Virtualization Overview

NSRC

Terminology

- Virtualization: dividing available resources into smaller independent units
- Emulation: using software to simulate hardware which you do not have
- The two often come hand-in-hand
  - e.g. we can *virtualize* a PC by using it to *emulate* a collection of less-powerful PCs

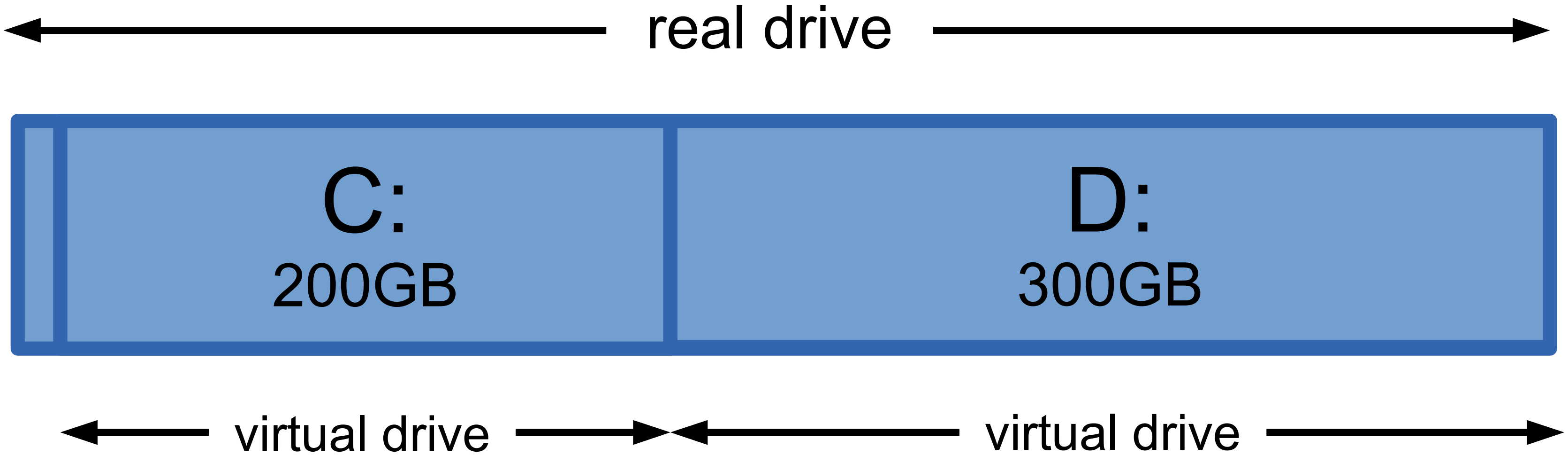
Benefits

- Consolidation
  - Most systems are under-utilized, especially the CPU is idle for much of the time
  - Do more work with less hardware
  - Reduced space and power requirements
- Management
  - Less hardware inventory to manage
  - Concentrate your resilience efforts
  - Increased isolation between services
  - Abstract away (hide) differences in hardware

Benefits

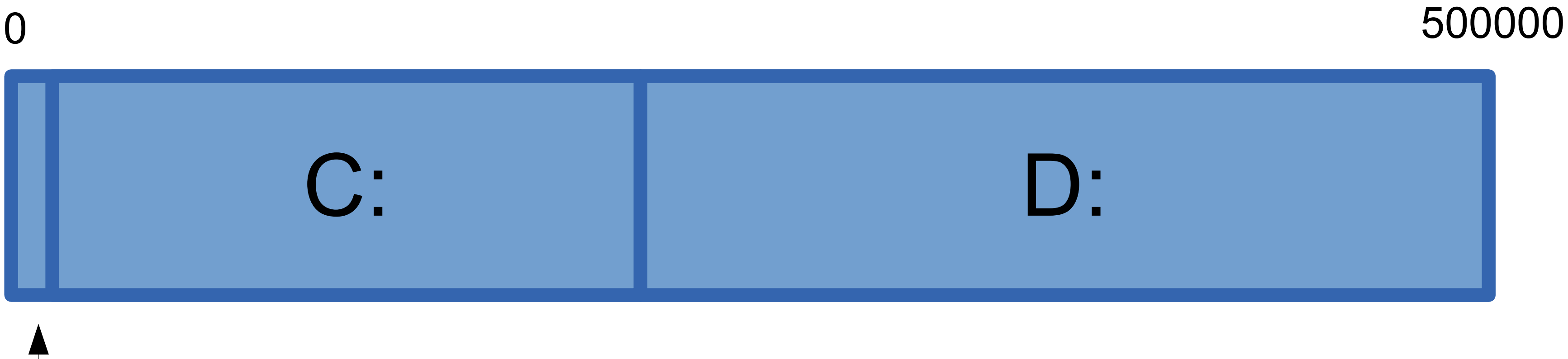
- Flexibility
  - Grow systems on demand (e.g. allocate more CPU or RAM where it is needed)
  - Create new services quickly without having to install new hardware every time
  - Dynamically create and destroy instances for testing and development
- New capabilities
  - Snapshot/restore, cloning, migration, …
  - Run different OSes on the same machine at once

