

Exploring I/O Virtualization Data paths for MPI Applications in a Cluster of VMs: A Networking Perspective

Anastassios Nanos, Georgios Goumas and Nectarios Koziris
{ananos,goumas,nkoziris}@cslab.ece.ntua.gr

VHPC 2010, Ischia – Naples, Italy

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



Aug. 31 2010

Introduction

In this work:

- network benchmarks
- real scientific application (advective equation)
- experiment with different data paths

Objective:

Explore the implications of deploying an HPC application in a cluster of VMs from a networking point of view



Outline

1 Introduction

- Motivation

2 Experimental Evaluation

- Testbed Setup
- Network Benchmarks
- Deploying an MPI application in a cluster of VMs
 - Process Placement
 - Performance Evaluation
- Discussion

3 Summary & Future Directions



Motivation – HPC Application demands

Cloud Computing infrastructures:

- provide
 - ▶ flexibility
 - ▶ dedicated, isolated execution of services
- are built on clusters of multicores
- offer huge processing power



Motivation – HPC Application demands

Cloud Computing infrastructures:

- provide
 - ▶ flexibility
 - ▶ dedicated, isolated execution of services
- are built on clusters of multicores
- offer huge processing power

High Performance Computing Applications

- communication libraries (MPI)
- bypass general purpose kernel mechanisms
 - ▶ 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 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 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 user-level interconnection networks (endpoint/process-based, isolation, multiplexing in firmware level)

Objective

framework to support user-level networking in VM environments



Why build such a framework ?

Towards this approach:

- network benchmarks in a cluster of Xen VMs
- scientific application from a networking perspective



Outline

1 Introduction

- Motivation

2 Experimental Evaluation

- Testbed Setup
- Network Benchmarks
- Deploying an MPI application in a cluster of VMs
 - Process Placement
 - Performance Evaluation
- Discussion

3 Summary & Future Directions

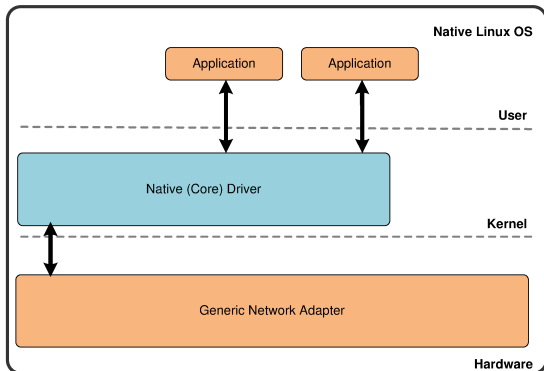


Testbed Setup

Using 2 dual quad-core VM containers we setup 8 dual-core VMs (4 in each VM container)

We consider 3 cases:

- **NATIVE**
- **BRIDGED**
- **IOV**

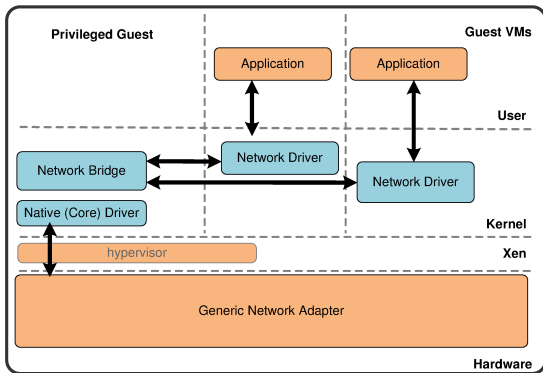


Testbed Setup

Using 2 dual quad-core VM containers we setup 8 dual-core VMs (4 in each VM container)

We consider 3 cases:

- NATIVE
- **BRIDGED**
- IOV

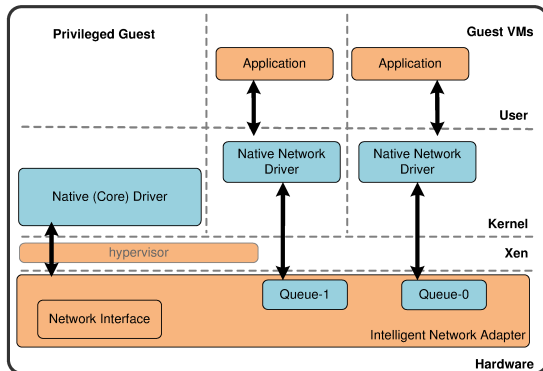


Testbed Setup

Using 2 dual quad-core VM containers we setup 8 dual-core VMs (4 in each VM container)

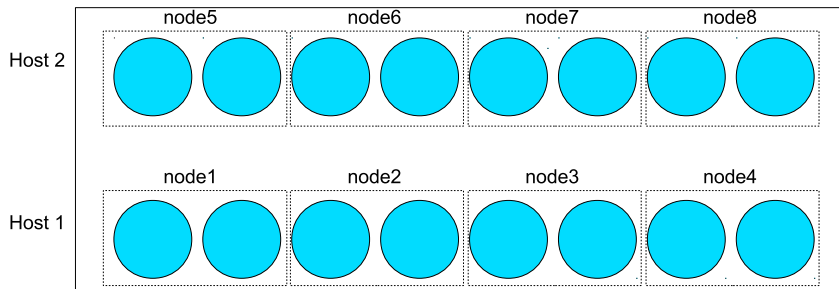
We consider 3 cases:

- NATIVE
- BRIDGED
- IOV



Network Benchmarks

- netperf
- iperf



Network Benchmarks – Netperf

- netperf
- iperf

Bandwidth achieved in MiB/sec

	node1	node2	node3	node4	total
NATIVE					811.73
BRIDGED	90.45	123.03	112.23	100.26	425.97
IOV	160.33	159.43	152.45	162.63	634.84

- degraded performance
- IOV outperforms the BRIDGED case



Network Benchmarks – Iperf

- netperf
- **iperf**

Bandwidth achieved in MiB/sec

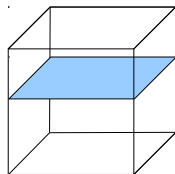
	node1	node2	node3	node4	total
NATIVE					1238
BRIDGED	205.00	190.00	181.25	172.50	748.75
IOV	221.25	222.25	221.25	220.00	884.75

- NATIVE achieves the theoretical maximum
- IOV and BRIDGED are ≈ 350 and 500 MB/sec lower

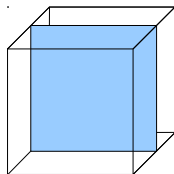


Deploying an MPI application in a cluster of VMs

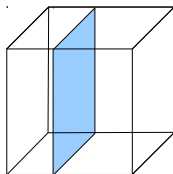
- Advective process
- PDE stencil (3D, nearest neighbor communication)
- linear communication pattern
- $T=512$ $X, Y, Z = 512, 512, 512$
- e.g. 16-core experiment: $1 \times 1 \times 16$, $1 \times 16 \times 1$, $16 \times 1 \times 1$



