



Association for
Computing Machinery

ASPLOS'22 Tutorial, 1st March 2022

Getting started with Unikraft

Alexander Jung <a.jung@{unikraft.org, lancaster.ac.uk}>



MANCHESTER
1824
The University of Manchester

NEC ARM OPENSYNERGY

THE
LINUX
FOUNDATION

Horizon 2020
European Union Funding
for Research & Innovation



LIÈGE
université

KIT
Karlsruher Institut für Technologie

UNICORE



Xen
Project



Engineering and
Physical Sciences
Research Council

Today's Tutorial

Time	Presentation	Presenter
Now! – 11:00	Getting started with Unikraft (I)	
11:00 – 11:30	 Coffee @ Foyer Garden 4 & 5	
11:30 – 12:00	Getting started with Unikraft (II)	
12:00 – 13:00	A look inside the build & config system	Razvan Deaconescu (UPB)
13:00 – 14:00		Lunch 
14:00 – 15:00	Running complex applications	Cristian Vijelie (UPB)
15:00 – 16:00	Running applications in binary compatibility	Razvan Deaconescu (UPB)
16:00 – 16:30	 Coffee @ Foyer Garden 4 & 5	
16:30 – 17:30	Using Unikrat or performance-oriented use cases	Vlad-Andrei Bădoi (UPB)
17:30 – 17:45	Unikernels: Paths to production & current trends	Hugo Lefevre (UoM)

