Virtuelle Datenträger für KVM

Michael Kofler

27. September 2012, OpenSource Trend Days

An diesen Schräubchen können wir drehen ...

- **Speicherformat**
  - Image-Datei
  - Logical Volume (LVM)
  - Partitionen
  - Netzwerk/iSCSI
- **Bei Image-Dateien**
  - RAW
  - QCOW2
  - QED
  - Mit/ohne zugrundeliegendem LVM
- **Treiber (virtio, ide, scsi)**
- **Caching, AIO**
- **I/O-Scheduler (Kernel)**

... mit diesen Konsequenzen

- Platzbedarf
- Flexibilität bei nachträglichen Größenänderungen
- Backup-Verfahren
- Geschwindigkeit

Standardeinstellungen im Virtual Machine Manager

- lokale RAW-Image-Datei in `/var/lib/libvirt/images`
- Treiber: default (virtio bei Linux-Gästen)
- Caching: default (none bei aktuellen QEMU-Versionen)

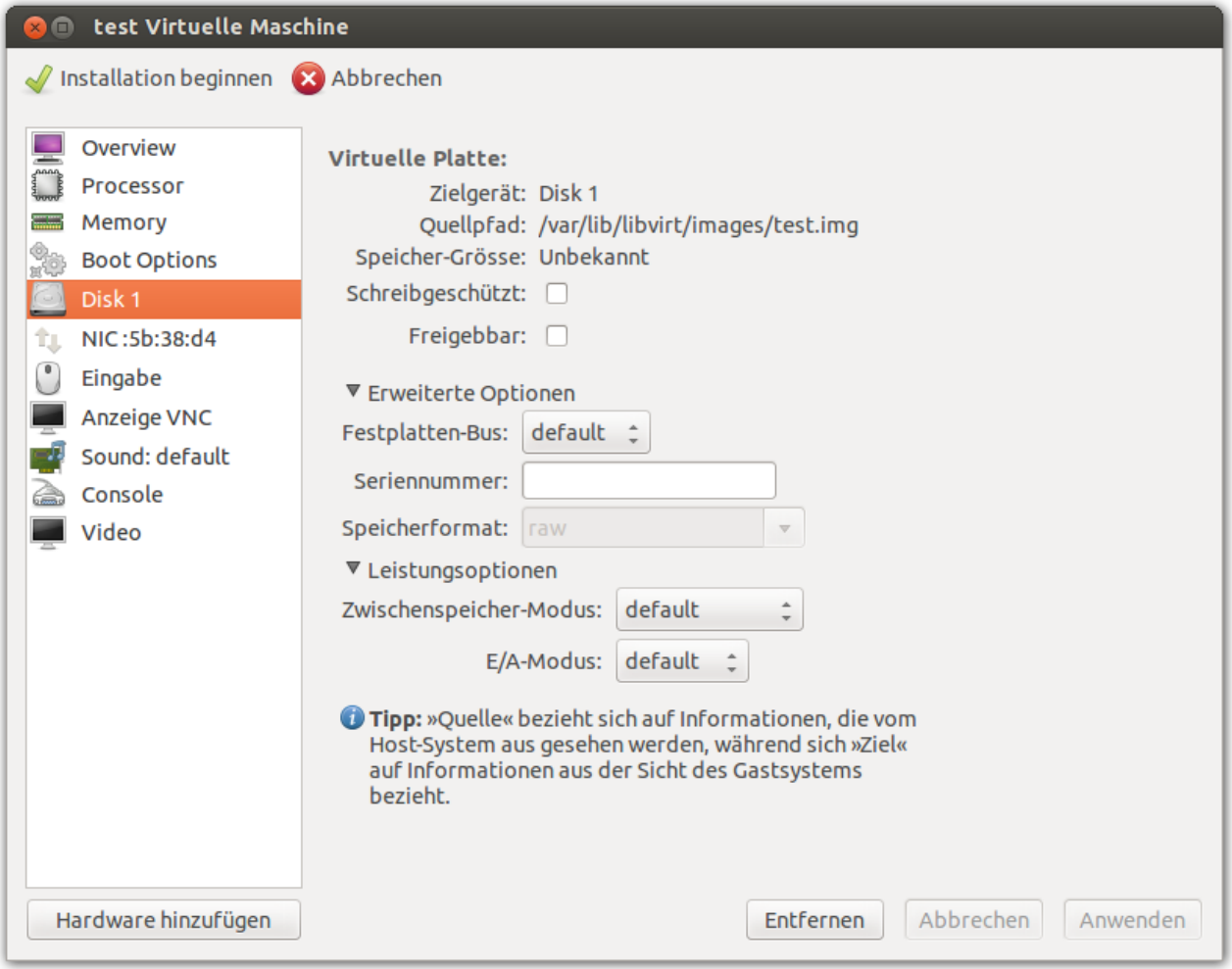


Image-Formate

- **RAW:** einfach und schnell.
  Normalerweise Sparse. Vorweg allozieren?
  Größenänderung mit `qemu-img` oder `dd`.
- **QCOW2:** anfänglich kleine Dateien (schnellere Backups).
  Snapshots, Komprimierung, Verschlüsselung.
  Etwas langsamer.
  Größenänderung ab QEMU 0.13 mit `qemu-img resize`.
- **QED:** weniger Funktionen als QCOW2, dafür etwas schneller.
  Sehr neu (Mitte 2011), keine Größenänderung.

Treiber ('Disk bus' in virt-manager)

- **virtio-block:** bei Linux-Gästen ab Kernel 2.6.25 standardmäßig, am schnellsten
- **ide:** höchste Kompatibilität, z.B. zur Windows-Installation
- **scsi:** erlaubt Hotplugging, Pass-through
- **virtio-scsi:** kombiniert Vorteile von virtio und scsi.
  Noch sehr neu (Kernel 3.4).
  Noch nicht im Virtual Machine Manager verfügbar.
  Zum Testen: Fedora 17.

Wichtig: Im Virtual Machine Manager Betriebssystemtyp angeben!

Caching-Verfahren

- **None**
- **Writethrough:** Schreib- und Lese-Caching durch Host, warten bei fsync
- **Writeback:** wie oben, aber ohne physikalische Synchronisation. Schneller, unsicher.
