ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 24, No. 3, pp. 105-112, 2018. 3
https://doi.org/10.5626/KTCP.2018.24.3.105

분산 인 메모리 DBMS 기반 병렬 K-Means의 In-database 분석 함수로의 설계와 구현

(Design and Implementation of Distributed In-Memory DBMS-based Parallel K-Means as In-database Analytics Function)

구 해 모 † 남 창 민 †† 이 우 현 ††† 이 용 재 †††† 김 형 주 †††††
(Heymo Kou) (Changmin Nam) (Woohyun Lee) (Yongjae Lee) (HyoungJoo Kim)

요 약 데이터의 양이 증가하면서 단일 노드 데이터베이스로는 저장과 처리를 동시에 수행하기에는 부족하다. 따라서, 데이터를 분산시켜 복수 노드로 구성된 분산 데이터베이스에 저장되고 있으며 분석 역시 효율성을 위해 병렬 기능을 제공해야한다. 전통적인 분석 방식은 데이터베이스에서 분석 노드로 데이터를 이동시킨 후 분석을 수행하기 때문에 네트워크의 비용이 발생하며 사용자가 분석을 위해 분석 프레임워크도 다를 수 있어야한다. 본 연구는 군집화 분석 기법인 K-Means 군집화 알고리즘을 관계형 데이터베이스와 칼럼 기반 데이터베이스를 이용한 분산 데이터베이스 환경에서 SQL로 구현하는 In-database 분석 함수로의 설계와 구현 그리고 관계형 데이터베이스에서의 성능 최적화 방법을 제안한다.

키워드: In-database 분석, K-Means 군집화, 분산 데이터베이스

Abstract As data size increase, a single database is not enough to serve current volume of tasks. Since data is partitioned and stored into multiple databases, analysis should also support parallelism in order to increase efficiency. However, traditional analysis requires data to be transferred out of database into nodes where analytic service is performed and user is required to know both database and analytic framework. In this paper, we propose an efficient way to perform K-means clustering algorithm inside the distributed column-based database and relational database. We also suggest an efficient way to optimize K-means algorithm within relational database.

Keywords: in-database analytics, K-means clustering, distributed database

· 이 논문은 2017년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.R0113-15-0005, 대규모 트랜잭션 처리와 실시간 복합 분석을 통합한 일체형 데이터 엔지니어링 기술 개발)

† 비 회 원 : 서울대학교 컴퓨터공학부(Seoul Nat'l Univ.)
hmkou@idb.snu.ac.kr
(Corresponding author임)
†† 비 회 원 : TmaxData F2팀 선임
changmin_nam@tmax.co.kr
††† 비 회 원 : TmaxData F2팀 책임
woohyun_lee@tmax.co.kr
†††† 비 회 원 : TmaxData Data1연구소 상무보
yongjae_lee@tmax.co.kr
††††† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@snu.ac.kr

논문접수 : 2017년 5월 25일
(Received 25 May 2017)
논문수정 : 2017년 10월 2일
(Revised 2 October 2017)
심사완료 : 2017년 12월 17일
(Accepted 17 December 2017)

정보과학회 컴퓨팅의 실제 논문지 제24권 제3호(2018. 3)

1. 서 론

기술과 기계의 발전과 함께 다양한 데이터가 등장하기 시작했다. 그동안 사람이 직접 데이터를 생성하더라도 엄청난 데이터의 양이 수집됐지만, IOT의 등장으로 여기저기서 센서를 통해 수집된 데이터로 인해 데이터의 양은 해가 지날수록 증가하고 있다. 데이터의 종류도 일반적인 데이터를 벗어나 이미지 그리고 동영상의 형태로도 수집돼서 다양한 곳에서 그 현상을 해석하기 위해 분석되고 있다. 이렇게 모인 데이터는 데이터베이스 내에서 안전하게 보관되고 있다. 마찬가지로 분석 과정도 기술의 발전과 더불어 그 방식도 다양해지고 있다. 전통적으로 데이터베이스는 정보의 입력과 저장을 가리키는 Online Transaction Processing(OLTP)를 담당하고 분석은 분석 패키지에서 수행했다. 이에 따라 매번 분석할 때마다 DB에서 질의를 이용하여 데이터를 분석 패키지로 불러와서 분석을 수행하고 있다.

기존에는 단일 노드로 구성된 데이터베이스로 충분했으나, 데이터가 폭발적으로 증가하면서 단일 데이터베이스 보다 다중 노드로 구성된 분산 데이터베이스로의 수요가 증가하고 있다. 그뿐만 아니라, 디스크 기반 관계형 데이터베이스를 탈피해 새 데이터베이스의 한 축이 된 칼럼 기반 인 메모리 데이터베이스도 높은 가용성 및 고성능 덕분에 사용되고 있다.

다중 노드 환경의 사용이 강요되면서 전처리 및 분석을 위해 대용량 처리 프레임워크로 Apache Hadoop이 먼저 등장하고 메인 메모리 대용량 처리 프레임워크로 Apache Spark가 등장했다. 대용량 처리 프레임워크는 다중 노드 환경에서 편리한 분산 처리를 지원하며 대용량 데이터를 사용한 기계학습도 가능해졌다.

