6.033 Spring 2018

Lecture #6

- **Monolithic kernels vs. Microkernels**
- **Virtual Machines**

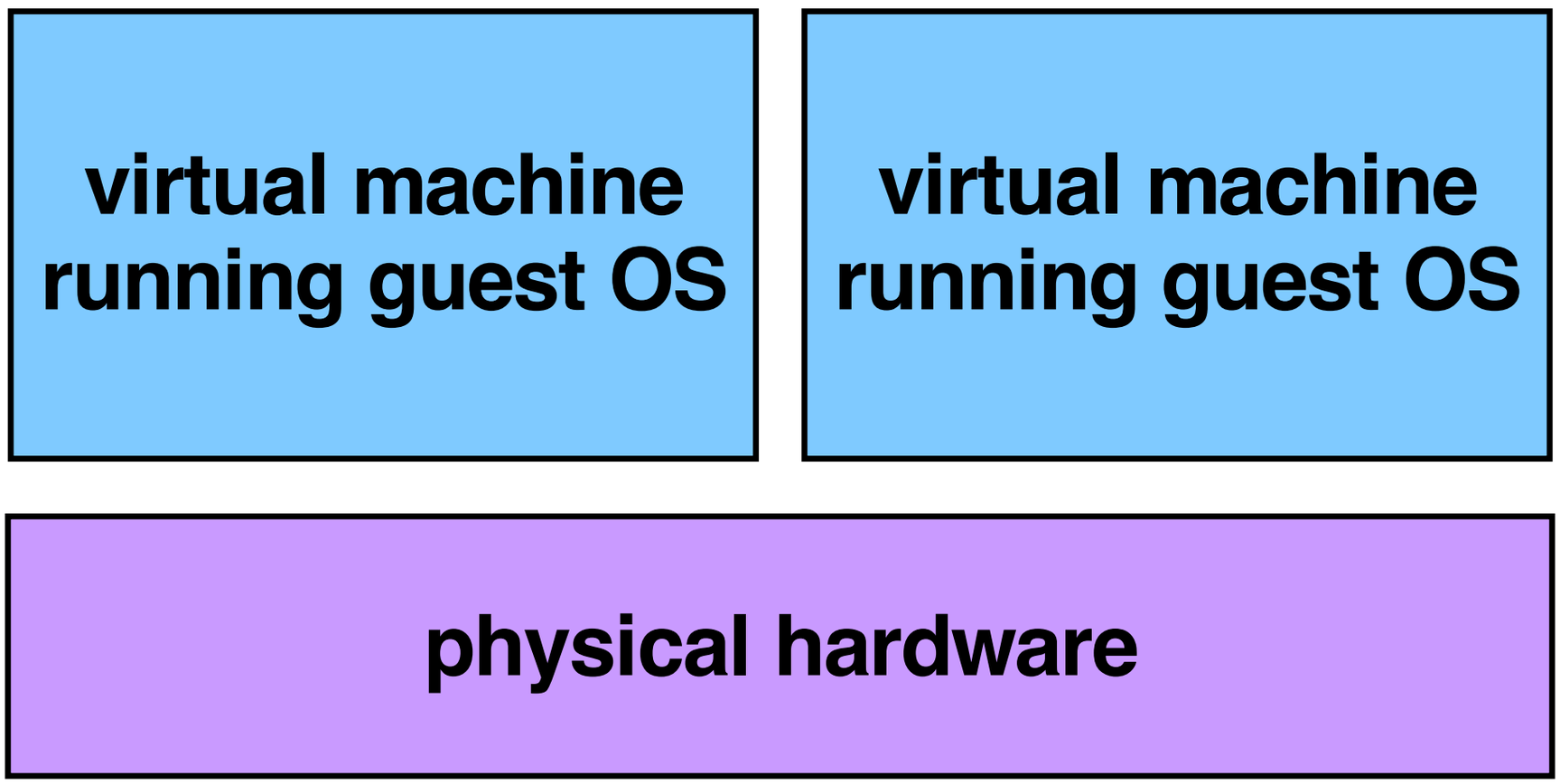
operating systems enforce modularity on a single machine using **virtualization**

in order to enforce modularity + build an effective operating system

1. programs shouldn't be able to refer to (and corrupt) each others' **memory** → **virtual memory**
2. programs should be able to **communicate** → **bounded buffers** (virtualize communication links)
3. programs should be able to **share a CPU** without one program halting the progress of the others → **threads** (virtualize processors)

today: running multiple OSes at once
(and dealing with kernel bugs)

