Implications on Database Management Systems with using Solid State Disks

B. Tania L. Fernando

A THISIS SUBMITTED TO
SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

December - 2011

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Mr. Prasanna S. Haddela
Supervisor

Approved for MSc. Research Project:

Dr. Samantha Thelijjagoda
MSc. Research Project Co-ordinator, SLIIT

Approved for MSc:

Dr. Samantha Thelijjagoda
MSc. Programme Co-ordinator, SLIIT

Declaration of Originality

This is to certify that the work is entirely my own and not of any other person, unless explicitly acknowledged (including citation of published and unpublished sources). The work has not previously been submitted in any form to the Sri Lanka Institute of Information Technology or to any other institution for assessment for any other purpose.

Signature

Date

Abstract

Implications on Database Management Systems with using Solid State Disks

Tania Fernando
MSc. in Information Technology
Supervisor: Mr. Prasanna S. Haddela
December 2011

Usage of mobile devices introduced flash memory based storage devices such as flash drives, Secure Digital (SD), Micro SD and Solid State Disks (SSD). These storage media became a viable replacement for Magnetic Disks due to various advantages. But with the different characteristics available in flash technology based storage devices, the data access performance can vary when accessing data. When considering Database Management Systems (DBMS), the most important factor that is considered is the read write costs in terms of performance and energy consumption. Therefore, when it comes to the decision of selecting between Magnetic Disks over SSDs or any other storage device, the read write mechanisms and the speeds should be considered. DBMS components are designed according to the fact that the read write costs being symmetric, with assuming that the only used storage device as the Magnetic disk. SSDs display an asymmetric nature when the reads and write performances are compared. If SSDs are to replace the Magnetic disks, the components of DBMS should be changed in accordance with asymmetric nature of read writes. The Research, "Implications on Database Management Systems with usage of Solid State Disks" focused on to find out the performance variance between the usage of HDD and flash based devices with different scenarios to find out good strategies to access data stored in the devices. After performing a literature review and extensive set of experiments, an algorithm is proposed to select the best performing block size for a given storage device and access pattern.

Acknowledgements

I would like to acknowledge the help and support of the many people whose hard work, guidance, friendship and understanding which were crucial to the finalization of this research.

To my project supervisor Mr. Prasanna S. Haddela, without whose knowledge and assistance this study would not have been successful.

To Mrs. V. Goonathilake and to my father, Terrance Fernando for the assistance given with my writing skills specially with the research papers submitted and published.

To all my family members for all the help, patience, encouragement and the invaluable resources lent to me.

Finally, to my fiancé Thimila Fernando, for supporting and encouraging me to pursue this degree. Without his encouragement, patience and support I would not have finished the research.

Thank you all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Study Background

The usual storage device that is used and also that is known to all is the Magnetic Disk. Since 1980s to 2000, with the introduction of the Personal Computers (PCs), the small form factor of the Magnetic Disks became a need. At the beginning of the 21st century, this target was achieved and the disk diameter was down to one inch [1, 2]. With the introduction of the mobile devices such as pocket PCs, palmtops, and hand held devices, the different Storage Media such as flash drives, Secure Digital (SD), Micro SD and Solid State Disks (SSD) comes in to play. The new Storage Media are mainly based on the flash memory technology. The characteristics of Flash memory are significantly different from Magnetic disks. Unlike the traditional Magnetic Disks with mechanical components, these storage media are based on the semiconductor chips which provide strong technical merits. This leads to no latency in flash memory where the magnetic disks have high latency due to its moving parts. This was promising for the expected need of the small form factor and the access speeds. Flash memory also has low power consumption and provides reliability. When considering SSDs, the benefits include high performance in random accesses and shock resistance. Specially, with using SSDs as a storage it is possible to achieve short startup times and most importantly it is possible to eliminate mechanical overheads such as seek time, rotational delay and spin up delay in the Magnetic Disks [7]. Due to these advantages of the flash memory technology over the Magnetic disks, SSDs has become the most promising substitute for Magnetic Disks.

Due to the reasons of light weight, small form factor, Flash based Storage Media are highly used in mobile devices. These devices are playing a major role in today's work environment. Hence, these devices manage fairly large amounts of

data. These data includes movie files, photos and considerably large databases. Therefore, the storage media used in these devices are important and should be monitored for performance. But unlike the high end servers which are monitored periodically for performance, these devices are not monitored for performance issues. However, these devices are expected to provide high performance, especially with regards to IO response time. Therefore, there is a specific and urgent requirement to analyze these storage devices' performance in detail.

When considering DBMSs, the most important factor that is always considered is the read write costs in terms of the performance and energy consumption. Therefore, when it comes to the decision of selecting between Magnetic Disks over SSDs or any other storage device, the read write mechanisms and the speeds should be considered. Unlike the Magnetic Disks where the read write performance is symmetric, SSDs provides a clear asymmetric nature. In SSDs, reading a page, which is the unit of read write operations, from the flash memory, has very low latency. With contrast to that, writing a page to the flash memory is an order of magnitude slower [8].

Due to these reasons it is clear that there will be differences in accessing data with using HDD against SSD or any flash based device. If a user frequently accesses a small file randomly, there will be a difference with the performance with the storage device used to store this file. The user may try to access a large file randomly or sequentially. This file can be having an index or it might not have an index. Depending on where the file is stored, how it is accessed and the file size, the access performance can vary. But these are not considered with the change of the storage device. Hence, there is a high need to find out good strategy to manage these situations. Therefore, the research aim was set based on this idea. The aim of the research was set to find out the performance variance between the usage of HDD and flash based devices with different scenarios to find out good strategies to access data stored in the devices.

To achieve the research goals, a series of experiments were carried out. Using the results of these experiments it was able to find new knowledge, which was documented as a research paper and it was accepted and published at IEEE International Conference on Industrial and Information Systems (ICIIS 2011). With using the results of these experiments an algorithm can be proposed. This algorithm proposes a method to change the block size of the accessed data with the access pattern used and also the device used for data storage. Throughout the document all these factors are elaborated. The next section comprises of the aim and objectives of the research in detail.

1.2 Aim and Objectives

There are many researches carried out specially to overcome the problem of write requests of the SSDs when working along with databases. These are mostly changes proposed for the components of the SSD [7, 5]. However, there is a limit on the changes that can be performed to the SSDs. This goes to show the need for changes from a database view point concerning different storage mediums. Therefore, there is a need to explore the possibilities of aligning DBMS components along with the characteristics of SSDs. The aim of the research was set to find out the performance variance between the usage of HDD and flash based devices with different scenarios to find out good strategies to access data stored in the devices. After the Literature Review carried out, the focus was narrowed down in to some specific objectives. These objectives were based on the following research questions.

- Is there a possibility in aligning the DBMS components with the characteristics of flash based storage devices so that the performance of the DBMS is increased?
- When the storage device changes, what other factors should be considered when accessing data with regards to algorithms?

With the intension of providing answers to the above mentioned qustions, the following objectives were set.

- Perform a detail study on the architectures of the flash based storage devices and the Hard Disk Drive (HDD).
- Compare performance measurements of storage media. In this context the focus is not just to state differences in the reads writes generally, but to analyze the performances according to the different read write patterns such as sequential reads, sequential writes, random reads and random writes and also to compare the preformances with different block sizes for different devices.
- Investigate possible improvements from the Database perspective so that the performance is increased when accessing data and propose an algorithm to increase data access performance.

For achieving these objectives, an extensive literature review and also various experiments were carried out. The next section contains the structure of the document explaining the contents of the different chapters.

1.3 Dissertation Structure

Chapter 2 contains the literature review that was carried out. Under this section the background for the research is presented along with the related work for the resesarch. The Literature Review is followed by the descriptions of the methodology of the research which is illustrated as two chapters; Chapter 3 and Chapter 4. Chapter 3 "Performance Analysis of Storage Devices", contains a detail description of all the experiments carried out in the research along with the experiment results. Chapter 4 "Algorithm Design", contains the design of the algorithm. This is followed by the chapter 5, which contains the evaluation of the results of the entire research. This chapter also includes the limitations of the findings. Finally, under Chapter 6 the conclusion of the research and the future direction of the research is given. All additional materials are provided under the Appendix and the references are given under the bibliography.

Chapter 2

Literature Review

2.1 Background

The Background study contains descriptions of the study on the Storage Media that were used in the research and also the basic background behind the research objectives.

2.1.1 Hard Disk Drive

The usual storage device that is known to all is the Hard Disk Drive (HDD) or the Magnetic Disk Drive. It is used as the primary storage device for a wide range of applications. The history of Magnetic Disks starts from 1950s. With replacing the punched card systems which relied on batch processing procedures, Ray Johnson implemented a rotating magnetic disk stack that can provide direct individual access to each record. The magnetic head which was used to read data from the tracks of the disk had a unique design. It had very narrow pole tips to maximize density with the head-to-disk spacing of 25 m. Fifty 61-cm (24-in) aluminum disks coated with iron oxide paint were needed to provide the capacity required for the targeted transaction processing applications. The drive, storing 5 MB at 100 bits/in and 20 tracks/in, had to be able to record or retrieve records spread over 240 square feet of surface area in less than a second. The first transaction-processing computer was the RAMAC 305 system designed around the RAMAC 350 disk drive [1].

Since the 1980s to 2000, with the introduction of the Personal Computers (PCs), the small form factor of the Magnetic Disks became a need. At the beginning of the 21st century, this target was achieved and the disk diameter was down to one inch [2]. The mechanical nature of the Magnetic disk affected on

the performance and the data access time depends on the location where the data block resides [3]. With the technological enhancements, the disk capacity has been increased about 2500 times during the period from 1993 to 2003. But, the disk bandwidth and latency has improved only about 140 times and 8 times respectively [5, 4]. Therefore, the Access Density has been declined over the years and with the increasing capacities of the disk drives, it can be decreased even further in the future [5]. The mechanical nature of the Magnetic Disks carries inherited problems such as long latencies in handling random accesses, excessively high power consumption and uncertain reliability [6]. Therefore, there had always been a need to find out mechanisms where these problems can be eliminated. But since these issues are rooted in the mechanical nature of the Magnetic Disks, these are difficult to be solved physically by disks themselves. Hence, the need for a replacement of the Magnetic Disk or the Hard Disk Drive (HDD) from a different technology aroused.

The structure of a Magnetic Disk should be considered in order to understand how it actually works. Magnetic Disk is a mechanical device with rotating platters and moving heads. Figure 2.1 [14], shows a simplified form of the structure of the magnetic disk. Data is stored in the magnetic disk as units of disk blocks and data is read and written to the disk with using this unit. A disk block is a contiguous sequence of bytes. There are one or more platters in a disk. Each platter consists of concentric rings which are known as tracks. Tracks can be recorded on one or both surfaces of the platter (single side or double sided platters). The tracks are divided into sectors. The sector size is a characteristic of the disk and cannot be changed. The size of a disk block can be set when the disk is initialized as a multiple of sector size. Also there is an array of disk heads, one per recorded surface. When one head is positioned over a block the other heads are in identical positions with respect to their platters; which forms a cylinder.

When accessing data, the heads should be directly on top of the data block. Therefore, the heads should be moved to the correct location before transferring data. The total access time of one given data block has many time factors involved. The main three time factors can be given as; Seek time, Rotational delay and Transfer time.

