Using KVM as a Transparent Hardware Abstraction Layer

David Ahern

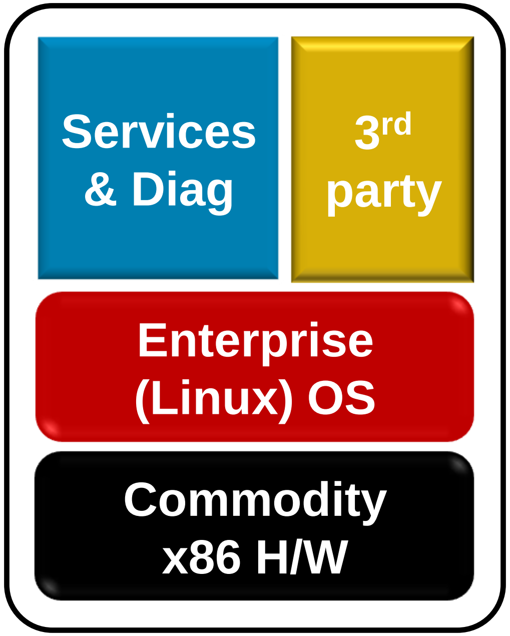
Agenda

- Introduction
  - Appliance-like, enterprise products
  - Component lifecycles – and impacts of their misalignments
- Design Overview
  - A high level look at how KVM is used as a solution for handling the impacts of the short lifecycle commodity H/W
- Design Details
  - A closer look at details such as installation, networking, monitoring, security and performance
- Final Thoughts

Problem Introduction

Appliance-Like, Enterprise Products

- "Bundled" data center solution
  - Product tied to H/W and includes OS
- Closed-box design
  - Product handles firmware & BIOS updates, configures H/W (RAID & BIOS), and includes H/W in monitoring and serviceability
- Commodity x86 Hardware
  - Typical data center servers using Xeon-class processors: e.g., IBM x3250 & x3650, HP DL320 & DL380
- Enterprise Linux OS

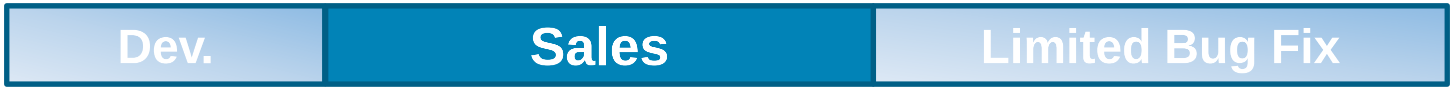


Product Deployment

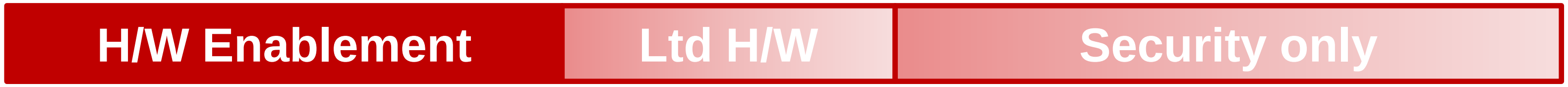
- Customer purchases specific version of product
  - Testing and evaluations (e.g., features & security) in lab environment before going into production environment
  - Time investment with a product version
- Preference to stay on version for several years
  - Significant impact to customer to change versions
  - Minor dot release updates ***might*** be acceptable
- Need for new servers over lifetime of deployment
  - Expansion - add more nodes to expand capacity or increase fault tolerance
  - Server replacement in the case of hardware failures
- Original H/W model deployed may be past end-of-sale
  - Need support for newer H/W generations in “legacy” releases

Independent Component Lifecycles

- Product with desired support for 5 years

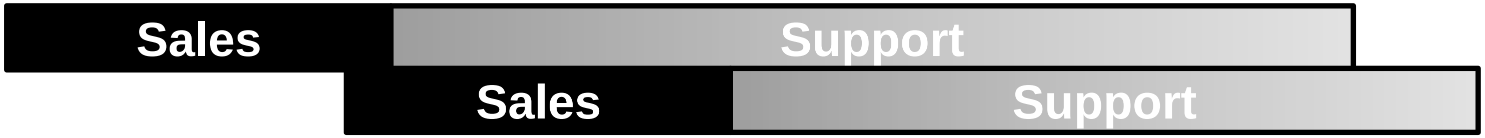

Dev.
Sales
Limited Bug Fix

- Enterprise OS with support lifecycle of 7 years


H/W Enablement
Ltd H/W
Security only

  - New hardware enablement limited to subset of lifetime

- Commodity x86 hardware revisions every 2-3 years


Sales
Support
Sales
Support

  - Support window through OS lifetime, with N, N-1 preference (e.g., support for RHEL4 and RHEL5)

Lifecycle Misalignments

- Lifecycles of the H/W, OS and product will never align
  - Independent vendors, setting timelines based on their goals
- H/W changes cause the most pain
  - New H/W has to be added to product release which means re-spin, maintenance or dot release
  - New H/W typically requires an OS update
- OS updates introduce churn to the product
  - even “minor” ones inevitably have negative impacts
- Hardware and OS are “infrastructure”
  - Changes are essentially cost overhead for product, i.e., profit impacting

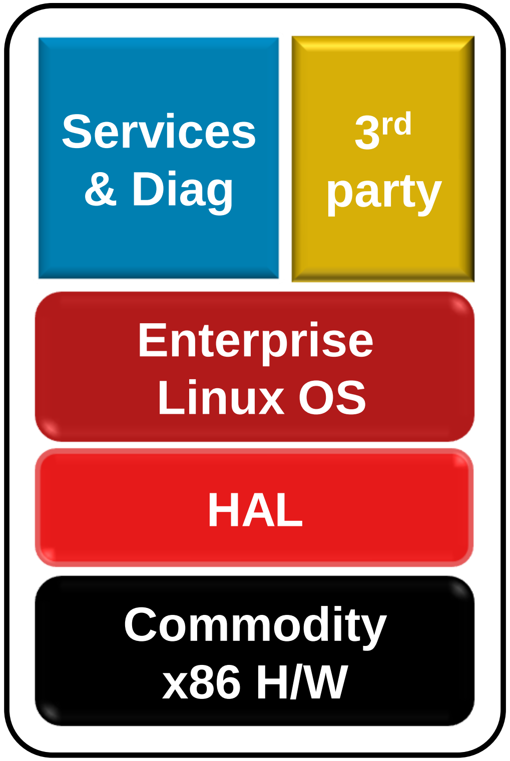
Handling H/W Turnover Within Release

- Lifetime buys of servers
  - Requires guess-timate on sales which will never be right
- Backport OS required for H/W version
  - Non-trivial amount of grunt work to change OS versions, especially if a major version change is required
- Reduce Product lifecycle
  - Shorten sales/support periods
  - Creates unhappy customers
- KVM as a transparent HAL
  - Separates H/W from revenue generating services
  - Use technology to address problem

Design Overview

Separating Product and H/W

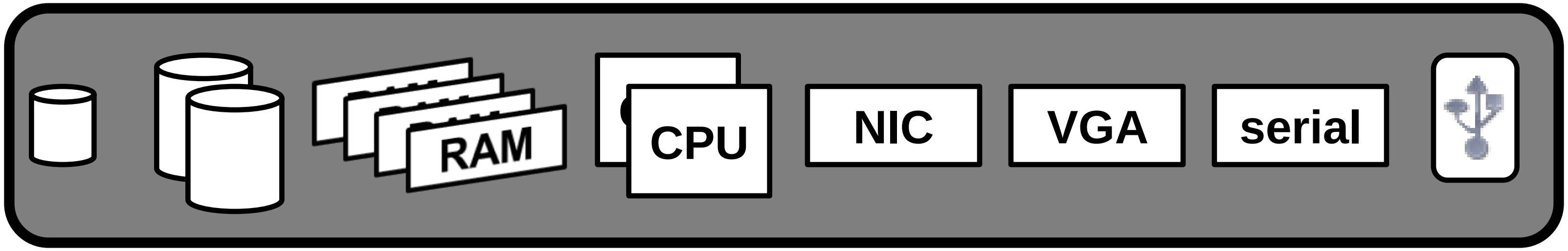
- Leverage virtualization technology to provide an abstraction layer
  - HAL handles H/W
  - OS for revenue generating services sees compatible virtual H/W
- Isolates primary services from hardware
  - HAL is a translator between product OS and H/W
  - H/W and the HAL can change with less churn to the overall product



Hardware Abstraction Layer

- Just another layer below product OS
  - Like a BIOS or firmware
- HAL is a newer version of a Linux OS
  - Compatible with new H/W generation
  - e.g., RHEL5 for HAL, RHEL 3 or 4 for product OS
- HAL == Host OS
  - Use virtualization to run product
  - Focused 1:1 virtualization deployment