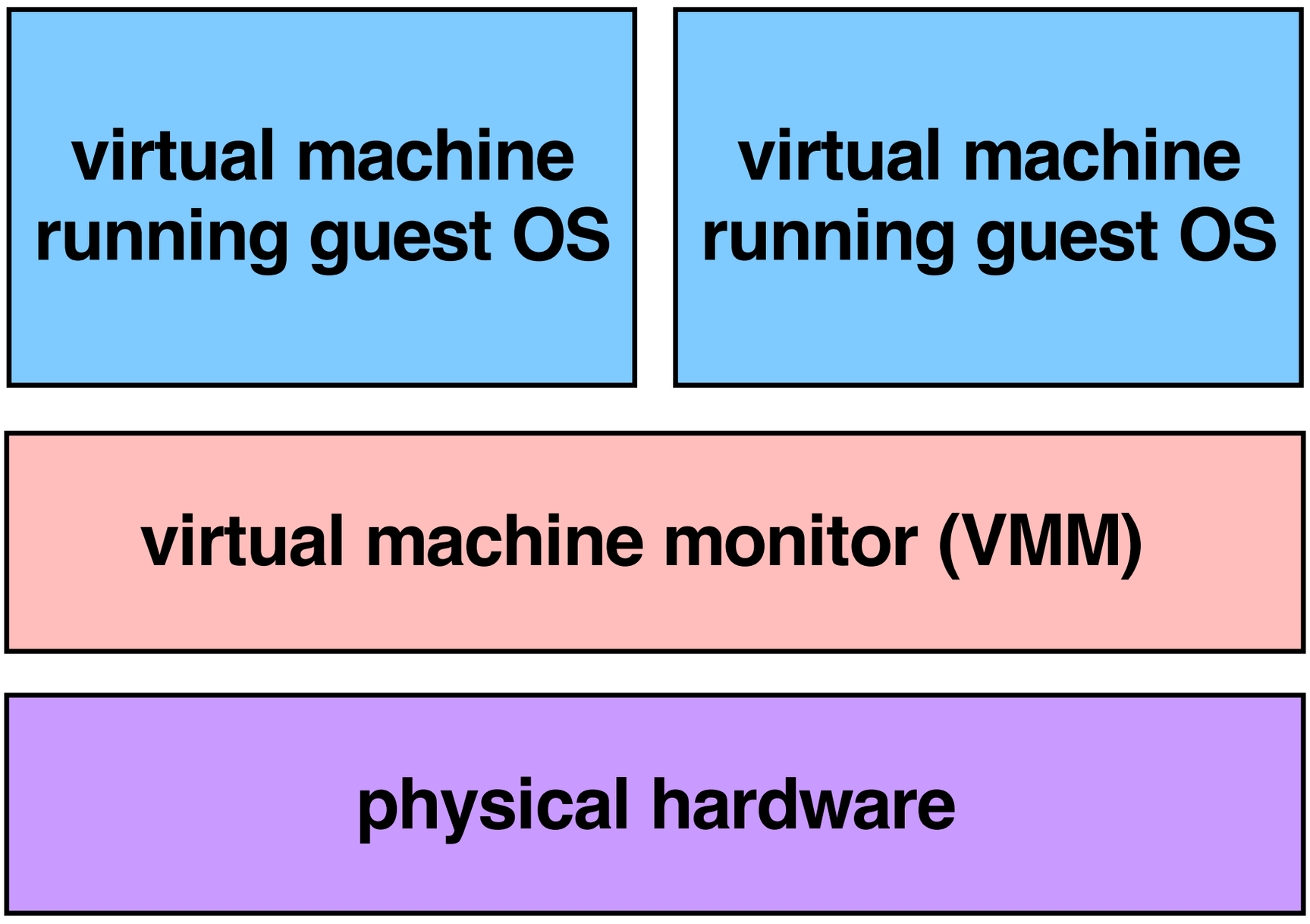
Virtual Machines

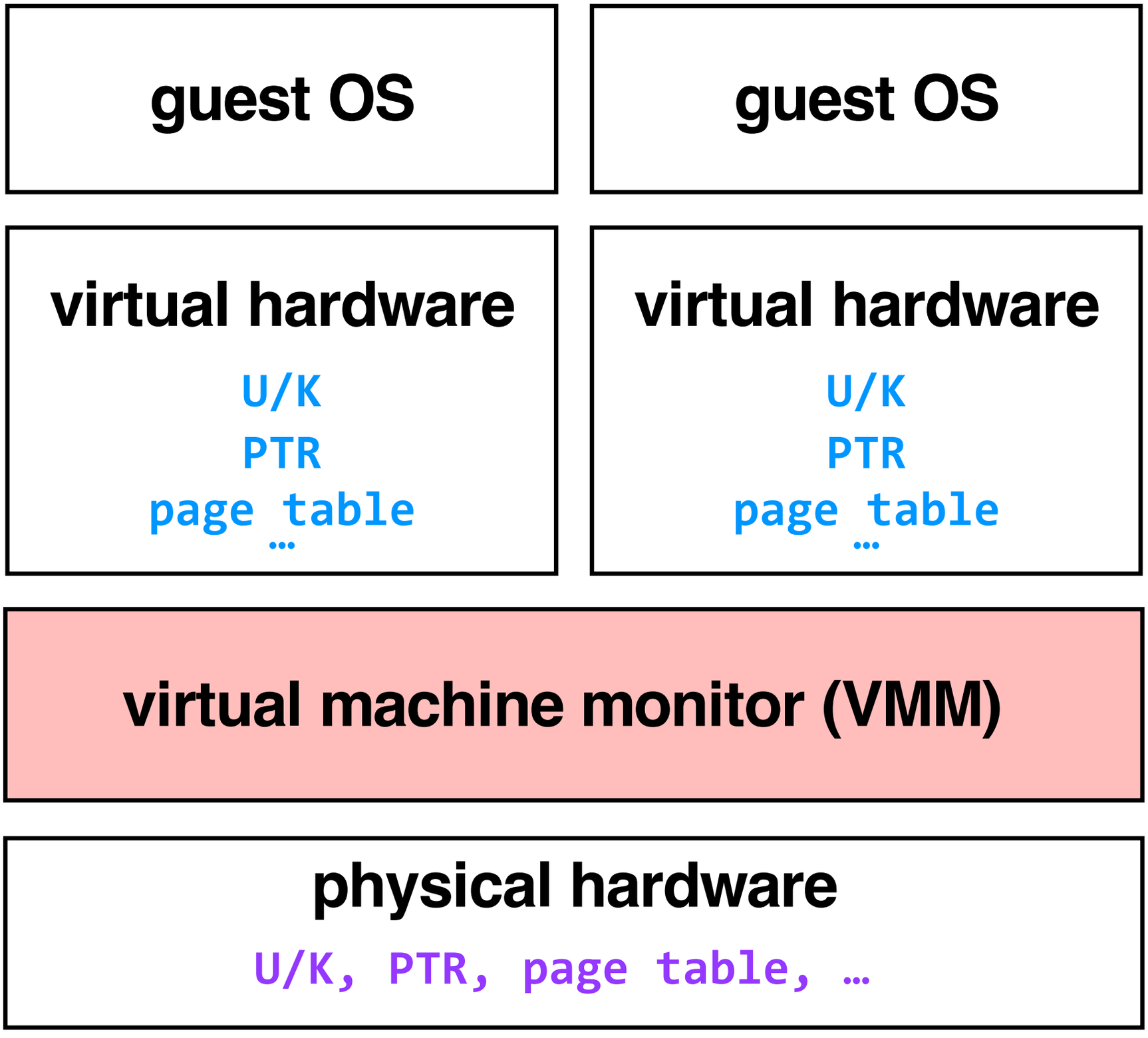


problem: how to (safely) share access to physical hardware?

