ECE 5984 Virtualization Technologies

Project Presentation

Pierre Olivier

Outline

1) Brief presentation of unikernels

2) HermitCore

3) Problem statement

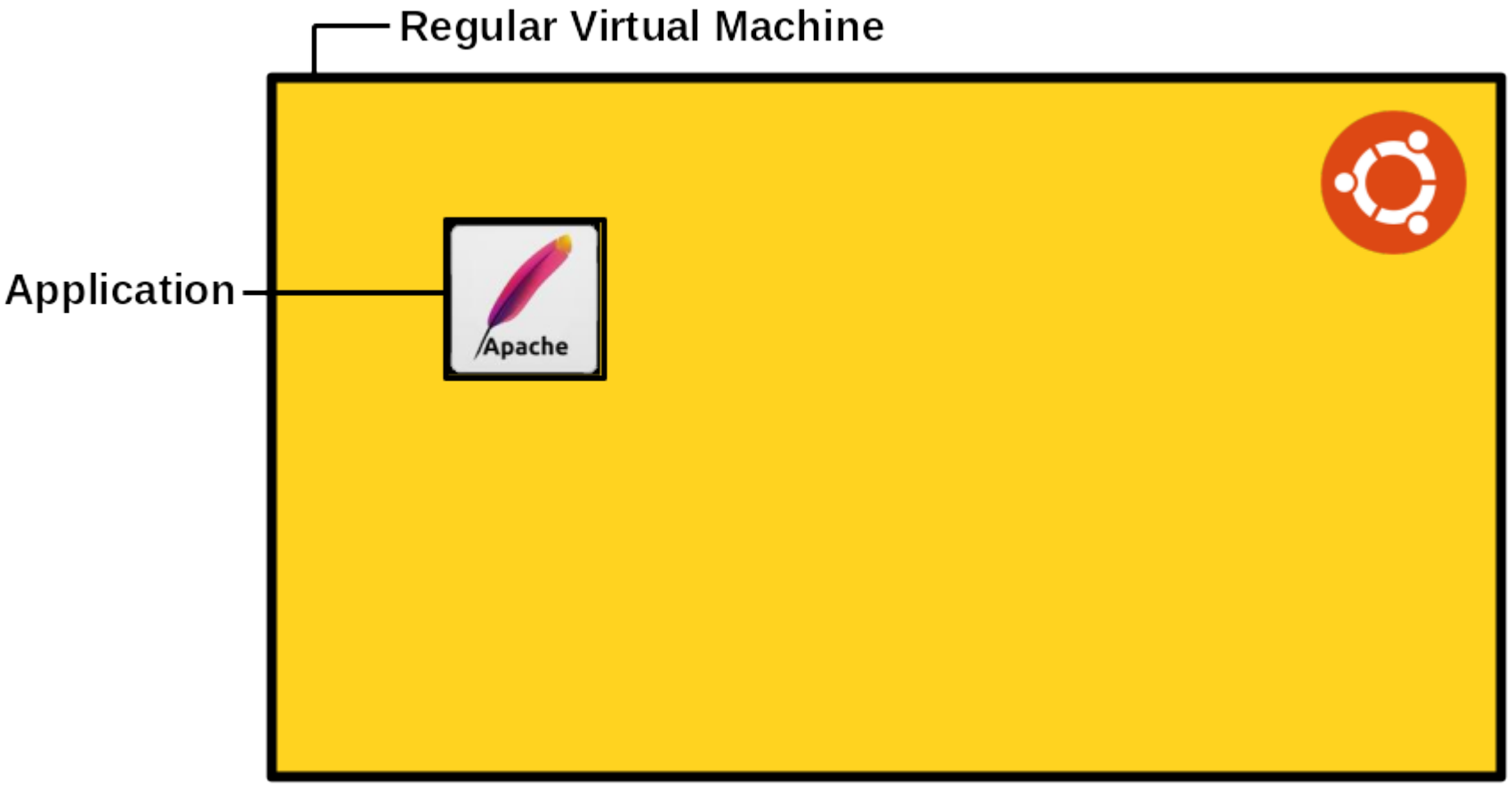
4) Project organization

Outline

1) Brief presentation of unikernels
2) HermitCore
3) Problem statement
4) Project organization

