CS 344: OPERATING SYSTEMS LAB
LAB 4

GROUP NUMBER: 12

<u>GROUP MEMBERS</u>:

MOHAMMAD HUMAM KHAN (180123057)

KARTIKEYA SINGH (180123021)

SIDDHARTHA JAIN (180101078)

AB SATYAPRAKASH (180123062)

FILE SYSTEM COMPARISON

ZFS (Zettabyte File System)

ZFS is a local file system and logical volume manager created by Sun Microsystems Inc. to direct and control the placement, storage and retrieval of data in enterprise-class computing systems.

The ZFS file system and volume manager is characterized by **data integrity**, **high scalability** and built-in storage features such as:

- **Replication** - the process of making a replica (a copy) of something.
- **Deduplication** - a process that eliminates redundant copies of data and reduces storage overhead.
- **Compression** - a reduction in the number of bits needed to represent data.
- **Snapshots** - a set of reference markers for data at a particular point in time.
- **Clones** - an identical copy of something.
- **Data protection** - the process of safeguarding important information from corruption and/or loss.

In this lab, we will demonstrate **ZFS with and without data deduplication feature and compression feature and analyze it for various benefits it offers with respect to other file systems**.

EXT3 File System

ext3, or **third extended filesystem**, is a journaled file system that is commonly used by the Linux kernel. Its main advantage over ext2 is journaling, which improves reliability and eliminates the need to check the file system after an unclean shutdown. Its successor is ext4. The performance (speed) of ext3 is less attractive than competing Linux filesystems, such as ext4, JFS, ReiserFS, and XFS, but ext3 has a significant advantage in that it allows in-place upgrades from ext2 without having to back up and restore data. It is easy to migrate from ext2 to ext3 and gain the benefits of a robust journaling file system without reformatting. ext3 adds the following features to ext2:

- **A journal**
- **Online file system growth**
- **HTree indexing for large directories**

Because ext3 aims to be backwards-compatible with the earlier ext2, many of the on-disk structures are similar to those of ext2. Consequently, ext3 lacks recent features, such as extents, dynamic allocation of inodes, and block sub-allocation. There is no online ext3 defragmentation tool that works on the filesystem level. ext3 does not support the recovery of deleted files.

In this lab, we will demonstrate **ext3 using data deduplication feature and compression feature and analyze it for these features w.r.t. ZFS file system**.

FEATURES OF FILE SYSTEM USED FOR STUDY

DATA DEDUPLICATION

Data deduplication is a technique for eliminating duplicate copies of repeating data. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the deduplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times (the match frequency is dependent on the chunk size), the amount of data that must be stored or transferred can be greatly reduced. Storage-based data deduplication reduces the amount of storage needed for a given set of files. It is most effective in applications where many copies of very similar or even identical data are stored on a single disk.

COMPRESSION

Data compression is a reduction in the number of bits needed to represent data. Compressing data can save storage capacity, speed up file transfer, and decrease costs for storage hardware and network bandwidth. Compression is performed by a program that uses a formula or algorithm to determine how to shrink the size of the data. Text compression can be as simple as removing all unneeded characters, inserting a single repeat character to indicate a string of repeated characters and substituting a smaller bit string for a frequently occurring bit string. Data compression can reduce a text file to 50% or a significantly higher percentage of its original size.

The main advantages of compression are a reduction in storage hardware, data transmission time and communication bandwidth and the resulting cost savings.

The main disadvantage of data compression is the performance impact resulting from the use of CPU and memory resources to compress the data and perform decompression.

COMPRESSION v/s. DATA DEDUPLICATION

Compression is often compared to data deduplication, but the two techniques operate differently. Deduplication is a type of compression that looks for redundant chunks of data across a storage or file system and then replaces each duplicate chunk with a pointer to the original. Data compression algorithms reduce the size of the bit strings in a data stream that is far smaller in scope and generally remembers no more than the last megabyte or less of data. Deduplication is most effective in environments that have a high degree of redundant data, such as virtual desktop infrastructure or storage backup systems. Data compression tends to be more effective than deduplication in reducing the size of unique information, such as images, audio, videos, databases and executable files.

