Agenda

Introduction to KVM

KVM in SUSE Linux Enterprise Server 11 SP2

Examples of KVM Enterprise Solutions

Using KVM: Virtualization tooling

Best Practices and Guidelines

Questions
Introduction to KVM
What is KVM?

- A virtualization technology?
- A kernel module?
- A Linux package?
- A kernel module + userspace program + tools?
- A SUSE® Linux Enterprise Server 11 Supported Virtualization Solution

All the above!

- KVM stands for “Kernel-based Virtual Machine,” meaning the infrastructure for creating virtual machines is included with the Linux kernel
KVM History

- Development began in 2006, first released 2007
  - Currently an active, healthy open source project
  - Development community has many contributors, including SUSE®
  - Merged with Linux kernel and QEMU projects
- Included in SUSE Linux Enterprise Server 11 since March, 2009
  - Tech Preview in GA
  - Supported since SP1
KVM Virtualization

• Uses AMD-V and Intel VT-x Hardware Virtualization
  - Found in commodity hardware as well as high-end servers

• Implements Full Virtualization
  - Guests run unmodified
  - Para-virtual drivers available, device pass through possible

• Leverages Linux to provide a superb virtualization platform
  - Virtualization hardware control: generic and vendor KVM kernel modules (kvm.ko, kvm_amd.ko, kvm_intel.ko)
  - The Linux kernel acts as a hypervisor
  - A KVM accelerated QEMU userspace process runs the guest, which is just another userspace process to Linux
KVM Architecture
Adds “Guest Mode” to Traditional Kernel and User Modes

Source: “Virtualization with KVM” training, B1 Systems GmbH
Supported Hardware

- CPU: AMD-V and Intel VT-x, 64 bit mode
- VT-d / IOMMU
- SR-IOV
- And all the additional virtualization features added as each new processor version come out: EPT, NPT, VPID, ASID, Restricted Guest, PCID/INVPCID, ...
QEMU

• QEMU project + KVM acceleration => qemu-kvm userspace program

• Emulates a PC style hardware platform, accommodates para-virtual devices (Virtio, clock), accelerators (KVM, vhost-net), and pass-through capabilities (virtFS, PCI and usb device assignment)

• Provisions host cpu, memory, storage, and networking resources to the guest securely and efficiently
KVM Virtual Machine Base Features

- “Modern PC style” machine architecture
- SMP and NUMA architecture
- Various cpu types and features selectable
- ISA, PCI, USB buses (including PCI hotplug)
- IDE, AHCI, SCSI and floppy storage interfaces and devices
- Common network adapters, sound cards
- Standard display, keyboard, mouse
- System BIOS, PXE BIOS's. Boot control
- Paravirtual devices: blk, net, clock, memory balloon
KVM Virtualization Features

- Supports latest hardware virtualization technologies
- Guest life-cycle controls
  - Start, stop, reboot, pause/resume, suspend/restore
  - Live migration
  - Snapshots, delta storage images
- Co-exists with other virtualization technologies
- PCI and USB host device pass through (incl. SR-IOV)
- VirtFS: filesystem “pass through”
- CPU, memory and disk over-commit
- Direct kernel boot option
- Emulated (TCG) CPU execution mode available
KVM Virtualization Features (continued)

- Transparent Huge Page (THP) optimized
- Kernel Samepage Merging (KSM) supported
- Non-root user support
- User-mode networking stack (DNS, DHCP, TFTP, BOOTP, SMB)
- Tap device, bridged, and vhost-net networking
- Guest details provided on the qemu-kvm command line
- Nested virtualization
- Guest Agent
- Built-in GDB server for guest debugging
- Various storage formats: raw, qcow2, qed, vmdk
<table>
<thead>
<tr>
<th>Name</th>
<th>Compression</th>
<th>Snapshot</th>
<th>Encryption</th>
<th>Deltas</th>
</tr>
</thead>
<tbody>
<tr>
<td>raw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qcow2</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>vmdk</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>
KVM Support
KVM – Supported Guest Systems