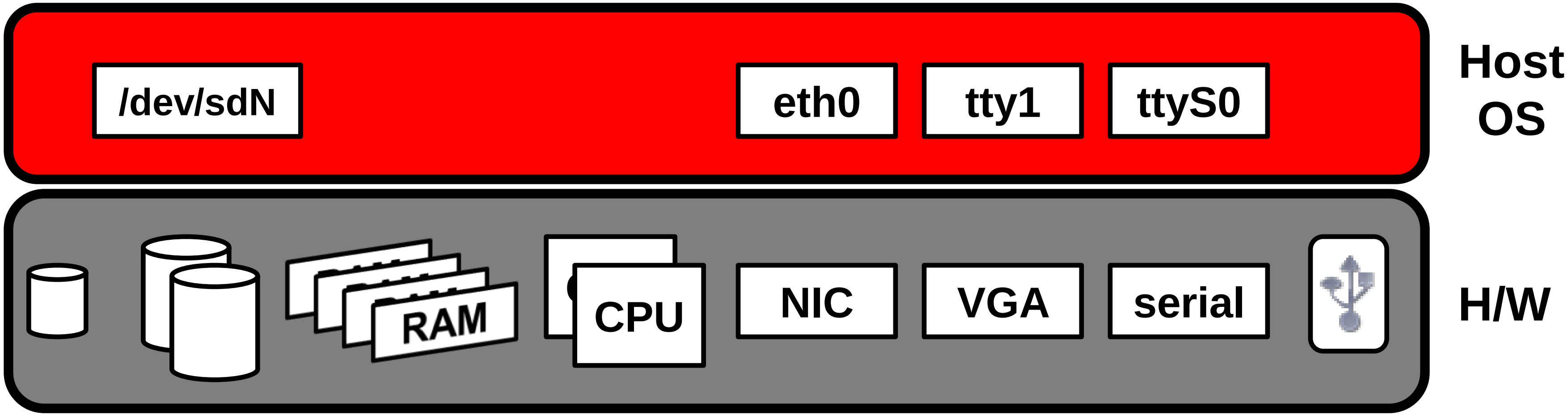
Start with Commodity Server



H/W

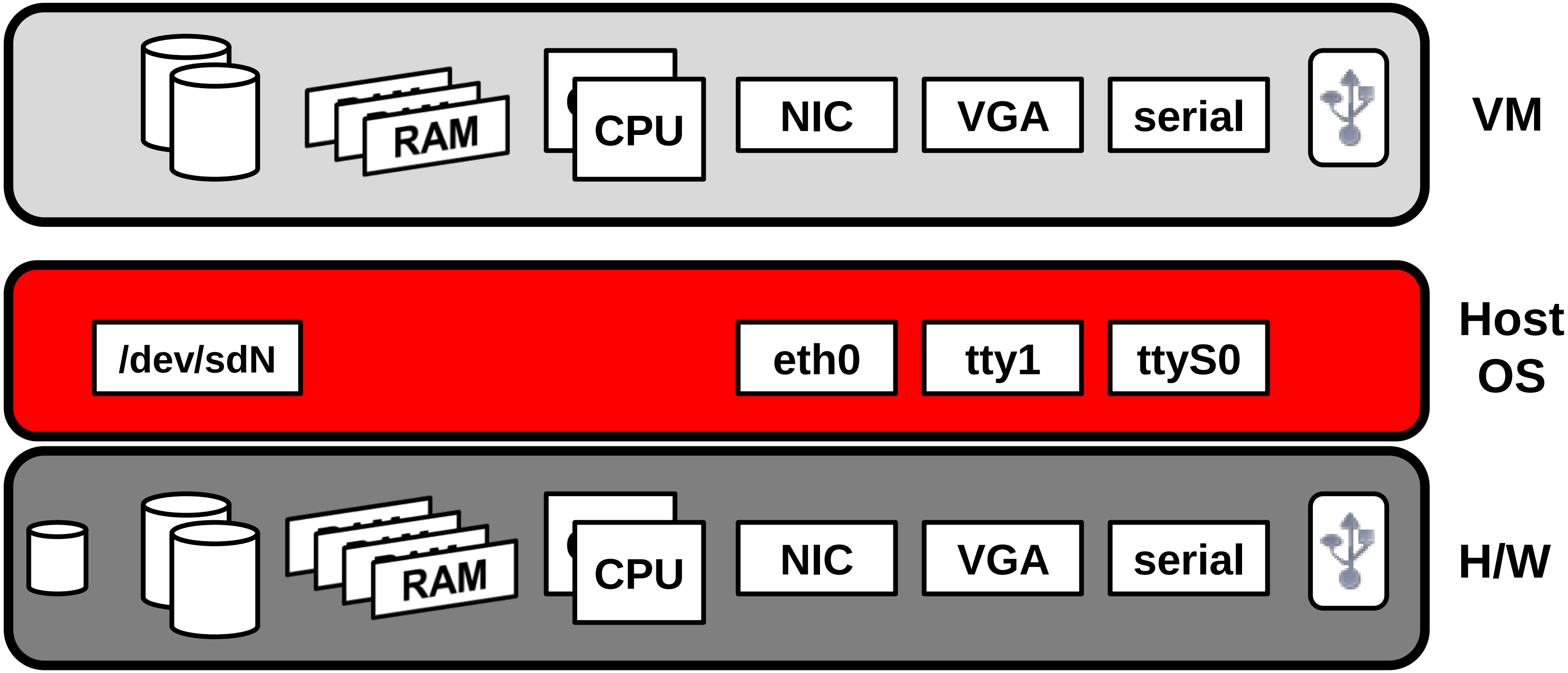
Install Host OS

- Host OS provides drivers to control real H/W



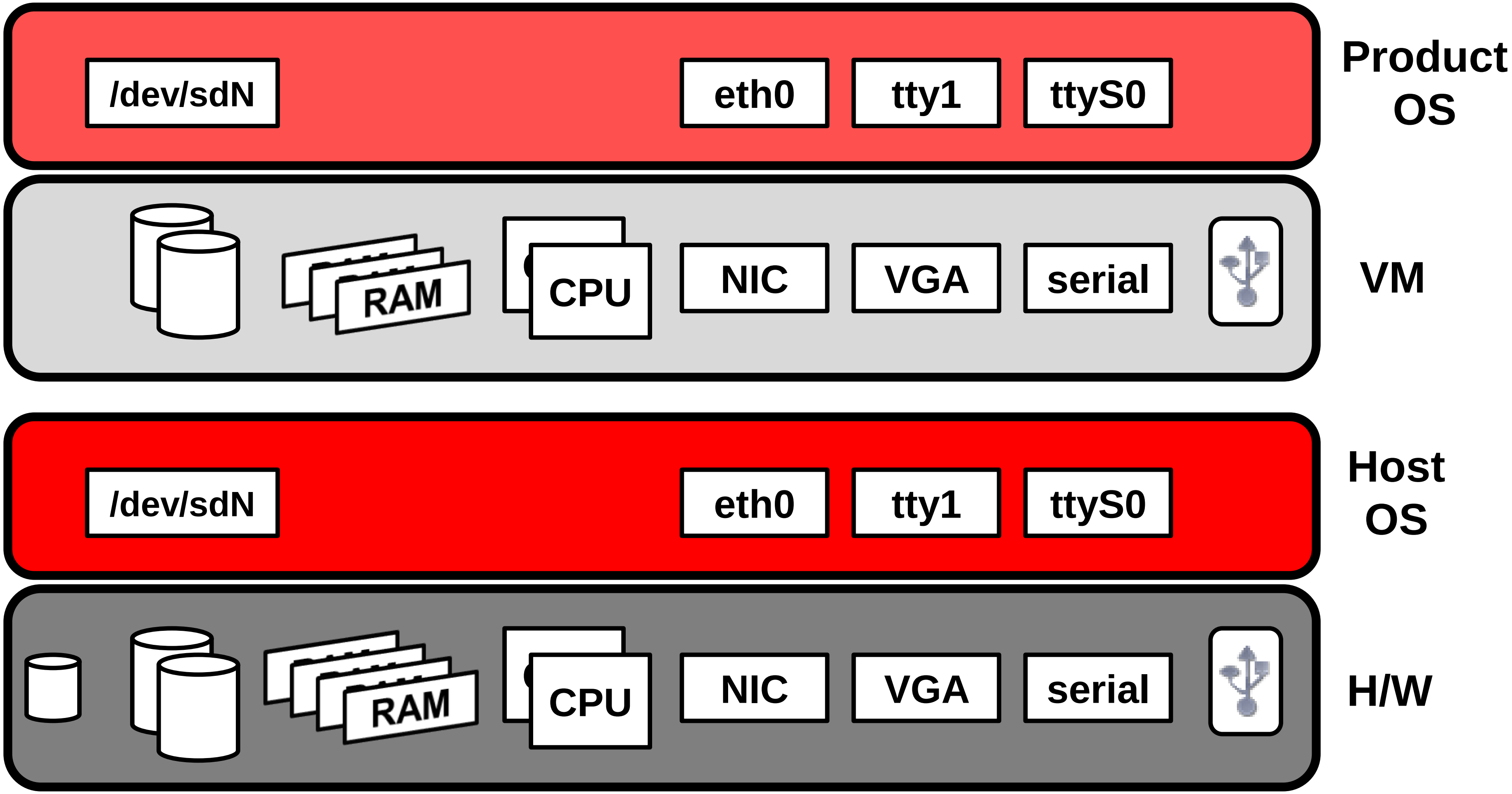
Start Virtual Machine

- Virtual Machine created with H/W “allocation”



