KIPS Tr. Software and Data Eng.
Vol.6, No.10 pp.487~492 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.10.487

Parallel k-Modes Algorithm for Spark Framework

Jaehwa Chung†

ABSTRACT

Clustering is a technique which is used to measure similarities between data in big data analysis and data mining field. Among various clustering methods, k-Modes algorithm is representatively used for categorical data. To increase the performance of iterative-centric tasks such as k-Modes, a distributed and concurrent framework Spark has been received great attention recently because it overcomes the limitation of Hadoop. Spark provides an environment that can process large amount of data in main memory using the concept of abstract objects called RDD. Spark provides Mllib, a dedicated library for machine learning, but Mllib only includes k-means that can process only continuous data, so there is a limitation that categorical data processing is impossible. In this paper, we design RDD for k-Modes algorithm for categorical data clustering in spark environment and implement an algorithm that can operate effectively. Experiments show that the proposed algorithm increases linearly in the spark environment.

Keywords : k-Modes, Categorical Data, Clustering, Spark

스파크 프레임워크를 위한 병렬적 k-Modes 알고리즘

정 재 화†

요 약

클러스터링은 빅데이터 분석 및 데이터 마이닝 분야에서 데이터 간 유사성을 파악하기 위해 사용하는 기법으로 다양한 클러스터링 기법 중 범주적 데이터를 위해 k-Modes 알고리즘이 대표적으로 사용된다. k-Modes와 같이 반복적 연산이 집중된 작업의 속도를 향상시키기 위해 많은 관심을 받고 있는 분산·병행 프레임워크 스파크는 하둡과 달리 RDD라는 추상화 객체 개념을 사용하여 대용량의 데이터를 메모리 상에서 처리 가능한 환경을 제공한다. 스파크는 다양한 기계학습을 위한 라이브러리인 Mllib을 제공하고 있으나 연속적 데이터만 처리 가능한 k-means만 포함되어 있어 범주적 데이터 처리가 불가능한 한계가 있다. 따라서 본 논문에서는 스파크 환경에서 범주적 데이터 클러스터링을 위한 k-Modes 알고리즘을 위한 RDD 설계하고 효과적으로 동작할 수 있는 알고리즘을 구현하였다. 실험을 통해 제안한 알고리즘이 스파크 환경에서 선형적으로 증가한다는 것을 보였다.

키워드 : k-모드, 범주적 데이터, 클러스터링, 스파크

1. 서 론

클러스터링은 ML, 데이터 마이닝 필드에서 데이터 간의 유사성을 기반으로 데이터 객체의 군집화 사용되는 주요 기법 중에 하나이다. 클러스터링 기법은 객체들을 유사한 속성을 갖는 군집으로 분할하여 데이터에 새로운 관점을 부여하기 위한 목적으로 데이터 마이닝 분야에서 주로 사용되기 시작하였다. 최근 빅데이터 분석, 머신 러닝 및 딥러닝 기반의 인공지능 기술에 대한 수요가 폭발적으로 증가하고 따라서 빅데이터 분석을 필요한 정보를 수집 및 분류하고 이를 분석하여 빠른 의사결정에 활용하는 기법의 일환으로 따라 데이터 클러스터링에 대한 수요와 사용 분야가 기하급수적으로 증가하고 있다. 또한 IoT 및 클라우드 컴퓨팅 기술의 대중화에 따라 데이터의 대량화로 클러스터링 비용을 최소화하기 위한 연구가 많은 관심을 받고 있다.

k-Means는 양적인 속성값을 갖는 데이터를 대상으로 데이터 객체 간 거리를 유클리드 거리[1]로 정의하고 클러스터의 평균을 계산하고 비유사도 함수의 값이 최소화되는 방향으로 클러스터를 구성하는 방식이다. 그러나 k-Means는 양적인 속성값 즉 연속적 데이터에만 적용이 가능한 한계가

※ 본 논문은 2016학년도 후기 한국방송통신대학교 학술연구비 지원으로 수행되었음.

