The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 19, No. 3, pp.105-114, Jun. 30, 2019. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2019.19.3.105
JIIBC 2019-3-14

차량 데이터 기반 빅데이터 처리 및 모니터링 시스템

Big Data Processing and Monitoring System based on Vehicle Data

신동윤*, 김주호**, 이승해***, 신동진****, 오재곤*****, 김정준******

Dong-Yun Shin*, Ju-Ho Kim, Seung-Hae Lee***,
Dong-Jin Shin****, Jae-Kon Oh*****, Jeong-Joon Kim******

요 약 4차 산업혁명의 발전에 따라 빅데이터의 기술들을 이용하여 연식이 오래된 차량들에서 확인할 수 없는 결과들을 모바일을 이용하여 즉각적으로 확인할 수 있는 시스템을 개발하였다. 먼저 OBD2 센서를 이용하여 차량의 데이터를 수집하였고 수집된 데이터를 라즈베리파이에 저장하여 라즈베리파이가 차량이 주행하는 것과 같은 상황을 두었다. 이후 차량의 데이터가 발생되면 데이터를 실시간으로 수집하고, 수집된 데이터를 여러 개의 노드를 이용해 분산저장한 뒤 시각화 하고자 하는 데이터를 가공, 정제, 처리하고 처리된 결과를 바탕으로 시각화하여 출력한다. 우리는 이와 같은 진행에 빅데이터를 이용하고 차량데이터를 빠르게 처리하여 모바일 기기를 통하여 효과적으로 확인할 수 있다.

Abstract As the Industrial Revolution progressed, Big Data technologies were used to develop a system that instantly identified the consequences of older vehicles using mobile devices. First, data from the vehicle was collected using the OBD2 sensor, and the data collected was stored in the raspberry pie, giving it the same situation that the raspberry pie was driving. In the event that vehicle data is generated, the data is collected in real time, stored in multiple nodes, and visualized and printed based on the processed, refined, processed and processed data. We can use Big Data in this process and quickly process vehicle data to identify it effectively through mobile devices.

Key Words : Vehicle Data, Big Data, Real Time, Monitoring

Ⅰ. 서 론

현재 우리는 4차 산업혁명의 시대에 살고 있다. 정보통신 기술을 기반으로 새로운 산업 시대를 대표하는 용어인 4차 산업혁명이 발전하면서 "빅데이터" 는 다양한 기술과 접목하여 함께 발전하고 있다. 빅데이터는 데이터 수집-저장-처리-분석-시각화 전반에 걸쳐 광범위한 기술을 모두 포함하는 개념이며, 최근 5V의 특징으로 불린

*준회원, 한국산업기술대학교 컴퓨터공학과 학부생
**준회원, 한국산업기술대학교 컴퓨터공학과 학부생
***준회원, 한국산업기술대학교 컴퓨터공학과 학부생
****준회원, 한국산업기술대학교 스마트팩토리융합학과 석사과정
*****정회원, ㈜진우산전 이사
******정회원, 한국산업기술대학교 컴퓨터공학과 조교수
접수일자 2019년 1월 23일, 수정완료 2019년 5월 3일
게재확정일자 2019년 6월 7일

Received: 23 January, 2019 / Revised: 3 May, 2019 /
Accepted: 7 June, 2019
******Corresponding Author: jjkim@kpu.ac.kr
Dept. of Computer Engineering, Korea Polytechnic University, Korea.

다. 첫 번째, 양(Volume)으로 기존 시스템 안에서 다루지 못할 만큼 많은 양의 데이터를 말한다. 두 번째, 속도(Velocity)로 데이터가 만들어지는 속도와 처리되는 속도가 빨라야 한다. 세 번째,다양성(Variety)로 다양한 형태의 데이터가 모여 빅데이터가 형성된다. 네 번째, 정확성(Veracity)으로 모호한 데이터 속에서 신뢰할 수 있는 데이터를 발견해야한다. 다섯 번째, 가치(Value)로 빅데이터에 존재하는 많은 데이터 속에서 유의미한 가치를 얻을 수 있는 데이터를 분별하는 것을 말한다. 빅데이터의 특징은 빅데이터의 분야가 더 발전하면서 추가적인 특징이 더해질 수 있기 때문에 온전히 정립된 특징이라고 보지 않는다[1].

우리가 생활하며 사용하는 모든 것들이 데이터가 된다. 차량을 운전하고, 웹서핑을 하며, 메시지를 통해 실시간으로 대화를 하는 모든 것들이 데이터를 통해 이루어지고 저장된다. 시간마다 생성되는 데이터 속에서 데이터를 저장, 관리, 분석하기 위해서 빅데이터 기술이 중요해지고, 최근 장비들은 빅데이터의 기술을 이용하여 서비스를 제공하는 경우가 많다. 그 사례로 스마트 에어컨, 커넥티드 카 등 생성되는 데이터에 따라 실시간적으로 처리, 분석, 시각화하여 사용자에게 사용방향을 알려주거나 사용 결과물을 보여준다. 하지만 그에 반해 오래된 장비들은 그러한 기능을 제공하지 못하고 있다. 그 사례로 연식이 오래된 차량에서는 연비는 확인할 수 없고 아날로그 계기판으로 정확한 수치가 아닌 어림잡아 연료잔량, 속도 등을 알 수 있다.

