CS344 (OS Lab) Assignment 4

Group 6 Members:

Adarsh Gupta	220101003
Aditya Nanda	220101006
Aryan Raj	220101019
Ashish Raj	220101020

Table of Contents:

Code Location:

https://github.com/CoolSunflower/ZFSvsEXT4

CS344Assignment4Group6

https://drive.google.com/drive/folders/1veDnkk-0AK9eyuL9w9EMRvwwwd13GL-Q?usp=sharing

Part 1: Selecting the FS and Feature

File Systems Selected:

- ext4
- ZFS

Features Selected:

- Large File Creation & Management
- Data Deduplication

File Systems Selected for Comparison

ZFS

ZFS (Zettabyte File System) is a high-performance, open-source file system originally developed by Sun Microsystems. What sets ZFS apart from other filesystems is that it functions as **both a file system and a volume manager**, which means it not only manages files and directories but also handles the allocation of storage across physical devices. This integrated approach streamlines storage management, eliminating the need for separate tools to manage disks and volumes.

Security and reliability are at the core of ZFS's design. Every operation in ZFS—from data writes to metadata updates—undergoes **verification checks** to prevent data corruption. Features like **checksumming** ensure that data on the disk is consistent, and if errors are detected, ZFS can correct them automatically. ZFS also supports **snapshots**, **copy-on-write** (CoW), and **data deduplication**, making it a powerful choice for enterprise storage and backup systems.

One of the standout features of ZFS is **data deduplication**, which we'll discuss in more detail later. In essence, deduplication ensures that identical data blocks are only stored once, significantly reducing disk usage in environments with redundant data, such as virtual machines and backup servers. However, the complexity and memory overhead involved in features like deduplication mean that ZFS is best suited for systems with high-performance hardware.

EXT4

The **Fourth Extended Filesystem (ext4)** is the successor to ext2 and ext3 and is one of the most widely used filesystems on Linux systems. Unlike ZFS, ext4 emphasises **speed, simplicity, and compatibility** over advanced data management features. This makes it ideal for personal computers, workstations, and general-purpose servers where low overhead and fast access are more important than data integrity guarantees.

A key innovation in ext4 is its use of **extents** to manage large files. Instead of tracking every block individually (as older file systems do), ext4 uses contiguous ranges of blocks called **extents**. This reduces fragmentation and makes it faster to read and write large files, as fewer disk operations are needed. For example, a large video file can be stored as a single extent, improving performance and reducing the number of pointers needed to manage the file.

Another performance boost comes from **delayed allocation**. Instead of allocating disk space immediately, ext4 waits until the data is ready to be written, allowing it to optimise block placement. This helps ensure that files are stored in **contiguous regions** on the disk, minimising fragmentation and improving read/write speeds.

While ext4 lacks some of the advanced features offered by ZFS (such as deduplication and checksumming), it more than makes up for it with **speed, efficiency, and low resource usage**. It strikes a balance between performance and simplicity, which is why it is the default filesystem on many Linux distributions.

Features selected for comparison

Large File Creation and Management

Large file creation and management refers to the ability of a file system to efficiently create, store, and handle large files without performance degradation. For workloads involving video files, disk images, or scientific datasets, having a filesystem optimised for large files is crucial.

Supported Filesystem: ext4

- **Implementation in ext4:**
  ext4 optimises large file management through several techniques:
  - **Extents Mapping:** Instead of tracking individual blocks, ext4 uses extents, which are contiguous chunks of disk space. This reduces fragmentation and makes access to large files faster.
  - **Delayed Allocation:** ext4 postpones the allocation of disk blocks until the file is flushed to disk. This helps the filesystem optimize

block placement, improving write performance and reducing fragmentation.
    - **Preallocation:** Applications can reserve disk space in advance using the `fallocate()` system call. This ensures that large files have contiguous space, minimising future fragmentation.
- **How ext4 Benefits Large Files:**
  With these optimizations, ext4 provides excellent performance when creating, storing, and accessing large files. It's particularly useful in environments where fast writes and minimal fragmentation are necessary, such as media storage and general-purpose file management. However, ext4 focuses more on performance than on advanced data integrity features.

