CS 3214 lecture #27 "VMs & containers"

SPOT ←

elastic compute cloud

Cloud computing characteristics ← EC2

- elasticity ← granularity is shrinking
- mutli-tenant ← statistical multiplexing
- need for isolation - "private view" of system

How do we get private view?

Process: address space
CPU

what isn't private?

- filesystem — rm /lib /etc/config
- sockets: port numbers
- pids are shared — ps

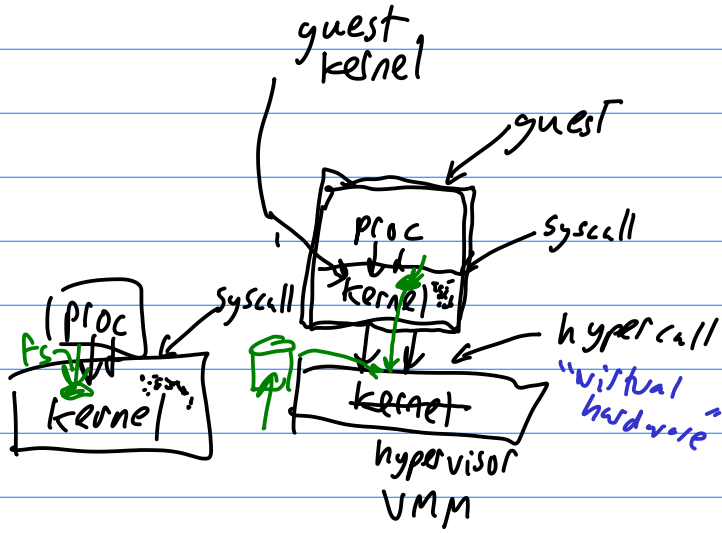
process is not a great abstraction for cloud

Alternatives

- bare metal
- virtual machines
- containers

Virtual machine

- virtual disk has everything app needs
- guest kernel (nothing shared)
- hyp/VMM multiplexing



1970s: Goldberg 1972, 1974
- improve test OS
- developing H/W diagnostics
- running different OSes / versions

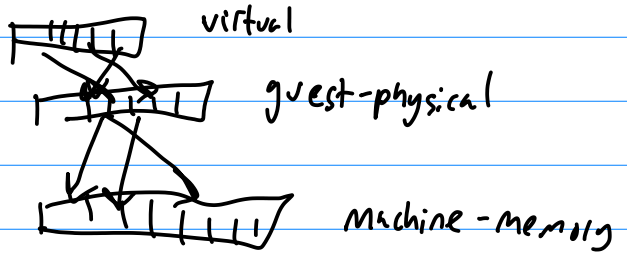
Popek / Goldberg requirements
- equivalence / fidelity
- resource control
- efficiency (most inst. need to execute directly on H/W)

How to run VM?
- basic idea: "deprivilege" (run in user mode)
- trap and emulate
  q: will all (privileged) instructions trap

  IBM/360 yes
  x86 (prior to VT-x) << no

Early 2000s
- modify guest: Xen paravirtualization (2002)
- dynamic binary translation: VMware (1999)
- fix the hardware: VT-x (2005/2006) AMD-V

lots of classic mechanisms to rethink
- Memory mgmt
- fixed RAM, CPU
- device I/O
- :



"Heavyweight" - image
- 2 kernels are running
- I/O

Containers: the OS already does this!

Kernel mechanisms
- chroot "jail"
- use some other dir as / (for proces + children)

Namespaces
- UTS - hostname
- PID
- mount
- user
- IPC
- network
- :

cgroups
- limit resource consumption
- CPU
- mem

clone()
unshare()

container = set of proceresses sharing - chroot
- ns
- cgroups

layered filesystems

kubernetes CNCF

microVMs