Journal of Digital Contents Society Vol. 15 No. 3 Jun. 2014(pp. 365-371)
http://dx.doi.org/10.9728/dcs.2014.15.3.365

맵리듀스를 사용한 디컴바인드 분산 VQ 코드북 생성 방법

이현진*

요 약

빅 데이터(Big Data)시대로 접어들면서 기존의 IT 환경에서 만들어진 알고리즘들은 하둡과 같은 분산 아키텍처에 그대로 적용할 수 없거나 효율이 떨어진다. 따라서, 맵리듀스와 같은 분산 프레임워크를 적용한 새로운 알고리즘들이 필요하다. 벡터 양자화에 많이 사용되는 Lloyd의 알고리즘도 맵리듀스를 사용하여 개발이 이루어지고 있다. 본 논문에서는 기존의 맵리듀스를 사용한 분산 VQ 코드북 생성 알고리즘을 수정하여 좀 더 빠른 분석 결과를 보일 수 있는 디컴바인드 분산 VQ 코드북 생성 알고리즘을 제안하였다. 제안하는 알고리즘을 빅 데이터에 적용한 결과 기존 방법보다 높은 성능을 보인 것을 확인할 수 있었다.

키워드 : 빅데이터, 맵리듀스, VQ 코드북 생성, 군집화, 분산 컴퓨팅

Decombined Distributed Parallel VQ Codebook Generation Based on MapReduce

Hyunjin Lee*

Abstract

In the era of big data, algorithms for the existing IT environment cannot accept on a distributed architecture such as hadoop. Thus, new distributed algorithms which apply a distributed framework such as MapReduce are needed. Lloyd's algorithm commonly used for vector quantization is developed using MapReduce recently. In this paper, we proposed a decombined distributed VQ codebook generation algorithm based on a distributed VQ codebook generation algorithm using MapReduce to get a result more fast. The result of applying the proposed algorithm to big data showed higher performance than the conventional method.

Keywords : Big Data, MapReduce, VQ Codebook Generation, Clustering, Distributed Computing

1. 서론

최근 IT 기술과 컴퓨터, 인터넷의 발달로 사회는 대용량 데이터를 활용하는 빅데이터 (Big Data) 환경으로 발전하고 있다. 데이터의 종류와 복잡도가 증가함에 따라 빅데이터를 단순히 수집, 변환, 저장하는 수준에서 벗어나서 특정한 목적을 위해 빅데이터를 분석할 필요성이 높아지고 있다.

벡터 양자화 (Vector Quantization, VQ)는 데이터 분석에서 가장 중요한 방법 중 하나이다 [1]. 벡터 양자화는 적용하기 쉽고, 훈련 과정이나 선 분류등의 작업 등 사용자의 개입이 최소화되기 때문에 다양한 분야에서 사용되고 있다 [2, 3].

맵리듀스(MapReduce)는 구글(Google)이 분산 컴퓨팅 환경에서 대용량 데이터 처리를 위해 제안한 분산 컴퓨팅 기술이다[4]. 하둡(Hadoop) 프로젝트는 분산 파일 시스템인 HDFS (Hadoop

※ 교신저자(Corresponding Author): Hyunjin Lee
접수일:2014년 4월 28일, 수정일:2014년 6월 01일
완료일:2014년 6월 16일
* 숭실사이버대학교 컴퓨터정보통신학과
Tel: +82-2-708-7863 , Fax: +82-2-708-7749
email: hjlee@mail.kcu.ac

Distributed File System)를 제공한다[5, 6]. 하둡은 자바(Java)기반의 소프트웨어 프레임워크로 분산 환경에서 데이터 집중적인 어플리케이션을 개발하는데 사용된다. 하둡과 맵리듀스는 대용량 데이터 저장소, 데이터 분석과 제어를 관리하는 기술로 잘 알려져 있다[7]. 현재 빅데이터에 저장된 데이터를 분석할 수 있는 알고리즘들에 대한 연구가 다양하게 진행되고 있다[8].

