



IBM

Improving HVM Domain Isolation and Performance

Jun Nakajima - jun.nakajima@intel.com

Daniel Stekloff – dsteklof@us.ibm.com

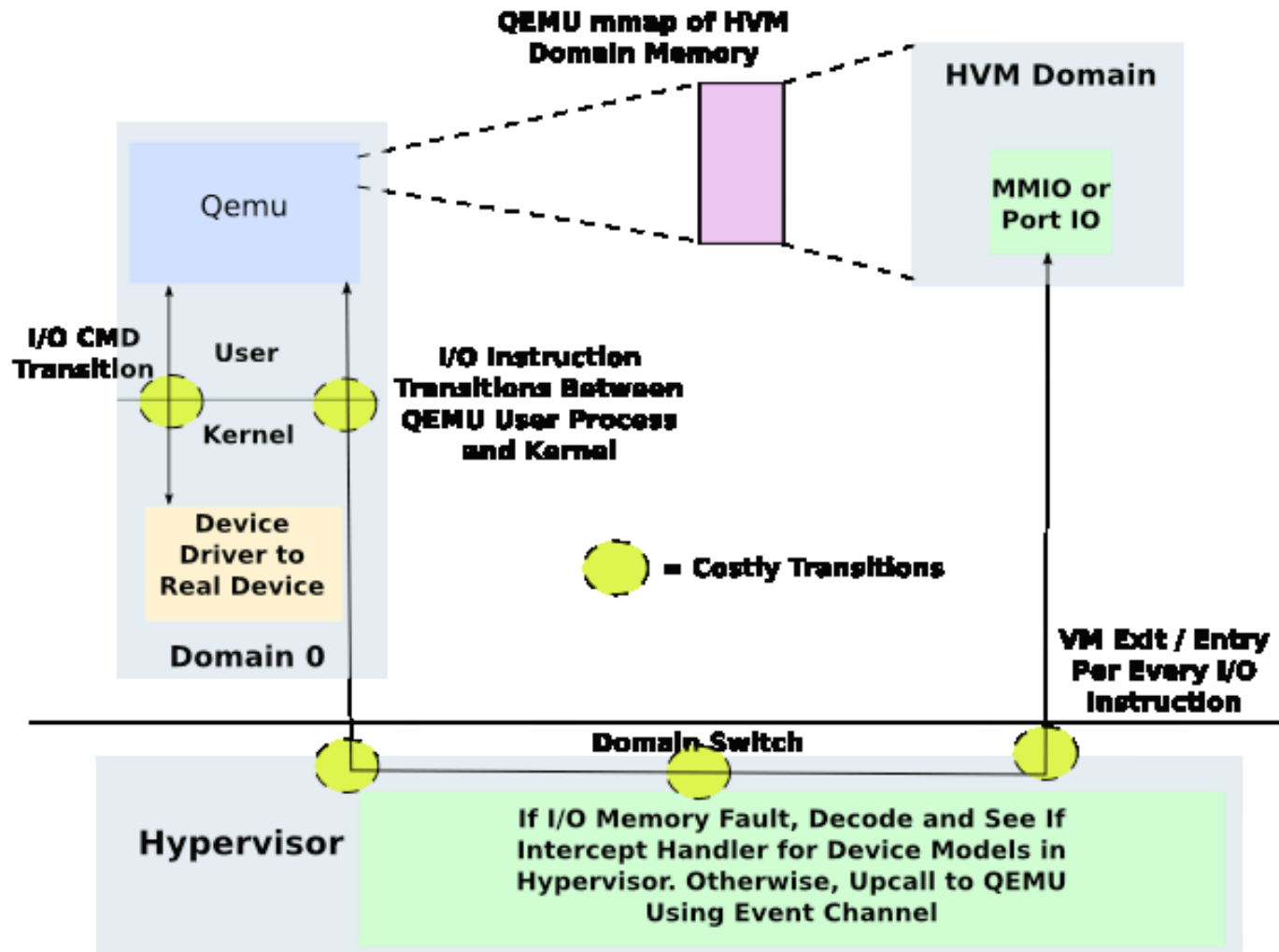
September 2006



Goals

- HVM Domain Isolation
 - Move QEMU Device Model Out of Domain0 and Into Stub Domain
 - Allow Proper Accounting
- Improve HVM Domain Performance
 - Reduce Expensive Transitions
- Discussion - Community Agreement

