IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 11, November 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

Information Sharing In Parallel Network File Systems

R.Mathivathani, Research Scholar, Department of Computer Science, Marudupandiyar College, Thanjavur
Dr Mala, Department of Computer Science, Marudupandiyar College.

Abstract

In parallel file systems, the file information is distributed transversely multiple storage space devices or nodes to allow parallel access by several tasks of a parallel application. That is classically used in large scale cluster computing that focuses on high concert and consistent fetch to large datasets. That higher I/O bandwidth is achieved through concurrent attractive data to multiple storage devices within large computing clusters, while data loss is confined through data mirroring using defect understanding striping algorithms. Few examples of high performance parallel file systems that are in the creation utilize are the IBM all-purpose Parallel Files System. Independent of the development of the cluster and high recital computing, the emergence of clouds and the MapReduce programming representation has resulted in file system such as the Hadoop Distributed File System (HDFS). In this work, we explore the issue of the secure many to many communications in the large scale network file systems which hold parallel fetch to multiple storing devices. Particularly, we tries to focus on how to switch the key materials and organization of the parallel secure sessions between clients and storage devices in the parallel Network File System (pNFS), the current Internet principles in well-organized and scalable approach. Parallel Network File System (pNFS) is an emerging industry average for parallel storage input/output (I/O), which is an discretionary feature of Network File System (NFS) v.4.1. The new technology involves a pNFS client and a pNFS server.

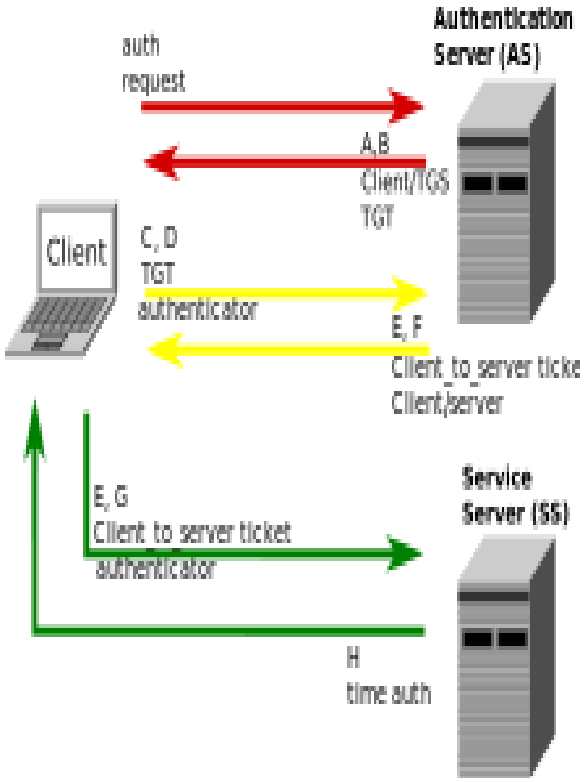
I. INTRODUCTION

The pNFS server deal with file metadata and correspond a file describe map to the client, and the client execute direct storage I/O based on the layout. The storage admittance protocol can be stand on files, blocks or objects. High-performance data midpoint has been aggressively moving in the direction of parallel technologies like clustered computing and multi-core processors. While these augmented uses of parallelism overcome the huge majority of computational blockage, it shifts the presentation bottlenecks to the storage I/O system. To make sure that calculate clusters deliver the greatest performance, storage systems have to be optimized for parallelism. Legacy Network Attached Storage (NAS) architectures stand on NFS v4.0 and earlier have severe performance bottlenecks and supervision challenges when implemented in conjunction with large scale, high routine compute clusters.

An association of storage industry technology privileged created a parallel NFS (pNFS) protocol as an possible extension of the NFS v4.1 standard. pNFS catch a different move toward by allowing calculate clients to read and write openly to the storage, eradicate filer head bottlenecks and agree to single file system capacity and performance to scale linearly.

pNFS is significant because it brings collectively the benefits of parallel I/O with the benefits of the ever-present standard for network file systems (NFS). This will agree to users to familiarity increased performance and scalability in their storage transportation with the added declaration that their

speculation is safe and their capability to choose best-of-breed solutions remains intact.NFS is THE communications protocol standard when it comes to network file systems. It is widely used in both HPC and venture markets today. The pNFS standard is pleasing to both vendors and customers alike. It allows HPC-centric storage retailer such as Panasas to distribute the advantages formerly delivered only via proprietary protocols into the NFS markets. It agrees to Enterprise-focused storage retailer to go through the HPC market more deeply. So for vendors it thickens their markets. For customers, it resources more options and competition for their business. It also agrees to customers to make simpler their IT environments by regulate on pNFS as their regular NAS protocol.



