School of Computer Science & Engineering

Trustworthy Systems Group

Carrels: Secure, High-performance Microservices on seL4

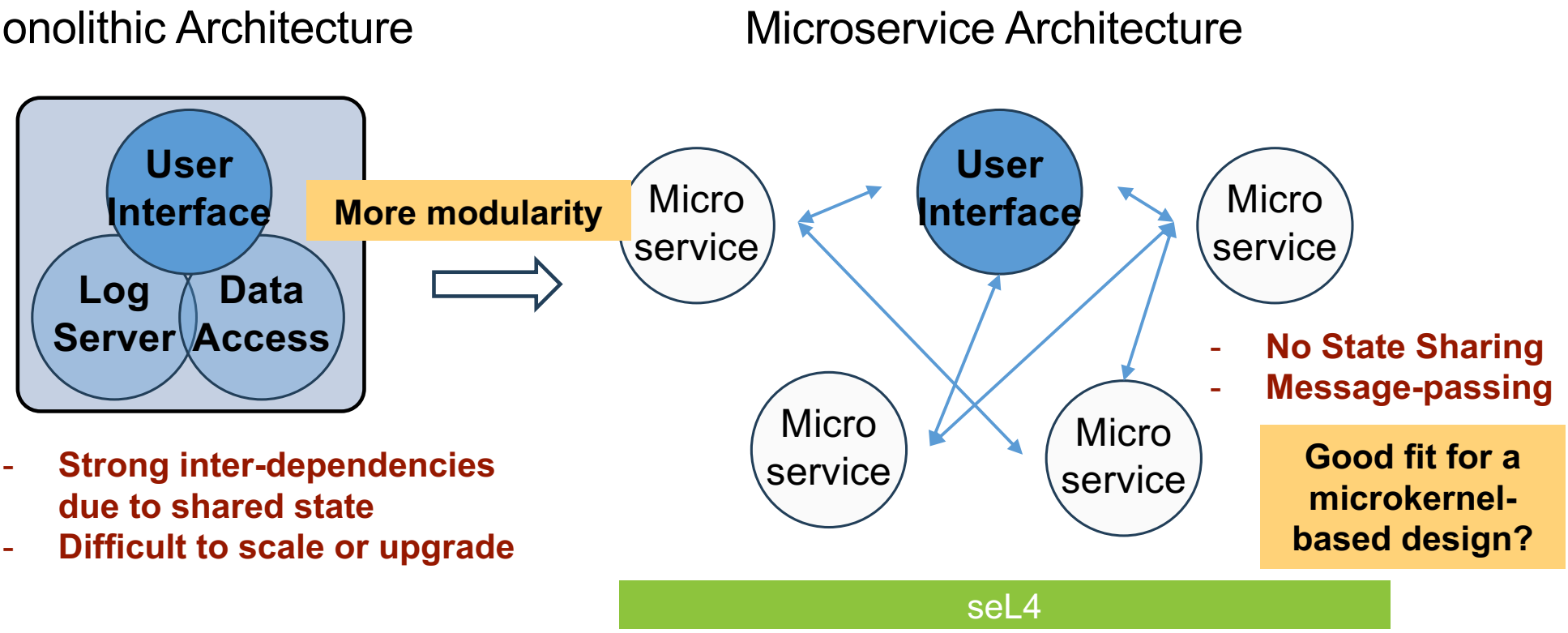
Guangtao Zhu

(presented by Gernot)

guangtao.zhu@unsw.edu.au

1. **Microservices and seL4**
2. Microservice Deployment and Life-cycle
3. Orchestration
4. Programming API

Microservices



Carrels: Microservices on seL4

LionsOS as starting point

Design Aspects:

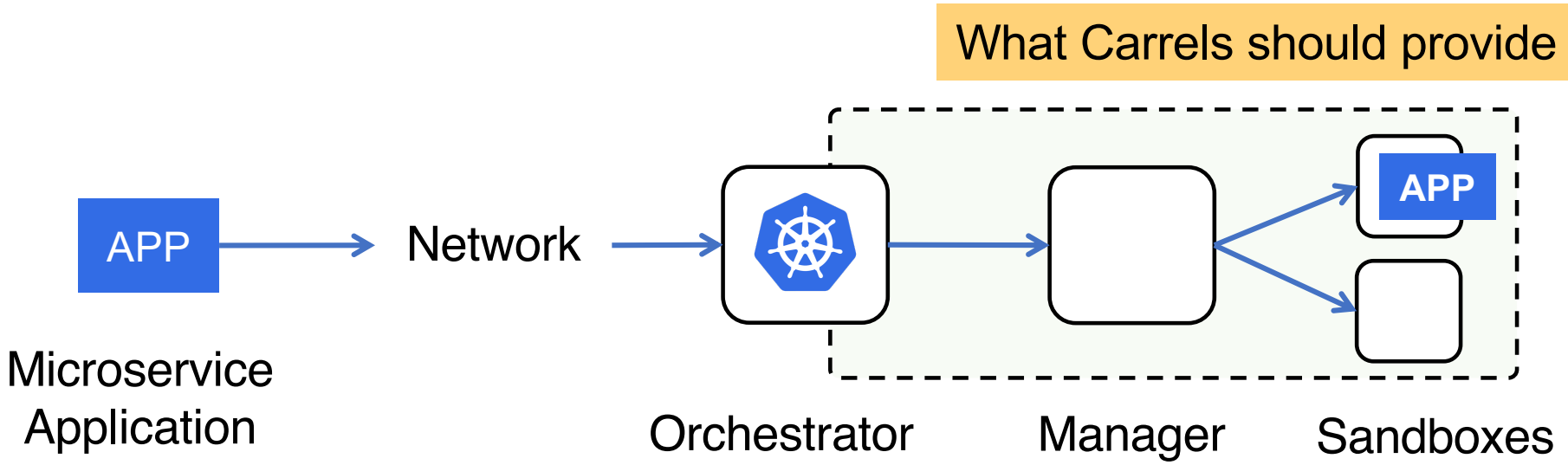
1. Support microservice live-cycle changes

2. Orchestration: managing services

3. Familiar API

1. Microservices and seL4
2. **Microservice deployment and life-cycle**
3. Orchestration
4. Programming API

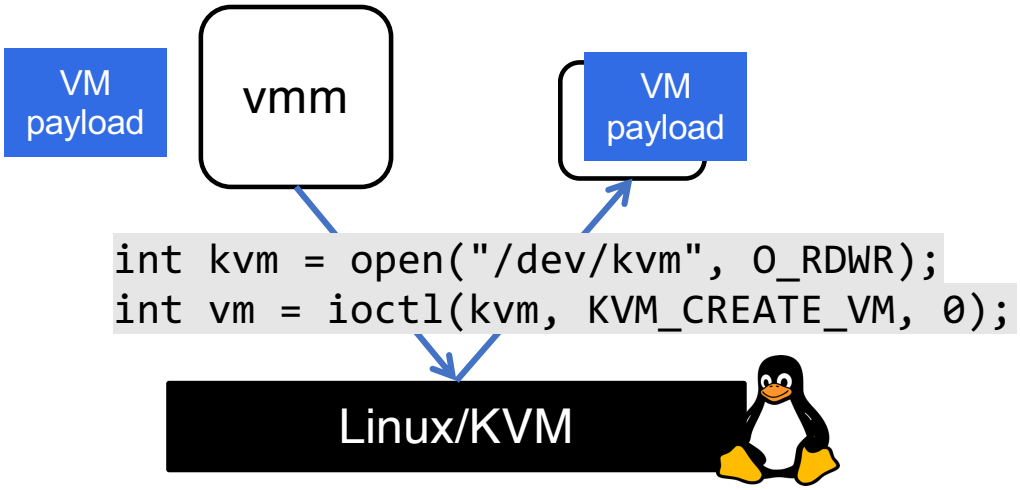
Microservice Deployment Chain



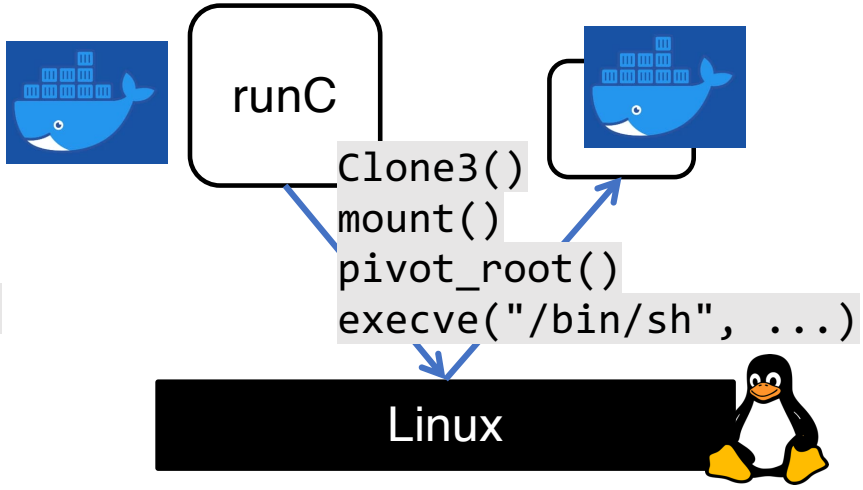
Payload Deployment in Sandboxes

How make pattern work on seL4?

