一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IA2012-47(2012-10)

Design and Implementation of File Authentication System using Timestamp Mechanism for a Distributed File System

Tetsuya OGAKI † ‡　Hidenobu WATANABE ‡　Tsukasa IWAMA ‡

Mitsue Den ‡　and　Ken T. MURATA ‡

† Graduate School of Information Systems,　The University of Electro-Communications

1-5-1 Chofugaoka, Chofu-city, Tokyo, 182-0021 Japan

‡ National Institute of Information and Communications Technology

4-2-1 Nukii-Kitamachi, Koganei-city, Tokyo, 184-0015 Japan

E-mail:　† ohgaki@hpc.is.uec.ac.jp

Abstract　Gfarm, a distributed file system for a petabyte scale data intensive computing or a shared storage for big data, is gradually used as system infrastructure in science research or business. It is a promising file system, however, it does not provide a function to prevent from data falsification. We implemented a file authentication system using timestamp mechanism to work with Gfarm. This paper reports the way to combine timestamp mechanism with Gfarm and the evaluated results using dummy files based on real files in a cloud service using Gfarm.

Keyword　distributed file system, cloud security, file authenticity, timestamp authentication

1. Introduction

Recently, Gfarm[1] becomes popular in the petabyte scale data intensive computing and as base of shared storage for big data. Gfarm is global distributed file system and developed to manage data intensive computing. It is used as base of storage for HPCI[1] and base of system for NICT Science Cloud[2][2]. Activation toward commercialization[3] is in progress, research and development about security and stable operation are going ahead.

According to the report[4] for Ministry of Economy Trade and Industry, one of concern using cloud service is security. Cloud security needs confidentiality, integrity and availability that are usual information security requirement. In addition, it needs authenticity, accountability, reliability. As to Gfarm security, research and development for accountability[5] is going on, but authenticity and reliability are undertaken.

We focus on authenticity and propose the file authentication system using timestamp[6]. Our system combines to Gfarm and timestamp server in application level and provides file admission and verification. This paper reports design and implementation of proposed system, and the method of combination between Gfarm and timestamp server.

We describe organization of this in the following paper.

[1] https://www.hpci-office.jp/
[2] https://seg-web.nict.go.jp/

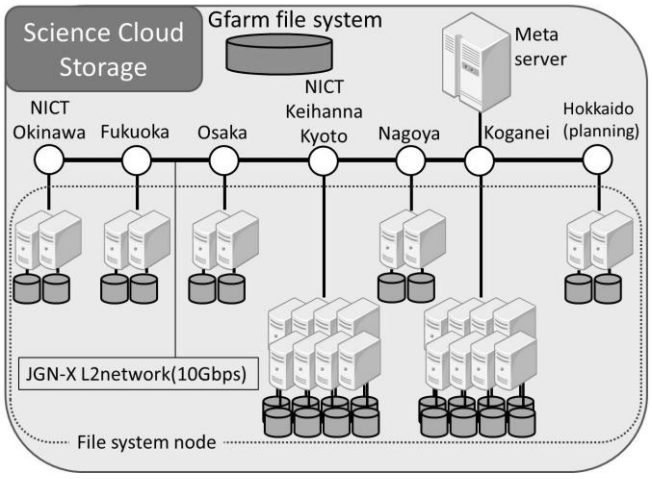

Fig.1 Storage system using Gfarm of NICT Science Cloud.

Section 2 describes Gfarm and authenticity. We present of the file authentication system in section 3, and present implementation of the file authentication system in section 4. We show a requirement and result of evaluation in section 5. Conclusions are in section 6.

2. Gfarm file system and authenticity

2.1. Gfarm file system

Gfarm is global distributed file system consists of one Meta server and file system nodes (FSN). Meta server manages Meta data for the files using a daemon (gfmd) and FSNs process the file operations using a daemon (gfsd). Figure 1 shows a storage system using Gfarm of

NICT Science Cloud. NICT Science Cloud has a Meta server at Koganei and 36 FSNs distributed at each region of Japan.

