Lernkurven, Stolpersteine und andere Hürden
Erfahrungsbericht einer Ceph-Cluster Implementation

Roter Faden:

Vorstellung

Was ist Ceph? Eine Übersicht

Was wird für ein Ceph-Cluster benötigt und wie funktioniert es?

Wo liegen wie die Daten in Ceph? in Kürze für Admins

Szenario - warum wir Ceph gewählt haben

Wie schlägt sich Ceph im Vergleich zum „alten“ Storage

Die Tücken im Detail

Monitoring

Performance Tuning und Messungen

<udo.lembke@albertbauer.com>

Vorstellung

Seit knapp 17 Jahren Admin bei Albert Bauer Companies
in einem dreiköpfigen Adminteam

Albert Bauer Companies in Kürze:

- Gründung 1960 als Albert Bauer KG, Bereich Druckvorstufe
- Inzwischen sechs Unternehmensbereiche: CREATION, PRINT, STUDIOS, DIGITAL, MEDIEN-IT und PACKAGING
- Zwei Standorte: Hamburg und München
- Über 200 Mitarbeiter

ALBERT BAUER
COMPANIES

Chemnitzer Linux-Tage
21. und 22. März 2015

Was ist Ceph?
Eine Übersicht

Ceph ist ein Object Store, der in drei Modi betrieben werden kann:

- als Object Store - via RadosGW Amazon S3 kompatibel
- als Ceph Filesystem (cephfs)
- als Ceph Block Device - ehemals Rados Block Device (rbd) genannt

Vorteile:

- Open Source
- Sehr gut skalierbar - bis in den ExaByte Bereich
- Ausfallsicher durch Redundanz und „Selbstheilung“
- Verwendung von Standard-Hardware - kein Vendor Lock

* der Vortrag behandelt nur die Nutzung als Ceph Block Device

Was wird für ein Ceph-Cluster benötigt
und wie funktioniert es?

Monitor Nodes

- wegen Ausfallsicherheit mindestens drei Stück
- die Mon-Nodes halten die aktuellen Cluster Maps (Monitor-, OSD-, PG-, CRUSH- und MDS-Map)

Storage Nodes

- aus Redundanzgründen mindestens drei Stück
- je Storage Node 1-n OSDs (Object Storage Device)
- je OSD ein Journal (entweder als Partition auf der OSD-Disk oder auf gesonderter SSD)
- OSDs mit SAS-Controller (nicht RAID) ansteuern
- bevorzugt zwei Netzwerkanbindungen (eins für Client/Mon-Traffic und eins zur Replikation der OSD-Daten)

Client(s) (in unserem Fall rbd-Zugriff via QEMU)

Optional eine Admin-Node zum Verwalten (ceph-deploy)

Wie funktioniert Ceph?
in Kürze für Admins

Aufbau einer OSD

```
/var/lib/ceph/osd/ceph-21/
├── activate.monmap
├── active
├── ceph_fsid
├── current
│   ├── 18.0_head
│   │   └── DIR_0
│   │       └── DIR_0
│   │           ├── DIR_6
│   │           │   ├── DIR_4
│   │           │   │   ├── rbd\udata.30f9ff82ae8944a.00000000000003ae__head_BD8E4600__12
│   │           │   │   ├── rbd\udata.312036a2ae8944a.00000000000005ab__head_FF834600__12
│   │           │   │   ├── rbd\udata.312d6d32ae8944a.0000000000000687__head_03D44600__12
│   │           │   │   ├── rbd\udata.33629be2ae8944a.00000000000002de2__head_E7B64600__12
│   │           │   │   └── ...
│   │           │   └── DIR_5
│   └── omap
│       ├── 008652.sst
│       ├── 008708.sst
│       └── ...
├── fsid
├── journal -> /dev/disk/by-partlabel/journal-21
├── keyring
├── lost+found
├── magic
├── ready
├── store_version
├── superblock
├── sysvinit
└── whoami
8952 directories, 1020000 files
```

```
$ cat /etc/ceph/ceph.conf
[global]
fsid = 591db070-15c1-4c7a-b107-67717bdb87d9
...
```

```
[osd.21]
        key = AQCuN4BU4FjPBhAAt0PqKXetFtOSDdbdSE46ZQ==
```

```
21
```