Virtualization: a familiar example



- Who has not seen this before?!
- Like having two (or more) hard drives
  - you get to choose the sizes
- *Why* is this useful?

How does partitioning work?

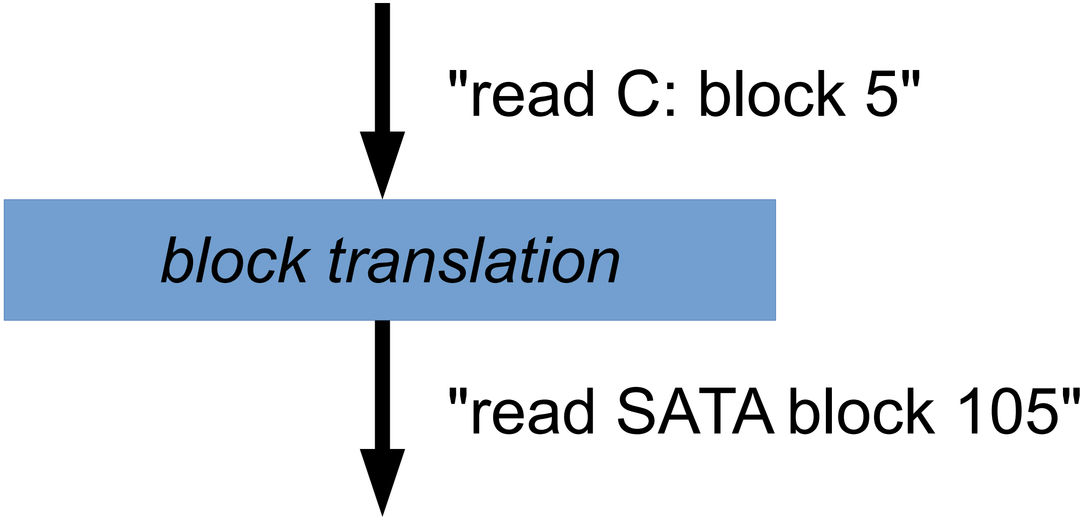


Partition table

	start offset	*end*
1	100	199999
2	200000	500000

- Partition table is an example of ***metadata***
- When the OS wants to access the Nth block, the real disk access is block (N+offset)

