ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

대용량 데이터 분석을 위한 맵리듀스 기반의 이상치 탐지☆

Outlier Detection Based on MapReduce for Analyzing Big Data

홍 예 진1 나 은 희1 정 용 환2 김 양 우1*
Yejin Hong Eunhee Na Yonghwan Jung Yangwoo Kim

요 약

가까운 미래에는 빅데이터의 많은 부분을 IoT 데이터가 차지할 것이라는 전망이 나오고 있다. 그에 따라, IoT 데이터의 많은 부분을 차지하는 센서 데이터에 관한 관심과 연구 또한 활발하게 진행되고 있다. 여러 분야에서 활용되고 있는 센서 데이터는 분석할 때 실제와는 다른 값인 이상치를 포함하게 되면 정확한 분석이 어려우며, 왜곡된 결과가 도출되어 활용할 수 없는 경우가 생긴다. 따라서 본 논문에서는 정확한 결과를 도출하기 위해 수집된 원자료를 분석하기 전에 이상치 탐지 및 제거를 하였다. 또한, 점점 늘어나고 있는 대용량의 데이터를 빠르게 처리하기 위해 메모리 접근방식인 스파크를 사용한 분산처리환경에서 처리하였다. 맵리듀스 기반의 이상치 탐지 및 제거는 총 4단계로 나누어 구현하였으며, 각 단계를 매퍼와 리듀스로 구현하였다. 제안한 기법의 평가를 위해서 3가지 환경에서 비교하였으며, 그 결과 이상치 탐지 및 제거를 하고자 하는 데이터의 용량이 커질수록 스파크를 이용한 분산처리환경에서의 처리가 가장 빠르다는 결과를 얻었다.

☞ 주제어 : 빅데이터, 이상치, 맵리듀스, 분산처리, 스파크

ABSTRACT

In near future, IoT data is expected to be a major portion of Big Data. Moreover, sensor data is expected to be major portion of IoT data, and its' research is actively carried out currently. However, processed results may not be trusted and used if outlier data is included in the processing of sensor data. Therefore, method for detection and deletion of those outlier data before processing is studied in this paper. Moreover, we used Spark which is memory based distributed processing environment for fast processing of big sensor data. The detection and deletion of outlier data consist of four stages, and each stage is implemented with Mapper and Reducer operation. The proposed method is compared in three different processing environments, and it is expected that the outlier detection and deletion performance is best in the distributed Spark environment as data volume is increasing.

☞ keyword : Big Data, Outlier, MapReduce, Distributed Processing, Spark

1. 서 론

IoT(Internet of Things)가 발전함에 따라 센서로부터 측정된 데이터가 대량으로 생겨나고 있다. 현재까지는 IoT 기술에 의해 생성된 데이터가 빅데이터의 큰 부분을 차지하진 않았으나, HP의 예측에 따르면 2030년에는 IoT 데이터의 양이 약 1조(GB)에 이르며 상당히 많은 부분을 차지할 것으로 예측되고 있다.[1] 수집된 IoT 데이터는 실시간 분석을 통해 기상 예측, 텔레메틱스 시스템, 제조 공정 시스템의 상태를 파악하는 등 많은 분야에 활용되고 있다. 이와 같은 분야에서는 실시간으로 수집되는 데이터를 분석한 뒤, 시스템에 분석 결과를 다시 적용시키기 때문에 정확하고 빠른 데이터의 분석이 요구된다. 그러나, 이러한 데이터를 분석할 때 실제와는 다른 값인 이상치를 포함할 경우에는 왜곡된 결과를 얻게 된다. 센서 데이터의 이상치는 비정상적인 자료로써 주로 센서의 관리 부족이나, 센서가 고장 난 상태로 잘못 측정이 되었을 때 발생할 수 있다.[2] 정확한 분석 결과를 도출하기 위해서는 수집된 원자료(Raw Data)의 데이터를 분석하기 전에 이상치를 탐지하여 제거하는 과정이 필수적이다. 또한, 점점 방대해지는 양의 데이터를 단일노드의 환경에서 처리하기에는 시간적인 제약이 존재한다. 따라서 본 논문에서는 대용량 데이터를 빠르게 처리하기 위하여 맵리듀스(MapReduce)[3] 기반의 분산처리환경에서 이상치

1 Department of Information and Communication Engineering, Dongguk University, 30, Pildong-ro 1-gil, Jung-gu,, Seoul, 04620, Korea.
2 Korea Institute of Science and Technology Information, Korea Advanced Institute of Science, 245, Daehak-ro, Yuseong-gu, Daejeon, 34141, Korea.
* Corresponding author (ywkim@dongguk.edu)
[Received 15 October 2016, Reviewed 8 November 2016, Accepted 30 November 2016]
☆ 본 연구는 한국과학기술정보연구원에서 수행하는 (소프트웨어 기반의 첨단과학기술 연구망 구축과 서비스)사업의 위탁연구로 수행되었습니다.

