Unikernel Monitors

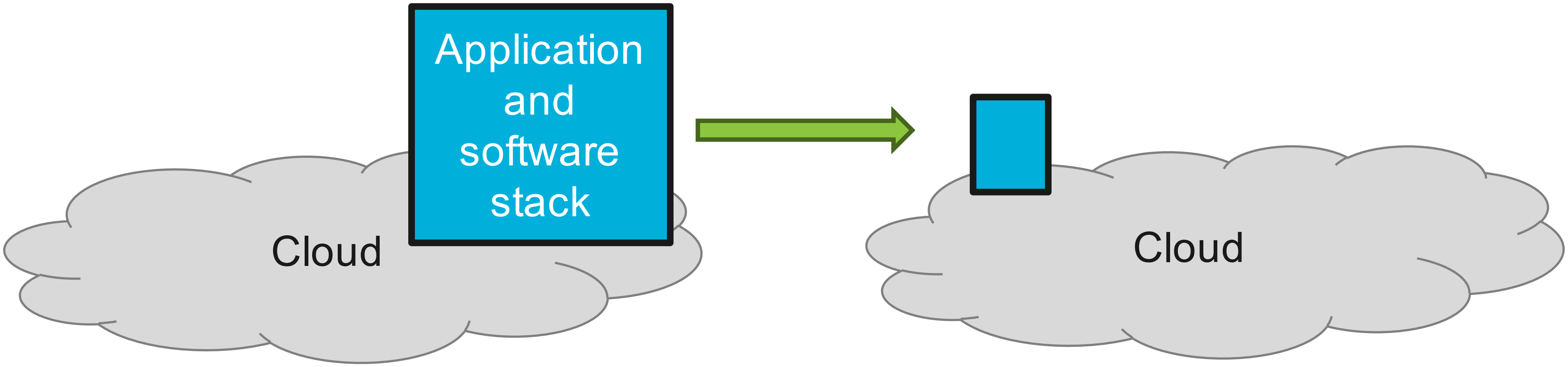
Extending Minimalism Outside of the Box

Dan Williams and Ricardo Koller, IBM Research

USENIX HotCloud 2016
June 20, 2016, Denver, CO

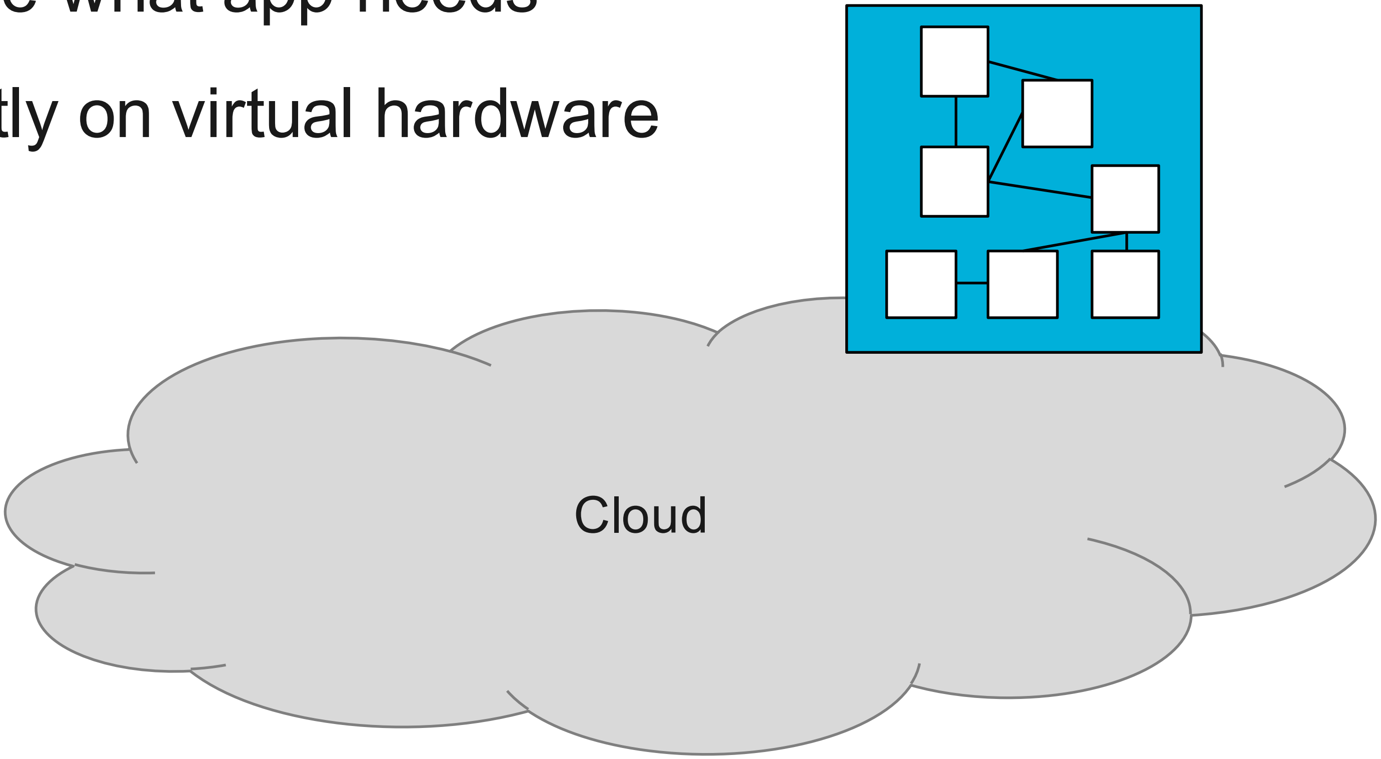
Minimalism is good

- Reduced attack surface
- Better understanding of the system
- Performance
- Management



