ABSTRACT

THE EFFECTS OF LIMITED REBUILD BUFFER AND TRACK BUFFERS ON REBUILD TIME IN RAID5

by
Chintan Shah

Redundant Arrays of Independent Disks (RAID) are very popular for creating large, reliable storage systems. A RAID array consists of multiple independent disks that achieve fault tolerance by parity coding. The contents on a failed disk can be reconstructed on demand by reading and exclusive-ORing the corresponding blocks on surviving disks. Upon disk failure, the array enters rebuild mode when it begins to systematically reconstruct the data of the failed disk on a spare disk, provided one is available. The fundamental element of rebuild is the Rebuild Unit (RU).

Surviving disks engaged in rebuild, process user requests at a higher priority. Since, not all RUs are available at the same time, available RUs must be stored in a buffer, called the rebuild buffer, which is a part of the disk array controller cache. Most studies assume that this buffer is infinite. However, with the advent of large sized disks, it is increasingly difficult to provide buffers large enough that do not prove to be bottlenecks. This work studies the effect of a limited rebuild buffer on the rebuild time in an effort to estimate its effect on the Mean Time to Data Loss (MTTDL) of the array.

Finally, this work studies the idea of using "track buffers" which aim to improve the rebuild time by reducing the number of times a track has to be read in order to be completely rebuilt.

THE EFFECTS OF LIMITED REBUILD BUFFER AND TRACK BUFFERS ON REBUILD TIME IN RAID5

by
Chintan Shah

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science, Newark, NJ

May 2006

APPROVAL PAGE

THE EFFECTS OF LIMITED REBUILD BUFFER AND TRACK BUFFERS ON REBUILD TIME IN RAID5

Chintan Shah

Dr. Alexander Thomasian, Thesis Advisor Professor of Computer Science, NJIT	Date
Dr. David Nassimi, Committee Member Associate Professor of Computer Science, NJIT	Date
Dr. Cristian Borcea, Committee Member Assistant Professor of Computer Science, NJIT	Date

BIOGRAPHICAL SKETCH

Author: Chintan Sudhir Shah

Degree: Master of Science

Date: May 2006

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2006

- Bachelor of Engineering in Electronics and Telecommunication,
  University of Mumbai, Mumbai, India, 2004

Major: Computer Science

"There are only two ways to live your life. One is as though nothing is a miracle; the other is as though everything is." -- A. Einstein

To my beloved family, whom I owe everything.

ACKNOWLEDGMENT

I would like to thank Dr. Alexander Thomasian, my thesis advisor. He has been a wonderful mentor and friend who provided me with constant support, encouragement, resources and most importantly a vision to succeed in my research endeavors. Without his valuable guidance, completion of this thesis would not have been possible.

I would also like to express my gratitude to Dr. David Nassimi and Dr. Cristian Borcea for promptly agreeing to participate on my thesis defense committee.

Other fellow graduate and doctoral students also merit gratitude for their help and guidance. These include Jun Xu, Lijuan Zhang, Saman Zarandioon and Akheel Ahmed who helped me familiarize with the complicated simulator that was to be used to deduce the results. I would like to thank Gang Fu for helping me understand the simulator that he has written himself and allowing me to modify the code so as to get the desired results. I must also state that I am highly impressed by the complexity and the functionality of the DASim toolkit.

I would also like to acknowledge that work on this thesis was partially supported by NSF through Grant 0105485 in Computer Systems Architecture

Finally I would like to thank my parents and roommates for helping me through those frustrating moments when I was unable to obtain the correct results.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Objective

As user requirements for storage increase, newer models for storage systems have been designed, to provide higher reliability along with greater quantities of storage. RAID5 (Redundant Array of Independent Disks, level 5) has been a particularly successful model. This 1-DFT (Disk Fault Tolerant) model possesses the capacity to rebuild the data on a failed disk provided a hot spare is available.

The basic unit of rebuild is a Rebuild Unit (RU). As RUs are read from corresponding disks, they are Exclusive-ORed (XOR) with each other. Once RUs from all surviving disks have been XORed, the resulting data is written onto the spare disk as the reconstructed RU. However, since RUs from all disks may not be immediately and simultaneously available, they are stored in a "rebuild buffer," which is a part of the disk array controller cache

While quite a few studies have been done on the performance of rebuild in RAID5 systems, most of these assume an infinite rebuild buffer. These studies were viable since disks available in the market needed small buffer sizes (~128 MB) to be considered infinite. With the advent of faster and larger disks, however, the size of the buffer is increasingly proving to be a bottleneck. A detailed study of the effects of limited rebuild buffer size is presented in this work.

Secondly, most techniques used for the improving rebuild times involve a trade-off with the user response time. While these may speed up the rebuild, improve the

reliability of the array and ultimately allow for longer disk deployment periods, it also brings about considerable frustration for the user due to longer response times during rebuild.

This document also explains how the "track buffer," which is used for caching rebuild reads, reduces the rebuild time without negatively impacting the user response times. A detailed study reveals the possible potential of this technique, especially on larger capacity drives.

All performance analyses presented in this work have been done using the Disk Array Simulator (DASim) developed at the Integrated Systems Lab at NJIT. Disk characteristics, needed for these studies, have been obtained from the website of the Parallel Data Lab at Carnegie Mellon University.

1.2 Basic Information

This section provides the basic information required to understand the functioning of RAID arrays in general and RAID5 in particular.

Amdahl's law illustrates that the performance of computer systems is limited by the performance of the I/O subsystem [5]. It is further evident from the rate at which CPU performance has been increasing as compared to storage performance. To utilize the improved the CPU performance, there is a need for parallelism in the I/O subsystem. A RAID array provides not just large capacity and greater reliability, but also provides a fair degree of parallelism.

1.2.1 RAID Array Structure

A RAID array consists of multiple independent disks arranged in an array. These disks are connected via busses to an Array Controller which is an abstraction layer for the RAID system. The host computer treats the entire array as a very large disk.

Figure 1.1 shows the architecture of a typical RAID array. Each disk is independent is connected to the array. These disks are "pluggable," which means that they can be removed and attached to their position. This facilitates replacement of failed disks. The Array Controller is connected to the host computer through high bandwidth busses. [7]

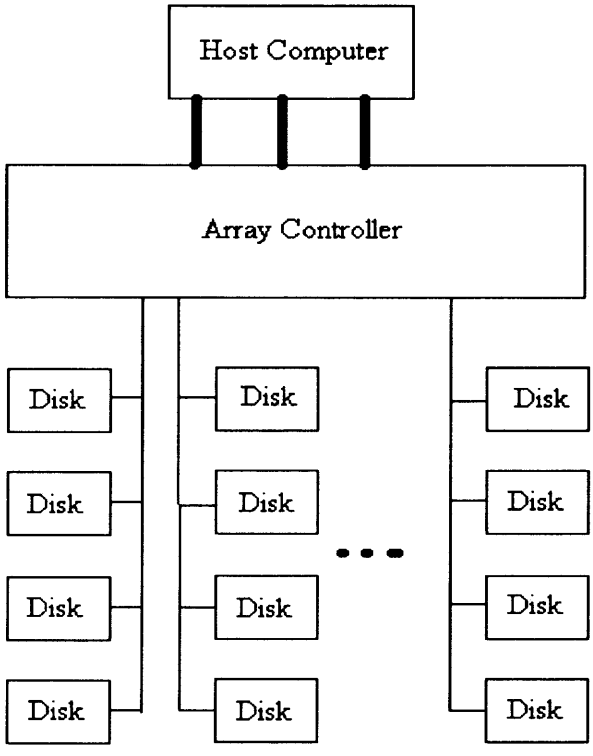

Figure 1.1 RAID Array Architecture.

The array controller performs system-related operations such as controlling individual disks; maintain address mapping and redundant information, and recovering from disk failures by rebuilding.

1.2.2 Stripe Units

The array controller also breaks down the contiguous data into units of constant size called "Stripe Units." [13, 14]. These stripe units are assigned to consecutive disks. This provides load balancing in case of concurrent workloads and increased bandwidth for the sequential transfer of large amounts of data by a single process.

The Figure 1.2 depicts a RAID level 0 non-redundant disk array. In this organization, data is striped, but has no duplication. It results in improved throughput. However, since the reliability of each disk must be multiplied with that of the other, it results in very poor reliability.

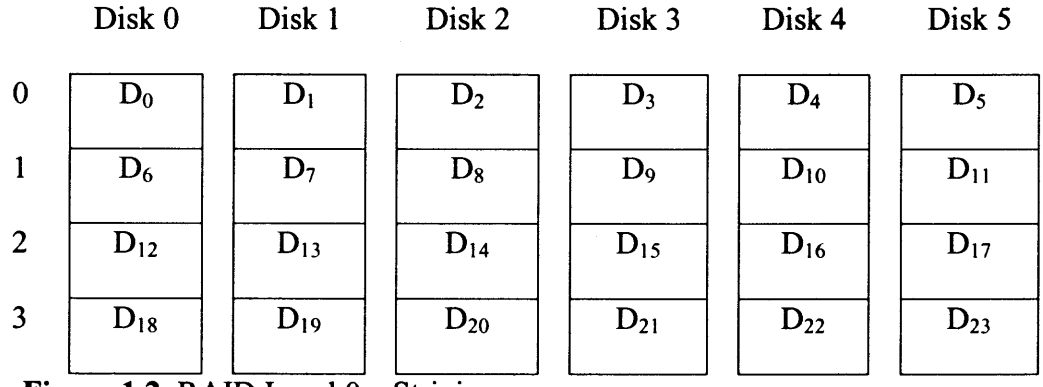

Figure 1.2 RAID Level 0 – Striping.

The obvious advantage of using Striping is that it allows for parallelism. Contiguous data can now be accessed through different disks rather than from the same

disk. This can speed up data access, specifically for requests of large amounts of contiguous data.

Description of all RAID levels is given in [13].

1.2.3 Rebuild Units

A rebuild unit (RU) is the basic unit of rebuild. It may be the same size or a fraction of a stripe unit (SU). RUs used during simulations in this study are typically of 256KB each. They are usually constant, but may be different for different zones in zoned disks.

1.3 RAID Level 5

1.3.1 RAID5 Organization

RAID5 uses rotated block-interleaved parity. This means that all parity blocks are not located on a single disk like RAID4. These are distributed across all the disks. This avoids making the parity disk a bottleneck during rebuild.

The RAID5 design considered here has a "left symmetric organization." [9]. In a left symmetric organization, the parity blocks are placed across the diagonal and the consecutive data stripe units are placed on the consecutive disks at the lowest available offsets. Parity is computed over a group of disks, called a "parity group." This organization is illustrated in Figure 1.3

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
0	D_0	D_1	D_2	D_3	D_4	P_{1-4}
1	D_6	D_7	D_8	D_9	P_{5-9}	D_5
2	D_{12}	D_{13}	D_{14}	P_{10-14}	D_{10}	D_{11}
3	D_{18}	D_{19}	P_{15-19}	D_{15}	D_{16}	D_{17}
4	D_{24}	P_{20-24}	D_{20}	D_{21}	D_{22}	D_{23}
5	P_{25-29}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}

Figure 1.3 RAID Level 5 (Rotated Block Parity).

1.3.2 RAID5 Reliability

One of the main purposes of employing disk arrays is to increase reliability. The reliability of single-disk fault tolerant arrays can be measured as the "mean time to data loss" (MTTDL) which is given in [13]:

$$MTTDL = \frac{(MTTF)^2}{N\,(N-1)\,MTTR_{disk}}$$

Where:

MTTDL = Mean Time To Data Loss

MTTF = Mean Time To Failure of a component disk

N = Number of disks in the array

$MTTR_{disk}$ = Mean Time To Repair a disk

The MTTF for a disk is typically one million hours and the $MTTR_{disk}$ is a few hours. It is the same as the rebuild time.

Based on the above equation, any improvement in the rebuild time will thus bring about the improvement in the MTTDL and ultimately allow the increase in the disk array deployment period which is usually five years.

CHAPTER 2

REBUILD PROCESSING

2.1 Introduction to Rebuild Processing

Rebuild processing is the systematic reconstruction of data on the failed disk, on a hot spare. It is begun as soon as a hot spare is available and continues till all the data on the failed disk is rematerialized onto the spare. The smallest unit that is rebuilt as a whole is called the "Rebuild Unit" (RU).

The time taken to rebuild the failed disk $T_{rebuild}(\rho)$, and the response time for user requests $T_{response}(\rho)$ where ρ is the disk utilization in the normal mode are the two main parameters to measure system performance. After a disk failure, the RAID5 array operates at a higher disk utilization $\rho' = 2\rho$ when all requests are read requests.

There are two kinds of rebuild – Stripe oriented and Disk oriented. In a stripe oriented rebuild, a new process is spawned for each RU to be rebuilt. It reads the RUs from the surviving disks, XORs them and writes the resulting RU on the spare disk. Disk oriented rebuild on the other hand dedicates one process for each disk that reads RUs from the surviving disks asynchronously and puts them into a buffer which successively XORs the RUs as they become available. Disk oriented rebuild has been shown to outperform stripe oriented rebuild in [6]. All results presented in this study, thus use disk oriented rebuild only.

Rebuild Requests can be processed in one of two ways: The Permanent Customer Model (PCM) processes rebuild requests at the same priority level as that of user

requests. However, only one rebuild request is inserted in the queue at a time – a new rebuild request is inserted at the end of the queue only when the previous one is served.

The Vacationing Server Model (VSM) on the other hand processes user requests at a higher priority level than rebuild requests. Thus, rebuild requests are served by the array only when no user requests exist in the queue i.e when the array is vacationing. However, no preemption is allowed, meaning that once a rebuild request has been accepted by the system, a user request will be served only after its completion.

Performance comparisons of VSM and PCM models have been done in [2]. It is shown that VSM gives lower rebuild and response times. Hence, this rebuild strategy has been used in this study.

Other strategies such as piggybacking [3] and rebuild request pre-emption [14] exist. However, these are not considered in this study.

2.2 Rebuild in RAID5

As soon as a disk failure occurs in the RAID5 array, the system enters the degraded mode of operation. If a hot spare is available, the array then immediately enters into a rebuild mode in which the data on the failed disk starts materializing on the spare disk.

Consider that disk 4 as shown in Figure 1.3 fails. Data on this disk can be rebuilt as follows:

$D_4 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus P_{1-4}$

$P_{5-9} = D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9$

And so on.

Thus a read request to any part of a failed disk is served by initiating a fork-join request to all the surviving disks during degraded and rebuild mode. This causes the disk utilization in rebuild mode or degraded mode to double as compared to the normal mode utilization (U).

In the rebuild mode, the array must serve two types of requests:

1. User Requests: Requests for data by the user application are called as user requests
2. Rebuild Requests: For rebuilding, every RU on the failed disk is rebuilt by reading the corresponding RUs on all the surviving disks and XORing them.

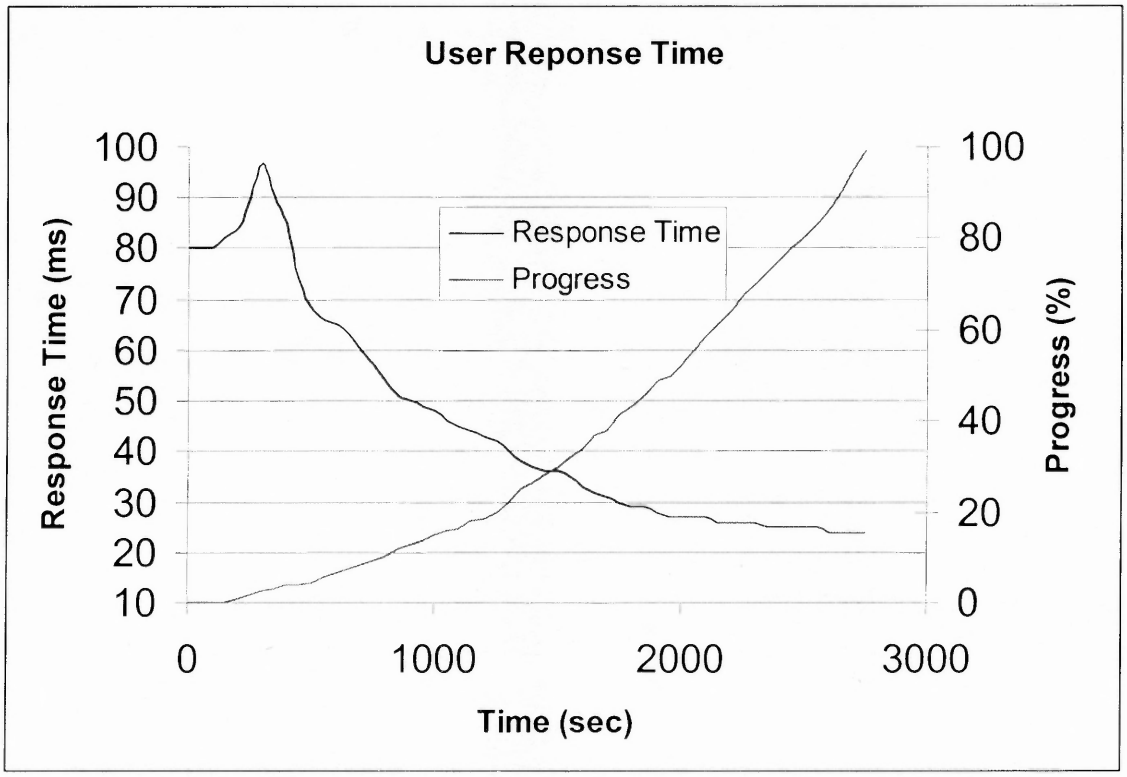

Figure 2.1 Change in User Response Time following a disk failure (N = 21, B = 256 MB, RU = 256 KB).

