Flash Storage Engineering & Latency Analysis Report

System Environment: ODROID (ARMv7 8-Core) | Target Media: SanDisk Ultra 128GB MicroSD | OS: Ubuntu 20.04 (Kernel 5.4.274)

Designed & Executed By: U.S.A._N.C._Tony_Casanova_2026 **AI Technical Assistant:** Google Gemini

1. Executive Summary & Problem Diagnosis

System performance testing under synchronous I/O workload conditions reveals a critical storage bottleneck. While average `fsync` latency remains low (~3.5 ms – 5.5 ms), the storage subsystem suffers from severe **tail latency spikes reaching up to 3.30 seconds (3,296.40 ms)**.

Root Cause Mechanics

The primary drive (`SanDisk SC128`, manufactured Oct 2017) lacks a dedicated DRAM cache and internal multi-channel flash controller. When `sys_fsync()` forces page-cache writes to physical NAND, the drive is forced into an inline **Read-Modify-Write cycle** and block erasure state, freezing kernel I/O queues for over 3 seconds.

2. Benchmark Execution Matrix (FIO 3.16)

Data parsed directly from physical kernel synchronous completion metrics (`sync` block):

Benchmark Run	IOPS	Mean Sync Latency	Max Sync Latency (Tail Spike)	Physical Cause
`fio_results_run_1.log`	179 IOPS	5.51 ms	**2,427.60 ms (2.43s)**	Initial cache drain; 2.4s NAND erase lockup
`fio_results_run_2.log`	264 IOPS	3.74 ms	**3,296.40 ms (3.30s)**	Worst-case controller GC block erase stall
`fio_results_run_3.log`	280 IOPS	3.52 ms	**1,501.00 ms (1.50s)**	Sustained GC pressure under block exhaustion
Aggregate Averages	**241.00 IOPS**	**4.25 ms**	**3,296.40 ms Peak Spike**	**Unsuitable for transactional workloads**

3. Hardware Telemetry & Architecture Profile

Host Architecture

- **CPU:** ARM Cortex-A7 (8 cores @ 2.0 GHz)
- **Architecture:** 32-bit `armv7l`
- **RAM:** 1.9 GiB Total (1.1 GiB Cache)

Storage Device Info

- **Device Node:** `/dev/mmcblk1`
- **Identity:** `SC128` (SanDisk 128GB)
- **Controller ID:** `0x000003 / 0x5344`

Filesystem Parameters

- **Root FS:** `ext4` on `/dev/mmcblk1p2`
- **Mount Flags:** `rw, noatime`
- **Journal:** `data=ordered`
- **Stripe Size:** 32753 blocks

- **OS:** Ubuntu 20.04.6 LTS (5.4.274)

- **Date / Rev:** Oct 2017 / HW 0x8
- **Scheduler:** `mq-deadline`

4. Mitigation Engineering Plan

Step 1: Remount ext4

Modify `/etc/fstab` to use writeback mode and extend commit windows:

```
rw,noatime,data=writeback,barrier=0,commit=60
```

Step 2: Smooth Dirty Memory

Configure sysctl to prevent dirty cache spikes that overwhelm storage:

```
vm.dirty_background_ratio = 5
vm.dirty_ratio = 10
```

Step 3: Hardware Evolution

Upgrade critical databases and WAL mounts to an **ODROID eMMC Module** or USB 3.0 NVMe drive with DRAM.

5. Storage Subsystem Educational Glossary

`fsync / sys_fsync()`
A POSIX system call that forces all dirty page-cache data for a specific file to be physically written to underlying non-volatile storage media before returning control to the application.

`clat (Completion Latency)`
In FIO metrics, completion latency measures the duration from when an I/O request is submitted to the OS kernel until the storage subsystem acknowledges completion back to the system.

`sync Latency`
The specific latency category reported by FIO during `--fsync=1` runs, measuring the exact time required to issue physical flush commands to non-volatile flash hardware.

`Tail Latency`
The extreme high-percentile latency values (e.g., p99, p99.99, max) experienced by a small subset of requests, representing worst-case response times during system stalls.

`Garbage Collection (GC)`
An internal flash controller process that reclaims fragmented, dirty flash blocks by moving valid data pages to clean blocks and erasing empty physical blocks for reuse.

`Read-Modify-Write (RMW)`
A mandatory cycle in flash storage where modifying a small chunk of data requires reading an entire physical block into RAM, updating it, erasing the physical flash block, and rewriting the full block.

`data=writeback`
An `ext4` filesystem mode where metadata changes are journaled, but file data writes are unordered and written straight to disk, significantly reducing `fsync` lockups.