ASSIGNMENT 4 OS LAB : FILE SYSTEMS

Group 6

Vani Krishna Barla	200101101
Mahek Vora	200101062
Siddharth Bansal	20010109

We have chosen **ZFS** and **EXT4** file systems.

ZFS:
ZFS (previously: Zettabyte file system) is a file system with volume management capabilities. It was created as part of the Sun Microsystems Solaris operating system in 2001. ZFS is designed for prioritizing security and it has optimized file and disk management. Many file and volume managers can't achieve this. It uniquely combines management of data and files that are stored on these logical block devices and physical volume management of one or more block storage devices. It provides a data deduplication feature too.

EXT4:
EXT4 focuses on reliability, performance and capacity. Data is allocated using extents(they are described by their starting and ending points on the hard drive) instead of fixed size blocks. This has 2 major advantages - reduced fragmentation and lesser number of pointers to store file data (as against storing pointers for all memory blocks that have file data). EXT4 implements delayed allocation thus allowing contiguous storage, as it knows the amount of memory required. This improves its performance. It does not support deduplication.

Features for comparison - Deduplication, Large file creation

Deduplication:

Data deduplication is a technique for eliminating duplicate copies of repeating data that can significantly improve storage utilisation and thus lower capital expenditure by reducing the overall amount of storage media required to meet storage capacity needs of a system.
The working of the process is described here briefly. Data "chunks"—unique, contiguous blocks of data—also referred to as "byte patterns"—must be compared as part of the deduplication process. This is typically done using hashing(e.g. secure hash like SHA256). These chunks are mapped to signatures in a hash table. The signature of new data can be compared to hash table entries and if a chunk is redundant, it can be replaced with a pointer/reference to the originally stored chunk whenever a match occurs.

This technique requires a high computational overhead so it is not used in all systems.
Deduplication can be implemented at file-level, block-level and byte-level. It can be synchronous (occurs as the data is being written) or asynchronous(redundant chunks are hashed and deleted when the CPU is free).
ZFS supports this, EXT4 does not.

Large file creation:

Extents (used in EXT4 as mentioned above) save space and time used to access/save huge mapping of blocks of data occupied by large files, so EXT4 is able to optimise large file creation and handling. Another EXT4 feature - multiblock allocation also helps in this case, where many blocks are allocated in a single call, instead of a one per call, this avoids a lot of overhead. Allocation of contiguous blocks of memory is also a plus, supported by EXT4 because of delayed allocation. Some statistics are mentioned here for comparison.

File system size supported

EXT4	1 EiB(ExbiByte)=2^{60} Bytes
EXT3	16TiB=2^{44} Bytes
ZFS	16TiB=2^{44} Bytes

EXT4 supports standard 4KiB blocks, with 48 bit block addressing. EXT3 supports 2TiB file size.

PTO

Experimental observations

Running workload on file systems:

workload1:
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

We are creating ***450 files (50*3*3) each of size 1MB*** in a nested folder structure of depth 2 and width 3. Then these files are being read sequentially for thirty seconds to monitor statistics (although this part is not important since the deduplication is done during file creation).

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* on the other hand is the size of the block which will be compared with pre-existing blocks to check for duplicates. We set it to 1MB because this is the size of one file. So basically, **half of the files will be duplicates of the other half**.

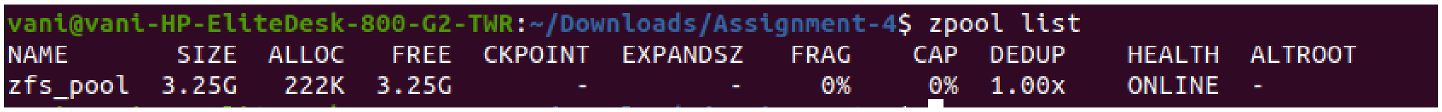
1. DEDUPLICATION FEATURE

a) ZFS has a data deduplication feature which we turned on using the command :
sudo zfs set dedup=on zfs_pool
b) We run this workload on the ZFS file system by setting anchor to the directory of the ZFS Pool (basically the folder pointing to the ZFS Pool) :
sudo ./vdbench -f workload1 anchor=/zfs_pool
c) We run this workload on the ext4 file system by setting anchor to the directory of the folder where the ext4 drive is mounted:
sudo ./vdbench -f workload1 anchor=/ext4_fs

