KIPS Tr. Comp. and Comm. Sys.
Vol.2, No.12 pp.569~576 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2013.2.12.569

A Study on Data Storage and Recovery in Hadoop Environment

Su-Hyun Kim† · Im-Yeong Lee††

ABSTRACT

Cloud computing has been receiving increasing attention recently. Despite this attention, security is the main problem that still needs to be addressed for cloud computing. In general, a cloud computing environment protects data by using distributed servers for data storage. When the amount of data is too high, however, different pieces of a secret key (if used) may be divided among hundreds of distributed servers. Thus, the management of a distributed server may be very difficult simply in terms of its authentication, encryption, and decryption processes, which incur vast overheads. In this paper, we proposed a efficiently data storage and recovery scheme using XOR and RAID in Hadoop environment.

Keywords : HDFS, Distributed Server, XOR, RAID

하둡 환경에 적합한 데이터 저장 및 복원 기법에 관한 연구

김 수 현† · 이 임 영††

요 약

최근 많은 관심을 받고 있는 클라우드 컴퓨팅이 해결해야할 가장 큰 문제는 바로 보안이다. 일반적인 클라우드 컴퓨팅 환경에서는 사용자의 데이터 보호를 위해 수많은 분산서버를 이용하여 데이터를 저장한다. 하지만 분산서버에 저장된 데이터를 암호화 과정을 거치지 않고 그대로 저장하게 된다면, 마스터 서버에 저장된 분산파일 위치를 추적하여 그대로 유출이 되는 문제가 발생할 수 있다. 이러한 문제를 방지하기 위해 비밀키를 이용하여 분산된 데이터를 암호화해야할 필요성이 존재한다. 그러나 대용량 데이터의 경우 수십, 수백 개의 조각으로 나누어지게 되는데 분산서버마다 각각의 비밀키를 이용하게 된다면, 관리의 어려움이 존재할 뿐 아니라 분산 서버에 대한 정당한 인증, 암복호화 과정을 수없이 거치게 되어 막대한 오버헤드가 발생하게 된다. 따라서 본 논문에서는 이와 같은 문제점을 해결하기 위해 Hadoop 환경에 적합한 XOR 및 RAID기반의 효율적인 분산 저장 및 복구 기법을 제안하였다.

키워드 : 하둡, 분산 서버, XOR, RAID

1. 서 론

최근 국내외로 클라우드 컴퓨팅에 관한 관심이 높아지며 많은 연구가 진행되고 있다. 많은 기업들이 IT기술의 성장을 발판으로 다양한 분야로 확장 가능하고, 컴퓨팅 파워의 효율적인 사용이 가능한 클라우드 컴퓨팅에 관심을 가지고 있다. 최근 수많은 인터넷 서비스 업체들은 인터넷 서비스 플랫폼의 중요성을 인식하고 자체적으로 연구를 수행하여, 저가 노드를 기반으로 한 대규모 클러스터 기반의 클라우드 컴퓨팅 기술을 개발 활용하고 있다[1]. 이러한 클라우드 컴퓨팅 환경에서는 사용자의 데이터를 수많은 분산 서버를 이용하여 저장, 보관하게 된다. 이와 같은 분산 컴퓨팅 환경에서는 사용자의 대용량 데이터를 관리할 수 있는 분산 관리가 주요 이슈로 떠오르고 있다. 하지만, 대용량 데이터의 저장, 보관과 같은 다양한 이용 형태에서 악의적인 공격자나 내부 사용자에 의한 보안 침해 및 데이터 손실에 대해 적절한 방안이 없는 실정이다. 따라서 본 논문에서는 이와 같은 문제점을 해결하기 위해 다양한 환경에 적합한 XOR 및 RAID기반의 효율적인 분산 저장 및 복구 기법을 제안하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안하는 기법의 이해를 돕기 위한 기존 연구들을 소개하고, 3장에서는 클라우드 컴퓨팅 환경이 갖추어야 할 기본적인 보안 요구사항에 대하여 알아보고, 4장에서는 제안 방식에 대하여 설명한다. 5장에서는 제안 방식의 안전성을 분석한

※ This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center)) support program (NIPA-2013-H0301-13-1003) supervised by the NIPA(National IT Industry Promotion Agency).
※ This work was supported by the Soonchunhyang University Research Fund.
† 준 회 원 : 순천향대학교 컴퓨터학과 박사과정
†† 종신회원 : 순천향대학교 컴퓨터학부 교수
논문접수 : 2013년 10월 14일
심사완료 : 2013년 11월 5일
* Corresponding Author : Im-Yeong Lee(imylee@sch.ac.kr)

고, 마지막으로 6장에서는 결론 및 향후 연구 방향으로 마치도록 한다.

2. 관련 연구

본 장에서는 본 논문에서 제안하는 기법의 이해를 돕기 위한 관련 기술과 기존 제안방식들을 소개한다.