이에 데이터베이스 분야에서는 데이터베이스 내에서 분석을 수행하는 In-database analytics에 대한 연구가 시작됐다[1]. 본 연구는 데이터베이스 내에서 분석함수를 구현하고 설계할 때 발생하는 문제점과 그 문제점을 해결하고 성능을 최적화하는 것을 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 인 데이터베이스 분석과 관련 연구를 소개하고, 본 논문에서 다룰 K-Means 알고리즘에 대하여 기술한다. 3장에서는 칼럼 기반 In-memory 데이터베이스 및 관계형 데이터베이스에서 분석 알고리즘의 구현과 성능 향상을 소개하고, 4장에서는 실험 환경과 성능 평가에 관해 설명한다. 마지막으로 5장에서 결론 및 향후 연구에 관해 기술한다.

2. 배경 지식 및 관련 연구

2.1 군집화 분석

군집화 분석은 데이터 마이닝 기법의 하나로 동등한 또는 유사한 성질을 가진 객체들을 같은 집단 또는 모임으로 묶는다. 기계 학습, 패턴 인식, 이미지 분석, 텍스트 마이닝 등 다양한 분야에서 사용되고 있으며 지금도 활발하게 연구되는 분야이기도 한다. 유사성에 따라 그룹으로 나누기 때문에 유사성을 나타내는 기준 및 방법에 따라 크게 중심점 모델, 분포 모델, 밀도 모델 등 다양하게 존재한다.

본 연구에서는 중심점 기반 군집화 기법의 하나인 K-means clustering 알고리즘을 데이터베이스 내에서 구현 및 최적화하는 방법을 소개한다.

2.1.1 K-Means 군집화 알고리즘

비지도 학습 방법 중에서 K-Means 알고리즘은 분석 분야에서 가장 많이 사용되는 알고리즘 중 하나이다[2]. K-Means 알고리즘은 군집화 영역에서도 오래된 역사를 갖고 있지만, 알고리즘이 단순하다는 점이 여러 강점을 가지게 한다. 점들의 집합을 $X = \{p_1, p_2, \cdots p_n\}$로 지정하고 군집 수를 k라고 했을 때 K-Means의 알고리즘은 다음과 같다.

Algorithm 1. K-Means Algorithm

1. Randomly pick k initial means
2. For every point in X, assign the closest mean
3. Calculate new means for new clusters
4. Repeat steps 2 and 3 until means converge

위의 알고리즘에서 1단계를 초기 중심점 생성 단계, 2단계를 모든 점별로 최단 거리에 위치한 입력 변수로 군집 개수만 받으므로 이 알고리즘을 사용하면서 다른 알고리즘에 비해 높은 자유도가 주어지지는 않지만 손쉽게 데이터의 분포를 요약할 필요가 있을 때 그 강점을 나타낸다.

2.1.2 분산 K-Means 알고리즘

K-Means 알고리즘은 알고리즘이 갖는 장점 중 하나로 점별로 최단 거리에 위치한 중심점을 찾는 2단계 과정을 분산 처리하기가 쉽다. 2000년에는 K-Means를 마스터와 슬레이브 형태로 나누는 연구를 이미 진행한 적이 있으며[3], 2009년에는 대용량 데이터 처리 모델인 MapReduce 기반으로 K-Means를 병렬처리하는 연구했다[4]. 인 메모리 대용량 처리 프레임워크인 Apache Spark의 기계학습 라이브러리인 MLLib에서 공식으로 병렬 K-Means Clustering을 제공하고 있다[5].

i번째 노드가 가진 점들의 집합을 $X_i = \{p_{i1}, p_{i2}, \cdots p_{in}\}$으로 지정하고 군집 수를 k 그리고 노드의 수를 c라고 지정한 다음 마스터와 슬레이브 형식으로 수행하는 병렬 K-Means의 알고리즘은 다음과 같다.

Algorithm 2. Parallel K-Means Algorithm

Master Process
1. Randomly pick k initial means
2. Send new means to slaves
3. Wait for all slaves to return
4. Aggregate result and calculate new means for new clusters
5. Repeat steps 2 to 4 until means converge

ith Slave Process
1. Receive k new means from master node
2. For every point in Xi, assign the closest mean
3. Aggregate values with same mean
4. Return master node the result

먼저 각 슬레이브 노드에 데이터를 나눠서 저장한 다음, 마스터 노드가 초기 중심점을 생성한다. 마스터 노드는 각 슬레이브 노드에 초기 중심점을 보내고 각 슬레이브 노드는 점별로 최단 거리 중심점을 탐색한 뒤 새 중심점 정보를 마스터 노드로 보낸다. 마스터 노드는 각 슬레이브 노드로부터 받은 새 중심점 좌표를 받아 수렴했는지를 확인하고 수렴하지 않았다면 새 중심점 정보를 다시 슬레이브 노드로 보내는 것을 반복한다.

