Secure Hadoop with Encrypted HDFS

Seonyoung Park and Youngseok Lee

Chungnam National University, Daejeon, Republic of Korea, 305-764
{siraman,lee}@cnu.ac.kr

Abstract. As Hadoop becomes a popular distributed programming framework for processing large data on its distributed file system (HDFS), demands for secure computing and file storage grow quickly. However, the current Hadoop does not support encryption of storing HDFS blocks, which is a fundamental solution for secure Hadoop. Therefore, we propose a secure Hadoop architecture by adding encryption and decryption functions in HDFS. We have implemented secure HDFS by adding the AES encrypt/decrypt class to `CompressionCodec` in Hadoop. From experiments with a small Hadoop testbed, we have shown that the representative MapReduce job on encrypted HDFS generates affordable computation overhead less than 7%.

Keywords: Hadoop, HDFS, Security, Encryption, Decryption, Cryptography.

1 Introduction

Apache Hadoop [1], that originated from Google's MapReduce and GFS [2, 3], has been recently popularized due to its scalable distributed computing framework and file system, because it enables a big data processing platform for many data-intensive applications and analytics. Hadoop is an open-source distributed computing framework implemented in Java, and provides the MapReduce programming model and the Hadoop Distributed File System (HDFS). MapReduce allows users to harness thousands of commodity machines effectively in parallel for processing massive data in the distributed system by simply defining map and reduce functions.

Since Hadoop is usually used in a large cluster or a public cloud service such as Amazon Elastic MapReduce where multiple users run their jobs at the same time, it is essential to provide the security of user data on HDFS. However, the security service in the current Hadoop project is at the early design stage [4] that the simple file permission and access control mechanisms are employed. Particularly, encryption is the key means for the secure HDFS where many datanodes store files to HDFS and transfer user files among datanodes while executing MapReduce jobs. It is reported that a future Hadoop software release will include encryption [5]. For the secure HDFS, a few studies assume that encryption is applied to HDFS [6-9]. However, the native encryption modules for Hadoop have not been fully implemented and tested.

In general, encrypted file systems are widely deployed in various Operating Systems (OSes) such as MS Windows, Linux, MacOS and FreeBSD, and it is known that encrypted file system does not perform well because of the high CPU utilization of

J.J. Park et al. (Eds.): GPC 2013, LNCS 7861, pp. 134–141, 2013.

encryption or decryption processes. However, recent CPU architectures equipped with multi-cores and special encryption accelerators can perform better than expectation. On the other hand, due to the development of CUDA [10] and OpenCL, GPUs can run general-purpose programs by augmenting the computing power with many parallel processing units. Recent studies [11-15] begin to take advantage of GPUs for developing network systems such as routers and distributed file system by utilizing high degrees of parallelism of GPUs and saving CPU computing capacity. Yet, the current data center generally does not deploy the CUDA-capable GPUs to the cluster, because a cluster node is equipped with the popular hardware devices at the low cost. In addition, GPUs usually consume more energy than CPUs so that they cannot be adopted by a large-scale data center where the most serious problem is the energy-efficient computing infrastructure. This means that we cannot directly utilize the GPU capability for enhancing computing power in the commodity data center.

Therefore, in this paper, we propose a secure HDFS architecture that is compatible with the current Hadoop applications and show its performance results on the Hadoop cluster testbed. From the experiments with an AES encryption algorithm [16], we present that the secure HDFS causes the computation overhead only less than 7% for the representative MapReduce jobs.

The remainder of this paper is organized as follows. In Section 2, we describe the related work. Our proposal for the secure HDFS is explained in Section 3, and its experimental results are presented in Section 4. Finally Section 5 concludes this paper.

2 Related Work

As Hadoop becomes the main framework for the cloud computing service, a few studies [4, 6-9] have presented secure HDFS methods. In [4], a secure HDFS architecture has been proposed such that Kerberos over SSL is used for strong mutual authentication and access control to enhance HDFS's security. Tahoe [6], a prototype of using SSL and integrating an encrypted distributed file system with Hadoop, has been presented, but its write speed is 10 times slower and its read speed is about the same with the generic HDFS. In [7], an application-level encryption MapReduce, that assumes the pre-uploaded plaintext to HDFS, was proposed to support the file system. In [8], hybrid encryption of HDFS was proposed with HDFS-RSA and HDFS-Pairing. However, both read and write performance of encrypted HDFS is lower than those of the generic HDFS. In the write case, the encrypted HDFS is slower by 2 times. In [9], Hybrid cloud, where sensitive data is stored at private storage cluster and the remainder of data is transferred to public or partner storage cluster, was proposed. For more security, sensitive data can be encrypted using trusted platform module (TPM).

