CS344 - Operating Systems Lab

Assignment 4

Group C40:

Abhinav Kumar Chaudary	: 200101005
Yash Garg	: 200101113
Nayanika Ghosh	: 200123036
Aarti Meena	: 180101001

Part 1:

For this assignment, we are picking the filesystems : **ZFS and ext3.** We would compare the features of **Data Deduplication and Data compression** in these file systems.
Firstly, Let us understand about both files systems :

ZFS : ZFS or Zettabyte file system began as a part of Solaris OS in 2001. It is used as a file system as well as a volume manager to allocate space on mass-storage devices. For ZFS, security was given the highest priority. For that verification of all steps related to file management takes place, and the process is highly optimized, which is not achieved if we used separate volume and file managers.

Ext3 : Third extended filesystem, also known as **ext3**, is a journaled file system that is frequently used by the Linux kernel. Journaling, which increases reliability and eliminates the need to check the file system after an unclean shutdown, is its main advantage over ext2. ext4 is its successor.Ext3 has a significant advantage over competing Linux filesystems in that it allows in-place upgrades from ext2 without the need to backup and restore data, despite having less desirable performance (speed) than competing Linux filesystems like ext4, JFS, ReiserFS, and XFS. Without reformatting, it is simple to switch from ext2 to ext3 and gain the advantages of a reliable journaling file system.

The features we are implementing and differentiating between both File systems:
Data Deduplication is one of the features of this file system. Deduplication is the removal of duplicate copies of data. This can free up a significant amount of space on the hard drive and is particularly helpful in some settings where a lot of duplicate data is present, with or without minor changes. However, using this is only advised in exceptional circumstances due to the trade-off of high overhead computations. By hashing (using a secure hash like SHA256) a portion of the data to create an approximate unique signature and storing these in a hash table, deduplication is achieved. Data with a pre-existing signature is considered to be a copy of the data whose signature matches it when new data's signature is compared to values that already exist in the hash table. Depending on the volume of data that is hashed to a signature, deduplication can be applied at varying levels, with an increasing trade-off between space

savings from redundant data not being copied and overhead computations. These are at the file, block, and byte levels. Depending on whether the process occurs as the data is being written or whether the copies are hashed and deleted when the CPU is free, deduplication can also be synchronous or asynchronous.
ZFS has the deduplication feature, and it uses block-level synchronous deduplication. EXT3 does not support deduplication.

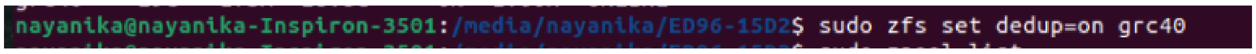
Data compression is a reduction in the amount of bits required to represent data. Data compression can reduce network bandwidth requirements, speed up file transfers, and save space on storage hardware. A programme that performs compression uses an algorithm or formula to decide how to reduce the size of the data. A single repeat character can be inserted to indicate a string of repeated characters, all unnecessary characters can be removed, and a smaller bit string can be used in place of a frequently occurring bit string to compress text.
A text file's original size can be reduced by 50% or even more through data compression.
ZFS has the compression feature whereas EXT3 does not support compression..

Part 2:

1. Deduplication

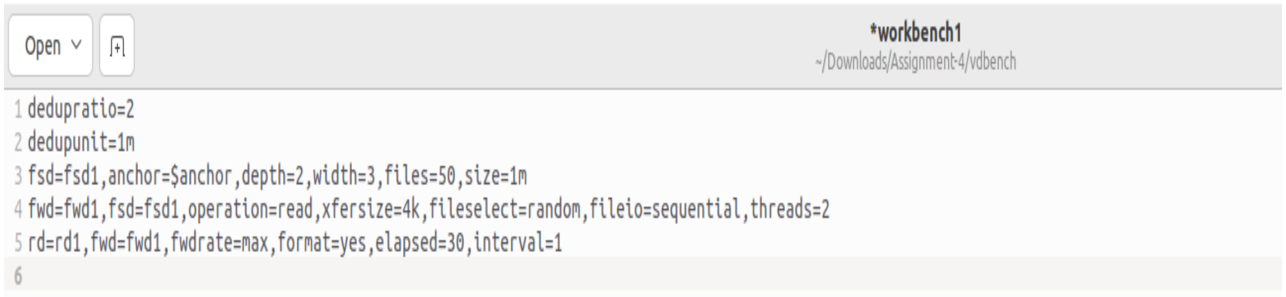
SETUP :-

a) We activated the data deduplication capability of **ZFS** using a pool we put up called **grc40**. We used the following command to turn it on :



```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zfs set dedup=on grc40
```

b) The following workload was made by us for data deduplication (workbench1).
With the help of this workload, we were able to compare how much space the new files in ZFS and ext4 took up.

