# High-Performance Hypervisor Architectures: Virtualization in HPC Systems

Ada Gavrilovska, Sanjay Kumar, Himanshu Raj, Karsten Schwan, Vishakha Gupta, Ripal Nathuji, Radhika Niranjan, Adit Ranadive, Purav Sarayia
College of Computing
Georgia Tech

# Virtualization and HPC

- Virtualization technology major focus in enterprise settings
  - power, cost, consolidation; manageability and portability…
- Adoption lags behind in HPC domain
  - fear of tapping into scare HPC platforms' resources
  - power, cost, consolidation – not critical constraints in HPC environments
- Our objective: understand feasibility and utility
  - Is there room and need for virtualization in HPC?
  - As we move to many-core?
  - Any new functionality/services?

# Overview

- Potential benefits
- Sidecore approach to VMM architecture design
  - Scalable hypervisor architectures for future many-core platforms
- Self-virtualizing devices (and accelerators)
  - challenges and opportunities they present
- Platform management in virtualized environments

- Ongoing work targeting general purpose multicore systems, from low-end, personal platforms to high-end data center environments

# Potential benefits

- Fault-tolerance: migration
- Fault-tolerance: monitoring
- Shared I/O and service nodes
- New functionality
- Portability and manageability
- Development, debugging and sharing
- Mixed use for capacity & capability computing

- => Worth further investigation…

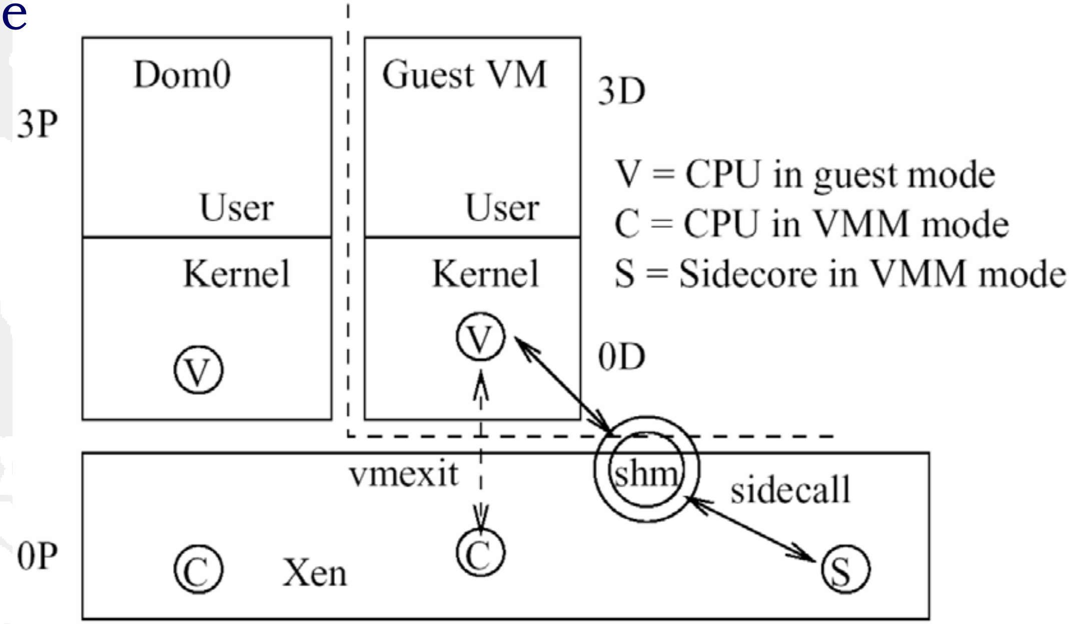
# Sidecore Approach

- VMMs in many-core platforms
  - coordinating VMM operations across many (80!) cores may introduce prohibitive noise levels and resource requirements
- Decompose VMM functionality
  - factor select subsets of VMM operations and assign their execution to a designated core(s)
  - eliminate or reduce expensive context/VMentry/VMexit switches; exploit locality
  - improve VMM scalability to number of cores
  - Sidecore-resident functionality
    - factored out from monolithic VMMs (e.g., Xen)
    - components in future modular/lightweight VMMs
- Architectural considerations
  - number and location of sidecores
  - VM core – Sidecore communication channels

