- dd-Transfer –
Effizientes Übertragen Großer Fesplattenimages

1 Einleitung

Von Zeit zu Zeit stellt sich die Aufgabe, den Inhalt einer kompletten Festplatte zu übertragen. Dies kann beispielsweise passieren, um eine Sicherung vorzunehmen, eine neue Festplatte in Betrieb zu nehmen oder aber, um den Inhalt eines Speichermediums Byte für Byte in die Cloud zu überspielen.

Handelsübliche Festplatten verfügen inzwischen über ein Fassungsvermögen von bis zu zehn Terrabyte und mehr. Dieses Datenvolumen ist nicht leicht über die vorhandene Bandbreite der üblichen Internetanschlüsse zu transportieren. Es kann sogar für lokale Netzwerke eine sehr langwierige Aufgabe sein. Aus dieser Problematik reifte die Frage, warum es nicht längst ein Werkzeug gibt, dass an dieser Stelle optimal weiter hilft.

Dieser Artikel beschreibt zunächst herkömmliche Mittel und Wege, dem Problem beizukommen. Aus der Angabe der Beschränkungen dieser Methoden leiten sich die Anforderungen und die Teilschritte zur Lösung des Problems ab, das mit dem Programm ddtransfer.sh geboten werden soll.

2 Herkömmliche Möglichkeiten

Das Kopieren, Übertragen und Einspielen der Daten eines USB-Stickst mit einer Größe von 16 GB dauert beispielsweise zu normalen Arbeitszeiten in Berlin 68 Minuten über eine Anbindung von 100 MB, die sich zur selben Zeit mehr als 120 Angestellte teilen (Stand April 2019).

Die Einzelschritte waren wie folgt:

1) Erstellen des Images

root@ddtransfer:~# time dd if=/dev/sdc bs=1G iflag=fullblock of=sdc.img

Dauer: 12 Minuten, 25 Sekunden

2) Übertragung

root@ddtransfer:~# time rsync --compress --compress-level=9 sdc.img root@158.222.102.233:/mnt/1/.

Dauer: 51 Minuten, 10 Sekunden

3) Einspielen der Daten in ein Storagevolume

root@las-transfer2:/mnt/1# time dd if=sdc.img of=/dev/vdd bs=1G iflag=fullblock

Dauer: 3 Minuten, 28 Sekunden

Einen Test mit CloneZilla[1] als Gegenprobe habe ich abgebrochen, weil ich dafür Passwortauthentifizierung hätte aktivieren müssen. Mit Bar Bar Copy (bbcp[2]), hätte sich die Übertragung erheblich beschleunigen lassen, die Daten wären jedoch unverschlüsselt übertragen worden. Zum andere ist

1 https://clonezilla.org/

das Ziel, sehr große Dateien zu übertragen. Werden aber zuvor Images angelegt, muss ein ebenso großer Speicher vorab bereit gestellt werden (auch trotz Komprimierung). Das Transferscript benötigt demgegenüber keinen extra freien Speicher auf der Festplatte (abgesehen von den üblichen Erforderniss eines Betriebssystems).

Ein anderer Weg, der Kunden anderer Firmen für die Übertragung großer Datenmengen angeboten wird, ist der Versand von Festplatten, oder das Bespielen eines Servers (ab ca. 10 TB), der sodann an den Provider zurückgeschickt wird. Weit jenseits der 10 TB mag diese Variante praktikabler sein, aber mit der weiteren technischen Entwicklung ist durchaus absehbar, dass ddtransfer.sh auch bei extrem großen Datenmengen ein gangbarer Weg ist. Eine Bandbreite von 1Gb/sec ist nicht mehr außergewöhnlich, der Transfer von 10TB dadurch eine machbare Aufgabe.

3 Anforderungen an ein neues Programm

Bei einer Imagegröße von 15 GB ist das zuvor beschriebene Vorgehen völlig normal und angemessen. Die eigentlichen Probleme treten erst in Erscheinung, wenn es sich

a) um erheblich größere Datenmengen handelt, die

b) über eine geringe Bandbreite transferiert werden müssen und

c) sich deshalb auch die Frage nach einer effizienten Prüfung der Datenintegrität stellt.

