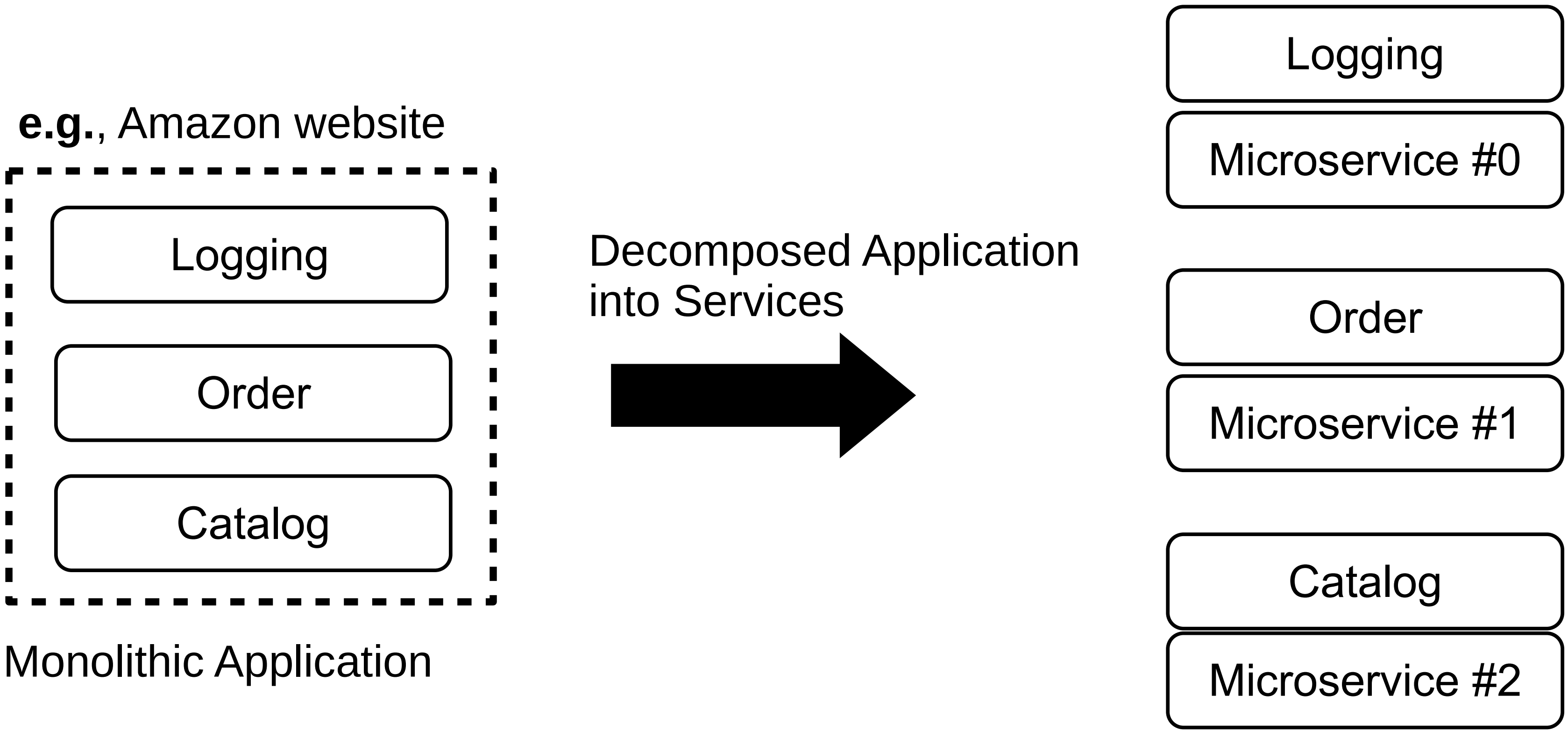
ToroV, a kernel in user-space, or sort of

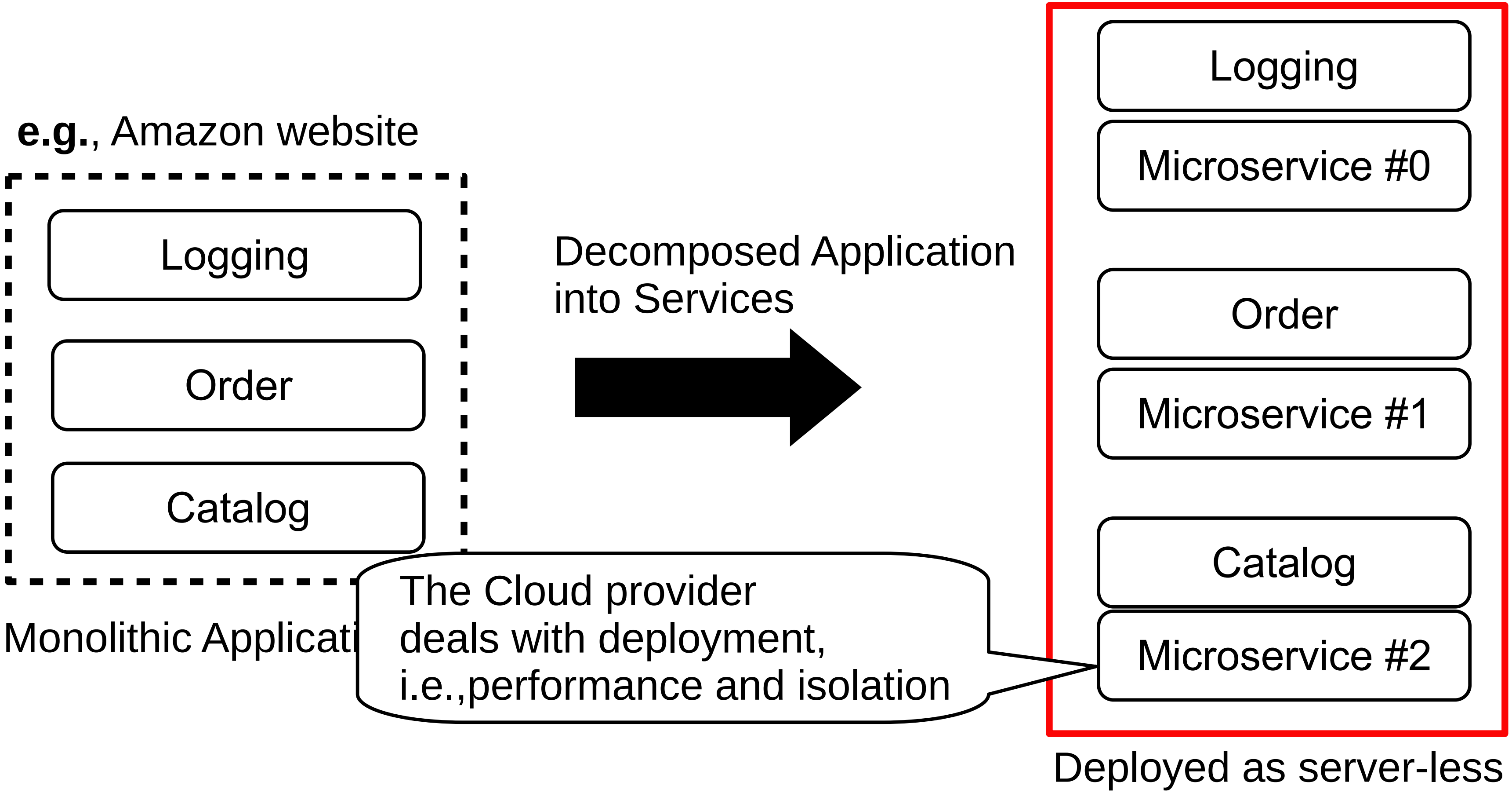
www.torokernel.io

Matias Vara Larsen
matiasevara@torokernel.io

Who am I?

- I enjoy working on operating systems and playing with virtualization
- I worked at Citrix, Tttech, Huawei ...
- https://github.com/MatiasVara


e.g., Amazon website
Logging
Order
Catalog
Monolithic Application
Decomposed Application into Services
Logging
Microservice #0
Order
Microservice #1
Catalog
Microservice #2