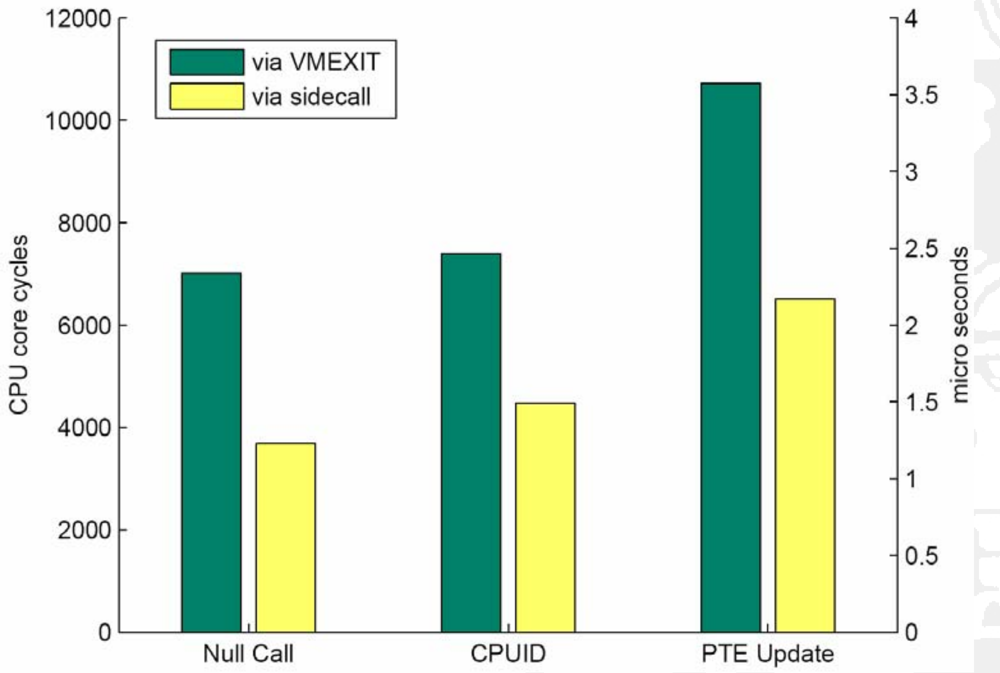
# Page Table manipulation with Sidecore

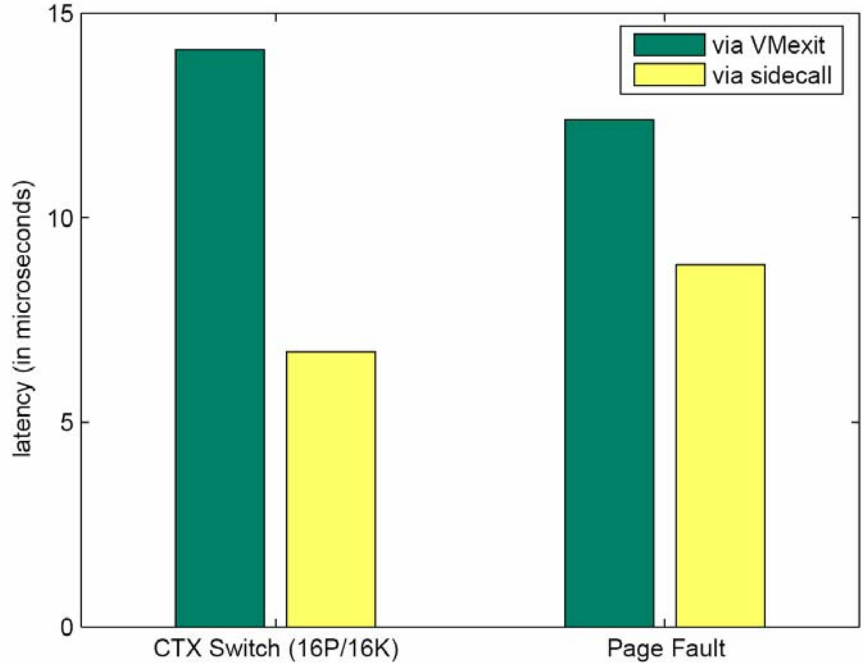
Case study:

- Modified Xen 3.0 to designate page-table management to a designated sidecore
- VM entry/exit operation eliminated
- Communication: shared memory with polling
- Basic feasibility and understanding of challenges
  - Evaluation challenge due to small core count on current platform



# Benefits of eliminating VM-switches

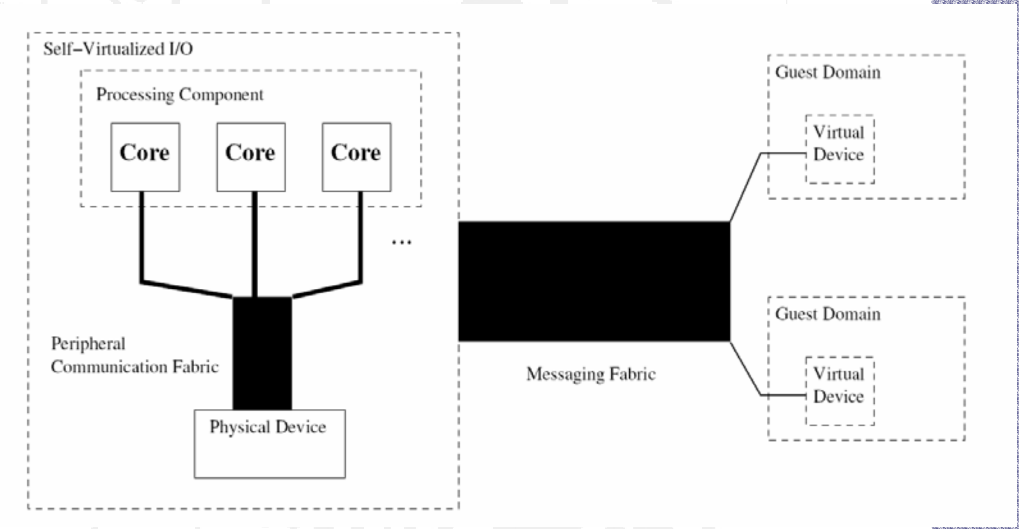




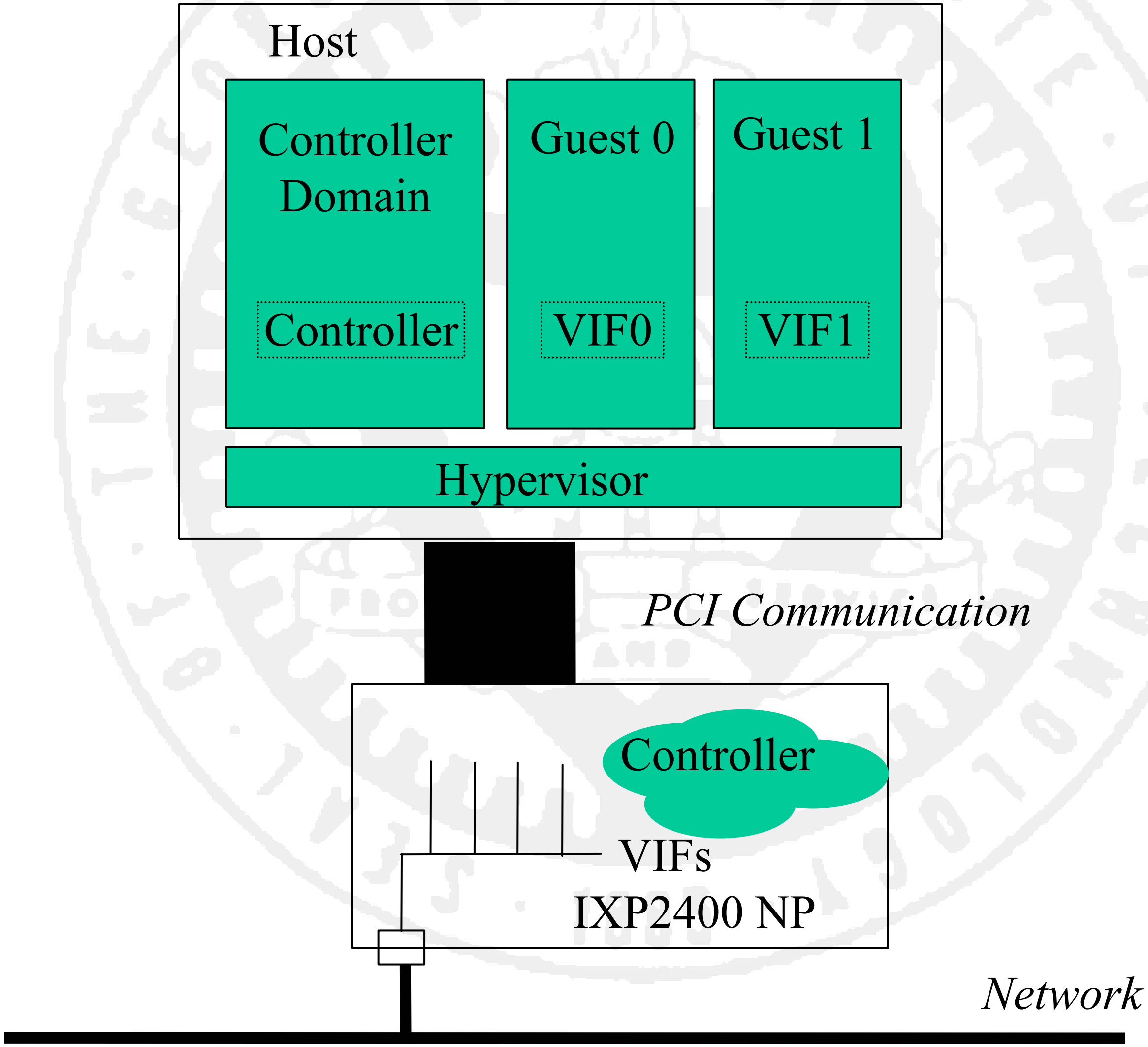
- Up to 41% reduction in page table update latency

