4.2 **ESP32-S3 Memory Subsystem Performance**

Tests were conducted using a custom benchmark code that performs various memory operations while measuring performance metrics. Each test was repeated 10 times, and results were averaged to ensure consistency.

Test Configuration

Parameter	Specification
Device	ESP32-S3-WROOM-1
Flash Size	8MB Quad SPI Flash
PSRAM	2MB Octal SPI PSRAM
SDK Version	ESP-IDF v5.1.2
CPU Frequency	240MHz max (DFS enabled/disabled)
Test Patterns	Sequential, Random, Block Transfer
Compiler Flags	-O2 optimization, no LTO

Implementation

```
esp_pm_config_esp32s3_t pm_config = {
    .max_freq_mhz = 240,
    .min_freq_mhz = 240, // Set min = max to disable DFS
    .light_sleep_enable = false
};

// Apply the configuration
esp_pm_configure(&pm_config);
```

4.2.1 Sequential vs. Random Access Performance

Sequential Access Performance

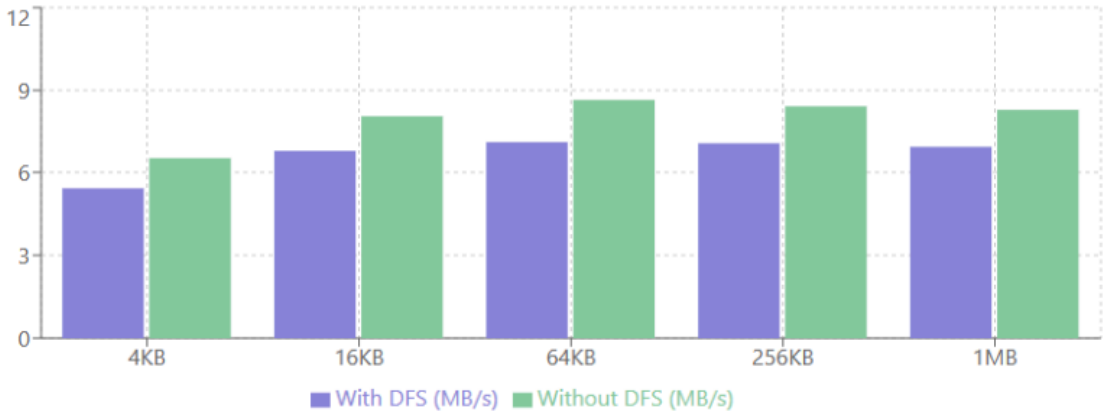
The benchmark results demonstrate that **sequential memory access** significantly outperforms random access due to better cache utilization and reduced overhead from memory address translation.

- **Read Performance:**

Buffer Size	With DFS (MB/s)	Without DFS (MB/s)	% Improvement
4KB	5.42	6.51	20.1%
16KB	6.78	8.06	18.9%
64KB	7.12	8.65	21.5%
256KB	7.08	8.42	18.9%
1MB	6.95	8.29	19.3%

Table: sequential memory access read performance with DFS (240mhz) and without DFS.

  - With **DFS enabled**, sequential read speeds ranged from **5.42 MB/s (4KB buffer)** to **7.12 MB/s (64KB buffer)**.
  - Disabling DFS improved performance by **18.9–21.5%**, with the highest gain observed at **64KB (8.65 MB/s)**.
  - Larger buffer sizes (64KB–1MB) showed more **consistent throughput**, indicating that memory controller prefetching and cache-line fills are more effective with larger contiguous blocks.