Current Device Model

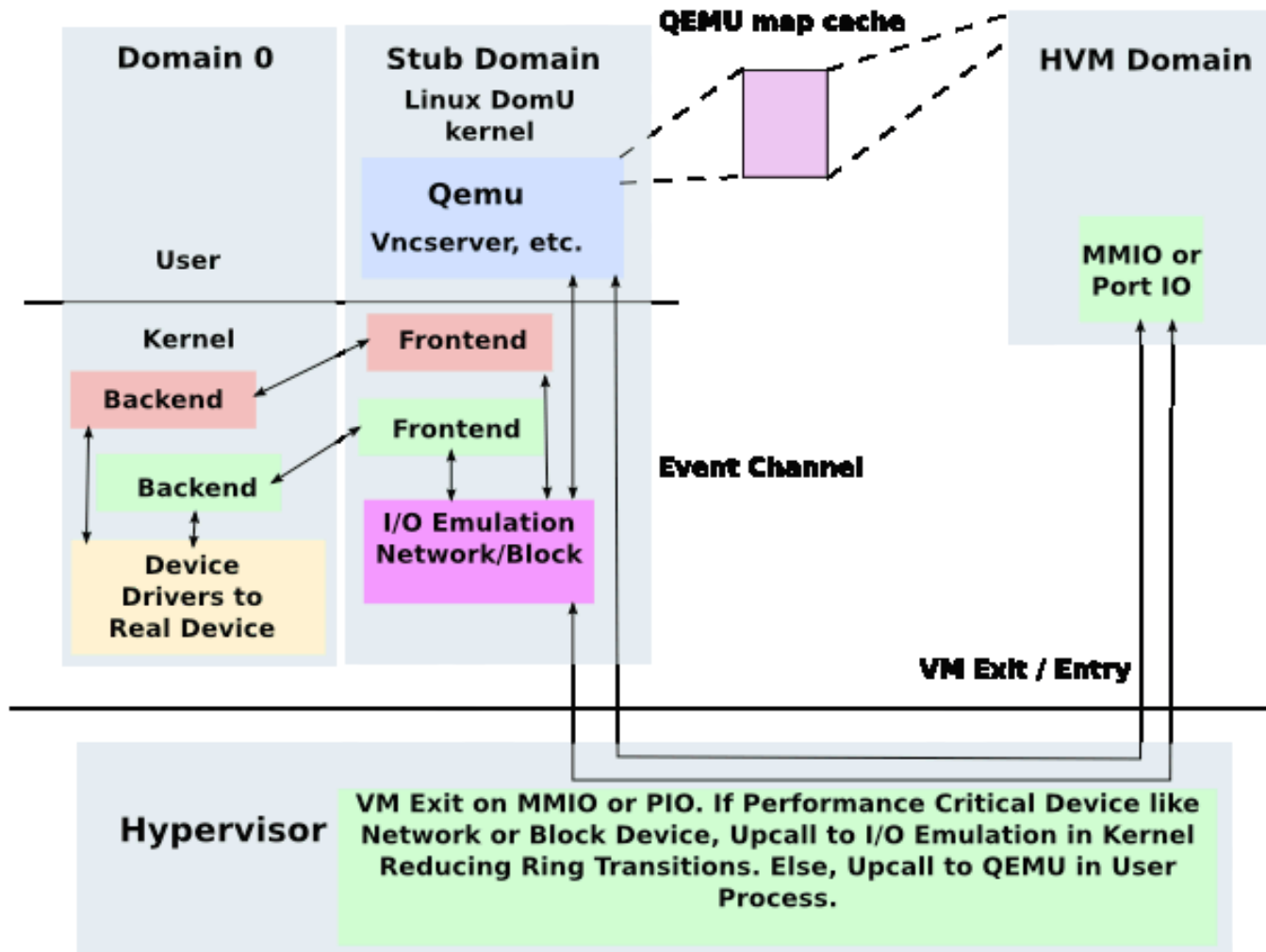


Current Device Model

- QEMU Device Emulation
 - Runs in Domain0 as User Process
 - No Isolation to Properly Accounting for HVM I/O
 - Doesn't Scale – Launch New QEMU-DM For Every HVM Domain, Currently Maps All HVM Memory
 - Many Costly Ring Transitions
 - On Every I/O Instruction Event Between QEMU and Kernel
 - On Every I/O Command Between QEMU and Real Device Drivers
 - Many Costly VM Exits for Every I/O Instruction
 - One I/O Command Consists of Many I/O Instructions



Stub Domain



Stub Domain Requirements

- Requirements
 - Need User Space Context for QEMU
 - Some Devices Require User Process - vncserver
 - Need Library Support for QEMU
 - Need SMP Support
 - Need to Run IO Emulation in Kernel Space
 - Need to Run Frontend Drivers
 - Need to Limit Impact to Build
 - Need to Limit QEMU Maintenance Impact

Stub Domain Implementation

- Use Current PV Domain
- Use Stripped Linux DomU Kernel
 - Use Same Build System with Special Config
- New Kernel Drivers to Patch In
 - Performance Critical IO Emulations – Network/Block
 - Frontend Drivers
- Create Tools Support for Pairing Domains
- Create Necessary Hypervisor Support
 - Upcall Routing
 - Scheduler “short cut” to Bind Domains



Stub Domain IO Process

- HVM Domain VM Exits with MMIO or Port IO
- Hypervisor Routes IO Request to Stub Domain
- If Performance Critical Emulation
 - Jump to Emulation in Stub Domain Kernel
 - Scheduler “short cut” to Bind Domains
 - Emulation Batches IO Instructions
 - Emulation Sends IO Command to Frontend Drivers
 - Frontend Drivers Work with Backend Drivers like PV Domain
- If Non-Performance Critical or No Split-Level Driver Support
