SOLID STATE DRIVES

Abdelmhsan Elbandac [1], Abdalhamed Alkawash[2], Anis Elgarduh[3]
[1] **Higher Institue of Agricultural, Techniques Al-khadra,Tarhuna, Libya**
[2] **Dept. of Computer Science, College of Technical Sciences, Derna, Libya**
[3] **Dept. of Computer Science, Omer Al-Mukhtar University, Derna, Libya**
Elbandaca@gmail.com

وسائط التخزين الثابتة

الملخص:

لقد أخذنا ثلاثة أجزاء في هذا البحث لمحاكاة محركات أقراص الحالة الصلبة وكانت كالتالي (I CACH, SSD Wear Leveling Simulator, Wear Leveling).

يتم توضيح مفاهيم تسوية التآكل من خلال إدخال أوجه القصور في ذاكرة Flash NANAD حيث المشكلة مع SSDs هو أن الكتابة الفوقية يمكن أن تقلل من عمرها لأن كل خلية لديها قدر محدود من عمليات المسح والكتابة.

حيث اننا قمنا بتشغيل محاكاة تسوية التآكل في وضعين مختلفين لخلية الذاكرة، MLC، الذي يخزن بتتين في كل خلية، و CLS الذي يخزن بت واحد لكل خلية مع I-CASH، الذي يستخدم مزيجًا ذكيًا من HDD و SSD في نفس الوقت يتم محاكاة MLC و CLS بواسطة C ++. حيث تظهر نتائج العمر الافتراضي لسِعات **SSD** المختلفة مع فئات مختلفة من المستخدمين و تثبت النتائج أنه على عكس محرك الأقراص الثابتة ، يمكن لمحرك الأقراص المزود بذاكرة مصنوعة من مكونات صلبة الاستفادة من زيادة حجم محرك الأقراص وبالتالي زيادة مساحة تسوية التآكل.

الكلمات المفتاحية: *رمز الوسائط التخزين ذو الحالة الثابتة (SSDs) – المحاكاة – توزيع التلف – المخبأ الذكي*

Abstract:

We took three parts in this Paper to simulate SSDs and they were as follows (Wear Leveling, SSD Wear Leveling Simulator, I CACH). The concepts of wear leveling are illustrated by introducing shortcomings in Flash NANAD memory. The problem with SSDs is that overwriting can reduce their life because each cell has a limited amount of erase and write operations. We run a wear leveling simulation in two different memory cell modes, MLC, which stores two bits in each cell, and CLS, which stores one bit per cell. With I-CASH, which uses a smart combination of HDD and SSD at the same time. MLC and CLS are simulated by C++. The results show the life span of different SSD capacities with different categories of users. The results prove that unlike a hard disk drive, a solid state drive can benefit from an increased drive size and thus an increased area of wear settlement.

Keywords: *SSDs, Simulated, Wear Leveling, I CACH*

Introduction

SSD has an advantage over typical magnetic hard disk drives in several aspects, including its low power consumption, data transfer and CPU utilization. The main component of SSD is the Flash NAND memory, which is structured as an array of blocks ranges from 512 KB to 1MB. In the first part of this report, wear-leveling concepts are clarified by introducing the shortcomings of Flash NANAD memory. The problem with SSDs is that the overwrite operation can decrease its lifetime as every cell has limited amount of erase and write operations. Therefore, Wear leveling is considered as a solution for the problem of endurance. Moreover, in this part, erase and write operation and also the needing of garbage collection according to some scientific papers are provided

In the second section, we run the simulation of wear-leveling in two different memory cell mode, MLC, which stores two bits in each cell, and CLS i.e. store one bitfor each cell. Then, we show the results of the lifetime and how the three factors could be prolonged the SSDs lifetime. These three factors are the capacity size, the available space for cold-data and the memory cell mode.

In the third section, we present a scientific paper that suggests using I-CASH, which uses an intelligent combination of HDD and SSD at the same time. The algorithm benefits from cache memory concepts by identifying a specific reference block. I-CASH approach can provide high computation power of multi-core processor. As regards to its performance, three benchmarks are evaluated by running them on difference storage architecture including I-CASH. In the end, the references of our work are included for further reading.

First Part: Wear Leveling

We went through several scientific papers about wear leveling, which are attached as references below. All of them deal with the endurance of the Solis State Drive (SSD), precisely with Flash NAND memory, which is the main concern faced by SSD.

We have started the topic by reviewing research presentations and shedding light on the main components of SSD, which include Flash NAND memory, the controller, and DRAM Buffer. As has been said, SSD has an advantage over typical magnetic hard disk drives in several aspects, including its low power consumption and non-volatile memory. “Flash NAND memory is structured as an array of blocks, with 512 k as the typical size, and each block is sectioned into pages. Wear leveling appears as a solution for the problem of endurance that is encountered by Flash NAND” (Hsieh & Ma,2010). In other way, Flash NAND memory is designed as an array of blocks, typically 512KB in size, and each block is paged. As a solution, the wear leveling of the endurance problem that Flash NAND faces is shown.

There are threeoperations in SSD: read, write, and delete. Read and write operations are done in pages while delete operations are done in blocks. The erase (or delete) operation reduces the lifetime of Flash NAND memory. The write operation can transfer to the erase and program operations in Flash NAND memory. Write and program operations perform the same tasks, but use different names. The problem is related to the write operation. Remember that each block has several pages. Unlike typical HDD, where one canoverwrite existing data, overwriting existing data cannot be done in SSD; a block must beerased first before we can rewrite to it. Each block has a limited number of

times that can be erased. When data is needed to be written to a page and whole pages are full, an erase operation must be performed. When a few blocks reach their limit, the whole drive age will be shortened.

