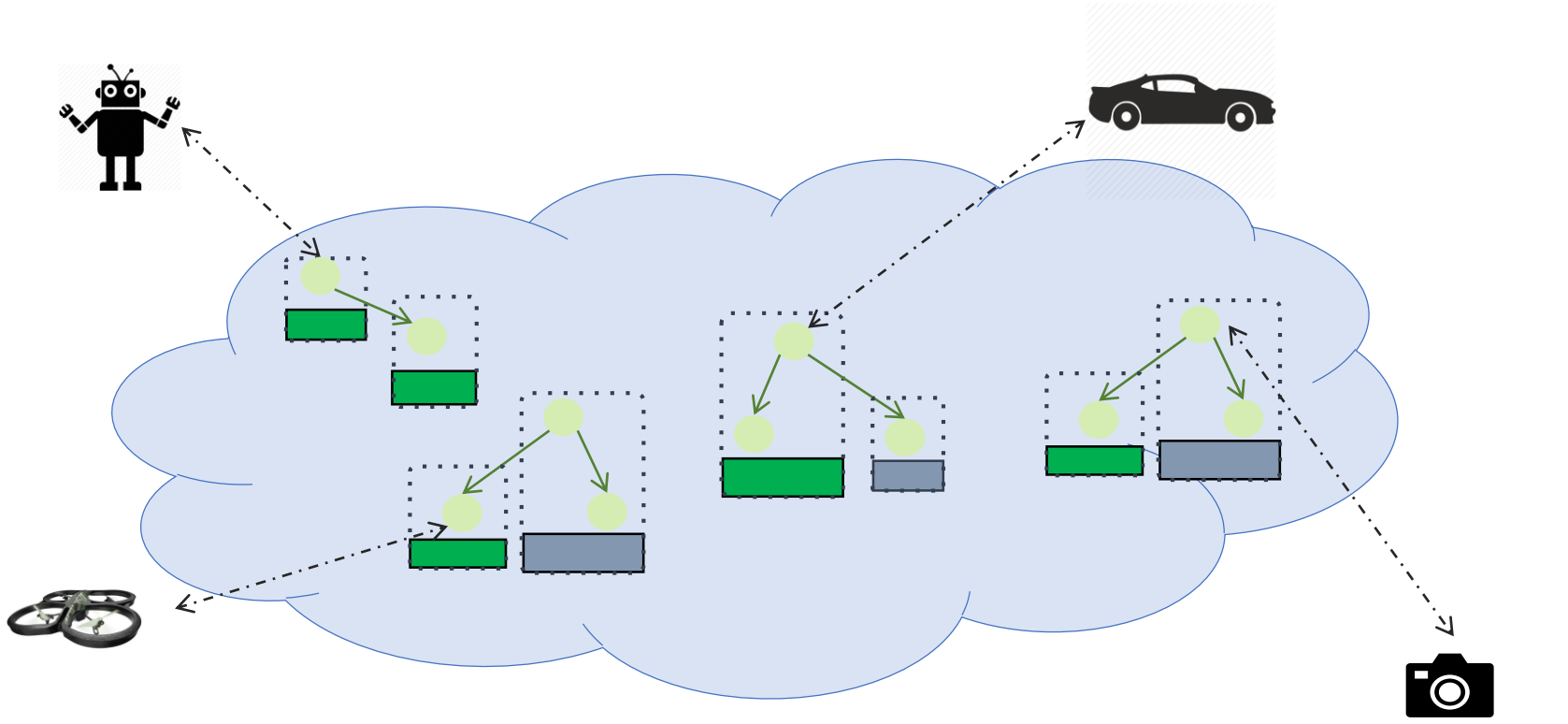
Enabling seL4 Containers to Support Legacy Applications

Hui Lu
Assistant Professor, SUNY Binghamton

A Secure Embedded Cloud Infrastructure?



