# Report

**Pranav Agarwal - 200123040**

**Somya Khandelwal - 200123056**

**Yashvi Panchal - 200123073**

- **The output for each test will be available in the "Outputs" folder of the submission**
- **ZFS and ext4 File Systems have been used**
- **Put the workload files in the vdbench folder**

## ZFS

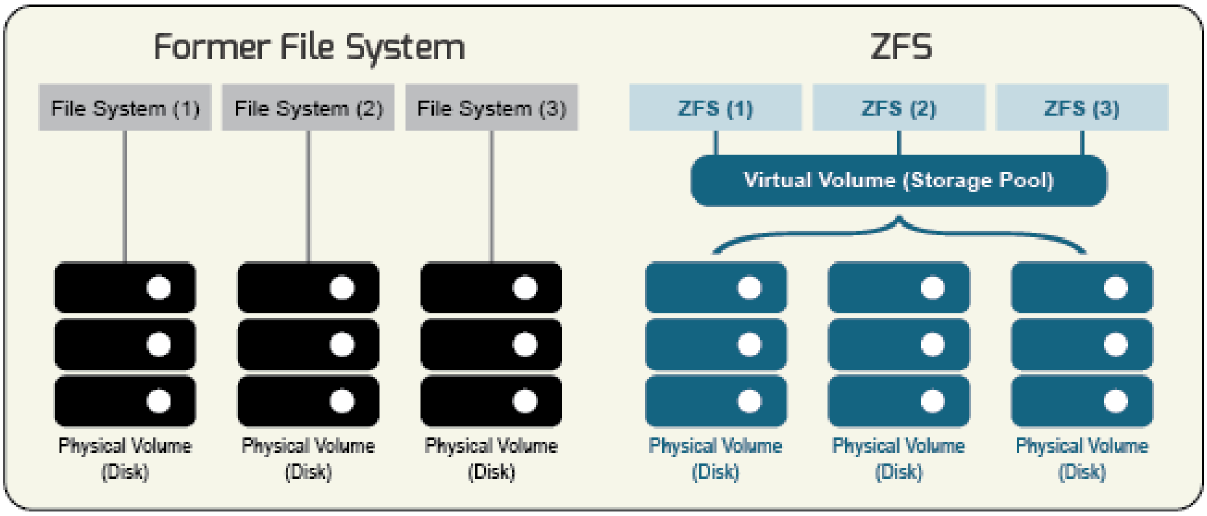
ZFS combines a file system with a volume manager. It began as part of the Sun Microsystems Solaris operating system in 2001 and later acquired by Oracle. However we have used OpenZFS for this lab. In 2013 OpenZFS was founded to coordinate the development of open source ZFS. OpenZFS maintains and manages the core ZFS code, while organizations using ZFS maintain the specific code and validation processes required for ZFS to integrate within their systems. OpenZFS is widely used in Unix-like systems.

ZFS comes with a lot of features like -

- Pooled storage
- Copy-on-write
- Snapshots
- Data integrity verification and automatic repair
- RAID-Z
- Maximum 16 Exabyte file size
- Maximum 256 Quadrillion Zettabytes storage
- Compression
- Encryption

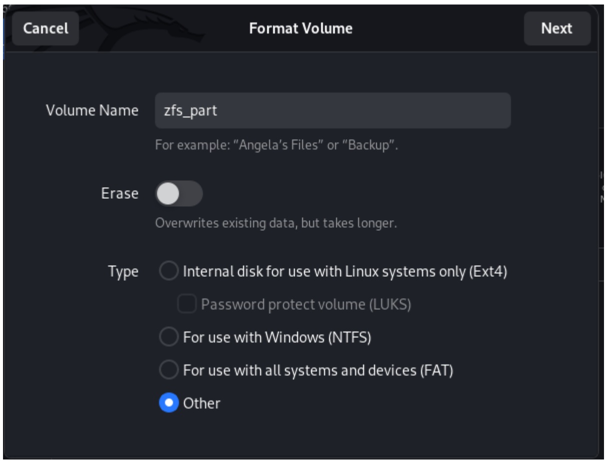
## EXT4

The EXT4 file system primarily focuses on performance and capacity. In this system, data allocation is in the form of extents, instead of fixed size blocks. Extents are described by just their starting and ending places on the hard drive. This form of storing the necessary location of the data in files makes use of the reduces fragmentation of the memory allocated by the EXT4 file system, and thus helps to store the location of data of the file with the help of a small number of pointers, instead of using a pointer pointing to all the blocks of memory occupied by the file. It also makes use of delayed allocation, which helps improve the performance, as well as helps the file system allocate contiguous blocks of memory, as it already knows how much memory it has to map before it starts allocating any memory.



