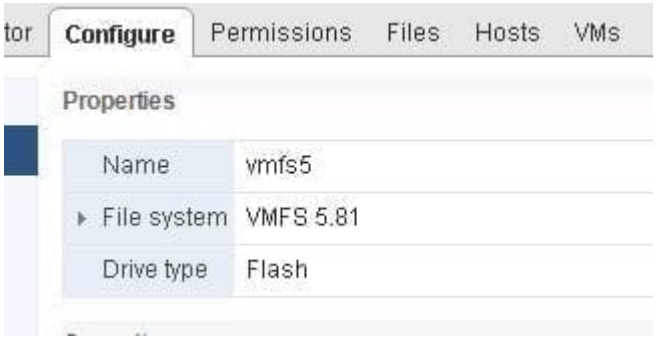
In-Guest UNMAP and VMware Snapshots



My initial answer was “no” but I thought about some changes in VMFS-6 and reconsidered. If you refer to the vSphere 6.5 documentation you can see this change for VMFS 6:

“SEsparse is a default format for all delta disks on the VMFS6 datastores. On VMFS5, SEsparse is used for virtual disks of the size 2 TB and larger”

As indicated in the documentation, VMware snapshots are traditionally in the format called “vmfssparse”. This is somewhat akin to thin virtual disks, but don’t report back to thin to the guest. For larger VMDKs, the snapshot format is “sesparse”, which is really like a thin virtual disk. In VMFS-6, the default for all snapshots is sesparse.

So why does this matter? Well, as noted in the documentation, sesparse supports space reclamation:

“This format is space efficient and supports space reclamation. With space reclamation, blocks that are deleted by the guest OS are marked and commands are issued to the SEsparse layer in the hypervisor to unmap those blocks. This helps to reclaim space allocated by SEsparse once the guest operating system has deleted that data.”

Which in theory, means you could still do in-guest UNMAP when a VMware snapshot is in existence.

In reality, it does not work. The main reason is that the block size for sesparse is 4k and VMFS is 1 MB. So UNMAP translation is a bit tougher than a standard thin virtual disk. I will keep a close eye on this though to see if this changes. I got this answer after I did the testing, and I was just going to delete the post, but decided to keep it and publish my doomed from the beginning blog post regardless.

One final point, with VVols all of this goes away because snapshots are handled differently. So I think it would be better to move to VVols than wait for this to be changed (as it may never be).

To see my testing, read on:

For those of you not familiar with in-guest UNMAP, the process is as such:

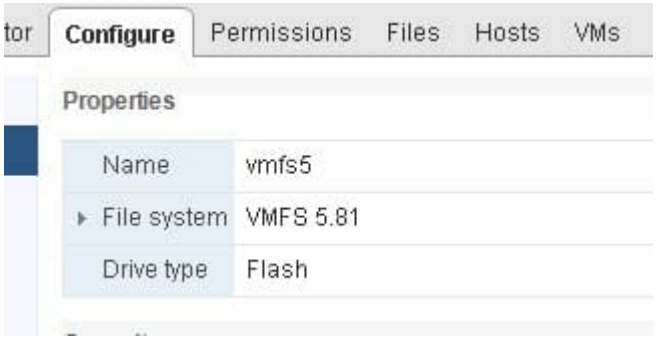
1. You delete a file on a file system in a guest
2. The guest (in one way or another) issues UNMAP

3. ESXi intercepts that UNMAP and then shrinks the thin virtual disk to remove the allocation where the old file used to be
4. Then either automatically, or manually or with 6.5 eventually, ESXi will issue UNMAP to the storage device on the array to actually reclaim the data

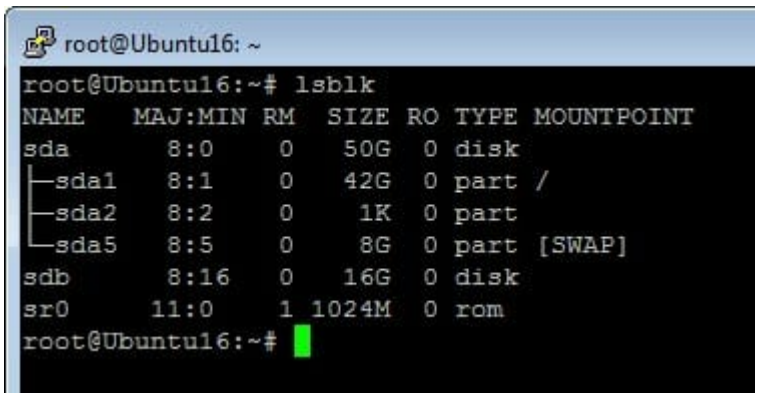
A nice benefit of this feature possibly existing with snapshots would be the smaller the snapshot is, the less space would have to be reconsolidated back to original VMDK when you delete the snapshot. This reduces the performance hit duration during reconsolidation. Especially if a lot of the data written during the existence of the snapshot.