- Linux
- Windows
- Solaris, OpenSolaris
- BSD Unix
KVM Guests Supported by SUSE® (I)

Linux – both 32 and 64 bit

- SUSE Linux Enterprise Server 11 SP2 (fully supported)
- SUSE Linux Enterprise Server 11 SP1 (fully supported)
- SUSE Linux Enterprise Server 10 SP4 (fully supported)
- SUSE Linux Enterprise Desktop 11 SP2 (tech. preview)
- Red Hat Enterprise Linux 4 (limited support)
- Red Hat Enterprise Linux 5 (limited support)
- Red Hat Enterprise Linux 6 (limited support)
KVM Guests Supported by SUSE® (II)

Microsoft Windows – both 32 and 64 bit
(fully supported from SUSE Linux Enterprise Server 11 SP2 on)

- Microsoft Windows 2003 SP2+
- Microsoft Windows 2008 SP2+ or R2+
- Microsoft Windows 2012
- Microsoft Windows XP SP3+ (limited support)
- Microsoft Windows Vista SP2+ (limited support)
- Microsoft Windows 7 SP1+ (limited support)
- Microsoft Windows 8 (limited support)
KVM Limits Supported by SUSE®

- Host RAM and CPU limits are the same with or without KVM modules loaded
- Guest RAM size: 512 GB
- Virtual CPUs per guest: 64
- NICs per guest: 8
- Block devices per guest: 4 emulated, 20 para-virtual (virtio-blk)
- Maximum number of guests: total vCPUs in all guests \( \leq 8 \) times total CPU cores in host
Xen and KVM: A Comparison

**Xen**
- VMM implementation of its own; hypervisor
- Kernel (dom0) used as I/O dispatcher and management domain
- Partly in upstream kernel
- Supports fully virtualized and paravirtualized VMs
- Uses older qemu for device model

**KVM**
- Kernel module
- Uses kernel as VMM
- In upstream kernel
- Supports fully virtualized VMs only
- Uses current qemu for device model
SUSE® Linux Enterprise Server 11 SP2

• SUSE Linux Enterprise Server 11 SP2 ships with both KVM and Xen virtualization solutions

• KVM and Xen considered to be on par, differentiated mainly by their different approaches to virtualization

• Toolset shipped in SUSE Linux Enterprise Server 11 SP2 supports both Xen and KVM

• Linux kernel 3.0, qemu-kvm version: 0.15.1

• SUSE Linux Enterprise Virtual Machine Driver Pack 2.0 supports both Xen and KVM
Examples of KVM Enterprise Solutions
Examples of KVM Enterprise Solutions

• SUSE® Studio Online
  - 300,000 users
  - 140,000 builds/year
  - Uses KVM for runners – build and testdrive

• openSUSE Build Service
  - 30,000 users, 30,000 repos
  - 150,000 packages
  - 27,000,000 builds/year

• SUSE® Cloud / OpenStack
Using KVM: Virtualization Tooling
Using KVM

• We recommend using libvirt and libvirt tools to access KVM
  - Includes: vm-install, virt-manager, virt-viewer, virsh commands
  - Adds additional security, configurability, compatibility, etc.

• Using qemu-kvm command-line also supported – documentation identifies supported parameters

• Qemu-img image management tool provided

• The SUSE® Virtual Machine Driver Pack (VMDP) provides Virtio drivers for Windows guests
libvirt

• Virtualization library for managing one host
  - Domains, networks, storage, host devices, ...

• Share application stack between hypervisors
  - Xen, qemu/kvm, LXC, VMware, VirtualBox, ...

• Long-term API/ABI stability and compatibility

• Integration with other SUSE® Linux Enterprise components
  - AppArmor, SELinux, CGroups, Linux Audit Framework, PolicyKit, ...

