# Towards Virtual Passthrough I/O on Commodity Devices

*Lei Xia, Jack Lange, Peter Dinda*

{*lxia, jarusl, pdinda*}*@northwestern.edu*

Department of
Electrical Engineering and Computer Science
Northwestern University

http://v3vee.org/

http://www.presciencelab.org/

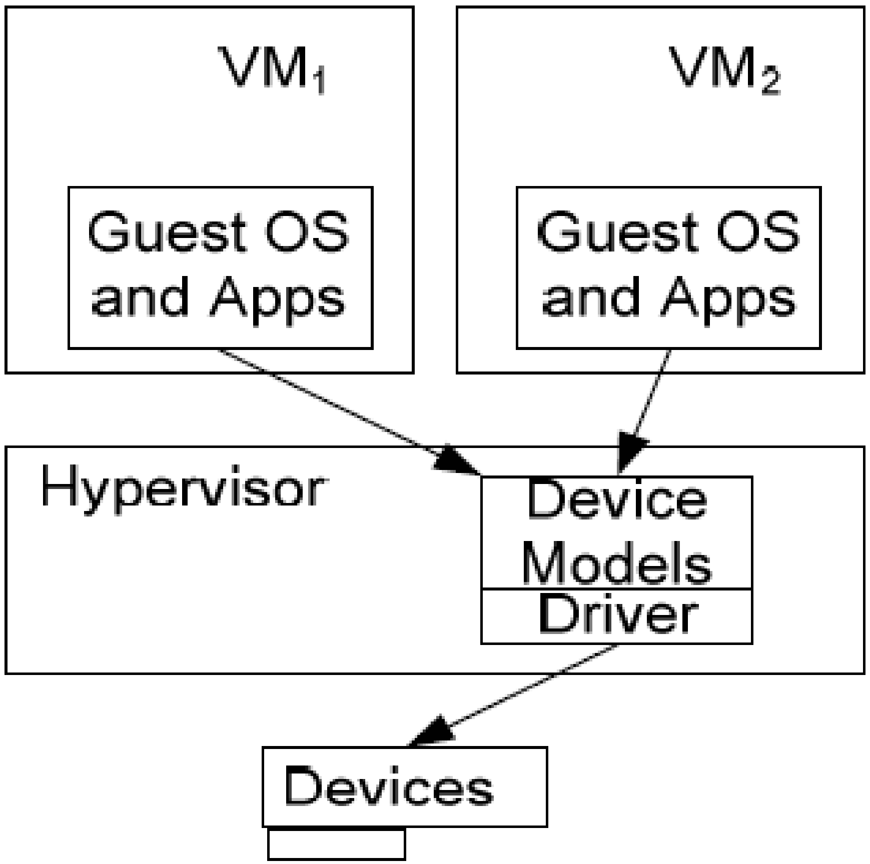
# Overview

- VPIO: A modeling-based approach to high performance I/O virtualization for commodity devices
  - Models could be provided by HW vendors
- Intermediate option between passthrough I/O, and emulated I/O
- Promising initial results
- Work in progress

# Outline

- ✓ I/O virtualization techniques
- ✓ Idea of VPIO
- ✓ Palacios VMM
- ✓ VPIO system
- ✓ Device Model
- ✓ Discussion
- ✓ Conclusion