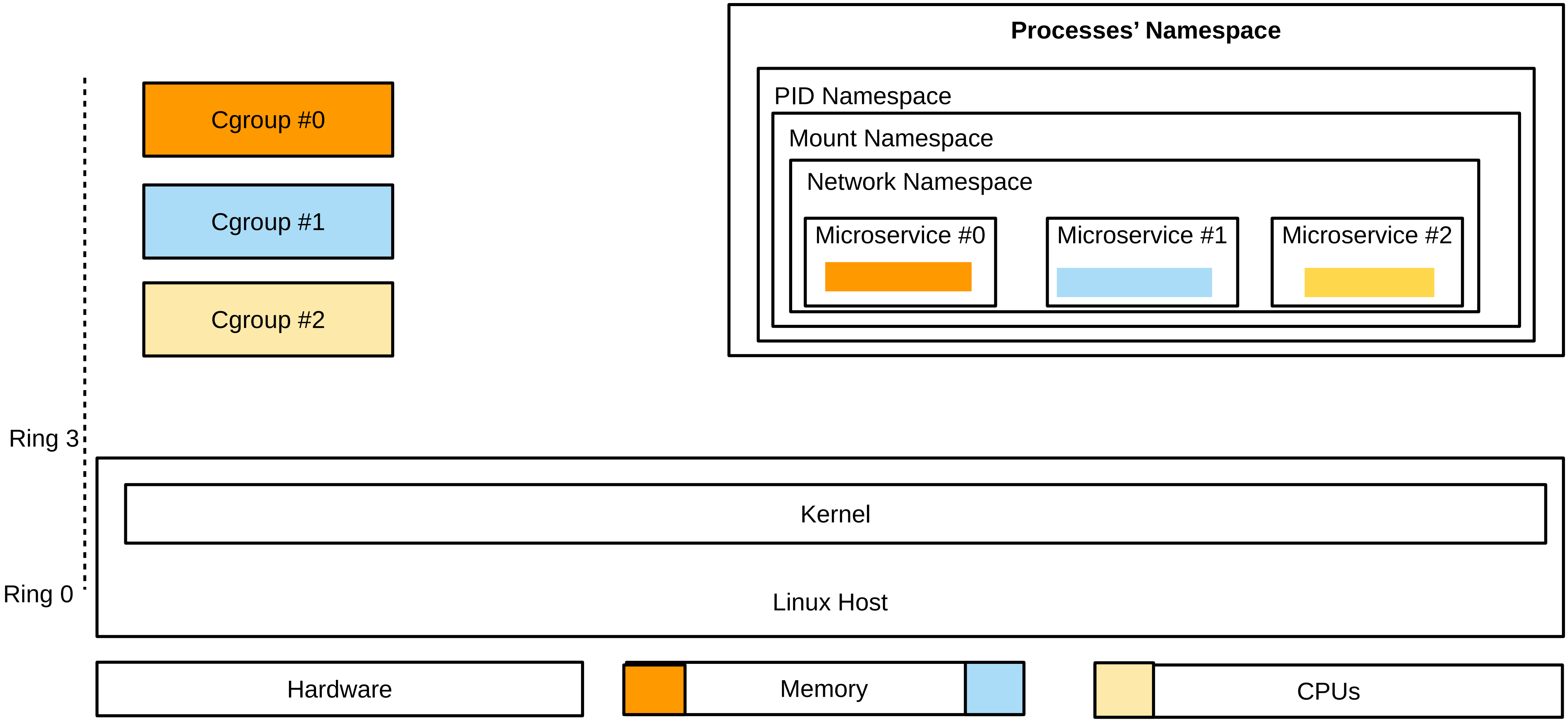


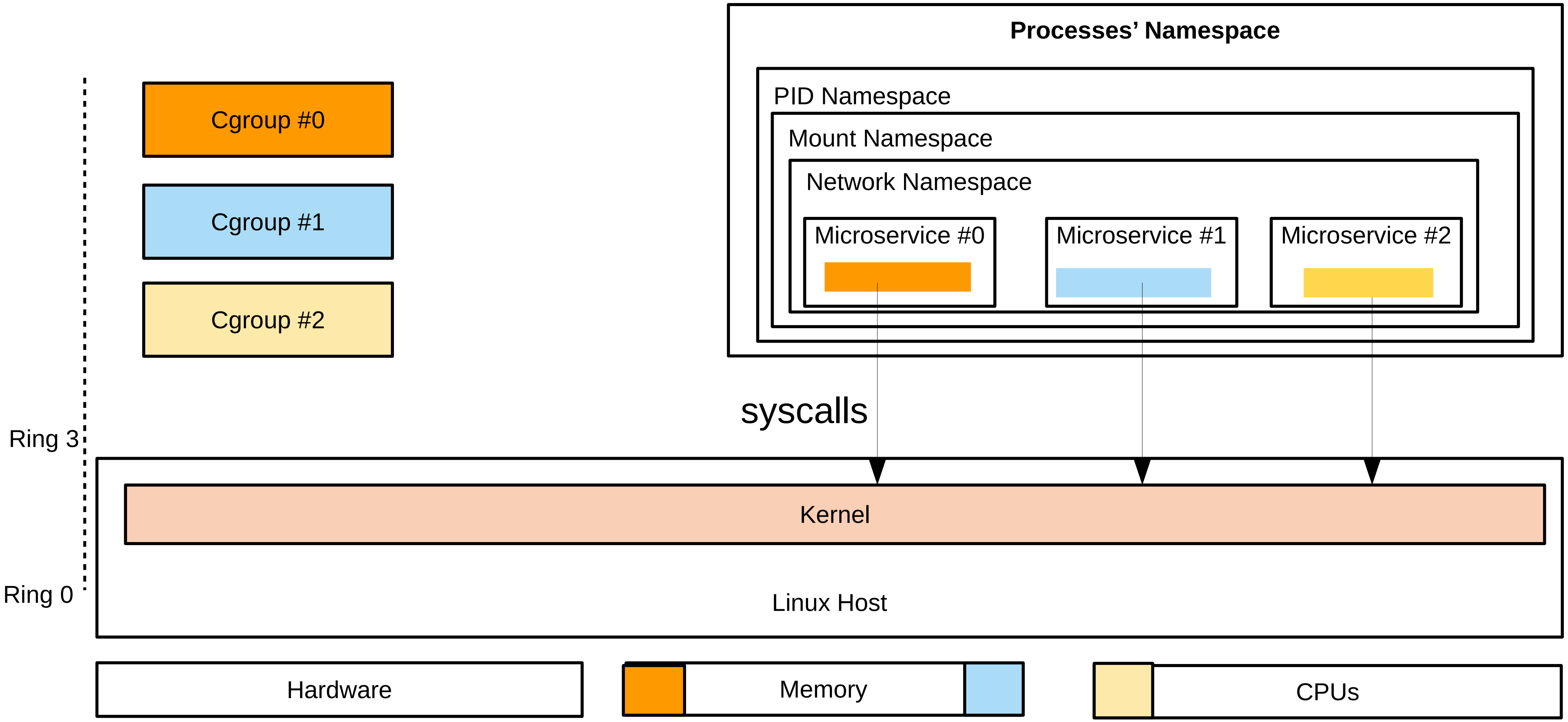
e.g., Amazon website
Logging
Order
Catalog
Monolithic Applicati
Decomposed Application into Services
The Cloud provider deals with deployment, i.e.,performance and isolation
Logging
Microservice #0
Order
Microservice #1
Catalog
Microservice #2
Deployed as server-less

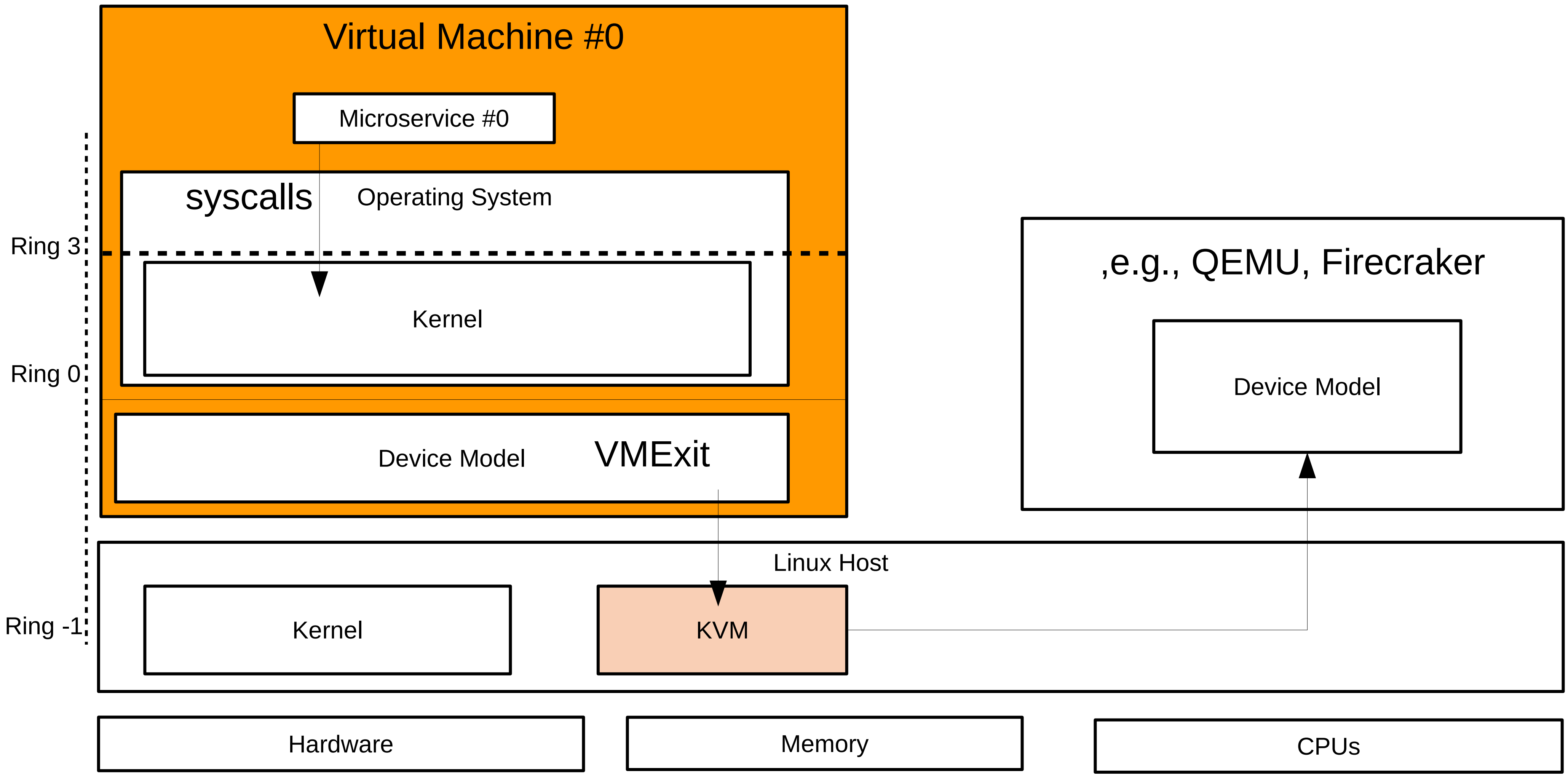
How server-less applications can be deployed?

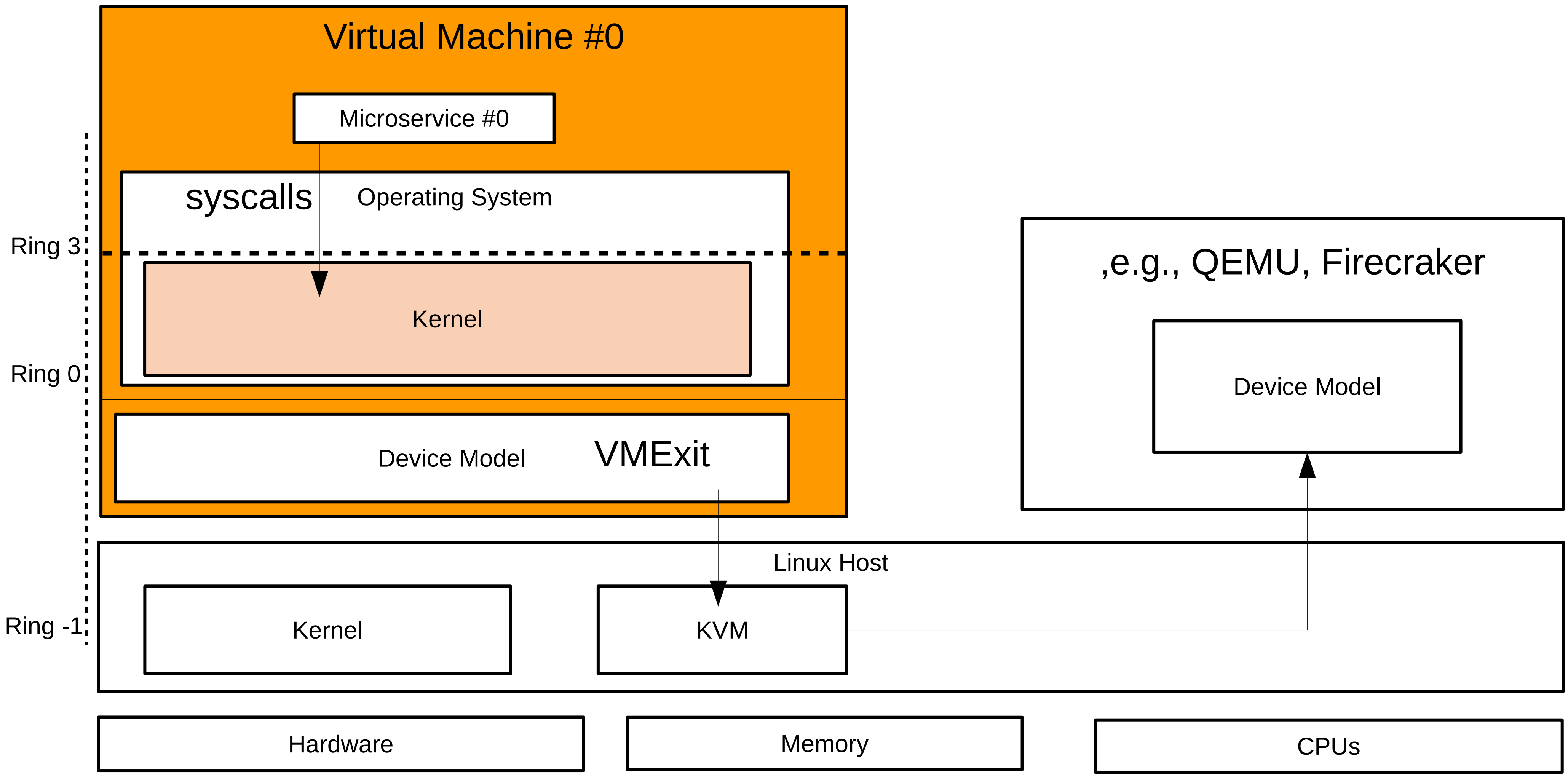
- By using containers, i.e., software-based virtualization
- By using VMs, i.e., hardware-based virtualization
  - General Purpose OS
  - unikernel

