Extremely Fast and Efficient NFV with Unikraft

Sharan Santhanam
Felipe Huici

NEC Laboratories Europe GmbH

29th Oct 2020

This work has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreements no.825377 (“UNICORE”). 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.
Overview

1. What we saw
2. Introduce Unikraft
3. Unikraft meets DPDK
4. Unikraft within DPDK
5. Performance Evaluation
6. Synergy between Unikraft and DPDK
Unikraft NFV

Sharan Santhanam
Felipe Huici

What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Performance Evaluation

Synergy between Unikraft and DPDK

VNF with DPDK Ecosystem

Can we do better?

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

Let's discuss Unikernel...
Can we do better?

⇝ Guest OS specialization
⇝ Boot Time
⇝ Isolation within the guest
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Performance Evaluation

Synergy between Unikraft and DPDK

VNF with DPDK Ecosystem

Can we do better?

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

Let’s discuss Unikernel...
Virtual Machine

Unikraft - Do One Thing and Do It Well

what we saw
Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Performance Evaluation
Synergy between Unikraft and DPDK
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Performance Evaluation

Synergy between Unikraft and DPDK

Unikernal - Do One Thing and Do It Well

Unikernal are purpose built

- Thin kernel layer
- Single image with application, specific kernel primitives
Unikernel - Do One Thing and Do It Well

What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Performance Evaluation

Synergy between Unikraft and DPDK

Unikernel are purpose built

- Thin kernel layer
- Single image with application, specific kernel primitives

No isolation within a Unikernel

- Flat address space

Unikernel are purpose built

- Thin kernel layer
- Single image with application, specific kernel primitives

No isolation within a Unikernel

- Flat address space
Unikernel - Do One Thing and Do It Well

Unikernel are purpose built
- Thin kernel layer
- Single image with application, specific kernel primitives

No isolation within a Unikernel
- Flat address space

Software 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
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

智能制造

Ooops!!
That’s 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
- Unikraft’s 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
- Unikraft’s two components:
  - Library Pool
  - Build Tool

**Unikraft says Hi!!**

Source is BSD-licensed

Kconfig based build system
Take an existing application

- For example, a Python application or a l2fwd
- **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

- 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**

<table>
<thead>
<tr>
<th>Appln</th>
<th>l2fwd</th>
<th>pypipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>drv</td>
<td>blkdev</td>
<td>mempool</td>
</tr>
<tr>
<td></td>
<td>console</td>
<td>heap</td>
</tr>
<tr>
<td></td>
<td>netdev</td>
<td>buddy</td>
</tr>
<tr>
<td></td>
<td>fs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Main</th>
<th>nw</th>
<th>fs</th>
<th>sched</th>
<th>libc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dpdk</td>
<td>mempool</td>
<td>rust</td>
<td>tracing</td>
</tr>
<tr>
<td></td>
<td>sal</td>
<td>heap</td>
<td>golang</td>
<td>etc.</td>
</tr>
<tr>
<td></td>
<td>lwp</td>
<td>buddy</td>
<td>python</td>
<td>trace</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>debug</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plat</th>
<th>container</th>
<th>solo5</th>
<th>baremetal*</th>
<th>firecracker</th>
</tr>
</thead>
<tbody>
<tr>
<td>libpool</td>
<td>linuxu</td>
<td>kvm</td>
<td>xen</td>
<td>firecracker</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arch</th>
<th>x86_64</th>
<th>arm_32</th>
<th>arm_64</th>
</tr>
</thead>
<tbody>
<tr>
<td>libpool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **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**
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?

What we saw
Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Performance Evaluation
Synergy between Unikraft and DPDK

Unikraft - DPDK Target Arch?

- l2fwd
- libukdpdk
- libpthread
- libvfscore
- libramfs
- libkvmplat
- uk_dpdk_kvm_x86_64
- libintrinsics
- libnetdev
- libvirtio
- libx86_64
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

Unikraft Build system

- Config.uk (Kconfig based)
  - Handles dependencies across library
  - Enable/Disable Function
### Build DPDK as an Unikraft Library

#### Unikraft Build system

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config.uk (Kconfig based)</td>
<td>Handles dependencies across library&lt;br&gt;Enable/Disable Function</td>
</tr>
<tr>
<td>Makefile.uk (make based)</td>
<td><img src="images" alt="Images" /></td>
</tr>
</tbody>
</table>
  - [LIBNAME].SRCS
  - [LIBNAME].CFLAG
  - CFLAG

