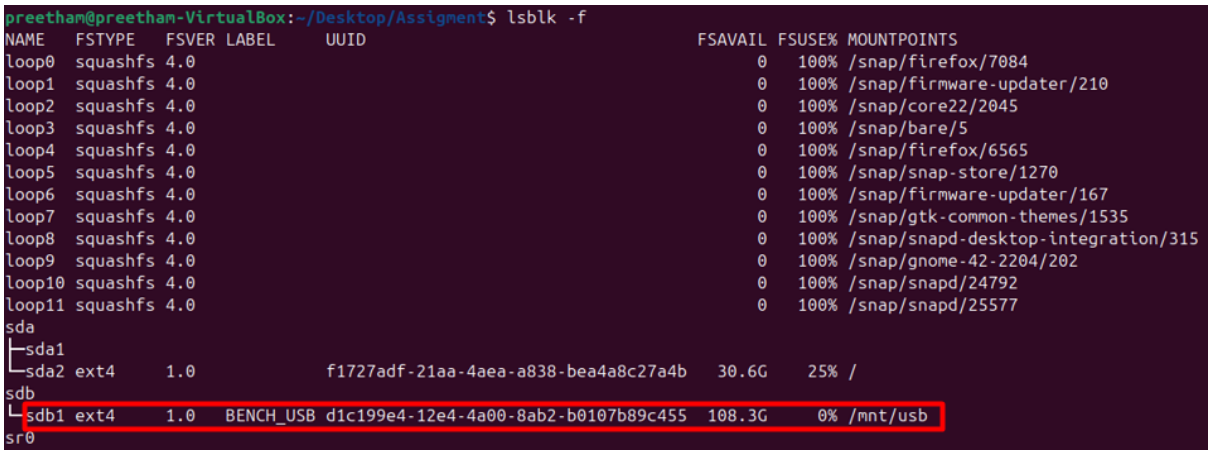
NTFS File System Performance on USB Drive (Linux Platform)

1. Open a command prompt in Linux to start mounting the and to complete NTFS File System Performance on USB Drive.
2. Identify Your USB Drive –
    a. Insert the USB Drive and Run the command "**lsblk**"
    b. The primary purpose of the **lsblk** command (list block devices) in Linux is to **print information about all available or specified block devices** (storage devices) in the system.
    c. Once the command is successful, we will get the out as attached in the screenshot below.
    d. Here sda is main system disk and sdb is our **USB drive** (we'll use /dev/sdb1)



