A Binary Compatible Unikernel

Pierre Olivier*, Daniel Chiba*, Stefan Lankes+, Changwoo Min*, Binoy Ravidnran*

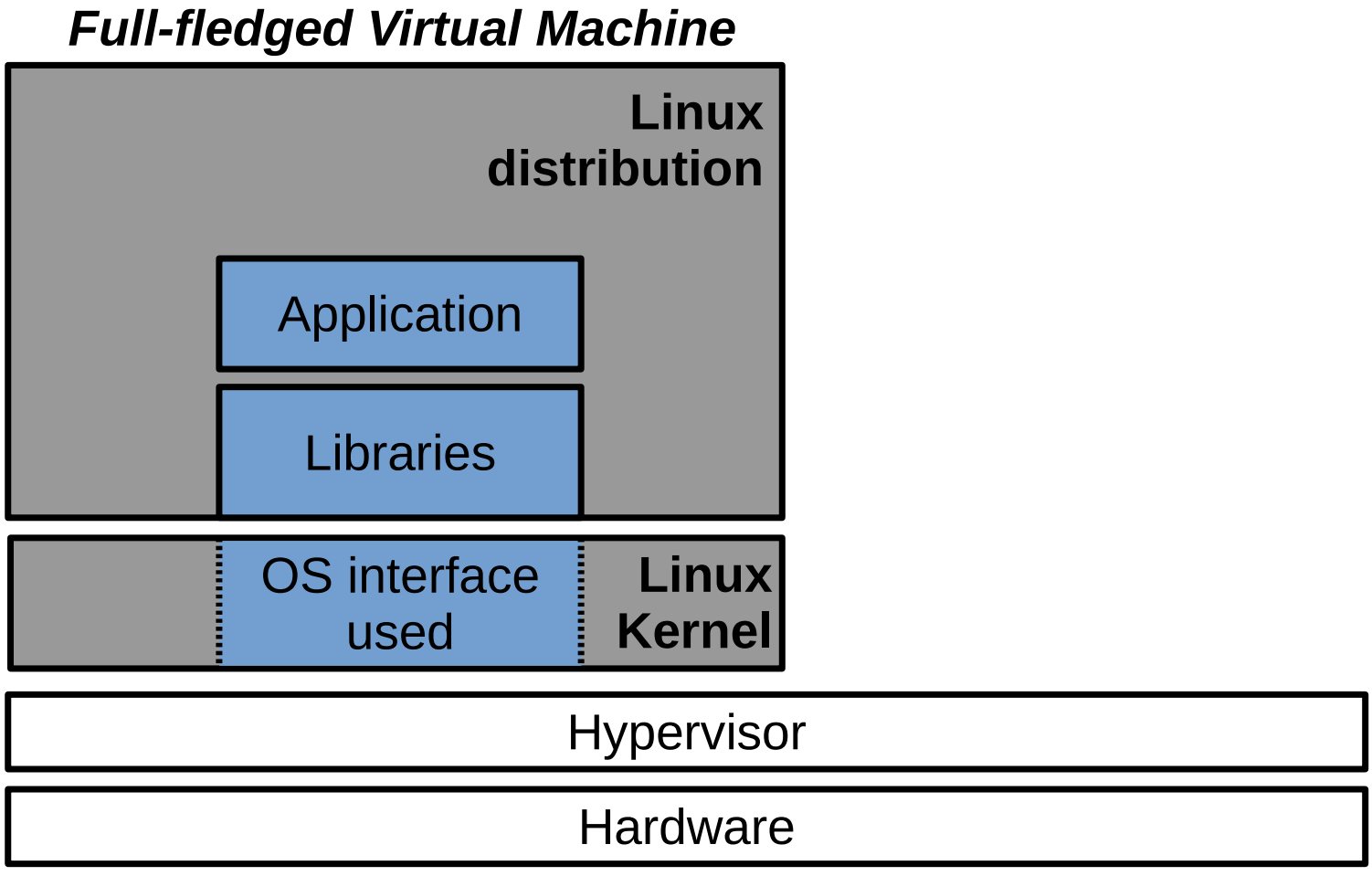
*Virginia Tech, +RWTH Aachen University

VEE'19 - 04/14/2019

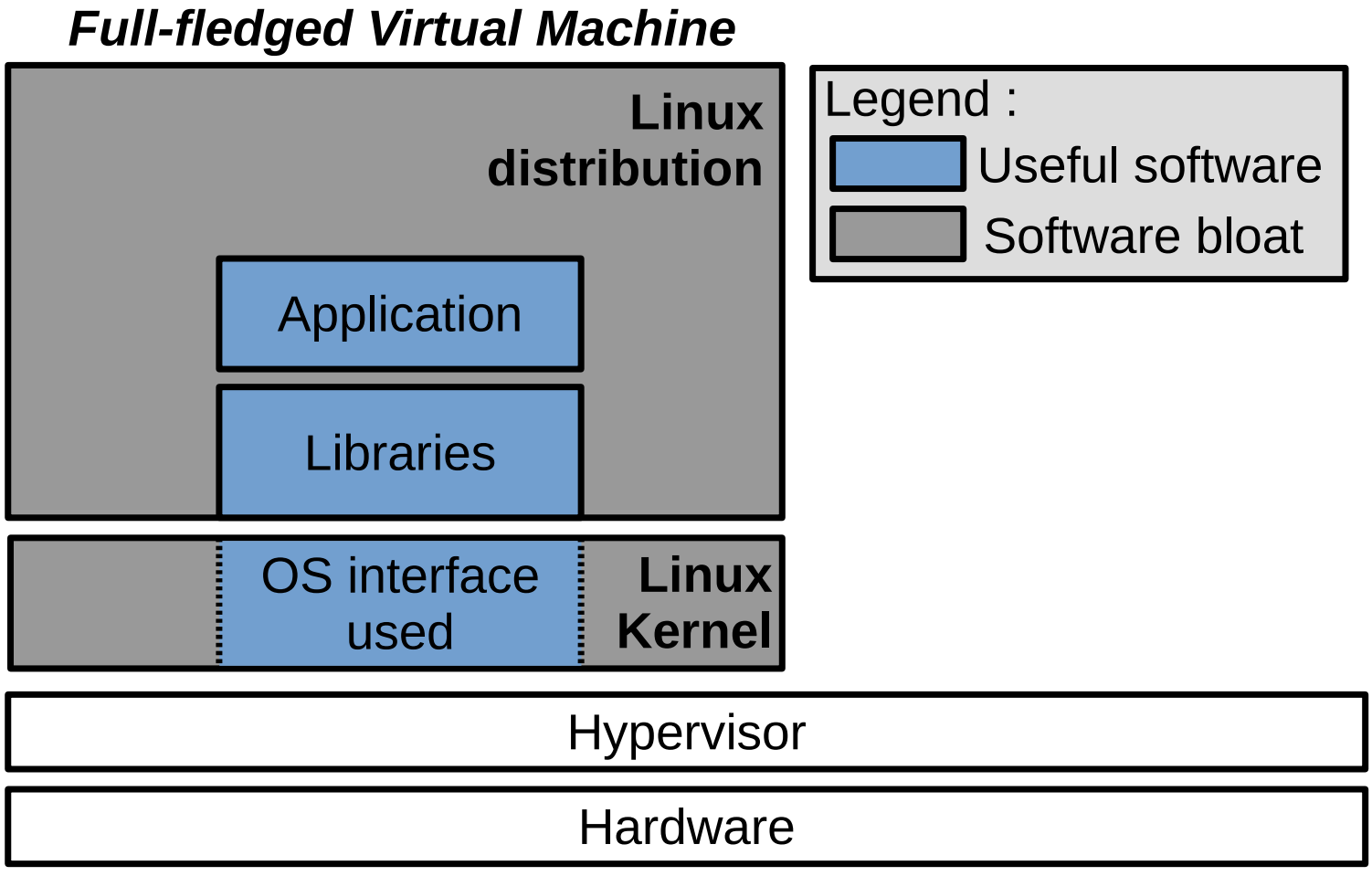
ece Bradley Department of Electrical & Computer Engineering