# SETUP STEPS

- First we create 2 partitions in memory, one for ZFS and other for ext4 FS.
- One partition should be mounted without any File System installed and ext4 partition can be initialised directly from the DISKS app in Linux Gnome.



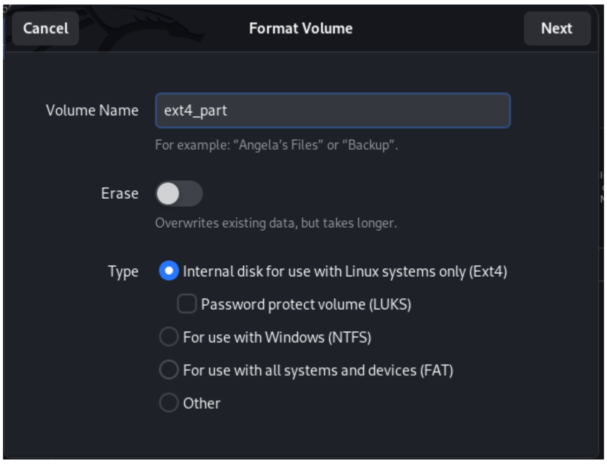
Volume Name: zfs_part

For example: "Angela's Files" or "Backup".

Erase: ○

Overwrites existing data, but takes longer.

Type:
○ Internal disk for use with Linux systems only (Ext4)
□ Password protect volume (LUKS)
○ For use with Windows (NTFS)
○ For use with all systems and devices (FAT)
● Other



Volume Name: ext4_part

For example: "Angela's Files" or "Backup".

