Overview
Why virtualize?
Different approaches
Projects
QEMU
VirtualBox
Xen
KVM
VServer / OpenVZ
libvirt
Debian

Virtualisation in Debian - Present and future

Xen, KVM, VirtualBox, VMWare, QEmu

Jan Lübbe

Debian

2008-08-15
DebConf 8

Overview
Why virtualize?
Different approaches
Projects
QEMU
VirtualBox
Xen
KVM
VServer / OpenVZ
libvirt
Debian

Contents

1 Overview
- Why virtualize?
- Different approaches

2 Projects
- QEMU
- VirtualBox
- Xen
- KVM
- VServer / OpenVZ
- libvirt

3 Debian

Overview
Why virtualize?
Different approaches
Projects
QEMU
VirtualBox
Xen
KVM
VServer / OpenVZ
libvirt
Debian

Contents

1 Overview
- Why virtualize?
- Different approaches

2 Projects
- QEMU
- VirtualBox
- Xen
- KVM
- VServer / OpenVZ
- libvirt

3 Debian

Why virtualize?

- using applications written for a different OS
- testing (software and networking)
- security
- consolidation
- increasing availability/redundancy
- “cloud computing“ infrastructure
- “follow the sun“
- something else?

Different approaches

- Full System Virtualization
  - VMWare, VirtualBox, KVM, ...
- Paravirtualization
  - Xen, lguest, UML, ...
- Container Virtualization
  - linux-vserver, Virtuzzo/OpenVZ, ...
- API Translation
  - Wine, ...
- System Emulation
  - QEMU, PearPC, DOSBox, Hercules, ...

Emulation

- The host emulates a different architecture with all necessary hardware
- Pro: supports unmodified guests on a wide range of hardware
- Con: very large overhead
- In some cases it may be faster than the real hardware despite the overhead (i.e. arm on fast amd64 machine)

Full System Virtualization

- The guest runs unmodified on emulated hardware (on the same architecture as the host)
- Most code is executed directly, only priviledged instructions need to be emulated
- Pro: supports unmodified guests
- Con: higher complexity, some guests may have problems

Paravirtualization

- The guest is aware of the virtualization and uses special drivers to access the host
- Pro: lower overhead compared to Full-System-Virtualization
- Con: guest needs to be modified

Container Virtualization

- The host provides isolated “containers“ which are unaware of each other
- Pro: minimal overhead, better resource sharing
- Con: same kernel for all guests, less isolation

API Translation

- Applications for a different OS can be run without modification
- Pro: no need to run an additional OS, low overhead
- Con: very complex, not all applications are supported

Overview
Why virtualize?
Different approaches
Projects
QEMU
VirtualBox
Xen
KVM
VServer / OpenVZ
libvirt
Debian

Contents

1 Overview
- Why virtualize?
- Different approaches

2 Projects
- QEMU
- VirtualBox
- Xen
- KVM
- VServer / OpenVZ
- libvirt

3 Debian

QEMU - Approach

- Supports many host and guest architectures and operating systems
- Has a large library of emulated devices and is easy to extend
- Uses dynamic binary translation (which is faster than CPU emulation)
- QEMU is used by Xen and KVM for device emulation

QEMU - Features

- VM snapshots
- USB passthrough
- For x86 on x86 and amd64 on amd64 KQEMU gives a large speedup
- Management with Qemulator (a lightweight GUI)

VirtualBox - Overview

- Started by Innotek, now developed by Sun
- VirtualBox OSE is free software, core of the commercial product
- Uses parts of QEMU and dynamic guest patching
- Good GUI, supports seamless windows
- No USB passthrough in the OSE variant

Xen - Overview

- Supports x86, amd64 and IA64
- Started as paravirtualization for Linux, now supports full-system-virtualization on modern CPUs
- Xen is a small hypervisor which starts a modified host system (Dom0)
- The guests run parallel to the host (as a DomU) with emulated devices in Dom0
- RedHat pushes the mainline merge