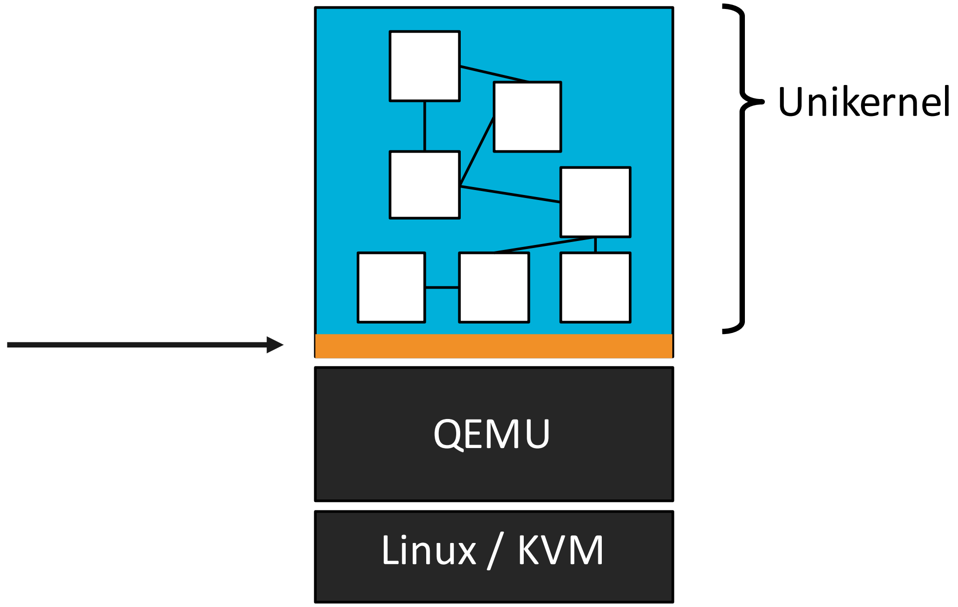
Unikernels: minimal systems?

- Built from fine-grained modules
- Only include what app needs
- Runs directly on virtual hardware



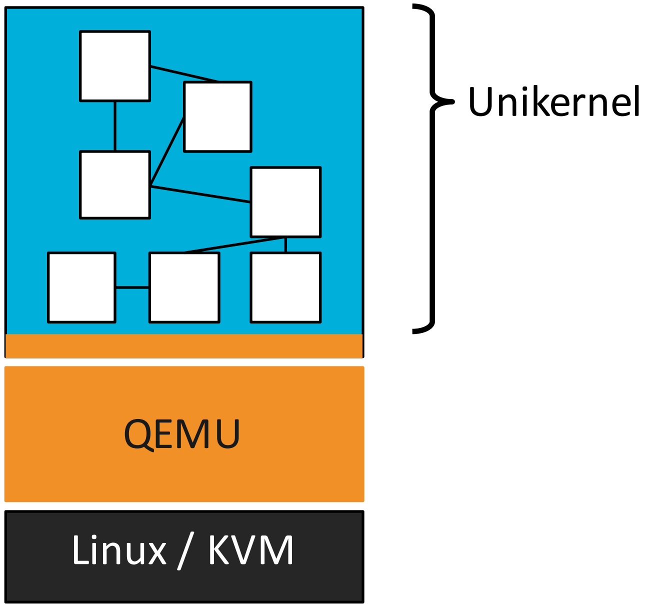
The extent of minimalism?

- Is the **interface** minimal?



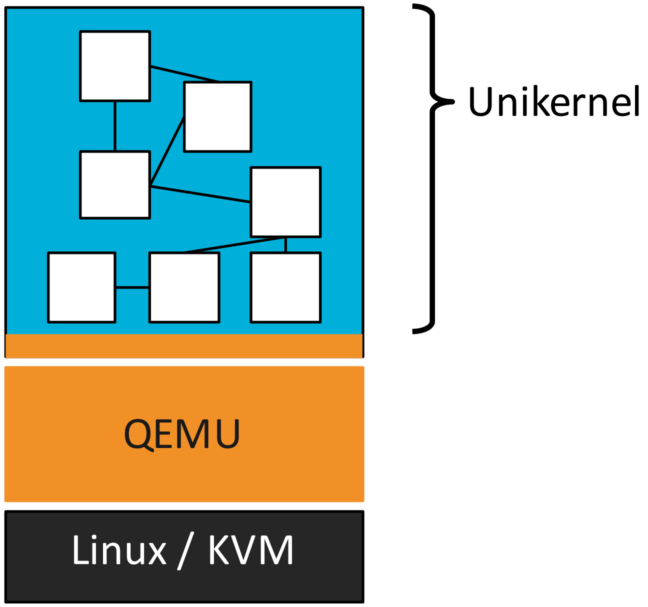
The extent of minimalism?

- Is the **interface** minimal?
- Is the **monitor** minimal?



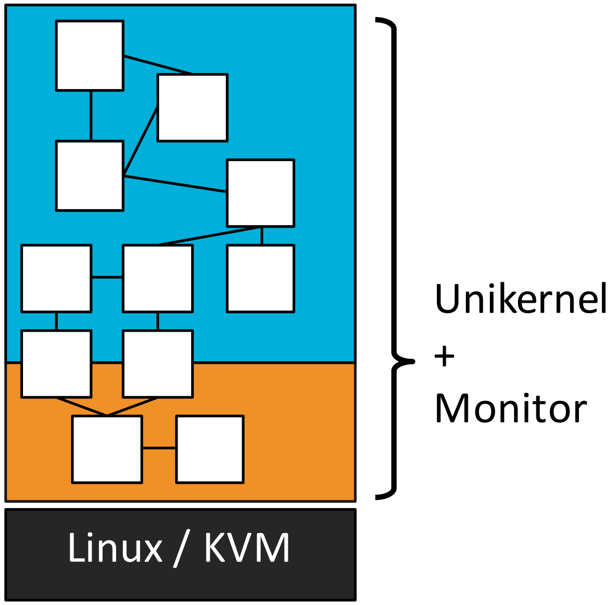
The extent of minimalism?

- Is the **interface** minimal?
- Is the **monitor** minimal?
- Can we use similar dependency-tracking techniques?