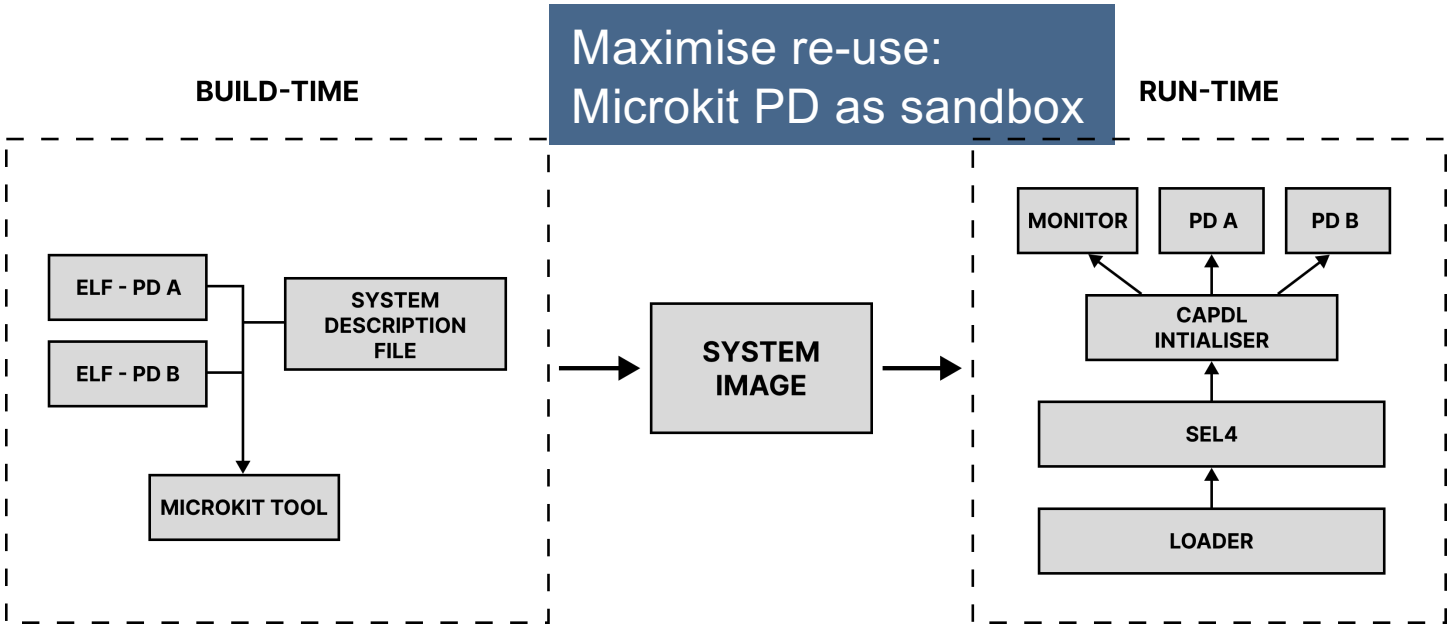
Case study: Linux VM



Case study: Linux Container



Microkit



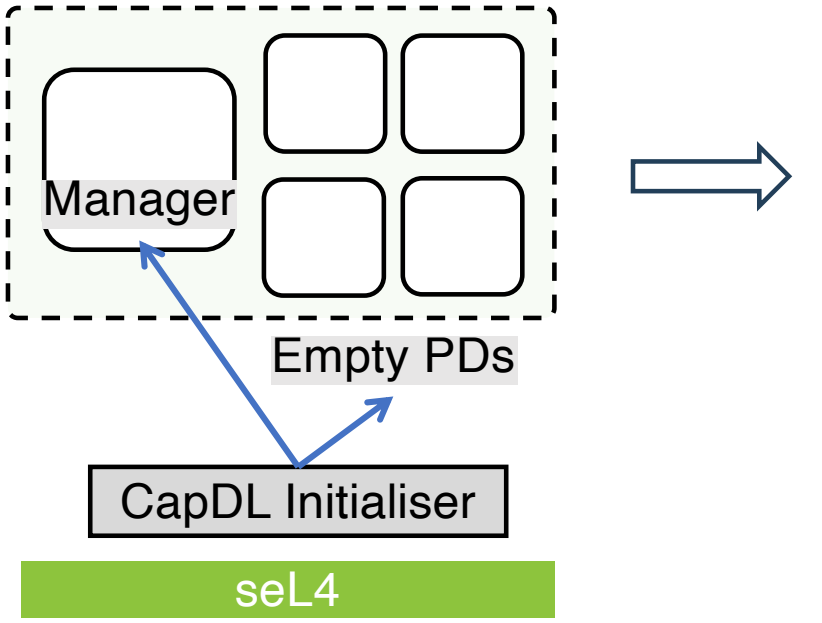
Need support for life-cycle changes

Trusted component to control PD changes

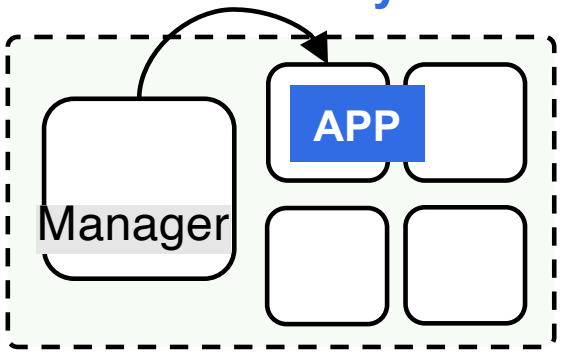
- ✓ Change the execution state of a PD
- ➢ Change the program image of a PD
- ➢ Change the resource view of a PD

