Performance Evaluation of Virtual Machines

Ryan Stern
stern@cs.colostate.edu

CS 655: Advanced Topics in Distributed Systems
Department of Computer Science
Colorado State University

Outline

- Quick KVM overview

- Looking at results from two papers:
  - Performance Issues In Clouds
  - Virtualization of Linux Servers

KVM

- kvm: The Kernel-based Virtual Machine

- Utilizes x86 virtualization extensions
  - Adding a hypervisor to Linux
  - Adds a guest operating mode, allows trapping privileged instructions

- Virtual machines are managed via a /dev/kvm node
  - Create the virtual machine
  - Allocate memory
  - Read / Write virtual registers
  - Injecting interrupts
  - Running a virtual CPU
KVM: Scheduling CPU

- Execute guest code until
  - Encounters I/O instruction
  - An external event occurs

- Guest mode is entered
  - Until a fault or interrupt

KVM: Memory

- Provided by x86
  - Page table (a radix tree) for virtual to physical page translation
    - Rooted in a hardware register
    - Instructions change which table is pointed to
  - Translation lookaside buffer (TLB)

- Hardware support provides hooks, but does not fully virtualize
  - GuestVirtual -> guestPhysical, not guest guestPhysical -> hostPhysical
  - Keeping shadow page table synchronized is a challenge

- Trade code complexity for context switch performance
  - Write protecting page tables, trapping writes
  - Emulating access to the pages

Performance Issues

- Difficult to express Quality of Service
  - Performance, dependability, capacity, availability, security, ...

- Issues can arise with:
  - Virtualization
  - Management of resources
  - Scalability
  - Storage
Infrastructure Management Systems

- Virtual infrastructure manager
  - Scheduling and monitoring components
  - Assigns VMs to physical machines
  - Orchestrates transfer of VMs

- Will look at two management systems
  - Nimbus
  - OpenNebula

Nimbus

- Reuses many technologies used by the grid computing community
  - Use of grid resources and existing resource schedulers
  - Allows organizations using grids to more easily adapt to cloud settings

- Modular design

- Libvirt is used to support multiple hypervisors
  - Under a single interface

OpenNebula

- Allows the manipulation of local virtual infrastructure
  - Creation and management of private clouds

- Allows management of remote VMs
  - Such as those in public clouds
  - Exposes functionality via public cloud provider’s management interfaces

- Able to manage multiple workloads
  - Enabling hybrid clouds (public and private)
  - Plugins for accessing provider-specific features

- Black box monitoring of systems
  - Uses Libvirt to access multiple virtualization technologies using a single interface

Nimbus vs. OpenNebula - Similarities

- Use of Libvirt for supporting hypervisors
  - Supports XEN, KVM, VMware
  - Prevents the need to worry about hypervisor updates

- Use of SCP for file transfer
  - Nimbus also offers GSIFTP

- Both have a modular design
  - Support for creating extensions
  - Support for utilizing external resources
Nimbus vs. OpenNebula - Differences

- Different use cases
- Neutrality
  - Nimbus: Amazon, limited support for others
  - OpenNebula: Any, with use of plugins
- Contextualization
  - Nimbus: Central broker service coordinates VM instances, even during
    OpenNebula: Context set at boot time
- Image propagation
  - Nimbus: Staging in/out using image repository
  - OpenNebula: Persist images on a remote host once staged in

Nimbus vs. OpenNebula - Overview

<table>
<thead>
<tr>
<th>Feature</th>
<th>Nimbus</th>
<th>OpenNebula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for using other providers</td>
<td>Limited</td>
<td>Support for using other providers via Cloud Storage</td>
</tr>
<tr>
<td>Contextualization Method</td>
<td>Central broker service coordinates VMs</td>
<td>Contextualizes over network via context broker service</td>
</tr>
<tr>
<td>Security (client)</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td>Image propagation</td>
<td>Staging in/out using image repository</td>
<td>Persist images on a remote host once staged in</td>
</tr>
<tr>
<td>Networking support</td>
<td>Support for installing/deleting VMs</td>
<td>Support for installing/deleting VMs</td>
</tr>
<tr>
<td>VMOS support</td>
<td>Support for popular hypervisors like Xen, KVM, VMware</td>
<td>Support for popular hypervisors like Xen, KVM, VMware</td>
</tr>
</tbody>
</table>

Benchmarks

- Hardware:
  - 4 Dell commodity servers
  - 4 core X3360 Intel Xeon
  - 4GB RAM, DDR2, 800MHz
  - 250GB SATA HDD, 16MB cache, 7200RPM

- Benchmarks:
  - Image propagation
  - IOZone
  - Bonnie++
**Benchmark Results – Image Propagation**

- **XEN**
  - Historically concentrated on Paravirtualization
  - Now supports multiple virtualization techniques
  - Requires modified device drivers and modern hardware

- **KVM**
  - Historically required hardware support
  - Utilizes latest hardware-assisted virtualization research
  - Being part of the Kernel, the introduction of bugs is less likely

- Both have full virtualization support
  - Via QEMU: Emulate I/O devices and CPU architectures
XEN vs. KVM - I/O Devices

- Have different PV architectures for I/O
- KVM uses device drivers
  - Integrated with QEMU
  - Abstraction with Virtio
  - Based on XEN
- XEN uses hypercalls
- Emulation of PCI devices
Background

- Solutions differ in the way the virtualization layer is implemented
- Classic virtualization
  - Trap instructions that modify physical machine state
  - In x86, not all such instructions are privileged
- Full virtualization
  - Emulation of hardware
  - QEMU is an example
- Paravirtualization
  - More direct access provided to hardware
  - Requires modifications to guest operating systems
- Operating system level virtualization
  - Single kernel, lower overhead

Benchmark: Kernel Compilation

- CPU and I/O intensive
- Reads and writes many small files
  - Stresses file system
- Results:
  - OS and PV perform best
Benchmark: Compression

- CPU intensive
- Requires memory

Benchmark: Dbench

- A file system benchmark
- Simulates the load of a file server
- Results:
  - Most perform under 30% of Linux alone
  - Xen shows better performance in other studies

Benchmark: dd

- Commands:
  - `dd if=/opt/iso/ubuntu-6.06.1-server-i386.iso of=/var/tmp/out.iso`
  - `dd if=/dev/zero of=/dev/null count=117187560`

Benchmark: Netperf

- Evaluates data exchange
- Results:
  - QEMU performs poorly
Benchmark: Rsync over network

- **Rsync (kernel tree)**
- **Rsync (ISO file)**

![Graphs comparing Rsync performance](image)

Figure 8: Evaluating data transfer in the network with Rsync

Benchmark: Sysbench

- 10000 MySQL database transactions

Results:
- VServer, Xen perform best

![Graphs comparing Sysbench performance](image)

Figure 9: Using the Sysbench OLT test benchmark to evaluate the performance of a database server

Testing Scalability

- **Sysbench (OLTP) at scale**
  - Aggregate throughput
  - Average throughput

![Graphs showing scalability](image)

Figure 10a: Evaluating the capacity of the virtualization solutions to manage and share the physical available resources with Sysbench. The left side of the picture shows the aggregate throughput, and the right side shows the average throughput per VM.

Figure 10b: Evaluating the scalability of VServer, Xen, and Linux running different settings.

Testing Scalability

- **Sysbench (OLTP test) at scale: using different settings**

![Graphs showing scalability](image)

Figure 11: Repeating the scalability evaluation with Linux, VServer, Xen, and Linux using different settings.
Problem With Experiments

- Most do not show how multiple concurrent VMs influence the benchmark results

Questions?