† 종신회원 : 한국방송통신대학교 컴퓨터과학과 조교수

Manuscript Received : July 11, 2017
Accepted : July 25, 2017

* Corresponding Author : Jaehwa Chung(jaehwachung@knou.ac.kr)

1) 두 데이터 $a=(a_1, a_2)$, $b=(b_1, b_2)$에 대해 두 점사이의 유클리드 거리는 $\sqrt{(b_1-a_1)+(b_2-a_2)}$ 이다.

있다. 최근 실생활에서의 인터넷 기반의 다양한 서비스의 보편화로 서비스 상에서 사용 및 발생되는 범주적(categorial) 데이터가 폭발적으로 증가하고 있어 k-Means의 과정과 유사하지만 범주적 데이터 처리가 가능한 k-Modes[1] 기법이 제안되었다.

k-Means와 k-Modes 알고리즘 모두에서 데이터 클러스터링은 동일 클러스터 내부의 데이터 객체는 유사한 속성을 갖도록 하며, 상이한 그룹에 속한 데이터와는 다른 속성을 갖도록 분할하는 기법이기 때문에 반복(iteration)적 연산 형태 기반의 k-Modes 알고리즘은 CPU-itensive이다. 따라서 k-Modes 실행 과정에서 발생하는 초대량의 워크로드를 효과적으로 분산하여 다수의 컴퓨팅 자원으로 실행하여 성능 향상이 중요하다.

k-Modes와 같이 발생된 대량의 워크로드를 효과적으로 분산 처리하기 위해 가정 우선적으로 등장한 플랫폼은 구글의 하둡(hadoop)이다. 하둡은 자바 기반의 분산·병행 프레임워크로빅데이터 처리를 위한 표준 플랫폼으로 하둡 분산 파일 시스템(HDFS)과 데이터를 분산시켜 처리한 뒤 하나로 합치는 기술인 맵리듀스(Map/Reduce) 프레임워크를 사용하여 데이터를 처리한다.

그러나 하둡은 처리과정에서 중간에 발생하는 데이터에 대한 읽기와 저장이 메모리와 디스크를 동시에 사용하기 때문에 디스크 I/O가 많이 발생하고 중간 데이터가 다른 노드로 복제되면서 네트워크 I/O 또한 크게 발생하는 문제가 있다. 따라서 분석 대상 데이터의 인메모리 처리를 기본으로 하여 보다 빠르고 지연 속도가 낮은 분석이 가능한 스파크[6]가 새로운 플랫폼이 대안으로 대두되었다.

최근 빅데이터를 처리하기 위한 범용적 분산 고성능 클러스터링 플랫폼 주목 받고 있는 스파크는 하둡의 맵리듀스 작업에서 성능의 병목현상으로 지목되던 디스크 I/O 비용을 최소화하고 데이터 분석 작업에 용이한 인메모리 컴퓨팅 기반의 범용적 데이터 분산처리 시스템이다. 또한, RDD라는 데이터 집합의 추상화 객체 개념을 도입하여 대용량 데이터에 대한 다양한 연산 작업이 가능하도록 한 것이 특징이다. RDD는 실제 물리적인 디스크에 저장된 데이터가 아닌 데이터 집합을 추상화하여 하나의 객체로 표현한다. 또한 여러 노드에 분산되어 다수의 파티션으로 관리되며 하나의 노드에 장애가 발생해도 데이터의 사용 이력 정보를 갖고 있는 lineage를 관리하여 복구 가능한 결함 포용성을 갖고 있는 장점이 있다.

스파크는 다양한 기계학습을 실행할 수 있도록 라이브러리인 Mllib을 제공하고 있다. 그러나 Mllib에는 연속적 데이터를 클러스터링하는 k-Means만 포함되어 있지만, 아직 카테고리컬 데이터를 클러스터링하는 알고리즘이 포함되어 있지 않아 범주적 데이터 처리가 불가능한 한계가 있다. 본 논문은 이러한 한계를 해결하고자 k-Modes 알고리즘을 스파크 환경에서 실행할 수 있는 알고리즘을 제공하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 범주젝 데이터 클러스터링 알고리즘인 k-Modes 알고리즘의 이론적 배경과 알고리즘을 설명한다. 3장에서는 스파크 환경에서 k-Modes 알고리즘을 효율적으로 실행하기 위한 RDD 설계 방법을 설명하고 4장에서는 제안된 스파크 환경에서의 k-Modes 알고리즘을 수행하고 수행 결과를 제공한다. 마지막으로 5장에서 실험 결과를 평가하고 향후 연구주제를 소개한다.