PDs as Sandboxes

- Partition resources at boot time
- Pre-provision PDs



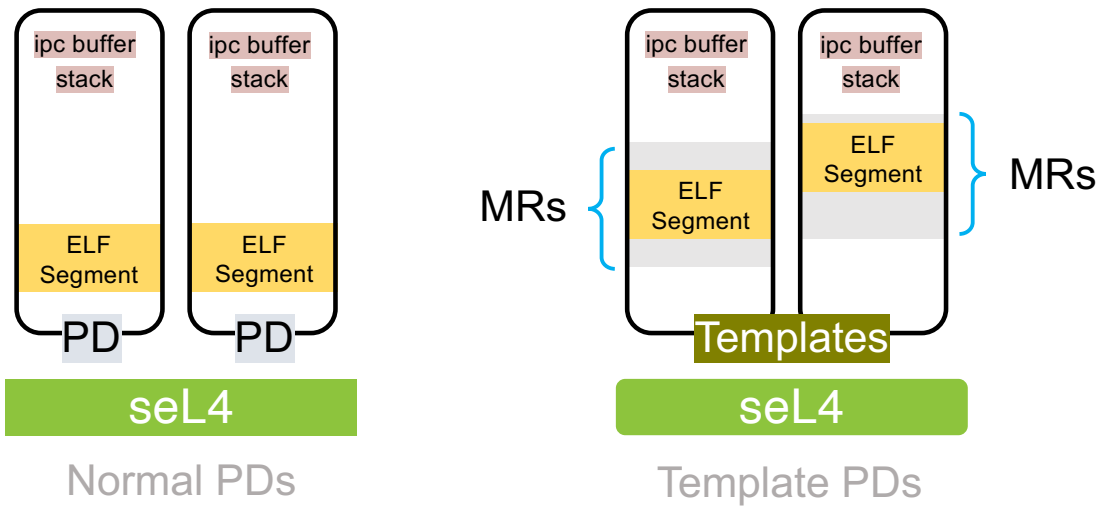
Instantiate empty PDs dynamically



Need to more dynamism in Microkit

Loading a New Image

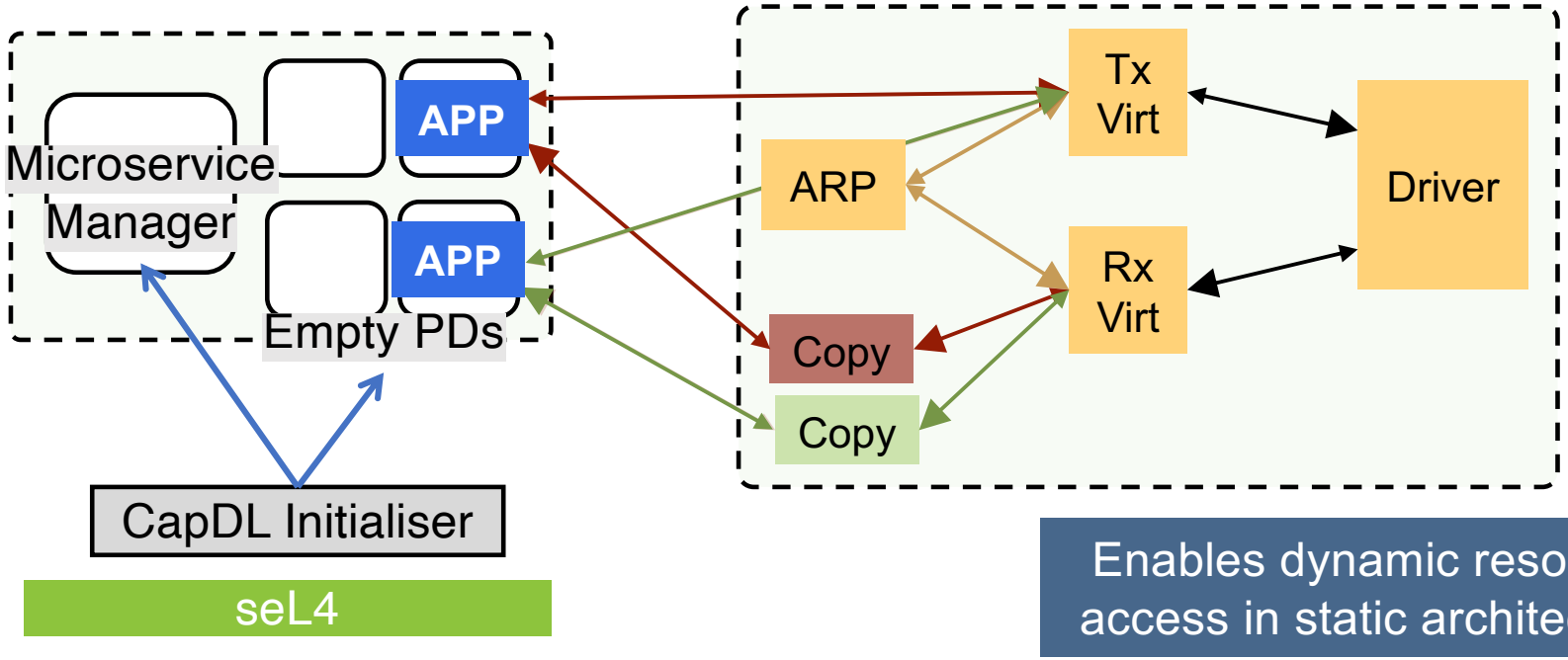
- **Template PD**
  - PDs without an image by default



- Template PD needs a fault handler and a loader
  - The loader requires access to the cap of template PD's thread control block

Resource Management

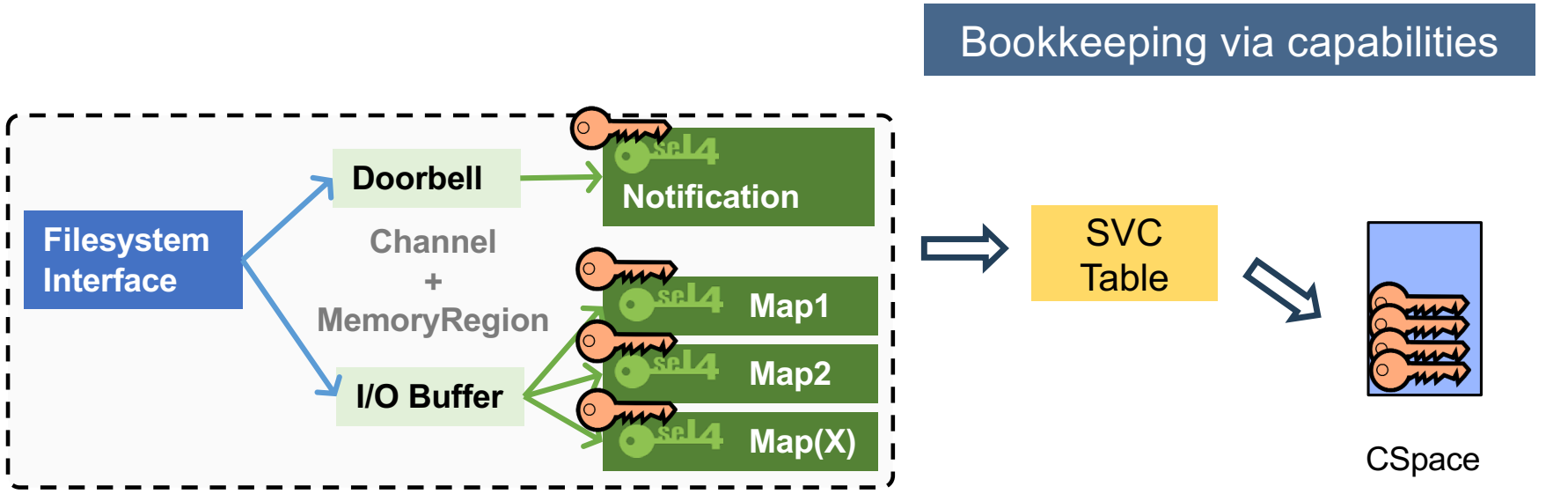
- Provision (ceiling of) connections at boot-time
- Establish authorised connections dynamically



Enables dynamic resource access in static architecture

Resource Bookkeeping

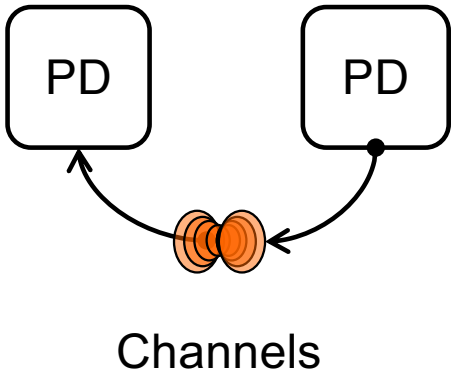
Abstract resources with Microkit abstractions Channel + MemoryRegion



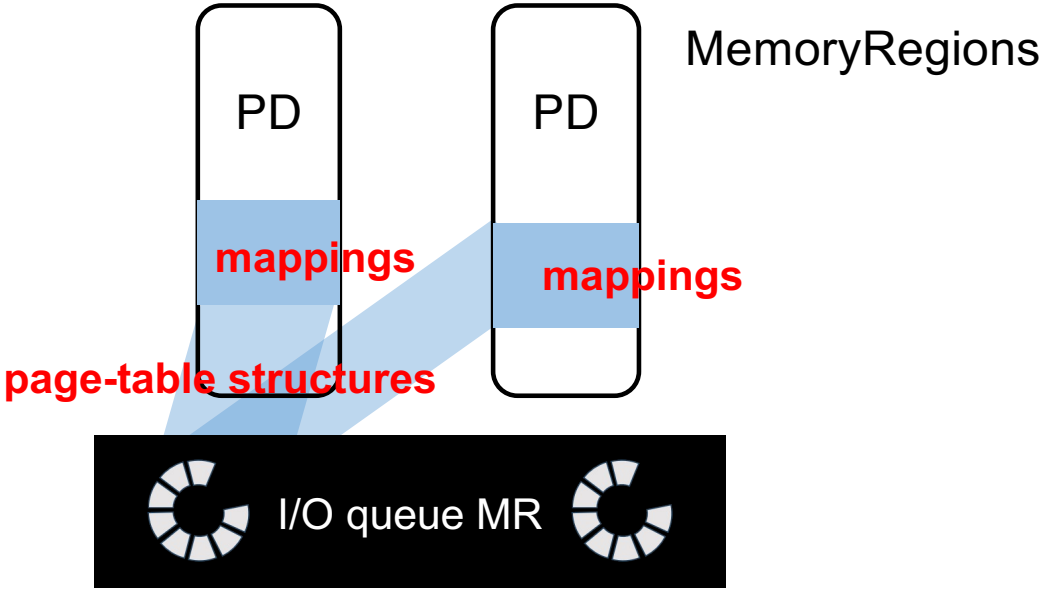
Use sdfgen/Acacia to generate SVC (services) table

