\Orchestrating a brighter world NEC

Unikraft

Unikernels Made Easy

Simon Kuenzer <simon.kuenzer@neclab.eu>

Senior Researcher, NEC Laboratories Europe GmbH

FOSDEM’19

5GCity

5G ESSENCE

This work has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreements no. 675806 (“5G CITY”) and 761592 (“5G ESSENCE”). 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.

\Orchestrating a brighter world

NEC brings together and integrates technology and expertise to create
the ICT-enabled society of tomorrow.
We collaborate closely with partners and customers around the world,
orchestrating each project to ensure all its parts are fine-tuned to local needs.

Every day, our innovative solutions for society contribute to
greater safety, security, efficiency and equality,
and enable people to live brighter lives.

VMs vs. Containers

VMs have been around for a long time

- They allow consolidation, isolation, migration, …

Then containers came and many people LOVED them. Why?

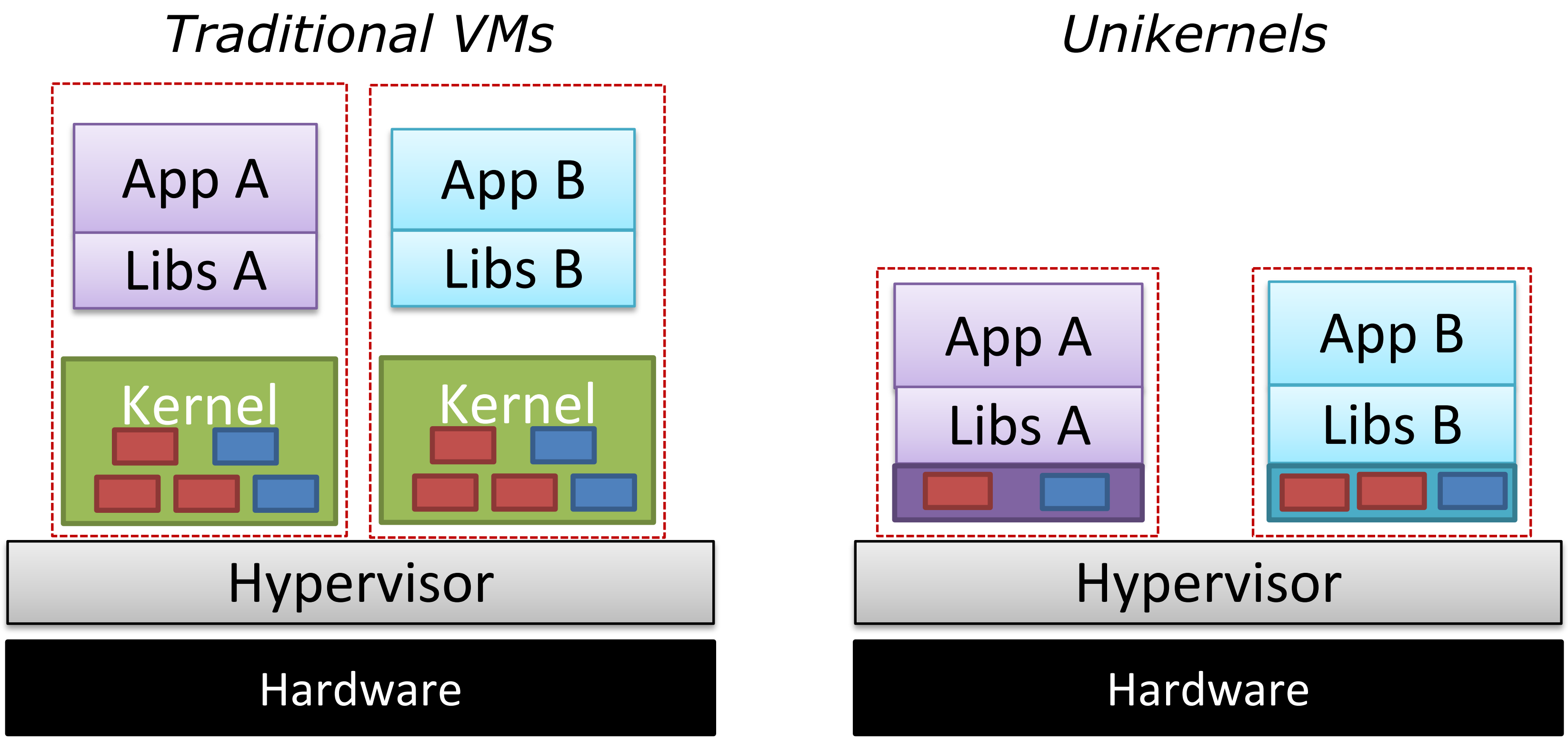
Containers are much easier to create
and deploy. I just write th
and I’r