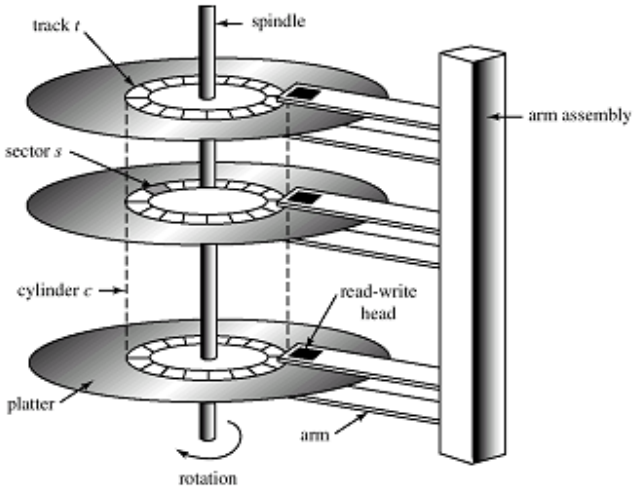

Figure 2.1: Basic structure of a Hard Disk Drive

Seek time is the time taken to move the disk heads to the track where the desired data block is located. Rotational Delay is the waiting time for the desired block to rotate under the disk head. Transfer time is the time to actually read or write the data the block once the head is positioned [3].

Hence, with these factors, the total time taken to access a one given data block depends on the location where the data block resides. The equation on the total access time can be given as below.

$$AccessTime = SeekTime + AverageRotationalDelay + DataTransferTime$$

When considering the operations that are there in the databases, all the operations are affected by this total access time.

The Hard Disk Drive today has improved significantly in the areas of performance and capacity. Along with this, the cost of the Hard Disk Drive has decreased compared to the early disk drives which were specialized and very expensive. The form factor has also improved up to 2.5 inches or smaller. The disk drive is a very significant component in the storage hierarchy, and the future of this device continues to be bright [15].

2.1.2 Flash Memory

The characteristics of Flash memory are significantly different from Magnetic disks. The first difference that can be pointed out is the absence of mechanical components in the flash memory. This leads to no latency in flash memory where the magnetic disks have high latency with its moving parts. When considering the flash memories, there are two types of non-volatile flash memory technologies that can be identified. The two types are NAND flash and NOR flash. NOR flash based solutions are best used for code storage and execution usually in small capacities where NAND flash based solutions are best used for high capacity data storage [12]. The mostly available flash memory type in the market is the NAND flash memory and also in this research, NAND flash memory based storage devices are used.

NAND flash memory can be also classified in to two categories; Single Level Cell (SLC) and Multi-Level Cell (MLC) A SLC flash memory cell stores only one bit, while a MLC flash memory cell can store two bits or even more. For both SLC and MLC NAND, a flash memory package consists of one or more dies (chips). Each die is segmented in to multiple planes and a plane can consist of thousands of blocks and one or two registers of the page size as an I/O buffer. A block typically contains 64 to 128 pages. Each page has 2 KB or 4 KB data and metadata area for storing Error Correcting Codes (ECC) and other information [6].

There are three main operations that are used in flash memories; read, write and erase. The read and write operation are performed in units of pages. Erase operations are performed in units of blocks [6, 8]. The two types of flash memory show differences in performance. Each read operation may take 25μs (SLC NAND) to 60μs (MLC NAND). When considering the write operation, pages in one block must be written sequentially, from the least significant to the most significant page addresses. Each write operation may take 250μs (SLC NAND) to 900μs (MLC NAND) [6]. Table 2.1 show the access time and energy consumption in flash memory when 4 KB data is read, written or erased [13]. From the data given in the Table 2.1, it is evident that the access time and the energy consumption of a write operation in NAND flash memory are 6.3 $\sim$ 6.4 times higher than the read operation. This ratio has become higher in the NOR flash memory. From these details it is clear that the flash memory has asymmetric read write operation characteristics in terms of performance and energy consumption.

Another problem associated with the write operation in the flash memory is, once a page is written the only available way to overwrite the page is to erase the entire block that the page is stored. Once the block is erased, the page can be

replaced. An erase operation can take as long as 3.5μs. As mentioned above, the erase operation can be performed only in units of blocks. Hence, the block is also called, erase block [6]. This makes the write process to slow down in another order of magnitude compared to a single page write. For one given page write, there is a high cost involved because of the erase operation.

Table 2.1: NAND and NOR Characteristics

NAND	Access Time (μs / 4KB)	Read	284.2
		Write	1833.0
		Erase	499.2
	Energy Consumption (μJ / 4KB)	Read	9.4
		Write	59.6
		Erase	16.5
NOR	Access Time (μs / 4KB)	Read	53.8
		Write	14054.4
		Erase	15616.0
	Energy Consumption (μJ / 4KB)	Read	8.6
		Write	3251.2
		Erase	3609.6

When the number of writes increases, the cost involved for the erase operation increases. Erase operations causes another problem since the flash memory has limited erase cycles and it wears out with the increasing number of erase operations. A typical MLC flash memory has around 10,000 erase cycles while a SLC has around 100,000 erase cycles. After wearing out, flash memory cells can no longer store data.

2.1.3 Solid State Disks

Flash memory based SSDs are built on an array of flash memory packages where a one given flash memory package provides around 40 MB / sec bandwidth. In a SSD high bandwidth can be achieved through parallel access [6]. A serial I/O bus connects the flash memory package to a controller. The controller will receive the process requests from the host machine through a connection interface such as SATA. Controller is responsible for issuing commands to transfer data from / to the flash memory array connected to it. Due to the need of large capacity and high performance multi-chip architecture is introduced. There are two types of multi chip configurations; shared control architecture and shared bus architecture. In shared control architecture, all flash chips have their own data path. But in

shared bus architecture, the data bus is shared among the flash chips. Here the shared control architecture shows better performance compared to the shared bus architecture, but it has a very high cost involved. The number of flash chips that can be shared in one data bus is limited because the performance will not increase when too many chips are connected to one data bus. When a chip on a shared bus is supposed to send or receive data, it has to wait until the bus is idle. In order to solve these problems, the number of channels can be increased. This design is what is known as the Multi Channel architecture. In the Multi Channel Architecture rather than using one shared bus which is shared among a number of flash chips, there are several channels introduced. Figure 2.2 [7] shows an example of a multi-channel architecture with 2 channels. In this Architecture, because the bus congestion is distributed over all channels performance is increased.

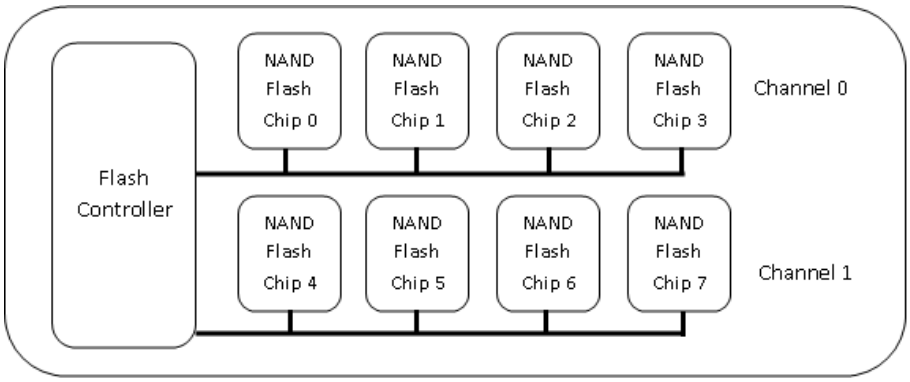

Figure 2.2: Multi Channel Architecture

When considering the architecture of the SSDs, in order to overcome the limitations of the flash disks, there is a management layer introduced. This is an intermediate software layer called Flash Translation Layer (FTL) which is in-charge of managing address translation and garbage collection. In address translation, conversion from a logical sector address to a physical flash memory address is performed. Garbage collection is to move the valid pages another block and to erase blocks to reclaim invalid pages.

Other than the FTL, an SSD contains of its own buffer manager. The main purpose of this is to increase the performance of the SSD. The buffer manager stores data from the host and then writes data to the flash memory afterwards. Buffer manager which act similar to a cache concerns only the write requests than comes to the flash memory. Read requests are need not to be considered as the

read requests are handled by the buffer in the host. The buffer manager in the SSD increases the performance of the write operations. Figure 2.3 [7] shows a summarized view of the SSD architecture.

As shown in the figure 2.3, all requests from the host are processed by the Buffer Manager. All requests are processed by the buffer by replacing its contents until the buffer is full or a flush request is received. Once the data is to be written, the write requests will be sent to the FTL. FTL decides on the destination to write the data and performs the write operations on the NAND flash memory. The FTL will first direct the write operations to the memory space which is already erased. If there are replacements needed, it will perform the erase operation before writing [7].

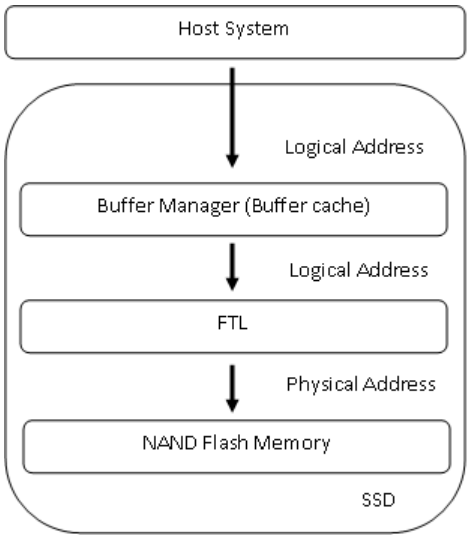

Figure 2.3: SSD Architecture

Currently, there are many SSDs available in the market and the performances as well as the price of these vary depending on many factors. These factors include flash memory type (MLC / SLC), RAM Buffer size and complexity of hardware controller.

2.1.4 Solid State Disks for Databases

From the inception of the Hard Disk Drives or Magnetic Disks, throughout the decades there had been various technology enhancements in the field of computing. With these enhancements, the disk capacity has been increased about 2500 times during the period from 1993 to 2003. But, the disk bandwidth and latency has improved only about 140 times and 8 times respectively [5, 4]. This imbalance

between the disk capacity and the access speed is causing performance issues especially in applications like Database Management Systems. This imbalance between the capacity and the access speed if often measured by a metric called Access Density. Access Density is the ratio of the number of I/O operations per second to the capacity of a disk drive. The access density has been declined in the past few decades and with the increasing capacities of disk drives it can be decreased even further in the future [5].

When considering DBMSs, the time taken for performing a transaction is highly important. Especially in Online Transaction Processing systems (OLTP), increasing the I/O throughput is important in order to improve the performance of the system. But with the available Magnetic Disk technology, the I/O crisis will still be available due to the low access density. Therefore, in order to overcome this problem, there is a need for breaking through the magnetic disk technology and move in to a technology where it promises high I/O throughput. This can be another reason why the SSDs qualify in replacing the Magnetic Disks.

The capacity of a flash memory drive is expected to keep expanding without sacrificing the access speed (or without decreasing the access density) [5]. This is another advantage of the SSDs. In an OLTP system there are large numbers of small I/O operations continuously occurring. The low latency of the flash memory SSD can reduce the average transaction commit time and improve the throughput of transaction processing significantly [4].

As mentioned above, SSD has become the viable replacement for Magnetic Disk due to various reasons. These reasons also include shock resistance, low power consumption and high performance. Another encouragement for replacing the Magnetic Disks with SSDs is because SSDs share a common interface with Magnetic Disks both physically and logically. Most SSDs support the same host interfaces used for Magnetic Disks such as Serial Advanced Technology Attachment (SATA). This avoids the tremendous overhead that can be caused when migrating from Magnetic Disks based systems to SSD based systems. Therefore, it proves again that SSDs can be the most competitive replacement for Magnetic Disks.