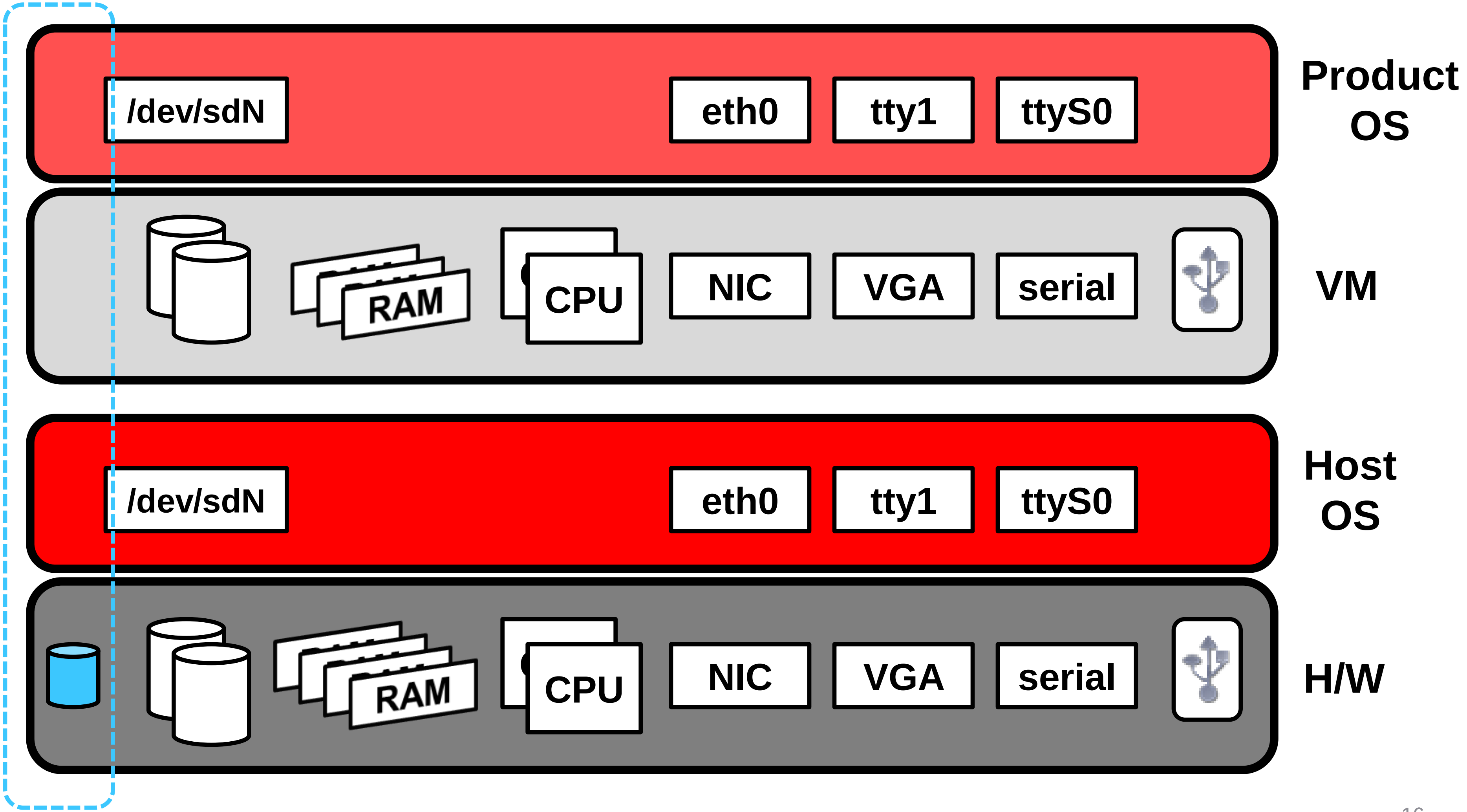
Install Product OS and Services

- Product OS is compatible with VM H/W



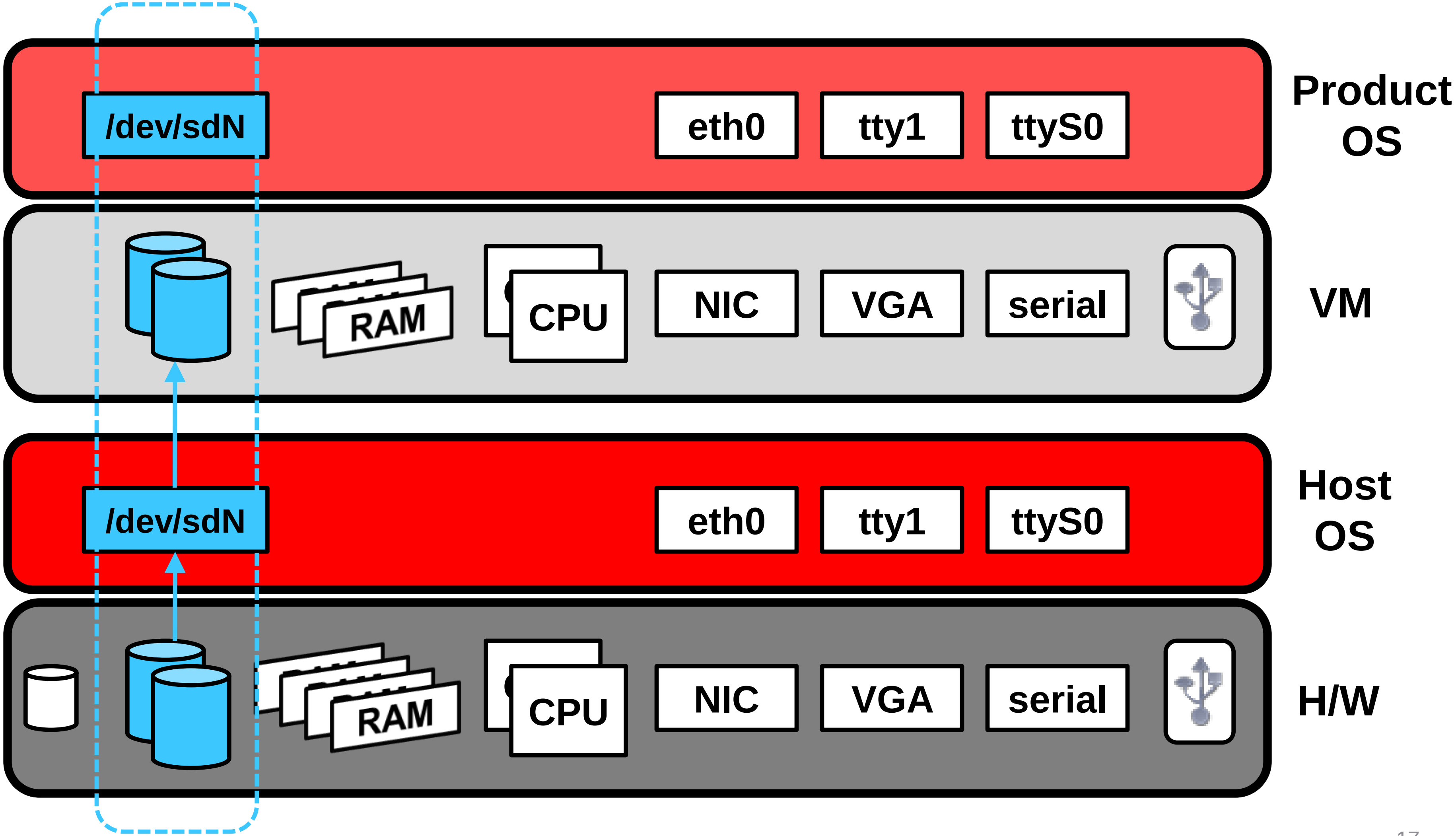
Storage Device for Host OS

- Internal USB key for host OS



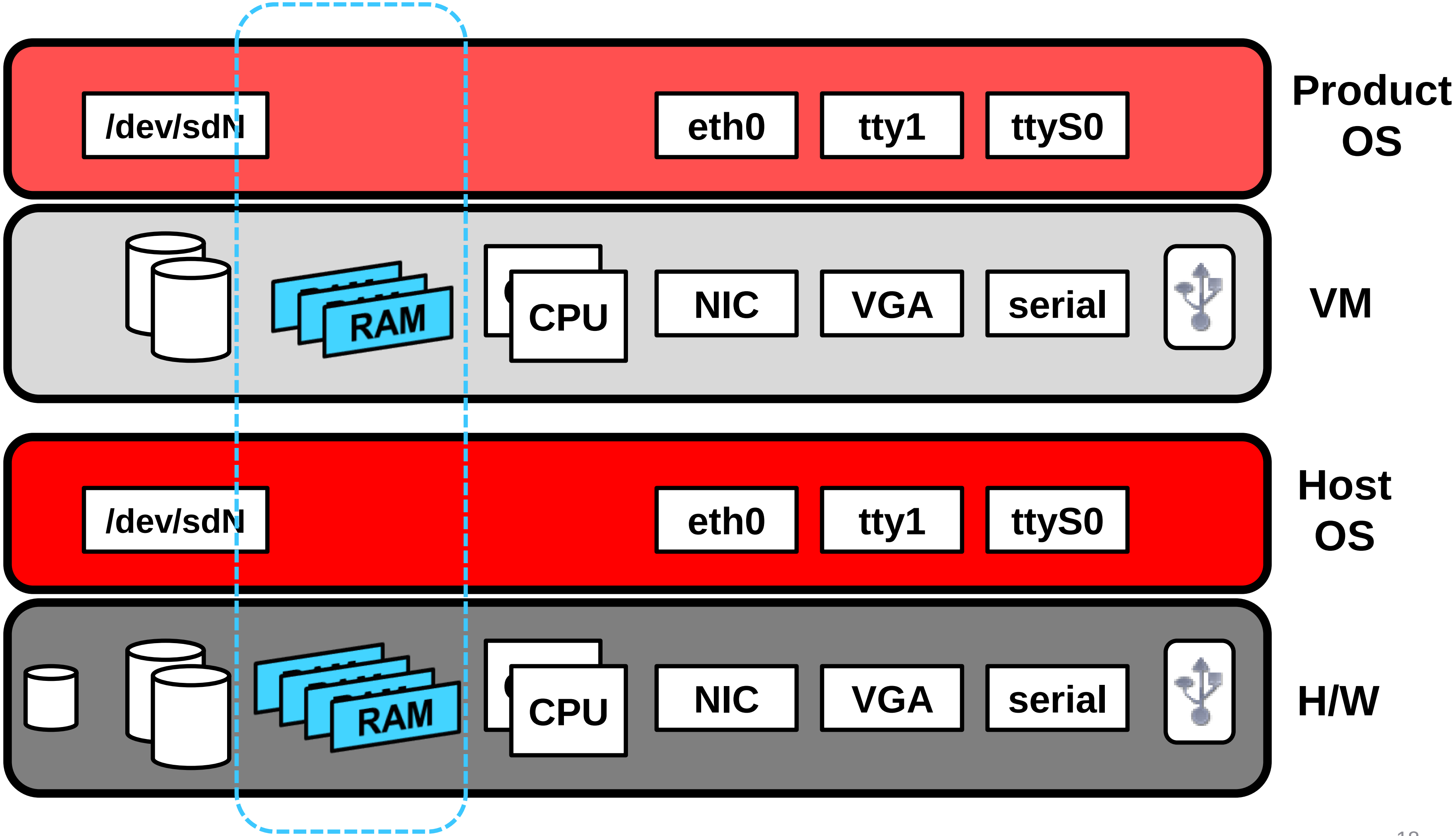
Hard Drives Given to VM

- Block devices in Host OS assigned as disks to VM



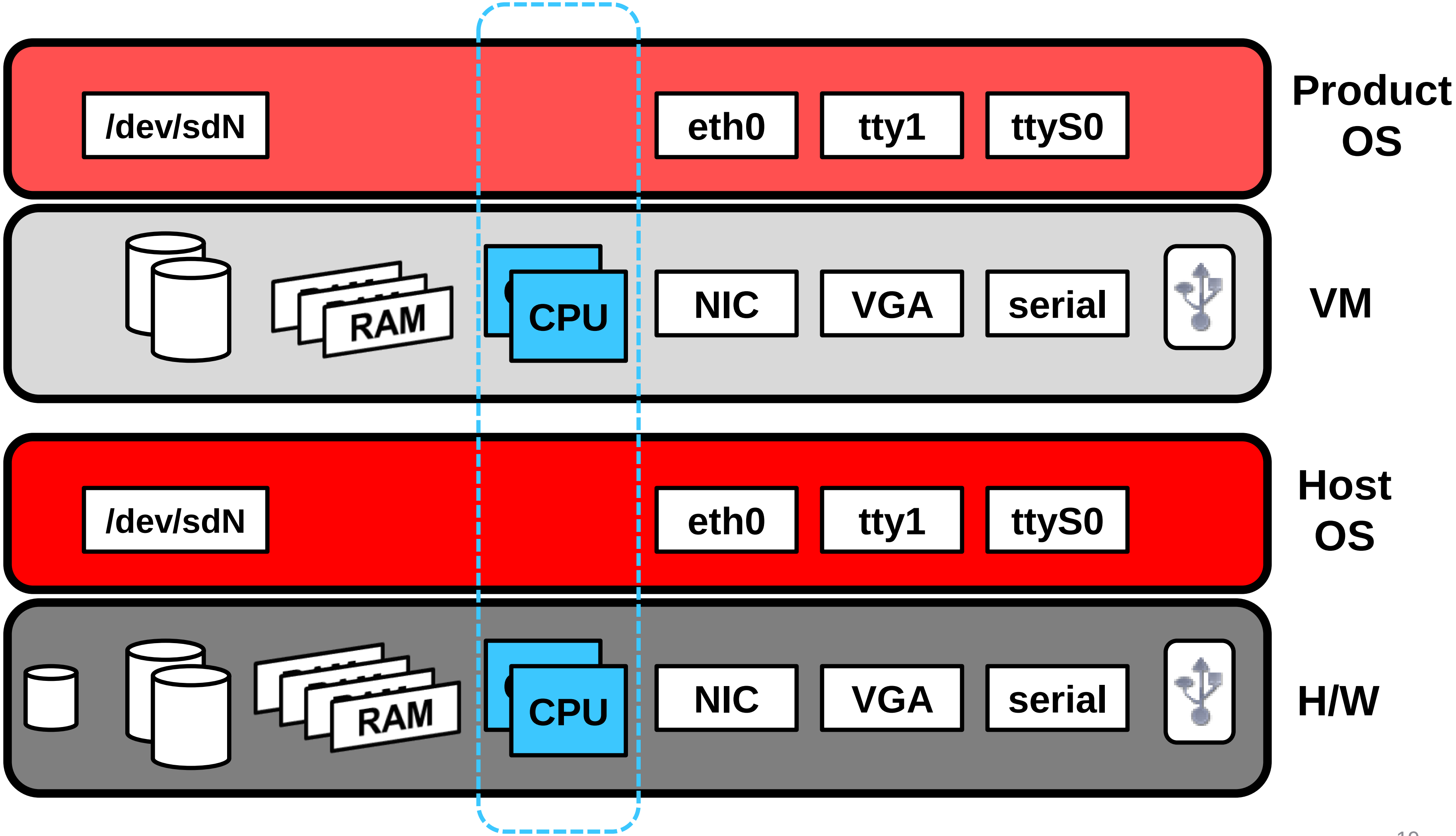
Memory Allocation

- Some RAM held back for Host OS



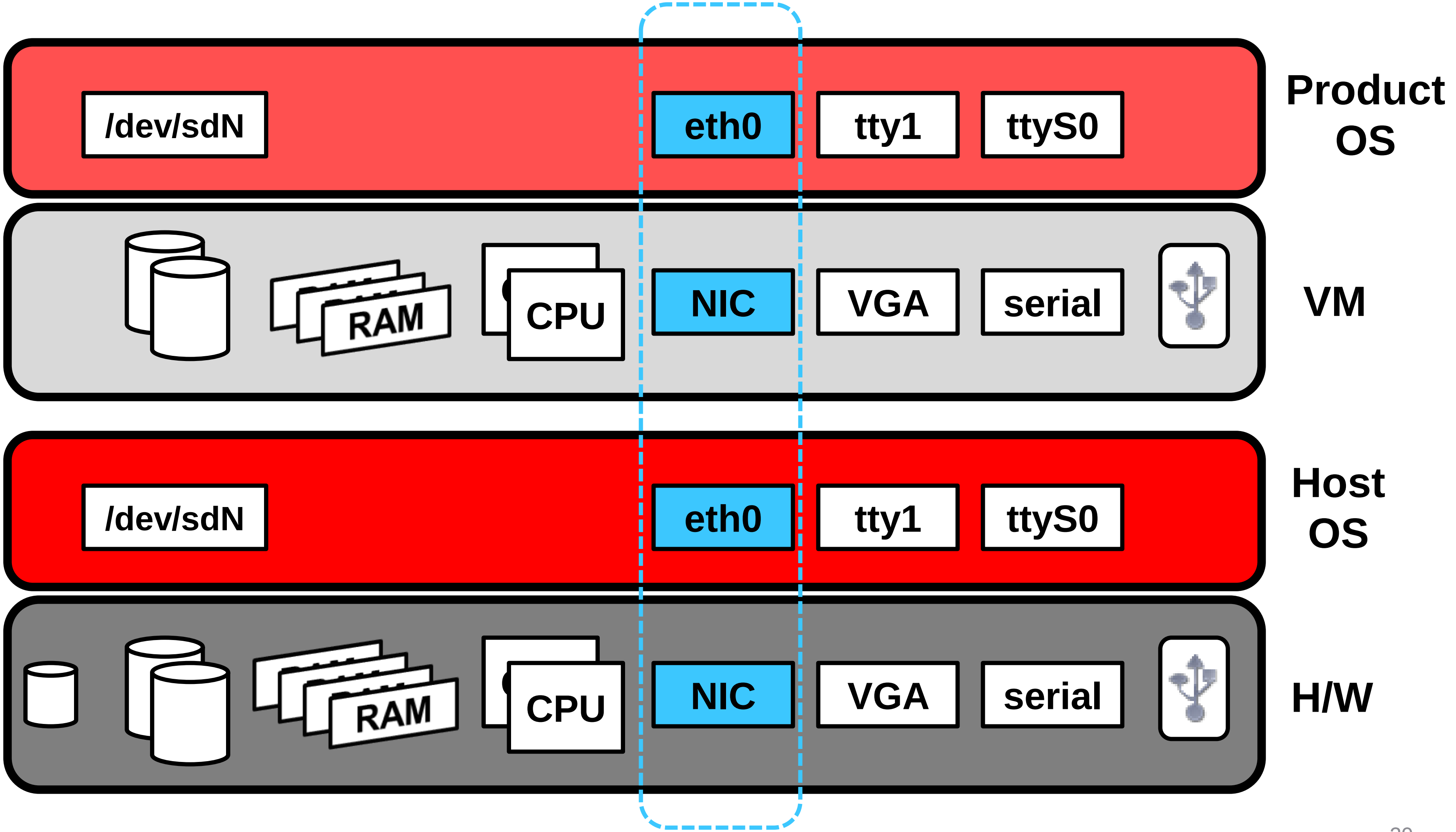
Processors

- # of Vcpus == # of Pcpus (excluding hyperthreads)



