Assignment 4: Quantitative Comparison of Filesystems

Pranshu Shah (200101085)
Vedant Shah (200101115)
Dhruv Shah (200101124)

ZFS

ZFS combines a file system with a volume manager. It began as part of the Sun Microsystems Solaris operating system in 2001 and was 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.

The ZFS file system and volume manager is characterized by **Data Integrity, High Scalability** and built in 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

In this assignment we have chosen Compression & Encryption as the 2 features to understand.

Feature 1 - Compression

The first feature that we have chosen is *compression*. It compresses files on the fly and therefore allows us to store more data with limited storage.

And we have chosen 2 instances of ***ZFS*** file system for evaluation:

- with **compression ON**
- with **compression OFF**

Understanding the feature (Compression)

ZFS uses the **LZ4** compression algorithm.

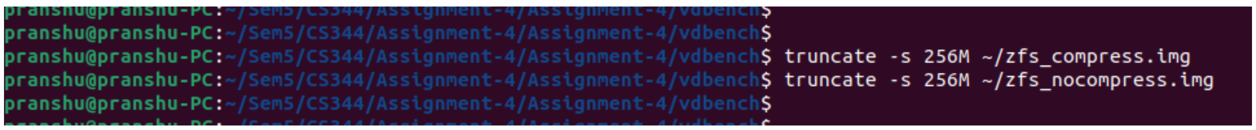
LZ4 is a lossless compression algorithm, providing compression speed greater than **500 MB/s per core** (>0.15 Bytes/cycle). It features an extremely fast decoder, with speed in multiple GB/s per core (~1 Byte/cycle).

- The LZ4 algorithm represents the data as a series of sequences. Each sequence begins with a one-byte token that is broken into **two 4-bit fields.**
- The first field represents the **number of literal bytes** that are to be copied to the output. The second field represents the **number of bytes to copy** from the already decoded output buffer (with 0 representing the minimum match length of 4 bytes).
- A value of 15 in either of the bit-fields indicates that the length is larger and there is an extra byte of data that is to be added to the length.
- A value of 255 in these extra bytes indicates that yet **another byte** to be added.
- Hence arbitrary lengths are represented by a series of extra bytes containing the value 255. The string of literals comes after the token and any extra bytes needed to indicate string length.
- This is followed by an **offset** that indicates **how far back in the output buffer** to begin copying. The extra bytes (if any) of the match-length come at the end of the sequence.

Workload for analyzing Compression

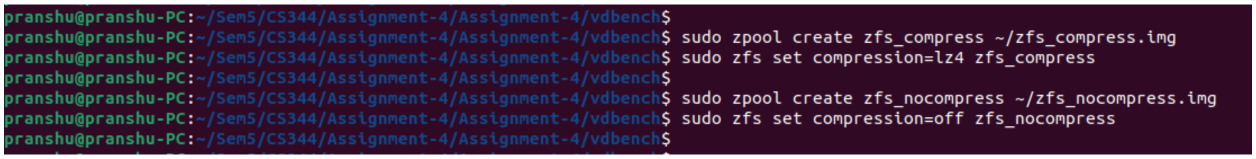
Note - Refer to file ***README*** for instructions describing how to run test code.

The given script was used to create **two virtual disk images** and install ZFS on each of them, one with compression (**zfs_compress.img**) and one with compression disabled (**zfs_nocompress.img**).