• libvirt.org
libvirt Architecture

virsh
vm-install
virt-manager
...

libvirt client

libvirtd
qemu
LXC
...
storage
network
legacy xen
ESX
libvirt – Host Management

• Storage Pools
  - Dedicated device, partition, directory, LVM, iSCSI, NFS

• Storage Volumes
  - raw, qcow2, vmdk

• Network Interfaces
  - Bonds, bridges, ethernet devices, VLANs

• Virtual Networks
  - NAT with DHCP
  - Routed
  - Isolated
libvirt – Domain Management

• Domains defined in XML

• Lifecycle management
  – Define, start, stop, pause, resume, save, restore, migrate

• Configuration management
  – Change virtual hardware, e.g. memory, cpu
  – Add, remove, modify devices

• Tuning
  – CPU, memory, blkio, NUMA
libvirt Tools

- **virsh**
  - In-tree command line application exposing libvirt API

- **vm-install**
  - Create virtual hardware configuration
  - Install an OS in a virtual machine

- **virt-viewer**
  - Graphical console client for virtual machines

- **virt-manager**
  - Graphical tool for administering virtual machines

- **vhostmd**
  - Metrics communication channel between host and virtual machines

- **libvirt-cim**
  - libvirt-based implementation of DMTF Virtualization Management standards
Guidelines and Best Practices
Guidelines and Best Practices: General Guidance

• We'll just touch on a few things here. There is so much more that could be said.

• Best Practice is “Practice makes Perfect” - the best approach is to experiment to find optimal solution.
  – Virtualization increase complexity on many levels. Validate your assumptions. Re-evaluate decisions as things evolve.

• Identify Goals: What trumps? Security, Flexibility, ROI, Consolidation, Maintainability, Performance, and more?

• Best options come from using latest host and guest releases (latest is most virtualization friendly)

• Minimize services run on the host
Guidelines and Best Practices: General Guidance (continued)

• “Virtualize and forget” attitude is just asking for trouble
  - Your workload will behave differently when virtualized
  - Use this opportunity to control more closely the memory and cpu needs of each workload

• Take advantage of the dynamic, decoupled nature of virtualization
  - Migration, pausing, specialized disk storage formats, snapshots, optimized VM to VM interaction, memory bindings
  - Easily stage new configurations, releases, patches
  - Additional networking and storage options permit better customization and performance
KVM Guidelines and Best Practices

“Walk this way”- no brainer choices (Well, mostly - there are exceptions to every rule!)

- Avoid over commitment (storage, cpu, memory)
- Use Para-virtual devices (virtio) over emulated ones
- Avoid swapping – swap within guests instead of host
- Use vhost-net (on by default)
- Bind guest to single numa node if possible
- Use kvmclock in guest
KVM Guidelines and Best Practices

“Choose you must”

• Use libvirt or qemu-kvm command-line

• Keep Guest Migratable?
  
  – Migration inhibitors: device passthrough, incompatible target cpu, non-cache coherent shared or non-shared storage, certain storage formats and cache modes, virtfs, -mem-path option
KVM Guidelines and Best Practices

“Beware the Jabberwock” Use caution, don't be silly!

- Using qemu-kvm command-line directly
- Using cache=unsafe
- Guest vcpu count > host cpu count
- NFS as shared storage
- Sparse file-backed storage
- Using unsupported features or configurations
  - Nested virtualization, tcg cpu emulation, sound hw, certain vcpu types and emulated devices, host running 32 bit kernel
KVM Guidelines and Best Practices

“Odds and Ends”

- Memory: THP, KSM, ballooning, Explicit Hugepages
  - How to implement a memory policy
- vcpu pinning – generally improves performance
- cache=none is generally best
- Host i/o scheduler: deadline is generally best
- Run qemu-kvm as non-root for better security
- Keep guest memory and vcpus at minimum needed
- Reserve resources for the host
- PCI NIC passthrough for higher network performance
Attend also session:

**ATT1802** Hands on with KVM Virtualization and Libvirt

**TT1489** Xen and KVM Together at Last—in the Cloud

Thank you.
Thank You!

Questions?
Unpublished Work of SUSE. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.