Wie funktioniert Ceph?
in Kürze für Admins

Aufbau einer OSD

Der Client stückelt seine Daten in 4MB-Objekte (fragt den Ceph-Monitor nach der aktuellen crush-map und berechnet daraus, wohin die Files zu schreiben sind)

```
$ rbd info pve_pool/vm-140-disk-1
rbd image 'vm-140-disk-1':
        size 10240 MB in 2560 objects
        order 22 (4096 kB objects)
        block_name_prefix: rbd_data.312036a2ae8944a
        format: 2
        features: layering

$ rados ls -p pve_pool | grep rbd_data.312036a2ae8944a
rbd_data.312036a2ae8944a.00000000000004b0
rbd_data.312036a2ae8944a.00000000000005ab
...

$ ceph osd map pve_pool rbd_data.312036a2ae8944a.00000000000005ab
osdmap e128513 pool 'pve_pool' (18) object 'rbd_data.312036a2ae8944a.00000000000005ab' -> pg
18.ff834600 (18.0) -> up ([21,29,48], p21) acting ([21,29,48], p21)

/var/lib/ceph/osd/ceph-21/
├── current
│   ├── 18.0_head
│   │   └── DIR_0
│   │       └── DIR_0
│   │           ├── DIR_6
│   │           │   ├── DIR_4
│   │           │   │   ├── rbd\udata.312036a2ae8944a.00000000000005ab__head_FF834600__12
```

Wie funktioniert Ceph?
in Kürze für Admins

Aufbau einer OSD

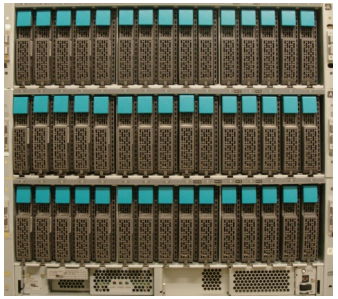
Der Client stückelt seine Daten in 4MB-Objekte (fragt den Ceph-Monitor nach der aktuellen crush-map und berechnet daraus, wohin die Files zu schreiben sind)

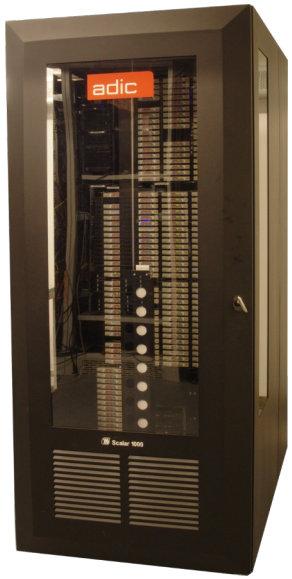
Die 4MB-Objekte werden „wild“ über die OSDs verteilt

```
Anlegen einer 10GB VM-Disk erzeugt erstmal nur eine header-Datei
$ rbd image 'vm-990-disk-1':
        size 10240 MB in 2560 objects
        order 22 (4096 kB objects)
        block_name_prefix: rbd_data.35cca2f2ae8944a
        format: 2
        features: layering


$ rados ls -p pve_pool | grep 35cca2f2ae8944a
rbd_header.35cca2f2ae8944a


Nach parted
$ rados ls -p pve_pool | grep 35cca2f2ae8944a
rbd_data.35cca2f2ae8944a.00000000000009ff
rbd_header.35cca2f2ae8944a
rbd_data.35cca2f2ae8944a.0000000000000000

Komplett mit Daten gefüllt werden alle 2560 Objects verwendet. Als Primarys fungieren 70/84 OSDs
- alle Replicas liegen auf allen 84 OSDs!
```

Szenario - wo wir her kamen

bunter Storage-Mix an verschiedenen FC-Raids





VMware ESX Cluster

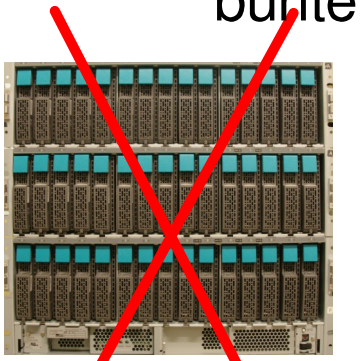
Bandarchiv mit HSM-SW

SUN Produktionsserver

Proxmox VE Cluster

Szenario - wo wir her kamen

bunter Storage-Mix an verschiedenen FC-Raids



VMware ESX Cluster

Bandarchiv mit HSM-SW

SUN Produktionsserver

PROXMOX

Proxmox VE Cluster

Szenario - wo wir hin wollen

„Massen“-Storage via Ceph

„Sonderfälle“ mit einem FC-Raid

Erweiterung des PVE Clusters auf 5 Nodes

Intern mit DRBD-Storage (SAS/SSD)