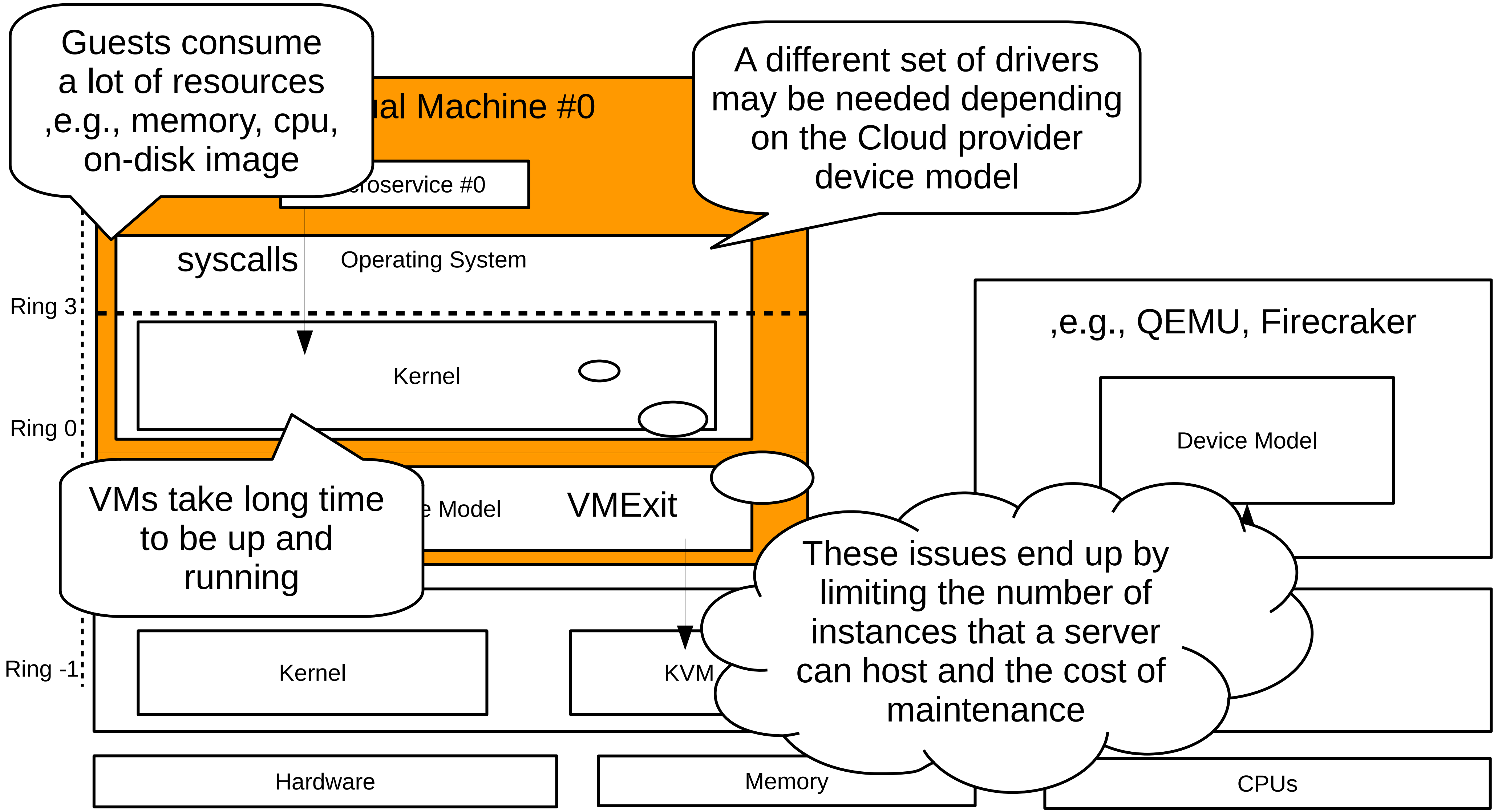
- These mechanisms are chosen based on a trade-off between performance and security


Processes’ Namespace
PID Namespace
Mount Namespace
Network Namespace
Microservice #0
Microservice #1
Microservice #2
Cgroup #0
Cgroup #1
Cgroup #2
Ring 3
Ring 0
Kernel
Linux Host
Hardware
Memory
CPUs

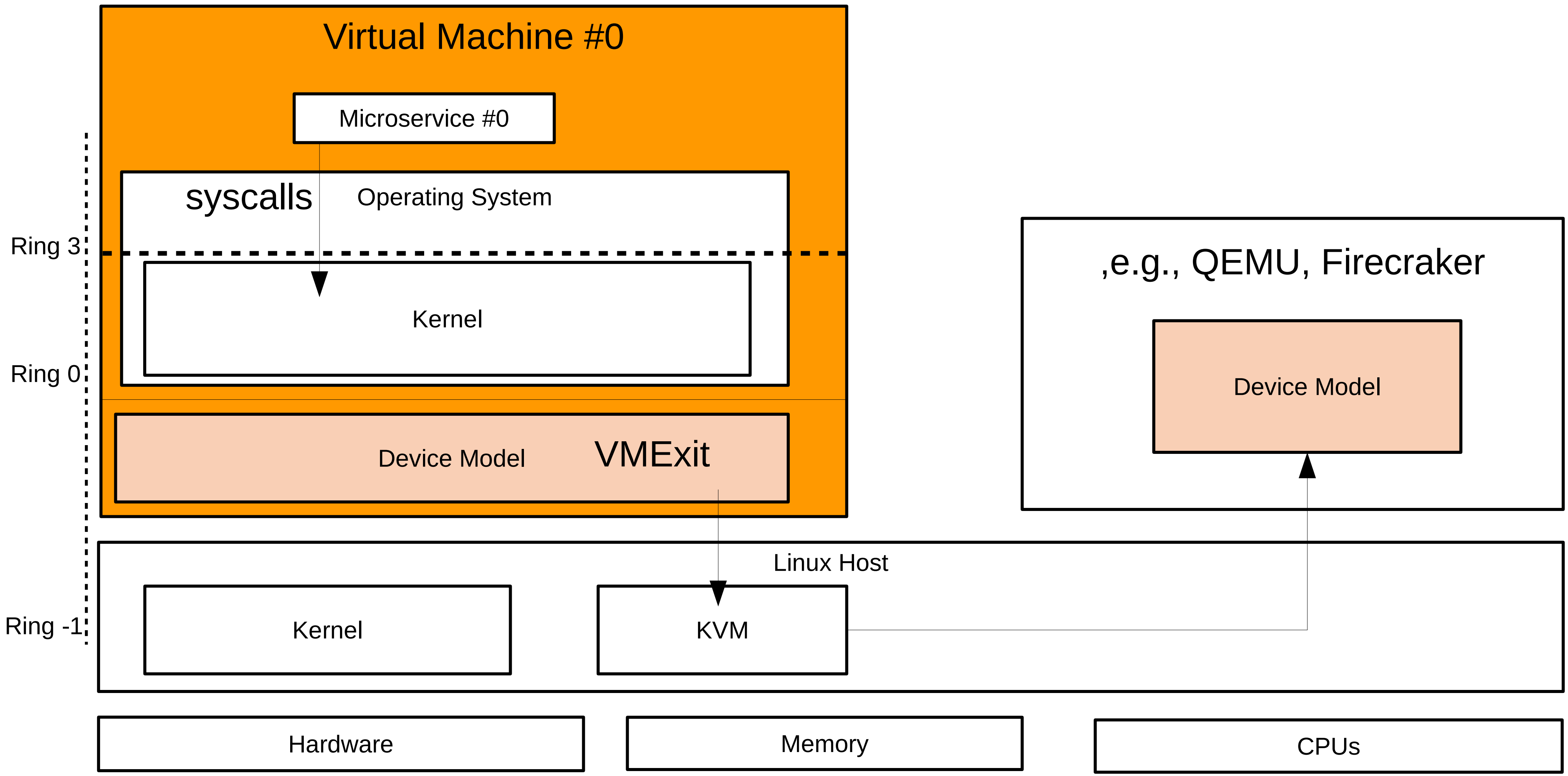

Processes’ Namespace
PID Namespace
Mount Namespace
Network Namespace
Microservice #0
Microservice #1
Microservice #2
Cgroup #0
Cgroup #1
Cgroup #2
syscalls
Ring 3
Ring 0
Kernel
Linux Host
Hardware
Memory
CPUs

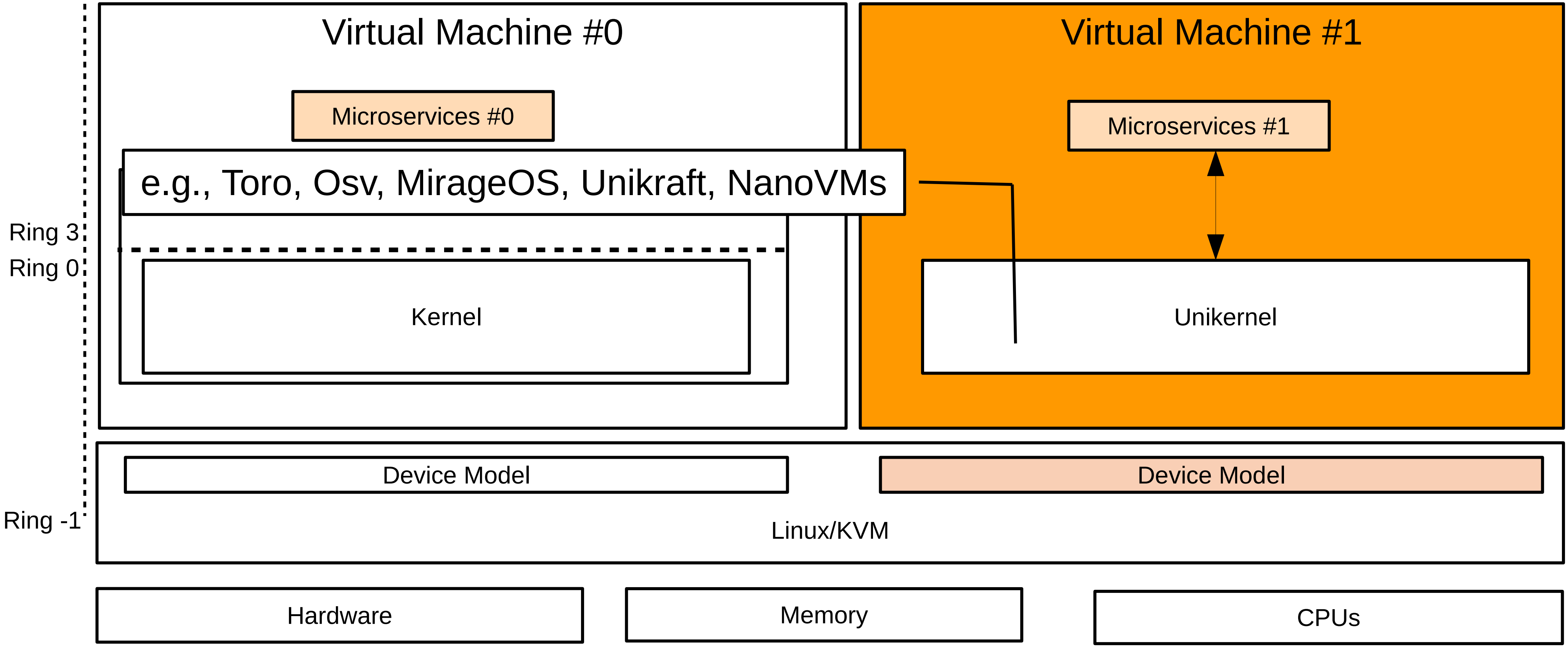

Virtual Machine #0
Microservice #0
syscalls
Operating System
Ring 3
Kernel
Ring 0
Device Model
VMExit
,e.g., QEMU, Firecraker
Device Model
Linux Host
Kernel
KVM
Ring -1
Hardware
Memory
CPUs


Virtual Machine #0
Microservice #0
syscalls
Operating System
Ring 3
Kernel
Ring 0
Device Model
VMExit
,e.g., QEMU, Firecraker
Device Model
Linux Host
Kernel
KVM
Ring -1
Hardware
Memory
CPUs