2.2 In-database 분석

인 데이터베이스 분석 기술은 데이터베이스 내에서 사용자가 바로 분석 작업을 수행할 수 있도록 하는 기술을 말한다. 전통적으로 데이터의 분석 과정은 데이터를 생성해서 데이터베이스에 저장한 뒤 분석 패키지로 데이터를 옮겨 분석을 수행한다. 하지만, 데이터의 양이 늘어나고 보안이 중요해지면서 데이터베이스에서 데이터를 밖으로 전송하지 않아도 분석을 가능케 하는 In-database 분석 제품이 등장한다. 상용화 제품으로는 Oracle사에서 Oracle Data Mining을, SAP에서는 Analytics Processing 제품을 In-database 분석 제품이 존재한다.

2.3 칼럼 기반 인 메모리 데이터베이스

관계형 데이터베이스는 1970년에 E.F. Codd에 의해 발표된 이후로 점진적인 발전을 이룩했다. 데이터베이스의 핵심인 데이터의 안전한 저장과 빠른 탐색을 발전시키기 위해 다양한 방법론이 제시되었지만, 트랜잭션의 핵심인 ACID 속성을 모두 만족하면서 성능을 높이기에는 한계가 존재했다.

데이터베이스의 연구는 메인 메모리 내에 데이터를 항상 가진 인 메모리 데이터베이스로 발전했고[6], 행렬 기반이었던 관계형 데이터베이스 대신 칼럼을 기준으로 하는 칼럼 기반 데이터베이스도 등장했다.

2.3.1 MonetDB

MonetDB는 2004년에 오픈 소스로 배포되기 시작한 다목적 분석 전용 칼럼 기반 데이터베이스이다. 인 메모리 칼럼 기반으로 작성된 데이터베이스 엔진은 분석 질의인 OLAP과 지리 정보 시스템의 처리나 RDF 등 과학 분석을 목적으로 고성능으로 수행하기 위해 작성됐으며[7], 사용하는 SQL 언어로는 SQL 2003 Standard를 지원하고 있다.

MonetDB는 관계형 테이블을 다룰 때 각 테이블의 열을 Binary Association Table (BAT) 형태로 저장한다. 각 BAT는 본래 데이터를 가리키는 고유 ID와 그 값으로 이루어져 있다. 이로 인해 OLAP 질의 수행을 요청했을 때 데이터베이스는 분석에 필요로 하는 열을 선택해서 사용할 수 있어서 관계형 데이터베이스보다 빠른 질의 수행이 가능하며 메모리에 미리 데이터를 가지고 있기 때문에 불필요한 하드웨어 I/O를 줄일 수 있다는 장점이 있다.

2.4 Apache Spark

아파치 스파크는 인 메모리 분산 처리 프레임워크로 기존의 대용량 분산 처리 프레임워크인 Apache Hadoop이 제공하는 MapReduce가 가진 한계점을 극복하기 위해 등장했다. Hadoop의 MapReduce를 반복적으로 사용할 경우 중간 결과를 매번 Hadoop 분산 파일 시스템(HDFS)에 저장하고 다시 읽어와야 한다. 따라서, 중간 결과를 매번 디스크에서 메모리로 읽어야 하지만, 스파크의 자료 구조인 Resilient Distributed Dataset(RDD)는 메모리가 허용하는 한 메모리에 중간 결과를 저장하고 반복 작업을 수행할 수 있다[8].

3. K-Means Implementation in SQL

이장에서는 데이터베이스 안에서 어떻게 K-Means 알고리즘을 SQL로 수행하는지를 설명한다. SQL은 국제 표준화 기구에 그 기준이 등록되어 있지만, DBMS 사별로 독자적인 SQL 문법을 갖고 있다. 우리는 먼저 DBMS별로 최대한 공통으로 가진 문법으로 작성하고 관계형 데이터베이스에서 성능을 향상할 수 있도록 최적화했다.

단일 노드를 기준으로 SQL 기반 K-Means를 구현할 때 고려해야 할 점과 디스크 기반 관계형 데이터베이스에서 성능을 향상하는 방법을 기술한다. 그리고 성능을 더 극대화하기 위해 분산 데이터베이스 환경에서 어떻게 구현했는지를 차례대로 소개한다.

표 1은 분석을 수행하기 위해 미리 선언한 테이블의 구조를 나타낸다. data 테이블은 분석 대상이 되는 점들을 가지고 있으며 centroids 테이블은 중심점을 그리고 prev_centroids는 중심점이 수렴했는지를 확인하기 위해 사용했다. distance 테이블은 모든 점별로 최단 거리 점을 찾을 때 사용된다.

