Experimenting with I/O data paths in Virtualized Environments

Anastassios Nanos
ananos@cslab.ece.ntua.gr

Computing Systems Laboratory
School of Electrical and Computer Engineering
National Technical University of Athens

Mar 4, 2012

Motivation

Virtualization is about:

(a) Multiplexing CPU/Memory subsystems

(b) Scheduling of VM workloads

(c) Multiplexing – Managing I/O access

Motivation – HPC Application demands

Virtualization is about:

(a) Multiplexing CPU/Memory subsystems

(b) Scheduling of VM workloads

(c) Multiplexing – Managing I/O access

High Performance Computing Applications

- communication libraries (MPI)
- mechanisms to bypass general purpose kernel algorithms
  - process scheduling (CPU affinity, process priority, etc.)
  - device access (user-level networking, direct I/O techniques such as zero-copy, page-cache bypass etc.)

Motivation – I/O techniques in virtualized environments

Split Driver Model

- backend (privileged guest)
- frontend (guest domains)
- communication mechanism (interrupt routing, page flipping, shared memory techniques)

In Virtualized environments

- the hypervisor multiplexes the execution of guest OS kernels
- these kernels are not (always) directly aware of the hardware

Motivation – I/O techniques in virtualized environments

Split Driver Model

- backend (privileged guest)
- frontend (guest domains)
- communication mechanism (interrupt routing, page flipping, shared memory techniques)

In Virtualized environments

- the hypervisor multiplexes the execution of guest OS kernels
- these kernels are not (always) directly aware of the hardware

Challenge:

bridge the gap between HPC application demands and I/O techniques in Virtualized environments

Motivation – HPC App. I/O demands in VM environments

Explore alternative data paths, direct or indirect

Integrate HPC adaptive layers in VMMs:

- Direct data path from VM kernels to NICs (SRIOV)
  - intelligent adapters export part of their capabilities to VMs
  - device access by multiple VMs is multiplexed in firmware
- Direct application–to–NIC data path
  - decouple data transfers from virtualization layers
  - exploit already existing virtualization semantics in I/O devices (e.g. endpoint/process–based, isolation, multiplexing in firmware level)

Objective

framework to support HPC I/O device sharing in VM environments

Components – Hardware / Software

Multiple Virtual Interfaces (or VFs)

- specific hardware support (hardware/firmware multiplexing – IOV) or
- software based using dedicated domains (software multiplexing PV/hybrid)

Message passing has to be stacked above these virtual interfaces provided to VMs

- using protocols like iWARP (over TCP/IP) or
- using lightweight protocols (CPU utilization?)

Using these Components we build a thin layer to export a Message Passing API

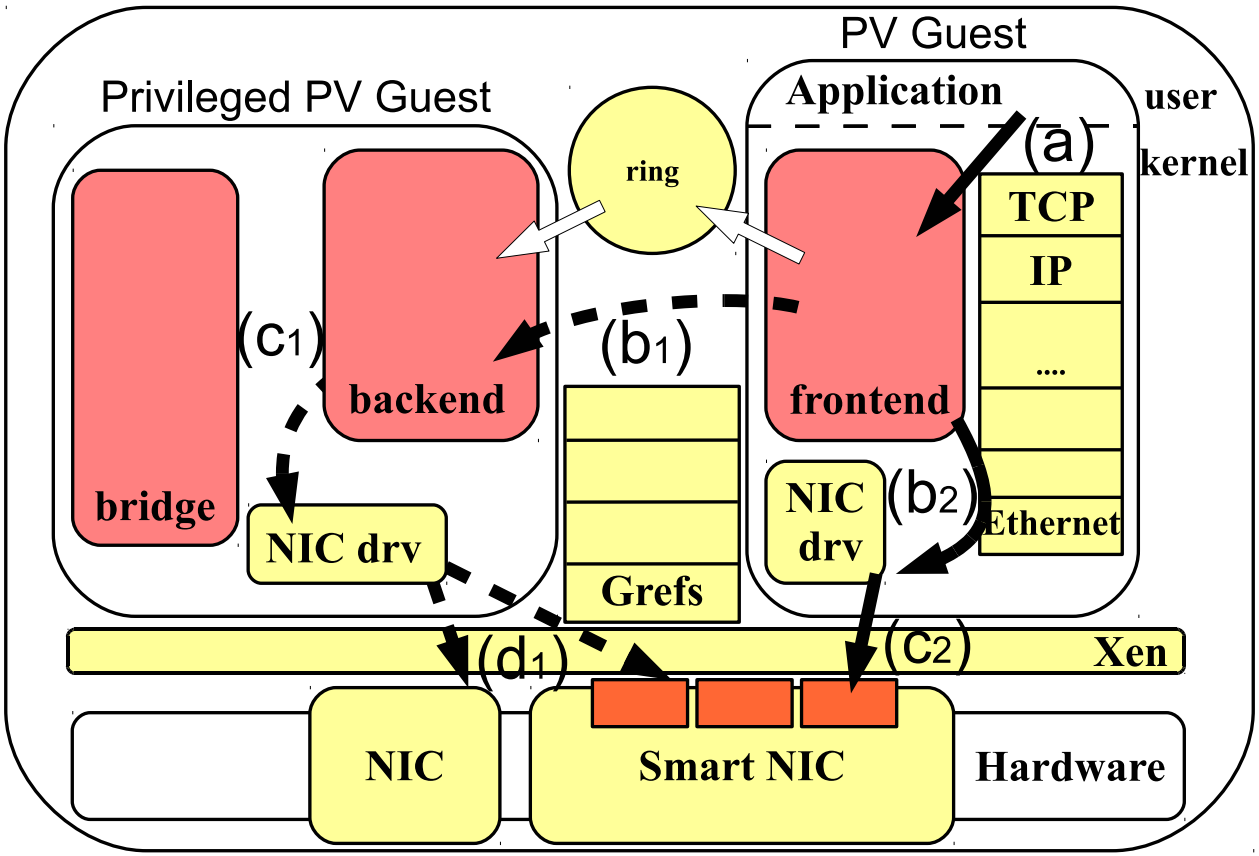
To applications: using existing semantics

- event channel mechanism (Xen)
- common communication techniques (events, asynchronous notification, etc.)

