Andreas Steinel

Andreas.Steinel@exirius.de
☎+49 6881 9999 5 0 | https://www.exirius.de

Speicherplatzoptimierte Backupstrategie

Intelligente Nutzung von ZFS für Proxmox VE Backups

Proxtalks 2017 - 25. Oktober 2017 - 13:45 bis 14:30

Abschnitt 1

Start und Motivation

Über mich

- Andreas Steinel, M.Sc. Visual Computing M.Sc. Informatik
- Linux seit 1998, Debian seit 2001
- arbeite seit 2006 für die eXirius IT Dienstleistungen GmbH
- Proxmox VE seit Version 3.4 in 2015
- seitdem auch im Forum mit Synonym `LnxBil` aktiv
- Aufstieg in die Top-Ten

Motivation

Warum das Ganze?

- Proxmox VE Backup: einfach (und) prima
- keine (schnellen) inkrementellen Backups möglich
- Backup daher riesig
- großer Speicherplatzbedarf oder geringe Anzahl
- großer Aufwand bei Off-Site-Backup

Was können wir da tun?

Abschnitt 2

Analyse

Einstieg

Was bietet Proxmox VE bzgl. Backup?

- funktioniert out-of-the-Box (klasse!)
- Backup über die GUI möglich (individuell oder zeitgesteuert)
- Aufruf via `vzdump` auf der Konsole
- erzeugt `*.vma` Dateien
- Kompression möglich - dann `*.vma.(gz|lzo|...)`
- VMA: PVE-eigenes Open-Source-Format, integriert in QEmu
- Speicherung lokal oder auch via NFS (generell: auf File-Storage)

Inkrementelle Backups

Drittanbieterlösung vorhanden:

- programmiert von Kamil Trzciński (`ayufan`)
- Patch für `vzdump`
- verwendet `xdelta3`
- Basiert auf einem Vergleich der VMA-Dateien
- gut, aber langsamerer Backup-Vorgang (Vervielfachung der Zeit)
- manuelles Patchen von PVE notwendig
- Basis- und differenzielles Backup für Wiederherstellung notwendig

Aktuell die einzige (mir bekannte) Lösung des Problems

Analyse VMA

Was genau macht VMA?

- VMA ist PVE-eigenes Open-Source-Format (s.o.)
- https://git.proxmox.com/?p=pve-qemu.git;a=blob;f=vma_spec.txt
- Backup verwendeter, nicht-leerer Blöcke
- Reihenfolge der Blöcke: sequenziell wie im Original

Optimierungsmöglichkeiten für VMs

- Speicherplatzbedarf kann durch “nullen” verbessert werden
- je nach virtuellem Blockstoragesystem auch via `TRIM`

Abschnitt 3

Exkurs ZFS

Historisches

Die (leider) nicht mehr existierende Firma SUN Microsystems

- veröffentliche die erste Version ZFS 2006 in Solaris 10
- Dateisystem für Server und Großrechenanlagen
- ZFS stand früher für Zettabyte File Systems, heute für alles mögliche
- sollte komplette Neuerfindung sein, was sogar gelang
- Integration von RAID-Funktionalität, Volume-Manager, Snapshots und CoW
- später sogar Kompression, Deduplizierung und Verschlüsselung
- Open Source unter Common Development and Distribution License (CDDL) in OpenSolaris
- Entwicklungsende von OpenSolaris in 2010 durch Oracle
- Weiterführung als OpenSource in OpenZFS

Seit Proxmox VE 3.4 integriert aus dem Projekt `OpenZFS-on-Linux` (`ZoL`).

Theoretische Maxima

Theoretisch, da in absehbarer Zeit niemand dieses Maximum erreichen wird!