Implementation: translation layer

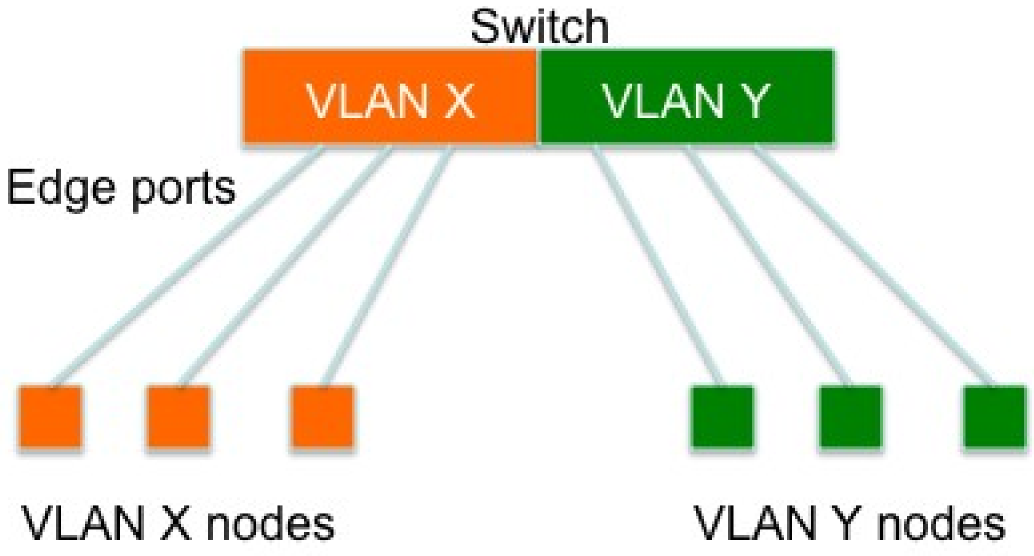


- Very simple and fast: just add offset
- Data is contiguous on disk
- Moving/resizing a partition can require copying all the data on the disk :-(

Another example

- Virtualize a switch: VLANs
  - like dividing a switch into separate switches
- Benefits:
  - can keep traffic separate (broadcast domains)
  - can create VLANs and how they are assigned to ports, purely through software configuration
  - can combine VLANs onto a single cable and split them out again (tagging/trunking)

VLANs

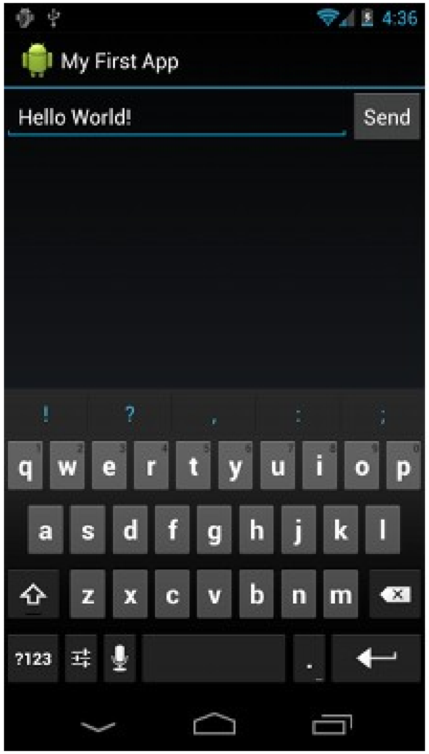


Emulation

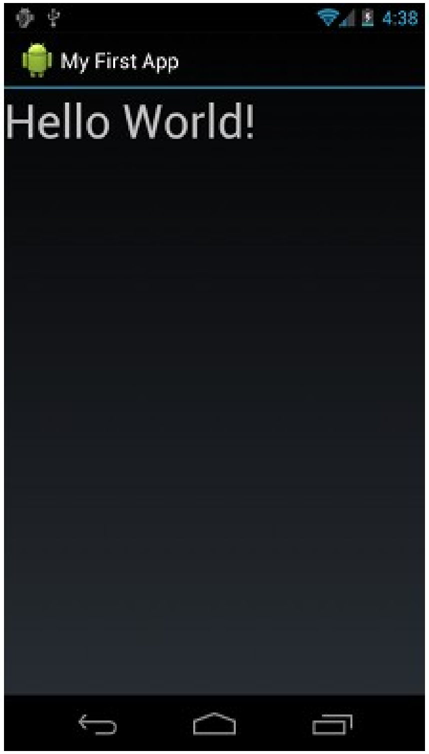
- In software, you can simulate the behaviour of a device which doesn't exist
- Example: emulation of a CD-ROM drive using an ISO file
  - a request to read block N of the (virtual) CD-ROM drive instead reads block N of the ISO file
  - similar to partition mapping
- You can simulate any hardware - including the CPU or an entire system!

Entire system emulation - examples

- Android SDK
  - Emulates an Android smartphone with ARM CPU
  - The "screen" is mapped to a window on your PC



Hello World!



Hello World!