EXPERIMENTAL SETUP FOR ZFS

Data Deduplication is built into Vdbench with the understanding that the deduplication logic included in the target storage device looks at each n-byte data block to see if a block with identical content already exists. When there is a match the block no longer needs to be written to storage and a pointer to the already existing block is stored instead. Since it is possible for dedup and data compression algorithms to be used at the same time, dedup by default generates data patterns that do not compress.

We installed **ZFS** on a **USB drive** to perform our study. The following steps were used for the same-

1) ZFS utilities were installed using the command -

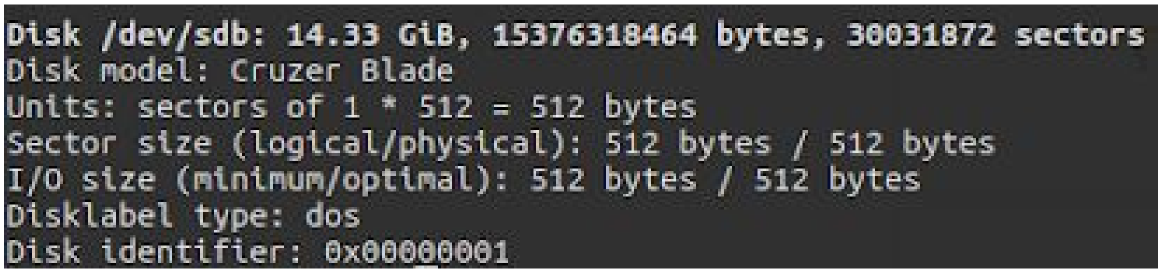
```
sudo apt install zfs-fuse
```

2) ZFS daemon is started using the command -

```
/etc/init.d/zfs-fuse start
```

3) The USB drive path was obtained using -

```
sudo fdisk -l
```



```
Disk /dev/sdb: 14.33 GiB, 15376318464 bytes, 30031872 sectors
Disk model: Cruzer Blade
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000001
```

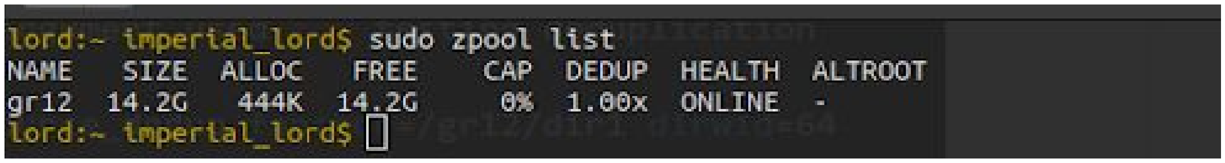
4) A ZFS pool was created on the USB drive using -

```
sudo zpool create gr12 /dev/sdb
```

***NOTE**: After running this command, the USB drive was mounted in the root directory as the folder **gr12**. All subsequent workload operations are done on this folder from vdbench.*

5) The created zfs pool can be seen using command -

```
sudo zpool list
```



```
lord:~ imperial_lord$ sudo zpool list
NAME    SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
gr12  14.2G   444K  14.2G     0%  1.00x  ONLINE  -
lord:~ imperial_lord$ 
```

6) Deduplication feature can be enabled/disabled using the following commands -

```
sudo zfs set dedup=on gr12    //for enabling dedup
sudo zfs set dedup=off gr12   //for disabling dedup
```

7) Compression feature can be enabled/disabled using the following commands -

```
sudo zfs set compression=on gr12   //for enabling compression
sudo zfs set compression=off gr12 //for disabling compression
```

EXPERIMENTAL SETUP FOR ext3

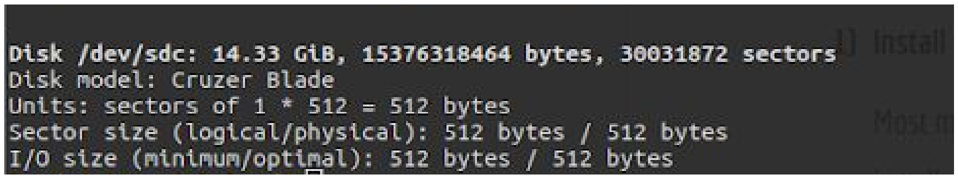
Ext3 is an old file system and does not support most features that it's modern counterparts (such as ZFS, NTFS, FAT etc.) do.

