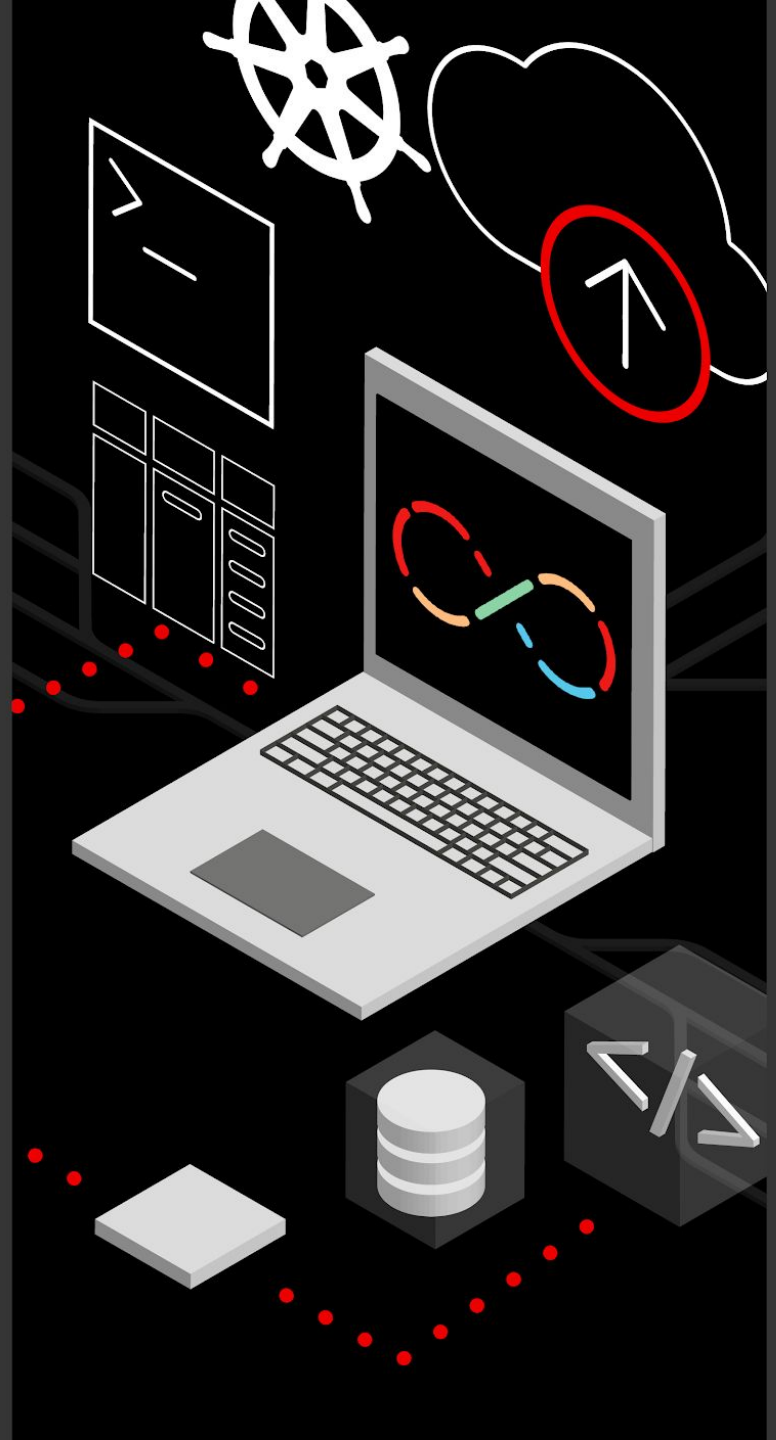


Linux Day Milano 2023

Unikernel

An idea for the future of Linux from the past

$ whoami
Natale Vinto
Developer Advocate Lead @ Red Hat
@natalevinto@mastodon.uno

Red Hat

Agenda

- Unikernel
- Unikernel & Cloud: what?
- Linux and Unikernel
- UKL
- Tests and performance
- Q&A