User response time increases significantly upon a disk failure and stays above normal for the period of the rebuild. However, it continues to reduce as more and more

data is rebuilt on the spare disk and can be served directly from there instead of by a fork-join request. Figure 2.1 shows the trend in user response time after a failure for a RAID5 for 19 disks using FCFS scheduling and VSM strategy for disk rebuild.

2.3 Disk Scheduling Policies

Scheduling refers to the policy by which the disk decides which of the requests on the queue it must serve next. While this may not be directly related to rebuilding, the order in which requests are processed still affects the rebuild time.

Two scheduling policies are used frequently – First Come First Served (FCFS) and Shortest Access Time First (SATF). FCFS policy involves serving the requests in the order that they arrived. SATF policy involves serving requests in the order of their access times.

Figure 2.2 shows the rebuild times for different arrival rates for a RAID5 array of 21 disks using FCFS and SATF scheduling with a RU size of 256 KB. The rebuild time increases progressively with higher arrival rates and the rebuild time for FCFS scheduling is higher than that for SATF scheduling at a given arrival rate.

It must also be noted that for a given arrival rate, the disk utilization for SATF is either equal to, or less than, that for FCFS scheduling. This can be seen from Figure 2.3

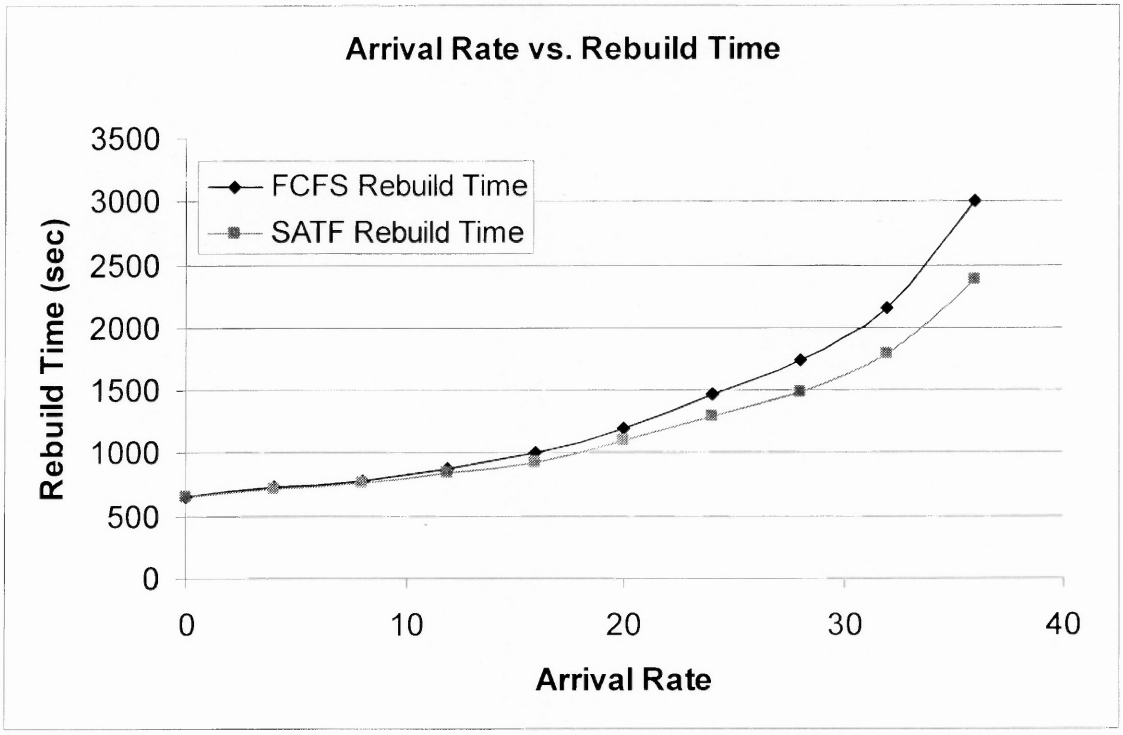

Figure 2.2 Rebuild Time vs. Normal Mode Arrival Rate (N = 21).

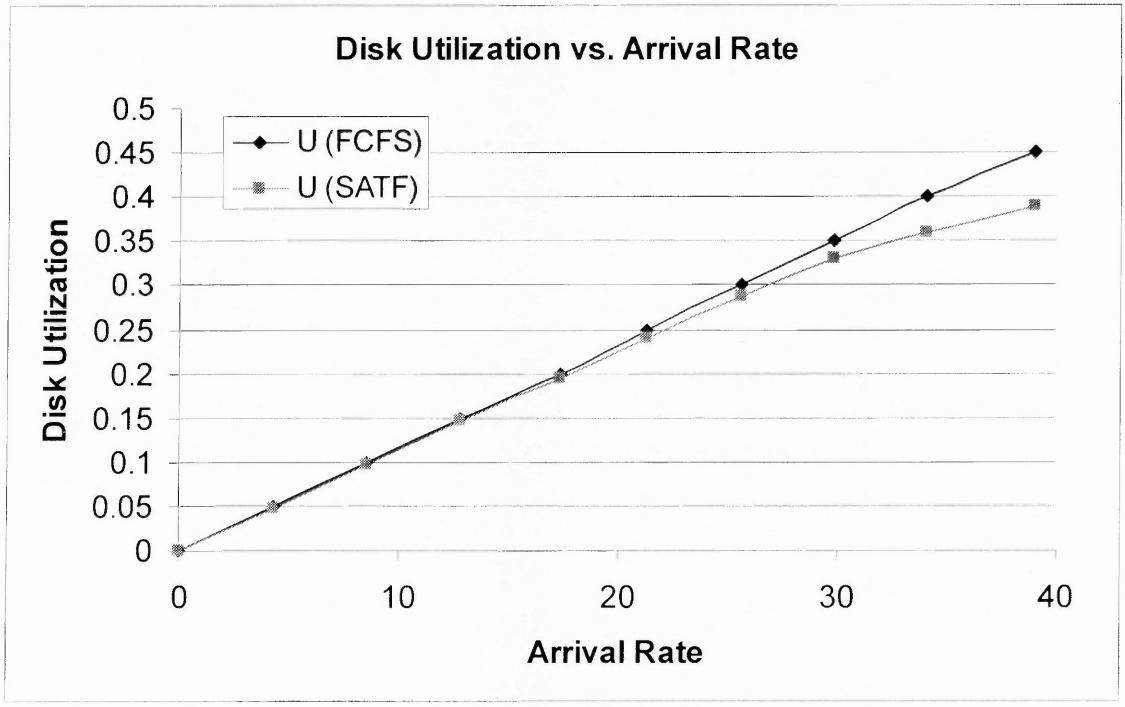

Figure 2.3 Disk utilization for a RAID5 system at different arrival rates with FCFS and SATF (N = 21).

This difference in the disk utilization is due to the fact that SATF policy is geared toward reducing access times and thus is more efficient at processing requests in a given amount of time.

The major drawback of SATF policy is that it can lead to "starvation." Since the policy dictates that the request that requires the minimum access time must be processed next, without any consideration of when the request arrived, a request that is waiting in queue but has a large access time may never be served. Most disk drives that use SATF scheduling today use some mechanism of "ageing" to address this problem. These are given in [17].

2.4 Factors Affecting Rebuild Time

Rebuild time is an important measure of the performance of disk arrays. This section details different factors that directly affect rebuild time.

The factors that affect rebuild time are:

1. Rebuild Unit Size (RU size)
2. Rebuild policy (VSM or PCM)
3. Number of disks (N)
4. Disk Utilization (U)
5. Rebuild Buffer Size (B)

2.4.1 Effect of Rebuild Unit Size on Rebuild Time

The RU size dictates the smallest unit that must be rebuilt before the next user request must be served. Consequently a small RU size would mean long rebuild times and short user response times while a large RU size would mean short rebuild times and long user response times. Figure 2.4 shows how RU size affects the rebuild time at different utilizations.

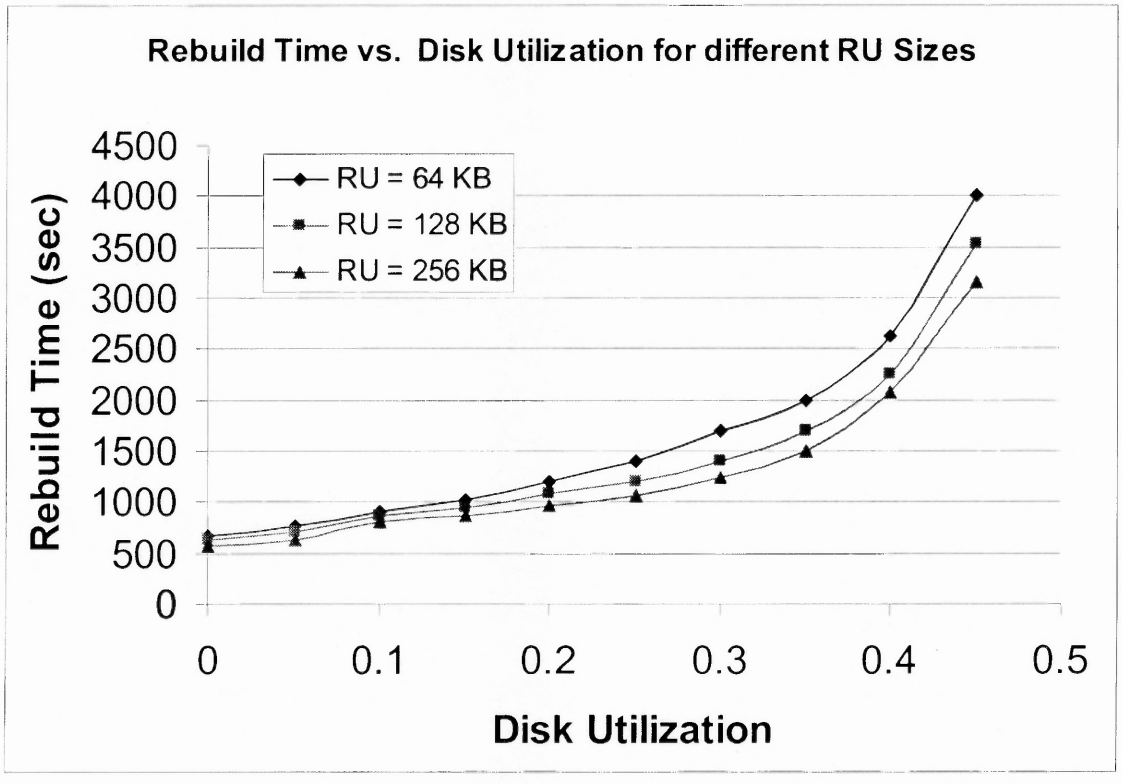

Figure 2.4 Rebuild Time vs. Disk Utilization for different RU sizes (N = 21, B = 256 MB).

The graph in Figure 2.4 shows that for a given utilization, rebuild time is lower for larger RU sizes. Change in RU sizes has a greater impact for higher utilizations since more user requests have to be served.

Detailed work has been done in [1].

2.4.2 Effect of Rebuild Policy on Rebuild Time

Section 2.1 contains a brief description of two rebuild policies – VSM and PCM that have been studied extensively in [3, 18, 19]. It has been determined that VSM is a superior policy to PCM since it has better rebuild and response times.

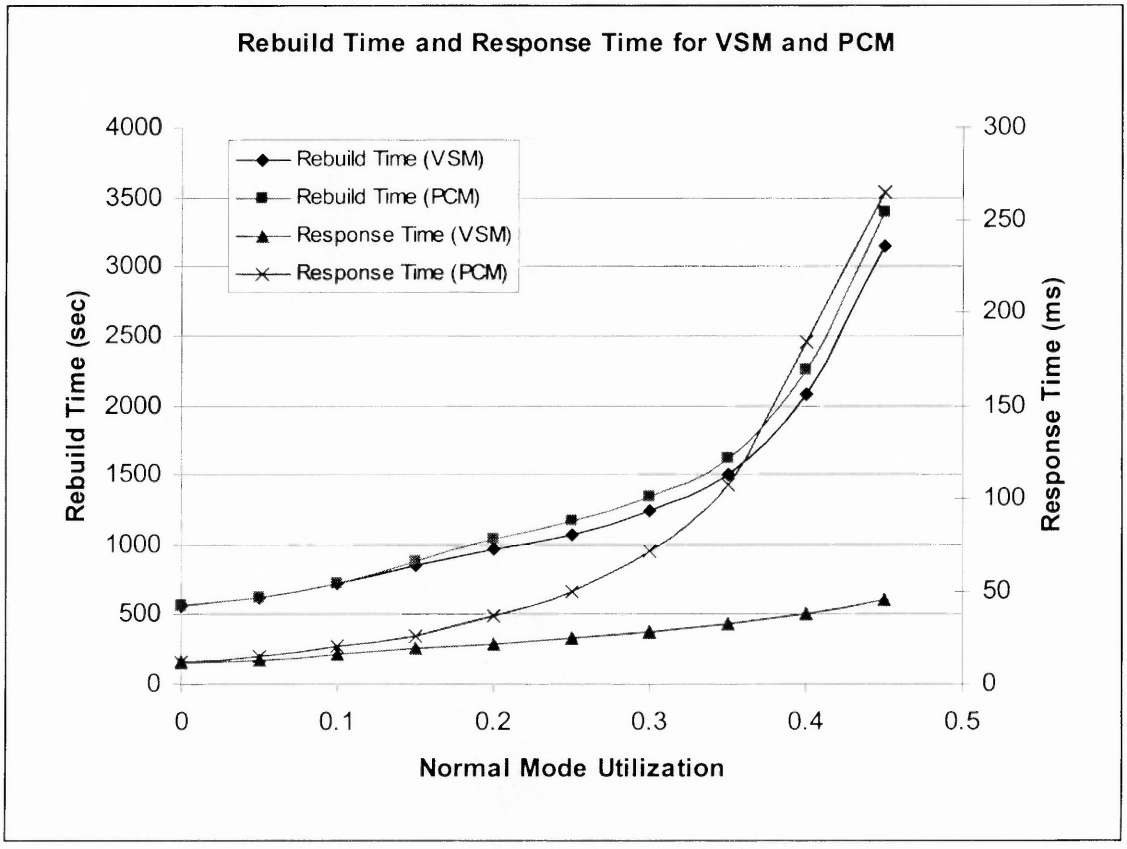

Figure 2.5 Rebuild and Response Times for VSM and PCM rebuild policies [2].

Figure 2.5 shows the rebuild time and response time for VSM and PCM for a RAID5 array of 21 disks using FCFS scheduling for a RU size of 256 KB. Detailed analytical study is given in [15, 16].

2.4.3 Effect of Number of Disks on Rebuild Time

It can be empirically proven, through multiple simulations that the number of disks does not significantly affect the rebuild time. Figure 2.6 shows that rebuild time does not change too significantly at utilization of 0.45 at buffer sizes of 32 MB, 64 MB and 128 MB (which may be considered infinite for the purpose of the disk used for simulation.)

