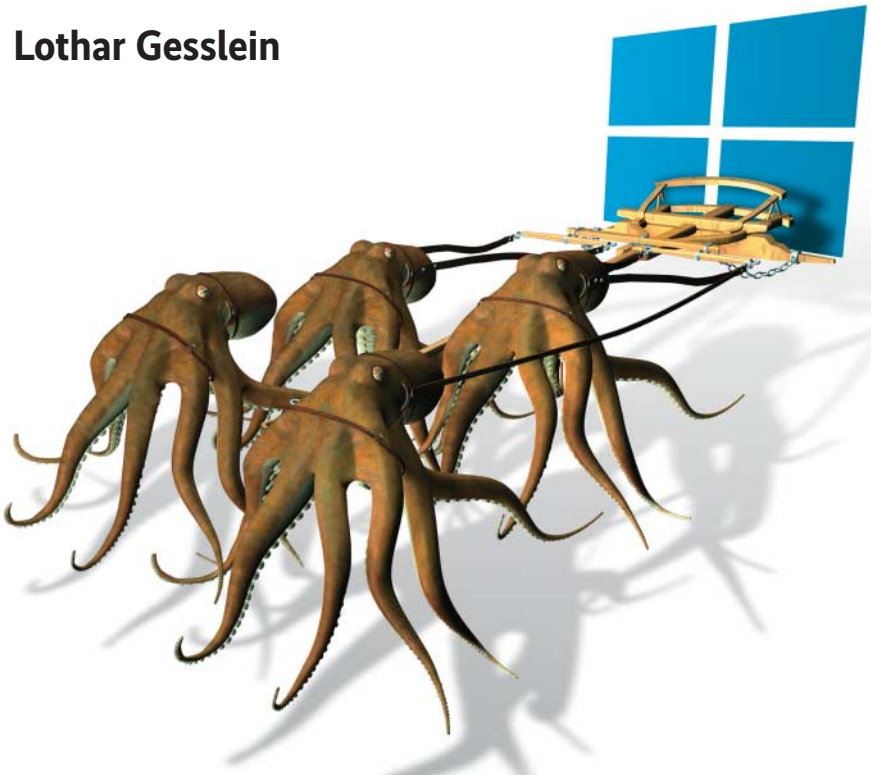
Ceph als iSCSI-Datastore für VMware

Starkes Gespann

Lothar Gesslein



Ein Einsatz als Cloud-Storage ist für Ceph ein Brot-und-Butter-Geschäft. Mit der richtigen Konfiguration lässt sich damit eben auch eine günstige Alternative zu EMC, NetApp und Co. realisieren.

Standardhardware kombiniert mit intelligenter Storage-Software als redundanter und ausfallsicher Speicher (Software-defined Storage) erfreut sich großer Beliebtheit und ist eine günstige und skalierbare Alternative zu den SAN- und RAID-Hardwareboxen der bekannten Hersteller. Im Open-Source-Bereich führt hier kein Weg mehr an Ceph vorbei. Viele kommerzielle Linux-Distributoren bieten es – ergänzt um Support und Administrationswerkzeuge – als eigenes Produkt an. So lässt sich Ceph selbst für unternehmenskritische Geschäftsanwendungen einsetzen.

Neben den Kosten und der Skalierbarkeit sind Hochverfügbarkeit und Redundanz von Ceph überzeugende Argumente, diese Technik als Speicher-Backend für möglichst viele Plattformen zu nutzen. Offensichtlich ist die Kombination Linux mit einem KVM-Hypervisor (oft in Form einer Private Cloud mit OpenStack), der über native Treiber für Ceph verfügt. Doch auch die proprietären Hypervisoren benötigen redundanten und hochverfügbaren Storage. Das hier beschriebene Konzept konzentriert sich auf die Kombination von Ceph mit VMware, ist jedoch genauso auf praktisch jede andere Hypervisor-Plattform übertragbar.

In einer Reihe von Artikeln stellte *iX* Design, Installation, Verwaltung und Tuning eines Ceph-Clusters vor [1–4]. Der weitere Artikel setzt einen Ceph-Cluster voraus und konzentriert sich auf das Installieren und Konfigurieren des iSCSI-Gateways. Mit dem Werkzeug *ceph-deploy* lässt sich, wie in Listing 1 beschrieben, in wenigen Schritten ein Ceph-Cluster zu Testzwecken installieren.