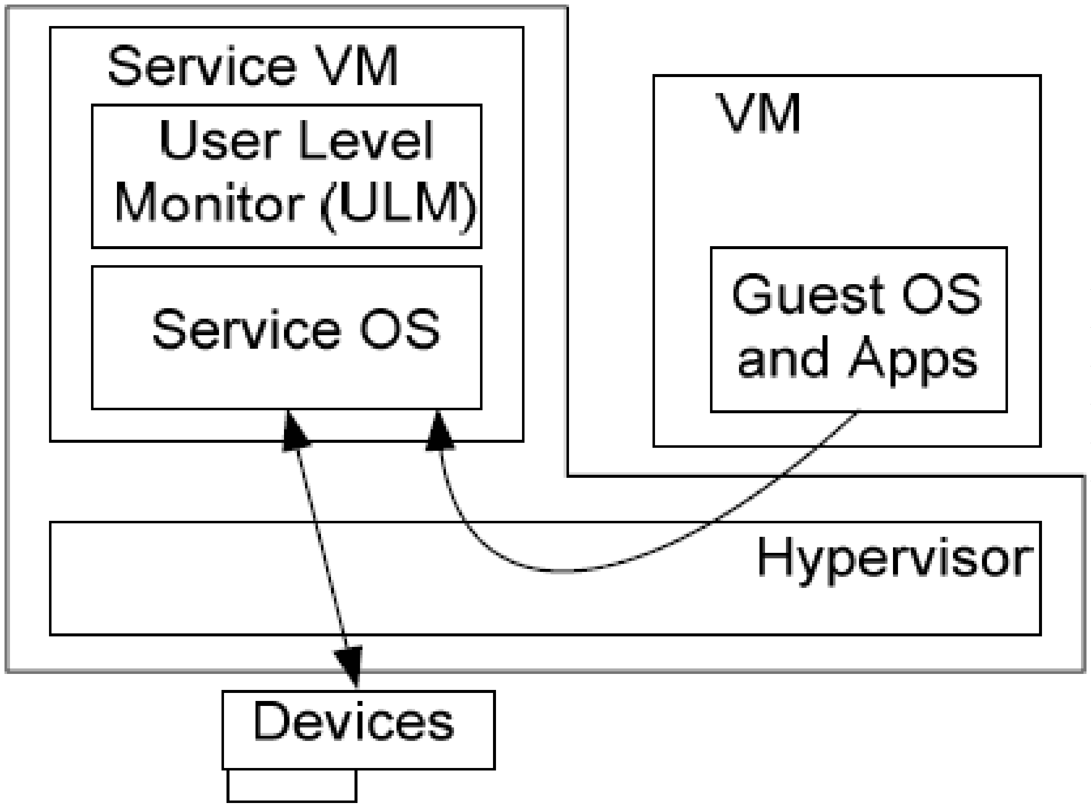
# I/O virtualization – full emulated I/O




- No guest software change
- Easy Migration
- All New Device Drivers
- High performance overhead

[Sugerman01]

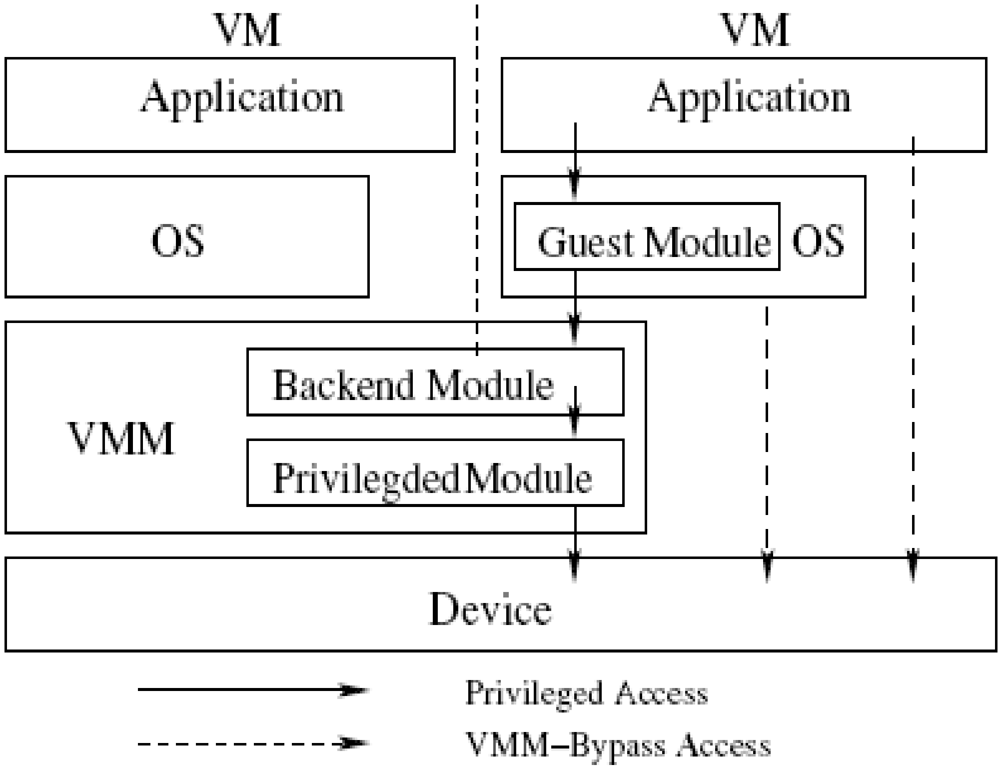
# I/O virtualization – paravirtualized I/O




