Building DPDK Unikernels with Unikraft

Sharan Santhanam

NEC Laboratories Europe GmbH

sharan.santhanam@neclab.eu

24th June 2019

UNICORE

This work has received funding from the European Unions Horizon 2020 research and innovation program under grant agreements no.825377 (UNICORE).This work reflects only the authors views and the European Commission is not responsible for any use that may be made of the information it contains.

Overview

1 What we saw

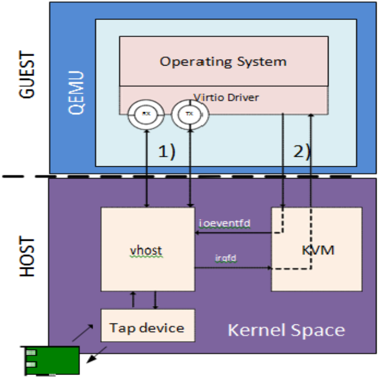
2 Introduce Unikraft

3 Unikraft meets DPDK

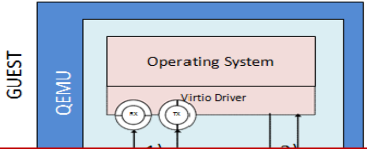
4 Unikraft within DPDK

5 Synergy between Unikraft and DPDK

VNF with DPDK Ecosystem



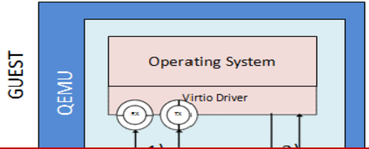
VNF with DPDK Ecosystem



Can we do better?

- ⇝ Guest OS specialization
- ⇝ Boot Time
- ⇝ Isolation within the guest

VNF with DPDK Ecosystem

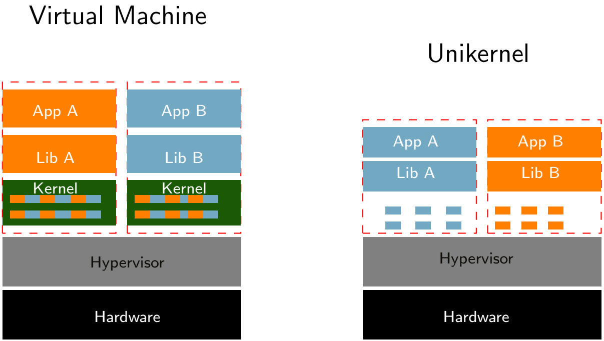


Can we do better?

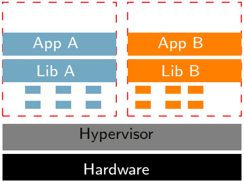
- ⇝ Guest OS specialization
- ⇝ Boot Time
- ⇝ Isolation within the guest

Let’s discuss Unikernel...

Unikernel - Do One Thing and Do It Well



Unikernel - Do One Thing and Do It Well

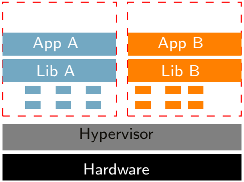


Unikernel

⇝ Unikraft are purpose built

- Thin kernel layer
- Single monolithic binary

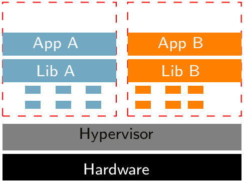
Unikernel - Do One Thing and Do It Well



Unikernel

- Unikraft are purpose built
  - Thin kernel layer
  - Single monolithic binary
- No isolation within the Unikernel needed
  - Flat address space

Unikernel - Do One Thing and Do It Well



Unikernel

- ⇝ Unikraft are purpose built
  - Thin kernel layer
  - Single monolithic binary
- ⇝ No isolation within the Unikernel needed
  - Flat address space
- ⇝ Full Stack Specialization

"Really Unikernels!"

⇝ Fast instantiation, destruction and migration times

- 10 milliseconds or less
  (LightVM [Manco SOSP 2017], Jitsu [Madhvapeddy, NSDI 2015])

”Really Unikernels!”

⇝ Fast instantiation, destruction and migration times

- 10 milliseconds or less
  (LightVM [Manco SOSP 2017], Jitsu [Madhvapeddy, NSDI 2015])

⇝ Low memory footprint

- Few MBs of RAM or less
  (ClickOS [Martins NSDI 2014])

"Really Unikernels!"

- ⇝ Fast instantiation, destruction and migration times
  - 10 milliseconds or less
    (LightVM [Manco SOSP 2017], Jitsu [Madhvapeddy, NSDI 2015])
- ⇝ Low memory footprint
  - Few MBs of RAM or less
    (ClickOS [Martins NSDI 2014])
- ⇝ High Deployment Density
  - 8k guests on a single x86 server
    (LightVM [Manco SOSP 2017])

"Really Unikernels!"