탐지 및 제거를 수행하고자 한다. 이와 더불어 대표적인 빅데이터 처리 기술인 하둡(Hadoop)[4]은 디스크 접근 횟수가 빈번하여 센서로부터 입력되는 대량의 데이터를 빠르게 분석하기에 적합하지 않다고 판단[5]되기 때문에 해당 실험은 메모리에 접근하여 데이터를 처리하는 BDAS(Berkely Data Analytics Stack)[6] 방식으로 구현하였다. 맵리듀스 기반의 이상치 탐지 및 제거는 총 4단계로 나누어 각 단계별로 매퍼(Mapper)와 리듀서(Reducer)로 구현하였으며, 이상치 탐지를 위한 어림값을 구하는 방법으로는 가중이동평균 분석법을 사용하였다.

본 논문의 2장에서는 이상치의 정의와 이상치 탐지 방법 그리고 BDAS의 특징에 관하여 살펴본다. 3장에서는 이상치 탐지를 위한 실험환경을 구축하고, 이상치 처리를 위해 각 단계를 매퍼와 리듀서를 구현하였다. 4장에서는 구축된 BDAS환경에서 맵리듀스 기반의 이상치 탐지 및 제거를 수행한 뒤, 그 결과를 살펴본다. 마지막으로 5장에서는 실험의 결과와 3장과 4장의 연구를 바탕으로 본 연구의 가치와 활용방안을 논의하고 향후 연구에 필요한 사항을 살펴본다.

2. 관련 연구

2.1 이상치 특징

이상치란 비정상적인 자료이다. 또한, 나머지 데이터들과는 다른 독특한 존재라고 정의되기도 한다. 이처럼 기존의 연구에서 정의하는 바와 같이 이상치는 데이터 집합에서 나머지 데이터와 연관성이 없는 별개의 값이다. 센서 데이터의 이상치는 주로 센서에 대한 주기적인 관리의 부족 또는 환경의 급작스런 변화나, 센서가 고장난 상태로 잘못 측정되었을 때 생겨난다.[2] 이와 같은 이유로 발생하는 이상치는 종종 데이터가 담고 있는 의미를 왜곡시킨다. 데이터를 분석하여 활용하는 데이터마이닝(DataMining)의 경우, 이러한 이상치를 포함하게 되면 데이터가 갖고 있는 의미와는 다른 결과가 나올 수 있다. 따라서 정확한 데이터마이닝을 함에 있어서 전 처리 단계의 이상치 탐지 및 제거는 필수적이다.

2.2 이상치 탐지 및 제거 방법

센서로부터 연속적으로 발생한 이상치는 원인이 다양함으로 전문가의 확인이 필요하지만, 간헐적으로 발생한 이상치의 경우에는 주변의 데이터를 분석하여 탐지 후, 제거가 가능하다. 따라서 본 논문에서는 센서로부터 추출된 데이터를 활용한 분석 시 보다 정확한 결과 도출을 위해 간헐적으로 발생하는 이상치를 탐지하고 제거하는 연구를 진행하였다.

현재 이상치를 탐지하고 제거하는 방법에 관하여 많은 연구가 진행되고 있으며, 이에 따라 다양한 이상치 탐지 및 제거 방법이 존재한다[7-9].

이상치 탐지 및 제거는 데이터의 형태를 변환하는 첫 번째 단계를 시작으로 가시적 탐지, 가중이동평균 분석법을 이용한 어림값 측정, 그리고 통계 분석을 통한 이상치 제거 등 4단계에 걸쳐서 이루어진다. 이상치 탐지 및 제거의 첫 번째 단계는 원자료를 분석에 알맞은 형태로 변환하는 단계이다. 두 번째 단계는 가시적 탐지 단계로서 데이터가 가질 수 있는 제한적인 범위의 값 외에 다른 값을 가지는 데이터를 제거하는 단계이다. 대략적인 데이터의 범위를 가정하고, 범위의 값 외에 다른 값을 가지는 데이터를 제거한다. 세 번째 단계는 간헐적으로 나타나는 이상치를 탐지하기 위한 단계이다. 측정된 데이터를 분석하여 어림값을 구하고, 각 어림값의 유의수준을 설정하기 위하여 표준편차를 구한다. 이때, 어림값을 구하기 위한 방법으로는 통계 분포를 분석하여 활용하는 통계기반 접근법[7], 블록 껍질(Convex hull)의 경계에 있는 데이터를 이상치로 탐지하는 깊이 기반 접근법[8], 주변 데이터들 사이의 거리를 측정하여 이상치를 판별하는 거리 기반 접근법[9] 등 여러 방법이 있다. 시계열 데이터의 경우 앞서 측정된 데이터의 영향을 많이 받기 때문에 본 논문에서는 가중이동평균 분석법(Weighted moving average analysis)[10]을 사용하여 어림값을 구하였다. 가중이동평균 분석법은 데이터의 어림값을 구할 때 가장 최근의 값에 더 많은 가중치를 부여하는 방법이다. 그 후, 앞서 구한 어림값의 표준편차를 이용하여 이상치를 구별하기 위한 유의수준을 설정하였다. 유의수준은 Grubbs진단기법에 따라 95%로 설정하였다.[11] 마지막으로, 네 번째 단계에서는 원자료와 어림값을 비교하여 원자료의 값이 유의수준 이내에 존재하지 않는다면 이를 이상치로 판단하고 제거한다.

