Journal of Digital Contents Society Vol. 16 No. 4 Aug. 2015(pp. 545-553)
http://dx.doi.org/10.9728/dcs.2015.16.4.545

토너먼트 기반의 빅데이터 분석 알고리즘

이현진*

요 약

모든 데이터는 그 자체로 가치를 가지고 있지만, 실세계에서 수집되는 데이터들은 무작위적이며 비구조화되어 있다. 따라서 이러한 데이터를 효율적으로 활용하기 위해서 데이터에서 유용한 정보를 추출하기 위한 데이터 변환과 분석 알고리즘들을 사용하게 된다. 이러한 목적으로 사용되는 것이 데이터 마이닝이다. 오늘날에는 데이터를 분석하기 위한 다양한 데이터 마이닝 기법뿐만 아니라, 대용량 데이터를 효율적으로 처리하기 위한 연산 요건과 빠른 분석 시간을 필요로 하고 있다. 대용량 데이터를 저장하기 위하여 하둡이 많이 사용되며, 이 하둡의 데이터를 분석하기 위하여 맵리듀스 프레임워크를 사용한다. 본 논문에서는 단일 머신에서 동작하는 알고리즘을 맵리듀스 프레임워크로 개발할 때 적용의 효율성을 높이기 위한 토너먼트 기반 적용 방안을 제안하였다. 본 방법은 다양한 알고리즘에 적용할 수 있으며, 널리 사용되는 데이터 마이닝 알고리즘인 k-means, k-근접 이웃 분류에 적용하여 그 유용성을 보였다.

키워드 : 빅 데이터, 데이터 마이닝, k-Means, k-근접 이웃 분류, 맵리듀스

An Algorithms for Tournament-based Big Data Analysis

Hyunjin Lee*

Abstract

While all of the data has a value in itself, most of the data that is collected in the real world is a random and unstructured. In order to extract useful information from the data, it is need to use the data transform and analysis algorithms. Data mining is used for this purpose. Today, there is not only need for a variety of data mining techniques to analyze the data but also need for a computational requirements and rapid analysis time for huge volume of data. The method commonly used to store huge volume of data is to use the hadoop. A method for analyzing data in hadoop is to use the MapReduce framework. In this paper, we developed a tournament-based MapReduce method for high efficiency in developing an algorithm on a single machine to the MapReduce framework. This proposed method can apply many analysis algorithms and we showed the usefulness of proposed tournament based method to apply frequently used data mining algorithms k-means and k-nearest neighbor classification.

Keywords : Big Data, Data Mining, K-Means, K-Nearest Neighbor Classification, MapReduce,

1. 서 론

데이터마이닝(Data Mining)은 데이터에 숨겨진 패턴을 발견하기 위해 다양한 분석 기법들을 적용하여 기존의 분석에서 발견할 수 없던 새로운 정보들을 찾아낼 수 있다. 데이터마이닝 기법에는 분류 (Classification), 군집화 (Clustering), 발견(Discovery), 회귀 (Regression)등이 있으며, 각 기법들마다 다양한 알고리즘들이 존재한다. 초기에는 데이터베이스나 파일과 같은 데이터를 대상으로 분석을 수행했으나, 최근에는 IT 기술 등의 발달로 대용량 데이터를 활용하는 빅데이터 (Big Data) 환경으로 발전함에 따라 데이터 분석 분야에서도 대용량 데이터에 대한 분석을

※ Corresponding Author : Hyunjin Lee
Received: July 21, 2015
Revised : August 16, 2015
Accepted : August 28, 2015
* Dept. of Computer Science & Software, Korea Soongsil Cyber University
Tel: +82-2-708-7863 , Fax: +82-2-708-7749
email: hjlee@mail.kcu.ac

수행하는 경향으로 발전하고 있다[1].

대용량 데이터는 기존 데이터베이스에 저장하는 것이 아니라 분산 환경에 저장되며, 하둡(Hadoop) 프로젝트는 분산 파일 시스템인 HDFS (Hadoop Distributed File System)를 사용하여 데이터를 분산하여 저장한다[2]. 맵리듀스(MapReduce) 프레임워크(Framework)는 구글이 분산 컴퓨팅 환경인 하둡에서 대용량 데이터 처리를 위해 제안한 분산 컴퓨팅 기술이다[3,4]. 하둡과 맵리듀스는 대용량 데이터 저장소, 데이터 분석과 제어를 관리하는 기술로 잘 알려져 있다[5].