- Performance
- Reuse Device Driver
- Change guest device driver

[Barham03, Levasseur04]

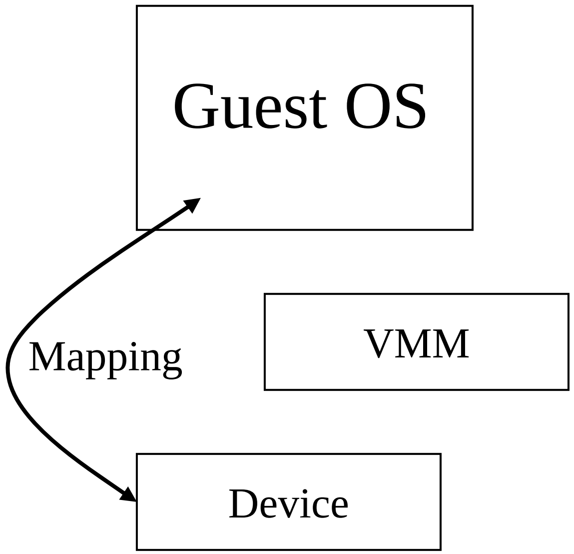
# I/O virtualization – Passthrough I/O




- Native performance
- Guest responsible for device driver
- Specialized hardware support
  - Self-virtualizing devices
- Migration

[Liu06, Raj07,Shafer07]

# I/O virtualization – Direct-Mapped I/O




- Reusablity/multiplexing
- Security Issue

# VPIO Goals

- Achieve *safe* and *secure* direct-mapped I/O
- With reusability/multiplexing
- To support commodity devices
  - No self-virtualized features
- Without losing too much performance
  - Expect a little more performance overhead compared to pass-through IO

# Two Requirements For VPIO

- Inexpensive formal model of device
  - Building model easier than building device driver
  - Inexpensive to drive such model
- Device can be context-switched

# Core Idea of VPIO

- VMM maintains a formal model of device
  - keeps track of the physical device status
  - driven by guest/device interactions
- Model can determines
  - Reusable state – whether device is currently serially reusable
  - DMA – whether a DMA is about to starting and what memory locations will be used
  - Other interesting states

# Introducing Palacios

- New VMM for HPC, architecture, systems, teaching, and other uses
  – Fully Open Source, BSD License
- Type I VMM, embeddable into existing kernels
- Operating System independent
  – Kitten HPC OS (Sandia National Labs)
  – GeekOS (University of Maryland)
- Implemented using Hardware Virtualization Extensions
- Part of the V3VEE project
  – Collaboration between NU and UNM
  – http://www.v3vee.org
- Available at:
  – http://www.v3vee.org/download

# Palacios Overview

- Supports 32 and 64 bit environments
  – Hosts and Guests
  – Currently supports Linux Guests
- Currently uses AMD SVM extensions
  – partial Intel VT support
- Full hardware virtualization
  – Does not use paravirtualization
  – Includes complement of virtual devices

- More details:
  – J. Lange, P. Dinda, “An Introduction to the Palacios Virtual Machine Monitor---Release 1.0”, Northwestern University EECS Technical Report NWU-EECS-08-11, November, 2008.
  – See us for more info