2.3 BDAS

BDAS는 Berkeley대학의 AMPLab에 의해 만들어진 오픈 소스 소프트웨어 스택이다.[6] BDAS는 하둡의 단점을 극복하기 위한 대안으로 제시되었으며, 스파크(Spark)를 데이터 처리의 핵심으로 구성하여 대용량 데이터를 빠르게 처리하도록 하는 것이 목적이다. 본 논문에서 사용한

주된 요소로 메모리나 CPU 등의 자원을 관리하는 메소스(Mesos)[12], 분산 처리 환경에서 데이터를 저장하는 플랫폼인 HDFS[4], 메모리 기반으로 데이터를 실시간으로 처리 하는 스파크[13]를 사용하였다. 이 외에도 필요에 따라서 BDAS의 구성 요소를 추가적으로 사용할 수 있다.

2.4 스파크

하둡의 맵리듀스는 매퍼와 리듀스를 번갈아 반복하면서 중간 결과 값을 스토리지에 저장하기 때문에 스토리지로의 불필요한 접근이 많다. 때문에 하둡은 낮은 지연시간을 필요로 하는 작업을 위해 사용하기에는 적합하지 않다.[5] 반면에 스파크는 RDD(Resilient Distributed DataSet)[14]라는 데이터를 집합으로 로드(load)하여 Mapper, Reducer, JOIN, GROUP BY 와 같은 필터 등 임의의 연산자를 사용하여 작업을 처리할 수 있는 프로그래밍 모델이다. 이를 통해 스트리밍 데이터 처리, 질의문 등을 효율적으로 처리할 수 있다. 또한, 스파크는 이런 연산자들에 의해 생성된 데이터를 메모리 내에 저장 할 수 있게 한다. 이로 인해 스파크를 이용하여 작업을 처리할 때 잦은 디스크 접근을 피할 수 있게 되어 빠른 데이터 처리가 가능하다.

3. 이상치 탐지 및 제거를 위한 환경 설계 및 구현

3.1 실험환경 설계 및 구현

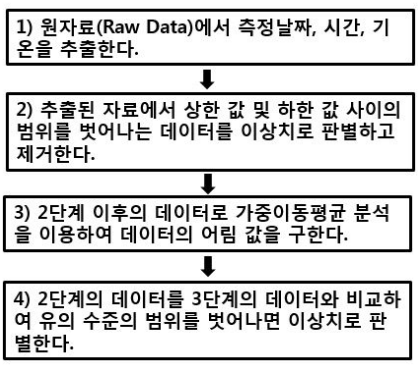
본 논문에서는 분산처리하기 위하여 가상화된 3대의 노드에서 BDAS환경을 설계하고 구축하였다. BDAS의 구성요소 중 메모리와 디스크 등의 자원 관리를 위한 용도로 메소스를 사용하고, 데이터 저장을 위해서는 HDFS를 사용했으며, 이상치 탐지 및 제거를 하기 위하여 스파크를 사용하였다.

VMWare로 가상화된 3대의 노드를 구성하여 실험 환경을 구축하였으며, 각 노드는 CentOs 6.4에서 메소스 0.22, 스파크 1.3.1, 하둡 1.2.1 버전으로 설치하고 진행하였다. 3대의 노드 중 1대의 노드는 마스터(Master) 노드로 사용하고, 나머지 2대의 노드는 각각 슬레이브(Slave) 노드로 사용하여 마스터-슬레이브 구조로 설계하였다. 마스터 노드에 4GB의 메모리를 할당하고, 나머지 2대의 슬레이브 노드에도 각각 4GB의 메모리를 할당하였다. 마스터 노드에서 작업을 생성하여 슬레이브 노드에게 할당하면, 2대의 슬레이브 노드에서 각 작업을 수행한다. 본 논문에서는 제안한 기법의 성능을 비교하기 위해 각각 다른 3가지 방식으로 환경을 구축하여 실험하였다. 그 중 본 논문에서 제안한 가상화된 3대의 노드에서 스파크를 사용한 분산처리환경은 (그림 1)과 같고, 다른 방식의 환경과 구분을 위하여 '환경A'라고 표기하였다.