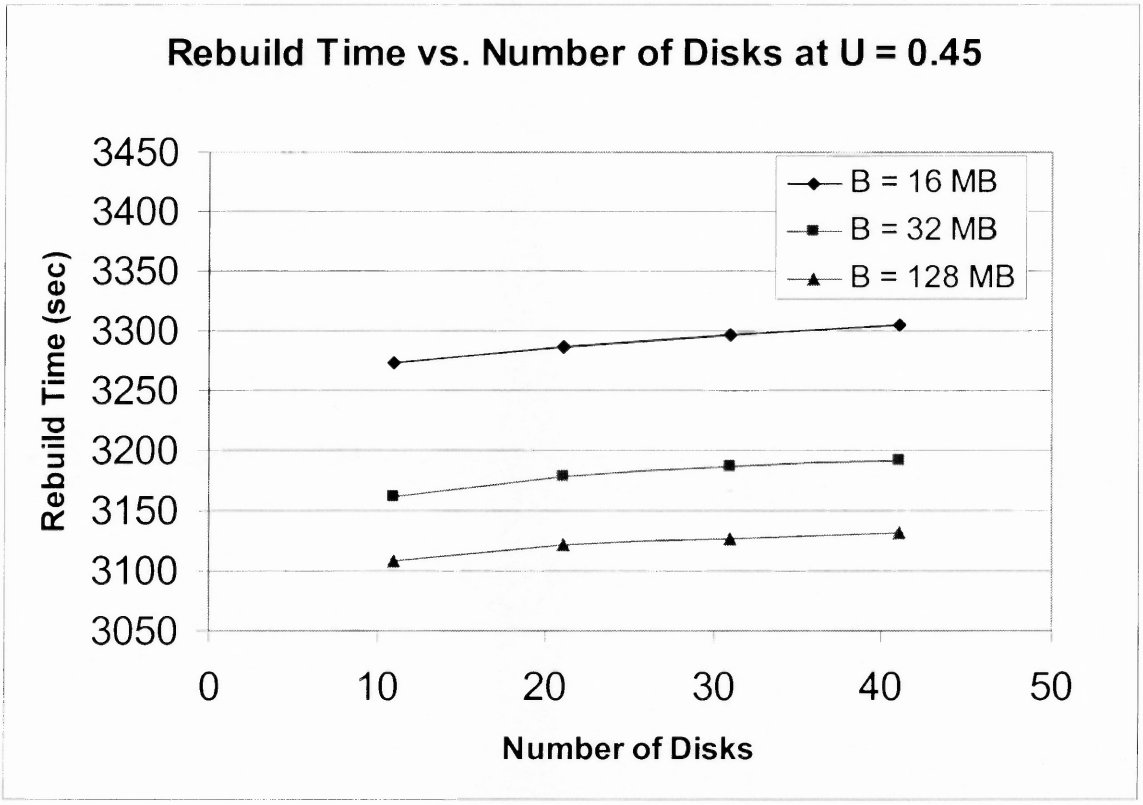

Figure 2.6 Rebuild time vs. Number of disks (B = 256 MB, RU = 256 KB, U = 0.45).

As seen from the above graph, while the buffer size impacts the rebuild time fairly significantly, the lines on the graph above are almost flat. The maximum amount of difference occurs from small buffer sizes. In the case of a 16 MB buffer, the rebuild time varies from 3,264 sec for N = 11 to 3,303 sec for N = 41. This represents an increase of only 0.7%. This change of 39 seconds can be considered negligible, given that the entire

rebuild operation takes more than 3,200 sec. It is also worthwhile to note that for disk arrays larger than 41 disks, RAID5 would be bad option and RAID6, which is a 2 Disk Fault Tolerant array, would be preferred. Thus, studying this trend for values of N > 41 would only be academic and would have no practical use.

2.4.4 Effect of Disk Utilization on Rebuild Time

The effect of disk utilization has been extensively studied in [2]. It proposes an empirical formula suggesting that for a RAID5 array with an infinite rebuild buffer, $T_{reb}(u) = T_{reb}(0) / (1 - \alpha u)$, where α was experimentally calculated to be 1.75. This result was obtained as a result of curve-fitting.

Figure 2.7 shows the effect of disk utilization on rebuild time for a RAID5 array of 21 disks with a RU size of 256 KB and an infinite rebuild buffer.

2.4.5 Effect of Rebuild Buffer Size on Rebuild Time

This is studied in detail in the next chapter.

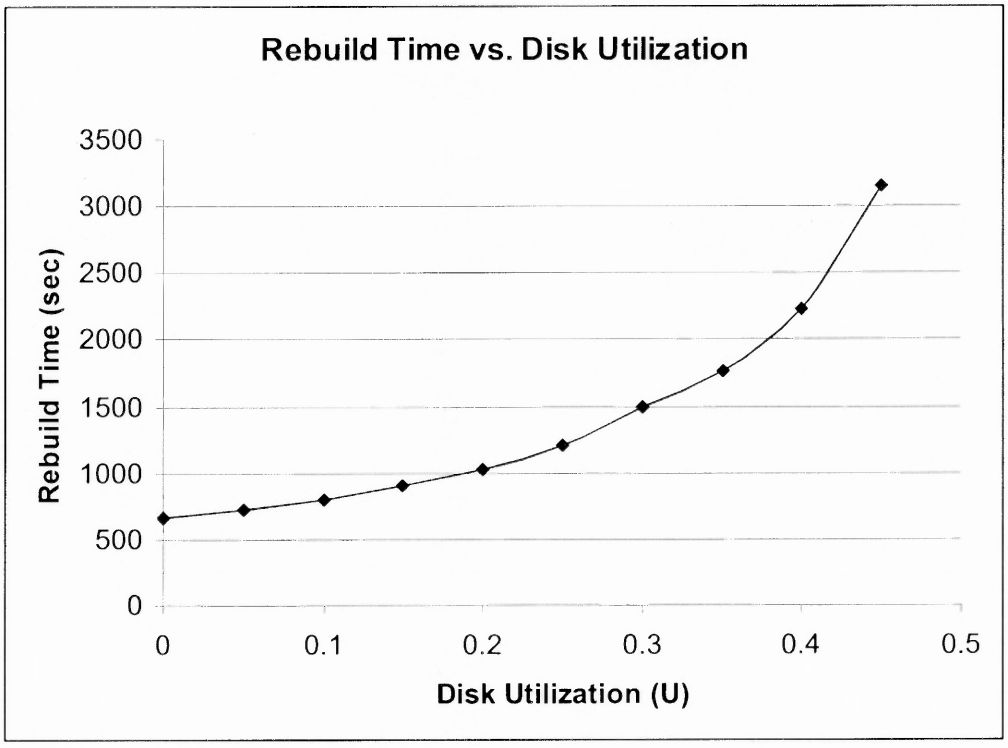

Figure 2.7 Rebuild Time vs. Disk Utilization (N = 21, RU = 256KB, B = 256 MB).

CHAPTER 3

IMPACT OF LIMITED REBUILD BUFFER

3.1 Limited Buffer

Rebuild involves the reading of data from the surviving disks, XORing the data and recreating the data that must be materialized on the spare disk. However, since not all disks rotate at the same speed and small amounts of inconsistencies exist, the data read from each disk is first stored on a buffer. Once XORed, the rebuilt data is written onto the spare disk.

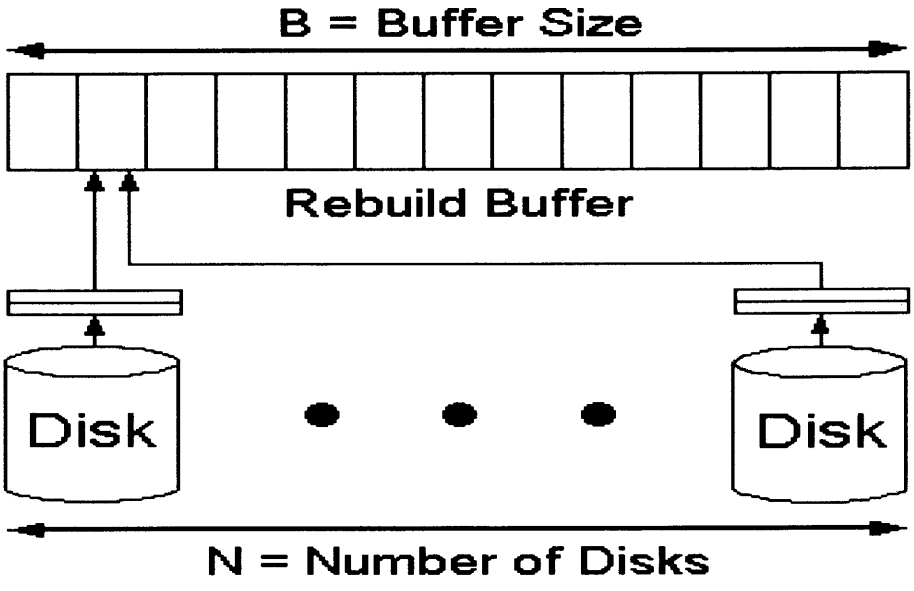

Figure 3.1 Rebuild Buffer.

Figure 3.1 shows the schematic of the rebuild buffer. As RUs from each disk become available they are successively XORed with the content of the mapped buffer slot. Once all RUs have been XORed, this data is written onto the spare disk as the rematerialized rebuild unit.

So far, most studies assume an infinite buffer thus ensuring that this does not prove to be a bottleneck. In this chapter, the impact of a limited buffer is presented. Previous work on this topic has been done in [2, 3]. The readings in Table 3.1 and the Figure 3.2 emphasize the impact.

These studies have been done for a RAID5 system consisting of N = 21 disks at a normal mode utilization of 0.45 (Rebuild Mode Utilization of 0.9). Unless otherwise mentioned, the RU size is 256 KB (i.e 64 blocks of size 4KB each) for all studies in this chapter.

Table 3.1 Impact of Buffer Size on Rebuild and Response Time

Buffer Size (MB)	Rebuild Time (sec)	Response Time (ms)
16	3287	44.23
32	3179	44.75
64	3157	44.77
128	3155	44.79
256	3154	44.79
512	3154	44.75

As seen from the table and the chart, it is clear that while rebuild times are significantly affected by limited buffer size, the response time is almost same. For further observations in this chapter, the effect on response time is neglected. The only worthwhile observation concerning response time is that as the buffer size decreases, user response time decreases too as disks have to stop processing rebuild requests while buffer utilization is lowered.

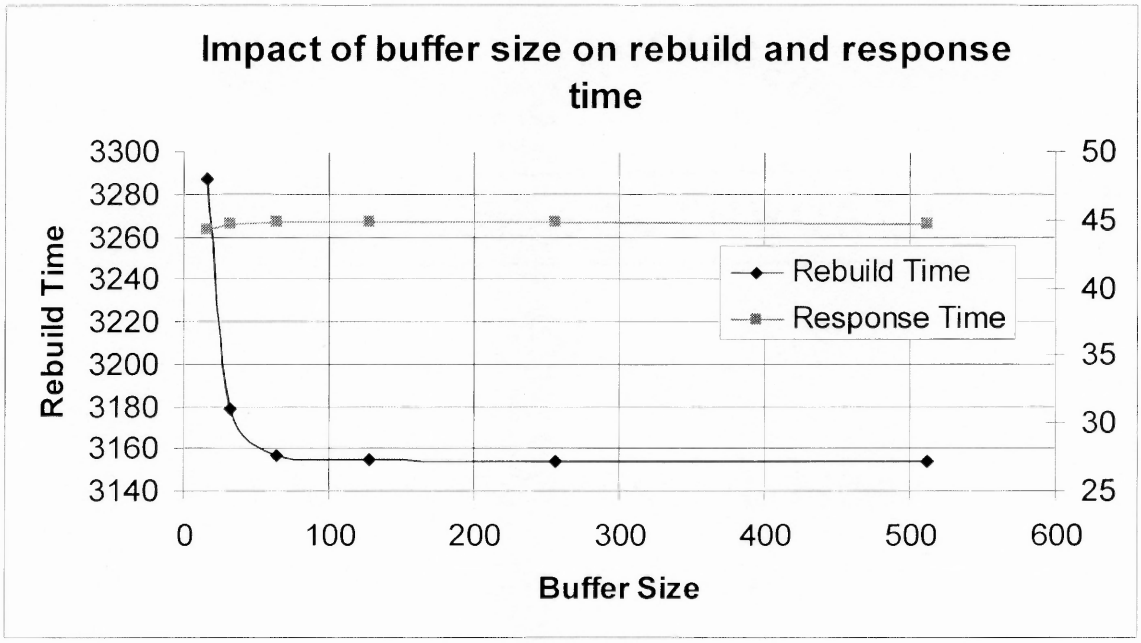

Figure 3.2 Impact of buffer size of rebuild and response time (N = 21, U 0.45, RU = 256KB).

From the above results it can be observed that there is a great deal of sensitivity when the buffer size is lower than 128 MB. This region is closely examined in Figure 3.3.

3.2 Analysis of Rebuild Time Trends

From the Table 3.2 and Figure 3.4, the following trends emerge: The change in rebuild time is significant for buffer sizes below 128 MB. As the size of the buffer is increased above 128 MB, no significant improvement is seen. This leads to the inference that for practical purposes, a 128 MB buffer may be considered infinite for a disk with 7,200 RPM.

As the buffer size is reduced below 64 MB, the rebuild time increases significantly. In Tables 3.2 and 3.3 and corresponding Figures 3.4 and 3.5, the impact of buffer sizes on RAID5 systems with 11 and 21 disks at normal mode utilizations of 0.3 and 0.45 is studied.

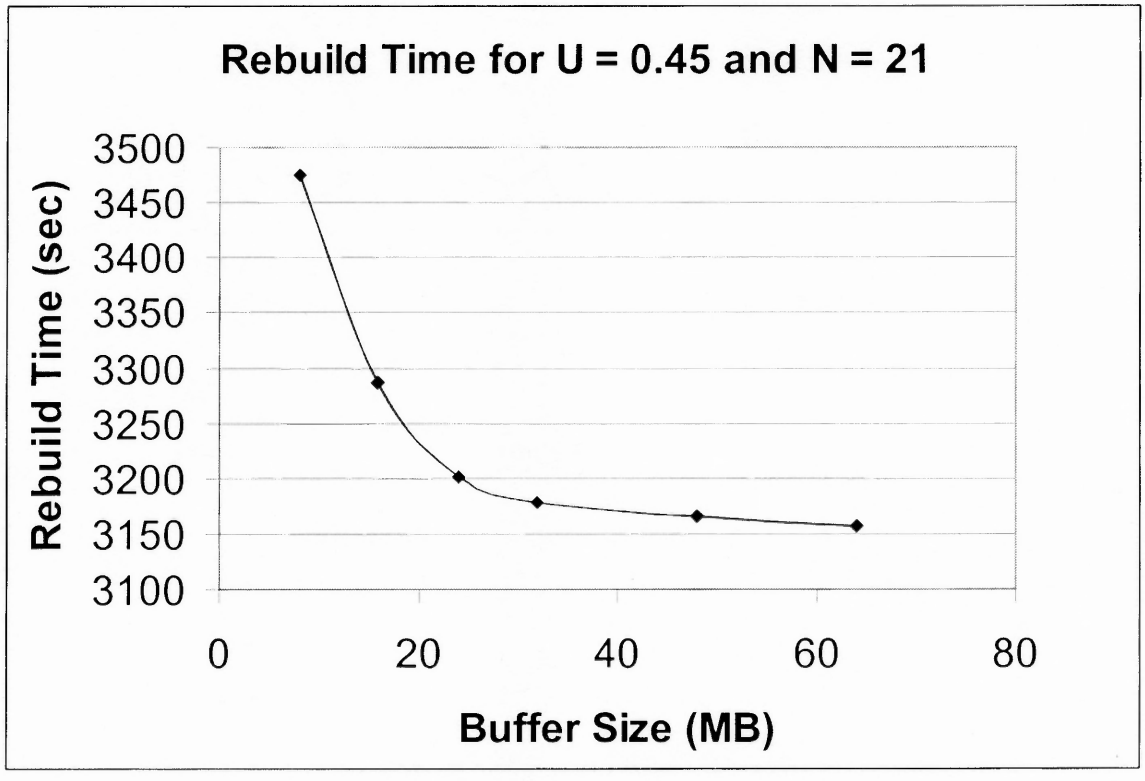

Figure 3.3 Rebuild Time for Buffer size < 128 MB (N = 21, U = 0.45, RU = 256 KB).

