

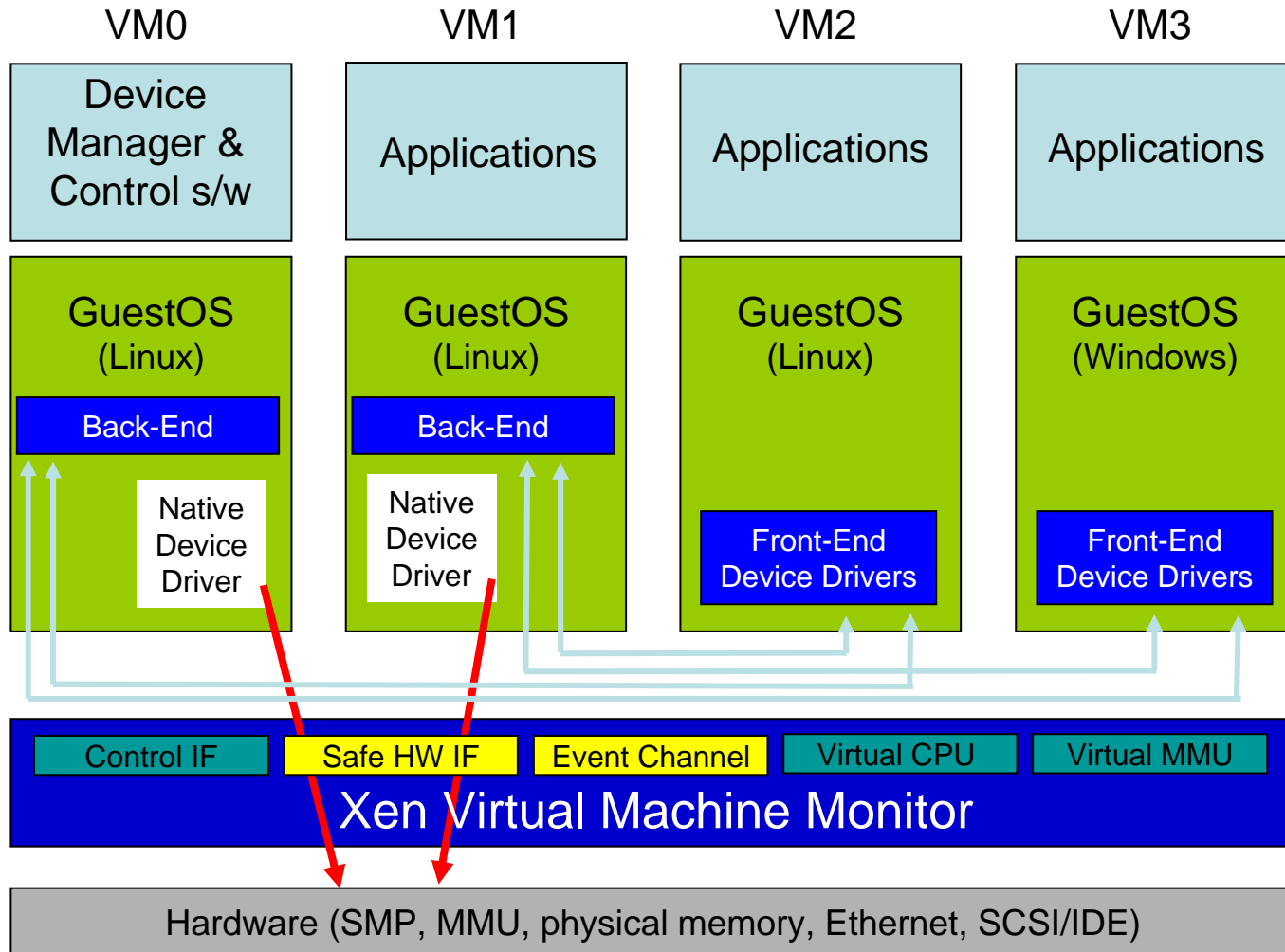


Support for Smart NICs

Ian Pratt

- Xen I/O Overview
 - Why network I/O is harder than block
- Smart NIC taxonomy
 - How Xen can exploit them
- Enhancing Network device channel
 - NetChannel2 proposal