We explain the procedure of Gfarm operation using figure 1. We supporse that the client is in Osaka. First, the client accesses to Gfarm Meta server at Koganei if the client creates a file at Gfarm. Second, Meta server determines the closest FSN from latency between client's host and the FSN. Of course, the FSN is Osaka node in this situation. Next, the client accesses to the target FSN and creates a file. Finally, Gfarm returns the completed result of the file creation to the client. History of these file operation are recorded to system log of Gfarm on the Meta server. Therefore, Gfarm achieves high data I/O by local disk I/O of FSN.

Gfarm is providing high confidentiality, reliability and availability using Gfarm functions which are GSI[7](Grid Security Infrastructure), a Meta server duplication, and data replication. In addition, log trace function was also proposed for accountability. This function guarantees that the copy file of a removed file does not exist in any FSNs. On the other hand, Gfarm does not provide authenticity and integrity because Gfarm is developed to process data intensive computing for peta byte scale, and it, makes much account of responsiveness for users.

NICT Science Cloud is providing an online storage service using Gfarm for science research as cloud service and is storing a lot of observation data.

Our motivation is to guarantee authenticity of Gfarm for NICT Science Cloud security, so that the data is needed to protect from falsification.

2.2. Authenticity

ACL (Access Control List) and WORM(Write Once Read Many) is commonly used for data access control in general cloud services. These technologies are able to protect from falsification before accessing data. Challenge and Response[8], Functional Encryption[9], and Timestamp authentication enable also to guarantee non-falsification after rewriting the data. To ensure a high authenticity, it is important to check non-falsification of data routinely in the system side in addition to data access control. We chose timestamp authentication from the viewpoint of many introduction example as the best way to guarantee non-falsification after rewriting data.

Timestamp authentication is PKI (Public Key Infrastructure) technique that incorporates reliable time. Timestamp server admits a timestamp token made of the time and the hash value of a file. Timestamp server is managed by CA(Certificate Authority) that is licensed legally. Timestamp token can ensure the time when a file was created. If a file was written by someone, the falsification of the file is proved because of different of hash value. Because of this, using timestamp to the file ensures authenticity. On the other hand, NICT is the highest in the time calibration organization. To adopt timestamp authentication, it is promising utility value for NICT Science Cloud.

3. Design file authentication system

3.1. Requirements for file authentication system

To realize file authentication system using timestamp in Gfarm, it is needed to combine Gfarm with timestamp, because Gfarm supports file operation only and timestamp supports file admission and verification only appointed by user using timestamp token.

Our proposed system adopts timestamp server(SX-3640 TSS). This timestamp server has dedicated timestamp API in which implemented admitting hash value to timestamp server. To use this function, requirement information for timestamp API is hash value of a file, so that we must implement hash calculation function. To check whether files are falsified or not, it is needed to compare present hash value of the target file to the admitted timestamp token. Accordingly, our proposed system specifies IP address for FSN where stores the file and file path of the file for admission, and manages timestamp tokens for verification.

In Gfarm, Meta server manages information of files. Because of this, our proposed system is needed to implement a function that detects new file creation on the Meta server. It is also needed to supply information of file creation on Meta server instantly to target FSN where stores the file.

Timestamp token which is admitted by timestamp server must manage mapping information that maps the file and timestamp token because it is required for file verification. In addition, to reduce accessing Meta server and to manage optimization, it is needed that manage IP address of FSN where stores file, file path, admission and verification time.

Our proposed system should reduce load in Meta server as less as possible because Gfarm is global distributed file system whose Meta server is central intensive model. Specially, calculating hash value needs machine resource, so that calculating hash value on FSN is desirable that store the file.

Straighten above items, below functions are needed.

(A). Monitoring to Gfarm log.

(B). Exchanging information Meta server and FSNs to authenticate files.

(C). Calculating a hash value.

(D). Comparing a present hash value to admitted timestamp token.

(E). Management of timestamp token.