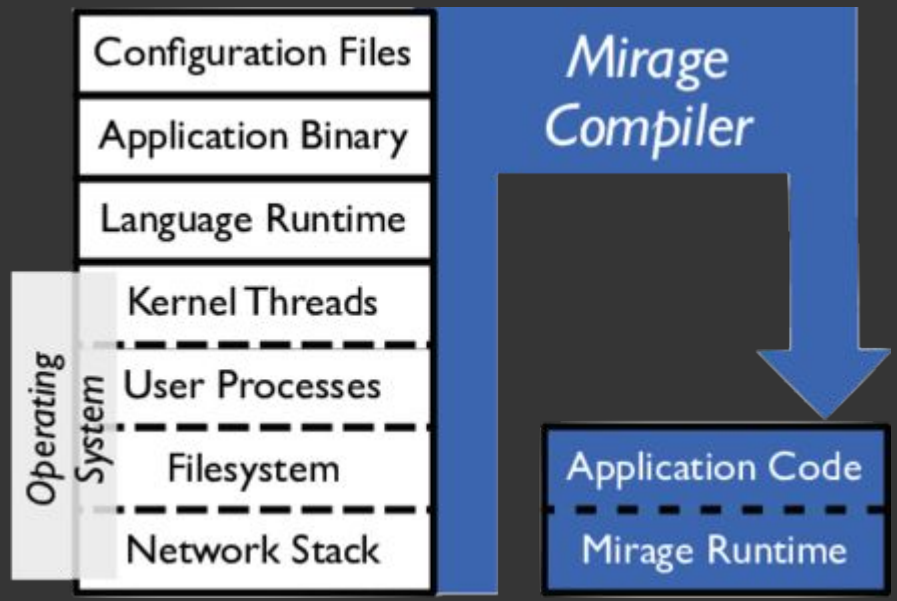
The unikernel architecture builds on concepts developed by Exokernel and Nemesis **in the late 1990s**

Unikernel

- A **unikernel** is a computer program **statically linked with the operating system** code on which it depends.
- Unikernels are built with a specialized compiler that identifies the operating system services that a program uses and links it with one or more library operating systems that provide them.
- Such a program **requires no separate operating system** and can run instead as the guest of a hypervisor

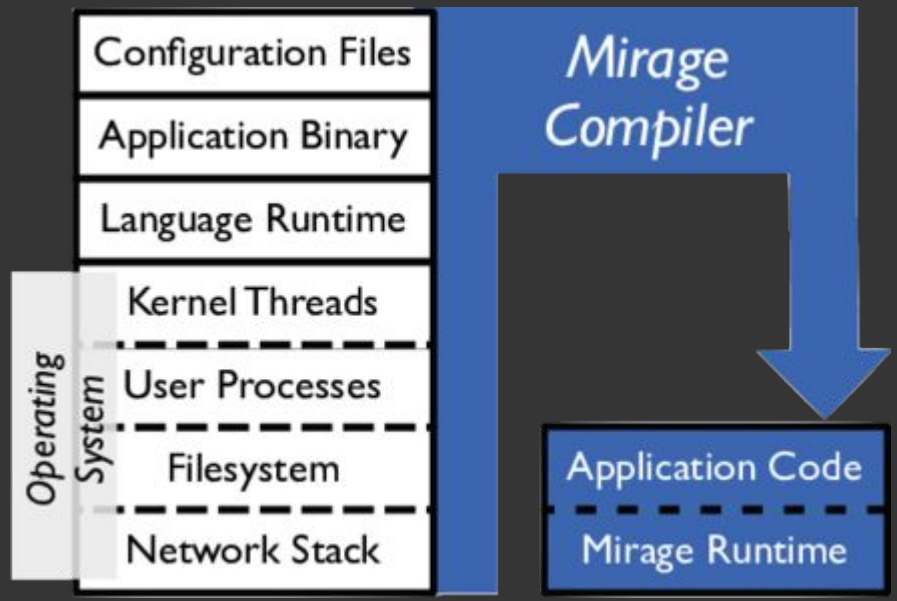
Unikernel: pros

- **Single address space**: no need to transition from user to kernel space and vice versa
- **Direct access to the hardware**: TRAP vs Context Switch
- **Lightweight**: smaller apps bundled directly into a kernel
- **Surface attack reduced**
- **Runs typically on VMs**



Unikernel: cons

- **Lack of standardisation**: there are many approaches to Unikernels but not a real standard
- **Hardware heterogeneity**: drivers
- **One app = One VM**: any new app is a VM, hardware density need to be considered
- **Complexity** : libraries, frameworks, documentation, community and enterprise support



Unikernel and Clouds?

