A Secure File Allocation Algorithm for Heterogeneous Distributed Systems

Yun Tian, Jiong Xie, Shu Yin
Ji Zhang, Xiao Qin
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama 36849-5347
http://www.eng.auburn.edu/∼xqin

Mohammed I. Alghamdi
Department of Computer Science
Al-Baha University
Al-Baha City
Kingdom of Saudi Arabia
Email: mialmushilah@bu.edu.sa

Meikang Qiu
Department of Electrical and Computer Engineering
University of Kentucky
Lexington, Kentucky
Email: mqiu@engr.uky.edu

Yiming Yang
Intel Corporation
NM 87124
Email: yiming.yang@intel.com

I. Abstract

***Abstract*—In this study we develop a secure allocating processing (SAP) algorithm for the S-FAS scheme [13] to improve the security level and consider its performance using the heterogeneous feature of a large distributed system. The SAP allocation algorithm considers load balancing, delayed effects caused by the workload variance of many consecutive requests, and the heterogeneous feature of the storage nodes in the system. We develop a prototype using the multi-threading technique for the S-FAS scheme with the SAP algorithm to guide the file allocation. We did some experiments and the results show that the proposed solution can not only improve the security level, but also improve the throughput of the distributed storage system with heterogeneous vulnerabilities by using the multi-thread technique.**

II. Introduction

A. Motivation

The confidentiality of security-sensitive files must be preserved in modern distributed storage systems, because they are exposed to an increasing number of attacks from malicious users [9].

In a large-scale distributed system, different storage sites have a variety of ways to protect data. The same security policy may be implemented in various mechanisms. Data encryption schemes may vary, even with the same encryption scheme, key lengths may vary across the distributed systems. The above factors can contribute to different vulnerabilities among storage sites. Although security mechanisms deployed in multiple storage sites can be implemented in a homogeneous way, different vulnerabilities may exist due to heterogeneities in computational units.

The file fragmentation technique is a popular method used in many distributed and parallel systems to improve availability and performance of files. Several file fragmentation schemes have been proposed to achieve high assurance and availability in large distributed systems [1][15].

There are tradeoffs among desired features(e.g., security and performance) of distributed storage systems. Most existing solutions improve the security of the systems at the cost of system performance. However, both high security and performance are among the top client desired features in many widely deployed data centers operated by Google, Amazon and Yahoo, etc.. The exploration and development of solutions that can improve not only system security, but also system permormance is in high demand.

B. Our Solutions

In our previous research [13] we proposed a scheme, S-FAS, to address security heterogeneity issues by dividing storage servers into different server groups. We focus on the development of a file fragmentation and allocation approach to improving the assurance and scalability of a heterogeneous distributed system. If one or more fragments of a file have been compromised, it is still very hard for a malicious user to reconstruct the file from the compromised fragments. Our solution is different from previous approaches, because ours captures heterogeneous features regarding to vulnerabilities among servers. In a server group, storage servers with the same vulnerability share the same weakness allowing attackers to reduce the servers' information assurance. Although it may be impractical to classify all servers in a system into a large number of groups, a reasonable way of identifying server types is to organize servers with similar vulnerabilities into one group. We built static and dynamic assurance models to evaluate the security level of our S-FAS scheme. Analysis results show that the system assurance level can be improved by the S-FAS scheme compared with the tradational methods without the consideration of heterogeneous features.

The time cost to reconstruct a file from its fragments is obviously greater than that of the non-fragmentation storage method; and performance degradation becomes inevitable [14]. In this research we investigate impact of secure fragmentation scheme on system performance. Similar research work on file assignment can be found in the literature [7] [3]. These studies showed that not only the queuing cost, and file heat of the file (represents the product of file access rate and the access service time) [3] [10], but also the variance of service time can influence the performance of homogeneous systems where each site has the same performance characteristics. In our investigated heterogeneous distributed systems, the variance in service time caused by heterogeneous servers with different

1530-2016/11 $26.00 © 2011 IEEE
DOI 10.1109/ICPPW.2011.40

IEEE computer society

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.

performance characteristics has a large influence on the overall system performance.

We investigate the possible parameters that influence both the system performance and the proposed S-FAS scheme. There are three aspects to a distributed storage system that influence its security and performance, workload, storage nodes and network interconnects. There are different elements of each aspect of the system. We extracted the possible key elements of each aspect by analyzing of the proposed S-FAS scheme.