데이터마이닝 알고리즘들은 기본적으로 단일 머신에서 동작하는 것을 전제로 만들어졌기 때문에 분산 환경에서 동작하기 위해서는 맵리듀스 프레임워크를 사용하여 개발해야 한다. 일반적인 알고리즘들을 바로 맵리듀스 프레임워크 방법으로 변환하였을 때 정상적으로 동작하지만, 경우에 따라서는 아예 동작하지 않거나, 동작 속도가 매우 느린 현상이 발생된다. 이는 맵리듀스 프레임워크와 하둡이 단일 머신과는 다른 특징을 가진다는 것을 인식하면 해결할 수 있다.

본 논문에서는 단일 머신 알고리즘을 맵리듀스를 사용하여 빅데이터를 분석하도록 변환하였을 때 분석 속도를 향상시킬 수 있는 방법을 제안한다. 토너먼트 방식으로 단계별로 전달되는 데이터의 양을 축소하여 데이터 저장을 위한 디스크 I/O와 데이터 전송을 위한 네트워크 부하를 감소시켜 수행 시간의 효율을 높이고자 한다. 본 방법은 맵리듀스로 수행되는 알고리즘의 수행 시간의 효율성을 높이는 방법으로 다양한 분석 알고리즘에 할 수 있는 방법이다. 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구인 맵리듀스와 하둡에 대해 살펴보고, 3장에서는 제안하는 방법을 적용하여 개발한 토너먼트 기반의 빅데이터 k-means 알고리즘과, 빅데이터 k-근접 이웃 분류 알고리즘에 대하여 살펴본다. 4장에서는 합성 데이터를 사용하여 수행을 한 실험 결과를 분석하고 5장에서 결론을 맺는다.

2. 관련 연구

2.1 하둡

대용량 데이터를 다루는 방법은 빅데이터라는 개념이 나오기 전에도 존재했으며, 그중 성능이 좋은 슈퍼컴퓨터를 이용해서 큰 데이터를 처리하는 방법은 높은 처리 비용과 대규모 저장 장치가 필요하다는 단점을 가지고 있다. 이에 반하여 하둡은 저사양의 컴퓨터들을 하나의 컴퓨터인 것처럼 묶어서 대용량 데이터를 처리하는 기술이다. 하둡은 수천대의 분산된 컴퓨터에 데이터들을 분산하여 저장할 수 있는 기능인 분산 파일 시스템 (HDFS)과 분산된 파일들을 분산된 컴퓨터를 사용하여 빠르게 처리할 수 있는 맵리듀스 프레임워크로 구성되어 있다. 저성능이지만 대량의 서버를 통하여 하둡 클러스터의 컴퓨팅 파워, 저장 용량 등을 쉽게 증대 시킬 수 있다. 또한 HDFS는 데이터 복제의 신뢰성, 빠른 장애 감지 및 자동 복구 등 분산 환경에서의 파일 관리를 위한 다양한 기능들을 제공하고 있다[6].

2.2 맵리듀스

단일 머신 환경에서는 간단한 계산이 분산 데이터 저장 시스템에서는 데이터의 분산 처리를 위한 동기화, 동시성, 로드 밸런싱 등 때문에 매우 복잡해질 수 있다. 맵리듀스 프레임워크 모델은 대용량 데이터에 대한 처리를 수행할 수 있도록 구글에 의해 제안되었다[3]. 이 프레임워크는 데이터의 배포, 작업 스케줄링, 결함 허용, M2M(Machine to Machine)통신 등 다양한 문제를 해결할 수 있다. 맵리듀스로 어떤 작업을 수행하기 위해서는 작업을 맵(map)과 리듀스(reduce) 2가지 함수로 표현하여야 한다.

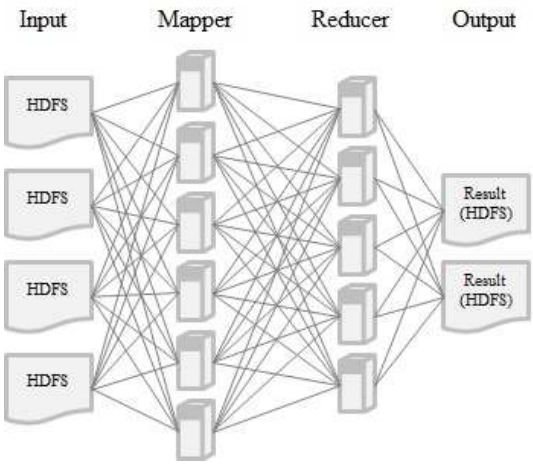
맵 함수는 키(key)와 값(value)의 쌍인 <key, value> 형태로 입력을 받아 키 값의 쌍들을 생성한다. 키는 값의 그룹 번호를 의미하고, 값은 의미 있는 데이터들을 나타내는 것으로 일종의 NoSQL이라고 할 수 있다. 맵리듀스 시스템은 맵 함수가 생성한 모든 키와 값의 쌍들을 모아 리듀스 함수로 전송한다. 리듀스 함수는 매퍼에서 전송된 값들 중 같은 키를 가지는 데이터들을 결합하여 작은 값의 집합을 구하는 "병합(merge)" 프로세스를 거친다. 이 프로세스는 단순 합이 아니라 복잡한 프로세스이다. 리듀서는 같은 키와 연관된 중간 값들을 결합하여 하나의 집합을 구성하는 역할을 하고, 이 모여진 값들에 대한 추가 연산을 수행하기도 한다.

