CS344 (OS LAB) ASSIGNMENT 4
GROUP G15

NAME	ANKET KOTKAR	ARYAN CHAUHAN	RITIK MANDLOI	RAHUL MALA
ROLL NO.	180101037	180101012	180101066	180101062

As mentioned in the assignment, the file systems we used for comparison of the features are ZFS and EXT4.

Features selected :

1)Data Deduplication (Comparison using ZFS with deduplication on and off and between ZFS and ext4)
2)Data Compression (Comparison using ZFS with compression on and off and between ZFS and ext4)
3)Fast Read-Write (Comparison between ext4 and ZFS)

1) Data Deduplication in ZFS

Implementation of deduplication is done using the below 2 steps:

1)Chunks of data (files, blocks or byte ranges) are checksummed using some hash function that uniquely identifies data with very high probability (using a secure hash function like SHA256 helps by decreasing the probability of collision to a large extent).

2)Then data is remembered in a table of some sort that maps the data's checksum to its storage location and reference count. **When you store another copy of existing data, instead of allocating new space on disk, the dedup code just increments the reference count on the existing data.**

Note the following important points:

1)Data can be deduplicated at the level of **files**, **blocks**, or **bytes**.
2)Deduplication can be either **synchronous** or **asynchronous** . In synchronous dedup, duplicates are eliminated as they appear. In asynchronous dedup duplicates are stored on disk and eliminated later.

2)Data Compression in ZFS

Compression in ZFS compresses your files on the fly and therefore lets you store more data using limited storage but ZFS compression comes with a cost - specifically, with extra CPU cycles needed to compress and uncompress ZFS data.

Following compression algorithms are available in ZFS file system:

I) **gzip** - standard UNIX compression.

II) **gzip-N** - selects a specific gzip level. gzip-1 provides the fastest gzip compression. gzip-9 provides the best data compression. gzip-6 is the default.

III) **lz4** - provides better compression with lower CPU overhead

IV) **lzjb** - optimized for performance while providing decent compression

V) **zle** - zero length encoding is useful for datasets with large blocks of zeros

3)Faster read-write in ext4 compared to ZFS

Ext4 and other new file systems use the following techniques to reduce read-write times:
1)**Optimised block allocation before writing them to the disk** which leads to higher speed in reading and writing (will be shown using the response time) compared to their older counterparts and other file systems.

2)**Delayed allocation** which allows it to coalesce writes and make better decisions about how to allocate blocks for the writes it has not yet committed