- **Unsafe:** Extremform von Writeback, noch schneller, aber bei Absturz nahezu garantierter Datenverlust :-(

Asynchrones I/O-Verhalten (AIO)

- **threaded (per Default):** ein Pool von Worker Threads arbeitet die I/O-Operationen ab
- **native:** I/O-Vorgänge werden direkt an den Kernel weitergeleitet. Bei Logical Volumes oder Block Devices evt. effizienter, vor allem bei vielen parallelen I/O-Zugriffen. Kann nicht mit Caching kombiniert werden!

PS: AIO heißt im Virtual Machine Manager *IO Mode*.

I/O-Scheduler des Kernels

- Normalerweise CFQ (Completely Fair Queuing)
- Bei starker I/O-Last: Deadline Scheduler
  Kerneloption `elevator=deadline` in Gast und Host
- Mögliche Alternative: `elevator=noop` im Gast

PS: AIO heißt im Virtual Machine Manager *IO Mode*.

Logical Volume Manager

- Variante 1: LVM für das Dateisystem mit den Image-Dateien (erleichtert Backups)
- Variante 2: Logical Volumes als virtuelle Datenträger: theoretisch effizienter als Disk Images. Nachteile: gesamter Speicherplatz ist sofort blockiert.

Von QCOW2 oder RAW nach LVM:

```
qemu-img convert -O raw disk.qcow disk.raw
cat disk.raw > /dev/vg1/kvmdisk
```

Partitionen und ganze Festplatten/SSDs

- Partitionen: keine Vorteile gegenüber LVM
- Ganze Festplatten: wird leider nicht unterstützt

iSCSI

- SCSI-Protokoll über TCP/IP
- direkte Unterstützung durch libvirt und Virtual Machine Manager

QCOW2-Snapshots (VM-Snapshots)

- speichert auch CPU-Zustand und RAM-Abbild
- nicht im Virtual Machine Manager
- extrem langsam, wenn Caching im Spiel
- virsh-Kommando snapshot-create

LVM-Snapshots

- für Disk Images in LVs sowie
- für LVs, die direkt als virtueller Datenträger dienen
- Wiederherstellung der virtuellen Maschine aus einem Snapshot ist wie Neustart eines Rechners nach Stromausfall

Backups

Ganz einfach, oder?