Unikernel for Cloud computing

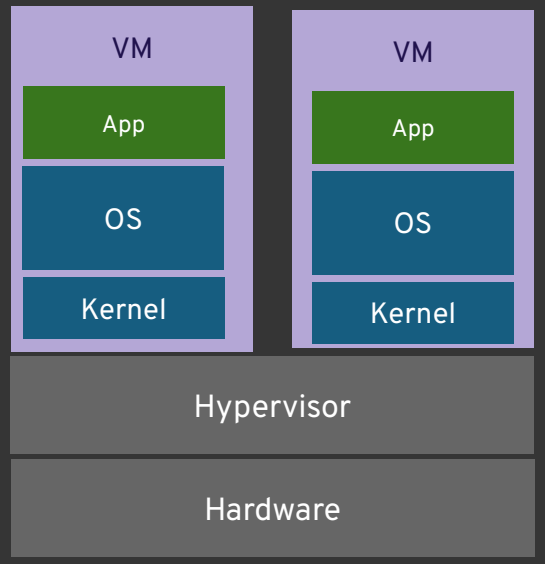
Unikernels offer improved scalability compared to traditional VMs and containers. As unikernels are lightweight and have a smaller resource footprint, they can be **more easily scaled up or down to meet the demands of an application**. This makes them particularly well-suited for cloud environments, where the ability to quickly and efficiently scale resources is critical.

Unikernels

Project	language	src	started	last commit	status
Unikraft	C	https://github.com/unikraft/unikraft	2017	recent	active
NanoOS	C	https://github.com/nanovms/nanos	2017	recent	active
Mirage OS	OCaml	https://github.com/mirage/mirage	2014	recent	active
OSv	C	https://github.com/cloudius-systems/osv	2012	recent	active
RustyHermit	Rust	https://github.com/hermitcore/rusty-hermit	2017	recent	active
HermiTux	C	https://github.com/ssrg-vt/hermitux	2017	recent	active
StardustOS	C, Rust	https://stardustos.gitbook.io/stardust/	2019	recent	active
Kontain	C	https://github.com/kontainapp/km	2018	recent	active