하지만 본 연구에서는 오래된 장비에서도 빅데이터 기술을 사용할 수 있다는 것을 증명하기 위하여 라즈베리파이를 이용하여서 실시간으로 차량데이터를 생성하고 빅데이터 기술을 이용해 처리, 분석 할 수 있는 방법을 제안한다.

제안하는 방법은 총 5단계로 1) OBD2 센서를 이용하여 차량의 데이터를 수집하고 라즈베리파이를 이용하여 차량이 주행하는 것과 같이 데이터 발생, 2) 발생되는 차량 데이터 수집, 3) 여러 개의 노드를 이용한 데이터 분산 저장, 4) 저장된 차량 데이터 처리, 정제, 5) 처리, 정제 결과 시각화를 수행한다. 제안 사항들을 통해 실시간적으로 차량의 정보를 확인할 수 있으며, 동시에 다른 차량과 비교하여 상관관계를 분석할 수 있는 그래프를 제공하여 빅데이터 수행 5단계의 방법의 유효성을 입증시킨다. 본 논문은 다음과 같이 구성된다. 2장에서는 관련 기술, 3장에서는 관련 연구, 4장에서는 본론, 5장에서는 결론을 기술한다.

Ⅱ. 관련 기술

1. 하둡 분산 파일 시스템(HDFS)

HDFS는 Hadoop Distributed File System의 약자로 여러 기계에 대용량 데이터를 나누어서 분산, 저장, 관리할 수 있는 시스템이다. 대용량 데이터는 기본적으로 설정되어있는 64MB 크기의 블록으로 나누어 저장된다.

HDFS는 하나의 NameNode와 다수의 DataNode로 구성되어있으며 NameNode는 HDFS내의 모든 메타데이터를 관리하고, DataNode는 데이터를 블록 단위로 저장하는 데이터 서버로 NameNode와 클라이언트의 데이터 입출력 요청을 처리한다[2].

2. Flume

Flume은 빅데이터의 데이터 수집 도구로서 많은 양의 로그 데이터를 효과적으로 수집하기 위한 분산형 소프트웨어이다. 단순하고 유연한 아키텍처를 갖추고 있으며, 사용자의 요구에 맞게 설정하여 사용할 수 있으며 기능에 따라서 전송간의 오류에 대한 대처 방법도 설정할 수 있다[3].

3. Pig

Pig는 야후 연구소에서 개발된 빅데이터 처리 도구로서 대용량 데이터 집합을 분석하는 기능이다. Pig는 하둡을 기반으로 맵리듀스 사용에 특화된 도구이며, 목적에 맞게 필요한 함수를 구현하여 더욱 고차원적으로 데이터를 처리할 수 있는 특징이 있다[3].

4. Hive

Hive는 Apache사에서 만든 빅데이터 처리 도구로서 하둡을 기반으로 동작하며 데이터를 요약하고, 질의 및 분석을 할 수 있는 기능을 제공한다. HiveQL이라 불리는 SQL과 같은 언어를 지원하며 맵리듀스의 기능을 제공한다[4].

Ⅲ. 관련 연구

1. OBD2를 이용한 자동차 정보 표시 장치 개발

본 관련 연구는 차량 제어를 위해 널리 사용되는 전자제어 장치들의 데이터를 수집하여 현재 상태를 판단하고

고장 유무 진단 및 차량 정보를 표시해 주는 시스템을 개발하는 것에 목표를 둔 연구이다. 최근에 차량에 사용되는 많은 전자 제어 장치들은 장치들 간 네트워크를 통해 서로 상호작용하며 전체 시스템을 제어하고 있다. 이 제어 장치들이 사용하는 많은 데이터들이 곧 차량의 상태들을 나타내는 정보들이다. 따라서 첫 부분과 같이 이 데이터들을 추출하여 보다 많은 정보들을 제공하는 것을 목표로 하는데 기존의 연구와는 다르게 차량의 정비소와 같은 전문적인 장소뿐만 아니라 일반 사용자들이 이용하고 있는 모바일 기기를 활용하여 정보를 제공함을 최종 목표로 둔 연구이다. 본 연구를 참조하여 본 졸업 작품 과제도 차량의 데이터 수집을 위해 모바일 어플리케이션을 활용하여 실제 차량의 데이터를 수집할 수 있게 되었다[5-7].

2. 차량정보 실시간 모니터링 기술-차량 및 운전자 정보관리 시스템

본 연구는 한국전자통신연구원에서 2007년부터 연구, 개발한 연구로서 차량과 가장 연관성이 있는 물류, 보험, 택시 등의 산업과 본 연구를 연동시켰을 때 가장 시너지가 높다고 판단하여 차량으로부터 오는 정보를 효율적으로 관리해 줌으로써 차량의 상태를 용이하게 점검하고, 운전자의 운전 패턴에 따라 안전운전, 경제운전, 환경운전을 유도하는 기술에 대한 업계의 요구들을 충족시키기 위한 연구이다. 위 연구와 동일하게 OBD-II 포트를 사용하여 차량과 직접 연결한 후 차량으로부터 차량 속도, 연료 소모량, 배출가스, 급가속, 급감속등의 차량상태 관련 정보를 실시간으로 수집한 뒤 수집된 정보는 VDMS 운행정보단말에 전송되고 수집된 정보를 이용하여 안전지수, 경제지수, 친환경 지수 등의 운전자 성향 관련 정보, 연비, CO2 배출량을 계산하여 운전자에게 시각적으로 표시해준다. 본 연구를 참조하여 본 졸업 작품에서도 속력과 연비 등의 데이터를 미리 구축해둔 서버에 전송하고 그 데이터들을 기반으로 실시간 그래프를 생성시켜 시각적으로 보여주는 기능을 삽입하게 되었다[8-10].