Resource Provisioning and Control

- **Pre-provision** – channel capability
- **Control** – CNode_Copy/Delete



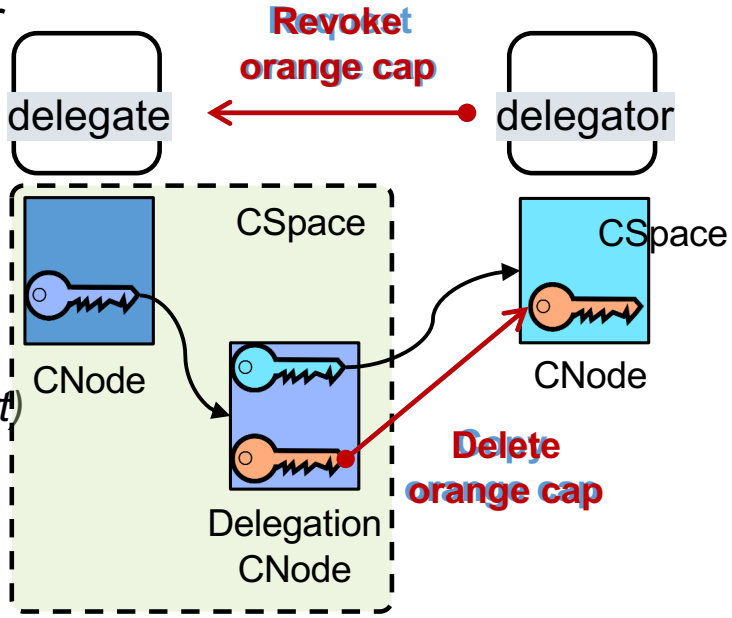
- **Pre-provision** – frame capabilities with page-table structures
- **Control** – Page_Map/Unmap



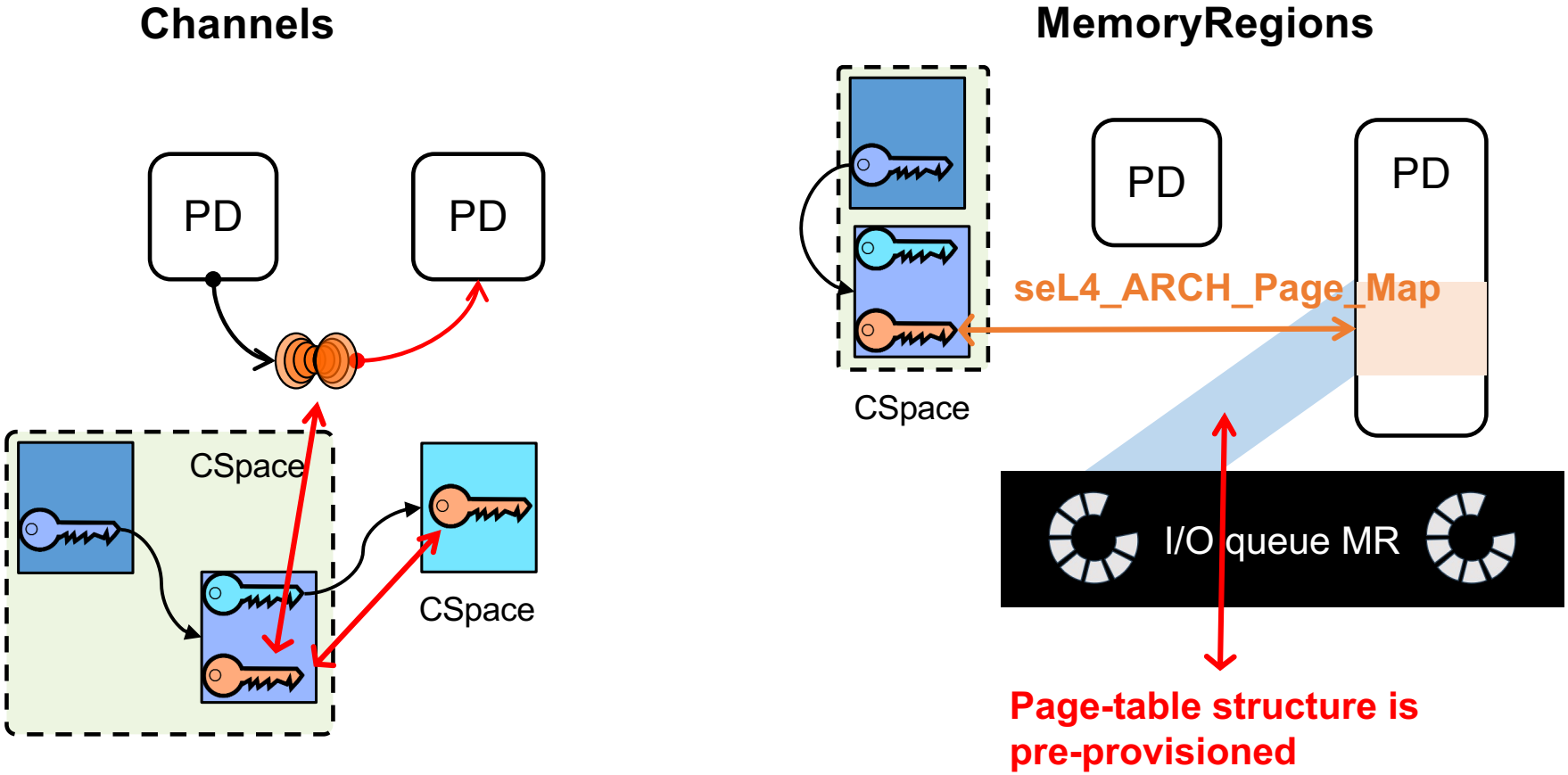
Change the resource view of a PD

- **Capability Delegation**
  - PDs can declare rights of use of capabilities as delegated
  - The delegated caps will be given to a trusted resource controller PD
  - This forms a delegation(delegate, delegator) pair
- **Implementation**
  - Microkit tool create a CNode for delegation
  - Delegation CNode contains caps of delegator
    - *VSpace*
    - *CSpace*
    - *All delegated resources (MRs, channels, x86 IOport)*
  - Delegation CNode is accessible to the delegate
  - Each delegation pair has one delegation CNode
- No runtime resource creation/reclaim allowed!!

