Adaptive Quality of Security Control in Storage Systems

BY

Mais W. Nijim,

A disirtation submitted to the Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of philosophy in Computr Science

New Mexico Institute of Mining and technology

Socorro, New Mexico 87801

May 2007

To my parents,

Wasef and Diana,

and to my fiance Hisham Albataineh,

who made all of this possible, for their endless encouragement and patience.

Since security is of critical importance for modern storage systems, it is imperative to protect stored data from being tampered or disclosed. Although an increasing number of secure storage systems have been developed, there is no way to adaptively choose security services to meet disk requests flexible security requirements. Furthermore, existing security techniques for disk systems are not suitable to guarantee desired response times of disk requests. In this dissertation research, we proposed a series of adaptive quality of security control strategies, which can judiciously select the most appropriate security service for disk requests while endeavoring to guarantee desired response times of all disk requests. Specifically, we proposed in this dissertation a quality of security framework or StReD for storage resources. The StReD framework integrates a quality of security adaptor with an array of security services. Next, an adaptive strategy or AWARDS was developed for local storage systems to improve dynamically improve quality of security without adversely affecting the response times of write requests. To prove the efficiency of AWARDS, we built an analytical model to measure the probability that a disk request is completed before its desired response time. Furthermore, we designed and implemented an adaptive quality of security control strategy or ASPAD for parallel disk systems. ASPAD makes it possible for parallel disk systems to adapt to changing security requirements and workload conditions. Last but not least, we devised a security-aware cache management mechanism or CaPaS for cluster storage systems. CaPaS aims at achieving high security and desired performance for data-intensive applications running on clusters. CaPaS is used in combination with the ASPAD strategy, thereby providing high quality of security for cluster storage systems. Extensive trace-driven simulations, which are based on a set of real applications as well as synthetic workloads with various disk I/O characteristics, show that the adaptive quality of security control schemes sig-

nificantly improve security of local disk systems, parallel disk systems, and cluster storage systems without adversely affecting the performance of storage systems.

Acknowledgments

First of all, I would like to express my heartfelt thanks to my advisor, Dr. Xiao Qin, for his guidance, support, and encouragement during the course of my graduate study at the University of New Mexico Tech. I am especially indebted to him for teaching me both research and writing skills, which have been proven beneficial for my current research and future career. Without his endless efforts, knowledge, patience, and answers to my numerous questions, this dissertation research would have never been possible. The experimental methods and results presented in this dissertation have been influenced by him in one way or the other. It has been a great honor and pleasure for me to do research under Dr. Xiao Qin's supervision.

I am also very grateful for having an exceptional doctoral committee and wish to thank Dr. Subhasish Mazumdar, Dr. Dongwan Shin, Dr. Ali El. Osery for their continual support and encouragement.

Finally, I'd like to thank my family. My father and my mother were a constant source of support. I'm grateful to my brothers Yousef and shadi for their encouragement and enthusiasm. I am also indebted to my sisters Lina, Nisreen, Majd, and ala' for there unlimitted support. I'm especially grateful to my fiance Hisham Albataineh whose endless encouragement has been my source of inspiration and I would never have succeeded without him.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

In the past decade, storage systems have been an object of substantial interest because of an increasing number of emerging data-intensive applications like video surveillance [1], long running simulations [32], digital libraries [31], remote-sensing database systems [7], and out-of-core applications [22]. This trend can be attributed to the advances in computational power, disk performance, and high-speed networks. There are many cases where data-intensive applications require enriched security to protect data stored in storage systems from talented intruders [35]. Further, a large number of data-intensive applications (e.g., Real-time I/O-intensive applications) require guaranteed response times for interactive or high-priority data sets [11]. Therefore, it is imperative that storage systems are capable of providing strong security and guaranteed response times for disk requests. However, conventional wisdom on design of real-time and data-intensive applications is inadequate for security-sensitive data-intensive applications due to the lack of consideration for timing and security constraints of disk requests issued by data-intensive applications.

We believe that a feasible way of addressing performance and security needs of real-time and data-intensive applications is to design adaptive quality of security control approaches. Specifically, adaptive quality of security control

mechanisms aim to improve the quality of security of data-intensive applications while guaranteeing the desired response times of disk requests issued by data-intensive applications.

This chapter first manifests the problem statement in Section 1.1. Section 1.2 describes the scope of this research work. In section 1.3, we summarize the main contributions of the dissertation, and Section 1.4 presents the outline of the dissertation.

1.1 Problem Statement

Let us start this section with an overview of data-intensive applications. Section 1.1.2 shows that there is a great need to address the issue of security in storage systems. Finally, section 1.1.3 presents the initial motivation for this dissertation research by illustrating the limitations of existing solutions.

1.1.1 Data-Intensive Applications

Each year more data are generated and need to be processed . These days, there are huge amount of scientific and enterprise applications that deal with a huge amount of data . Such applications are called data intensive. In order to process and manipulate these kinds of applications, they need high performance storage and high performance computing infrastructure.

As data-intensive applications increase continuously in various domains such as high-energy physics, astrophysics, genomic computing and digital library, they need to be saved, retrieved, and analyzed rapidly. Storage systems are used to store and manage those data intensive applications while maintaining the availability and the ratability of the data. Storage systems are becoming an increas-

ingly important platform for data intensive applications [49]. Such applications include but not limited to image acquisition and processing computations, digital libraries, high performance massive data assimilation, and distributed data mining. The storage resources in the Data Grid plays an important role to employ or acquire data from massive data sets where are too large to be stored in a single site.

1.1.2 Security in Storage Systems

In the past ten years, storage systems are used o develop data-intensive applications. The data stored in data-intensive applications should be private and must be safeguards against unauthorized access.

Parallel disk systems play an important role in the development of high-performance data-intensive applications. This is mainly because parallel disk systems, which are highly scalable in nature, can alleviate the problem of disk I/O bottleneck. An efficient way of improving performance of disk systems is to make use of I/O parallelisms, which can be achieved by partitioning and distributing data among an array of disks. Disk I/O parallelisms can be provided in forms of either inter-request or intra-request parallelism. While inter-request parallelism enables multiple independent requests to be served simultaneously by a parallel disk system, intra-request parallelism allows a single disk request to be processed by multiple disks in parallel [28].

In the last few years, Data Grids have become popular for various scientific and commercial applications [37]. A Data Grid is a collection of geographically dispersed computing and storage resources [38][39] providing services to fit needs of applications like simulation analysis tools [40] and high-energy physics applications [41] [42]. A data grid is comprised of five kinds of resources: com-

putation, storage, software, communications, and network bandwidth [43]. Essentially, the Data Grid is defined as an infrastructure that manages large scales data-intensive files and provides computational resources across widely distributed systems [44][43]. Storage systems including hard disks and any storage media are the most commonly used resources in the Data Grids. Securing storage resources in Data Grid computing environments has increasingly become a major concern to make Data Grids attractive for a wide range of data-intensive applications, because security requirements are often imposed on storage resources to support security-critical applications. Many existing disk systems and Grids computing environments have not employed any security mechanism to counter security threats [45][46]. As such, it is extremely necessary to deploy security services to guard data stored in disk systems and storage resources like parallel disk systems in Data Grids. Snooping, alteration, and spoofing are three common attacks. Therefore, we considered three security services (confidentiality services, integrity services, and authentication services) to protect against common threats. Snooping, an unauthorized interception of information can be remedied by confidentiality services. Alteration, an unauthorized change of information can be countered by integrity services. Spoofing, an impersonation of one entity by another can be countered by authentication services [47][48]. With these three different types of security services in place, users can flexibly select the security mechanisms to dynamically build an integrated security protection against a diversity of threats and attacks in disk systems or Data Grids.

With the advent of powerful processors, fast interconnects, and low-cost storage systems, clusters of commodity PCs have emerged as a primary and cost-effective infrastructure for large-scale scientific and web-based applications. A large fraction of scientific and web-based applications are parallel and data-

intensive, because these applications deal with a large amount of data transferred either between memory and storage systems or among nodes of a cluster via interconnection networks. Importantly, cluster storage systems have become increasingly popular for data-intensive applications running on high-performance computing platforms, which assure the effectiveness of the nation's critical infrastructures [78][79]. The concept of cluster storage systems was introduced in conjunction with different domains of applications like Internet data caches [80] and video servers [81][82]. Cluster storage systems are ideal storage candidates for these data-intensive applications running on high-performance clusters, since the applications need extensive computation coupled with access to diverse data repositories.

1.1.3 Limitation of Existing Approaches

There is a large body of work in improving performance of disks, because disk I/O has become a serious performance bottleneck of computer systems. Previous techniques supporting high performance storage systems include disk striping [4][27][28], parallel file systems [8][16][21], load balancing [22][28], caching and buffering [12][13][17].

Although conventional storage systems are aimed at improving access times and storage space, many existing storage systems are vulnerable to a wide variety of potential threats. As such, the existing disk systems fail to meet the security requirements of modern data-intensive applications. To protect storage systems against all possible security threats, researchers have developed various ways of ensuring security of data in storage systems.

1.2 Motivation

Security critical data intensive applications have mandatory security requirements as well as stringent timing constraints. There are many cases where data-intensive applications require enriched security to protect data stored in storage systems from talented intruders [35]. Further, a large number of data-intensive applications such that real time I/O data intensive applications require guaranteed response times for interactive or high-priority data [11]. Therefore, storage systems are required to provide strong security and guaranteed response times for disk requests. This demanding requirement is increasingly becoming a critical and challenging issue in the development of the next generation storage systems.

1.3 Scope of the Research

This dissertation research focuses on securing storage systems for local and parallel disk systems as well as providing adaptive quality of security control for data-intensive applications running on clusters and Data Grids.

For sake of simplicity without loss of generality, we ignore the overhead caused by key management. This assumption is reasonable because the key management overhead can be readily included as part of security overhead in our model. With respect to various type of security services, we considered in this research three common services, namely confidentiality, integrity, and authentication. Please note that our model is very flexible in the way that it can be easily extended to include other types of security services.

In this dissertation study, we have explored the following. First, we have proposed a quality of security framework for storage resources in Data Grids. Second, we have proposed an adaptive quality of security control strategy that can

judiciously select the most appropriate security service for each write request while endeavoring to guarantee the desired response times of all disk requests. Third, we have developed an adaptive quality of security control scheme for parallel disk systems that makes it possible for parallel disk systems to adapt to changing security requirement and workload conditions. Last but not least, we have proposed a cache partitioning adaptive quality of security control mechanism that increases the quality of security by reducing cache miss rate.

1.4 Contributions

Adaptively enhancing quality of security in the context of storage systems is increasingly becoming an important and critical issue. As such, this dissertation research has made the first attempt to address the challenging issue of maximizing quality of security for data-intensive applications supported by storage systems while maintaining a high level of performance in terms of response time. In what follows, let us summarize the four contributions of this dissertation research

- **An Adaptive Quality of Security Control Framework for Storage Resources** : We proposed a quality of security control framework for storage resources in Data Grids, thereby significantly increasing the quality of security for data-intensive applications running in Data Grids.

- **An Adaptive Quality of Security Control Scheme for Local Disk Systems** : We devised an adaptive strategy, which can judiciously select the most appropriate security service for each disk request while endeavoring to guarantee the desired response time of all disk requests.

- **An Adaptive Quality of Security Scheme of Parallel Disk Systems** : We designed an adaptive quality of security control scheme for parallel disk

systems. The new scheme makes it possible for parallel disk systems to adapt to changing security and timing requirements and workload conditions.

- **Security-Aware Cache Management for Cluster Storage Systems** : we developed a security-aware cache management mechanism for cluster storage systems. Our security-aware cache management mechanism can be used in combination with a security control mechanism that can adapt to changing security requirements and workload conditions, thereby providing high quality of security for cluster storage systems.

Chapter 2

Related Work

Storage systems have become popular because of the increasing number of data-intensive applications. Also, Clusters have become popular as high-performance computing platforms for a variety of applications, and Grid is a promising next generation high-performance computing platform. This chapter briefly discusses existing techniques found in the literature that are most relevant to this dissertation research from two perspectives: the performance of storage systems, and security needs for real-time data-intensive applications running on clusters and Grids.

2.1 Storage Systems

There is a large body of work that he been done to increase the performance of storage systems. Designing reliable storage systems requires the use of various protection techniques like backup and remote mirroring. Predicting the dependability of the protection techniques, which is necessary for the system design, is to some extent difficult. Keeton and Merchant proposed a framework for evaluating the dependability of storage systems including data protection techniques and their compositions [50]. Watson and Coyne designed parallel I/O architecture and mechanisms, Parallel Transport Protocol (PTP), parallel FTP, and parallel client

Application Programming Interface (API) used by high-performance storage systems [51]. He et al. introduced a new benchmark tool called Storage Performance Evaluation Kernel Module (SPEK) for evaluating the performance of block-level storage systems in the presence of faults [52]. SPEK can be applied to both Direct Attached Storage (DAS) and block level networked storage systems such as storage area networks (SAN). Chambliss et al. presented a distributed controller coined as Service Level Enforcement Discipline for Storage System (SLEDS) [53]. SLEDS provides statistical performance guarantees on a storage system and a production-capable outboard controller that manage client workloads to meet the quality of service goals. Moreover, SLEDS can adaptively handle unpredictable workload conditions in a way that storage systems continue to get needed performance.

Recently, researchers have considered reducing energy cost in storage systems because the problem of energy consumption is significant where data centers heavily relies on high-end storage to meet their needs. Energy consumption has become a critical problem in modern storage systems, because increasing the need for data storage systems significantly increases the power budget in the disk array. Kim et al. proposed a new approach in which the energy consumption can be reduced without stopping the disk rotation [54]. The proposed approach is called Multiple Idle States (MIS), it reduces the energy consumption by model modulates the disk RPM to optimize the energy consumption during idle periods.

2.1.1 Local Disk Systems

Many previous studies have addressed the issue of improving performance of disks. This is mainly because disk I/O is a serious performance bottleneck for data-intensive applications. Previous techniques supporting high-performance storage

systems include disk striping [4][27][28], parallel file systems [8][16][21], load balancing[22][28], caching and buffering[12][14][17].

It is worth noting that disk scheduling algorithms play an important role in reducing the performance gap between processors and disk I/O [9][15][29][36]. The shortest seek time first (SSTF) algorithm is efficient in minimizing seek times, but it is starvation-bound and unfair in nature [10]. The SCAN scheduling algorithm can solve the unfairness problem while optimizing seek times[10]. Reist and Daniel proposed a parameterized generalization of the SSTF and SCAN algorithms [24]. However, the above disk scheduling algorithms are unable to guarantee desired response times of disk requests.

Many data-intensive applications require that data is stored or retrieved before a desired response time[25]. The SCAN-EDF can be employed to fulfil this requirement[29]. Recently, many disk schedulers were implemented for a mixed-media data set, a mixture of data accessed by multimedia applications and best-effort applications [2][5]. Several disk-scheduling algorithms were proposed to provide quality of service guarantees to different classes of applications [6][23][30].

2.1.2 Parallel Disk Systems

As we mentioned earlier, disk I/O has become a performance bottleneck for data-intensive application due to the widening gap between processor speeds and disk access speeds [57]. To help alleviate the problem of disk I/O bottleneck, a large body of work has been done on parallel disk systems. Kallahalla and Varman designed an on-line buffer management and scheduling algorithm to improve performance of parallel disks [58]. Scheuermann et al. addressed the problem of making use of striping and load balancing to tune performance of parallel disk systems[59]. Rajasekaran and Jin developed a practical model for parallel disk

systems [60]. Kotz and Ellis proposed investigated several write back policies used in a parallel file system implementation [61].

2.2 Security Issues in Storage Systems

Nowadays security is of critical importance for a wide range of data-intensive applications. In what follows, Section 2.2.1 describes security issues in storage systems. Section 2.2.2 summarizes previous work related to the concept of Quality of Security.

2.2.1 Security Needs for Data-Intensive Applications

Many data-intensive applications embrace a rich variety of security services to protect data residing in disk systems from talented intruders [35]. Thus, the issue of security in storage systems has been addressed and reported in the literature. Riedel et al. developed a common framework of core functions required for any secure storage system [26]. To protect data in untrusted storage systems, researchers designed and implemented cryptographic file systems where data is stored in encrypted form [3][14]. Several key distribution schemes were proposed in SFS [18] and SNAD system [19].

Data-intensive applications rely on stored and accessed data, supporting the availability, integrity, and confidentiality of these data is crucial. Whlie et al. developed a survivable storage system which guarantees that the data is persist, continuously accessible, cannot be destroyed, and kept confidential [62]. Leung and Miller proposed a scalable and efficient protocol for security in high performance storage systems, which increase the performance without sacrificing security primitives [63]. Miller et al. developed scheme to secure network-attached

storage systems against any type of attacks [19]. They used cryptography scheme to hide the data from unauthorized users.

However, the aforementioned techniques are not appropriate for disk requests with desired response times, because the existing techniques have no means of choosing security services to meet disk requests' flexible security requirements. Our strategy presented in Chapters 4 and 5 remedy this situation for local disk systems and parallel disk systems, respectively. Our goal is to propose an adaptive quality of security control mechanism that can judiciously choose the most appropriate security service for each write request while making the best effort to guarantee the desired response time of all disk requests. Note that experimental results reported in Chapters 4 and 5 were recently published in a number of journal and conference papers (see, for example, [64][65][66][67]).

2.2.2 Quality of Security

Quality of security can be considered as a dimension of Quality of Service (QoS). Several years ago, Irvine and Levin first presented an approach to managing security as a dimension of quality of service. More specifically, they developed a theory of Quality of Security Service and a related security framework that supports extension of quality of security service functionality to embrace existing and emerging security technologies [68]. Since sender and receiver in multicast network might own different security services to communicate with each other, different security services must be negotiated upon the senders and the receivers before the communication is established. Xia et al. proposed a novel mechanism to provide needed mechanisms of quality of security service to dynamically negotiate quality of security upon senders and receivers in the network [69]. Huang et al. proposed a middleware-oriented Global Resource Management System, or GRMS, which

provides distributed applications with end-to-end QOS negotiation and adaptation [70]. In addition, Xie et al. proposed a dynamic scheduling algorithm with security awareness that is capable of achieving high quality for real-time tasks while improving resource utilization [71]. In [65][67], we have proposed two algorithms with security awareness. Our schemes are capable of achieving high quality of security for data-intensive applications in local disk systems and parallel disk systems, respectively.

2.3 Cache Management

Modern disk storage systems make extensive use of cache memory to increase the performance of storage systems. In this section, we summarize the work that has been done to increase the performance of storage systems using caches in section 2.3.1. Then, Section 2.3.2 describes the previous studies related to cache management.

2.3.1 High Performance Storage Systems

Modern storage systems make use of cache memory to increase the performance. Large write caches used to improve storage system throughput by taking advantage of both temporal and special localities as data may be overwritten several times or combined together before being written to disks [72]. Researchers have been investigating several ways of how to use the cache memory to increase the performance of the storage systems. Hu et al. presented a new cache architecture called Redundant, Asymmetrically parallel, and inexpensive disk cache (RAPID-Cache) to provide fault tolerant caching for disk I/O systems inexpensively. . RAPID cache is used to increase the performance of storage systems by present-

ing a backup cache. The RAPID cache presents an asymmetric architecture with a fast write fast read being the primary cache and a fast write slow read being the backup cache [72]. Increasing the cache hit ratio will gradually increases the response time, which results in increasing the performance of storage systems. Ou et al. proposed a unified cache (uCache), for multilevel I/O systems to provide high cumulative hit ratios in multiple storage client cache systems, for both high correlated and low correlated workloads [73].

Direct map caching suffers from high miss rate ratio when compared to the set associative cache, but direct map cache is more attractive for high-speed processors that require very low access time. Jouppi proposed the victim cache [74] that stores cache block evicted from the man cache as a results of replacements. Victim cache is used to improve the miss rate of the direct map cache to increase the performance of the storage systems without affecting their access time. Stiliadis and Varama proposed an improvement to the victim cache called Selective Victim Caching [75]. In the selective victim caching, incoming blocks into the first level cache are placed selectively in the main cache or a small victim cache by the use of a prediction scheme based on their past history of use.

2.3.2 Cache Management

In modern storage system, not all requests to storage systems go directly to disks. Modern Storage systems use caches to improve the performance of storage systems. For instance, the EMC symmetrix storage systems use up to 128 GB of main memory as storage cache [56]. Zhu et al. proposed algorithm called power-aware replacement algorithm or PA-LRU [55], which selectively keeps blocks from inactive disks in caches longer, thereby increasing idle periods for disks that can be powered down for a longer time periods. Zhu et al. also developed another

algorithm, which estimates the correlation between the disk's energy consumption and its cache partition size. Their algorithm can dynamically divide a cache space into separate partitions such that the energy consumption of storage systems can be minimized [55].

There has been some work on applying caches to boosting performance of parallel disk systems. Kotz and Ellis investigated cache management techniques used in parallel file systems to close the performance gap between processor and disk speeds [61]. Karedla et al. examined the use of caching as a means to reduce system response times and to improve data throughput of disk systems [76]. Although the above cache management approaches can achieve high performance in parallel disk systems, the existing caching techniques are not focused on optimizing quality of security for storage systems. To our best knowledge, our cache management scheme presented in chapter 6 is the first of its kind to make use of caches to improve security and performance for cluster storage systems. In the following Section, we summarize related work in the arena of cluster storage systems.

2.4 Cluster Storage Systems

Cluster storage systems are an important and essential building block for any high-performance cluster computing infrastructures. In next-generation data grids, some of clusters will be dedicated to data storage systems to support distributed data manipulation functionalities [83]. In section 2.4.1 we summarize the applications that are running on clusters. In section 2.4.2, we the related work that has been done on PVFS which considered as an example of parallel file system.

2.4.1 Application on Cluster

A cluster is a group of computers, which are loosely connected together to provide fast and reliable services. There are many applications built on cluster systems such as distributed and parallel database applications, telecommunications systems, and internet/intranet servers [84]. Cluster systems are cost-effective computing platforms that maintain higher performance and reliability that traditional mainframes, supercomputers and fault-tolerant systems. Developing reliable applications on a cluster is not an easy task. However, researchers have been investigating various ways to develop reliable applications on cluster. Huang developed reliable telecommunication services on cluster computing systems [84]. In cluster systems, scheduling [85] and load balancing mechanisms [86] are used to improve the performance of parallel applications by fully utilizing computing nodes in clusters. A number of distributed load balancing mechanisms for clusters were developed. Very recently, Qin et al. proposed two I/O-aware load-balancing schemes to improve the overall performance of a cluster system with a general and practical workload including I/O activities [86].

Recently, increasing attention has been directed toward the issue of security in clusters. Apvrille and Pourzandi developed a new security algorithm named distributed security policy (DSP) for clusters. Yurcik et al. developed tools for managing cluster security via process monitoring [46]. Connelly and Chien proposed an approach to protect tightly coupled, high-performance component communication [1]. Xie and Qin proposed a security-aware real time heuristic strategy for clusters or (SAREC) which integrates security requirements into the scheduling for real-time applications on cluster [88]. However, to the best of our knowledge, the security issue in cluster storage system has not been addressed

in the literature. In chapter 6, we developed a security-aware cache management mechanism for cluster storage systems. Our algorithm aims at achieving high security and desired performance for data-intensive applications running on clusters.

2.4.2 Parallel File Systems

Parallel file systems are used to deliver high performance and scalable storage by connecting independent storage devices together. In this section, we are focused on parallel file systems developed for cluster computing platforms.

An important example of parallel file systems designed and implemented for high-performance cluster computing environments is parallel virtual file system or PVFS for short [91]. There were several studies that have been done on the PVFS parallel file system. For example, Vilayannur et al. proposed a kernel level caching to improve the I/O performance of concurrently executing processes in PVFS [93]. Apon et al. described measurement tests of PVFS and Network File System (NFS) over commodity Linux cluster connected with Myrinet [94]. Ligont et. al designed a scheduling scheme so that the service order of different requests on each server is determined by their locations in the space of logical block Address [95]. Wu et al. implemented a transport layer for PVFS by trading transparency and generality for performance [96].

Fault-tolerance is considers as one of the most important issues for parallel file systems. Zhu et al. developed and implemented a cost-effective, fault-tolerant parallel virtual file system (CEFT-PVFS), which provides parallel I/O services [89]. Moreover, Zhu et. al implemented a scheme to optimize the write performance of CEFT-PVFS [92]. Calderon et. al described a fault-tolerance support provided by Expand - a parallel file system based on standard servers. Basically,

Expand can be used to define different fault-tolerant mechanisms at the level of file systems [90].

Chapter 3

A Quality of Security Framework

Securing storage resources in Grid environments has increasingly become a major concern to make Grids attractive for a wide range of data-intensive applications, because security requirements are often imposed on storage resources to support security-critical applications. However, existing storage systems are unable to dynamically adjust quality of security to meet the flexible security needs of complex data-intensive applications. To remedy this deficiency, we propose in this paper a quality of security framework or StReD for storage resources in Data Grids. In this framework, we integrate quality of security adaptor with an array of security services. Applications running in the framework are enabled to specify flexible security requirements and desired response times of disk requests. The framework leverages the adaptor to dynamically control quality of security for disk requests, thereby achieving good tradeoffs between security and storage system throughput. Experimental results based on a simulated Grid with storage resources show that the proposed framework is capable of significantly improving quality of security and guaranteeing desired response times of disk requests.

The rest of the chapter is organized as follows. Section 3.1 presents a Data Grid model. Section 3.2 describes the control framework. Section 3.3 introduces a way of quantifying quality of security. In Section 3.4, an analytical model is

built to estimate the response time of each disk request. Section 3.5 evaluates the effectiveness of the proposed framework using a simulated disk system. Section 3.6 concludes the chapter with summary and future research directions.

3.1 System Model

In this chapter we propose a Data Grid model, which can be envisioned as a collection of storage subsystems. We model a Data Grid as a network of n storage subsystems with various system architectures. Each storage subsystem consists of data stores and computational facilities. Fig.3.1 shows the block diagram of the system model of a Data Grid. A job manager in each storage subsystem accommodates data-intensive jobs submitted to the Data Grid. The job manger aims at tracking load information by periodically changing load status with other job manager in the Data Grid. The incoming data-intensive jobs are placed into a waiting queue managed by a job scheduler in each subsystem.

The complete functionality of the quality of security manager is two-fold. First, the manager chooses the most appropriate security services provided in the security mechanism for disk requests. Second, the manager selects the most suitable security services in a judicious way to guarantee the security and the timely requirements for disk requests.

3.2 Quality of Security Framework

In this section we propose a quality of security framework, which is a development environment for security-critical applications. The framework is constructed with the goals of providing application programmers with an efficient way to adjust the security levels and the desired response time for their applications in a timely

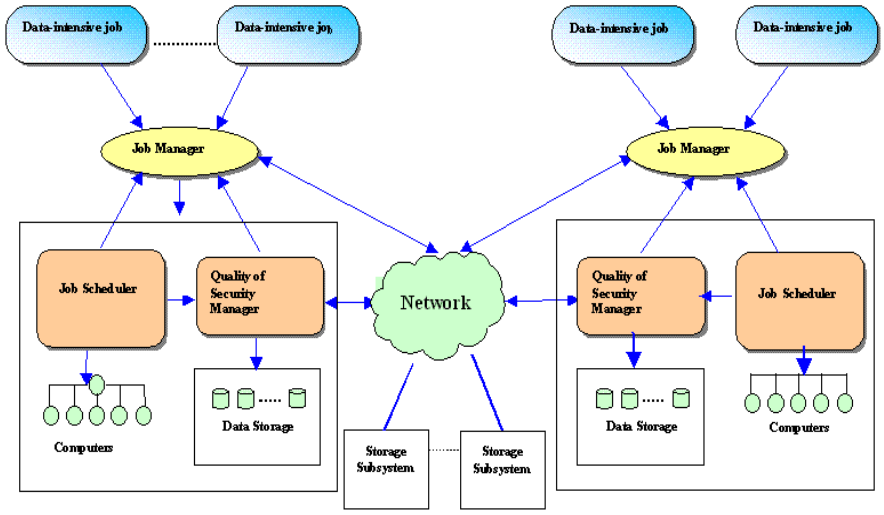


Figure 3.1: Block Diagram of the System Model of a Data Grid

manner to meet both the timely and security requirements. With the framework in place, end-users are allowed to dynamically adjust security levels and desired response times for applications in accordance with changing workload conditions.

The proposed framework for disk systems consists of requirement specification APIs, a quality of security adaptor, a security mechanism, a response time estimator, a low-level disk I/O APIs, and a disk system (see Fig.3.2). Disk requests can specify their flexible security and timely requirements using the specification APIs. The APIs are used to specify security and timing parameters, defining how these parameters may be degraded under high system load. The specification API layer, which is an abstraction of underlying security services, are implemented in light of the security services coupled together to form a security mechanism. Disk requests' security and timing requirements, defined as level of security services and desired response times, are derived from corresponding data-intensive applications. A security level specified by applications may be high, medium, or low. The desired response time of a disk request is the time at which the request must

be completed.

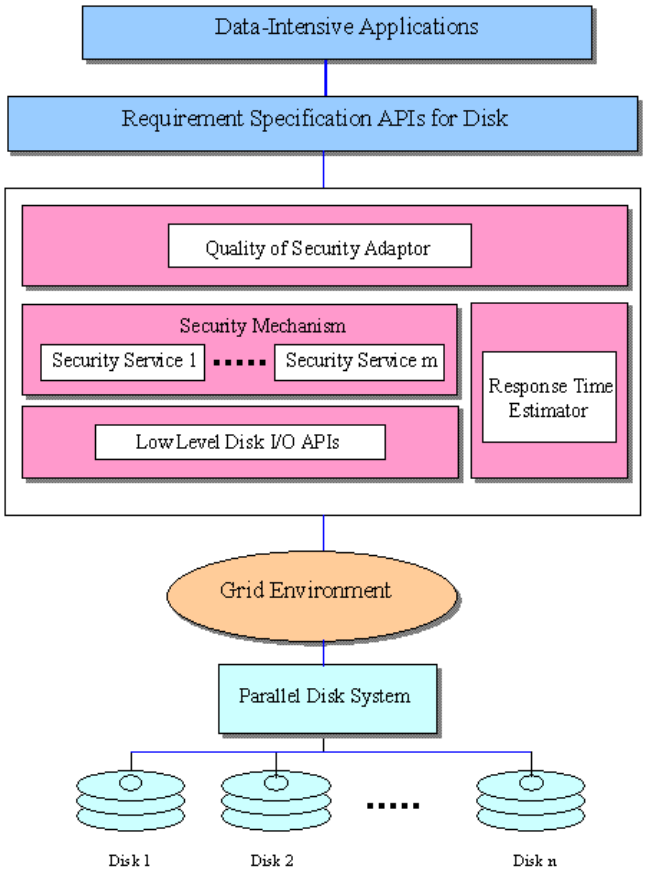


Figure 3.2: A Quality of Security Framework for Disk System in a Data Grid

Application developers are enabled to make use of the requirement specification API to implement applications with high flexibility in response time and security service selection. As such, applications with flexible security and timing constraints can be efficiently implemented using the proposed framework.