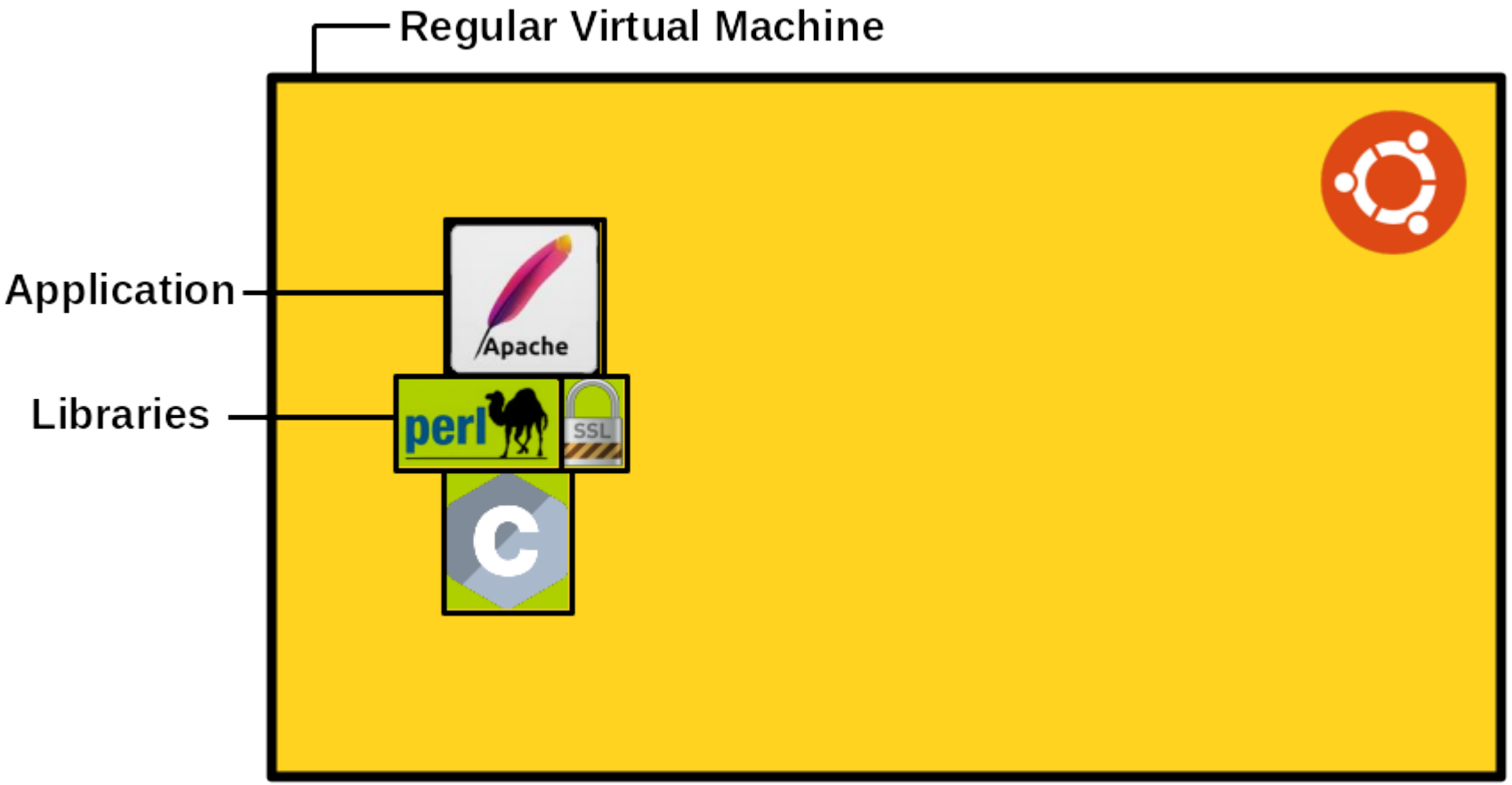
Unikernels

- Motivation: my website in the cloud



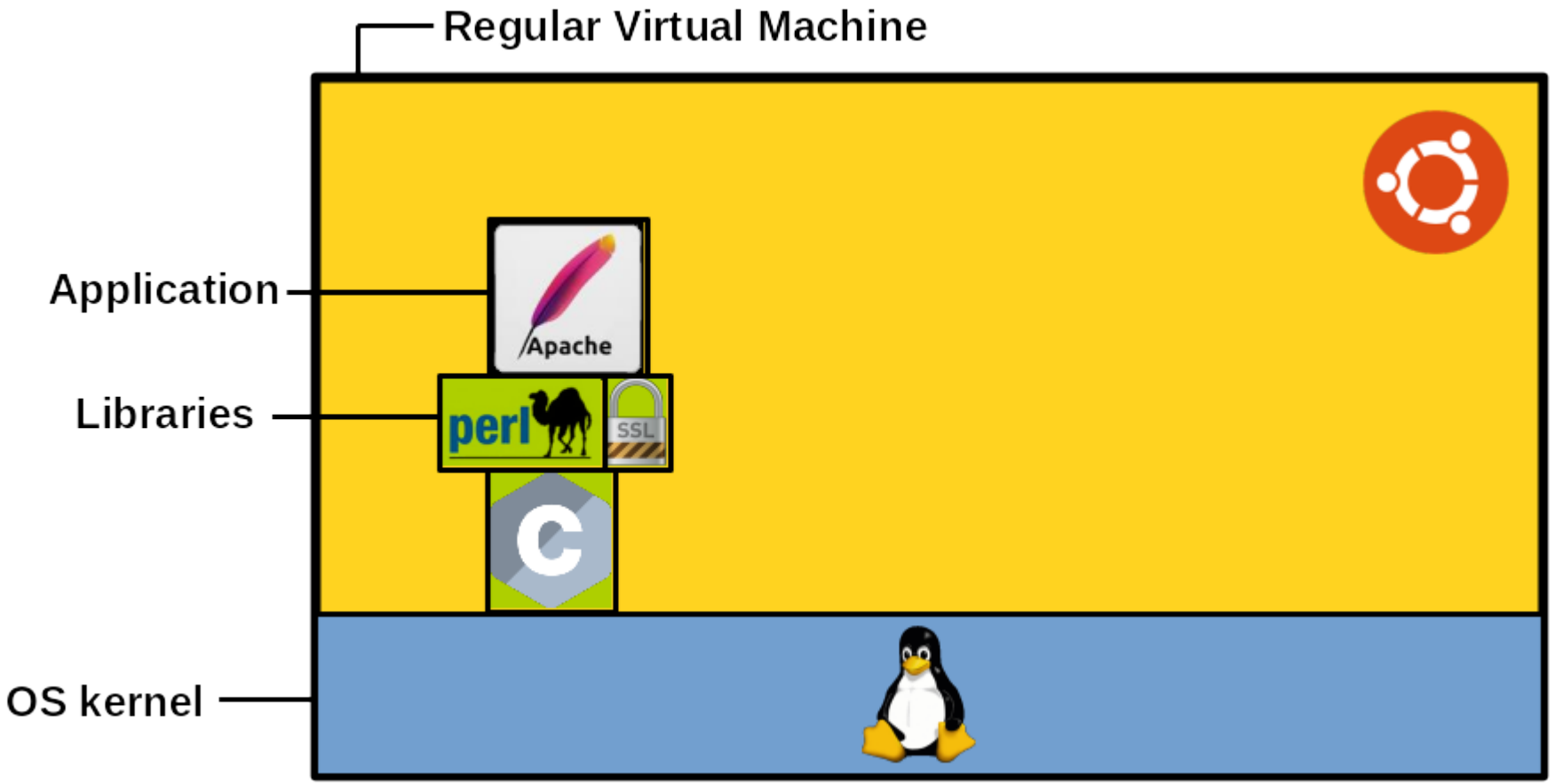
Unikernels

- Motivation: my website in the cloud



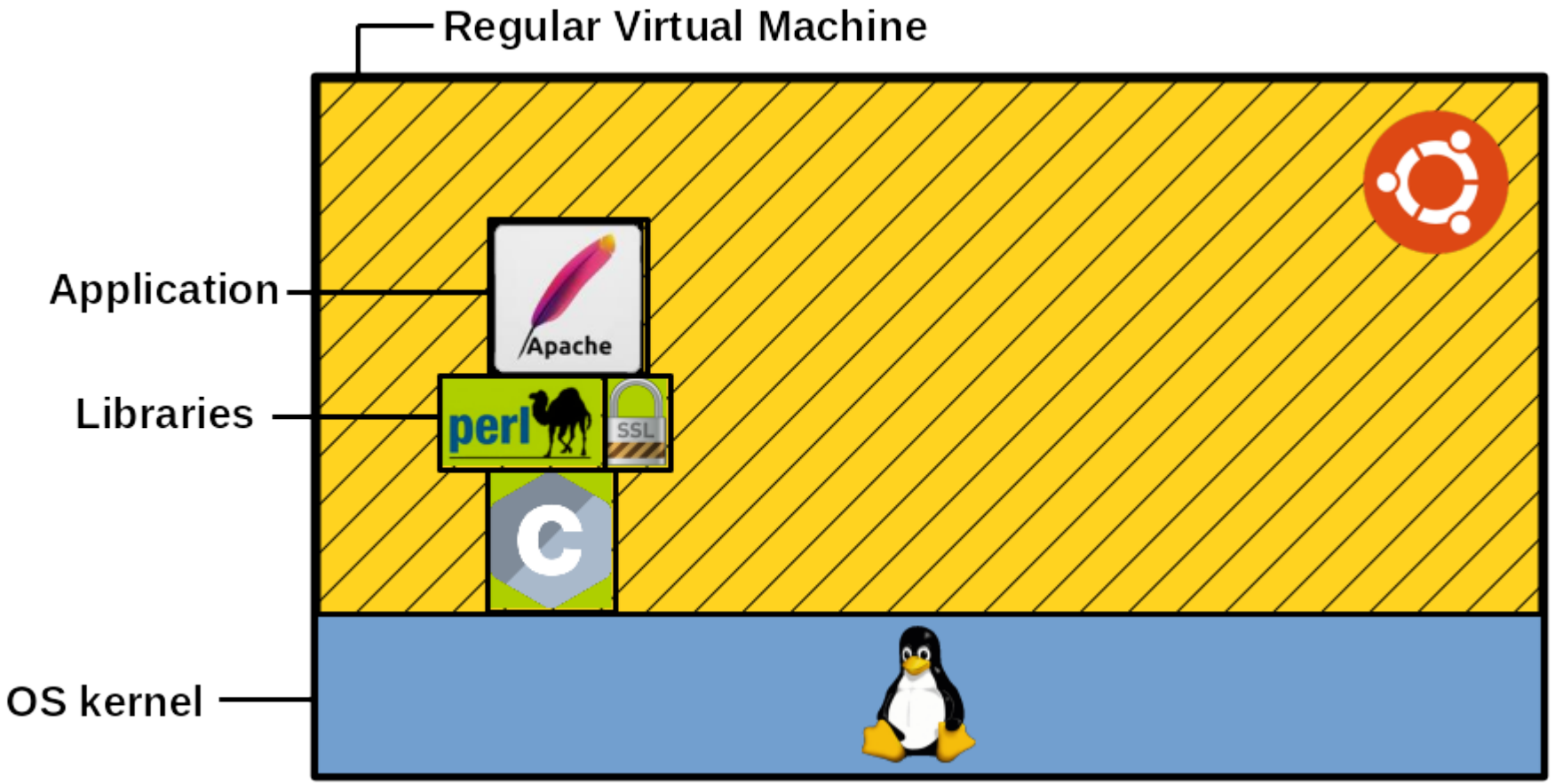
Unikernels

- Motivation: my website in the cloud



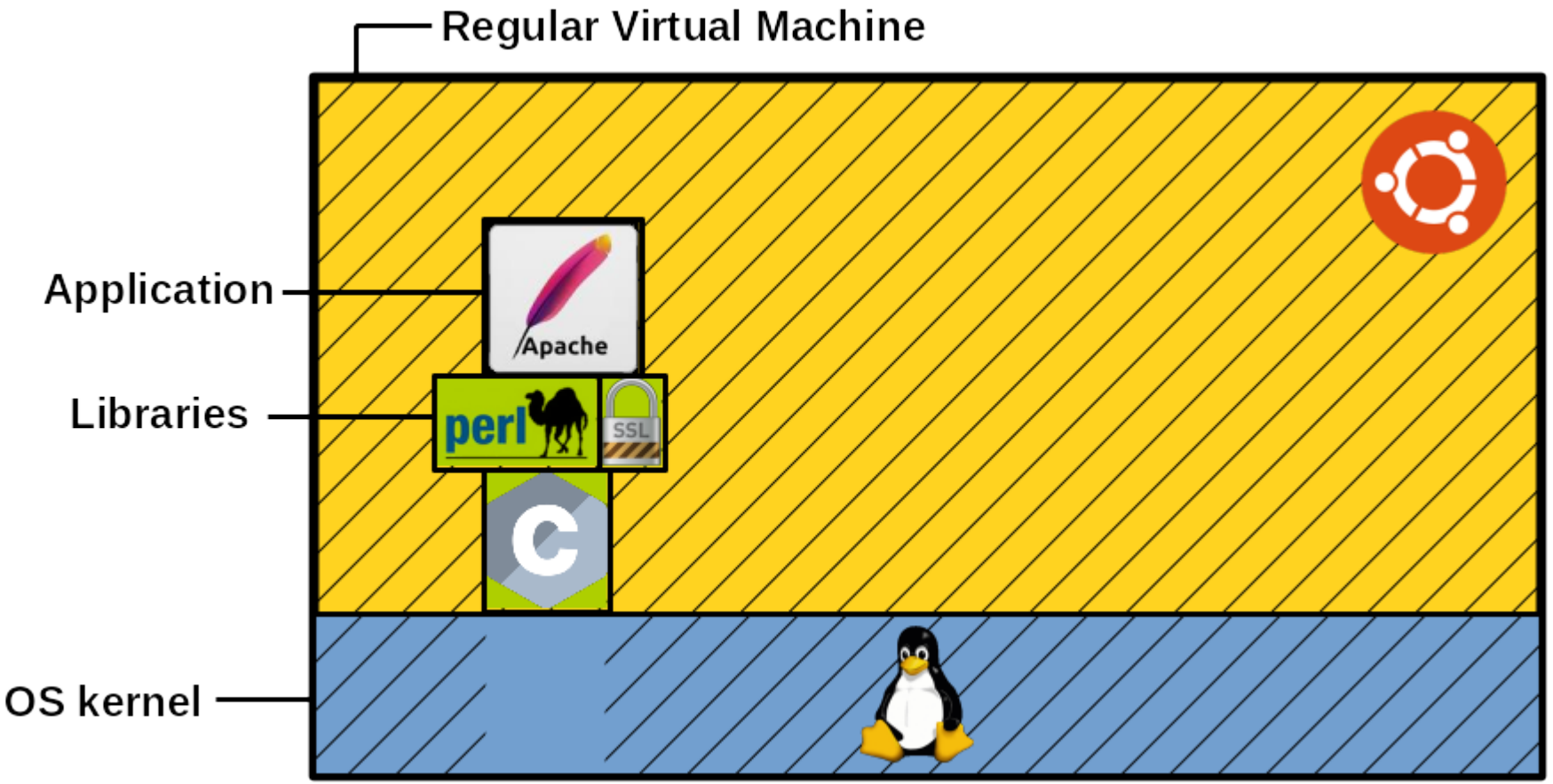
Unikernels