But, the special features of the flash technology introduce new challenges to the architecture of computer systems, the design of system software and also to data management technologies. Hence, traditional database architectures and algorithms designed for Magnetic Disk based storage fail to utilize the new flash technology storage media efficiently. The asymmetric nature of the read and write operations in the flash memory bring challenges to the database performance. Without improving the random write performance of the flash memory, it will

not provide the most practical replacement for Magnetic Disks. The main reason behind this is because randomly scattered write requests are very common in the databases when it comes to OLTP systems. Along with this, a need arises to analyze and modify the querying algorithms, indexes, buffer management schemes and transaction processing techniques used in database management systems to get the best performance with flash technology based storage devices.

The DBMS components are designed according to the given fact that the read write costs being symmetric, with assuming that the only used storage device as the Magnetic disk. Due to all the advantages of SSDs and the flash technology, if the SSDs are to replace the Magnetic disks, the components of the DBMS should be changed in accordance with the above mentioned asymmetric nature of read writes. Therefore, the initial focus of this research was to find out the most important components of the DBMS that should be changed according to the SSD characteristics and also to propose the ways and means of the changes that should be made. This is an emerging topic in the area of DBMS research and there are various researches carried out with this idea [10]. Section 2.2 includes some descriptions on the current researches on this topic.

2.2 Related Work

As mentioned in section 2.1.4 there are many research work carried out focusing on improving the SSD performance. These researches are mainly based on the querying algorithms [16], indexes [17], buffer management schemes [8, 18] and transaction processing protocols. All these researches are either ongoing researches or recently completed projects. The First International Workshop on Flash-based Database Systems (Flash DB 2011) was held in April 2011 [10]. This proves that this topic is now becoming important in database systems research.

Keeping aligned with the facts that are given under the section 2.1.4, all researches again and again proves that there should be changes introduced to the Database Management Systems with using SSDs as the storage Media. When it comes to Databases, the buffer manager plays a major role. The buffer manager is responsible in bringing in and out the data pages according to the requests made. It uses a page replacement policy to evict pages when the memory buffer is full. Due to the assymetric nature of the flash memory devices, the energy taken for wrinting data pages will be greater compared to reading. Therefore, as solutions for this problem, there are many algorithms proposed.

FAB (Flash-Aware Buffer) is a buffer management policy designed for portable

media players equipped with flash memory [19]. This article suggests a flash aware buffer management scheme that reduces the number of erase operations by selecting a victim based on its page utilization rather than based on the traditional LRU (Least Recently Used) policy. FAB manages a block-level LRU list. The victim block is the block that contains the largest number of pages.

Another replacement algorithm, CFLRU (Clean First LRU) is a flash aware replacement algorithm for operating systems based on the LRU algorithm [18]. CFLRU addresses the asymmetry of flash I/O by allowing dirty pages to stay in the buffer longer than the clean pages. It splits the LRU list into two parts; working region at the MRU end of the list, clean-first region at the LRU end. The policy evicts clean pages preferentially in the clean-first region until the number of page hits in the working region is preserved in suitable level. Only when clean pages are not present in the clean first region, the dirty page at the tail is selected as victim. The size of the clean-first region is determined by a parameter w called the *window size*.

Multi-Buffer Manager is another buffer replacement algorithm for databases with flash memory [20]. In the research paper, the researches state that the Multi-Buffer Manager outperforms CFLRU. This paper is given as the first paper that is focused specifically on addressing the buffer management problem for logging based databases that run on flash memory. In the algorithm, the novelty is the recognition of different pages with different read write costs. In order to incorporate the different costs of reading and writing pages, the global buffer is divided into set of local buffers where each buffer contains pages of the same cost. LRU-WSR is also a flash aware algorithm [21]. This algorithm is based on LRU and Second chance using only a single list as auxiliary data structure. This idea is to evict clean and cold-dirty pages and keep the hot-dirty pages in the buffer as long as possible. Another algorithm for buffer replacement is REF (Recently Evicted First) which is also based on LRU []. This paper propose a technique that selects the victim page to be evicted from buffer cache considering the recent victim page sent to the flash log buffer.

The CFLU algorithm had been a very important idea which was improved by another group of researchers by eliminating the problems. This algorithm is known as the CFDC (Clean-First Dirty-Clustered) [8]. Same as the CFLRU algorithm, CFDC also has two regions, but they are differently organized. The two regions are: the *working region* for keeping hot pages that are frequently revisited and the *priority region* responsible for optimizing replacement costs by assigning varying priorities to pages. A parameter, priority window, determines the size ratio of the

priority region to the total buffer. Both regions do not bound to any replacement policy. In the generalized two-region scheme of the algorithm, the working region uses the LRU and the priority region assign higher priority to the dirty pages. When a buffer fault occurs, a victim is selected from the priority region to make room for a page currently in the working region. Then the requested page can enter the working region. The dirty queue in the CFDC uses page clustering. To administrate these clusters, a hash table is used with cluster numbers as keys. Upon a buffer fault, if the clean queue is empty, the first page in the lowest priority cluster will be selected as the victim. The CFDC algorithm is considered as the superior to its competitor algorithms [10, 23].

With the asymetric nature of the flash based reads and writes, there is another design proposed called "In page logging (IPL)" for flash memory based database servers [22]. This approach caters for the erase before write characteristic of flash memory by partitioning a block into a data region and a log region. The data region stores data pages and the log region stores update to the data pages in the form of log pages.

Another research area under this topic is query processing. Digest Join is a new join method proposed for join processing on flash disks [24]. This method is a two-phase flash based join processing method that makes use of random reads and reduces writing of intermediate join results [10]. In the first phase projecting the join attributes followed by a join on the projected attributes is done. Then in the second phase the full tuples that satisfy the join in order to produce the final join results are fetched. For this fetching, three heuristic fetching strategies are also proposed by the authors. In Digest Join, the focus had been exploring the possibility of further improving non-index join algorithms by utilizing fast random reads of flash disks.

"Query Processing Techniques for Solid State Drives" is another paper that has considered query processing area. In this paper the investigation had been based on data structures and algorithms that leverage fast random reads to speed up selection, projection and join operations in relational query processing [16]. The focus of the research had been on investigating query processing techniques that improves the performance of complex data analysis queries, which are typical in Business Intelligence (BI) and data warehousing workloads. They also introduce a new joining algorithm; Flash join. This is a general pipelined join algorithm that minimizes accesses to base and intermediate relational data. In the research conclusions, the authors point out that many query execution plans that were not appropriate for HDDs become appropriate for SSDs.

Current file system and device level I/O scheduler design is optimized for rotational hard disk drives. Since SSD has drastically different properties and structure there is a need to rethink the design of some aspects of the file system and scheduler levels of the I/O subsystem. The paper "A New I/O Scheduler for Solid State Devices", addresses this area [25]. In this paper the authors prove that the current I/O scheduler design may not be the ideally suited design for the SSDs. With this fact, the authors have also designed a new I/O Scheduler which utilizes the structure of the SSD.

As discussed above, there is a vast number of research projects started in this area. But, according to the ideas shared at the very first International Workshop on Flash-based Database Systems (Flash DB 2011), there are many unaddressed areas available under this topic. The main area to be addressed is to clearly identify main challenges and issues for flash based enterprise database applications. While SSDs have been adopted as an alternative storage for enterprise database applications, architectures, data structures, and algorithms optimized for such applications should be developed in accordance with the performance objective such as information access speed, energy efficiency, or even endurance of flash devices [10].

Hence, being based on these facts, the research aim was set to find out the performance variance between the usage of HDD and flash based devices with different scenarios to find out good strategies to access data stored in the devices. How this aim was achieved and the experiments that were performed in order to achieve it is given under the next chapter.

Chapter 3

Performance Analysis of Storage Devices

The performance analysis is performed using a series of experiments. In analyzing performance, access time of data needs serious consideration as it plays a major role towards performance. Therefore, experiments were carried out to measure the performance for different access patterns over different storage media and block sizes. Further, the tests were compared to discover performance patterns with regards to access time and block sizes. The selected storage media for the experiments were Hard Drive, Flash Drive (Thumb drive), Micro SD, Memory Stick and Solid State Disk (SSD).

3.1 Performance Measurements

3.1.1 Access Patterns

When accessing a Storage Medium, there are different access patterns available. They are namely; Sequential Reads, Random Reads, Sequential Writes and Random Writes.

Sequential access simply means that the data is read and written to and from the Storage Media in a sequential manner that is, one record after the other in the order that they are stored. Therefore, when we consider the Sequential transfer rate, it is the amount of data, which is read and written from adjacent sectors of the storage media in one second. The measurement for IOs per second for Sequential Access is the number of Input and Output operations that can be performed in one second. Random access is different from this. Here the records can be read or written in any order. This is not the same order as the order that the data

is stored. Therefore, the transfer rate for Random Access can be given as the amount of data that can be read and written from non-adjacent sectors of the storage device in one second. The measurement of IOs per second for Random access can be given as the number of inputs and output operations that can be performed on non-adjacent sectors of the storage device in one second.

All these four access patterns and the measurements are used in this research for the experiments.

3.1.2 Block Size

All the tests were performed for different block sizes. The considered block sizes include; 4kb, 16kb, 64kb, 256kb and 1MB (1024kb). For one given device, for each access pattern, experiments were performed for all these mentioned block sizes.

3.2 Experiments

The experiments can be listed under two main categories. The first set of experiments is based on identifying the I/O performance for different block sizes and different access patterns. For each block size and for each access pattern the experiments were performed on each of the storage devices mentioned above. IOs per second, MBs per second and the Average IO response time was monitored and recorded for analysis. The used block sizes and the access patterns were mentioned under section 3.1

The second set of experiments was performed to validate whether there is a relationship with the varying block size, the device and the file size. For this experiments set, the device performance was tested against different file sizes.

3.3 Test Environment and Setup

The test machine had an Intel[R] Core[TM] Duo processor, 1GB memory and is running Microsoft Windows XP Professional Version 2002 Service Pack 2. Magnetic Disk is connected through IDE interface, and for the experiments a logical partition of 41 MB was used. The flash drive used is a Kingston Data Traveler with 4GB memory capacity. The micro SD is Kingston with 1 GB memory capacity and the memory stick used is a SONY Memory Stick PRO Duo (Magic Gate) with 1 GB memory. The flash drive, micro SD and the Memory stick was connected to the test machine with using USB 2.0. The SSD used was a Toshiba THN-

SNB062GMCJ ATA Device with 62GB capacity. This SSD technology is NAND flash based.

For the first set of experiments, the tests were carried out using an I/O performance monitoring tool; IO Meter (version 2003.12.16). All the storage Media were tested with IO Meter for different read write patterns; 100% Sequential Reads (SR), 100% Sequential Writes (SW), 100% Random Reads (RR) and 100% Random writes (RW). These tests were performed for different block sizes.

For the second set of experiments, the tool used was HD Tune Pro version 4.60. HD Tune Pro is a Software that can be used to measure the device drive's performance which is provided by EFD Software [26].

3.4 Results and Analysis

In this section the results of the two experiment categories are described in detail.

3.4.1 Experiment 1: Analysis of Performance with Different Block Sizes

As mentioned under section 3.2 the first category of tests was performed with changing the block sizes and the access patterns for each of the devices. The experiment plan is given in Table 3.1. As given in the experiments plan all the tests were carried out with the experiment settings mentioned under section 3.3. According to the experiments plan, for each data block size the access pattern is changed and for all the combinations, the experiment is performed for each device.