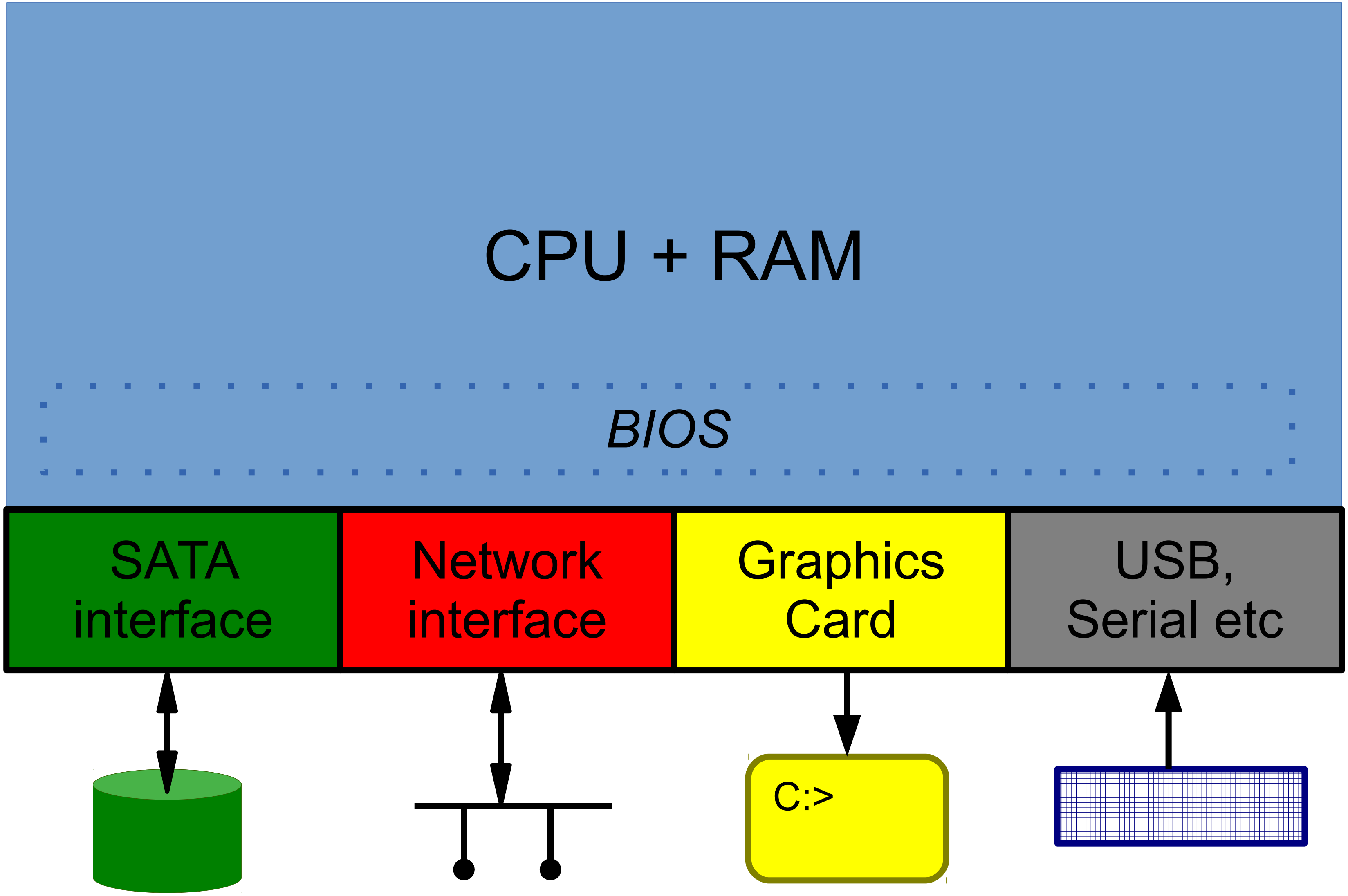
Emulated devices

- There is no physical phone hardware
- So when the software executes an instruction which tries to write to the "screen", this is intercepted and does something else
- It instead updates a buffer in memory which then gets drawn in a window
- The software running inside the emulator is unaware that this is happening

More system emulation examples

- Dynamips / Dynagen / GNS3
  - Emulates a Cisco router with MIPS CPU and network interfaces
- QEMU
  - Emulates an entire PC (i386 processor and interfaces)

What's in a PC?