- Motivation: my website in the cloud



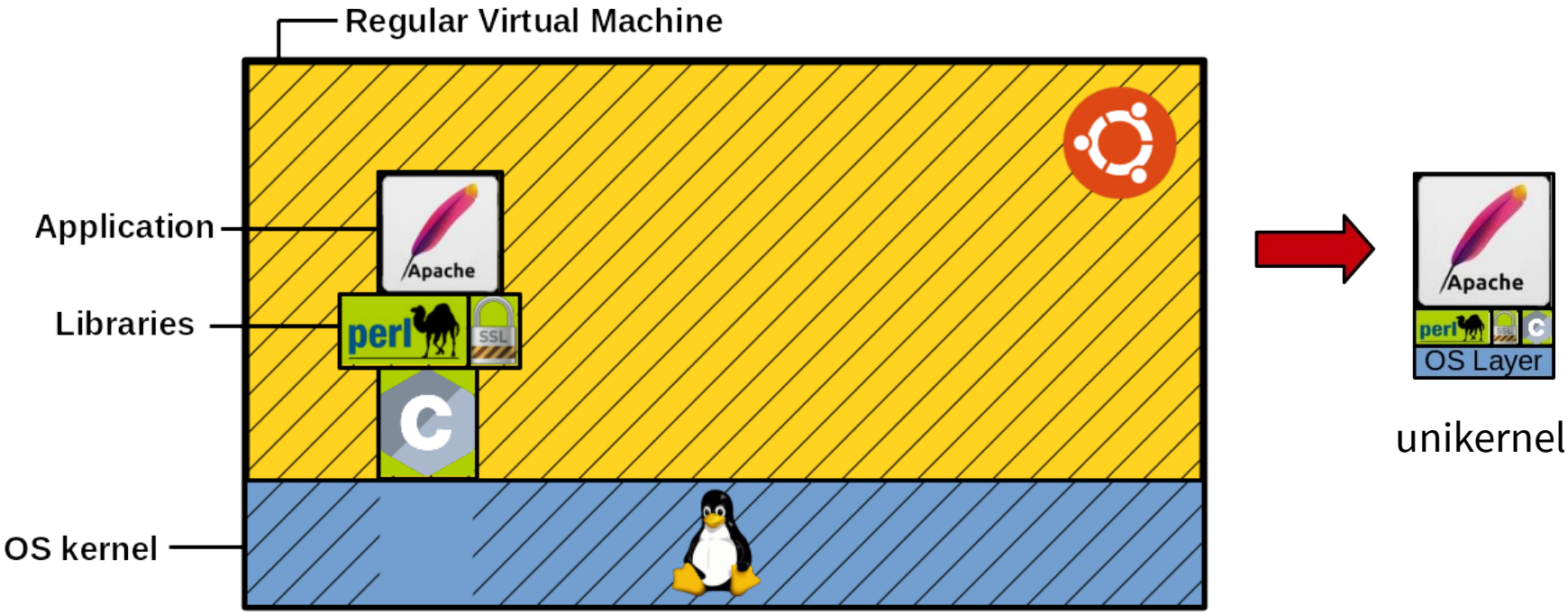
Unikernels

- Motivation: my website in the cloud



Unikernels

- Motivation: my website in the cloud



Unikernels

- **Unikernel:**
  - Statically compiled **application + libraries + thin OS layer** running as a guest on top of an hypervisor
