November 22, 2019

UNICORE

UNICORE: A toolkit to automatically build unikernels

Grascomp Doctoral Day 2019

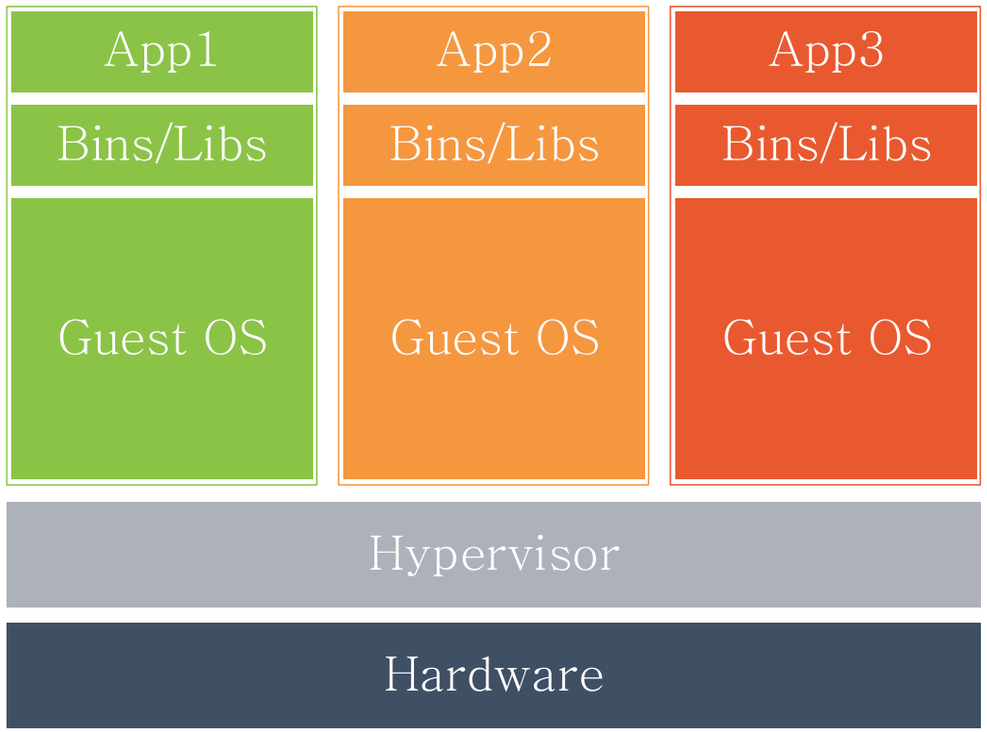
Gaulthier Gain Cyril Soldani *Prof.* Laurent Mathy