- Dateisystemgröße von 256 Zebibyte (2^{78} Bytes)
- Dateigröße von 16 Exbibyte (2^{64} bytes)
- 2^{64} Platten in einem Storage-Pool
- Anzahl Dateien “beschränkt” auf 2^{48} pro Verzeichnis
- Größe der Attribute “beschränkt” auf Größe einer Datei
- Anzahl Attribute “beschränkt” auf Anzahl der Dateien
- Gesamtanzahl im Dateisystem unbegrenzt
- Dateinamenlänge von `255` ASCII Zeichen (`UTF-8` ggf. weniger)
- unbegrenzte Snapshots
- kein(!) Cluster-Dateisystem

Seth Lloyd, “Ultimate physical limits to computation.” Nature 406, 1047-1054 (2000)

Terminologie

Natürlich gibt es auch eine ZFS-spezifische Terminologie

- Storage-Pool - größte Einheit, die alles vereint
- virtuelle Geräte (`vdev`) - physikalische Geräte unter einer “Zusammenlegungsvorschrift”
- Filesystem - Teil eines Storage-Pools (mit Mountpoint) als Dateidienst, im Baum organisiert
- Volume - Teil eines Storage-Pools (ohne Mountpoint) als Blockdienst
- Snapshot - Inhalt, zu einem bestimmten Zeitpunkt eingefroren
- Clone - klonen eines Snapshots und verwenden als CoW-System
- Scrubbing - durchsuchen der Platte nach Fehlern und heilen derselben (im Hintergrund, wie ECC)
- Resilvering - Neuaufbau einer ersetzten Platte (nur belegter Speicher)
- Export/Import - Senden kompletter, aber auch inkrementeller Online-Dateisysteme in PIPE-Manier

Verbesserungen gegenüber vorher

Wichtige Verbesserungen sind:

- einführen von Checksummen - Detektion von “Silent-Data-Corruption” (SDC)
- somit ist kein FSCK mehr notwendig, und das Dateisystem heilt sich selbst
- mehr als eine Kopie der Daten pro Festplatte möglich (Selbstheilung mit nur einer Platte)
- variable Blockgröße (Standard: 128K)
- generell immer Thin-Provisioning
- freier Speicherplatz nicht pro Dateisystem/Volume belegt, sondern pro Pool
- komplett paralleler Zugriff auf alles

Weitere Verbesserungen

Außerdem:
Einführung von RAID-Z1 bis RAID-Z3 als Software-RAID:

- setzt keine spezielle Hardware voraus und ist somit wesentlich günstiger
- bringt das `I` in RAID zurück - inexpensive
- Selbstheilung on-the-fly beim Lesen eines korrupten Blocks

Zusätzlich:

- Integration aller Administrationstools in zwei Programme `zpool` und `zfs`
- Adaptive Endianess - egal, wo es läuft, es klappt immer
- Platten werden komplett integriert, und ZFS kümmert sich um GPT, Partionen usw.
- Backupzeiten dank inkrementeller Snapshots extrem verkürzt

Wichtige Kommandos

Für uns heute wichtige Kommandos sind:

- anlegen eines ZFS Dateisystems `1000` unter `rpool/proxmox`:

  ```
  $ zfs create rpool/proxmox/1000
  ```

- anlegen eines Snapshots namens `2017-10-26_13-45-00`:

  ```
  $ zfs snapshot rpool/proxmox/1000@2017-10-26_13-45-00
  ```

- klonen Dateisystem nach `1000-clon`:

  ```
  $ zfs clone rpool/proxmox/1000@2017-10-26_13-45-00 \
    rpool/proxmox/1000-clon
  ```

- zerstören des geklonten Dateisystems:

  ```
  $ zfs destroy rpool/proxmox/1000-clon
  ```

- Größe eines Snapshots:

  ```
  $ zfs send -i #{last} #{current} | wc --bytes
  ```

proxtalks

Deduplizierung

Gleiche Blöcke werden nur einmal gespeichert

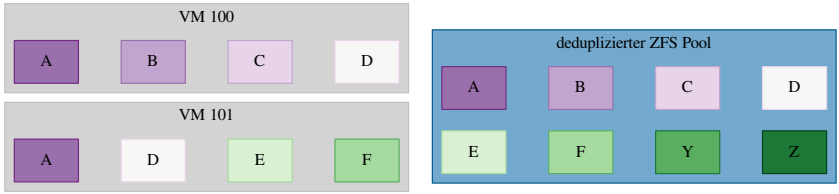

Abbildung: Deduplizierte Blöcke

Probleme mit Deduplizierung

Was ist ein Block, und wie kann man ihn beeinflussen?

- Blockgröße von bis zu 128K ungeeignet für VMs

Geeignete Blockgröße ist abhängig vom Gastsystem

- meist 4K Blöcke (ZFS-Einstellung `recordsize`)
- Alignment wichtig (Partitionsgrenzen)
- Fragmentierung

Somit leider:

- sequentielles Schreiben der VMA Datei kann alles kaputt machen
- 4K Blöcke für Deduplizierung erhöhen RAM-Bedarf drastisch
- Deduplizierung nur schnell, wenn der DDT in den RAM passt

Abschnitt 4

Zurück zum Backup

Was steckt in der VMA drin?

Entpacken via `vma extract`, VMA beinhaltet:

- Festplattenabbild im Format `RAW`
- VM Konfiguration
- Firewall Konfiguration
- (potentiell noch mehr)

Rohes Festplattenabbild klingt doch mal spannend:

- `sparse file`
- Platte 1:1 wie in Proxmox VE

Wieder verpacken: analog via `vma create`.

Beispiel

Beispieldatei (VM 9193 vom 16.10.2017 um 23:17:20):

```
$ vma extract vzdump-qemu-9193-2017_10_16-23_17_20.vma out
DEVINFO out/tmp-disk-drive-scsi0.raw 104857600
Formatting 'out/tmp-disk-drive-scsi0.raw', fmt=raw size=104...

$ ls -hl out/
insgesamt 8
-rw-r--r-- 1 root root 100M Okt 20 14:38 disk-drive-scsi0.raw
-rw-r--r-- 1 root root  380 Okt 20 14:38 qemu-server.conf
```

Was machen wir jetzt damit?

Idee für besseres Backup

Ein intelligentes Backup mit folgenden Dingen:

- CoW-Features von ZFS
- extrahiertes Backup
- ZFS-Snapshots

Jetzt “nur” noch kombinieren, bzw. herausfinden was wir nach dem ersten Extrahieren machen müssen!

Einfache Lösung via `cp` oder `dd`

Einfaches Kopieren via `cp`

- es gibt `--reflink=always`, was nur Änderungen schreiben soll
- klappt nur, wenn es ein Dateisystem ist und leider nicht auf ZFS

oder stupides Schreiben mit `dd`

- kopiert Quelle in Ziel (genau was wir brauchen)
- überschreibt alle Blöcke

Leider Seiteneffekte auf CoW Systemen

- Copy-on-Write, und schreiben tun wir
- erneutes Schreiben unveränderter Blöcke

Somit leider nicht anwendbar.

Synchronisierung der Daten

Nächstes Werkzeug aus der Kiste:

- natürlich `rsync`
- synchronisiert Dateien und Ordner (auch inkl. Löschen)

Korrekte Parameter (via `trial-and-error`):

- `--inplace` (sonst Kopie der Datei)
- `--no-whole-file` (sonst wie `dd`)

Anwendbarkeit?

- funktioniert prima, meistens zumindest
- leider teilweise extrem langsam (Zeiten > 48h)

Blockbasierte Synchronisierung

Idee eines eigenen Programms:

- eine Art intelligentes `dd`
- kleines C-Programm (<100 Zeilen)
- 4K Blöcke aus Quelle und Ziel

Gewählte Einschränkung, dass Dateigrößen identisch sein müssen.

- Lesen eines Blocks, wenn unterschiedlich in Ziel schreiben, sonst zum nächsten Block

Genau was wir brauchen, schnell und einfach, jedoch nur anwendbar, wenn wir bereits ein Ziel haben und Dateigrößen identisch, daher kombiniert mit `rsync`.

Abschnitt 5

ZFS-basiertes Backup

Genereller Aufbau VM-Backup

Idee, jede VM eigenständig zu behandeln:

- eigenes ZFS Dateisystem
- Snapshot pro Backup-Datei
- inkrementelle Übertragung möglich
- natürlich auch einzeln löschbar

Genereller Aufbau Backupserver

Was muss man am Backup ändern?

- generell nichts, wenn man bereits NFS verwendet

Ein extra Server mit ZFS als Dateisystem

- NFS-Server für Backup-Storage Richtung PVE
- Skripte zum Einladen der Backups

Potentielle Verzeichnisstruktur

Im folgendem Beispiel gehen wir von folgender Dateisystemstruktur aus:

- NFS-Server mit Export `/nfs_export`, in dem ein Unterordner `dump` existiert, in dem die PVE Backups abgelegt werden.
- ZFS Dateisystem `rpool/proxmox` unter `/rpool/proxmox/`, in dem die VM-Backups abgelegt werden.

Als Betriebssystem kann natürlich Proxmox VE verwendet werden, möglich ist aber jedes System, das

- ZFS verwenden kann und
- auf dem `vma` lauffähig ist (z.B. alle Debian-basierten Linux-Distributionen)

Im Beispiel hier verwenden wir Proxmox VE.

Vorgehen des Erstimports

Wenn eine neue VM gebackupt werden soll, bzw. noch kein ZFS Dateisystem existiert:

- Extraktion `ID` und `Datum` aus Backup-Datei
- Anlage ZFS Dateisystem mit Namen `ID`
- Extraktion VMA auf das ZFS Dateisystem mit Namen `ID` (`rpool/proxmox/<ID>`)
- kopieren Backup-Logdatei auf das ZFS Dateisystem
- Anlage Snapshot mit `Datum`

```
$ zfs snapshot rpool/proxmox/<ID>@<Datum>
```

ggf. Löschen der Eingabe-VMA-Datei

Vorgehen nach dem Erstimport

Wenn bereits ein ZFS Dateisystem existiert:

- Extraktion `ID` und `Datum` aus Backup-Datei
- Extraktion VMA in einen temporären Ordner außerhalb
- Synchronisierung der extrahierten VMA auf das ZFS Dateisystem
- Anlage Snapshot mit `Datum`
- löschen der temporären Dateien

ggf. Löschen der Eingabe-VMA-Datei

Beispiel

Wie könnte das aussehen?

```
$ zfs list -t all -r -o name,use...ratio rpool/proxmox/1000
NAME                                      USED  REFER  RATIO
rpool/proxmox/1000                       4,57G  2,82G  2.57x
rpool/proxmox/1000@2017_03_03-18_30_01   81,7M  1,88G  1.95x
rpool/proxmox/1000@2017_04_28-18_30_02   72,6M  1,99G  1.92x
rpool/proxmox/1000@2017_06_30-18_30_02    123M  2,39G  1.92x
rpool/proxmox/1000@2017_07_14-18_30_01    118M  2,43G  1.93x
rpool/proxmox/1000@2017_08_04-18_30_02   47,2M  2,37G  1.86x
rpool/proxmox/1000@2017_08_18-18_30_02    108M  2,62G  1.87x
rpool/proxmox/1000@2017_09_01-18_30_02    110M  2,66G  1.88x
rpool/proxmox/1000@2017_09_08-18_30_02    113M  2,67G  1.89x
rpool/proxmox/1000@2017_09_15-18_30_02    120M  2,72G  1.89x
rpool/proxmox/1000@2017_09_22-18_30_02       0  2,74G  1.89x
```

proxtalks

Vorgehen bei VM-Wiederherstellung

Der Zustand einer VM zum Zeitpunkt `T` soll wiederhergestellt werden

- klonen des Snapshots zum Zeitpunkt `T` als ZFS-Dateisystem `X`
- extrahieren der VM-Diskkonfiguration

```
$ grep '#qmdump' qemu-server.conf
#qmdump#map:virtio0:drive-virtio0:san-lvm-fast:raw:
```

- zusammenbauen der VMA in NFS-Freigabeordner via

```
$ vma create /nfs-export/dump/vmdump-qemu-<ID>-<X>.vma \
  -c qemu-server.conf 'drive-virtio0=disk-drive-virtio0.raw'
```

- ggf. noch komprimieren

```
$ lzop -U /nfs-export/dump/vmdump-qemu-<ID>-<X>.vma
```

- zerstören des ZFS-Dateisystems `X`

Extrahieren von Dateien einer VM

Man kann auch einzelne Dateien aus einem Backup extrahieren:

- klonen des Snapshots zum Zeitpunkt `T` als ZFS-Dateisystem `X`
- verfügbar machen der Partitionen vom virtuellen Dateisystem

```
$ kpartx -a /rpool/proxmox/<X>/data/disk-drive-virtio1.raw
add map loop1p1 (252:2): 0 1610610688 linear /dev/loop1 2048
```

