IMPROVING RELIABILITY, ENERGY-EFFICIENCY AND SECURITY OF STORAGE SYSTEMS AND REAL-TIME SYSTEMS

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

_____

Kiranmai Bellam

Certificate of Approval:

_____	_____
David Umphress Associate Professor Computer Science and Software Engineering	Xiao Qin, Chair Assistant Professor Computer Science and Software Engineering
_____	_____
Cheryl Seals Assistant Professor Computer Science and Software Engineering	George T. Flowers Interim Dean Graduate School

IMPROVING RELIABILITY, ENERGY-EFFICIENCY AND SECURITY OF STORAGE SYSTEMS AND REAL-TIME SYSTEMS

Kiranmai Bellam

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulllment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 10, 2009

IMPROVING RELIABILITY, ENERGY-EFFICIENCY AND SECURITY OF

STORAGE SYSTEMS AND REAL-TIME SYSTEMS

Kiranmai Bellam

_____

Signature of Author

_____

Date of Graduation

DISSERTATION ABSTRACT

IMPROVING RELIABILITY, ENERGY-EFFICIENCY, AND SECURITY OF STORAGE SYSTEMS AND REAL-TIME SYSTEMS

Kiranmai Bellam

Doctor of Philosophy, August 10, 2009
(M.S. New Mexico Tech, 2006)
(B.S. University of Madras, India, 2003)

150 Typed Pages

Directed by Xiao Qin

Many life-critical systems are required to operate without a system failure for a given period of time. Examples are nuclear, aerospace, spacecraft and other such systems. Many of these applications are either storage intensive or real time intensive, hence it is crucial to look at the reliability improvements of computer systems, mainly storage and real time systems. With respect to storage system reliability, in the business arena, data preservation and data mining has proven to be a boon in shaping business strategy. For individuals, storage is being called upon to preserve sentimental and historical artifacts such as photos, movies and personal documents. In both these areas, storage must keep pace with a growing need for efficient, reliable, long term storage. When it comes to real time systems, performance is their most important

characteristic--right up until the point where it stops working. Then, suddenly, you don't care how fast it is--or rather, *was*. You just want it to start working again. The importance of reliability in both storage and real time systems is made clear by the above statements. Reliability of these two computer systems can be affected by a wide range of novel technologies, including energy conservation techniques and security mechanisms. In what follows, we describe two challenging issues of improving reliability in storage systems and real-time systems. The first issue is improving reliability and energy efficiency of storage systems, the second challenge is the improvement of reliability and security in real time systems.

Modern day storage systems offer high levels of performance and disk capacity at low costs. In this dissertation, we propose methods for building energy-efficient and reliable large scale storage systems. The primary focus of our research is to achieve the twin goals of maximizing reliability and minimizing energy consumption by incorporating these energy-efficient and reliable techniques to large scale storage systems. The experimental results using both synthetic and real world applications (traces) have shown that the energy consumption could be significantly reduced while guaranteeing maximum reliability for storage disks with a marginal degradation of performance.

Reliability of real time systems, second challenging issue, is studied in the later part of the dissertation. For real-time embedded systems we proposed techniques to integrate fault recovery and security services in real time embedded systems. Simulation results show that our techniques can significantly improve security over the conventional approaches, while achieving an efficient means of fault recovery.

ACKNOWLEDGMENTS

In the first place I would like to express my sincere gratitude to Dr. Xiao Qin for his supervision, advice and guidance for this research. Above all, he provided me with unflinching encouragement and support in various ways. Without his persistent help this dissertation would not have been possible.

I gratefully acknowledge my committee members, Dr. David Umphress and Dr. Cheryl Seals for guiding me through the dissertation process. I am very thankful to Dr. Shiwen Mao for serving as the outside reader and proofreading my dissertation.

I am also thankful to my research group, who include Xiaojun Ruan, Ziliang Zong, Adam Manzanares and Shu Yin, for their support and collaboration. I would like to take this opportunity to thank all my friends in Auburn who helped me during my study at Auburn University

My deepest gratitude goes to my parents Venkateshwar Rao Bellam and Rani Bellam, for their encouragement through my PhD. Special thanks to my brother Shyam Bellam, for injecting motivative and competitive spirit into me. I would like to extend my gratitude to my new family members, my parents-in-law for bringing new joy to my life.

Nitish Kosaraju, my husband, without him in my life this effort would have been worth nothing. Thank you for your endless love, support and encouragement. Thank you for believing in me.

Style manual: IEEE Standard for Research Papers

Software used: Microsoft Word 2007, Microsoft Excel 2007, Linux GCC Compiler, Microsoft Visio 2007, Eclipse, Adobe Photoshop, C/C++/Java

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

The increasing number of large scale complex applications call for high performance computing platforms such as large scale storage systems, which provide a cost effective infrastructure for the most complicated scientific applications as well as commercial applications. Large scale storage systems translate directly to an increasing number of spinning disks creating a huge and growing energy crisis. Reliability is also an important characteristic of large scale storage systems, because sometime s the data stored on the disks might be mission critical, if not, the amount of energy saved by using some of the energy consumption techniques might have to be spent restoring the disk if it fails.

Hence there is a need for highly reliable and energy efficient storage systems. By proposing energy efficient techniques which guarantee reliability, this problem can be alleviated. The first objective of this research is to explore highly energy efficient technologies to reduce the power consumption of large scale storage systems while guaranteeing the reliability of the system.

Completing a task before the deadline is a very vital criteria in real time embedded systems. Along with the deadline, task security and fault tolerance are equally important in real time applications such as online banking and aircraft control systems. Hence there is a need for real time embedded systems which guarantee security while providing fault tolerance as two main goals. Many existing algorithms for real time embedded systems concentrated on one of the above goals while ignoring the other. To bridge this technology gap we have proposed two approaches to remedy this problem. The second objective of this dissertation is to integrate security and fault tolerance in real time embedded systems.

This chapter first presents the problem statement in Section 1.1. In Section 1.2, we describe the scope of this research. Section 1.3 highlights the main contributions of this dissertation, and Section 1.4 outlines the dissertation organization.

1.1 Problem Statement

In this section, we start with an overview of new trends in large scale storage systems. Section 1.1.2 introduces the serious data center energy crisis and section 1.1.3 presents the reliability issues of the disk systems, presenting the initial motivation for the first objective of our dissertation research, improving reliability and energy efficiency of storage systems. Section 1.1.4 outlines the security concerns of real time systems and section 1.1.5 details the importance of the reliability of real time systems. These motivated the second objective of our research, integrating the security and reliability of real time systems.

1.1.1 The Era of Large Scale Storage Systems

Commodity disk drives have made online storage a way of life because of their declining costs. Hundreds of thousands of data servers are deployed in the data centers like Google and Yahoo. These data centers make life easy by proving the data or processing the data with few mouse clicks. Many large scale applications such as scientific computing, weather forecast, search engines and medical research projects are only possible after the advent of large scale storage systems. These data centers not only make life easy but they are also very important for the survival of human species in this twenty first century: For example predicting a future flood alert using weather forecast.

1.1.2 The Data Center Energy Crisis

Energy crisis is a great bottleneck to any economy. Current worldwide data growth is estimated at a compound annual growth rate of 50.6% through the decade [7]. Several studies have indicated that data centres are headed towards a serious energy crisis. The energy crisis problem has a rippling effect as additional power is required. For instance 22% of the total energy consumption is due to cooling systems used to keep temperatures of data centres from rising high degrees. The solution to power consumption problem has been addressed at the hardware level to a certain extent. But the problem remains largely open in many deployments of large scale storage systems. Energy consumption of storage systems in data centres is becoming a predominant concern in the IT industry. Increasing evidences have shown that the powerful computing capability of data centers is actually in the cost of huge energy consumption. For example, Energy User News stated that the power requirements of

today's data centers range from 75 W/ft^2 to 150-200 W/ft^2 and will increase to 200-300 W/ft2 in the nearest future. The new data center capacity projected for 2005 in U.S. would require approximately 40 TWh ($4B at $100 per MWh) per year to run 24x7 unless they become more efficient. The supercomputing center in Seattle is forecast to increase the city's power demands by 25%. Even worse, the EPA predicted that the power usage of servers and data centers will be doubled again within five years if the historical trends are followed. However, most previous research about large scale storage systems primarily focused on the improvement of performance, security, and reliability. Energy conservation issue is neglected because it is taken for granted. With the growing energy crisis, now it has come in to attention of many researchers and industrialists. Many pioneer research groups started addressing the energy crisis issue but lot has to be done before this problem is totally rectified. Our research is motivated by the current need of energy efficiency for data centers.

1.1.3 Reliability of the Disk Systems

Reliability is a major concern in today's world because of the importance of data that is stored on the disks. Enormous amount of data is stored on disks on a daily basis and all this data need to be saved for later use or for the next generations. It is not possible to keep multiple copies of this data to improve reliability because of the limited resources. Data saved on these disks consume huge amount of energy for processing. To address this issue a number of energy efficient techniques are proposed. Most of these techniques aggressively save energy without considering their effects on reliability of the disks. Moreover, once a disk fails it needs to be recovered and it can only be done if there exists a copy. To recover the data again a

significant amount of energy is spent, so a part of energy saved by using the energy efficient algorithms will again be spent for its recovery. Hence there is a need for techniques that not only consider energy efficiency but also maintains a good reliability. Our motivation to study the reliability issues in this research comes from the above defined problems.

1.1.4 Security Issues of Real Time Systems

Real time systems are very sensitive with respect to security. Security of these systems cannot be compromised at any cost. Once the security of the real time system such as online banking is compromised it is very hard to gain the trust of the consumer again. It puts a great deal of loss on the commercial or organizational benefits. When applications like military air craft control are considered, compromising the security of these applications puts the human life at stake. Hence improving the security of these systems not only benefits the world financially but only improves the trust worthiness of the applications, thereby providing peace of mind. Security of the real time applications can be attained by applying the wide variety of security mechanisms available today. A system can be made highly secure if all the existing security mechanisms can be applied to it, as it is not practically feasible the security if often achieved and improved by carefully selecting the important security mechanisms. This motivated the second part of our research to study the security mechanisms of real time systems.

1.1.5 Reliability Concerns of Real Time Systems

Reliability of a real time system needs very careful attention. A real time application such as aircraft control system cannot tolerate any faults when it is detonated in field. Because of the critical nature of the application itself these systems need to be one hundred percent fool proof. When an application such as online banking is considered, and if a fault occurs during a high level transaction then the total banking operations are left in jeopardy. The two examples discussed here outline the importance of reliable operations in real time systems. Our motivation to study the reliability of real time systems and to combine it with the security of real time systems comes from the above mentioned examples.

1.2 Scope of Research

Our research is focused on two goals. First one is to design energy-efficient and reliable techniques for large scale data centers. Existing energy conservation techniques such as operating the disks in different power modes are used in our research in an innovative way to guarantee the reliability. To reduce the impact of disk transitions on energy efficiency and reliability we also focused on writing an algorithm to control the power state transitions. Finally a very energy efficient and reliable approach was proposed in which popular data was skewed on to few disks to save energy. Safe utilization zone is a terminology we have come up with to define the utilization range in which the disk reliability is higher. By operating the disks in this zone high reliability is guaranteed.

For our second design goal we focused on the real time embedded systems issues such as security and fault tolerance. We addressed the issue of security by implementing confidentiality and integrity services in to the system and achieved fault tolerance by using uniform and non uniform check point strategies.

1.3 Contributions

The major contributions of this research are summarized as follows:

For large scale storage systems arena: (Design goals – Energy Efficiency and Reliability)

(1) We developed a mathematical reliability model to estimate disk failure rate as a function of disk utilization and ages. We derived that high disk reliability is achieved by operating disks only in the safe utilization zones.

(2) We provided a dynamic power management policy that aims to reduce energy dissipation in parallel I/O systems with mirroring disks.

(3) We proposed an algorithm to control the power state transitions which negatively affect reliability and energy efficiency if not controlled.

(4) We design an utilization based reliability aware energy efficient algorithm to achieve high thresholds of energy efficiency and reliability

(5) We conduct extensive experiments for large scale storage systems. These experimental results could be used by other researchers in their related research area, if applicable.

For real time and embedded systems: (Design goals – Security and Reliability)

(6) We presented the overhead models for the security services and for the fault recovery model using uniform and non uniform check points.

(7) We proposed two methods to combine the security with reliability in a real time task. Next, we integrated both the fault recovery mechanisms with adaptive quality of security

1.4 Dissertation Organization

This dissertation is organized as follows. In Chapter 2, related work in the literature is briefly reviewed.

In Chapter 3, we propose the energy efficiency and reliability of storage systems design. We proposed an algorithm to incorporate reliability and energy efficiency in to storage systems. To control the power state transitions of the algorithm proposed in Chapter 3, we develop a control algorithm for power state transitions in Chapter 4.

In Chapter 5, we study the utilization based reliability aware energy efficient techniques. Chapter 6 details the proposed approaches to integrate security and reliability in real time embedded systems.

Chapter 7 explains the interplay of fault recovery and quality of security, by providing two different strategies for security implementation and ways to integrate these strategies with fault tolerant approaches.

Finally, Chapter 8 summarizes the main contributions of this dissertation and comments on future directions for this research.

Chapter 2

Literature Review

In this chapter, we briefly summarize the previous literatures which are most relevant to our research in terms of energy-efficiency and reliability for large scale storage systems. Next we have summarized the work done previously on security and reliability of real time embedded systems. Section 2.1 will introduce related work on energy-efficient storage systems followed by the reliability work on storage systems in section 2.2. Section 2.3 and 2.4 illustrate the work in the area of security and reliability of real time embedded systems respectively.

2.1 Related Work on Energy-Efficiency of Disks

Extensive research has been carried out in developing energy efficient storage systems. Dynamic voltage scaling [14][33][47], dynamic power management[71], compiler directed energy optimizations [65][68] are some of the state of the art energy conservation techniques.

Du el al. studied the dynamic voltage scaling technique with a real-time garbage collection mechanism to reduce the energy dissipation of flash memory storage

systems [85]. A dynamic spin down technique for mobile computing was proposed by Helmbold et al. [15]. A mathematical model for each Dynamic Voltage Scaling - enabled system is built and their potential in energy reduction is analyzed by Lin Yuan and Gang Qu [47]. Carrera et al. [21] proposed four approaches to conserving disk energy in high-performance network servers and concluded that the fourth approach, which uses multiple disk speeds, is the one that can actually provide energy savings.

An energy saving policy named eRAID [13] for conventional disk based RAID-1 systems using redundancy is given by Li et al. Energy efficient disk layouts for RAID-1 systems [12] have been proposed by Lu et al. Yue et al. investigated the memory energy efficiency of high-end data servers used for supercomputers [34]. Son et al. proposed and evaluated a compiler-driven approach to reduce disk power consumption of array-based scientific applications executing on parallel architectures [65][68][33].

Dempsey, a disk simulation environment that includes accurate modeling of disk power consumption is presented by Zedlewski et al [35]. They also demonstrated that disk power consumption can be simulated both efficiently and accurately. Optimal power management policies for a laptop hard disk are obtained with a system model that can handle non-exponential inter-arrival times in the idle and the sleep states [71].

Gurumurthi et al. [58] provided a new approach called DRPM to modulate disk speed (RPM) dynamically, which gives a practical implementation to exploit this mechanism. They showed that DRPM can provide significant energy savings without heavily compromising performance. Rosti et al. presented a formal model of the behavior of CPU and I/O interactions in scientific applications, from which they

derived various formulas that characterize application performance [20]. D. Colarelli and D. Grunwald presented an architecture called “Massive Arrays of Idle Disks” or MAID [11]. In their work they did not consider the reliability issue again. Another framework similar to MAID, called Popular Data Concentration [18], was proposed by E. Pinheiro and R. Bianchini. The basic idea of PDC is to migrate data across disks according to frequency of access, or popularity. The goal is to lay data out in such a way that popular and unpopular data are stored on different disks. This layout leaves the disks that store unpopular data mostly idle, so that they can be transitioned to a low-power mode. However, PDC is a static offline algorithm. In some cases, it is impossible for the system to exactly know which data is popular and which is not. This is especially true for the ever-changing workload, in which some data is popular at a particular period but becomes unpopular the next period.

In contrast with both MAID and PDC, we implemented a utilization based reliability aware energy efficient algorithm to control energy consumption and to increase the reliability.

2.2 Related Work on Reliability of Disks

Schroeder and Gibson presented and analyzed the field-gathered disk replacement data from five systems in production use at three organizations [5]. They found evidence that failure rate is not constant with age, and that there was a significant infant mortality effect. The infant failures had a significant early onset of wear-out degradation. Significant levels of correlation between failures, including autocorrelation and long-range dependence, were also found.

Pinheiro et al [19] presented failure statistics and analyzed the correlation

between failures and several parameters generally believed to impact longevity [19]. Four causes of variability and an explanation on how each is responsible for a possible gap between expected and measured drive reliability, are elaborated by Elerath and Shah [32].

All of the previously mentioned work either concentrated on the power conservation or on the disk reliability. Not many researchers address both energy efficiency and reliability. It is very important for a data disk to be very reliable, while consuming less power. The importance of energy efficiency and reliability, and the lack of research of their relationship, motivates the research conducted in this dissertation.

2.3 Related Work on Reliability of Real Time Embedded Systems

It is known that, much attention has been paid towards fault-tolerant real-time scheduling. Most of conventional real time scheduling algorithms used timeline or backup approaches for providing fault recovery [63][1]. Such approaches with low overhead are adequate for real-time embedded systems [26][45][4][50] when storage is not a primary concern. During transient faults, re-executions of tasks are usually performed to recover faults [24][84].

Sultan *et al.* developed management algorithms that are efficient in bounding checkpoints and logs [24]. A protocol that supports both security and reliability aspects in database systems are presented in [44][4]

A variety of fault-tolerant [57][39][24][63] activities are incorporated in both static and dynamic scheduling without impairing the feasibility of pre-guaranteed

tasks and minimizing the number of reliability activities [16]. In addition, a method, slot shifting was used to integrate static and dynamic scheduling. Uniform and non-uniform check-pointing policies were developed to recover from failures and reduce power consumption in real-time embedded systems [57][55][88]. We leverage a similar approach to support fault recovery. However, our work is fundamentally different from the previous studies in the sense that, ours addresses the issue of improving security [72][16] without adversely affecting reliability, whereas, the previous studies were focused on improving reliability of real-time embedded systems.

2.4 Related Work on Security of Real Time Embedded Systems

Song *et al.* developed security driven scheduling algorithms for Grids [57][66]. Tao *et al.* designed and implemented security-aware real-time scheduling algorithms, which make use of three types of security services to guard real-time embedded systems from potential threats [72]. They investigated the integration of security services to protect against a diversity of threats and attacks in cluster computing environment. A very interesting security application where life and death relevance have become the security challenges was addressed [84].

Bertram *et al.* proposed a set of algorithms for security-constrained optimal power flow (SCOPF) and their configuration in an integrated package for real-time security enhancements [6]. An introduction of non-convex SCOPF methods for modeling utility operating policy is also given. This study relies on a security overhead model that is presented in [72], which depicts the overhead model for an array of security

services. Protocols for replication and voting in a family of applications are investigated in [86][27]. Myers et al proposed a method of building a trustworthy distributed system by the process of Construction. The literature proves that very limited or no research has been done in integrating the fault tolerance and security in real-time systems. To bridge this gap we conducted research to integrate security and reliability in real time embedded systems.

2.5 Summary

There are two objectives in this dissertation research. The first objective is to develop energy-aware and reliable storage disk systems. The second one is to integrate security and reliability mechanisms in real-time embedded systems.. This chapter outlines a variety of existing techniques related to (1) energy efficiency of storage disks and their reliability issues and (2) reliability and security issues in real-time and embedded systems.

Chapter 3

Energy Efficiency and Reliability of Storage Disks

Numerous energy saving techniques have been developed to aggressively reduce energy dissipation in parallel disks. However, many existing energy conservation schemes have substantial adverse impacts on the reliability of disks. To remedy this deficiency, we address the problem of making tradeoffs between energy efficiency and reliability in parallel disk systems with data mirroring. Among several factors affecting disk reliability, the most two important factors – disk utilization and ages – are the focus of this study. In this chapter, we build a mathematical reliability model to quantify the impacts of disk age and utilization on failure probabilities of mirrored disk systems. In light of the reliability model, we propose a novel concept of safe utilization zone, within which energy dissipation in disks can be reduced without degrading reliability. We developed an approach to improving both reliability and energy efficiency of disk systems through disk mirroring and utilization control that enforces disk drives to be operated in safe utilization zones. This is the first

utilization-based control scheme that seamlessly integrates reliability with energy saving techniques in the context of fault-tolerant systems. Experimental results show that our approach can significantly improve reliability while achieving high energy efficiency for disk systems under a wide range of workload situations.

This chapter is organized as follows. In section 3.1, we introduced the system model that we considered in this research. In section 3.2, we presented the disk failure model. Next, we provided the reliability aware energy conservation model in section 3.3 along with the Reliability Aware and Energy Efficient algorithm (RAREE). Analysing the reliability of the mirrored disk with our approach is presented in section 3.4. Experimental environment and simulation results are shown in section 3.5. Finally, section 3.6 concludes this chapter by summarizing the main contributions of the chapter.

3.1 System Model

Energy consumption can be reduced by dynamically operating parallel disks at three power states – active, idle and sleep. Mirroring disks (a.k.a., RAID1) – uses a minimum of two disks; one primary and one backup – are widely adopted to provide fault tolerance. Therefore, in this research we devote attention to parallel I/O systems with disk mirroring. Traditional energy conservation schemes have concentrated on reducing energy dissipation by waking up backup disks only when the utilization of primary disks exceeds a certain threshold. However, these techniques do not address the issues of reliability. Load balancing techniques keeps both primary and backup disks equally busy to optimize I/O performance. Unfortunately, existing load balancing methods are not energy efficient. To remedy the above deficiencies, we aim

to develop a reliability-aware power management policy, in which disks were operated at different power modes in accordance with disk utilization, which can lead to reduced disk failure rates.

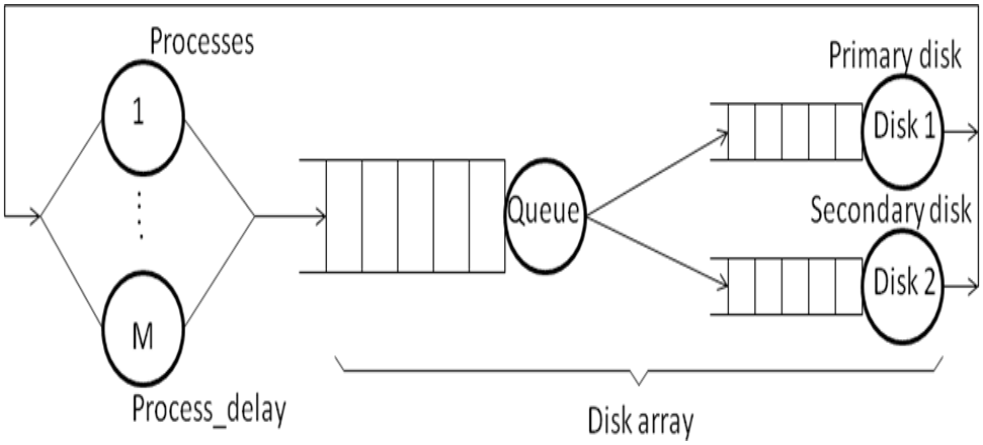

Figure 3.1. System model of parallel I/O systems with mirroring disks.

Fig. 3.1 depicts the system model of parallel I/O systems with mirroring disks. The processor in Fig. 3.1 generates read requests to be processed by the mirroring disks. These requests are queued up; the utilization levels are calculated accordingly. The utilization levels are checked against the limits of the safe utilization zone, which is specified in light of our novel reliability model. If the calculated utilization level falls below the safe utilization zone, then both the disks stay in the sleep state. If the utilization is within the limits of the safe zone, then the primary disk stays active whereas the backup disk sleeps. When the utilization exceeds the safe utilization zone, then both disks are kept in the active state to achieve high performance through load sharing. Empirical results show our approach not only can improve reliability of RAID 1, but also saves a significant amount of energy.

3.2 Disk Failure Model

When one disk in a disk I/O system fails, there is a strong likelihood that failure occurs in other disks of the system. To quantitatively investigate impacts of energy conservation techniques on reliabilities of disk systems, we focused on effects of disk utilization and age on the reliability of disks. Utilization is of paramount importance for disk reliability; because it is a critical bridge between energy saving and reliability in disk systems. More specifically, high disk utilizations generated by high workload conditions give rise to large disk energy consumption. Meanwhile, different disk utilizations lead to different levels of disk annual failure rates, depending on disk ages.

Since the utilization of a disk system is a centerpiece in the development of the disk failure model, we propose a disk utilization model before proceeding to develop the disk failure model. The salient feature of our utilization model is that it captures the characteristics and behavior of data-intensive tasks issuing disk request to a disk system. A second unique aspect of the utilization model is that it can be readily applied to storage systems at the application level. Our utilization model is general in the sense that the model can deal with mixed workloads where there are a set of data-intensive tasks with a variety of disk I/O requirements.

Let $Z+$ be the set of positive integers. Without loss of generality, we consider a workload condition where there are $m \in Z+$ disk I/O phases. The utilization U_i of the ith ($1 \le i \le m$) I/O phase is a constant that can be straightforwardly derived from the disk I/O requirements of data-intensive tasks running within the ith I/O phase. Let ϕ_i be the number of data-intensive tasks running in the ith phase. Let λ_{ij}, $1 \le j \le \phi_i$,

denote the arrival rate of disk request submitted by the *j*th data-intensive task to the disk system. Let s_{ij}, $1 \le j \le \phi_i$ be the average data size of disk requests of the *j*th task. The disk I/O requirement of the *j*th task in the *i*th phase is a product of the task's request arrival rate and the average data size of disk requests issued by the task, i.e., $\lambda_{ij} \cdot s_{ij}$. The accumulative disk I/O requirements R_i, measured in terms of MByte/Sec., of all the tasks running in the *i*th phase can be written as:

$$R_i = \sum_{j=1}^{\phi_i} (\lambda_{ij} \cdot s_{ij}) \quad . \tag{3.1}$$

Note that R_i in Eq. (3.1) can be environed as accumulative data amount access per time unit.

The utilization of a disk system within a given I/O phase equals to the ratio of the accumulative disk requirement R_i and the bandwidth of the disk system. Thus, the utilization μ_i of the *i*th I/O phase can be expressed as

$$\mu_i = \frac{R_i}{B_{disk}} = \frac{\sum_{j=1}^{\phi_i} (\lambda_{ij} \cdot s_{ij})}{B_{disk}} \quad , \tag{3.2}$$

where B_{disk} is the bandwidth of the disk system.

The utilization *U* of the disk system during the *m* I/O phases is the weighted sum of the disk utilization of all the *m* phases. Thus, the utilization *U* is expressed by Eq. (3.3) as follows:

$$\mu = \frac{R_i}{\sum_{i=1}^{m} R_i} \cdot \mu_i = \frac{\left[\sum_{j=1}^{\phi_i} (\lambda_{ij} \cdot s_{ij})\right]^2}{B_{disk} \cdot \sum_{i=1}^{m} \sum_{j=1}^{\phi_i} (\lambda_{ij} \cdot s_{ij})}. \quad (3.3)$$

Given I/O requirements of data-intensive tasks issuing disk request to a disk system, one can leverage the above model to quantify utilization of the disk system. The notation for the utilization model is summarizes in Table 3.1.

Table 3.1 List of parameters for the disk utilization model.

m	total number of I/O phases
i	the ith I/O phase
ϕ_i	total number of I/O-intensive tasks in phase i
j	the jth task
λ_{ij}	arrival rate of disk requests of task i in phase j
s_{ij}	data size of disk requests of task i in phase j
R_i	accumulative disk requirement of phase i
U_i	utilization of phase i
U	disk utilization

It is worth noting that the reliability of the disk system can be measured in terms of disk annual failure rates. Using previously published data [19], we build a disk failure model in the format of annual failure rate percentiles as functions of both disk ages and disk utilization levels. We obtained annual failure rate percentiles for the low (i.e., 25%), medium (i.e., 50%), and high utilization (i.e., 90%) levels for disks of ages from 3 month to 5 years. These values are used to calculate the annual failure rates percentiles for the other utilization levels of the disks aged from 3 month to 5

years.

To determine the failure rate for a given utilization rate, we took the points from the Google study [19] and used the cubic spline interpolation method to approximate annual failure rate of a disk with certain utilization and age. The disk failure model can be modeled as an n+1 dimensional vector $\theta' = [\theta_0, \theta_2, ..., \theta_n]$, where θ_i, $0 \le i \le n$, is the vector $\theta_i = (\mu_i, f_i)$ that captures the correlations between utilization μ_i and disk failure rate f_i.

To develop the disk failure rate model, we have to determine n+1dimensional vector θ'. Thus, given the value of utilization x_i, one can make use of the failure rate model to calculate the failure rate f_i in component θ_i of vector θ'. To achieve this goal, we adopted the cubic spline interpolation to generate failure rates n+1dimensional vector θ' such that it results in a smooth curve for a given disk age. We plot the annual failure rate percentiles for a disk of age 6 months (see Fig. 3.2.), 2 years (see Fig. 3.3.) and 3 years (see Fig. 3.4.) and as functions of disk utilizations.

