Networked I/O for Virtual Machines

Approaches and Challenges

Muli Ben-Yehuda, Ben-Ami Yassour, Orit Wasserman

{muli,benami,oritw}@il.ibm.com

IBM Haifa Research Lab

Table of Contents

Virtualization

For foundations, see [Popek74]. This talk deals mainly with the open-source hypervisors Xen [Barham03] and KVM [Kivity07].

Network I/O is tough

- High packet rate (1GE $\Rightarrow$ 10GE)
- Data must often be copied on receive
- High bandwidth, high throughput, low latency

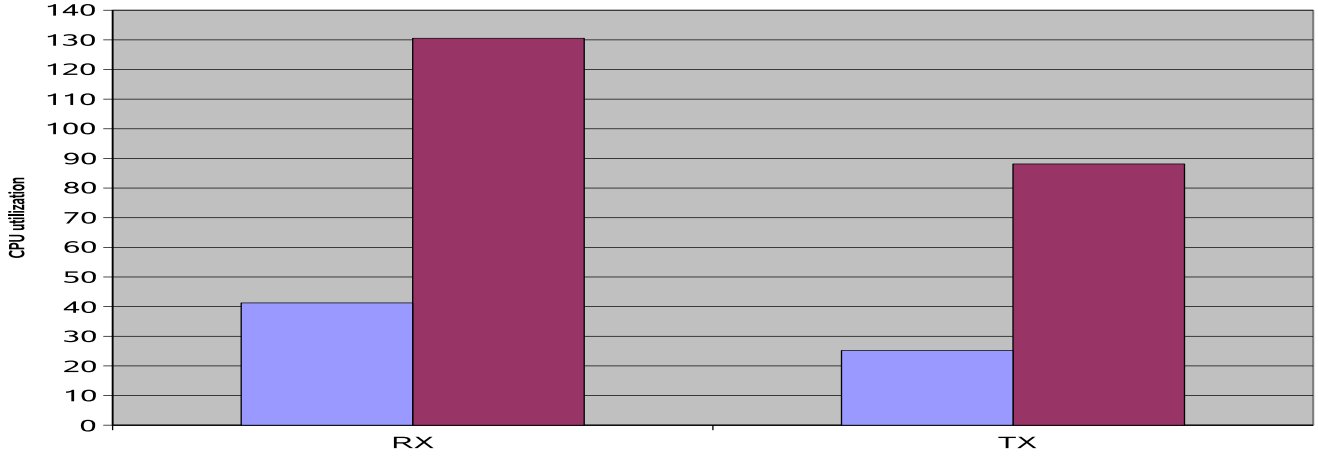


Figure 1: Xen network CPU utilization vs. Linux [Santos07]

Virtual Machine I/O

- Virtual machines use three models for I/O
  - Emulation
  - Para-virtualized drivers
  - Pass-through access

Emulation

- Hypervisor emulates real I/O devices [Sugerman01]
- Virtual machine uses its standard drivers
- Hypervisor traps device accesses (MMIO, PIO)
- Hypervisor emulates interrupts and DMA
- Interface limited to low-level, real device interface!
  - Which is not a good fit for software emulation

$\Longrightarrow$ High compatibility but low performance.

Para-virtualization

- Hypervisor and VM cooperate for more efficient I/O [Barham03]
- Hypervisor specific drivers installed in the VM
- Network device level or higher up the stack

$\Longrightarrow$ Low compatibility but better performance [Santos08].

Pass-through

- Give VM direct access to a hardware device
- Without any software intermediaries between the virtual machine and the device
- Examples:
  - Legacy adapters [Ben-Yehuda06]
  - Self-virtualizing adapters [Liu06], [Willman07]

$\Longrightarrow$ Best performance—but at a price. .