Table 3.1: Experiments Plan

Data Block Size		4kb				16kb				64kb				256kb				1MB			
Access Patten		SR	RR	SW	RW	SR	RR	SW	RW	SR	RR	SW	RW	SR	RR	SW	RW	SR	RR	SW	RW
Storage Device	HDD	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Flash Drive	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Micro SD	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Memory Stick	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	SSD	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- SR – 100% Sequential Reads
- RR – 100% Random Reads
- SW – 100% Sequential Writes
- RW – 100% Random writes

The results were analyzed comparing the different Storage Media and different block sizes. With these comparison results, it was interesting to see how the storage device I/O performances vary with the block sizes. The results also varied unexpectedly for different read write patterns.

Figure 3.1 shows I/Os per second comparison results for all devices for 100% Sequential Reads. For smaller block sizes, the number of IOs per second in a Hard Disk is higher than the flash drive. But at one given point, number of IOs becomes the same for both devices. This observance is shown again with the comparison on the number of MBs per second. This is given in Figure 3.2. This behavior is more clearly visible if the SSD performance is removed from the diagram. This is shown in the Figure 3.3. When the performance is monitored for all devices, it is clear that SSD out performs all the other devices regardless of the block size. But, when considering only the SSD, the performance grows with the block size. According to Figure 3.2 the best performance for SSD is given at 1MB. According to Figure 3.2, for smaller block sizes, the Hard Drive performance is higher than the flash drive. But when the block size increases, the flash drive performs well. Up to 64kb, the

performance of flash drive increases, but, after that, with the block size there is no significant development in performance. The best performance of flash drive is shown at 64kb of block size and the HDD gives its best performance at 16kb.

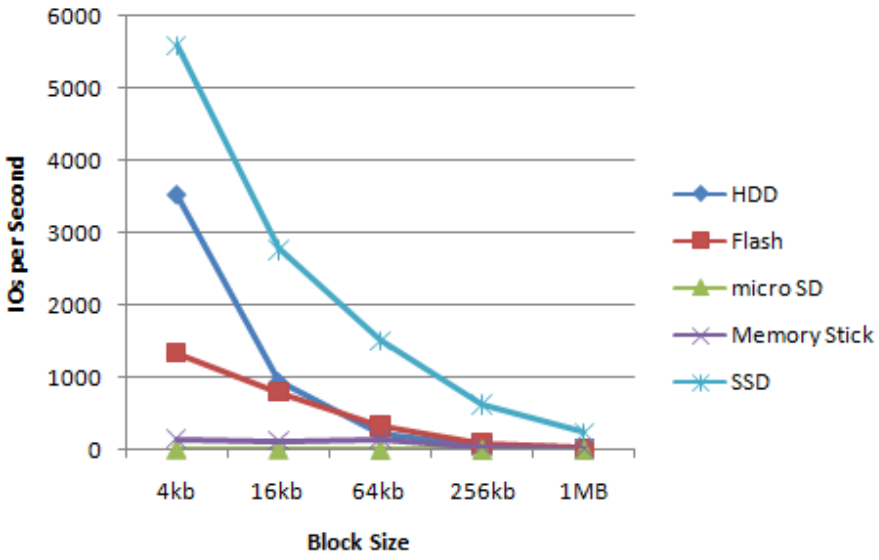

Figure 3.1: IOs per second - Comparison of all devices for 100% Sequential Reads

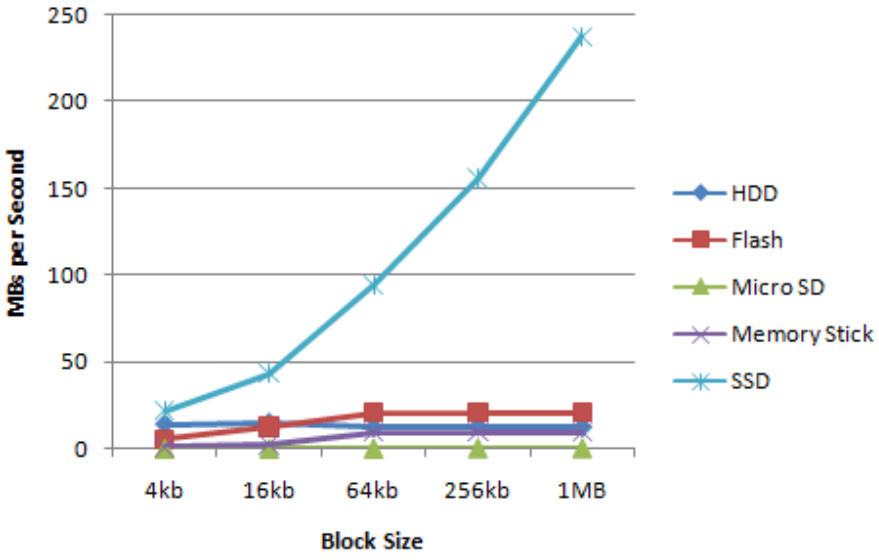

Figure 3.2: MBs per second - Comparison of all devices for 100% Sequential Reads

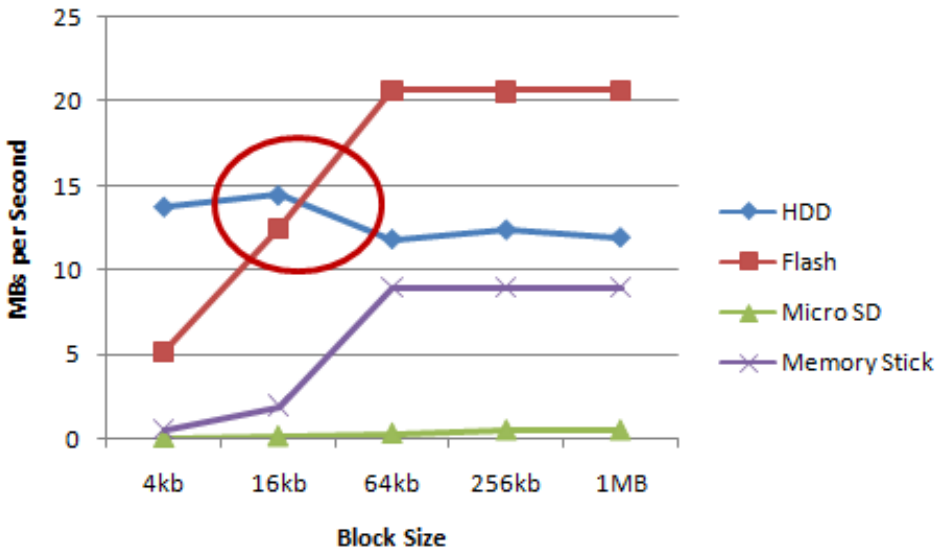

Figure 3.3: MBs per second - Comparison of all devices (without SSD) for 100% Sequential Reads

These results significantly differ in Random Reads. Figure 3.4 shows the comparison graph for IOs per second for Random Reads and Figure 3.5 is the graph for MBs per second. For Random Reads, the highest performance is given by the SSD. The flash drive also performs better than the HDD. Even for smaller block sizes, flash drive has comparatively good performance for Random reads. It improves with the block size. The memory stick also performs comparatively higher than the Hard Drive for smaller block sizes. When considering the SSD, for all block sizes, it is better than the other devices. At small block sizes, the performance is comparatively low. But with the increasing block size, the performance also increases. The best performance of the SSD is reached at a block size of 1MB. For HDD the best performance size is also 1MB. For flash drives, it is 64kb.

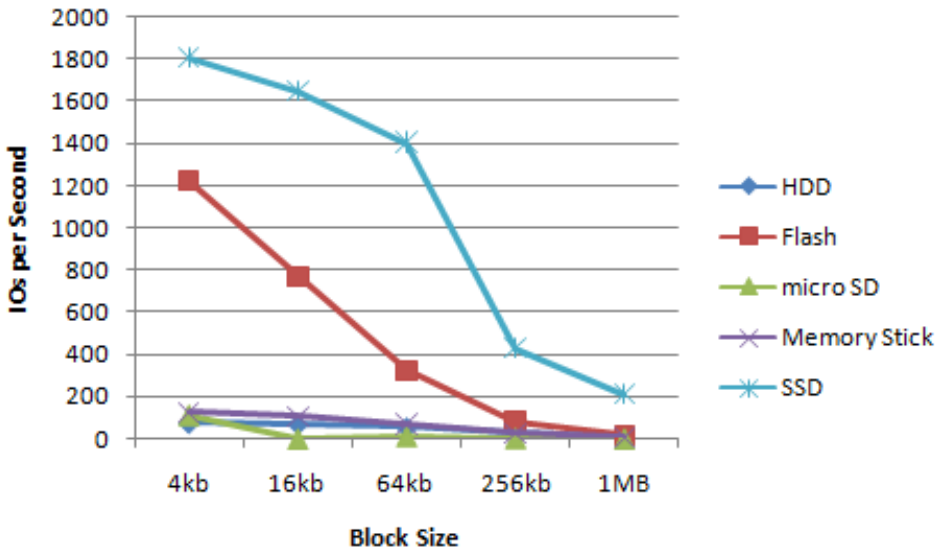

Figure 3.4: IOs per second - Comparison of all devices for 100% Random Reads

All the test results given above are for read performance. The results for write performance are comparatively different from the read performances. This is specially seen when comparing Hard Drive with the flash drive and SSD. Figure 3.6 is the IOs per second comparison for Sequential Write performance. In Sequential Writes, there is no significant difference between the high block sizes for HDD and flash drive. But for smaller block sizes up to 256kb, the performance of the Hard Drive is better. This difference is clearly seen in the comparison chart for MBs per second, which is given in Figure 3.7. There it is clear that the highest performance is from the Hard Drive when compared with the flash drive. But, with the SSD, the performance of the HDD is comparatively low for higher block sizes.

