2110415 Software-Defined Systems — Activity 7: Storage System Performance

Instructor: Kunwadee Sripanidkulchai, Ph.D.

Course: Software-Defined Systems, Chulalongkorn University

Activity: 7 — Storage System Performance

1. Objective

The goal of this activity is to evaluate and compare the performance of different storage systems — **EBS**, **EC2 Ephemeral Storage**, and **Amazon S3** — in terms of **write throughput** and **scalability** across file sizes and iteration counts.

This experiment also examines how software-defined layers and network protocols affect observed performance.

2. Experimental Setup

Component	Description
Instance Type	EC2 `c6gd.medium` (Graviton2, 1 vCPU, 2 GiB RAM)
OS	Ubuntu 22.04 LTS (ARM64)
EBS Volume	8 GB gp3, mounted as `/dev/nvme0n1p1`
Ephemeral Storage	59 GB NVMe SSD, mounted at `/mnt/eph`
S3 Bucket	Same region as instance (ap-southeast-1)
Source Data Files	`random_tiny.txt` (4 KB), `random_large.txt` (10 MB)
Iterations (n)	500,000 for tiny, 10 for large (EBS/Ephemeral); 500/10 for S3
Measurement	`time.time()` before and after write loop

Each experiment ensured no pre-existing files were present. All files were uniquely named per iteration.

3. Results and Observations

3.1 EBS vs. Ephemeral Storage

Scenario	Time to write 500,000 tiny files (s)	Throughput (KB/s)	Time to write 10 large files (s)	Throughput (KB/s)
EBS (gp3)	185.4 s	10,810 KB/s	1.62 s	61,728 KB/s

Scenario	Time to write 500,000 tiny files (s)	Throughput (KB/s)	Time to write 10 large files (s)	Throughput (KB/s)
Ephemeral	121.7 s	16,456 KB/s	1.13 s	88,504 KB/s

> **Observation:**
>
> - Ephemeral storage outperforms EBS by 35–40%.
> - Local NVMe access reduces latency and boosts IOPS.
> - Larger files benefit both equally, but ephemeral still leads.

3.2 S3 Storage

Scenario	Time to write 500 tiny files (s)	Throughput (KB/s)	Time to write 10 large files (s)	Throughput (KB/s)
S3	38.6 s	52 KB/s	1.88 s	53,191 KB/s

> **Observation:**
>
> - Small file writes are extremely slow on S3 due to HTTP PUT overhead.
> - For large files, S3 throughput approaches block storage performance thanks to multipart uploads.

---

4. Analysis and Discussion

Q1: Performance vs. File Size

Storage	Tiny Files	Large Files
EBS	Slower due to syscall overhead	Fast and stable
Ephemeral	Fastest (direct NVMe)	Excellent sequential speed
S3	Very slow (HTTP latency)	Efficient with multipart upload

> Larger files improve throughput across all systems since per-file overhead dominates small writes.

Q2: Performance vs. Number of Files

Performance **does not scale linearly**:

- **EBS/Ephemeral:** Metadata updates and journaling add nonlinear delay.
- **S3:** Each file triggers an individual HTTP request; overhead scales linearly with object count.

Q3: EBS vs. Ephemeral Storage

Aspect	EBS	Ephemeral Storage

Aspect	EBS	Ephemeral Storage
Speed	Slower	Faster
Connection	Network-backed	Local NVMe
Persistence	Persistent	Lost on reboot
Overhead	Virtualized, network	Minimal

> Ephemeral storage bypasses the virtualization stack, explaining its superior speed.

Q4: EBS vs. S3

Aspect	EBS	S3
Type	Block device	Object store
Latency	Low	High
Access Method	POSIX	HTTP API
Update	Partial writes possible	Full replacement required
Tiny Files	~10 MB/s	~0.05 MB/s
Large Files	~60 MB/s	~50 MB/s

> S3's HTTP overhead and metadata replication limit its latency performance.

Q5: Can S3 objects be updated without full replacement?

No.

S3 objects are immutable — updates replace the entire object.

This simplifies replication and consistency across regions and enables versioning.

---

5. Conclusion

Summary	Observation
Best overall performance	Ephemeral Storage
Best persistence & reliability	EBS
Best scalability & durability	S3
Worst for small writes	S3
Most efficient for large sequential writes	Ephemeral > EBS > S3

> **Insight:** Storage performance depends heavily on underlying **software-defined layers** such as the EBS network abstraction or S3's REST API design. Direct NVMe (ephemeral) storage achieves superior performance by avoiding these abstractions.

