Memory-Efficient WebAssembly Containers

Matthijs Jansen, Maciej Kozub
Alexandru Iosup, Daniele Bonetta

m.s.jansen@vu.nl

https://atlarge-research.com/mjansen/

@Large Research
Massivizing Computer Systems

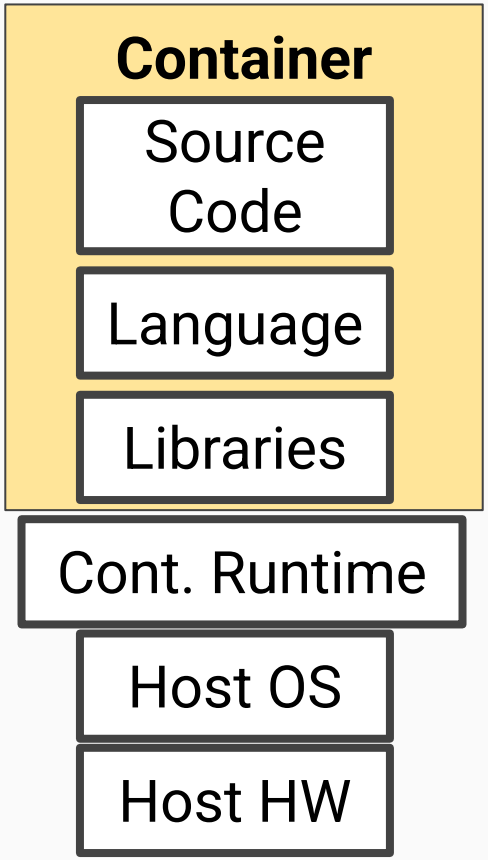
VU VRIJE UNIVERSITEIT AMSTERDAM

Open-source Code

Containers

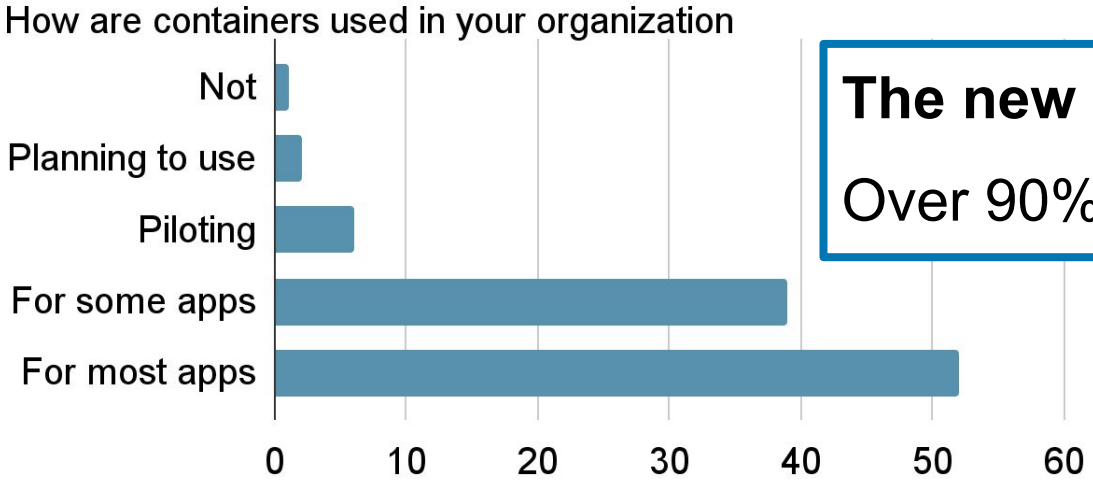
Containers are an isolation mechanism

- **Portable**, **maintainable**
- Easy to **scale** and operate
- **Isolation** for performance, security