The security mechanism in the framework is comprised of an array of security services with various security levels to meet applications' flexible security needs. Commonly used security services include but not limited to authentication service, auditing service, encryption service and access controller. Quality of security provided by each service largely depends on the robustness of the mechanism used to implement the service, on how long the service has been used, and on how rigorously the mechanism was tested.

The security mechanism is constructed in a flexible way that new security services can be readily incorporated and old security services can be dropped. The quality of security adaptor is responsible for making use of specified security and timing constraints to choose appropriate security services in the security mechanism for disk requests. The adaptor aims at selecting security services in a judicious way to guarantee both security and timely requirements of disk requests. The input of the adaptor is security and timing constraints of disk requests specified by the requirement specification APIS, whereas the output is an appropriate security service that can meet both the security and timely requirements.

The design of the quality of the security adaptor is a challenging task, because the adaptor has to optimize the quality of security for each disk request based on dynamically changing workload characteristics. The first step toward the development of the adaptor is an approach to predicting the response time of a disk request. The response time estimator, which is focused on estimate response times, plays a critical role in the framework. We analytically present the response time estimator in Section 3.4.

Framework-private services provide specific functions in addition to the underlying middleware services to meet framework's own needs. Low-level disk I/O APIs are employed to perform disk access operations, including read and write operations. The low-level disk access operations in most cases are implemented at operating system kernel level.

3.3 Modeling Quality of Security

3.3.1 Quality of Security

Recall that the proposed framework encompasses an array of security services providing various quality of security. Each service is characterized by its quality of security measured by security level ranging from 0.1 to 1.0 [35].

Thus, the higher the security level of a service, the better the security quality of the service. Security services generally fall into three categories: confidentiality, integrity, and availability. Without loss of generality, in our quality of security model we address the confidentiality issues by employing nine cryptographic algorithms in our framework [65]. Note that the quality of security model can be easily extended to incorporate the other two security service categories. We assume that the clients, network, and parallel disk system are always available by the virtue of fault-tolerant mechanisms residing in these components. This assumption is valid, because the overhead of supporting reliability in the system can be envisioned as a part of security overhead.

Security overheads incurred by the cryptographic algorithms depend on size of data to be encrypted and performance of the algorithms, each of which is assigned a security level as shown in table 3.1. In accordance with the cryptographic algorithms' performance, each algorithm is assigned a security level in the range from 0 to 1. For example, we assign security level 1 to the strongest yet slowest encryption algorithm IDEA (see Table 3.1). Security levels for the rest algorithms can be computed by Equation 3.1, where μ_i^c is the performance of the ith ($1 \leq i \leq 8$) encryption algorithm.

$$sl_i^c = 13.5/\mu_i^c, 1 \leq i \leq 8 \tag{3.1}$$

Table 3.1: **Cryptographic Algorithms used for Encryption Services**

Cryptographic Algorithms	Security Level	Performance (Mb/sec)
SEAL	0.1	16.64
WAKE	0.2	14.12
RC4	0.3	3.06
Blowfish	0.4	1.62
RC5	0.5	1.42
SAFER	0.6	1.31
Rijndael	0.7	0.53

Security levels of the algorithms are proportional to the algorithms' performance [39]. Since computation overhead caused by encryption mainly depends on the cryptographic algorithms used and the size of data to be protected, Figure 3.3 shows encryption time in seconds as a function of encryption algorithms and size of secured data as measured on a 175 MHz Dec Alpha600 machine [20].

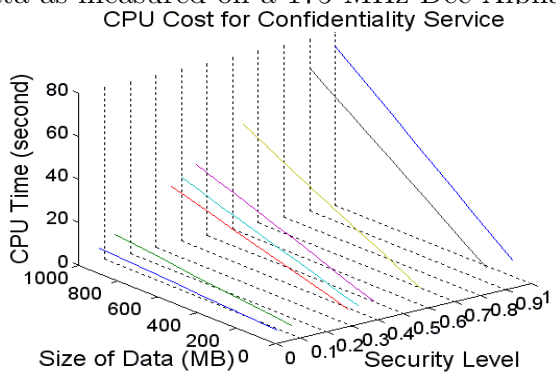


Figure 3.3: A Quality of Security Framework for Disk System in a Data Grid

Let S_i^e be the confidentiality security level of task T_i and the computation overhead of a selected confidentiality service can be calculated using Equation

3.2, were l_i is the amount of data whose confidentiality must be guaranteed, and $\sigma^c(s_i^c)$is a function used to map a security level to its corresponding encryption method's performance.

$$c_i^c(s_i^c) = l_i/\sigma^c(s_i^c), 1 \leq i \leq 8 \tag{3.2}$$

3.3.2 Security and timing constraints

We consider in this study data-intensive applications with both security and performance constraints, meaning that disk requests issued by the applications to a parallel disk system impose both security and performance requirements. While the security requirement of a request, e.g., r_i, is specified by a lower bound s_i on security level that the networked disk system has to provide. Hence, the security service controller must ensure that σ_i is greater than or equal to s_i. In this chapter, we investigate desired response time, which is a specific performance requirement; and the desired response time of request r_i is represented by t_i. We denote parallelism degree of r_i by p_i. It is worth noting that parallelism degrees play a critical role in performance tuning of networked parallel disk systems. As such, it is appealing to devise the data partitioning mechanism to automatically determine a parallelism degree for each request in a way to improve throughput of the system. Given parallelism degrees of requests, quality of security for the requests can be tuned in a judicious manner by the security service controller.

The first step toward improving quality of security is to quantitatively measure security benefits of a disk request. To achieve this goal, we calculate the security benefit of request r_i using the following security level function.

$$s(r_i) = \sum_{j=1}^{p_i} \sigma_{ij}, \sigma_{ij} \geq s_i \; and \; p_i \leq m \tag{3.3}$$

where m is the number of disks in the parallel disk system and σ_{ij} is the security level of a confidentiality service chosen for the jth stripe unit of r_i.

Given a sequence of requests $R = (r_1, ..., r_n)$, we can obtain the security benefits experienced by the requests. Thus, we have

$$S(R) = \sum_{i=1}^{n} S(r_i) \tag{3.4}$$

Now we obtain the following non-linear optimization problem formulation to compute the optimal security benefit of the networked parallel disk system.

$$maximize \;\; S(R) = \sum_{i=1}^{n} \sum_{j=1}^{p_i} \sigma_{ij}$$

subject to

$$(a) \;\; \forall 1 \leq i \leq n : max_1 \leq j \leq p_i \{\theta_{ij}\} \leq t_i,$$

$$(b) \;\; \sigma_{ij} \geq s_i \; and \; p_i \leq m \tag{3.5}$$

where θ_{ij} is the response time for the jth striping unit of request r_i. In an effort to enhance security of the system, we have to guarantee that the following three conditions are met. First, the response time of all striping unit in request

r_i must be smaller than the desired response time. Second, the low bound on security level can not be violated. Third, the parallelism degree of r_i has to be smaller than or equal to the number of disks in the system.

When a disk request is issued by a client to a storage subsystem in the Data Grid, the proposed framework of the storage subsystem inserts the newly arrived request into a waiting queue based on the requests' earliest desired response times. The maximum response time of the stripe units of the disk requests is estimated using the response time estimator. The StReD algorithm keep increasing the security level of each stripe unit of the disk request until it can not meet the desired response time for the disk request.

3.4 Response Time Estimator

To adaptively adjust security levels for disk requests, we need to estimate each request's maximum response time, which is defined as the interval between the time a request is sent by a client and the time the parallel disk system completes corresponding disk I/O operations. Given a newly issued request r, the response time of r is estimated by Eq. (3.6).

$$T(r, p, \sigma) = T_{queue} + T_{partition} + max_{i=1}^{p}\{T_{proc}^{i}(r, p, \sigma_i)\} \tag{3.6}$$

where p is the parallelism degree determined by the data partitioning mechanism (see Eq. 3.6), $\sigma = (\sigma_1, \sigma_2, ..., \sigma_p)$ is the request's vector of security levels for p stripe units, T_{queue} is the queueing delay at the client side, $T_{partition}$ is the time spent in data partitioning, and T_{proc}^{i} is the system processing delay experienced by the ith stripe unit of the request. With respect to the ith stripe unit of the

request, the system processing delay T^i_{proc} can be expressed as:

$$T^i_{poc}(r, p, \sigma_i) = T^i_{security}(r, p, \sigma_i) + T^i_{disk}(r, p, \sigma_i) \tag{3.7}$$

where $T^i_{security}$,$T^i_{network}$, and T^i_{disk} are the delays at the security mechanism, network subsystem, and parallel disk subsystems, respectively.

The delay at the security mechanism, which is also referred to as security overhead, depends on the assigned security level and data size of the stripe unit. Thus, we can easily derive $T^i_{security}(r, p, \sigma_i)$ from Eq. (3.2) as

$$T^i_{security}(r, p, \sigma_i) = T_{security}(\sigma_i, \frac{d}{p}) = \frac{d}{p.P(\sigma_i)} \tag{3.8}$$

where d is the data size of the request, and d/p is the data size of the ith stripe unit.

We assume that when the ith stripe unit of a request arrives at the network queue, there are k stripe units waiting to be delivered to the parallel disk subsystem. Suppose stripe units are transmitted in a first-in-first-out order, all the stripe units that are already in the queue prior to the arrival of the ith stripe unit must be transmitted earlier than the ith stripe unit. Hence, the delay in the network subsystem $T^i_{network}(r, p, \sigma_i)$ can be written as:

$$T^i_{network}(r, p, \sigma_i) = \frac{i.\frac{d}{p} + \sum_{j=1}^{k} d_j}{B_{network}} \tag{3.9}$$

where d_j is the data size of the jth stripe unit in the network queue, and

$B_{network}$ is the effective network bandwidth.

Similarly, it is assumed that when the ith stripe unit of the request arrives at disk j, there are k disk requests must be processed by disk j before handling the stripe unit. Thus, the delay in the disk subsystem $T^i_{disk}(r, p, \sigma_i)$ is given by the following formula:

$$T^i_{disk}(r, p, \sigma_i) = T_{disk,j}(\frac{d}{p}) + \sum_{l=1}^{k} T_{disk,j}(d_l) \tag{3.10}$$

where $T_{disk,j}(d)$ is the disk processing time of a request containing d bytes of data. We can quantify $T_{disk,j}(d)$ as follows

$$T_{disk,j}(d) = t_{seek} + T_{rot} + \frac{d}{B_{disk}} \tag{3.11}$$

where T_{seek} and T_{rot} are the seek time and rotational latency, and $\frac{d}{B_{disk}}$ is the data transfer time depending on the data size d and disk bandwidth B_{disk}.

3.5 Experimental Results

The proposed framework takes full advantage of the quality of security adaptor to guarantee a diversity of security requirements while finishing disk requests before their desired response times in a Data Grid. To evaluate the effectiveness of the framework, in this section we compare a simulated Data Grid where our framework is integrated with another Data Grid without employing the framework. In our simulation experiments, we made use of the following three metrics to demonstrate the effectiveness of the proposed scheme.

- *Satisfied ratio* is a fraction of total arrived disk requests that are found to be finished before their desired response times.

- *Security level* is a sum of security level values of all disk requests issued to the parallel disk systems.

- *Overall performance* is performance metric measured by a product of the satisfied ratio and the security level.

3.5.1 Overall Performance

This experiment aims to compare a simulated Data Grid with the StReD framework against the same Data Grid that makes no use of StReD. Note that storage subsystems used in our simulations consist of parallel disks. With different settings of parallelism degrees and data sizes, we study the impacts of varying arrival rates on system performance. To achieve this goal, we increased the arrival rate of I/O requests from 0.1 to 0.5 No./Sec. while setting the parallelism degree to 3 and data size to 100KB, 10MB, and 100 MB, respectively.

Fig. 3.4 plots the satisfied ratios, security levels, and overall performance of a storage subsystem with and without StReD. Figs 3.4(aa), (ba) and (ca) reveal that the StReD framework yields satisfied ratios that are very close to those of the storage subsystems making no use of StReD. This is mainly because StReD endeavors to guarantee timing constraints of disk requests while maximizing security of parallel disks in storage subsystems. Figs. 3.4(ab), 3.4(bb), and 3.4(cb) illustrate that StReD significantly improves the quality of security of the storage subsystems by an average of 126%. We can attribute the improvement in security to the fact that StReD strives to increase security level of each parallel disk request provided that the corresponding real-time requirement can be met. It is

observed that as the value of arrival rate increases, the security levels of the both systems decrease. This result is not surprising because high arrival rates lead to high workload, forcing the storage subsystems to merely meet the minimal security requirements of a vast majority of requests in order to process a large number of requests in a timely manner.

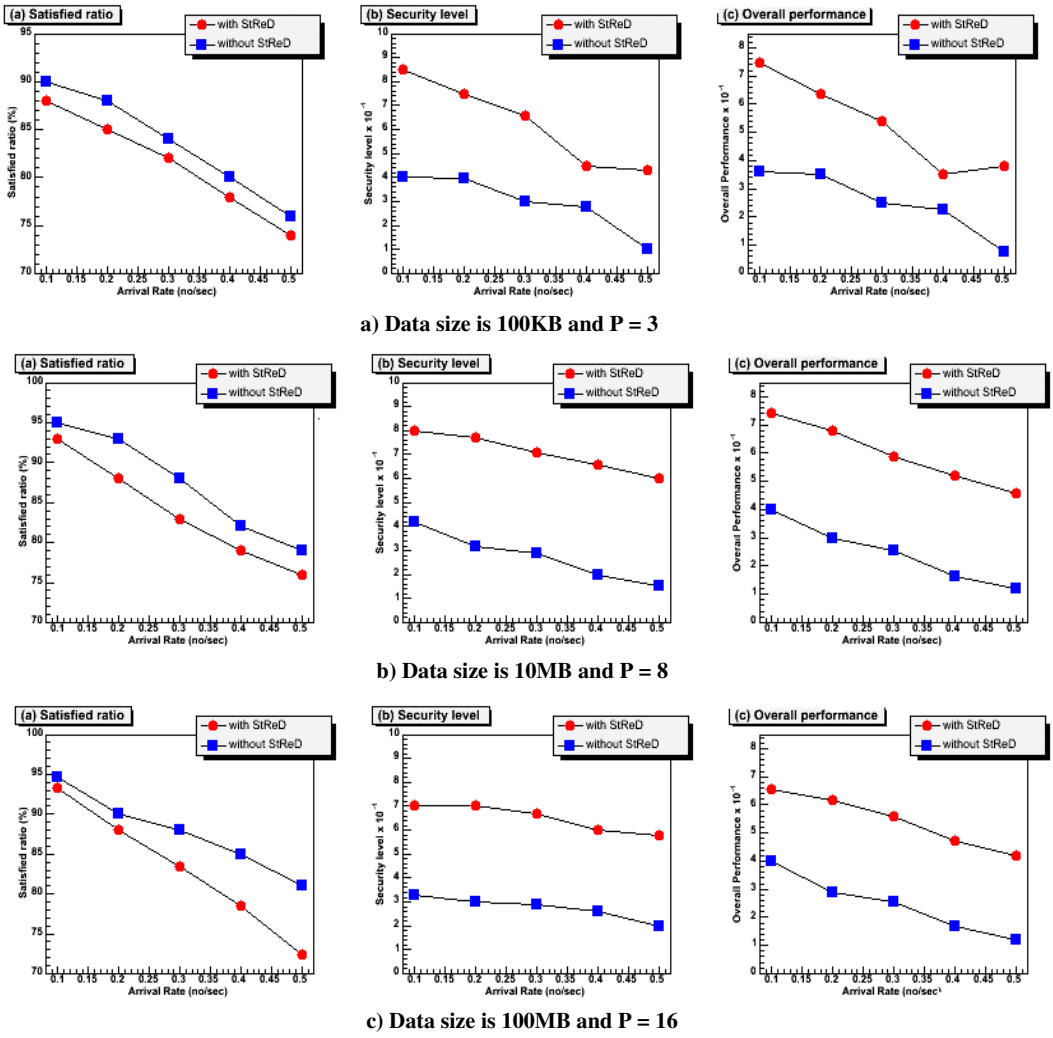


Figure 3.4: Impact of arrival rate on performance of storage subsystems in a Data Grid

Interestingly, StReD always achieves higher security levels compared with the storage subsystems without StReD. It is worth noting that the security improvement comes at the cost of satisfied ratios (see Figs 3.4(aa), 3.4(ba) and

3.4(ca), because the satisfied ratios of StReD are slightly reduced due to relatively high security overhead caused by the StReD strategy. Figs 3.4(ac), 3.4(bc), and 3.4(cc) reveal that StReD substantially boosts the overall performance. The reason of the expected overall performance improvement is two-fold. First, StReD adaptively enhances the security levels for disk I/O requests. Second, the performance gains in security level can eventually offset the extra security overhead.

3.5.2 Impacts of security requirements

In this group of experiments, we investigate the performance impacts of security requirements imposed by storage subsystems with parallel disks. As mentioned earlier, the security requirement of each disk request is characterized by a security range varying from smin to smax. We varied smax from 0.5 to 0.9 while fixing smin to 0.1.

We observed from Figs. 3.5(aa), 3.5(ba), and 3.5(ca) that increasing the maximal security level of the security range leads to the decreasing values of satisfied ratio of storage subsystems in two Data Grids. This is because when security level requirements of disk requests are high, the parallel disks in the storage subsystems need to fulfill the security requirements with high security overheads, which in turn reduce satisfied ratios. Again, the satisfied ratios of StReD are reasonably close to those of the alternative strategy. It is observed form Figs. 3.5(ab), 3.5(bb), and 3.5(cb) that when smax goes up, the security levels of the two evaluated storage subsystems in the Data Grids gradually increase. This result implies that the security levels of the storage subsystems heavily rely on the security requirements of disk requests. Figs. 3.5(ac), 3.5(bc), and 3.5(cc) illustrate that the overall performance of the two Data Grids is improved with the increasing values of smax, because the overall performance is more sensitive

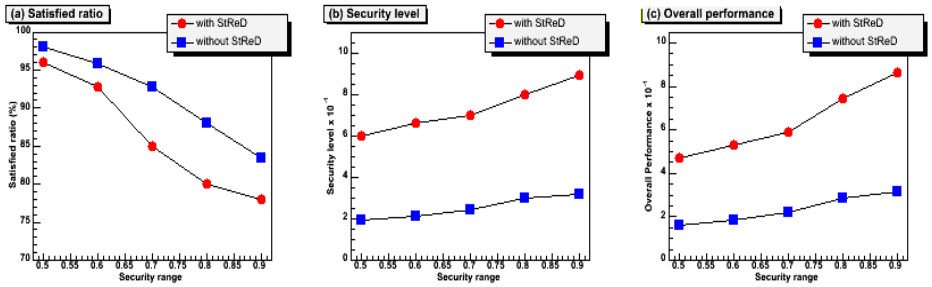
to security level than to satisfied ratio.

3.6 Summary

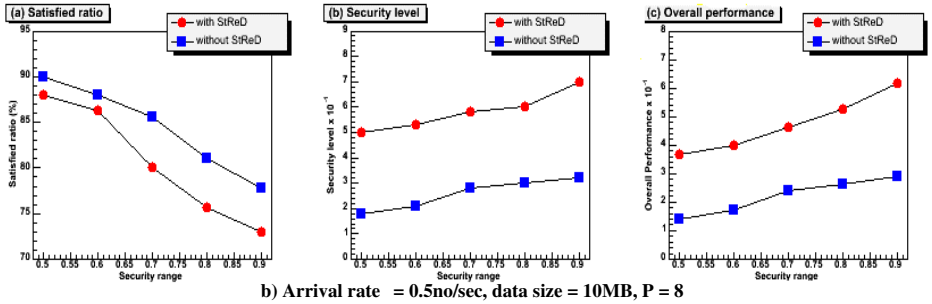
In this chapter, we presented a novel quality of security framework for storage resources in a Grid Environments. The quality of security adaptor, a centrepiece of the framework, is responsible for choosing the most appropriate security services for disk requests to guarantee both security and timely requirements. This adaptor paves the way to the design of applications with flexible security and timely requirements. The effectiveness of the proposed framework was evaluated by comparing the performance of a Grid with the framework against that of the same Grid without the framework. Experimental results based on disk traces show that the proposed framework is capable of significantly improving quality of security and guaranteeing desired response times of disk requests.

Future studies in this research can be performed in the following directions.

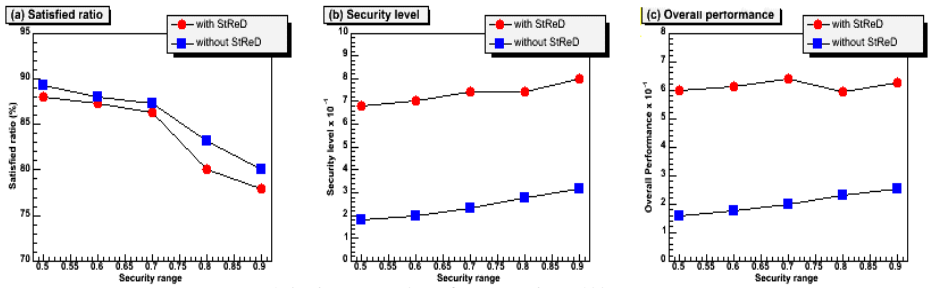
- we will implement the framework in a real world Grid.
- Second, we plan to integrate memory resources into our framework to further improve performance of data-intensive applications.
- we will extend the framework to deal with heterogeneous Data Grids, where different sites have different storage capabilities.



a) Arrival rat e = 0.5no/sec, data size = 100KB, P = 3



b) Arrival rate = 0.5no/sec, data size = 10MB, P = 8



c) Arrival rate = 0.5no/sec, data size = 100MB,

Figure 3.5: Impact of Security Range on performance of storage subsystems in a Data Grid

Chapter 4

Modeling and Improving Security of a Local Disk System for Write-Intensive Workloads

In the previous chapter, we have designed a Quality of Security Framework for storage resources in Data Grids which security-sensitive data-intensive applications are allowed to exploit an array of security services to enhance trustworthy executions of the applications. To make the framework presented in Chapter 3 more practical, we have to address the issue of security for both local disk systems and parallel disk systems.

This chapter is organized as follows. The architecture of the system and the disk requests are modeled in sections 4.2. In section 4.3, the AWARDS algorithm is described. We build an analytical model in section 4.4. Section 4.5 and 4.6 present the experimental results based on both synthetic benchmarks (read/write) and real I/O-intensive applications. Finally, Section 4.7 concludes this chapter with future directions.

4.1 Motivation

Security-critical data-intensive applications have mandatory security requirements in addition to stringent timing constraints. So it is mandatory to pro-

tect data stored in the storage systems of these applications from talented intruders. Thus, storage systems are required to provide strong security and guaranteed response times for disk requests. This demanding requirement is increasingly becoming a critical and challenging issue in the development of the next generation storage systems.

In this chapter, we seek to present a novel adaptive write strategy for local disk systems providing a diversity of security services with various quality of security. The strategy can be seamlessly integrated into disk scheduling mechanisms in disk systems. The proposed strategy is conducive to the achievement of high security for the local disk systems while making the best effort to guarantee desired response times of requests. To prove the efficiency of the proposed approach, we build an analytical model to measure the expected value of security levels and the probability that a disk request is completed before its desired response time.

4.2 Architecture and Disk Requests with Security Requirements

4.2.1 Architecture of a Security-Aware Storage System

In this study we focus on local disk systems, and storage systems in parallel and distributed environments are out the scope of this study. Our new algorithm is based on a security-aware storage system architecture. Fig. 4.1 depicts the main components, i.e., a disk driver, a security mechanism, and a disk scheduling core, of this architecture.

The architecture is briefly overviewed as follows. The disk driver is responsible for controlling access to an untrusted local disk. The security mechanism

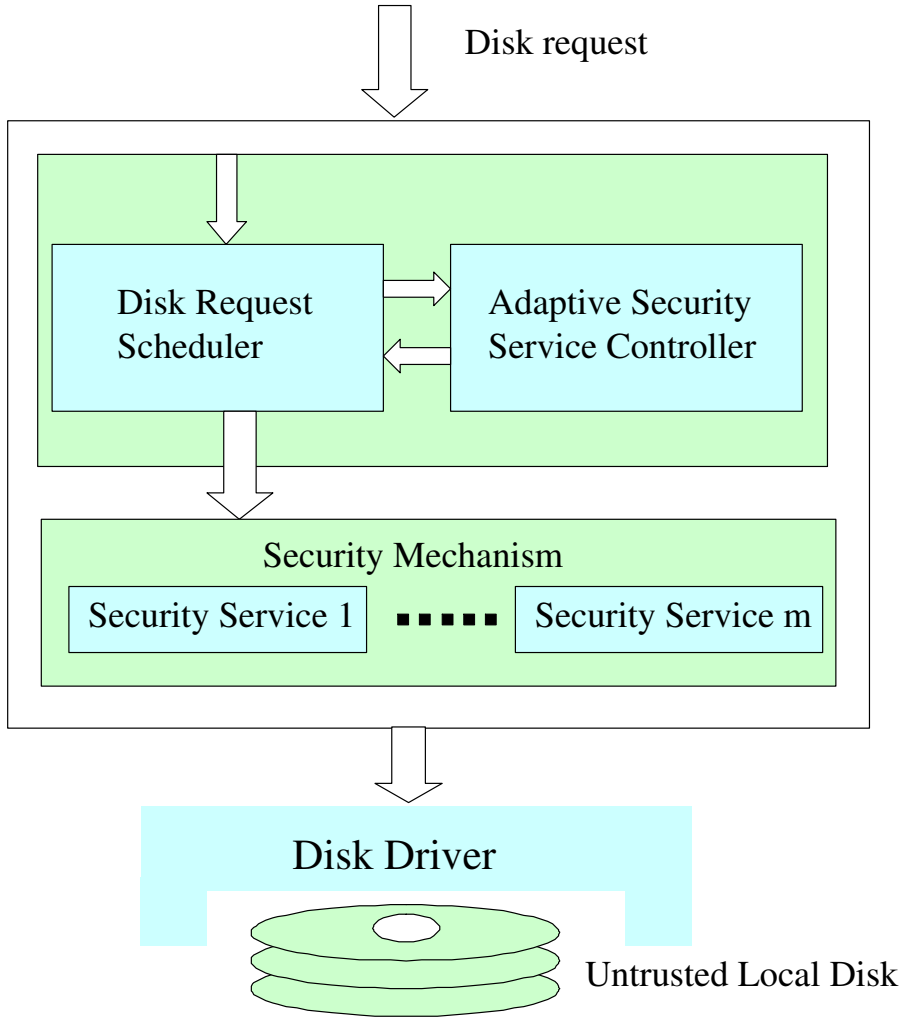


Figure 4.1: Architecture of a security-aware storage system

provides an array of security services to guard data blocks residing in the local disk against unauthorized access and information theft. Without loss of generality, we consider in this study only confidentiality security services, where it is assumed that keys are known only to the owner, reader and writers [4]. The security mechanism can be readily extended to employ integrity and availability services. The disk scheduling core consists of two parts: a disk scheduler and an adaptive security service controller. While the scheduler implements generic logic and timing mechanisms for scheduling and waiting, the security service controllor dynamically choose the most

appropriate security service for each disk request. Since the security service controller is independent of disk scheduling policies, the service controller is implemented separately for each disk scheduler. As such, it is easy to apply the security service controller to any disk scheduling policy implementation.

4.2.2 Modeling Disk Requests with Security Requirements

Each disk request submitted a security-aware storage system specifies quality of service requirements, including security and performance requirements. A security requirement is defined as a lower bound security level, which is a value from 0.1 to 1.0. A performance requirement is posed as a desired response time. The quality of service requirements of disk requests can be derived from applications issuing these disk operations. Security quality of encryption services implemented in the security mechanism is measured by security levels. An encryption service with a high security level means the high quality of security provided by the service. For example, a disk request specifies a lower bound security level as 0.4. In this case, encryption services with security levels higher than or equal to 0.4 can successfully meet the disk request's security requirements.

A disk request r is characterized by five parameters, $r = (o,\ a,\ d,\ s,\ t)$, where o indicates that the request is a read or write, a is the disk address, d is the data size measured in KB, s is the lower security level bound, and t is the desired response time.

