# 6.033 Spring 2017

## Lecture #6

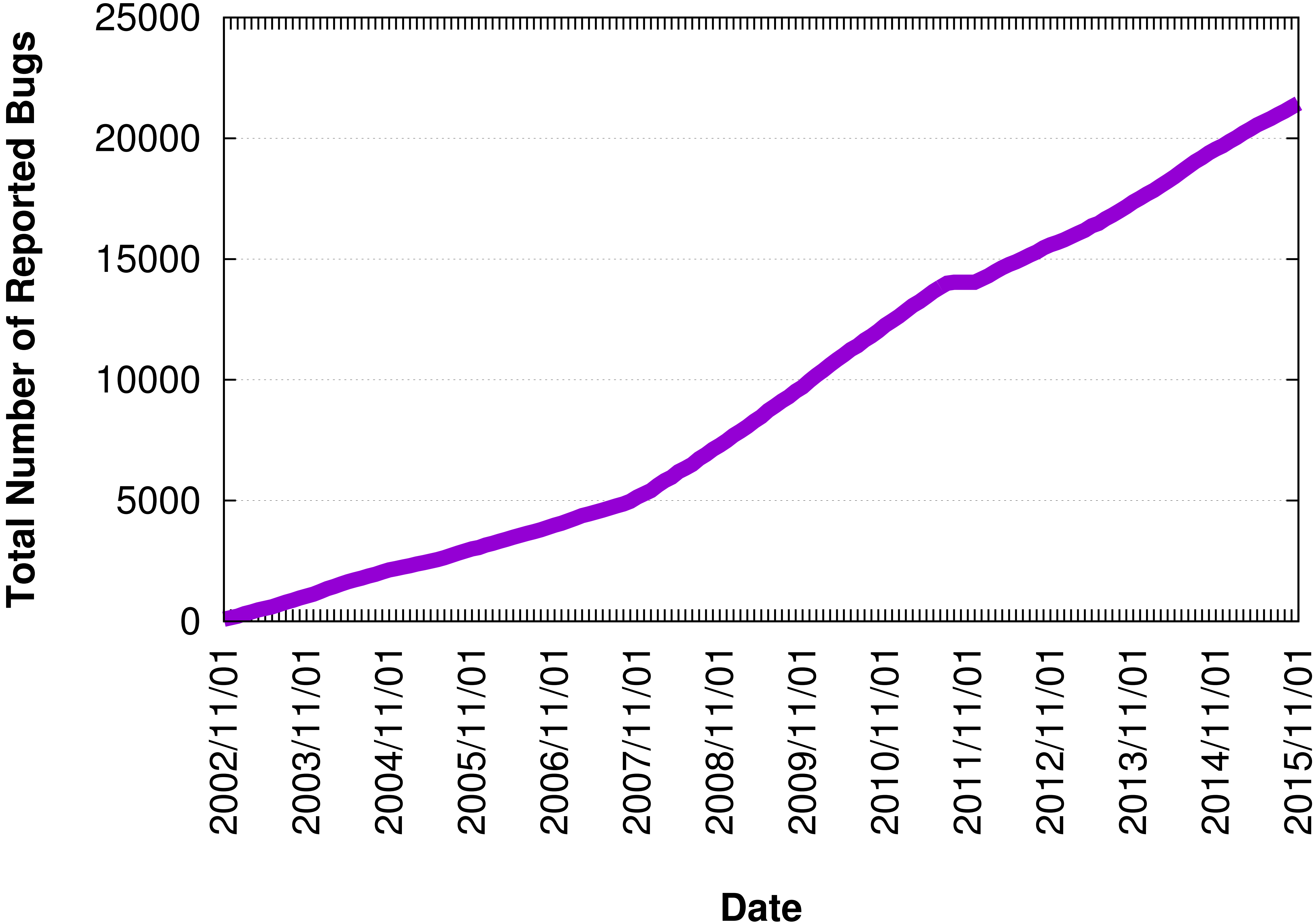
- **Monolithic kernels vs. Microkernels**
- **Virtual Machines**

