Journal of the Korea Convergence Society
Vol. 9. No. 10, pp. 13-19, 2018

ISSN 2233-4890
https://doi.org/10.15207/JKCS.2018.9.10.013

빅데이터 처리시간 감소와 저장 효율성이 향상을 위한 맵리듀스 기반 빅데이터 처리 기법 구현

이협건[1*], 김영운[1], 김기영[2]
[1]한국폴리텍대학 서울강서캠퍼스 데이터분석과 교수, [2]서일대학교 소프트웨어공학과

Implement of MapReduce-based Big Data Processing Scheme for Reducing Big Data Processing Delay Time and Store Data

Hyeopgeon Lee[1*], Young-Woon Kim[1], Ki-Young Kim[2]
[1]Professor, Department of Data Analysis, Seoul Gangseo Campus of Korea Polytechnic
[2]Professor, Department of Computer Software, Seoil University

요 약 맵리듀스는 하둡의 필수 핵심 기술로 하둡 분산 파일 시스템을 기반으로 빅데이터를 처리하는 가장 보편화되어 사용되고 있다. 그러나 기존 맵리듀스 기반 빅데이터 처리 기법은 하둡 분산 파일 시스템에 정해진 블록의 크기대로 파일 나눠 저장되는 특징으로 인해 인프라 자원의 낭비가 극심하다. 이에 본 논문에서는 효율적인 맵리듀스 기반 빅데이터 처리 기법을 제안한다. 제안하는 기법은 처리할 데이터를 사전에 맵리듀스에서 처리하기 적합한 데이터 형태로 변환 및 압축하여 빅데이터 인프라 환경의 저장 효율성을 증가시킨다. 또한 제안하는 기법은 저장 효율성을 중점으로 구현했을 때 발생할 수 있는 데이터 처리 시간의 지연 문제를 해결한다.

주제어 : 빅데이터, 빅데이터 처리 기법, 빅데이터 압축 기법, 하둡 분산 파일 시스템, 맵리듀스

Abstract MapReduce, the Hadoop's essential core technology, is most commonly used to process big data based on the Hadoop distributed file system. However, the existing MapReduce-based big data processing techniques have a feature of dividing and storing files in blocks predefined in the Hadoop distributed file system, thus wasting huge infrastructure resources. Therefore, in this paper, we propose an efficient MapReduce-based big data processing scheme. The proposed method enhances the storage efficiency of a big data infrastructure environment by converting and compressing the data to be processed into a data format in advance suitable for processing by MapReduce. In addition, the proposed method solves the problem of the data processing time delay arising from when implementing with focus on the storage efficiency.

Key Words : Big Data, Big Data Process Scheme, Big Data Compress Scheme, Hadoop Distributed File System, MapReduce

1. 서론

맵리듀스[1,2]는 빅데이터 처리의 핵심 기술 중에 하나로 방대한 양의 데이터를 분산된 멀티 노드를 활용하여 처리한다. 맵리듀스 기반 빅데이터 처리 기술의 특징은 하둡 분산 파일 시스템에 저장된 빅데이터들의 처리를 수행하는 노드가 많으면 많을수록 데이터 처리 성능은 급격하게 향상된다. 따라서 맵리듀스는 분산 처리 환경을 기반으로 구현된 빅데이터 처리기술이다.

그러나 기존의 맵리듀스 기반 빅데이터 처리 기법은 하둡 분산 파일 시스템에 정해진 블록의 크기대로 파일 나누고 분산된 노드들에 동일한 데이터가 중복 저장되는

*Corresponding Author : Hyeopgeon Lee (hglee67@kopo.ac.kr)
Received August 6, 2018 Revised August 31, 2018
Accepted October 20, 2018 Published October 28, 2018

특징으로 인해 네트워크 사용률 및 데이터 저장 공간 등의 인프라 자원의 낭비가 극심하다.

