Journal of the Korean Data & Information Science Society
2016, **27**(3), 609–619

http://dx.doi.org/10.7465/jkdi.2016.27.3.609
한국데이터정보과학회지

빅데이터 K-평균 클러스터링을 위한 RHadoop 플랫폼[†]

신지은[1] · 오윤식[2] · 임동훈[3]

[13]경상대학교 정보통계학과 · [2]경상대학교 생명과학부

접수 2016년 3월 7일, 수정 2016년 3월 22일, 게재확정 2016년 3월 23일

요 약

본 논문에서는 대용량 데이터를 처리 및 분석하기 위해 RHadoop 플랫폼에서 실제 데이터와 모의 실험 데이터를 가지고 K-평균 클러스터링을 구현하고, MapReduce의 컴바이너 사용여부에 따른 처리 속도를 비교하고자 한다. 또한, K-평균 클러스터링에서 최적의 군집수 결정방법을 MapReduce 프로그램으로 구현하여 실제 데이터에 적용하고자 한다. 그리고 제안된 RHadoop 플랫폼의 확장 가능성을 보이기 위해 실제 데이터에서 R의 기본 패키지에서 kmeans() 함수와 bigmemory 패키지 상에서 유용한 bigkmeans() 함수와 처리 속도를 비교하고자 한다.

주요용어: 빅데이터, Hadoop, K-평균 클러스터링, R, RHadoop

1. 서론

현재 우리사회는 정보혁명에 이은 '스마트 혁명'이라는 '제4의 물결' 시대를 맞이하고 있다. 새로운 스마트시대 주요 패러다임을 선도하기 위해서는 빅데이터 (big data)의 활용이 핵심이며, 그 수준이 미래의 경쟁력과 성패를 좌우한다고 해도 과언이 아니다.

빅데이터에서 클러스터링 (clustering)은 빅데이터 마이닝(big data mining) 기법 중 하나로 개체들 간의 유사도 (similarity)에 따라 유사한 개체들끼리 묶는 통계적 방법으로 인공지능, 패턴인식, 경제학, 생태학 그리고 마케팅 등 여러 분야에서 활용되고 있다. K-평균 클러스터링 (K-Means clustering)은 비계층적 클러스터링 (non-hierarchical clustering)의 대표적인 분류방법으로 대용량의 데이터에 적합한 방법이다.

Hadoop은 대용량 데이터를 분산처리 할 수 있는 자바 기반의 오픈소스 프레임 워크로서 분산파일 시스템인 HDFS (Hadoop distributed files system)에 데이터를 저장하고 분산처리 시스템인 MapReduce를 이용하여 데이터를 처리 한다 (White, 2012). Hadoop에서 컴바이너 (combiner)는 MapReduce 파이프라인에서 Mapper와 Reducer 사이에 존재하는 인터페이스로 적절히 사용하면 성능개선에 도움이 된다 (Anchalia, 2014).

R과 Hadoop의 통합환경으로는 Rhipe (Guha, 2010; Ko와 Kim, 2013; Jung 등, 2014)와 Rhadoop 등이 있다. RHadoop은 레볼루션 어낼리틱스 (revolution analytics)에 의해 개발되어 Prajapati (2013), Oancea와 Dragoescu (2014), Harish 등 (2015)에 의해 연구가 진행되고 있고 국내에서는 주로 Park 등 (2013), Shin 등 (2015)에 의해 연구가 이루어져왔다.

