KUBERNETES COMMUNITY DAYS BEIJING 2025

Exploring the Next Generation of Secure Containers: gVisor and Kata Fusion

Xuewei Niu Software Engineer, Ant Group

Hang Su Software Engineer, Ant Group

Tiwei Bie* Staff Engineer, Ant Group

Content

01 Secure Containers Overview

02 Inside Existing Solutions

03 User-mode Linux

04 NanoPVM

Part 01

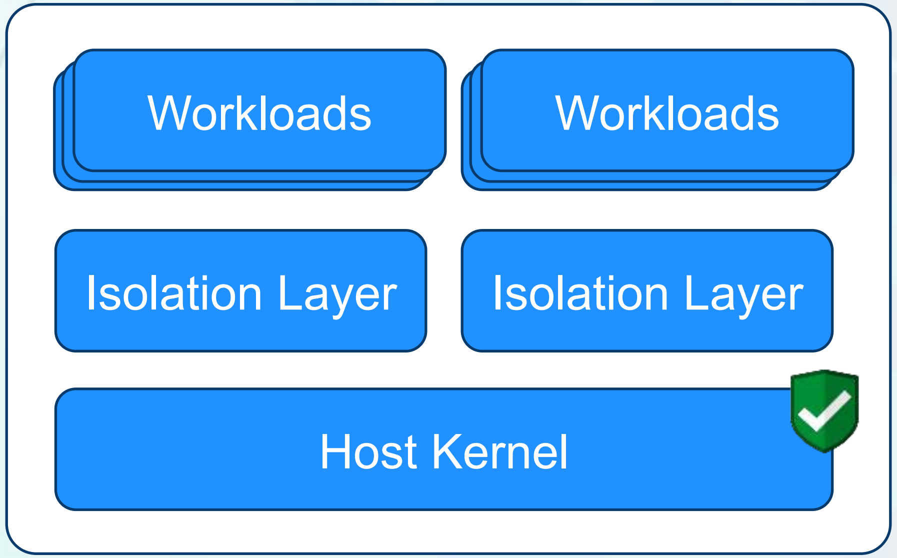
Secure Containers Overview

Why secure containers?

Secure Containers Overview

Secure containers are more than security

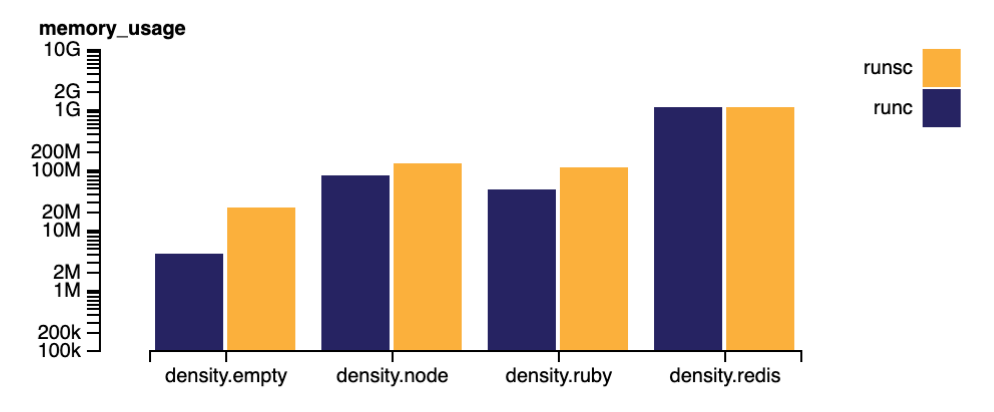
- **Security isolation** prevents sensitive instruction escapes
  - Executing untrusted code
  - Multi-tenancy
- **Performance isolation** prevents scheduling, networking, I/O interference
  - Online-offline hybrid deployment
- **Fault isolation** prevents shared kernel crashes/faults

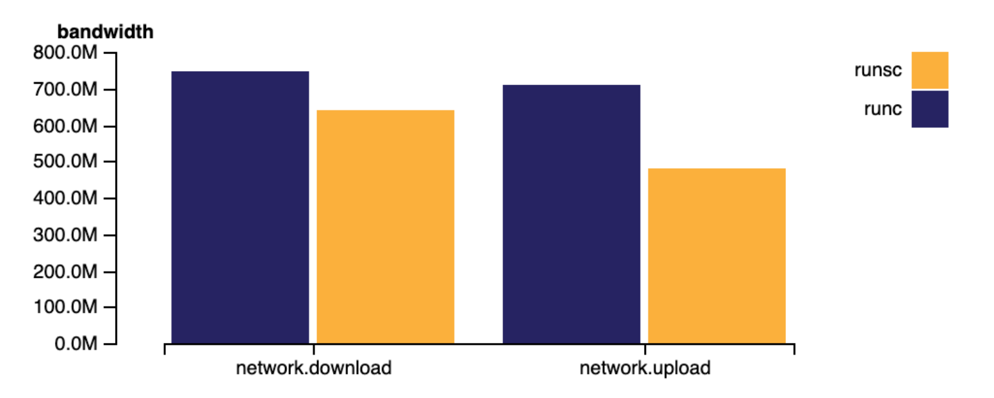


Secure Containers Overview

At the mean time…

- Runtime overheads cause increased latency, reduced throughput or density
- Resource footprints from additional components, e.g. guest kernel/Sentry



