SSD TRIM OPERATIONS:EVALUATION AND ANALYSIS

Author

Chau-yuan Mao

Supervisor

Li-Pin Chang

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
NATIONAL CHIAO TUNG UNIVERSITY
HSINCHU, TAIWAN
JULY 2013

E
S
A
1896

Table of Contents

List of Tables

List of Figures

Abstract

NAND flash-based solid-state disks (SSDs) have become widely available and to replace HDDs. Because NAND flash characteristic cause the disadvantage of SSD to degradation of write performance. To improve the write performance, the TRIM command is used. In this paper we discuss how TRIM effect the performance, we sperate affecting factors into three parts:1.Host behaviors, 2.Interface bandwidth and 3. Device internal behaviors. We design several experiments to observed the affecting factors. Discussed when and why TRIM command is benefit for write performance or when can degradation write performance.

Acknowledgements

My deepest gratitude goes first and foremost to Professor Li-Pin Chang, for his constant encouragement and guidance. My research have bottlenecks before, without his encouragement and guidance I can not even finish this thesis, thanks for you who have instructed and helped me a lot in the past two years.

Second I want to thanks all the colleagues of ESS laboratory. For Wen-Ping Li, Po-Han Sung, Tung-Tang Chou, Ti-Kang Chang and Cheng-Yi Wen, your passion and unselfish share of experience that help me solve lots of problem saving time. Thanks for Sheng-Min Huang, Cheng-yu Hung , we struggle together and across the difficulty. Thanks Chun-Yu Lin, Pin-Xue Lai, Guan-Chun Chen, Ming-Chi Yan, Po-Hong Chen, Tu-Syun Liou give lab cohesion and give good atmosphere to study. And Thanks for National Chiao Tung University create great environment to doing research.

Last my thanks would go to my family and friends who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis.

Chapter 1

Introduction

Flash-based solid-state disks (SSDs) have become widely available in recent years. SSD has many advantages over hard disk drive such as low-power consumption, small size and high shock resistance. SSD provides three operations: read, write and erase. The read, write operation unit is page and erase unit is block. Due to these NAND flash characteristic, SSD design is an challenging task.The read performance is high, while writes will drop due to its "erase-before-write", which means a block must be erase before a data is written into. There are several ways manufactures attempt to reduce erase-before-write requirement penalty, including overproportioning of the SSD, flash translation layer (FTL) design, and implementing TRIM command.

TRIM is a simple command to declare which file is no longer needed. The TRIM original design to performs thin-provision. Nowadays, TRIM useful to SSD that ignore trimmed sectors. Implement TRIM has many benefit like reduce erase-before-write penalty, improved endurance,system and SSD. The figure 1.1 shows the example of inconsistent problem. Consider one situation w lower write amplification, higher throughput and can solve inconsistent between fileithout TRIM, assume file A is already exists. Right after a deletion of file A like figures step 1 , the file system will edit the metadata and mark file A is free it is shown in figure step 2 and step 3. The three step is all file system can do, but at the same time the solid state drives may still considered it even though file A doesn't exist anymore. This problem effect that if garbage collection happened, the more detail is in after chapter.

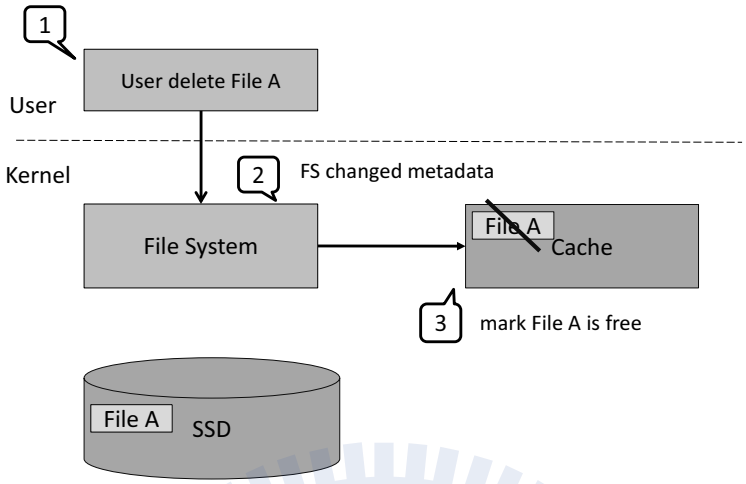


Figure 1.1: The figure shows data inconsistent problem between file system and low device storage

The TRIM command can solve the inconsistent problem because when the file system is done changing the metadata and mark file A is free, it will have further move send TRIM to low storage device. The SSD FTL received TRIM and will know the file A is no longer needed.

It seems like enabling TRIM should be beneficial to SSD write performance all the times. However, in some circumstances, using TRIM unexpectedly degrade he file-operation performance. This paper aims at finding when and why using TRIM will improve or degrade the performance.The rest of paper is organized as follow: Section 2 introduces the research about implementing TRIM , Section 3 presents the factors that effect performance and the TRIM format, Section 4 reports the experiments that about TRIM, Section 5 concludes the work.

Chapter 2

Related Work

TRIM command can improve over-provision and lower write amplification and effect the performance. One of the research is to quantify the over-provision and write amplification with TRIM, run workload on simulation and present the mathematical model to measure the value like reference in [1] , [2] and [3].The [4] provide a simple method about utilization when given trim intensity and the number of pages. The other is measure the performance and analyze. In [5], it compare trim-enable and trim-disable write performance on two different file system, NTFS and EXT4. To compare NTFS and EXT4 is because how the file system performs TRIM right after TRIM is a factors that affect performance. This paper compare the write performance, re-write performance and random write performance. The result shows trim-enable is better on random write for both file system. In [6] said TRIM may drop performance so it provide balanced way issue TRIM, and [7] propose a non-reorderable persistent TRIM (PTRIM) to compare with TRIM. And in Linux, it also have different way of send TRIM called Fstrim, This make the file system to find all unused blocks in device that have not been discarded since the mount. Use Fstrim is because that TRIM (called discard in Linux) option will slower in some cases. In our design experiments we use virtual drive to count the TRIM command and analyze the format of received TRIM command. The virtual drive [8] is present a virtual platform of SSD.

