

MyriXen: Message passing in Xen VMs over Myrinet and Ethernet

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

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



August 25, 2009



- 1 Introduction
 - Motivation
- 2 Background
 - Xen Network I/O Architecture
 - Myrinet/MX
- 3 MyriXen Design
 - Architecture
 - Software Stack
- 4 Summary & Future Directions



- 1 Introduction
 - Motivation
- 2 Background
 - Xen Network I/O Architecture
 - Myrinet/MX
- 3 MyriXen Design
 - Architecture
 - Software Stack
- 4 Summary & Future Directions



Cloud Computing research

- consolidating
- building large data centers
- power/cooling
- Virtualization
- workloads



Cloud Computing research

- consolidating
- building large data centers
- power/cooling
- Virtualization
- workloads

HPC applications

- overheads by Virtualization layers



Cloud Computing research

- consolidating
- building large data centers
- power/cooling
- Virtualization
- workloads

HPC applications

- overheads by Virtualization layers
- offloading needs hardware support



Cloud Computing research

- consolidating
- building large data centers
- power/cooling
- Virtualization
- workloads

HPC applications

- overheads by Virtualization layers
- offloading needs hardware support
- bridging the gap between High-performance I/O needs and Virtualization techniques



Why not benefit from Virtualization ?

- Clusters of VMs instead of COWs – Clusters of SMPs
- semantics in communication are Virtualization aware
- frameworks for secured, controlled access to hardware



Why not benefit from Virtualization ?

- Clusters of VMs instead of COWs – Clusters of SMPs
- semantics in communication are Virtualization aware
- frameworks for secured, controlled access to hardware

What do we need ?

Architectural support

Why not benefit from Virtualization ?

- Clusters of VMs instead of COWs – Clusters of SMPs
- semantics in communication are Virtualization aware
- frameworks for secured, controlled access to hardware

What do we need ?

Architectural support





- 1 Introduction
 - Motivation
- 2 Background
 - Xen Network I/O Architecture
 - Myrinet/MX
- 3 MyriXen Design
 - Architecture
 - Software Stack
- 4 Summary & Future Directions



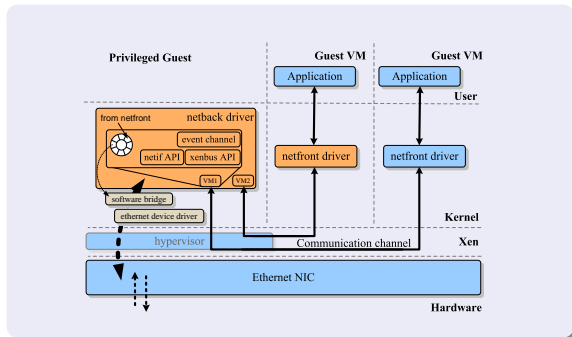
- 1 Introduction
 - Motivation
- 2 Background
 - Xen Network I/O Architecture
 - Myrinet/MX
- 3 MyriXen Design
 - Architecture
 - Software Stack
- 4 Summary & Future Directions

Background



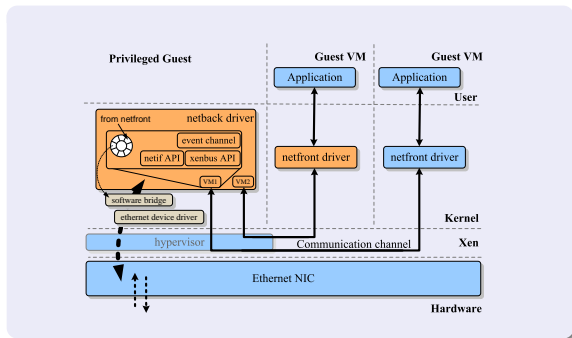
- Xen net I/O
- Myrinet/MX

- Xen net I/O
- Myrinet/MX



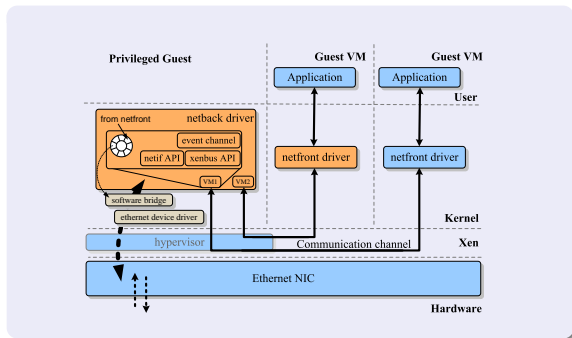
- driver domains
- memory handling (grants)

