Assignment 4

ZFS:

ZFS (Zettabyte File System) is a robust filesystem and volume manager, created by Sun Microsystems, commonly used on UNIX systems like Linux and FreeBSD. Designed for high data integrity, ZFS uses **checksums** to detect and correct data errors, **deduplication** to save space, **copy-on-write (COW)** for efficient snapshot creation, and **compression** for storage efficiency. ZFS also supports **pooled storage** across multiple disks, **snapshots and clones** for backup, and is highly **scalable** (up to 256 trillion zettabytes).

Advantages:

- Strong reliability with data correction and protection
- Space-saving with deduplication and compression
- Easy-to-use backup through snapshots and clones
- Optimized for large datasets

Disadvantages:

- High memory usage, especially with deduplication
- Complexity in setup and maintenance
- Slower write performance in certain configurations

Use Cases: ZFS is ideal for server, cloud storage, and backup environments where data integrity, efficiency, and scalability are crucial.

EXT4:

The **EXT4** filesystem, widely used in Linux, builds on EXT2 and EXT3 with enhanced performance, reliability, and scalability. Key features include journaling for data protection, support for large files and volumes (up to **16 TB** files and **1 EB** volumes), extent-based storage to reduce fragmentation, delayed allocation for faster write speeds, and backward compatibility with **EXT3** and **EXT2.** Ext4's pre-allocation and improved timestamps make it efficient for handling large files.

Advantages:

- High reliability and data integrity

- Optimized for large files with minimal fragmentation

- Lightweight and backward-compatible

Disadvantages:

- Lacks advanced data integrity, deduplication, compression, and snapshot capabilities

Use Cases:

Ext4 is suitable for general Linux desktop and server use, particularly in performance-sensitive environments where simplicity is prioritized.

Here is a description of both the features we have picked to compare for this assignment:

Deduplication:

Deduplication is a data optimization technique that eliminates duplicate copies of data to save storage space and improve efficiency. By identifying and removing redundant data, deduplication allows only unique pieces of data to be stored, while duplicate instances are replaced with references to the original data. This technique is especially useful in environments where the same or similar data is repeatedly stored, such as backups, virtual machines, or large file systems.

Pros

- **Storage Efficiency:** Reduces storage needs by 50-90%.

- **Cost Savings**: Less storage hardware required.

- **Improved Backup/Recovery:** Less data to back up.

Cons

- **High CPU Usage:** Intensive processing to find duplicates.

- **Slower Write Performance:** Due to deduplication processing.

- **Data Rehydration Overhead**: Reconstructing data from deduplicated storage can slow read speeds.

Large File Creation:

Large file creation is the process of writing large files (hundreds of MBs or GBs), essential for performance in data-intensive systems. Modern filesystems use various techniques to manage large files efficiently, reduce fragmentation, and speed up access times.

Key Techniques

1. **Extent-based Allocation:** Groups blocks into contiguous extents to reduce fragmentation.

2. **Delayed Allocation: Holds** data in memory temporarily, creating fewer, larger writes.

3. **Pre-allocation:** Reserves space for growing files, minimizing fragmentation.

4. **Write Coalescing:** Combines smaller writes into larger operations for speed.

5. **Efficient Metadata Handling:** Reduces performance impact when generating metadata.

Benefits: These techniques improve performance, reduce storage overhead, and enhance reliability, making filesystems like ext4, XFS, and NTFS suitable for handling large files in media, databases, and big data applications.

1. **Deduplication:**
   a. ZFS has a data deduplication feature which we turned on using (zfs_pool is the name of the zfs pool we set up):

```
rohan@rohan-virtual-machine:~$ sudo zfs set dedup=on zfs_pool
```

b. We created the following workload for data deduplication (**workload1**). We will use this workload to compare the space occupied by the new files in ZFS with the space occupied in ext4.

```
1 dedupunit=1m,dedupratio=2
2 fsd=fsd1,anchor=$anchor,depth=2,width=3,files=40,size=1m
3 fwd=fwd1,fsd=fsd1,operation=read,xfersize=4k,fileio=sequential,fileselect=random,threads=2
4 rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=30,interval=1
```