#### DPDK Build System

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic config generation</td>
<td>CPU feature flags</td>
</tr>
<tr>
<td>Makefile (gmake)</td>
<td><img src="images" alt="Images" /></td>
</tr>
</tbody>
</table>
  - 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.
```
# 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
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

What we saw
Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Performance Evaluation
Synergy between Unikraft and 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
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.
Unikraft NFV
Sharan Santhanam
Felipe Huici

What we saw
Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Performance Evaluation
Synergy between Unikraft and DPDK

KVM Summit 2020 | © NEC Corporation 2020
Interface between Unikraft and DPDK

What we saw
Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Performance Evaluation
Synergy between Unikraft and DPDK

Unikraft NFV
Sharan Santhanam
Felipe Huici

librte_ethdev
app_rx_burst
rte_eth_rx_burst
virtio_recv_pkts_packed

libuknetdev
app_rx_burst
uk_netdev_rx_one
virtio_netdev_recv
Interface between Unikraft and DPDK

Unikraft NFV
Sharan Santhanam
Felipe Huici

What we saw
Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Performance Evaluation
Synergy between Unikraft and DPDK
Unikraft NFV

Sharan Santhanam
Felipe Huici

What we saw
Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Performance Evaluation
Synergy between Unikraft and DPDK

Interface between Unikraft and DPDK
Test Setup

Unikraft/Linux GuestVM Setup

- CPU Family: Sandy Bridge (Server) (Family: 6 and Model: 45)
- CPU Model: Intel(R) Xeon(R) CPU E5-1650 0 @ 3.20GHz
- Nr of Cores: 6
- RAM: 16GB
- Nr of NUMA nodes: 1
- Host Linux Kernel: Debian 4.19
- Guest Linux Kernel: Debian 4.19
- Qemu Version: 4.0.0
- DPDK Version (vhost-user): 19.08

Packet Generator/Receiver Setup

- CPU Family: Ivy Bridge (Server) (Family: 6 and Model: 62)
- CPU Model: Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz
- Nr of Cores: 4
- RAM: 16GB
- Nr of NUMA nodes: 1
- Host Linux Kernel: Debian 4.19
- DPDK Version (vhost-user): 19.08
Packet Send: Unikraft vs LinuxVM

TX throughput Comparison between Unikraft and Linux VM

- LinuxVM DPDK with vhost-net
- LinuxVM DPDK with vhost-user
- Unikraft with vhost-net
- Unikraft with vhost-user
- Unikraft with Vhost-user

Packet Size (Bytes)

Throughput (Mpps)
Packet Receive: Unikraft vs LinuxVM

RX throughput Comparison between Unikraft and Linux VM

Throughput (Mp/s) vs Packet Size (Bytes)

- Linux VM with vhost-net
- Linux VM with vhost-user
- Unikraft with vhost-net
- Unikraft with vhost-user
### Key Value: Unikraft vs LinuxVM

<table>
<thead>
<tr>
<th></th>
<th>Linux Guest VM</th>
<th>Unikraft using DPDK(mbuf)</th>
<th>Unikraft using Netdev(netbuf)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.1Mp/s</td>
<td>6.1Mp/s</td>
<td>6.0Mp/s</td>
</tr>
</tbody>
</table>
## Resource Usage: Unikraft vs LinuxVM

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Unikraft</th>
<th>Linux VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>1GB</td>
<td>6GB</td>
</tr>
<tr>
<td>Boot Times</td>
<td>87ms</td>
<td>12s</td>
</tr>
<tr>
<td>Image Size</td>
<td>1.4MB</td>
<td>2.5GB</td>
</tr>
</tbody>
</table>
Future Work

- Add SMP support
- Add NUMA support
- Use DPDK drivers directly
Join us!

Project Home Page
http://unikraft.org

Documentation
http://docs.unikraft.org/

Sources
https://github.com/unikraft
http://xenbits.xen.org/gitweb/ (Namespace: Unikraft)
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

<table>
<thead>
<tr>
<th>Unikraft</th>
<th>DPDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟 Support multiple platforms</td>
<td>🌟 Performance of Network stack</td>
</tr>
<tr>
<td>🌟 Specialized Guest OS</td>
<td>🌟 Specialized VNF</td>
</tr>
<tr>
<td>🌟 Simpler Management Device</td>
<td>🌟 Wealth of knowledge DPDK driver</td>
</tr>
<tr>
<td>🌟 Increased control for an application</td>
<td>🌟 Increased application base</td>
</tr>
</tbody>
</table>

What do you think?