Wear leveling technique

Wear leveling is the method or technique of distributing erase operations in an even way across the whole Flash NAND memory in order to extend the lifetime of the flash by preventing any block from reaching its limit. Wear leveling is conducted by the flash transitional layer (FTL), which is embedded software. Address mapping as well as garbage collection are also performed by FTL. Address mapping maps logical addresses from operation systems to physical addresses in Flash NAND memory. “Most SSDs implement garbage collection that maintain a pool of erased blocks” (Shrestha & Xu, 2010).

Cold vs. Hot Data

Data are classified as hot or cold depending on their update frequency. Data with high update frequency are considered hot, while data with low update frequency is considered cold. Placement data between blocks is very important for wear leveling. A physical block is considered to be old or young depending on its erase count, which is the number of times a block has been erased. Each block is attached by a counter that maintains the erase count. The block with the minimum erase count has the highest probability of storing cold data while the block with the maximum erase count has the highest probability of storing hot data. Typically, wear leveling will transfer cold data to an old block.

A wear leveling scheme can be proactive or passive, or both. The goal of a proactive scheme is to put data in suitable blocks; however, in the case of a passive scheme, a data must be swapped between blocks to avoid early wearing out of the block.

There are three states for each page: clean, valid, and invalid. As we know, each write operation is preceded by an eraser operation. A page is considered to be clean after it has been erased and before it is programmed. That means that a clean page is ready to receive new data. However, a page can be either valid or invalid after programming but before erasing. This depends on whether programmed data is current or not. If it is current, a page is considered to be valid; on the other hand, if data is old, a page is considered to be invalid. When the number of clean pages gets low, garbage collection There are three states for each page: clean, valid, and invalid. As we know, each write operation is preceded by an eraser operation. A page is considered to be clean after it has been erased and before it is programmed. That means that a clean page is ready to receive new data. However, a page can be either valid or invalid after programming but before erasing. This depends on whether programmed data is current or not. If it is current, a page is considered to be valid; on the other hand, if data is old, a page is considered to be invalid. When the number of clean pages gets low, garbage collection is needed. It reclaims the pages that are invalid by erasing them.

Erase Strategies

A significant question that must be asked is which block among the all of the blocks must be chosen to be erased. Two strategies are suggested: random eraser count and lowest erase count. In random erase count, a block is chosen randomly from a set of blocks that are currently invalid. However, in the case of lowest erase count, an invalid

block with the lowest erase count is chosen to be next block to be erased.

Hot-cold swapping aims to equilibrate erase cycles by exchanging hot data in an old block with cold data in a young block to avoid putting pressure on one block. An example of this is operating a system file. A system file is usually located in the same place, without having moved for a long time, except when there is an update. An advantage can be taken by moving an operating system file to an old block that is considered exhausted from a highly used erase operation and having that block (the previous block where the operating system was) reside on the available space for other files that require frequent updates.

The operating system deals with SSD differently than with HDD due to its internal hardware components, which are clearly distinct from SSD and HDD. Data can be requested to be written on a page by an operating system. The operating system is aware of the logical address where data is kept, but it is not aware of the physical address of the storage. In the case of HDD, when data is erased, it is not important for the operating system to inform storage simply because it can overwrite to that location. On the other hand, in the case of SSD, since overwriting is forbidden, storage must be informed. So when an operating system writes to a page that is full, it marks that page as invalid and writes to a new location so the next time that page can be erased, which is performed by garbage collection.

Second Part: SSD Wear-Leveling Simulator

A simulation was written to determine the length of life of an SSD under various stress conditions. Two versions were written. The first was modeling based on a distribution of various sizes of files being written to the SSD in a month period of time. The second was modeling based on typical user write sizes written to the SSD in a month period of time for a variety of user types representing a spectrum of use cases. For each implementation, the span of months of lifetime was measured. It should be noted that the writes considered in this case were “net-hot-writes”, meaning the net-write quantities that are temporarily overwritten and then later erased and rewritten. The SSDs were also pre-filled to certain levels to make simulation results more realistic. This “cold” fill data was assumed to be permanent, which would represent files that are rarely, if ever, overwritten, such as operating system files.

Simulation Parameters

SSD drive sizes ranging from 8GB to 1TB were simulated in multiples of two in terms of drive size up to the maximum. The block size for each SSD drive was assumedto be 1MB. For each drive size, various levels of “cold-fill” data were utilized in increments of 1/8 of the total drive space up to half of the drive space. This was to indicate the dramatic loss of lifetime from overfilling an SSD. SLC and MLC type drives were considered with a parameter controlling the maximum number of individual block writes until failure, which was determined as 100,000 and 10,000, respectively, for the two cell types. For the first simulation implementation, small, medium, and large files were considered based on their sizes and quantities of monthly writes. For the second simulation implementation, four classes of users (home light, home heavy, office, and enterprise) were considered in terms of their monthly write quantities. It should be noted that the values chosen for these write quantities were only for comparative and illustrative purposes and do not represent a broad-spectrum of real-world data. The lifetimes of SSD drives were considered for a

period of 240 months (20 years) at which point it was determined that the SSD would have an almost certain probability of replacement based on technological advancement.

Simulation Operation

The simulation allocates a large enough block of memory to represent the number of writes to each block of the largest potential drive size, which in this case was 1 TB. For the first simulation, monthly writes were performed based on the number and size of each file type. During each write operation, the number of block writes performed on the newly-overwritten cells was checked against the appropriate threshold for the drive's cell technology. If this threshold was exceed, it was determined that the drive had failed and the number of months of lifespan and the parameters considered were then written to an output log. All drive statistics, time passage counters, and other internal counters are then reset and then next set of parameters are initialized. This allows both simulation implementations the ability to generate large cross-sections of data without the need for manual intervention.