Based on the both performance and security analysis, we developed a secure allocating processing (SAP) algorithm for the S-FAS scheme to both improve the security level and consider system performance by using the heterogeneous feature of large distributed systems. The SAP allocation algorithm considers load balancing, delays caused by the workload variance of many consecutive requests, and the heterogeneous nature of storage nodes in a system. We developed a prototype of S-FAS using the multi-threading technique to implement the SAP algorithm to guide file allocations. The experiment results show that the proposed security solution can not only improve security, but also improve the throughput of a distributed storage system with heterogeneous vulnerabilities by virtue of the multi-threading.

C. Main Contributions

In this study, we made the following contributions:

- First, we developed a secure allocating processing (SAP) algorithm to improve security and system performance by considering the heterogeneous feature of a large distributed system.
- Second, in order to conduct the performance analysis for the S-FAS scheme and SAP allocating algorithm, we developed a prototype for our S-FAS scheme.
- Third, we implemented the prototype and conducted some experiments on the throughput of the proposed scheme and algorithm.

D. The Organization of this Paper

The rest of the paper is organized as follows. In Section III, we review the related work. Section IV briefly describes the S-FAS scheme. Section VI presents the the architecture and design of our prototype. In Section V, the SAP allocating algorithm is presented and a sample allocating process is illustrated. Section VII describes experimental scenarios and results of this study. Section VIII summarizes this paper and outlines our future work.

III. Related Work

Much work has been done to improve both the security and performance of distributed systems. This section briefly presents previous approaches to the problem which are most relevant to our solution in three areas: fragmentation technique, secret sharing and load balancing.

A. Fragmentation Techniques

A fragmentation technique partitions a security sensitive file into multiple fragments that are distributed across different storage servers in a distributed system. A lot of fragmentation schemes have been proven to be valuable tools for improving the security of data stored in distributed systems (see [5]). Many fragmentation approaches aim to improve availability and performance of distributed systems by applying data replication methods. For example, Dabek *et al.* developed a wide-area cooperative storage system in which a fragmentation scheme was implemented to improve availability and to facilitate load balancing [4].

Although combining a fragmentation and replication scheme can enhance performance and availability, data replications may increase security risks due to an increasing number of file fragments handled by distributed storage servers. A file is more likely to be compromised when more replications of the file are stored in distributed storage servers.

Existing file fragmentation technologies are inadequate in addressing the issue of heterogeneous vulnerabilities in large-scale distributed systems. Our preliminary results show that security can be improved in a distributed storage system when a fragmentation scheme incorporates the heterogeneous-vulnerability feature with the S-FAS scheme.

B. Secret Sharing

Secret sharing–independently invented by Shamir and Blakley–is a method of distributing a secret among a group of participants, each of which is allocated a share of the secret. The secret can be successfully reconstructed only when a sufficient number of shares are collected and combined [8][11].

Shamir proposed the (k, n) secret sharing scheme that divides data D into n pieces in such a way that D can be easily reconstructed from any k pieces. If fewer than k pieces are disclosed, no one can reconstruct D from the revealed pieces.

The secret sharing scheme has been extended and employed in different application domains [12]. For example, Bigrigg *et al.* proposed an architecture called PASIS for secure storage systems. The PASIS architecture integrates the secret sharing scheme with information dispersal to improve security, integrity and availability [16][18]. In a storage system with PASIS, even if an attacker compromises a limited (i.e., fewer than the threshold) subsets of storage nodes, the confidentiality of data stored in the system is still preserved. The aforementioned secret-sharing solutions designed for distributed storage systems ignore the issue of heterogeneous vulnerabilities. This fact motivates us to extend the secret sharing scheme by considering heterogeneity of vulnerabilities in the context of distributed storage systems.

C. Load Balancing to Improve System Performance

We can classify existing load balancing approaches into different categories such as static and dynamic or homogenous and heterogeneous. The focus of homogeneous load balancing

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.

schemes is to improve the performance of homogeneous parallel and distributed systems. On the other hand, heterogeneous load balancing approaches attempt to boost the performance of heterogeneous clusters, which comprise a variety of nodes with different performance characteristics in computing power, memory capacity, and disk speed. A static load balancing scheme for Computational Fluid Dynamics simulations on a network of heterogeneous workstations has been studied by Chronopoulos et al [2]. Their load-balancing algorithm takes both the CPU speed and memory capacity of workstations into account. To dynamically balance computational loads in a heterogeneous cluster environment, Cap and Strumpen explored heterogeneous task partitioning and load balancing. Xiao et al [17]. have investigated an approach that considers both system heterogeneity and effective usage of memory resources, thereby minimizing both CPU idle time and the number of page faults in heterogeneous systems.

