Dynamic load balancing of SCSI WRITE and WRITE SAME commands

Vladimir Tikhomirov

Supervised by Dr. Simo Juvaste

Advised by Antti Vikman and Timo Turunen

Master's thesis.

May 27, 2013

School of Computing

University of Eastern Finland, Joensuu, Finland.

Abstract

Load balancing is a technique to distribute workload, which allows to perform the same tasks with faster speed. Application of load balancing methods can help to fulfil the same problem using less amount of resources. This methodology is very important in any computational system, because it can improve the speed, which is one of the crucial properties.

The thesis describes the communication model between SCSI controller and SCSI disks by different write commands. The application system sends SCSI WRITE and WRITE SAME commands to the disks through the controller to realize the complete erasure of the disks. The aim of the research is to figure out if the parallel strategy of overwriting the disks is optimal. The thesis discusses the parameters, which can slow down the speed of erasure and how we can avoid these parameters if it is possible.

The results of the thesis show that disk speed is the limiting factor for the erasure, if we perform it by WRITE SAME command. Using the WRITE command, the bus is the bottleneck, but by varying transfer length of the buffer it is possible to find the optimal way of sending the commands depending on the amount of disks.

Keywords:

Load balancing, RAID, SCSI, pass through commands.

Contents

List of abbreviations

ATA	Serial ATA (AT Attachment)
HP	Hewlett Packard
IDE	Integrated Drive Electronics
IT	Information Technology
JBOD	Just a Bunch of Disks
RAID	Redundant Array of Independent Disks
SAS	Serial attached SCSI
SCA	Single Connector Attachment
SCSI	Small Computer System Interface
SMART	Self-Monitoring, Analysis and Reporting Technology
VHDCI	Very-High-Density Cable Interconnect

Chapter 1

Introduction

Information technology (IT) is growing up so fast nowadays that we cannot even compare what was 10 years ago and what is now. Current situation shows that every month, every day, every minute IT-community goes forward and the steps are so huge that some products, which were so popular 1 or 2 years ago, are already old and people even cannot use them, because the company already stopped supporting these solutions. All over the world people start using more mobile phones, laptops, electronic books and so on. This is the reason why the amount of information is increasing so fast. From the user's side the devices should work without any errors and delays. Moreover, user wants that it will be easy to use and the information will keep privacy that nobody can get the access to it. From technical side the systems start to be very complicated, because the processes start to take more memory and the commands start to mix up, that is why for the device it started to be more difficult to handle all these things.

Every computation system has the limits of speed, memory or some other parameters. However, if the program even works correct the speed can be

decreased by some reason that can be not because of software. It is also possible to happen that the application would start taking too much memory. That is why IT-community started to focus on these problems a lot. In this research we will consider the optimization, which is a general technique of finding the solutions for this topic. In mathematics and computer science, optimization is the selection of a best element from some set of available alternatives with regard to some criteria. In current work we will try to find the optimal speed for the erasure process, which depends on a lot of parameters. Load balancing methods, which is part of optimization theory, will help us to find the right solution.

The methodology called *Load balancing* was invented not so long time ago and serves for making the computation system faster with less usage of the resources [19]. Mostly this technology is using for the Internet services [8], for example, one of the most used common applications of load balancing is to provide a single Internet service from multiple servers. In this paper we will consider load balancing methodology on the example of communication between Small Computer System Interface (SCSI) controller and different amount of SCSI disks.

Lets imagine that some program sends the SCSI WRITE and SCSI WRITE SAME commands from the computer though the controller to the disks to perform the complete erasure. There are a lot of parameters, which can prevent the communication speed between the computer program and the disks. Moreover, the disks are behind the SCSI controller, which adds some more values for consideration. In the thesis we will apply several load balancing strategies and test the communication model with different parameters. The task is to figure out the technique, which is the fastest one.

The thesis discusses the parameters, which are more important for the speed of the erasure. Tests with HP Smart Array 642 controller shows how the speed depends on such parameters as transfer length, disk cache, disk capacity, amount of the disks and so on. Moreover, the thesis suggests applying special dynamic load balancing system to make the erasure faster. The aim of this system is finding the optimal transfer length for the buffer, which allows to send the data in optimal way.

Current thesis has 6 chapters, including concurrent parts as introduction and conclusion. The second chapter gives an open view to the basic concepts of the thesis. Chapter 3 discusses the mathematical way of the problem and considers all parameters, which can influence on the communication process between the devices. The fourth chapter is one of the most important ones, because we describe there the main idea of application the dynamic load balancing system. The principal concepts of new system are based on the chapter 5, where we present different results from testing. Chapter 6 concludes all work and sum up the ideas from the thesis.

Chapter 2

Introduction to load balancing and RAID controllers

In this chapter, we will focus on basic information for understanding the topic of the paper. We will discuss different types of load balancing systems and consider the things, which programmer should take into attention during implementing this kind of systems. This chapter also describes different types of RAID, which is undivided part of the controllers. Finally, the last section tells about different write commands, which we are going to use during the thesis. This section gives the information about the structure of the commands and opens many command parameters in details. This chapter keeps the basic knowledge, which reader needs for understanding the results of the paper.

2.1 Load balancing

In general, load balancing is the methodology that increases the speed by balancing the resources. This part of computer science is quite new, but the scientists started to focus on that almost 30 years ago [16]. The main idea is to do as much as possible with least amount of resources. Mostly it depends on the processes, which we can divide by threads and launch them parallel. Sometimes load balancing divides to the four types: static, dynamic, combined, and preemptive. In scientific articles, combined and preemptive load balancing systems are mentioned very rarely because of their specificity and usefulness only in some certain cases. That is why in this research we will consider two main different types: static and dynamic.

Static load balancing system makes the analysis before the application starts working. That is why the programmer should know some information for creating good load balancing system via static methods. Firstly, he should know the amount of distributed resources before starting the implementation of the program. Secondly, the programmer should divide the time in such a way that the duration of the tasks would be approximately the same. There are several methods to hand out the tasks such as Round Robin algorithm, Random algorithm, Recursive Bisection algorithm and some others [14], [18]. Frequently scientists take the initial parameters from the previous launch results and genetic algorithms are used. We can take an example of load balancing system from image processing. Let consider 16×16 segmented image and eight processors. With that image, each processor would work on four segments. The most important piece of all this division of work is that each processor determines this information and which segments it will work on.

Dynamic load balancing systems have more possibilities because the balancing process can be done during the application running [5]. This fact gives the advantage of resource movement from busy part to less busy. Thus, we will achieve that all parts of the application will be in average busy condition, which is the ideal solution. The disadvantage of that method is that the system should also collect the information about the status of application, which follows the increasing of the resources. The programmer should follow on three general criteria during implementation the dynamic load balancing system:

1. Which process is busy or free
2. Priority of the task to be done
3. Length and/or duration of the task

The program, which realized the dynamic load balancing system, should check the loading of computation units, connection possibility, and frequency of sending commands. Dynamic load balancing methodology has many methods, such as Bidding algorithm, Drafting algorithm [13], Recursive Coordinate Bisection (RCB) and some others. In this paper, we will call these methods as strategies, but in the literature, it is also possible to find them as policies and logics.

2.2 RAID

RAID - Redundant Array of Independent Disks - is a storage technology that combines multiple disk drive components into a logical unit. That means that this technology gives a possibility to keep the information in a specific way. RAID controllers are the devices, which realize this technology. In simple words without a RAID controller computer can see physical disks as a logical disks without any difference. However, the computer will see only the logical disks, if the RAID controller is connected. Logical disks are definitely different in comparison in physical disks. Nevertheless, logical disks can be equal to physical if RAID controller is in special mode called JBOD (Just Bunch of Disks). RAID controller takes all issues about creation of these logical disks.

There are several types of RAID: RAID 0, RAID 1, RAID 5, RAID 6 and some others [4]. These types of RAID are basic types of RAID, thus there are many other types, which also combine them together. For example, RAID 10 is actually combining of RAID 0 and RAID 1. Each type or architecture provides a different balance between the key goals: reliability and availability, performance, and capacity. That means that every RAID keeps the information in a different way.

One of the huge advantages of using RAID is that if one of the disks breaks down others can recover the information on that disk. Not all RAID architectures support this functionality, but most of them do. Some types of RAID provide fault tolerance of even two drive failures (RAID 6).

Let consider some types of RAID more detailed. RAID 1 called also as "mirroring" is one of the main fundamental types of RAID, which refers to

maintaining duplicate sets of all data on separate disk drives. There must be two disks in the configuration and there is a cost disadvantage as the usable capacity is half the number of available disks. RAID 1 provides cost-effective, high fault tolerance for configurations with two disk drives. The prototype of RAID 1 is presented on the Figure 2.1.