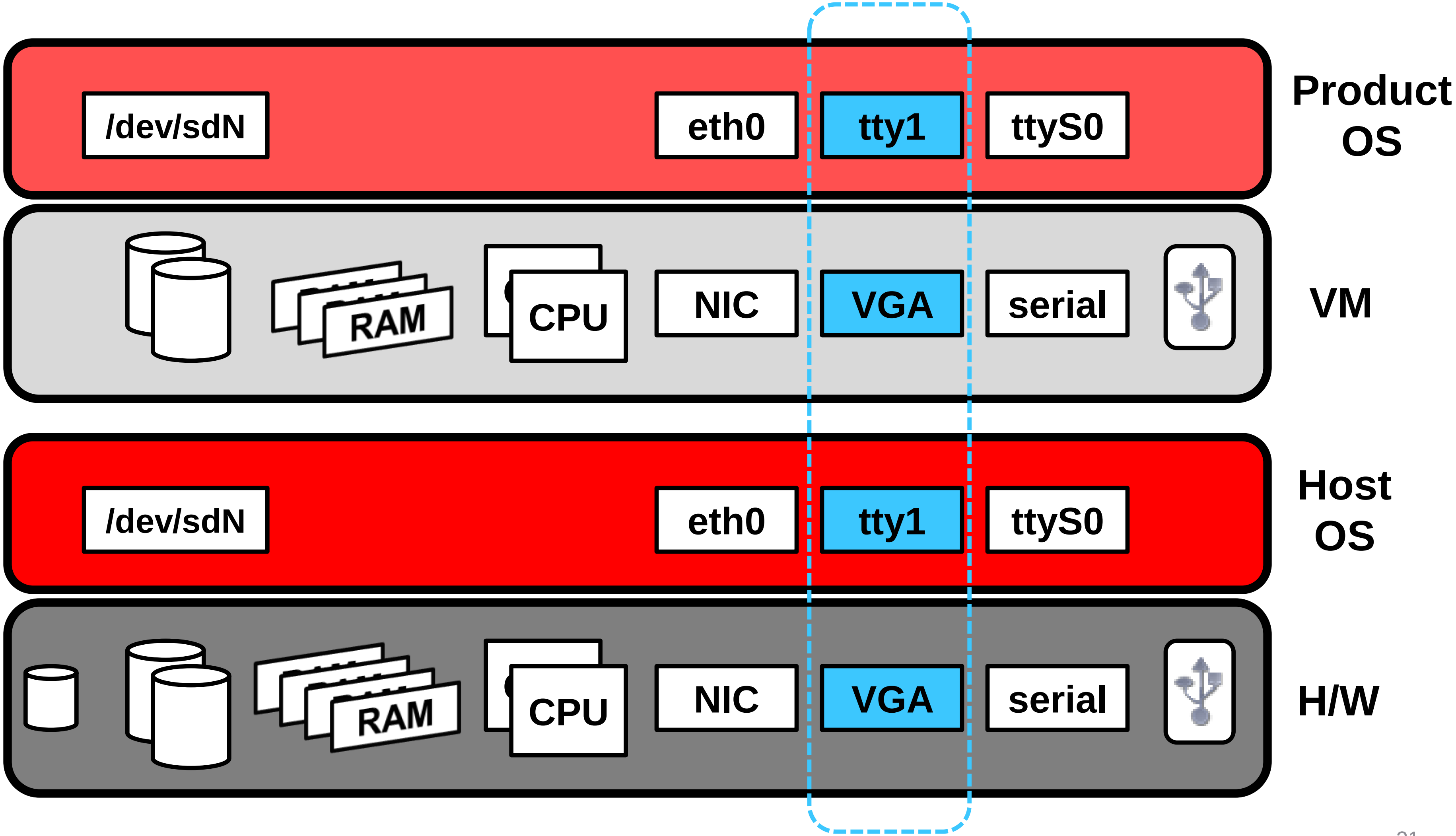
Networking

- First NIC of VM bridged to eth0 of host



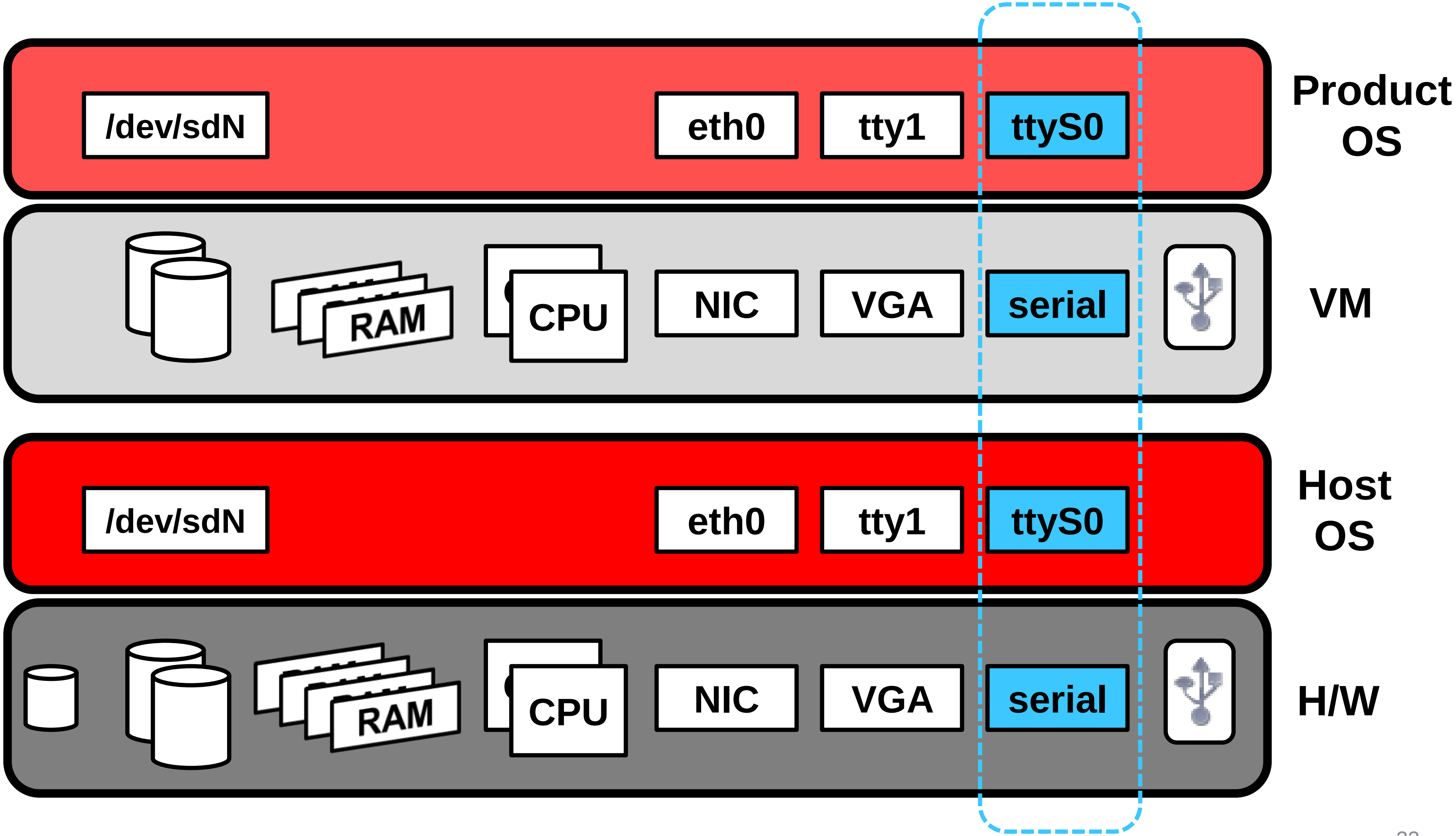
Console Display

- Console of VM connected to tty1 of host



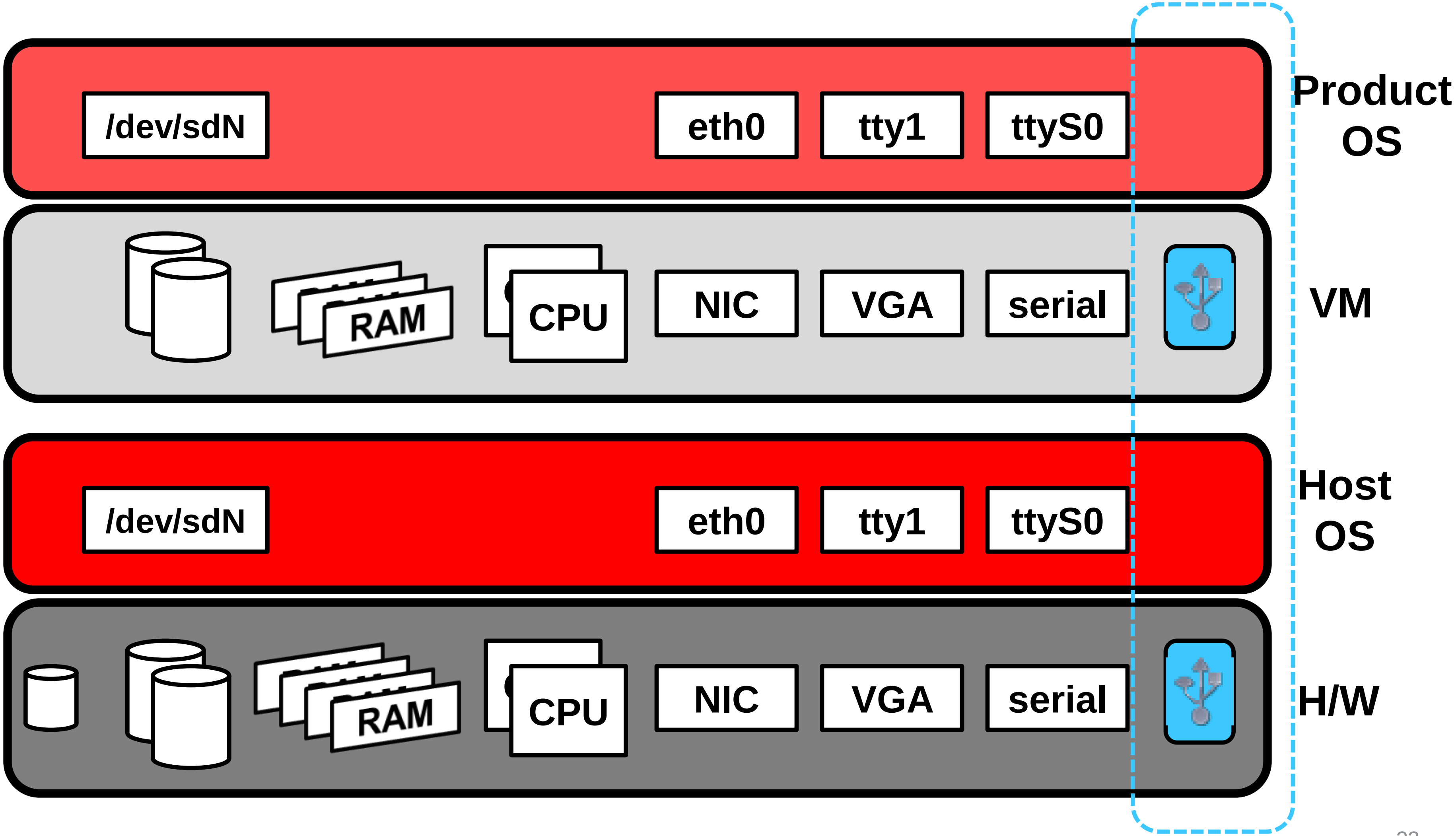
Serial Port

- Serial port of VM connected to HW



USB Devices

- USB devices passed to VM – a few exceptions



Marketing Constraints

- No change to end user experience
  - Installation, administration, monitoring, access
  - i.e., make HAL completely transparent to customer
- One DVD for product release that works for all supported hardware in the release
- HAL only installed on servers that need it
  - No user intervention required
- HAL removed upon upgrade to product release that recognizes H/W natively
  - No user intervention required
- Toggle between product releases enables/disables HAL
  - No user intervention required

Where Does KVM Fit In?