벡터 양자화를 수행할 때 중요한 것은 왜곡을 최소화하는 분류 집합을 찾는 것이고, 이 분류 집합을 중심 집합 또는 VQ 코드북이라고 한다. VQ 코드북을 계산하는 방법은 Lloyd의 알고리즘(Lloyd's Algorithm), 비균질 이진 분할, LBG 알고리즘 (Linde, Buzo, Gray) 등이 있는데 가장 기본이 되는 알고리즘은 K-Means라고도 불리우는 Lloyd의 알고리즘이다[9, 10].

본 논문에서는 맵리듀스를 사용하여 빅데이터에 대한 빠른 VQ 코드북을 생성하는 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구인 맵리듀스와 하둡에 대해 살펴보고, 3장에서는 제안하는 맵리듀스를 적용한 VQ 코드북 생성 방법을 살펴본다. 4장에서는 합성 데이터를 사용하여 수행을 한 실험 결과를 분석하고 5장에서 결론을 맺는다.

2. 관련 연구

2.1 하둡(Hadoop)

데이터 집합의 크기가 하나의 저장 공간의 용량을 초과하면, 해당 데이터를 여러 개의 독립적인 컴퓨터에 분산하는 것이 필요하다. 네트워크에 의해 컴퓨터 간의 파일을 관리하는 시스템을 분산 파일 시스템이라고 한다. 일반적인 하둡 분산 파일 시스템은 수천 개의 서버를 포함하고 있고, 각 서버에는 파일 시스템의 일부 데이터가 저장되어 있다. 많은 서버와 간단한 설정을 통하여 하둡 클러스터의 컴퓨팅 파워, 저장 용량 및 IO 대역폭을 쉽고 저렴하게 개선할 수 있다. 또한, HDFS는 데이터 복제의 신뢰성, 빠른 장애 감지 및 자동 복구 등 다양한 기능들을 제공하고 있다[11].

2.2 맵리듀스(MapReduce)

분산 데이터 저장 시스템에서는 데이터를 병렬로 처리할 때 동기화, 동시성, 로드 밸런싱(load balancing)등에 대해 고려해야 한다. 이런 이유로 일반 환경에서는 간단한 계산이 분산 환경에서는 매우 복잡해질 수 있다. 맵리듀스 프로그래밍 모델은 대용량 데이터에 대해 생성과 처리를 수행할 수 있게 구글에 의해 2004 년에 제안되었다[4]. 이 프레임워크는 데이터 배포, 작업 스케줄링, 결함 허용, M2M 통신 등 다양한 문제를 해결할 수 있다. 리듀스 프로그래밍 모델은 맵(map)과 리듀스(reduce) 함수로 매퍼(Mapper)와 리듀서(Reducer)인터페이스를 구현한다. 이것이 작업의 핵심을 형성한다.

2.1.1 맵퍼(Mapper)

맵(Map)함수는 키(key)와 값(value)의 쌍을 입력으로 받아서 작업을 진행하고, 중간 키와 값의 쌍인 그룹을 생성한다. <key,value>는 두 부분으로 구성되며, 키는 값의 그룹 번호를 의미하고, 값은 업무에 관련있는 데이터들을 의미한다. 맵리듀스는 같은 키를 가진 중간 값들을 결합하고, 이것을 리듀스 함수로 전송한다. 맵 알고리즘의 과정은 다음과 같다.

1 단계: 하둡과 맵리듀스 프레임워크는 각 입력 조각에 대해 <key, value>에 기반하여 맵 작업을 생성한다.

2 단계: 입력된 <key,value>에 대한 처리로 새로운 <key,value>을 생성하는 맵 작업을 실행한다. 이 과정은 같은 키를 가진 데이터에 대해서 연관된 값을 생성하는 것이다. 출력 키와 값의 쌍은 입력 키와 값의 쌍과 같은 형태를 취하지 않는다. 주어진 입력 키와 값 쌍은 0 이상의 출력 쌍으로 맵핑 될 수있다.

3 단계: 매퍼의 출력은 각 리듀서에 할당 될 수 있게 정렬된다. 전체 블록 수와 리듀스 작업의 수는 동일하다. 사용자가 특정 키가 특정 리듀서에 할당될 수 있게 조절할 수 있다.