maller. My VM
er only a few

Containers are much faster to bring up than
VM
co

Did you hear about Unikernels? VMs have they advantages, most importantly **strong isolation**.

Unikernels as VMs



- Unikernels are purpose-built
  - Thin kernel layer, *only what application needs*
  - Single monolithic binary *that contains OS* ***and*** *application*
- No isolation within Unikernel, done with hypervisor
  - One application → Flat and single address space
- Further advantages from specialization

Unikernel Gains

Fast instantiation, destruction and migration time

- 10s of milliseconds

Low memory footprint

- Few MB of RAM

High density

- 10k guests on a single server node

High Performance

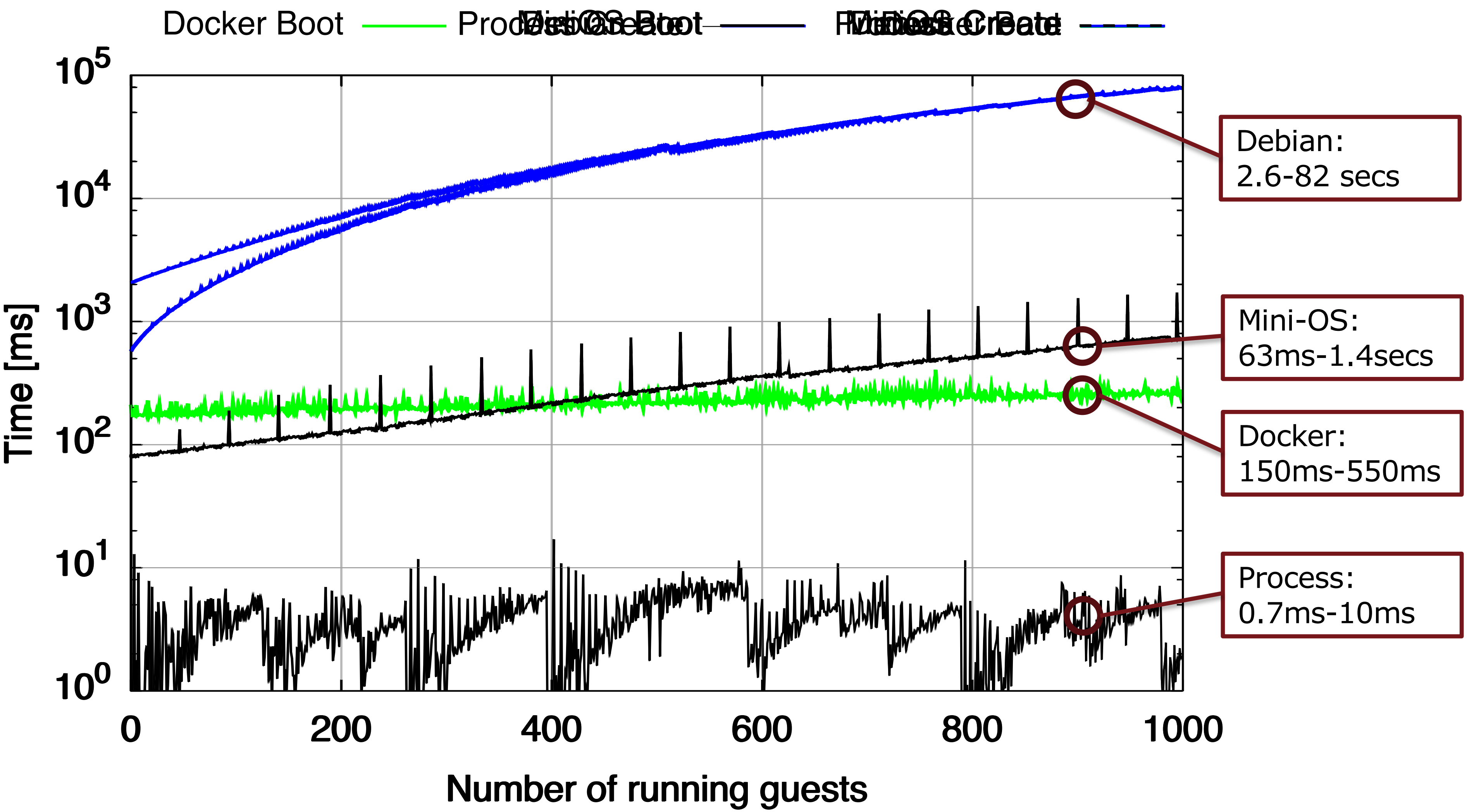
- 10-40Gbit/s throughput with a single guest CPU