<key, value>는 맵리듀스 프레임워크에서 통신 인터페이스로 사용된다. 맵리듀스 프레임워크는 자동으로 정확하게 같은 키를 가진 값들을 그룹화 한다. 따라서 개발자는 데이터 통신의 세부사항에 대해서 고려하지 않고, 정확하게 의미있는 <key, value>를 설정하는 것만 고려하면 된다.

맵 작업과 리듀스 작업 사이에는 맵 작업의 결과를 미리 결합하여 중간 결과를 생성하는 중간 결합자(Intermediate Combiner)를 둘 수도 있다. 중간 결합자는 한 노드에서 일어난 맵의 결과에 대해서만 계산을 수행하는 것으로, 리듀스의 계산량을 줄여주는 역할을 수행한다. 맵 작업과 리듀스 작업은 분리될 수 없고, 맵리듀스 프로세스에서 함께 사용되어야 한다.

맵리듀스 프로세스를 도식화하면 (그림 1)과 같다. 맵리듀스 프로세스에서 맵 작업과 리듀스 작업 노드별로 분산되어서 노드 내에서 각각 병렬로 실행되고, 맵과 리듀스 작업 자체는 순차적으로 실행된다. 즉, 맵 작업이 종료해야 리듀스 작업이 실행된다. 하나의 맵리듀스 프로세스와 다음 맵리듀스 프로세스가 순차적으로 실행되기 때문에 배치 실행이라고도 한다. 이런 작업들의 동기화는 개발자가 고려하지 않고, 맵리듀스 시스템에 의해 자동으로 수행된다.

(그림 1) 맵리듀스의 처리과정 도식도



(Figure 1) Representation of MapReduce Process

맵리듀스 프레임워크는 단일의 마스터 잡 트래커 (Job Tracker)와 여러 개의 태스크 트래커 (Task Tracker)로 구성된다. 클러스터의 각 노드는 태스크 트래커가 될 수 있다. 잡 트래커는 입력 데이터의 분리, 작업의 일정 관리, 머신의 오류 관리, 실패한 작업의 재할당, 머신간의 통신과 작업의 상태 관리등을 담당한다. 태스크 트래커는 잡 트래커에 의해서 할당된 작업을 실행한다.

맵리듀스는 대용량 데이터를 처리하기에 적합한 형태를 취하고 있다. 따라서, 물리적인 메모리에 적재할 수 없는 대용량 데이터에 대한 마이닝 작업을 수행하는데 적합한 프레임워크라고 할 수 있다.

2.3 맵리듀스를 적용한 데이터마이닝 알고리즘

군집화(Clustering)분야에 맵리듀스를 적용하여 분산병렬처리를 수행하고, 성능을 향상시키는 연구들이 진행되고 있다. Zhou 등은 맵리듀스와 하둡을 사용하여 단일 컴퓨터에서 동작하는 k-Means 알고리즘을 하둡 분산 환경에서 동작할 수 있도록 맵리듀스를 이용하여 대용량 데이터를 군집화하는 방법에 대하여 연구하였다[6]. Gu 등은 디스크 기반의 하둡에서 동작하는 k-Means와 클라우드 환경에서 동작하는 k-Means를 구현하여 클라우드 환경에서 우수한 성능을 보이는 방법을 제안하였다[7]. Lee는 VQ Codebook을 생성하는데 군집화 기법을 사용하여 하둡 환경에서 맵리듀스를 이용하여 구현하였으며, 디컴바인드 분산 알고리즘을 이용하여 우수한 성능을 보이는 방법을 제안하였다[8]. 분류(Classification)분야에서는 Anchalia 등은 하둡 분산 환경에서 동작할 수 있도록 k-NN에 맵리듀스를 이용하는 방법을 제안하여 대용량 데이터에 대해 단일 머신의 k-NN보다 우수한 성능을 보이는 방법을 제안하였다[9].