2. k-Modes 알고리즘

이번 장에서는 범주적 데이터 클러스터링 알고리즘은 k-Modes 알고리즘의 이론적 배경, 구체적인 실행 단계와 적용 예를 설명한다.

2.1 이론적 배경

k-Modes 알고리즘은 k-Means 알고리즘의 과정을 유지하면서 대용량의 범주적 데이터를 처리할 수 있도록 제안된 기법이다. 범주적 데이터의 유사성은 k-Means에서 사용하는 유클리드 거리 기반의 유사도 함수를 통하여 계산되지 않기 때문에 범주적 데이터의 유사성을 계산할 수 있는 새로운 함수가 요구된다.

범주적 데이터 집합 $O = \{o_1, o_2, \cdots, o_n\}$ 으로 구성되며 각각의 데이터 객체 o_i는 m개의 범주적 속성에 대한 속성값 $o_i = (o_i^1, o_i^2, \cdots, o_i^m)$ 을 갖는다. 두 데이터 객체 o_i와 o_j의 유사성은 다음과 같은 비유사도 함수 $d(o_i, o_j)$로 정의된다.

$$d(o_i, o_j) = \sum_{k=1}^{m} \delta(o_i^k, o_j^k)$$

이때, o_i^k는 데이터 객체 o_i의 k번째 속성값을 의미하며, $\delta(o_i^k, o_j^k)$는 다음과 같이 정의된다.

$$\delta(o_i^k, o_j^k) = \begin{cases} 0 & o_i^k = o_j^k \text{ 일 때} \\ 1 & o_i^k \neq o_j^k \text{ 일 때} \end{cases}$$

데이터 집합 O에 대응되는 모드(mode) 벡터 v^*는 m개의 속성값 $(v_1^*, v_2^*, \cdots, v_m^*)$ 을 갖는다. v^*는 O에 포함되는 데이터 객체들과 비유사성이 가장 적은 방향으로 정의된다. 즉 모드 v^*는 벡터 $v = (v_1, v_2, \cdots, v_m)$ 중에서 비유사성의 합 $D = (v, O)$이 최소로 하는 벡터로 결정된다. D는 다음의 수식으로 정의된다.

$$D(O, v) = \sum_{i=1}^{n} d(o_i, v)$$

이와 같은 벡터 v^*를 찾기 위해서 속성 a_j가 속성값 $c_{j,k}$를 갖는 빈도수를 $n_{j,k}$라고 가정하면, a_j가 $c_{j,k}$를 가질

상대 빈도값은 다음 함수 f로 계산된다.

$$f(a_j = c_{j,k}|O) = \frac{n_{j,k}}{n}$$

이 때, $k = 1, \cdots, p_j$ 이다. 그러면 모든 $j = 1, \cdots, m$ 에 대하여 v^*는 다음과 같다.

$$f(a_j = q_j^*|O) = f(a_j = c_{j,k}|O)$$

이 조건을 만족하는 $v = (v_1, v_2, \cdots, v_m)$ 는 비유사성의 합 $D = (v, O)$를 최소화 하므로 결과적으로 데이터 집합 O의 모드가 된다. 즉 속성별로 빈도값이 가장 큰 속성값들의 조합이 그 클러스터의 모드가 된다.

