Virtualization and Performance

NSRC

Overhead of full emulation

- Software takes many steps to do what the hardware would do in one step
- So pure emulation (e.g. QEMU) is slow
  - although much clever optimization is done
- One obvious choice: if the CPU of the guest is the same type as the CPU of the host, we would prefer the CPU to run the code directly
- But we must also intercept those points where hardware is accessed

Hardware support

- CPU vendors have added support to make virtualization more efficient
  - Intel call it "VT-x", AMD call it "AMD-V"
- Needs support from both the CPU and motherboard
  - you may need to enable it in the BIOS settings
- Most hypervisors work better when this is available
- Some hypervisors won't work without it (KVM)

Paravirtualization

- Guest OS is modified to be aware of the hypervisor and communicate with it
- Especially reduces the overhead of virtual disk and virtual network access
  - can also add features like "balloon memory"
- Examples:
  - Xen
  - virtio (add-on for disk and network PV)
- You are limited to guests OSes with PV suppo

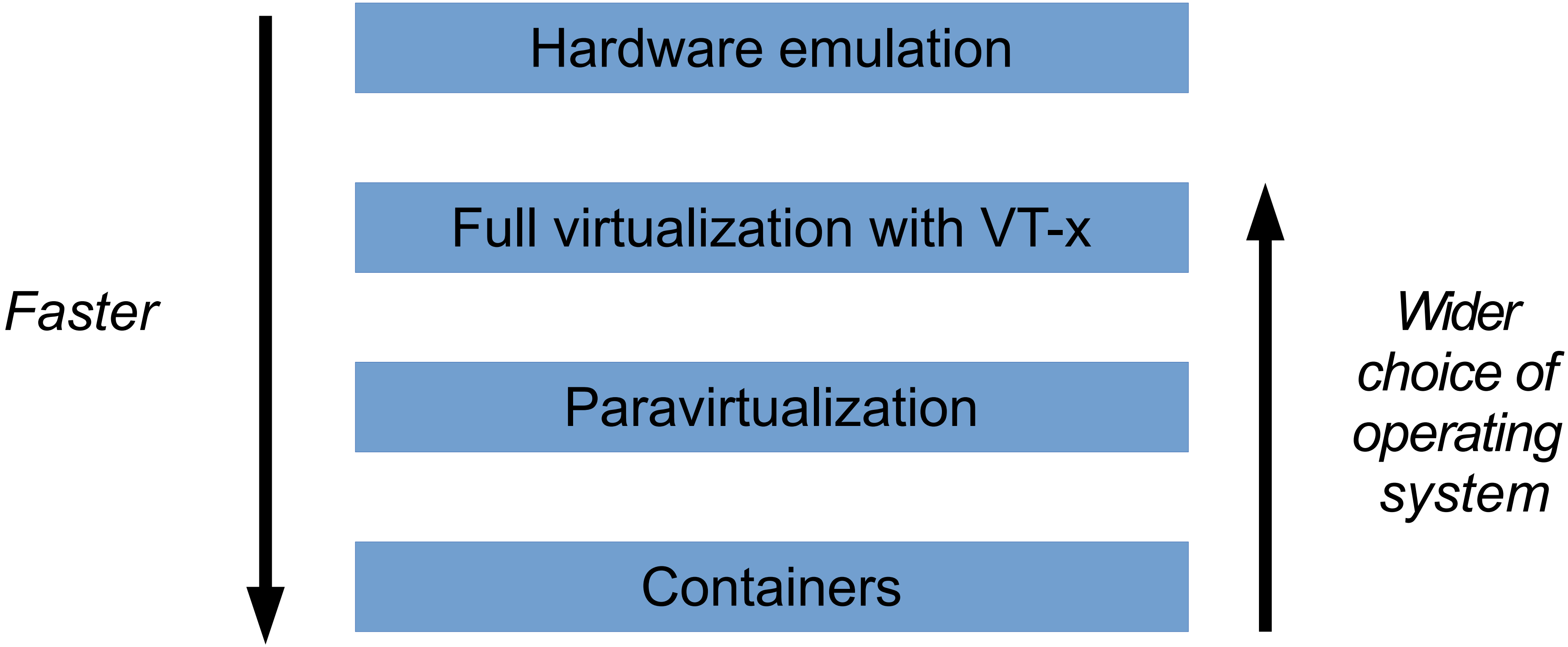
virtio is easy to set up

- Simply configure your hypervisor to use virtio NICs and/or virtio disk interfaces for the guest(s) you wish to speed up
- If the guest supports it, it will boot just fine
- Some things may appear differently
  - e.g. in Linux you may see /dev/vda instead of /dev/sda

Containers (OS level virtualization)

- Forget hardware emulation completely
- Single OS, single running kernel
- Kernel modified to provide separate filesystems, network stacks, PIDs etc
- Examples:
  - Linux: LXC, OpenVZ, Vserver, Docker
  - FreeBSD: Jails
  - Solaris: Zones
- Very efficient, but less isolation

Comparison



Faster

Hardware emulation

Full virtualization with VT-x

Paravirtualization

Containers

Wider choice of operating system

Limitations of the host hardware

- Virtualization doesn't magically make your hardware work faster!
- You will be increasing the load on your hardware by running multiple VMs
- Often the major bottleneck is disk I/O

Disk limitations

- A hard drive is "spinning rust"
- By far the slowest part of the computer
  - 7200rpm drive = 120 revs per second = 8.3ms/rev
  - Typical 100MB/s transfer rate = 10ms per megaby
  - Data transfer will require seeking the head, then (average) half a revolution, then the transfer
  - Expect only 100-200 operations per second!
- Many small transfers much worse than few large transfers

Increasing disk performance

- Buy faster hard drives (e.g. 15K RPM)
- Install multiple hard drives
  - They can be independently seeking to different da
  - Allows more concurrent accesses
- Use SSD (expensive, limited life?)

Beware parity RAID

- A single write on RAID5/6 requires multiple reads, parity calculation and multiple writes
- Don't use RAID5/6 if you care about write performance!
- Use RAID10 instead
  - Striping with mirroring - no parity accesses
  - But requires more disks (2 x required storage)
- ZFS an option to consider

Network bandwidth

- Some people put their data on remote storage
  - Remote filesystem: e.g. NFS
  - Remote block device: e.g. iSCSI, nbd
- The network can then become a bottleneck
- 1Gbps network = only 100MB/s max
- Use a separate storage NIC
- Tune MTU=9000 ("jumbo frames") on the storage LAN if your NICs/switches support it
- Consider 10G networking

RAM

- Each guest expects to have a certain amount of RAM to itself, so make sure you have enough RAM in total
- Host swapping to disk is a no-no
- Some clever tweaks possible
  - e.g. Linux ksmd: kernel shared memory daemon
- Far better not to overcommit your RAM in the first place
- RAM is (relatively) cheap, but do use ECC/parity memory for reliability

Summary - choosing hardware

- Choose servers with VT-x or AMD-V support and 64-bit processor
- Buy enough RAM for all your VMs combined
- Install multiple hard drives
  - but don't use RAID5 or RAID6
- Multiple NICs are useful
  - e.g. separate management network,disk transfer network, and service network
  - can also bond for redundancy/load sharing