Chapter 3

Background

3.1 Performance-Affecting Factors

As our opinion, the system with trim-enable of factors that affecting performance fell into three parts: 1.Host behaviors, 2.Interface bandwidth, and 3.Device internal behaviors. The architecture is as Figure 3.1. The host behaviors is about the file system behaviors after user performs delete operation, like when does file system send TRIM command after delete operation, if file system send TRIM immediately it may cause many TRIM commands. If not, it means the file system may organized the information of deleted files and can reduce

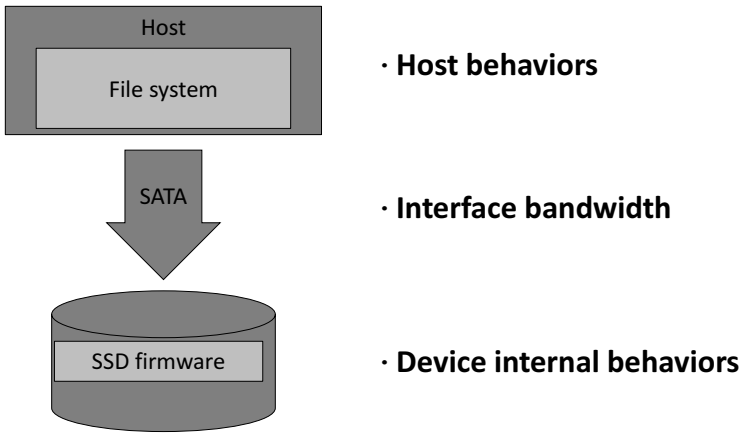
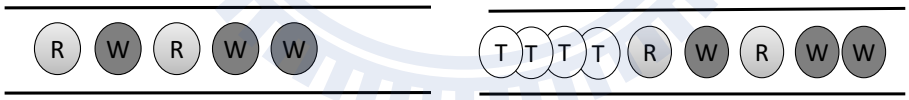


Figure 3.1: This figure shows the three parts factors of effect performance

TRIM commands. For Interface bandwidth, it is about the system can overhead TRIM commands or not. And the device internal behaviors is the SSD firmware design, after the SSD FTL receive TRIM, the FTL decide what to do. The third part is an unknown black box, we can not know the firmware designed in different manufacturers. As we said before, the benefit can reduce garbage collection cost. Consider one situation in trim-disable system because the inconsistent problem. the SSD does not know the information of deleted data, and if the garbage collection happened. It performs copy move that the page which belongs the deleted file to new free block, and see the new free block as data block. The copy deleted file's page is unnecessary moves. And in trim-enable system, the SSD will know the deleted data information, it will not make the same fault like trim-disable does. It will save the unnecessary copy and can reduce garbage collection cost. If we use the interface view, like 3.2. Assume the user is performs delete operation, right after delete user performs read and write operations. The system with trim-disable is 3.2a, the bandwidth has write commands and read commands. The system with trim-enable is 3.2b, we can find that the bandwidth filled with TRIM commands, system will deal the TRIM first, and it will delay the read, write commands that original can do. If the bandwidth has many TRIM commands to deal with, then it may drop the performance, the extra time to deal with TRIM commands is TRIM overhead.



(a) the interface with trim-disable (b) the interface with trim-enable

Figure 3.2: This figure is using interface view, the 3.2b shows the interface view with the system is trim-enable, the 3.2a shows the interface view with the the system is trim-disable. every circle means one commands. The letter R means read command, letter W means write command, letter T mean TRIM command

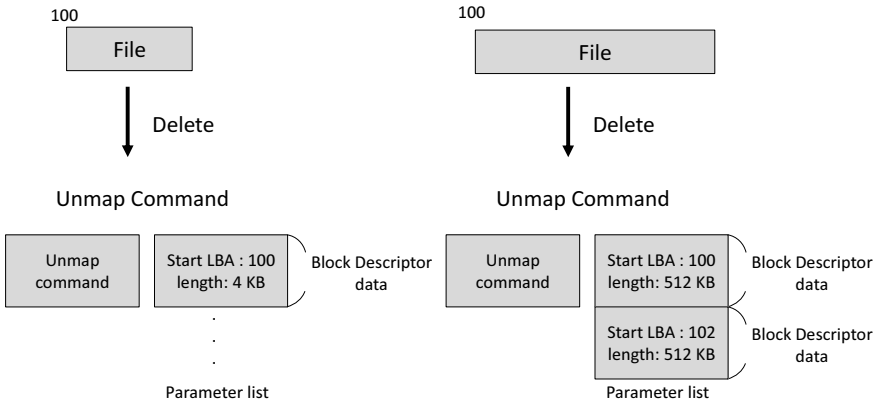
3.2 TRIM Format

TRIM command also named UNMAP command in SCSI. The format is defined in [9]. So we use UNMAP to replace call the TRIM in this section. To make sure UNMAP command work, first the operation system must support UNMAP command, and second the device

storage must ensure itself for UNMAP. The version of operation system support TRIM is Windows 7, Windows 8, Linux 2.6.28 and later version. To ensure a block device enable itself for UNMAP, the device must support vital product data (VPD) first. VPD is the information about stored computer's device, the VPD page structures are returned by an INQUIRY command. For support VPD, setting the EVPD to 1 and specifying the page code which page code effect about choose type of the extend VPD structure. One of the types is Block Limits VPD page, it has fields Maximum UNMAP LBA Count and Maximum UNMAP Block Descriptor Count, the Maximum UNMAP LBA count field indicates the maximum number of LBAs that may be unmapped by an UNMAP command. And the Maximum UNMAP Block Descriptor Count field indicates maximum number of UNMAP block descriptors. One of the type is Logical Block Provisioning VPD, to make sure the device support UNMAP, it must set VPD page logical block provisioning structure's LBPU filed bit to 1.

