J. Parallel Distrib. Comput. 71 (2011) 288–301

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Quality of security adaptation in parallel disk systems

Mais Nijim [a,*], Ziliang Zong [d], Shu Yin [c], Kiranmai Bellam [b], Xiao Qin [c]

[a] Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, TX 78363-8202, United States
[b] Department of Computer Science, Prairie View A&M University, Prairie View, TX-77446-0519, United States
[c] Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849-5347, United States
[d] Department of Mathematics and Computer Science, South Dakota School of Mines and Technology, SD 57701, United States

ARTICLE INFO

Article history:
Received 22 January 2010
Received in revised form
13 August 2010
Accepted 25 August 2010
Available online 21 September 2010

Keywords:
Quality of security
Adaptive control
Parallel disk systems
Desired response time
Data partitioning
Security services

ABSTRACT

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 support data-intensive applications. Although a variety of parallel disk systems were developed, most existing disk systems lack a means to adaptively control the quality of security for dynamically changing workloads. 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 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.

1. Introduction

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 [3], digital libraries [20], and long-running simulations [21]. The high performance of data-intensive applications relies heavily on the performance of the underlying disk systems due to the rapidly widening gap between CPU and disk I/O speeds [14,13]. 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. The parallelism degree of a data request is the number of disks where the requested data resides [19].

Many data-intensive applications embrace a rich variety of security services to protect data residing in disk systems from talented intruders [23]. Further, it is often desirable for next-generation parallel disk systems to be highly flexible in order to support different quality of security at different times during a data-intensive application lifetime. This trend is especially true for data-intensive applications where the disk requests need to be completed within specified response times [4,11,12]. As such, parallel disk systems aim to achieve two major performance goals: high quality of security and high response time 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 paper, we propose an *adaptive quality of security control framework for parallel disk systems*, or ASPAD for short. 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. 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

* Corresponding author.
E-mail addresses: maisnijim@gmail.com (M. Nijim), xqin@auburn.edu (X. Qin).

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.08.014

most appropriate security services for disk requests in such a way to improve the security of parallel disk systems while making the best effort to guarantee completions of a vast majority of the requests before the desired response times.

The rest of the paper is organized as follows. In Section 2, we summarize related work. Section 3 details a model of disk requests and the new architecture of storage systems. In Section 4, we propose the adaptive write strategy for security-aware storage systems. We build an analytical model in Section 5. Section 6 presents extensive experimental results on a wide range of workload conditions. Finally, Section 7 concludes the paper with future directions.

2. Related work

Disk I/O has become a performance bottleneck for data-intensive applications due to the widening gap between processor speed and disk access speed [15]. To help alleviate the problem of disk I/O bottlenecks, a large body of work has been done on parallel disk systems. Kallahalla and Varman designed an online buffer management and scheduling algorithm to improve performance of parallel disks [6]. Scheuermann et al. addressed the problem of making use of striping and load balancing to tune the performance of parallel disk systems [19]. Rajasekaran and Jin developed a practical model for parallel disk systems [16]. Kotz and Ellis proposed several writeback policies used in a parallel file system implementation [7]. Our approach is fundamentally different from the aforementioned techniques in that we focus on quality of security adaptation for parallel disk systems. Further, our strategy is orthogonal to the existing techniques in the sense that our scheme can be readily integrated into existing parallel disk systems to substantially improve the security of the systems.

Many data-intensive applications embrace a rich variety of security services to protect data residing in disk systems from talented intruders [23]. 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 [18]. To protect data in untrusted storage systems, researchers designed and implemented cryptographic systems in which data is stored in encrypted form [2,5]. Several key distribution schemes were proposed in the SFS [9] and SNAD systems [10]. Data-intensive applications rely on stored and accessed data; supporting the availability, integrity, and confidentiality of these data is crucial. While et al. developed a survivable storage system which guarantees that the data is persist, continuously accessible, cannot be destroyed, and is kept confidential [22]. Leung and Miller proposed a scalable and efficient protocol for security in high-performance storage systems, which increases the performance without sacrificing security primitives [8]. Miller et al. developed a scheme to secure network-attached storage systems against any type of attack [10]. They used a cryptography scheme to hide the data from unauthorized users.

In our previous work, we developed an array of security-aware scheduling algorithms for embedded real-time systems [28], clusters [26,24,27], Grids [23], and distributed systems [25]. Very recently, we investigated an adaptive write strategy for local disk systems [11,12]. Since our previous approaches were designed for either computing resources or single-disk systems, these schemes are inadequate for quality of security adaptation in parallel disk systems.

3. Architecture and disk requests

3.1. System architecture

In this study, 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 Fig. 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 (NASs) and storage area networks (SANs).

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 different 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 middleware 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. The 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 over which the security service has been used. The data-partitioning mechanism is geared to divide a large amount of data into fixed-size 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 types [19]. The process of data partitioning is detailed in Section 4.1. 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 4.2 provides a means of estimating the response times of disk requests submitted to a parallel disk system. The security quality controller is developed to make a decision to select 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 trade-off 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 partitioning gives rise to a balanced load across all the disks in the system. In contrast, dynamic data partitioning is performed for write requests to alleviate an imbalanced load among the disks due to skewed access frequencies. The security quality controller selects the most appropriate security service to protect the 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, the disk requests issued by clients are write requests unless otherwise specified. After the process of data partitioning and security service controlling is accomplished for a write request, 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.

3.2. Quality of security

Recall that a security-aware parallel disk system encompasses a security service middleware providing an array of security services with different quality of security. The quality of security for each

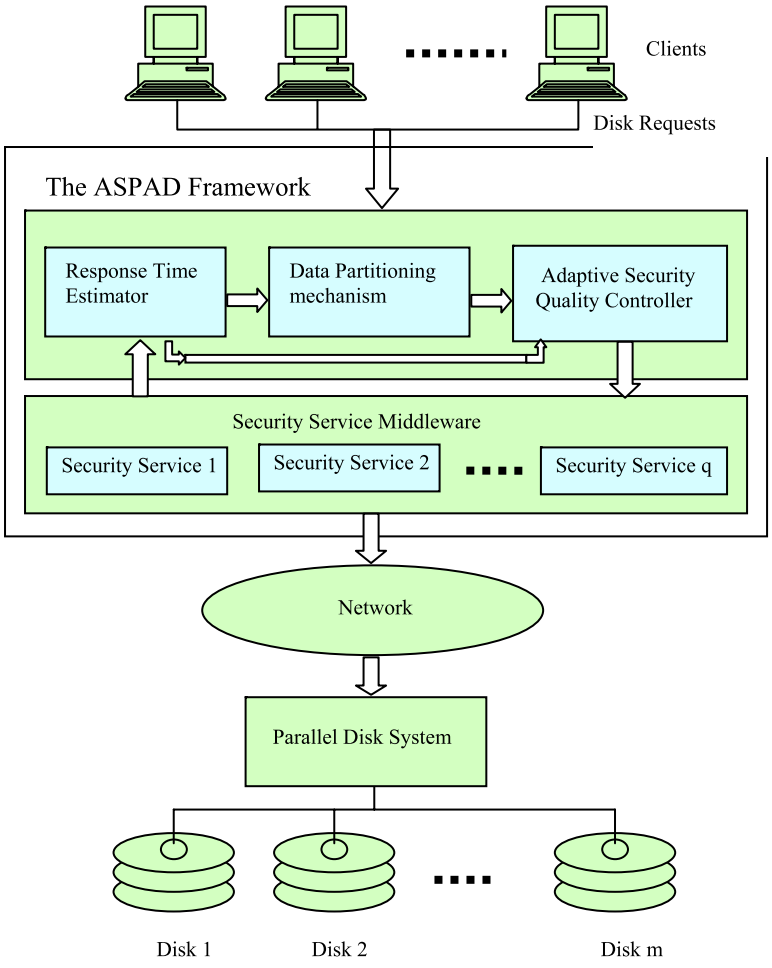


Fig. 1. Architecture of a security-aware parallel disk system.

security service is measured by security level ranging from 0.1 to 1.0 [11,12,23]. 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 the range between 0.1 and 1.0. A higher security level indicates a better quality of security of the service. More generally speaking, security services fall into three categories: confidentiality, integrity, and availability. In our quality of security model, we address the confidentiality issues by employing nine cryptographic algorithms in our framework [11,12]. Note that the quality of security model can be easily extended to incorporate the other two security service categories.