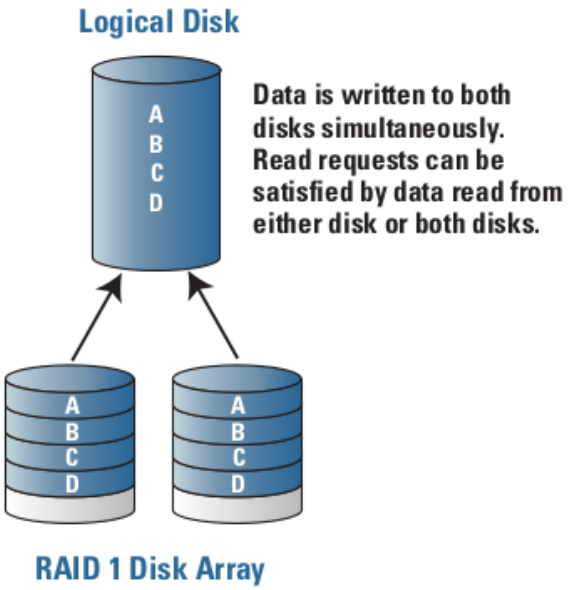

Figure 2.1: Prototype of RAID 1

Another type RAID 5, which is presented on the Figure 2.2, uses data striping in a technique designed to provide fault-tolerant data storage, but doesn't require duplication of data like RAID 1 [2]. Data is striped across all of the drives in the array, but for each stripe through the array (one stripe unit from each disk), one stripe unit is reserved to hold parity data calculated from the other stripe units in the same stripe. Read performance is therefore very good, but there is a penalty for writes, since the parity data has to be recalculated and written along with the new data. RAID 5 requires a

minimum of three disks and a maximum of 16 disks. RAID 5 usable capacity is between 67% - 97% depending on the number of data drives in the RAID set. Moreover, nowadays there are RAID controllers, which already have

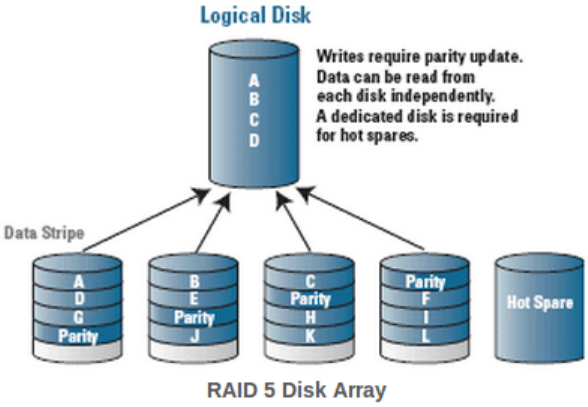

Figure 2.2: Prototype of RAID 5

load balancing system inside and the user can set the mode, which is more comfortable in the some situations. This system takes care only about logical disks, but we need to communicate only with physical disks. That is why we need to have our own load balancing system, which will help us to distribute the resources before the commands come to the controller. Because in our case the commands should be sent straight to the disks, load balancing system from the controller does not help at all. Moreover, during communication directly with the disks, we do not need to remember if there is any RAID or not, because the commands are sent through the controller without any changes.

The Figure 2.3 presents one of the SATA RAID controllers. From the name of the device, it is clear that this controller can be connected to 8 SATA disks. It supports following types of RAID: 0, 1, 5, 10, 50 and JBOD (Just

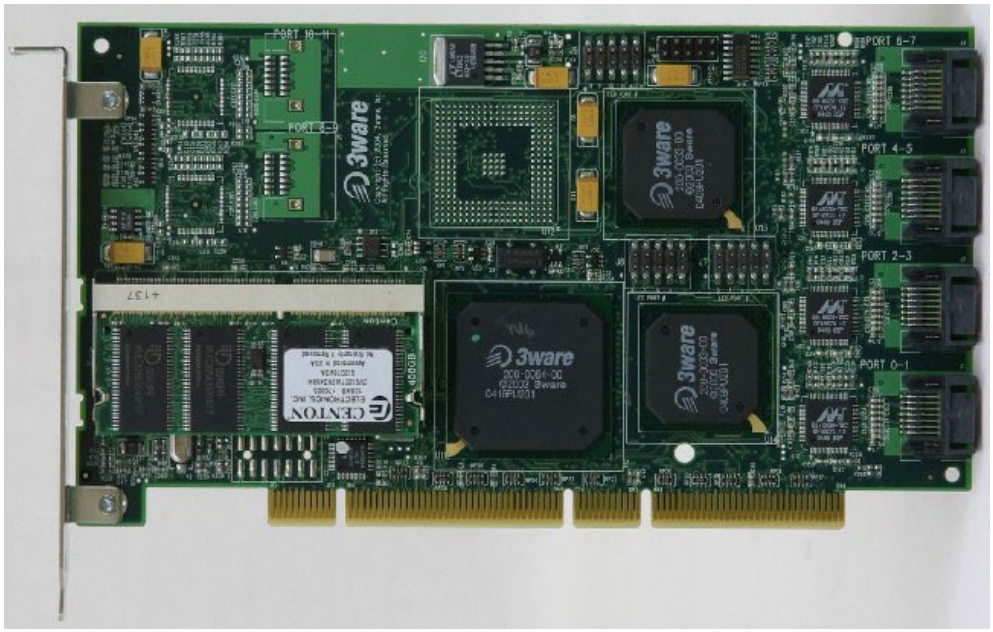

Figure 2.3: 3ware SATA RAID controller 9500S-8

a Bunch of Disks).

2.3 SCSI RAID controllers

SCSI - Small Computer System Interface - is a set of standards for physically connecting and transferring data between computers and peripheral devices [15]. The SCSI standards define commands, protocols, and electrical and optical interfaces. There are also other interfaces such as SATA (Serial ATA (AT Attachment)), IDE (Integrated Drive Electronics), SAS (Serial attached SCSI). In general, we divide the interfaces to two categories: ATA and SCSI. In the following content, it means the according protocol. IDE and SATA interfaces relate to ATA and SAS belongs to SCSI. Several types of disks are presented on the Figure 2.4. In this paper, the topic mostly is about SCSI interface, but sometimes we compare it with ATA. If we compare SAS and SCSI, we consider them as similar interfaces. However, SAS is a new version of SCSI and it gives a single channel for each disk. SCSI has only one channel for all disks.

(a) SCSI disk

(b) SATA and IDE disks

Figure 2.4: Three different types of disks

2.4 SCSI WRITE and WRITE SAME commands

This research is made for the Blancco Oy Ltd, which produce the erasure software. The main part of the erasure is performed by SCSI WRITE and WRITE SAME commands [9]. In general, all commands belongs to three groups: Non-Data commands, Data-In and Data-Out. For the last two groups it means that the programmer should send the command including the buffer. Data-Out means that the data is sent to the disk. For the Data-In group it is in opposite way and data comes from the disk. The example of Data-In command is any SCSI READ command. Write commands belongs to the Data-Out group because these commands must have Data-Out buffer, containing the data, which we write to the disk. Nowadays, for SCSI there are 4 different versions of write command and 3 for the write same command.

Lets consider SCSI WRITE commands first. Mostly the commands are different because of the length. Each command has not only operation code, which is the main criteria for the command, but also some other variables such as Logical Block Address (LBA), transfer length, control and some others. These variables do not include Data-Out buffer and the Figure 2.5 shows the example how the commands are defined in specification [9].

The following commands exist for writing the data to the disk:

- WRITE (10)
- WRITE (12)
- WRITE (16)
- WRITE (32)

The numbers in brackets shows the length of the command in bytes. It is obvious that the WRITE (10) command is the base for others. It seems that by

Table 108 — WRITE (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)							
•••	LOGICAL BLOCK ADDRESS							
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

Figure 2.5: Definition of WRITE 10 command

now the best choice is WRITE (16) because in [9] there are some notes, that «Migration from the WRITE (10) command to the WRITE (16) command is recommended for all implementations». WRITE (32) should be used only in some special cases [9]. In this research, we will focus only on WRITE 10 command, because after some test, it started to be understandable, that bus plays an important role in this model, and there is no reason to send more data.

SCSI WRITE SAME commands have the same purposes as SCSI write commands. The difference and biggest advantage is that SCSI WRITE SAME command can be sent once and the Data-Out buffer can be written to the disk several times. That gives faster speed because the bus is not busy any more - only one command was sending, but the buffer is still writing.

There are three different types of write same commands, as we mentioned before:

- WRITE SAME(10)
- WRITE SAME(16)

- WRITE SAME(32)

The number in brackets also shows the length of the command in bytes. Number of logical blocks is one of the most important variables in these commands because we write this number of blocks to the disk with the same buffer. It is obvious that if the length of the SCSI WRITE SAME command is quite big, we can write to the disks much more number of blocks. We will consider only WRITE SAME 10 command, because the results after few tests showed that there is no reason to send bigger buffer, because the disk could not work faster. ATA specification has the similar command, but it should be send differently and the device should support SMART (Self-Monitoring, Analysis and Reporting Technology) [10]. It is a monitoring system for computer hard disk drives to detect and report on various indicators of reliability.

Chapter 3

Mathematical problem definition

This chapter opens the model from the mathematical point of view and defines the problem that should be solved in the paper. In the beginning of the chapter we consider the parameters that describe how many commands should be sent for the complete erasure of all amount of disks and in which order. Later we model the dynamic load balancing system, which could help with improving the speed, and, finally, we pose the optimization problem. After that we try to estimate what kind of limitations exist from theoretical and technical sides. This part of the chapter gives an idea what we should expect during practical tests of the erasure. Moreover, it helps to understand what parameters we should take into consideration more during searching the solution for our problem.

3.1 Communication model between SCSI controller and SCSI disks

Currently the time costs a lot that is why it is important to make programs as fast as possible. Because of that optimization problem is one of the most significant ones. In general it is the problem of finding the best solution from the set of all feasible solutions. It is often done with different algorithms, which analyse the solutions during variable changing. In our case the optimization problem is based on time optimization. That means that the time will be optimized by several variables which can be changed.

Lets consider the model presenting on the picture 3.1, which shows how SCSI controller is connected to the disks and CPU (Central processing unit). This picture can be divided to 5 parts: CPU, where Blancco software is running, bus a - from CPU through motherboard to the SCSI controller, SCSI controller b, buses c - from SCSI controller to the disks d. Let consider that the SCSI controller is connected to N disks.

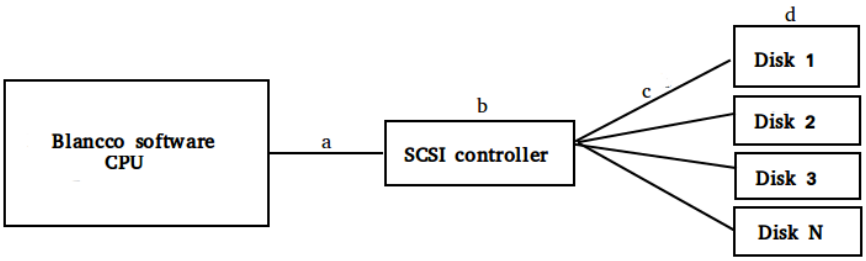

Figure 3.1: Communication model of SCSI controller

This Figure presents how SCSI controller is connected in our case.