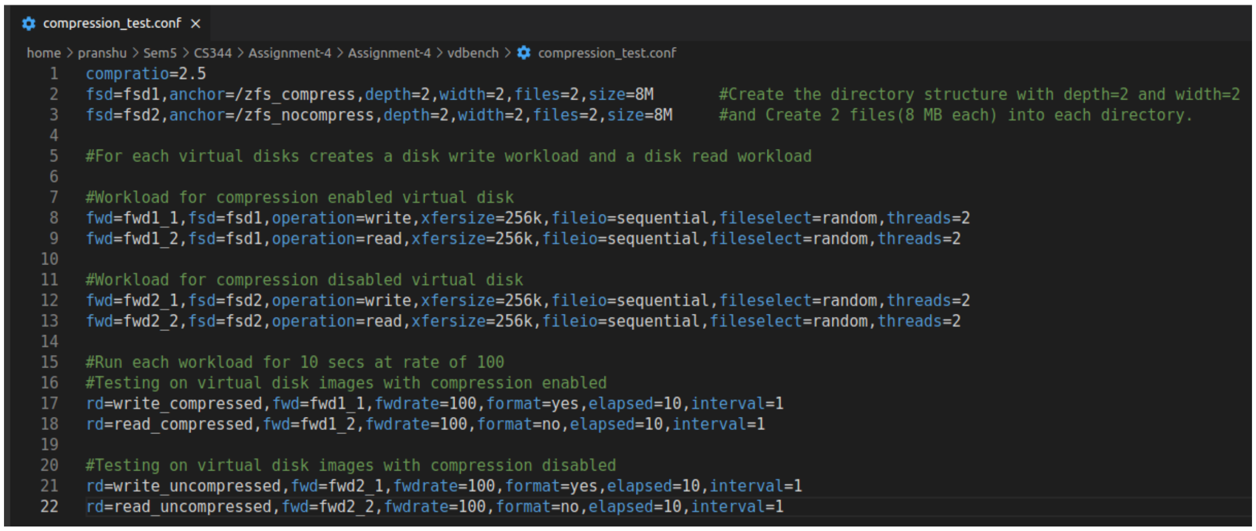
```
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ truncate -s 256M ~/zfs_compress.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ truncate -s 256M ~/zfs_nocompress.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
```

The allocated memory is then converted into a ZFS storage pool using zpool command.



```
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ sudo zpool create zfs_compress ~/zfs_compress.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ sudo zfs set compression=lz4 zfs_compress
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ sudo zpool create zfs_nocompress ~/zfs_nocompress.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ sudo zfs set compression=off zfs_nocompress
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
```