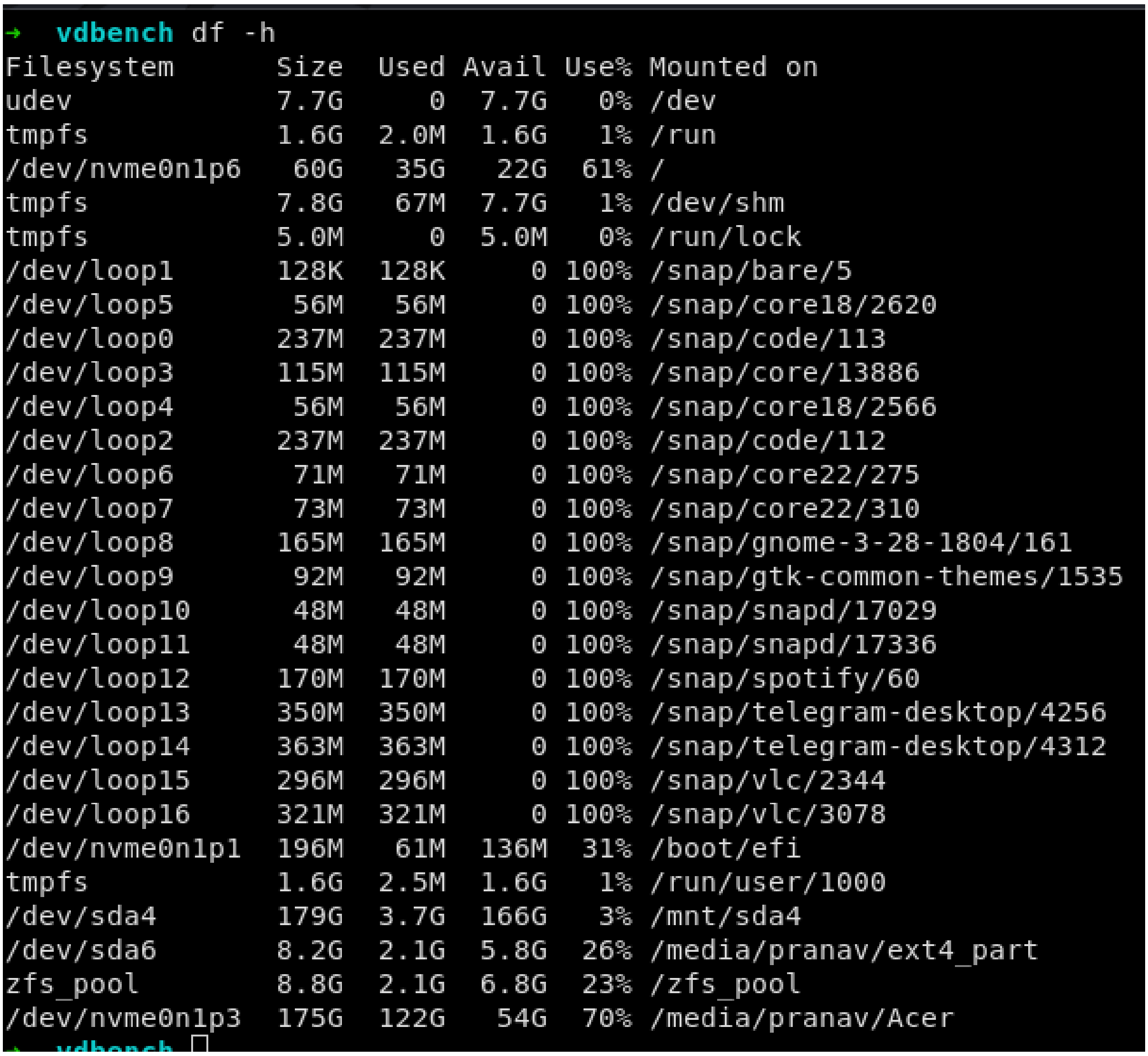
Erase: ○

Overwrites existing data, but takes longer.

Type:
● Internal disk for use with Linux systems only (Ext4)
□ Password protect volume (LUKS)
○ For use with Windows (NTFS)
○ For use with all systems and devices (FAT)
○ Other

- The ZFS partition can now be initialized with the command

```
sudo zpool create zfs_pool /dev/sda5
```



```
→  vdbench df -h
Filesystem       Size  Used Avail Use% Mounted on
udev             7.7G     0  7.7G   0% /dev
tmpfs            1.6G  2.0M  1.6G   1% /run
/dev/nvme0n1p6    60G   35G   22G  61% /
tmpfs            7.8G   67M  7.7G   1% /dev/shm
tmpfs            5.0M     0  5.0M   0% /run/lock
/dev/loop1       128K  128K     0 100% /snap/bare/5
/dev/loop5        56M   56M     0 100% /snap/core18/2620
/dev/loop0       237M  237M     0 100% /snap/code/113
/dev/loop3       115M  115M     0 100% /snap/core/13886
/dev/loop4        56M   56M     0 100% /snap/core18/2566
/dev/loop2       237M  237M     0 100% /snap/code/112
/dev/loop6        71M   71M     0 100% /snap/core22/275
/dev/loop7        73M   73M     0 100% /snap/core22/310
/dev/loop8       165M  165M     0 100% /snap/gnome-3-28-1804/161
/dev/loop9        92M   92M     0 100% /snap/gtk-common-themes/1535
/dev/loop10       48M   48M     0 100% /snap/snapd/17029
/dev/loop11       48M   48M     0 100% /snap/snapd/17336
/dev/loop12      170M  170M     0 100% /snap/spotify/60
/dev/loop13      350M  350M     0 100% /snap/telegram-desktop/4256
/dev/loop14      363M  363M     0 100% /snap/telegram-desktop/4312
/dev/loop15      296M  296M     0 100% /snap/vlc/2344
/dev/loop16      321M  321M     0 100% /snap/vlc/3078
/dev/nvme0n1p1   196M   61M  136M  31% /boot/efi
tmpfs            1.6G  2.5M  1.6G   1% /run/user/1000
/dev/sda4        179G  3.7G  166G   3% /mnt/sda4
/dev/sda6        8.2G  2.1G  5.8G  26% /media/pranav/ext4_part
zfs_pool         8.8G  2.1G  6.8G  23% /zfs_pool
/dev/nvme0n1p3   175G  122G   54G  70% /media/pranav/Acer
→  vdbench
```

```
The mount point of ZFS is /zfs_pool and ext4 is /media/$USER/ext4_part
```