- einhängen unter `/mnt/recovery` (ggf. anlegen)

```
$ mount /dev/mapper/loop1p1 /mnt/recovery
```

- dann entsprechend arbeiten in `/mnt/recovery`
- aushängen

```
$ umount /mnt/recovery; kpartx -d /dev/loop1
```

- ZFS-Dateisystem `X` zerstören

proxtalks

Ein paar Zahlen

VM-Import:

- Debian Stretch, 32 GB Disk, GitLab-Instanz
- `vzdump` in knapp unter 2 Minuten durch (320 MB/sec)
- Import ins ZFS: insgesamt 7,5 Minuten

wobei sich diese Zeit aufteilt in:

- 3,5 Minuten VMA entpacken
- 4 Minuten Datei auf ZFS synchronisieren
- atomarer Snapshot
- wenige Sekunden für das Löschen temporärer Dateien

Import läuft via Cron an

Einsparpotential (täglich)

Einsparung beim täglichen Backup ist enorm (Daten in MB):

Datum	vDisk	VMA.lzop	dZFS	Einsparung
17-09-28	34.359	1.495	2.411	-60 %
17-10-02	73.012	16.261	1.487	91 %
17-10-03	90.191	20.418	1.362	94 %
17-10-04	90.191	20.551	1.019	96 %
17-10-05	90.191	21.258	1.047	96 %
17-10-09	90.191	22.108	2.096	91 %
17-10-10	90.191	26.193	5.894	78 %
17-10-11	90.191	26.234	1.551	95 %
17-10-12	90.191	26.296	1.255	96 %
17-10-16	90.191	42.639	2.680	94 %
17-10-17	90.191	42.577	947	98 %
17-10-18	90.191	42.516	1.553	97 %
17-10-19	90.191	42.496	1.445	97 %

proxtalks

Einsparpotential (wöchentlich)

Einsparung ist immer noch groß, jedoch nicht in der gleichen Klasse:

Datum	vDisk	VMA.lzop	dZFS	Einsparung
17-09-15	1.009.282	322.195	59.773	82 %
17-09-22	1.065.113	329.495	48.574	86 %
17-09-29	1.056.528	339.904	85.338	75 %
17-10-06	1.365.757	429.057	106.438	76 %
17-10-13	1.640.633	468.588	135.476	72 %
17-10-20	1.501.048	468.283	49.856	90 %

Und die Zahlen aus dem ZFS sind nicht komprimiert!

Abschnitt 6

Abschließendes

Aufräumarbeiten

Generell sollte man sich Gedanken machen über

- Zeiträume, über die man Snapshots aufheben möchte (“Ausdünnung”)
- Mehrfache externe Replikation muss koordiniert sein!
- Müssen alle Daten über PVE gesichert werden (z.B. ZFS in VM)?
- Kann man die Daten CoW-freundlicher in der VM ablegen?
- Verschlüsselung (LUKS, bald direkt in ZFS)

Was ist eigentlich mit Containern?

Gleiche Logik wie bei “richtigen” VMS ist anwendbar, jedoch folgende Unterschiede:

- direktes Entpacken möglich, da `tar` Archiv (`vma` fällt weg)
- `zfs diff` sinnvoll einsetzbar
- Wiederherstellen wird ohne `vma` auch einfacher
- einzelne Dateien wiederherzustellen ebenfalls direkt möglich

Haben wir mangels Anwendungsfall bisher noch nicht implementiert, da unsere Container gesondert in einem ZFS liegen, das bereits mit ZFS-internen Mitteln gesichert wird.

Abschluss

Vorteile:

- automatischer Test des Backups
- nur Differenz, daher inkrementell übertragbar
- Sicherungsmaßnahmen aktiv (`silent-data-corruption`)
- Dateiwiederherstellung sehr schnell möglich

Nachteile:

- nachgelagertes Entpacken notwendig
- erneutes Zusammenpacken als VMA nötig, um es einspielen
- langsamerer Gesamtsicherungsprozess, aber asynchron
- ZFS hat höhere Hardwareanforderungen (z.B. RAM)

Andreas Steinel

Andreas.Steinel@exirius.de
☎+49 6881 9999 5 0 | https://www.exirius.de

Vielen Dank für ihre Aufmerksamkeit