(A) is to implement a function to detect a new file creation on the Meta server. Gfarm has log trace function that outputs information of a file operation to the log, so that we designed combination interface that detects a new file creation automatically monitoring the trace log. To admit a file to timestamp server, it is needed a file path of the target file, IP address of FSN storing the file. This information is outputted to Gfarm trace log, so that combination interface also gets this. (B) is needed to reduce the Meta server load. We designed 2 daemons which residents on Meta server(tmd) and on each FSNs(tsd). Tmd has communication interface to output the information collected by combination interface to the target FSN. Tsd has communication interface to return the result to Meta server and timestamp API for admission. (C) is needed to get a timestamp token. (D) is needed to check whether the files are falsified or not. (C) and (D) need machine resource, so that we implement these functions to tsd considering Meta server load. (E) manages timestamp token completed admission. The requirements of managing timestamp token is file path of the timestamp token, file path of the admitted file, and the admitted time. Our proposed system manages timestamp token using XML (Extensible Markup Language) database because Gfarm supports XML extended attribute.

Gfarm's load is concentrated to Meta server because it is global distributed file system whose Meta server is central intensive model. Because of this, our proposed system must reduce the load as less as possible, so that Meta server runs processing order and each FSNs run timestamp processing. Specifically, tmd residents on the Meta server and tsd residents on the FSNs, and tmd runs processing order and tsd runs timestamp processing.

3.2. System architecture

Our proposed system combines to Gfarm and timestamp server in application level, and consists of tmd and tsd, and timestamp server. Figure 2 shows the proposed system architecture. We called the modules which is the inside of dashed line Authentication Module. Each daemon has two modes that run admission and verification to the target file. Tmd has 3 functions that monitor Gfarm trace log on the

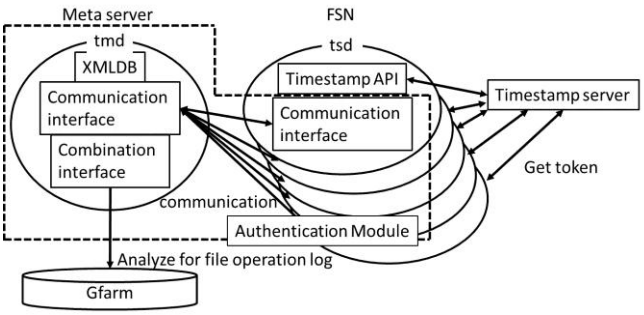

Fig.2 Proposed system architecture.

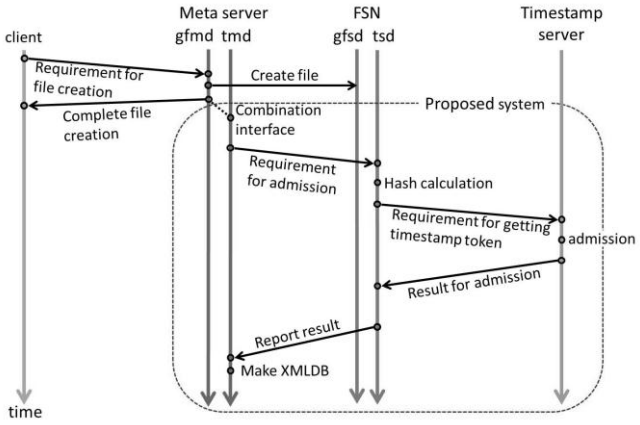

Fig.3 Process of timestamp admission.

Meta server, combination interface analyze the trace log, and communication interface which is able to communicate with tsd. Tsd has 4 functions that communication interface which is able to communicate with tmd, calculate hash value of the target file, verifying timestamp tokens, and timestamp API. Timestamp API provides admitting request to timestamp server. To manage the timestamp token uses XML database. Gfarm has a function to define an extended attribute using XML by user, so that manages timestamp token using XML database.

We describe the process of timestamp admission and verification in the following sections.

3.2.1. Process of timestamp authentication

We implemented Authentication Module. Figure 3 shows the process of timestamp admission. First, Gfarm access the gfmd to inquire the placement saving the file when a client creates a file to Gfarm. Then, Gfarm creates a file to the target FSN. Combination interface recognizes these processing. After that, admission order runs tmd to the target tsd which is created the file. Second, tsd calculates hash value of the file and admits timestamp token to the timestamp server using timestamp API. Finally, timestamp server reports the result of the admission, tsd