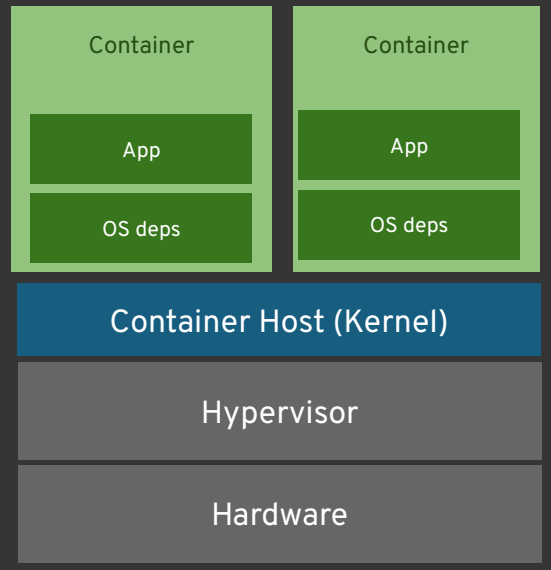
Unikernel: a comparison

VIRTUAL MACHINES



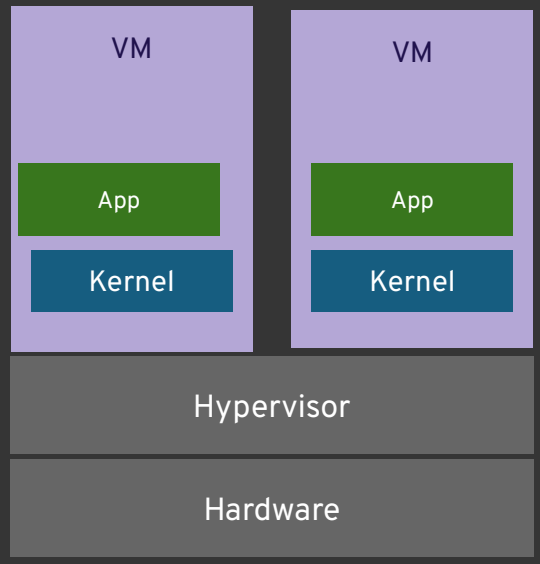
VM isolates the hardware

CONTAINERS



Container isolates the process

UNIKERNELS



App is bundled into Kernel

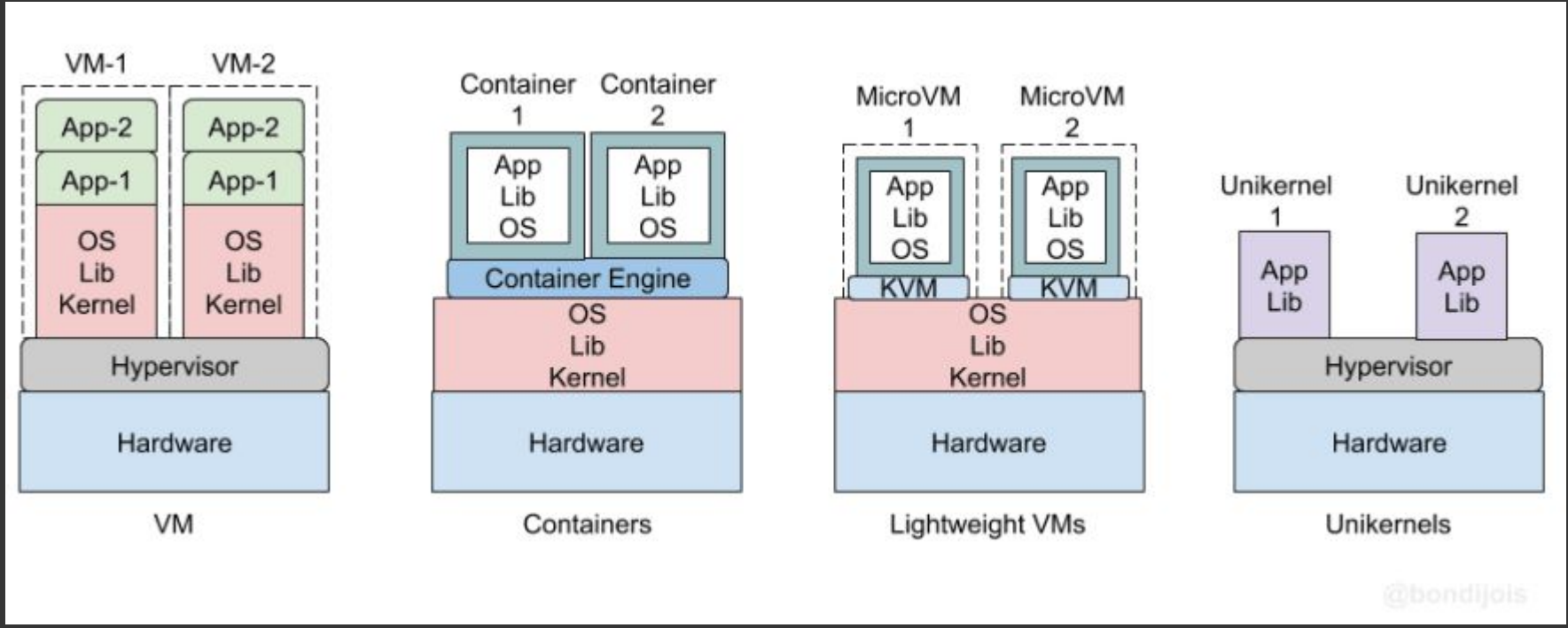
Red Hat Developer

Red Hat

Unikernel: a comparison

	Virtual Machines	Containers	Unikernels
Time performance	Slowest of the 3	Fast	Fast
Memory footprint	Heavy	Depends on the number of features	Light
Security	Very secure	Least secure of the 3	Very secure
Features	Everything you could think of	Depends on the needs	Only the absolute necessary

