FlexCap: Exploring Hardware Capabilities in Unikernels and Flexible Isolation Oses

CHERITech'24

Pierre Olivier
pierre.olivier@manchester.ac.uk

Objectives

- Investigate on Morello (real hardware), in the context of specialised minimal single address space operating systems (**unikernels**), the following:
  - Various approaches to **compartmentalisation in CHERI hybrid capability mode** and the resulting trade-offs in terms of performance, scalability, security, and engineering effort
  - The benefits of **safe C** obtained through the use of **CHERI pure capabilities**
  - How CHERI can help address one of the fundamental challenges of single address space operating systems: **multi-process applications support**

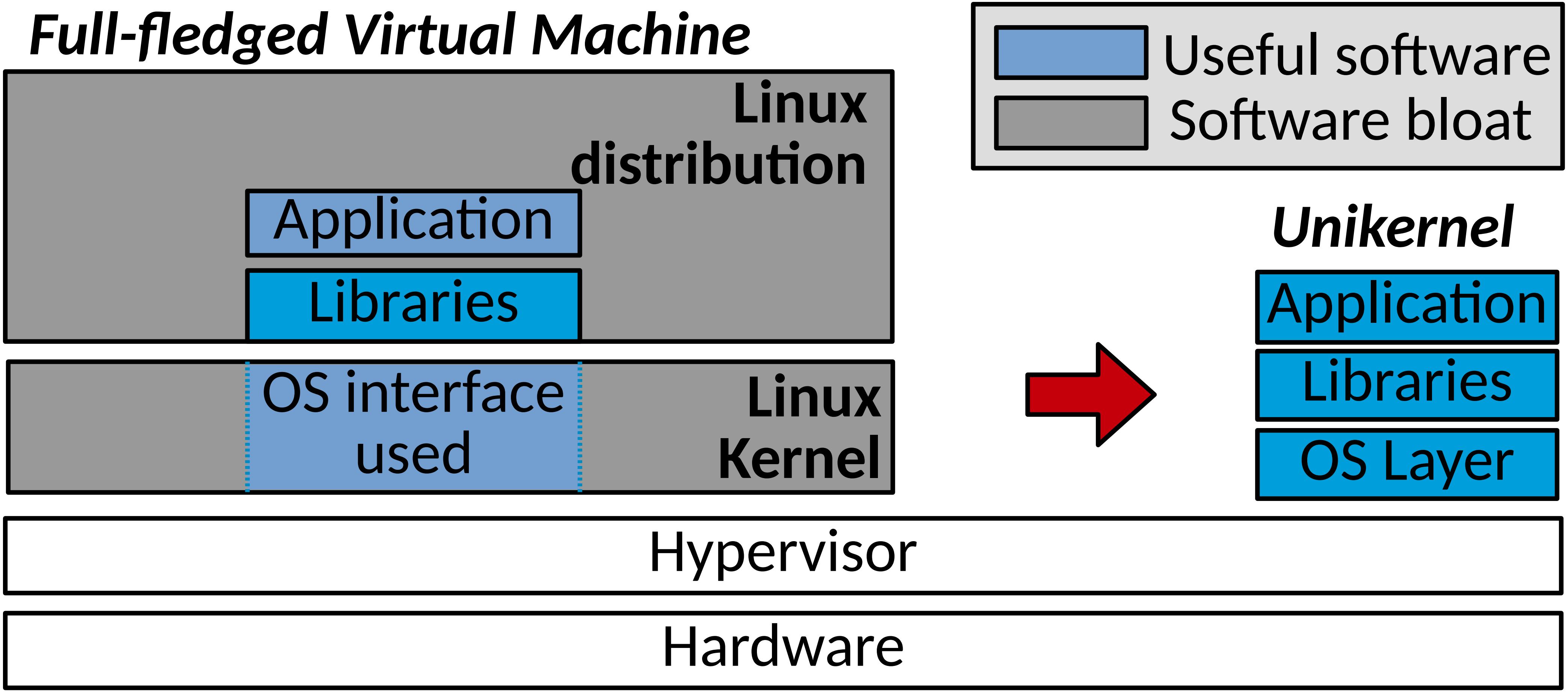
Outline

1) Unikernels
2) Progress on Unikernel Compartmentalisation
3) Progress on Purecap Unikernels
4) Progress on Support for Multi-Process Applications
5) Conclusion

