CS344 Assignment 4

Group M20

Aditya Pandey: 200123004

Aman Bucha: 200123006

Sarthak Diwan: 200123076

The file system we have considered in order to give analysis on different features is the zfs file system.

ZFS

ZFS (Zettabyte file system) is a file system with volume management capabilities. The file system was first introduced as part of the Sun Microsystems Solaris operating system in 2001. During 2005 to 2010 the open source version of ZFS was ported to Linux, Mac OSX(continuing as the MacZFS) and FreeBSD. In 2013, OpenZFS was founded to coordinate the development open source ZFS. OpenZFS maintains and manages the core ZFS code, while organizations using ZFS maintain specific code and validation processes required for ZFS to integrate within their systems. OpenZFS is widely used in Unix-like systems.

ZFS is different from other file systems as unlike most storage systems it unifies the role of both the volume manager and the file system. Thus, it has complete knowledge of both physical disks and volumes and also of the file stored on them. ZFS is designed to ensure that the data stored on disks cannot be lost due to physical errors or misprocessing by the hardware or operating system.

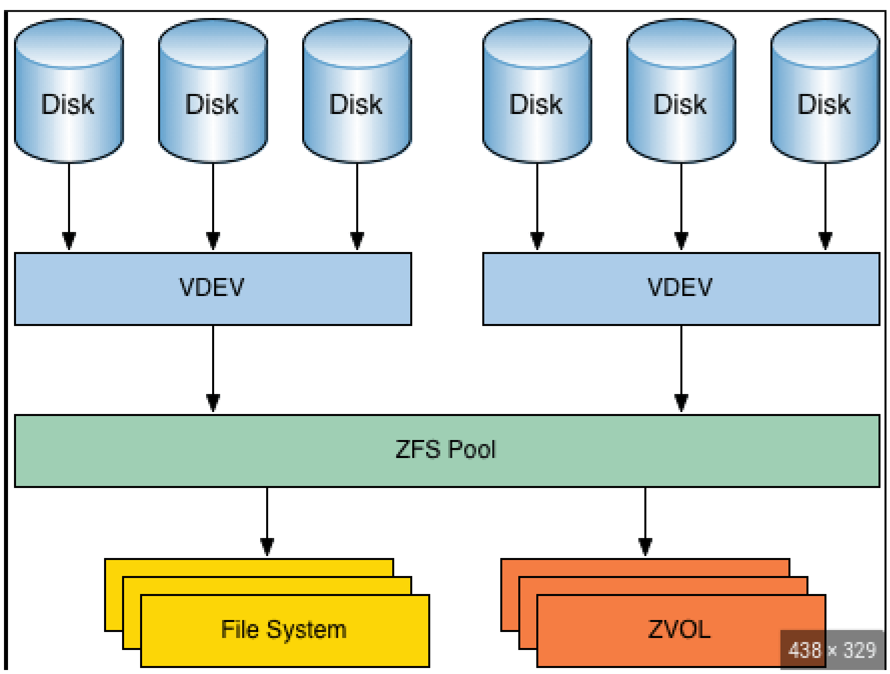
ZFS also includes the mechanism for dataset and pool-level snapshots and replication, including cloning which maybe described as one of its most powerful features, implementing features that other file systems even with the snapshot feature lack.

Some of the features of the ZFS file system can be enumerated in the following manner:

1. Data Integrity

2. RAID-z
3. Capacity
4. Encryption
5. Read/Write Efficiency
6. Caching mechanism
7. Copy on write transactional model
8. Snapshots and clones
9. Dynamic striping
10. Variable block sizes

    and so on…..



The first feature of ZFS we wanted analyse was the deduplication feature:

The researched details of performance for the deduplication feature have been referenced from the following paper:
https://www.usenix.org/system/files/login/articles/1916-galvin.pdf

EXT4

This file system primarily focuses on performance and capacity. In this sytem, data allocation is in the form of extents, instead of fixed size blocks. Extents are described starting and ending places on the hard drive. This reduces fragmentation of memory allocated by the EXT4 file system and thus helps in storing location of 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.

Deduplication

Deduplication works by “thumb printing” in which an entity is checksummed , resulting in a hash value. Hashing is very effective, providing unique values for the data stored in almost all cases. While there are chances for collision, it is fairly reasonable to say that us the hash values for 2 data entities are same then the data stored is same as well. If the hash value for the data is computed and the value already exists in the data deduplication table, a pointer oertaining to the same data is stored rather than a new copy of the data in the same location.

