Westfälische Hochschule

- Linux Hochverfügbarkeit - (Lokale HA)

Christoph Hoopmann
Michael Krane
Marcel Rothe

Agenda

I. Grundsätze Überlegungen (Rothe)
II. HW HA: Netzteile & CPUs (Krane)
III. SW HA: Tools (Krane)
IV. Lokaler Storage (Rothe)
V. LVM & BTRFS (Hoopmann)

Westfälische Hochschule

I
Grundsätzliche Überlegungen

Was ist Hochverfügbarkeit?

Allgemein: Verfügbarkeit eines Dienstes oder eines Systems mit minimalen Unterbrechungen

Aber:

- 99,9% Verfügbarkeit → ca 9h Downtime pro Jahr
- 99,9999% Verfügbarkeit → ca 30s Downtime pro Jahr

Verfügbarkeit einer Komponente: MTBF / (MTBF+MTTR)

Klassifizierung von Verfügbarkeit

Verfügbarkeitsklassen nach IEEE:

Klasse	Verfügbarkeit	Ausfall/Jahr
2	99%	ca. 3d 16h
3	99,9%	ca. 9h
4	99,99%	ca. 1h
5	99,999%	ca. 5m
6	99,9999%	ca. 30s

Klassifizierung von Verfügbarkeit

Verfügbarkeitsklassen nach Harvard Research Group (HRG)

Klasse	Bezeichnung	Bechreibung
AEC-0	Conventional	Unterbrechung + Datenverlust
AEC-1	Highly Reliable	Unterbrechung, kein Datenverlust
AEC-2	High Availability	Unterbrechung nur kurz oder wenn es nicht stört
AEC-3	Fault Resistent	Unterbrechung nur wenn es nicht stört
AEC-4	Fault Tolerant	24/7-Betrieb
AEC-5	Disaster Tolerant	Betrieb unter allen Umständen

„High Availability“ schon bei IEEE-Klasse 2 (3,6 Tage Ausfall/Jahr)

Hochverfügbarkeit erreichen

Alle Komponenten versagen im Alter

Wo möglich Redundante Komponenten verwenden

- Festspeicher, Netzwerkverbindungen, Stromversorgung

Aber: Nicht alle Komponenten redundant verfügbar

Daher: Teilsysteme im Netzwerk redundant einsetzen

Parallelität gegen Single Point of Failure (SPoF)

Hochverfügbarkeit erreichen

Daraus folgen 3 Schwerpunkte:

1: Lokale Hochverfügbarkeit

2: Hochverfügbarkeit im Netzwerk

3: Disaster Recovery, Plan haben wenn doch ein Ausfall eintritt

Je komplexer ein System, desto wahrscheinlicher SpoF

Soweit wie möglich KISS anwenden

Westfälische Hochschule

II
Hardware HA: Netzteile & CPUs

Lokale Hochverfügbarkeit

Nur mit Redundanz der Komponenten möglich!

- Welche Komponenten müssen redundant sein?
- Welche Komponenten sollten/können überwacht werden?
- Wie lassen sich Fehler im Vorfeld erkennen?
- Wie erhält man brauchbare Fehlermeldung ?
- Wie überwacht man die Wächter?

Fehlerquellen

Temperatur

- Jeder Kalt/Warm-Zyklus beansprucht Bauteile
- Server laufen 24/7 => permanente Wärmeentwicklung
- Lüfter im Gehäuse und klimatisiertes Umfeld nötig

Stöße/G-Force

- Problem für HDDs
- Stoß sichere Montage der Festplatten im Server
- Umfallen von ausgebauten Festplatten

Netzteile

Absicherung gegen Defekt und Stromausfall

Sinnvoll in Kombination mit einem USV

Korrekte Dimensionierung für Netzteil und USV

CPU

Abhängig vom Einsatzzweck

- Generell mehr Leistung ==> mehr Abwärme

Redundanz möglich?