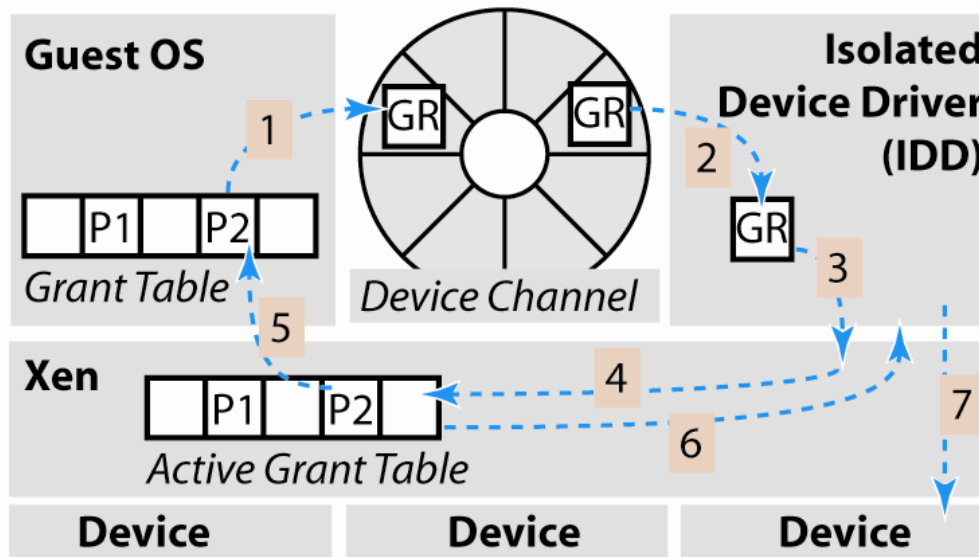
I/O Architecture



Grant Tables

Guest Requests DMA:

1. Grant Reference for Page P2 placed on device channel
2. IDD removes GR
3. Sends pin request to Xen



4. Xen looks up GR in active grant table
5. GR validated against Guest (if necessary)
6. Pinning is acknowledged to IDD
7. IDD sends DMA request to device

- Allows pages to be shared between domains
- No hypercall needed by granting domain
- Grant_map, Grant_copy and Grant_transfer operations
- Signalling via event channels

High-performance secure inter-domain communication

- Block I/O is much easier to virtualize than Network I/O:
 - Lower # operations per second
 - The individual data fragments are bigger (page)
 - Block I/O tends to come in bigger batches
 - The data typically doesn't need to be touched
 - Only need to map for DMA
 - DMA can deliver data to final destination
 - (no need read packet header to determine destination)

- Single free buffer, RX and TX queues
- TX and RX checksum offload
- Transmit Segmentation Offload (TSO)
- Large Receive Offload (LRO)
- Adaptive interrupt throttling
- MSI support

- (iSCSI initiator offload – export blocks to guests)
- (RDMA offload – will help live relocation)

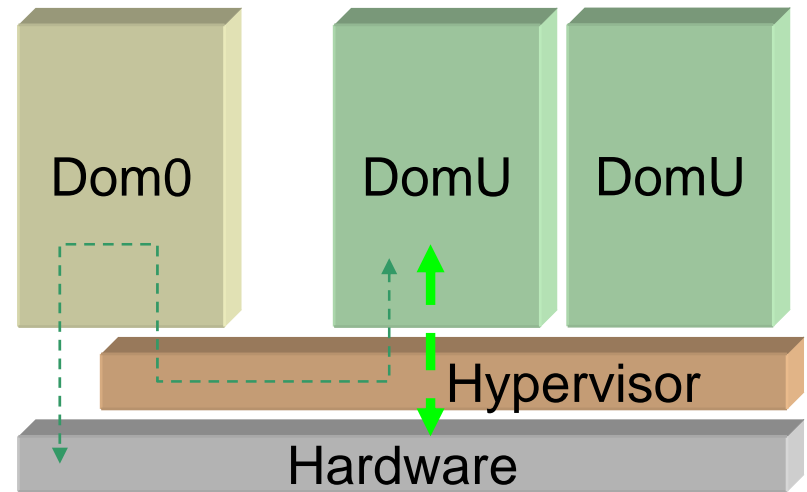
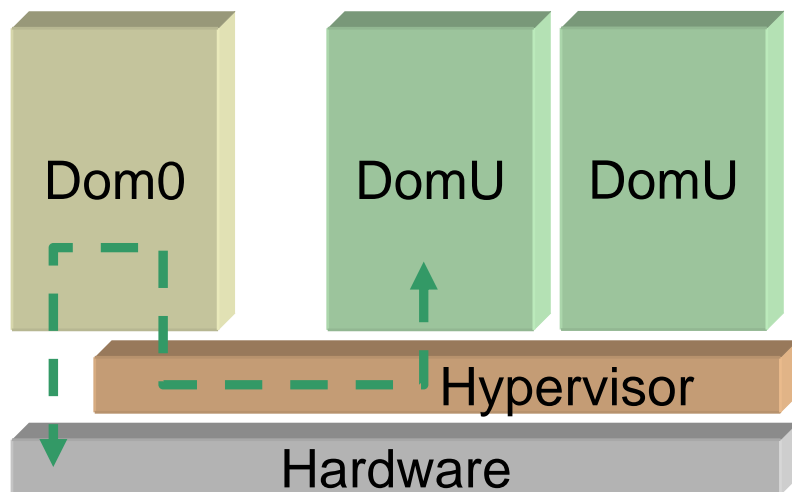
- NIC supports multiple free and RX buffer Q's
 - Choose Q based on dest MAC, VLAN
 - Default queue used for mcast/broadcast
- Great opportunity for avoiding data copy for high-throughput VMs
 - Try to allocate free buffers from buffers the guest is offering
 - Still need to worry about bcast, inter-domain etc
- Multiple TX queues with traffic shapping