Deduplication is done by file systems in two manners:

1. Performing deduplication on blocks post processing, that is, they store all entities on wrote request and then later compare the entities and remove duplicates. This implementation is used by the NetApp FAS.
2. ZFS implements deduplication at the time of writing. Although this takes a penalty at the time of writing, it is space efficient in terms memory usage in comparison to post processing deduplication

ZFS deduplication, as with other features of ZFS such as compression, only works on data written after the specific feature is enabled. If a lot of data already exists in a ZFS pool, there is no native way to have that deduplicated. Any new data will be deduplicated rather than written, but for the existing data to be deduplicated, that data would need to be copied to another pool (for example) or replicated to a ZFS file system with enabled deduplication.

In zfs, once deduplication is enabled the variable dedupratio shows how much effect deduplication is having on data in a ZFS pool. ZFS has inbuilt file system checksumming. Deduplication enables a stronger checksum for the file system when enabled. By default deduplication uses SHA256. Hashing almost always results in matches only when the hashed entities exactly match. However the almost always and and always have a huge gap between them, meaning there can still be collisions leading to corruption of data.

First we want to experimentally show the difference between zfs with its deduplication feature and ext4 without its deduplication feature.

1. First we need to setup the deduplication feature by creating a zfs file system with the parameter dedup = on.

Large file creation:

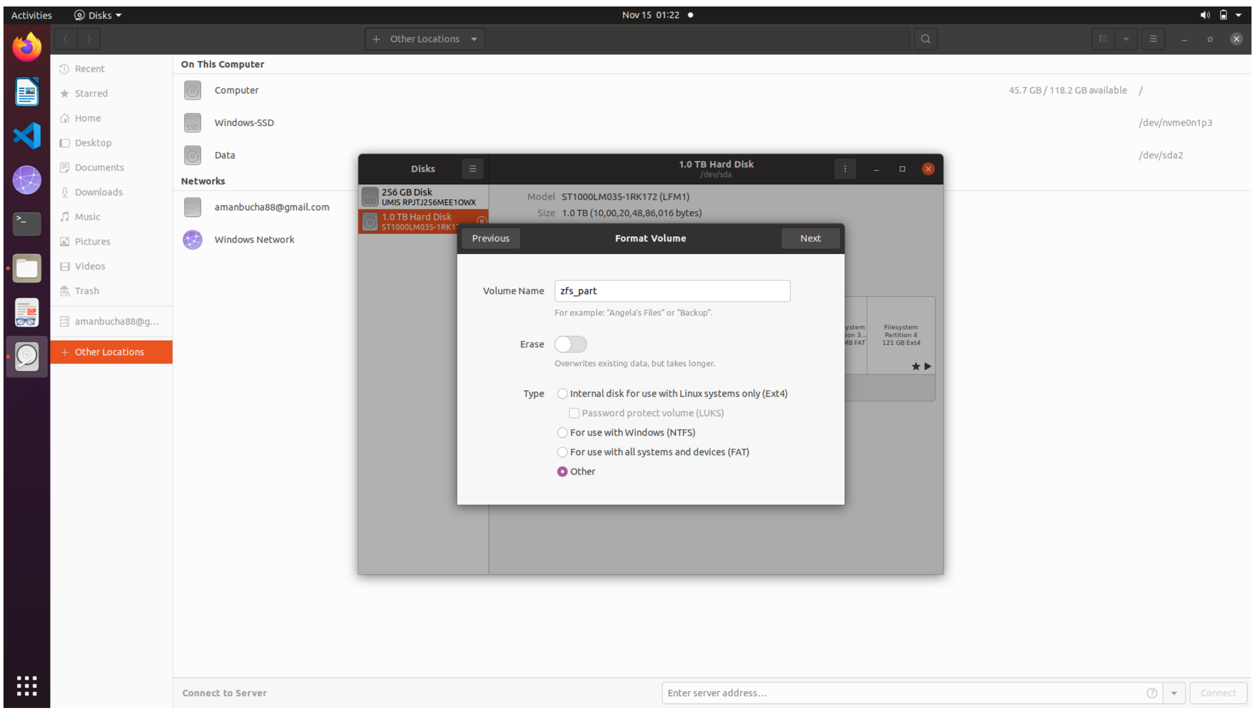
The EXT4 file system allows for a maximum file size of 16 TiB (TebiBytes) - 2^44 Bytes with the common 4 KiB blocks and 48 bit block addressing, as well as a maximum volume of 1 EiB (ExbiByte) = (260) Bytes.
EXT3 only permits file sizes of 2 TiB and a file system size of 16 TiB.
ZFS supports file systems with a 16 TiB size.
Very huge files can be created and handled very efficiently with EXT4.
This is because large files take up a lot of space and time to save and access due to extensive mapping of data blocks.