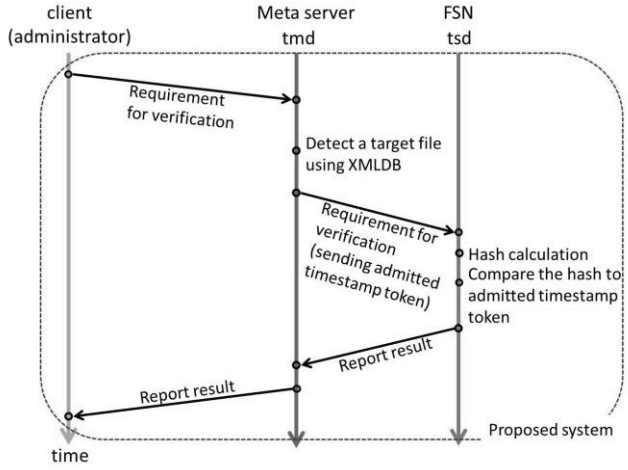

Fig.4 Process of timestamp verification.

Table 1 The format and example of Gfarm log.

Format	operation_sequence_number/second_from_unix_epoch.microsecond/ global_user/client_ip_or_hostname/client_port/**operation**/ metadb_server_host/metadb_server_port/**gfsd_hostname**/ **inode_number**/inode_generation_number1/ inode_generation_number2///"**pathname1**"///"pathname2"
Log code	Example of output log
CREATE	131908/1345723320.926539/ohgaki/gfarm-meta01/37406/ **CREATE**/gfarm-meta01/601//**29582**/5////"**/home/ohgaki/1.txt**///
REPLICATE	131909/1345723320.927305//// **REPLICATE**/gfarm-meta01/601/**gfarm-fsn01**/**29582**/5///////
UPDATEGEN	131910/1345723320.929230//// **UPDATEGEN**/gfarm-meta01/601//**29582**/5/6//////

returns the file path of the file, timestamp token, and admission time to tmd. Tmd outputs the result of the admission to the XML database. Figure 4 shows the process of timestamp verification. First, administrator runs timestamp verification, verification order runs tmd to the target tsd which is storing the file using XML database. Second, tsd calculates present hash value of the file and compare the hash value to admitted timestamp token. Finally, tsd returns the result and verification time to tmd, then complete the file verification.

4. Implementation file authentication system

We implemented combination interface, communication interface, and XML database in the Authentication Module. Timestamp API uses that SEIKO precision already developed.

4.1. Combination interface

Combination interface detects a new file creation from Gfarm. It has a function to pass required information to command processing for tmd. Table 1 shows Gfarm trace logs relating to file creation. First, when user requests to create a file, Gfarm outputs "CREATE" log which has file path. Next, when Gfarm determine a node to save the file, outputs "REPLICATE" log which has IP address for FSN.

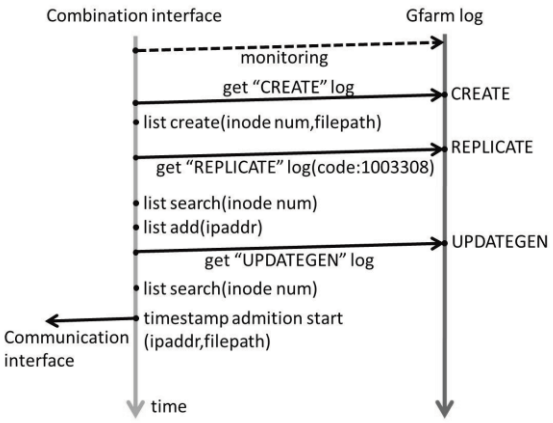

Fig.5 Process of combination interface.

Lastly, when Gfarm complete file creation, outputs "UPDATEGEN" log which is finishing the file creation. As mentioned above, Gfarm outputs log one by one for each file operation. For that reason, we can't get required information for timestamp authentication by only one log, so that we must get Gfarm trace logs together in the same file. Gfarm trace log has an inode number. Authentication Module builds a simple database whose key is inode number using queue list, and outputs to tmd for "UPDATEGEN" timing. The reason why we chose queue list is in Gfarm log that outputs by turn for one file operation, so that searching list is hit the first element.

Figure 5 shows the process of combination interface. First, add a queue list whose key is inode number and value1 is file path when Gfarm outputs "CREATE" log. Second, search the list using inode number and if hit the number, add value2 that is IP address for FSN to the hit list when Gfarm outputs "REPLICATE" log. Finally, search the list using inode number and if hit the number, delete the element from the list and start admission process. The reason why delete from the list is considering memory load.