# Self-Virtualizing Devices

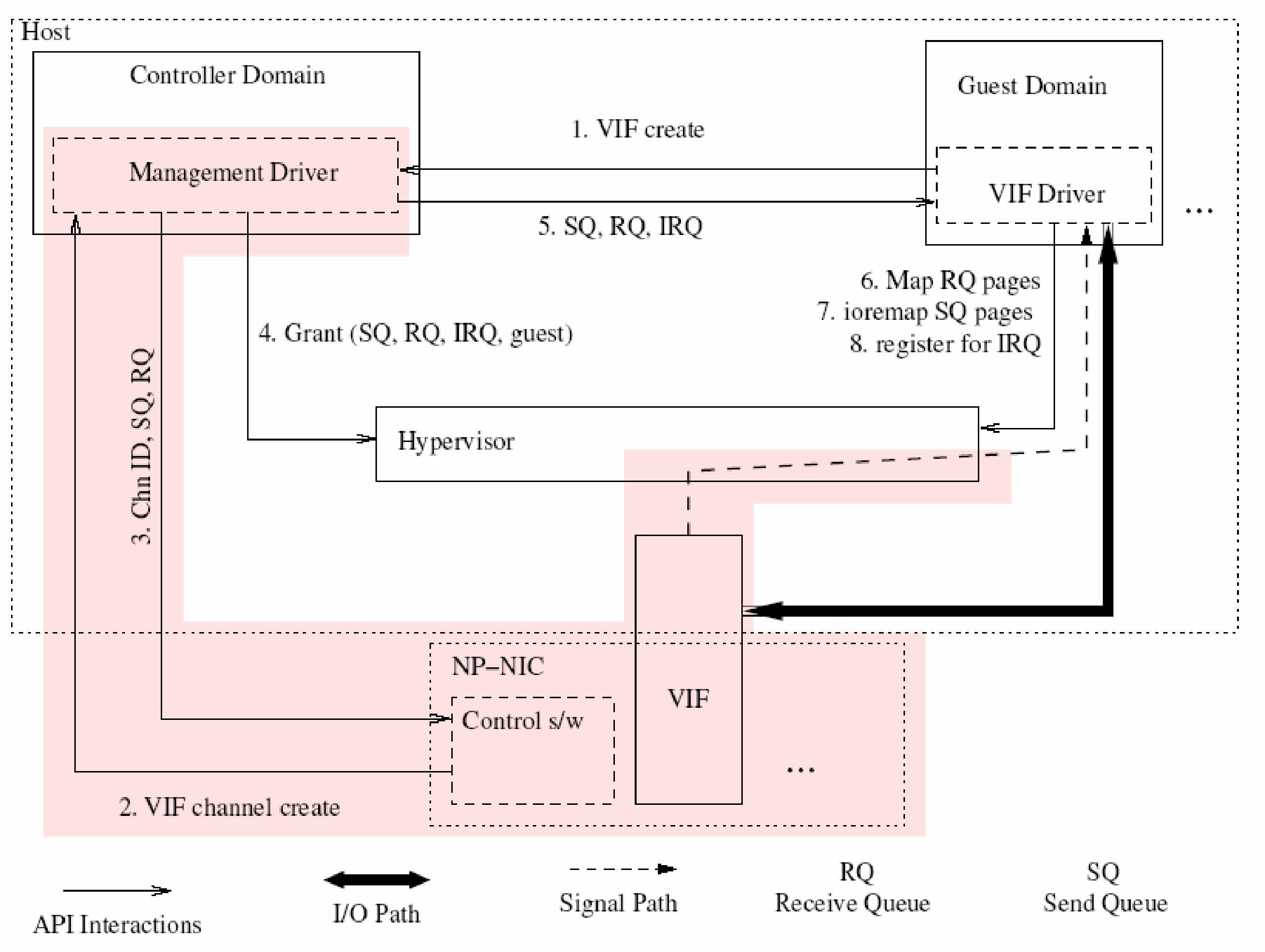
- Device-level virtualization-awareness
  - Safely mux/demux access to device resources
  - Associate self-virtualizatio functionality with device-resident or device-near cores
  - Challenges:
    - device-VM notifications -> interrupts vs. polling
    - IOMMU operations