CNCF 2024 Annual Survey

VU VRIJE UNIVERSITEIT AMSTERDAM

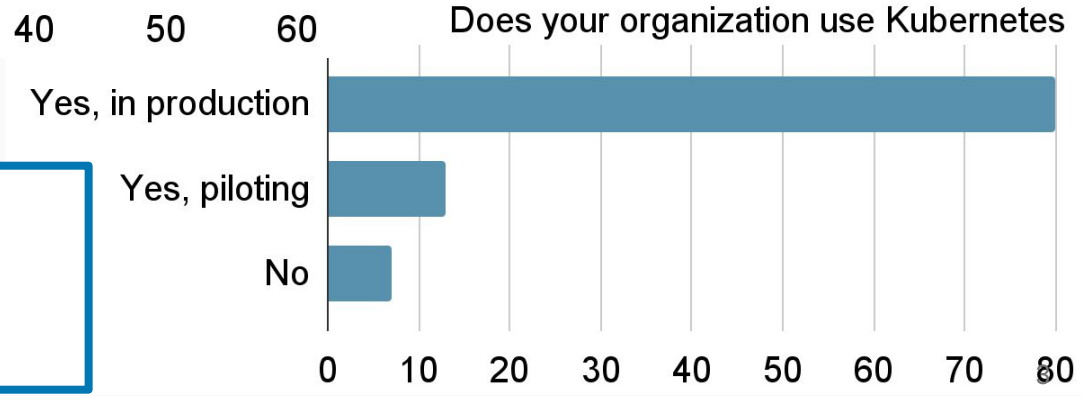


The new normal

Over 90% uses containers

The production standard

80% uses Kubernetes



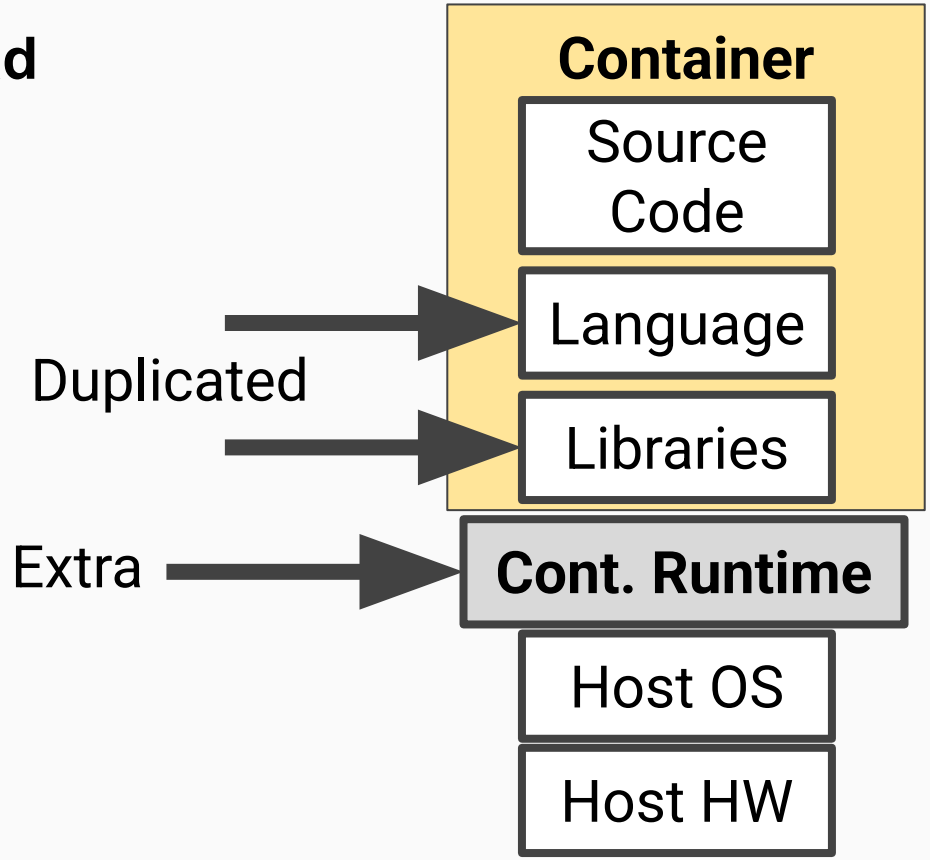
Container Downsides

But: Containers add overhead

- CPU
- Memory
- Storage
- Network

Result:

- **Slower code**
- **Increased cost**
- **Increased energy use**

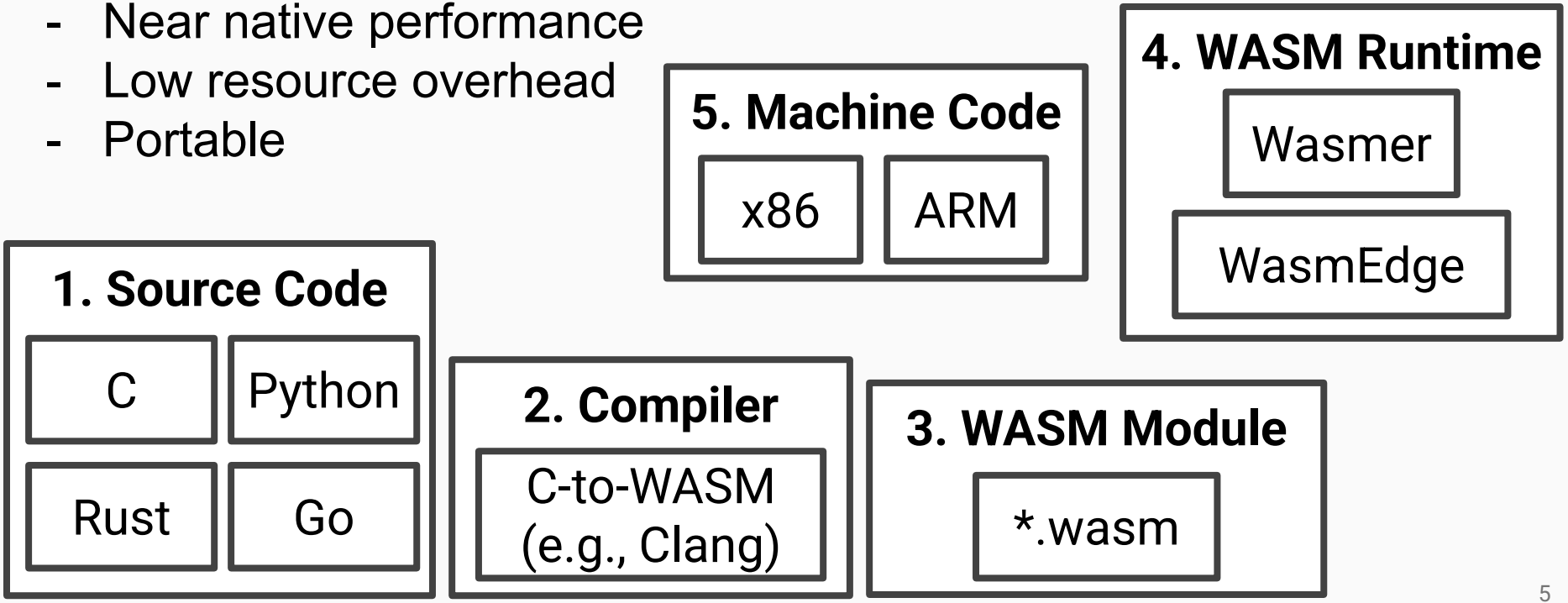


Efficiency with WebAssembly

VU VRIJE UNIVERSITEIT AMSTERDAM

WASM: Stack-based VM

- Near native performance
- Low resource overhead
- Portable



Containerize WebAssembly

VU VRIJE UNIVERSITEIT AMSTERDAM

WebAssembly modules can be packaged and distributed as containers

- OCI compliant
- No base image

```
# Dockerfile
FROM scratch
COPY main.wasm /main.wasm
ENTRYPOINT [ "/main.wasm" ]
```

Digest	OS/ARCH	Last pull	Compressed Size ⓘ
fd610b24a34f	linux/amd64	a month ago	45.48 MB
84bad8b4bd4b	wasi/wasm	18 days ago	451.38 KB

WASM containers can run side-by-side with non-WASM containers

WebAssembly on Kubernetes

VU VRIJE UNIVERSITEIT AMSTERDAM

WASM containers have less overhead than non-WASM containers

WASM containers are compatible with Kubernetes (OCI)

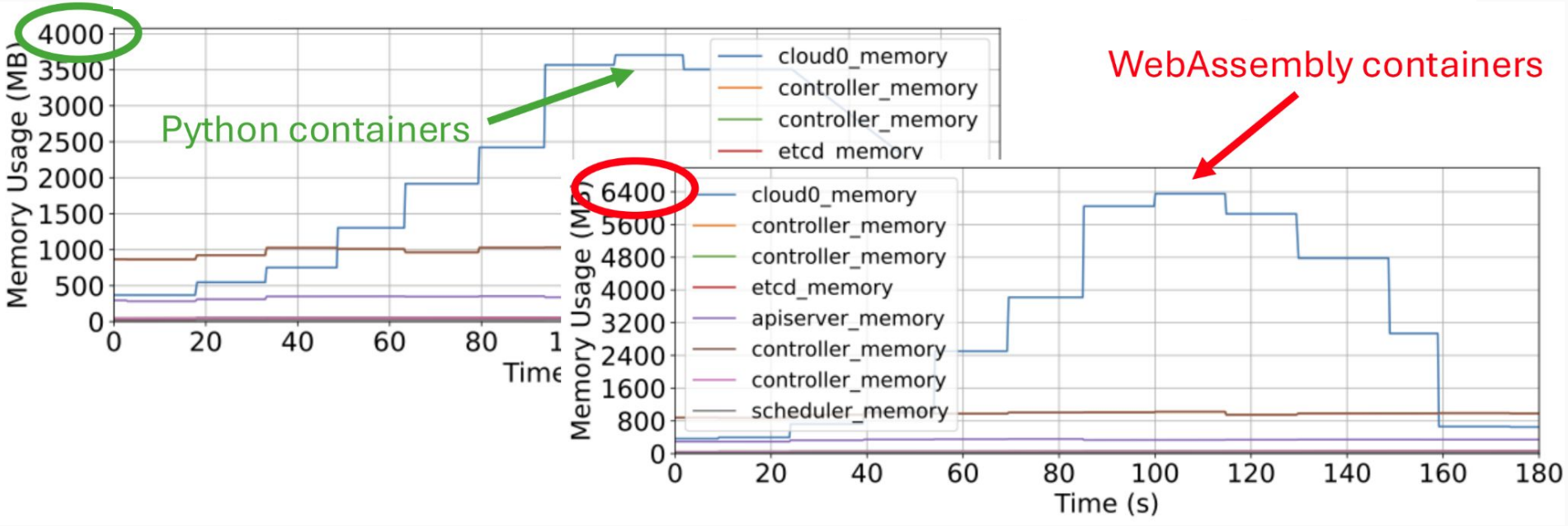
WASM containers should be the better choice for Kubernetes

Right?