 - Upcall to QEMU in Stub Domain User Space
 - Example - vncserver



Stub Domain Benefits

- Provides Isolation for HVM Domains
- QEMU Move Enables Scaling for Many HVM Domains
- Performance Improvements
 - Fewer Transitions
- Uses Existing Code and Framework
 - Use Linux DomU Kernel, Separate Config
 - No QEMU Maintenance Issues
 - Use Existing QEMU-DM
 - Fewer Maintenance Issues
 - Need QEMU-DM Functionality for Save/Restore

Stub Domain Issues

- QEMU User Process in Stub Domain Requires Network Connection – Example: Vncserver
- Directly Scheduling Critical I/O Performance in Stub Domain Requires More Thought
- I/O Emulation in Kernel Needs Access to QEMU User Process Map Cache
- No Hotplug to Start
- Management Tools Impact
 - Another Domain to Handle
 - Handling Paired Domains

QEMU Move Issues

- Privileged Access to HVM Domain:
 - QEMU Access to HVM Domain Memory
 - Xenstore Directory Access
- Can All of QEMU Emulation Be Moved Off of Domain0?
 - Devices with No Split-Level Driver Support?
 - How to Handle VGA?
- Routing I/O Upcalls in Hypervisor to Appropriate Domain
- Connecting QEMU Device Emulation to Frontend Drivers

Device Emulation

- Better Device Emulation Support
 - Improve Performance with Devices that Require Less Interaction
 - Less PIO and Fewer VM Exits, Better Performance
 - Batching in Kernel Device Emulations



Future Optimizations

- Integrating Stub Domain into HVM Domain
 - Benefits:
 - One Domain, Two Kernels
 - Reduces Domain Switching
 - Issues:
 - Tracking Two Register Sets for HVM Domain
 - Invasive Changes to Hypervisor
 - Domain with PV and HVM Dual Personalities
- V2E IO Implementation, Reduce Exits by Running in Emulation