2.1.2 리듀서(Reducer)

리듀스(Reduce)함수는 중간 키와 값 쌍을 처리하는 함수로 사용자에 의해 개발된다. 리듀스 함수는 이 값들을 병합(merge)하여 작은 값의

집합을 구한다. 이 작업은 단순 합이 아니라 복잡한 과정이 존재한다. 리듀서는 같은 키와 연관된 중간 값들을 결합하여 하나의 집합을 구성한다.

맵리듀스 프레임워크에서 <key,value>가 맵리듀스 모델의 통신 인터페이스로 사용되기 때문에 개발자는 데이터 통신의 세부 사항에 대해서는 고려하지 않아도 된다. <key,value>는 "편지"로 볼 수 있다. 키는 편지의 주소이고, 값은 편지의 내용이다. 같은 주소의 편지들은 같은 장소에 배달된다. 맵리듀스 프레임워크가 자동으로 정확하게 같은 키를 가진 값들을 그룹화하기 때문에, 개발자는 정확한 <key,value> 설정만 고려하면 된다. 리듀스 알고리즘의 과정은 다음과 같다.

1 단계: 셔플 (Shuffle). 리듀서의 입력은 정렬된 매퍼의 출력이다. 이 단계에서, 맵리듀스는 각 리듀서에 관련된 블록들을 할당한다.

2 단계: 정렬 (Sort). 다른 맵퍼의 출력이 동일한 키를 가질 수 있기 때문에, 이 단계에서 리듀서의 입력이 키에 따라 그룹화 된다. 셔플과 정렬 단계는 동기화되어야 한다.

3 단계: 2차 정렬. 중간 단계에서 키를 그룹화하는 규칙이 리듀스에서 사용하는 규칙과 다르다면, 리듀스를 위한 키로 변환하는 비교기(Comparator)가 필요하다.

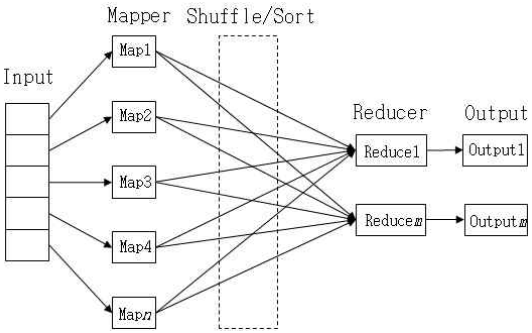
맵 작업과 리듀스 작업 사이에는 맵 작업의 결과를 미리 결합하여 중간 결과를 생성하는 중간 결합자(Intermediate Combiner)가 있을 수 있다. 맵 작업과 리듀스 작업은 분리될 수 없고, MR 프로그램에서 함께 사용되어야 한다. MR 과정(맵리듀스 프로세스)에서 맵 작업과 리듀스 작업 각각 병렬로 실행되고, 맵과 리듀스 작업은 순차적으로 실행된다. 하나의 MR 과정과 다음 MR 과정은 순차적으로 실행되며, 이런 작업 사이의 동기화는 개발자의 개입 없이, MR 시스템에 의해 보장된다. 맵리듀스 작업을 도식화하면 (그림 1)과 같다.

3. 분산 VQ 코드북 생성 방법

3.1 VQ 코드북 생성 방법

VQ 코드북 생성은 보통 군집화 알고리즘을 사용한다. Lloyd의 알고리즘은 대표적인 군집화 알고리즘이다[12, 13]. 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.

(그림 1) 맵리듀스 운영절차



(Figure 1) MapReduce Operation Process

3.2 맵리듀스를 사용한 분산 VQ 코드북 생성

맵리듀스를 적용한 분산 분석 VQ 코드북 생성 알고리즘은 <표 1>과 같다.