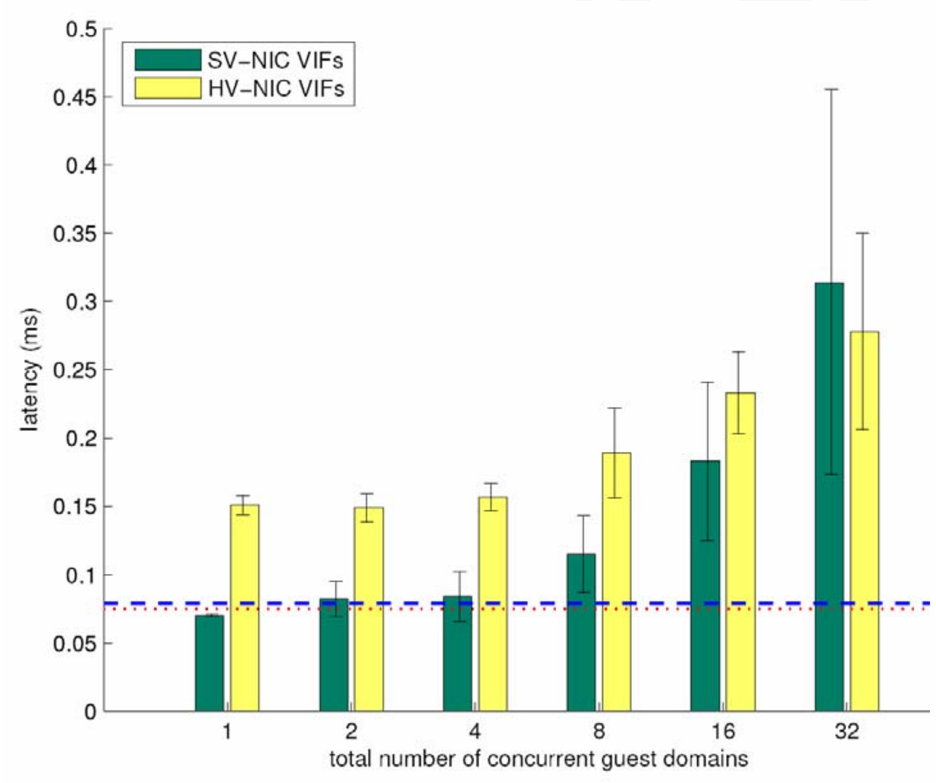
# Self-virtualized NICs using the IXP2400 NP