Table 3.2 Impact of Buffer Size on Rebuild Time for RAID5 at U=0.3

Buffer Size (MB)	T_{reb} for 11 disks (sec)	T_{reb} for 21 disks (sec)
8	1759	1776
16	1688	1706
24	1622	1642
32	1581	1601
48	1531	1545
64	1516	1523
128	1494	1499

Table 3.3 Impact of Buffer Size on Rebuild Time for RAID5 at U=0.45

Buffer Size (MB)	T_{reb} for 11 disks (sec)	T_{reb} for 21 disks (sec)
8	3468	3476
16	3264	3287
24	3186	3201
32	3159	3179
48	3128	3151
64	3113	3131
128	3109	3128

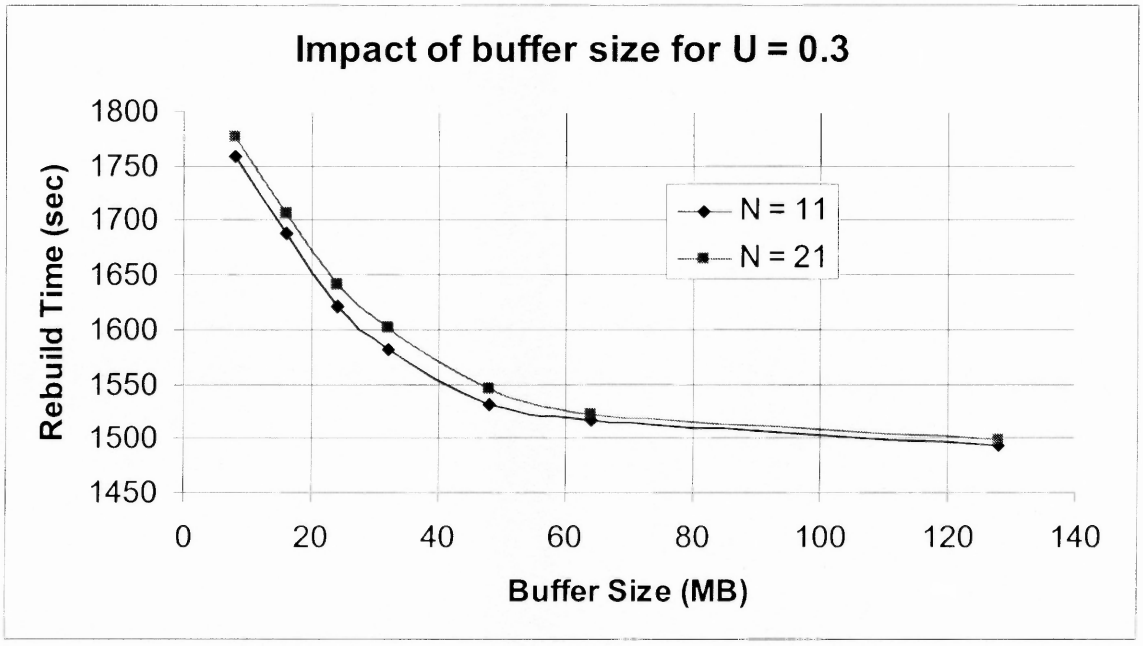

Figure 3.4 Impact of buffer size on rebuild time at U = 0.3 (RU = 256 KB).

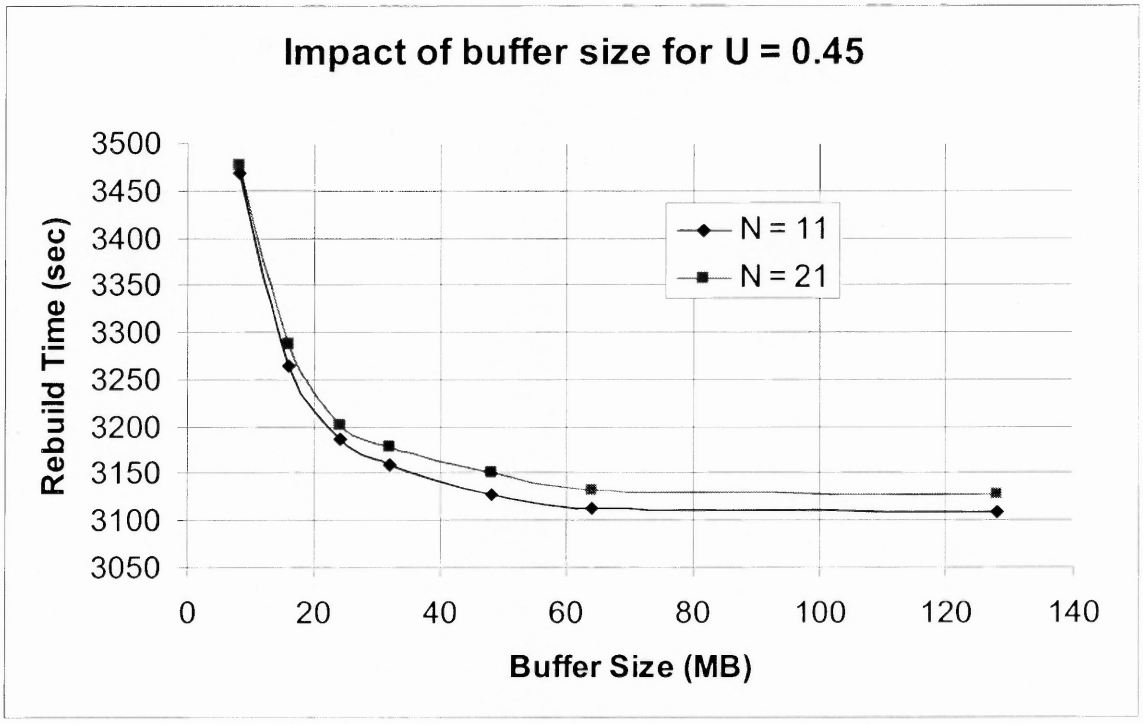

Figure 3.5 Impact of buffer size on rebuild time at U=0.45 (RU = 256KB).

Another observation that can be readily made from the above graphs is that as the utilization increases, the effect of limited buffer size becomes progressively less pronounced.

The next study presents the effect of disk utilization with respect to the buffer utilization. Buffer utilization is the average percentage of buffer slots that are occupied by valid data over the time period when rebuilding is being done. This is measured as the percentage of the total buffer being utilized.

An experimental study of buffer utilization vs. buffer size reveals that buffer utilization drops progressively as the buffer size is increased. It can also be seen that the buffer utilization varies widely with disk utilization. It can also be observed that for a buffer size smaller than 128 MB, the buffer utilization is greater than 60% for

intermediate disk utilizations. This suggests that the disks must stop many times since buffer utilization goes above 100% periodically. This trend is shown in Figure 3.6

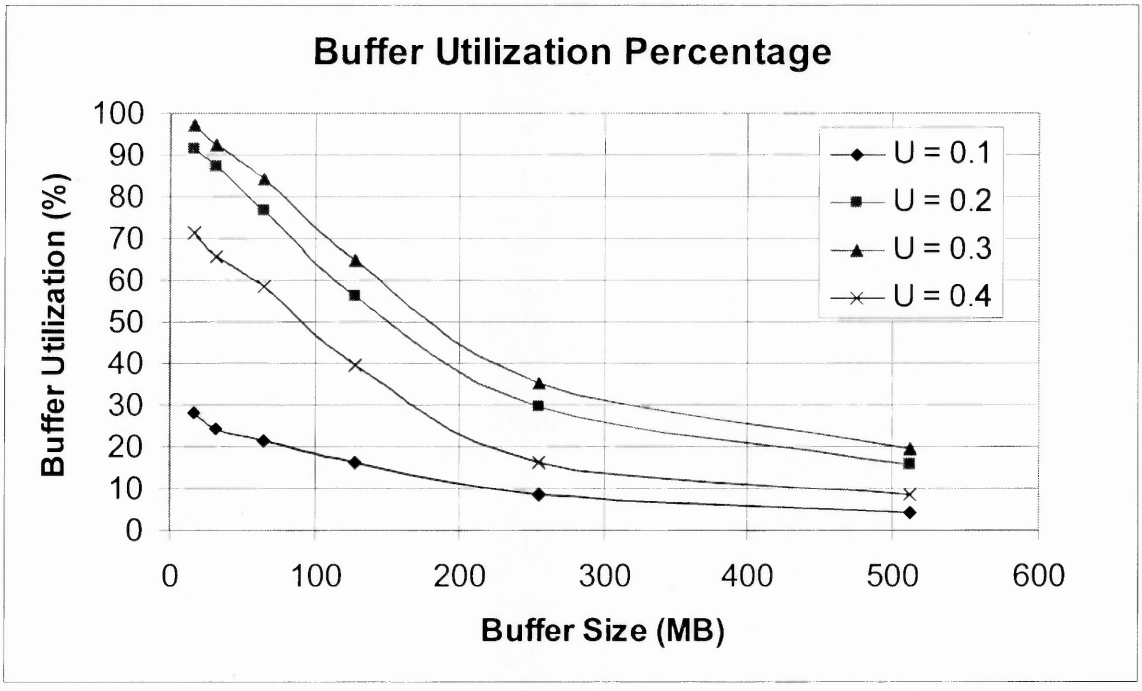

Figure 3.6 Buffer utilization for different buffer sizes (RU = 256 KB, N = 21).

As mentioned above, the buffer utilization varies widely with disk utilization. Theory dictates that for low normal mode disk utilizations, i.e less than 0.15, the buffer utilization would be very low since all disks can work in synchronization without being interrupted by user requests too often. For example, if no user requests arrive, the buffer utilization should be almost zero since the maximum discrepancy would be 1 track.

In a similar manner, for very high disk utilizations, the buffer would also be underutilized, since most of the time would be spent by disks serving user requests. However, buffer utilization in this case would still be higher than buffer utilization at low disk utilization since the disks would tend to read the RU's only when it gets a little time

from serving user requests. This would also lead to the disks idling for a longer time, though the number of stops would be small. The number of stops per disk and time per stop are covered in the *Disk Stopping* section of this chapter.

The above inferences that can be drawn from theory are confirmed in Figure 3.7. It shows the graph for buffer utilization vs. disk utilization at buffer sizes of 32 MB and 64MB. Close inspection of these graphs suggests a peak value of buffer utilization at normal mode disk utilization of approximately 0.25. This graph starts flattening as the buffer size is increased above 64 MB.

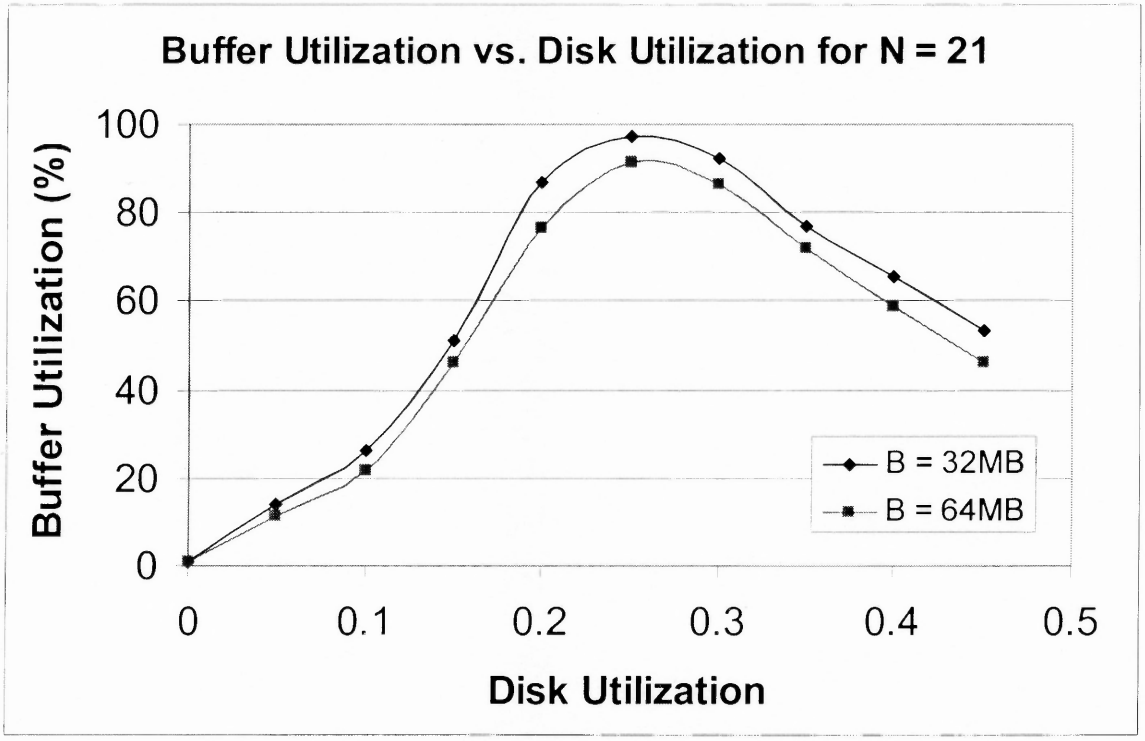

Figure 3.7 Buffer Utilization vs. Disk Utilization (N = 21, RU = 256 KB).

3.3 Disk Stopping

During the rebuilding process, as the buffer utilization reaches 100%, the disks serving the rebuild requests (i.e., the surviving disks) must stop and wait for the buffer utilization to be lowered. This can be done only when the data is taken out of the buffer and written to the spare disk.

Since the VSM model is used for rebuilding, a rebuild request that is currently being served must be halted while the RU is being written from the buffer onto the spare disk. This halted rebuild request cannot even be pre-empted as dictated by the VSM model [3, 14]. A study of the average number of stops per disk is presented below:

Figures 3.8 and 3.9 as well as tables 3.4 and 3.5 presents the average number of stops and the time per stop in milliseconds, for an array of 11 disks at different normal mode utilizations for a 32 MB and a 64 MB buffer.

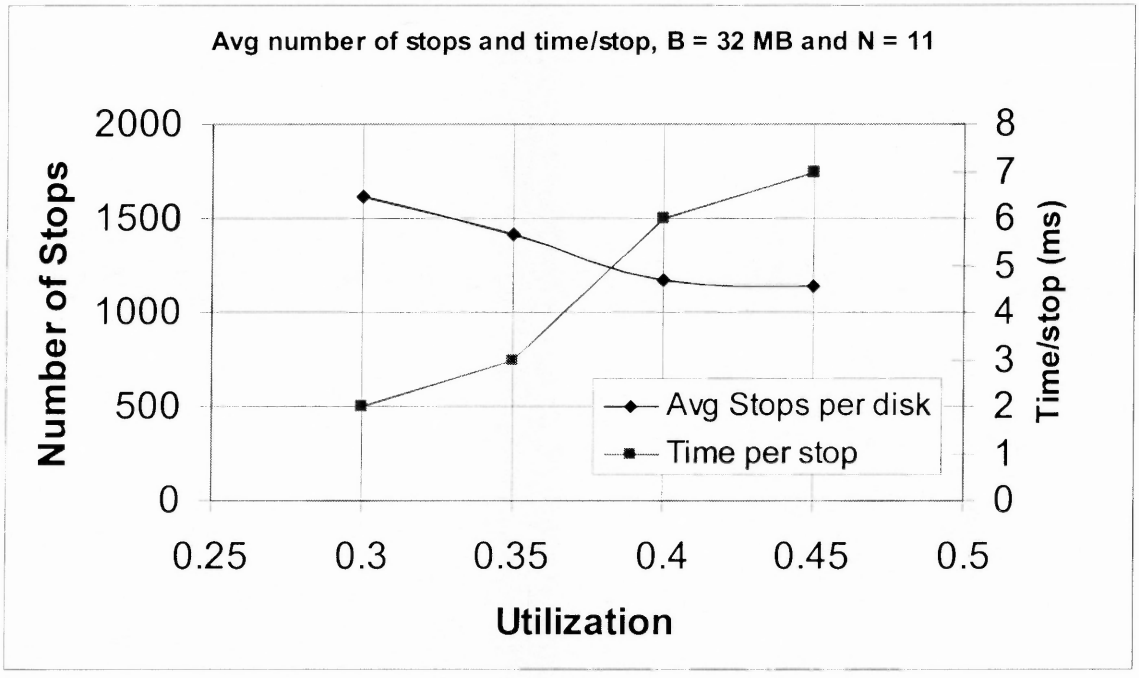

Figure 3.8: Average number of stops and time per stop for Buffer size = 32MB (N = 11, RU = 256 KB).