2.1 Google File System(GFS)

구글에서 사용하는 구글 파일 시스템은 대용량의 데이터에 적합하도록 개발되었으며 핵심 데이터 저장 및 검색 엔진에 최적화 되어 있다[2]. GFS는 대용량의 데이터를 저장하기 위해 수많은 저비용 스토리지 서버를 사용하여 분산 저장하게 된다. GFS의 구조는 마스터 서버와 청크 서버로 구성이 되며, 데이터는 다수의 청크 서버에 64메가바이트 단위로 나눠진다. 마스터 서버에는 각 청크 서버의 생성 시점에 고유의 64비트 레이블을 할당하고, 논리적 매핑을 이용해 청크 서버와 연결을 유지한다. 하지만 다수의 청크 서버를 사용함으로써 빈번하게 발생할 수 있는 장애를 대처할 수 있는 방안을 필요로 하는 단점이 존재한다.

2.2 Apache Hadoop Distributed File System(HDFS)

HDFS는 기성 하드웨어에서 실행 가능하도록 제작된 파일 시스템으로 기존의 분산 파일 시스템과 많은 유사점을 가지고 있다. 하지만 많은 차이점도 보이는데 높은 장애복구 기능과 저가의 하드웨어에 적용이 가능하도록 설계되었다. HDFS는 Amazon의 EC2, Yahoo 등 다양한 IT기업들의 클라우드 컴퓨팅 플랫폼으로 가장 많이 활용 되고 있다[3].

HDFS의 설계와 구현을 위한 구조는 GFS와 대부분 동일하며 다른 점은 자바를 사용하여 다양한 플랫폼에 이식성이 뛰어나다는 점이 크게 다르다. 높은 이식성을 지닌 자바 언어의 사용은 자바를 지원하는 다양한 서버들에서 HDFS가 구동할 수 있다는 장점을 지닌다[4].

2.3 HDFS에서의 데이터 저장 및 복구

HDFS클라이언트는 사용자의 데이터를 128MB 블록 단위로 분할하여 NameNode에 저장 요청을 하게 된다. NameNode는 데이터 블록이 저장될 DataNode의 주소를 파악하여 다시 클라이언트에게 보내주게 된다. 3개의 DataNode주소를 받은 클라이언트는 첫 번째 DataNode에 직접데이터를 전송하게 되고, 첫 번째 DataNode는 데이터 수신과 동시에 복사 본을 다음 DataNode에 송신하게 된다. 이렇게 하나의 블록은 데이터의 손실이나, DataNode의 고장 및 오류에 대비하기 우해 총 3개의 DataNode에 백업되어 저장된다(Fig. 1).

2.4 Cloud Shredder

1) 기본 개념

Cloud Shredder은 2011년도에 기존 데이터 암호 방법인 FDE(Full Disk Encryption)의 문제점을 해결하기 위해 제안되었다[5]. Cloud Shredder에서는 클라우드 환경에 저장된 데이터는 서비스 제공업체에 의해 암호를 해독할 수 있는 가능성이 존재함에 따라, 안전하지 않다고 간주하고 있다. 또한, FDE에 의해 저장된 데이터는 공격자에 의해 노출 될 경우, 아주 우수한 성능의 컴퓨팅 환경이 갖추어 진다면 전수조사 공격이 충분히 가능하기 때문에 Cloud Shredder는 데이터가 유출되더라도 해당 데이터를 쓸모없는 데이터로 만드는 것을 목표로 하고 있다.

2) 데이터 분산 및 복원

위와 같은 조건을 충족시키기 위해 Cloud Shredder는 XOR-method 또는 Ratio-method를 이용해 데이터를 LS(Local Share)와 RS(Remote Share)로 분할하여 저장한다. 분할된 데이터 LS와 RS는 각각 소유자의 로컬장치와 클라우드 서버에 저장되어 둘 중 하나라도 없다면 복원 불가능하다. 하지만 LS는 데이터 소유자의 로컬 장치에 저장

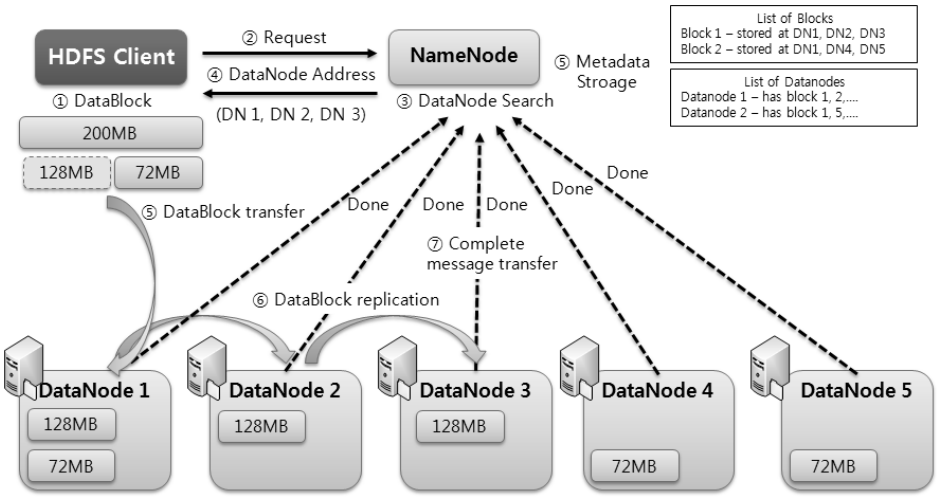


Fig. 1. Writing data in HDFS cluster

되므로 항상 소지하고 있어야 하므로, 클라우드 서비스의 높은 가용성을 활용할 수 없다는 단점이 존재한다.