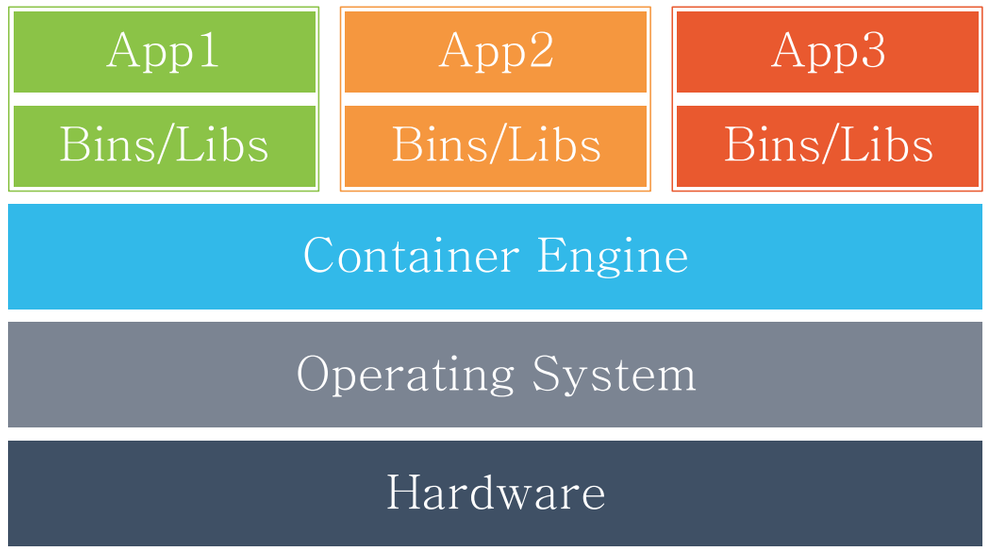
[first].[last]@uliege.be

LIÈGE université

Funded by the Horizon 2020 Framework Programme of the European Union under agreement No 825377