2.2 k-Modes 알고리즘

k-Means 알고리즘의 기본 구조를 유지하면서 데이터 간의 유사성을 계산 함수가 유클리드 거리에서 일치하는 속성의 빈도수로 치환된다. k-Modes 알고리즘은 다음과 같이 5단계로 실행된다.

k-Modes 알고리즘은 실행 단계가 간단하기 때문에 범주적 자료를 포함하는 대용량 데이터 적용에 적합하다. 그러나 첫 단계에서 초기 모드를 어떻게 선택하느냐에 따라 k-Means와 마찬가지로 클러스터링 결과 크기가 상이해지기 때문에 초기 mode 설정에 신중할 필요가 있다.

[2, 3]에서는 초기 모드를 선택 하는 방법으로 두 가지를 제안하였다. 첫 번째 방법은 데이터 집합 O에서 랜덤하게 서로 다른 k개의 객체를 선택하여 초기 모드로 지정하는 방법이다. 두 번째 방법은 알고리즘을 실행하기 이전에 각 속성별로 속성값의 빈도를 구해서 최대 빈도를 갖는 속성값을 k개의 초기 모드에 균일하게 배정하고, 각각의 초기 모드와 가장 근사한 객체를 O에서 선택하여 초기 모드로 사용하는 방법이다.

알고리즘 1: k-Modes

Input: Dataset O, Initial mode

1. k개 클러스터의 초기 모드를 정의한다.
2. 각 객체를 비유사성이 가장 적은 군집으로 할당한다. 유사성은 객체 간 일치하지 않는 속성의 개수로 정의한다.
3. k개 클러스터에 대해 속성별로 빈도가 가장 큰 범주 값으로 모드를 갱신한다.
4. 모든 객체에 대해 갱신된 k개 모드와 유사성을 구해서 그 결과값이 가장 큰 객체에 재할당한다.
5. 클러스터의 결과가 변경되지 않을 때까지 3과 4를 반복한다.

2.3 k-Modes 알고리즘 적용 예

이번 절에서는 2.1과 2.2절에서 제시한 k-Modes의 이론적 배경과 알고리즘을 샘플 데이터 집합을 사용하여 실행과정을 보인다.

Fig. 1은 k-Modes 알고리즘의 전체적인 실행 과정을 나타낸다. 주어진 데이터 집합 $O = \{o_1, o_2, \cdots, o_6\}$ 와 k = 2에 대하여 초기 최빈값을 구하기 위해 클러스터 1과 모드 2에 대해 랜덤하게 선택된 두 객체 o_1과 o_5를 할당한다고 가정한다. 랜덤으로 선택된 두 객체는 초기 클러스터의 최빈값으로 사용된다. 모든 객체는 각 클러스터의 최빈값과의 거리 연산을 통해 가장 가까운 거리의 클러스터로 할당된다. 예를 들면, 첫 번째 객체 o_1는 클러스터 1과 클러스터 2의 최빈값과 거리를 계산하며, 가장 가까운 클러스터 1번에 포함된다. 각 클러스터의 최빈값은 각 속성별로 현재까지 해당 속성에 몇 개의 데이터가 얼마나 포함되었는지를 별도의 테이블로 저장하고 있다. 클러스터 1에 객체 [A, B]가 포함되면서 첫 번째 속성에는 A라는 값이, 두 번째 속성에는 B라는 값이 추가되면서 각 속성별 테이블에서의 값도 1씩 증가한다. 알고리즘의 단계 4에 의해서 초기 최빈값이 구축된 이후에 유사성이 큰 객체를 클러스터에 할당하게 된다. 따라서 첫 번