We installed **ext3** on a **USB drive** to perform our study. The following steps were used for the same-

1) Install mk2fs (to create ext2/ext3/ext4 file system inside a partition) -

   Most modern Ubuntu versions come preinstalled with mk2fs, however, it can be installed by the following command as well:

```
sudo apt-get install e2fsprogs
```



```
Disk /dev/sdc: 14.33 GiB, 15376318464 bytes, 30031872 sectors
Disk model: Cruzer Blade
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

2) The USB drive path was obtained using -

```
sudo fdisk -l
```

3) Create a mount point -

```
mkdir /ext_gr12
```

4) Edit the /etc/fstab (after giving appropriate permissions) -

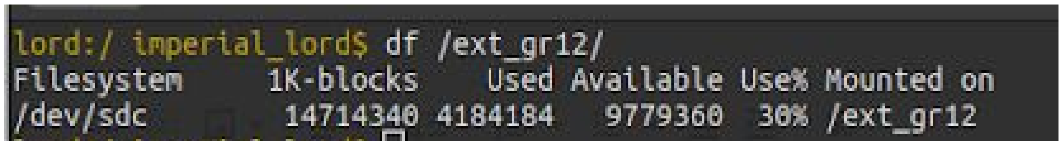
```
gedit /etc/fstab
```

Add the following line to the file:

```
/dev/sdbc /ext_gr12 ext3 defaults 1 2
```

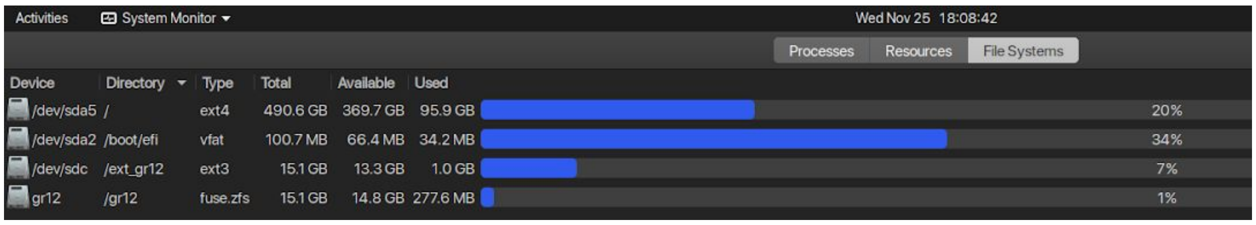
5) The ext3 based directory details can be seen using the following command -

```
df /ext_gr12/
```



```
lord:/ imperial_lord$ df /ext_gr12/
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sdc        14714340 4184184   9779360  30% /ext_gr12
```

After completing the above steps, we can see the created file systems by going into the **system monitor** in ubuntu. A screenshot of the same is attached below to verify that we have installed **ZFS** and **EXT3** file systems in the USB drive which are mounted in respective directories in the root of the OS.



```
Device      Directory   Type      Total     Available  Used
/dev/sda5   /           ext4      490.6 GB  369.7 GB   95.9 GB   20%
/dev/sda2   /boot/efi   vfat      100.7 MB  66.4 MB    34.2 MB   34%
/dev/sdc    /ext_gr12   ext3      15.1 GB   13.3 GB    1.0 GB    7%
gr12        /gr12       fuse.zfs  15.1 GB   14.8 GB    277.6 MB  1%
```

Workload Creation and Execution for comparing Deduplication

The workload was created using vdbench using **dedupratio = 20** and a **dedupunit = 4k**

- ★ fanchor = file system place where directories and files will be created.
- ★ dirwid = width of the directories i.e. number of directories created
- ★ numfiles = number of files per directory.
- ★ filesize = size of a file in k, m, g (e.g. 16k = 16KBytes).
- ★ fxfersize = file I/O transfer size in kbytes.
- ★ thrds = number of threads or workers.
- ★ etime = how long to run in minutes (m) or hours (h).
- ★ itime = interval sample time (e.g. 30 seconds).
- ★ dirdep = depth of the directory tree.
- ★ filrdpct = percentage of reads (e.g. 90 = 90 percent reads).
- ★ -p process number = optional specify a process number, only needed if running multiple vdbenchs at the same time, the number should be unique.
- ★ -o output file = describes what is being done and some config info.