- Xen net I/O
- Myrinet/MX



- driver domains
- memory handling (grants)
- copies — page flipping

- Xen net I/O
- Myrinet/MX



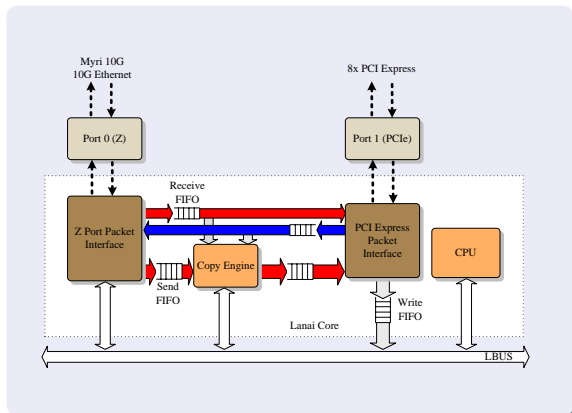
- driver domains
- memory handling (grants)
- copies — page flipping
- split driver model
 - ▶ frontent / backend
 - ▶ event channels
 - ▶ interrupt handling (?)

Background – Myrinet/MX



- Xen net I/O
- **Myrinet/MX**

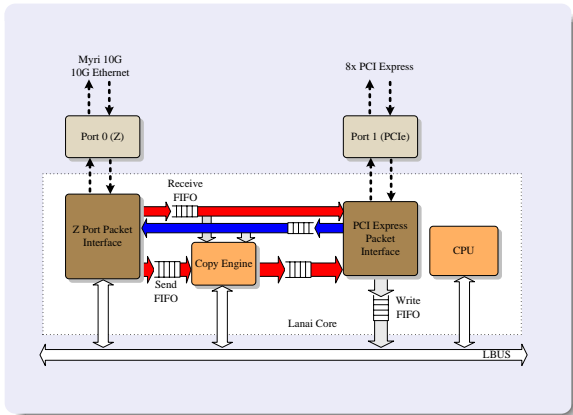
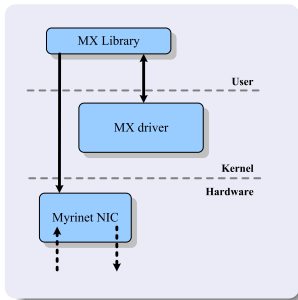
- Xen net I/O
- **Myrinet/MX**



- smart (programmable) NIC

Background – Myrinet/MX

- Xen net I/O
- **Myrinet/MX**

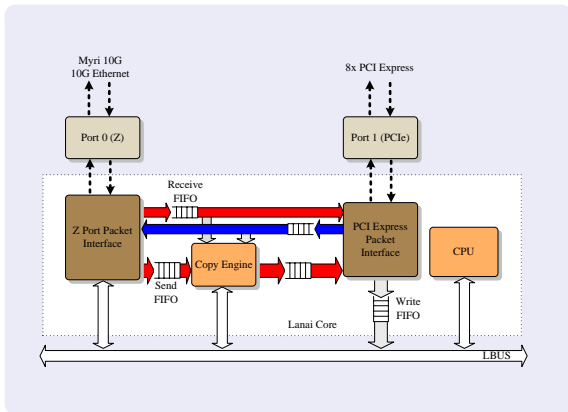
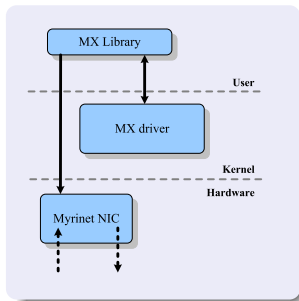


- smart (programmable) NIC
- User-level protocol (MX)



Background – Myrinet/MX

- Xen net I/O
- **Myrinet/MX**



- smart (programmable) NIC
- User-level protocol (MX)
- MX endpoint: instance exported to userspace



- 1 Introduction
 - Motivation
- 2 Background
 - Xen Network I/O Architecture
 - Myrinet/MX
- 3 MyriXen Design
 - Architecture
 - Software Stack
- 4 Summary & Future Directions