이에 본 논문에서는 효율적인 맵리듀스 기반 빅데이터 처리 기법을 제안한다. 제안하는 기법은 빅데이터 변환 단계와 빅데이터 재생성 단계로 구분하여 구현한다. 제안하는 기법은 맵리듀스가 실행되기 전에 수행되며, 처리할 빅데이터를 맵리듀스 처리에 적합한 형태로 파일을 변환 및 압축을 통해 저장 용량 효율성을 증가시키고, 데이터 처리 시간을 감소시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 맵리듀스와 맵리듀스 기반 빅데이터 처리 기법을 살펴보고, 요구사항을 분석한다. 3장에서는 본 논문에서 제안하는 효율적인 맵리듀스 기반 빅데이터 처리 기법 구현을 제시한다. 4에서는 제안한 기법의 성능을 분석하고, 마지막 V장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련연구

2.1 맵리듀스

맵리듀스[3,4]는 방대한 양의 데이터를 분산된 멀티 노드를 활용하여 빅데이터를 처리한다. 맵리듀스는 크게 맵 함수와 리듀스 함수로 구성되며, 맵리듀스 프레임워크를 활용하여 맵 함수와 리듀스 함수의 구현을 통해 빅데이터 처리 및 분석을 수행한다. 맵리듀스를 통한 빅데이터 처리는 하둡 분산 파일 시스템에 존재하는 데이터만 처리가 가능하다 또한 맵리듀스의 실행은 하둡 환경에서만 가능하다.

맵리듀스의 주요 동작 단계[5]는 맵리듀스 프레임워크에 정의된 Mapper 객체를 상속받은 자바 객체의 맵 함수 실행 단계, 맵 함수를 통해 처리된 데이터 결과를 병합하고 정렬하는 Shuffle and Sort 단계와 맵리듀스 프레임워크에 정의된 Reducer 객체를 상속받은 자바 객체의 리듀스 함수 실행되어 결과가 생성되는 단계로 구성된다.

Fig. 1은 맵리듀스의 주요 동작 단계를 나타낸다.

Fig. 1. Step of MapReduce

Fig. 1과 같이 맵리듀스의 주요 동작 단계는 순차적으로 진행하며 빅데이터를 분석한다. 특히 맵 함수 실행단계는 주로 수집되는 데이터들에 대해 1차 가공 처리를 수행하며, 수집된 데이터들을 정제하는 알고리즘들이 정의된다.

2.2 맵리듀스 기반 빅데이터 처리 기법

맵리듀스 기반 빅데이터 처리 기법[6, 7, 8]은 자바 기반의 맵리듀스 프레임워크를 활용한다. 맵리듀스 기반 빅데이터 처리 기법은 하둡 분산 파일 시스템에 저장된 데이터를 처리 및 분석하기 위해 수집된 데이터를 로드하며 맵리듀스 전처리 과정을 거쳐 실행된다. Fig. 2는 맵리듀스 기반 빅데이터 처리 기법의 진행 과정을 나타낸다.

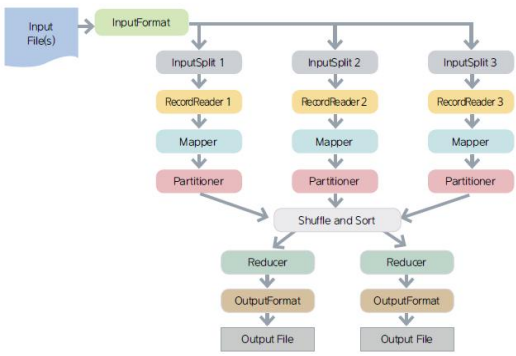

Fig. 2. Process of the MapReduce-based big data processing scheme

Fig. 2와 같이 맵리듀스 기반 빅데이터 처리 기법은 맵리듀스 프레임워크에 정의된 다양한 객체들을 활용하여 수집된 데이터들을 데이터 형태에 맞게 변환하며 로드한다. 수집되는 빅데이터들의 형태는 음악 제작 사용자의 이용 패턴 로그, 스트리밍, 바이너리 및 다양한 텍스트 형태의 파일들이다. 변환된 데이터들은 앞서 설명한 맵 함수 실행단계와 Shuffle and Sort 단계, 리듀스 함수 실행단계를 거쳐 처리된다.