The security benefit gained by a disk request r_i can be measured by the security level i of an encryption service facilitating confidentiality for the disk request. Likewise, the quality of security offered by a local storage

system can be measured by a sum of security benefit of all incoming disk requests. Let R be a set of incoming disk requests. Our proposed AWARDS strategy strives to maximize the security benefit of the storage system. Thus, we can obtain the following non-linear optimization problem formulation to maximize the security benefit, where i is the real response time of the ith disk request.

$$Maximize \sum_{r_i \in R} \sigma_i$$

$$Subject\ to\ \forall r_i \in R : s_i \leq \sigma_i \leq 1, and\ \rho_i \leq t_i \tag{4.1}$$

4.2.3 Security Overhead Model

Now we consider security overhead incurred by confidentiality services. The security overhead model can be easily extended to incorporate other security services. Encryption is used to encrypt data blocks residing in local storage systems. There are ten encryption algorithms (see Table 4.1) implemented in the security mechanism. Based on the encryption algorithms' performance, each algorithm is assigned a security level. For example, level 0.9 implies that we use 3DES, which is the strongest yet slowest encryption function among the alternatives. Note that overhead of encryption depends on the chosen cryptographic algorithm and the size of data block. Fig. 4.2 plots enciphering time in seconds as a function of the encryption algorithms and data size on a 175 MHz Dec Alpha600 machine [22][36][37]. Let σ_i be the security level of r_i, and the security overhead can be calculated using Eq. 4.2, where d_i is the data size and *P(i)* is a function used to map a security

level i to the corresponding performance of encryption service listed in Table 4.1.

$$T_{security}(\sigma_i, d_i) = \frac{d_i}{P(\sigma_i)} \tag{4.2}$$

Table 4.1: Cryptographic Algorithms Used for Encryption Services

Cryptographic Algorithms	Security Level	Performance (Mb/sec)
SEAL	0.1	16.64
WAKE	0.2	14.12
RC4	0.3	3.06
Blowfish	0.4	1.62
RC5	0.5	1.42
SAFER	0.6	1.31
Rijndael	0.7	0.53

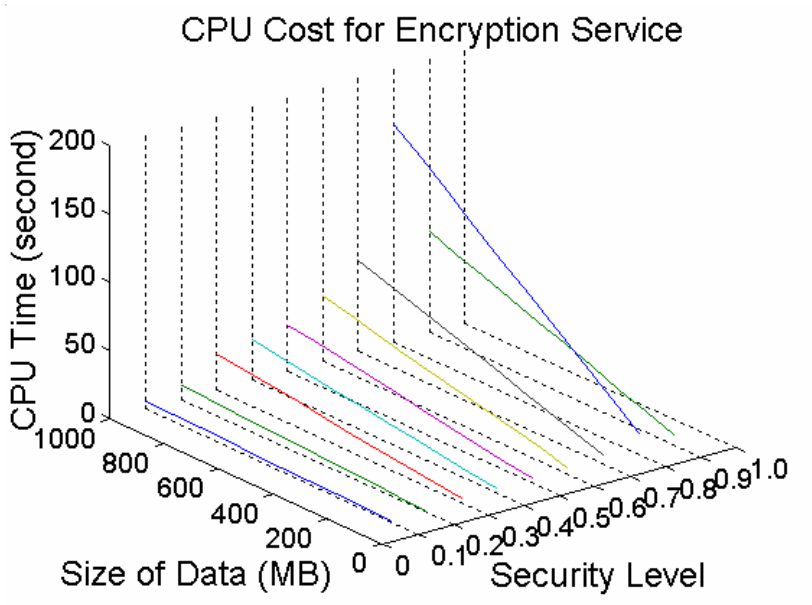


Figure 4.2: Security overhead of encryption services

4.3 The Adaptive Write Strategy

This section presents the proposed adaptive write strategy, which is referred to as AWARDS throughput this chapter. We assume that the overhead of AWARDS is negligible when compared to the processing times of disk requests. In this study we consider a local disk system providing nine encryption services with different security levels (see Section 4.3). AWARDS aims at improving quality of security for local disk systems. To achieve this goal, AWARDS aggressively increases the security level of each incoming disk request under the condition that the request's response time does not exceed the desired response time. We make use of an example to elaborate the basic idea behind the AWARDS strategy. Suppose there are three write requests submitted to a local disk system at time 0. Table 4.2 shows the important parameters of the three write requests. We assume that the disk bandwidth is 30MB/Sec., and for each request, the sum of rotational latency and seek time is 8ms.

Table 4.2: Important parameters of the three write requests. Requests(R), data size(d_i), Minimal security levels(s_i), Desired Response Time(t_i), Response time under AWARDS(T), Security Level under AWARDS(σ_i)

R	d_i	s_i	t_i	T	σ_i
r_1	90KB	0.2	18ms	17.7ms	0.8
r_2	150KB	0.1	41ms	40.7ms	0.7
r_3	30KB	0.3	55ms	54.5ms	0.9

Prior to writing the data of a request to the local disk, the disk system enciphers the data using a selected encryption service. Similarly, the system deciphers the data of a read request after the data is retrieves from the disk.

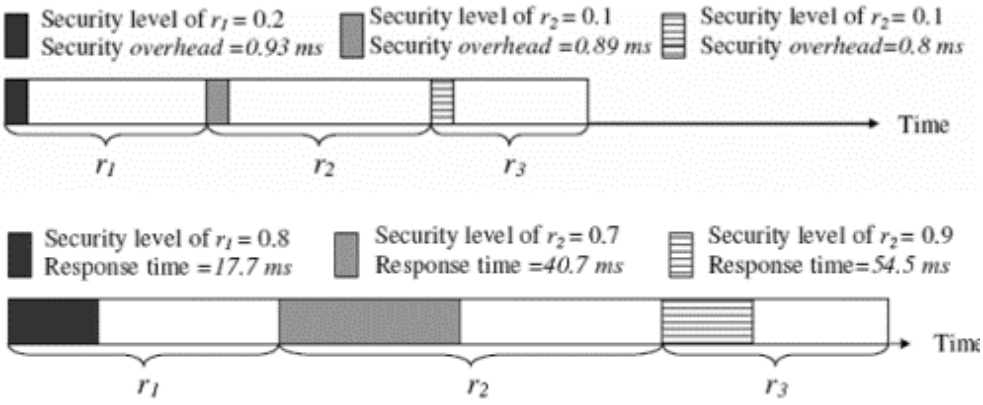


Figure 4.3: a)Security levels and overhead of the request(the system is running without AWARDS)
b)Security levels and response times of the request(the system is running with AWARDS)

Hence, the processing time of each disk request logically consists of two parts: security overhead indicated by a shaded block and disk service time represented by an unshaded block (see Eq. 4.1 and Fig. 4.3). Figs. 4.3a illustrates the security levels and response times of the requests when the disk system is running without AWARDS. In this case, the minimal security requirement of each request is met at the minimal security overhead. In contrast, AWARDS can significantly increase the security levels provided that the desired response times can be guaranteed (see Fig. 4.3b). For example, the response times of the three requests are 17.7ms, 40.7ms, and 54.5ms, which are less than the respective desired response times (see Table 4.2). Specifically, with the AWARDS strategy in place, the security levels are improved by an average of 366.7%.

Now we present the AWARDS strategy, which adaptively adjusts security levels of write requests (see Fig. 4.4). It is worth noting that AWARDS is unable to adjust security levels of read requests, because the local disk

system has to use a corresponding encryption service to decipher the data of a read request after the cipher is read from the disk. Before increasing the security level of a write request r_i, AWARDS must ensure that check that r_i and those write requests with earlier desired response times can be finished before their desired response times (see condition (4.2) in the following property). Therefore, the following property needs to be satisfied in the AWARDS strategy.

Property 1. If the security level of a write request r_i is increased by 0.1, the following conditions must hold:

(1) *The current security level of* r_i *is less than* 0.9,

$$i.e.,\ \sigma_i < 0.9; \tag{4.1}$$

(2) $$\forall r_k \in Q, t_k \geq t_i : es(r_k) + T(r_k, \sigma_k) \leq t_k; \tag{4.2}$$

where Q is the waiting queue, $es(r_k)$ is the start time of request r_k, and $T(r_k, \sigma_k)$ is the processing time of $r_i \in Q$. The start time of $es(r_k)$can be expressed by

$$es(r_k) = \sum_{r_l \in Q, t_l \leq t_l} T(r_l, \sigma_l) \tag{4.3}$$

where the expression on the right side of Eq. (4.3) is the totalprocessing time of disk requests whose desired response time is earlier than that of r_k.The processing time $T(r_k, \sigma_k)$ in condition (4.2) can becomputed by Eq. (4.4).

$$T(r_i, \sigma_i) = T_{seek}(a_i) + T_{rot}(a_i) + \frac{d_i}{B_{disk}} + T_{security}(d_i, \sigma_i) \qquad (4.4)$$

where T_{seek} and T_{rot} are the seek time and rotational latency, $\frac{d_i}{B_{disk}}$ is the data transfer time that largely depends on the data size d_i and the disk bandwidth B_{disk}, and $T_{security}(\sigma_i, d_i)$ is the security overhead that lies in the security level and security-critical data size (see Eq. (4.2)).

The adaptive write strategy for secure local disk systems, or AWARDS, is described in Fig. 4.4 AWARDS aims at improving quality of security and guaranteeing disk requests' desired response times. To achieve high security, the AWARDS strategy optimizes the security levels of write requests (see Step 7).

1. **Input:** r_i, a disk request r_i submitted to the local disk system; Q, a waiting queue
2. Insert r_i into Q based on the earliest desired response time first policy;
3. **for** each write request r_j in the waiting queue Q **do**
4. $\sigma_j \leftarrow s_j$; /* Initialize the security levels */
5. **for** each write request r_j in the waiting queue Q **do**
6. **if** $\sigma_j < 0.9$ **then** /* σ_j can be further increased */ (see **property 1(1)**)
7. Increase security level σ_j by 0.1;
8. **for** each request r_k with longer desired response time **do** (see **property 1(2)**)
9. **if** r_k can not be completed before its desired response time **then**
10. Decrease security level σ_j by 0.1; break;
11. **end for**
12. **if** σ_j is not increased in Step 7 **then** break;
13. **end for**

Figure 4.4: The Adaptive write strategy for local disk systems

Upon the arrival of a disk request, AWARDS insert the request into the waiting queue based on the earliest desired response time first policy, meaning that disk requests with earlier response time are processed first. Before proceeding to the optimization of security levels of write requests in the queue, AWARDS first initialize the security levels of all the write requests to the

minimal levels (see Step 4). Step 7 then gradually enhance the security level of each request r_j under the conditions that (1) the current security level of r_j does not exceed 0.9 (see Step 6); and (2) the desired response times of the requests being processed later than r_j can be achieved (see Steps 8-10). The process of optimizing security levels repeatedly performs (see Step 5); and stops when a request's desired response time can not be met (see Step 12). In doing so, the AWARDS strategy can maximize the security levels of write requests (see Step 7) while guaranteeing the desired response times of all the disk requests in the queue (see Steps 9 and 10). The time complexity of AWARDS is evaluated as follows.

Theorem 1. The time complexity of AWARDS is $O(n^2)$, where n is the number of disk requests in the waiting queue.

Proof. To increase the security level of a request, it takes $O(n)$ time check condition (4.2) (see Step 8). Since there are $O(n)$ number of write requests in the waiting queue, the time complexity of optimizing security levels of write requests is: $O(n)O(n) = O(n^2)$.

4.4 Analytical Model

In this section, we first derive the probability that a disk request is completed before its desired response time. Second, we calculate the expected value of security levels assigned to disk requests with security requirements.

4.4.1 Satisfied Ratio

We first calculate, for every disk request r_i submitted to a local disk system, the probability that r_i can be finished within the desired response time t_i, i.e., $Pr(\rho_i \leq t_i)$ where ρ_i is the real response time. At any time when a disk request arrives, the request is inserted in the queue in a way that all requests with earlier desired response times will be given higher priority and executed first. Note that n waiting requests in the queue are indexed by their priorities so that the desired response time of r_i is smaller than that of r_j if $i < j$, i.e., $\forall 1 \leq i, j \leq n : i < j \Rightarrow t_i < t_j$. In this study we assume that processing times of different disk requests are statistically independent.

Recall that $T(r_i,\sigma_i)$ is the processing time of a disk request $r_i \in Q$. $T(r_i, \sigma_i)$ is computed by Eq . 4.4. Let p_x be the probability that the disk request r_i requires x time unit to complete, i.e. $p_x = P_r(T(r_i, \sigma_i) = x)$. Similarly, let q_y be the probability that the total required processing time of disk requests with higher priorities is y,
i.e. $q_y = Pr[\sum_{j=1}^{i-1} T(r_j, \sigma_j) = y]$. The probabilities that r_i is unable to be finished within the desired response time t_i is computed as follows:

$$
\begin{aligned}
Pr(\rho_i > t_i) = Pr\Big[T(r_i, \sigma_i) = 1 \Big| \sum_{j=1}^{i-1} T(r_j, \sigma_j) \geq t_i\Big] + \\
\vdots \\
+Pr\Big[T(r_i, \sigma_i) = k \Big| \sum_{j=1}^{i-1} T(r_j, \sigma_j) \geq t_i + 1 - k\Big] + \\
\vdots \\
+Pr(\Big[T(r_i, \sigma_i) = t_i + 1 \Big| \sum_{j=1}^{i-1} T(r_j, \sigma_j) \geq 0\Big]
\end{aligned}
$$

$$= p1 \sum_{y=t_i}^{\infty} q_y + p2 \sum_{y=t_i-1}^{\infty} q_y + \cdots + p_{t_i} \sum_{y=1}^{\infty} q_y + p_{t_i+1} \sum_{y=0}^{\infty} q_y$$
$$= \sum_{x=1}^{t_i+1} \left[p_x \sum_{y=t_i+1-x}^{\infty} q_y \right] \quad (4.5)$$

where the second line on the above equation indicates the conditional probability that the required processing time of disk request r_i is k given that it requires at least t_i*+1-k* time unit to complete disk requests with higher priorities.

The probability that a disk request r_i is completed within its desired response time is given by

$$Pr(\rho_i \leq t_i) = 1 - Pr(\rho_i > t_i)$$
$$= 1 - \sum_{x=1}^{t_i+1} \left[p_x \sum_{y=t_i+1-x}^{\infty} q_y \right] \quad (4.6)$$

4.4.2 Quality of Security

To evaluate quality of security for a local disk system, we derive in this section the expected security level experienced by disk requests. Before proceeding to the calculation of the expected security level, we compute the probability $Pr(\sigma_i = z)$ that the security level of each submitted disk request r_i equals to z. Recall that the security level of a disk request relies on the desired response time, data size of the request, and processing times of other waiting requests with higher priorities. As such, $Pr(\sigma_i = z)$ can be calculated as

$Pr(\sigma_i = z) =$

$$
\begin{aligned}
& Pr(d_i = d_{min}). \sum_{j=t_{min}}^{t_{max}} \Big\{ Pr(t_i = j). Pr\Big[\sum_{k=1}^{j-1} T(r_k, \sigma_k) = j - \bar{T}(d_{min}, z) \Big] \Big\} \\
& \qquad \vdots \\
& + Pr(d_i = l). \sum_{j=t_{min}}^{t_{max}} \Big\{ Pr(t_i = j). Pr\Big[\sum_{k=1}^{i-1} T(r_k, \sigma_k) = j - \bar{T}(l, z) \Big] \Big\} \\
& \qquad \vdots \\
& + Pr(d_i = d_{max}). \sum_{j=t_{min}}^{t_{max}} \Big\{ Pr(t_i = j). Pr\Big[\sum_{k=1}^{i-1} T(r_k, \sigma_k) = j - \bar{T}(d_{max}, z) \Big] \Big\} \\
= & \sum_{l=d_{dmin}}^{d_{max}} \Big\{ Pr(d_i = l). \sum_{j=t_{min}}^{t_{max}} \Big\{ Pr(t_i = j). Pr\Big[\sum_{k=1}^{i-1} T(r_k, \sigma_k) = j - \bar{T}(l, z) \Big] \Big\} \Big\}
\end{aligned}
\tag{4.7}
$$

where $\bar{T}(l, z)$ is the processing time of r_i if the data size is l and security level is set to z, i.e.,

$$
\bar{T}(l, z) = T_{seek}(a_i) + T_{latency}(a_i) + [d_i / B_{disk}] + T_{security}(l, z) \tag{4.8}
$$

The data size and desired response time of each disk request are two random variables distributed according to two probability density functions, which are know a priori. We let u_l denote the probability that the data size of the disk request is l, and let v_j denote the probability that the desired response time equals to j. The minimal and maximal data sizes are represented by d_{min} and d_{max}. Likewise, the minimal and maximal desired response times are denoted by t_{min} and t_{max}. Based on Eq. 4.4, the probability $Pr(\sigma_i = z)$ can be expressed as

$$
Pr(\sigma_i = z) = \sum_{l=d_{dmin}}^{d_{max}} \Big\{ u_i . \sum_{j=t_{min}}^{t_{max}} [v_i . q_{j - \bar{T}(l,z)}] \Big\}, \tag{4.9}
$$

where $q_{i-\bar{T}(l,z)} = Pr\Big(\sum_{k=1}^{i-1} T(r_k, \sigma_k) = j - \bar{T}(l,z) \Big)$.

The expected security level experienced by disk requests with security requirements can be directly derived from Eq. 4.10. Thus, the expected security level is given by

$$E(\sigma) = \sum_{i=1}^{9} \Big([i/10].Pr(\sigma = [i/10]) \Big) \tag{4.10}$$

4.5 Synthetic Benchmarks

To quantitatively evaluate the performance of the AWARDS strategy, we implemented a basic prototype of AWARDS to work with a simulated local disk system. Workloads with both synthetic benchmarks and real world I/O intensive applications were generated and evaluated in the prototype of AWARDS.

Our experimental tested consist of the prototype, the simulated disk system, and the nine encryption services described in Table 4.2 Disk parameters summarized in Table 4.3 are similar to those of the IBM Utrastar 36Z15.

Table 4.3: Disk Parameters

IBM Ultrastar 36Z15	
Size	18.4 GB
RPM	15000
Seek Time, T_{seek}	7.18 ms
Rotational Time, T_{rot}	4.02 ms
Disk Bandwidth, B_{disk}	30 MB/Sec

The following four important performance metrics are used to evaluate the AWARDS strategy. (1) *Satisfied ratio* is defined as a fraction of total ar-

rived disk requests that are found to be completed before their desired response times. (2) *Average security level* is the average value of security levels achieved all disk requests issued to the disk system. (3) *Average security overhead* is measured in seconds. (4) *Overall performance* is measured by a product of the satisfied ratio and average security level.

To provide fair comparisons, we obtained empirical results based on a wide range of synthetically generated workload and environmental conditions, which closely resemble a variety of I/O intensive applications. Table 4.4 outlines the important workload configuration parameters for the simulated disk system used in our experiments.

Table 4.4: Workload Configuration

parameter	Value(Fixed)-(Varied)
Disk Bandwidth	30KB/s
Request Arrival Rate	(0.1,0.2,0.3,0.4,0.5) No./Sec.
Desired Response Time	10 Sec.
Security Level	(0.5)-(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)
Write Ratio	(100%)-(0%, 10%, 20%, 30%, 100%)
Data Size	(500 KB)-(300,400,500,600,700) KB

In Section 4.6 we present performance comparisons between a disk system with AWARDS and another system without employing AWARDS (referred to as the Original strategy). Section 4.6.1 studies the impact of data size on disk performance. Section 4.6.2 examines performance sensitivities to disk bandwidth. Performance impact of security requirements is evaluated in Section 4.6.3 Finally, Section 4.6.4 demonstrates the performance impact of increasing write ratio.

4.5.1 Overall Performance Comparisons

This experiment is aimed at comparing AWARDS against Original, which is a strategy without making use of AWARDS. To stringently evaluate the performance of AWARDS and its competitive strategy, we set write ratio to 100%. The impact of write ratio on system performance is studied in Section 6.5. We increased disk request arrival rate from 0.1 to 0.5 No./Sec. Other workload parameters were fixed to the same values as those listed in Table 4.

Figure 4.5 plots the four performances metrics for the AWARDS and Original strategies. Fig. 4.5a reveals that AWARDS maintains a very close performance in satisfied ratio to the Original strategy. Fig. 4.5b shows that AWARDS significantly outperforms Original in average security level by an average of 138.2%. As the request arrival rate increases, the average security levels of the two strategies decrease. However, AWARDS always achieves higher average security levels compared with those of the Original strategy. This result can be explained in part by Fig. 4.5c, which shows that the average security overhead of AWARDS is constantly higher than that of the alternative. In other words, high security levels of AWARDS are achieved at the cost of high security overheads. Fig. 4.5d clearly reveals that AWARDS noticeably outperforms the Original one in terms of overall performance. Specifically, AWARDS obtains an improvement in overall performance over the Original strategy by an average of 125.6%. The performance improvement can be attributed to the fact that AWARDS adaptively enhance security levels of each write requests under the condition that all requests in the disk can achieve their desired response times.

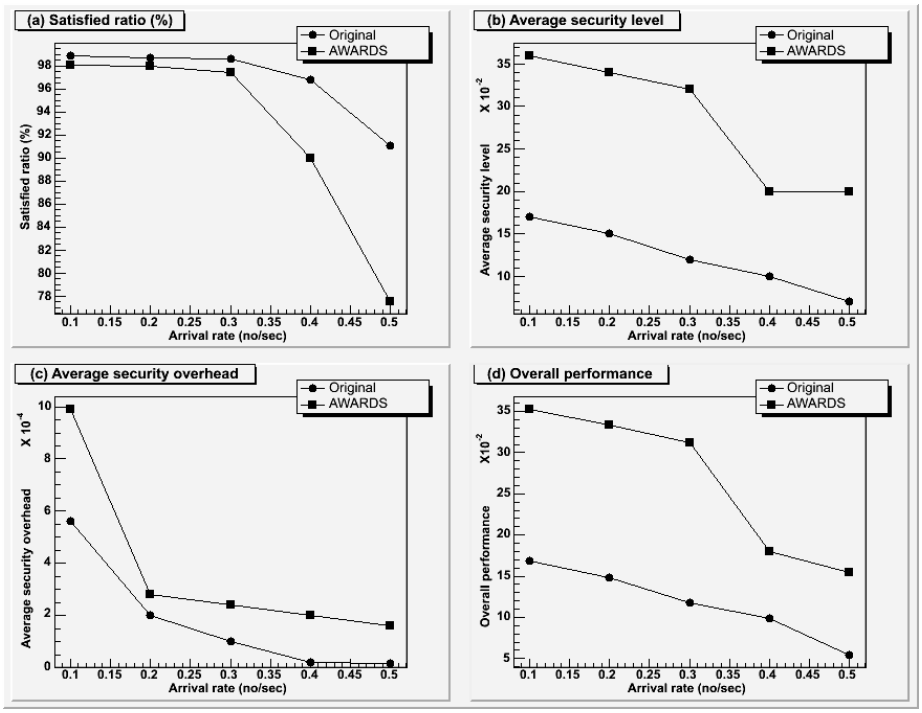


Figure 4.5: Performance impact of arrival rate

4.5.2 Impact of Data Size

In this section, we varied data size from 300 to 700 KB to examine the performance impact of data size on the local disk system. Again, other workload parameters were kept unchanged (see Table 4.4).

Fig. 4.6a shows that when the data size increases from 300 to 700 KB the AWARDS strategy delivers similar satisfied ratios as those of Original. This result, which is consistent with the result presented in Fig. 4.5a, demonstrates that AWARDS achieves a good performance in satisfied ratio. Like Fig. 4.5b, Fig. 4.6b shows a significant improvement of AWARDS in security level over Original.

Interestingly, it is observed from Fig.4.6b that the average security level gradually drops with the increasing value of data size. This is because

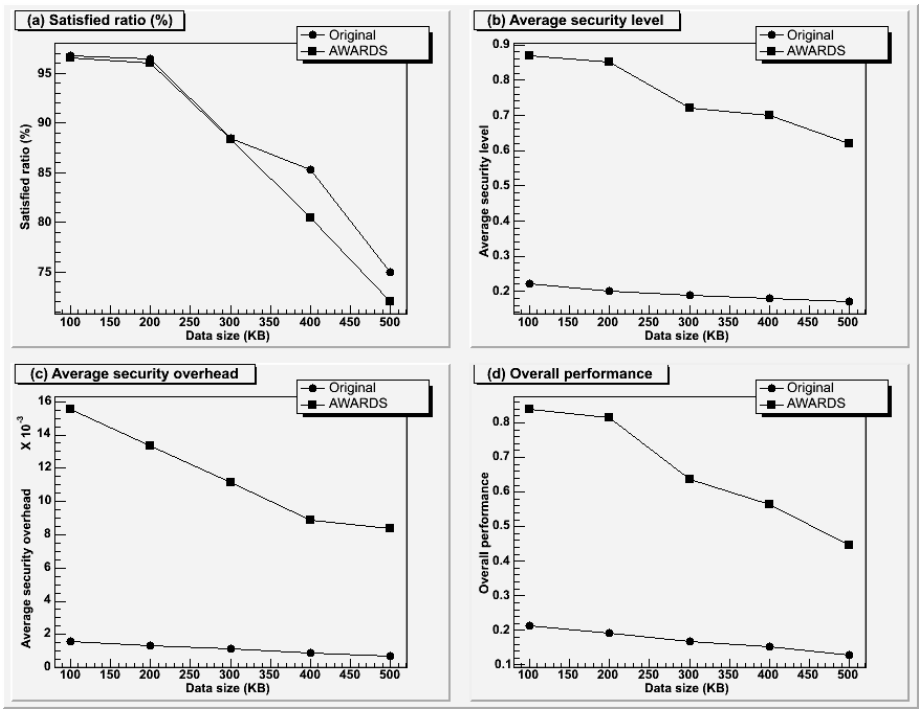


Figure 4.6: Performance impact of data size

the increasing data size results in a overloaded condition, which in turn leads to decreasing security levels of disk requests in the queue in order to finish most of the requests before their desired response time. Further, we observe from Figs. 4.6c and 4.6d that when the data size goes up, the average security overhead and overall performance of AWARDS decreases, because the security levels of disk requests are lowered down due to the high workload. By contrast, the average security level and overhead of the Original strategy only slight reduce with the increasing value of data size. These results indicates that Original is insensitive to data size.

4.5.3 Impact of Disk bandwidth

This experiment we investigated the performance of AWARDS and Original when the disk bandwidth varies from 10 MB/sec to 50 MB/sec. An important observation drawn from Fig. 4.7a is that as the bandwidth increases, the satisfied ratios of the two strategies rise gently. The result can be explained by the fact that the high disk bandwidth leads to short transfer times, which in turn result in short processing times of disk requests. Consequently, more disk requests can be finished before respective desired response times.

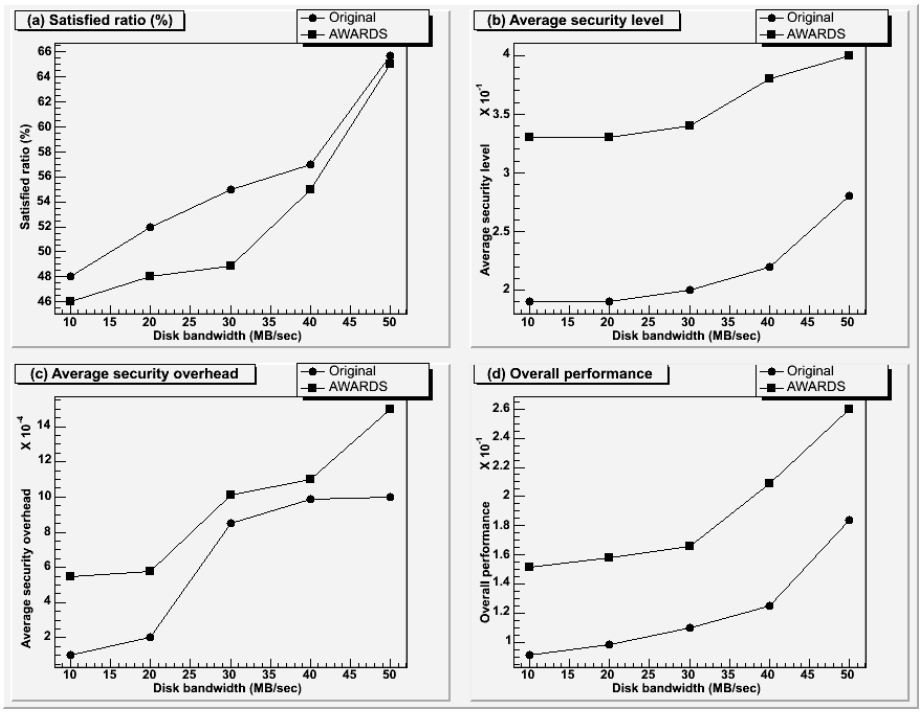


Figure 4.7: Performance impact of disk bandwidth

It is worth noting that the satisfied ratio curves of the two alternatives begin to merge when the bandwidth is larger than 40 MB/sec. Fig. 4.7b shows that the average security level increases as the disk bandwidth is increased,

because processing times of disk requests become smaller in the light of high disk bandwidth. The shortened processing times enable AWARDS to further increase security levels at the expense of higher security overheads (see Fig. 4.7c). Thanks to the increasing satisfied ratio and average security level, the overall performance of AWARDS is substantially boosted in case of a disk system providing high disk bandwidth (see Fig. 4.7d).

4.5.4 Security Level range

In this group of experiments, we studied performance impact of the security requirements of disk requests. The maximal required security level is varied from 0.5 to 0.9, whereas the minimal required security level is fixed at a value of 0.1.

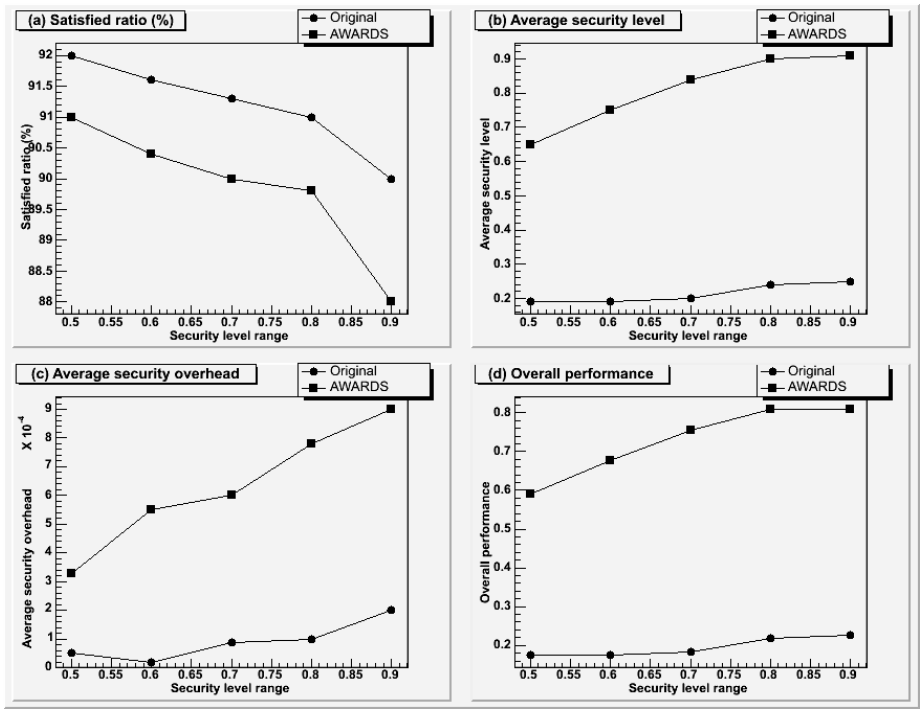


Figure 4.8: Performance impact of security level range

It is observed from Fig. 4.8a that when the maximal required security levels increase, the satisfied ratios of the two strategies decrease. The main reason for this result is that when disk requests require higher security levels, the security overhead is inevitably grows (see Fig. 4.8c). The growing security overhead in turn causes a significant drop in the satisfied ratio. We observe from Fig 4.8b that the amount of improvement in average security level becomes more prominent with the increasing value of maximal required security level. This performance trend can be explained by the fact that the larger the maximal required security level, the more opportunities for AWARDS to dynamically increase security level of each write request. Because the average security level rapidly increases, the overall performance of AWARDS also goes up as the maximal required security levels is increased (see Fig. 4.8d).