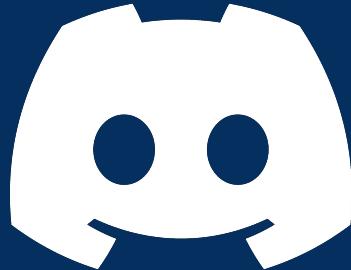
Tutorial Material

<https://asplos22.unikraft.org>

Online Attendees

For live help & support by the open-source community

<https://bit.ly/UnikraftDiscord>



The Unikraft Community

12

Publications

40+

Talks & Presentations

~50

Active Contributors

10

Institutes & Companies

6

Countries

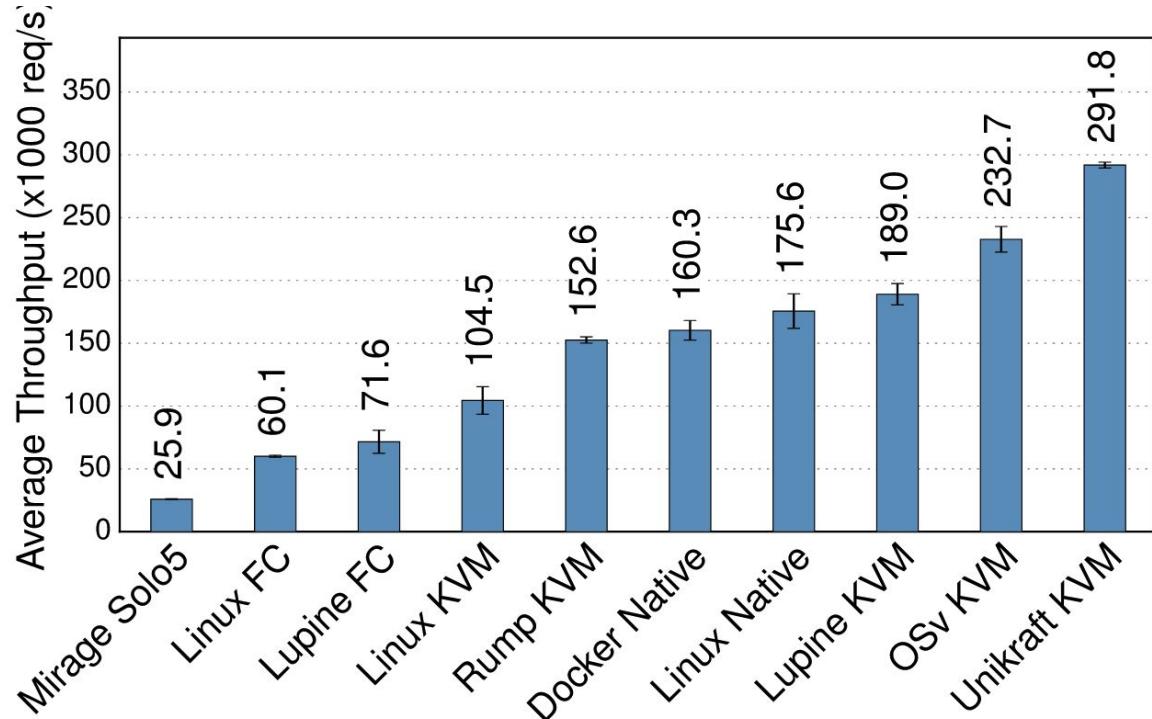
4

Years since first
commit

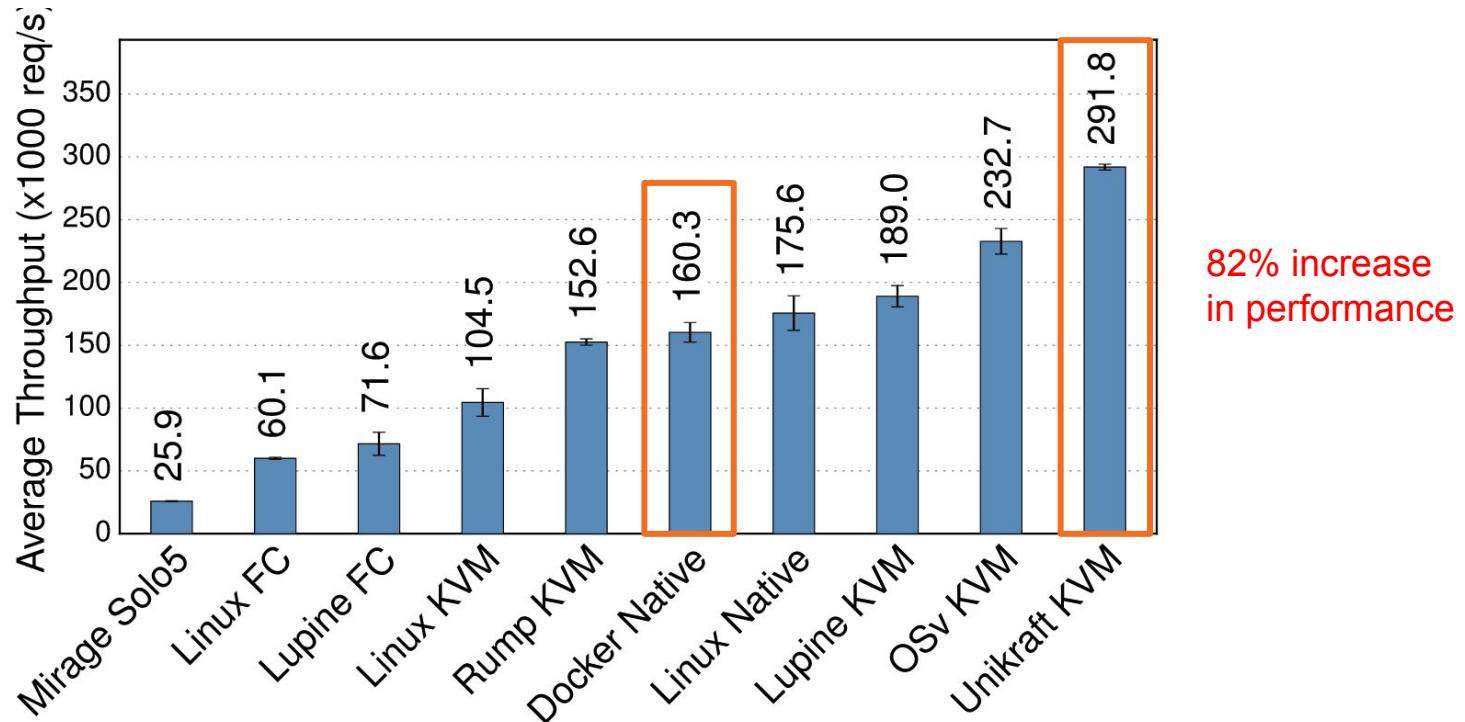
Supported “Native” Applications



Unikraft offers better performance

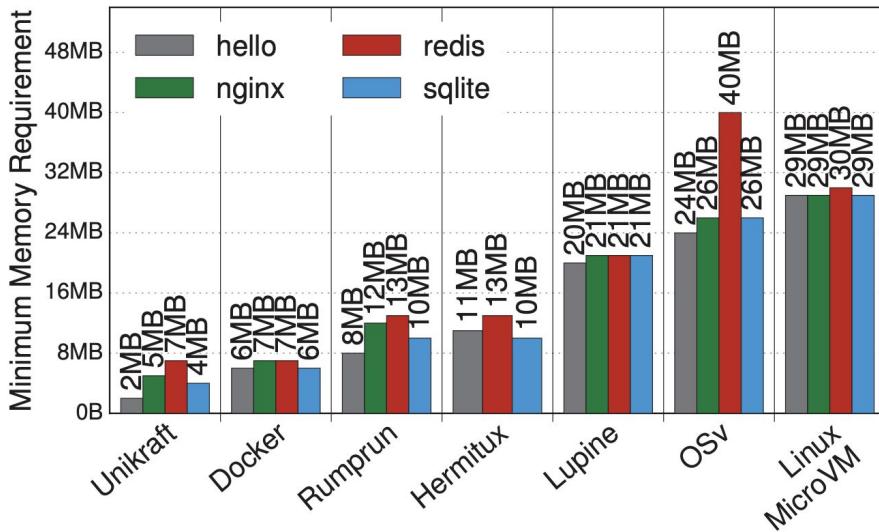