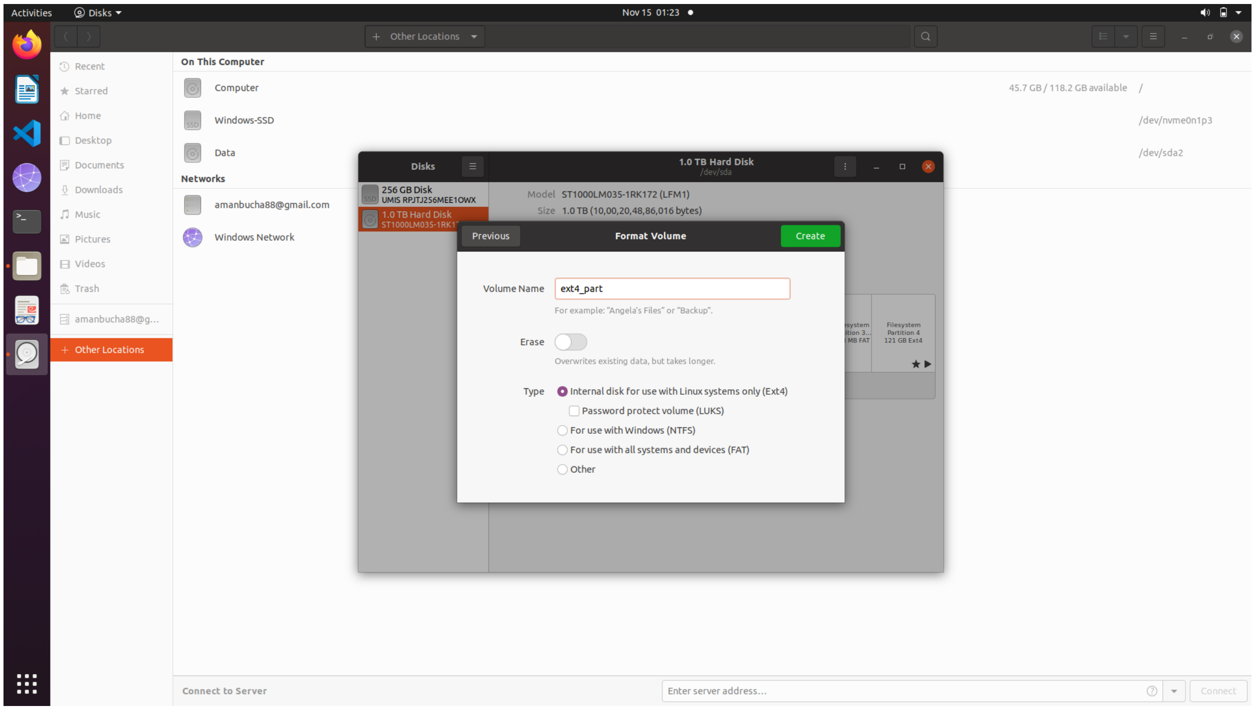
Other features in EXT4 also aid in the proper and effective operation of this new extent-based mapping system.
Multiblock allocation is a feature of EXT4 that allows for easy allocation of contiguous blocks of memory while avoiding a significant amount of cost by allocating many blocks in a single call as opposed to one block each call.

This works in conjunction with delayed allocation, which doesn't write to disc on every write operation instead noting the data to be written before using multiblock allocation to write a sizable amount of data into a contiguous memory segment.