- Lokal nicht ohne spezielle HW und SW
- Ohne CPU-Failover => Hochverfügbarkeit auf Netzwerkebene

Netzwerkkarten

Mehrere physikalische Karten zu logischen zusammengefasst

Vorteile

- Redundanz: Bei n Karten können (n-1) Karten ausfallen
- Erhöhter Datendurchsatz, abhängig vom Modus:
  - Modi die nur Redundanz haben (active-backup)
  - Modi die Lastenverteilung und Redundanz haben (balance-xor, -rr)

Switches im Netzwerk müssen Bonding unterstützen

Westfälische Hochschule

III
Software HA: Tools

CPU-Lastverteilung

Tools für die Verwaltung und Überwachung

- ulimit
- cpusets
- monit

ulimit – Erlaubt Ressourcen Limitierung auf Benutzerebene

cpusets

Prozesse können bestimmen CPU-Kernen zugewiesen werden

Ziel: kritische/System Prozesse eigener Kern

- Strikte Trennung von Anwenderprozessen

Loadbalancing für Prozesse

Konfiguration nötig für kritische Prozesse

monit

Watchdog für System-Parameter

- Loadavg, memory/cpu usage
- Interaktion mit System

Verschiedene Modi:

- Passive-Mode : Überwachung/Alarm
- Active-Mode : Überwachung/Alarm + Aktion-Trigger
- Manual-Mode : Für Cluster nur Überwachung

Fazit

- Server brauchen lokal redundanten Bauteile
- Bauteile qualitativ hochwertig / Anforderungen entsprechend
- Thermische Gesundheit gegeben und permanent überwacht
- Selektive Ressourcen-Aufteilung erhöht Verfügbarkeit
- Software zur Ressourcen-Überwachung für Stabilität nötig

Westfälische Hochschule

IV
Lokaler Storage

Auswahl von Speichermedien

Klassisch:

- SCSI/SAS für Server: schnell,ausdauernd, Controllerredundanz, bessere ED/EC, mehr Laufwerke möglich, deutlich teurer
- IDE/SATA für Storage-Pool/Nutzerhardware: langsamer, niedrigere Lebensdauer, simple Konfiguration, 5-10x billiger pro GB

Keine brandneue Hardwaregeneration benutzen, da Erfahrungswerte fehlen

Auswahl von Speichermedien

Überlegung: SSD kann keine mechanischen Fehler haben

Aber: Mehr Löschzyklen → Höhere Ausfallwahrscheinlichkeit

Beschriebene Sektoren müssen gelöscht werden, bevor sie wieder beschrieben werden können

Effekt kann im RAID verschlimmert werden

Fazit: Teuer und zu viele Nachteile, lieber klassischer Ansatz

Ausfall vorraussehen

Google-Studie: 2/3 aller Festplattenausfälle kündigen sich an

Festplattentode wahrscheinlicher im 1. und nach dem 4. Jahr

Ideale Temperatur 30°-40°C

Viele Scanfehler und belegte Reservesektor -> Ausfall nahe

Bei Geräuschen ist es meistens schon zu spät

Daten über S.M.A.R.T. verfügbar

S.M.A.R.T.

Self-Monitoring Analysis and Reporting Technology

Logging-Controller in Festplatten

Liefert Nutzungsstatistiken, Fehlerstatistiken und Zustandsdaten

Einzelne Fehler stellen kein Problem dar

Häufiges Auftreten der selben Fehler Hinweis auf baldiges Versagen

Aber: Man kann nicht schließen: Keine Fehler → Kein Ausfall

S.M.A.R.T.

Regelmäßiges auslesen von Hand skaliert nicht

Es gibt Tools zum automatischen Messen und auswerten:

- Dienst polled S.M.A.R.T.-Werte
- Bei Unregelmäßigkeiten wird Skript/Befehl ausgeführt
- Beispielaktion: Meldung an Administrator

Unter Linux: smartmontools

S.M.A.R.T.