Reduced attack surface

- Less components exist in Unikernel
- Strong isolation by hypervisor

LightVM [Manco SOSP 2017], Elastic CDNs [Kuenzer VEE 2017], Superfluid Cloud [Manco HotCloud 2015] , ClickOS [Martins NSDI 2014]

In Numbers: Instantiation Times



Debian:
2.6-82 secs

Mini-OS:
63ms-1.4secs

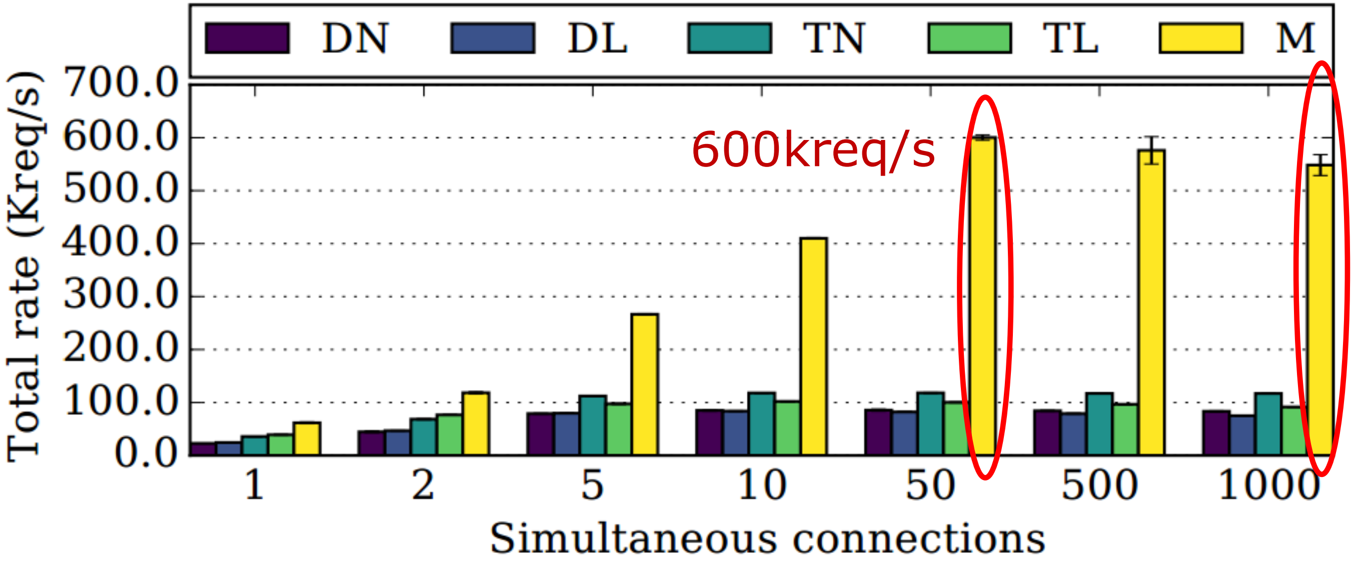
Docker:
150ms-550ms

Process:
0.7ms-10ms

Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

In Numbers: Performance

MiniCache Unikernel: Purpose-built static HTTP Webserver



32Gbit/s

<u>Legend:</u> MiniCache (M) versus Debian (D) and Tinyx (T). L=lighttpd and N=nginx.