- KVM is the enabling technology for the HAL
  - behind the scenes
- qemu-kvm provides the virtual machine and device models
- KVM provides the virtualization efficiencies

Why KVM?

- KVM’s architecture ideal for “embedded” use cases
  - Allows use of standard linux distribution as the host OS
  - Use of virtualization not relevant and can be hidden from end-user
- Host OS has same ‘look-and-feel’ as product OS
  - Install and runtime (development perspective) and diagnostics (customer support perspective)
  - Essentially creating a linux-on-linux stack – load kernel module, start userspace command
- Qemu and Linux are both powerful Swiss Army knives
  - Plenty of options to meet transparency requirements and maintain appliance-like design

Installation, device handling, console, VM management, etc

HAL Design Details

Design Details

- **Installation**
- Networking
- Product CLI
- SNMP Monitoring
- Syslog and Alarm Generation
- Devices – DVD and USB
- File Sharing
- Upgrades and Disabling HAL
- …

Product Installation

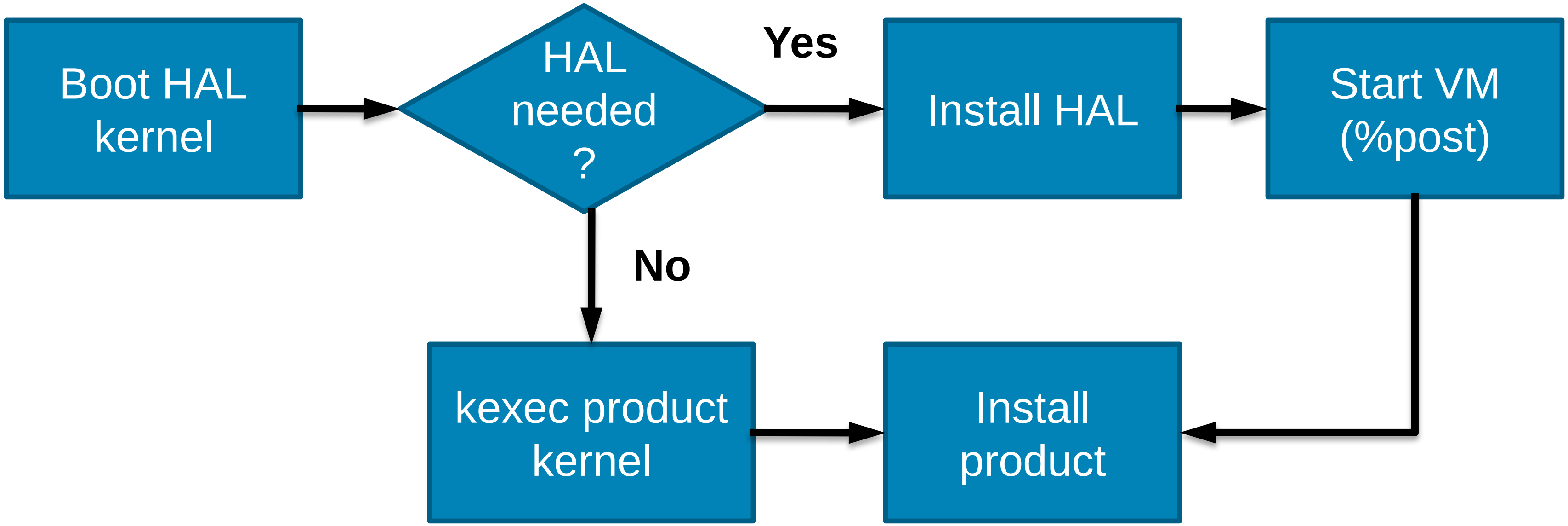
- Typical Install Sequence
  - Verify H/W is a supported platform
  - Update BIOS / firmware as needed
  - Configure BIOS and RAID
  - Install OS
  - Install product
  - Configure

Installation Impacts

- H/W aspects done during HAL install
  - Commands for H/W recognition, firmware updates, configuring RAID/BIOS, etc need to be moved to HAL
  - Updated for compatibility with Host OS
- HAL and product are two different OS installs
  - Each needs to be done within its context
  - Separate kernel + initrd.img for each OS
  - Co-exist on single DVD

Installation: One DVD For All Servers

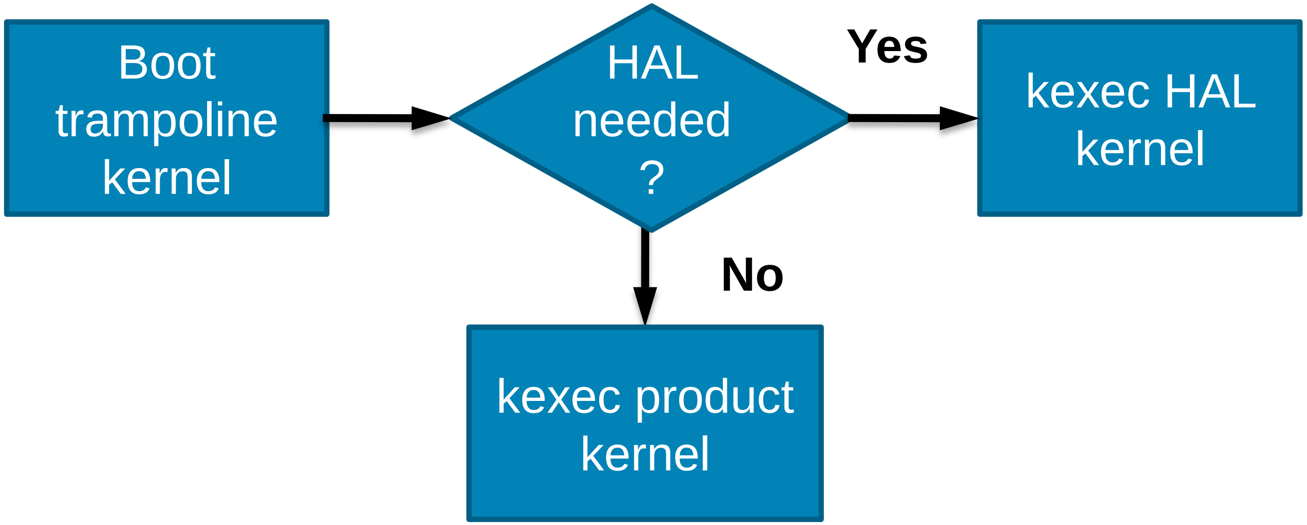
- DVD boots HAL kernel + initrd
- Inspects H/W and decides if HAL is needed



- Non-HAL servers
  - Jump to product’s OS happens quickly; install proceeds

Trampoline Kernel for DVD

- Some cases need a “trampoline” to jump between kernel versions
  - e.g., transitioning from RHEL5, 64-bit for HAL to RHEL3, 32-bit for product
- Trampoline is a very small kernel + initrd
  - No ACPI, no kernel modules
  - Just enough runtime env for basic platform detection
  - Takes less than 1 second to run and jump to next kernel



Installation with HAL

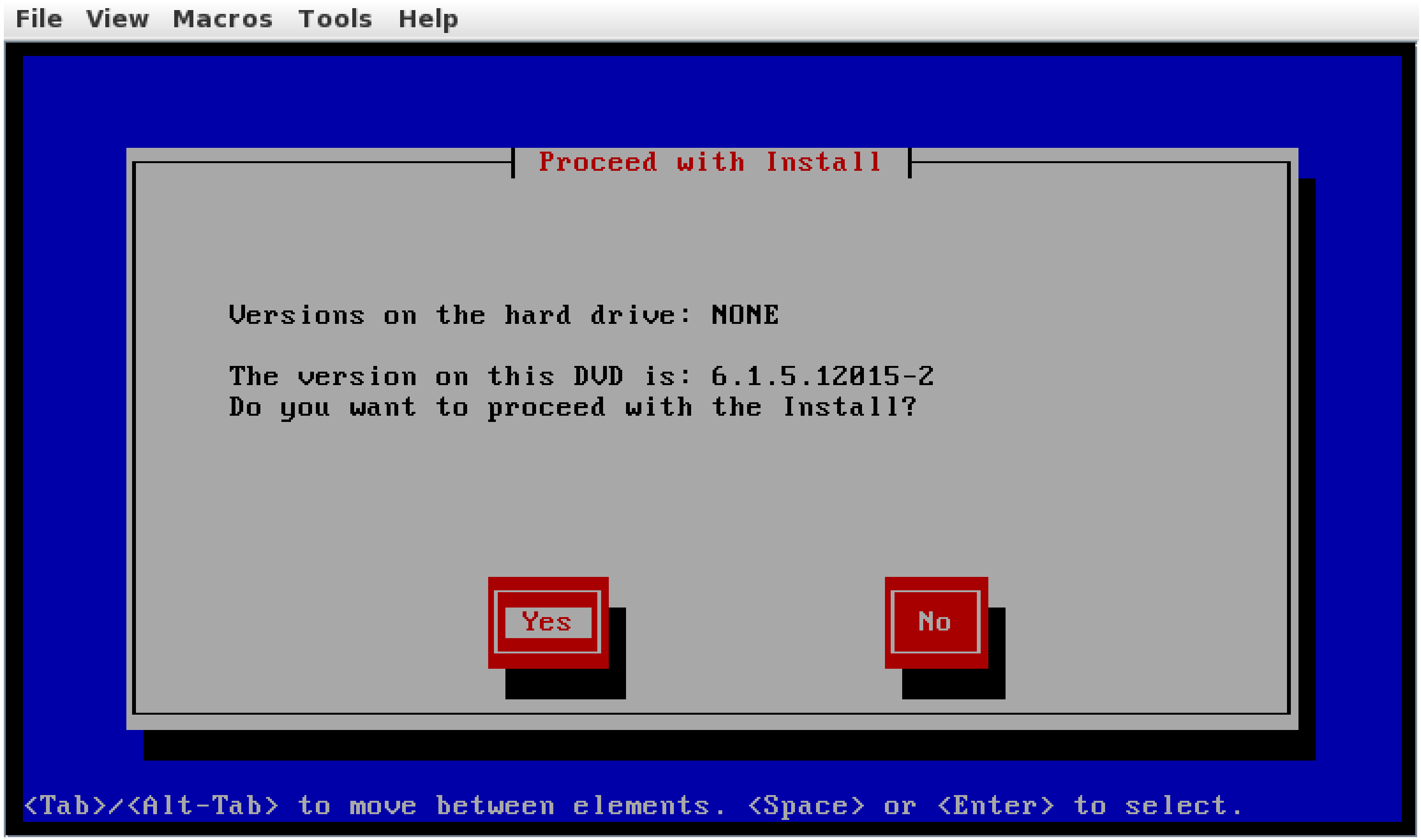
- H/W interaction done by HAL installer
  - Valid platform detection
  - BIOS, firmware updates
  - RAID/BIOS configuration
- Finds block device for Host OS
  - e.g., internal USB key
- Installs Host OS
- Starts VM for product install in post-install phase
  - Leverage –kernel and –initrd options of qemu
  - No reboot between install of HAL and product
  - Allows DVD to be used for install in VM without user intervention (ie., re-inserting)

Product Install

- Product install starts seamlessly after HAL install
  - VM console on host’s tty1
  - qemu started with –curses display option
  - stdin/stdout set to /dev/tty1
- Same installation sequence as bare metal
  - Minus the H/W updates/configuration
- Maintains current user experience with console based installs
- Put installer into ‘text’ mode (e.g., text option for Anaconda)

Initial Install Screen – HAL Installer

- Same sequence of input screens as bare metal
- First screen presented by HAL installer