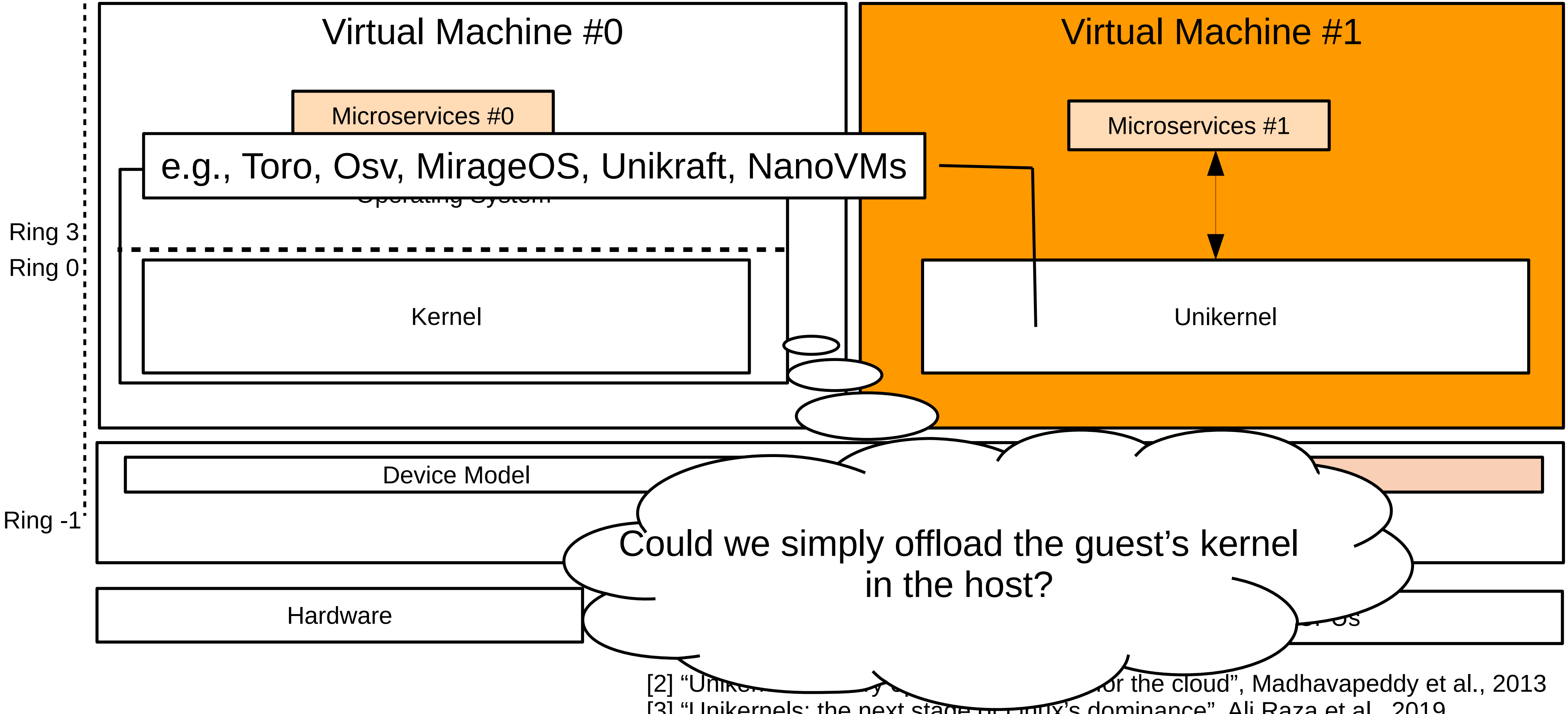


Guests consume a lot of resources ,e.g., memory, cpu, on-disk image
...ual Machine #0
...croservice #0
A different set of drivers may be needed depending on the Cloud provider device model
syscalls
Operating System
Ring 3
Kernel
Ring 0
VMs take long time to be up and running
...e Model
VMExit
,e.g., QEMU, Firecraker
Device Model
These issues end up by limiting the number of instances that a server can host and the cost of maintenance
Ring -1
Kernel
KVM
Hardware
Memory
CPUs


Virtual Machine #0
Microservice #0
syscalls
Operating System
Kernel
Device Model
VMExit
,e.g., QEMU, Firecraker
Device Model
Ring 3
Ring 0
Ring -1
Linux Host
Kernel
KVM
Hardware
Memory
CPUs



[2] “Unikernels: library operating systems for the cloud”, Madhavapeddy et al., 2013
[3] “Unikernels: the next stage of Linux’s dominance”, Ali Raza et al., 2019

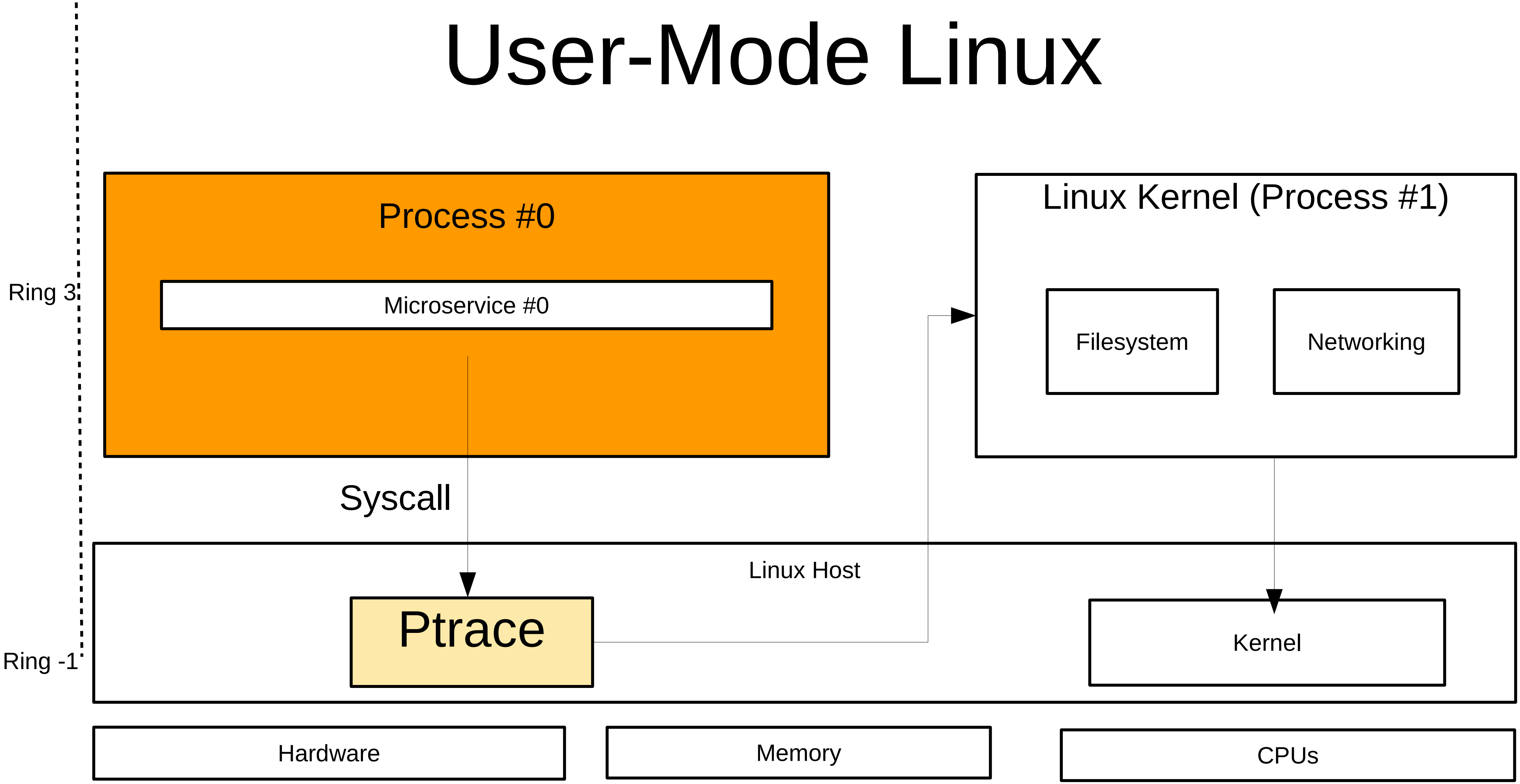


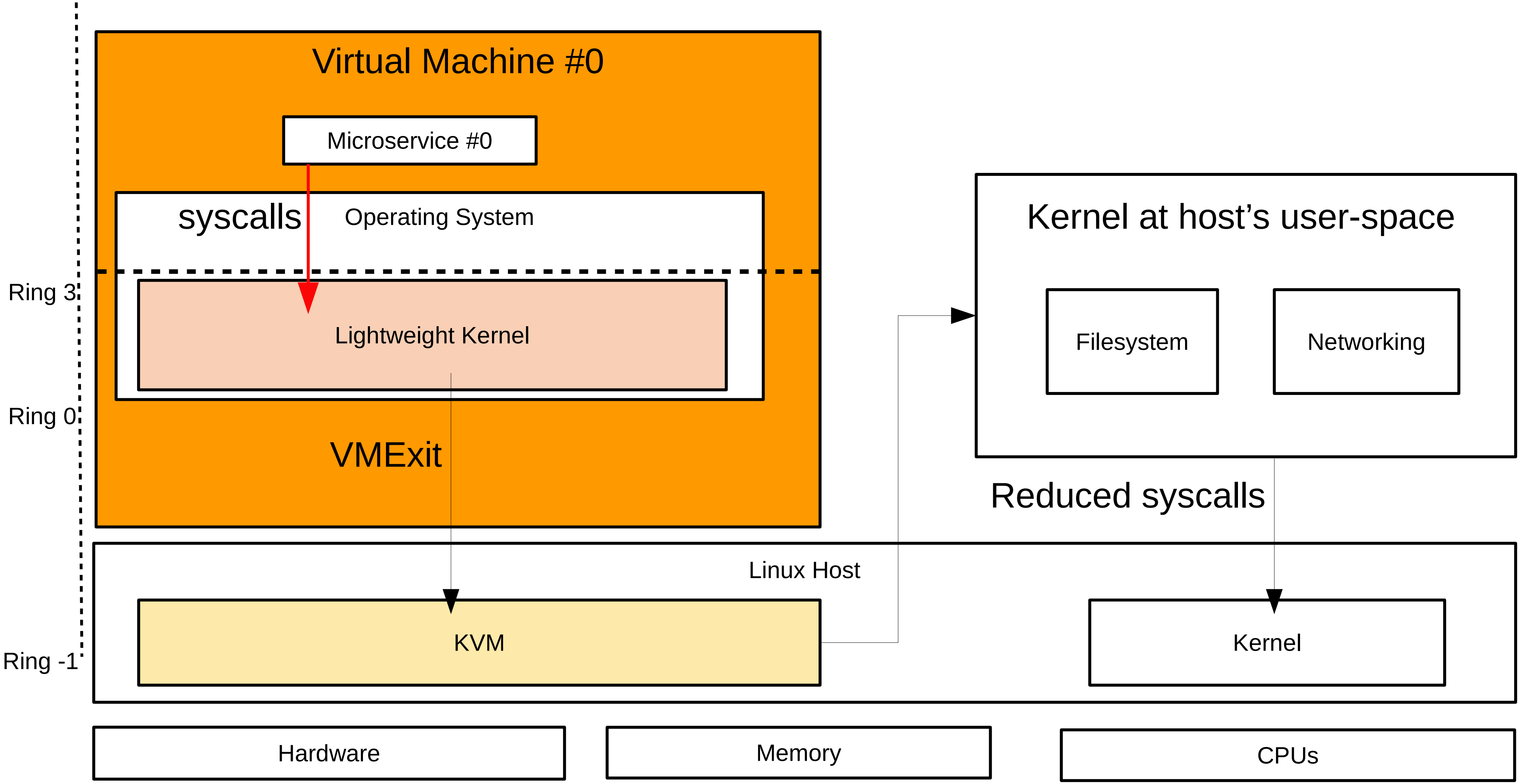
[2] “Unike... ...or the cloud”, Madhavapeddy et al., 2013
[3] “Unikernels: the next stage of Linux’s dominance”, Ali Raza et al., 2019