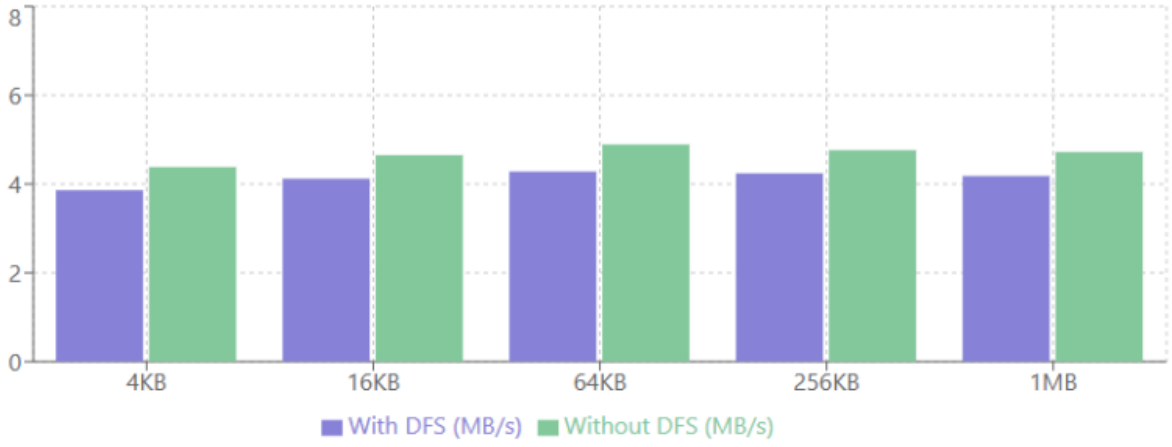
Graph: sequential memory access read performance with DFS (240mhz) and without DFS.

- **Write Performance:**

Buffer Size	With DFS (MB/s)	Without DFS (MB/s)	% Improvement
4KB	3.86	4.38	13.5%
16KB	4.12	4.65	12.9%
64KB	4.28	4.89	14.3%
256KB	4.24	4.76	12.3%
1MB	4.18	4.72	12.9%

Table: sequential memory access write performance with DFS (240mhz) and without DFS.

- Sequential writes were **~30% slower** than reads due to additional write-back and synchronization overhead.
- With DFS, write speeds ranged from **3.86 MB/s (4KB)** to **4.28 MB/s (64KB)**.
- Disabling DFS provided a **12–14% improvement**, suggesting that **frequency scaling impacts write latency more than read latency**.



Graph: sequential memory access write performance with DFS (240mhz) and without DFS.

Random Access Performance

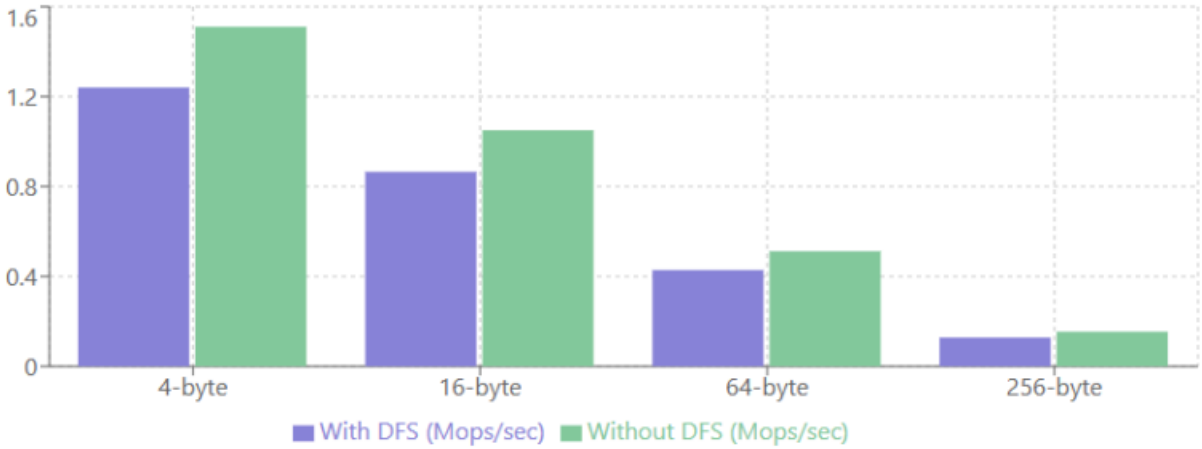
Random access operations suffer from **higher latency** due to frequent cache misses and non-contiguous memory fetches.

