Virtualization

&& (and)

|| (or)

^^ (exor)

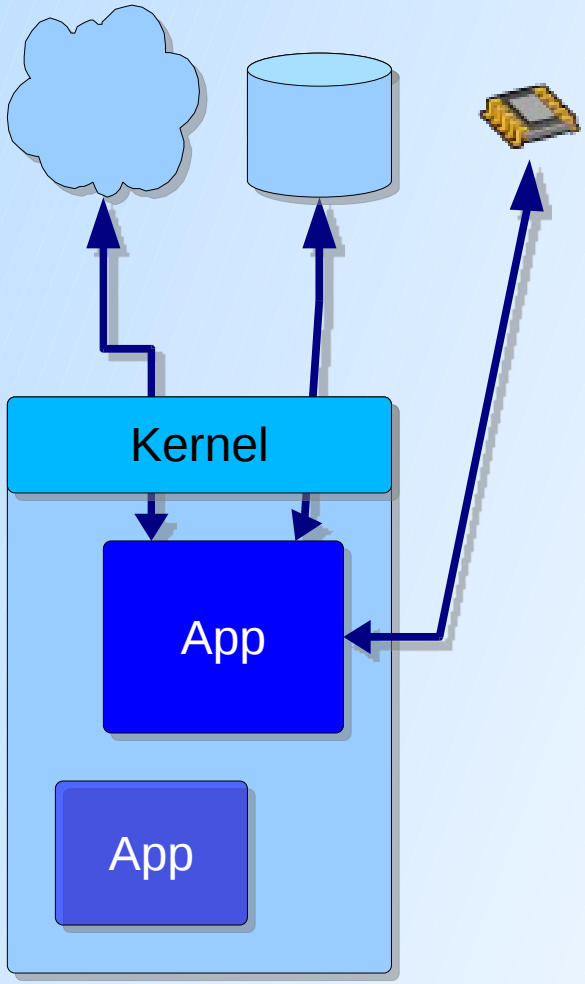
Security

Virtual use cases

- (see “your standard intro to VMs” for *hypervisor, guest, dom0, domU, image, Type-1, Type-2, para* vs *full,* ..); not talking about “OS compartments”, chOS, emulators etc – but concepts are similar. And “Virtualization” is an old concept...
- General “virtual” use cases:
  - Efficiency – pool underused machines
  - HA – live migration, “hide” interventions
  - Load balancing - “elastic” supply
  - Security: segregation, containment, honeypots
- HEP-specific: (see the other talks..)
  - Batch job segregation (CHEP – KIT/HTW)
  - Analysis environment, “VO appliance” (ATLAS/PANDA)
  - “Clouds”, User-supplied images (CERNVM)?