Simulation Findings

The figures on the next several pages show the results and findings from executing the simulation implementations. It was necessary to make various parameter adjustments to assure a strong cross-section of results were found. Obvious patterns can be witnessed due to the uniform application of data writes and the consistent increase of drive sizes. The lifetime limitations and benefits of increased drive size are clearly evident in the figures below.

Considerations

Based on the information gathered, it was evident that SSD lifetimes are in fact quite limited, especially compared with traditional magnetic HDD alternatives. The figures in the next several pages illustrate our findings and provide evident indications of SSD limitations. Unlike HDDs, SSDs benefit in performance and lifetime from larger drive capacities. The performance effect is in that new writes do not run into the problem of needing erasure before writing if the drive is large enough to allow for adequate "pre-erased" space, which is a matter of effective garbage collection. SSDs operate on a bit of a contradiction in that it is necessary to minimize write operations to maintain drive writability, which implies writing permanent "cold" files that are frequently accessed while at the same to maximizing available space on the SSD to allow effective wear-leveling. This makes SSDs fundamentally better as specialized caches or a type of "deep" storage than as general-purpose mass storage.

The ideal files to write to an SSD are files that are rarely-written and frequently-accessed. Specifically, application files (executables) and code library files are well-suited to this purpose. This means that SSDs function excellently as "pre-instruction-fetch" caches, meaning that an SSD is capable of quickly serving fetched applications that in-turn are used to serve fetched instructions once in memory. Additionally, "red hot" writes with no level of permanency, such as internet browser cache files, make excellent candidates for SSD storage. Although they are written almost constantly during browsing, these files have such short lifespans that they are quickly erased before the SSD's wear-leveling mechanism comes back around to reuse their space. Thus, while they contribute to write-counts, they are not responsible for the artificial shrinkage of the drive witnessed with files that are permanently or semi-permanently stored. The

simulation results that we gathered demonstrated the proof of concept of SSD lifetimes and gave insight into the types of considerations that need to be taken in designing and deploying SSds. Overall, our Simulation efforts provided excellent insight into the benefits of large-capacity SSDs and the comparative lifetimes of SSDs under variousparameters and usage conditions.

Simulation Results

1. Single Level Cell (SLC).

slcresults.txt

Drive Size (MB)	Fill Percentage	Lifetime (Months)
8192	12.5	8
8192	25	7
8192	37.5	6
8192	50	4
16384	12.5	17
16384	25	14
16384	37.5	12
16384	50	9
32768	12.5	34
32768	25	29
32768	37.5	24
32768	50	19
65536	12.5	69
65536	25	59
65536	37.5	49
65536	50	39
131072	12.5	138
131072	25	118
131072	37.5	98
131072	50	78
262144	12.5	240
262144	25	236
262144	37.5	197
262144	50	157
524288	12.5	240
524288	25	240
524288	37.5	240
524288	50	240
1048576	12.5	240
1048576	25	240
1048576	37.5	240
1048576	50	240

2. Multi Level Cell (MLC).

mlcresults.txt

Drive Size (MB)	Fill Percentage	Lifetime (Months)
8192	12.5	0
8192	25	0
8192	37.5	0
8192	50	0
16384	12.5	1
16384	25	1
16384	37.5	1
16384	50	0
32768	12.5	3
32768	25	2
32768	37.5	2
32768	50	1
65536	12.5	6
65536	25	5
65536	37.5	4
65536	50	3
131072	12.5	13
131072	25	11
131072	37.5	9
131072	50	7
262144	12.5	27
262144	25	23
262144	37.5	19
262144	50	15
524288	12.5	55
524288	25	47
524288	37.5	39
524288	50	31
1048576	12.5	110
1048576	25	94
1048576	37.5	78
1048576	50	63

Results of Single Level Cell (SLC) in Graph:

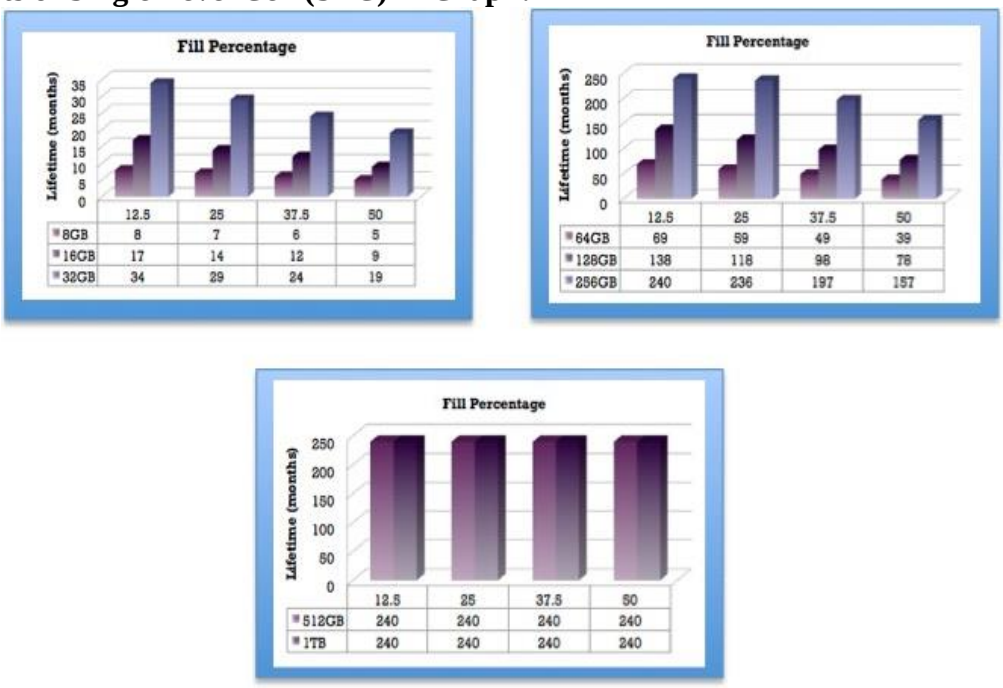

Figure 1. The results of simulation of SLC Memory cell mode

Results of Multi Level Cell (MLC) in Graph:

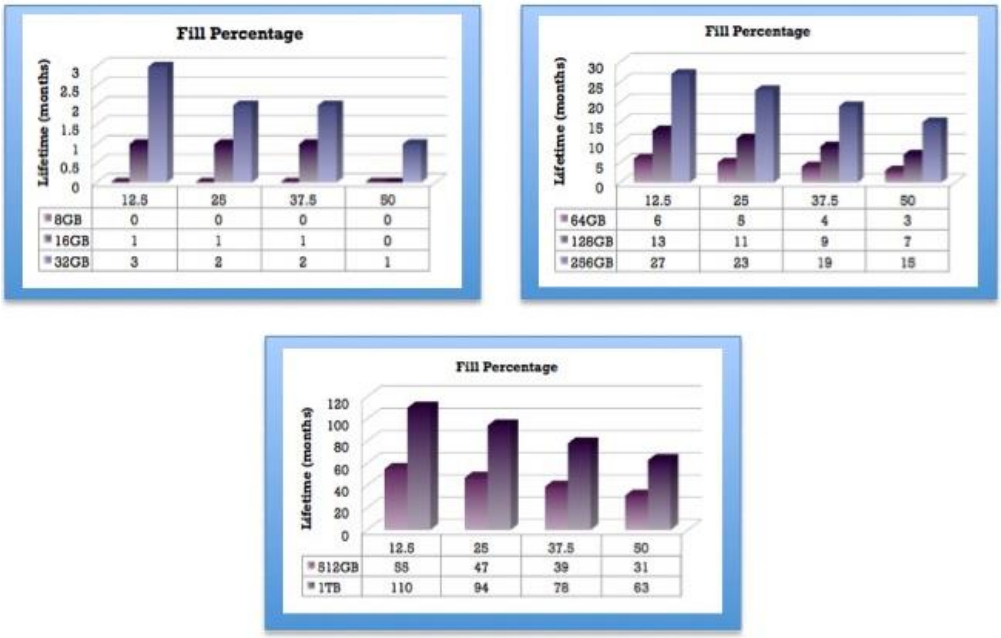

Figure 2 The results of simulation of MLC Memory cell mode

Analysis of SLC and MLC results

The results show that SSD could be benefited from increased drive size and thus increased its lifetime. As we notice in Figure 1 and 2, SSD lifetime would decrease when the cold-data is getting higher percentage (fill percentage). The third observation is if the SSD applies SLC mode, lifetime would be higher from 2-18 times as shown in Figure 3 and this variation is based on its fill percentage (cold-data) and its capacity size.

In the following graph, a comparison of results of Multi Level Cell (MLC) and Single Level Cell (SLC) i.e. the results shows the result of the division of SLC over MLC.

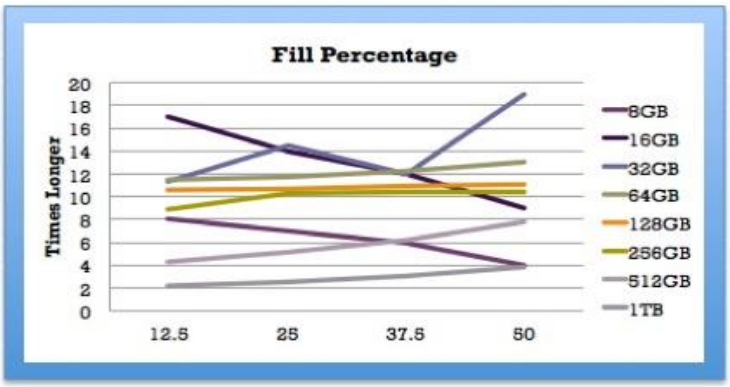

Figure 3 Comparison of the results of SLC and MLC

Third Part: I-CASH

Although the total amount of stored information that a storage device can hold and the CPU processing power have increased rapidly and noticeably, data bandwidth and access times of disk I/O systems have not increased as rapidly. Therefore, there is a big difference between CPU speed and disk I/O system speed. Even though using disk arrays improve I/O throughput, involving mechanical operations increases highly random access latency.

One of the new ideas for enhancing disk I/O performance is to create cooperation between using the advancement of a Solid State Disk (SSD) and multi-core processors. Therefore, I-CASH is used because it creates intelligent cooperation between a Solid State Disk (SSD) and a Hard Disk Drive (HDD). Thus, using I-CASH offers some significant features:

1. Read performance is very fast when using a Solid State Disk (SSD).
2. Sequential and durable write performance when using a Hard Disk Drive (HDD).
3. High computation power of multi-core processor.

Technically, "**I**ntelligently **C**oupled **A**rray of **S**SD and **H**DD (I-CASH) is a new disk I/O architecture composed of an array of a flash memory Solid State disk (SSD) and a Hard Disk Drive (HDD) that are intelligently coupled by a special algorithm" [Yang&Ren,2001]. Architecturally, each storage element in the I-CASH consists of an SSD and an HDD, which are intelligently coupled by an algorithm. SSD stores a block that rarely changes and reads reference blocks, which are stored in SSD. Nonetheless, HDD stores a log of deltas of data block I/Os that are currently accessed with its corresponding reference blocks.