Trade-offs:

- While ext4 is fast and lightweight, it lacks advanced integrity features like checksumming. This makes it more vulnerable to silent data corruption compared to filesystems like ZFS.
- It also doesn't support features such as CoW, meaning data changes are written directly to disk, which can expose data to risks in the event of crashes during write operations.

Data Deduplication:

Data deduplication is a technique that identifies and eliminates duplicate data blocks to save storage space. It is particularly valuable in environments with repetitive data, such as virtual machine images or backups, where many files contain identical data.

Supported Filesystem: ZFS

- **Implementation in ZFS:**
  ZFS performs deduplication by generating a **hash** (e.g., SHA-256) for every data block written to the disk. If two blocks generate the same hash, ZFS stores only one copy of the block and updates the metadata to point to this block for all relevant files.
    - **Copy-on-Write (CoW):** When a file sharing a deduplicated block is modified, ZFS creates a new version of the block instead of overwriting the original. This preserves data integrity and ensures that no files are accidentally corrupted.

    - **Adaptive Replacement Cache (ARC):** ZFS keeps deduplication tables in memory to speed up lookups. However, large datasets may require more memory to maintain these tables efficiently.
- **How Deduplication Saves Space:**
  In environments with repetitive data (e.g., backups or VMs), deduplication can significantly reduce disk usage. ZFS ensures that only one copy of duplicate data is stored, freeing up valuable space.

Trade-offs:

- While deduplication provides excellent space savings, it comes at the cost of **higher CPU and memory usage**. Maintaining deduplication tables requires significant RAM, and computing hashes adds overhead to the system.
- Deduplication works best with large datasets and can cause performance degradation in systems with limited memory or heavy write workloads.

Part 2: Quantifying the benefits

Large File Creation & Management

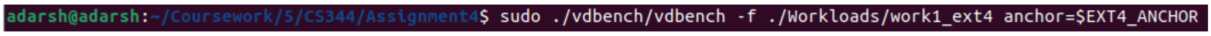
VDBench Workload used for testing and comparing large file creation and management in ZFS and ext4:

```
fsd=fsd1,anchor=/zfs_pool_tests,depth=0,width=1,files=2,size=1G
fwd=fwd1,fsd=fsd1,operation=create,fileio=sequential,fileselect=random,threads=4
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=60,interval=5
```

This workbench creates 2 files of size 1 GB each.

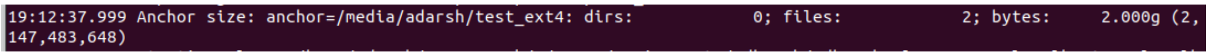
ext4 FS

Command Executed:

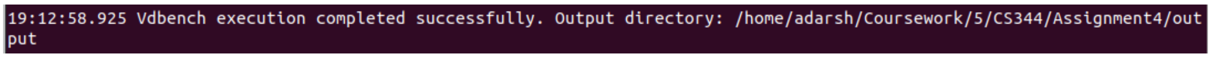


```
adarsh@adarsh:~/Coursework/5/CS344/Assignment4$ sudo ./vdbench/vdbench -f ./Workloads/work1_ext4 anchor=$EXT4_ANCHOR
```

Successful Run:

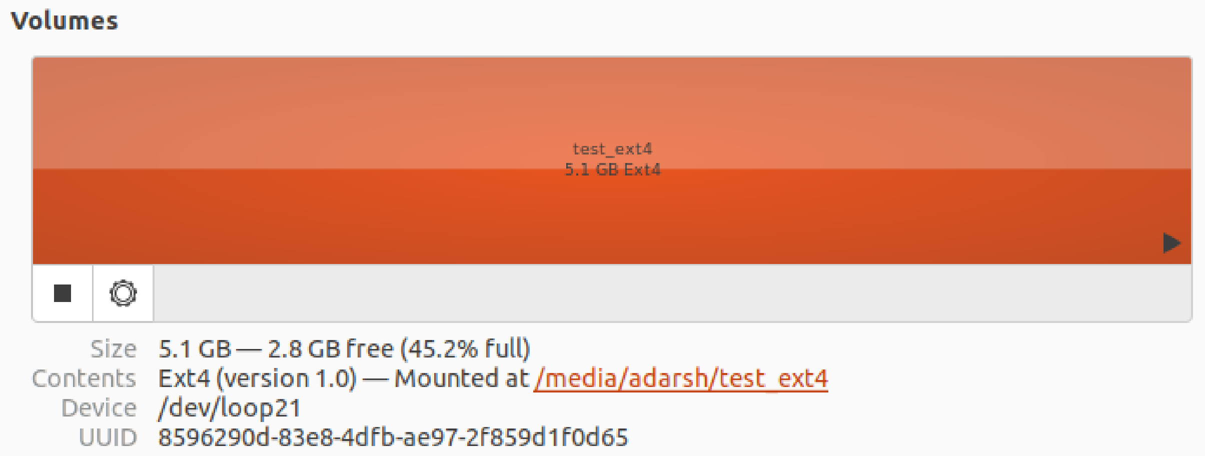


```
19:12:37.999 Anchor size: anchor=/media/adarsh/test_ext4: dirs:          0; files:          2; bytes:     2.000g (2,
147,483,648)
```

