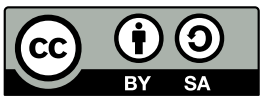
Virtualisierung

Andreas Teuchert

9. Februar 2015



Einführung

- Virtualisierung: Aufteilung physikalischer Ressourcen in mehrere virtuelle
- Beispiel: CPUs, Festplatten, RAM, Netzwerkkarten
- effizientere Nutzung von Hardware
- höhere Verfügbarkeit durch einfachere Migration
- Host: physikalischer Rechner
- Gast: virtueller Rechner auf einem Host
- Hypervisor verwaltet die Hardware

Container

- Hardware wird nicht emuliert
- Host-Kernel wird mitverwendet
- auch ohne spezielle Hardwareunterstützung sehr effizient
- weniger Isolation
- auch bekannt als Jails oder Zones
- Beispiele: OpenVZ, LXC

Diese Art der Virtualisierung wird von vielen Betriebssystemen als Container (Linux), Jails (FreeBSD) oder Zones (Solaris) angeboten. In diesem Fall läuft in der virtuellen Maschine kein vollständiges Betriebssystem, sondern der Kernel des Hosts wird mitverwendet. Es findet lediglich eine Isolierung der Prozesse, Netzwerk-Verbindungen und des Dateisystems statt. Durch den Verzicht auf das Emulieren von Hardware sind diese Lösungen sehr leichtgewichtig.

Paravirtualisierung

- „Pseudo-Hardware“ wird emuliert
- angepasstes Gast-Betriebssystem benötigt
- auch ohne spezielle Hardwareunterstützung sehr effizient
- mehr Isolation als bei Containern
- echte Hardware kann für den Gast verfügbar gemacht werden, dann aber nur exklusiv von diesem Gast nutzbar
- Beispiel: Xen (PV-Modus)

Paravirtualisierung setzt keine spezielle Hardware voraus. Es wird lediglich ein angepasstes Betriebssystem benötigt, das Treiber für die „Pseudo-Hardware“ enthält. Durch den Verzicht auf vollständige Hardware-Emulation ist auch diese Lösung sehr effizient. Dadurch, dass jede virtuelle Maschine über ihren eigenen Kernel und über eigene Hardware verfügt, ist eine größere Isolation als bei Containern vorhanden.

Vollvirtualisierung

- vollständig emulierte Hardware
- jeder Gast ist ein vollständiger virtueller Rechner
- führt vollständiges Betriebssystem aus
- auch unmodifizierte Gast-Betriebssysteme unterstützt
- ohne Hardwareunterstützung sehr ineffizient
- Hardwareunterstützung seit einigen Jahren in allen x86-CPUs vorhanden
- echte Hardware kann für den Gast verfügbar gemacht werden, dann aber nur exklusiv von diesem Gast nutzbar
- Beispiele: VirtualBox, QEMU, KVM, Xen (HVM-Modus)

Vollvirtualisierung emuliert einen kompletten Rechner mit virtueller Hardware, die vom Gastbetriebssystem wie echte Hardware angesprochen werden kann. Daher können auch unmodifizierte Betriebssysteme ausgeführt werden.

Mit modernern Prozessoren, die Hardware-basierte Virtualisierung unterstützen, ist auch Vollvirtualisierung ohne große Performanzeinbußen möglich.

Vollvirtualisierung kann auch verwendet werden, um andere Prozessor-Architekturen zu emulieren, was vor allem in der Entwicklung von Software für eingebettete Systeme sehr nützlich ist. Dies ist zwar in jedem Fall sehr ineffizient, aber für Testzwecke ausreichend. Vor allem QEMU ist für diesen Anwendungsfall geeignet.

KVM

- Kernel-based Virtual Machine
- seit 2007 Teil des Linux-Kernels
- normaler Linux-Kernel kann als Hypervisor verwendet werden
- virtuelle Maschinen laufen als Prozesse
- können mit dem Programm `kvm` gestartet werden
- Eigenschaften der VMs werden über Kommandozeilen-Argumente festgelegt
- Vollvirtualisierung, mit Paravirtualiserungs-Einflüssen über VirtIO
- benötigt CPU mit Hardware-Virtualisierung

KVM ist dank der vollständigen Integration in den Linux-Kernel eine beliebte Vollvirtualisierungs-Lösung unter Linux, da im Gegensatz zu Alternativen wie Xen und VirtualBox kein spezieller Kernel oder Kernel-Module benötigt werden.