All the time people want to do everything as fast as possible, but of course there are limitations in different places. The same situation is in this model - each part has the limitation of the speed. Moreover, there are also limitations for the memory.

Lets consider the Figure 3.1 more detailed. In this model CPU part will be skipped, because there are so powerful computers in current time that it should not be taken into consideration. The main idea of Blancco software is to write the data to the disks. Lets discuss what *data* means in this context. On the Figure 3.1 there are N disks and each of them has its own capacity d_i, where $i = \overline{1, N}$ shows the number of the disk. Capacity d_i is calculated in megabytes. Each disk i should get the amount of data, which is at least equal to the capacity of the disk. Moreover, we maybe need to send the commands for setting the connection with the disk.

We discussed in the section 2.4 that Blancco software uses two different versions of write command. Each of these commands has its own capacity. Let define these capacities in a mathematical way. The capacity of the WRITE command is c_1, the capacity of the WRITE SAME command is c_2 and the capacity of other commands is c_3. In these context other commands means the commands for identification the device and setting the connection. All these three variables c_1, c_2, c_3 are constant and are defined by Blancco software. Hence for replacing all data on N disks it should be sent F MB from SCSI controller b to the disks d through the buses c. Let F be called as *complete erasure*, is it is calculated by the following formula:

$$F = \sum_{i=1}^{N} F_i, \tag{3.1}$$

where F_i is the amount of megabytes, that should be sent to the disk i, which

is calculated as

$$F_i = c_1 m_{i1} + c_2 m_{i2} + c_3 m_{i3}, \tag{3.2}$$

where m_{ik} is the number of times that command c_k should be sent, where $k = \overline{1,3}$. Furthermore, complete erasure F should fulfil the following condition:

$$c_1 c_2 = 0, \forall F_i, i = \overline{1,N}. \tag{3.3}$$

This condition means that only 1 command c_1 or c_2 can write to each disk. Thus the software can not erase the same disk with both commands c_1 and c_2. That means that only software decides which command is better to use and dynamic load balancing system should apply the strategy how to apply it more efficiently.

For clear vision and comprehension it is convenient to present the set of F_i as a matrix F_m:

$$F_m = \begin{pmatrix} F_1 \\ \vdots \\ F_N \end{pmatrix} = \begin{pmatrix} c_1 m_{11} & c_2 m_{12} & c_3 m_{13} \\ \dots & \dots & \dots \\ c_1 m_{N1} & c_2 m_{N2} & c_3 m_{N3} \end{pmatrix}, \tag{3.4}$$

where F_m is the matrix with dimension $N \times 3$.

The capacity of each disk i should be less than amount of data, which is sent for writing. That means that the following condition should be fulfilled:

$$d_i < F_i, i = \overline{1,N}. \tag{3.5}$$

But this condition is true only for the WRITE command c_1, because the WRITE SAME command c_2 sends the same buffer several times, which means that we do not need to send this buffer all the time. Thus, for the WRITE SAME command c_2 we have another condition:

$$d_i > F_i, i = \overline{1,N}. \tag{3.6}$$

The main aim of the research is to find the way how to send the data faster to the disks. Of course, that it depends on all the devices that are in the model, on the strategy how the software sends the commands and also on the parameters of the command that we use for erasure. Let define the variable S_{max} for maximum speed and it is obvious that this variable is constant. According to the formula 3.1 and maximum speed S_{max} it is possible to calculate the minimum time T_{min} for realization complete erasure F by the following formula:

$$T_{min} = \sum_{i=1}^{N} t_i = \frac{\sum_{i=1}^{N} F_i}{S_{max}} = F/S_{max}, \tag{3.7}$$

where t_i is the time for sending F_i MB to i disk with the maximum speed S_{max}.

3.2 Mathematical problem definition

Let consider that in the same model there is dynamic load balancing system and it calculates the application state $m = F/s$ times, where F is the amount of megabytes to complete the application task and s is the step. In this context step s means the amount of commands and m means how many times the dynamic load balancing system should apply another input data for the next step.

There is matrix $H = \{h_{ij}\}$, where $i = \overline{1, N}$ is the disk number, $j = \overline{1, m}$ is the step number of the dynamic load balancing system and h_{ij} is the number of commands, which is sent to the i disk at j step. That means that for complete erasure F we need the following amount of commands:

$$\sum_{i=1}^{N} \sum_{j=1}^{m} h_{ij} = \sum_{i=1}^{N} \sum_{k=1}^{3} m_{ik}. \tag{3.8}$$

As an example of matrix H it is convenient to consider 4 disks, which can be erased by 100 commands with the condition that we can not send more than 100 commands per step. Thus the matrix H can be presented as

$$H_1 = \begin{pmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{pmatrix} \tag{3.9}$$

or a bit more complicate:

$$H_2 = \begin{pmatrix} 50 & 10 & 20 & 20 \\ 20 & 50 & 10 & 20 \\ 20 & 20 & 50 & 10 \\ 10 & 20 & 20 & 50 \end{pmatrix}. \tag{3.10}$$

Matrix H_1 and H_2 are practical examples of the matrix H. The first matrix shows the trivial strategy - to send as many commands as the program can to fulfil the condition during each step. Matrix H_2 shows that it could be done in a different way and we can divide the whole amount of commands and send it separately. The strategy should be applied by some dynamic load balancing algorithm and the aim of the research is to find out what is the optimal matrix H_{opt} and could load balancing help in this case or not.

Also there is matrix $Q = q_{ij}$, where q_{ij} is the order number in the queue for sending commands to i disk at j step. In this case it is possible to show to which disk the program should send the command first. The dimension of Q is the same as dimension of H:

$$\dim(Q) = \dim(H) = N \times m, \tag{3.11}$$

where N is the number of disks and m is the number of steps in the load balancing system. Of course, that if we apply the parallel strategy matrixes H

and Q do not make any sense, because we will not know how many commands are sent to which device.

For mathematical problem definition it is convenient to introduce the functional of time. Let the *functional of time* be defined as the functional

$$T = \sum_{j=1}^{m} t_j, \tag{3.12}$$

where t_j is the time for sending $\sum_{i=1}^{N} h_{ij}$ commands at j step of dynamical load balancing process.

The main idea of the optimization problem is to minimize the functional of time T

$$\min_{F,H,Q,s} T. \tag{3.13}$$

According the formula 3.7 current minimization problem can be rewritten as

$$T \xrightarrow[F,H,Q,s]{} T_{min}. \tag{3.14}$$

That means that the application of dynamic load balancing algorithm should make the program faster and the value of T should come close to the minimum time T_{min}. The algorithm should use F, H, Q, s as parameters for making the program faster. As it was mentioned usage parallel strategy removes some of these parameters and the amount of megabytes F, which we need to send to erase all N disks, starts to be the most important value. But complete erasure F also hides some interesting parameters such as amount of sending commands and buffer size of the command, which will play quite important role in the following paper.

3.3 Technical limitations

Every device has its own technical limitations. For example, a car has a limitation for the speed, capacity of petrol tank, maximum engine power and so on. Discussing the SCSI controllers every device has special characteristics, which we can compare with characteristics of the car.

Let introduce some parameters, which can influence on the communication between controller and disk. In this paper the word SCSI is used a lot, but there was no clear explanation what is it. As it is written in the beginning of the paper SCSI is Small Computer System Interface. The main word in this abbreviation is interface, which shows *how* the devices can be connected with each other. Lets again consider the Figure 3.1 and the situation that we write some data from CPU to the disk. Controller b has 2 interfaces, because it is connected to the CPU and disks d. Let call bus a as *input* interface and buses c as *output* interface. For the disks d there is only input interface, that is why it is possible to skip the word *input* and use only interface. So, if there is a sentence, which includes "the controller has SCSI interface", that means that the controller has the possibility to connect SCSI disks and the output interface is SCSI. Moreover, because controller has 2 connections there is also input interface, which can be, for example, different modifications of PCI (Peripheral component interconnect) such as PCI-X, PCI64, PCI Express or some others. In general words if there is a discussion about SCSI interface it means only the buses c.

Moreover, SCSI is also a data protocol, which shows what commands can be sent through buses c. Depends on the previous content SCSI interface connection has different technical limitations. There are several SCSI interfaces such as SCSI-1, Fast-Wide SCSI, Ultra2 SCSI, Ultra-320 SCSI and

some others, but in this research Ultra-320 SCSI interface will be considered more, because the testing device is supported that interface. Nowadays there is interface with faster bandwidth - 640 MB/s - but it did not get popular, because it supports maximum 2 devices per cable.

Ultra320 SCSI [6] is the seventh generation of SCSI I/O (Input/Output) technology. The dominant feature is that the speed is increased to 320 megabyte per second (MB/s). In Ultra320 SCSI devices all support packetized protocol and may support Quick Arbitration and Selection (QAS). Expander communications techniques have also been defined. In general Ultra320 shows that only buses c from the Figure 3.1 have this speed and not disks d, and of course, not controller b. In the ideal situation to get the maximum speed of the device communication we should fulfil the following condition:

$$S_a > S_b > S_c > S_d. \tag{3.15}$$

Otherwise there is a question, how controller is going to manage several disks if it has the same speed. Currently there are different PCI buses, which can have the maximum bandwidth up to 4 GB/s. Moreover, there are also some possibilities to increase the performance of PCI [1]. But even it is not the limit, because there are also some other buses, such as QPI (QuickPath Interconnect) and HyperTransport with different modification, which can easily have the bandwidth around 25 GB/s. By now the maximum bandwidth has HyperTransport 3.1 with value 51,20 GB/s. Of course, that bus depends on the frequency quite a lot. HyperTransport 3.1 has the maximum frequency 3,2 GHz. Depends on the data content and the number of the disks the data for the transport through bus a can increase, but the numbers show that it seems that it is not a problem, but still it is better to keep this fact in mind.