4.5.5 Impact of Write Ratio

In the previous experiments, we assume that the write ratio is fixed to 100%. This experiment, however, seeks to measure the impact of write ratio on disk performance. We increased the write ratio of the workload from 0% to 100% with increments of 10%. Fig. 4.9 plots the four performance metrics as functions write ratio.

Like the empirical results presented in Figs. 4.5-4.8, results shown in Fig. 4.9 illustrates the performance improvements of AWARDS over the alternative. Fig. 4.9a shows that the satisfied ratio is marginally reduced with the increasing write ratio, because AWARDS aggressively enhances the security levels of write requests, which experience longer processing times due

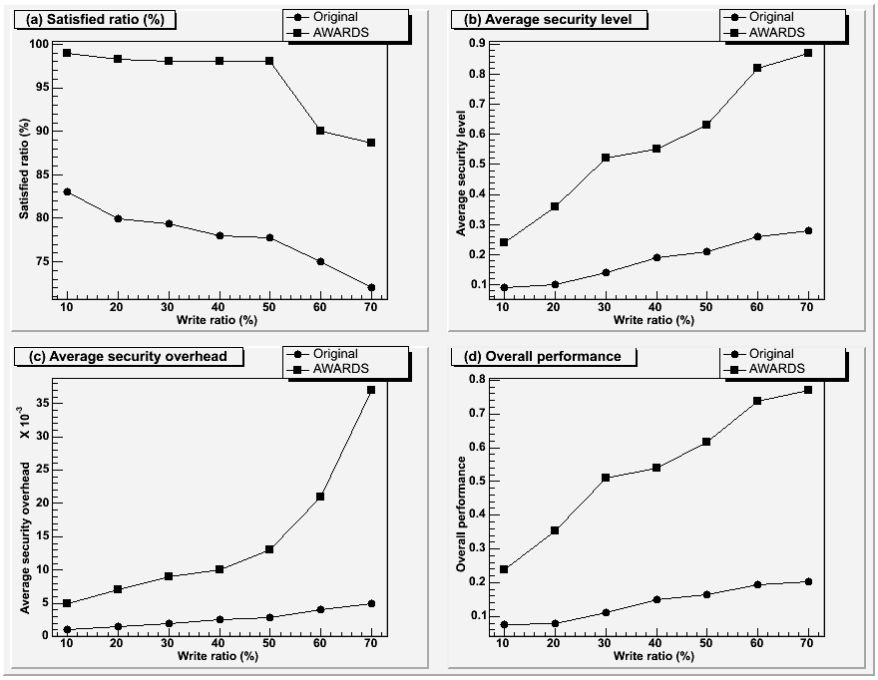


Figure 4.9: Performance impact of write ratio

to higher security overheads (see Fig. 4.9c). It is intriguing to observe from Fig. 4.9b that the performance improvement of AWARDS in security over the Original strategy become more pronounced for higher write ratio. The rationale behind this result is that AWARDS is conducive to the improvement in security for write requests and, thus, increasing number of write requests offers more opportunities for AWARDS to significantly improve security performance by choosing higher security levels for the write requests. Consequently, the overall performance improvement over the competitive strategy is more striking for higher write ratio (see Fig. 4.9d). This result suggests that the proposed AWARDS approach is suitable for improving security of write-intensive applications like transaction processing, logfile updates, and data collection.

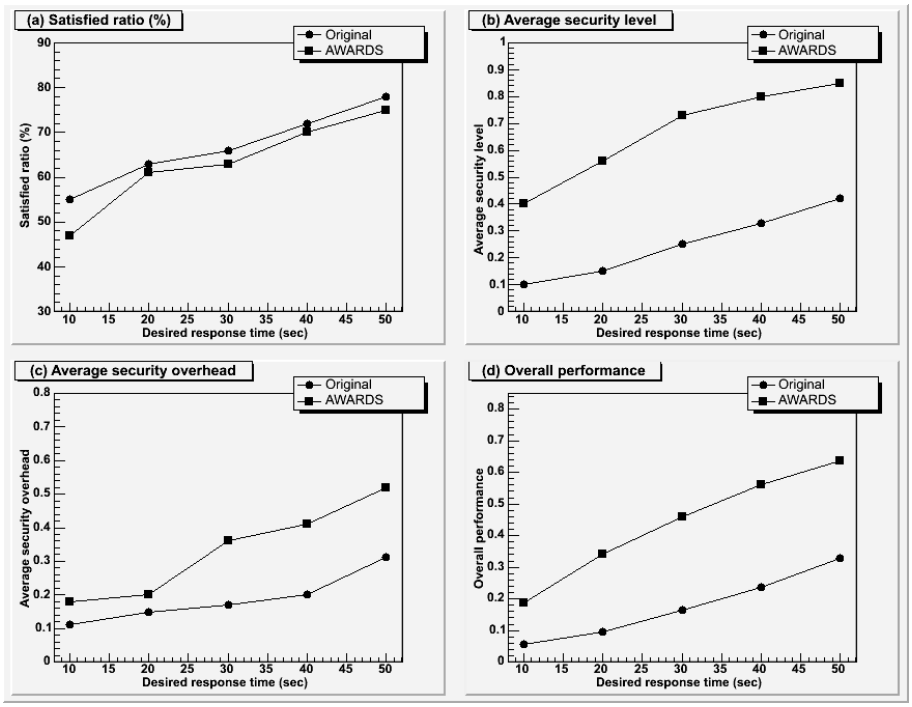


Figure 4.10: Performance impact of desired response time. LU Decomposition

4.6 Real I/O-intensive Applications

To validate the results from the synthetic workload, we used disk traces of real world I/O-intensive applications to evaluate the performance of our strategy in comparison to the Original scheme. We chose two common I/O-intensive applications: LU decomposition [14] and sparse cholesky [1], which have different I/O patterns. The LU decomposition application tries to compute the dense LU decomposition of an out-of-core matrix, whereas the sparse cholesky application is used to calculate Cholesky decomposition for sparse, symmetric positive-definite matrices.

First of all, we studied how desire response times affect satisfied ratios and security levels. Throughout this set of experiments, the disk bandwidth

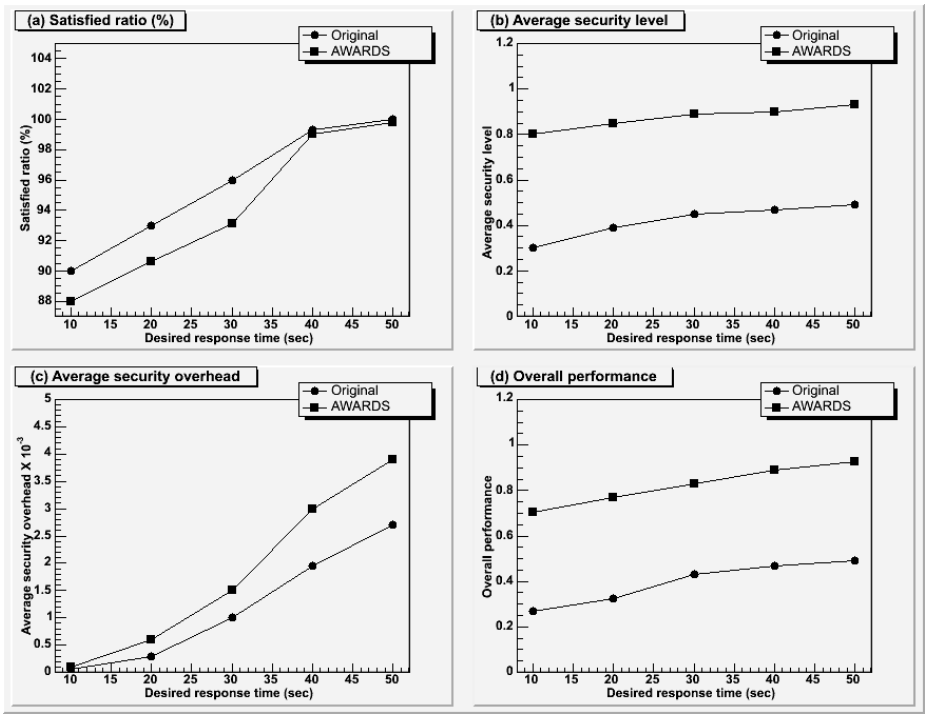


Figure 4.11: Performance impact of desired response time. Sparse Cholesky

was set to 30MB/Sec. The results for the LU decomposition and sparse cholesky applications are plotted in Figs. 4.10 and 4.11, respectively. Figs 4.10a and 4.11a show that for the two I/O-intensive applications, satisfied ratios yielded by AWARDS are close to those of the original strategy. This observation is especially true when the desired response time is long. Figs 4.10b and 4.11b demonstrate that as the desired response time increases, the average security levels for all cases increase. This is mainly because when the desired response time is enlarged, the possibility of improving security levels without violating timing constraints increases. This argument is supported by the results summarized in Figs 4.10c and 4.11c, which reveal that the average security overheads for all cases grow with the increase in desired response times. Figs 4.10d and 4.11d summarize the overall performance of the two schemes. In general, the performance effect of desired response

depends in part on the applications. By comparing the two applications, we observe from Figs 4.10d and 4.11d that LU decomposition is more sensitive to desired response time while sparse cholesky is less sensitive. The cause of this performance difference can be explained as follows. The disk request arrival rate and data size are two dominant factors for satisfied ratio and average security level. High arrival rate and large data size of disk requests give rise to LU decomposition's small satisfied ratios and average security levels (See Figs 4.10a and 4.10b). Consequently, the increase in desired response time provides more opportunities for LU decomposition to improve both satisfied ratio and average security level, which in turn induce a more pronounced improvement in overall performance when the desired response time is enlarged.

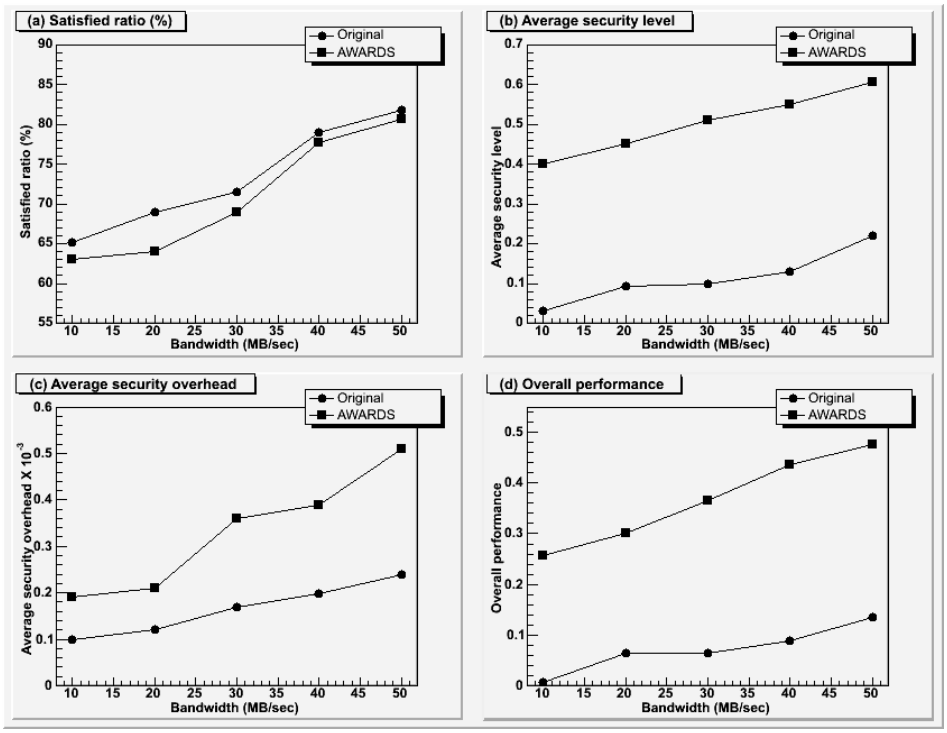


Figure 4.12: Performance impact of impact of disk bandwidth. LU decomposition

Now we evaluate the impact of disk bandwidth on the two real applica-

tions. In this group of experiments, the disk bandwidth varies from 10 to 50 MB/Sec with increment of 10MB/Sec. Figs. 4.12 and 4.13 depict the disk bandwidth effect on the AWARDS and Original strategies when LU decomposition and sparse cholesky are used as the workload. Figs 4.12a-4.12c and 4.13a-c illustrate that for all the cases we have examined, the increased disk bandwidth is a driving force of the improved satisfied ratios and average security levels. These results are consistent with the disk bandwidth impact seen for the synthetic benchmarks in Section 4.6.3.

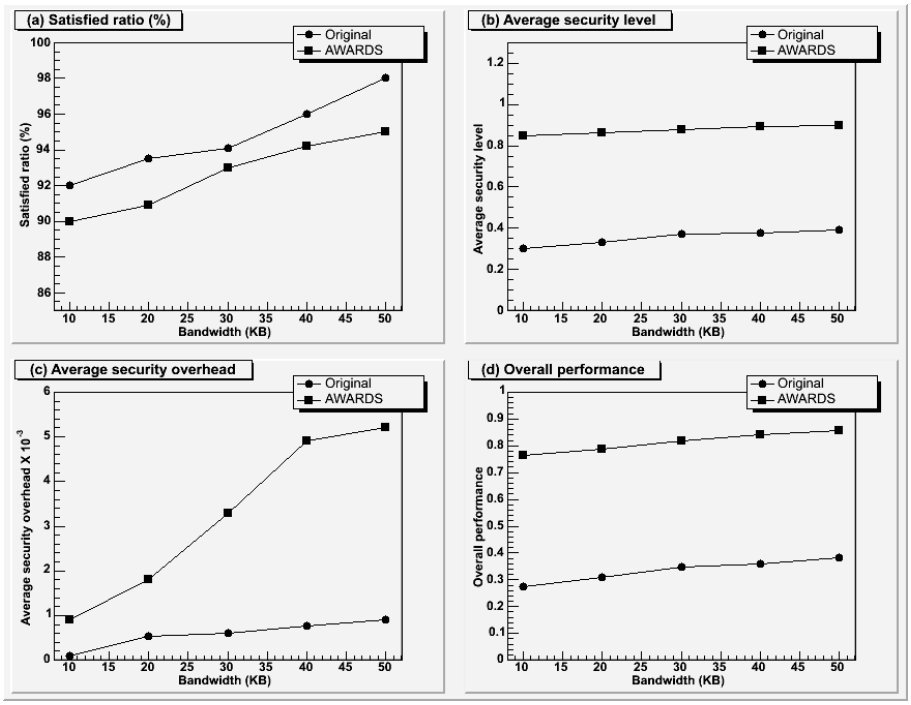


Figure 4.13: Performance impact of disk bandwidth. Sparse Cholesky range

Not surprisingly, the disk bandwidth effect on the two strategies relies in part on the applications. More specifically, Figs 4.12d and 4.13d conclusively show that the overall performance improvement achieved by AWARDS is much more pronounced for the workload with the LU decomposition application as compared to the workload with sparse cholesky. This can be

partially attributed to the high sensitivity of LU decomposition to disk bandwidth. LU decomposition compared with sparse cholesky is more sensitive to disk bandwidth, because the relatively large arrival rate and data size of the workload with LU decomposition induce small satisfied ratios and average security levels, which allow more room for improvement in overall performance.

4.7 Summary

In this chapter, we considered the flexible security requirements of disk write requests in the context of a local disk system. To protect stored data from being tampered or disclosed, we proposed a security-aware storage system architecture. Next, we developed an adaptive write strategy for secure disk systems (AWARDS for short). The AWARDS strategy can adaptively choose an appropriate security service for each write request in a way to maximize the security of the local disk system while making an effort to achieve desired response times of all incoming disk requests. Further, we constructed an analytical model to estimate the probability that a disk request is completed before its desired response time. The model also can be use to derive the expected value of disk requests' security levels. We implemented a prototype of AWARDS and evaluated its performance using synthetic workloads as well as two real world I/O-intensive applications. Experimental results demonstratively show that our strategy outperforms an existing scheme in security and overall performance by up to 325.0% and 358.9% (with averages of 199.5% and 213.4%). In the next chapter, we will evaluate an extended version of AWARDS for parallel disk systems. Both

data placement and load balancing algorithms are being incorporated into the extended version.

Chapter 5

Quality of Security Adaptation in Parallel Disk Systems

In the previous chapter, we proposed an adaptive write strategy for local disk systems, in this chapter we propose a security-aware framework for parallel disk systems (ASPAD).

In the past decade parallel disk systems have been highly scalable and able to alleviate the problem of disk I/O bottleneck, thereby being widely used to build data intensive applications. Although a variety of parallel disk systems were developed, most existing disk systems lack a means to adaptively control quality of security for dynamically changing workloads. In this chapter, we address this gap in disk technology by designing, implementing, and evaluating a quality of security control framework for parallel disk systems, or ASPAD for short, that makes it possible for parallel disk systems to adapt to changing security requirements and workload conditions. The ASPAD framework comprises four major components, namely, a security service middleware, a dynamic data partitioning mechanism, a response time estimator, and an adaptive security quality controller. The framework is conducive to adaptively and expeditiously determining security services for requests submitted to a parallel disk system in a way to improve secu-

rity of the disk system while making an effort to guarantee desired response times of the requests. We conduct extensive experiments to quantitatively evaluate the performance of the proposed ASPAD framework. Empirical results show that ASPAD significantly improves the overall performance of parallel disk systems over the same disk systems without using the ASPAD framework.

The rest of the Chapter is organized as follows. Section 5.2 details a model of disk requests and the new architecture of storage systems. In section 5.3, we propose the adaptive write strategy for security-aware storage systems. We build an analytical model in Section 5.4. Section 5.5 presents extensive experimental results on a wide range of workload conditions. Finally, Section 5.6 concludes the chapter with future directions.

5.1 Motivation

In the past ten years, disk systems have been widely used to develop data intensive applications, including but not limited to video surveillance [1] , remote-sensing database systems [31], and long running simulations [32]. The high performance of data-intensive applications heavily relies on the performance of underlying disk systems due to the rapidly widening gap between CPU and disk I/O speeds [44]. Parallel disk systems play an important role in the development of high-performance data-intensive applications, because the high scalability and parallelism of parallel disk systems can alleviate the problem of disk I/O bottleneck. To exploit I/O parallelism in parallel disk systems, the common wisdom is to partition and distribute data among an array of disks. Disk I/O parallelisms can be provided

in forms of inter-request and intra-request parallelism. Inter-request parallelism allows multiple independent requests to be served simultaneously by a parallel disk system, whereas intra-request parallelism enables a single disk request to be processed by multiple disks in parallel. A parallelism degree of a data request is the number of disks where the requested data resides Error! Reference source not found..

Many data-intensive applications embrace a rich variety of security services to protect data residing in disk systems from talented intruders [35]. Further, it is often desirable for next generation parallel disk systems to be highly flexible in order to support varying quality of security at different times during a data intensive application lifetime. This trend is especially true for data-intensive applications where disk requests need to be completed within specified response times [64][65]. As such, parallel disk systems aim to achieve two major performance goals, including high quality of security and response times guarantee ratio. Since any high-performance parallel disk is required to adapt to changing security requirements and workload conditions, the necessity of automatic tuning security services and parallelism degrees is increasingly becoming a critical and challenging issue in the design and development of modern parallel disk systems.

In this chapter, we propose an Adaptive quality of Security control framework for Parallel Disk systems, or ASPAD for short. The ASPAD framework comprises three major components, namely, a dynamic data partitioning mechanism, a response time estimator, and an adaptive security quality controller. Integrating the ASPAD framework with parallel disk systems makes it possible for disk systems to adapt to changing security requirements and

workload conditions, thereby providing a rich set of data storage environments for data-intensive applications. To achieve this design goal, ASPAD endeavors to choose the most appropriate security services for disk requests in such a way to improve security of parallel disk systems while making the best effort to warrant completions of a vast majority of the requests before desired response times.

5.2 Architecture and Disk Request

5.2.1 System Architecture

In this chapter we consider a security-aware parallel disk system, which encompasses a parallel disk system, networks, and an ASPAD framework. The architecture of the security-aware parallel disk system is delineated in Figure 5.1 It should be noted that the proposed architecture is general enough to accommodate a wide range of storage systems including, of course, both network attached storage devices (NAS) and storage area networks (SAN).

The ASPAD framework, which is at the heart of the proposed system architecture, comprises a security service middleware, a dynamic data partitioning mechanism, a response time estimator, and an adaptive security quality controller. The security service middleware features an array of security services with various quality of security to support read and write requests issued by clients with flexible security needs. In the security service middleware, common security services include encryption services, authentication services, auditing services, and access controllers. The security service mid-

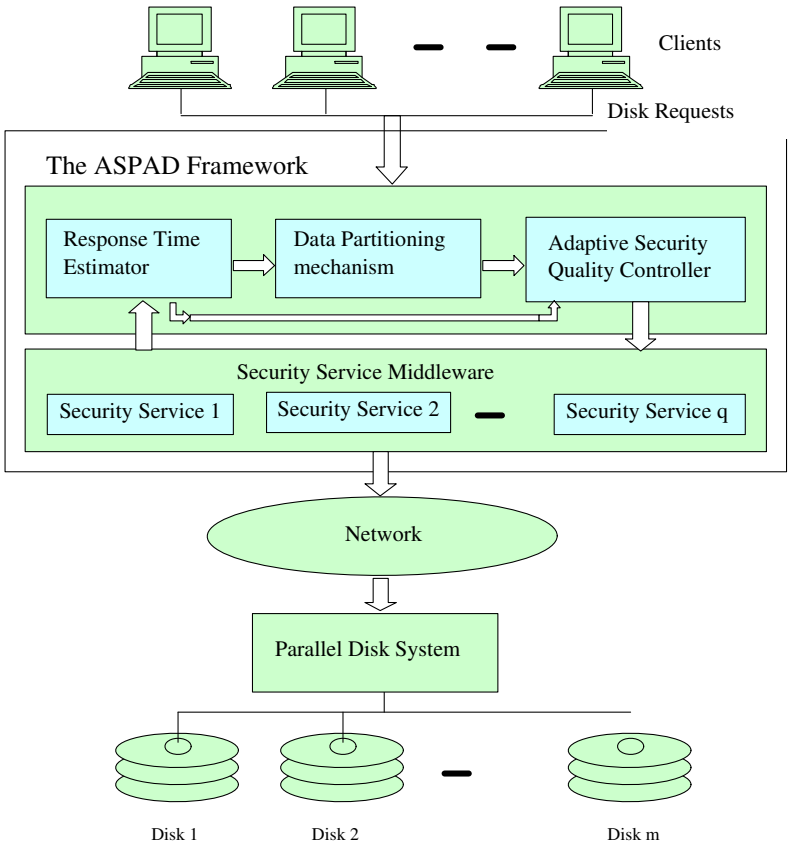


Figure 5.1: Architecture of a security-aware parallel disk system

dleware is highly flexible in the sense that the middleware allows new security services to be readily incorporated and old security services to be easily removed. Quality of security exhibited by a security service depends on the robustness of the corresponding security mechanism used to implement the service, on the rigorous way in which the security mechanism is tested, and on the period of time in which the security service has been used. The data partitioning mechanism is geared to divide a large amount of data into fixed-size of data units stored on a number of disks. We consider file striping in this study, because it is a generic method for a wide variety of data type [28]. The process of data partitioning is detailed in Section (5.3.1) To determine an optimal parallelism degree (also known as stripe unit size) for each disk

request, the data partitioning mechanism has to leverage the response time estimator to predict the response time of the request. Furthermore, the response time estimator is indispensable for the adaptive security quality controller, because estimating the maximum response time of a disk request is the first step towards choosing the most appropriate security level for the disk request. Section (5.3.2) provides a means of estimating response times of disk requests submitted to a parallel disk system. The security quality controller is developed to make a decision of selecting a security service to protect data for each request and, therefore, the security quality controller has to be adaptive to both variation in runtime security demands and dynamically changing workloads. Thus, the controller aims to achieve the best tradeoff between system security and performance.

On top of the ASPAD framework clients issue read and/or write requests to the parallel disk system through network links. With respect to data partitioning and security quality control, read and write requests are treated in distinct ways. Data partitioning and security service selections for read requests are carried out statically, because static portioning gives rise to balanced load across all the disks in the system. In contrast, dynamic data partitioning is performed for write requests to alleviate imbalanced load among the disks due to skewed access frequencies. The security quality controller selects the most appropriate security service to protect data for each disk request. In this paper, we restrict our attention to adaptive quality of security control for write requests, because security for read requests can be optimized offline. Throughout this paper, disk requests issued by clients are write unless otherwise specified. After the process of data partitioning

and security service controlling is accomplished for a write, the stripe units of data in encrypted form are transferred through the network links. Finally, the stripe units are written to multiple disks in parallel.

5.2.2 Quality of Security

Recall that the security-aware parallel disk system encompasses a security service middleware providing an array of security services with various quality of security. The quality of security for each security service is measured by security level ranging from 0.1 to 1.0 [64][65][35]. Alternatively, the quality of security for a security service can be qualitatively measured using, for example, the following seven levels: extremely high, very high, high, medium, low, very low, and no security protection. A translation mechanism can be easily implemented to convert any of the seven levels into a value in a range between 0.1 and 1.0. A higher security level of a security service indicates a better quality of security of the service. More generally speaking, security services fall into three categories: confidentiality, integrity, and availability. Without loss of generality, in our quality of security model we address the confidentiality issues by employing nine cryptographic algorithms in our framework [64][65]. Note that the quality of security model can be easily extended to incorporate the other two security service categories.

Our quality of security model is also readily apply to any disk system where there are only a few cryptographic algorithms implemented, because each cryptographic mechanism can provide various quality of security with vastly different key lengths. In other words, a cryptographic mechanism is enabled to facilitate the ASPAD framework with a variety of cryptographic services

using the same mechanism with different key lengths. We assume that the clients, network, and parallel disk system are always available by the virtue of fault-tolerant mechanisms residing in these components. This assumption is valid, because the overhead of supporting reliability in the system can be envisioned as a part of security overhead.

Security overheads incurred by the cryptographic algorithms depend on size of data to be encrypted and performance of the algorithms, each of which is assigned a security level. Let σ_i denote the security level of a cryptographic algorithm used to encrypt the data for disk request r_i, and d_i is the size of data to be encrypted. We can obtain the security overhead of request r_i using Eq. 1, where $P(\sigma_i)$ is a function mapping security level i to the performance (measured by KB/ms) of the corresponding confidentiality service [7].

$$T_{security}(\sigma_i, d_i) = \frac{d_i}{P(\sigma_i)} \tag{5.1}$$

5.2.3 Disk requests with security and performance requirements

We consider in this study data-intensive applications with both security and performance constraints, meaning that disk requests issued by the applications to a parallel disk system impose both security and performance requirements. It should be noted that disk requests' security and timing requirements, defined as level of security services and desired response times, are derived from corresponding data-intensive applications. A security level specified by applications may be extremely high, very high, high, medium, low, very low, or no security protection. Again, the translation mechanism can be used by the ASPAD framework to convert any of the seven levels

into a value in a range between 0.1 and 1.0. While the security requirement of a request, e.g., r_i, is specified by a lower bound s_i on security level that the disk system has to provide. Hence, the security service controller must ensure that $\sigma_i$is greater than or equal to s_i. In this chapter, we investigate desired response time, which is a specific performance requirement; and the desired response time of request r_i is represented by t_i. We denote parallelism degree of r_i by p_i. It is worth noting that parallelism degrees play a critical role in performance tuning of parallel disk systems. As such, it is appealing to devise the data partitioning mechanism to automatically determine a parallelism degree for each request in a way to improve throughput of the system. Given parallelism degrees of requests, quality of security for the requests can be tuned in a judicious manner by the security service controller.

The first step toward improving quality of security is to quantitatively measure security benefits of a disk request. To achieve this goal, we calculate the security benefit of request r_i using the following security level function.

$$S(r_i) = \sum_{j=1}^{p_i} \sigma_{ij}, \sigma_{ij} \geq s_i \; and \; p_i \leq m \tag{5.2}$$

where p_i is the parallelism degree of the request,m is the number of disks in the parallel disk system and σ_{ij} is the security level of a confidentiality service chosen for the jth stripe unit of r_i. It is intuitive to argue that (1) the parallelism degree p_i can not exceed the total number m of disks in the system, and (2) the security level σ_{ij} must be higher than or equal to the level of the minimal security requirement s_i.

Given a sequence of requests $R = \{r_1, r_2,..., r_n\}$ we can obtain the security benefits experienced by the requests. Thus, we have

$$S(R) = \sum_{i=1}^{n} S(r_i) \tag{5.3}$$

Now we obtain the following non-linear optimization problem formulation to compute the optimal security benefit of the networked parallel disk system

$$maximize \;\; S(R) = \sum_{i=1}^{n} \sum_{j=1}^{p_i} \sigma_{ij}$$

subject to

$$(a) \;\; \forall 1 \leq i \leq n : max_{1 \leq j \leq p_i}\{\theta_{ij}\} \leq t_i,$$

$$(b) \;\; \sigma_{ij} \geq s_i \; and \; p_i \leq m \tag{5.4}$$

where θ_{ij} is the response time for the jth striping unit of request r_i. In an effort to enhance security of the system, we have to guarantee that the following three conditions are met. First, the response time of all striping unit in request r_i must be smaller than the desired response time. Second, the low bound on security level can not be violated. Third, the parallelism degree of r_i has to be smaller than or equal to the number of disks in the system.

5.3 The ASPAD Framework

In this section, we delineate the three main components in the ASPAD framework. Specifically, we first outline the data partitioning mechanism. Next, we describe a way of estimating the response time of a request issued by a client. Finally, and importantly, we investigate an adaptive quality of security controller in the context of a parallel disk system.

5.3.1 Data Partitioning

One of the major components in the proposed framework is the method of data partitioning that determines optimal parallelism degrees for disk requests. Dynamic data partitioning is of importance for the ASPAD framework, because the data partitioning method helps in minimizing the response times of requests, thereby creating more space to enhance security. As such, the first phase in ASPAD is to dynamically calculate the optimal parallelism degree of a request (see Fig. 5.2, Step 3).