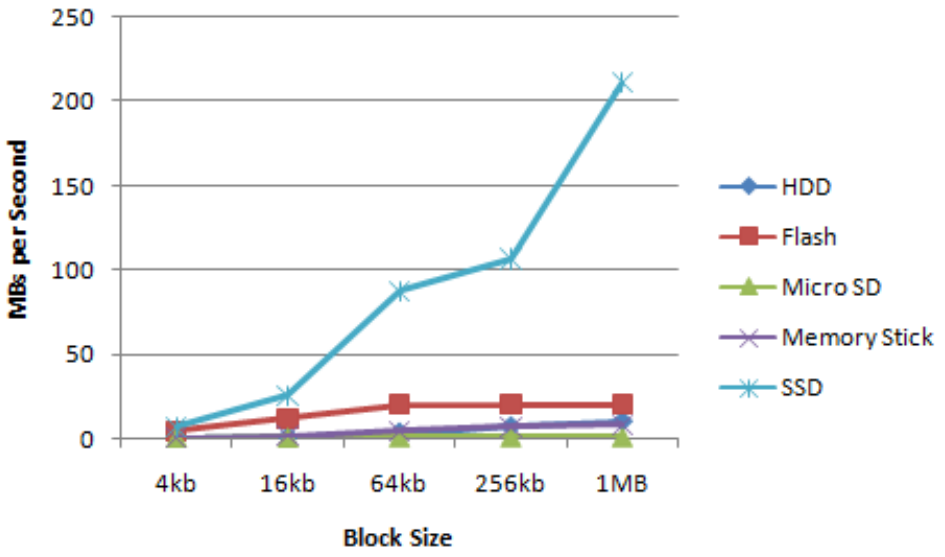

Figure 3.5: MBs per second - Comparison of all devices for 100% Random Reads

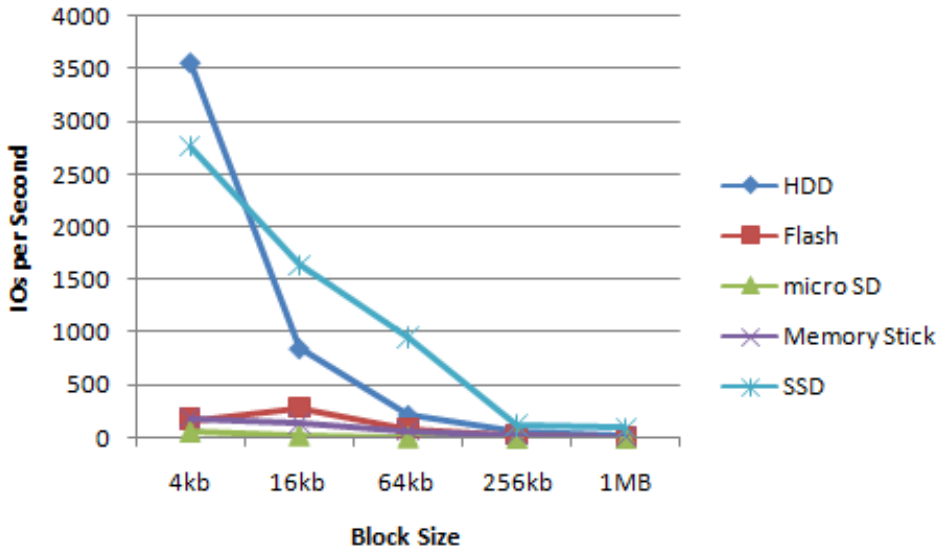

Figure 3.6: IOs per second - Comparison of all devices for 100% Sequential Writes

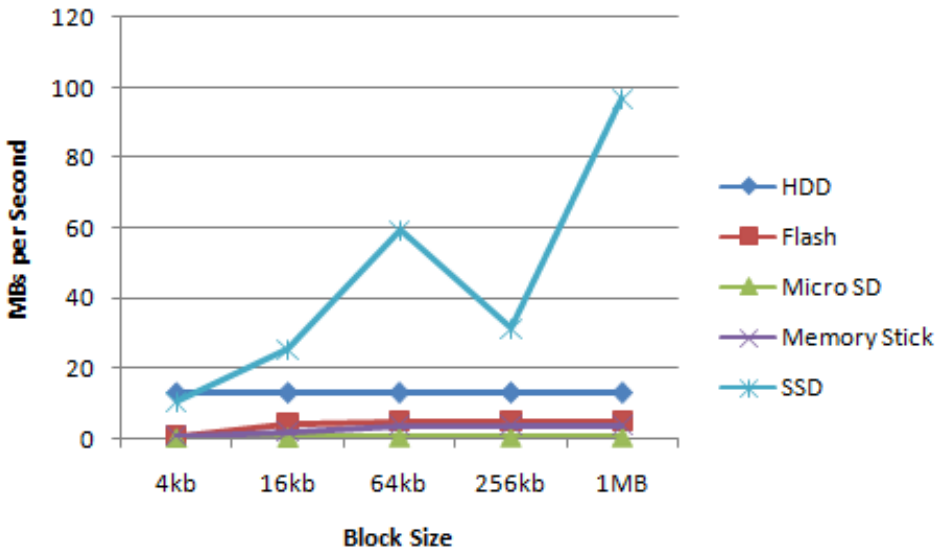

Figure 3.7: MBs per second - Comparison of all devices for 100% Sequential Writes

When referring to the Figure 3.6 and also Figure 3.7 it is clearly shown that at one given point, the performance of the SSD and the HDD becomes the same. At lower block sizes, HDD out performs the SSD and at higher block sizes, the SSD out performs the HDD. Out of all the devices, the performance of the micro SD is very low. Considering the results, the best performance block sizes can be identified as follows. For HDD, the best block size will be 4kb. Flash drive and SSD will be 16kb and 1MB respectively.

For Random Writes the same tests were performed. Figure 3.8 shows the comparison for IOs per second and Figure 3.9 is the comparison for MBs per second over the Block sizes.

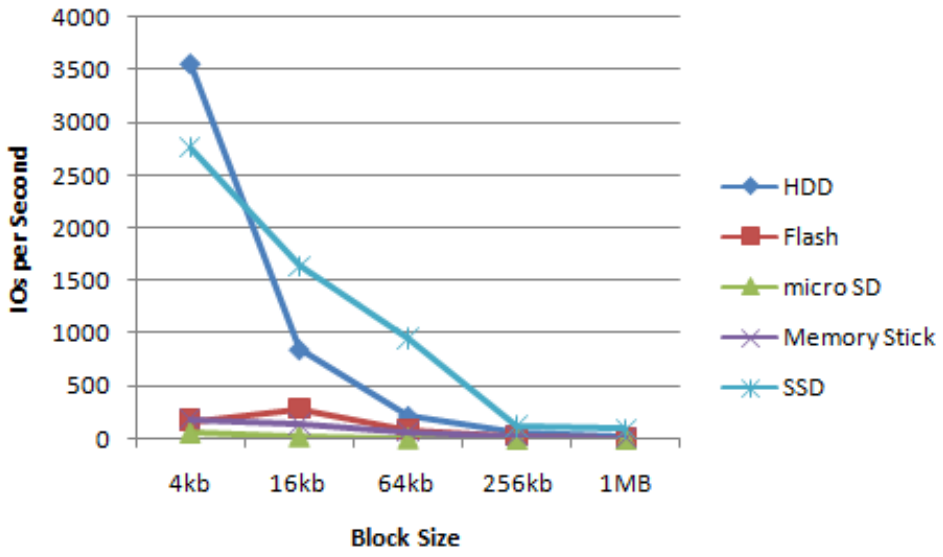

Figure 3.8: IOs per second - Comparison of all devices for 100% Random Writes

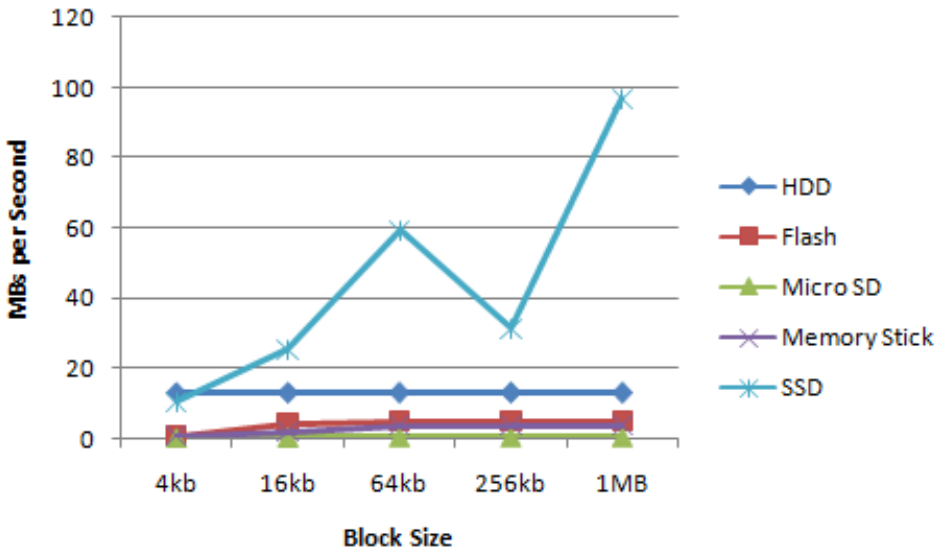

Figure 3.9: MBs per second - Comparison of all devices for 100% Random Writes

For Random Writes, when comparing with Micro SD and the Memory Stick, the flash drive shows comparatively better performance for large block sizes. According to the results for Random Writes, the best performance is shown by the SSD. HDD also have comparatively good performance. For the HDD, the realized number of

IOs per second decreases with the block size. The best performance of the HDD is given at a block size of 256kb. For the Flash Drive, the best block size is 1MB. Same as the Flash Drive, the best performance for SSD is given at 1MB.

The summary for the smallest and the largest block sizes, 4kb and 1 MB respectively can be given as shown in Figure 3.10 and Figure 3.11.

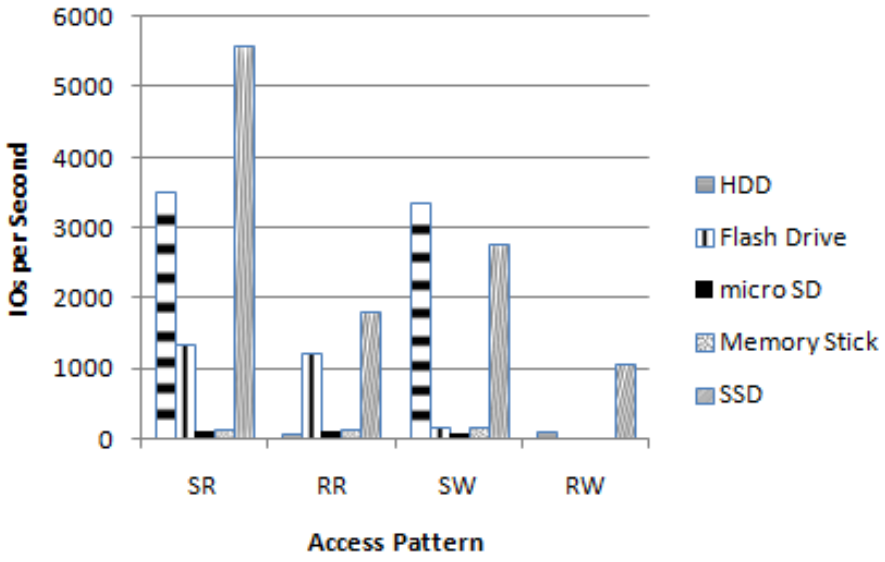

Figure 3.10: IO performance for all access patterns for block size 4kb

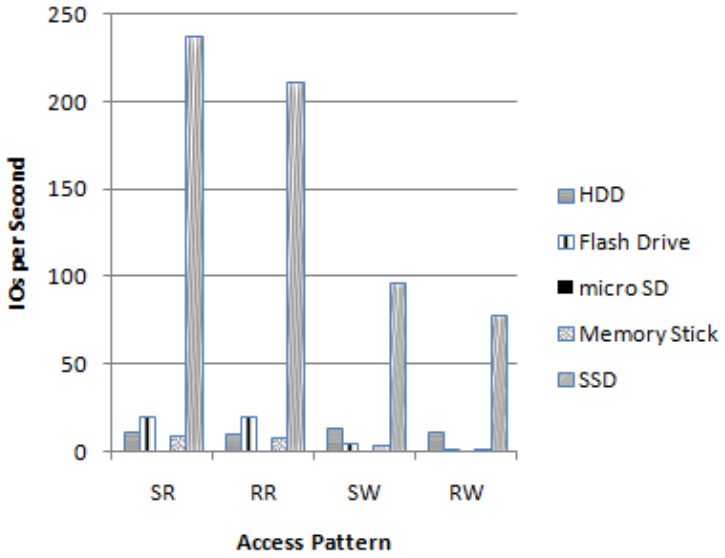

Figure 3.11: IO performance for all access patterns for block size 1MB

If the SSD is not considered, for small block sizes, the best IO performance is given by the Hard Drive except for the Random Reads, where the flash drive performs better than the Hard Drive. For large block sizes, Read performance is much better in flash drives compared to the read performance of the Hard Drive. But the write performance of the flash drive is low. The summary of the best performance without considering the SSD is given in Table 3.2.