Werte:

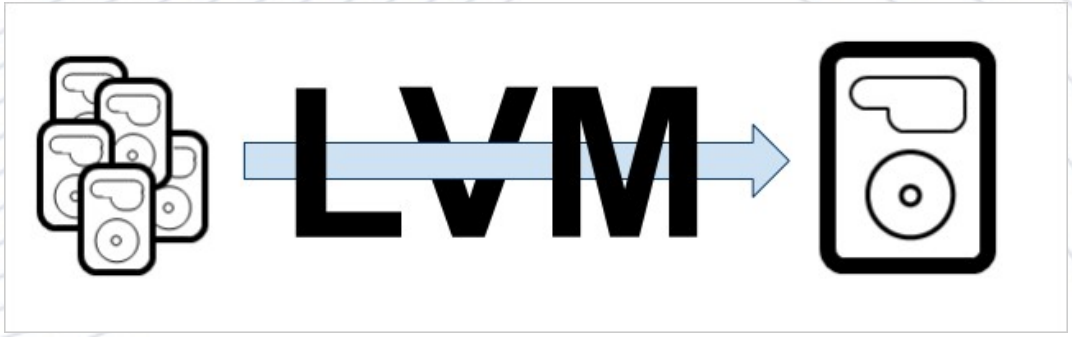
- VALUE (aktueller Wert)
- WORST (schlimmster gemessener Wert)
- THRESH (Herstellerminimum)

VALUE < THRESH → Kritischer Zustand, Platte tauschen

Westfälische Hochschule

V
LVM & BTRFS

- **L**ogial **V**olume **M**anager
- Virtualisierung
- Flexibilität
- Live-Reduce/Extend
- Cross-device FS
- „Snapshots“



PV → Physical Volume (**pv**display, **pv**create, ..)

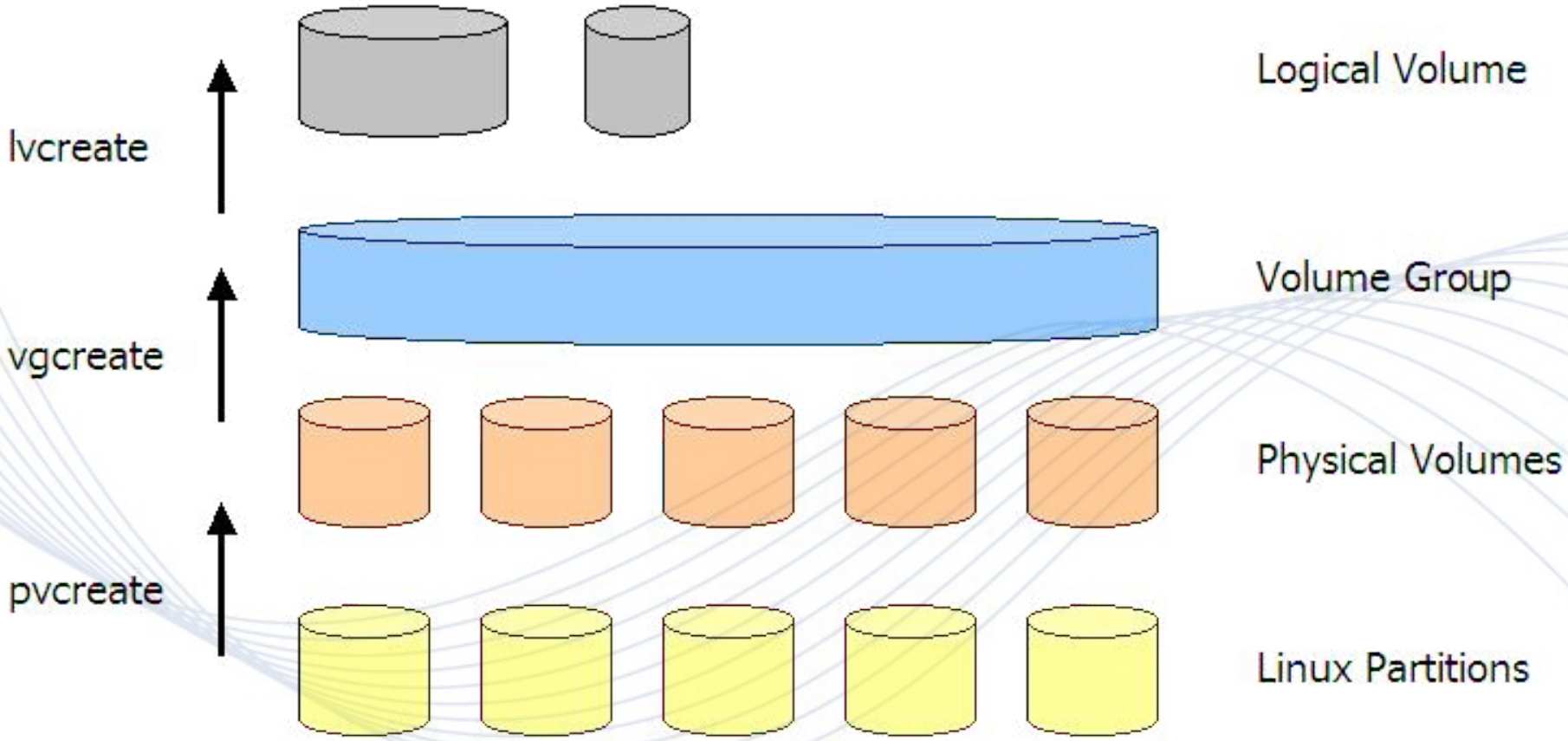
VG → Volume Group (**vg**s, **vg**remove, ..)