- Fast instantiation, destruction and migration times
  - 10 milliseconds or less
    (LightVM [Manco SOSP 2017], Jitsu [Madhvapeddy, NSDI 2015])
- Low memory footprint
  - Few MBs of RAM or less
    (ClickOS [Martins NSDI 2014])
- High Deployment Density
  - 8k guests on a single x86 server
    (LightVM [Manco SOSP 2017])
- High Performance
  - 10-40Gbit/s Ethernet throughput with a single guest CPU
    (ClickOS [Martins NSDI 2014], Elastic CDNs [Kuenzer VEE 2017])

"Really Unikernels!"

- ⇝ Fast instantiation, destruction and migration times
  - 10 milliseconds or less
    (LightVM [Manco SOSP 2017], Jitsu [Madhvapeddy, NSDI 2015])
- ⇝ Low memory footprint
  - Few MBs of RAM or less
    (ClickOS [Martins NSDI 2014])
- ⇝ High Deployment Density
  - 8k guests on a single x86 server
    (LightVM [Manco SOSP 2017])
- ⇝ High Performance
  - 10-40Gbit/s Ethernet throughput with a single guest CPU
    (ClickOS [Martins NSDI 2014], Elastic CDNs [Kuenzer VEE 2017])
- ⇝ Reduced attack surface
  - Small trusted compute base
  - Strong isolation by hypervisor

La-la Land

So, Unikernel

- High Performance
- Isolation and reduced attack surface.
- Faster Instantiation Time
- Smaller image size

La-la Land

So, Unikernel

- 👍 High Performance
- 👍 Isolation and reduced attack surface.
- 👍 Faster Instantiation Time
- 👍 Smaller image size

The problem with Unikernel development:

- 👎 Building take several months or longer
- 👎 Potentially repeat the process for each target application
- 👎 **"Specialization" is hard to build**

☹ **Ooops!!**

Thats not an effective way of doing things!

What is Unikraft?

Objectives

- ⇝ Support wide range of use cases
- ⇝ Simplify building and optimizing
- ⇝ Common and shared code base
- ⇝ Support different hypervisors
- ⇝ CPU architectures

What is Unikraft?

Objectives

- Support wide range of use cases
- Simplify building and optimizing
- Common and shared code base
- Support different hypervisors
- CPU architectures

Unikraft

- Everything is a library
- Decomposed OS functionality
- Unikrafts two components:
  - Library Pool
  - Build Tool

What is Unikraft?

Objectives

- Support wide range of use cases
- Simplify building and optimizing
- Common and shared code base
- Support different hypervisors
- CPU architectures

→

Unikraft

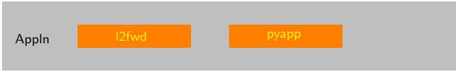
- Everything is a library
- Decomposed OS functionality
- Unikrafts two components:
  - Library Pool
  - Build Tool

☺ **Unikraft says Hi!!**

Source is BSD-licensed

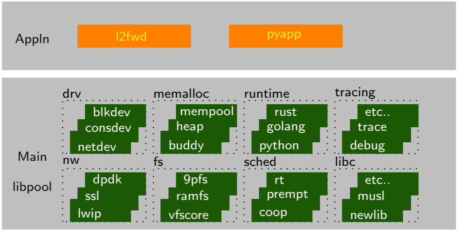
Kconfig based build system

libukforest - Unikraft System Overview



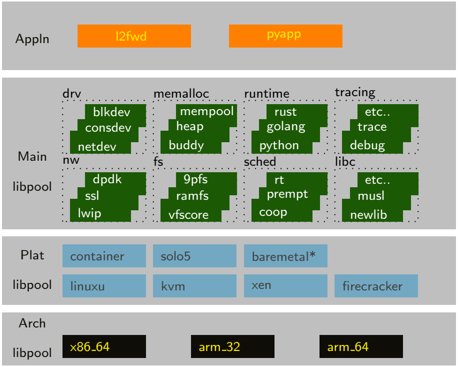
- Take an existing application
  - For example, a Python application or a l2fwd

libukforest - Unikraft System Overview



- Take an existing application
  - For example, a Python application or a l2fwd
- Pick Unikraft functionality
  - Pool of drivers and standard libraries

libukforest - Unikraft System Overview



Appln: l2fwd, pyapp

Main libpool:

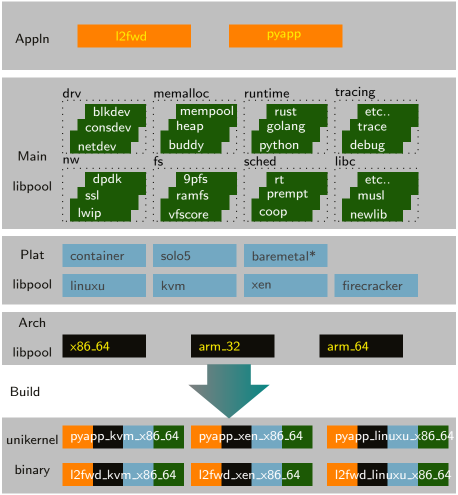
drv: blkdev, consdev, netdev
memalloc: mempool, heap, buddy
runtime: rust, golang, python
tracing: etc.., trace, debug
nw: dpdk, ssl, lwip
fs: 9pfs, ramfs, vfscore
sched: rt, prempt, coop
libc: etc.., musl, newlib

Plat libpool: container, solo5, baremetal*, linuxu, kvm, xen, firecracker

Arch libpool: x86_64, arm_32, arm_64

- ⇝ Take an existing application
  - For example, a Python application or a l2fwd
- ⇝ Pick Unikraft functionality
  - Pool of drivers and standard libraries
- ⇝ Pick a platform and architecture
  - Pool of drivers and standard libraries

libukforest - Unikraft System Overview



Appln

l2fwd

pyapp

Main libpool

drv: blkdev, consdev, netdev

memalloc: mempool, heap, buddy

runtime: rust, golang, python

tracing: etc.., trace, debug

nw: dpdk, ssl, lwip

fs: 9pfs, ramfs, vfscore

sched: rt, prempt, coop

libc: etc.., musl, newlib

Plat libpool

container, solo5, baremetal*

linuxu, kvm, xen, firecracker

Arch libpool

x86_64, arm_32, arm_64

Build

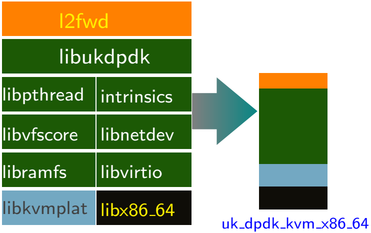
unikernel binary

pyapp_kvm_x86_64, pyapp_xen_x86_64, pyapp_linuxu_x86_64

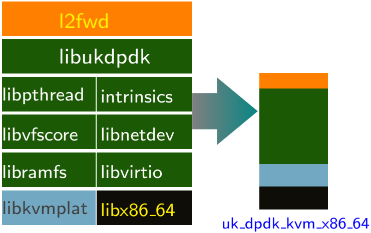
l2fwd_kvm_x86_64, l2fwd_xen_x86_64, l2fwd_linuxu_x86_64

- ⇝ Take an existing application
  - For example, a Python application or a l2fwd
- ⇝ Pick Unikraft functionality
  - Pool of drivers and standard libraries
- ⇝ Pick a platform and architecture
  - Pool of drivers and standard libraries
- ⇝ Build Unikraft application

Unikraft - DPDK Target Arch?



Unikraft - DPDK Target Arch?



Challenges!!

- Build System Integration
- Specialization of Guest OS
- Minimize modification to DPDK library

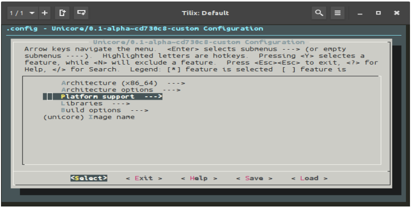
Build DPDK as an Unikraft Library

Unikraft Build system

- Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function

DPDK Build System

- Automatic config generation
  - CPU feature flags



```
.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 >
```

Build DPDK as an Unikraft Library

Unikraft Build system

- ⇝ Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function
- ⇝ Makefile.uk (make based)
  - [LIBNAME]_SRCS
  - [LIBNAME]_CFLAG
  - CFLAG

DPDK Build System

- ⇝ Automatic config generation
  - CPU feature flags
- ⇝ Makefile (gmake)
  - SRCS
  - INCLUDE
  - CFLAG
  - DIRS

Build DPDK as an Unikraft Library

Unikraft Build system

- ⇝ Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function
- ⇝ Makefile.uk (make based)
  - [LIBNAME]_SRCS
  - [LIBNAME]_CFLAG
  - CFLAG
- ⇝ exportsyms.uk

DPDK Build System

- ⇝ Automatic config generation
  - CPU feature flags
- ⇝ Makefile (gmake)
  - SRCS
  - INCLUDE
  - CFLAG
  - DIRS
- ⇝ version map

Build DPDK as an Unikraft Library

Unikraft Build system

- ⇝ Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function
- ⇝ Makefile.uk (make based)
  - [LIBNAME]_SRCS
  - [LIBNAME]_CFLAG
  - CFLAG
- ⇝ exportsyms.uk

DPDK Build System

- ⇝ Automatic config generation
  - CPU feature flags
- ⇝ Makefile (gmake)
  - SRCS
  - INCLUDE
  - CFLAG
  - DIRS
- ⇝ version map

libukdpdkbuild

- ⇝ Process DPDK Makefile.