Anforderungsprofil Redundanz

Für wichtige Anwendungen gibt es im Prinzip nur zwei Möglichkeiten: Entweder sie implementieren die (Daten-)Redundanz selbst als Teil der Softwareebene (etwa eine Datenbankreplikation), oder sie verlassen sich auf eine zugrunde liegende Plattform oder Hardware, die die nötige Ausfallsicherheit schafft. Häufig nutzt man dazu die Virtualisierungsplattform, die heutzutage sowieso kaum noch eine Anwendung umgehen sollte. Eine virtuelle Maschine bei einem Hardwaredefekt innerhalb weniger Minuten wieder zu starten, erfüllt zwar nicht die allerstrengsten Servicegarantien, ist aber simpel, günstig, betriebssystemübergreifend und lässt sich vor allem vollständig automatisieren.

Dafür unverzichtbar ist eine redundante Datenspeicherung, unabhängig von den lokalen Festplatten der Hypervisor-Hardware. Klassisch verzichtet man auf Festplatten direkt im Hypervisor und verwendet externe Storage-Hardware, die eigenständig und in sich geschlossen die Daten auf mehrere Festplatten verteilt und durch redundante Komponenten eine hohe Verfügbarkeit garantiert. Steigt der Platzbedarf, lässt sich der Storage in Kapazität oder Leistung skalieren. Es bietet sich also an, einen großen Verbund wie Ceph als unternehmensweite Ressource bereitzustellen und auch für weitere Systeme oder Plattformen zu nutzen.

Ceph kommuniziert mit Client-Anwendungen ausschließlich über das ob-

iX-TRACT

- Ceph-Storage lässt sich per iSCSI-Protokoll freigeben, um Plattformen ohne native Ceph-Treiber anzubinden.
- Ein hochverfügbares Setup besteht aus zwei iSCSI-Gateway-Systemen, die Konfiguration erfolgt per *gwcli*.
- Als iSCSI-Clients konfigurierte VMware-Hosts verbinden sich mit den Gateways und können so Ceph als Storage-Backend für VMs nutzen.

jektbasierte RADOS-Protokoll, bei dem einzelne Dateien oder Datenstrukturen als einzelne Objekte redundant auf mehrere Storage-Server (OSD) abgelegt und von diesen wieder heruntergeladen werden können. Darauf aufbauend lassen sich RADOS Block Devices (RBD) definieren, die eine logisch zusammenhängende „Festplatte“ in kurze Datenblöcke unterteilen und als einzelne Objekte im Ceph-Cluster speichern.

Anwender können das RDB – durch einen Linux-Kerneltreiber als Blockgerät dargestellt – ganz normal mit einem Dateisystem formatieren und mounten. So können beliebige lokale Applikationen auch ohne spezielle Unterstützung für das Ceph-Protokoll den Storage verwenden. Schwierig ist aber, damit im Fehlerfall sauber und ohne Datenverlust denselben Storage automatisch einem Ersatzsystem zu präsentieren oder die gleichen Daten auf mehreren Systemen parallel zu nutzen.

Eleganter ist ein nativer und anwendungsspezifischer Treiber, wie er mittlerweile für den Open-Source-Hypervisor QEMU/KVM existiert. Ohne weitere Umwege über Blockgeräte, Dateisysteme oder Kerneltreiber kann dieser mit minimalem Overhead die virtuellen Festplatten eines Gastsystems über das Netzwerk direkt aus dem jeweiligen Ceph-Cluster bereitstellen.

Übersetzer gesucht

Für eine geschlossene Plattform wie VMware gibt es jedoch keinen eigenen Ceph-Client. Das erfordert einen „Übersetzer“ für das Ceph-RADOS-Protokoll in ein von der Plattform unterstütztes Netzwerkprotokoll. Für Ethernet-basierte Speichernetze ist iSCSI das Protokoll der Wahl; es lässt sich von praktisch jedem Betriebssystem und auch von VMware nutzen. Ein Linux-System dient als iSCSI-Gateway (iSCSI-Target), das mit dem Ceph-Cluster kommuniziert und ein oder mehrere RADOS Block Devices als iSCSI-Festplatte (LUN) präsentiert. Diese können die Clients (iSCSI-Initiatoren) wie ein Blockgerät einbinden (siehe Abbildung 1).

Damit es nicht zum Flaschenhals wird, benötigt das iSCSI-Gateway ausreichend Bandbreite zu den Ceph-Storage-Knoten und den iSCSI-Clients – idealerweise über separate Netzwerkkarten. Die Anforderungen an CPU und RAM sind moderat: Für eine große Zahl individueller, einzeln per iSCSI bereitgestellter LUNs sollte man pro LUN etwa 90 MByte Arbeitsspeicher einplanen.