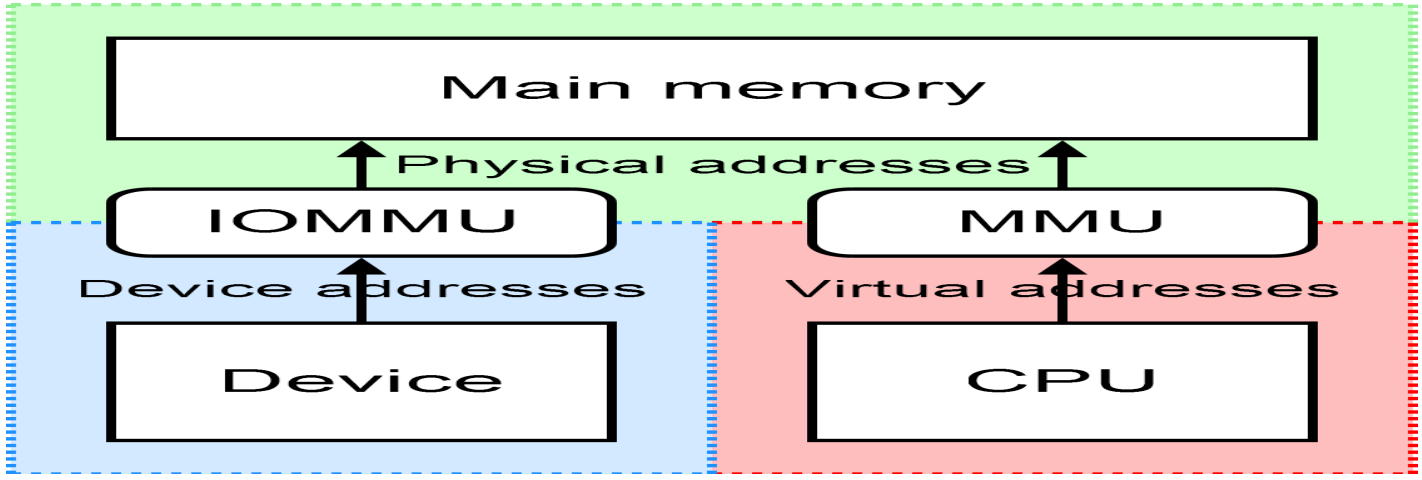
Pass-through security

- Untrusted VM programs a device, without any supervision.
- Device is DMA capable (all modern devices are).
  - Which means the domain can program the device to overwrite any memory location.
- . . . including where the hypervisor lives . . . game over.