## Feature 1 - Deduplication

- Deduplication is the process of eliminating duplicate copies of data. This can save a lot of space on the hard drive, and is especially useful in some environments where a lot of duplicate data is encountered, with or without minor changes.
- However, this also comes with a tradeoff of high overhead computations, and is thus only recommended to be used in rare scenarios. The deduplication is achieved by hashing(using a secure hash like SHA256) a portion of data to a unique (approximately) signature, and storing these in a hash table.
- The signature of new data is compared to pre-existing values in the hash table, and data with pre-existing signature is deemed to be a copy of the data whose signature matches with it.
- Deduplication can be implemented in different levels, depending on the size of data that gets hashed to a signature, with an increasing amount of tradeoff between overhead computations and space saved due to redundant data not being copied.
- These are file-level, block-level and byte-level. Deduplication can also be synchronous or asynchronous, depending on whether the process happens as the data is being written, or whether the copies are hashed and deleted when the CPU is free.

- ZFS has the deduplication feature, and it uses block-level synchronous deduplication. EXT4 does not support deduplication.

## Deduplication Test

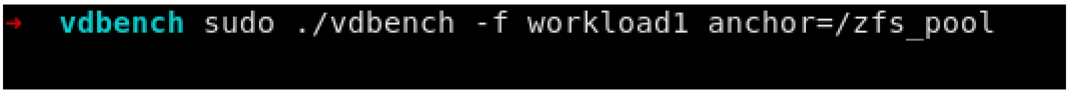
- We created the following workload for data deduplication (workload1). We will use this workload to compare the space occupied by the new files in ZFS with the space occupied in ext4.

```
dedupunit=1m,dedupratio=2
fsd=fsd1,anchor=$anchor,depth=2,width=3,files=50,size=1m
fwd=fwd1,fsd=fsd1,operation=read,xfersize=4k,fileio=sequential,fileselect=random,threads=2
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=30,interval=1
```

- Basically, we are creating 450 files (50 x 3 x 3) each of size 1MB in a nested folder structure of depth 2 and width 3. Then these files are being read sequentially for thirty seconds to monitor statistics (although this part is not important since the deduplication is done during file creation).

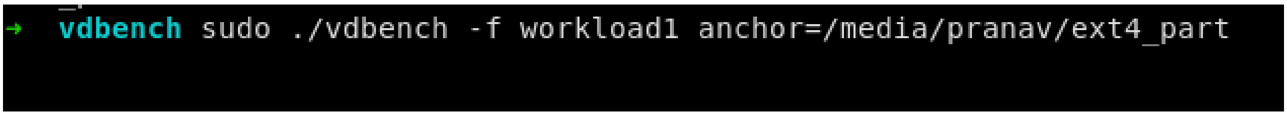
- dedupunit is set to 1MB and dedupratio is set to 2. dedupratio is the ratio of the total number of blocks (of size dedupunit) with the number of blocks containing unique data. dedupunit on the other hand is the size of the block which will be compared with pre-existing blocks to check for duplicates. We set it to 1MB because this is the size of one file. So basically, half of the files will be duplicates of the other half.

- We run this workload on the ZFS file system by setting anchor to the directory of the ZFS Pool (basically the folder pointing to the ZFS Pool):



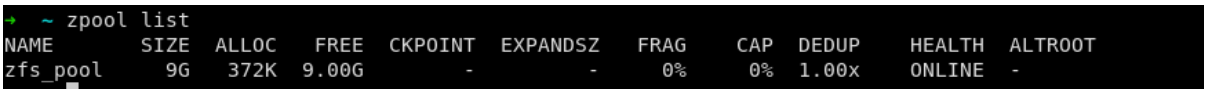
```
➜  vdbench sudo ./vdbench -f workload1 anchor=/zfs_pool
```