Table 3.4 Average Number of Stops per Disk and Time for Buffer Size = 32 MB

Utilization	Avg Stops per disk	Time per stop (ms)
0.30	1612	2
0.35	1411	3
0.40	1167	6
0.45	1143	7

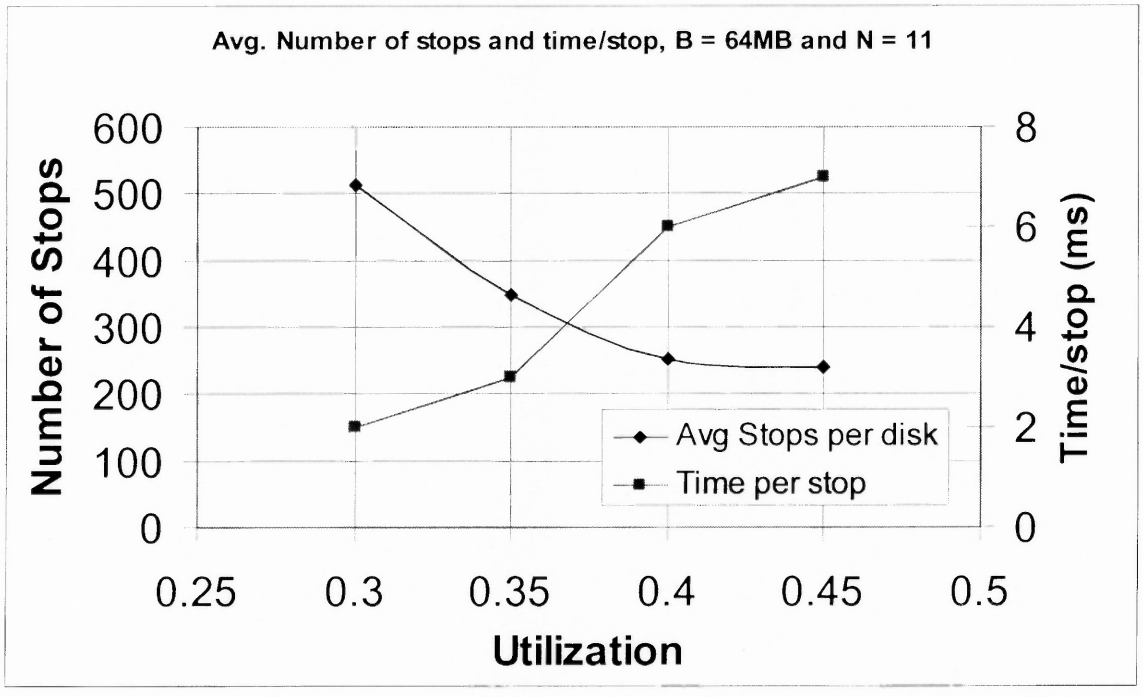

Figure 3.9 Average number of stops and time per stop for Buffer size = 64MB (N = 11, RU = 256 KB).

Table 3.5 Average Number of Stops per Disk and Time for Buffer Size = 64 MB

Utilization	Avg Stops per disk	Time per stop (ms)
0.3	513	2
0.35	347	3
0.4	253	6
0.45	241	7

From the above simulation results, the following trends emerge: As utilization increases, for a given buffer size and a given number of disks in the array, the number of stops decrease with increasing time/stop. Another critical observation is that the time per stop for a given utilization is almost the same for different buffer sizes. These trends are further confirmed in corresponding results taken with a RAID5 array of 21 disks.

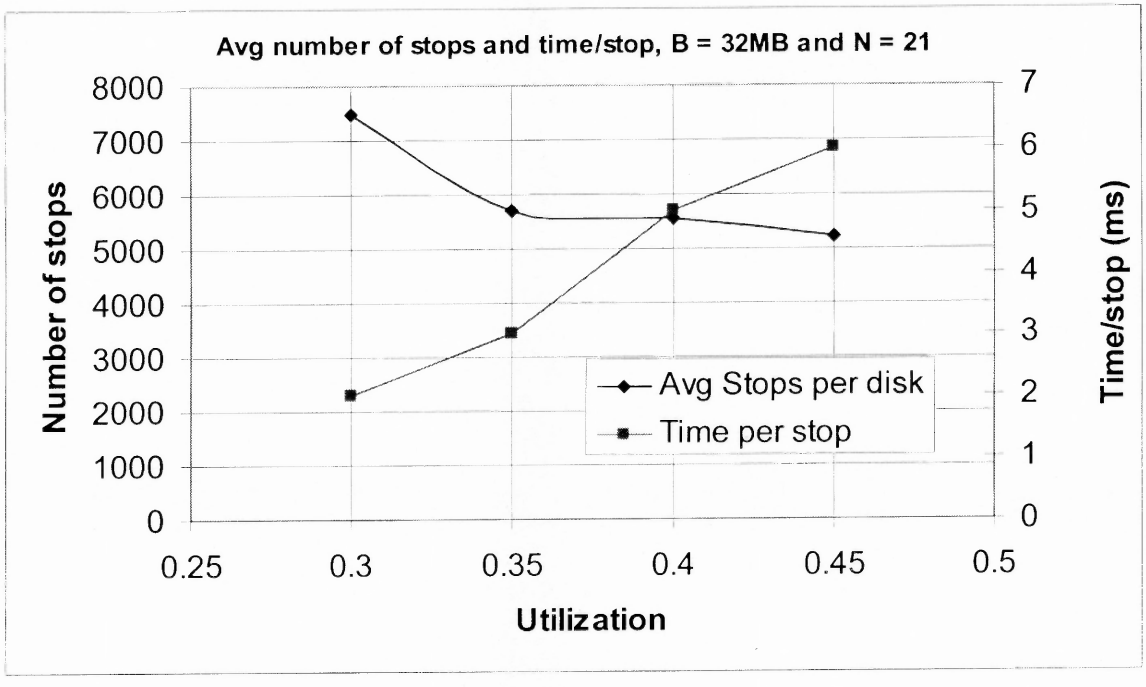

Figure 3.10 Average number of stops and Time per stop for Buffer size = 32 MB (N = 21, RU = 256KB).

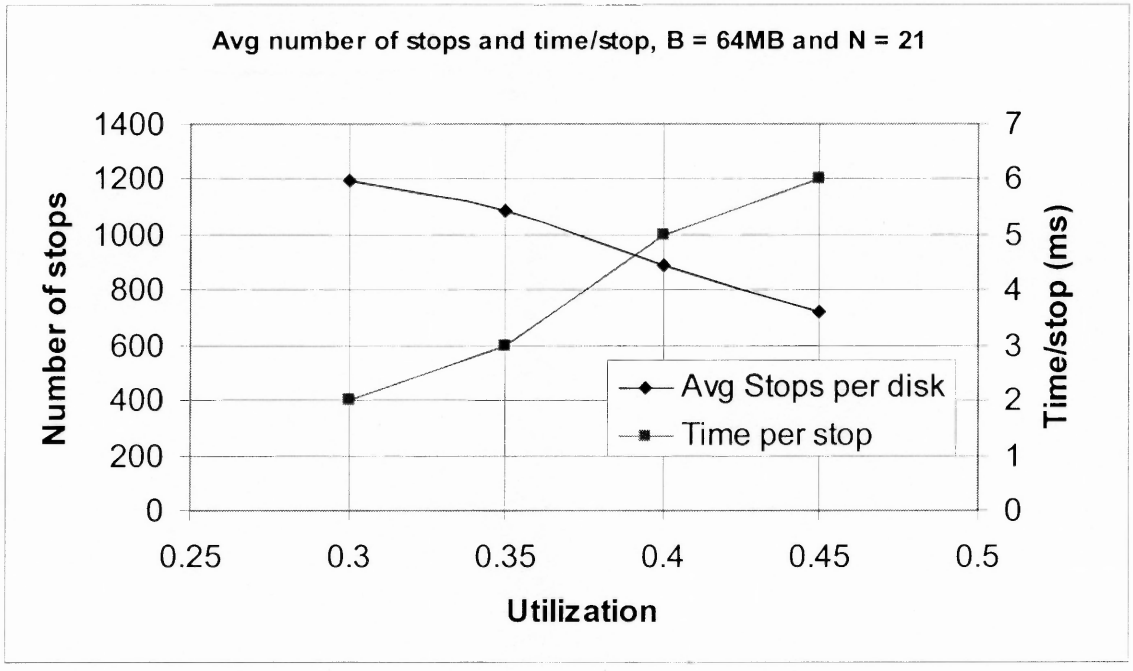

Figure 3.11 Average number of stops and Time per stop for Buffer size = 64 MB (N = 21, RU = 256 KB).

The fewer but longer stops for higher utilizations in a RAID5 array with a given number of disks is due to the fact that as more user requests arrive, the disk spends more time serving them thus allowing the buffer to be utilized at a lower value as also shown in Figures 3.10 and 3.11. Moreover, the stops are longer as it incorporates some of the time required to serve the user request.

These trends also suggest that larger the number of disks, the flatter the curve. This suggests that larger number of disks in the array would result in slow decrease in the number of stops.

Finally the trend in the number of stops is studied when different buffer sizes are used at a given utilization. Table 3.6 and Figure 3.12 show this trend for a RAID5 system consisting of 11 disks and 21 disks at a normal mode utilization of 0.3.

Table 3.6 Number of Stops for Different Buffer Sizes at U = 0.3

Buffer Size (MB)	RAID5 with 11 disks	RAID5 with 21 disks
16	3448	12342
32	1612	7472
64	513	1197
128	0	0

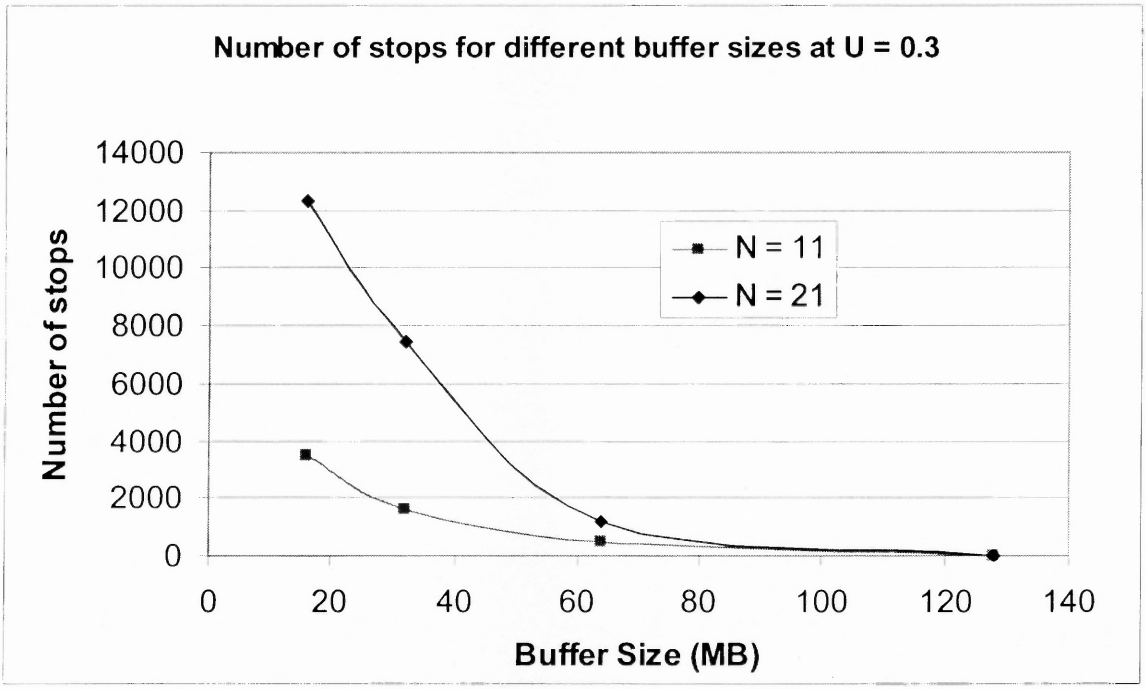

Figure 3.12 Number of stops for different buffer sizes at U = 0.3 (RU = 256 KB).

As it can be seen from Figure 3.12, the number of stops is higher for disk arrays consisting of more disks and lower for disk arrays consisting of fewer disks. However,

from the previous simulation studies already discussed within this chapter, it is seen that while the number of stops is lower, the time/stop in milliseconds is higher.

Another observation that merits attention is that the number of stops decreases with the increase in buffer size. This explains the fall in rebuild time as the buffer size is increased and inches closer to 128 MB. At buffer size of 128 MB, the average number of stops is zero. While there are indeed a few stops, these are too few and justifies our assumption that for the disk used, i.e., IBM 18ES, a 128 MB buffer may be considered infinite resulting in maximum efficiency during rebuild.

These trends are further confirmed in Table 3.7 and Figure 3.13 which examines the trend in the number of stops for N = 11 and N = 21 for a normal mode utilization of 0.45.

Table 3.7 Number of stops for Different Buffer Sizes at U = 0.45

Buffer Size (MB)	**RAID5 with 11 disks**	**RAID5 with 21 disks**
16	2965	9238
32	1143	5211
64	261	721
128	0	0

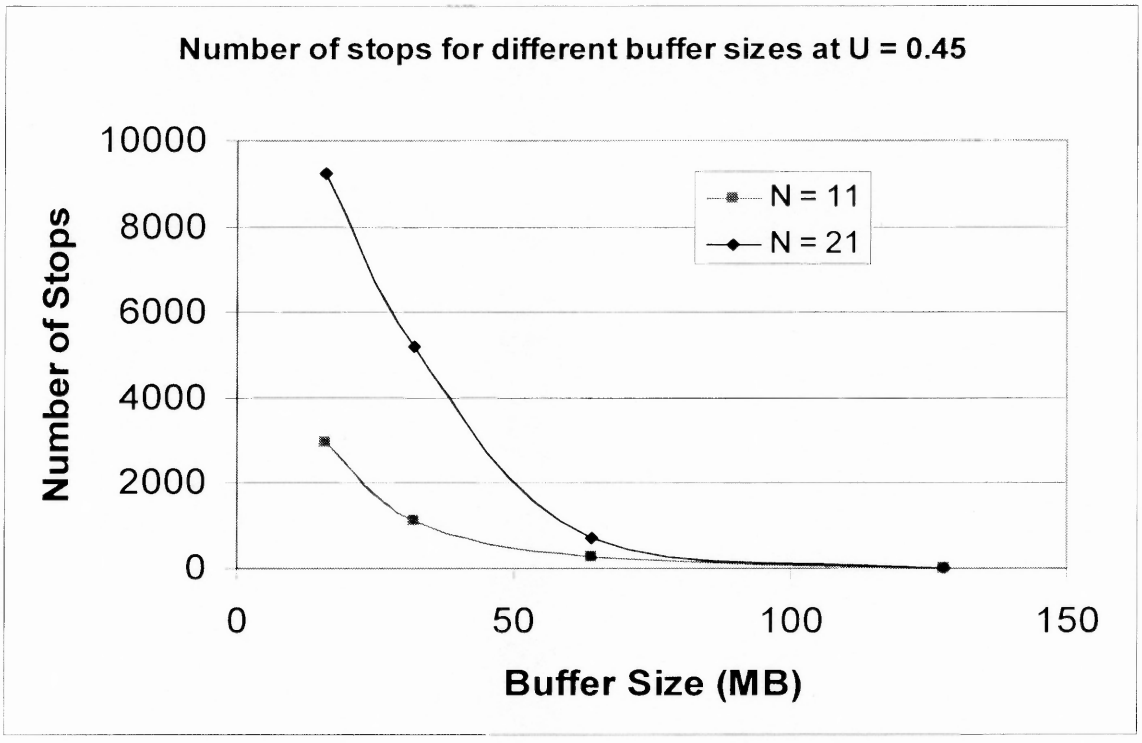

Figure 3.13 Number of stops for different buffer sizes at U = 0.45 (RU = 256 KB).

3.4 Dependence of Rebuild Time on Rebuild Buffer Size

As this study clearly shows, the rebuild time of a disk array depends to a fair extent on the size of the rebuild buffer. Other graphs in the above study also lead to the conclusion that the rebuild time depends on other factors too. An overview of the dependence of rebuild time on other factors is given in section 2.4 of this work.