Our quality of security model also readily applies to any disk system where there are only a few cryptographic algorithms implemented, because each cryptographic mechanism can provide different 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, networks, and parallel disk systems are always available by 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 part of the security overhead.

Security overheads incurred by the cryptographic algorithms depend on the size of the data to be encrypted and the performance of the algorithms, each of which is assigned a security level. Let σ_i denote the security level of the cryptographic algorithm used to encrypt the data for disk request r_i, and d_i the size of the 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 σ_i to the performance (measured in kB/ms) of the corresponding confidentiality service [11,12].

$$T_{\text{security}}(\sigma_i, d_i) = \frac{d_i}{P(\sigma_i)}. \tag{1}$$

3.3. Disk requests with security and performance requirements

In this study, we consider data-intensive applications with both security and performance constraints, meaning that the disk requests issued by the applications to a parallel disk system impose both security and performance requirements. It should be noted that the disk requests' security and timing requirements, defined as level of security services and desired response times, respectively, are derived from the corresponding data-intensive applications. The 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 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 the security level that the 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 paper, we investigate the desired response time, which is a

specific performance requirement; the desired response time of request r_i is represented by t_i. We denote the 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 the parallelism degree for each request in a way to improve the throughput of the system. Given the parallelism degrees of requests, the quality of security for the requests can be tuned in a judicious manner by the security service controller.

The first step towards improving the quality of security is to quantitatively measure the 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.

$$S(r_i) = \sum_{j=1}^{p_i} \sigma_{ij}, \quad \sigma_{ij} \geq s_i \text{ and } p_i \leq m, \tag{2}$$

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 the confidentiality service chosen for the jth stripe unit of r_i. It is intuitive to argue that (1) the parallelism degree p_i cannot 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 $R = \{r_1, r_2, \ldots, r_n\}$, we can obtain the security benefits experienced by the requests. Thus, we have

$$S(R) = \sum_{i=1}^{n} S(r_i). \tag{3}$$

Now we obtain the following nonlinear optimization problem formulation to compute the optimal security benefit of the parallel disk system:

$$\begin{aligned} \text{maximize} \quad & S(R) = \sum_{i=1}^{n} \sum_{j=1}^{p_i} \sigma_{ij}, \\ \text{subject to} \quad & \text{(a)}\ \forall 1 \leq i \leq n : \max_{1 \leq j \leq p_i} \{\theta_{ij}\} \leq t_i, \\ & \text{(b)}\ \sigma_{ij} \geq s_i \quad \text{and} \quad p_i \leq m, \end{aligned} \tag{4}$$

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

4. 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.

4.1. Data partitioning

One of the major components in the proposed framework (see Section 4.3) is the method of data partitioning that determines the optimal parallelism degrees for the 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. 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 we proceed to the analysis of optimal parallelism degrees, we formally derive the expected value of disk service time $T_{\text{disk}}(d_i, p_i)$ for request r_i. Thus, the expected disk service time can be computed as

$$E(T_{\text{disk}}(d_i, p_i)) = E(T_{\text{seek}}(p_i)) + E(T_{\text{rot}}(p_i)) + E(T_{\text{trans}}(d_i, p_i)), \tag{5}$$

where $E(T_{\text{seek}}(p_i))$, $E(T_{\text{rot}}(p_i))$, and $E(T_{\text{trans}}(d_i, p_i))$ are the expected values of seek time, rotation time, and transfer time, respectively. It was shown in [19] that the expected seek time can be approximated as below, where C is the number of cylinders on a disk, and a and b are two disk-type-independent constants, whereas e and f are disk-type-dependent constants.

$$E(T_{\text{seek}}(p_i)) = eC(1 - a - b\ln(p_i)) + f. \tag{6}$$

The expected value of rotation time can be expressed as Eq. (7), where T_{ROT} is the rotation time of a disk. Note that a similar expression can be found in [19].

$$E(T_{\text{rot}}(p_i)) = \frac{p_i}{p_i + 1} \cdot T_{\text{ROT}}. \tag{7}$$

The expected transfer time can be approximated by Eq. (8), where B_{disk} is the disk bandwidth.

$$E(T_{\text{trans}}(d_i, p_i)) = \frac{d_i}{p_i} \cdot \frac{1}{B_{\text{disk}}}. \tag{8}$$

Substituting Eqs. (6)–(8) into Eq. (5), we obtain the expected value of the disk service time as

$$\begin{aligned} E(T_{\text{disk}}(d_i, p_i)) = {} & eC(1 - a - b\ln(p_i)) + f + \frac{p_i}{p_i + 1} \cdot T_{\text{ROT}} \\ & + \frac{d_i}{p_i} \cdot \frac{1}{B_{\text{disk}}}. \end{aligned} \tag{9}$$

Now we are positioned to calculate the optimal parallelism degree of request r_i by determining the minimum of the function $E(T_{\text{disk}}(d_i, p_i))$. Thus, we can obtain the optimal value of p_i by solving Eq. (10).

$$\begin{aligned} \frac{\mathrm{d}E(T_{\text{disk}}(d_i, p_i))}{\mathrm{d}E(p_i)} &= \frac{T_{\text{ROT}}}{p_i + 1} - \frac{p_i \cdot T_{\text{ROT}}}{(p_i + 1)^2} - \frac{eCb}{p_i} - \frac{d_i}{p_i^2} \cdot \frac{1}{B_{\text{disk}}} \\ &= 0. \end{aligned} \tag{10}$$

The parallelism degree determined by Eq. (10) cannot exceed m, which is the number of disks in the system. Therefore, the optimal parallelism degree is given by $\min(p_i, m)$.

4.2. Estimating response times

To adaptively adjust the security levels for the 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 the corresponding disk I/O operations. Given a newly issued request r, the response time of r is estimated by Eq. (11).

$$T(r, p, \sigma) = T_{\text{queue}} + T_{\text{partition}} + \max_{i=1}^{p} \{T^i_{\text{proc}}(r, p, \sigma_i)\}, \tag{11}$$

where p is the parallelism degree determined by the data-partitioning mechanism (see Eq. (11)), $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_p)$ is the request's vector of security levels for p stripe units, T_{queue} is the queuing delay at the client side, $T_{\text{partition}}$ is the time spent in data partitioning, 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

$$\begin{aligned} T^i_{\text{proc}}(r, p, \sigma_i) = {} & T^i_{\text{security}}(r, p, \sigma_i) + T^i_{\text{network}}(r, p, \sigma_i) \\ & + T^i_{\text{disk}}(r, p, \sigma_i), \end{aligned} \tag{12}$$

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** $\sigma_{ij} < 0.9$ **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} cannot 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**
19. **end for**

Fig. 2. The processing algorithm for the ASPAD framework in a parallel disk.

where T^i_{security}, T^i_{network}, and T^i_{disk} are the delays at the security mechanism, network subsystems, 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 $T^i_{\text{security}}(r, p, \sigma_i)$ from Eq. (1) as

$$T^i_{\text{security}}(r, p, \sigma_i) = T^i_{\text{security}}\left(\sigma_i, \frac{d}{p}\right) = \frac{d}{p \cdot P(\sigma_i)}, \tag{13}$$

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 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 $T^i_{\text{network}}(r, p, \sigma_i)$ can be written as

$$T^i_{\text{network}}(r, p, \sigma_i) = \frac{i \cdot \frac{d}{p} + \sum_{j=1}^{k} d_j}{B_{\text{network}}}, \tag{14}$$

where d_j is the data size of the jth stripe unit in the network queue, and B_{network} is the effective network bandwidth. The optimal parallelism degree is determined by the data-partitioning mechanism (see Eq. (10) in Section 4.1).