Again, we denote the parallelism degree and data size of a request r_i by p_i and d_i, respectively. Before proceed to the analysis of optimal parallelism degrees, we formally derive the expected value of disk service time $T_{disk}(d_i, p_i)$ for request r_i. Thus, the expected disk service time can be computed as

$$E(T_{disk}(d_i, p_i)) = E(T_{seek}(p_i)) + E(T_{rot}(p_i)) + E(T_{trans}(d_i, p_i)) \quad (5.5)$$

where $E(T_{seek}(p_i))$,$E(T_{rot}(p_i))$, and $E(T_{trans}(d_i, p_i))$ are expected values of seek time, rotation time, and transfer time, respectively. It was shown in

[11] that the expected seek time can be approximated as below, where C is the number of cylinders on a disk, a and b are two disk-type-independent constants, whereas e and f are disk-type-dependent constants.

$$E(T_{seek}(p_i) = eC(1 - a - bln(p_i)) + f \tag{5.6}$$

The expected value of rotation time can be expressed as Eq. (5.7), where T_{ROT} is the rotation time of a disk. Note that a similar expression can be found in [11].

$$E(T_{rot}(p_i)) = \frac{p_i}{p_i + 1}.T_{ROT} \tag{5.7}$$

The expected transfer time can be approximated by Eq. (5.8), where B_{disk} is the disk bandwidth.

$$E(T_{trans}(d_i, p_i)) = \frac{d_i}{p_i}.\frac{1}{B_{disk}} \tag{5.8}$$

Substituting Eqs. (5.6)-(5.8) into Eq. (5.5), we obtain the expected value of disk service time as

$$\begin{aligned} E(T_{disk}(d_i, p_i)) = {} & eC(1 - a - bln(p_i)) + f \\ & + \frac{p_i}{p_{i+1}}.T_{ROT} + \frac{d_i}{p_i}.\frac{1}{B_{disk}} \end{aligned} \tag{5.9}$$

Now we are positioned to calculate the optimal parallelism degree of request r_i by determining the minimum of the function $E(T_{disk}(d_i, p_i))$. Thus, we

can obtain the optimal value of p_i by solving Eq. (5.10).

$$\frac{dE(T_{disk}(d_i,p_i))}{dE(p_i)} = \frac{T_{ROT}}{p_i+1} - \frac{p_i.T_{ROT}}{(p_i+1)^2} - \frac{eCb}{p_i} - \frac{d_i}{p_i^2}.\frac{1}{B_{disk}} = 0 \qquad (5.10)$$

The parallelism degree determined by Eq. (5.10) can not exceed m, which is the number of disks in the system. Therefore, the optimal parallelism degree is given by $min(p_i, m)$.

5.3.2 Estimating response times

To adaptively adjust security levels for disk requests, we need to estimate each request's maximum response time, which is defined as the interval between the time a request is sent by a client and the time the parallel disk system completes corresponding disk I/O operations. Given a newly issued request r, the response time of r is estimated by Eq. (5.11).

$$T(r,p,\sigma) = T_{queue} + T_{partition} + max_{i=1}^{p}\left\{T_{proc}^{i}(r,p,\sigma_i)\right\} \qquad (5.11)$$

where p is the parallelism degree determined by the data partitioning mechanism (see Eq. 5.11), $\sigma = (\sigma_1, \sigma_2, ..., \sigma_p)$ is the request's vector of security levels for p stripe units, T_{queue} is the queueing delay at the client side, $T_{partition}$ is the time spent in data partitioning, and $T_{proc}^{i}$is the system processing delay experienced by the ith stripe unit of the request. With respect to the ith stripe unit of the request, the system processing delay T_{proc}^{i} can be

expressed as

$$T^i_{proc}(r,p,\sigma_i) = T^i_{security}(r,p,\sigma_i) + T^i_{network}(r,p,\sigma_i) + T^i_{disk}(r,p,\sigma_i) \quad (5.12)$$

where $T^i_{security}$,$T^i_{network}$,and T^i_{disk} are the delays at the security mechanism, network subsystem, and parallel disk subsystems, respectively.

The delay at the security mechanism, which is also referred to as security overhead, depends on the assigned security level and data size of the stripe unit. Thus, we can easily derive $T^i_{security}(r,p,\sigma_i)$ from Eq. (5.1) as

$$T^i_{security}(r,p,\sigma_i) = T_{security}(\sigma_i, \frac{d}{p}) = \frac{d}{p.P(\sigma_i)} \quad (5.13)$$

where d is the data size of the request, and d/p is the data size of the ith stripe unit.

We assume that when the ith stripe unit of a request arrives at the network queue, there are k stripe units waiting to be delivered to the parallel disk sub-system. Suppose stripe units are transmitted in a first-in-first-out order, all the stripe units that are already in the queue prior to the arrival of the ith stripe unit must be transmitted earlier than the ith stripe unit. Hence, the delay in the network subsystem $T^i_{network}(r,p,\sigma_i)$ can be written as

$$T^i_{network}(r,p,\sigma_i) = \frac{i.\frac{d}{p} + \sum_{j=1}^{k} d_j}{B_{network}} \quad (5.14)$$

where d_j is the data size of the jth stripe unit in the network queue, and

$B_{network}$ is the effective network bandwidth. It is worth noting that k in Eq. (5.14) the optimal parallelism degree determined by the data partitioning mechanism (see Eq. 5.10 in Section 5.3.1).

Similarly, it is assumed that when the ith stripe unit of the request arrives at disk j, there are k disk requests must be processed by disk j before handling the stripe unit. Thus, the delay in the disk subsystem $T^i_{disk}(r, p, \sigma_i)$ is given by the following formula

$$T^i_{disk}(r, p, \sigma_i) = T_{disk,j}(\frac{d}{p}) + \sum_{l=1}^{k} T_{disk,j}(d_l) \tag{5.15}$$

where $T_{disk,j}(d)$ is the disk processing time of a request containing d bytes of data. We can quantify $T_{disk,j}(d)$ as follows

$$T_{disk,j}(d) = T_{seek} + T_{rot} + \frac{d}{B_{disk}} \tag{5.16}$$

where T_{seek} and T_{rot} are the seek time and rotational latency, and $\frac{d}{B_{disk}}$ is the data transfer time depending on the data size d and disk bandwidth B_{disk}.

5.3.3 The processing algorithm for the ASPAD framework

Now we are positioned to develop an adaptive quality of security control algorithm for the proposed ASPAD framework in parallel disk systems. The processing algorithm separately repeats the process of controlling the quality of security for each disk request on the fly. Thus, the algorithm is geared to adaptively choose the most appropriate security services for stripe units of a disk request while warranting the desired response time of the request.

```
Input: r: a newly arrived disk request
       t_i: desired response time of the ith request
       s_i: the ith request's lower bound on security level
       Q, a waiting queue at the client side
1. Insert r into Q based on the earliest desired response time first policy
2. for each request r_j in the waiting queue Q do
     /* Phase 1: dynamic data partitioning */
3.   Calculate the optimal parallelism degree p_i of r_i
4.   Partition the request into p_i stripe units
5.   for each stripe unit of r_i do
6.     Initialize the security level σ_ij of the jth stripe unit to the minimal value s_i
       /* Phase 2: response time estimation */
7.     Apply Eqs. (11) and (12) to estimate the response time of the jth stripe unit;
       /* Phase 3: adaptive security quality control */
8.     while (estimated response time < desired response time t_i) do
9.       if σ_ij < 0.9 then /* σ_ij can be further increased */ (see property 1)
10.        Increase security level σ_ij by 0.1
11.        Apply Eqs. (11) and (12) to estimate the response time of the jth stripe unit;
12.      else break /* σ_ij can not be further increased */
13.    end while
14.    if (estimated response time > desired response time t_i) then
15.      Decrease security level σ_ij by 0.1;
16.    Apply the security service with level σ_ij to the jth stripe unit
17.    Deliver the jth unit through the network subsystem to the disk subsystem
18. end for
```

Figure 5.2: The processing algorithm for the ASPAD framework in parallel disk systems

Specifically, the algorithm is carried out in three phases: dynamic data partitioning (see Section 5.3.1), response time estimation (see Section 5.3.2), and adaptive security quality control. To heuristically improve security of the disk systems, ASPAD endeavours to minimize the response time of a request. Hence, the first phase dynamically calculates the optimal parallelism degree of the request, thereby reducing delays at the parallel disk subsystems (see Eq. 5.15). During the second phase of the algorithm, the response time of each stripe unit is estimated (see Eqs. 5.11 and 5.12). Phase three, guided by the estimated response time obtained and desired response time, optimizes the security level of each stripe unit provided that the request's response time given by Eq. (5.11) does not exceed the request's desired response time. The complete algorithm for quality of security control is outlined in Fig. 5.2.

When a client issues a request to the system, ASPAD inserts the newly arrived requests into the waiting queue based on the earliest desired response time first policy (see Step 1). After the data portioning of each request in the queue, ASPAD initializes the security levels of all stripe units of request ri to the minimal levels si (see Step 6). In an effort to gradually increase the security levels of stripe units, ASPAD guarantees that all requests will be completed before their desired response times. Thus, the following property needs to be satisfied in ASPAD.

Property 1. With respect to the ith request, the following two conditions must hold if the jth stripe unit's security level is increased by 0.1:

(1) The current security level σ_{ij} is less than 0.1;

(2) $T_j(r_i, p_i, \sigma_i) \leq t_i$, where T_j is the response time of the jth stripe unit,

t_i is the desired response time of the request, and $T_j(r_i, p_i, \sigma_i) = T_{queue} + T_{partition} + T^{ij}_{proc}(r_i, p_i, \sigma_{ij})$.

Steps 10-11 are repeatedly performed to optimize security levels until a request's desired response time can not be guaranteed (see Step 12) or the security levels are approaching 1.0. Consequently, the ASPAD algorithm dramatically increases the security levels while making the best effort to finish all the disk requests before their desired response times. The time complexity of ASPAD is evaluated (see Theorem 1) as the number of waiting requests and maximum parallelism degree vary.

Theorem 1. The time complexity of ASPAD is *O(np)*, where n is the number of disk requests in the waiting queue, p is the maximum parallelism degree.

Proof. It takes *O(10)* time to increase the security level of each stripe unit (see Step 8). Since there are *O(p)* number of stripe units in each disk request, the time complexity of optimizing security levels of a write request is *O(10p) = O(p)*. There are n disk requests in the waiting queue and, therefore, the time complexity of improving security of all the requests is: $O(n^2)O(p) = O(n^2p)$.

Compared with processing times at the network subsystems and parallel disk subsystem, the overhead of ASPAD can be negligible. This argument is especially true for workload conditions where arrival rates of disk requests are relatively low, because the number of waiting disk requests in these cases is small.

5.3.4 Optimality of the security quality controller in the ASPAD framework

Before building an analytical model to evaluate performance of ASPAD, we prove the optimality of the ASPAD framework. Theorems 2-4 below are of help in the proof of Theorem 5, which signify that the ASPAD framework optimizes quality of security for disk requests submitted to a parallel disk system

Theorem 2. Given the *jth* and *kth* stripe units in a disk request r_i (the number of units in r_i is $p_i \geq 2$, and $1 \leq j,\ k \leq p_i$) and a security-aware parallel disk system, optimizing security level of the *jth* stripe unit is independent of the optimization of security level σ_{ij} of the *kth* stripe unit.

Proof. The algorithm outlined in Fig. 5.2 shows that optimizing the security level σ_{ij} of the *jth* stripe unit in disk request r_i depends on the estimated response time of the request (see Step 8 in Fig. 5.2). The estimated response time of r_i in turn relies on is the queueing delay T_{queue} at the client side, time $T_{partition}$ spent in the data partitioning mechanism, and maximal system processing delay T_{proc}experienced by all the stripe units of the request (see Eq. 5.11). Since the two stripe units have the same values of T_{queue} and $T_{partition}$, we are concerned with the system processing delays with respect to the *ith* and *jth* stripe unit. Eq. 5.12 shows that the system processing delay of the *jth* stripe unit is a summation of the delays at the security middleware, network subsystem, and parallel disk subsystems, respectively. The two stripe units are independent of each other with respect to their system processing delays, because the *ith* stripe units' delay times in security middleware, network subsystem, and parallel disk subsystems are not affected by the other stripe unit. This means that the security levels of the two

stripe units can be optimized independently. Hence the proof of Theorem 2 is complete.

The following theorem shows that a disk request's quality of security is optimized by maximizing the security level of each stripe unit of the disk request.

Theorem 3. If the security levels of all the stripe units in a disk request r_i are maximized in a security-aware parallel disk system, then the adaptive security quality controller in the ASPAD framework optimizes the quality of security for disk request r_i. More formally, we have

$$\forall 1 \leq j \leq p_i : \sigma_{ij} is maximized. \rightarrow \sum_{j=1}^{p_i} \sigma_{ij}$$

Proof. The proof of this theorem is immediate from Theorem 2. Since Theorem 2 proves that the optimization of security level of the *jth* stripe unit is independent of the optimization of security levels of the other stripe units of the disk request r_i, the value of can be maximized without adverse effects upon the security levels of the other stripe of r_i. Therefore, the security levels of all the stripe units within disk request r_i can be maximized simultaneously by the adaptive security quality controller in the ASPAD framework. Consequently, the summation of the security levels of all the stripe unit of the disk request r_i is maximized by the security quality controller, i.e., $\sum_{j=1}^{p} \sigma_{ij}$ is maximized. This completes the proof of the theorem.

In the following, we show that the security quality controller in ASPAD maximizes the security level of each stripe unit in a disk request. To facilitate

the proof of Theorem 4, we introduce an important concept of non-secure response time.

Definition 1. The non-secure response time $T^{ij}_{non-secure}$ of the jth stripe unit in disk request r_i is defined as the response time of the stripe unit that is not secured by any security mechanism. Thus, we have $T^{ij}_{non-secure} = T_{queue} + T_{partition} + T^{ij}_{security} + T^{ij}_{network} + T^{ij}_{disk}$.

Theorem 4. Given any stripe unit of a disk request r_i (e.g., the jth stripe unit) and a security-aware parallel disk system, the adaptive security quality controller in the ASPAD framework maximizes the security level σ_{ij} of the stripe unit.

Proof. Without loss of generality, let us consider the jth stripe unit in disk request r_i. We have to prove the theorem by demonstrating that (1) the security level σ_{ij} of the jth stripe unit cannot be further increased, and (2) once σ_{ij} is maximized, σ_{ij} is not affected by the security levels of the other stripe units in disk request r_i. We can easily prove the correctness of the second fact, because Theorem 2 shows that the optimizations of all the stripe units in a disk request are independent of each other. Now let us prove the first fact, i.e., cannot be further increased by the processing algorithm (see Fig. 5.1). The first phase in the algorithm attempt to minimize the non-secure response time (see Definition 1) of all stripe units in disk request r_i using the dynamic data partitioning mechanism (see Steps 3 and 4 in Fig. 5.2). Consequently, the non-secure response time any stripe unit is minimized. Thus, the non-secure response time $T_{queue} + T_{partition} + T^{ij}_{network} + T^{ij}_{disk}$ of the jth stripe unit is minimized. Since $T_{queue} + T_{partition}$ can not be reduced by the ASPAD framework, the processing algorithm minimizes the

value of $T^{ij}_{network}+T^{ij}_{disk}$. Step 8 ensures that the inequality $T_{queue}+T_{partition}+T^{ij}_{network}+T^{ij}_{disk} \leq t_i$ holds and, therefore, we have $T^{ij}_{security} \leq t_i - (T_{queue}+T_{partition}+T^{ij}_{network}+T^{ij}_{disk})$. We proved that $T_{queue}+T_{partition}+T^{ij}_{network}+T^{ij}_{disk}$ on the right-hand side of the inequality is minimized by the first phase of the algorithm, indicating that time spent in security services is maximized. As a result, Steps 9 and 10 in the algorithm leverage the maximized time $T^{ij}_{security}$ to optimize the security level σ_{ij}. Hence the proof of Theorem 4 is complete.

Now we are in a position to prove the optimality of security quality controller in the ASPAD framework.spent in security services is maximized.

Theorem 5.Given a disk request r_i and a security-aware parallel disk system, the adaptive security quality controller in the ASPAD framework optimizes the quality of security for disk request r_i.

Proof. The proof of the theorem is immediate from Theorems 3 and 4. First, Theorem 4 proves that each stripe unit in disk request r_i has the corresponding security level maximized. Next, Theorem 3 shows if the security levels of all the stripe units in a disk request r_i are maximized in a security-aware parallel disk system, then the adaptive security quality controller in the ASPAD framework optimizes the quality of security for disk request r_i. Hence, the proof of Theorem 5 is complete.

5.4 Analytical Model

In this section, we build an analytical model that helps to evaluate the benefit of using the ASPAD framework in parallel disk systems. Specifically, we first derive the probability that a disk request is completed before its

desired response time in a security-aware parallel disk system. Next, we calculate the expected value of security levels assigned to disk requests by the security quality controller in the ASPAD framework.

5.4.1 Satisfied Ration

Given a parallel disk system with the ASPAD framework, we consider the probability that a disk request r_i can be finished within the desired response time t_i, i.e., $Pr(T(r_i, p_i, \sigma_i) \leq t_i)$ where $T(r_i, p_i, \sigma_i)$ is the response time of r_i. When disk request r_i arrives, the request is inserted in the queue in an increasing order of all requests' desired response time. In doing so, disk requests with the earliest desired response times are given the highest priority and responded by the parallel disk system first. To simply the analysis that follows, we assume that processing times of different disk requests are statistically independent. Suppose in the queue there are n pending disk requests indexed by their priorities, and the desired response time of r_i is smaller than that of r_j if $i \leq j$, i.e., $\forall 1 \leq i, j \leq n : i \leq j \Rightarrow t_i < j$.

Eq. (5.11) signifies that $T(r_i, p_i, \sigma_i)$ is disk request r_i's estimated response time computed as the summation of the queueing delay, the time spent in data partitioning, and the system processing delay. Let p_x be the probability that the disk request r_i requires $x$time unit to complete, i.e, $u_x = Pr(T(r_i, p_i, \sigma_i) = x)$. Let q_y be the probability that the total required processing time of disk requests with higher priorities is y, i.e.,
$v_y = Pr(\sum_{j=1}^{i-1} T(r_j, p_j, \sigma_j) = y)$. The probability that r_i is unable to be finished within the desired response time ti is computed as follows:

$$Pr(T(r_i,p_i,\sigma_i) > t_i) = Pr\Big[T(r_i,p_i,\sigma_i) = 1 \Big| \sum_{j=1}^{i-1} T(r_j,p_i,\sigma_j) \geq t_i\Big] +$$
$$\vdots$$
$$+Pr\Big[T(r_i,p_i,\sigma_i) = t_i + 1 \Big| \sum_{j=1}^{i-1} T(r_j,p_i,\sigma_j) \geq 0\Big]$$
$$= u1 \sum_{y=t_i}^{\infty} v_y + u2 \sum_{y=t_i-1}^{\infty} v_y + \cdots + v_y \sum_{y=1}^{\infty} v_y + u_{t_i+1} \sum_{y=0}^{\infty} v_y$$
$$= \sum_{x=1}^{t_i+1} \Big[u_x \sum_{y=t_i+1-x}^{\infty} v_y\Big] \qquad (5.17)$$

where the second line on the above equation indicates the conditional probability that the required processing time of disk request r_i is k given that it requires at least $t_i + 1 - k$ time unit to complete disk requests with higher priorities.

The probability that a disk request r_i is completed within its desired response time is given by

$$Pr(T(r_i,p_i,\sigma_i \leq t_i) = 1 - Pr(T(r_i,p_i,\sigma_i) > t_i)$$
$$= 1 - \sum_{x=1}^{t_i+1} \Big(u_x \sum_{y=t_i+1-x}^{\infty} v_y\Big) \qquad (5.18)$$

In what follows, we drive two important probabilities u_x (see Eq. 5.26) and v_y (see Eq. 5.30) used in Eqs. (5.17) and (5.18) to approximate the satisfied ratio. First, let us derive the probability $u_x = Pr(T(r_i,p_i,\sigma_i) = x)$ that the disk request r_i requires x time unit to complete. Eq. (5.11) in Section shows a way of computing and, thus, we have

$$T(r_i, p_i, \sigma_i) = T^i_{queue} + T^i_{partition} + max^p_{k=1} T^{ik}_{proc}(r_i, p_i, \sigma_{ik}) = x \qquad (5.19)$$

where T^i_{queue} is the queueing delay of disk request r_i, $T^i_{partition}$ is the data partitioning overhead of r_i, and $T^{ik}_{proc}$is the system processing delay experienced by the ith stripe unit of r_i.

The queueing delay t^i_{queue} of disk request r_i is the summation of the data partitioning times and security overhead experienced by other disk requests with higher priority (i.e., with earlier desired response time). Thus, T^i_{queue} can be expressed as

$$\begin{aligned} T^i_{queue} &= \sum_{j=1}^{i=1} (T^j_{partition} + T^j_{security}) \\ &= (i-1)T_{partition} + \sum_{j=1}^{i=1} T^j_{security} \end{aligned} \qquad (5.20)$$

where the partitioning overhead of all the submitted requests is assumed to be a constant $T_{partition}$. Since the security overhead of disk request r_j is the summation of the security overhead for all the stripe units of r_j, as per Eq. (5.13) we have the following equation to calculate $T^i_{security}$ on the right-hand side of Eq. (5.20).

$$T^j_{security} = \sum_{k=1}^{p_i} \frac{d}{p_j P(\sigma_{jk})} \qquad (5.21)$$

Therefore, we can obtain the following equation from Eqs. (5.19), (5.20), and (5.21).

$$\max_{k=1}^{p_i} T_{proc}^{ik}(r_i, p_i, \sigma_i)x - iT_{partition} - \sum_{j=1}^{i-1}\sum_{k=1}^{p_i}\frac{d}{p_j.P(\sigma_{jk})} \qquad (5.22)$$

Since the probability

$$Pr(max_{k=1}^{p_i} T_{proc}^{ik}(r_i, p_i, \sigma_{ik}) = x - i.T_{partition} - \sum_{j=1}^{i-1}\sum_{k=1}^{p_i}\frac{d}{p_j.P(\sigma_{jk})})$$

equals to the probability $u_x = Pr(T(r_i, p_i, \sigma_i) = x)$,$u_x$ used in Eqs. (5.17) and (5.18) can be written as

$$\begin{aligned} u_x &= Pr(max_{k=1}^{p_i} T_{proc}^{ik}(r_i, p_i, \sigma_{ik}) \\ &= x - i.T_{partition} - \sum_{j=1}^{i-1}\sum_{k=1}^{p_i}\frac{d}{p_j.P(\sigma_{jk})}) \end{aligned} \qquad (5.23)$$

Let $Pr(max_{k=1}^{p_i} T_{proc}^{ik}(r_i, p_i, \sigma_{ik}) = a)$ be the probability that $\max_{k=1}^{p_i} T_{proc}^{ik}(r_i, p_i, \sigma_{ik})$ equals to $a.Pr(max_{k=1}^{p_i} T_{proc}^{ik}(r_i, p_i, \sigma_{ik}) = a)$

can be rewritten as

$$\begin{aligned} Pr(max_{k=1}^{p_i} T_{proc}^{ik}(r_i, p_i, \sigma_{ik}) = a) = \\ Pr(T_{proc}^{i1}(r_i, p_i, \sigma_i) = a).\prod_{k=2}^{p_i} Pr\Big(T_{proc}^{i1}(r_i, p_i, \sigma_{i1}) < a) \\ + Pr(T_{proc}^{i2}(r_i, p_i, \sigma_i) = a).\prod_{k=1,k=2}^{p_i} Pr(T_{proc}^{i2}(r_i, p_i, \sigma_{i2}) < a) \end{aligned}$$

$$\vdots$$

$$+Pr(T^{ip_i}_{proc}(r_i, p_i, \sigma_i) = a).\prod_{k=1}^{p_i} Pr(T^{ip_i}_{proc}(r_i, p_i, \sigma_{ip_i}) < a) \qquad (5.24)$$

We denote $Pr(T^{ik}_{proc}(r_i, p_i, \sigma_{ik}) = a)$ by $c_{ik}(a)$, and $Pr(T^{ik}_{proc}(r_i, p_i, \sigma_{ik}) < a)$ by $\acute{c}_{ik}(a)$ Then, Eq. (5.24) can be expressed as below

$$Pr(\max_{k=1}^{p_i} T^{ik}_{proc}(r_i, p_i.\sigma_{ik}) = a) = \sum_{k=1}^{p_i}(c_{ik}(a).\prod_{i=1,j=k}^{p_i} \acute{c}_{ik}(a)) \qquad (5.25)$$

Consequently, the probability u_x can be derived from Eqs. (5.23) and (5.25) by substituting

$$x - i.T_{partition} - \sum_{j=1}^{i-1}\sum_{k=1}^{p_i} \frac{d}{p_j.P(\sigma_{jk})} \;\; for \;\; variable \;\; a \;\; in \;\; Eq.5.25$$

$$u_x = \sum_{k=1}^{p_i} (c_{ik}[x - i.T_{partition} - \sum_{j=1}^{i-1}\sum_{k=1}^{p_i} \frac{d}{p_j P(\sigma_{jk})}].\prod_{i=1,i\neq k} \acute{c}_{ik}[x - i.T_{partition} - \sum_{j=1}^{i-1}\sum_{k=1}^{p_i} \frac{d}{p_j.P(\sigma_{jk})}]) \qquad (5.26)$$

Now we derive the second important probability $v_y = Pr(\sum_{j=1}^{i-1} T(r_j, p_j, \sigma_j) = y)$

From $u_x$computed by Eq. (5.26). To facilitate the derivation of v_y, we need the following definition.

Definition 2. $\phi_y^{i-1}$is a set of vectors $(y1, y2, ..., y_{i-1})$ in which the summation of all the elements in each vector equals to y. Formally, the set of vectors is

defined as:

$$\phi_y^{i-1} = (y1, y2, ..., y_{i-1}) where \sum_{j=1}^{i-1} y_j = y$$

The probability v_y is derived from u_x and set ϕ_y^{i-1} as below

$$v_y = \sum_{(y1,...,y_{i-1}) \in \phi_y^{i-1}} Pr(T(r_1, p_1, \sigma_1) = y1).Pr(T(r_{r2}, p_2, \sigma_2) = y_2)... \\ Pr(T(ri-1, p_{i-1}, \sigma_{i-1}) = y_{i-1}) \quad (5.27)$$

5.4.2 Quality of security

To quantify security quality offered by a parallel disk system, we derive in this section the expected security level experienced by disk requests. First, we compute the probability $Pr(\sigma_{ij} = z)$ that the security level of the jth stripe unit of a disk request r_i equals to z. Recall that the security level of a stripe unit depends on the desired response time of its disk request, data size of the request, and processing times of other waiting requests with higher priorities. Therefore,$Pr(\sigma_{ij} = z)$ can be expressed as

$$Pr(\sigma_{ij} = z) = \\ Pr(d_i = d_{min}). \sum_{k=t_{min}}^{t_{max}} Pr(t_i = k).Pr(\sum_{l=1}^{i-1} T(r_l, p_l, \sigma_l) = k - \bar{T}(\frac{d_{min}}{p_i}, z))$$

$$\vdots$$

$$+Pr(d_i = a).\sum_{k=t_{min}}^{t_{max}} Pr(t_i = k).Pr(\sum_{l=1}^{i-1} T(r_l, p_l, \sigma_l) = k - \bar{T}(\frac{a}{p_i}, z))$$
$$+Pr(d_i = d_{max}).\sum_{k=t_{min}}^{t_{max}} Pr(t_i = k).Pr(\sum_{l=1}^{i-1} T(r_l, p_l, \sigma_l) = k - \bar{T}(\frac{d_{max}}{p_i}, z))$$
$$= \sum_{a=d_{d_{min}}}^{d_{max}} Pr(d_i = a).\sum_{k=t_{min}}^{t_{max}} Pr(t_i = k).Pr(\sum_{l=1}^{i-1} T(r_l, p_l, \sigma_l) = k - \bar{T}(\frac{a}{p_i}, z)) \quad (5.28)$$

where $\bar{T}(\frac{a}{p_i})$ is the processing time of r_i if the data size is a and security level is set to z, data size of any disk request is in the range between d_{min} and d_{max}, and the desired response time of any request is in the range between t_{min} and t_{max}.

The data size and desired response time of each disk request are two random variables distributed according to two probability density functions, which are know a priori. Let $f_a$denote the probability that the data size of the disk request is a, and let w_j denote the probability that the desired response time equals to j. Thus, the probability $Pr(\sigma_{ij} = z)$can be derived from f_a, w_j and v_y (see Eq. 5.27).

$$Pr(\sigma_{ij} = z) = \sum_{a=d_{dmin}}^{d_{max}} f_a.\sum_{k=t_{min}}^{t_{max}} [w_k.v_{k-\bar{T}(\frac{a}{p_i},z)}] \quad (5.29)$$

where

$$v_{k-\bar{T}(\frac{a}{p_i},z)} = Pr\Big(\sum_{l=1}^{i-1} T(r_i, p_i, \sigma_i) = k - \bar{T}(\frac{a}{p_i}, z)\Big)$$

The expected security level experienced by disk requests with security re-

quirements can be directly derived from Eq. (5.29). Hence, the expected security level is given by

$$E(\sigma_i) = p_j . \sum_{k=1}^{9} \left(\frac{k}{10} . Pr(\sigma_{ij} = \frac{k}{10}) \right) \tag{5.30}$$

5.5 Evaluation

To evaluate the performance of the ASPAD framework in an efficient way, we simulated a security-aware parallel disk system, into which nine encryption services were integrated. Table 5.1 summarizes important system parameters used to resemble real world disks such as Seagate Barracuda 36ES2 [25].

In our ASPAD framework, we implemented a data-partitioning algorithm to optimize parallelism degrees of large disk I/O requests. We will first compare the performance of a security-aware parallel disk system with the ASPAD framework with that of a non-security-aware parallel disk system without employing ASPAD. We will then study effects of varying arrival rates on the performance of the two disk systems. Next, we will compare and evaluate the two disk systems with respect to security requirements imposed by disk requests. Finally, we also analyze performance impacts of parallelism degrees on the parallel disk systems.

In our simulation experiments, we made use of the following three metrics to demonstrate the effectiveness of the ASPAD scheme.

1. *Satisfied ratio*is a fraction of total arrived disk requests that are found

Table 5.1: Worst-case parameters of disks in the simulated parallel disk system and workload conditions.