  - Single purpose: 1 application
  - Single binary
  - Single address space
    - Shared by the application and the kernel, no privilege separation, system calls are function calls
  - More info:
    - http://unikernel.org/
    - http://unikernel.org/files/2014-cacm-unikernels.pdf
    - https://www.youtube.com/watch?v=24rvIB4_v4U&t=607s
- **Interesting benefits in terms of security, cost reduction, and performance**

Unikernels

- **One claim made by unikernel supporters is that systems calls are fast**
  - Common function calls
  - No ‘world switch’ between user space and kernel
- **Interesting paper studying the cost of system calls:**
  - Soares, Livio, and Michael Stumm. "**FlexSC: Flexible system call scheduling with exception-less system calls.**" Proceedings of the 9th *USENIX conference on Operating systems design and implementation* (OSDI). USENIX Association, 2010.

Outline

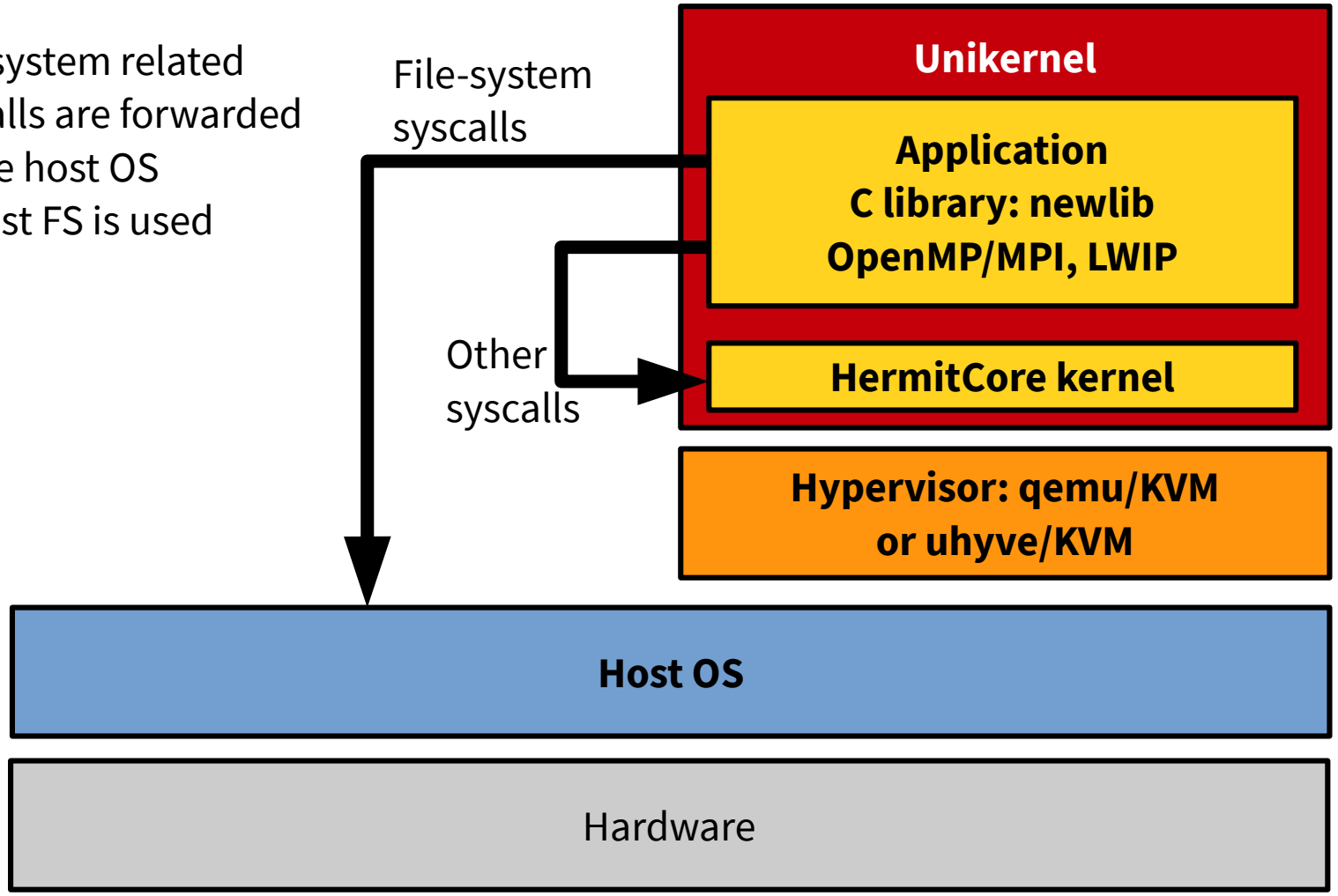
1) Brief presentation of unikernels
2) HermitCore
3) Problem statement
4) Project organization