- virtuelle Maschine herunterfahren
- Image-Datei oder des LVs erstellen
- virtuelle Maschine wieder starten

Praxis: Backups auf zwei Ebenen

- in der virtuelle Maschine
- und außerhalb (Host-System) im Live-Betrieb

Live-Backup mit QCOW2-Image

```
# Backup
virsh snapshot-create-as vmname snapname
cp /var/lib/libvirt/images/vmname.img  /backup
cp /etc/libvirt/qemu/vmname.xml        /backup
cp /var/lib/libvirt/qemu/snapshot/vmname/snapname.xml \
   /backup

# Restore
cp /backup/vmname.img /var/lib/libvirt/images
virsh define /backup/vmname.xml
virsh snapshot-create vmname /backup/snapname.xml --redefine
virsh snapshot-revert vmname snapname --running
```

Live-Backup mit Disk Images + LVM

```
# Backup
cp /etc/libvirt/qemu/vmname.xml /backup
mkdir /snap
lvcreate -s -L 2G --name images-snap /dev/vg1/lv-images
mount /dev/vg1/images-snap /snap
cp /snap/vmname.img /backup
umount /snap
lvremove /dev/vg1/images-snap
rmdir /snap

# Restore
cp /backup/vmname.img /var/lib/libvirt/images/
virsh define /backup/vmname.xml
```

Live-Backup mit LV als virt. Datenträger

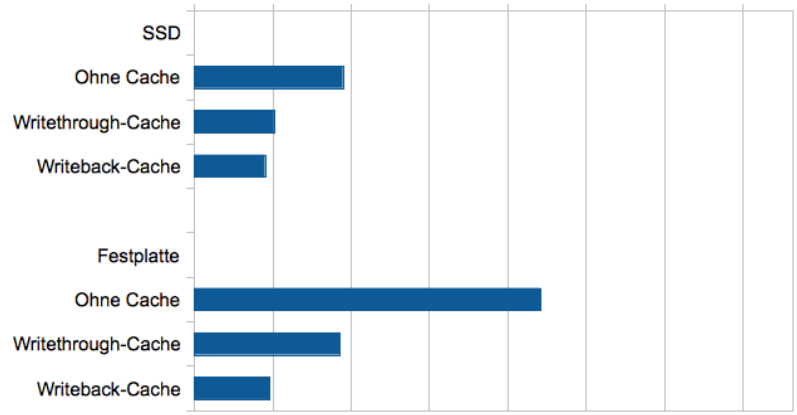
```
# Backup
cp /etc/libvirt/qemu/vmname.xml /backup
lvcreate -s -L 2G -n vmname_snap /dev/vg1/vmname
cat /dev/vg1/vmname_snap > /backup/vmname.img
lvremove -f /dev/vg1/vmname_snap

# Restore
cat /backup/vmname.img > /dev/vg1/vmname_snap
virsh define /backup/vmname.xml
```

Benchmarks I (Caching bei RAW-Images)

PARAMETER	REAL	USER	SYSTEM
RAW auf SSD, kein Cache, Threaded AIO *	2:44	0:16	0:58
RAW auf SSD, Writethrough-Cache, Threaded AIO	1:28	0:15	0:56
RAW auf SSD, Writeback-Cache, Threaded AIO	1:18	0:14	0:55
RAW auf SSD, kein Cache, Native AIO	2:42	0:16	1:01
RAW auf Festplatte, kein Cache, Threaded AIO *	6:22	0:15	0:58
RAW auf Festplatte, Writethrough-C., Threaded AIO	2:41	0:15	0:54
RAW auf Festplatte, Writeback-C., Threaded AIO	1:23	0:15	0:56
RAW auf Festplatte, kein Cache, Native AIO	6:21	0:15	1:01