3. REVIEW OF LITERATURE

D. Boneh, C. Gentry, And B. Waters, 2005 presented a paper ***Collusion Resistant Broadcast Encryption With Short Cipher texts And Private Keys*** In a broadcast encryption scheme [FN93] a broadcaster encrypts a message for some subset S of users who are listening on a broadcast channel. Any user in S can use his private key to decrypt the broadcast. However, even if all users outside of S collude they can obtain no information about the contents of the broadcast. Such systems are said to be collusion resistant. The broadcaster can encrypt to any subset S of his choice. We use n to denote the total number of users. Broadcast encryption has several applications including access control in encrypted file systems, satellite TV subscription services, and DVD content protection.

Y. ZHU AND Y. HU. SNARE, 2003 presented a paper ***Strong Security for Network-Attached Storage*** was Survey Computer storage is an increasingly important part of the Internet, and ensuring the security and integrity of stored data is a crucial problem. Attacks by hackers and insiders have led to billions of dollars in lost revenue and expended effort to fix the resulting problems. Most organizations rely heavily on their distributed computing environment, which usually consists of workstations and a shared file system. This file system is typically stored on a centralized file server that is managed by a system administrator with super-user privileges, leaving the data vulnerable to anyone who can obtain (legitimately or otherwise) super-user access. Recently, however, network-attached storage has begun to replace traditional centralized storage systems. In such systems, disks are attached directly to a network, and rely upon their own security rather than using the server's protection. This arrangement makes security more difficult because the disk is directly exposed to potential attacks instead of being

hidden behind a single server that can be "hardened." Most existing secure storage systems provide either authentication or encryption, but not both. For example, CFS encrypts data, but does not easily permit authentication of data or sharing with other users. Systems such as SFS-RO and NASD use encryption to provide network security and authentication, but store data in the clear. Recently, systems such as TCFS and SUNDR have incorporated both authentication and encryption, but at a relatively high penalty to performance. We have developed a security system for network attached storage that relies upon strong cryptography to protect data stored in a distributed storage system.

S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, and C. Maltzahn, 2005 has introduced ***Ceph: A Scalable, High-Performance Distributed File System*** System designers have long sought to improve the performance of file systems, which have proved critical to the overall performance of an exceedingly broad class of applications. The scientific and high-performance computing communities in particular have driven advances in the performance and scalability of distributed storage systems, typically predicting more general purpose needs by a few years. Traditional solutions, exemplified by NFS, provide a straightforward model in which a server exports a file system hierarchy that clients can map into their local name space.

More recent distributed file systems have adopted architectures based on object-based storage, in which conventional hard disks are replaced with intelligent object storage devices (OSDs) which combine a CPU, network interface, and local cache with an underlying disk or RAID. OSDs replace the traditional block-level interface with one in which clients can read or write byte ranges to much larger (and often variably sized) named objects, distributing low-level block allocation decisions to the devices themselves. Clients typically interact with a metadata server (MDS) to perform metadata operations (open, rename), while communicating directly with OSDs to perform file I/O (reads and writes), significantly improving overall scalability.

F.B. Schmuck and R.L. Haskin, 2002 Has proposed a new system in ***Gpfs: a Shared-Disk File System for Large Computing Clusters*** since the beginning of computing, there have always been problems too big for the largest machines of the day. This situation persists even with todays powerful CPUs and shared-memory multiprocessors. Advances in communication technology have allowed numbers of machines to be aggregated into computing clusters of effectively unbounded processing power and storage capacity that can be used to solve much larger problems than could a single machine. Because clusters are composed of independent and effectively redundant computers, they have a potential for fault-tolerance. This makes them suitable for other classes of problems in which reliability is paramount.