The failure rate plotted in Figure 3.2 can be modeled using the following function. Similar functions are used to generate the annual failure rate graphs for 1 year and 3 year old disks, depicted in Figs. 3.3 and 3.4.

$$\phi(\alpha = 0.5, \mu) = \begin{cases} 0.06\mu - 0.000064\mu^3, 0 \le \mu \le 0.25 \\ 0.8 - 0.02914\mu + 0.001029\mu^2 - 0.000013\mu^3, 0.25 \le \mu \le 0.5 \\ -0.868 - 0.102\mu + 0.0037\mu^2 - 0.000024\mu^3, 0.5 \le \mu \le 0.75 \\ 0.8 + 0.038\mu - 0.00048\mu^2 + 0.000021\mu^3, 0.75 \le \mu \le 1 \end{cases}$$

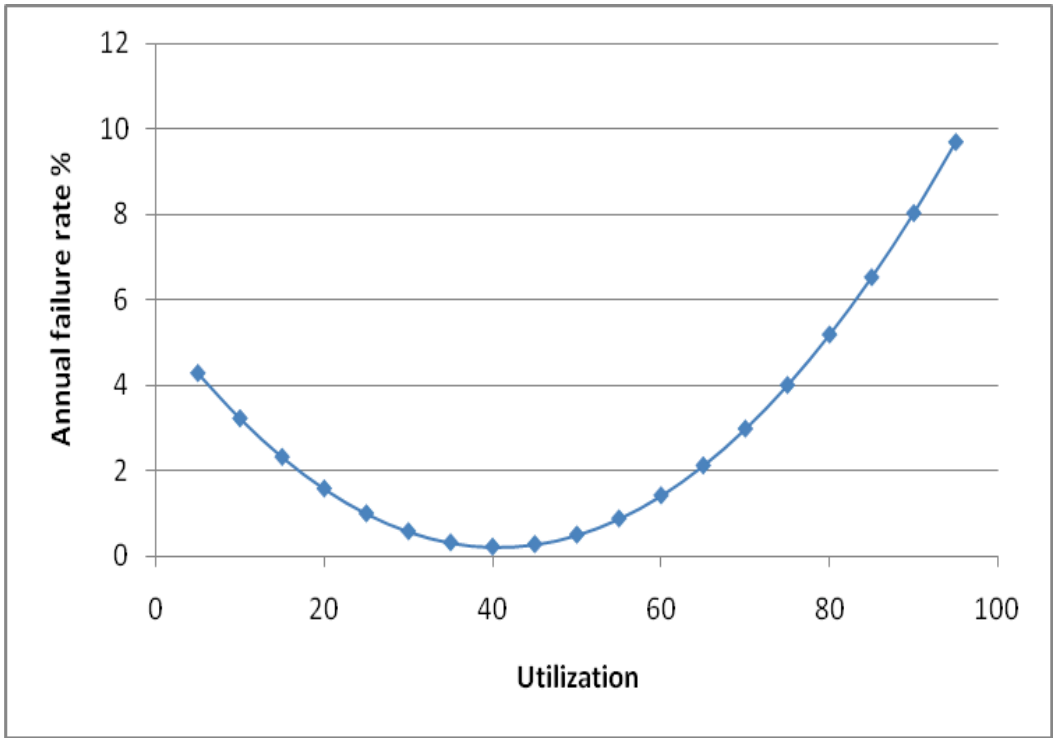

Figure 3.2. Annual failure rate for 6-month old disks with respect to utilization.

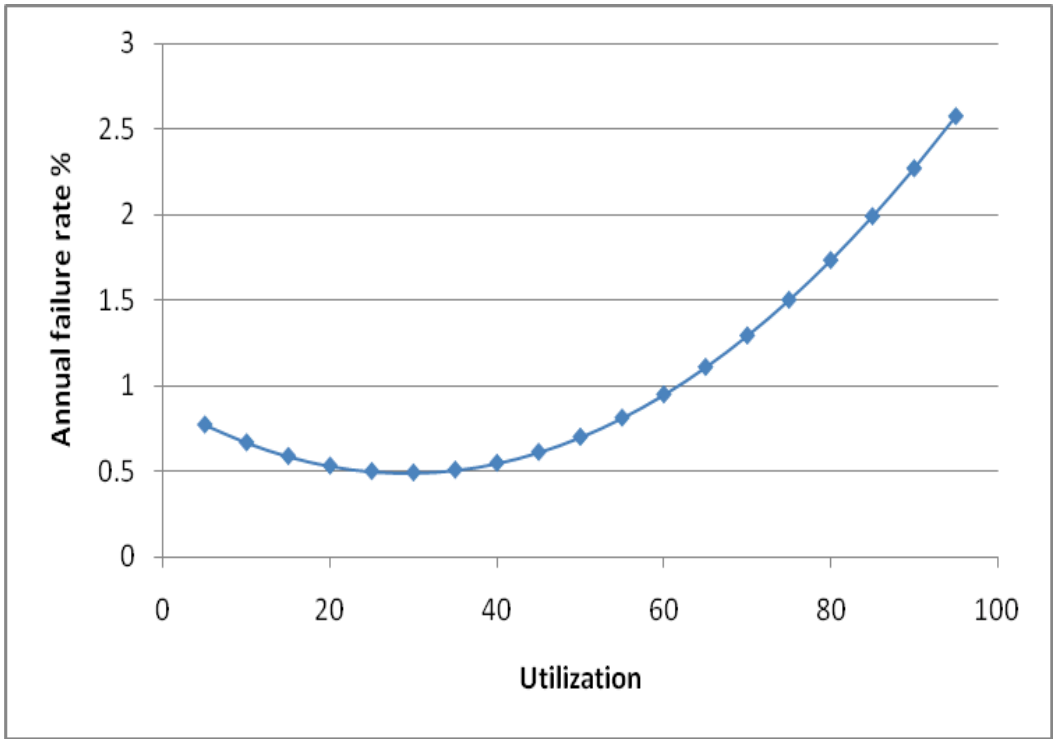

Figure 3.3. Annual failure rate for 1-year old disks with respect to utilization.

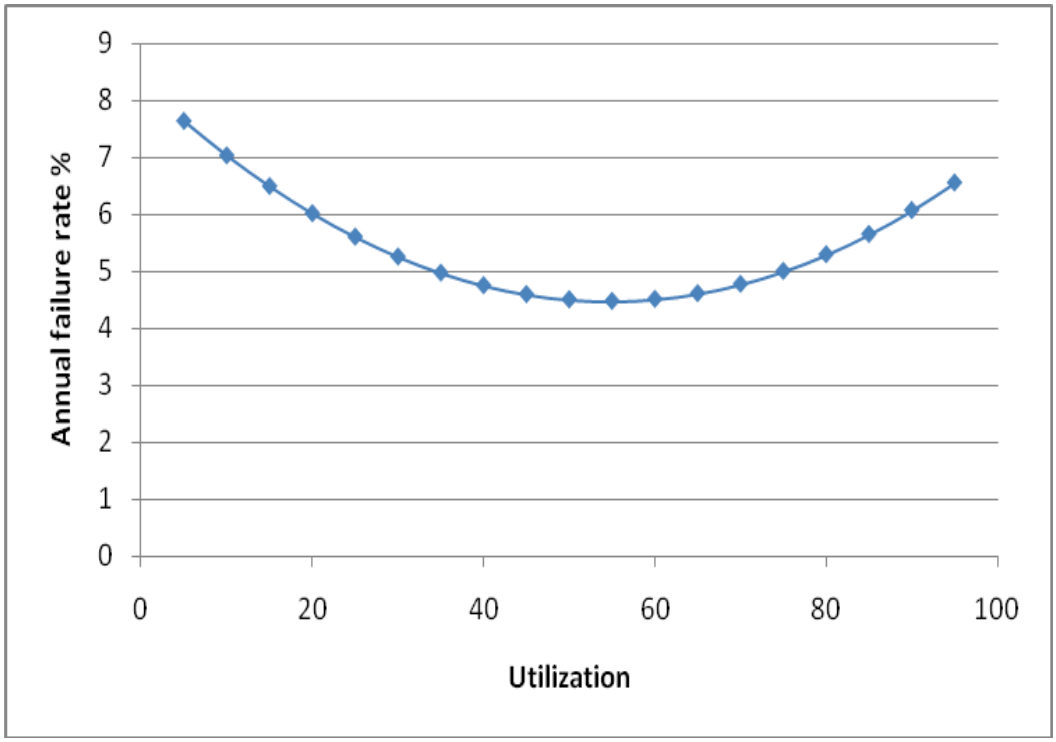

Figure 3.4. Annual failure rate for 3-year old disks with respect to utilization.

Interestingly, results shown in Figs. 3.2, 3.3 and 3.4 contradict the findings of the previous studies that indicate that lower utilization levels produce lower failure rates whereas higher utilization levels correspond to higher failure rates. The trend of a disk annual failure rate as the disk utilization grows is different from that reported in the literature. Specifically, our disk failure rate model built from real-world data of disk failures suggests that the probability that a disk fails under either very low or very high utilization levels is very high. In contrast, when the utilization levels stay within a certain range (e.g., between 20% and 60% when the disk age is 6 months old, see Fig. 3.2), the probability of failures is far smaller than a specified threshold (e.g., smaller than 2%). We term this range of utilization levels as the safe utilization zone, which largely depends on disk ages. Given a disk system, its safe utilization zone is a function of the disk's age and the specified low failure rate threshold. The definition

of safe utilization zone is formally given below.

Definition: Given a disk system, we denote the disk's utilization, age and the threshold for low failure rates as μ, α and τ, respectively. Recall that the failure rate of the disk is a function of utilization μ and age α, i.e., $f(\mu, \alpha)$. A utilization range [μ_{min}, μ_{max}] is called a safe utilization zone with respect to disk age α and failure rate threshold τ if the following condition is held:

$$\forall \mu_{\min} \le \mu \le \mu_{\max} : f(\mu, \alpha) \le \tau. \tag{3.4}$$

The above condition indicates that the disk failure rate of utilization levels within a safe utilization zone is always smaller than or equal to the low failure rate threshold. We formally describe this property as below:

Property: Given a disk with age α and low failure rate threshold, τ, one can ensure that a utilization level within its safe utilization zone results in a failure rate lower than the failure rate threshold, τ.

With the concept of safe utilization zone in place, it is efficient to dynamically control disk utilization in a way to keep a disk's failure rate lower than a specified threshold. The above statements are true for the results depicted in Figs. 3.2 through 3.4.

3.3 Reliability-Aware Energy Conservation Model

RAID 1 is popular and is widely used for disk drives. RAID 1 is implemented with a minimum of two disks, which are the primary and back disks. Initially the data is stored to the primary disk and then it is mirrored to the backup disk. This mirroring helps to recover the data when there is a failure in the primary disk. It also helps to

increase the performance of the RAID 1 system by sharing the workload between the disks. We considered RAID 1 for all of our experiments.

The processor in the system generates the I/O stream, which is queued to the buffer. The utilization of the disk is calculated using the request arrival rate. Please refer to Section 3.2 for details of the description for the disk utilization model.

It should be noted that all requests here are considered as read requests. At any given point of time the disks can be in the following three states.

- State 1: Both the disks in sleep mode
- State 2: Primary disk in active and backup disk in sleep mode
- State 3: Both the disks in active and share the load.

Let us consider that the disks are in state 1 at the beginning. Once the utilization is calculated, it is compared with the safe utilization zone range. If the calculated value falls below the range then disks stay in state 1. If the calculated value is within the range, then the primary disk is made active while the backup disk continues to stay in the sleep mode. This represents a transition to state 2. If the calculated value is beyond the range then both the disks are made active and both of them share the load, which corresponds to state 3. Transition of states from one power mode to another involves disk spin up and/or spin down. The disk spin ups and spin downs also consume a lot of energy.

The state transition diagram (see Fig. 3.5) gives a detailed explanation of the state transitions at different utilizations and idle times. It can be observed from the diagram that there are 5 possible state transitions:

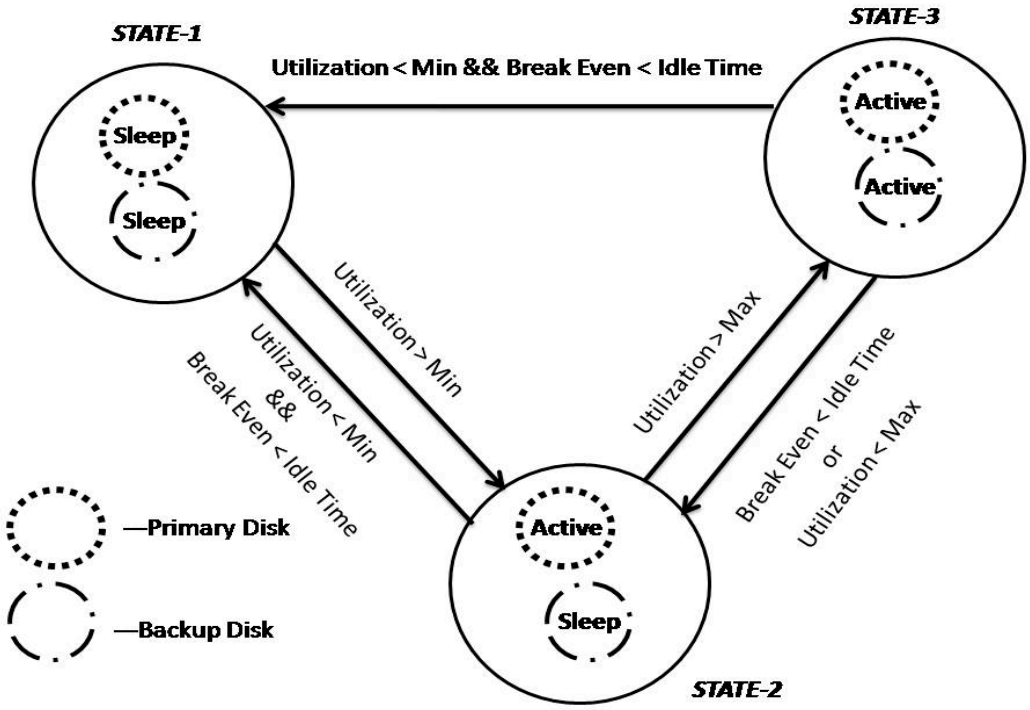

Figure 3.5 State transition diagram.

1. State 1 to state 2:- If the calculated utilization falls within the safe utilization zone then the system transitions from state 1 to state 2.
2. State 2 to state 3:- When the utilization exceeds the safe zone, the system transitions from state 2 to state 3.
3. State 3 to state 1:- When the utilization becomes zero or when the utilization falls below the safe zone or when the idle time is higher than the break even time the system transitions from state 3 to 1.
4. State 2 to state1:- When the utilization becomes zero or when the utilization falls below the safe zone or when the idle time is higher than the break even time the system transitions from state 2 to 1.
5. State 3 to state 2:- When the utilization falls back into the safe zone range or when the idle time is higher than the break even time.

Input: *M*: Number of processors in the system

λ: Poisson arrival rate of requests generated by each of the M processors

Ω: Time interval between arrivals follows exponential distribution

S_u: Spin up power

S_d: Spin down power

P_a: Active power

P_i: Idle power

P_a: Sleep power

α: Break even time (Spin up + spin down power)

Q: Buffer

1. Insert λ into *Q* buffer
2. Calculate the utilization μ of the disk based on the number of requests in the buffer
3. Calculate the safe utilization zones (min, max) for the disks of different age groups
4. Compare the μ value against the min and max values
5. **If** the system is in state 1 **then**
   - **if** μ <min **then** no state transition occurs
   - **if** μ >min **then** system transits from state 1 to state 2
6. **If** the system is in state 2 **then**
   - **if** μ <min && α < Ω **then** system transits from state 2 to state 1
   - **if** μ >min && μ <max **then** no state transition occurs
   - **if** μ >max **then** system transits from state 2 to state 3
7. **If** the system is in state 3 **then**
   - **if** μ <min && α < Ω **then** system transits from state 3 to state 1
   - **if** μ >min && μ <max or α < Ω **then** system transits from state 3 to state 2
   - **if** μ >max **then** no state transition occurs

Figure 3.6 Reliability aware energy efficient (RAREE) algorithm for disk drives

Table 3.2: Main characteristics of two SCSI disks and an IDE laptop disk

Parameter	IBM 36Z15 Ultrastar(high perf)	IBM 73LZX Ultrastar(low perf)	IBM 40GNX Travestar (laptop)
Standard interface	SCSI	SCSI	IDE
Capacity	18 GBytes	18 GBytes	20 GBytes
Number of platters	4	2	2
Rotations per minute	15000	10000	5400
Disk controller cache	4 Mbytes	4 Mbytes	8 Mbytes
Average seek time	3.4 msec	4.9 msec	12 msec
Average rotation time	2 msec	3 msec	5.5 msec
Internal transfer rate	55 MB/sec	53 MB/sec	25 MB/sec
Power(active)	13.5 W	9.5 W	3.0 W
Power(idle)	10.2 W	6.0 W	0.82 W
Power(standby)	2.5 W	1.4 W	0.25 W
Energy(spin down)	13.0 J	10.0 J	0.4 J
Time(spin down)	1.5 sec	1.7 sec	0.5 sec
Energy(spin up)	135.0 J	97.9 J	8.7 J
Time(spin up)	10.9 sec	10.1 sec	3.5 sec

3.4 Analysis

In this section, we analyze the reliability of a mirrored disk system with our approach. In what follows, we term our approach as RAREE (Reliability aware energy efficient approach).

We aim to derive failure rate $p(\alpha, \mu)$ of a parallel I/O system with disk mirroring. The failure rate largely depends on the age α and utilization μ of a pair of primary and backup disks. Let $\alpha = (\alpha_P, \alpha_B)$ represent the ages of the primary and backup disks. Note that subscripts P and B represent primary and backup disks, respectively. We denote $p_P(\alpha_P)$ and $p_P(\alpha_B)$ as the failure rate of the primary and backup disks. The pair of disks is considered failed if both disks have failed. Given that disk failures are independent, we compute reliability $r(\alpha)$ of the parallel I/O system as

$$r(\alpha) = 1 - p_P(\alpha_P) \cdot p_B(\alpha_B) \quad (3.5)$$

Let $q_P(\mu)$ denote the probability that the utilization of the primary disk is μ; let $q_B(\mu)$ be the probability that the utilization of the backup disk is μ. $f(\alpha, \mu)$ represents the failure rate of an α-year old disk with utilization μ. Failure rate $p_P(\alpha_P)$ in Eq. (3.5) can be expressed as

$$p_P(\alpha) = \sum_{\mu} (q_P(\mu) \cdot f(\alpha, \mu)) \quad (3.6)$$

Similarly, failure rate $p_B(\alpha_B)$ in Eq. (3.5) is given by

$$p_B(\alpha) = \sum_{\mu} (q_B(\mu) \cdot f(\alpha, \mu)) \quad (3.7)$$

Now we analyse energy efficiency of our approach. Let $P_{P,A}$ and $P_{B,A}$ denote the power of the primary and backup disks when they are in the active state. Given a list $R = (R_1, R_2, ..., R_n)$ disk requests, we can compute energy E_A consumed by serving all the requests as

$$E_A = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} \left(x_{i,P} \cdot P_{A,P} \cdot t_{i,P} + \overline{x}_{i,P} \cdot P_{A,B} \cdot t_{i,B} \right) \quad (3.8)$$

where element $x_{i,P}$ is "1" if request i is responded by the primary disk and is "0", otherwise. $t_{j,P}$ and $t_{j,B}$ are service times. We obtain the analytical formula for the energy consumed when disks are in the sleep state:

$$E_S = P_{P,S} \cdot T_{P,S} + P_{B,S} \cdot T_{B,S}, \quad (3.9)$$

where $P_{P,A}$ and $P_{B,A}$ are the power of the primary and backup disks when they are is in the sleep state. $T_{P,S}$ and $T_{B,S}$ are the time intervals when the disks are in the sleep state. Let f_i be the completion time of request i. Then, $T_{P,S}$ and $T_{B,S}$ can be derived from I/O

processing times and completion time of the last request served by each disk. Thus, we have

$$T_{P,S} = \max_{i=1}^{n}\left(x_{i,P} \cdot f_i\right) - \sum_{i=1}^{n}\left(x_{i,P} \cdot t_{i,P}\right), \tag{3.10}$$

$$T_{B,S} = \max_{i=1}^{n}\left(\bar{x}_{i,P} \cdot f_i\right) - \sum_{i=1}^{n}\left(\bar{x}_{i,P} \cdot t_{i,B}\right) \tag{3.11}$$

The total energy consumption E of the parallel I/O system with disk mirroring can be derived from Eqs. (3.8) and (3.9) as

$$E = E_A + E_S. \tag{3.12}$$

3.5 Performance Evaluation

We conducted extensive experiments on three types of IBM disks: Ultrastar 36Z15, Ultrastar 73LZX, and Travelstar 40GNX. We compared our approach called RAREE with the load balancing scheme and the traditional dynamic power management scheme.

Read requests are generated based on Poisson process. The experimental results plotted in Figs. 3.7-3.9 show the energy consumed by parallel I/O systems with the three different types of IBM disks. Fig. 3.7 clearly shows that when it comes to the IBM 40GNX disk, our approach significantly improves energy efficiency over the traditional energy saving and load balancing scheme by up to 22.4% and 31.3%.

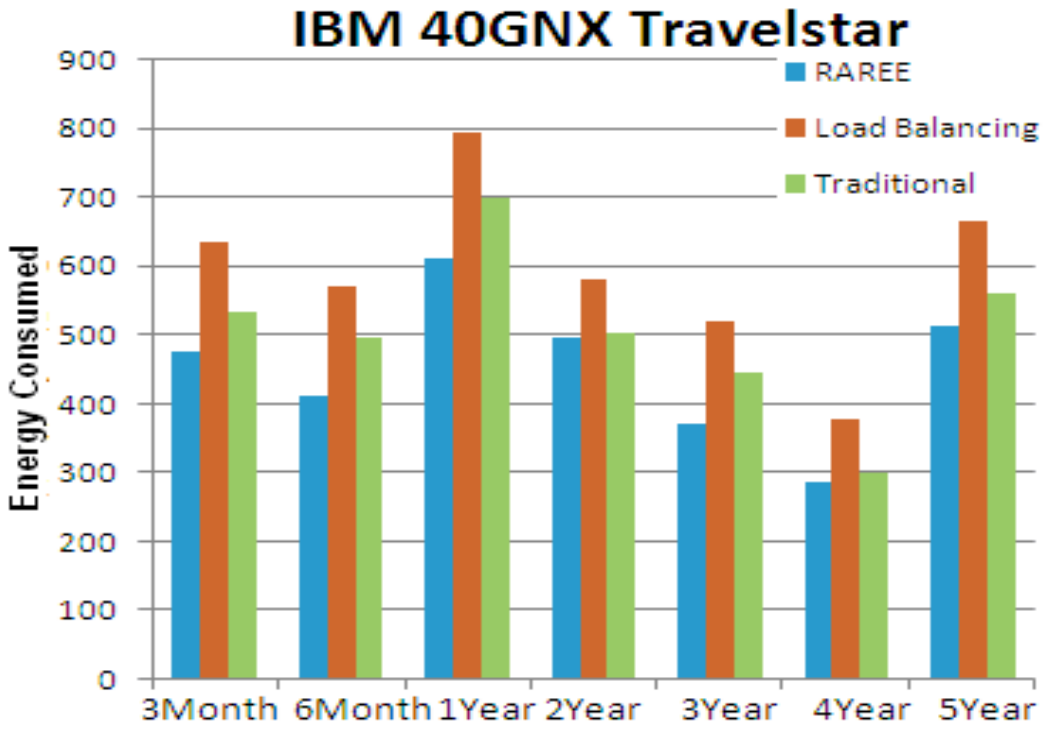

Figure 3.7 Energy dissipation (Joules). IBM 40GNX.

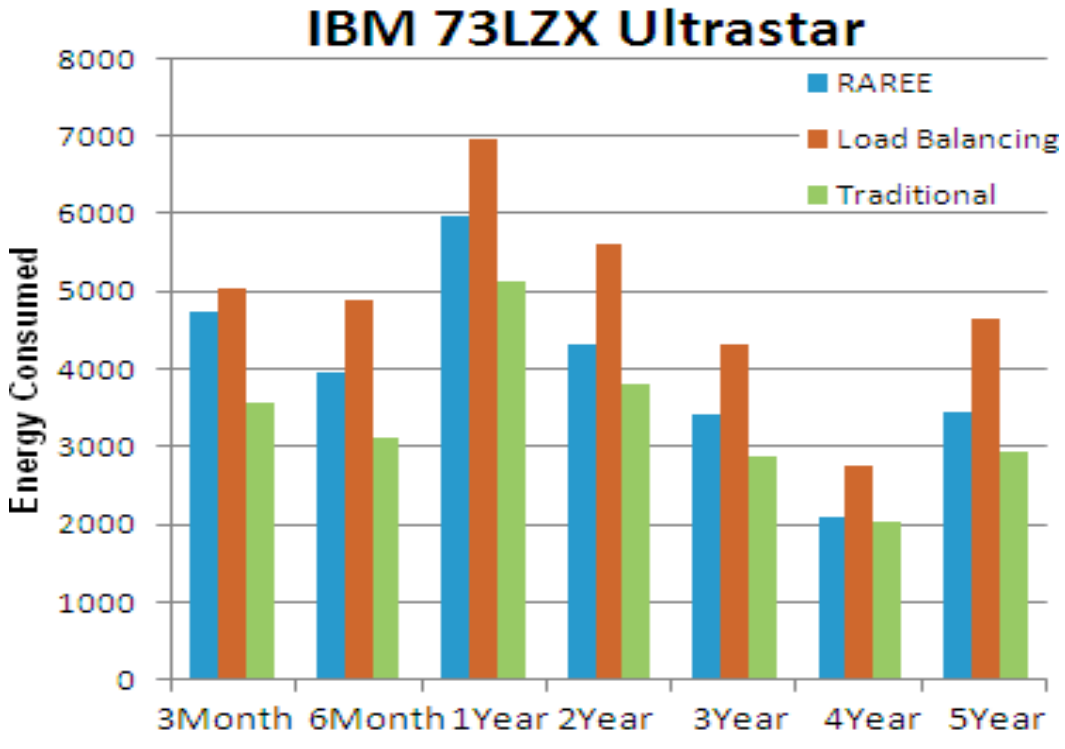

Figure 3.8 Energy dissipation (Joules). IBM 73LZX.

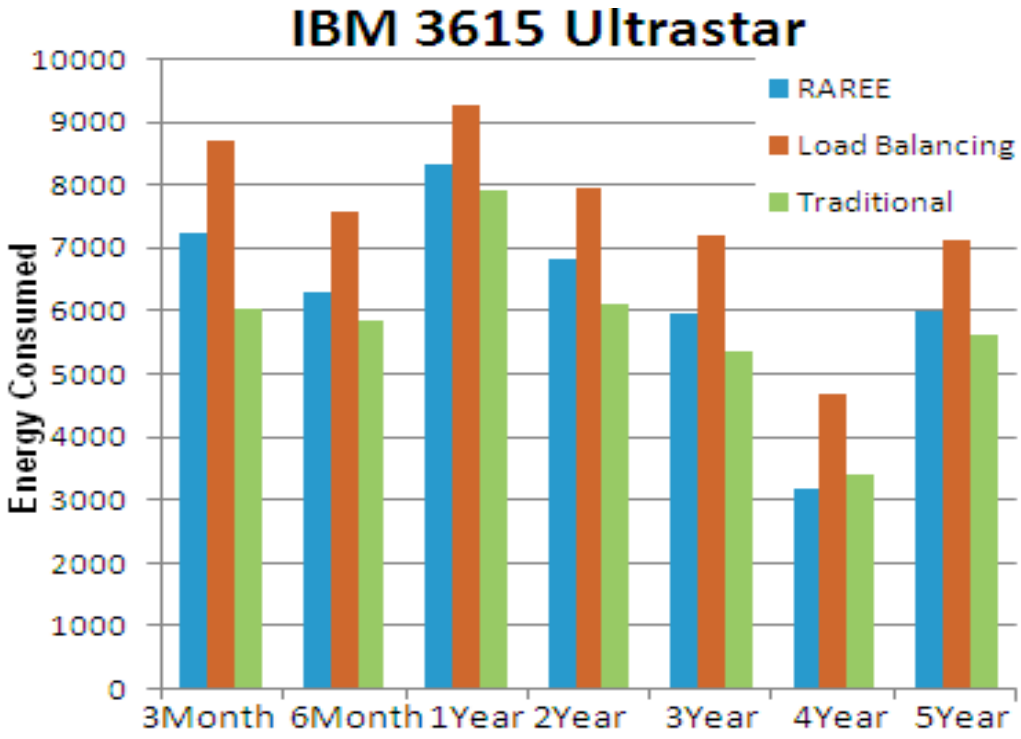

Figure 3.9 Energy dissipation (Joules). IBM 36Z15.

Interestingly, we observe from Figs. 3.8 and 3.9 that in case of the IBM 36Z15 and IBM 73LZX disks, RAREE energy efficiency is in between those of the load balancing and traditional DPM techniques. The result can be partially explained by the low disk spin up and spin down power of IBM 40GNX. This result indicates that disk spin down and disk spin up power can play a vital role in energy dissipation of parallel I/O systems. An intriguing conclusion drawn from this set of experiments is that our scheme is very energy-efficient for mobile disks as opposed to high-performance disks.