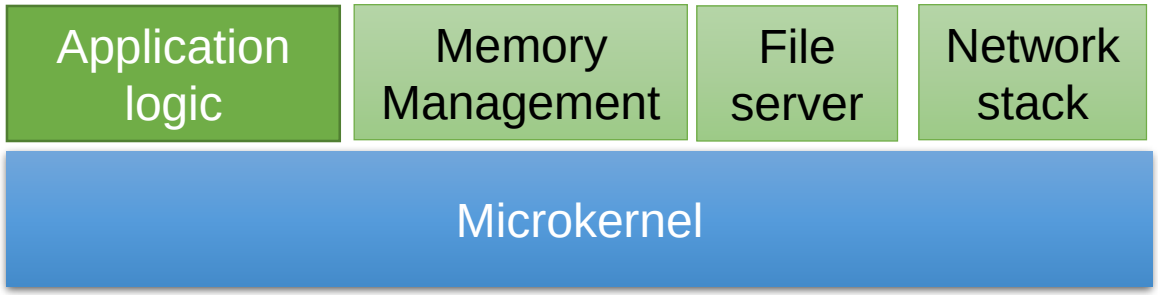
seL4 nodes

Traditional nodes

Services of a cloud application

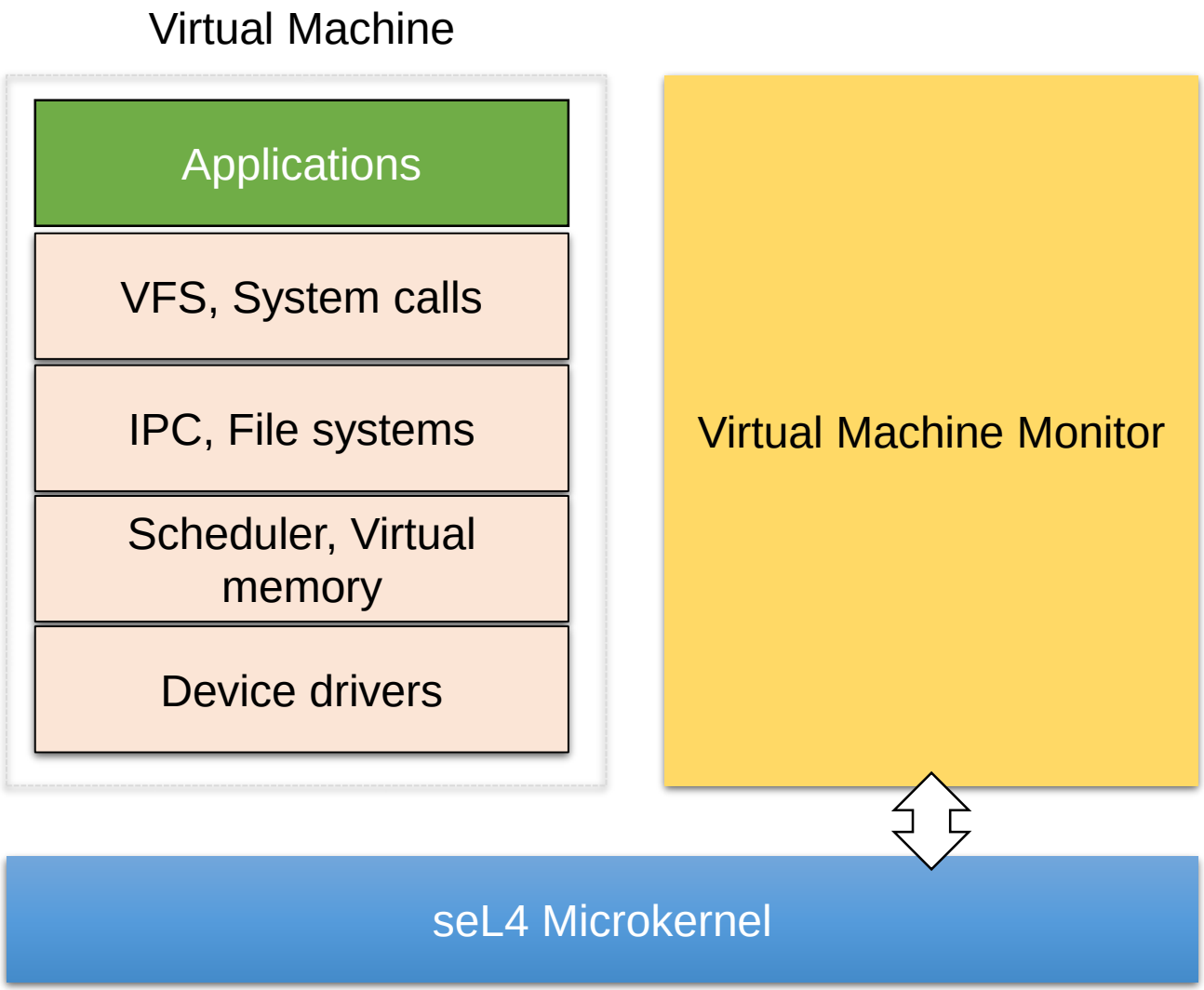
A Pragmatic Challenge from Microkernel

- It is tough to implement applications
  - from scratch with
    - a new (component-based) programming model (e.g., CAmkES)
    - caring more platform/hardware details (e.g., memory management, file system, network drivers, IPC, etc.)
- In practice, few opportunities for engineering a system from scratch for security



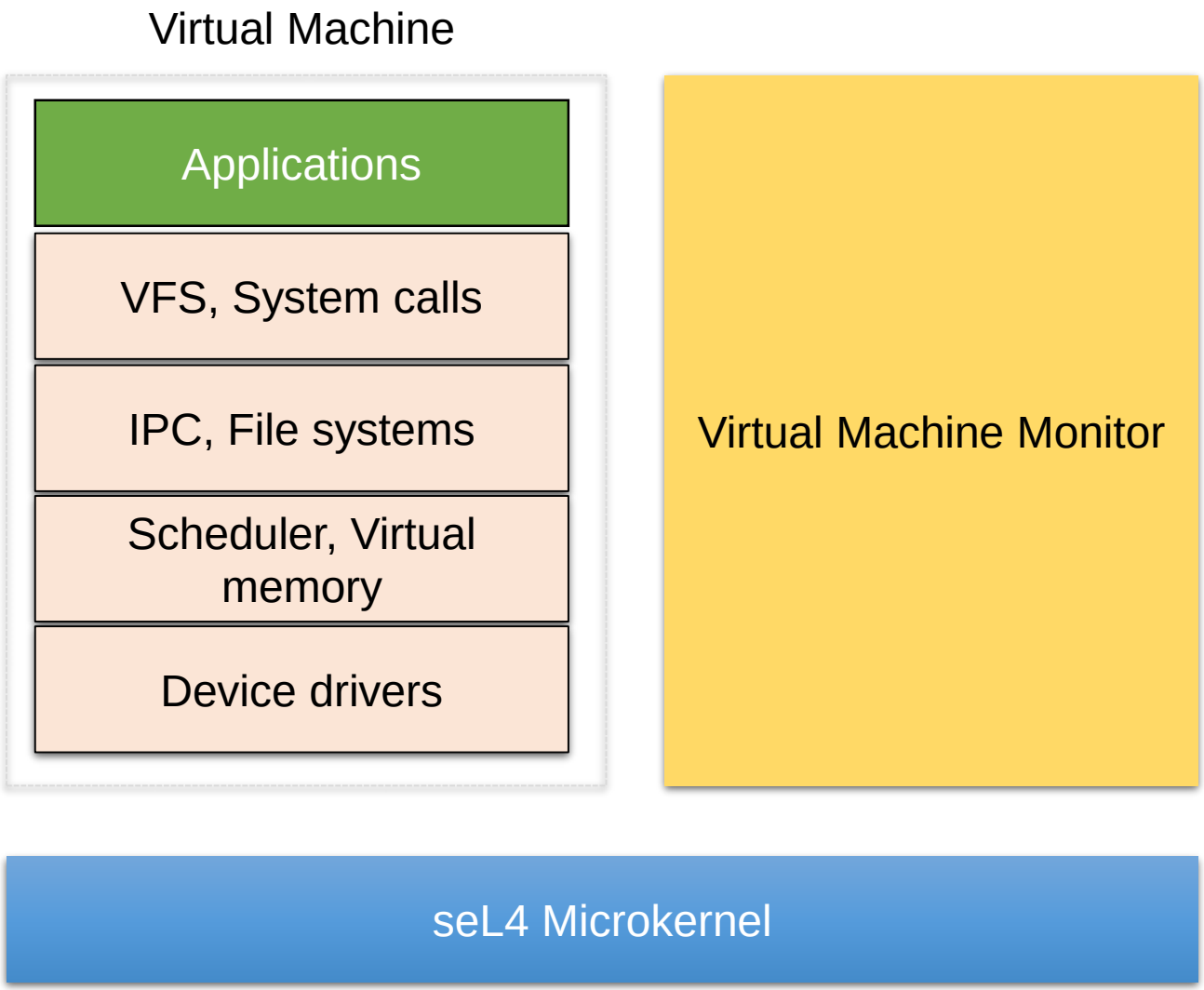
Retrofit seL4 microkernel for legacy applications?