After delte file, storage device received UNMAP command along with UNMAP parameter list see figure 3.4, it shows the command format. The UNMAP parameter list included block descriptor data, every block descriptor data descriptor the unmapped file of the start LBA and the length of blocks, the format shows in figure 3.4b and 3.4c. Figure 3.4a, the UNMAP command fields include operation code, parameter list length, and control field. The operation code field is the first byte of command descriptor block (CDB) that identifying the operation. Parameter list length field specifies the length in bytes of the UNMAP parameter data. Figure 3.4b, the UNMAP parameter list contains the data sent by application client along with UNMAP command. UNMAP parameter list include parameter list header and UNMAP Block Descriptor Data. UNMAP parameter list header field indicates the following length in bytes of UNMAP Block Descriptor Data. The UNMAP Block Descriptor has field of first LBA of the block should be unmapped and field of the number of the LBAs to be unmapped began with the first LBA. There are not only one UNMAP Block Descriptor Data in the UNMAP parameter list.

For example like Figure 3.3. If we delete a file, the starting logical block address is 100 and the file size 4 KB. We assume length filed of the block descriptor data only can show the maximum size is 524288 in bytes (512 KB). The format is shown in figure 3.3a, block descriptor data filed Start LBA and length is 100 and 4096. If we delete large file, the



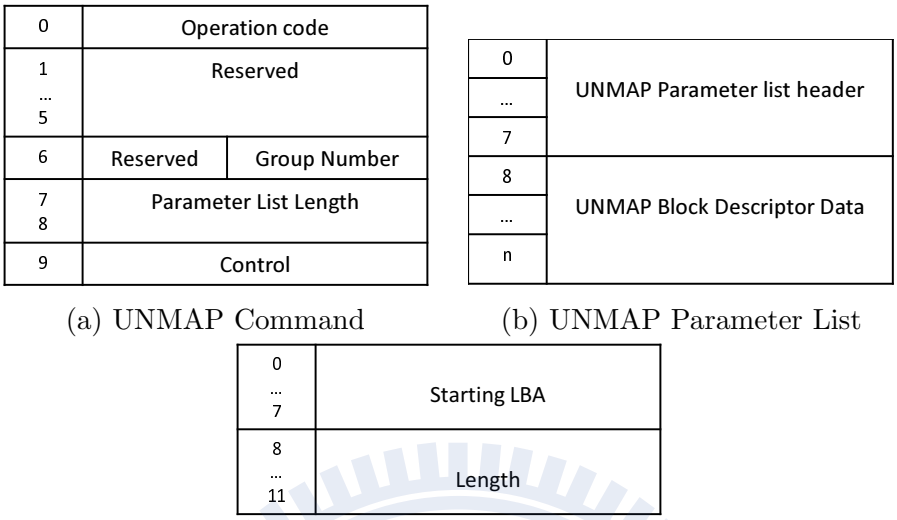
(a) Small file (b) Large file

Figure 3.3: The figure shows the example of unmap command and unmap parameter list, figure (a) after delete 4KB-file the receded format and starting logical block address is 100, figure (b) after delete 1024KB-file the receded format and starting logical block address is 100

starting logical block address is 100 and the file size 1024 KB. The length maximum of block descriptor data is still 512 KB. The format is shown in figure 3.3b, because the file is large, so there have two block descriptor data. One is block descriptor data filed start LBA and length is 100 and 524288. And the other block descriptor data start LBA field is 100 and length field 524288. The parameter list have limited block descriptor data. So we can find that, with the file becomes bigger the only one TRIM command can not describe the file, it may have more TRIM commands to describe it. It can extend one phenomenon the large the file deleted the more the TRIM commands issues.

nd the setup of Block Limits VPD Maximum UNMAP LBA Count field is 256 and Maximum UNMAP Block Descriptor Count field is 16, The UNMAP command will set parameter list length 18, and parameter list header will set UNMAP data length 16, UNMAP Block Descriptor Data Length 10. The setting of UNMAP Block Descriptor Data will show the file starting LBA address and length is 4096 in bytes.

After make sure low device support TRIM, we have make the operating system support TRIM. In nowadays, we have manual to make the operating system support TRIM. In Windows, firs have to use command prompt, and type the following "fsutil behavior set disabledeletenotify 0". After that, the operating system will support TRIM. To disable TRIM, type "fsutil behavior set disabledeletenotify 1". And If want to know the Windows



(a) UNMAP Command (b) UNMAP Parameter List

(c) BLOCK Descriptor Data

Figure 3.4: the figure(a) and (b) shows the format of unmap command and unmap parameter list, figure (c) is block descriptor data which is included by unmap parameter list, each block descriptor data shows the starting address and length of unmapped file

support TRIM or not, type "fsutil behavior query disabledeletenotify". The output is 1 means trim-disable , the output is 0 means trim-enable. In Linux, to enable TRIM command is to edit the fstab file and insert "discard" before after the file system format. Or can using the command "mount" and open the option mode "-o discard" to mount the SSD.