```
dedupratio=20
dedupunit=4k
fsd=fsd1,anchor=!fanchor,depth=!dirdep,width=!dirwid,files=!numfiles,
size=!filesize
fwd=fwd1,fsd=fsd1,rdpct=!filrdpct,xfersize=!fxfersize,fileselect=rand
```

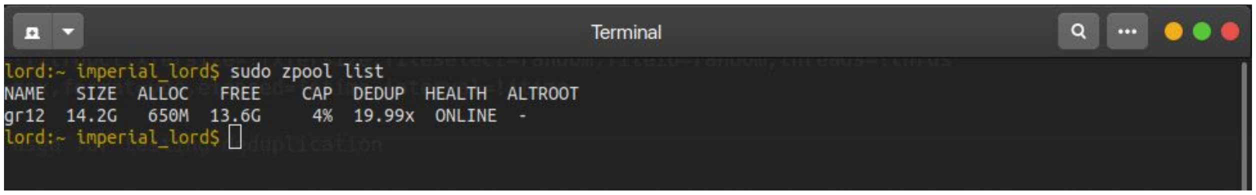
```
om,fileio=random,threads=!thrds
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=!etime,interval=!itime
```

Command to run the workload

```
./vdbench -f source_dedup fanchor=/gr12/dir1 dirwid=64
numfiles=16384 filesize=4k fxfersize=4k thrds=64 etime=1m
itime=30 dirdep=1 filrdpct=0 -p 5576 -o
group12_output_dedup
```

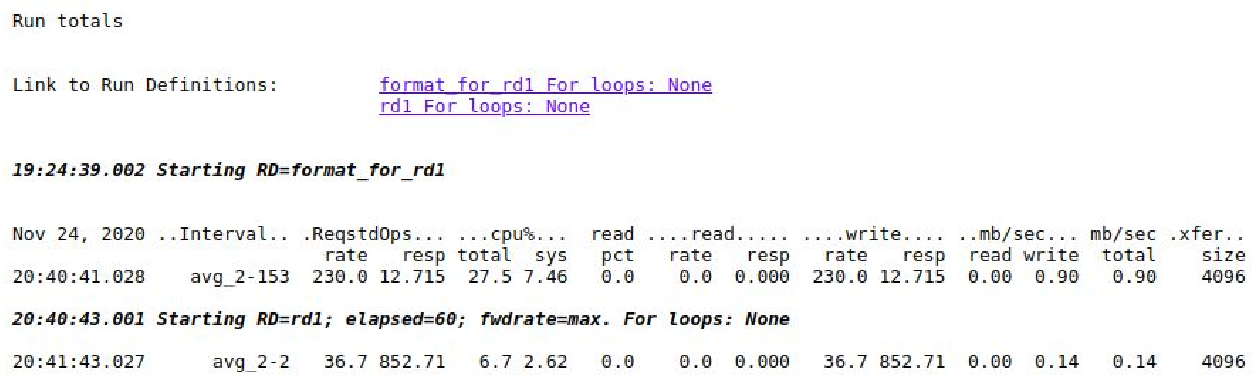
Observations

★ First, the deduplication feature was **enabled** and the vdbench workload was run to create similar files with **dedup ratio** = **20**. A total of 1M files (64 directories each with 16384 files) each of size 4 kB was created which accounted for a total size equal to 4GB. In this case, It was observed that the total space in the disk used was equal to **"650MB"** which is very less than "**4GB**". Also after running **"sudo zpool list"** command it can be observed that the dedup value is equal to **19.99x** i.e. approximately 20 which was "dedupratio" value in workload used for the study.



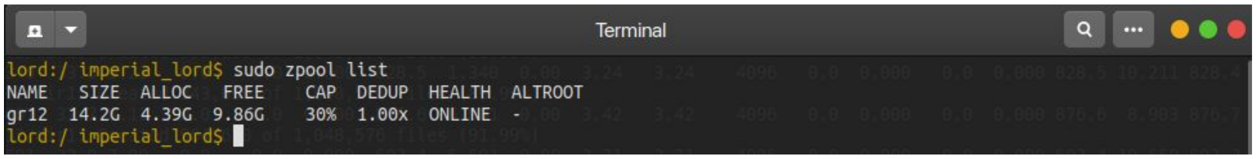
```
lord:~ imperial_lord$ sudo zpool list
NAME   SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