State of the Art – VM Virtualization



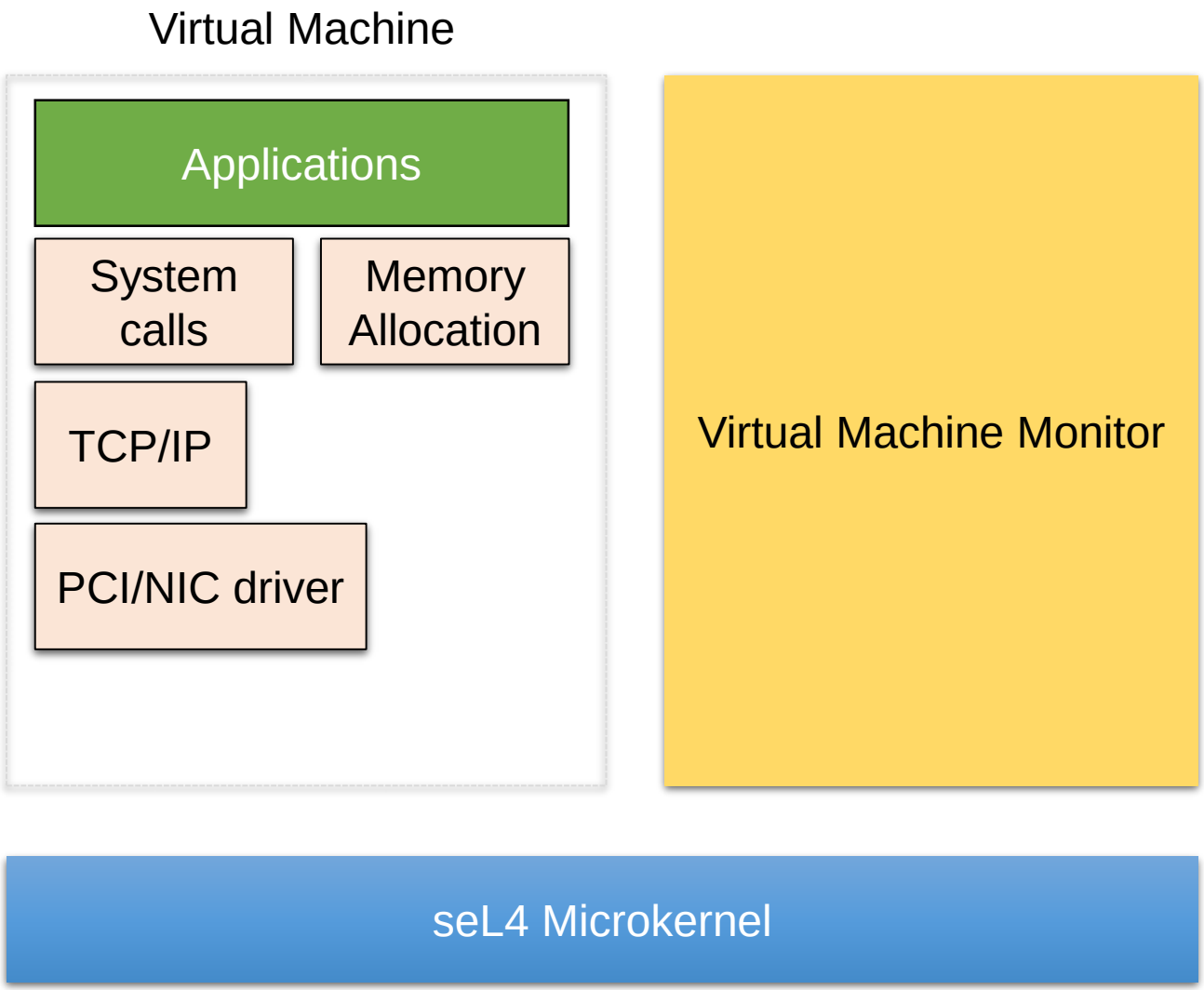
- Performance overhead
- Security concerns

A Light-Weight Alternative?



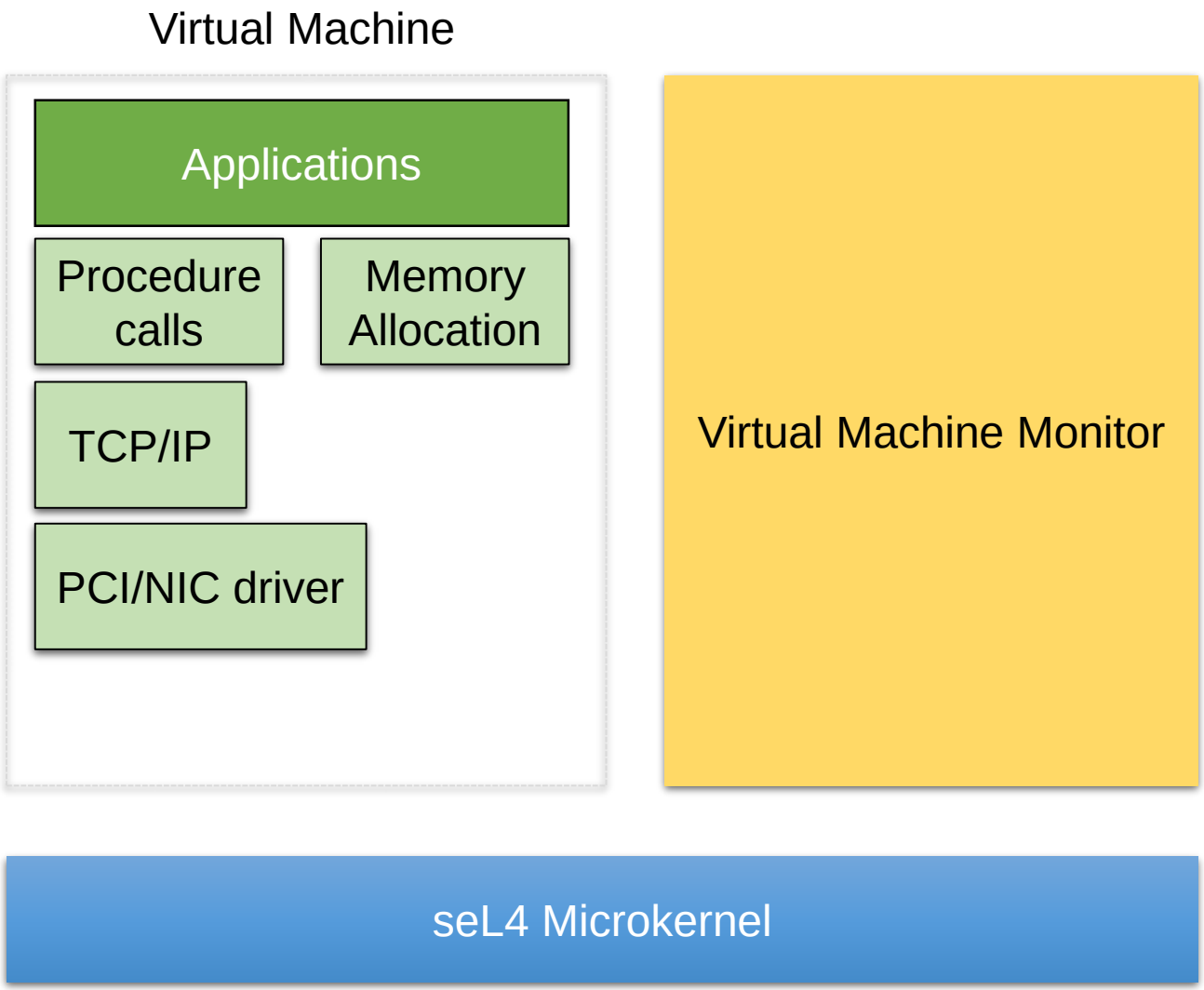
- A tailored kernel with only required drivers and the basic support routines for these drivers to function

