Practical Erase Suspension for Modern Low-latency SSDs

Shine Kim†§ Jonghyun Bae† Hakbeom Jang* Wenjing Jin† Jeonghun Gong†
Seoungyeon Lee§ Tae Jun Ham† Jae W. Lee†

†Seoul National University  §Samsung Electronics  *Sungkyunkwan University
Today’s NAND flash-based SSDs in datacenters

- NAND flash-based SSDs have become a *de-facto* standard in datacenters
  - Superior throughput, low average latency, and relatively low price

PCIe Gen 3 X 8 lane NVMe SSD\(^1\)
- Seq. Read \(\rightarrow\) 6300MB/s

Low Latency SSD Controller with LL-NAND\(^2\)
- 4KB Random Read QD1 \(\rightarrow\) 15µs

3D NAND & QLC-based SSD
- \(\rightarrow\) 0.1$/GB\(^3\)

---

\(^1\) [https://www.samsung.com/semiconductor/ssd/enterprise-ssd/](https://www.samsung.com/semiconductor/ssd/enterprise-ssd/)

\(^2\) IEEE ISSCC’18, W. Cheong et al., A flash memory controller for 15us ULL-SSD using high-speed 3D NAND flash with 3us read time

\(^3\) [www.amazon.com: SAMSUNG 860QVO 1TB](https://www.amazon.com/SAMSUNG-860QVO-1TB/dp/B07QFZSWYR)
Read tail behavior of NAND flash-based SSD

- Challenge: Despite low average response time, read tail latency can be very long

Latency (μs)

99% 99.9% 99.99% 99.999% Maximum

433 4948 5932 10814 11600

AS-IS
11ms

TO-BE
Sub-200μs

Maximum read tail latency

Average: 160μs

Competitive with emerging NVM-based SSDs

Read latency distribution of a PCIe 3 X 4 NVMe low-latency SSD, 4KB, Queue Depth 16, 70% reads and 30% writes
Motivation: Two major sources of long read tail latency

- Garbage collection (GC) (e.g., 100ms → 10ms)
  - GC-induced read tail latency has been optimized by sophisticated GC schemes
- Block erase operation (e.g., 10ms/block)
  - Has become most dominant source of read tail latency

[1] Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Motivation: Two major sources of long read tail latency

- Garbage collection (GC) (e.g., 100ms → 10ms)
  - GC-induced read tail latency has been optimized by sophisticated GC schemes
- Block erase operation (e.g., 10ms/block)
  - Has become most dominant source of read tail latency

[1] Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Motivation: Two major sources of long read tail latency

- Garbage collection (GC) (e.g., 100ms → 10ms)
  - GC-induced read tail latency has been optimized by sophisticated GC schemes
- Block erase operation (e.g., 10ms/block)
  - Has become most dominant source of read tail latency

[1] Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Motivation: Two major sources of long read tail latency

- Garbage collection (GC) (e.g., 100ms → 10ms)
  - GC-induced read tail latency has been optimized by sophisticated GC schemes
- Block erase operation (e.g., 10ms/block)
  - Has become most dominant source of read tail latency

[1] Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Motivation: Two major sources of long read tail latency

- Garbage collection (GC) (e.g., 100ms $\rightarrow$ 10ms)
  - GC-induced read tail latency has been optimized by sophisticated GC schemes

- Block erase operation (e.g., 10ms/block)
  - Has become most dominant source of read tail latency
  - Erase suspension\(^1\) can effectively decrease block erase latency

\[1\] Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Motivation: Two major sources of long read tail latency

- Garbage collection (GC) (e.g., 100ms → 10ms)
  - GC-induced read tail latency has been optimized by sophisticated GC schemes
- Block erase operation (e.g., 10ms/block)
  - Has become most dominant source of read tail latency
  - *Erase suspension*[^1] can effectively decrease block erase latency

[^1]: Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012

However, existing erase suspension can cause write starvation and NAND reliability problem!
Our contributions: Practical erase suspension

• Observation
  – Modern SSDs perform erase operation with multiple discrete pulses to provide well-aligned safe points for suspending an ongoing erase
Our contributions: Practical erase suspension

• Observation
  – Modern SSDs perform erase operation with multiple discrete pulses to provide well-aligned safe points for suspending an ongoing erase

• We propose three practical erase suspension schemes
Our contributions: Practical erase suspension

• Observation
  – Modern SSDs perform erase operation with multiple discrete pulses to provide well-aligned safe points for suspending an ongoing erase

• We propose three practical erase suspension schemes
  – Immediate erase suspension (I-ES): Aborts erase immediately and restarts from previous safe-point
Our contributions: Practical erase suspension

• Observation
  – Modern SSDs perform erase operation with multiple discrete pulses to provide well-aligned safe points for suspending an ongoing erase

• We propose three practical erase suspension schemes
  – Immediate erase suspension (I-ES): Aborts erase immediately and restarts from previous safe-point
  – Deferred erase suspension (D-ES): Waits until the current erase pulse is finished