The following two graphs lead to the conclusion that the rebuild time is a function of both disk utilization (U) and buffer size (B). Note that rebuild time is also a function of other factors including number of disks (N) but are not considered in the following figures

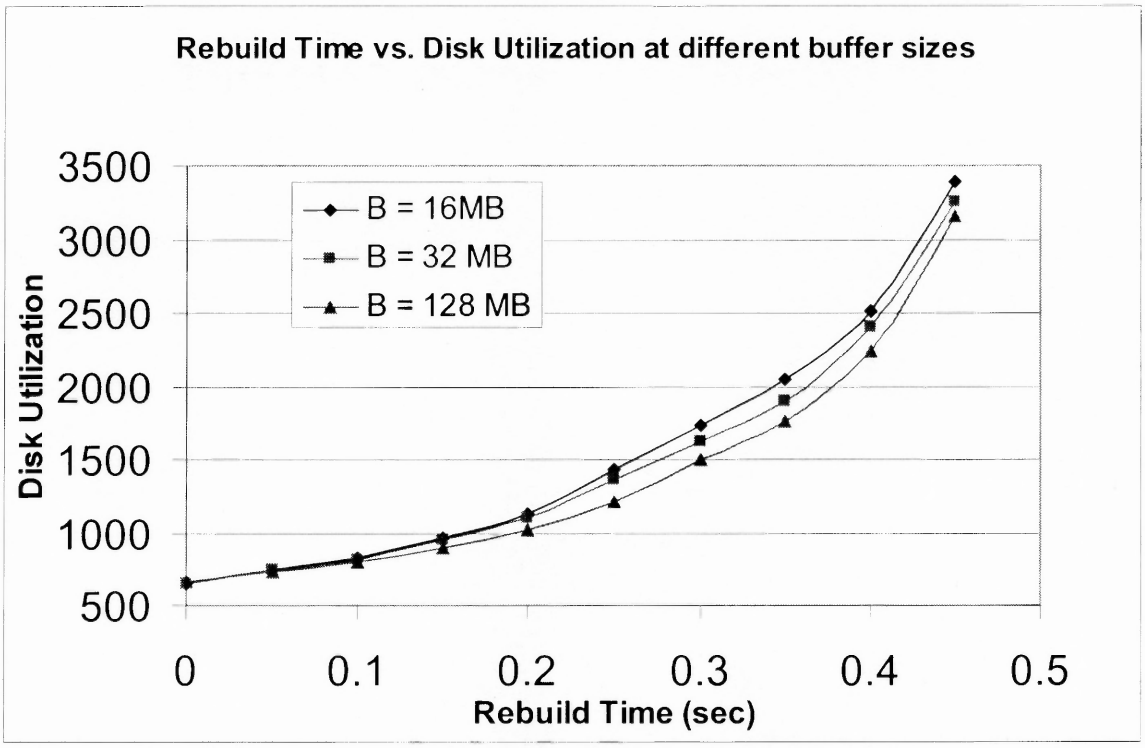

Figure 3.14 Rebuild Time vs. Disk Utilization at different buffer sizes (RU = 256 KB, N = 21).

From Figure 3.14, it can be seen that while there is very little difference in the rebuild time at disk utilization of 0, the difference opens up and is the highest at disk utilization of 0.25 which also represents the peak buffer utilization as seen in Figure 3.7. At this level, the difference between rebuild time with a buffer size of 128 MB (which can be considered infinite) and that with a buffer size of 16 MB is a little above 13%. This represents a sizeable difference and cannot be ignored in calculations. Figure 3.15 shows rebuild time vs. buffer size at different utilizations.

From this figure, a few observations can be made. These include that rebuild time decreases with increase in disk utilization and an increase in buffer size. Observing the change in the rebuild time also shows, higher the utilization, the greater the change in

rebuild time as the buffer is varied. However, the percentage of change is the highest for utilizations of 0.2 and 0.3 which is close to about 14%.

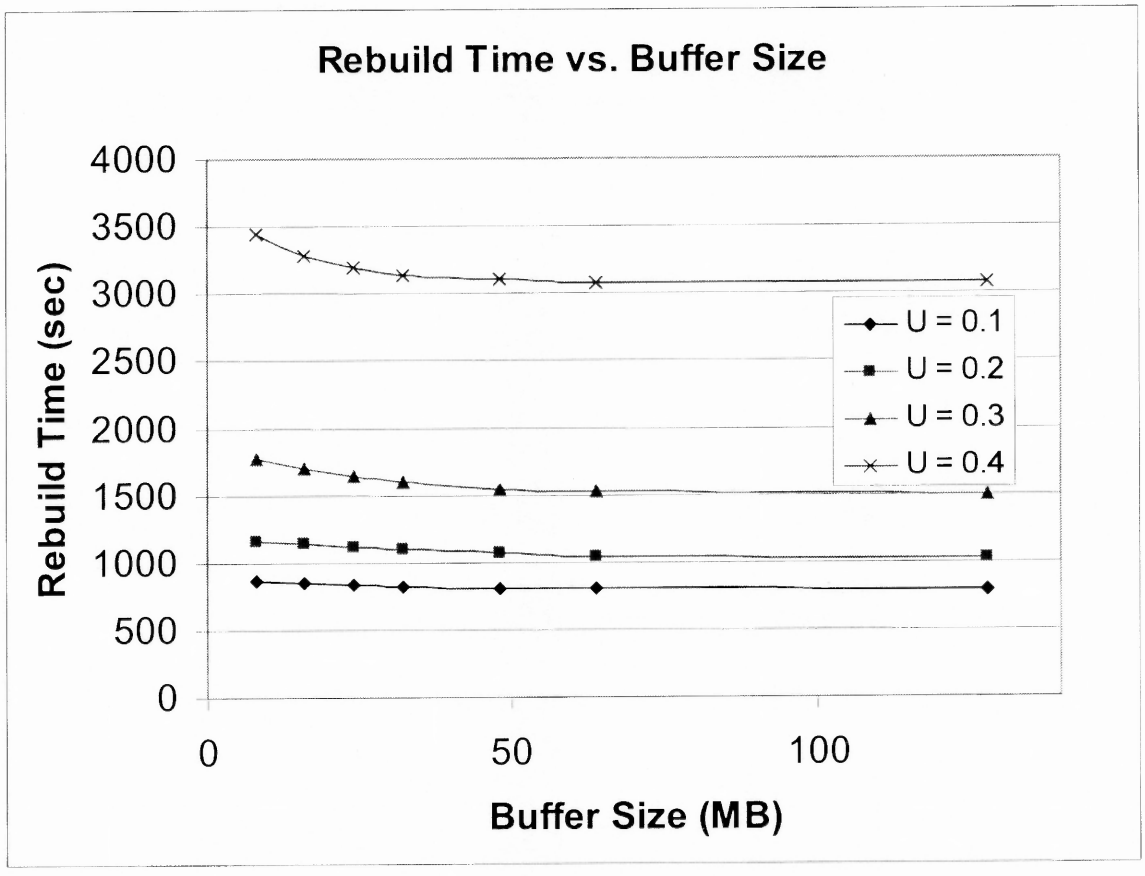

Figure 3.15 Rebuild Time vs. Buffer Size at different disk utilizations (N = 21, RU = 256KB).

3.5 Conclusion

So far, a majority of studies conducted on rebuild strategies were done assuming that an infinite rebuild buffer exists. The real buffer size on modern disks is not known as these are classified as trade secrets by manufacturers.

One of the most important conclusions from this study is that rebuild times and other parameters start varying widely after the buffer size is reduced below 128 MB. Since the disk used in these simulations (IBM 18 ES) rotates at 7,200 rpm, this size can be considered infinite for these disks. However, as newer disks with 15,000 rpm enter the market, the limited buffer will present a bottleneck to the rebuilding of disk arrays made out of these. This study brings forth the various parameters to be considered when designing disk arrays.

The following conclusions can be drawn from the detailed studies of limited buffer sizes: The rebuild time increases exponentially when the buffer size is made smaller than a cut-off level. In this case, the rebuild time increases appreciably after a buffer size of 64 MB. This is the result of multiple stops that each disk is forced to make as the buffer is full and cannot contain newer data that must be written to the spare disk for rebuilding.

The buffer utilization, and consequently the number of stops for a given buffer size, is low for low disk utilization levels and progressively increases and peaks between normal mode disk utilizations of 0.25 and 0.3. Above this utilization, the buffer utilization (and number of stops resp.) starts reducing again. This leads to the conclusion that for disk arrays supporting applications that generate moderate disk utilization levels (in the range of 0.2 to 0.4) need higher buffer sizes than those that generate low or high disk utilization levels.

CHAPTER 4

TRACK BUFFER

4.1 Problem Statement

When a RAID5 array enters into the rebuild mode, each surviving disk must serve two kinds of requests:

1. Rebuild requests – Data from these will be used to rebuild the data on the failed disk
2. User requests – Data requested by the user

When the user requests data and the array is in degraded mode of operation, those requests that require data from the failed disk are served by initiating a fork-join request to all the surviving disks. The data read from all the surviving disks must be XORed to obtain the data on the failed disk to satisfy the user request. This causes the load on the surviving disks to double for read requests.

When a disk failure occurs and a hot spare is available, disk rebuild begins. Using the VSM model, rebuild requests are issued and treated at a lower priority to user requests. The smallest unit that is read, once a rebuild request is decided to be served, is a Rebuild Unit (RU). Once this rebuild unit has been completely read, all the pending user requests must be served before the next RU is read. Serving the user request may lead the read/write head to jump to a different track on the disk. When all the user requests are served, the disk head reaches the first track to read the subsequent RU.

However, the disk head may land on a different section of the track and may have to wait till it reaches the first sector of the RU that must be read. The track buffer aims to make this time that the disk wastes while waiting to reach the first sector of the next RU, useful.

4.2 Track Buffer Concept

The track buffer is a small cache aboard each disk that buffers all the sectors starting with the first sector on which the track head lands till the first sector from which the RU that is to be served, begins. The concept is better explained in Figure 4.1

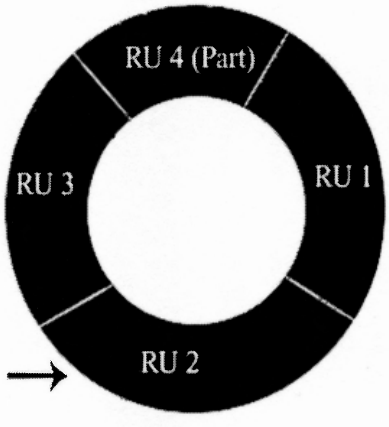

Figure 4.1 Schematic representation of a track containing 3 complete and 1 partial RU.

Figure 4.1 shows a particular track on a disk which contains 3 complete RUs (RU 1, 2 and 3) and a partial RU (RU 4). Assuming that the next rebuild request is to fetch RU 1. However, when the disk head reaches this track, it arrives at a sector pointed by the

arrow, which is a part of RU 2. In the absence of the track buffer, the disk must wait till it reaches the first sector of RU 1.

With the availability of the track buffer, the disk starts buffering all the sectors beginning from the first sector on the track that it can read. It does this till the time it reaches the first sector of RU 1. At this point, it starts sending out the RU 1 to the disk array controller for rebuild processing. When the whole RU 1 is read and the track starts processing RU 2, it needs to read only those sectors of RU 2 that haven't already been read into the buffer. Once all those sectors are read, the remaining sectors of RU 2, all sectors of RU 3 and all the sectors of RU 4 can directly be served from the buffer. Ensuring that the RUs are read in strict order by the disk obviates the need of associating RU numbers.

This technique does not affect the user response time as it utilizes the unproductive time and does not interfere in the manner in which user requests are served.

It can be assumed that buffer read time is far lower than that of the disk. In this manner, the use of a track buffer can lead to speeding up of rebuilding and result in lower disk utilizations as well as faster rebuild times.

4.3 Buffer Parameters and Simulation Assumptions

The disk used in the simulations used to study this effect of the track buffer is the IBM 18ES. According to the disk specifications [8], this 9.17 GB, 7,200 rpm zoned disk has 11 zones and have between 247 and 390 sectors/track. Each sector is 512 bytes, which makes the size of the largest sector equal to 390 sectors/track x 512 bytes/sector = 195 KB/track.

As shown from the previous section, in the worst case, the buffer may be required to store one full track. This means that the buffer size must be equal to the track with the highest capacity. Each disk in the RAID5 disk array must have a 195KB buffer.

Other simulation parameters are:

1. RAID 5 system with 21 disks (unless otherwise mentioned)
2. FCFS scheduling
3. Vacationing Server Model [3] for rebuilding
4. Track Buffer Size of 256KB (greater than 195KB)
5. Poisson arrival of user requests
6. Read requests to 4KB blocks
7. Normal Mode Utilizations varied from 0 to 45%

4.4 Simulation Results

The rebuild unit size is one of the major factors that determine the rebuild time of a RAID 5 arrays. Other factors were looked into in the previous chapter. Given that the VSM model does not allow user requests to pre-empt rebuild requests, a very large RU size will cause the disk to rebuild faster, but increase user response time. Meanwhile, a very small RU size will speed up user response but increase the rebuild time.

It has been shown in [13] that the Mean Time To Data Loss (MTTDL) varies with the Mean Time To Failure (MTTF) and the Mean Time To Repair (MTTR) of the disk. This makes the rebuild times of disks very critical as it directly impacts the MTTDL.

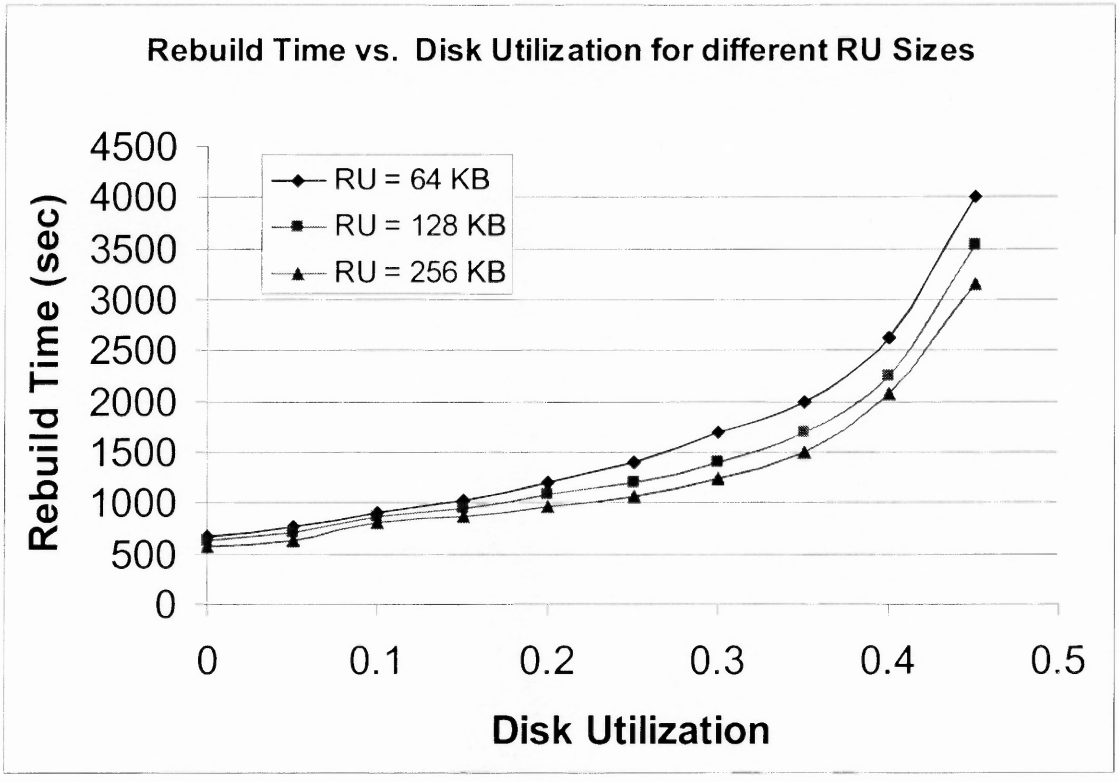

Figure 4.2 Rebuild Time vs. Disk Utilization at different RU sizes (N = 21, B = 256 MB)

The Figure 4.2 shows the variation of Rebuild Time with respect to Normal Mode Disk Utilization (U) for different RU sizes. As it can be seen, the larger the RU size, the faster is the rebuild. It can also be seen that the improvement in the rebuild time for larger RU size is greater for higher utilizations. This can be explained by considering the fact that for higher utilizations, a smaller RU allows greater user requests to be served while a larger RU allows fewer requests.

When the disk starts reading the first track for rebuilding, it may be interrupted by user requests and may have to serve that once the first RU has been read. The following figures study the number of times the first track has to be revisited so that the whole track

can be read by rebuild requests. The first track has 390 sectors (195 KB) and only rebuild requests are considered.