A Light-Weight Alternative?



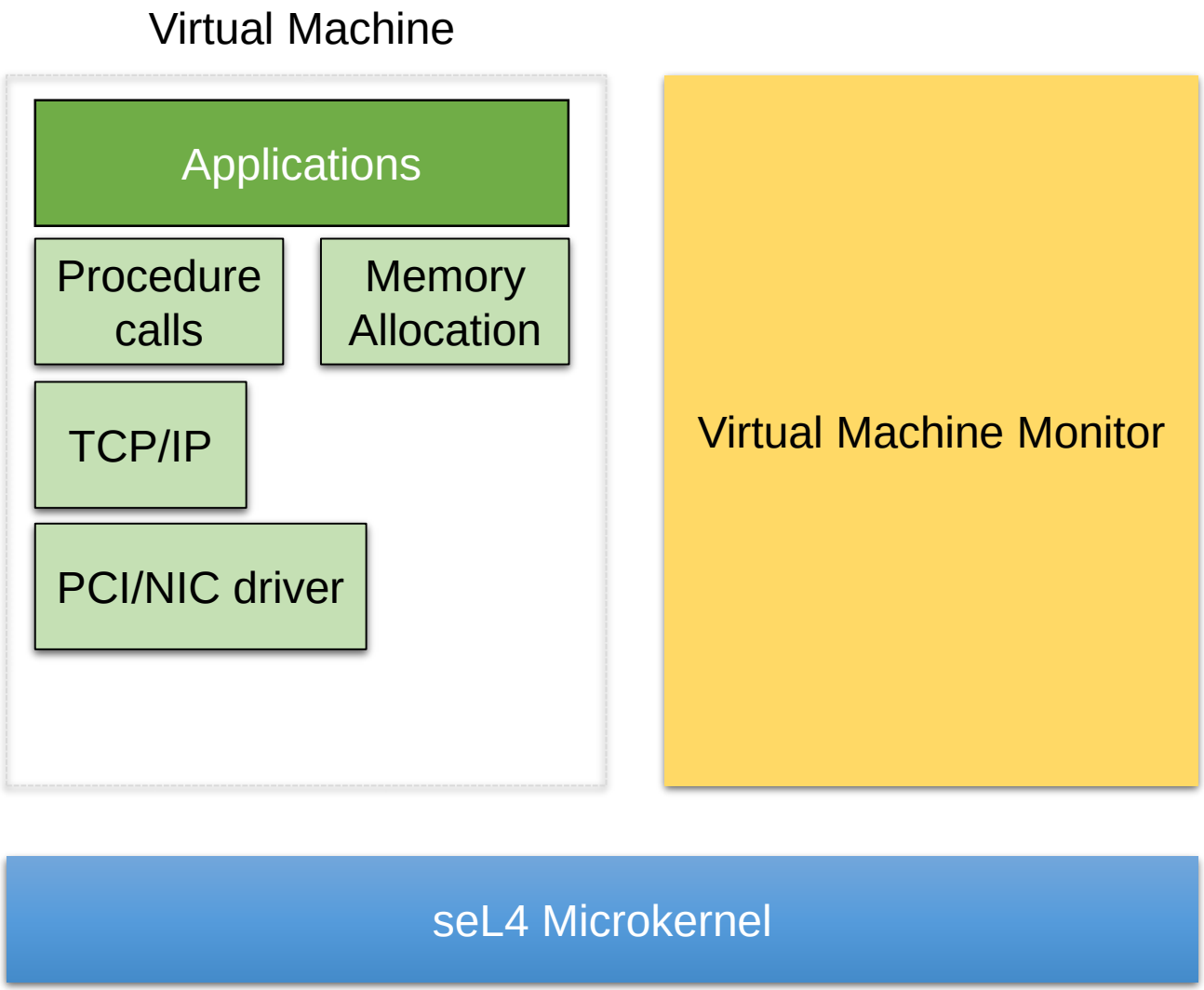
- A tailored kernel with only required drivers and the basic support routines for these drivers to function
  - A minimal “kernel”
- No user/kernel space separation needed

A Light-Weight Alternative?