표 1 K-Means 분석에 사용되는 테이블 스키마
Table 1 Table Schema used in K-Means clustering

data	id, lat, lng, ts, clusterId
centroids	clusterId, lat, lng, ts
prev_centroids	clusterId, lat, lng, ts
distance	id, clusterId, distance

3.1 Single Node Standard K-Means

3.1.1 초기 중심점 생성 단계

Algorithm 1에서 소개한 것과 같이 K-Means의 전체 알고리즘은 크게 3단계로 구성되어 있다. 임의의 초기 중심점 k개를 먼저 생성하는 과정이다. 테이블에서 임의의 행을 추출하기 위해 DBMS에서는 SAMPLE을 제공한다. SAMPLE을 이용해 꺼낸 임의의 점을 하나씩 총 k개가 될 때까지 반복한다.

```
SET centroidsCount = 0;
WHILE (centroidsCount < k) DO
  INSERT INTO centroids
  SELECT lat, lng, ts
  FROM data
  SAMPLE 1;
  SET centroidsCount = (SELECT COUNT(*)
                        FROM centroids);
END WHILE;
```

3.1.2 최단거리 중심점 지정 단계(Assign step)

초기 중심점이 생성되면 모든 점에 대해 최단 거리에 위치한 중심점을 탐색하는 과정으로 넘어간다. 점별로 중심점을 지정하기 위해 모든 점별로 모든 중심점 간의 거리를 계산해야 한다. 아래 SQL은 점별로 최단 거리 중심점을 탐색한다. INSERT 문에 있는 distance()는 UDF로 각 점과 중심점 좌표 간의 거리를 계산해 출력한다. 거리 계산은 시공간 분석을 위해 weighted Euclidean distance (Mahalanobis distance)를 이용하고 있으며 식으로 나타낼 경우 다음과 같다.

$$d(X, C) = \sqrt{\sum_{i=1}^{n} w_i (x_i - c_i)^2}$$

모든 점별로 모든 중심점 간의 최단거리를 계산하기 위해 각 점과 중심점 간의 거리를 변수로 기록할 필요가 있다. 데이터베이스에서 제공하는 PLSQL은 변수를 제공하며 테이블도 변수로 사용할 수 있도록 제공한다. 모든 점에 대해 중심점별로 거리를 계산해 {id, clusterId, distance}인 schema를 갖는 테이블에 저장한다. 그 뒤 점별로 거리를 최소로 갖는 중심점을 지정하면 지정 단계가 종료되고 좌표 갱신 단계를 시작된다.

```
INSERT INTO distance
SELECT id, a.clusterId, distance()
FROM data, centroids AS a;

UPDATE data AS c
SET c.clusterId =
  (SELECT b.clusterId
    FROM (SELECT id, MIN(distance) as d
          FROM distance
          GROUP BY id) a, distance b
    WHERE a.d = b.distance)
WHERE c.id = id;
```

3.1.3 중심점 좌표 계산 단계

최단거리 중심점 지정 단계에서 모든 점에 대해 가장 근접한 중심점의 지정이 완료되면 새 중심점의 좌표를 계산할 차례가 된다. SQL로 군집의 새 좌표를 계산할 때 Aggregation 함수를 사용할 수 있다는 장점이 있다.

```
INSERT INTO centroids
SELECT clusterId, AVG(lat), AVG(lng), AVG(ts)
FROM data
GROUP BY clusterId;
```

먼저 기존 중심점 좌표를 prev_centroids 테이블로 옮기고 GROUP BY와 AVG() 함수를 이용하여 새 중심점의 좌표를 계산한 다음 두 테이블이 일치 한지를 검사함으로써 수렴했는지를 확인하고 종료할지 아니면 Assign 단계를 다시 수행할지를 정한다. 알고리즘이 종료되면 최종적으로 점별로 가장 근접한 중심점이 지정되며, 찾아진 중심점은 지역 최적 해의 조건을 만족한다.

이다음으로는 SQL 기반으로 구현된 표준 K-means 알고리즘의 성능을 데이터베이스의 구조와 기능에 맞춰 향상하게 시키는 방법을 소개한다.

3.2 데이터베이스에서 K-Means 최적화

본 절에서는 데이터베이스에서 분석 함수를 설계하고 구현할 때 발생하는 문제점을 소개하고 그 문제점을 해결하기 위한 전략을 기술한다.

3.2.1 변수 대신 중첩 질의 사용

3.1절에서 구현된 SQL 기반 표준 K-means 알고리즘은 점별로 중심점 간의 거리를 기록하기 위한 변수로 테이블을 사용하고 있다. 테이블을 변수로 사용할 경우 데이터베이스는 정전 및 오류 등으로 인한 비정상적인 종료로부터 복구하기 위해 모든 트랜잭션에 대해 로그를 저장한다. 따라서, 모든 점과 중심점 간의 거리를 계산하는 과정에서 발생한 모든 변화 과정을 redo 로그

형식으로 남기게 된다. 그러나, 중첩 질의를 사용해 중간 계산 결과를 테이블에 저장하지 않고 바로 사용할 경우 redo 로그의 발생을 줄일 수 있다. 아래 SQL은 점별로 최단거리 중심점을 지정하는 assign step에서 중간 계산 결과를 테이블에 기록하지 않고 바로 최단 중심점을 탐색하는 SQL로 관계형 데이터베이스에서는 이를 통해 성능을 향상할 수 있다.

```
UPDATE data as c
SET c.clusterId =
  (SELECT clusterId
   FROM
  (SELECT id, clusterId
   FROM
     (SELECT id, b.clusterId, distance() AS d,
                MIN(distance) OVER
                (PARTITION BY id) AS min_d
      FROM data a, centroids b)
    WHERE d = min_d)
WHERE c.id = id);
```