Workload 1

c. **File Creation**: A total of 360 files (calculated as 40*3*3), each 1MB in size, are generated within a nested folder structure that has a depth of 2 and a width of 3.
d. **File Reading**: The files are read sequentially for thirty seconds to monitor statistics, although this reading process is not critical since deduplication occurs during file creation.
e. **Deduplication Settings**:
   - The **deduplication unit** is set to 1MB, which is the size of each file.
   - The **deduplication ratio** is set to 2, indicating that for every block of data, there are two blocks being compared, one of which contains unique data. This setup means that half of the files are duplicates of the other half.
f. **File System**: This workload is executed on the ZFS file system, with the operation directed to a specified directory within the ZFS Pool.

```
rohan@rohan-virtual-machine:~/Desktop/vdbench$ sudo ./vdbench -f workload1 anchor=/zfs_pool
[sudo] password for rohan:


Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Vdbench distribution: vdbench50407 Tue June 05  9:49:29 MDT 2018
For documentation, see 'vdbench.pdf'.

23:38:55.943 input argument scanned: '-fworkload1'
23:38:55.951 input argument scanned: 'anchor=/zfs_pool'
23:38:56.198 Anchor size: anchor=/zfs_pool: dirs:        12; files:        360; bytes:    360.000m (377,487,360)
23:38:56.468 Starting slave: /home/rohan/Desktop/vdbench/vdbench SlaveJvm -m localhost -n localhost-10-241029-23.38.55.685 -l localhos
t-0 -p 5570
```

g. We run this workload on the ext4 file system by setting anchor to the directory of the folder pointing to the ext4 drive:

```
rohan@rohan-virtual-machine:~/Desktop/vdbench$ sudo ./vdbench -f workload1 anchor=/media/rohan/ext_disk


Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Vdbench distribution: vdbench50407 Tue June 05  9:49:29 MDT 2018
For documentation, see 'vdbench.pdf'.
```

h. We found the following results:
   i. **ZFS**:

- Initially, the empty ZFS folder contained 105KB of data.
- After running the workload, the ZFS folder grew to 181MB of data.
- A deduplication ratio of 1.99x was observed, which met our expectations.
- The new files occupied 181MB of space, while the intended space was 360MB (1MB * 360).
- Thanks to the data deduplication feature, instead of storing complete blocks of duplicate data, ZFS creates pointers to the existing data when duplicates are found.
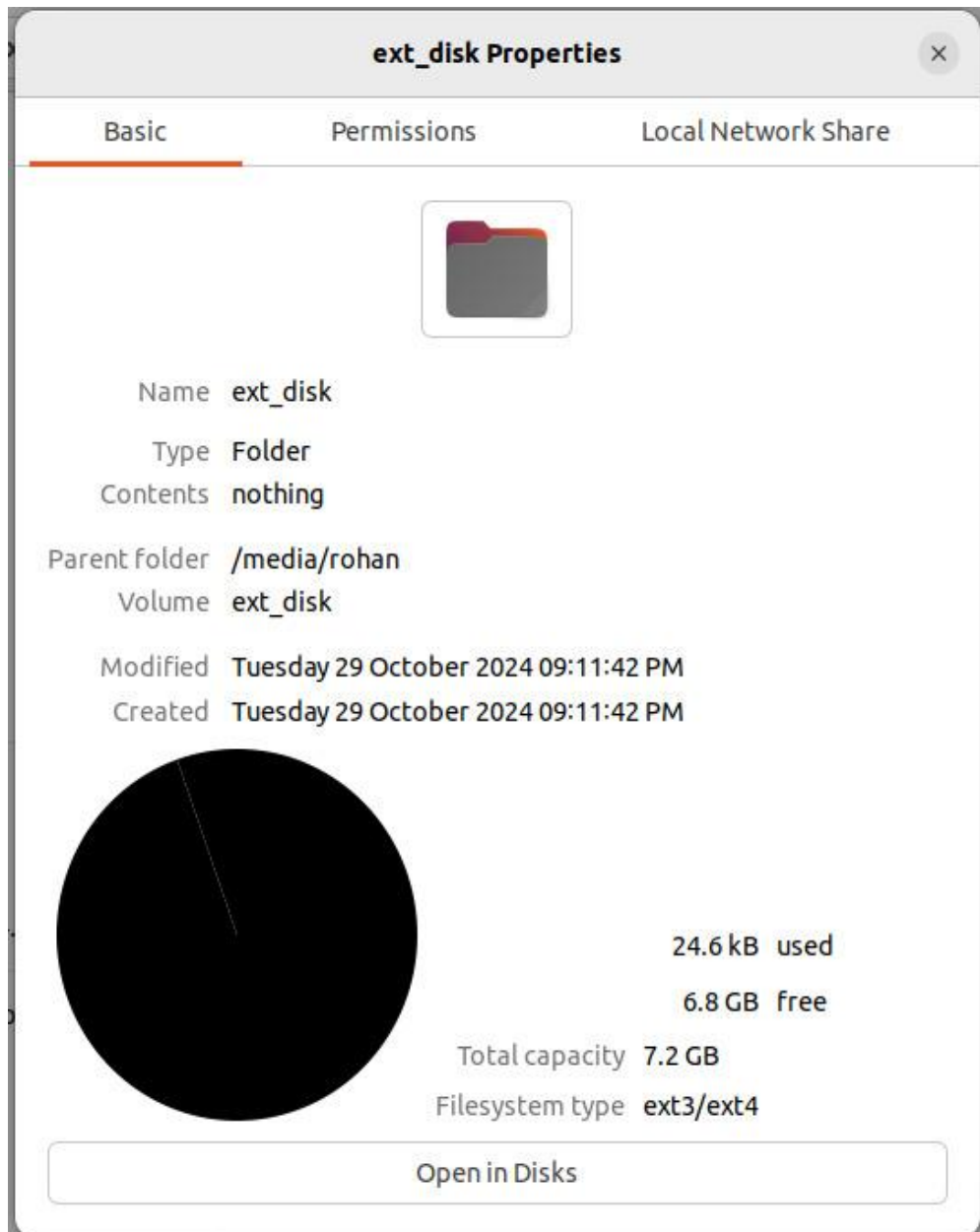
Before Workload:

```
rohan@rohan-virtual-machine:~/Desktop/vdbench$ zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ    FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool     7G   105K   7.00G       -         -      0%     0%  1.00x    ONLINE  -
```