a) XOR-method

XOR기반의 데이터 분산 방식은 원본 메시지와 같은 길이의 랜덤 바이트 스트림 생성으로 시작한다. 원본 메시지 바이트 스트림과 새롭게 생성된 랜덤 바이트 스트림을 XOR 연산을 수행하고, 그 결과 값인 새로운 바이트 스트림(share R)을 클라우드에 저장한다. 맨 처음 생성한 랜덤 바이트 스트림(share L)은 Local에 저장한다. 데이터의 복원은 클라우드에 저장된 share R과 local에 사용자가 보관하고 있는 share L을 XOR연산을 통해 원본 메시지를 복구하게 된다. 이 방식은 원본 메시지와 동일한 길이의 일회용 암호로 원본 메시지를 암호화 하는 것으로 보안 수준이 매우 높지만, 성능 저하의 단점은 무시할 수 없다. 즉, 보안 강도는 높으나, 성능저하의 우려가 있으며, 실제 파일의 크기가 큰 경우 전송시간이 많이 소모된다는 단점이 있다.

b) Ratio-method

Ratio기반의 분산 방식은 원본 메시지 X를 함수 F를 통해 X′으로 변환한다. 그 후, X′을 서로 다른 길이로 조각을 내며, 짧은 길이의 데이터는 share L, 긴 길이의 데이터는 share R로 정하여 각각 클라우드와 Local에 저장한다. 이 방식은 원본 메시지보다 짧은 길이의 데이터를 전송하여 XOR방식보다 전송 속도는 빠르다.

나누어지는 데이터 블록의 크기는 원격비율(Remote Ratio:RR)과 실제 파일 크기로 정의한다. RR=1/N으로 가정하면, 원격 공유조각처럼 모든 N블록을 모아서 조립한다. L byte의 파일 X의 경우, 원격 공유조각은 L/N byte 이상이 된다. 예를 들면, 100byte 원본파일 X가 있다면, 이 파일을 2조각으로 나누었을 경우 파일 한 조각의 크기는 100/2가 되므로, 한 조각이 50byte 이상이 되어야 한다.

데이터 복원 시, 데이터 조각들을 모두 모은 후 함수 F의 역함수를 이용하여 복원한다. 이때 함수 F-1은 다음 효과를 만족시켜야 한다. X를 복구하기 위해서는 X′ 전체가 필요하다. 이는 블록암호의 '확산'속성의 특징으로, 함수 F는 대칭 암호알고리즘을 사용한다.

2.5 개선된 Cloud Shredder 방식

2013년 park 등은 기존의 Cloud Shredder 방식의 문제점을 해결하고자 새로운 방식을 제안하였다[7]. 기존 Cloud Shredder에서는 원본 데이터를 나눈 조각 중 하나는 데이터 소유자의 로컬 장치에 저장되므로 항상 소지하고 있어야 하기 때문에, 클라우드 서비스의 높은 가용성을 활용할 수 없다는 단점이 있다.

위와 같은 단점을 보완한 park의 개선된 Cloud Shredder 방식에서는 비밀분산 방식과 클라우드 서비스를 동시에 사용하였다. 하나의 파일을 다수의 조각으로 분할하고, 분할된 조각들을 다양한 클라우드 서비스에 저장하여 기존 Cloud Shredder방식의 문제점이었던 가용성과 기밀성을 동시에 제공하고자 하였다. 또한, 여러 개의 클라우드 서비스를 사용함으로써 기존 클라우드 서비스의 취약점으로 인한 보안사고 발생 가능성에 대해 피해를 최소화 할 수 있다고 하였다. 하지만, 데이터의 조각 개수가 수십, 수백 개로 나눠진다면 그 조각 개수만큼의 각각 다른 클라우드 서비스를 사용해야 하기 때문에 현실적으로 적용이 쉽지 않다는 단점이 있다.

3. 보안요구사항

클라우드 컴퓨팅의 핵심 메커니즘 중 하나는 바로 대용량 데이터의 효율적인 관리이다. 대용량 데이터의 다양한 이용 형태로부터 악의적인 공격자나 내부 사용자에 의한 보안 취약성 및 프라이버시 침해가 발생할 수 있기 때문에, 다음과 같은 보안 요구사항을 필요로 하게 된다[8].