3.2.2 임의의 초기 중심점 선출 속도 향상

기존에 연구된 SQL 기반 K-means 연구에서는 임의의 k개의 중심점을 선택할 때 SAMPLE syntax를 이용하여 한 개씩 총 k개의 중심점을 뽑았다[9]. 상용화되는 관계형 데이터베이스에서는 표준 SQL은 아니지만, ORDER BY RANDOM을 지원하며 질의 결과는 임의의 순서로 정렬된다. 따라서, 테이블을 임의의 순서로 정렬시켜 k개의 행을 한꺼번에 지정함으로써 초기 중심점 선택 단계를 향상하게 시킬 수 있다.

```
INSERT INTO centroids
SELECT ROWNUM - 1 AS clusterId, lat, lng, ts
FROM (SELECT *
        FROM data
        ORDER BY DBMS_RANDOM.RANDOM)
WHERE ROWNUM < k+1;
```

3.2.3 TRUNCATE의 사용

K-Means 알고리즘을 수행하면서 각 과정별로 중간 결과를 기록해야한다. 하지만, 매 회차 동일한 테이블을 사용하므로, 새 회차를 수행하기 전에 이전 테이블을 삭제할 필요가 있다.

```
DELTE FROM centroids;
DELETE FROM distance;
```

기존에 연구된 데이터베이스 내에서 K-Means를 수행하는 연구에서도 이 과정에서 DELETE를 사용했다[9,10]. 하지만, 테이블을 비우기 위해 중간 결과를 반복해서 삭제해야하는 경우 DELETE 함수의 사용은 성능 저하를 일으킬 수 있다.

데이터베이스의 DELETE 함수는 데이터를 잘못 삭제했을 때 트랜잭션을 되돌리기 위해 rollback을 지원한다. Rollback을 위해 데이터베이스는 redo 로그를 남기게 되며, 이로 인해 I/O가 발생한다. Where 구절이 없는 DELETE FROM Table인 경우에도 테이블을 스캔하고 redo 로그를 남긴 다음 데이터를 삭제한다. 분석 알고리즘의 경우 중간 결과를 일일이 기록할 필요가 없으며 변수 대신 사용한 테이블을 매번 빠르게 초기화할 방법으로 TRUNCATE를 사용할 수 있다. SQL 2008 표준에서 추가된 TRUNCATE Syntax[11]는 테이블을 생성한 초기 상태로 돌리며 rollback이 불가능하다는 특징을 가지고 있으며 덕분에 수행 속도가 DELETE보다 성능이 빠르다.

3.3 분산 데이터베이스를 이용한 병렬 K-Means

그동안 단일 노드 환경에서 데이터베이스를 사용해왔으나, 데이터의 양이 증가하고 재난 및 동시 접속을 처리하기 위해 고가용성이 있어야 하는 등 다양한 이유로 데이터베이스 업계에서도 분산 데이터베이스를 연구하고 있다. 분산 데이터베이스에 데이터가 분산되어 저장된다면 분석 역시 이에 맞춰 분산 처리를 함으로써 성능을 향상할 수 있다.

동시에 여러 노드에 접근하기 위해서는 외부 데이터베이스에 접근할 수 있는 인터페이스가 필요하다. 국산 관계형 데이터베이스인 Tibero는 외부 데이터베이스 노드를 연결할 수 있는 DBLINK를 지원한다. 이 기능을 이용해 다른 노드로 질의를 보내 원격으로 트랜잭션을 수행할 수 있다. 비슷하게 MonetDB는 Messaging Application Programming Interface (MAPI)를 이용하여 원격 MonetDB 인스턴스에 존재하는 테이블에 접근할 수 있다. 덕분에 외부 데이터베이스 테이블에 접속함으로써 분산 데이터베이스를 구축할 수 있으며 단일 질의로 복수의 DBMS에서 트랜잭션을 처리할 수 있다.

그림 1은 전체 분산 데이터베이스 상에서 분산 K-means 알고리즘을 수행할 때의 전체 구조다. N개로 나누어진 각각의 슬레이브 노드와 하나의 마스터 노드에 DBMS를 설치하고 슬레이브 노드별로 미리 나눈 데이터의 테이블을 생성한다. 그 뒤 DBMS별로 제공되는 원격 테이블 접속 인터페이스를 사용해 마스터 노드 DBMS에서 전체 테이블을 View로 생성한다. 여기까지 미리 세팅을 하면 분산 K-Means를 수행할 수 있다. 중심점을 생성하거나 재계산할 때는 마스터 노드에서 계산을 수

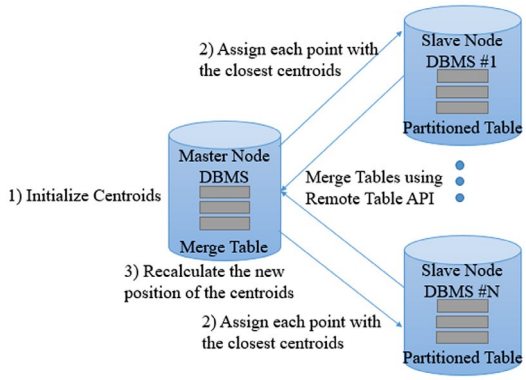

그림 1 병렬 DBMS를 이용한 분산 K-means 수행 구조
Fig. 1 Overall structure for processing distributed K-Means clustering using distributed databases

행하고, 점별로 최단거리 중심점을 탐색할 때는 각 슬레이브 노드에 수행하도록 지시한다.