Proxmox VE Cluster

Vergleich FC-Raid mit Ceph-Storage
mit 48TB netto

Bei einem FC-Raid mit raidlevel-6 werden 15 4TB-Platten benötigt:
(15 - 2) * ~3.7TB = ~48TB

- 1 Raid mit 3HE Platz
- gute single thread Performance
- durch Dual-Controller relativ ausfallsicher
- Kosten < 12k€

Mit Ceph sind es für die gleiche Datenmenge 60 4TB-Platten:
(60 * ~3.7TB) / 3 * 65% = ~48TB

- 5 Nodes mit 15HE Platz
- mäßige single thread Performance
- höhere Read-IOPS durch mehr Spindeln
- theoretisch sehr ausfallsicher, kein SPOF, „selbstheilend“
- Kosten < 50k€

Und nun?
Rotstift ansetzen!

Sparpotentiale ausschöpfen

- Server selbst aufgebaut 
- erstmal mit nur 16GB RAM pro OSD-Node gestartet 
- Vermeidung von teuren Enterprise-SSDs für's Journal 
- nur eine Journal-SSD für 12 OSDs 
- günstige AMD FX-8350 CPUs 

Erkenntnisse während des Betriebs

Nicht alle SSDs eignen sich als Journal-SSD

Filebasiertes Journal sollte vermieden werden (und erst recht auf LVM-Storage)

ceph-deploy ist das Mittel der Wahl zum Erstellen der OSDs

Aufteilung in mehreren Pools! Richtige Wahl der PGs (können nicht reduziert werden)

XFS sollte von Anfang an mit den richtigen Mount-Options verwendet werden
(wir hatten bis zu 20% Fragmentation nach weniger als einem Jahr)

ext4 performt für uns besser (ergab ungefähr Halbierung der Latenz)

Füllgrad der OSDs beachten (bis zu 20% Unterschied)

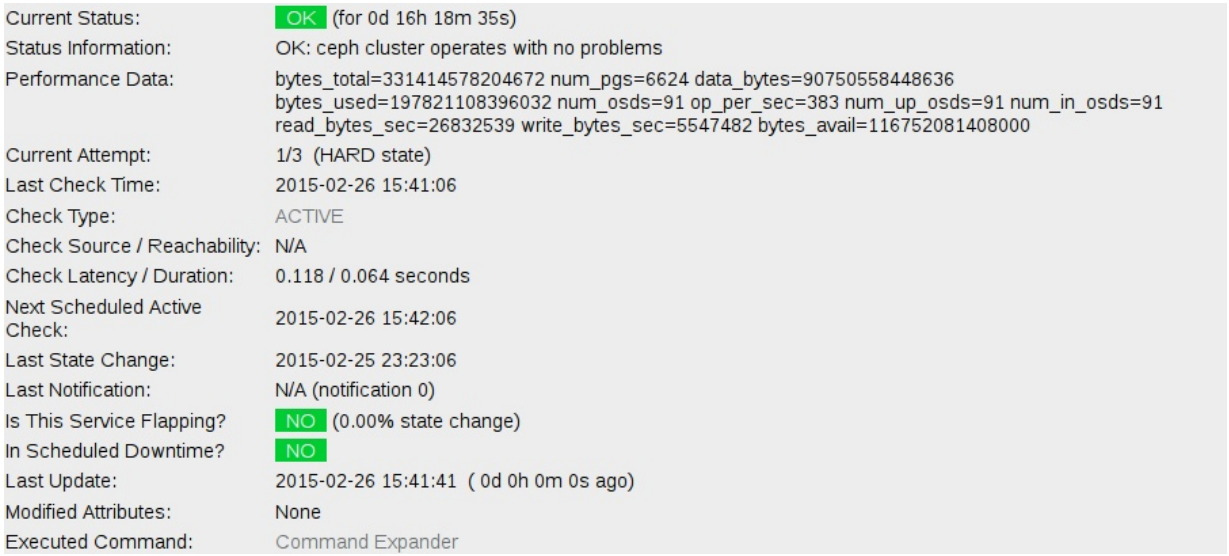
Begrenzung der Recovery-Jobs (damit der normale Betrieb während Recovery nicht zu sehr leidet)