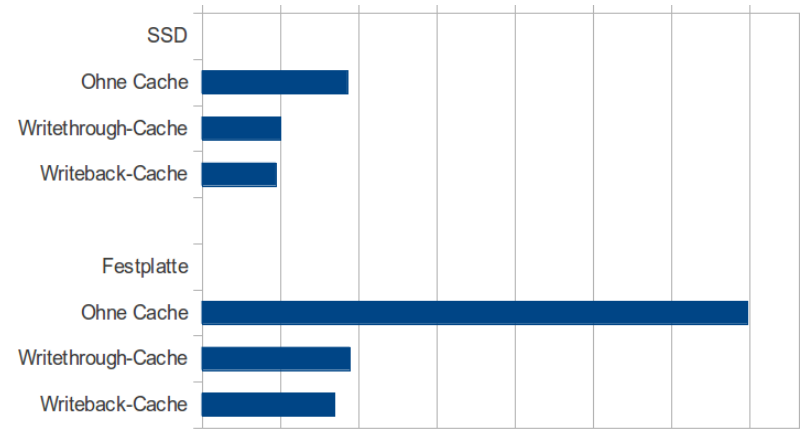
Mit QCOW2 und SSD ganz ähnliche Ergebnisse.

QCOW2 und Festplatte z.T. deutlich langsamer als RAW.



Benchmarks II (Caching bei Logical Volumes)

PARAMETER	REAL	USER	SYSTEM
LV auf SSD, kein Cache, Threaded AIO	2:41	0:16	0:57
LV auf SSD, Writethrough-Cache, Threaded AIO	1:27	0:16	0:57
LV auf SSD, Writeback-Cache, Threaded AIO	1:22	0:15	0:56
LV auf SSD, kein Cache, Native AIO *	2:41	0:15	1:00
LV auf Festplatte, kein Cache, Threaded AIO	10:03	0:15	0:57
LV auf Festplatte, Writethrough-Cache, Threaded AIO	2:44	0:15	0:46
LV auf Festplatte, Writeback-Cache, Threaded AIO	2:27	0:14	0:55
LV auf Festplatte, kein Cache, Native AIO *	9:49	0:15	1:00



Benchmarks III (LVM versus Disk Image)

Host-System mit SSD

PARAMETER	REAL	USER	SYSTEM
Host	0:41	0:13	0:27
Gast – Partition, Writethrough-Cache, Threaded AIO	1:23	0:15	0:46
Gast – LV, Writethrough-Cache, Threaded AIO	1:27	0:16	0:57
Gast – RAW-Image, Writethrough-Cache, Threaded AIO	1:28	0:15	0:56
Gast – QED-Image, Writethrough-Cache, Threaded AIO	1:31	0:15	0:58
Gast – QCOW2-Image, Writethrough-Cache, Threaded AIO	1:38	0:15	0:57

Host-System mit herkömmlicher Festplatte

PARAMETER	REAL	USER	SYSTEM
Host	1:25	0:13	0:27
Gast – Partition, Writethrough-Cache, Threaded AIO	2:43	0:15	0:46
Gast – LV, Writethrough-Cache, Threaded AIO	2:44	0:15	0:46
Gast – RAW-Image, Writethrough-Cache, Threaded AIO	2:41	0:15	0:54
Gast – QED-Image, Writethrough-Cache, Threaded AIO	3:07	0:14	0:46
Gast – QCOW2-Image, Writethrough-Cache, Threaded AIO	4:21	0:15	0:54

Empfehlungen

- **Disk Images:** RAW ist schneller, QCOW2 hat Snapshots
- **Disk Images auf LVM-System:** kein negativer Einfluss auf Geschwindigkeit, aber viel bessere Backup-Möglichkeiten!
- **Logical Volumes als Datenträger:** ein paar Prozent schneller als Disk Images, ohne Caching evt. auch langsamer!
- **Caching:** Writethrough-Caching beschleunigt I/O spürbar (vor allem bei herkömmlichen Festplatten), macht aber QCOW2-Snapshots sehr langsam.
  Writethrough- und Unsafe-Caching ist unsicher (evt. zur Installation).

Tuning lohnt sich nur für virt. Maschinen mit hoher I/O-Aktivität!

The end ...

http://kofler.info/ — kontakt@kofler.info