CPU + RAM

BIOS

SATA interface

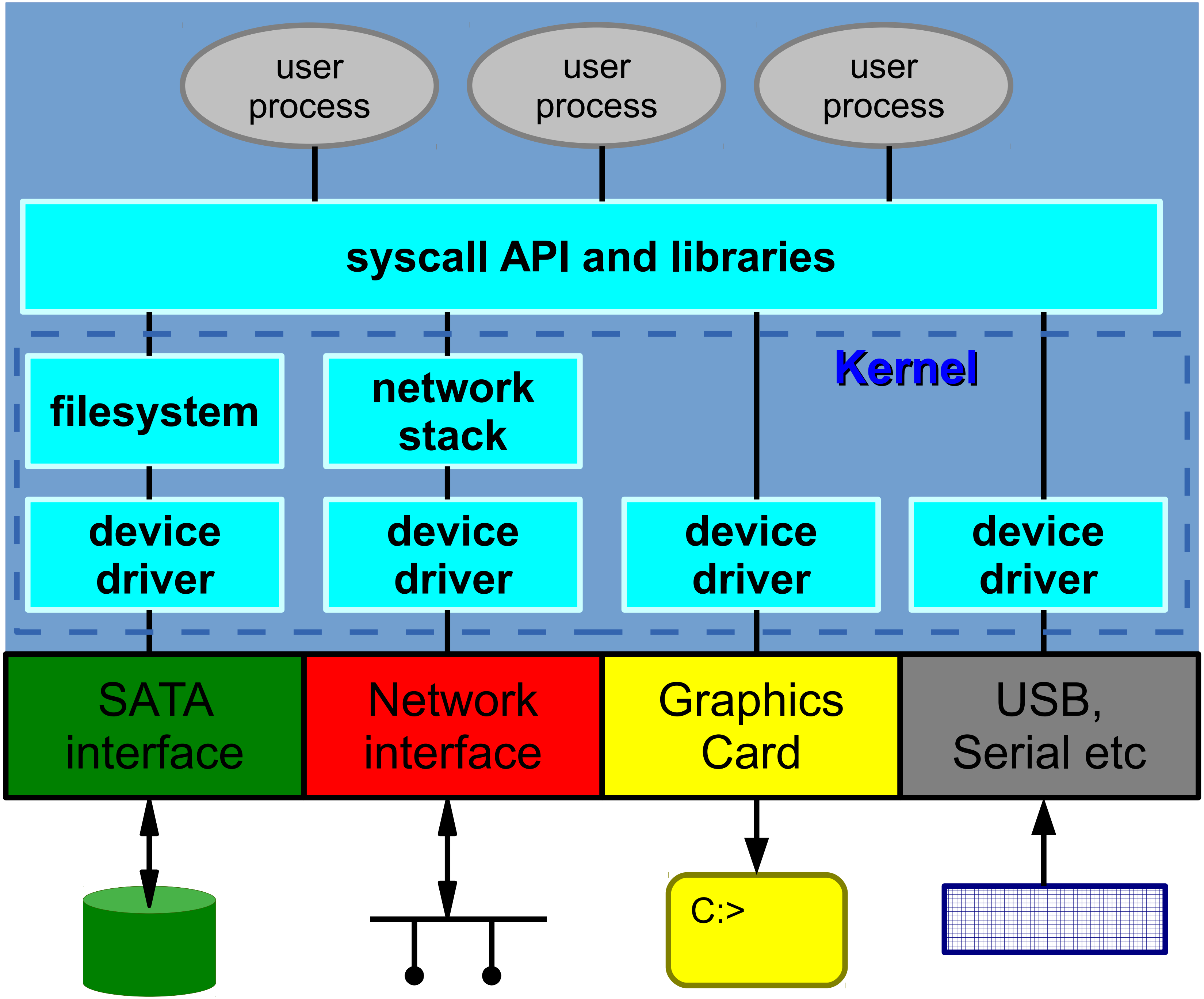
Network interface

Graphics Card

USB, Serial etc

Boot up sequence

- A small program (the BIOS) runs when machine is switched on
- It uses the hardware to load an operating system
  - boot from hard drive, USB/CD-ROM, network...
- Modern operating systems then ignore the BIOS from that point onwards
- The next slide shows a machine after it has booted up (simplified)