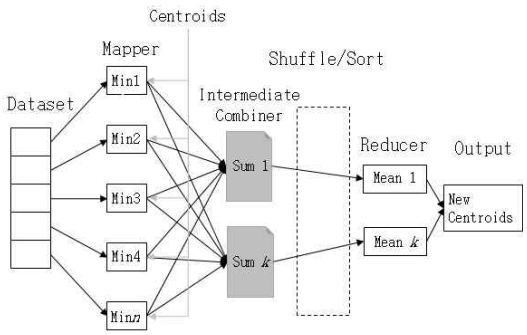
<표 1> 병렬 VQ 코드북 생성 알고리즘

```
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> Parallel VQ Codebook generation algorithm

(그림 2) 병렬 VQ코드북 생성 알고리즘 절차



(Figure 2) Parallel VQ Codebook generation algorithm Process

이를 도식화하면 (그림 2)와 같으며[11, 14], 맵퍼, 중간 결합자, 리듀서가 모두 사용된다.

3.3 디컴바인드 분산 VQ 코드북 생성

제안하는 단축된 분산 VQ 코드북 생성 알고리즘은 <표 2>와 같으며, 이를 도식화 하면, (그림 3)과 같다. <표 1>의 분산 VQ 코드북 생성 알고리즘은 맵 단계에서 키는 군집을 의미하고, 각 군집에 속한 데이터의 좌표를 값에 할당하기 때문에, <n, v>를 위해 n_m*n_{dm}의 데이터 공간이 필요하다. 여기서, n_m은 맵의 개수이고, n_{dm}은 각 맵에 할당된 데이터의 개수이다. 그리고, n_m*n_{dm}=n_d가 된다. 여기서, n_d는 전체 데이터의 개수이다. 하둡은 맵의 출력 결과가 보통 디스크에 저장되기 때문에 데이터의 크기가 크고, 높은 디스크 I/O로 성능 저하가 발생하게 된다. 또한, 리듀스 단계에서 키에 의해 데이터가 나누어져도 전체 데이터의 크기가 필요하기 때문에 디스크 공간과 메모리 사용량이 증가하게 된다.

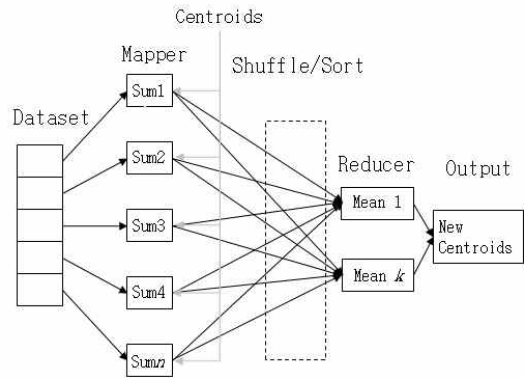
<표 2> 디컴바인드 병렬 VQ 코드북 생성 알고리즘

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 = sum of the coordinates of the data point
nv = number of data points which

assigned to the cluster
output the <*n*, *v*> (assign data point *v* to cluster *n*)
output the <*n*, *nv*> (assign number of data point *nv* to cluster *n*)
end for
end for
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 2> Decombined parallel VQ Codebook generation algorithm

(그림 3) 디컴바인드 병렬 VQ 코드북 생성 알고리즘의 절차



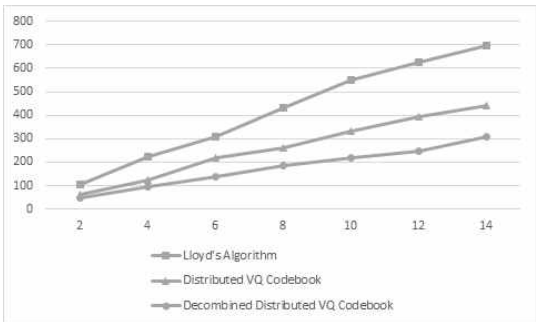
(Figure 3) Decombined parallel VQ Codebook generation algorithm process

제안하는 디컴바인드 분산 VQ 코드북 생성 알고리즘은 맵 단계에서 키는 군집을 의미하고, 값은 개별 데이터의 좌표를 각각 할당하는 것이 아니라, <표 1>의 중간 결합자 작업에서 수행하는 데이터들의 평균을 할당하게 된다. 하지만, 평균을 직접 계산하면 리듀스 단계에서 각 군집의 평균을 계산할 수 없기 때문에 평균을 이루는 기본 요소인, 데이터 좌표의 합과 군집에 속한 데이터의 수를 계산한다. 제안하는 방법은 <*n*, *v*>와 <*n*, *nv*>를 위해 (n_m*n_c)*2의 데이터 공간이 필요하다. 여기서, n_c는 군집의 개수이다. 일반적으로 빅 데이터 환경에서 군집의 수는 매퍼의 데이터의 수보다 작기 때문에 (n_{dm}* >> 2*n_c) 메모리와 디스크 사용량이 감소한다. 이는 리듀스 단계에서도 마찬가지로 적용되어 메모리와 디스크 사용량이 감소하게 되며, 전체 수행 시간이 감소하게 된다.

4. 실험환경 및 결과