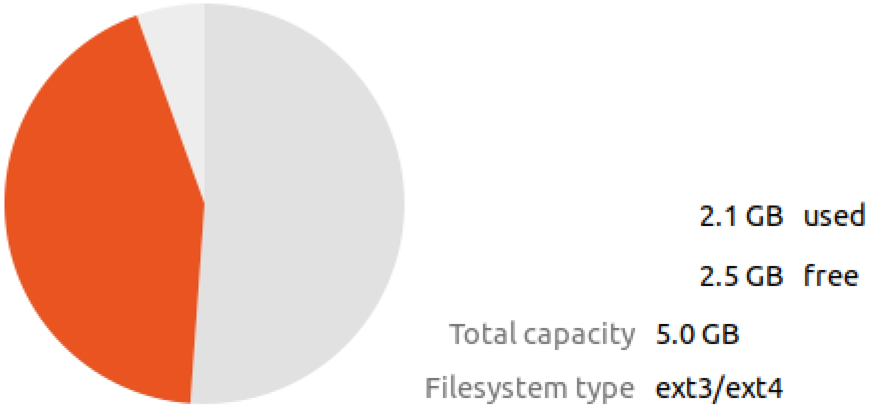


```
19:12:58.925 Vdbench execution completed successfully. Output directory: /home/adarsh/Coursework/5/CS344/Assignment4/out
put
```

After Workload:



Size 5.1 GB — 2.8 GB free (45.2% full)
Contents Ext4 (version 1.0) — Mounted at /media/adarsh/test_ext4
Device /dev/loop21
UUID 8596290d-83e8-4dfb-ae97-2f859d1f0d65



2.1 GB used
2.5 GB free
Total capacity 5.0 GB
Filesystem type ext3/ext4

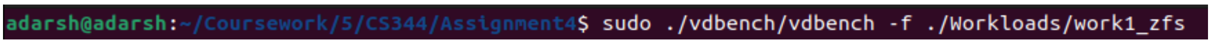
Execution Statistics:

Total Time Taken: 16 Seconds

Average Write Speed: 128 Mb/s

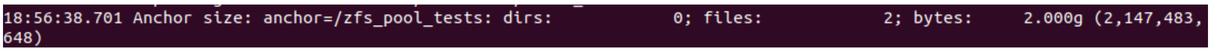
ZFS FS

Command Executed:

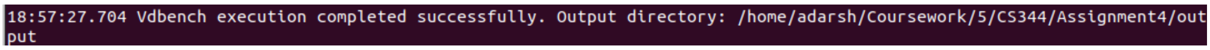


```
adarsh@adarsh:~/Coursework/5/CS344/Assignment4$ sudo ./vdbench/vdbench -f ./Workloads/work1_zfs
```

Successful Run:

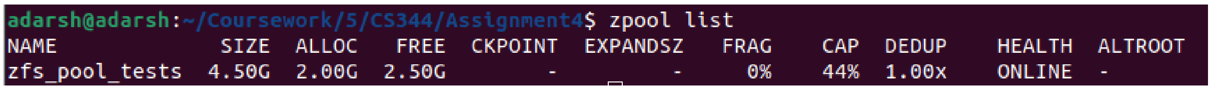


```
18:56:38.701 Anchor size: anchor=/zfs_pool_tests: dirs:          0; files:          2; bytes:    2.000g (2,147,483,
648)
```



```
18:57:27.704 Vdbench execution completed successfully. Output directory: /home/adarsh/Coursework/5/CS344/Assignment4/out
put
```