Outline

1) **Unikernels**
2) Progress on Unikernel Compartmentalisation
3) Progress on Purecap Unikernels
4) Progress on Support for Multi-Process Applications
5) Conclusion

Unikernels



Full-fledged Virtual Machine
Linux distribution
Application
Libraries
OS interface used
Linux Kernel
Useful software
Software bloat
Unikernel
Application
Libraries
OS Layer
Hypervisor
Hardware

Unikernels

- **Unikernel: application + dependencies + thin OS layer compiled as a static binary running on top of a hypervisor**
- **Single purpose OS:** 1 instance runs 1 application
    - OS can be specialised for the app (libOS/Exokernel model)
- **Lightweight:** fast boot time, low memory/disk footprint
    - Costs and attack surface reduction
- **Single binary and single address space** for the OS + application
    - ***No isolation within the unikernel***, system calls are (fast) function calls

Madhavapeddy et al., “Unikernels: Library Operating Systems for the Cloud”, ASPLOS’13

Unikernels + CHERI/Morello

- **Single address space** nature of unikernels aligns well with the protection model suggested by CHERI/Morello
- Today the **lack of isolation** inside a unikernel instance is concerning
- Motivated us to study bringing the security benefits of CHERI's compartmentalisation/safe C for unikernels while maintaining their lightweight and high-performance nature

Outline

1) Unikernels
2) **Progress on Unikernel Compartmentalisation**
3) Progress on Purecap Unikernels
4) Progress on Support for Multi-Process Applications
5) Conclusion

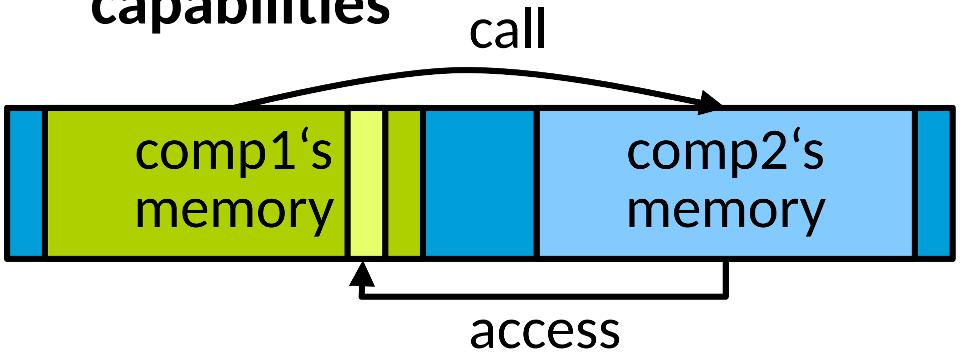
Effort on Compartmentalisation

- Basic port of FlexOS[1] (compartmentalisation-aware version of the Unikraft[2] Unikernel) **to run bare metal on Morello A64**
- Development of compartmentalisation abstractions leveraging **hybrid mode** (for compatibility) with protection domains defined by DDC/PCC
- Development of **two methods of cross-compartment data sharing** trading off engineering effort/scalability to many compartments/security
- **Performance, security and engineering effort evaluation** of the prototype with 2 popular applications

[1]Lefeuvre et al., "FlexOS: Towards Flexible OS Isolation", ASPLOS'22
[2]Kuenzer et al. "Unikraft: fast, specialized unikernels the easy way", EuroSys'21