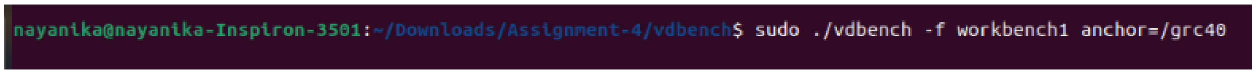


```
dedupratio=2
dedupunit=1m
fsd=fsd1,anchor=$anchor,depth=2,width=3,files=50,size=1m
fwd=fwd1,fsd=fsd1,operation=read,xfersize=4k,fileselect=random,fileio=sequential,threads=2
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=30,interval=1
```

c) In essence, we are producing 450 files (50*3*3), each measuring 1 MB, in a nested folder structure with depth 2 and width 3.
Then, for a further thirty seconds, these files are read sequentially to track statistics (although this step is not crucial because deduplication is completed at file creation).

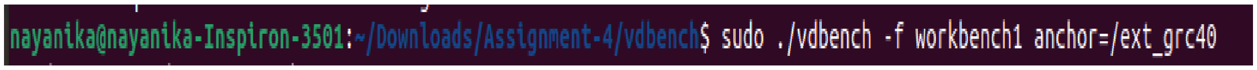
d) Dedupratio and dedupunit are both set to 2. The dedupratio measures the proportion of total blocks (of size dedupunit) to blocks with unique data. On the other hand, the dedupunit is the size of the block that will be compared to earlier blocks to look for duplication. We chose 1MB since it represents the size of a single file. In essence, this means that half of the files will be copies of the other half.

e) By setting anchor to the directory of the ZFS Pool (essentially, the folder pointing to the ZFS Pool), we perform this workload on the ZFS file system :



```
nayanika@nayanika-Inspiron-3501:~/Downloads/Assignment-4/vdbench$ sudo ./vdbench -f workbench1 anchor=/grc40
```

f) By setting anchor to the directory of the folder pointing to the ext4 drive, we perform this workload on the ext4 file system:

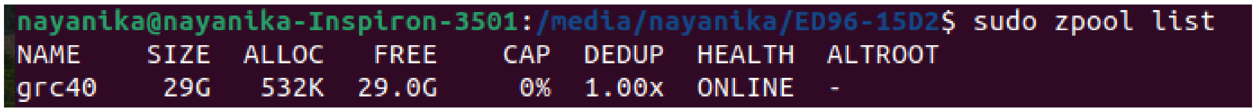


```
nayanika@nayanika-Inspiron-3501:~/Downloads/Assignment-4/vdbench$ sudo ./vdbench -f workbench1 anchor=/ext_grc40
```

RESULTS :-

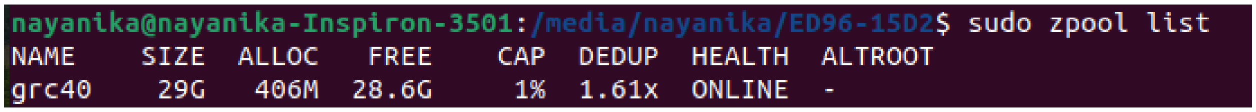
a) ZFS :-

Before Workload :-



```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zpool list
NAME     SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
grc40     29G   532K  29.0G     0%  1.00x  ONLINE  -
```

After Workload :-



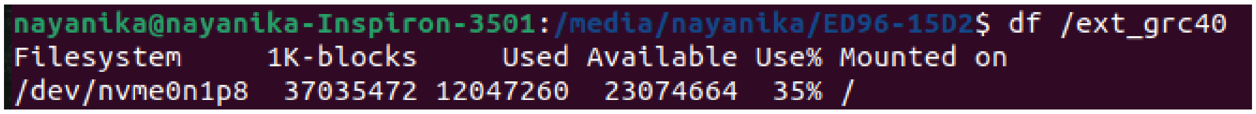
```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zpool list
NAME     SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
grc40     29G   406M  28.6G     1%  1.61x  ONLINE  -
```

i) The initial data in the empty ZFS folder was **532 KB**.
ii) The ZFS folder contained **406 MB** of data once the job was completed.
iii) A deduplication ratio of **1.61x** was seen, which was what we were looking for.
iv) This indicates that around **405.5 MB** were consumed by the new files.
The desired space, however, is **450MB (1MB*450)**.