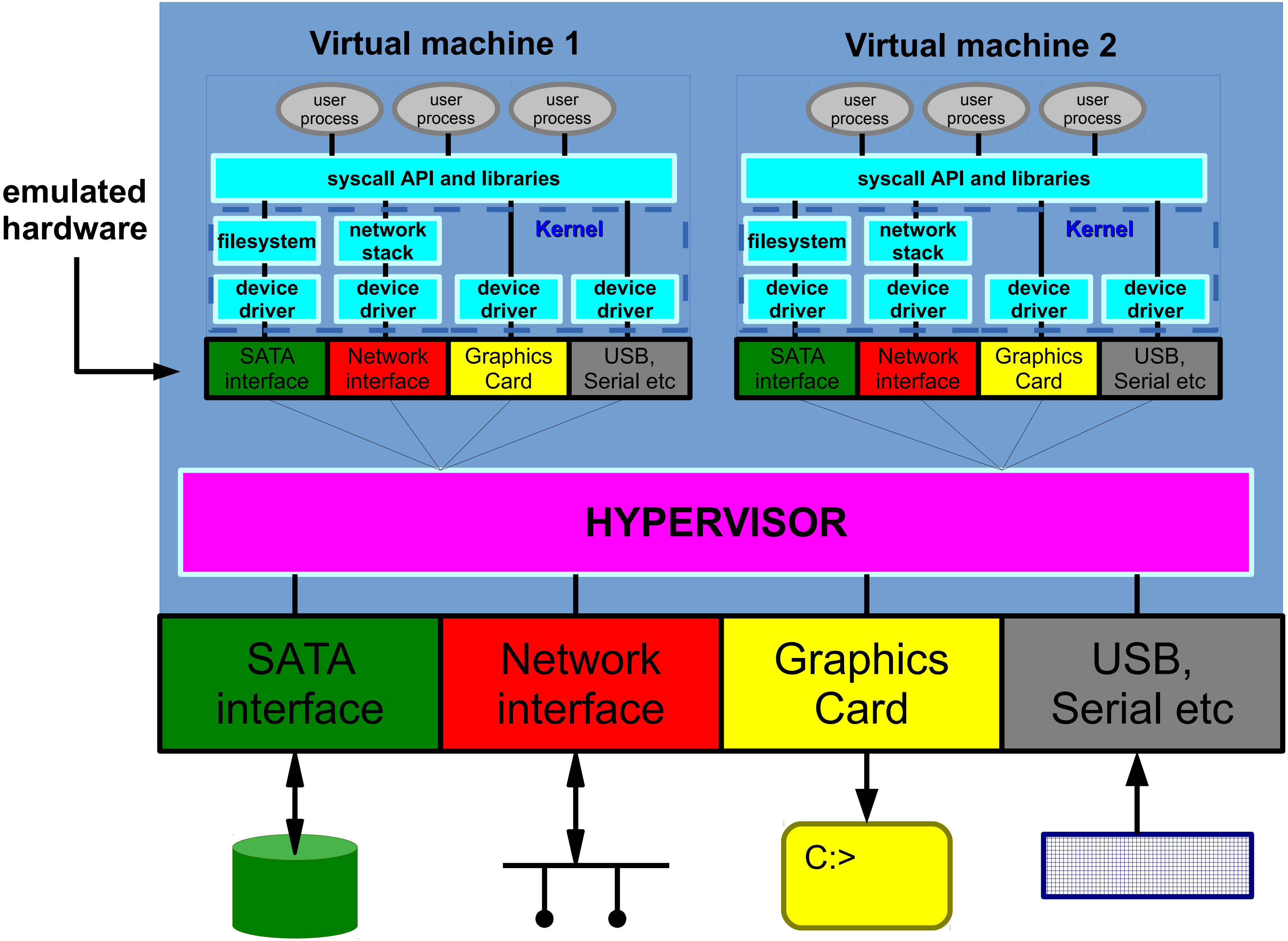

user process
user process
user process
syscall API and libraries
Kernel
filesystem
network stack
device driver
device driver
device driver
device driver
SATA interface
Network interface
Graphics Card
USB, Serial etc
C:>

Points to note

- The device drivers in the OS interact with the hardware
- User processes are forbidden by the OS from interacting directly with the hardware
  - the OS configures protection mechanisms to enforce this

What we need

- To emulate a PC we must emulate all the components of the PC
  - hard disk interface, network card
  - graphics card, keyboard, mouse
  - clock, memory management unit etc
- We want multiple instances to co-exist and not be able to interfere with each other
  - access to memory must also be controlled
- The software to do this is called a *hypervisor*