X



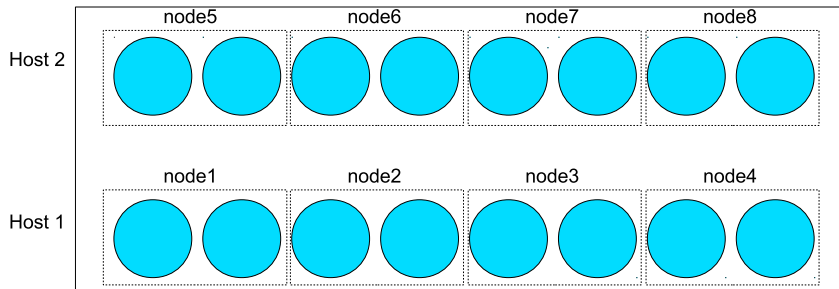
Y



Z

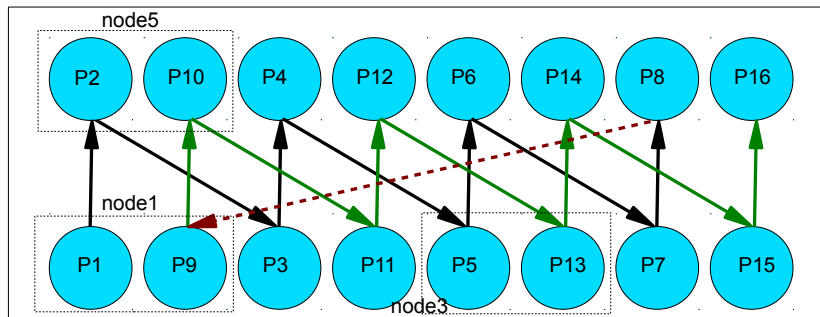
Process Placement

We place all MPI processes using different configurations in order to study the application's behavior and the optimum ad-hoc data path