LV → Logical Volume (**lv**convert, **lv**extend, ..)

PE → Physical Extends

LE → Logical Extends

1 PE = 1 LE = 4 Mb


lvcreate
vgcreate
pvcreate
Logical Volume
Volume Group
Physical Volumes
Linux Partitions

Erstellung eines LVMs

```
root ~# pvcreate /dev/loop[01234]
root ~# vgcreate VG1 /dev/loop[01234]
root ~# lvcreate -L 3GB VG1 -n rootfs
root ~# lvcreate -L 1GB VG1 -n srv
root ~# mkfs.ext4 /dev/VG1/rootfs
...
```

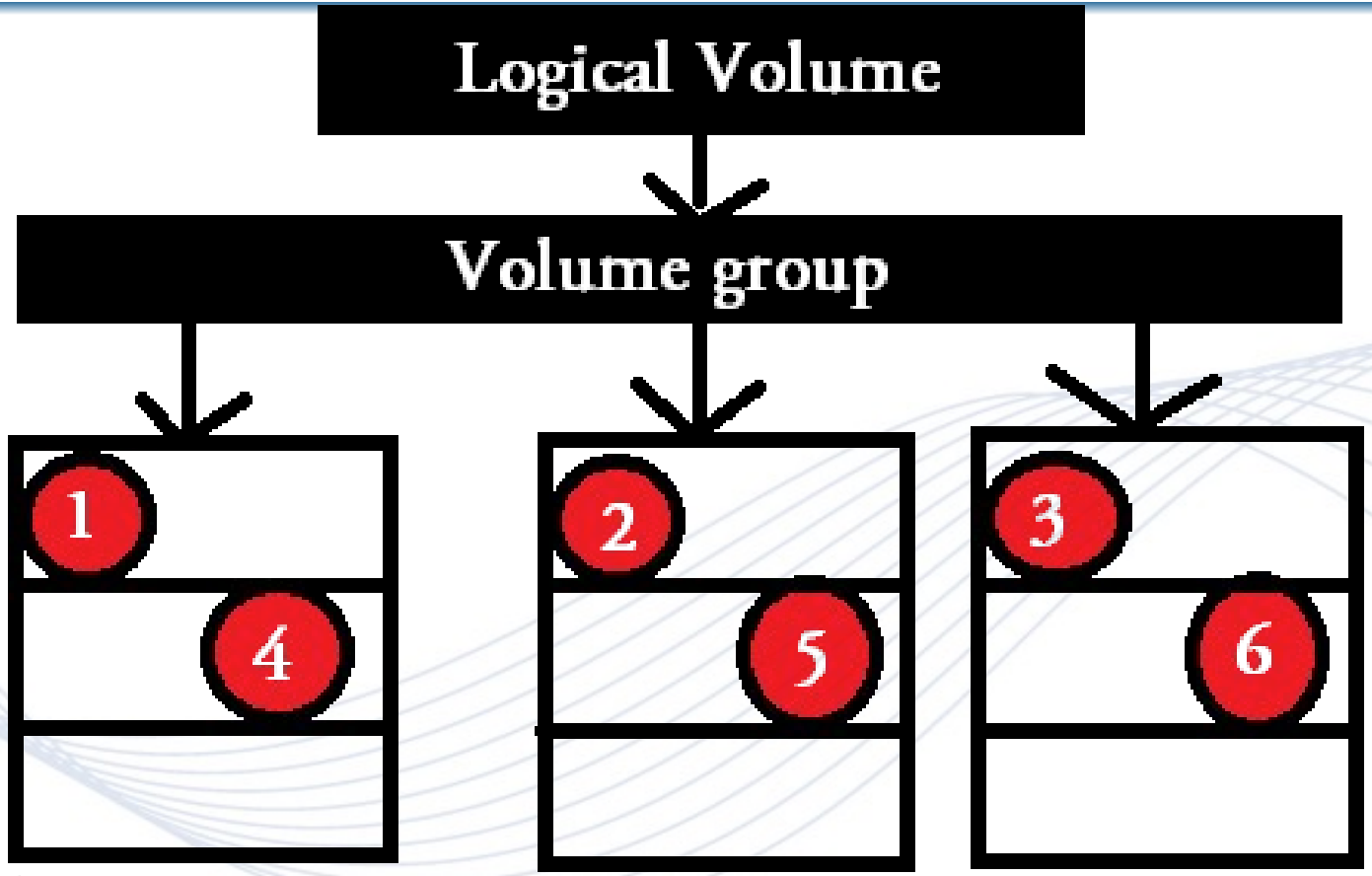
Erstellung eines LVMs

```
root ~# pvscan
  PV /dev/loop0   VG VG1  [1020,00 MiB / 0    free]
  PV /dev/loop1   VG VG1  [1020,00 MiB / 0    free]
  PV /dev/loop2   VG VG1  [1020,00 MiB / 0    free]
  PV /dev/loop3   VG VG1  [1020,00 MiB / 1004,00 MiB free]
  PV /dev/loop4   VG VG1  [1020,00 MiB / 0    free]
  Total: 5 [4,98 GiB] / in use: 5 [4,98 GiB] / in no VG: 0 [0   ]
```