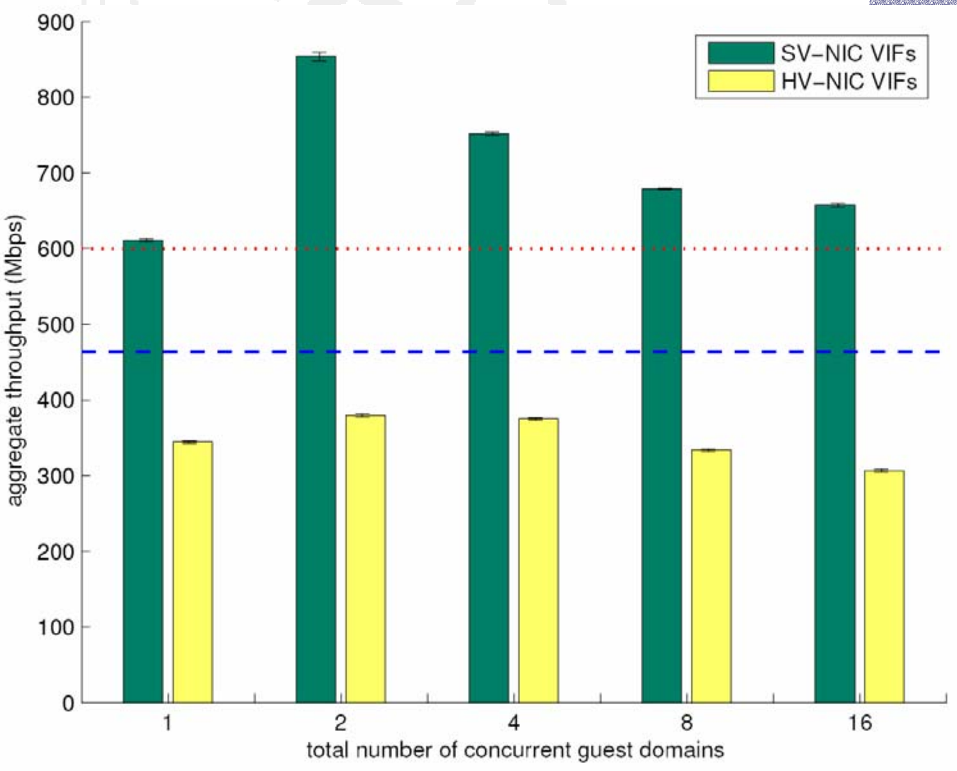


# Interaction in an S-VNIC

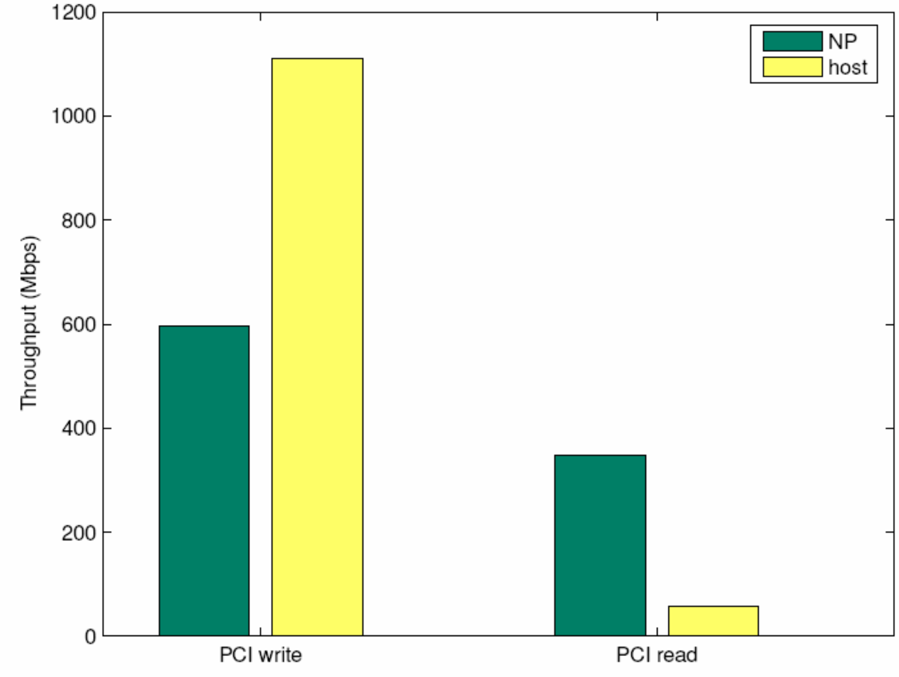


# Improvements in latency and bandwidth

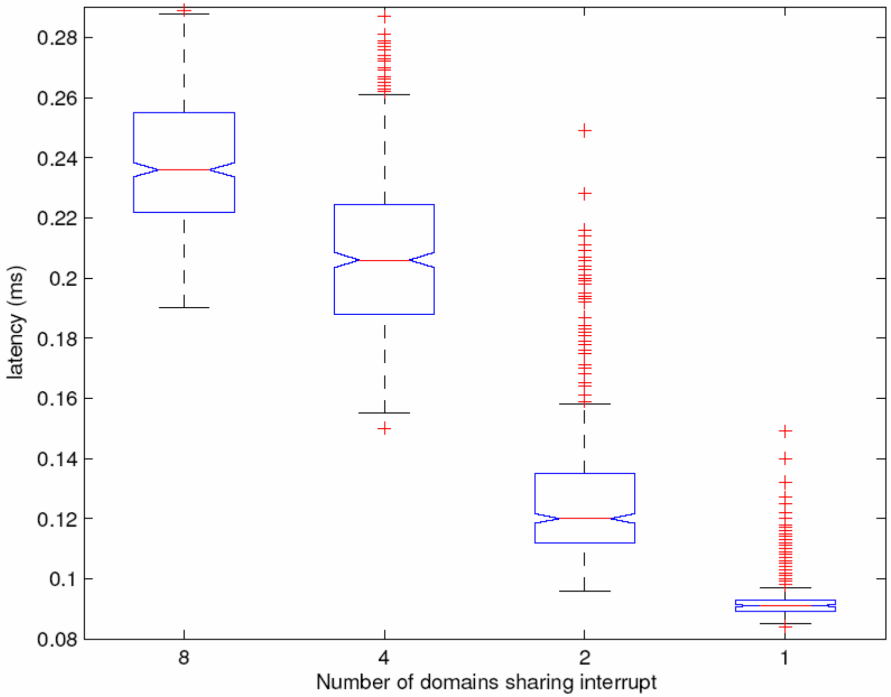




# Architectural limitations

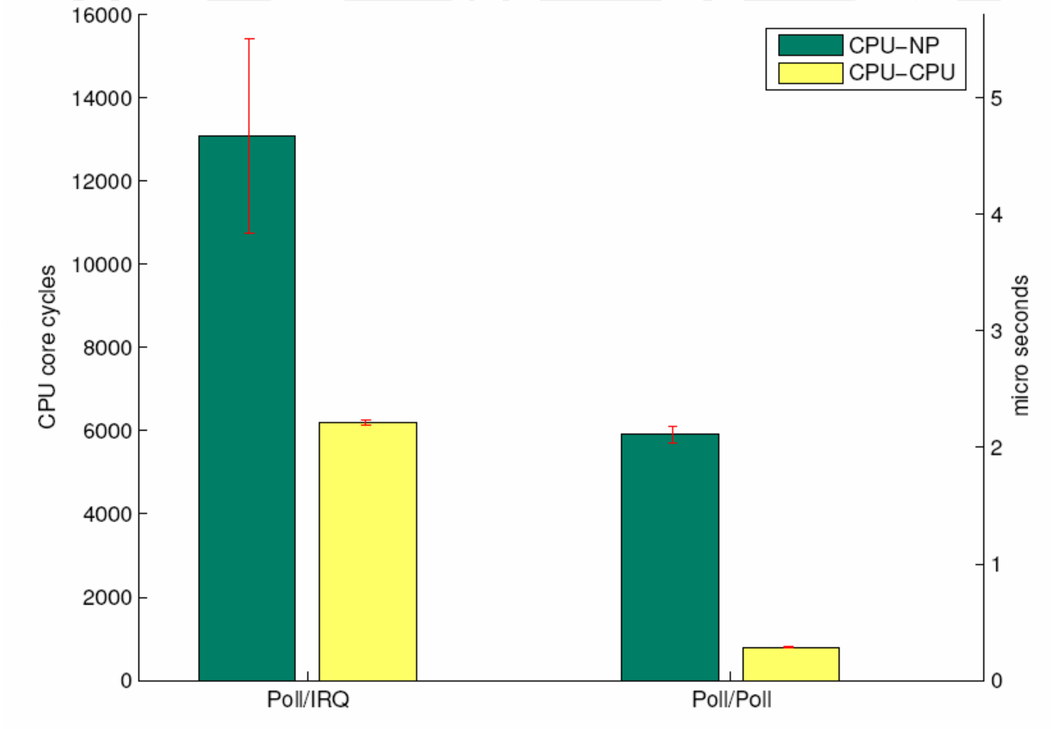


Throughput of the PCI path

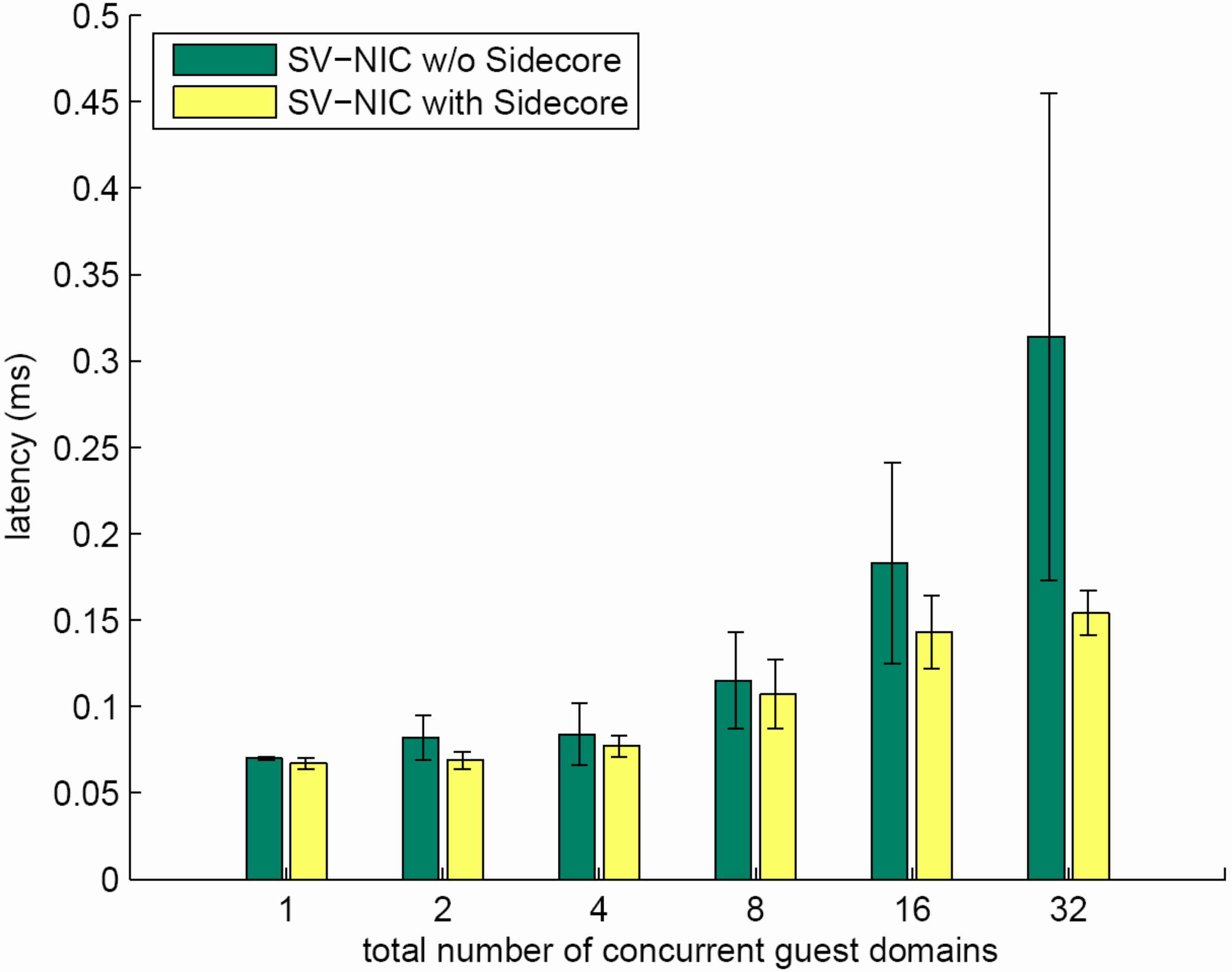


