

Volume Manager und RAID unter Linux

Matthias Mewis

7. Juni 2001

Volume Manager

- Was ist ein Volume Manager?
 - System zur Verwaltung persistenter Speicher
 - Abstraktionsschicht zu den physikalischen Medien
 - Alternative zu Hardware RAID-Lösungen

- Anwendungsfelder von Volume Managern
 - Trennung von logischem und physikalischem Speicherbereich
 - flexible Verwaltung des logischen Speicherbereiches
 - repräsentationsunabhängige Hardwareänderungen

- spezielle Anwendungen in Linux
 - Logical Volume Manager (LVM)
 - Multiple Devices (MD)

Volume Manager in Linux

- Zielstellung von LVM
 - Realisierung des RAID-Level 0
 - * Verbindung physikalischer Medien zu einem
 - * *Striping* – gleichmäßige Verteilung der Daten auf den Medien
 - Beibehaltung der logischen Repräsentation bei . . .
 - * Größenänderung der logischen Partition
 - * Hinzufügen und Entfernen physischer Medien

- Zielstellung von MD
 - Realisierung der RAID-Level 0,1
 - * Verbindung von physischen Medien zu einem gesamten
 - Realisierung der RAID-Level 4,5
 - * Erhöhung der Verfügbarkeit und Ausfallsicherheit durch redundante Informationen

Volume Manager allgemein

- Vorteile durch die „Verbindung“ der physischen Medien
 - optimale Nutzung von Speicherfragmenten
 - Bereitstellung sehr großer virtueller Partitionen
- Vorteile bei der Verwaltung des virtuellen Speichers
 - dynamische Anpassbarkeit der virtuellen Partitionsgröße
 - transparente Änderungen an physischen Medien möglich
- Vorteile im Hinblick auf Effizienz und Fehlertoleranz
 - geringere Ausfallwahrscheinlichkeit des gesamten virtuellen Speichers
 - durch „Striping“ . . .
 - * effizientere Zugriffe
 - * höhere Bandbreite
- Nachteile:
 - zusätzliche Abstraktionsschicht birgt potentielle Fehler in der Implementierung
 - Systemressourcen für Verwaltung notwendig

Logical Volume Manager (LVM) in Linux

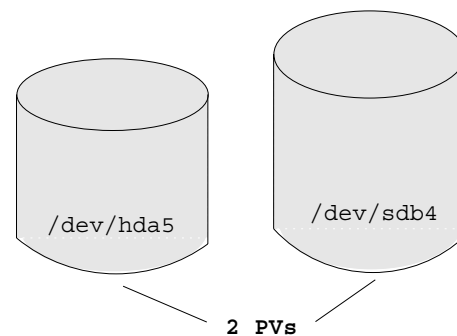
- Entwicklungsstand
 - 2.2er Kernel benötigt Patch
 - in 2.4er Kernel enthalten
 - Verwaltungswerkzeuge
 - * lvm-0.8final (<http://linux.msede.com/lvm>)
 - create, resize, delete, . . .
- Fähigkeiten
 - Verbinden von physischen Medien (PVs)
 - * linear - kontinuierliche Datenablage
 - * striped - Gleichverteilung der Datenblöcke (in „Streifen“)
 - hinzufügen/ entfernen von PVs
 - Größenänderung zur Laufzeit
 - * gleichzeitige Größenänderung des ext2-FS (resize2fs)
 - Snapshots - Momentaufnahme des LVM für Backupzwecke

Architektur des Linux LVM (1)

- Partition physischer Medien entspricht einem Physical Volume (PV)
- Erstellung:
 - `pvcreate /dev/hda5`
 - `pvcreate /dev/sdb4`

- PV - Programme

- `pvchange` - Attribute ändern
- `pvcreate` - Initialisierung eines PV
- `pvdata` - debugging
- `pvdisplay` - Informationen
- `pvmove` - verschiebe Datenposition
- `pvscan` - Suche nach PVs