# Enforcing Modularity via Virtualization

in order to enforce modularity + build an effective operating system

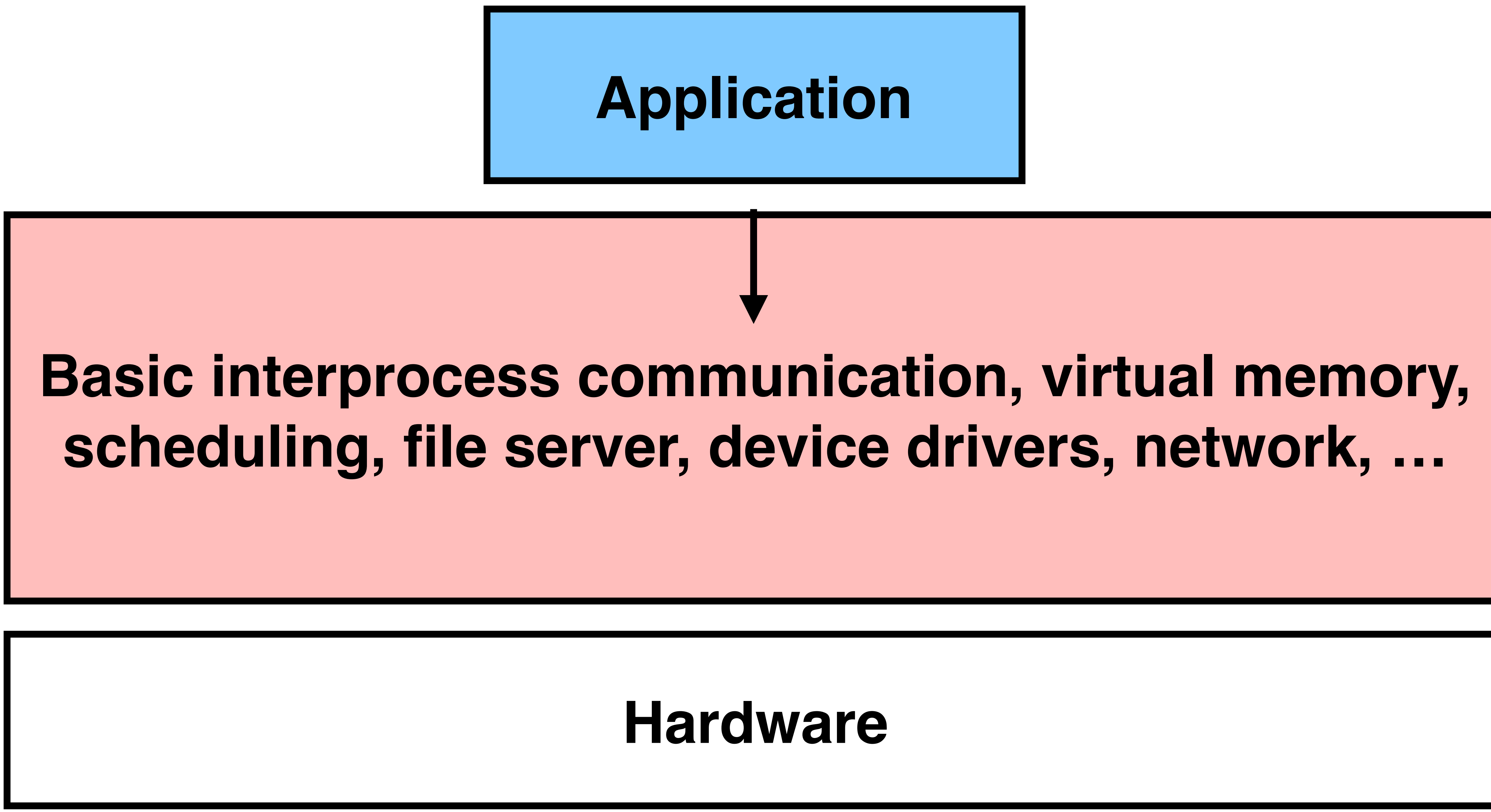
1. programs shouldn't be able to refer to (and corrupt) each others' **memory** → **virtual memory**
2. programs should be able to **communicate** → **bounded buffers** (virtualize communication links)
3. programs should be able to **share a CPU** without one program halting the progress of the others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