![Diagram showing voltage changes over time with erase and verify pulses and an arrival of read request.]
Our contributions: Practical erase suspension

- **Observation**
  - Modern SSDs perform erase operation with multiple discrete pulses to provide well-aligned safe points for suspending an ongoing erase

- **We propose three practical erase suspension schemes**
  - Immediate erase suspension (I-ES): Aborts erase immediately and restarts from previous safe-point
  - Deferred erase suspension (D-ES): Waits until the current erase pulse is finished
  - Timeout-based erase suspension (T-ES): Adaptively switches between I-ES and D-ES

![Diagram](attachment://diagram.png)
Prior work: Problems with existing erase suspension\cite{wu12} (1)

- **Problem #1: Write starvation**
  - With bursty reads

1) Remaining erase pulse (9ms) may fail to make a progress by incoming reads

\textit{Erase (and Write) Starvation!}

\cite{wu12} Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Prior work: Problems with existing erase suspension \[1\] (2)

- **Problem #2: Endurance degradation**
  - With bursty reads

2) Erase suspension/resumption causes additional stress to NAND

Over-erase NAND blocks → Increase uncorrectable bit error rate (UBER)

Endurance degradation of SSD!

[1] Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Practical erase suspension: Background

- **NAND erase operation**
  - Pulls electrons out of floating gate by applying very high voltage

- **Incremental Step Pulse Erasing (ISPE)**
  - Standard technique to minimize damages on NAND cells
  - Applying several, discrete pulses (of ~1ms) with increasingly higher nominal voltages

![Diagram showing ERS Voltage over time](chart)

- Square: Erase pulse
  - Erase cells in a NAND block
- Dot: Verify pulse
  - Sense which cells are erased
Practical erase suspension: Immediate erase suspension (I-ES)

- **I-ES operations**
  - Suspend: Immediately terminates ongoing erase step (taking ~ 100\(\mu\)s)
  - Resume: Restarts the suspended erase pulse from the beginning
Practical erase suspension: Immediate erase suspension (I-ES)

- I-ES operations
  - Suspend: Immediately terminates ongoing erase step (taking \( \sim 100\mu s \))
  - Resume: Restarts the suspended erase pulse from the beginning

Diagram showing:
- ERS Voltage
- Arrival of read request
- ERS Loop
- Time (ms)

Legend:
- : Erase pulse
- : Verify pulse
Practical erase suspension: Immediate erase suspension (I-ES)

- **I-ES operations**
  - Suspend: Immediately terminates ongoing erase step (taking \( \sim 100\mu s \))
  - Resume: Restarts the suspended erase pulse from the beginning
Practical erase suspension: Immediate erase suspension (I-ES)

- **I-ES operations**
  - Suspend: Immediately terminates ongoing erase step (taking ~ 100μs)
  - Resume: Restarts the suspended erase pulse from the beginning
Practical erase suspension: Immediate erase suspension (I-ES)

- **I-ES operations**
  - **Suspend**: Immediately terminates ongoing erase step (taking ~ 100µs)
  - **Resume**: Restarts the suspended erase pulse from the beginning
Practical erase suspension: Immediate erase suspension (I-ES)

- **I-ES operations**
  - Suspend: Immediately terminates ongoing erase step (taking ~ 100µs)
  - Resume: Restarts the suspended erase pulse from the beginning

![Diagram showing the process of I-ES operations]

1. Arrival of read request
2. Erase suspend
3. Read
4. Erase resume

ERS Voltage

Time (ms)

- Erase pulse
- Verify pulse
Practical erase suspension: Immediate erase suspension (I-ES)

- **I-ES operations**
  - **Suspend**: Immediately terminates ongoing erase step (taking ~ 100μs)
  - **Resume**: Restarts the suspended erase pulse from the beginning
  - Does not guarantee forward progress of erase operation → Write starvation problem!

### Write Tail Latency

Baseline: 1ms

Original ES: >10s

I-ES: >10s

FIO Thread #1: 128KB Read QD1, Thread #2: 128KB Write QD1
Practical erase suspension: Deferred erase suspension (D-ES)

- **D-ES operations**
  - Suspend: Waits until current erase step is finished (erase and verify pulse)
  - Resume: Start the next erase pulse

![Diagram showing the sequence of events in D-ES operations](image)

(1) Arrival of read request

ERS Voltage

Time (ms)

- : Erase pulse
- : Verify pulse

Nth ERS Loop
Practical erase suspension: Deferred erase suspension (D-ES)

- **D-ES operations**
  - Suspend: Waits until current erase step is finished (erase and verify pulse)
  - Resume: Start the next erase pulse
Practical erase suspension: Deferred erase suspension (D-ES)

- **D-ES operations**
  - Suspend: Waits until current erase step is finished (erase and verify pulse)
  - Resume: Start the next erase pulse

![Diagram showing erase suspension process](image_url)
Practical erase suspension: Deferred erase suspension (D-ES)