Unikraft offers better performance



Unikraft offers better storage

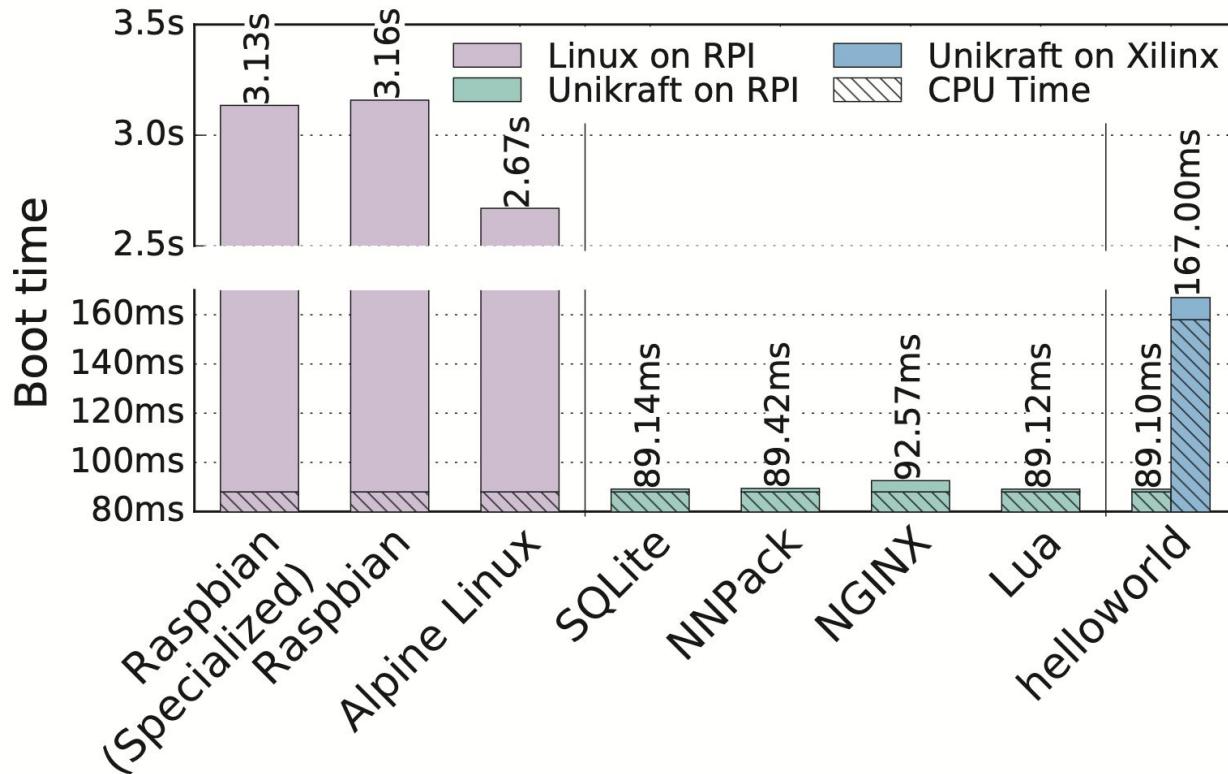
Memory Usage



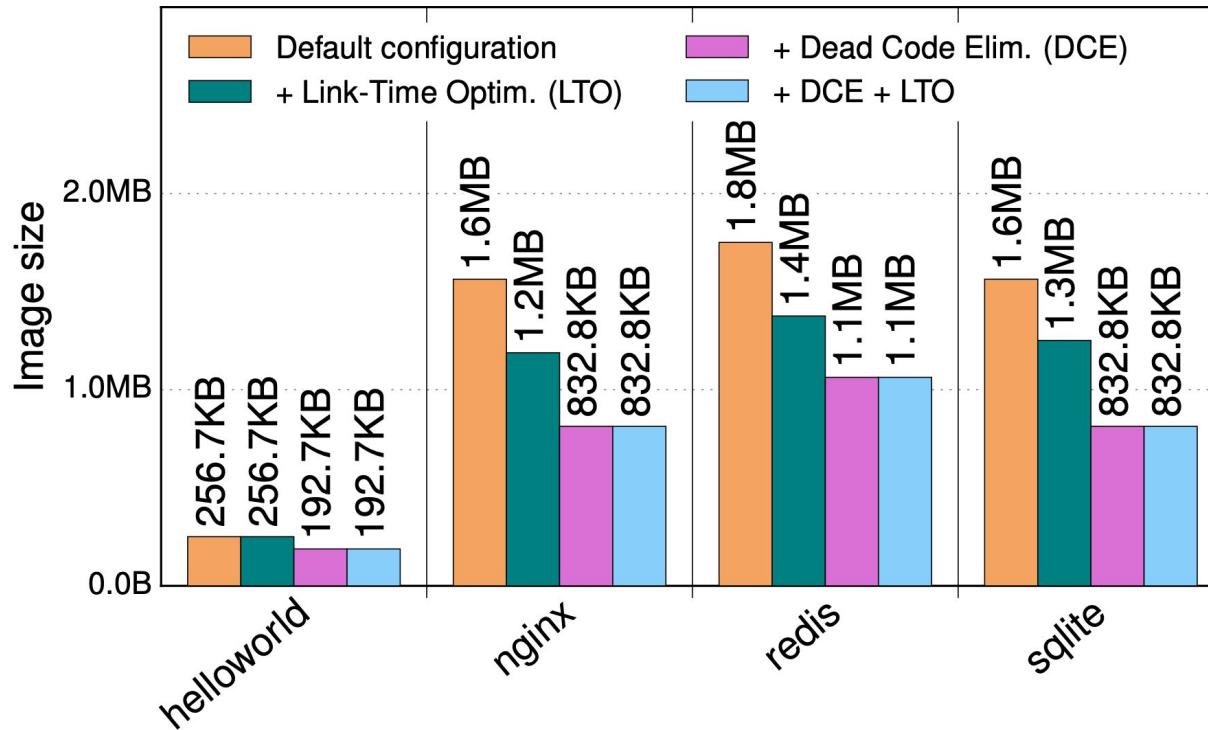
Disk Space

Image	Size
docker.io/nginx:1.15.6	42.62 MB
unikraft.io/nginx:1.15.6	1.3 MB

Unikraft offers better performance



Unikraft offers better optimization



The Unikraft Model

The Unikraft Model

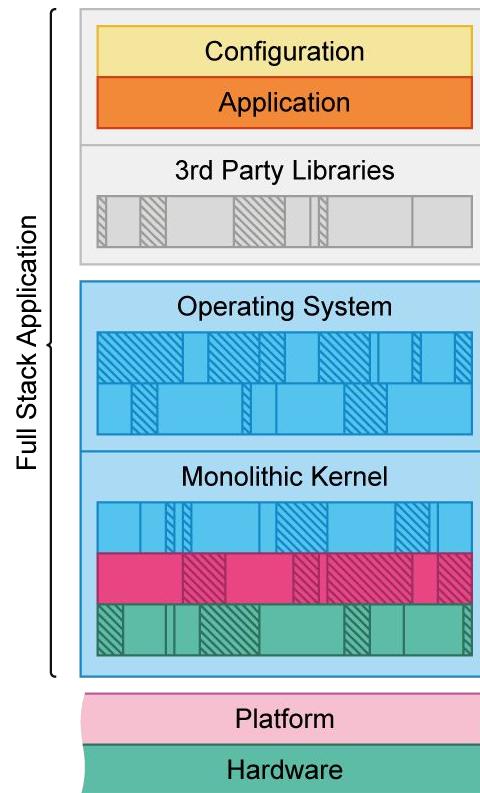
A Core Build System

The Unikraft Model

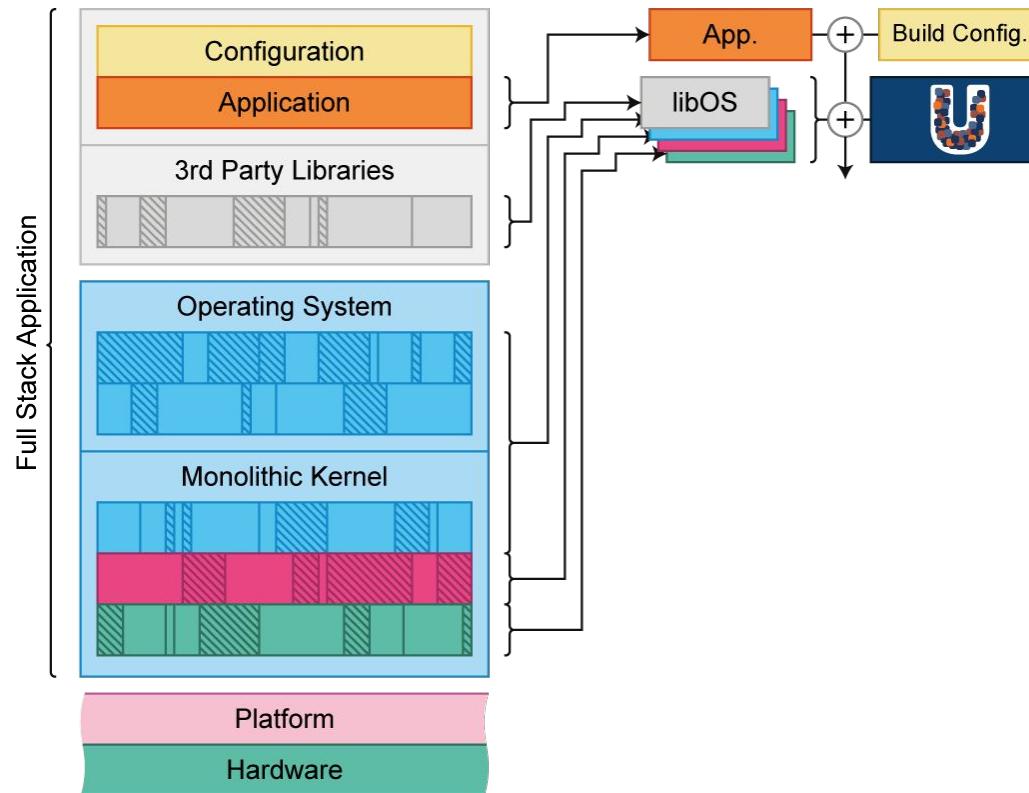
A Core Build System

“Everything is a library”