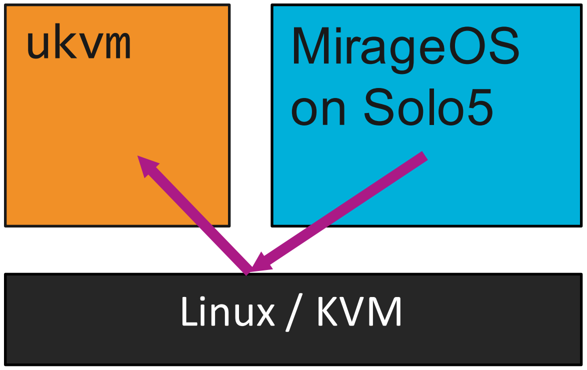
Unikernel monitors

- Executables contain both application and specialized **monitor**



Prototype monitor: `ukvm`

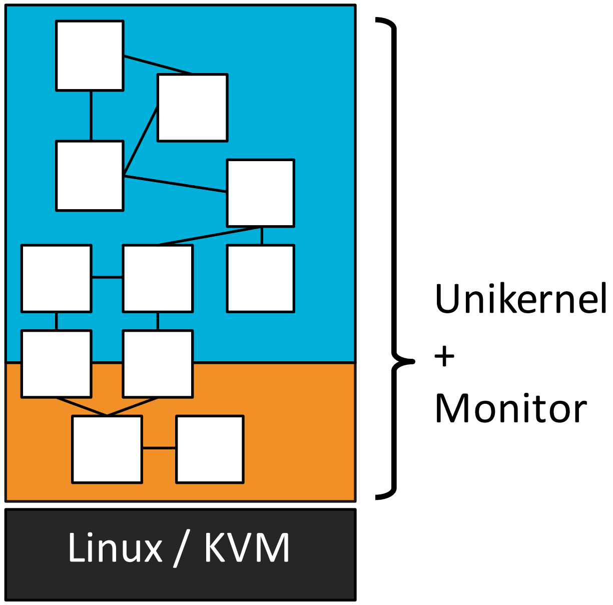
- Type-II hypervisor
  - Sets up memory, VCPU
- HW-support for virtualization
  - provides isolated processor context
- All exits routed to monitor



- Runs MirageOS unikernels on Solo5 unikernel base
- https://github.com/djwillia/solo5

Advantages of unikernel monitors

- Minimal interfaces
- Simplified monitor implementation and interface *(~ 5% code size)*
- Fast boot time *(~ 10 ms)*



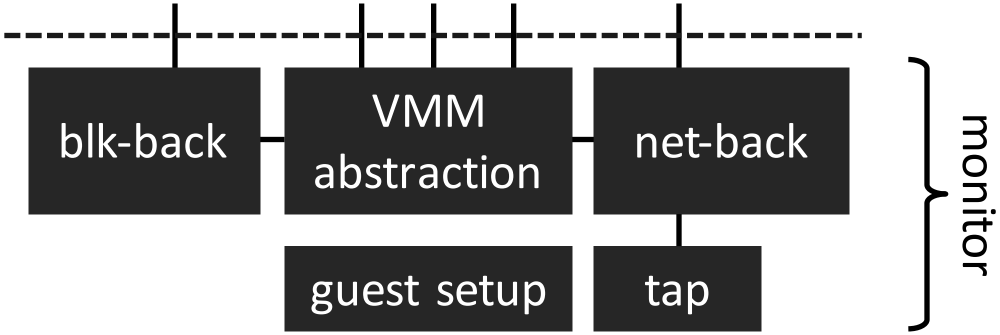
Minimal interfaces

- Interfaces to today's clouds are wide and **general-purpose**
  - Full virtualization, paravirtualization, OS-level (containers)