Für ein Test-Setup kann man die Gateway-Software auf einem der Ceph-Storage-Knoten mitbetreiben. Zum besseren Skalieren und Segmentieren empfiehlt sich für den Produktivbetrieb aber ein eigenständiges System. Falls kein getrenntes iSCSI-Netzwerk nötig ist, lässt sich das Netz für die Ceph-Client-Zugriffe auch für den iSCSI-Verkehr nutzen.

Auf dem Ceph-Cluster sollte man die Timeouts der Storage-Knoten reduzieren. Die Werte bestimmen maßgeblich, wie schnell der Cluster bei einem Ausfall das IO wieder aufnehmen kann. Dazu dienen folgende Einträge in der Datei *ceph.conf*:

```
[osd]
osd heartbeat grace = 20
osd heartbeat interval = 5
```

Kann man auf die Werte keinen Einfluss nehmen, sollte man die gesetzten Werte dieser Timeouts trotzdem kennen, um auf den iSCSI-Clients etwas höhere einzustellen.

Maßnahmen für iSCSI ergreifen

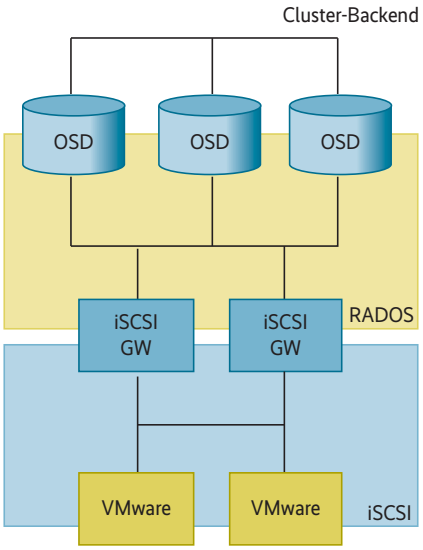
In der simpelsten Umsetzung nutzt ein iSCSI-Gateway die fast universelle Kombinationsmöglichkeit von Linux-Blockgeräten: Der Kernel kann schon seit Langem lokale Blockgeräte per iSCSI exportieren, und somit ist es nicht weit hergeholt, auch ein Ceph-RADOS-Blockgerät weiterzureichen. Dieser Ansatz funktioniert, ist durch die verschiedenen beteiligten Schichten und Kontextwechsel jedoch nicht ideal und somit besonders für hochverfügbare Setups nicht geeignet.

Das früher dominierende iSCSI-Framework *tgt* bekam deswegen ein eigenes RBD-Backend. Das umgeht den Kernel und kann so unabhängig von diesem aktuelle Ceph-Bibliotheken nutzen. Doch die Entwicklung hat die rein im Userland laufende iSCSI-Implementierung *tgt* aus diversen Gründen heute überholt. Viele Distributionen enthalten die Pakete aber noch.

Stattdessen konzentriert sich mittlerweile die Entwicklungsarbeit auf das Kernel-Framework „LIO“ (Linux IO), das seit der Aufnahme in den Kernel manchmal mit der Abkürzung „TCM“ (Target Core Module) auftaucht. Auch dafür existiert inzwischen ein spezielles Ceph-Backend, und erneut wollten die Entwickler statt einer streng an den Kernel gebundenen, komplexen Implementierung die normalen Ceph-Client-Bibliotheken nutzen und eine Userspace-Implementierung schaffen. Das Ceph-Backend steckt deswegen als Plug-in im sogenannten *tcmu-runner*.

Listing 1: Testcluster aufsetzen

```
ceph-deploy install ceph-1 ceph-2 ceph-3
ceph-deploy new ceph-1 ceph-2 ceph-3
ceph-deploy mon create-initial
ceph-deploy mgr create ceph-1 ceph-2 ceph-3
ceph-deploy osd create --data /dev/sdb ⤷
                                    ceph-1
ceph-deploy osd create --data /dev/sdb ⤷
                                    ceph-2
ceph-deploy osd create --data /dev/sdb ⤷
                                    ceph-3
```



Zwei hochverfügbare Gateways dienen als Protokollvermittler zwischen Ceph-RADOS und iSCSI (Abb. 1).