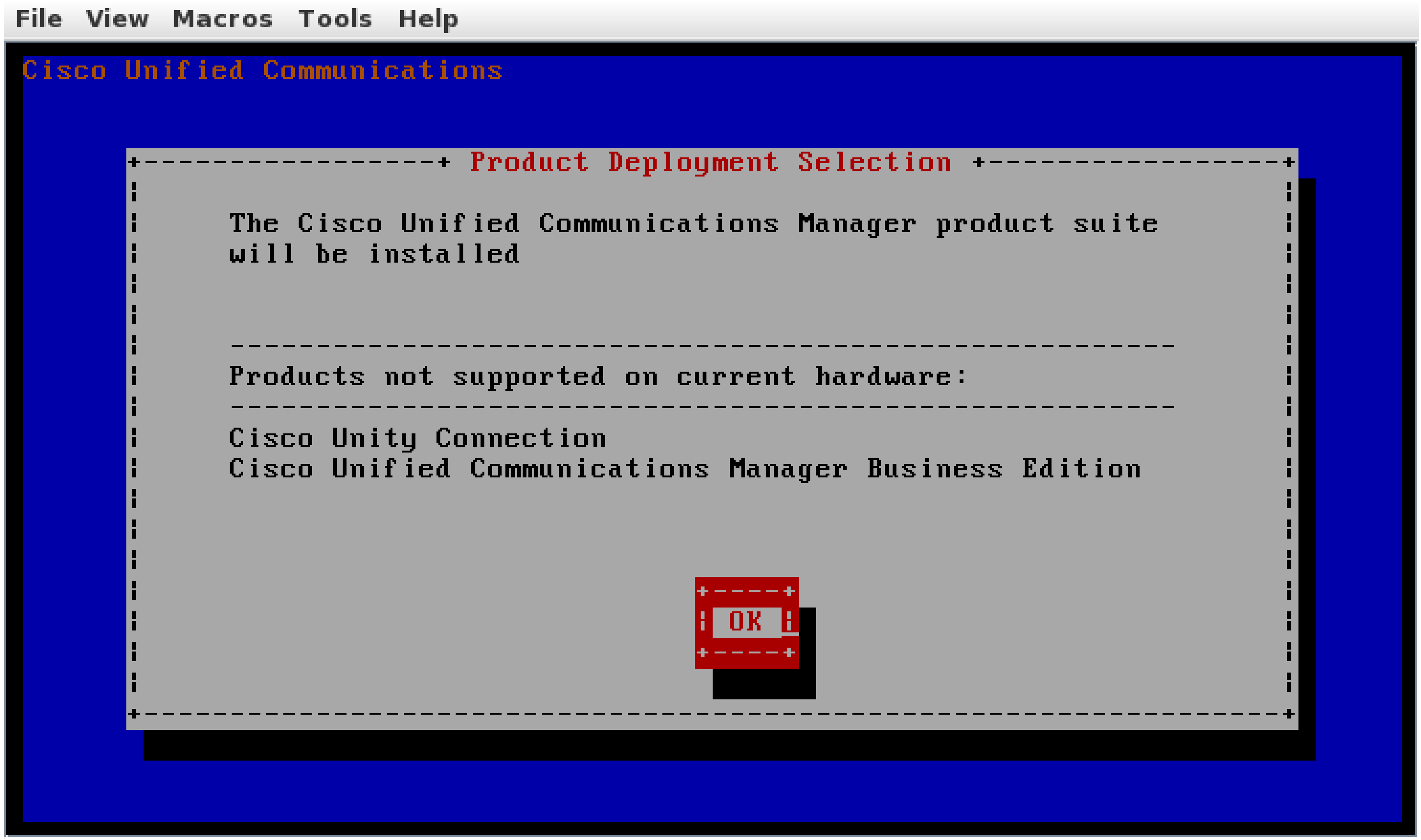


```
Proceed with Install

Versions on the hard drive: NONE

The version on this DVD is: 6.1.5.12015-2
Do you want to proceed with the Install?
```

```
<Tab>/<Alt-Tab> to move between elements. <Space> or <Enter> to select.
```

Initial Product Install Screen

- Second input screen to user is from product installer
  - Slight loss in aesthetics due to qemu + curses path



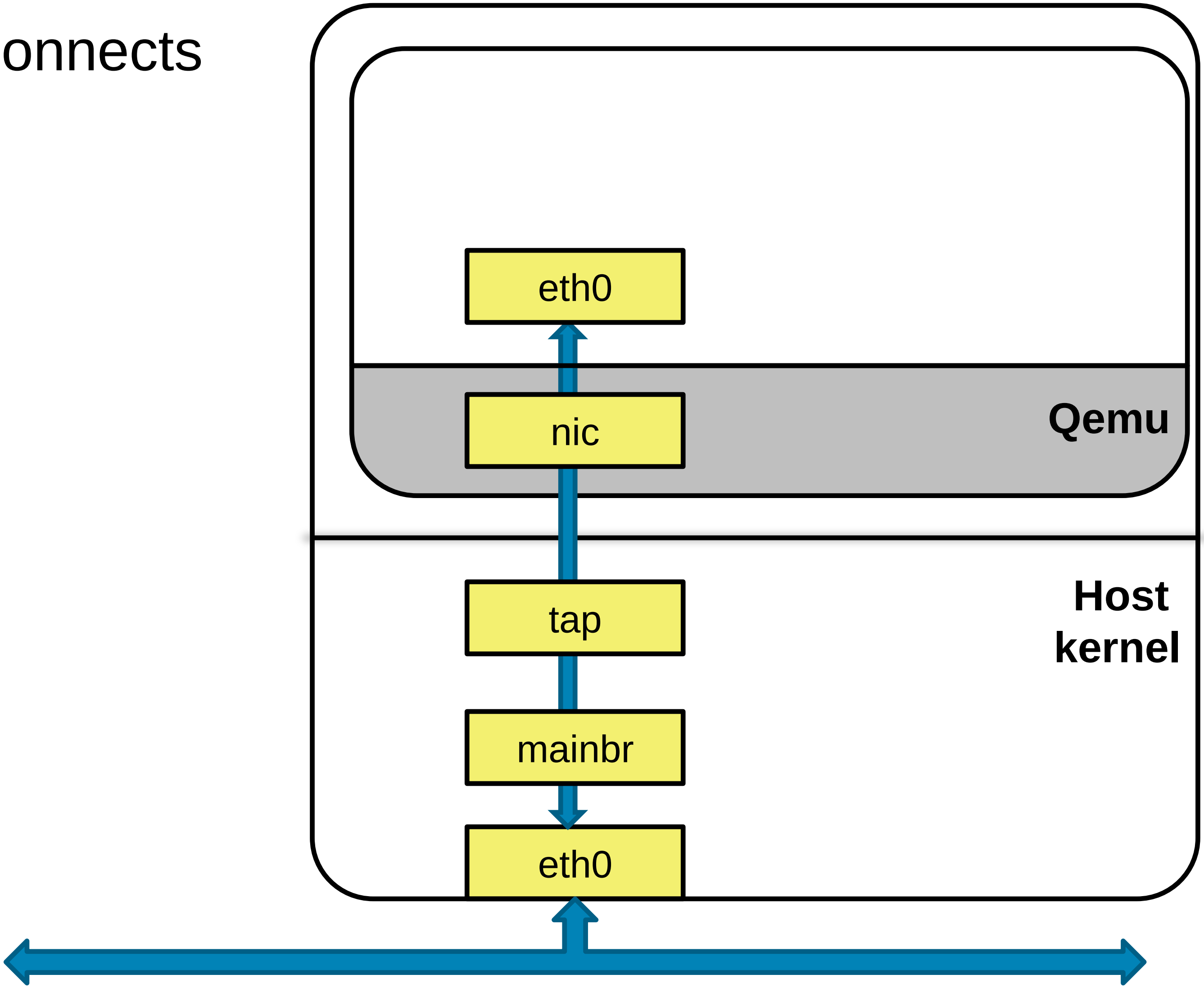
```
Cisco Unified Communications

+------------------+ Product Deployment Selection +------------------+

     The Cisco Unified Communications Manager product suite
     will be installed

     -------------------------------------------------------
     Products not supported on current hardware:
     -------------------------------------------------------
     Cisco Unity Connection
     Cisco Unified Communications Manager Business Edition

                             +-----+
                             | OK  |
                             +-----+
```

Design Details

- Installation
- **Networking**
- Product CLI
- SNMP Monitoring
- Syslog and Alarm Generation
- Devices – DVD and USB
- File Sharing
- Upgrades and Disabling HAL
- …