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

$$T^i_{\text{disk}}(r, p, \sigma_i) = T_{\text{disk},j}(d/p) + \sum_{l=1}^{k} T_{\text{disk},j}(d_l), \tag{15}$$

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

$$T_{\text{disk},j}(d) = T_{\text{seek}} + T_{\text{rot}} + \frac{d}{B_{\text{disk}}}, \tag{16}$$

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

4.3. 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 guaranteeing the desired response time of the request. Specifically, the algorithm is carried out in three phases: dynamic data partitioning (see Section 4.1), response time estimation (see Section 4.2), and adaptive security quality control. To heuristically improve the security of the disk systems, ASPAD endeavors 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. (15)). During the second phase of the algorithm, the response time of each stripe unit is estimated (see Eqs. (11) and (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. (11) does not exceed the request's desired response time. The complete algorithm for quality of security control is outlined in Fig. 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 partitioning of each request in the queue, ASPAD initializes the security levels of all stripe units of request r_i to the minimal levels s_i (see Step 6). In an effort to gradually increase the security

levels of the 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* 1.0.
(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*

$$T_j(r_i, p_i, \sigma_i) = T_{\text{queue}} + T_{\text{partition}} + T^{ij}_{\text{proc}}(r_i, p_i, \sigma_{ij}).$$

Steps 10–11 are repeatedly performed to optimize the security levels until a request's desired response time cannot be guaranteed (see Step 12) or the security levels are approaching 1.0. Consequently, the ASPAD framework 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 and 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 the security levels of a write request is $O(10p) = O(p)$, where 10 is the number of security levels. There are n disk requests in the waiting queue and, therefore, the time complexity of improving security of all the requests is $O(n)O(p) = O(np)$. □

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 the arrival rates of the disk requests are relatively low, because the number of waiting disk requests in these cases is small.

4.4. Optimality of the security quality controller in the ASPAD framework

Before building an analytical model to evaluate the performance of ASPAD, we prove the optimality of the ASPAD framework. Theorems 2–4 are of help in the proof of Theorem 5, which signifies that the ASPAD framework optimizes the quality of security for the 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* σ_{ij} *of the jth stripe unit is independent of the optimization of security level* σ_{ik} *of the kth stripe unit.*

Proof. The algorithm outlined in Fig. 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. 2). The estimated response time of r_i in turn relies on the queueing delay T_{queue} at the client side, time $T_{\text{partition}}$ spent in the data-partitioning mechanism, and the maximal system processing delay T_{proc} experienced by all the stripe units of the request (see Eq. (11) in Section 4.2). Since the two stripe units have the same values of T_{queue} and $T_{\text{partition}}$, we are concerned with the system processing delays with respect to the ith and jth stripe units. Eq. (12) shows that the system processing delay T^j_{proc} 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 unit's delay times in the 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*

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

Proof. The proof of this theorem is derived from Theorem 2. Since Theorem 2 proves that the optimization of security level σ_{ij} of the jth stripe unit is independent of the optimization of security levels of the other stripe units of disk request r_i, the value of can be maximized without adverse effects upon the security level 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 units of disk request r_i is maximized by the security quality controller, i.e., $\sum_{j=1}^{p_i} \sigma_{ij}$ 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, that of *non-secure response time*.

Definition 1. The non-secure response time $T^{ij}_{\text{non-secure}}$ 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

$$T^{ij}_{\text{non-secure}} = T_{\text{queue}} + T_{\text{partition}} + T^{ij}_{\text{security}} + T^{ij}_{\text{network}} + T^{ij}_{\text{disk}}.$$

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., that σ_{ij} cannot be further increased by the processing algorithm (see Fig. 1). The first phase in the algorithm attempts 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. 1). Consequently, the non-secure response time of any stripe unit is minimized. Thus, the non-secure response time $T_{\text{queue}} + T_{\text{partition}} + T^{ij}_{\text{network}} + T^{ij}_{\text{disk}}$ of the jth stripe unit is minimized. Since $T_{\text{queue}} + T_{\text{partition}}$ cannot be reduced by the

ASPAD framework, the processing algorithm minimizes the value of $T^{ij}_{\text{network}} + T^{ij}_{\text{disk}}$. Step 8 ensures that the inequality

$$T_{\text{queue}} + T_{\text{partition}} + T^{ij}_{\text{network}} + T^{ij}_{\text{disk}} + T^{ij}_{\text{security}} \le t_i$$

holds and, therefore, we have

$$T^{ij}_{\text{security}} \le t_i - (T_{\text{queue}} + T_{\text{partition}} + T^{ij}_{\text{network}} + T^{ij}_{\text{disk}}).$$

We proved that $T_{\text{queue}} + T_{\text{partition}} + T^{ij}_{\text{network}} + T^{ij}_{\text{disk}}$ 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. □

Now we are in a position to prove the optimality of the quality of security controller, which is considered as the main part of the ASPAD framework (see Fig. 1).

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 that if the security levels of all the stripe units in 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. □

5. 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 the disk requests by the security quality controller in the ASPAD framework.

5.1. Satisfied ratio

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., $\Pr(T(r_i, p_i, \sigma_i) \le t_i)$, 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 times. In doing so, the disk requests with the earliest desired response times are given the highest priority and responded to by the parallel disk system first. To simplify the analysis that follows, we assume that the processing times of different disk requests are statistically independent. Suppose that in the queue there are n pending disk requests indexed by their priorities, and that the desired response time of r_i is smaller than that of r_j if $i < j$, i.e.,

$$\forall 1 \le i, \quad j \le n : i < j \Rightarrow t_i < t_i.$$

Eq. (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 U_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 V_y be the probability that the total required processing time of disk requests with higher priorities is y, i.e.,

$$v_y = \Pr\left(\sum_{j=1}^{i-1} T(r_i, p_i, \sigma_i) = y\right).$$

The probability that r_i is unable to be finished within the desired response time t_i is computed as follows:

$$\begin{aligned}
&\Pr(T(r_i, p_i, \sigma_i) > t_i) \\
&= \Pr\left(T(r_i, p_i, \sigma_i) = 1 \middle| \sum_{j=1}^{i-1} T(r_i, p_i, \sigma_i) \ge t_i\right) \\
&\quad \vdots \\
&\quad + \Pr\left(T(r_i, p_i, \sigma_i) = t_i + 1 \middle| \sum_{j=1}^{i-1} T(r_i, p_i, \sigma_i) \ge 0\right) \\
&= u_1 \sum_{y=t_i}^{\infty} v_y + u_2 \sum_{y=t_i-1}^{\infty} v_y + \cdots + u_{t_i} \sum_{y=1}^{\infty} v_y + u_{t_i+1} \sum_{y=0}^{\infty} v_y \\
&= \sum_{x=1}^{t_i+1} \left(u_x \sum_{y=t_i+1-x}^{\infty} v_y\right), \qquad (17)
\end{aligned}$$

where the second line in 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 units to complete the disk requests with higher priorities.