All SCSI devices are always backward compatible, which means that when

newer SCSI devices are attached to a system with devices from a previous generation, the newer devices will fall back to the maximum operating speed of the older generation when they are talking. When not talking to older devices, the newer devices operate at their normal speeds. That means that if one of SCSI drives from the Figure 3.1 works on Ultra160 interface, the controller, which supports Ultra320, will talk to the disk using Ultra160 interface.

Most of the time in the hardware if the device or cable can be connected to some place that means that it will work. The same situation is with SCSI interfaces, but for some generations connectors are the same. For the interfaces Ultra2 Wide, Ultra3, Ultra320 and Ultra640 the connectors can be 68-pin and 80 pin which belongs to the connection type SCA/SCA-2 (Single Connector Attachment). The examples of the connectors are presented on the Figure 3.2.

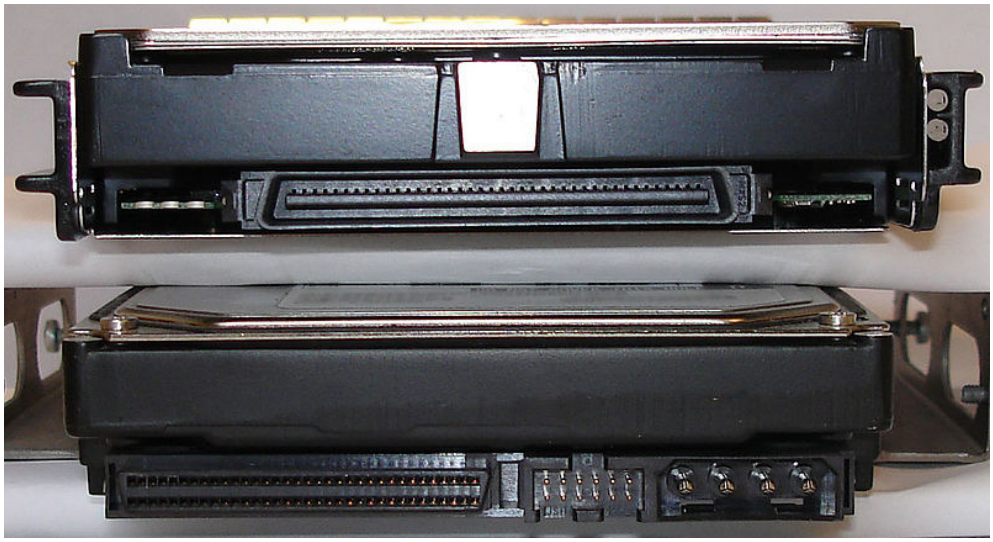

Figure 3.2: 68-pin and 80-pin SCA connectors

Mostly the discussion about technical limitations contained the information about the buses a and c, but what about controller and disks? Let start from the disks. In this chapter it was already introduced that each disk i has capacity d_i, speed S_d and special interface. But there are some other parameters, which could be also important. It is *cache*, *spindle speed*, average *seek time* for reading and writing, connector type and dimension. The last parameter is not very interesting, because we assume that there is enough space.

In general cache is used in many situations, for example, in operation systems, in controllers, in disks and some other places. Cache is the part of memory, which has very fast speed access [12]. For the disks it means that cache is stored before the physical hard disk platter, which gives the possibility to get the data much faster. Currently disks can have cache with capacity from 8 to 64 MB. Of course, that big cache gives higher performance for the disk. Moreover there are different cache algorithms such as LRU (Least Recently Used), MRU (Most Recently Used), LFU (Least-Frequently Used), Direct-mapped cache and some others [20], [17]. The aim of every cache algorithm is to manage the data in the cache and show, which data block to delete and which one to keep. Depends on the disk it can be applied different cache algorithms, which could give the advantage or disadvantage in the current problem. Also, depends on the number of the disks maybe load balancing system should find some tricky steps through disk cache, which can make the process faster.

Another important characteristic of disk is spindle speed, which shows the frequency of rotation. This value is measured in revolutions per minute (RPM). The value of spindle speed shows how many full rotations were completed in one minute around a fixed axis. Usually for ATA disks spindle speed can be

5400 or 7200 rpm (90 or 120 Hz) and in this case SCSI disks win, because their spindle speed can be 10,000 or 15,000 rpm (160 or 250 Hz). This value is constant and is set by manufacture. So, there is no reason to try make it higher, the only possibility is to take the disk with higher spindle speed.

Both of described parameters cache and spindle speed are influenced on the seek time of the disk. Seek time means the time, which needs for the head of the disc to move to the right position. It is clear that this time can be only average, because in different situations there are different values of cache, spindle speed and data task. In general there are two values of seek time - one for reading operations and another for writing, but usually it is only one value for both operations. Most of the time in the research it will be considered only as a writing seek time, but the value of seek time for the reading should not be thrown out. There are two seek measurements called track-to-track and full stroke. The track-to-track measurement is the time required to move from one track to an adjacent track. This is the shortest (fastest) possible seek time. In hard disk drives (HDD) this is typically between 0.2 and 0.8 ms. The full stroke measurement is the time required to move from the outermost track to the innermost track. This is the longest (slowest) possible seek time.

Lets consider the controller now and figure out what parameters this device has. First of all, it has the possibility to connect buses a and c, which was decided to call as input interface and output interface, which also follow the specific parameter, such as speed. In the literature it is also possible to see that input interface is called as host interface. Secondly, it is the amount of RAIDs, which is supported by this controller. There are also some more parameters, which are still important, such as cache, cache function or algorithm, maximum amount of physical disks, maximum amount of logical

disks. We discussed what does cache mean and controllers cache has more power in comparison with disk cache. Currently cache of the controllers can have values up to 512 MB. Moreover, some controllers have even their own cache algorithms, which make it stronger. But this fact is obvious, because the controller should have enough power to communicate with all connected disks.

There is one parameter, which should be marked out. It is the bandwidth of the controller, which definitely influences on the performance of the device. But most of the time in the parameters of the controller it is possible to find only the bandwidth of the input and output interfaces. It is like this, because currently the controllers are so powerful, that they can manage all the data, which comes to the device. That means that the speed of the controller is not the bottleneck and is not very important in our case.

3.4 Theoretical estimations

Lets model the real situation and try to estimate how long time does it take to write some data to the disk. Let consider that there is a Compaq SCSI disk BD01864552 with the following parameters:

- Capacity: 18.2 GB
- Spindle speed: 10,000 RPM
- Interface: Ultra3 SCSI
- Seek time: 3 ms

From these parameters mostly we will focus on the interface, which has a speed 160 MB/s. That means that the speed of the bus c is equal to 320

MB/s. Basically all data on disks are divided to the sectors. Most of the time the size of the sector is 512 bytes, but in some cases this value can be different. In this research we consider that the sector size has 512 bytes value. We will focus on the parameters of the commands and the disks because the controller should not be a bottleneck, but we still need to think that there are some limitations also in the controller.

WRITE SAME command Let consider the WRITE SAME 10 command, which comes from the CPU [9]. This command has 32 bits for the LBA (Logical block addressing) and 16 bits for the Number of Logical Blocks. The main huge advantage of that command is that we can send the same buffer to several logical blocks only with one command. It gives us a possibility to write much more data than we really send by the command.

We can calculate the maximum amount of data that we can write with one WRITE SAME 10 command to the disk:

$$\frac{2^{16} * 512}{1024^2} = 32MB. \tag{3.16}$$

Let consider that the model 3.1 consists of 1 disk and we do not need any other commands except WRITE SAME one. That means that in condition 3.3 parameter c_2 will be equal to 0. We also assume that we do not need any special commands for the communication and c_3 is equal to 0. It gives a possibility to calculate complete erasure F as a multiplication of command c_1 and amount of commands m_{11}. Because we know the capacity of the disk we could compute the number of commands m_{11} that we need to send for erasure 18.2 GB disk.

$$m_{11} = \frac{F_1}{c_1} = \frac{18.2 * 1024}{32} = \frac{18636.8MB}{32MB} = 582.4. \tag{3.17}$$

For performance of real erasure the data should be divided for filling the last part of the disk, because we could not send 0.4 command. The main aim of the research is testing the erasure time that is why we will not take this little part into consideration and will use the approximate value of 582.

The size of WRITE SAME 10 command is 10 bytes. Including the buffer, which is 512, we get that one whole command is 522 B. So, to erase one 18.2 GB disk we need to send the following amount of data:

$$F_1 = (10 + 512) * 582 = 303804B = 304kB. \tag{3.18}$$

That means that for complete erasure of one disk we need to send only 304 kB. Another command WRITE 10 does not have so huge advantage, but lets compare these two commands, because in some cases WRITE SAME 10 command does not work.

WRITE command Let consider another pass through command WRITE 10 [9]. The structure of this command is completely the same in comparison with WRITE SAME 10 command, but some parameters have different values. For example, WRITE SAME 10 command has operation code 0x41 and WRITE 10 command has 0x2A. The main difference of these 2 commands is that with WRITE 10 command we need to send whole buffer all the time. The buffer length is calculated by the following Formula:

$$Buffer\ length = Sector\ size * Tranfer\ length. \tag{3.19}$$

In our situation the disks have constantly sector size 512 and this value depends only on the disk. But buffer length can be varied by transfer length, which can have a value up to 65536. That means that if we decide to set the maximum transfer length we could send 32 MB with one WRITE 10

command. In comparison with WRITE SAME 10 command, which has 512 B buffer, our buffer can increase to 512*65536, which is quite big value if we consider that the controller connect to several disks. Usually, in Linux driver of the controller there is a value of maximum buffer, which is 4096*512 in our case. It can be optimal for some amount of disks, but it should be checked, because this is one of the parameters that we can influence.

