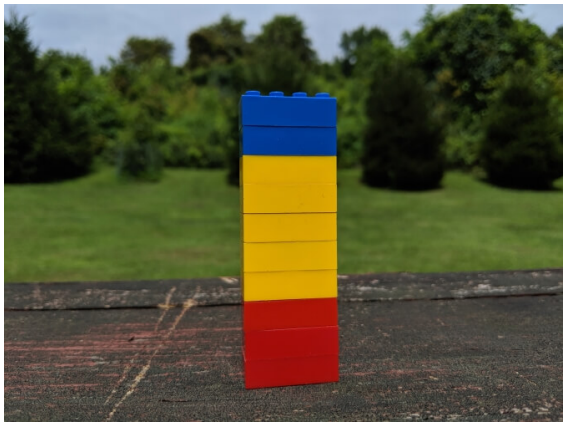
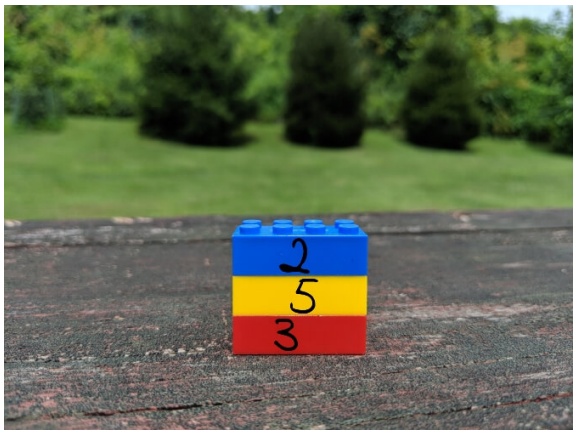
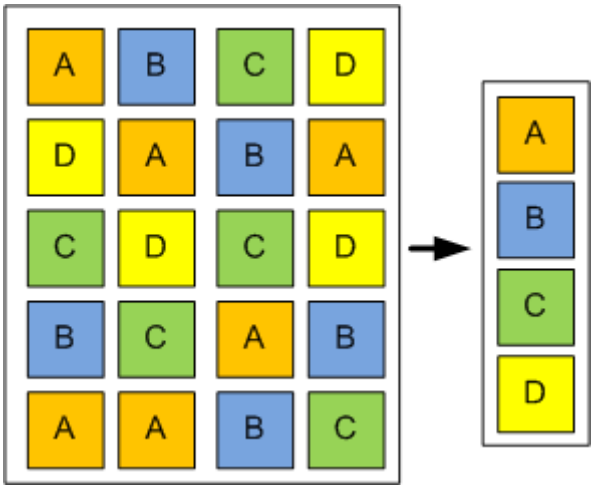
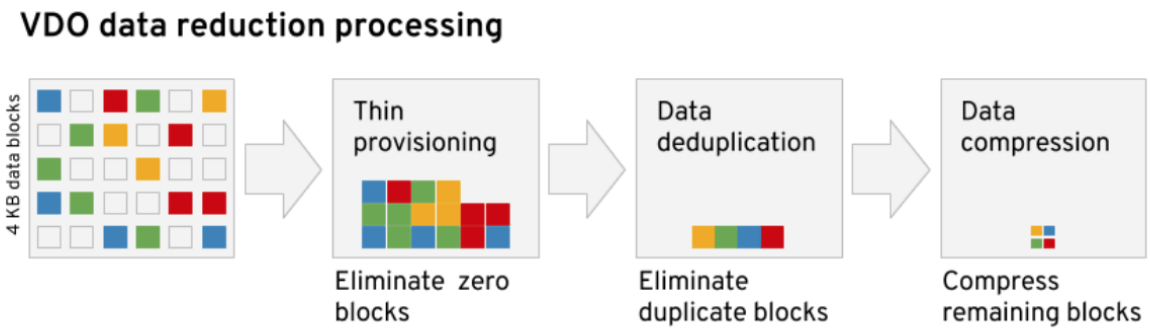




Недавно мы с вами разбирали утилиту gzip, которая позволяет сэкономить пространство с помощью сжатия данных и реализовали это для бэкапа домашней директории. gzip работает с файлами и на этих картинках показан простой пример. Но у такого сжатия есть пара нюансов - со сжатыми файлами работать не получится, их нужно предварительно разжимать. Точнее, есть всякие утилиты, которые позволяют работать с сжатыми файлами, но это не годится для всего. Да и сжимается не абсолютно всё, а только то, что вы укажете.