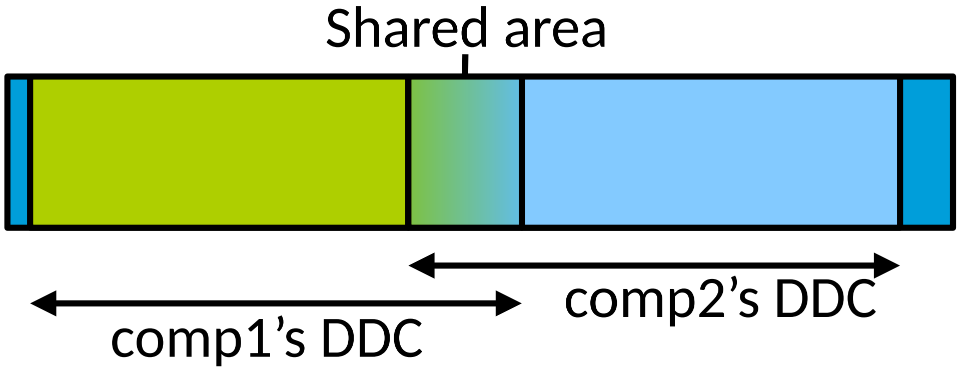
Compartmentalisation: Cross-Compartments Data Sharing

- Method 1: **pass shared data as capabilities**



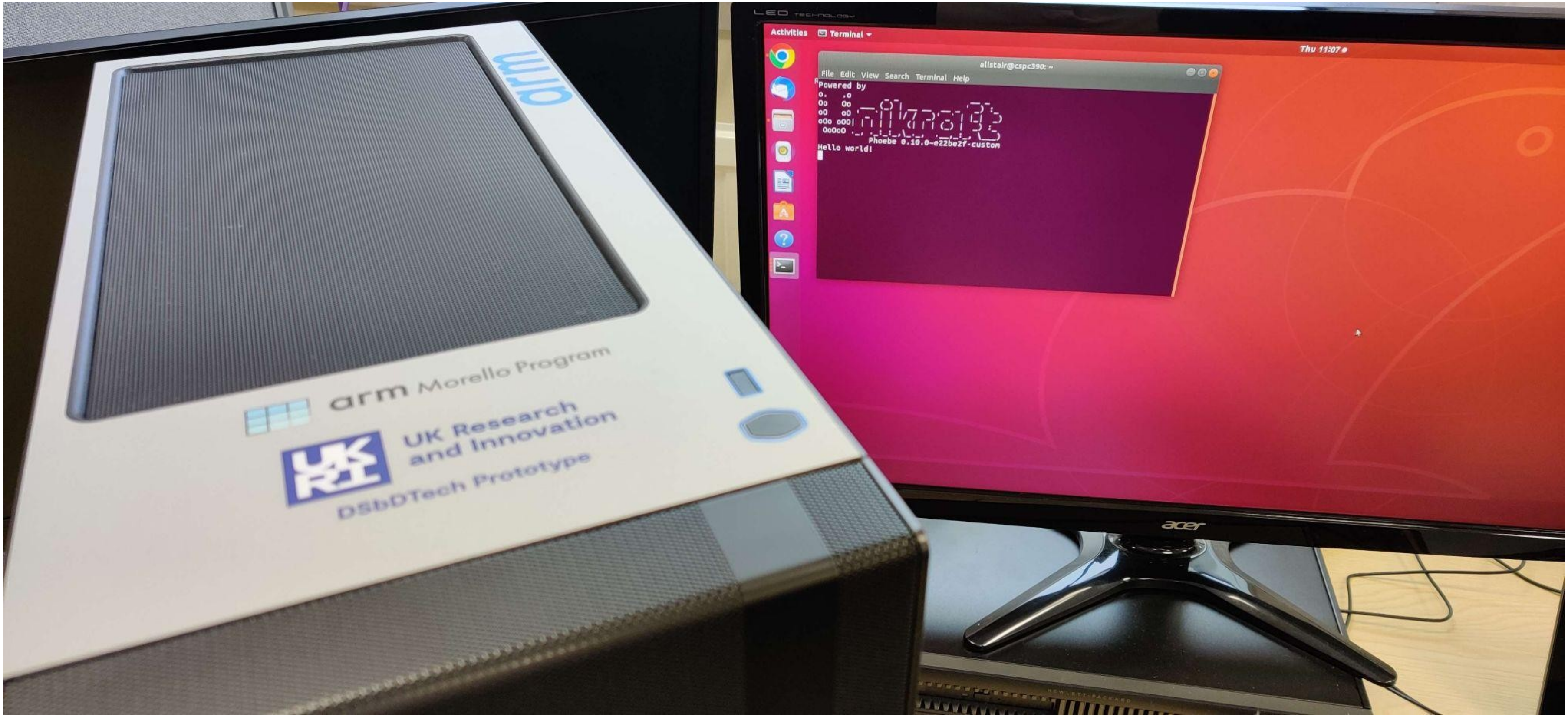
- Pros: security, scalability to high numbers of compartment
- Cons: **engineering effort**/scalability to large compartments