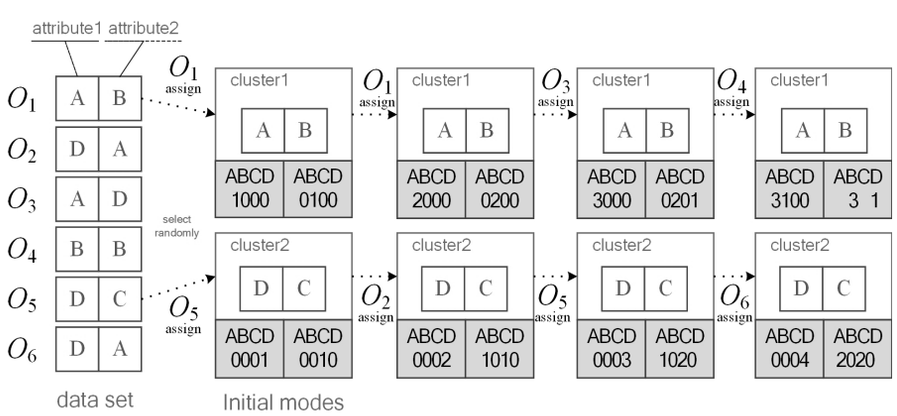

Fig. 1. Processing Steps of k-Modes Algorithm

째로 o_1과 o_6이 각각 클러스터 1과 클러스터 2에 할당된다. 이 과정은 남아있는 데이터 객체가 소진될 때까지 반복되며 이 후에 o_3과 o_5가, o_4과 o_6이 할당되며 그에 따라 최빈값이 업데이트 된다. 이 과정은 단계 5에서와 같이 최빈값이 전 반복에서와 같이 변경이 없을 때 까지 연속된다.

3. 스파크 기반 *k*-Modes 알고리즘

본 논문에서 제안하는 병렬 *k*-Modes 알고리즘은 스파크 환경에서 동작하기 때문에 처리하고자 하는 데이터를 RDD로 변환한다. RDD는 몇 개의 파티션으로 나누어지며, RDD 연산은 파티션 단위로 실행된다. Fig. 2는 전체 데이터 셋이 2개의 파티션으로 구성된 RDD로 변환된 예제이다. 각 파티션 마다 *k*-Modes 알고리즘이 실행된다.

3.1 초기 최빈값 선택

k-Modes 알고리즘은 초기 최빈값(mode)을 설정하는 것부터 시작된다. *k*-Modes 알고리즘은 클러스터링 결과의 정확도를 높이기 위한 다양한 초기 최빈값 설정 방법이 있다[2-5]. 우리는 스파크 환경에서 *k*-Modes 알고리즘을 구현하는데 초점을 맞추었기 때문에 대표적인 초기값 설정 방법인 랜덤 선택 방법을 사용한다. *k*개의 클러스터를 구성하는 것을 가정하였을 경우, 전체 데이터 셋에서 *k*개의 객체를 랜덤을 선택하여 각 클러스터의 최빈값으로 사용한다. Fig 2에서는 두 개의 클러스터(*k* = 2)로 구분하기 위해 전체 데이터 셋에서 2개의 데이터를 임의로 선택하여 각 클러스터의 최빈값으로 설정한다. 1번 클러스터의 최빈값은 [A, B]이고, 2번 클러스터의 최빈값은 [D, C]이다.

3.2 클러스터 할당

초기 최빈값이 설정되면 파티션별로 객체는 각각의 클러스터의 최빈값과 거리가 가장 가까운 클러스터에 할당된다. 클러스터의 최빈값과 객체와의 거리 연산은 식을 이용한다. 예를 들어 Fig. 2에서 파티션 1의 [A, B] 객체는 각 클러스터의 최빈값의 거리 비교를 통해 1번 클러스터에 할당되고, [D, A]는 2번 클러스터에 할당된다. 객체가 할당될 클러스터에 대한 정보는 해당 객체에 추가한다. 예를 들어, [A, B] 객체가 1번 클러스터에 할당된다는 정보는 [A, B] 객체의 속성을 확장해서 추가해준다. 객체가 추가되어야 할 클러스터 정보는 객체가 그룹별로 클러스터될 때 키값으로 사용된다. 여러 개의 속성으로 구성된 객체는 클러스터에 할당되면서 클러스터에 포함된 객체가 속성별로 가지고 있는 데이터의 빈도수를 기록해야 한다. RDD 기반에서는 Flatmap 연산을 통해 하나의 여러 개의 속성으로 객체를 각 속성으로 분할하여 새로운 객체로 구성된 RDD를 만든다. 새로운 객체는 Fig. 3과 같이 객체가 포함될 클러스터 번호, 속성 인덱스, 데이터, 카운트를 위한 '1'로 구성된다. 하나의 [A, B] 객체는 Flatmap 연산을 통해 A와 B로 분할되면서(클러스터 번호, 속성 인덱스, 데이터)를 키값, 카운트를 위한 1을 데이터로 하는 두 개의 키-값쌍이 생성된다.