Однако есть механизм, который позволяет экономить пространство ещё на уровне блоков данных - дедупликация. В целом механизм напоминает сжатие, но работает не на уровне файлов и алгоритмов сжатия, а на уровне блоков - дублирующиеся блоки заменяются ссылками на один. Очень яркий пример использования - если у вас множество виртуальных машин. Представьте, что вы подняли 100 виртуалок с Centos-ом, каждая выполняет свою задачу. На всех одна и та же операционная система, ядро, библиотеки и куча утилит - всё это одно и тоже, дублирующееся на сотни виртуалок. Файлы работающих виртуалок не сожмёшь с помощью gzip-а, да и эффективность от такого сжатия будет сомнительная. А вот дедупликация просто увидит кучу одинаковых блоков и заменит их ссылками, что значительно сэкономит пространство. Минусы, конечно, тоже есть - дедупликация постоянно работает, из-за чего увеличивается использование оперативной памяти и диска, соответственно, где-то проседает скорость работы. Т.е. на домашних компьютерах это не так нужно, в рабочей среде зависит от конкретной задачи.



С недавних пор у RedHat появилась технология VDO - virtual data optimizer. Она позволяет экономить пространство диска, при этом жертвуя ресурсами процессора и оперативки. И кроме дедупликации она избавляется от пустых блоков, в которых одни нули, а также сжимает итоговые данные алгоритмом LZ4. Также она обеспечивает thin provisioning - т.е. когда вы компьютеру выделили, скажем, 50 гигабайт, а операционной системе говорите, что там 500.

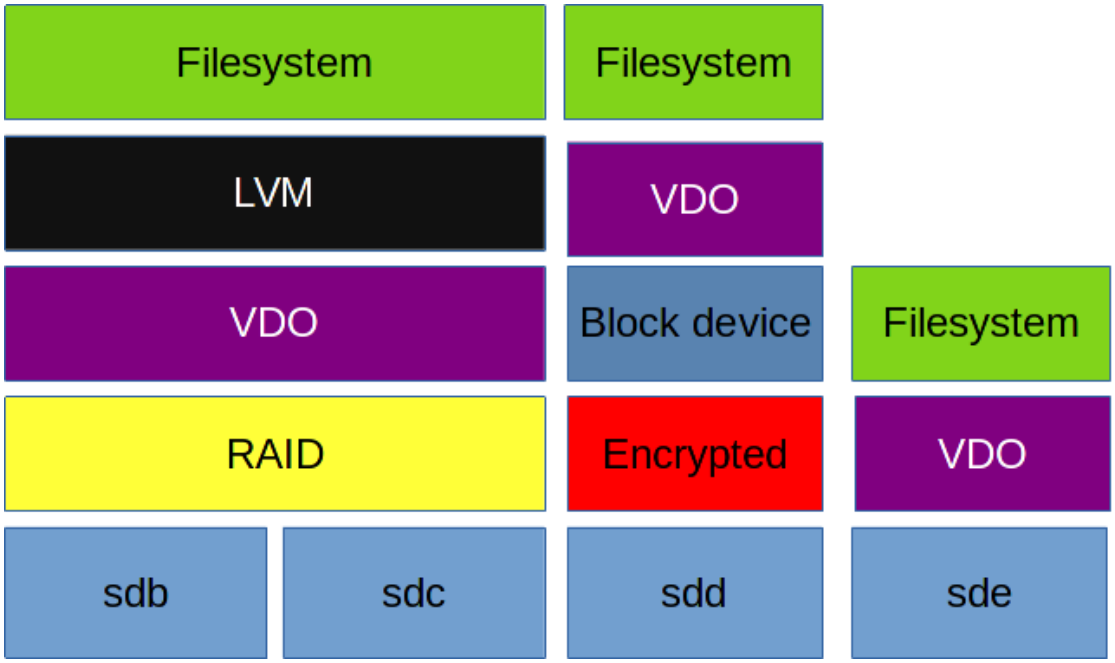
Зачем это нужно? Представьте, что вам нужно поднять несколько виртуальных машин и каждой дать по 50 гигабайт, на будущее, потому что количество данных будет расти. Но, во первых, сейчас каждая виртуалка будет занимать не больше 5 гигабайт, а во вторых у вас сейчас нет таких больших дисков. Будут расти данные - вы добавите новые диски. Если вы сейчас выделите в виртуалках всего по 5 гигабайт, то вам постоянно нужно будет в них увеличивать пространство по мере добавления дисков. А с thin provisioning-ом вам не нужно это делать - виртуалка будет занимать столько места, сколько ей нужно на самом деле, а вам просто нужно следить, чтобы выделенный вами диск не закончился. Если закончится - будет плохо, потому что виртуалки будут видеть свободное пространство, но не смогут туда писать, потому что реального пространства больше нет. И это приведёт к проблемам. Но если быть внимательным и до этого не доводить - всё будет хорошо.