환경 A

마스터		슬레이브 1		슬레이브 2	
RAM	4GB	RAM	4GB	RAM	4GB
Hard Disk	150GB	Hard Disk	150GB	Hard Disk	150GB
OS	CentOS 6.4	OS	CentOS 6.4	OS	CentOS 6.4
Hadoop	Hadoop 1.2.1	Hadoop	Hadoop 1.2.1	Hadoop	Hadoop 1.2.1
Spark	Spark 1.3.1	Spark	Spark 1.3.1	Spark	Spark 1.3.1
Mesos	Mesos 0.22.0	Mesos	Mesos 0.22.0	Mesos	Mesos 0.22.0

(그림 1) 가상화된 3대의 노드에서 스파크를 사용한 분산처리 환경
(Figure 1) Distributed processing environment with spark in a virtualized three nodes



(그림 2) 이상치 탐지 및 제거 단계
(Figure 2) Outlier detection and removal stage

3.2 이상치 제거 설계 및 구현

앞서 언급한 바와 같이, 이상치 탐지 및 제거 과정은 (그림 2)와 같이 총 4단계로 이루어진다. 원자료의 값을 구분자를 기준으로 잘라서 <Key, Value>형태로 정리하는 1단계를 시작으로 두 단계를 거쳐 이상치를 탐지한 후,

```
[pseudo code1] SplitMapper(txtFile)

Input: Raw Data.txt
Output: (key: a string for date and time; value: the
value of temperature)

1)load the raw data to RDD
2)split the data by separator
3)date<-data[1]
4)time<-data[2]
5)temperature<-data[3]
6)output(date+time, temperature)
```

(그림 3) 원자료를 분석에 적합한 형태로 변환하는 1단계 의사 코드

(Figure 3) Step 1 pseudo code for converting raw data into a form suitable for analysis

```
[pseudo code2] removeLimitData (key, value)

Input: (key: a string for date and time; value:
temperature)
Output: (key': a string for date and time; value': the
value of temperature)

1) While(value.hasNext()){
2)       get value.next
3)       if(temperature>max || temperature<min)
4)          remove data(key,temperature)
5)       else
6)          value=original temperature
7) }
```

(그림 4) 가시적인 이상치를 제거하는 2단계 의사 코드

(Figure 4) Step 2 pseudo code for removing visible outlier

최종 단계에서 이상치 제거를 함으로써 제거되는 이상치에 대한 정확성을 높이고자 한다.

(그림 3)은 이상치 탐지 및 제거의 1단계로서, 원자료를 BDAS환경에서 사용할 수 있는 알맞은 형태로 변환하기 위해 데이터를 스파크의 작업 단위인 RDD의 형태로 불러온다. 원자료는 구분자로 각 컬럼(Column)이 구분되어 있으므로 구분자를 기준으로 데이터를 나눠서 측정날짜, 시간, 온도를 추출하였다. 그리고 추출한 측정날짜와 시간을 결합하여 Key값으로 정하고, Value값은 각 Key값에 따른 온도 값으로 지정하였다. 1단계의 수행결과는 <Key, Value>형태에 맞추어 <날짜와 시간, 온도>의 형태로 데이터를 변환하여 출력하였다.

1단계에서 변환된 <Key, Value>형태의 데이터를 가지고 2단계를 시작한다. 2단계는 (그림 4)와 같이 가시적 제거 단계로서, 데이터가 가질 수 있는 제한적인

```
[pseudo code3] calculateWMA (key, value)

Input: (key: a string for date and time; value:
temperature)
Output: (key': a string for date and time; value1:
temperature; (value2: estimation, value3: standard
deviation)

1) parallelize the input data
2) split the input data for sliding window
3) make data in window to array
4) wma<- weighted moving average of data(0) to
data(window-1)
5) std<- weighted standard deviation
4) return (key, (temperature, (estimation,std)))
```

(그림 5) 가중이동평균 분석법을 이용하여 데이터의 어림값을 구하는 3단계 의사 코드

(Figure 5) Step 3 pseudo code using a weighted moving average method to obtain approximate value of data

범위의 값 외에 다른 값을 가지는 데이터를 제거하는 단계이다. 데이터가 가질 수 있는 범위의 상한 값과 하한 값을 정하고, 이를 벗어나면 이상치로 간주하여 제거하였다. 본 논문에서는 지구의 역대 최저 기온이 -89.4℃이고, 최고 기온이 57.8℃ 인 것을 고려하여 상한 값은 57.8℃, 하한 값은 89.4℃로 설정하였다.[15] 2단계의 출력은 가시적 이상치 제거 단계를 거쳐서 일부의 데이터가 제거된 <Key, Value>형태의 데이터이다.