2.3 요구사항 분석

본 절에서는 앞서 설명한 맵리듀스와 기존의 맵리듀스 기반 빅데이터 처리 기법과 관련연구[9-15]에 제시하는 빅데이터 처리 기법에 대한 문제점을 도출하고, 요구사항을 분석한다.

2.3.1 데이터 저장 용량 효율성

맵리듀스 기반 빅데이터 처리 기법은 하둡 분산 파일 시스템을 통해 빅데이터를 저장하기 때문에 많은 양의 멀티 노드와 저장 공간을 필요하다. 특히 하둡 분산 파일 시스템은 데이터의 안정성을 위해 기본적으로 3중화한다. 즉, 하둡 분산 파일 시스템에 저장되는 데이터는 동일하게 2개가 추가적으로 저장된다. 따라서 빅데이터를 처리하기 위해 하둡 분산 파일 시스템에 저장되는 데이터들은 많은 양의 인프라 자원을 사용하기 때문에 하둡 분산 파일 시스템이 적용된 서버들의 효율적인 인프라 자원 사용이 요구된다.

2.3.2 빅데이터 처리 시간

빅데이터 처리 시간은 수집된 빅데이터를 처리 및 분석하기 위해 사용되는 총 시간으로 빅데이터 처리 시스템에서 매우 중요한 요소이다. 따라서 앞서 설명한 데이터 저장 용량 효율성에 맞게 구현된 빅데이터 처리 기법은 기존 빅데이터 처리 기법에 비해 성능적인 측면에서 부족하지 않아야 한다.

3. 제안하는 빅데이터 처리 기법

제안하는 맵리듀스 기반 빅데이터 처리 기법은 앞서 분석한 요구사항에 적합하도록 빅데이터 변환 단계와 빅데이터 재생성 단계로 구분하여 구현한다. 빅데이터 변환 단계와 빅데이터 재생성 단계는 맵리듀스가 실행되기 전에 수행한다. 따라서 빅데이터 처리 기법은 맵리듀스 잡을 실행하기 전에 수행하는 선행 처리 기법이다.

3.1 빅데이터 변환 단계

빅데이터 변환 단계는 분석을 위해 하둡 분산 파일 시스템에 저장되는 빅데이터들의 용량을 줄이기 위한 데이터 압축 알고리즘과, 빅데이터 처리 기법을 일원화하기 위해 저장된 데이터의 형태를 변환하는 데이터 변환 알고리즘으로 구성된다.

3.1.1 데이터 변환 알고리즘

데이터 변화 알고리즘은 데이터 압축 알고리즘의 압축 효율 향상 및 저장되는 데이터 낭비를 최소화하기 위해 처리할 데이터들을 시퀀스 파일 형태로 변환한다. 시퀀스 파일은 바이너리 키와 값으로 형태로 구성되는 파일 포맷으로 하둡을 통한 데이터 처리에 효율적이다. 또한 시퀀스 파일은 생성된 키를 통해 값에 존재하는 데이터를 압축하고 해제하기에 적합한 구조이다. Fig. 3은 데이터 변환 알고리즘의 Pseudo Code를 나타낸다.

```
1   //Create MapReduce Job
2   Job job = new Job();
3
4   //Set Data Translate Algorithm
5   job.setJarByClass(ConvertAlgorithm);
6
7   //Load Data for Data Analysis
8   FileOutputFormat.setOutputPath(job,
9                       new Path(args[1]));
10
11  //Translate Sequence File
12  job.setOutputFormatClass(SequenceFile);
```

Fig. 3. Pseudo code of the data translate algorithm

데이터 변환 알고리즘은 맵리듀스의 설정 및 기본 실행을 위한 드라이버를 활용한다. 데이터 변환 알고리즘은 첫 번째로 맵리듀스 잡을 생성하여 다양한 환경 설정들을 정의한다. 특히 데이터 변환 알고리즘은 생성된 맵리듀스 잡에 데이터 변환 알고리즘을 선언하고, 하둡 분산 파일 시스템에 저장된 분석 데이터를 로드한다. 이 때 로드되는 분석 데이터의 파일 포맷은 의미 없는 키를 가진 키와 값 형태이다. 마지막으로 로드된 결과는 데이터 압축 알고리즘을 적용하기 위해 시퀀스 파일 포맷으로 변경한다.