With the SSD considered in the analysis, SSD out performs all other devices in all access pattern and the block sizes except for the Sequential Writes at 4kb block size. As per the results of the analysis, the best performing Storage Media for different block sizes and for different access patterns can be identified and it is given in Table 3.3. Here in Table 3.3 all the devices used in the experiments have been considered. If comparison is performed only for the flash drive, micro SD and the Memory stick, the flash drive has the best IO performance compared to the other two. But the Memory stick also has significantly high performance when compared with the micro SD, which shows a similar kind of performance for all access patterns.

Table 3.2: Best Performance Observation (Without SSD)

Block Size	Access Pattern			
	SR	RR	SW	RW
4kb	Hard Drive	Flash Drive	Hard Drive	Hard Drive
16kb	Hard Drive	Flash Drive	Hard Drive	Hard Drive
64kb	Flash Drive	Flash Drive	Hard Drive	Hard Drive
256kb	Flash Drive	Flash Drive	Hard Drive	Hard Drive
1MB	Flash Drive	Flash Drive	Hard Drive	Hard Drive

Table 3.3: Best Performance Observation

Block Size	Access Pattern			
	SR	RR	SW	RW
4kb	SSD	SSD	Hard Drive	SSD
16kb	SSD	SSD	SSD	SSD
64kb	SSD	SSD	SSD	SSD
256kb	SSD	SSD	SSD	SSD
1MB	SSD	SSD	SSD	SSD

With these facts, it is clearly evident that there is a need to rethink about the storage Media that are used for different applications. Hence, from these results, can arrive at a conclusion such that the access performance of a Storage Media depends on the data block size and also the access pattern. As such, to obtain the best performance out from a device, the data block size should be changed depending on the access pattern and the application that is used.

The Experiment 1 results has been recorded and presented as a novel research finding and is accepted and published by 6th IEEE International Conference on Industrial and Information Systems (ICIIS 2011) [11]. The research paper is available under Appendix 6. This result has also being accepted and was scheduled to be presented as a poster presentation at eWorld Forum 2011 organized by Department of Information Technology Ministry of Communications and IT, Government of India which was held in New Delhi India. Also, this result was accepted in National IT Conference 2011 which was organized by Computer Society of Sri Lanka (CSSL).

3.4.2 Experiment 2: Analysis of Performance with Different File Sizes

The second category of experiments was carried out to discover whether there is a relationship between the block sizes and the file sizes with the storage media used. For this set of experiments, the devices used are; HDD, Flash Drive, Memory Stick and Micro SD. This experiment was carried out by using a Performance monitoring tool; HD Tune Pro. For each device the test was performed with changing the file size that is accessed. The performance was monitored for the same file size with different block sizes. As example, the file size of 512MB is read and written using various block sizes for on particular device. The block sizes considered varied from 0.5kb to 1024kb. After performing experiments, the results obtained were analyzed and the best block size for a given file size was selected. The final result set contains the best block size for a given file size. This result is obtained for each device that are used in the experiment. This experiment is carried out for both reads and writes and the final set of values are mapped in to a graph for clear understanding. The results for reads is given in the Figure 3.12 and the results for writes is given in the Figure 3.13.

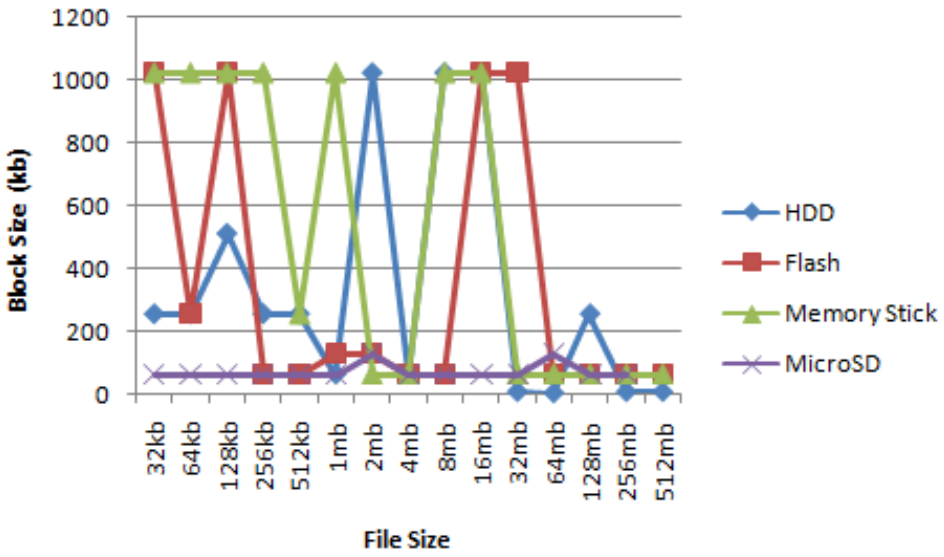

Figure 3.12: Performance analysis for Reads - File size with Block size

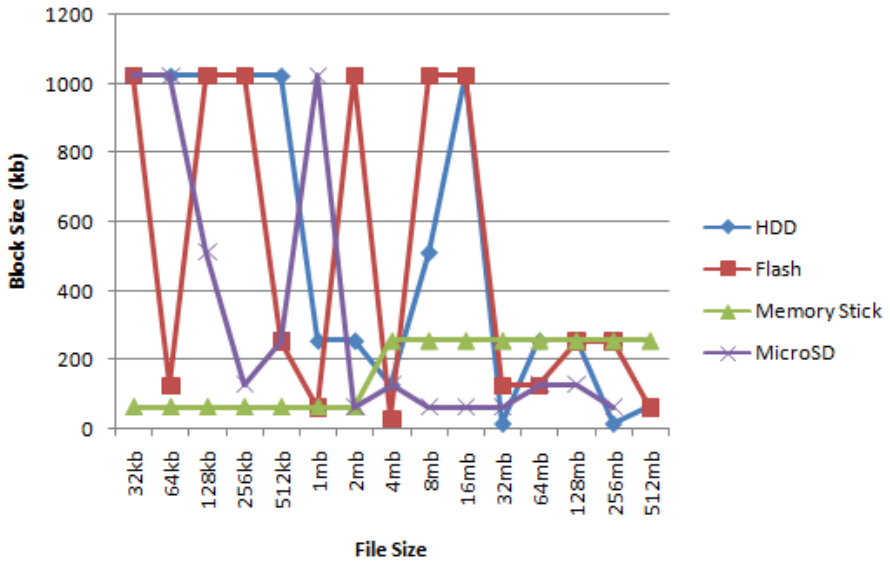

Figure 3.13: Performance analysis for Writes - File size with Block size

When looking at the experiment results for reads, it is clear that there is a pattern in the change of the block size with the file size. To make the pattern clearer, a graph can be drawn for the average of block sizes for all devices. This is given in the Figure 3.14.

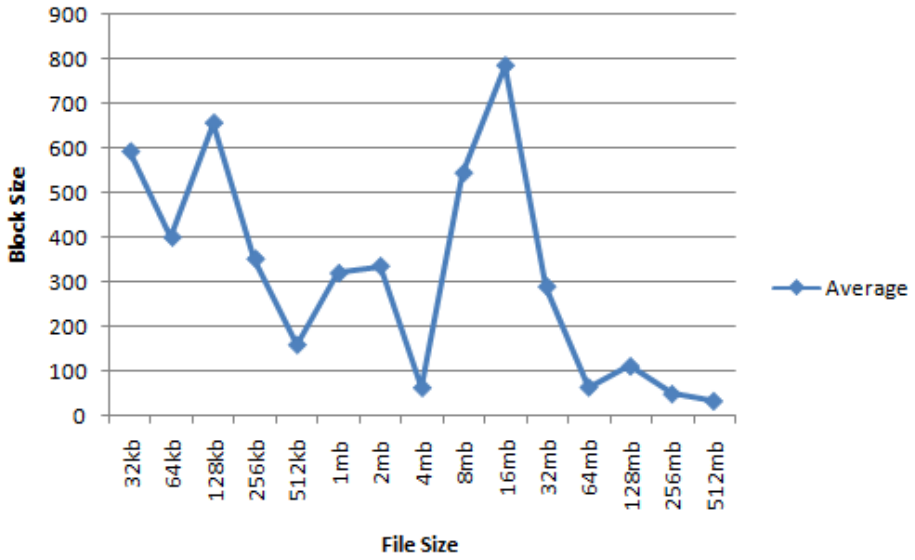

Figure 3.14: Average Performance analysis for Reads - File size with Block size

From the pattern in the Figure 3.14, it is clear that all device performance can vary with the file size and the block size. Therefore, the file size should also be considered when changing the block size. But when looking at the writes, which is given in the Figure 3.13, there is no particular pattern that can be identified. Therefore, it is not possible to come up with a average pattern for the writes.

With the experiment 2, it was clear that the file size should be considered in the performance of a device, especially for the reads. But there is a limitation involved with this set of experiments. The access pattern of the experiment was not defined. This is due to a limitation in the tool that was used. Hence, the above given results were not considered when designing the algorithm which is explained in the next chapter. But, this can be an opening for a new research since it is clear that the file size also should be considered in the performance of devices.

Chapter 4

Algorithm Design

As illustrated under Chapter 3, there are two types of experiments that are carried out in the research. From the results that are obtained from these experiments it was clear that there is a need to change the block size when accessing data from a storage device. From all the experiments that are performed, it was possible to identify the best block sizes for different access patterns and for different block sizes. With these information, an algorithm can be proposed for the best block size selection for a given storage device. The Best Storage Block size Selection Algorithm is as given in Algorithm 1 under section 4.1.

In the Algorithm 1, the storage device can be either a Hard Disk Drive which is denoted as HDD in the algorithm, a Flash Drive which is denoted as FD in the algorithm and a Solid State Drive which is denoted as SSD. AP refers to the Access Pattern and the access pattern given is as below.

- SR – 100% Sequential Reads
- RR – 100% Random Reads
- SW – 100% Sequential Writes
- RW – 100% Random writes

The block sizes are selected from a range of block sizes from 4kb to 1024kb which is explicitly explained under Chapter 3. The block sizes used in the algorithm are best performance for that given access pattern. A detail description of the algorithm with referring to each line of code is given under the next section.

4.1 Best Storage Block size Selection Algorithm

The experiment results from the analysis of performance with different block sizes the algorithm is proposed. The experiment results from the analysis of performance with different file sizes is not considered for the algorithm development due to some limitations which are given under section 5.2. This algorithm can be used for the best block size selection for a given storage device.

Algorithm 1 Best Storage Block size Selection Algorithm

```
1:  if (storageDevice == HDD) then
2:          if (AP == SR) then
3:                  SETBlockSize = 64
4:          else if (AP == RR) then
5:                  SETBlockSize = 1024
6:          else if (AP == SW) then
7:                  SETBlockSize = 4
8:          else if (AP == RW) then
9:                  SETBlockSize = 256
10:         end if
11: else if (storageDevice == FD) then
12:         if ((AP == SR) OR (AP == RR)) then
13:                 SETBlockSize = 64
14:         else if (AP == SW) then
15:                 SETBlockSize = 16
16:         else if (AP == RW) then
17:                 SETBlockSize = 1024
18:         end if
19: else if (storageDevice == SSD) then
20:         SETBlockSize = 1024
21: end if
```

In the algorithm 1 the *storageDevice* is a variable that refers to a storage device. This can be a Hard Disk Drive (HDD), Flash Drive (FD) or Solid State Drive (SSD). In the line numbers 1, 11 and 19 the selection of the storage device is performed by using conditional statements (if conditions). For each condition the algorithm can be explained as below.