Erstellung eines LVMs

```
root ~# lvscan
  ACTIVE     '/dev/VG1/rootfs' [3,00 GiB] inherit
  ACTIVE     '/dev/VG1/srv' [1,00 GiB] inherit
```

PV Striping

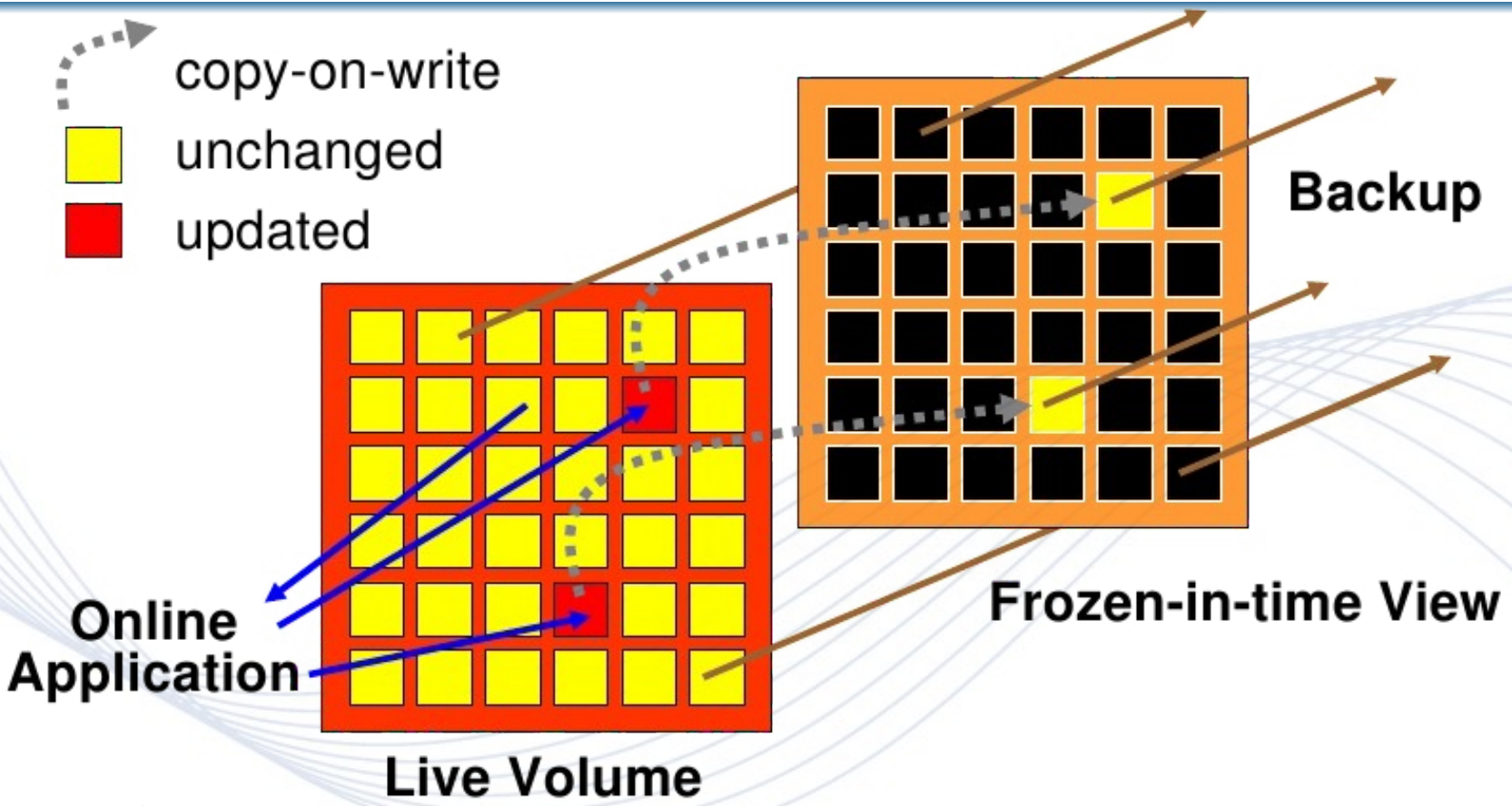


```
root ~# lvremove VG1/rootfs
root ~# lvcreate -L 3GB VG1 -n rootfs -i 4
root ~# pvscan
  PV /dev/loop0   VG VG1  [1020,00 MiB / 252,00 MiB free]
  PV /dev/loop1   VG VG1  [1020,00 MiB / 252,00 MiB free]
  PV /dev/loop2   VG VG1  [1020,00 MiB / 252,00 MiB free]
  PV /dev/loop3   VG VG1  [1020,00 MiB / 248,00 MiB free]
  PV /dev/loop4   VG VG1  [1020,00 MiB / 0    free]
  Total: 5 [4,98 GiB] / in use: 5 [4,98 GiB] / in no VG: 0 [0   ]
```

Extending 4-striped LV

```
root ~# df -h /dev/VG1/rootfs
   /dev/VG1/rootfs      3G    0  2,9G    99% /
root ~# vgextend VG1 /dev/loop[6789]
root ~# lvextend -L+1.5GB VG1/rootfs
root ~# resize2fs /dev/VG1/rootfs
root ~# df -h /dev/VG1/rootfs
   /dev/VG1/rootfs      4,5G    0  2,9G    64% /
```

Snapshots


copy-on-write
unchanged
updated
Backup
Online
Application
Frozen-in-time View
Live Volume

Creating Snapshots

```
root ~# lvcreate -L 300Mb -s -n SNAP VG1/rootfs
root ~# lvs
  LV     VG  Attr       LSize   Pool Origin Data%  Meta%
  SNAP   VG1 swi-a-s--- 300,00m      rootfs 0,00
  rootfs VG1 owi-a-s---   4,50g
  srv    VG1 -wi-a-----   1,00g
```