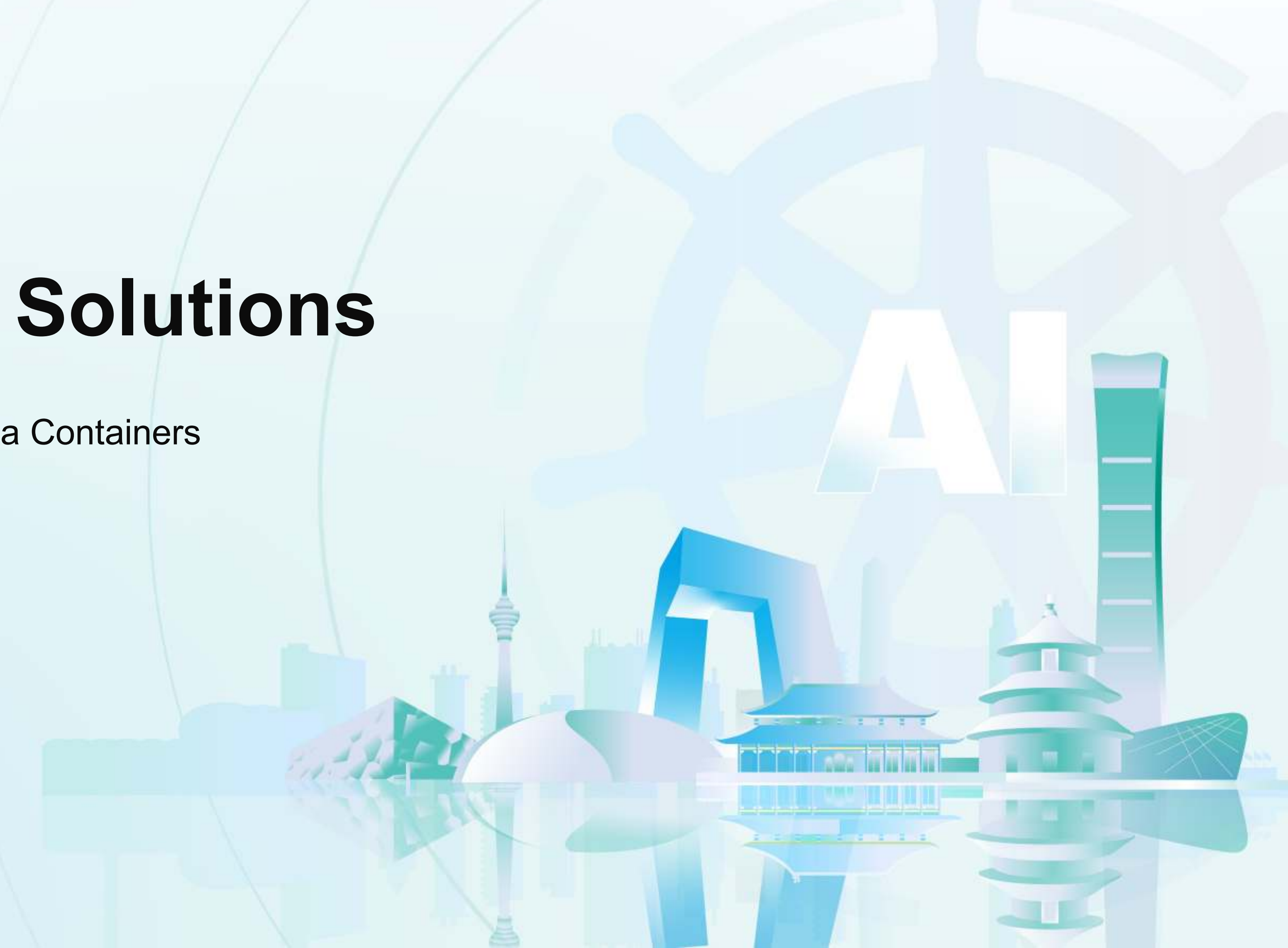


*Data from gVisor Documentation
https://gvisor.dev/docs/architecture_guide/performance/