- We need a secure, controlled way to access hardware



- We need a secure, controlled way to access hardware
 - ▶ without compromising system's security
 - ▶ fitted in Xen
 - ▶ in an $1 \longleftrightarrow N$ way



- We need a secure, controlled way to access hardware
 - ▶ Xen's split driver model



- We need a secure, controlled way to access hardware
 - ▶ Xen's split driver model
- We need a fast, copy-free way to exchange messages



- We need a secure, controlled way to access hardware
 - ▶ Xen's split driver model
- We need a fast, copy-free way to exchange messages
 - ▶ without overheads imposed by intermediate software layers



- We need a secure, controlled way to access hardware
 - ▶ Xen's split driver model
- We need a fast, copy-free way to exchange messages
 - ▶ build a direct application-to-NIC data path



- We need a secure, controlled way to access hardware
 - ▶ Xen's split driver model
- We need a fast, copy-free way to exchange messages
 - ▶ build a direct application-to-NIC data path

How do we combine these two ?

... by integrating MX semantics (MX endpoints, protection) to the Xen split driver model.

So, we build:



myrifront

- passes requests to the backend via the event channel mechanism

So, we build:

myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

So, we build:



myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

myriback

So, we build:



myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

myriback

- forward requests from myrifront to the core MX driver

So, we build:



myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

myriback

- forward requests from myrifront to the core MX driver
- event channels

So, we build:



myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

myriback

- forward requests from myrifront to the core MX driver
- event channels
- install direct app-to-NIC data path (grants + PCI memory map)

So, we build:



myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

myriback

- forward requests from myrifront to the core MX driver
- event channels
- install direct app-to-NIC data path (grants + PCI memory map)

event channels

So, we build:

myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

myriback

- forward requests from myrifront to the core MX driver
- event channels
- install direct app-to-NIC data path (grants + PCI memory map)

event channels

- Xen \leftrightarrow data path

So, we build:



myrifront

- passes requests to the backend via the event channel mechanism
- communicates with the application preserving MX semantics

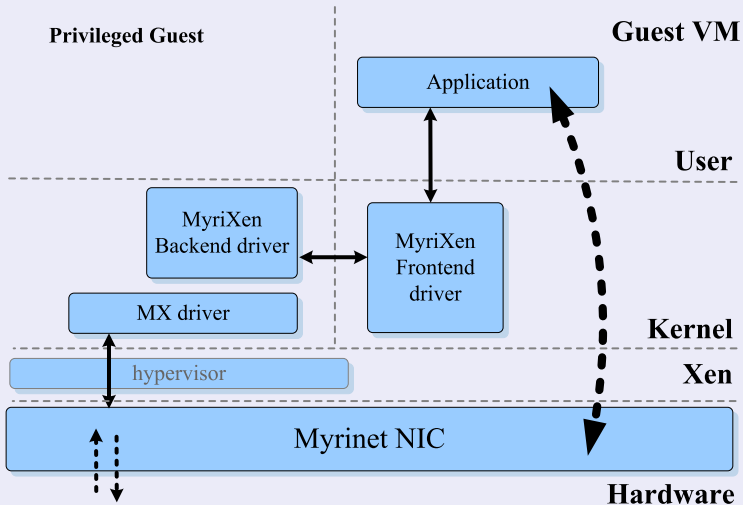
myriback

- forward requests from myrifront to the core MX driver
- event channels
- install direct app-to-NIC data path (grants + PCI memory map)

event channels

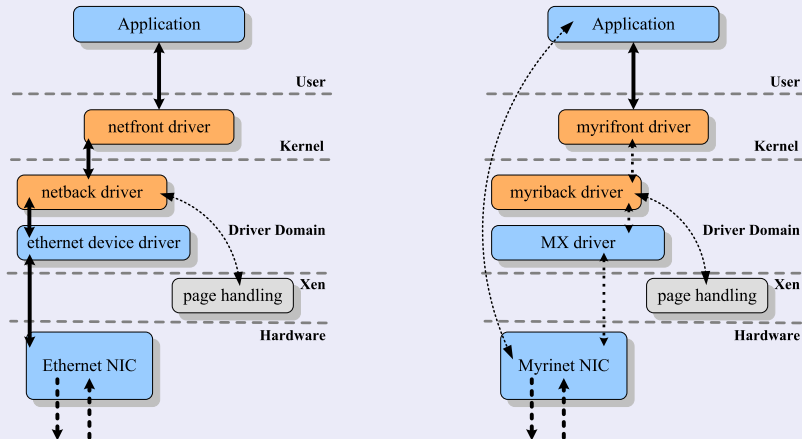
- Xen \leftrightarrow data path
- MyriXen \leftrightarrow control path