3단계는 이상치를 판별하기 위한 단계로 (그림 5)와 같이 진행된다. 3단계에서는 앞서 <Key, Value>형태로 출력된 값을 분산처리를 위한 데이터의 형태로 변환하였다. 변환된 데이터는 슬라이딩 윈도우방식을 통하여 나누었다. 나누어진 각 윈도우마다 가중이동평균 분석법을 이용하여 어림값을 구하고, 유의수준을 설정하기 위하여 표준편차를 구한다. 3단계에서의 최종 출력은 RDD형태인 <Key, (Value1, (Value2, Value3))>로 저장된다. 이때 Key는 측정날짜와 시간, Value1은 측정된 온도, Value2는 어림값, Value3는 표준편차이다.

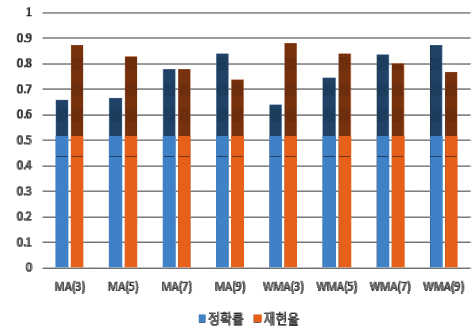
마지막 4단계에서는 (그림 6)과 같이 원자료를 3단계에서 구한 어림값과 비교하여 이상치를 제거하는 단계이다. 어림값을 기준으로 원자료의 데이터 값이 유의수준 이내에 존재하지 않는다면 이를 이상치로 판단하고 제거하였다. 앞서 산출된 <Key, (Value1, (Value2, Value3))>를 가지고 표준편차인 Value3을 이용하여 Value2의 유의수준을 계산한 뒤, Value1이 해당 유의수준 범위 내에 존재하는지 비교하고 범위를 벗어나면 제거하였다. 4단계의 최종 출력은 이상치가 제거된 <Key, Value>형태의 값이다.

```
[pseudo code4] outlierReducer (key, value)

Input: (key: a string for date and time; value1: temperature; value2: estimation, value3: standard deviation))
Output: (key': a string for date and time; value: temperature)

1) If ( value1< valu2's significant level)
2)   return value1
5) else
6)   remove the data
7)  return (date+time, temperature)
```

(그림 6) 이상치를 판별하여 제거하는 4단계 의사 코드
(Figure 6) Step 4 pseudo code for determining and removing outlier



(그림 7) 각 이동평균 방식에 따른 정확률과 재현율
(Figure 7) Precision and recall ratio of each moving average method

4. 실험 및 평가

4.1 실험 데이터

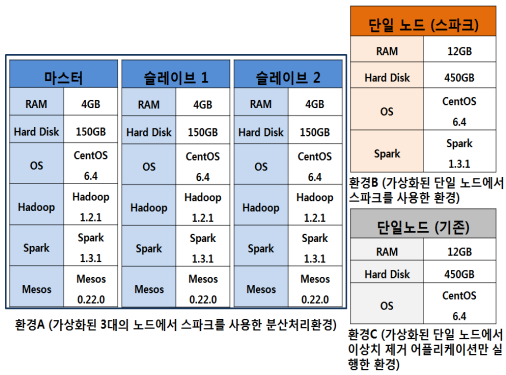
본 논문에서는 우리나라 기상청에서 측정한 데이터[16]를 가지고 실험하였다. 이 데이터는 1분마다 기온을 측정한 것으로 총 1,041,158개의 기온 값으로 이루어져 있으며, 용량은 2GB이다. 본 논문에서는 제안한 이상치 탐지 및 제거 기법의 성능 평가를 위하여 기존 데이터에 189,189개의 이상치를 랜덤하게 추가하여 실험데이터를 만들었다.

4.2 이상치 탐지 결과

가중이동평균 분석법은 슬라이딩 윈도우의 크기에 따라 계산된 결과 값이 다르다. 슬라이딩 윈도우의 크기에 따라 해당 되는 데이터 값과 그 개수가 다르기 때문이다. 따라서 본 논문에서는 좀 더 정확한 결과를 구하기 위해 이동평균 분석법(MA)과 가중이동평균 분석법(WMA)을 이용하여 두 가지 방식을 슬라이딩 윈도우 사이즈를 달리하여 실험하였다.