Listing 2: iSCSI-Target einrichten

```
/backstores/user:rbd> cd /iscsi
/iscsi> create
Created target iqn.2003-01.org.linux-iscsi.ceph-4.x8664:sn.dac86917407b.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.

# Client-Zugriff erlauben

/iscsi> cd iqn.2003-01.org.linux-iscsi.ceph-4.x8664:sn.dac86917407b/tpg1/acls
/iscsi/iqn.20...07b/tpg1/acls> create iqn.1996-04.de.suse:01:f2d7678fb16c
Created Node ACL for iqn.1996-04.de.suse:01:f2d7678fb16c

# LUN aktivieren

/iscsi/iqn.20...07b/tpg1/acls> cd ../luns
/iscsi/iqn.20...07b/tpg1/luns> create /backstores/user:rbd/disk5
Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.1994-05.com.example:testclient
```

Nur diese Implementierung wird aktiv weiterentwickelt und gepflegt und verfügt über die für HA-Setups nötigen Funktionen. Zunächst folgt hier jedoch das manuelle Einrichten auf einem einzelnen Gateway. Die Pakete finden sich zum Beispiel in einem openSUSE Leap 42.3 und der Aufruf

```
zypper install tcmu-runner ⤸
              tcmu-runner-handler-rbd targetcli
```

installiert sie dort.

Zur manuellen Konfiguration ist zuerst der Dienst *tcmu-runner* zu starten:

```
systemctl daemon-reload
systemctl enable tcmu-runner
systemctl start tcmu-runner
```

Ist noch kein RBD-Pool im Ceph-Cluster vorhanden, muss man diesen erstellen:

```
ceph osd pool create rbd 1024 1024
rbd pool init rbd
```

Anschließend lässt sich per

```
rbd create rbd-disk1 --size 100G
```

ein RBD-Gerät anlegen. Für die weiteren Schritte kommt die interaktive Konfigurationskonsole *targetcli* zum Einsatz. Um sicherzustellen, dass der *rbd*-Treiber verfügbar und geladen ist, lässt man sich die verfügbaren Backends auflisten

```
/> ls /backstores/
o- backstores .
[...]
o- user:rbd
[...]
```

Nach dem Wechsel in den entsprechenden Zweig kann man auch hier dasselbe RBD-Gerät anlegen

```
cd /backstores/user:rbd/
/backstores/user:rbd> create tcmu-disk1 5G ⤸
                                  /rbd/rbd-disk1
Created user-backed storage object disk5 size ⤸
                                  5368709120.
```

Listing 3: *[/etc/ceph/iscsi-gateway.cfg](#)*

```
[config]
cluster_name = ceph
gateway_keyring = ceph.client.admin.keyring
api_secure = false
api_user = admin
api_password = admin
api_port = 5000
trusted_ip_list = \ ⤸
                  192.168.0.151,192.168.0.152
```

Das Erstellen und Einrichten des iSCSI-Target erfolgt anschließend in einem anderen Zweig (siehe Listing 2).

Nach dem Erzeugen des Target im Verzeichnis *iscsi* muss man den Client-Zugriff erlauben. Der erfolgt über den „IQN“ (iSCSI Qualified Name), der für jeden Initiator einmalig und oft nicht veränderbar ist. Anschließend wird das RBD-Gerät als erste LUN für das neue Target aktiviert. Ab jetzt lässt es sich durch den iSCSI-Initiator nutzen.

Hochverfügbarkeit ist essenziell

Unverzichtbar für einen hochverfügbaren und wartbaren Dienst ist die Redundanz des iSCSI-Gateways. Durch mehrere im Verbund agierende Systeme und mit korrekter Konfiguration des sogenannten Multipathing auf dem iSCSI-Client kann man die Hochverfügbarkeit gewährleisten.

Multipathing bedeutet, dass ein Client dieselbe iSCSI-Festplatte über mehrere, idealerweise unabhängige Netzwerkpfade auf mehreren Gateways erreichen kann. Fällt einer der Pfade oder ein Gateway aus, setzt der Client nach kurzem Timeout den IO einfach über einen verbleibenden Pfad fort. Allerdings funktioniert nur eine Aktiv-passiv-Nutzung, die Zugriffe auf eine bestimmte LUN findet also immer nur über das für diese LUN primäre Gateway statt, erst im Fehlerfall erfolgt der Wechsel auf ein anderes Gateway. Dadurch bekommt man automatisch eine rudimentäre Lastverteilung, da alle Gateways für etwa gleich viele aktive LUNs das primäre Gateway sind. Die Wahl des primären Gateways fällt beim Anlegen des LUN automatisch auf das Gateway mit den wenigsten aktiven LUNs – ein dynamisches oder manuelles Umverteilen ist momentan nicht vorgesehen.