HermitCore

- **Unikernel originally dedicated to HPC**
  - Described in this paper:
    - Lankes, Stefan, Simon Pickartz, and Jens Breitbart. "HermitCore: a unikernel for extreme scale computing." *Proceedings of the 6th International Workshop on Runtime and Operating Systems for Supercomputers*. ACM, 2016.
    - http://www.hermitcore.org
  - Written in C, very simple codebase:
    - ~10KLOC
    - Good candidate for a project
  - Supports running on top of the following hypervisors:
    - Qemu (and Qemu/KVM)
    - Uhyve
      - ➔ Custom minimal hypervisor ~2KLOC

HermitCore

- File-system related syscalls are forwarded to the host OS
  - Host FS is used



Outline

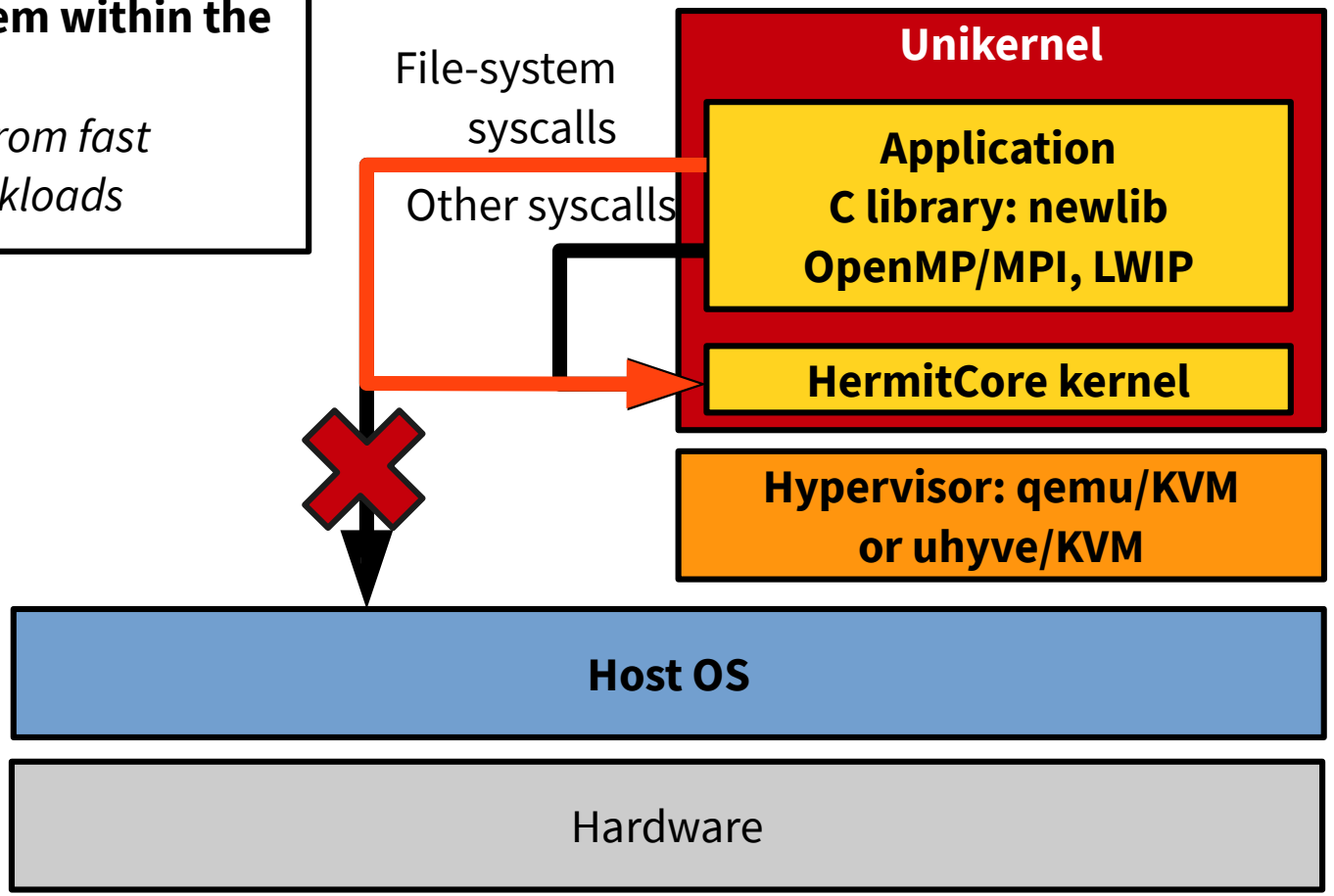
1) Brief presentation of unikernels
2) HermitCore
3) **Problem statement**
4) Project organization

Problem statement

- **Unikernels system calls are fast**
  - Common function calls, no world switch
  - Was never clearly demonstrated
- **Most syscall intensive macro-benchmarks are file-system and network intensive**
  - We will not consider network (LWIP is pretty slow by itself)
  - HermitCore file system calls are forwarded to the host:
    - The bottleneck is this forwarding process, will not see any performance improvement from running a benchmark as a unikernel!

Problem statement

Project objective:
Implement a *simple* file-system within the HermitCore unikernel
- To explore potential benefits from fast system calls in FS intensive workloads



Solution design

What does *implementing a FS within HermitCore* means?

- Implement the FS syscall interface and define how files are stored, organized and retrieved in/from the backing store
- Backing store (storage medium)?
  - Strong advice: Implement the FS as a **ramdisk**