**today:** can we rely on the kernel to work properly?

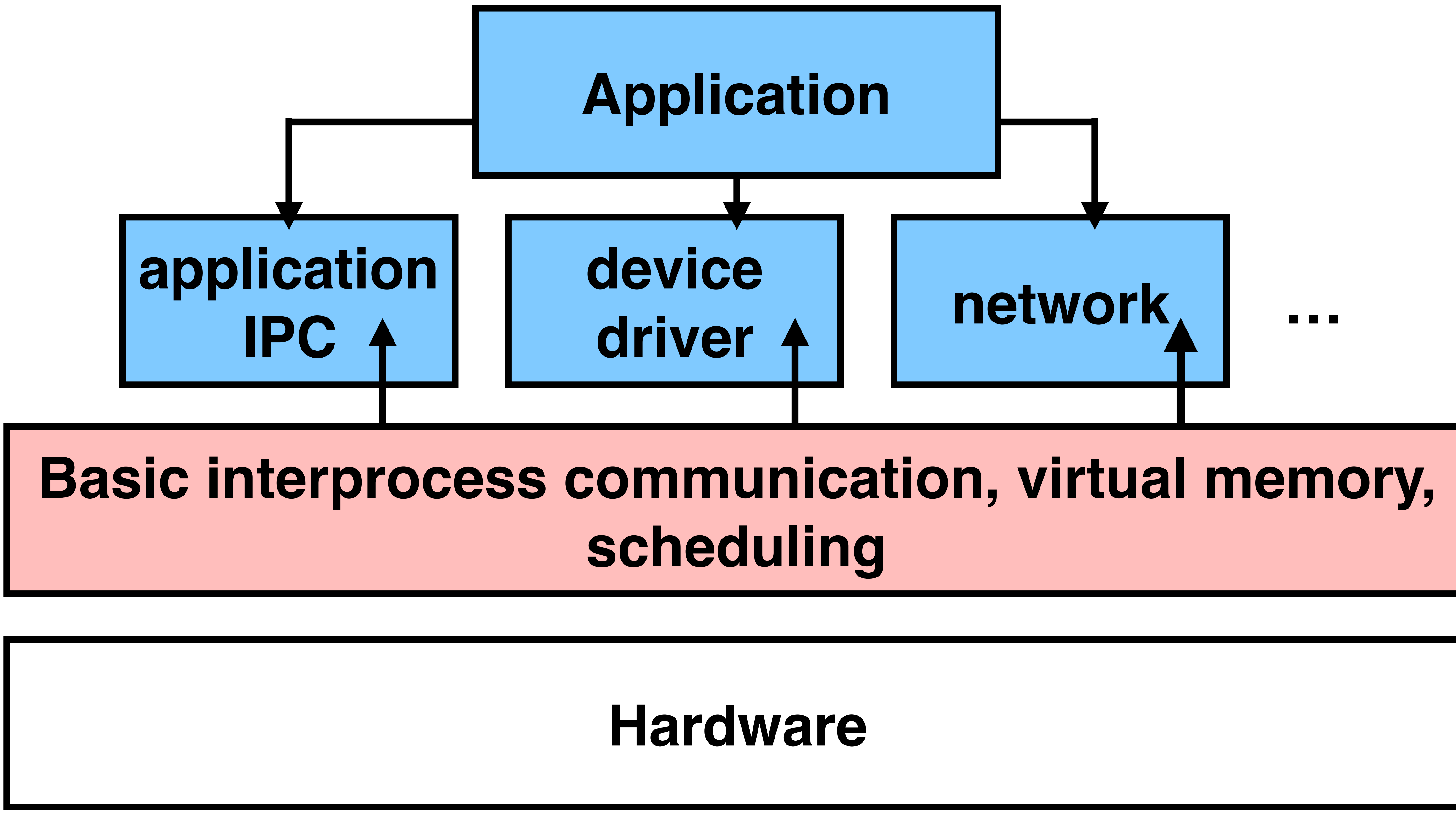



source: bugzilla.kernel.org, count of all bugs currently marked NEW, ASSIGNED, REOPENED, RESOLVED, VERIFIED, or CLOSED, by creation date

