To TRIM or Not to TRIM: Judicious TRIMing for Solid State Drives

Choulseung Hyun†, Jongmoo Choi††, Donghee Lee†, Sam H. Noh†††
†University of Seoul, ††Dankook University, †††Hongik University
Seoul, Korea

Introduction

- Discrepancy exists between file system and SSDs' view of storage
- TRIM command works to eliminate this discrepancy
- Propose new policy to reap benefits intended of TRIM

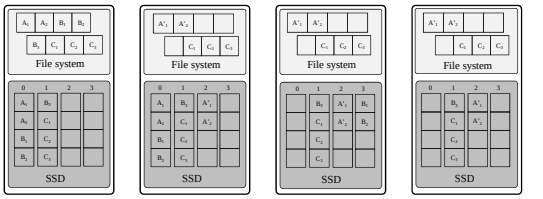
Contributions

(1) We propose the Selective TRIM (SeT) policy

(2) Through implementation and experiments, we show up to 32% improvement for SeT relative to current default policy

TRIM command and Observation on the Effect of TRIM Usage

- **TRIM command: informs SSDs of deletion**
- **The discrepancy: an example**



(a) Initial state (b) File system overwrites A and deletes B (c) SSD performs garbage collection (GC) (d) File system issues TRIM and SSD performs GC

Figure 1. File system and SSD view discrepancy

Fig.1(a): Files A and B initially exist in file system and SSD

Fig.1(b): Two different (file system and SSD) viewpoints after file system overwrites file A with A' and deletes file B

Fig.1(c): Pages (B_1, B_2, and B_3) in SSD are not removed even after garbage collection

Fig.1(d): Changes to SSD as TRIM is first issued in Fig.1(b) and then, garbage collection performed in SSD
⌘ FTL takes note that the pages of B have been deleted (through the TRIM command) resulting in the pages being erased during garbage collection

Observation 1: Overhead of issuing TRIM is non-negligible

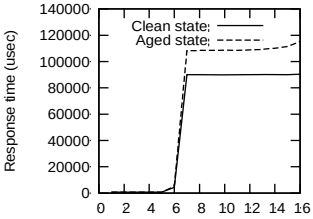


Figure 2. Overhead of TRIM command

Observation 2: Performance benefit of TRIM can be garnered most when SSD utilization is high

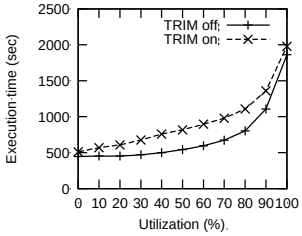


Figure 3. Utilization vs GC time

Observation 3: Other I/O operations are negatively impacted by TRIM

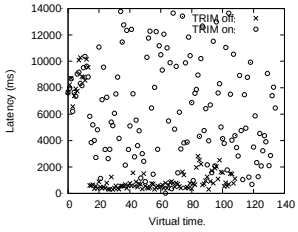


Figure 4. Time distribution of journal commit

Selective TRIM (SeT): a new TRIM issuing policy

Key idea

(1) Balancing observations (1) and (2)
- ➔ Issue TRIM when benefits are maximized
- ➔ Do not issue TRIM when overhead may be greater than benefits

(2) Based on observation (3)
- ➔ Issue TRIM when it will have less impact on other I/O operations

a. Do not TRIM when utilization continues to increase
b. Issue TRIM when utilization continues to decrease

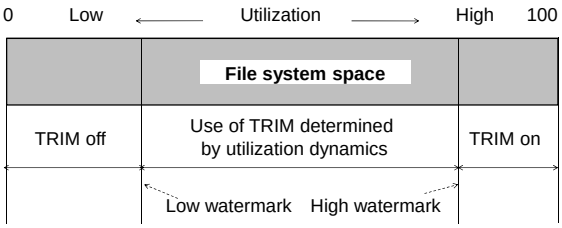


Figure 5. Overview of Selective TRIM (SeT) policy

Experimental Results

Platform: Linux version 2.6.39.4 on Intel i3 CPU with 4GB of memory with 60GB SSD

Benchmarks

(1) Postmark benchmark
- ➔ Short-term workloads
- ➔ Long-term workload: repetition of Setup & Transaction stages

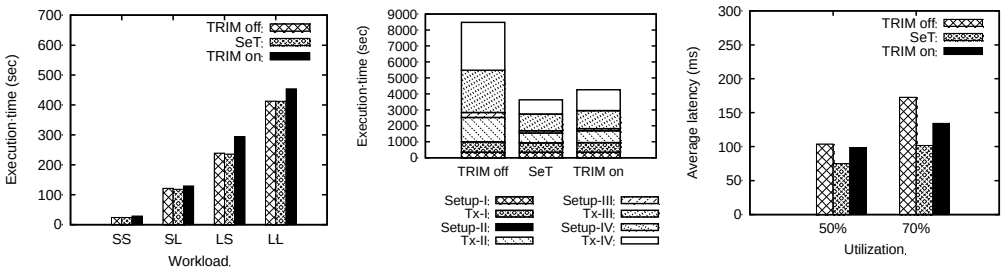
a. Utilization setup stage: 60% ➔ 40% ➔ 70% ➔ 60%
b. Transaction stage: running LLL workload

Table 1: Parameters of the Postmark benchmark

Type	Workload	File size	No. of files	No. of transactions
Short-term	SS	9-15KB	10,000	100,000
	SL	9-15KB	200,000	100,000
	LS	0.1-3MB	1,000	20,000
	LL	0.1-3MB	4,250	20,000
Long-term	LLL	0.1-8MB	2,000	10,000

(2) Filebench benchmark
- ➔ Execute Filebench file server workload for one hour
- ➔ Initial SSD utilization set to either 50% or 70%

Results



(a) Postmark short-term workload (b) Postmark long-term workload (c) Filebench fileserver workload

Figure 6. Benchmark results

Analysis of experiments

(1) Figure 6(a): When SSD utilization is low, the device is negatively affected as invoking TRIM incurs overhead

(2) Figure 6(b):
- 'TRIM on', the default TRIM issuing policy in Linux, compared to 'TRIM off' improves performance by as much as 100%
- SeT improves performance by as much as 17% compared to 'TRIM on'

(3) Figure 6(c): SeT performance is the best, improving as much as 32% over 'TRIM on'

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. R0A-2007-000-20071-0) and by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. 2009-0085883).