- Method 2: **overlapping DDCs**



- Pros: low engineering effort
- Cons: security, scalability to many compartments

Unikraft/FlexOS on Morello



https://unikraft.org/blog/2022-12-01-unikraft-on-morello

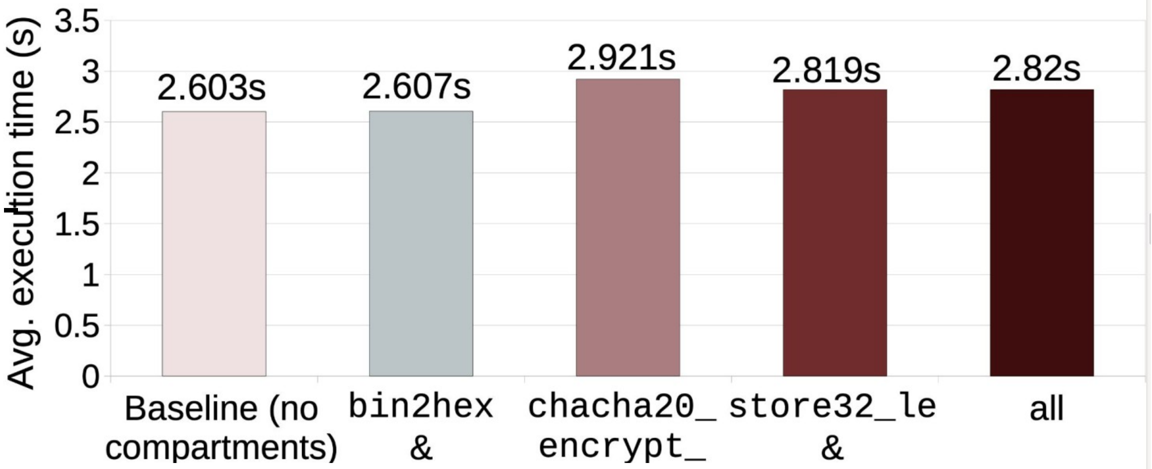
Compartmentalisation: Evaluation

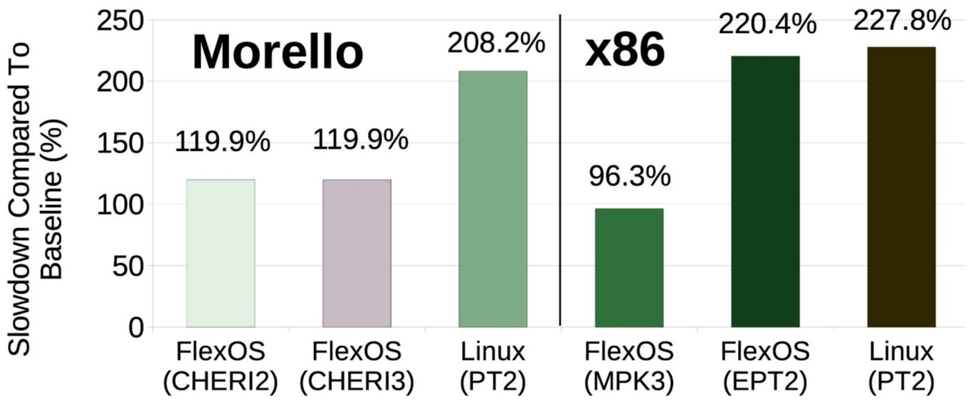
Shared data as capabilities:

- With carefully selected functions overhead is low (Libsodium: 0.1%-12.2%)

Overlapping DDC:

- Performance overhead same order of magnitude to MPK and lower than EPT on FlexOS (SQLite)
- Runs faster than same benchmark on Linux with user/kernel isolation (SQLite)



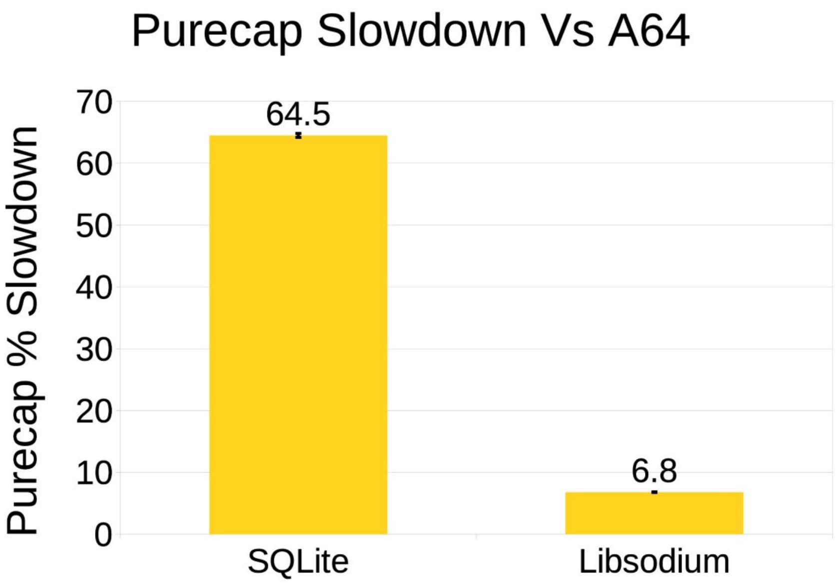


Outline

1) Unikernels
2) Progress on Unikernel Compartmentalisation
3) **Progress on Purecap Unikernels**
4) Progress on Support for Multi-Process Applications
5) Conclusion