(그림 7)은 각 슬라이딩 윈도우 사이즈(3, 5, 7, 9) 별로 이동평균 분석법(MA)과 가중이동평균 분석법(WMA) 두 가지 방법을 이용하여 이상치 탐지한 결과에 대한 정확률과 재현율을 그래프로 나타내고 있다. 이때, 정확률은 식 (1)과 같이 구하고, 재현율은 식 (2)와 같이 구한다.

$$\text{정확률} = \frac{\text{실제 이상치 개수}}{\text{이상치로 판별된 데이터의 개수}} \quad (1)$$



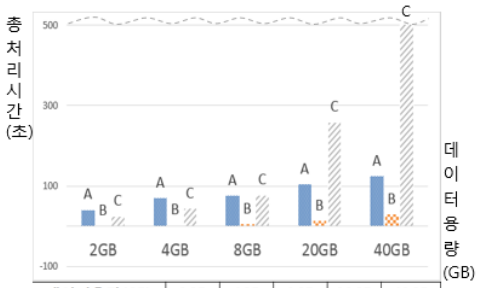
(그림 8) 3가지 실험환경
(Figure 8) Three different experimental environments

$$\text{재현율} = \frac{\text{제거된 실제 이상치 개수}}{\text{실제 이상치 개수}} \quad (2)$$

이상적인 정확률과 재현율의 결과는 모두 1의 값을 갖는 것이며, 정확률과 재현율 모두 높을수록 성능이 좋다고 판단한다.[17] 실험 결과, 슬라이딩 윈도우 사이즈를 7로 설정하여 가중이동평균 분석법을 사용한 WMA(7)이 정확률과 재현율 모두 0.8 이상으로 가장 적합하다고 판단되어 WMA(7)로 설정하여 이후 실험을 진행하였다.

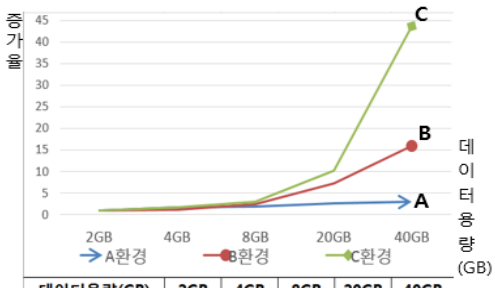
4.3 실행 결과

해당 실험은 (그림 8)과 같이 VMWare로 가상화된 3대의 노드에서 스파크를 사용한 분산처리환경(환경A), VMWare로 가상화된 단일 노드에서 스파크를 사용한 환경(환경B), VMWare로 가상화된 단일 노드에서 다른 프레임워크 없이



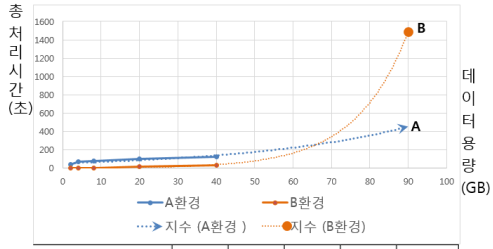
데이터용량(GB)	2GB	4GB	8GB	20GB	40GB
A환경	40.5	69.5	75	104.4	124.74
B환경	1.9	2.16	4.87	13.8	30.3
C환경	25.33	43.9	77.3	259	1104.1

(그림 9) 각 환경에서의 데이터 처리시간
(Figure 9) Data processing time in each environment



데이터용량(GB)	2GB	4GB	8GB	20GB	40GB
A환경	1	1.72	1.85	2.58	3.08
B환경	1	1.14	2.56	7.26	15.95
C환경	1	1.73	3.05	10.23	43.59

(그림 10) 각 환경에서의 처리시간 증가율
(Figure 10) Processing time increasing rate for each environment



데이터용량(GB)	2GB	4GB	8GB	20GB	40GB
A환경	40.5	69.5	75	104.4	124.74
B환경	1.9	2.16	4.87	13.8	30.3

(그림 11) 지수 함수를 이용한 처리시간 추세 예측[18]
(Figure 11) Processing time trend forecasting using exponential function

이상치 제거 어플리케이션만 실행한 환경(환경C) 등 총 3가지 환경에서 실험하였다. 환경A의 각 노드는 메모리를 4GB씩 할당하여 3대의 노드가 총 12GB의 메모리를 갖고, 환경B와 환경C도 동일하게 각각 메모리 12GB을 갖는다.

(그림 9)는 환경A, 환경B, 환경C 3가지 환경에서 각각 2GB, 4GB, 8GB, 20GB, 40GB의 데이터를 처리하는데 걸린 시간을 나타내고 있다. 환경A와 환경B에서 사용된 스파크는 데이터를 메모리에 로드한 후, 데이터에 대한 연산처리가 필요할 때만 데이터에 접근한다. 때문에 스파크를 사용하지 않고 데이터를 처리할 때마다 매번 하드디스크에 반복적으로 접근하는 환경C보다 처리시간이 적게 소요되었다. 환경A는 환경B와 동일하게 스파크를 사용하였지만 처리속도가 환경B보다 느리게 측정되었다. 이러한 결과는 메모리 기반으로 작업을 수행하는 스파크의 특성에 기안한 것으로, 큰 단일 메모리를 갖는 환경B가 전체적으로는 같지만 분산된 메모리들로 구성된 환경A 보다 우수한 성능을 보이는 것으로 판단된다.