- We run this workload on the ext4 file system by setting anchor to the directory of the folder pointing to the ext4 drive:



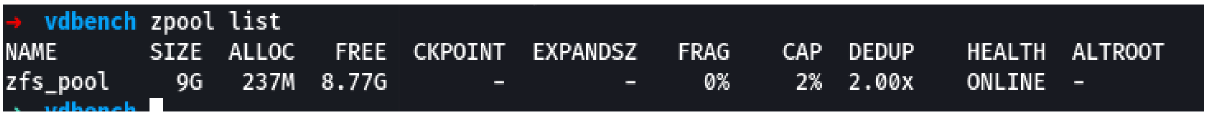
```
➜  vdbench sudo ./vdbench -f workload1 anchor=/media/pranav/ext4_part
```

- We found the following results:
    - ZFS:
        - Initially, the empty ZFS folder had 372 KB of data.
        - After running the workload, the ZFS folder had 230 MB of data.
        - We observed a deduplication ratio of 2.00x (which is what we wanted).
        - This means that the new files took 230 MB of space. However, the intended space is 450MB (1MB*450). Hence, using the data deduplication feature, instead of maintaining whole blocks of data, when duplicates are found, ZFS simply makes a pointer to the old data.
        - **Before Workspace**



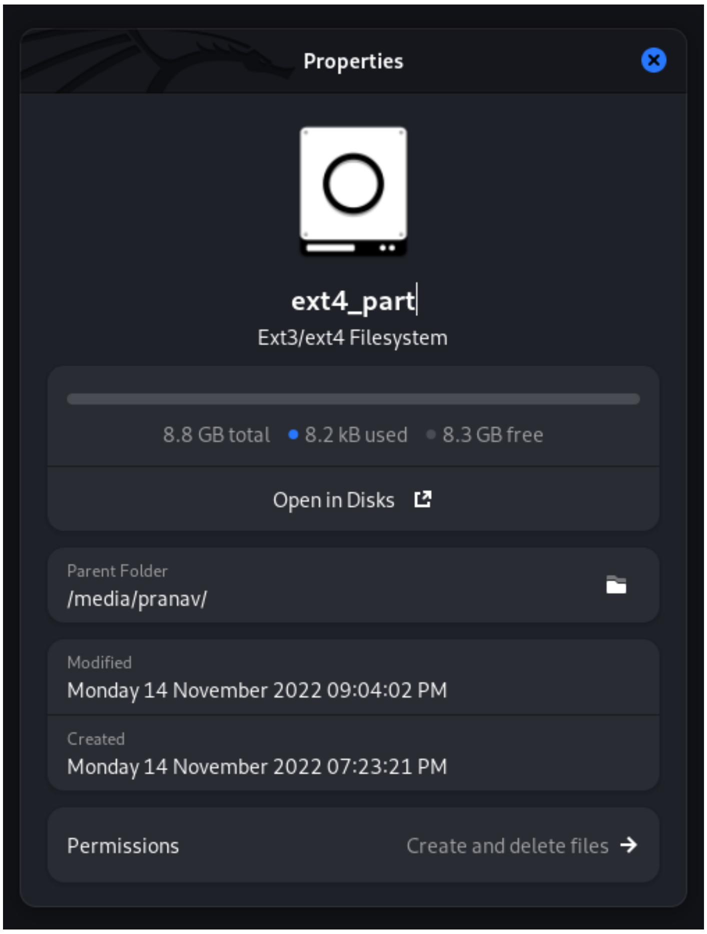
```
➜ ~ zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool      9G   372K  9.00G        -         -     0%     0%  1.00x    ONLINE  -
```

- **After Workspace**



```
➜ vdbench zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool      9G   237M  8.77G        -         -     0%     2%  2.00x    ONLINE  -
➜ vdbench
```

- ext4
  - Initially the empty ext4 folder had 8.2 KB of data.
  - After running the workload, the ext4 folder had 471.9 MB of data.
  - The new files thus took 447 MB of space (a little more than intended because of metadata overhead).
  - **Before Workspace**