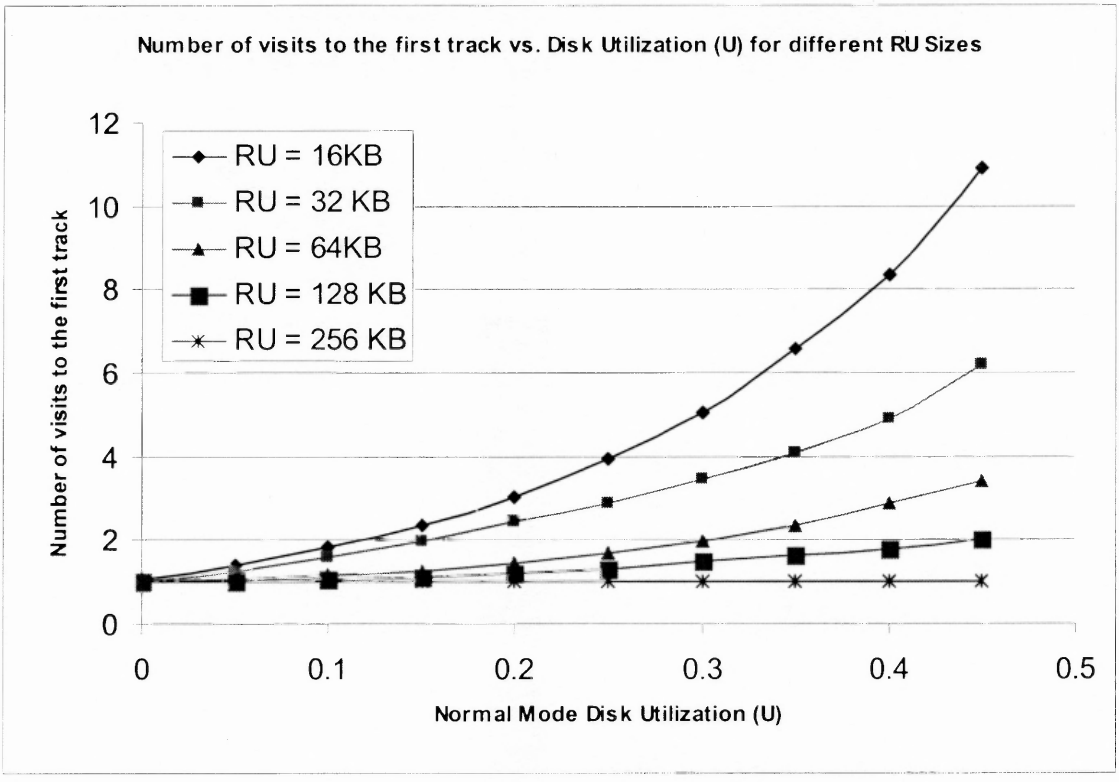

Figure 4.3 Number of times the first track is visited before it could be completely rebuilt (N = 21).

Figure 4.3 shows that the track has to be visited a multiple number of times before it can be completely rebuilt. It may be noted that for RU size = 256 KB, the track is read only once since this is greater than the size of that track, which is 195 KB.

While the most common RU sizes for for this disk would probably be 64 KB, 128 KB and 256 KB, the RU sizes of 32 KB and 16 KB are also considered. Utilizing the track buffer can bring down this number and thus improve rebuild time.

Figure 4.4 shows the time spent on the first track while it is being read. Note that this is calculated only when the read/write head is reading the first track that has to be rebuilt. It is very similar to Figure 4.3 as the number of visits to the track is directly related to the amount of time spent reading it. This graph however, enables us to gain a better measure of how much time is wasted and could be saved by the track buffer.

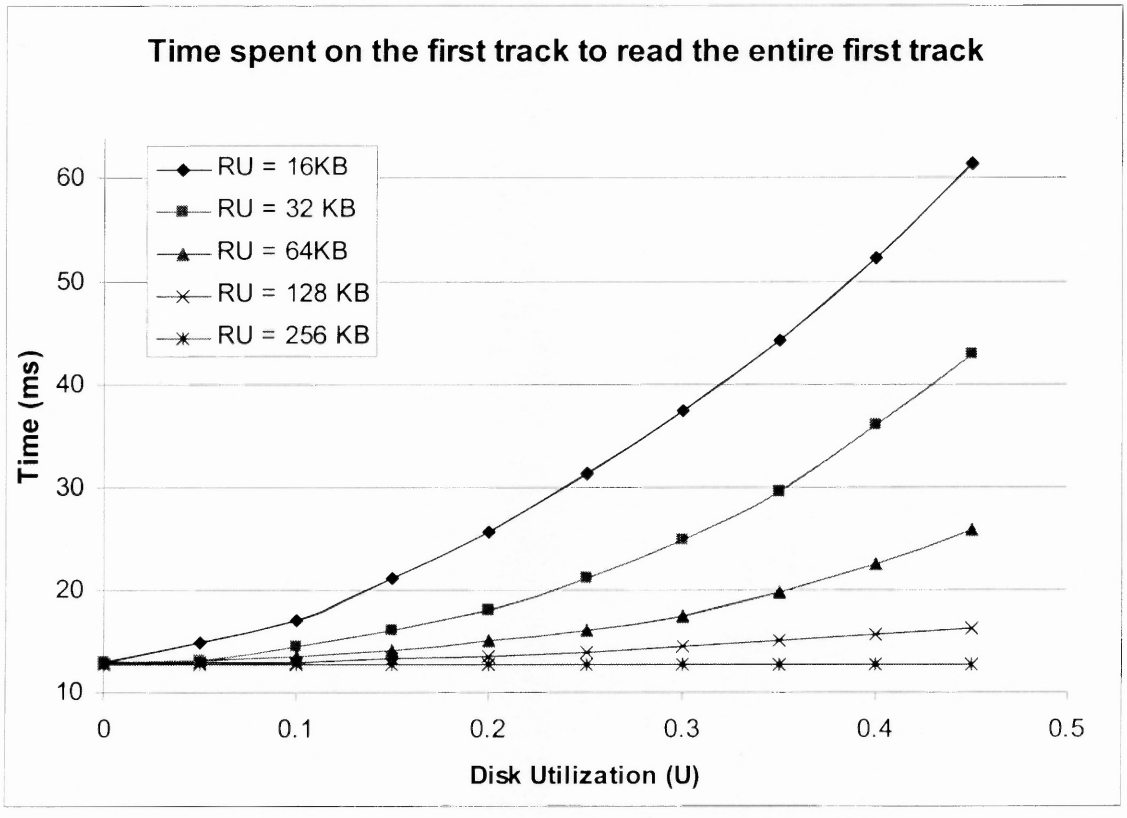

Figure 4.4 Amount of time spent on the first track to be rebuilt, to read the entire track (N = 21).

A careful study of Figures 4.3 and 4.4 shows that there is a great deal of inefficiency involved in reading the track.

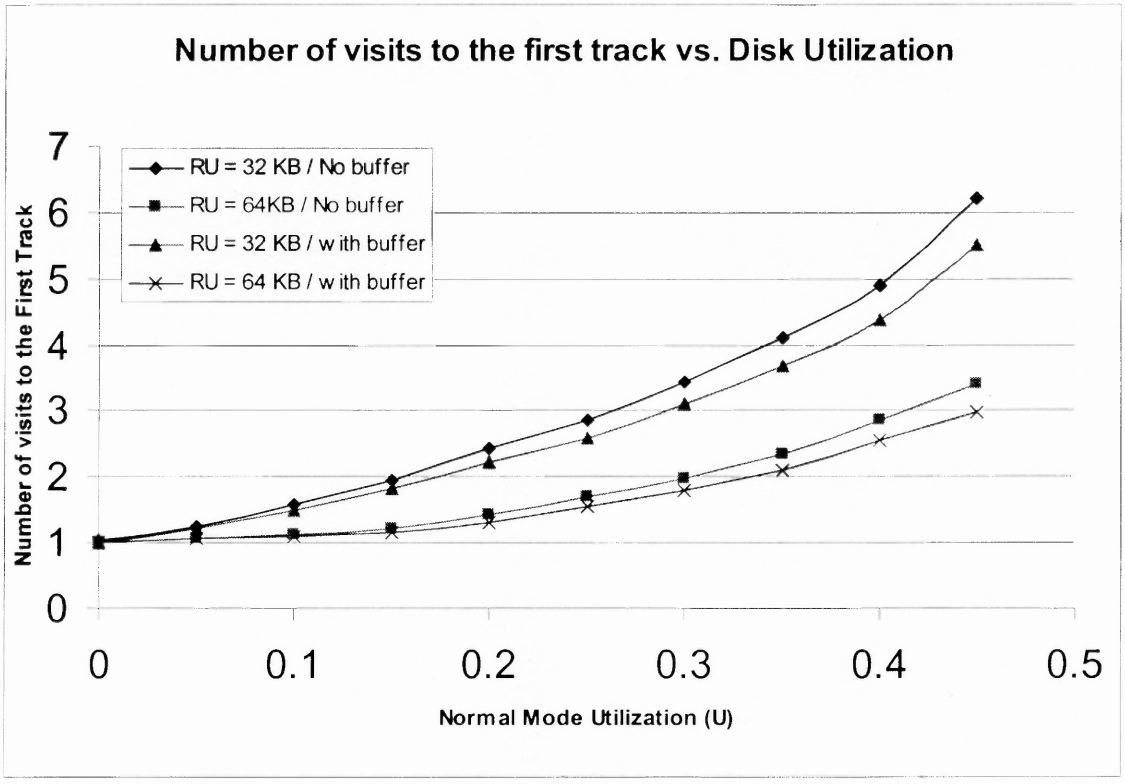

Figure 4.5 Number of times the first track is visited with and without track buffer (N = 21).

The Figure 4.5 shows the improvement in the number of times the first track had to be visited when the track buffer is enabled. It must however be noted, that as the rebuild progresses, the number of times a track has to be visited will change since some of the requests may be served directly from the rebuilt disk. The number of times that a track is visited also changes with the size of the track which is variable due to Zoning.

From this study, it can be seen that by the use of the track buffer, the number of revisits is reduced by about 13% for a 32 KB RU size. This allows for the estimation of the improvement that can be expected by implementation of this concept. However, it must also be remembered that this number has been obtained for the first track, when

rebuilding begins. Other tracks that are to be rebuilt are smaller than or equal to the size of this track.

Table 4.1 Variation of the Response Time with Utilization for RU = 64 KB

Utilization	Response Time (ms) without buffer	Response Time (ms) with buffer
0.00	11.88	11.88
0.05	14.52	14.50
0.10	17.07	17.06
0.15	19.20	19.18
0.20	20.96	20.96
0.25	22.01	21.99
0.30	22.45	22.40
0.35	26.77	26.73
0.40	33.62	33.57
0.45	40.10	40.04

From the concept of the track buffer, it is clear that user response time is not impacted by its implementation. Simulation results for the response time with and without the track buffer confirm this. They are shown in the table 4.1. The table shows a marginal improvement in the user response time which may be attributed to the fact that with the enabling the buffer, the rebuild time reduces and to measuring inaccuracies.

Thus, implementation of the track buffer will not result in any visible changes to the user response. It does however improve the rebuild time. This can be seen from the results shown in Figure 4.6 and table 4.2.

These results show an improvement of 8.57% for an RU size of 32KB at peak utilization. Other results for larger RU sizes show similar improvement.

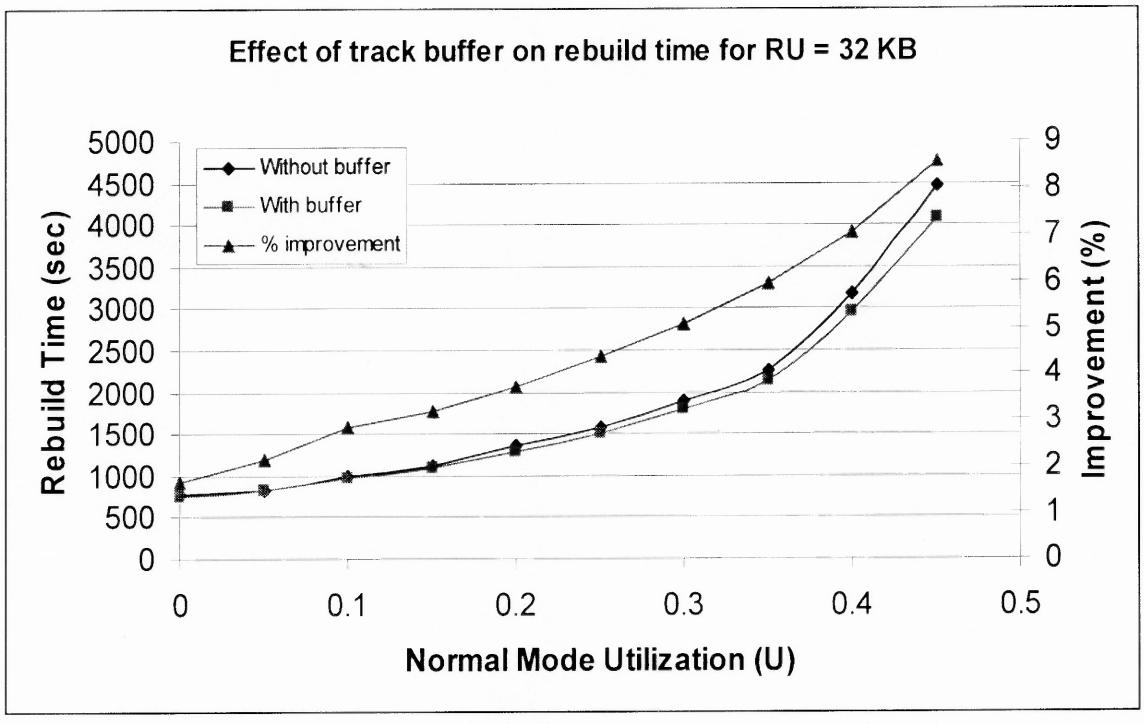

Figure 4.6 Effect of track buffer on the rebuild time for RU = 32 KB (N = 21, B = 256 MB).

Table 4.2 Rebuild Times for Different Utilizations for RU = 32 KB

Utilization (U)	T_r without buffer (sec)	T_r with buffer (sec)	% Improvement
0.00	774	761	1.68
0.05	833	815	2.16
0.10	992	964	2.82
0.15	1124	1088	3.20
0.20	1348	1298	3.71
0.25	1578	1509	4.37
0.30	1897	1801	5.06
0.35	2266	2131	5.96
0.40	3187	2963	7.03
0.45	4469	4086	8.57

Similar trends are observed for RU = 64 KB and RU = 128 KB as follows:

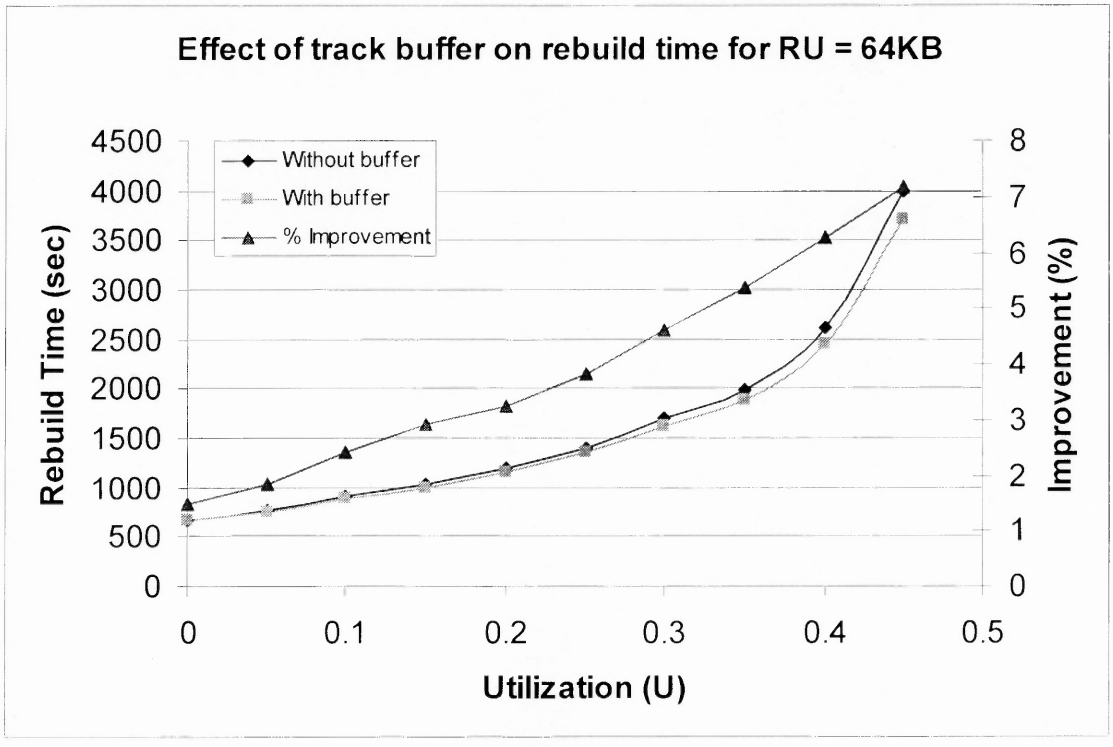

Figure 4.7 Effect of track buffer on the rebuild time for RU = 64 KB (N = 21, B = 256 MB).