Ⅳ. 본론

본 논문의 목적은 빅데이터의 분산저장시스템을 활용하여 차량의 데이터를 실시간으로 처리하는 시스템을 구축하는 것이다. 구축된 시스템을 통해 차량의 소유자는 언제 어디서든 웹, 모바일 어플리케이션을 이용하여 차량의 실시간적인 상태를 모니터링할 수 시스템을 개발하는 것이다.

차량의 데이터는 OBD2 센서를 이용하여 csv 형식의 데이터로 사전에 수집을 하였고, 수집된 차량 데이터는 라즈베리파이를 이용하여 실제 자동차가 주행할 때 데이터를 발생하는 것처럼 데이터를 발생시켜준다.

표 1 라즈베리파이 차량 데이터 생성코드
Table 1. Raspberry Pi vehicle data generation code

```
public class Main extends Thread {

        public static void main(String args[]){

                FileReader fileread = null;
                FileWriter filewrite = null ;
                BufferedReader bufferread = null
;
                BufferedWriter bufferwrite = null
;
                try{
                fileread = new
FileReader("C:\\cardata.csv");
                bufferread = new
BufferedReader(fileread);

                filewrite = new
FileWriter("C:\\cardata.txt", true);
                bufferwrite = new
BufferedWriter(filewrite);
                String s;

while((s=bufferread.readLine())!=null){
                                bufferwrite.write(s);
                                sleep(1000);
                        flush();

bufferwrite.newLine();}
                }catch(Exception e){
                                e.printStackTrace();

                }finally{
                if(bufferread != null) try{

bufferread.close();}catch(IOException e){}
                                if(fileread != null)try{
                                fileread.close();
        }catch(IOException e){}
                                if(bufferwrite !=
null)try{
                                bufferwrite.close();

}catch(IOException e){}
                                if(filewrite != null)
try{
                                filewrite.close();
                                }catch(IOException
e){
    }
   }
  }
}
```

표 1은 라즈베리파이에 미리 수집한 차량의 데이터를 저장한 후 해당데이터를 차량과 같은 방식으로 발생시키기 위하여 자바의 파일 입출력을 이용한 코드이다. 해당 코드를 이용하면 차량이 주행하는 것과 같이 데이터가 1초마다 발생된다.

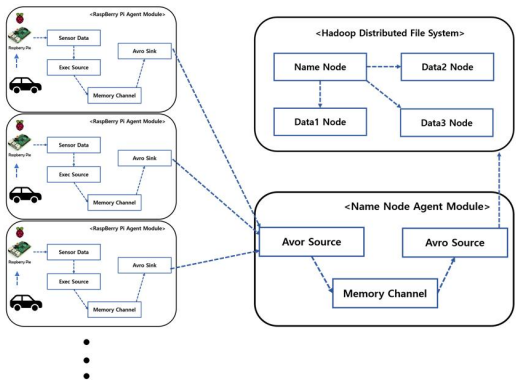

그림 1. Flume을 이용한 데이터 수집 과정
Fig. 1. Process of data collection using Flume

그림 1은 차량의 데이터가 라즈베리파이에서 발생되고, 발생된 데이터를 NameNode에 저장하기 위한 과정을 표현한 것으로 라즈베리파이에서 발생된 차량데이터는 빅데이터의 데이터수집 도구로 사용되는 Flume을 사용하여 NameNode와 통신하고 NameNode에서 설정한 위치로 데이터를 저장한다. Flume을 사용해 통신하는 것은 Step 1: 라즈베리파이 모듈, Step 2: NameNode 모듈 두 가지로 구성된다.

Step 1: 라즈베리파이 모듈

차량의 데이터를 발생시킬 수 있는 각각의 라즈베리파이에 Flume을 설치한 후 Flume을 수행할 때 참조하는 파일인 conf 파일을 조건에 맞도록 설정하여준다.

표 2. 라즈베리파이의 Flume 설정 명령어
Table 2. Flume setup command for a raspberry pi

```
cardata01.sources = execGenSrc
cardata01.channels = memoryChannel
cardata01.sinks = avroSink
# Sources
cardata01.sources.execGenSrc.type = exec
cardata01.sources.execGenSrc.command =
tail -F /home/risingsunz/Downloads/cardata.txt
cardata01.sources.execGenSrc.batchSize = 10
cardata01.sources.execGenSrc.channels =
 memoryChannel
# Channels
cardata01.channels.memoryChannel.type = memory
cardata01.channels.memoryChannel.capacity = 100000
cardata01.channels.memoryChannel.transactionCapacity =
10000
# Sinks
cardata01.sinks.avroSink.type = avro
cardata01.sinks.avroSink.hostname = 192.168.109.101
cardata01.sinks.avroSink.port = 33333
cardata01.sinks.avroSink.batch-size = 1000
cardata01.sinks.avroSink.channel = memoryChannel
```