3.1.2 데이터 압축 알고리즘

데이터 압축 알고리즘은 앞서 제안한 데이터 변환 알고리즘을 통해 하둡 분산 파일 시스템에 생성된 데이터를 Snappy 코덱을 통해 압축하여 저장되는 빅데이터들의 용량을 줄인다. Snappy 코덱은 리눅스에서 많이 사용되는 Gzip과 유사하며, 특히 다양한 압축 알고리즘에 비해 압축해제에 대한 처리속도 및 데이터의 압축률이 우수하다. Snappy 코덱은 다른 압축 알고리즘과 달리 하둡 분산파일시스템의 특징인 블록단위 저장방식에 적합한 압축 방식을 제공한다. Fig. 4는 데이터 압축 알고리즘의 Pseudo Code을 나타낸다.

```
1   //Set Configuration for Algorithm
2   FileOutputFormat.setCompressOutput();
3
4   //Set SnappyCodec
5   FileOutputFormat.setOutputCompressor
6                   (job, SnappyCodec);
7
8   //Set Compression Type
9   SequenceFile.setCompressionType(
10          job, CompressionType.BLOCK);
```

Fig. 4. Pseudo Code of the data compress algorithm

데이터 압축 알고리즘은 압축 알고리즘을 사용하기 위한 설정을 시작한 뒤, 압축 코덱을 정의한다. 압축 코덱은 Snappy 코덱의 블록 방식으로 정의한다. 블록 방식은 파일 헤더의 정보 중 레코드 번호를 제외한 파일 헤더와 키, 값 등 모든 정보를 압축한다.

3.2 빅데이터 재생성 단계

빅데이터 재생성 단계는 앞서 설명한 빅데이터 변환 단계를 통해 생성된 압축된 시퀀스 파일을 로드하여 맵리듀스에서 처리할 수 있는 파일로 새롭게 생성한다. 즉, 빅데이터 재생성 단계는 압축된 빅데이터의 압축 해제하고, 맵리듀스에서 그 데이터를 로드하여 빅데이터 처리를 할 수 있도록 하는 역할을 수행한다. Fig. 5는 데이터 재생성 알고리즘의 Pseudo Code를 나타낸다.

```
1   //Create MapReduce Job
2   Job job = new Job();
3
4   //Set Re-Create Algorithm
5   job.setJarByClass(ReCreateAlgorithm);
6
7   //Load Data for Data Analysis
8   FileOutputFormat.setOutputPath();
9   job.setInputFormatClass(SequenceFile);
10
11  //Create new File for No Compression
12  job.waitForCompletion(true)
```

Fig. 5. Pseudo Code of the data re-create algorithm

데이터 재생성 알고리즘은 앞서 설명한 데이터 변환 알고리즘과 같이 맵리듀스의 설정 및 기본 실행을 위한 드라이버를 활용한다. 데이터 재생성 알고리즘은 맵리듀스 잡을 생성하여 다양한 환경 설정들을 정의 한 뒤, 압축된 빅데이터를 로드한다. 그 뒤 데이터 재생성 알고리즘은 압축된 빅데이터를 압축 해제를 수행하고 맵리듀스에서 처리가 가능한 시퀀스 파일 포맷으로 변경한다.

4. 성능평가

본 장에서는 제안하는 맵리듀스 기반 빅데이터 처리 기법을 성능평가를 수행한다. 성능평가 항목은 앞서 관련 연구에서 도출한 요구사항을 기반으로 데이터 저장 용량 효율성과 빅데이터 처리 시간을 분석한다.

성능평가를 위한 환경 구성은 아파치와 같은 웹서버에서 발생되는 웹 로그를 기반으로 빅데이터를 처리하며, 하둡을 기반으로 분산 환경을 구축하여 검증한다. 하둡 기반 분산 환경은 서버들을 활용하여 3개의 데이터 노드와 1개의 네임노드 및 관리노드로 구성한다. Table 1은 성능평가를 위한 주요 환경 구성을 나타낸다.