Experiments were conducted on Intel Xeon E5 1630v3 3.7GHz, 32GB DDR4 RAM, Mellanox ConnectX-3 40Git/s Ethernet, Xen 4.4.2, Debian Jessie with Linux 4.0.0 as Dom0 and booted from RAM

Application Domains

Minimal SW Stack

Reactive vNFs,
Serverless,
Lambda,
etc.

Fast boot, migration destroy

Resource efficient

Minimal SW Stack

Serverless,
(Per-customer) vNFs,
IoT,
MEC,
etc.

Specialization

NFV,
MEC,
etc.

High performance

Mission critical

Small code base
→ Low attack surface
→ Cheaper verification

Automotive,
(Industrial) IoT,
etc.

The Devil is in the Details

So, Unikernels:

- Give similar speed and size of containers
- But add **strong isolation** with *virtualization* and increase **security** due to *smaller code base*

The problem is *Unikernel development:* Optimized Unikernels are manually built

- Building takes several months or even longer
  - *We've done it before, multiple times*
- Potentially repeat the process for each target application
  - *We've done that too…*

That's not an effective way of doing things!

Unikraft - A Unikernel Framework

Motivation

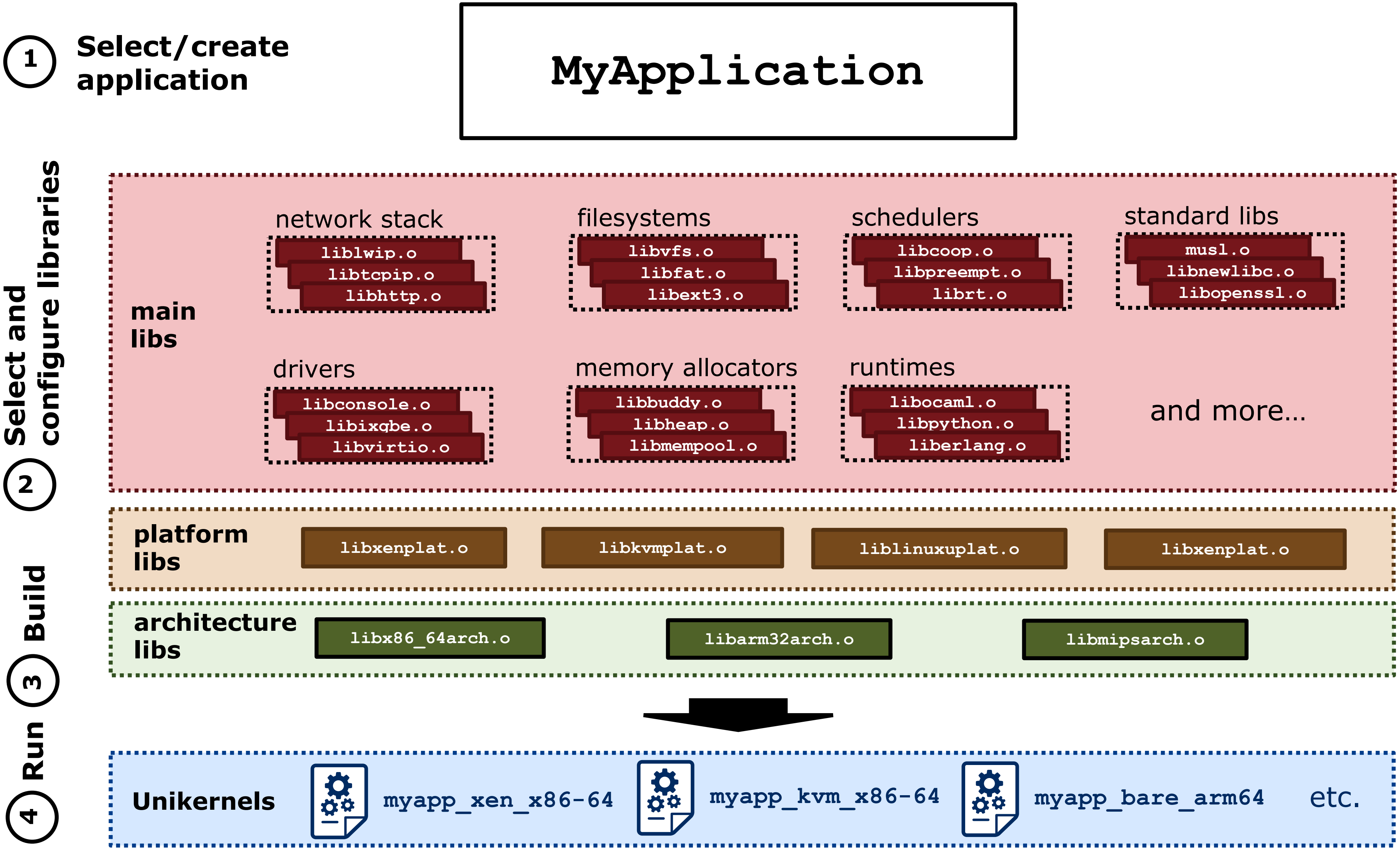
- Support wide range of use cases
- Simplify building and optimizing
- Simplify porting of existing applications
- Common and shared code base for Unikernel creators
- Support different hypervisors and CPU architectures