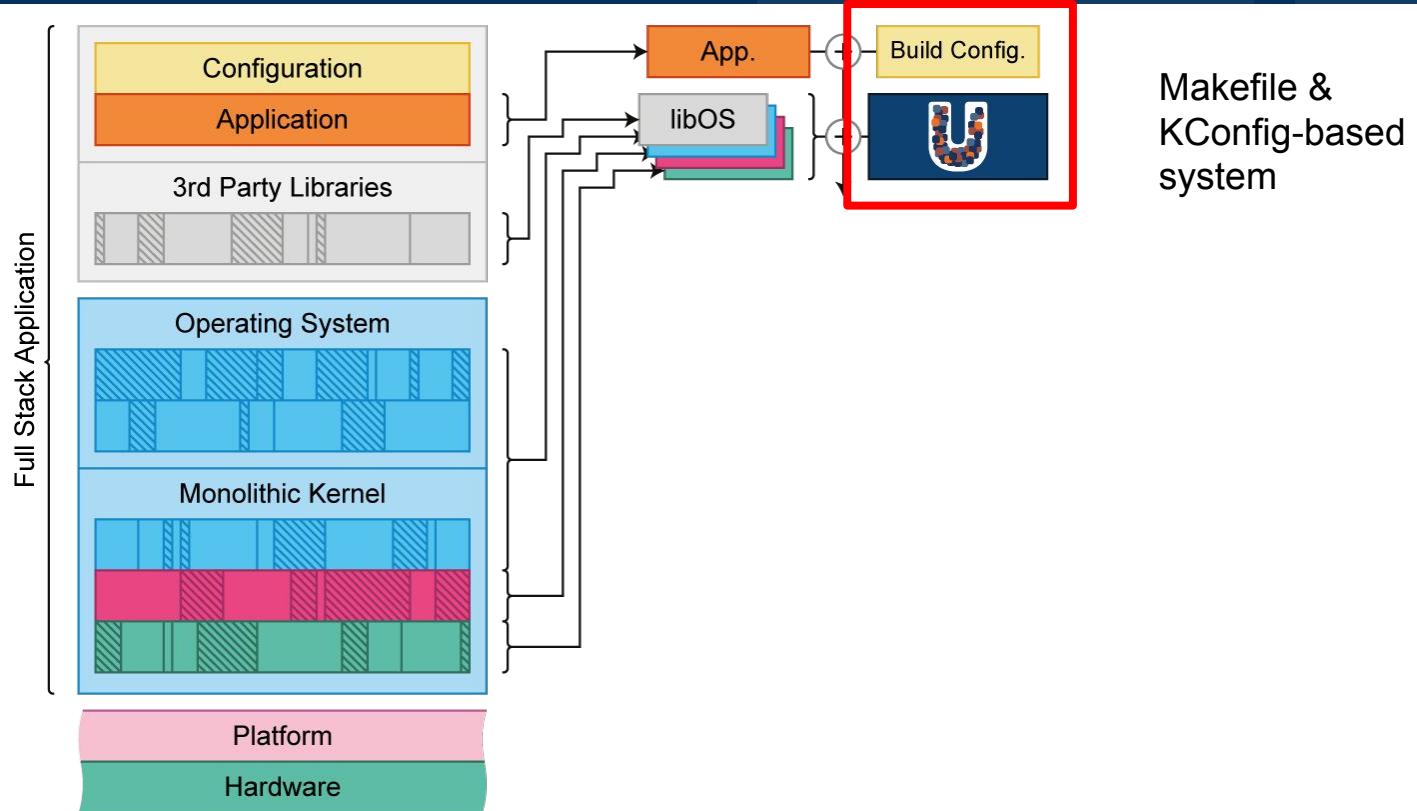
The Unikraft Model



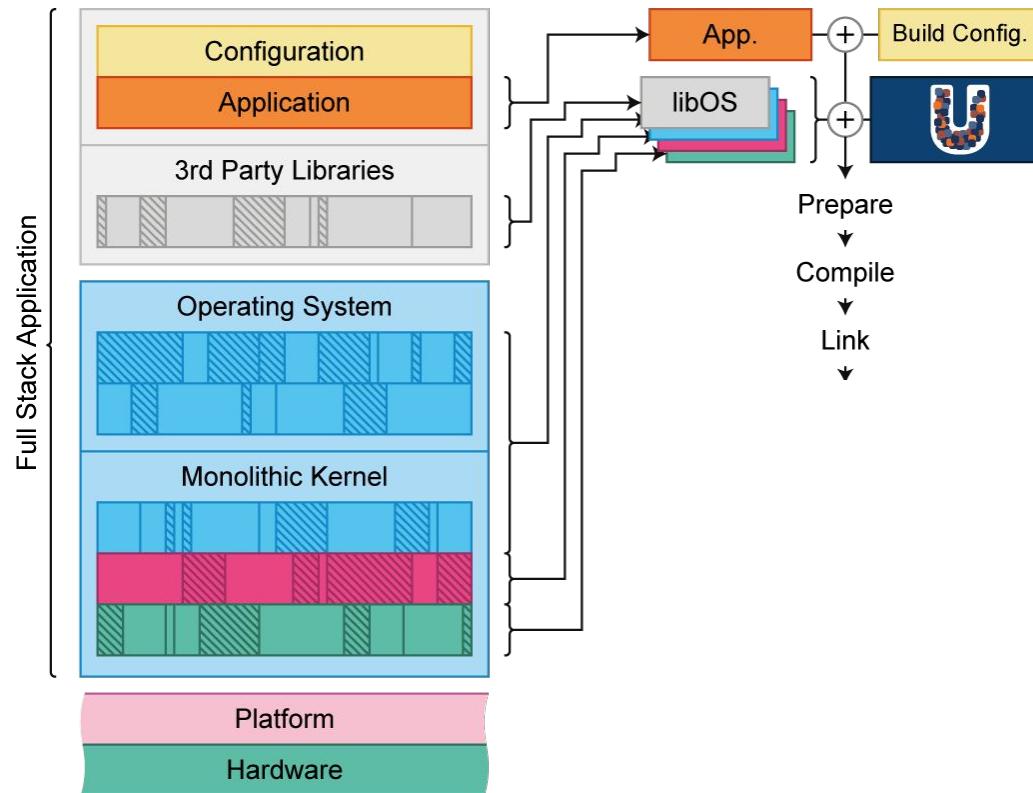
The Unikraft Model



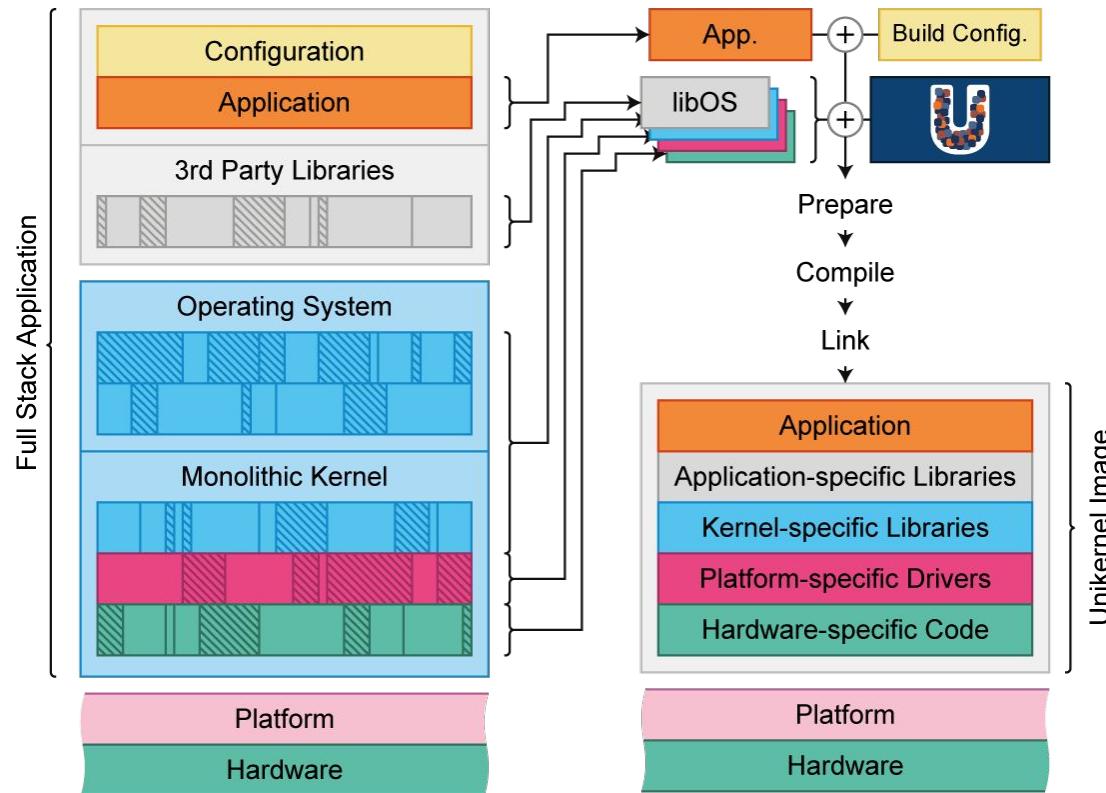
The Unikraft Model



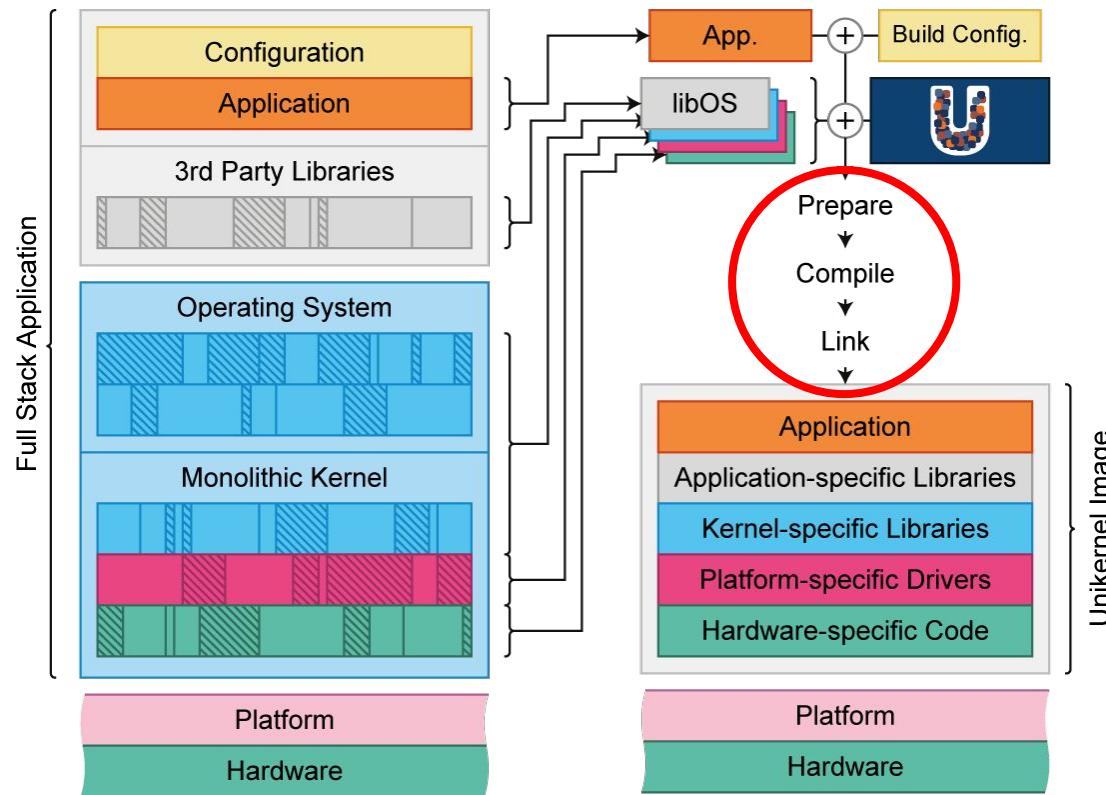
The Unikraft Model



The Unikraft Model



The Unikraft Model



Types of Unikraft libraries

A “native” library

- Unikraft “core” internal libs
- Has “_SRC-y”