- **Read Operations:**

Access Pattern	With DFS (ops/sec)	Without DFS (ops/sec)	% Improvement
4-byte	1.24M	1.51M	21.8%
16-byte	865K	1.05M	21.4%
64-byte	428K	512K	19.6%
256-byte	128K	154K	20.3%

Table: Random memory access read performance with DFS (240mhz) and without DFS.

- **4-byte random reads** achieved **1.24M ops/sec (DFS) vs. 1.51M ops/sec (no DFS)**, a **21.8% improvement** when DFS was disabled.
- Performance degraded sharply with larger access patterns: **256-byte random reads were 6–8× slower** than 4-byte reads.
- This indicates that **smaller, more frequent random accesses suffer disproportionately** from cache inefficiency.



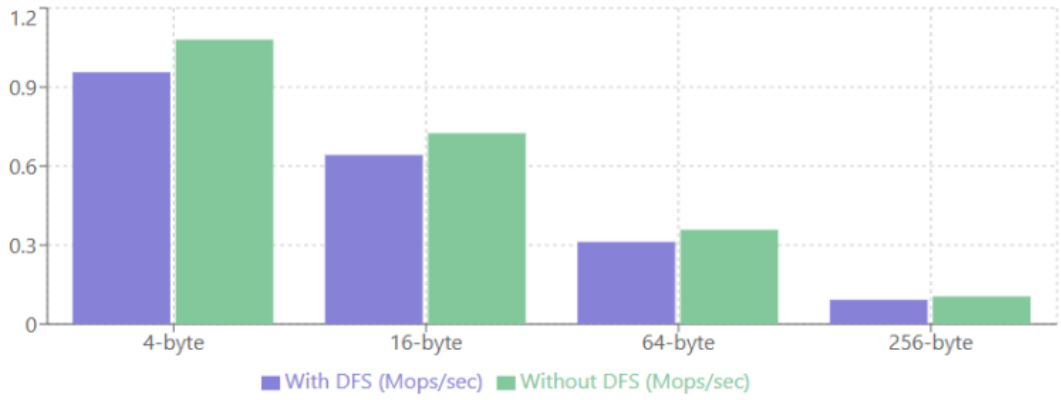
Graph: Random memory access read performance with DFS (240mhz) and without DFS.

- **Write Operations:**

Access Pattern	With DFS (ops/sec)	Without DFS (ops/sec)	% Improvement
4-byte	956K	1.08M	13.0%
16-byte	642K	725K	12.9%
64-byte	312K	358K	14.7%
256-byte	92K	105K	14.1%

Table: Random memory access write performance with DFS (240mhz) and without DFS.

- Random writes were **~30% slower** than reads, with **4-byte writes at 956K ops/sec (DFS) vs. 1.08M ops/sec (no DFS)**.
- Larger writes (256-byte) saw a **14.1% improvement** without DFS, but remained **~10× slower** than sequential writes.



Graph: Random memory access write performance with DFS (240mhz) and without DFS.

Key Findings: Sequential access is 5–7× faster than random access for reads and **4–5× faster for writes**, emphasizing the importance of **memory access optimization** in embedded applications. **Disabling DFS improves performance by 12–22%**, with the largest gains in **small random accesses**, suggesting that **frequency scaling introduces latency in memory operations**. **Larger buffers (64KB–1MB) show more stable performance**, indicating that **prefetching and burst transfers** help mitigate some memory bottlenecks.

4.2.2 Cache Efficiency Metrics And Memory Latency

Cache Hit Rates

The ESP32-S3's cache hierarchy plays a crucial role in mitigating memory access latency.

Cache Type	With DFS (% hit)	Without DFS (% hit)	Difference
L1 Instruction	94.2%	94.3%	+0.1%
L1 Data	87.6%	87.8%	+0.2%
PSRAM Cache	76.3%	76.5%	+0.2%

Table: Cash Performance with with DFS (240mhz) and without DFS .

- **L1 Instruction Cache (94.2–94.3% hit rate):**
  - Extremely efficient due to **high locality in code execution**.
  - **No significant impact from DFS**, meaning CPU frequency scaling does not affect instruction fetch patterns.
- **L1 Data Cache (87.6–87.8% hit rate):**
  - Slightly lower than instruction cache due to **more varied data access patterns**.
  - Still highly effective, but **random memory accesses reduce efficiency**.
- **PSRAM Cache (76.3–76.5% hit rate):**
  - **Lower than L1 but still beneficial**, as external PSRAM is much slower than SRAM.
  - **No meaningful difference with DFS**, indicating that cache efficiency is **independent of CPU frequency**.