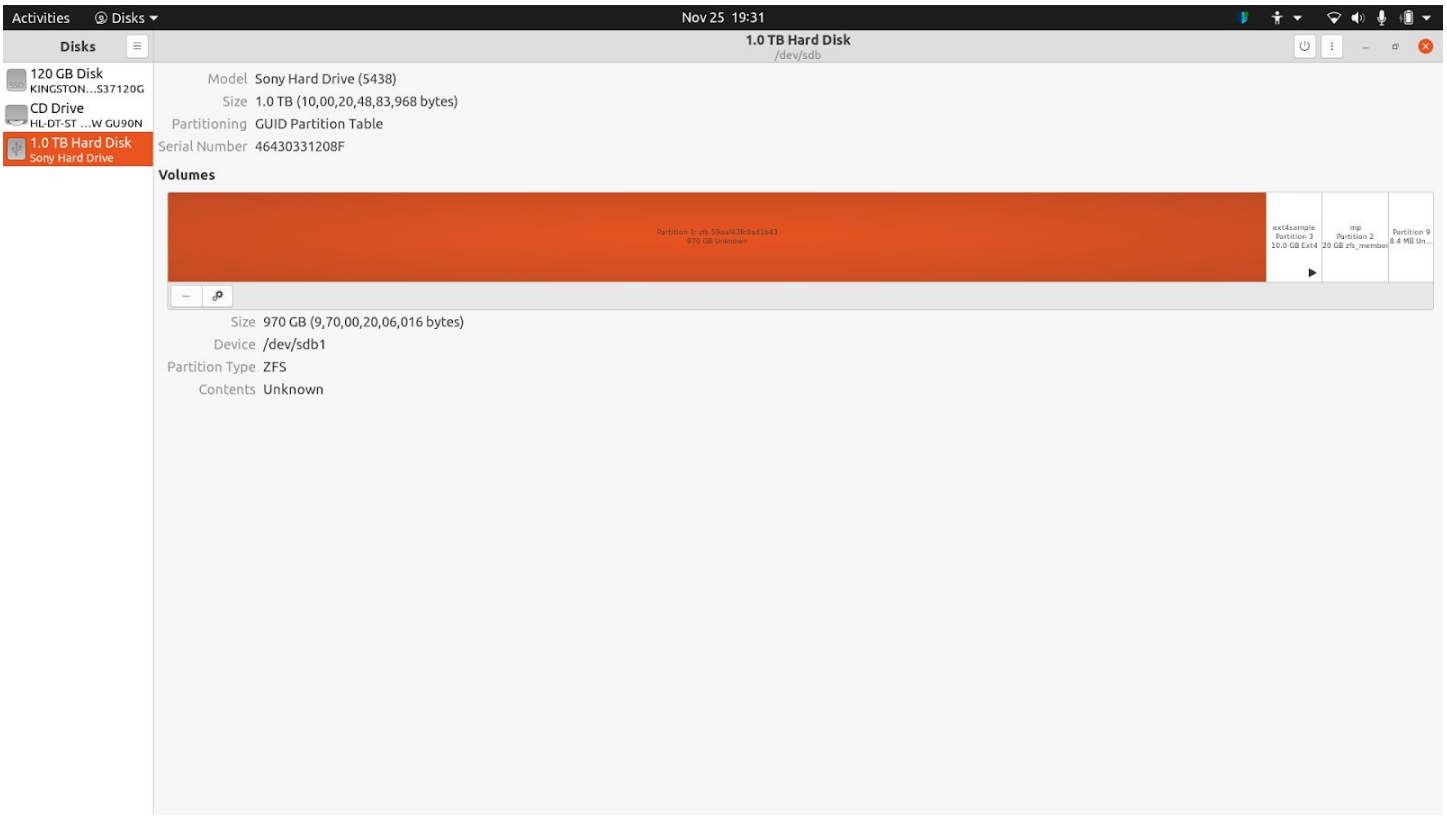
3)**fallocate()** which guarantees the availability of the space and attempts to find contiguous space for it) without first needing to write to it.

Along with these other features like **journal checksumming ,faster filesystem checks** and **online defragmentation (using e4defrag)** helped in increasing the performance.

HOW EXT4 and ZFS were installed and how VDBENCH was used for experiments:

To create a storage pool for zfs use the following command:
sudo zpool create -f /dev/<partition location> <name of the pool>

To create an ext4 partition, open the disks utility in ubuntu .



Vdbench was used to create the workloads (consists of 8 files of 50m each) to test the mentioned features.

For the exact commands used refer to README.md.
Command format: sudo ./vdbench -f <test case file containing parameters> -o <report output directory name>
zfs partition has name "mp" and ext4 has "ext4sample".

COMPARING THE FEATURES:

1)DATA DEDUPLICATION

a)Switching deduplication on and off in ZFS:

(DEDUPLICATION OFF IN ZFS)

```
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$ zpool list
NAME   SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
mp    18.5G   407M  18.1G        -         -     0%     2%  1.00x    ONLINE  -
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$
```

(DEDUPLICATION ON IN ZFS)

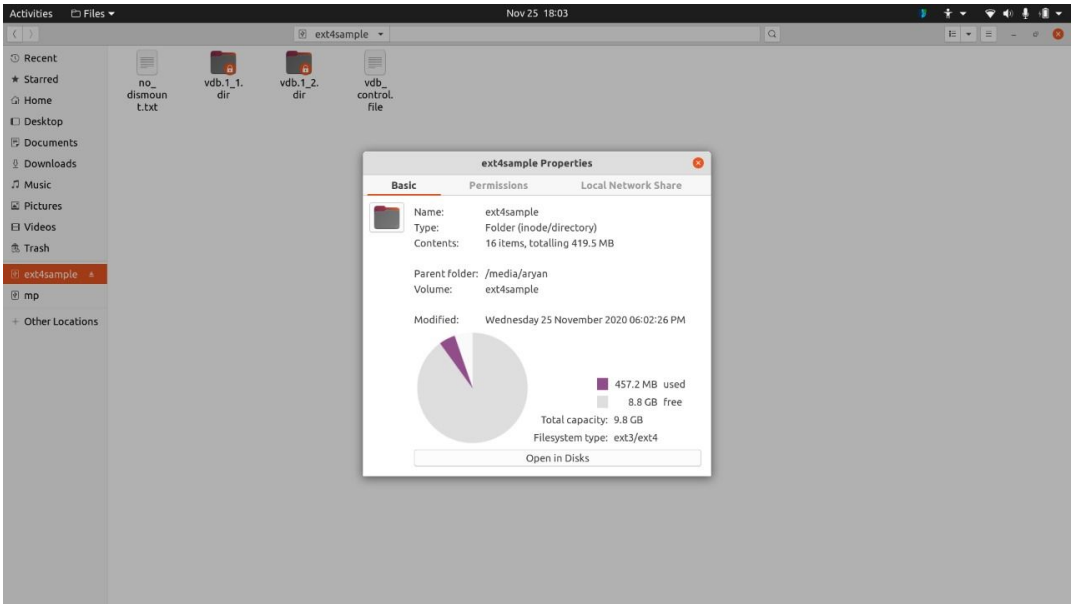
```
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$ zpool list
NAME   SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
mp    18.5G   206M  18.3G        -         -     0%     1%  1.99x    ONLINE  -
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$
```