1872 VirginiaTech

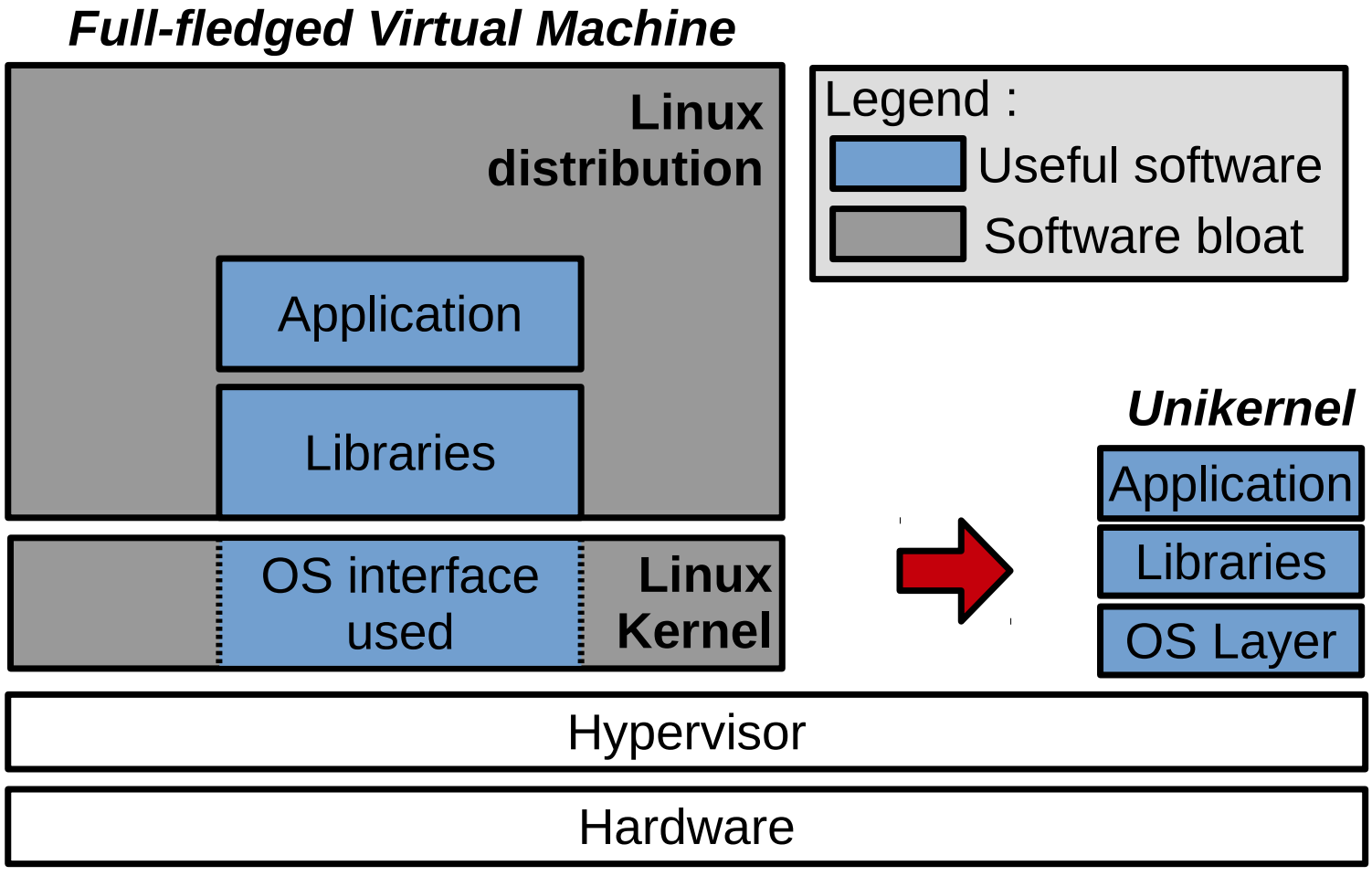
Unikernels
Presentation



Unikernels
Presentation



Unikernels
Presentation



Unikernels
Presentation (2)

Unikernel: application + dependencies + thin OS compiled as a static binary running on top of a hypervisor

Unikernels
Presentation (2)

Unikernel: application + dependencies + thin OS compiled as a static binary running on top of a hypervisor

- **single-***
  - Single purpose: run 1 application
  - Single process
  - Single binary and single address space for application + kernel
    - No user/kernel protection needed

Unikernels
Presentation (2)

Unikernel: application + dependencies + thin OS compiled as a static binary running on top of a hypervisor

- **single-***
  - Single purpose: run 1 application
  - Single process
  - Single binary and single address space for application + kernel
    - No user/kernel protection needed
- **Lightweight virtualization, alternative to containers**
  - Security advantage: small attack surface and high isolation

Unikernels
Presentation (2)

Unikernel: application + dependencies + thin OS compiled as a static binary running on top of a hypervisor

- **single-***
  - Single purpose: run 1 application
  - Single process
  - Single binary and single address space for application + kernel
    - No user/kernel protection needed
- **Lightweight virtualization, alternative to containers**
  - Security advantage: small attack surface and high isolation
- **Per-application tailored kernel**
  - LibOS/Exokernel model

Unikernels
Presentation (2)

Unikernel: application + dependencies + thin OS compiled as a static binary running on top of a hypervisor

- **single-***
  - Single purpose: run 1 application
  - Single process
  - Single binary and single address space for application + kernel
    - No user/kernel protection needed
- **Lightweight virtualization, alternative to containers**
  - Security advantage: small attack surface and high isolation
- **Per-application tailored kernel**
  - LibOS/Exokernel model
- **Reduced OS noise, increased performance**
  - Low system call latency
    - App + kernel in ring 0, system calls are function calls

Unikernels
The Issue

- Unikernels have plenty of benefits to bring
- Unikernels have plenty of application domains
- They are very popular in academia ...
- **...but why (nearly) nobody uses them in the industry?**