- A general purpose interface **cannot** be minimal

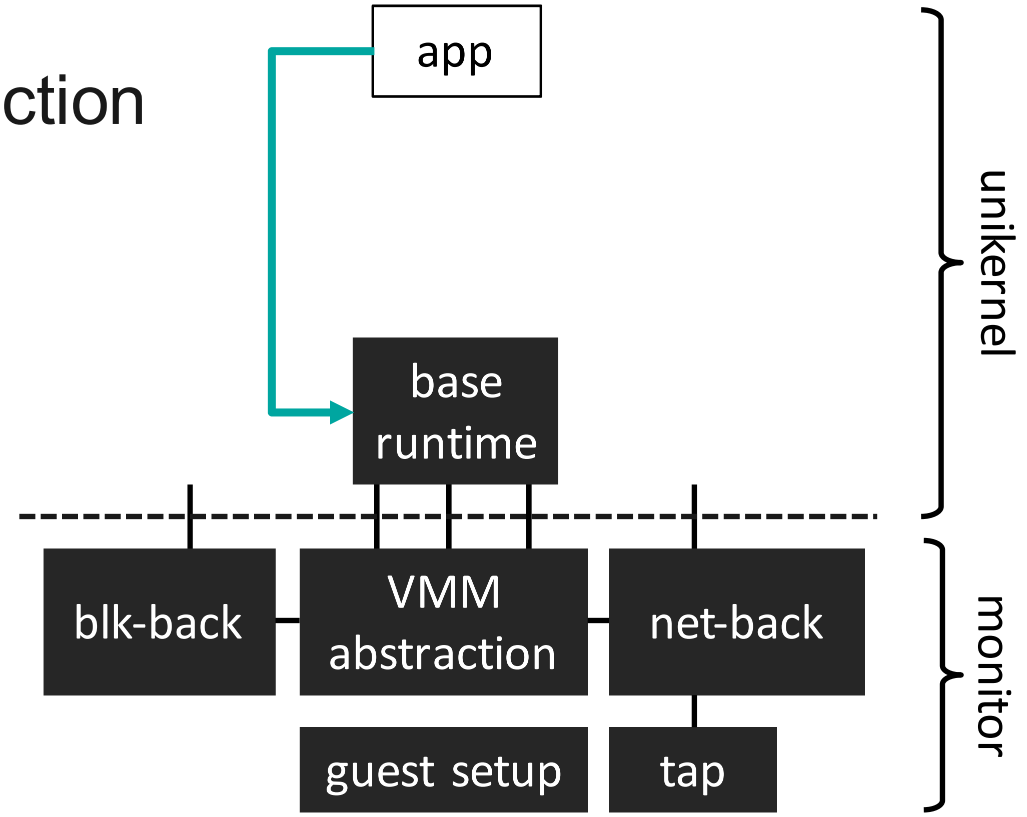
Building a unikernel

- Default monitor provides **generic** virtual HW abstraction



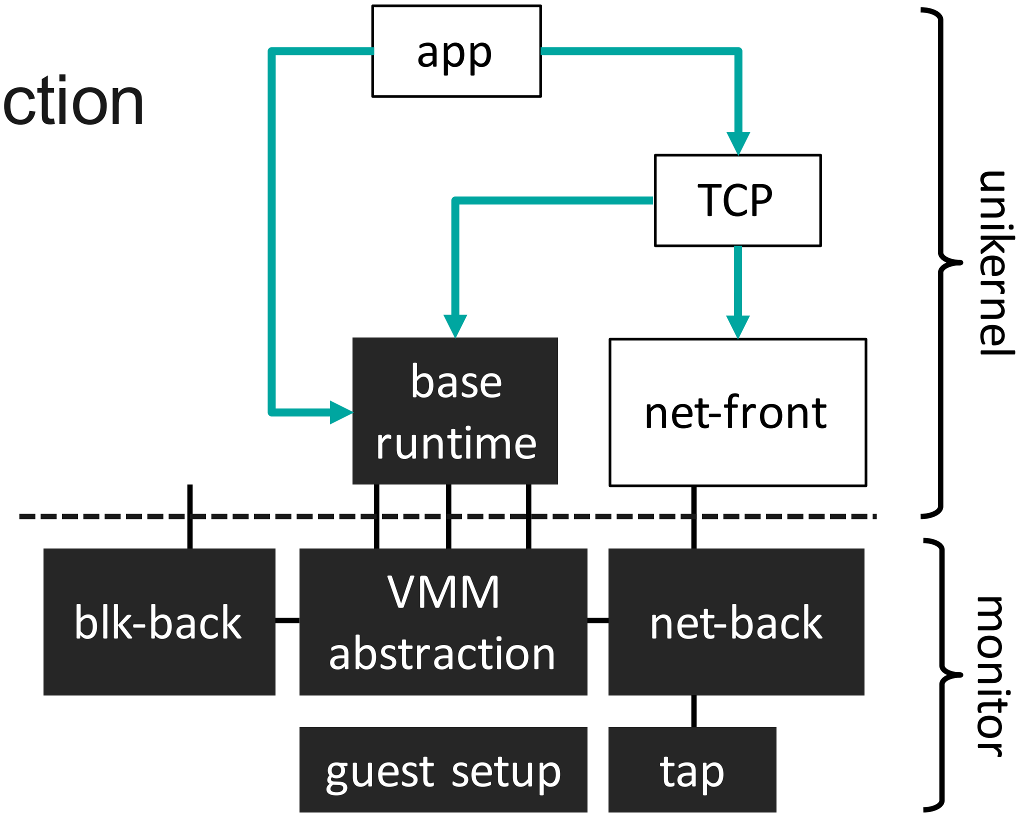
Building a unikernel

- Default monitor provides **generic** virtual HW abstraction
- Application depends on
  – base runtime



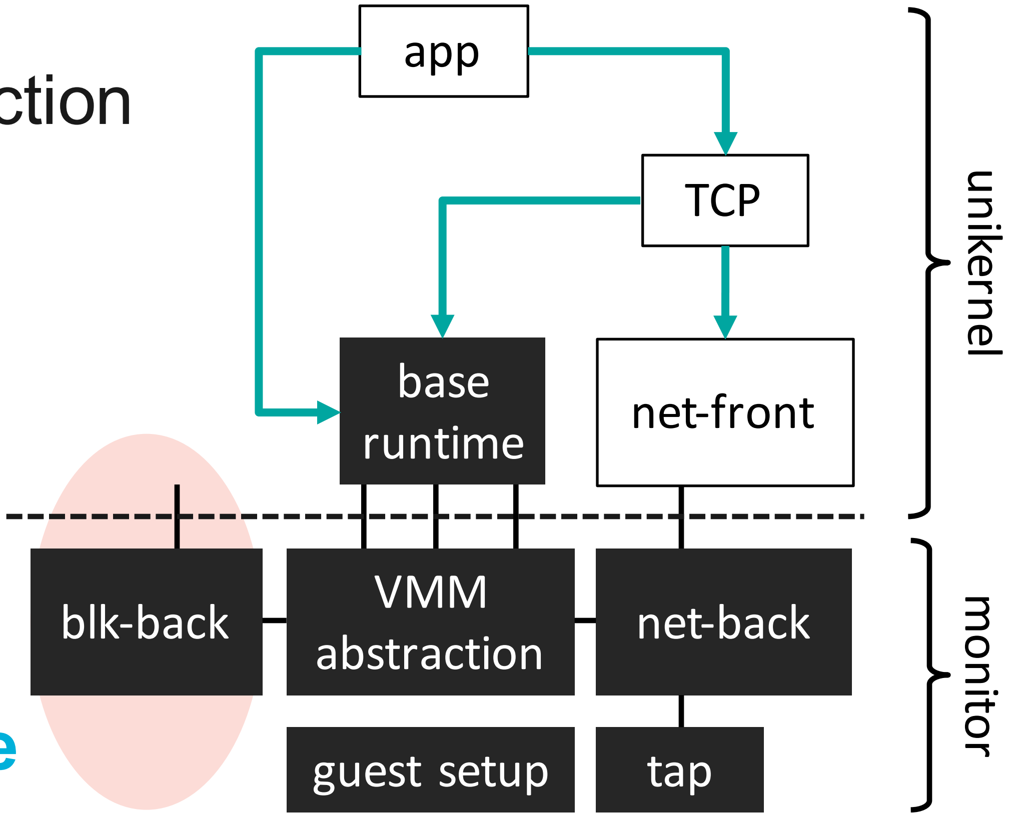
Building a unikernel

- Default monitor provides **generic** virtual HW abstraction
- Application depends on
  - base runtime
  - TCP stack