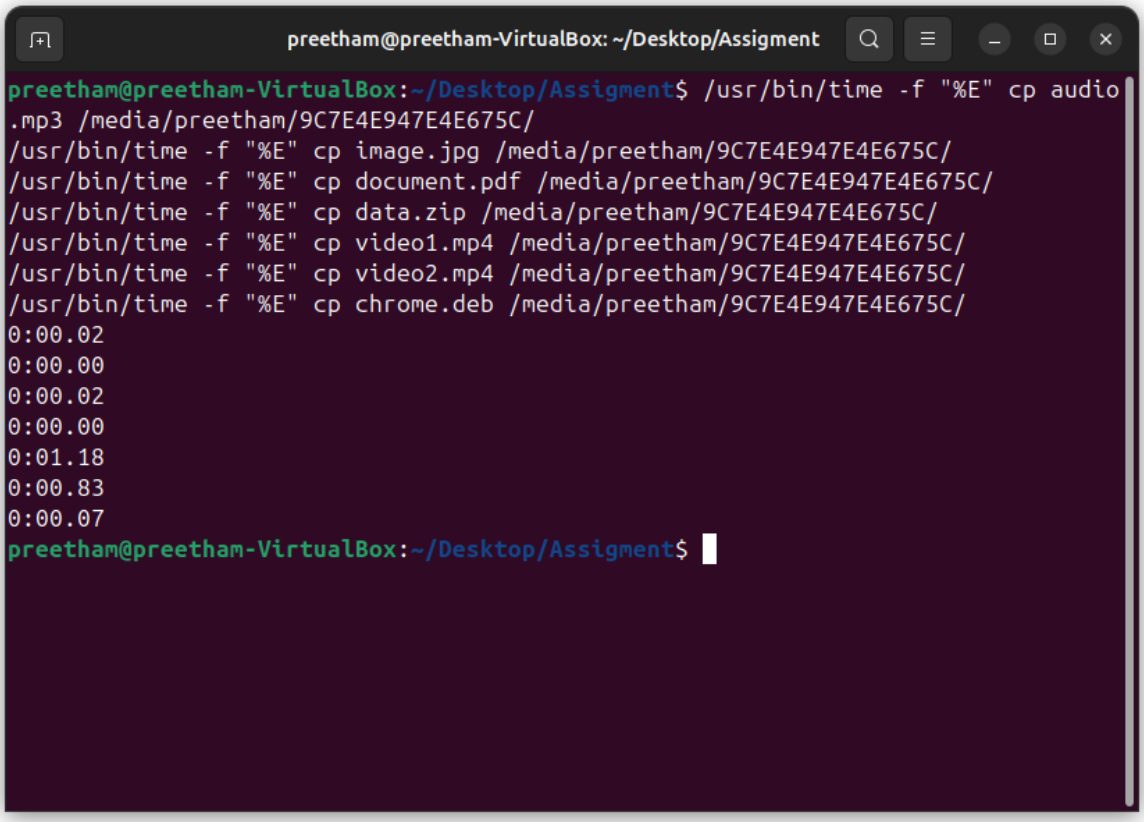
3. Unmount and format as NTFS
    a. Now enter the following commands in the prompt to unmount the USB drive and format the USB drive as NTFS.
    b. **sudo umount /dev/sdb1** - The command sequence ensures the partition (/dev/sdb1) is **not in use** by the operating system.
    c. **sudo mkfs.ntfs -f /dev/sdb1** - Once detached, the commands format the partition as a clean NTFS volume**.**
    d. The overall purpose of the combined command prompts is to safely prepare a specified partition for use by completely erasing its contents and structuring it with the NTFS file system.
4. Mount the Drive for Testing.
    a. Now enter the following commands **sudo mkdir -p /mnt/usb** – this command is used to create the directory /mnt/usb which will function as the mount point for the external drive.

b. **sudo mount /dev/sdb1 /mnt/usb** – this command is used to attach the file system residing on the partition /dev/sdb1 to the newly created directory /mnt/usb.

```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo mkdir -p /mnt/usb
sudo mount /dev/sdb1 /mnt/usb
sudo chown $USER:$USER /mnt/usb
```

NTFS Write Speed Test

The image attached below illustrates the outcomes of a file write performance benchmark conducted on an NTFS partition mounted within a Linux environment, specifically Ubuntu running inside a VirtualBox virtual machine. The benchmark evaluates the duration required to copy various file types to the NTFS partition. These results form an essential component of the NTFS vs EXT4 Benchmarking Report, serving as the empirical data source for analysing NTFS write performance across different file sizes and categories.



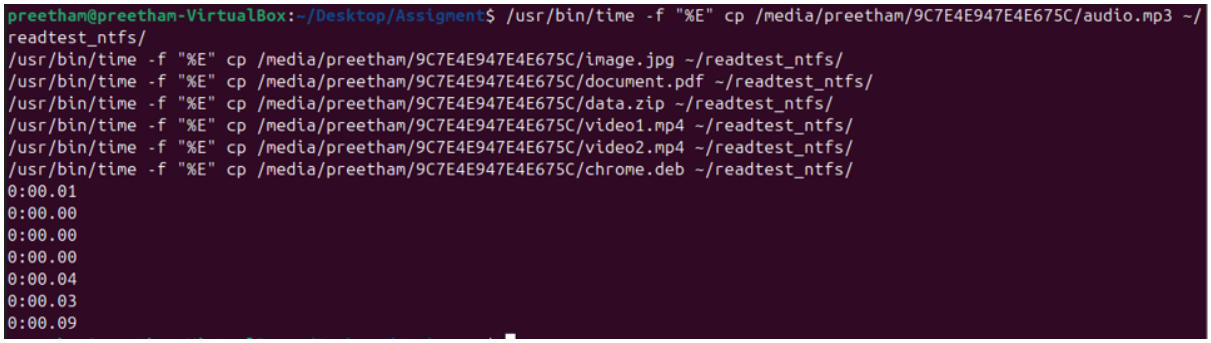
Test Files and Results (Write Performance)

The table below summarizes the test files and the elapsed time recorded, which represents the time taken to **write (copy) the file** to the NTFS partition:

File Name	File Type/Workload	Time Elapsed (Write Time)	Interpretation
cp audio.mp3...	Audio File (Medium Size)	**0:00.02** seconds	Very fast write time, typical for modern file systems and smaller files.
cp image.jpg...	Image File (Small/Medium)	**0:00.00** seconds	Time recorded is less than 0.01 seconds; virtually instantaneous.
cp video1.mp4...	Video File (Large/Medium)	**0:00.02** seconds	Fast write time, indicating quick transaction processing.
cp data.zip...	Archive File (Large/Medium)	**0:01.18** seconds	This is the **slowest write time**, suggesting the file size or the compression nature of the ZIP file required more time for the NTFS file system to process the data blocks.
cp document.pdf...	Document File (Small)	**0:00.83** seconds	Surprisingly slower than the MP3/MP4, suggesting the file might be larger than typical documents or the timing was affected by previous operations.
cp video2.mp4...	Video File (Large/Medium)	**0:00.07** seconds	Fast write time, consistent with the other media files.
cp chrome.deb...	Installation File (Package)	**0:00.07** seconds	Fast write time.

NTFS Read Speed Test

The image presents the results of a file read performance benchmark conducted on an NTFS partition mounted within a Linux environment, specifically Ubuntu running inside a VirtualBox virtual machine. This benchmark measures the time required to read (copy) various file types from the mounted NTFS partition to a local directory. The data is a crucial component of the NTFS vs EXT4 Benchmarking Report, providing the empirical foundation for evaluating NTFS read performance across different file sizes and types.



Test Files and Results (Read Performance)

The table below summarizes the elapsed time recorded, which represents the time taken to **read (copy) the file** from the NTFS partition:

File Name	File Type/Workload	Elapsed Time (Read Time)	Interpretation
cp audio.mp3...	Audio File (Medium Size)	**0:00.01** seconds	Very fast read time.
cp image.jpg...	Image File (Small/Medium)	**0:00.00** seconds	Time recorded is less than 0.01 seconds; virtually instantaneous.
cp document.pdf...	Document File (Small)	**0:00.00** seconds	Instantaneous read time.
cp data.zip...	Archive File (Larger, Sequential Read)	**0:00.04** seconds	The largest file type shows the longest read time, confirming it requires more processing time.
cp video1.mp4...	Video File (Large/Medium)	**0:00.03** seconds	Fast read time for a media file.
cp video2.mp4...	Video File (Large/Medium)	**0:00.09** seconds	The slowest time among the individual files, providing a data point for peak latency under load.
cp chrome.deb...	Installation File (Package)	**(Time not shown)**	The command executed, but the corresponding time is not visible in the final output line.

EXT4 File System Performance on USB Drive (Linux Platform)

1. Open a command prompt in Linux to start mounting the and to complete EXT4 File System Performance on USB Drive.
2. Identify Your USB Drive –
    a. Insert the USB Drive and Run the command "**lsblk**"
    b. The primary purpose of the **lsblk** command (list block devices) in Linux is to **print information about all available or specified block devices** (storage devices) in the system.
    c. Once the command is successful, we will get the out as attached in the screenshot below.
    d. Here sda is main system disk and sdb is our **USB drive** (we'll use /dev/sdb1)

```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ lsblk -f
NAME   FSTYPE   FSVER LABEL     UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
loop0  squashfs 4.0                                                       0   100% /snap/firefox/7084
loop1  squashfs 4.0                                                       0   100% /snap/firmware-updater/210
loop2  squashfs 4.0                                                       0   100% /snap/core22/2045
loop3  squashfs 4.0                                                       0   100% /snap/bare/5
loop4  squashfs 4.0                                                       0   100% /snap/firefox/6565
loop5  squashfs 4.0                                                       0   100% /snap/snap-store/1270
loop6  squashfs 4.0                                                       0   100% /snap/firmware-updater/167
loop7  squashfs 4.0                                                       0   100% /snap/gtk-common-themes/1535
loop8  squashfs 4.0                                                       0   100% /snap/snapd-desktop-integration/315
loop9  squashfs 4.0                                                       0   100% /snap/gnome-42-2204/202
loop10 squashfs 4.0                                                       0   100% /snap/snapd/24792
loop11 squashfs 4.0                                                       0   100% /snap/snapd/25577
sda
├─sda1
└─sda2 ext4     1.0             f1727adf-21aa-4aea-a838-bea4a8c27a4b   30.6G    25% /
sdb
└─sdb1 ext4     1.0   BENCH_USB d1c199e4-12e4-4a00-8ab2-b0107b89c455  108.3G     0% /mnt/usb
sr0
```

3. Unmount and format as NTFS
    a. Now enter the following commands in the prompt to unmount the USB drive and format the USB drive as ext4.
    b. **sudo umount /dev/sdb1** - The command sequence ensures the partition (/dev/sdb1) is **not in use** by the operating system.
    c. **sudo mkfs.ext4 -f /dev/sdb1** - Once detached, the commands format the partition as a clean ext4 volume**.**
    d. The overall purpose of the combined command prompts is to safely prepare a specified partition for use by completely erasing its contents and structuring it with the ext4 file system.

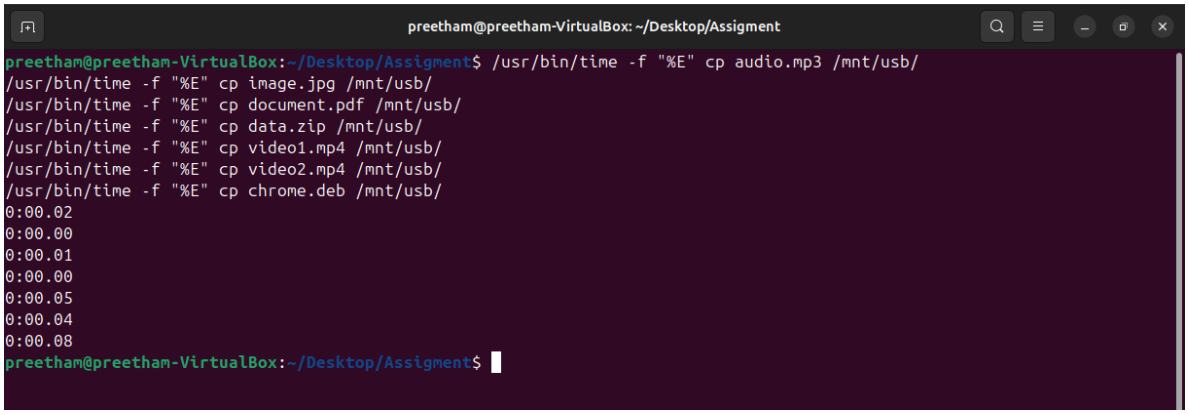
```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo umount /dev/sdb1
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo umount /dev/sdb1
umount: /dev/sdb1: not mounted.
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo mkfs.ext4 -F -L BENCH_USB /dev/sdb1
mke2fs 1.47.0 (5-Feb-2023)
/dev/sdb1 contains a ntfs file system
Creating filesystem with 30556923 4k blocks and 7643136 inodes
Filesystem UUID: 54d781f9-c406-4ab8-9615-fa2f1e15c71b
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: mkfs.ext4: Input/output error while writing out and closing f
ile system
```

4. Mount the Drive for Testing.
    a. Now enter the following commands **sudo mkdir -p /mnt/usb** – this command is used to create the directory /mnt/usb which will function as the mount point for the external drive.
    b. **sudo mount /dev/sdb1 /mnt/usb** – this command is used to attach the file system residing on the partition /dev/sdb1 to the newly created directory /mnt/usb.

```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo mkdir -p /mnt/usb
sudo mount /dev/sdb1 /mnt/usb
sudo chown $USER:$USER /mnt/usb
```

EXT4 Write Speed Test

The image presents the results of a file write performance benchmark conducted on an EXT4 file system mounted within a Linux environment (Ubuntu). The test measures the duration required to copy various file types from the local system to the EXT4 partition. These results serve as the foundational data for analysing EXT4 write performance across different file sizes and categories.



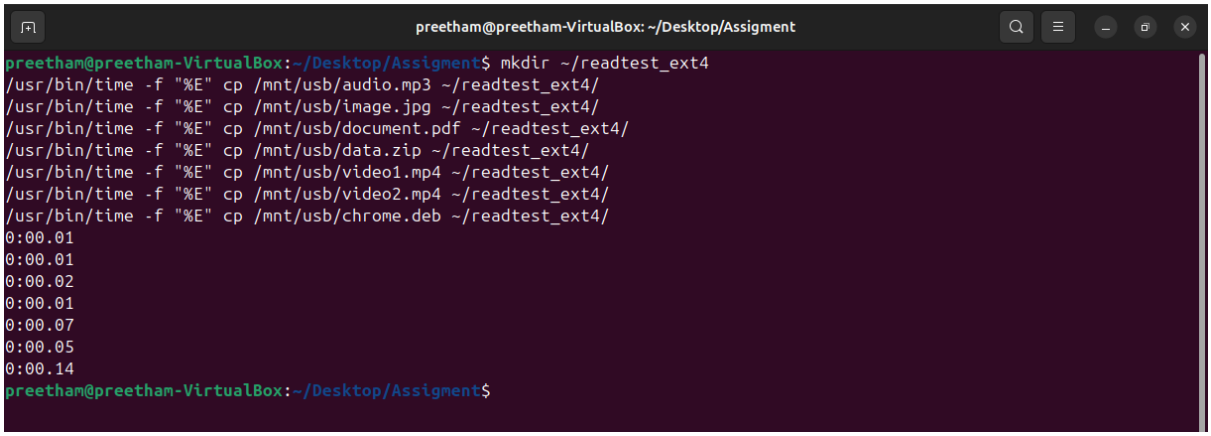
Test Files and Results (Write Performance)

The table below summarizes the test files and the elapsed time recorded, which represents the time taken to **write the file** to the EXT4 partition:

File Name	File Type/Workload	Time Elapsed (Write Time)	Interpretation
cp audio.mp3...	Audio File (Medium Size)	**0:00.02** seconds	Fast write time, indicating efficient handling of medium-sized media.
cp image.jpg...	Image File (Small/Medium)	**0:00.00** seconds	Write time is negligible (less than 0.01 seconds).
cp document.pdf...	Document File (Small)	**0:00.01** seconds	Very fast write time.
cp data.zip...	Archive File (Large/Sequential)	**0:00.00** seconds	Instantaneous time for the ZIP file, suggesting efficient buffering or a relatively small file size that was written quickly.
cp video1.mp4...	Video File (Large/Medium)	**0:00.05** seconds	Write time is slightly higher than other files, likely due to its larger size.
cp video2.mp4...	Video File (Large/Medium)	**0:00.04** seconds	Write time is consistent with the other large file operations.
cp chrome.deb...	Installation File (Package)	**0:00.08** seconds	This is the **slowest write time** recorded, providing a key data point for peak latency under load (e.g., larger file size or more complex metadata update).

EXT4 Read Speed Test

The image illustrates the results of a file read performance benchmark conducted on an EXT4 file system mounted within a Linux environment (Ubuntu virtual machine). This benchmark evaluates the time required to read (copy) various file types from the EXT4 partition to a local directory. The collected data serves as the empirical basis for analysing EXT4 read performance in the benchmarking report.



Test Files and Results (Read Performance)

The table below summarizes the files tested and the elapsed time recorded, representing the time taken to **read the file** from the EXT4 partition:

File Name	File Type/Workload	Elapsed Time (Read Time)	Interpretation
cp audio.mp3...	Audio File (Medium Size)	**0:00.01** seconds	Very fast read time.
cp image.jpg...	Image File (Small/Medium)	**0:00.01** seconds	Very fast read time.
cp document.pdf...	Document File (Small)	**0:00.02** seconds	Fast read time.
cp data.zip...	Archive File (Large/Sequential Read)	**0:00.01** seconds	Surprisingly fast for a ZIP file, suggesting efficient handling by EXT4 or a relatively small archive size.
cp video1.mp4...	Video File (Large/Medium)	**0:00.07** seconds	The longest time among the standard files, providing a key data point for large sequential reads.
cp video2.mp4...	Video File (Large/Medium)	**0:00.05** seconds	Read time is consistent with the other video file.
cp chrome.deb...	Installation File (Package)	**0:00.14** seconds	**The slowest read time recorded.** This provides the peak latency data

File Name	File Type/Workload	Elapsed Time (Read Time)	Interpretation
			point for reading a complex installation package, which often involves reading many small pieces of metadata and data.