I-CASH Architecture:

The I-CACH architecture includes the hybrid of SSD and HDD coupled by an algorithm. Data could be stored horizontally in SSD and HDD. SSD stores read data, called *reference blocks*, while HDD stores *delta blocks*, a log of deltas, as shown in Figure 1. A delta is the difference between and the "reference block stored in the SSD and the data block of an active disk I/O operation" [Yang & Ren,2001]. In an I/O write, as shown in Figure 1b, I-CASH computes the delta by selecting the corresponding reference block. With regard to I/O read, the combination of the delta and its corresponding reference block would be returned as a data block, as shown in Figure 1c. The deltas are stored in compact form since their size is small, as the data blocks' regularity and applying the content locality and therefore, one HDD operation.

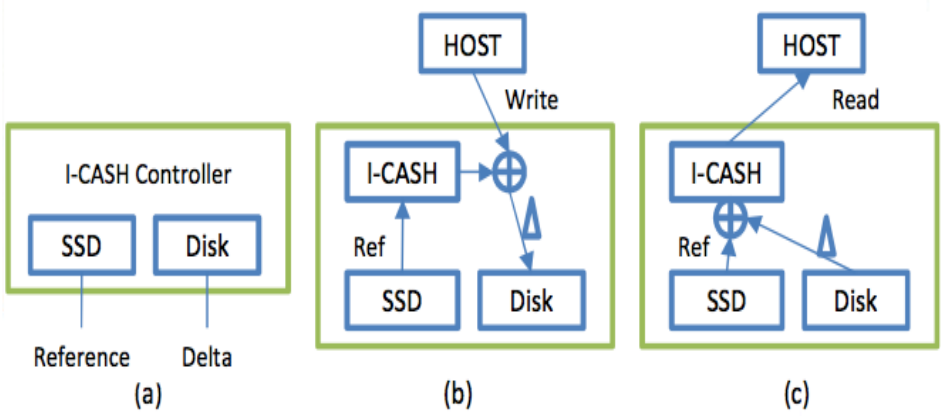

Figure 4. Block diagram of the I-CASH architecture.

results many I/Os. The algorithm here avoids the traditional seek-rotation-transfer I/O operations on HDD, which cost tens of milliseconds, and rather, it involves mainly SSD reads and computations that take tens of microseconds. I-CACH architecture can provide high-speed "read performance of reference blocks stored in SSDs. Moreover, it can pack a large number of small deltas in one delta block stored in HDD"[Yang&Ren,2001]. This delta block can be also be cached in the RAM. Furthermore, the results of the SSD and HDD integration show higher CPU performance.

I-CASH Implementation:

This new technology could be implemented in the following two ways:

Hardware

-CASH architecture is embedded inside the controller board of the Host Bus Adaptor (HBA) or Hard Disk Drive (HDD). As we see in the figure below, the controller board contains the following components:

1. A NAND-Gate Flash SSD that is used to store reference blocks.
2. An embedded processor that performs patches derivation, similarity detection, and combing delta with reference blocks.
3. RAM stores delta and data blocks for current I/O operations.
4. Interfaces that are connected to the host system or HDD.

Thus, from the above explanation, we can state that I-CASH performance is very nice, especially when we compare it with the second method of implementation because I-CASH is implemented inside the disk controller. Nevertheless, the main disadvantage for hardware implementation is that this hardware needs to be built.

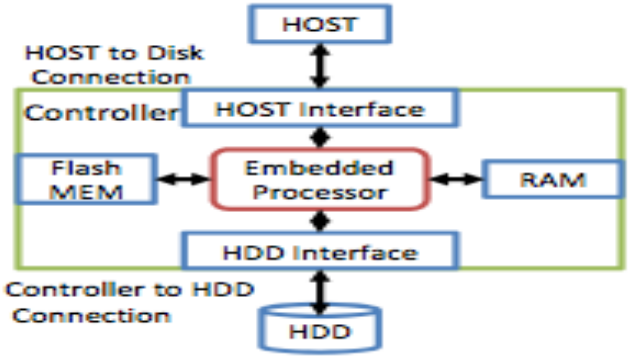

Figure 5 CASH'sHardware

Software:-

In this method of implementation, the entire software is running on the host CPU. In addition, the system RAM contributes as a temporal buffer for delta (patches) with data blocks. This method of implementation affects the system's resources, including the RAM,CPU, and system bus. Furthermore, as we know, some operating systems require specific conditions or special requirement for software.

Reference Blocks Selection:

The main point in I-CASH is how to choose and identify reference blocks. Keeping track of two significant factors could make selecting reference blocks easy. The first factor is to keep track of access frequency. The second factor is to keep track of the content signature of a data block. In order to implement these factors, the following must be done:

- Each block is divided into S sub blocks.
- Each of the S sub blocks has its own a sub signature.
- There is a special two-dimensional array called a Heatmap.

The relationship between the sub blocks and the Heatmap is as follows:

When a block is accessed with a sub block signature, the popularity is increased immediately. Therefore, we can see in the below picture that when a block is accessed with a sub block signature 55, then the popularity in the Heatmap is increased by 1(i++). In addition, when a block is accessed with a sub block signature 00, then the popularity is increased by 1(j++). Therefore, by this method,a reference block can be easily identified (Agrawal,prabhakaram,Ted,John,Mark&Rina,2008).

Hint: The popularity value of a data block is the sum of all its sub block popularity values in the Heatmap.