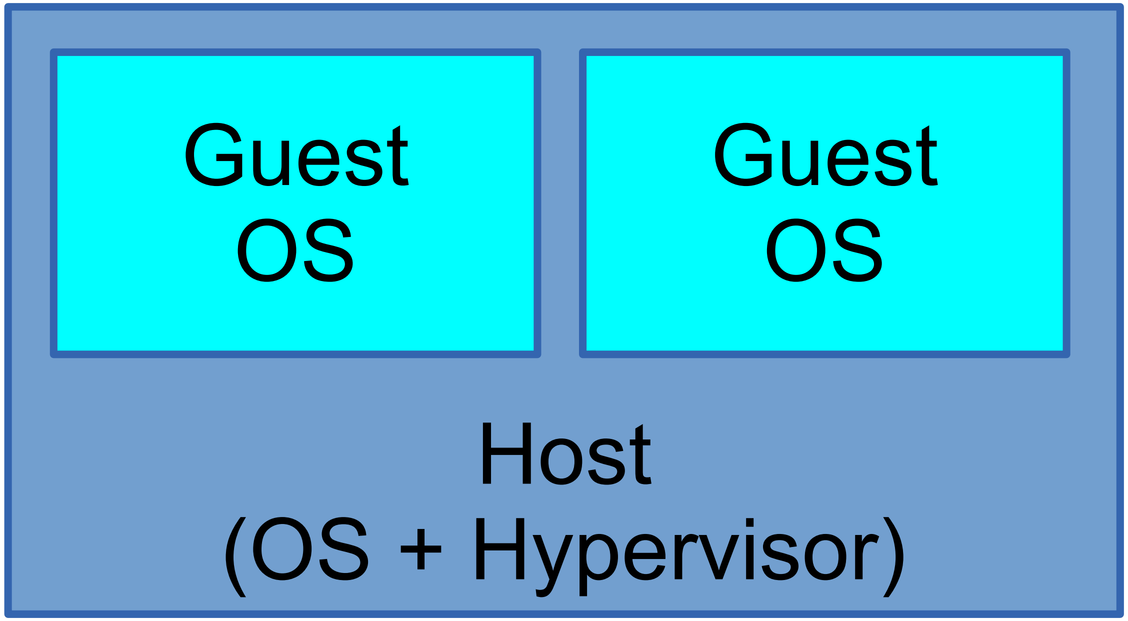

Virtual machine 1
Virtual machine 2
emulated hardware
user process
user process
user process
syscall API and libraries
filesystem
network stack
Kernel
device driver
device driver
device driver
device driver
SATA interface
Network interface
Graphics Card
USB, Serial etc
user process
user process
user process
syscall API and libraries
filesystem
network stack
Kernel
device driver
device driver
device driver
device driver
SATA interface
Network interface
Graphics Card
USB, Serial etc
HYPERVISOR
SATA interface
Network interface
Graphics Card
USB, Serial etc
C:>

Virtual Machines

- Each emulated PC is a "virtual machine"
- Hypervisor allocates some real system RAM to each VM, and shares the CPU time
- Hypervisor emulates other hardware, e.g. disk and network interfaces
- Within each VM you can boot an operating system
- Full hardware virtualization means different VMs can be running different OSes

Virtualization terminology

- The host is the machine running the emulation
- The guest is the emulated (virtual) machine
- One host could be running many guests



The Hypervisor

- Note that the Hypervisor itself is a component of an operating system *
  - It needs device drivers, a filesystem, a network stack for remote management, etc
- So there is a host OS for the hypervisor, plus guest OSes

* Even so-called "bare-metal" or "Type 1" Hypervisors include a cut-down operating system

Emulated disk hardware

- A hard drive is a "block device"
  - OS makes requests like "read block number 42", "write block number 99"
- Real hard drives have a fixed size!
  - This is what the guest OS will expect to see
- So the hypervisor must redirect these accesses to something else

Emulated disk hardware

- Options include:
  - a disk image file on the host (simple)
  - a partition or logical volume on the host (faster)
  - a remote file or remote block device (via network)
- A disk image file is easy to backup and transfer from host to host
- There are different ways to make a disk image file. Suppose we want the guest to see a 10GB virtual hard drive?