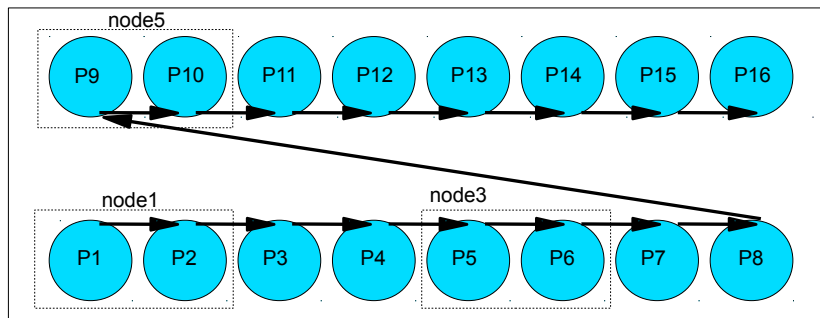
Process Placement – Inter-node

100% of communication messages are on the network



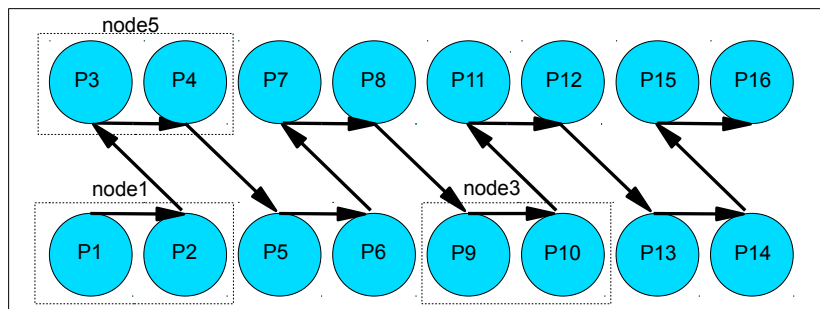
Process Placement – Intra-node

$\approx 6\%$ of communication messages are on the network



Process Placement – Hybrid

$\approx 46\%$ of communication messages are on the network



Performance Evaluation

Inter-node communication

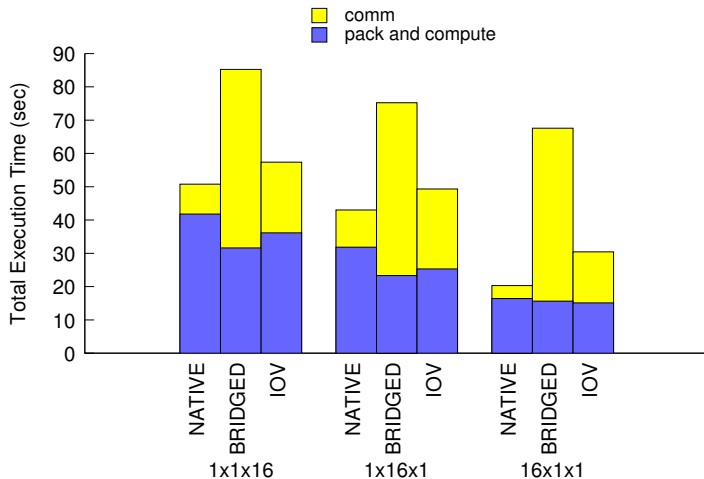
Intra-node communication

Scaling

Speedup



Performance Evaluation – Inter-node



100% communication messages on the network

Performance Evaluation – Inter-node

Inter-node communication

- IOV outperforms the BRIDGED case
- achieves $\approx 80\%$ of the NATIVE case

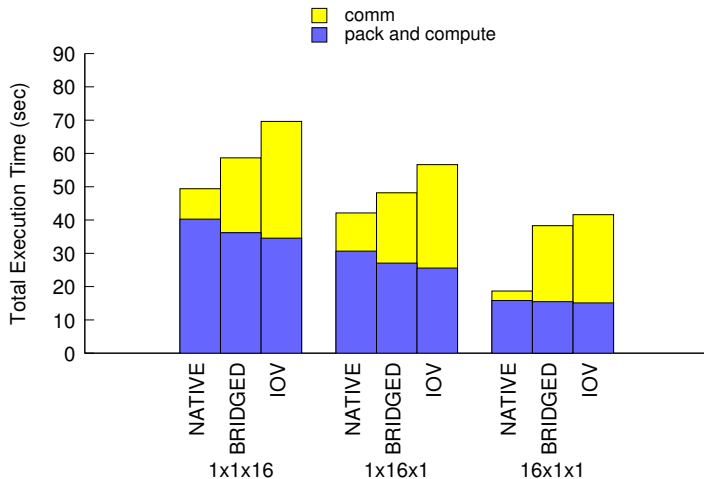
Intra-node communication

Scaling

Speedup



Performance Evaluation – Intra-node



≈ 6% communication messages on the network

Performance Evaluation – Intra-node

Inter-node communication

- IOV outperforms the BRIDGED case
- achieves $\approx 80\%$ of the NATIVE case

Intra-node communication

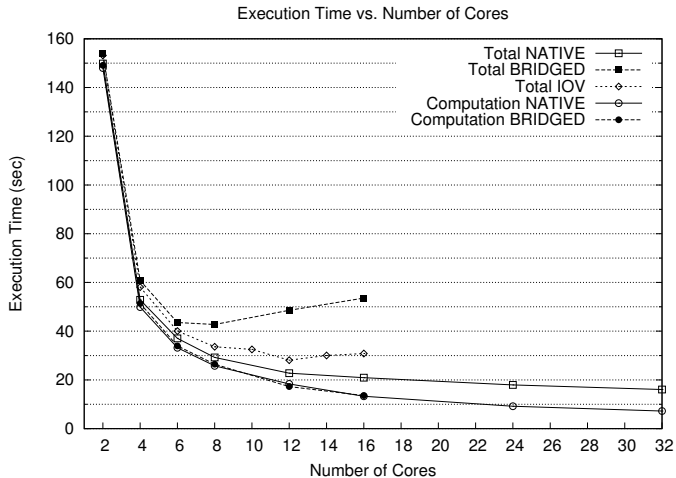
- performance degradation in the IOV case