Allows resources changes of PDs

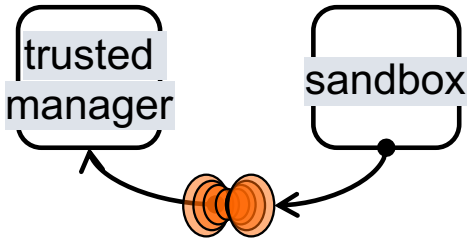


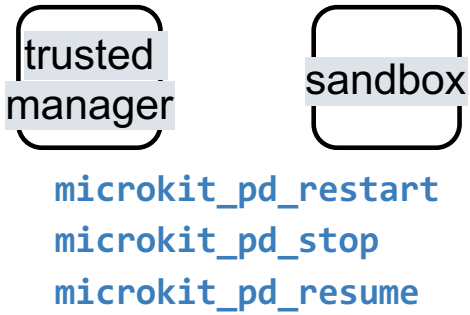
Resource Control via Cap Delegation

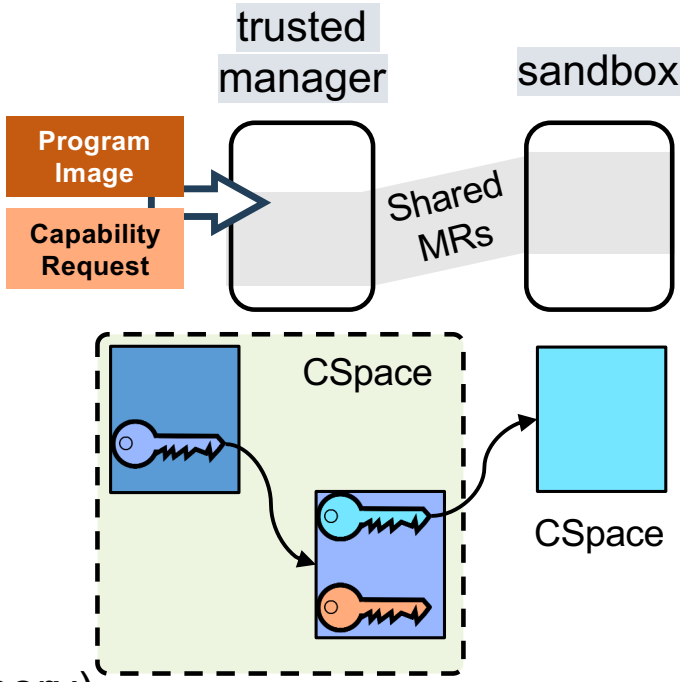


Sandbox With Dynamic Features

Trusted manager is delegate of template PD

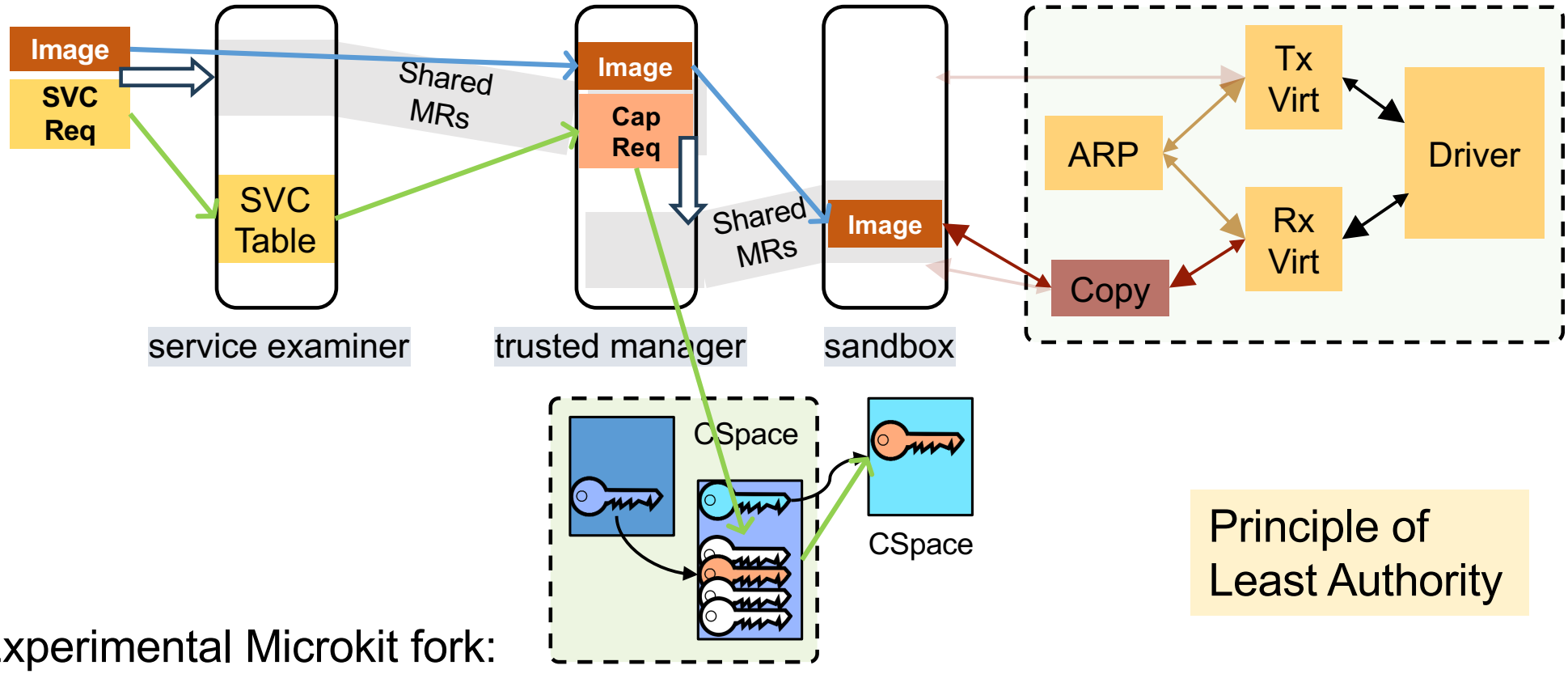






- The trusted manager
  - Serves sandboxes as a server/fault handler
  - Controls the life-cycle changes of sandboxes
  - Dynamically loads program image (via shared memory)
  - Performs delegated-capability bookkeeping

Example: Using Network Rx Virtualiser



Principle of Least Authority

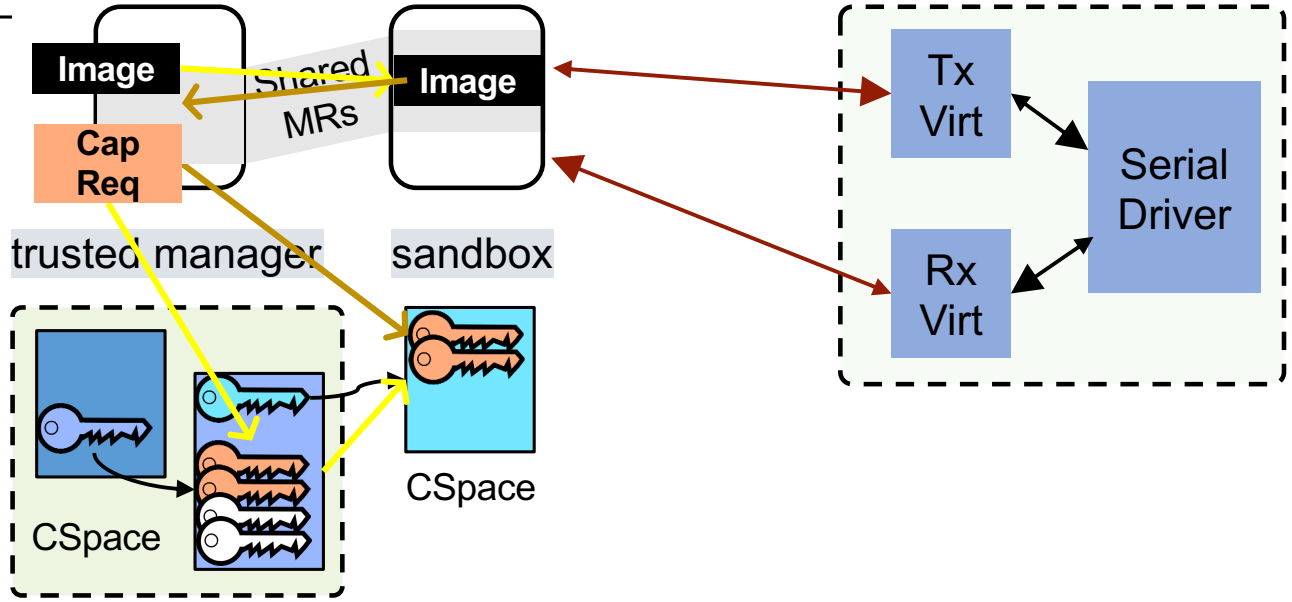
Experimental Microkit fork:
https://github.com/au-ts/dynamic-microkit

Preliminary Results: Startup Latency

ISA	Platform	Startup-teardown Latency
ARM64	Cortex-a55 at 1.2Ghz	284,400 cycles (237 µs)
x86_64	Xeon® W-1250 at 3.3Ghz	139,500 cycles (42 µs)

Payload setup

- Resource view
- (serial connection)
  - 34 frames to map
  - 1 channel to grant
- Image size
  - 156 KB in total
  - 16 KB to load