Unikernel VS Traditional and Micro VMs



- No need for OS
- Hypervisor type 1

Red Hat Developer

v0000000

Red Hat

Unikernel VS VMs and Micro VMs

VMs	Containers	Lightweight VMs	Unikernels
Runs on a Hypervisor	Requires a Host OS	Requires a Host OS	Runs on a Hypervisor
Uses own Kernel + Allocated Resources	Relies on Host Kernel + Consumes Host Resources	Dedicated virtualized Kernel(KVM) + Allocated Resources	Built-in Kernel libraries + Allocated Resources
Multiple Processes	Multiple Processes	Multiple Processes	Single Process
Large Attack Surface	Relatively Low Attack Surface	Low Attack Surface	Minimal Attack Surface

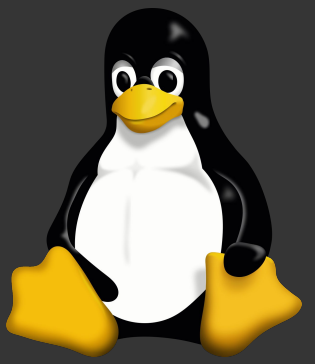
Unikernel development

There are essentially two approaches to Unikernel development:

- **Clean slate**: Kernel is written from scratch, this gives full control on programming language choice and methodology.
- **Strip down**: existing kernel codebase is stripped of functionality deemed unnecessary for the unikernel.

They both present disadvantages such as drivers compatibility for different hardware or out-of-tree forks hard to maintain and keep up to date.

UKL: UniKernel Linux



- UKL is a **patch to Linux** and **glibc** which goal is to minimize changes both to the Linux kernel and to applications.
- By reusing Linux, UKL gains the advantages of Linux for free, especially wide driver support.
- App and kernel are linked with the Linux kernel into a final **vmlinuz** and run in **kernel space**
- Project started by Boston University in collaboration with Red Hat Research

Linux performance

There is growing evidence that the structure of today's general-purpose operating systems is problematic for a number of key use cases such as I/O, Networking

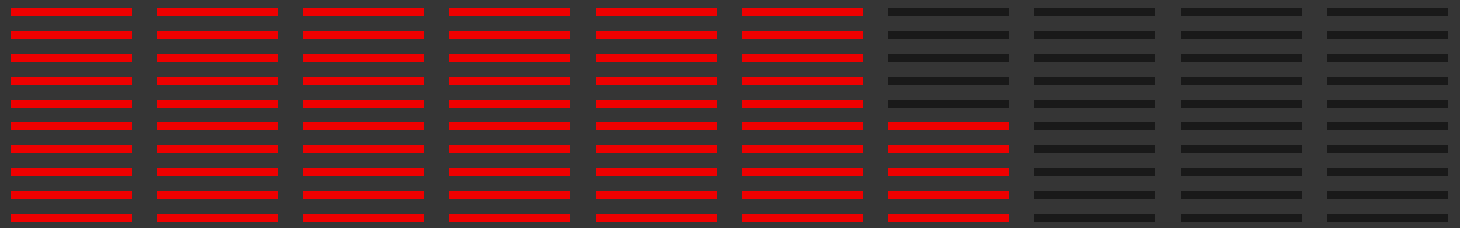
Some approaches:

- DPDK
- XDP/eBPF
- io_uring
- Interrupt Affinity, NUMA, etc

Source:

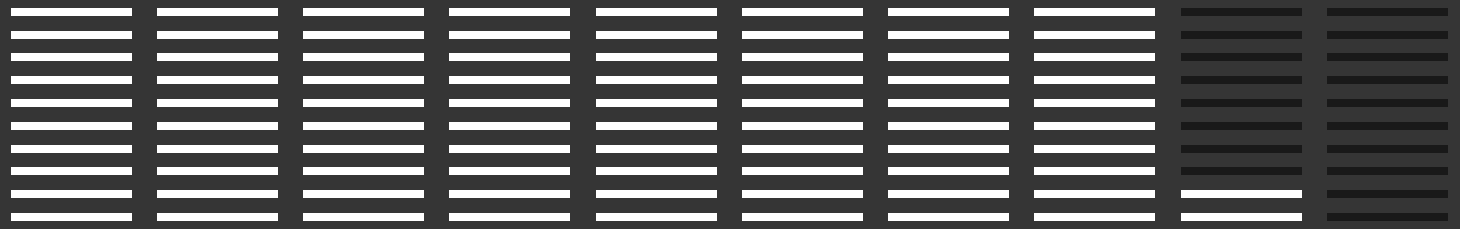
Insert source data here

UKL Performance vs Generic Linux



+5%

Simplest UKL, default settings. Base path is 500 lines of code.