GPUs have been also applied to distributed storage systems and Hadoop. In [11], a GPU-based library that accelerates hashing-based primitives for distributed storage system has been presented. In [14], a framework for integrating GPU computing into storage systems has been proposed, and it has been prototyped in the Linux kernel. The AES cipher powered by GPUs is reported to achieve 4 GBps, whereas the results

with CPUs are less than half of GPU's performance in [14]. Shredder [15] uses GPUs for incremental storage and computation in Hadoop by mitigating the CPU bottlenecks of content-based chunking. Generally, GPU-based approaches put emphasis on performance enhancement, but they do not reveal the energy efficiency and consider form factors that are currently deployed by commodity servers in a data center. Nowadays, dedicated encryption accelerators and CPUs with special encryption features such as Intel AES New Instructions (NI) [17] have been developed.

3 Secure Hadoop

3.1 Overview

HDFS consists of a master (namenode) and multiple slaves (datanodes). In HDFS, a file is chunked by a block with the fixed size (64 MB by default). The namenode manages the file system metadata and controls access to files from clients by maintaining the mapping between datanodes and blocks for a file. Each block belonging to a file is replicated three by default in HDFS. Hadoop provides a MapReduce programming framework that runs multiple tasks for a job. A MapReduce job divides its job into multiple maps or reduce tasks to process many HDFS blocks in parallel. HDFS is well suited with a write-once-read-many access model.

We assume that every file is encrypted and decrypted before it is written and read in the secure HDFS. In addition, we presume that each datanode is a commodity server that will encrypt and decrypt files with CPUs. Clients' requests to read or write a file in HDFS will trigger decryption or encryption functions to HDFS blocks at each datanode. We use 128-bit AES which is one of the most popular block cipher algorithm and suitable for handling HDFS blocks. There are a few modes of operation for AES: ECB, CBC, OFB, CFB, CTR and XTS [16]. We choose AES ECB, because its computation can be concurrently performed in a distributed computing environment. AES CBC is the most commonly used mode, but it is not suitable for HDFS cluster consisting of many nodes because HDFS blocks must be processed sequentially on one slave node.

3.2 Encrypting Files in HDFS

As shown in Fig. 1, following the same procedure of the file write operation in HDFS, a HDFS client splits a file by a fixed size, encrypts every block and saves it to HDFS. In HDFS, the encryption function itself can be easily implemented by writing an encryption Java class in the same way that the `CompressionCodec` is used for compressing and uncompressing files [5]. Based on `CompressionCodec`, we have devised an AES encryption module (AESCodec) that executes the encryption algorithm on the CPU. The HDFS client runs AESCodec class to perform encryption and to pass the encrypted HDFS blocks to a datanode. Then, the first datanode, that receives the encrypted HDFS blocks from the client, will stream the encrypted blocks to other datanodes for replication. In contrast to decryption, encryption of a file is

performed at a HDFS client node, because the file write procedure in HDFS is sequentially carried out by a client. In this work, we have implemented only the AES function as the encryption algorithm. However, other encryption algorithms such as RSA or DES can be simply extended.

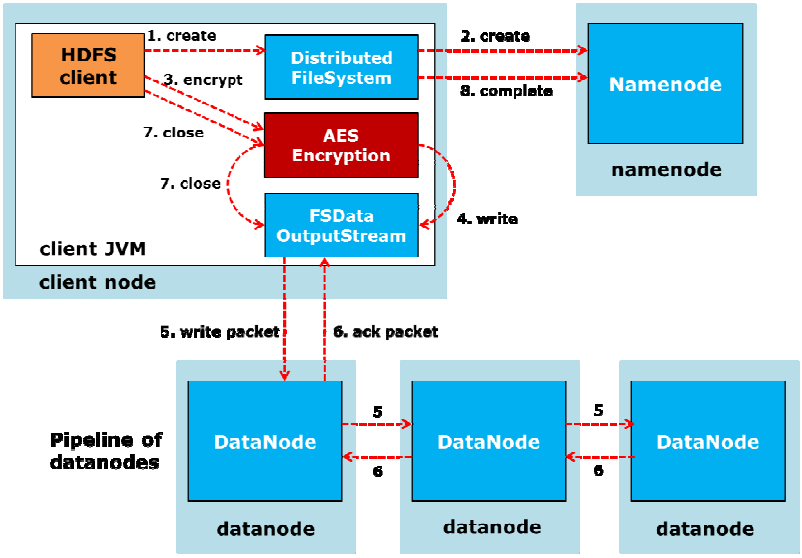

Fig. 1. Writing a file by adding an encryption step in HDFS

3.3 Decrypting Files in HDFS

With our own AESCodec, reading an encrypted file in HDFS is performed by multiple HDFS datanodes in parallel. That is, every block is processed by a map task at the HDFS datanode. Thus, assuming the same file read operation in HDFS, as shown in Fig. 2, we have added the decryption step with AESCodec when a task tracker launches a map task that reads a block. In general, multiple map tasks are executed by a Hadoop worker up to the number of available map task slots which is usually constrained by the number of CPU cores. Since HDFS assumes the write-once-read-many model, our concurrent decryption per-HDFS block architecture fits well for various MapReduce jobs.