If a lot of data go with one command through the controller it can make the erasure process slower because the cache of the controller will be full of messages, waiting in the queue. We should influence on the erasure speed by this parameter and find the optimal value during testing. By now this value is set to 256, which means that we send 128 kB at once. We can calculate how many commands we need to erase one 18.2 GB disk by WRITE 10 command with 128 KB buffer:

$$m_{11} = \frac{F_1}{c_2} = \frac{18.2GB}{128kB} = \frac{18636.8MB}{0.125MB} = 149094.4 \tag{3.20}$$

The derived value is 256 times bigger than the amount of commands that we need to send for erasure by WRITE SAME. It is obvious, because the current value of buffer length is 256 bigger. In the chapter 4 we will see the results of erasure testing with different transfer lengths.

Chapter 4

Application of load balancing strategies to the HP Smart Array 642

This part of the thesis shows how is it possible to apply load balancing during communication between SCSI controller and disks. We discuss several strategies of sending the commands and try to estimate which one is better. Moreover, we suggest to apply a dynamic load balancing system for finding the optimal values for the erasure. That system solves our optimization problem 3.13, which is the main task in this research.

4.1 HP Smart Array 642

HP Smart Array 642 is a SCSI RAID controller, which supports protocol Ultra-320 SCSI [3]. This device has one internal and one external VHDCI

(Very-High-Density Cable Interconnect) SCSI ports. The internal port gives the possibility to connect up to 6 internal disks and the external one up to 14 disks. That means that HP Smart Array 642 controller supports 20 disks. Maximum capacity for all disks in total is 6 TB. Controller has PCI-X bus, which has a 133-MHz frequency. That means that the maximum bandwidth of the bus is 1 GB/s. Moreover, the architecture is 64-bit.

The Figure 4.1 presents the described controller.

Figure 4.1: HP Smart Array 642 controller

4.2 Load balancing strategies

This section presents a description of the methods for solving the problem presented above in the chapter 3. The solutions that we offer are simple, but effectively show the results in practice. We applied them for testing the erasure of several disks and find out which one is optimal. The main problem of the research is to minimize the functional of time T or prove that

it is not possible because of some limiting factor. That means that we try to find the fastest way of sending the write commands to the disks through the controller.

We applied three load balancing strategies during testing the erasure of the disks behind HP Smart Array 642 controller. In several papers [7], [11] there were some discussions about strategies. Because of the specificity of the problem it was difficult to apply them in our case, but it gave an idea how it should work. First two strategies, which we applied, are sequential and the third one is parallel, which works more effectively than others. Because first two strategies are much slower than the third one, we tested them with 5 disks connected with the backplane to the internal port of the controller. The third strategy was tested with the enclosure of 14 disks connected to the external port of the controller. In both cases we chose 18.2 GB disks from Compaq. The section 3.4 discusses the theoretical expectations about these disks.

Lets consider the strategies in more detail. The first strategy performs the complete erasure disk by disk. That means that until we do not erase the current disk the program does not start the erasure of the next disk. From the section 3.4 we know that for the erasure one 18.2 GB disk we need to send 582 WRITE SAME 10 commands. According to the point that for testing first strategy we took 5 disks we get the following matrix H_{s1} from the model

3.8 for complete erasure:

$$H_{s1} = \begin{pmatrix} 582 & 0 & 0 & 0 & 0 \\ 0 & 582 & 0 & 0 & 0 \\ 0 & 0 & 582 & 0 & 0 \\ 0 & 0 & 0 & 582 & 0 \\ 0 & 0 & 0 & 0 & 582 \end{pmatrix} \tag{4.1}$$

From the matrix H_{s1} we can conclude that first strategy has only 5 steps in the algorithm.

Second strategy sends one command per disk and then changes the disk. That means that in this case matrix H is completely different:

$$H_{s2} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.2}$$

For the second strategy the dimension of matrix H is 5×2910. Value 2910 came from multiplication of number of disks and number of commands that we need to send for the erasure. From the logic of sending commands the devices should be changed quite often, which can be a disadvantage of this strategy.

The third strategy is using parallel computing, which makes the process for complete erasure much faster. The application creates separate thread for each disk and start sending WRITE SAME commands to that disk. It is obvious that during increasing the amount of disks the value of functional of time will increase too, but slowly, because threads are working completely separately. However, important limitations are the speeds of the disks and

the bus. We can see from Figure 3.1 that separate buses c give the possibility to send data from the controller b to the disks d. In our situation, only one bus c exists, which is connected to the enclosure containing several disks. If we write information in parallel to several disks the bus speed can be the limitation for data transfer. The maximum speed of the current bus is 320 MB/s. For the third strategy it is difficult to present matrix H, because we do not know in which order WRITE SAME commands come to the disks.

4.3 Planning dynamic load balancing system

During whole research we try to find the optimal parameters for sending write commands to the disks, but it seems that our conclusions about them can be wrong before some testing or special calculations. As we mentioned in the beginning of the paper, load balancing system should take care of these things. Lets discuss what kind of static and dynamic load balancing systems we could provide for our situation. There are so many parameters which depend on the speed of erasure that it is much better to use dynamic load balancing system. However, it is also possible to apply static load balancing system using some special conditions. Currently we consider the situation when we send WRITE 10 commands to several disks using the parallel strategy. We do not take in consideration WRITE SAME command, because the bottleneck is the bus in this case and we cannot influence on that.

Main condition that we should focus on is knowledge about the cache of the controller. If we know this parameter before the erasure process it is possible to calculate the optimal buffer and optimal number of disks for erasure. The problem is that getting cache from the controller is very specific task

and sometimes maybe even impossible one. However, we can manually find the maximum value for the transfer buffer inside the Linux driver. In our situation for the HP Smart Array 642, which uses the driver cciss, this value is defined as $MAX\_KMALLOC\_SIZE(4096*512)$, which means that the maximum buffer can not be larger than 2 MB. We do not know exactly if it is cache, but the value looks quite similar to the cache size. If we consider this value as cache we can calculate what is the closest value for the transfer length to the optimal one using several disks:

$$\frac{MAX\_BUFFER}{SECTOR\_SIZE * MAX\_DISKS} \tag{4.3}$$

In our case we send the buffer with the value of power of two, that is why we should find the value, which is closest from the bottom, otherwise we exceed the cache size. If we consider 14 disks, we can see that the value $4096 * 512/512 * 14 = 292.57$ is closest to 256. Blancco software sends the buffer exactly with the transfer length 256, but it is optimal only when the number of disks is more than 8. Lets calculate the transfer length in case that we have 8 disks: $4096 * 512/512 * 8 = 512$. This fact explains that we should use another transfer length to get an optimal way of sending commands. Moreover, we understand that for 4 disks we need 1024 and for 2 disks - 2048. We proved this theory by making several tests. The Figure 5.12 shows that if we base on our theory Blancco software sends the optimal buffer only for disks 9-14. If we have less than 9 disks in the enclosure, we can erase them faster using another transfer length.

Based on our theory if we have 9 disks and transfer length is 512, the buffer will exceed the limit of 2 MB and the erasure process should go slower than with transfer length 256. However, in practise, the results of testing show that even for disks 9-14 the buffer with transfer length 512 allows to erase

the devices faster than with transfer length 256. The Figure 5.13 explains it more clear. From these results we could make a conclusion that some other parameters, that we did not include, influence on the speed. Anyway, suggested static load balancing system helps to improve the erasure, which gives a possibility to do it faster than Blancco software currently does. Moreover, if we base on the testing results, we could provide even faster load balancing system.

In some versions of Blancco software it is possible to add disks during the erasure process. In this case our static load balancing system should recalculate the optimal transfer length and set it correctly to the commands during the erasure process. From the programming point of view it could be difficult to access the function, which performs the erasure, but not impossible. If we cannot influence on the transfer length of the currently working disks, it could be better to wait until the application erases them, because the bus can be already filled completely. So, if we decide to add more commands because of new disks, we could come to the situation that the commands will stay in the queue before coming to the bus. It can make the speed of the erasure much slower.

If we consider the situation that we can influence on the transfer length during the erasure process, we get that our static load balancing system transforms to the dynamic load balancing system. The purpose of this load balancing system is to calculate and set the optimal transfer length for the write command depending on the amount of disks. In the situation of ending the erasure this load balancing system can help us one more time. Of course, that if we have disks, which have the same size and rotation speed, there is no reason to make any more calculations for searching the optimal parameters after some disks are erased, because the time should be almost equal to each

other. But if we have disks with the different capacities, it makes sense to set another transfer length after some disks are erased. That means that suggested dynamic load balancing system should work in three situations:

1. Before the erasure
2. During the erasure, when some disks are added or stopped
3. After some disks finish the erasure

However, if we consider the situation that the optimal value was found and look to the Figure 5.6 we can notice that the maximums of 2 curves belongs to different disks. It gives an idea that not only speed of the disk and buffer length influence on the erasure speed. It could be possible to find the way of sending commands, which will give the medium time of all disks, but it can be difficult because for each disk a separate thread exists. This medium time will give us a great advantage, because the time for complete erasure will decrease and it will be almost equal for all disks. Moreover, our load balancing system will not need to analyse the optimal transfer length for last disks, which do not finish the erasure yet.

One of the possible ways of trying to find the medium time is to analyse each thread. For example, we can make that each thread will return the percentage of process with some step. Using this value we can analyse which disk works faster or slower. After that the dynamic load balancing system stops the erasure process for some fast disks. Thus, we will get some free space in the bus and could increase the transfer length for the slow disk. This action could give us an advantage if the limit is not a disk speed. After the next returns from the threads if the result of the slow disk is larger than fast disks we turn on the erasure for all disks. Because user is allowed to stop or add disks this strategy can give several problems for the software developer.

Chapter 5

Experiments and results

In this chapter we present the test results of the application, which sends WRITE 10 and WRITE SAME 10 commands with different parameters. We made all the tests with WRITE 10 command, when the disks were connected with the enclosure to the controller. But some tests with WRITE SAME 10 command were done with the backplane. Mostly we are interested in decreasing the time of complete erasure, but sometimes, because of the different speed of the disks, the results started to be useless and strange, that is why we started to calculate also the time of erasure for single disks.