+26%

Optimization for applications like Redis that always uses TCP to call the underlying routines directly. Full model patch is 1250 lines of code.

Red Hat Developer

Source:
Insert source data here
Insert source data here
Insert source data here

Red Hat

UKL Performance vs Generic Linux

- The 64-bit virtual address space layout for non-UKL user process.
- In orange are the segments that are relocated to the kernel half for the UKL process.

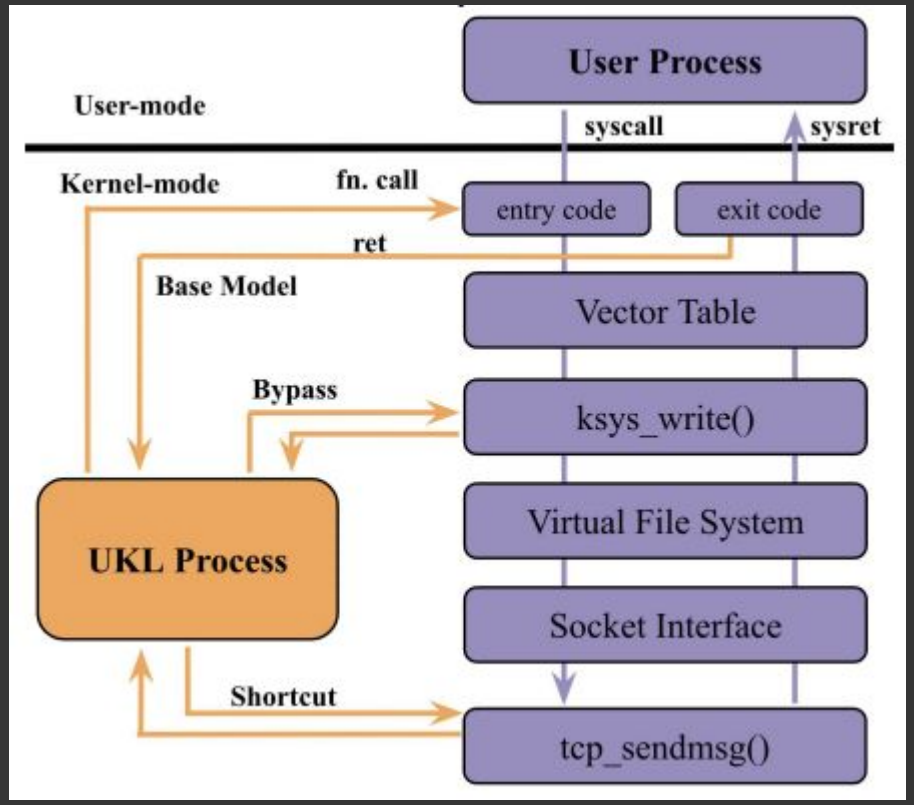
Red Hat Developer

V0000000

Red Hat

UKL Performance vs Generic Linux

- A schematic of a write system call destined for a network device.
- The three alternative internal entry points that a UKL process exercises are shown in orange.