단일 질의문으로 동시에 다중 노드 상에서 수행할 수는 있지만, 하나의 세션 상에서 각각의 노드에 각자의 질의를 수행시킬 경우 순차적으로 처리하게 된다. 분산 K-means 알고리즘에서 분산 처리해야 하는 과정은 점별로 최단 거리 중심점을 지정하는 단계이다. 이 문제를 해결하기 위해 상위 언어로 작성된 wrapper를 이용해 동시에 여러 노드 상에서 UDF를 수행했다.

wrapper의 구조는 그림 2와 같다. 먼저 마스터 노드에서 세션을 열어 임의의 초기 중심점을 생성하고 assign step을 각 슬레이브 노드별로 세션을 열어 수행을 시킨다. 이 과정이 종료되면 슬레이브 노드에 열었던 세션을 닫고, 마스터 노드에서 재계산 단계를 수행한다.

중심점 테이블은 마스터 노드에 있기 때문에 최단거리 중심점 지정 단계에서 각 슬레이브 노드는 외부 인터페이스를 활용해서 중심점 테이블을 복사해간다. 그다음은 단일 노드일 때와 마찬가지로 각자가 가진 모든 점에 대해 최단 거리에 위치한 중심점을 탐색한다.

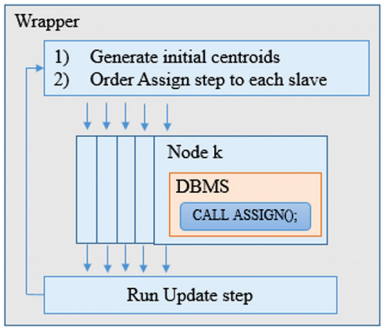

그림 2 분산 K-means 수행을 위한 wrapper의 구조
Fig. 2 Wrapper architecture of distributed K-Means clustering

```
INSERT INTO centroids
SELECT clusterId, AVG(x), AVG(y), AVG(ts)
FROM merged_data
GROUP BY clusterId;
```

모든 슬레이브 노드로부터 최단 거리 중심점 탐색이 종료되면 마스터 노드는 새 중심점의 좌표를 계산해서 수렴했는지를 검사하게 된다. 아래의 SQL이 분산 환경에서의 update 과정이 수행되며 단일 노드와는 FROM 테이블만이 다르다.

통합된 테이블을 사용하기 위해 먼저 원격 테이블을 가져와 view의 등록할 수 있다. 이후 이 view에 대해 질의를 수행하더라도 원격 접근 인터페이스에 맞춰 외부 인스턴스에 질의를 보내 결과를 받아 처리한다. 이 view들에 대해 union 한 테이블을 merged table이라고 하며 각 분산 노드 환경에서 각 테이블을 엮어 전체 테이블을 구축할 수 있다.

각 슬레이브 노드로부터 최단거리 중심점 지정 단계가 종료됐으면 마스터 노드가 통합 테이블을 이용해 중심점의 값을 재계산한 후 수렴했는지를 판단해서 알고리즘을 종료하거나 다시 계산을 수행한다.

4. 실험 및 성능 평가

본 연구에서는 최대 8대의 노드로 클러스터를 구성해 분산 데이터베이스를 구축해서 K-Means 알고리즘의 수행을 비교 분석했다. 실험에 사용한 각 노드는 CPU Intel® Core™ i5-2400 CPU 3.10GHz, RAM 4G, Ubuntu 16.04 32bit의 사양을 가지고 있으며 실험에 사용한 DBMS는 MonetDB Jul 2015-SP4와 Tibero 5 SP1이다. 병렬로 수행을 하기 위해 사용한 Wrapper 역할로 Python은 2.7.12버전, Oracle Java 8, 그리고 비교분석을 위해 대용량 인 메모리 처리 프레임워크로 Apache Spark 2.0.0 버전을 사용했다. 또, Spark로 실험할 때 노드 별로 전체 코어를 사용했다.