VMFS-5 and VMFSSparse Snapshots

I have a datastore is named" vmfs5" which is of course VMFS version 5:

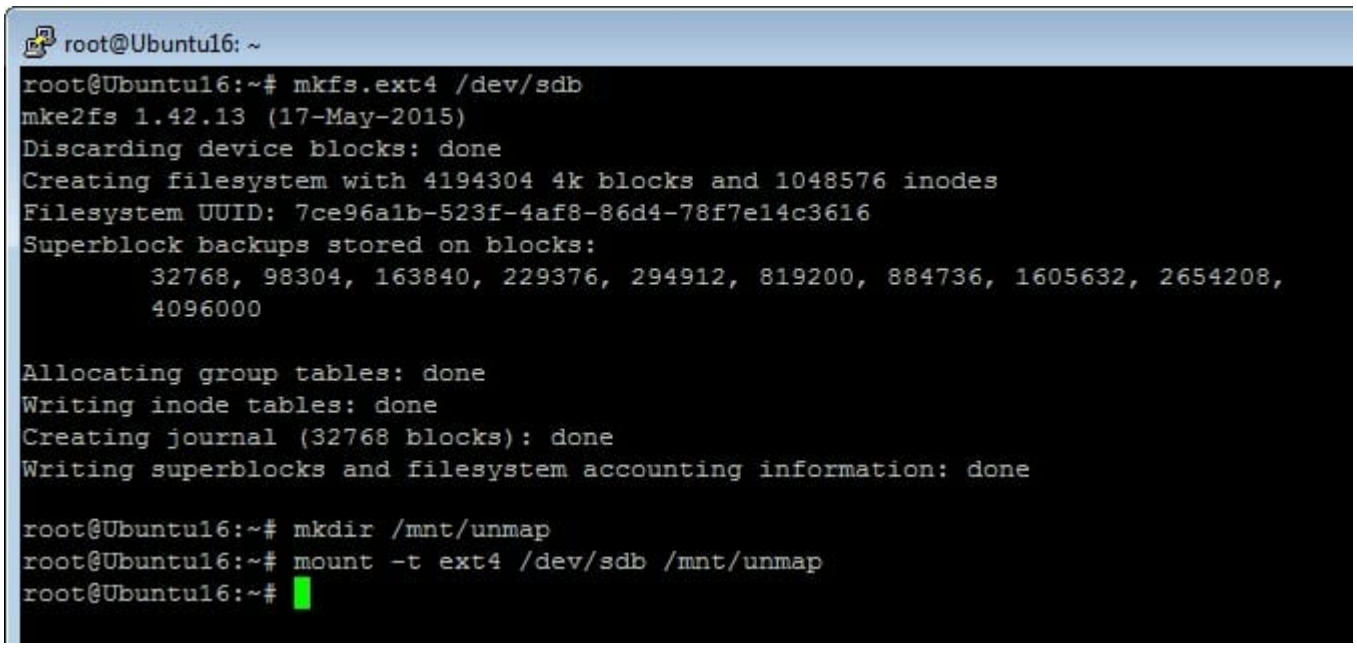


First I will add a 16 GB thin virtual disk to my VM on that datastore:

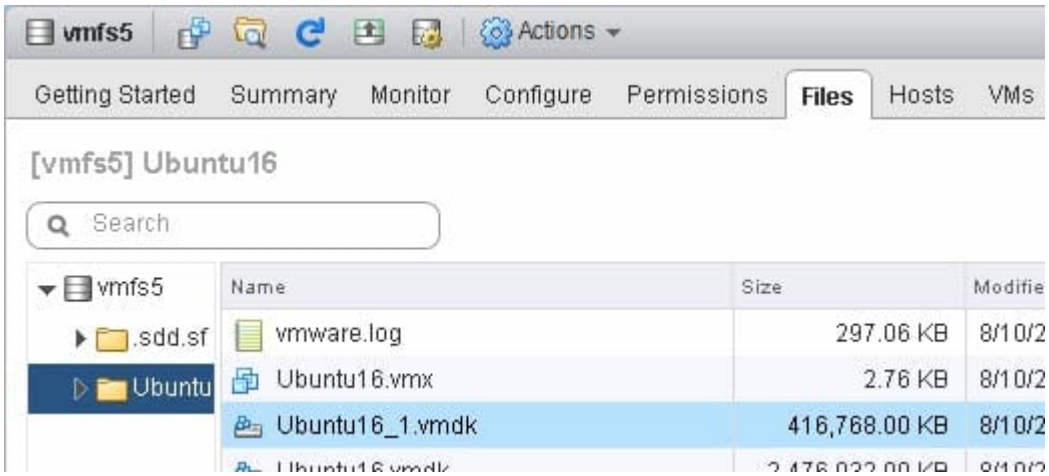


Then put ext4 on it and mount it (this is Ubuntu 16.04–note that Linux only works with in-guest UNMAP with

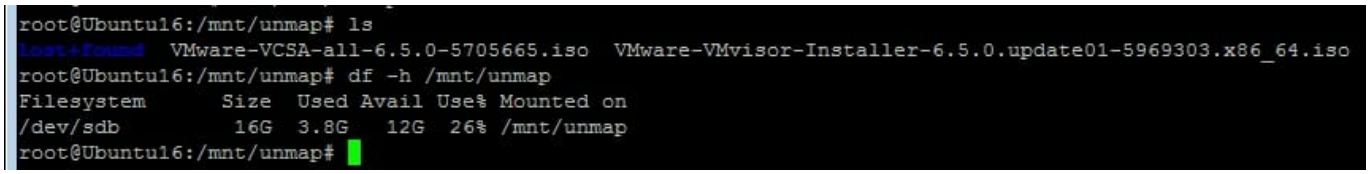
ESXi 6.5):
[crayon-64278dabe5c43602648838/]



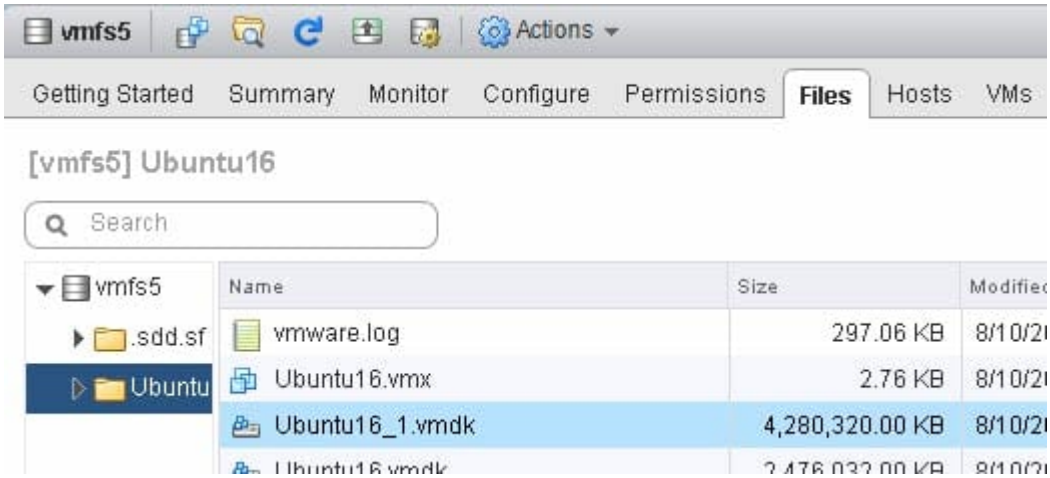
I have sg3_utils installed so I will run sg_vpd to see the UNMAP support of the device in the “logical block provisioning” page:
[crayon-64278dabe5c50091467721/]
UNMAP command supported (LBPU) is set to 1, meaning “yup UNMAP is supported”. After the format etc, my virtual disk reports as 400 MB:



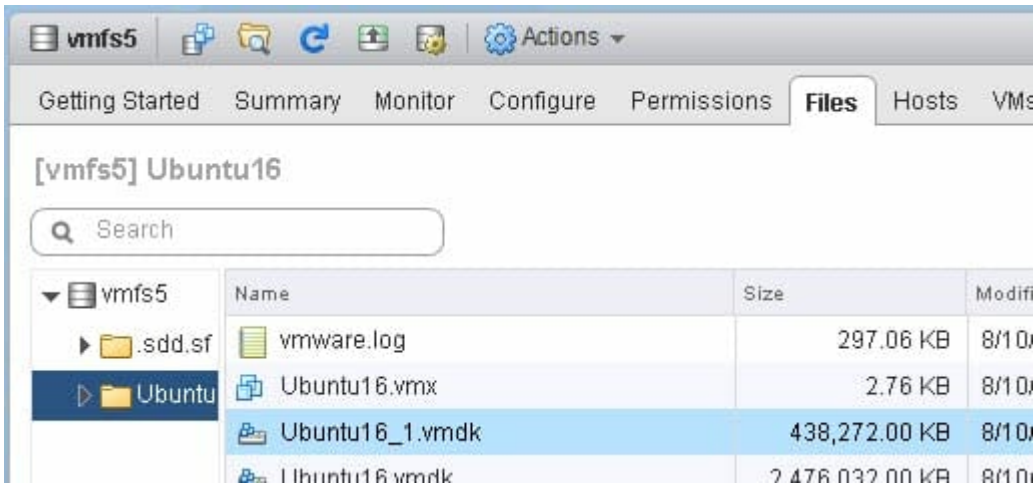
So let’s put some data on it. A few ISOs.



Now the VMDK is consequently larger since data has been written to it:

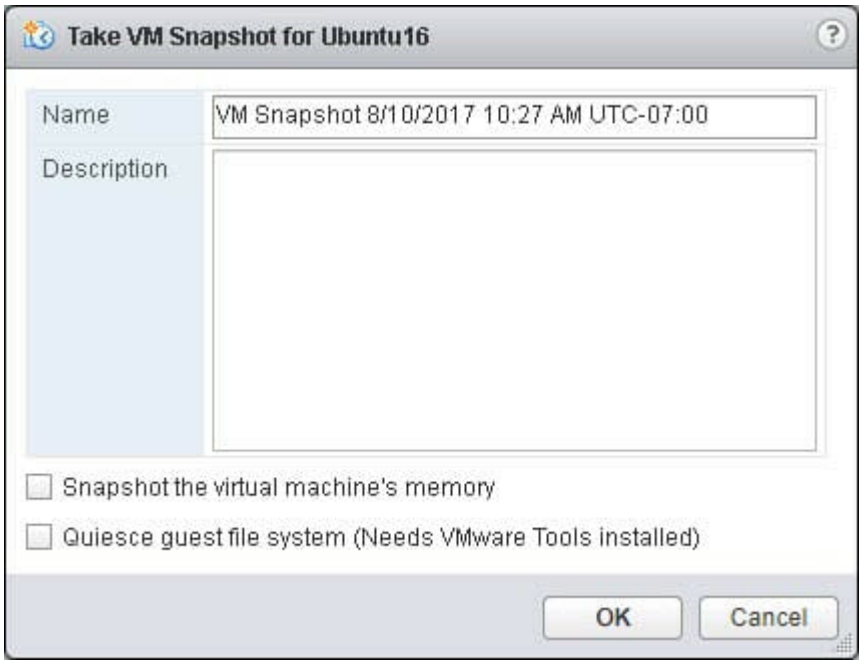


I will now delete the ISOs:
[crayon-64278dabe5c52083834955/]
I can now run unmap (well really trim) via fstrim:
[crayon-64278dabe5c53772663992/]
We can see my VMDK has now shrunk back to the original size:



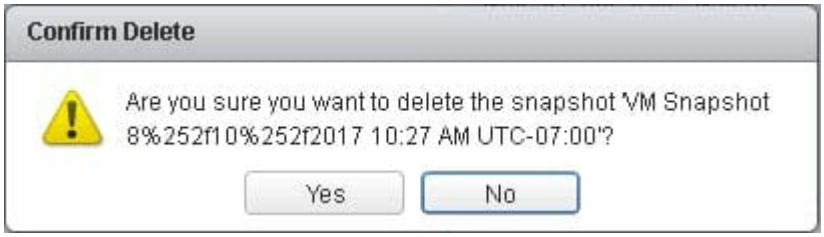
So yay we proved it works in normal fashion.