[†] 이 연구는 2015년도 경상대학교 발전기금재단 재원으로 수행되었음.
[1] (660-701) 경남 진주시 가좌동 900번지, 경상대학교 정보통계학과, 석사수료.
[2] (660-701) 경남 진주시 가좌동 900번지, 경상대학교 생명과학부, 교수
[3] 교신저자: (660-701)경남 진주시 가좌동 900번지, 경상대학교 정보통계학과, 교수 및 RINS.
E-mail: dhlim@gnu.ac.kr

본 논문에서는 빅데이터를 위한 RHadoop 플랫폼에서 Hadoop의 컴바이너를 사용하여 K-평균 클러스터링의 처리 속도를 개선하고자 한다. 또한, K-평균 클러스터링에서 최적의 군집 수를 결정하기 위한 Elbow 방법 (Kodinariya와 Makwana, 2013)을 처음으로 MapReduce 프로그램으로 구현하고, 제안된 RHadoop 플랫폼의 확장가능성 (scability)을 보이기 위해 실제 데이터와 모의실험 데이터에서 기존의 R함수 kmeans()와 bigmemory 패키지 상에서 유용한 bigkmeans() (Kane과 Emerson, 2010a, 2010b)와 처리 속도를 비교하였다.

본 논문은 다음과 같이 구성되어 있다. 제 2절과 3절에서는 각각 Hadoop과 RHadoop에 대해 간략하게 살펴보고, 제 4절에서는 RHadoop 플랫폼에서 K-평균 클러스터링의 성능에 대해 살펴보고, 또한 군집수 결정 방법에 대해서도 논의하고자 한다. 그리고 제 5절에서 결론을 맺고자 한다.

2. Hadoop 개요

Hadoop은 대용량 데이터를 처리할 수 있는 소프트웨어로 HDFS와 MapReduce로 구성되어 있다. HDFS는 물리적으로 네임 노드 (name node)와 데이터 노드 (data node)로 구성되어 있다. 네임노드는 파일의 디렉토리 구조, 권한과 같은 메타 정보를 저장하고, 실제 데이터는 여러 대의 데이터 노드에 분산되어 저장한다. MapReduce는 잡 트랙커 (job tracker)와 테스크 트랙커 (task tracker)로 구성되고 HDFS에 저장된 파일에서 데이터를 읽어서 가공하는 역할을 수행한다 (Jung 등, 2014; Sammer, 2012).