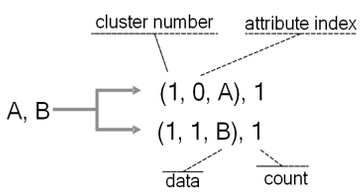

Fig. 3. Splitting Object Attributes for Cluster Operations

3.3 클러스터의 속성별 데이터 카운트

키-값쌍은 각 클러스터의 속성별로 데이터의 빈도수를 카운트하는데 사용된다. (클러스터번호, 속성 인덱스, 데이터)을 키값으로 설정한 다음, 동일한 키값을 갖는 객체에 카

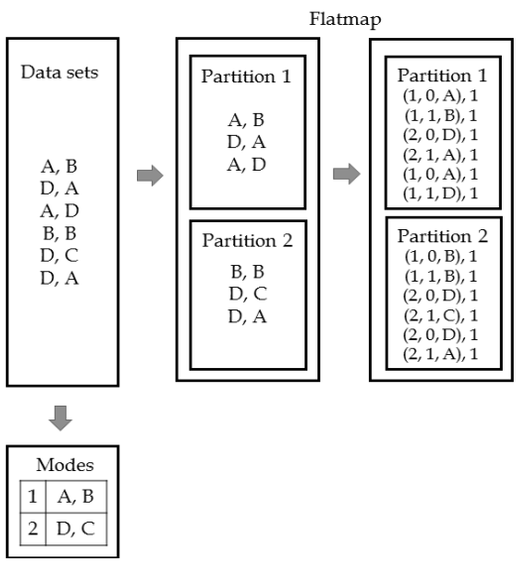

Fig. 2. Initial Mode Selection and Data Partitioning

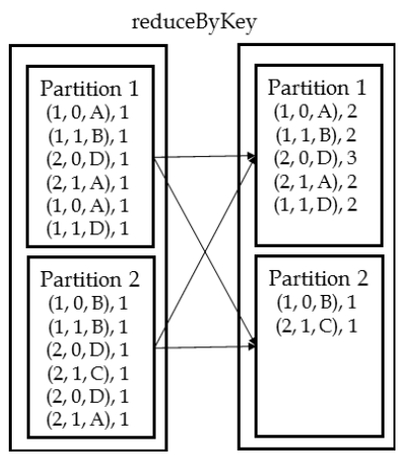

Fig. 4. Integration to Count Data of Each Attribute in Each Cluster Using reduceByKey

운트 값을 더하기 위해 reduceByKey 연산을 사용한다.

Fig. 4에서 키값 (2, 0, D)는 2번 클러스터에 포함된 객체 중에서 인덱스 0의 값이 D인 객체를 의미한다. 동일한 키값을 갖는 객체는 파티션 1에 1개, 파티션 2에 2개가 있다. reduceByKey을 이용하여 카운트 값을 더하여 새로운 RDD를 만들면서 (2, 0, D)를 키값으로 갖고 카운트 값이 3인 객체를 만든다.

3.4 클러스터 최빈값 계산

각 클러스터의 속성별로 데이터의 빈도수 카운트가 완료되면, 최빈값 계산을 통해 클러스터의 센터를 재설정한다. RDD에서 클러스터의 최빈값을 계산하기 위해서는 키-값쌍의 구조를 변경한다. Fig. 4에서 만들어진 RDD는 (클러스터 번호, 속성 인덱스, 데이터)를 키, 카운트 수를 값으로 하는 키-값쌍 튜플로 구성되어 있다. map 연산을 통해 (클러스터 번호, 속성 인덱스)를 키, (데이터, 카운트수)를 값으로 하는 튜플로 변경한다.

