Unleashing the Power of Unikernels with Unikraft

Florian Schmidt, Research Scientist, NEC Laboratories Europe

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreements no. 675806 ("5G CITY") and 761592 ("5G ESSENCE"). 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.

VMs vs Containers

VMs have been around for a long time

- They allow consolidation, isolation, migration, …

VMs vs Containers

- VMs have been around for a long time
  - They allow consolidation, isolation, migration, …
- Then containers came and many people LOVED them. Why?

VMs vs Containers

VMs have been around for a long time

- They allow consolidation, isolation, migration, …

Then containers came and many people LOVED them. Why?

- “Containers are much faster to bring up than VMs.
  My VM takes a minute to boot, my container only a second.”

VMs vs Containers

VMs have been around for a long time

- They allow consolidation, isolation, migration, …

Then containers came and many people LOVED them. Why?

- “Containers are much faster to bring up than VMs.
  My VM takes a minute to boot, my container only a second.”
- “Containers are much smaller.
  My VM takes 10 GB, my container only a few hundred MB.”

VMs vs Containers

VMs have been around for a long time

- They allow consolidation, isolation, migration, …

Then containers came and many people LOVED them. Why?

- “Containers are much faster to bring up than VMs.
  My VM takes a minute to boot, my container only a second.”
- “Containers are much smaller.
  My VM takes 10 GB, my container only a few hundred MB.”
- “Containers are much easier to create and deploy.
  I just write this Dockerfile and I’m done.”

Containers vs Unikernels

- I don’t want to bash containers.
  - Containers can be great!
  - For example, I love them for build environments
- But VMs do not have to be large, slow, and complicated

Containers vs Unikernels

I don’t want to bash containers.

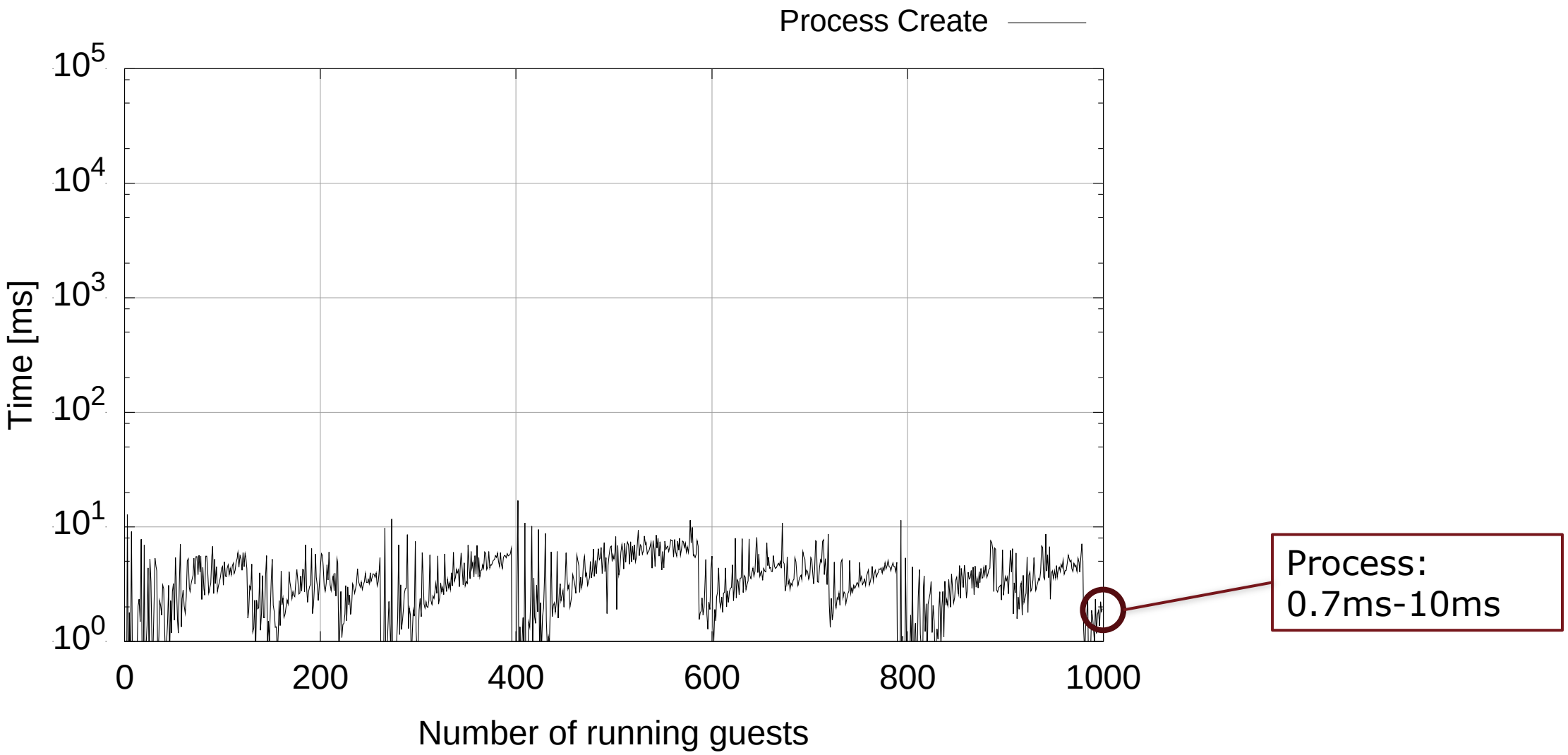
- Containers can be great!
- For example, I love them for build environments

But VMs do not have to be large, slow, and complicated

This is where unikernels come in

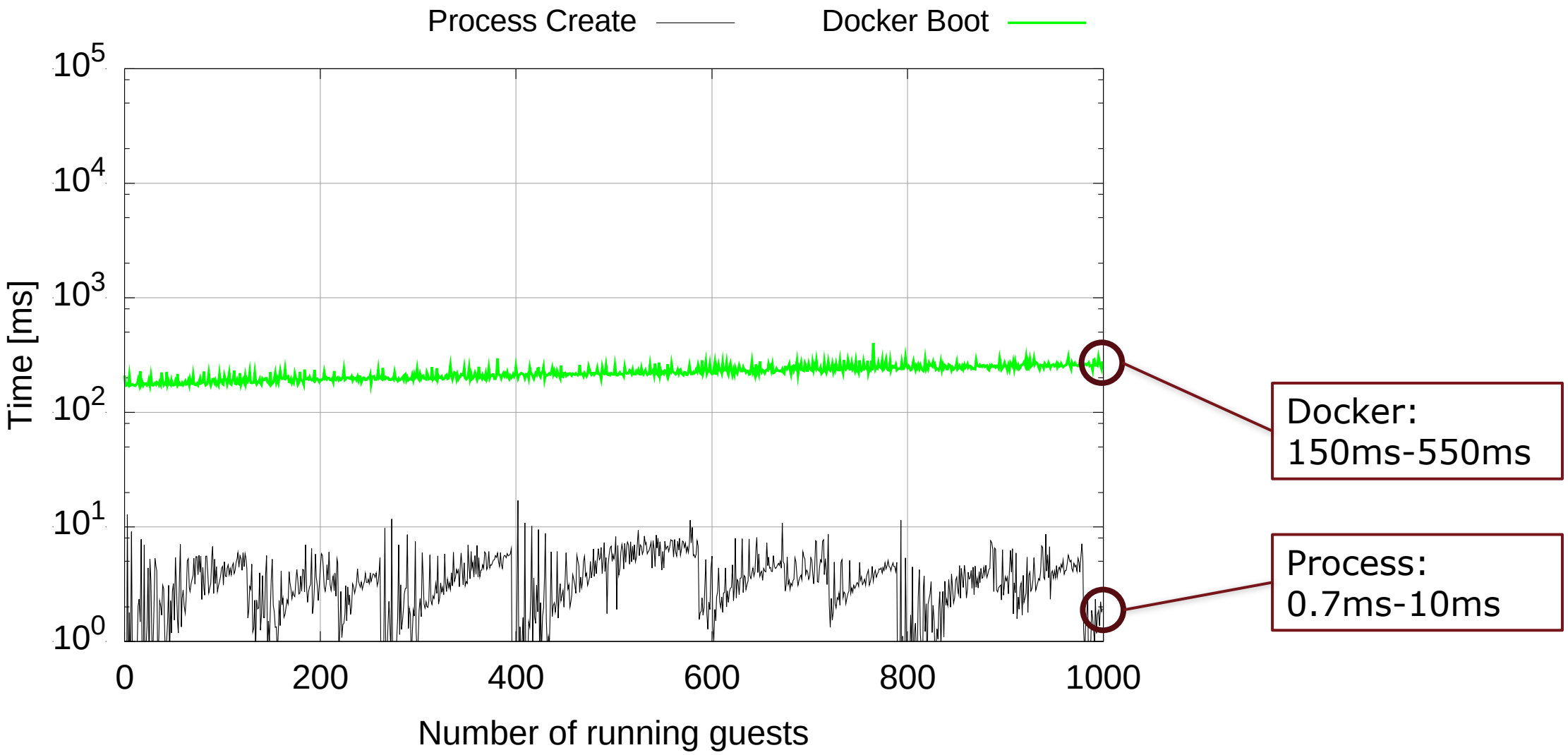
- A single binary containing OS and (single) application
- A shared address space, no mode switching
- No isolation within unikernel, but by the hypervisor
- Can be extremely small and blazingly fast

Example: Instantiation Time Comparison



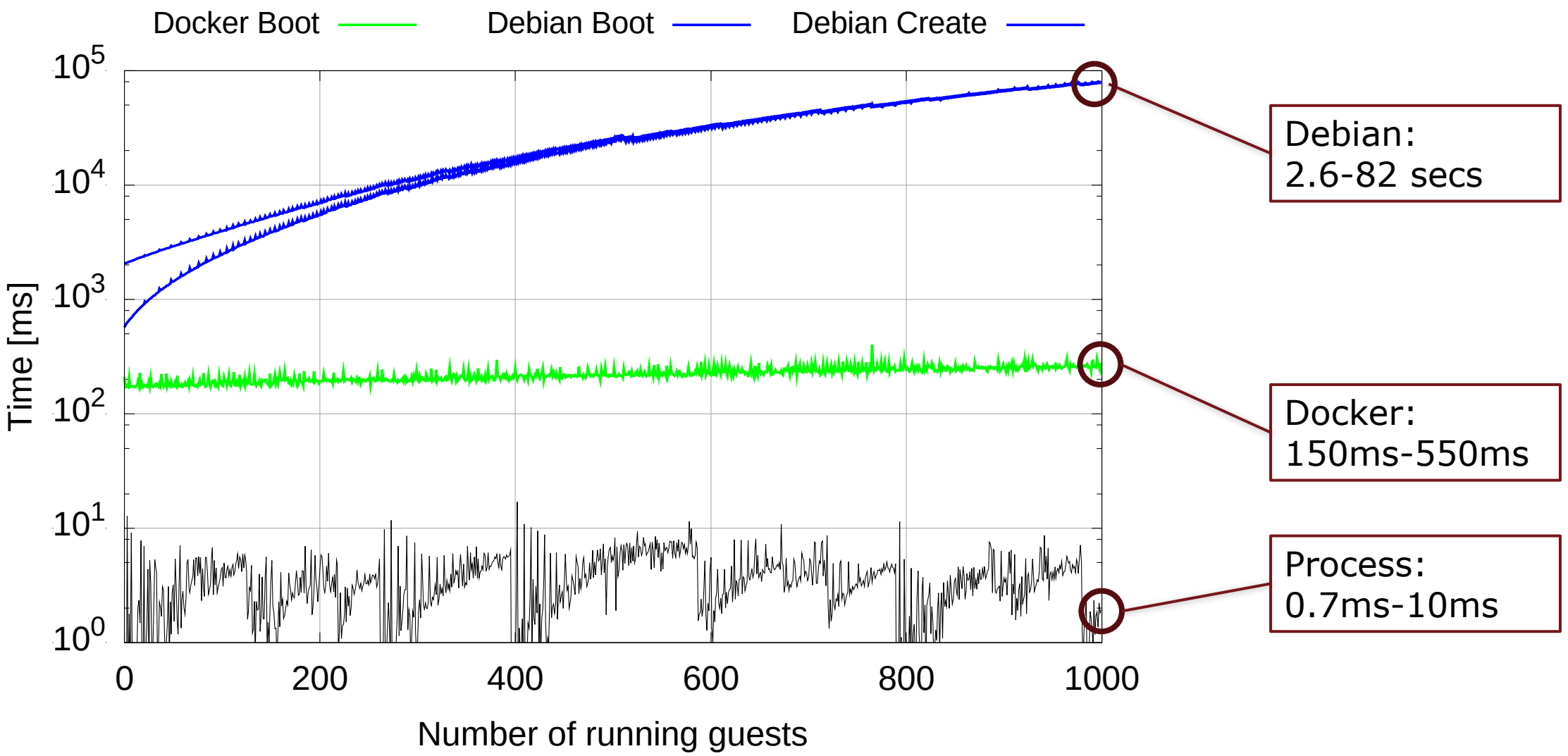
Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

Example: Instantiation Time Comparison



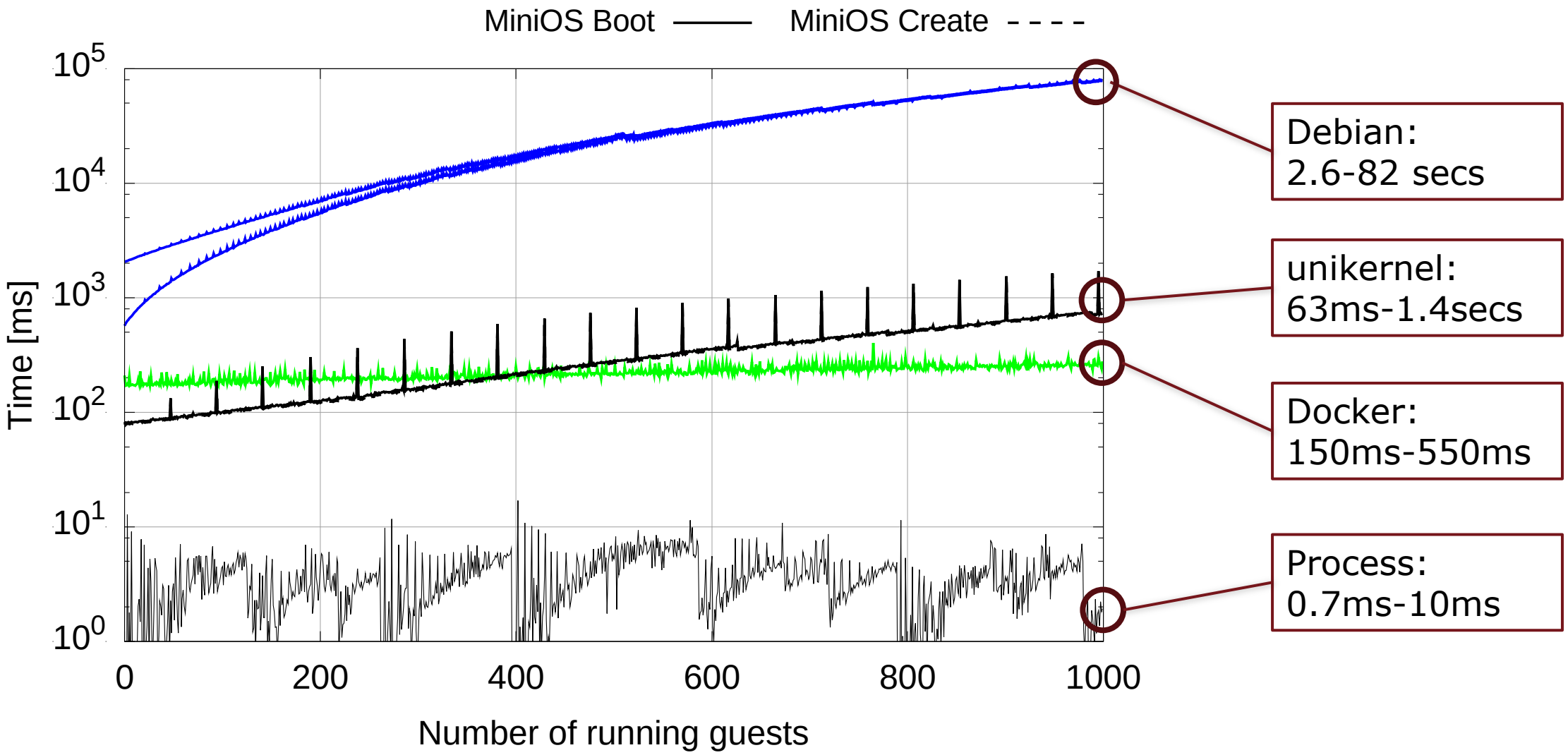
Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