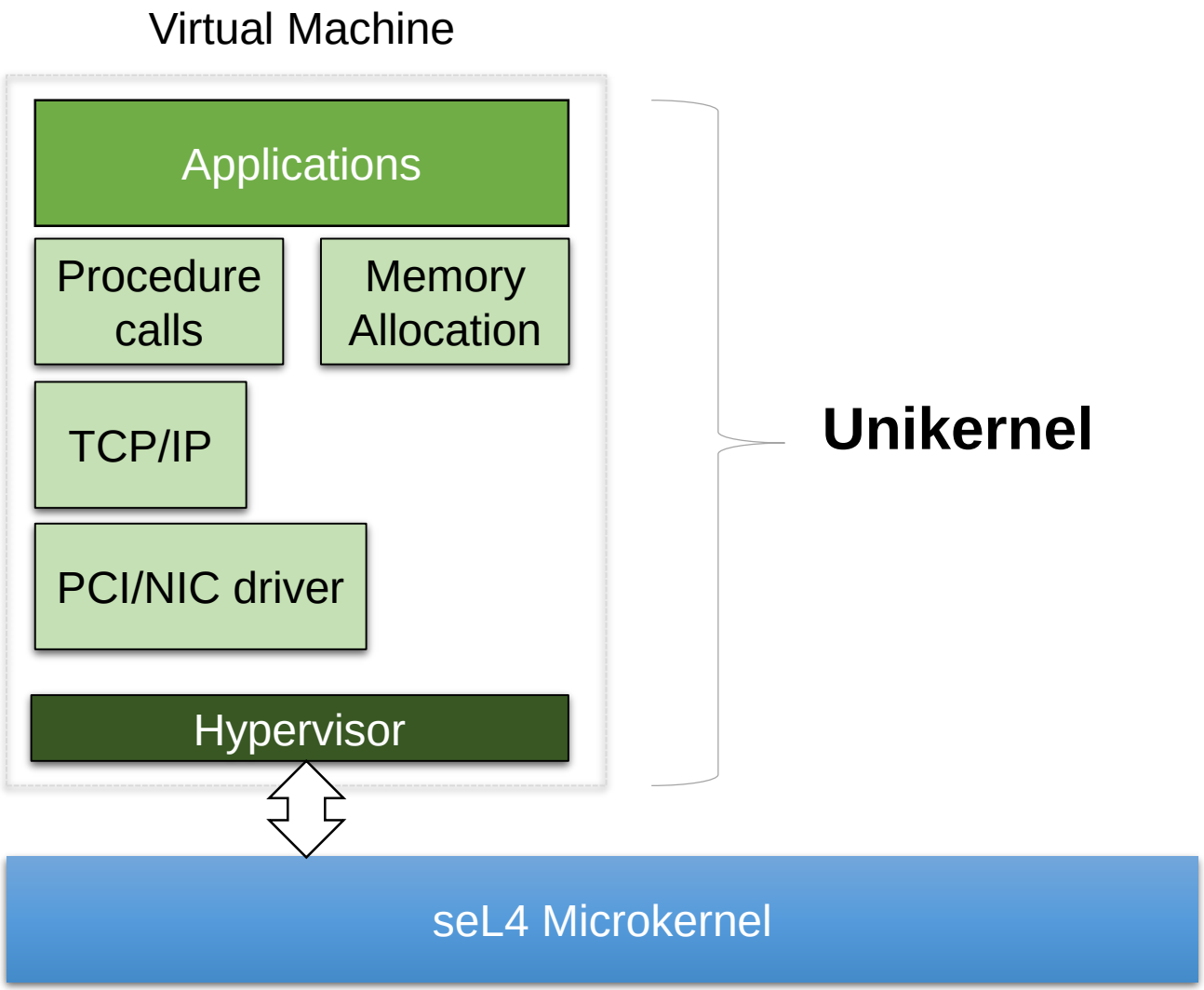
- A tailored kernel with only required drivers and the basic support routines for these drivers to function
  - A minimal “kernel”
- No user/kernel space separation needed
  - Fast access to kernel from user applications

A Light-Weight Alternative?



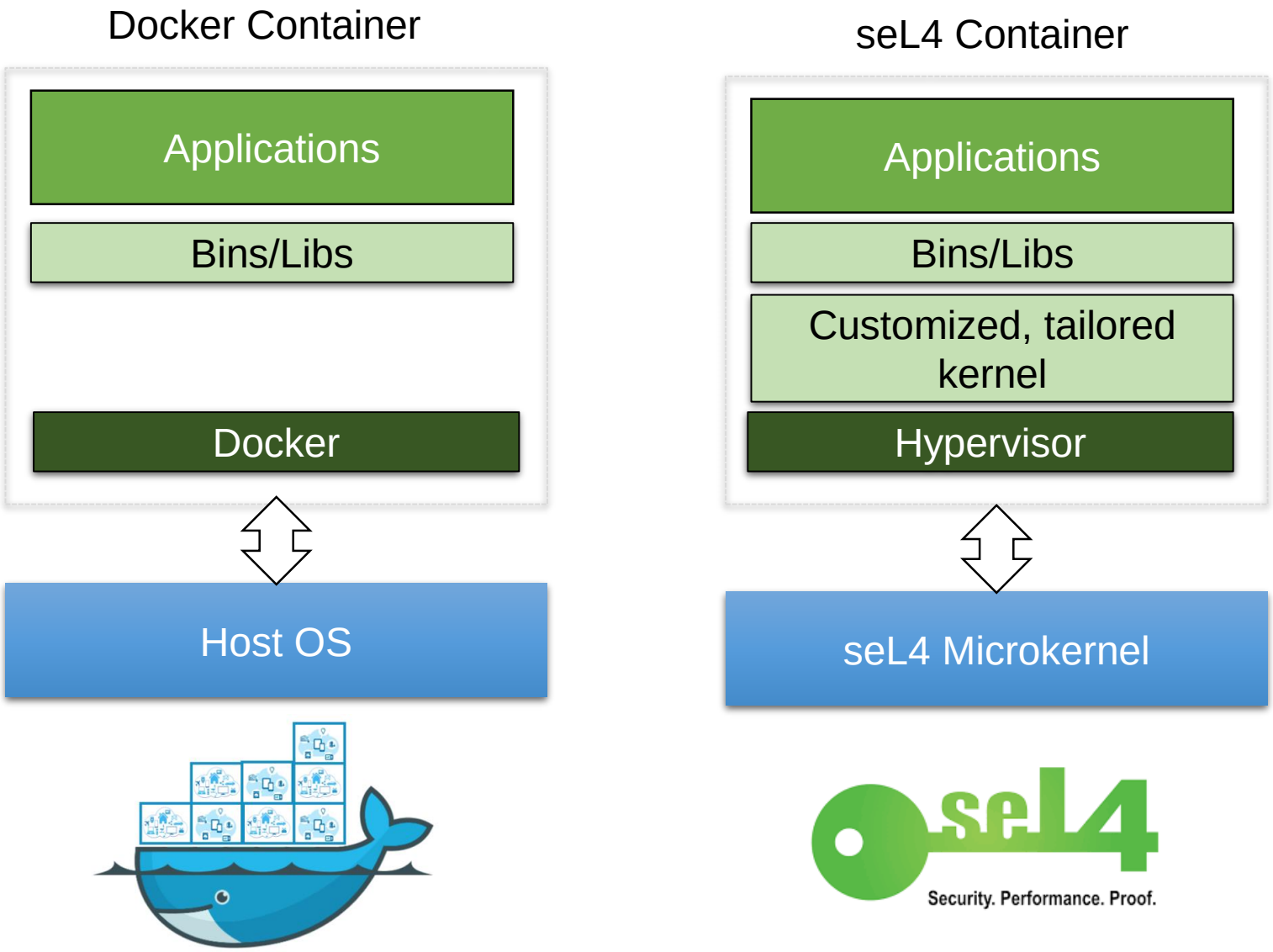
- A tailored kernel with only required drivers and the basic support routines for these drivers to function
  - A minimal “kernel”
- No user/kernel space separation needed
  - Fast access to kernel from user applications
- A thin, platform-specific software layer to access underlying resources

A Light-Weight Alternative?