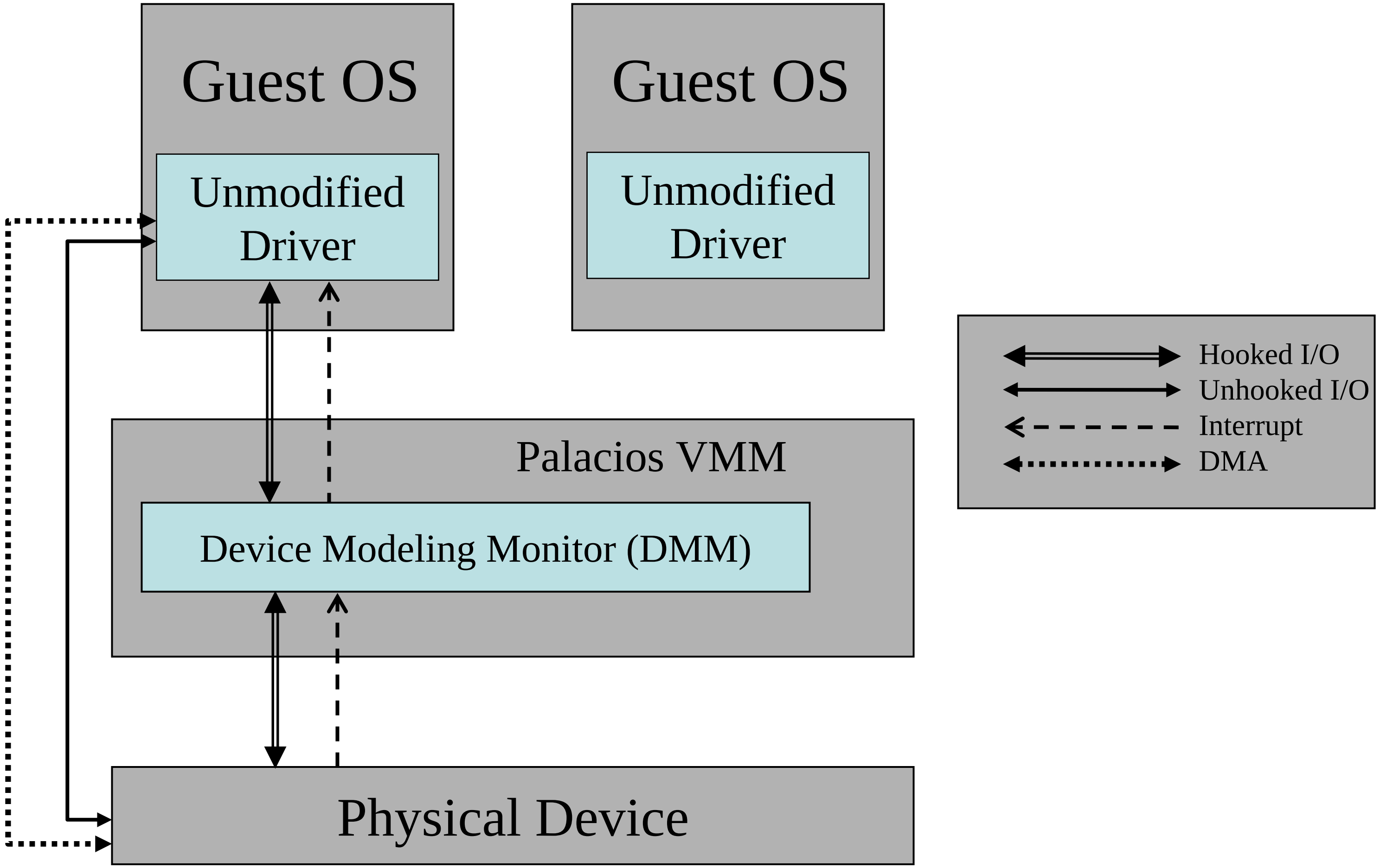
# Palacios Details

- Virtualization Interface
  - Hook IO Ports
  - Hook Memory Regions
  - Hook interrupts
  - Handle VMExits
- Host Interface
  - Access standard OS facilities
    - (debugging, memory allocation)
  - Hook host events
    - Interrupts, timer, keyboard, etc…
- Can use shadow or nested paging