Part 02

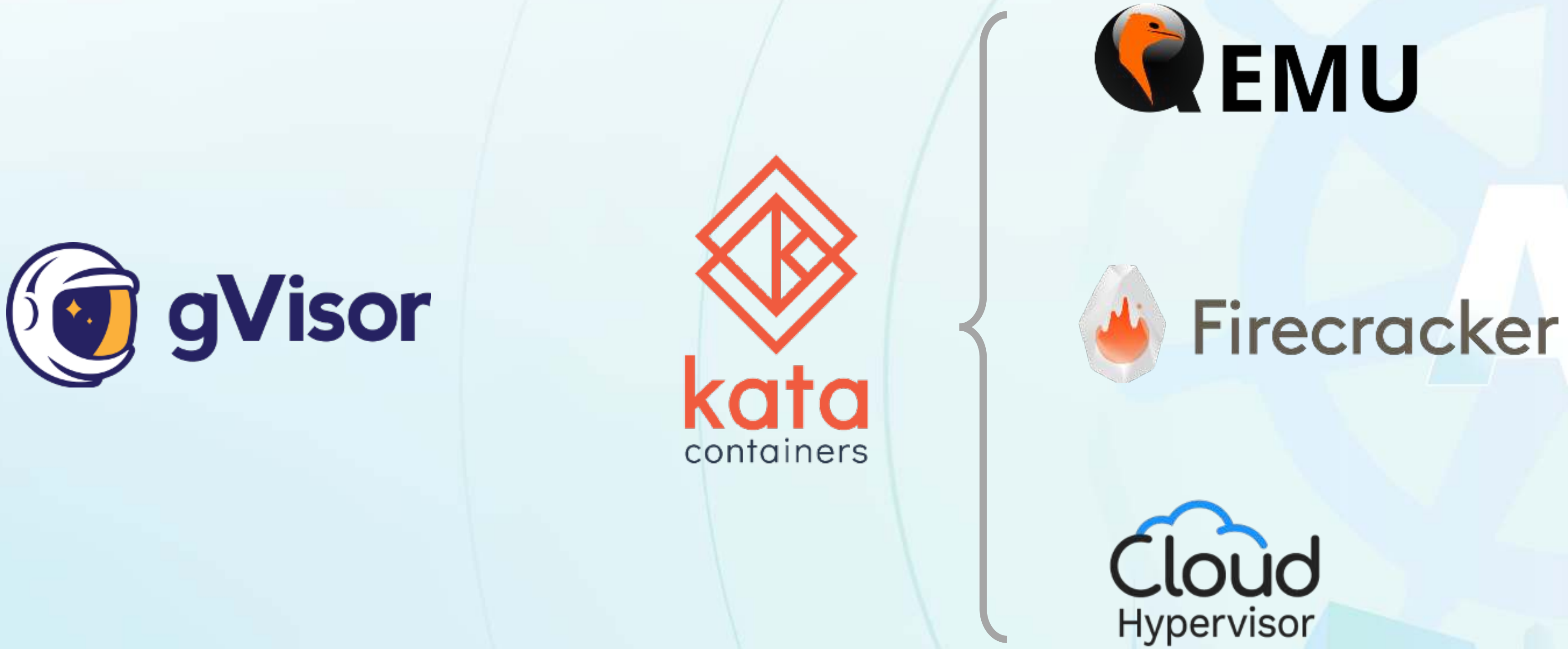
Inside Existing Solutions

Brief Introduction to gVisor and Kata Containers



Inside Existing Solutions

Secure Containers Landscape



gVisor

kata containers

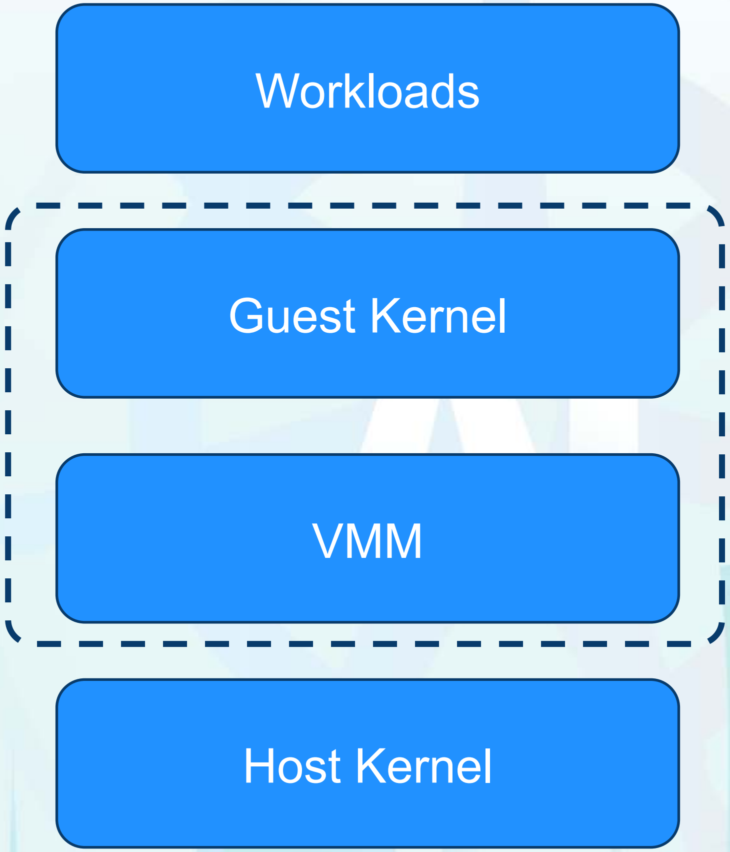
QEMU

Firecracker

Cloud Hypervisor

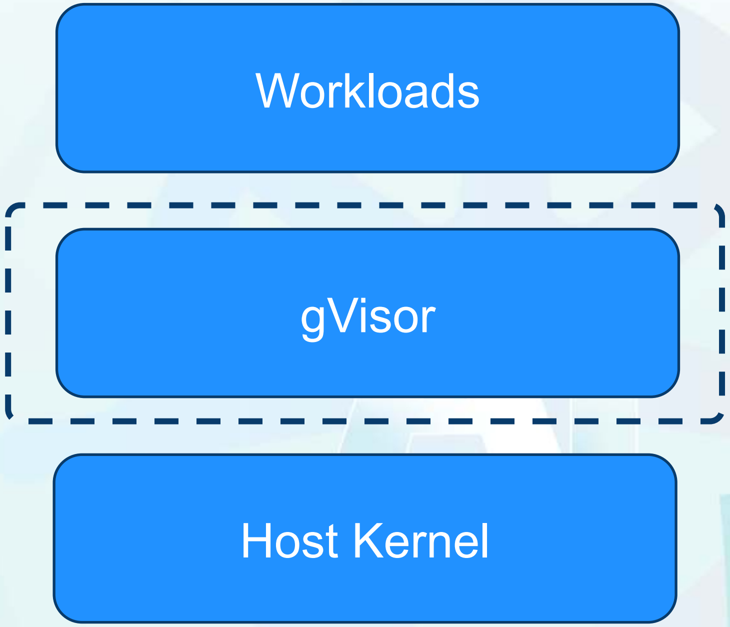
Inside Existing Solutions: Kata Containers

- The speed of containers, the security of VMs
- Kata provides a **virtual machine**: a typical model with a clear layered structure
  - VMM provides virtual devices
  - Guest kernel provides dedicated runtime environment
- Most of the work is focused on reducing overheads
  - MicroVMs: Firecracker (for FaaS), Cloud-hypervisor (for general tasks), etc.
  - Disabled unnecessary kernel configs

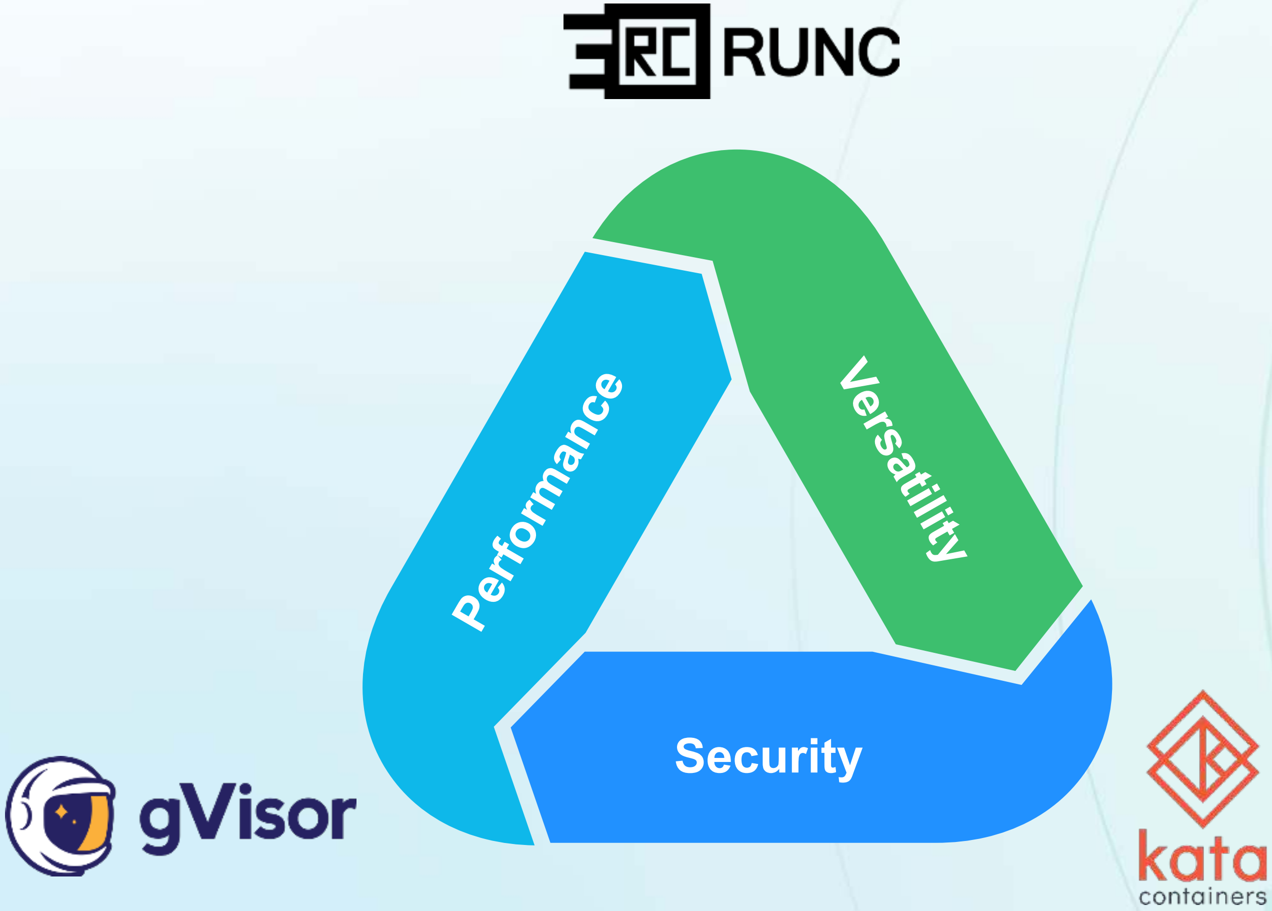


Inside Existing Solutions: gVisor

- gVisor is an independent kernel running in user-space, exposing a *Linux-like* interface.
- gVisor provides a **virtual kernel** with merged VMM and guest kernel layers. The model eliminates unnecessary virtualization assumptions.
- Supported platforms: KVM, systrap and ptrace
- Sentry ≈ kernel: When the application makes a system call, the platform redirects the call to the Sentry, which will do the necessary work to service it.



Inside Existing Solutions



So… what is the next?