5.1 Results with WRITE SAME 10 command

Using WRITE SAME 10 command we tested 3 different strategies that we considered before. The tests for the first and second strategies were done with 5 disks, connected with the backplane to the internal port of the HP Smart Array 642 controller. The third strategy was tested with 14 disks, connected with the enclosure to the external port of the controller.

During testing erasure with first strategy we got the results presented in the Table 5.1.

Table 5.1: Results for complete erasure by first strategy

Disks	Capacity	Time	Average time/disk	Average speed
1	18.2 GB	8 min 32 sec	8 min 32 sec	35.5 MB/s
2	36.4 GB	18 min 12 sec	9 min 6 sec	33.3 MB/s
3	54.6 GB	28 min 5 sec	9 min 21 sec	32.4 MB/s
4	72.8 GB	39 min 7 sec	9 min 46 sec	31.0 MB/s
5	91 GB	48 min 45 sec	9 min 45 sec	31.1 MB/s

Figure 5.1 presents the results from the Table 5.1 more clearly.

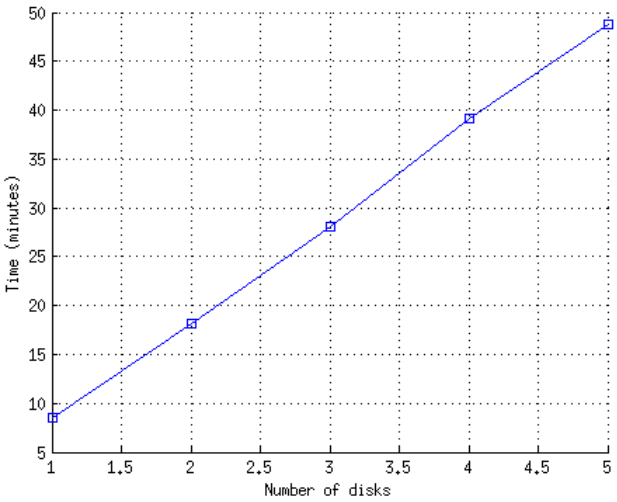

Figure 5.1: Time results for complete erasure by first strategy

During testing erasure with second strategy we got the results presented in the Table 5.2.

Table 5.2: Results for complete erasure by second strategy

Disks	Capacity	Time	Average time/disk	Average speed
1	18.2 GB	10 min 54 sec	10 min 54 sec	27.8 MB/s
2	36.4 GB	20 min 42 sec	10 min 21 sec	29.3 MB/s
3	54.6 GB	30 min 41 sec	10 min 13 sec	29.6 MB/s
4	72.8 GB	40 min 25 sec	10 min 6 sec	30.0 MB/s
5	91 GB	50 min 10 sec	10 min 2 sec	30.2 MB/s

Figure 5.2 presents the results from the Table 5.2 more clearly.

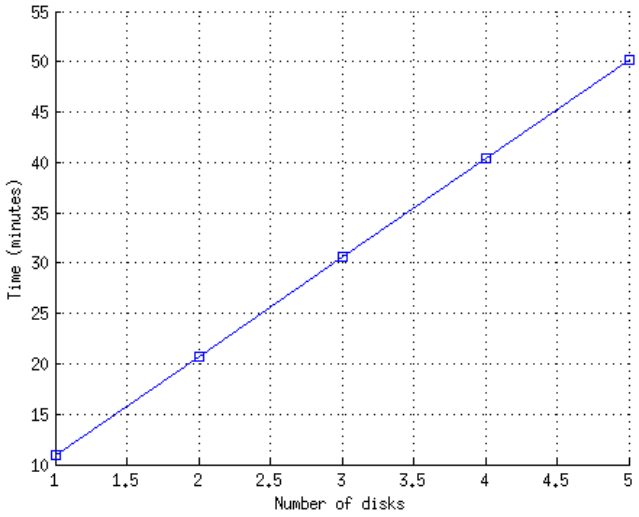

Figure 5.2: Time results for complete erasure by second strategy

During testing erasure with third strategy we got the results presented in the Table 5.3.

Table 5.3: Results for complete erasure by third strategy

Disks	Capacity	Time	Average time/disk	Average speed
1	18.2 GB	6 min 5 sec	6 min 5 sec	49.8 MB/s
2	36.4 GB	8 min 29 sec	4 min 15 sec	71.5 MB/s
3	54.6 GB	9 min 45 sec	3 min 15 sec	93.3 MB/s
4	72.8 GB	10 min 57 sec	2 min 44 sec	111.2 MB/s
5	91 GB	10 min 57 sec	2 min 11 sec	138.5 MB/s
6	109.2 GB	10 min 57 sec	1 min 50 sec	166.2 MB/s
7	127.4 GB	10 min 57 sec	1 min 34 sec	194.0 MB/s
8	145.6 GB	10 min 57 sec	1 min 22 sec	221.7 MB/s
9	163.8 GB	10 min 58 sec	1 min 13 sec	249.0 MB/s
10	182 GB	10 min 57 sec	1 min 6 sec	277.1 MB/s
11	200.2 GB	11 min 0 sec	1 min 0 sec	303.4 MB/s
12	218.4 GB	11 min 0 sec	55 sec	331.0 MB/s
13	236.6 GB	13 min 26 sec	1 min 2 sec	293.6 MB/s
14	254.8 GB	13 min 28 sec	58 sec	315.5 MB/s

Figure 5.3 presents the results from the Table 5.3 more clearly.

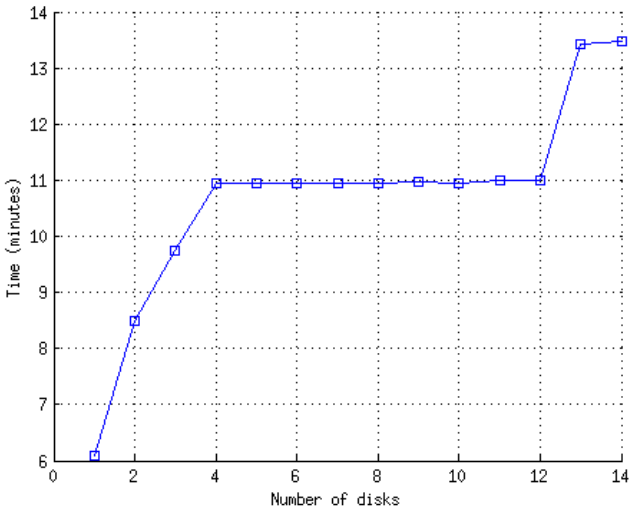

Figure 5.3: Time results for complete erasure by third strategy

From the Figure 5.3 we can see that for disks from 4 to 12 the time of complete erasure is constant. Third strategy sends the commands parallel to several disks. That means if there is one slow disk in the task, the whole time of the erasure will be slow. Therefore, we decided to calculate the time for erasure of single disk.

During testing erasure for single disks with third strategy we got the results presented in the Table 5.4. Moreover, in the Table 5.4 we present the time of erasure for single disk during complete erasure.

Table 5.4: Results for erasure of single disks by third strategy

Disks	Capacity	Time	Time during complete erasure
1	18.2 GB	6 min 5 sec	6 min 5 sec
2	36.4 GB	9 min 39 sec	10 min 59 sec
3	54.6 GB	9 min 44 sec	9 min 43 sec
4	72.8 GB	10 min 57 sec	10 min 59 sec
5	91 GB	10 min 55 sec	10 min 53 sec
6	109.2 GB	9 min 42 sec	9 min 42 sec
7	127.4 GB	6 min 37 sec	6 min 37 sec
8	145.6 GB	8 min 29 sec	8 min 29 sec
9	163.8 GB	6 min 38 sec	6 min 37 sec
10	182 GB	6 min 5 sec	6 min 5 sec
11	200.2 GB	10 min 13 sec	10 min 12 sec
12	218.4 GB	9 min 51 sec	10 min 31 sec
13	236.6 GB	9 min 55 sec	9 min 55 sec
14	254.8 GB	6 min 6 sec	6 min 5 sec

Figure 5.4 presents the results from the Table 5.4 more clearly.

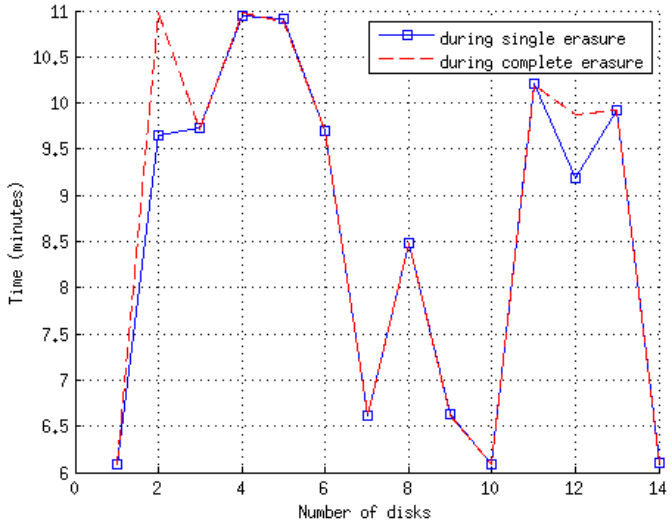

Figure 5.4: Time results for complete erasure by third strategy

By replacing disks 4,5 and 11 we got different results for single disks during complete erasure, which are shown on the Figure 5.5.

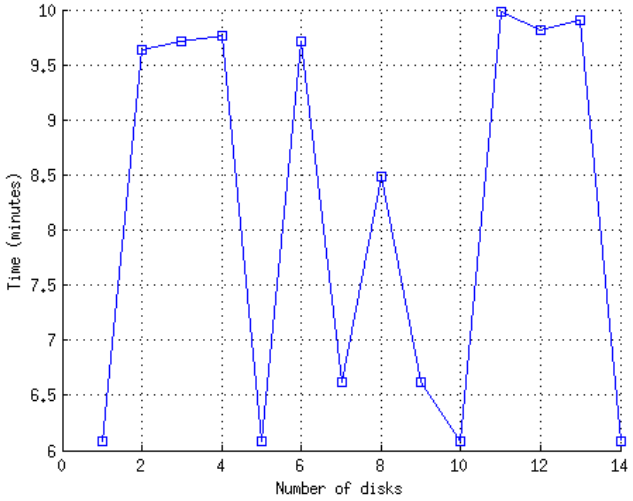

Figure 5.5: Time results for single erasure by third strategy

5.2 Results with WRITE 10 command

During research with WRITE SAME 10 command we understood that the parallel strategy works fine with that controller that is why experiments with WRITE 10 command were done only with the third strategy. Several tests were done with WRITE 10 command with transfer length 256. During testing erasure for single disks with third strategy we got the results presented in the Table 5.5.