3. 빅데이터 분석 알고리즘

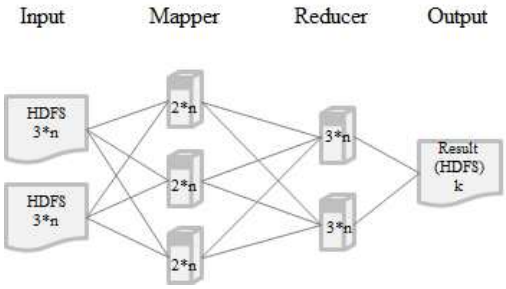
3.1 토너먼트 기반의 빅데이터 분석 알고리즘

하둡에서 동작하는 맵리듀스 프레임워크를 사용하는 빅데이터 분석 알고리즘은 대규모 데이터 전체를 분석할 수 있다는 장점을 가지고 있

다. 하지만, 대부분의 알고리즘은 기본 알고리즘이 단일 머신에서 동작하는 것을 전제로 만들어지기 때문에 단일 머신의 알고리즘을 맵리듀스로 전환할 때 분산 환경의 특징을 고려하지 않으면 분석 속도의 저하된다.

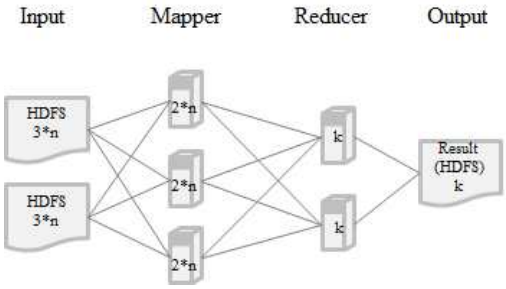
빅데이터 분석 알고리즘을 개발할 때 고려해야 할 사항은 다음과 같다. 첫째, 하둡은 분산 디스크 기반 환경이기 때문에 디스크 읽기(read)가 많이 발생하면, 속도 저하가 발생한다. 둘째, 하둡 노드들 사이에 데이터가 이동할 필요가 있을 때는 네트워크를 사용하기 때문에 매퍼와 리듀서 사이에 이동하는 데이터가 많으면, 속도 저하가 발생한다.

(그림 2) 맵리듀스 단계별 데이터 크기



(Figure 2) Number of Data in each MapReduce Stage

(그림 3) 토너먼트 방식의 맵리듀스 단계별 데이터의 크기



(Figure 3) Number of Data in each Stage for Tournament Method

즉, 하둡이 대용량 데이터를 처리할 수 있는 환경이지만, 속도 향상을 위해서는 처리하는 데이터의 양을 되도록 줄이는 것이 좋다. 이를 위해서는 분석 알고리즘을 변환할 때 토너먼트 방식으로 매퍼에서 대부분의 데이터를 처리하고, 리듀서로 넘기는 데이터의 양은 최소화하는 방식을 취한다.

토너먼트 방식의 특징을 그림으로 나타내면 (그림 2), (그림 3)과 같다. 입력 데이터가 $6*n$이고, 출력이 k개인 알고리즘에서 일반적인 맵리듀스 방법은 맵과 리듀스 단계 모두 데이터의 합은 (그림 2)와 같이 $6*n$개가 된다. 하지만, 토너먼트 방식은 리듀스 단계로 전달할 데이터의 개수를 최소화하는 방식이기 때문에 (그림 3)과 같이 리듀스 단계에서의 데이터는 $2*k$과 같은 출력의 배수가 된다.

3.2 k-means

k-means는 대표적인 군집화 알고리즘이다[10, 11]. 처음에 알고리즘은 k개의 초기 데이터를 임의로 선택한다. 초기 데이터 각각은 군집의 중심을 의미한다. 나머지 데이터들은 각 군집과의 거리에 따라서 가장 가까운 군집에 할당된다. 각 군집에 속한 데이터들을 이용하여 군집의 중심을 다시 계산한다. 이 과정은 에러에 대한 기준 함수를 만족할 때까지 반복된다. <표 1>은 맵리듀스를 적용한 분산 분석 k-means 알고리즘이다[6, 7, 8].

<표 1> 분산 k-means 알고리즘

```
Create current_centroids and new_centroids in
  the filesystem
Write new_centroids with the first k points in
  the input files
repeat
  delete current_centroids
  rename new_centroids to current_centroids
  create empty new_centroids
  for all map tasks do
    read the data points from the input files
    read curren_centroids
    for all data points do
      calcuate the distances between the
        data point and each centroid
      n = identity of the cluster with the
```

```
        closest centroid
    v = coordinates of the data point
    output the <n, v> (assign data point v
      to cluster n)
  end for
 end for
 Run the combine function to sum the
  values of data points assigned to the same
  cluster and output <n, V> for each distinct n
  where V is a composite value of the
  coordinates of the centroid of thd data points
  being combined and the number of data points
  associated with n
 for all reduce tasks do
  mean all the values generated by the
    map task for new cluster centroids
  write the new centroids to new_centroids
 end for
until the difference between the centroids in
  current_centroids and new_centroids is less
  than a threshold or the number of iterations
  reaches the maximum value.
```