표 2는 라즈베리파이에서 Flume을 실행하기 위해 작성한 car.conf 파일이다. 먼저 Flume은 Source, Channel, Sink의 단계로 구성된다. 이 구성에서 Source는 외부에서 이벤트를 입력받는 부분으로 각각의 라즈베리파이에서 생성되는 데이터는 데이터가 발생할 때 마다 데이터가 정해진 문서에 추가되기 때문에 차량의 데이터가 저장되는 경로인 /home/risingsunz/Downloads/data.txt를 리눅스의 tail 명령어를 통하여 수집하고 한번에 Source에 전달할 이벤트의 크기를 10으로 설정하였다. Channel은 이벤트를 임시로 저장하여 Sink로 전달하는 역할을 하며 채널은 빠르고 고성능인 메모리로 설정하였다. 데이터의 전송과정에서 여러 파일이 생기는 것을 막기 위하여 이벤트 크기는 100000개로 지정하고 트랜잭션의 크기는 10000개로 설정하였다. 마지막으로 Sinks는 이벤트를 외부로 출력하는 부분으로 NameNode로 전송되는 부분이기 전송받는 NameNode의 IP주소인 192.168.109.101과 임의의 Port 주소 33333을 한 번에 받을 수 있는 이벤트의 수를 1000개로 정의하였다.

표 3. 라즈베리파이의 Flume 실행
Table 3. Run Flume on RaspBerry Pi

```
bin/flume-ng agent -c conf/ --conf-file conf/car.conf —
name cardata01 - Dflume.root.logger=INFO,consol
```

표 3는 표 2에서 Flume을 실행하기 위해 설정한 파일의 상위 디렉토리 위치에서 car.conf를 실행하여 NameNode로 전송해 주는 명령어이다.

표 4. NameNode 의 Flume 설정 명령어
Table 4. NameNode Flume Setup Command

```
carmaster01.sources = avroGenSrc
carmaster01.channels = memoryChannel
carmaster01.sinks = HDFS
#source
carmaster01.sources.avroGenSrc.type = avro
carmaster01.sources.avroGenSrc.bind = 192.168.109.101
carmaster01.sources.avroGenSrc.port = 33333
carmaster01.sources.avroGenSrc.channels = memoryChannel
#channel
```

```
carmaster01.channels.memoryChannel.type = memory
carmaster01.channels.memoryChannel.capacity = 100000
#sink
carmaster01.sinks.HDFS.type = HDFS
carmaster01.sinks.HDFS.hdfs.path = hdfs://master:9000/sensor
carmaster01.sinks.HDFS.hdfs.fileType = DataStream
carmaster01.sinks.HDFS.hdfs.writeFormat = text
carmaster01.sinks.HDFS.hdfs.batchSize = 1000
carmaster01.sinks.HDFS.hdfs.rollSize = 0
carmaster01.sinks.HDFS.hdfs.rollCount = 10000
carmaster01.sinks.HDFS.hdfs.rollInterval = 1800
carmaster01.sinks.HDFS.hdfs.useLocalTimeStamp = true
carmaster01.sinks.HDFS.channel = memoryChannel
```

표 4는 NameNode에서 Flume을 사용하기 위해 작성한 master.conf 파일이다. 라즈베리파이의 Sink에서 출력되는 이벤트는 NameNode Flume 구조에서 Source 부분이 받게 된다. 이벤트를 수신 받을 NameNode의 IP와 Port를 지정해 주고 사용하는 Channel은 라즈베리파이와 동일하게 메모리로 구현하였다. Sink는 NameNode에서 분산저장하기 위해 HDFS로 type을 지정해 주었고 파일이 저장될 위치를 설정해 주었다. HDFS내에 쌓이게 되는 차량의 데이터의 형식은 Text, 이벤트수의 제한은 1000개, 파일을 나누는 기준은 30분으로 설정하였다.

표 5. NameNode 의 Flume 실행
Table 5. Run Flume on NameNode

```
bin/flume-ng agent -c conf/ --conf-file conf/master.conf --
name carmaster01 - Dflume.root.logger=INFO,consol
```

표 5은 NameNode에서 Flume을 실행하기 위해 표 4에서 설정한 master.conf파일이 저장된 상위 디렉토리에서 Flume을 실행하여 분산 저장 시스템인 HDFS내로 차량 데이터를 실시간으로 분산저장 하는 명령어다.

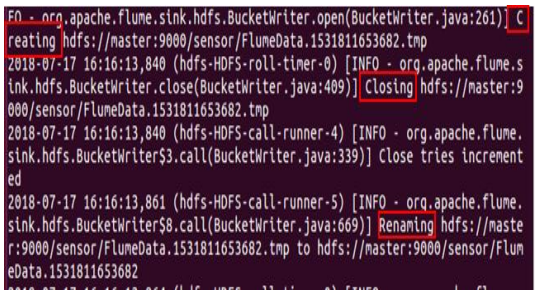

그림 2. Flume을 통한 데이터 저장 과정
Fig. 2. Process of storing data through Flume

그림 2와 같이 라즈베리파이와 NameNode가 Flume을 이용하여 통신에 성공한다면 Creating, Closing, Renaming 단계를 거치게 되며 HDFS의 지정된 위치에 데이터가 수집되는 것을 확인할 수 있다. Creating 단계에서는 데이터가 수집되는 과정에서 생기는 임시파일 즉 .tmp 파일을 생성하고 Flume의 이벤트 설정에 맞게 데이터의 수집이 완료되면 Closing 단계를 거쳐 Renaming 단계에서 .tmp 파일이 아닌 표 3에서 지정한 타입 text로 데이터가 저장된다.