To quantify the benefits we have used the following workload -



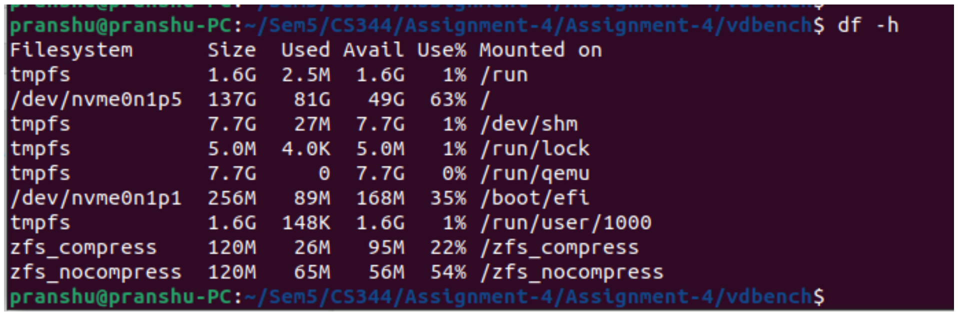
compression_test.conf

home > pranshu > Sem5 > CS344 > Assignment-4 > Assignment-4 > vdbench > compression_test.conf

```
compratio=2.5
fsd=fsd1,anchor=/zfs_compress,depth=2,width=2,files=2,size=8M       #Create the directory structure with depth=2 and width=2
fsd=fsd2,anchor=/zfs_nocompress,depth=2,width=2,files=2,size=8M     #and Create 2 files(8 MB each) into each directory.

#For each virtual disks creates a disk write workload and a disk read workload

#Workload for compression enabled virtual disk
fwd=fwd1_1,fsd=fsd1,operation=write,xfersize=256k,fileio=sequential,fileselect=random,threads=2
fwd=fwd1_2,fsd=fsd1,operation=read,xfersize=256k,fileio=sequential,fileselect=random,threads=2

#Workload for compression disabled virtual disk
fwd=fwd2_1,fsd=fsd2,operation=write,xfersize=256k,fileio=sequential,fileselect=random,threads=2
fwd=fwd2_2,fsd=fsd2,operation=read,xfersize=256k,fileio=sequential,fileselect=random,threads=2

#Run each workload for 10 secs at rate of 100
#Testing on virtual disk images with compression enabled
rd=write_compressed,fwd=fwd1_1,fwdrate=100,format=yes,elapsed=10,interval=1
rd=read_compressed,fwd=fwd1_2,fwdrate=100,format=no,elapsed=10,interval=1

#Testing on virtual disk images with compression disabled
rd=write_uncompressed,fwd=fwd2_1,fwdrate=100,format=yes,elapsed=10,interval=1
rd=read_uncompressed,fwd=fwd2_2,fwdrate=100,format=no,elapsed=10,interval=1
```

Outcome:

- **Advantages of Compression**
  1. **Lower Space Utilization** : When compression is enabled then space used for storing the same amount of data is less as compared to when it's not enabled. This is visible in the disk usage column shown below.
  2. **Less Storage costs** : As cost of storage is directly proportional to space occupied, compression leads to lower storage costs.
  3. **Less Data Transmission time** : The amount of space taken up is inversely related to the amount of time it takes to transfer a file. Therefore, compression makes file transfers faster.



```
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           1.6G  2.5M  1.6G   1% /run
/dev/nvme0n1p5  137G   81G   49G  63% /
tmpfs           7.7G   27M  7.7G   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           7.7G     0  7.7G   0% /run/qemu
/dev/nvme0n1p1  256M   89M  168M  35% /boot/efi
tmpfs           1.6G  148K  1.6G   1% /run/user/1000
zfs_compress    120M   26M   95M  22% /zfs_compress
zfs_nocompress  120M   65M   56M  54% /zfs_nocompress
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
```

- **Disadvantages of Compression**

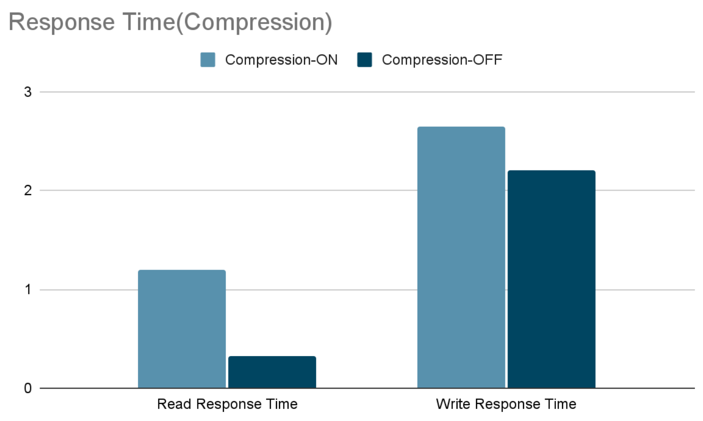
1. **More CPU Usage**: It was observed that the CPU utilization for reading and writing was higher when compression was enabled.This is because -
   - ❖ Data must be compressed using the LZ4 compression technique while being written. Thus writing without compression uses less processing power than this.
   - ❖ While reading, the data has to be uncompressed. This requires **more computation** power than reading without compression.

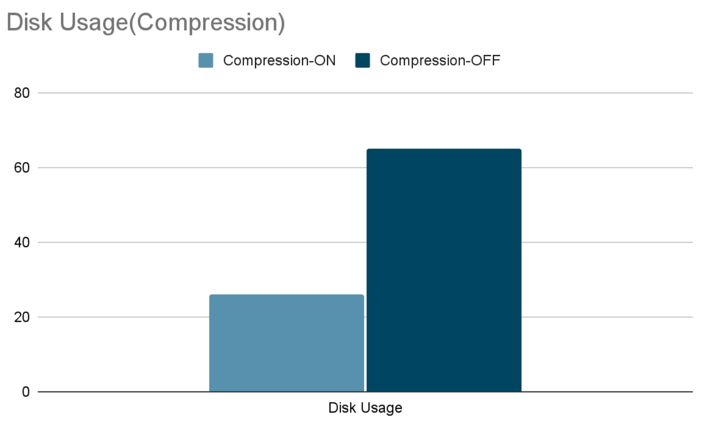
   The details can be seen in the table below.