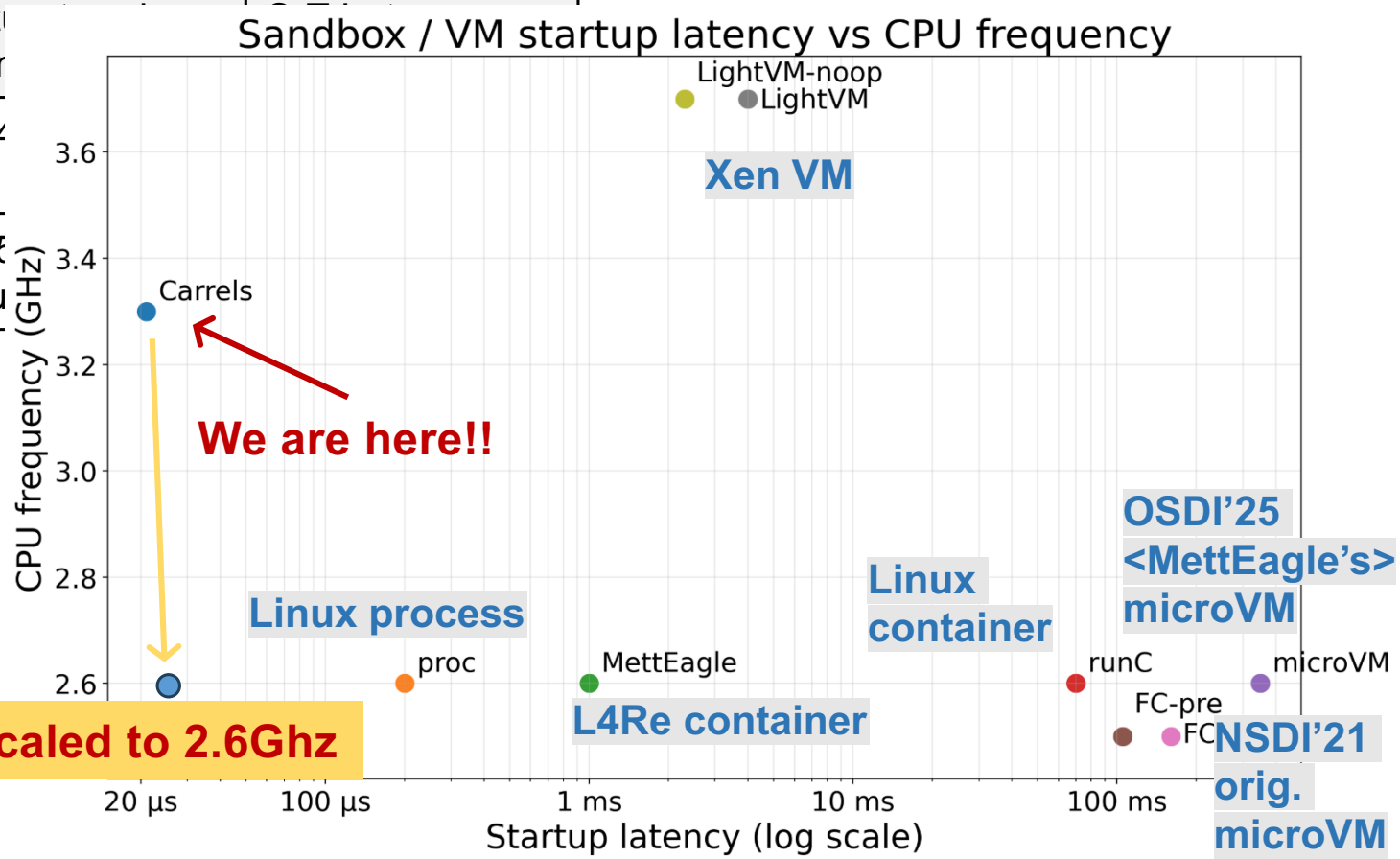


Preliminary Results: Tear-Down Latency

ISA	Platform	Start Later
ARM64	Cortex-a55 at 1.2Ghz	284,4 (237
x86_64	Xeon® W-1250 at 3.3Ghz	139,5 (42 µ

Payload setup

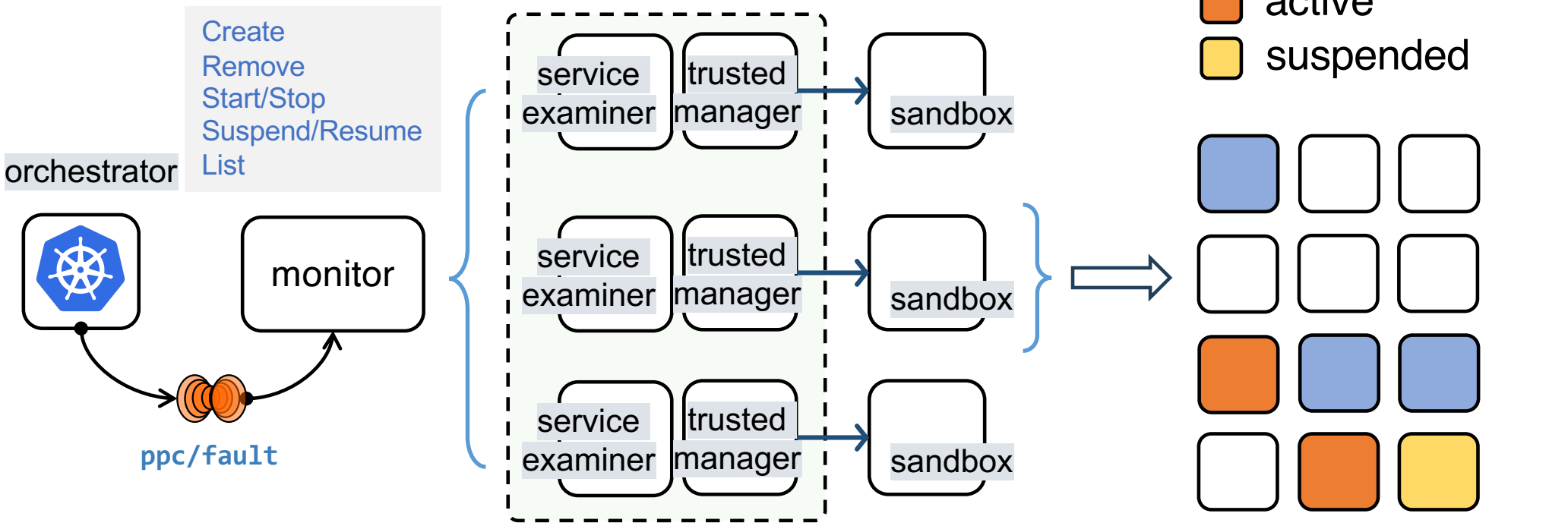
- Resource view
- (serial connection)
  - 34 frames to map
  - 1 channel to grant
- Image size
  - 156 KB in total
  - 16 KB to load