- Concept: “Everything is a library”
  - Decomposed OS functionality
- Two components:
  - Library Pool
  - Build Tool

Unikraft

Overview

1) Library Pool



1 Select/create application

MyApplication

2 Select and configure libraries

3 Build

4 Run

main libs

network stack
liblwip.o
libtcpip.o
libhttp.o

filesystems
libvfs.o
libfat.o
libext3.o

schedulers
libcoop.o
libpreempt.o
librt.o

standard libs
musl.o
libnewlibc.o
libopenssl.o

drivers
libconsole.o
libixgbe.o
libvirtio.o

memory allocators
libbuddy.o
libheap.o
libmempool.o

runtimes
libocaml.o
libpython.o
liberlang.o

and more…

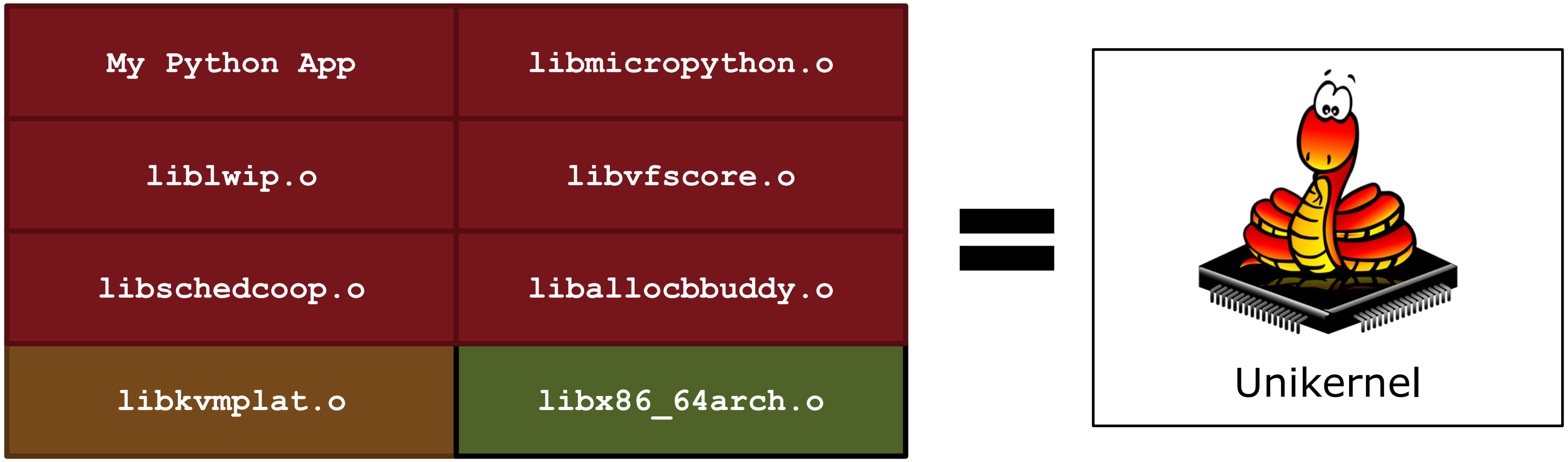
platform libs
libxenplat.o
libkvmplat.o
liblinuxuplat.o
libxenplat.o