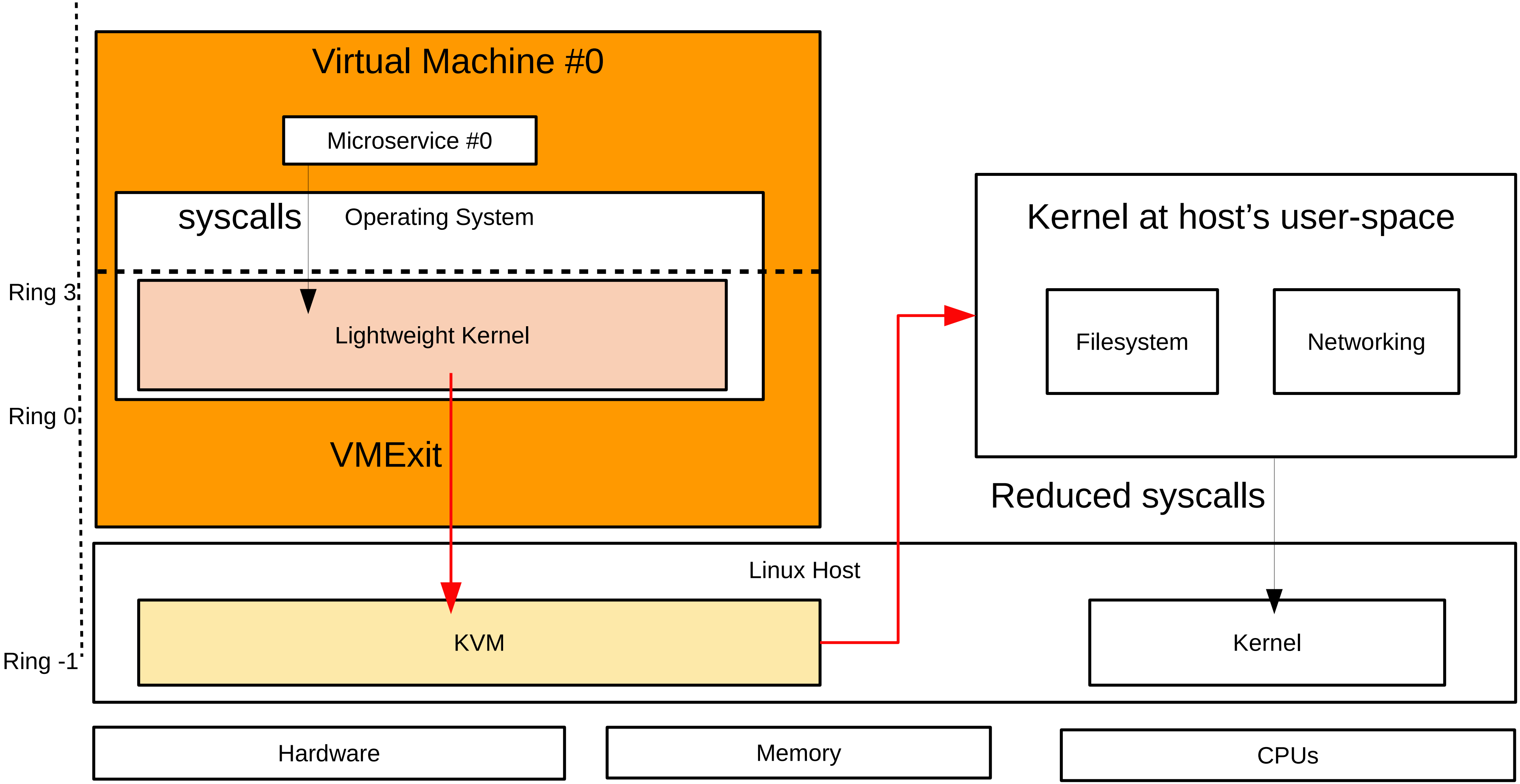
What is a kernel in user’s space?

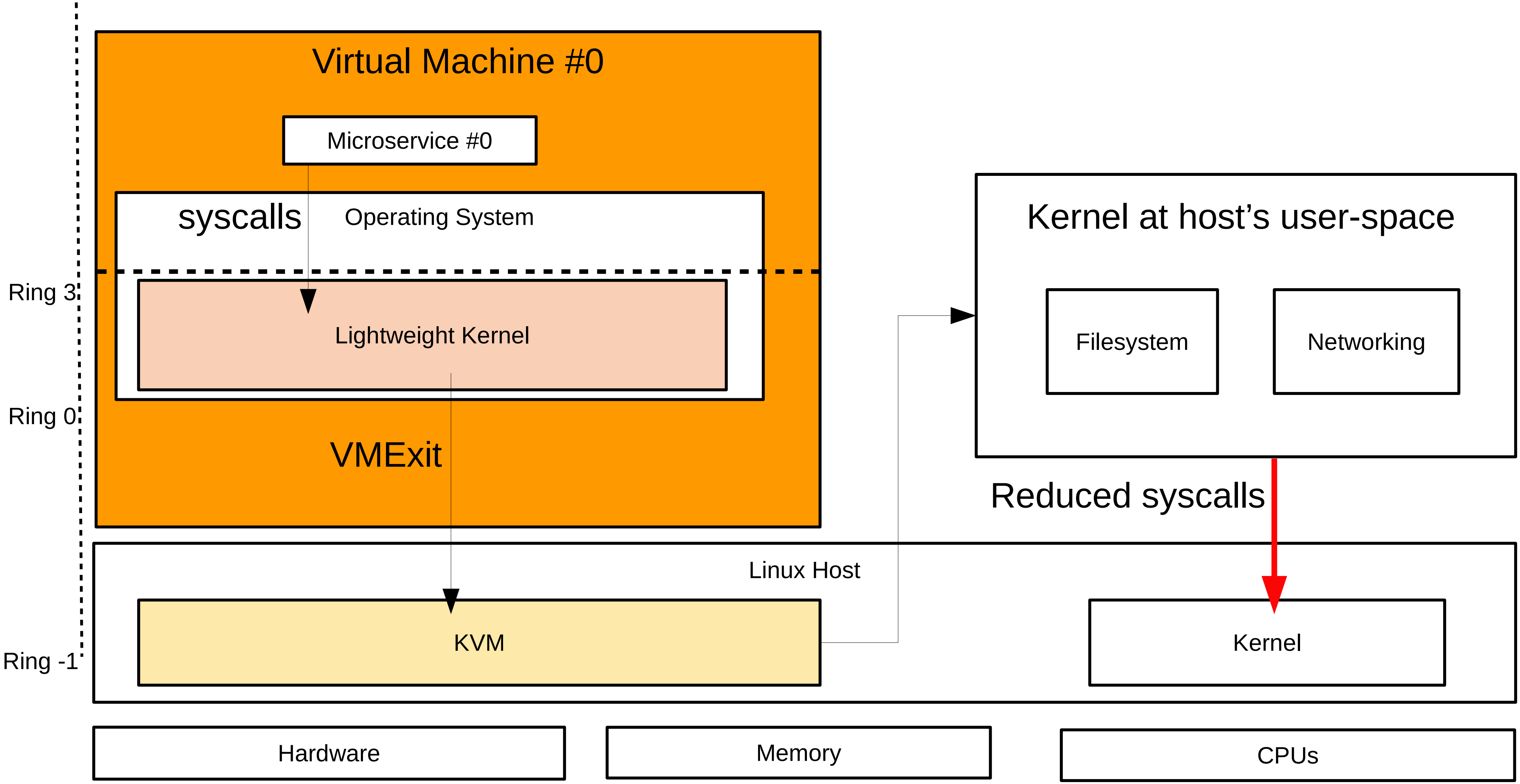
- It is a way to offload guest’s kernel in the host
- It allows a guest application to run without a device model thus reducing the attack surface of the host
- It prevents the host to be exposed by emulating kernel services.
- Approaches:
  - User-Mode Linux
  - Gvisor
  - ToroV

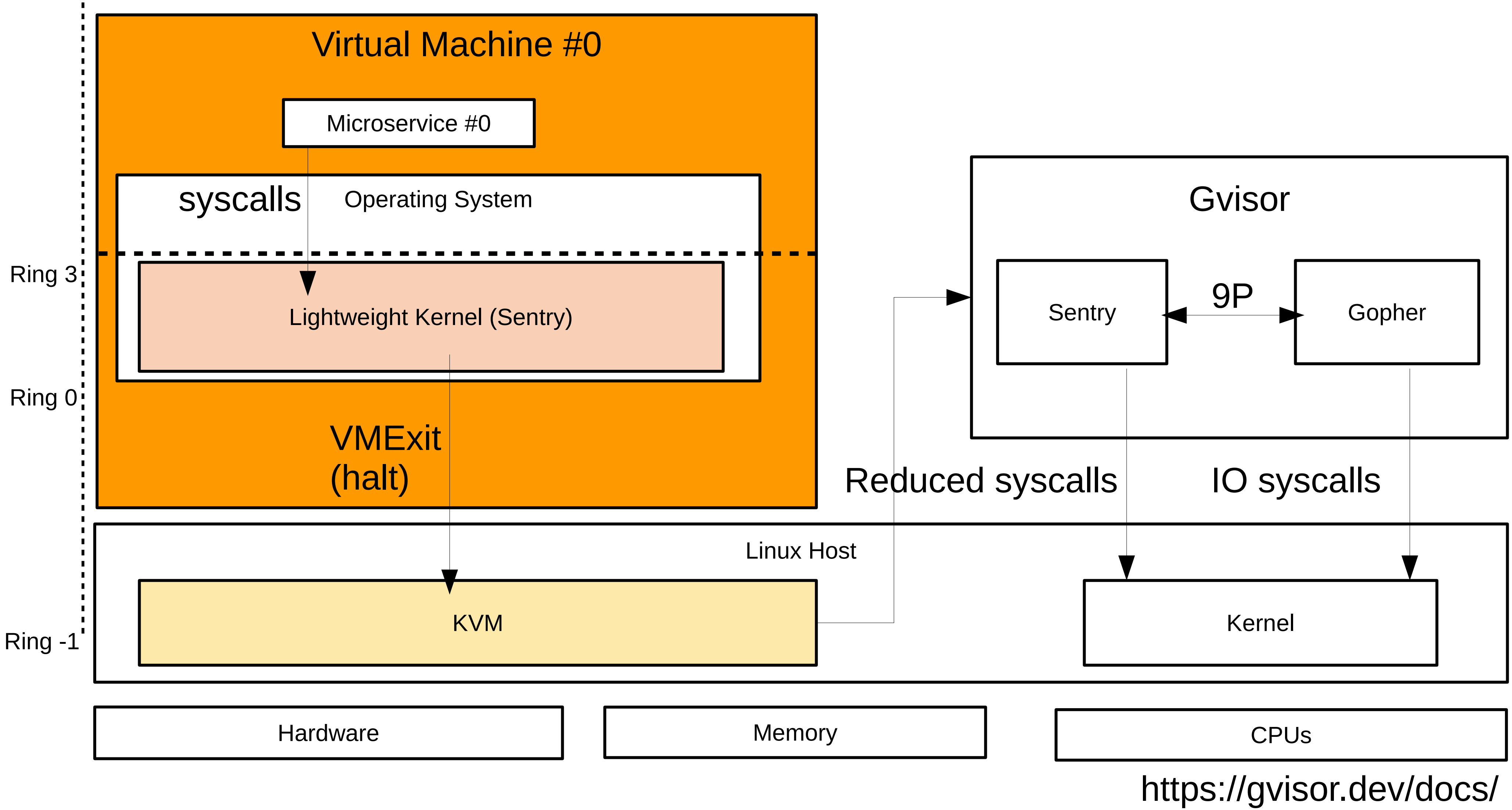
User-Mode Linux

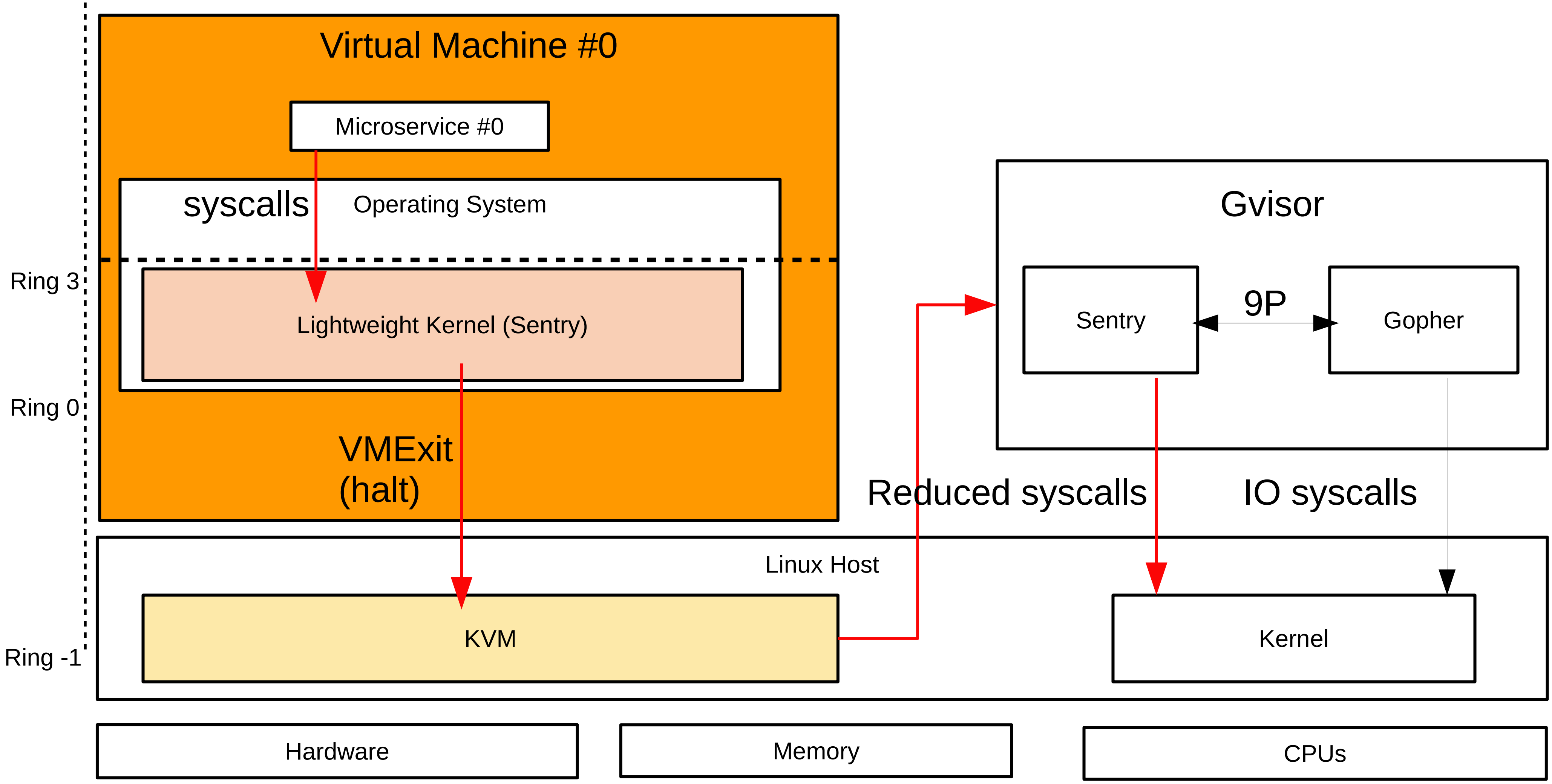



Virtual Machine #0
Microservice #0
syscalls
Operating System
Ring 3
Lightweight Kernel
Ring 0
VMExit
Kernel at host’s user-space
Filesystem
Networking
Reduced syscalls
Linux Host
KVM
Kernel
Ring -1
Hardware
Memory
CPUs