Scenario 1 : If the data storage used is Hard Disk Drive and the access pattern is Sequential Reads.

- Line 1 : Condition checking for the storage device
  This is the first level of condition checking in the algorithm. Here the condition will be true if the storage device selected is a Hard Disk Drive. When

the condition is true, the control of the algorithm will be at the line number 2 where the second level of condition checking starts.

- Line 2 : Condition check for the access pattern
  This is the start of the second level of condition checking inside the first storage device selection. In line number 2, the condition checks whether the access pattern is sequential read. If the access pattern is sequential read, the condition will become true and the control of the algorithm is transferred to the line number 3.

- Line 3: Change block size
  In line number 3, the block size of the data will be changed to 64. This value is the best value resulted for sequential reads for Hard Disk Drive.

Once this step is over, the algorithm ends for this scenario.

Scenario 2 : If the data storage used is Hard Disk Drive and the access pattern is Random Reads.

- Line 1 : Condition checking for the storage device
  The condition will be true since the storage device selected is a Hard Disk Drive. When the condition is true, the control of the algorithm will be at the line number 2 where the second level of condition checking starts.

- Line 2 : 1st Condition check for the access pattern
  In line number 2, the condition checks whether the access pattern is sequential read. But according to the scenario, the access pattern that is considered is random read. Therefore, condition in line number 2 becomes false. Then the control of the algorithm is directed to the next condition check in line number 4.

- Line 4 : 2nd Condition check for access pattern
  In line number 4, again the access pattern is checked. If the access pattern is random reads the condition check will be true. According to the scenario, the condition will become true and the control will be transferred to the line number 5.

- Line 5: Change block size
  In line number 5, the block size of the data will be changed to 1024. This value is the best value resulted for random reads for Hard Disk Drive.

Once this step is over, the algorithm ends for this scenario.

Scenario 3 : If the data storage used is Hard Disk Drive and the access pattern is Sequential Writes.

- Line 1 : Condition checking for the storage device
  The condition will be true since the storage device selected is a Hard Disk Drive. When the condition is true, the control of the algorithm will be at the line number 2 where the second level of condition checking starts.

- Line 2 : 1st Condition check for the access pattern
  In line number 2, the condition checks whether the access pattern is sequential read. According to the scenario, since the access pattern considered is Sequential Writes, condition in line number 2 becomes false. Then the control of the algorithm is directed to the next condition check in line number 4.

- Line 4 : 2nd Condition check for access pattern
  In line number 4, again the access pattern is checked. If the access pattern is random reads the condition check will be true. But according to the scenario, this condition will also become false and the control will be transferred to the line number 6.

- Line 6 : 3rd Condition check for access pattern
  In line number 6, again the access pattern is checked. Since the access pattern is Sequential Writes the condition check will be true. Therefore, the control will be transferred to the line number 7.

- Line 7: Change block size
  In line number 7, the block size of the data will be changed to 4. This value is the best value resulted for Sequential Writes for Hard Disk Drive.

Once this step is over, the algorithm ends for this scenario.

Scenario 4 : If the data storage used is Hard Disk Drive and the access pattern is Random Writes.

- Line 1 : Condition checking for the storage device
  The condition will be true since the storage device selected is a Hard Disk Drive. When the condition is true, the control of the algorithm will be at the line number 2 where the second level of condition checking starts.

- Line 2 : 1st Condition check for the access pattern
  In line number 2, the condition checks whether the access pattern is sequential read. According to the scenario, since the access pattern considered is Random Writes, condition in line number 2 becomes false. Then the control of the algorithm is directed to the next condition check in line number 4.

- Line 4 : 2nd Condition check for access pattern
  In line number 4, again the access pattern is checked. If the access pattern is random reads the condition check will be true. But according to the scenario, this condition will become false and the control will be transferred to the line number 6.

- Line 6 : 3rd Condition check for access pattern
  In line number 6, again the access pattern is checked. Since the access pattern is not Random Writes the condition check will be false. Then the control will be transferred to the line number 8.

- Line 8 : 4th Condition check for access pattern
  In line number 8 the condition checks whether the access pattern is Random Write. This condition becomes true in this scenario and the control will be transferred to the line number 9.

- Line 9: Change block size
  In line number 9, the block size of the data will be changed to 256. This value is the best value resulted for random writes for Hard Disk Drive.

Once this step is over, the algorithm ends for this scenario. With this, the access pattern check for the Hard Disk Drive ends. The above mention four scenarios covers all the possible options available in the algorithm for Hard Disk Drives. The next condition in the first level of condition checking is the check for Flash Disk. The scenarios listed below will contain the description on the lines of the algorithms which are associated with the Flash Disk Drive.

Scenario 5 : If the data storage used is Flash Disk and the access pattern is Sequential Reads or Random Reads.

- Line 1 : 1st Condition checking for the storage device
  Algorithm starts with the first level of condition checking. Here the condition will be true if the storage device selected is a Hard Disk Drive. Since the

condition is false, the control of the algorithm will be transferred to the line number 11.

- Line 11 : 2nd Condition checking for the storage device
  This is the start of the second condition checking for the storage device. Here the condition will check whether the storage device is a Flash Drive (FD). Since this condition becomes true for this scenario, the control will be to the next line.

- Line 12 : Condition check for access pattern
  This is the first condition check for the access patterns inside the second storage device selection. In line number 12, the condition checks whether the access pattern is sequential read or random reads. If either on of these conditions become true, the control will be given to the line number 13.

- Line 13: Change block size
  In line number 13, the block size of the data will be changed to 64. This value is the best value resulted for sequential reads and also for random reads, for Flash Disk Drive.

With the change of the block size, algorithm for this scenario ends.

Scenario 6 : If the data storage used is Flash Disk and the access pattern is Sequential Write.

- Line 1 : 1st Condition checking for the storage device
  Again the algorithm starts with the first level of condition checking. Here the condition will be true if the storage device selected is a Hard Disk Drive. Since the condition is false, the control of the algorithm will be transferred to the line number 11.

- Line 11 : 2nd Condition checking for the storage device
  This is the start of the second condition checking for the storage device. Here the condition will check whether the storage device is a Flash Drive (FD). Since this condition becomes true for this scenario, the control will be to the next line.

- Line 12 : 1st Condition check for access pattern
  In line number 12, the condition checks whether the access pattern is sequential read or random reads. Since this becomes false, the control moves to the next condition check.

- Line 14 : 2nd Condition check for access pattern
  In line number 14, the condition checks whether the access pattern is sequential write. According to the scenario this condition becomes true. Therefore, the conditions moves to the next line.

- Line 15: Change block size
  In line number 15, the block size of the data will be changed to 16. This value is the best value resulted for sequential writes for Flash Disk Drive.

With this change, algorithm for this scenario ends.

Scenario 7 : If the data storage used is Flash Disk and the access pattern is Random Write.

- Line 1 : 1st Condition checking for the storage device
  The algorithm starts with the first level of condition checking. Since the device is not Hard Disk Drive, the condition will be false. Then the control of the algorithm will be transferred to the line number 11.

- Line 11 : 2nd Condition checking for the storage device
  Here the condition will check whether the storage device is a Flash Drive (FD). Since this condition becomes true for this scenario, the control will be to the next line.

- Line 12 : 1st Condition check for access pattern
  In line number 12, the condition checks whether the access pattern is sequential read or random reads. Since this becomes false, the control moves to the next condition check.

- Line 14 : 2nd Condition check for access pattern
  In line number 14, the condition checks whether the access pattern is sequential write. This will also become false and the control moves to line number 16.

- Line 16: 3rd Condition check for access pattern
  Here the condition check will be for the random writes. Therefore, this condition will become true and the the next line will be taken.

- Line 17 : Change block size
  In line number 17, the block size of the data will be changed to 1024. This value is the best value resulted for random writes for Flash Disk Drive.

With this change, algorithm for this scenario ends. Also, this ends the selection for the Flash Disk Drives. The other storage device considered for the algorithm is the Solid State Disk. The next scenario will describe the algorithm related to Solid State Disk.

Scenario 8 : If the data storage used is Solid State Disk

- Line 1 : 1st Condition checking for the storage device
  Algorithm starts with the first level of condition checking. Here the condition will be true if the storage device selected is a Hard Disk Drive. Since the condition is false, the control of the algorithm will be transferred to the line number 11.

- Line 11 : 2nd Condition checking for the storage device
  This is the start of the second condition checking for the storage device. Here the condition will check whether the storage device is a Flash Drive (FD). This will also become false in this scenario and the control will be moved to line number 19.

- Line 19 : 3rd Condition checking for the storage device
  This is the third condition checking for the storage device. This condition will whether the storage device is a Solid State Disk (SSD). This condition becomes true for this scenario and the control will move to the next line.

- Line 20: Change block size
  In line number 20, the block size of the data will be changed to 1024. This value is the best value resulted for all access patterns, for Solid State Disk.

With this the algorithm for this scenario ends. In the scenario 8, for the Solid State Disk there is only one statement to change the block size. For all the access patterns, there is no difference in the block sizes. All these scenarios cover all the possibilities for the Algorithm proposed. The next section comprises of the Evaluation of the algorithm with using sample scenarios.

Chapter 5

Evaluation

This chapter contains the evaluation of the algorithm with using some sample scenarios, the major limitations of the algorithm and also the future directions of the research.

5.1 Test Scenarios

The algorithm 1 is evaluated with using some practical scenarios. These scenarios were already mentioned under the Section 1.1 as situations that should be considered when the storage devices change. Therefore, the same scenarios are used in this section to evaluate the algorithm. For each of these scenarios the behavior of the algorithm can be checked.

Scenario 1 : A request to select all the data from a table in the database.

Before moving in to the algorithm, it should be able to decide on what storage device this table or the database is stored. Then the decision should be taken whether the data access will be done sequentially or randomly. In this scenario, since all the data should be read, this can be taken as a sequential read.

If the device is a Hard Disk Drive the algorithm will behave in the below mentioned manner.

- Check the first condition against the storage device variable. When the storage device is HDD, condition will become true.
- Once the condition is true for the Storage device, then the conditions will check for the access pattern.

- Since the access pattern is sequential read, when the AP becomes equal to SR, the condition will become true.
- When the condition become true for the access pattern, then the block size will be changed.

With this, the block size to read will be set to the best block size available for that given data access pattern. Hence, the performance of access this set of data will be increased. This will be the same for the other devices.

Scenario 2 : A request to select data from a table in the database with a given selection condition (e.g: where clause).

In this scenario also the storage device and the access pattern should be decided at first. Then the algorithm can be evaluated. Since there is a selection condition the data accessing can be taken as a random read and the device should also be decided.

If the device is a flash disk, the algorithm will behave in the below mentioned manner.

- Check the condition for the storage device until the storage device is flash disk. When the condition for the storage device becomes true the access pattern will be checked.
- According to the scenario, the access pattern will be taken as random read. When the AP in the algorithm is equal to RR, then the condition becomes true.
- Once the condition becomes true for the access pattern, then the block size will be changed.

Since the block size that is used from the algorithm is the best performance block size available for the random reads, the performance will be higher.