architecture libs
libx86_64arch.o
libarm32arch.o
libmipsarch.o

Unikernels
myapp_xen_x86-64
myapp_kvm_x86-64
myapp_bare_arm64
etc.

Example System

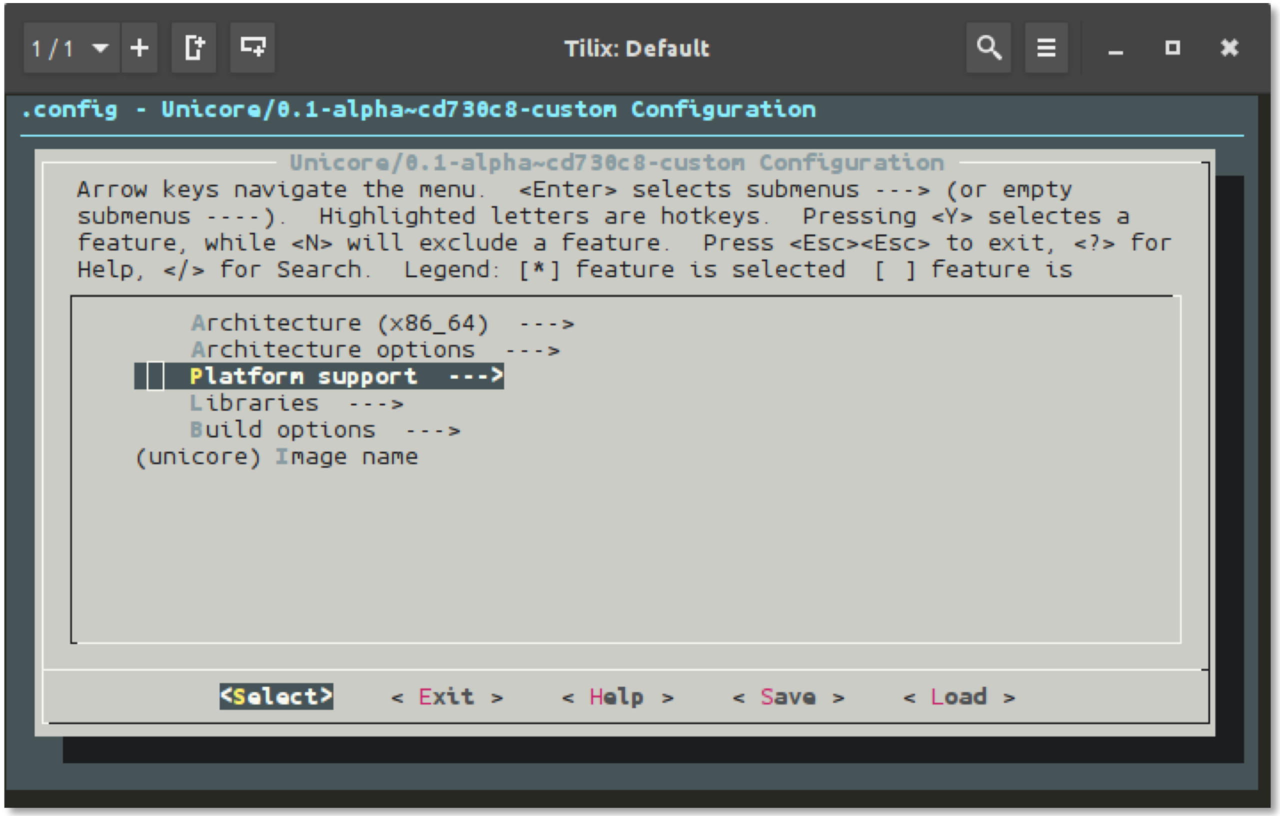
Python Unikernel for KVM on x86_64



My Python App	libmicropython.o
liblwip.o	libvfscore.o
libschedcoop.o	liballocbbuddy.o
libkvmplat.o	libx86_64arch.o