- A tailored kernel with only required drivers and the basic support routines for these drivers to function
  - A minimal “kernel”
- No user/kernel space separation needed
  - Fast access to kernel from user applications
- A thin, platform-specific software layer to access underlying resources

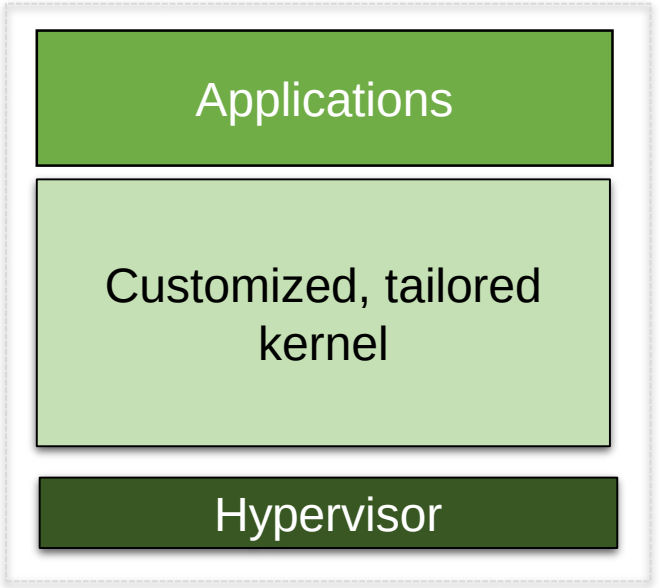
seL4 Container

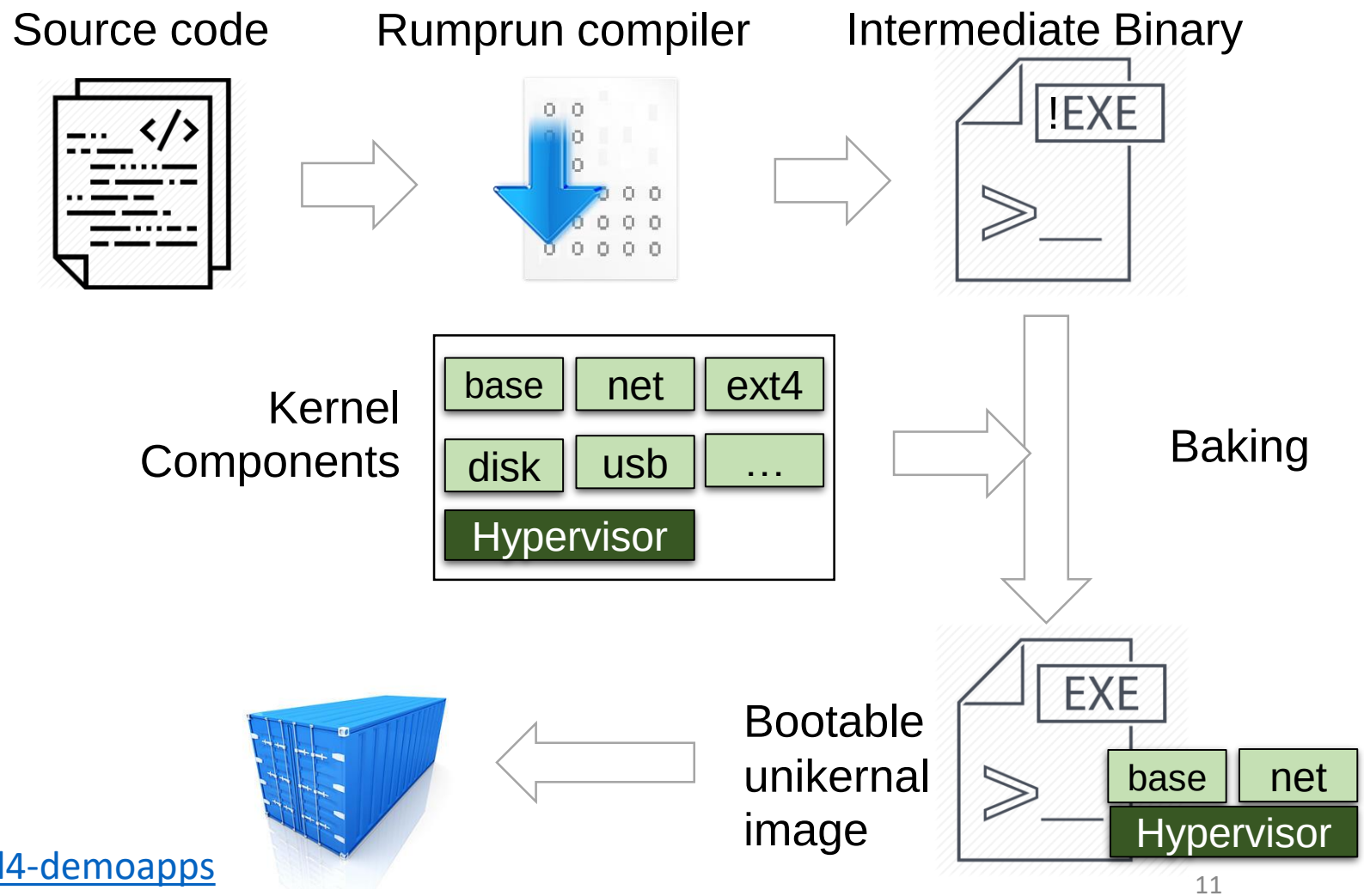


- Run legacy applications as native seL4 threads in a **self-contained environment**, including
  - application code
  - dependencies (libraries)
  - required system services (e.g., file systems and network devices)
  - and hypervisor

A Rumprun Unikernel Implementation

How to build a seL4 container image?