Virtual Machines vs Containers

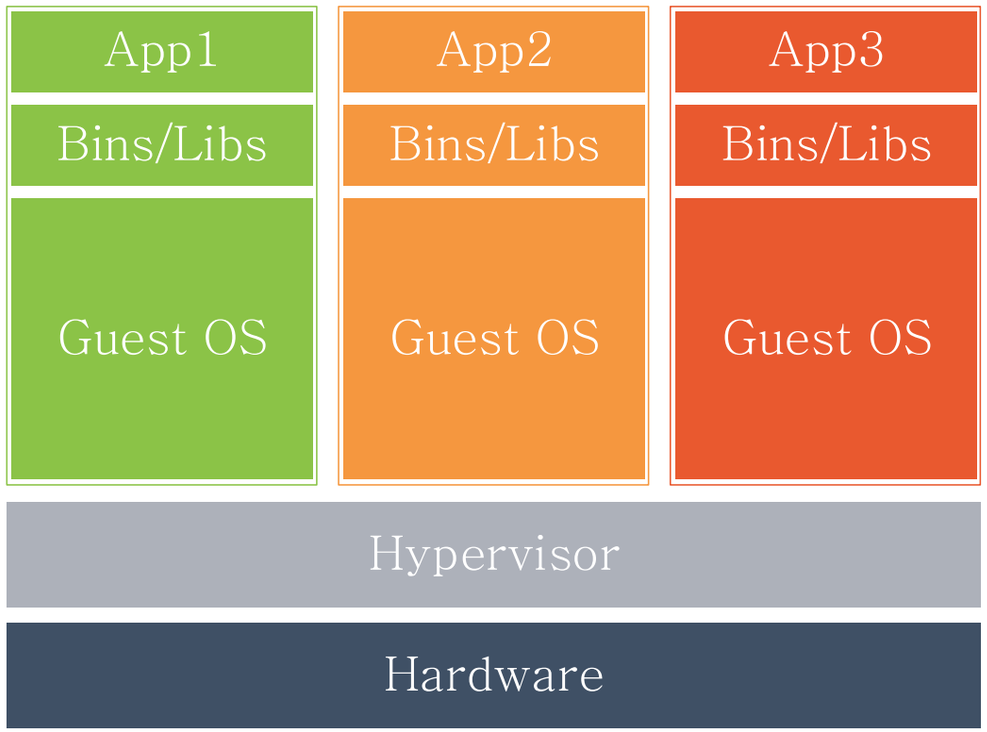


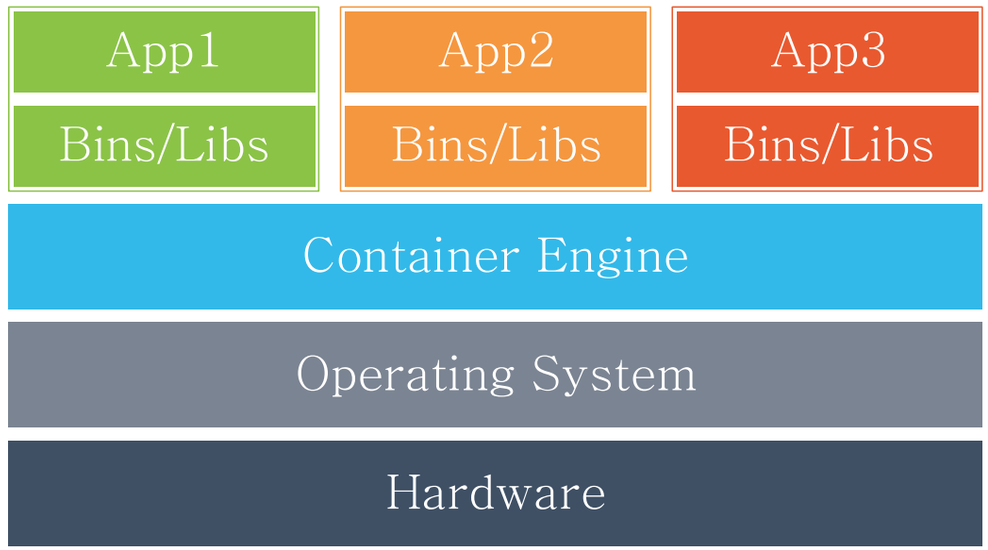


Virtual Machines (VMs) Containers

- Each virtual machine requires its own underlying guest operating system as well as a hypervisor.
- It provides strong isolation, virtual machines are heavyweight since they require a full OS image to run.

Virtual Machines vs Containers





Virtual Machines (VMs) **Containers**

- Each container shares the OS kernel which provides better efficiency than virtual machines.
