LVM - ArchWiki

5-6 minutes

Thin provisioning

Note: When mounting a thin LV file system, always remember to use the `discard` option or to use fstrim regularly, to allow the thin LV to shrink as files are deleted.

From lvmthin(7):

> Blocks in a standard lvm(8) Logical Volume (LV) are allocated when the LV is created, but blocks in a thin provisioned LV are allocated as they are written. Because of this, a thin provisioned LV is given a virtual size, and can then be much larger than physically available storage. The amount of physical storage provided for thin provisioned LVs can be increased later as the need arises.

Example: implementing virtual private servers

Here is the classic use case. Suppose you want to start your own VPS service, initially hosting about 100 VPSes on a single PC with a 930 GiB hard drive. Hardly any of the VPSes will actually use all of the storage they are allotted, so rather than allocate 9 GiB to each VPS, you could allow each VPS a maximum of 30 GiB and use thin provisioning to only allocate as much hard drive space to each VPS as they are actually using. Suppose the 930 GiB hard drive is `/dev/sdb`. Here is the setup.

Prepare the volume group, `MyVolGroup`.

```
# vgcreate MyVolGroup /dev/sdb
```

Create the thin pool LV, `MyThinPool`. This LV provides the blocks for storage.

```
# lvcreate --type thin-pool -n MyThinPool -l 95%FREE MyVolGroup
```

The thin pool is composed of two sub-volumes, the data LV and the metadata LV. This command creates both automatically. But the thin pool stops working if either fills completely, and LVM currently does not support the shrinking of either of these volumes. This is why the above command allows for 5% of extra space, in case you ever need to expand the data or metadata sub-volumes of the thin pool.

For each VPS, create a thin LV. This is the block device exposed to the user for their root partition.

```
# lvcreate -n SomeClientsRoot -V 30G --thinpool MyThinPool MyVolGroup
```

The block device `/dev/MyVolGroup/SomeClientsRoot` may then be used by a VirtualBox instance as the root partition.

Use thin snapshots to save more space

Thin snapshots are much more powerful than regular snapshots, because they are themselves thin LVs. See Red Hat's guide [4] for a complete list of advantages thin snapshots have.

Instead of installing Linux from scratch every time a VPS is created, it is more space-efficient to start with just one thin LV containing a basic installation of Linux:

```
# lvcreate -n GenericRoot -V 30G --thinpool MyThinPool MyVolGroup
```

```
*** install Linux at /dev/MyVolGroup/GenericRoot ***
```

Then create snapshots of it for each VPS:

```
# lvcreate -s MyVolGroup/GenericRoot -n SomeClientsRoot
```

This way, in the thin pool there is only one copy the data common to all VPSes, at least initially. As an added bonus, the creation of a new VPS is instantaneous.

Since these are thin snapshots, a write operation to `GenericRoot` only causes one COW operation in total, instead of one COW operation per snapshot. This allows you to update `GenericRoot` more efficiently than if each VPS were a regular snapshot.

Example: zero-downtime storage upgrade

There are applications of thin provisioning outside of VPS hosting. Here is how you may use it to grow the effective capacity of an already-mounted file system without having to unmount it. Suppose, again, that the server has a single 930 GiB hard drive. The setup is the same as for VPS hosting, only there is only one thin LV and the LV's size is far larger than the thin pool's size.

```
# lvcreate -n MyThinLV -V 16T --thinpool MyThinPool MyVolGroup
```

This extra virtual space can be filled in with actual storage at a later time by extending the thin pool.

Suppose some time later, a storage upgrade is needed, and a new hard drive, `/dev/sdc`, is plugged into the server. To upgrade the thin pool's capacity, add the new hard drive to the VG:

```
# vgextend MyVolGroup /dev/sdc
```

Now, extend the thin pool:

```
# lvextend -l +95%FREE MyVolGroup/MyThinPool
```

Since this thin LV's size is 16 TiB, you could add another 15.09 TiB of hard drive space before finally having to unmount and resize the file system.

Note: You will probably want to use reserved blocks or a disk quota to prevent applications from attempting to use more physical storage than there actually is.

Customizing

Some customisation is available by editing `/etc/lvm/lvm.conf`. You may find it useful to customize the output of `lvs` and `pvs` which by default does not include the % sync (useful to see progress of conversion between e.g. linear and raid type) and type of logical volume:

```
/etc/lvm/lvm.conf
```

```
report {
        lvs_cols =
"lv_name,lv_attr,lv_active,vg_name,lv_size,lv_layout,lv_allocation_policy,copy_
percent,chunk_size"
        pvs_cols = "pv_name,vg_name,pv_size,pv_free,pv_used,dev_size"
}
```