> The following data was collected using spinning disks as backend, on a system with 32 GB RAM and 12 cores at 2.4Ghz.
>
> Column 'deploy to file system' shows the time it took to copy a directory with ~5GB of data from RAM (/dev/shm) to the filesystem. Column 'copy on file system' shows the time required to make a single copy of the 5GB directory which was deployed in the first step. The time was averaged over multiple runs, always after removing the data and emptying the file system caches. Publicly available texts from the Project Gutenberg were used as data, quite nicely compressible.

filesystem backend	deploy to file system	copy on file system
XFS ontop of normal LVM volume	28sec	35sec
XFS on VDO device, async mode	55sec	58sec
XFS on VDO device, sync mode	71sec	92sec

Ну и стоит упомянуть падение производительности при использовании VDO. По скрину видно, что просадка довольно таки значительная, если без VDO с XFS на LVM копирование 5 гигабайт занимает 28 секунд, то с VDO в асинхронном режиме - 55 секунд, а в синхронном и вовсе 71. При синхронном режиме VDO ждёт ответа от диска о том, что данные на него записались. Но суть в том, что диск может немного соврать об этом, потому что он может перед записью использовать кэш. Только вот если сервер вдруг вырубится, то

данные в кэше пропадут. Тут, конечно, есть нюансы - кэш может быть с батарейками или диск со сквозным кэшэм - но, в любом случае, sync стоит ставить только если вы уверены в том, что делаете. При асинхронном режиме VDO не гарантирует файловой системе или приложению запись на диск, тем самым они сами заботятся о сохранности. По умолчанию же используется режим auto, который сам подбирает нужный режим в зависимости от накопителя.



Сам VDO ставится поверх блочного устройства, ниже файловой системы и LVM. Если у вас RAID - VDO нужно ставить поверх рейда. Если зашифрованный раздел - VDO ставится поверх него, иначе он не увидит что там дедуплицировать и сжимать. У VDO есть ряд требований к размеру диска и оперативке, с которыми вы можете ознакомиться по ссылке. Например, для диска размером до терабайта нужно примерно 700 мегабайт оперативки под задачи VDO и 2.5 гигабайта пространства под метаданные.

```
[user@centos8 ~]$ sudo fdisk -l /dev/sdd
Disk /dev/sdd: 6 GiB, 6442450944 bytes, 12582912 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[user@centos8 ~]$ sudo dnf install vdo kmod-kvdo -y
Last metadata expiration check: 0:14:51 ago on Fri 26 Mar 2021 03:53:34 PM +04.
Package vdo-6.2.3.114-14.el8.x86_64 is already installed.
Package kmod-kvdo-6.2.3.114-74.el8.x86_64 is already installed.
```

Для теста я добавил нашей виртуалке ещё один диск на 6 гигабайт - минимально необходимое пространство для VDO это почти 5 гигабайт - sudo fdisk -l /dev/sdd. Также нам понадобится установить утилиту и модуль ядра для работы с vdo - sudo dnf install vdo kmod-kvdo.

```
[user@centos8 ~]$ ls -l /dev/disk/by-id | grep sdd
lrwxrwxrwx. 1 root root  9 Mar 26 16:05 ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6 -> ../
../sdd
[user@centos8 ~]$ ls /dev/disk/by-id/ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6
/dev/disk/by-id/ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6
[user@centos8 ~]$
```

И так, у нас есть диск sdd, на котором требуется развернуть VDO. Но, как мы обсуждали, имя sdd динамически выдаётся udev-ом при включении, т.е. может поменяться при определённых условиях. VDO предпочитает постоянные имена. Попробуем найти sdd по идентификатору - ls -l /dev/disk/by-id/ | grep sdd. Это - идентификатор диска, символическая ссылка, которая постоянно будет вести на нужно устройство, поэтому используем его. Хотя VDO и сам делает попытку найти нужную символическую ссылку, тем не менее мы это сделали за него, чтобы было нагляднее.

```
[user@centos8 ~]$ sudo vdo create --name myvdo --device /dev/disk/by-id/ata-VBOX_HAR
DDISK_VB8210cd9d-e92a63b6 --vdoLogicalSize=12G
Creating VDO myvdo
vdo: ERROR - Kernel module kvdo not installed
vdo: ERROR - modprobe: FATAL: Module kvdo not found in directory /lib/modules/4.18.0-147.8.1.e
l8_1.x86_64
[user@centos8 ~]$
```