After Workload:

```
rohan@rohan-virtual-machine:~/Desktop/vdbench$ zpool list
NAME        SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ    FRAG    CAP  DEDUP    HEALTH  ALTROOT
zfs_pool     7G   181M   6.82G       -         -      0%     2%  1.99x    ONLINE  -
```

ii.**ext4**:

- Initially the empty ext4 folder had **24.6kB** of data.
- After running the workload, the ext4 folder had **377.6MB** of data.
- The new files thus took **377.6MB** of space (a little more than intended because of metadata overhead).

Before Workload:

ext_disk Properties

Basic   Permissions   Local Network Share

Name ext_disk
Type Folder
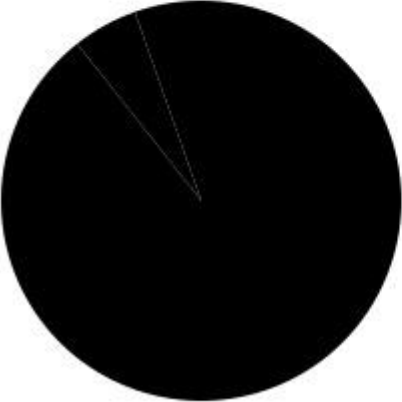Contents nothing

Parent folder /media/rohan
Volume ext_disk

Modified Tuesday 29 October 2024 09:11:42 PM
Created Tuesday 29 October 2024 09:11:42 PM

24.6 kB used
6.8 GB free
Total capacity 7.2 GB
Filesystem type ext3/ext4

Open in Disks

After Workload:

2. Large File Creation:

    a. **Comparison of File Systems**:
- Ext4 optimizes large file creation better than ZFS, as demonstrated by our workload.

b. **Workload Details**:

```
1 fsd=fsd1,anchor=$anchor,depth=0,width=1,files=3,size=800m
2 fwd=fwd1,fsd=fsd1,operation=create,fileio=sequential,fileselect=random,threads=2
3 rd=rd1,fwd=fwd1,fwdrate=max,format=yes,elapsed=30,interval=1
```

- We created a specific workload for testing large file creation (referred to as workload 2).
- The workload involves creating 3 files, each 800MB in size, within a single folder.
- The operation used for this test is "create," focusing specifically on file creation.

c. **Execution on File Systems**:
- This workload is executed on the ZFS file system, with the anchor set to the directory pointing to the ZFS pool.

```
rohan@rohan-virtual-machine:~/Desktop/vdbench$ sudo ./vdbench -f workload1 anchor=/zfs_pool
[sudo] password for rohan:


Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Vdbench distribution: vdbench50407 Tue June 05  9:49:29 MDT 2018
For documentation, see 'vdbench.pdf'.


23:38:55.943 input argument scanned: '-fworkload1'
23:38:55.951 input argument scanned: 'anchor=/zfs_pool'
23:38:56.198 Anchor size: anchor=/zfs_pool: dirs:          12; files:          360; bytes:    360.000m (377,487,360)
23:38:56.468 Starting slave: /home/rohan/Desktop/vdbench/vdbench SlaveJvm -m localhost -n localhost-10-241029-23.38.55.685 -l localhos
t-0 -p 5570
```