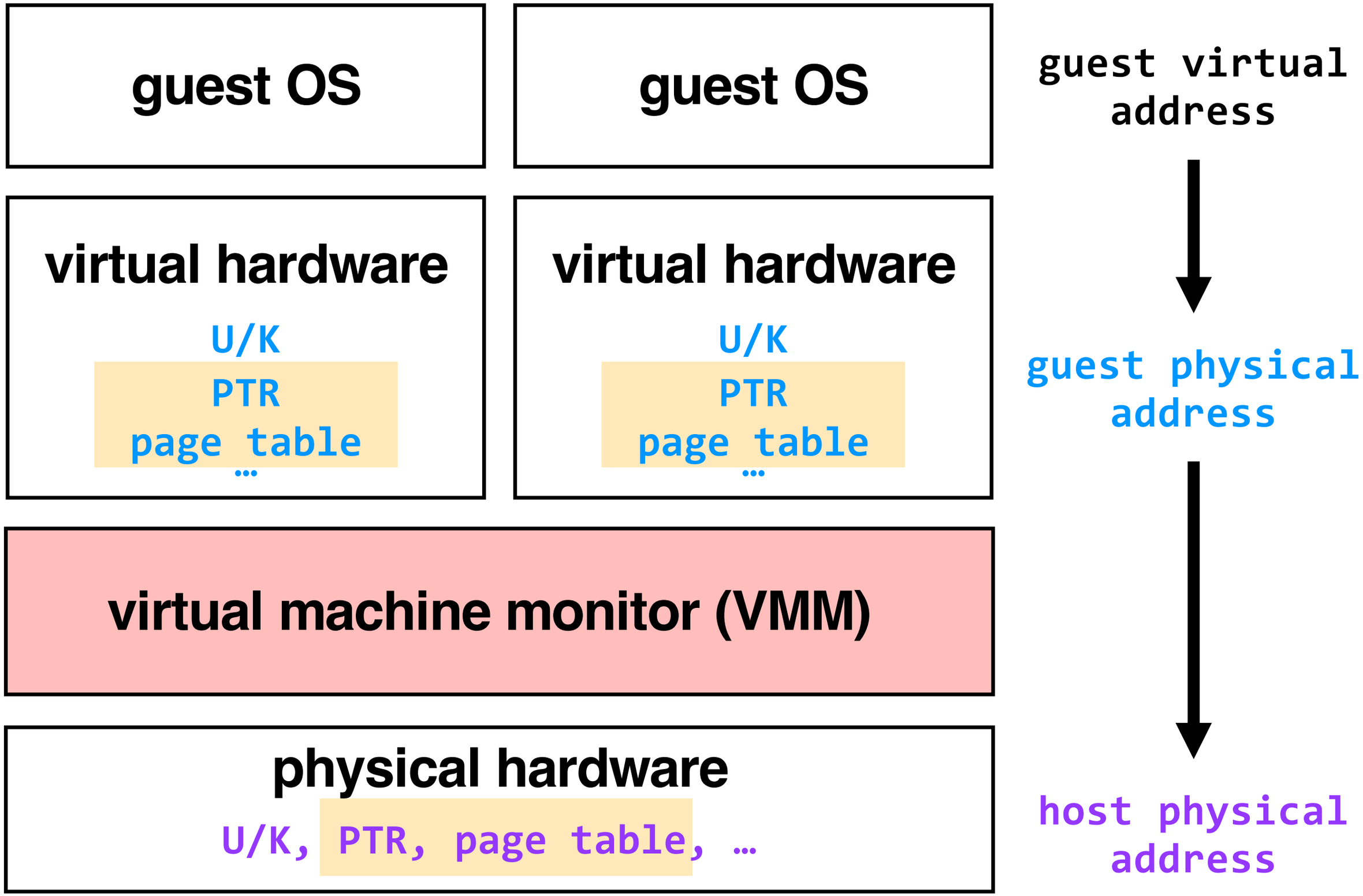
Virtual Machines

VMM runs in kernel-mode on hardware





VMM's goal: virtualize hardware


guest OS
guest OS
virtual hardware
U/K
PTR
page table
…
virtual hardware
U/K
PTR
page table
…
virtual machine monitor (VMM)
physical hardware
U/K, PTR, page table, …
guest virtual address
guest physical address
host physical address

guest OS

virtual hardware

guest OS page table

PTR

load PTR

intercept!