<Table 1> Distributed k-means algorithm

제안하는 토너먼트 기반의 빅 데이터 k-means 분석 알고리즘은 <표 2>와 같다. <표 1>의 분산 k-means 알고리즘은 맵 단계에서 키는 군집을 의미하고, 각 군집에 속한 데이터의 좌표를 값에 할당하기 때문에, $< n, v >$를 위해 $n_m * n_{d_m}$의 데이터 공간이 필요하다. 여기서, n_m은 맵의 개수이고, n_{d_m}은 각 맵에 할당된 데이터의 개수이다. 그리고, $n_m * n_{d_m} = n_d$가 된다. 여기서, n_d는 전체 데이터의 개수이다. 하둡은 맵의 출력 결과가 보통 디스크에 저장되기 때문에 데이터의 크기가 커지면, 높은 디스크 I/O로 성능 저하가 발생하게 된다. 또한, 리듀스 단계에서 키에 의해 데이터가 나누어져도 전체 데이터의 크기가 필요하기 때문에 디스크 공간과 메모리 사용량이 증가한다.

제안하는 토너먼트 기반의 k-means 생성 알고리즘은 맵 단계에서 키는 군집을 의미하고, 값은 개별 데이터의 좌표를 각각 할당하는 것이 아니라, <표 1>의 중간 결합자 작업에서 수행하는 데이터들의 평균을 할당한다. 따라서 각 매퍼의 군집에서 저장하는 데이터의 개수는 개별 데이터에 소속 군집을 할당한 개수인 n_{d_m}개에서, 군집 별 평균의 개수인 n_c개로 줄어든다. 하지만 맵 단계에서 평균을 직접 계산하면 리듀스 단계에서 각 군집의 평균을 계산할 수 없다. 따라서 맵 단계에서는 평균 계산식인 $\bar{x} = \frac{\sum x}{n}$에 착안하여, 군집에 속한 데이터 좌표의 합과 군집에 속한 데이터의 수를 계산한 후 리듀스 단계에서 평균을 계산하였다. 이 방법으로 각 매퍼 단계에서 저장하는 데이터의 개수는 $2*n_c$개가 된다. 제안하는 방법은 $< n, v >$와 $< n, n_v >$를 위해 $(n_m * n_c)*2$의 데이터 공간이 필요하다. 여기서, n_c는 군집의 개수이다. 일반적으로 빅 데이터 환경에서 군집의 수는 매퍼의 데이터의 수보다 작기 때문에 $(n^*_{d_m} \gg 2*n_c)$ 메모리와 디스크 사용량이 감소된다. 이는 리듀스 단계에서도 마찬가지로 적용되어 메모리와 디스크 사용량이 감소되어, 전체 수행 시간이 감소된다.

<표 2> 토너먼트 기반 k-means 알고리즘

```
Create current_centroids and new_centroids in
  the filesystem
Write new_centroids with the first k points in
  the input files
repeat
  delete current_centroids
  rename new_centroids to current_centroids
  create empty new_centroids
  for all map tasks do
    read the data points from the input files
    read current_centroids
    for all data points do
      calculate distances between data points
        and each centroid
      n = identity of the cluster with the
          closest centroid
      v = sum of the coordinates of the data
          point
      n_v = number of data points which
            assigned to the cluster
      output the <n, v> (assign data point v
```

```
      to cluster n)
    output the <n, n_v> (assign number of
      data point n_v to cluster n)
  end for
 end for
 for all reduce tasks do
  for each cluster
    sum v and n_v
    calculate new centroids
      new_centroids = n/n_v
  end for
 end for
until the difference between the centroids in
  current_centroids and new_centroids is less
  than a threshold or the number of iterations
  reaches the maximum value.
```

<Table 2> Tournament based k-means algorithm

3.3 k-근접 이웃 분류

k-근접 이웃 분류(k-NN: k-Nearest Neighbor)는 비 파라미터의 lazy 학습 알고리즘이다[8]. 비 파라미터 방식이기 때문에 데이터의 분포에 대한 가정을 하지 않는다. 실 환경에서 데이터의 분포를 가정하는 것은 어려운 일이기 때문에 이는 매우 큰 장점이다. k-근접 이웃 분류는 일반화를 만들기 위해 학습 데이터를 필요로 하지 않기 때문에 lazy 알고리즘이다. 학습 단계는 존재하지 않고, 모든 학습 데이터를 보유하고 있는 상태에서, 새로운 데이터에 적용할 때 모든 학습 데이터를 이용하여 분류를 수행하게 된다. 최근에는 k-NN을 대용량 분석에 사용하는 사례가 증가하고 있다[9, 12, 13].

k-NN을 맵리듀스 프레임워크 형태로 변환하면 다음과 같다[11]. <표 3>과 <표 4>의 알고리즘은 k-NN 알고리즘을 MapReduce로 구현한 것이다. 매퍼에서 필요한 데이터 공간은 $\sum_m n_t * n_{d_m}$으로 n_t는 테스트 데이터의 개수, n_{d_m}은 각 맵에 할당된 학습 데이터의 개수이고, m은 매퍼를 하며, $\sum n_t * n_{d_m} = n_t * n_d$가 된다. 여기서, n_d는 전체 데이터의 개수이다.