Chapter 4

Experiments Results

4.1 Experimental Setup

This section we provide several experiments to discussed TRIM. The experiments consist of two parts. The first part section B. is to observed host behaviors, the second part section C. is to compare trim-enable and trim-disable performance on real SSD.

In first part, the host behavior is about to know when the file system send TRIM command, the relations about deleted files and number of TRIM commands, and how the file size and file number affect TRIM commands. We set several tests in these part. To observe these host behaviors, we use virtual drive to accepted TRIM and calculate TRIM commands. The tests is excess on two environments. One is on Windows 7 and setting the file system is NTFS, the other is Linux 3.9.11 and setting the file system is EXT4. To choose these two file system is because this is popular file system and most important the two file system is support TRIM.

In second part, there has four experiments in these part. We used two benchmark to test the effect: Iometer and Postmark, the describe detail of benchmark is in after section. In every benchmark we will make sure that to write full random data to bypass the compression engine inside of the solid-state disk. We choose two popular real solid-state

SSD	Intel SSD 520 series	Plextor M5 Pro
Disk size	120 GB	128 GB
controller	SandForce SF-2281	Marvel 88SS9187
DRAM cache	NO	256 MB
SATA	SATA 6Gb/s	SATA 6Gb/s

Table 4.1: Characteristics of SSD

OS	File Size	Idle Time	Received TRIM
Windows	10 KB	0 (s)	1
Windows	10 KB	30 (s)	2
Linux	10 KB	0 (s)	0
Linux	10 KB	30 (s)	0

Table 4.2: The result table of first test

disks for our test, plextor SSD and intel SSD, the information can see Table 4.1. We also conducted the experiment on both Windows 7 and Linux 3.9.11 and with file system NTFS and EXT4, each run of experiments has options enable or disable TRIM. We compare the write performance of trim-enable and trim-disable and analyze it. The setup detail is in its own sections.

4.2 Host behaviors on TRIM

This section we provide several tests to observed the behaviors of TRIM command. The First test is to know when the file system issues trim after file deletion. The test procedure in step by step is that, first step we created a 10 kilobyte file on virtual drive, and we known delete 10 kilobyte-file will cause virtual drive received one TRIM command. In second step, immediate deleted the file to produce TRIM command. Third step, we wait an idle time, the idle time is variable of 0 second and 30 seconds. Fourth, after waiting we do first step and second step again. If the result in different idle time received same commands, then we can make sure that file system issues TRIM immediately because that, If file system create same file soon than the file will in different logical address, so it will have one TRIM commands descriptor deferent address, otherwise then the trimming is not immediate.

The result is shown in Table 4.2. The result is in last column. In Windows, when idle

OS	File Size	N	Received TRIM	TRIM with fsync
Windows	10 KB	100	10	-
Windows	10 KB	10000	951	-
Linux	10 KB	100	0	5
Linux	10 KB	10000	48	105

Table 4.3: The result table of second test

time is 0 second, then the received TRIM command is 1. If the idle time is 30 seconds, then the received TRIM command is 2. As we explains before, received different TRIM commands means not issue TRIM immediately. The result can shows that the NTFS will not issues TRIM command immediately but the delay is not very large. In Linux, no matter the idle time is 0 second or the idle time is 30 seconds, the received TRIM command is 0. The reason do not received TRIM command is because the file is deleted before the transaction committed. If we use command to force transaction commit then it will affect the experimental correctness, so we can not know the EXT4 is immediately issue TRIM or not.

The second test is to observed the relation between the number of deleted files and the number of TRIM commands. The test procedure in step by step is that, first step we created N files on virtual drive that file size is also set 10 kilobyte. Second step, after created we deleted all those N files. The N is variable of file numbers, to comparison we set the variable N to 100 files or 10000 files. We expect that the bigger the N is the more TRIM commands will appear because many files needed many TRIM commands to describe.

Because in Linux there will have delete file before the transaction committed problem, this problem is mentioned as before test, so we use 'fsync' to force transaction commit.The result is shown in Table 4.3, The last two column is the result of received TRIM commands and received TRIM commands with 'fsync'. In Windows, if the file number is 100 we received 10 TRIM commands, and the file number is 10000 we received 951 TRIM commands, the Windows do not need 'fsync' so the TRIM with fsync in Windows is '-'. In Linux without 'fsync the file number is 100 the received command is 0, the file number is 10000 the received command is 48. And with 'fsync',the file number is 100 we received 5 TRIM commands, and the file number is 10000 we received 105 TRIM commands. the result is as expect that the more the files deleted, the more TRIM command received.

OS	File Size	N	Received TRIM	TRIM with fsync
Windows	4 MB	1	7	-
Windows	4 KB	1024	9	-
Linux	4 MB	1	0	1
Linux	4 KB	1024	0	10

Table 4.4: The result table of third test

The third test is to know the effect of file size and file number. So the test procedure in step by step is that first step, we set same writing amounts file with large files and small files on virtual drive, and second step is delete all the files on virtual drives. Setting the same writing amounts and to compare large file and small files which one will issues more TRIM commands. We set the write amounts is 4 megabyte. The variable is that, one is with 1024 files with file size is 4 kilobytes the total write amounts is 4MB , and the other is one large file with file size is 4 megabyte the total write amounts is 4MB. We expect that deleting many files requires many TRIM commands because large file is easily describe by using starting address and length, and small files is in random starting address and it need more TRIM commands to describe.