VMM

guest OS page table + VMM page table → host page table

physical hardware

PTR → host page table

guest OS page table

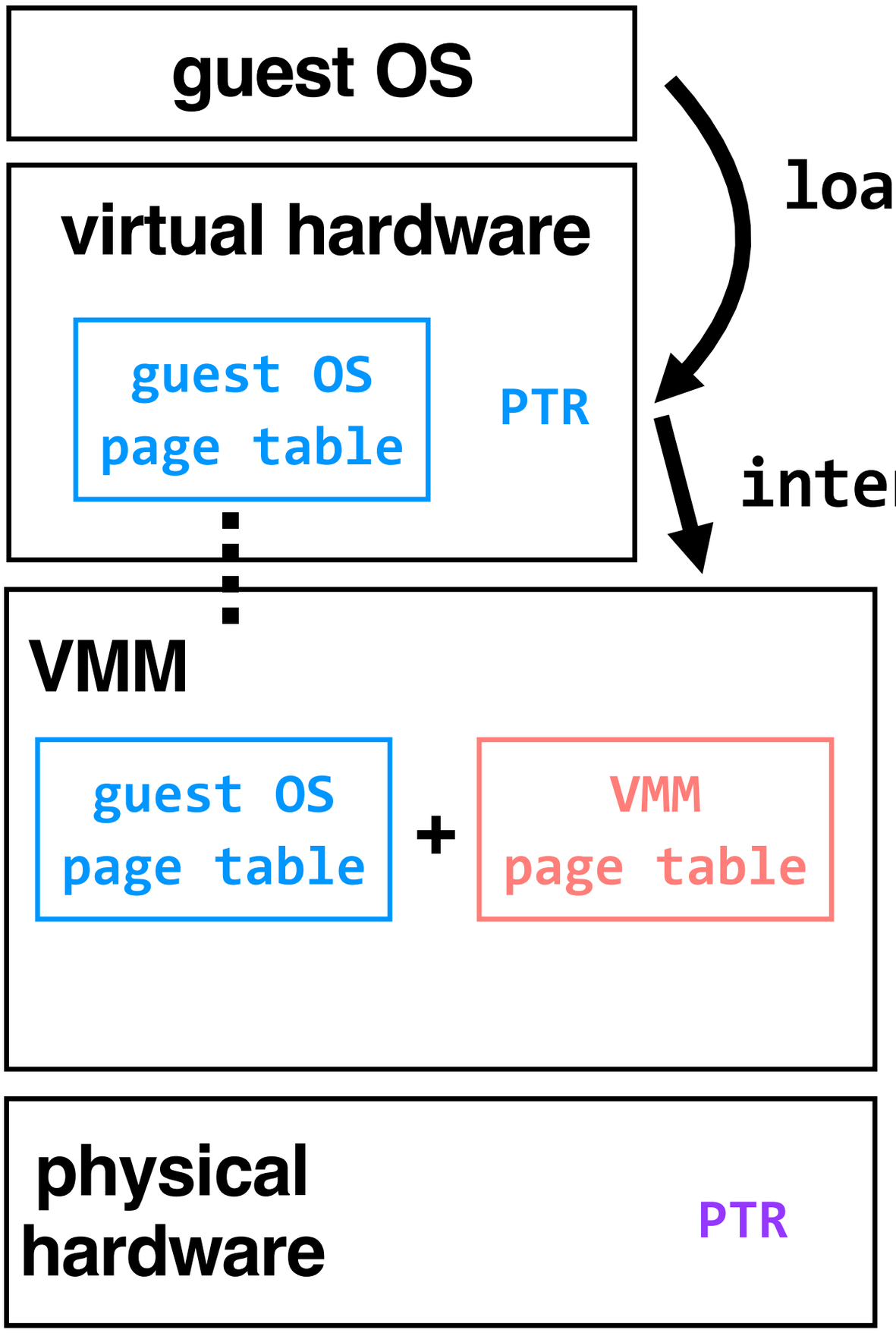
guest virtual -> guest physical

VMM page table

guest physical -> host physical

host page table

guest virtual -> host physical