Effects of virtual interrupt sharing

# Insights for future multicore systems



# Virtualized interrupts with Sidecore

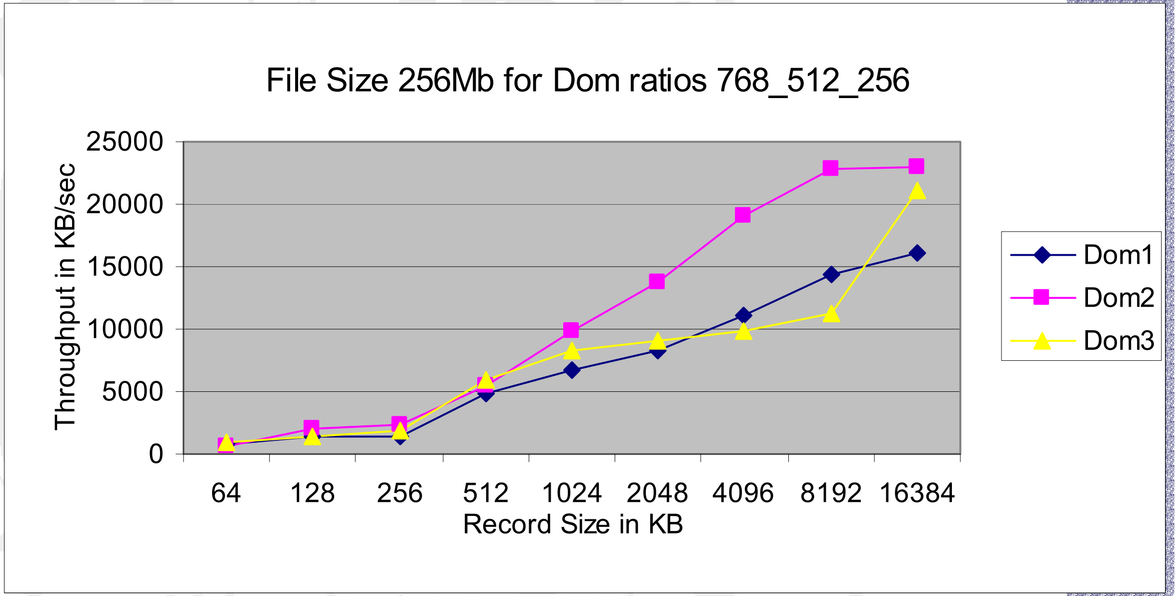


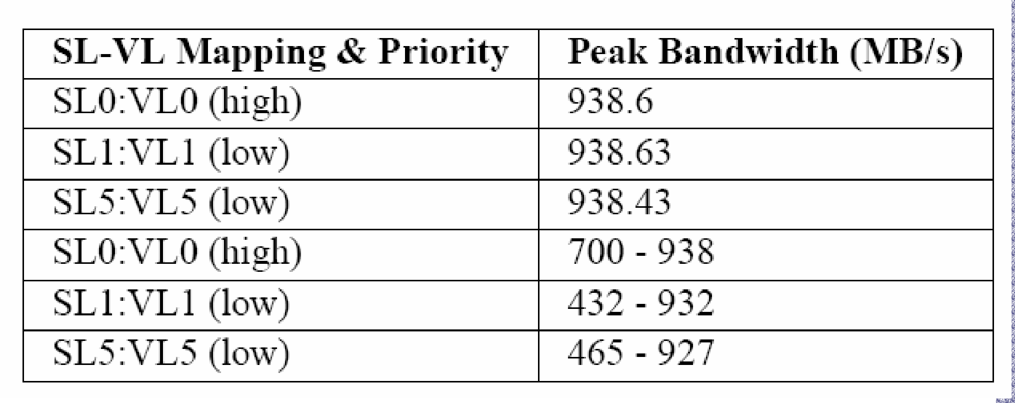
# Importance of S-V I/O

- Performance
  - Hypervisor acceleration/bypass
- End-to-end QoS
- VM Migration & Device Remoting
- “Logical” Devices