Bei EC-Pools Größe und Objekte im vorgeschalteten Cache Tier definieren

Monitoring

ist ein Muss

- Ceph-Health



Current Status:	OK (for 0d 16h 18m 35s)
Status Information:	OK: ceph cluster operates with no problems
Performance Data:	bytes_total=331414578204672 num_pgs=6624 data_bytes=90750558448636 bytes_used=197821108396032 num_osds=91 op_per_sec=383 num_up_osds=91 num_in_osds=91 read_bytes_sec=26832539 write_bytes_sec=5547482 bytes_avail=116752081408000
Current Attempt:	1/3 (HARD state)
Last Check Time:	2015-02-26 15:41:06
Check Type:	ACTIVE
Check Source / Reachability:	N/A
Check Latency / Duration:	0.118 / 0.064 seconds
Next Scheduled Active Check:	2015-02-26 15:42:06
Last State Change:	2015-02-25 23:23:06
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	2015-02-26 15:41:41 (0d 0h 0m 0s ago)
Modified Attributes:	None
Executed Command:	Command Expander

`https://github.com/Crapworks/ceph-dash/`

Monitoring

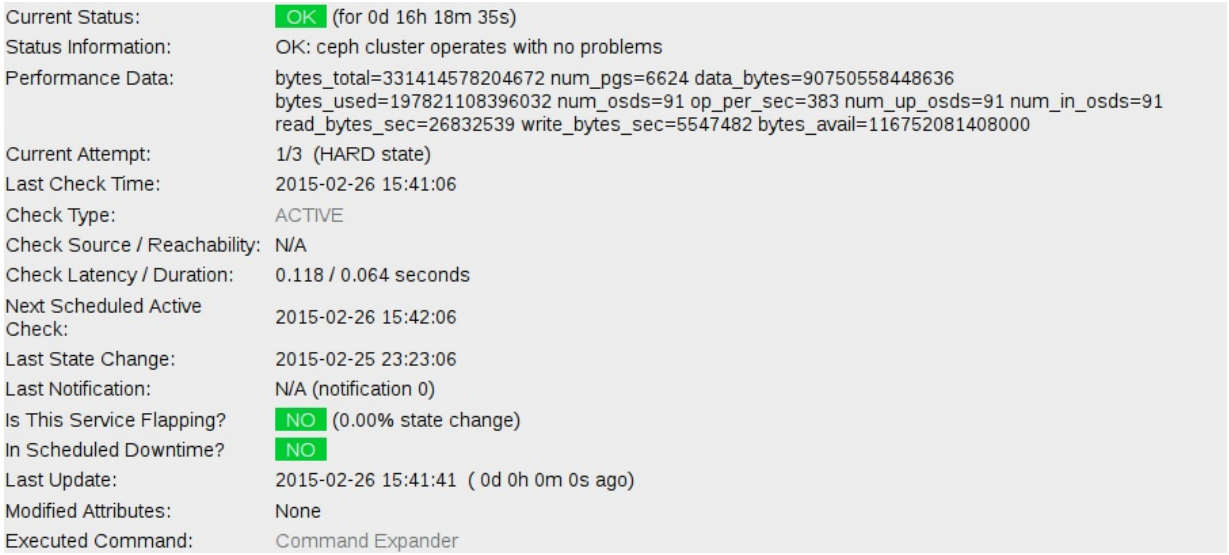

Ceph Dashboard
Ceph Cluster Overall Status
Cluster Status:HEALTH_OK
Ceph Monitor Status
Monitor A: HEALTH_OK
Monitor B: HEALTH_OK
Monitor C: HEALTH_OK
Ceph Cluster OSD Status
Total
63
In
63
Up
63
Unhealthy
0
Ceph Cluster Placement Group Status
Storage
Cluster storage utilization
84.20 TB / 215.28 TB (39%)
Write / second
162.41 MB
Read / second
344.44 MB
Operations / second
2185
PG Status
5088 placementgroups in cluster


Ceph Dashboard
Ceph Cluster Overall Status
Cluster Status:HEALTH_WARN
1 pgs recovering
28 pgs recovery_wait
29 pgs stuck unclean
recovery 6198/40634990 objects degraded (0.015%)
Ceph Monitor Status
Monitor A: HEALTH_OK
Monitor B: HEALTH_OK
Monitor C: HEALTH_OK
Ceph Cluster OSD Status
Total
63
In
63
Up
63
Unhealthy
0
Ceph Cluster Placement Group Status
Storage
Cluster storage utilization
83.85 TB / 215.28 TB (39%)
Write / second
175.65 MB
Read / second
350.81 MB
Operations / second
2476
PG Status
5088 placementgroups in cluster
Recovering bytes / second
21.52 MB
Recovering keys / second
0
Recovering objects / second
5

Monitoring

ist ein Muss

- Ceph-Health



Current Status:	OK (for 0d 16h 18m 35s)
Status Information:	OK: ceph cluster operates with no problems
Performance Data:	bytes_total=331414578204672 num_pgs=6624 data_bytes=90750558448636 bytes_used=197821108396032 num_osds=91 op_per_sec=383 num_up_osds=91 num_in_osds=91 read_bytes_sec=26832539 write_bytes_sec=5547482 bytes_avail=116752081408000
Current Attempt:	1/3 (HARD state)
Last Check Time:	2015-02-26 15:41:06
Check Type:	ACTIVE
Check Source / Reachability:	N/A
Check Latency / Duration:	0.118 / 0.064 seconds
Next Scheduled Active Check:	2015-02-26 15:42:06
Last State Change:	2015-02-25 23:23:06
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	2015-02-26 15:41:41 (0d 0h 0m 0s ago)
Modified Attributes:	None
Executed Command:	Command Expander

- SSD-Lifetime
- eventuell „`ceph osd set noout`“ wenn eine Storage-Node down ist

Performance Tuning

Parameter in der ceph.conf setzen, oder „on the fly“ mittels injectargs
Aktive Werte mittels Admin-Socket auslesen
die Schreib-Performance beeinflusst die Lese-Performance
read-ahead im Client anpassen!!
Scrubbing „entschärfen“

Messungen + Fehlersuche
wer misst, misst Mist

Messungen zum Teil schwierig (da durch verschiedenes caching, parallele Zugriffe und Scrubbing die Werte (stark) schwanken)

Fehlersuche nicht einfach, da es ein komplexes Zusammenspiel ist

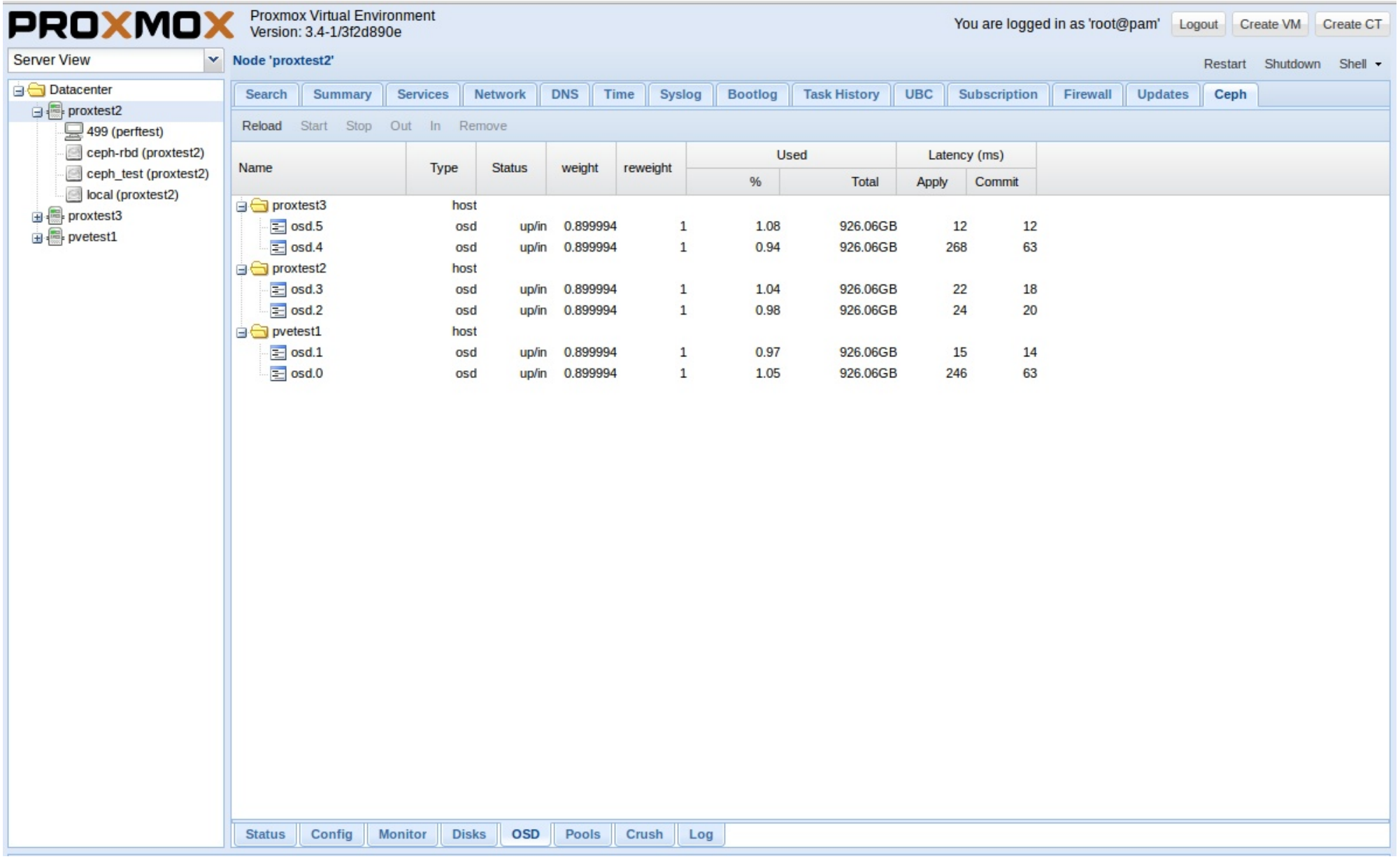
Linux-Bordmittel

- atop
- iperf
- ping
- fio (innerhalb der VM)

Ceph-tools

- osd perf/bench
- rados bench

zum Ausprobieren



http://proxmox.com/downloads
http://pve.proxmox.com/wiki/Ceph_Server

noch Fragen?

Anhang

http://www.sebastien-han.fr/blog/2014/10/10/ceph-how-to-test-if-your-ssd-is-suitable-as-a-journal-device/

https://ceph.com/pgcalc/

```
osd_mount_options_xfs = "rw,noatime,inode64,logbsize=256k,delaylog,allocsize=4M"
filestore_xfs_extsize = true

osd_mkfs_type             = ext4
osd_mount_options_ext4    = "user_xattr,rw,noatime"
osd_mkfs_options_ext4     = "-J size=1024 -E lazy_itable_init=0,lazy_journal_init=0"
filestore_xattr_use_omap = true

$ ceph osd reweight 2 0.8
$ ceph osd reweight-by-utilization

$ ceph osd tree
# id    weight  type name       up/down reweight
-1      300.5   root default
-3      300.5           rack unknownrack
-2      42.96                   host ceph-01
0       3.58                            osd.0   up      1
1       3.58                            osd.1   up      0.9065
2       3.58                            osd.2   up      0.8
3       3.58                            osd.3   up      0.7788
4       3.58                            osd.4   up      1
...

osd_max_backfills = 1
osd_recovery_max_active = 1

$ ceph osd pool set ssd-archiv target_max_bytes 53687091200  # 50GB
$ ceph osd pool set ssd-archiv target_max_objects 750000
```

Anhang

```
[osd]
osd_op_threads = 4
osd_disk_threads = 1

$ ceph tell osd.* injectargs '--osd_op_threads 4'

$ ceph --admin-daemon /var/run/ceph/ceph-osd.2.asok config show | grep threads
  "osd_op_threads": "4",
  "osd_disk_threads": "1",
  "osd_recovery_threads": "1",
  "osd_op_num_threads_per_shard": "2",
  "filestore_op_threads": "4",
  "keyvaluestore_op_threads": "2",
  "internal_safe_to_start_threads": "true"}

$ echo 4096 > /sys/block/vdb/queue/read_ahead_kb

osd_disk_thread_ioprio_class  = idle
osd_disk_thread_ioprio_priority = 7
```

Anhang

```
$ echo 3 > /proc/sys/vm/drop_caches

# rbd_cache löschen durch aus-/einschalten der VM

$ ceph osd set noscrub
$ ceph osd set nodeep-scrub


$ ping -s 8972 -M do 192.168.3.12
PING 192.168.3.12 (192.168.3.12) 8972(9000) bytes of data.
ping: local error: Message too long, mtu=1500

$ fio --numjobs=1 --readwrite=write --blocksize=4M --size=8G \
      --ioengine=libaio --direct=1 --name=fiojob

$ ceph osd perf
osd fs_commit_latency(ms) fs_apply_latency(ms)
  0                     0                    1
  1                     0                 1358
  2                     0                  834
  3                     0                    1
  4                     0                 1291
  5                     0                    0
...

$ ceph tell osd.1 bench
{ "bytes_written": 1073741824,
  "blocksize": 4194304,
  "bytes_per_sec": "104972434.000000"}

$ rados -p test bench 60 write --no-cleanup
$ rados -p test bench 60 seq   --no-cleanup
$ rados -p test cleanup benchmark_data
```