    - ➔ No driver to write, no need to virtualize the Host HDD
- Interface with the application
  - HermitCore redirects the following syscalls to the host:
    - ➔ open, read, write, lseek, close
    - ➔ **Change that to internal processing in HermitCore**
  - **Add new system calls**
    - ➔ creat, mkdir, rename, readlink, etc.

Recommended project steps

1) Being able to run a simple program

- Write a unikernel that opens a single file with O_CREAT, write something, read it, then close

2) Add directories

- Write a unikernel creating a directory with a few files in it then list the content of the directory

3) Scale up

- Create and access thousands of files and directories

4) Macro-benchmarks & performance evaluation

- Port postmark to HermitCore
- Compare your file-system performance (postmark & micro-benchmarks) to
  1) Regular HermitCore and/or
  2) Native Linux

5) Additional functionalities

- Pre-charge the ramdisk before application execution
- Other ideas?

Outline

1) Brief presentation of unikernels
2) HermitCore
3) Problem statement
4) Project organization

Project organization

- **Group work: can be accomplished in groups of 2 students max**
- **Timeline:**
  - Project assignment PDF released tomorrow
  - 'Getting started with HermitCore' technical guide released today
  - For next monday
    - Reproduce all the items in the technical guide on your own
    - Come up with a FS design
      - ➔ What are the system calls you will support, and a draft of what happen under the hood when each is executed
    - We'll review the designs together
- **Results to hand by *2018-02-20 11:59PM*:**
  - Source code, tests & performance evaluation programs
  - Project report: 7 pages minimum
  - 1 archive per group (i.e. 1 report & set of source files)