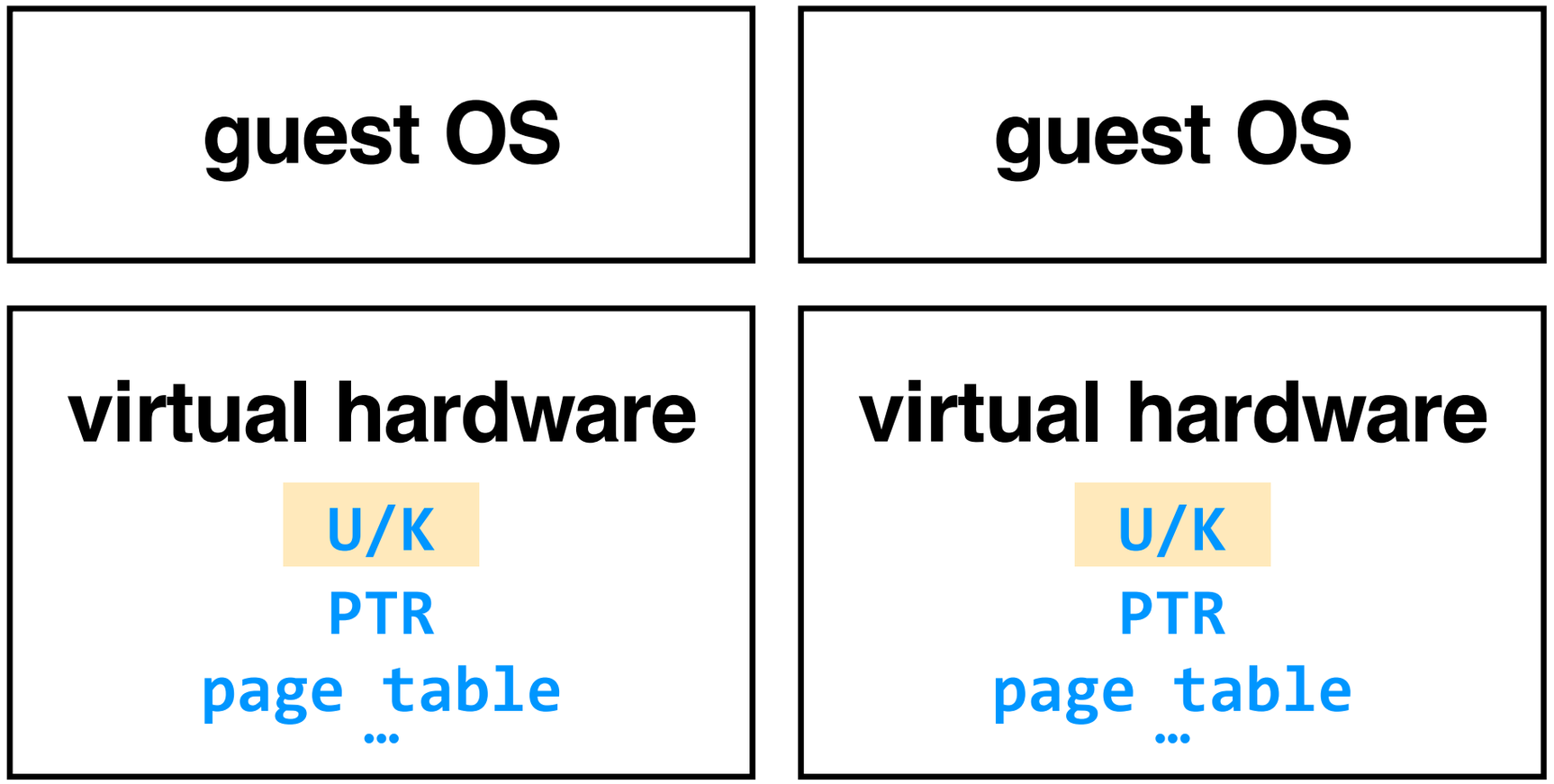
guest OS
page table

guest virtual ->
guest physical

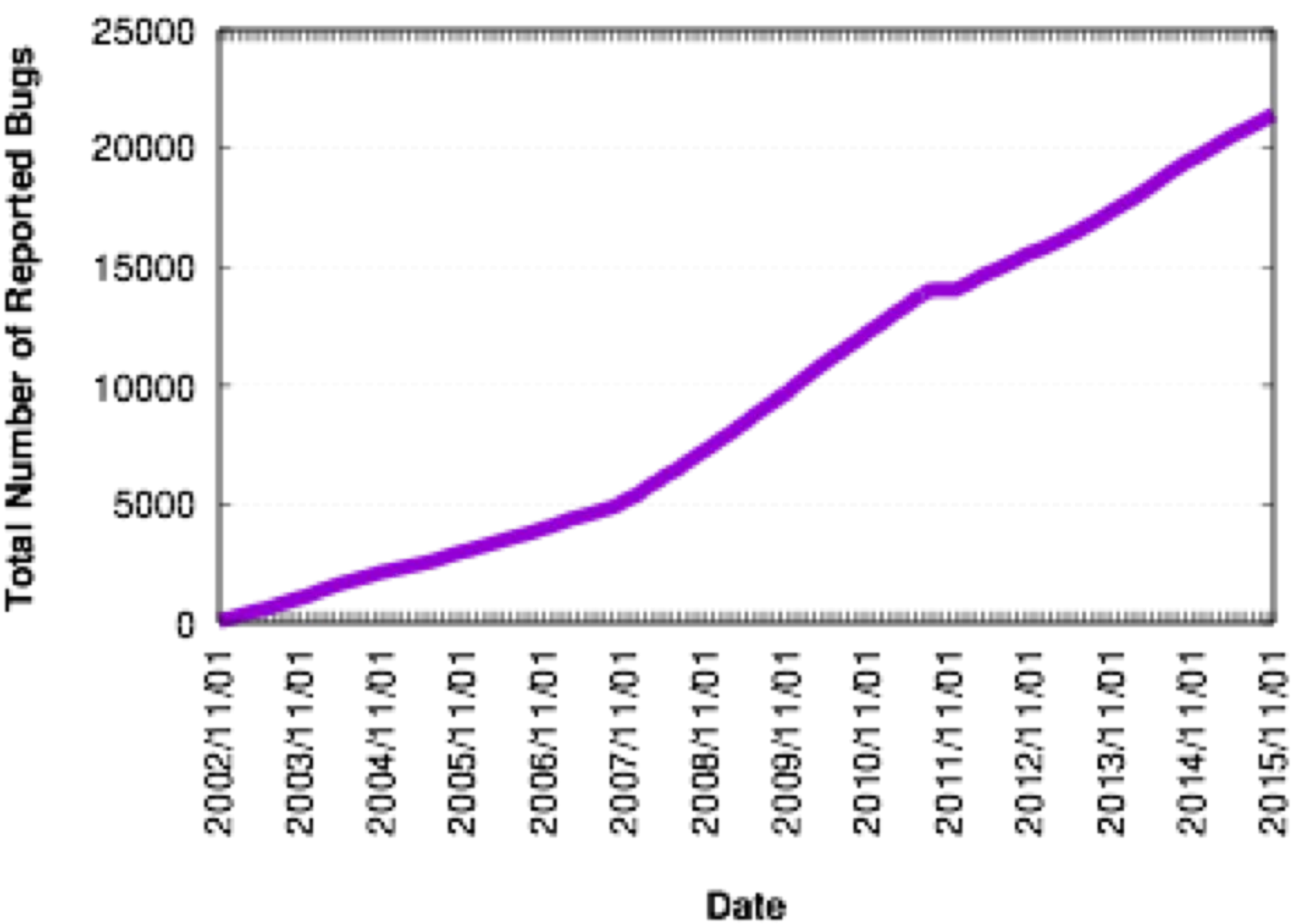
VMM
page table

guest physical ->
host physical

in modern hardware, the physical hardware is aware of both page tables, and performs the translation from guest virtual to host physical itself