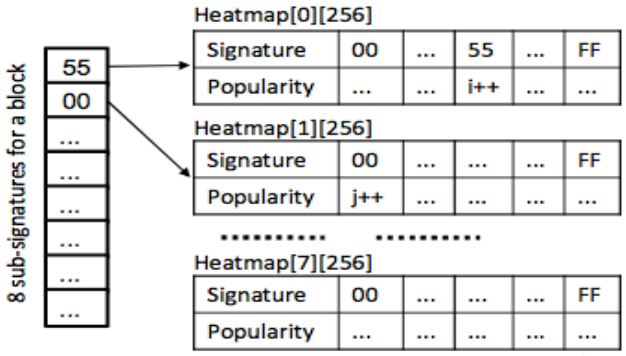

Figure 6 Sub-signatures with Heatmap

Next, we will explain the idea of reference selection and the algorithm of Heatmap. We want to use another example to illustrate this more clearly. Therefore, let us assume that each block is divided into two sub blocks. Therefore, each sub signature has only four possible values. All possible content of the sub blocks are A, B, C, and D and their corresponding signatures are a, b, c, and d, respectively. The Heatmap, then, is as follows:

Table 1 Illustration of Heatmap Array

I/O Sequence	Content	Signature	Heatmap[0] a b c d	Heatmap[1] a b c d
			0 0 0 0	0 0 0 0
LBA1	A B	a b	1 0 0 0	0 1 0 0
LBA2	C D	c d	1 0 1 0	0 1 0 1
LBA3	A D	a d	2 0 1 0	0 1 0 2
LBA4	B D	b d	2 1 1 0	0 1 0 3

We can find the popularity of all blocks, as follows:

Table 2 Results of Popularity

Content	Popularity
A B	2 + 1 =3
C D	1 + 3 =4
A D	2 + 3 =5
B D	1 + 3 =4

Therefore, the most popular block is the data block at address LBA3 with content (A,D) and its popularity is 5. Thus, block (A,D) should be chosen as the reference block.

Performance:

Two servers, a primary server and a workload generator, are interconnected by a gigabit Ethernet switch. The first one is a Dell PowerEdge T410 with 1.8GHz Xeon CPU, 8GB RAM, 160G Seagate SATA drive, and 80GB SLC SSD. The second is a Dell Precision 690 with 1.6 GHz Xeon CPU, 2GB RAM, and 400GB Seagate SATA drive. Both servers have Ubuntu 9.19 64 bit as an operating system with Ubuntu 8.10, Ubuntu 10.04, and Windows 2003 installed on a virtual machine to execute benchmarks. Based on these benchmarks, virtual machine RAM size would range from 128MB to 512MB (Yang&Ren,2001).

The I-CASH storage includes the Fusion-io ioDrive 80GB SLC SSD and Seagate SATA drive160GB HDD. Fusion-io, RAID0, DeDUP, and LRU are installed on the same hardware environment. The Fusion-io is using the Fusion-io io-Drive 80GB SLC without using HDD, while RAID0 will use 4 SATA disks and Linux MD is installed on the virtual machine as a RAID controller. The third one is the data deduplication (DeDup) which is used for identical blocks (i.e., stored on one copy of data in SSD). As to LRU, SSD would be used as an LRU cache above the SATA disk drive. DeDup, LRU, and I-CACH would use the same amount of SSD space (i.e., 10% of the size of data set). The performance is evaluated based on workloads that are available for the research community, as shown in Table 1.

Table 3 Benchmarks used in performance evaluation

Name	Description
SysBench	Database server workload
TPC-C	OLTP benchmark
SPEC sfs	NFS file server

Three benchmarks

The SysBench benchmark, measures the capability of a system to run a database under an intensive load. TPC-C simulates the execution of distributed and online transactions, such as insert, delete, and update, on databases in multiple warehouses. The third benchmark is SPEC-sfs, which measures the performance of an CIFS or NFS file server (e.g., LOOKUP, READ, WRITE, CREATE). In the following table, the characteristics of each benchmark are represented.

Table 4 Characteristics of benchmarks.

	# of Read	# of Write	Avg. Read Len	Avg. Write Len	Data Size	VM RAM
SysBench	619K	236K	6656B	7680B	960MB	256MB
TPC-C	339K	156K	13312B	10752B	1.2GB	256MB
SPEC-sfs	64K	715K	6144B	17408B	10GB	512MB

SysBench is running on the five storage architectures. For I-CACH, LRU, and DeDup, 128MB SSD space is allocated, while RAID0 and FusionIO are stored in HDD. The result of the experiment is shown in the following figure.

Analysis of the benchmarks results

It can be seen that I-CASH is faster than FusionIO although FusionIO uses only SSD while LRU and DeDup have better performance than RAID0 due to the data locality. Also, the figure shows I-CASH is faster than RAID0, LRU, and DeDup by 2.24x, at 9% and 18%, respectively.