Scaling

Speedup



Performance Evaluation – Scaling



Performance Evaluation – Scaling

Inter-node communication

- IOV outperforms the BRIDGED case
- achieves $\approx 80\%$ of the NATIVE case

Intra-node communication

- performance degradation in the IOV case

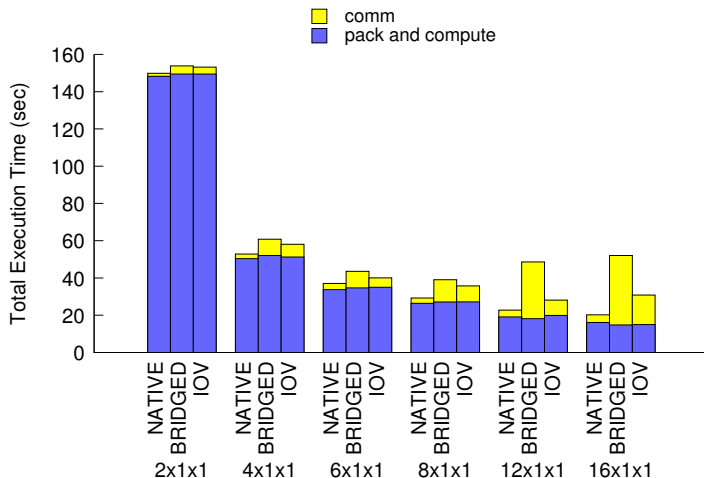
Scaling

- computation time remains the same for all cases

Speedup



Performance Evaluation – Scaling



Performance Evaluation – Scaling

Inter-node communication

- IOV outperforms the BRIDGED case
- achieves $\approx 80\%$ of the NATIVE case

Intra-node communication

- performance degradation in the IOV case

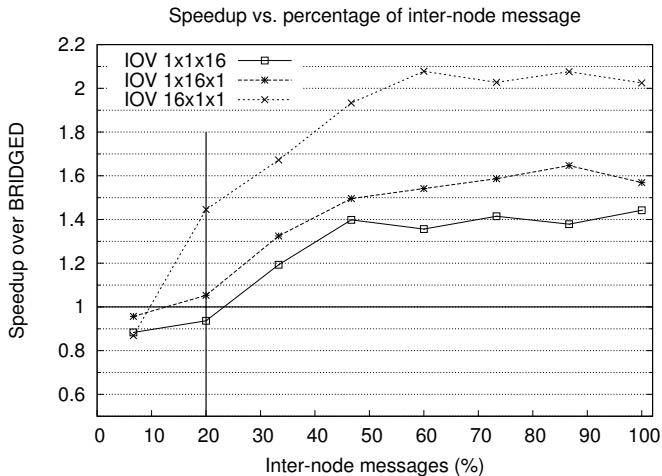
Scaling

- computation time remains the same for all cases
- more than 8 cores \Rightarrow communication overhead

Speedup



Performance Evaluation – Speedup



Performance Evaluation – Summary

Inter-node communication

- IOV outperforms the BRIDGED case
- achieves $\approx 80\%$ of the NATIVE case

Intra-node communication

- performance degradation in the IOV case

Scaling

- computation time remains the same for all cases
- more than 8 cores \Rightarrow communication overhead

Speedup

- IOV when more than 20% of the messages on the network



Discussion

Virtualization

- CPU, Memory, I/O



Discussion

Virtualization

- CPU, Memory, I/O

Xen Networking

- Generic case puts pressure on the privileged guest
- IOV requires specialized hardware and specific software support
- SR/MR IOV is currently implemented for ethernet adapters



Discussion

Virtualization

- CPU, Memory, I/O

Xen Networking

- Generic case puts pressure on the privileged guest
- IOV requires specialized hardware and specific software support
- SR/MR IOV is currently implemented for ethernet adapters

HPC applications in clusters of VMs

- Computation takes almost the same time: The apparent overhead is due to the communication.



Outline

1 Introduction

- Motivation

2 Experimental Evaluation

- Testbed Setup
- Network Benchmarks
- Deploying an MPI application in a cluster of VMs
 - Process Placement
 - Performance Evaluation
- Discussion

3 Summary & Future Directions



Summary

- application characterization
- alternative data paths for network communication

HPC applications in VM environments

- communication pattern
- optimum ad-hoc data path

We propose

- A framework to support High-performance Interconnects in VM environments.
- Protocol processing & multiplexing in firmware level



Future directions

- evaluate different applications from a networking point of view
- synthetic microbenchmark integrating various communication patterns



Future directions

- evaluate different applications from a networking point of view
 - synthetic microbenchmark integrating various communication patterns
-
- shared memory
 - higher level frameworks for application parallelism (MapReduce)



Future directions

- evaluate different applications from a networking point of view
 - synthetic microbenchmark integrating various communication patterns
-
- shared memory
 - higher level frameworks for application parallelism (MapReduce)
-
- develop a simple communication protocol to deploy these applications on top of our custom interconnect (RDMA over generic Ethernet)



Thanks!

Questions?