그림 3. 분산 저장된 데이터 확인
Fig. 3. Verifying Distributed Saved Data

그림 7에서는 차량 데이터가 수집되어 분산저장 된 것을 확인할 수 있으며, NameNode 하둡 경로인 마스터 노드의 바로 아래 /sensor 에 데이터가 수집된 것을 확인할 수 있다.

표 6. 분산 저장된 데이터
Table 6. Distributed stored data

Longitude	Latitude	GPS Speed	Horizontal Dilution of Precision	Altitude	Bearing	G(x)	G(y)	G(z)
127.122984	36.64130243	0	3	139	0	-1.76	7.03	7.98
127.122984	36.64130244	0	3	139	0	-3.21	6.61	7.98
127.122984	36.64130243	0	3	139	0	-4.15	6.44	5.06
127.122984	36.64130241	0	3	139	0	-1.32	7.17	8.65
127.122984	36.643024	0	3	139	0	2.15	7.09	3.21
127.122984	36.643024	0	3	139	0	2.51	8.09	7
127.1229835	36.6430131	0	4	139	0	2.23	7.99	6.85

표 6은 수집된 로그 데이터를 hadoop -fs cat /sensor/* 명령어로 확인한 표이다. 수집된 차량데이터는 GPS Time, 위도, 경도, 연료소모량 등 52가지의 컬럼으로 이루어져 있으며, 실시간으로 수집되는 차량 데이터는 처리 프로세스 단계를 거쳐 다양한 데이터 중에서 의미 있는 데이터, 필요한 데이터만을 처리하여 시각화 단계를 진행할 것이다.

표 7. 데이터 처리를 위한 Pig 스크립트 실행
Table 7. Running a Pig Script to Process Data

```
cardata = LOAD '/sensor/*' USING PigStorage(',');
RisingSunZ = FOREACH cardata GENERATE

$1 AS time:chararray,
$4 AS spped:double,
$35 AS fuel:double,
$53 AS distance:double;

STORE RisingSunZ INTO '/result1';
```

표 7은 Pig 스크립트로, 수집된 데이터를 처리하는 단계이다. 차량데이터 52개의 컬럼에서 처리하여 시각화에 사용될 1번, 4번, 35번 53번 컬럼을 추출하는 작업을 진행한다.

표 8. Hive 사용을 위한 테이블 생성 및 처리
Table 8. Creating and processing tables for Hive use

```
create table car (
                time string,
                speed double,
                fuel double,
                distance double)

ROW FORMAT DELIMITED
FIELDS TERMINATED By '\t'
LOCATION '/result1';
INSERT OVERWRITE DIRECTORY '/result2'
select (distance/fuel)/3.7 from car
drop table car;
```

표 8 에서는 HiveQL을 이용하여 HDFS의 /result1 디렉토리에 저장된 데이터를 ‘,’ 단위로 분할하여 불러온다. 이후 create table명령어를 사용하여 표 9에서 Pig를 이용하여 추출한 컬럼을 알맞은 자료형에 맞춰 테이블을 생성한다. 이후 추출한 컬럼중에서 연비를 계산하기 위해 거리와 연료소모량을 연비계산법을 이용하여 연산하고 result2에 저장하여준다.

표 9. HDFS내 /result2에 저장된 결과파일 확인
Table 9. Check the results file stored in HDFS /result2

1	15.41244369	39.43	43.32
2	15.41244369	39.43	43.32
3	15.41807432	41.5	43.32
4	15.41807432	41.5	43.32
5	15.41807432	43.3	43.32
6	15.34518975	43.3	43.32
7	15.34518975	44.92	43.32
8	15.35219157	44.92	43.32
9	15.35219157	46.36	43.32
10	15.35779303	46.36	43.32
11	15.35779303	47.65	43.32
12	15.36479485	47.65	43.32
13	15.36479485	48.81	43.32
14	15.3703963	48.81	43.32
15	15.3703963	48.81	43.32
16	15.3703963	49.89	43.32
17	15.37879849	49.89	43.32
18	15.37879849	50.83	43.32
19	15.38439994	50.83	43.32
20	15.38439994	51.6	43.31

표 8까지 모든 처리과정을 완료하고 HDFS내 /result2 디렉토리를 확인하면 결과 파일이 생성된 것을 표 9에서 확인할 수 있다. hadoop -fs cat /result2/* 명령어를 이용하여 파일을 열어보면 Hive에서 쿼리를 실행했던 것과 같이 순서, 연비, 주행속도, 연료잔량의 연산된 결과가 표 9와 같이 출력되는 것을 확인할 수 있다.

표 10. 반복 처리 쉘 스크립트
Table 10. Repetitive Processing Shell Script

```
for((;;)); do

pig car.pig
cd /home/hadoop/bigdata/hive/bin
hive -f ~/car.hive

done
```

Flume은 사용자가 실시간으로 데이터가 HDFS에 저장되기 때문에 데이터가 저장되는 대로 Pig와 Hive의 처리도 지속적으로 이루어져야 한다. 따라서 표 10과 같이 Pig와 Hive의 처리 프로세스를 쉘 스크립트의 반복문을 사용하여 지속적으로 데이터를 처리되도록 구성하였다.