WASM Inefficiency

VU VRIJE UNIVERSITEIT AMSTERDAM

Current WASM overhead exceeds non-WASM container overhead!



Research Objective

VU VRIJE UNIVERSITEIT AMSTERDAM

Improve the memory footprint of WASM containers
(1) compared to existing WASM runtimes
(2) compared to non-WASM containers

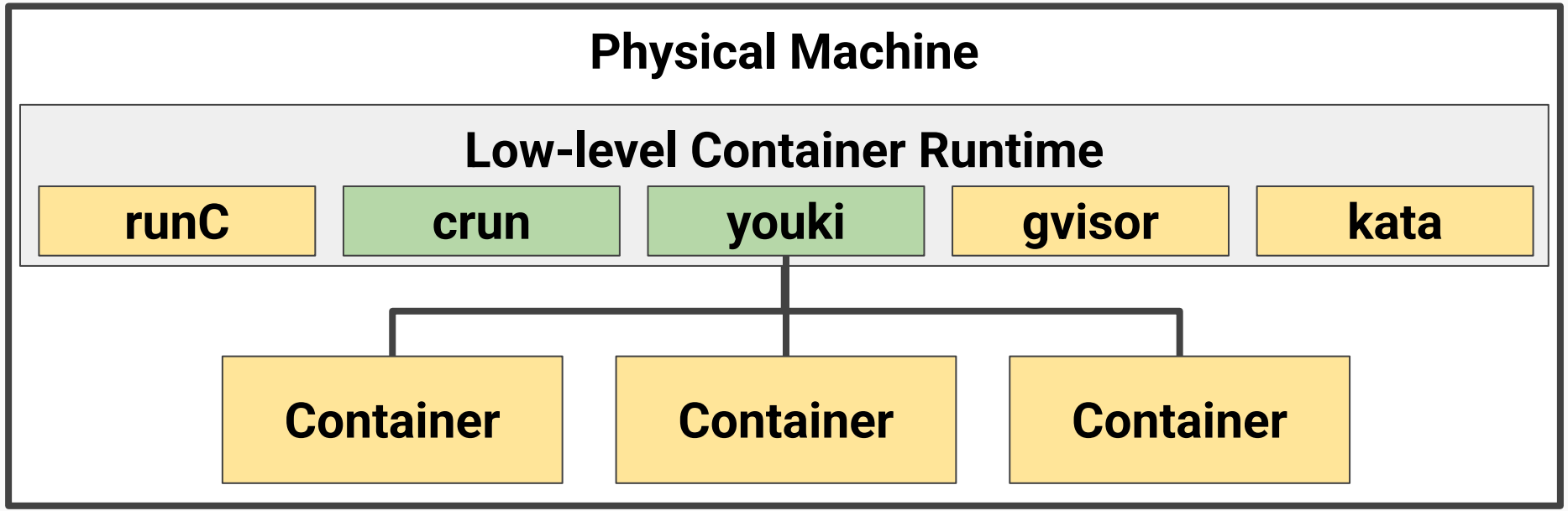
(1) What is the (WASM) container landscape?