gr12  14.2G   650M  13.6G     4%  19.99x  ONLINE  -
lord:~ imperial_lord$ 
```

Also from the statistics, it can be observed that the simulation runs for about **"1hr 10min"** to create all the files in the workload (Observe start and end time).



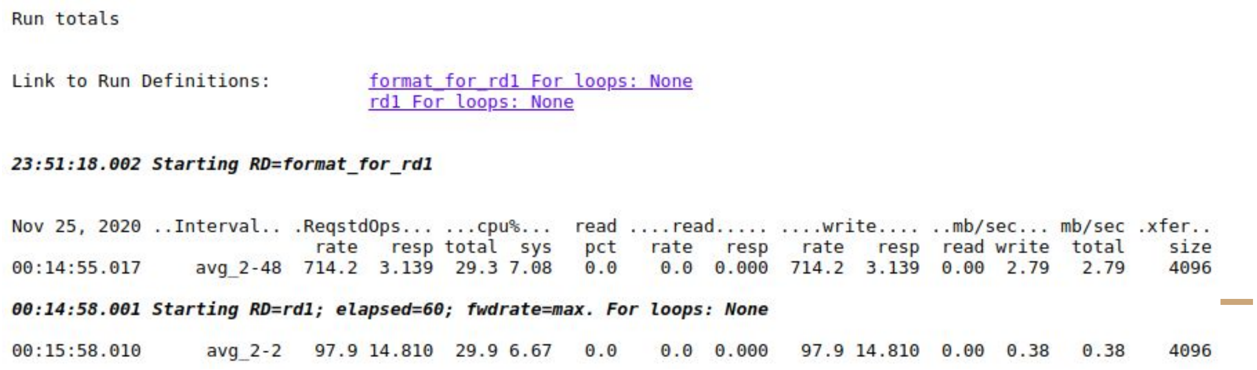
```
Run totals

Link to Run Definitions:          format_for_rd1 For loops: None
                                  rd1 For loops: None

19:24:39.002 Starting RD=format_for_rd1

Nov 24, 2020 ..Interval.. .ReqstdOps... ...cpu%...  read ....read..... ....write.... ..mb/sec... mb/sec .xfer..
                            rate   resp total  sys   pct   rate   resp   rate   resp  read write  total    size
20:40:41.028    avg_2-153  230.0 12.715  27.5 7.46   0.0    0.0  0.000  230.0 12.715  0.00  0.90   0.90    4096

20:40:43.001 Starting RD=rd1; elapsed=60; fwdrate=max. For loops: None

20:41:43.027      avg_2-2   36.7 852.71   6.7 2.62   0.0    0.0  0.000   36.7 852.71  0.00  0.14   0.14    4096
```

★ In the second case, the <u>deduplication feature was **disabled**</u> and the same procedure was repeated. It was observed that the total disk space used was **4.39 GB** which is much larger than the space used with deduplication enabled **(650 MB)**. The dedup value, in this case, is observed to be **1.00x** which indicates that no deduplication has occurred.