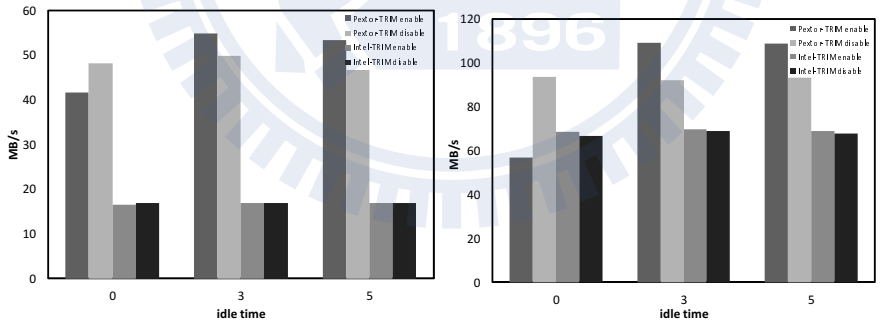
The result is shown in Table 4.4. In Windows, the large file received 7 TRIM commands and the small files received 9 TRIM commands. In Linux with 'fsync' to force transaction commit the result is in last column, the large file received 1 TRIM commands and the small files received 10 TRIM commands. The result is as expect that is mentioned but in Windows the difference is not very obviously. It is because that in Windows the writing file address is adjacent so it also can easily express by TRIM, it the file address is random it will have more TRIM commands to describe and the TRIM commands difference is more obviously.

This section is focus on the first affecting performance factor: Host behavior. After the tests we can summary that. The first test shows that file system does not immediately issue TRIM right after delete but very soon, and second test we can find that the more the files deleted the more the TRIM commands send. In the last test, we set the same writing amounts, delete small files will produce more TRIM commands than large files. The amounts of TRIM commands produced will affect second factors: Interface bandwidth. The more the TRIM commands issued the more the TRIM overhead is.

4.3 Experiment on real SSD

This section we have design four different experiments procedure and running benchmark on two real SSDs. The SSDs is mentioned as before. We choose two benchmark for our experiment: Iometer and Postmark. Iometer is an I/O subsystem measurement and characterization tool for systems. It was developed by Intel Corporation. The postmark is a benchmark to measure performance about the class of application like electronic mail. it means that the postmark will frequently access large amounts of small files. The postmark execute procedure is sperate in three phases. First phase it will create file pool, we can setting the file numbers and file size of file pool. Second phase is transitions, the transactions will perform create, write, read, and delete operation for file pool, we can set the transaction times and ratio of create/delete, write/delete. The last phase is delete, it will delete all the files that the tool created on disk.

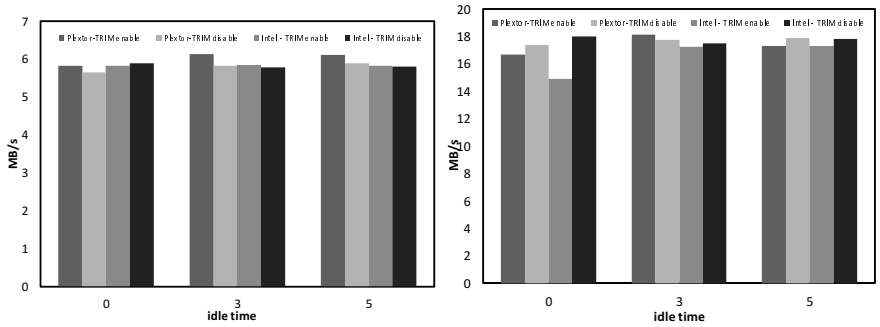
Experiment1



(a) Windows (b) Linux

Figure 4.1: This figure shows the Experiment 1 result running iometer benchmark, figure 4.1a shows iometer on Windows, figure 4.1b shows iometer on Linux

The first experiment is to know how garbage collection benefited from trimming. To achieved the purpose, the experimental procedure has four step, first step is to fill up 90% disk size, and second step is delete all the files that step 1 created on disk. The step 1 and step 2 is order to produce large amounts of TRIM commands in our experiment, and we want to compare trim-enable and trim-disable performance and analyze it. The third step is to wait an idle time. This step is order to give the system time to handle TRIM



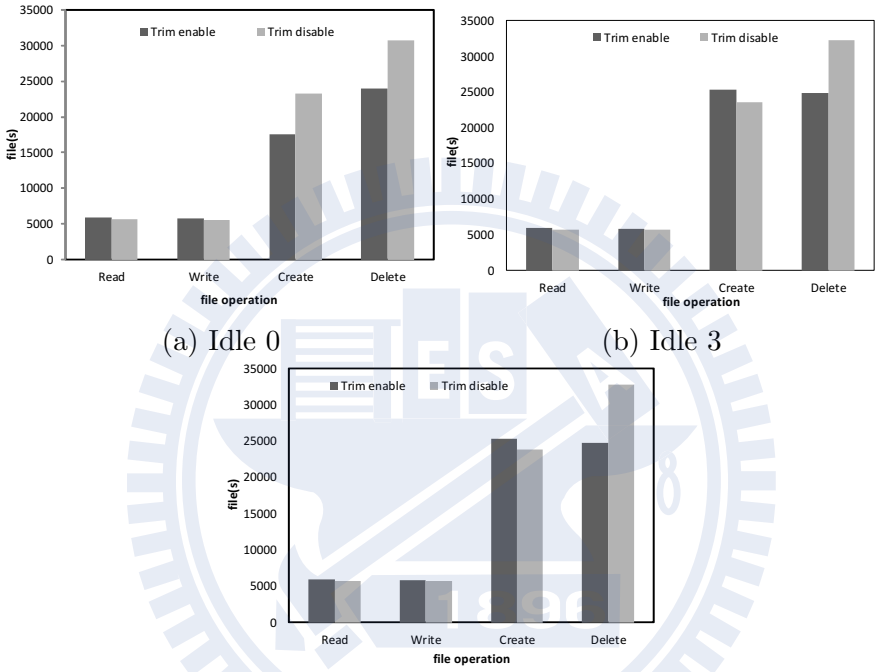
(a) Windows (b) Linux

Figure 4.2: This figure shows the Experiment 1 result running postmark benchmark, figure 4.2a shows postmark on Windows, figure 4.2b shows postmark on Linux