Table 1. Environment for performance analysis

Type	Description
Big Data Computing Environment	Hadoop 2.8.2 Distributed Mode : 5ea
Server(Node)	Dell PowerEdge R730 - Intel Xeon 3.4Ghz 8Core 64G RAM
Analysis Data : Web Log	Compressed Web Log (Proposed Scheme)
	Web Log (Existing Scheme)
Using Mapreduce Job	Mapreduce Job for IP Analysis

4.1 데이터 저장 용량 효율성 분석

데이터 저장 용량 효율성 분석은 분산 모드로 설정된 하둡 분산 파일 시스템에 압축된 시퀀스 파일 형태의 로그와 일반 로그의 파일 증가에 따른 낭비되는 저장 용량 분석을 수행한다.

파일 증가에 따른 낭비되는 저장 용량 분석은 배치 잡을 통해 하둡 분산 파일 시스템에 주기적으로 100MB 용량을 가진 웹 로그를 업로드하여 하둡 분산 파일 시스템

의 블록에 낭비되는 용량을 분석한다. 파일 증가에 따른 저장 용량 분석의 비교 대상은 일반적인 하둡 분산 파일 시스템(이하 기존 기법, E-Scheme)과 업로드된 파일을 압축된 시퀀스 파일 형태로 변환한 하둡 분산 파일 시스템(이하 제안 기법, P-Scheme))이다. [Table 2]는 저장 용량 분석을 위한 정의한 환경 구성을 나타낸다.

Table 2. Defined system parameter

Type	Value	Description
n	1-20(min - max)	Count of Upload File
d	Default Value 64MB	Block Size of Hadoop Distributed File System
s	100MB	Size of Web Log
er	0.6%	Average Compression Rate

Table 2의 정의에 따라 기존 기법의 낭비되는 저장 용량 분석은 식(1)과 같다.

$$\sum_{n=1}^{\infty} n(d-(s \bmod d)) \qquad \text{식(1)}$$

제안 기법의 낭비되는 저장 용량 분석은 식(2)와 같다.

$$\sum_{n=1}^{\infty} (d-((ns) \bmod d)) \times er \qquad \text{식(2)}$$

Fig. 6과 Table 3은 파일 증가에 따른 낭비되는 저장 용량 분석 결과를 나타낸다.

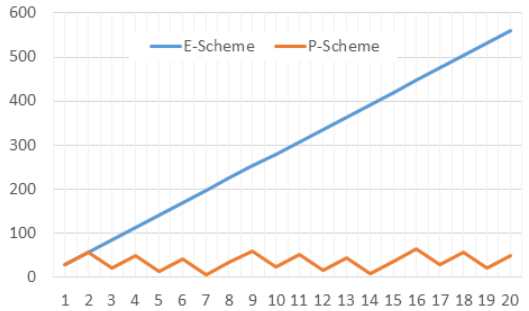

Fig. 6. Comparison of the wasted data store rate

Table 3. Result of the wasted data store rate

n	1-5	6-10	11-15	16-20	Total
E-Scheme	420	1,120	1,820	2,250	5,880
P-Scheme	98.4	96	93.6	129.6	417.6
Efficiency Rate	77%	91%	95%	95%	93%

Fig. 6과 Table 3의 분석 결과에 따르면, 제안 기법에서 낭비되는 저장 용량은 417.6MB, 기존 기법은 5,880MB로 제안하는 기법은 기존 기법에 비해 약 93%의 저장 용량의 효율성을 보인다. 이러한 결과가 발생한 원인은 저장되는 여러 파일들이 하나의 시퀀스 파일로 변환 및 압축을 수행하고, 그 파일이 블록 단위로 나눠서 저장되기 때문이다.

분석된 저장 용량 효율성은 저장되는 파일의 수가 증가하면 할수록 77%에서 95%까지 증가됨을 알 수 있다. 따라서 제안하는 기법은 파일이 증가할수록 저장 효율성이 증가된다.