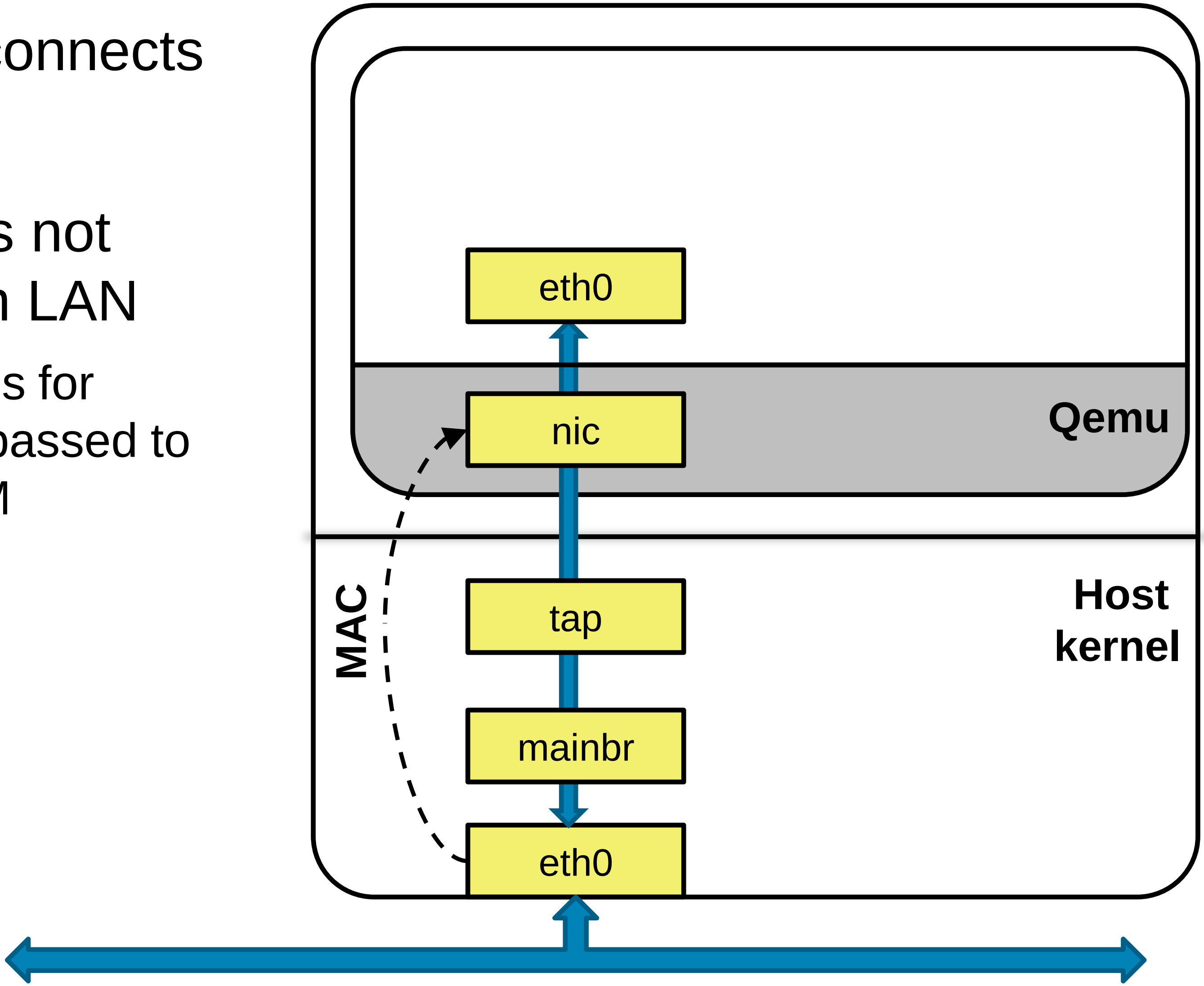
Networking

- Main bridge connects VM to LAN



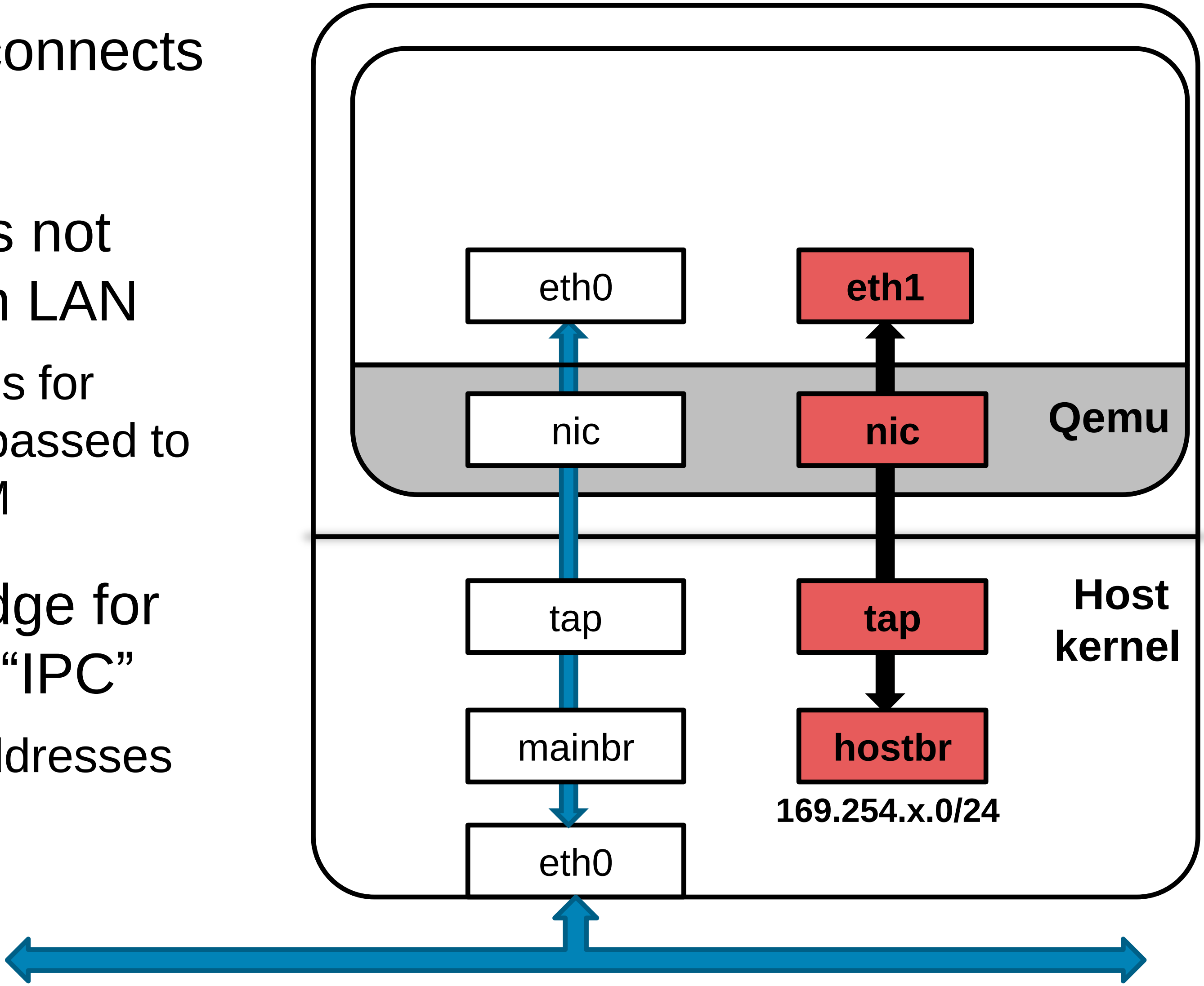
Networking

- Main bridge connects VM to LAN
- Host OS does not have an IP on LAN
  - MAC address for physical NIC passed to first NIC of VM



Networking

- Main bridge connects VM to LAN
- Host OS does not have an IP on LAN
  - MAC address for physical NIC passed to first NIC of VM
- Host-only bridge for Product-HAL "IPC"
  - Link-local addresses



Internal Network

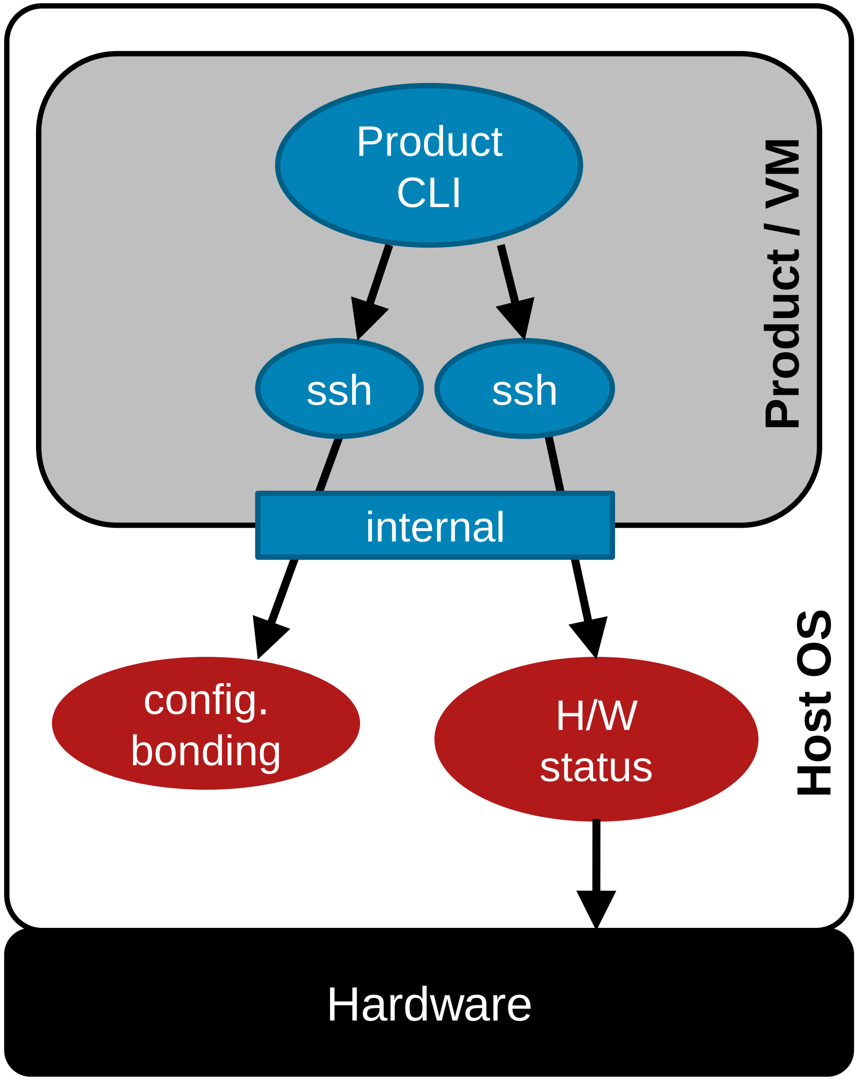
- Basis for HAL Transparency
- Allows access to H/W and HAL from within product
  - e.g., configuration & diagnostics
  - Standard protocols used for HAL-Product IPC: SNMP, syslog, ssh, NFS
- HAL-VM communications restricted to the “internal” network
  - Firewall rules in both layers restrict traffic for the link-local addresses to expected interfaces

Details

- Installation
- Networking
- **Product CLI**
- SNMP Monitoring
- Syslog and Alarm Generation
- Devices – DVD and USB
- File Sharing
- Upgrades and Disabling HAL
- …

Product CLI

- Commands that configure or query H/W need to be run in Host OS
    - e.g., bonding, NIC or RAID status
- ssh over internal network
    - public key authentication
- CLI software modified to prepend HAL wrapper to backend commands
- Backend commands updated for Host OS

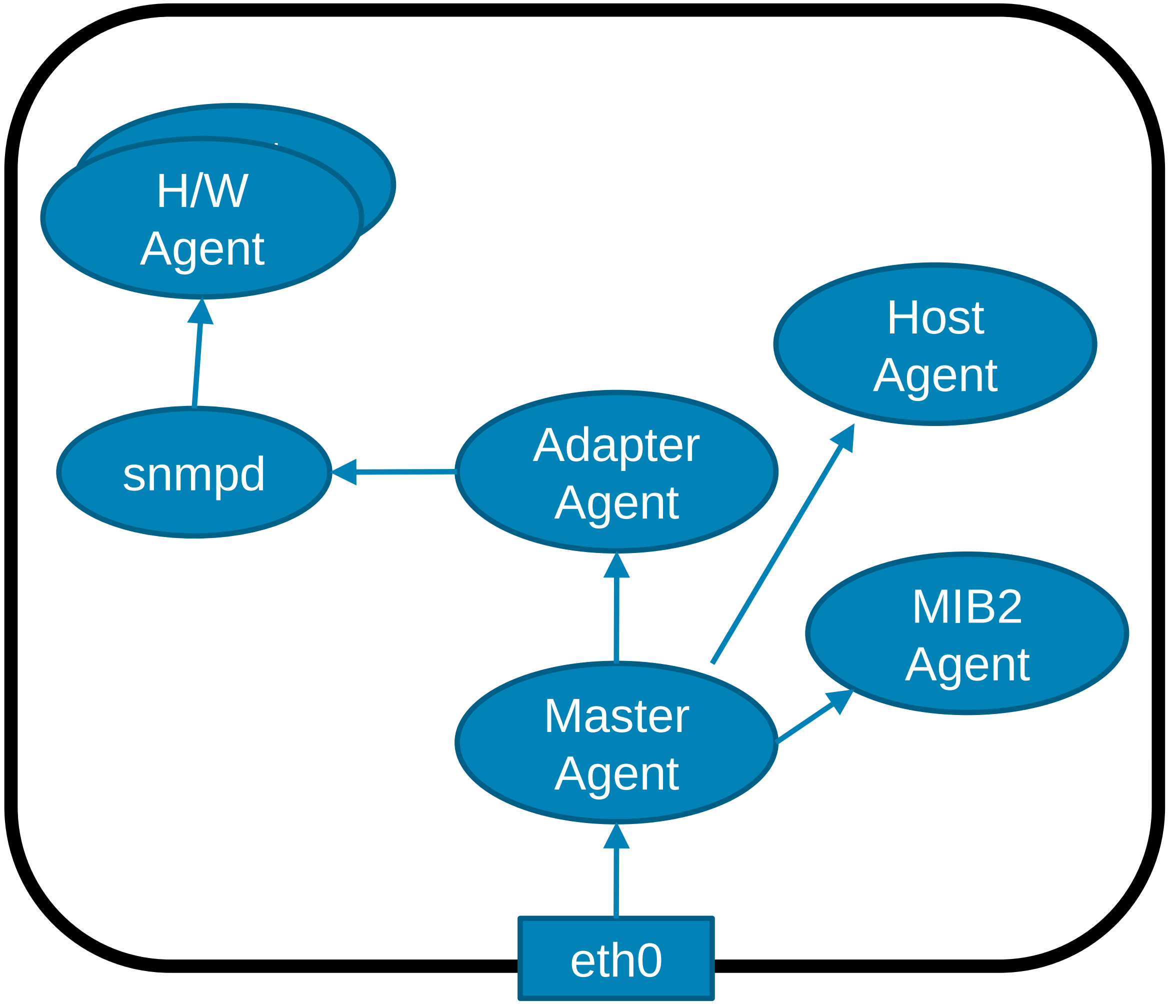


Design Details

- Installation
- Networking
- Product CLI
- **SNMP Monitoring**
- Syslog and Alarm Generation
- Devices – DVD and USB
- File Sharing
- Upgrades and Disabling HAL
- …