As a result, there has been great interest in clustering technology in the past several years. One fundamental drawback of clusters is that programs must be partitioned to run on multiple machines, and it is difficult for these partitioned programs to cooperate or share resources. Perhaps the most important such resource is the file system. In the absence of a cluster file system, individual components of a partitioned program must share cluster storage in an ad-hoc manner.

C. Adams, 1996, presented a study on ***Idup and spkm: developing public-key-based apis and mechanisms for communication security services*** The increasing awareness of the need for security in virtually all forms of digital communication today has been coincident with the distressing realization by many application developers that security is not generally a simple "feature" which can be added into an existing product between one release and the next. In many cases, such "quick fixes" have severe security weaknesses

due to the use of insecure algorithms, insecure protocols, or insecure implementation practices (such as placing clear text keys in message headers, disk files, or easily visible areas of memory).

Although relatively strong cryptographic algorithms and protocols can be found in the open literature, it is clear that what is needed is a straightforward way to make these tools available to application developers in a simple and easy-to-use manner. It is also clear from growing experience that hiding the often difficult and complex issues of key management from the calling applications (particularly in environments with very large user populations) would be of great benefit.

Application Program Interfaces (APIs) offering security services are quickly becoming recognized as the most practical way to meet these needs. They can provide a range of features to an application through a small number of API calls, thus ensuring high security without the substantial effort on the part of the application developer to implement the features “from scratch”. Furthermore, the standardization of security APIs within such bodies as the Internet Engineering Task Force gives applications relatively transparent portability to any number of API implementation vendors (each offering its own set of security services and features).

4. Data Encryption Standard (Des)

Encryption is used to make over a data into some incomprehensible form so that authenticated anyone only can read/access the data. It necessitate some secret information to make over the plain text to cipher text; it is typically referred as key

ALGORITHM REQUIREMENTS:

1. The Key will be reserved secret and should be arbitrary.
2. It should not be achievable to find the key even if the plain text and Cipher text are recognized.

4.1 SECURE HASH ALGORITHM

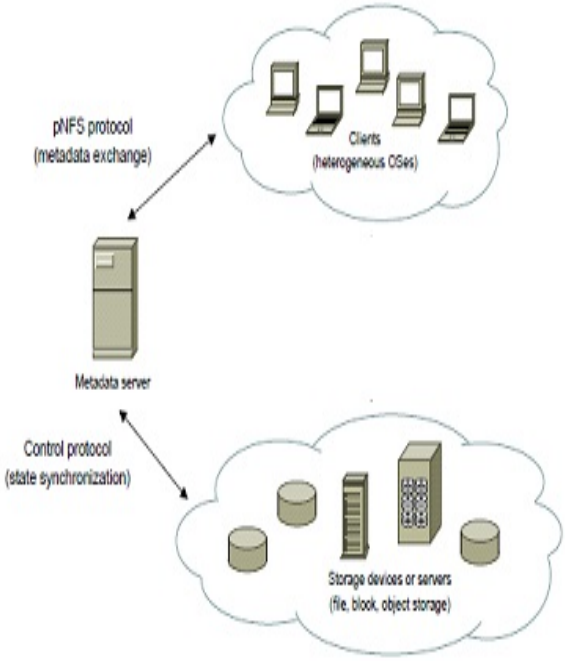
The Secure Hash Algorithm is a people of cryptographic hash functions available by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS), including:

- SHA-0: A retronym functional to the original description of the 160-bit hash function published in 1993 under the name "SHA". It was reserved shortly after publication due to an unrevealed "significant flaw" and reinstate by the slightly modify version SHA-1.
- SHA-1: A 160-bit hash function which be similar to the previous MD5 algorithm. This was planned by the National Security Agency (NSA) to be element of the Digital Signature Algorithm. Cryptographic weakening was exposed in SHA-1, and the typical was no longer accepted for most cryptographic uses after 2010.
- SHA-2: A family of two comparable hash functions, with dissimilar block sizes, recognized as SHA-256 and SHA-512. They fluctuate in the word size; SHA-256 uses 32-bit words somewhere SHA-512 uses 64-bit words. There are also shortened versions of each standard, known as SHA-224, SHA-384, SHA-512/224 and SHA-512/256. These were also intended by the NSA.
- SHA-3: A hash function previously called Keccak, selected in 2012 after a public opposition among non-NSA designer. It chains the same hash extent as SHA-2, and it’s inside structure fluctuate significantly from the rest of the SHA family.