Types of Unikraft libraries

A “native” library

- Unikraft “core” internal libs
- Has “_SRC-y”

A “wrapper” library

- An external library, e.g. openssl
- Markup to point to “origin”
- Still has “_SRC-y”

Types of Unikraft libraries

A “native” library

- Unikraft “core” internal libs
- Has “_SRC-y”

A “wrapper” library

- An external library, e.g. openssl
- Markup to point to “origin”
- Still has “_SRC-y”

Bonus: “Binary Compatibility” shared-objects
(Covered later by Razvan)

The Unikraft Model

A Core Build System

“Everything is a library”

The Unikraft Model

But,

How do we manage
many libraries?

kraft

- Easily manage multiple libraries from different sources
- Quickly access updates and change between versions
- Automatically download application source dependencies

PLATFORMS	VERSION	RELEASED	LAST CHECKED
unikraft	0.5	4 hours ago	1 hour ago
gcp	0.5	06 Feb 21	1 hour ago
digitalocean	0.5	06 Feb 21	1 hour ago
solo5	0.5	06 Feb 21	1 hour ago
aws	0.5	6 days ago	1 hour ago
firecracker	98327b0	23 Oct 20	1 hour ago

LIBRARIES	VERSION	RELEASED	LAST CHECKED
eigen	0.5	06 Feb 21	1 hour ago
libunwind	0.5	06 Feb 21	1 hour ago
lwip	0.5	5 hours ago	1 hour ago
pthreadpool	0.5	06 Feb 21	1 hour ago
psimd	0.5	06 Feb 21	1 hour ago
click	0.4	05 Feb 21	1 hour ago
pthread-embedded	0.5	6 days ago	1 hour ago
gcc	0.5	06 Feb 21	1 hour ago
zlib	0.5	06 Feb 21	1 hour ago
googlebenchmark	0.5	06 Feb 21	1 hour ago
nginx	0.5	06 Feb 21	1 hour ago
openssl	0.5	06 Feb 21	1 hour ago
newlib	0.5	16 Apr 21	1 hour ago
libucontext	0.5	06 Feb 21	1 hour ago
libuuid	0.5	06 Feb 21	1 hour ago
libfxdiv	0.5	06 Feb 21	1 hour ago
compiler-rt	0.5	06 Feb 21	1 hour ago
libcxxabi	0.5	06 Feb 21	1 hour ago
googletest	0.5	06 Feb 21	1 hour ago
pcre	0.5	06 Feb 21	1 hour ago
intel-intrinsics	0.5	06 Feb 21	1 hour ago
libfp16	0.5	06 Feb 21	1 hour ago
libcxx	0.5	06 Feb 21	1 hour ago
lua	0.5	06 Feb 21	1 hour ago
http-parser	0.5	06 Feb 21	1 hour ago
axtls	0.5	06 Feb 21	1 hour ago
bzip2	0.5	06 Feb 21	1 hour ago
python3	0.4	23 Oct 20	1 hour ago
wasmr	0.5	06 Feb 21	1 hour ago
libgo	0.5	06 Feb 21	1 hour ago

--More--

kraft can auto-configure your apps

- Unikernels defined in a `kraft.yaml`

kraft can auto-configure your apps

- Unikernels defined in a `kraft.yaml`

```
specification: '0.5'
unikraft: '0.5'
targets:
  - architecture: x86_64
    platform: kvm
libraries:
  newlib: stable
lwip:
  version: stable
kconfig:
  - CONFIG_LWIP_UDP=y
  ...
```

kraft can auto-configure your apps

- Unikernels defined in a `kraft.yaml`
- Add new library, target or version dependencies via the CLI:

```
$ kraft configure --use openssl \
                  --yes LIBCRYPTO

# or use the TUI

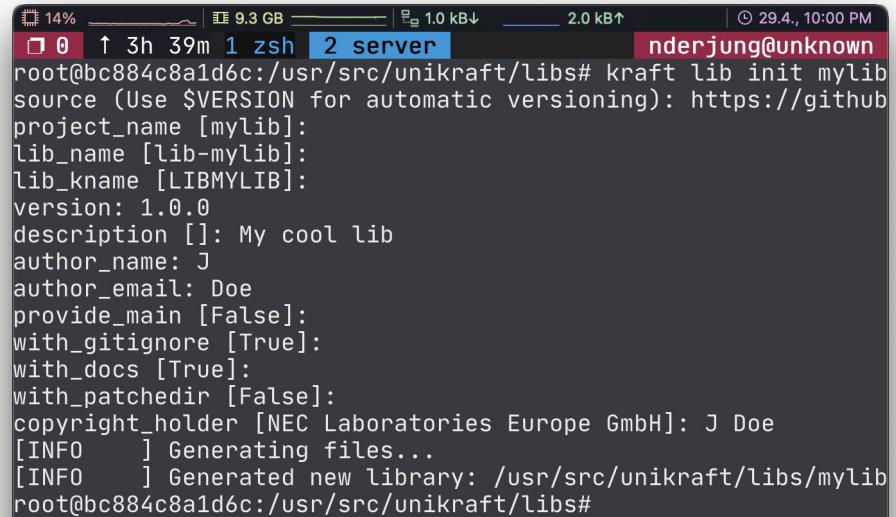
$ kraft menuconfig
```

```
specification: '0.5'
unikraft: '0.5'
targets:
  - architecture: x86_64
    platform: kvm
libraries:
  newlib: stable
  lwip:
    version: stable
  kconfig:
    - CONFIG_LWIP_UDP=y
  ...
```

kraft can help you build new libs

- Automatically generate a library from a cookiecutter template with on-screen help prompts:

`kraft lib init`



A screenshot of a terminal window titled "zsh" with a blue background. The window shows a command-line session where the user runs "kraft lib init mylib". The terminal displays several configuration prompts for the library, such as "source (Use \$VERSION for automatic versioning): https://github.com/nderjung/mylib", "project_name [mylib]:", "lib_name [lib-mylib]:", "lib_kname [LIBMYLIB]:", "version: 1.0.0", "description []: My cool lib", "author_name: J Doe", "author_email: Doe", "provide_main [False]:", "with_gitignore [True]:", "with_docs [True]:", "with_patchedir [False]:", "copyright_holder [NEC Laboratories Europe GmbH]: J Doe", "[INFO] Generating files...", and "[INFO] Generated new library: /usr/src/unikraft/libs/mylib". The terminal also shows the user's name "nderjung@unknown" and the date and time "29.4., 10:00 PM".

```
root@bc884c8a1d6c:/usr/src/unikraft/libs# kraft lib init mylib
source (Use $VERSION for automatic versioning): https://github.com/nderjung/mylib
project_name [mylib]:
lib_name [lib-mylib]:
lib_kname [LIBMYLIB]:
version: 1.0.0
description []: My cool lib
author_name: J Doe
author_email: Doe
provide_main [False]:
with_gitignore [True]:
with_docs [True]:
with_patchedir [False]:
copyright_holder [NEC Laboratories Europe GmbH]: J Doe
[INFO] Generating files...
[INFO] Generated new library: /usr/src/unikraft/libs/mylib
root@bc884c8a1d6c:/usr/src/unikraft/libs#
```

Access to build VMs

Pre-installed with all the tools you need!