Usual OS diagram

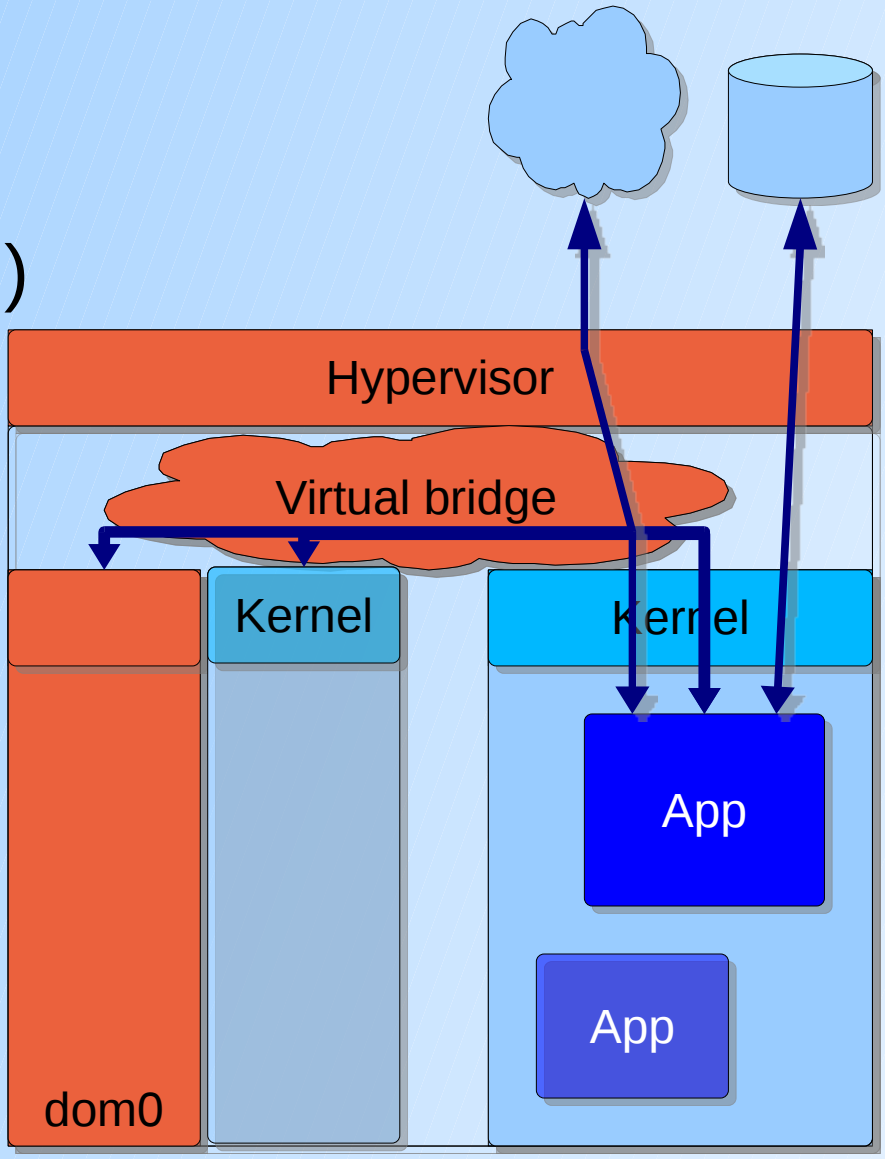
- Traditional kernel:
  - Control HW access
    - CPU (timesharing)
    - (virtual) memory
    - Disk (File system)
    - Devices (net, console, video, audio etc)
  - IPC/signals
  - Low-level network
  - Identity mgmt (UID/GID)