(2) How to create a new WASM runtime integration

(3) Evaluation

Deploying Containers - Low Level

Low-level: Create, start, stop, delete container with system calls

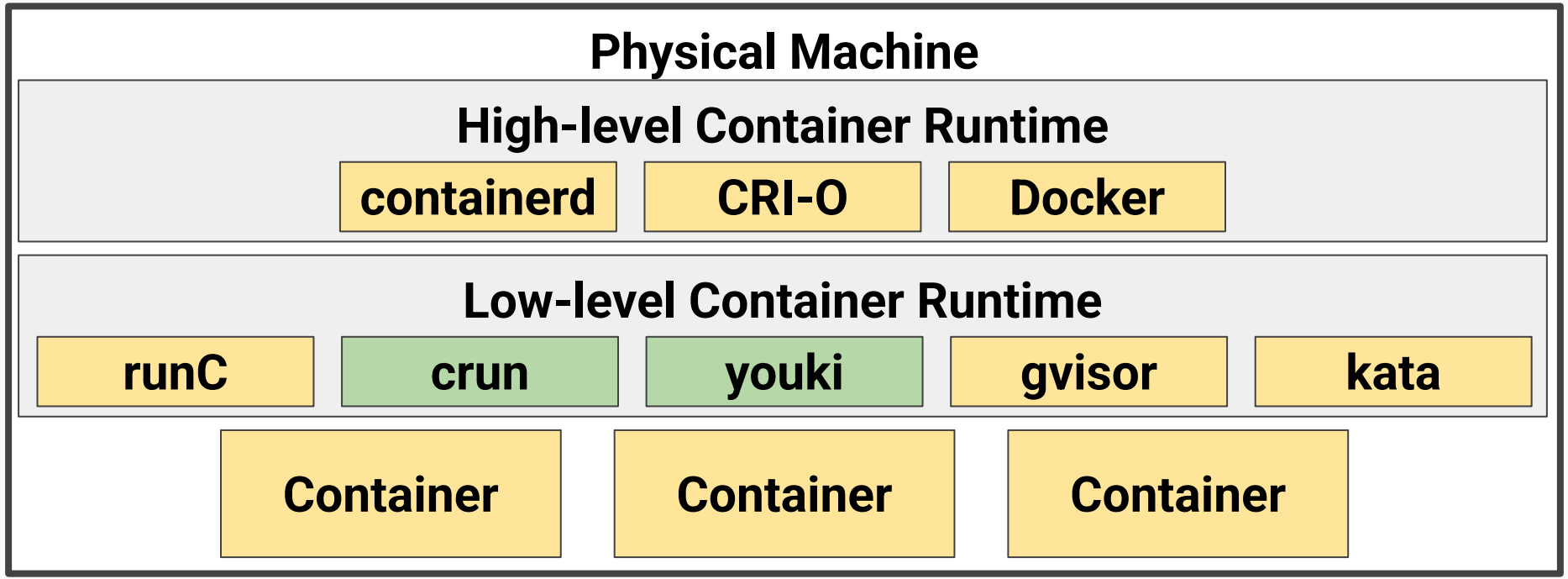


In green: Only runtimes that currently support WASM

Deploying Containers - High Level

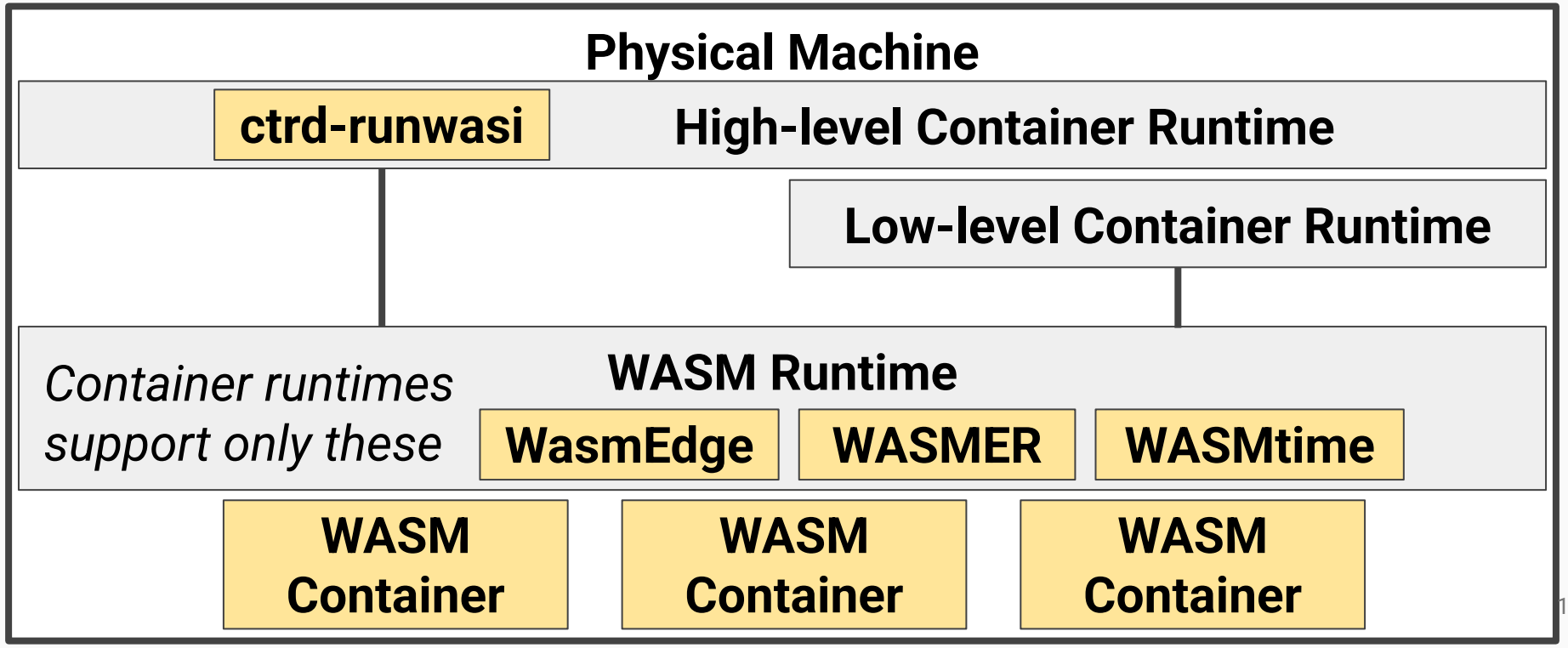
VU VRIJE UNIVERSITEIT AMSTERDAM

High-level: Manage images, networking, volume mounting, logging

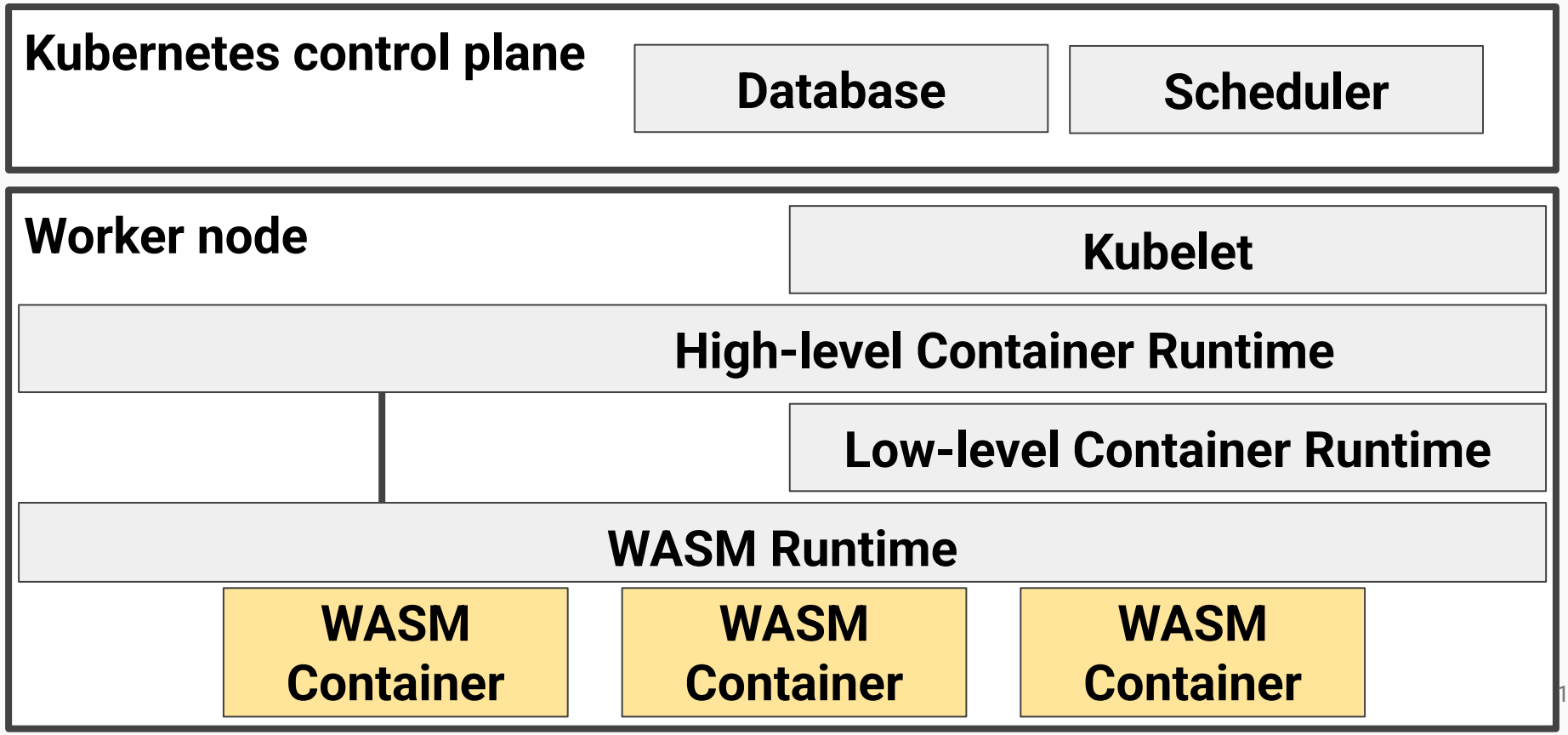


Deploying Containers - WASM

WASM: Different runtimes to manage WASM



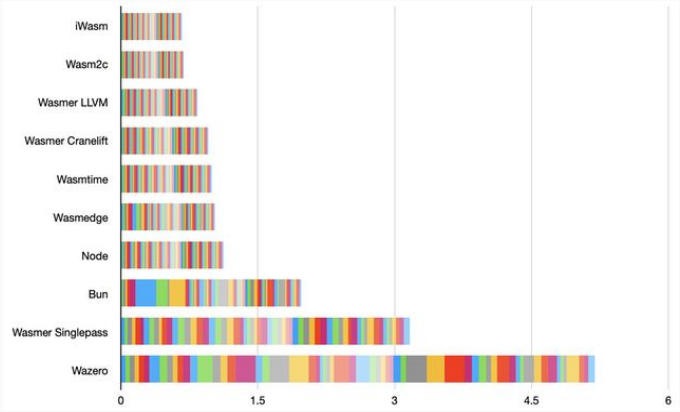
Deploying Containers - Kubernetes



Improving WASM integration

The WASM container runtime stack:

1. **High-level** container runtime
2. **Low-level** container runtime
3. **WebAssembly** runtime
   - Currently, only *WasmEdge, WASMER, WASMtime* supported by container runtimes