# SV-I/O and QoS support

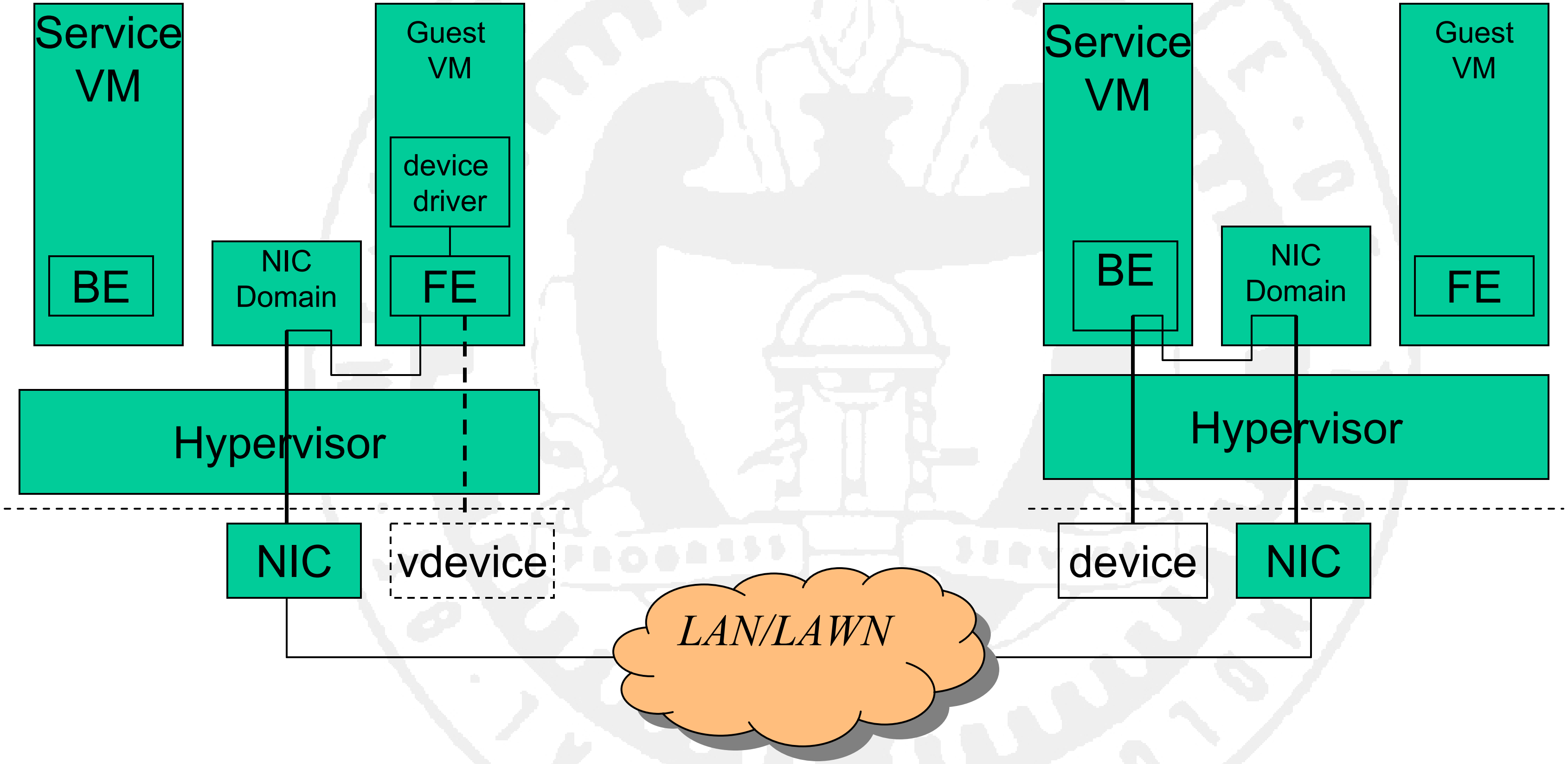
- VM's resource requirements need to include device-level resources
- Need for coordinated hypervisor- and device- level scheduling decisions





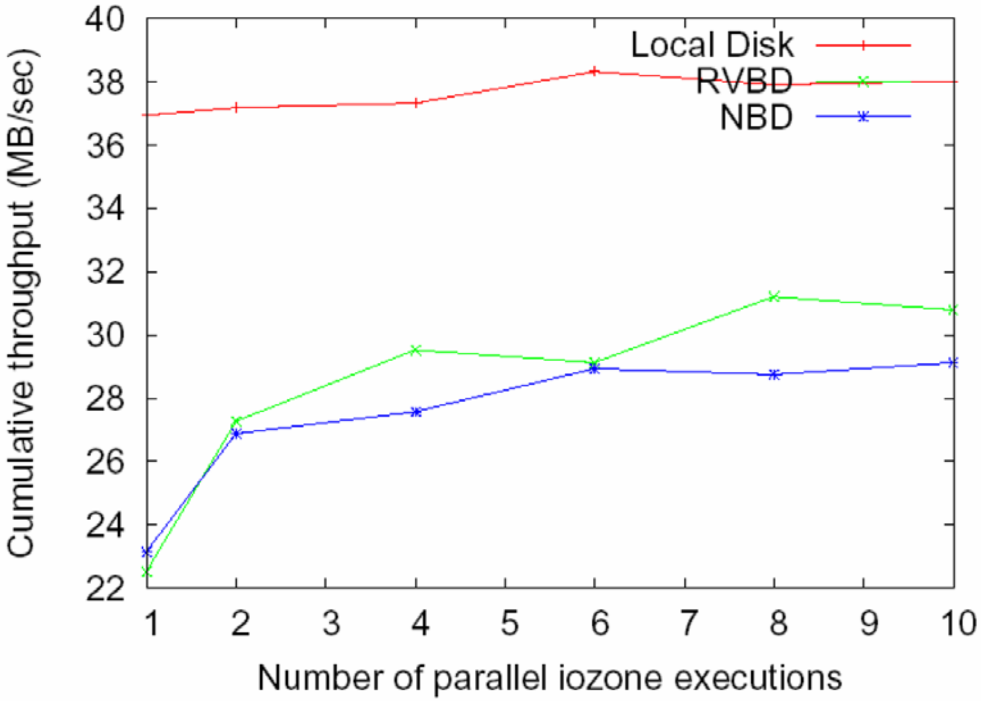
| SL-VL Mapping & Priority | Peak Bandwidth (MB/s) |
|---|---|
| SL0:VL0 (high) | 938.6 |
| SL1:VL1 (low) | 938.63 |
| SL5:VL5 (low) | 938.43 |
| SL0:VL0 (high) | 700 - 938 |
| SL1:VL1 (low) | 432 - 932 |
| SL5:VL5 (low) | 465 - 927 |