Table 5.5: Results for erasure of single disks by third strategy

Disks	Capacity	Time	Time during complete erasure
1	18.2 GB	19 min 52 sec	53 min 43 sec (-3)
2	36.4 GB	23 min 24 sec	45 min 6 sec (+4)
3	54.6 GB	23 min 28 sec	49 min 25 sec (+3)
4	72.8 GB	24 min 43 sec	54 min 15 sec (0)
5	91 GB	24 min 38 sec	57 min 39 sec (-1)
6	109.2 GB	23 min 28 sec	60 min 19 sec (+3)
7	127.4 GB	20 min 27 sec	54 min 27 sec (+2)
8	145.6 GB	22 min 15 sec	57 min 47 sec (-3)
9	163.8 GB	20 min 24 sec	42 min 40 sec (+2)
10	182 GB	19 min 50 sec	50 min 37 sec (-5)
11	200.2 GB	24 min 1 sec	33 min 0 sec (-5)
12	218.4 GB	23 min 32 sec	35 min 10 sec (+1)
13	236.6 GB	23 min 39 sec	38 min 14 sec (-6)
14	254.8 GB	19 min 51 sec	52 min 40 sec (-5)

Figure 5.6 presents the results from the Table 5.5 more clearly.

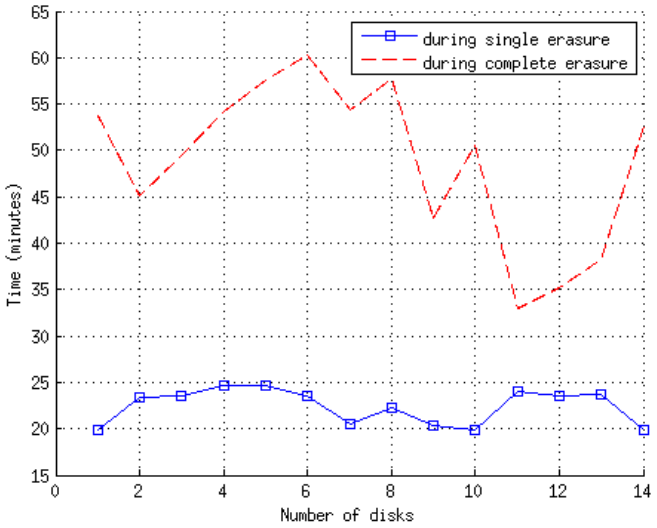

Figure 5.6: Time results for complete erasure with transfer length 256

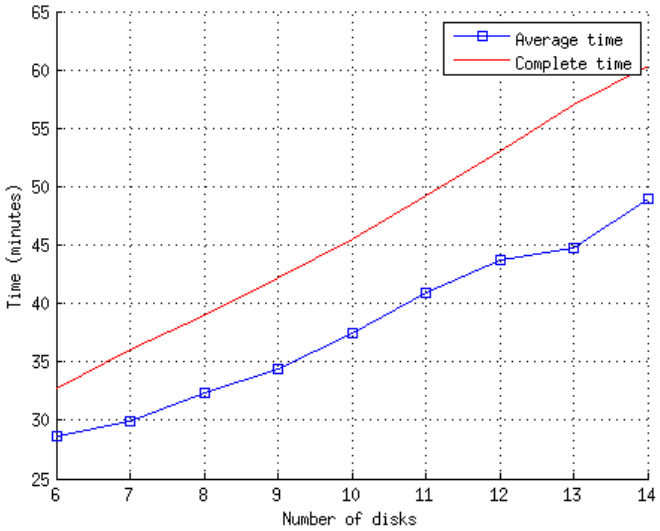

Figure 5.7: Time results for the complete erasure depending on the amount of disks

Figure 5.8 presents the time results for the disk 1 depending on the different transfer length.

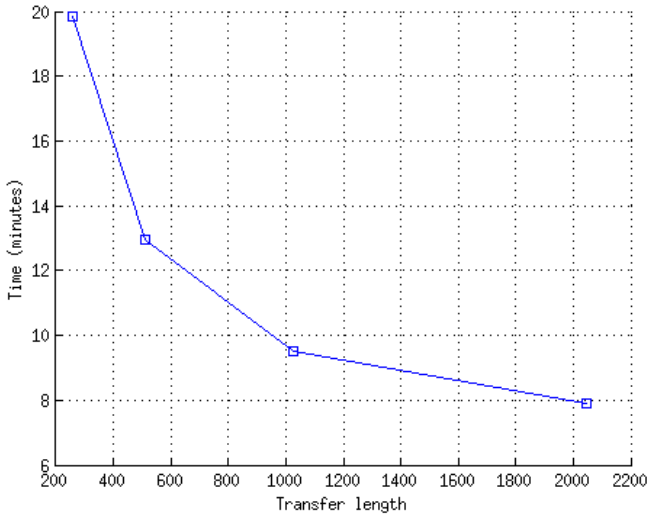

Figure 5.8: Time results for the disk 1

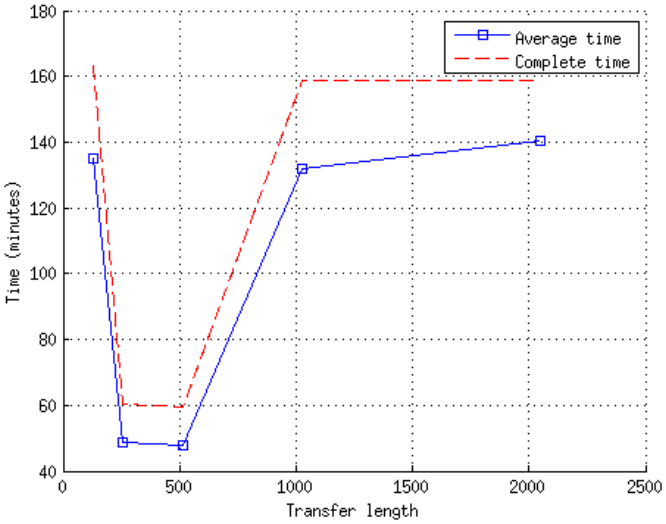

Figure 5.9: Time results for average erasure and complete erasure depending on the transfer length

Figure 5.10 presents the time results of complete erasure with transfer length 256 and 512.

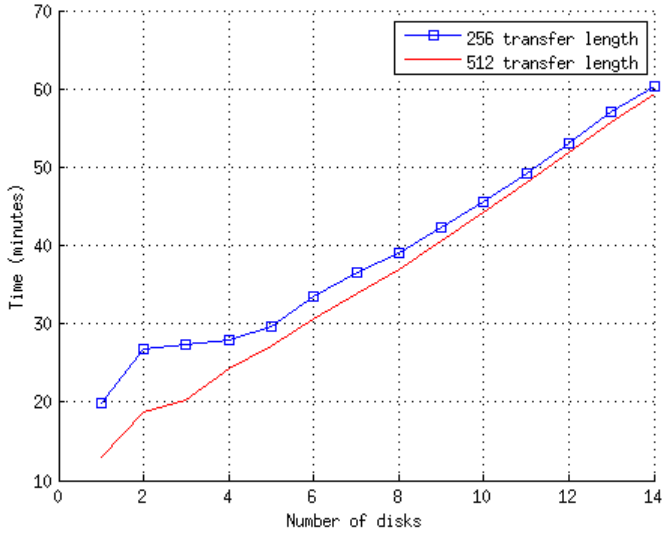

Figure 5.10: Results for complete erasure depending on the transfer length

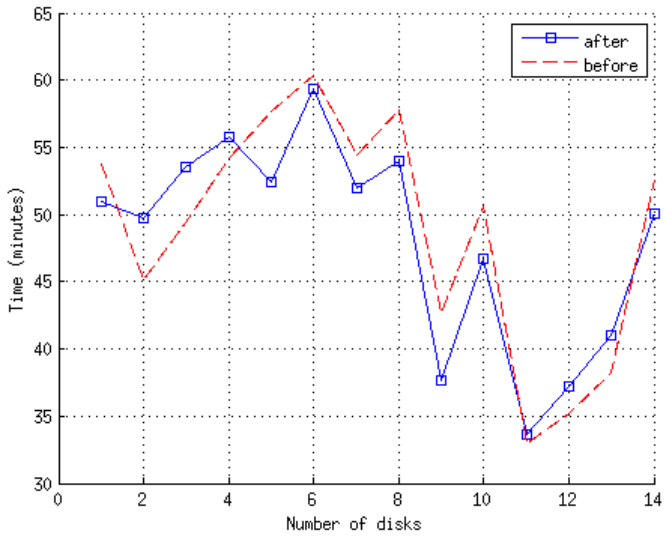

Figure 5.11: Results after replacement of the disks 4, 5 and 11.

Figure 5.12 presents the time results of theoretical optimal complete erasure in comparison with the results with transfer length 256.