   - Existing benchmarks show that **WAMR** has better performance and lower memory footprint



Improving WASM integration

VU VRIJE UNIVERSITEIT AMSTERDAM

The WASM container runtime stack:

1. **High-level** container runtime
2. **Low-level** container runtime
   - crun and youki already support WASM
   - Or bypass lower-level with RunWASI?
3. **WebAssembly** runtime: **WAMR**

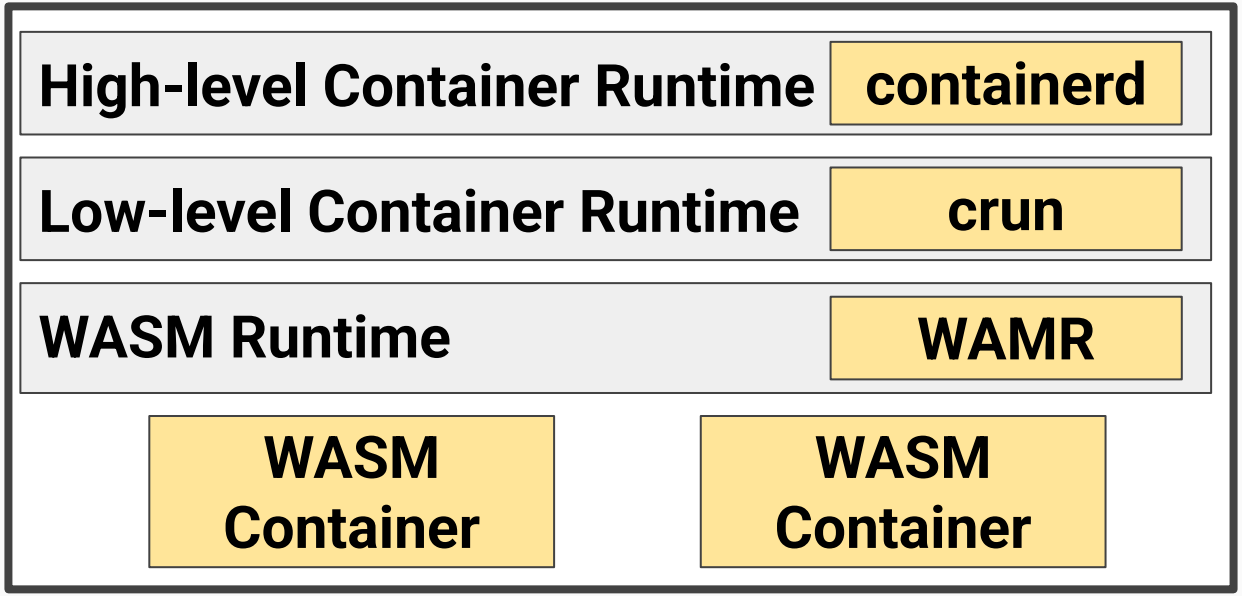
crun: Faster than youki

Runtime	Time (mean ± σ)	Range (min … max)	vs youki(mean)	Version
youki	111.5 ms ± 11.6 ms	84.0 ms ± 142.5 ms	100%	0.3.3
runc	224.6 ms ± 12.0 ms	190.5 ms ± 255.4 ms	200%	1.1.7
crun	47.3 ms ± 2.8 ms	42.4 ms ± 56.2 ms	42%	1.15

crun: Full container support (POSIX, syscalls), unlike RunWASI

Improving WASM integration

1. **High-level** container runtime
   - Need to support OCI for crun and CRI for Kubernetes
   - **containerd** is industry standard outside of OpenShift (CRI-O)