Die Dauer der Übertragen kann schnell mehrere Tage in Anspruch nehmen. Sollte es in dieser Zeit zu einem Ausfall kommen, muss der Vorgang der Übertragung neu gestartet werden. Je weniger Daten nun neu übertragen werden müssen, desto geringer fällt der Zeitverlust ins Gewicht.

Wurde der Transfer eines Teilschrittes abgeschlossen, lässt sich durch das Erstellen von Checksummen im Anschluss an die Übertragung feststellen, ob es zu einem Bitflip oder anderweitigen Veränderungen im Datenbestand gekommen ist. Bei den bisherigen Verfahren ist diese Prüfung erst als eine Gesamtprüfung durchführbar, sollte nun eine Veränderung festgestellt werden, ist dies am Ende eines sehr langen Vorgangs umso ärgerlicher.

Die geringe Bandbreite ist der Teil, der für die meiste Zeit in der Verarbeitung sorgt. Die wenigsten Programme sind jedoch in der Lage, diese voll auszunutzen. Ein effektiver Weg, die Bandbreite noch weitgehender zu nutzen, ist das Aufrufen paralleler Verbindungen. Die jeweiligen Verbindungen werden keinesfalls gleichartig schnell abgearbeitet. Es ist durchaus nicht ungewöhnlich, dass wenn 16 Verbindungen nacheinander mit einem Abstand von ein paar Sekunden gestartet werden, die zehnte Übertragung als erste abgeschlossen wird.

Das Script ddtransfer.sh soll genau diesen Anforderungen gerecht werden.

1. Die zu übertragenden Daten werden in Blöcke aufgeteilt, die zu kopieren ’dd‘ ggf. mehrfach parallel aufgerufen wird.

2. Für jeden Block wird während seiner Erstellung eine Checksumme gebildet.

3. Je nach Bandbreite werden für das Senden der Blöcke acht parallele Verbindungen genutzt. Dies kann je nach Bedarf angepasst werden.

2 http://www.slac.stanford.edu/~abh/bbcp/ und http://pcbunn.cithep.caltech.edu/bbcp/using_bbcp.htm

4. Die Blöcke werden nach der Übertragung auf dem Zielsystem direkt in das Zieldevice geschrieben.

5. Nach dem Schreiben einer jeden Übertragung auf das Zieldevice wird von dem fertig bearbeiteten Block erneut eine Checksummen gebildet.

6. Am Ende werden die Checksummen verglichen. Sollte es zu Differenzen gekommen sein, wird der betroffene Block erneut übertragen und berechnet.

7. Im Falle einer Unterbrechung soll der Prozess wieder aufgenommen werden können.

Soweit mir bekannt, erfüllt derzeit kein verfügbares Werkzeug für die Übertragung großer Devices diese Bedingungen. Das Programm setzt auf ‚dd' auf, welches zu den sogenannten Core-Utils von Unix gehört, es dürfte also auf jedem Linux laufen, dass eine Shell bietet, auch der ‚ssh-Agent' sollten vorhanden sein. Somit lässt es sich mit einem gängigen Live-Linux einsetzen, beispielsweise um Windows-Installationen zu übertragen. Ebenso lässt es sich auf kleiner Hardware wie dem Raspberry Pi einsetzen, der beispielsweise als Übertragungsstation dient.

Der eigentliche Knackpunkt der gesamten Prozedur ist die Internetverbindung, es ist für schmale Bandbreiten optimiert. Die Anwendung des Scriptes hat in lokalen Netzen ebenfalls Sinn, dabei sollte die Komprimierung abgeschaltet werden, ebenso im Fall einer Anbindung an das Internet mit großer Bandbreite.

4 Problematiken

1 Plattenplatz

Weil parallel vom Quelldevice gelesen und ins Zieldevice geschrieben wird, gibt es keine besonderen Anforderungen an vorhandenen freien Speicherplatz.

2 Bandbreite

Wenn das Programm mit sehr vielen Verbindungen aufgerufen wird, verlangsamt sich der Durchsatz für alle anderen, die das selbe Netz benutzen. Das bedeutet aber auch, wie eben schon beschrieben, dass die Übertragung zur Nachtzeit oder am Wochenende den Vorgang deutlich beschleunigt.