```
lord:/ imperial_lord$ sudo zpool list
NAME   SIZE  ALLOC   FREE   CAP  DEDUP  HEALTH  ALTROOT
gr12  14.2G  4.39G  9.86G   30%  1.00x  ONLINE  -
lord:/ imperial_lord$
```

From the statistics, it can be observed that the simulation runs for about **"23 mins"** to create all the files in the workload (Observe start and end time).



```
Run totals

Link to Run Definitions:          format_for_rd1 For loops: None
                                  rd1 For loops: None

23:51:18.002 Starting RD=format_for_rd1

Nov 25, 2020 ..Interval.. .ReqstdOps... ...cpu%...  read ....read..... ....write.... ..mb/sec... mb/sec .xfer..
                            rate   resp total  sys   pct   rate   resp   rate   resp  read write  total    size
00:14:55.017     avg_2-48  714.2  3.139  29.3 7.08   0.0    0.0  0.000  714.2  3.139  0.00  2.79   2.79    4096

00:14:58.001 Starting RD=rd1; elapsed=60; fwdrate=max. For loops: None

00:15:58.010      avg_2-2   97.9 14.810  29.9 6.67   0.0    0.0  0.000   97.9 14.810  0.00  0.38   0.38    4096
```

Workload Creation and Execution for comparing Compression

The workload was created using vdbench using **compratio=20.**

The meaning of parameters remains the same as above described in workload setup for deduplication features.

```
compratio=20
fsd=fsd1,anchor=!fanchor,depth=!dirdep,width=!dirwid,files=!numfiles,
size=!filesize
fwd=fwd1,fsd=fsd1,rdpct=!filrdpct,xfersize=!fxfersize,fileselect=rand
om,fileio=random,threads=!thrds
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=!etime,interval=!itime
```

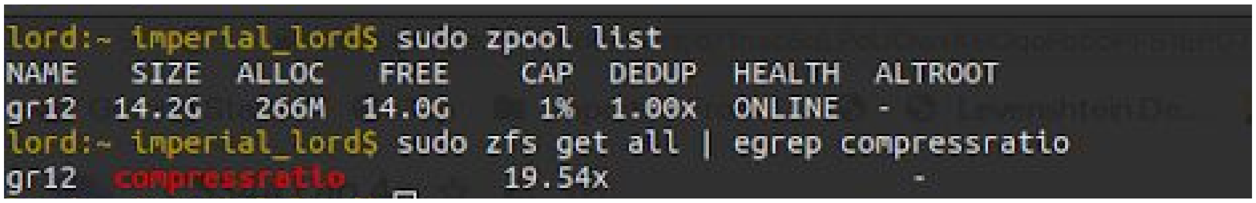
Command to run the workload

```
./vdbench -f source_comp fanchor=/gr12/dir1 dirwid=64
numfiles=512 filesize=128k fxfersize=128k thrds=64 etime=1m
itime=30 dirdep=1 filrdpct=0 -p 5576 -o group12_output_comp
```

Observations

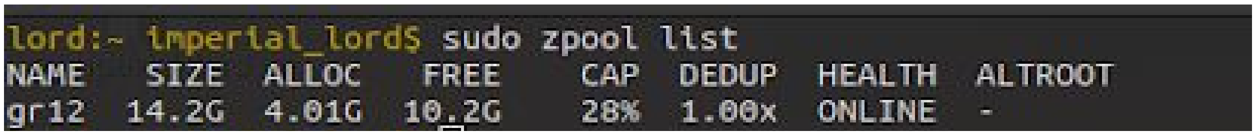
★ First, the compression feature was **enabled** and the vdbench workload was run to create files with **compression ratio** = **20**. A total of 32,768 (2^{15}) files (64 directories each with 512 files) each of size 128 kB was created which accounted for a total size equal to 4GB. In this case, after running the **"sudo zpool list"** command it can be observed that the total space in the disk used was equal to **"266MB"** which is very less than

"4GB". Also after running **"sudo zfs get all"** command, we can see that **"compress ratio = 19.54x".**



```
lord:~ imperial_lord$ sudo zpool list
NAME   SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
gr12  14.2G   266M  14.0G     1%  1.00x  ONLINE  -
lord:~ imperial_lord$ sudo zfs get all | egrep compressratio
gr12  compressratio        19.54x                    -
```

★ In the second case, the <u>compression feature was **disabled**</u> and the same procedure was repeated. It was observed by running "sudo zpool list" that the total disk space used was **4.01 GB** which is equal to the original size of files created. Hence no compression has occurred in this case.