- NIC allows Q pairs to be mapped into guest in a safe and protected manner
 - Unprivileged h/w driver in guest
 - Direct h/w access for most TX/RX operations
 - Still need to use netfront for bcast, inter-dom
- Memory pre-registration with NIC via privileged part of driver (e.g. in dom0)
 - Or rely on architectural IOMMU in future
- For TX, require traffic shaping and basic MAC/srcIP enforcement

Level 2 NICs e.g. Solarflare / Infiniband



- Accelerated routes set up by Dom0
 - Then DomU can access hardware directly
- NIC has many Virtual Interfaces (VIs)
 - VI = Filter + DMA queue + event queue
- Allow untrusted entities to access the NIC without compromising system integrity
 - Grant tables used to pin pages for DMA



- NIC presents itself as multiple PCI devices, one per guest
 - Still need to deal with the case when there are more VMs than virtual h/w NIC
 - Same issue with h/w-specific driver in guest
- Full L2+ switch functionality on NIC
 - Inter-domain traffic can go via NIC
 - But goes over PCIe bus twice

- Time to implement a new more extensible protocol (backend can support old & new)
 - Variable sized descriptors
 - No need for chaining
 - Explicit fragment offset and length
 - Enable different sized buffers to be queued
 - Reinstate free-buffer identifiers to allow out-of-order RX return
 - Allow buffer size selection, support multiple RX Q's

- Allow longer-lived grant mappings
 - Sticky bit when making grants, explicit un-grant operation
 - Backend free to cache mappings of sticky grants
 - Backend advertises it's current per-channel cache size
 - Use for RX free buffers
 - Works great for Windows
 - Linux “alloc_skb_from_cache” patch to promote recycling
 - Use for TX header fragments
 - Frontend copies header (e.g. 64 bytes) into a pool of sticky mapped buffers
 - Typically no need for backend to map the payload fragments into virtual memory, only for DMA

- Try to defer copy to the receiving guest
 - Better for accounting and cache behaviour
 - But, need to be careful to avoid a slow receiving domain from stalling TX domain
 - Use timeout driven `grant_copy` from dom0 if buffers are stalled
- Need transitive grants to allow deferred copy for inter-domain communication

- Maintaining good isolation while attaining high-performance network I/O is hard
- NetChannel2 improve performance with traditional NICs and is designed to allow Smart NIC features to be fully utilized

Last talk



- Privileged/unprivileged NIC driver model
- Free/rx/tx descriptor queues into guest
- Packet demux and tx enforcement
- Validation of frag descriptors
- TX QoS
- CSUM offload / TSO / LRO / intr coalesce

- Packet demux to queues
 - MAC address (possibly multiple)
 - VLAN ttag
 - L3/L4 useful in some environments
- Filtering
 - Source MAC address and VLAN enforcement
 - More advanced filtering
- TX rate limiting: x KB every y ms

- Inter-VM communication
 - Bounce via bridge on NIC
 - Bounce via switch
 - Short circuit via netfront
- Broadcast/multicast
- Running out of contexts
 - Fallback to netfront
- Multiple PCI devs vs. single
- Card IOMMU vs. architectural

- Pre-registering RX buffers is easy as they are recycled
- TX buffers can come from anywhere
 - Register all guest memory
 - Copy in guest to pre-registered buffer
 - Batch, register and cache mappings
- Pinning can be done in Xen for architectural IOMMUs, dom0 driver for NIC IOMMUs

- Privileged state relocated via xend
 - Tx rate settings, firewall rules, credentials etc.
- Guest can carries state and can push down unpriv state on the new device
 - Promiscuous mode etc
- Heterogeneous devices
 - Need to change driver
 - Device independent way of representing state
 - (more of a challenge for RDMA / TOE)

- Proxy device driver
 - Simplest
 - Requires guest OS to have a driver
- Driver in stub domain, communicated to via netchannel like interface
 - Overhead of accessing driver
- Driver supplied by hypervisor in guest address space
 - Highest performance
- “Architectural” definition of netchannel rings
 - Way of kicking devices via Xen