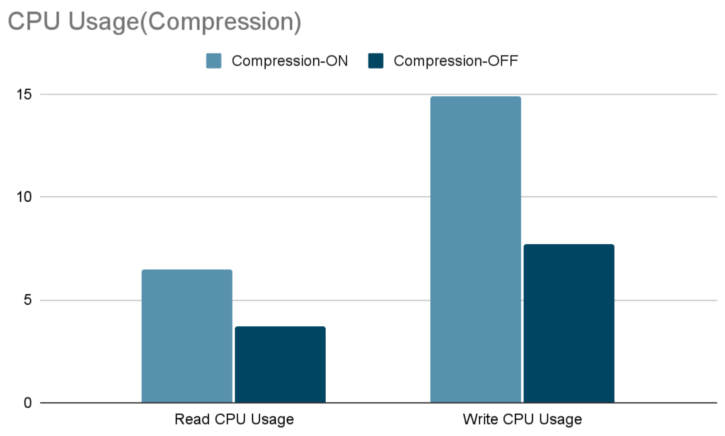
2. **More Time Taken**: Reads and writes take more time when compression is on. This is due to -
   - ❖ Data must be compressed using the LZ4 compression technique while being written. Some **additional time** gets used in running the algorithm.
   - ❖ While reading, the data has to be uncompressed. Some **additional time** gets utilized for running the algorithm.

   The details can be seen in the table below.

	Disk Usage	Read Response time	Read CPU Usage	Write Response time	Write CPU Usage
Compression - ON	26 MB	1.197 ms	6.5%	2.642 ms	14.9%
Compression - OFF	65 MB	0.327 ms	3.7%	2.200 ms	7.7%







Feature 2 - Encryption

The second feature that we have chosen is **encryption**. It enables files to be transparently encrypted to protect confidential data from attackers with physical access to the computer.

We have chosen 2 instances of ZFS file system for evaluation:

- With **Encryption ON**
- With **Encryption OFF**

Understanding the feature (Encryption)

ZFS uses the **AES-GCM** 256 bit authenticated encryption algorithm.

AES-GCM or **Advanced Encryption Standard with Galois Counter Mode** is a block cipher mode of operation that provides high speed authenticated encryption and data integrity. It provides high throughput rates for state-of-the-art, high-speed data transfer without any expensive hardware requirements.

The algorithm takes **4 inputs** -

1. **Secret key** - Secret key is the cipher key of length 256 bit.
2. **Initialization vector (IV)** - A randomly generated number that is used along with a secret key for data encryption
3. **Unencrypted text** - This is the plain-text that has to be encrypted
4. **Additional Authenticated Data (AAD)** - It is a string that can be used later on to decrypt the encrypted data. It is like a password which when later given, can decrypt the data

The algorithm gives **2 output** -

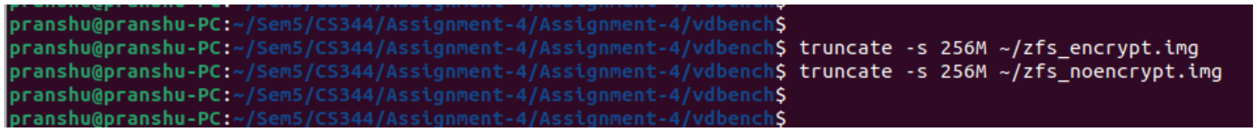
1. **Message Authentication Code (MAC or Tag)** - The code is a short piece of information used to authenticate a message. It can be used later on for authentication of the user
2. **Cipher Text** - This is the encrypted text that the algorithm outputs.