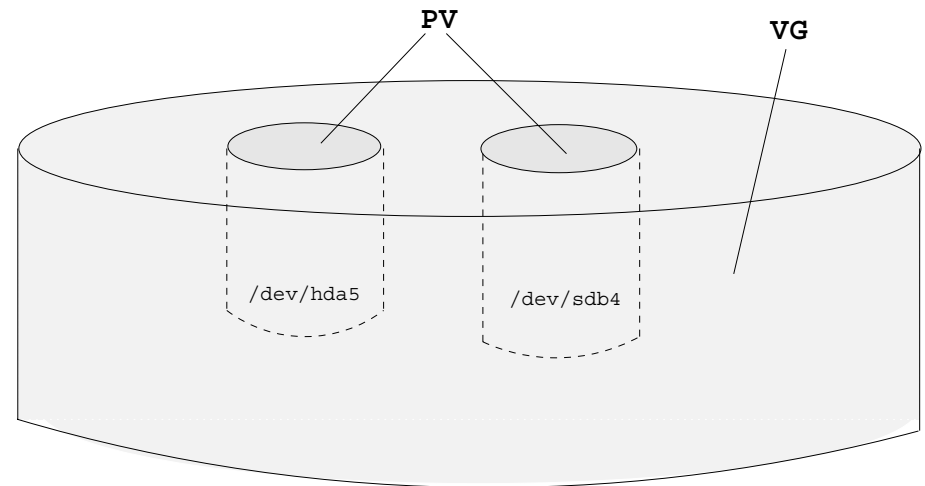
Architektur des Linux LVM (2)

- Verbinden der PVs zu einer Virtual Group (VG)

- `vgcreate My_vg /dev/hda5`
- `vgextend My_vg /dev/sdb4`
- Zugriff: `/dev/My_vg/`

- Kommandos einer VG

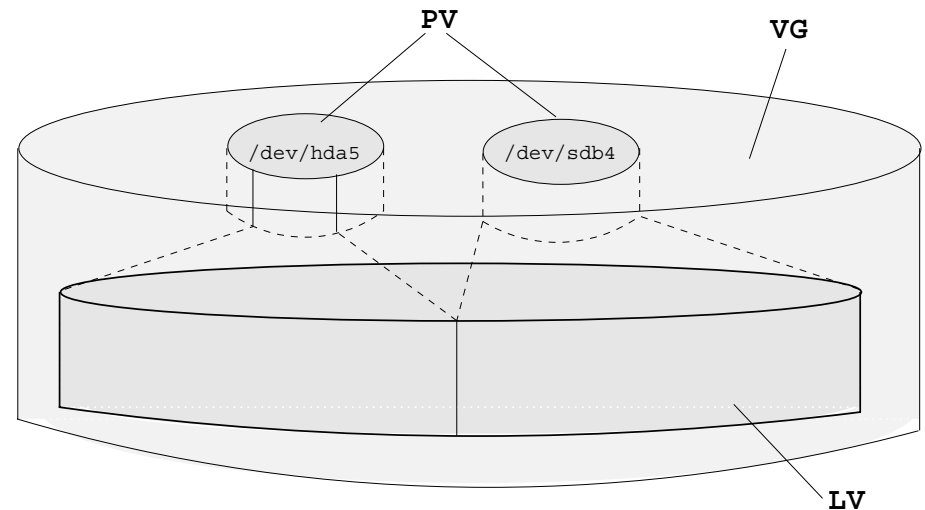
<code>vgchange</code>	-	Attribute ändern
<code>vgck</code>	-	check VG
<code>vgcreate</code>	-	erzeugen von VGs
<code>vgextend</code>	-	ändern von VGs
<code>vgmerge</code>	-	zusammenfügen von 2 VGs
<code>vgreduce</code>	-	verkleinern von VGs online
<code>vgsplit</code>	-	teilen in 2 VGs
<code>vgremove</code>	-	löschen leerer VGs



Architektur des Linux LVM (3)