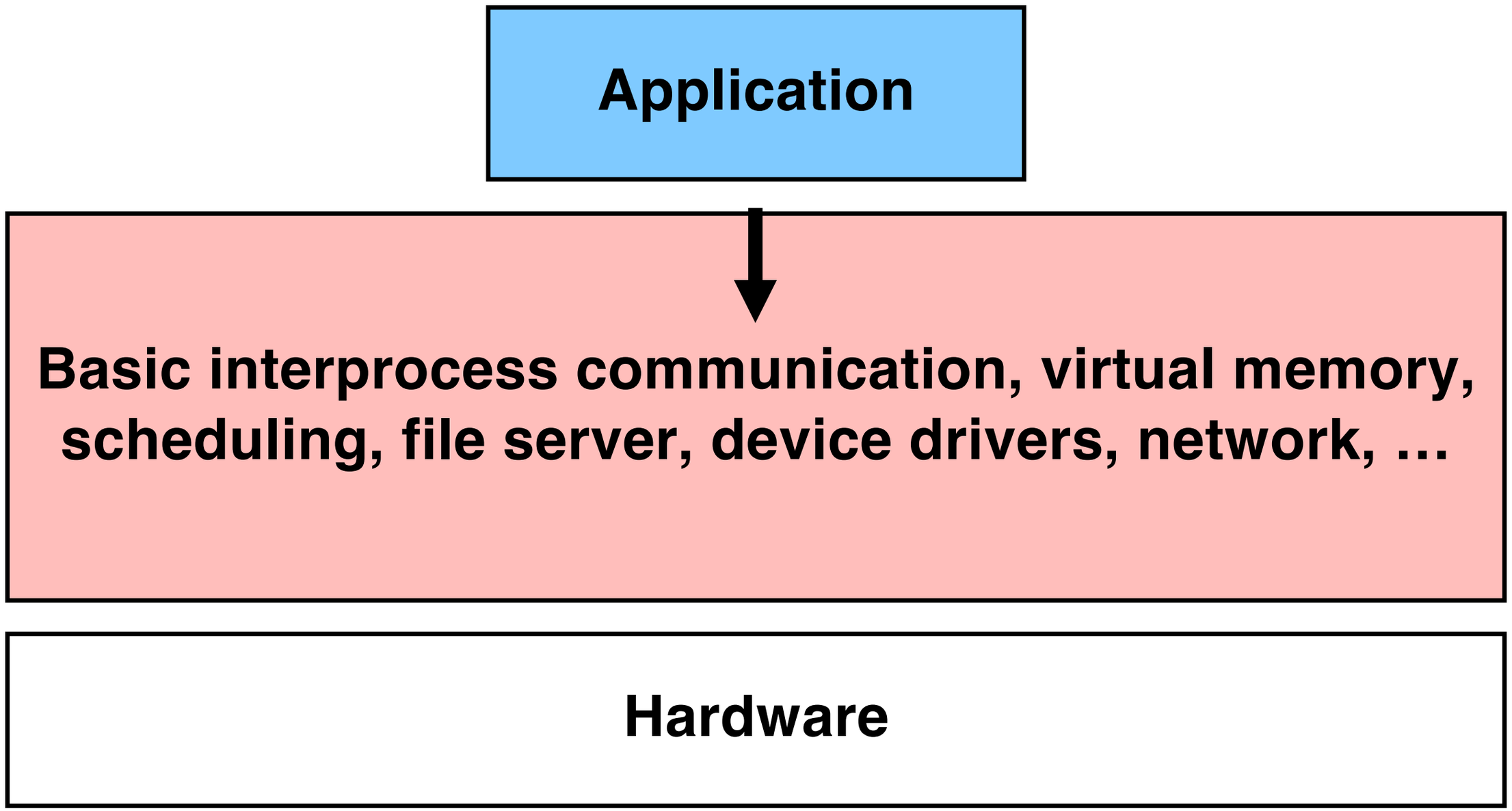


VMM's goal: virtualize hardware

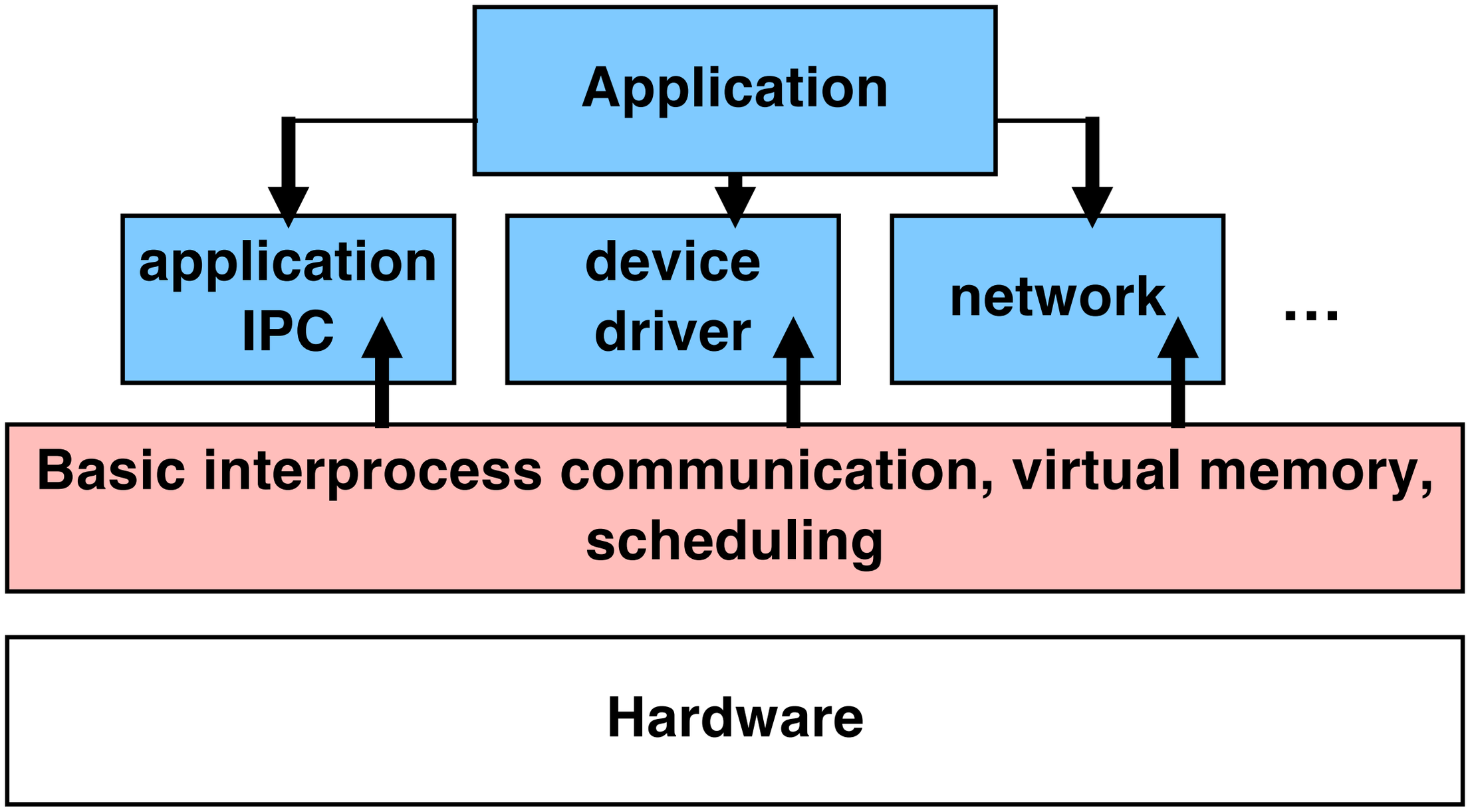


source: bugzilla.kernel.org, count of all bugs currently marked NEW, ASSIGNED, REOPENED, RESOLVED, VERIFIED, or CLOSED, by creation date