1. Microservice and seL4
2. Microservice Deployment and Life-cycle
3. **Orchestration**
4. Programming API

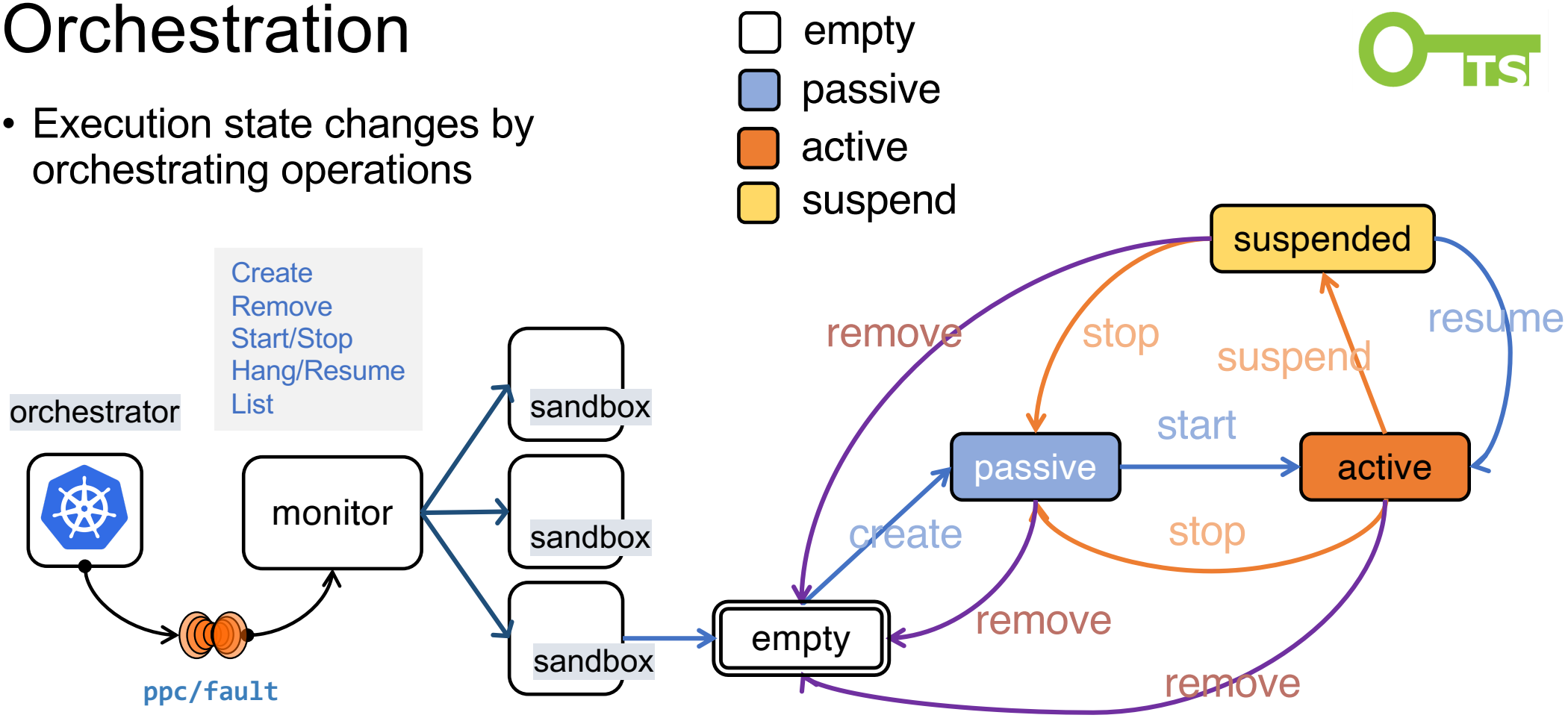
Orchestration

- Life-cycle state management
- Orchestrator

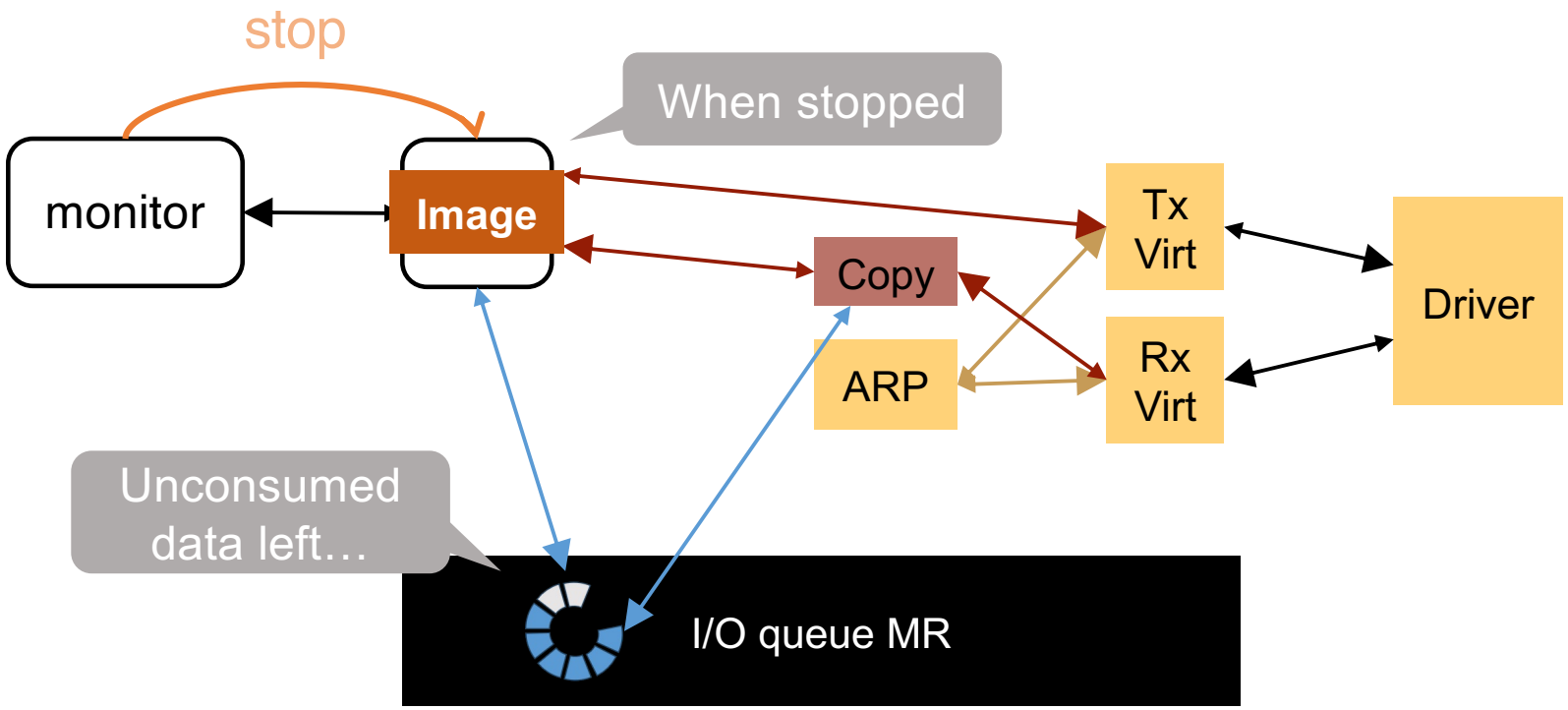


Orchestration

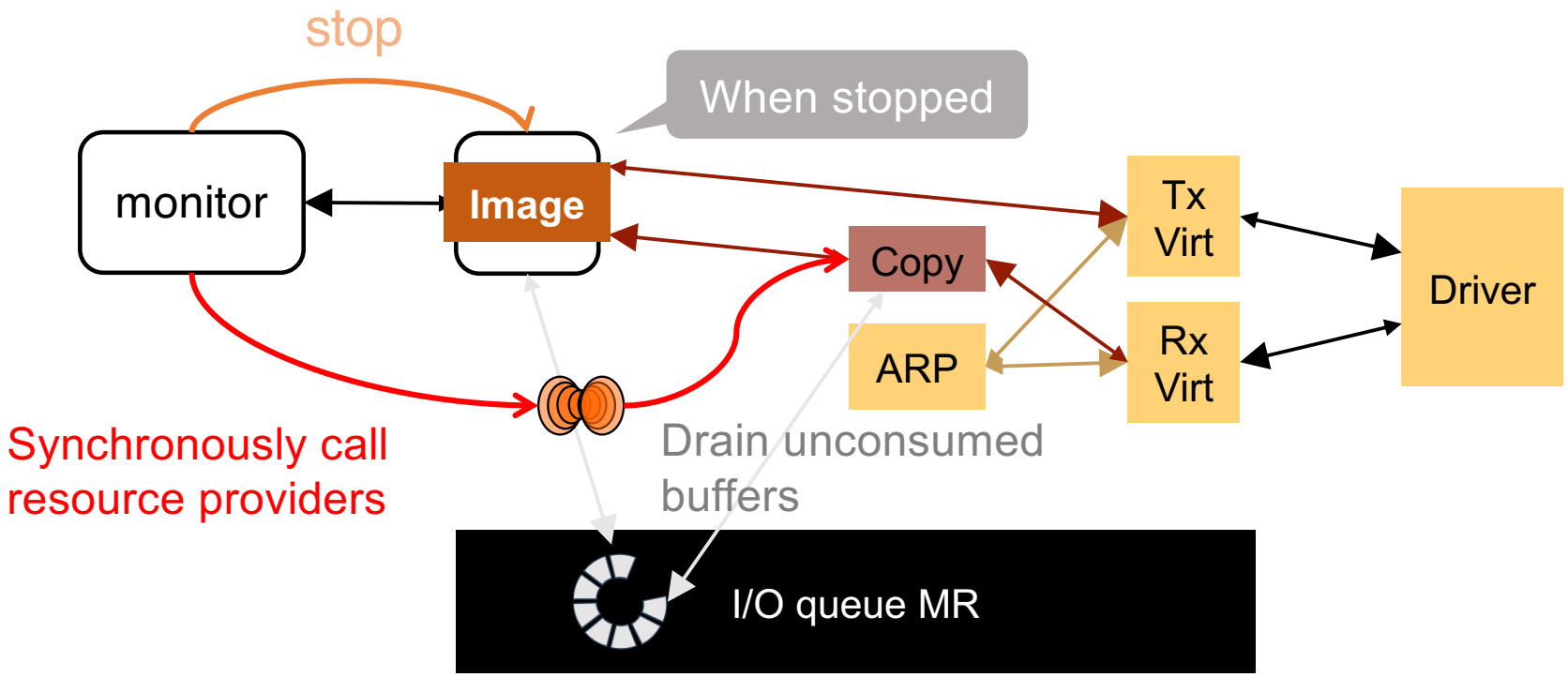
- Execution state changes by orchestrating operations