b) COMPARISON BETWEEN ZFS (deduplication on) **AND EXT4** (no deduplication)

(DEDUPLICATION ON IN ZFS)

```
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$ zpool list
NAME   SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
mp    18.5G   206M  18.3G        -         -     0%     1%  1.99x    ONLINE  -
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$
```

(EXT4) no deduplication check the space used



OUR OBSERVATIONS: (advantages and disadvantages of using this feature)

ADVANTAGE:

From the above photos it can be observed that Allocated space decreased from 407 M to 206 M after switching on deduplication and the amount of disk space used decreased to 1% from 2%.This is great advantage as space decreased to almost 50% (after setting dedupratio=2).

DISADVANTAGE:

On top of this it can be observed that zfs has more CPU utilization (**32.4 compared to 40.4**)and slower read-write speeds as compared to ext4.(**The effect can be seen through the summary files submitted in zip although the effect seen is not significant as the number of files generated by us were low**).

Reason for slow read-write and CPU utilization:

When writing new blocks, it uses this table to determine whether a block has been written yet, or not.Over time, the table becomes larger and larger, and since every write operation has to use it, it should be kept in main memory to avoid unnecessary extra reads from disk .The bigger the deduplication table grows, the slower write performance will become, as more and more writes trigger more and more extra reads for dedup table lookups.

2) DATA COMPRESSION

a)Switching data compression on and off in zfs

(**DATA COMPRESSION ON**) value=10

```
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$ zpool list
NAME   SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
mp    18.5G  40.2M  18.5G        -         -     0%     0%  1.00x    ONLINE  -
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$
```

(**DATA COMPRESSION OFF**)

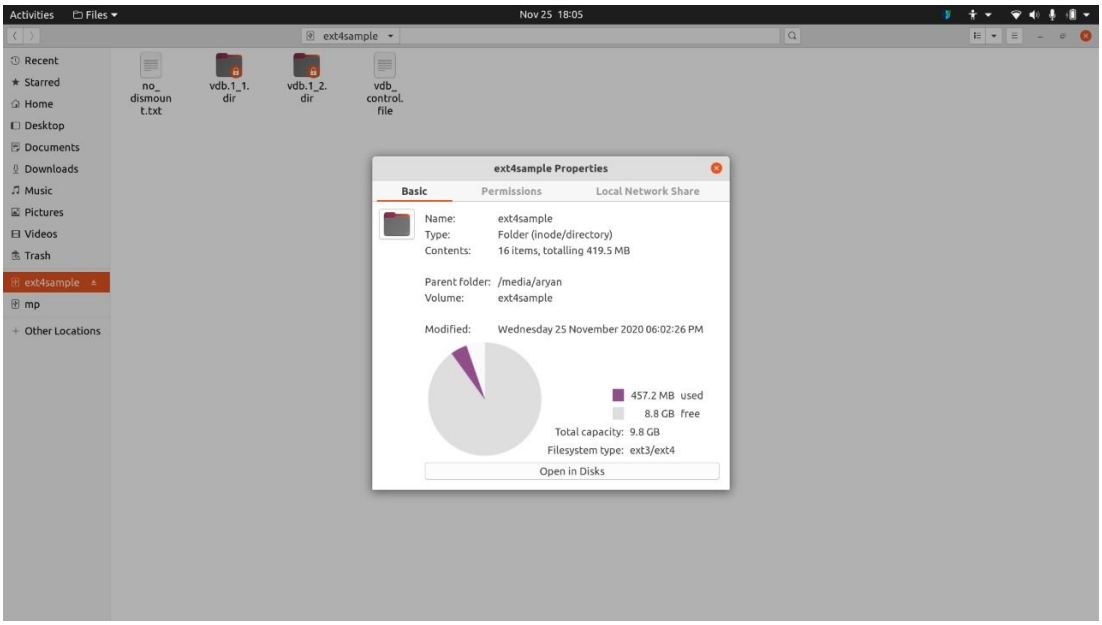
```
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$ zpool list
NAME   SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
mp    18.5G   407M  18.1G        -         -     0%     2%  1.00x    ONLINE  -
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$
```