<https://guacamole.grid.pub.ro/>



Username:

asplos

Password:

hakuna-matata

Kindly provided to us/you by University POLITEHNICA Bucharest 😊

Building your first Unikraft unikernel

1. Install build dependencies

bash

```
$ sudo apt-get install -y --no-install-recommends \
    build-essential
    libncurses-dev
    libyaml-dev
    flex
    git
    python3 python3-pip
    wget
    socat
    bison
    unzip
    uuid-runtime
```

Building your first Unikraft unikernel

2. Install kraft: the command-line companion for Unikraft

bash

```
$ pip3 install https://github.com/unikraft/kraft.git@staging
```

Building your first Unikraft unikernel

3. Set a access token to Github

<https://github.com/settings/tokens/new>

Select “repo:repo_public”

bash

```
$ export UK_KRAFT_GITHUB_TOKEN=ghp_..
```

Summary of kraft commands

\$ kraft list update	Update the manifest
\$ kraft list	List known libraries, apps, platforms & versions
\$ kraft list add https://github.com/me/lib-repo.git	Add a repo to the manifest
\$ kraft list pull	Pull a remote repo to your workspace
\$ kraft fetch	Fetch the “origin” of a Unikraft wrapper library
\$ kraft menuconfig	Open the KConfig menuconfig
\$ kraft configure	Configure the application based on kraft.yaml
\$ kraft build	Build the unikernel
\$ kraft run	Run the unikernel
\$ kraft up	Shortcut for fetch + configure + build + run

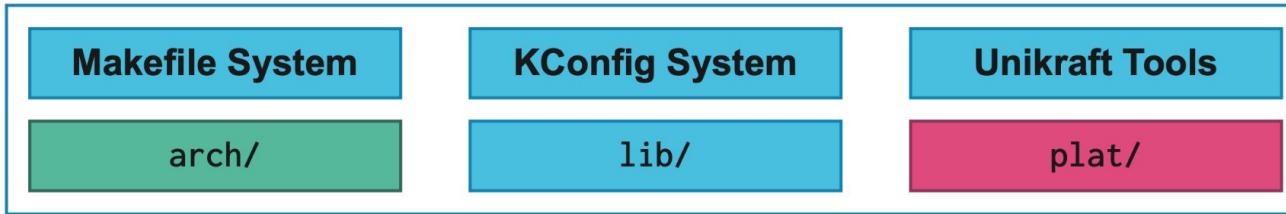
You have now
built a Unikraft
unikernel



You have now
built a Unikraft
unikernel

But how does Unikraft work?