Purecap Unikraft

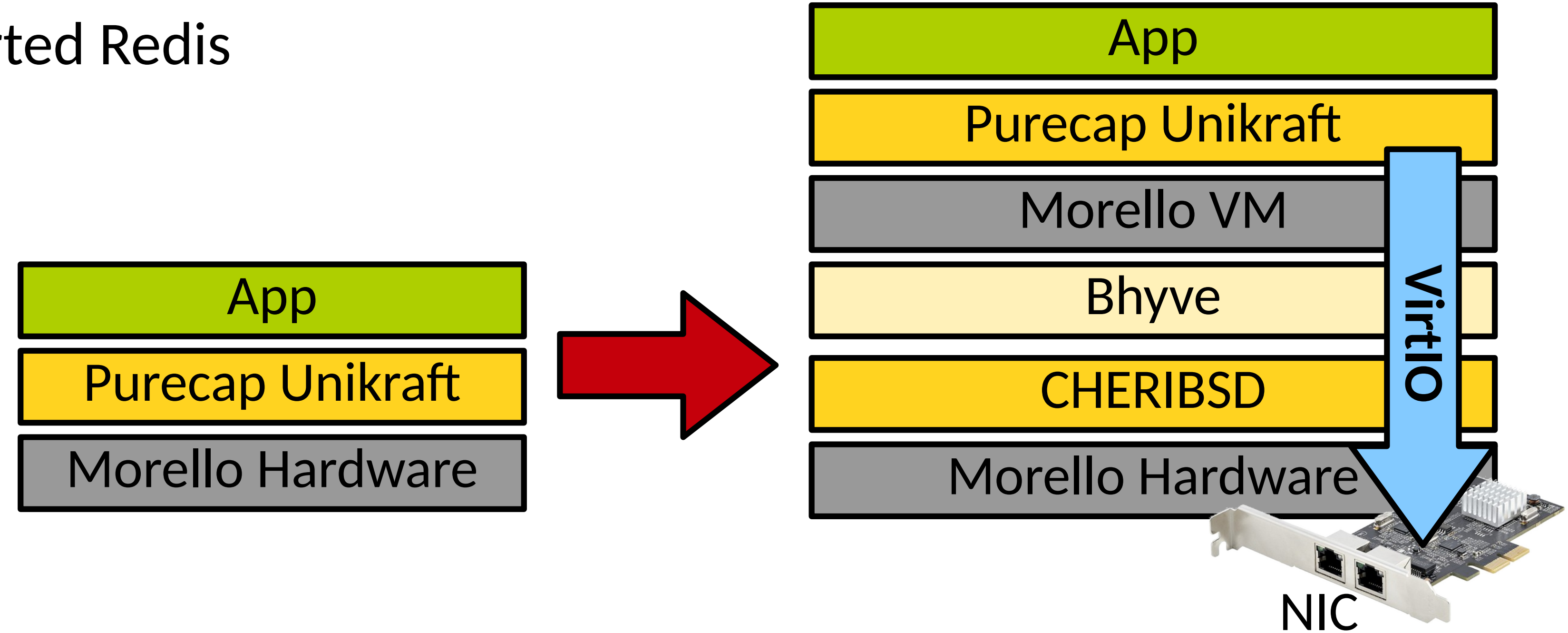
- We ported **Unikraft Unikernel OS to run bare metal on Morello in purecap mode**
  - Updates made to the platform code (boot process), memory allocator, and various other low-level subsystems (pointer arithmetics)
- We ported **libsodium and SQLite to run on top of purecap Unikraft**



Benchmark	Avg. A64 cycles	Avg. Purecap cycles
SQLite	2.6M	4.2M
Libsodium	130.2M	139M

Beyond Bare Metal

- A custom OS on bare metal is limited by the lack **drivers for I/O**
- Ported Unikraft to run on top of the **bhyve hypervisor** and to use the **virtio-net paravirtualised network driver**
- Ported Redis



Beyond Bare Metal

```
CRIT: [libredis_server] Server config file /redis.conf
1:C 01 Jan 1970 00:00:00.091 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 01 Jan 1970 00:00:00.093 # Redis version=5.0.6, bits=64, commit=c5ee3442, modified=1, pid=1, just started
1:C 01 Jan 1970 00:00:00.094 # Configuration loaded
CRIT: [libredis_server] Pre init server
CRIT: [libredis_server] post init server
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.6 (c5ee3442/1) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
 |    `-._   `._    /     _.-'    |     PID: 1
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

1:M 01 Jan 1970 00:00:00.113 # Server initialized
1:M 01 Jan 1970 00:00:00.114 * Ready to accept connections
```

Outline

1) Unikernels
2) Progress on Unikernel Compartmentalisation
3) Progress on Purecap Unikernels
4) **Progress on Support for Multi-Process Applications**
5) Conclusion