본 실험에서는 빅 데이터에 맞는 합성 데이터를 만들어서 사용했다. 데이터들은 2차원 데이터로 12개의 군집으로 이루어졌으며, 각 데이터들은 군집의 중심을 기준으로 가우시안 분포를 따른다. 군집에 속하지 않는 노이즈 데이터가 약 1% 정도 존재한다. 맵리듀스를 사용한 분산 VQ 코드북 생성 방법과 제안하는 디컴바인드 분산 VQ 코드북 생성 방법은 하나의 컴퓨터에서 동작하는 Lloyd의 알고리즘을 기반으로 하기 때문에 VQ 코드북을 생성한 결과는 모두 동일하다.

(그림 4) Loyd's 알고리즘, 분산 병렬 VQ 코드북 알고리즘, 디컴바인드 분산 병렬 VQ 코드북 알고리즘의 실행시간 비교

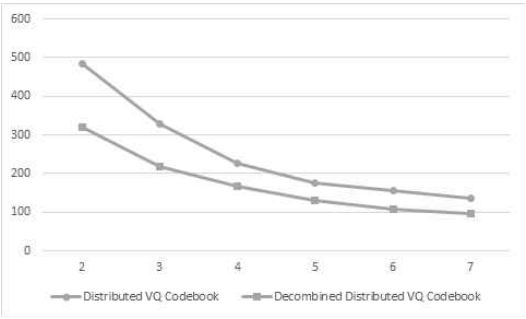


(Figure 4) The Executing Time of Stand-alone Lloyd's Algorithm, the Distributed Parallel VQ Codebook Algorithm and Decombined Distributed Parallel VQ Codebook Algorithm.

(그림 4)는 한 개의 컴퓨터에서 맵리듀스를 사용하지 않고 Lloyd의 알고리즘을 수행한 결과와 맵리듀스를 사용한 분산 VQ 코드북 생성 방

법, 제안하는 디컴바인드 분산 VQ 코드북 생성 방법을 수행한 결과를 하나의 iteration의 수행 시간을 비교한 그래프이다. 맵리듀스는 3노드 환경에서 수행하였다. 제안하는 디컴바인드 분산 VQ 코드북 생성 알고리즘이 하나의 컴퓨터에서 수행한 결과 뿐만 아니라 맵리듀스를 사용한 분산 VQ 코드북 생성 알고리즘 보다 좋은 결과를 보였다. 또한, 데이터가 증가할수록 수행 시간도 거의 선형의 증가를 보이고 있어서, 더 많은 데이터에 대해서도 안정적으로 동작할 수 있음을 확인할 수 있다.

(그림 5) 노드수를 변경시킨 상태에서의 병렬 VQ 코드북 알고리즘의 결과



(Figure 5) Results of Parallel VQ Codebook Algorithm under Different Numbers of Nodes

하둡 분산 시스템의 노드 수에 따른 군집화의 수행 시간의 비교는 (그림 5)와 같다. 맵리듀스를 사용한 분산 VQ 코드북 생성 알고리즘과 디컴바인드 분산 VQ 코드북 생성 알고리즘을 비교하였으며, 2차원 데이터의 수는 1억 건이고, 군집의 수는 12개이다. (그림 4)의 실험과 마찬가지로 한번의 iteration의 시간을 측정하였다. 디컴바인드 분산 VQ 코드북 생성 알고리즘의 성능이 더 우수했으며, 노드 수가 많아질수록 수행 시간이 감소하는 추세를 보였다. 이는 노드 수가 많아질수록 하나의 노드(컴퓨터)에서 수행되어야 할 연산이 감소하기 때문이다.

5. 결 론

본 논문은 맵리듀스를 사용한 분산 VQ 코드북 생성 알고리즘의 성능을 개선하는데 목적을 가지고 있다. 성능을 측정하는 여러 척도 중 수행 시간 관점에서 접근을 하였고, 실험 결과 제안하는 방법에 의하여 수행 시간이 감소하는 것을 확인할 수 있었다.

분산 VQ 코드북 생성 알고리즘도 일반적인 VQ 코드북 생성 알고리즘인 Lloyd의 알고리즘이 가지는 단점을 가지고 있다. 즉, 군집의 개수인 k의 수를 미리 설정해야 하고, 처음 군집의 중심을 초기화하는 방법에 따라 최종적으로 다른 군집의 결과를 보일 수 있으며, 그 품질도 큰 차이를 보인다.

컴퓨터의 능력(computing power)이 증가하면서 일반적으로 데이터 분석 알고리즘의 수행 시간은 단축되었다. 하지만, 빅 데이터 시대로 오면서 가용 데이터의 양이 극적으로 증가하게 되어, 빅 데이터에 대한 분석 알고리즘의 수행 시간은 증가하고 있다.

본 논문에서는 맵리듀스 기반의 분산 분석 알고리즘을 통하여, 빅 데이터에 대한 분석 속도를 향상시킬 수 있었다. 또한, Lloyd의 알고리즘의 경우 기존의 단점을 극복하기 위한 다양한 기법이 연구되고 있으며, 이를 분산 환경에 적용하게 되면, 분석 성능을 보다 향상 시킬 수 있을 것이다.

References

