Building DPDK Unikernels with Unikraft

Sharan Santhanam

NEC Laboratories Europe GmbH

sharan.santhanam@neclab.eu

24th June 2019

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
Unikraft
DPDK
Sharan
Santhanam

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

Can we do better?
⇝
Guest OS specialization
⇝
Boot Time
⇝
Isolation within the guest

Let’s discuss Unikernel...

VNF with DPDK Ecosystem
VNF with DPDK Ecosystem

Can we do better?

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

Introduce Unikraft

Unikraft meets DPDK

Synergy between Unikraft and DPDK

Can we do better?

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

Let’s discuss Unikernel...
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK

Virtual Machine

Unikernel
Unikraft are purpose built
- Thin kernel layer
- Single monolithic binary
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK

Unikraft are purpose built
- Thin kernel layer
- Single monolithic binary

No isolation within the Unikernel needed
- Flat address space
What we saw

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
So, Unikernel

- High Performance
- Isolation and reduced attack surface.
- Faster Instantiation Time
- Smaller image size
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!!
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
- 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
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK
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
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK

→ 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
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK

```
<table>
<thead>
<tr>
<th>Appln</th>
<th>l2fwd</th>
<th>pyapp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>drv</td>
<td>memalloc</td>
</tr>
<tr>
<td>nw</td>
<td>blkdev</td>
<td>mempool</td>
</tr>
<tr>
<td>fs</td>
<td>heap</td>
<td>buddy</td>
</tr>
<tr>
<td>sched</td>
<td>9pfs</td>
<td>ramfs</td>
</tr>
<tr>
<td></td>
<td>lwip</td>
<td>vfscore</td>
</tr>
<tr>
<td></td>
<td>dpdk</td>
<td></td>
</tr>
<tr>
<td>libpool</td>
<td>ssl</td>
<td></td>
</tr>
<tr>
<td>nw</td>
<td>l2fwd</td>
<td></td>
</tr>
<tr>
<td>libpool</td>
<td>linuxu</td>
<td>kvm</td>
</tr>
<tr>
<td>Plat</td>
<td>container</td>
<td>solo5</td>
</tr>
<tr>
<td>libpool</td>
<td>linuxu</td>
<td>kvm</td>
</tr>
<tr>
<td>Arch</td>
<td>x86_64</td>
<td>arm_32</td>
</tr>
<tr>
<td>libpool</td>
<td>x86_64</td>
<td>arm_32</td>
</tr>
<tr>
<td>Build</td>
<td>pyapp_kvm_x86_64</td>
<td>pyapp_xen_x86_64</td>
</tr>
<tr>
<td>uni kernel</td>
<td>pyapp_kvm_x86_64</td>
<td>pyapp_xen_x86_64</td>
</tr>
<tr>
<td>binary</td>
<td>l2fwd_kvm_x86_64</td>
<td>l2fwd_xen_x86_64</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)**
What we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK

Unikraft - DPDK Target Arch?

l2fwd
libukdpdk
libpthread intrinsics
libvfscore libnetdev
libramfs libvirtio
libkvmplat libx86_64

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

Challenges!!
Unikraft - DPDK Target Arch?

- Unikraft
- DPDK
- What we saw
- Introduce Unikraft
- Unikraft meets DPDK
- Unikraft within DPDK
- Synergy between Unikraft and DPDK

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

- **Makefile.uk (make based)**
  - `[LIBNAME].SRCS`
  - `[LIBNAME].CFLAG`
  - `CFLAG`

DPDK Build System

- **Automatic config generation**
  - CPU feature flags

- **Makefile (gmake)**
  - `SRCS`
  - `INCLUDE`
  - `CFLAG`
  - `DIRS`
What we saw

Introduction

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK

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
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
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
Synergy between Unikraft and DPDK
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 saw

Introduce Unikraft
Unikraft meets DPDK
Unikraft within DPDK
Synergy between Unikraft and DPDK

### What we think

<table>
<thead>
<tr>
<th><strong>Unikraft</strong></th>
<th><strong>DPDK</strong></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 we saw

Introduce Unikraft

Unikraft meets DPDK

Unikraft within DPDK

Synergy between Unikraft and DPDK

### 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?