- 기밀성 : 데이터 저장 서버와 클라이언트 단말기 간의 통신 데이터는 정당한 개체만이 확인 가능해야 한다.
- 인증 : 데이터 저장 서버가 정당한 개체인지 검증 가능해야 하고, 정당한 사용자만이 데이터에 대한 접근을 가능하게 해야 한다.
- 가용성 : 대용량의 데이터를 전송 시 가용성을 보장하기 위해 인증 및 기밀성이 빠른 속도로 이루어 져야 한다.
- 연산효율성 : 수시로 이루어지는 데이터 전송 시 클라이언트 단말기 및 클라우드 서버의 오버헤드를 줄이기 위해 최소한의 연산만을 보장해야 한다.

4. 제안방식

4.1 시스템 모델 및 가정

전체적인 클라우드 컴퓨팅 환경은 Apach의 HDFS를 바탕으로 설계되었다. 클라우드 컴퓨팅 환경에서는 대용량 데이터 저장을 위해 128MB 블록 단위로 나누어 분산 저장하게 된다. 각각 분산 저장서버에 저장된 데이터는 암호화 되지 않은 평문상태로 저장되어 있어 공격자에 의해 분산 저장 서버가 탈취되는 경우 데이터 일부가 그대로 노출되는 문제점이 발생한다. 이를 방지하기 위해 암호화를 사용자에게 권고하지만, 시스템 자체적으로는 제공되지 않고 있다. 따라서 본 논문에서는 분할된 데이터의 일부 블록만 유지되어도 원본 데이터가 복구 가능한 XOR 및 RAID기반의 데이터 복구 기법을 제안한다.

4.2 XOR 기반 데이터 분산 저장 과정

1) 데이터 분산 과정

XOR 기반 데이터 분산 저장 방식에서는 블록암호화 모드 중 하나인 CBC모드의 Channing을 응용하여 원본 데이터 블록을 임의의 데이터 블록으로 변환한다(Fig. 2)[9]. 변환된 데이터 블록에 일정 패리티 블록이 추가로 생성이 되는데 이 패리티 블록은 추후 원본 데이터 블록에 오류가 발

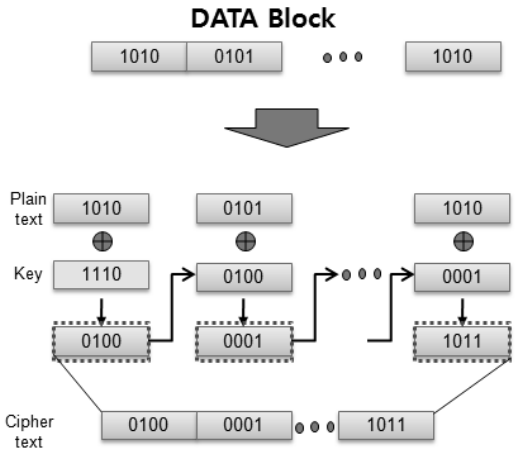


Fig. 2. Data distribution process using Chaning of CBC mode

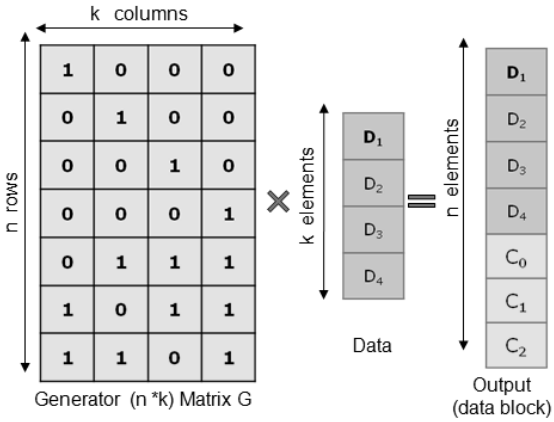


Fig. 3. Production rules of parity blocks

생하였을 시에 원본 데이터 블록을 복구하는 역할을 수행한다(Fig. 3).

패리티 블록의 생성 규칙은 (식 1)과 같다.

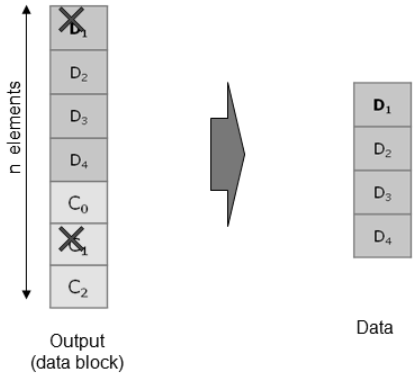
$$C_i = D_{0\sim n} \oplus D_i \quad (1)$$
$$\cdots$$
$$C_{n-1} = D_{0\sim n} \oplus D_{n-1}$$

-C1 생성 과정

$$C_0 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_0 = D_1 \oplus D_2 \oplus D_3$$
$$C_1 = D_0 \oplus D_2 \oplus D_3$$
$$C_2 = D_0 \oplus D_1 \oplus D_3$$