Properties

ext4_part

Ext3/ext4 Filesystem

8.8 GB total ● 8.2 kB used ● 8.3 GB free

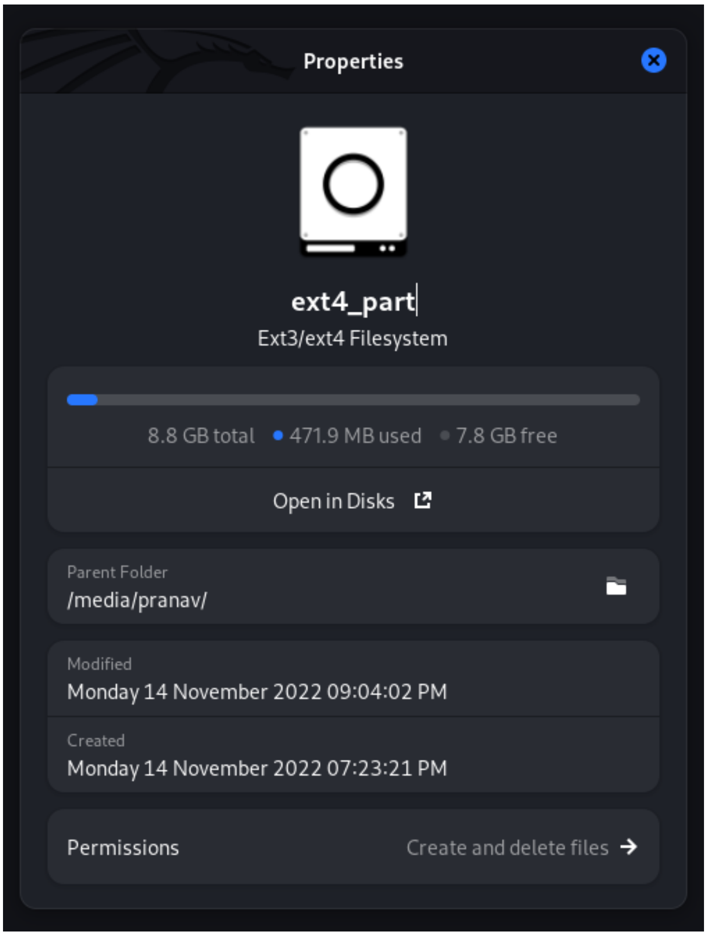
Parent Folder
/media/pranav/

Modified
Monday 14 November 2022 09:04:02 PM

Created
Monday 14 November 2022 07:23:21 PM

Permissions Create and delete files

- **After Workspace**



Properties

ext4_part

Ext3/ext4 Filesystem

8.8 GB total • 471.9 MB used • 7.8 GB free

Open in Disks

Parent Folder
/media/pranav/

Modified
Monday 14 November 2022 09:04:02 PM

Created
Monday 14 November 2022 07:23:21 PM

Permissions — Create and delete files →

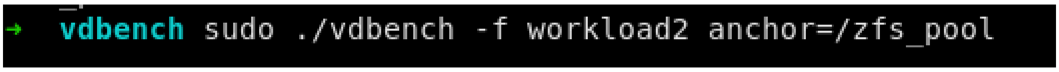
## Large File Creation Test

- We know that ext4 optimizes large file creation better than ZFS does as we can clearly see using our workload.
- We created the following workload for testing large file creation (workload2):

```
fsd=fsd1,anchor=$anchor,depth=0,width=1,files=2,size=1G
fwd=fwd1,fsd=fsd1,operation=create,fileio=sequential,fileselect=random,threads=2
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=30,interval=1
```

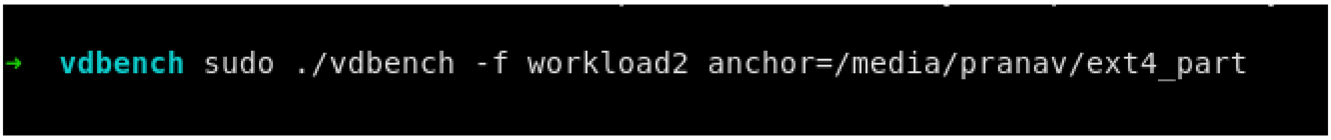
- What we are doing here is creating two files of size 1GB in one folder. The operation used is “create” since we are testing file creation.