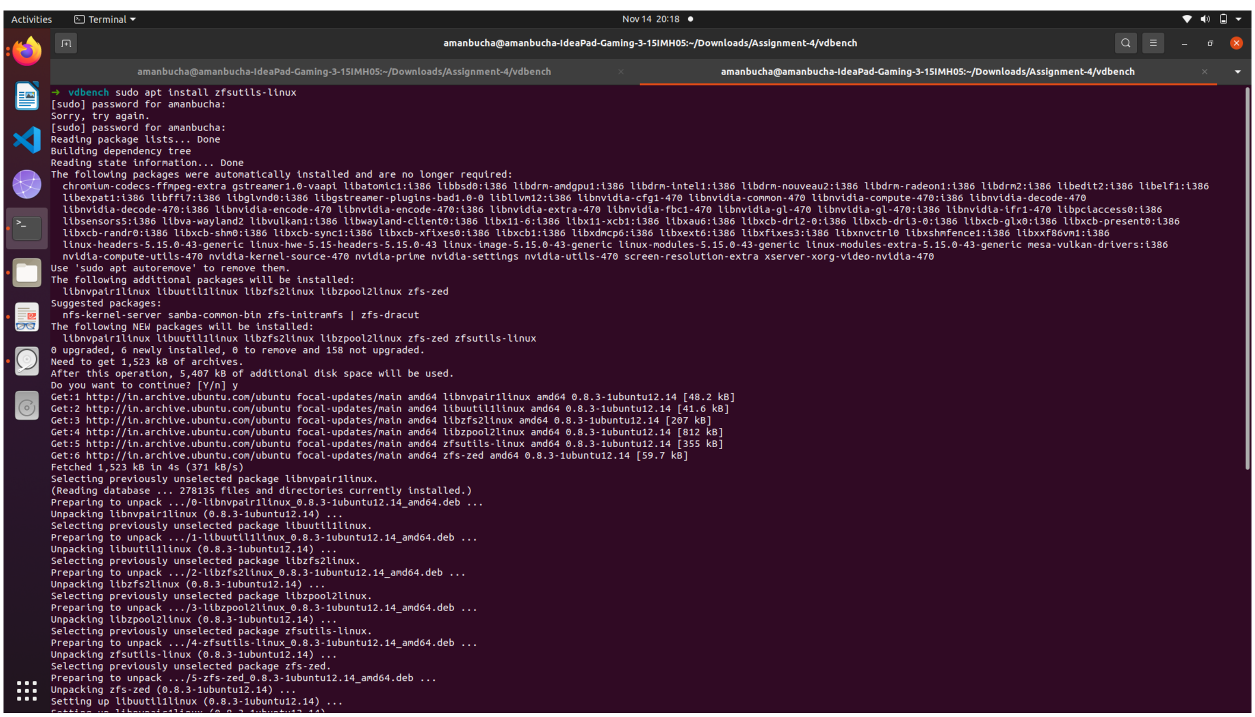
Steps for experiment:

We create the partitions of the disk: zfs_part and ext4_part

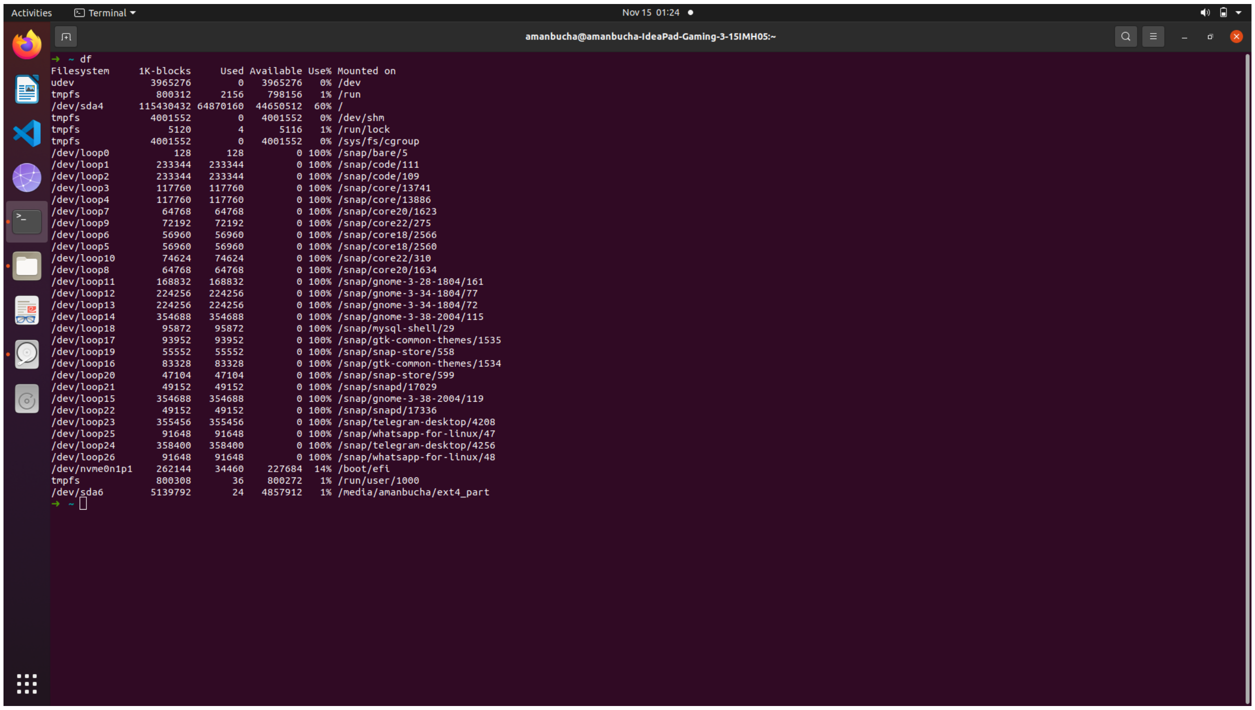




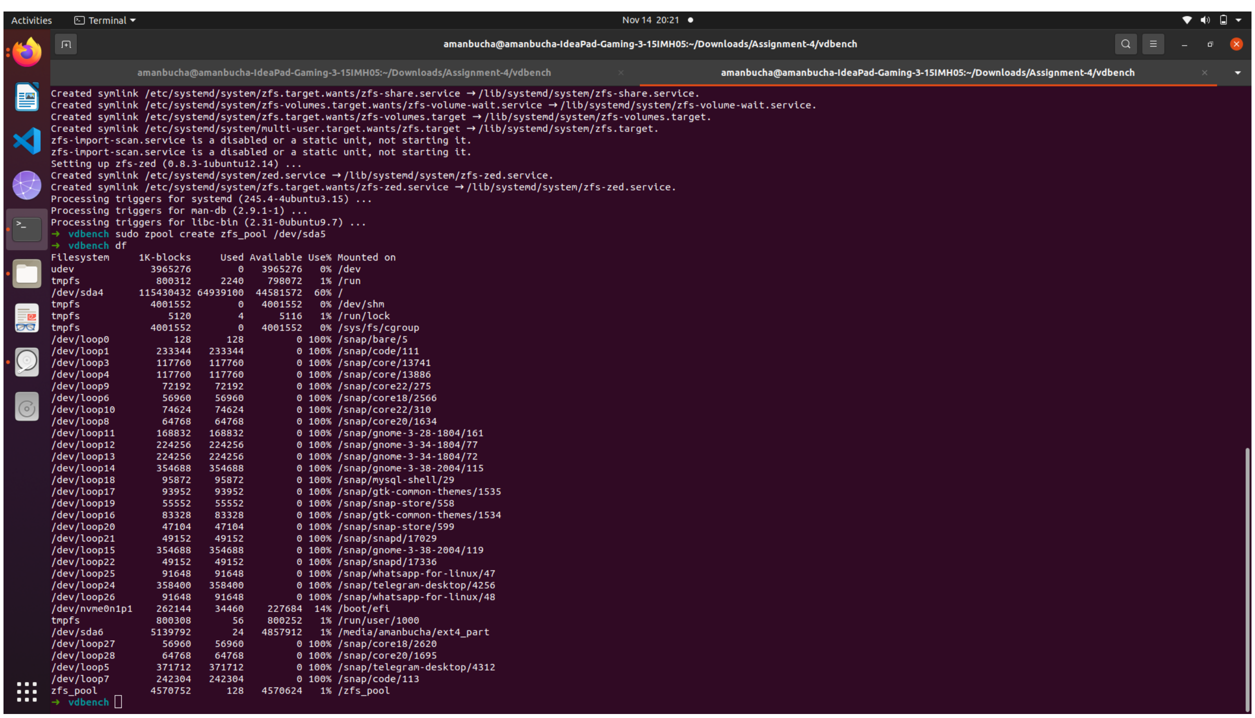
Next, we install zfs file system on the volume zfs_part



We noticed that zfs file system was not listed when we ran **df** command



We corrected this by initialising zfs file system on zfs_part volume:



The mount point of zfs is /zfs_pool

The mount point of ext4 is /media/$USER/ext4_part

We use the following workloads and accordingly compare the two file systems based on the output:

1) Workload 1

workload1
~/Downloads/Assignment-4/vdbench/GM20

```
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
```

2) Workload 2

workload2
~/Downloads/Assignment-4/vdbench/GM20

```
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
```

1. Deduplication

a. There is a data deduplication feature in `zfs` which is turned on by the command

```
sudo zfs set dedup=on zfs_pool
```

b. We now compare the space utilization of `zfs` and `ext4` using the following workload

c. In the workload, we are creating **450 files** (50 * 3 * 3) each of size 1MB in a nested folder with depth = 2 and width = 3. We are then reading the files in sequence for 30 seconds.

d. `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` is the size of the block which will be compared with pre-existing blocks to check for duplicates. So in our case, half of the files are duplicates since the file size is 1MB.

e. Setting anchor to the ZFS Pool directory, we execute this workload on the ZFS file system using the following command.

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

f. Setting anchor to the ext4 drive, we execute this workload on the ext4 file system using the following command.

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

Results:

1. zfs

   - The empty ZFS folder initially contained **396 KB** of data.
   - After the workload was executed, the ZFS folder had **229 MB** of data.
   - This means files took **228.6 MB**
   - We noticed a deduplication ratio of 2.0x, which was the desired value.
   - Therefore instead of **450 MB** the new files took **223 MB** of space.