Part 03

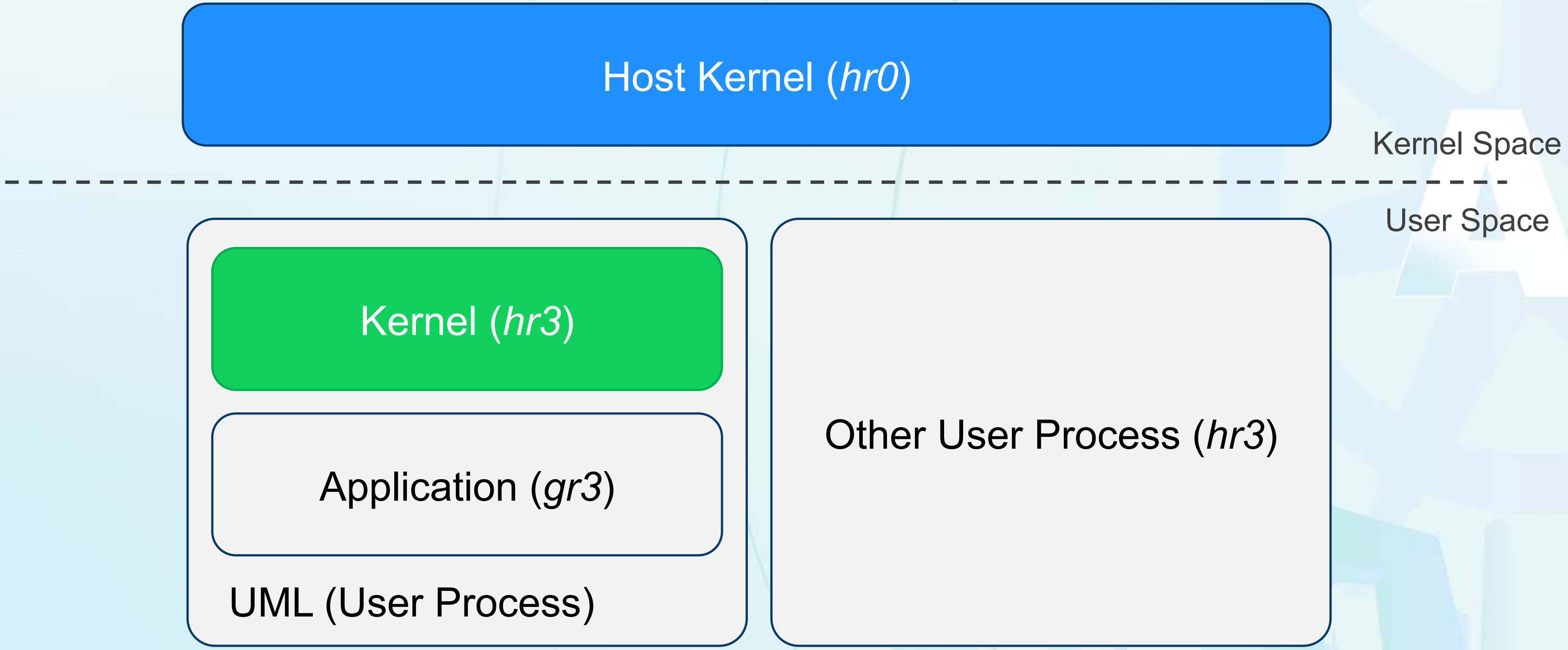
User-Mode Linux

The Next Generation of Secure Containers

User-Mode Linux

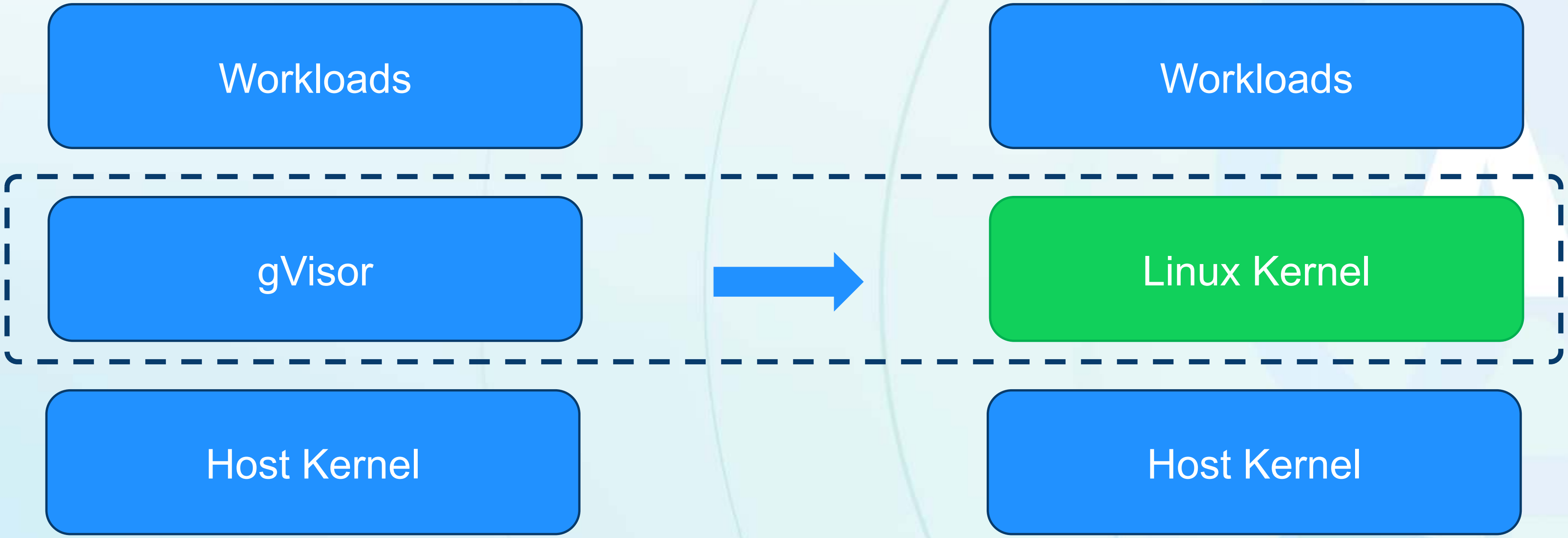
UML was first released in 1991 and has been merged into the Linux mainline.