Therefore, when duplicates are discovered, ZFS merely makes a pointer point to the old data utilising the data deduplication capability rather than maintaining entire blocks of data.

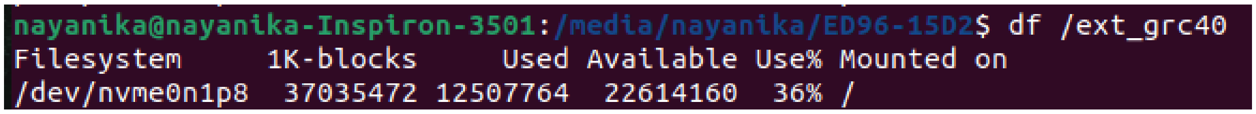
b) **ext3 :-**

Before Workload :-



```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ df /ext_grc40
Filesystem      1K-blocks     Used Available Use% Mounted on
/dev/nvme0n1p8   37035472 12047260  23074664  35% /
```

After Workload :-



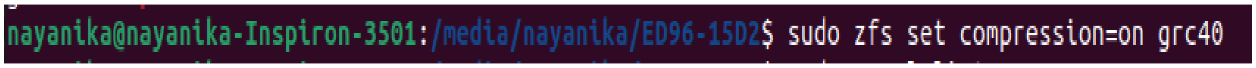
```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ df /ext_grc40
Filesystem      1K-blocks     Used Available Use% Mounted on
/dev/nvme0n1p8   37035472 12507764  22614160  36% /
```

i) The empty ext3 folder held **12.05 MB** of data at first.
ii) The ext3 folder contained **12.51 MB** of data when the workload had been completed.
iii) As a result, the new files used **0.46 MB** of space, which was somewhat more than expected due to metadata overhead.

2. Compression

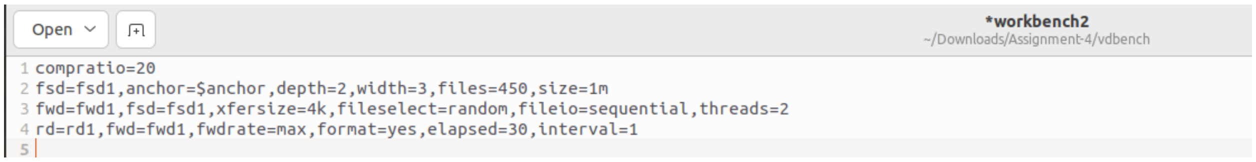
SETUP :-

a) We activated the data compression capability of **ZFS** using a pool we put up called **grc40**. We used the following command to turn it on:



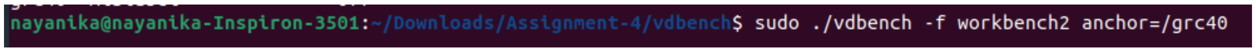
```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zfs set compression=on grc40
```

b) We created another workload (workbench2) for data compression. The workload was generated using vdbench. The parameters' meaning is unchanged from what was previously described in the workload setup for the deduplication features.



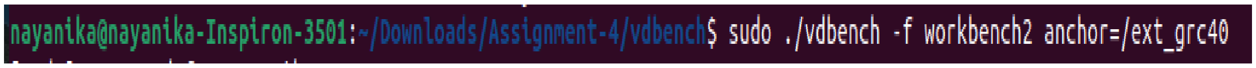
```
compratio=20
fsd=fsd1,anchor=$anchor,depth=2,width=3,files=450,size=1m
fwd=fwd1,fsd=fsd1,xfersize=4k,fileselect=random,fileio=sequential,threads=2
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=30,interval=1
```

c) We run this workload on the ZFS file system by setting anchor to the directory of the ZFS Pool i.e. grc20.



```
nayanika@nayanika-Inspiron-3501:~/Downloads/Assignment-4/vdbench$ sudo ./vdbench -f workbench2 anchor=/grc40
```

d) For running it on ext3 we used :

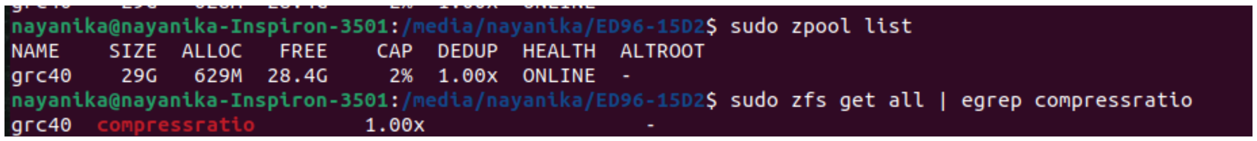


```
nayanika@nayanika-Inspiron-3501:~/Downloads/Assignment-4/vdbench$ sudo ./vdbench -f workbench2 anchor=/ext_grc40
```