Number of disks	2,4,6,8,10,12,14,16
capacity of the disk	18.4GB
Average Seek Time	11.36ms
Average Rotation Time	8.37ms
Blocks revolution per minute	7200RPM
Transfer rate per disk	30MB/Sec.
Arrival Rate	0.1,0.2,0.3,0.4,0.5 No./Sec.
Data Size	100KB, 10MB, 100MB
Security Range	0.5, 0.55, 0.6, 0.65, 0.85, 0.9

to be finished before their desired response time.

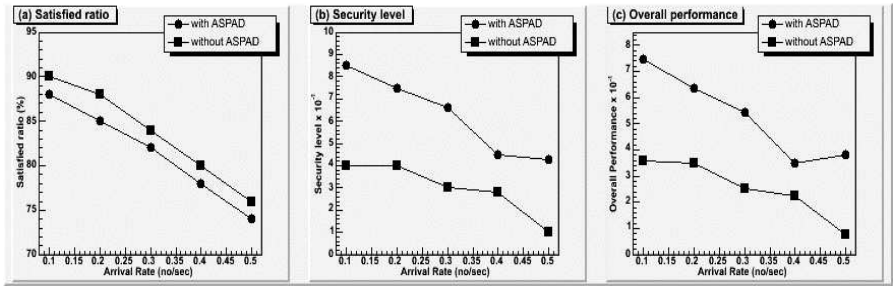
2. *Security level*is a sum of security level values of all disk requests issued to the parallel disk systems.

3. *Overall performance*is performance metric measured by a product of the satisfied ratio and the security level.

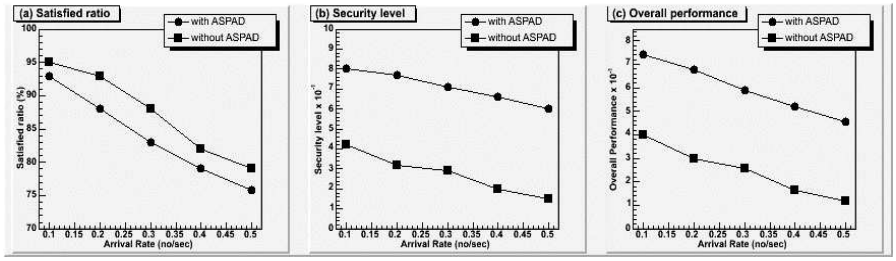
5.5.1 Impacts of arrival rate

This experiment is aimed at comparing the ASPAD strategy with a baseline scheme that makes no use of ASPAD. With different settings of parallelism degrees and data sizes, we study the impacts of varying arrival rates on system performance. To achieve this goal, we increased the arrival rate of I/O requests from 0.1 to 0.5 No./Sec. while setting the parallelism degree to 3 and data size to 100KB, 10MB, and 100 MB, respectively.

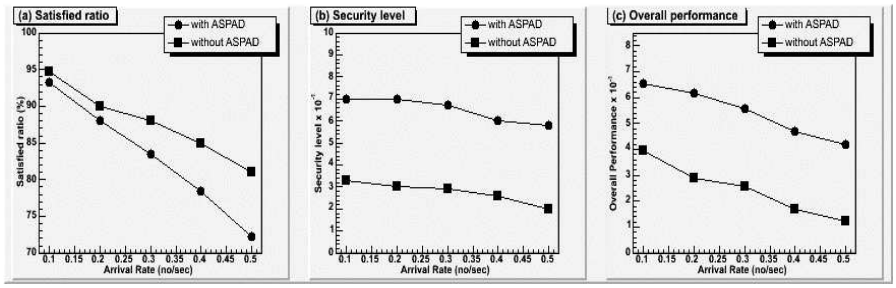
Fig. 5.3 plots the satisfied ratios, security levels, and overall performance of the parallel disk systems with and without ASPAD. Figs 5.3(aa), (ba) and (ca) reveal that the ASPAD strategy yields satisfied ratios that are



a) data size is 100KB and P = 3



b) data size is 10MB and P = 8



c) data size is 100MB and P = 16

Figure 5.3: Impact of arrival rate on the performance of security-aware parallel disk systems

very close to those of the parallel disk system making no use of ASPAD. This is mainly because ASPAD endeavors to guarantee timing constraints of disk requests while maximizing security of the parallel disk system. Figs. 5.3(ab), 5.3(bb), and 5.3(cb) illustrate that ASPAD significantly improves the quality of security of the disk system by an average of 126%. We can attribute the improvement in security to the fact that ASPAD strives to increase security level of each parallel disk

request provided that the corresponding real-time requirement can be met.

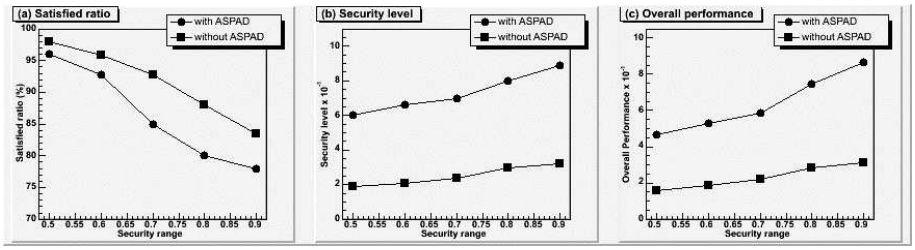
It is observed that as the value of arrival rate increases, the security levels of the both systems decrease. This result is not surprising because high arrival rates lead to high workload, forcing the parallel disk systems to merely meet the minimal security requirements of a vast majority of requests in order to process a large number of requests in a timely manner.

Interestingly, ASPAD always achieves higher security levels compared with the parallel disks without ASPAD. It is worth noting that the security improvement comes at the cost of satisfied ratios (see Figs 5.3(aa), 5.3(ba) and 5.3(ca)), because the satisfied ratios of ASPAD are slightly reduced due to relatively high security overhead caused by the ASPAD strategy. Figs 5.3(ac), 5.3(bc), and 5.3(cc) reveal that ASPAD substantially boosts the overall performance. The reason of the expected overall performance improvement is two-fold. First, ASPAD adaptively enhances the security levels for disk I/O requests. Second, the performance gains in security level can eventually offset the extra security overhead.

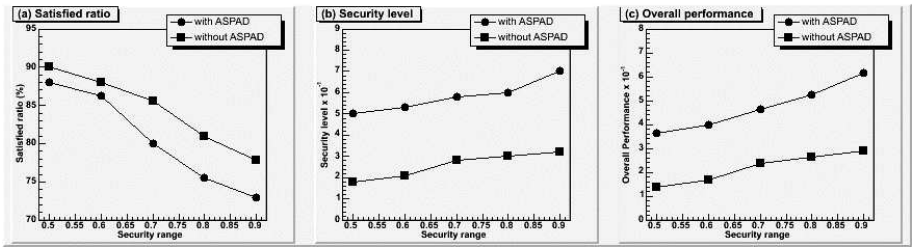
5.5.2 Impacts of security requirements

In this group of experiments, we investigate the performance impacts of security requirements imposed by parallel disk requests. As mentioned earlier, the security requirement of each disk request is characterized by a security range varying from smin to s_{max}. We varied s_{max} from 0.5 to 0.9 while fixing smin to 0.1.

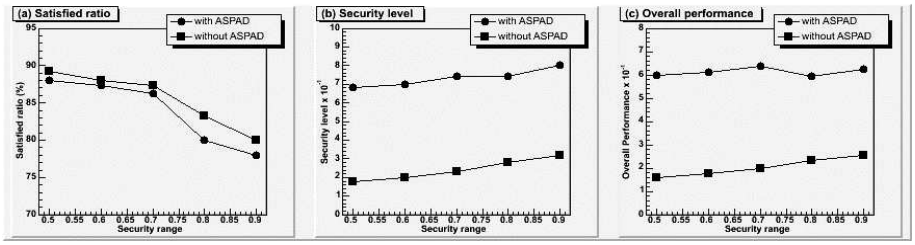
We observed from Figs. 5.4(aa), 5.4(ba), and 5.4(ca) that increasing the maximal security level of the security range leads to the decreasing values of satisfied ratio of the two examined disk systems. This is because when



a) Arrival rate = 0.5no/sec, data size = 100KB, P = 3



b) Arrival rate = 0.5no/sec, data size = 10MB, P = 8



c) Arrival rate = 0.5no/sec, data size = 100MB, P = 16

Figure 5.4: Impact of security requirements on the performance of security-aware parallel

security level requirements of disk requests are high, the parallel disk systems need to fulfill the security requirements with high security overheads, which in turn reduce satisfied ratios.

Again, the satisfied ratios of ASPAD are reasonably close to those of the alternative strategy. It is observed form Figs. 5.4(ab), 5.4(bb), and 5.4(cb) that when smax goes up, the security levels of the two evaluated schemes gradually increase. This result implies that the security levels of the two schemes heavily rely on the security requirements of disk requests. Figs. 5.4(ac), 5.4(bc), and 5.4(cc) illustrate that the overall performance of the

two systems improves with the increasing values of smax, because the overall performance is more sensitive to security level than to satisfied ratio.

5.5.3 Impacts of parallelism degree

Disk I/O parallelisms are implemented in forms of inter-request and intra-request parallelism. Without loss of generality, in this study we considered the intra-request parallelism. Nevertheless, the proposed ASPAD strategy can be readily applied to disk workload conditions with inter-request parallelism.

Now we focus on the effects of parallelism degree on the performance of ASPAD and the alternative. In this set of experiments, the parallelism degree was varied from 2 to 16. Fig. 5.5a shows that when the parallelism degree increases, the performance in terms of satisfied ratio is improved. The rationale behind this observation is that increasing parallelism degrees indicates the increasing number of disks over which a request is served (thereinafter referred to as striping width) and, thus, increasing the striping width has a strong likelihood to reduce response times of the requests and enhance throughput of the parallel disk systems.

It is intriguing to observe from Fig. 5.5b that the high parallelism degrees give rise to high levels of security. We attribute this performance trend to positive impacts of the large striping widths, which substantially decrease the response times. Thus, the decrease in the response times ultimately makes it possible for an increasing number of disk requests to be completed before their deadlines. As shown in Fig. 5.5c, the overall performance improvement of ASPAD over the alternative is more prominent

when the parallelism degree becomes high. This is because, first, ASPAD's performance degradation in satisfied ratio becomes less significant as the parallelism degree increases (see Fig. 5.5a). Second, the security level discrepancy between ASPAD and the competitive scheme is widened as the parallelism degree rises, meaning that high parallelism degrees offer more opportunities for ASPAD to increase security levels of disk requests in a judicious manner.

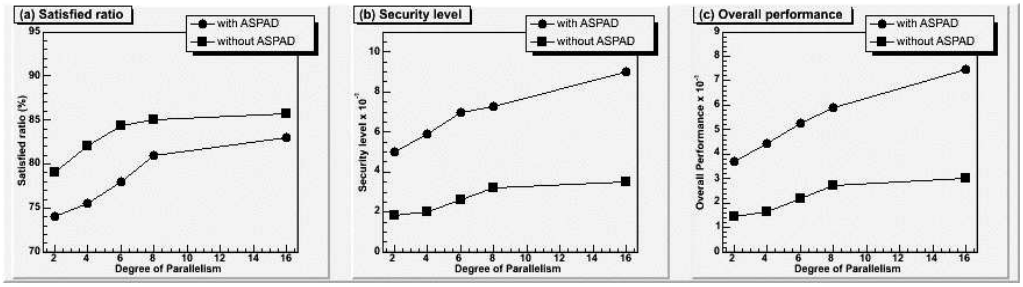


Figure 5.5: The impact of the parallelism degree when arrival rate = 0.5 No./sec.

5.6 Summary

High quality of security and guaranteed response times are two major performance goals to be achieved by parallel disk systems. The reason is two-fold. First, it is often desirable for next generation parallel disk systems to be highly flexible in order to support varying quality of security at different times during a data intensive application lifetime. Second, this trend is especially true for data-intensive applications where disk requests need to be completed within specified response times. In this chapter, we have proposed and evaluated an adaptive quality of security control scheme for parallel disk systems (or ASPAD for short) to protect information in data-intensive applications from being tampered by talented intruders. ASPAD aims at

adapting to changing security requirements and workload conditions in the context of parallel disk systems. To achieve this goal, ASPAD is carried out in three phases: dynamic data partitioning, response time estimation, and adaptive security quality control. Specifically, ASPAD determines the most appropriate cryptographic schemes for disk requests issued by clients, thereby improving security for parallel disk systems and guaranteeing desired response times for the requests. We simulated a security-aware parallel disk system where a data partitioning method was implemented to find optimal values of parallelism degrees. Experimental results demonstratively show that ASPAD significantly outperforms an existing scheme that does not employ the adaptive quality of security controller.

Several important problems remain open. For example, we have ignored network delays in the data portioning method. Our future work will focus on impacts of network delays on performance of ASPAD. Further, we will integrate ASPAD with fault tolerance techniques to provide high availability for critical data-intensive applications.

Chapter 6

A Caching Strategy to Improve Security of Cluster Storage Systems

Cluster storage systems have emerged as high-performance and cost-effective storage infrastructures for large-scale data-intensive applications. Although a large number of cluster storage systems have been implemented, the existing cluster storage systems lack a means to optimize quality of security in dynamically changing environments. We solve this problem by developing a security-aware cache management mechanism (or CaPaS for short) for cluster storage systems. CaPaS aims at achieving high security and desired performance for data-intensive applications running on clusters. CaPaS is used in combination with a security control mechanism that can adapt to changing security requirements and workload conditions, thereby providing high quality of security for cluster storage systems. CaPaS is comprised of a cache partitioning scheme, a response-time estimator, and an adaptive security quality controller. These three components help in increasing quality of security of cluster storage systems while allowing disk requests to be finished before their desired response times. To prove the efficiency of CaPaS, we simulate a cluster storage system into which CaPaS, eight cryptographic, and seven integrity services are integrated. Empirical results show that Ca-

PaS significantly improves overall performance over two baseline strategies by up to 73% (with an average of 52%).

The rest of the chapter is organized as follows. The next section summarizes the related work. Section 3 presents an architecture for cluster storage systems. In section 4, we propose the CaPaS cache partitioning strategy for security-aware cluster storage systems. Section 5 analyzes experimental results. Finally, Section 6 concludes the chapter with future directions.

6.1 **Motivation**

With the advent of powerful processors, fast interconnects, and low-cost storage systems, clusters of commodity PCs have emerged as a primary and cost-effective infrastructure for large-scale scientific and web-based applications. A large fraction of scientific and web-based applications are parallel and data-intensive, because these applications deal with a large amount of data transferred either between memory and storage systems or among nodes of a cluster via interconnection networks. An increasing number of data-intensive applications [77] include long running simulations [77], remote-sensing applications [97], and biological sequence analysis [98], to name just a few. Importantly, cluster storage systems have become increasingly popular for data-intensive applications running on high-performance computing platforms, which assure the effectiveness of the nation's critical infrastructures [78][79]. The concept of cluster storage systems was introduced in conjunction with different domains of applications like Internet data caches [80] and video servers [81][82]. Cluster storage systems are ideal storage candidates for these data-intensive applications running on high-performance clusters,

since the applications need extensive computation coupled with access to diverse data repositories.

Cluster storage systems are an important and essential building block for any high-performance cluster computing infrastructures. In next-generation data grids, some of clusters will be dedicated to data storage systems to support distributed data manipulation functionalities [78]. In the proposed research we will build an infrastructure to conduct research on high-performance cluster storage systems. The specific research issues to be addressed in cluster storage systems are security, scalable disk bandwidth, and high I/O performance. In this study we focus on adaptive security issues in cluster-based architectures for storage systems to support data-intensive computing. This is mainly because clusters make use of inexpensive off-the-shelf PC component to provide high-performance and scalable disk I/O support for data-intensive applications [96]. At the current prices of disks, 100 TB of disk storage can be integrated to a cluster for less than 50,000. Cluster storage systems can provide scalable disk bandwidth by stripping and storing large files across a number of cluster nodes. Thus, cluster systems can readily achieve over 1 GB/Sec aggregate bandwidth by distributing large files across multiple nodes. Note that this performance is 33 times higher than that of a single disk system.

Security services that protect data residing in storage systems from talented intruders are important for data-intensive applications that are security-sensitive in nature [65]. In addition, many data intensive applications require disk requests to be completed within a specified response time [81][65]. Consequently, improving quality of security and guaranteeing desired response

times are two major concerns when it comes to the development of cluster storage systems. In this paper, we propose a cache management technique (or CaPaS for short) that aims at achieving high security and desired performance for cluster storage systems. The cache management scheme is used in combination with a security control mechanism that can adapt to changing security requirements and workload conditions, thereby providing high quality of security for disk systems in general and for cluster storage systems in particular. A salient feature of our cache management mechanism that it contains a cache partitioning technique that helps in increasing quality of security of cluster storage systems while allowing disk requests to be finished before their desired response times.

6.2 **Architecture**

In this study we consider a networked cluster storage system, which consists of a cluster storage system, array of security services, volatile or non-volatile memory, an adaptive security service controller, a security-aware cache partitioning mechanism, and an untrusted network. The architecture of the networked cluster storage system is depicted in Fig.6.1 It is worth noting that this system architecture is general enough to accommodate a wide range of storage systems like network attached storage and storage area networks. The security-aware cache partitioning mechanism, the adaptive security service controller, and the security service controller are at the heart of the cluster storage system connected to the unsecured network. The array of security services is developed to protect disk requests issued by clients.

In the architecture of networked cluster storage systems, clients issue disk

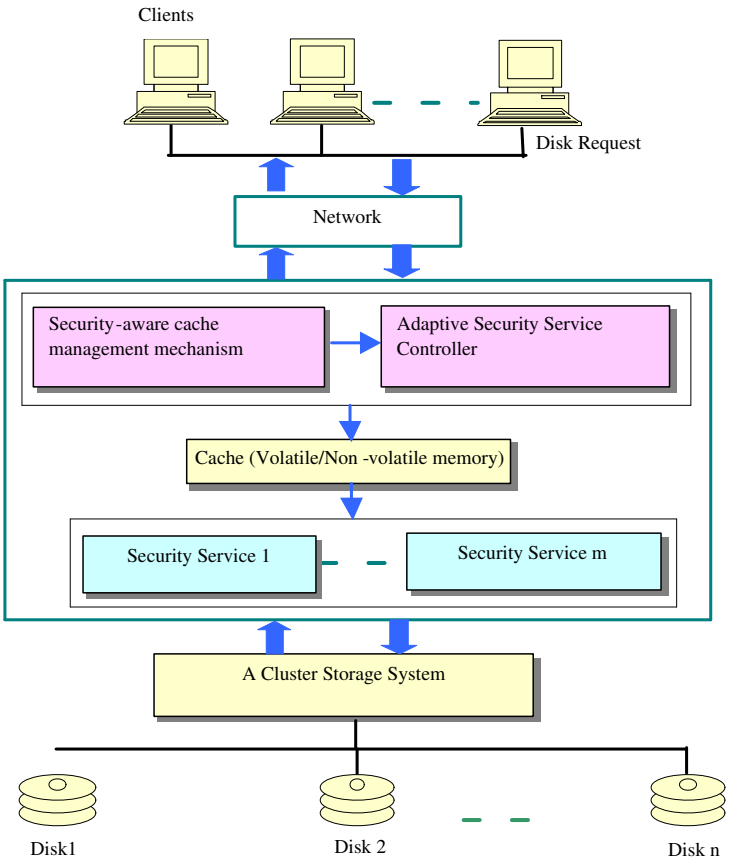


Figure 6.1: Architecture of networked cluster storage systems

requests to a cluster storage system through an untrusted network. The entire cache of the cluster storage system is divided into separate partitions, one for each disk, by a security-aware cache partitioning mechanism. It is believed that if the caches are separately treated, it is easy to control cache miss sequences. Each partition for a disk will be managed separately using the conventional LRU (Least Recently Used) replacement algorithm. The process of cache partitioning is undertaken in a way to maximize quality of security for all disk requests handled by the storage system. A security service controller selects the most appropriate security service to protect data for each disk request.

Note that disk requests issued by clients are read or write requests. It is

called a cache hit if a requested data item is found in the caches. When a requested data item is not in the cache, a cache miss occurs. If there is a cache hit for a read request, the requested data item will be retrieved from the cache and returned to the client through the network interconnection. If it is a cache miss of the read request, the requested data item must be retrieved from the disk. We make use of the write back policy when writing to the cache. Thus, all the information is written to the cache in the first place. The modified data items residing in the cache are written to the cluster storage system only when they are replaced.

Cluster storage systems can exploit I/O parallelism in two possible ways: inter-request (inter-operation) and intra-request (intra-operation) parallelism. Inter-request can be achieved if independent requests can be served by the disk system at the same time. The intra-request allows parallel execution by multiple disks for a single disk request. Our cluster storage system architecture can be applied to address the issues of both inter-request and intra-request parallelism.

6.3 **Security-Aware Cache Management**

We present in this section the security-aware cache management scheme, which aims at improving quality of security of cluster storage systems with dynamically changing workload conditions. In what follows, we first outline the main idea of the security-aware cache management scheme. Next, we describe a way of estimating response times for disk requests submitted to a networked cluster storage system. Finally, we delineate the security-aware cache management algorithm, where dynamic cache partitioning plays an

important role.

6.3.1 **Cache Partitioning**

The performance of cluster storage systems can be substantially improved by employing a large memory space as a cache to reduce the number of disk accesses. Cache management policies are of importance in cluster storage systems, because cache management can judiciously cache data blocks for future accesses. In addition to boosting performance by caching data blocks, cache management mechanism can be geared to improve quality of security by affecting the sequence of disk accesses. Specifically, a cache management scheme can shorten the average response time of disk request, making it possible to increase security overheads to improve security while guaranteeing disk requests' desired response time. In this chapter, we focus on a security-aware cache management scheme or CaPaS, which is used in conjunction with the conventional cache replacement algorithm (the Least Recently Used algorithm or LRU). Note that our cache management approach does not limit itself to the LRU policy; rather, our approach can be employed in combination with other replacement policies (see, for example, MQ [55] and 2Q Error! Reference source not found.) with the inclusion property. CaPaS dynamically partitions cache blocks in such a way to maximize quality of security without violating desired response times. The entire cache of the cluster storage system is divided into separate partitions, one for each disk, by a security-aware cache partitioning mechanism. The cache partitioning process is motivated by observations that storage systems workloads are not equally distributed among the disks [99]. If the caches

are separately treated, it is easy to control cache miss sequences. As such, each cache partition for a disk is managed separately using the conventional LRU replacement algorithm.

To maximize security of a cluster storage system using an optimal cache partitioning, we have to quantify the quality of security experienced by disk requests processed by each disk. Before formalizing the problem, we model a collection of security services required by a disk request r_i as $S_i = s_i^1, s_i^2, ..., s_i{}^4)$ where s_i^j is a required security level range of the jth security service. The adaptive security service controller is intended to determine an appropriate point s_i , i.e, $s_i = s_i^1, s_i^2, ..., s_i^4$ where $s_i^j \in S_i^j$, $1 \leq j \leq q$.

In this study we are concerned with desired response time, which is one of performance requirements. As such, we denote the desired response time of disk request r_i by t_i. We focus on modeling security benefits gained by disk requests with intra-request parallelisms, because disk requests with inter-request parallelisms can be considered as special cases of requests with intra-request parallelisms. Let p_i represent parallelism degree of r_i. The security benefit of request r_i can be expressed as Eq. (6.1)

$$S(r_i, P_d) = \sum_{j=1}^{p} SL(r_{ij}, P_d), p_i \leq m, P_d \leq P \tag{6.1}$$

where P is the total cache size, and P_d is the partition size of the dth disk, m is the number of disks in the cluster storage system and SL_{ij} is the security level of a combination of security services chosen for the jth stripe unit of r_i. It is intuitive to show that the parallelism degree p_i can not exceed the

total number m of disks in the cluster storage system. The security benefit SL_{ij} (see Eq. 6.1) experienced by the jth stripe unit of r_i is written as

$$SL(r_{ij}), P_d) = \sum_{k=1}^{q} w_i^k s_{ij}^k(P_d) \tag{6.2}$$

where s_{ij}^k is the security level of the kth security service applied to the jth stripe unit of ri, and is the weight of the kth security service for the disk request. Note that users specify in disk requests the weights to reflect relative priorities given to the required security services. Given a sequence of disk requests submitted to the dth disk of the cluster storage system via the network, we model the security benefits of the disk requests processed by the dth disk with Pd partition size as the summation of the security levels of all the requests in Rd. Thus, we have:

$$S(R_d, P_d) = \sum_{i=1}^{n} SL(r_{ij}^d, P_d) \tag{6.3}$$

Now we formalize the following non-linear optimization problem as below

$$maximize \ \sum_{d=1}^{m} S(R_d, , P_d) = \sum_{d=1}^{m} \sum_{i=1}^{n} SL(r_{ij}^d, P_d)$$

subject to

$$
\begin{aligned}
&(a)\ \sum_{d=1}^{m} P_d \leq P \\
&(b)\ \forall 1 \leq i \leq n : max_{1 \leq j \leq P}\{\theta_{ij}\} \leq t_i \\
&(c)\ \sigma_{ij} \geq s_i, p_i \leq m
\end{aligned} \tag{6.4}
$$

where θ_{ij} is the response time for the jth stripe unit of request r_i. In an effort to enhance security of the system, we need to guarantee that the following four conditions are met. First, the sum the partition sizes of all the disks must be less than or equal to the total cache size in the cluster storage system. Second, response time of all stripe unit in request r_i must be smaller than the desired response time. Third, the low bound on security level can not be violated. Last but not least, the parallelism degree of r_i has to be smaller than or equal to the number of disks in the system.

6.3.2 **Response Times**

To adaptively adjust the cache size of each disk in a way to improve security levels for disk requests, it is imperative to estimate the disk request's response time, which can be defined as the interval between the time a request is sent by a client and the time the parallel disk system completes corresponding disk I/O operations. Given a newly issued disk request r, the response time of r is estimated by Eq. (6.5).

$$
T(r, P, SL(r)) = T_{queue} + max_{i=1}^{p}\{T_{proc}^{i}(r, P_{d_{(i)}})\} \tag{6.5}
$$

where T_{queue} is the the queueing delay at the client side, p is the parallelism degree, $p_{d(i)}$ is the cache size of the disk serving the ith stripe unit, $SL(r_i, P_{d(i)}$ is the security level (see Eq. 6.2) experienced by the ith stripe unit of request r,

$SL(r) = (SL(r_1, P_{d(1)}), SL(r_1, P_{d(2)}), ..., SL(r_1, P_{d(p)})$ is the request's vector of security levels for its p stripe units, and T^i_{proc} is the system processing delay experienced by the ith stripe unit of the request. With respect to the ith stripe unit of the request, the system processing delay T^i_{proc} can be expressed as

$$T^i_{proc}(r, P_{d(i)}, SL(r_i, P_{d(i)})) = T^i_{security}(r, P_{d(i)}, SL(r_i, P_{d(i)})) + T^i_{network}(r) + T_{cache} + h^i_{cache}(r, P_{d(i)} T^i_{disk}(r)) \quad (6.6)$$

where $T^i_{security}$, $T^i_{network}$, T_{cache}, and T^i_{disk} are the delays at the security mechanism, network subsystem, cache access time, and parallel disk subsystems, respectively. $h^i_{cache}(r, P_{d(i)})$ in Eq. (10) is a step function indicating whether there is a cache hit or miss for the ith stripe unit. Thus, we have

$$h^i_{cache}(r, p_{d_i}) = \begin{cases} 0 & \text{if there is a cache hit} \\ 1 & \text{otherwise, there is a cache miss} \end{cases} \quad (6.7)$$

The delay at the security mechanism, which is also referred to as security overhead, depends on the assigned security level and data size of the stripe

unit. Thus, we can easily derive $T^i_{security}(r, p, \sigma_i)$ from Eq. (6.1) as

$$T^i_{security}(r, P_{d(i)}, SL(r_i, P_{d(i)})) = T_{security}(SL(r_i, P_{d(i)}, \frac{d}{p}) = \sum_{k=1}^{q} \frac{d}{p\mu^k(s^k_i(P_d(i)))} \quad (6.8)$$

where d is the data size of the request , $\frac{d}{p}$ is the data size of the ith stripe unit, $\mu^k(s^k_i(P_{d(i)}))$ is the throughput of the security service whose security level is $s^k_i(P_{d(i)})$. Eq. (6.8). Note that the overhead induced by each security service equals to the data size divided by the throughput of the corresponding security service.

We assume that when the ith stripe unit of a request arrives at the network queue, there are k stripe units waiting to be delivered to the cluster storage system. Suppose stripe units are transmitted in a first-in-first-out order, all the stripe units that are already in the queue prior to the arrival of the ith stripe unit must be transmitted earlier than the ith stripe unit. Hence, the delay in the network subsystem $T^i_{network}$ can be written as:

$$T^i_{network}(r) = \frac{i.\frac{d}{p} + \sum_{j=1}^{k} dj}{B_{network}} \quad (6.9)$$

where d_j is the data size of the jth stripe unit in the network queue, and network is the effective network bandwidth. Similarly, it is assumed that when the ith stripe unit of the request arrives at disk j, there are k disk requests must be processed by disk j before handling the stripe unit. Thus, the delay in the disk subsystem T^i_{disk} is given by the following formula:

$$T^{i}_{disk}(r) = T_{disk,j}\frac{d}{p} + \sum_{l=1}^{k} T_{disk,j}(d_l) \tag{6.10}$$

where $T_{disk,j}(d)$ is the disk processing time of a request containing d bytes of data. We can quantify $T_{disk,j}(d)$ as follows

$$T_{disk,k}(d) = T_{seek} + T_{rot} + \frac{d}{B_{disk}} \tag{6.11}$$

where T_{seek} and T_{rot} are the seek time and rotational latency, and $\frac{d}{B_{disk}}$ is the data transfer time depending on the data size d and disk bandwidth B_{disk}.

6.3.3 **The CaPaS Algorithm**

We proposed a novel security-aware cache management algorithm or CaPaS for cluster storage systems. The CaPaS algorithm adaptively determines the most appropriate security levels of security mechanisms applied to each read or write request while guaranteeing the desired response time of the disk request. More specifically, the CaPaS algorithm is carried out in the following three phases: Cache Partitioning (see Section 4.1), response time estimation (see Section 4.2), and adaptive quality of security control.