UKL Performance vs Generic Linux

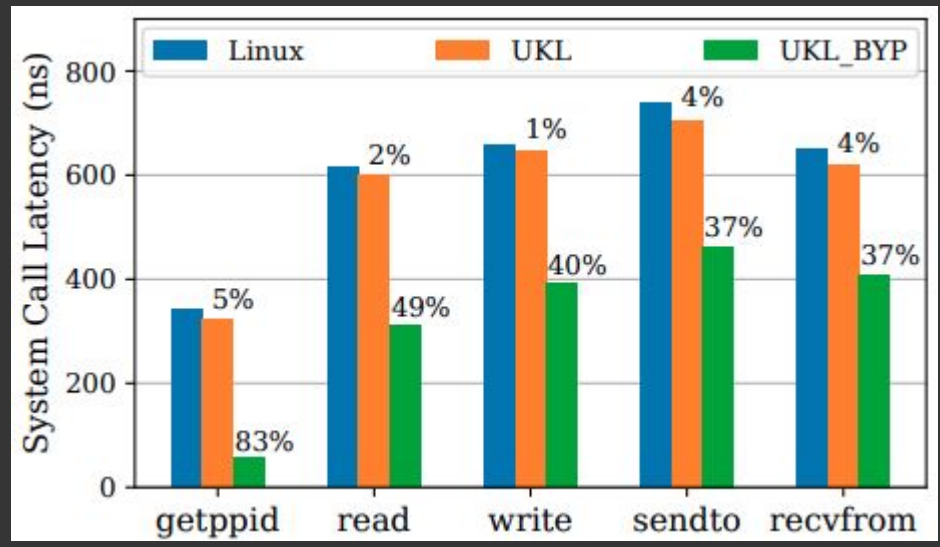
On the transition between the execution models, UKL performs the same entry and exit code of the Linux kernel with the difference that:

- Transitions to kernel code are done with a procedure call rather than a syscall
- Transitions from the kernel to application code are done via a ret rather than a sysret or iret; see "Base Model"

This transition code includes changing between application and kernel stacks, RCU handling, checking if the scheduler needs to be invoked, and checking for signals. In addition, it includes setting a per-thread ukl_mode to identify the current mode of the thread so that subsequent interrupts, faults, and exceptions will go through normal transition code when resuming interrupted application code

UKL Performance vs Generic Linux

- UKL_BYP is a UKL configuration that allows the application, on a per-thread basis, to tell UKL to bypass entry and exit code for some transitions between application and kernel code
- A developer can invoke an internal kernel routine directly, where no automatic transition paths exist
- UKL_RET replaces iret with ret



Red Hat Developer

Red Hat

Test with Redis: +26% performance

Single-threaded application

- Developed a kernel call “shortcut” that would directly invoke the underlying TCP routines **tcp_sendmsg** and **tcp_recvmsg**
- The change to Redis to invoke this routine required only 10 lines of code to be modified.

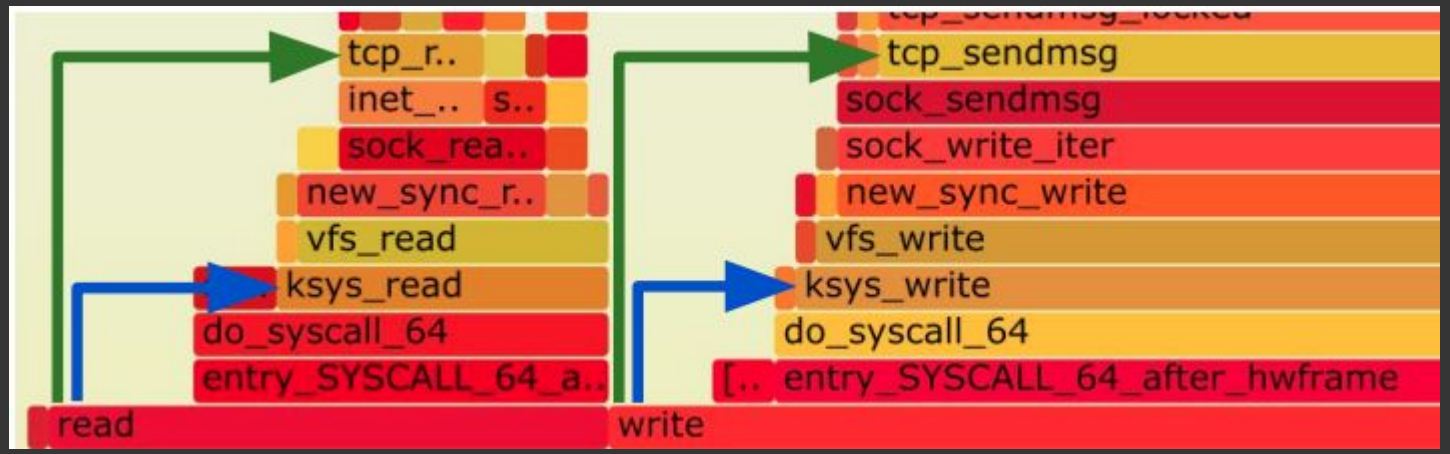
redis

Redis-UKL

Optional subheading

Redis-UKL

UKL_BYP



Test with Memcached: +10% performance

Multi-threaded application

- UKL_RET_BYP (shortcut) with virtio network para-virtualization drivers
- UKL_RET_BYP (shortcut) gets up to 10% tail latency improvement over Linux, even as the load on the Memcached server increases

DEMO!