4 Performance Evaluation

4.1 Experimental Environment

For the performance evaluation of encrypted HDFS, we have established a small Hadoop testbed consisting of a master node and three worker nodes. Each node has 8- core 2.83 GHz CPU, 4 GB memory, and 2 TB hard disk. All Hadoop nodes are connected with1 Gigabit Ethernet cards.

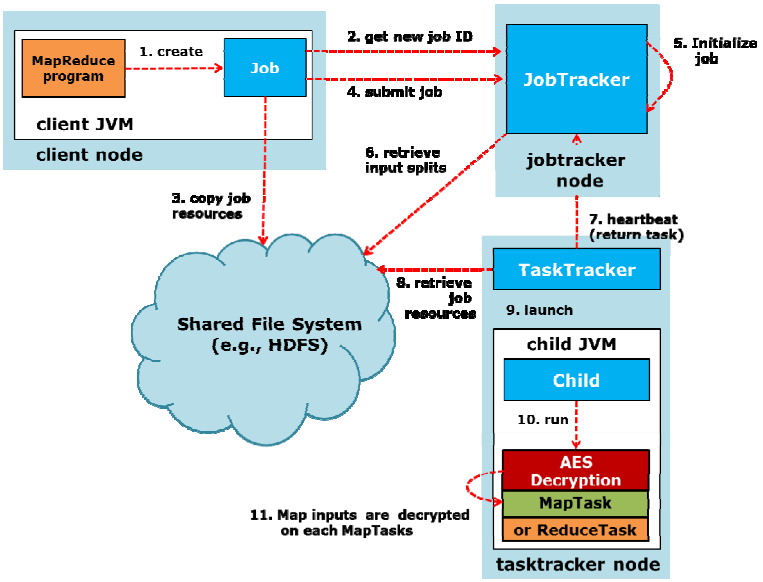

Fig. 2. A MapReduce job that read an encrypted file in HDFS

4.2 File Write: Generic HDFS vs. Encrypted HDFS

First, we have compared the file write time between generic HDFS and encrypted HDFS. As shown in Fig. 3, it took 430 minutes to write a 1 TB unencrypted file to HDFS, whereas 585 minutes to encrypt a file in HDFS, which is 36% performance degradation. The throughput of writing files in HDFS is 41 MB/s for the generic HDFS, while 30 MB/s for the AES encrypted HDFS. In the encryption phase, the HDFS client is in charge of encrypting the whole files, which is a bottleneck of uploading files to HDFS.

4.3 File Read and Computation of MapReduce Jobs: Generic HDFS vs. Encrypted HDFS

In order to evaluate the usefulness of encrypted HDFS, we have considered MapReduce jobs on multiple HDFS datanodes that read and compute encrypted files in HDFS. Thus, we have run a representative WordCount MapReduce job on the testbed that processes encrypted files on HDFS by decrypting files with AESCodec. Fig. 4 shows the performance of MapReduce jobs on unencrypted or encrypted HDFS. We can observe that 604 minutes was taken for running a WordCount MapReduce job for unencrypted HDFS for 1 TB file tests, while 635 minutes for the encrypted HDFS. In the read case, the overhead of decrypting files in HDFS is less than or equal to 5% for almost all cases except 7% for 128 GB. In contrast to the write case, the decrypt/read operation on encrypted files is executed in parallel, which mitigates the performance degradation, and it fits well for the write-once-read-many model.

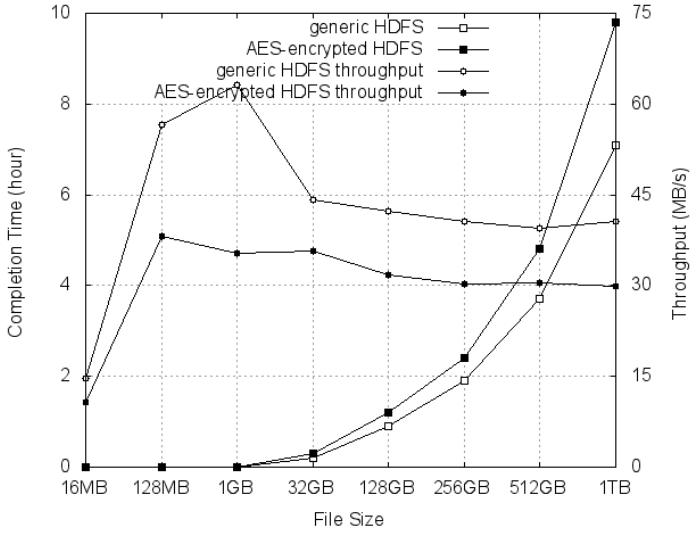

Fig. 3. File write performance under different file sizes: generic HDFS vs. AES-encrypted HDFS