# Palacios People

- Northwestern University
  - Jack Lange (Lead Ph.D student)
  - Lei Xia (Ph.D student)
  - Peter Dinda (PI, Project Lead)
- University of New Mexico
  - Zheng Cui (Ph.D student)
  - Patrick Bridges (PI)
- Others
  - Trammell Hudson and Kevin Pedreti (Sandia)

# VPIO In Context Of Palacios



# Current Status

- VPIO is a work in progress
- What is implemented in Palacios
  - hook I/O ports
  - hook memory address (byte)
- What is tested outside
  - Device Model embedded and tested on QEMU PC emulator version 0.9.1

# Device requests and interrupts

- Guest talks to device by IN/OUT
  - Memory-mapped I/O will be similar
  - *hooked I/O* list, a list of I/O port operations for read/write or both. VMM intercepts these I/Os
  - I/O port reads/writes drive model and HW
  - *unhooked I/O* list (ideally as large as possible)
- Interrupts
  - All physical interrupts are handled by VMM
  - Interrupts drive model and are also delivered to guest

# DMA

- DMA is initiated directly on physical device by guest
  - DMM is aware of guest's DMA operations due to hooked ports
  - DMA destination physical address is checked before the physical DMA operation is performed
  - If validated, let DMA occur, otherwise, deny it.
- Dealing with DMA failure
  - How to notify the guest?
    - Ignore the DMA?
    - Machine Check Exception?
- Challenge: Dealing with physical address translations