При создании VDO указываем желаемое имя - --name myvdo, указываем на устройство --device и указываем логический размер - то что я говорил про thin provisioning. Т.е. физический диск у меня 6 гигабайт, а файловая система будет видеть все 12. Но команда у меня не сработала - как видите, вышла ошибка "Kernel module not installed", а дальше нам показывают, что ядро не нашло модуля в директории /lib/modules/4.18.0-147.

```
[user@centos8 ~]$ find /lib/modules -name vdo
/lib/modules/4.18.0-240.15.1.el8_3.x86_64/weak-updates/kmod-kvdo/vdo
/lib/modules/4.18.0-240.10.1.el8_3.x86_64/extra/kmod-kvdo/vdo
[user@centos8 ~]$ ls /boot/vmlinuz-4.18.0-*
/boot/vmlinuz-4.18.0-147.8.1.el8_1.x86_64  /boot/vmlinuz-4.18.0-240.15.1.el8_3.x86_64
/boot/vmlinuz-4.18.0-147.el8.x86_64
[user@centos8 ~]$ uname -r
4.18.0-147.8.1.el8_1.x86_64
[user@centos8 ~]$ reboot
```

Но так как установка прошла успешно, попытаемся найти модуль вручную - find /lib/modules -name vdo. Как видите, find нашла две директории с более новыми версиями ядра - 240.10 и 240.15 и в них эти модули. Скорее всего установка модуля подтянула новую версию ядра и в директории /boot как раз есть новые версии - ls /boot/vmlinuz-*. На всякий случай проверим текущую версию ядра - uname -r - 147. Значит нам нужно просто перезагрузиться - reboot - чтобы загрузиться с новым ядром. После перезагрузки слетели драйвера виртуалбокса, пришлось переустанавливать их для новой версии - сначала ставим необходимые пакеты - sudo dnf install kernel-headers kernel-devel - после чего с диска запускаем установку гостевых дополнений. Потом опять перезагружаемся.

```
[user@centos8 ~]$ sudo vdo create --name myvdo --device /dev/disk/by-id/ata-VBOX_HAR
DDISK_VB8210cd9d-e92a63b6 --vdoLogicalSize=12G
Creating VDO myvdo
      The VDO volume can address 2 GB in 1 data slab.
      It can grow to address at most 16 TB of physical storage in 8192 slabs.
      If a larger maximum size might be needed, use bigger slabs.
Starting VDO myvdo
Starting compression on VDO myvdo
VDO instance 0 volume is ready at /dev/mapper/myvdo
[user@centos8 ~]$
```

Пробуем команду опять - и всё получается. Утилита нам говорит, что создала slab-ы - т.е. блоки в понимании vdo размером в 2 гигабайта. И с таким размером у нас получится использовать только 16 терабайт пространства, а для больших объёмов требуется указать размер слабов больше. Максимумом являются slab-ы размером 32 гигабайта и пространство объёмом 256 терабайт.

```
[user@centos8 ~]$ sudo mkfs.xfs /dev/mapper/myvdo
meta-data=/dev/mapper/myvdo      isize=512    agcount=4, agsize=786432 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=3145728, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

Давайте сразу на него запишем файловую систему - sudo mkfs.xfs /dev/mapper/myvdo.

```
[user@centos8 ~]$ sudo blkid /dev/mapper/myvdo
/dev/mapper/myvdo: UUID="124403d9-8b46-48f2-8769-f026c5c218ed" TYPE="xfs"
[user@centos8 ~]$ sudo nano /etc/fstab
[user@centos8 ~]$ tail -1 /etc/fstab
UUID="124403d9-8b46-48f2-8769-f026c5c218ed" /backups xfs defaults,x-systemd.requires=vdo.service 0 0
[user@centos8 ~]$
```

После чего выясним UUID - sudo blkid /dev/mapper/myvdo - и добавим строчку в fstab. Обратите внимание на опцию монтирования - x-systemd.requires=vdo.service. Она нужна, чтобы монтировать эту файловую систему только если сервис vdo запущен.

```
[user@centos8 ~]$ sudo mkdir /backups
[user@centos8 ~]$ sudo mount /backups
[user@centos8 ~]$ df -h /backups
Filesystem          Size  Used Avail Use% Mounted on
/dev/mapper/myvdo    12G  118M   12G   1% /backups
[user@centos8 ~]$
```

Попробуем создать директорию и примонтировать - sudo mkdir /backups; sudo mount /backups; df -h /backups. Как видите, всё примонтировалось.