Beim Verbinden mehrerer Gateways besteht die eigentliche Herausforderung im Abgleichen von Konfiguration und Zustandsinformationen zwischen allen Gateway-Systemen. Jedes einzelne muss jederzeit alle LUNs bedienen können, falls ein anderes ausfällt. Nach dem Hinzufügen eines neuen Target oder LUN oder einer Passwortänderung et cetera müssen diese Konfigurationsdaten sofort und automatisch allen Gateways zur Verfügung stehen und angewendet werden. Ein Administrator kann dies manuell nicht schnell und zuverlässig genug leisten, deswegen nutzt man eine Architektur, die alle Systeme miteinander kommunizieren lässt und die Konfiguration abstrahiert und automatisiert.

Dazu startet der Dienst *rbd-target-api* auf jedem Gateway einen HTTP-Server auf Port 5000, der von den anderen erreichbar sein muss. Mit einfachen REST-Befehlen kann man darüber die Konfigurationselemente erzeugen, verändern und löschen. Auch das primär zur händischen Konfiguration herangezogene Kommandozeilenwerkzeug *gwcli* nutzt ausschließlich die API. Diese Architektur ermöglicht eine Anbindung an grafische Werkzeuge oder andere Automatismen. Teil der API-Logik ist der Konfigurationsabgleich mit den anderen Gateways: Der API-Daemon des einen Gateways ruft also für jeden Befehl auch die API der anderen auf, um diese über spezielle Methoden identisch zu konfigurieren.

Auf jedem Gateway gibt es zudem einen weiteren Daemon *rbd-target-gw*, der aber im Gegensatz zur API nicht über das Netzwerk kommuniziert. Er kümmert sich nur um die lokale Konfiguration des zugrunde liegenden Kernel-Frameworks. Clever, aber irgendwie naheliegend: Der Daemon liest und schreibt den internen aktuellen Konfigurationszustand nicht etwa in eine lokale Datei, sondern nutzt dafür ein Ceph-Objekt. Dieses Konfigurationsobjekt direkt aus dem Ceph-Cluster ist für alle Gateways dasselbe, was ein Auseinanderdriften der Konfiguration verhindert.

Auf dem Weg zum iSCSI-Gateway

Ein Blick in die offizielle Dokumentation des Ceph-Projekts eröffnet einen nicht gerade einfachen Weg zum Setup des iSCSI-Gateways (siehe ix.de/ix1806134). Basierend auf einem Red Hat Enterprise Linux 7.5 – das erst im April erschienen ist und noch nicht als CentOS-Variante vorliegt – sind dafür die nötigen Komponenten direkt aus einigen Git-Repositories zu übersetzen und einzuspielen. Dieser Weg ist jedoch nicht unbegründet, da Red Hat die Ceph-Entwicklung federführend vorantreibt und dort alle wichtigen Änderungen als Backports zum Beispiel in den sehr alten RHEL-Kernel 3.10 einfließen lässt.

Listing 4: Host Groups nutzen

```
cd /iscsi-target/iqn.2003-01.com.redhat.iscsi-gw:iscsi-igw/host-groups
$ create cluster
ok
$ disk add rbd.vmware_datastore
ok
$ cd ../../hosts
$ create iqn.1998-01.com.vmware:test1
ok
/iscsi-target...vmware:test1> auth chap=username/secretpassword
ok
/iscsi-target@$p>...vmware:test1> cd ../../host-groups/cluster
/iscsi-target...roups/cluster> host add iqn.1998-01.com.vmware:test1
```

Im Upstream-Kernel fügte Red Hat diese Änderungen jedoch erst kürzlich hinzu, weswegen man direkt zum noch nicht finalen Kernel 4.17 rät – oder zum Kernelzweig der Ceph-Entwickler.

Als fertige RPM-Pakete gibt es die gesamten Komponenten von Red Hat nur als Teil des lizenzpflichtigen Red Hat Enterprise Ceph-Storage (RHCS). Da die Software untereinander eng verzahnt ist und stetig weiterentwickelt wird, ist die Pflege eines stabilen Repository, in das gleichzeitig aber auch alle wichtigen Bugfixes einfließen, nicht einfach.