MyriXen Architecture



Xen and MyriXen comparison

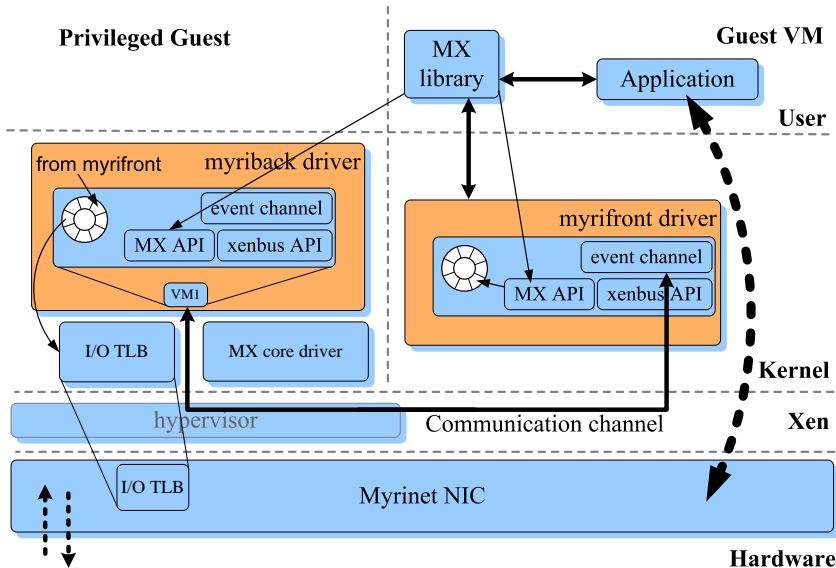
Software Stack





- Initialization
- Endpoint management
- Memory Registration
- Message Matching
- Address translation

MyriXen Architecture – Network I/O



Outline



- 1 Introduction
 - Motivation
- 2 Background
 - Xen Network I/O Architecture
 - Myrinet/MX
- 3 MyriXen Design
 - Architecture
 - Software Stack
- 4 Summary & Future Directions

Summary



- HPC in Cloud Computing
 - ▶ software management freedom
 - ▶ consolidation of services & application execution
 - ▶ workloads

Summary



- HPC in Cloud Computing
 - ▶ software management freedom
 - ▶ consolidation of services & application execution
 - ▶ workloads
- MyriXen
 - ▶ Integrating MX semantics in Virtualization platforms
 - ▶ Build direct data paths eliminating copies but ...



- HPC in Cloud Computing
 - ▶ software management freedom
 - ▶ consolidation of services & application execution
 - ▶ workloads
- MyriXen
 - ▶ Integrating MX semantics in Virtualization platforms
 - ▶ Build direct data paths eliminating copies but ...

keep secured, controlled access to hardware

Future directions



- We are currently implementing MyriXen

Future directions



- We are currently implementing MyriXen
- Deploy MPI applications on top of MyriXen in a cluster of VMs

Future directions



- We are currently implementing MyriXen
- Deploy MPI applications on top of MyriXen in a cluster of VMs
- Explore overheads that will arise

Future directions



- We are currently implementing MyriXen
- Deploy MPI applications on top of MyriXen in a cluster of VMs
- Explore overheads that will arise
- firmware changes (we need Myricom's firmware 😊)

Thanks!



Questions?



- leaving init, endpoint management, memory registration out of the critical path



- leaving init, endpoint management, memory registration out of the critical path
- decoupling data transfers from the Virtualization layers



- leaving init, endpoint management, memory registration out of the critical path
- decoupling data transfers from the Virtualization layers
- isolation



- leaving init, endpoint management, memory registration out of the critical path
- decoupling data transfers from the Virtualization layers
- isolation
- CPU & memory contention on the NIC