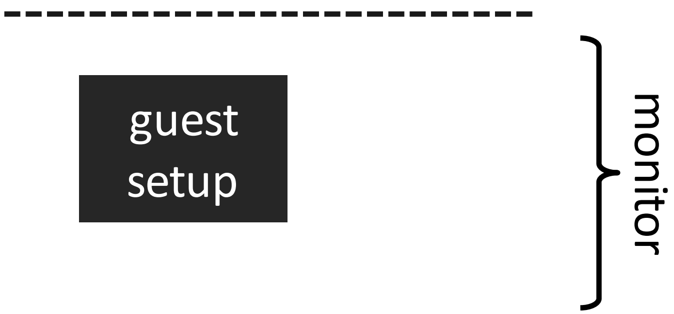
Building a unikernel

- Default monitor provides **generic** virtual HW abstraction
- Application depends on
  - base runtime
  - TCP stack
  - No disk
- **Monitor and interface are not minimal!**
  - VENOM attack



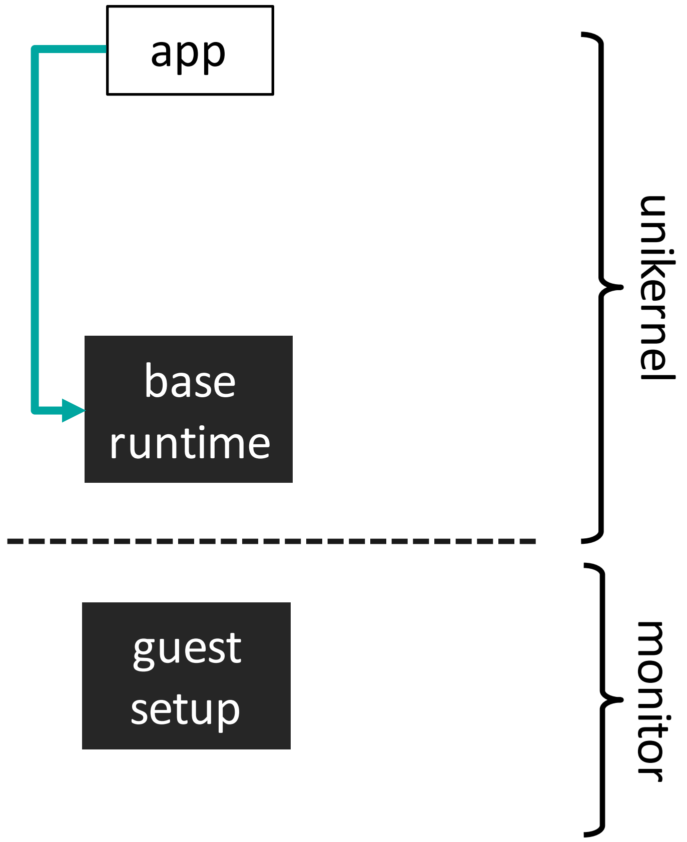
Building a unikernel and monitor

- Default monitor only provides isolated guest context
  - Destroys unikernel on any `exit`



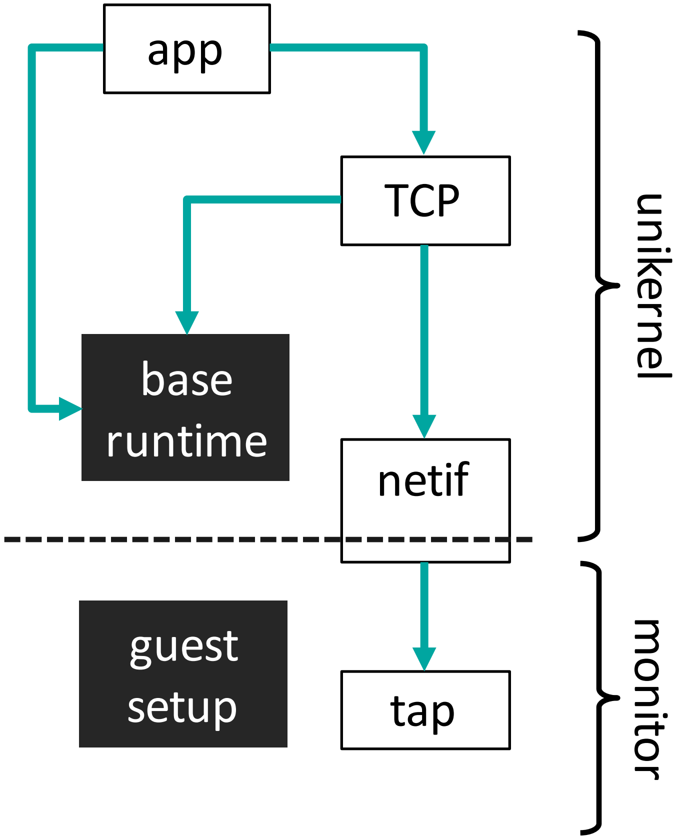
Building a unikernel and monitor

- Default monitor only provides isolated guest context
  - Destroys unikernel on any `exit`
- Application depends on
  - base runtime