<표 3> k-NN을 위한 매퍼 디자인

```
k-NN Mapper
  Create testList in the filesystem to
    maintain data points in the testing
    data-set
  for all map tasks do
    Create trainList in the filesystem to
      maintain data points in the training
      data-set
    for all testList do
      calculate distances between data
        points and each trainList
      n = identity of the test data
      v = coordinates of the distance and
          its class
      output the <n, v>
    end for
  end for
```

<Table 3> Mapper design for k-NN

<표 4> k-NN을 위한 리듀서 디자인

```
k-NN Reducer
  Load the value of 'k'
  for all reduce tasks do
    for each test data n
      Initialize counters for all class labels
        num_class[k] = 0
      for top k distance do
        num_class[k_i]++
      end for
      class = classLabel(max(num_class
                                [k_i]))
    end for
  end for
```

<Table 4> Reducer design for k-NN

<표 3>과 <표 4>는 가장 가까운 k개의 이웃을 전체 학습 데이터에서 찾는다는 기본적인 사상에 충실하게 구현된 알고리즘으로 최종 단계인 리듀서에서 k개의 근접 이웃을 결정한다. 이 방식은 리듀서로 입력 데이터 n을 모두 전달한다. 제안하는 토너먼트 방식의 k-NN은 매퍼에서 k개의 근접 이웃을 선택하고, 리듀서에서 다시 한 번 k개의 근접 이웃을 결정하는 방식으로 리듀서로 n이 아닌 k개의 데이터를 전달하기 때

문에, 저장 공간과 분석 시간을 절약할 수 있다. 이 알고리즘은 토너먼트 방식으로 데이터의 개수를 단계별로 줄여가는 방법으로 제안하는 토너먼트 방식의 k-NN은 <표 5>와 같다.

<표 5> 토너먼트 k-NN을 위한 매퍼 디자인

```
tournament k-NN Mapper
  Create testList in the filesystem to
    maintain data points in the testing
    data-set
  for all map tasks do
    Create trainList in the filesystem to
      maintain data points in the training
      data-set
    for all testList do
      calculate distances between data
        points and each trainList
      sort distances
      for top k distance do
        n = identity of the test data
        v = coordinates of the distance
            and its class
        output the <n, v>
      end for
    end for
  end for
```

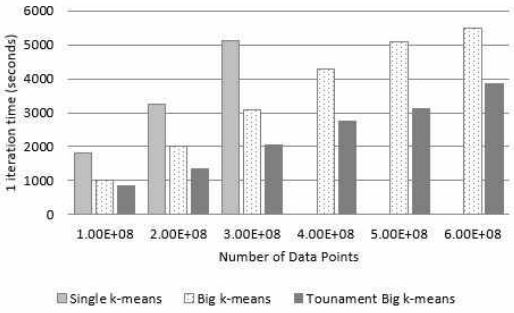
<Table 5> Mapper design for tournament k-NN

리듀서는 <표 4>의 'k-NN Reducer'를 사용할 수 있다. 제안하는 알고리즘은 매퍼 단계에서 k개의 이웃 만을 선택하기 때문에 필요한 데이터 공간은 $\sum_m n_t * k = n_t * n_m * k$ 이다. 여기서, n_m은 매퍼의 개수이다. $n_m * k \ll n_d$이기 때문에 필요한 데이터 공간은 축소되고, 분석 시간도 감소된다.

4. 실험환경 및 결과

본 실험은 한 개의 Namenode와 3개의 Datanode를 가지는 하둡 환경에서 이루어졌다. 실험을 위한 데이터는 빅 데이터에 맞는 합성 데이터를 만들어 사용하였다. 데이터들은 10차원 데이터로 6개의 클래스를 가지며, 각 데이터들은 클래스의 중심을 기준으로 가우시안 분포를 따르게 만들었다. 1억개의 포인트를 가지는 약 700MB의 데이터를 생성 한 후 1억개씩 증가시켜가면서 6억개의 포인트, 약 4.5GB의 데이터까지 6개의 데이터 집합을 생성 하였다.

