JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 3 : 576~585 Mar. 2014

맵리듀스 기반 대용량 개인정보 분산 암호화 처리 시스템

김현욱* · 박성은 · 어성율

The Distributed Encryption Processing System for Large Capacity Personal Information based on MapReduce

Hyun-wook Kim* · Sung-eun Park · Seong-yul Euh

Research Institute, KSIGN Co., Ltd., Seoul 135-515, Korea

요 약

대량의 개인정보가 수집되어 활용됨에 따라 개인정보 유출 등의 보안 문제가 발생하고 있다. 이에 최근에는 수집된 개인정보를 암호화 하여 저장하고 활용하는 방법이 사용되고 있다. 본 논문에서는 기존에 수집된 대량의 개인정보를 단시간에 암호화하기 위한 방법으로 맵리듀스 기반의 분산 암호화 처리 방법을 제안하고, 시스템을 설계하고 구현하였다. 또한 맵리듀스 기반의 분산 암호화 처리 방법의 성능을 검증을 위해 테스트 환경을 구축하여 비교 실험을 진행하였다. 실험 결과 토큰 서버의 암호화 처리 시간이 순차처리 대비 평균 시간 효율이 약 15.3% 정도 향상 하였으며, 병렬처리대비 약 3.13%정도 향상되는 것을 확인 하였다.

ABSTRACT

Collecting and utilizing have a huge amount of personal data have caused severe security issues such as leakage of personal information. Several encryption algorithms for collected personal information have been widely adopted to prevent such problems. In this paper, a novel algorithm based on MapReduce is proposed for encrypting such private information. Furthermore, test environment has been built for the performance verification of the distributed encryption processing method. As the result of the test, average time efficiency has improved to 15.3% compare to encryption processing of token server and 3.13% compare to parallel processing.

키워드 : 분산 컴퓨팅, 하둡, 맵리듀스, 암호화, 정보보안

Key word : Distributed Computing, Hadoop, MapReduce, Encryption, Information Security

접수일자 : 2013. 10. 24 심사완료일자 : 2013. 11. 19 게재확정일자 : 2013. 12. 04

* **Corresponding Author** Hyun-Wook Kim(E-mail:khw@ksign.com, Tel:+82-070-7090-7940)
Research Institute, KSIGN Co., Ltd, Seoul 135-515, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.3.576 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

정보화 사회에서는 개인정보의 수집과 이용이 보편화되면서, 개인정보의 경제적 가치가 증대됨에 따라 최근에는 개인정보의 수집 범위가 크게 증가하였다. 하지만 민간 사업자 또는 공공기관 개인정보보호 의식이 저조하여 저장된 개인의 정보가 악의적인 해킹에 노출되는 경우가 많이 발생하고 있는 실정이다[1].

개인 정보가 노출이 되면 개인의 기본권인 인격권의 침해로 이어질 가능성이 높으며, 개인 정보 유출로 인한 기업이 겪는 이미지 실추와 피해 보상금은 막대하다[2].

이에 따라 2011년 9월 30일부터 시행된 개인정보보호법에서는 주민등록번호, 카드번호 등 고유식별정보가 변조, 유출, 도용되지 않도록 암호화 등 안전성 확보를 위한 조치를 취하도록 명시하고 있다. 특히 데이터베이스에 저장되는 개인정보를 암호화하여 관리하는 것은 내·외부에서 악의적인 의도를 가지고 접근하거나 유출을 시도하는 위험으로부터 가장 안전하게 보호할 수 있는 방법이다. 하지만 항시적인 서비스가 요구되고 데이터베이스가 대용량화되는 현실에서 데이터 암호화를 진행하게 된다면 필연적으로 시스템의 성능 저하를 발생시키며, 이를 위한 장시간의 서비스 중단은 사실상 허용이 불가능하다[3].

본 논문에서는 데이터베이스에 존재하는 데이터 암호화 시 시스템의 성능저하를 피하기 위해 SAM (Sequential Access Method)파일에 존재하는 대량의 개인정보를 맵리듀스(MapReduce) 기반으로 암호화하는 방법을 제안하고 그 시스템을 설계하였다. 그리고 테스트 환경을 구축하여 성능을 평가하였다.

본 논문의 구성은 2장에서 하둡(Hadopo)을 구성하는 HDFS(Hadoop Distributed File System)와 맵리듀스에 대해 설명하고, 3장에서 맵리듀스 기반의 분산 암호화 처리 시스템을 설계하고 구현하였으며, 4장에서는 테스트 환경을 구성하여 실험을 통해 성능을 평가하였다. 마지막으로, 5장에서 결론을 맺는다.

Ⅱ. 하 둡

하둡은 여러 대로 구성된 컴퓨터 클러스터(Computer Cluster)를 이용하여 데이터를 처리하기 위한 분산 응용 프로그램을 지원하는 자바(JAVA) 기반의 오픈소스 프레임워크이다[4].

하둡은 상황에 따라 효율적으로 적용할 수 있도록 아래 그림 1과 같이 다양한 서브프로젝트가 존재하며, 이를 하둡 에코시스템(Hadoop EcoSystem)이라고 한다, 하둡 에코시스템은 데이터를 분산 저장하는 HDFS와 데이터를 분산 처리 및 분석하는 맵리듀스가 하둡의 코어 프로젝트(Core Project)에 해당 [5]하며, 나머지는 서브 프로젝트(Sub Project)에 해당한다.

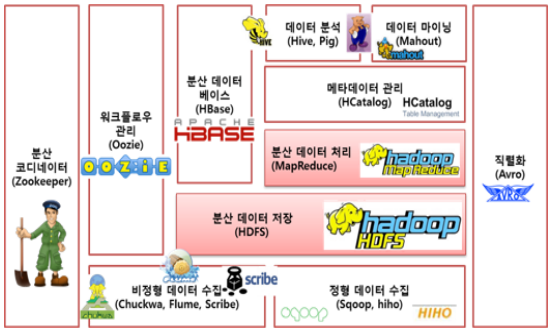

그림 1. 하둡 에코시스템
Fig. 1 Hadoop EcoSystem

하둡은 아래 그림 2와 같이 기본적으로 Master/Slave 구조를 가지며, HDFS/MapReduce 계층으로 분리된다. 하둡 클러스터(Hadoop Cluster)는 하나의 마스터 노드(Master Node)와 여러대의 슬레이브 노드(Slave Node) 구성된다. 하나의 마스터 노드에는 최대 4096대까지 슬레이브 노드로 구성된 시스템을 구축할 수 있다.

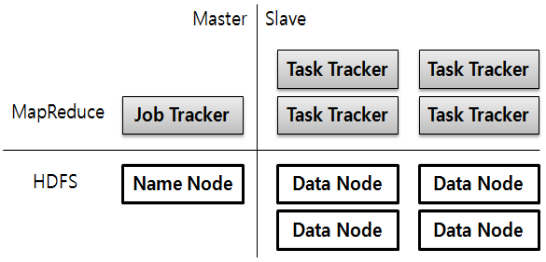

그림 2. 하둡 구조
Fig. 2 Hadoop architecture

HDFS의 마스터 노드는 네임 노드(Name Node)라고 명칭하며, 슬레이브 노드는 데이터 노드(Data Node)로 명칭하고 있다. 맵리듀스에서의 마스터 노드는 잡 트랙커(Job Tracker)로 명칭하며, 슬레이브 노드는 태스크 트랙커(Task Tracker)로 명칭하고 있다.

2.1. HDFS

HDFS는 파일 분산 저장을 목적으로 하는 파일시스템으로 아래 그림 3과 같이 네임 노드, 데이터 노드 2가지 형태의 서버로 구성된다. 네임 노드는 마스터 개념의 네임 노드와 네임 노드의 파일 시스템 이미지 갱신을 위해 체크포인트 역할을 수행하는 보조 네임 노드(Secondary Name Node) 1대씩으로 구성된다. 데이터 노드는 실제 파일이 블록(Block)단위로 분산, 복제되어 저장되는 서버로 하나의 클러스터에 최대 4096대까지 존재할 수 있다. 네임 노드는 데이터 노드에 저장되는 파일에 대한 메타데이터를 관리하는 역할을 한다[6].

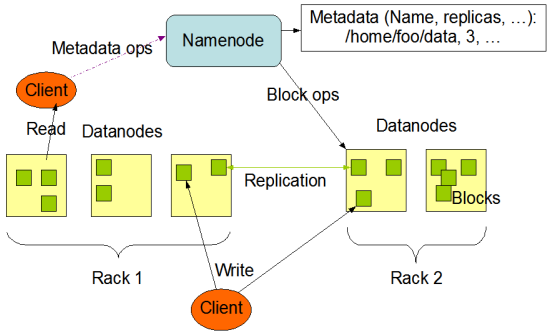

그림 3. HDFS 구조
Fig. 3 HDFS architecture

2.2. 맵리듀스

맵리듀스는 맵(map())과 리듀스(reduce()) 메서드로 이루어진 프로그래밍 모델이며, 대규모 분산 컴퓨팅 혹은 단일 컴퓨팅 환경에서 대량의 데이터를 병렬로 분석하는 프레임워크이다[7].

맵과 리듀스 메서드는 Key/Value 입출력 형태를 가지 것을 특징으로 NoSQL에서 사용하기 적합하다.

맵리듀스 구조는 1개의 잡 트랙커와 여러 개의 태스크 트랙커로 구성된다. 태스크 트랙커는 하나의 잡 트랙커에 4096개까지 존재할 수 있다.

잡 트랙커는 맵리듀스 잡(Job) 실행 요청을 받아 하나의 잡을 맵과 리듀스로 분리하여 태스크(Task) 단위로 적절한 태스크 트랙커에게 할당하고, 수행에 실패한 태스크에 대해서 재스케쥴링을 진행하는 역할을 한다. 태스크 트랙커는 잡 트랙커로부터 받은 태스크를 실행하는 노드이다.

잡 트랙커에서 잡 실행 명령을 받게 되면, 블록단위로 저장된 파일이 존재하는 데이터 노드의 태스크 트랙커에서 맵 태스크(Map Task)와 리듀스 태스크(Reduce Task)를 실행한다. 이때 입력 파일의 블록은 하나 이상의 조각으로 나누어져 (Split) 맵 태스크의 입력 값으로 사용된다.

맵 태스크가 실행되면 아래 그림 4와 같이 새로운 키와 값을 가지는 임시 파일로 저장되게 된다. 이렇게 임시파일로 저장된 키와 값의 쌍을 이루는 파일은 동일한 키를 가지는 파일의 묶음으로 리듀스 태스크로 전달된다. 이 과정을 셔플링(Shuffling)이라고 한다[8]. 리듀스 태스크 에서는 최종 결과 파일을 생성하게 된다.

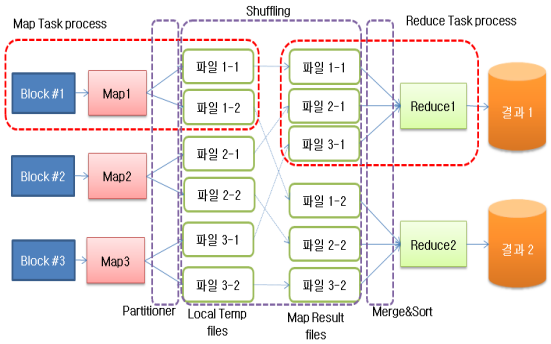

그림 4. 맵리듀스 프레임워크 개념도
Fig. 4 MapReduce framework introduction

Ⅲ. 시스템 설계 및 구현

3.1. 맵리듀스 기반 분산 암호화 처리 개요

본 논문에서 제안하는 분산 암호화 처리 시스템은 아래 그림 5와 같이 SAM파일의 저장된 개인정보를 맵리듀스 프로그램을 통해 암호화하는 시스템이다.

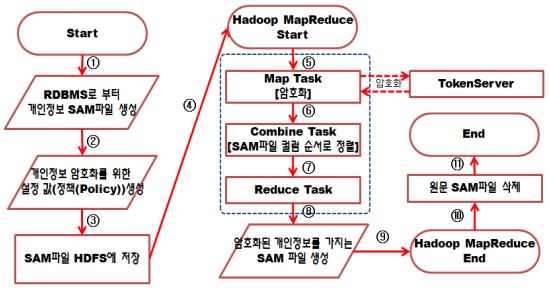

그림 5. 맵리듀스 기반 분산 암호화 처리 순서도
Fig. 5 Ditributed encryption processing algorithm based on MapReduce

위 그림 5에서 점선으로 표시된 부분이 맵리듀스 프로그램의 내부과정이다. 개인정보는 토큰암호화 방식으로 암호화 서버(Token Server)를 거쳐 암호화 된다. 암호화 과정을 맵 태스크에서 진행하는 이유는 분산으로 처리하기 위해서이다. 리듀스 태스크는 맵퍼(Mapper)의 결과 파일을 취합하여 최종 파일을 출력하는 역할을 하기 때문에 내부 프레임워크에 의해 하나의 노드에서 진행되기 때문이다. 맵퍼는 맵 태스크 프로그램이다.

3.2. 맵리듀스 기반의 분산 암호화 처리 목적

암호화 서버에 순차적으로 암호화를 요청하는 방식에서 데이터를 암호화 할 경우 암호화 서버에서는 아래 그림 6과 같이 암호화를 위한 다음 데이터를 수신하기까지 대기시간(Wait Time)이 발생하게 된다.

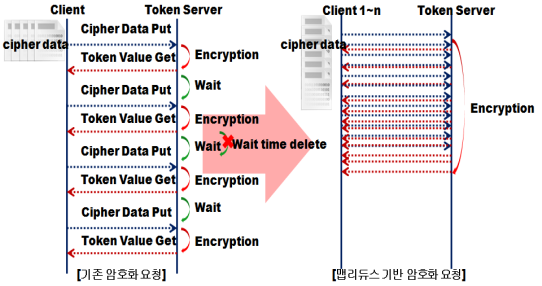

그림 6. 요청방식에 따른 암호화 처리 과정 비교
Fig. 6 Comparison of processing by the request method

대기 시간을 줄이기 위해서 병렬로 처리하거나 분산된 노드에서 암호화 요청을 하는 방법이 사용된다. 하지만 파일의 수, 데이터양에 따라 계속 프로그램 구조를 변경해주어야 하는 경우가 발생한다.

이를 본 논문에서는 분산처리프레임워크인 맵리듀스를 적용하여 해결하고자 한다.

3.3. 맵리듀스 프로그램 설계 및 구현

분산 암호화 처리 특성상 암호화 서버에서는 동시에 많은 암호화 요청을 받게 된다. 이는 CPU, 메모리, 네트워크 등 자원에 과부화를 야기한다[10]. 때문에, 암호화 서버의 사양을 고려해야한다. 그렇다고 데이터를 한건씩 암호화 할 경우 네트워크 트래픽이 증가하여 효율적인 통신을 할 수 없으며, 또한 최대한 많은 데이터를 보내게 된다면, 인터럽트 처리 시간에 의해 시스템이 다운될 수 있음을 고려해야 한다.

3.3.1. 맵 태스크(Map Task) 설계 및 구현

맵 태스크는 아래 그림 7과 같이 SAM파일에 저장되어 있는 개인정보를 암호화한 값으로 출력하는 역할을 한다.

맵 태스크는 아래와 같은 특징을 가지도록 설계하고 구현하였다.

첫 번째, map()에서는 파일에 저장되어 있는 데이터에 대한 암호화 정책을 적용하고, 암호화 하지 않는 데이터를 분리하기 위해서 저장되어 있는 세로 한 줄을 하나의 컬럼(Column)으로 인식하고, 컬럼 단위로 암호화 정책을 적용한다.

두 번째, 맵 태스크는 분산되어 있는 태스크 트랙커에서 생성되어 실행되는 구조를 고려해야 한다. 특히 이번 맵 태스크 에서 사용한 변수 값은 사용이 불가능하며, 각 map()은 독립적으로 실행되는 운영 구조를 고려해야 한다. 이에 공통으로 사용되는 변수, 메서드를 싱글톤 패턴(Singleton Pattern)으로 정의하여 사용한다.

세 번째, 입력파일의 끝, 또는 split의 끝을 알리는 구분문자가 존재하거나, 또는 지정된 데이터 건 수만큼 개인정보 만큼 쌓였을 때 암호화 서버로 암호화 요청을 한다.

네 번째, 기존 파일의 데이터 순서대로 리듀스태스크에서 정렬할 수 있도록 암호화된 값에 기존 파일에서의 컬럼 위치 정보를 추가하여 출력 값을 생성한다.

다섯 번째, 출력 키 값은 먼저 들어온 값이 리듀

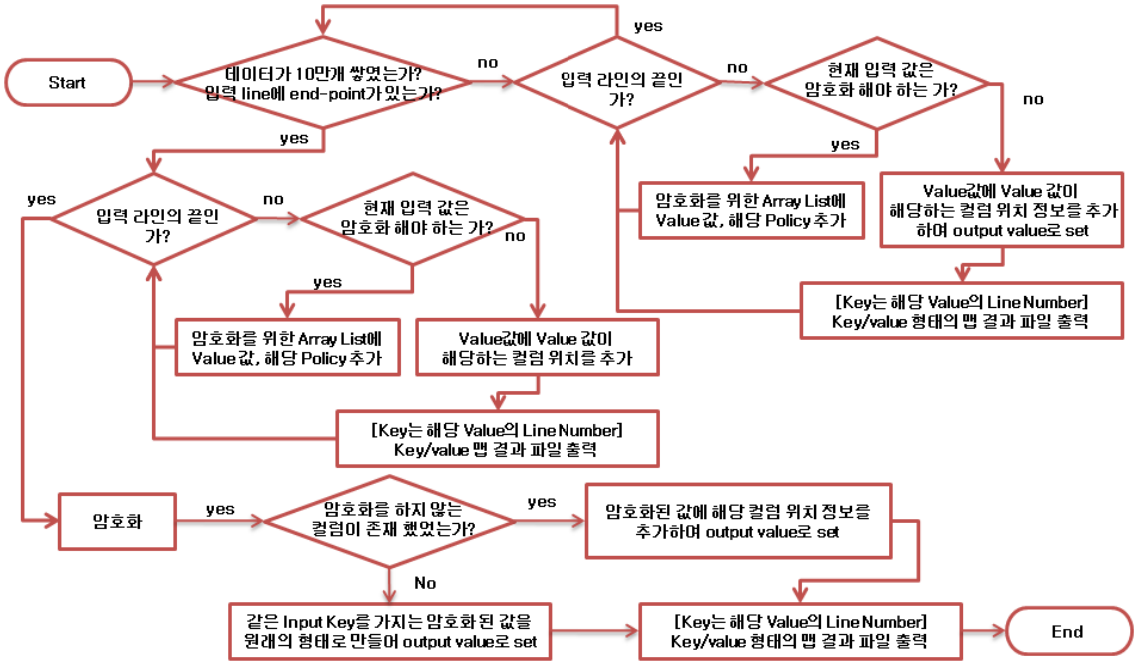

그림 7. Map Task 순서도
Fig. 7 Map Task algorithm

서(Reducer)에 먼저 전달되도록 하기 위해서, 맵퍼(Mapper)의 입력 키값인 LongWritable 형태의 행 번호(Line Number) 값을 이용한다. 리듀서는 리듀스 태스크 프로그램이다.

먼저 싱글톤 객체에 저장된 데이터가 지정된 수만큼 저장되어있는지, 또는 입력 받은 데이터가 split 또는 맵 file의 마지막인지 확인 한다. 만약 조건이 맞지 않는다면 계속 데이터를 저장하며 질의를 반복하게 된다. 조건이 성립하면, 싱글톤 객체의 암호화 메서드를 실행하여 배열에 저장된 데이터를 암호화 하고, 암호화 한 값에 컬럼 위치 값을 추가한 값을 생성하고, 해당 값의 라인번호를 키(key)로 결과를 출력한다. 만약 암호화를 하지 않는 데이터가 존재할 경우 라인번호를 키 값으로, 데이터에 컬럼 위치 값을 추가한 값을 임시파일을 출력하게 된다. 이렇게 입력파일의 끝을 만나서 처리할 때까지 반복하여 동작하게 된다.

아래 그림 8은 맵 태스크를 구현한 맵퍼 클래스다이어그램이다.

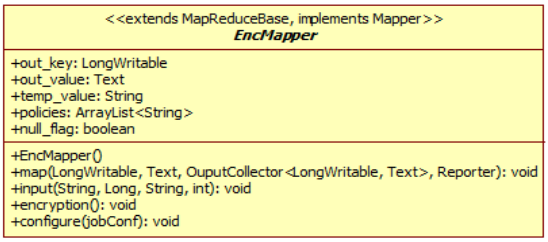

그림 8. Mapper 클래스 다이어그램
Fig. 8 Mapper class diagram

out_key, out_value는 맵퍼에서 출력하기 위한 값을 저장하기 위한 클래스 변수 이다. policies는 암호화 정책 명을 저장하고 있다. null_flag는 암호화를 하지 않는 컬럼을 구별하기 위해 사용한다.

map()은 실제 맵퍼의 기능을 정의한 메서드 이다. map()의 역할은 End Point를 만났거나 배열에 지정된 만큼 데이터가 저장되어있을 경우 토큰서버를 통해 암호화를 진행하고, 아닐 경우 계속하여 배열에 데이터를 저장한다. 암호화를 진행하게 된 이후에는 암호화 결과

값을 key/value 형태의 임시파일을 출력하게 되는데 이때 출력 키 값은 입력 키 값과 동일하며, 출력 값은 암호화 값으로 정의하였다.

input()은 파일로부터 읽어드린 값과 키, 그리고 정책, 컬럼 번호를 싱글톤 객체로 저장하는 역할을 한다. encryption()은 암호화 요청을 하는 역할로, 싱글톤 객체에 정의되어 있는 encryption()을 실행 한다. configuration()에서는 파일로 저장된 정책정보 또는 입력받은 정책정보를 설정하는 역할을 한다.

아래 그림 9는 맵퍼에서 사용하는 싱글톤 객체의 클래스 다이어그램이다.

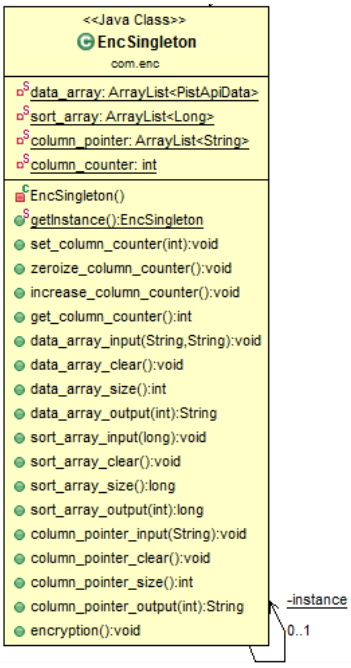

그림 9. 싱글톤 객체 클래스다이어그램
Fig. 9 Singleton object class diagram

data_array는 암호화하기 위한 데이터와 정책을 저장하는 배열이며, sort_array는 암호화를 진행 한 후 원래의 키와 Mapper에서 매칭을 시켜주기 위한 키값을 가지고 있다. column_pointer는 일부만 암호화를 진행할 때, 컬럼들의 순서를 알기위한 배열이다.

data_array_input(), data_array_clear(), data_array_size(), data_array_output()은 data_array 관리를 위한 메서드이다.

sort_array_input(), sort_array_clear(), sort_array_size(), sort_array_output()은 암호화된 데이터를 정렬하기 위한 sort_array관리를 위한 메서드이다.

column_pointer_input(), column_pointer_clear(), column_pointer_size(), column_pointer_output()은 column_pointer를 관리하기 위한 메서드 이다.

encryption()은 현재 data_array에 저장되어 있는 데이터들을 토큰서버를 통해 암호화 하는 메서드이다.

3.3.2. File Input Format 설계 및 구현

입력 파일은 InputFormat 클래스와 RecordReader 인터페이스로 구성된 File Input Format에 의해 Split되어 key/value값으로 전달된다. 이때 Split 단위는 하둡 내부 프레임워크에 의해서 이루어지게 된다.

Split된 데이터들을 n개씩 암호화 서버로 암호화를 요청할 경우 전체 데이터 수가 n개 단위로 나누어 지지 않을 경우 발생하는 나머지 데이터에 대해서는 암호화 처리가 불가능하게 된다.

이에 그림 10과 같이 Split의 마지막과 파일의 마지막을 알리는 문자를 삽입하는 File Input Format을 설계하고 구현하였다.

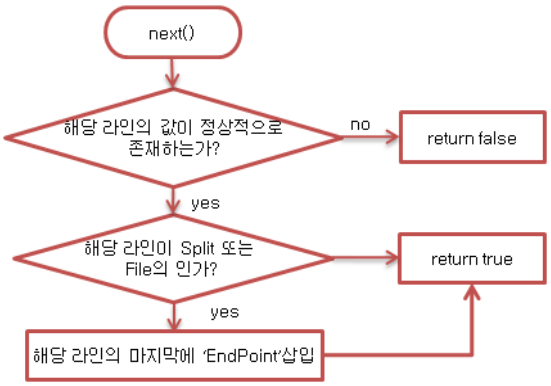

그림 10. File Input Format 순서도
Fig. 10 File Input Format algorithm

InputFormat 클래스는 하둡에서 제공하는 Text Input Format을 오버라이딩하여 구현하였으며, Record Reader 인터페이스는 아래 그림 11과 같이 구현하였다.

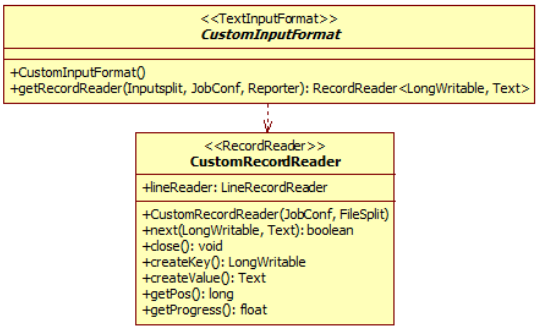

그림 11. File Input Format 클래스다이어그램
Fig. 11 File Input Format class diagram

CustomInputFormat 클래스는 TextInput Format을 오버라이딩한 InputFormat이며, CustomRecordReader 클래스는 입력라인이 Split의 끝이거나, 파일의 마지막임을 알리는 구분 문자를 추가해주도록 RecordReader 인터페이스를 구현한 사용자 정의 클래스이다.

3.3.3. Combiner, Reduce Task 설계

Reduce Task에서는 아 그림 12와 같이 암호화된 값을 취합하여 최종 결과파일을 출력한다.

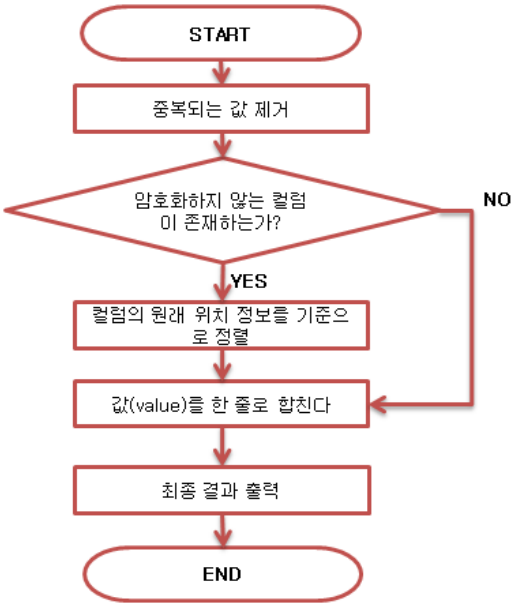

그림 12. Reduce Task 순서도
Fig. 12 Reduce Task algorithm

Reduce Task의 작업 시간을 단축하기 위해 위 그림 5와 같이 Map과 Reduce Task 사이에 Combiner를 삽입하였다. Combiner에서는 Mapper의 출력 값에 추가되어 있는 컬럼 위치 값을 기준으로 값을 정렬하고, 컬럼 위치 값을 제거한 후 결과파일 출력하는 역할을 한다. Combiner에서 출력된 파일은 Reducer의 입력파일로 전달된다. Combiner와 Reducer는 아래 그림 13과 같은 클래스를 사용한다.

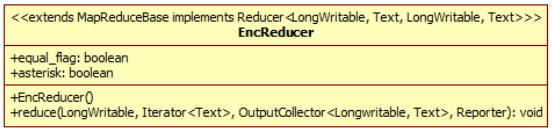

그림 13. Reducer 클래스 다이어그램
Fig. 13 Reducer class diagram

equal_flag는 입력 key값이 중복될 경우 하나의 파일로 통합해 주기 위한 변수이며, asterisk는 컬럼 위치값과 실제데이터를 구분하기 위한 구분자 변수이다. reduce()는 Reducer를 정의한 부분이다.

Ⅳ. 성능 측정

4.1. 테스트 환경 구성

맵리듀스 기반의 분산 암호화 처리 테스트 환경은 그림 14와 같이 구성된다.

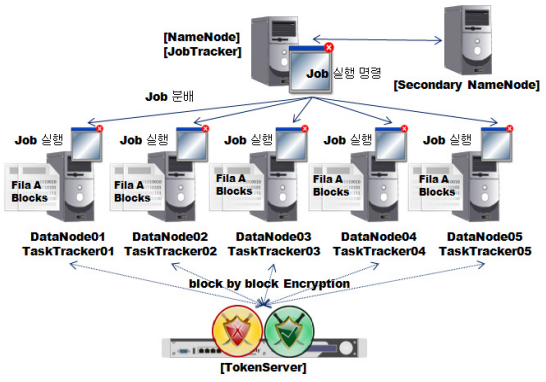

그림 14. 분산 암호화 테스트 환경 구성
Fig. 14 Distributed encryption test environment

암호화 서버 1대와, 하둡 노드 7대를 아래 표 1과 같은 사양을 가지는 VM(Virtual Machine)으로 구성했다.

표 1. VM 제원 정리
Table. 1 VM specification

구분	항목	사양
Hadoop Node	CPU	1core
	Memory	8GB
	HDD	2TB
	O/S	Ubuntu 12.04 server 64bit
Token Server	CPU	1core
	Memory	8GB
	HDD	8GB
	O/S	CentOS 6.4 64bit

하이퍼바이저(Hypervisor)는 KVM(Kernel-based Virtual Machine)을 사용하였다. VM이 구성되는 호스트 PC의 사양은 아래 표 2와 같다.

표 2. 테스트 PC 사양
Table. 2 PC specification

항목	사양
CPU	4GHz * (8core)
Memory	32GB
HDD	4TB(7200rpm, SATA3)
POWER	700W
O/S	CentOS 6.4 64bit

분산암호화처리의 성능을 비교하기 위한 순차 암호화 처리, 병렬 암호화 처리 환경을 아래 그림 15와 같이 구성하였다. 암호화 서버는 분산암호화처리에서 사용하는 암호화 서버와 동일하다.

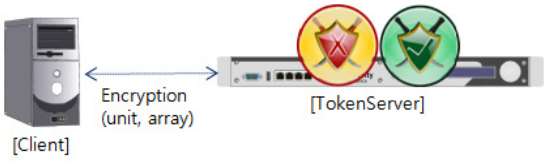

그림 15. 싱글 노드 암호화 테스트 환경
Fig. 15 Single node encryption test environment

싱글 노드 테스트 환경에 사용된 PC의 제원 사항은 아래 표 3과 같다.

표 3. 싱글 노드 사양
Table. 3 Single node specification

항목	사양
CPU	3.4GHz * (4core, 8 thread)
Memory	8GB
HDD	1TB(7200rpm, SATA3)
POWER	500W
O/S	Windows 7 64bit

4.2. 성능 측정 방법

싱글노드에서 순차적으로 암호화 요청 시 암호화 서버에서 암호화를 시작한 시간과 끝나는 시간, 싱글노드에서 병렬로 암호화 요청 시 암호화 서버에서 암호화를 시작한 시간과 끝나는 시간을 아래 표 4와 같이 측정하였다.

싱글노드에서 SAM파일은 Local Storage에 저장하였으며, 분산 암호화 처리에서 사용되는 SAM파일은 HDFS에 저장하였다.

암호화에 사용된 파일은 주민등록 번호와 같은 13자리 숫자와 전화번호와 같이 11자리 숫자로 구성된 데이터가 저장되어있다.

표 4. 성능 측정 방법
Table. 4 Performance measure method

번호	방법
1	파일 5개 (총 2,000만건) 암호화 시간 측정
2	파일 10개 (총 4,000만건) 암호화 시간 측정
3	파일 15개(총 6,000만건) 암호화 시간 측정
4	주민번호, 전화번호 각 2,000만개로 구성된 파일 10개 (총 4억건) 암호화 시간 측정

암호화 알고리즘으로 일반 암호화 알고리즘 방식보다 3~5배가량 속도가 빠른 FPE(Format Preserving Encryption)토큰화 기술[9] 중 FFX(Fast Function Extraction)을 사용하였다. 암호화 방식은 Block Token 방식을 사용하였으며, 13자리 숫자 데이터는 마지막 6자리를 암호화 하고, 11자리 숫자 데이터는 마지막 4자리를 암호화 하는 정책을 적용하였다.

테스트 프로그램은 JAVA로 구현하였다.

4.3. 성능 측정 결과

각 테스트마다 싱글노드에서 순차적 암호화 처리, 병렬 암호화 처리, 맵리듀스 기반 분산 암호화 처리하는 방식으로 5번씩 진행하였으며, 그 결과는 아래 표 5와 같다. 아래 표 5에 값은 측정된 시간들의 평균이다.

표 5. 암호화 시간 측정 결과
Table. 5 Encryption time measurement result

구분	2천만건	4천만건	6천만건	4억건
순차처리	0:07:19	0:14:37	0:22:07	2:26:53
병렬처리	0:06:44	0:13:35	0:18:42	2:11:49
맵리듀스	0:06:03	0:12:25	0:18:08	2:10:59

위와 같이 맵리듀스 기반으로 진행했을 때 암호화 서버에서 데이터를 처리하는 시간이 단축되는 것을 확인할 수 있었다.

순차처리 대비 맵리듀스 처리 시간의 효율과 병렬처리 대비 시간 효율을 아래 표 6과 같다.

표 6. 맵리듀스 처리 시간 효율
Table. 6 Encryption time measurement result

구분	2천만건	4천만건	6천만건	4억건
순차처리 대비	17.31	15.05	18.01	10.82
병렬처리 대비	0.27	8.59	3.03	0.63

맵리듀스 기반으로 암호화를 처리하였을 때 암호화 서버에서의 처리 효율이 순차처리 대비 약 15%, 병렬처리 대비 약 3.13%정도 향상되는 것으로 확인 되었다.

아래 표 7은 암호화 처리 시간과 암호화된 최종 파일이 저장되는 시간을 포함한 측정 시간이다.

표 7. 파일 저장 시간 포함 측정 결과
Table. 7 Measurement results containing the file save times

구분	2천만건	4천만건	6천만건	4억건
순차처리	0:07:20	0:14:38	0:22:08	2:29:53
병렬처리	0:06:58	0:13:40	0:18:46	2:13:11
맵리듀스	0:07:00	0:13:55	0:21:18	2:20:00

하지만 암호화된 데이터를 파일로 저장하는 시간까지 측정을 해보면 싱글노드에서 병렬로 암호화를 요청했을 때 시간이 단축되는 것을 위 표 7과 같이 확인할 수 있었다. 이는 HDFS의 복재(Replication) 정책에 의해 데이터 노드로 분산 복제되어 저장되는 시간이 존재하기 때문이다.

Ⅴ. 결 론

대용량 개인정보를 효과적으로 암호화 할 수 있는 맵리듀스 기반의 File Input Format, Map Task, Combiner, Reduce Task를 설계하고 구현하였다. 그리고 하둡 코어 프로젝트 기반의 분산 암호화 처리 시스템을 구축하였다. 또한 테스트를 통해 성능을 비교 측정한 결과 암호화 서버에서의 대기시간을 최소화를 통해 암호화 처리 시간이 단축되는 것을 확인하였으며, 순차처리 및 병렬처리 대비 처리시간 효율이 높아지는 것을 확인하였다.

본 논문에서 제안하는 맵리듀스 기반의 분산 암호화 처리 방식은 파일의 형식, 데이터 수, 파일의 수에 상관없이 대용량 개인정보 암호화를 효과적 처리할 수 있다. 이는 분산 처리가 필요한 다양한 환경의 암호화 시스템에 적용되어 성능을 충분히 개선할 수 있을 것으로 기대한다.

감사의 글

This work was supported by the IT R&D program of MSIP/KEIT. [10041579, Development the Personal Information Security service solution using tokenization technology]

REFERENCES

[1] Y. J. Song, K. Y. Park, H. J. Kim, J. M. Do, and D. H. Lee , “A study on the secret sharing scheme for managing a large quantity of data including individual information,” Dongguk University, KISA : CA, Report KISA-RP-2009-0013, Sep. 2009.

[2] J. W. Kang, "A Study of Effective Privacy Protection System on High Concurrent Transaction Database System", *Convergence Security Journal*, vol. 12, no. 2, pp.107-113, May. 2012.
[3] J. H. Hong, I. R. Jeong, "A case study on the performance of encrypted data", *Journal of the Korea Institute of Information Security and Cryptology*, vol. 22, no. 6, pp.1301-1313, Dec. 2012.
[4] Ken Mann, m. Tim Jones. (2008, December). Distributed computing with Linux and Hadoop [Internet]. Available: http://www.ibm.com/developerworks/linux/library/l-hadoop
[5] T. H. Kum, "Design and Implementation of A Monitor for Hadoop Cluster", M.S. dissertation, Computer Engineering, Hanyang University, Feb.2011.
[6] D. Borthaku. (2013, April). HDFS Architecture Guide. *The Apache Software Foundation* [Online]. Available: http://hadoop.apache.org/docs/stable/hdfs_ design.html
[7] J. H. Kwak, J. W. Yoon, Y. H. Jung, J. g. Hahm, D. I. Park, "Large-scale Data Analysis based on Hadoop for Astroinformatics", *Journal of KIISE : Computing Practices and Letters*, vol. 17, no. 11, pp.587-591, Nov. 2011.
[8] J. W. Lee, S.K. Kim, "Complementary research and Analysis for hadoop" in *The Korea Society of Computer and Information Winter Conference 2012*, vol. 20, no. 2, pp.3- 6, July. 2012
[9] E. M. Park, "A Study on Database Encryption Scheme for Privacy Protection under Packaged Software Environment", M.S. dissertation, Hanyang Cyber University, Feb. 2013.
[10] J. K. Heo, "Web Application Security using Distributed Encipherment", *The Korea Contents Association Journal*, vol. 8, no. 4, pp.10-16, Apr, 2008.

김현욱(Hyun-Wook Kim)

2009년 : 광운대학교 컴퓨터공학과 공학사
2013년 : 광운대학교 일반대학원 전자공학과 석 · 박사통합과정 수료
2013년~현재 : (주)케이사인 정보보안연구실 주임연구원
※관심분야 : 클라우드 컴퓨팅, 하둡, 기계학습, 분산처리

박성은(Sung-Eun Park)

1998년 : 아주대학교 컴퓨터공학과 공학사
2000년 : 아주대학교 컴퓨터공학과 석사
2002년 : 아주대학교 컴퓨터공학과 박사과정 수료
2009년~현재 : (주)케이사인 정보보안연구실 연구실장
※관심분야 : 클라우드 컴퓨팅, Virtual Appliance, Software-Defined Security

어성율(Seong-yul Euh)

1997년 : 아주대학교 컴퓨터공학과 공학사
1999년 : 아주대학교 컴퓨터공학과 석사
2001년~현재 : (주)케이사인 개발본부 총괄
※관심분야 : 클라우드 컴퓨팅, 빅데이터, 개인정보 암호화 처리