Creating Snapshots

```
root ~# dd if=1.vid of=VG1/rootfs bs=1M count=300
root ~# lvs
  LV     VG  Attr       LSize   Pool Origin Data%  Meta%
  SNAP   VG1 swi-a-s--- 300,00m      rootfs 83,66
  rootfs VG1 owi-a-s---   3,00g
  srv    VG1 -wi-a-----   1,00g
```

Creating Snapshots

root ~# lvremove VG1/SNAP

→ Keeping changes since SNAP

root ~# lvconvert –-merge VG1/SNAP

→ Undoing changes since SNAP

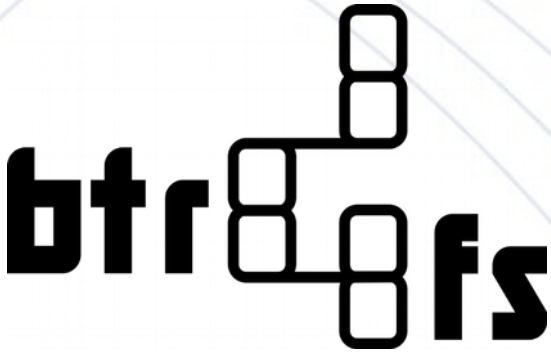
Mirrors

```
root ~# lvremove VG1/rootfs
root ~# lvcreate -L 1.5GB VG1 -n rootfs -m 1
```

→ Kein Striping, kein Auto-Rebuild

BTRFS

- CoW-FS
- Efficient (small files, folders)
- Data Compression
- Snapshots
- Subvolumes (ZFS-style, LVs)
- Integrated RAID (0,1,5,6,10)
- Online Rebuild/Fsck/Resize
- Cross device support



BTRFS Subvolumes

```
root ~# mkfs.btrfs /dev/loop[01234]
root ~# mount /dev/loop0 /mnt
root ~# btrfs sub create /mnt/rootfs
root ~# touch /mnt/rootfs/test.txt
root ~# mkdir /mnt/snaps
root ~# btrfs sub snap /mnt/rootfs /mnt/snaps/saved
```

BTRFS Subvolumes

```
root ~# rm /mnt/roofs/test.txt
root ~# ls -Alh /mnt
root ~# umount /mnt
root ~# btrfs sub list /mnt
    ID 258 gen 8 top level 5 path rootfs
    ID 259 gen 8 top level 5 path snaps/saved
root ~# mount -o subvol=snaps/saved /dev/loop0 /mnt
root ~# ls /mnt
    -rw-r--r-- 1 root root  0 10. Jun 19:39 test.txt
```

BTRFS Showoff

```
root ~# btrfs fi show /
   Label: 'ROOT'  uuid: 6f012252-7724-4175-beef-7a5c33954769
     Total devices 5 FS bytes used 484.57GiB
     devid    1 size 931.51GiB used 122.77GiB path /dev/mapper/hdd1
     devid    2 size 931.51GiB used 122.77GiB path /dev/mapper/hdd2
     devid    3 size 931.51GiB used 122.77GiB path /dev/mapper/hdd3
     devid    4 size 931.51GiB used 122.77GiB path /dev/mapper/hdd4
     devid    5 size 931.51GiB used 122.77GiB path /dev/mapper/hdd5
```

```
root ~# btrfs fi df -h /
   Data, RAID5: total=488.00GiB, used=483.48GiB
   System, RAID5: total=64.00MiB, used=48.00KiB
   Metadata, RAID5: total=3.00GiB, used=1.08GiB
   GlobalReserve, single: total=384.00MiB, used=0.00B
```

BTRFS Showoff

```
root ~# btrfs sub list /
   ID 258 gen 862051 top level 5 path __active
   ID 2668 gen 607327 top level 5 path __snapshot/autosnap-2016-05-12
   ID 2669 gen 615418 top level 5 path __snapshot/autosnap-2016-05-13
   ID 2671 gen 622737 top level 5 path __snapshot/autosnap-2016-05-14
   ID 2672 gen 629297 top level 5 path __snapshot/autosnap-2016-05-15
   .....
   ID 2699 gen 857465 top level 5 path __snapshot/autosnap-2016-06-10
```

Vielen Dank

Noch Fragen?