D. Comparison of Our Work with Existing Solutions

Our algorithm, prototype and experiments on the S-FAS fragment allocation solution described in this paper are entirely different from existing fragment allocation schemes, because our approach aim to incorporate the vulnerability heterogeneity feature of distributed storage systems into file fragment allocation. Our solution captures heterogeneous features of the nodes regarding vulnerabilities and performance to improve security and performance of the distributed storage system. In this study the data replication technique is not considered, because fragment allocation and data replication are independent of each other. Thus, the reliability and performance of fragment allocation schemes can be improved when data replication modules are integrated.

IV. A S-FAS SCHEME

A. System Model and S-FAS: A Secure Fragment Allocation Scheme

The S-FAS fragmentation allocation scheme was designed for a distributed storage system where each storage site is a cluster storage subsystem. Different cluster storage subsystems may be connected within some subnetworks to form a larger scale distributed storage system.

In the previously described distributed system, when fragmentation technique and random allocating method are used, it is possible that the fragments of a file be allocated to one group of servers that have the similar vulnerabilities. In such a case, a malicious user can ealily compromise all of the required fragments of a file. To solve the above security problem, we have incorporated vulnerability heterogeneities into fragment allocation schemes. Specifically, we designed a simple yet efficient approach to allocating file fragments to storage nodes with various vulnerabilities. Since allocating fragments of a file into different storage clusters can degrade performance, our S-FAS scheme attempts to allocate fragments to storage nodes within a cluster. If the number of nodes with different vulnerabilities can not meet the aforementioned criterion, file fragments must be allocated across multiple clusters. To improve the assurance of a distributed storage system while maintaining high I/O performance, each cluster storage subsystem must be built with high vulnerability heterogeneity. In doing so, the fragments of a file are less likely to be distributed across multiple storage clusters.

Due to the following two reasons, the S-FAS scheme can significantly improve data security when fragments are stored in a large-scale distributed storage system. First, S-FAS integrates the fragmentation technique with secret sharing. Second, S-FAS addresses the issue of heterogeneous vulnerabilities when file fragments are allocated to a distributed storage system.

The S-FAS scheme makes fragment allocation decisions by following the four policies below:

- **Policy 1:** All the storage nodes in a distributed storage system are classified into multiple server-type groups (server group for short) based upon their various vulnerabilities. Each server group consists of storage nodes with the same or similar vulnerability set.
- **Policy 2:** To improve security of a distributed storage system, S-FAS allocates fragments of a file to storage nodes belonging to as many different server groups as possible. In doing so, it is impossible to compromise the file's fragments using a single successful attack.
- **Policy 3:** The fragments of a file are allocated to nodes with a wide range of vulnerability levels within a single cluster storage subsystem. The goal of this policy is to improve performance of the storage system by making the fragments less likely to be distributed across multiple clusters.
- **Policy 4:** The (m, n) secret sharing scheme is integrated with the S-FAS allocation mechanism.

If a file's fragment-allocation decisions are guided by the above policies, successful attacks against less than m server groups have little chance to gain unauthorized access of files stored in a distributed system. In other words, if the number of compromised fragments of a file is less than m, attackers are unable to reconstruct the file from the fragments that are accessed by the attackers. The S-FAS scheme can improve information assurance of files stored in a distributed storage system without enhancing confidentiality services deployed in cluster storage subsystems of the distributed system, because S-FAS is orthogonal to security mechanisms that provide confidentiality for each server group in a distributed storage system. Thus, S-FAS can be seamlessly integrated with any confidentiality service employed in distributed storage systems to offer enhanced security services.

B. Assurance Models and Evaluation of System Assurance

We developed a static assurance model and a dynamic assurance model to quantitatively evaluate security of a heterogeneous distributed storage system in which S-FAS handles fragment allocations. Our current study focuses on the static assurance model. Since the details of the assurance model do not have direct influence on the understanding of the current

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.

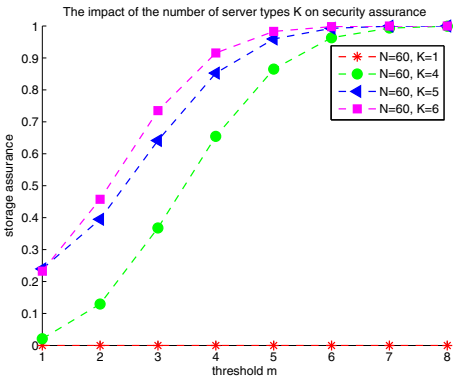

Fig. 1. *Heterogeneous system and homogeneous system using secret sharing scheme.* In all the four test cases, N is set to 60. K is set to 1, 4, 5, and 6, respectively. When K is 1, there is only one server group in the system.

study, we will not present the assurance models here. We will simply present some key experiment results which have direct influence on the current study.