Now, let's take a VMware snapshot:



Then let's rerun the inquiry.
[crayon-64278dabe5c54504855773/]
The unmap support has gone to 0 and so has the provisioning type (type 2 means thin, 0 means thick). The reason for this is that you are no longer running from the thin virtual disk, the guest is now running from the snapshot which is a different type (vmfssparse) so the VPD information changes and it no longer supports unmap.

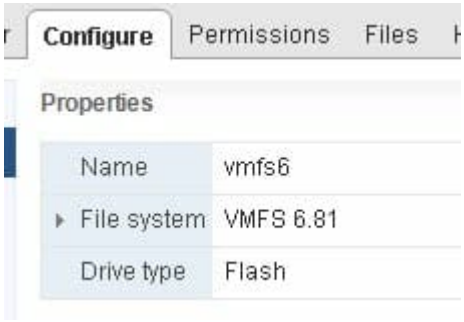
Accordingly, if I add my ISOs back, then delete them, and then try to run fstrim again, you will get an error:
[crayon-64278dabe5c58261392662/]
Now if I delete the snapshot:

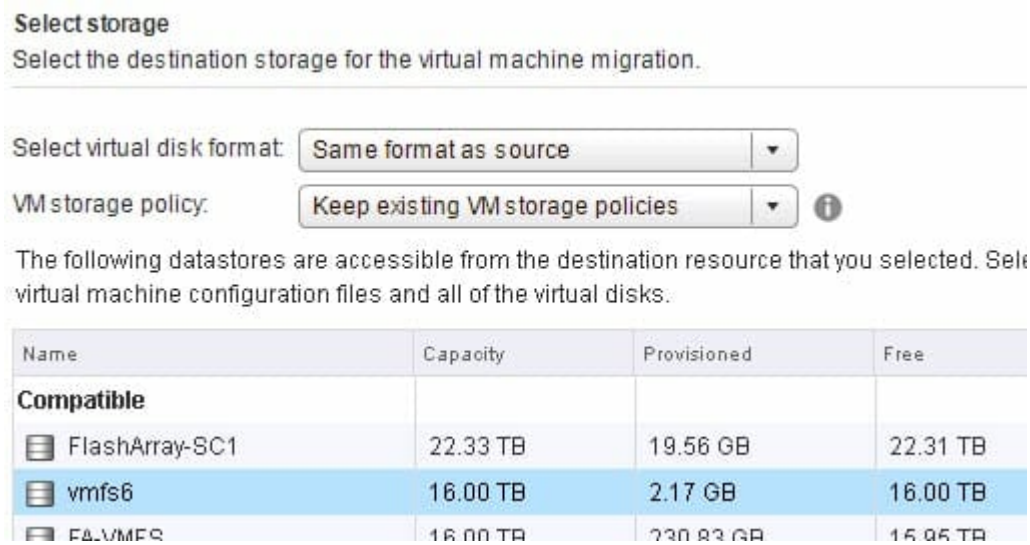


And try fstrim again, it will work.

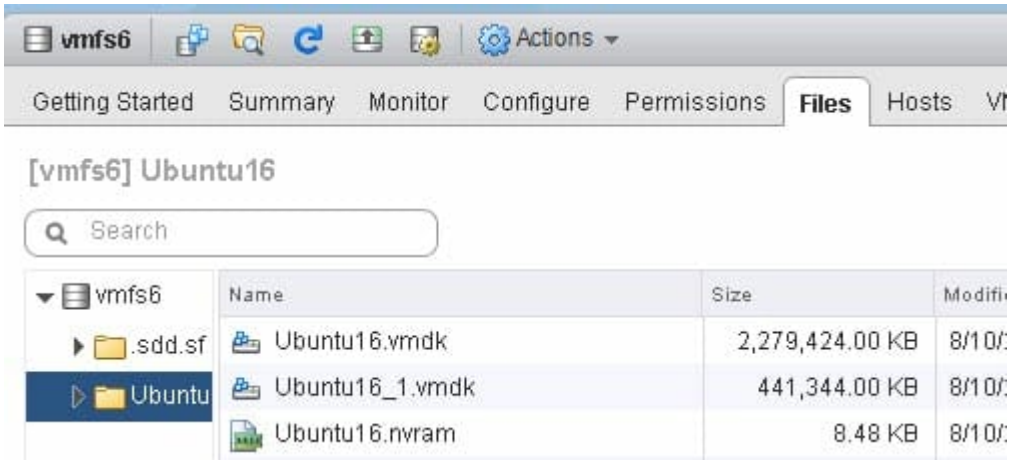
VMware 101: VMFS-6 and SESparse Snapshots