Before Workload

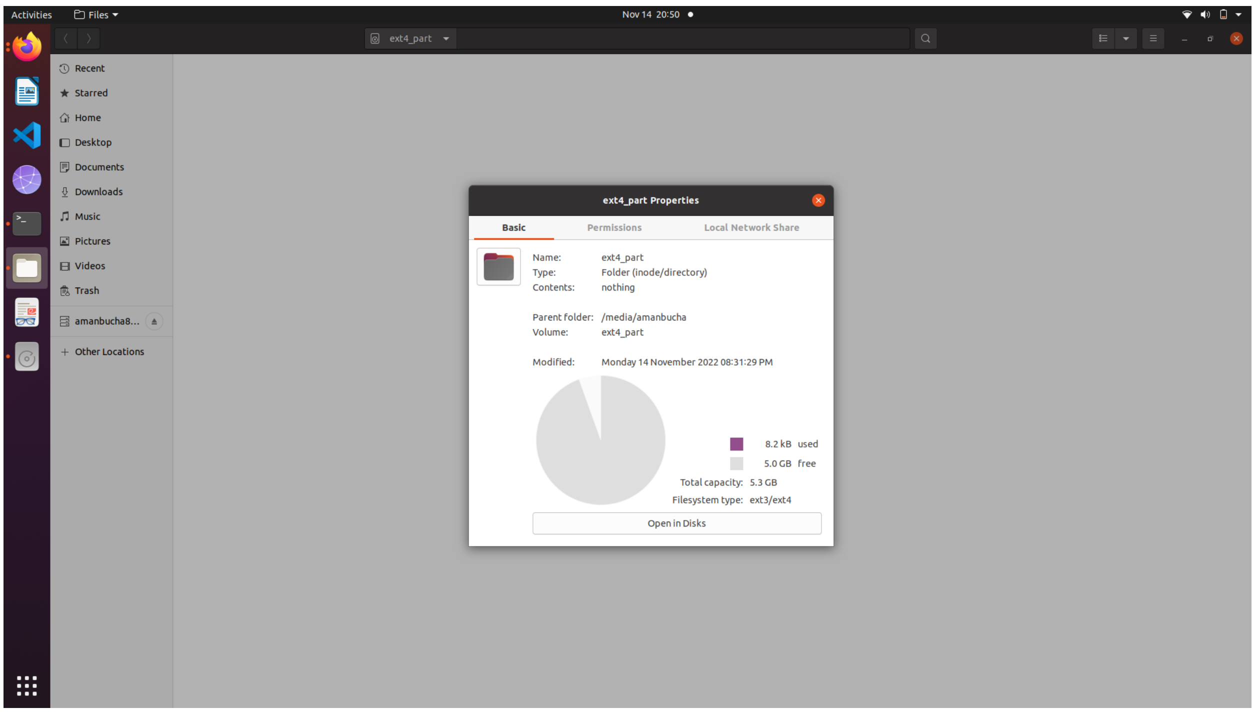
```
➜ vdbench zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool   4.50G   396K  4.50G        -         -     0%     0%  1.00x    ONLINE  -
```