The data is considered as a series of blocks of size **128 bits**. Blocks are numbered sequentially, and then this block number is combined with an initialization vector and encrypted with the **Secret key**. The cipher-text blocks are considered coefficients of a polynomial which is evaluated at key-dependent points, using finite field arithmetic. The result is then XORed with the unencrypted text, to produce the final cypher text and the **Message Authentication Code**. A random or arbitrary Initialization vector is required for each encryption or else it would result in a less secure cipher-text.

Workload for analyzing Compression

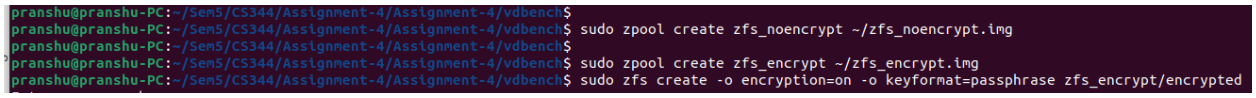
Note - Refer to file ***README*** for instructions describing how to run test code.

The given script was used to create two virtual disk images and install **ZFS** on each of them, one with compression (**zfs_encrypt.img**) and one with compression disabled (**zfs_noencrypt.img**).



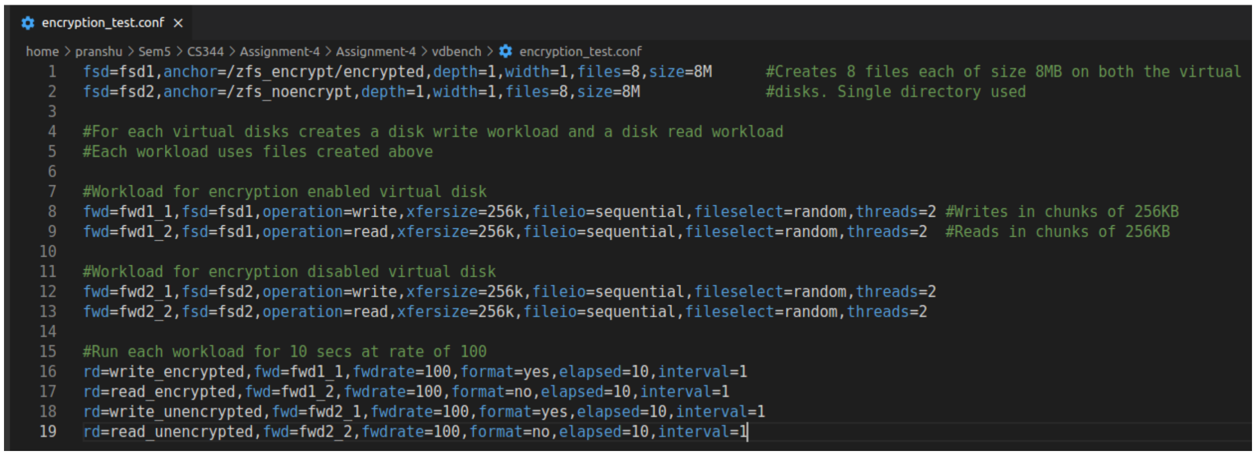
```
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ truncate -s 256M ~/zfs_encrypt.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ truncate -s 256M ~/zfs_noencrypt.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
```

The allocated memory is then converted into a ZFS storage pool using zpool command.



```
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ sudo zpool create zfs_noencrypt ~/zfs_noencrypt.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ sudo zpool create zfs_encrypt ~/zfs_encrypt.img
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ sudo zfs create -o encryption=on -o keyformat=passphrase zfs_encrypt/encrypted
```