**monolithic kernels:** no enforced modularity within the kernel itself

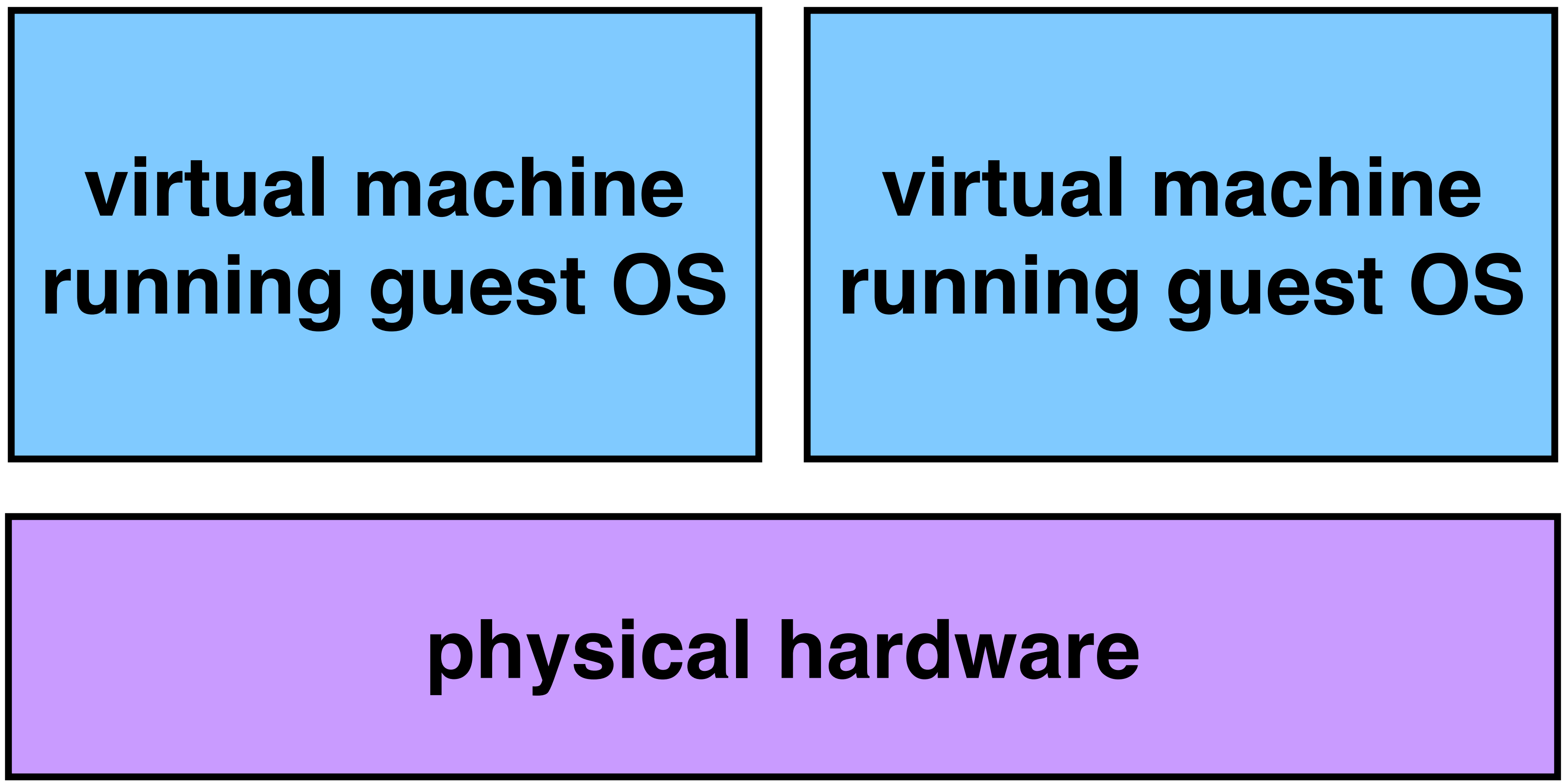


**microkernels:** enforce modularity by putting subsystems in user programs



**problem:** how do we deal with bugs in the Linux kernel without redesigning Linux from scratch?