Now, let's try the process with VMFS 6. I will Storage vMotion the VM to my VMFS-6 version datastore fittingly named "vmfs6"

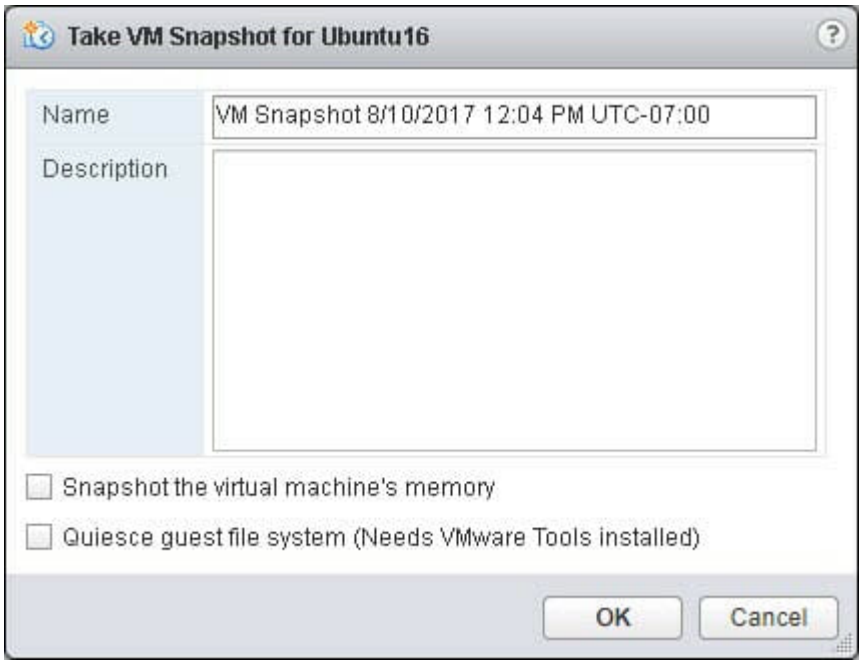




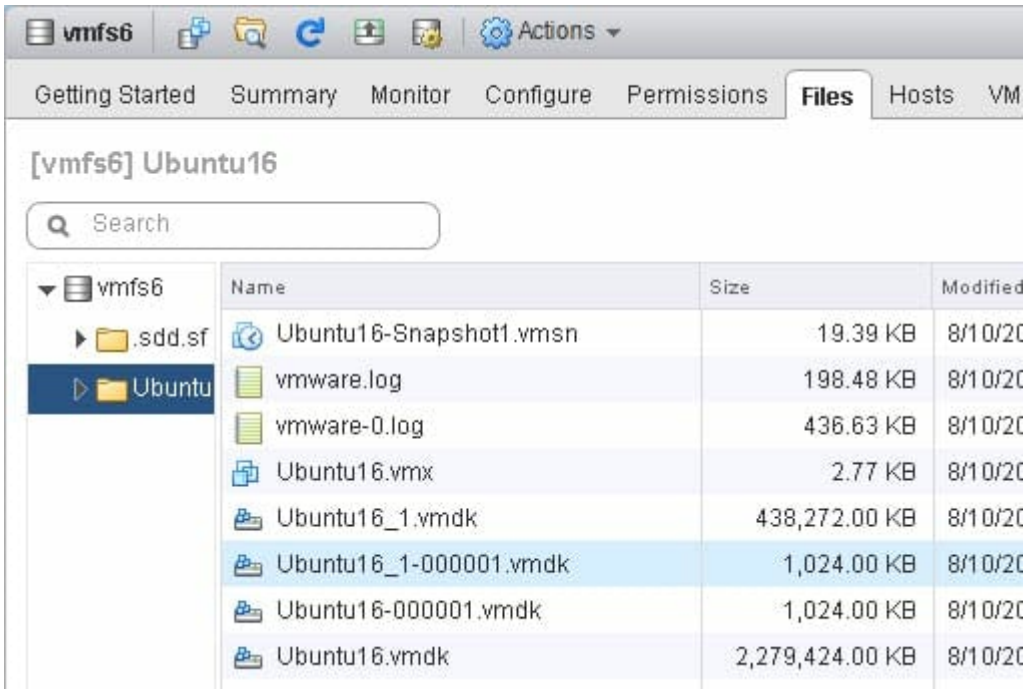
Now we can see my VM is on VMFS-6 and my second virtual disk is back at the 400 MB size



So now if I take a snapshot:



We see my new delta vmdks created as normal, one for each of my two virtual disks (the -000001 files):



If I run the inquiry on the device, we will see that it still supports UNMAP:
[crayon-64278dabe5c59938330138/]
This is different than before, because while we are no longer running off of the thin virtual disk, we are running off of the SESparse virtual disk which supports UNMAP.