The assurance models indicate that system assurance is affected by the number K of storage types as it is plotted in Fig. 1. N denotes the number of storage nodes in the system. We compared our approach with a traditional fragment allocation scheme that does not consider vulnerability heterogeneities. We evaluated a distributed storage system with the threshold value m. The default number n of fragments of a file is set to 12 and S_j=$\frac{N}{K}$ for all j from 1 to K .

Fig. 1 shows the impact of the number K of storage types on system assurance. We can see that if all storage nodes in the evaluated distributed system are identical in terms of vulnerability, the probability that fragments of a file are compromised using one successful attack method is 1. Results plotted in Fig. 1 suggest that in a distributed system with homogeneous vulnerability, threshold m has no impact on system assurance. When it comes to a distributed system with heterogeneous vulnerabilities, the system assurance increases significantly with increased values of K and threshold m. Such a trend implies that a high heterogeneity level of vulnerability gives rise to high confidentiality assurance. This result implies that we should allocate the fragments of a file to as many types of storage nodes as possible to improve the security level of the system.

V. SAP Allocation Algorithm

A. Factors Affecting Performance and Security

There are three main sources affecting the processing delay of a request from a client: workload, storage nodes, and network interconnects.

Concerning the workload, the service time, S_i, for each file F_i, and the access rate of this file, A_i, are the two main characteristics that affect the performance on the file. These two characteristics are usually combined in a joint metric called "heat" to evaluate the active rate of a file [6]. The service time is directly influenced by file size and two parameters-m and n-in the threshold (m, n). Since we consider the performance of heterogeneous systems, service time S_i for file F_i is not fixed, because different allocation nodes are chosen to store the fragments of the file.

In order to fully benefit from the performance capabilities of a homogeneous distributed or parallel system, one has to insure that the load must be uniformly distributed among all storage nodes. Otherwise, some disks may become performance bottleneck, severely increasing the response time of requests, and reducing the overall system throughput [6]. As such reducing the time cost on network and slave nodes may be achieved by minimizing the utilization of each node and by minimizing the variance of service times among all the nodes. Most of the current published work on this topic concentrated on minimizing the nodes utilization by balancing load across all disks while ignoring the minimization of the variance of service time in the context of heterogeneous distributed systems. When fragments of a wide variety of sizes are intermixed on each node, small fragment requests are likely to wait for large fragment requests accesses that were queued ahead of the smaller ones. This is inefficient, especially when the load is heavy and the queuing delays dominate the response time. Thus, in addition to load balancing, the performance of a distributed system can be improved by reducing the variance of service times among fragment storage nodes. System network interconnects also affect performance of heterogeneous distributed systems. Thus, our file fragment allocation algorithm has to take the network delays into account.

Experimental results described in section show that diversity of a system with heterogeneous features can possibly increase the system security. To make use of the heterogeneous nature of vulnabilities in storage nodes, the SAP algorithm should allocate the fragments of a file to as many types of storage nodes as possible to improve the system security.

Before presenting details of the allocation algorithm, let us summarize in Table I the notation used throughout this paper.

While considering the impact of different bandwidths of networks, we define the coefficient w_N (See weight in Eq. 1). Equation 1 is used to determine load to be distributed to the Nth node concerning the network speed variance of different storage node. This equation helps to guarantee that storage nodes connected by fast networks handle more I/O loads.

$$w_N = \frac{b_N}{\sum_{i=1}^{N} b_N} \tag{1}$$

Based on workload analysis, we use $burden$(see Eq. 2 below) to evaluate the workload that a fragment brings to the storage node that stores the fragment.

$$Burden_{ij} = \lambda_{ij} * DataSize_{ij} \tag{2}$$

In order to balance the loads imposed on different nodes in the system, we distribute the workload expressed by Eq. 3 where a number of fragments of multiple files are stored in node N.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.