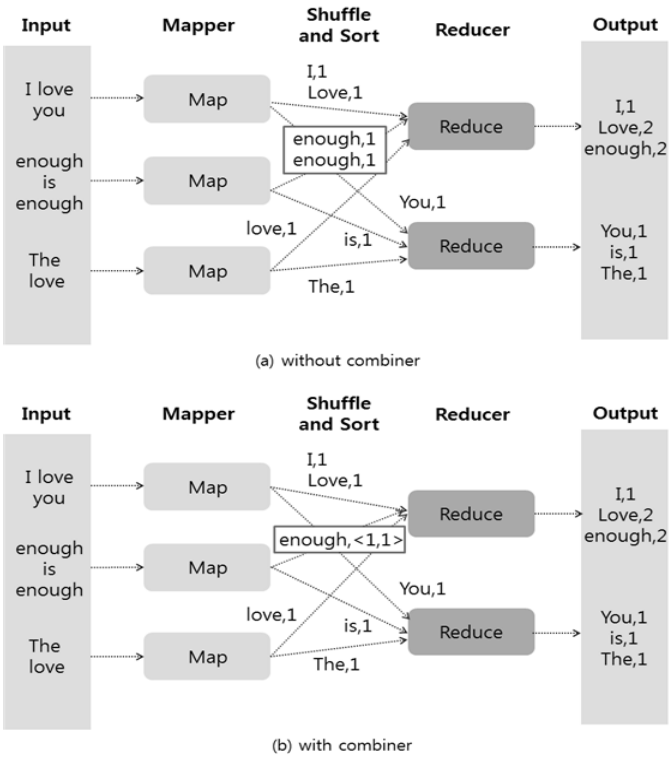

Figure 2.1 MapReduce parallel processing (a) without combiner (b) with combiner

MapReduce의 병렬처리는 함수의 입력과 출력이 모두 <키 (key), 값 (value)> 쌍으로 구성되어 있다. Figure 2.1는 단어 세기 (word count) 예를 가지고 컴바이너 사용여부에 따라 MapReduce의 동작 과정을 보여주고 있다. Figure 2.1에서 "Input"은 입력데이터이고 "Mapper"와 "Reducer"는 각각 Map 함수와 Reduce 함수의 기능을 수행하는 과정이고 "Suffle and Sort"는 Mapper로부터 같은 키를 갖는 값들로 분류한 후 여러 Reducer 작업을 위해 파티션 (partition)되어 이동하는 과정을 말한다. 그리고 "Output"은 출력데이터를 나타낸다. MapReduce의 동작과정에서 컴바이너를 사용하면 Mapper 과정에서 각 노드마다 계산된 결과를 키를 기준으로 집계하여 Reducer로 전달됨으로서 Reducer의 입력 데이터의 양을 줄여 준다.

3. RHadoop 개요

RHadoop은 R과 Hadoop의 통합환경을 제공하는 오픈 소스 플랫폼으로 3개의 주요한 R패키지 즉, rmr, rhdfs 그리고 rhbase으로 구성되어 있다. rmr은 R에서 MapReduce 기능을 제공하는 패키지이고, rhdfs는 R에서 HDFS 파일 관리 기능을 제공하는 패키지이고, rhbase는 R에서 HBase 데이터베이스 관리를 제공하는 패키지이다. 본 논문은 데이터베이스 연동과 관련이 없으므로 rmr과 rhdfs 패키지만을 사용하고 있다 (Prajapati, 2013).

Table 3.1는 rmr과 rhdfs 패키지에 있는 함수를 나타낸 표이다.

Table 3.1 Various functions for HDFS and MapReduce operations

Category	Function	Description
Initialization	hdfs.init()	Initialize HDFS
File manipulation	hdfs.put()	Copy files from the local filesystem to HDFS filesystem
	hdfs.get()	Copy files from HDFS filesystem to the local filesystem
	hdfs.move()	Move a file from one HDFS directory to another HDFS directory
	hdfs.rm()	Delete the directory of file from HDFS
	hdfs.chmod()	Change permissions of some files
File read /write	hdfs.file()	Initialize a file to be used for read/write operation
	hdfs.read()	Read the file from HDFS
	hdfs.write()	Write in the file stored at HDFS
	hdfs.close()	Close the stream when a file operation is complete
rmr	from.dfs()	Read R objects from the HDFS filesystem
	to.dfs()	Write R objects from or to the filesystem
	keyval()	Create and extract key-value pairs
	mapreduce()	Execute MapReduce job
Utility	hdfs.ls()	List the directory from HDFS

다음은 제 2절의 단어 세기 예를 가지고 RHadoop에서 MapReduce 프로그램 구현에 대해 설명하고자 한다.

• Map 함수

Map 함수에서는 HDFS상의 데이터를 가공하여 새로운 <키, 값> 쌍의 집합을 출력하는 역할을 수행한다. 다음은 단어 세기에 대한 RHadoop에서 Map 스크립트를 나타내고 있다.

$$
\begin{aligned}
map \quad = \quad & function(., v)\{ \\
& words.list \leftarrow strsplit(v, split = \text{""}) \\
& words \leftarrow unlist(words.list) \\
& keyval(words, 1) \\
\}&
\end{aligned}
$$

위 스크립트에서 map 함수는 HDFS에서 불러온 데이터를 리스트 형태로 읽어 들여 처리한다. 그리고 keyval() 함수를 이용하여 <키, 값> 쌍의 집합을 생성한 후 HDFS로 전송한다.