(그림 10)은 환경A, B, C 각 처리방식 별로 2GB의 데이터를 처리 하는 데 걸리는 시간을 1로 가정하고, 2GB에 비하여 4GB, 8GB, 20GB, 40GB의 데이터를 처리하는 시간의 증가율을 나타낸 그래프이다. 40GB의 처리시간 증가율의 경우, 환경C에서 처리했을 때는 2GB의 처리시간 대비 43.59배 증가하였으며, 환경B에서 처리했을 때는 15.95배 증가하였다. 반면에 환경A에서 40GB를 처리한 경우에는 2GB를 처리하는 시간대비 3.08배 증가로 가장 작게 증가하였다. (그림 10)의 결과를 통해 데이터의 용량이 커질수록 환경A에 비하여 환경B와 환경C의 처리시간이 급격하게 증가되고 있음을 확인할 수 있었다. 이러한 결과를 토대로 데이터의 용량이 늘어날수록 환경A에서의 처리 방식이 3가지 환경 중 가장 효과적일 것이라 예상할 수 있다.

(그림 11)은 환경A와 환경B에서 처리해야할 데이터의 용량이 40GB보다 더 커졌을 때 각 환경에서의 처리시간을 예측한 그래프이다. 앞서 실험에서 측정한 용량별 데이터의 처리시간을 이용하여 지수 함수방정식을 도출하고, 도출된 방정식에 데이터 용량의 값을 늘리면서 각 방식의 처리시간을 예측하였다.[19] 추세 예측 그래프는 엑셀 2016의 지수 함수를 이용한 추세선 예측 기능을 사용하였다.[18] 그래프에서 보여주고 있듯이 환경A에서 처리한 시간을 나타낸 그래프는 비교적 완만하게 오르고 있는 반면에, 환경B에서 처리한 시간을 나타내고 있는 그래프는 데이터의 용량이 커질수록 급격하게 오르고 있는 것을 볼 수 있다. 이러한 추세 예측결과를 통해 데이터의 양이 커질수록 분산처리환경에서 스파크를 사용한

환경A에서의 처리가 좋은 성능을 보일 수 있을 것이라 예상할 수 있었다.

5. 결 론

IoT 분야에서 데이터를 분석하여 활용하고자 하는 경우에 실제와는 다른 값인 이상치를 포함할 경우에는 왜곡된 결과를 얻게 된다. 따라서 정확한 결과를 도출하기 위해서는 데이터 분석 전처리 단계에서의 이상치 탐지 및 제거가 필수적이다. 또한, IoT 데이터는 점차 그 양이 매우 빠르게 증가하는 추세이기 때문에 이러한 대용량의 데이터를 빠르게 처리하기 위해서는 빅데이터 기술을 기반으로 한 이상치 제거 또한 필수적이다. 이를 위해 본 논문에서는 빅데이터 기술을 이용한 전처리 과정에서의 이상치 탐지 및 제거에 관하여 연구하였다. 맵리듀스 기반으로 이상치를 처리하기 위해 이상치 탐지 및 제거 과정을 총 4단계로 나누고, 각 단계를 매퍼와 리듀서로 구현한 뒤, 설계된 BDAS환경에서 실험을 하였다.

실험에 사용된 스파크는 그 자체의 특성상 메모리 기반으로 작업을 수행하기에 처리시간이 메모리 용량에 영향을 받는다. 때문에 처리해야할 데이터의 크기가 비교적 작을 경우에는 노드 1대의 메모리 용량이 상대적으로 큰 환경B가 적합할 수 있다. 하지만 앞서 실험의 처리시간 증가율에서 보이듯이, 처리해야할 데이터의 크기가 커지면 커질수록 환경B의 처리시간 증가율이 환경A에 비해 급격하게 오르는 것을 확인할 수 있었다. 각 데이터의 처리시간 결과와 처리시간 증가율을 바탕으로 데이터 용량이 더 증가했을 경우를 예측한 결과, 40GB이상의 대용량 데이터에서 이상치 제거는 환경A에서의 처리가 보다 효과적일 것이라 예상 할 수 있었다. 이러한 실험 결과를 통하여 본 논문에서는 대용량 데이터의 이상치 제거를 할 경우에는 환경A, 환경B, 환경C 총 3가지 환경 중 분산처리환경에서 스파크를 사용한 환경A가 가장 적합하다고 판단된다.

본 논문에서 정확한 데이터 분석을 위해 간헐적으로 나타나는 이상치를 탐지하고 제거할 수 있는 방안을 제시하였다. 또한, 대용량 데이터의 이상치를 보다 빠르게 제거할 수 있는 맵리듀스 기반의 분산처리 방안을 제시하였다. 향후에는 데이터를 저장한 후, 저장된 데이터에 한하여 이상치를 탐지하는 것이 아닌 실제 데이터를 스트리밍 방식으로 이상치를 탐지하고 제거하는 연구와 함께 그 효용성을 높이는 연구가 필요하다.