- It results into a poor isolation and containers are subject to many vulnerabilities (bigger attack surface).

Dilemma

Virtual Machines (VMs)

✓ Strong isolation

✗ Heavyweight
– Degrade performance

Containers

✓ Lightweight

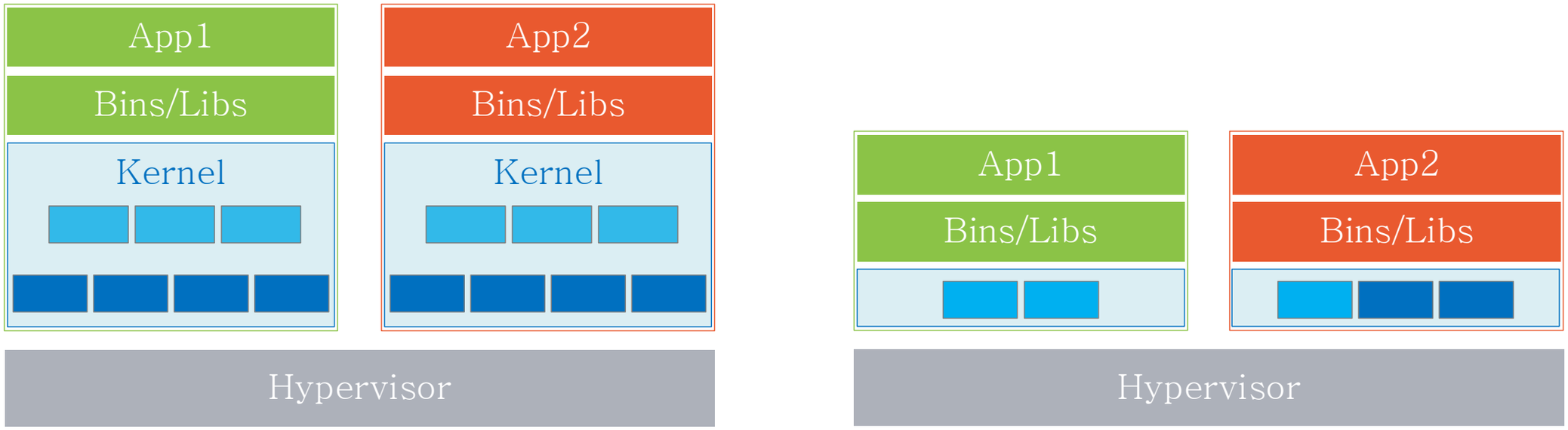
✗ Poor isolation
– A lot of exploits

Dilemma

Virtual Machines (VMs)	**Containers**
✓ Strong isolation	✓ Lightweight
✗ Heavyweight	✗ Poor isolation
– Degrade performance	– A lot of exploits