Red Hat Developer

What's Next?

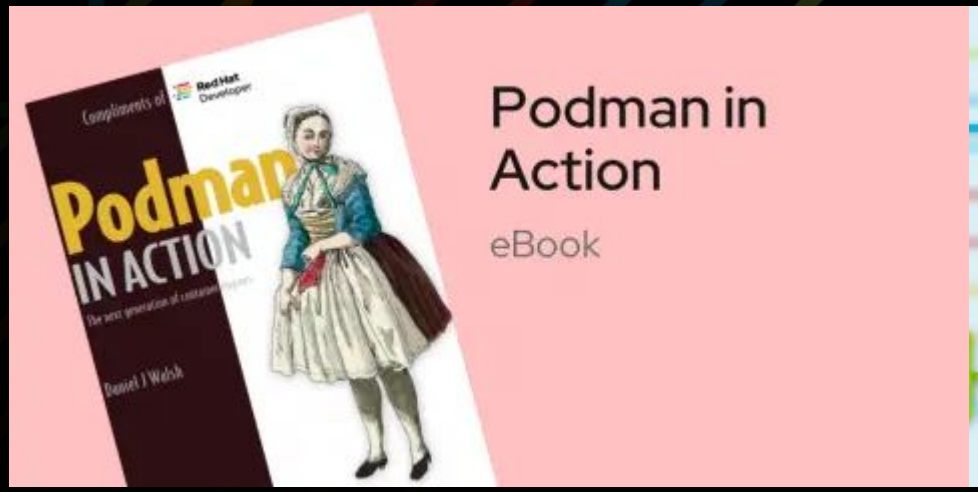
- Fedora COPR service
- Linux community acceptance: unikernel module
- Community and commercial use cases
- Try it out!

Links

- RHRQ August 2023, “Unikernel Linux moves forward”
- Unikernel Linux - EuroSys '23: Proceedings of the Eighteenth European Conference on Computer Systems, May 2023, Pages 590–605
- UKL Github

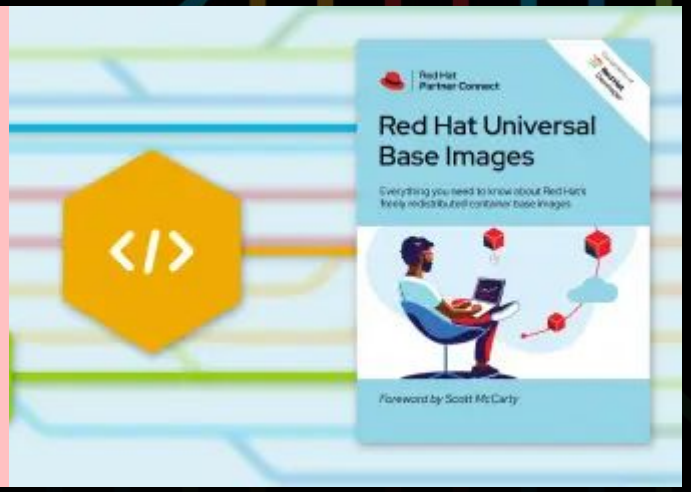
Red Hat

Red Hat Developer



Podman in Action

eBook



youtube.com/RedHatDevelopers

linkedin.com/showcase/red-hat-developer

facebook.com/RedHatDeveloper

twitter.com/rhdevelopers

Red Hat

Red Hat Developer

Join Red Hat Developer.

Build here. Go anywhere.

youtube.com/RedHatDevelopers

linkedin.com/showcase/red-hat-developer

facebook.com/RedHatDeveloper

twitter.com/rhdevelopers

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/showcase/red-hat-developer

facebook.com/RedHatDeveloper

youtube.com/RedHatDevelopers

twitter.com/rhdevelopers

Red Hat