(그림 4) 단일 k-means, 빅데이터 k-means, 토너먼트 기반 빅데이터 분석 k-means의 1회 수행시간 비교



(Figure 4) One Iteration Time of Single k-means, Big Data Analysis k-means and Tournament based Big Data Analysis k-means

(그림 4)는 k-means에 대한 실험 결과를 비교한 그래프이다. 단일 머신에서 k-means를 수행한 결과(Single k-means)와 Zhou 등이 제안한 맵리듀스를 사용한 빅 데이터 분석 k-means(Big k-means)[6], 제안하는 토너먼트 방식의 빅 데이터 분석 k-means(Tournament Big k-means)의 분석 속도를 비교하였다. 데이터 수마다 iteration 수가 달라질 수 있기 때문에 1회의 iteration에 대한 시간만 비교하였다. 데이터가 작은 경우 (약 10MB 내외)에는 단일 머신에서 동작하는 k-means가 빅 데이터 분석 k-means 보다 더 빠른 속도를 보였다. 하지만, 데이터양이 많아지면, (그림 4)와 같이 빅 데이터 분석 k-means가 더 빠른 속도를 보이는 것을 알 수 있다. 제안하는 토너먼트 방식의 빅 데이터 분석 k-means는 일반적인 빅 데이터 분석 k-means 보다 20% 정도의 속도향상을 보이는 것을 확인할 수 있었다. 단일 머신에서의 k-means는 모든 데이터를 메모리에 올려놓고 수행하는 구조이기 때문에 4억개 데이터(2.1GB) 이상부터는 메모리 부족으로 인하여 실

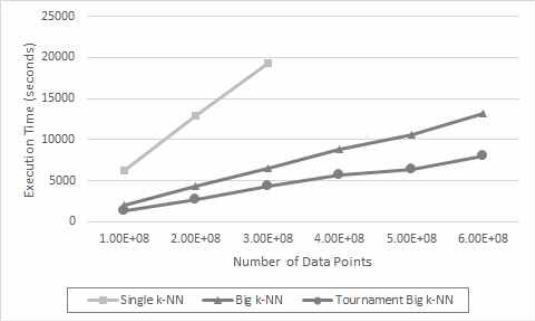
험이 이루어지지 않았다.

k-means 알고리즘은 첫 군집의 중심을 임의로 선정하기 때문에 수행할 때 마다 결과가 달라진다. 비교를 위하여 첫 군집의 중심을 결정한 상태에서 동일한 조건으로 실험을 하였다. 3억개의 데이터를 이용한 실험 결과에서 세 알고리즘 모두 k-means의 반복 횟수는 17회로 동일하였고, 분할된 데이터의 수도 각각 17.7%, 18.3%, 18.1%, 12.4%, 18.2%, 15.3%로 동일한 결과를 보였다. Big k-means와 Tournament Big k-means 모두 Single k-means 알고리즘을 분산 환경에서 동일하게 수행되도록 구성된 알고리즘이기 때문에 동일한 데이터와 동일한 초기 중심값을 사용한 경우에 결과는 동일하였다.

(그림 5)는 k-NN에 대한 실험 결과를 비교한 그래프이다. 단일 머신에서 k-NN을 수행한 결과(Single k-NN)와 Prajesh 등이 제안한 맵리듀스를 사용한 빅 데이터 분석 k-NN(Big k-NN)[11], 제안하는 토너먼트 방식의 빅 데이터 분석 k-NN(Tournament Big k-NN)의 분석 속도를 비교하였다. 한 번에 찾을 이웃의 개수인 k는 5로 하였고, 전체 데이터 중 70%는 학습 데이터, 30%는 테스트 데이터로 사용하였다. 데이터가 작은 경우 (약 10MB 내외)에는 단일 머신에서 동작하는 k-NN이 빅 데이터 분석 k-NN보다 더 빠른 속도를 보였다.

데이터양이 많은 경우 (그림 5)에서와 같이 빅 데이터 분석 k-NN이 더 빠른 속도를 보이는 것을 확인 할 수 있다. 또한, k-means와 마찬가지로 제안하는 토너먼트 방식의 빅 데이터 분석 k-NN이 30% 정도 더 빠른 속도를 보이는 것을 확인 할 수 있다. k-means와 동일하게 5개의 실험 데이터 중 3억개의 데이터에 대한 실험결과를 보면 세 알고리즘 모두 테스트 데이터에 대해 94.6%의 인식률을 보였다. Big k-NN, Tournamet Big k-NN 모두 Sigle k-NN 알고리즘에 기반을 두어 만든 알고리즘이고, 학습 데이터와 테스트 데이터를 미리 구분하고 실험하였기 때문에 동일한 결과를 보인 것을 확인할 수 있었다.