The corresponding values are FIPS PUB 180 (original SHA), FIPS PUB 180-1 (SHA-1), FIPS PUB 180-2 (SHA-1, SHA-256, SHA-384, and SHA-512). NIST has modernized Draft FIPS Publication 202, SHA-3 Standard separate from the Secure Hash Standard (SHS).

4.2 Architecture

During the authentication of the process by using the scanner the input can be taken and then the input can be divided into the number of segments and region. That is quadrangle is applied to the input.



4.3 Proposed Methodology

Our most important goal in this work is to design proficient and secure authenticated key exchange protocols that meet definite requirements of pNFS. Mostly, we effort to meet the following attractive properties, which either have not been Adequately achieved or are not attainable by the present Kerberos-based solution:

• Scalability – the metadata server make possible access requests from a customer to multiple storage devices be supposed to bear as small workload as possible such to facilitate the server will not turn into a performance blockage, but is competent of supporting a very large number of clients;

• Forward confidentiality – the protocol should assurance the security of past meeting keys when the long-term covert key of a client or a storeroom device is compromised;

• Escrow-free – the metadata server be supposed to not learn any in sequence about any session key used by the client and the storage device, supply there is no involvement among them.

The main results of this paper are three new provably protected authenticated key exchange protocols. Our protocols, increasingly designed to realize each of the above belongings, demonstrate the trade-offs between efficiency and security. We demonstrate that our protocols can reduce the workload of the metadata server by in organize of half compared to the current Kerberos-based protocol, while realize the desired security properties and keeping the computational transparency at the clients and the storage devices at a reasonably low point. We define a suitable security model and establish that our protocols are secure in the model.

We propose a range of authenticated key replace protocols that are designed to address the more than issues. We illustrate that our protocols are competent of reducing up to roughly of the workload of the metadata server and alongside supporting forward confidentiality and escrow-freeness. All this necessitate only a small fraction of increased working out overhead at the client.

MERITS

- It reduces a workload of server.
- It contain metadata server which make user to access file easily.
- It reduces the use of key generator.
- It makes the computation of the client easier. Key the computation of the client easier.

5. CONCLUSIONS

We proposed three authenticated key exchange protocols for parallel network file system (PNFS). Our protocols offer three appealing advantages over the existing Kerberos-based Pnfs protocol. First, the metadata server executing our protocols has much lower workload than that of the Kerberos-based approach. Second, two our protocols provide forward secrecy: one is partially forward securing (with respect to multiple sessions within a time period), while the other is fully forward secure (with respect to a session). Third, we have designed a protocol which not only provides forward secrecy, but is also escrow-free.

To improve the performance of the KDC, the developers of HDFS chose to use a number of tokens f or communication secured with an RPC digest scheme. The Hadoop security design makes use of Delegation Tokens, Job Tokens, and Block Access Tokens. Each of these tokens is similar in structure and based on HMAC-SHA1. Delegation Tokens are used for clients to communicate with the Name Node in order to gain access to HDFS data; while Block Access Tokens are used to secure communication between the Name Node and Data Nodes and to enforce HDFS file system permissions. On the other hand, the Job Token is used to secure communication between the Map Reduce engine Task Tracker and individual tasks. Note that the RPC digest scheme uses symmetric encryption and depending upon the token type, the shared key may be distributed to hundreds or even thousands of hosts.

7. REFERENCES

- ***D. Boneh, C. Gentry, And B. Waters, 2005 Collusion Resistant Broadcast Encryption With Short Cipher texts And Private Keys***
- ***Y. ZHU AND Y. HU. SNARE, 2003 Strong Security for Network-Attached Storage.***
- ***S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, and C. Maltzahn, 2005 Ceph: A Scalable, High-Performance Distributed File System.***
- ***F.B. Schmuck and R.L. Haskin, 2002 Gpfs: a Shared-Disk File System for Large Computing Clusters***
- ***C. Adams, 1996, Idup and spkm: developing public-key-based apis and mechanisms for communication security services.***