2) Build Tool

- KConfig based and Makefile “Magic”
- Type “`make menuconfig`”
  - Choose options in the menu that you want for your application
  - Choose your target platform(s), e.g., Xen, KVM, bare-metal, Linux
- Save your config and type “`make`”

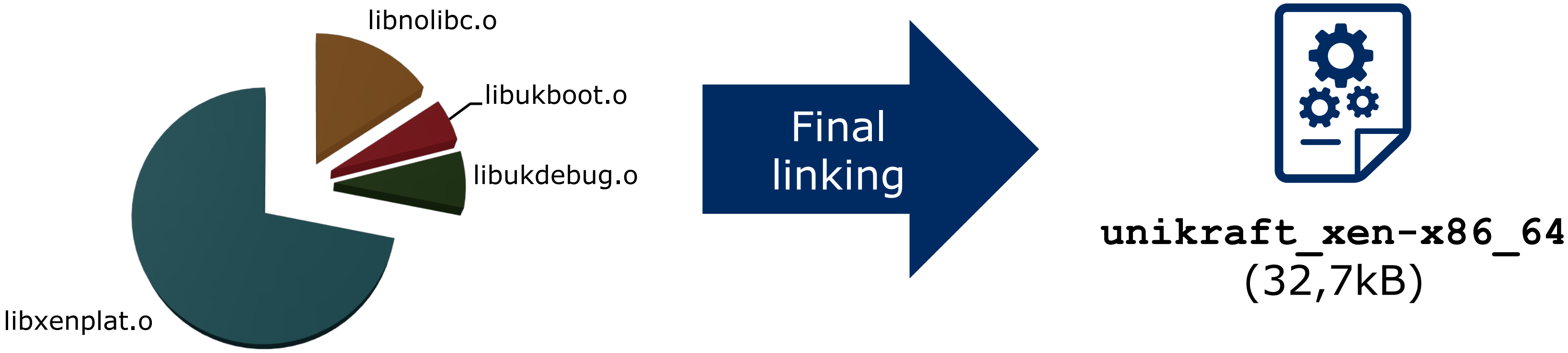


```
.config - Unicore/0.1-alpha~cd730c8-custom Configuration

                 Unicore/0.1-alpha~cd730c8-custom Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> selectes a
  feature, while <N> will exclude a feature.  Press <Esc><Esc> to exit, <?> for
  Help, </> for Search.  Legend: [*] feature is selected  [ ] feature is

            Architecture (x86_64)  --->
            Architecture options  --->
            Platform support  --->
            Libraries  --->
            Build options  --->
        (unicore) Image name

         <Select>    < Exit >    < Help >    < Save >    < Load >
```

An Baseline Example…

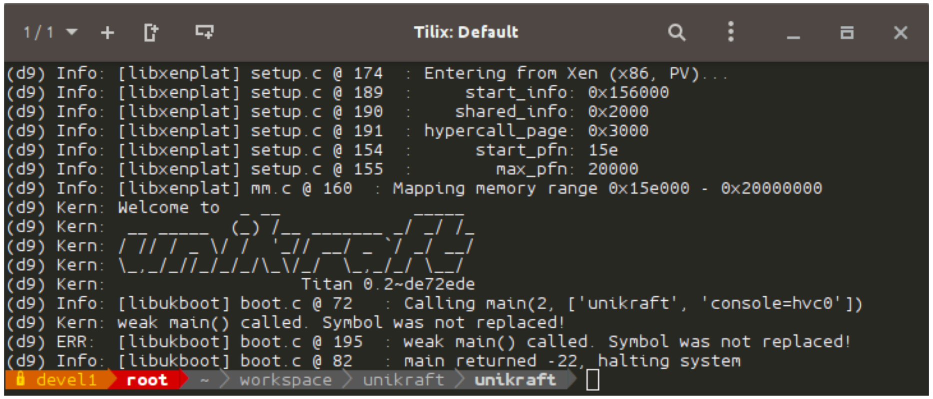
- *Xen PV x86_64 binary*
- Compiles to a 32.7kB image