After Workload:



```
adarsh@adarsh:~/Coursework/5/CS344/Assignment4$ zpool list
NAME             SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool_tests  4.50G  2.00G  2.50G        -         -     0%    44%  1.00x    ONLINE  -
```

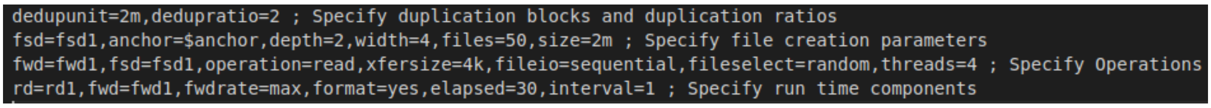
Execution Statistics:

Total Time Taken: 46 Seconds

Average Write Speed: 44.51 Mb/s

Data Deduplication

VDBench Workload used for testing data deduplication strategy benefits in ZFS file system:



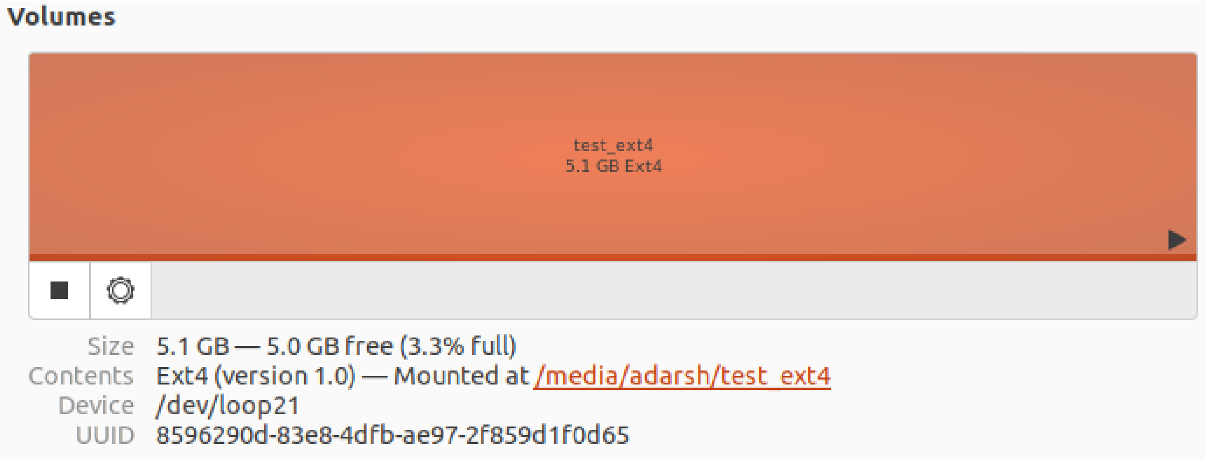
```
dedupunit=2m,dedupratio=2 ; Specify duplication blocks and duplication ratios
fsd=fsd1,anchor=$anchor,depth=2,width=4,files=50,size=2m ; Specify file creation parameters
fwd=fwd1,fsd=fsd1,operation=read,xfersize=4k,fileio=sequential,fileselect=random,threads=4 ; Specify Operations
rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=30,interval=1 ; Specify run time components
```

This workload creates a nested folder directory of depth 2 and width 4 with 50 files in each folder, therefore at the first level there are 4 directories and at the second level there are 16 directories. These 16 directories contain 50 files each totalling 800 files, with 2MB per file totalling approximately 1.563GB.
These files are then read sequentially. The total space occupied by these files will be
The value of dedupratio (= 2) specifies the ratio of total number of blocks (block size = 2MB) to the number of blocks containing unique data. dedupunit is the size of the block which will be compared with pre-existing blocks to check for duplicates. We set it to equal the file size which is 2MB. So overall, half the files will be duplicates of the other half.