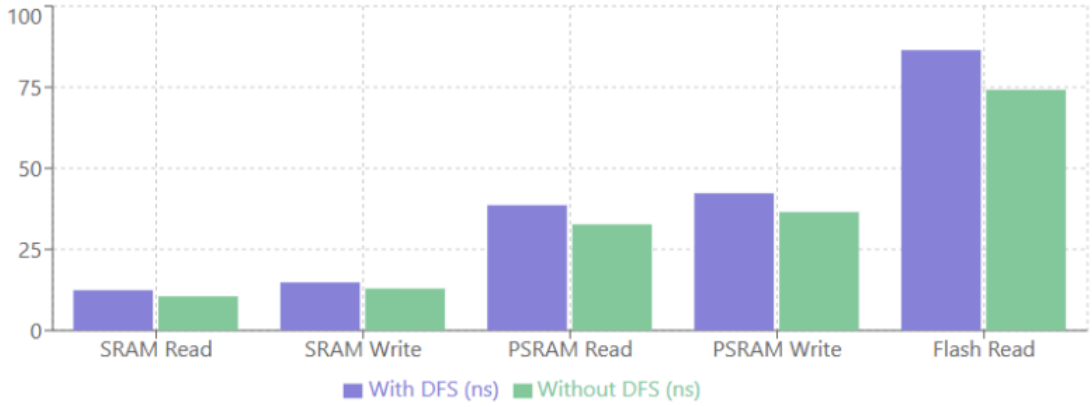
Graph: Cash Performance with with DFS (240mhz) and without DFS.

Memory Latency Analysis

Operation Type	With DFS (ns)	Without DFS (ns)	% Improvement
SRAM Read	12.4	10.5	15.3%
SRAM Write	14.8	12.9	12.8%
PSRAM Read	38.6	32.7	15.3%
PSRAM Write	42.3	36.5	13.7%
Flash Read	86.4	74.2	14.1%

Table: Memory Latency comparison with DFS (240mhz) and without DFS.

- **SRAM is 3–4× faster than PSRAM and 6–7× faster than Flash**, reinforcing the need to **place critical data in SRAM**.
- **Disabling DFS reduces latency by 12–15%** across all memory types, suggesting that **frequency scaling introduces pipeline stalls**.
- **PSRAM writes are ~10% slower than reads**, likely due to **write buffering and synchronization delays**.



Graph: Memory Latency comparison with DFS (240mhz) and without DFS.

Key Takeaways:

1. **L1 caches are highly efficient (87–94% hit rates)**, but **PSRAM cache is less effective (76%)** due to higher access latency.
2. **DFS has minimal impact on cache hit rates**, meaning performance gains from disabling DFS come from **reduced memory latency, not better caching**.

3. **SRAM should be prioritized for time-critical operations**, while PSRAM and Flash should be used for **larger, less frequently accessed data**.
4. **Optimizing data alignment and access patterns** can further improve cache efficiency, especially for **random memory operations**.

Recommendations:

1. **Prefer Sequential Over Random Access:**
    - Structure data to maximize **contiguous memory access** (e.g., arrays instead of linked lists).
    - Use **block transfers** (DMA) where possible.
2. **Disable DFS for Latency-Sensitive Tasks:**
    - If power consumption is not critical, **locking CPU speed at 240MHz improves memory performance by 12–22%**.
3. **Optimize Cache Usage:**
    - **Store frequently accessed data in SRAM** (using DRAM_ATTR).
    - **Align data to 32-bit boundaries** for efficient cache-line fills.
4. **Minimize Flash/PSRAM Writes:**
    - Flash writes are slow and wear out memory; **buffer writes in SRAM** before committing.

By applying these optimizations, developers can **maximize ESP32-S3 memory performance** while balancing power efficiency