Pass-through memory addressing

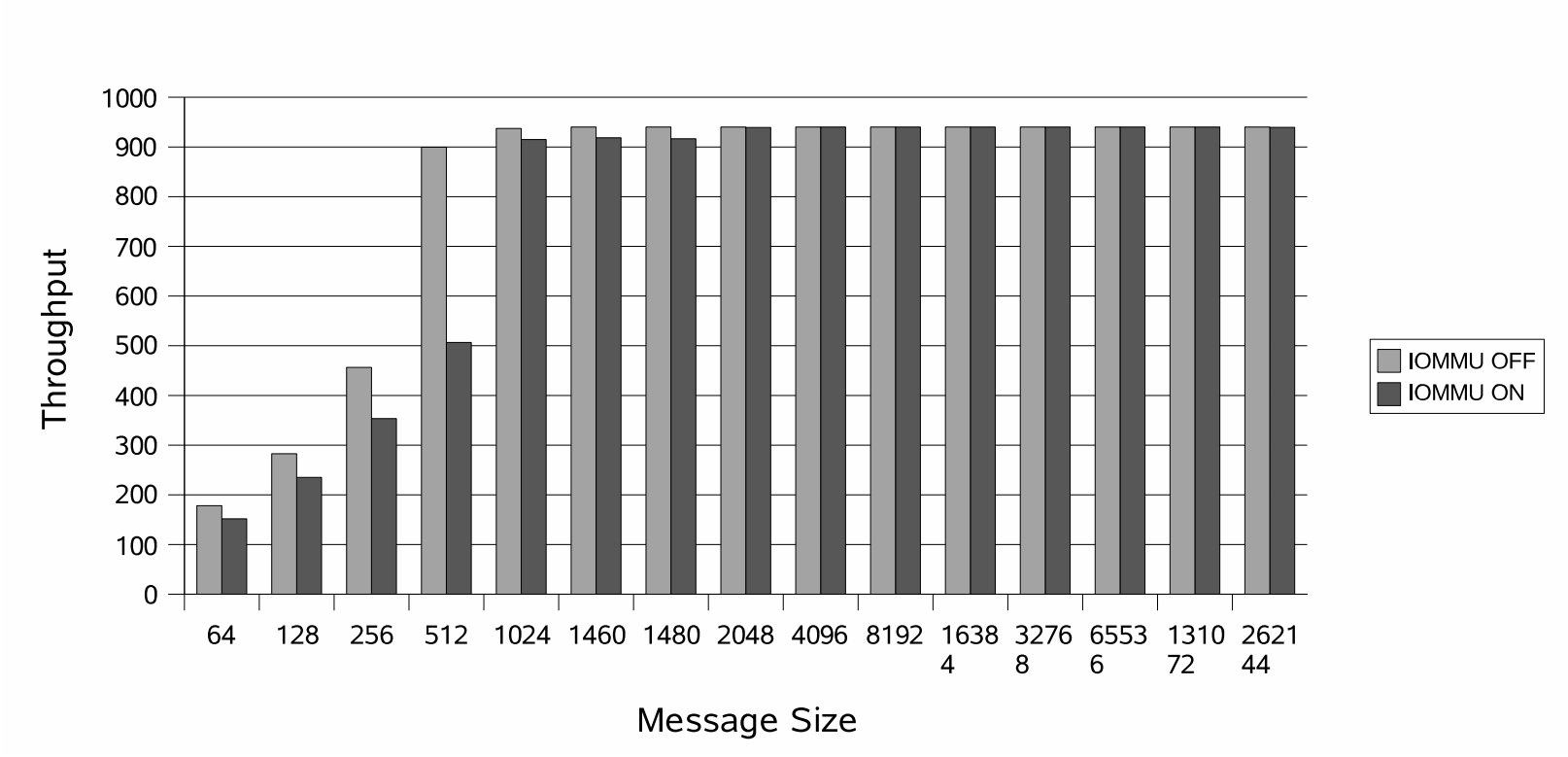
- VM is not aware of host physical memory.
- VM is only aware of its own guest “physical” memory.
- Device DMAs need to end at the right place (host, not guest “physical” memory).
- VM programs device with guest physical addresses $\Rightarrow$ DMAs end up at the wrong place!