2) 데이터 복구 과정

위와 같이 생성된 패리티 블록은 원본 데이터 블록에 오류가 발생 했을 시 복구하는 역할을 하게 된다. 임의의 패리티 블록을 생성하였을 시 데이터 블록을 복구하는 과정은 아래와 같다. 각각 생성된 패리티 블록과 원본 데이터에 오류가 발생하더라도 (Fig. 4)와 같이 복구가 가능하다.



$$C_0 = D_2 \oplus D_3 \oplus D_4$$
$$C_1 = D_1 \oplus D_3 \oplus D_4$$
$$C_2 = D_1 \oplus D_2 \oplus D_4$$
$$D_1 = C_1 \oplus D_2 \oplus D_4$$

Fig. 4. Recovery process of the data block

-D1 복구 과정

$$D_1 = C_1 \oplus D_1 \oplus D_4$$

4.3 RAID 기반의 데이터 분산 저장

1) 시스템 계수

· $h : 0,1^{l(s-1)} \rightarrow 0,1^{l}$

· $g : 0,1^{l} \rightarrow 0,1^{l(s-1)}$

· s : 평문 블록 수

· $m_1, m_2, ..., m_s (m_i \in 0,1^{l})$: 평문 블록

· $x_1, x_2, ..., x_s$: 의사평문블록

2) AONT 분할 과정

n개의 조각으로 나누어진 데이터 블록은 분산 저장 시 데이터의 안전성을 증가시키기 위하여 AONT 분할 과정을 거친다(Fig. 5)[10]. AONT변환의 안전성은 h(hash function), g(pseudo-random number generator)수의 안전성을 전제로 한다.

Step 1. AONT 변환을 위해 평문 데이터 m을 $m_1, m_2, ..., m_s (m_i \in 0,1^{l})$로 분할한다. 그리고 h함수를 이용하여 μ_s를 계산한다.

$$- \ \mu_s = h(m_1 || m_2 || ... || m_{s-1})$$

Step 2. 계산된 μ_s와 ms 그리고 g함수를 이용하여 다음과 같이 계산한다.

$$x_1 || x_2 || ... || x_{s-1} = (m_1 || m_2 || ... || m_{s-1}) \oplus g(\mu_s \oplus m_s)$$

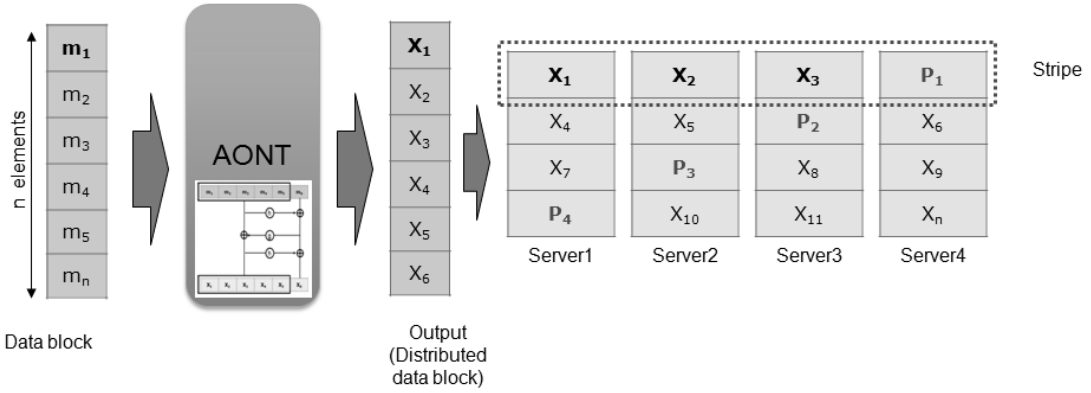


Fig. 5. Distributed data storage based on RAID

Step 3. 다음으로 x_s를 아래와 같이 계산하여 의사평문 블록을 생성한다.

$$x_s = (\mu_s \oplus m_s) \oplus h(x_1 \| x_s \| ... \| x_{s-1})$$

3) RAID-5기반 데이터 분산 저장 과정

Fig. 5에서 각 데이터가 교차되어 모인 열을 Stripe라 한다. 패리티 데이터는 각각 교차되어 다른 Stripe에 위차하게 되며, 이러한 방법을 Parity Rotation이라 한다. 패리티 데이터는 다른 모든 데이터 조각의 XOR연산 결과치가 된다.

- $P_1 = X_1 \oplus X_2 \oplus X_3$

4개의 서버로 구성된 Fig. 5에서 세 번째 서버에 장애가 발생했다고 가정하면, 남아있는 모든 데이터 항목을 XOR연산을 수행하여 손실된 데이터 값을 찾게 된다.