Solution $\longrightarrow$ **Unikernels**

Unikernels



Virtual Machines (VMs) Unikernels

- Unikernels are purpose-built:
  - Thin kernel layer (only the features that the application needs).
  - Essential functions $\longrightarrow$ dead-code elimination.
  - One application $\longrightarrow$ single address space (isolation).

Unikernels gains

- Fast instantiation, destruction and migration time:
  - Hundred of milliseconds.
- Small per-instance memory footprint:
  - Few MBs or even KBs.
- High density:
  - 10k instances on a single host.
- High performance:
  - 10-40 Gbps throughput.
- Reduced attack surface.

$\rightarrow$ Well suited for cloud computing and IoT.

Challenges

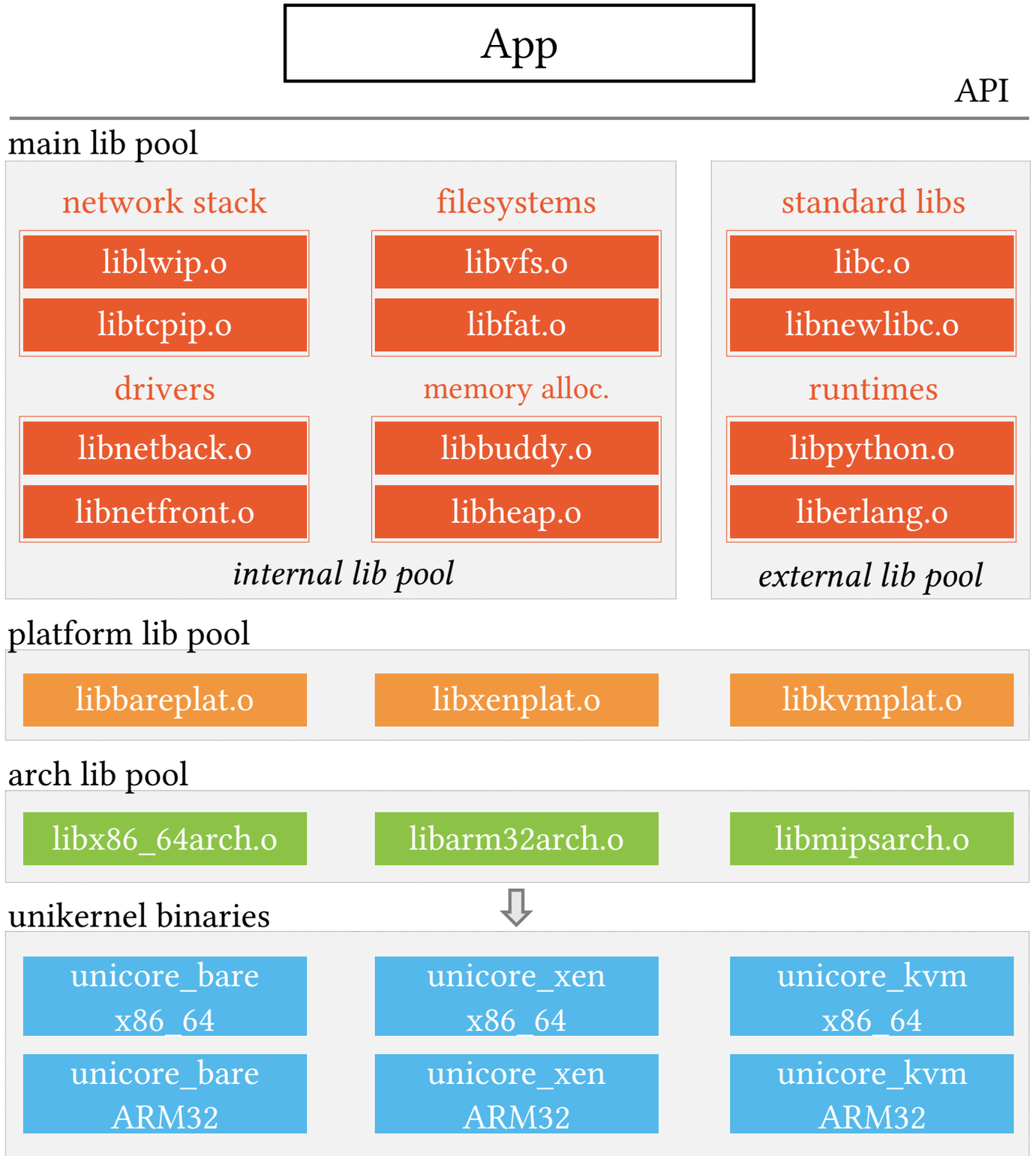
- Manual process requiring **significant expert resources** and takes a **lot of development time**:
  - Dependency analysis: gathers the right symbols, system calls and shared libraries of an **existing** application.
  - Rewrite and migrate libraries and kernel primitives.
- Multiple cycles consisting of measurement, programming improvements and tweaking.
- The build process must be repeated for **each** target **platform**, **architecture** and **application**.

The UNICORE project

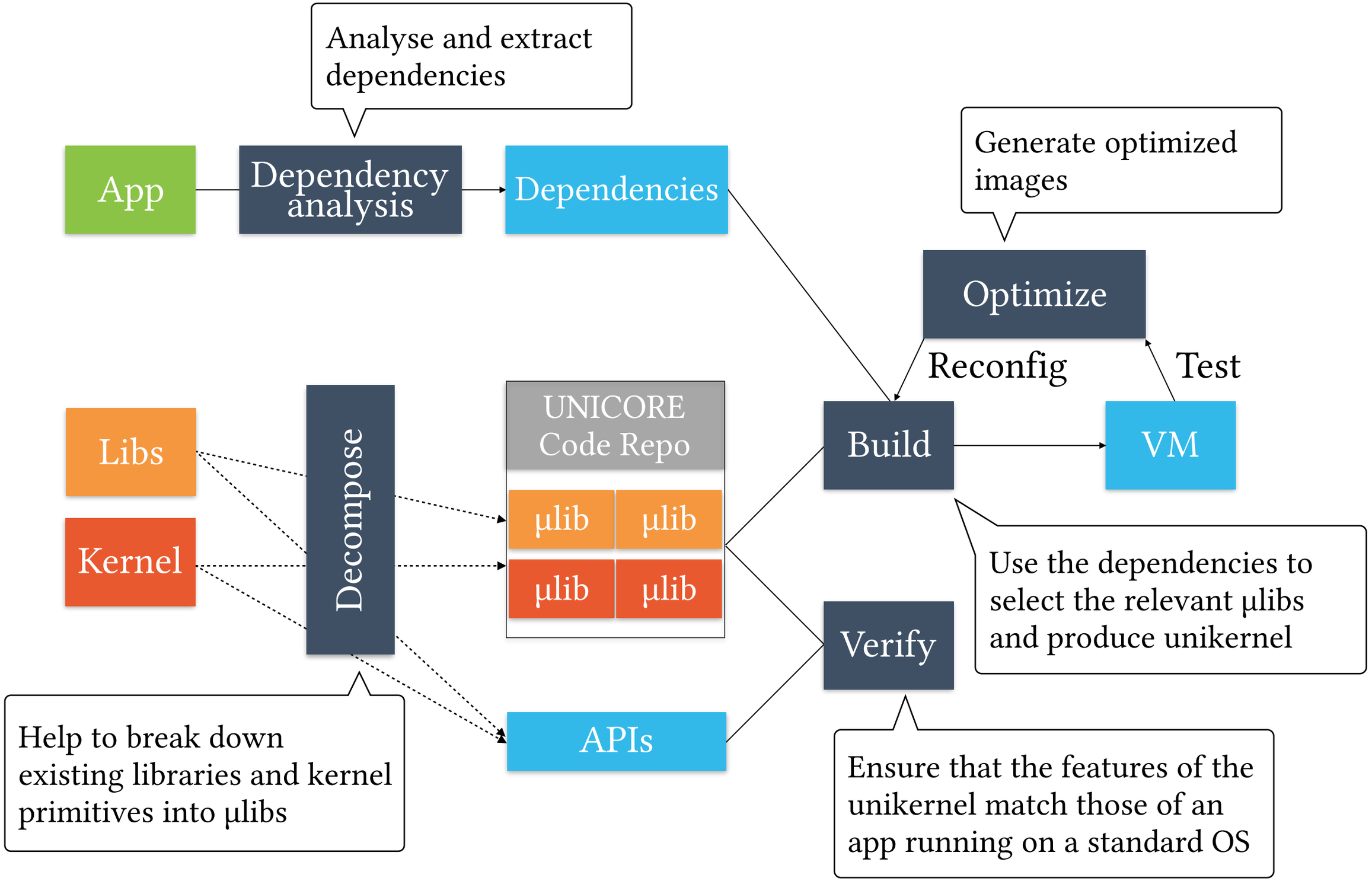
- Motivations:
  - Simplify creation, deployment and management of unikernels.
  - Concentrate efforts on a **single base unikernel** (follow-up of the Unikraft project).
- Open-source toolkit that will enable secure and portable unikernels deployment.
- Concept:
  - Everything is a **library**.
  - Monolithic software code is broken down into smaller components (**µlibs**).

Two main components: Library pools

1. Library pools:
   - Internal/External µlibs.
   - Platform µlibs.
   - Architecture µlibs.
2. Toolchain:
   - Set of tools to automatically build unikernels.
   - Generate binaries for multiple platforms and architectures.



Two main components: Toolchain



Dependency Analysis Tool

- The goal find a **sufficient**, but **minimal** superset of other components that are required by an existing application to work correctly.
- To gather information of an existing application, we considered **2** mechanisms:

1. **Static analysis**: examining binary file without execution.
   - Limited since applications can be stripped or obfuscated.
2. **Dynamic analysis**: analysing binary file by running it.
   - Find all the execution path of application: unfeasible.
   - Heuristic: high application code coverage (Tests with expected inputs, fuzz-testing, ...)

Automatic Build Tool

- The goal of the automatic build tool is to produce unikernel(s) that can run the target application(s).
- The tool is divided into **2** components:
  - A **controller** that uses the previous inputs to select the right µlibs (from library pools).
  - A **build system** to compile and link the unikernel into target VM images.
- For now, sources of an existing application are required.
- **Future work**: Consider binary rewriting techniques.
  - Port an existing binary application as unikernel.

Porting an existing application (SQLite)

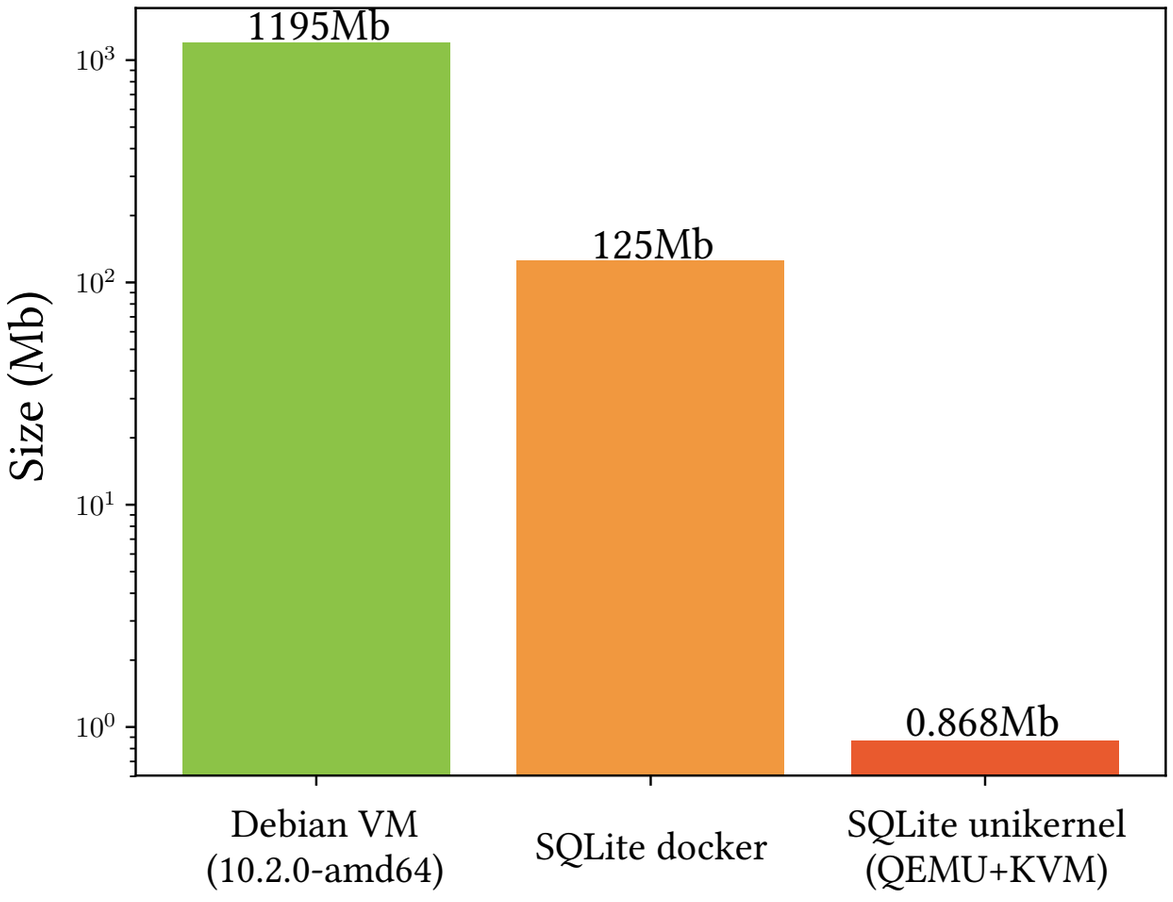


Fig1: Size (in megabytes) of the SQLite application on a Debian VM, a Docker container and a Unikernel.