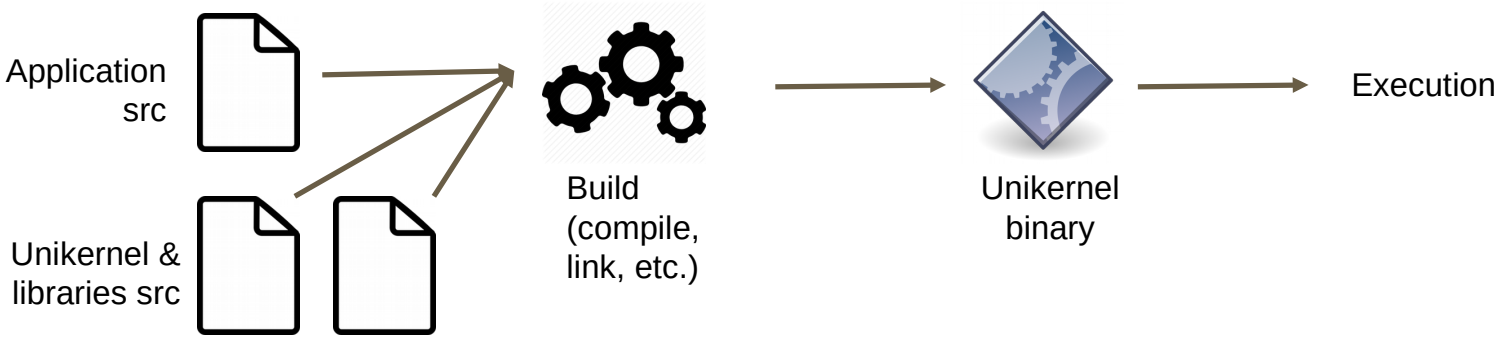
Unikernels
The Issue

- Unikernels have plenty of benefits to bring
- Unikernels have plenty of application domains
- They are very popular in academia ...
- **...but why (nearly) nobody uses them in the industry?**

Because it is hard to port existing applications!