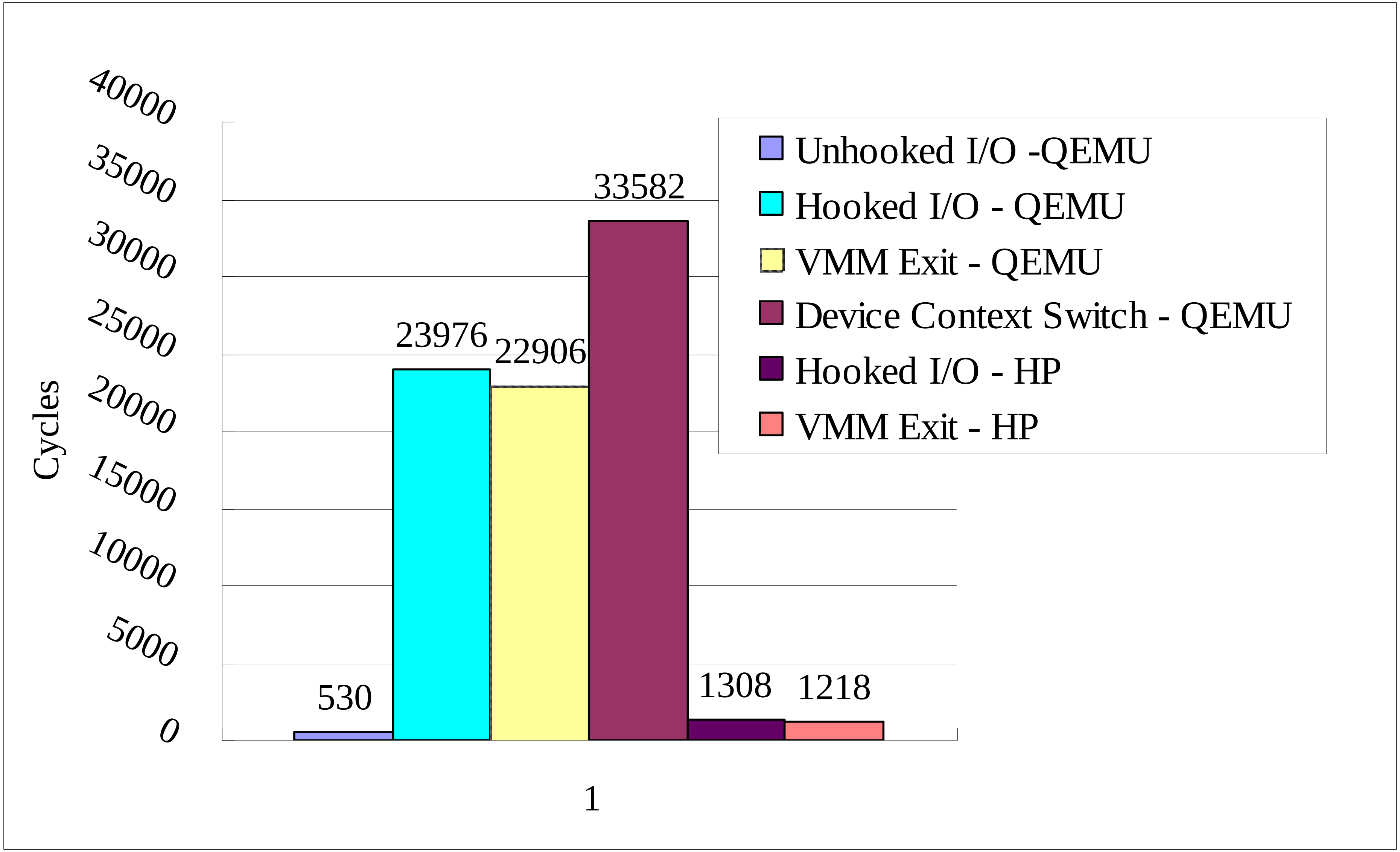
# Device Multiplexing

- Context switch between guests
  - Switching when device in reusable state
  - Device context (related registers, model)
  - If not owning device, guest's operation requests are suspended
- Challenge: Device handoff on interrupt
  - Handle incoming packet for NIC
  - Coming back later

# Cost of VPIO – Experimental Setup

- Guests' I/Os performance overhead
  - Palacios running on Qemu and HP system
  - Qemu PC emulator 0.9.1
  - HP ML115 1.8GHz, AMD Opteron 1210

# Issue: Cost of exits (Palacios / AMD SVM)



# VPIO Issue 1/2 : Exit Costs

- Fundamental issue: O(1000) cycle exits.
- Try to minimize number of hooked I/O ports
  - Model is cheaper in terms of exits than an emulated device
  - Not all I/O ports are needed to drive model

# VPIO Issue 2/2 : Model

- Can we build such models?
  - Is it feasible to build the model?
  - How easy to build such model, easier than device driver?
- How expensive are they to run?

# Device Model

- Not for verification, not a complete behavioral model
- Only used to determine…
  - Whether and when the device is reusable
  - Whether DMA is to be initiated
  - What device requests are needed to update model
- State machine, with extra information
  - Events: device requests, interrupts
  - *Checking functions*, triggered by state transition
  - State transition is approved or denied