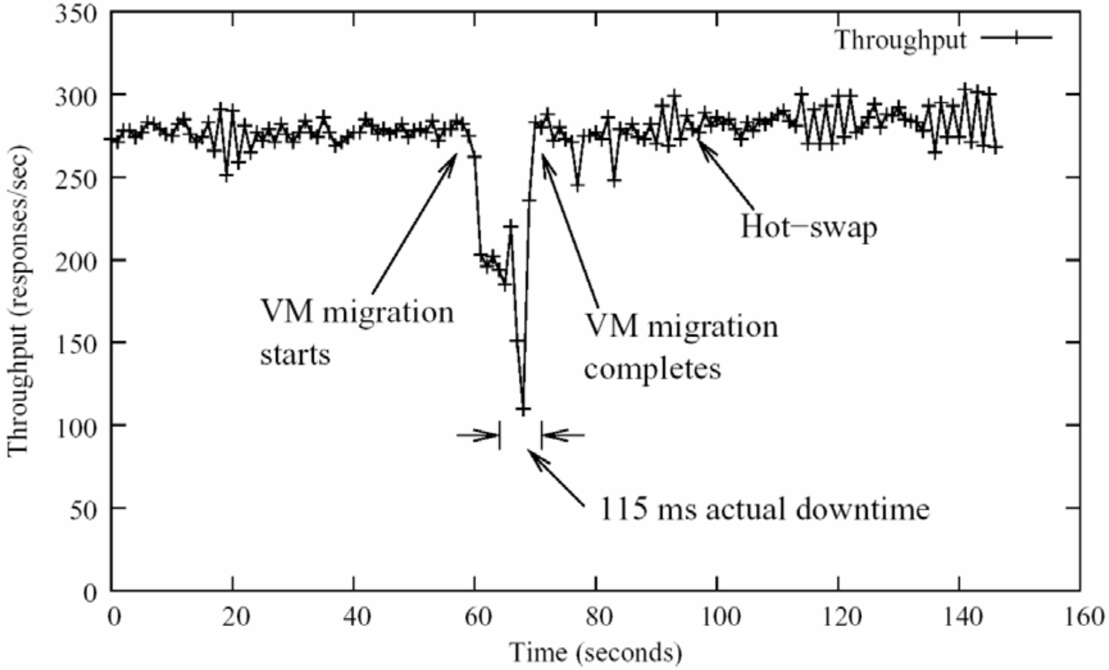
# Remote Device Virtualization



- important for VM migration
- device-centric S-VIO -> data path through BE and NIC domain is pushed on device
- current numbers: ~11% latency reduction

- Device remoting
  - feasibility and utility

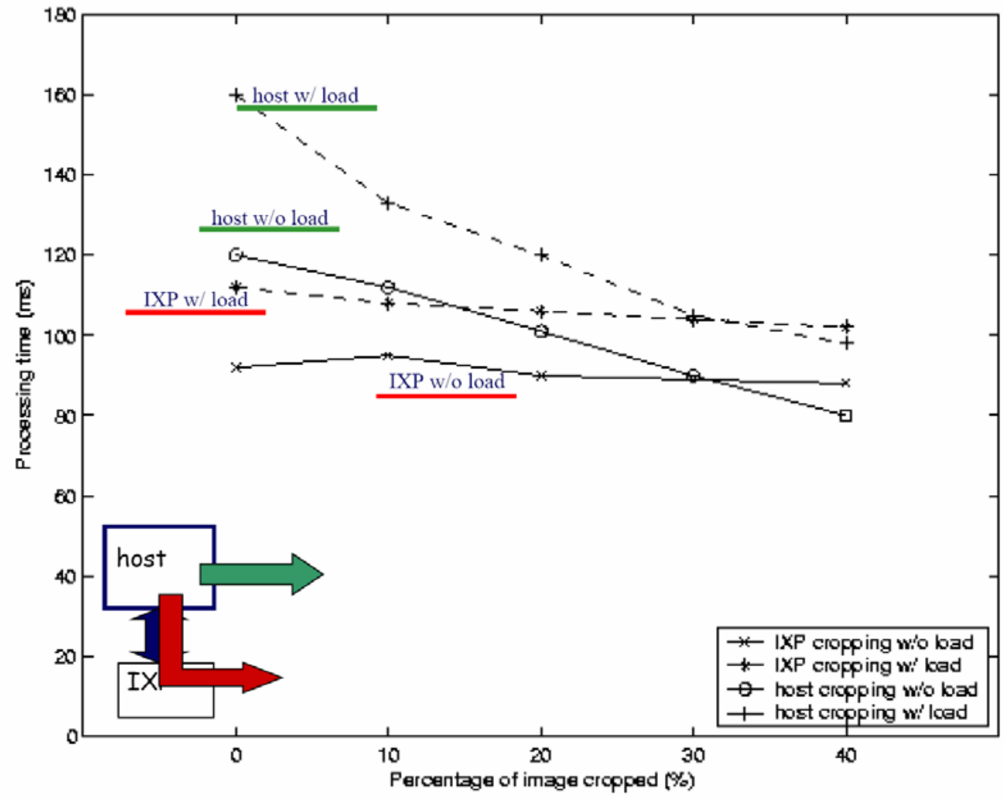




# “Logical” Devices

- Once a device is virtualized, there is no reason for it to be “real”
- May associate codes with S-V IO processing components to implement upper-level functionality
  - Data reformating
    - e.g., file system issues
  - Filtering
    - e.g., security/privacy issues, threshold comparisons…
  - QoS properties

- Case study:
  - image reformatting



# Other issues

- Device sharing among multiple virtualized platforms
  - Cannot afford a single device domain to become hotspot
  - S-VIO implements coordination and management functionality
- Monitoring and QoS
  - Interface and support on device for monitoring and scheduling among virtual device instances (e.g., VIFs)
  - Metadata management on S-V I/O
  - Resource management in virtualized environments:
    - system-wide vs. platform-wide vs. VM- management objectives
    - current primary consideration in our work power
      - portable to other domain
      - power may become relevant factor in HPC too – if the capabilities to control it are present

# Conclusions

- Virtualization may have important contributions in HPC infrastructures
- Technical challenges to attain benefits
  - efficient hypervisor design to eliminate overheads and noise
  - scalable hypervisors for multicore platforms
  - Improved performance on I/O path
  - Better support for device remoting, needed for efficient VM migration
  - Ability to instantiate 'logical' devices and better meet application requirements
  - Finer grain support for ensuring end-to-end QoS
  - Grater scalability for shared virtualized devices
  - Coordinated management (e.g., power?) mechanisms
- Prototype realization of a S-V NIC
  - Gives us insights into access and control APIs
- Proof of concept concept results
  - Efficient & scalable Hypervisor for target platforms