이 실험은 데이터베이스 내에서 군집화 분석을 수행했을 때 전통적인 분석 패키지에서 수행했을 때보다 얼마나 성능이 향상되는지를 관찰했다. 먼저 단일 데이터베이스에서 실험 데이터를 넣은 상태에서 User Defined Function으로 구성된 분석 함수를 호출해서 수행해서 함수 호출이 종료되는 시점을 기록하였다. 그 뒤 다중 노드에 각각 데이터베이스를 구축한 다음 기존 데이터를 노드 수에 맞춰 나눠서 넣은 다음 각각의 데이터베이스에서 제공하는 원격 테이블 호출 API인 MonetDB

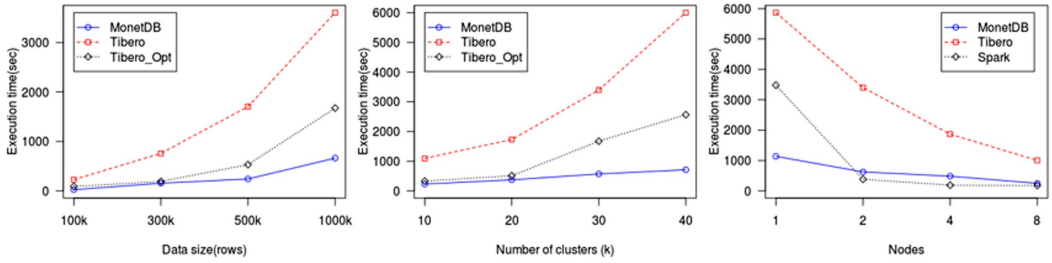

그림 3 (a) 데이터의 크기 증가에 따른 수행 시간 비교(k = 30), (b) 전체 데이터를 100만 개로 고정하고 군집 k의 수 변화에 따른 수행 시간 비교, (c) 전체 데이터 200만 개 군집 k의 수는 30일 때 분산 환경에서 노드의 수 증가에 따른 수행 시간 변화

Fig. 3 (a) Execution time comparison on a single node while increasing data size (k = 30), (b) Execution time comparison on a single node while increasing cluster number (n = 1M), (c) Execution time comparison on distributed environments (k = 30, n = 2M

의 Remote Table과 Tibero의 DBLINK를 사용해 가상의 전체 테이블을 만든 다음 wrapper를 수행해 병렬로 분석 함수를 수행하도록 하였다. 실험을 위해 GDELT Project의 GDELT 2.0 Event Database (2016.06.16 version)를 실험 데이터로 사용했다. GDELT Project는 100개 이상의 국가로부터 생성되는 뉴스를 모니터링하며 언제 어디서 어떤 기사가 게재됐는지를 기록해서 배포하는 데이터로 15분을 주기로 갱신한다[12]. 전체 데이터에는 뉴스의 제목부터 시작해서 뉴스가 어떤 카테고리인지를 포함하는 메타데이터를 전부 포함한 60개가 넘는 칼럼으로 구성되어 있지만, 실험을 위해 고유 ID와 시공간 자료인 위도, 경도, 시간으로 총 4개를 추출해서 사용했다.

4.1 단일 노드 환경에서 비교 분석

먼저 단일 노드 DBMS 상에서 군집화 분석을 수행했을 때 인 메모리 데이터베이스와 관계형 데이터베이스 간의 성능을 측정했다. 데이터는 미리 테이블에 저장한 상태에서 UDF로 K-means 군집화 알고리즘을 호출해 k의 변화에 따라 열 기반 인 메모리 데이터베이스와 디스크 기반 데이터베이스 간의 수행 시간이 얼마나 다른지를 확인했다.

그림 3(a)는 군집의 수를 30으로 고정하고 분류해야 할 데이터의 양을 증가시켰을 때의 수행 시간의 차이를 비교했다. 그림 3(b)는 군집해야 할 데이터를 100만 개로 고정한 상태에서 군집의 수 변화에 따른 수행 시간을 관찰했다. 실험에 사용한 디스크 기반 데이터베이스와 인 메모리 데이터베이스 둘 다 수행 시간의 증가가 선형을 띄우고는 있으나, 인 메모리 데이터베이스보다 디스크 기반 데이터베이스는 오버헤드가 더 높다. 하지만, 디스크 기반 데이터베이스의 최적화 모델인 Tibero_Opt의 경우 인 메모리 데이터베이스에 근접하는 성능을 낼 수 있다는 것을 확인할 수 있다. 이 실험에서 데이터베이스의 튜닝을 일절 없었으며 오로지 SQL만으로 성능을 따라잡을 수 있었다. 그런데도 데이터의 양이 증가할수록 디스크 기반 데이터베이스의 수행 시간 상승 폭은 점차 증가하고 있는 것은 두 데이터베이스의 구조상의 차이가 존재하기 때문이다.

4.2 분산 환경에서 결과 비교 분석

그림 3(c)는 다중 노드로 클러스터를 구성해 노드 수의 증가에 따라 병렬로 수행했을 때 성능이 얼마나 향상되는지를 기록했다. 분산 데이터베이스 환경에서의 K-Means 군집화 수행 성능을 비교하기 위해 같은 환경에서 Apache Spark로 수행한 기록을 비교했다. 200만 개의 점에 대해 군집의 수를 30개로 고정한 상태에서 노드의 수를 변화시켰다. 인 메모리 데이터베이스와 디스크 기반 데이터베이스는 수행 시간이 선형적으로 감소하는 것을 통해 확장성을 증명하고 있다. 인 메모리 데이터베이스는 특정 구간을 제외하고 스파크와 거의 비슷한 성능을 가진다. 스파크의 경우 노드의 메모리보다 데이터의 크기가 큰 경우 성능 저하를 확인할 수 있다.