TABLE I
NOTATIONS

Notations	Definition
S	the size of file F
N	Number of server nodes in the system
u	server type in the system
K	the total number of server type
S_i	the size of server type i in a cluster
m	threshold for the secret sharing scheme used
$\|F\|$	the total number of files
$\|F_i\|$	the total number of fragments in file $\|F_i\|$
Fij	the jth fragment of the ith file
$Burden_{ij}$	the work load that the jth fragment of the ith file brings to the node where it is stored
$DataSize_{ij}$	the jth fragment size of the ith file
b_N	Bandwidth of the connected network of the Nth node
λ_{ij}	the access rate of the jth fragment of ith file
N_{uv}	the v th node of server type u u,v= 0,1...
I_{ij}	Decreasing sorted list of fragments to be allocated (Fragments of the same file are consecutive and belong to the same row)
$TLoad\,(N_{uv})$	total available load of node N_{uv}
$CLoad\,(N_{uv})$	current load of node N_{uv}
$RLoad\,(N_{uv})$	N number of element vector recording the total available storage capacity for all nodes
CurrentN[u]	the current available node in the uth type of servers
LoadBN_{uv}	the most load that shoud be assigned to node N_{uv}

$$NBurden_{ij} = w_N * \sum_{i=1}^{i=d} \sum_{j=1}^{j=|F_i|} \lambda_{ij} * DataSize_{ij} \quad (3)$$

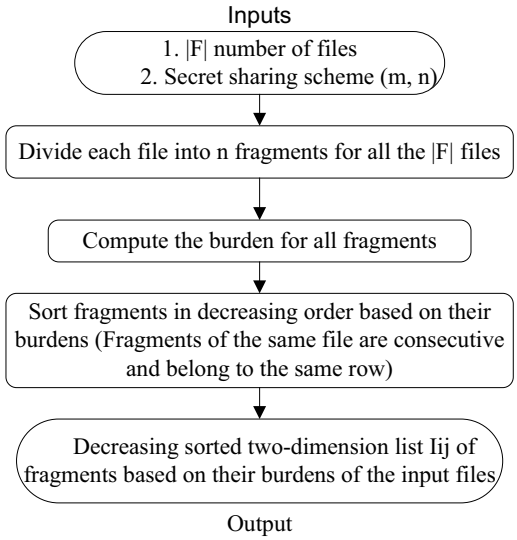

Fig. 2. *The SAP fragment Partition and Sorting Step. The chosen secret sharing scheme (m, n) is employed.*

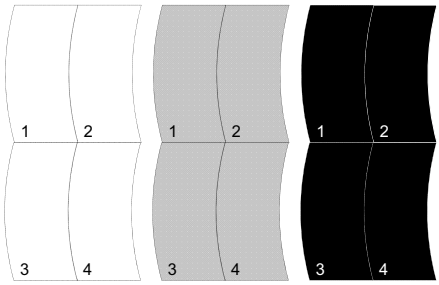

Fig. 3. Allocation Load Sample

B. Allocation Algorithm

We developed a static allocation algorithm integrated with the S-FAS scheme. we focus on the static algorithm rather than its dynamic counterpart, because we plan to consider the movement of data in our future work.

Initially, all nodes within the same server type are listed in a decreasing order of processing speed under a certain workload. All files are sorted in list I in a descending order of fragment sizes. Then, the nodes are allocated to the file fragments in order to optimize both performance and security. The algorithm includes the partitioning and sorting part(see Fig. 2) and fragment allocation steps(see Algorithm 1).

C. An Example

Let us use an example to demonstrate the process of assigning storage nodes to file fragments. There are three files and each file has four fragments to be allocated(see Fig. 3). The data size, fragment size, access rate, and file heat are listed in Table II. Here we assume the size unit of fragment or file is megabyte(MB). The access rate is measured by how many times' requests to a file per second. Heat is defined as in [6] to evaluate the active rate of a file. Here we assume the heat value of the hottest file in a system to be 100, and the heat values of other files are comparably set to values from 0 to 100. Because our analysis is independent of the unit of each item as previously described, we do not list the units in Tables II, III and IV.

TABLE II
FEATURES OF EXAMPLE FILES

Aspects of the File	$F1$	$F2$	$F3$
FileSize	80	60	40
Fragment Size	20	15	10
AccessRate	3	6	5
Heat	60	90	50

We assume that there are three server types, each of which contains three servers. The network bandwidth of each server is listed in the second column in Table III. According the

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 SAP Allocating Process Step:

Input:
I_{ij}
$CLoad\ (N_{uv})$
$TLoad\ (N_{uv})$

Step1. Sort all nodes in a two-dimension list in decreasing order based on their vulnerability type and processing speed.