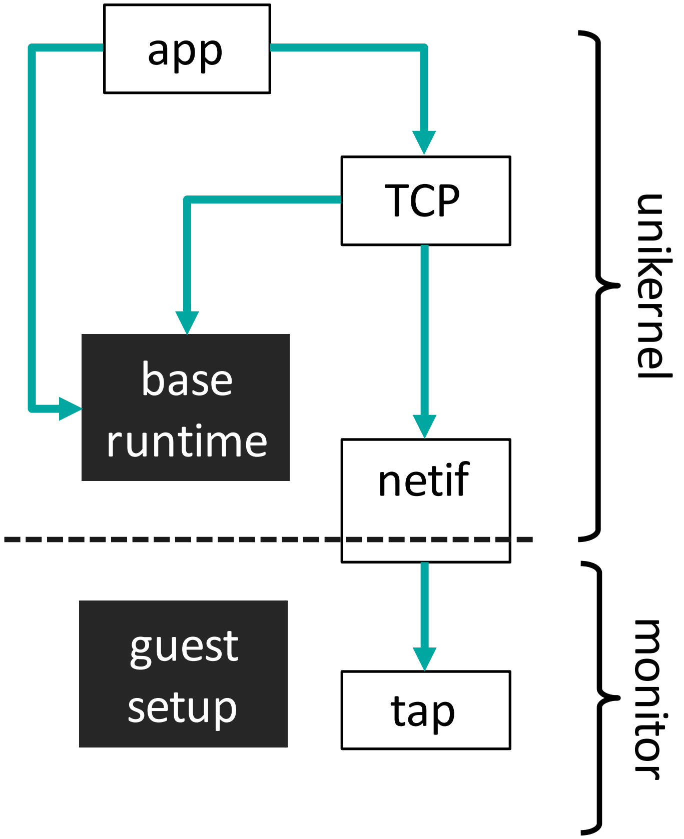
Building a unikernel and monitor

- Default monitor only provides isolated guest context
  - Destroys unikernel on any `exit`
- Application depends on
  - base runtime
  - TCP stack



Building a unikernel and monitor

- Default monitor only provides isolated guest context
  - Destroys unikernel on any `exit`
- Application depends on
  - base runtime
  - TCP stack
  - No disk



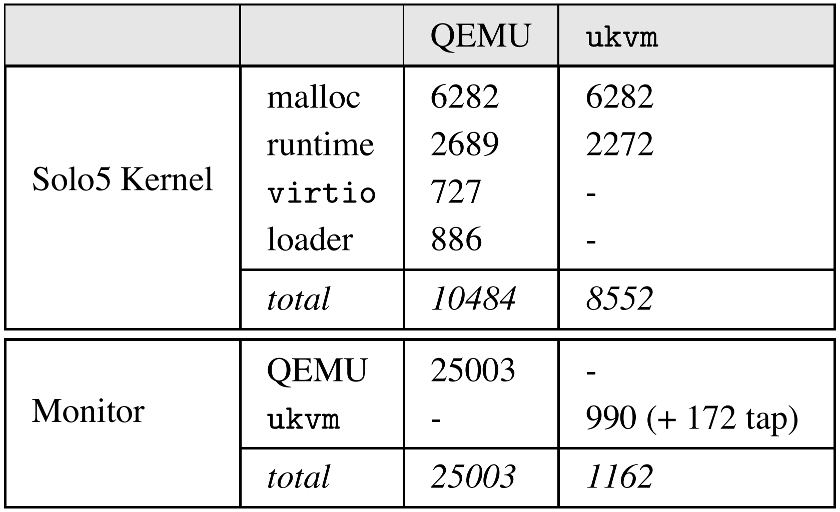
- **Monitor and interface is minimal!**
  - "Off by default"

Simplicity

- Legacy standards are unnecessary for the cloud
  – BIOS? PCI?
- Example: shared memory to send network packet

```
/* UKVM_PORT_NETWRITE  */
struct ukvm_netwrite {
    void *data; /* IN  */
    int len;    /* IN  */
    int ret;    /* OUT */
}
```

- What level of abstraction?
  – Generality tax
- Specialized interfaces
  – E.g., avoid VM introspection



		QEMU	ukvm
Solo5 Kernel	malloc runtime virtio loader	6282 2689 727 886	6282 2272 - -
	total	*10484*	*8552*
Monitor	QEMU ukvm	25003 -	- 990 (+ 172 tap)
	total	*25003*	*1162*

Boot time

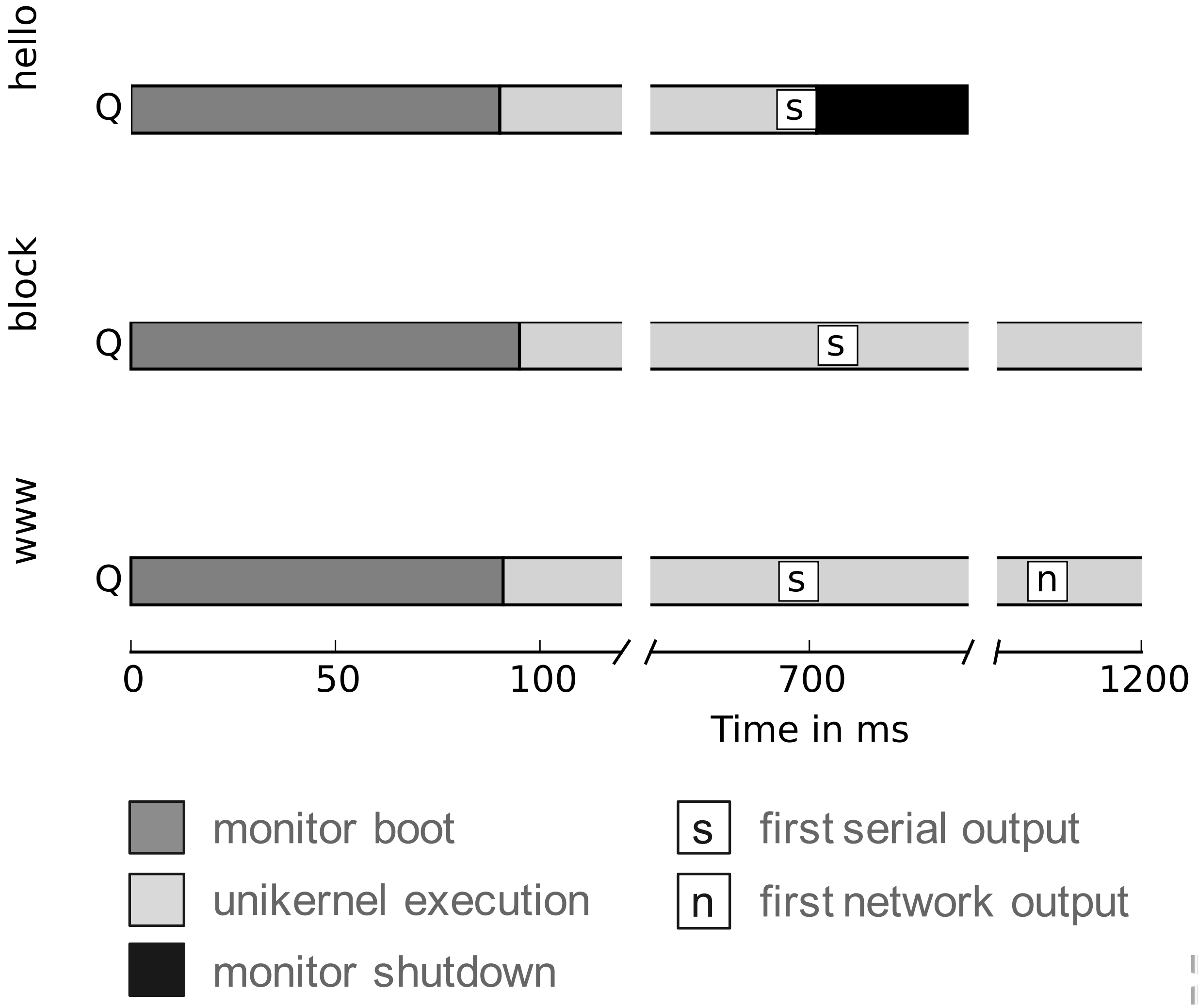
- New application domains require on-the-fly service creation
  - IoT, NFV, Amazon Lambda
  - Zero-footprint cloud, transient microservices

