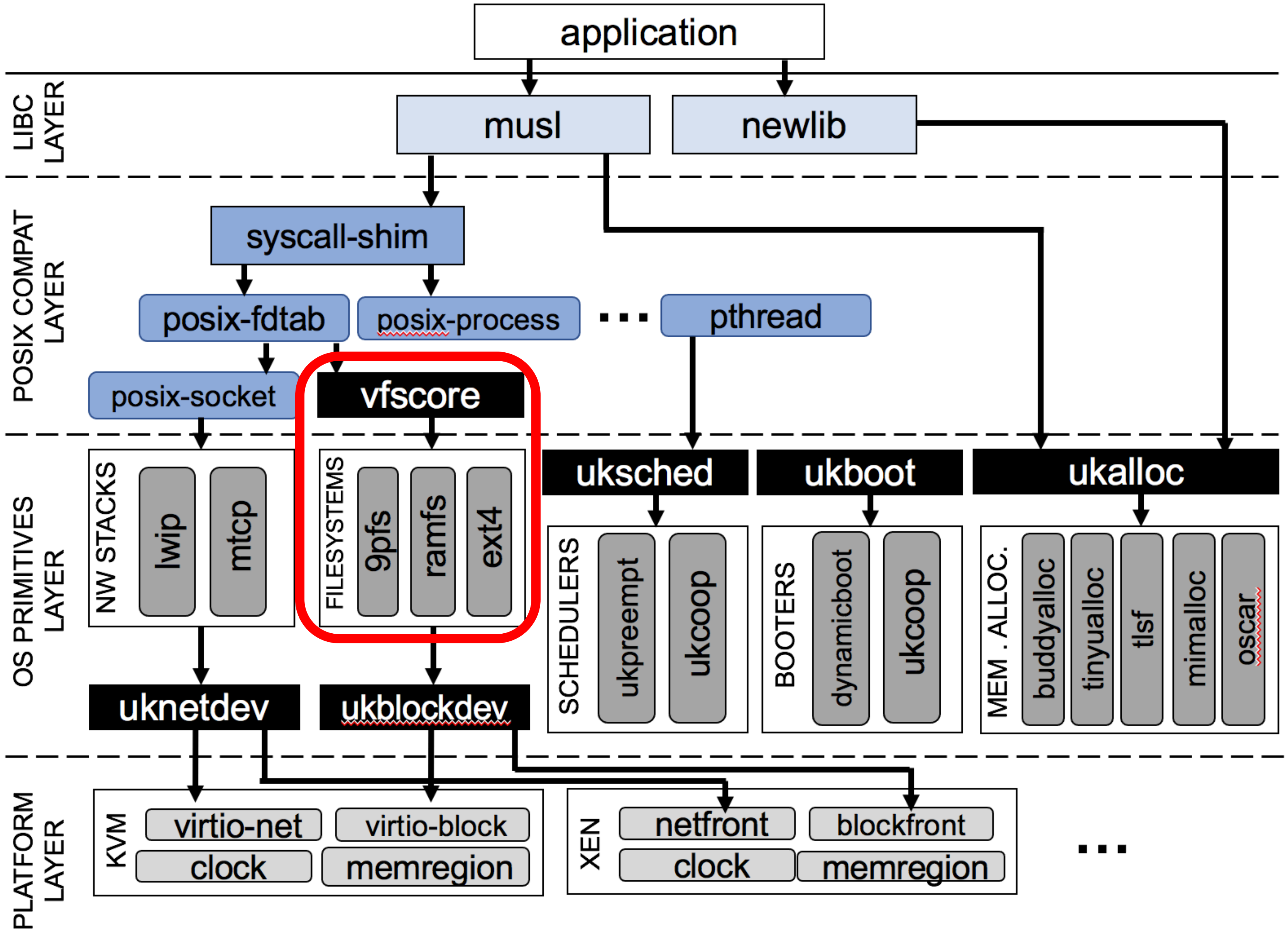


Redis Throughput
Different Allocators

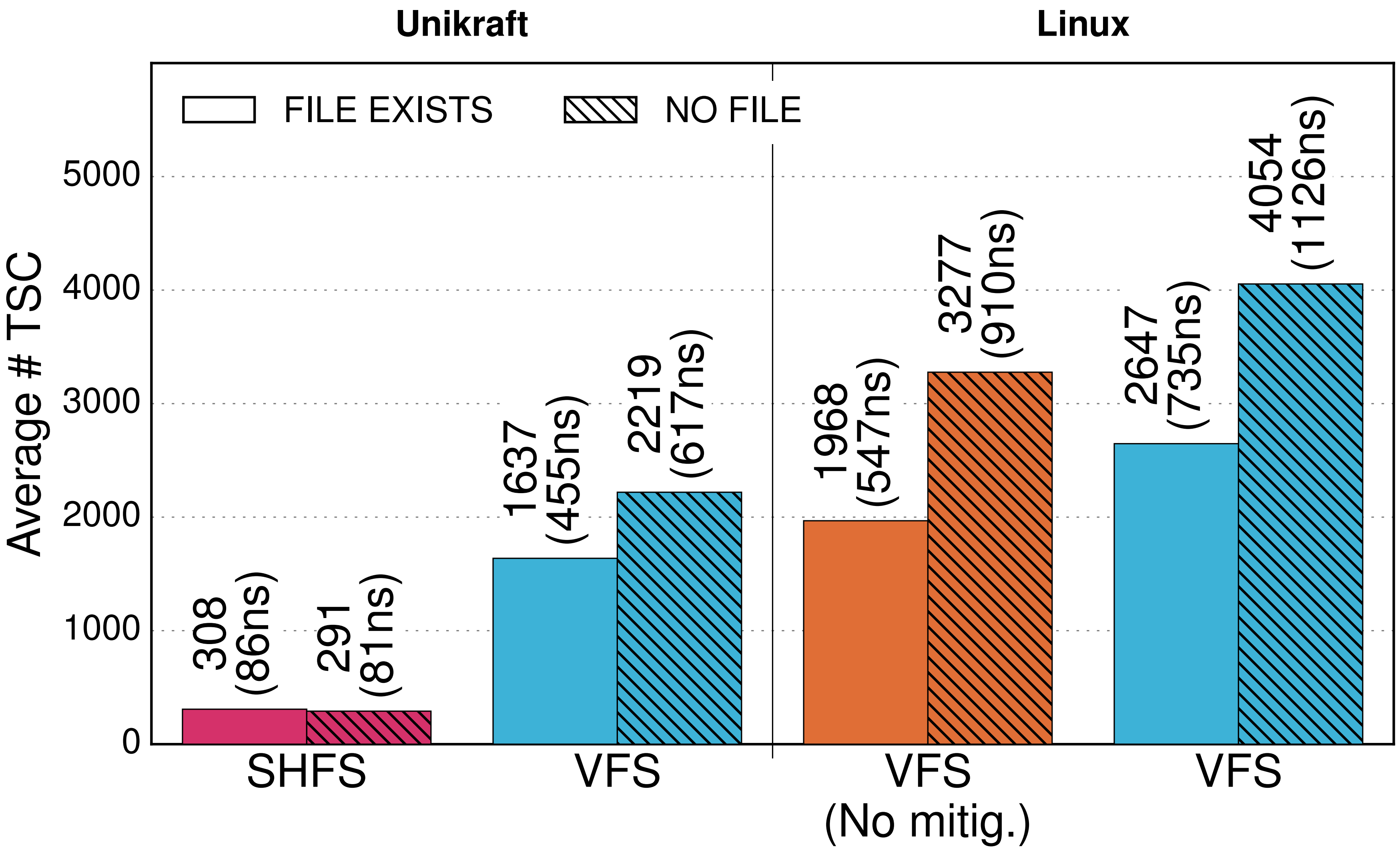


Specialized APIs

Specialization Benefits – Filesystem Performance



Filesystem Specialization – SHFS



THE LINUX FOUNDATION

Xen® Project

Artifacts Available acm V1.1

Artifacts Evaluated acm Functional V1.1

Results Reproduced acm V1.1

Unikraft

High performance POSIX unikernels are now a reality!

Info: https://unikraft.org/

Code: https://github.com/unikraft

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