RESULTS:-

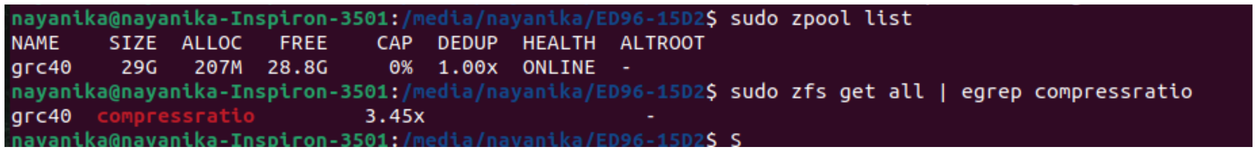
a) ZFS:

Before Workload (compression turned off) :-



```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zpool list
NAME     SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
grc40    29G   629M  28.4G     2%  1.00x  ONLINE  -
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zfs get all | egrep compressratio
grc40  compressratio         1.00x                  -
```

After Workload (compression turned on) :-

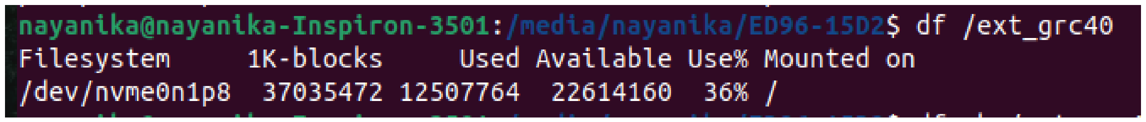


```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zpool list
NAME     SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
grc40    29G   207M  28.8G     0%  1.00x  ONLINE  -
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ sudo zfs get all | egrep compressratio
grc40  compressratio         3.45x                  -
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ $
```

We can clearly see the same workload is compressed to 1/3.45 size.

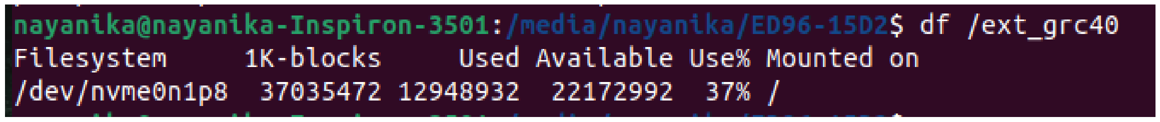
b) ext3:

Before Workload :-



```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ df /ext_grc40
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/nvme0n1p8  37035472 12507764  22614160  36% /
```

After Workload :-



```
nayanika@nayanika-Inspiron-3501:/media/nayanika/ED96-15D2$ df /ext_grc40
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/nvme0n1p8  37035472 12948932  22172992  37% /
```

After using the "df /ext_grc40/" command, we can see that the disc space needed to create (difference between used space before and after workload) the files was "450 MB," which is the same as the initial size. Therefore, we can conclude that ext3 does not support compression.

Part 3 :-

i) By publishing duplicate blocks only once and creating pointers to other copies of duplicate blocks, **ZFS** can greatly reduce the amount of disc space needed to store the data when we employ its **data deduplication** features and have a workload that writes comparable files to the disc. However, it should be noted that even while disc space is reduced, writing the same size of files to disc now takes much longer than it did when the **deduplication** option was **disabled**. This indicates that **deduplication** causes a **higher CPU utilisation**.

ii) By applying **compression** and storing data in compressed form, **ZFS** can dramatically **reduce** the amount of **disc space** needed for data storage if **compression** features are **enabled**.

iii) A significant disadvantage of **data compression** is the impact on the **performance** which is a consequence of the **utilisation of CPU** and memory resources to compress the data and perform decompression.

iv) When we execute the identical workload on the **ext3** file system, there is **no reduction** in the amount of disc space needed to hold the files. This is because **data deduplication** and **compression** functions are **not supported** by **ext3**.

In our observation, we found that **ZFS** is a relatively new file system with several cutting-edge capabilities like **data deduplication** and **compression**. **ZFS** is appropriate for usage in databases and huge data management systems thanks to these features.
On the other hand, **ext3** is a less sophisticated, older file system with a lower throughput.