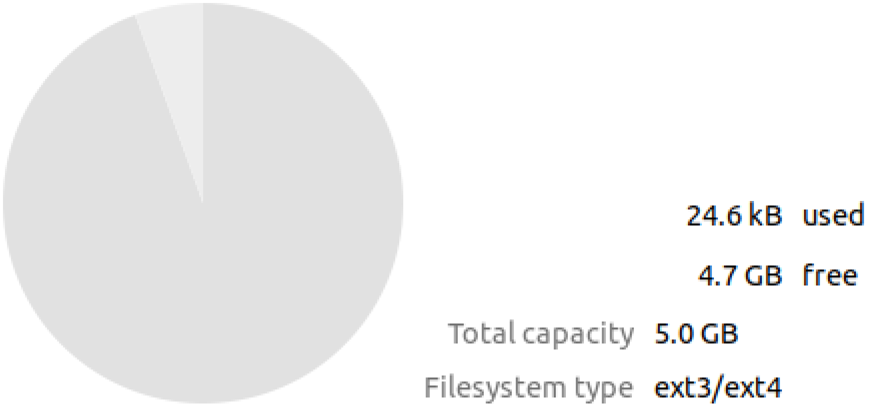
ext4 FS

Using the exported anchor (EXT4_ANCHOR)(see README) we can run the following command:

Before Workload:

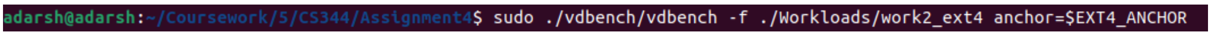


Size 5.1 GB — 5.0 GB free (3.3% full)
Contents Ext4 (version 1.0) — Mounted at /media/adarsh/test_ext4
Device /dev/loop21
UUID 8596290d-83e8-4dfb-ae97-2f859d1f0d65



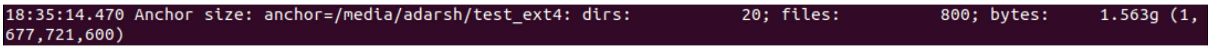
24.6 kB used
4.7 GB free
Total capacity 5.0 GB
Filesystem type ext3/ext4

Command Executed:

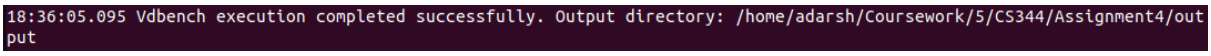


```
adarsh@adarsh:~/Coursework/5/CS344/Assignment4$ sudo ./vdbench/vdbench -f ./Workloads/work2_ext4 anchor=$EXT4_ANCHOR
```

Successful Run:

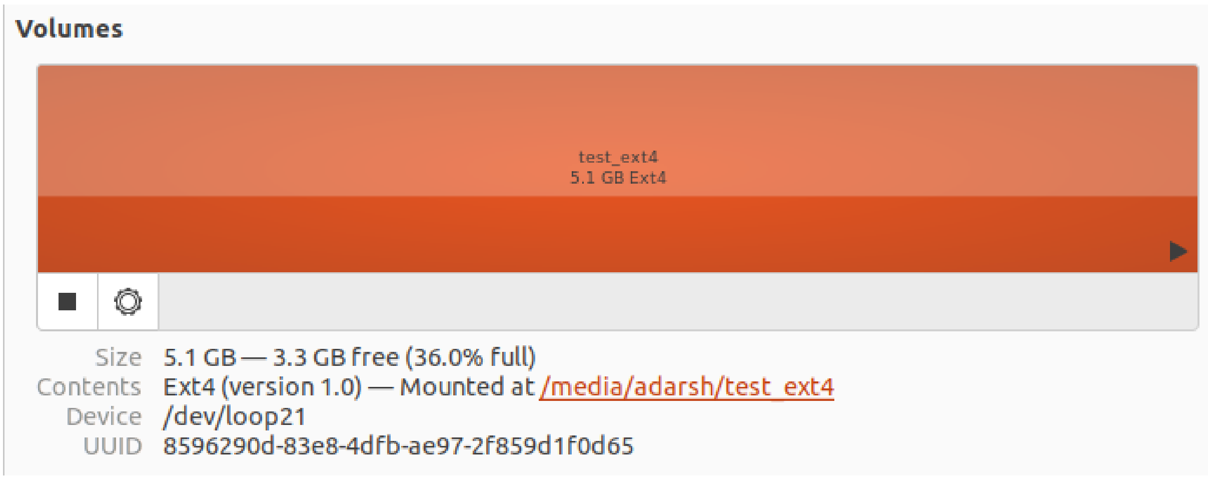


```
18:35:14.470 Anchor size: anchor=/media/adarsh/test_ext4: dirs:         20; files:          800; bytes:     1.563g (1,
677,721,600)
```

