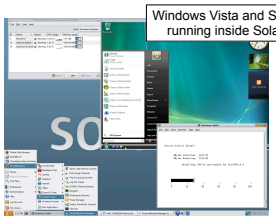
Characterizing & Optimizing Virtualization Overhead for Portable High-Performance Networking

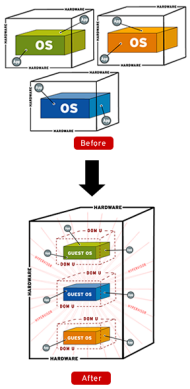
ACM Best Undergraduate Student Poster Award

Gabriel E. Martinez (mystal@vt.edu), Mark K. Gardner (mkg@vt.edu), and Wu-chun Feng (feng@cs.vt.edu)

Motivation

- Why Platform Virtualization?
  - Server consolidation
  - OS version mismatch in HPC
  - Completely sandboxed testing
  - Noise isolation for improved performance
- Why **NOT** Platform Virtualization?
  - Significant performance impact … relative to the network and I/O.





Hypothesis

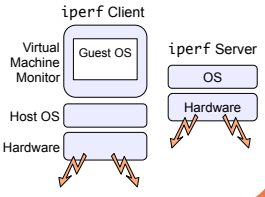
Network performance suffers due to high CPU utilization within virtual machines (VMs).

Approach

Characterize network performance on VirtualBox, VMware Server, and Xen *relative to* raw performance on host OS.

Experimental Set-Up

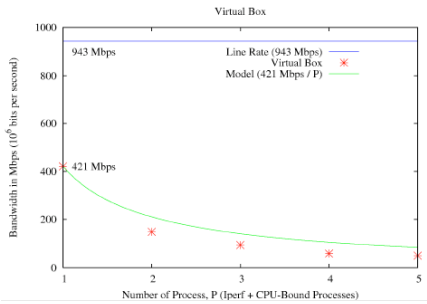
- Use `iperf` & ANL Microbenchmark Suite to measure network performance
- Run multiple CPU-bound processes to reduce cycles available to `iperf`
- Test in Ubuntu 8.04 on quad-core AMD Opterons that do *not* have hardware virtualization support



VirtualBox

Enhanced Version of QEMU

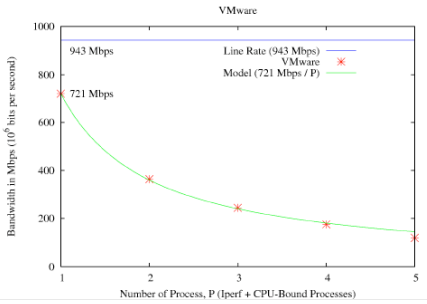
- Guest OSes run in ring-1
  - *Faults into ring-0 which kernel module handles*
  - *Dynamic re-compilation limits number of faults*
  - *Emulation takes place in rare cases*
- Max: 421 Mb/s over 1-Gb/s Ethernet connection



Inside a VM, `iperf` throughput drops off with increased number of CPU-bound processes

VMware Server

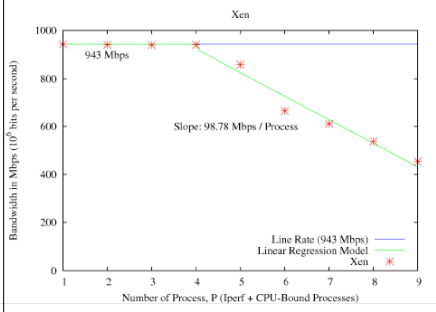
- Guest OS *user* code runs *natively*
  … but guest OS *kernel* code *cannot*
  - *Converts privileged kernel code to unprivileged utilizing binary translation*
- Max: 721 Mb/s over 1-Gb/s Ethernet connection



Like VirtualBox, `iperf` throughput depends on the # of CPU-bound processes running in the VM

Xen

- A *hypervisor* that runs on bare hardware
- Utilizes paravirtualization for guest OSes
  - *Effectively removes second network stack*
  - *Requires guests to be VM-aware*
- Max: 943 Mb/s on 1-Gb/s Ethernet connection

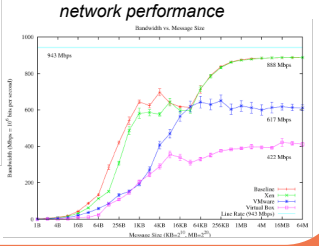


`iperf` performance is high and only drops with 5 or more processes. Why? Xen effectively uses each available AMD core for VMs

Insight & Conclusions

- Performance and # of other CPU-bound processes *inversely related* for VirtualBox and VMware Server (see graphs above)
  - Inside of a VM a normally I/O-bound program … `iperf` is instead CPU-bound

- *The above relationship explains the significant differences in network performance*



Future Work

- Further explore the impact of optimizing VMs via paravirtualization.
  - Started with Xen, now look into VMware ESX and Sun xVM.
- Continue to benchmark virtualized hardware to better understand virtualization implementations, e.g.,
  - Preliminary testing with virtualized Intel Gigabit NIC:
    - VirtualBox: 50%+ increase in bandwidth over the default driver
    - VMware: 30%+ decrease in bandwidth over the default driver
- Understand and characterize *why Xen performance decays linearly as # of VMs exceeds # of cores.*
- Conduct detailed analysis of *why* multiple processors help Xen but *not* VMware Server with respect to bandwidth (BW)
  - VMware: Single CPU BW: 721 Mb/s → Dual CPU BW: 323 Mb/s
- Extend the study to understand the impact of hardware virtualization.
- Incorporate findings into the broader project of providing virtual machines for pedagogy and related projects in bioinformatics & green computing.

VMs for Pedagogy
http://service.cs.vt.edu/