- $X_3 = X_1 \oplus X_2 \oplus P_1$

$P_2 = X_4 \oplus X_5 \oplus X_6$

$X_8 = X_7 \oplus P_3 \oplus X_9$

$X_{11} = P_4 \oplus X_{10} \oplus X_{12}$

5. 제안방식 분석

본 장에서는 두 개의 XOR 방식과 RAID 방식을 통해 각 방식의 장단점 및 기존방식과의 차이점을 분석하여 본다.

5.1 XOR기반 제안 방식

1) 인증 및 기밀성

사용자의 데이터는 마스터 서버를 통하여 분산서버에 여러 조각으로 나뉘어 저장된다. 사용자로부터 데이터 요청이 들어왔을 경우 분산되어 있는 데이터를 수집하는 과정이 필요하다. 이 때, 분산 서버는 마스터 서버와 물리적, 논리적으로 분리되어 있기 때문에, 각각의 분산서버에 대한 인증과 조각 데이터 전송 시 통신로 상에서의 기밀성을 제공해야 한다. 하둡 1.0 버전 이후부터 Kerberos를 지원하여 사용자의 접근권한을 중앙집중적으로 관리하여 사용자에 대한 인증을 수행 하고 있다. 하지만 별도의 데이터 암호과정은 지원하고 있지 않기 때문에, 본 논문에서 제안하는 XOR 및 RAID기반의 데이터 저장 방식을 사용한다면 기존의 암호 시스템보다 비용적, 연산적 측면에서 매우 효율적이라고 할 수 있다.

2) 저장 공간의 효율성

기존의 클라우드 시스템은 전체 데이터 n개의 3배인 3n 만큼의 저장 공간을 필요로 하지만 XOR기반 데이터 분산 저장방식은 2n-1 만큼의 저장 공간을 필요로 한다. Cloud Shredder의 경우 HDFS를 기반으로 구현한다고 가정하였을 경우, 이는 약 29% 이상의 공간 효율성을 보장 할 수 있다 (Fig. 6).

또한, 데이터의 블록 용량은 500kB로 고정되어 있고, 데이터 조각의 개수를 유동적으로 변화하여 이에 따른 공간

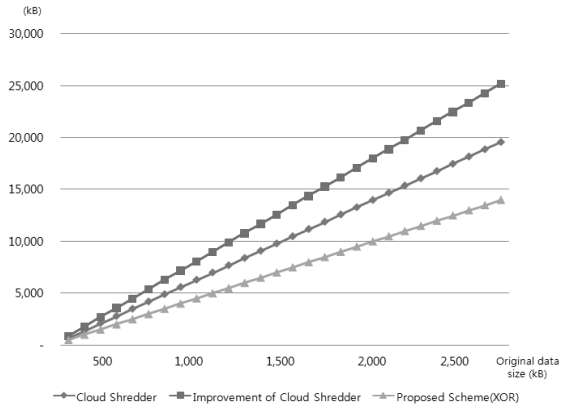


Fig. 6. Changing the storage space by size of data blocks(number of data blocks=3)

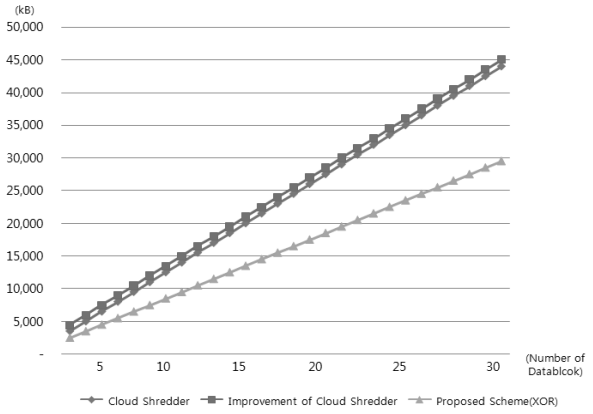


Fig. 7. Changing the storage space by number of data blocks(size of data blocks=500kB)

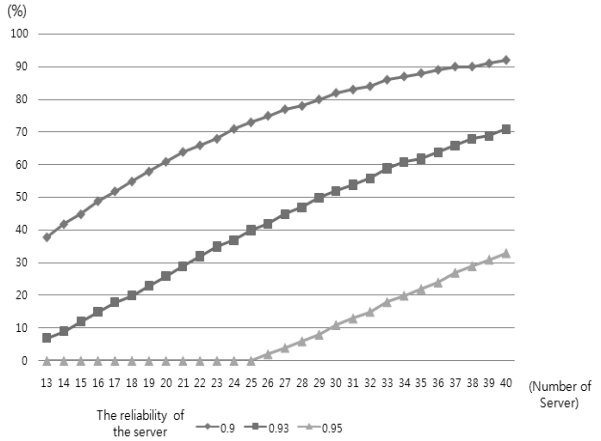


Fig. 8. Probability of data loss in three years

소모비율을 비교하였다(Fig. 7). 이는 원본 데이터 용량이 클수록 Cloud Shredder 방식은 보다 많은 클라우드 서비스를 이용하기 때문에 제안 방식이 Cloud Shredder과 비교하여 약 33% 이상의 공간 효율성을 보장 한다.