지금까지 수행하였던 단계를 각 라즈베리파이마다 구성을 해 주면 라즈베리파이 한 대가 차량 한 대 와 같은

역할을 수행할 수 있다. 본 논문에서는 라즈베리파이를 두 대로 구성하여 차량 두 대를 가지고 연구를 진행하였다. 처리 및 정제가 완료된 차량 데이터는 Web과 Mobile Application 환경에서 실시간으로 시각화하여 나타낸다.

Web을 배포하기 위해서 Apache Tomcat 서버와 Eclips를 연동하여 서버구축을 하였고, 구축된 서버는 이용하여 HTML과 JavaScript를 사용해 시각화한다. 시각화한 결과물은 Web, Mobile Application에서 확인할 수 있도록 배포한다. 시각화는 Step 1: Web , Step 2: Mobile Application 두 가지로 구성된다.

Step 1: Web

Web은 Eclipse에서 Tomcat을 이용하여 JSP서버를 생성하고 HTML코드와 CSS를 이용하여 홈페이지 UI들을 구성하였다. 그래프의 실시간 처리를 진행하기 위하여 비동기적으로 서버와 클라이언트가 통신할 수 있는 Ajax를 사용하였다. 각 라즈베리파이를 기준으로 연비, 주행속도, 연료 잔량 3가지 그래프를 각 한 페이지에 구성하였으며 데이터가 갱신되는 순간에 그래프가 refresh되도록 설정하였다. 마지막 페이지에는 각 페이지마다 표시된 서로 다른 차량의 그래프들을 오버레이형식으로 시각화하여 다른 운전자들과 비교할 수 있는 페이지를 구성하였다.

그림 4와 5는 라즈베리파이(CAR1)에서 생성되는 차량데이터를 기준으로 처리된 연비, 연료 잔량, 주행속도를 실시간 그래프로 나타낸 것으로 X축은 데이터의 발생 순서가 되고 Y축은 각 데이터 값을 기준으로 범주가 나타나게 된다.

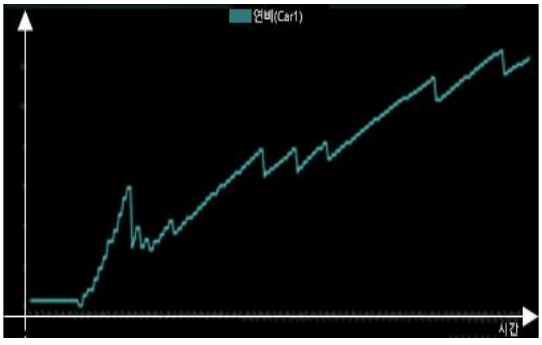

그림 4. CAR1 실시간 그래프_1
Fig. 4. CAR1 Real-Time Graphs_1

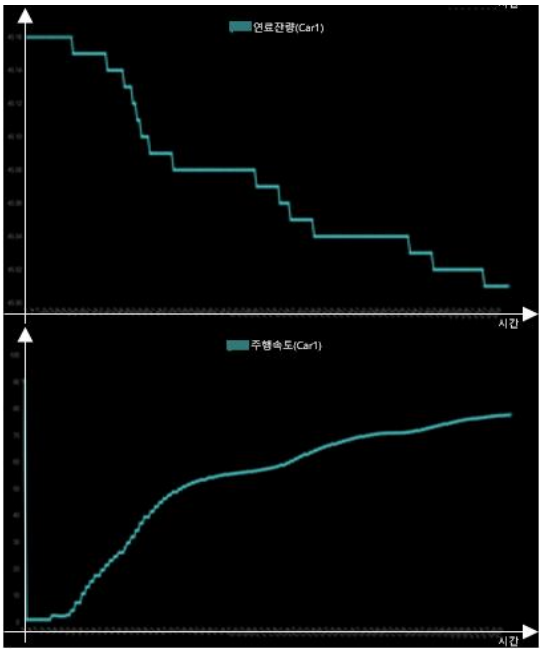

그림 5. CAR1 실시간 그래프_2
Fig. 5. CAR1 Real-Time Graphs_2

그림 6과 7은 다른 라즈베리파이(CAR2)에서 생성되는 데이터를 기반으로 앞서 설명한 내용과 같이 연비, 연료잔량, 주행속도를 실시간 그래프로 나타낸 모습이다.