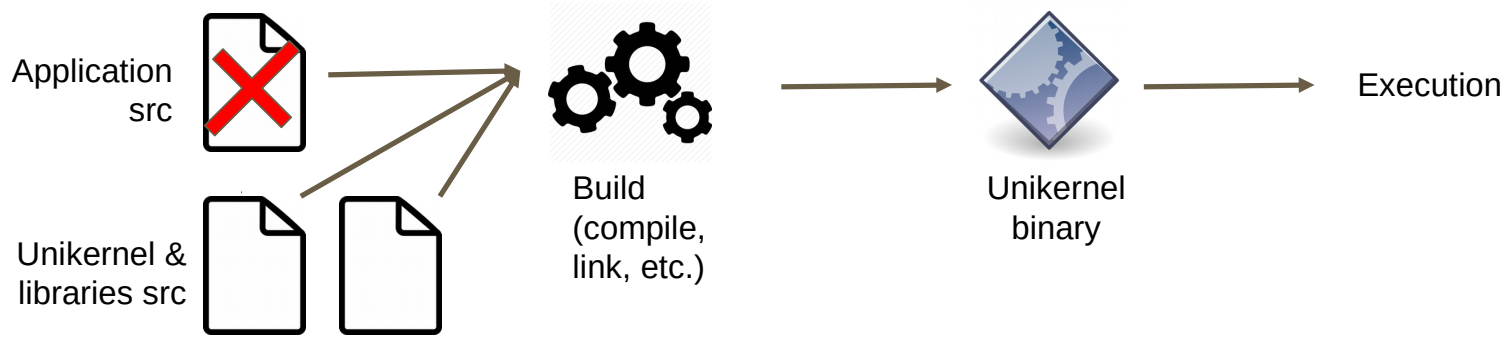
Unikernels

The Issue: Porting to Unikernels



Unikernels

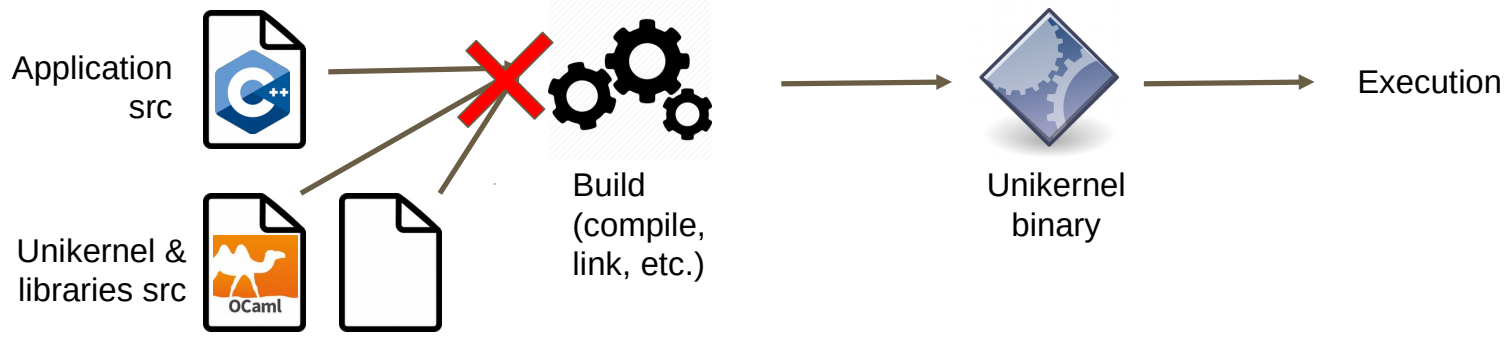
The Issue: Porting to Unikernels



- Proprietary software → source code not available

Unikernels

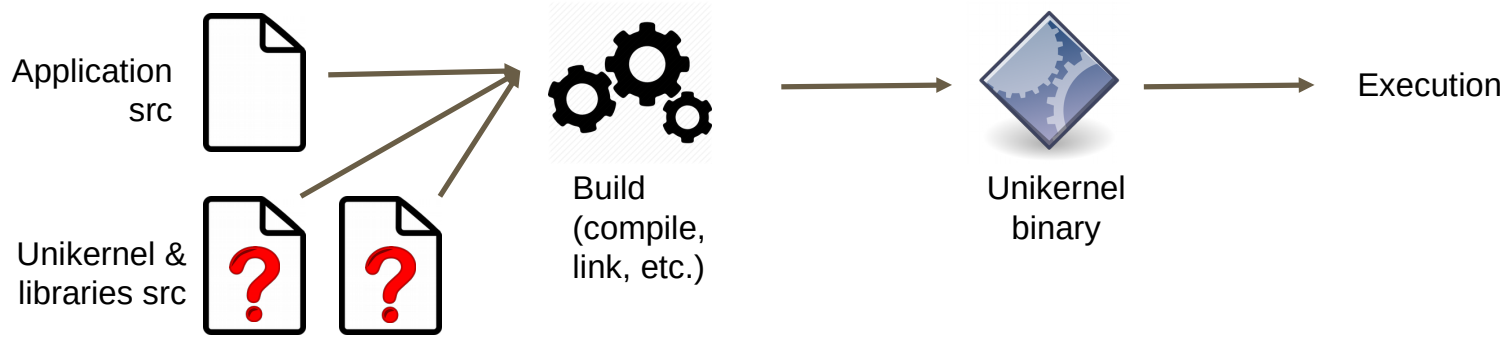
The Issue: Porting to Unikernels



- Proprietary software → source code not available
- Incompatible language

Unikernels

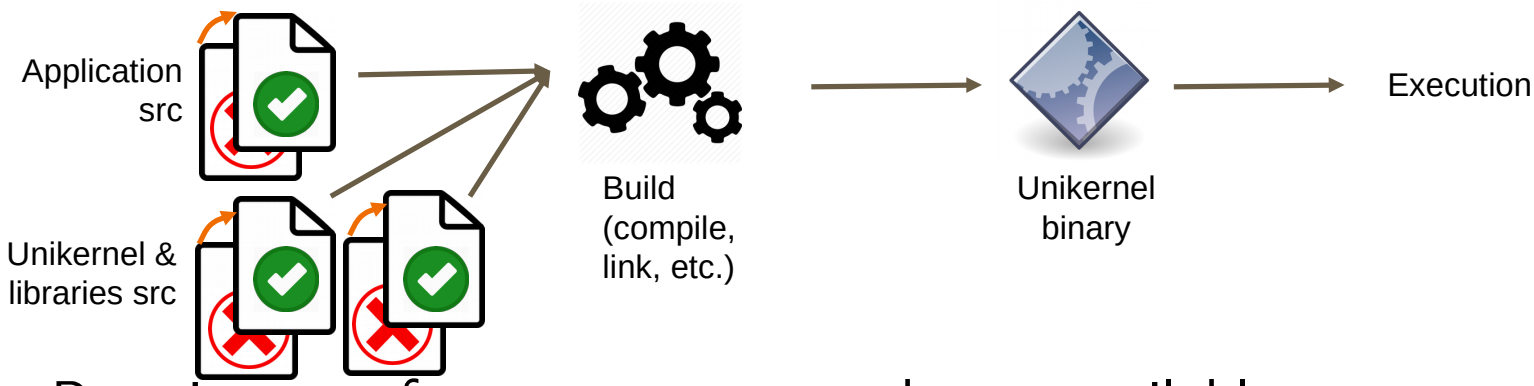
The Issue: Porting to Unikernels



- Proprietary software → source code not available
- Incompatible language
- Unsupported features

Unikernels

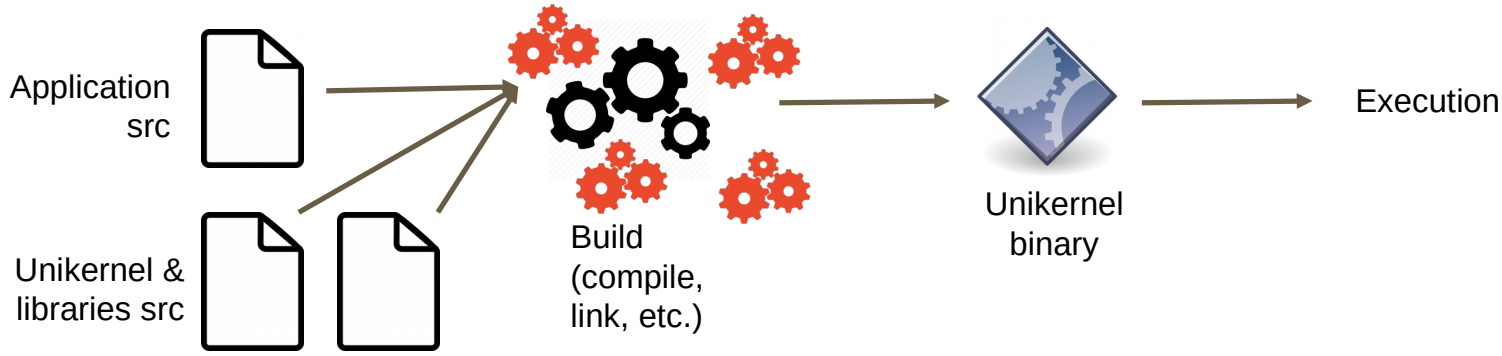
The Issue: Porting to Unikernels



- Proprietary software → source code not available
- Incompatible language
- Unsupported features
- Porting is hard, needs knowledge about both application and unikernel

Unikernels

The Issue: Porting to Unikernels



- Proprietary software → source code not available
- Incompatible language
- Unsupported features
- Porting is hard, needs knowledge about both application and unikernel
- Complex build toolchains

Unikernels
The Issue: Porting to Unikernels

- Proprietary software → source code not available
- Incompatible language
- Unsupported features
- Porting is hard, needs knowledge about both application and unikernel
- Complex build toolchains

HermiTux Solution

- **A unikernel binary-compatible with Linux**
  - For x86-64 for now

Unikernels
Overview

- Linux ABI convention:

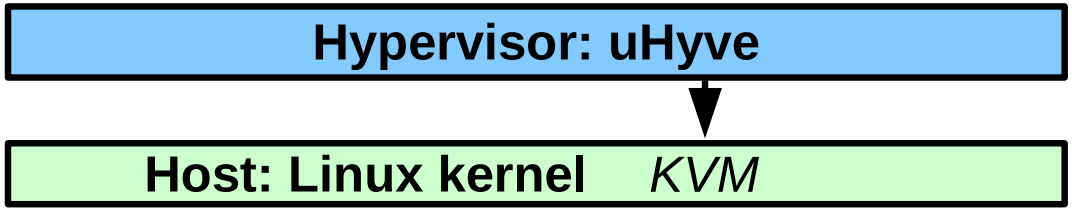
Unikernels
Overview

- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout

Unikernels
Overview

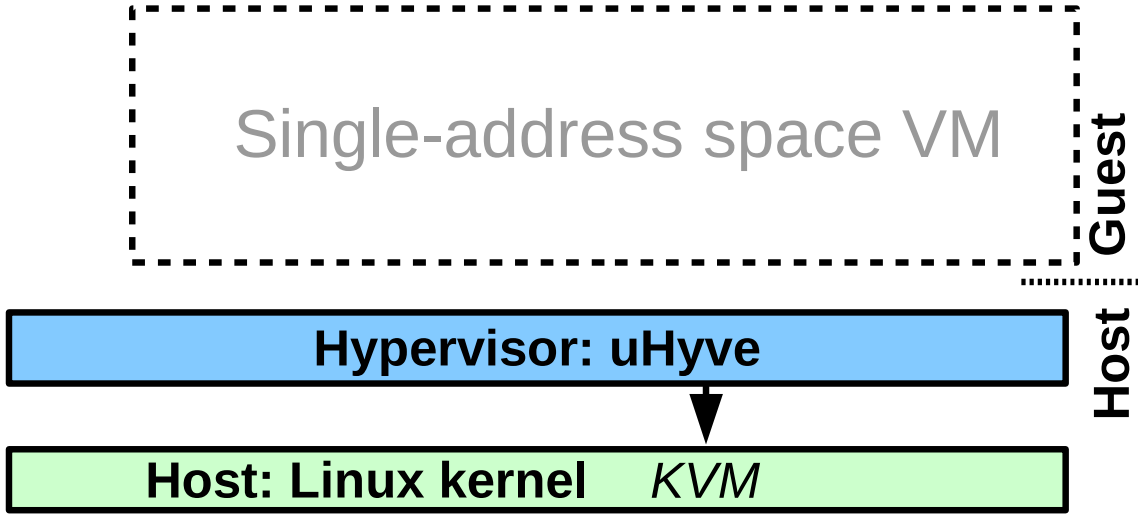
- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls

Unikernels
Overview



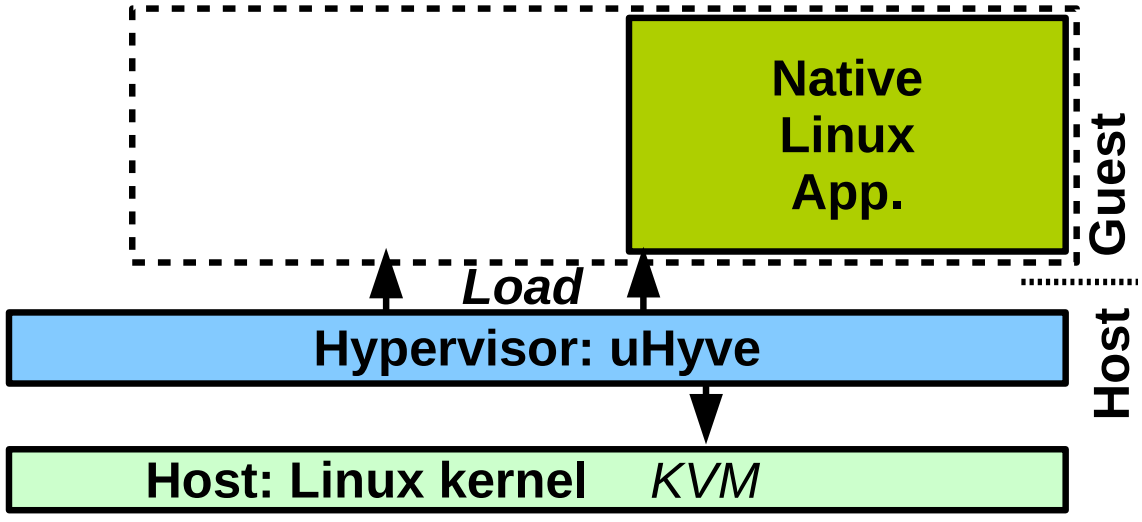
- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore

Unikernels
Overview



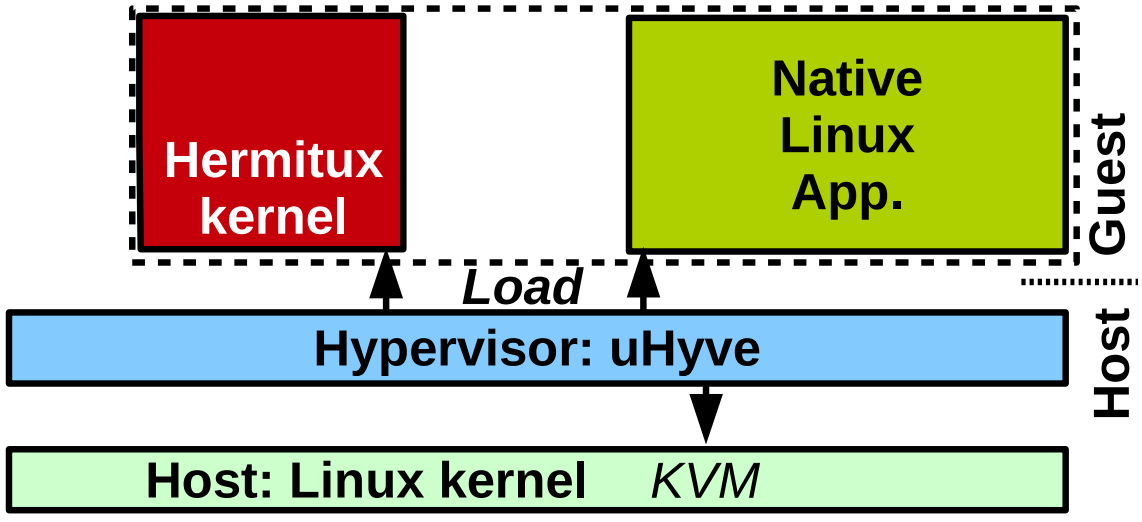
- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore

Unikernels
Overview



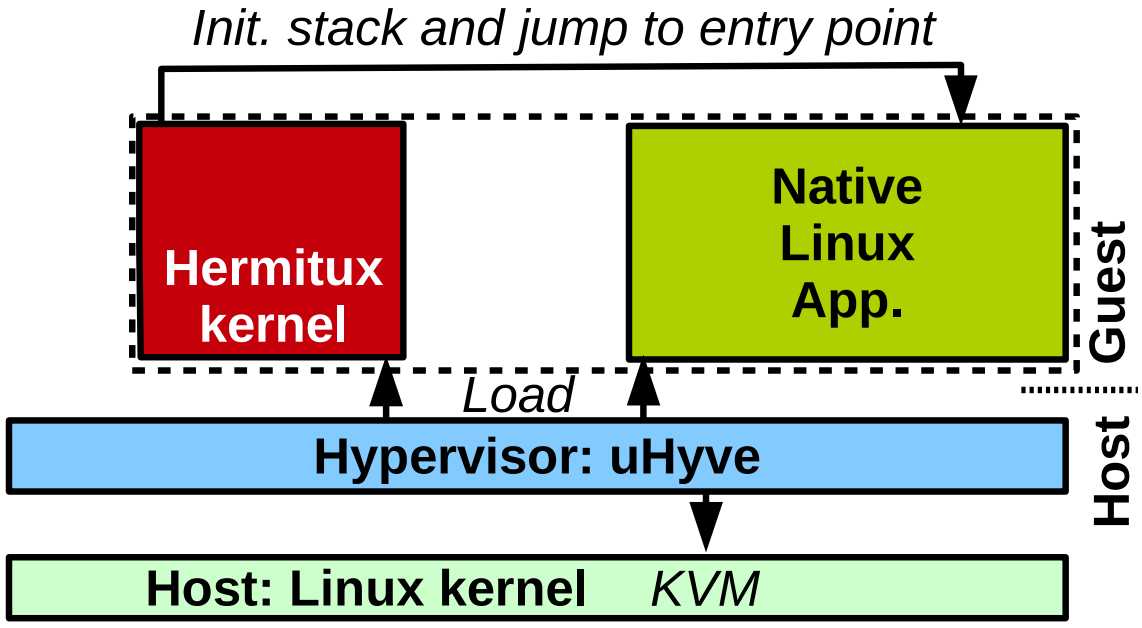
- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore

Unikernels
Overview



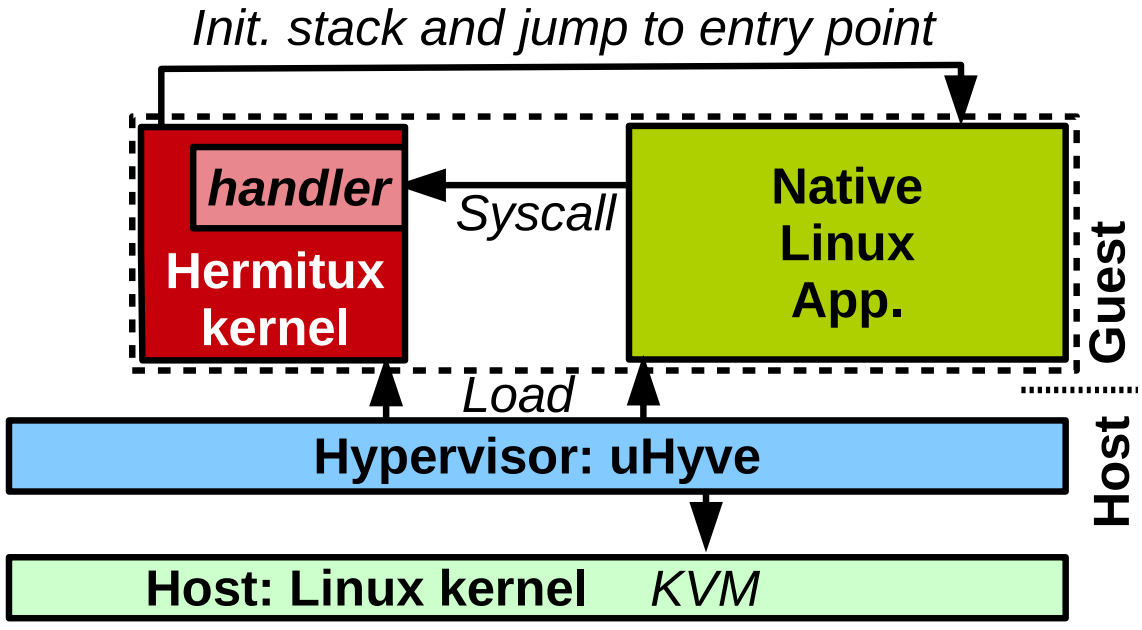
- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore

Unikernels
Overview



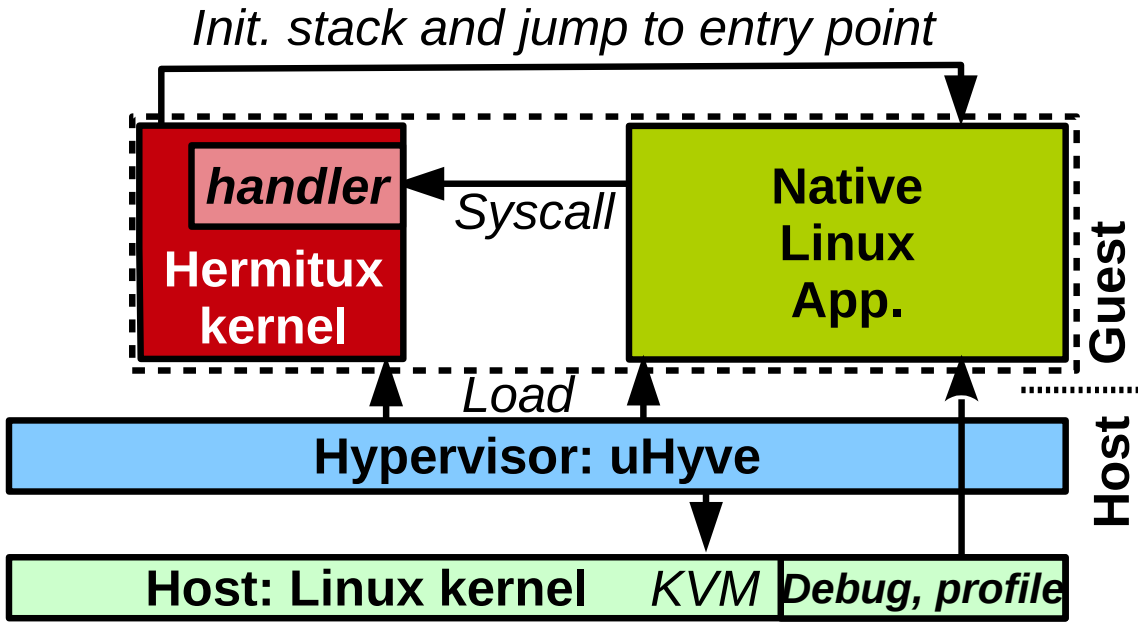
- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore

Unikernels
Overview



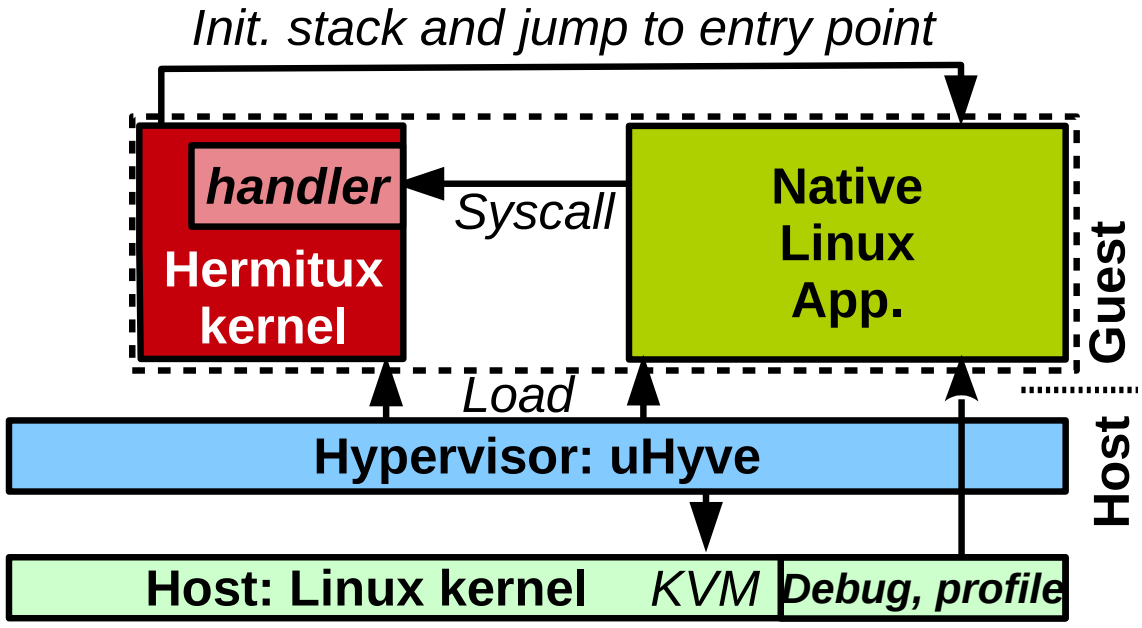
- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore
- Complete/partial support for 80+ syscalls

Unikernels
Overview



- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore
- Complete/partial support for 80+ syscalls