3 Arbeitsweise von dd

'dd' kann immer nur entweder lesen oder schreiben. Aufgrund dessen ist es von Vorteil, einigermaßen große Blöcke in einem Arbeitsschritt zu verarbeiten. Gleichzeitig vermag dieses Programm gezielt bestimmte Positionen einer Datei bzw. eines Blockdevices block- oder auch byteweise zu adressieren. Damit ist es das einzige Werkzeug, welches für unseren Zweck in Frage kommt. Die genaue Positionsbestimmung ist wesentlich für die Aufteilung der zu übertragenden Daten in klar abgegrenzte Einzelschritte. Die Block- und Dateigröße ist deshalb immer ein Vielfaches der Blockgröße von 512 Byte. Bisher habe ich je Prozessorkern nur einen dd-Prozess zugelassen (gibt es nur einen Kern, sind es zwei). Es ist noch zu untersuchen, ob es sinnvoll sein kann, noch mehr zu starten.

Die Arbeitsweise macht es etwas umständlich festzustellen, ob dd noch in ein Device schreibt, oder der jeweilige Vorgang bereits abgeschlossen ist. Anhand der Größe einer Datei muss dies ggf. vorab kalkuliert und dann laufend überprüft werden. Bei der Entwicklung des Skriptes hat es sich deshalb

als sinnvoll erwiesen, angefangene Schreibvorgänge nach einer Unterbrechung erneut zu starten. Die Option ‚status=progress‘ wird zwar in neuen Implementierungen von ‚dd‘ angeboten, ist aber nicht immer verfügbar und im vorliegenden Fall nicht hilfreich.

5 Wiederaufnahme nach Unterbrechung

Sollte es zu einem Ausfall des Quell- rechners oder Zielrechners kommen, kann das Programm neu aufgerufen werden. Als Option muss der Name einer Report-Datei genannt werden. In dieser stehen die Optionen des vorhergehenden Aufrufs. Dann wird der Stand der Datenübertragung und der erstellten Blöcke geprüft. Wo das nötig ist, werden unterbrochene Teilprozesse neu gestartet und der Gesamtprozess fortgesetzt.

Das Hauptmittel ist die Report-Datei, die ausgelesen und dann fortgeschrieben wird. Zum anderen werden angefangene Übertragungen erneut veranlasst.

6 Abschluss

Am Ende wird die Reportdatei verarbeitet und geprüft, ob die Start- und die End-Checksummen übereinstimmen oder welche sogar fehlen. Das Fehlen von Prüfsummen der Endverarbeitung kann vorkommen, wenn die Übertragung unterbrochen worden ist. Wenn die Endsumme fehlt oder sich unterscheidet, wird sie erneut berechnet.

7 Kommandozeilenhilfe

```
$ ./ddtransfer.sh

Usage: ./ddtransfer.sh -l|--local_device 'source_device' -r|--remote_device 'target_de-
vice' -H|--TargetHost 'target_host'
(--no_compression (per default compression is active)
(-p|--ssh_port 'ssh_port' (default is '22'))
(-m|--max_connections (max transmissions in parallel, default is 8))
(-S|--restart 'formerly written report file' (any other given option will be overwritten))

A running ssh-agent is necessary.
Tested with ext4, ntfs, xfs, btrfs.
It is recommended not to start any other dd execution on the local or target host while
this programm is running.

For sudo users like Ubuntu, do:
1) 'sudo su -'
2) 'eval `ssh-agent -s`'
3) 'ssh-add /home/user/.ssh/id_rsa'
Now you can start the dd-transfer script.

Example:

./ddtransfer.sh -l /dev/vdd -r /dev/vdd -H 46.16.76.151
```

./ddtransfer.sh --local_device /dev/vdd --remote_device /dev/vdd --TargetHost 46.16.76.151

This transfers the contents from the local storage volume /dev/vdd to the remote volume /dev/vdd on host 46.16.76.151.

Partitions like /dev/vdd1 are also usable.

8 Link

Downloadquelle https://github.com/ionos-enterprise/ionos-network-helper