- Erzeugen eines Logical Volumes (LV) als Teil einer VG
 - `lvcreate -L500 -nMy_lv My_vg`
 - Zugriff: `/dev/My_vg/My_lv`
- Zuordnungseinheiten physisch ↔ logisch
 - Physikal Extents (PE)
 - Logical Extents (LE)
- Kommandos einer LV (Auswahl)

<code>e2fsadm</code>	-	Größe verändern (benötigt <code>resize2fs</code>)
<code>lvcreate</code>	-	erzeugen
<code>lvextend</code>	-	vergrößern zur Laufzeit
<code>lvreduce</code>	-	verkleinern zur Laufzeit
<code>lvremove</code>	-	löschen eines ungenutzeten LV
<code>lvscan</code>	-	nach LVs suchen



Linux von LVM booten

- Root-Partition auf einem LV (ab LVM Version 0.7 möglich)
 - /boot-Partition zum Starten notwendig (beachte: 1024 Zylindergrenze)
- Vorgehensweise
 1. Erstellung einer komprimierten, initialen Ramdisk /boot/initrd.gz
 - Anlegen durch: `lvcreate_initrd <Kernelversion>` (z.B. 2.4.4)
 2. Erstellung des root-LVs
 - Dateisystem anlegen: `mke2fs /dev/My_vg/My_lv`
 - mounten: `mount /dev/My_vg/My_lv /mnt/newroot`
 - Kopieren des root-Filesystems nach /mnt/newroot
 3. Konfiguration der /etc/lilo.conf
 - `image = /boot/vmlinuz`
 - `initrd = /boot/initrd.gz`
 - `root = /dev/My_vg/My_lv`
 - `append = ramdisk_size=8192"`
 4. Konfiguration der /etc/fstab auf dem neuen root-LV
 - Root-Partition anpassen: `/dev/My_vg/My_lv / ext2 defaults 0 1`

Betrachtung des Linux LVM

- Vorteile eines LVM in Linux
 - Abstraktion von den physischen Medien
 - Verbindung einzelner physischer Speicherfragmente zu einem virtuellen Ganzen
 - Transparente Größenänderung des virtuellen Speichers
 - Hinzufügen und Entfernen physischer Medien ohne Änderungen an der Repräsentation

- Nachteile eines LVM in Linux
 - Eine weitere Abstraktionsschicht erhöht die Fehleranfälligkeit
 - Weitere Werkzeuge sind für Systemverwaltung / Systemrettung notwendig
 - Fehler einer Partitionstabelle gefährden alle Daten
 - Daten werden nicht redundant gespeichert

RAID - was ist das?

- RAID = „Redundant Arrays of Inexpensive Disks“
 - Patterson, Gibson, Katz (1987) University of California Berkeley
- Ziel: viele kleine und kostengünstige Medien zur Performanzsteigerung verbinden
 - MTBF - Zeit zwischen dem Auftreten von Fehlern sinkt
 - deshalb weiteres Ziel:
 - * Erhöhung der Redundanz → höhere Fehlertolleranz
- Hardware-Raid ↔ Software-Raid
 - Hardware: effizienter, aber teurer
 - Software: preisgünstig, geringere Effizienz, benötigt Ressourcen des BS
- Software-Raid in Linux: Multiple Devices (MD)
 - im 2.4er Kenel enthalten
 - stellen Raid-Level 0,1,4,5 bereit
- Linux Multiple Devices befinden sich im stabilen Stadium
 - werden nicht mehr weiter entwickelt
 - werden nicht mehr supported

RAID - Level (1)

- Level 0
 - zwei Ausprägungen:
 - * lineare Datenverwaltung
 - * gleichverteilung der Daten - „stripping“
 - Daten auf mehrere Medien verteilt
 - keine Redundanz - dadurch geringe Fehlertoleranz
 - höhere Performanz durch verteilte, parallele Zugriffe
- Level 1
 - zwei Medien werden gebraucht
 - kleinstes Medium bestimmt Größe des RAID
 - Redundanz durch doppeltes schreiben auf verschiedene Medien (mirror)
 - schnell beim lesen, langsam beim schreiben
 - Fehlertolerant - kein Datenverlust bei einem Mediendefekt
 - hohe Kosten durch doppelte Hardware