[1] Tzu-Chuen Lu, and Ching-Yun Chang, “A Survey of VQ Codebook Generation,” Journal of Information Hiding and Multimedia Signal Processing, vol. 1, no. 3, pp. 190-203, 2010.

[2] C. W. Tsai, C. Y. Lee, M. C. Chiang, and C. S. Yang, “A Fast VQ Codebook Generation Algorithm via Pattern Reduction, Pattern Recognition Letters,” vol. 30, pp. 653-660, 2009.

[3] Xiao-Gang W, and Yue L, “Web mining based on user access patterns for web personalization,” ISECS International Colloquium on Computing, Communication, Control, and Management. 1: 194-197, 2009.

[4] J. Dean and S. Ghemowat, “MapReduce : Simplified data processing on large clusters,” in OSDI, 2004.

[5] Seongeun Yang, Changyeol Choi, Hwangkyu Choi, “Design and Implementation of Vehicle Route Tracking System using Hadoop-Based Bigdata Image Processing,“ Journal of Digital Contents Society, vol.14, no.4, pp.447-454, 2013.

[6]Yang, Hadoop, “http://hadoop.apache.org/”

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

[8] Sim, Gyu-Seok ; Kim, Yeong-Hun ; Lee, Jeong-Hun ; Kim, Jin-Hyeon ; Park, Yun-Jae , "Current Research Trends in MapReduce Algorithms for Big Data Analysis," Communications of the Korean Institute of Information Scientists and Engineer, Vol. 32, No. 1, pp. 27-32, 2014.

[9] Krishnamoorthy R, Kalpana J, “Minimum distortion clustering technique for orthogonal polynomials transform vector quantizer,” Proc. 2011 Inter. Conf. Communication, Computing & Security. 443-448, 2011.

[10] Dumitrescu S, Wu X, “On properties of locally optimal multiple description scalar quantizers with convex cells,” IEEE Trans. Inform. Theor. 55: 5591-5606, 2009.

[11] 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.

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

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

[14] 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.

이 현 진

1996년: 순천향대학교 전산학과(공학사)
1998년: 연세대학교 대학원 컴퓨터과학 (공학석사)
2002년: 연세대학교 대학원 컴퓨터과학 (공학박사)
2003년~현재: 숭실사이버대학교 컴퓨터정보통신학과 부교수
관심분야 : 스마트러닝, 기계학습, 빅데이터