Example: Instantiation Time Comparison



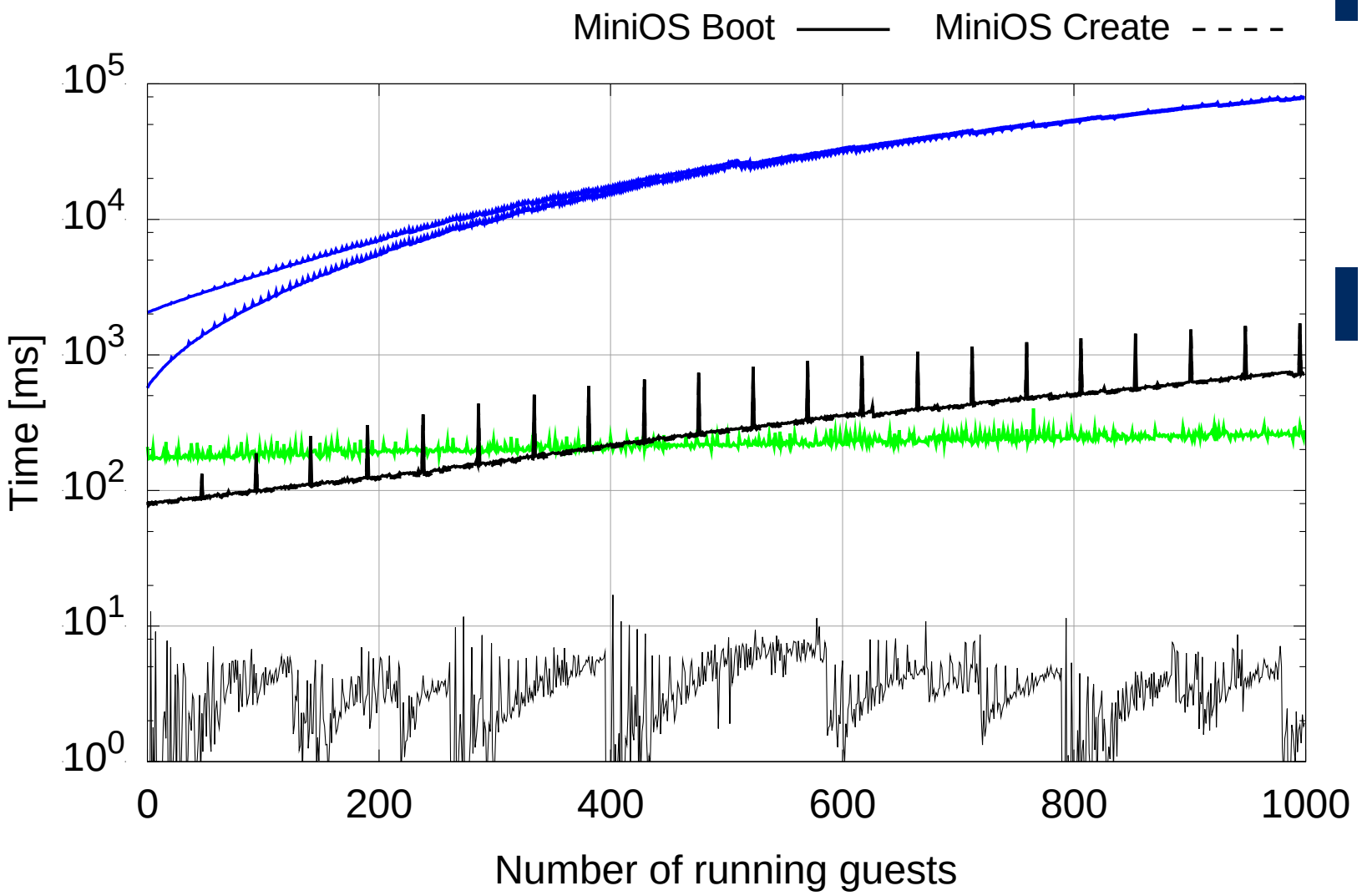
Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

Example: Instantiation Time Comparison



Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

Two messages from those results



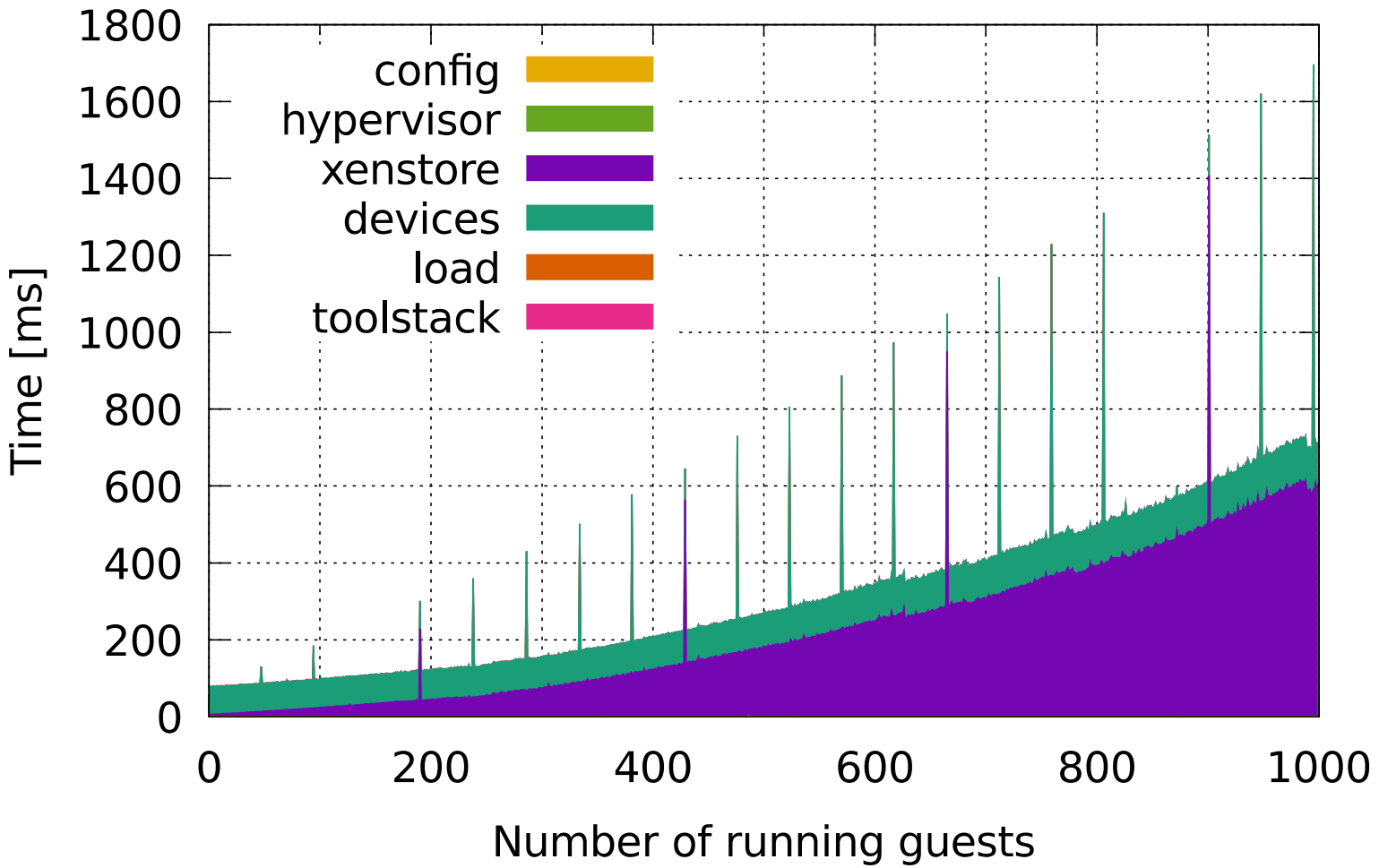
Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

- Unikernels can instantiate as fast as containers
- Except when many are colocated
  - Speaking of which, what **is** going wrong there?!

Intermission

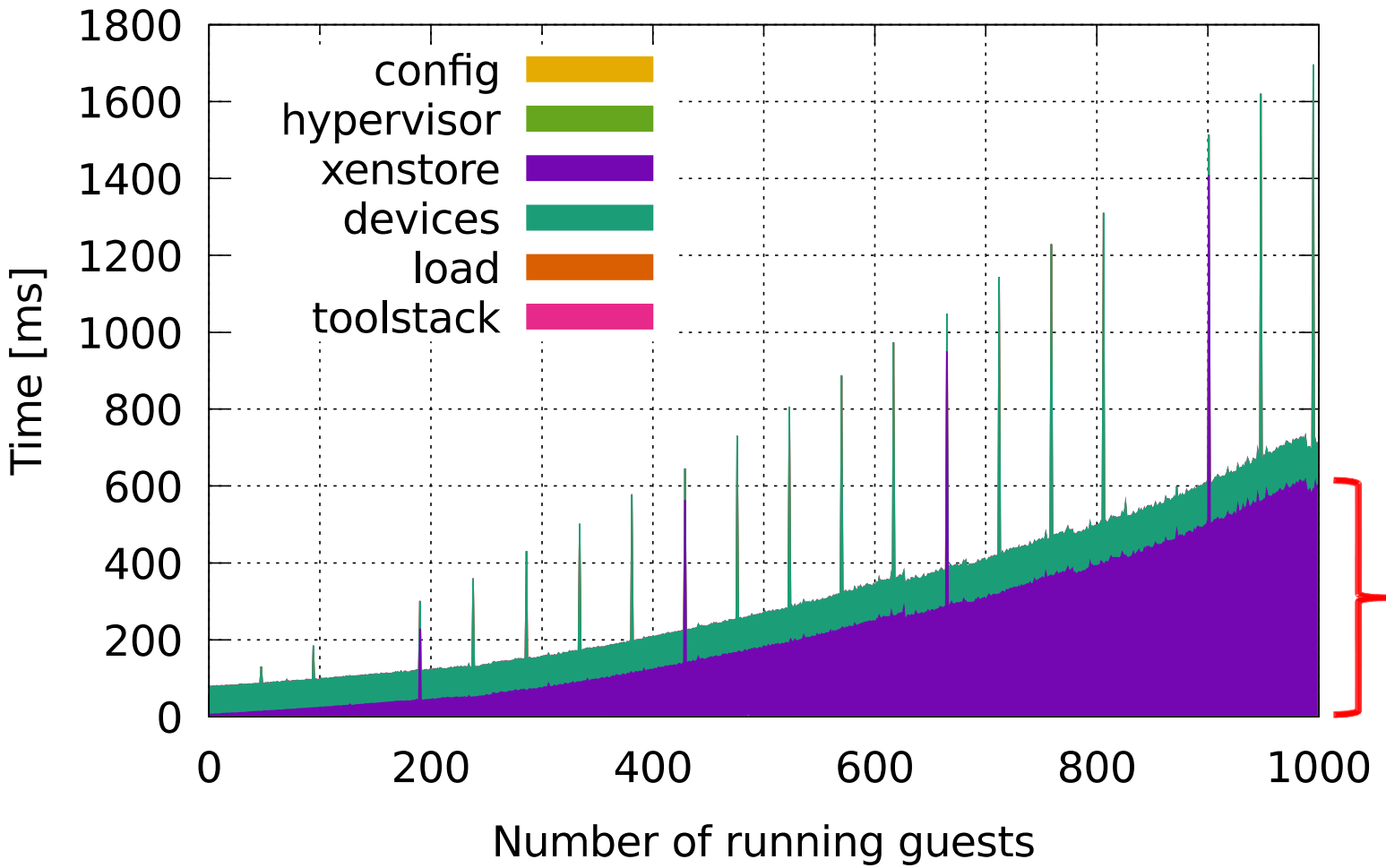
Anybody remember last year?

- Filipe Manco: “Death to the XenStore!” (Xen Developer and Design Summit 2017)



Anybody remember last year?

- Filipe Manco: “Death to the XenStore!” (Xen Developer and Design Summit 2017)



Yep, the XenStore isn’t exactly fast (and that’s already with oxenstored)

There’s an implementation now!