The pseudocode of the CaPaS algorithm is outlined in Fig. 4. When a disk request ri is arriving to a cluster storage system, CaPaS inserts the newly arrived request into a waiting queue based on the desired response time policy (see Step 1 in Fig. 4). The first phase is responsible for dividing the

cache space into n separate partitions (one partition for each disk depending on the workload of each disk). After the cache partitioning phase, CaPaS initializes the security levels of request ri to the minimal levels (see Step 4 in Fig. 4). If the request is read, CaPaS checks whether the requested data item is residing in the cache. If there is a cache hit for the disk request (i.e., the requested data item can be retrieved from the cache), CaPaS repeatedly increase the security level of the security mechanisms serving the request until the desired response time of the request can not be guaranteed or the security levels of the request are approaching 1.0 (see steps 9-11).

If there is a cache miss for the request, the requested data must be accessed from the corresponding disk (see steps 19-20).

Similarly, if there is a cache hit for a write request newly arriving to the storage system, CaPaS adaptively increases the security levels of the selected security mechanisms protecting the write request until its desired response time can not be guaranteed or the security levels exceed the maximal value which is 1.0 in the implementation of CaPaS. In case of a cache miss for a write request, one cached data item must be written back to a disk in accordance with the write back policy (see steps 25-29 in Fig. 4).

Theorem 1. The time complexity of CaPaS is O(nm), where n is the number of disk requests in the waiting queue, m is the maximum security levels.

Proof. It takes O(m) time to increase the security level of each disk request (see Step 8). Since there are n disk requests in the waiting queue and, therefore, the time complexity of improving security of all the requests is: O(n)O(m) = O(nm).

```
Input: r: a newly arrived disk request
       t_i: desired response time of the ith request
       s_i: the ith request's lower bound on security level
       Q, a waiting queue at the client side
1. Insert r into Q based on the earliest desired response time first policy
   /*Phase 1: Cache Partitioning
2. Partition the cache space based on the disks' workload conditions;
3. for each request r_j in the waiting queue Q do
     /* Phase 2: Response Time Estimation */
4.   Initialise the security level σ_ij to the value s_i
5.   Estimate the response time
6.     if the request r_j is read
7.       if the request data is stored in the cache (hit)
           /* Phase 3: adaptive security quality control */
8.         while estimated response time < desired response time t_i do
9.           if σ_ij < 0.9 then /* σ_ij can be further increased */
10.            Increase security level σ_ij by 0.1;
11.            Estimate the response time of the r_j request;
12.          else break /* σ_ij can not be further increased */
13.        end while
14.        if estimated response time > desired response time t_i then
15.                  Decrease security level σ_ij by 0.1;
                 Apply the security service with level σ_ij to the rj request
17.        Deliver the r_j request through the network subsystem to the client
18.    else (miss) /* the request ri is not stored in the cache
19.        Grab the data from disk i and store it in the cache partition C_i
20.        Repeat step 8-17 (Phase 3)
21.  if the request r_j is write request
22.    if the request r_j is in the cache (hit)
23.        Apply steps 8-16
24.        Deliver the data to the disk subsystem
25.        else (miss)
26.          Apply the LRU replacement algorithm to manage the cache
27.          Use the write back policy to write the request in the cache
28.          Repeat steps 8-16
29.          Deliver the request r_j to the disk subsystem
31. end for
```

Figure 6.2: The CaPaS Algorithm

6.3.4 **Optimality of CaPaS Algorithm**

Now we show that the optimality of CaPaS in terms of quality of security in each disk request. Throughout this section we use the following concept of non-secure response time.

The non-secure response time $T^{ij}_{non-secure}$ of the jth stripe unit in disk request r_i is defined as the response time of the stripe unit that is not secures by any security mechanism. We have $T^{ij}_{non-secure} = T_{queue} + T_{cache} + T^{ij}_{network} + T^{ij}_{disk}$.

Theorem 2. Given any stripe unit of a disk request r_i (e.g., the jth stripe unit) and a security-aware parallel disk system, the CaPaS algorithm maximizes the security level σ_{ij} of the stripe unit.

proof. Without loss of generality, let us consider the jth stripe unit in disk request ri. We have to prove the theorem by demonstrating that the security level σ_{ij} of the jth stripe unit cannot be further increased. The first phase in the algorithm attempt to minimize the non-secure response time of all stripe units in disk request ri using the cache partitioning mechanism (see Step 2 in Fig. 4). Consequently, the non-secure response time any stripe unit is minimized. Thus, the non-secure response time $T_{queue} + T_{cache} + T^{ij}_{network} + T^{ij}_{disk}$ of the jth stripe unit is minimized. Since $T_{queue} + T_{partition}$ can not be reduced by CaPaS, the processing algorithm minimizes the value of $T^{ij}_{network} + T^{ij}_{disk}$. Step 8 ensures that the inequality $T_{queue} + T_{cache} + T^{ij}_{network} + T^{ij}_{disk} + T^{ij}_{security}\ leqt_i$ holds. Therefore, we have $T^{ij}_{security} \leq t_i - (T_{queue} + T_{partition} + T^{ij}_{network} + T^{ij}_{disk})$. It is evident that the term $T_{queue} + T_{partition} + T^{ij}_{network} + T^{ij}_{disk}$ on the right-hand side of the inequality is minimized by the first phase of the algorithm, indicating that time $T^{ij}_{security}$ spent in security services is maximized. As a

result, Steps 9 and 10 in the algorithm leverage the maximized time $T^{ij}_{security}$ to optimize the security level σ_{ij}. Hence the proof of Theorem 4 is complete.

Theorem 3. If the security levels of all the stripe units in a disk request ri are maximized in a security-aware parallel disk system, then the CaPaS algorithm optimizes the quality of security for disk request ri. More formally, we have

Proof. The value of can be maximized without adverse effects upon the security levels of the other stripe of ri. Therefore, the security levels of all the stripe units within disk request ri can be maximized simultaneously by the CaPaS algorithm. Consequently, the summation of the security levels of all the stripe unit of the disk request ri is maximized by Steps 9 and 10. i.e., $\sum_{j=1}^{p} \sigma_{ij}$ is maximized. This completes the proof of the theorem.

Theorem 4 below shows the optimality of the CaPaS algorithm.

Theorem 4. Given a disk request ri and a security-aware parallel disk system, CaPaS optimizes the quality of security for disk request ri.

Proof. First, Theorem 2 proves that each stripe unit in disk request ri has the corresponding security level maximized. Next, Theorem 3 shows if the security levels of all the stripe units in a disk request ri are maximized in a security-aware parallel disk system, then the CaPaS algorithm optimizes the quality of security for disk request ri. Hence, the proof of the theorem is immediate from Theorems 2 and 3.

6.4 **Experimental Results**

To evaluate the performance of the CaPaS strategy in an efficient way, we simulated a networked cluster storage system, in which nine confidentiality and nine integrity services were integrated. Table 3 summarizes important system parameters used to resemble real world cluster storage systems. In addition, we implemented the CaPaS cache management algorithm to optimize the quality of security of the cluster storage system. We evaluate the performance of CaPaS using extensive simulation experiments. To reveal the performance improvements gained by our proposed algorithm, we compare CaPaS with two baseline algorithms, which are briefly described below.

1. No Cache Partitioning: the cache space of the cluster storage system is not partitioned; all the disks in the system share the same cache.
2. Uniform Partitioning: the cache space is partitioned equally among the disks. We first investigate the impact of cache size on cache miss rate. Second, we study the effects of varying arrival rates on the performance of the three algorithms. Third, we compare the performance of the three algorithms from the standing point of data size. Fourth, we test the sensitivity of the three cache management algorithms to variations in disk bandwidth. Finally, we analyze the performance impacts of cache size on cluster storage systems.

In our simulation experiments, we made use of the following four metrics to demonstrate the effectiveness of the CaPaS scheme.

Table 6.1: **Disks parameters in the simulated cluster storage system and workload conditions**

Cache Size	128MB
Number of Disks	8
capacity of the disk	18.4GB
Average Seek Time	11.36ms
Blocks revolution per minute	7200RPM
Transfer rate per disk	30MB/Sec.
Arrival Rate	0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8 No./Sec.
Data Size	100,150,200,250,300,350,400,450KB
Disk Bandwidth	10MB/Sec.,15MB/Sec.,20MB/Sec., 25MB/Sec.,30MB/Sec,35MB/Sec,40MB/Sec

1. Satisfied Ratio. It is a fraction of total arrived disk requests that are finished before their desired response times.
2. Average Security Level. It is calculated as a sum of security level values of all disk requests divided by the total number of disk requests.
3. Average Security Overhead. It is the mean time spent in security mechanisms. This metric is measured in seconds.
4. Overall Performance. It is expressed as a product of the satisfied ratio and the average security level.
5. Cache Miss Ratio (or Miss Ratio for short). It is the number of cache misses divided by the number of cache references.

6.4.1 Impact of Cache Size on Cache Miss Ratio

In this section, we evaluate cache miss ratio performance of the three algorithms. The cache size is increased from 1MB to 100MB with an increment of 1MB. Fig. 5 shows the impacts of cache size on the cache miss ratio. We

observe from Fig. 5 that CaPaS has the lowest miss ratio compared with the two alternatives, because CaPaS partitions the cache space in such a way that larger size of cache is allocated to disks with highest workloads.

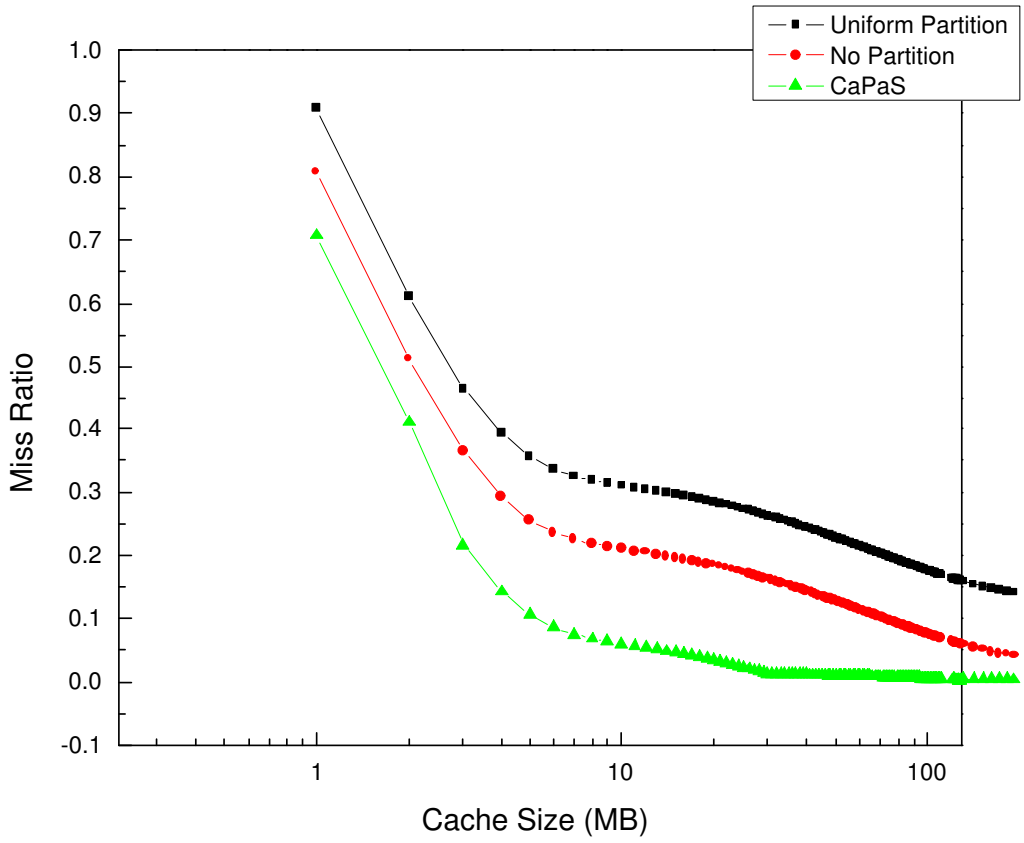


Figure 6.3: Impacts of Cache Size on Cache Miss Ratio

When it comes to the cache management algorithm without partitioning, disks with low load may acquire a large cache size because all the disks share the same cache space. The miss ratio of the algorithm with uniform cache partitions consistently has the highest miss ratio. This is mainly because, using the uniform partition algorithm, disks with low workload simply utilize a very small portion of cache space assigned to them (e.g., some disks only use less than 50% of the allocated cache space). As such, the low cache

utilization of the uniform partition algorithm results in its high miss ratios.

6.4.2 **Impact of Disk Request Arrival Rate**

This set of experiments is aimed at comparing the CaPaS strategy with the two-baseline schemes with respect to disk request arrival rates. With various settings of data size, disk bandwidth, and cache size, we study the impacts of arrival rates on cluster storage system performance. To achieve this goal, we increased the arrival rate of disk requests from 0.1 to 0.8 No./Sec. while setting the data size to 300KB, the disk bandwidth to 25MB/Sec., and cache size to 40MB. Figs. 6 and 7 show performance impacts of the confidentiality and integrity services, respectively. Fig. 6a reveals that CaPaS outperforms the other two algorithms in terms of satisfied ratio significantly. We attribute this improvement to the fact that CaPaS judiciously partitions the system cache space among the multiple disks in the system; therefore, CaPaS allows more disk requests to be finished before their desired response times by lowering down cache miss ratios (see Fig. 5).

When it comes to the cache management algorithm without partitioning, disks with low load may acquire a large cache size because all the disks share the same cache space. The miss ratio of the algorithm with uniform cache partitions consistently has the highest miss ratio. This is mainly because, using the uniform partition algorithm, disks with low workload simply utilize a very small portion of cache space assigned to them (e.g., some disks only use less than 50% of the allocated cache space). As such, the low cache utilization of the uniform partition algorithm results in its high miss ratios.

We observe from Figs. 6b and 6c that CaPaS delivers better average security

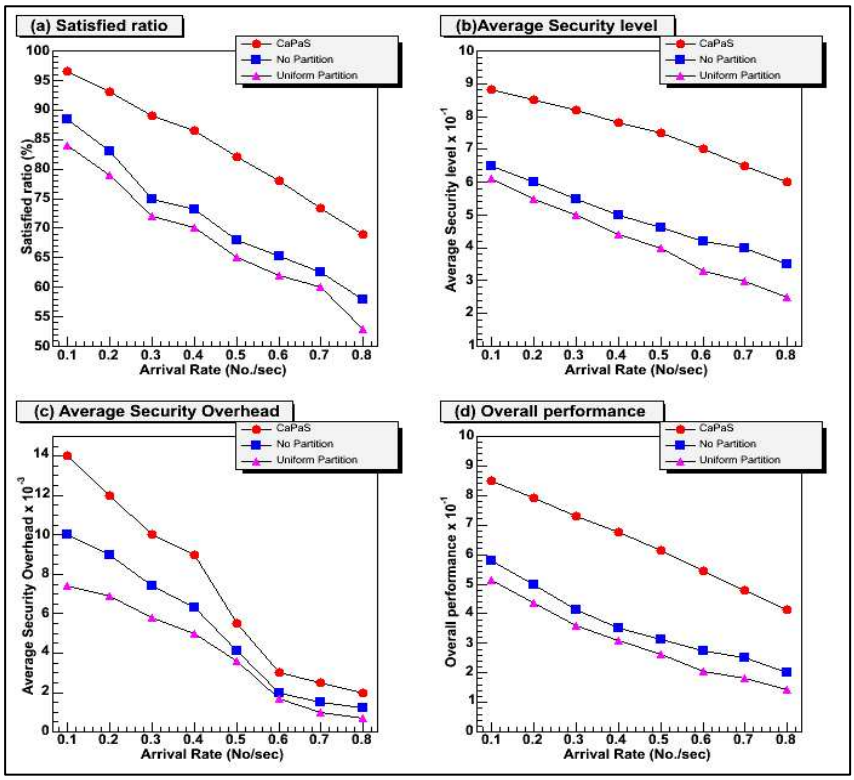


Figure 6.4: Performance impact of confidentiality services where data size = 300KB, disk bandwidth = 25MB/Sec, and cache size = 40 MB

levels and higher security overheads compared with the other two algorithms under a wide range of workload conditions. Comparing the experimental results in Figs. 6 and 7, we realize that the security improvements become more pronounced for the confidentiality services than the integrity services. This is because the performance of the confidentiality services are higher than that of the integrity services, leaving larger slack times to further improve security levels. Besides, the average security overhead of the integrity services is significantly higher than that of the confidentiality services. The main reason is that the performance of the nine algorithms in integrity services is extremely low compared with those in the confidentiality services; and the integrity services have higher security overheads (see Fig. 6C). However, CaPaS always achieves higher average security levels compared with

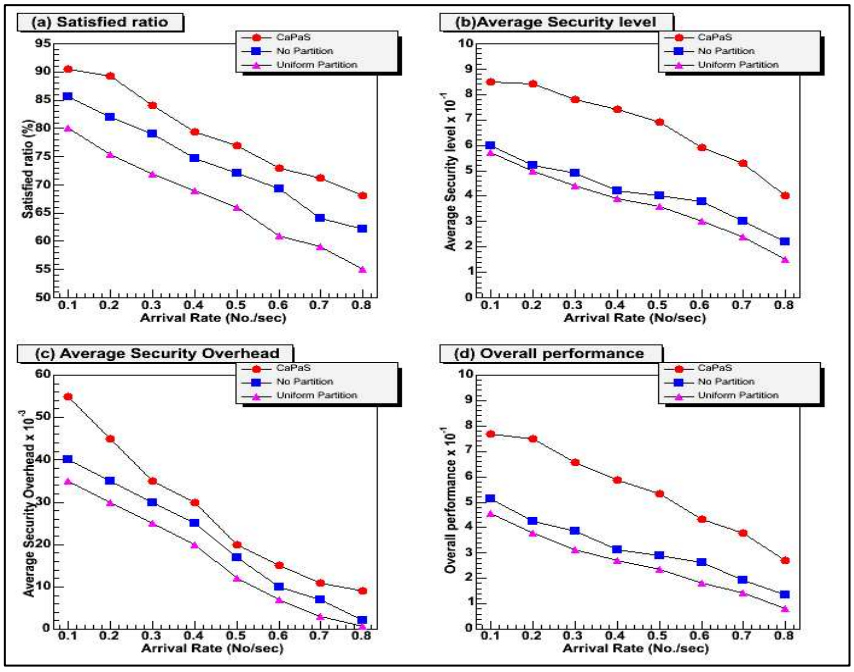


Figure 6.5: Performance impact of integrity services where data size = 300KB, disk bandwidth = 25MB/Sec, and cache size = 40 MB

the other two strategies. The results can be explained in part by Figs. 6c and 6d showing that the average security overhead of CaPaS is constantly higher than that of the alternatives. Thus, high security levels of CaPaS are achieved at the cost of high security overheads. Fig. 6d clearly reveals that CaPaS noticeably outperforms the alternative strategies in terms of overall performance. Specifically, CaPaS obtains an improvement in overall performance over the alternatives by up to 44% and 37% for the confidentiality and integrity services, respectively. The performance improvement can be explained by the fact that CaPaS adaptively enhances security levels of each disk request under the condition that all requests served in the cluster storage system can be finished before their desired response times.

6.4.3 **Impacts of Data Size**

In this set of experiments, we vary data size from 100KB to 450 KB to examine the performance impacts of data size on cluster storage systems. The other workload parameters are kept unchanged (see Table 3). Figs 8 and 9 show the impacts of data size on a cluster storage system where the confidentiality and integrity services are integrated. Figs. 8a and 8b reveal that when the data size increases from 100KB to 450KB, CaPaS consistently exhibits higher satisfied ratios than the alternative approaches. This observation, which is consistent with Figs. 5 and 6, demonstrates that CaPaS achieves higher performance in satisfied ratio.

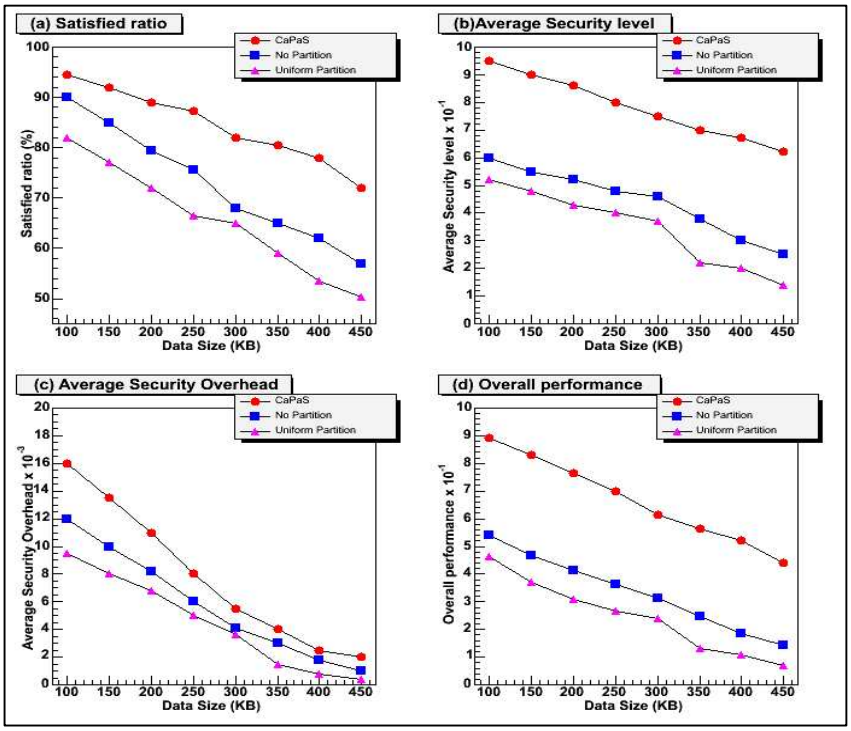


Figure 6.6: Performance impact of confidentiality services where arrival rate = 0.5 No/Sec, disk bandwidth = 25MB/Sec, and cache size = 40 MB.

Likewise, Figs 8b and 9b show significant improvements of CaPaS in security

level over the other two algorithms. The performance improvements can be explained by the fact that CaPaS has the lowest miss rate compared with the alternatives (see Fig. 5). Furthermore, we notice from Figs. 8b and 9b that the average security levels decrease with the increasing value of the data size, because the security overheads drop when the data size increases (see Figs. 8c and 9c). Again, high security levels of CaPaS are achieved at the cost of high security overheads. Finally, Figs. 8d and 9d reveal that CaPaS obtains improvements in overall performance over the alternatives by up 73% and 67% in confidentiality and integrity services, respectively.

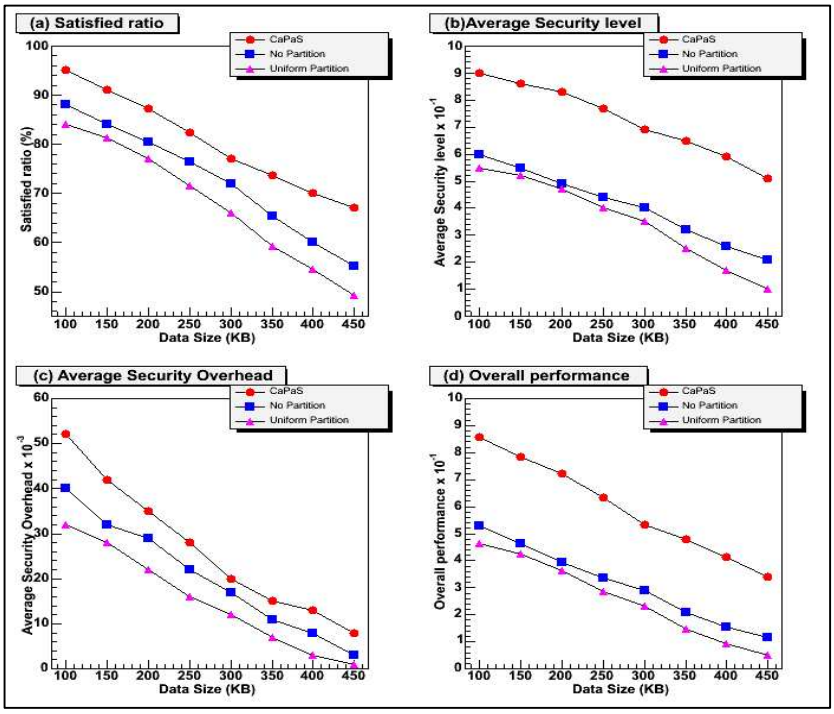


Figure 6.7: Performance impact of integrity services where arrival rate = 0.5 No/Sec, disk bandwidth = 25MB/Sec, and cache size = 40 MB.

6.4.4 **Impact of Disk Bandwidth**

This section is focused on performance impacts of disk bandwidth in cluster storage systems. Specifically, Figs 10 and 11 show the impacts of disk bandwidth on the three evaluated cache management strategies. The disk bandwidth is increased from 10MB/Sec. to 45MB/Sec., whereas the other workload parameters remain unchanged (see Table 6.3).

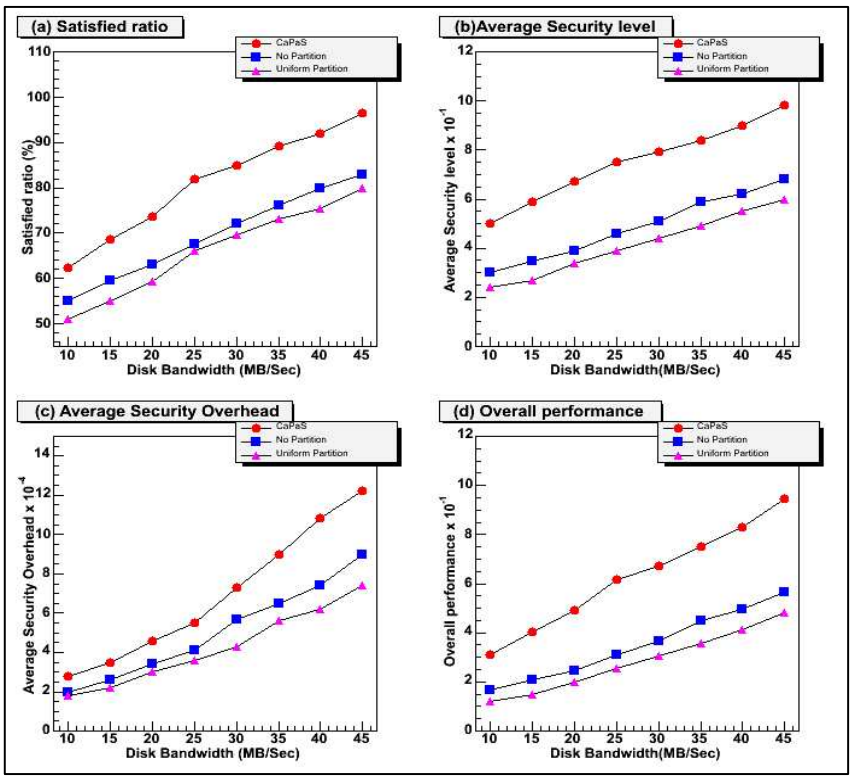


Figure 6.8: Performance impact of confidentiality services where arrival rate = 0.5 No/Sec, data size = 300KB, and cache size = 40 MB.

An important observation drawn from Figs. 10a and 11a is that when the disk bandwidth increases, the satisfied ratios of all the three evaluated strategies are noticeably improved. We attribute this performance trend by the fact that high disk bandwidth gives rise to short data transfer times, which in turn result in shortened processing times of disk requests. Consequently,

more disk requests can be finished before their desired response times.

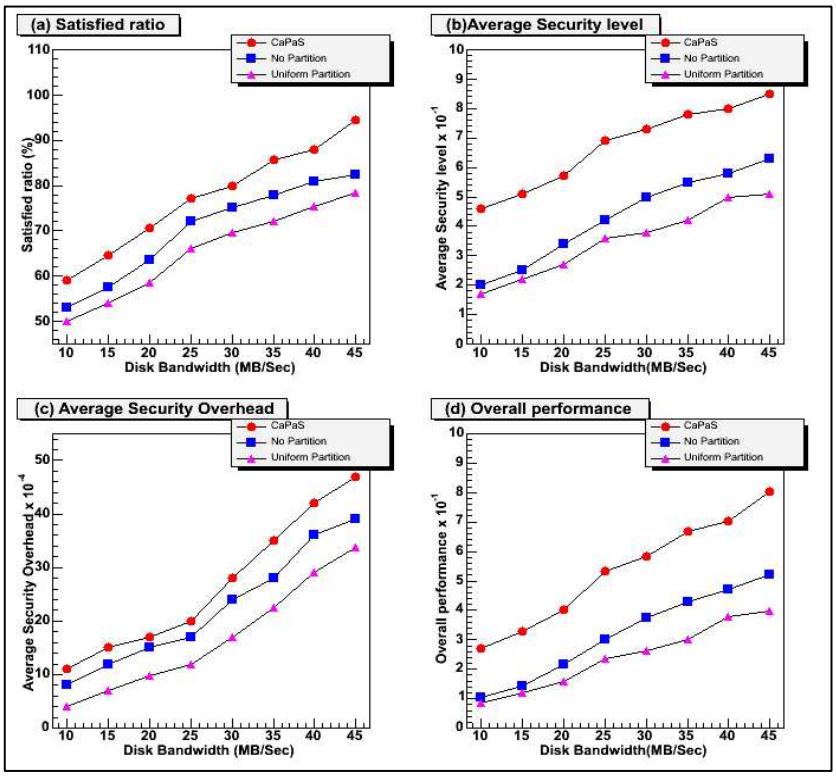


Figure 6.9: Performance impact of integrity services where arrival rate = 0.5 No/Sec, data size = 300KB, and cache size = 40 MB.

Again, Figs. 10b and 11b show that CaPaS is superior to the other two strategies in security level. In addition, the average security level of the confidentiality services is higher than that of the integrity services; this result is consistent with the result reported in Fig. 6b and 7b. Thanks to shortened disk processing time as the disk bandwidth goes up, CaPaS maintains high average security level at the expense of high security overhead (see Figs. 10c and 11c). Because of the increased satisfied ratios and average security levels, the overall performance of CaPaS significantly outperforms the other two algorithms as shown in Figs. 10d and 11d.