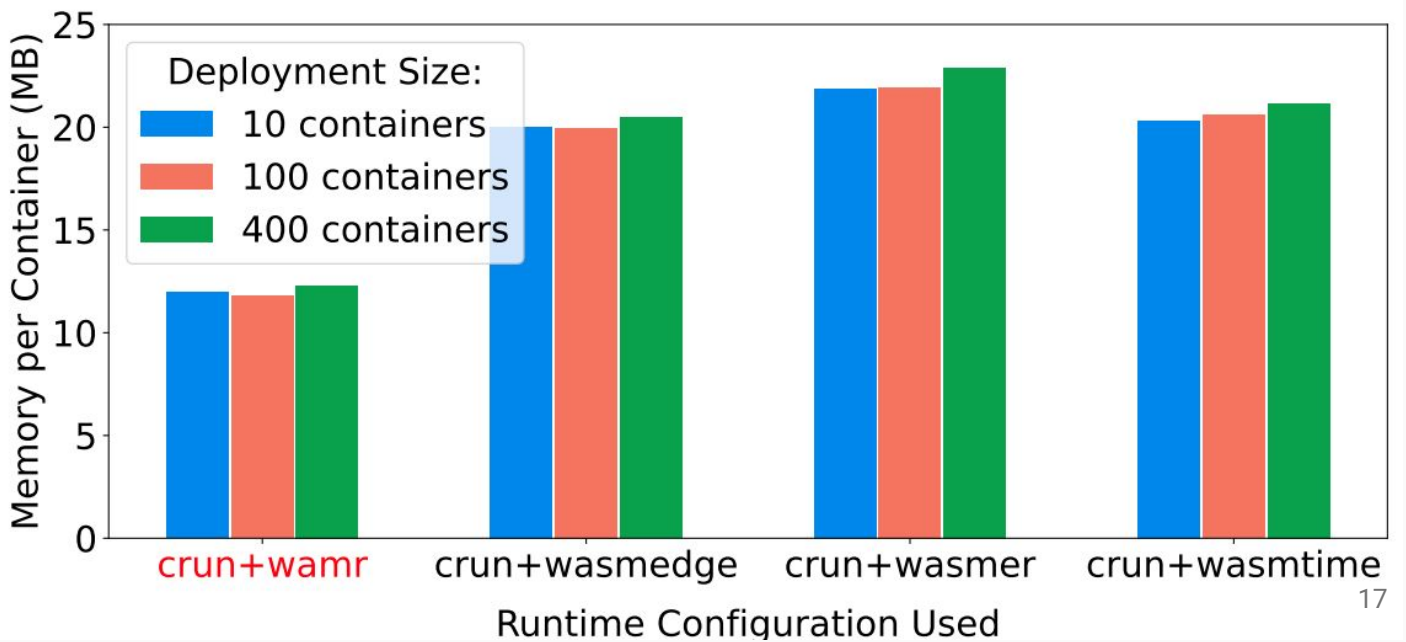


Evaluation: Memory Usage crun

- Deploy 10 / 100 / 400 WASM containers
- Average per-container memory overhead (OS)
- Empty Python application

At least 40.0% less memory!

Most efficient crun integration



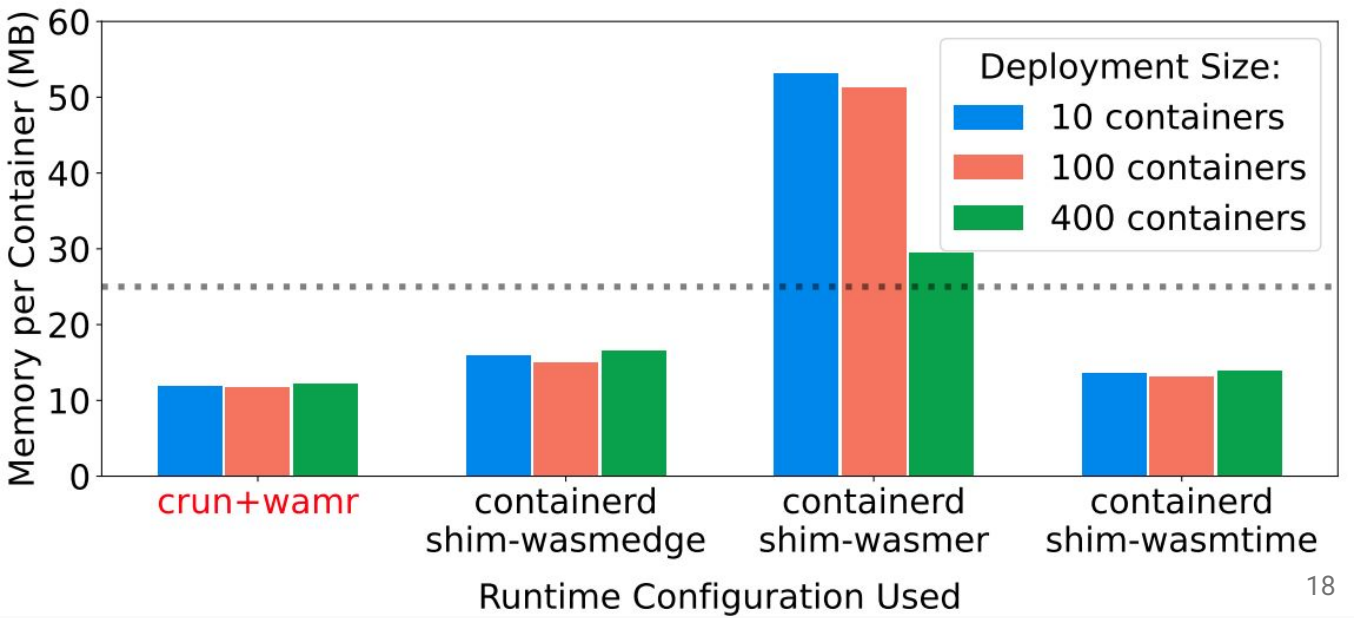
Evaluation: Memory Usage RunWASI

VU VRIJE UNIVERSITEIT AMSTERDAM

- Compared to containerd RunWASI
- RunWASI performs better than low-level runtime alternatives
- Later: RunWASI startup time does not scale

At least 10.9% less memory!

Most efficient WASM integration

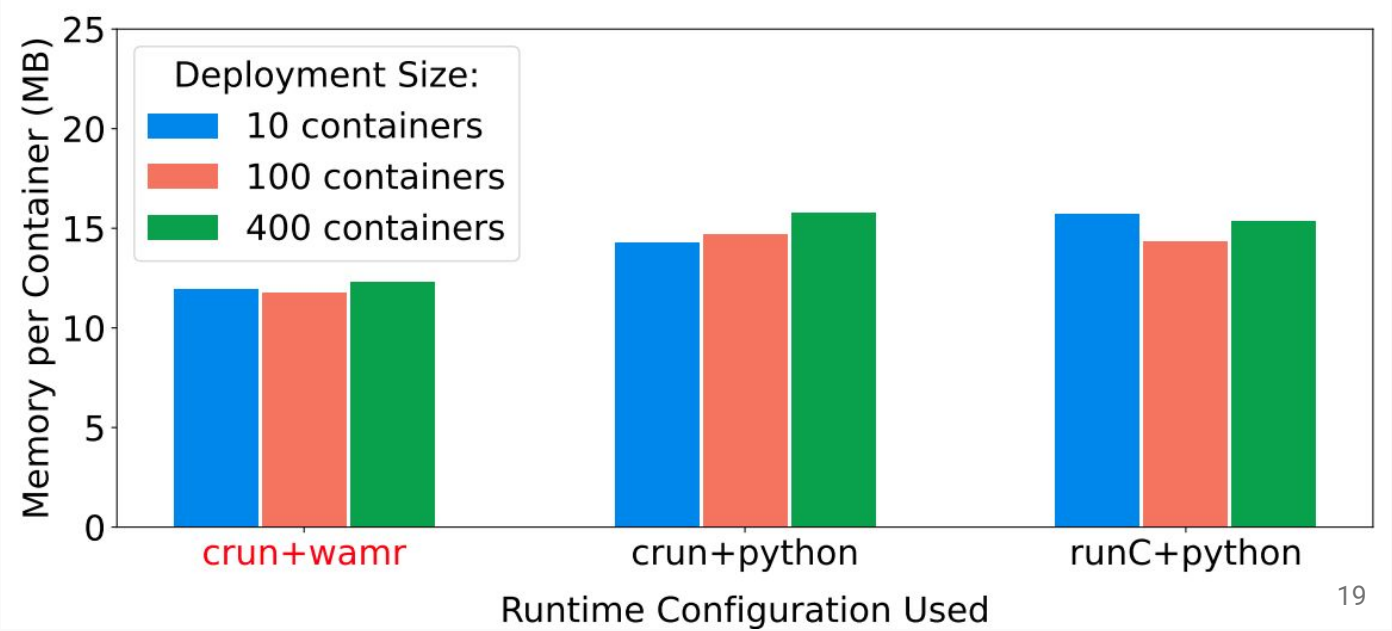


Evaluation: Memory Usage non-WASM

- Compared to Python Debian-slim container

At least 16.4% less memory!