b)Comparing ext4(no compression) and zfs(compression=10)

(**ZFS DATA COMPRESSION ON**) value=10

```
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$ zpool list
NAME   SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
mp    18.5G  40.2M  18.5G        -         -     0%     0%  1.00x    ONLINE  -
(base) aryan@aryan-Inspiron-5570:~/Pictures/Assignment-4/vdbench$
```

(**EXT4 no data compression**)



OUR OBSERVATIONS: (advantages and disadvantages of using this feature)

ADVANTAGES:

After setting the compression value to 10 it can be seen from above that the space used has decreased from 407M to 40.2 M (error rate of +5% to -5%).

Difference between deduplication and compression:

Compression stores the files on the disk in less space than original sizes of the files. It compresses the file size in a reversible way using different well established algorithms. Deduplication deals with the duplicate data present and data redundancy. Deduplication works in a way that random access into a file is possible, using an index such that location of any byte can be located randomly. Deduplication can be treated as a highly specialized and a slightly changed form of the compression.

Faster read-writes can also be seen in this case. (**check the response time in the summary file submitted in zip for comp-on-zfs and comp-off-zfs**).

DISADVANTAGES:

CPU utilisation increases in this case as more CPU cycles are required to implement the compression algorithm. Even to retrieve back the data time required will be more as the time will be required to decompress the previous data.

3)READ WRITE SPEED COMPARISON IN EXT4 and ZFS

EXT4

```
read ....read..... ....write.... .
 pct   rate   resp   rate   resp
 0.0    0.0  0.000 3200.0  0.257
 0.0    NaN  0.000    NaN  0.000
```

```
90; For loops: None
```

```
read ....read..... ....write.... .
 pct   rate   resp   rate   resp
49.3   35.0  0.021   36.0  0.040
52.7   48.0  0.018   43.0  0.032
51.8   59.0  0.017   55.0  0.034
50.0   51.0  0.016   51.0  0.035
57.3   51.0  0.019   38.0  0.037
42.9   48.0  0.021   64.0  0.039
43.1   47.0  0.020   62.0  0.042
41.1   37.0  0.019   53.0  4.368
44.3   43.0  0.043   54.0  1.082
48.0   48.0  0.020   52.0  0.046
47.8   48.0  0.021   52.4  0.644
        6.0  0.048    8.2  7.369
       59.0  0.986   64.0 110.96
```

ZFS

```
read ....read..... ....write.... .
 pct   rate   resp   rate   resp
 0.0    0.0  0.000 3200.0  0.639
 0.0    NaN  0.000    NaN  0.000
```

```
100; For loops: None
```

```
read ....read..... ....write.... .
 pct   rate   resp   rate   resp
49.3   35.0  0.079   36.0  0.125
55.2   64.0  0.063   52.0  0.079
49.5   54.0  0.069   55.0  0.086
48.8   40.0  0.070   42.0  0.097
55.3   57.0  0.061   46.0  0.102
42.7   41.0  0.094   55.0  0.126
44.2   42.0  0.106   53.0  0.116
41.5   49.0  0.094   69.0  0.121
44.2   46.0  0.086   58.0  0.131
47.7   42.0  0.079   46.0  0.124
47.7   48.3  0.079   52.9  0.110
        8.4  0.052    8.0  0.064
       64.0  0.541   69.0  0.632
lly
```

OUR OBSERVATIONS:

From the above photos it can be seen that response time for read and write operations in ext4 is less compared to the same in zfs**(check the resp column for read and write).**

This can be seen through factors like disk fragmentation and marking unallocated block groups (In ext4, unallocated block groups and sections of the inode table are marked as such. This enables e2fsck to skip them entirely on a check and greatly reduces the time it takes to check a file system of the size ext4 is built to support.**) along with other factors mentioned above at the start.**