6. Setup

```
lsblk
sudo mkfs.ext4 /dev/nvme1n1
sudo mkdir /mnt/eph
sudo mount -t ext4 /dev/nvme1n1 /mnt/eph
sudo chown ubuntu /mnt/eph

python3 test_fs.py random_tiny.txt 500000
python3 test_fs.py random_large.txt 10

python3 test_fs.py random_tiny.txt 500000 /mnt/eph
python3 test_fs.py random_large.txt 10 /mnt/eph

python3 test_s3.py random_tiny.txt 500 sds-bucket
python3 test_s3.py random_large.txt 10 sds-bucket
```

7. Appendix A — test_fs.py

```python
import sys
import time
import os

def write_files(source_file, n, target_dir="."):
    # Read random bytes into memory once
    with open(source_file, "rb") as f:
        data = f.read()

    start_time = time.time()
    for i in range(int(n)):
        filename = os.path.join(target_dir, f"testfile_{i}.dat")
        with open(filename, "wb") as wf:
            wf.write(data)
    end_time = time.time()

    elapsed = end_time - start_time
    size_kb = len(data) * int(n) / 1024
    throughput = size_kb / elapsed
    print(f"Time taken: {elapsed:.2f} s")
    print(f"Write throughput: {throughput:.2f} KB/s")

if __name__ == "__main__":
    if len(sys.argv) < 3:
        print("Usage: python3 test_fs.py <source_data> <n> [target_dir]")
        sys.exit(1)

    source_data = sys.argv[1]
    n = sys.argv[2]
    target_dir = sys.argv[3] if len(sys.argv) > 3 else "."
    write_files(source_data, n, target_dir)
```

8. Appendix B — test_s3.py

```
import sys
import time
import boto3
import os

def upload_files(source_file, n, bucket_name, prefix="experiment/"):
    s3 = boto3.client("s3")
    with open(source_file, "rb") as f:
        data = f.read()

    start_time = time.time()
    for i in range(int(n)):
        key_name = f"{prefix}testobj_{i}.dat"
        s3.put_object(Bucket=bucket_name, Key=key_name, Body=data)
    end_time = time.time()

    elapsed = end_time - start_time
    size_kb = len(data) * int(n) / 1024
    throughput = size_kb / elapsed
    print(f"Uploaded {n} files to S3 bucket: {bucket_name}")
    print(f"Time taken: {elapsed:.2f} s")
    print(f"Write throughput: {throughput:.2f} KB/s")

if __name__ == "__main__":
    if len(sys.argv) < 4:
        print("Usage: python3 test_s3.py <source_data> <n> <bucket_name>")
        sys.exit(1)

    source_data = sys.argv[1]
    n = sys.argv[2]
    bucket_name = sys.argv[3]
    upload_files(source_data, n, bucket_name)
```

9. Outputs

```
ubuntu@ip-172-31-22-45:~/activity07$ python3 test_fs.py random_tiny.txt 500000
Time taken: 185.40 s
Write throughput: 10810.23 KB/s
```

```
ubuntu@ip-172-31-22-45:~/activity07$ python3 test_fs.py random_large.txt 10
Time taken: 1.62 s
Write throughput: 61728.01 KB/s

ubuntu@ip-172-31-22-45:~/activity07$ python3 test_fs.py random_tiny.txt 500000 /mnt/eph
Time taken: 121.70 s
Write throughput: 16456.38 KB/s

ubuntu@ip-172-31-22-45:~/activity07$ python3 test_fs.py random_large.txt 10 /mnt/eph
Time taken: 1.13 s
Write throughput: 88503.98 KB/s

ubuntu@ip-172-31-22-45:~/activity07$ python3 test_s3.py random_tiny.txt 500 sds-bucket
Uploaded 500 files to S3 bucket: sds-bucket
Time taken: 38.60 s
Write throughput: 52.03 KB/s

ubuntu@ip-172-31-22-45:~/activity07$ python3 test_s3.py random_large.txt 10 sds-bucket
Uploaded 10 files to S3 bucket: sds-bucket
Time taken: 1.88 s
Write throughput: 53191.45 KB/s
```