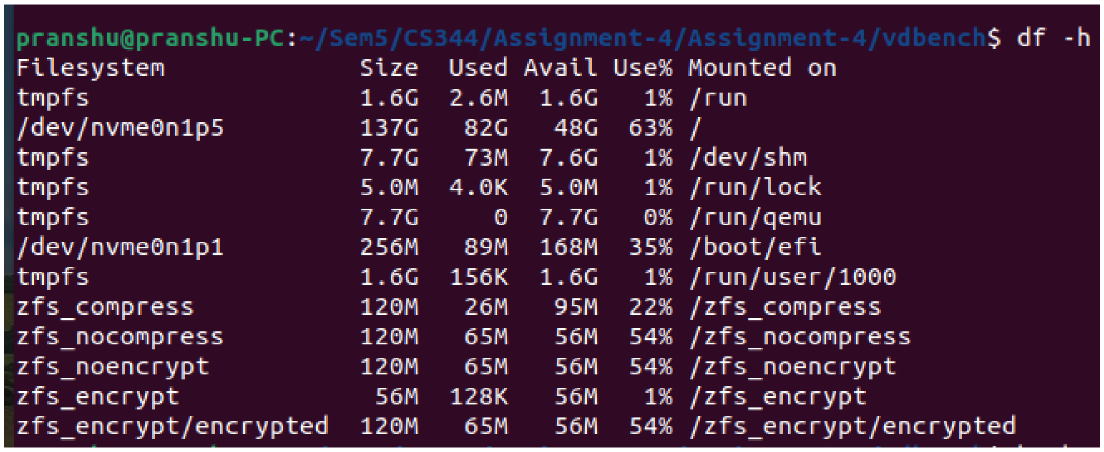
To quantify the benefits we have used the following workload -



encryption_test.conf

home > pranshu > Sem5 > CS344 > Assignment-4 > Assignment-4 > vdbench > encryption_test.conf

```
fsd=fsd1,anchor=/zfs_encrypt/encrypted,depth=1,width=1,files=8,size=8M      #Creates 8 files each of size 8MB on both the virtual
fsd=fsd2,anchor=/zfs_noencrypt,depth=1,width=1,files=8,size=8M              #disks. Single directory used

#For each virtual disks creates a disk write workload and a disk read workload
#Each workload uses files created above

#Workload for encryption enabled virtual disk
fwd=fwd1_1,fsd=fsd1,operation=write,xfersize=256k,fileio=sequential,fileselect=random,threads=2 #Writes in chunks of 256KB
fwd=fwd1_2,fsd=fsd1,operation=read,xfersize=256k,fileio=sequential,fileselect=random,threads=2  #Reads in chunks of 256KB

#Workload for encryption disabled virtual disk
fwd=fwd2_1,fsd=fsd2,operation=write,xfersize=256k,fileio=sequential,fileselect=random,threads=2
fwd=fwd2_2,fsd=fsd2,operation=read,xfersize=256k,fileio=sequential,fileselect=random,threads=2

#Run each workload for 10 secs at rate of 100
rd=write_encrypted,fwd=fwd1_1,fwdrate=100,format=yes,elapsed=10,interval=1
rd=read_encrypted,fwd=fwd1_2,fwdrate=100,format=no,elapsed=10,interval=1
rd=write_unencrypted,fwd=fwd2_1,fwdrate=100,format=yes,elapsed=10,interval=1
rd=read_unencrypted,fwd=fwd2_2,fwdrate=100,format=no,elapsed=10,interval=1
```



```
pranshu@pranshu-PC:~/Sem5/CS344/Assignment-4/Assignment-4/vdbench$ df -h
Filesystem              Size  Used Avail Use% Mounted on
tmpfs                   1.6G  2.6M  1.6G   1% /run
/dev/nvme0n1p5          137G   82G   48G  63% /
tmpfs                   7.7G   73M  7.6G   1% /dev/shm
tmpfs                   5.0M  4.0K  5.0M   1% /run/lock
tmpfs                   7.7G     0  7.7G   0% /run/qemu
/dev/nvme0n1p1          256M   89M  168M  35% /boot/efi
tmpfs                   1.6G  156K  1.6G   1% /run/user/1000
zfs_compress            120M   26M   95M  22% /zfs_compress
zfs_nocompress          120M   65M   56M  54% /zfs_nocompress
zfs_noencrypt           120M   65M   56M  54% /zfs_noencrypt
zfs_encrypt              56M  128K   56M   1% /zfs_encrypt
zfs_encrypt/encrypted   120M   65M   56M  54% /zfs_encrypt/encrypted
```