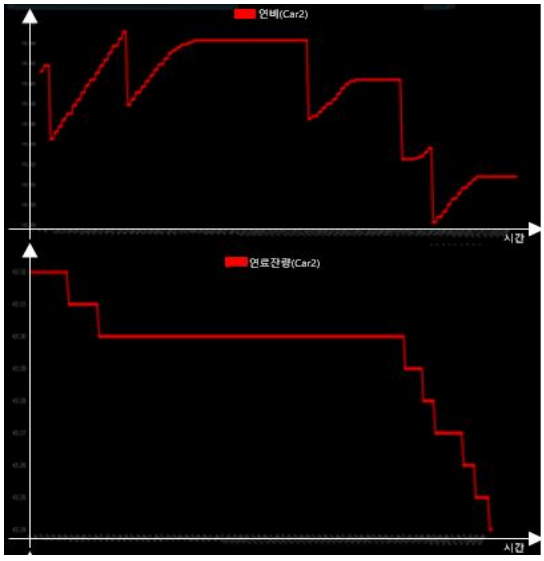

그림 6. CAR2 실시간 그래프_1
Fig. 6. CAR2 Real-Time Graphs_1

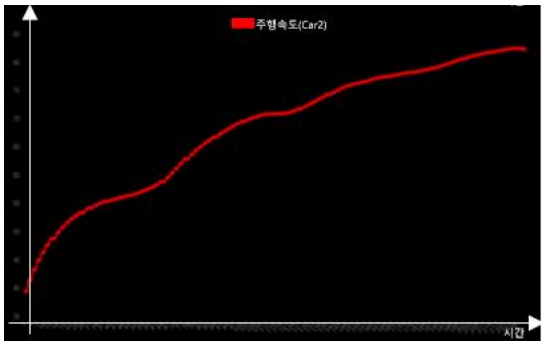

그림 7. CAR2 실시간 그래프_2
Fig. 7. CAR2 Real-Time Graphs_2

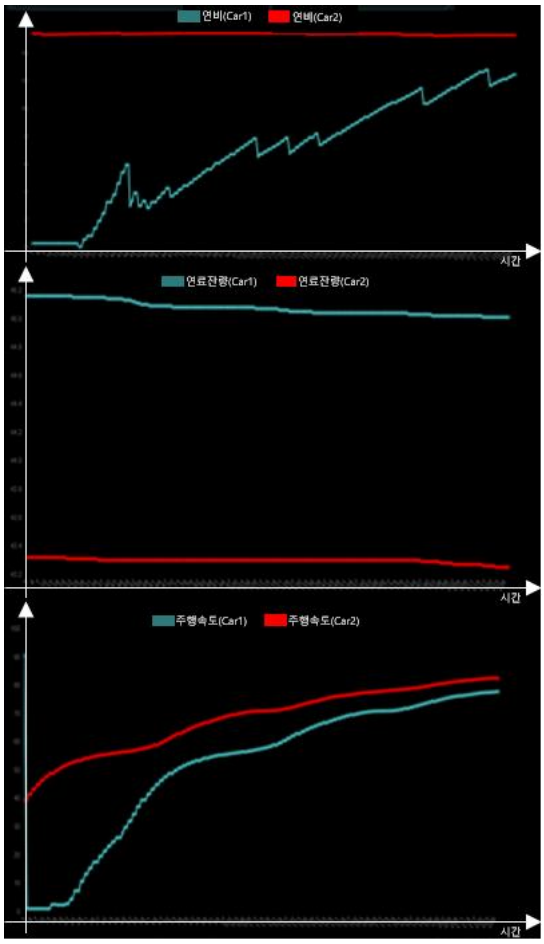

그림 8. OverLay 실시간 그래프
Fig. 8. Overlay Real-Time Graphs

그림 4, 5와 그림 6, 7에서 표현하였던 그래프들의 상관관계를 한 번에 확인할 수 있도록 그림 8에서 나타나는 것과 같이 두 개의 차량 그래프를 한 화면에 출력하여 서로를 비교할 수 있도록 구성하였다.

Step 2: Mobile Application

어플리케이션은 Android Studio를 사용하여 제작하였다. Step 1에서 구현된 웹 홈페이지를 웹 뷰 형식으로 제작하였고 웹 홈페이지를 제작하는 단계에서 그래프의 크기를 사용자가 사용하는 기기의 화면 크기에 맞게 자동으로 변화하도록 설정하여 그래프의 크기는 스마트폰 화면에 맞춰져 있어 따로 설정하지 않았다.

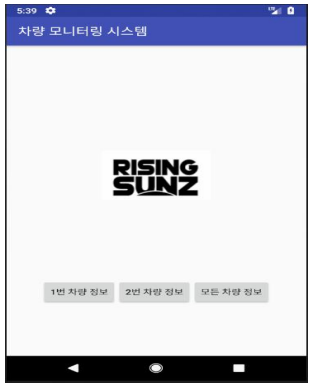

그림 9. 어플리케이션 메인화면
Fig. 9. Application main screen

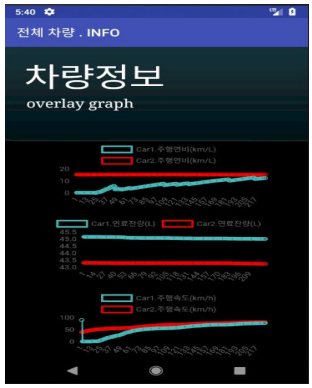

그림 10. 실시간 정보 페이지
Fig. 10. Real-time information page

그림 9는 어플리케이션의 메인화면으로 실시간 정보를 확인하기 위하여 해당 차량의 버튼을 누르면 그림 8의 화면으로 전환된다. 그림 10에서는 시간에 따라 실시간으로 그래프가 변화하는 것을 확인할 수 있다.