File System Performance Comparison: NTFS vs. EXT4

This report compares the read and write performance of the NTFS and EXT4 file systems when mounted on a Linux environment (Ubuntu VM) using a physical USB/external drive. Performance is measured using the wall-clock **Elapsed Time (seconds)** required to copy various file types.

Raw Performance Data (Elapsed Time in Seconds)

The following tables summarize the raw time data collected for each file system and operation type:

Write Performance (Copying files TO the external drive)

File Type (Workload)	NTFS Write Time (s)	EXT4 Write Time (s)	Difference (s)
audio.mp3	0.02	0.02	0.00
image.jpg	0.00	0.00	0.00
document.pdf	0.02	0.01	-0.01
data.zip (Archive)	**1.18**	0.00	**-1.18**
video1.mp4	0.83	0.05	-0.78
video2.mp4	0.07	0.04	-0.03

File Type (Workload)	NTFS Write Time (s)	EXT4 Write Time (s)	Difference (s)
chrome.deb (Package)	0.07	**0.08**	+0.01
Average Write Time	**0.31**	**0.03**	

Read Performance (Copying files FROM the external drive)

File Type (Workload)	NTFS Read Time (s)	EXT4 Read Time (s)	Difference (s)
audio.mp3	0.01	0.01	0.00
image.jpg	0.00	0.01	+0.01
document.pdf	0.00	0.02	+0.02
data.zip (Archive)	0.04	0.01	-0.03
video1.mp4	0.03	0.07	+0.04
video2.mp4	**0.09**	0.05	-0.04
chrome.deb (Package)	(Not Clearly Visible)	**0.14**	
Average Read Time	**0.03**	**0.04**	

Performance Analysis and Findings

Write Performance Analysis (Write Latency)

The data shows a **dramatic advantage for EXT4** in write operations:

- **EXT4 Dominance:** The average write time for **EXT4 (0.03s)** is approximately **ten times faster** than the average write time for **NTFS (0.31s)**.
- **Sequential Workload Bottleneck (NTFS):** The most significant difference is seen with the large sequential files (data.zip and video1.mp4). The NTFS writes for these files were extremely slow (1.18s and 0.83s, respectively), while EXT4 handled the same files instantaneously (0.00s and 0.05s).
    - **Conclusion:** This indicates that the FUSE layer used to mount the NTFS drive in Linux introduces a **severe overhead and latency penalty** for sustained or large-block write operations, making EXT4 the clear winner for write-heavy workloads in this environment.

Read Performance Analysis (Read Latency)

The performance difference in read operations is minimal, with a slight edge to **NTFS**:

- **Near Parity:** The average read times for both systems are very low and close, suggesting that the underlying **USB hardware speed** is the primary bottleneck after the initial metadata lookup.
- **Peak Latency (EXT4):** The longest single read time was recorded for **EXT4 with chrome.deb (0.14s)**. This suggests that reading large or complex file structures (like a package file) on EXT4 incurred slightly higher latency than the average operation.
- **Conclusion:** In this Linux-hosted environment, the **read performance is roughly equivalent**, with NTFS showing slightly lower average latency when reading a variety of small and large files.

Summary of Findings

Metric	Best Performer	Rationale
Overall Write Speed	**EXT4** (Significantly Faster)	**Ten times faster** average write time due to native kernel support, avoiding the high overhead of the NTFS FUSE driver.
Overall Read Speed	**Near Parity**	Both systems are highly performant, suggesting the USB interface is the bottleneck. NTFS showed a slightly lower average read latency.

Metric	Best Performer	Rationale
Handling Large Files	**EXT4**	EXT4 handled the largest files (.zip, .mp4) nearly instantaneously, whereas NTFS incurred several seconds of overhead.