IOMMU to the rescue



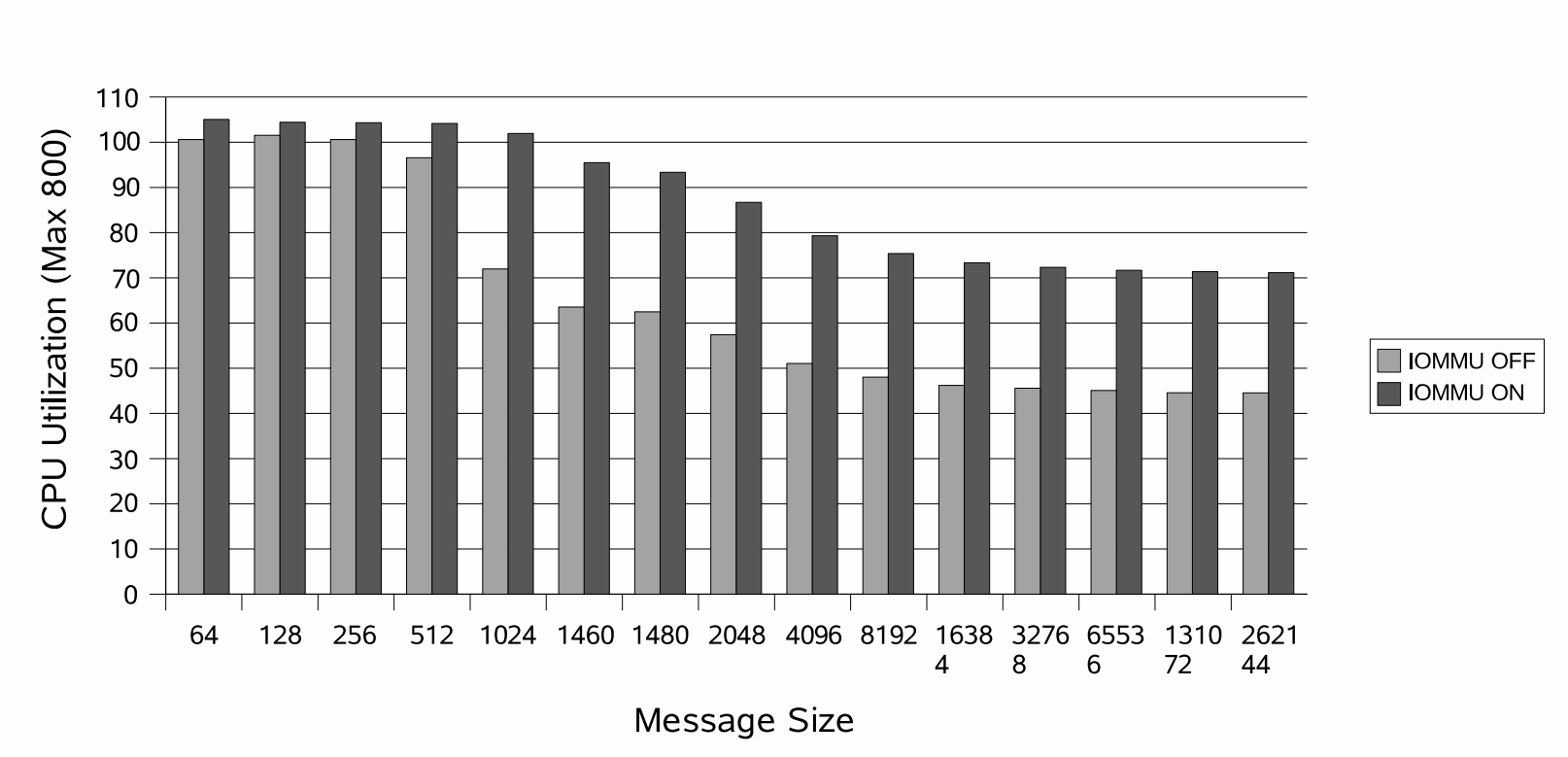
- IOMMU—think MMU for I/O devices—separate address spaces, protection from malicious devices!
- IOMMUs enable pass-through access for para-virtualized and fully-virtualized VMs.
- Intra-VM vs. Inter-VM protection [Willman08]
- But: IOMMUs have costs too [Ben-Yehuda07]