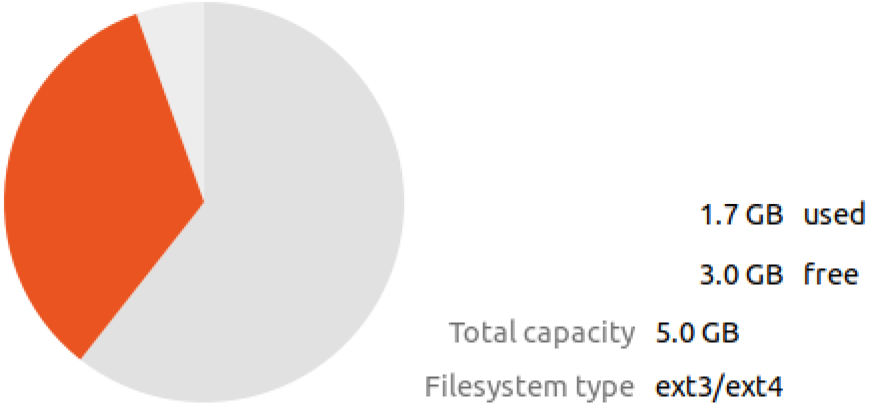


```
18:36:05.095 Vdbench execution completed successfully. Output directory: /home/adarsh/Coursework/5/CS344/Assignment4/out
put
```

After Workload:



Size 5.1 GB — 3.3 GB free (36.0% full)
Contents Ext4 (version 1.0) — Mounted at /media/adarsh/test_ext4
Device /dev/loop21
UUID 8596290d-83e8-4dfb-ae97-2f859d1f0d65



1.7 GB used

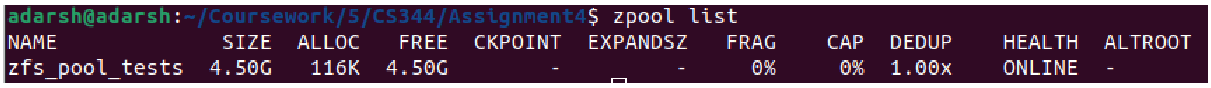
3.0 GB free

Total capacity 5.0 GB

Filesystem type ext3/ext4

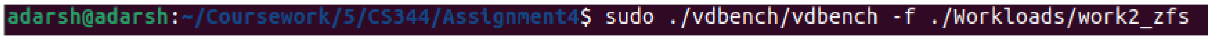
ZFS FS

Before Workload:



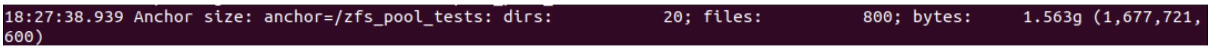
```
adarsh@adarsh:~/Coursework/5/CS344/Assignment4$ zpool list
NAME               SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool_tests   4.50G   116K  4.50G        -         -     0%     0%  1.00x    ONLINE  -
```

Command Executed:

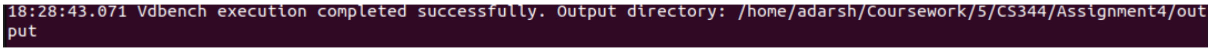


```
adarsh@adarsh:~/Coursework/5/CS344/Assignment4$ sudo ./vdbench/vdbench -f ./Workloads/work2_zfs
```

Successful Run:

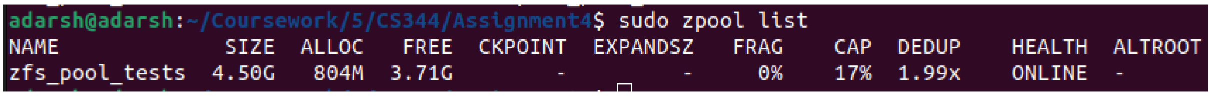


```
18:27:38.939 Anchor size: anchor=/zfs_pool_tests: dirs:          20; files:          800; bytes:     1.563g (1,677,721,
600)
```



```
18:28:43.071 Vdbench execution completed successfully. Output directory: /home/adarsh/Coursework/5/CS344/Assignment4/out
put
```