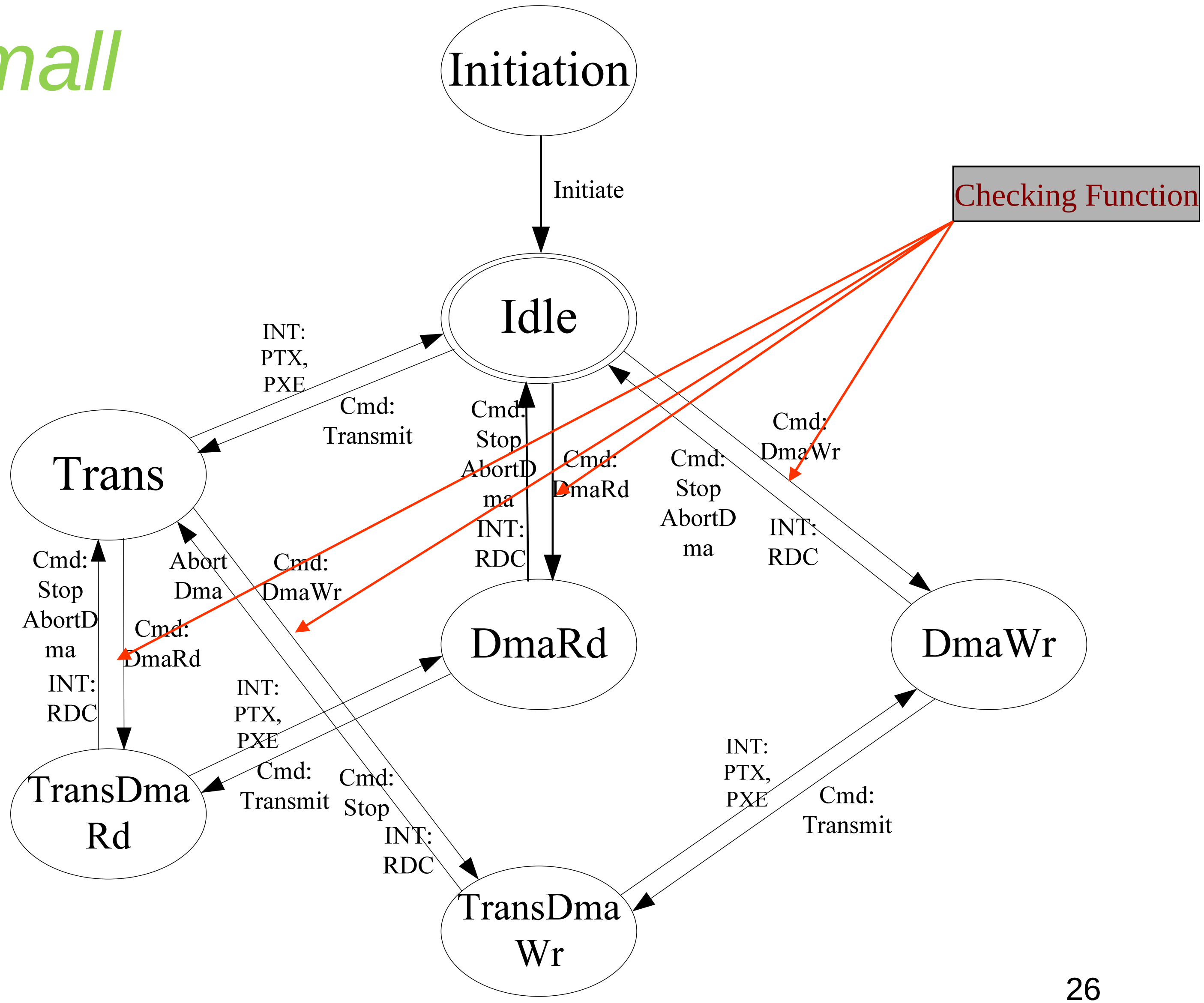
# Experimental Setup

- Testing setup
  - Embedded model in QEMU PC emulator version 0.9.1
  - Tested model on a set of network applications running in guest OS on Qemu

# NE2K NIC Example

*Model is small*

*~700 LOC*



# NE2K Model Performance

| Scenario | Total I/O | I/O hooked | Ratio (%) |
|---|---|---|---|
| Linux: ssh | 1865055 | 257324 | 13.80 |
| Linux: small dl | 2602002 | 69700 | 2.68 |
| Linux: large dl | 294508810 | 9429917 | 3.20 |
| Windows: ssh | 3769071 | 286671 | 7.61 |
| Windows: small dl | 1081738 | 39089 | 3.61 |
| Windows: large dl | 132898230 | 988535 | 0.74 |

*Windows XP sp2*
*Ubuntu Linux 6.**

*Only a small fraction of I/Os are needed to drive model, Great!*

# Experience with NE2K Model

- Only about 1 in 30 I/O port reads/writes need to be intercepted to drive the model

- Average non-exit cost of updating the model is ~*100* cycles
  - And could be done in parallel with device execution

# Challenges

- VM exit performance is primary concern
  - Further reduce I/O operations intercepted
  - Move model into guest
    - Either cooperatively or by code injection
- Handling incoming device input
  - Network card receive when incorrect guest has control of NIC
- Hardware manufacturers could provide models along with device drivers

# Summary

- VPIO: A modeling-based approach to high performance I/O virtualization for commodity devices
- Intermediate option between passthrough I/O and emulated I/O
- Promising initial results
- Work in progress

# Thanks!

# Questions??

*http://v3vee.org/*
*http://www.presciencelab.org/*


*lxia@northwestern.edu*
*http://www.cs.northwestern.edu/~lxi990/*