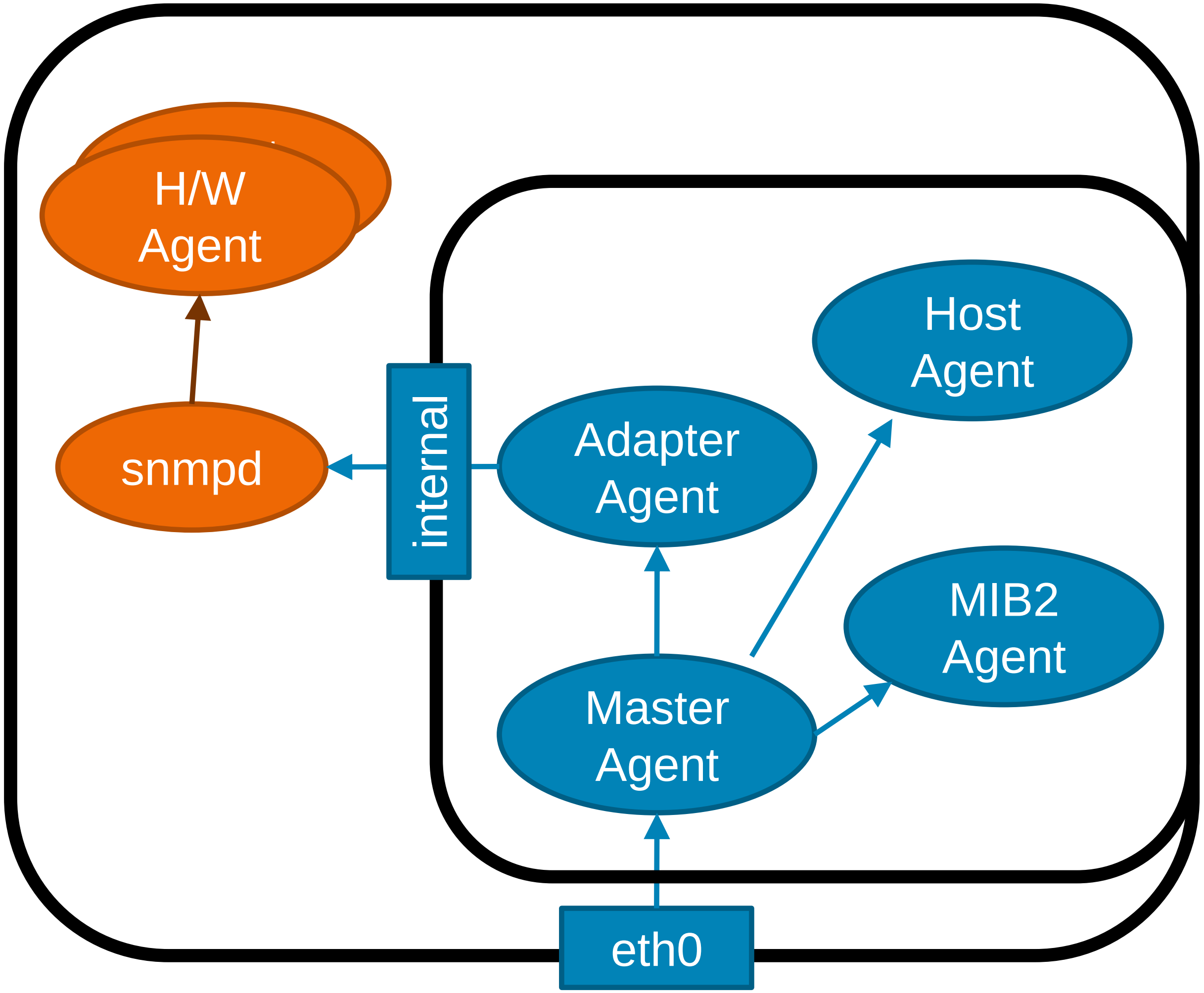
SNMP Architecture – no HAL

- Master Agent receives request
- Adapter agent used to connect to H/W agents from vendors
  - Forward done over loopback interface



SNMP Architecture – with HAL

- H/W-based agents need to run in the Host OS
- Configure adapter agent to forward over internal address

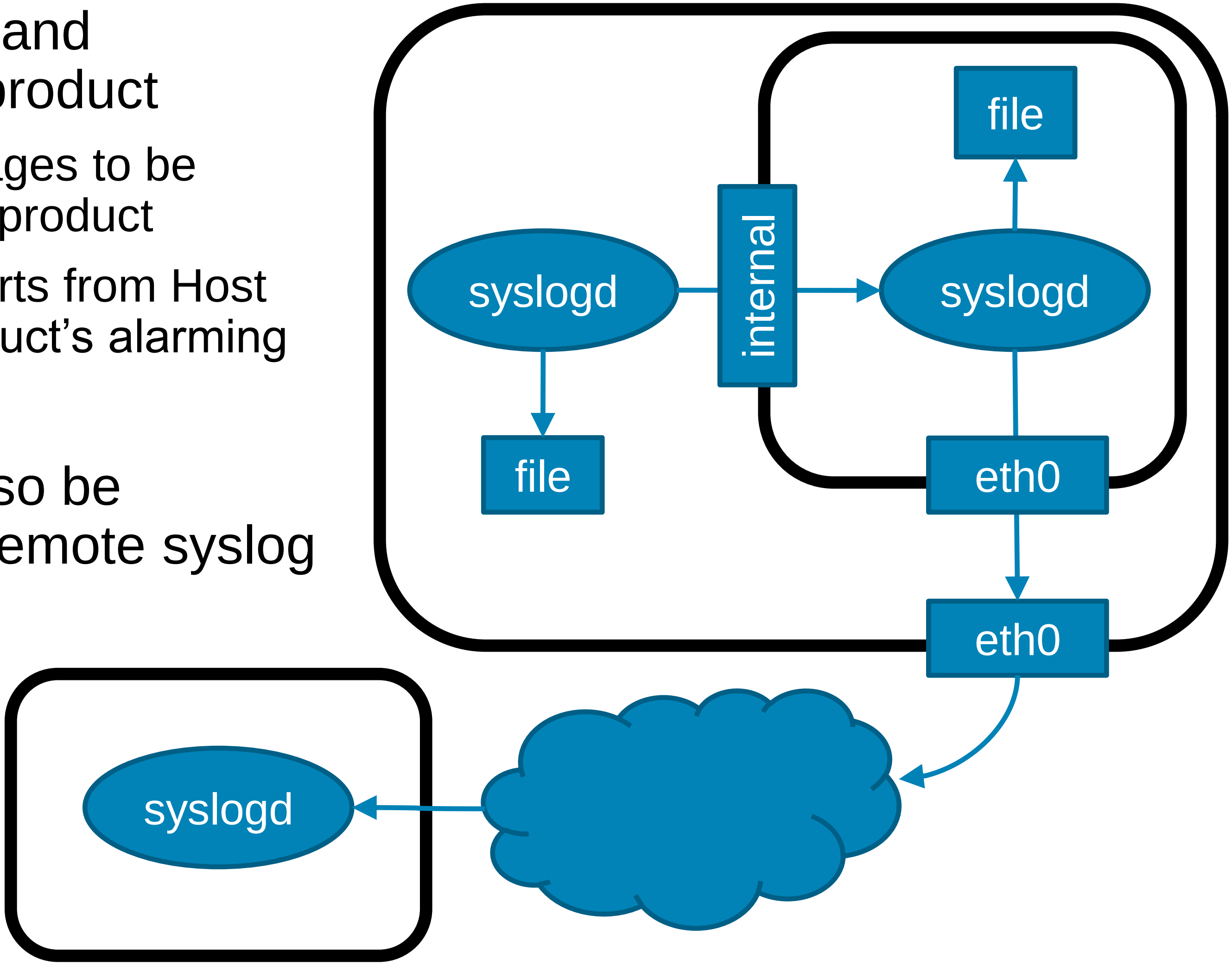


Design Details

- Installation
- Networking
- Product CLI
- SNMP Monitoring
- **Syslog and Alarm Generation**
- Devices – DVD and USB
- File Sharing
- Upgrades and Disabling HAL
- …

Syslog Forwarding

- syslog entries from HAL written locally and forwarded to product
  - Allow messages to be processed by product
  - e.g., HW alerts from Host OS enter product's alarming infrastructure
- Entries can also be forwarded to remote syslog daemon



Design Details

- Installation
- Networking
- Product CLI
- SNMP Monitoring
- Syslog and Alarm Generation
- **Devices – DVD and USB**
- File Sharing
- Upgrades and Disabling HAL
- …

Device “Passthrough”

- Emulation sense, not VT-d
- DVD device opened by VM when it boots
  - access to DVD for upgrades or other functions (e.g., password reset)
  - /dev/cdrom in Host OS maps to /dev/cdrom in VM
- H/W serial port connected to VM serial port via character device in Host OS
- USB devices added to and removed from VM when inserted/removed from hardware
  - udev script in Host OS
  - e.g., USB serial cable, audio device, storage device

Design Details

- Installation
- Networking
- Product CLI
- SNMP Monitoring
- Syslog and Alarm Generation
- Devices – DVD and USB
- **File Sharing**
- Upgrades and Disabling HAL
- …

File Sharing

- Parts of HAL filesystem exported to VM using NFS
    - restricted to internal network only
- Allows access to HAL log files from within VM
- Allows product to push files to the HAL
    - HAL, BIOS, firmware upgrades

Design Details

- Installation
- Networking
- Product CLI
- SNMP Monitoring
- Syslog and Alarm Generation
- Devices – DVD and USB
- File Sharing
- **Upgrades and Disabling HAL**
- …

Upgrades

- Dual boot partitions allows toggle between product releases
  - Use dual boot partitions for HAL too
- HAL can be upgraded as part of ongoing maintenance releases of product
  - From product perspective HAL update similar to BIOS or firmware update
- HAL can be disabled if new product version does not require it
  - Product upgrade to version with native support
  - e.g., Toggle boot order (for host OS running on USB key)

Automatic Enable / Disable of HAL

- HAL only installed on servers that need it
- Use Case: current product version requires HAL, upgrade to new version that recognizes H/W natively
  - HAL disabled on toggle to new version
  - On revert to older version, HAL re-enabled
- HAL runs from separate storage device
  - e.g., internal USB key
  - HAL enabled / disabled by modifying boot order
- Without HAL Product boots directly from harddrives
  - Recall: harddrives passed to VM, boot loader installed in typical fashion

Design Details

- Installation
- Networking
- Product CLI
- SNMP Monitoring
- Syslog and Alarm Generation
- Devices – DVD and USB
- File Sharing
- Upgrades and Disabling HAL
- …

HAL Diagnostics

- HAL performance data collected using collectd
  - System resource usage (storage, cpu, RAM)
  - Per-process statistics for processes running in Host OS
- Data sent to collectd peer running in VM
  - Provides access to HAL performance, trending data within product
- Root-level account can be mirrored from VM to HAL
- Syslog messages forwarded to syslog in VM

Restart & Shutdown

- HAL follows VM for “commanded” actions by user
  - if VM is restarted (ie., rebooted) HAL also reboots
  - If VM is shut down HAL also shuts down
- VM lockups
  - Guest manager running in Host OS exchanges heartbeats every 60 seconds with counterpart running in VM
  - VM restarted after N missed heartbeats
- HAL lockups
  - Handled by H/W functionality (e.g., ASR for HP)

HAL Security

- Minimal footprint OS
- VM runs as non-root user
- Leverage SELinux in Host OS
- Root disabled
- No default passwords
  - ssh public-key authentication for automated commands
  - sync root-level accounts when created/deleted within product
- HAL upgrades/patches applied via product
  - Like BIOS/firmware update

Performance

- Nehalem class processors
  - Overhead of virtualization layer is relatively low
- Newer hardware compensates for virtualization overhead
- Can leverage standard Linux OS features as needed
  - e.g., cpu affinity, huge pages, ksm

Net Results

- Transparency
  - Not 100%, but darn close – e.g., Host OS boot messages
  - Primary goal met - No interaction or intervention required from end user
- A Few Data Inconsistencies – e.g., SNMP
  - MIBs from H/W vendors have some overlaps with standard MIBs
  - H/W agents and standard agents run in different OS'es → data retrieved from two different views of the hardware

Advantages of HAL Approach

- More cost-effective approach to meeting preferred product timelines
  - Lifetime buys – money-based solution
  - OS backports – staff-based solution
  - HAL – technology-based solution
- Smaller development and test effort than OS backport
  - Changes primarily limited to new H/W
- Maintains closed-box, appliance model
- Forward looking
  - Leverages newer OS for HAL
  - Early experience with next OS that product will eventually move to with code re-use
- Re-usable design
  - Once hooks exist within product can be leveraged in future releases

Final Thoughts

- Not intended to allow products stay on an outdated OS indefinitely
  - Other factors drive need to change OS
  - e.g., Support for 3rd party code, security errata
- Lot of details had to be omitted for brevity
  - Please ask if you have questions

CISCO