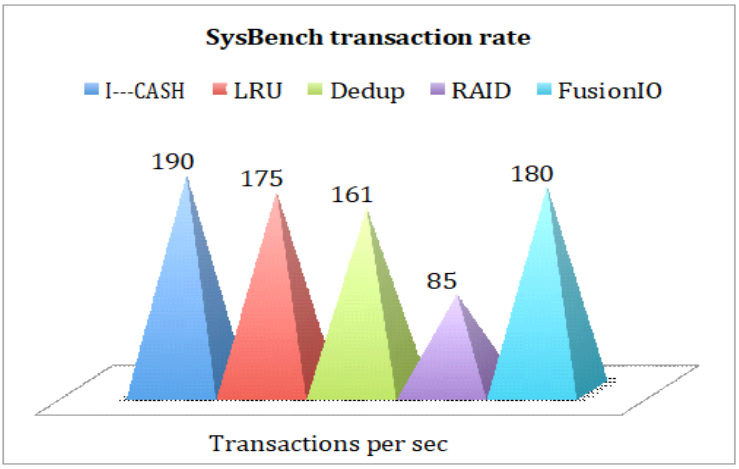

Figure 7 SysBench transaction rate

In Figure 3, the running of SysBench can also be measured by the average response times for read and write I/Os. Figure 3 shows different response times for write and read I/Os in five different architectures. It can be seen that Fusion-io average read time is double of that of I-CASH, while the I-CASH average write time is more than 10 times faster of that of Fusion-io.

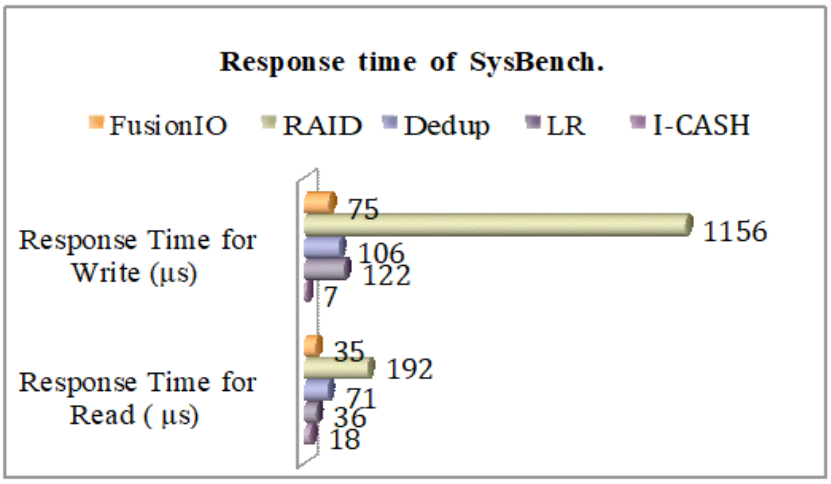

Figure 8. Response time of SysBench

With regard to TPC-C, in Figure 4, I-CASH can perform more transactions per minute than Fusion-io and RAID0 by 14% and 45%, respectively.

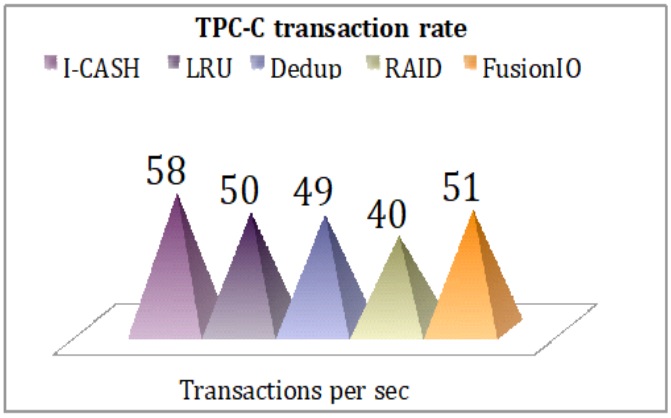

Figure 9. TPC-C transaction rate

In addition, I-CASH has better performance in terms of the application level response time by 64% over Fusion-io and 81% compared to RAID0, as shown in Figure 5.

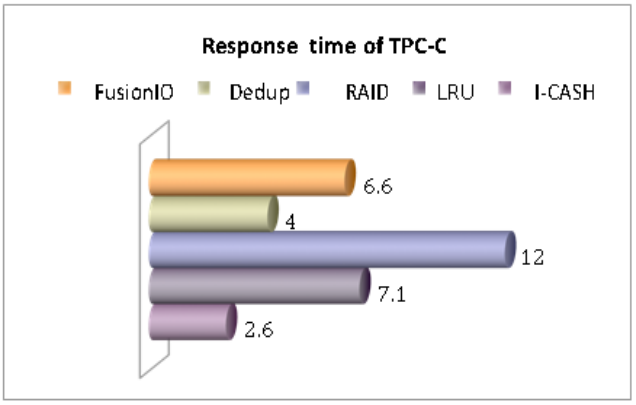

Figure 10. Response time of TPC-C

Figure 6 plots the measured response time while running SPEC-sfs benchmark. I- CASH is configured to use 1GB SSD with 128MB RAM delta buffer. From this figure, we can see that I-CASH performs as well as Fusion-io while using only one-tenth of the SSD space. As shown in Table 4, SPEC-sfs is a write-intensive benchmark. For the DeDup cache, changing a block that is shared by several other, identical blocks results in a new copy of data so that write performance is slowed down. The reduction of the response time of I-CASH over DeDup is 28% because I-CASH is able to exploit the content similarity between the new data and the old data to store only the changed data in small deltas.

With regard to the SPEC-sfs benchmark, although I-CASH uses only one-tenth of SSD, SPEC-sfs performs the same as Fusion-io, as shown in Figure 6. Also, I- CASH has better response time than DeDup and LRU by 28%.