According to the explained two scenarios, the performance will be higher if the algorithm is used when accessing data from different storage devices using different accessing patterns. This performance increase will be evident for all the access patterns and for all the devices. The reason for this increase is clearly due to the selection of the best performing block size. Given these advantage, there are limitations also included in the algorithm. These are given in the next section.

5.2 Limitations

In all aspects there are limitations. Keeping with that, the algorithm proposed also carries some limitations. This section comprises of these limitations.

As given under the section 3.2, there are two types of experiments that was carried out. The first set of experiments was to analyze the performance with different block sizes for different storage media. The second set of experiments was to analyze the performance with different file sizes for different storage media and block sizes. Both these experiment sets were successful enough to prove that there should be a change in the algorithms with considering the block sizes and the file sizes when accessing data. But when designing the algorithm, the file size variable was not considered. This is due to the reason of inability to clearly test the performance of the storage media for different access pattern along with the changing block sizes. Though it is clear that there should be a change in the algorithm along with the file size, it is not included because the behavior could not be tested for different access patterns for different file sizes. This would have been possible to test with using different set of testing tools, but due to the limited time factor that will be added to the future work.

This algorithm only addresses the access patterns and the block sizes for different storage media. But when using flash memory technology devices there are many other changes required in the algorithms. There are changes needed in the buffer manager, query processing techniques and also some other changes needed are yet to be found. There are different research areas available under this topic due to this reason and some of these are given under the next section; future direction.

5.3 Future Direction

As mentioned in section 5.2, the algorithm proposed only consider the changes in the block size with regards to the access pattern and the device used for storage. But as it was proved that the file size should also be considered in the changes, it can be given as the further improvement for this algorithm. Therefore, this algorithm can be further improved by considering the file sizes also in the algorithm.

While performing the literature review, it was clear that there are many other changes that are required when the storage media changes. Mainly, the focus can be given to the buffer manger changes. When considering the data access in a database, the buffer manager plays a major role. If the flash memory devices are introduced as the storage device for the database, the buffer manager should behave

differently specially with the replacing policies due to the performance difference of the flash devices. Therefore, flash aware buffer replacement policies are needed. There are many researches carried out under this topic but still with many future work left [7, 8, 18, 19].

There are other changes required in areas like query processing and I/O scheduling as well. All these areas are still available for further researching. With these future directions, the conclusions for this research can be expressed which is given under Chapter 6.

Chapter 6

Conclusions

The mechanical nature of the Magnetic Disks carries inherited problems such as long latencies in handling random accesses, excessively high power consumption and uncertain reliability [6]. Therefore, there had always been a need to find out mechanisms where these problems can be eliminated. But since these issues are rooted in the mechanical nature of the Magnetic Disks, these are difficult to be solved physically by disks themselves. Hence, the need for a replacement of the Magnetic Disk or the Hard Disk Drive (HDD) from a different technology aroused.

Along with the introduction of the mobile devices different Storage Media such as flash drives, Secure Digital (SD), Micro SD and Solid State Disks (SSD) comes in to play. These new Storage Media are mainly based on the flash memory technology. The characteristics of Flash memory are significantly different from Magnetic disks. Unlike the traditional Magnetic Disks with mechanical components, these storage media are based on the semiconductor chips which provide strong technical merits. These devices have no latency and were promising for the expected need of the small form factor and the access speeds. When considering SSDs, the benefits include high performance in random accesses and shock resistance. Specially, with using SSDs as a storage it is possible to achieve short startup times and most importantly it is possible to eliminate mechanical overheads such as seek time, rotational delay and spin up delay in the Magnetic Disks [7]. Due to these advantages of the flash memory technology over the Magnetic disks, SSDs has become the most promising substitute for Magnetic Disks.

When considering DBMSs, the most important factor that is always considered is the read write costs in terms of the performance and energy consumption. Therefore, when it comes to the decision of selecting between Magnetic Disks over SSDs or any other storage device, the read write mechanisms and the speeds should be considered. Unlike the Magnetic Disks where the read write performance is sym-

metric, SSDs provides a clear asymmetric nature. However, previous researches prove that there is more than an order of magnitude improvement in transaction throughput and response time with replacing Magnetic Disks with SSDs [9]. The DBMS components are designed according to the given fact that the read write costs being symmetric, with assuming that the only used storage device as the Magnetic disk. If the SSDs are to replace the Magnetic disks, the components of the DBMS should be changed in accordance with the above mentioned asymmetric nature of read writes. Therefore, the research aim was set based on this idea. The aim of the research was set to find out the performance variance between the usage of HDD and flash based devices with different scenarios to find out good strategies to access data stored in the devices.

With the aim of the research there was research questions formed. Revisiting the section 1.2, following are the research questions.

- Is there a possibility in aligning the DBMS components with the characteristics of flash based storage devices so that the performance of the DBMS is increased?
- When the storage device changes, what other factors should be considered when accessing data with regards to algorithms?

According to the literature review that was performed, the answer to the first question will be a yes. There are many areas in the DBMS that should be aligned with the characteristics of the flash based storage devices. Some of these are mentioned in the Future Directions in section 5.3. In order to answer the second research question experiments were performed to analyze the performance of different storage media in accessing data. Therefore, experiments were carried out to measure the performance for different access patterns over the different storage media for block sizes and file sizes. Further, the tests were compared to discover the performance patterns with regards to the access time and the block sizes. After performing the tests individually for all these storage Media, the results were analyzed by comparing the different storage Media. As example, HDD performance was compared against flash drive for all access patterns separately. With these comparison results, it was interesting to see how the storage device IO performances vary with the block sizes. The results also varied unexpectedly for the different read write patterns.

With all these experiments it can be proved that there is a need for changing the DBMS components if the used storage device is changed. The results discovered from the analysis leads us also to re-think about the storage Medias that are used

in different applications. As answer to the second research question, in order to gain good performance for the applications used in all devices, there is a need to consider the block sizes, file sizes, access pattern when the device changes.

Keeping with the research aim and with considering the experiment results, an algorithm is proposed that can be used to optimize the access speed of data. This algorithm proposes a method to change the block size of the accessed data with the access pattern used and also the device used for data storage.

As conclusions from the research, the following outcomes were achieved. First, the best performing storage device / devices for different block sizes and different access patterns are identified. Most importantly an algorithm is proposed for changing the block sizes along with the storage media that is used and also the access pattern used.

Bibliography

[1] Albert S. Hoagland, "History of Magnetic Disk Storage Based on Perpendicular Magnetic Recording", In Proceedings on IEEE Transactions on Magnetics, Vol. No.4, July 2003.

[2] Albert S. Hoagland, "Early History and a 50 year Perspective on Magnetic Disk Storage: The Genesis of the Current Revolution in Information Storage", Magnetic Disk Heritage Center, February 2005.

[3] Ragu Ramakrishnan and Johannes Gehrke, Database Management Systems, Third Edition, MaGraw-Hill, 2003, chapter 9, pg.306-309.

[4] David A. Patterson, "Latency Lags Bandwidth", Communications of the ACM, 47(10): 71-75, October 2007

[5] Sang-Won Lee, Bongki Moon and Chanik Park, "Advances in Flash memory SSD Technology for Enterprise Database Applications", In Proceedings of ACM SIGMOD 2009, July 2009.

[6] Feng Chen, David A. Kaufaty and X. Zhang, "Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives", In Proceedings of ACM SIGMETRICS/Performance 2009, June 2009.

[7] J. Seal, H. Shim, J. Kim and S. Maeng, "A Buffer replacement Algorithm: Expliting Multi-Chip Prallelism in Solid State Disks", ACM, CASES 2009, October 2009.

[8] Yi Ou, T. Harder and P. Jin, "CFDC - A Flash aware Replacement Policy for Database Buffer Management", In Proceedings of the 5th International Workshop on Data Management on New Hardware 2009, June 2009.

[9] Sang-Won Lee, Bongki Moon, Chanik Park, Jae-Myyung Kim and Sang-Woo Kim, "A Case for Flash Memory SSD in Enterprise Database Applications", In Proceedings of ACM SIGMOD 2008, June 2008

[10] X. Meng, P. Jin, W. Cao and L. Yue, "Report on the First International Workshop on Flash-Based Database Systems (FlashDB 2011)", SIGMOD Record, June 2011 (Vol.40, No. 2).

[11] T. Fernando and P. S. Haddela, "I/O Performance of Mobile Devices over different storage media and block sizes", In Proceedings of 6th IEEE International Conference on Industrial and Information Systems 2011, pp. 448-452, Aug. 2011

[12] M-Systems, "Two technologies compared: NOR vs. NAND", White paper, July 2003.

[13] H. Lee and N. Chang, "Low-Energy Heterogeneous Non-volatile Memory Systems for Mobile Systems", Journal of Low Power Electronics, vol.1, no.1, pp.52-62, Apr.2005.

[14] Connexions, "Introduction to Computers and Programming", Hardware and Software, July 7th 2009, [Online], Available: http://cnx.org/content/m27277/latest/, [Accessed: 15th October 2010].

[15] E. Grochowski and R.D. Halem, "Technological impact of magnetic hard disk drives on storage systems", IBM Systems Journal, Vol 42, No 2, 2003.

[16] D. Tsirogiannis, S. Harizopoulos, M. Shah, J. Wiener and G. Graefe, "Query Processing Techniques for Solid State Drives", In Proceedings of SIGMOD 2009, pp.59-72, 2009.

[17] S. Yin, P. Pucheral and X. Meng, "PBFilter: Indexing Flash-Resident Data Through Partitioned Summaries", In Proceedings of CIKM 2008, pp. 1333-1334, 2008

[18] S. Park, D. Jung, J. Kang, J. Kim and J. Lee, "CFLRU: A Replacement Algorithm for Flash Memory", In Proceedings of International Conf. on compilers, architechture and synthesis for embedded systems, pages 234-241, 2006.

[19] H. Jo, K. Kang, S. Park, J. Kim and J. Lee, "FAB: Flash-Aware Buffer Management policy for portable media players", IEEE Transactions on Consumer Electronics, 52(2): 485-493, 2006.

[20] U. Cesana and Z. He, "Multi-Buffer Manager: Energy Efficient Buffer Manager for Databases on Flash Memory", ACM Transactions on Embedded Computing Systems (TECS), Vol 9, Issue 3, Feb 2010.

[21] H. Jung, "LRU-WSR: integration of LRU and write sequence recording for flash memory", IEEE Transactions on Consumer Electronics, Vol 54, Issue 3, Aug. 2008

[] D. Seo and D. Shin, "Recently-evicted-first buffer replacement policy for flash storage devices", IEEE Transactions on Consumer Electronics, Vol 54, Issue 3, Aug. 2008

[22] S. Lee and B. Moon, "Design of Flash-Based DBMS: An In-Page Logging Approach", In Proceedings of ACM SIGMOD 2007.

[23] Y. Ou, T. Harder and D. Schall, "Performance and Power Evaluation of Flash-Aware Buffer Algorithms", Springer 2011.

[24] Y. Li, S. T. On, J. Xu, B. Choi and H. Hu, "DigestJoin: Expliting Fast Random Reads for Flash-based Joins", In Proceedings of IEEE Mobile Data Management 2009, pp. 152/161, 2009.

[25] M. Dunn and A. L. N. Reddy, "A New I/O Scheduler for Solid State Devices", Department of Electrical and Computer Engineering Texas A and M University, Tech. Rep. TAMU-ECE-2009-02-3, 2009.

[26] EFD Software, "HD Tune", [Online], Available: http://www.hdtune.com/index.html, [Accessed: 25/10/10]

Appendix A