Virtual Machine #0
Microservice #0
syscalls
Operating System
Lightweight Kernel
VMExit
Ring 3
Ring 0
Ring -1
Kernel at host’s user-space
Filesystem
Networking
Reduced syscalls
Linux Host
KVM
Kernel
Hardware
Memory
CPUs


Virtual Machine #0
Microservice #0
syscalls
Operating System
Lightweight Kernel
VMExit
Ring 3
Ring 0
Ring -1
Kernel at host's user-space
Filesystem
Networking
Reduced syscalls
Linux Host
KVM
Kernel
Hardware
Memory
CPUs

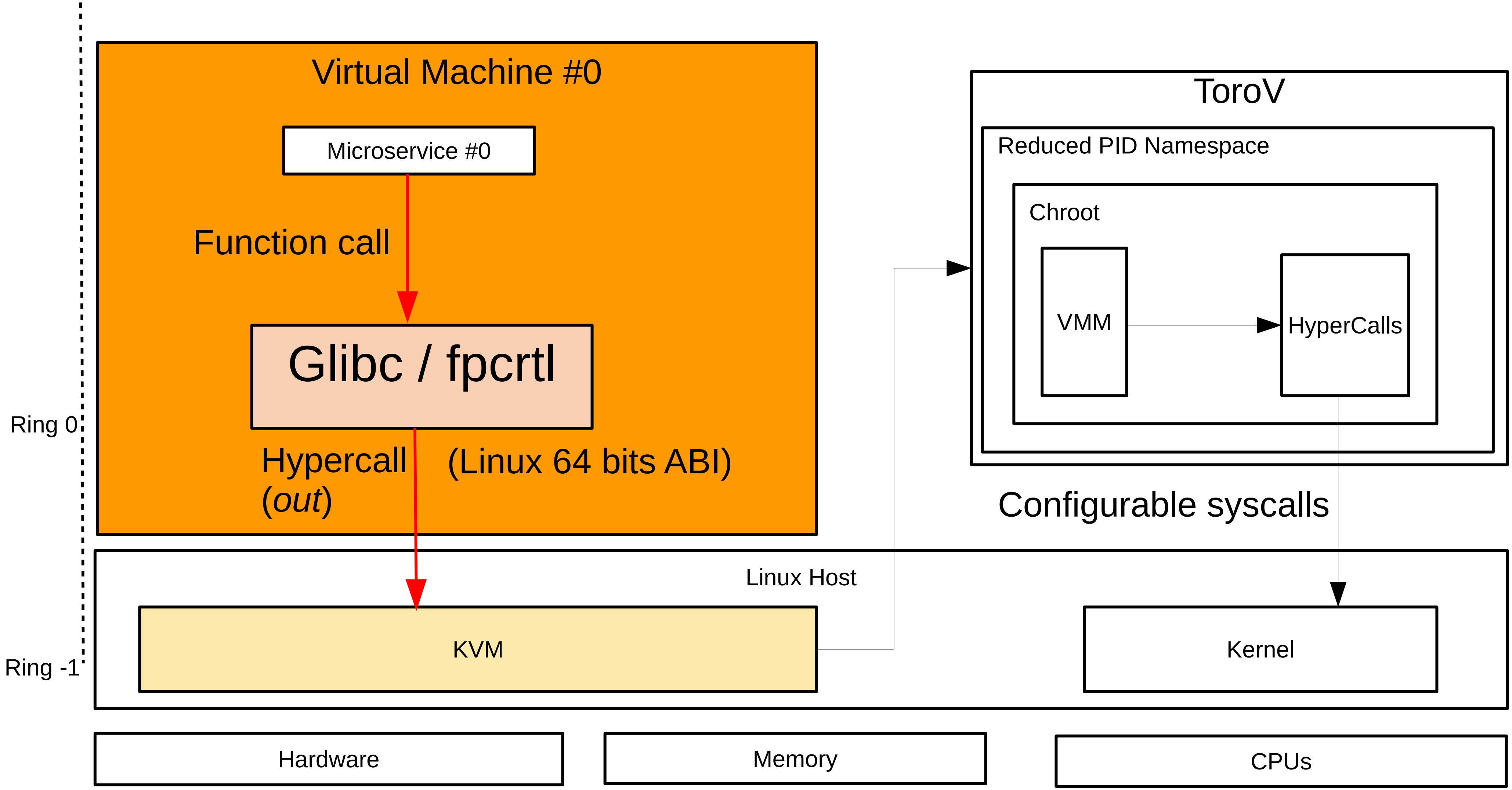

Virtual Machine #0
Microservice #0
syscalls
Operating System
Ring 3
Lightweight Kernel (Sentry)
Ring 0
VMExit
(halt)
Gvisor
Sentry
9P
Gopher
Reduced syscalls
IO syscalls
Linux Host
KVM
Kernel
Ring -1
Hardware
Memory
CPUs
https://gvisor.dev/docs/



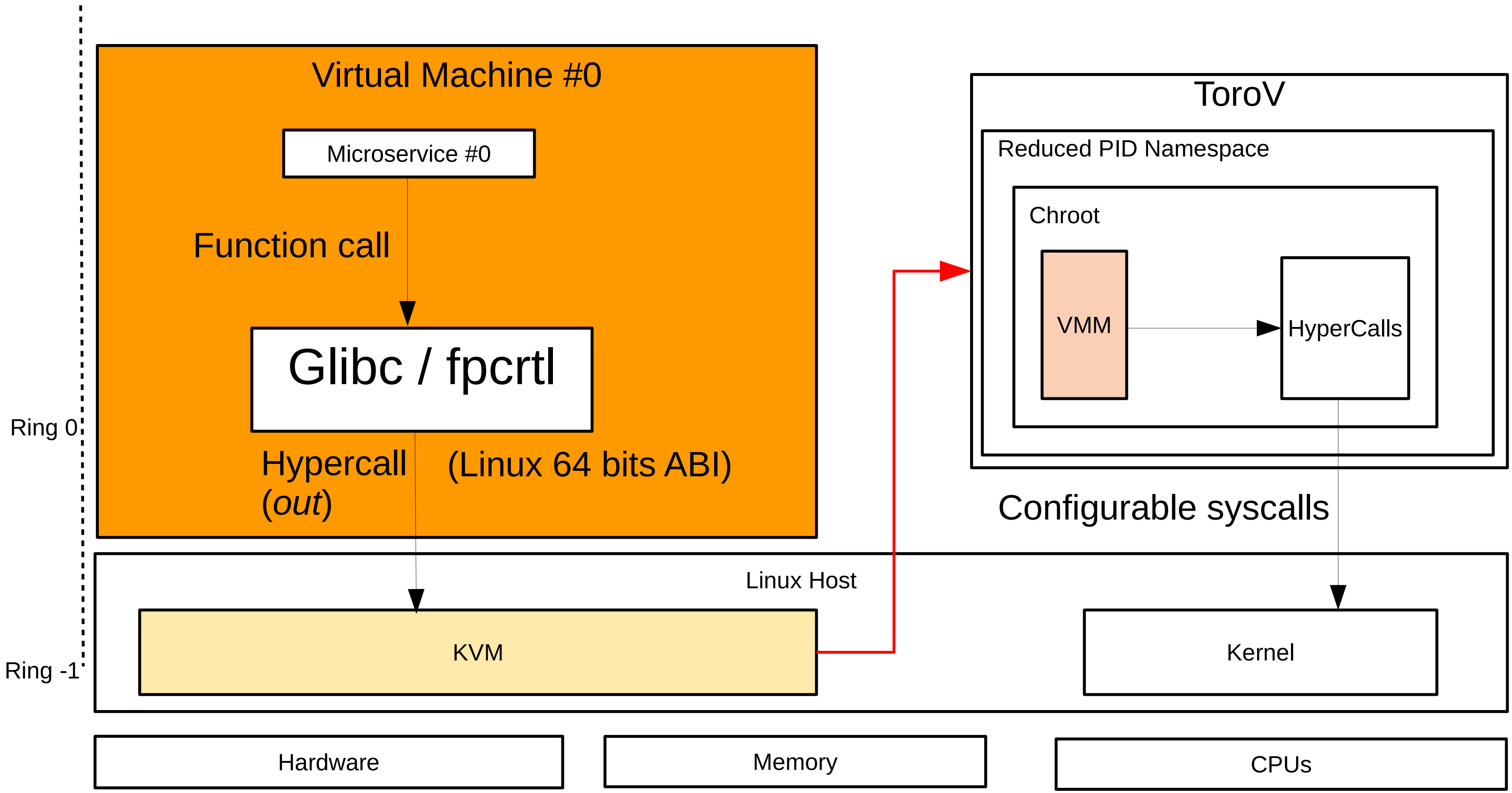
https://gvisor.dev/docs/

What is ToroV?