commands, the idle time is variable, which can be 0 minutes, 3 minutes, or 5 minutes. The fourth step is running benchmark on the real SSD, we use two benchmark: iometer and postmark. When using postmark, we skipped file deletion in transactions phase to avoid further trimming. We expect that the performance of trim-enable will better because it have less garbage collection cost.

The result of Running iometer is shown in Figure 4.1. We can observed that the intel SSD seems no different with trim-enable or trim-disable, no matter the environment is Windows or Linux. And in using plextor SSD the trim-enable performance is less than trim-disable when idle time is 0 because the TRIM overhead effect the performance, system will spent lots of time to handle large amounts of TRIM commands, after handling TRIM commands the performance shows better because the benefit of TRIM that trim-enable system has less garbage collection cost. The effect shows that when idle time is 3 minutes and 5 minutes, trim-enable performance is better than trim-disable performance. This phenomenon shows in both operation system Windows and Linux. The result of Running postmark is shown in Figure 4.2. the intel results are neither stable nor conclusive. In plextor SSD when idle time is 0 the trim-enable performance is less than trim-disable, after handling TRIM the performance is better except in Linux idle 5 minutes, the reason is same as before. But the performance shows not obviously and the performance is drop in Linux idle 5 minutes. The reason why with trim-enable on Linux with idle period is 5 will drop can explained when we observe postmark every file operation handles files, see Figure 4.3. It shows how many files process per file operation per second. The delete file operation shows

that the trim-enable handle less files than trim-disable. So the postmark write performance is not obviously because it is effect by delete operation. If the performance is expect delete operation than the performance will obviously like running iometer's result that the trim-enable system when idle period is 0 minutes the performance will less than trim-disable, and the trim-enable performance is better when idle period is 3 minutes and 5 minutes.



(a) Idle 0

(b) Idle 3

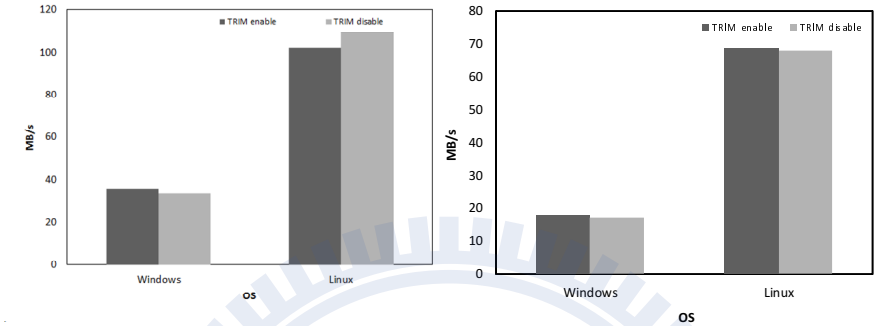
(c) Idle 5

Figure 4.3: This figure shows every operations performs files per second on Linux with postmark in every idle period, figure4.3a shows result in idle period 0, figure4.3b shows result in idle period 3. figure4.3c shows result in idle period 5.

4.3.1 Experiment2

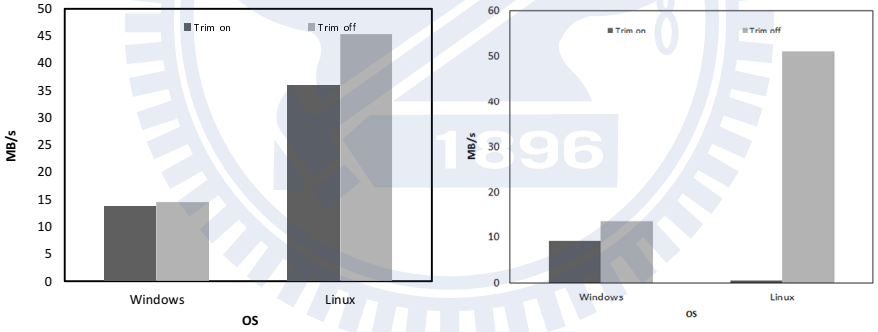
The second experiment is to know how garbage collection effect under high flash space utilization. To achieved the purpose, the experiment procedure is that, first step is to fill up 90% disk size, this step is to make high utilization disk. Second step is running benchmark to measure the write performance. We compare the performance with trim-enable and trim-disable and analyze it. In our procedure, it does not produce TRIM because there are no delete operation, so we set create/delete file in postmark phase 2

transactions, and the ratio of create to delete is 9:1. We expect that the trim overhead will drop the performnce. The iometer dose not performs delete itself and there are no delete operation in out procedure, so we expect when running iometer it will make no different between trim-enable and trim-disable.