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

$$\begin{aligned}
\Pr(T(r_i, p_i, \sigma_i) \le t_i) &= 1 - \Pr(T(r_i, p_i, \sigma_i) > t_i) \\
&= 1 - \sum_{x=1}^{t_i+1} \left(u_x \sum_{y=t_i+1-x}^{\infty} v_y\right). \qquad (18)
\end{aligned}$$

In what follows, we derive two important probabilities u_x (see Eq. (26)) and v_y (see Eq. (30)) used in Eqs. (17) and (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 units to complete. Eq. (11) in Section 4.2 shows a way of computing this, and thus, we have

$$T(r_i, p_i, \sigma_i) = T^i_{\text{queue}} + T^i_{\text{partition}} + \max_{k=1}^{p}\{T^{ik}_{\text{proc}}(r_i, p_i, \sigma_{ik})\} = x, \qquad (19)$$

where T^i_{queue} is the queuing delay of disk request r_i, $T^i_{\text{partition}}$ 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 queuing 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

$$\begin{aligned}
T^i_{\text{queue}} &= \sum_{j=1}^{i-1} (T^j_{\text{partition}} + T^j_{\text{security}}) \\
&= (i-1) T_{\text{partition}} + \sum_{j=1}^{i-1} T^j_{\text{security}}, \qquad (20)
\end{aligned}$$

where the partitioning overhead of all the submitted requests is assumed to be a constant, $T_{\text{partition}}$. Since the security overhead of disk request r_j is the summation of the security overheads for all the stripe units of r_j, as per Eq. (13) we have the following equation to calculate T^i_{security} on the right-hand side of Eq. (20).

$$T^j_{\text{security}} = \sum_{k=1}^{p_j} \frac{d}{p_j P(\sigma_{jk})}. \qquad (21)$$

Therefore, we can obtain the following equation from Eqs. (19)–(21).

$$\max_{k=1}^{p_i}\{T^{ik}_{\text{proc}}(r_i, p_i, \sigma_{ik})\} = x - i \cdot T_{\text{partition}} - \sum_{j=1}^{i-1} \sum_{k=1}^{p_j} \frac{d}{p_j \cdot P(\sigma_{jk})}. \qquad (22)$$

Since the probability

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

is equal to the probability $u_x = \Pr(T(r_i,p_i,\sigma_i) = x)$, u_x used in Eqs. (17) and (18) can be written as

$$u_x=\Pr\left(\max_{k=1}^{p_i}\{T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})\}=x-i\cdot T_{\text{partition}}-\sum_{j=1}^{i-1}\sum_{k=1}^{p_j}\frac{d}{p_j\cdot P(\sigma_{jk})}\right). \quad (23)$$

Let $\Pr\left(\max_{k=1}^{p_i}\{T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})\}=a\right)$ be the probability that $\max_{k=1}^{p_i}\{T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})\}$ is equal to a. $\Pr\left(\max_{k=1}^{p_i}\{T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})\}=a\right)$ can be written as

$$\begin{aligned}
&\Pr\left(\max_{k=1}^{p_i}\{T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})\}=a\right)\\
&=\Pr(T_{\text{proc}}^{i1}(r_i,p_i,\sigma_{i1})=a)\cdot\prod_{k=2}^{p_i}\Pr(T_{\text{proc}}^{i1}(r_i,p_i,\sigma_{i1})<a)\\
&\quad+\Pr(T_{\text{proc}}^{i2}(r_i,p_i,\sigma_{i2})=a)\\
&\quad\times\prod_{k=1,k\neq 2}^{p_i}\Pr(T_{\text{proc}}^{i2}(r_i,p_i,\sigma_{i2})<a)\\
&\quad\vdots\\
&\quad+\Pr(T_{\text{proc}}^{ip_i}(r_i,p_i,\sigma_{ip_i})=a)\\
&\quad\times\prod_{k=1}^{p_i-1}\Pr(T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})<a). \quad (24)
\end{aligned}$$

We denote $\Pr(T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})=a)$ by $c_{ik}(a)$, and $\Pr(T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})<a)$ by $c'_{ik}(a)$. Then, Eq. (24) can be expressed as follows:

$$\begin{aligned}
&\Pr\left(\max_{k=1}^{p_i}\{T_{\text{proc}}^{ik}(r_i,p_i,\sigma_{ik})\}=a\right)\\
&=\sum_{k=1}^{p_i}\left(c_{ik}(a)\cdot\prod_{i=1,i\neq k}^{p_i}c'_{ik}(a)\right). \quad (25)
\end{aligned}$$

Consequently, the probability u_x can be derived from Eqs. (23) and (25) by substituting $x-i\cdot T_{\text{partition}}-\sum_{j=1}^{i-1}\sum_{k=1}^{p_j}\frac{d}{p_j\cdot P(\sigma_{jk})}$ for variable a in Eq. (25).

$$\begin{aligned}
u_x=\sum_{k=1}^{p_j}\Bigg(&c_{ik}\left(x-i\cdot T_{\text{partition}}-\sum_{j=1}^{i-1}\sum_{k=1}^{p_j}\frac{d}{p_j\cdot P(\sigma_{jk})}\right)\\
&\times\prod_{i=1,i\neq k}^{p_j}c'_{ik}\left(x-i\cdot T_{\text{partition}}-\sum_{j=1}^{i-1}\sum_{k=1}^{p_j}\frac{d}{p_j\cdot P(\sigma_{jk})}\right)\Bigg). \quad (26)
\end{aligned}$$

Now we derive the second important probability $v_y = \Pr\left(\sum_{j=1}^{i-1} T(r_j,p_j,\sigma_j) = y\right)$ from u_x computed by Eq. (26). To facilitate the derivation of v_y, we need the following definition.

Definition 2. Φ_y^{i-1} is a set of vectors $(y_1, y_2, \ldots, y_{i-1})$ in which the summation of all the elements in each vector equals to y. Formally, the set of vectors is defined as follows:

$$\Phi_y^{i-1}=\left\{(y_1,y_2,\ldots,y_{i-1}),\ \text{where}\ \sum_{j=1}^{i-1}y_j=y\right\}.$$

The probability v_y is derived from u_x and set Φ_y^{i-1} as follows:

$$\begin{aligned}
v_y=\sum_{y_1,\ldots,y_{i-1}\in\Phi_y^{i-1}}&\{\Pr(T(r_1,p_1,\sigma_1)=y_1)\Pr(T(r_2,p_2,\sigma_2)=y_2)\\
&\times\cdots\Pr(T(r_{i-1},p_{i-1},\sigma_{i-1})=y_{i-1})\}. \quad (27)
\end{aligned}$$

5.2. Quality of security

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

$$\begin{aligned}
\Pr(\sigma_{ij}=z)=\Pr(d_i=d_{\min})\cdot&\sum_{k=t_{\min}}^{t_{\max}}\left\{\Pr(t_i=k)\right.\\
&\left.\times\Pr\left(\sum_{l=1}^{i-1}T(r_l,p_l,\sigma_l)=k-\overline{T}\left(\frac{d_{\min}}{p_i},z\right)\right)\right\}\\
&\vdots\\
+\Pr(d_i=a)\cdot&\sum_{k=t_{\min}}^{t_{\max}}\left\{\Pr(t_i=k)\right.\\
&\left.\times\Pr\left(\sum_{l=1}^{i-1}T(r_l,p_l,\sigma_l)=k-\overline{T}\left(\frac{a}{p_i},z\right)\right)\right\}\\
&\vdots\\
+\Pr(d_i=d_{\max})\cdot&\sum_{k=t_{\min}}^{t_{\max}}\left\{\Pr(t_i=k)\right.\\
&\left.\times\Pr\left(\sum_{l=1}^{i-1}T(r_l,p_l,\sigma_l)=k-\overline{T}\left(\frac{d_{\max}}{p_i},z\right)\right)\right\}\\
=\sum_{a=d_{d\min}}^{d_{\max}}&\left\{\Pr(d_i=a)\cdot\sum_{k=t_{\min}}^{t_{\max}}\left\{\Pr(t_i=k)\right.\right.\\
&\left.\left.\times\Pr\left(\sum_{l=1}^{i-1}T(r_l,p_l,\sigma_l)=k-\overline{T}\left(\frac{a}{p_i},z\right)\right)\right\}\right\}, \quad (28)
\end{aligned}$$

where $\overline{T}\left(\frac{a}{p_i},z\right)$ is the processing time of r_i if the data size is a and the security level is set to z, the 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 known *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 is equal to j. Thus, the probability $\Pr(\sigma_{ij} = z)$ can be derived from f_a, w_j and v_y (see Eq. (27)).

$$\Pr(\sigma_{ij}=z)=\sum_{a=d_{d\min}}^{d_{\max}}\left\{f_a\cdot\sum_{k=t_{\min}}^{t_{\max}}\left[w_k\cdot v_{k-\overline{T}\left(\frac{a}{p_i},z\right)}\right]\right\}, \quad (29)$$

where

$$v_{k-\overline{T}\left(\frac{a}{p_i},z\right)}=\Pr\left(\sum_{l=1}^{i-1}T(r_l,p_l,\sigma_l)=k-\overline{T}\left(\frac{a}{p_i},z\right)\right).$$