(그림 5) 단일 k-NN, 빅데이터 분석 k-NN, 토너먼트 기반 빅데이터 분석 k-NN의 실행시간 비교



(Figure 5) The Executing Time of Single k-NN, Big Data Analysis k-NN and Tournament based Big Data Analysis k-NN

5. 결 론

본 논문은 맵리듀스를 사용한 분석 알고리즘을 개발하는 데 있어서 토너먼트 방식을 적용하여 매퍼에서 리듀서 사이에 전달되는 데이터의 양을 줄임으로써 성능을 개선하는 목적을 가지고 있다. 성능을 측정하는 여러 척도 중 수행 시간 관점에서 접근을 하였고, 군집화 기법 중 k-means와 분류 기법 중 k-NN에 제안하는 방법을 적용한 알고리즘을 개발하여, 제안하는 방법의 범용성을 보였다. 단일 머신에서의 알고리즘들, 일반적인 빅 데이터 분석 알고리즘들과 제안하는 토너먼트 방식의 빅 데이터 분석 알고리즘들의 성능을 비교한 실험 결과 제안하는 방법에 의하여 k-means와 k-NN 모두 수행효율이 높아진 것을 확인할 수 있었다.

컴퓨팅 파워가 증가하면서 데이터 분석 알고리즘의 수행 시간은 단축되었다. 하지만, 빅 데이터 시대로 오면서 가용 데이터의 양이 극적으로 늘어났기 때문에 빅 데이터에 대한 분석 알고리즘의 수행 시간은 증가하게 되었다. 본 논문에서 제시한 토너먼트 방식의 맵리듀스 기반의 분산 분석 알고리즘은 빅 데이터에 대한 분석 속도를 향상시킬 수 있으며, 이 기법을 다른 알고리즘들에도 확대 적용할 수 있다.

References

[1] Sungmin Kang, Seokjoo Lee, Jun-ki Min, "An Efficient Clustering Method based on Multi Centroid Set using MapReduce," KIISE Transactions on Computing Practices, Vol.21, No.7, pp.494-499, 2015.

[2] Hadoop, "http://hadoop.apache.org/"

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, Vol. 51, No. 1, pp. 107-113, 2008.

[4] Seung-jun Choi, Jea-Won Park, Jong-Bae Kim, Jae-Hyun Choi, "A Quality Evaluation Model for Distributed Processing Systems of Big Data," Journal of Digital Contents Society, Vol. 15, No. 4, pp. 533-545, 2014.

[5] S. Ghemowat, H. Gobioff, and S. T. Leung, "The Google file system," 19th Symposium on Operating Systems Principles, pp. 29-43, 2003.

[6] P. Zhou, J. Lei, and W. Ye, "Large-Scale Data Sets Clustering Based on MapReduce and Hadoop," Journal of Computational Information systems, vol. 7, No. 16, pp. 5956-5963, 2011.

[7] Lin G., Zhonghua S., Zhiqiang M., Xiang G., Charles Z., and Yoohui J., "K-Means of Cloud Computing: MapReduce, DVM, and Windows Azure," in CLOUD COMPUTING 2013, pp. 13-18, 2013.

[8] Hyunjin Lee, "Decombined Distributed Parallel VQ Codebook Generation Based on MapReduce," Journal of Digital Contents Society, Vol. 15, No. 3, pp. 365-371, 2014.

[9] Prajesh P. Anchalia, and Kaushik Roy, "The k-Nearest Neighbor Algorithm Using MapReduce Paradigm," 2014 Fifth International Conference on Intelligent System, Modeling and Simulation, pp. 512-518. 2014.

[10] H. Maulik, and S. Bandyopadhyay. "Genetic Algorithm-Based Clustering Technique," Pattern Recognition, Vol.33, pp. 1455-1465, 2000.

[11] D. Arthur and S. Vassilvitskii. "K-Means++: The Advantage of Careful Seeding," Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[12] Young Joon Kim, Keon Myung Lee, "Big Numeric Data Classification Using Grid-based Bayesian Inference in the MapReduce Framework," International Journal of Fuzzy Logic and Intelligent Systems, Vol. 14, No.4, 2014.

[13] Chi Zhang, Feifei Li, and Jeffrey Jestes, "Efficient parallel kNN joins for large data in MapReduce," Proceedings of the 15th International Conference on Extending Database Technology, pp. 38-49, 2012.

이 현 진

1996년: 순천향대학교 전산학과 공학사
1998년: 연세대학교 대학원 컴퓨터과학과 공학석사
2002년: 연세대학교 대학원 컴퓨터과학과 공학박사
2003년~현재: 숭실사이버대학교 컴퓨터소프트웨어학과 부교수
관심분야 : 이러닝, 기계학습, 빅데이터