LVM und ZFS reloaded

gehalten auf dem Frühjarsfachgesprch 2015 in Stuttgart

Oliver Rath
oliver.rath@greenunit.de

GreenUnit UG, München

27. März 2015

Zur Person

<Werbung>

- beschäftigt bei der GreenUnit UG in München
- Entwicklung von Netzbootlösungen (1000 Clients/Server)
- Entwicklung einer transparenten Desktop Managment Infrastruktur (DMI)

</Werbung>

Warum sich mit LVM und ZFS beschftigen?

- Bedarf an performanten Images, die über Netzwerk exportiert werden
- Starker Entwicklungsschub bei LVM und ZFS unter Linux
- Neue Features

Was gibt es neues bei LVM und ZFS?

- LVM: Thinpools ($\rightarrow$ Snapshots)
- LVM: Cachepools
- LVM: RAID-1,5,6 Support
- ZFS: Implementierung von ZFS im Linux-Kernel
- ZFS: Mglichkeit, trotz inkompatibler Lizenzen ZFS unter Linux zu benutzen

Was gibt es neues bei LVM und ZFS?

- LVM: Thinpools ($\rightarrow$ Snapshots)
- LVM: Cachepools
- LVM: RAID-1,5,6 Support
- ZFS: Implementierung von ZFS im Linux-Kernel
- ZFS: Mglichkeit, trotz inkompatibler Lizenzen ZFS unter Linux zu benutzen

Kurz die Grundlagen

- Device(s) (/dev/sda, /dev/sdb etc.) werden als Physical Volumes (PV) markiert
- PVs werden zu einer Volume-Group (VG) zusammengefasst

Kurz die Grundlagen

- Device(s) (/dev/sda, /dev/sdb etc.) werden als Physical Volumes (PV) markiert
- PVs werden zu einer Volume-Group (VG) zusammengefasst
- Logische Volumes (LV) entnehmen ihre Blöcke aus der VG

Kurz die Grundlagen

- Device(s) (/dev/sda, /dev/sdb etc.) werden als Physical Volumes (PV) markiert
- PVs werden zu einer Volume-Group (VG) zusammengefasst
- Logische Volumes (LV) entnehmen ihre Blöcke aus der VG
- Demo

Kurz die Grundlagen

- Device(s) (/dev/sda, /dev/sdb etc.) werden als Physical Volumes (PV) markiert
- PVs werden zu einer Volume-Group (VG) zusammengefasst
- Logische Volumes (LV) entnehmen ihre Blöcke aus der VG
- Demo

Was sind Thinpools?

- Komplexes LV aus LV_{meta} und LV_{data}
- Problem: Schicht zwischen VGs und LVs nicht vorgesehen
- Lösung
  - LV_{meta} und LV_{data} werden versteckt (erscheinen nur in /dev/mapper) zu einem Thinpool zusammengefasst
  - Aus einem $LV_{thinpool}$ werden die Thinpool-LVs generiert

Was sind Thinpools?

- Komplexes LV aus LV_{meta} und LV_{data}
- Problem: Schicht zwischen VGs und LVs nicht vorgesehen
- Lösung
  - LV_{meta} und LV_{data} werden versteckt (erscheinen nur in /dev/mapper) zu einem Thinpool zusammengefasst
  - Aus einem $LV_{thinpool}$ werden die Thinpool-LVs generiert
  - Hier können beliebig viele (rw) Snapshots generiert werden!

Was sind Thinpools?

- Komplexes LV aus LV_{meta} und LV_{data}
- Problem: Schicht zwischen VGs und LVs nicht vorgesehen
- Lösung
  - LV_{meta} und LV_{data} werden versteckt (erscheinen nur in /dev/mapper) zu einem Thinpool zusammengefasst
  - Aus einem $LV_{thinpool}$ werden die Thinpool-LVs generiert
  - Hier können beliebig viele (rw) Snapshots generiert werden!
- Demo

Was sind Thinpools?

- Komplexes LV aus LV_{meta} und LV_{data}
- Problem: Schicht zwischen VGs und LVs nicht vorgesehen
- Lösung
  - LV_{meta} und LV_{data} werden versteckt (erscheinen nur in /dev/mapper) zu einem Thinpool zusammengefasst
  - Aus einem $LV_{thinpool}$ werden die Thinpool-LVs generiert
  - Hier können beliebig viele (rw) Snapshots generiert werden!
- Demo

Was sind Cachepools?

- Idee: LV wird gegen ein schnelles Device (SSD?) gecached
- Komplexes *LV* aus LV_{meta}, LV_{cache} und LV_{data}

Was sind Cachepools?

- Idee: LV wird gegen ein schnelles Device (SSD?) gecached
- Komplexes *LV* aus LV_{meta}, LV_{cache} und LV_{data}
- Ansatz:

Was sind Cachepools?

- Idee: LV wird gegen ein schnelles Device (SSD?) gecached
- Komplexes *LV* aus LV_{meta}, LV_{cache} und LV_{data}
- Ansatz:
  - aus LV_{meta} und LV_{cache} wird komplexes $LV_{cachepool}(SSD)$

Was sind Cachepools?

- Idee: LV wird gegen ein schnelles Device (SSD?) gecached
- Komplexes LV aus LV_{meta}, LV_{cache} und LV_{data}
- Ansatz:
  - aus LV_{meta} und LV_{cache} wird komplexes $LV_{cachepool}(SSD)$
  - explizites Auswählen des Devices möglich :-)

Was sind Cachepools?