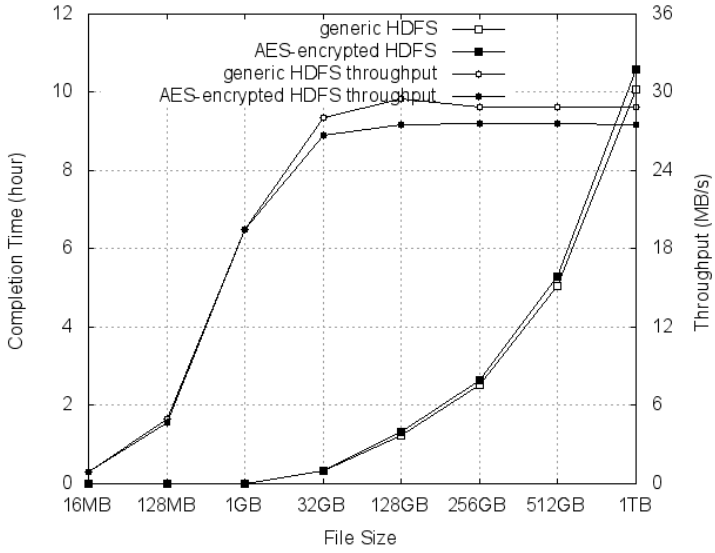

Fig. 4. Performance of MapReduce jobs under different file sizes: generic HDFS vs. AES-encrypted HDFS

5 Conclusion

Since generic Hadoop lacks in secure file management, it is necessary to be upgraded with encryption in HDFS. Though encryption is the essential file protection method, its real implementation has not been fully examined. In this paper, we presented a secure HDFS by adding encryption and decryption function as a built-in encryption/decryption class in Hadoop. Based on `CompressionCodec`, we have implemented AESCodec into Hadoop and shown that it is useful for securing MapReduce job in HDFS with marginal performance degradation less than 7%.

Acknowledgment. This research was supported by the Basic Science Research Program through the NRF, funded by the Ministry of Education, Science and Technology (NRF 2010-0021087). The corresponding author of this paper is Youngseok Lee.

References

1. Hadoop, `http://hadoop.apache.org/`
2. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Cluster. In: OSDI (2004)
3. Ghemawat, S., Gobioff, H., Leung, S.: The Google File System. In: ACM Symposium on Operating Systems Principles (October 2003)
4. O'Malley, O., Zhang, K., Radia, S., Marti, R., Harrell, C.: Hadoop Security Design, Technical Report (October 2009)
5. White, T.: Hadoop: The Definitive Guide, 1st edn. O'Reilly Media (2009)
6. Cordova, A.: MapReduce over Tahoe. Hadoop World (2009)
7. Majors, J.H.: Secdoop: a confidentiality service on Hadoop clusters. Auburn University Master Thesis (May 2011)
8. Lin, H., Seh, S., Tzeng, W., Lin, B.P.: Toward Data Confidentiality via Integrating Hybrid Encryption Schemes and Hadoop Distributed FileSystem. In: IEEE AINA (2012)
9. Yang, Y., Wu, Z., Yang, X., Zhang, L., Yu, X., Lao, Z., Wang, D., Long, M.: SAPSC: Security Architecture of Private Storage Cloud Based on HDFS. In: Proceedings of 26th IEEE Workshops of International Conference on Advanced Information Networking and Applications (2012)
10. NVIDIA CUDA Programming Guide, `http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_ProgrammingGuide.pdf`
11. Al-Kiswany, S., Gharaibeh, A., Santos-Neto, E., Yuan, G., Ripeanu, M.: StoreGPU: exploiting graphics processing units to accelerate distributed storage systems. In: ACM HPDC (2008)
12. Han, S., Jang, K., Park, K., Moon, S.: PacketShader: A GPU accelerated Software Router. In: Proceedings of the ACM SIGCOMM (2010)
13. Jang, K., Han, S., Han, S., Moon, S., Park, K.: SSLShader: Cheap SSL Acceleration with Commodity Processors. In: Proceedings of NSDI (2011)
14. Sun, W., Ricci, R., Curry, M.L.: GPUstore: Harnessing GPU Computing for Storage Systems in the OS Kernel. In: ACM SYSTOR (June 2012)

15. Bhatotia, P., Rodrigues, R., Verma, A.: Shredder: GPU-Accelerated Incremental Storage and Computation. In: USENIX FAST (February 2012)
16. Advanced Encryption Standard, http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
17. Intel, http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni