Agenda

• Unikernel concept and benefits
• What obstructs wide adoption of Unikernels
• The first revolution
• Unikraft brings the second revolution
• Features supported on Arm
• The gaps on Arm
• Summary
Unikernel basic concept

Unikernels are specialized, single address space images
- Constructed by a special collection of needed libraries.
  - Just enough, no more
  - Can be deployed directly as minimal dependencies
Unikernel benefits and use cases

Benefits

- No Operating System
  - Shorten the distance between hardware and software
  - Function call instead of system call
  - High deterministic performance
- Contains only needed libraries
  - Tiny footprint
  - Small attack surface
  - boots fast

Use cases

- Cloud application
  - Employed as a special kind of Container
- Host services
  - MirageOS has been used in Hyperkit, VPNkit and Datakit as Container host services like DHCP.
- IoT devices
  - Camera, Vehicle and other IoT things
- HPC
  - Scalable and predictable runtime behavior - HermitCore
What obstructs wide adoption of Unikernel?

Removing Operating System brings several drawbacks:

• Resource isolations for multiple Unikernels
  – No process to protect jobs’ contexts
  – No scheduler to arrange jobs

• Rewriting massive libraries for rapidly changing hardware
  – Unable to re-use existing device drivers in OS

• Start from scratch to create an Unikernel application
  – No as many libraries as OS can provide
  – Can’t re-use previous research and development easily
Fortunately, modern hypervisors provide virtual machines with:

- **Strong context isolation**
  - Hardware isolation for resources

- **Schedulers**
  - Arrange jobs as scheduling Virtual Machine

- **Consistent set of virtual devices**
  - Consistent set of libraries for virtual devices
It’s still not easy to create Unikernels

However, in most cases, we still need to re-write almost all the existing applications for Unikernels:

- Porting manually
- Consuming lots of time
- Introducing mistakes easily
- Hard to re-use existing researches and developments
Unikraft, introduced by NEC Laboratories Europe, is a development model – it’s an SDK to:

- Reduce the effort of converting existing applications to Unikernels by
  - Reusing existing research and development
  - Configuring easily
  - Porting effort requires no rewriting
    - Recompiling application code in the best case
    - Small changes to the actual application code in the worst
Two important components of Unikraft

- **Library pool**
  - Architecture libraries
  - Platform libraries
  - OS functional libraries
  - Standard and runtime libraries

- **Build toolchain**
  - Provides a Linux style kconfig menu
  - Provides scripts to integrate/reuse existed library
  - Generates binaries for multiple platforms automatically
Internal libraries and external libraries

- Internal libraries
  - Memory allocators,
  - Schedulers,
  - Debug and boot
- External libraries
  - Lwip,
  - Newlib,
  - Shell.
Internal libraries VS External libraries

Internal libraries are no different than external ones, except for the fact that:

• They are part of the main Unikraft repository,

-rw-r--r-- 1 root root 4096 May 13 10:06 arch
-rw-r--r-- 1 root root 1880 May 13 10:06 CODING_STYLE.md
-rw-r--r-- 1 root root 4245 May 13 10:06 Config.uk
-rw-r--r-- 1 root root 7216 May 13 10:06 CONTRIBUTING.md
-rw-r--r-- 1 root root 21352 May 13 10:06 COPYING.md
drwxr-xr-x 3 root root 4096 May 13 10:06 doc
drwxr-xr-x 3 root root 4096 May 13 10:06 include
drwxr-xr-x 11 root root 4096 May 13 10:06 lib
-rw-r--r-- 1 root root 4899 May 13 10:06 MAINTAINERS.md
-rw-r--r-- 1 root root 25497 May 13 10:06 Makefile
-drwxr-xr-x 5 root root 4096 May 13 10:06 plat
-rw-r--r-- 1 root root 2111 May 13 10:06 README.md
drwxr-xr-x 5 root root 4096 May 13 10:06 support
-rw-r--r-- 1 root root 139 May 13 10:06 version.mk

• They do not use any external source files,

• They must not have dependencies on external libraries.
Creating Unikernels with toolchain

Using toolchain to port or develop an application to Unikraft:

Create 3 Basic Files

#1. Makefile

```make
UK_ROOT := $(PWD)/../unikraft
UK_LIBS := $(PWD)/../unikraft-lib
LIBS := $(UK_LIBS)/lib1:$(UK_LIBS)/lib2:$(UK_LIBS)/libN

all:
  @make -C $(UK_ROOT) A=$(PWD) L=$(LIBS)

$(MAKECMDGOALS):
  @make -C $(UK_ROOT) A=$(PWD) L=$(LIBS) $$(MAKECMDGOALS)
```
Creating Unikernels with toolchain

Using toolchain to port or develop an application to Unikraft:

Create 3 Basic Files

#1. Makefile

#2. Makefile.uk

#3. Makefile.uk
Creating Unikernels with toolchain

Using toolchain to port or develop an application to Unikraft:

Create 3 Basic Files

#1. Makefile
#2. Makefile.uk
#3. Config.uk

```bash
### Invisible option for dependencies
config APPHELLOWORLD_DEPENDENCIES
  bool
default y
select LIBNOLIBC if !HAVE_LIBC

### App configuration
config APPHELLOWORLD_PRINTARGS
  bool "Print arguments"
default y
help
  Prints argument list (argv) to stdout
```
Creating Unikernels with toolchain

Using toolchain to port or develop an application to Unikraft:

Create 3 Basic Files

1. Makefile
2. Makefile.uk
3. Config.uk
Decomposing python for example

- Creating Makefile and Makefile.uk to tell Unikraft how to build python source files,
- Using Config.uk to populate the kconfig menu and select libraries from it,
- Build and Run.
Developing an external library

Developing or porting an external library isn't too different from porting an application

- No Makefile is needed for external libraries
- Makefile.uk follows the same format of application
- One difference relates to Config.uk
  - You surround your settings with ```menuconfig``` that enables selecting and deselecting the library.
Arm support status

The initial version of Unikraft came with Arm32 supports:
What we have done on Arm

Now we are working on the Arm64 Linux-KVM / QEMU, we have (not yet merged:)

- Improved multi-arch and multi-plat support,
  - By modifying the build scripts and restructuring the folders
- Added boot code for Arm64 QEMU-KVM,
- Support for single CPU for the first version,
- Enabled MMU
- Setup a 1:1 mapping for physical memory and virtual memory,
- Added an exception table,
  - That handles SYNC, IRQ and other exceptions
- Device tree support,
- PL011 UART for console,
  - Early debug console and STDIO
- Virtual timer for ticks.
Support multiple thread on Arm

Like most Unikernel projects, Unikraft supports single process but multiple threads. Current support status:

- Unikraft scheduler library is in reviewing (by Constan Lupu)

- Need to implement GIC interrupt controller libraries for timer interrupt.
  - GICv2 for low cost Arm SoC, like IoT devices
  - GICv3 and GICv4 for high performance Arm SoC, like Arm Servers
  - GICv2m and GICv3-ITS for MSI/MSI-X will not be supported initially

- Need to implement ARMv8 virtual timer library
  - Current timer library only provides ticks for timestamp
  - Need to sync timer APIs for scheduler with Costin
Foreseeable libraries on Arm

• Bare essential device libraries
  – GICv2, GICv3, GICv4 and ARMv8 virtual timer
• Bus libraries
  – virtio-mmio for Kvm
  – xenbus for Xen
  – Generic ECAM PCI host controller (optional, required by PCI pass-through)
• Virtual device libraries
  – netfront and blkfront for Xen
  – virtio-net and virtio-blk for Kvm
  – tapdev on Linux-user
• PSCI driver
  – implement a PSCI interface to shutdown virtual machine.
Footprint and boot time

Footprint:

- helloworld_kvm-arm64, 27 Kbytes
  
  Minimal memory usage, 132 Kbytes
  - 64KiB for DTB
    - Can be optimized, if you don’t need device tree
  - 28KiB for image (text, data, and bss)
    - 4KiB alignment
  - 20KiB for page table (not bss, a reserved memory area)
    - Can be optimized, if you don’t need page table
  - Left 20KiB for stack and heap

Boot time:

- ~50ms on Arm64 Cortex-A53 with QEMU
Summary

Unikraft reduces the barrier of converting an application to Unikernel greatly.

• It would be conducive to expand the Unikernel ecosystem.

But, Unikraft is new, it still has the following gaps:

• Need to implement more OS functionality libraries
• Need to improve the compatibility of standard LibC
• Need to implement more external libraries
• Add High-level language support (ocaml, ruby, node.js and lua)
• Support more platforms and hypervisors (kvmtool, ukvm and etc)
• Support enhanced profiling and tracing
References

Unikraft project wiki:
https://wiki.xenproject.org/wiki/Category:Unikraft

Unikraft project repositories:
http://xenbits.xen.org/gitweb/

Unikraft mailing list:
minios-devel@lists.xen.org (Shared with mini-os)

Unikraft Arm64 QEMU-KVM supports patches:
https://github.com/Weichen81/unikraft/tree/staging
Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!