Figs. 3.10 and 3.11 show the impact of spin-down energy on the energy dissipation of the parallel disk system when the disks are 6-month and 1-year old, respectively. Though the energy efficiency of our approach is slightly lower than that

of the traditional DPM technique, ours substantially improves the reliability of parallel disk systems over the DPM and load-balancing schemes.

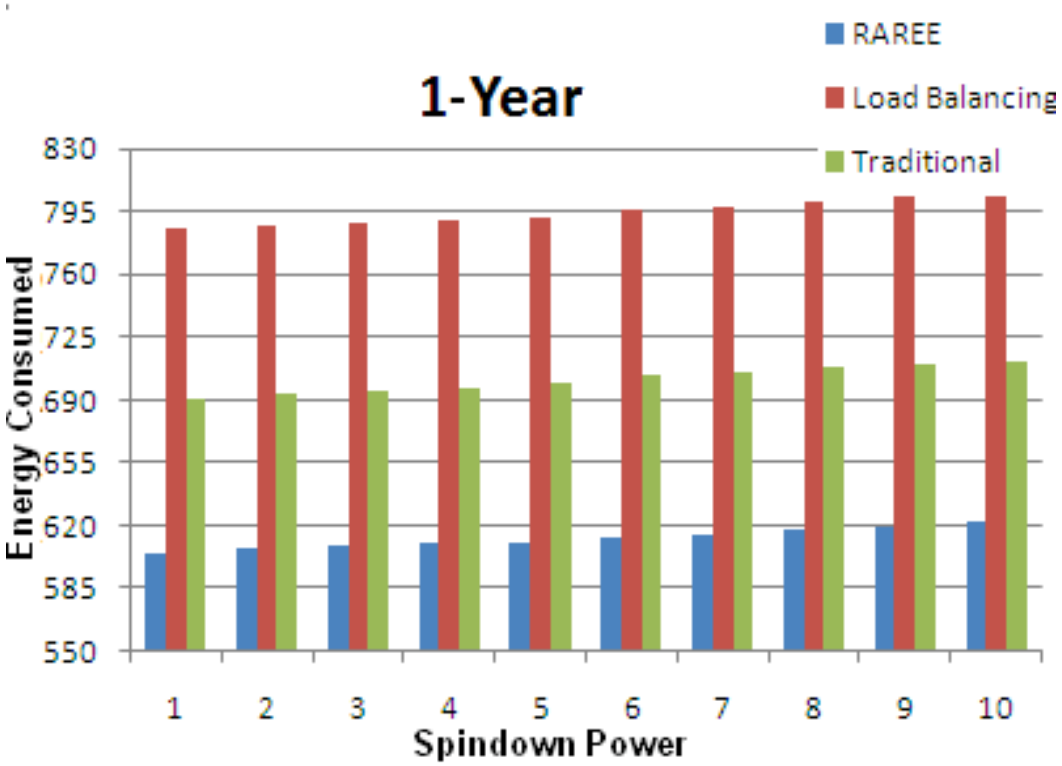

Figure 3.10 Spin-Down Energy (Joules) vs. Energy Dissipation (Joules).

Figs. 3.12 and 3.13 show the effect of spin-up energy on the energy dissipation in the parallel disk system with 6-month-old and 1-year-old ultrastar disks. In all the cases, our strategy provides noticeable energy savings compared with the other two schemes. In addition, Figs. 3.12 and 3.13 illustrate that for the three strategies, energy consumption slightly increases with the increasing value of spin-up energy. RAREE is more sensitive to spin-up energy than the alternatives do.

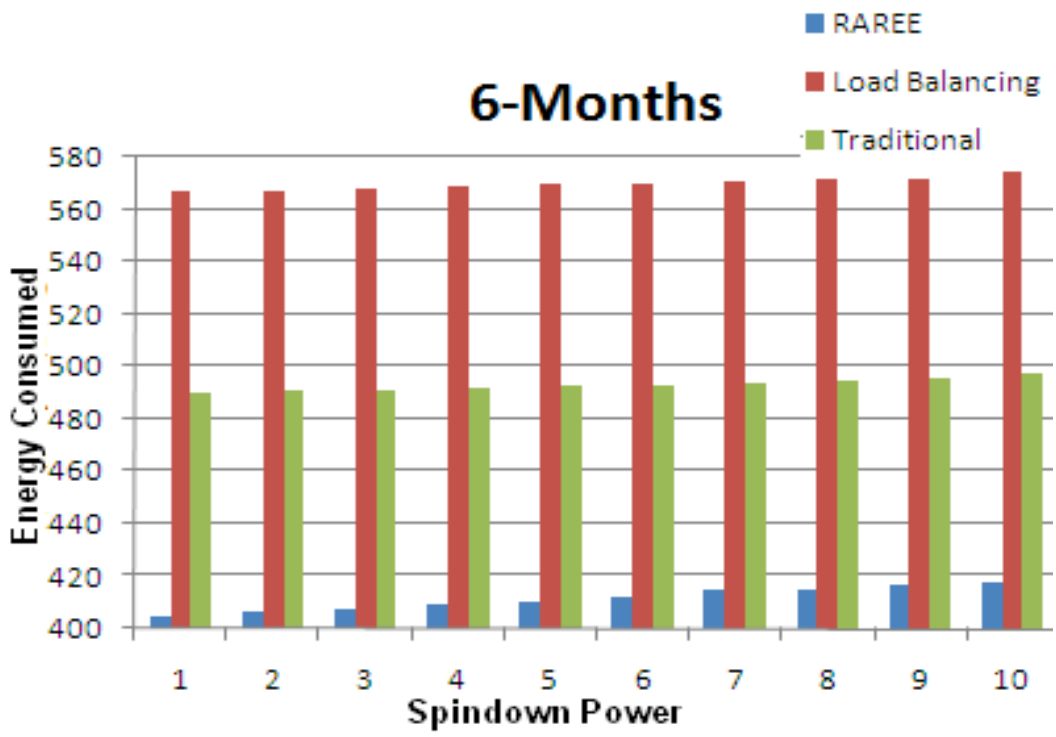

Figure 3.11. Spin-Down Energy (Joules) vs. Energy Dissipation (Joules).

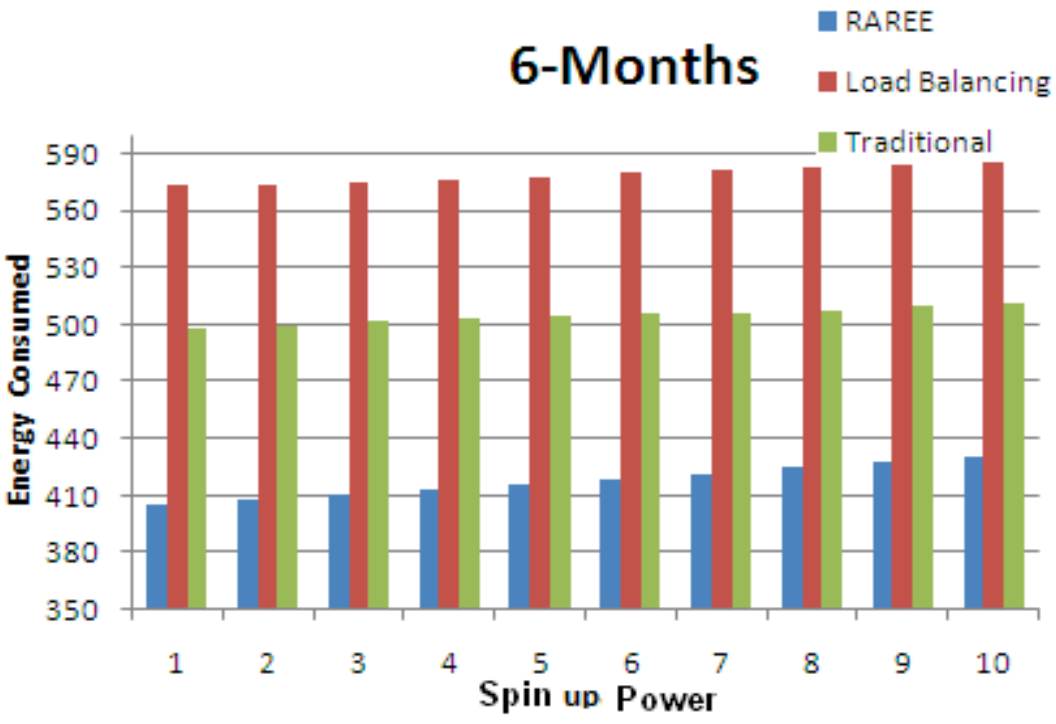

Figure 3.12. Spin-Up Energy (Joules) vs. Energy Dissipation (Joules).

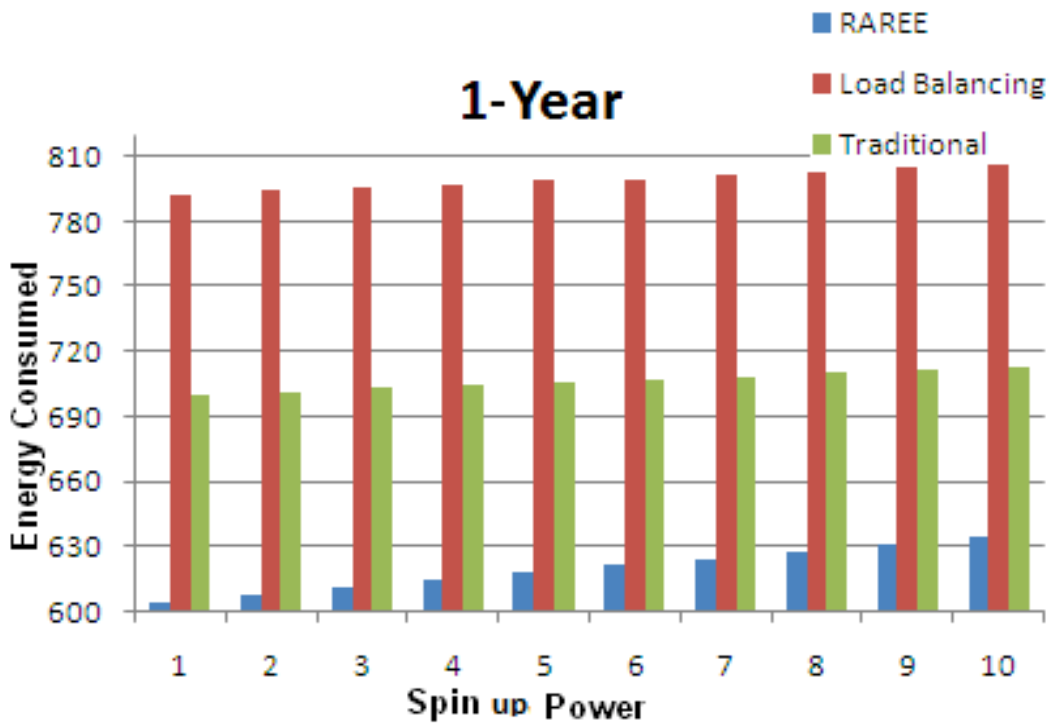

Figure 3.13. Spin-Up Energy (Joules) vs. Energy Dissipation (Joules).

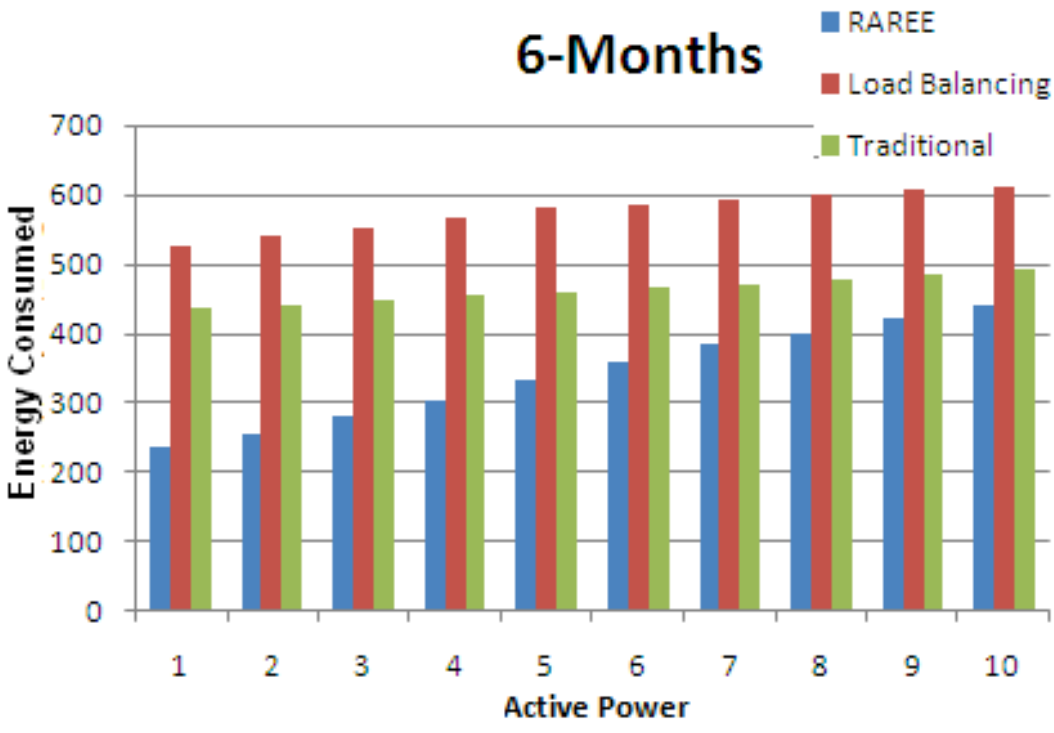

Figure 3.14. Active Power (Watts) vs. Energy Dissipation (Joules).

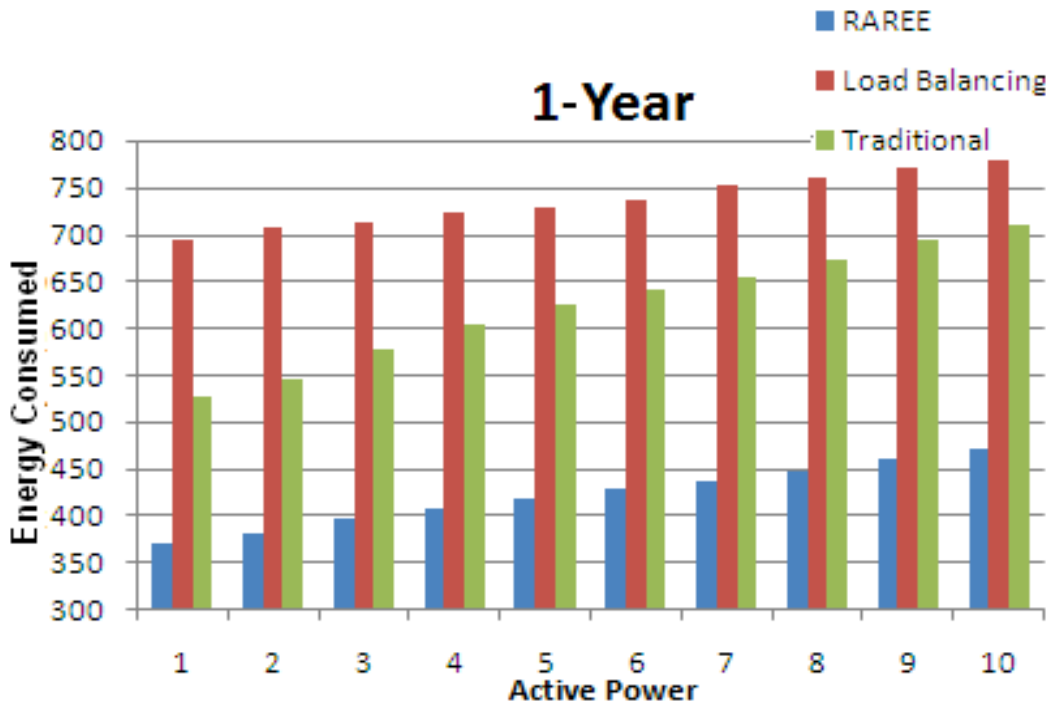

Figure 3.15. Active Power (Watts) vs. Energy Dissipation (Joules).

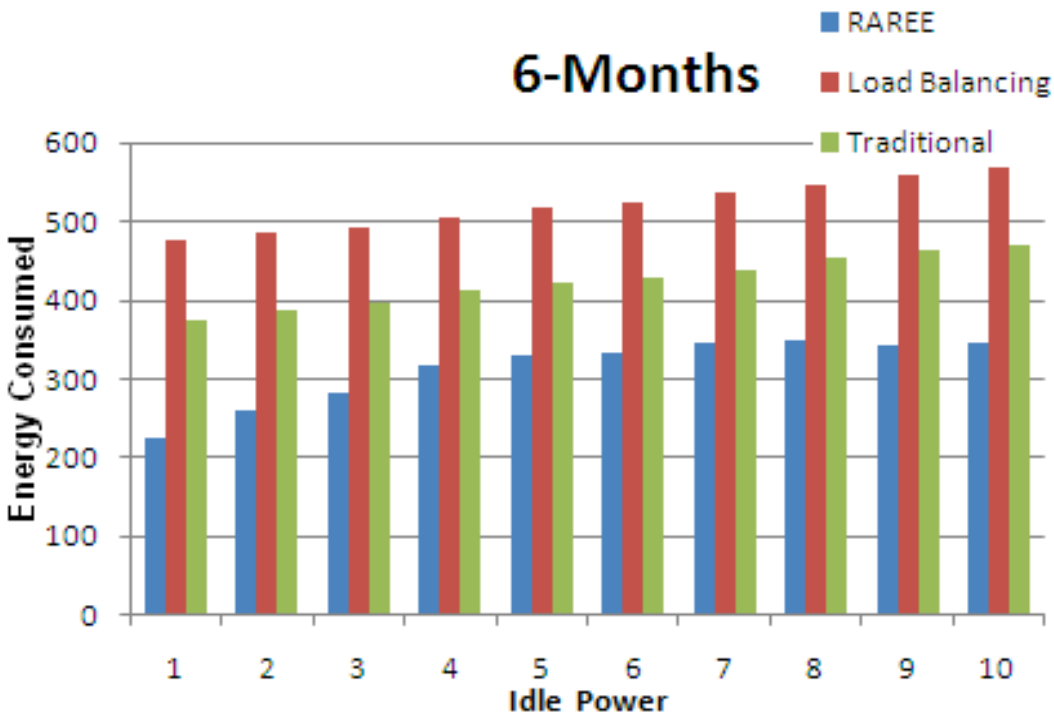

Figure 3.16. Idle Power (Watts) vs. Energy Dissipation (Joules).

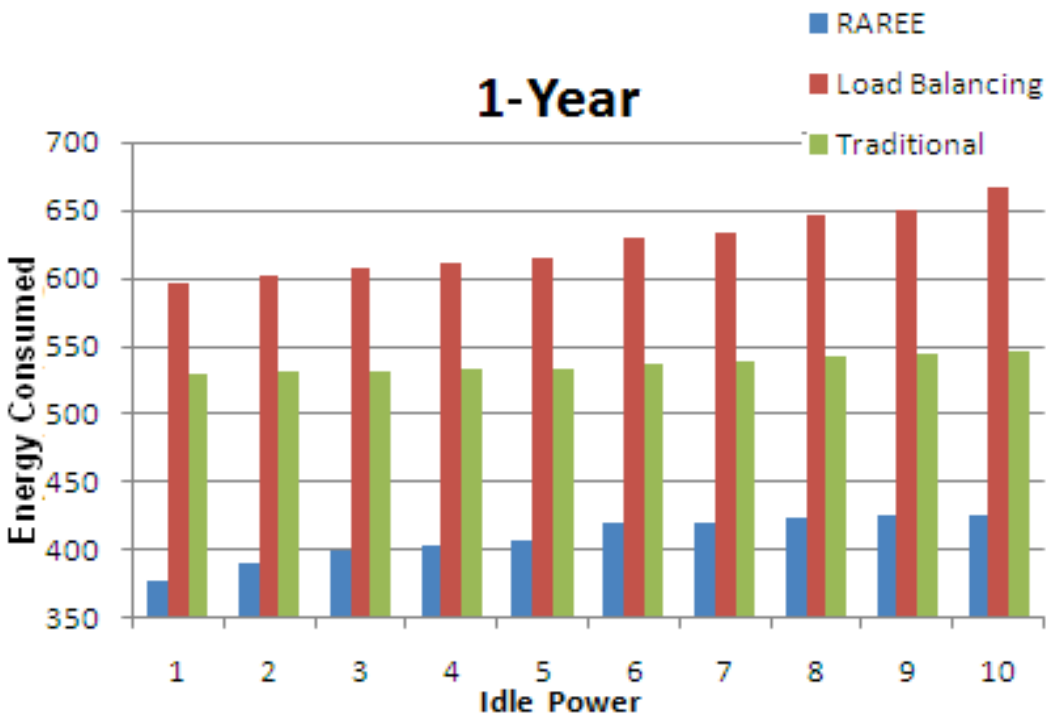

Figure 3.17. Idle Power (Watts) vs. Energy Dissipation (Joules). Disk Age = 1 Year.

Figs. 3.14 and 3.15 show the impact of active power on energy consumed by parallel disk systems. Figs. 3.14 and 3.15 reveal that regardless of disk age, energy savings provided by RAREE become more pronounced when one reduces active power.

Figs. 3.16 and 3.17 shows that when idle power is larger than 6, the energy efficiency of RAREE is no longer sensitive to idle power. Fig. 3.18 shows that although the average response time of our approach is slightly longer that than those of the other two in some cases, the performance degradation is usually less than 2 milliseconds. We believe that it is worth to trade marginal performance degradation for high reliability (see Table 1) and energy efficiency.

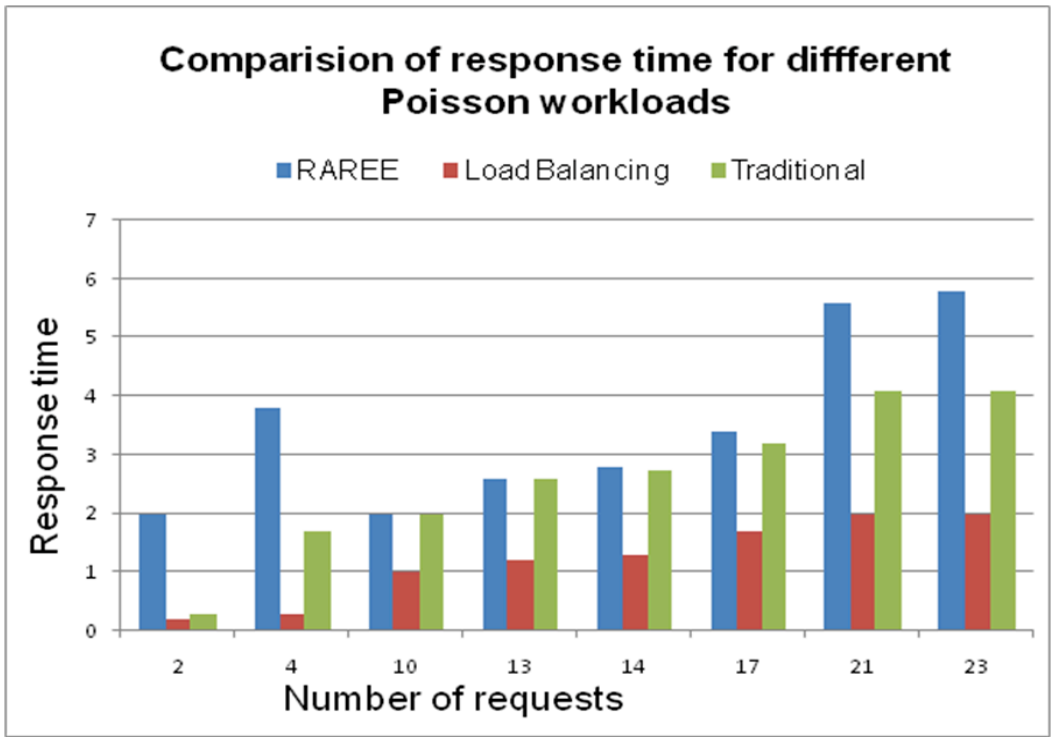

Figure 3.18 Arrival rate vs. response time

Table 3.3: Failure rate (%) of the parallel disk systems.

Age (Year)	Traditional/ Load Balancing	RAREE	Improvement (%)
0.25	9.827	0.017	99.8
0.5	47.080	5.502	88.3
1	52.729	34.646	34.3
2	24.957	17.205	31.1
3	0.923	0.425	54.0
4	8.263	3.996	51.6
5	24.414	13.003	46.7

Table 3.3 shows failure rates of parallel I/O systems with disk mirroring. Results summarized in Table 3.3 illustrate that our strategy significantly improves the reliability of parallel I/O systems over the two existing schemes. For example, RAREE can reduce failure rate by up to 99.8% with an average of 58.0%.

3.6 Summary

Although an array of energy conservation techniques have been proposed for disk systems, most energy saving schemes are implemented at the cost of disk reliability. In order to solve this problem, we first built a model to quantify the failure probabilities of disk systems as a function of disk age and utilization. In particular, we focused on parallel I/O systems with mirroring disks, where data sets are mirrored to backup disks from primary disks. Traditional power management methods wake up the backup disks when the utilization of the primary disks exceeds a certain threshold. Load balancing techniques, on the other hand, keep both primary and backup disks always active to balance the load between the two disks to achieve high performance. However, the load balancing methods consume a massive amount of energy for large-scale parallel I/O systems. Hence, we aimed at developing a utilization-based control mechanism to reduce energy dissipation in parallel disk systems without degrading system reliability. This goal was achieved by enforcing parallel disks to be operated in safe utilization zones, within which disk failure probability is minimized. This is the first utilization-based control scheme that seamlessly integrates reliability with energy saving techniques in the context of parallel I/O systems. Experimental results show that our approach can significantly improve reliability while achieving high energy efficiency for parallel I/O systems with disk mirroring.

Chapter 4

Effects of Power State Transitions on Reliability of Energy Efficient Storage Systems

In the previous chapter, we have designed a reliable and energy-efficient algorithm for large scale storage systems. The algorithm switches the disks between different power modes to save the energy and operates in safe utilization to attain reliability. However, switching the disk system from one state to another state to save energy involves an overhead caused by spin up and spin down. Therefore analysis of power management mechanism described in chapter 3 has to take into account the overheads involved in disk spin up and/or spin down during transition of states from one power mode to another. The overhead involved affects both reliability and the energy efficiency of the disks. In this chapter, we present an analytical model to quantify energy efficiency, reliability and the effect of power state transitions on both.

This chapter is organized as follows. Section 4.1 presents the motivation of this study. In section 4.2, we defined the system model used in this study and it also presents the energy efficiency and disk reliability explanation. Next, in section 4.3, we

discuss the control algorithm for power state transitions in detail. In section 4.4, we present the performance evaluation of our approach, with a detailed explanation about the experimental set up and experimental results in section 4.2.1 and 4.2.2 respectively. Finally, section 4.5 summarizes the entire chapter.

4.1 Motivation

The ever increasing number of large scale complex applications call for high performance data centres. This translates directly to an increasing volume of spinning disks creating a huge and growing energy crisis. In order to provide reliable storage services, often, disk-array based storage systems (such as RAID) are used. However these systems have been designed in a manner to gain in performance without considering the energy constraints. Current worldwide data growth is estimated at a compound annual growth rate of 50.6% through the decade [7]. Several studies have indicated that data centres are headed towards a serious energy crisis. The energy crisis problem has a rippling effect as additional power is required. For instance 22% of the total energy consumption is due to cooling systems used to keep temperatures of data centres from rising high degrees. The solution to power consumption problem has been addressed at the hardware level to a certain extent. But the problem remains largely open in many deployments of large scale storage systems. Energy consumption of storage systems in data centres is becoming a predominant concern in the IT industry.

On the other hand RAID based solutions have been used to dramatically reduce the down-time due to hardware problems, minimizing data loss due to drive failures. RAID based solutions have been widely used in the storage systems industry over the

last ten years because of the affordable price of energy. However, with an increasing shortage of power resources this is not true anymore. RAID implementations provide a degree of fault tolerance depending on the specific RAID configuration used. However reliability and fault tolerance being two sides of the same coin, the overall reliability of a RAID based storage system depends on the reliability of individual disks. Therefore we still see a large scope for studying ways to improve disk reliability. Nevertheless, energy is nowadays becoming more and more effective as a major expense in every businesses budget. Indeed, the extensive success and use of the RAID based technology has led the industrial and research communities to focus more on ways to lower the unaffordable amount of energy consumed.

To find a solution to this problem we have developed an algorithm which saves energy by operating the disks in different power modes and attains reliability by operating the disks only in safe utilization zone. However, to operate disks in different power modes they will be switched back and forth which involves a significant amount of overhead with respect to disk spin ups and spin downs. Disk manufacturers also warn that the disks can hold only particular number of spin ups and spin downs, after that the disk reliability is going to getting effected. Hence there is a need to control the number of state transitions that occur in a disk system. In this chapter we proposed an algorithm to control the disk transitions thereby improving the disk reliability while achieving the desired energy efficiency.

4.2 System Model

Disk age and utilization are studied as the primary factors that affect disk drive reliability. Based on failure statistics presented by Google [19] for disk systems, we

studied the failure probabilities with respect to disk age and utilization. We then estimated safe utilization zones for disks of differing ages. Safe utilization zone is the range of utilization levels within which the probabilities of disk failures are minimal.

We designed a policy where disks operate at different power modes based on the current utilization levels. As for reliability we first take a step towards quantifying the effects of disk utilization and disk age on the overall disk reliability. We present an empirical reliability metric called Disk Reliability Indicator (DRI) which incorporates the effects of utilization and disk age on disk drive reliability in terms of Annual Failure Rate (AFR). We then study the relative trade-offs between percentage energy savings and reliability in terms of AFR with every power state transition. Finally we propose a control mechanism to ensure an operation of disks in safe utilization zone in an energy efficient manner with an optimal tradeoff in terms of reliability.

4.2.1 Energy Efficiency

Based on the state transition diagram in Fig 3.5, it is quite evident that maximum energy efficiency is achieved by operating disks in State 1. This however is not feasible for server side disk storage systems as the workloads on them often have idle time slots too small to justify the overhead caused by frequent spin up and spin down. At the same time energy is not saved by keeping the disk systems in State 3 as both the disks run in active mode all the time in this state. A key factor that determines the amount of energy consumed in such a system is the power state transition frequency *f*. It represents the number of state transitions that occur per day and is cumulative i.e. it includes transitions between all the states. Let f_P and f_B represent the transition frequency of the primary and backup disks. The values of f_P and f_B can be derived

from the knowledge of state transition history of the entire RAID-1 system per day. It is to be noted here that both f_P and f_B comprise equal number of spin-ups and spin-downs. This is because we assume only 2-speed disks for our study which means the disks can only switch between two states.

The total energy consumed is simply the sum of energies consumed by individual disks i.e.

$$E = E_P + E_B \tag{4.1}$$

Let $P_{P,A}$ and $P_{B,A}$ denote the power of the primary and backup disks when they are in the active state and $P_{P,S}$ and $P_{B,S}$ the corresponding powers when they are in sleep mode. We have the following set of equations to calculate E_P and E_B.

$$E_P = (P_{P,A} * T_{P,A}) + (P_{P,S} * T_{P,S}) + \frac{f_P}{2} * (T_{P,spindown} * P_{spindown} + T_{P,spinup} * P_{spinup}) \tag{4.2}$$

$$E_B = (P_{B,A} * T_{B,A}) + (P_{B,S} * T_{B,S}) + \frac{f_B}{2} * (T_{B,spindown} * P_{spindown} + T_{B,spinup} * P_{spinup}) \tag{4.3}$$

where,

- $P_{P,A}$ and $P_{B,A}$ → power consumed in active state by primary and backup disks respectively
- $P_{P,S}$ and $P_{B,S}$→ power consumed in sleep state by primary and backup disks respectively
- $T_{P,A}$ and $T_{P,S}$ →are the times spent by the primary disk in active and sleep state respectively
- $T_{B,A}$ and $T_{B,S}$ →are the times spent by the backup disk in active and sleep state respectively

- $P_{spin\text{-}down}$ and $P_{spin\text{-}up}$ → are the spin down and spin up power consumptions by the disk and is same for both primary and backup disks.

Given a list $R = (R_1, R_2, \ldots R_n)$ of disk requests, with a given pattern we can calculate values for utilization from which we can arrive at a sequence of state transitions and the amount of time spent in each state. Using equation (4.1) we can then calculate the amount of energy consumed for the total duration. It is to be noted that our analysis also considered the energy cost of power state transitions.

4.2.2 Disk Reliability

The most challenging aspect of research in reliability of storage systems is to quantify the relationship between utilization (and therefore power state transitions) and reliability. A frequency-reliability function based on a combination of the spindle start/stop failure rate adder suggested by IDEMA [30] and the modified Coffin-Manson model was built in [76]. The rationale behind their work is that disk speed transitions and spindle start/stops essentially generate the same type of disk failure mode, spindle motor failure, though with different extents. Since power state transitions directly correspond to speed transitions of a disk we believe the results of [76] will be a good starting point for our studies.

Each time a hard disk drive undergoes a power cycle or speed transition, damage is caused due to temperature change and accumulates with repeated cycles of temperature change. Such cycles induce a cyclical stress, which weakens materials and eventually makes the disk fail [21]. A well known mathematical model that evaluates the reliability effects of cycles of stress or frequency of change in temperatures is the Coffin-Manson model. Modified Coffin-Manson model based on

quadratic curve fitting is used to obtain the following reliability-frequency function where *R* is the reliability in AFR and *f* is the disk power state transition frequency [76].

$$R(f) = 1.51e^{-5} f^2 - 1.09e^{-4} f + 1.39e^{-4} \tag{4.4}$$

Next, we analyze the reliability of a mirrored disk system with our approach. We aim to derive failure rate p(α, μ) of a parallel I/O system with disk mirroring. The failure rate largely depends on the age α and utilization μ of a pair of primary and backup disks. Let α = (αP, αB) represent the ages of the primary and backup disks. Note that subscripts P and B represent primary and backup disks, respectively. We denote $P_P(\alpha_P)$ and $P_B(\alpha_B)$ as the failure rate of the primary and backup disks. Given that disk failures are independent, we compute reliability r(α) of the parallel I/O system as

$$r(\alpha) = 1 - R(f_P) \cdot R(f_B) \tag{4.5}$$

$R(f_P)$ and $R(f_B)$ represent the failure rate of primary and backup disks (of given age) with given power state transition frequency.

Input: *M*: Number of processors in the system

λ: Poisson arrival rate of requests generated by each of the M processors

Ω: Time interval between arrivals follows exponential distribution

S_u: Spin up power

S_d: Spin down power

P_a: Active power

P_i: Idle power

P_a: Sleep power

α: Break even time (Spin up + spin down power)

Q: Buffer

1. Insert λ into *Q* buffer
2. Calculate the utilization μ of the disk based on the number of requests in the buffer
3. Get history values for utilization.
4. **If** μ is 20% more than history value then

use the new value

else use old value

5. Incorporate current value μ to history
6. Calculate the safe utilization zones (min, max) for the given disk
7. Compare the μ value against the min and max values
8. **If** the system is in state 1 **then**

if μ <min **then** no state transition occurs

if μ >min **then** system transits from state 1 to state 2

9. **If** the system is in state 2 **then**

if μ <min && α < Ω **then** system transits from state 2 to state 1

if μ >min && μ <max **then** no state transition occurs

if μ >max **then** system transits from state 2 to state 3

10. **If** the system is in state 3 **then**

if μ <min && α < Ω **then** system transits from state 3 to state 1

if μ >min && μ <max or α < Ω **then** system transits from state 3 to state 2

if μ >max **then** no state transition occurs

Figure 4.1 CAPST Algorithm

4.3 Control Algorithm for Power State Transition (CAPST)

Although we have established the fact that in order to achieve energy efficient and reliable operation of disks they have to be operated in safe utilization zone. In this section we propose a control mechanism called CAPST - Control Algorithm for Power State Transition that induces a certain degree of control over fluctuation of utilization values due to dynamic workloads.

In order to control the fluctuations in utilization, we use a time-window based control mechanism. To determine the operational state of the storage system, CAPST first forecast the utilization levels in the next time-window based on history of observed values. At the beginning of each time-window, CAPST sets the utilization values based on the current and the historical value and then does a power state transition if the current values of utilization are consistent with historical data. Every time a new value for utilization is generated, CAPST moves the time-window by one margin adding the latest value to its historical data. The algorithm is summarized in Figure 4.1.

4.4 Performance Evaluation

4.4.1 Experimental Setup

We developed an execution-driven simulator that models an RAID-1 array of 2-speed disks. We believe multi-speed disks have not been largely manufactured and deployed currently. Therefore, there are very few or zero reported results about impacts of utilization on disk reliability of multi-speed disks. Owing to the infancy of

multi-speed disks, we derived corresponding 2-speed disk statistics from parameters of a IBM Ultrastar 36Z15 disk. For the sake of simplicity we considered array of 2 disks. However the results and trends hold good for a larger set of disks too.

The disk parameters that are used in the experiments and their parameters are given in Table 4.1.

Table 4.1: Disk Parameters of IBM36Z15

Parameter	IBM 36Z15 Ultrastar
Standard interface	SCSI
Capacity	18 GBytes
Number of platters	4
Rotations per minute	15000
Disk controller cache	4 Mbytes
Average seek time	3.4 msec
Average rotation time	2 msec
Internal transfer rate	55 MB/sec
Power(active)	13.5 W
Power(idle)	10.2 W
Power(standby)	2.5 W
Energy(spin down)	13.0 J
Time(spin down)	1.5 sec
Energy(spin up)	135.0 J
Time(spin up)	10.9 sec

The experimental results are compared against two traditional state of the art methods. In the first method, ‘load balancing’ in which both the disks are always made active. Load balancing achieves very high performance because both the disks share the load. The second method, ‘traditional method’ is where the primary disk is made always active and the backup disk is kept in sleep mode. The backup disk is

made active only when the utilization of the primary disk exceeds 100%, also known as saturation.

4.4.2 Experimental Results

We first evaluated our approach and then compared against traditional state of the art methods. Our main goal in evaluating our approach was to study the variation in disk utilization over the entire duration of experiment, which triggered state transitions. This in turn was used to calculate the energy consumed in the process using equation (4.1). For different request arrival rates we obtained different power state transition frequency. We then used those values of frequency to study cost of energy savings in terms of reliability as given by equation (4.5).

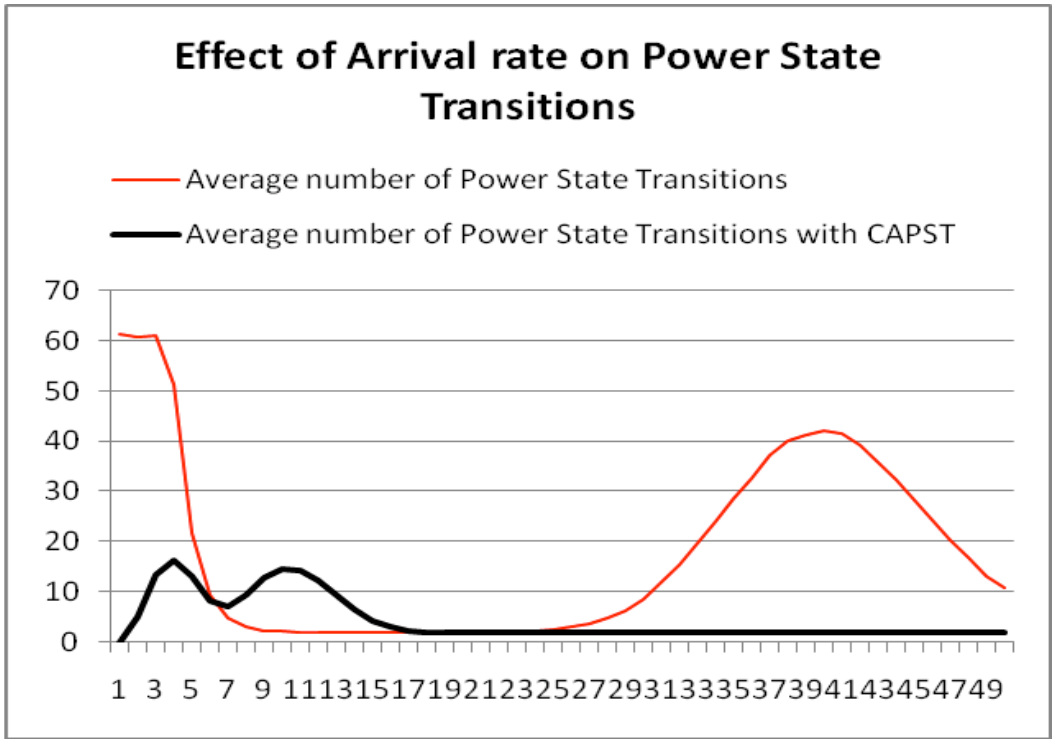

Figure 4.2 Effect of arrival rate on Power State Transitions

From figure 4.2 we observe that for lighter workloads the system makes many power state transitions. At this stage the system is operating in state 1 and state 2. As

the arrival rate increases further the number of power state transition frequency becomes steady. We believe it is at this stage that the system operates most energy efficiently with optimal reliability. With further increase in the workload the fluctuations in the arrival pattern also increases causing more state transitions i.e. the transition frequency increases further and the systems starts switching more towards state 3. At very high workloads where the utilization levels are high the system operates at state 3 with both the disks serving the requests reducing the power transition frequency. At this stage energy consumed is high, and the disks operate outside the safe utilization as well. We make a note of the range of values of μ for which utilization is within the safe utilization zone.

This gives us an idea of the rate at which the requests must be serviced in order for the disk system to operate in a safe utilization zone.

Next, we ran the simulation and noted the fluctuation in the values of utilization with and without CAPST. Fig 4.3 depicts a plot of the recorded values over the length of the simulation.

It is quite evident from fig 4.3 that the number of Power State Transitions is evened out, minimizing the number of state transitions there by reducing the reliability overhead of the transitions. It is to be noted that the utilization values with CAPST vary between 20% and 60% which is the safe utilization zone. We calculated the total energy consumed to be 21425.3 J and reliability to 99.3% calculated as per equations (4.1) and (4.5)

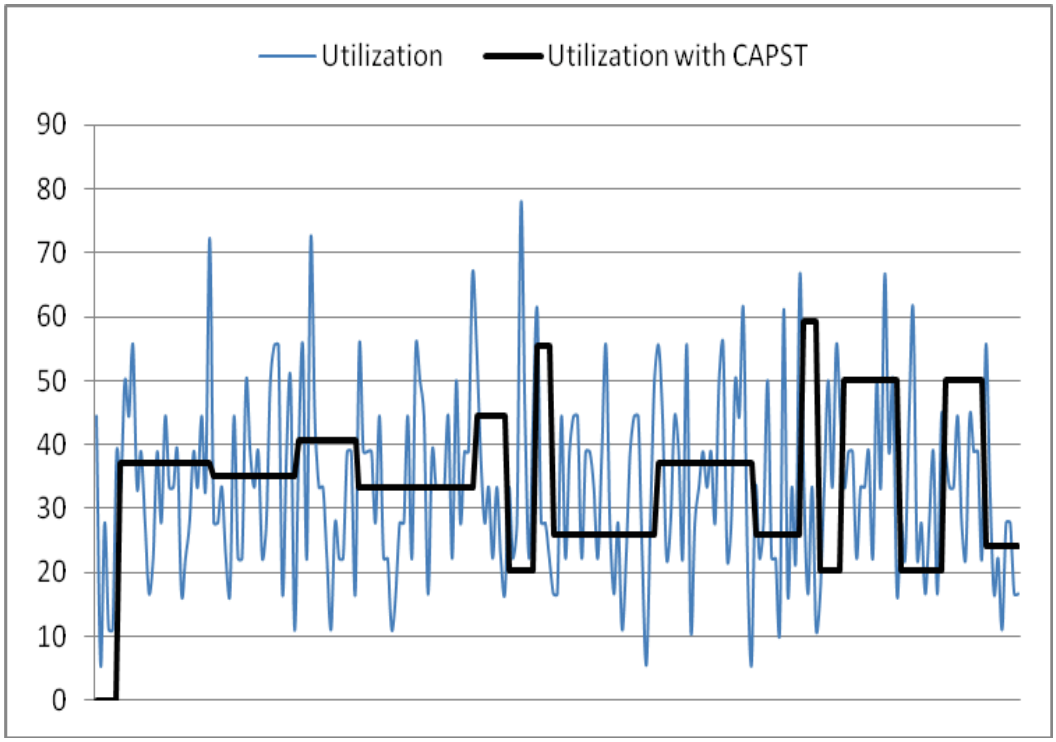

Figure 4.3 Power State Transitions with CAPST

Based on the disk parameters mentioned in table (4.1), we next studied the variation in energy consumption and reliability. The results are depicted in Fig 4.4 and Fig 4.5 we notice that for the value of $f < 3$ both reliability increases and energy consumption decreases in a steady manner. This is in strong agreement with the results shown in Fig 3.2. Looking back we notice that values of $f < 3$ for workloads with arrival rate 14 and above i.e. when the disk system operated in safe utilization zone.

Traditional methods wake up the backup disk when the utilization of the primary disk exceeds 100 percent. Load balancing technique keeps both the primary and back up disks always active to share the load. These methods consume a massive amount of energy, as the disks stay active even when there are no requests to serve for a long period of time. The reliability of these disks is also ignored most of the time.

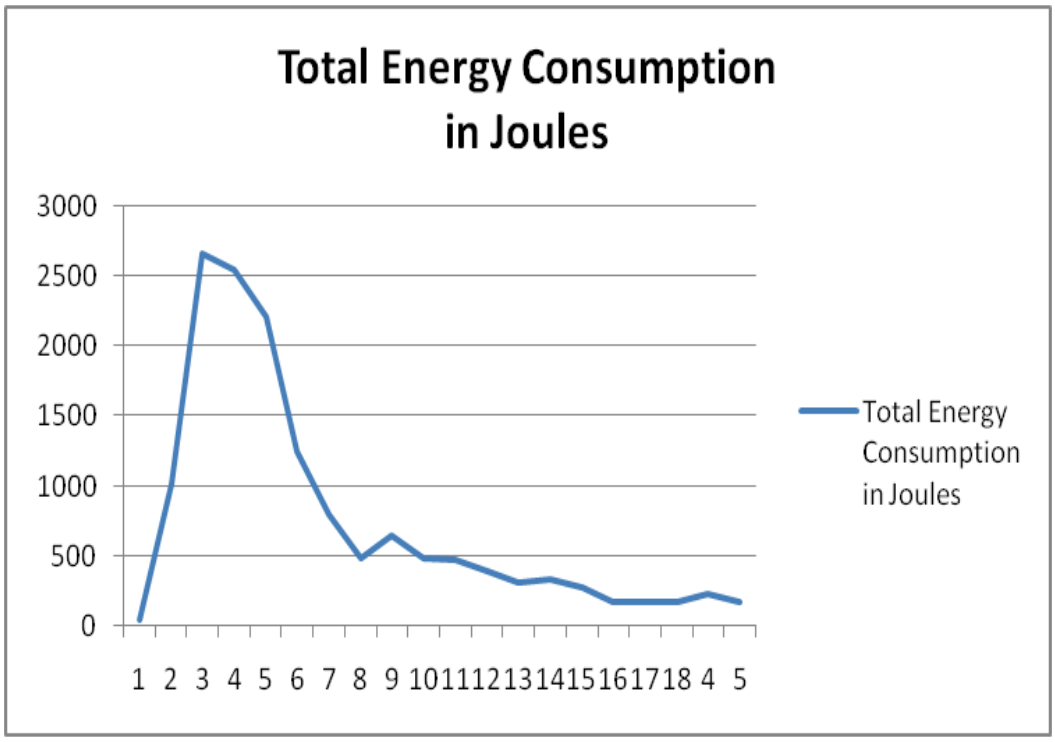

Figure 4.4 Variation of Energy consumed with Power Transition Frequency

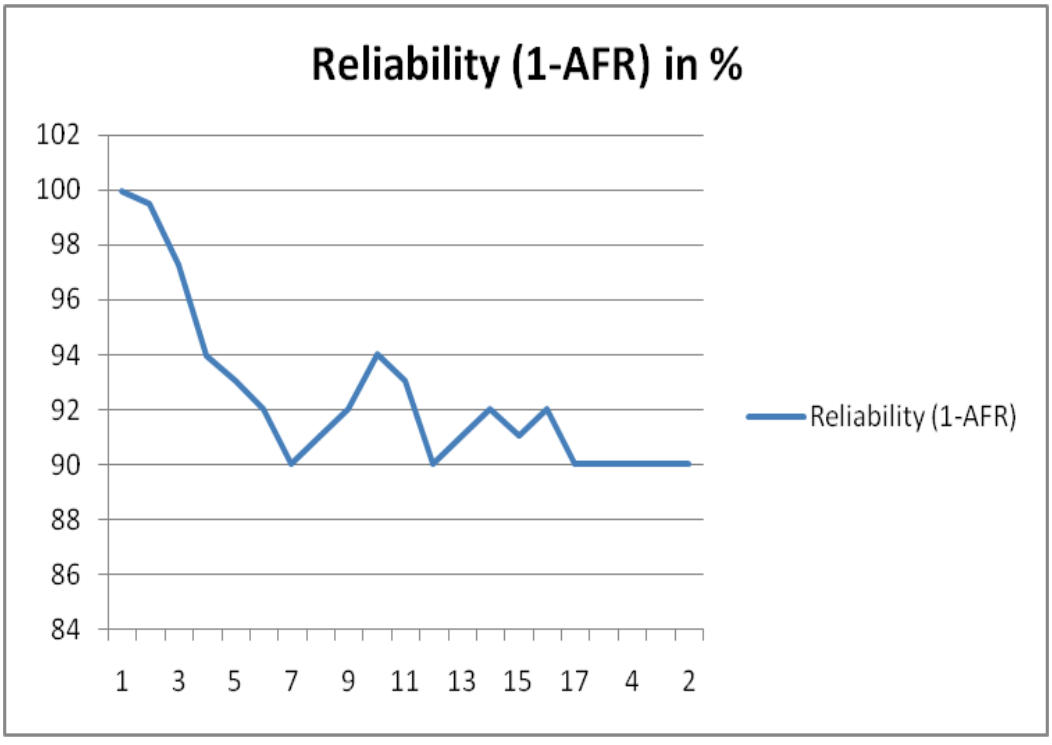

Figure 4.5 Variation of Reliability with Power Transition Frequency

Following table 4.2 depicts the reliability improvement as compared to the other 2 methods.

Table 4.2: Comparison of Reliability

Age (Year)	Traditional/ Load Balancing	CAPST based Speed Control	Improvement (%)
0.25	9.827	0.023	99.8
0.5	47.08	4.214	91.0
1	52.729	23.752	55.0
2	24.957	14.328	42.6
3	0.923	0.302	67.3
4	8.263	2.172	73.7
5	24.414	10.202	58.2

4.5 Summary

In this chapter, we designed and implemented a utilization control mechanism called CAPST to improve both reliability and energy efficiency of disk systems, after proposing a novel concept of safe utilization zone where energy of the disk can be conserved without degrading reliability,. The utilization control mechanism ensures that disk drives are operated in safe utilization zones to minimize the probability of disk failure. We integrate the energy consumption technique that operates the disks at different power modes with our proposed reliability approach. Simulation results show that our approach can significantly improve reliability while achieving high energy efficiency for disk systems.

Chapter 5

Utilization based Reliable Energy Efficient Disks - UREED

In the previous chapters, we have designed an energy efficient and reliable algorithm for storage disks. We also proposed a way to control the power state transitions to increase the energy efficiency while guaranteeing the reliability. In this chapter we further extend our work by designing the utilization based reliable energy efficient disks algorithm, called UREED. Skewing popular data to the buffer disks and controlling the utilization of the disks is the main idea implemented in this chapter, to achieve high levels of reliability along with maximum energy efficiency.

This chapter is organized as follows. Section 5.1 presents the motivation of this study. In section 5.2, we define the models description of our file server. Next, in section 5.3, we discuss the UREED algorithm. In section 5.4, we present the mathematical models of file servers UREED, MAID and PDC. Section 5.5 details the performance evaluation along with the simulation description and simulation results. Finally, section 5.6 summarizes the entire chapter.

5.1 Motivation

Large scale storage systems have been the centre of many researches due to the substantial growth of high performance data centres. Large scale complex applications running on these high performance data centres consume an enormous amount of energy. Current worldwide data growth is estimated at a compound annual growth rate of 50.6% through the decade [7]. Several studies have indicated that data centres are headed towards a serious energy crisis. The energy crisis problem has a rippling effect as additional power is required. For instance 22% of the total energy consumption is due to cooling systems used to keep temperatures of data centres from rising high degrees.

The importance of energy conservation for large scale disks has been identified by many pioneer researchers and it has been studied thoroughly [18] but most of the published work in the arena of storage systems concentrates on the issue of energy conservation of the disks, where as equally important concern reliability is ignored in many studies. Making a disk operate with high energy efficiency is complimented only when it runs without causing the disk to fail. Most of the energy conservation techniques adversely affect the disk reliability. Once a disk fails the amount of energy spent in rebuilding the disk outweighs the amount of energy conserved by operating the disks in energy efficient mode. Hence there is a need for very reliable and highly energy efficient storage disks.

The research explained in this chapter is motivated by the above reasons. Here we use the data skewing technique to keep very few disks active in the system to serve requests and thereby increase energy efficiency. By operating these skewed disks and

data disks only in the safe utilization until unless it is necessary to operate those outside we also attained good reliability.

5.2 Model Description

In this section we propose the utilization based reliable energy efficient disk model, referred as UREED. The model outlines the working conditions and the strategies followed to achieve the energy efficiency and reliability.

Let the total number of disks in a disk array be D with U data disks and V buffer disks.

$$D = U + V \tag{5.1}$$

Where U and V are positive integers and ($1 < U < D$) and ($1 \leq V < D$) and $U = \{1,2 \ldots p\}$; $V = \{ 1, 2, \ldots q\}$

Assign n files $f_1, f_2 \ldots f_n$ among U data disks. Each file is partitioned in to $I = \{1, \ldots, m\}$ blocks where I_i is the set of indices corresponding to the files assigned to disk U_i. For simplicity of presentation, file partitioning is not considered in this work; thus each file must be assigned in its entirety to one disk. Similar assumptions were made in [48]. This does not restrict the generality of our model since if a file is portioned, each partition can be viewed as a standalone file.

Poisson process with a mean access rate of λ_i, known a priori is used to model the disk accesses of each file [8]. We assume a fixed service time s_i for each file f_i (Similar assumptions were made in [48]. Disk accesses on each file are directly linked to the popularity of the data block. It has been showed that many network server workloads have highly skewed file access frequencies [14][47][18] and that the web

server file access frequency conforms to a Zipf distribution. We used Zipf's law to predict the frequency of access or popularity £ of a file

Initially all files are sorted in descending order *[£I1, £I2 ...£In]* based on their popularity in to list *X* of size *n*. All the sorted files are copied on to the data disks by randomly choosing the disks and allocating the next contiguous file from the sorted list. This allows the distribution of files on to disks more or less evenly. Now we have balanced file assignment on data disks.

A list *Y* is generated to hold the popular data for buffer disks with size *k* (where *1<k<n*) which is same as the capacity of the buffer disks. The first *k* popular files *[£I1, £I2 ...£Ik]* of list *X* are sorted in descending order on to list *Y*. The reason for generating list *Y* instead of using the files from list *X* directly is because the list *Y* will be used later on repeatedly to apply UREED periodically. The top segment of the list *Y* is assigned to the buffer disk V_1 and the next file segment is assigned to the buffer disk V_2 and so on until the buffer disk V_q is assigned a segment then the process is repeated from buffer disk V_1 to copy the remaining segments. This achieves balance on the buffer disks with respect to the popularity of the files saved on them. Operating the disks within the safe utilization promises reliability, so to keep the disk utilization within the safe utilization zone the disk utilization value is computed and then it is compared with the μ_{Max} and μ_{min} (where μ_{Max} and μ_{min} are the maximum and minimum values of the safe utilization zone)

The utilization of a disk can be calculated using the formulae

$$\mu_i = \frac{Ri}{B_{disk}} = \frac{\sum_{j=1}^{\phi_i} (\lambda_{ij} \cdot \beta_{ij})}{B_{disk}} \qquad (5.2)$$

Where B_{disk} is the bandwidth of the disk, ϕ_i is the number of data-intensive tasks running in the *i*th phase, λ_{ij} , $1 \le j \le \phi_i$, denote the arrival rate of disk request submitted by the *j*th data-intensive task. β_{ij} , $1 \le j \le \phi_i$ be the average data size of disk requests of the *j*th task and R_i is the accumulative data amount access per time unit.

If the computed utilization value is within the range then the requests are served by the buffer disks as they arrive, if the value is beyond the range then the corresponding data disks are brought to active mode to serve the requests in a timely manner, which promises performance and at the same time keeping the disks in the safe utilization zone improves reliability. If the computed value is less than the range then the disk requests are kept in the queue until the utilization reaches μ_{min}, then the requests are served. To avoid the performance bottleneck the requests are made to wait in the queue only until the defined threshold value expires; once the threshold value expires the requests are served, without this condition, the requests might have to wait in the queue forever, which degrades performance. The chances that the computed value is less than μ_{min} are close to zero because the data servers have very high workloads at any given point of time.

Metadata is maintained for all the disks. Metadata management is out of scope of this chapter.

5.3 UREED Algorithm

Using metadata management the data blocks with higher frequency of access are kept track off. Every time a data block is accesses the frequency of access rate value is incremented by one in the metadata.

Since the popularity of the files keeps changing this algorithm should be applied periodically. The above section described how the disk system is set up at the beginning and after a period of time the system is ready to reapply the algorithm. First the popular data is gathered from the metadata information and it is sorted and placed in the list Y of size K, which means only K number of popular files are selected to replicate in the buffer disks.

Now the first segment of the list Y is compared against all the files of buffer disk V_1, if the disk access rate of a file that is in buffer disk V_1 is less than that of the first segment of list Y and if it is also the least popular file in that disk then the file is replaced with the new popular file. This process is repeated for all the segments of list Y and for all the buffer disks.

By repeating this algorithm periodically it is made sure that only the popular data resides in the buffer disks which reducing the overhead of waking up data disks to serve the requests.

5.4 Mathematical Models of the File Servers

In this section we explain three different file servers used for our research. UREED is the file server designed by us to improve the energy efficiency and reliability of the disks systems using utilization control and also data skewing methodologies. The two most cutting edge research file servers available in the arena of energy efficient disk systems are popular disks concentration, known as PDC [18] and Massive arrays of idle disks, referred as MAID [11]. These two file server models are proposed by the pioneers in the area of energy efficiency; it should be noted here that these two models completely ignore the reliability issue. To address this issue we

developed our UREED file server. In what follows we propose the mathematical models for all the three file servers. We used these models to generate the data about the energy consumption for all the three models. We compared this data to our simulation results and thus validated our model.

5.4.1Massive Arrays of Idle Disks (MAID)

MAID follows a non-migratory, block-level design and divided the system into zero or more "cache drives". These cache drives remain constantly spinning where as "data drives" are spin-down after a period of inactivity. It mainly concentrated on the power management policy, data layout and cache use. Power management is controlled by a simple non-activity policy, that when the disks are inactive for a particular amount of time then they are spun down. By spinning down the disks when they are not active MAID saves the energy. Cache drives are updated periodically using the LRU policy. Next we explain the mathematical model of MAID.

5.4.2 Mathematical Model of MAID

MAID file server has two types of disks drives, data disks and cache/buffer disks. We assume that D be the total number of disks in the disk system with U data disks and V buffer disks.

$$U_1 = \{U_{11}, U_{12}, U_{13} \ldots\ldots U_{1m1}\}$$

$$U_2 = \{U_{21}, U_{22}, U_{23} \ldots\ldots U_{2m2}\}$$

$$\vdots$$

$$U_n = \{U_{n1}, U_{n2}, U_{n3} \ldots\ldots U_{nmn}\}$$

Where m_i is the number of files on the i^{th} data disk

Data disks holds the files that are least recently used and the most recently used once are copied on to the buffer disks. Each data disk can hold the same number files; here we assume that the file size is fixed.

$$V_1 = \{V_{11}, V_{12}, V_{13} \ldots\ldots V_{1m1}\}$$

$$V_2 = \{V_{21}, V_{22}, V_{23} \ldots\ldots V_{2m2}\}$$

$$\vdots$$

$$V_n = \{V_{n1}, V_{n2}, V_{n3} \ldots\ldots V_{nmn}\}$$

Here m_i is the number of files on the i^{th} buffer disk

Each file that is stored on either data disks or buffer disks has a popularity value. This value is used to determine whether the file needs to reside in the data itself or it should be moved to the buffer disks. Popularity of the file disks are assigned by the least recently used policy in MAID. Least recently used policy works well to identify the popular files only when the disks access follows the pattern. When the disk access pattern varies, popularity of the file changes dynamically. This issue is not addressed in MAID.

$\rho_{v_{ij}}$ is the popularity of the i^{th} buffer disk and j^{th} file of the MAID file server.

The files in any of the buffer disk have higher popularity than the data disk files.

We assumed a list Z of size k same as the capacity of the buffer disks to hold the popular data for the buffer disks that is

$$Z = \left\{ Z\mu_1, Z\mu_2 \cdots Z\mu_k \right\} \tag{5.3}$$

Z_{μ_i} is the i^{th} popular data stored in the list *Z* and the value of *i* is between *1* and *K*. MAID caches the data into popular disks or buffer disks using least recently use (LSU) policy.

MAID data layout is either linear or striped across multiple disks. The popularity of the files in the buffer disks and data disks changes dynamically. So periodically the files should be checked for the updated popularity values and the most popular files should be copied on to the buffer disks.

That is to move all the popular data on to the buffer disks MAID should be performed periodically. For that

$$\forall(i \le j \le m_i) \& Z = [1,2 \ldots k]$$
$$\rho_{v_{ij}} < \rho_{Z_{\mu_x}} \tag{5.4}$$

The popularity of the first segment of the list *Z* is compared with that of all the segments of the buffer disk and if the file in the list *Z* has higher popularity then it is replaced by the file in buffer disks.

That is v_{ij} is replaced by the Z_{μ_x}. The process is repeated for *k* iterations until the $z \in \theta$

Now all the popular data is copied on to the buffer disks while the non popular data resides in the data disks. These disks keep serving the requests by keeping the buffer disks active most of the time and keeping the data disks in idle mode.

Utilization of the i^{th} disk is calculated by 5.5 without the copying overhead

$$\rho_i^1 = \sum_{j=1}^{m_i} \rho_{ij}^1 \tag{5.5}$$

The utilization of the i^{th} disk with overhead for a given time frame *T is*

$$\rho_i^1 = \frac{I/O\,time + Copying\,time}{T} \tag{5.6}$$

$$= \frac{\sum_{j \in F_i^1} \lambda_{ij}^1 t_{ij} + \sum_{j \in F_i^m} t_{ij}}{T} \tag{5.7}$$

$$= \sum_{j \in F_i^1} \lambda_{ij}^1 t + \frac{\sum_{j \in F_i^m} t_{ij}}{T} \tag{5.8}$$

By using equation 5.9 utilization of the i^{th} disk can be calculated along with the overheads. Summation of the disk utilization gives the total system utilization from which the amount of energy consumed can be calculated.

$$= \sum_{j=1}^{m_i^1} \rho_{ij}^1 + \frac{\sum_{j \in F_i^m} t_{ij}}{T} \tag{5.9}$$

MAID model presented here is derived to appear very close to our UREED algorithm for fair comparisons between the two.

5.4.3 Popular Data Concentration (PDC)

The idea behind PDC is to concentrate the most popular disk data by migrating it to a subset of the disks. This concentration should skew the disk load towards this subset, while other disks become less loaded. These other disks can then be sent to low-power modes to conserve energy. More specifically, the goal of PDC is to lay data across the disk array so that the first disk stores the most popular disk data, the second disk stores the next set of most popular disk data, and so on. The least popular disk data and the data that are never accessed will then be stored on the last few disks. In fact, the last few disks will also include the data that most frequently hit in the main memory cache [18].

5.4.4 Mathematical Model of PDC

Unlike MAID and UREED PDC file server does not contain any buffer disks. The second major difference of PDC file server is that it uses file migration instead of file copying. This technique saves the additional disk cost involved with buffer disks but it comes at the cost of disk space.

Let U be the total number of data disks in the disk system. All the files are sorted in the data disks according to their popularity. The most popular files are stored on disk one and the next popular files are stored on disk two and so on. So the last disk contains the least popular data compared to all other disks.

$$U_1 = \{U_{11}, U_{12}, U_{13} \ldots\ldots U_{1m1}\}$$

$$U_2 = \{U_{21}, U_{22}, U_{23} \ldots\ldots U_{2m2}\}$$

$$\vdots$$

$$U_n = \{U_{n1}, U_{n2}, U_{n3} \ldots\ldots U_{nmn}\}$$

Here m_i is the number of files on the i^{th} buffer disk. PDC allows the most popular disks to stay in the active mode while the least popular disks slip in to sleep or idle modes. This is the main principle behind PDC energy conservation.

$$\rho_{v_{11}} > \rho_{v_{12}} \ldots\ldots > \rho_{v_{1mi}} \tag{5.10}$$

Where $\rho_{v_{ij}}$ is the popularity of the i^{th} data disk and j^{th} file. The above inequality states that the files in the data disk have increasing popularity and for the disk to have the balanced popularity it should satisfy the following equation

$$\forall (1 \le k \le n) \tag{5.11}$$

$$\frac{1}{n} \sum_i \sum_j \rho_{v_{ij}} = \delta + \sum_j \rho_{v_{kj}} \tag{5.12}$$

δ is a constant. If the above equality is satisfied then the buffer disks are perfectly balanced.

We assumed a list X of size k same as the capacity of the buffer disks to hold the popular data for the buffer disks that is

$$X = \left\{ x_{\mu_1}, x_{\mu_2} \cdots x_{\mu_k} \right\} \tag{5.13}$$

The popularity of files in the disks keeps changing so the files should be periodically checked and will be skewed to the popular disks when the popularity of particular file increases. To move all the popular data on to the buffer disks PDC should be performed periodically. For that

$$\begin{aligned} &\forall (i \le j \le m_i) \& X = [1,2 \ldots k] \\ &\rho_{v_{ij}} < \rho_{x_{\mu_x}} \end{aligned} \tag{5.14}$$

When the above equation is true the j^{th} file of the i^{th} data disk v_{ij} is replaced by the first segment of the list X which is x_{μ_x}. The process is repeated for k iterations until the $x \in \theta$

Utilization of the i^{th} disk is calculated by (5.10) without the copying overhead

$$\rho_i^1 = \sum_{j=1}^{m_i} \rho_{ij}^1 \tag{5.15}$$

The utilization of the i^{th} disk with overhead for a given time frame T *is*

$$\rho_i^1 = \frac{I/O time + Migration time}{T} \tag{5.16}$$

$$= \frac{\sum_{j \in F_i^1} \lambda_{ij}^1 t_{ij} + \sum_{j \in F_i^m} (t_{ij}^{in} + t_{ij}^{out})}{T} \tag{5.17}$$

Using equation 5.18 the utilization of the i^{th} can be calculated and the summation of the disk utilization gives the utilization of the system, with which the PDC file server total energy consumed can be calculated.

$$= \sum_{j=1}^{m_i^1} \rho_{ij}^1 + \frac{\sum_{j \in F_i^m} (t_{ij}^{in} + t_{ij}^{out})}{T} \quad (5.18)$$

5.4.5 Utilization Based Reliable Energy Efficient Disks (UREED)

UREED concentrates on popular data as well. Same as in PDC, UREED identifies the popular data but instead of migrating the popular data UREED copies it to the buffer disks. By keeping the buffer disks in service mode and data disks in idle mode UREED achieves the energy efficiency. It also operates the disks only in safe utilization zone to maintain and improve the reliability. Both data disks and buffer disks are operated in safe utilization zone until and unless it is necessary to operate them outside the zone.

5.4.6 Mathematical model of UREED

UREED uses both data disks and buffer disks in the file server. Data disk hold the un-popular data, while the buffer disks hold the popular data.

Let D be the total number of disks in the disk system with U data disks and V buffer disks.

$$U_1 = \{U_{11}, U_{12}, U_{13} \ldots\ldots U_{1m1}\}$$

$$U2 = \{U_{21}, U_{22}, U_{23} \ldots\ldots U_{2m2}\}$$

$$\vdots$$

$$U_n = \{U_{n1}, U_{n2}, U_{n3} \ldots\ldots U_{nmn}\}$$

Where m_i is the number of files on the i_{th} data disk

$$V_1 = \{V_{11}, V_{12}, V_{13}\ldots.. V_{1m1}\}$$

$$V_2 = \{V_{21}, V_{22}, V_{23}\ldots.. V_{2m2}\}$$

:

$$V_n = \{V_{n1}, V_{n2}, V_{n3}\ldots.. V_{nmn}\}$$

Here m_i is the number of files on the i_{th} buffer disk.

Each file in the disk system has a certain level of popularity which determines whether the file stays in data disks or buffer disks. The popularity of file in the buffer disks is more or less same. In the case of PDC the disks have decreasing popularity. When the disks have decreasing popularity the disk access on the popular disks increases enormously, leaving the disk open to failures because of the excessive utilization. Hence in UREED we tried to balance the whole buffer disk popularity by distributing the popular data on to them evenly based on their rankings.

$\rho_{v_{ij}}$ is the popularity of the i^{th} data disk and j^{th} file. For the buffer disk to have the balanced popularity it should satisfy the following equation

$$\forall(1 \le k \text{ p } n) \tag{5.19}$$

$$\frac{1}{n}\sum_i \sum_j \rho_{v_{ij}} = \delta + \sum_j \rho_{v_{kj}} \tag{5.20}$$

δ is a constant. If the above equality is satisfied then the buffer disks are perfectly balanced.

By skewing the data on to few disks and operating them in active mode to serve majority of the requests and using the data disk sonly when the requested data or files are not available saves the energy.

To maintain reliability the disks must be operated in the safe utilization zone whose values are between μ_{max} and μ_{min}

$$\mu_{\min} \le \sum_{i=1}^{n} \rho_{v_i} \le \mu_{\max}; \forall 1 \le i < j \le n \tag{5.21}$$

The utilization of the disks are calculated periodically and checked against the safe utilization zone values. Based on the UREED algorithm either the requests will be served immediately or they will wait for awhile before they can be served. By operating the disks in the safe utilization zone UREED achieves reliability.

All the data is first sorted and stored on the data disk in a balanced manner. Then the popular data is identified and sorted based on the ranking or popularity on to the list, it is then copied on to the buffer disks. We assume a list *Y* of size *k* same as the capacity of the buffer disks to hold the popular data for the buffer disks that is

$$Y = \left\{ y_{v_1}, y_{v_2} \cdots y_{v_k} \right\} \tag{5.22}$$

The popularity of the files keeps changing so the popular data in the buffer disks of the UREED file server should be updated periodically. That is, to move all the popular data on to the buffer disks UREED should be performed periodically. For that

$$\forall (i \le j \le m_i) \& Y = [1,2 \ldots k]$$
$$\rho_{v_{ij}} < \rho_{y_{\mu_x}} \tag{5.23}$$

Here v_{ij} is the j^{th} file of the i^{th} buffer disk. The popularity of the first segment of the list is checked against the files in the buffer disks. At the first occurrence of a file that is least popular than the first segment and if it also the least popular file in the entire buffer disks then it is replaced by the first y_{μ_x} segment of the list .

The process is repeated for *k* iterations until the list *Y* becomes empty, $y \in \theta$

The i^{th} buffer disk is $v_i = \{v_{i1}, v_{i2} \cdots v_{im_i}\}$ at the beginning or after the zero iteration. UREED is applied to the buffer disks periodically. After the first iteration $v^1{}_i = \{v^1{}_{i1}, v^1{}_{i2} \cdots v^1{}_{im_i}$ -

the power is raised by 2 for the second iteration and soon after the k^{th} iteration $v^k_i = \{v^k_{i1}, v^k_{i2} \cdots v^k_{im_i}\}$.

It should be noted that here we did not show the iterations for data disks because of space complexity.

Utilization of the i^{th} disk is calculated using 5.24 (without the copying overhead)

$$\rho^1_i = \sum_{j=1}^{m_i} \rho^1_{ij} \tag{5.24}$$

The utilization of the i^{th} disk with overhead for a given time frame T is

$$\rho^1_i = \frac{I/O\,time + Copying\,time}{T} \tag{5.25}$$

$$= \frac{\sum_{j \in F^1_i} \lambda^1_{ij} t_{ij} + \sum_{j \in F^m_i} t_{ij}}{T} \tag{5.26}$$

$$= \sum_{j \in F^1_i} \lambda^1_{ij} t + \frac{\sum_{j \in F^m_i} t_{ij}}{T} \tag{5.27}$$

The utilization of a single disk can be calculated using equation 5.28. Summation of the disk utilization gives the UREED file server utilization from which the UREED file server energy consumed can be calculated.

$$= \sum_{j=1}^{m^1_i} \rho^1_{ij} + \frac{\sum_{j \in F^m_i} t_{ij}}{T} \tag{5.28}$$

5.5 Performance Evaluation

Our main goal here is to determine the conditions where UREED conserves energy and improves reliability when compared to PDC and MAID for two disk array types and a wide range of parameters.

We choose simulation to run our experiments as the real implementation of these experiments is very time consuming and also because of the limited resource availability.

To validate our simulation results we compared them against the results we derived from the mathematical model. Simulation ran on both synthetic and real traces.

5.5.1 Simulation

We developed an execution-driven simulator that models a RAID-1 array of 2-speed disks. We believe multi-speed disks have not been largely manufactured and deployed currently. Therefore, there are very few or zero reported results about impacts of utilization on disk reliability of multi-speed disks. Owing to the infancy of multi-speed disks, we derived corresponding 2-speed disk statistics from parameters of an IBM Ultrastar 36Z15 disk.

Table 5.1: Disk Parameters of IBM36Z15

Parameter	IBM 36Z15 Ultrastar
Standard interface	SCSI
Capacity	18 GBytes
Number of platters	4
Rotations per minute	15000
Disk controller cache	4 Mbytes
Average seek time	3.4 msec
Average rotation time	2 msec
Internal transfer rate	55 MB/sec
Power(active)	13.5 W
Power(idle)	10.2 W
Power(standby)	2.5 W
Energy(spin down)	13.0 J
Time(spin down)	1.5 sec
Energy(spin up)	135.0 J
Time(spin up)	10.9 sec

The disk parameters that are used in the experiments and their parameters are given in Table 1.

The experimental results are compared against two popular energy conservation methods namely PDC and MAID.

The processor generates the requests and the files are accesses randomly for a specified amount of time. We collected results for our file servers namely UREED-1, UREED-1, MAID-1, MAID-2 and PDC-1, PDC-2.

UREED and MAID make use of buffer disks to hold the popular data where as in PDC popular data is skewed on to few of the data disks itself. So, to make a fair comparison between PDC, MAID and UREED we following disk arrangements.

For UREED -1 it has 15 data disks and 5 buffer disks where as UREED – 2 has 20 data disks and 5 buffer disks. MAID -1 and MAID -2 follow the same disk arrangement as UREED -1 and UREED -2 respectively. Since PDC does not require any buffer disks all the disks in PDC -1 are data disks and we used 20 data disks for PDC -1 and 25 for PDC -2

We conducted the experiments and collected the results for all the file servers and also for an energy oblivious server, referred as EO. EO does not allow disks to power down at any point of time. To compute the percentage energy savings of the file servers we compared it with the results of EO.

5.5.2 Simulation results

In this section we present our simulation results. We evaluated the approach by running the simulation for each file server characteristics. We studied the effect of different parameters on the energy saving of UREED, PDC and MAID. These are

then compared against the EO values to determine the percentage energy savings.The values for our workload characteristics were chosen based on the parameters where PDC and MAID perform well.

Fig 5.1 shows the energy savings of the file servers when different file request rates are imposed on them. These energy savings are computed in comparison against EO.

It is clear from the figure that low request rate produces high percentage of energy savings, as expected. The maximum energy savings are substantial when the savings are compared with EO. As per the figure the energy savings of MAID are much less compared to UREED and PDC. It can be observed from the figure that PDC-1, PDC – 2, UREED – 1 and UREED - 2 energy savings are close to each other, but the PDC algorithm does take reliability in to consideration. UREED achieves the energy savings higher or same as PDC and definitely higher compared to MAID along with the reliability of the file server.

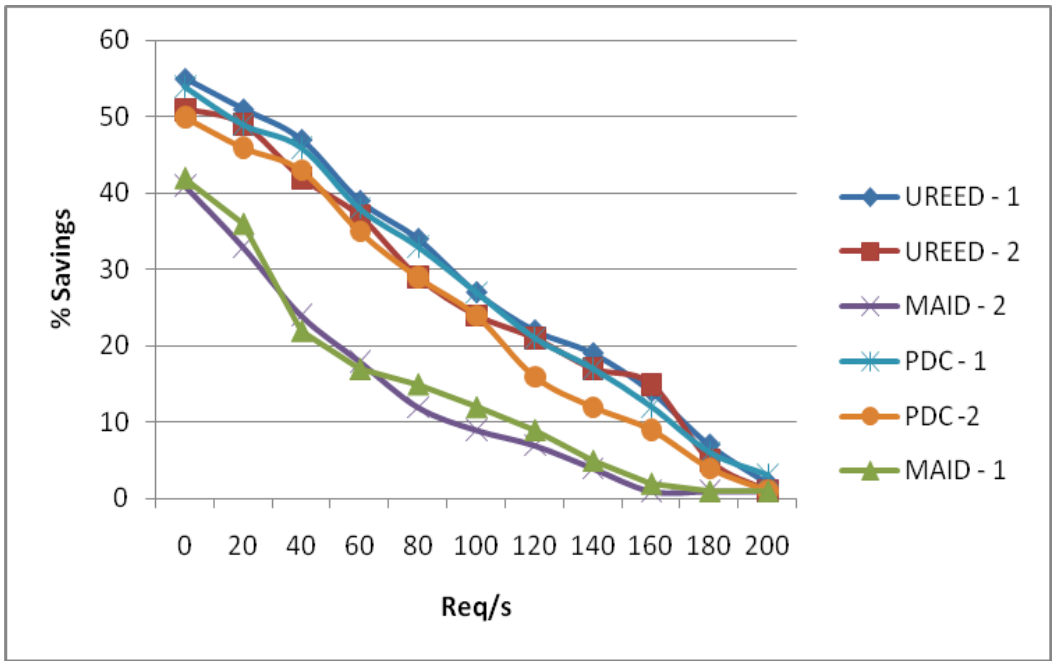

Fig 5.1 Energy savings per file request rate

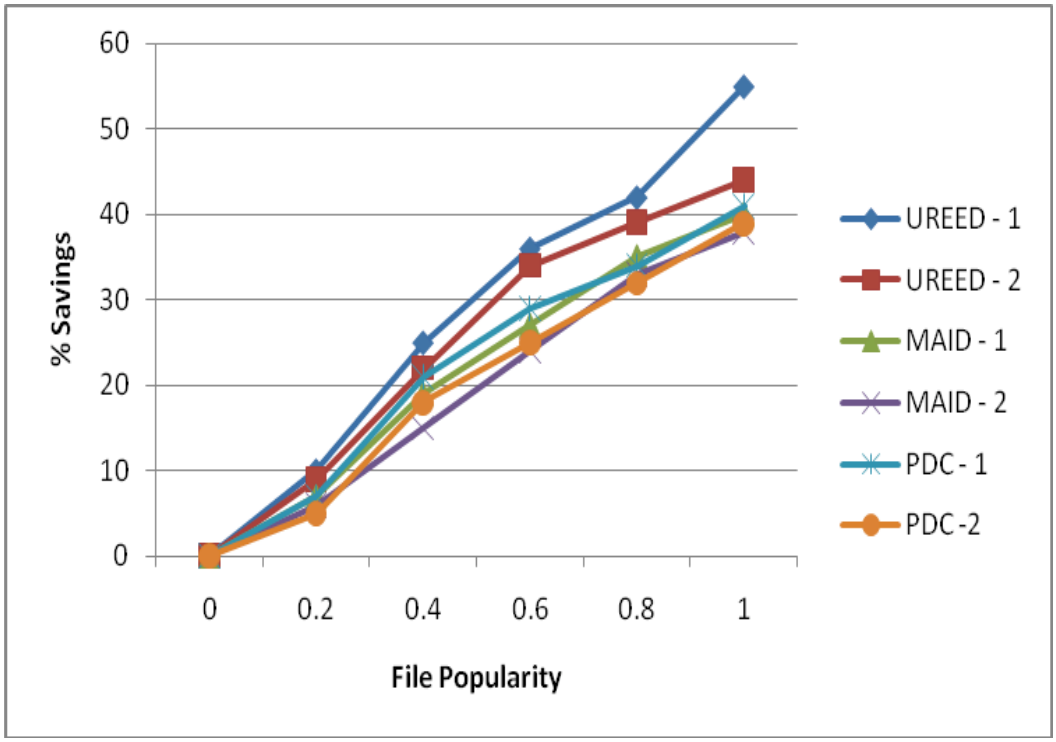

Fig 5.2 Energy savings per file popularity

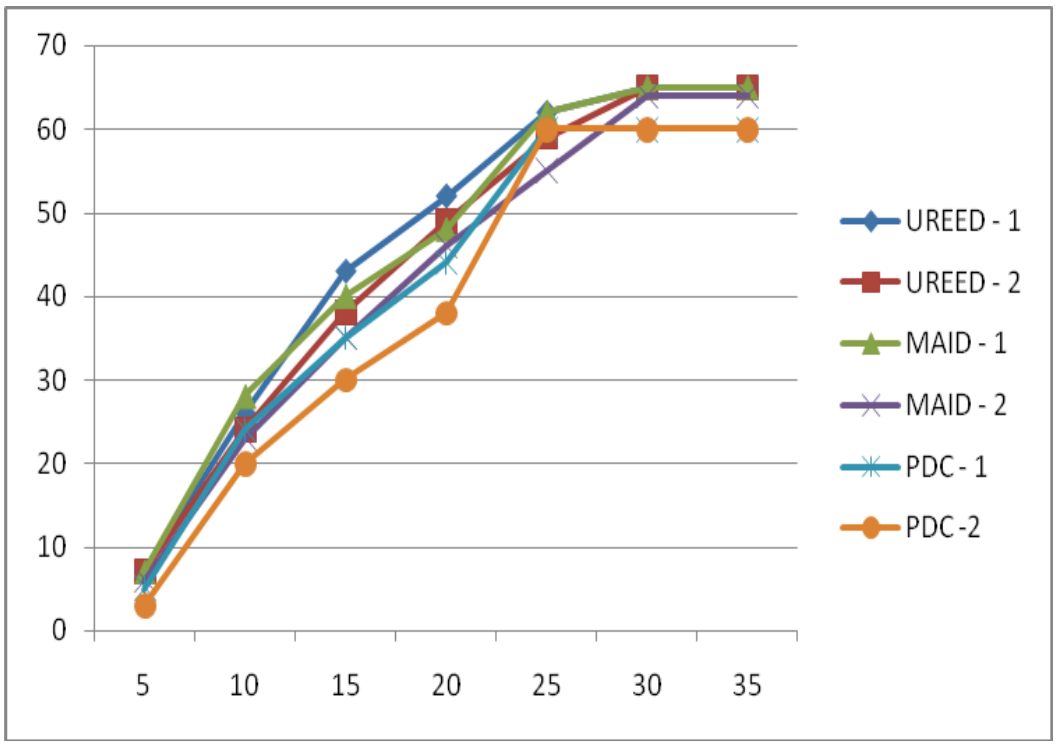

Fig 5.3 Energy savings per number of disks

Figure 5.2 shows the energy savings of the different file servers with the increase of file popularity. File popularity changes with the variation in Zipf coefficient. The figure shows that for all the file servers the energy savings increases with the increase in popularity of the file. The reason being, all the file servers use the technique of skewing the popular data, so with the increase of popularity of the files there is very high probability that they are available in buffer disks for service, which reduces the overhead of waking up the data disks. The energy savings here are very close to each other, but when observed closely it can be noticed that the energy savings on UREED are a little higher compared to the other two file servers.

Figure 5.3 shows the energy savings of the file servers with respect to the number of disks in the file server. It should be noted here that the increase of number of disks does not have an effect on the total file server storage. That is, the file server storage is kept constant but the files stored on the individual disks become fewer. More over the increase in disks does not change the number of buffer disks. From fig 5.3 it can be observed that with the increase of number of data disks the energy savings also increases, as expected. The fewer number of files on the data disks decreases the accesses directed to them and thus operate in low power modes producing higher energy efficiency.

We used the synthetic workload to generate the simulation results provided above. It is important to obtain confidence on our simulator. So we calculated the total energy consumed values for our simulation and also for the mathematical model. Table 5.2 compares the energy values for the three different file servers calculated using the simulation and model. It shows the energy consumption values of each server using the simulation match closely with that of the model (with 8% to 13%

error). This validates the simulator and gives the confidence that these values can be used to illustrate the teal world conditions.

Table 5.2 Simulator validation with mathematical model (energy in joules)

Description	Total Energy Consumed		
	Sim	Model	Error (%)
UREED-1	170294	192145	11.37214
PDC-1	197896	215482	8.161239
MAID-1	181568	210245	13.6398

Table 5.3 illustrates the failure rates of all the file servers. Different disks ages have different failure rate because of the variation in the reliability curve, which changes the values of the safe utilization zone. As opposed to PDC and MAID, UREED has very minor failure rate figures. It can be stated that the UREED algorithm not only saves energy but also significantly improves the reliability of the disks.

Table 5.3 Failure rate of the file servers

Disk Age	UREED - 1	UREED – 2	MAID - 1	MAID - 2	PDC - 1	PDC -2
0.5	0.027	0.01	9.01	8.027	17.023	16.239
1	3.082	2.372	25.634	23.872	37.123	35.345
2	12.123	9.049	30.592	24.674	39.198	36.876
3	9.673	7.542	27.362	25.984	40.23	39.384
4	4.927	3.128	34.721	32.165	36.234	38.123
5	6.096	6.91	29.121	27.901	35.654	33.703

5.6 Summary

In this chapter, we proposed a technique to conserve energy and improve reliability of the disks called as UREED. UREED skews the popular data onto the buffer disks while keeping the non-popular data in the data disks. This allows the data disks to stay in low modes because of the less frequency of access. Buffer disks stay active most of the time to serve the request. By switching the data disks to low power modes the URRED technique conserves energy. Our concentration is on energy efficiency and reliability of the disks, so to achieve reliability the UREED technique operates only in the safe utilization zone. UREED algorithm is compared with two existing energy conservation techniques known as PDC and MAID. PDC, popular data concentration is technique where the popular data is skewed onto few of the disks. Data migration is the technique used in PDC. MAID, massive arrays of idle disks, also skews popular data, but unlike PDC it skews the popular data on to buffer disks. Instead of data migration data copying is used in MAID. We conducted extensive simulation experiments using the synthetic traces and generated the result for all three file servers namely UREED, PDC, and MAID. From the results it can be observed that UREED conserves more energy when compared with the PDC and MAID while achieving significant reliability improvement.

Chapter 6

Integrating Security and Reliability in Real-time Embedded Systems

In the previous three chapters, we have addressed the energy conservation and reliability issues for large scale storage systems. In this chapter we divert our attention towards the real time embedded systems and addressed the issues of security and reliability of real time systems. We implemented the confidentiality and integrity services to provide security to the real time system and used check pointing strategy to counter the reliability issues. Next we proposed techniques to integrate both security and reliability for real time systems.

The rest of this chapter is organized as follows. In section 6.1, we present the motivation of this study. Section 6.2 illustrates check points for fault recovery. In section 6.3, we demonstrate the security implementation. Integration of fault recovery along with security for real time embedded systems are described in section 6.4. Performance evaluation is presented in section 6.5. Finally, section 6.6 will summarize the primary contributions of this chapter.

6.1 Motivation

In the recent past, mandatory security requirements and fault recovery have become critical criteria for most real-time embedded systems such as military aircraft control systems and online banking. Although many conventional fault recovery or security approaches were investigated and applied to real-time embedded systems, most existing schemes only addressed either security demands ignoring the fault recovery requirements or vice versa. To bridge this technology gap in real-time embedded systems, in this chapter we propose a way to integrate fault recovery and security services in real time embedded systems.

In this chapter, we exploited the slack that exists in a schedule to integrate security and fault recovery in to the task. Checkpoint insertion strategy is used to enable recovery from failures. Confidentiality and Integrity, the most popular among the wide variety of security services available are considered for the security overhead model. The novel integration of security and fault recovery makes it possible to implement next-generation real-time embedded systems with high reliability and quality of security.

6.2 Checkpoints for Fault Recovery

Real-time applications can be generally classified into two types: hard real-time applications and soft real-time applications. Systems in which a missed deadline results in disastrous consequences are termed hard real-time systems, while systems in which a missed deadline results in degraded performance, but with no extreme consequences, are termed soft real time systems [87]. In this research, we consider

hard deadlines in real-time applications. Hence, our approach can be readily applied to both hard and soft real-time applications. The basic idea over here is to recover hard real-time embedded systems from transient failures by inserting checkpoints. It should be noted that checkpoint insertions can be accomplished in a uniform or non-uniform manner. We implemented uniform check point approach in this chapter.

6.2.1 Real-time application

A real-time application in this study is envisioned as a set of tasks, which has to be completed before its specified deadlines to accomplish an overall mission. We assume a real-time task'*t*' has a deadline'*d'* derived from hard real-time constraints. It is assumed that task*'t'* takes '*C'* number of CPU cycles for its worst-case execution. Cycle consumption of memory references varies with processor speed. Hence, '*C'* is considered independent of the processor speed. Abundant research has been done to prove the degree of pessimism in the definition of '*C'* [55].

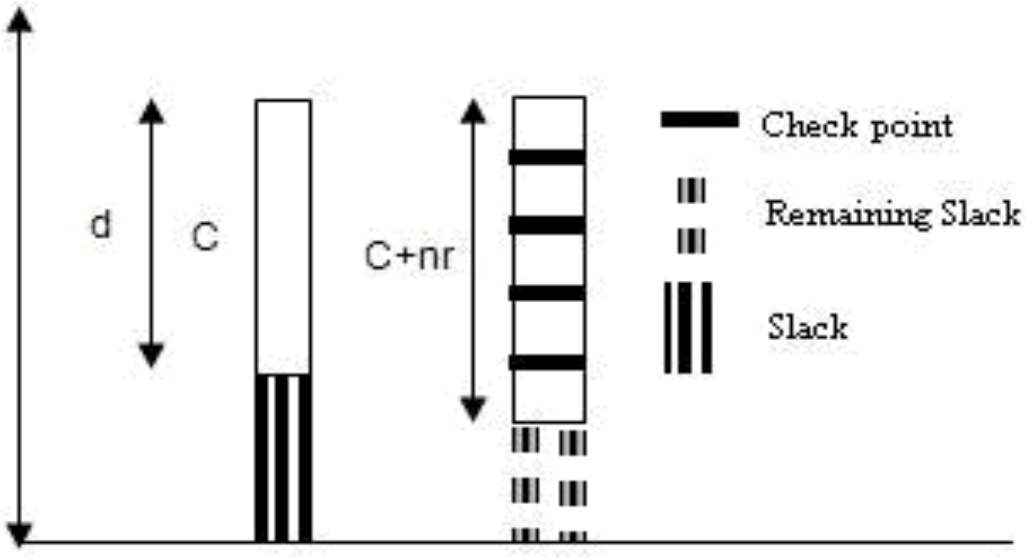

Figure 6.1 Fault Recovery Model

The time remaining to reach deadline‘*d*’ after considering the worst-case execution time ‘*C’* of the real-time task is the slack time denoted as ‘S_t’. This slack time ‘S_t’ can be used to recover from transient faults in a real-time embedded system. It is to be noted that slacks may be static or dynamic. Specifically, a static slack time is obtained when deadlines and worst-case execution times are known a priori. In contrast, dynamic slack times can be derived only at runtime executions. Slack times considered in this work are static slacks (here after referred to as slacks).

6.2.2 Fault recovery model

If a fault occurs during the execution of a task then the entire task has to be re-executed but the slack available may not be sufficient for such an execution. Hence, checkpoints [49] are inserted in to a real-time task to efficiently recover the system from any transient failures. If there is a failure in the execution of the task, then, the failure is detected before the placement of the next checkpoint and it is easy to re-execute the task from the previous checkpoint where the failure occurred [26]. Transient failures [16] during the course of the task’s execution can be readily detected by applying existing fault-detection techniques. Without loss of generality, it is assumed that no more than one transient failure might occur during the execution of the task. Similar assumptions were imposed in other studies [55].

Our goal, of course, is to efficiently leverage slack time ‘S_t’ to improve quality of security and to provide fault recovery. The overhead involved with generating checkpoints is modeled as *‘r’*. It is assumed that in task*‘t’, ‘n’* number of checkpoints are inserted to recover the system from failure, if there exists any. Therefore, the total time spent in creating all the checkpoints and doing the corresponding diagnosis is

expressed as *'nr'*. If there exists a fault, then it takes (C/*n*+1) time for a potential roll back. In a previous study, Melhem *et al.* showed that it takes *(C/n)* time for a potential rollback to recover from a fault if there exists any [55], where they inserted the first checkpoint of *n* at the very start of the execution of the task. We argue that there is no significant use in inserting a checkpoint at the very beginning of the task and hence we considered another effective strategy where the time spent in processing a potential roll back is *(C/n+1)*. We can easily prove the correctness of this expression by showing that a fault cannot occur before the execution of a task. This argument indicates that, the strategy we used does not insert checkpoint at the very beginning of each task. Hence, a task *t* should satisfy the equation (6.1) to execute before the deadline guaranteeing the fault recovery.

$$d \geq C + (n \times r) + (C/(n+1)) \tag{6.1}$$

It is intuitive to illustrate that, the more the number of checkpoints made in a real time-task, the more the task is fault-tolerant. However, it is imperative to carefully choose the number of checkpoints, because there is a chance that the overhead might exceed slack time 'S_t', which makes it infeasible to finish the task before its deadline.

Simple mathematical manipulation of inequality (6.1) yields

$$n = (x \pm \sqrt{(x^2 - (4r(2C - d)))})/2r \tag{6.2}$$

$$\text{where } x = d - C - r \tag{6.3}$$

Using the Eq. (6.2) we can calculate the optimal value of *n*. In cases where reliability is the only performance goal, Eq. (6.2) can be applied to obtain an optimized *n*. In our experiments, we make use of Eq. (6.2) to optimize *n* resulting

maximal reliability of a real-time task while guaranteeing the completion of the task before its deadline.

6.3 Security Implementation

Security has become an important requirement for modern real-time embedded systems [52]. There are wide varieties of security mechanisms that can be applied to real-time applications. However, improving security inevitably leads to high overheads. Thus, it is difficult if not impossible to implement all security mechanisms available to a real-time task. Therefore, vital security mechanisms have to be chosen carefully to provide sufficient level of security for real-time tasks.

As confidentiality and integrity are two major security issues that real-time tasks may encounter, we describe the overhead model for these two types of security services [86][4][66]. It is worth noting that our approach can readily incorporate other types of security services like availability [87], authentication etc., to achieve higher security.

6.3.1 Confidentiality model

Confidentiality encounters snooping, which is unauthorized interception. Confidentiality is achieved by the implementation of cryptographic algorithms. All the data of a real-time application, which has to be protected from unauthorized interceptions, must be enciphered in one way or another.

Many cryptographic algorithms are available to implement the confidentiality [6][42][28]. The best candidate cryptographic algorithm is the one consuming less

time while ensuring maximal security. There is a rich set of cryptographic algorithms that can be used for confidentiality. The most common ones are used in our research.

Table 6.1 Cryptographic Algorithms for Confidentiality

S.No	Cryptographic Algorithms	sl_i^c : SL Security	μ_i^c :KB/ms
E1	SEAL	0.08	168.75
E2	RC4	0.14	96.43
E3	Blowfish	0.36	37.5
E4	Knufu/Khafre	0.40	33.75
E5	RC5	0.46	29.35
E6	Rijndael	0.64	21.09
E7	IDEA	1.00	13.5

Table 6.1 depicts the cryptographic algorithms that are used in this research to attain confidentiality [72]. 'Sl_i' signifies the security level assigned to each algorithm in the range from 0.08 to 1. The strongest encryption algorithm IDEA is assigned the maximum-security level 1 and its performance is 13.5 KB/ms. The security levels for the other algorithms are normalized by applying the following equation [74].

$$sl_i^c = 13.5 / \mu_i^c, \; where 1 \le i \le 7 \tag{6.4}$$

Assume that 'l_i' is the amount of data whose confidentiality has to be guaranteed for task 't_i'.

$$l_i / \mu_i^c = P \tag{6.5}$$

Where 'P' is the amount of time the corresponding algorithm of '$\mu^c{}_i$' takes to encrypt the data with size 'l_i'. Hence, the appropriate cryptographic algorithm can be selected based on the amount of time that can be allocated for the security mechanisms. $\sigma(S_c)$ is the function used to map a security level to its corresponding encryption methods performance [72].

For a task to execute within its deadline while achieving the confidentiality, it should satisfy the following condition.

$$d \geq C + (l_i^c / \mu_i^c), \; where 1 \leq i \leq 7 \tag{6.6}$$

6.3.2 Integrity model

Integrity encounters alterations, which are unauthorized changes of information. Integrity can be achieved by implementing hash functions. Use of hash functions ensure that no one can modify or tamper data and applications without authorization. The seven most commonly used hash functions coupled with their performance are shown in Table 6.2 Each function is assigned a security level in the range from 0.18 to 1. The strongest and yet slowest hash function Tiger is assigned a maximum-security level of 1 and its performance is 4.36 KB/ms.

Table 6.2 Hash Functions for Integrity

S.No	Hash Functions	s_i^g :Security Level	$\mu^g(s_i^g)$:KB
H1	MD4	0.18	23.90
H2	MD5	0.26	17.09
H3	RIPEMD	0.36	12.00
H4	RIPEMD-128	0.45	9.73
H5	SHA-1	0.63	6.88
H6	RIPEMD-160	0.77	5.69
H7	Tiger	1.00	4.36

The security levels for other hash functions are calculated by equation (6.7)

$$sl_i^g = 4.36 / \mu_i^g, 1 \leq i \leq 7 \tag{6.7}$$

Assume that 'l_i' is the amount of data whose integrity must be achieved for task 't_i'.

$$l_i / \mu_i^g = Q \tag{6.8}$$

Where 'Q' is the amount of time the corresponding algorithm of 'μ^g_i' takes to hash the data with size 'l_i'. Hence, an appropriate integration algorithm can be selected based on the amount of time that can be allocated for achieving the integrity of the task. $\sigma(S_i)$ is the function that maps a security level to its corresponding hash function.

For a task to attain integrity without missing the deadline, it should satisfy the following equation

$$d \geq C + (l_i^g / \mu_i^g) \tag{6.9}$$

If the above equation can be satisfied, then it proves that the maximum possible integrity of a task is attained without missing the deadline.

Equations (6.6) and (6.7) can be used when a particular security mechanism among confidentiality or integrity has to be attained. Since the deadline is guaranteed, these formulae are very effective.

6.4 Integration of Fault Recovery with Security Mechanisms

In the previous sections, we have described a way to provide optimal security mechanism and the fault recovery model along with their overheads. In this section, we illustrate how to integrate fault recovery with the security mechanisms to enhance the overall performance of real-time tasks.

We first delineate the integration of confidentiality with fault recovery. Given a real-time task with its timing constraint, the aggregate of the overheads should be less than or equal to the slack S_t. The implementation of fault recovery in addition to security mechanisms is possible only if the following inequality is satisfied

$$d \geq C + (n \times r) + (C/(n+1)) + (l_i / \mu_i) \tag{6.10}$$

The above expression guarantees both reliability and confidentiality for real-time tasks. If integrity is considered instead of confidentiality, then expression (6.10) can be rewritten as

$$d \geq C + (n \times r) + (C/(n+1)) + (l_i / \mu_i) \tag{6.11}$$

Simple manipulation of (6.10) leads to

$$\mu_i^c = \frac{l_i^c}{d - C - (n \times r) - (C/(n+1))} \tag{6.12}$$

Using the above formula, we can determine the optimal security level for a real-time task while guaranteeing high reliability, and without violating the task's deadline. By mapping the value of 'μ_i^c' to the corresponding encryption algorithm using the $\sigma(S_c)$ function, the feasible security can be implemented.

Now, by manipulating (6.11) we have the following equation

$$\mu_i^g = \frac{l_i^g}{d - C - (n \times r) - (C/(n+1))} \tag{6.13}$$

The mapping function $\sigma(S_i)$ is employed to map the value of '$\mu^g{}_i$' to the corresponding hash function. The equations (6.12) and (6.13) are used to identify the most appropriate security levels for real-time tasks. To use the above formula, one has to predetermine *n*. Consequently, we have the flexibility to choose the level of reliability. If a real-time task is security critical, then we can choose a lower value for *n* to achieve higher security. On the other hand, if the real-time task is not security critical, then, by choosing a higher value for *n* we can achieve high level of reliability for the task while satisfying the timeline requirements. The equation given below is

used to handle this special case. Fig (6.14) yields two values for *'n'*. The positive value should be considered.

$$n = \frac{(x-r) \pm \sqrt{(x-r)^2 - (4r(C-x))}}{2r} \tag{6.14}$$

$$\text{where} \quad x = d - C - (l_i^c / \mu_i^c) \tag{6.15}$$

$$\text{or} \quad x = d - C - (l_i^g / \mu_i^g) \,. \tag{6.16}$$

For confidentiality Eq. (6.15) has to be used, whereas for the case of integrity, Eq. (6.16) has to be considered. Using Eq. (6.14), we can maximize the reliability to the possible extent if security levels are known a priori based on the security requirement of real-time tasks. In such case, the value of *n* is calculated. Here, the reliability is maximized while achieving the basic security without crossing the deadline Eq. (6.14).

Now, we are in a position to address the issue of having both optimal security and reliability. In other words, the security and reliability of a real-time task can be optimized simultaneously by choosing the optimal value of security level and determining the optimal value of the number of checkpoints. An alternative approach to achieve this goal is, to first calculate the optimal number of checkpoints *'n'* followed by deciding the optimal value of security level. In doing so, both security and reliability can be efficiently optimized.

6.5 Performance Evaluation

In this section, we evaluate the effectiveness of the proposed approach of integrating the fault recovery (through checkpoints) and quality of security (measured as security levels). To show the strength of our scheme, we conducted extensive

experiments. It is to be noted that worst-case execution times are derived from real-world applications.

6.5.1 Confidentiality versus reliability

In Fig 6.2 (a), the change in the security levels with the increase in the number of checkpoints is observed for three different data sizes of 100MB, 150MB, 200MB, respectively. The results indicate that for a given task, initially when the number of checkpoints increases, the security level also increases. The reason for this phenomenon is that, when the number of checkpoints is too small, the amount of time involved in the recovery, if there exists a fault, is much higher because of the low frequency of check pointing. When the number of checkpoints are increased, the recovery time decreases leaving a larger slack time for security. Hence, the security level increases with the increase in the number of check points, but after a certain increase, it remains constant and then decreases. The point where it stays constant is the optimal security level with optimal reliability. This indicates that by selecting the number of checkpoints carefully we can achieve optimal security and reliability.

Fig. 6.2(b) reveals the relationship between the rho (to simplify the presentation, C/d is denoted as Rho) and security levels. The higher the amount of Rho, lesser the amount of slack time is left. Fig.6. 2(b) shows the impact of decreasing slack time on the security level. As expected, the security level of the tasks decreases as the slack decreases. After a certain period, even the minimum security cannot be achieved because of the limited slack.

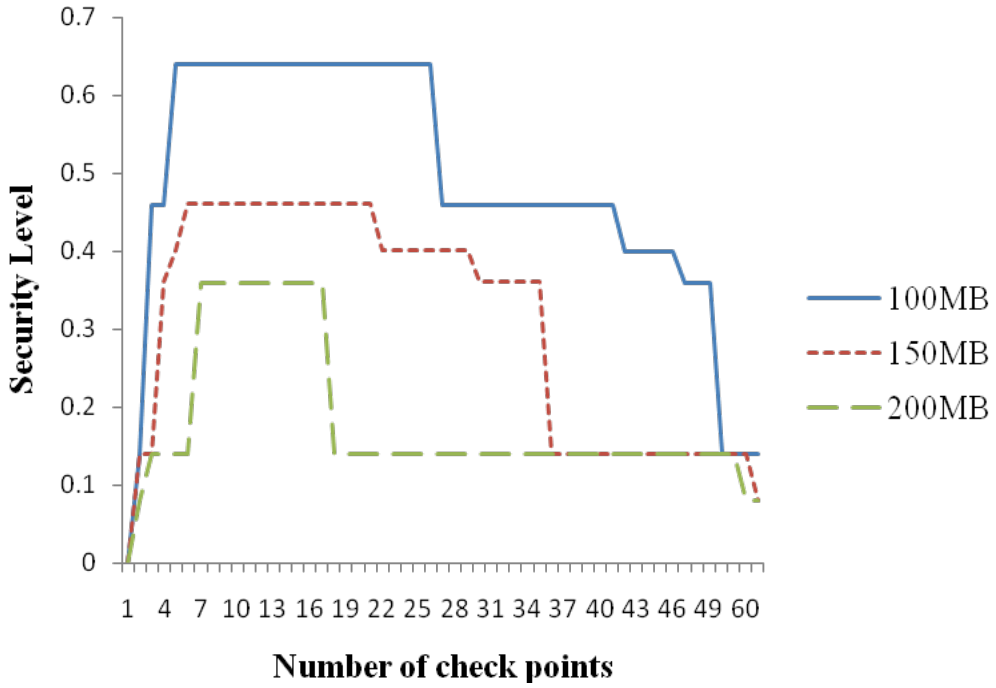

Figure 6.2(a) Number of checkpoints vs. Security Level (Confidentiality)

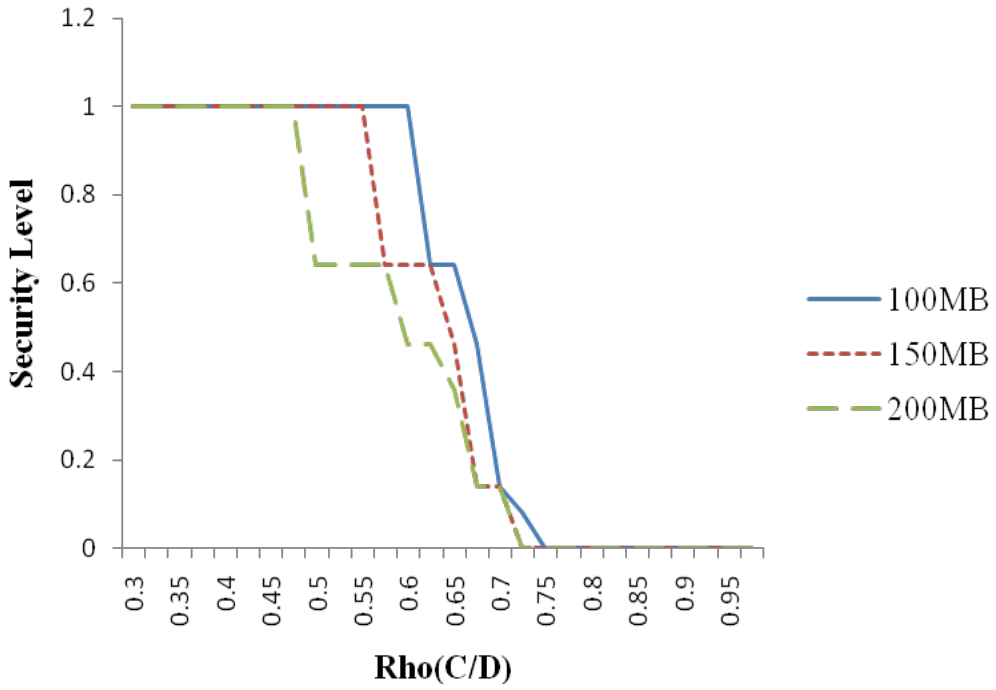

Figure 6.2(b): Rho (C/d) vs. Security Level

In general, a task, which has a prolonged deadline, can be made more secure than other tasks with shorter deadlines. The same phenomenon is observed for three different tasks with data size of 100MB, 150MB, and 250MB. For any given task, as the deadline increases, the slack provided to accommodate security level also increases and hence, the security level increases. The Fig. 6.2(b) proves the correctness of the above argument.

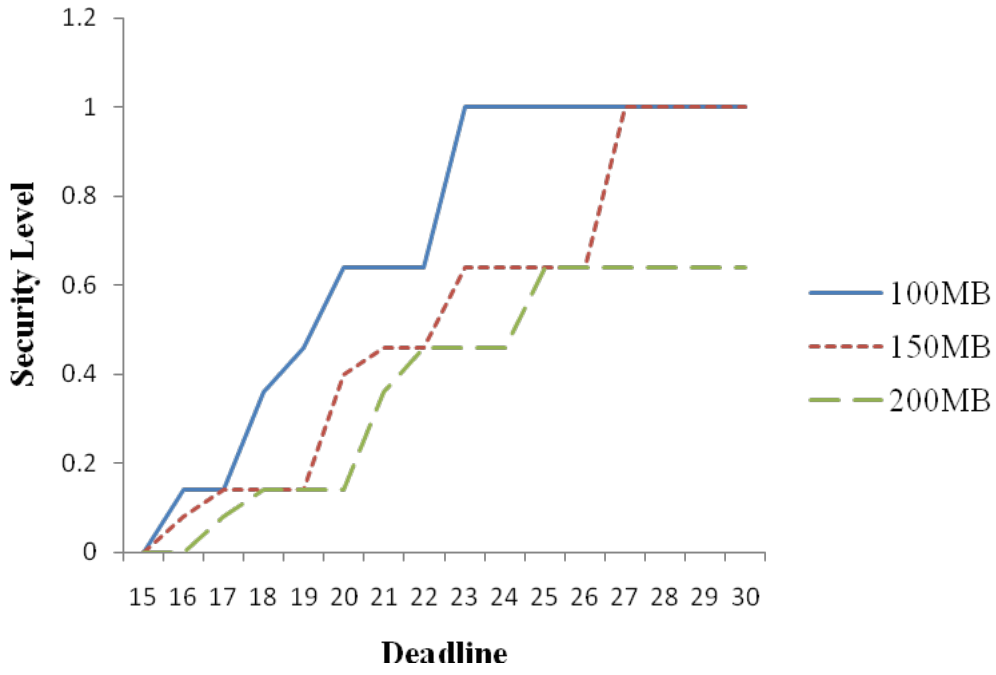

Figure 6.2(c): Deadline vs. Security Level

It can be observed from Fig. 6.2(c) that, the security level can be increased or decreased by carefully choosing the number of checkpoints. If a task is very security critical, the minimum number of checkpoints can be chosen which still guarantees the basic reliability requirement. However, if security is not a major concern, the number of checkpoints can be increased for better fault recovery, which in turn leads to a higher level of reliability.

6.5.2 Integrity versus Reliability

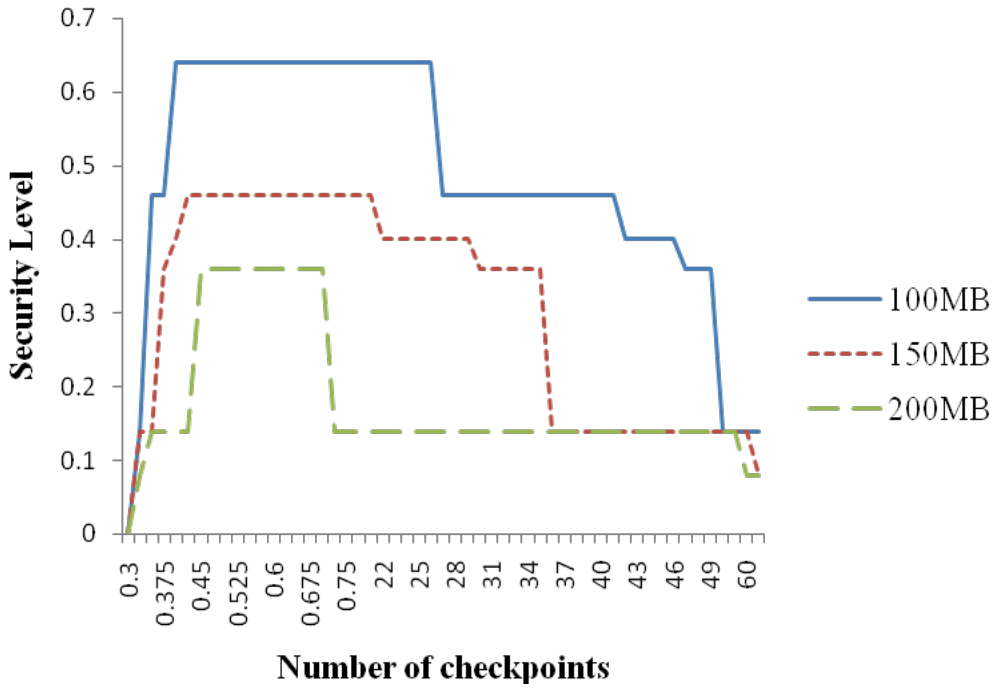

Figure 6.3(a) Number of Checkpoints vs. Security Level (Integrity)

Fig. 6.3 (a) is similar to Fig. 6.2 (a) except that the integrity of a real-time task is considered instead of confidentiality. In this set of experiments, the hash function techniques are used instead of the cryptographic algorithms. Even here, the security level increases along with the increasing checkpoints. The reason for this phenomenon is same as that mentioned for Fig 6.2(a). The point where the security remains constant is the optimal security and reliability zone.

Fig 6.3(b), security decreases as the Rho increases because of the decreasing slack time. Furthermore, the increasing data size decreases the security level. This result is expected because of the overhead of time spent in securing data using the corresponding algorithms.

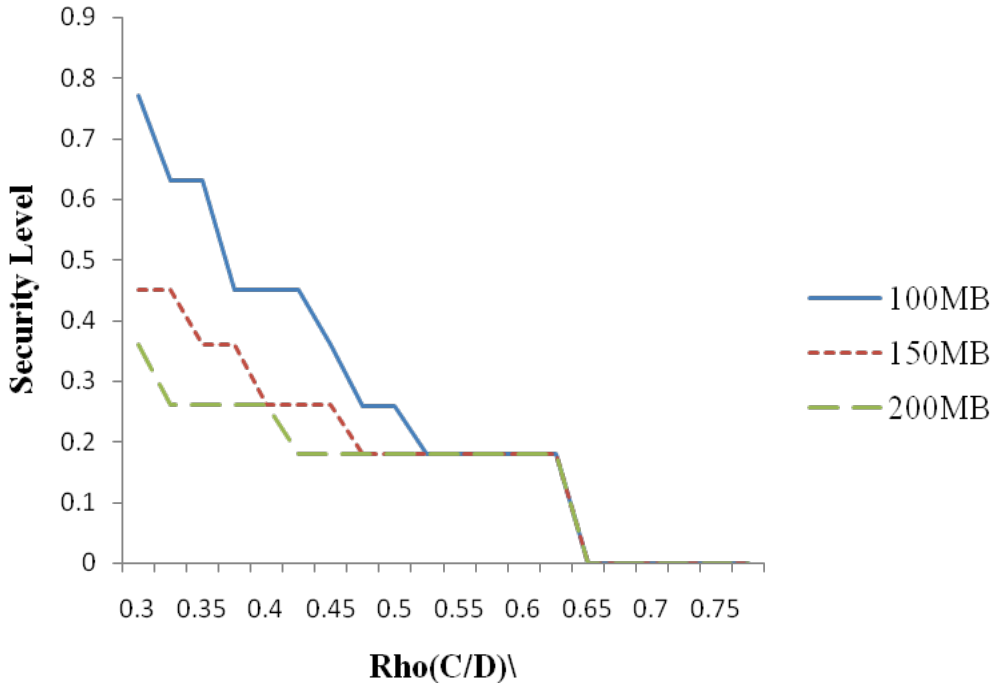

Figure 6.3(b): Rho(C/D) vs. Security Level(Integrity)

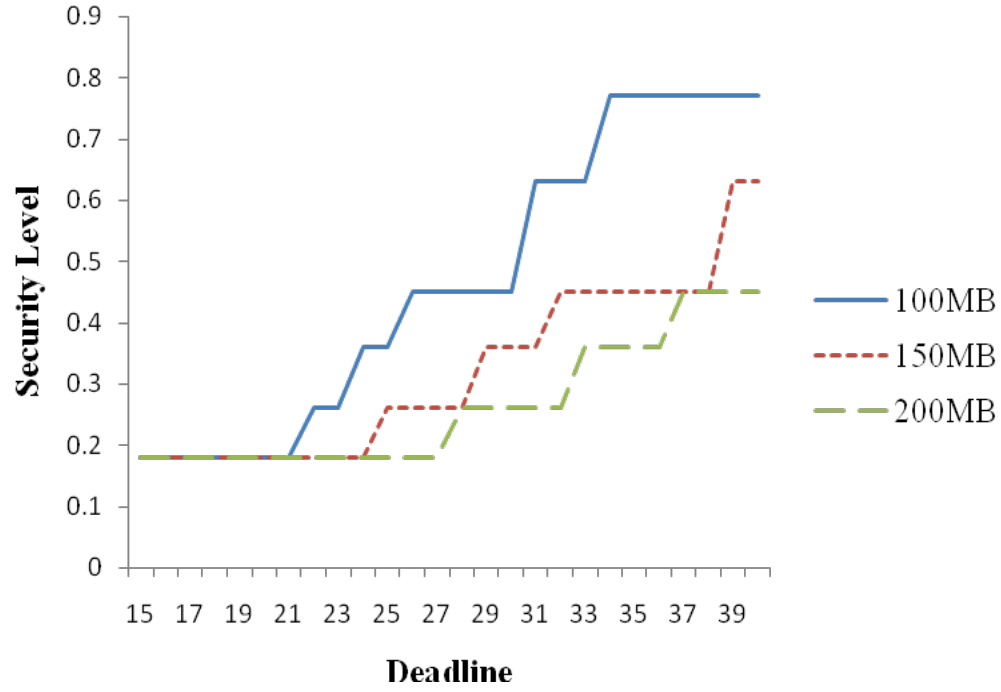

Figure 6.3(c) Deadline vs. Security Level (Integrity)

Fig 6.3(c), the security level goes up as the deadline increases because of the abundant slack time available to accommodate the security along with fault recovery. The results are similar to Fig. 6.2(c). We can observe from the above figures that, the effect on security levels is same for either confidentiality or integrity mechanisms.

6.5.3 Security versus reliability

Integrity Fig. 6.4 shows the relation between reliability and security when confidentiality is considered as the security mechanism.

Fig. 6.4 shows results for three different data sizes. For the smaller data set, the levels of security and reliability are high because of large available slack times. As the size of the data increases, the security and reliability levels are reduced. Again, we observe that higher security levels lead to low reliability with small number of checkpoints. Basic security levels give rise to high reliability with a large number of checkpoints. By using Figure 6.4, one can derive the optimum values for the number of checkpoints and security levels.

Fig. 6.5 shows the relation between security and reliability when integrity is implemented as the security mechanism. The trend is very similar to the one plotted in Fig. 6.4. Now, we can draw the conclusion that security and reliability are inversely proportional to each other. In other words, with the increase of reliability, security will decrease and vice versa. It should be noted here that, determining optimal values of security levels and number of checkpoints is the underpinning of this research.

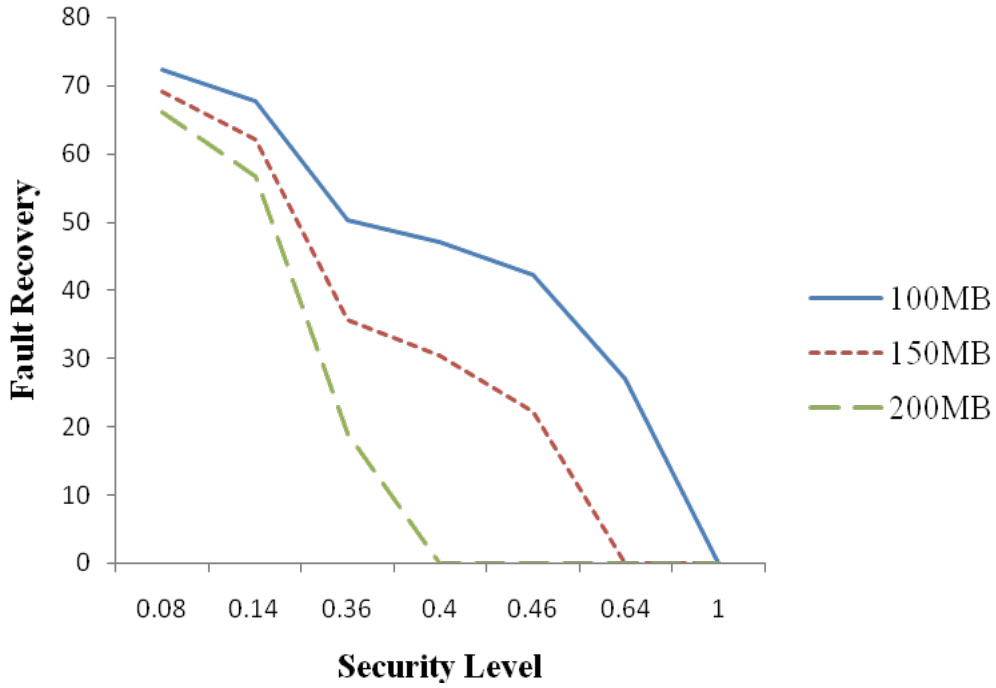

Figure 6.4 Security Level (Confidentiality) vs. Fault Recovery (Checkpoints)

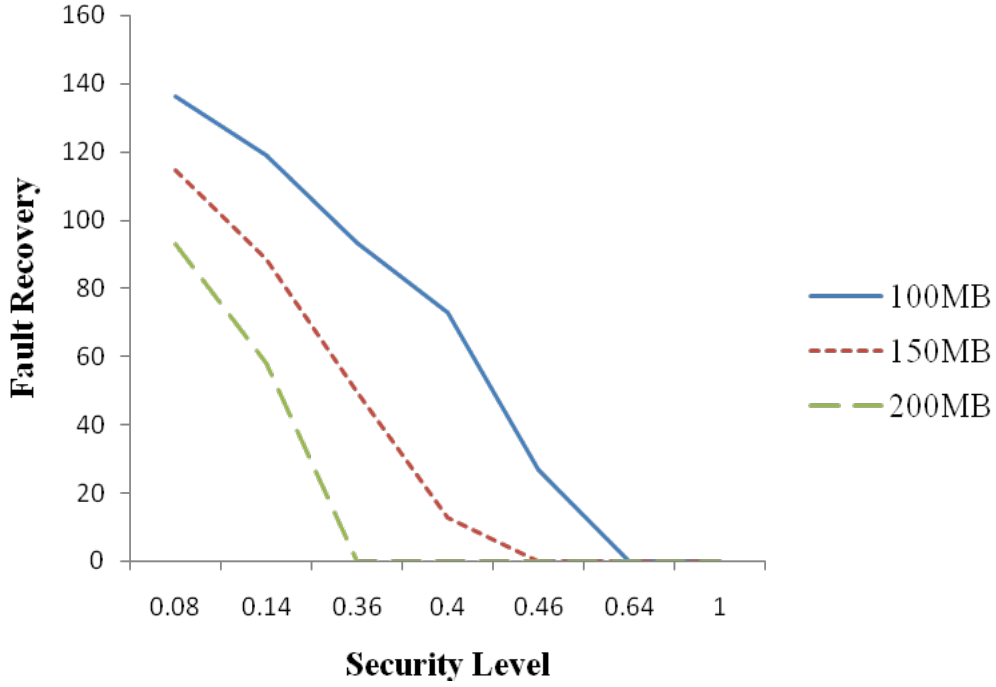

Figure 6.5 Security Level (Integrity) vs. Fault Recovery (Checkpoints)

6.6 Summary

In this chapter, we studied an approach to integrate fault recovery and quality of security. In the proposed scheme, we considered fault recovery and two types of security services - confidentiality and integrity services. This research was undertaken mainly in the following three steps. First, the overhead models for the two kinds of security services coupled with fault recovery overhead model were presented. The overheads imposed by an array of security mechanisms were quantitatively computed. Next, we integrated a fault recovery mechanism with quality of security. To facilitate the presentation of the integration, we introduced the mathematical notation for the implementation of such an innovative integration. Third, we proposed a novel way of maximizing quality of security with minimum reliability. Additionally, we described a means of optimizing reliability while providing basic quality of security. Our approach aims to maximize quality of security of a real-time embedded system while providing checkpoints to efficiently support fault recovery. Experimental results showed that, compared to conventional schemes where basic security is provided; our approach improves the quality of security significantly over the conventional approaches.

Chapter 7

Interplay of Fault Recovery and Quality of Security using non-uniform check points

In the previous chapter, we have addressed the issue of integrating security and fault recovery in real time embedded systems. We used uniform check pointing policy for fault recovery and implemented confidentiality and integrity services for security. Work proposed in the previous chapter was further extended in this chapter by using non-uniform check pointing policy for fault recovery and also integrating security services with the non uniform check pointing policy.

In this chapter, we first propose non uniform check pointing policy for real time embedded systems and then we propose two approaches to integrate the non uniform policy with the previously proposed security mechanisms.

The rest of this chapter is organized as follows. In section 7.1, we present the motivation of this study. Section 7.2 illustrates non uniform check point strategy for fault recovery. In section 7.3, we demonstrate the security implementation and also integration of fault recovery along with security for real time embedded systems.

Performance evaluation is presented in section 7.4. Finally, section 7.5 summarizes the primary contributions of this chapter.

7.1 Motivation

Real time applications such as military aircraft flight control systems and online banking systems are critical with respect to security and reliability. Although many conventional fault-tolerant or security approaches were investigated and applied to real-time systems, most existing schemes only addressed either security demands ignoring the fault-tolerant requirements or vice versa. To bridge this technology gap in real-time systems, we propose a way to integrate security services (confidentiality and integrity services) and reliability (uniform and non-uniform checkpoints) for security-aware and high-reliable real-time systems. Slack time exploitations interact in subtle ways with security in regards to the placement of checkpoints. Uniform check point placement strategy is straight forward, where the check points are inserted in to the task periodically. This approach is presented in the previous chapter. Unlike uniform check point strategy the check point's placement is uneven in non uniform check point strategy. To improve the reliability further in this chapter we proposed the non uniform check point policy.

In this chapter, we exploited the slack that exists in a schedule to integrate security and fault recovery in to the task. The issue of data security is addressed in two ways. The first method introduces the security for the entire data set at once, whereas in the second method the data set is divided into n uneven sections and each section is separately secured. Initially, basic security services are considered in a real-time task depending on slack time availabilities. The quality of security is increased

gradually for the rest of the real-time task. If a failure occurs, at that point the security level remains unchanged due to the limited slack time

7.2 Checkpoints for Fault Recovery

As mentioned in chapter 6, checkpoint insertions could be accomplished in a uniform or non-uniform manner. We used non- uniform checkpoint strategy in this chapter

7.2.1 Fault tolerant model for non - uniform check point strategy

Similar to the uniform check point strategy explained in chapter 6, the overhead involved with generating checkpoints is modeled as *r*. It is assumed that in task *t, n* checkpoints are inserted to recover the system from failures if there exist any.

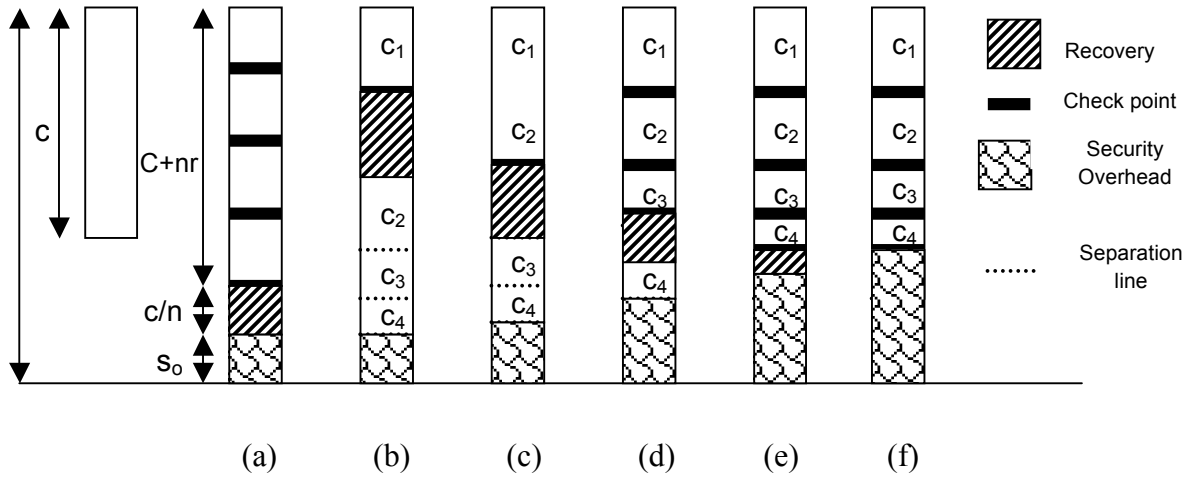

Fig.7.1 Uniform and Non-uniform distribution of check points with security over head

Therefore, the total time spent in creating all the checkpoints and to do the corresponding diagnosis is expressed as $n \times r$, but we did consider that no more than one fault can occur during the task's execution so if a fault is detected, then thereafter it is not necessary to insert checkpoints.

Hence, the overhead involved with check points relies on where the fault occurred. Fig 7.1 shows the uniform and non uniform distribution of check point. As the approach we considered here is non-uniform, the *n* check points are placed in the task *t* such that the *C* cycles of *t* are divided into *n* sections based on the below formulae

$$nx + (n-1)x + (n-2)x + \ldots + 2x + 1x = C \tag{7.1}$$

$$x = \frac{2C}{n(n+1)} \tag{7.2}$$

After calculating the value of *x* from Eq. (7.2) checkpoints are inserted in a way such that

$$C_1 + C_2 + C_3 + \ldots + C_{n-1} + C_n = C \tag{7.3}$$

$$\text{Where } C_1 < C_2 < C_3 < \ldots < C_{n-1} < C_n \tag{7.4}$$

The main idea behind this approach is that at the beginning of the execution the task hasn't consumed much of available slack and therefore, low frequency of checkpoints is possible because most of the slack is available to accommodate a large amount of work at risk. Similar to the overhead of checkpoints the recovery time also depends on where the fault occurred. That is, if a fault is detected at the first checkpoint then the amount of rollback required would be *nx* because the first check point is placed after *nx* as shown in fig 7.1(b). If the fault is detected at the second checkpoint then the rollback would be *(n-1)x* as shown in fig 7.1(c), similarly it would follow as *(n-2)x* and *(n-3)x* and so on for the faults that are detected thereafter as shown in fig 7.1(d) and 7.1(e) respectively. Fig 7.1(f) shows the distribution of check points where a fault did not occur.

Hence a task *t* should satisfy the below equation to execute before the deadline and to guarantee the fault tolerance. Assume that a fault might have occurred at section k where $0 \le k \le n$ then

$$D \ge \sum_{i=1}^{k} (r + C_i) + C_k + \sum_{i=k+1}^{n} C_i \tag{7.5}$$

Where $\sum_{i=1}^{n} C_i = nx + (n-1)x + \ldots + 2x + 1x = C$ (7.6)

In equation (7.5) term $\sum_{i=1}^{k} (r + C_i)$ represents the execution of a task up to a failure at a section k. The term C_k represents the roll back time and the term $\sum_{i=k+1}^{n} C_i$ represents the execution of the remainder of the task with no checkpoints because we considered that a fault can occur not more than once. Hence further insertion of check points at this point is not required. It is intuitive to illustrate that the more the number of checkpoints made in a real-time task the more the task is fault tolerant. However, it is imperative to carefully choose the number of checkpoints, because there is a chance that the overhead might exceed slack time S_l, which makes it infeasible to finish the task before its deadline. Also, if the value of *n* is very small, then the value of *nx* will be comparatively bigger, which can be inferred from (7.1), and the slack available might not be sufficient for the rollback if a fault is detected in this section. Hence the value of n has to be increased. This can be easily checked by the following inequality

$$nx < S_l \tag{7.7}$$

If the above inequality is not true then the value of n has to be increased by 1 until the inequality is true.

7.3 Security Implementation

Two security mechanisms confidentiality and integrity described in chapter 6, to provide security to real time embedded systems are used here again. Please refer to section 6.3 for more details.

7.3.1 Integration of Security Mechanisms with Fault Recovery using Non-Uniform check points

The implementation of security along with reliability in a real time task can be done in two ways. First method considers the overhead involved for the entire data that has to be secured at once, that is if we are considering confidentiality alone, based on the requirement, any algorithm can be applied to encrypt the data all at once, similarly for integrity.

That is security overhead S_o= P or Q. For simplicity, we considered the implementation of confidentiality and integrity individually for this method. The security over head that can be utilized for a particular task t can be calculated using the formulae

$$\mu_i^c = \frac{l_i}{D - \sum_{i=1}^{k}(r + C_i) + C_k + \sum_{i=k+1}^{n} C_i} \tag{7.8}$$

Equation (7.8) is derived from equation (7.5) and (6.5) for the confidentiality over head.

$$\mu_i^g = \frac{l_i}{D - \sum_{i=1}^{k}(r + C_i) + C_k + \sum_{i=k+1}^{n} C_i} \tag{7.9}$$

Equation (7.9) is derived from equation (7.5) and (6.8) for the integrity overhead.

Any mapping function can be used to map the values obtained from equation (7.8) and (7.9) to find the corresponding algorithm that provides maximum security to the task.

Whereas in the second method, the data is divided in to *n* uneven sections, same as the tasks CPU cycles, using the formulae

$$nx + (n-1)x + (n-2)x + ... + 2x + 1x = l \tag{7.10}$$

Where *l* is the amount of data that has be secured for the entire task.

$$x = \frac{2l}{n(n+1)} \tag{7.11}$$

Once the data is divided into *n* sections, the security is implemented to each and every section separately. The equation below shows the implementation of reliability and security for *n* different sections of data using non uniform check point strategy.

$$D \geq \sum_{i=1}^{k} (r + C_i + \frac{l_i}{\mu_i^c} + \frac{l_i}{\mu_i^g}) + C_k + \sum_{i=k+1}^{n} (C_i + \frac{li}{\mu_k^c} + \frac{li}{\mu_k^g}) \tag{7.12}$$

For any given task *t* first the number of check points *n* that are required for fault recovery are assumed, then the CPU cycles *C* and the amount of data *l* that has be to secured is divided in to *n* sections using the formulae (7.1) and (7.10) and then the task is executed by placing a check point after each section and also applying the security mechanisms for each section separately. For the first section the basic confidentiality and integrity are provided by choosing the corresponding E1 and H1 algorithms from Tables 6.1 and 6.2. For the next section the security can be increased by one that is, E2 and H2 can be used for security. Similar way is used to increase the

security for other sections. If a fault occurrs at section k then the security mechanisms Ek and Hk that are used for that particular section will be used for the rest of the sections too, because at this point a portion of slack has been already utilized for the roll back. Further increasing the security might decrease the slack even more, resulting in an effect which might leave the task's execution beyond the deadline

. Hence the security at this point is not increased. In Eq. (7.12) the term $\sum_{i=1}^{k}(r + C_i + \frac{l_i}{\mu_i^c} + \frac{l_i}{\mu_i^g})$ represents the execution of task by placing check points for fault detection and using the algorithms of confidentiality and integrity for security. The security algorithms are increased after each section by one until a fault has been detected at section *k*. The term C_k represents the roll back of the section where a fault occurred. The last term $\sum_{i=k+1}^{n}(C_i + \frac{li}{\mu_k^c} + \frac{li}{\mu_k^g})$ represents the execution of the rest of the task without any check points. The security mechanisms that will be used for the rest of the sections are going to be same as the one that has been used for the section *k*, because of the limited slack that is available.

7.4 Performance Evaluation

In this section we evaluate the effectiveness of the proposed approach of integrating fault recovery (through checkpoints) and quality of security (measured as security levels). To demonstratively show the strength of our scheme, we conducted extensive experiments. It is to be noted that worst-case execution times are derived from real-world applications.

7.4.1 Security versus reliability (Non-Uniform Check points)

Fig. 7.2 depicts how the confidentiality levels vary when the check points are placed in Uniform and Non-Uniform pattern. Here K=1, 2, 3 represents that fault might have occurred in section 1, 2, 3 respectively in Non-Uniform check point strategy. Uni is used to represent the uniform check point strategy. For uniform check points, the confidentiality level gradually increases with increase in the number of check points from a minimum, whereas in the case of Non-Uniform check points, we observe a lot of variation depending upon where the fault is occurred.

The higher the values of k the higher the security levels are. However, the average confidentiality level guaranteed in non-uniform check point strategy is either higher or equal to the average confidentiality levels in uniform check point strategy. It is to be noted from the graph that the Confidentiality levels corresponding to k=2, n=1 and k=3 and n=1, n=2 are zero because when we assume only one check point in a task, the possibility of K=2 is impossible because $0 \leq k \leq n$.

Fig.7.3 indicates the improvement of confidentiality levels achieved by adopting Non-Uniform check point placements over the Uniform check point strategy. Here security for the entire data is considered at once. NU represents average security levels for non uniform check point strategy where different k is considered.

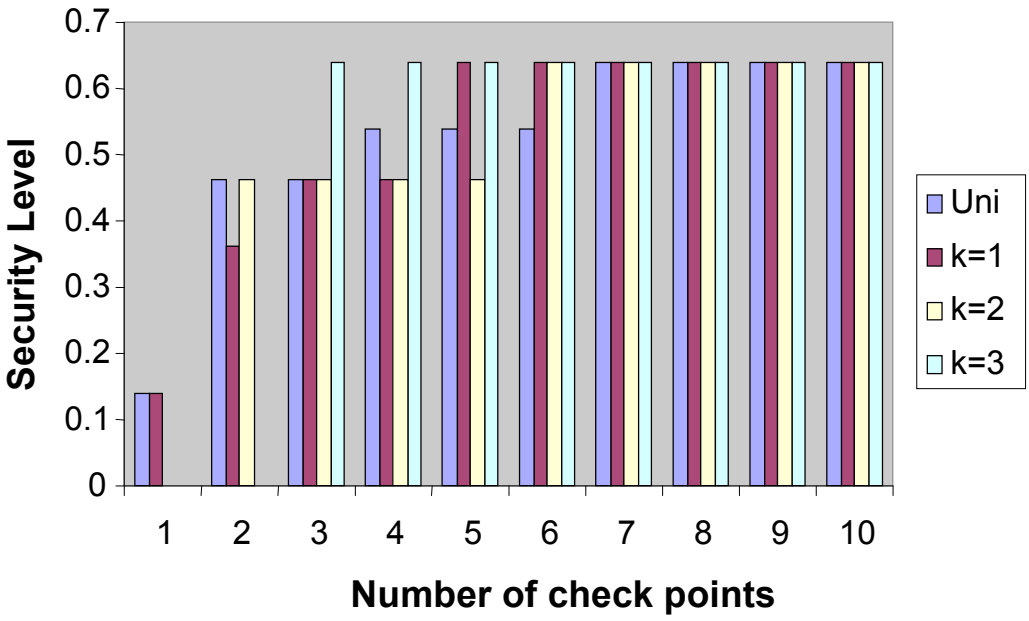

Figure 7.2 Confidentiality levels for Uniform check pointing and Non-Uniform check pointing when fault occurred at K=1,2,3

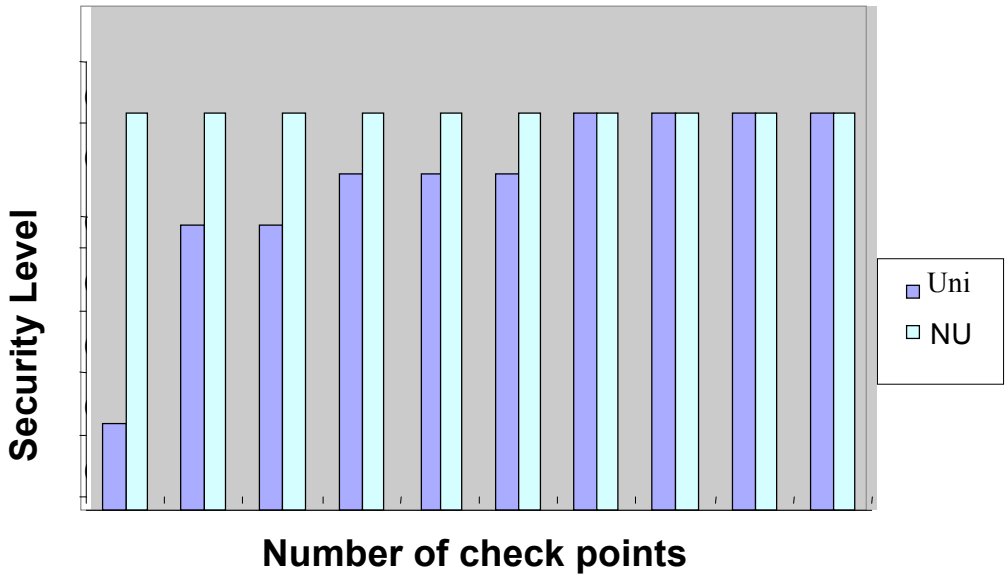

Figure 7.3 Uniform Vs Non-Uniform check pointing for Confidentiality

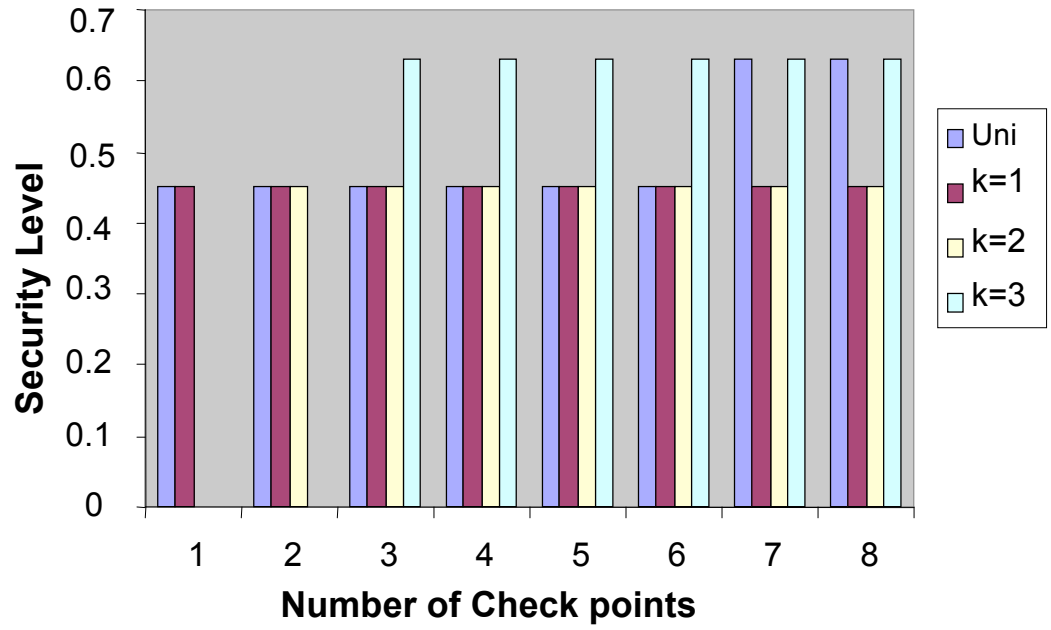

Figure 7.4 Integrity levels for Uniform check pointing and non uniform check pointing when fault occurred at k=1,2,3

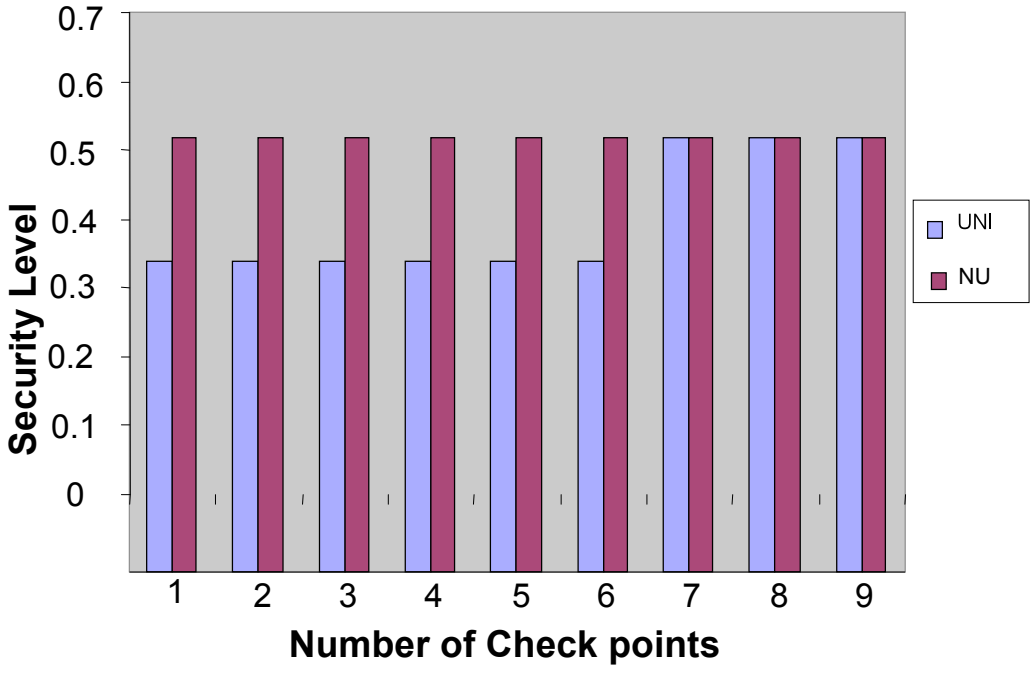

Figure 7.5 Uniform Vs Non-Uniform check pointing for Integrity

Fig 7.4 shows the effect on integrity levels when uniform and non-uniform check points are considered for k= 1, 2 and 3. For uniform check point policy, the security levels increase with the number of check points, whereas, for non-uniform check point policy, the security levels either increase or remain constant for different k, that is, the values are either greater or equal to the values in the uniform check point policy. This proves that non uniform approach is definitely better than uniform approach.