Unikernels
Overview



- **Linux ABI convention:**
  - ELF loader convention
  - Load-time Stack layout
  - Syscalls
- Kernel adapted from HermitCore
- Complete/partial support for 80+ syscalls

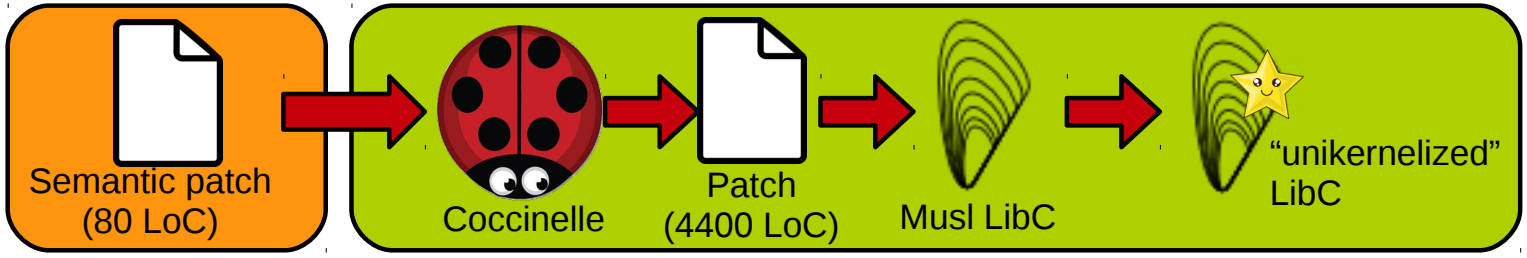
- **How to maintain unikernel benefits without access to the application sources?**
  - *Fast system calls* and *modularity*

Unikernels
Fast Syscalls with Libc Substitution

- HermiTux's syscall handler is invoked by the `syscall` instruction
  - Reintroduce high latency for system calls due to the world switch

Unikernels
Fast Syscalls with Libc Substitution

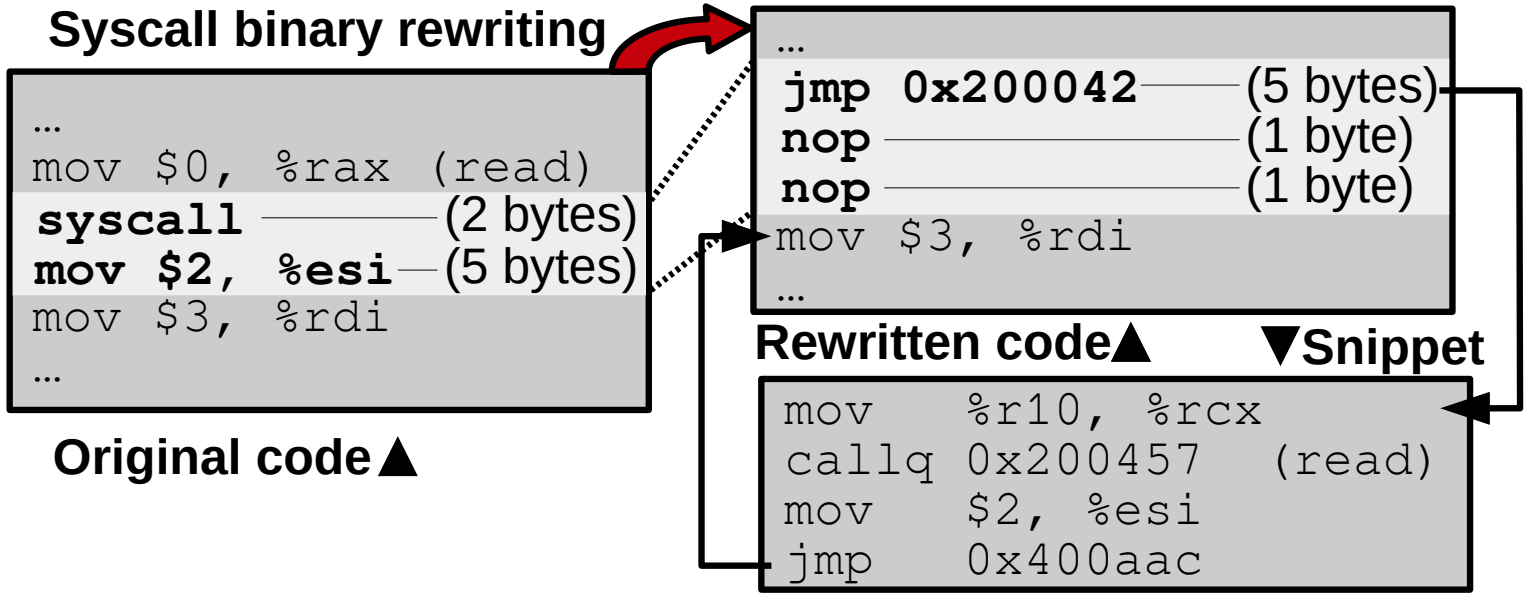
- HermiTux's syscall handler is invoked by the `syscall` instruction
  - Reintroduce high latency for system calls due to the world switch
- For **dynamically compiled programs:**
  - At runtime **load a unikernel-aware Libc**
    - Making for system calls (fast) function calls directly into the kernel
  - Automatically transformed version of Musl Libc with Coccinelle



Unikernels

Fast Syscalls with Binary Rewriting

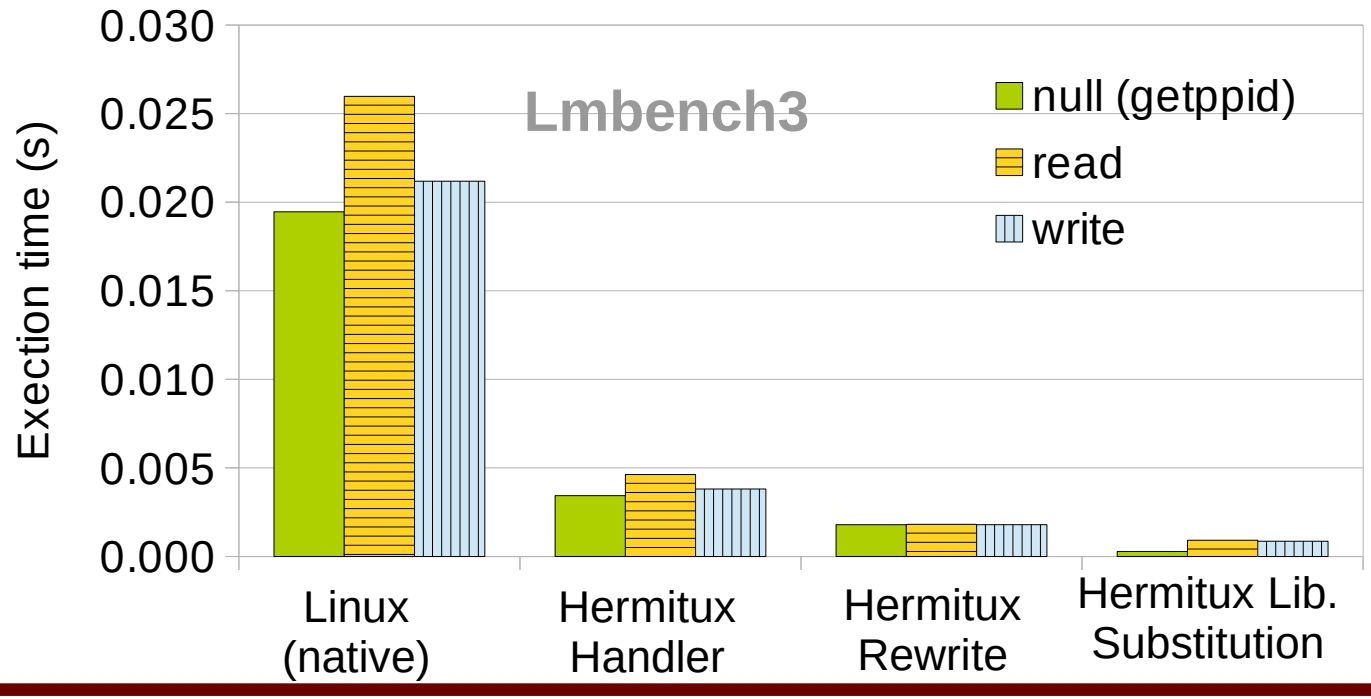
- What about **static binaries?**
- (Statically) **binary-rewrite `syscall` instructions to direct jumps to the syscall implementation**
  - Problem: `syscall` is 2 bytes long and any `call`/`jmp` instruction will be larger