Build DPDK as an Unikraft Library

Unikraft Build system

- ⇝ Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function
- ⇝ Makefile.uk (make based)
  - [LIBNAME]_SRCS
  - [LIBNAME]_CFLAG
  - CFLAG
- ⇝ exportsyms.uk

DPDK Build System

- ⇝ Automatic config generation
  - CPU feature flags
- ⇝ Makefile (gmake)
  - SRCS
  - INCLUDE
  - CFLAG
  - DIRS
- ⇝ version map

libukdpdkbuild

- ⇝ Process DPDK Makefile.
  - Add DPDK library

Build DPDK as an Unikraft Library

Unikraft Build system

- ⇝ Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function
- ⇝ Makefile.uk (make based)
  - [LIBNAME]_SRCS
  - [LIBNAME]_CFLAG
  - CFLAG
- ⇝ exportsyms.uk

DPDK Build System

- ⇝ Automatic config generation
  - CPU feature flags
- ⇝ Makefile (gmake)
  - SRCS
  - INCLUDE
  - CFLAG
  - DIRS
- ⇝ version map

libukdpdkbuild

- ⇝ Process DPDK Makefile.
  - 👍 Add DPDK library
  - 👍 Support newer version of DPDK

Build DPDK as an Unikraft Library

Unikraft Build system

- ⇝ Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function
- ⇝ Makefile.uk (make based)
  - [LIBNAME]_SRCS
  - [LIBNAME]_CFLAG
  - CFLAG
- ⇝ exportsyms.uk

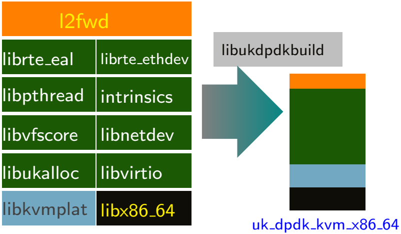
DPDK Build System

- ⇝ Automatic config generation
  - CPU feature flags
- ⇝ Makefile (gmake)
  - SRCS
  - INCLUDE
  - CFLAG
  - DIRS
- ⇝ version map

libukdpdkbuild

- ⇝ Process DPDK Makefile.
  - 👍 Add DPDK library
  - 👍 Support newer version of DPDK
- ⇝ Add dpdk specific configuration file.

Unikraft - DPDK Target Arch



Unikraft as EAL in DPDK*

Specialize the Guest OS

- ⇝ Memory management
- ⇝ Bus/Device Management
- ⇝ CPU Scheduling and CPU Features

Unikraft as EAL in DPDK*

Specialize the Guest OS

- ⇝ Memory management
- ⇝ Bus/Device Management
- ⇝ CPU Scheduling and CPU Features

Memory Management

- 👍 Unikraft: flat page table since boot
- 👍 Huge pages based 2MB sized pages
- 👍 Memory region can be explicitly assigned to the Application
- 👍 Custom memory allocator per memory region

Unikraft as EAL in DPDK*

Specialize the Guest OS

- ⇝ Memory management
- ⇝ Bus/Device Management
- ⇝ CPU Scheduling and CPU Features

Bus/Device Management

- 👍 A simpler bus/device interface
- 👍 Directly attached device and usable by DPDK with unikraft

Unikraft as EAL in DPDK*

Specialize the Guest OS

- ⇝ Memory management
- ⇝ Bus/Device Management
- ⇝ CPU Scheduling and CPU Features

CPU Scheduling and CPU Features

- 👍 Application decides on scheduling on the core.
- 👍 Minimal interference / resource usage for other purpose within guest.

Future Work

- ⇝ Add SMP support
- ⇝ Add NUMA support
- ⇝ Evaluate performance
- ⇝ Memory allocation natively by DPDK
- ⇝ Use DPDK drivers directly

Join us!

Wiki

https://wiki.xenproject.org

Documentation

http://www.unikraft.org

Sources

http://xenbits.xen.org/gitweb/ (Namespace: Unikraft)

Mailing List

minios-devel mailing list

NEC Maintainer Team

NEC Unikraft Team

What we think

Unikraft

- 👍 Support multiple platforms
- 👍 Specialized Guest OS
- 👍 Simpler Management Device
- 👍 Increased control for an application

DPDK

- 👍 Performance of Network stack
- 👍 Specialized VNF
- 👍 Wealth of knowledge DPDK driver
- 👍 Increased application base

What we think

Unikraft

- Support multiple platforms
- Specialized Guest OS
- Simpler Management Device
- Increased control for an application

DPDK

- Performance of Network stack
- Specialized VNF
- Wealth of knowledge DPDK driver
- Increased application base

☺ **What do you think?**