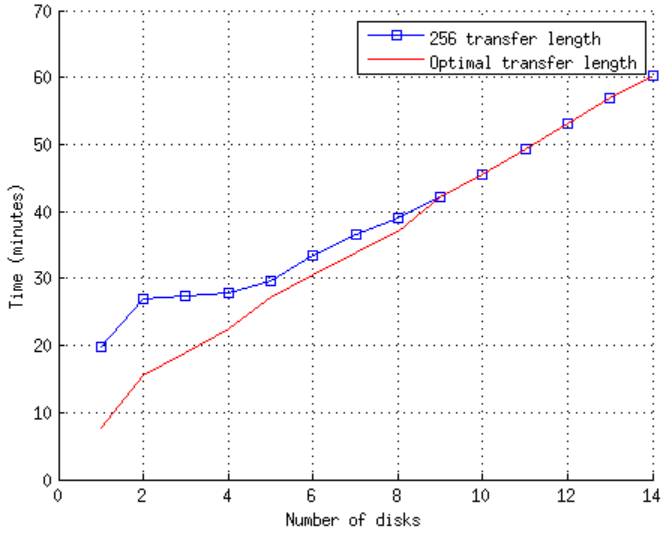

Figure 5.12: Results for theoretical optimal erasure

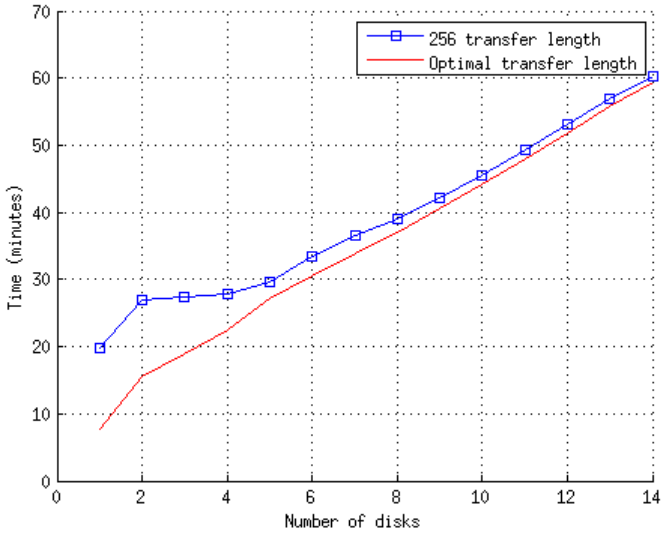

Figure 5.13: Results for practical optimal erasure

5.3 Analysis of the results

After testing with WRITE SAME 10 command we can notice that the second strategy is working slower than the first one, which is obvious because it needs to change the devices quite often. However, if we compare the Figures 5.1 and 5.2 we can see that the difference is only about 1 minute, which is insignificant value in comparison with time for complete erasure of 5 disks, which takes around 50 minutes. The third strategy works without any problems, but the results of several disks are quite different, which gives an idea that there are some other parameters of software or hardware, which influence on the speed of the erasure.

We can see from the Figures 5.1 and 5.2 that the time of first and second strategies is increasing linearly. We could not say the same about the third strategy because the disks have different speed or some other parameters affect on the maximum speed of the disk. Figure 5.4 shows that the difference is very low between the times of erasure for single disk and for single disk during complete erasure. That means that the bus is not a bottleneck in the situation, when we send WRITE SAME 10 command and 14 disks are connected to the controller. Thus, only rotation speed of the disks is a limiting factor. The Figure 5.5 proves this fact more time by the results for single disk during complete erasure after replacement of 3 slow disks. The results show that the time of complete erasure decreased from 11 to 10 minutes. However, it is still strange that the difference of the time of erasure for several disks can be even 4 minutes, which is almost half time of the complete erasure, but the disk parameters are completely the same.

The testing results do not depend on the application running and the erasure time is not divided randomly to different disks. Table 5.5 proved this fact

by results from the same test running several times, which gave the whole difference with -13 seconds for erasure of 14 disks by WRITE 10 command. That means that even for erasure of the fastest disk 13 seconds will take only 0.01 of whole erasure. That is why we could ignore this difference.

The Figure 5.6 presents how big is the time difference for single disk if we decided to erase 14 disks together using WRITE 10 command with transfer length 256. The lower curve shows us that for erasure one of these 14 disks we need from 20 to 25 minutes. However, if we launch the application together with 14 disks, for the disk 2 it will take 45 minutes, but for disk 10 already 50 minutes. Anyway, the complete erasure takes 60 minutes, which is just 3 times bigger than single erasure for the fastest disk. The Figure 5.7 presents the graphic of erasure time depending on the amount of disks. From these results we can make a conclusion that addition of 1 disk gives us addition of 3 minutes to the complete erasure.

We can notice that during complete erasure by WRITE SAME 10 command disks 2, 4 and 5 showed the worst time. In the same time during complete erasure by WRITE 10 command disks 6 and 8 showed the worst time. That gives an idea that there are some other parameters that also depending on the time of erasure. Moreover, Figure 5.11 shows how changes the results after replacement disks 4, 5 and 11. We can conclude that time for complete erasure decreased for 1 minute and the same disks show the maximum and minimum times. It is also strange that during single erasure by WRITE 10 command disks 1 and 14 were the fastest and during complete erasure their results are very closed to very slow ones.

Figures 5.8 and 5.9 show the results of testing different transfer length, which is one of the parameters that we can influence. To summarize these results we

can say that transfer lengths 512 and 256 gives almost the same results, but the optimal value for 14 disks is 512, which is easier to see from Figure 5.10. However, when we do not have so many disks, we could apply the suggested load balancing system from the section 4.3 and calculate the optimal value by the formula 4.3. This load balancing application will help to find the optimal transfer length for making the erasure faster. Thus, if we apply only our theoretical calculations the Figure 5.12 shows the results of time improvement. Moreover, using our practical results we can suggest to apply the results from the Figure 5.13, which improves the Blancco software for any amount of disks. The results shows that the optimal transfer length, which was calculated by testing and load balancing system, makes the erasure faster for WRITE 10 command from disk 1 to disk 14.

Chapter 6

Conclusion

The aim of the research was to analyse the communication model between SCSI controller and disks and find out the optimal way of sending WRITE and WRITE SAME commands. Several strategies of sending commands were applied, which gave the food for understanding where the bottlenecks are hidden. Moreover, these results supported to build the dynamic load balancing system, which can find the optimal values for commands.

The tests showed that if we send commands in series, the erasure time increases linearly. After finish of sending consecutive commands we decided apply the parallel strategy. SCSI controller and disks handled it without any problems that is why all the later experiments were done with the parallel strategy. The results gave a possibility to make a several conclusions. First of all, usage of WRITE SAME command allows to erase one 18.2 GB disk by sending only 304 kB of information and the time of the erasure depends directly from the disk speed. Secondly, the bottleneck of sending WRITE commands is the bus, because with that command we need to send the amount of data, which exceeds the sum of capacity of the disks. But that

fact gave a possibility to vary the parameters and different tests helped to find out that the size of sending buffer plays the main role. To make the process faster the buffer size should fit in the allocation memory of the SCSI controller that was found in the Linux driver. These facts played very important roles in building the suggested dynamic load balancing system, because it helps to calculate the optimal values for sending the commands for the erasure.

For the future research we suggest to figure out if we discussed all the parameters of this communication. For example, there is the question why the erasure times of the disks with the same properties are so different during single erasure. Moreover, it would be a great step forward to balance the results of single erasure, which gives a possibility to make lower the value of complete erasure. In this research only one SCSI controller was considered and it is possible that another controller will have similar behaviour, but with its own specificities.

Bibliography

[1] *Techniques for Increasing PCI Performance.* Intel Corporation, 1999.

[2] *RAID Technology Overview.* Hewlett-Packard Development Company, L.P., 2007.

[3] *HP Smart Array 642 Controller.* Hewlett-Packard, 2009.

[4] *Which RAID Level is Right for Me?* Adaptec, Inc., 2010.

[5] Ali M. Alakeel. *A Guide to Dynamic Load Balancing in Distributed Computer Systems.* IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, 2010.

[6] Michael Arellano. *Ultra320 SCSI: New Technology - Still SCSI.* SCSI Trade Association, 2001.

[7] Amnon Barak and Amnon Shiloh. A distributed load-balancing policy for a multicomputer. *Softw. Pract. Exper.*, 15(9):901–913, September 1985.

[8] Valeria Cardellini, Michele Colajanni, and Philip S. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 3(3):28–39, May 1999.

[9] T10 Technical Committee. *Working Draft. Revision 35. Information technology - SCSI Block Commands - 3 (SBC-3).* Technical Committee of Accredited Standards Committee INCITS, 2012.

[10] T13 Technical Committee. *Working Draft. Revision 4c. AT Attachment 8 - ATA/ATAPI Command Set (ATA8-ACS).* American National Standard of Accredited Standards Committee INCITS, 2007.

[11] Antonio Corradi, Letizia Leonardi, and Franco Zambonelli. Diffusive load-balancing policies for dynamic applications. *IEEE Concurrency*, 7(1):22–31, January 1999.

[12] Carlo Kopp. *An Introduction to SCSI Disk Performance.* Dr Carlo Kopp's Industry Publications, 2005.

[13] Lionel M. Ni, Chong-Wei Xu, and Thomas B. Gendreau. A distributed drafting algorithm for load balancing. *IEEE Trans. Softw. Eng.*, 11(10):1153–1161, October 1985.

[14] Abraham Silberschatz; James Peterson. *Operating System Concepts, 8th edition.* Addison-Wesley, 1982.

[15] Gary Field; Peter M. Ridge. *The Book of SCSI, 2nd Edition.* Publishers Group West, 2000.

[16] Asser N. Tantawi and Don Towsley. Optimal static load balancing in distributed computer systems. *J. ACM*, 32(2):445–465, April 1985.

[17] Ramakrishna Karedla; J. Spencer Love; Bradley G. Wherry. *Caching Strategies to Improve Disk System Performance.* Computer, 1994.

[18] Roy Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency*, 3:457–481, 1991.

[19] B. Litow; S.H. Hosseini; K. Vairavan; G.S. Wollfe. *Performance Characteristics of a Load Balancing Algorithm.* Journal of Parallel and Distributed Computing, 1995.

[20] Andrew S. Tanenbaum; Albert S. Woodhull. *Operating Systems: Design and Implementation (Third Edition).* Pearson Education, Inc., 2006.