- Build with *make ARCH=um*



User-Mode Linux

UML is more like gVisor from perspective of architecture



User-Mode Linux

Compared with gVisor

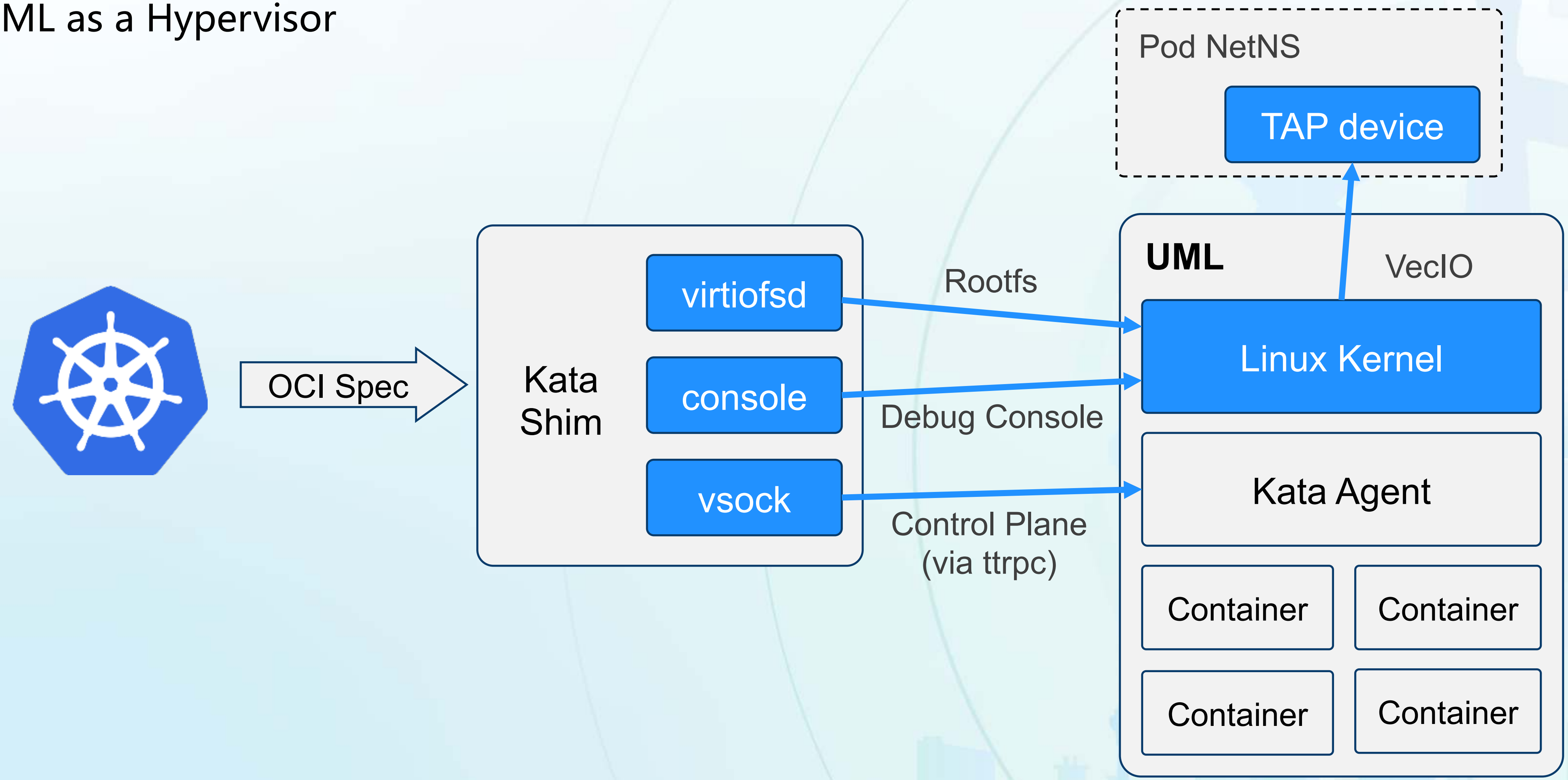
- Running applications without modification or adaptation
- Time-tested kernel (also known as implementation costs)
- Costs of Golang: runtime, garbage collection, etc.
- Free to use any customized kernels

Compared with Kata Containers

- With **NanoPVM**, running in environments that do not support nested virtualization
- Simpler virtualization model
- vGPU support (nvproxy)

User-Mode Linux

UML as a Hypervisor



Part 04

NanoPVM

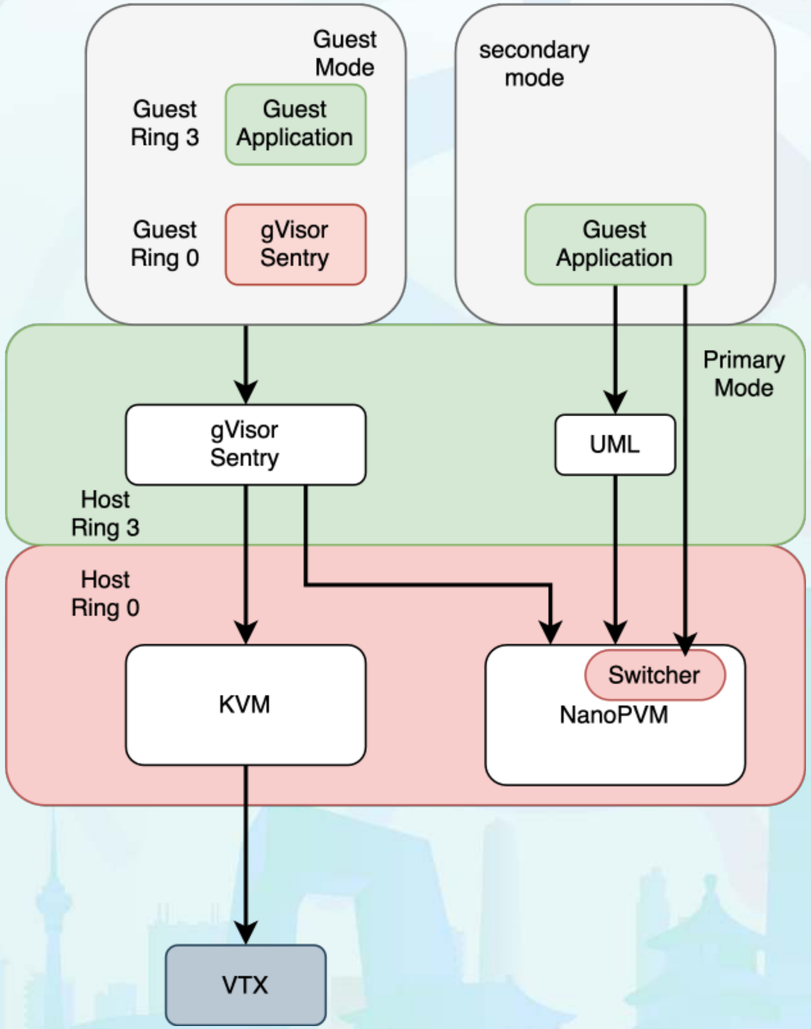
Enables User Mode Kernel to Run in Any Environments

What is NanoPVM ?

- Same architectural level as KVM ,

  Currently exists as an out-of-tree Linux kernel module

- **4 Core points:**
  - **New virtualization model paradigm**
  - Memory isolation utilizes **Direct Memory Management**
  - **Executable Context Management**, No instruction emulation
  - Minimalist Interface



New virtualization model paradigm

Popek and Goldberg virtualization requirements:

- Equivalence / Fidelity
- Resource control / Safety
- Efficiency / Performance

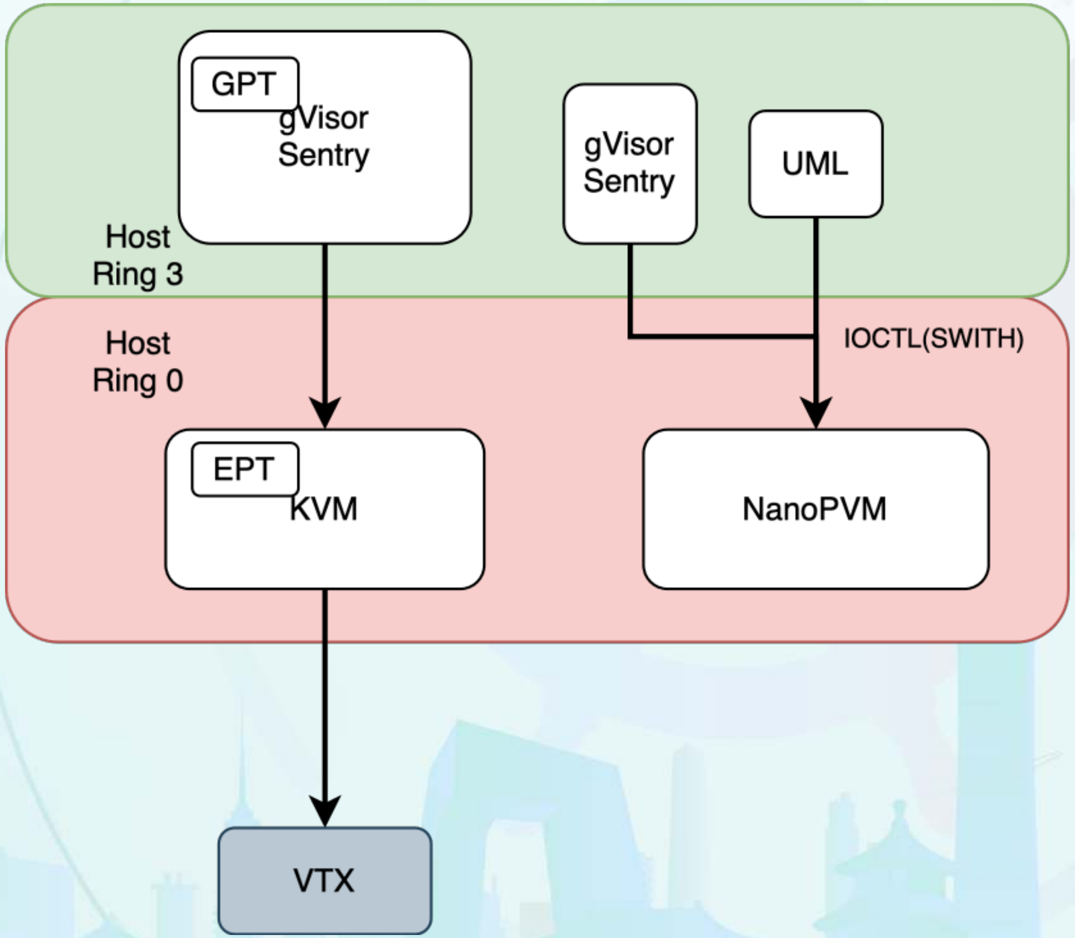
- Provide 2 Virtualization ability: **Context Execution, Address Space Management**
- Implementation with 3 Control Interfaces : **Map, Unmap, Switch**

OS Interface Complete

OS Function	Implement method	Interface
Create Process	Allocate address space and setup initial process status	map + switch
Process Exit	Release all memory mappings and resource	unmap
Memory Manage	Map/Unmap mapping on demand	map + unmap
Syscall Handle	Trap and handle after VM-Exit	switch
Exception Handle	Trap and handle after VM-Exit	switch
Device IO	Map device memory or port to host	map + switch

Secondary Mode

- Direct Memory Management via Map and Unmap.
- Intercept Syscall and Exception to User Mode Kernel.

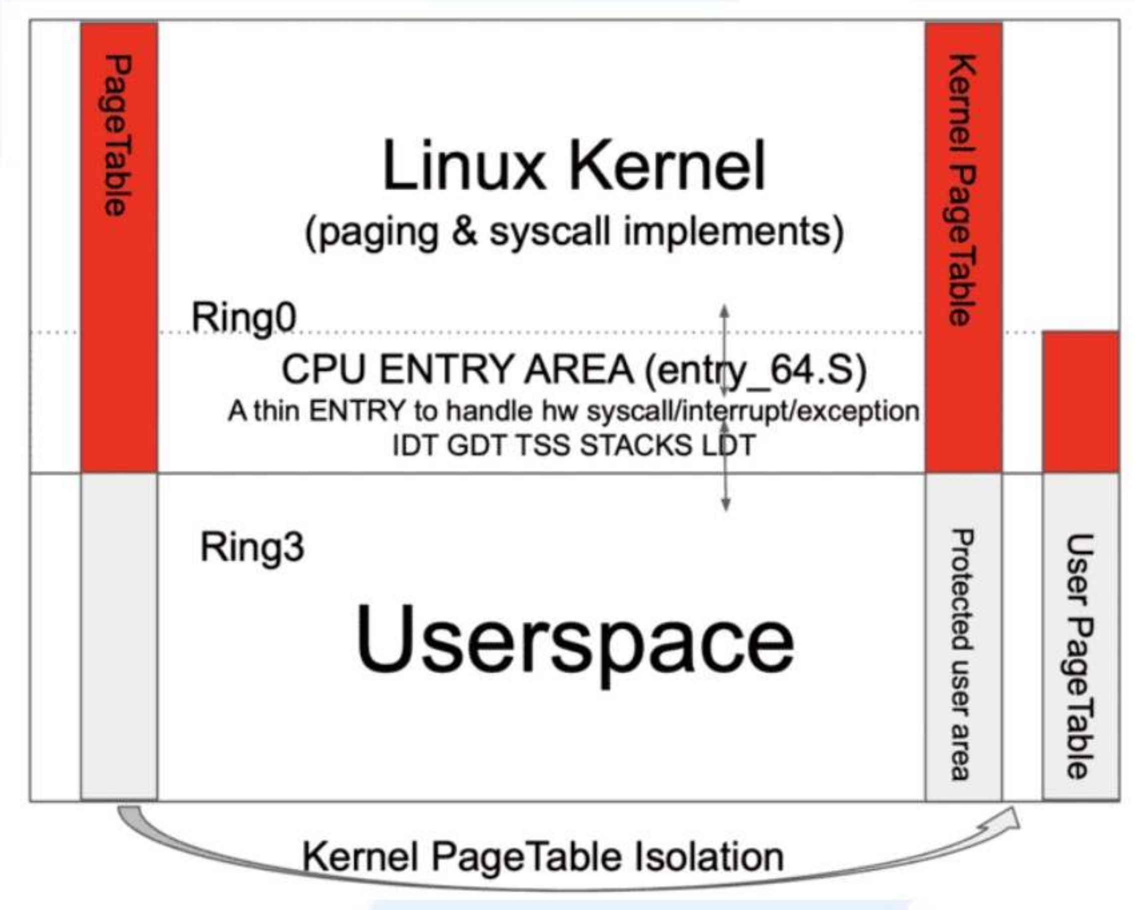


Interface and Usage example

```
/* Create compute instance: */

instance_fd = open("/dev/nanopvm");

as_fd = ioctl(instance_fd, NANOPVM_AS_CREATE);

ioctl(asfd, NANOPVM_MAP, *mem_range);


/* Virtualization loop: */

do {

    ioctl(asfd, NANOPVM_SWITCH, *context_regs);

    ...

    handle_vm_exit();

} while(1);


/* Process exit: */

ioctl(asfd, NANOPVM_UNMAP, *mem_range);
```