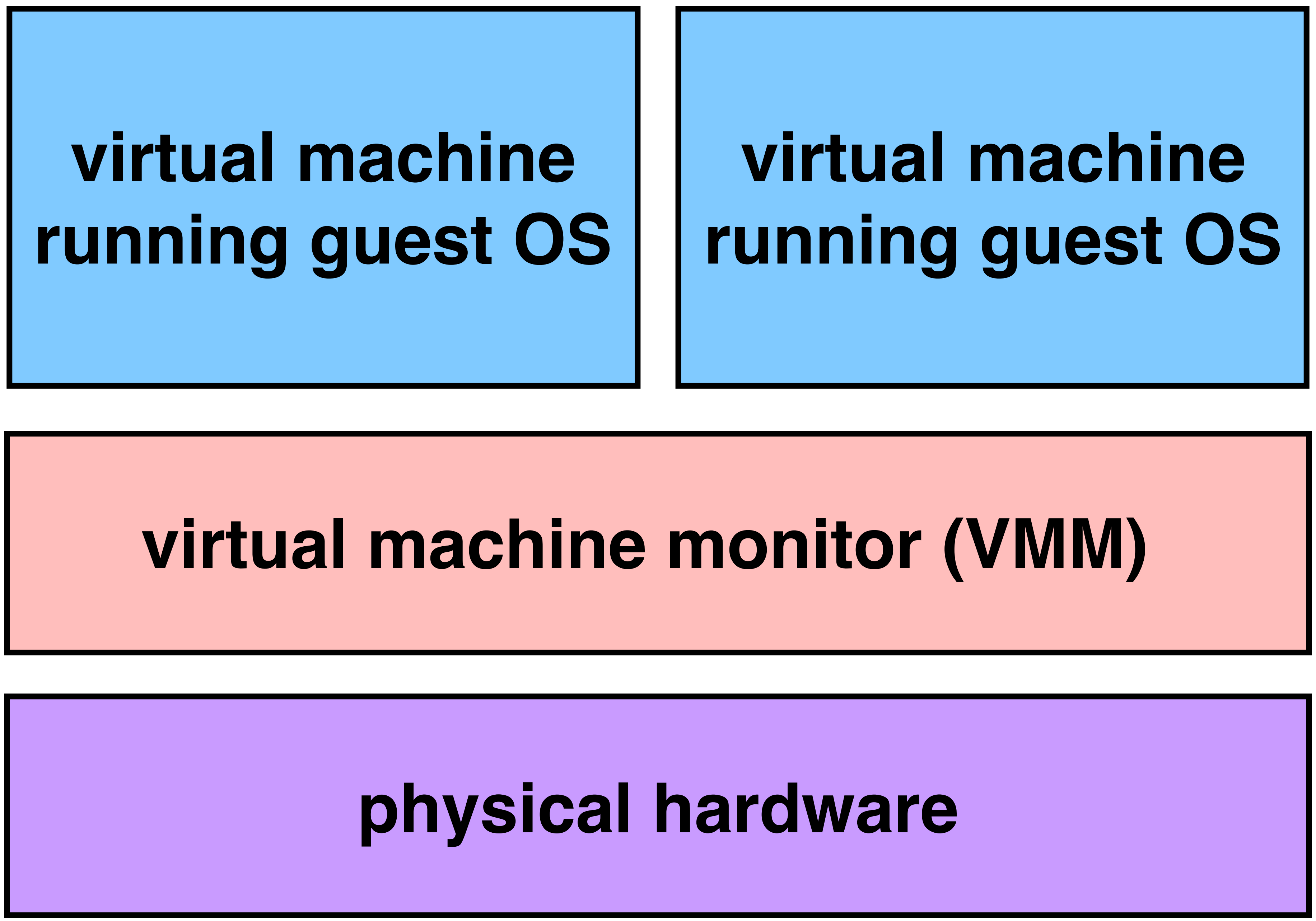
# Virtual Machines

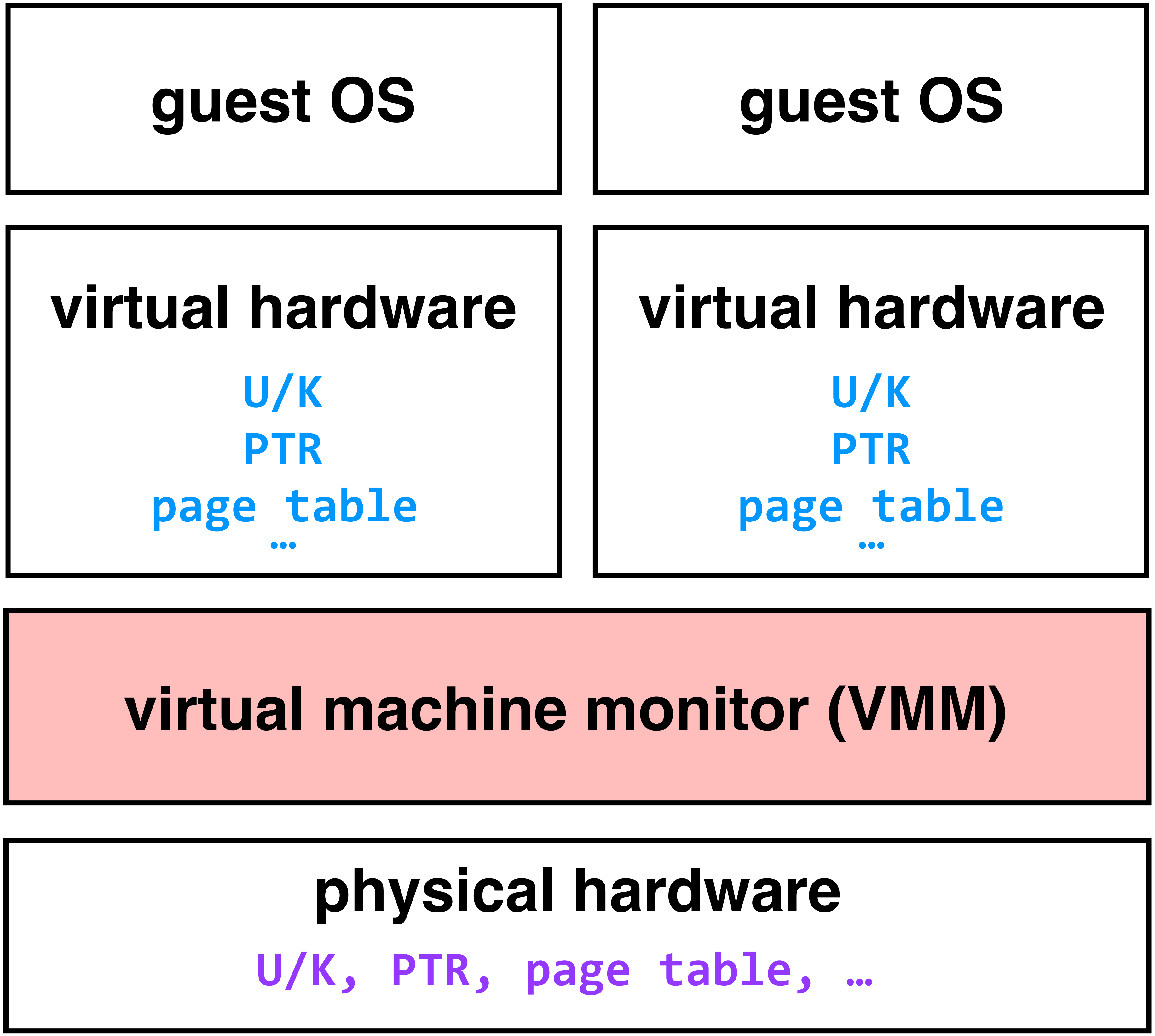



**problem:** if guest OSes run in kernel mode, we haven't solved our original program; if they run in user mode, they can't execute privileged instructions