4.2. Communication interface

Communication interface is to communicate tmd to tsd each other for timestamp admission and verification. The communication interface of tmd sends required information to the target FSN for processing timestamp admission and verification. The communication interface of tsd has 4 functions that send required information to the timestamp API, return the result for timestamp authentication to tmd, calculate a hash value for the target file, and compare the hash value to admitted timestamp token. The hash value is calculated with RSA-2048. We use the linux "diff" command to compare the hash value to ad-

```
<?xml version="1.0" encoding="utf-8"?>
<home>
  <ohgaki>
    <gfarm-disk>
      <file.txt>
        <ipaddr_or_hostname>gfarm-fsn01</ipaddr_or_hostname>
        <TStoken>/usr/local/tmd/recv/gfarm-fsn01_file.txt</TStoken>
        <admition_time>2012_08_30_16:27</admition_time>
        <verification_time>2012_08_30_03:45</verification_time>
      </file.txt>
    </gfarm-disk>
  </ohgaki>
</home>
```

Fig.6 Example of XML database.

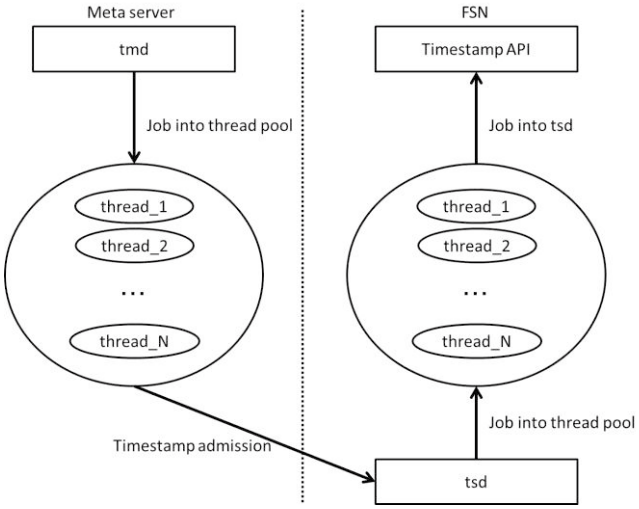

Fig.7 Thread management between tmd and tsd.

mitted timestamp token.

4.3. XML database

After a file admission, the database mapping the file and authenticated timestamp token is needed to verify time stamp token. Gfarm has a function to define file attribute information by the user using XML. Because of this, Authentication Module builds a XML database mapping a target file, authenticated timestamp token, and admission time. Figure 6 shows an example of XML database. After a timestamp admission processing, XML database includes a target file, authenticated timestamp token, and admission time from tmd. Timestamp verification process searches a target file from XML database using linux "xpath" command, and runs verification processing using required information which is file path and admitted timestamp token.

4.4. Thread management

Tmd and tsd communicate each other using socket connection. Figure 7 shows the architecture of thread management between tmd and tsd. This thread is constructed tmd of Meta server and tsd of each FSNs. When the combination interface recognized creation of a new file, timestamp admission process starts and generates a thread. Because 1 file to 1 thread is high load, we create JVM thread pool to collect threads, and each jobs are into the thread pool. The thread pool is able to change the number of job capability with configuration file. If the jobs are more than configuration number, new thread pool is created and jobs into this pool.

Table 2 Admission processing time.

Data size	Write time (sec/file)	Admission time(sec/file)	Combination Interface time(sec/file)	Total time (sec/file)
200KB	0.008581	0.01426	0.0001417	0.02298
100KB	1.5233	1.2350	0.0001567	2.7585
1GB	26.5592	55.1429	0.0001839	81.7023

5. Evaluation

5.1. Requirement of evaluation

This paper evaluates performance of combination interface in admission processing, and discusses whether the interface is bottleneck or not.

We used based on real files stored in NICT Science Cloud for this evaluation. Examples of the real files are GPS observation data (200KB/file), visualization processing data (100MB/file), and simulation data (1GB/file). NICT Science Cloud is storing 350 observation files/min, 30 visualization files/min, and 30 simulation files/min.