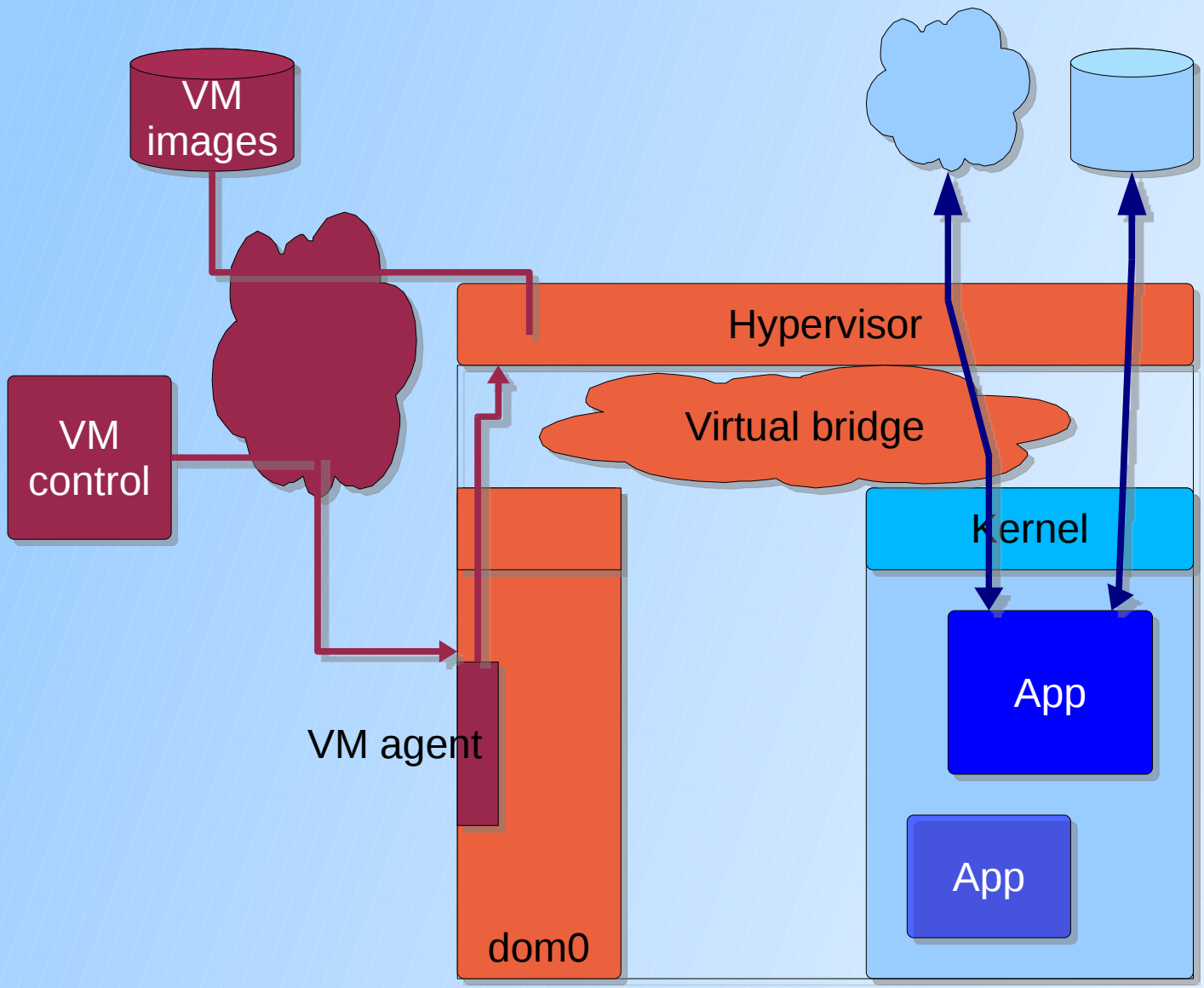
VM-in-a-box

Add:

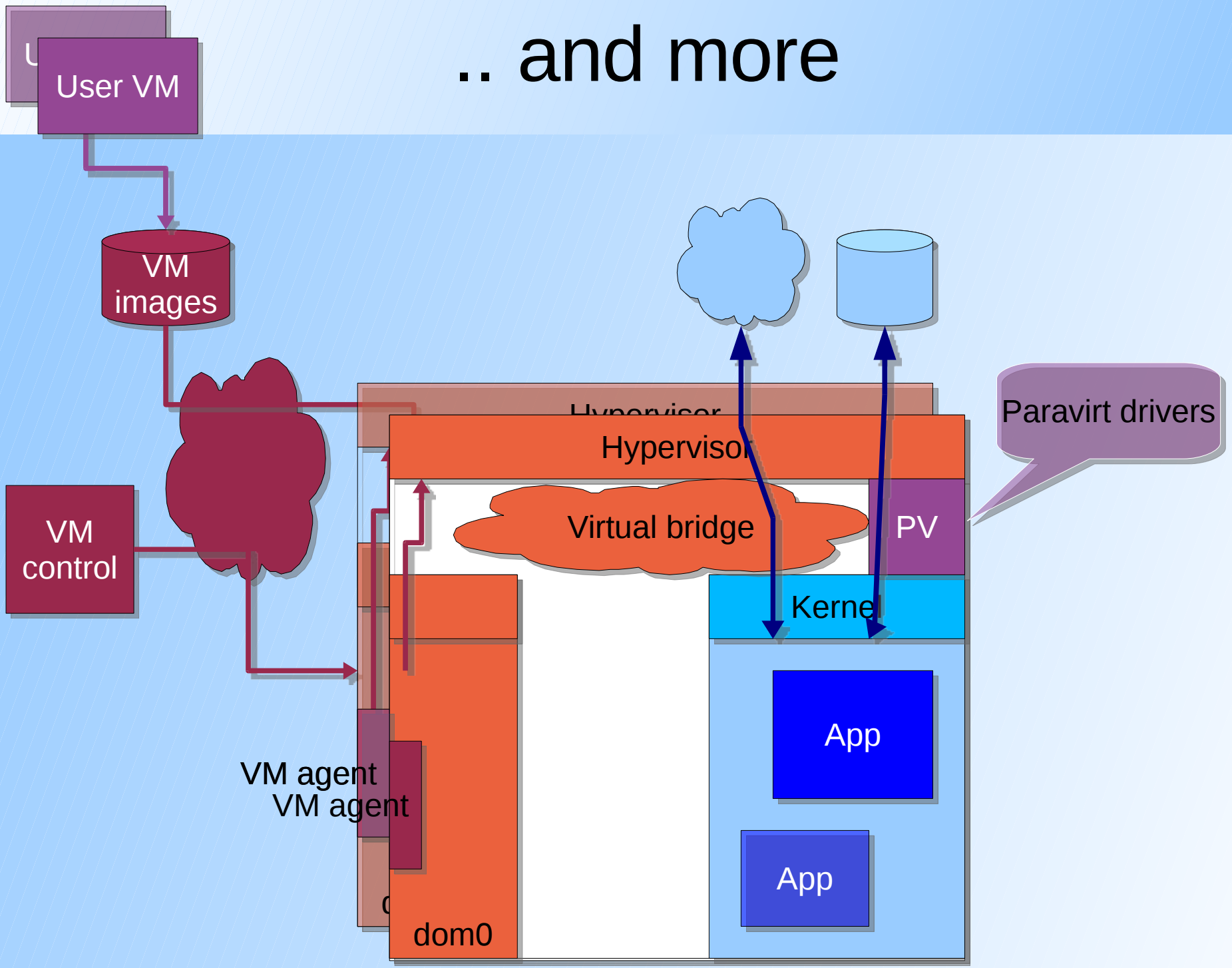
- Hypervisor (HW access)
- Network (virtual)
- “Controlling” dom0



Wait, there is more..



.. and more


User VM
VM images
VM control
Hypervisor
Virtual bridge
PV
Paravirt drivers
Kernel
App
App
VM agent
VM agent
dom0

VMs: “virtual” security

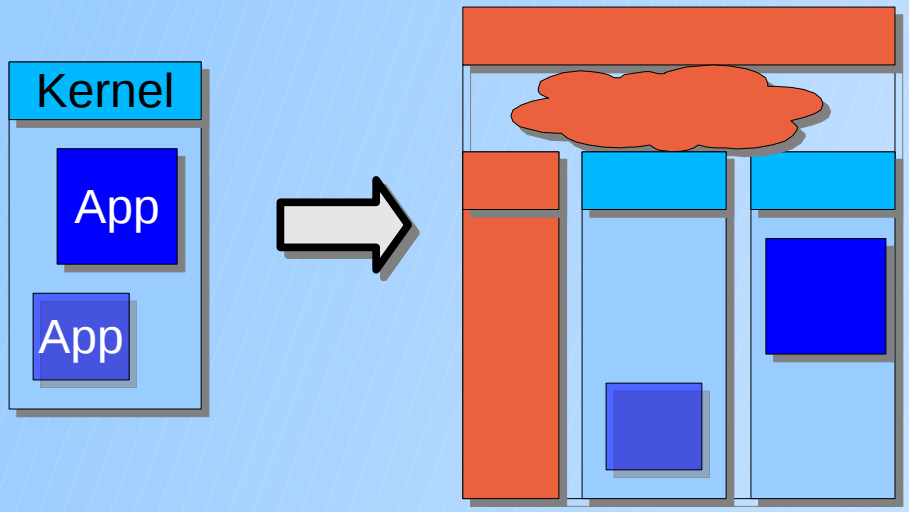
General: new code, new interfaces = new attacks.