Implementation available

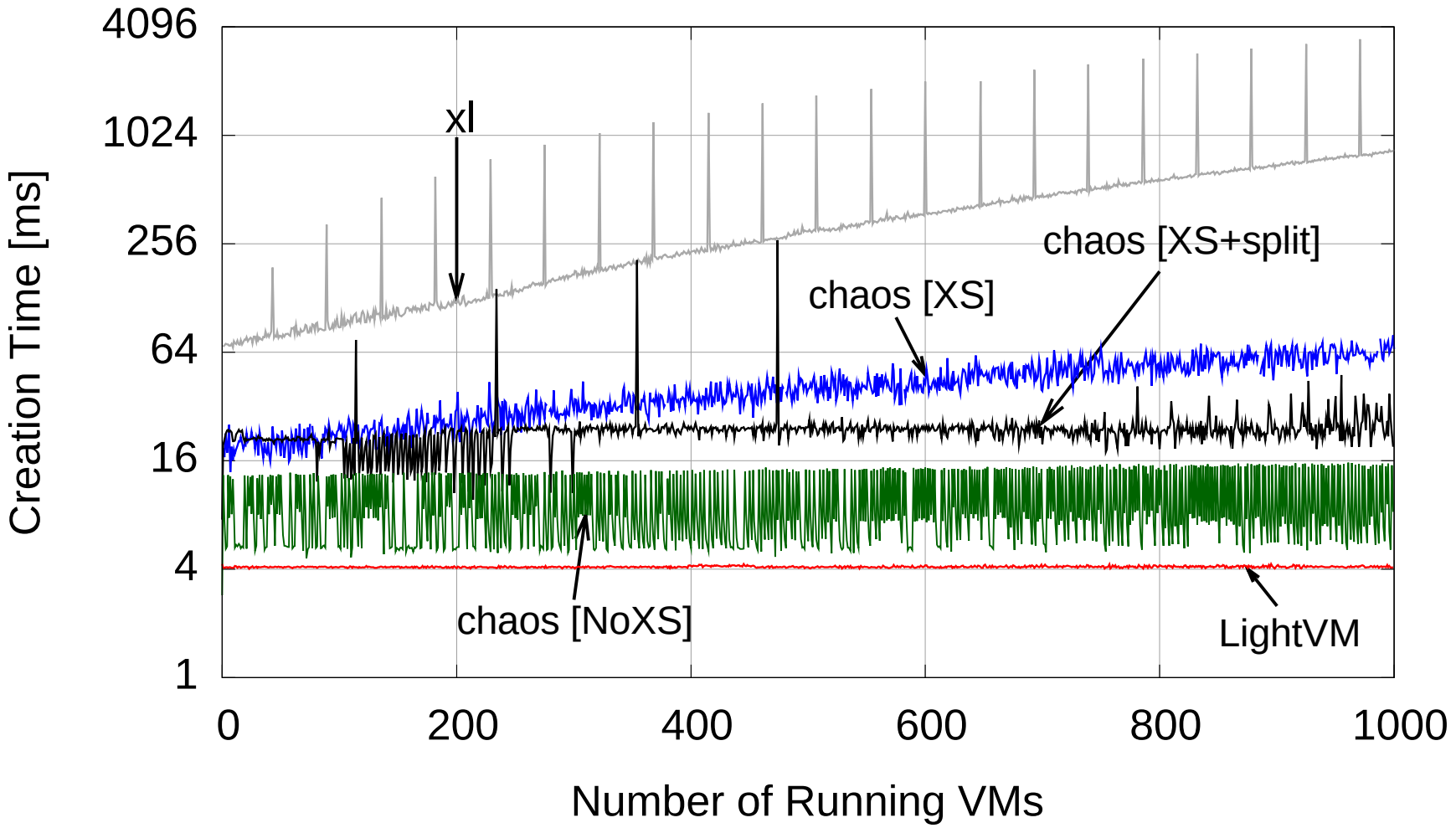
- NoXenstore: managing Xen VMs without a central management unit
- Plus assorted other bits and pieces of goodies
- Technology readiness level: research prototype
  (so don’t expect to see this exact implementation hit upstream anytime soon…)

Materials:

- http://sysml.neclab.eu/projects/lightvm/
- https://github.com/sysml/lightvm
- Manco et al., “My VM is Lighter (and Safer) than your Container”, SOSP 2017

LightVM Creation Performance

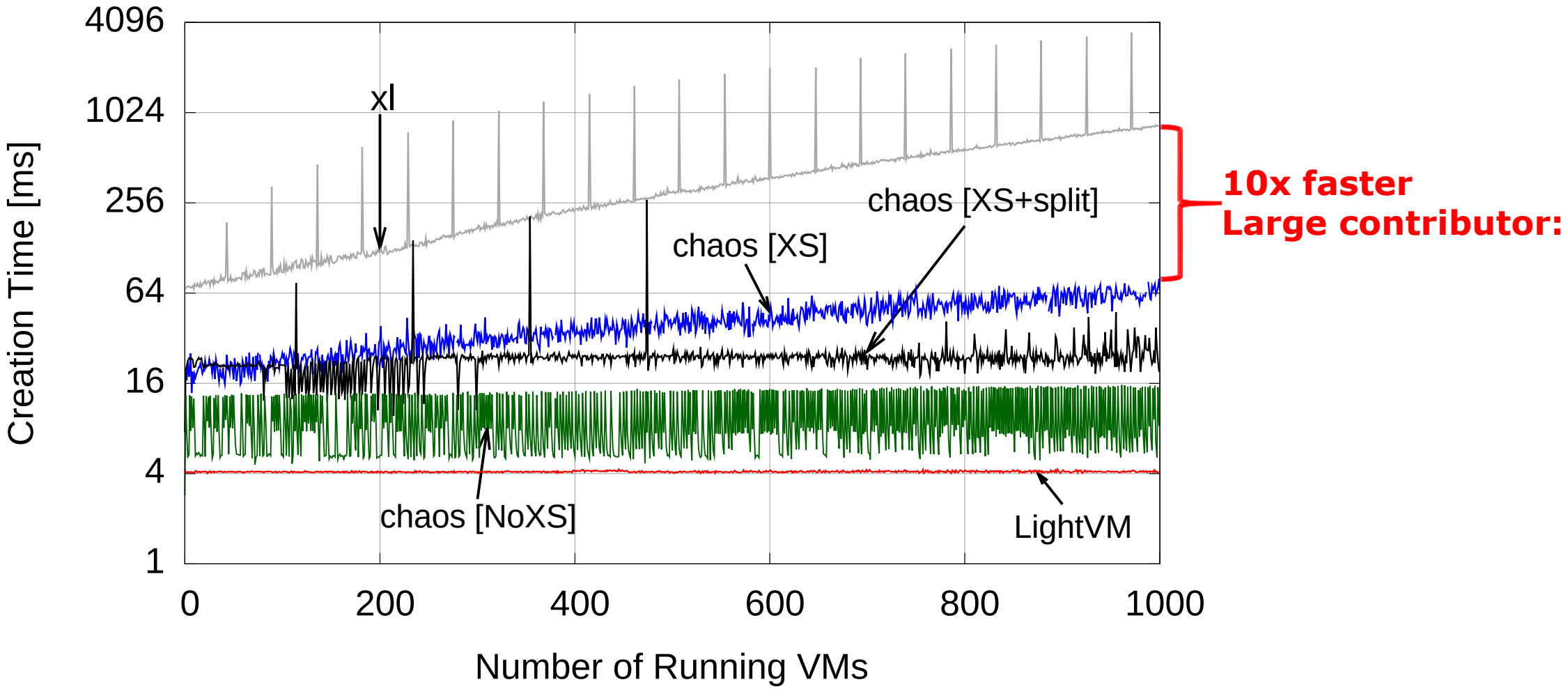
- On a dummy unikernel that does nothing, with 1 net device



Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

LightVM Creation Performance

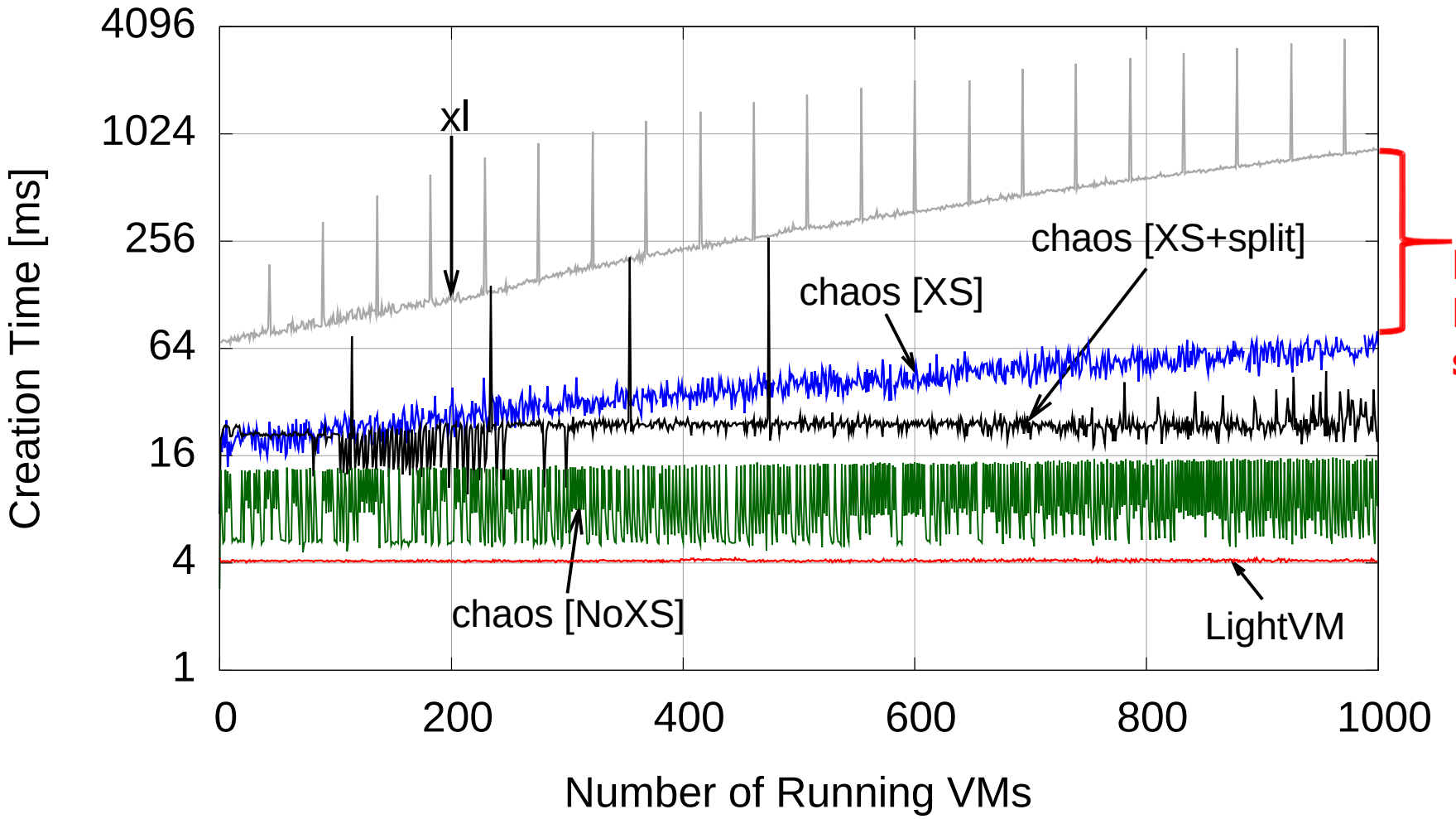
- On a dummy unikernel that does nothing, with 1 net device



Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

LightVM Creation Performance

- On a dummy unikernel that does nothing, with 1 net device

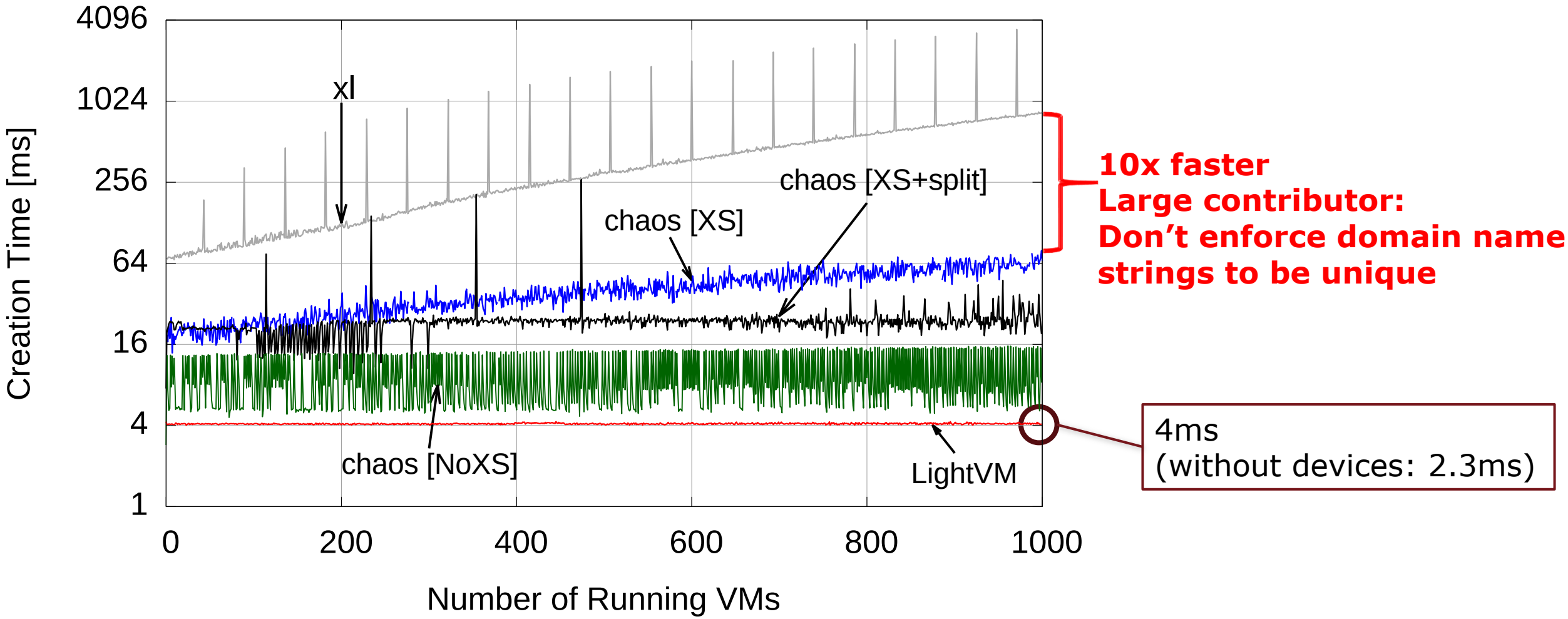


10x faster
Large contributor:
Don't enforce domain name strings to be unique

Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

LightVM Creation Performance

- On a dummy unikernel that does nothing, with 1 net device



Server: Intel Xeon E5-1630 v3 CPU@3.7GHz (4 cores), 128GB DDR4 RAM, Xen/Linux versions 4.8

OK, back to the main topic

The Downside

- So, unikernels are great and everybody should use them now?

The Downside

- So, unikernels are great and everybody should use them now?
- Hold your horses…
- The big problem is unikernel *development:*
  Optimized unikernels are manually built

The Downside

- So, unikernels are great and everybody should use them now?
- Hold your horses…
- The big problem is unikernel *development:*
  Optimized unikernels are manually built
  - Building takes several months or even longer
    - We've done it before, multiple times

The Downside

So, unikernels are great and everybody should use them now?

Hold your horses…

The big problem is unikernel *development:*
Optimized unikernels are manually built

- Building takes several months or even longer
  - We've done it before, multiple times
- Potentially lather, rinse, repeat for each target application
  - We've done that too…

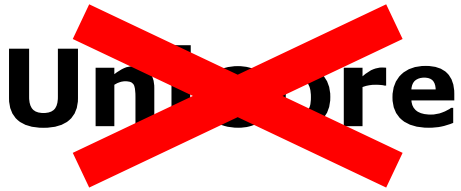
The Downside

- So, unikernels are great and everybody should use them now?
- Hold your horses…
- The big problem is unikernel *development:*
  Optimized unikernels are manually built
  - Building takes several months or even longer
    - We've done it before, multiple times
  - Potentially lather, rinse, repeat for each target application
    - We've done that too…
- That's not an effective way of doing things

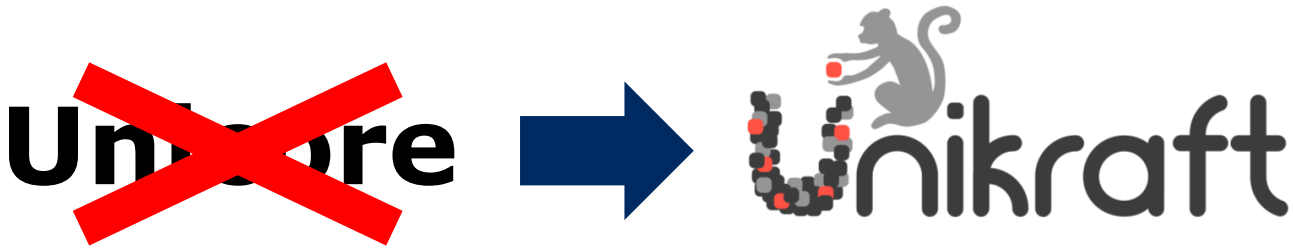
Unikraft - A Unikernel Framework

Unicore

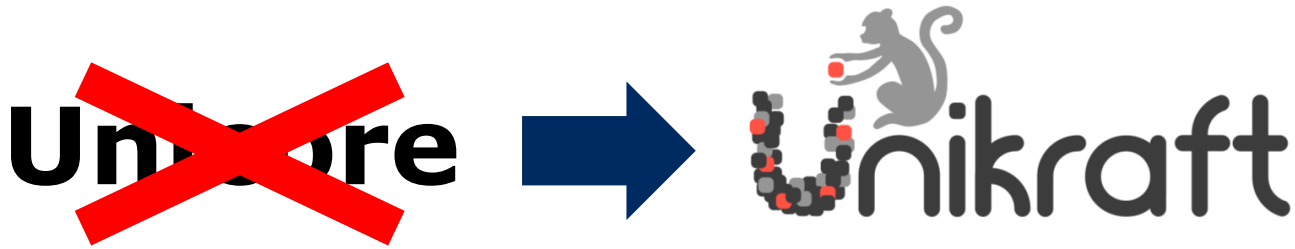
Unikraft - A Unikernel Framework



Unikraft - A Unikernel Framework



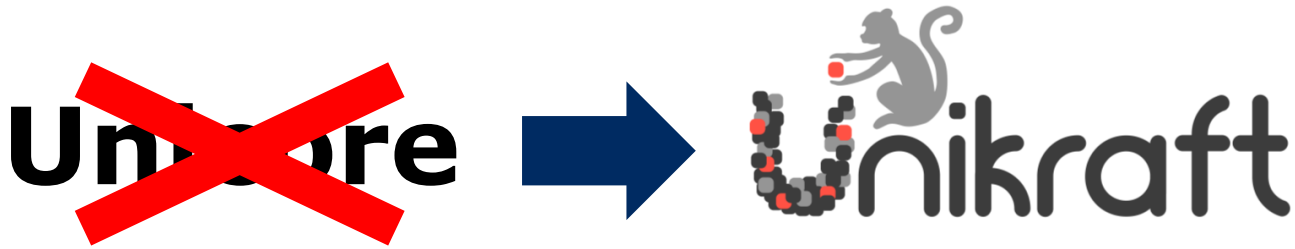
Unikraft - A Unikernel Framework



Motivation

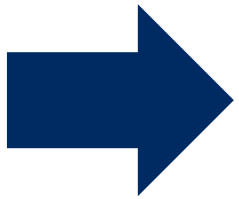
- Core principle: Support a wide range of use cases
- Provide common code base for unikernel development
- Simplify building and optimizing

Unikraft - A Unikernel Framework

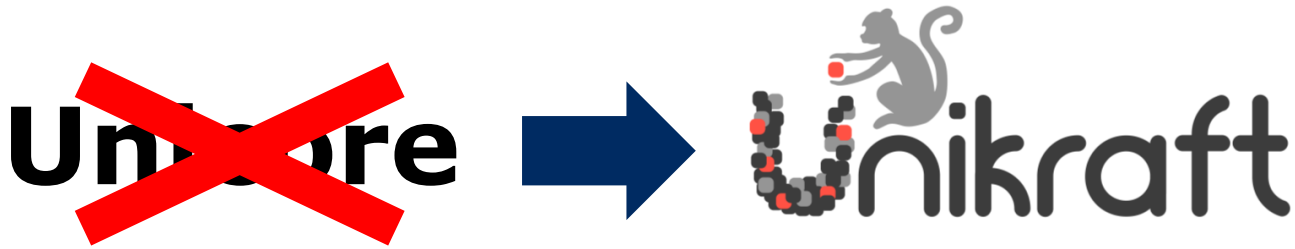


Motivation

- Core principle: Support a wide range of use cases
- Provide common code base for unikernel development
- Simplify building and optimizing



Unikraft - A Unikernel Framework



Motivation

- Core principle: Support a wide range of use cases
- Provide common code base for unikernel development
- Simplify building and optimizing

Our Approach

- Decompose OS functionality into libraries
- Unikraft's two components:
  - Library Pool
  - Build Tool

Unikraft - A Unikernel Framework

~~Unicore~~ → Unikraft

Motivation

- Core principle: Support a wide range of use cases
- Provide common code base for unikernel development
- Simplify building and optimizing

→

Our Approach

- Decompose OS functionality into libraries
- Unikraft's two components:
  - Library Pool
  - Build Tool

- Started as an internal project at NEC Labs in early 2017
- Made public early on
  - Discussed ideas at Xen Summit 2017
  - Accepted as an incubator project in October 2017
  - First public code release in December 2017

The Unikraft Way

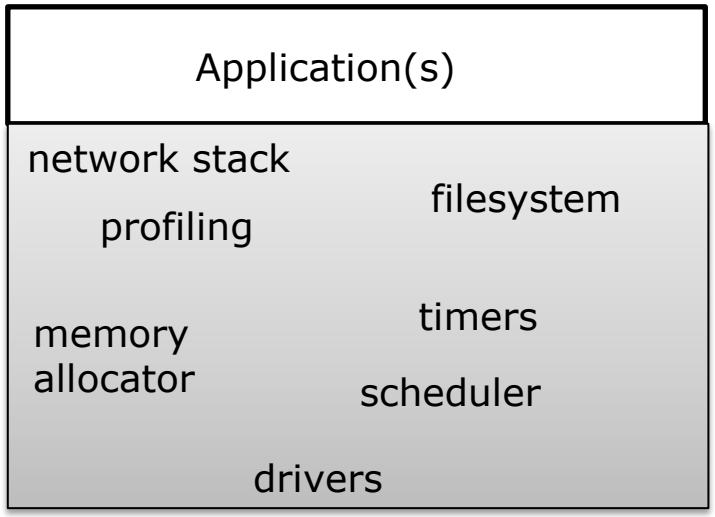
Decompose OS into a set of libraries (“Everything is a library”)

Recompose them to meet the needs of particular applications

The Unikraft Way

Decompose OS into a set of libraries (“Everything is a library”)

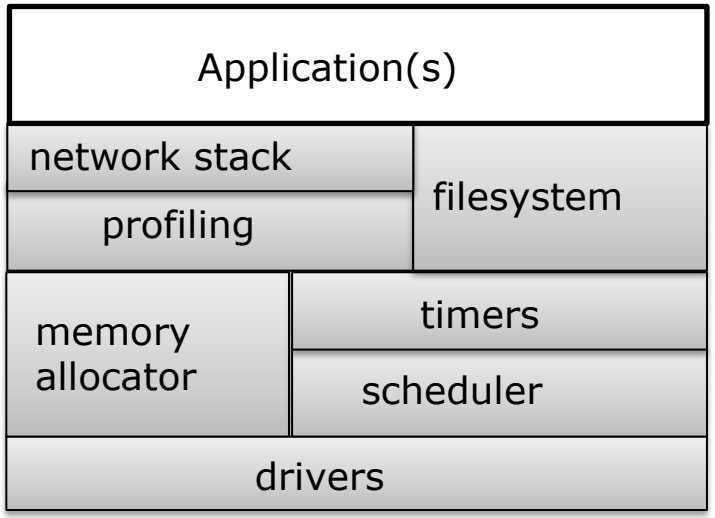
Recompose them to meet the needs of particular applications



The Unikraft Way

Decompose OS into a set of libraries (“Everything is a library”)

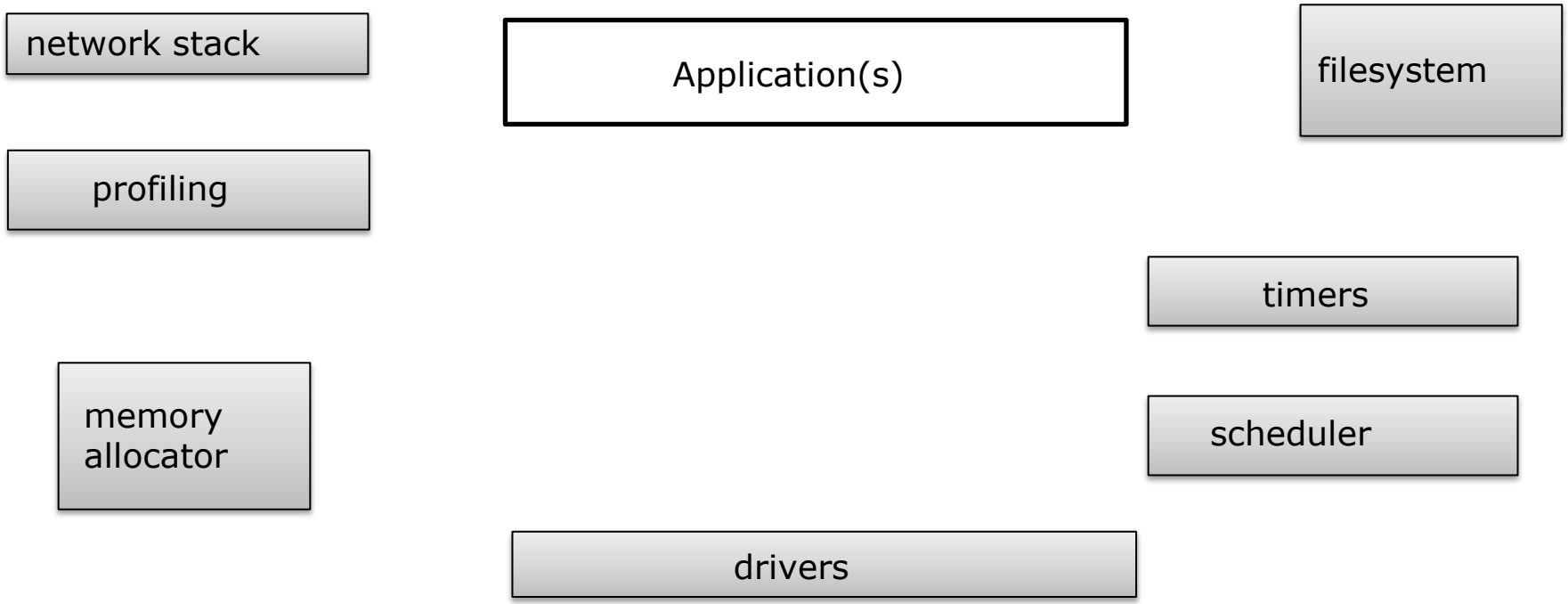
Recompose them to meet the needs of particular applications



The Unikraft Way

Decompose OS into a set of libraries (“Everything is a library”)

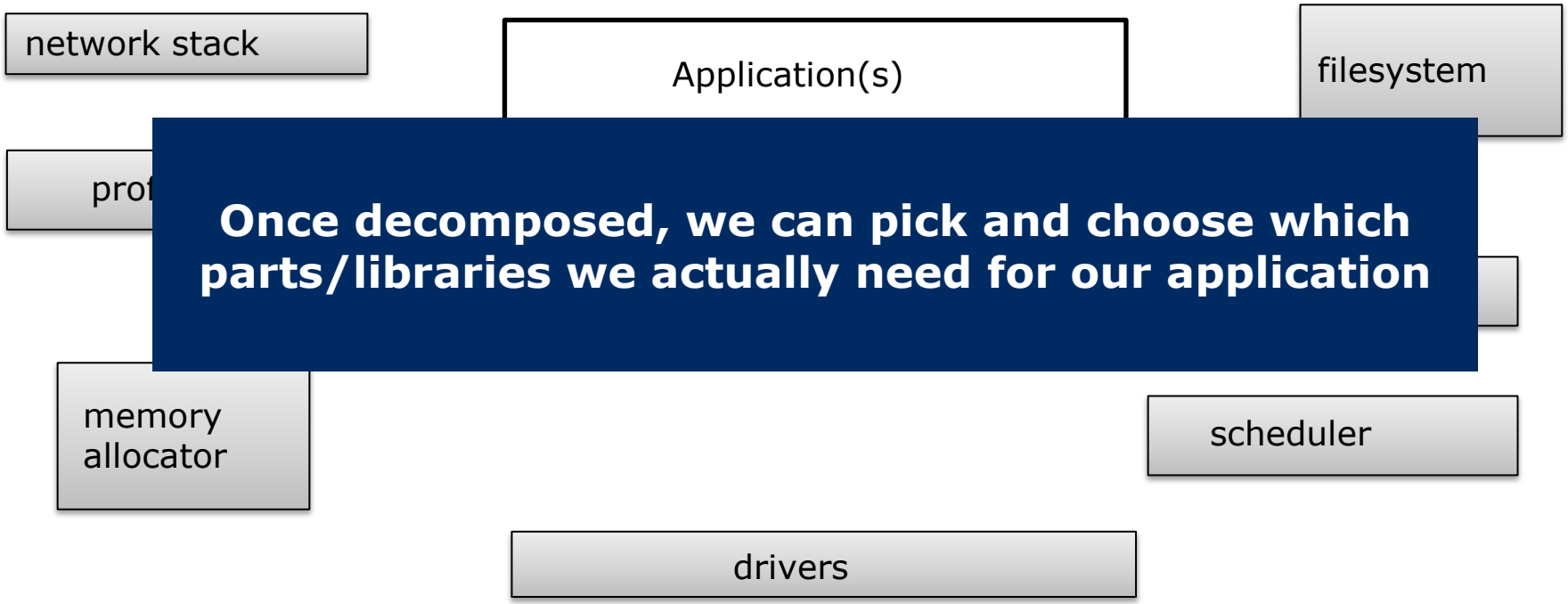
Recompose them to meet the needs of particular applications



The Unikraft Way

Decompose OS into a set of libraries (“Everything is a library”)

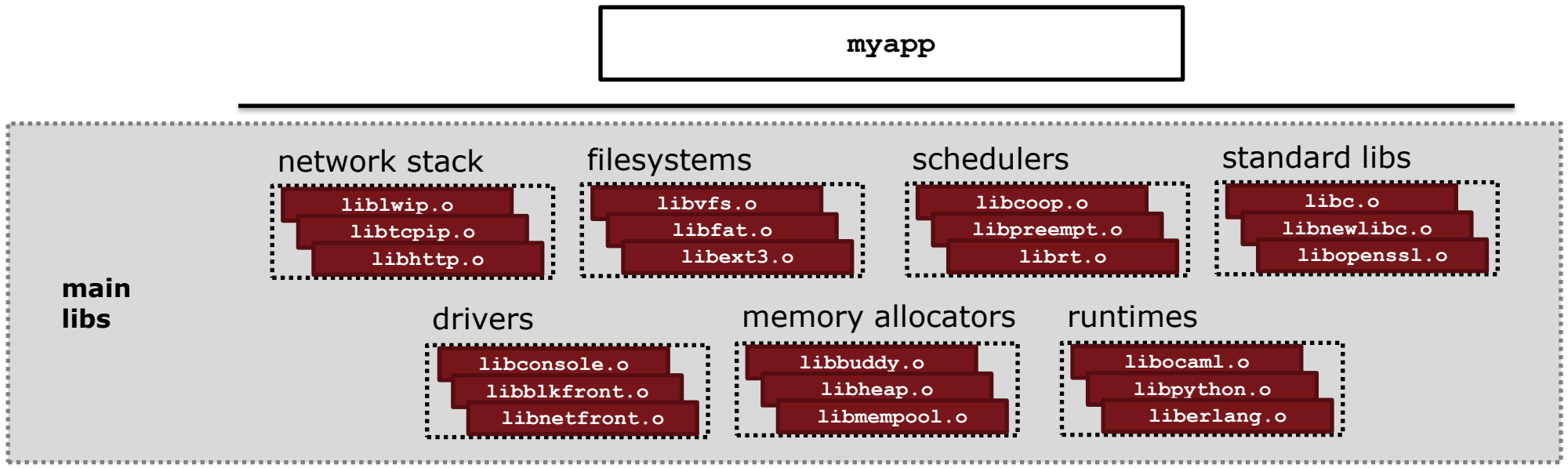
Recompose them to meet the needs of particular applications



Unikraft Overview – Everything as a Library

① **Select / build / port Application**

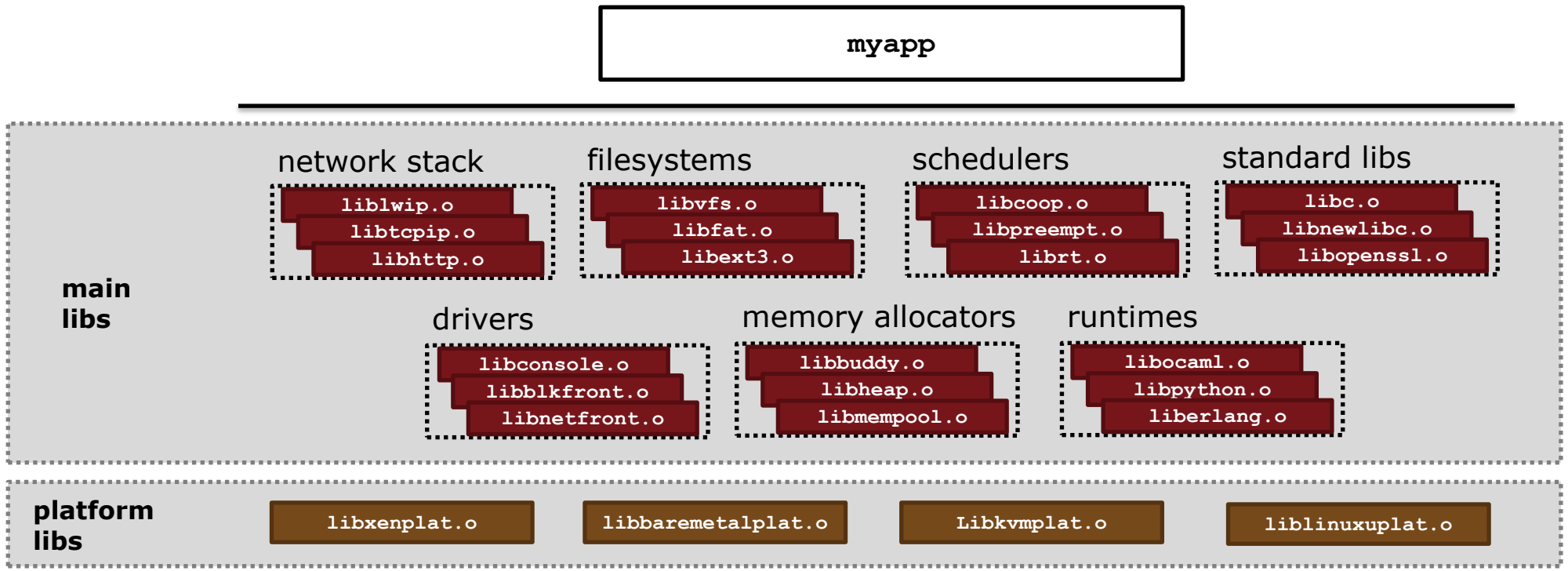
Unikraft Overview – Everything as a Library



1. **Select / build / port Application**

2. **Select and configure libraries**

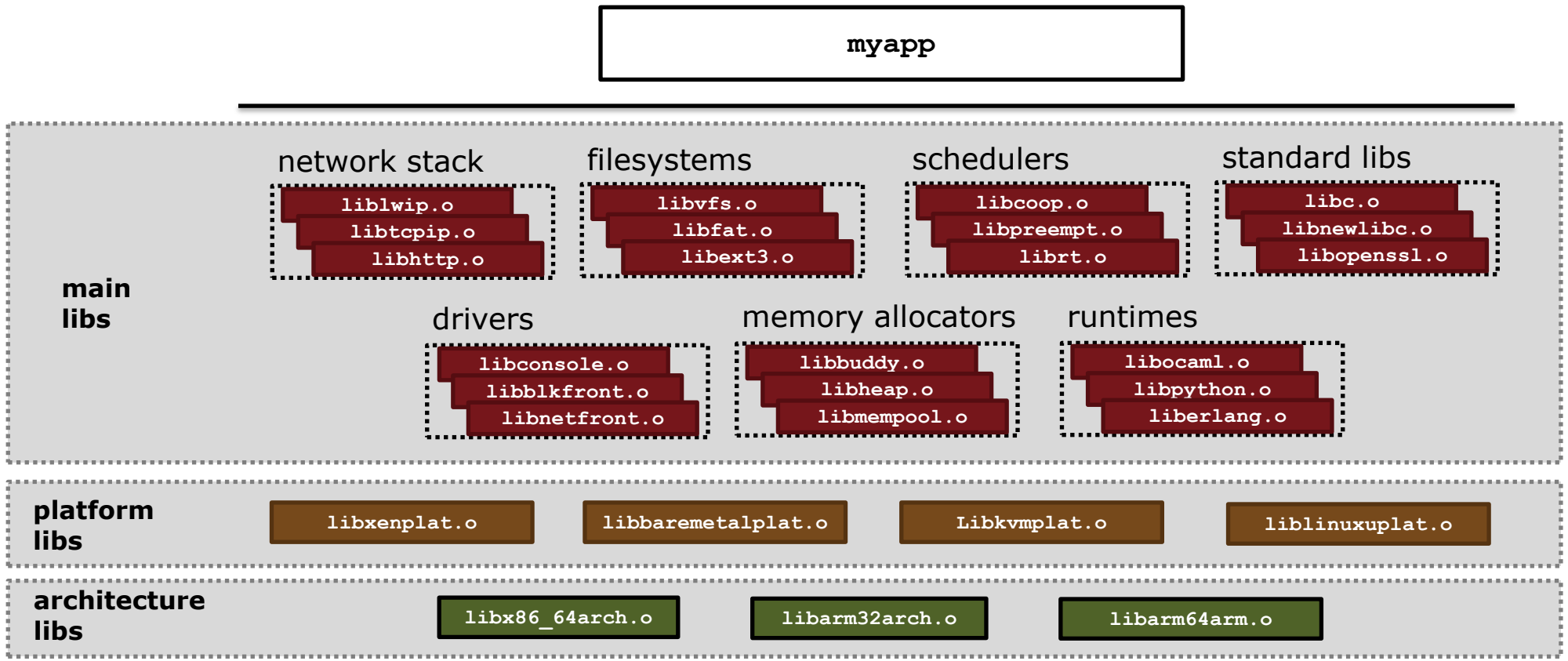
Unikraft Overview – Everything as a Library



① **Select / build / port Application**

② **Select and configure libraries**

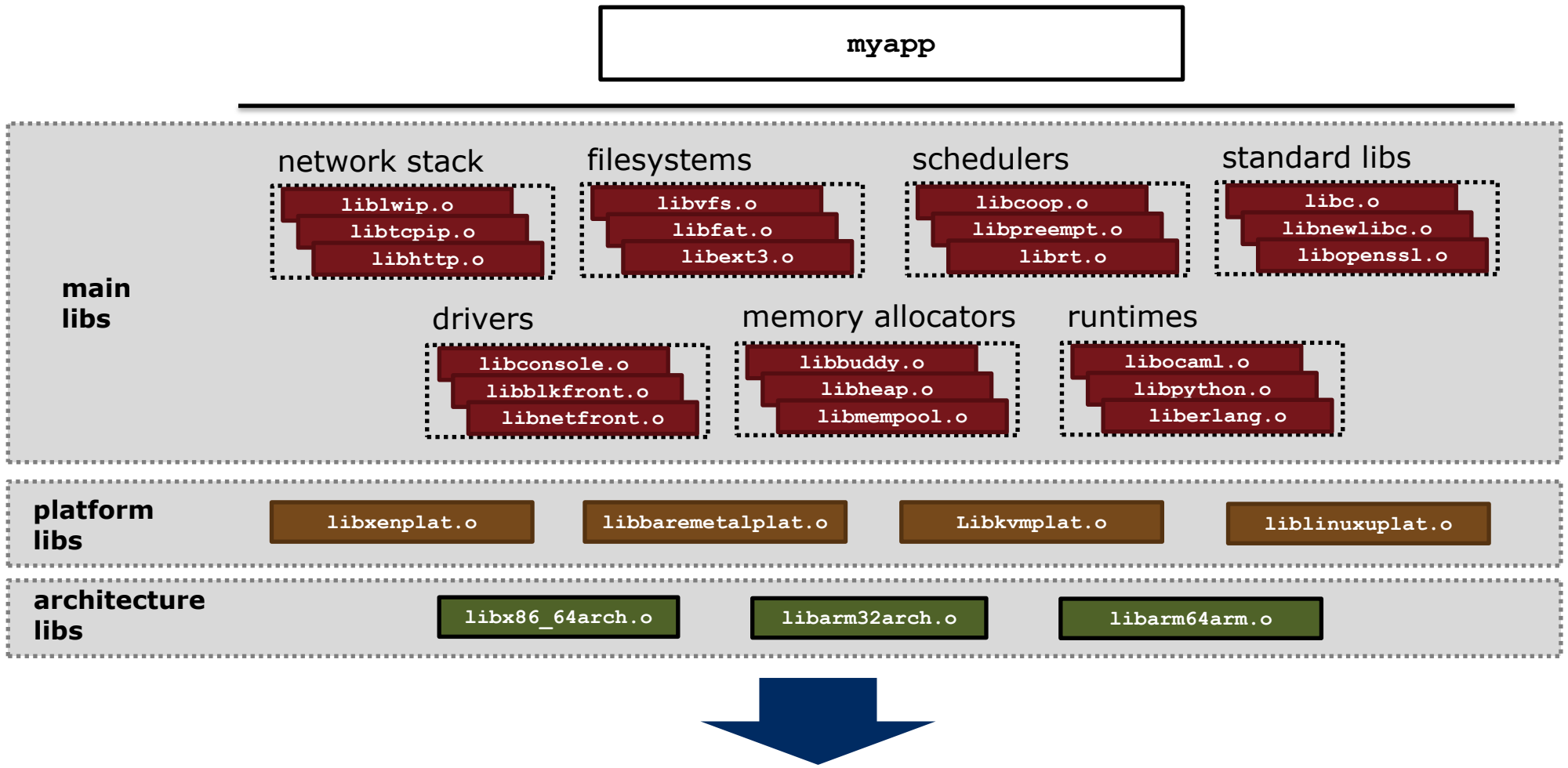
Unikraft Overview – Everything as a Library



① **Select / build / port Application**

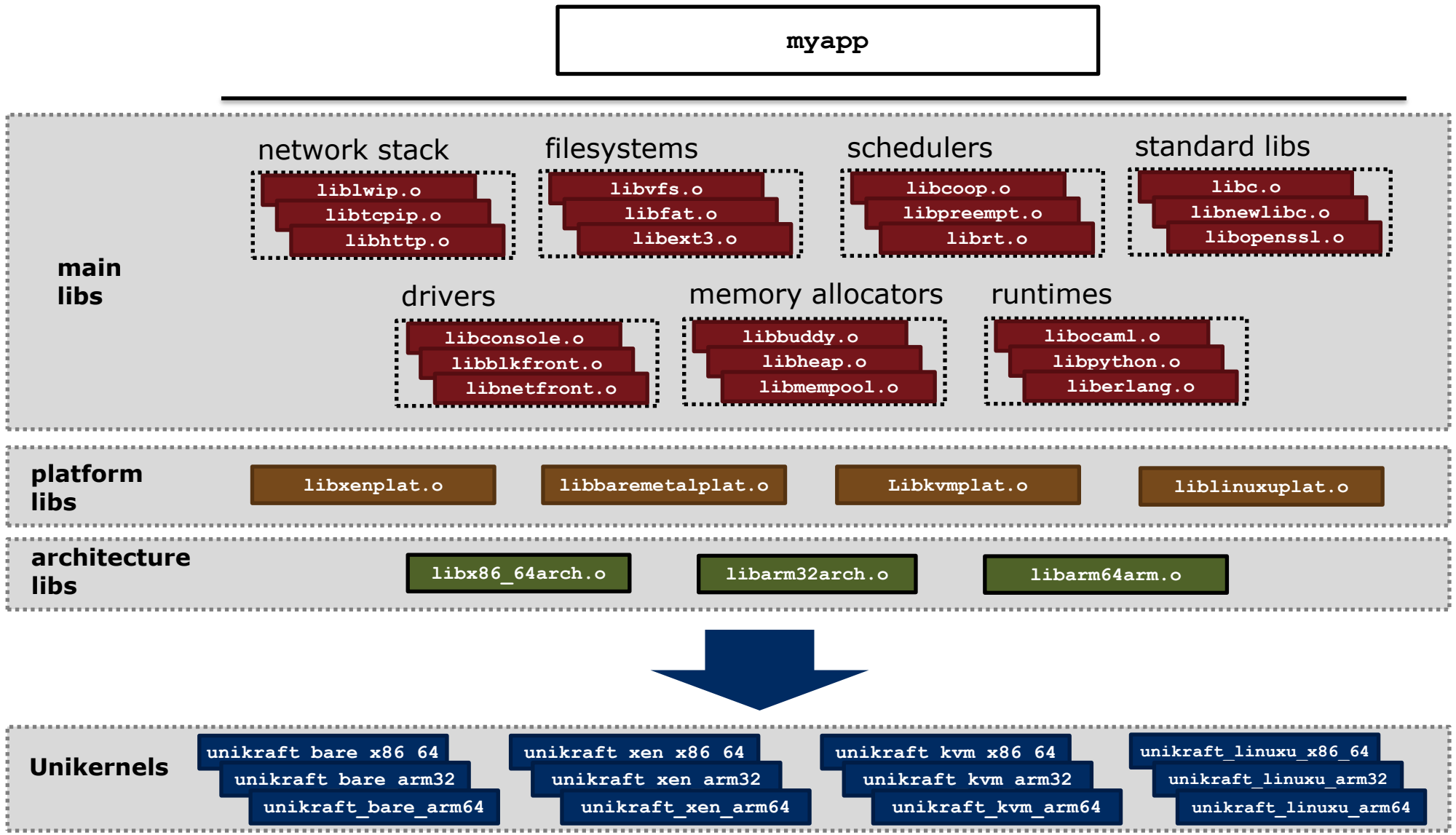
② **Select and configure libraries**

Unikraft Overview – Everything as a Library



1. **Select / build / port Application**
2. **Select and configure libraries**
3. **Build**

Unikraft Overview – Everything as a Library



Two Library Types

- **Built-in**: functionality specific to Unikraft, live in the main unikraft repo
  - ukboot
  - ukschedpreempt
  - …

Two Library Types

- **Built-in**: functionality specific to Unikraft, live in the main unikraft repo
  - ukboot
  - ukschedpreempt
  - …
- **External**: software projects external to Unikraft, have their own unikraft-lib repos
  - lwip
  - micropython
  - …

Example System

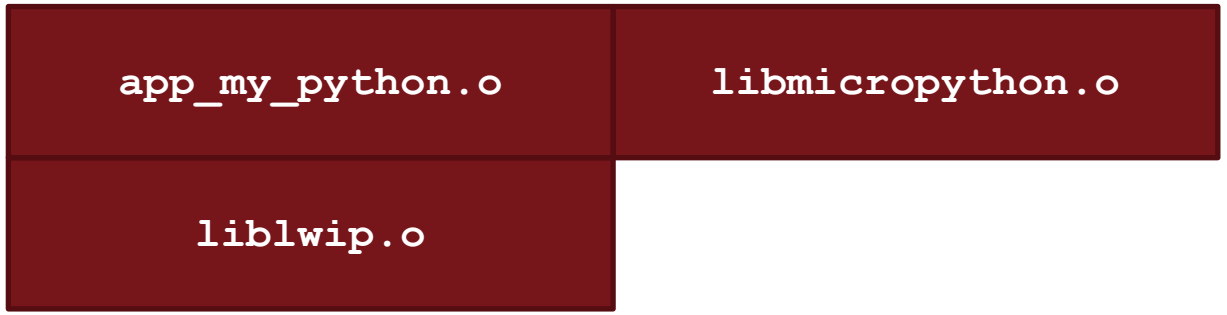
- Micropython Unikernel for Xen on x86_64

Example System

- Micropython Unikernel for Xen on x86_64

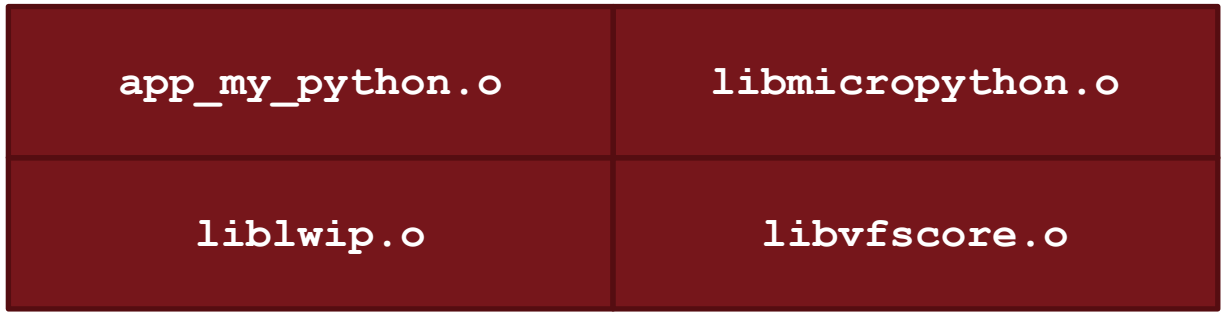
Example System

- Micropython Unikernel for Xen on x86_64



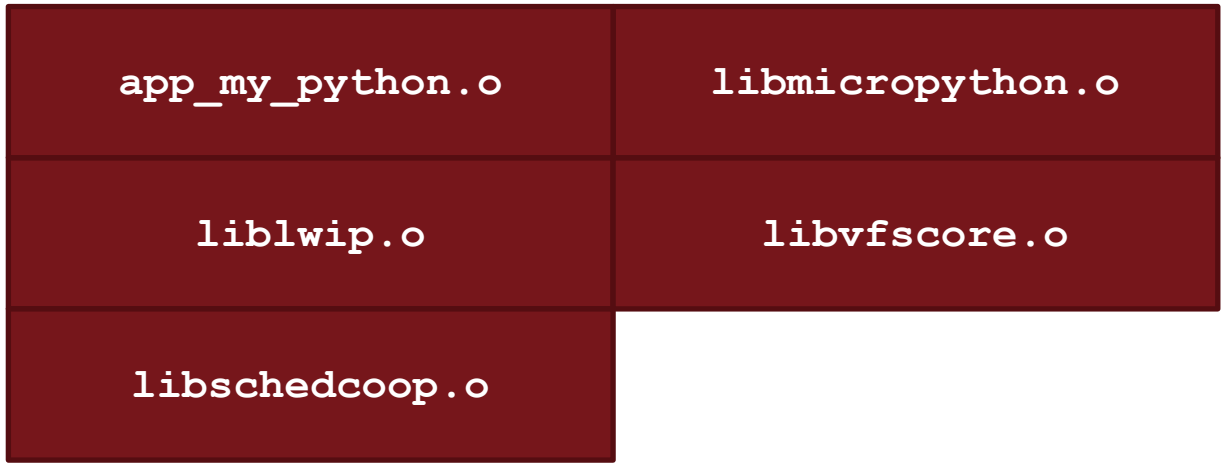
Example System

- Micropython Unikernel for Xen on x86_64



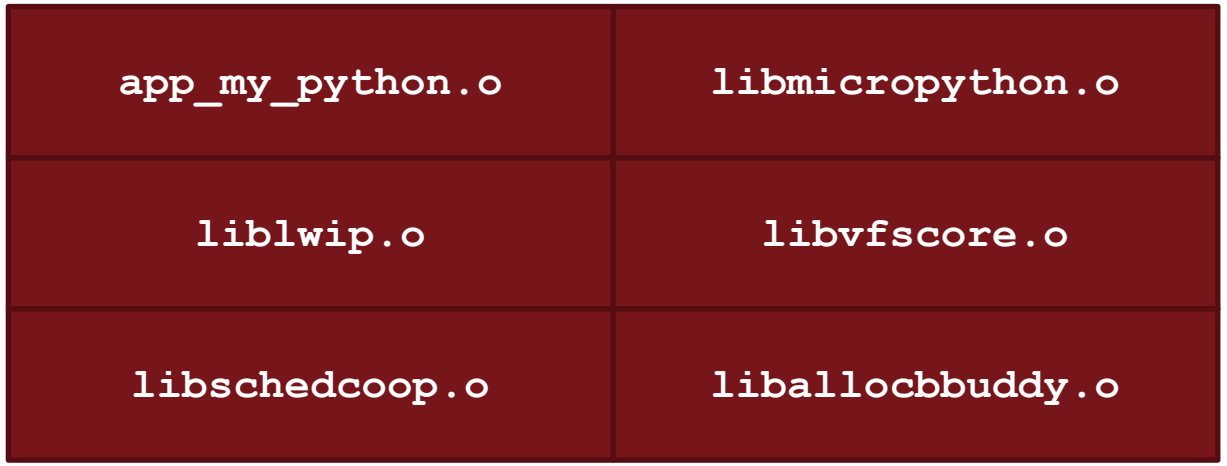
Example System

Micropython Unikernel for Xen on x86_64



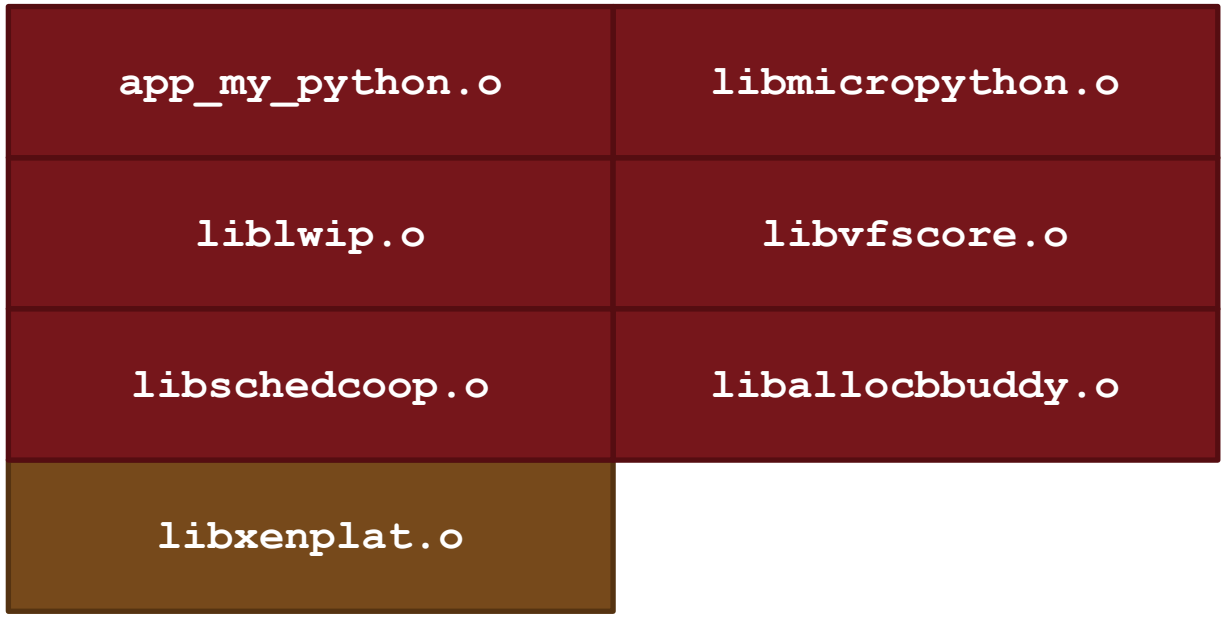
Example System

Micropython Unikernel for Xen on x86_64



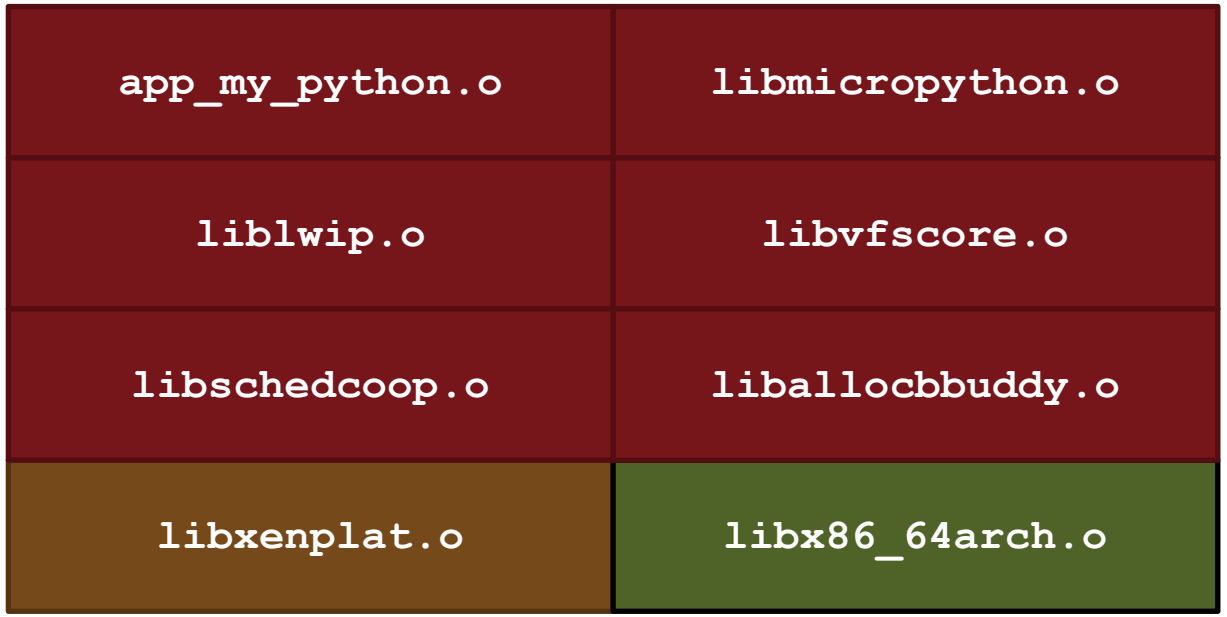
Example System

Micropython Unikernel for Xen on x86_64



Example System

Micropython Unikernel for Xen on x86_64

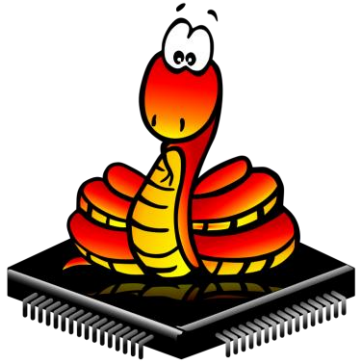


Example System

Micropython Unikernel for Xen on x86_64

app_my_python.o	libmicropython.o
liblwip.o	libvfscore.o
libschedcoop.o	liballocbbuddy.o
libxenplat.o	libx86_64arch.o

=



Unikernel

Putting Things Together – The Unikraft Build Tool

Putting Things Together – The Unikraft Build Tool

- Kconfig/Makefile based

Putting Things Together – The Unikraft Build Tool

Kconfig/Makefile based

```
make menuconfig
```

Putting Things Together – The Unikraft Build Tool

Kconfig/Makefile based

`make menuconfig`

- Choose options in the menu that you want for your application

Putting Things Together – The Unikraft Build Tool

Kconfig/Makefile based

`make menuconfig`

- Choose options in the menu that you want for your application
- Choose your target platform(s) (currently: Xen, KVM, Linux) and architectures

Putting Things Together – The Unikraft Build Tool

- Kconfig/Makefile based

  `make menuconfig`

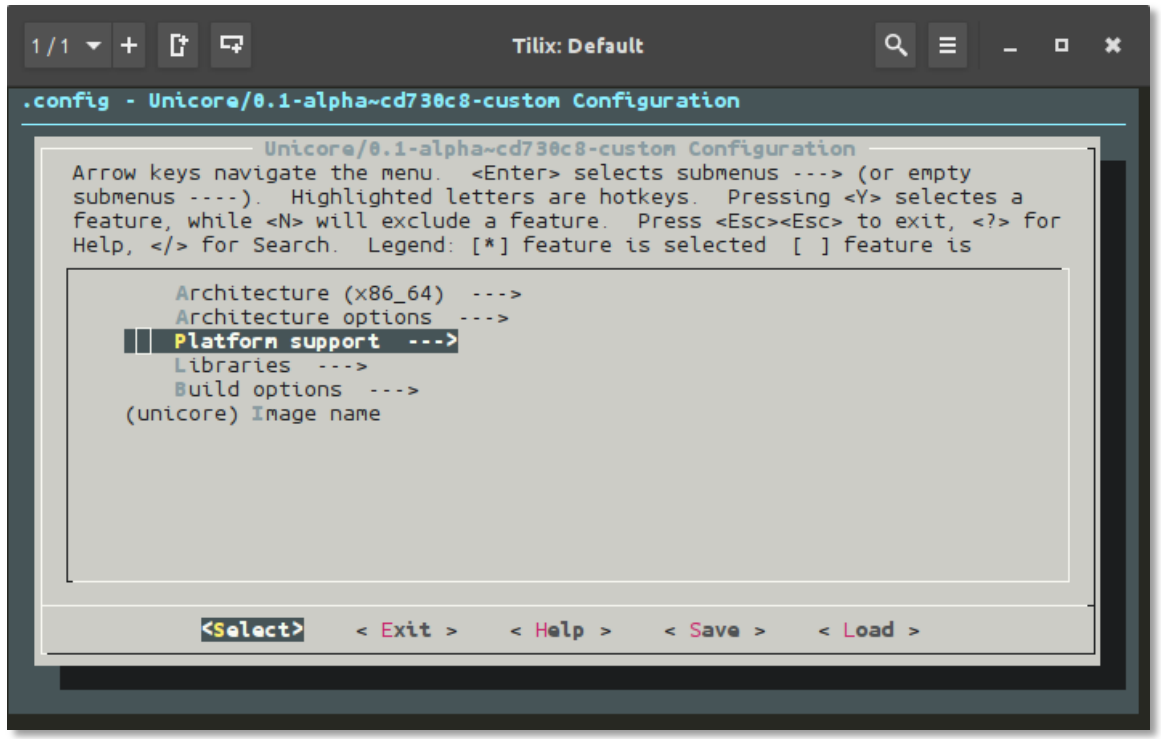
  - Choose options in the menu that you want for your application
  - Choose your target platform(s) (currently: Xen, KVM, Linux) and architectures
- Save config and `make`

Putting Things Together – The Unikraft Build Tool

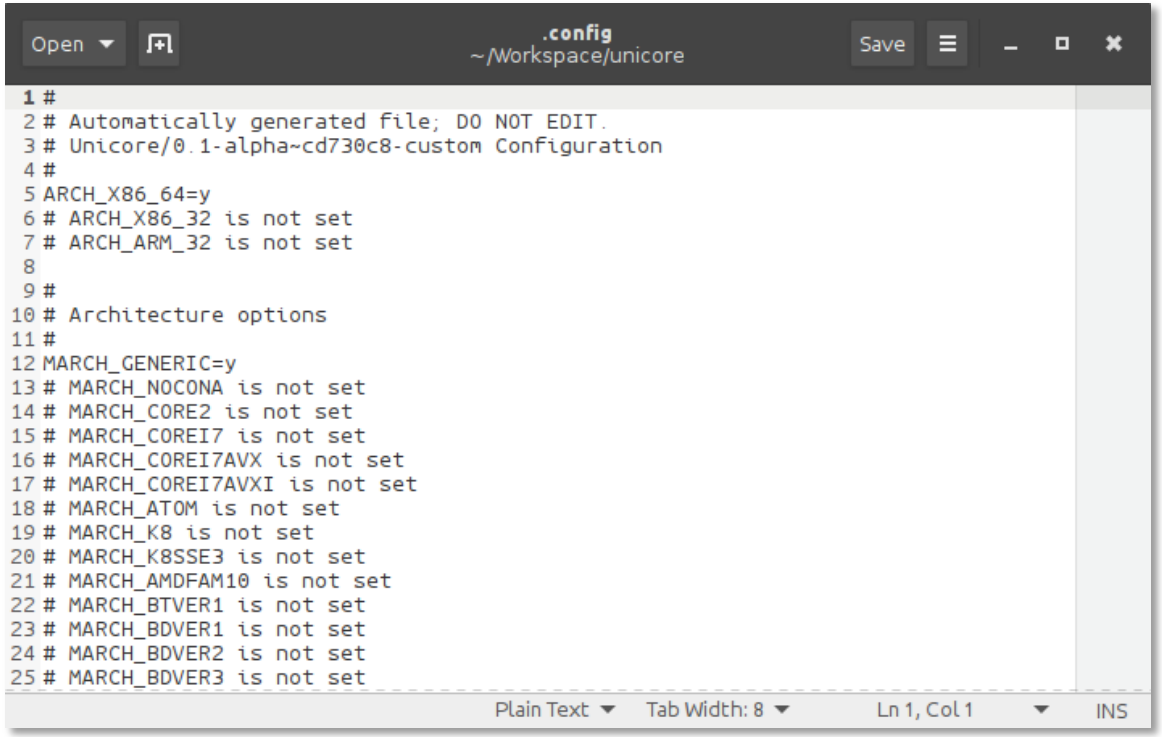
- Kconfig/Makefile based

  `make menuconfig`

  - Choose options in the menu that you want for your application
  - Choose your target platform(s) (currently: Xen, KVM, Linux) and architectures
- Save config and `make`



.config



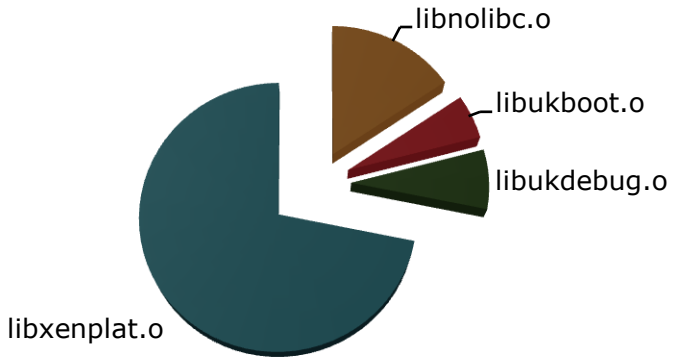
A Baseline Example…

- *Xen PV x86_64 binary*

A Baseline Example…

Xen PV x86_64 binary

`unikraft_xen-x86_64.o`



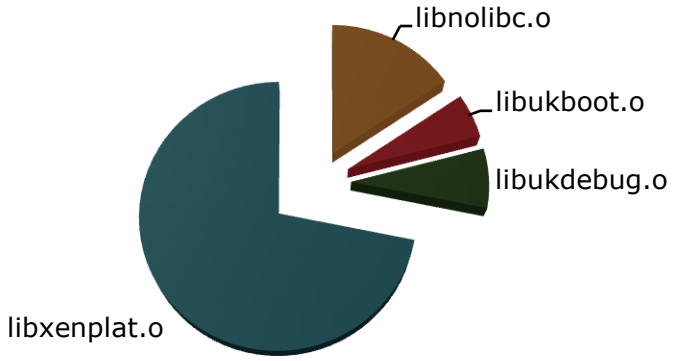
Final linking

`unikraft_xen-x86_64`
(32,7kB)

A Baseline Example…

Xen PV x86_64 binary

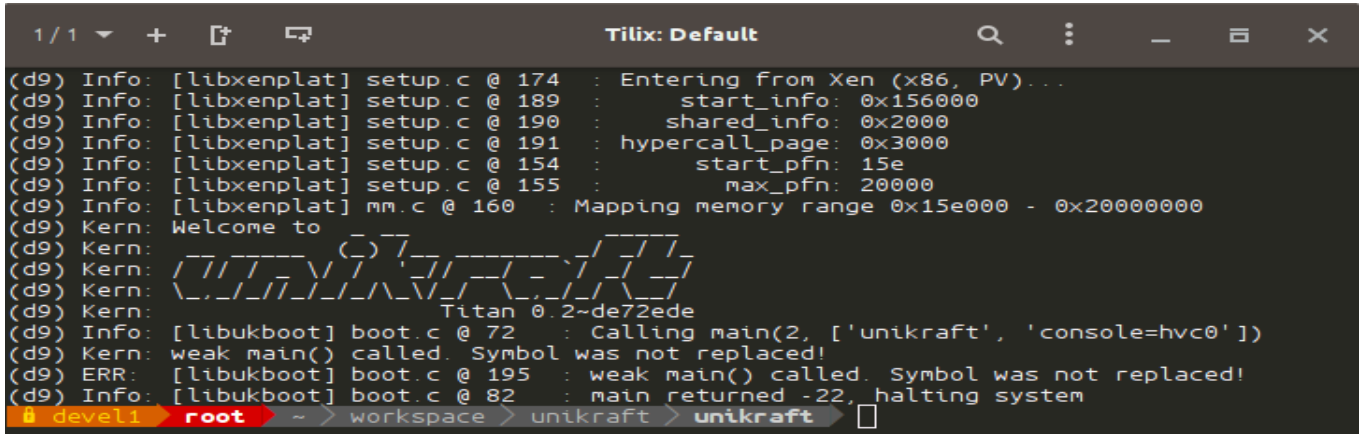
unikraft_xen-x86_64.o



Final linking

unikraft_xen-x86_64
(32,7kB)

Boots and prints messages to debug console (with min. 208kB RAM)



Building a Unikraft Hello World App

Repo Structure

- Clone the main Unikraft repo

```
git clone git://xenbits.xen.org/unikraft/unikraft.git
```

- Clone any external library repos

```
git clone git://xenbits.xen.org/unikraft/libs/newlib.git
```

- Create repo for the actual application

Repo Structure

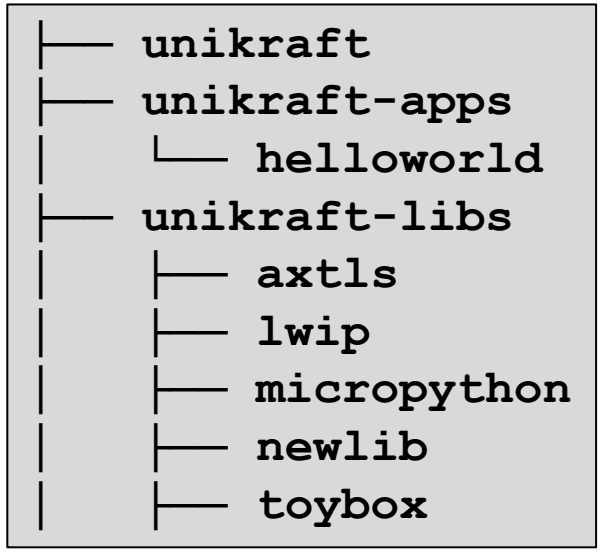
- Clone the main Unikraft repo

```
git clone git://xenbits.xen.org/unikraft/unikraft.git
```

- Clone any external library repos

```
git clone git://xenbits.xen.org/unikraft/libs/newlib.git
```

- Create repo for the actual application



Repo Structure

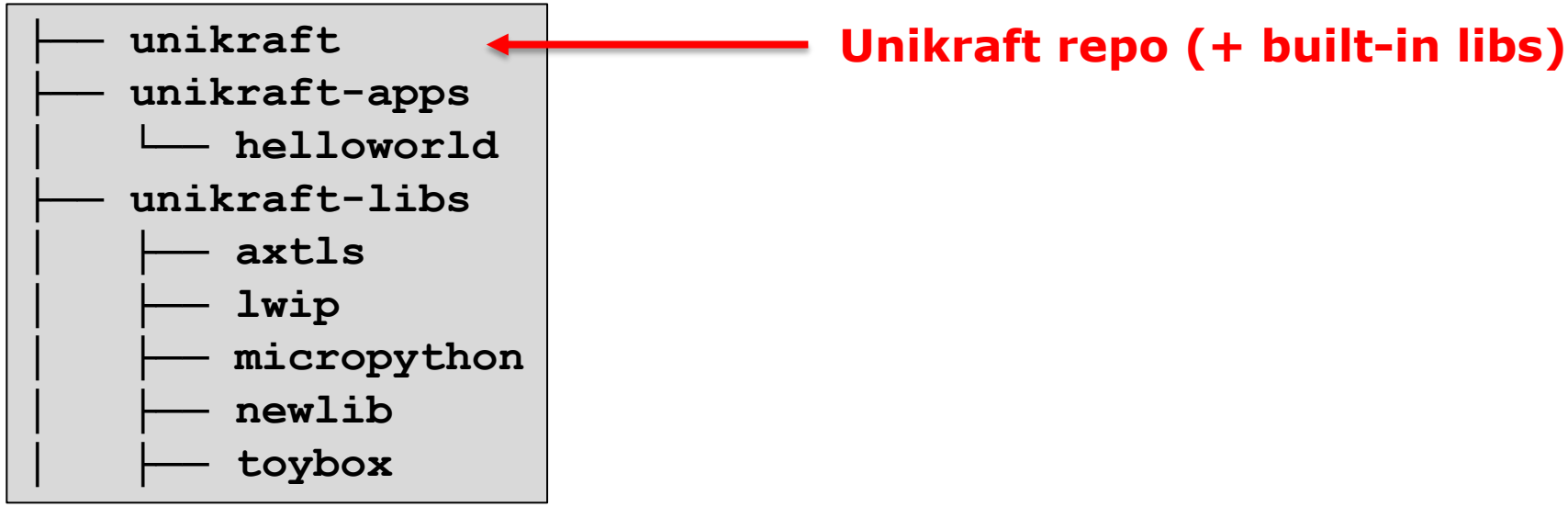
- Clone the main Unikraft repo

```
git clone git://xenbits.xen.org/unikraft/unikraft.git
```

- Clone any external library repos

```
git clone git://xenbits.xen.org/unikraft/libs/newlib.git
```

- Create repo for the actual application



Repo Structure

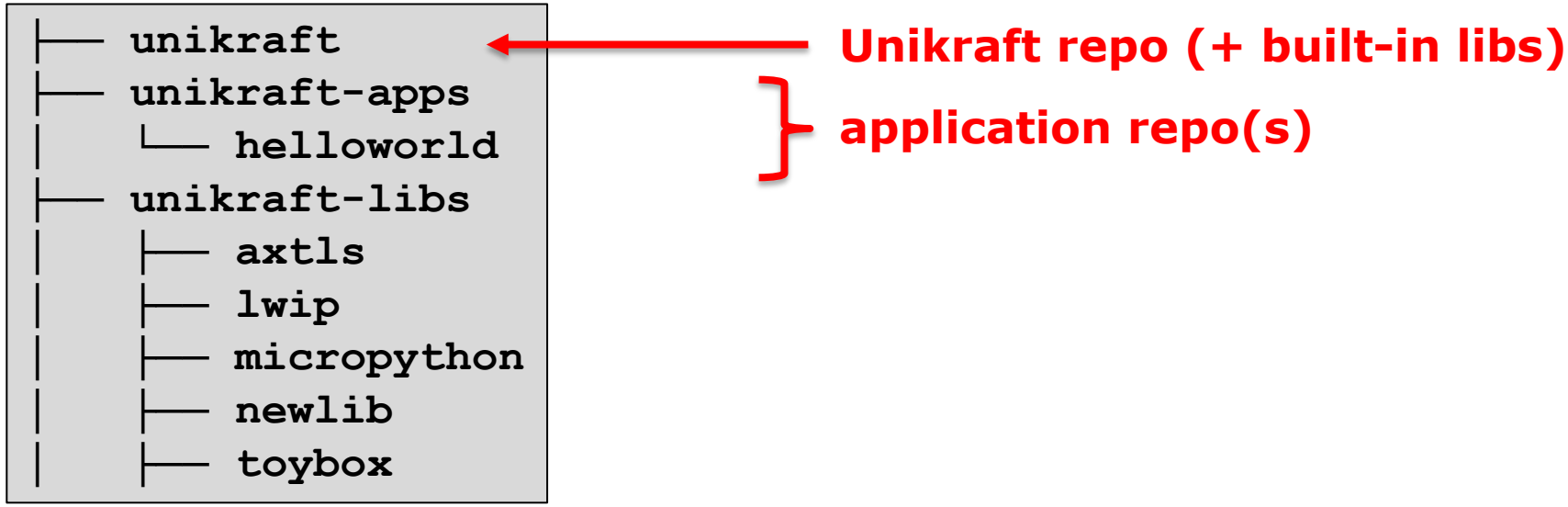
- Clone the main Unikraft repo

```
git clone git://xenbits.xen.org/unikraft/unikraft.git
```

- Clone any external library repos

```
git clone git://xenbits.xen.org/unikraft/libs/newlib.git
```

- Create repo for the actual application



Repo Structure

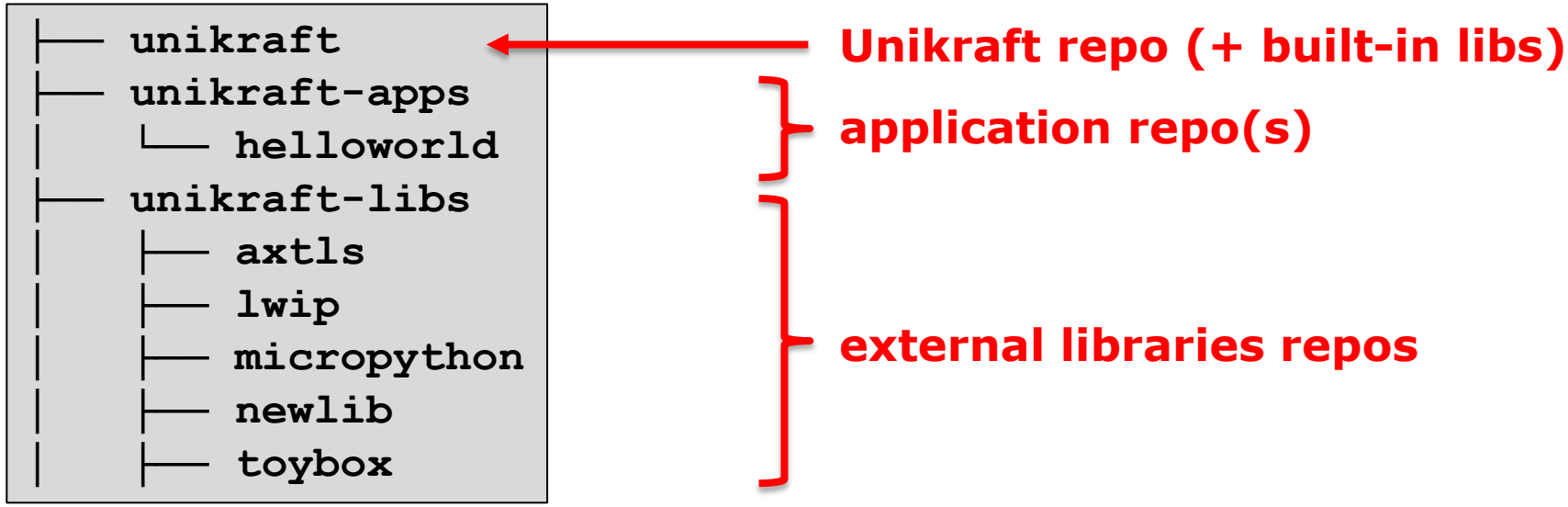
- Clone the main Unikraft repo

```
git clone git://xenbits.xen.org/unikraft/unikraft.git
```

- Clone any external library repos

```
git clone git://xenbits.xen.org/unikraft/libs/newlib.git
```

- Create repo for the actual application



Hello World – Four Required Files (I)

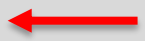
Makefile: specify where the main Unikraft repo is,
as well as repos for external libraries

```
UK_ROOT ?= $(PWD)/../../unikraft
UK_LIBS ?= $(PWD)/../../unikraft-libs
LIBS := $(UK_LIBS)/newlib

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

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

Hello World – Four Required Files (I)

Makefile: specify where the main Unikraft repo is,
as well as repos for external libraries

```
UK_ROOT ?= $(PWD)/../../unikraft
UK_LIBS ?= $(PWD)/../../unikraft-libs
LIBS := $(UK_LIBS)/newlib

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

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



path to unikraft repo

Hello World – Four Required Files (I)

Makefile: specify where the main Unikraft repo is,
as well as repos for external libraries

```
UK_ROOT ?= $(PWD)/../../unikraft          <-- path to unikraft repo
UK_LIBS ?= $(PWD)/../../unikraft-libs     <-- path to external libs
LIBS := $(UK_LIBS)/newlib

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

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

Hello World – Four Required Files (I)

Makefile: specify where the main Unikraft repo is,
as well as repos for external libraries

```
UK_ROOT ?= $(PWD)/../../unikraft
UK_LIBS ?= $(PWD)/../../unikraft-libs
LIBS := $(UK_LIBS)/newlib

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

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

← path to unikraft repo

← path to external libs

← external libs needed

Hello World – Four Required Files (II)

Makefile.uk: specifies the sources to build for the application

```
$(eval $(call addlib,apphelloworld))

APPHELLOWORLD_SRCS-y += $(APPHELLOWORLD_BASE)/main.c
```

Hello World – Four Required Files (II)

Makefile.uk: specifies the sources to build for the application

```
$(eval $(call addlib,apphelloworld))

APPHELLOWORLD_SRCS-y += $(APPHELLOWORLD_BASE)/main.c
```


register app with unikraft build system

Hello World – Four Required Files (II)

Makefile.uk: specifies the sources to build for the application

```
$(eval $(call addlib,apphelloworld))

APPHELLOWORLD_SRCS-y += $(APPHELLOWORLD_BASE)/main.c
```

register app with unikraft build system

Add main.c to build

Hello World – Four Required Files (III)

Config.uk: to populate Unikraft's menu with application-specific option

```
### 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
```

Hello World – Four Required Files (IV)

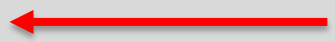
main.c: source file to provide (at least) a `main()` function

```
#include <stdio.h>
/* Import user configuration: */
#include <uk/config.h>

int main(int argc, char *argv[])
{
	printf("Hello world!\n");
#if CONFIG_APPHELLOWORLD_PRINTARGS
	int i;
	printf("Arguments:s");
	for (i=0; i<argc; ++i)
		printf(" \"%s\"", argv[i]);
	printf("\n");
#endif
}
```

Hello World – Four Required Files (IV)

main.c: source file to provide (at least) a `main()` function

```
#include <stdio.h>
/* Import user configuration: */
#include <uk/config.h>

int main(int argc, char *argv[])
{
        printf("Hello world!\n");
#if CONFIG_APPHELLOWORLD_PRINTARGS
        int i;
        printf("Arguments:s");
        for (i=0; i<argc; ++i)
                printf(" \"%s\"", argv[i]);
        printf("\n");
#endif
}
```



Unikernel entry point after boot

Hello World – Four Required Files (IV)

main.c: source file to provide (at least) a `main()` function

```
#include <stdio.h>
/* Import user configuration: */
#include <uk/config.h>

int main(int argc, char *argv[])
{
        printf("Hello world!\n");
#if CONFIG_APPHELLOWORLD_PRINTARGS
        int i;
        printf("Arguments:s");
        for (i=0; i<argc; ++i)
                printf(" \"%s\"", argv[i]);
        printf("\n");
#endif
}
```

Unikernel entry point after boot

defined by Config.uk

Porting an External Library

How To Port an External Library

How To Port an External Library

- Write **`Makefile.uk`** and add the external library source files to it
  - The library's original Makefile can serve as a template
  - I'll show how to in a second

How To Port an External Library

- Write **`Makefile.uk`** and add the external library source files to it
  - The library's original Makefile can serve as a template
  - I'll show how to in a second
- Write **`Config.uk`** with library-specific options

How To Port an External Library

Write `Makefile.uk` and add the external library source files to it

- The library's original Makefile can serve as a template
- I'll show how to in a second

Write `Config.uk` with library-specific options

- Isn't strictly required
- But at least a simple "library on/off" option for kbuild is a good idea

How To Port an External Library

Write **`Makefile.uk`** and add the external library source files to it

- The library's original Makefile can serve as a template
- I'll show how to in a second

Write **`Config.uk`** with library-specific options

- Isn't strictly required
- But at least a simple "library on/off" option for kbuild is a good idea

Sometimes a bit of *glue* code is needed

- E.g., in newlib to link POSIX thread creation to Unikraft's thread library

How To Port an External Library

- Write **`Makefile.uk`** and add the external library source files to it
  - The library's original Makefile can serve as a template
  - I'll show how to in a second
- Write **`Config.uk`** with library-specific options
  - Isn't strictly required
  - But at least a simple "library on/off" option for kbuild is a good idea
- Sometimes a bit of *glue* code is needed
  - E.g., in newlib to link POSIX thread creation to Unikraft's thread library
- In this early phase: implement core unikraft functionality that is required to support your library, for example:
  - File descriptors/sockets
  - Threading support

How To Port an External Library – Makefile.uk

```
##############################################################################
# Library registration
##############################################################################
$(eval $(call addlib_s,libaxtls,$(LIBAXTLS)))
```

How To Port an External Library – Makefile.uk

```
##########################################################################
# Library registration
##########################################################################
$(eval $(call addlib_s,libaxtls,$(LIBAXTLS)))
```

← **Register library with unikraft build system**

How To Port an External Library – Makefile.uk

```
##########################################################################
# Library registration
##########################################################################
$(eval $(call addlib_s,libaxtls,$(LIBAXTLS)))

##########################################################################
# Source Download
##########################################################################
# Nothing here: sources are small and included directly in the uk library repo
```

← **Register library with unikraft build system**

How To Port an External Library – Makefile.uk

```
##############################################################################
# Library registration
##############################################################################
$(eval $(call addlib_s,libaxtls,$(LIBAXTLS)))

##############################################################################
# Source Download
##############################################################################
$(eval $(call fetch,libnewlibc,$(LIBNEWLIB_URL)))
$(eval $(call patch,libnewlibc,$(LIBNEWLIB_PATCHDIR),newlib-$(LIBNEWLIB_VERSION)))
```

← **Register library with unikraft build system**

← **Download and patch library code**

How To Port an External Library – Makefile.uk

```
##############################################################################
# Library registration
##############################################################################
$(eval $(call addlib_s,libaxtls,$(LIBAXTLS)))

##############################################################################
# Source Download
##############################################################################
$(eval $(call fetch,libnewlibc,$(LIBNEWLIB_URL)))
$(eval $(call patch,libnewlibc,$(LIBNEWLIB_PATCHDIR),newlib-$(LIBNEWLIB_VERSION)))

##############################################################################
# Library includes
##############################################################################
CINCLUDES-y +=    -I$(LIBAXTLS_BASE)/include \
                  -I$(LIBAXTLS_BASE)/crypto  \
                  -I$(LIBAXTLS_BASE)/ssl
```

← **Register library with unikraft build system**

← **Download and patch library code**

← **The library's original include directories**

How To Port an External Library – Makefile.uk

```
##############################################################################
# Library registration
##############################################################################
$(eval $(call addlib_s,libaxtls,$(LIBAXTLS)))

##############################################################################
# Source Download
##############################################################################
$(eval $(call fetch,libnewlibc,$(LIBNEWLIB_URL)))
$(eval $(call patch,libnewlibc,$(LIBNEWLIB_PATCHDIR),newlib-$(LIBNEWLIB_VERSION)))

##############################################################################
# Library includes
##############################################################################
CINCLUDES-y +=    -I$(LIBAXTLS_BASE)/include \
                  -I$(LIBAXTLS_BASE)/crypto  \
                  -I$(LIBAXTLS_BASE)/ssl

##############################################################################
# sources
##############################################################################
LIBAXTLS_SRCS-y += $(LIBAXTLS_BASE)/crypto/aes.c
LIBAXTLS_SRCS-y += $(LIBAXTLS_BASE)/crypto/bigint.c
…
LIBAXTLS_SRCS-y += $(LIBAXTLS_BASE)/crypto/sha512.c
```

← **Register library with unikraft build system**

← **Download and patch library code**

← **The library's original include directories**

← **The library's original source files**

Unikraft 0.2 Titan

Current Status

Available Libraries

Core Libraries

- libfdt
  - Flat device tree parser
- libnolibc
  - A tiny libc replacement
- libukalloc
  - Memory allocator abstraction
- libukallocbbuddy
  - Binary buddy allocator
- libukargparse
  - Argument parser library
- libukboot
  - Unikraft bootstrapping
- libukdebug
  - Debug and kernel printing
  - Assertions, hexdump
- libuksched
  - Scheduler abstraction
- libukschedcoop
  - Cooperative scheduler
- libukbus
  - abstraction for device buses, e.g., PCI
- libuklock
  - mutexes and semaphores
- libukmpi
  - message-passing interface
- libukswrand
  - pseudo-RNG interface
- libuktimeconv
  - time calculation/conversion
- libvfscore
  - basic file descriptor management / mapping / handling

External Libraries

- libnewlib
  - libc originally aimed at embedded devices
- liblwip
  - lightweight TCP/IP stack

Architecture Libraries

- libarmmath
  - 64bit arithmetic on ARMv7

Platform Libraries

- libxenplat
  - Xen (PV)
  - x86_64, ARMv7
- libkvmplat
  - QEMU/kvm
  - x86_64
- liblinuxu
  - Linux userspace
  - x86_64, ARMv7

Coming soon: in the pipeline

Core Libraries

- libukschedpreempt
  - Pre-emptive scheduler

External Libraries

- libclick
  - Click modular router (e.g., for NFV)
- libaxtls
  - TLS support aimed at embedded devices
- libmicropython
  - Python implemented for microcontrollers

Platform Libraries

- libxenplat
  - ARM64 support(?)
  - netfront support
- libkvmplat
  - ARM64 support
  - virtio-net support
- liblinuxu
  - tap device based networking support

Talk by Kaly (ARM) later today!

The road ahead

- Project got accepted as incubator project in December
- Initially, mostly internal contributors

- Currently contributions in the pipeline from contributors from
  - Romania (netfront, scheduling; from University Politehnica Bucharest)
  - Israel (bare-metal support)
  - China (ARM64 support; from ARM)

- We welcome additional contributors!

- Resources:
  - Code: xenbits.org/unikraft/*
  - On-line documentation: unikraft.org
  - IRC: #unikraft @ freenode
  - Mailing list: shared with minios: minios-devel@lists.xen.org


Xen
Project
DEVELOPER AND DESIGN
SUMMIT