Options for a 10GB image file (1)

- A "raw" file is a just a plain 10GB data file
  - Nth block of the virtual hard drive corresponds to the Nth block in the image file
  - if this is allocated up-front, you use 10GB of (hopefully) contiguous space on the host
  - Fast in operation, avoids fragmentation on the host
  - Wasteful of space
  - Slow to create
  - Slow to copy

Options for a 10GB image file (2)

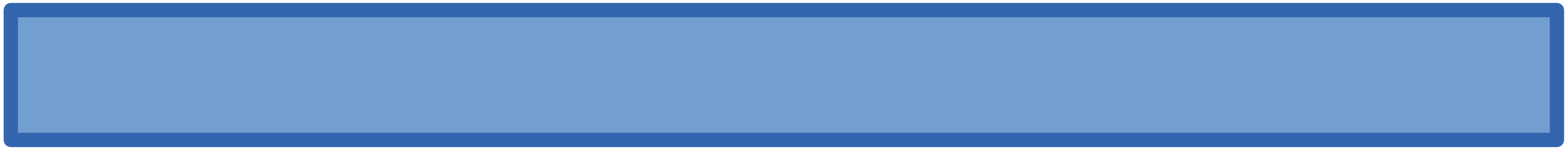
- Some OSes support "sparse" files or "holes"
  - still looks like a plain 10GB file
    - doesn't allocate space until each block is written to
    - reading from unallocated space reads zeros
    - the size of the file ("ls -l") is larger than the disk space used by the file ("ls -s" or "du")
  - can lead to fragmentation
  - can lead to failures if filesystem becomes full
  - if you are not careful, may expand to full 10GB when you copy it

Options for a 10GB image file (3)

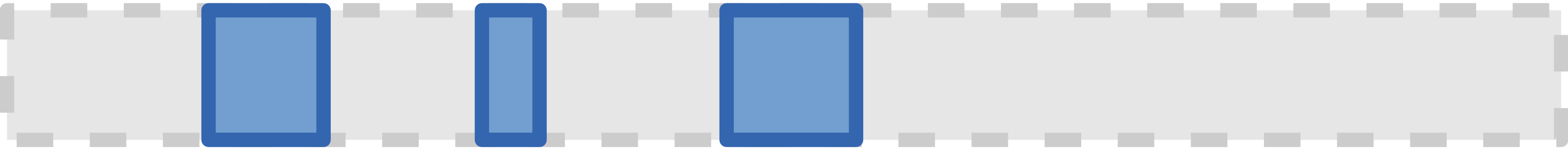
- Custom VM image format with header and data
  - doesn't require OS support for sparse files
  - can be copied without losing its "sparseness"
  - also leads to fragmentation, unless you pre-allocate all the space
- Various formats, e.g. VDI (virtualbox), VMDK (VMware), QCOW2 (qemu/kvm)
- Also add features like snapshots

Disk image types

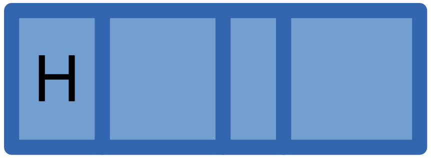
Raw file (preallocated)



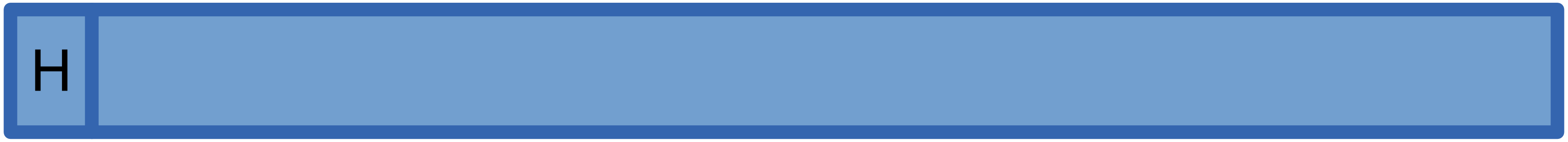
Raw file (sparse)



Growable image file



Pre-allocated image file



Emulated network hardware

- Each guest NIC gets a fake MAC address
- Different ways to interconnect with host NIC
- "NAT": outbound packets translated to share the host's IP address
- "Bridging": packets sent/received untranslated over the host's physical NIC
  - Each VM gets its own IP address on the ext network
  - More transparent
  - Does not always work on wireless NICs though

Summary

- Virtualization can make better use of your hardware by emulating more machines than you really have
- The emulated environment is provided by a <u>hypervisor</u>
- The hypervisor (host) lets you start up virtual machines (guests) each with its own operating system and emulated devices
- Guest hardware emulated using resources on the host