5.2 RAID 기반 시스템에서의 데이터 손실 확률

Fig. 8은 RAID-5기반의 시스템에서 3년 내에 데이터 손실이 발생할 확률을 나타내는 표이다. 13개의 서버 중 하나의 서버에서 데이터 손실이 발생할 확률은 약 38%로 굉장히 높은 수치를 보여주고 있다.

데이터 손실률을 계산하기 위한 수식은 (식 2)와 같다[11]. 이 식에서 사용된 계수는 다음과 같다.

- j : 데이터용 디스크 개수
- n : 총 디스크 개수
- RHDD : 디스크 신뢰도

$$R_{array} = \sum_{j=k}^{n} \binom{n}{j} R_{HDD}^{j} (1 - R_{HDD})^{(n-j)} \quad (2)$$

(식 2)를 기반으로 디스크의 신뢰도를 0.9로 가정한 후, 13개의 디스크를 RAID-5로 구성할 경우, 13개 중 12개가 데이터용, 나머지 하나의 디스크가 패리티용 디스크로 간주하여 계산하게 된다. 이 시스템의 신뢰성 계산은 아래와 같다.

$$R_{array} = \frac{13!}{12!(13-12)!} 0.9^{12} (1-0.9)^{(13-12)} + \frac{13!}{13!(13-13)!} 0.9^{12} (1-0.9)^{(13-13)}$$

$$= 0.3672 + 0.2542 = 0.6213$$

즉, 3년 이상 데이터 손실이 발생하지 않은 확률이 약 62%라는 것을 나타낸다. 거꾸로 말해, 같은 기간에 데이터 손실이 발생할 확률이 38%라는 것이 된다.

이처럼 RAID-5기반의 시스템에서는 서버의 오류를 고려하고 있기 때문에 전체 시스템에 큰 영향을 끼치진 않지만 2개 이상의 서버에서 오류가 발생할 확률을 고려하여 추가적인 연구가 필요하다.

5.3 기존방식과의 비교 분석

본 논문에서 제안한 XOR 및 RAID 기반의 두 가지 방식은 모두 데이터를 안전하고 효율적으로 관리 할 수 있다. 기존의 Cloud Shredder과 개선된 Cloud Shredder 방식과 다르게 데이터 블록에 오류가 발생하더라도 원본 데이터 블록을 복구할 수 있다(Table 1). 데이터의 분할 과정에서는 Cloud Shredder 방식에 비해 연산량은 많을 수 있으나, 고성능 기반의 클라우드 기반의 서비스에서는 무시할 수 있을 만큼의 차이를 보이고 있다.

클라우드 서비스를 사용하는 기업이나 개인의 측면에서 가장 우려되는 부분이 바로 서비스 제공업체에서 발생할 수 있는 보안ㅋ사고이다. Cloud Shredder과 개선된 Cloud Shredder방식은 데이터 조각을 하나의 서비스업체가 아닌 개인과 클라우드 서버 혹은 서로 다른 클라우드 서버에 저장을 함으로써 하나의 클라우드 서비스 업체에서 발생한 보안 사고에 대해서는 사용자의 데이터가 유출될 가능성은 거의 없다. 하지만, XOR기반의 제안방식의 경우 확률 상으로 극히 드물지만, 원본데이터가 분할된 데이터 조각들의 수집이 가능하다면 복원이 가능하다.

데이터의 가용성 측면에서는 Cloud Shredder 방식의 경우, 반드시 사용자가 데이터 조각을 소유하고 있어야 원본 데이터를 복원 할 수 있었다. 이는 클라우드 서비스의 가장 큰 특징인 가용성이 낮아지는 결과를 초래한다. 반면에 개선된 Cloud Shredder 방식에서는 위와 같은 데이터 가용성 측면에서 사용자가 어디서든지 클라우드 서비스를 이용할 수 있도록 하기 위해, 사용자가 별도로 보관해야 하는 데이터 조각을 모두 클라우드 서비스를 통해 맡기는 방식을 제안하였다. 하지만 이 방식에서는 각각의 데이터 조각들을 하나의 클라우드 서비스가 아닌 모두 다른 서비스를 이용함

Table 1. Comparison with existing methods

	Cloud Shredder		Improvement of Cloud Shredder		Proposed Scheme (XOR)	Proposed Scheme (RAID)
Data Recovery	X		X		O	O
Data Splitting (3 piece)	XOR	Ratio	XOR	Ratio	4E	4E+B
	2E	2B	2E	2B		
The cloud service provider's security incident	X (Requires RS)		X (Require other pieces of the data)		△ (Require other pieces of the data)	X (AONT based Encrypted)
Availability of data	△ (Must have LS)		△ (Requires a cloud server of large number)		O (Fault tolerant system)	O (Fault tolerant system)
Storage	(n-1)(3m)+m		n(3m)		(2m-1)n	(3m-1)n

n : Data capacity, m : Number of blocks distributed, E : XOR operations, B : Bit operation