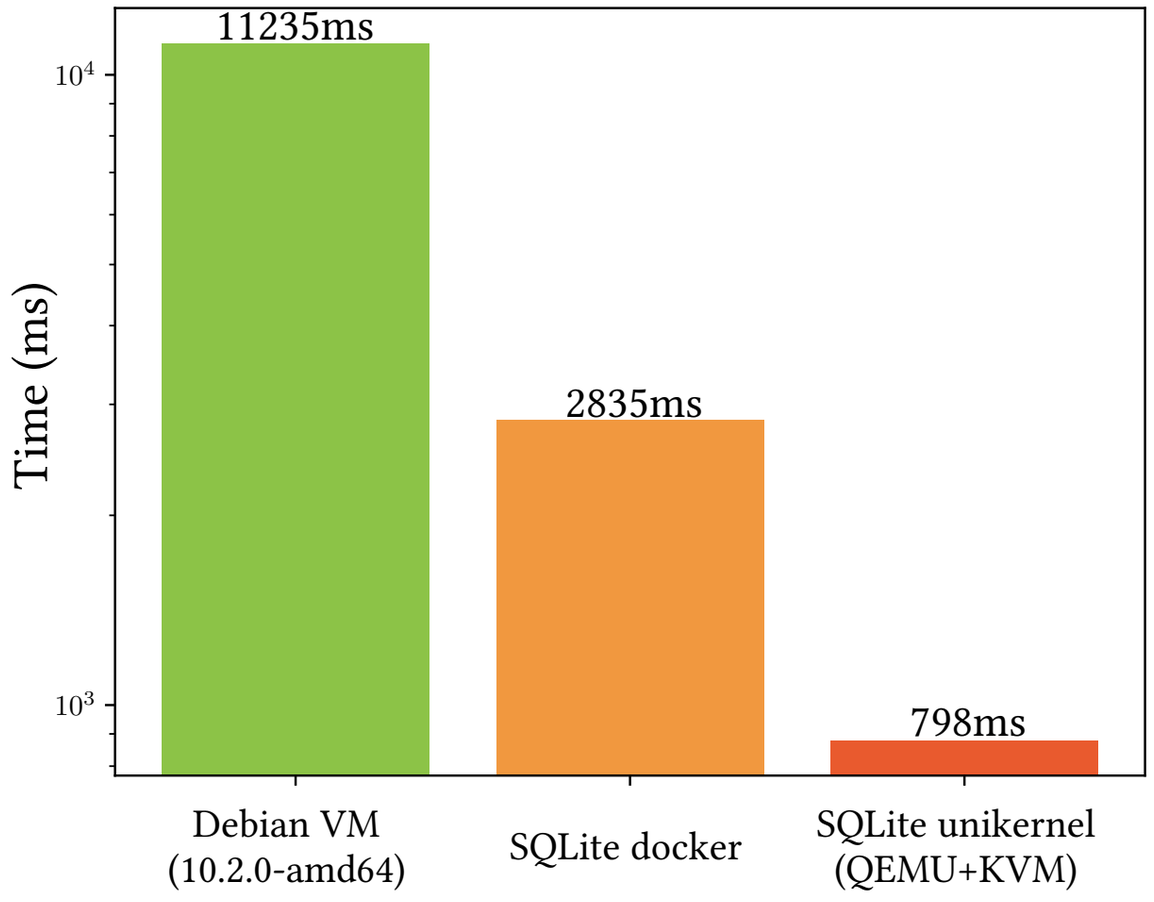


Fig2: Total life cycle* (in milliseconds) of the SQLite application on a Debian VM, a Docker container and a Unikernel.

*creation, boot, shutdown, and destroy times

Conclusion

- Unikernels:
  - Replace the heavyweight virtual machines and insecure containers.
  - Require expert resources and time-consuming to develop.
  - Not ready to be used $\longrightarrow$ Need an adequate tool.
- UNICORE:
  - Based on Unikraft.
  - Provides a toolkit to develop and deploy unikernels.
  - Solution to quickly bring unikernels in the software industry.
  - Still at early stages.

References

- F. Manco, C. Lupu, F. Schmidt, Jose M., Simon K., Sumit S., Kenichi Y., Costin R., and Felipe H.. 2017. My VM is Lighter (and Safer) Than Your Container. In Proc. of SOSP 2017.
- Unikernels.org. [n. d.]. Unikernels: Rethinking Cloud Infrastructure. http://unikernel.org/.
- Xen Project. [n. d.]. Unikraft Development Team. https://xenproject.org/developers/teams/unikraft/.
- UNICORE. [n. d.]. Quickly developing applications. http://unicore-project.eu.

THANK FOR YOU ATTENTION

QUESTIONS?

Funded by the Horizon 2020 Framework Programme of the European Union under agreement No 825377