State management



State management: Plan



1. Microservice and seL4
2. Microservice Deployment and Life-cycle
3. Orchestration
4. **Programming API**

Programming API for microservices

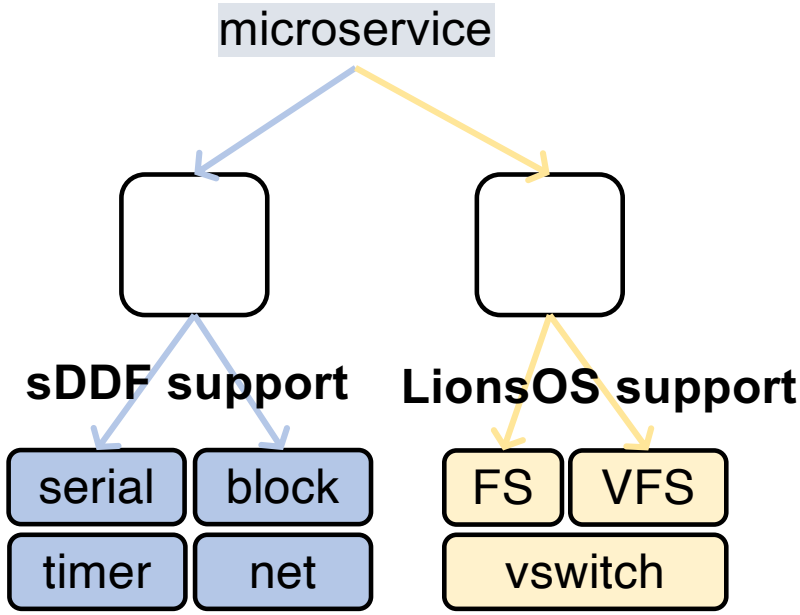
- **Native API**

  - Flexible for building different OS personalities
  - Not familiar to developers

libmicrokit

```c
void init(void)
{
    microkit_dbg_puts("hello, world\n");
}

void notified(microkit_channel ch)
{
}
```



Unikraft

RESEARCH-ARTICLE | OPEN ACCESS |

Unikraft: fast, specialized unikernels the easy way

Authors: Simon Kuenzer, Vlad-Andrei Bădoiu, Hugo Lefeuvre, Sharan Santhanam, Alexander Jung, Gaulthier Gain, Cyril Soldani, Costin Lupu, Ştefan Teodorescu, Costi Răducanu, + 5 Authors

Info & Claims

EuroSys '21: Proceedings of the Sixteenth European Conference on Computer Systems • Pages 376 - 394
https://doi.org/10.1145/3447786.3456248

Applications	NGINX, SQLite, Redis, memcached, Click modular router, lighttpd (ongoing).
Frameworks	Intel DPDK, TensorFlow Lite, PyTorch.
Compiled Languages	C/C++, Go, Web Assembly (WAMR), Lua, Java/OpenJDK (ongoing), Rust (ongoing)
Interpreted Languages	Python, Micropython, Ruby, JavaScript/v8 (ongoing).

Table 3. Applications, frameworks and languages currently supported by Unikraft.

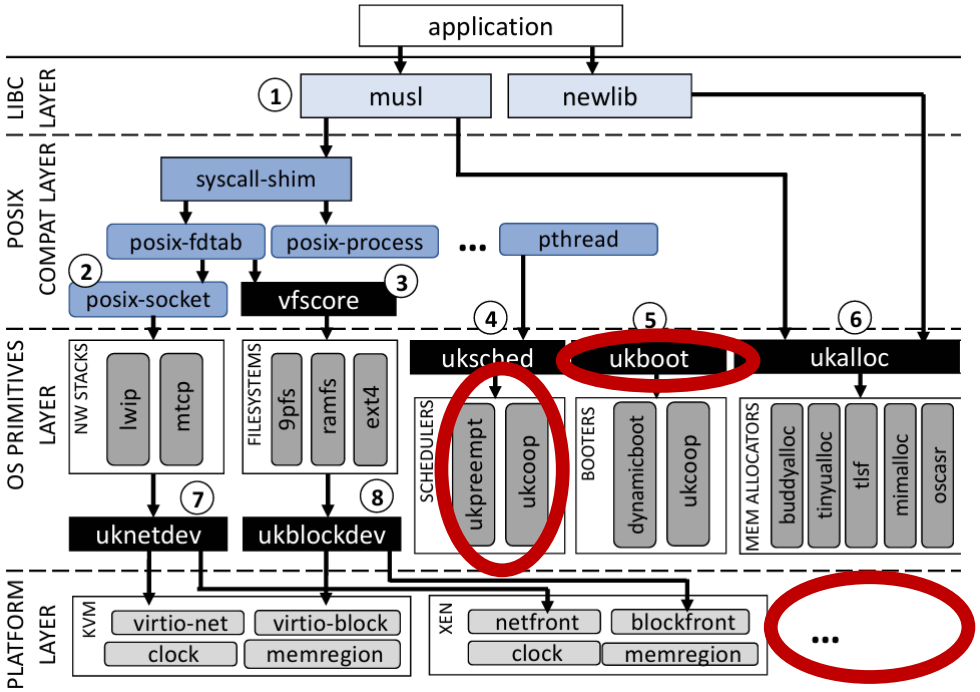
Experimental Carrels support for Unikraft

- lib/uksched, lib/ukschedcoop: +33 LoC
- lib/ukpcpuvar/arch/arm64: +38,-9 LoC

Patch needed

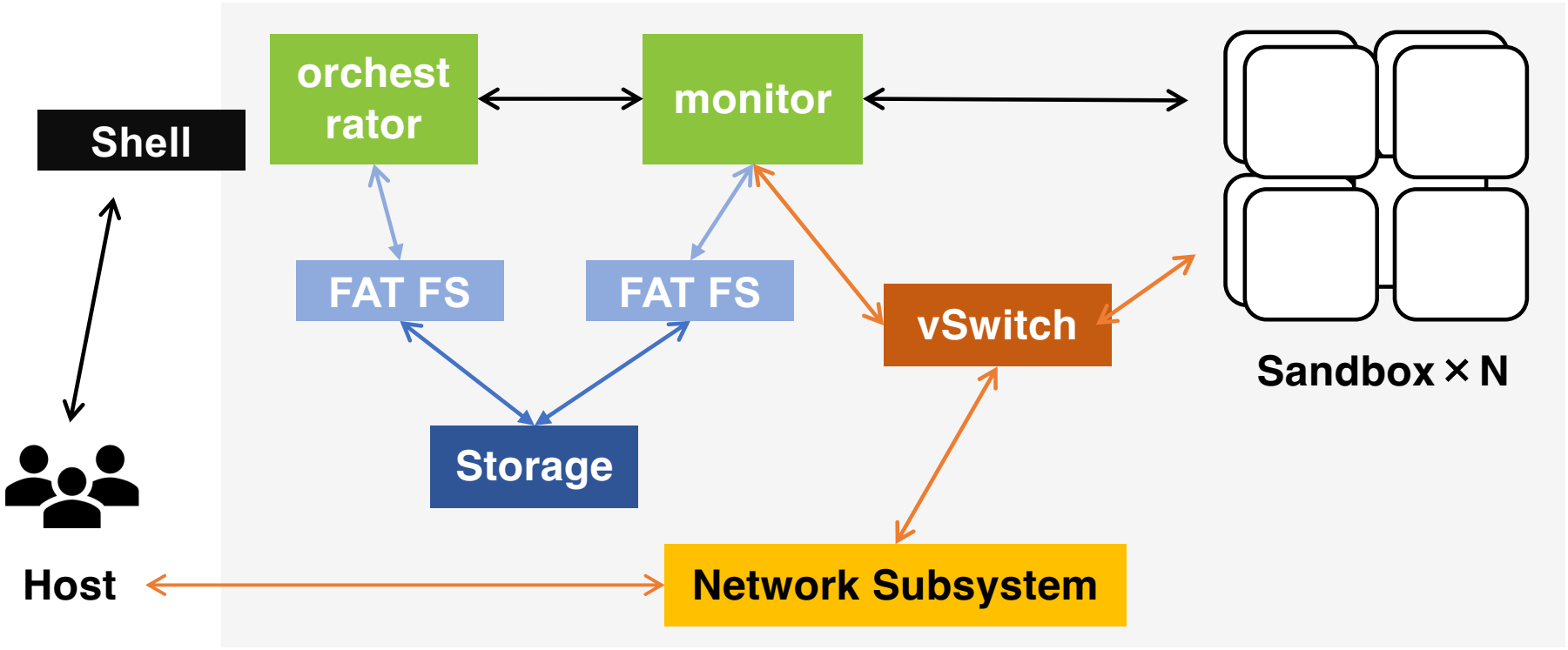
- plat/carrels: +1,101 LoC
- drivers/sddf: +530 LoC
  - network
  - timer
  - serial

New



plat/carrels

Demo



Questions?

Thanks for listening!