After Workload

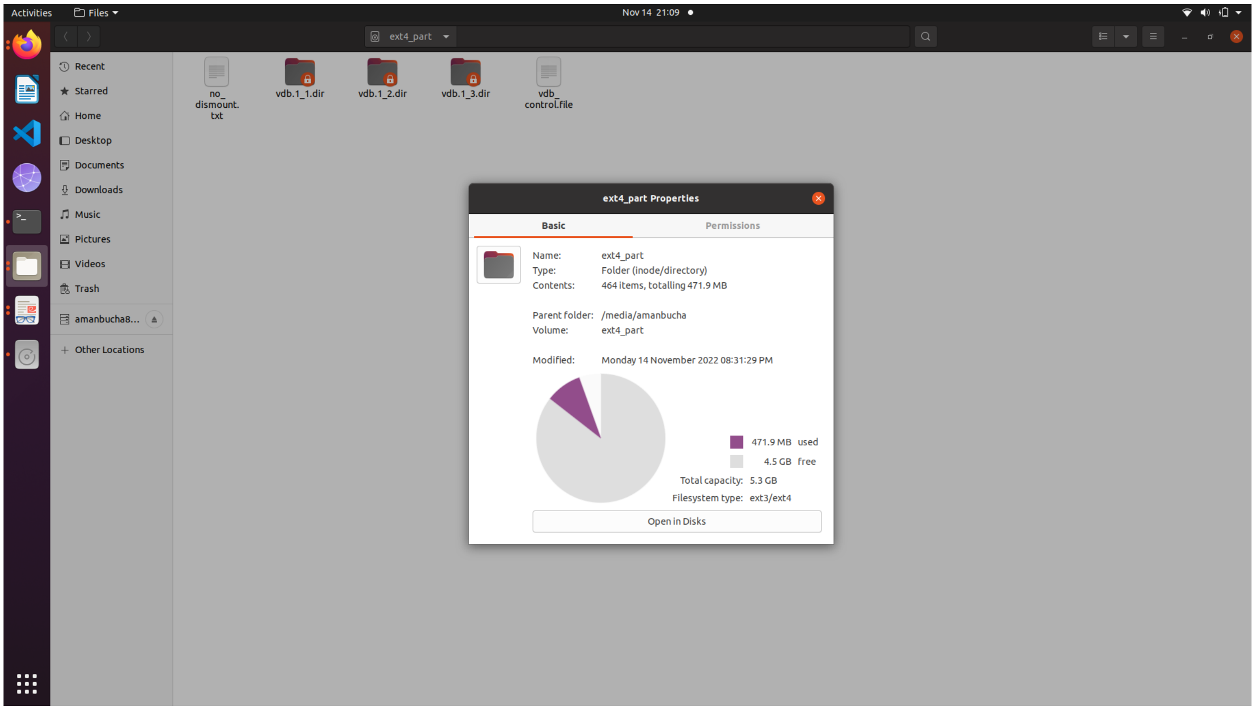
```
➜ vdbench zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool   4.50G   229M  4.28G        -         -     0%     4%  2.00x    ONLINE  -
```

1. ext4
    - The empty ext4_part folder initially contained **8.2 KB** of data.
    - After the workload was executed, the ext4_part folder had **471.9 MB** of data.
    - Therefore, the new files took **471.8 MB** of space. It is a little higher than 450 because of metadata.

Before Workload



After Workload



2. Large File Creation

a. We developed the following workload to test the production of huge files:

b. In this case, we are producing two 1GB files in a same folder. Since we are testing file creation, the `create` action is utilised.

c. Setting anchor to the ZFS Pool directory, we execute this workload on the ZFS file system using the following command.

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

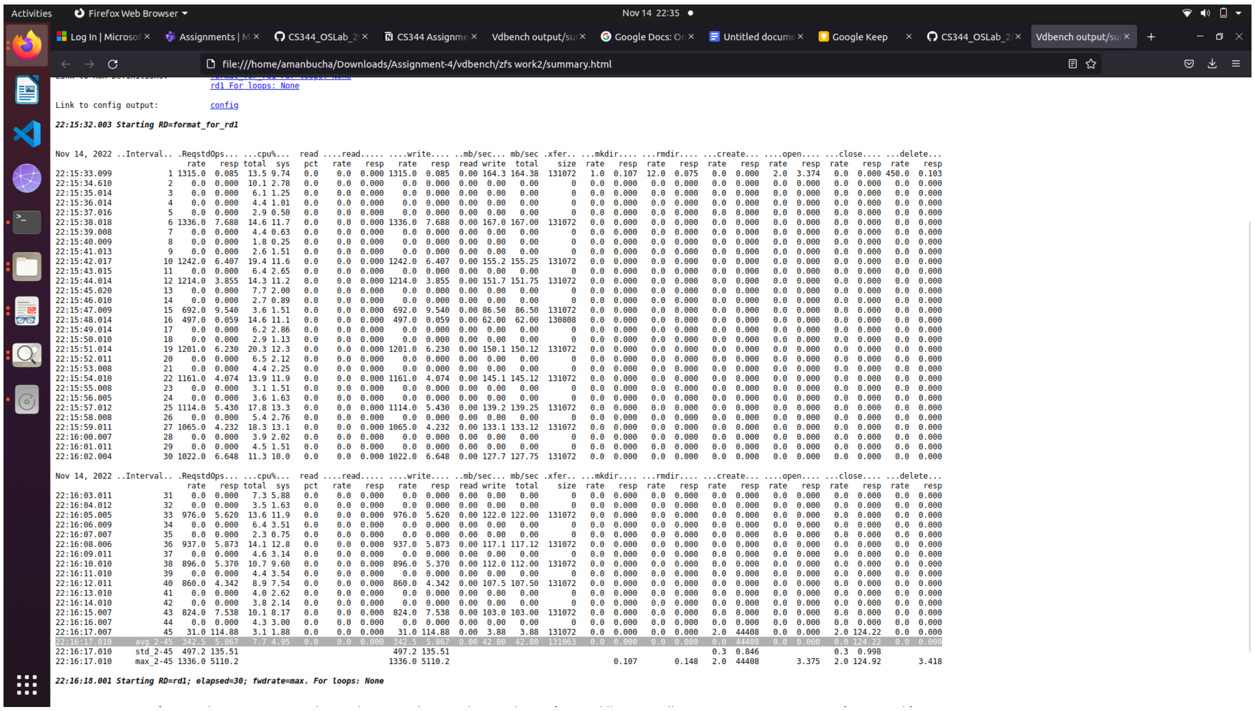
d. Setting anchor to the ext4 drive, we execute this workload on the ext4 file system using the following command.