Syscall/Exception Intercept

- Intercept without Ptrace, but **hook the trampoline entry**.
- Provide “ Executable ” context, and 4 capabilities:
  - enter vm
  - force kick out (bounce)
  - exception exit
  - syscall exit
- With the help of **Switcher**



PageTable
Linux Kernel
(paging & syscall implements)
Ring0
CPU ENTRY AREA (entry_64.S)
A thin ENTRY to handle hw syscall/interrupt/exception
IDT GDT TSS STACKS LDT
Ring3
Userspace
Kernel PageTable
Protected user area
User PageTable
Kernel PageTable Isolation

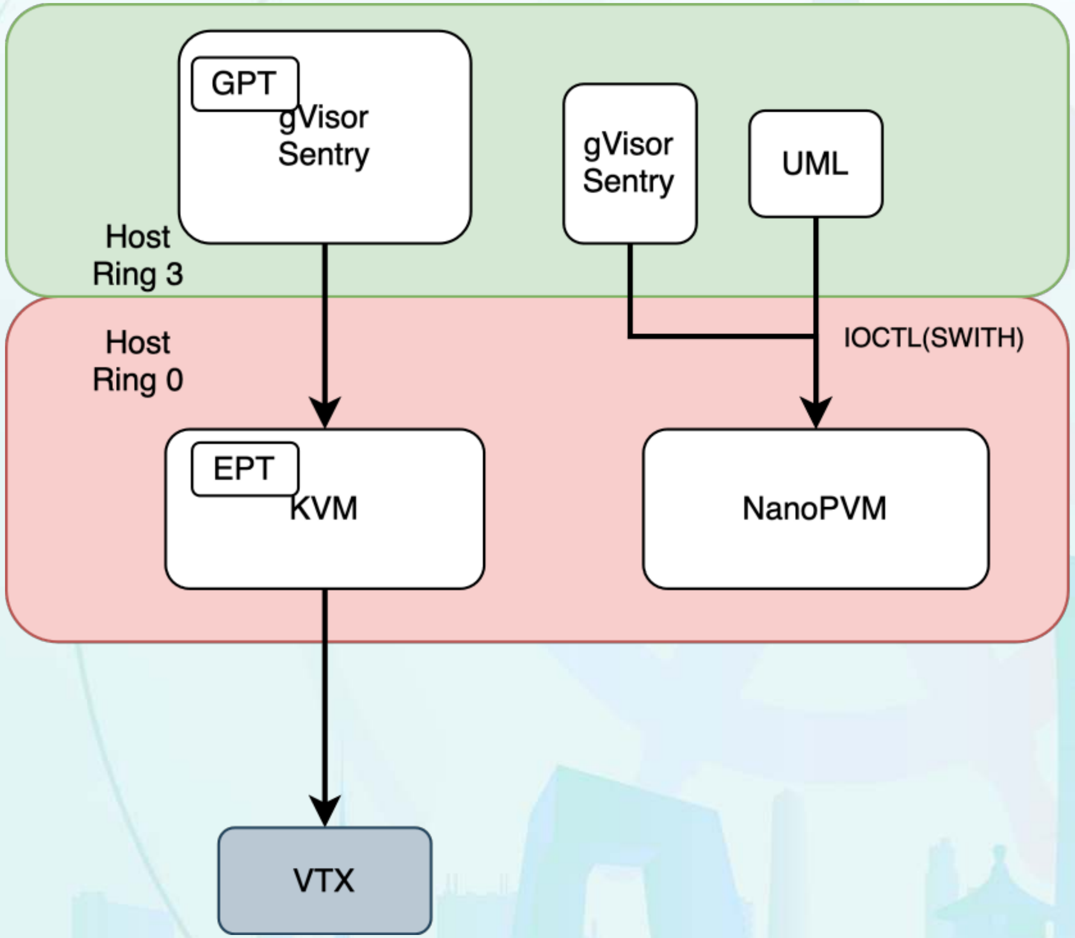
Secondary Mode Switch

- Called with context_switch by UML on purpose
- Call with Guest_Context_Regs {

    // fields similar to ptrace_regs

    ……

}

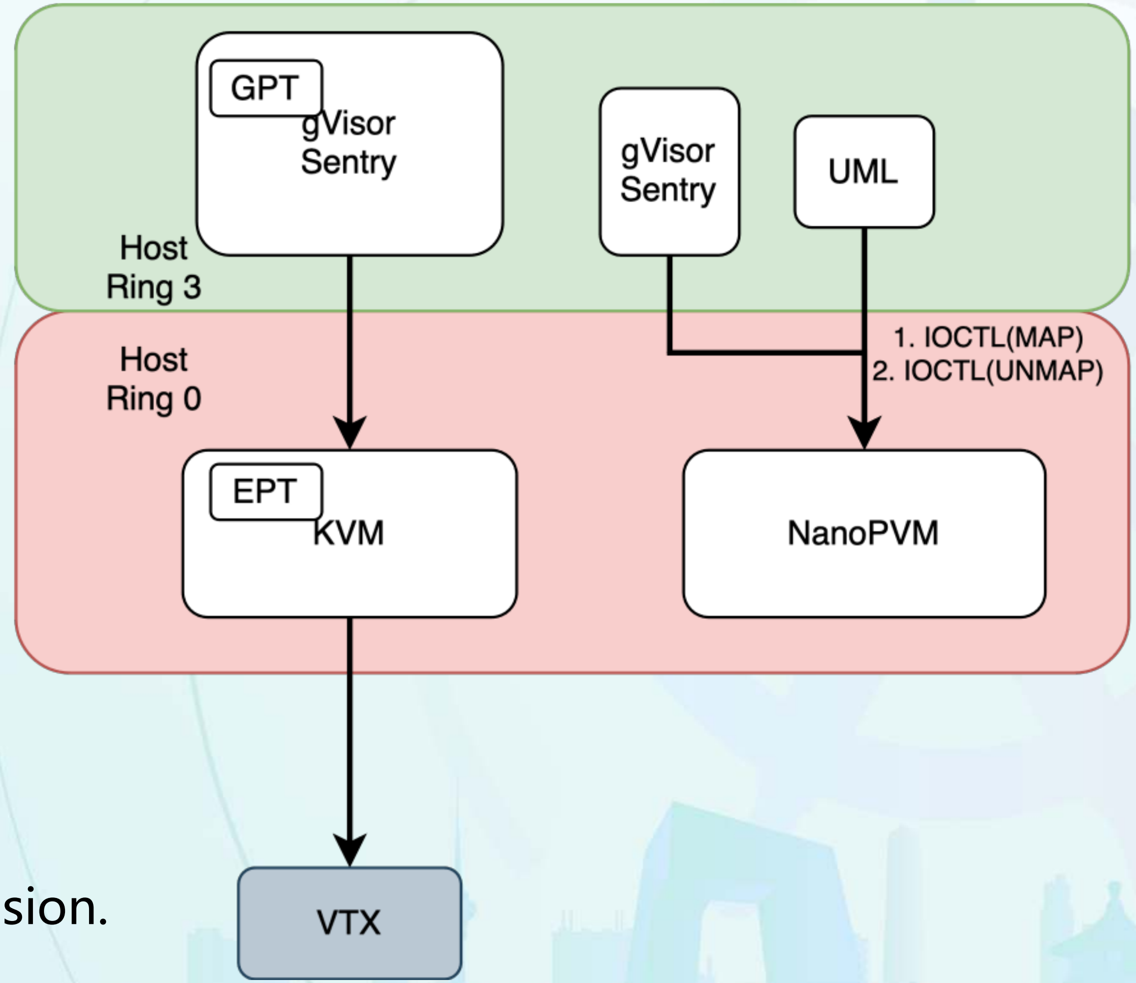


Memory Management

- Called from mmap, brk, handle_user_fault and so on.
- Call with Guest_Memory_Range {