Step2. Allocate each fragment in list I to servers
i=1
for $u = 0$ and $u < K$ **do**
  CurrentN[u] $= 0$
end for
$u = 0$
v = CurrentN[u]
CurrentN [u] $= u$
use the equations for w_N and $NBurden_{ij}$ to calculate LoadB N_{ij}
for $i = 1$ and $i < |F|$ **do**
  //Allocate all fragments of file i
  for $j = 1$ and $j < |F_i|$ **do**
    if $CLoad\ (N_{uv}) \geq$ LoadBN$_{uv}$ **then**
      CurrentN [u] $++$
    end if
    $CLoad\,(N_{uv})$= $CLoad\,(N_{uv})$+$Burden_{ij}$
    $u++$
    v = CurrentN[u]
    if $u = K$ **then**
      $u = 0$
    end if
  end for
end for

Output:
The updated fragment list for multi-thread fragment writing

Equations 1, 2 and 3, we can calculate each node's upper bound of affordable burden (see the third column in Table III) for these files to be allocated.

TABLE III
FEATURES OF A HETEROGENEOUS STORAGE SYSTEM EXAMPLE

Nodes	Bandwidths Weight	BurdenAfford–Upbound
N_{11}	1	40
N_{12}	2	80
N_{13}	3	120
N_{21}	4	160
N_{22}	1	40
N_{23}	2	80
N_{31}	3	120
N_{32}	3	120
N_{33}	1	40

Fig. 4 shows the allocation results made by the SAP algorithm described in Fig. 2 and Algorithm 1.

Table IV summarizes the allocated burden and the over-loaded load imposed on each node in the system. The negative and positive symbols mean that final allocated burdens exceed or below expected bounds. The nodes that afford a little bit of more load for these bunchs of files, it may be assigned less amount of load while allocating other bunchs of files. The allocation result deviation will be neutralized during the whole system's working process.

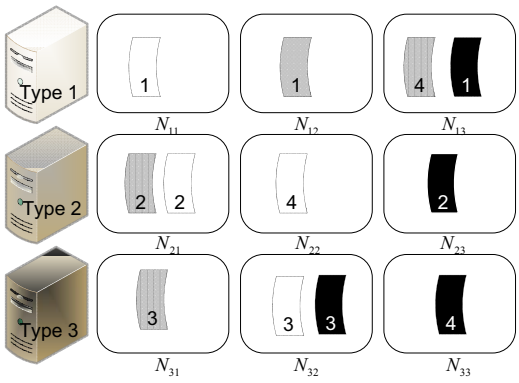

Fig. 4. Alocation Result

TABLE IV
ALLOCATION RESULT BY THE SAP ALGORITHM

Nodes	BurdenAllocated	BurdenOverBound
N_{11}	60	20
N_{12}	90	-10
N_{13}	140	-20
N_{21}	150	+10
N_{22}	60	+20
N_{23}	50	+30
N_{31}	90	+30
N_{32}	110	+10
N_{33}	50	-10

VI. SYSTEM IMPLEMENTATION

Fig. 5 illustrates system architecture of a heterogeneous distributed system, where nodes are classified into three main groups - computing nodes (clients), storage nodes, and a storage server. The storage server is responsible for allocating file fragments using our SAP algorithm, managing metadata, and dealing with incoming file requests(e.g., reads or writes) from clients. The storage nodes are divided into several groups; nodes in each group share similar heterogeneous vulnerability. The client nodes can directly access storage servers through the network interconnections.

Fig. 6 shows the modules implemented in the prototype, which is comprised of the following components.

- The first part is a set of configuration files for servers, files, and all fragments of all file fragments stored in storage nodes.
- The second component is the active lists for servers (see the data structure of the example server node, which

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.

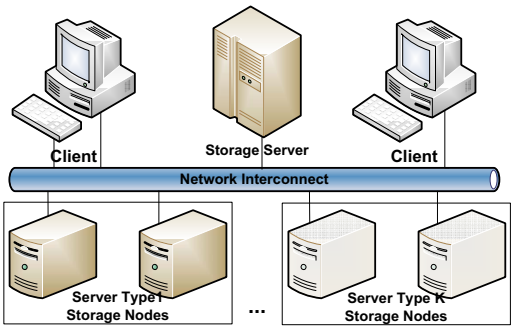

Fig. 5. *System architecture.*The storage server manages metadata. The storage nodes are logically grouped into different server types. Client nodes can directly access storage servers through the network connection.

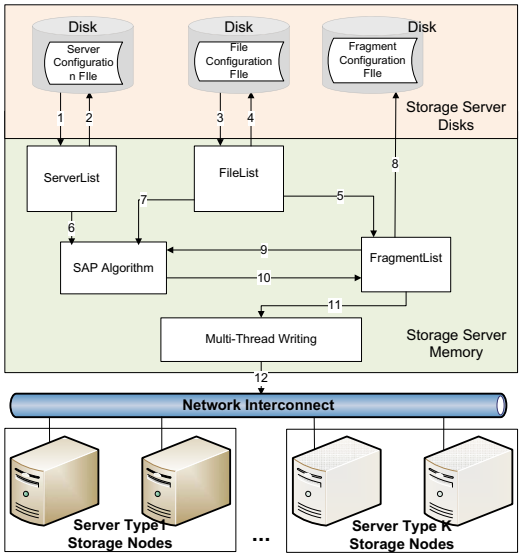

Fig. 6. *The Prototype design.*The upper part of this figure represents the disks of a storage server where the storage node configuration file, file configuration file, and fragment configuration file are stored. The middle part is the memory part of the storage server node, where the key allocating module parts such as the SAP algorithm, multi-thread writing, and the three active linklists are stored. The three active lists records the most updated information for storage nodes, stored files and fragments in the system. The lower layer in the figure represent the client nodes in the system.

forms the above serverlist) , files, and all fragments in the memory of storage node.

- The third part is the SAP algorithm and multiple threads located in the storage server.
- The last component is composed of distributed storage nodes which are used to store file fragments.

The prototype runs on the storage server node and directly communicates with the client nodes. The upper layer in Fig. 6 represents the disks of the storage node, where the server configuration file, the file configuration file, and the fragment configuration file are stored. The middle layer is residing in the main memory part of the server node where the key allocation modules such as the SAP algorithm, multiple writing threads, and the three active linklists are stored. The three active lists record the most recent information of the server, stored files and fragments in the system. The bottom layer in the figure represents the client nodes in the system.

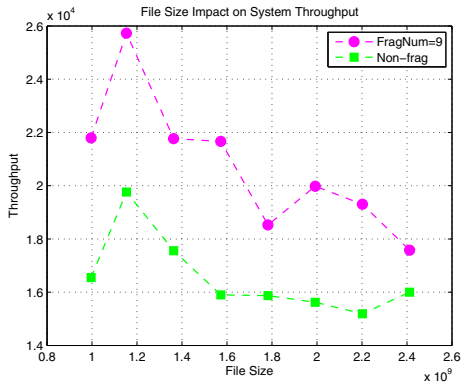

Fig. 7. *Impact of file size on system throughput.*

VII. Performance Evaluation of the S-FAS

A. Evaluation Methodology

To evaluate the performance of our SAP algorithm, we implement a prototype (see Section VI) and analyze the performance of the prototype. We pay attention to two system parameters, namely, data size and the number of fragments for each file. We conducted an experiment to test the file allocation process by varying workload conditions and fragment number in the prototype.

- File size: In our experiments, the file size was varied from 150 MB to 1500MB.
- Fragment number of a file: We varied the fragment number from 3 to 15. All the upper bounds are smaller than 16, because there are 16 storage nodes in our testbed, and the prototype performs better when each storage node stores at most one fragment from each file.

B. Experiment Results

Fig. 7shows the impact of file size on system throughput measured in Byte/Sec. From Fig. 7 we observe that compared with the traditional scheme, our approach significantly improves the throughput. The performance improvement is attributed to themulti-threading method used to handle multiple fragments of a file inprototype. The results show evidence that multiple threading is an efficient way of implementing the SAP algorithm. A second observation drawn from Fig. 6 is that increasing file size can reduce the system throughput. This trend is observed for the two tested schemes. The results obtained from this experiment indicate that file partitioning and multiple threading can significantly improve the performance of a heterogeneous distributed system.

Fig. 8 shows the impact of the fragment number on the throughput. From Fig. 8 we can see that the throughput

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.

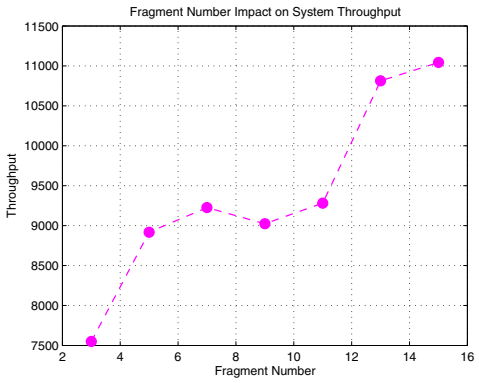

Fig. 8. *Impact of fragment number on system throughput.*

increases when the number of fragments goes up. A large number of fragments in a file can improve the read and write performance of the system with respect of the file, because many storage nodes can access fragments in parallel. With multiple threads in place, multiple storage nodes are able to provide a high aggregated bandwidth.

Figs. 7 and 8 illustrate that file fragmentation and multiple threading are two key techniques to improve performance of the SAP file allocation algorithm designed for heterogeneous distributed systems.

VIII. Conclusion and Future Work

It is critical to maintain the confidentiality of files stored in a distributed storage system without significantly degrading performance. We developed a SAP file allocation algorithm based on the analysis of the assurance model and the proposed S-FAS scheme proposed in our previous research. In order to evaluate the performance of the S-FAS scheme and the algorithm, we built a prototype in a test bed for distributed systems in our laboratory. The experimental results offer us inspiration on how to use S-FAS to efficiently improve security and performance in distributed storage systems with heterogeneous vulnerabilities.

There are a few future research directions of this study. First, we will integrate a data replication technique to be integrated with S-FAS to enhance reliability and performance of the fragment allocation scheme for distributed systems. Second, we will develop a dynamic file allocation algorithm for a distributed system where data replicas are achieved and maintained. Third, we will make an effort to extend the SAP algorithm to address security issues into a cloud storage system. Last but not least, we will continue to upgrade our prototype for future scalability testing.

References

[1] S. Jajodia A. Mei, L. V. Mancini. Secure dynamic fragment and replica allocation in large-scale distributed file systems. *IEEE Trans. on Parallel and Distributed Systems*, 14(9):885 – 896, Sept. 2003.

[2] A.T. Chronopoulos and D. Grosu. Static load balancing for cfd simulations on a network of workstations. In *Network Computing and Applications, 2001. NCA 2001. IEEE International Symposium on*, pages 364 –367, 2001.

[3] George Copeland, William Alexander, Ellen Boughter, and Tom Keller. Data placement in bubba. *SIGMOD Rec.*, 17(3):99–108, 1988.

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215, New York, NY, USA, 2001. ACM.

[5] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGPLAN Not.*, 35:190–201, November 2000.

[6] James F. Kurose and Rahul Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Comput.*, 38(5):705–717, 1989.

[7] Lin-Wen Lee, Peter Scheuermann, and Radek Vingralek. File assignment in parallel i/o systems with minimal variance of service time. *IEEE Trans. Comput.*, 49(2):127–140, 2000.

[8] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, 1992. Springer-Verlag.

[9] M. Pourzandi, D. Gordon, W. Yurcik, and G. A. Koenig. Clusters and security: distributed security for distributed systems. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 96 – 104 Vol. 1, May 2005.

[10] Peter Scheuermann, Gerhard Weikum, and Peter Zabback. Data partitioning and load balancing in parallel disk systems. *The VLDB Journal*, 7(1):48–66, 1998.

[11] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[12] G. J. Simmons. How to (really) share a secret. In *CRYPTO '88: Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, pages 390–448, London, UK, 1990. Springer-Verlag.

[13] Y. Tian, M. I. Alghamdi, Y. Shu, J. Xie, J. Zhang, M.-K. Qiu, Y.-M. Yang, and X. Qin. Secure fragment allocation in a distributed storage system with heterogeneous vulnerabilities. In *Proceedings of the 2011 IEEE International Conference on Networking, Architecture, and Storage*, NAS '11. IEEE Computer Society, 2011.

[14] Manghui Tu, Peng Li, I-Ling Yen, Bhavani Thuraisingham, and Latifur Khan. Secure data objects replication in data grid. *IEEE Transactions on Dependable and Secure Computing*, 7:50–64, 2010.

[15] T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant applications. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, pages 7–7, Berkeley, CA, USA, 1999. USENIX Association.

[16] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kiliccote, and P. K. Khosla. Survivable information storage systems. *Computer*, 33(8):61 –68, Aug 2000.

[17] Li Xiao, Sonqing Chen, and Xiaodong Zhang. Dynamic cluster resource allocations for jobs with known and unknown memory demands. *IEEE Trans. Parallel Distrib. Syst.*, 13:223–240, March 2002.

[18] G. Zanin, A. Mei, and L. V. Mancini. Towards a secure dynamic allocation of files in large scale distributed file systems. In *HOT-P2P '04: Proceedings of the 2004 International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 102–107, Washington, DC, USA, 2004. IEEE Computer Society.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:55:18 UTC from IEEE Xplore. Restrictions apply.