- **D-ES operations**
  - Suspend: Waits until current erase step is finished (erase and verify pulse)
  - Resume: Start the next erase pulse

![Diagram showing the process of D-ES operations](image-url)
Practical erase suspension: Deferred erase suspension (D-ES)

- **D-ES operations**
  - Suspend: Waits until current erase step is finished (erase and verify pulse)
  - Resume: Start the next erase pulse
  - No erase and write starvation problem, but *longer read tail!* (i.e., length of single step, ~ 1ms)
Practical erase suspension: Timeout-based erase suspension (T-ES)

• T-ES operations
  1. Performs I-ES until erase operation is suspended for a timeout period ($N$ ms)
  2. If a timeout happens, switches to D-ES to avoid erase and write starvation
Practical erase suspension: Timeout-based erase suspension (T-ES)

• **T-ES operations**
  1. Performs I-ES until erase operation is suspended for a timeout period ($N$ ms)
  2. If a timeout happens, switches to D-ES to avoid erase and write starvation

• **Choice of erase timeout period ($N$)**
  - Provides an effective control knob for read/write latency
  - Trades maximum write tail latency for reduced read latency

\[
\text{Ex) } N = 64\text{ms, and GC Write Latency} = 35\text{ms}
\]
\[
\Rightarrow \text{Maximum Write Latency} \leq 100\text{ms}
\]
Evaluation: Methodology

- NVMe SSD simulator: MQSim\[^1\]
- Benchmarks: Flexible I/O Tester, Aerospike Certification Tool (ACT) and TPC-C
- Comparison of six designs:
  - **Baseline** (no suspension) and **Ideal-ES** (erase suspension with zero penalty)
  - Erase suspension (**ES**)\[^2\]
  - Immediate-ES (**I-ES**), Deferred-ES (**D-ES**), and, Timeout-based-ES (**T-ES**)

<table>
<thead>
<tr>
<th>PCIe Gen 3 X 4 Lane, 240GB, NVMe SSD Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND Configurations: 4 channels, 4 chips/channel, 1 die/chip</td>
</tr>
<tr>
<td>FTL Schemes: Page Mapping, Preemptible GC</td>
</tr>
<tr>
<td>NAND Latency: Read: 3μs, Program: 100μs, Block Erase: 1ms per step (5 steps), Erase Suspension Penalty: 100μs, T-ES timeout: 64ms</td>
</tr>
</tbody>
</table>

\[^1\] Tavakkol et al, MQSim: A framework for enabling realistic studies of modern multi-queue SSD devices, USENIX FAST 2018
\[^2\] Wu et al, Reducing SSD Read Latency via NAND Flash Program and Erase Suspension, USENIX FAST 2012
Evaluation: Flexible I/O Tester (FIO)

- **FIO random test**
  - Read 70%, Write 30%, 4KB QD 16

![Graph](image)

- Baseline → ~5ms (entire erase operation)
- D-ES → ~1ms (single erase pulse)
- ES, I-ES, T-ES → ~100µs (suspension latency)

- I-ES, T-ES → Long write latency due to repeated erase suspension
Evaluation: Aerospike Certification Tool (ACT)

- ACT: Database benchmark
  - Consists of three threads, and gradually increases I/O rate in integer multiples

<table>
<thead>
<tr>
<th>ACT workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>o T1: 2K small (1.5KB, QD1) reads/s</td>
</tr>
<tr>
<td>o T2: 24 large (128KB, QD1) reads/s</td>
</tr>
<tr>
<td>o T3: 24 large (128KB, QD1) writes/s</td>
</tr>
</tbody>
</table>

Test Item | Evaluation Criteria | SSD #1 | SSD #2 |
----------|---------------------|--------|--------|
Performance Test | i) 95% of I/O < 1ms  
ii) 99% of I/O < 8ms  
iii) 99.9% of I/O < 64ms | 10X | 8X |
Stress Test | iv) I/O latency < request period | 2X | 10X |
**Evaluation: Aerospike Certification Tool (ACT)**

- **ACT test results**
  - Baseline shows poor *performance test* result (14x) due to long-tail latency of read request
  - ES and I-ES suffer write starvation problem (22x)
  - D-ES and T-ES demonstrate good results (30x) for both stress and performance tests

![Latency Graph](image)

- *30x workload multiplier*
Evaluation: Transaction processing benchmark (TPC-C)

- TPC-C from SNIA

(a) Read tail latency
- Baseline → ~5ms (entire operation)
- D-ES, T-ES → ~1ms (single erase pulse)
- ES, I-ES → Failure by write command timeout

(b) Write tail latency
- T-ES → Timeout (64ms) + GC latency (24ms)
Conclusion

- Practical erase suspension harnesses the full potential of NAND flash-based SSDs
  - Minimizes the impact of erase operation on read tail latency
  - Achieves very low read tail latency without write starvation and endurance degradation

[Diagram showing read tail latency improvements with practical erase suspension]
Thank You!

Our simulator is available at
https://github.com/SNU-ARC/MQSim-Practical-ERS-SUS