4.2 빅데이터 처리 시간 분석

빅데이터 처리 시간 분석은 Table 1에 정의한 환경에 맞춰 빅데이터 환경을 구성하여 데이터 로드부터 맵리듀스 실행까지의 빅데이터 처리 시간을 측정한다. 빅데이터 처리 시간 분석에서 측정할 데이터는 기존 기법인 일반적인 웹 로그와 제안 기법의 압축된 시퀀스 파일과 압축되지 않은 시퀀스 파일이다. Fig. 7과 Table 4는 빅데이터 처리 시간 분석을 나타낸다.

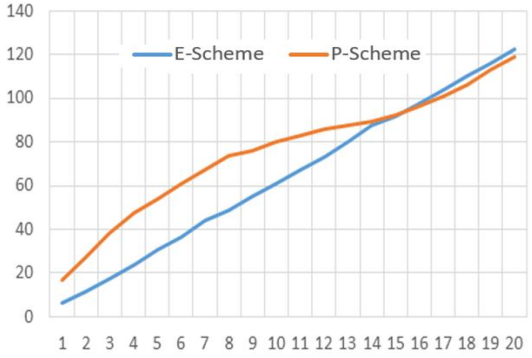

Fig. 7. Comparison of the big data processing scheme delay time

Table 4. Comparison of the big data processing scheme delay time

n	1-5	6-10	11-15	16-20	Total
E-Scheme	89.6	245.5	399.3	549.51	1283.91
P-Scheme	183.77	358.33	438.22	536	1516.32
Efficiency Rate	-105%	-46%	-10%	2%	-18%

Fig. 7과 Table 4의 분석 결과에 따르면, 제안 기법에서 빅데이터 처리 시간은 초반에 약 2배 이상 느림을 알 수 있다. 그러나 제안 기법의 빅데이터 처리 시간은 처리할 데이터 양 및 수가 증가될수록 기존 기법과의 격차는 급속도록 감소하였으며, 특히 1.5G부터 제안 기법의 빅데이터 처리 시간이 기존 기법에 비해 약 2% 이상 빠르게 처리하고 있다. 따라서 제안 기법은 많은 양과 수의 데이터를 처리할 때 보다 빠른 빅데이터 처리 성능 보였다.

5. 결론

맵리듀스는 하둡의 필수 핵심 기술로 하둡 분산 파일 시스템을 기반으로 빅데이터를 처리하는 가장 보편화되어 사용되고 있다. 그러나 기존의 맵리듀스 기반 빅데이터 처리 기법은 하둡 분산 파일 시스템에 정해진 블록의 크기대로 파일 나누고 분산된 노드들에 동일한 데이터가 중복 저장되는 특징으로 인해 네트워크 사용률 및 데이터 저장 공간 등의 인프라 자원의 낭비가 극심하다.

이에 본 논문에서는 효율적인 맵리듀스 기반 빅데이터 처리 기법 구현하였다. 제안하는 기법은 처리할 데이터를 사전에 맵리듀스에서 처리하기 적합한 데이터 형태로 변환 및 압축하여 빅데이터 인프라 환경의 저장 효율성을 증가시켰다. 또한 제안하는 기법은 저장 효율성을 중점으로 구현했을 때 발생할 수 있는 데이터 처리 시간의 지연 문제를 해결하였다.

REFERENCES

[1] H. G. Lee, Y. W. Kim & K. Y. Kim (2017), Implementation of an Efficient Big Data Collection Platform for Smart Manufacturing. *Journal of Engineering and Applied Sciences, 12(2Si),* 6304-6307. DOI: 10.3923/jeasci.2017.6304.6307

[2] B. Mahjani, S. Toor, C. Nettelblad & S. Holmgren (2016). A Flexible Computational Framework Using R and Map-Reduce for Permutation Tests of Massive Genetic Analysis of Complex Traits. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14(2),* 381-392. DOI: 10.1109/TCBB.2016.2527639

[3] Y. W. Kim & H. G. Lee (2017). Implementation of Big Data Analysis System to Prevent Illegal Sales in the Cable TV Industry. *Journal of Engineering and Applied Sciences, 12(3Si),* 6542-6545. DOI: 10.3923/jeasci.2017.6542.6545

[4] H. G. Lee, Y. W. Kim, K. Y. Kim & J. S. Choi. (2018). Design of GlusterFS Based Big Data Distributed Processing System in Smart Factory, *Journal of Korea Institute of Information, Electronics, and Communication Technology*, *11(1)*, 70-75.

[5] H. J. Park. (2016). A Study about Performance Evaluation of Various NoSQL Databases, *Journal of Korea Institute of Information, Electronics, and Communication Technology, 9(3)*, 298-305.

[6] H. G. Lee, Y. W. Kim, K. Y. Kim & J. S. Choi (2018). Design of Splunk Platform based Big Data Analysis System for Objectionable Information Detection, *Journal of Korea Institute of Information, Electronics, and Communication Technology, 11(1)*, 76-81.

[7] S. H. Kim, S. H. Chang & S. W Lee (2017). Consumer Trend Platform Development for Combination Analysis of Structured and Unstructured Big Data, *Journal of Digital Convergence, 15(6)*, 133-143.

[8] C. Y. Lee (2017). A Study on Synchronization Effect of A Multi-dimensional Event Database for Big Data Information Sharing, *Journal of Digital Convergence, 15(10)*, 243-251.

[9] Y. U. Jeong (2015). U-healthcare Service Management Scheme for Big Data of Patient Information, *Journal of Convergence for Information Technology, 5(1)*, 1-6.

[10] J. H. Ku (2017). A Study on the Platform for Big Data Analysis of Manufacturing Process, *Journal of Convergence for Information Technology, 7(5)*, 177-182.

[11] I. H. Joo (2017). Spatial Big Data Query Processing System Supporting SQL-based Query Language in Hadoop, *Journal of Korea Institute of Information, Electronics, and Communication Technology, 10(1)*, 1-8.

[12] Y. J. Baek, W. C. Jeong, S. W. Hong & J. H. Park (2017). A step-by-step service encryption model based on routing pattern in case of IP spoofing attacks on

clustering environment, *Journal of Korea Institute of Information, Electronics, and Communication Technology, 10(6)*, 580-586.

[13] E. H. Jeong & B. K. Lee. (2017). A Design of Hadoop Security Protocol using One Time Key based on Hash-chain, *Journal of Korea Institute of Information, Electronics, and Communication Technology, 10(4)*, 340-349.

[14] Y. S. Lee (2015). Authentication Method for Safe Internet of Things Environments, *Journal of Korea Institute of Information, Electronics, and Communication Technology, 8(1)*, 51-58.

[15] J. T. Seong (2017). Analysis of Signal Recovery for Compressed Sensing using Deep Learning Technique, *Journal of Korea Institute of Information, Electronics, and Communication Technology, 10(4)*, 257-267.

이 협 건(Hyeopgeon Lee) [정회원]

- 2011.03 ~ 2015.08 : 숭실대학교 컴퓨터학과 공학박사
- 2015.12 ~ 현재 : 한국폴리텍대학 서울강서캠퍼스 데이터분석과 교수
- 관심분야 : 빅데이터, 실시간 분석, 데이터분석
- E-Mail : hglee67@kopo.ac.kr

김 영 운(Young-Woon Kim) [정회원]

- 2004..09 ~ 2008.08 : 숭실대학교 컴퓨터학과 공학박사
- 2015.12 ~ 현재 : 한국폴리텍대학 서울강서캠퍼스 데이터분석과 교수
- 관심분야 : 빅데이터, 실시간 분석, 데이터분석
- E-Mail : luckkim@kopo.ac.kr

김 기 영(Ki-Young Kim) [정회원]

- 2004.03 ~ 현재 : 서일대학교 소프트웨어공학과 교수
- 관심분야 : 빅데이터, 무선인터넷, 사물인터넷
- E-Mail : ganet89@seoil.ac.kr