Results:

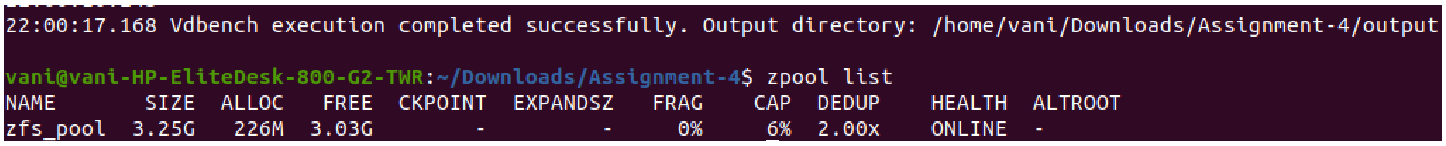
i) ZFS
After running the workload, the ZFS folder had 226 MB of data. We observed a deduplication ratio of 2x. The new files took 225 MB of space, however, the intended space is 450MB (1MB*450). Hence, using the data deduplication feature, instead of maintaining whole blocks of data, when duplicates are found, ZFS simply makes a pointer to the old data.

Before workload:



```
vani@vani-HP-EliteDesk-800-G2-TWR:~/Downloads/Assignment-4$ zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool   3.25G   222K  3.25G        -         -     0%     0%  1.00x    ONLINE  -
```

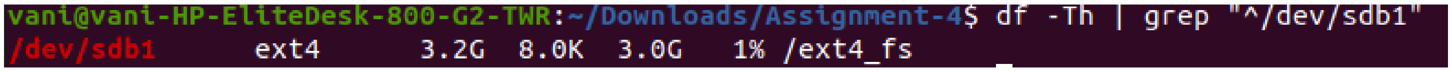
After workload:



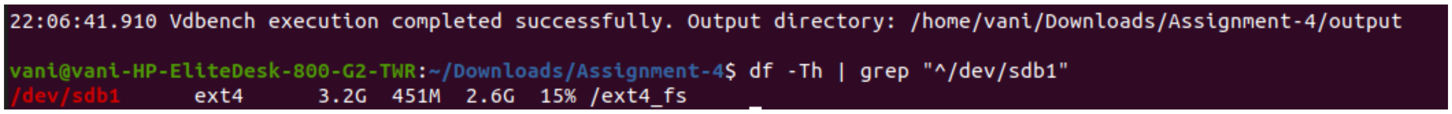
```
22:00:17.168 Vdbench execution completed successfully. Output directory: /home/vani/Downloads/Assignment-4/output

vani@vani-HP-EliteDesk-800-G2-TWR:~/Downloads/Assignment-4$ zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool   3.25G   226M  3.03G        -         -     0%     6%  2.00x    ONLINE  -
```

ii) EXT4

The disk partition originally has 8KB allocated space, after running workload used space increases to 451MB. Indicating 450 MB space of files created.



```
vani@vani-HP-EliteDesk-800-G2-TWR:~/Downloads/Assignment-4$ df -Th | grep "^/dev/sdb1"
/dev/sdb1      ext4      3.2G  8.0K  3.0G   1% /ext4_fs
```



```
22:06:41.910 Vdbench execution completed successfully. Output directory: /home/vani/Downloads/Assignment-4/output

vani@vani-HP-EliteDesk-800-G2-TWR:~/Downloads/Assignment-4$ df -Th | grep "^/dev/sdb1"
/dev/sdb1      ext4      3.2G  451M  2.6G  15% /ext4_fs
```

workload2:

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

2. LARGE FILE CREATION FEATURE

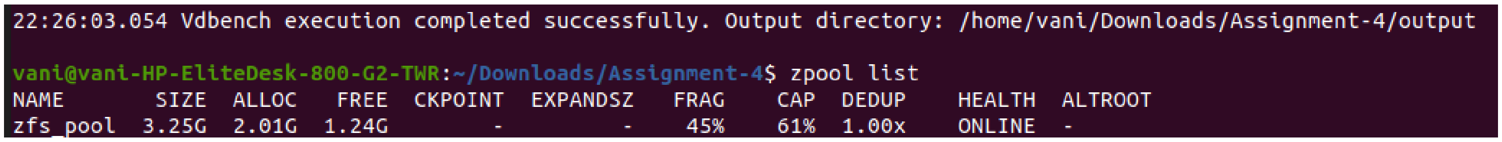
Our **workload2** is creating two files of size 1GB in one folder. The operation used is “create” since we are testing file creation.