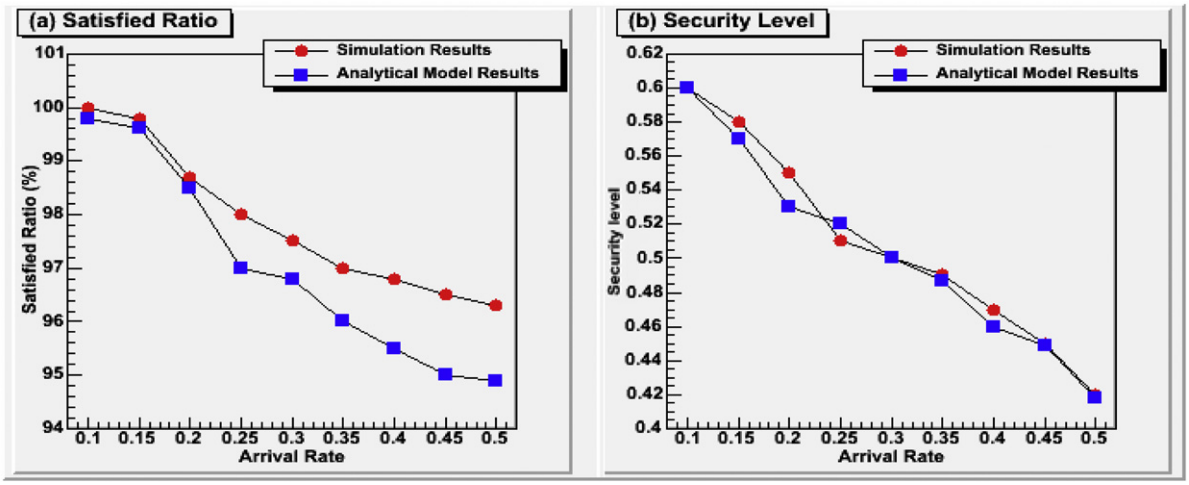


Fig. 3. Results from the analytical model.

The expected security level experienced by disk requests with security requirements can be directly derived from Eq. (29). Hence, the expected security level is given by

$$E(\sigma_i) = p_i \cdot \sum_{k=1}^{9} \left(\frac{k}{10} \cdot \Pr\left(\sigma_{ij} = \frac{k}{10} \right) \right). \tag{30}$$

To validate our analytical mode, we compare the value of the satisfied ratio and the security level that were calculated from the analytical model equations with those that were calculated in the simulator. Fig. 3(a) and (b) reveal that the satisfied ratio and the security level that were calculated using the analytical model are very similar to those that were calculated using the security-aware parallel disk system simulator.

6. 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 1 summarizes the important system parameters used to resemble real-world disks such as Seagate Barracuda 36ES2 [17].

In our ASPAD framework, we implemented a data-partitioning algorithm to optimize the 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 that assigns the minimum security requirements for the requests; this case is considered as a security-aware parallel disk system without employing ASPAD. We will then study the effects of varying the 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 the disk requests. Finally, we also analyze the 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 to be finished before their desired response time.
(2) *Security level* is the sum of security level values of all disk requests issued to the parallel disk systems.
(3) *Overall performance* is a performance metric measured by the product of the satisfied ratio and the security level.

Table 1
Worst-case parameters of disks in the simulated parallel disk system and workload conditions.

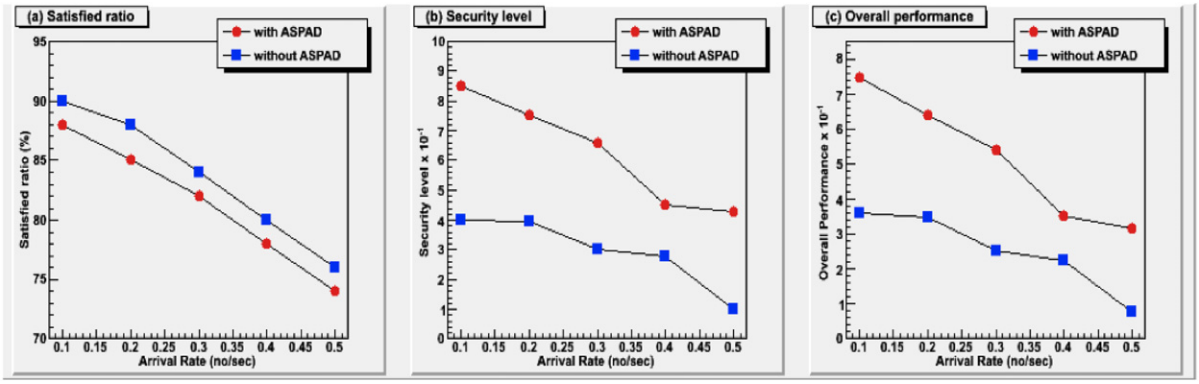
Parameter	Value
Number of disks	2, 4, 6, 8, 10, 12, 14, 16
Capacity of one disk	18.4 GB
Average seek time	11.36 ms
Average rotation time	8.37 ms
Blocks revolution per minute	7200 RPM
Transfer rate per disk	30 MB/s
Arrival rate	0.1, 0.2, 0.3, 0.4, 0.5 No/second
Data size	100 kB, 10 MB, 100 MB
Security range S_{max}	0.5, 0.55, 0.6, 0.65, . . ., 0.85, 0.9

6.1. Impacts of arrival rate

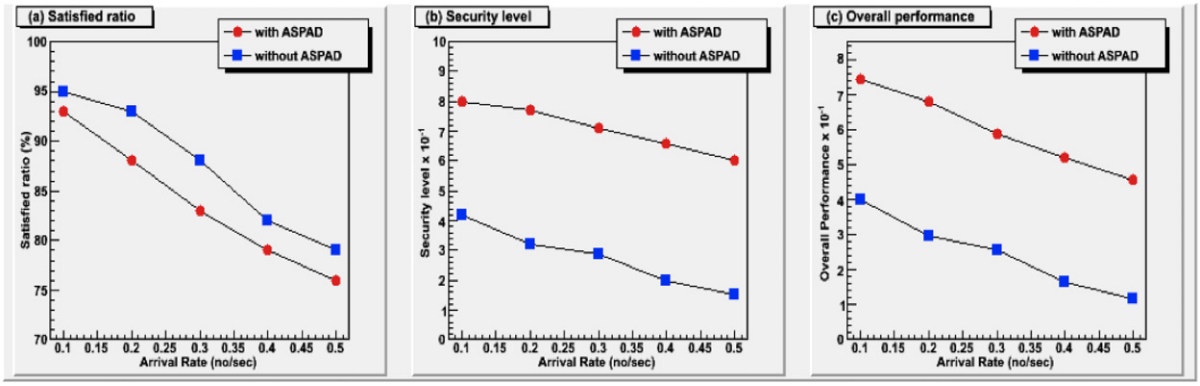
This experiment aims at comparing the ASPAD strategy with a baseline scheme that makes no use of ASPAD. With different settings of parallelism degrees and data sizes, we study the impacts of varying the arrival rates on system performance. To achieve this goal, we increased the arrival rate of I/O requests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

Fig. 4 plots the satisfied ratios, security levels, and overall performance of the parallel disk systems with and without ASPAD. Fig. 4(a)–(a), (b)–(a) and (c)–(a) reveal that the ASPAD strategy yields satisfied ratios that are very close to those of the parallel disk system making no use of ASPAD. This is mainly because ASPAD endeavors to guarantee the timing constraints of the disk requests while maximizing the security of the parallel disk system. Fig. 4(a)–(b), (b)–(b), and (c)–(b) 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 the 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 both systems decrease. This result is not surprising because high arrival rates lead to high workload, forcing 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.