- VMs are inherently **less** secure --- all other things being equal
- More code, more interfaces = more “attack surface”
- Typically “all other things” *not* equal.

VMs provide scope for more secure configurations

- Potentially *less impact* of a vulnerability/exploit – containment might offset complexity
- So need to compare before/after ..

Examples

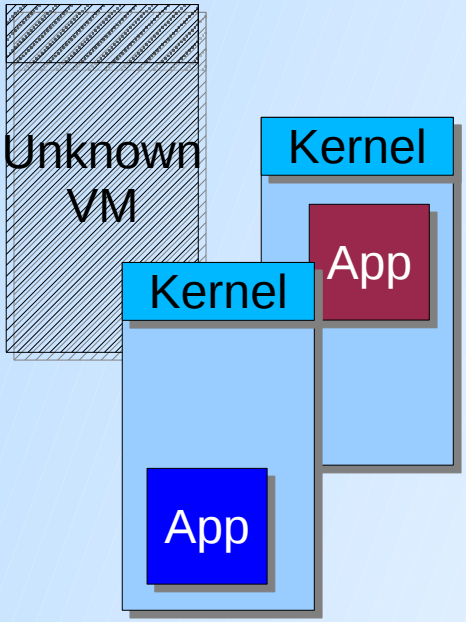


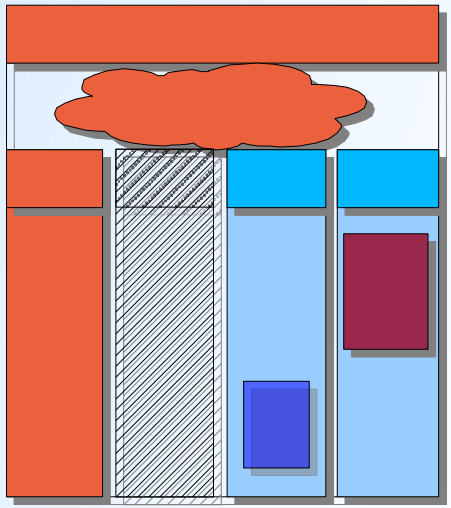
applications from same machine (or trust zone) go into separate VMs

→ ~~more secure~~

applications from different trust zones go into same hypervisor

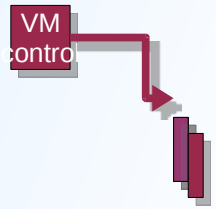
→ less secure





VM-specific attack points

- **VM escape**: break out of VM, into hypervisor or host OS = access to other VMs. (rare)
- "Paravirtual" drivers: (tradeoff: speed vs "virtual")
  - Can attack guest or hypervisor (occasional)
  - Usual assumption: (high-perf/DMA) drivers are trusted
  - Mitigated by IO-MMU
- "VM network":
  - sniffing, IP-spoofing, ARP-spoofing, DoS...
- Controlling infrastructure (?)
  - homegrown or vendor-supplied – remote reset, "elastic" things...



Other security issues around VMs

- Access to VM images
  - Storage/transit: secure? encrypt? (who trusts whom?)
  - Persistent changes to image (or not?) - sandbox?
  - Patching – when, by whom?
    - Potentially easy on trusted guests
    - Unknown on user-supplied = “museum system”

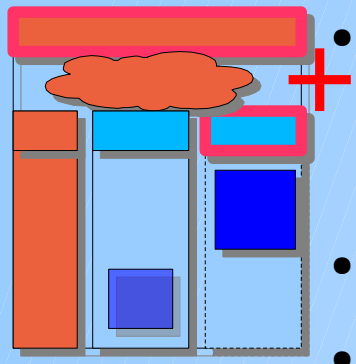
    (Difficult on hypervisors - migration or downtime)
- Snapshot images – full memory dump
- Privileged network access?