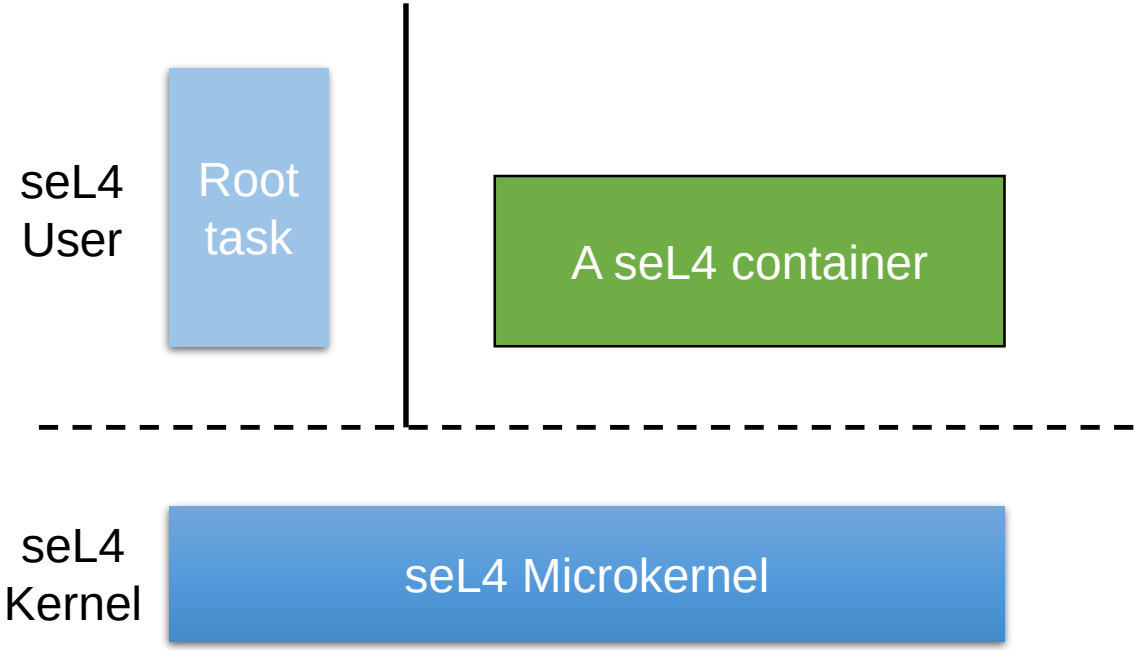




https://github.com/SEL4PROJ/rumprun-sel4-demoapps

A Rumprun Unikernel Implementation

How to launch a seL4 container?

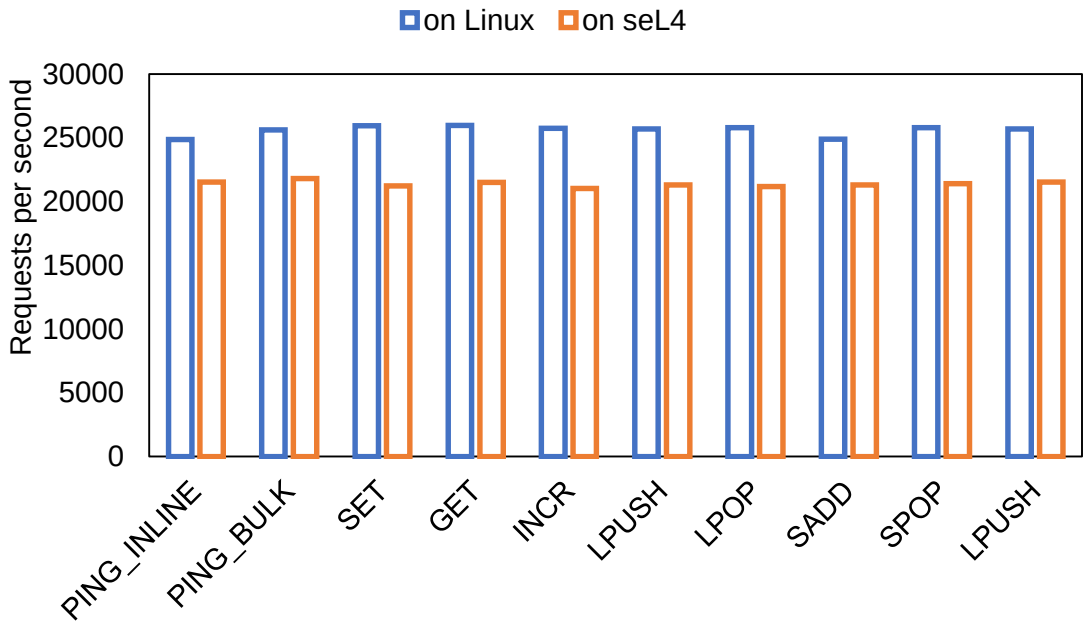


<u>seL4 Root task:</u>

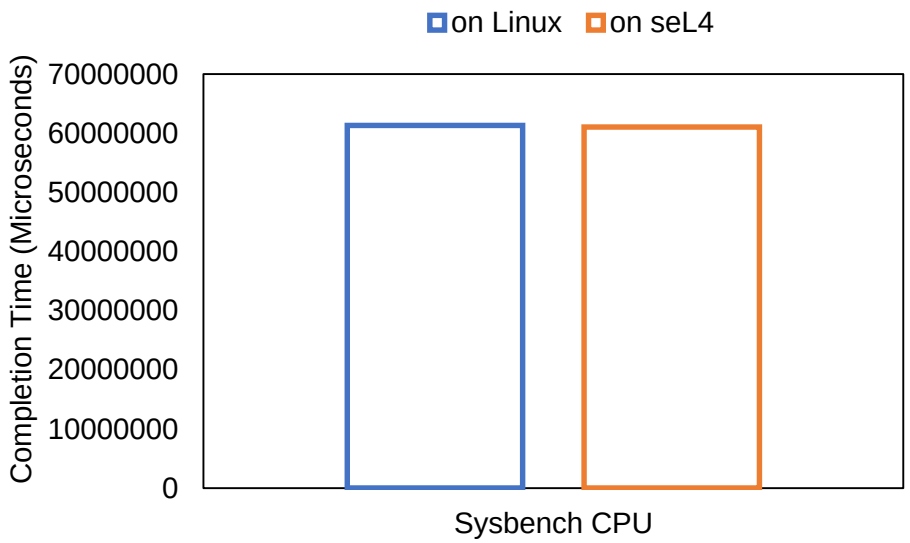
- Prepares the running properties for a seL4 container instance:
  - Capability space
  - Address space
  - I/O space
  - Interrupts
- Launches the seL4 container instance with the container image