To drivers: using Xen hypercalls

- glue/wrapper–code for native drivers
- smart mappings to achieve hypervisor/driver domain-bypass data exchange
- keep a software I/O TLB

Proposed Architecture



$a \rightarrow b_1 \rightarrow c_1 \rightarrow d_1$: our initial approach (implemented)

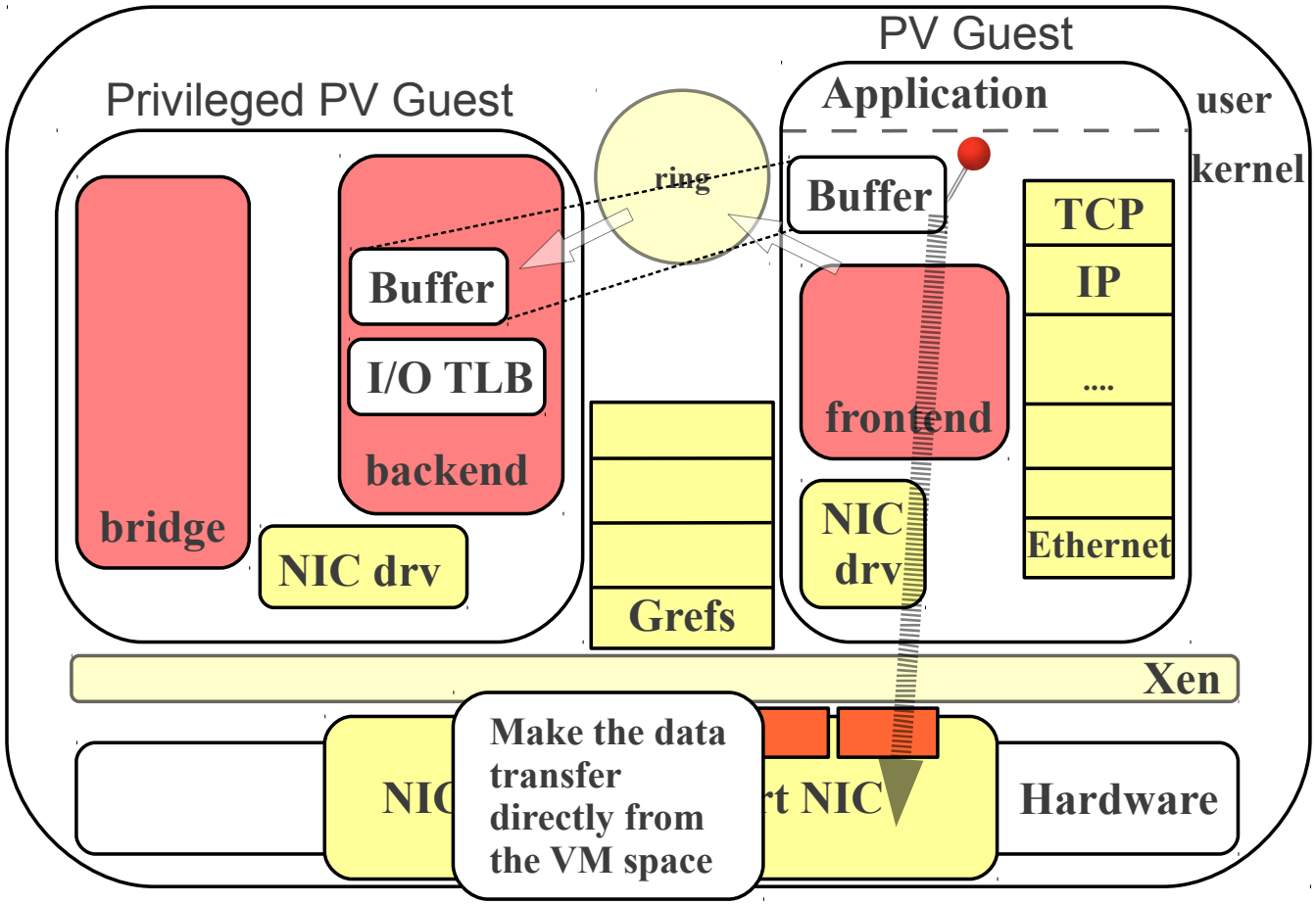
$a \rightarrow b_2 \rightarrow c_2$: alternative approach (IOV based)

Overview – Future Work

- finishing our framework (in progress) – porting a popural low-level protocol (Myrinet/MX / Open-MX) to support higher-level frameworks
- We aspire to:
  - Deploy MPI applications over our framework and evaluate its performance:
  - Examine SR-IOV techniques (lots of BUGs at the time of testing)
  - Explore multiple privileged domains to handle network traffic
  - Examine the posibility of using Hardware Accelerators in VM environments or exploiting them for faster Virtualization (less overheads)

Thanks!

Example – Data send



National Technical University of Athens CSLab