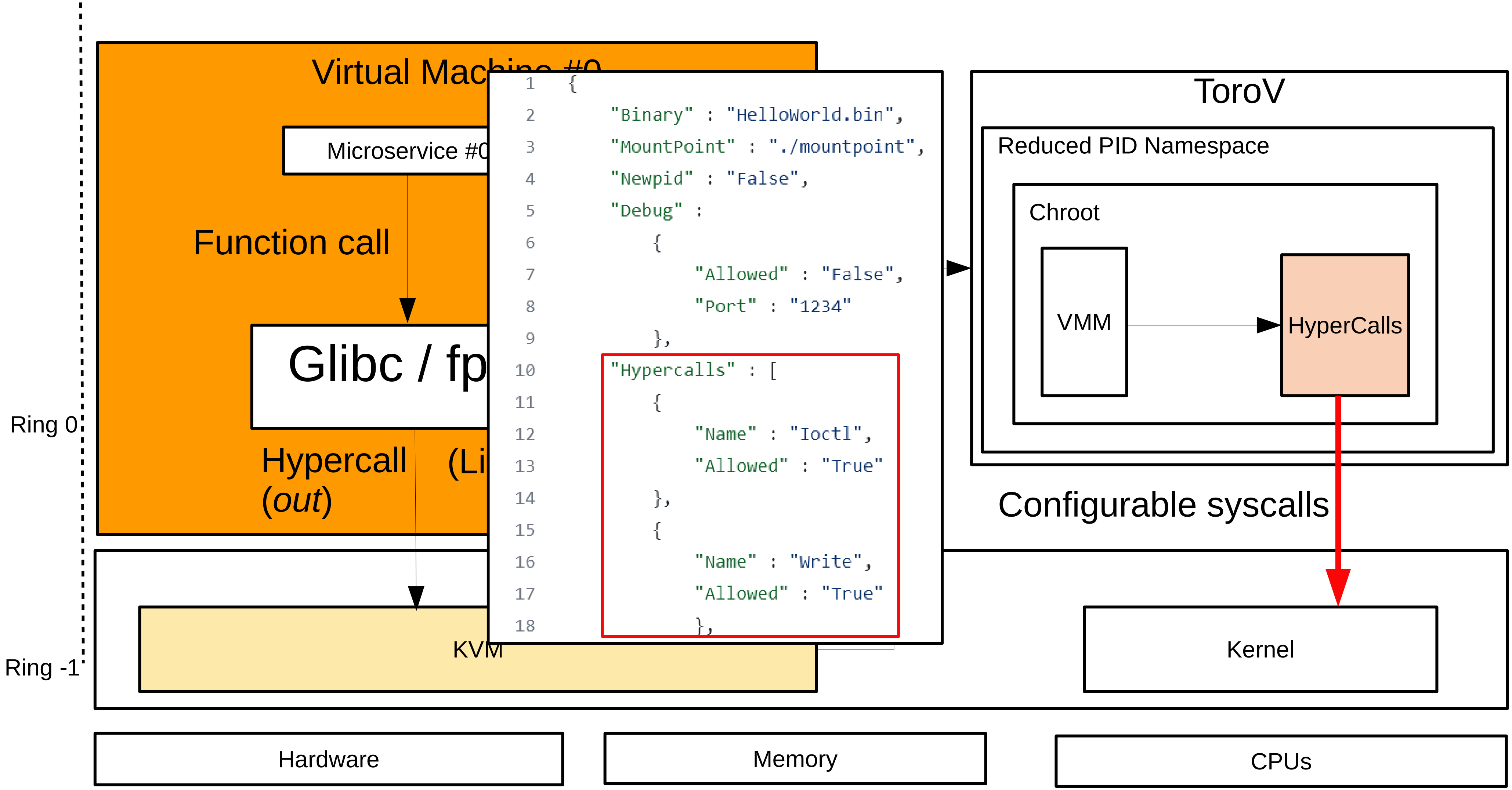
- It is a minimalist kernel in user space in which syscalls from the guest are forwarded to the host.
- It allows the user to configure what syscalls are allowed per application.
- It provides a modified stdlib that the user’s application must be compiled within.
- It exposes a POSIX API based on hypercalls to the guest.
- It runs as a containerized process to reduce host attack surface.
- It allows the user to debug guest’s applications by simply using a GDB.

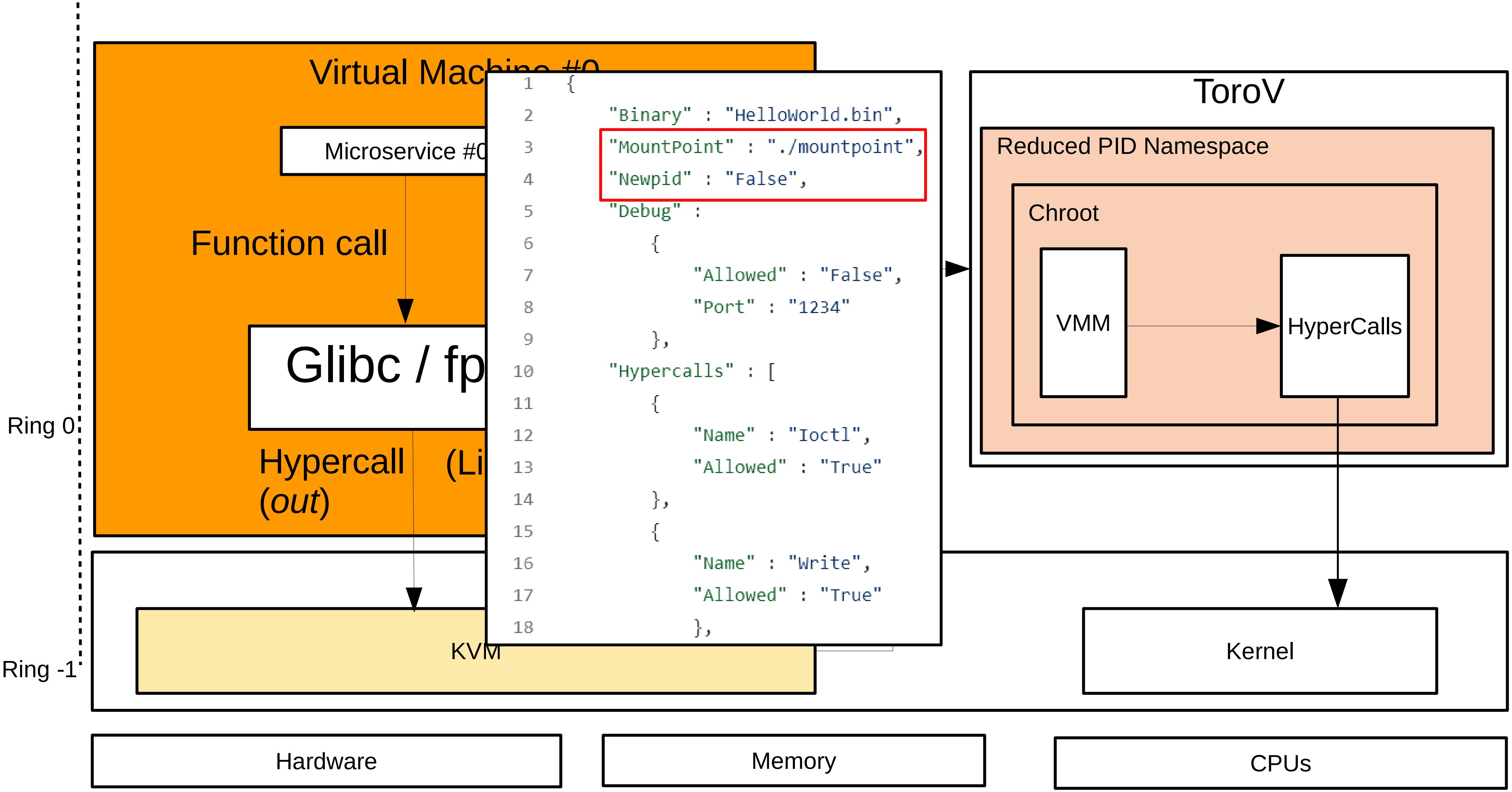


Virtual Machine #0
Microservice #0
Function call
Glibc / fpcrtl
Hypercall
(*out*)
(Linux 64 bits ABI)
ToroV
Reduced PID Namespace
Chroot
VMM
HyperCalls
Configurable syscalls
Ring 0
Ring -1
Linux Host
KVM
Kernel
Hardware
Memory
CPUs

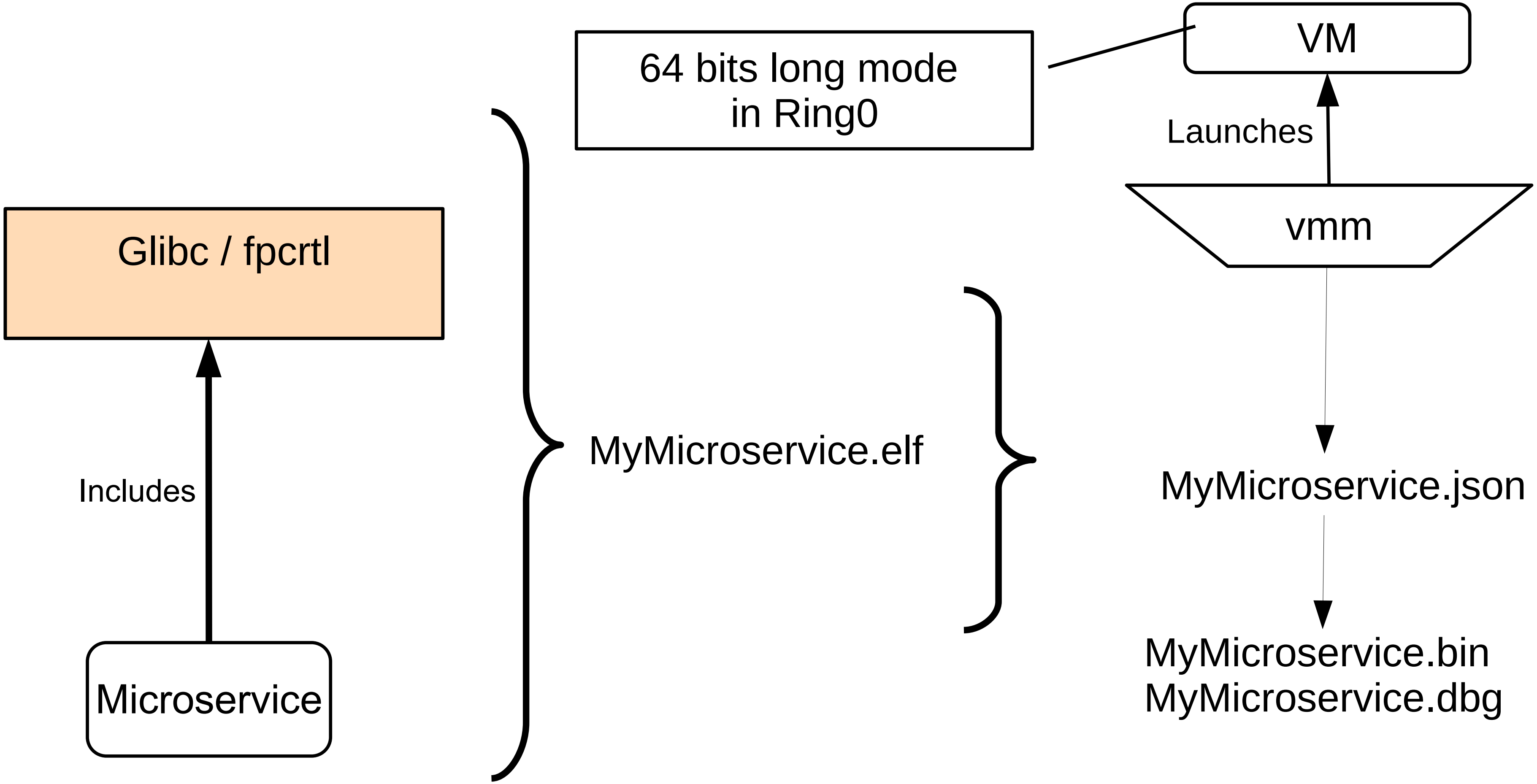


Virtual Machine #0
Microservice #0
Function call
Glibc / fpcrtl
Hypercall
(*out*)
(Linux 64 bits ABI)
Ring 0
Ring -1
ToroV
Reduced PID Namespace
Chroot
VMM
HyperCalls
Configurable syscalls
Linux Host
KVM
Kernel
Hardware
Memory
CPUs