Our items are as follows.

・**Processing time**: performance time of each processing.

・**Processing load**: the memory and CPU load of each processing.

For processing time consists of write time which is file creation time in Gfarm, admission time by timestamp server, and combination interface time is processing time in our system.

Also, we measure a memory and CPU load of each processing using linux "top" command while admission processing.

5.2. Result

5.2.1. Processing time

Table 2 shows admission processing time for each data size. This result is average of 5 times for 1 file when admission processing runs for the number of files determined from requirement of evaluation. For result, full time for admission is 200 KB to about 0.02298 sec/file, 100 MB to about 2.7585 sec/file, 1 GB to about 81.7023 sec/file, and combination interface percentage is 200 KB to about

0.6 %, 100 MB to about 0.006 %, 1 GB to about 0.0002 %. These percentage means combination interface processing time is less than 0.0002 sec/file in any data size.

These result shows that combination interface is not overhead. The reason why combination interface processing time is very short, searching list hits at the first element.

5.2.2. Processing load

We measure a memory and CPU load for combination interface while admission processing. For result, the memory and CPU load is regardless to data size, memory load for virtual is about 3.9 MB, for physical is about 500 KB, CPU load is nearly equal to 0 %. These result shows that combination interface is not overhead. The processing load of combination interface is very small because searching list hit at the first element, so that usage of memory for the list is constant, and CPU load is very small for list searching process.

6. Conclusion

We designed file authentication system using timestamp to ensure authenticity in Gfarm and implemented an interface that combines to Gfarm and timestamp server for application level. Authentication Module built in a Meta server and FSNs provide monitoring of Gfarm log, management of timestamp token, and verification including hash calculation.

Combination interface for monitoring and analyzing Gfarm is able to process for 0.0002 sec/file regardless of data size, and CPU and memory loads are very small.

In future, we will evaluate our proposed system in detail and implement this system into NICT Science Cloud.

Bibliography

[1] O. Tatebe, K. Hiraga, N. Soda, "Gfarm Grid File System", New Generation Computing, Ohmsha, Ltd.and Springer, Vol.28, No.3, pp.257-275, 2010.

[2] K. T. Murata, S. Watari, T. Nagatsuma, M. Kunitake, H. Watanabe, K. Yamamoto, Y. Kubota, H. Kato, T. Tsugawa, K. Ukawa, K. Muranaga, E. Kimura, O. Tatebe, K. Fukazawa, and Y. Murayama, "A Science Cloud for Data Intensive Sciences", Proc. of the 1st ICSU WDS Conference on Data Science Journal, Kyoto, Japan, Sep. 2011, in press.

[3] BEST SYSTEMS, "GfarmFSE", http://www.bestsystems.co.jp/solutions/filesystems/gfarm.html

[4] Ministry of Economy, Trade and Industry, Society for the study of dependability and security of information system software in an advanced information society, "Strengthening Efforts to Enhance Dependability and Security of Information System Software ~Toward a rich, safe/secure advanced information society~ -Interim Report-", Ministry of Economy, http://www.meti.go.jp/english/press/data/20090528_01.html, May.28,2009.

[5] K. Onishi, W. Hasegawa, S. Takeuchi and H. Takasugi, "Design, Implementation and Evaluation of Data Traceability on a Large-scale Global Distributed File System for Cloud Computing", IPSJ SIG Technical Report, HPC-130(35), PP.1-8, 2011.

[6] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, IETF (2001).

[7] R. Butler, V. Wlch, D. Engert, I. Foster, S. Tuecke, J. Volmer, and C. Kesselman, "A National-scale Authentication Infrastructure", IEEE Computer Society Press, Vol.33, No.12, pp.60-66, 2000.

[8] M. Backes, A. Cortesi, R. Focardi, M. Maffei, "A calculus of challenges and responses," Proc. 5th ACM workshop on Formal methods in security engineering, pp.51-60, Fairfax, Virginia, USA, Nov. 2007.

[9] A. Sahai, H. Seyalioglu, “Worry-free encryption: functional encryption with public keys”, Proc. 17th ACM conference on Computer and communications security, pp.463-472, Chicago, Illinois, USA, Oct.2010.