Table 4.3 Rebuild Times for Different Utilizations for RU = 64 KB

Utilization (U)	T_r without buffer (sec)	T_r with buffer (sec)	% Improvement
0.00	673	663	1.49
0.05	763	749	1.83
0.10	912	890	2.41
0.15	1024	994	2.93
0.20	1198	1159	3.26
0.25	1408	1354	3.84
0.30	1697	1619	4.60
0.35	1993	1886	5.37
0.40	2621	2457	6.26
0.45	3998	3712	7.15

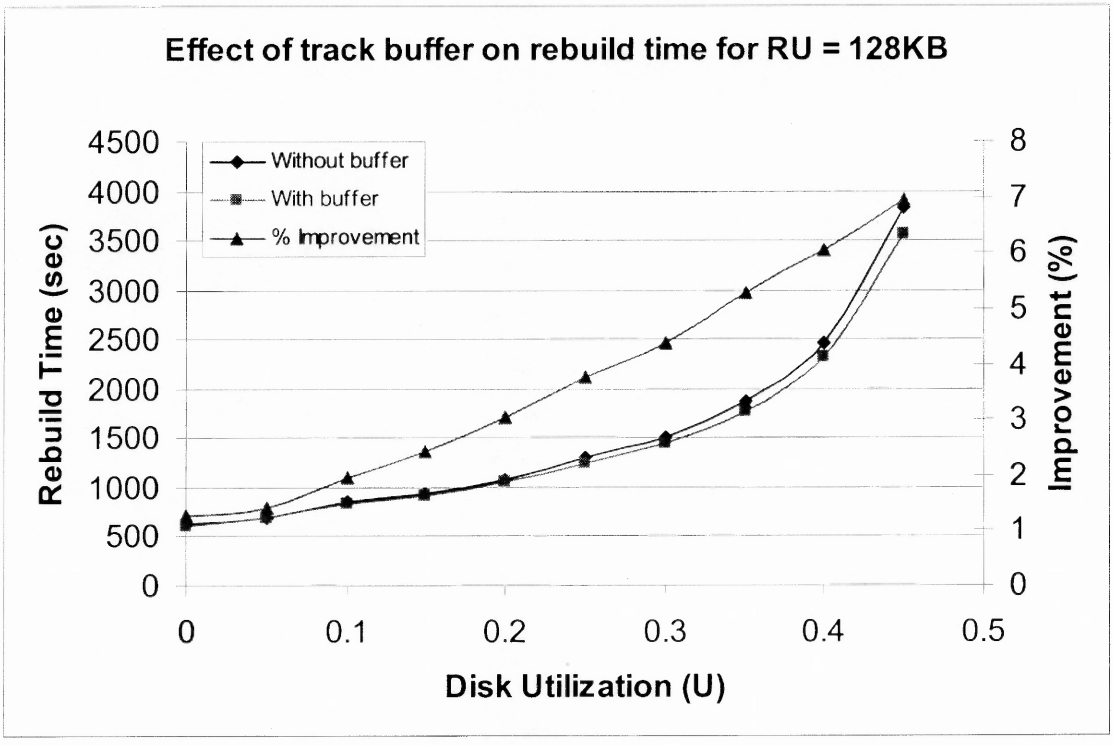

Figure 4.8 Effect of track buffer on the rebuild time for RU = 128 KB (N = 21, B = 256 MB).

Table 4.4 Rebuild Times for Different Utilizations for RU = 128 KB

Utilization (U)	T_r without buffer (sec)	T_r with buffer (sec)	% Improvement
0.00	628	620	1.27
0.05	701	691	1.43
0.10	862	845	1.97
0.15	946	923	2.43
0.20	1088	1055	3.03
0.25	1298	1249	3.78
0.30	1509	1443	4.37
0.35	1875	1776	5.28
0.40	2461	2312	6.05
0.45	3828	3562	6.95

These figures and tables show that the rebuild time can be improved with the implementation of the track buffer. Some of the conclusions that can be drawn from these results are as follows:

Higher the disk utilization, the greater is the improvement due to the use of the track buffer. This can be explained by the fact that as more user requests have to be served, the disks are more likely to be interrupted between reading successive RUs. As a result, the reading head has to go to other tracks regularly and arrive at the same track again to read the next RU. This will result in increased utilization of the track buffer.

As the size of the RU is increased, the improvement due to the use of the track buffer decreases. As the size of the RU increases, more and more of the track will be read at once since rebuild requests cannot be pre-empted. Thus, if a RU is greater than or equal to the size of the track, the track will never be revisited twice for a rebuild request. This means that the only time the track buffer is used is when the track head first arrives on the track and can buffer a few sectors before it reaches the start of the RU on that track. This reduces the use of the track buffer, rendering it ineffective.

The percentage of improvement is nearly linear after normal mode disk utilizations of 0.15. This suggests a direct and constant variation with respect to disk utilization.

While a lower RU size of 16 KB or even 8 KB will certainly yield more improvement in user response time, it is pointless to discuss those as it would have no practical use. Given that requests are for 4 KB blocks, having an RU of 8 KB would mean a RU of 2 blocks. Moreover, many OLTP applications have block sizes of 16 KB.

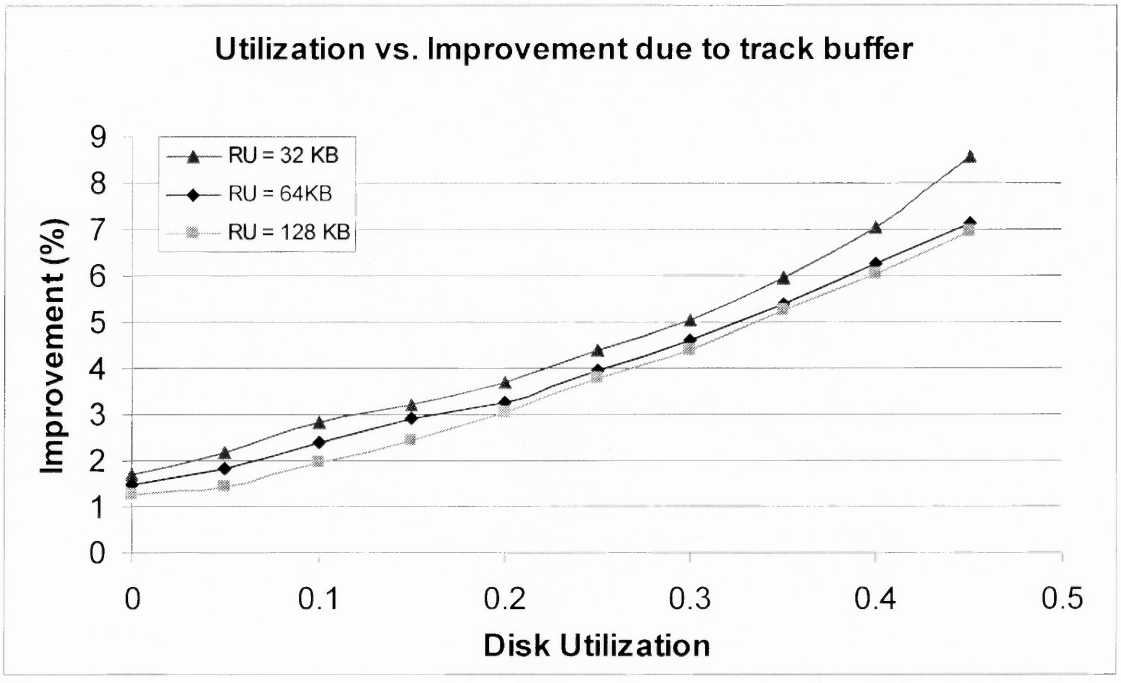

Figure 4.9 Improvement due to track buffer at different utilizations and RU sizes (N = 21, B = 256 MB).

Finally, the Figure 4.9 shows the improvement due to the implementation of the track buffer at different utilizations. It shows that the improvement holds steady for all utilizations and follows the general trends, viz. Greater improvement for higher utilizations and lower RU sizes.

It can also be seen from the same figure that as the RU size is increased, the change in the improvement starts to become progressively insignificant.

4.5 Conclusion

The detailed simulations above comprehensively prove that by using the track buffer mechanism, an improvement can be made to the rebuild time. Moreover, it is also worthy to note that this improvement does not come at the expense of the user response time.

As the figures above suggest, an impressive amount of improvement to the rebuild time is not achieved. The improvement is around 7% for a 128 KB RU and a little higher for smaller RU sizes.

Despite preliminary studies, regarding the number of visits to the first track and the amount of time spent on the first track, given in Figures 4.3 and 4.4, showing a great deal of inefficiency in the reading process at higher disk utilizations, the implementation of the track buffer does not show the same amount of improvement. This suggests that while the track buffer is successful in reducing the inefficiency, there may still be small parts of the track that have never been read into the buffer causing the head to read the same parts of the track multiple times.

While this may not be very impressive at the first glance, this technique may become important for larger disks that are already available in the market. It must be borne in mind that this simulation was conducted for a 9.17GB disk, while 100GB hard drives are very common in the market as of this writing.

For such high capacity drives, the RU size may not have been proportionally increased. This could result in an improvement of more than 10% for larger disks as the RU may be a smaller fraction of each track. It was however not possible to conduct simulation studies with newer disks as the detailed specifications for these are unavailable from disk manufacturers. The simulator used for deriving the results quoted

here uses disk specifications that are publicly available through the website of Parallel Data Labs at Carnegie Mellon University. [8]

In the future, this experiment must be conducted with faster, higher capacity drives and RU sizes that would be relevant to the applications of the time. Moreover, since this method of Rebuild Time improvement does not result in deterioration of the user response time, the only compromise to be made is the additional cost incurred in fabricating the few hundred KB of cache memory on to each disk and modifying the disk controllers to use the cache in the rebuild mode.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

The study presented in this work was aimed at presenting a complete and elaborate insight into rebuild in RAID5 systems. As is evident from all the work presented, rebuild is a complicated process – one which is influenced by several factors such as Rebuild Unit Size which has already been investigated in previous studies.

The study of limited rebuild buffer assumes importance in view of the fact that newer disk drives, which have higher capacities, will require larger rebuild buffer sizes to function as predicted by studies which assume infinite buffers. A 128 MB rebuild buffer is shown to be equivalent to infinite for a 7,200 rpm, 9.17GB disk. Providing larger buffers to newer disks whose capacities exceed 100GB and rotation speed exceed 15,000 rpm may not be possible, meaning that the impact of buffer size on rebuild time will be increasingly felt. It is seen that as the buffer size is reduced beyond a cut-off point, it starts becoming a bottleneck during rebuild. RAID arrays of larger disks will have to either provide larger rebuild buffer sizes or account for the loss of rebuild performance as shown in this study.

Finally, this work also presented the concept of the track buffer and simulation results that prove its effectiveness. This concept demonstrates a technique by which rebuild time can be improved without any increase in user response time. While the figures for improvement in rebuild time shown in this study may not be very impressive, it must be noted that for larger disk drives an RU size of 256KB may be only a small

percentage of a track. This would mean greater improvement in the rebuild time, Mean Time to Data Loss (MTTDL) and ultimately reliability.

5.2 Future Work

While this study looks at a broad range of values, the typical values of many parameters remain unknown since disk and disk array manufacturers refuse to release detailed data that is required for accurate simulations. The Parallel Data Lab [8] at Carnegie Mellon University has not released newer disk drive characteristics since 1998. There is no certain answer as to what RU size, Striping Unit size, etc. is used in RAID5 arrays available in the market today.

Most simulations were conducted on IBM18ES disk drive which is fairly outdated. Detailed simulations could not be conducted on newer disks due to the non-availability of specifications of such disks.

The results in this study are geared to provide trends that can be applied to newer disks with reasonable accuracy. However, more detailed research must be done as and when disk specifications and parameter values for newer disks become available.

APPENDIX

CONFIGURING DASIM FOR RAID5 SIMULATIONS

Disk Array Simulator (DASim), written in C/C++, has been developed at NJIT by Gang Fu and Chunqi Han under the guidance of Dr. Alexander Thomasian. It is a detailed simulator currently capable of simulating Single Disk, RAID0, RAID1, Clustered RAID5 (CRAID5) and Heterogeneous Disk Array (HDA) besides RAID5.

The RAID5 module in DASim uses command line arguments to set up the simulation environment. The normal and degraded mode is simulated in the file RAID5Sim.h, and the rebuild mode in RAID5Sim2.h. The file RAID5Frame.h contains definitions for the RAID5 controller which in turn use the definitions of the single disk controller. These are defined in SDFrame.h. Each simulation generates 4 files in a folder whose name can be specified directly into the main function.

The file of primary interest is rebuild.txt which gives the statistics for rebuild and mean response times. The following parameters can be directly specified from the highest level of access: buffer size, RU size, read redirection, model, model parameters, rebuild type, request arrival rate, results folder name, name of the disk, scheduling policy (FCFS/SATF), number of priority classes, threshold, number of disks (N), failed disk, stripe unit size, read/write ratio, number of requests processed before rebuild, number of requests processed after rebuild, and the cache size.

REFERENCES

[1] Ahmed, A (2005). Rebuild Performance Enhancement using Onboard caching and Delayed Vacation Termination in Clustered RAID5. Thesis submitted to NJIT College of Computing Sciences, May 2005.

[2] Fu, G. Thomasian, A. Han, C. and Ng, S.W. (2004). Rebuild strategies for clustered redundant disk arrays. *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems – SPECTS'04,* San Jose, CA, July 2004, pp. 598-607.

[3] Fu, G. Thomasian, A. Han, C. and Ng, S. W. (2004). Rebuild strategies for redundant disk arrays. *Proceedings of the IEEE/NASA Conference on Mass Storage Systems and Technologies,* College Park, MD, May 2004, pp. 3-19

[4] Gibson, G. A. (1992). *Redundant Disk Arrays: Reliable, Parallel Secondary Storage.* MIT Press, Boston.

[5] Hennessy, J. L. Patterson, D. A. and Goldberg, D. (2003). *Computer Architecture: A Quantitative Approach (3rd Ed).* Morgan-Kauffman Publishers.

[6] Holland, M. Gibson, G. A and Siewiorek, D. P. (1994). Architectures and algorithms for online failure recovery in redundant disk arrays. *Journal of Distributed and Parallel Databases, 2 (3)*, pp. 295-335.

[7] Holland, M. Zelenka, J. et al (1996). RAIDFrame : A Rapid Prototyping Tool for RAID Systems. Parallel Data Laboratory, Carnegie Mellon University.

[8] Database for validated disk parameters for DiskSim. Parameters posted on the website of Parallel Data Lab at Carnegie Mellon University. Retrieved on April 28, 2006 from the world wide web:
http://www.pdl.cmu.edu/DiskSim/diskspecs.html

[9] Lee, E. and Katz, R. (1991). Performance consequences of parity placement in disk arrays. *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems,* Santa Clara, CA, April 1991, pp. 190-199.

[10] Merchant, A. and Yu, P. S. (1996). Analytic Modeling of Clustered RAID with Mapping Based on Nearly Random Permutation. *IEEE Trans. Computers 45(3)*, pp. 367-373.

[11] Muntz, R.R. and Lui, J. C. S. (1990). Performance analysis of disk arrays under failure. *Proceedings of the 16th VLDB Conference,* Brisbane, Australia, August 1990, pp. 162-173.

[12] Ng, S. W. and Mattson, R. L. (1994). Uniform parity distribution in disk arrays with multiple failures. *IEEE Trans. Computers 43 (4)*, pp. 501-506.

[13] Patterson, D. Gibson, G. and Katz, R. A. (1988). A case for redundant arrays of inexpensive disks (RAID), *Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD),* Chicago, IL, June 1988, pp. 109-116.

[14] Thomasian, A. (1995). Rebuild options in RAID5 disk arrays. *Proceedings of the 7th IEEE Symposium on Parallel and Distributed Systems,* San Antonio, TX, October 1995, pp. 511-518.

[15] Thomasian, A. Fu, G. and Ng, S. W. (2006). Analysis of rebuild processing in RAID5 disk arrays. *The Computer Journal, Oxford University Press, to appear 2006.*

[16] Thomasian, A. Han, C. Fu, G. and Liu, C. (2004). A performance tool for RAID disk arrays. *Quantitative Evaluation of Systems (QEST'04),* Enschede, The Netherlands, September 2004, pp. 8-17.

[17] Thomasian, A. and Liu, C. (2004). Performance evaluation of variations of SATF disk scheduling policy. *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems – SPECTS'04,* San Jose, CA, July 2004, pp. 431-437.

[18] Thomasian, A. and Menon, J. (1997). RAID5 performance with distributed sparing. *IEEE Transactions on Parallel and Distributed Systems – TPDS 8 (6).* June 1997, pp. 640-657.

[19] Thomasian, A. and Menon, J. (1994). Performance analysis of RAID5 disk arrays with a vacationing server model. *Proceedings of the 10th IEEE conference on Data Eng. – ICDE'94,* Houston, TX, February 1994, pp. 111-119.