Pass-through network throughput



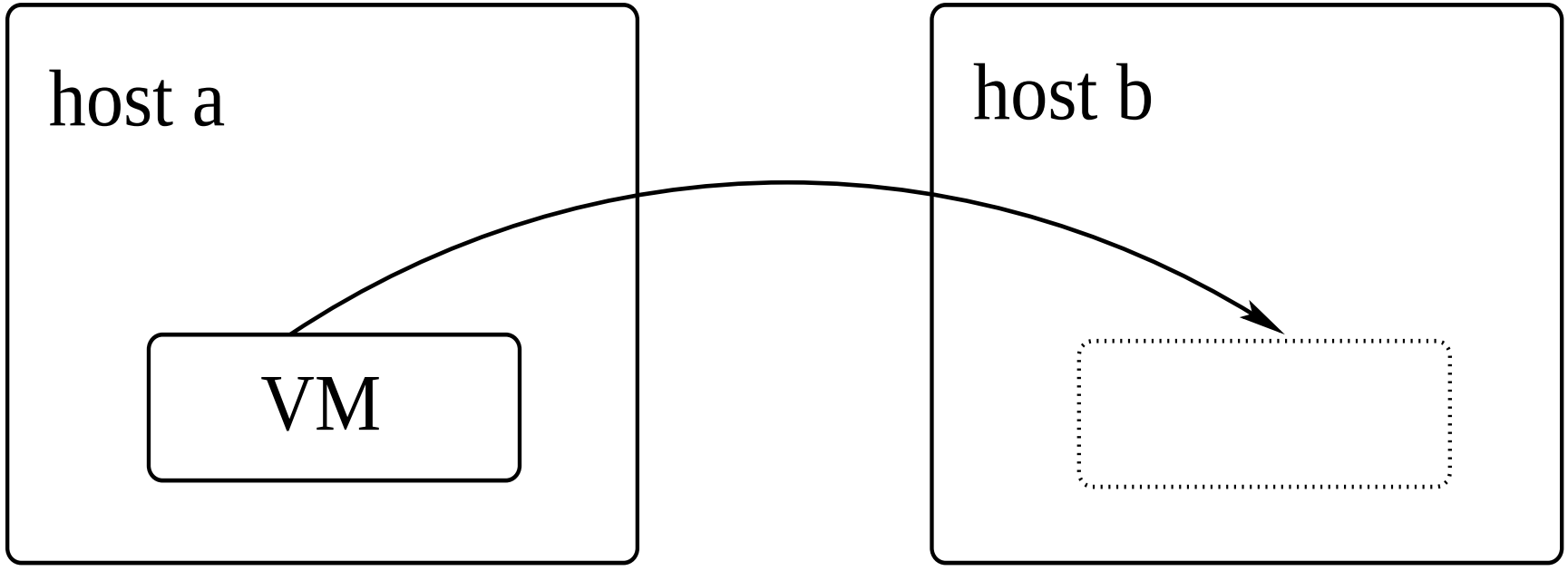
- Msg size < 1024: throughput as much as 45% less.
- Msg size >= 1024: throughput barely affected.

Pass-through network CPU utilization



Pass-through CPU utilization is up to 40%–60% more!

Live VM migration



Tying it all together

- How can we get the same performance as bare metal?
  - Throughput and CPU utilization
  - . . . on 10GbE
- How can we get the performance of bare-metal with the benefits of virtual drivers? (e.g., live migration)
  - A hybrid approach? [Zhai08]
  - Custom-made devices? [Liu07]

Bibliography

- Barham03: “Xen and the Art of Virtualization”, SOSP ’03
- Ben-Yehuda06: “Utilizing IOMMUs for Virtualization in Linux and Xen”, OLS ’06
- Ben-Yehuda07: “The Price of Safety: Evaluating IOMMU Performance”, OLS ’07
- Liu06: “High Performance VMM-Bypass I/O in Virtual Machines”, USENIX ’06
- Liu07: “Nomad: migrating OS-bypass networks in virtual machines”, VEE ’07
- Kivity07: “kvm: The Kernel-Based Virtual Machine for Linux”, OLS ’07

Bibliography cont.

- Popek74: “Formal Requirements for Virtualizable Third Generation Architectures”, CACM 17(7), ’74
- Santos08: “Bridging the Gap between SW & HW Techniques for I/O Virtualization”, USENIX ’08
- Sugerman01: “Virtualizing I/O Devices on VMware Workstation’s Hosted Virtual Machine Monitor”, USENIX ’01
- Willman07: “Concurrent Direct Network Access for Virtual Machine Monitors”, HPCA ’07
- Willman08: “Protection Strategies for Direct Access to Virtualized I/O Devices”, USENIX ’08
- Zhai08: “Live Migration With Pass-Through Device for Linux VM”, OLS ’08