Prinzipiell emuliert KVM einen vollständigen Rechner, auf dem auch unmodifizierte Gastbetriebssysteme ausgeführt werden können. Zur Effizienzsteigerung kann aber bei vorhandener Unterstützung im Gast auch Paravirtualisierung für Festplatte, Netzwerkkarte, Grafikkarte und RAM verwendet werden. Dieses Feature wird als VirtIO bezeichnet.

KVM – Beispiel

- `kvm -hda test.img -cdrom install.iso -boot d -m 2048`
- startet eine VM mit `test.img` als Festplatte, `install.iso` als CDROM-Laufwerk, Boot von CDROM und 2048 MB RAM
- viele weitere Optionen möglich (siehe Manpage)
- Nachteil: VM läuft im Vordergrund, unpraktisch bei der Verwaltung mehrerer VMs

Das Starten von KVM-Gästen über die Kommandozeile ist zwar möglich, aber auch recht unkomfortabel. Aus diesem Grund werden üblicherweise zusätzliche Verwaltungslösung verwendet.

libvirt

- API und Werkzeuge zur Verwaltung vieler verschiedener Virtualisierungslösungen
- u.a. KVM, Xen, OpenVZ, VMware, Hyper-V
- API-Bibliotheken für viele verschiedene Programmiersprachen
- Kommandozeilen-Werkzeug `virsh`
- grafisches Frontend `virt-manager`
- intern von OpenStack und RedHat Enterprise Virtualisation bzw. oVirt verwendet
- unter Debian/Ubuntu im Paket `libvirt-bin`

libvirt – Komponenten

- `libvirtd`: Hintergrunddienst; verwaltet VMs, virtuelle Netzwerke und Storage, läuft auf jedem Host
- `virsh`: Kommandozeilentool zur Kommunikation mit `libvirtd`
- virtuelle Maschinen
- virtuelle Netzwerke
- Storage

Virtuelle Maschinen

- Konfiguration liegt als XML-Dateien unter `/etc/libvirt/qemu/`
- kann mit `virsh edit <VM-Name>` editiert werden
- oder einfacher über `virt-manager` (vgl. Kurse zu Storage und Netzwerk)
- Anlegen am einfachsten über `virt-manager`
- oder bei vorhandener XML-Beschreibung mit `virsh define <Datei>`

Virtuelle Netzwerke

- Konfiguration unter `/etc/libvirt/qemu/networks/`
- kann mit `virsh net-edit <Net-Name>` editiert werden
- oder über `virt-manager`: Rechtsklick auf den Host, Details und dann im Reiter Virtual Networks
- können entweder isoliert oder mit Verbindung zur Außenwelt konfiguriert werden
- zur Verbindung NAT oder echte geroutete Adressen möglich
- alternativ: auf dem Host mit `brctl` eine Bridge einrichten und diese außerhalb von libvirt konfigurieren
- Standard-Netz `default` ermöglicht Netzwerk-Zugriff über NAT

Storage

- Konfiguration unter `/etc/libvirt/storage/`
- kann mit `virsh pool-edit <Pool-Name>` editiert werden
- oder über `virt-manager` (wie Netzwerke, aber Reiter Virtual Storage)
- unterstützt u.a. Image-Dateien in Verzeichnissen und LVM
- Standard-Storage `default` unter `/var/lib/libvirt/images/`

Neue VM anlegen

- in `virt-manager` auf New klicken
- Name eingeben und Installations-Modus wählen
- Betriebssystems-Typ auswählen
- RAM festlegen
- Storage auswählen (entweder Image im Standard-Storage oder anderen Speicherort auswählen)
- vor der Installation können noch Parameter angepasst werden

Lab 10.1: libvirt installieren

- libvirt installieren
- Storage über NFS einbinden
- VM anlegen

Nützliche virsh-Befehle

- `virsh list`: Laufende VMs anzeigen
- `virsh shutdown <VM-Name>`: VM herunterfahren
- `virsh destroy <VM-Name>`: VM sofort abschalten
- `virsh start <VM-Name>`: VM starten
- `virsh dumpxml <VM-Name>`: VM-Konfiguration als XML exportieren
- `virsh define <XML-Datei>`: VM-Konfiguration importieren

Lab 10.2: virsh

- VM mit virsh exportieren
- VM auf anderem Rechner importieren und starten