Xen - Features

- Mature implementation
- Several management tools available
- Live migration (interruption $<$ 0.5 seconds)
- PCI passthrough
- Management via xenman, xen-tools, xen-shell, ganeti

Xen - Problems

- Missing support for current kernels as Dom0 and only basic support for DomU
  - Only DomU kernel support in lenny, 2.6.28 will probably support Dom0
- Hypervisor does not support suspend/powersaving, problematic on notebooks, 3d acceleration on the host difficult
- Code duplication between hypervisor and Linux kernel

KVM - Overview

- Supports x86 and amd64 (IA64, PowerPC 44x and S390 currenly in development)
- Started as a full-system-virtualization feature to the Linux kernel
- Hardware support (Intel VT or AMD-V) is necessary, support for paravirt-ops (the Linux paravirtualization interface) can speed up I/O
- The guests run as process on the host
- Uses QEMU to emulate hardware
- Is the base for Qumranet's product Solid ICE
- Ubuntu uses KVM as the default hypervisor

KVM - Features

- No external hypervisor necessary, smaller code size than Xen
- All features available on the host (power saving, suspend, 3d grapics, realtime)
- Experimental support for Xen guests via Xenner
- Can be used as an unprivileged user
- Live migration
- Guest swapping

KVM - Problems

- Still under heavy development, regressions happen sometimes (stability and performance)
- No integrated management tools

VServer / OpenVZ - Approach

- Virtualizes the Linux kernel, each container has its own view of the system
- Linux-VServer is a community project, OpenVZ was released by Parallels and is the base for their commercial product
- OpenVZ is working on merging their code into mainline Linux
- Provides isolation for: filesystems, users, processes, network, devices
- CPU, memory and bandwidth is shared according to quotas

VServer / OpenVZ - Features

- The host can directly access all files and processes of the guests (which simplifies administration)
- Memory and disk caches can be shared more efficiently
- OpenVZ supports checkpointing and live migration (although not as seamless as Xen/KVM)

VServer / OpenVZ - Problems

- VServer is not released for the latest kernels, experimental patch available since two days ago
- The isolation is weaker than with full-system-virtualization, especially disk bandwidth and kernel resources

libvirt - Overview

- Toolkit to interact with many virtualization systems
  - Supports Xen, QEMU, KVM, LXC, OpenVZ
  - Sponsored by RedHat
- Management of virtual machines, virtual networks and storage
- Access via DBus/PolicyKit or SSH/Kerberos/x509
- Zero-conf discovery using Avahi multicast-DNS

Overview
Why virtualize?
Different approaches
Projects
QEMU
VirtualBox
Xen
KVM
VServer / OpenVZ
libvirt
Debian

Contents

1 Overview
- Why virtualize?
- Different approaches

2 Projects
- QEMU
- VirtualBox
- Xen
- KVM
- VServer / OpenVZ
- libvirt

3 Debian

Current state

- Xen has no kernel support for Dom0, all other tools are in lenny
  - For Lenny you may to use the Etch Dom0 kernel until Lenny+Half
- VServer is supported in etch, not yet in lenny
- OpenVZ will be added in lenny
- QEMU, VirtualBox OSE, KVM, OpenVZ, libvirt will be in lenny
  - Most are easy to install and use

Future

- Xen will probably have Dom0 support in mainline with 2.6.28
- OpenVZ may be merged into mainline
- libvirt will make management easier and more consistent
- QEMU?
- Long term: OSs become more aware of virtualization
  - Better resource sharing, performance
  - Pass-through for 3D graphics
- ...?

Help needed!

- How can Debian improve regarding virtualization?
- Testing needed for many different ...
  - processors (Core Duo, Core 2 Duo, Xeon *, AMD, ...)
  - architectures (i386, amd64, ppc, ia64, s390, arm)
  - guest systems
- Hard to do on porter machines (root access, kernel patches)

Thanks for listening!

http://wiki.debian.org/SystemVirtualization