으로써, 데이터 조각의 개수만큼 클라우드 서비스를 사용해야 한다는 단점이 존재한다. 즉, 데이터의 조각이 수십 개만 넘어간다 하더라도 이용해야 하는 클라우드 서비스의 종류 또한 수십 개가 되고, 이에 관리해야 하는 ID와 패스워드도 수십 개에 달하므로 사용자 측면에서 관리의 어려움을 불러올 수 있다. 하지만 XOR 및 RAID 기반의 제안 방식에서는 하나의 클라우드 서비스를 이용하되, CBC 및 AONT기반의 데이터 블록 분할을 통해 원본 데이터를 유추 할 수 없도록 만들고, 패리티 블록을 추가하여 데이터 블록의 오류에 대응하여 원본데이터를 복원 할 수 있도록 제안하였다.

6. 결 론

본 논문에서는 대용량 데이터의 다양한 이용 형태로부터 악의적인 공격자나 내부 사용자에 의한 보안 취약성 및 프라이버시 침해를 방지하기 위해 XOR 및 RAID기반의 효율적인 분산 저장 및 복구 기법을 제안하였다. 전체 시스템의 과부하를 줄이기 위해 별도의 데이터 암호화 과정 없이 추측 불가능한 데이터 조각을 나누어 정당한 사용자만이 복구 가능하도록 제안하였다. 또한, 데이터 블록의 오류가 발생했을 경우, 원본 데이터를 복구할 수 있는 방법을 제공함으로써, 기존 방식들이 제공하지 못했던 문제도 해결할 수 있었다. 이를 바탕으로 제안한 방식은 안전성을 향상시키고 효율성 측면에서 XOR 연산 기반의 비밀분산 방식을 이용하였다. 제안 프로토콜은 기밀성이 높은 데이터, 사용자의 개인정보를 포함하는 다양한 대용량 데이터를 안전하고 효율적으로 분산 관리하는 구조로서 클라우드 컴퓨팅 환경에서 보다 효율적으로 사용될 것으로 기대한다.

참 고 문 헌

[1] Stephen E. Arnold, "The Google Legacy," infonortics, 2005

[2] Sanjay Ghemawat , Howard Gobioff , Shun-Tak Leung, "The Google file system," ACM SIGOPS Operating Systems Review, Vol.37 No.5, December, 2003.

[3] Who uses Hadoop, http://wiki.apache.org/hadoop/PoweredBy

[4] Apache Hadoop, 2009, http://hadoop.apache.org/

[5] N. Zhang, J. Jing, and P. Liu, "CLOUD SHREDDER: Removing the laptop on-road data disclosure threat in the cloud computing era," in Proc. IEEE Int. Conf. Security and Privacy Comput. Commun., pp.16-18, Changsha, China, Nov., 2011.

[6] Raghu Rmakrishnan, "Sherpa: Cloud Computing of the Third Kind," Data- Intensive Computing Symposium, 2008.

[7] Minsu Park, Singhoon Kang and Seungjoo Kim. "Weakness and Improvement of Cloud Shredder." THE JOURNAL OF KOREA INFORMATION AND COMMUNICATIONS SOCIETY, 38(5), pp.401-409, 2013.

[8] D. Hubbard and M. Sutton, "Top threats to cloud computing," in Cloud Security Al liance, Mar., 2010.

[9] Su-Hyun, In-Sik Hong, Im-Yeong Lee, "Secret Sharing based on XOR for Efficient Data Recovery in Cloud Computing Environment", CISC-S'12, 23(1), pp.49-52, 2013.

[10] Su-Hyun, Im-Yeong Lee, "High-Performance Data Recovery Scheme based on RAID-5 for Distributed Storage Server in Cloud Computing", Proceedings of the conference on Korea Multimedia Society, 16(1), pp.54-57, 2013.

[11] Understanding RAID-5 & I/O Processors, eslim. Jun 2003. April 2013 <http://www.eslim.co.kr/pds/PDSInfo.asp?BD=2&PG=1&BN=3&OPT=&KW=>

김 수 현

e-mail : kimsh@sch.ac.kr

2010년 2월 순천향대학교 정보기술공학부 (학사)

2012년 2월 순천향대학교 컴퓨터학과 (석사)

2012년 3월~현 재 순천향대학교 컴퓨터학과 박사과정

관심분야 : 클라우드 컴퓨팅, 인증, 전자서명

이 임 영

e-mail : imylee@sch.ac.kr

1981년 2월 홍익대학교 전자공학과(학사)

1986년 2월 오사카대학 통신공학전공(석사)

1989년 2월 오사카대학 통신공학전공(박사)

1985년~1994년 한국전자통신연구원 선임연구원

1994년~현 재 순천향대학교 컴퓨터학부 교수

관심분야 : 암호이론, 정보이론, 컴퓨터 보안