Figure 7.5 shows the difference of integrity levels for uniform and non uniform check points when the security is applied to the entire data at once. Same as the confidentiality, the integrity levels are either same or higher than the uniform check point integrity levels. With this observation it can be stated that irrespective of the values of k, the non uniform check point strategy is definitely better than the uniform check point strategy.

Table 7.1: Security levels for Uniform and non-Uniform check points for different k

Security Levels			
Fault Occurred at	Uniform	Non-Uniform	
		Method 1	Method 2
K=1	0.32	0.26	0.26
K=2	0.32	0.32	0.37
K=3	0.32	0.32	0.53
K=4	0.32	0.4	0.56

Table 7.1 shows the security levels for uniform and non uniform check points when C/D and r/D are taken as 0.54 and 0.0045 respectively. The average percentage

increase in security between uniform and non uniform check points is 34.4 when method 2 is considered and 1.56 when method 1 is considered. The average percentage increase from method 1 to method 2 is 32.3. This proves that either method in non uniform check points approaches are comparatively better than uniform check point approach in real time systems where as method 2 has higher percentage increase, so, it is a better approach than method 1.

7.5 Summary

In this chapter, we studied the approaches to integrate fault recovery and adaptive control of quality of security. In the proposed scheme we considered fault recovery and security service – confidentiality and integrity. This research was undertaken mainly in the following steps. First, the overhead models for the security service and for the fault recovery model non uniform check points were presented. Two methods are proposed in combining the security with reliability in a real time task. First method considers the security overhead for the entire data at once and in the second method, the data is divided into *n* different sections and the security overhead for each section is considered different. Next, we integrated both the fault recovery mechanisms with adaptive quality of security. To facilitate the presentation of the integration, we introduce the mathematical notations. Third, we proposed a novel way of maximizing quality of security with minimum reliability. Additionally, we described a means of optimizing reliability while providing basic quality of security. Our approach aims to maximize the security of a real-time system while providing checkpoints to efficiently support fault tolerance.

The simulation proves that the non uniform check point strategy gives a 34.4 percent higher security than the uniform check point when the second method is considered for security. It also proves that the second method is better than the first method by 32.3 percent. The overhead for all the approaches are represented in mathematical notations. The approach presented here maximizes the security and reliability in an innovative way.

Chapter 8

Conclusions and Future Work

In this dissertation, we proposed approaches to improve the energy efficiency and reliability of storage systems and we also concentrated on improving the security and reliability of real time embedded systems. We have developed an algorithm to improve the energy conservation of storage disks while maintaining good reliability, and then we extended the algorithm to incorporate additional overheads. We proposed another approach to skew the popular data on to few disks of the storage systems and thereby increase the energy efficiency and reliability. With respect to embedded systems we concentrated on improving their security and reliability by mainly incorporating the vital security services and check pointing policies.

This chapter concludes the dissertation by summarizing the contributions and describing future directions. The chapter is organized as follows: section 8.1 highlights the main contributions of the dissertation. In section 8.2, we concentrate on some future directions, which are extensions of our past and current research on large scale storage systems.

8.1 Main Contributions

To meet the today's need of energy for data centers a number of energy efficient techniques are proposed. These are very important to conserve electricity thereby reduce the electricity bills and in turn reduce the air pollutants and save environment. The existing energy efficient techniques are well studied in the arena of energy conservation but there is no mention about the reliability. Saving the energy of the disks is worthwhile only when the disks are available without crashing during the process. Previous research studies have proved that aggressively conserving energy without considering the disk reliability impacts the disks reliability negatively. Hence there is a need to design energy efficient and reliable approaches for storage systems. The objective of our research is to propose ways to improve energy conservation while attaining reliability of the disk systems.

Real time embedded systems such as online banking is used by hundreds of millions of people every day to keep track of their finances or to make purchases and etc. These online banking systems need to be very secure and reliable as they deal with the finances. Ignoring either security or reliability of these systems creates a very major concern both to the consumer and also the producer. Hence there is a need to design highly secure and reliable real time embedded systems. In this dissertation we have discussed ways to improve security and reliability of real time systems and we have also concentrated on their integration.

The main contributions are summarized as follows:

1. Reliability Aware Energy Efficient Algorithm for Storage Systems (see section 8.1.1)

2. Control Algorithm for Power State Transitions (see section 8.1.2)
3. Utilization Based Reliable and Energy Efficient Disks Systems (see section 8.1.3)
4. Security and Reliability for Real time embedded systems (see section 8.1.4)
5. Security and Reliability of real time systems using non uniform check point strategy (see section 8.1.5)

8.1.1 Reliability Aware Energy Efficient Algorithm for Storage Systems

As far as the existing literature on the storage systems is concerned there is zero or very minimal research that has been studied to address both energy efficiency and reliability. Most of the studies concentrated on either one ignoring the other, both energy conservation and reliability are very important and complement each other only when they are implemented together. We proposed an algorithm which improves the reliability of the disks by operating the disks only in safe utilization zone where the disks reliability is very high and it switches the disks back and forth between active and idle states to conserve energy when there is no or minimal activity on the disks. Our experimental evaluation proves that the strategy provided is indeed an efficient approach with respect to the existing state of the art technologies.

8.1.2 Control Algorithm for Power State Transitions

First chapter of this dissertation proposes an algorithm to improve energy efficiency and reliability of storage disks. Energy efficiency is achieved by operating disks in different power modes and reliability is achieved by operating disks mostly in safe utilization zone. Operating the disks in different power modes involves disk spin

ups and spin downs to traverse the disks between different power states. This frequent switching is necessary to keep the disks running in low power modes whenever possible to reduce energy consumption. However switching the disks between different power modes too aggressively involves the switching overhead and it also negatively affects the disk reliability because of the maximum number of transitions that will be imposed on the disks. Hence we extended the previously proposed algorithm to put a check on the number of power state transitions that will be allowed and also to traverse the disks in a more elegant way to reduce the overhead thereby increasing the energy conservation and improving reliability. Experimental results conducted supported our argument by proving that the algorithm gracefully transits between different states only when necessary and increases the reliability and energy efficiency.

8.1.3 Utilization Based Reliable and Energy Efficient Disks Systems

Popular data concentration (PDC) and Massive array of idle disks (MAID) are two existing popular techniques to improve the energy efficiency of the disk systems. PDC and MAID both skew the popular data on to few of the disks thereby allowing other disks to stay in low power modes to conserve energy. These two are good approaches for energy efficiency and it might have been more beneficial if they had taken into account the issue of disk reliability. In this dissertation we proposed an algorithm which skews the popular data on to few disks and allows the remaining disks to stay in low power modes to save energy as in the case of PDC and MAID, along with the energy we have also considered reliability here, and to achieve reliability we made sure that the disks operate only in safe utilization to keep the reliability high. Extensive experimental results conducted proved that our algorithm

saved more energy with respect to PDC and MAID while achieving high levels of reliability.

8.1.4 Security and Reliability for Real time embedded systems

Real time embedded systems such as online banking and etc are huge part of today's day to day life. With the increasing usage it becomes highly crucial to make those systems very reliable and secure at the same time. Most of the existing work has been done on either security ignoring reliability or the other way around. In this dissertation we have proposed ways to improve security and reliability and we have also designed ways to integrate both in to the real embedded systems. Uniform check pointing policies are used to improve reliability while two crucial security mechanisms are used to improve reliability. Experimental results show that our approach significantly improves security and reliability

8.1.5 Security and Reliability of real time systems using non uniform check point strategy

We have further extended our work to incorporate non uniform check point strategy to improve the reliability of the system. We have also proposed two different ways to integrate security along with reliability for real time embedded systems. Our results proved that non uniform check point strategy is a step ahead of the uniform check point strategy in performance

8.2 Future Work

In the course of designing and evaluating energy- efficient and reliable techniques for high-performance storage systems, we have found several interesting issues that

are still unresolved. This section overviews some of these open issues that need further investigation. These open issues present opportunities for my future research.

8.2.1 Power Sensitive Applications

Embedded/mobile devices are even more sensitive to power consumption due to the limited battery life. The usage of these embedded/mobile devises is increasing tremendously. I will extend my previous energy-aware and reliable research to embedded devices/sensor networks and evaluate previous algorithms in terms of energy efficiency and reliability in a more power sensitive environment.

8.2.2 Meta Data Information

Metadata plays a key role in designing some of the most efficient storage systems. The fifth chapter in this dissertation explains an algorithm that skews the popular data to increase reliability and energy efficiency of the storage systems. We did not explain the metadata management in that study. We will further extend the work to study how the metadata management impacts the energy efficiency and reliability of the proposed algorithms. It is very important to study the effects of metadata on power consumption because it gives a whole new direction to the research if the metadata impacts the algorithm in a negative way.

8.2.3 Dynamic Data Access Pattern Prediction

In this dissertation we considered that the data access pattern is known a priori for the data disks running in large scale data centers. It was an assumption made to conduct the experiments, but in real world the data access pattern is very robust and dynamic. Hence the algorithm should be extended to see how it behaves for a dynamic data access pattern and how the energy and reliability are affected.

8.2.4 Writes, creates and purge operations

The work we have described in this dissertation on large scale storage systems considers only reads for the simplicity of the work. We plan to extend it further to include the very basic and important operations such as writes, create and purge.

8.2.5 Validating Reliability Model

The reliability model experimental results are generated using the simulation, in this research. To gain confidence in our reliability model performance evaluation results, we would like to further extend the work by implementing the model in real time and studying the effects on disk reliability.

References

[1] A. L Leistman and R H Campbell, “A fault-tolerant scheduling problem,” *IEEE Transactions on Software Engineering*, v.12 n.11, p.1088-1089, NOV. 1986

[2] A. Sung, SAREC: A Security-Aware Scheduling Strategy for Real-Time Applications on Clusters, *Proceedings of the 2005 International Conference on Parallel Processing*, 2005

[3] A.S. Tanenbaum, M. Steen, “Distributed Systems: Principles and Paradigms”, ISBN 0130888931, *Prentice Hall, 1st edition,* January 15, 2002.

[4] Agrawal, D. and El Abbadi, A. 1990. Integrating security with fault-tolerant distributed databases. *Comput. J.* 33, 1 (Feb. 1990), 71-78.

[5] B. Schroeder, and G.A. Gibson, “Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you?” *Proc. 5th Conf. USENIX Conf. on File and Storage Technologies,* vol.5, San Jose, CA, Feb 2007.

[6] Bertram, T.J.; Demaree, K.D.; Dangelmaier, L.C., "An integrated package for real-time security enhancement," *IEEE Trans on Power Systems*, vol. 5, no. 2, pp.592-600, May 1990.

[7] Brett Battles, Cathy Belleville, Susan Grabau, Judith Maurier. “Reducing Datacenter Power Consumption through Efficient Storage”. *Network Appliance, Inc.* February 2007.

[8] C. Ruemmler and J. Wilkes, “UNIX disk access patterns,” *Proc. of the Winter*

1993 USENIX, pp. 405–420, Jan 1993.

[9] C. Wang, W.A. Wulf, “Towards a Framework for Security Measurement,” *Proc. National Information Systems Security Conference*, Baltimore, MD, pp. 522-533, October, 1997.

[10] Clark, R.K.; Greenberg, I.B.; Boucher, P.K.; Lunt, T.F.; Neumann, P.G.; Wells, D.M.; Jenson, E.D., "Effects of multilevel security on real-time applications," *Computer Security Applications Conference, 1993. Proceedings., Ninth Annual* , vol., no., pp.120-129, 6-10 Dec 1993

[11] D. Colarelli and D. Grunwald, “Massive Arrays of Idle Disks for Storage Achieve,” *Proc. of Supercomputing*, November 2002.

[12] D. Li and J. Wang, “EERAID: Energy-Efficient Redundant and Inexpensive Disk Array”, *Proc. of the 11th ACM SIGOPS European Workshop*, Sept 2004.

[13] D. Li, and J. Wang, "Conserving Energy in RAID Systems with Conventional Disks," *Proc. Int'l Work-shop Storage Network Arch. and Parallel I/Os,* 2005.

[14] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems", *Proc. IEEE/ACM Int'l conf. Computer-aided design,* pp. 35-40, 2004.

[15] D.P. Helmbold, D.D. Long, and B. Sherrod, "A dynamic disk spin-down technique for mobile computing," *Proc. 2nd annual Int'l Conf. Mobile Computing and Networking*, pp. 130-142, 1996.

[16] Drumea, A.; Svasta, P.; Popescu, C., "Remote access solutions for industrial control systems," Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 2004. *27th International Spring Seminar ,* May 2004.

[17] E. Nahum, S. O'Malley, H. Orman, R. Schroeppel, “Towards High Performance Cryptographic Software,” *Proc. IEEE Workshop Architecture and Implementation of High Performance Communication Subsystems*, August 1995.

[18] E. Pinheiro, and R. Bianchini, “Energy conservation techniques for disk array-based servers,” *Proc. 18th Annual Int’l Conf. Supercomputing*, 2004.

[19] E. Pinheiro, W. Weber, and L. A. Barroso, “Failure trends in a large disk drive population,” *Proc. 5th Conf. File and Storage Tech.,* San Jose, CA, Feb. 2007.

[20] E. Rosti, G. Serazzi, E. Smirni and M.S. Squillante, "Models of Parallel Applications with Large Computation and I/O Requirements," *IEEE Trans. Software Eng.* ,vol. 28 , no.3, pp. 286-307, 2002.

[21] E. V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving Disk Energy in Network Servers," *Proc. 17th Int'l Conf. Supercomputing*, pp. 86-97.

[22] E.V. Carrera, E. Pinheiro, and R. Bianchini, “Conserving Disk Energy in Network Servers”, *Proc. of the 17th International Conference on Supercomputing (ICS’03),* June 2003.

[23] F. Douglis, P. Krishnan, and B. Marsh, “Thwarting the Power-Hunger Disk,” *Proc. Winter USENIX Conf.*, pp.292-306, 1994.

[24] F. Sultan, T. Nguyen, L. Iftode, “Scalable Fault-Tolerant Distributed Shared Memory”, *ACM/IEEE Supercomputing Conference* 2000, ISSN: 1063-9535

[25] H. Kopetz H. Kantz G. Grunsteidl P. Puschner and J. Reisinger, “Tolerating Transient Faults in MARS,” *20th Ann. Int'l Symp. Fault-Tolerant Computing ,* pp. 466-473, June 1990.

[26] Hardekopf, B. and Kwiat, K, “Exploiting the Overlap of Security and Fault-Tolerance” *Proceedings of the Academia/industry Working Conference on*

Research Challenges (April 27 - 29, 2000). AIWORC. IEEE Computer Society, Washington, DC, 361

[27] Hardekopf, B.; Kwiat, K.; Upadhyaya, S., "Secure and fault-tolerant voting in distributed systems," *Aerospace Conference, 2001, IEEE Proceedings.* , vol.3, no., pp.3/1117-3/1126 vol.3, 2001

[28] http://www.cryptopp.com/

[29] IBM Hard Disk Drive – Ultrastar 35Z15.

[30] IDEMA Standards, "Specification of Hard Disk Drive Reliability", document number R2-98.

[31] J. A. Stankovic, M. Spuri, K. Ramamritham, and G.C. buttazzo, "Deadline Scheduling for Real-Time Systems – EDF and Related Algorithms*," Kluwer Academic Publishers*, 1998

[32] J. G. Elerath and S. Shah, "Server class disk drives: how reliable are they," *IEEE Reliability and Maintainability Symp.,* pp. 151-156, Jan 2004.

[33] J. Mao, C. G. Cassandras, Q. Zhao, "Optimal Dynamic Voltage Scaling in Energy-Limited Nonpreemptive Systems with Real-Time Constraints," *IEEE Trans. Mobile Computing,* vol. 6, pp. 678-688, Jun 2007.

[34] J. Yue, Y. Zhu, and Z. Cai, "Evaluating Memory Energy Efficiency in Parallel I/O Workloads", *Proc. IEEE Int'l Conf. Cluster Computing*, Austin, Texas, 2007.

[35] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, R. Wang, "Modeling Hard-Disk Power Consumption," *Proc. USENIX Conf. File and Storage Technologies*, 2003.

[36] K. Bellam, A. Manzanares, and X. Qin, "Improving Reliability and Energy

Efficiency of Disk Systems," *Proc. 46th ACM Southeast Conference*, March 2008

[37] K. Bellam, A. Manzanares, X. Ruan, and X. Qin, "Integrating Security and Reliability in Real-time Embedded Systems," *Journal of Autonomic and Trusted Computing*, Mar 2008.

[38] K. Bellam, A. Manzanares, X. Ruan, X. Qin, and Y.-M. Yang, "Improving Reliability and Energy Efficiency of Disk Systems via Utilization Control," *Proc. IEEE Symposium on Computers and Communications*, July 2008

[39] K. Bellam, R. K. Vudata, X. Qin, Z.-L. Zong, M. Nijim, and X.-J. Ruan, "Interplay of Security and Reliability using Non-Uniform Checkpoints," *Proc. 16th Int'l Conference on Computer Communications and Networks (ICCCN),* Honolulu, Hawaii, Aug. 2007

[40] K. Bellam, R. K. Vudata, X. Qin, Z.-L. Zong, M. Nijim, X.-J. Ruan, "Interplay of Security and Reliability using Non-Uniform Checkpoints," *Proc. 16th IEEE Int'l Conf. Computer Comm. and Networks*, 2007.

[41] K. Bellam, Z. Zong, M. Alghamdi, M. Nijim, and X. Qin, "Integrating Fault Recovery and Quality of Security in Real-time Systems," *Proc. IEEE International Symposium on Ubisafe Computing*, Ontario, Canada, May 2007

[42] K. Kwiat, "Panel Abstract," srds, p. 262, *19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, 2000

[43] K. Okada, N. Kojima, and K. Yamashita, "A Novel Drive Architecture of HDD: Multimode Hard Disc Drive," *Proc. Int'l Conf. on Consumer Electronics*, pp. 92-93, June, 2000.

[44] Kang, K. and Son, S. H. 2006, "Towards security and QoS optimization in real-

time embedded systems," *SIGBED Rev.* 3, 1 (Jan. 2006), 29-34.

[45] Kim, K.H., "Incorporation of security and fault tolerance mechanisms into real-time component-based distributed computing systems," *Reliable Distributed Systems, 2001. Proceedings. 20th IEEE Symposium on* , vol., no., pp.74-75, 2001

[46] L. Lu, P. Varman, and J. Wang, "DiskGroup: Energy Efficient Disk Layout for RAID1 Systems," *Int'l Conf. Networking, Architecture, and Storage,* pp. 233-242.

[47] L. Yuan, and G. Qu, "Analysis of energy reduction on dynamic voltage scaling-enabled systems," *IEEE Tran. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 12, pp. 1827-1837.

[48] L.W. Lee, P. Scheuermann, and R. Vingralek, "File assignment in parallel I/O systems with minimal variance of service time," *IEEE Transactions on Computers*, Vol. 49, Issue 2, pp. 127-140, 2000.

[49] Liestman and R. Campbell, "A Fault-Tolerant Scheduling Problem," *IEEE Trans. Software Eng.*, vol. 12,no. 11,pp. 1089-1095, Nov.1986.

[50] M. K. Reiter, K.P. Birman, and R Van Renesse, "A Security Architecture for Fault-Tolerant Systems," *Technical Report*. UMI Order Number: TR93-1354., Cornell University.

[51] P. M. Greenawalt, "Modeling Power Management for Hard Disks," Proc. Int'l Workshop Modeling, Analysis, and Simulation on Computer and Telecom. Systems, pp.62-66, 1994.

[52] Q. Ahmed and S. Vrbsky, "Maintaining security in firm real-time database systems," *Proc. 14th Ann. Computer Security Application Conf.*, 1998.

[53] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao, “Reducing energy consumption of disk storage using power-aware cache management,” in *Tenth International Symposium on High Performance Computer Architecture (HPCA-10)*, (Madrid, Spain), Feb. 14–18, 2004.

[54] R. Bianchini and R. Rajamony, “Power and Energy Management for Server Systems,” *Technical Report DCS-TR-528, Department of Computer Science, Rutgers University*, June 2003

[55] Rami Melhem, Daniel Mosse, Elmootazbellah (Mootaz) Elnozahy, "The Interplay of Power Management and Fault Recovery in Real-Time Systems," *IEEE Trans on Computers* , Feb.2004.

[56] S. Cheng and Y. Huang, “Dynamic real-time scheduling for multi-processor tasks using genetic algorithm,” *Proc. 28th Ann. Int’l Conf. Computer Software and Applications*, pp. 154 – 160, Sept. 2004.

[57] S. Ghosh, D. Mosse, and R. Melhem, “Implementation and Analysis of a Fault-Tolerant Scheduling Algorithm,” *IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 3, Mar. 1997.

[58] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, “DRPM: dynamic speed control for power management in server class disks,” *Proc. 30th Int’l Symp. Computer Architecture,* pp. 169-18, May 2003.

[59] S. Gurumurthi, J. Zhang, A. Sivasubramaniam, M. Kandemir, H. Fanke, N. Vijaykrishnan, and M. Irwin, “Interplay of Energy and Performance for Disk Arrays Running Transaction Processing Workloads,” Proc. of *ISPASS*, pp. 123-132, March 2003.

[60] S. H. Son, R. Mukkamala, and R. David, "Integrating security and real-time requirements using covert channel capacity," *IEEE Trans. Knowledge and Data Engineering*, Vol. 12 , No. 6, pp. 865 – 879, 2000.

[61] S. H. Son, R. Zimmerman, and J. Hansson, "An adaptable security manager for real-time transactions," *Proc. 12th Euromicro Conf. Real-Time Systems*, pp. 63 – 70, June 2000.

[62] S. Punnekkat , A. Burns , R. Davis, "Analysis of Checkpointing for Real-Time Systems," *Real-Time Systems*, v.20 n.1, p.83-102, Jan. 2001

[63] S. Ramos-Thuel and J. Strosnider, "Scheduling Fault Recovery Operations for Time-Critical Applications," *Proc. 4th IFIP Conf. Dependable Computing for Critical Applications*, Jan. 1995.

[64] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges," *ACM Trans. Embedded Computing Systems*, Vol. 3, No. 3, pp. 461-491, Aug 2004

[65] S. Son, G. Chen, M. Kandemir, and A. Choudhary, "Exposing disk layout to compiler for reducing energy consumption of parallel disk based systems," *Proc. ACM Symp. Principles and Practice of Parallel Programming*, pp. 174-185, 2005.

[66] S. Song, Y. K. Kwok, K. Hwang, "Trusted Job Scheduling in Open Computational Grids: Security-Driven Heuristics and A Fast Genetic Algorithms," *Proc. Int'l Symp. Parallel and Dist'd ,*2005

[67] S. Srinivasan and N.K. Jha, "Safety and Reliability Driven Task Allocation in Distributed Systems", *IEEE Trans. Parallel and Distributed Systems*, Vol. 10, No. 3, pp. 238-251, March 1999.

[68] S. W. Son, G. Chen, O. Ozturk, M. Kandemir, and A. Choudhary, "Compiler-Directed Energy Optimization for Parallel Disk Based Systems," *IEEE Trans. Parallel and Distr. Sys., vol. 18, no.*9, pp. 1241-1257, 2007.

[69] S. Yajnik, S. Srinivasan, and N.K. Jha, "TBFT: A Task-Based Fault Tolerance Scheme for Distributed Systems", *Proc. Int'l Conf. Parallel and Distributed Computer Sys.*, Oct. 1994.

[70] Sayantan Chakravorty, Laxmikant V. Kale, "A Fault Tolerant Protocol for Massively Parallel Systems," *ipdps, 18th International Parallel and Distributed Processing Symposium (IPDPS'04)* - Workshop 11, 2004

[71] T. Simunic, L. Benini, and G. De Micheli, "Dynamic Power Management of Laptop Hard Disk," *Proc. Design Automation and Test*, Europe, 2000.

[72] T. Xie and X. Qin,"Scheduling Security-Critical Real-Time Applications on Clusters," *IEEE Transactions on Computers*, vol. 55, no. 7, pp. 864-879, July 2006.

[73] T. Xie, X. Qin, "A new allocation scheme for parallel applications with deadline and security constraints on clusters", *Proc. IEEE Int'l Conf. Cluster Computing,* Sep 2005

[74] T. Xie, X. Qin, A. Sung, "SAREC: A Security-Aware Scheduling Strategy for Real-Time Applications on Clusters," *Proc. 34th Int'l Conf. Parallel Processing*, pp.5-12, Norway, June 14-17, 2005.

[75] T. Xie, X. Qin, A. Sung, M. Lin, and L. Yang, "Real-Time Scheduling with Quality of Security Constraints," *Int'l Journal of High Performance Computing and Networking,* 2006.

[76] Tao Xie, Yao Sun, "Sacrificing Reliability for Energy Saving: Is It Worthwhile

for Disk Arrays?" *The 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008),* Miami, Florida, USA, April 14-18, 2008.

[77] V. Kalogeraki, P.M. Melliar-Smith, and L.E. Moser, "Dynamic scheduling for soft real-time distributed object systems," *Proc. IEEE Int'l Symp. Object-Oriented Real-Time Distributed Computing*, pp.114-121, 2000.

[78] X. Qin and H. Jiang, "A Dynamic and Reliability-driven Scheduling Algorithm for Parallel Real-time Jobs on Heterogeneous Clusters", *Journal of Parallel and Distributed Computing*, vol. 65, no. 8, pp. 885-900, Aug. 2005.

[79] X. Qin, "Improving Network Performance through Task Duplication for Parallel Applications on Clusters," *Proc. 24th IEEE Int'l Performance, Computing, and Communications Conf.,* Phoenix, Arizona, April 2005.

[80] X. Qin, H. Jiang, and D. R. Swanson, "An Efficient Fault-tolerant Scheduling Algorithm for Real-time Tasks with Precedence Constraints in Heterogeneous Systems," *Proc. 31st Int'l Conf. Parallel Processing,* pp.360-368. Aug. 2002.

[81] X. Qin, H. Jiang, C. Xie, and Z. Han, "Reliability-driven scheduling for real-time tasks with precedence constraints in heterogeneous distributed systems," *Proc. Int'l Conf. Parallel and Distributed Computing and Systems* Nov., 2000.

[82] X. Qin, Z. Han, H. Jin, L. Pang, and S. Li. "Real-time Fault-tolerant Scheduling in Heterogeneous Distributed Systems," *Proc. Int'l Workshop Cluster Computing-Tech., Environments, and App.*, Vol. I, pp.421-427. June 2000.

[83] X. Yao and J. Wang, "RIMAC: A Redundancy-based Hierarchical I/O Architecture for Energy-Efficient Storage Systems", *Proc. of the 1st ACM EuroSys Conference*, Apr 2006.

[84] Y Yang; Z Wang; F Bao; Deng, R.H., "Secure the image-based simulated

telesurgery system," Circuits and Systems, 2003. ISCAS '03. *Proc of the 2003 Int'l Symposium,* May 2003.

[85] Y. Du, J. Dong, and M. Cai, "Dynamic Voltage Scaling of Flash Memory Storage Systems for Low-Power Real-Time Embedded Systems," *Proc. Int'l Conf. Embedded Software and Systems,* pp. 152-157, 2005.

[86] Y. Minsky, et al. "Cryptographic Support for Fault-Tolerant Distributed Computing," *Proc. 7th ACM SIGOPS European Workshop* (September 1996).

[87] Y. Zhang , K. Chakrabarty, "Fault Recovery Based on Checkpointing for Hard Real-Time Embedded Systems," *Proc 18th IEEE Int'l Sym on Defect and Fault Tolerance in VLSI Systems*, 2003

[88] Y. Zhang and K. Chakrabarty, "Dynamic adaptation for fault tolerant and power management in emb real-time systems," *ACM Trans on Emb Computing Systems,* May 2004.

[89] Z.-L. Zong, A. Manzanares, B. Stinar, and X. Qin, "Energy-Efficient Duplication Strategies for Scheduling Precedence Constrained Parallel Tasks on Clusters," *Proc. IEEE 8th International Conference on Cluster Computing* (Cluster'06), Sept. 2006.