- Legacy protocols/emulation, virtual hardware negotiation, and range of guest support can slow things down

Boot times

- 3 applications
  - Hello world
  - Block device test
  - Static Web server

- QEMU: standard monitor

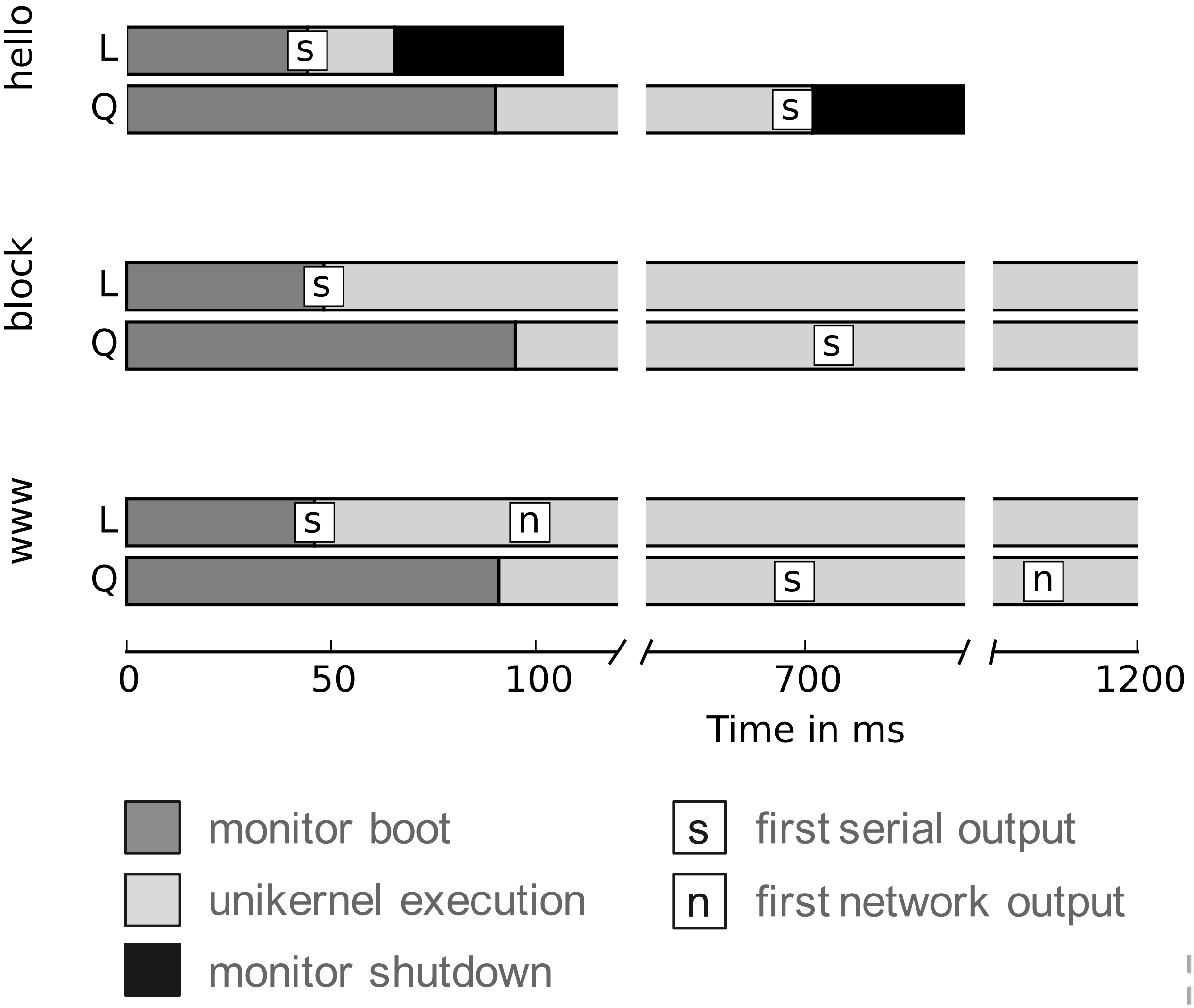


Boot times

- 3 applications
  - Hello world
  - Block device test
  - Static Web server

- QEMU: standard monitor

- `lkvm`: lightweight monitor



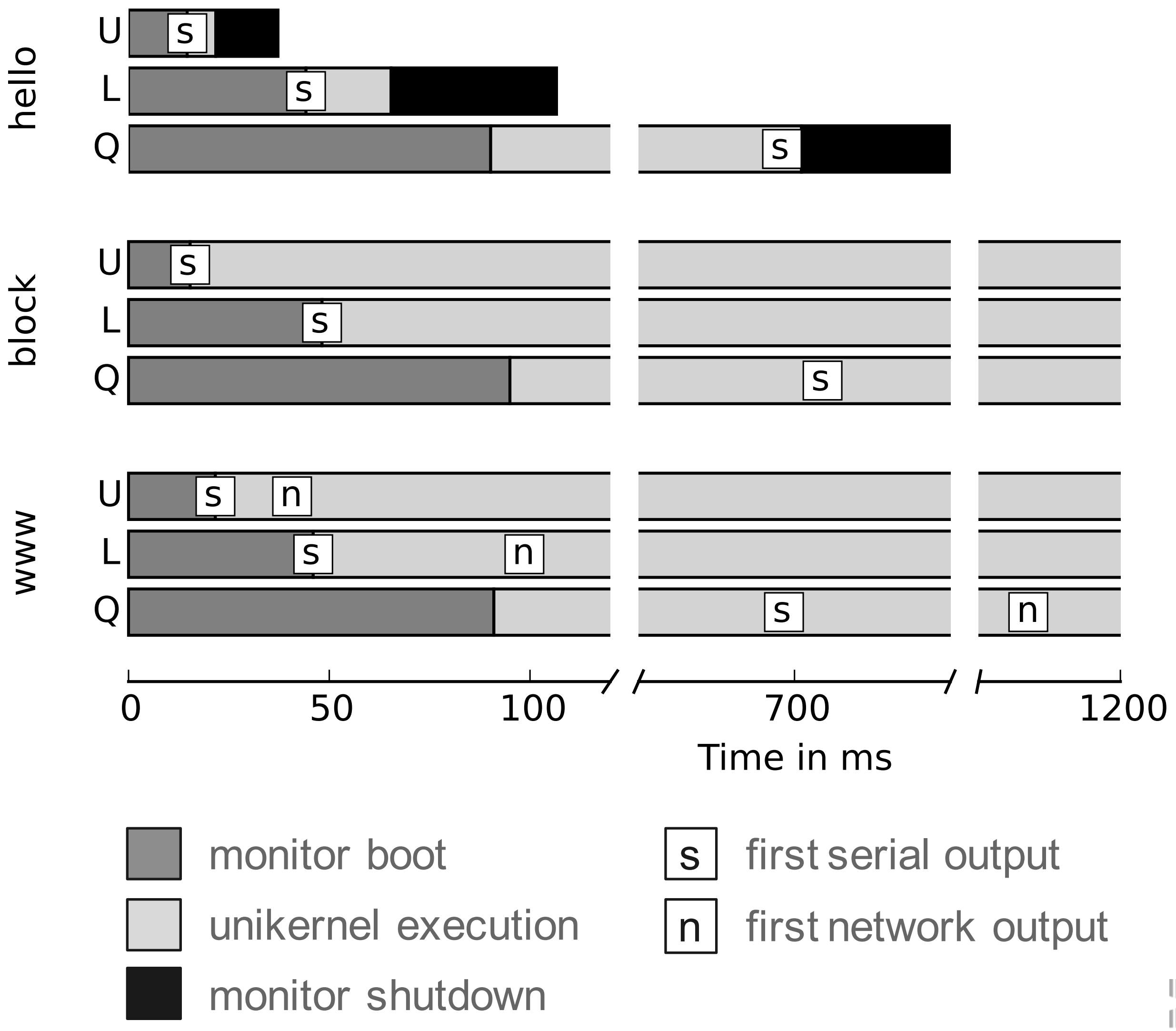
Boot times

- 3 applications
  - Hello world
  - Block device test
  - Static Web server

- QEMU: standard monitor

- `lkvm`: lightweight monitor