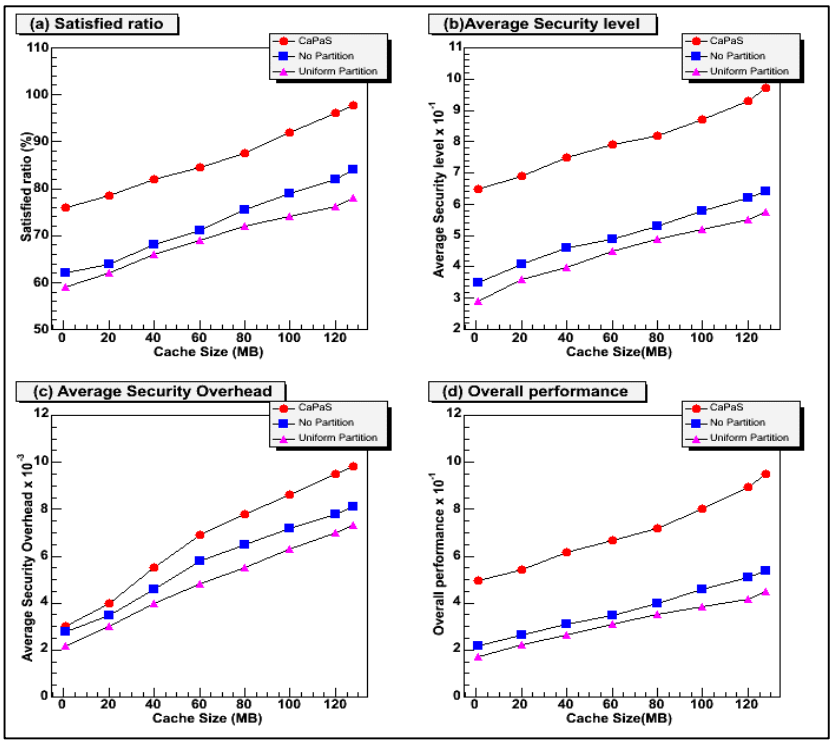


Figure 6.10: Performance impact of confidentiality services where arrival rate = 0.5 No/Sec, data size = 300KB, and disk bandwidth = 25MB/Sec.

6.4.5 **Impact of Cache Size**

In this group of experiments, we studied the performance impacts of cache size when it is varied from 1MB to 128MB with an increment of 20MB.

Figs. 12a and 13a show that when the cache size is gradually increased all the way up to 128MB, the satisfied ratios of the three algorithms also increase accordingly. This is because the increasing cache size leads to low cache miss ratios. A second observation is that CaPaS outperforms the other two alternatives in term of satisfied ratio; this can be explained by Fig. 5 which shows that CaPaS has the lowest cache miss rate among the three evaluated schemes. It is revealed from Fig. 12b and 13b that for all the three strategies, the increasing value of the cache size helps to increase

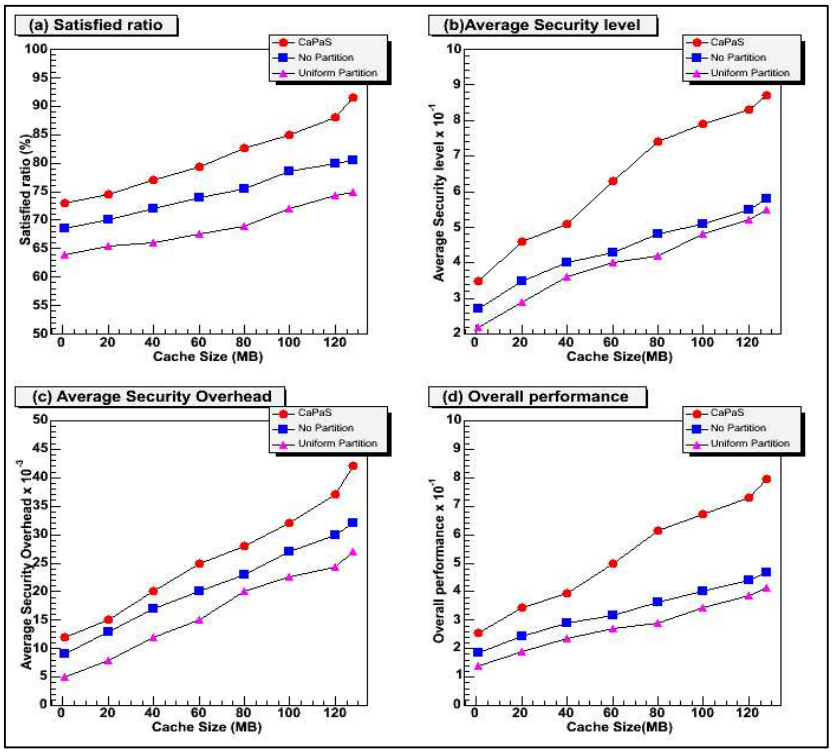


Figure 6.11: Performance impact of integrity services where arrival rate = 0.5 No/Sec, data size = 300KB, and disk bandwidth = 25MB/Sec.

the average security levels. It is worth noting that the security overhead of CaPaS is higher than those of the other two schemes (see Figs 12c and 13c); the overall performance of CaPaS is the best among the three evaluated cache management approaches (see Figs. 12d and 13d).

6.5 **Summary**

It is often desirable for next generation cluster storage systems to be highly flexible in order to support varying quality of security at different times during a data-intensive application lifetime. In addition, this trend is especially true for data-intensive applications where disk requests need to be completed within specified response times. As such, high quality of secu-

rity and guaranteed response times are two major performance goals to be achieved by cluster storage systems. In this chapter, we have proposed a security-aware cache management mechanism (or CaPaS for short) for cluster storage systems. CaPaS, which is used in combination with a security control mechanism, can achieve high security and desired performance for cluster storage systems. CaPaS is conducive to providing high quality of security for cluster storage systems by adapting to changing security requirements and workload conditions. Importantly, CaPaS is comprised of a cache partitioning scheme, a response-time estimator, and an adaptive security quality controller. These three components help in increasing quality of security of cluster storage systems while allowing disk requests to be finished before their desired response times. We simulated a cluster storage system where the CaPaS cache management approach, eight cryptographic, and seven integrity services are implemented. Experimental results demonstratively show that CaPaS significantly improves overall performance over two baseline strategies by up to 73

Our future work will focus on the implementation of CaPaS in a parallel file system (e.g. PVFS) on clusters. Further, we will integrate fault tolerance schemes with CaPaS in a way to provide high reliability for critical data-intensive applications. Last but not least, we plan to address load balancing issues in the context of security-aware caching strategies.

Chapter 7

Conclusions and Future Work

In this dissertation, we have developed a number of adaptive quality of security control strategies for data-intensive applications. This chapter concludes the dissertation by summarizing the contributions and describing future directions. The chapter is organized as follows: Section 7.1 highlights the main contributions of the dissertation. In Section 7.2, we concentrate on some future directions, which are extensions of our past and current research on load-balancing support for data-intensive applications.

7.1 Main Contributions

7.1.1 A Quality of Security Framework for Storage Resources

In the first phase of this study, we have developed a quality of security framework or StReD for storage resources in Data Grids. In this framework, we integrate quality of security adaptor with an array of security services. Applications running in the framework are enabled to specify flexible security requirements and desired response times of disk requests. The framework leverages the adaptor to dynamically control quality of security for disk re-

quests, thereby achieving good tradeoffs between security and storage system throughput. Experimental results based on a simulated Grid with storage resources show that the proposed framework is capable of significantly improving quality of security and guaranteeing desired response times of disk requests.

7.1.2 Modeling and Improving Security of Local Disk Systems

In the second part of the dissertation study, we devised an adaptive strategy (referred to as AWARDS) that can judiciously select the most appropriate security service for each write request while endeavoring to guarantee the desired response times of all disk requests. To prove the efficiency of the proposed approach, we build an analytical model to measure the probability that a disk request is completed before its desired response time. The model also can be used to derive the expected value of disk requests' security levels. Empirical results based on synthetic workloads as well as real I/O-intensive applications show that AWARDS significantly improves overall performance over an existing scheme by up to 358.9% (with an average of 213.4%).

7.1.3 Quality of Security Adaptation in Parallel Disk Systems

We designed, implemented, and evaluated a quality of security control framework for parallel disk systems, or ASPAD for short, that makes it possible for parallel disk systems to adapt to changing security requirements and workload conditions. The ASPAD framework comprises four major components, namely, a security service middleware, a dynamic data partitioning mechanism, a response time estimator, and an adaptive security quality controller.

The framework is conducive to adaptively and expeditiously determining security services for requests submitted to a parallel disk system in a way to improve security of the disk system while making an effort to guarantee desired response times of the requests. We conduct extensive experiments to quantitatively evaluate the performance of the proposed ASPAD framework. Empirical results show that ASPAD significantly improves the overall performance of parallel disk systems over the same disk systems without using the ASPAD framework.

7.1.4 **A Caching Strategy to Improve Security of Cluster Storage Systems**

Cluster storage systems have emerged as high-performance and cost-effective storage infrastructures for large-scale data-intensive applications. Although a large number of cluster storage systems have been implemented, the existing cluster storage systems lack a means to optimize quality of security in dynamically changing environments. We solve this problem by developing a security-aware cache management mechanism (or CaPaS for short) for cluster storage systems. CaPaS aims at achieving high security and desired performance for data-intensive applications running on clusters. CaPaS is used in combination with a security control mechanism that can adapt to changing security requirements and workload conditions, thereby providing high quality of security for cluster storage systems. CaPaS is comprised of a cache partitioning scheme, a response-time estimator, and an adaptive security quality controller. These three components help in increasing quality of security of cluster storage systems while allowing disk requests to be finished

before their desired response times. To prove the efficiency of CaPaS, we simulate a cluster storage system into which CaPaS, eight cryptographic, and seven integrity services are integrated. Empirical results show that CaPaS significantly improves overall performance over two baseline strategies by up to 73% (with an average of 52%).

7.2 Future Work

In this dissertation research, there are several important opening issues, which will be addressed in my future studies. This section concludes the dissertation by outlining several future directions.

7.2.1 Reliability-Aware Load Balancing in Parallel Storage Systems

The design goal of load balancing in parallel disk systems is to allocate disk requests on individual disks to achieve good throughput. To achieve this goal, I plan to take a holistic way to develop a load balancing mechanism, which keeps track of the following statistics:

- 1 The heat of extents and disks, where the heat is defined as the sum of the number of block accesses of an extent or disk per time unit, as determined by statistical observation over a certain period of time.
- 2 The temperature of extents, which is defined as the ratio between heat and size.

In addition, a reliability model will be seamlessly integrated into the load-balancing mechanism. With the novel reliability model in place, the load

balancing mechanism will be striving to make the best tradeoff between performance and reliability.

References

[1] D. Avitzour, "Novel scene calibration procedure for video surveillance systems," IEEE Trans. Aerospace and Electronic Systems, Vol. 40, No. 3, pp. 1105-1110, July 2004.

[2] E. Balafoutis, G. Nerjes, P. Muth, M. Paterakis, G. Weikum, P. Triantafillou, "Clustered Scheduling Algorithms for Mixed-Media Disk Workloads in a Multimedia Server," J. of Cluster Computing, Vol. 6, No. 1, pp. 75-86, 2003.

[3] M. Blaze, "A Cryptographic File System for UNIX," Proc. ACM Conf. Communications and Computing Security, 1993.

[4] R. Bordawekar, R. Thakur, and A. Choudhary, "Efficient Compilation of Out-of-core Data Parallel Programs," Technical Report, SCCS-662, NPAC, April 1994.

[5] P. Bosch and S. J. Mullender, "Real-Time Disk Scheduling in a Mixed-Media File System," Proc. Real-Time Technology and Applications Symp., pp. 23-33, 2000.

[6] J. Bruno, E. Gabber, B. Ozden, and A. Silberschatz, "Disk Scheduling Algorithms with Quality of Service Guarantees," Proc. IEEE Conf. Multimedia Computing Sys., June 1999.

[7] C. Chang, B. Moon, A. Acharya, C. Shock, A. Sussman, and J. Saltz. "Titan: a High-Performance Remote-Sensing Database," Proc. the 13th Int'l Conf. Data Engineering, Apr 1997.

[8] Y. Cho, M. Winslett, M. Subramaniam, Y. Chen, S. Kuo, and K. E. Seamons, "Exploiting Local Data in Parallel Array I/O on a Practical Network of Workstations," Proc. the Fifth Workshop on I/O in Parallel and Distributed Systems, pp.1-13, Nov. 1997.

[9] Jr. Coffman and M. Hofri, "Queueing Models of Secondary Storage Devices," Stochastic Analysis of Computer and Communication Systems, Ed. Hideaki Takagi, North-Holland, 1990.

[10] P. J. Denning, "Effects of Scheduling on File Memory Operations," Proc. AFIPS Conf., 1967.

[11] Z.Dimitrijevic and R. Rangaswami, "Quality of Service Support for Real-time Storage Systems," Proc. Int'l Conf. IPSI, Sv. Stefan, Montenegro, October 2003.

[12] B. Forney, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Storage-Aware Caching: Revisiting Caching for Heterogeneous Storage Systems," Proc. Int'l Symp. File and Storage Tech., 2001.

[13] J. Huber, C. L. Elford, D. A. Reed, A. A. Chien, and D. S. Blume, "PPFS: A high Performance Portable Parallel File System," Proc. 9th ACM Int'l Conf. Supercomputing, pp.385-394, July 1995.

[14] J.Hughes and D. Corcorna, "A Universal Access, Smart-Card-Based, Secure File System," Atlanta Linux Showcase, Oct. 1999.

[15] D. Jacobson and J. Wilkes, "Disk Scheduling Algorithms Based on Rotational Position," Technical Report, HPL-CSP-91-7, February 1991.

[16] W. B. Ligon and R. B. Ross, "Implementation and Performance of a Parallel File System for High Performance Distributed Applications," Proc. IEEE Int'l Symp. High Performance Distributed Computing, pp.471-480, August 1996.

[17] X. Ma, M. Winslett, J. Lee, and S. Yu, "Faster Collective Output through Active Buffering," Proc. Int'l Symp. Parallel and Distributed Processing, 2002.

[18] D. Mazieres, M. Kaminsky, M. Kaashoek and E. Witchel, "Separating key management from file system security," Proc. ACM Symp. Operating System Principles, December 1999.

[19] E. Miller, D. Long, W. Freeman and B. Reed, "Strong Security for Distributed File Systems," Proc. Symp. File Systems and Storage Technologies, January 2002.

[20] E. Nahum, S. O'Malley, H. Orman, R. Schroeppel, "Towards High Performance Cryptographic Software," Proc. IEEE Workshop Arch. and Impl. of High Perf. Comm. Subsystems, August 1995.

[21] K. W. Preslan, A. P. Barry, J. E. Brassow, G. M. Erickson, E. Nygaard, C. J. Sabol, S. R. Soltis, D. C. Teigland, and M. T. O'Keefe, "64-bit, Shared Disk File System for Linux," Proc. NASA Goddard Conference on Mass Storage Systems, March 1999.

[22] X. Qin, H. Jiang, Y. Zhu, and D. R. Swanson. "Improving the Performance of I/O-Intensive Applications on Clusters of Workstations" Cluster Computing: The Journal of Networks, Software Tools and Applications, Vol. 8, No. 4, Oct. 2005.

[23] A. Reddy and J. Wyllie, "Intergraded QoS Management for Disk I/O," Proc. IEEE Conf. Multimedia Computing Sys., June 1999.

[24] R. Reist and S. Daniel, "A Continuum of Disk Scheduling Algorithms," ACM Trans. on Computer Sys., pp.77-92, Feb. 1987.

[25] L. Reuther, M. Pohlack, "Rotational-Position-Aware Real-Time Disk Scheduling Using a Dynamic Active Subset," Proc. IEEE Real-Time System Symp, 2003.

[26] E. Riedel, M. Kallahalla, and R. Swaminathan, "A Framework for Evaluating Storage System Security," Proc. the 1st Conf. File and Storage Technologies, Monterey, CA, Jan. 2002.

[27] K. Salem and H. Garcia-Molina, "Disk Striping," Proc. the 2nd Int'l Conf. Data Engineering, pp.336-342, Feb. 1986.

[28] P.Scheuermann, G. Weikum, P. Zabback, "Data Partitioning and Load Balancing in Parallel Disk Systems," The VLDB Journal, pp. 48-66, July, 1998.

[29] M. Seltzer, P. Chen, and J. Ousterhout, "Disk Scheduling Revisited," Proc. USENIX Technical Conf., pp.313-323, 1990.

[30] P. Shenoy and H. Vin, "Cello: A Disk Scheduling Framework for Next Generation Operating Systems," Proc. ACM SigMetrics, June 1998.

[31] T. Sumner and M. Marlino, "Digital libraries and educational practice: a case for new models," Proc. ACM/IEEE Conf. Digital Libraries, pp. 170 - 178, June 2004.

[32] T. Tanaka, "Configurations of the Solar Wind Flow and Magnetic Field around the Planets with no Magnetic field: Calculation by a new

MHD," J. Geophysical Research, pp.17251-17262, Oct. 1993.

[33] T. Xie and X. Qin, "A New Allocation Scheme for Parallel Applications with Deadline and Security Constraints on Clusters," Proc. IEEE Int'l Conf. Cluster Computing, Boston, USA, Sept. 2005.

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

[35] T. Xie and X. Qin, "Enhancing Security of Real-Time Applications on Grids through Dynamic Scheduling," Proc. 11th Workshop Job Scheduling Strategies for Parallel Processing, June 2005.

[36] P. S. Yu, M. S. Chen, and D. D. Kandlur, "Grouped Sweeping Scheduling for DASD-Based Multimedia Storage Management," ACM Multimedia Systems, Vol.1, No.3, pp.99-109, 1993.

[37] B. Tierney, W. Johnston, J. Lee, M. Thompson, A data-intensive distributed computing architecture for grid applications, Future Generation Computer Systems, 16(5), (2000), pp. 473-481.

[38] X. Qin, " Design and Analysis of a Load Balancing Strategy in Data Grids," Future Generation Computer Systems 23 (1) (2007) 132-137.

[39] T. Xie and X. Qin, "Scheduling Security-Critical Reat-Time Applications on clusters," IEEE Transaction on Computers, vol. 55, no. 7, pp. 864-879, july 2006.

[40] A. Breckenridge, L. Pierson, S. Sanielevici, J. Welling, R. Keller, U. Woessner, J. Schulze, Distributed, on-demand, data-intensive and collaborative simulation, analysis, Future Generation Computer Systems, 19(6), (2003), pp.849-859.

[41] W. Allcock, et al, J. Bresnahan, J. Bunn, Grid-enabled particle physicz event analysis: experiences using a 10 gb, high-latency network for a high-energy physics application, Future Generation Computer Systems, 19(6), (2003), pp.983-997.

[42] M. S. Prez, J. Carretero, F. Garca, J. M. Pea, V. Robles, Mapfs: A flexible multiagent parallel file system for cluster, Future Generation computer Systems, 22(5), (2006), pp. 620-632.

[43] M. Tang, B.-S. Lee, X. Tang, C.-K. Yeo, "The impact of data replication on job scheduling performance in the data grid," Future Generation Computer Systems, 22(3) (2006) 254-268.

[44] X. Qin and H. Jiang, "Data Grids: Supporting Data-Intensive Applications in Wide Area Networks," High Performance Computing: Paradigm and Infrastructure, pp. 481-494, edited by Laurence T. Yang and Minyi Guo, John Wiley and Sons, spring, 2005.

[45] W.W. Chu, M.T. Lan, and J. Hellerstein, "Estimation of inter module communication (IMC) and its applications in distributed processing systems," IEEE Trans. on Computers, pp.691-299, Aug., C-33, 1984.

[46] K. Connelly and A. A. Chien, "Breaking the barriers: high performance security for high performance computing," Proc. Workshop on New security paradigms, Virginia, Sept. 2002.

[47] M. Bishop, Computer Security, ISBN 0-201-44099-7, Addison-Wesley, 2003.

[48] Jorge Barbosa, C. Morais, R. Nobrega, and A.P. Monteiro, "Static scheduling of dependent parallel tasks on heterogeneous clusters,"

IEEE International Conference on Cluster Computing, Boston, Massachusetts, USA, September 27 - 30, 2005.

[49] M. Faerman, A. Su, R. Wolski, F. Berman, "Adaptive Performance prediction for distributed data-intensive applications,"ACM Special Interest Group on Computer Architecture", 1999.

[50] K. Keeton, A. Merchant," A framework for evaluating storage system dependability," IEEE International Conference on Dependable System and Networks, 2004

[51] R. Watson, R. Coyne, "The parallel I/O architecture of the high-performance storage system (HPSS)," Proc. Of the fourteenth IEEE Symposium on Mass Storage Systems, 1995.

[52] Xubin He, Ming Zhang, Qing Yang," SPEK: a storage performance evaluation kernel module for block-level storage systems under faulty conditions," IEEE Transaction on Dependable and Secure Computing, Volume 2, issue 2, pp. 138-149.

[53] Chambliss, et al.," Performance virtualisation for large-scale storage systems,"Proc. 22nd International Symposium on Reliable Distributed Systems, 2003.

[54] Hogil Kim, E.J. Kim and Rabi N Mahapatra, " Power Management in RAID Server Disk System Using Multiple Idle States", in the Proceedings of International Workshop on Unique Chips and Systems (UCAS) 2005, pp. 53-59.

[55] Qingbo Zhu, Asim Shankar and Yuan Zhou, "PB-LRU: A Self-Tuning Power Aware Storage Cache Replacement Algorithm". The 18th Annual

ACM International Conference On Supercomputing (ICS-04), Saint-Malo, France, June26-july 1, 2004.

[56] Qinbgo Zhu, Francis M. David, Christo F. Devaraj, Zhenmin Li, Yuanyuan Zhou and Pei Cao, " Reducing Energy Consumption of Disk Storage Using Power-Aware Cach Management", the Tenth International Symposium on High Performance Computer Architecture (HPCA-10), Madrid, Spain, February 14-18, 2004.

[57] S. Rajasekaran, "Selection algorithms for parallel disk systems," Proc. Int'l Conf. High Performance Computing, pp.343-350, Dec. 1998.

[58] M. Kallahalla and P. J. Varman, "Improving parallel-disk buffer management using randomized writeback," Proc. Int'l Conf. Parallel Processing, pp. 270-277, Aug. 1998.

[59] P. Scheuermann, G. Weikum, and P. Zabback, "Data portioning and load balancing in parallel disk systems," VLDB Journal, Vol.7, pp.48-66, 1998.

[60] S. Rajasekaran and X. Jin, "A practical realization of parallel disks Parallel Processing," Proc. Int'l Workshop Parallel Processing, pp. 337-344, Aug. 2000.

[61] D. Kotz and C. Ellis, "Cashing and writeback policies in parallel file systems," Proc. IEEE Symp. Parallel and Distributed Processing, pp. 60-67, Dec. 1991.

[62] J. Wylie, M. Bigrigg, M. Strunk, G. Ganger, H. Kilicote, p. Khosla," Survivable information storage systems," IEEE Computer Society vol. 33, no. 8, pp. 61-68, 2000.

[63] W. Leung, L. Miller, " Scalable security for large, high performance storage systems," Proc. of the second ACM workshop on storage security and survivability, pp. 29-40, 2006.

[64] M. Nijim, X. Qin, T. Xie, " Modeling and improving security of a local disk system for write-intensive workloads", ACM Transaction on Storage, vol. 2, no. 4, pp. 400-423, 2006.

[65] M. Nijim, T. Xie, X. Qin, " AWARDS: An Adaptive Write Scheme for Secure Local Disk Systems," Proc. IEEE International Conference on Information Reuse and Integration, Aug. 2005.

[66] M. Nijim, T. Xie, Z. Zong, X. Qin, " An Adaptive Strategy for Secure Distributed Disk Systems", NASA/IEEE Conference on Mass Storage Systems and Technologies, Work-in-progress Session, May 2006.

[67] M. Nijim, X. Qin" Adaptive Quality of Security Control In Networked Parallel Disk Systems (ICCCN), Arlington, Virginia, Oct. 2006.

[68] C. Irvince, T. Levin, " Quality of security service," Proceeding of the 2000 workshop on new security paradigms, pp. 91-99, 2001.

[69] Z. Xia, J. Wang, X.L. Qin, " Dynamically negotiated security service for multicast application in the active network," IEEE Proceeding on Communications, vol. 153, no.5, 2006.

[70] J. Huang, Y. Wang and F. Cao, "On Developing Distributed Middleware Services for QoS- and Criticality-Based Resource Negotiation and Adaptation," Real-Time Systems 16(2): 187-221; May 1999.

[71] T. Xie, A. Sung, X. Qin," Dynamic task scheduling with security awareness in real-time systems", Proc. 19th IEEE International Conference on Parallel and Distributed Processing Symposium, 2005.

[72] Y. Hu, T. Nightingale, Q. Yang," RAPID-Cache-A reliable and inexpensive write cache for high performance storage systems, "IEEE Transactions on Parallel and Distributed Systems, vol. 13, no.3, 2007.

[73] L. Ou, X. He, M. Kosa, S. Scott, "a Unified multiple-level cache for high performance storage systems," 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005.

[74] N. Jouppi, " Improving Direct-Mapped Cache performance by the Addition of a Small Fully-Associative Cache Prefetch Buffers," Proc. 17th Int'l Symp, Computer Architecture, pp. 364-373, May 1990.

[75] D. Stiliadis, A. Varma, "Selective Victim Caching: a method to improve the performance of direct-mapped caches, "IEEE Transaction on Computers, vol. 46, no.5, 1997.

[76] R. Karedla, J. Spencer Love, and B. Wherry, "Caching Strategies to Improve Disk System Performance," IEEE Computer, vol. 27, no. 3, pp. 38-46, March 1994.

[77] H. Eom and J.K. Hollingsworth, "Speed vs. Accuracy in Simulation for I/O-Intensive Applications," Proc. Int'l Symp. Parallel and Distributed Processing Symposium, pp. 315 - 322, May 2000.

[78] J.M. Perez, F. Garcia, J. Carretero, A. Calderon, L.M. Sanchez, "Data Allocation and Load Balancing for Heterogeneous Cluster Storage Systems," Proc. 3rd Int'l Symp. Cluster Computing and the Grid, 2003.

[79] P.J., Varman, R.M Verma, "Tight Bounds for Prefetching and Buffer Management Algorithms for Parallel I/O Systems," IEEE Trans. Parallel and Distributed Systems, vol. 10, no. 12, pp. 1262 - 1275.

[80] B. Tierney, W. Johnston, H. Herzog, G. Hoo, G. Jin, J. Lee, L. Chen, D. Rotem, "Distributed Parallel Data Storage System: A Scalable Approach to High Speed Image Servers," Proc. ACM Intl' Conf. Multimedia, 1994.

[81] K. Bell, A. Chien, and M. Lauria, "A High-Performance Cluster Storage Server," Proc. High-Performance Distributed Computing, July 2002.

[82] R. Muntz, J.R. Santos, and S. Berson, "Rio: A Real-Time Multimedia Object Server," ACM Performance Evaluation Review, vol. 25, no. 2, pp. 29-35, Sept. 1997.

[83] J.M. Perez, F. Garcia, J. Carretero, A. Calderon, L.M. Sanchez, "Data Allocation and Load Balancing for Heterogeneous Cluster Storage Systems," Proc. 3rd Int'l Symp. Cluster Computing and the Grid, 2003.

[84] Y. Huang, "Developing reliable applications on cluster systems,"Proc. Of the 15th Symposium on Reliable Distributed Systems, 1996.

[85] A.C Dusseau, R.H. Arpaci, and D.E. Culler, " Effective Distributed Scheduling of Parallel Workload," Proc. ACM SIGMETRICS CANT on Measuring and Modeling of Computer System, pp. 25-36, May 1996.

[86] X. Qin, H. Jiang, Y. Zhu, and D. Swanson, "Towards Load Balancing Support for WO-Intensive Parallel Jobs in a Cluster of Workstations," Proc. the Ah IEEE lnf'l Conference CIusfer Computing (CIuster 2003). pp.100-107, Dec. 2003.

[87] X. Qin, H. Jiang, Y. Zhu, D.R. Swanson," Improving the performance of I/O Intensive Applications on Clusters of WorkStations," cluster Computing: The Journal of Networks, Software Tools and Applications , Vol 9, no. 3.

[88] T. Xie, X. Qin, "Scheduling-Security-Critical-Real-Time Applications on Cluster," IEEE Transactions on Computers, vol. 55, no. 7.

[89] Y. Zhu, H. Jiang, X. Qin, and D. Swanson," A Case Study of Parallel I/O for Biological Sequence Analysis on Linux Clusters," Proc. the 5th IEEE Int'l conf. Cluster Computing, pp. 308-315, Dec. 2003.

[90] A. Caledron, " A fault tolerant MPI-IO implementation using the Expand Parallel File ystem"

[91] PPVFS

[92] Yifeng Zhu, Hong Jiang, Xiao Qin, Dan Feng, and David R. Swanson," Scheduling for improved write performance in a fault-tolerant parallel virtual file system" Technical report TR02-10-05, UNIVERSITY OF NEBRASKA-Lincoln, oct. 2002.

[93] Murali Vilayannur, Mahmut Kandemir, and Anand Sivasubramaniam, " Kernel-level caching for Optimizing I/O by exploiting inter-application data sharing," proceeding of 2002 IEEE International Conference on Cluster computing, Sept. 2002.

[94] A. W. Apon, P.D. Wolinski, and G.M Amerson, "Sensitivity of cluster file system access to I/O server selection," Cluster Computing and the Grid 2nd IEEE/ACM International Symposium CCGRID2002, 2002, pp. 183-192.

[95] Ligon, III, W.B., and Ross, R. B., " Server-Side Scheduling in Cluster Parallel I/O Systems," Calculateurs Paralleles Journal Special Issue on Parallel I/O for Cluster Computing, Oct, 2001.

[96] Wu, j, Wyckoff, p, Dhabaleswar Panda, " PVFS over InfiniBand: design and performance evaluation," Proc 2003 International Conference on Parallel Processing, pp. 125-132, 2003 ,

[97] D.B. Trizna, "Microwave and HF Mmulti-Frequency Radars for Dual-Use Coastal Remote Sensing Applications," Proc. MTS/IEEE OCEANS, pp. 532 - 537, Sept. 2005.

[98] J. Hawkins and M. Boden, M., "The Applicability of Recurrent Neural Networks for Biological Sequence Analysis," IEEE/ACM Trans. Computational Biology and Bioinformatics, vol. 2, no. 3, pp. 243 - 253, July-Sept. 2005.

[99] T. Johnson and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," VLDB, pp. 439-450, Los Altos, CA, 1995.