```
lord:~ imperial_lord$ sudo zpool list
NAME    SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
gr12  14.2G  4.01G  10.2G    28%  1.00x  ONLINE  -
```

<u>Workload Creation and Execution for ext3 file system</u>

Ext3 is an old file system and does not support most features like data deduplication and compression that its modern counterparts (such as ZFS, NTFS, FAT etc.) do.

We created a vdbench workload to test the ext3 file system for data deduplication and compression features.

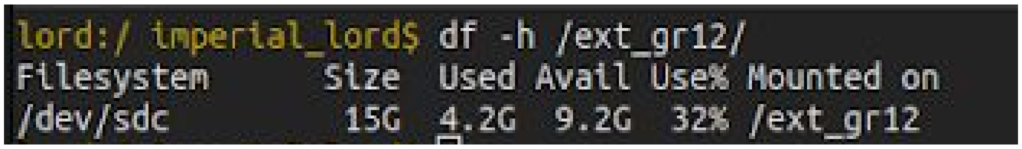
```
fsd=fsd1,anchor=!fanchor,depth=!dirdep,width=!dirwid,files=!numfiles,
size=!filesize
fwd=fwd1,fsd=fsd1,rdpct=!filrdpct,xfersize=!fxfersize,fileselect=rand
om,fileio=random,threads=!thrds
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=!etime,interval=!itime
```

Command to run the workload

```
./vdbench -f source_ext3 fanchor=/ext_gr12/dir1 dirwid=64
numfiles=512 filesize=128k fxfersize=128k thrds=64 etime=1m
itime=30 dirdep=1 filrdpct=0 -p 5576 -o group12_output_ext3
```

Observations

After running **"df -h /ext_gr12/"** command we can observe that the disk space used for creating the files is equal to **"4.2GB"** which is equal to the original size of files being created. So we can conclude that there is **no** compression and data deduplication feature available in ext3 file system.



```
lord:/ imperial_lord$ df -h /ext_gr12/
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdc         15G  4.2G  9.2G  32% /ext_gr12
```

COMPARISON : ext3 vs ZFS

From the study of various features in this lab, we make the following differentiation.

ZFS is a modern file management system that has several powerful features like deduplication, compression, COW, throughput improvements and so on. These make it an ideal candidate for use in several fields such as database management, big data handling etc.	Ext3 on the other hand is an old file system which lacks all the features and just provides basic file handling. This makes it unsuitable for sophisticated uses.

CONCLUSIONS

From the observations made in the study we can conclude following important points about **ZFS** and **ext3** file systems:

1) If we use data **deduplication** features for **ZFS** and have a workload which writes similar files on the disk, then **ZFS** can significantly reduce the disk space used for storing the data by writing duplicate blocks only once and creating pointers to further copies of duplicate blocks. However it can be noted that although disk space is reduced, it takes very large time compared to dedup feature off to write the same size of files on disk. This means there is more cpu utilization in case of deduplication.
2) If we use **compression** features for **ZFS** then ZFS can significantly reduce the disk space utilized for storing data by using compression algorithms and storing data in compressed form.
3) Data deduplication and compression features are not available in **ext3**, so no reduction in disk space to store the files is observed when we run the same workload on ext3 file system.

ZFS is a relatively new file system with many advanced features like **data deduplication**, and **compression** which we observed in this study. These features make **ZFS** suitable for use in **databases** and big data management systems . On the other hand, **ext3** is an older file system that lacks much sophistication, and also has a lower throughput. There are several other file systems out there such as **NTFS, FAT32, ext4** all of which have different advantages and are unique in their own ways!