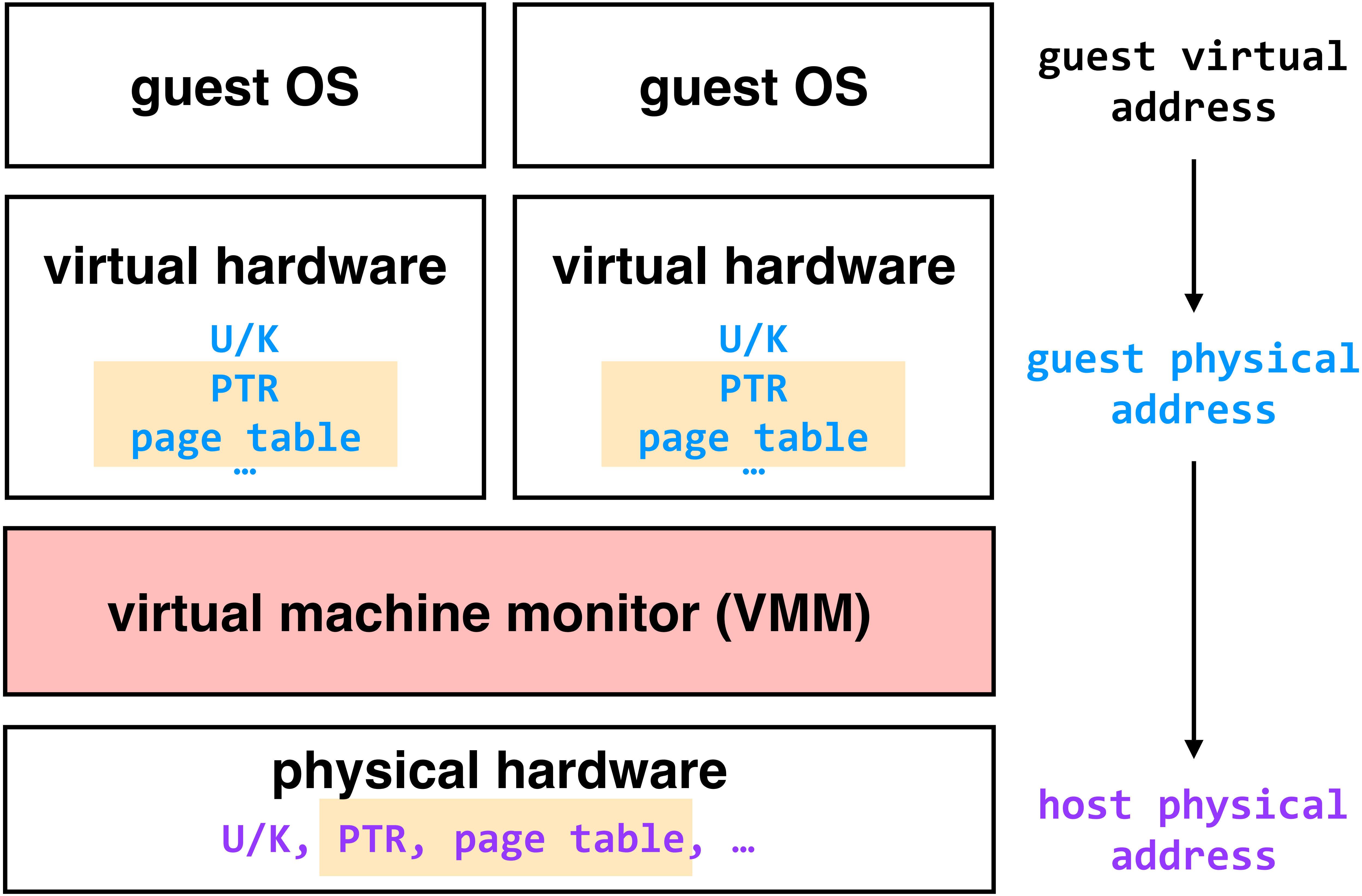
# Virtual Machines

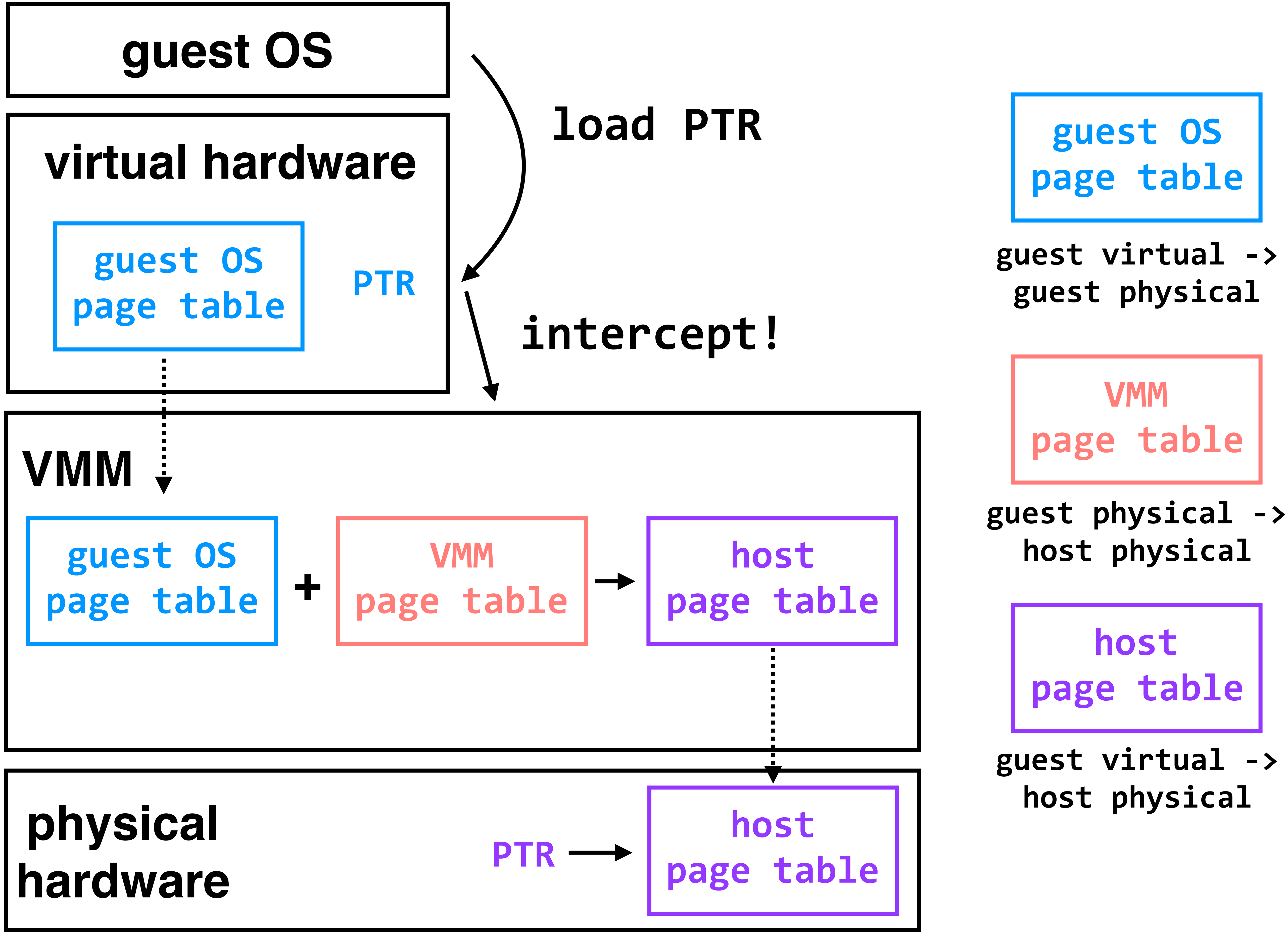
**VMM runs in kernel-mode on hardware**

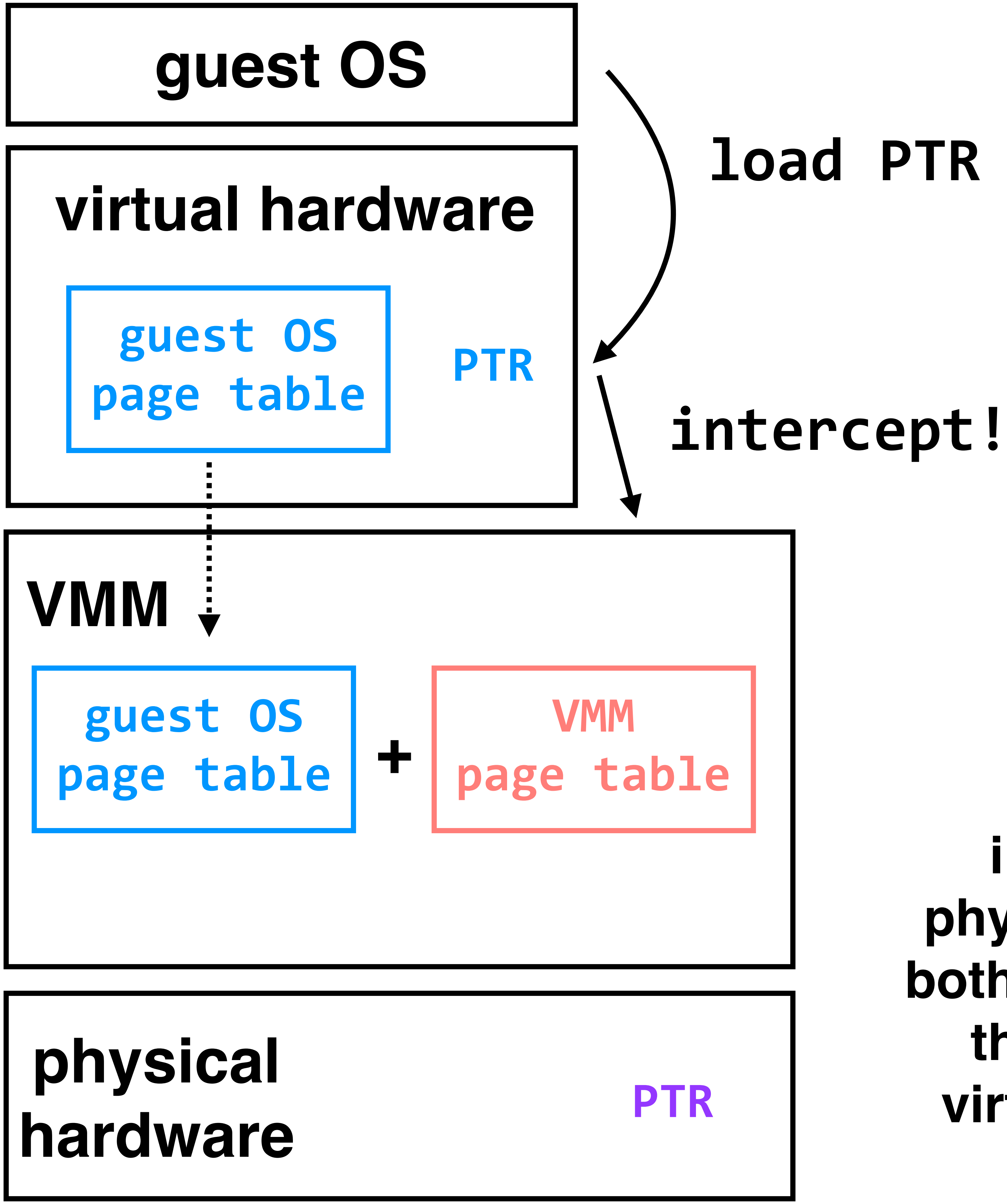





**VMM's goal:** virtualize hardware

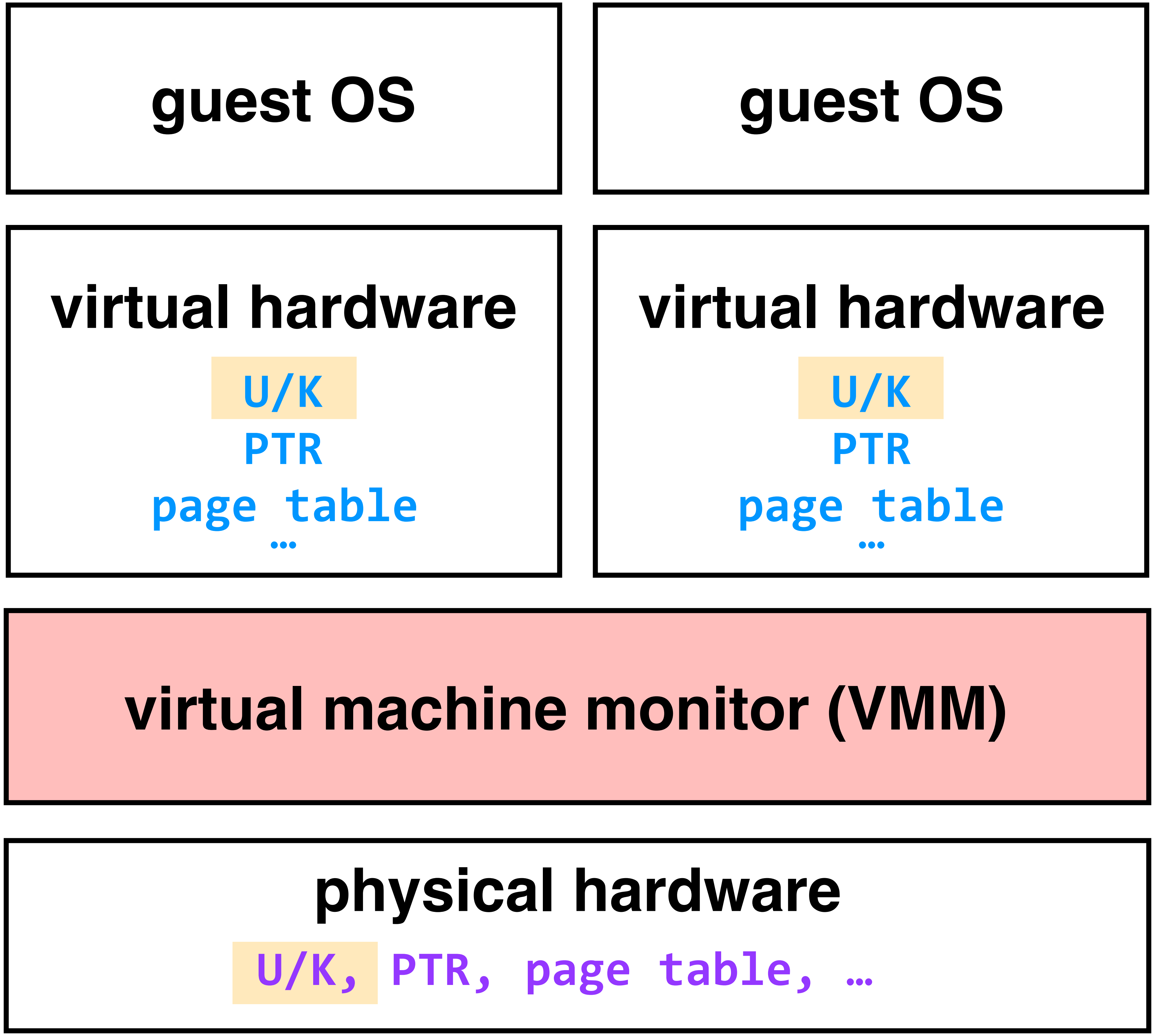

guest OS
guest OS
guest virtual address
virtual hardware
U/K
PTR
page table
…
virtual hardware
U/K
PTR
page table
…
guest physical address
virtual machine monitor (VMM)
physical hardware
U/K, PTR, page table, …
host physical address


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





**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

- **Kernel Structure**

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

- **Virtual Machines**

  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, and disk).