After Workload:



```
adarsh@adarsh:~/Coursework/5/CS344/Assignment4$ sudo zpool list
NAME               SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool_tests   4.50G   804M  3.71G        -         -     0%    17%  1.99x    ONLINE  -
```

Part 3: Quantifying the disadvantages

Disadvantages of optimising Large File Creation and Management:

The ext4 filesystem optimises the management of large files by using **extents**, delayed allocation, and contiguous block allocation. These optimizations boost performance by reducing fragmentation and speeding up large file operations. However, these improvements come with some **downsides**, particularly when handling small files and ensuring data integrity.

1. Increased Metadata Overhead for Small Files

Although ext4 performs well with large files, it suffers from **excessive metadata overhead** when working with small files. During the experiment, we observed a discrepancy between the size of the files written and the actual space consumed.

- **Workload Size:** The workload required only **1.56 GB** of space.
- **Actual Space Used:** ext4 reported **1.7 GB** used, meaning an additional **140 MB of overhead**.

This overhead occurs because **extents** are optimised for large, contiguous data. For small files, however, maintaining extents and block pointers results in **unnecessary overhead**. In contrast, ZFS was more efficient for small files, with minimal overhead for metadata.

2. Limited Data Recovery Options

Another downside of ext4's optimization strategy is the **lack of robust data recovery mechanisms**. Since ext4 minimises the amount of metadata stored for large files, it becomes difficult to implement effective error correction.

- **Delayed Allocation:** While this improves performance, it also makes ext4 vulnerable to **data loss during sudden crashes**, as unsaved data might still be in memory when the system goes down.
- **Lack of Checksumming:** Unlike ZFS, which maintains **checksums** for every block to detect and correct corruption, ext4's design focuses on minimising metadata. As a result, **file integrity checks are limited**, making it challenging to recover from data corruption.

This trade-off means ext4 is better suited for **performance-oriented workloads** but is less ideal in situations where **data integrity and recovery** are a priority, such as in enterprise storage or backup systems.

Disadvantages of Data Deduplication:

Data deduplication in ZFS eliminates redundant copies of identical data, which saves disk space and is especially useful in backup environments or with virtual machines. However, this feature introduces several downsides, particularly in terms of **performance and CPU usage**.

1. Performance Overhead

Our experiments showed that enabling deduplication in ZFS adds a significant **performance penalty** to write operations. The time taken to set up the filesystem and write data was noticeably higher compared to ext4.

- **Setup Time:** ZFS took **65 seconds** to initialise and set up the filesystem, while ext4 completed the same task in just **40 seconds**.
- **Write Speed:** The average write speed for ZFS was **12.36 MB/s**, while ext4 achieved a much higher speed of **42.5 MB/s**.

This performance hit occurs because ZFS has to **scan, compare, and index blocks** before writing them to disk, adding overhead to every write operation. On workloads involving many similar files, the deduplication feature slows down data writes, which may be undesirable for applications requiring high-speed I/O.

2. Increased CPU Utilisation

One of the most noticeable trade-offs with ZFS deduplication is the significant increase in **CPU usage**. Deduplication requires ZFS to hash, compare, and manage block references constantly, which consumes more processing power.

- **During Setup:** ZFS used an average of **21.9% CPU** compared to **16.1%** in ext4.
- **During Workload:** ZFS reached **17.8% CPU usage**, while ext4 maintained a more modest usage of **8.1%**.

> It’s clear that CPU Utilisation for Zfs is more than ext4 ,but at any point of time CPU Utilisation during workload is less than that of during setup.This is because the time for setup in Zfs is 30 seconds and for ext4 it’s 13 seconds, also workload time for Zfs is 30 second and for ext4 is also 30 seconds. Clearly showing more CPU utilisation is during workload than setup.

This overhead may not be a problem for systems with powerful processors but can **degrade performance on CPU-constrained devices** or when multiple high-demand processes run simultaneously.