“Nobody is using IP-based access controls nowadays!”

Wrong.. LHC-OPN, NFS et.al., site firewalls, DMZs,...

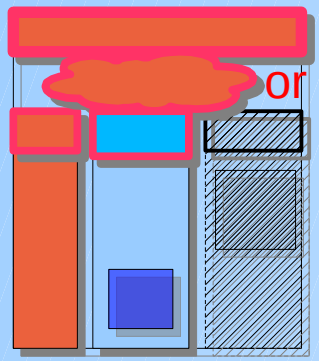
Crucial point: “**trust**” the VM?

- “**Trusted**” VM: → minimal operational changes



  - Admin access restricted, similar to physical machines
    - Can trust kernel access control, UID/GIDs, local logging, network config.. = additional layer
  - operating system is maintained/patched
  - Inventory control/tracking
  - Batch, Accounting, resource quotas work (machine-local tools)
  - (Might contain 3rd-party SW)

- untrusted/**user-supplied** VM:



  - Trust boundary shifts to hypervisor ↔ guest kernel
  - Full access to “virtual network” (bridge/NAT)
  - Batch/accounting/quota tasks shift to hypervisor

Trust, cont.

- trusted-third-party VM? ("experiment")
  - "Historically difficult" to get single SW version from experiments (internal fragmentation = no single "production" version)
  - Different focus: functionality+stability vs security
    - Series of "frozen" releases, patching unlikely
    - Expect "service creep" - maintainer cannot refuse new things
    - Deployment is open issues – push to sites, cache ,..
  - CERNVM-style might be a solution - pull in exp sw as needed

Corollary: how to track "trusted" images over time? Sign+Expire? image+mod-on-boot?

Other security considerations

- (user/job) credentials in the VM?
  - Easy for “trusted” images
    - re-use existing tools, get at runtime from Batch/MyProxy/..
  - Harder for “clouds”
    - (typically pushed to VM user)
- Operational overhead from VMs
  - do your procedures scale x10 ?
  - (asset management / tracking, policy enforcement,..)

(interesting) Incident Investigation Impact (ideas)

- Prevention:
  - Restrictive network access != HEP/Grid ? (various callbacks, experiment monitoring etc)

Patching? Compliance checking?

- Detect: NIDS.
  - HyIDS – hypervisor as (N)IDS – interesting idea

–How to profile behaviour for unknown images?

- Forensics
  - Snapshot: *good* (but: who gets access?)
  - Unreliable local audit: *bad*
  - Yet another dimension: which VM was running?

What are others doing

or: are sites too paranoid?

- AMAZON EC2, FlexiScale, Rackspace etc
  - All take user-supplied images..
  - Well-known AMIs configure “ownership on boot”. Nothing more.
- Model: “hosting service” != HEP computing
  - Usually: passive protection of guest
    - Xen (customized) + paravirt guests; effort on virtual net
    - Inbound firewall on hypervisor (but can open)
    - Good recommendations but “VM **security not their problem**”.
  - Economic incentive against active attacks:
    - $$/network, $$/CPU, take offline, owner's credit card.

Summary

Lots of benefits from virtualization..

- Security perhaps isn't the strongest.

But: also lots of new complexity

- Nothing fundamentally challenging, but ...
- lots of issues to consider ...
- (Ideally before deployment.)

Question

- What is the “HEP site” stance on user-supplied images?

  (“cloud”-style image vs grid-job-style computing)

– Have you seen real-life / production / serious requests for user images? By whom?

  - (individual) Users
  - Experiments

–Will /do you run them?

  - Constraints? Experiences?

Discuss.