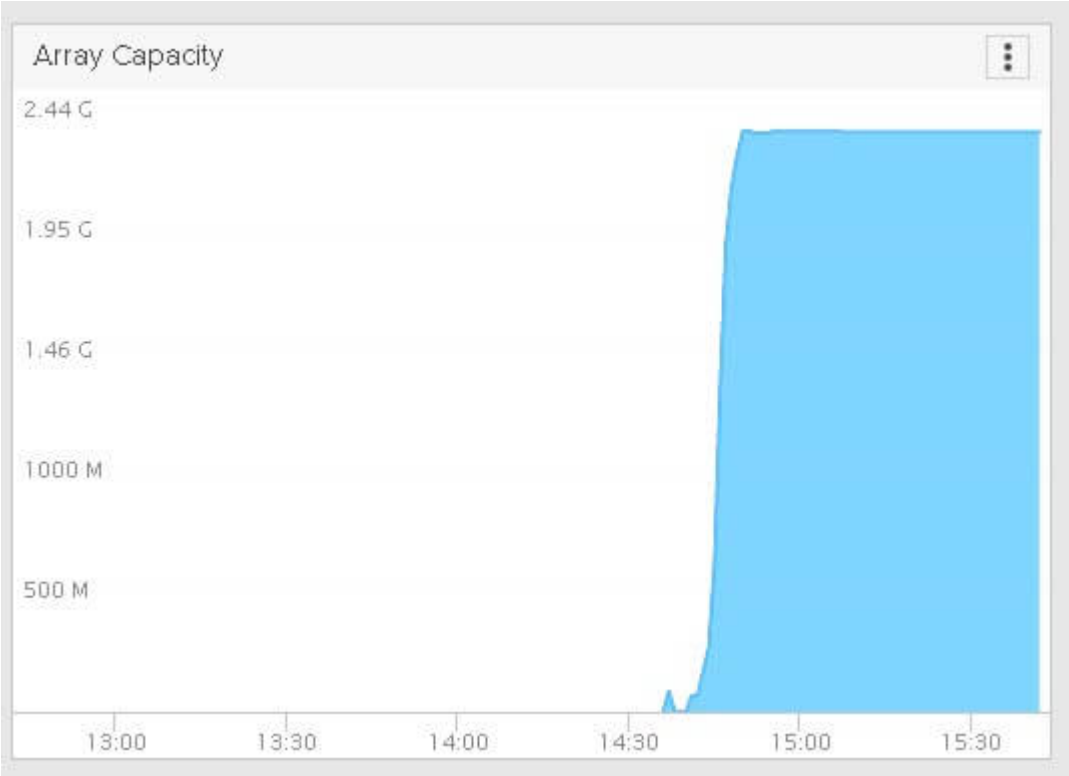
Now if I put some data on that snapshot, you will see the snapshot grow:

My snapshot is now ~7GB, while of course the original VMDK is still at 400 MB. If I now delete my data that I put on, that snapshot is still storing the space, because VMware doesn't know the files have been deleted. This is what UNMAP solves.

```
root@Ubuntu16:/mnt/unmap# ls
lost+found  VMware-VCSA-all-6.5.0-5705665.iso  VMware-VCSA-all-6.5.0-5973321.iso
root@Ubuntu16:/mnt/unmap# rm VMware-VCSA-all-6.5.0-5705665.iso VMware-VCSA-all-6.5.0-5973321.iso
root@Ubuntu16:/mnt/unmap#
```

Now run fstrim. It runs.
[crayon-64278dabe5c5a431024028/]
But nothing happens. It just silently works, but nothing actually changes.

I first thought, maybe it is misaligned. If it is all misaligned ESXi doesn't shrink the vmdk but instead just issues zeroes. But looking at the FlashArray capacity of that VMFS, that idea doesn't pan out.



When zeroes are written the FlashArray discards the space, effectively equaling the behavior of UNMAP. No capacity change at all.

So the UNMAPs/trims just fall into a mysterious pit of despair.