- The same workload is also run on the ext4 file system, with the anchor set to the directory pointing to the ext4 drive.

```
rohan@rohan-virtual-machine:~/Desktop/vdbench$ sudo ./vdbench -f workload1 anchor=/media/rohan/ext_disk


Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Vdbench distribution: vdbench50407 Tue June 05  9:49:29 MDT 2018
For documentation, see 'vdbench.pdf'.
```

a. We found the following results:

i.**ext4**:

1. Finishes creating files in just **9 seconds**! Remember, these are **800MB** files!

2. The average write rate is **266.67 MB/s**!

ext_disk Properties

Basic   Permissions

Name ext_disk

Type Folder

Contents 6 items, totalling 2.5 GB
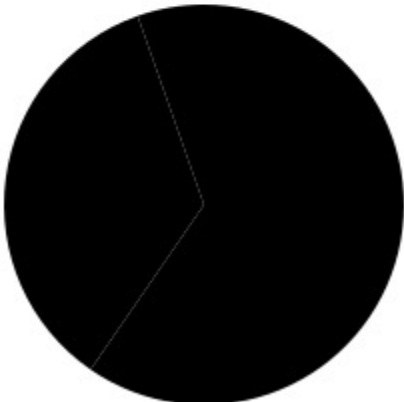
Parent folder /media/rohan

Volume ext_disk

Modified Tuesday 29 October 2024 09:11:42 PM

Created Tuesday 29 October 2024 09:11:42 PM

2.5 GB used

4.3 GB free

Total capacity 7.2 GB

Filesystem type ext3/ext4

Open in Disks

ii.   **ZFS**:

Before Workload2


```
rohan@rohan-virtual-machine:~/Desktop/vdbench$ zpool list
NAME        SIZE   ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG   CAP  DEDUP   HEALTH  ALTROOT
zfs_pool     7G    181M   6.82G      -          -      0%    2%  1.99x   ONLINE  -
rohan@rohan-virtual-machine:~/Desktop/vdbench$ sudo ./vdbench -f workload2 anchor=/zfs_pool


Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Vdbench distribution: vdbench50407 Tue June 05  9:49:29 MDT 2018
For documentation, see 'vdbench.pdf'.

00:03:49.703 input argument scanned: '-fworkload2'
00:03:49.715 input argument scanned: 'anchor=/zfs_pool'
00:03:49.974 Anchor size: anchor=/zfs_pool: dirs:         0; files:        3; bytes:     2.344g (2,516,582,400)
00:03:50.305 Starting slave: /home/rohan/Desktop/vdbench/vdbench SlaveJvm -m localhost -n localhost-10-241030-00.03.49.35
localhost 0  n 5570
```

After Workload2

1. Takes a whole **26seconds** to create the files.
2. The average write rate is **92.31MB/s.**

Disadvantages of Deduplication:

1. **Performance:**
   - In the second workload, the ZFS system took 26 seconds to set up the file system, while ext4 took only 9 seconds.
   - ZFS had an average write speed of 92.31 MB/s, compared to ext4's average write speed of 266.67 MB/s.
   - Overall, deduplication negatively impacts the performance of a file system due to added overhead.
2. **CPU Utilization:**
   - During the file structure setup in the first workload, ZFS had an average CPU utilization of 80.7%, whereas ext4 had a significantly lower utilization of 49.8%.
   - In the second workload, ZFS CPU utilization was 84.6%, while ext4's was 69.9%.
   - The deduplication feature leads to higher CPU usage in both workloads, reflecting its overhead.

   Note: Overhead described at the beginning of report when describing deduplication.

Disadvantages of Optimizing Large File Creation:

1. **Greater Metadata Overhead for Small Files:**
   - In workload 1, only 360 MB was required for the files, but the additional space used after running workload 1 was 377.6 MB, resulting in 17.6 MB of overhead.
   - ZFS had very little overhead, but more space is utilized for maintaining extent trees compared to the actual data when dealing with small files.
2. **No Possible Recovery from Corruption:**
   - Ext4 optimizes large file creation through delayed and contiguous allocation, as well as extents.

- This optimization limits the possibility of data correction mechanisms since very little metadata is stored for large files in contiguous blocks.
- If checksums were maintained, it raises the question of why not also maintain individual block pointers, as they occupy a similar amount of space.