- Idee: LV wird gegen ein schnelles Device (SSD?) gecached
- Komplexes *LV* aus LV_{meta}, LV_{cache} und LV_{data}
- Ansatz:
  - aus LV_{meta} und LV_{cache} wird komplexes $LV_{cachepool}(SSD)$
  - explizites Auswählen des Devices möglich :-)
  - LV_{data} wird mit $LV_{cachepool}$ verknpft

Was sind Cachepools?

- Idee: LV wird gegen ein schnelles Device (SSD?) gecached
- Komplexes LV aus LV_{meta}, LV_{cache} und LV_{data}
- Ansatz:
  - aus LV_{meta} und LV_{cache} wird komplexes $LV_{cachepool}(SSD)$
  - explizites Auswählen des Devices möglich :-)
  - LV_{data} wird mit $LV_{cachepool}$ verknpft
- Demo

Was sind Cachepools?

- Idee: LV wird gegen ein schnelles Device (SSD?) gecached
- Komplexes LV aus LV_{meta}, LV_{cache} und LV_{data}
- Ansatz:
  - aus LV_{meta} und LV_{cache} wird komplexes $LV_{cachepool}(SSD)$
  - explizites Auswählen des Devices möglich :-)
  - LV_{data} wird mit $LV_{cachepool}$ verknpft
- Demo

Raid mit LVM

- Vorgeschichte: LVs verteilen sich automatisch über die verfügbaren Devices
- Ansatz: Stripes wie Raid spiegeln/anordnen
- So wird Raid 1, 5, 6, 10, 12 möglich
- DEMO → bei Bedarf am Ende

Weitere Feautures LVM

- Cachen aller Metadaten ber daemons (lvmetad und dmmetad)
- Cluster (clvmd)

Weitere Feautures LVM

- Cachen aller Metadaten ber daemons (lvmetad und dmmetad)
- Cluster (clvmd)
- VGs migrieren: vgexport/vgimport

Weitere Feautures LVM

- Cachen aller Metadaten ber daemons (lvmetad und dmmetad)
- Cluster (clvmd)
- VGs migrieren: vgexport/vgimport
- LVs aktivieren / deaktivieren ("activation skip")

Weitere Feautures LVM

- Cachen aller Metadaten ber daemons (lvmetad und dmmetad)
- Cluster (clvmd)
- VGs migrieren: vgexport/vgimport
- LVs aktivieren / deaktivieren ("activation skip")

Zetabyte (2^{128} byte) File System

- Bekannt aus Solaris
- sehr stabil
- sehr sicher
- nicht das allerschnellste ;-)
- Alleinstellungsmerkmal Raid7 (bei ZFS: raidz3), sprich drei Parity-Devices (!)

Lizenzprobleme

- Problem: ZFS-Lizenz inkompatibel mit Kernel-Lizenz GPLv2
- Lösung:

Lizenzprobleme

- Problem: ZFS-Lizenz inkompatibel mit Kernel-Lizenz GPLv2
- Lösung:
  - ZFS-Code wird auerhalb der Kernels gepflegt

Lizenzprobleme

- Problem: ZFS-Lizenz inkompatibel mit Kernel-Lizenz GPLv2
- Lösung:
  - ZFS-Code wird auerhalb der Kernels gepflegt
  - Übersetzen gegen einen bestehenden Kernel ist lizenztechnisch unproblematisch

Lizenzprobleme

- Problem: ZFS-Lizenz inkompatibel mit Kernel-Lizenz GPLv2
- Lösung:
  - ZFS-Code wird auerhalb der Kernels gepflegt
  - Übersetzen gegen einen bestehenden Kernel ist lizenztechnisch unproblematisch
  - ".. is tainting kernel"-Meldung eher politischer Natur

Lizenzprobleme

- Problem: ZFS-Lizenz inkompatibel mit Kernel-Lizenz GPLv2
- Lösung:
  - ZFS-Code wird auerhalb der Kernels gepflegt
  - Übersetzen gegen einen bestehenden Kernel ist lizenztechnisch unproblematisch
  - ".. is tainting kernel"-Meldung eher politischer Natur

Lizenzprobleme

- Problem: ZFS-Lizenz inkompatibel mit Kernel-Lizenz GPLv2
- Lösung:
  - ZFS-Code wird auerhalb der Kernels gepflegt
  - Übersetzen gegen einen bestehenden Kernel ist lizenztechnisch unproblematisch
  - ".. is tainting kernel"-Meldung eher politischer Natur

Funktionsweise von ZFS

- zpool erzeugt einen Pool von Device-Blcken, der sofort als Dateisystem verfügbar ist
- zfs kann aus dem Pool auch Block-Devices erzeugen

Funktionsweise von ZFS

- zpool erzeugt einen Pool von Device-Blcken, der sofort als Dateisystem verfügbar ist
- zfs kann aus dem Pool auch Block-Devices erzeugen
- Snapshots sind möglich (ro)

Funktionsweise von ZFS

- zpool erzeugt einen Pool von Device-Blcken, der sofort als Dateisystem verfügbar ist
- zfs kann aus dem Pool auch Block-Devices erzeugen
- Snapshots sind möglich (ro)
- Clone machen Snapshots wieder schreibfähig

Funktionsweise von ZFS

- zpool erzeugt einen Pool von Device-Blcken, der sofort als Dateisystem verfügbar ist
- zfs kann aus dem Pool auch Block-Devices erzeugen
- Snapshots sind möglich (ro)
- Clone machen Snapshots wieder schreibfähig
- Demo

Funktionsweise von ZFS

- zpool erzeugt einen Pool von Device-Blcken, der sofort als Dateisystem verfügbar ist
- zfs kann aus dem Pool auch Block-Devices erzeugen
- Snapshots sind möglich (ro)
- Clone machen Snapshots wieder schreibfähig
- Demo

Weitere Feautures ZFS

- nix neues :-)

Noch Fragen?