Unikraft Repository



Unikraft Repository

Makefile System

arch/

KConfig System

lib/

Unikraft Tools

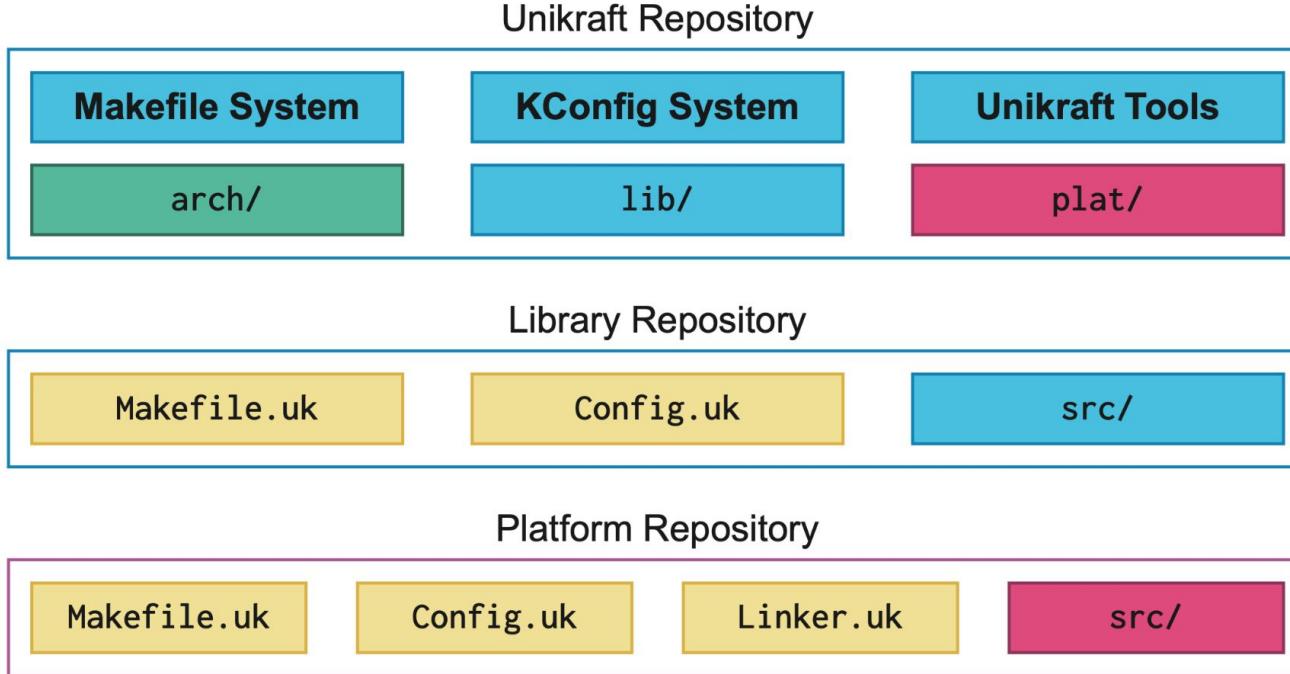
plat/

Library Repository

Makefile.uk

Config.uk

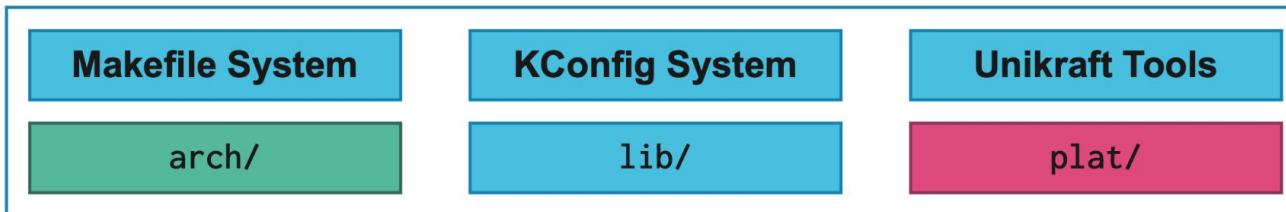
src/



Application Repository



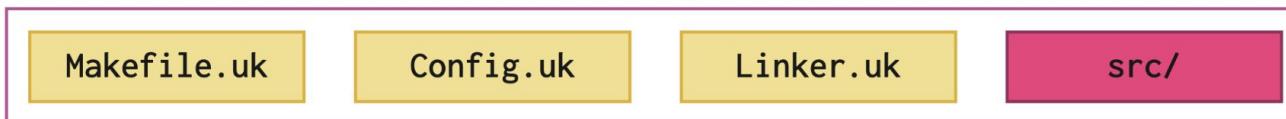
Unikraft Repository



Library Repository



Platform Repository

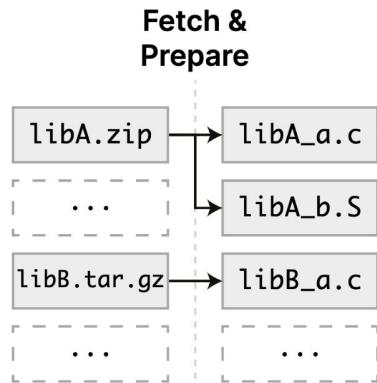


The Unikraft Model



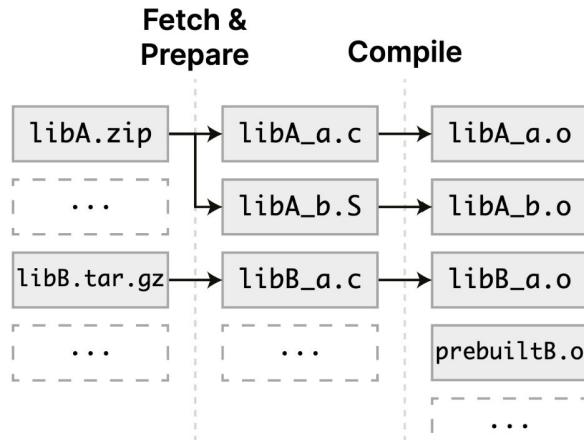
Makefile declares: LIB*_URL

The Unikraft Model



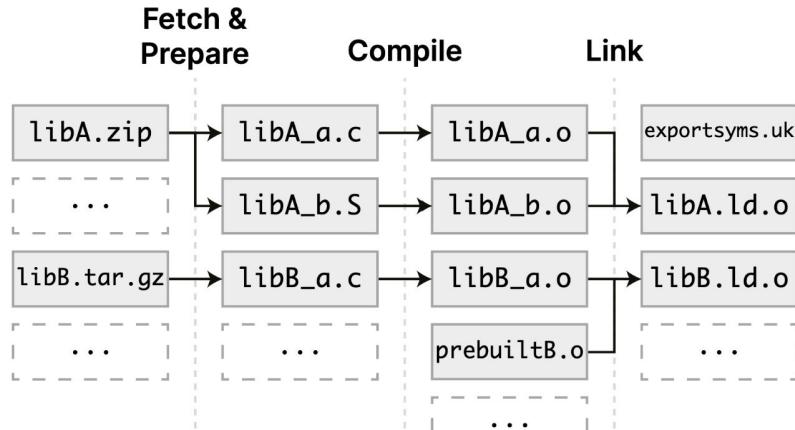
Makefile declares: LIB*_URL *_SRCS-y

The Unikraft Model



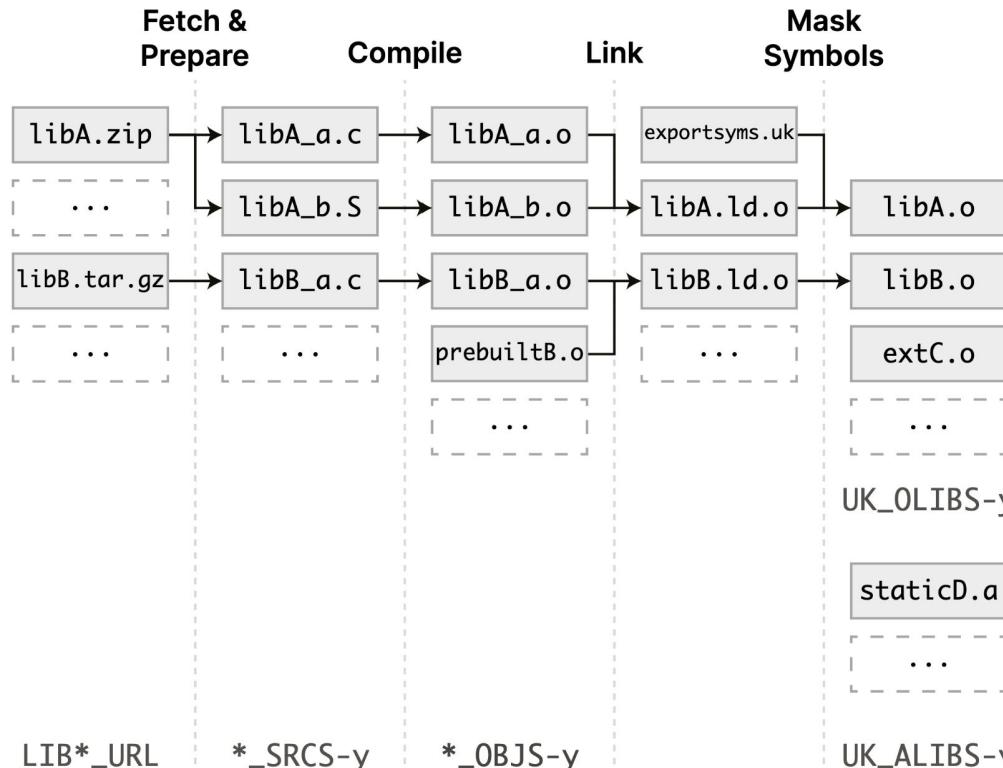
Makefile declares: LIB*_URL *_SRCS-y *_OJJS-y

The Unikraft Model



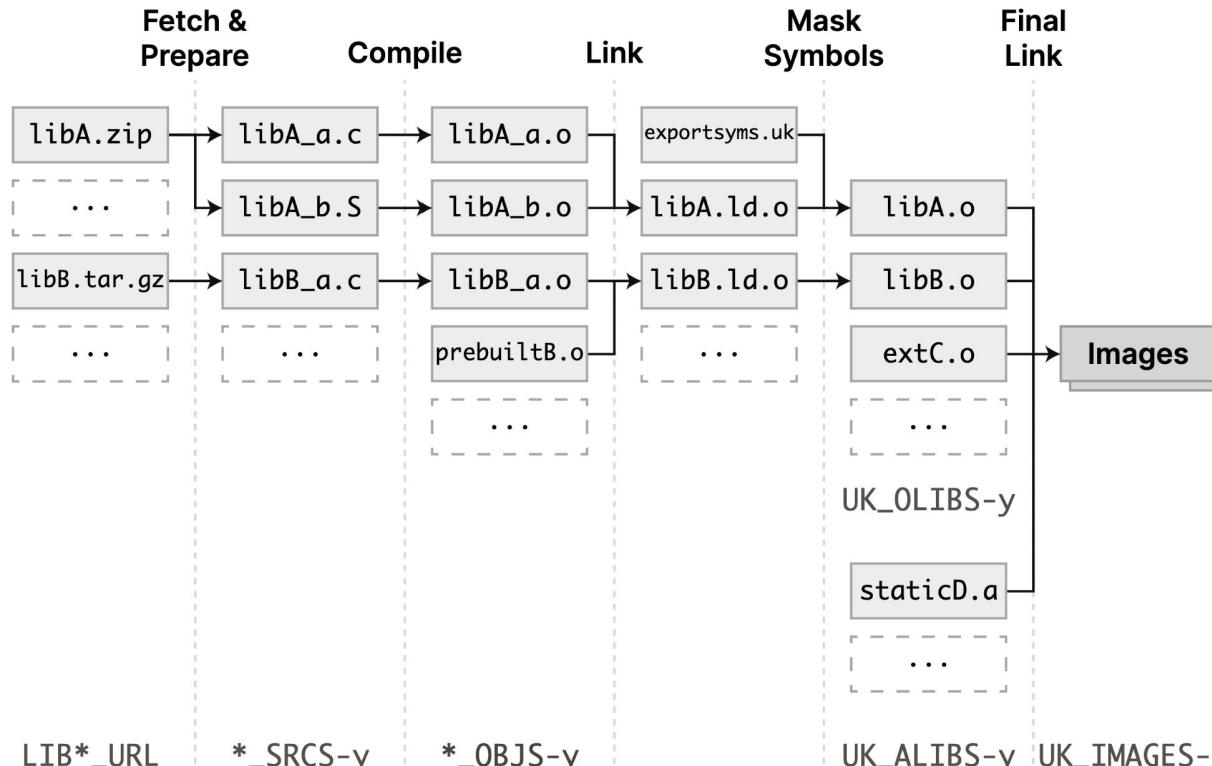
Makefile declares: LIB*_URL *_SRCS-y *_OBJS-y

The Unikraft Model



Copyright © Unikraft UG.

The Unikraft Model



Makefile declares: LIB*_URL *_SRCS-y *_OBJS-y

UK_OLIBS-y UK_ALIBS-y UK_IMAGES-y

Makefile.uk

<pre>\$eval \$(call addlib_s,LIBMYLIB,\$(CONIG_LIBMYLIB))</pre>	Register library
<pre>LIBMYLIB_VERSION=2.1.2 LIBMYLIB_URL=https://releases.mylib.org/v\$(LIBMYLIB_VERSION).zip \$eval \$(call fetch,libmylib,\$(LIBMYLIB_URL))</pre>	Fetch sources (optional)
<pre>\$(LIBMYLIB_BUILD)/.prepared: # my preparation steps here UK_PREPARE-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_BUILD)/.prepared</pre>	Custom prepare steps (optional)
<pre>LIBMYLIB_PDIR=\$(LIBMYLIB_BASE)/patches \$eval \$(call patch,libmylib,\$(LIBMYLIB_PDIR),\$(LIBMYLIB_VERSION))</pre>	Patch sources (optional)
<pre># Include from library directory LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_BASE)/include # Include from extracted archive LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_ORIGIN)/include</pre>	Include paths
<pre># Include from library directory LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_BASE)/source_a.c # Include from extracted archive LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_ORIGIN)/source_b.c</pre>	Include sources to build
<pre>LIBMYLIB_OBJS-y += \$(LIBMYLIB_ORIGIN)/prebuilt.o UK_ALIBS-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_ORIGIN)/static_lib.a</pre>	External objects (optional)