Performance Evaluation

- A rough idea about performance of seL4 container

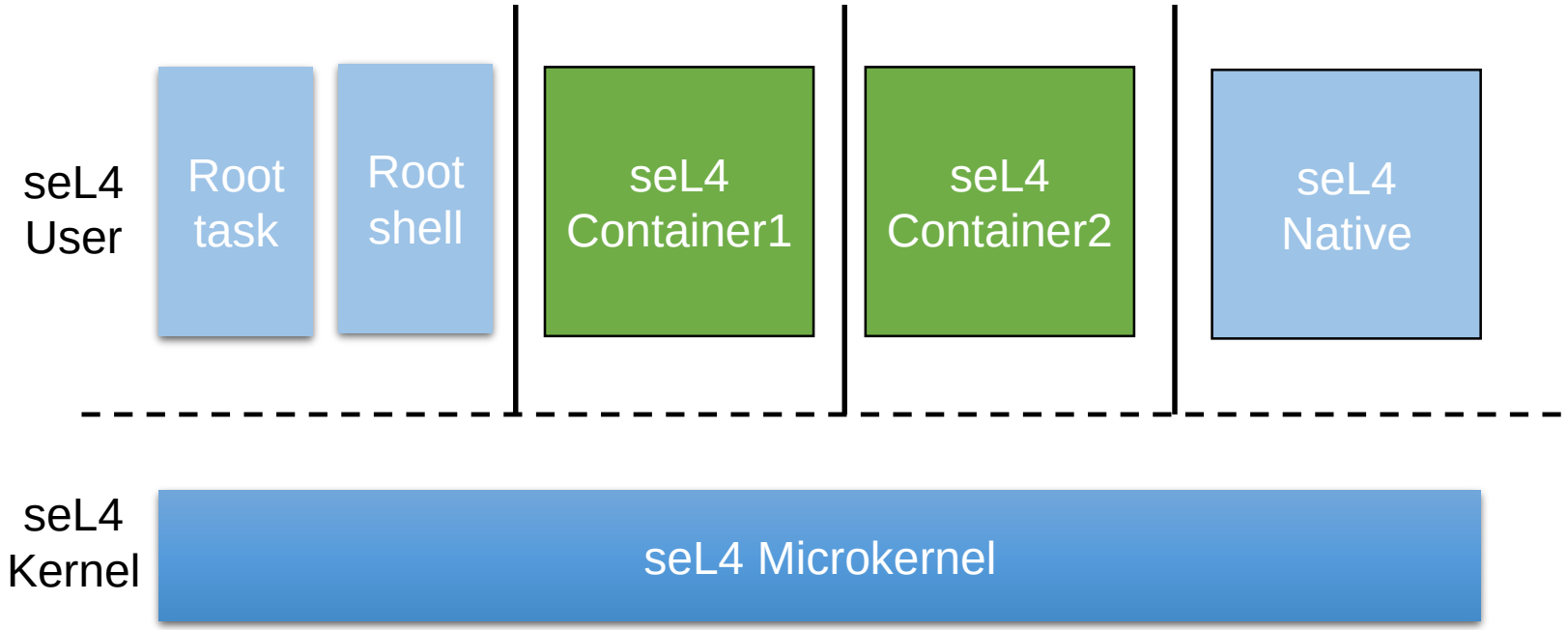


Redis key-value database
(Memory intensive)



Sysbench (CPU intensive)

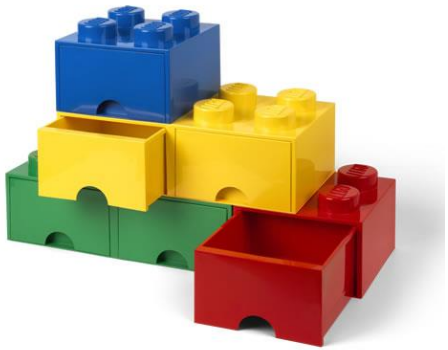
Supporting Multiple seL4 Containers



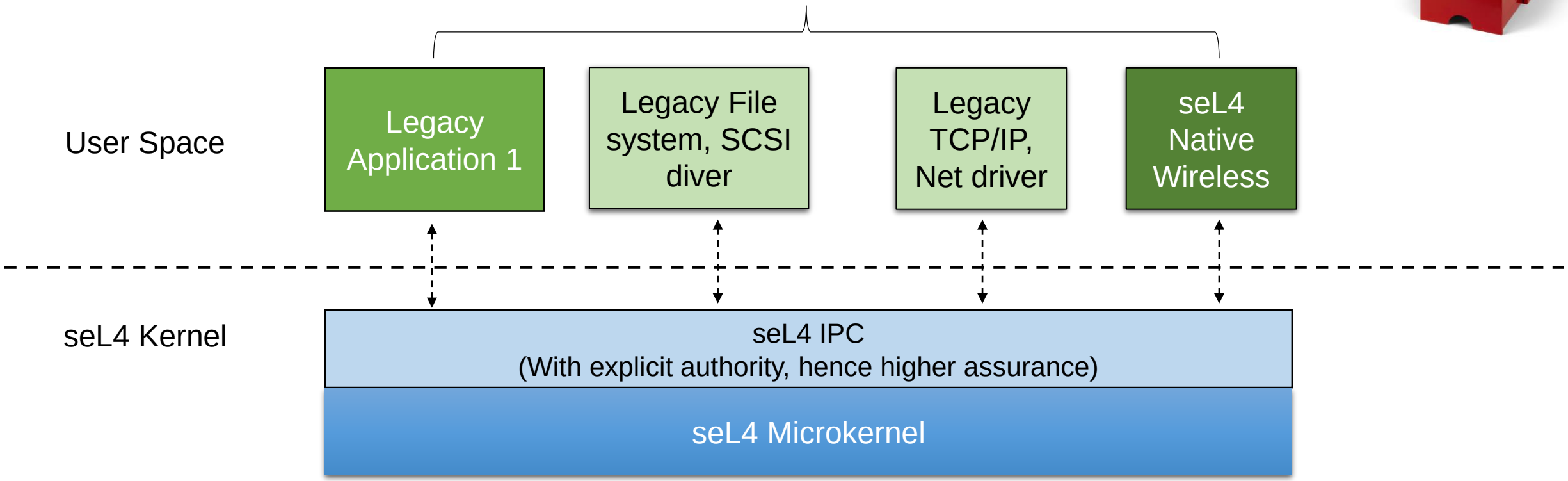
Limitations

- Need non-trivial effort to port a legacy application
- A “single-core” solution
- Non-preemptive scheduling
- Hardware resources are not shared
- …

Future Work: An Enhanced Design



A **seL4 container** running as a group of customized, composable seL4 threads



Conclusions

- “Small” kernel space -> “Big” user space
- More design exploration in the user space is needed

- seL4 containers could serve as one option to support legacy applications in a **light-weight** manner
- We do **not** need extra support from seL4 kernel
- We do need the source code of legacy applications

Thanks

- Lok Yan for advising throughout the project
- Daniel Limbrick for insightful discussions
- https://github.com/SEL4PROJ/rumprun-sel4-demoapps
- Air Force Research Lab and Griffis Institute