a. We run this on the ZFS file system by setting anchor equal to the directory pointing to the ZFS pool:
   sudo ./vdbench -f workload2 anchor=/zfs_pool
b. We run this workload on the ext4 file system by setting anchor equal to the directory pointing to the ext4 drive:
   sudo ./vdbench -f workload2 anchor=/ext4_fs

Result:

i) ZFS

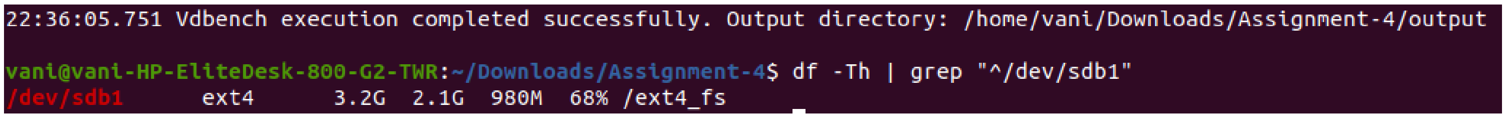
Takes up more time to create the files. (372 secs)



```
22:26:03.054 Vdbench execution completed successfully. Output directory: /home/vani/Downloads/Assignment-4/output

vani@vani-HP-EliteDesk-800-G2-TWR:~/Downloads/Assignment-4$ zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool   3.25G  2.01G  1.24G        -         -    45%    61%  1.00x    ONLINE  -
```

ii) EXT4

Finishes earlier (176 secs), as ext4 optimises large file creation better than ZFS.



```
22:36:05.751 Vdbench execution completed successfully. Output directory: /home/vani/Downloads/Assignment-4/output

vani@vani-HP-EliteDesk-800-G2-TWR:~/Downloads/Assignment-4$ df -Th | grep "^/dev/sdb1"
/dev/sdb1      ext4      3.2G  2.1G  980M  68% /ext4_fs
```

Disadvantages of deduplication:

1. Performance:
In the first workload, the ZFS system set up the file system in **43 seconds** whereas ext4 just took **1 seconds**. ZFS had an **average write speed** of 9.76 MB/s while ext4 had an average write speed of 450 MB/s. In the second workload too, as we have seen above in the 'large file optimisation' section, ext4 was significantly faster. This is partially due to large file optimisation of ext4 and partially due to the deduplication overhead of ZFS. This shows that deduplication harms performance of a file system due to overhead.

2. CPU Utilisation:
In the first workload, during file structure setup (the other section is for read which doesn't have much to do with deduplication), the average **CPU system utilisation** of ZFS was 47.8% while that of ext4 was 35.3% which is significantly lower. In the second workload too, the CPU system utilisation of ZFS was 2.41% while that of ext4 was 2.58%. This shows that the deduplication feature has significantly higher CPU usage in the first workload which uses deduplication. This is due to deduplication overhead.
Since ZFS stored pointers for the duplicate data, the **response time** for reading those data blocks becomes more as compared to ext4. This can be observed in our experiment where the average response time for ZFS was about 5 milliseconds, which was much lower for ext4 at 2 milliseconds (in workload 1).

Disadvantages of Optimising Large File Creation:

1. Greater metadata overhead for small files:
When running workload1, only 450 MB was required by the files. But, additional used space after running workload1 was 451 MB (~1 MB of overhead). In ZFS however, the overhead was very small (<700 KB). Additional space is used in maintaining the extent trees compared to the actual data (for small files).

2. No possible recovery from corruption:
Ext4 optimises large file creation by using delayed and contiguous allocation, and extents. This makes it impossible for any data correction mechanisms to exist since very little metadata is stored for large files stored in many contiguous blocks.

Viewing output
After running a workload on a particular filesystem, the summary of the run is stored in **summary.html** in the **output** folder in the **vdbench** directory. The *output* directory of each workload is submitted in zip.