- **`ukvm`**: specialized monitor



Securing the monitors

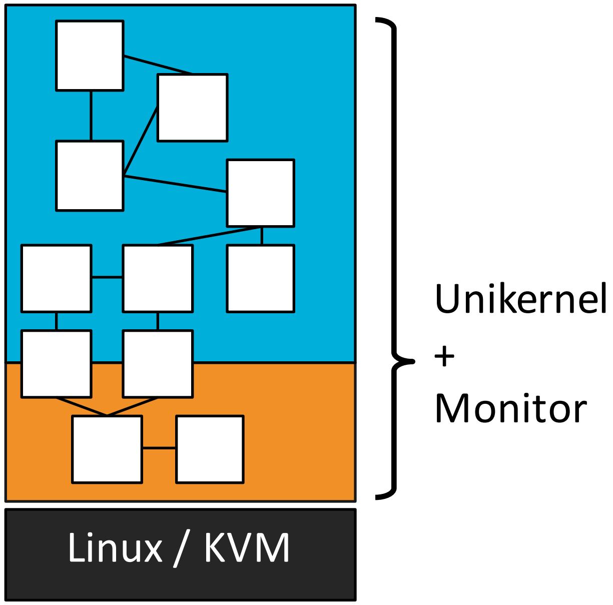
- Monitor is outside hardware protection domain

- Small enough for formal verification, audit?
- Cloud providers restrict monitors to certified modules?

Summary

- Extend minimalism through both unikernel and specialized monitor
  - Better security
  - Better performance
  - Better management

- Prototype: `ukvm`
  - https://github.com/djwillia/solo5
  - Currently being upstreamed as MirageOS backend
  - Thank you to MirageOS community, (especially Martin Lucina, Docker)


Unikernel
+
Monitor
Linux / KVM