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

Results:

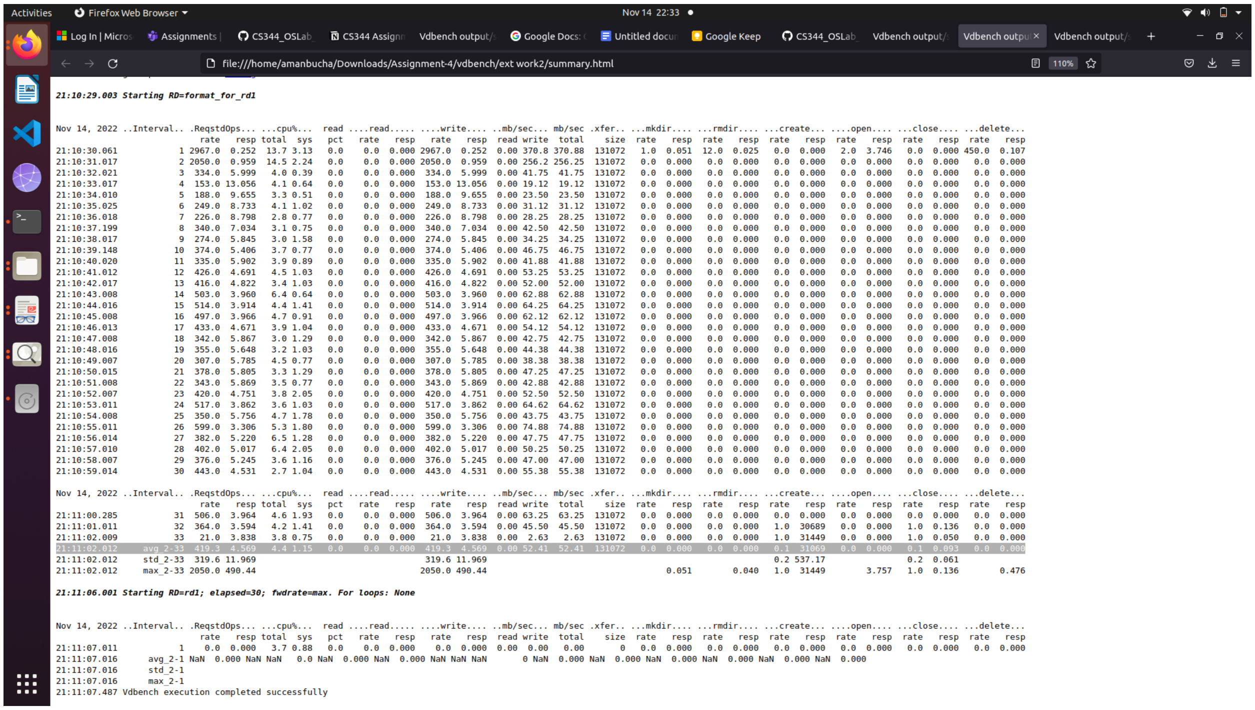
1. zfs

- Time taken to create files: **47.85 seconds**
- Average write rate: **42.80 MB/s**



1. ext4

- Time taken to create files: **39.07 seconds**
- Average write rate: **52.41 MB/**



Disadvantages of deduplication:

1. **<u>Performance:</u>**

   In the second workload (large file optimization), `ext4` was quicker. This is largely the result of `ext4`'s file optimization and `ZFS`'s deduplication costs.

   This demonstrates that deduplication has a negative impact on the performance of a file system owing to its overhead.

Disadvantages of large file creation optimization:

1. **<u>Metadata Overhead:</u>**

   In workload1 (in `ext4`), only **450 MB** was needed for the files, but the metadata overhead was much higher which made the total space utilized to be **471.8 MB.**

   However, `ZFS` has a very minimal overhead.

   Maintaining the extent trees (in `ext4`) requires a substantial amount of extra storage space compared to the real data (for small files).

2. **No data correction mechanisms:**

   `ext4` optimises the construction of huge files via the use of delayed and contiguous allocation and extents. This makes it difficult for any data correction procedures to exist, since huge files with numerous contiguous blocks retain relatively little information.