`unikraft_xen-x86_64.o` (50,2kB)



`unikraft_xen-x86_64` (32,7kB)

- Boots and prints messages to debug console (with min. 208kB RAM)



```
(d9) Info: [libxenplat] setup.c @ 174  : Entering from Xen (x86, PV)...
(d9) Info: [libxenplat] setup.c @ 189  :     start_info: 0x156000
(d9) Info: [libxenplat] setup.c @ 190  :    shared_info: 0x2000
(d9) Info: [libxenplat] setup.c @ 191  : hypercall_page: 0x3000
(d9) Info: [libxenplat] setup.c @ 154  :      start_pfn: 15e
(d9) Info: [libxenplat] setup.c @ 155  :        max_pfn: 20000
(d9) Info: [libxenplat] mm.c @ 160  : Mapping memory range 0x15e000 - 0x20000000
(d9) Kern: Welcome to  _ __             _____
(d9) Kern:  __ _____  (_) /__ _______ _/ __/ /_
(d9) Kern: / // / _ \/ /  '_// __/ _ `/ _/ __/
(d9) Kern: \_,_/_//_/_/_/\_\/_/  \_,_/_/ \__/
(d9) Kern:                   Titan 0.2~de72ede
(d9) Info: [libukboot] boot.c @ 72   : Calling main(2, ['unikraft', 'console=hvc0'])
(d9) Kern: weak main() called. Symbol was not replaced!
(d9) ERR:  [libukboot] boot.c @ 195  : weak main() called. Symbol was not replaced!
(d9) Info: [libukboot] boot.c @ 82   : main returned -22, halting system
 devel1  root  ~  workspace  unikraft  unikraft  
```

Unikraft 0.3 Iapetus

Upcoming Release

Supported Features

Target support

- Xen: x86_64, Arm32
- KVM: x86_64, Arm64
- Linux userspace: x86_64, Arm32
- Bare-metal: x86_64 (with KVM target)

Core Functionality

- Cooperative scheduler
- Binary buddy heap

Networking

- Low-level API for high-speed I/O
  - virtio-net
- TCP/IP stack: Lightweight IP (lwIP)

Filesystems

- VFS

Libc's

- nolibc *(Unikraft internal)*
- Newlib

Roadmap

Concentrating effort on:

- Completing Arm64 support
  - (Virtual) Device drivers for Arm platforms
  - Other platforms
- More standard libraries
  - musl, libuv, zlib, openssl, libunwind, libaxtls (TLS), etc.
- Language environments
  - Javascript (v8), Python, Ruby, C++, etc.
- OCI container target support
- Filesystems
  - In-RAM and (Virtual) Disk filesystems
- Network drivers
  - Xen (netfront), Linux (tap)
- Frameworks:
  - Node.js, PyTorch, Intel DPDK, etc.

It is Open Source!

We need you!

Join Us!

- Unikraft is OpenSource since Dec 2017 and under the umbrella of

Xen Project   THE LINUX FOUNDATION

- Community is growing!
  - Active contributors rose 91%, from 2 contributors to 23.
- External contributors from
  - Romania: *networking, scheduling; from University Politechnica Bucharest*
  - Israel: *bare-metal support, VGA driver*
  - China: *Arm64 support from Arm*

...but there is still a lot to do!
Get in touch with us!

Drop us a mail
minios-devel@lists.xen.org
Join our IRC channel
#unikraft on Freenode

Example

Demo Time

Unikraft Resources

- Wiki
  - https://wiki.xenproject.org/ (Search for Unikraft)
- Documentation
  - http://www.unikraft.org
- Sources (GIT)
  - http://xenbits.xen.org/gitweb/ (Namespace: Unikraft)
- Mailing list (shared with Mini-OS)
  - minios-devel@lists.xen.org
- IRC Channel on Freenode
  - #unikraft
- NEC-Team
  - http://sysml.neclab.eu

\Orchestrating a brighter world

NEC