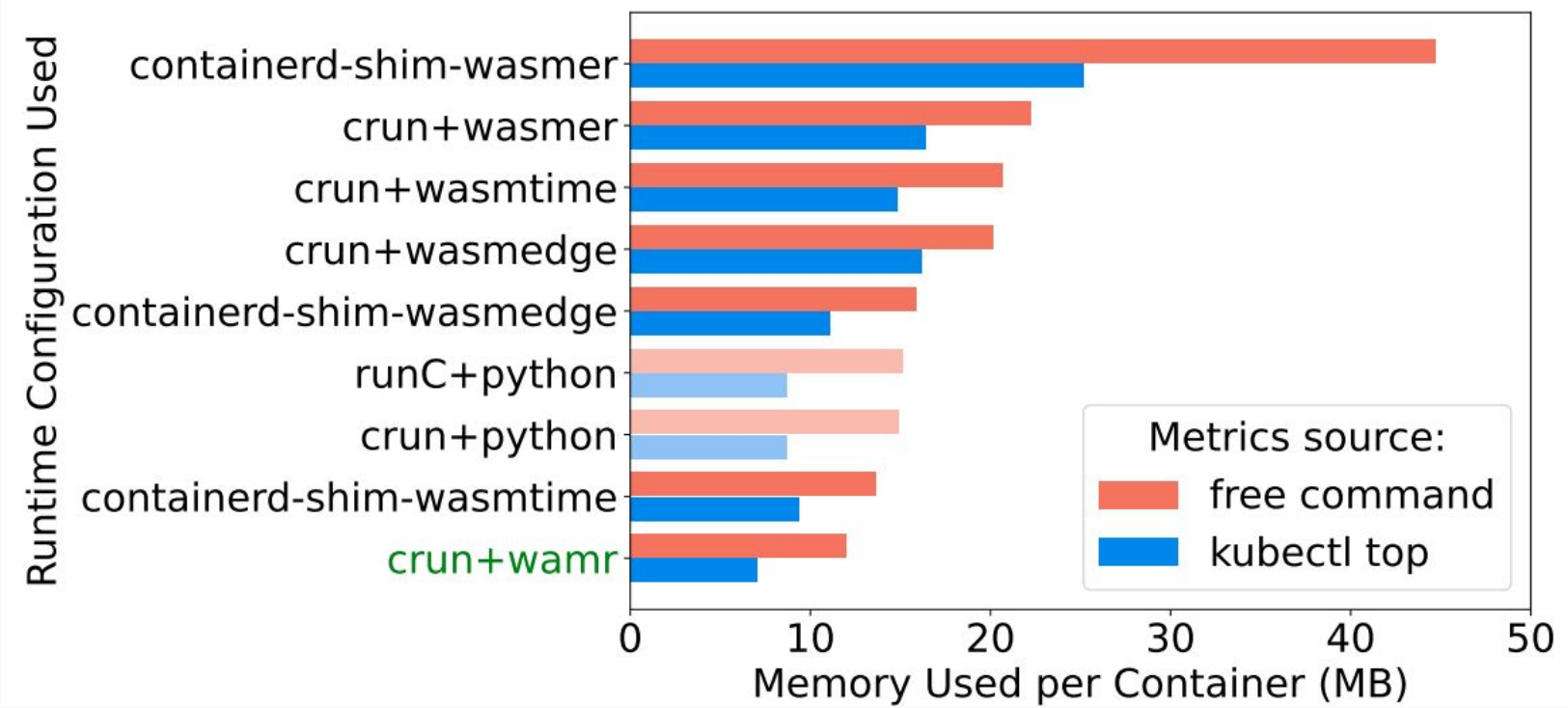
Outperforms non-WASM containers



Evaluation: Memory Usage Summary

VU VRIJE UNIVERSITEIT AMSTERDAM

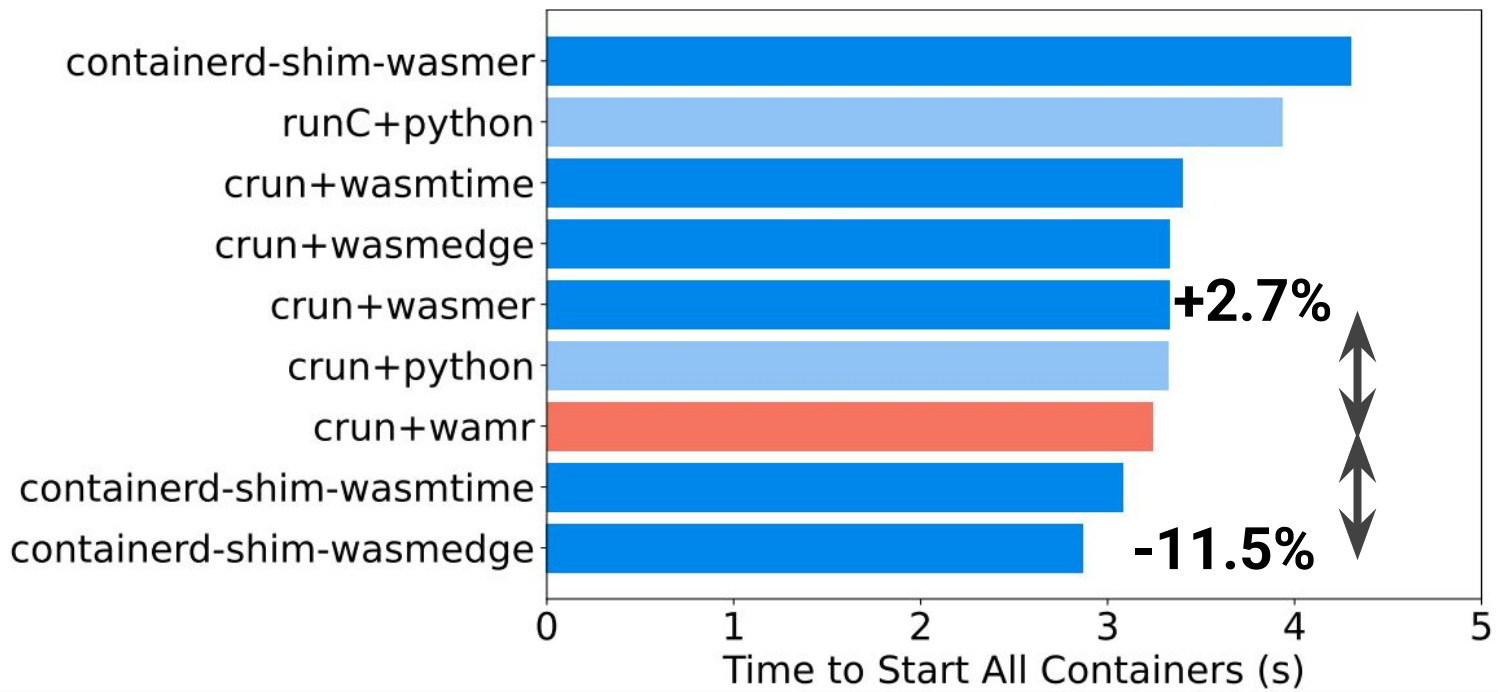
Our containerd + crun + wamr integration outperforms all