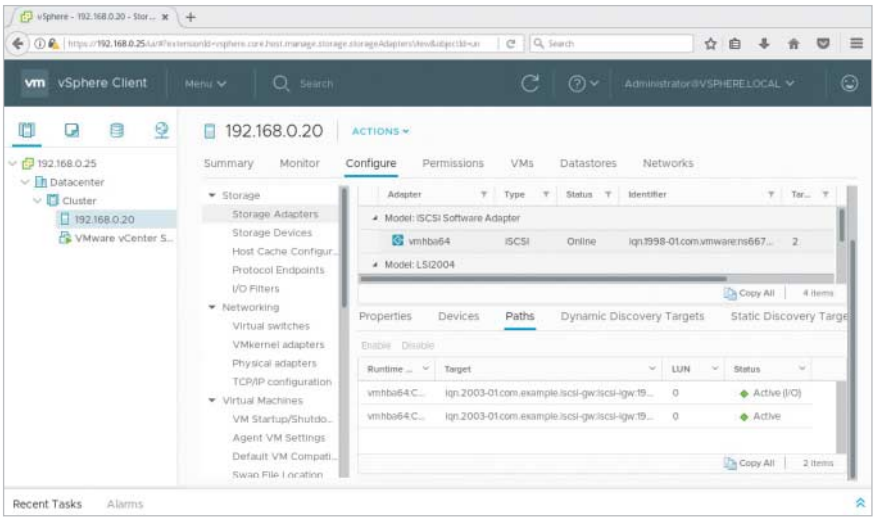
RPM-Pakete des aktuellen Entwicklungszustands für CentOS 7 erhält man aus dem Build-System des Ceph-Projekts (siehe ix.de/ix1806134). Folgt man den Repo-Links von *kernel*, *tcmu-runner*, *python-rtslib*, *ceph-iscsi-config* und *ceph-iscsi-cli* und installiert die zugehörigen Pakete, lässt sich das hier beschriebene Vorgehen auch ohne RHCS-Lizenz auf CentOS 7 nachvollziehen. Außer für einen Test eignen sich diese Pakete aber natürlich nicht für den dauerhaften Einsatz. Teil der stabilen Ceph-Release oder einer Community-Distribution sind die Pakete leider noch nicht.

SUSE ging schon früh einen technisch anderen Weg, der jedoch in der Community bisher keine große Akzeptanz fand. Auch hier trägt das kommerzielle Produkt die Entwicklung, SUSE Enterprise Storage (SES). Es bietet schon seit mehreren Jahren ein hochverfügbares iSCSI-Gateway. Als technischer Vorreiter war die Eigenentwicklung sinnvoll, doch die Kernelkomponente fand Upstream keine Befürworter. Daher ist heute absehbar, dass sich auch der Management-Daemon *lrbd* als Konzept nicht durchsetzen dürfte. Das aktuelle SES 5 liefert – als Technology Preview ohne Support – schon den gesamten Stack rund um *tcmu-runner* mit, und man kann spekulieren, dass SUSE früher oder später ebenfalls darauf wechseln wird.

In RHCS lassen sich alle Ceph-Cluster-Komponenten bevorzugt mit Ansible konfigurieren. Es existiert auch ein Ansible-Modul für das iSCSI-Gateway. Doch auch das manuelle Installieren ist nicht schwer und lässt sich besser auf andere Umgebungen übertragen.

Ein registriertes Basissystem mit RHEL 7.4 oder 7.5 benötigt die Zusatz-Repositories rhel-7-server-rhceph-3-tools-rpms und rhel-7-server-extras-rpms. Nach dem Installieren der Pakete ceph-common, ceph-iscsi-cli, ceph-iscsi-config und tcmu-runner wird die Konfigurationsdatei */etc/ceph/iscsi-gateway.cfg* mit dem Inhalt von Listing 3 erstellt.

Hinzufügen des iSCSI-Software-Adapters zu einem Host in vSphere (Abb. 2)



Das System hat die redundanten Pfade zu beiden Gateways erkannt (Abb. 3).

Wichtig ist, hier alle IPs aufzuführen, von denen Konfigurationsoperationen (Erstellen von Targets et cetera) erlaubt sein sollen, etwa die anderen Gateway-Knoten. Die API-Software führt einen bitgenauen Vergleich dieser Datei zwischen allen Gateways durch, sie muss deswegen überall absolut identisch sein. Erzeugt man die Zertifikate als */etc/ceph/iscsi-gateway.crt* und */etc/ceph/iscsi-gateway.key* selbst, lässt sich über den Parameter *api_secure = true* die SSL-Verschlüsselung aktivieren. Man muss nur sicherstellen, dass der OSD-Pool existiert, oder ihn bei Bedarf (auf einem System mit Ceph-Admin-Rechten) anlegen:

```
ceph osd pool create rbd 1024 1024
rbd pool init rbd
```

Der API-Dienst hängt von allen anderen nötigen Komponenten ab und startet diese automatisch mit

```
systemctl daemon-reload
systemctl enable rbd-target-api
systemctl start rbd-target-api
```

Nun lässt sich das Kommandozeilentool *gwcli* zum Konfigurieren nutzen. Es funktioniert sehr ähnlich zu *targetcli*, das darf man aber nicht für Änderungen verwenden, um die Konsistenz der Konfiguration auf den Gateways zu gewährleisten.