또 스파크는 노드의 개수가 4개일 때보다 8개일 때 수행 시간이 평균적으로 증가했다. 인 메모리 분산 처리를 지원하는 스파크는 범용적으로 수행하기 위해 마스터가 일을 task 단위로 나눠 슬레이브 노드에 전달하는 방식으로 진행된다. 따라서, 노드의 수가 증가할수록 네트워크로 인해 손실되는 시간이 증가한다.

5. 결 론

본 논문에서는 다중 노드로 구성된 분산 데이터베이스 환경에서 분산 K-means 군집화 방법을 제안했다. SQL과 wrapper를 이용한 방법으로 분산 데이터베이스 환경 속에서 분석 함수를 병렬로 수행함으로써 순수하

게 SQL만으로 수행할 때 보다 성능을 향상하게 시킬 방법을 제안했다. 실험 결과 노드 수가 증가하더라도 성능 향상이 보장되는 것을 확인할 수 있었으며 특정 구간에서 인 메모리 대용량 처리 프레임워크인 스파크보다 더 준수한 성능을 낼 수 있다.

또, 관계형 데이터베이스를 기반으로 중첩 질의를 사용해 [9]에서 제안된 방법보다 성능을 향상하게 시킬 수 있는 것을 확인하였다.

References

[1] J. Taylor. (2013). *In-Database Analytics* [Online]. Available: https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper2/in-database-analytics-106725.pdf (downloaded 2017. May 10)

[2] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and information systems*, Vol. 14, No. 1, pp. 1-37, Jan. 2008.

[3] S. Kantabutra and A. L. Couch, "Parallel K-means clustering algorithm on NOWs," *NECTEC Technical Journal*, Vol. 1, No. 6, pp. 243-247, Jan. 2000.

[4] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," *IEEE International Conference on Cloud Computing*, pp. 674-679, 2009.

[5] Apache Spark. (2016. Dec 28). Clustering - RDD-based API [Online]. Available: https://spark.apache.org/docs/2.1.0/mllib-clustering.html#k-means (downloaded 2017. May 10)

[6] D. J. Dewitt, R. H. Katz, F. Olken, L. D. Shapiro, M. R. Stonebraker, and D. A. Wood, "Implementation techniques for main memory database systems," *ACM*, Vol. 14, No. 2, pp. 1-8, Jun. 1984.

[7] P. A. Boncz, M. L. Kersten, and S. Manegold, "Breaking the memory wall in MonetDB," *Communications of the ACM*, Vol. 51, No. 12, pp. 77-85, Dec 2008.

[8] M. Zaharia, M. Chowdhury, T. Das, A. Dave, and J. Ma, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 2-2, 2012.

[9] C. Ordonez, "Integrating K-means clustering with a relational DBMS using SQL," *IEEE transactions on Knowledge and Data engineering*, Vol. 18, No. 2, pp. 188-201, Dec. 2006.

[10] ISO/IEC. 9075:2008: Information technology - Database languages - SQL

[11] S. Nandagopalan, C. Dhanalakshmi, B. S. Adiga, and N. Deepak, "A fast K-Means algorithm for the segmentation of echocardiographic images using DBMS-SQL," *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, pp. 162-166, 2010.

[12] K. Leetaru and P. A. Schrodt, "Gdelt: Global data on events, location, and tone, 1979-2012," *ISA Annual Convention*, Vol. 2. No. 4. 2013.

구 해 모

2014년 서울대학교 컴퓨터공학부 학사
2014년~현재 서울대학교 컴퓨터공학부 석박사통합과정 재학 중. 관심분야는 데이터베이스, 빅데이터, 데이터 마이닝

남 창 민

2009년 한국과학기술원 수리과학과 학사
2011년 한국과학기술원 수리과학과 석사
2016년 한국과학기술원 수리과학과 박사
2016년~현재 (주) 티맥스데이터 연구소, 관심분야는 데이터베이스

이 우 현

2010년 Univ. of Wisconsin-Madison Computer Engineering 학사. 2015년 서울대학교 컴퓨터공학부 석사. 2015년~현재 (주) 티맥스데이터 연구소, 관심분야는 데이터베이스

이 용 재

1999년 서울대학교 전기공학부 학사. 1999년~2002년 (주) 데이콤 전자상거래팀. 2003년~현재 (주) 티맥스데이터 연구소 관심분야는 인메모리 데이터베이스, 임베디드 데이터베이스, 소프트웨어 품질관리

김 형 주

1982년 서울대학교 전산학과 학사. 1985년 Univ. of Texas at Austin 석사. 1988년 Univ. of Texas at Austin 박사. 1988년~1990년 Georgia Institute of Technology 부교수. 1991년~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 데이터베이스, XML, 시맨틱 웹, 빅데이터