```
[user@centos8 ~]$ sudo vdostats myvdo --human-readable
Device                    Size      Used Available Use% Space saving%
myvdo                     6.0G      4.0G      2.0G  66%           98%
[user@centos8 ~]$
```

Теперь давайте протестируем. Например, сделаем несколько бэкапов домашней директории. Но перед этим проверим статистику - sudo vdostats myvdo --human-readable. Несмотря на то, что пока здесь нет никаких данных, vdo уже зарезервировал 4 гигабайта места для информации о дедупликации. Поэтому нам остаётся всего 2 гигабайта, хотя df нам показал все 12.

```
[user@centos8 ~]$ sudo vdo
activate              disableDeduplication  list                  status
changeWritePolicy     enableCompression     modify                stop
create                enableDeduplication   printConfigFile
deactivate            growLogical           remove
disableCompression    growPhysical          start
[user@centos8 ~]$ sudo vdo
```

Но, как я уже говорил, нужно быть осторожным и не допускать заполнения этих двух гигабайт, либо увеличивать размер диска под vdo. Вообще у VDO множество возможностей - останавливать или запускать дедупликацию и компрессию, менять размер логического и физического диска, менять тип записи с синхронного на асинхронный и наоборот и т.д. Это вполне понятные ключи, но советую быть осторожным и предварительно прочесть документацию, прежде чем выполнять эти операции.

```
[user@centos8 ~]$ sudo tar -czf /backups/user.tar.gz /home/user
tar: Removing leading `/' from member names
[user@centos8 ~]$ sudo vdostats myvdo --hu
Device                    Size      Used Available Use% Space saving%
myvdo                     6.0G      4.2G      1.8G  69%           5%
[user@centos8 ~]$ sudo tar -czf /backups/user2.tar.gz /home/user
tar: Removing leading `/' from member names
[user@centos8 ~]$ sudo vdostats myvdo --hu
Device                    Size      Used Available Use% Space saving%
myvdo                     6.0G      4.2G      1.8G  69%          48%
[user@centos8 ~]$ du -h /backups/*
179M    /backups/user2.tar.gz
179M    /backups/user.tar.gz
[user@centos8 ~]$
```

И так, давайте приступим. Для начала создадим один бэкап - sudo tar -czf /backups/user.tar.gz /home/user - и посмотрим статистику - sudo vdostats myvdo --hu. Как видите, почти 200 мегабайт потратилось под бэкап. Теперь попробуем сделать ещё один бэкап - sudo tar -czf /backups/user2.tar.gz /home/user. Проверим ещё раз статистику - а там всё равно 200 мегабайт, при этом обратите внимание, как сильно выросло значение Space saving. В итоге у нас получилось два файла, вместе они занимают почти 400 мегабайт - du -h /backups/* - но благодаря vdo второй файл практически не занимает пространства. Да, в реальных условиях вряд ли все файлы будут одинаковы, но так как дедупликация работает на блочном уровне, в определённых случаях получится немало сэкономить.

```
[user@centos8 ~]$ sudo vdo status -n myvdo
VDO status:
  Date: '2021-03-26 17:10:57+04:00'
  Node: centos8
Kernel module:
  Loaded: true
  Name: kvdo
  Version information:
    kvdo version: 6.2.3.114
Configuration:
  File: /etc/vdoconf.yml
  Last modified: '2021-03-26 16:11:12'
VDOs:
  myvdo:
    Acknowledgement threads: 1
    Activate: enabled
```

```
[user@centos8 ~]$ sudo vdo status -n myvdo | tail -1
        write policy: async
[user@centos8 ~]$
```

Более детальную информацию можно узнать с помощью ключа status - sudo vdo status -n myvdo, например, о типе записи.

```
VDO(8)                          System Manager's Manual                          VDO(8)

NAME
       vdo - manage kernel VDO devices and related configuration information

SYNOPSIS
       vdo  {  activate  |  changeWritePolicy  |  create  |  deactivate  |  disableCompression  |
            disableDeduplication |  enableCompression  |  enableDeduplication  |  growLogical  |
            growPhysical  | import | list | modify | printConfigFile | remove | start | status |
            stop | version } [ options... ]

DESCRIPTION
       The commands available are:
```

Напоследок, посоветую не забывать пользоваться man, где есть и примеры создания, и то что нужно указывать в fstab и информация по многим ключам.

Подведём итоги. Сегодня мы с вами разобрали VDO - технологию для дедупликации и сжатия данных. Её легко развернуть, но нужно быть внимательным и учитывать требования по оперативке, по занимаемому пространству и падение производительности. Это не так эффективно при небольших объёмах данных, но для определённых задач, например, для бэкап хранилища или хранилища для дисков виртуальных машин, эта технология незаменима.