V. 결 론

본 논문에서는 실제 차량에서 발생하는 데이터를 수집하였고, 라즈베리파이를 이용하여 차량이 주행하는 것과 같은 상황을 만들어 데이터를 발생시켰다. 이후 발생된 데이터를 실시간으로 수집, 저장, 처리, 시각화를 하는 과정을제시하였다. 이러한 제안 사항을 통하여 신형 차량에서는 계기판을 통하여 주행 연비, 연료잔량 등이 시각적으로 잘 표시가 되지만 연식이 오래된 차량의 경우에는 아날로그적 계기판으로 실질적으로 주행 연비가 얼마나 나오는지 연료의 잔량의 수치가 얼마인지 알기가 힘들다는 점을 해결할 수 있고, 나와 같은 차량을 운전하는 사람들을 비교하고 더 나아가 이상탐지를 하여 빅데이터를 이용한 서비스를 제공할 수 있다. 하지만 본 논문에서 개선해야할 점은 차량의 정보를 나타내는 그래프의 개수가 3개인것과, 차량의 대수가 2대인 것으로 차량의 대수와 시각화 하고자하는 데이터가 증가할수록 데이터를 수집하고 처리하는 과정에서 컴퓨터의 과부하가 발생하였다. 이러한 방법은 빅데이터를 전문적으로 처리할 수 있는 고 사양컴퓨터를 사용한다면 2-3개의 항목이 아닌 더 많은 항목과 더 많은 차량을 비교할 수 있다. 이러한 문제가 개선된다면 빅데이터의 분석을 조금 더 효율적으로 진행할 수 있다.

References

[1] Hong-Kyu Park, Num-Hun Park, "Efficient Dynamic Bloom Filter Method in Big Data Environment," Journal of KIIT, Vol. 17, No. 2, pp. 13-20, Feb, 2019.

[2] Dong-Jin Shin, Jong-Min Eun, Ho-Geun Lee, Myoung Gyun Lee, Jeong-Min Park, Jeong-Joon Kim, "Big Data-based Log Collection and Analysis in IoT Environments," Journal of Engineering and Applied Sciences, Vol. 13, No. 5, pp. 1064-1072, May 2018.

[3] Dong-Jin Shin, Ho-Geun Lee, Jong-Min Eun, Jeong-Joon Kim, Jeong-Min Park, "Bigdata-based Anomaly Detection Technology in Web Services Environment," Journal of Asia-pacific Multimedia Services Convergent with Art, Humanities, and Sociology, Vol. 8, No. 4, pp. 231-250, Apr, 2018.

[4] Dong-Jin Shin, Ji-Hun Park, Ju-Ho Kim, Kwang-Jin Kwak, Jeong-Min Park, Jeong-Joon Kim, "Big Data-based Sensor Data Processing and Analysis for IoT Environment," The Journal of the Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 19, No. 4, pp. 117-126, Feb, 2019.

[5] Mi-Jin Kim, "A Development of EURO 6 Smart Cluster System Using OBD-II Standard Interfaces and Smart Cluster Interface" Transactions of KSAE, Vol. 26, No. 1, pp. 85-96, 2018.

[6] Seong-Hee Lee, Seong-Hyung Lee, Sang-Moon Lee, Seung-Hoon Hwang, "IoT Equipment Implementation for OBD Car Diagnostic Information," The Journal of Korean Institute of Communications and Information Sciences, Vol. 41, No. 12, pp. 1851~1857, 2016.

[7] Da-Woon Jeong, Jae-hyun Nam, Jong-wook Jang, "A Implementation of motorcar consumption diagnostic management iPhone based software with OBD-II and WiFi network," Journal of the Korea Institute of Information and Communication Engineering, Vol. 15, No. 11, pp. 2347~2352, Nov, 2011.

[8] Min-Young Kim, "Design and Implement a Smart Automobile Self-Diagnosis System based on The Driving information," Journal of the Korea Institute of Information and Communication Engineering, Vol. 17, No. 9 pp. 2153~2159, Sep, 2013.

[9] Won-Gok Lee, Ji-Hun Back, June Ahn, Jin-Ku Choi, 2011, "Design and Implementation of Vehicle Real-Time Tracking/Management System," The Journal of Korean Institute of Information Technology, Vol. 9, No. 8, pp. 41~51.

[10] In-taek Jung, Kyu-soo Chong, "Development of Information Technology Infrastructures through Construction of Big Data Platform for Road Driving Environment Analysis," Journal of the Korea Academia-Industrial cooperation Sociey(JKAIS), Vol. 19, No. 3, pp. 669-678, 2018.

]저 자 소 개

신 동 윤(준회원)

•Dongyun Shin is currently studying at Korea Polytech University for a and is attending computer engineering. His research interests include Big Data, Data Processing, etc.

김 주 호(준회원)

•JuHo Kim is currently studying at Korea Polytech University for a and is attending computer engineering. His research interests include Big Data, Data Processing, etc.

이 승 해(준회원)

•Seung-Hae Lee is currently a student at the Korea Institute of Industrial Technology. His research interests include Big Data, Data Processing, etc.

신 동 진(준회원)

•Dong-Jin Shin received his BS in Engineering at Korea Polytechnic University in 2018. He is currently a Master's course in the department of Smart Manufacturing Engineering at Korea Polytechnic University. His research interests include Big Data, Internet of Things(IoT), Network Security, etc.

오 재 곤(정회원)

•Jae-Kon Oh received his BS and MS at Kwangwoon University in 1994 and Ajou University in 2005, respectively. In 2017, he received his PhD in at Chonbuk University. He is currently a CEO at SEINSystems. His research interests include Database Systems, BigData, Semantic Web, Geographic Information Systems (GIS) and Ubiquitous Sensor Network (USN), etc.

김 정 준(정회원)

•Jeong Joon Kim received his BS and MS in Computer Science at Konkuk University in 2003 and 2005, respectively. In 2010, he received his PhD in at Konkuk University. He is currently a professor at the department of Computer Science at Korea Polytechnic University. His research interests include Database Systems, BigData, Semantic Web, Geographic Information Systems (GIS) and Ubiquitous Sensor Network (USN), etc.