Zum Selbstschutz erlaubt *gwcli* erst nach dem Erstellen von mindestens zwei Gateways die weitere Konfiguration, weshalb man mit diesen beginnt. Die Gateways bedienen das gleiche Target, sodass dies das übergeordnete Element bildet:

```
cd /iscsi-target
create iqn.2003-01.com.redhat.iscsi-gw: ⤸
                                    iscsi-igw
```

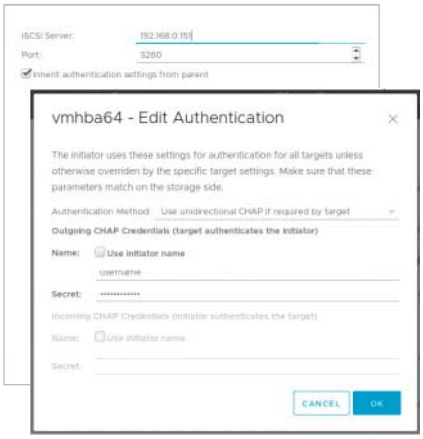
Die Gateways fügt man mit ihrem Hostnamen und einer von den Clients aus erreichbaren IP-Adresse hinzu. *skipchecks=true* umgeht eine Versionsprüfung, die momentan nur auf RHEL 7.4 (die Kompatibilität mit RHEL 7.5 ist technisch gewährleistet, aber der Versionscheck noch nicht korrigiert) und mit einer sehr exakten Kombination der restlichen Software funktioniert:

```
cd iqn.2003-01.com.redhat.iscsi-gw: ⤸
                                    iscsi-igw/gateways
create ceph-4 192.168.0.151 skipchecks=true
create ceph-5 192.168.0.152 skipchecks=true
```

Nun kann man eine virtualisierte Festplatte erzeugen. Im Gegensatz zu *targetcli* muss dieses Gerät aber nicht schon im *rbd*-Pool existieren. Das Gateway erstellt es dort. Gibt es bereits ein RBD-Gerät mit dem angegebenen Namen, importiert es das nur. So lassen sich auch bestehende Daten übernehmen:

```
cd /disks
create pool=rbd image=vmware_datastore ⤸
                                    size=1024G
```

Im nächsten Schritt erhält der Client mit der spezifizierten IQN Zugriff auf das Target. Hierbei ist eine Passwortauthenti-



Ein Häkchen bei „Inherit authentication settings …“ führt zum Konfigurationsdialog für die Authentifizierung (Abb. 4).

fizierung per CHAP zwingend nötig, die aber jeder Initiator beherrscht:

```
cd /iscsi-target/iqn.2003-01.com.redhat. ⤸
                        iscsi-gw:iscsi-igw/hosts
$ create ⤸
      iqn.1998-01.com.vmware:ns6679117-30cf2649
ok
$ auth chap=username/secretpassword
ok
```

Als erste LUN für diesen Client gibt

```
/iscsi-target...9117-30cf2649> disk add ⤸
                                rbd.vmware_datastore
```

die Festplatte frei, die jetzt vom Initiator aus nutzbar ist.

Clients kann man in Host Groups zusammenfassen, um ihnen identische LUNs zuzuweisen, zum Beispiel bei einem VMware-Cluster. Allerdings muss man die Hosts trotzdem einzeln erstellen, bevor man sie der Gruppe hinzufügen kann. Hat ein Host bereits eine direkt zugewiesene LUN, muss man diese erst wieder entfernen, bevor er sich einer Gruppe hinzufügen lässt (siehe Listing 4).

Grafische Konfiguration via vSphere

Damit sich iSCSI auf einem von vSphere verwalteten Host nutzen lässt, muss man zuerst in dessen Storage-Adapter-Konfiguration via „Add Software Adapter“ den iSCSI-Adapter erzeugen (siehe Abbildung 2). Der erscheint danach als *vmhba64*. Über den IQN aus den Eigenschaften kann man ihn als Client bei den Gateways hinzufügen und ihm LUNs zuweisen (Abbildung 3). Über *Dynamic Discovery Targets* gibt man die IP eines der Gateways an und konfiguriert die CHAP-Authentifizierung (Abbildung 4). Nach einem Rescan des Adapters erkennt vSpehre die anderen Gateways und alle bereitgestellten LUNs und bindet sie redundant ein. Über die Reiter *Devices* und *Paths* lässt sich das verifizieren.