Unikernels

Fast Syscalls with Binary Rewriting

- What about **static binaries?**
- (Statically) **binary-rewrite `syscall` instructions to direct jumps to the syscall implementation**
  - Problem: `syscall` is 2 bytes long and any `call`/`jmp` instruction will be larger

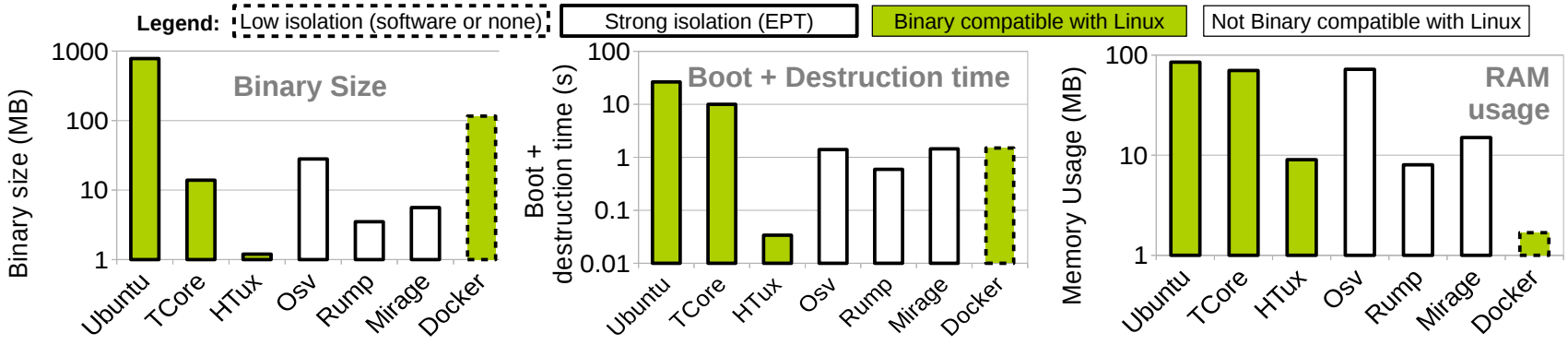


Unikernels
System-call-based Modularity

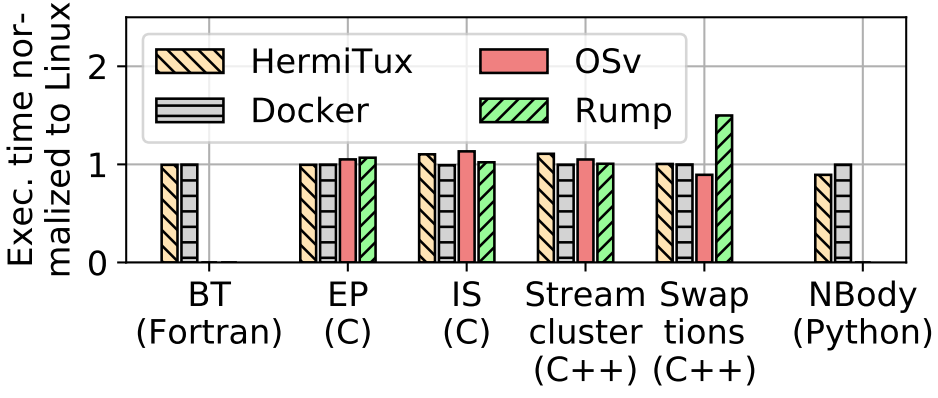
- System-call based modularity
  - Compile a kernel with support for only the necessary system calls
  - How to identify syscall needed without access to the sources?
    - **Use binary analysis** to find out what is the value in `%rax` for each `syscall` invocation

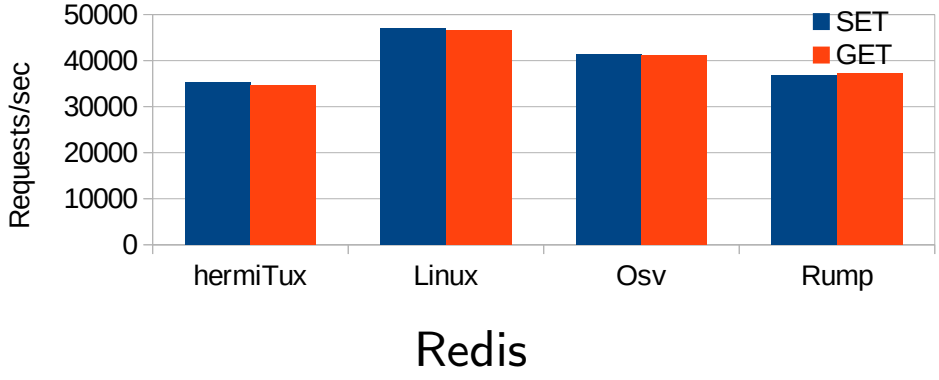
Program	Number of system calls	Kernel `.text` size reduction
Minimal	5	21.87 %
Hello world	10	19.84 %
PARSEC Blackscholes	15	17.05 %
Postmark	26	14.36 %
Sqlite	31	11.34 %
Full syscalls support	64	00.00 %

Unikernels
Evaluation



- Image 650x smaller, boot time 780x faster, RAM usage 9x lower than a Linux VM!





Conclusion

- Porting to unikernels is hard
  - Hinders their adoption in the industry
- **HermiTux provides binary-compatibility with Linux applications**
- HermiTux **maintains unikernel benefits:**
  - Fast boot times, small footprints
  - Various techniques to get fast system calls and modularity

It's open source, try it out!

`https://ssrg-vt.github.io/hermitux/`