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)
problem: how to (safely) share physical hardware?
Virtual Machines

VMM runs in kernel-mode on hardware

- virtual machine running guest OS
- virtual machine running guest OS
- virtual machine monitor (VMM)
- physical hardware
VMM’s goal: virtualize hardware
virtual machine monitor (VMM)

guest OS

virtual hardware

virtual hardware

guest virtual address

guest physical address

host physical address

physical hardware

U/K, PTR, page table, ...

U/K, PTR, page table, ...

U/K, PTR, page table, ...

U/K, PTR, page table, ...

6.033 | spring 2019 | lacurts@mit.edu
In modern hardware, the physical hardware is aware of both page tables, and performs the translation from guest virtual to host physical itself.
virtual machine monitor (VMM)

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 (rough estimate!)
monolithic kernels: no enforced modularity within the kernel itself

Application

Basic interprocess communication, virtual memory, scheduling, file server, device drivers, network, …

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

Basic interprocess communication, virtual memory, scheduling
• **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.