In den Advanced Options des Adapters sollte man den Parameter *RecoveryTimeout* auf *25* erhöhen (sie Abbildung 5). Der steuert, nach wie vielen Sekunden ohne Antwort ein Pfad oder Gateway als „kaputt“ erkannt und der IO auf dem anderen Pfad fortgesetzt wird. Den Ausfall eines Storage-Knotens muss der Ceph-Cluster intern abfangen und lösen, wobei natürlich seitens des Clients unvermeidbar ist, dass so lange eventuell das IO pausieren muss. Der Wert des RecoveryTimeout muss dazu größer sein als die Ceph-internen Timeouts der Storage-Knoten, damit man bei deren Ausfall nicht auch unnötigerweise einen Schwenk des Gateways vornehmen oder gar beide Gateways als „ausgefallen“ markieren muss.

Die erkannten LUNs lassen sich wie jede andere Festplatte als Datastore einbinden (siehe Abbildung 6). Die Konfiguration eines einzelnen ESXi ist vom Vorgehen identisch zu vSphere.

Fazit

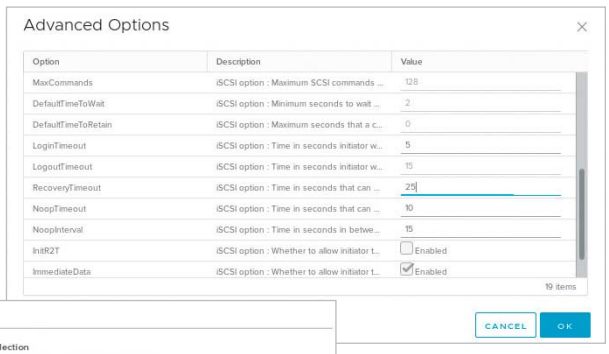
Zwar basiert das iSCSI-Gateway auf einem einfachen Konzept, doch historisch bedingt scheint die Implementierung rund um *tcmu-runner* noch nicht ganz reif. Lediglich für das Setup eines nicht hochverfügbaren Einzel-Gateways finden sich frei erhältliche Pakete. Trotz komplett quelloffener Entwicklung kann man keinem der Distributoren, die maßgeblich die Technik hinter dem Konzept entworfen haben, vorwerfen, die Technik nur für sich behalten zu wollen.

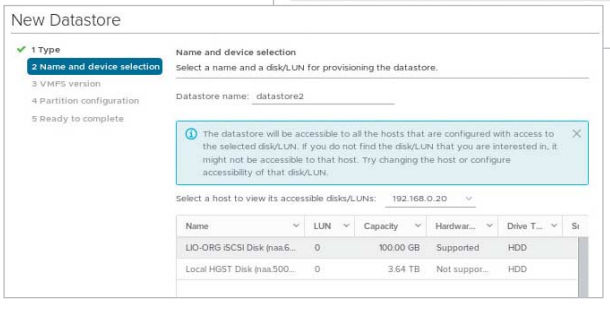
Allerdings haben sie es bislang noch nicht in die jeweiligen Community-Ableger der Enterprise-Distributionen oder die gut gepflegten stabilen Paket-Repositories des Ceph-Projekts geschafft, eine Situation, die sich hoffentlich noch verbessern wird. Die Software selbst aus dem Git zu holen, wie die offizielle Dokumentation es vorschlägt, ist abseits von Entwicklungsmaschinen nicht tragbar. So bleibt derzeit nur nur die Empfehlung, die Evaluationslizenzen der Enterprise-Produkte zu testen und für produktive Setups zu kaufen. (avr@ix.de)

Lothar Gesslein

arbeitet als Linux-Consultant bei der B1 Systems GmbH.

Für eine Abstimmung mit den Ceph-Parametern muss man den *RecoveryTimeout* des iSCSI-Adapters erhöhen (Abb. 5).





vSphere bindet die erkannte iSCSI-Festplatte als Datastore ein (Abb. 6).

Literatur

[1] Martin Gerhard Loschwitz; Cluster-Dateisysteme; Krakenstube; Tutorial, Teil 1: Speicher skalieren mit Ceph; *iX* 7/2014, S. 104

[2] Martin Gerhard Loschwitz; Cluster-Dateisysteme; Krakenpflege; Tutorial, Teil 2: Ceph einrichten und konfigurieren; *iX* 8/2014, S. 124

[3] Martin Gerhard Loschwitz; Cluster-Dateisysteme; Den REST gegeben; Tutorial, Teil 3: Client-Zugriff auf Ceph über das QEMU-RBD und das Ceph Object Gateway; *iX* 9/2014, S. 146

[4] Martin Gerhard Loschwitz; Storage II; In Bestform; Performance-Tuning für Ceph; *iX* 2/2018, S. 38

Alle Links: ix.de/ix1806134