- We run this on the ZFS file system by setting anchor equal to the directory pointing to the ZFS pool:



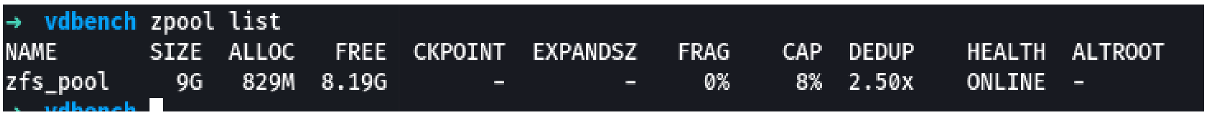
```
➜  vdbench sudo ./vdbench -f workload2 anchor=/zfs_pool
```

- We run this workload on the ext4 file system by setting anchor equal to the directory pointing to the ext4 drive:



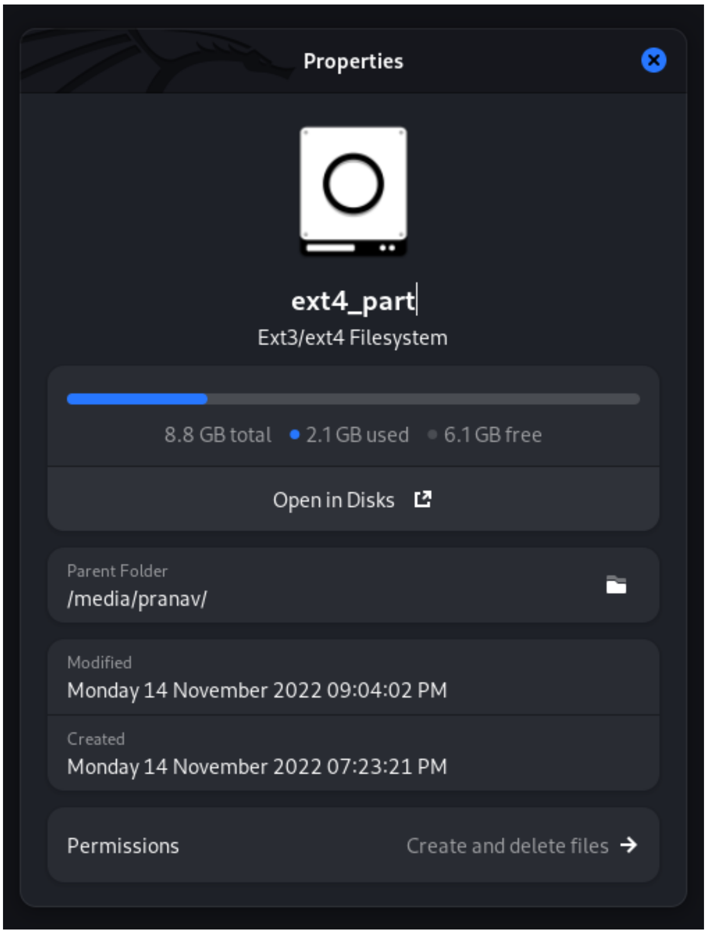
```
➜  vdbench sudo ./vdbench -f workload2 anchor=/media/pranav/ext4_part
```

- We found the following results:
  - ZFS
    - The average write speed is 104 MB/s, which means it completes the file creation in less than 20 seconds



```
→  vdbench zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool      9G   829M  8.19G        -         -     0%     8%  2.50x    ONLINE  -
→  vdbench
```

  - ext4
    - The average write speed is 147 MB/s, which means it completes the file creation in 14 seconds



ext4_part

Ext3/ext4 Filesystem

8.8 GB total • 2.1 GB used • 6.1 GB free

Parent Folder
/media/pranav/

Modified
Monday 14 November 2022 09:04:02 PM

Created
Monday 14 November 2022 07:23:21 PM

Permissions: Create and delete files →

- **The space occupied by the files is only 829 MB in ZFS whereas it is 2.1 GB in ext4**
- **This is due to the deduplication in ZFS**