Makefile.uk

```
$eval $(call addlib_s,LIBMYLIB,$(CONIG_LIBMYLIB))  
  
LIBMYLIB_VERSION=2.1.2  
LIBMYLIB_URL=https://releases.mylib.org/v$(LIBMYLIB_VERSION).zip  
$eval $(call fetch,libmylib,$(LIBMYLIB_URL))  
  
$(LIBMYLIB_BUILD)/.prepared:  
    # my preparation steps here  
UK_PREPARE-$CONFIG_LIBMYLIB += $(LIBMYLIB_BUILD)/.prepared  
  
LIBMYLIB_PDIR=$(LIBMYLIB_BASE)/patches  
$eval $(call patch,libmylib,$(LIBMYLIB_PDIR),$(LIBMYLIB_VERSION))  
  
# Include from library directory  
LIBMYLIB_CINCLUDES-y += -I$(LIBMYLIB_BASE)/include  
# Include from extracted archive  
LIBMYLIB_CINCLUDES-y += -I$(LIBMYLIB_ORIGIN)/include  
  
# Include from library directory  
LIBMYLIB_SRCS-y += -I$(LIBMYLIB_BASE)/source_a.c  
# Include from extracted archive  
LIBMYLIB_SRCS-y += -I$(LIBMYLIB_ORIGIN)/source_b.c  
  
LIBMYLIB_OBJS-y += $(LIBMYLIB_ORIGIN)/prebuilt.o  
UK_ALIBS-$CONFIG_LIBMYLIB += $(LIBMYLIB_ORIGIN)/static_lib.a
```

Register library

Fetch sources
(optional)

Custom prepare
steps (optional)

Patch sources
(optional)

Include paths

Include sources
to build

External objects
(optional)

Makefile.uk

\$eval \$(call addlib_s,LIBMYLIB,\$(CONIG_LIBMYLIB)))	Register library
LIBMYLIB_VERSION=2.1.2 LIBMYLIB_URL=https://releases.mylib.org/v\$(LIBMYLIB_VERSION).zip \$eval \$(call fetch,libmylib,\$(LIBMYLIB_URL))	Fetch sources (optional)
\$(LIBMYLIB_BUILD)/.prepared: # my preparation steps here UK_PREPARE-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_BUILD)/.prepared	Custom prepare steps (optional)
LIBMYLIB_PDIR=\$(LIBMYLIB_BASE)/patches \$eval \$(call patch,libmylib,\$(LIBMYLIB_PDIR),\$(LIBMYLIB_VERSION))	Patch sources (optional)
# Include from library directory LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_BASE)/include # Include from extracted archive LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_ORIGIN)/include	Include paths
# Include from library directory LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_BASE)/source_a.c # Include from extracted archive LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_ORIGIN)/source_b.c	Include sources to build
LIBMYLIB_OBJS-y += \$(LIBMYLIB_ORIGIN)/prebuilt.o UK_ALIBS-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_ORIGIN)/static_lib.a	External objects (optional)

\$eval \$(call addlib_s,LIBMYLIB,\$(CONIG_LIBMYLIB))	Register library
LIBMYLIB_VERSION=2.1.2 LIBMYLIB_URL=https://releases.mylib.org/v\$(LIBMYLIB_VERSION).zip \$eval \$(call fetch,libmylib,\$(LIBMYLIB_URL))	Fetch sources (optional)
\$(LIBMYLIB_BUILD)/.prepared: # my preparation steps here UK_PREPARE-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_BUILD)/.prepared	Custom prepare steps (optional)
LIBMYLIB_PDIR=\$(LIBMYLIB_BASE)/patches \$eval \$(call patch,libmylib,\$(LIBMYLIB_PDIR),\$(LIBMYLIB_VERSION))	Patch sources (optional)
# Include from library directory LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_BASE)/include # Include from extracted archive LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_ORIGIN)/include	Include paths
# Include from library directory LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_BASE)/source_a.c # Include from extracted archive LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_ORIGIN)/source_b.c	Include sources to build
LIBMYLIB_OBJS-y += \$(LIBMYLIB_ORIGIN)/prebuilt.o UK_ALIBS-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_ORIGIN)/static_lib.a	External objects (optional)

\$eval \$(call addlib_s,LIBMYLIB,\$(CONIG_LIBMYLIB))	Register library
LIBMYLIB_VERSION=2.1.2 LIBMYLIB_URL=https://releases.mylib.org/v\$(LIBMYLIB_VERSION).zip \$eval \$(call fetch,libmylib,\$(LIBMYLIB_URL))	Fetch sources (optional)
\$(LIBMYLIB_BUILD)/.prepared: # my preparation steps here UK_PREPARE-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_BUILD)/.prepared	Custom prepare steps (optional)
LIBMYLIB_PDIR=\$(LIBMYLIB_BASE)/patches \$eval \$(call patch,libmylib,\$(LIBMYLIB_PDIR),\$(LIBMYLIB_VERSION))	Patch sources (optional)
# Include from library directory LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_BASE)/include # Include from extracted archive LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_ORIGIN)/include	Include paths
# Include from library directory LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_BASE)/source_a.c # Include from extracted archive LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_ORIGIN)/source_b.c	Include sources to build
LIBMYLIB_OBJS-y += \$(LIBMYLIB_ORIGIN)/prebuilt.o UK_ALIBS-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_ORIGIN)/static_lib.a	External objects (optional)

\$eval \$(call addlib_s,LIBMYLIB,\$(CONIG_LIBMYLIB))	Register library
LIBMYLIB_VERSION=2.1.2 LIBMYLIB_URL=https://releases.mylib.org/v\$(LIBMYLIB_VERSION).zip \$eval \$(call fetch,libmylib,\$(LIBMYLIB_URL))	Fetch sources (optional)
\$(LIBMYLIB_BUILD)/.prepared: # my preparation steps here UK_PREPARE-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_BUILD)/.prepared	Custom prepare steps (optional)
LIBMYLIB_PDIR=\$(LIBMYLIB_BASE)/patches \$eval \$(call patch,libmylib,\$(LIBMYLIB_PDIR),\$(LIBMYLIB_VERSION))	Patch sources (optional)
# Include from library directory LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_BASE)/include # Include from extracted archive LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_ORIGIN)/include	Include paths
# Include from library directory LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_BASE)/source_a.c # Include from extracted archive LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_ORIGIN)/source_b.c	Include sources to build
LIBMYLIB_OBJS-y += \$(LIBMYLIB_ORIGIN)/prebuilt.o UK_ALIBS-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_ORIGIN)/static_lib.a	External objects (optional)

\$eval \$(call addlib_s,LIBMYLIB,\$(CONIG_LIBMYLIB))	Register library
LIBMYLIB_VERSION=2.1.2 LIBMYLIB_URL=https://releases.mylib.org/v\$(LIBMYLIB_VERSION).zip \$eval \$(call fetch,libmylib,\$(LIBMYLIB_URL))	Fetch sources (optional)
\$(LIBMYLIB_BUILD)/.prepared: # my preparation steps here UK_PREPARE-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_BUILD)/.prepared	Custom prepare steps (optional)
LIBMYLIB_PDIR=\$(LIBMYLIB_BASE)/patches \$eval \$(call patch,libmylib,\$(LIBMYLIB_PDIR),\$(LIBMYLIB_VERSION))	Patch sources (optional)
# Include from library directory LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_BASE)/include # Include from extracted archive LIBMYLIB_CINCLUDES-y += -I\$(LIBMYLIB_ORIGIN)/include	Include paths
# Include from library directory LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_BASE)/source_a.c # Include from extracted archive LIBMYLIB_SRCS-y += -I\$(LIBMYLIB_ORIGIN)/source_b.c	Include sources to build
LIBMYLIB_OBJS-y += \$(LIBMYLIB_ORIGIN)/prebuilt.o UK_ALIBS-\$CONFIG_LIBMYLIB += \$(LIBMYLIB_ORIGIN)/static_lib.a	External objects (optional)



<https://github.com/unikraft>



<https://unikraft.org>



info@unikraft.io



[@UnikraftSDK](https://twitter.com/UnikraftSDK)



Ultimate Performance & Security.