Outcome:

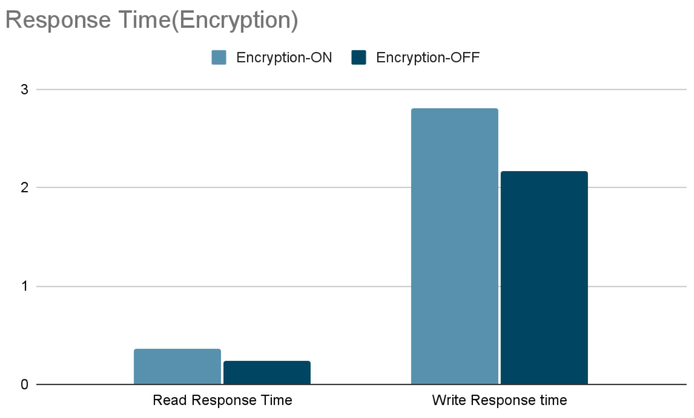
- **Advantages of Encryption**
    1. **Prevention of Unauthorized Access**: It gives the option of limiting access to particular files on shared computers or networks by unauthorized users. Users are still able to access other files on the disc as a result of this without being restricted.
    2. **Enhanced Data Integrity**: Encryption keeps your data safe from alterations, and recipients of the data will be able to see if it has been tampered with.

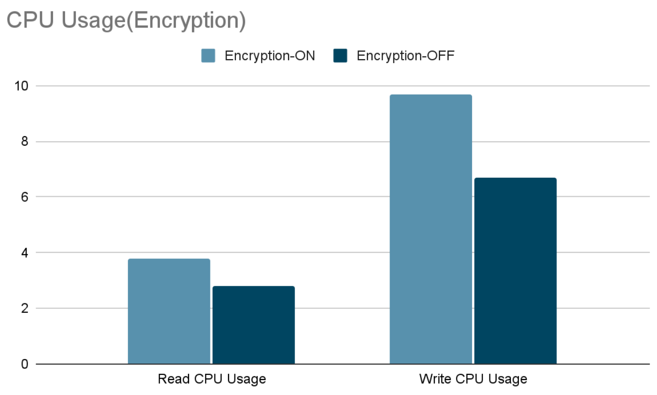
- **Disadvantages of Encryption**
    1. **Risk Of Losing Data**: Data cannot be recovered by any means if the user forgets the password.
    2. **More Time** : With encryption ON, running time observed was much higher while reading and writing. This is because-
        - ❖ Everytime the data is written, it has to be encrypted by the **AES-GCM** encryption algorithm. Some **additional time** gets utilized for running the computationally heavy Encryption algorithm.
        - ❖ Everytime the data is read, it has to be decrypted by the algorithm. Some **additional time** gets utilized for decrypting the cipher-text.

       The effects can be observed in the table shown below.
    3. **More CPU Usage**: CPU usage was observed to be higher with encryption ON. This is because-
        - ❖ Everytime the data is written, it has to be encrypted by the **AES-GCM** encryption algorithm. This requires much **more computation** power than writing without encryption.
        - ❖ Everytime the data is read, it has to be decrypted by the algorithm. This requires much **more computation** power than reading without encryption.

       The effects can be observed in the table shown below.

	Read Response time	Read CPU Usage	Write Response Time	Write CPU Usage
Encryption - ON	0.358 ms	3.8%	2.802 ms	9.7%
Encryption - OFF	0.239 ms	2.8%	2.168 ms	6.7%


Response Time(Encryption)
Encryption-ON
Encryption-OFF
Read Response Time
Write Response time


CPU Usage(Encryption)
Encryption-ON
Encryption-OFF
Read CPU Usage
Write CPU Usage