참고문헌(Reference)

[1] Hewlett Packard Enterprise, "Internet of things research study", report, pp. 1-3, 2015.

[2] Zhang, Yang, Nirvana Meratnia, and Paul Havinga. "Outlier detection techniques for wireless sensor networks: A survey." Communications Surveys & Tutorials, IEEE 12.2 pp.159-170, 2010. http://ieeexplore.ieee.org/document/5451757/

[3] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1, pp.107-113, 2008. http://dl.acm.org/citation.cfm?id=J79

[4] Shvachko, Konstantin, et al. "The Hadoop distributed file system." Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, pp.1-10, 2010. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5488875

[5] Zaharia, Matei, et al. "Spark cluster computing with working sets", Hot Cloud 10, pp.10-10, 2010.

[6] David Culler, Michael Franklin (Director), Amplab UC BERKELEY, BDAS, the Berkeley Data Analytics Stack, 2014https://amplab.cs.berkeley.edu/software/

[7] Murphy, Kevin P, "Machine learning: a probabilistic perspective" MIT press, 2012.

[8] Preparata, Franco P., and Michael Shamos, "Computational geometry: an introduction", Springer Science & Business Media, 2012.

[9] Knorr, Edwin M., and Raymond T. Ng. "Finding intensional knowledge of distance-based outliers." VLDB, Vol.99, pp.211-222, 1999.

[10] Zhuang, Yongzhen, et al. "A weighted moving average-based approach for cleaning sensor data." Distributed Computing Systems, ICDCS'07. 27th International Conference on. IEEE, pp.38-38,2007. http://ieeexplore.ieee.org/document/4268192/

[11] Davies, Paul L. "Statistical evaluation of interlaboratory tests." Fresenius' Zeitschrift für analytische Chemie 331.5 , pp.513-519, 1988.

[12] Apache Mesos, The Apache software foundation, http://mesos.apache.org, 2012-2015.

[13] Zaharia, Matei, et al. "Fast and interactive analytics over Hadoop data with Spark." USENIX; login 37.4, pp.45-51, 2012.

[14] Zaharia, Matei, et al. "Resilient distributed data sets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, pp.2.2 2012.

[15] Changyong Park and Youngeun Choi. "Validation of quality control algorithms for temperature data of the republic of korea." Vol.22 No.3, pp.299-307, 2012. https://www.researchgate.net/publication/264105200_Validation_of_Quality_Control_Algorithms_for_Temperature_Data_of_the_Republic_of_Korea

[16] Open Data Portal, "Environment& Weather", https://www.data.go.kr

[17] Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informed, markedness and correlation." Journal of Machine Learning Technologie, 2011

[18] Brown, Angus M. "A step-by-step guide to non-linear regression analysis of experimental data using a Microsoft Excel spreadsheet." Computer methods and programs in biomedicine 65.3, pp.191-200,2001. http://www.sciencedirect.com/science/article/pii/S0169260700001243

[19] Aitchison, John, and Ian Robert Dunsmore, "Statistical prediction analysis", CUP Archive, 1980.

◐ 저 자 소 개 ◑

홍 예 진(Yejin Hong)
2015년 중부대학교 정보통신학과(공학사)
2015년~현재 동국대학교 정보통신학과 석사과정
관심분야 : 빅데이터, 분산처리, 스파크, 클라우드 컴퓨팅
E-mail : sally6157@hanmail.net

나 은 희(Eunhee Na)
2014년 동국대학교 정보통신학과(공학사)
2016년 동국대학교 대학원 정보통신학과 (공학석사)
관심분야 : 빅데이터, 이상치 처리, 분산처리, 클라우드 컴퓨팅
E-mail : d_neh@naver.com

정 용 환(Yongwan Jung)
2002년 숭실대학교 컴퓨터공학부(공학사)
2004년 숭실대학교 대학원 컴퓨터공학(공학석사)
2004년~현재 한국과학기술정보연구원 선임연구원
관심분야 : 클라우드, SDN, 정보보호, etc.
E-mail : paul7931@kisti.re.kr

김 양 우(Yangwoo Kim)
1984년 연세대학교 전자공학과(공학사)
1986년 Syracuse Univ. 컴퓨터공학전공(공학석사)
1992년 Syracuse Univ. 컴퓨터공학전공(공학박사)
1992년~1996년 한국전자통신연구원 선임연구원
1996년~현재 동국대학교 정보통신학과 교수
관심분야 : 클라우드 및 그리드 컴퓨팅, 분산컴퓨팅 시스템
E-mail : ywkim@dongguk.edu