(a) Windows (b) Linux

Figure 4.4: This figure shows the Experiment 2 result running iometer benchmark, figure(a) shows result with plextor SSD, figure(b) shows result with pintel SSD



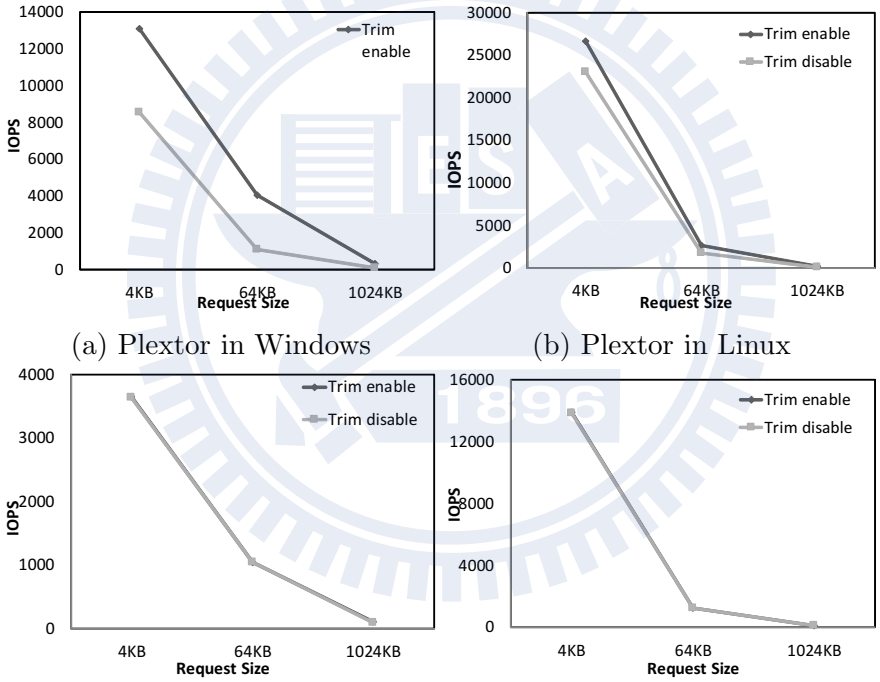
(a) Windows (b) Linux

Figure 4.5: This figure shows the Experiment 2 result running postmark benchmark, figure(a) shows result with plextor SSD, figure(b) shows result with pintel SSD

Running Iometer result is shown in Figure 4.4, the little difference is because we use full random write data, it will cause the different result in every run. The iometer result shows no different no mater the environment is Windows or Linux. The postmark result is shown in Figure 4.5. as expect that the performance of trim-enable is less than trim-disable, it because we trim overhead is affect the write performance. When use intel SSD in Linux the trim-enable performance is much less than the trim-disable, it shows that the TRIM overhead is bigger when using intel on Linux.

4.3.2 Experiment3

The result is shown in Figure 4.6. In figure 4.6c and 4.6d, intel SSD seems no different between trim-enable and trim-disable. The result is same as experiment 1 with intel SSD. In figure 4.6a and 4.6b, the result of plextor SSD is as expect. When the request size is 4 KB, the performance with trim-enable is much better than the trim-disable. And when request size becomes bigger the performance is becomes closer with trim-enable and trim-disable. The reason is as before said, the bigger the request size the less the garbage collection cost, and TRIM does not have much benefit with the large request size.



(a) Plextor in Windows (b) Plextor in Linux

(c) Intel in Linux (d) Intel in Windows

Figure 4.6: This shows the result of Exepriment3, figure(a) use plextor SSD on Windows, figure(b) use plextor SSD on Linux, figure(c) use intel SSD on Windows,figure(d) use intel SSD on Linux

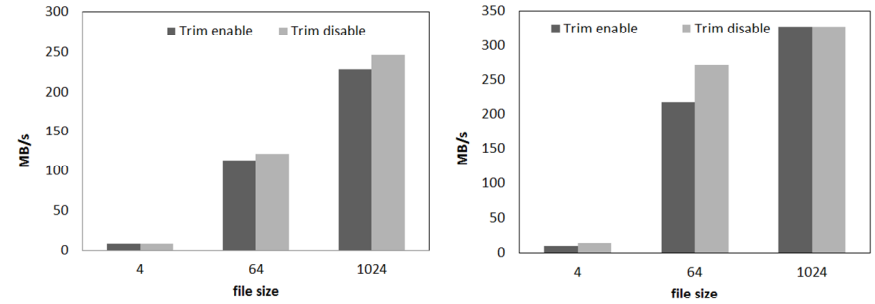
The result is shown in Figure 4.6. In figure 4.6c and 4.6d, intel SSD seems no different between trim-enable and trim-disable. The result is same as experiment 1 with intel SSD. In figure 4.6a and 4.6b, the result of plextor SSD is as expect. When the request size is 4 KB, the performance with trim-enable is much better than the trim-disable. And when request size becomes bigger the performance is becomes closer with trim-enable and

trim-disable. The reason is as before said, the bigger the request size the less the garbage collection cost, and TRIM does not have much benefit with the large request size.

4.3.3 Experiment4

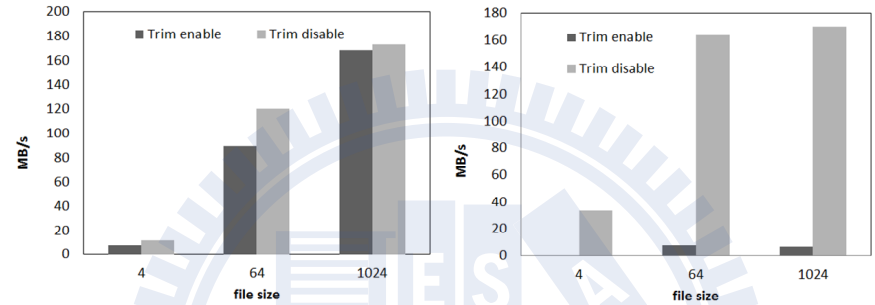
The fourth experiment is to know Different garbage collection pressures with a high flash space utilization. To achieved the purpose, the experiment procedure is that, first step is to fill up 90% disk size, as experiment 2 this step is to make high utilization disk. Second step is running postmark to measure the write performance. The reason why choose postmark is because the experiment procedure step 1 does not perform delete operation. So we have to use benchmark who has delete operation itself to produce TRIM commands to compare. We set create and delete file operation in postmark phase 2 transactions, and the ratio of create to delete is 9:1. The variable of file size is selected among 4KB, 64KB, 1024KB. We expect that the trim-enable performance is much less than trim-disable when request size is 4 KB because the garbage collection cost is higher and now plus the overhead of TRIM commands.