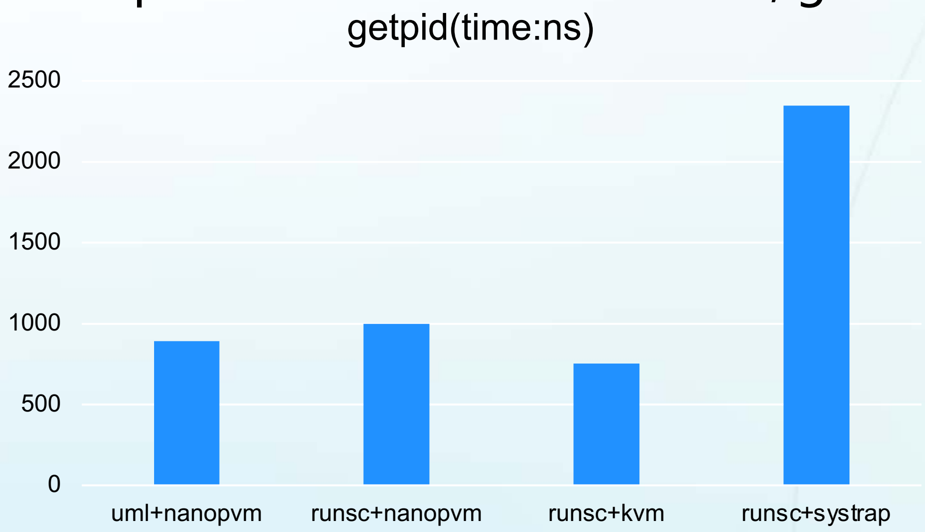
```
        void *gva;
        void *hva;
        uint len;
        uint prot;
        ……
}
```

- **No conception of GPT** at all.
- All Guest address mapped with User permission.
- Memory region Management with rbTree.

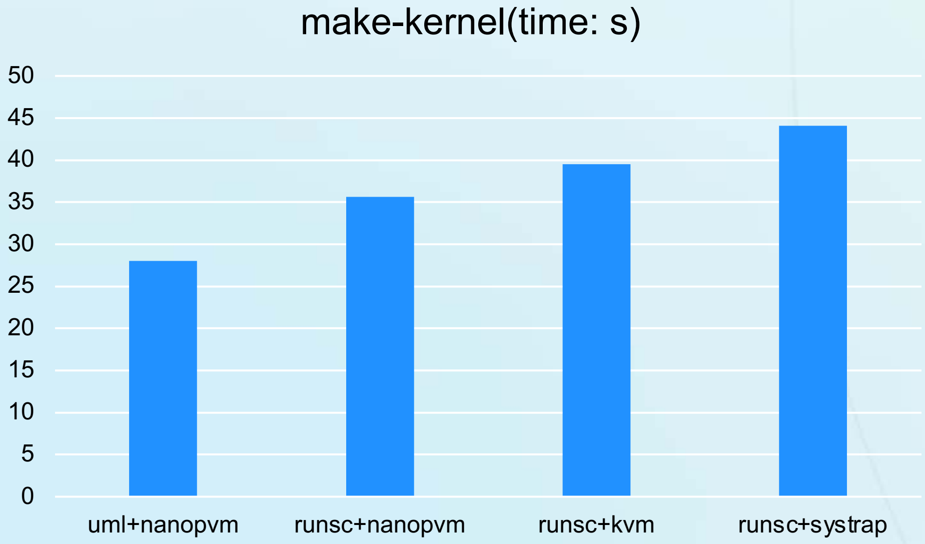


Performance

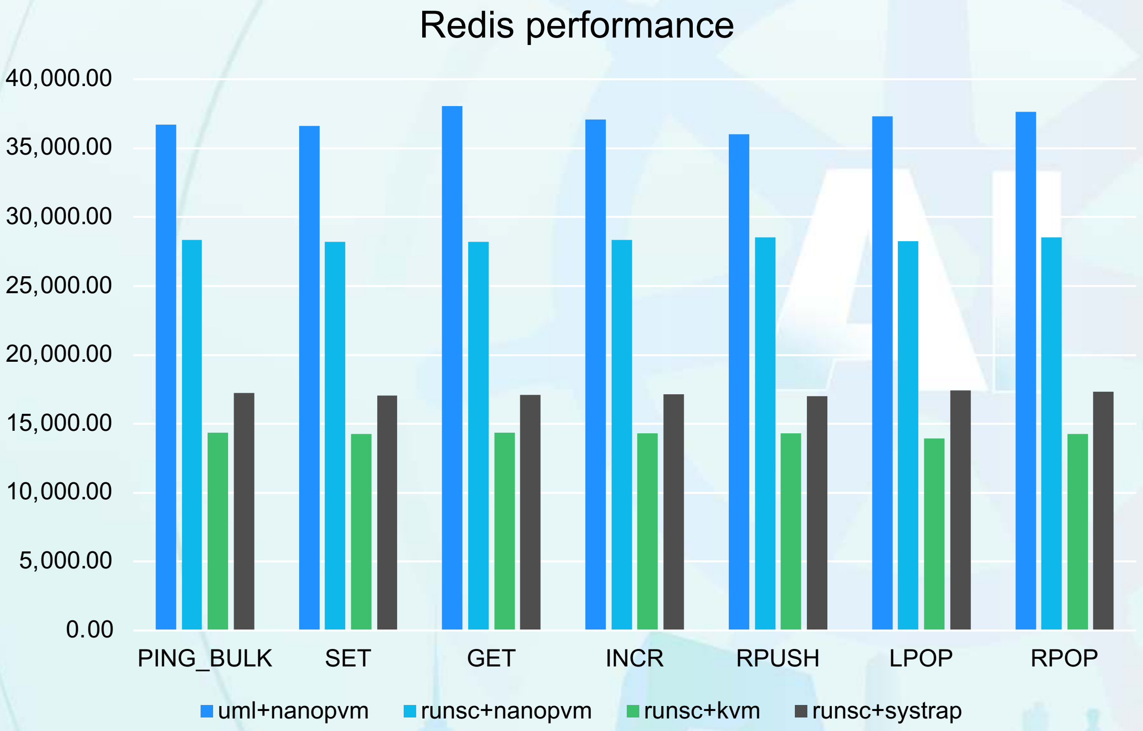
Compared with UML+NanoPVM, gVisor+NanoPVM, gVisor+KVM, gVisor-systrap



getpid(time:ns)



make-kernel(time: s)



Redis performance

Future

- The basement of Next Security Container.
- Alternative base tech for other kind of Sandbox, e.g. WebAssembly Runtime.
- (Maybe) Remove the GPT from UML.

Thanks and QA