SASOses & POSIX `fork()`

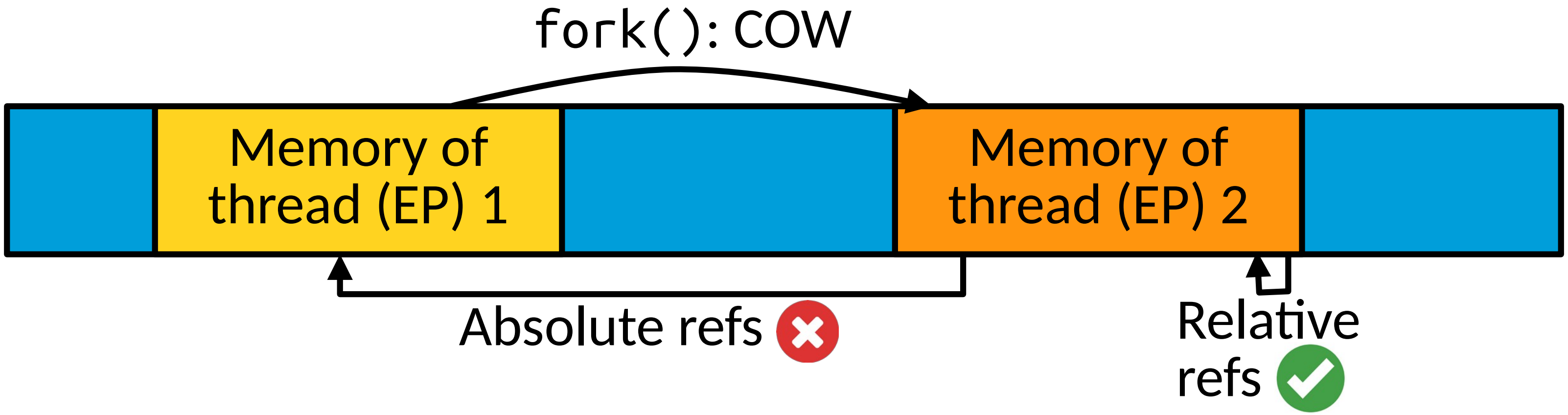
- Support for multi-process applications (i.e. POSIX `fork()`) is a well-know design limitation of single address space OSes (SASOSes)
- Existing solutions for unikernels[1,2] spawn 1 unikernel per process and implement IPCs in the hypervisor
  - This break the fundamental ”single address space“ nature of SASOSes, loose some benefits
- **How can we support `fork()` within a single address space?**

[1]Zhang et al., “KylinX: A Dynamic Library Operating System for Simplified and Efficient Cloud Virtualization“, ATC’18

[2]Lupu et al., “Nephele: Extending Virtualization Environments for Cloning Unikernel-Based Vms“, EuroSys’23

SASOses & POSIX fork()

- Key idea: **emulate processes with** threads
  - Locate the memory (code & data) relevant to each emulated process (EP) in a specific area
  - Upon fork, COW that space somewhere else and create another thread
  - **Challenges: inter-emulated process isolation, memory references**



SASOses & POSIX fork()

- **CHERI-powered solutions:**
  - Assume **PIE** to maximise relative memory references, and fixup absolute references on-demand during the COW by **scanning tagged memory to track pointers**
  - Inter-emulated processes **isolation based on purecap**, becomes a twofold problem:
    1) Ensure no capability leak between parent and child (i.e. properly fixup all absolute references during COW)
    2) Segregate and isolate the kernel's memory from the emulated processes as it is now an ambient authority
- Solutions in the process of being implemented

Outline

1) Unikernels
2) Progress on Unikernel Compartmentalisation
3) Progress on Purecap Unikernels
4) Progress on Support for Multi-Process Applications
5) **Conclusion**

Conclusion

- We look at various aspects of single address space OSes running on top of Morello:
  - **Hybrid compartmentlisation**
  - **Safe C/purecap**
  - **Support for multiprocess applications**
- Please come check out our poster on `fork()` support in SASOSes today
- And our PLOS'23 paper:
  J. Kressel, H. Lefeuvre, P. Olivier, **Software Compartmentalization Trade-Offs with Hardware Capabilities**, PLOS'23
- https://flexcap-project.github.io/