(클러스터 번호, 속성 인덱스)로 구성된 키 값을 기반으로 groupByKey를 통해 동일한 키 값을 가지고 있는 객체를 하나의 튜플로 그룹화시킨다. Fig. 5를 보면, 파티션 1에 있는 '(1, 0), (A, 2)'는 파티션 2에 있는 '(1, 0), (B, 1)'은 동일한 키 값을 가지고 있다. groupByKey를 통해 두 개의 객체는 하나의 객체인 (1, 0), [(A, 2), (B, 1)]로 통합된다. Fig. 6을 보면 1번 클러스터의 0 인덱스 속성에는 A가 2개, B가 1개가 할당되었고, 카운트가 2인 A가 최빈값으로 선택된다.

Fig. 6에서 볼 수 있듯이 연산이 완료되면 RDD에는 (클러스터 번호, 속성 인덱스)를 키로 하고, 해당 범주적 데이터를 값으로 하는 키-값쌍 객체가 남는다. 최종적으로 남는 값이 한 번의 반복연산으로 새롭게 계산된 클러스터별 최빈값이다. 기존의 클러스터별 최빈값과 비교하여 차이가 발생하면 3.2 클러스터 할당에서부터 다시 반복한다. 만약 기존의 클러스터별 최빈값과 새롭게 만들어진 클러스터별 최빈값이 변동이 없는 경우, 클러스터링을 종료한다.

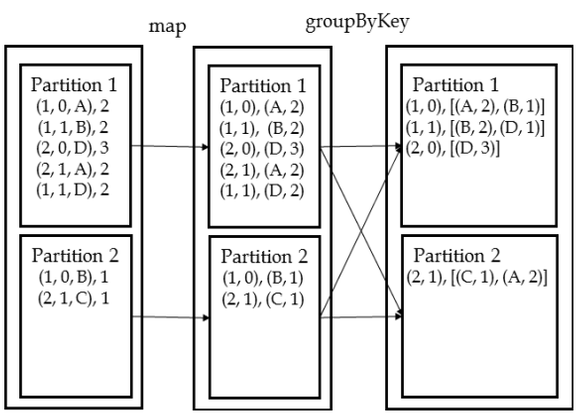

Fig. 5. Preparing for Selection of Mode in Each Cluster

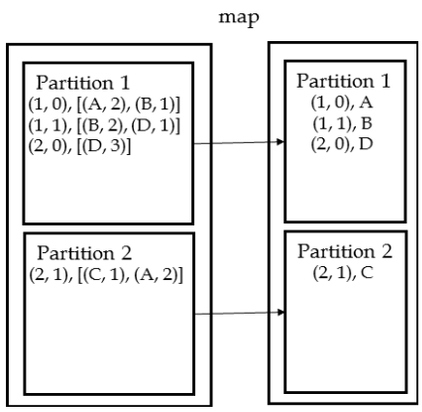

Fig. 6. Updateing Mode

4. 실험 및 평가

이번 장에서는 제안한 스파크 기반의 *k*-Modes 알고리즘을 구현하고 실행한 결과를 제시하여 제안된 알고리즘의 성능을 평가한다.

본 논문에서 제안한 스파크 기반의 *k*-Modes 알고리즘의 실행속도를 평가하기 위해 기존의 단일 머신 환경에서 실행되는 *k*-Modes 알고리즘의 실행속도와 비교하였다. 단일 머신 환경에서는 오픈소스 통계 분석 도구인 R에서 제공하는 *k*-modes package를 이용하여 실험을 수행하였다. 비교 실험을 위해 객체의 개수가 1,000개, 10,000개, 100,000개로 구성된 데이터 셋을 생성하였다. 각 데이터 셋의 객체는 5개의 범주적 속성으로 구성된다. 각 속성은 알파벳 A부터 Z까지 임의로 선택하여 구성된다.

Fig. 7을 보면 데이터의 개수가 작을 때는 단일머신 환경이나 스파크 환경이나 실행 속도에 큰 차이가 나지 않는 것을 알 수 있다. 데이터의 개수가 작을 때는 분산 처리를 위해 추가적으로 필요한 작업인 데이터 분할 및 병합 과정에 시간이 소모되어 오히려 스파크 환경의 실행 속도가 늦는 경우가 있다. 데이터 개수가 증가하면서 R은 수행속도가 급

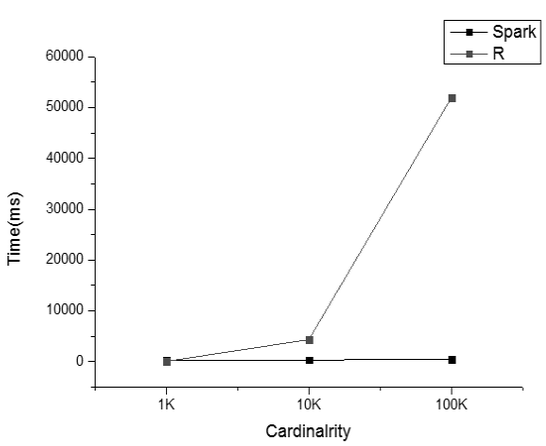

Fig. 7. Experiment Results

격히 증가한다. 하지만 스파크 환경에서는 데이터 개수가 증가해도 수행 속도의 증가량을 크지 않은 것을 알 수 있다.

5. 결 론

본 논문에서는 스파크 환경에서 범주적 데이터 클러스터링을 위한 k-Modes 알고리즘을 위한 RDD 설계하고 효과적으로 동작할 수 있는 알고리즘을 구현하였다. 스파크에는 기계학습을 지원하기 위해 Mllib이라는 라이브러리를 제공하고 있지만, 아직까지 수치형 데이터만을 처리할 수 있는 라이브러리만 제공된다. 따라서 현실 세계에서 발생하는 범주적 데이터를 처리하는 아직 어려움이 있다. 우리는 현실 세계에서 발생하는 범주적 데이터를 스파크 환경에서도 처리할 수 있도록 대표적인 범주적 클러스터링 알고리즘인 k-modes를 구현하였다. 실험을 통해 통해 제안한 알고리즘이 스파크 환경에서 선형적으로 증가한다는 것을 보였다. 향후 연구로는 연속적과 범주적 데이터가 혼합된 k-prototypes 알고리즘을 스파크 환경에서 구현하고자 한다.

References

[1] Z. Huang, “A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining,” In *Research Issues on Data Mining and Knowledge Discovery*, pp.281-297, 1997.

[2] Y. Sun, Q. Zhu, and Z. Chen, “An Iterative initial points refinement algorithm for categorical data clustering,” *Pattern Recognition Letters*, Vol.23, pp.875-884, 2002.

[3] P. S. Bradley and U. M. Fayyad, “Refining Initial Points for K-Means Clustering,” *Proceedings of the 15th International Conference on Machine Learning (ICML98)*, San Francisco, Morgan Kaufmann, 1998.

[4] S. S. Khan, “A. Ahmad, Cluster center initialization algorithm for Kmeans clustering,” *Pattern Recognition Letters*, Vol.25, No.11, pp.1293-1302, 2004.

[5] S. S. Khan and S. Kant, “Computation of Initial Modes for K-modes Clustering Algorithm using Evidence Accumulation,” *IJCAI-07*, pp.2784-2789, 2007.

[6] Z. Huang, “Extensions to the k-means algorithm for clustering large data sets with categorical values,” *Data Mining Knowl. Discov.*, Vol.2, No.2, pp.283-304, 1998.

정 재 화

e-mail : jaehwachung@knou.ac.kr

1999년 고려대학교 컴퓨터교육과(이학사)

2011년 고려대학교 컴퓨터교육과 (이학석・박사)

2012년～현 재 한국방송통신대학교 컴퓨터과학과 조교수

관심분야 : 공간질의처리 및 인덱싱, 분산 컴퓨팅플랫폼 (mapreduce, spark), 모바일 데이터 관리, RFID, 무선 센서 네트워크, 컴퓨터 교육, 컴퓨터적 사고력