RAID - Level (2)

- Level 2
 - verwendet Hamming-Code zur Fehlererkennung
 - nur bei IDE Medien notwendig
 - SCSI enthält bereits diese Fehlererkennung
 - wird selten verwendet
- Level 3
 - verteilt Daten auf Byteebene über mehrere Medien
 - Checksummen werden auf separaten Medium verwaltet
 - geringe Zugriffseffizienz durch kleinste Zugriffseinheiten
 - * Unterstützung durch Hardware erforderlich
 - Fehlertollerant - kein Datenverlust bei einem Mediendefekt
 - relativ geringe Hardwarekosten

RAID - Level (3)

- Level 4
 - verteilt Datenblöcke über mehrere Medien
 - * dadurch bessere Performance als Level 3
 - Checksummen werden auf separatem Medium verwaltet
 - Datenwiederherstellung bei Fehler eines Mediums
 - Performance beim Lesen ist sehr gut - wie Level 0
 - geringere Geschwindigkeit bei Schreibzugriffen
 - * Schwachstelle ist das Checksummen - Medium

- Level 5
 - Datenverwaltung wie in Level 4
 - Checksummen werden über die Medien verteilt, dadurch . . .
 - * höhere Effizienz bei Schreiben als Level 4
 - * geringere Lesegeschwindigkeit durch das Überspringen der Checksummen
 - hohe Fehlertoleranz / Wiederherstellung nach einem Medienfehler
 - relativ geringe Hardwarekosten
 - häufig eingesetztes Softwareraid

Aufbau der Multiple Devices in Linux

- im 2.4er Standardkernel enthalten
 - 2.2er Kenel benötigt Patch
- Werkzeuge zur Verwaltung notwendig „Raidtools“
 - `ftp://ftp.kernel.org/pub/linux/daemons/raid/alpha/`
 - `raidtools-19990824-0.90.tar.gz`
- Vorbereitungen
 - RAID-Unterstützung und Raid-Level im Kernel aktivieren
 - Raidtools installieren
 - Informationen über aktuellen Zustand der Raids: `/proc/mdstat`
 - Zugriff über: `/dev/md[0|1|2|3]`
 - Konfigurationsdatei: `etc/raidtab`

Erstellung eines Softwareraid unter Linux

- Beispielkonfiguration `/etc/raidtab` →
- Raid-Level 5
 - 3 physische Medien
 - 1 Ersatzmedium
- Persistenter Superblock:
 - Infos über ungemountetes RAID
- Chunk sizes: kleinste Zuordnungseinheit

- Raid initialisieren: `mkraid /dev/md0`
- Starten durch `/etc/init.d/-Script`:
 - `raidstart /dev/md0`
- Formatieren mit ext2-FS:
 - `mke2fs -b 4096 -R stride=8 /dev/md0`
- Automount: Partitionstyp `0xFD`

<code>raiddev</code>	<code>/dev/md0</code>
<code>raid-level</code>	<code>5</code>
<code>nr-raid-disks</code>	<code>3</code>
<code>nr-spare-disks</code>	<code>1</code>
<code>persistent-superblock</code>	<code>1</code>
<code>parity-algorithm</code>	<code>left-symmetric</code>
<code>chunk-size</code>	<code>32</code>
<code>device</code>	<code>/dev/sdb6</code>
<code>raid-disk</code>	<code>0</code>
<code>device</code>	<code>/dev/sda6</code>
<code>raid-disk</code>	<code>1</code>
<code>device</code>	<code>/dev/hdc6</code>
<code>raid-disk</code>	<code>2</code>
<code>device</code>	<code>/dev/hda6</code>
<code>spare-disk</code>	<code>0</code>