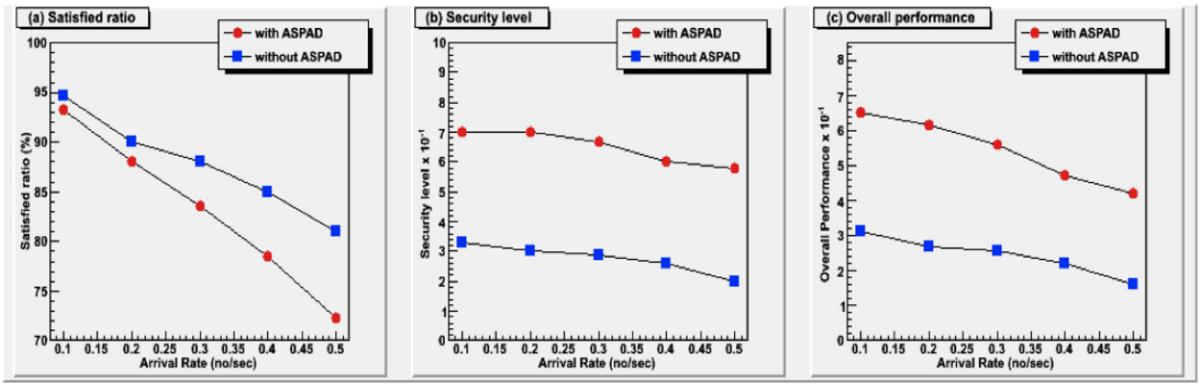
Interestingly, ASPAD always achieves higher security levels compared with the parallel disks without ASPAD. It is worth noting that the security improvement comes at the cost of satisfied ratios (see Fig. 4(a)–(a), (b)–(a) and (c)–(a)), because the satisfied ratios of



(a) The data size is 100 kB and $P = 3$.



(b) The data size is 10 MB and $P = 8$.



(c) The data size is 100 MB and $P = 16$.

Fig. 4. Impact of arrival rate on the performance of security-aware parallel disk systems.

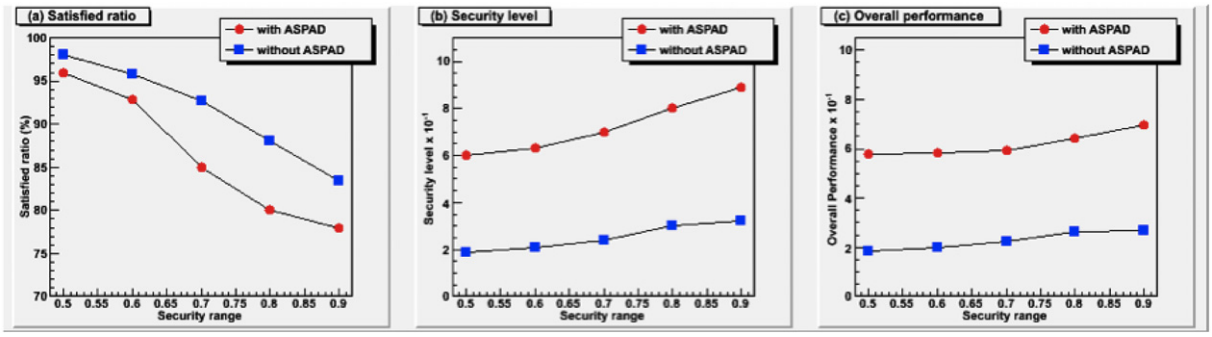
ASPAD are slightly reduced due to relatively high security overhead caused by the ASPAD strategy. Fig. 4(a)–(c), (b)–(c), and (c)–(c) reveal that ASPAD substantially boosts the overall performance. The reasons for the expected overall performance improvement are twofold. First, ASPAD adaptively enhances the security levels for disk I/O requests. Second, the performance gains in security level can eventually offset the extra security overhead.

6.2. Impacts of security requirements

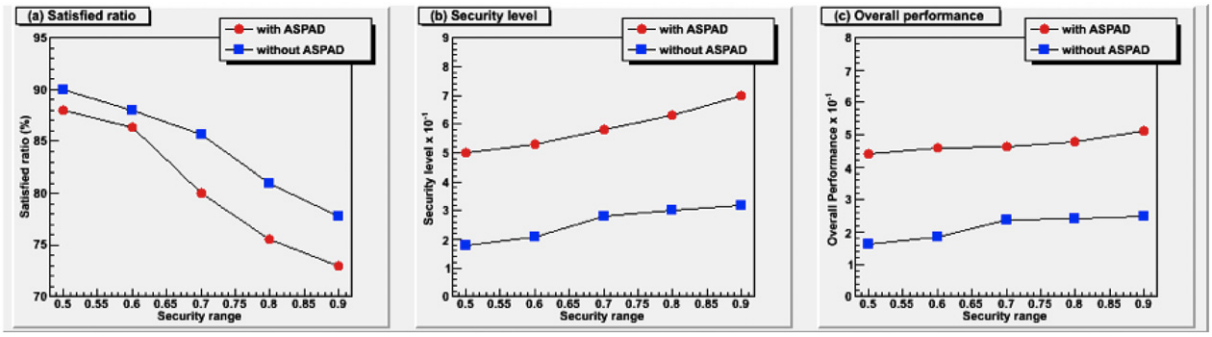
In this group of experiments, we investigate the performance impacts of security requirements imposed by parallel disk requests. As mentioned earlier, the security requirement of each disk request is characterized by a security range varying from $s_{\min}$ to $s_{\max}$. We varied $s_{\max}$ from 0.5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

We observed from Fig. 5(a)–(a), (b)–(a), and (c)–(a) that increasing the maximal security level of the security range leads to decreasing values of the satisfied ratio of the two disk systems examined. This is because, when the security level requirements of the disk requests are high, the parallel disk systems need to fulfill the security requirements with high security overheads, which in turn reduces the satisfied ratios.

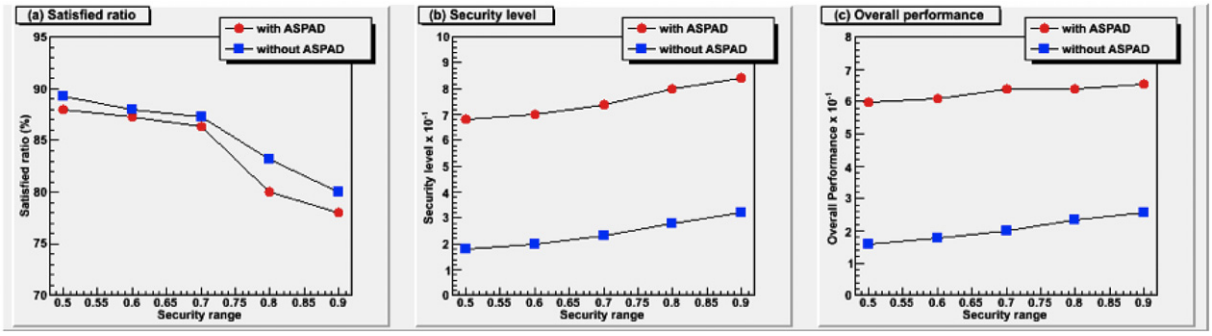
Again, the satisfied ratios of ASPAD are reasonably close to those of the alternative strategy. It is observed from Fig. 5(a)–(b), (b)–(b), and (c)–(b) that, when $s_{\max}$ increases, the security levels of the two evaluated schemes gradually increase. This result implies that the security levels of the two schemes heavily rely on the security requirements of the disk requests. Fig. 5(a)–(c), (b)–(c), and (c)–(c) illustrate that the overall performance of the two systems improves with the increasing values of $s_{\max}$, because the



(a) Arrival rate = 0.5 No/second, data size = 100 kB, $P = 3$.



(b) Arrival rate = 0.5 No/second, data size = 10 = MB, $P = 8$.



(c) Arrival rate = 0.5 No/second, data size = 100 MB, $P = 16$.

Fig. 5. Impact of security requirements on the performance of security-aware parallel disk systems.

overall performance is more sensitive to the security level than to the satisfied ratio.

6.3. 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 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. 6(a) 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 (hereinafter referred to as the striping width) and, thus, increasing the striping width has a strong likelihood of reducing the response times of the requests and enhancing the throughput of parallel disk systems.

It is intriguing to observe from Fig. 6(b) that 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. 6(c), 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. 6(a)). 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 the security levels of disk requests in a judicious manner.

7. Improved ASPAD

The original ASPAD algorithm suffers from a small problem. The original ASPAD algorithm inserts the requests into the queue

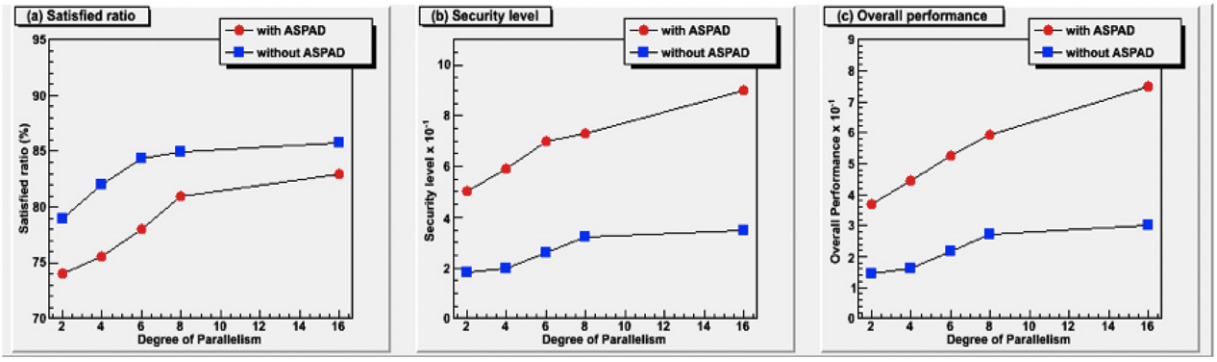
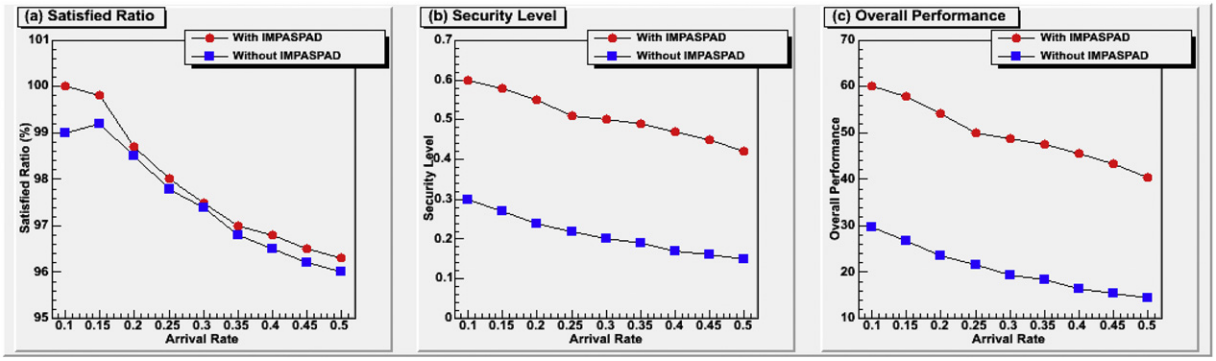
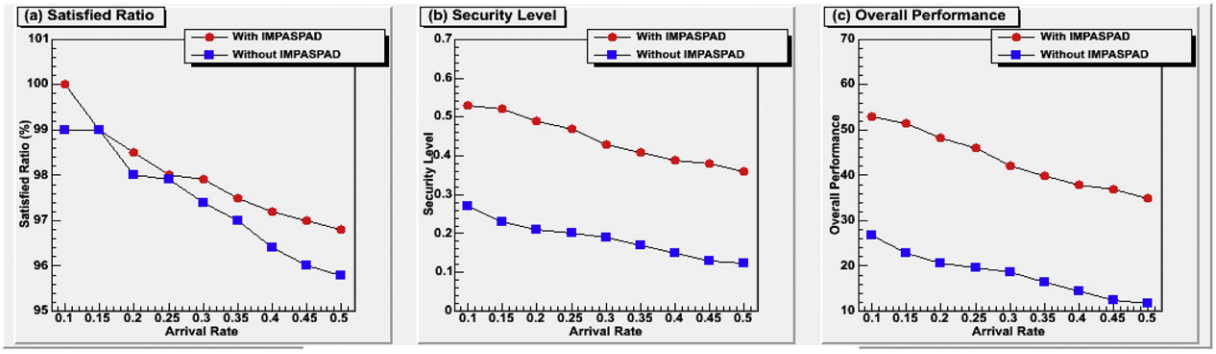


Fig. 6. The impact of the parallelism degree when the arrival rate = 0.5 No/second.



(a) The data size is 100 kB and $P = 3$.



(b) The data size is 10 MB and $P = 8$.

Fig. 7. Impact of arrival rate on the performance of the improved security-aware parallel disk systems.

based on their desired response time. The algorithm keep checking whether the first request can meet its desired response time: if yes, its security level will be increased; otherwise, the algorithm will check if the next request can meet its desired response time, and so on. This technique will cause a starvation for the lower priority requests. To solve this problem, we propose an updated ASPAD algorithm and call it Improved ASPAD (IMPASPAD). The first phase of IMPASPAD is to dynamically calculate the optimal parallelism degree of the request. The third phase is to optimize the security level of each stripe unit using Eq. (11).

When a client issues a request to the system, IMPASPAD inserts the newly arrived requests into the waiting queue based on the earliest desired response time. After the data partitioning of each request in the queue, IMPASPAD initializes the security levels of all stripe units of request r_i to the minimal levels s_i. For the first request in the queue, IMPASPAD checks if that request can meet its desired response time: if yes, the algorithm will check if all the requests in the queue can meet their desired response time; if yes, the security level of the first request will be increased as long as all the requests in the queue can meet their desired response time.

7.1. Improved ASPAD evaluation

To evaluate the performance of the improved ASPAD, we slightly modified the simulator presented in Section 6. The old simulator checks if the estimated response time of the request r_i is less than its desired response time and the security level is less than 0.9; then the security level of the r_i will be increased by 0.1. In the previous simulator, some requests might miss their deadline. To remedy this problem, before increasing the security level of request r_i, we have to make sure that all requests in the queue can meet their desired response time.

7.1.1. Impacts of arrival rate

This experiment aims at comparing the IMPASPAD strategy with a baseline scheme that makes no use of IMPASPAD. Fig. 7 plots the satisfied ratios, security levels, and overall performance of the parallel disk systems with and without IMPASPAD. Fig. 7(a)–(a) and (b)–(a) reveal that the IMPASPAD strategy yields very high satisfied ratios. This is mainly because IMPASPAD endeavors to guarantee the timing constraints of all disk requests in the queue while maximizing the security of the parallel disk system. Fig. 7(a)–(b) and (b)–(b), illustrate that IMPASPAD improves the quality of security of the disk system. It is worth noting that the satisfied ratio using IMPASPAD is higher than the satisfied ratio using ASPAD, and the security level using IMPASPAD is slightly lower than the security level using ASPAD. This is because IMPASPAD does not increase the security level of the request unless it guarantees that all other requests in the queue can meet their desired response time.

8. Summary

High quality of security and guaranteed response times are two major performance goals to be achieved by parallel disk systems. The reason is twofold. First, it is often desirable for next-generation parallel disk systems to be highly flexible in order to support different 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 paper, 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 with 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 has 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 in which 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-partitioning method. Our future work will focus on impacts of network delays on performance of ASPAD. Further, we will integrate ASPAD with fault-tolerant techniques to provide high availability for critical data-intensive applications.

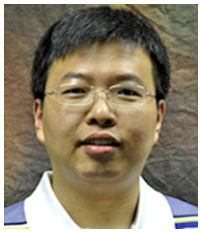
Acknowledgments

The work reported in this paper was supported by the US National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0917137 (CSR), CNS-0915762 (CSR), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS), as well as Auburn University under a start-up grant, a gift (Number 2005-04-070) from the Intel Corporation, South Dakota School of Mines and Technology under the Nelson Research Grant, CNS-1048432, CNS-1007641 and Texas A&M University-Kingsville start-up fund.

References

[1] D. Avitzour, Novel scene calibration procedure for video surveillance systems, IEEE Transactions on Aerospace and Electronic Systems 40 (3) (2004) 1105–1110.
[2] M. Blaze, A cryptographic file system for UNIX, in: Proc. ACM Conf. Communications and Computing Security, 1993.
[3] C. Chang, B. Moon, A. Acharya, C. Shock, A. Sussman, J. Saltz, Titan: a high-performance remote-sensing database, in: Proc. the 13th Int'l Conf. Data Engineering, April 1997.
[4] Z. Dimitrijevic, R. Rangaswami, Quality of service support for real-time storage systems, in: Proc. Int'l Conf. IPSI, Sv. Stefan, Montenegro, October 2003.
[5] J. Hughes, D. Corcorna, A universal access, smart-card-based, secure file system, in: Atlanta Linux Showcase, October 1999.
[6] M. Kallahalla, P.J. Varman, Improving parallel-disk buffer management using randomized writeback, in: Proc. Int'l Conf. Parallel Processing, August 1998, pp. 270–277.
[7] D. Kotz, C. Ellis, Cashing and writeback policies in parallel file systems, in: Proc. IEEE Symp. Parallel and Distributed Processing, December 1991, pp. 60–67.
[8] W. Leung, L. Miller, Scalable security for large, high performance storage systems, in: Proc. of the Second ACM Workshop on Storage Security and Survivability, 2006, pp. 29–40.
[9] D. Mazieres, M. Kaminsky, M. Kaashoek, E. Witchel, Separating key management from file system security, in: Proc. ACM Symp. Operating System Principles, December 1999.
[10] E. Miller, D. Long, W. Freeman, B. Reed, Strong security for distributed file systems, in: Proc. Symp. File Systems and Storage Technologies, January 2002.
[11] M. Nijim, X. Qin, T. Xie, Modeling and improving security of a local disk system for write-intensive workloads, ACM Transactions on Storage (2007) (in press).
[12] M. Nijim, X. Qin, T. Xie, M. Alghamdi, Awards: an adaptive write scheme for secure local disk systems, in: Proc. 25th IEEE Int'l Performance Computing and Communications Conference, Phoenix, AZ, April 2006.
[13] X. Qin, Design and analysis of a load balancing strategy in data grids, Future Generation Computer Systems: The Int'l Journal of Grid Computing (2007) (in press).
[14] X. Qin, Performance comparisons of load balancing algorithms for I/O-intensive workloads on clusters, Journal of Network and Computer Applications (2007) (in press).
[15] S. Rajasekaran, Selection algorithms for parallel disk systems, in: Proc. Int'l Conf. High Performance Computing, December 1998, pp. 343–350.
[16] S. Rajasekaran, X. Jin, A practical realization of parallel disks parallel processing, in: Proc. Int'l Workshop Parallel Processing, August 2000, pp. 337–344.
[17] L. Reuther, M. Pohlack, Rotational-position-aware real-time disk scheduling using a dynamic active subset (DAS), in: Proc. 24th IEEE Int'l Real-Time Systems Symp., Cancun, Mexico, December 2003.
[18] E. Riedel, M. Kallahalla, R. Swaminathan, A framework for evaluating storage system security, in: Proc. the 1st Conf. File and Storage Technologies, Monterey, CA, January 2002.
[19] P. Scheuermann, G. Weikum, P. Zabback, Data partitioning and load balancing in parallel disk systems, VLDB Journal 7 (1998) 48–66.
[20] T. Sumner, M. Marlino, Digital libraries and educational practice: a case for new models, in: Proc. ACM/IEEE Conf. Digital Libraries, June 2004, pp. 170–178.
[21] T. Tanaka, Configurations of the solar wind flow and magnetic field around the planets with no magnetic field: calculation by a new MHD, Journal of Geophysical Research (1993) 17251–17262.
[22] J. Wylie, M. Bigrigg, M. Strunk, G. Ganger, H. Kilicote, P. Khosla, Survivable information storage systems, IEEE Computer Society 33 (8) (2000) 61–68.
[23] T. Xie, X. Qin, Enhancing security of real-time applications on grids through dynamic scheduling, in: Proc. 11th Workshop Job Scheduling Strategies for Parallel Processing, June 2005.
[24] T. Xie, X. Qin, A new allocation scheme for parallel applications with deadline and security constraints on clusters, in: Proc. IEEE Int'l Conf. Cluster Computing, Boston, USA, September 2005.
[25] T. Xie, X. Qin, A security-oriented task scheduler for heterogeneous distributed systems, in: Proc. 13th Annual IEEE Int'l Conf. High Performance Computing, HiPC, Bangalore, India, December 18–21, 2006.
[26] T. Xie, X. Qin, Scheduling security-critical real-time applications on clusters, IEEE Transactions on Computers 55 (7) (2006) 864–879.
[27] T. Xie, X. Qin, M. Nijim, SHARP: a new real-time scheduling algorithm to improve security of parallel applications on heterogeneous clusters, in: Proc. 25th IEEE Int'l Performance Computing and Communications Conf., Phoenix, AZ, April 2006.
[28] T. Xie, X. Qin, Improving security for periodic tasks in embedded systems through scheduling, ACM Transactions on Embedded Computing Systems (2007) (in press).

Mais Nijim is an Assistant Professor in the Department of Electrical Engineering and Computer Science at Texas A&M University-Kingsville. Prior joining Texas A&M University-Kingsville, she was an Assistant Professor at the School of Computing at The University of Southern Mississippi. She received her Ph.D. in Computer Science from New Mexico Institute of Mining and Technology in 2007. Her research interests include parallel and distributed systems, real-time computing, storage system, fault tolerance, and performance evaluation.



Ziliang Zong received his Ph.D. degree in Computer Science from Auburn University, in 2008. From October 2003 to October 2004, he studied as a research assistant student in the Artificial Intelligence Laboratory of Toyama University, Japan. Currently, he is an assistant professor in the Mathematics and Computer Science Department of South Dakota School of Mines and Technology. His research interests include multi-core technologies, parallel programming, high-performance computing and distributed storage systems. He has served as program committee members of several international conferences, including the MICS 2009 conference, SIMUL 2009 conferences, ITNG 2010 conferences and IEEE NAS 2010 conference. In 2009, he received an NSF Computer and Networked Systems (CNS) Award working on data-mining-based pre-fetching techniques for hybrid storage systems.

Shu Yin received his B.S. in Communication Engineering from Wuhan University of Technology (WUT) in 2006. He received his M.S. degree in Signal and Information Processing from WUT in 2008. Currently, he is a Ph.D. student in the Department of Computer Science and Software Engineering at Auburn University. His research interests include storage systems, reliability modeling, fault tolerance, energy-efficient computing, and wireless communications.

Kiranmai Bellam received her B.S. degree in Information Technology from Madras University, India, and her M.S. degree in Computer Science from New Mexico Institute of Mining and Technology. She received her Ph.D. in Computer Science from Auburn University in 2009. Currently she is working as an Assistant Professor in the Department of Computer Science at Prairie View A&M University. Her research interests are in energy efficiency, security, storage systems, real-time computing, distributed systems, and reliability.

Xiao Qin received his B.S. and M.S. degrees in Computer Science from Huazhong University of Science and Technology in 1992 and 1999, respectively. He received his Ph.D. degree in Computer Science from the University of Nebraska-Lincoln in 2004. Currently, he is an Associate Professor in the Department of Computer Science and Software Engineering at Auburn University. Prior to joining Auburn University in 2007, he had been an Assistant Professor at the New Mexico Institute of Mining and Technology (New Mexico Tech) for three years. He won an NSF CAREER award in 2009. His research interests include parallel and distributed systems, storage systems, fault tolerance, real-time systems, and performance evaluation. His research is supported by the US National Science Foundation, Auburn University, and Intel Corporation. He had served as a subject area editor of IEEE Distributed System Online (2000–2001). He has been on the program committees of various international conferences, including IEEE Cluster, IEEE IPCCC, and ICPP.