Evaluation: Start-up Time

VU VRIJE UNIVERSITEIT AMSTERDAM

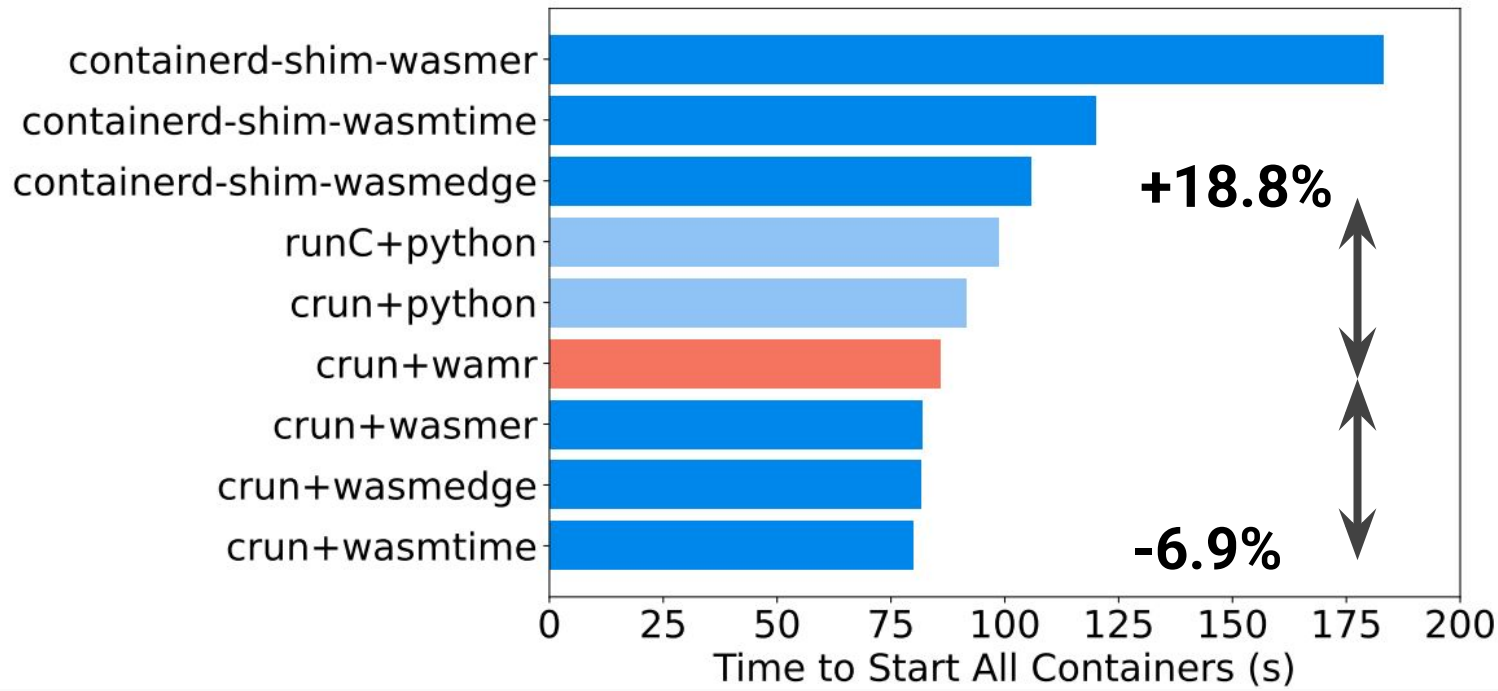
For 10 containers: Rank 3/9



Evaluation: Start-up Time

For 400 containers: Rank 4/9

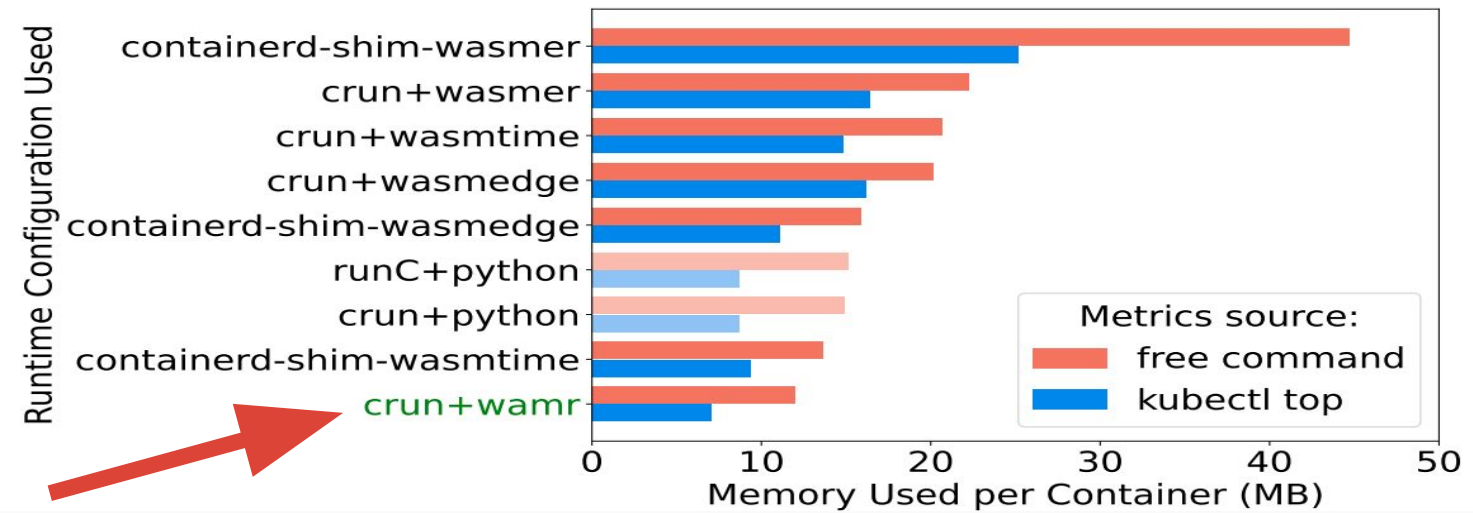
Mixed results compared to 10 → On average no performance loss



Conclusion

VU VRIJE UNIVERSITEIT AMSTERDAM

New WASM integration with lowest memory footprint
Comparable startup time to alternatives



We make WASM competitive with traditional containers