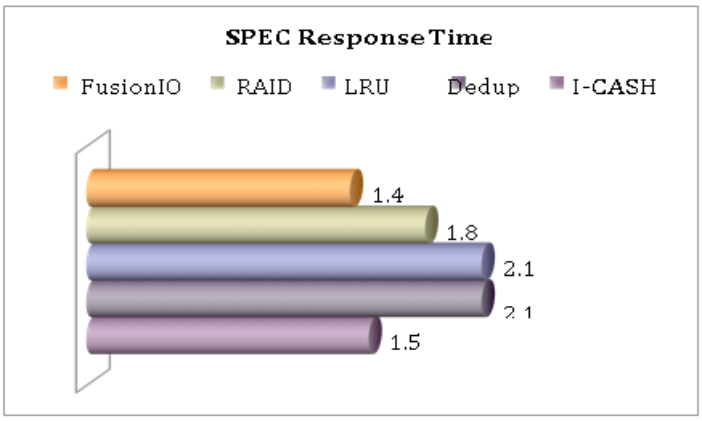

Figure 11. SPEC Response Time

Finally, Figure 7 shows the measurement of the number of write I/Os on SSD for I-CASH, LRU, DeDup, and Fusion-io. The less write I/Os requests there are, the more the SSD lifetime can be prolonged. We can see that for SysBench, I-CASH would reduce write requests compared to Fusion, DeDup, and LRU by 73%, 83%, and 84%, respectively; for the TPC-C benchmark, the result would be 69%, 81%, and 82% compared to Fusion-io, DeDup, and LRU. In the last benchmark, SPEC-sfs, the reduction of write requests when using I-CASH rather than other storage system storage is relatively seldom (i.e., write I/Os requests can be reduced by 11%, 8%, and 7% of Fusion-io, DeDup, and LRU , respectively).

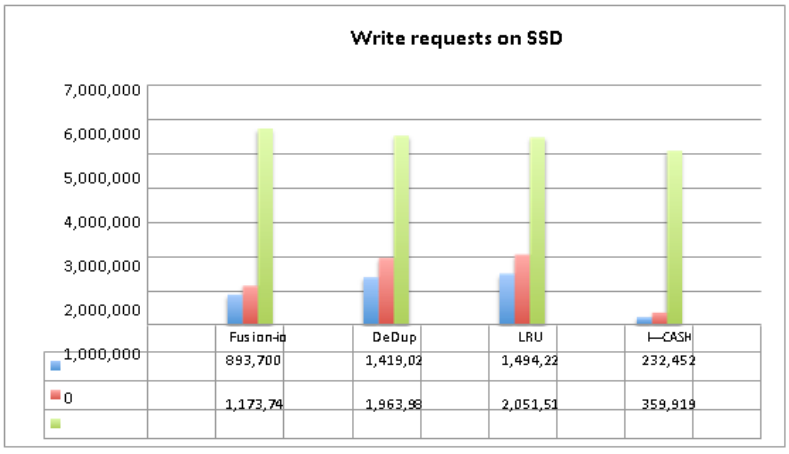

Figure 12. Write requests on SSD

Conclusion

The number of erase operations that performed on a block limits the lifetime of a SSD. So, wear leveling is one of the significant algorithms that are used to prolong the SDD lifetime by distributing erase operations evenly across all blocks.

SSD Wear-leveling concept for both types of memory cell i.e. MLC and CLS, is simulated by C++ The results show the lifetime of different SSD capacity size with different categories of users. The results prove that unlike HDD, SSD can benefit from increased drive size and thus increased wear-leveling space. Another factor can decrease

the SSD lifetime, is the less amount of free space, lifetime becomes less. Moreover, the results show SLC memory cell can increase the life time ranges from 2- 18 times based on the free space of SSDs (hot-data) and its size.

After that we talked about I-CASH exploits the high random access speed of flash memory SSDs. Intelligently, the new disk I/O architecture of I-CASH are coupled an array of SSD and HDD. So, I-CASH tries to take full advantages of SSD and HDD. So, the first advantage is to take the advantage of SSD that is a high random access speed. The second main advantage is high computing power of multi-core processor.

The third significant advantage is reliable and durable write performance of HDD. Essentially, read I/Os are done mostly in SSD and write I/Os are done in HDD. As we mentioned in our report that I-CASH has a special algorithm for similarity detection.

References:

Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., & Panigrahy, R. (2008). Design Tradeoffs for {SSD} Performance. In *2008 USENIX Annual Technical Conference (USENIX ATC 08)*.

Dirik, C., & Jacob, B. (2009). The Performance of PC Solid-- State Disks (SSDs) as a Function of Bandwidth, Concurrency, Device Architecture, and System Organization. In 2009 ISCA Proceedings of the 36th annual international symposium on Computer architecture (PP. 279-289). Association for Computing Machinery.

Hsieh, W. K., & Ma, H. P. (2010, April). Conditional threshold wear-leveling algorithm for multi-channel NAND flash memory. In *Proceedings of 2010 International Symposium on VLSI Design, Automation and Test* (pp. 147-150). IEEE.

Liu, H., Nie, H., Liu, Q., Xie, Q., Li, M., & Xu, H. (2012, July). A Group-Based Hybrid Wear-Leveling Algorithm for Flash Memory Storage Systems. In *2012 Third International Conference on Digital Manufacturing & Automation* (pp. 58-61). IEEE.

Ruemmler, C., & Wilkes, J. (1994). An introduction to disk drive modeling. *Computer*, *27*(3), 17-28.

Shrestha, M., & Xu, L. (2010, December). A quantitative framework for modeling and analyzing flash memory wear leveling algorithms. In *2010 IEEE Globecom Workshops* (pp. 1836-1840). IEEE.

Wang, C., & Wong, W. F. (2012, June). Observational wear leveling: an efficient algorithm for flash memory management. In *Proceedings of the 49th Annual Design Automation Conference* (pp. 235-242).

Yang, Q., & Ren, J. (2011). I-CASH: Intelligently coupled array of SSD and HDD. In 2011 IEEE 17th International Symposium on High Performance Computer Architecture (pp. 278-289). IEEE.