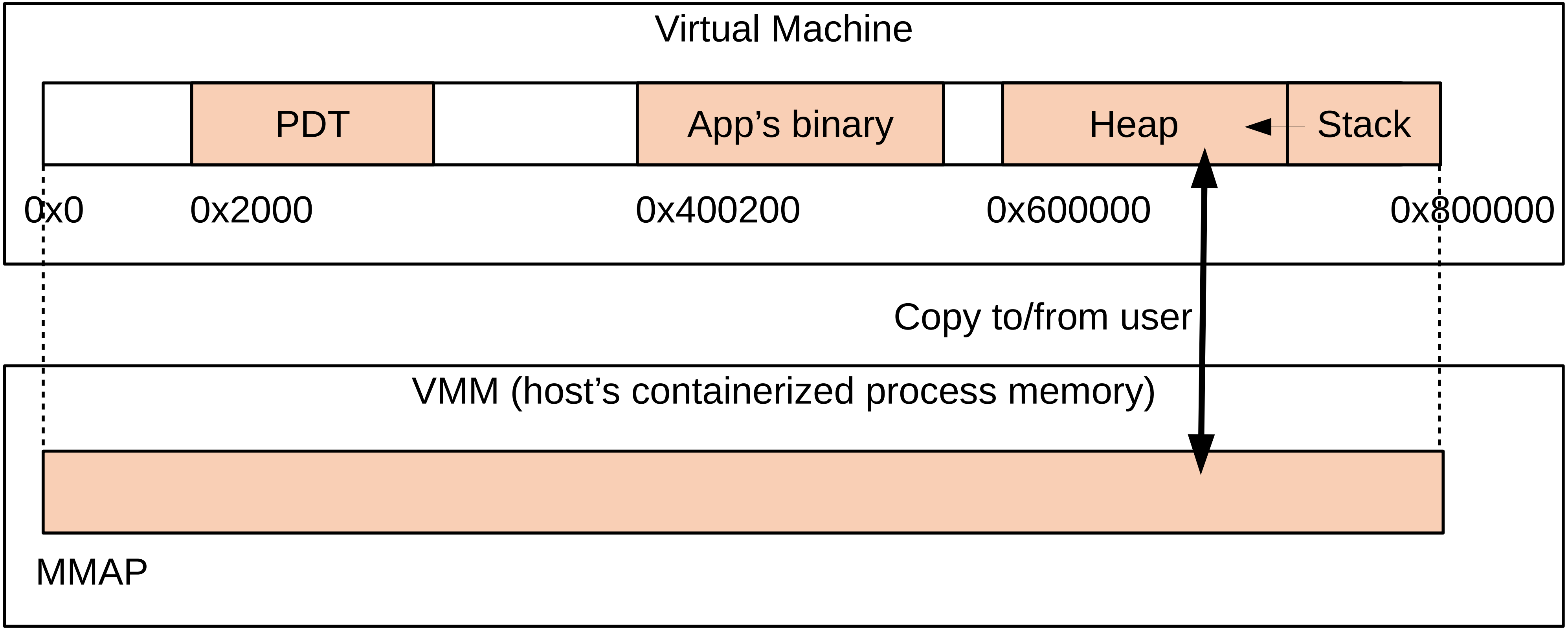

Virtual Machine #0
Microservice #0
Function call
Glibc / fp
Hypercall
(out)
(Li
Ring 0
Ring -1
KVM
{
"Binary" : "HelloWorld.bin",
"MountPoint" : "./mountpoint",
"Newpid" : "False",
"Debug" :
{
"Allowed" : "False",
"Port" : "1234"
},
"Hypercalls" : [
{
"Name" : "Ioctl",
"Allowed" : "True"
},
{
"Name" : "Write",
"Allowed" : "True"
},
ToroV
Reduced PID Namespace
Chroot
VMM
HyperCalls
Configurable syscalls
Kernel
Hardware
Memory
CPUs


Virtual Mach
Microservice #0
Function call
Glibc / fp
Hypercall
(out)
(Li
KVM
Ring 0
Ring -1
{
"Binary" : "HelloWorld.bin",
"MountPoint" : "./mountpoint",
"Newpid" : "False",
"Debug" :
{
"Allowed" : "False",
"Port" : "1234"
},
"Hypercalls" : [
{
"Name" : "Ioctl",
"Allowed" : "True"
},
{
"Name" : "Write",
"Allowed" : "True"
},
ToroV
Reduced PID Namespace
Chroot
VMM
HyperCalls
Kernel
Hardware
Memory
CPUs

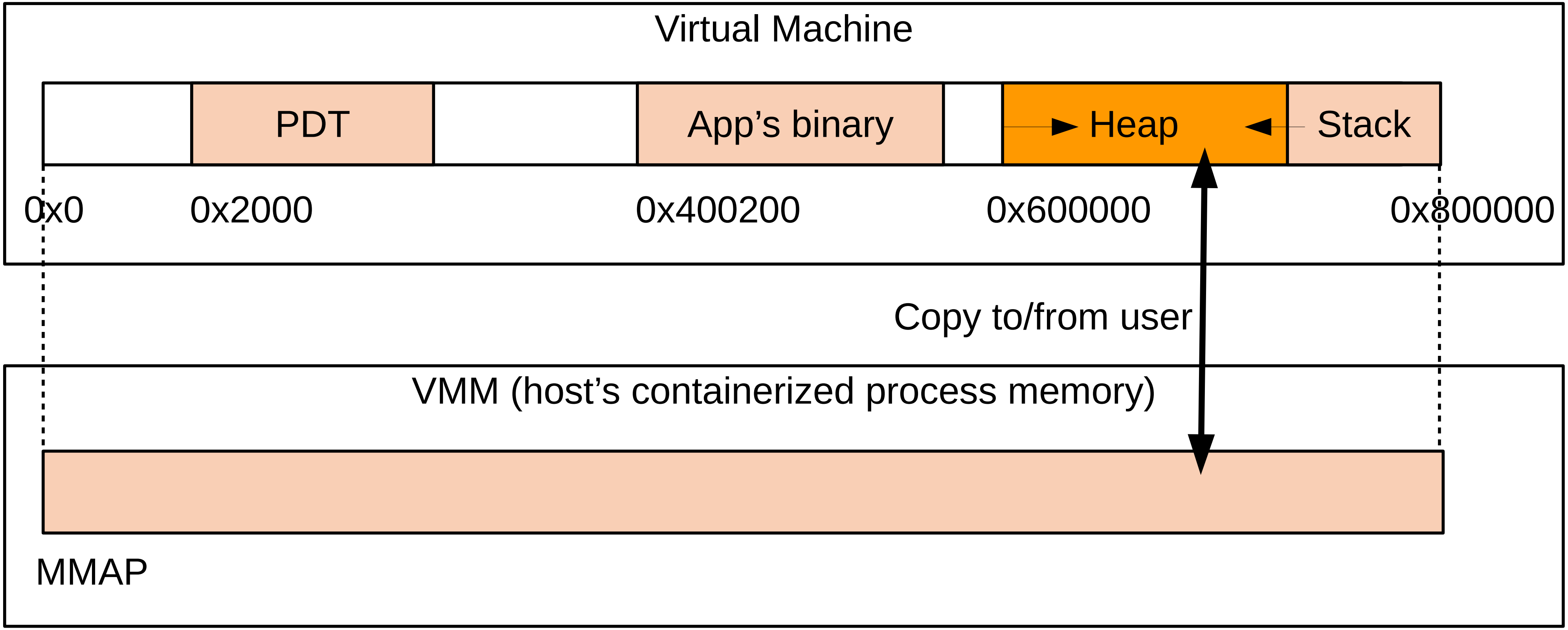
Compilation Process



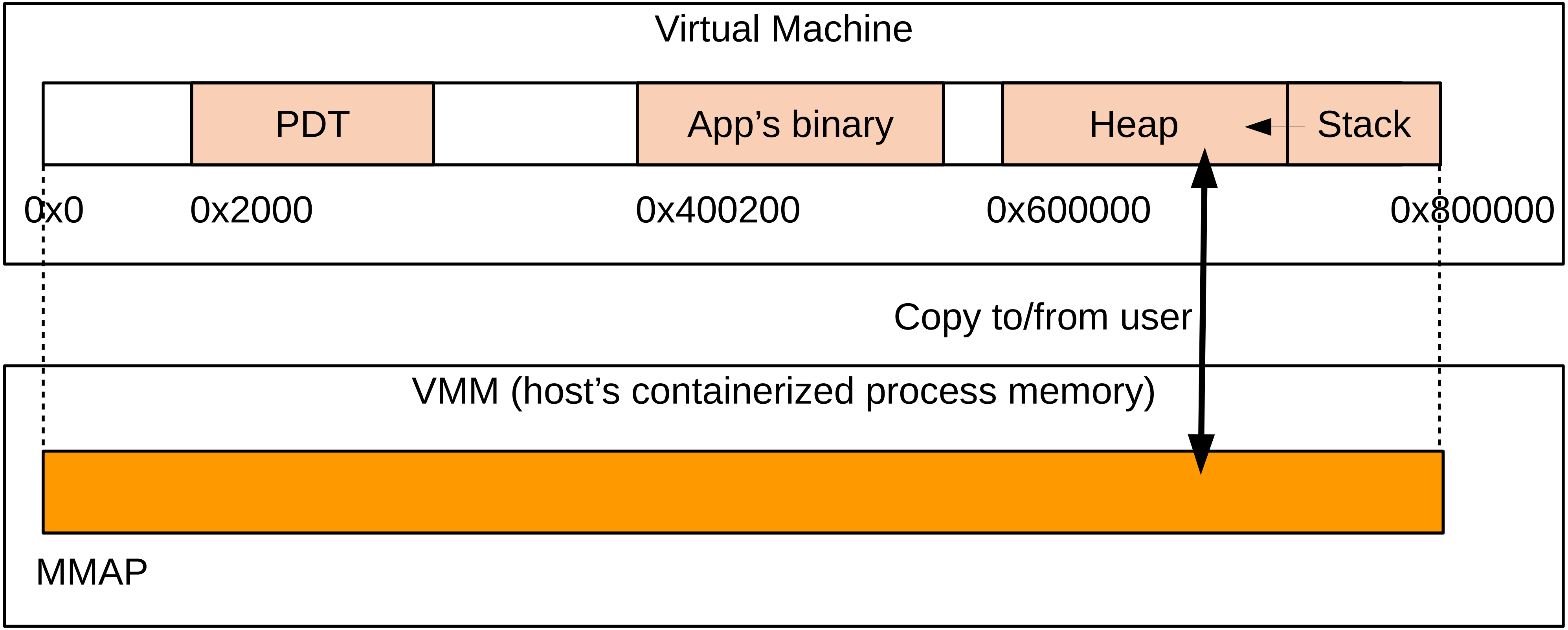
Memory Layout



Memory Layout



Memory Layout



HelloWorld example

```json
{
   "Binary" : "HelloWorld.bin",
   "MountPoint" :
"./mountpoint",
   "Newpid" : "False",
   "Debug" :
      {
         "Allowed" : "False",
         "Port" : "1234"
      },
   "Hypercalls" : [
      {
         "Name" : "Ioctl",
         "Allowed" : "True"
      },
      {
         "Name" : "Write",
         "Allowed" : "True"
      },
      {
         "Name" : "Getrlimit",
         "Allowed" : "True"
      }
   ]
}
```

strace -f ../../src/vmm/vmm helloworld.json

```
ioctl(5, KVM_CREATE_VCPU, 0)           = 6
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0) = 0x7f8908125000
ioctl(6, KVM_GET_SREGS, {cs={base=0xffff0000, limit=65535, selector=61440, type=11, present=1,
dpl=0, db=0, s=1, l=0, g=0, avl=0}, ...}) = 0
ioctl(6, KVM_SET_SREGS, {cs={base=0, limit=4294967295, selector=8, type=11, present=1, dpl=0,
db=0, s=1, l=1, g=1, avl=0}, ...}) = 0
ioctl(6, KVM_SET_REGS, {rax=0, ..., rsp=0x7fff00, rbp=0, ..., rip=0x400200, rflags=0x2}) = 0
clone(child_stack=0x513a30, flags=CLONE_VM|CLONE_FILES|CLONE_NEWNS|SIGCHLD) =
16528
wait4(16528, strace: Process 16528 attached
 <unfinished ...>
[pid 16528] chroot("./mountpoint")      = -1 ENOENT (No such file or directory)
[pid 16528] chdir("/")                  = 0
[pid 16528] ioctl(6, KVM_RUN, 0)        = 0
[pid 16528] ioctl(6, KVM_GET_REGS, {rax=0x1, ..., rsp=0x7ffe28, rbp=0x7ffe38, ..., rip=0x4005c5,
rflags=0x2}) = 0
[pid 16528] write(2, "Hello World, I am ToroV!\n", 25Hello World, I am ToroV!
) = 25
[pid 16528] ioctl(6, KVM_SET_REGS, {rax=0x19, ..., rsp=0x7ffe28, rbp=0x7ffe38, ..., rip=0x4005c5,
rflags=0x2}) = 0
[pid 16528] ioctl(6, KVM_RUN, 0)        = 0
```

HelloWorld example

- ~ 1.5 MB of memory (top)
- ~ 7 ms (median)
- Write() syscall ~ 0.10 ms ~ x10 slower (0.012 ms)

Future Work

- Work on Glibc and other languages like Go or Rust
- Research about how ToroV compares with seccomp
- Enable the use of binaries without recompilation by replacing “syscall” opcode by “out” opcode thus starting the program as-is
- Port the whole project to Rust
- Replace the current syscall mechanism for an asynchronous mechanism, .e.g, virtio device for syscalls.
- Enable that different components handles different syscalls, e.g., SOA

Resources

- GitHub repository at https://github.com/torokernel/torov
- “Debugging applications that run as VM by using GDB”, https://youtu.be/QC8pYtMOWe4
- “Using ToroV to isolate an app by using virtualization and containerization technologies”, https://youtu.be/YDpE8jlwVPA
- “Simple HelloWorld in C in ToroV”, https://youtu.be/E_bQPc64WIM
- “Simple Echo server by relying on POSIX hypercalls”, https://youtu.be/aJpcmZhDqMw

Thanks!

www.torokernel.io