monolithic kernels: no enforced modularity within the kernel itself



microkernels: enforce modularity by putting subsystems in user programs

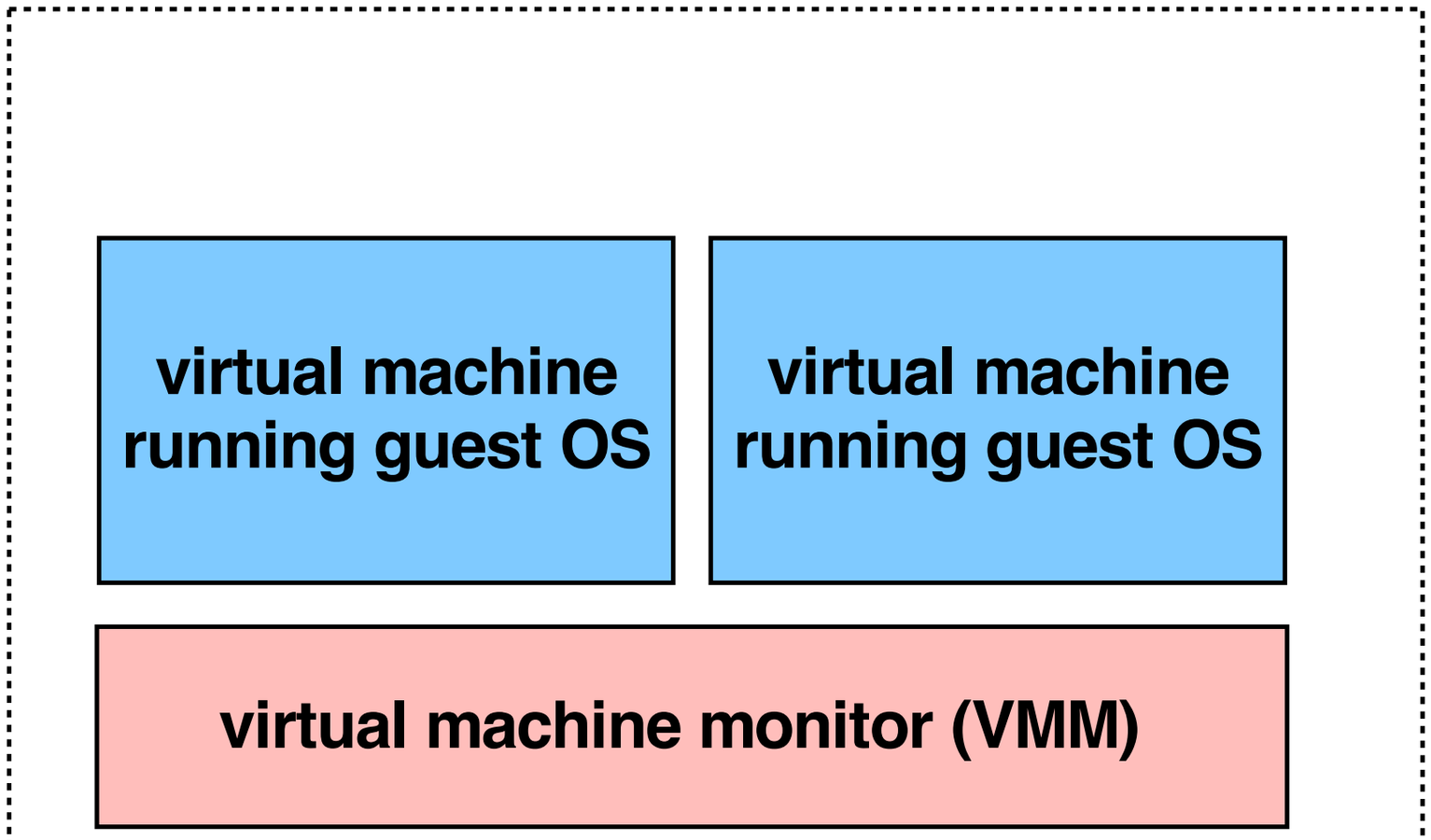


- **Virtual Machines** allow us to run multiple **isolated** OSes on a single physical machine, similar to how we used an OS to run multiple programs on a single CPU. VMs must handle the challenges of virtualizing the hardware (examples: virtualizing memory, the U/K bit).

- **Monolithic kernels** provide no enforced modularity within the kernel. **Microkernels** do, but redesigning monolithic kernels as microkernels is challenging.

Virtual Machines

(in the host OS model, there was actually a host OS)



6.033 Computer System Engineering
Spring 2018