The result is shown in Figure 4.7. In plextor SSD, when the request size select 4 KB the performance shows no different between trim-enable and trim-disable, it because the garbage collection is heavy in high disk utilization. The performance is worse that TRIM benefit can not help much, so the performance can not see obviously different. The reason is same in intel SSD on Windows when the request size is 4KB. The intel on Linux the TRIM overhead is too much that trim -enable performance is much less than trim-disable. Expect the intel on Linux, we compare the performance when request size is 64 KB and 1024 KB, the write performance of trim-enable with request size is 1024 KB is much close trim-disable than with request size is 64 KB, the reason is because that the bigger request size has less numbers of TRIM commands. It also means it has less TRIM overhead, so the performance has bigger difference in request size 64 than the 1024 KB.



(a) Plextor in Windows

(b) Plextor in Linux



(c) Intel in Linux

(d) Intel in Windows

Figure 4.7: This shows the result of Exepriment4, figure(a) use plextor SSD on Windows, figure(b) use plextor SSD on Linux, figure(c) use intel SSD on Windows,figure(d) use intel SSD on Linux

4.4 Summary

In this section, it shows the performance of real SSD. We find that the TRIM benefits and TRIM drawbacks. The benefit is shown as experiment 1 and experiment 3, if can give system the enough time to handle TRIM commands, the write performance will better than trim-disable system. The drawback is writing right after deletion, the performance will drop by system have handle TRIM commands first. The experiment can see the affect of third affecting performance factor: Device internal behaviors. Comparing two different SSD in our experiments: intel and pletor. The intel SSD firmware has bad design for handling TRIM, the intel SSD with trim-enable always shows less or equal performance than trim-disable while plextor can performs better performance with trim-enable than trim-disable. Because intel SSD use Sandforce controller, the controller use compression data. We speculate that because compression data it improves the complexity of design FTL to handle TRIM commands, so when trim-enable it may spent more times to deal

with TRIM and cause the performance decrease.

Chapter 5

Conclusion

In this paper, we can find that when and why using TRIM will improve or degrade the SSD write performance. We can know that that how the host behavior and interface bandwidth affect performance. During our experiment we can observe the host behavior that the file system does not immediately issue TRIM right after delete files, the more the files deleted the more the TRIM accessed, and in same writing amounts the small files will cause more TRIM commands. The TRIM overhead is that the system will spent time to deal with TRIM, if there are too much TRIM commands, than it will drop performance badly. If the delete operation is perform and we want writing immediately, than the write performance will drop because TRIM overhead. So it is huge issue how to reduce TRIM overhead, to reduce the overhead is also have a way to reduce TRIM commands. The file system does not immediately issues TRIM, it may have merge the information of delete files to reduce TRIM commands, but in our experiments it is not enough, the performance will still drop after large TRIM commands. The device internal hehaviors is a huge issue, too. The firmware design for handling TRIM have great impact of performance like intel and plextor, we can find that when using intel SSD has design not very well.

Bibliography

[1] T. Frankie, G. Hughes, and K. Kreutz-Delgado, "A mathematical model of the trim command in nand-flash ssds," in *Proceedings of the 50th Annual Southeast Regional Conference*, ser. ACM-SE '12. New York, NY, USA: ACM, 2012, pp. 59–64. [Online]. Available: http://doi.acm.org/10.1145/2184512.2184527

[2] T. Frankie, G. Hughes, and K. Kreutz-Delgado, "Solid State Disk Object-Based Storage with Trim Commands," *ArXiv e-prints*, Oct. 2012.

[3] T. Frankie, G. Hughes, and K. Kreutz-Delgado, "Analysis of trim commands on overprovisioning and write amplification in solid state drives," *arXiv preprint arXiv:1208.1794*, 2012.

[4] B. Gu, J. Lee, B. Jung, J. Seo, and H.-J. Shin, "Utilization analysis of trim-enabled nand flash memory," in *Consumer Electronics (ICCE), 2013 IEEE International Conference on*, 2013, pp. 645–646.

[5] G. Kim and D. Shin, "Performance analysis of ssd write using trim in ntfs and ext4," in *Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on*, 2011, pp. 422–423.

[6] C. Hyun, J. Choi, D. Lee, and S. H. Noh, "To trim or not to trim: Judicious triming for solid state drives."

[7] M. Saxena, M. M. Swift, and Y. Zhang, "Flashtier: a lightweight, consistent and durable storage cache," in *Proceedings of the 7th ACM european conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/2168836.2168863

[8] Y.-C. Lee, C.-T. Kuo, and L.-P. Chang, "Design and implementation of a virtual platform of solid-state disks," *Embedded Systems Letters, IEEE*, vol. 4, no. 4, pp. 90–93, 2012.

[9] T. Group. (2010) Scsi commands reference manual. [Online]. Available: http://www.seagate.com/staticfiles/support/disc/manuals/Interface20manuals/100293068c.pdf