• Reduce 함수

Reduce 함수는 Map에서 전달된 <키, 값> 쌍을 같은 키를 갖는 값들로 묶어 새로운 <키, 값> 쌍의 집합을 만든 후, 집계연산을 수행하여 새로운 <키, 값> 쌍의 집합을 생성하는 역할을 수행한다. 다음은 단어 세기에 대한 RHadoop에서 Reduce 스크립트를 나타내고 있다.

$$\begin{aligned} reduce \quad = \quad & function(k, v)\{ \\ & keyval(k, sum(v)) \\ \} & \end{aligned}$$

위 스크립트에서 Reduce는 sum() 함수에 의해 집계된 <키, 값> 쌍을 keyval() 함수를 사용하여 HDFS로 전송한다.

MapReduce 실행은 다음의 mapreduce() 함수를 이용하여 Map과 Reduce를 실행시키는 역할을 수행한다.

$$mapreduce(input, input.format, map, reduce, output, combine)$$

위 mapreduce() 함수는 input 인자, map 인자, reduce 인자 등 다양한 인자를 지정한다. 이때, 텍스트 파일을 읽어올 경우 input.format='text'로 지정하고, combine 인자는 컴바이너 사용 여부를 지정하는 인자이며 기본값은 FALSE이다.

4. RHadoop 플랫폼에서 K-평균 클러스터링 성능 분석

4.1. 실험환경

본 논문에서 사용된 실험환경은 Table 4.1과 같이 Hadoop 0.20.0 기반 하에 6대의 개인용 PC를 사용하여 그 중 1대 PC를 마스터 (master) 노드, 나머지 5대 PC를 슬레이브 (slaves) 노드로 설정하여 Figure 4.1과 같이 클러스터를 구축하였다.

Table 4.1 RHadoop cluster environment

Nodes	master		1 EA
	slaves		5 EA
Physical spec	CPU		Intel(R)Core(TM)2 Duo CPU E8300@2.83GHz
	RAM	master	4GB
		slaves	2GB
	Network Interface Card		RealTek
Software version	OS		12.04LTS
	Java		1.7.0
	Hadoop		0.20.2
	R		3.1.0
	Rhipe		0.73.1
	Google Protocal Buffer		2.4.1
Switch Hub	Cisco Catalyst 2960 (1G Ethernet)		

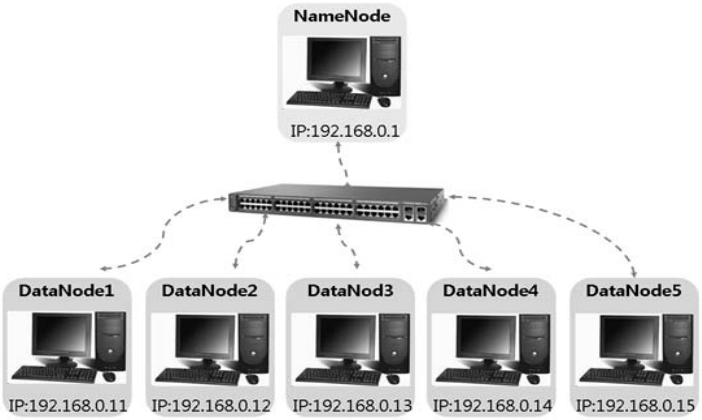

Figure 4.1 Experimental setup of 6 nodes

본 논문에서 시스템 처리시간은 R의 system.time() 함수에서 제공하는 elapsed time을 가지고 측정하였다. 여기서 elapsed time은 프로세스의 전체 경과시간을 나타내는데 흔히, 벽시계 시간 (wall clock time)을 의미한다. 동일한 시스템 조건하에서 패키지들의 성능을 비교하기 위해 각 프로세스 실행 시 메모리를 초기화 (clear)한 후 측정하였으며 측정오차를 최소화하기 위해 CPU의 유휴상태 (idle state)에서 시간 측정을 실시하였다 (Ciliendo와 Kunimasa, 2007). 여기서 CPU의 유휴상태 판단은 CPU 유휴 (idle) 값이 98% 이상 높은 수치를 나타내는 경우로 제한하였다.

4.2. 실험 데이터

(1) 실제 데이터

실제 데이터는 2009년 ASA (American Standards Association:미국 규격 협회)에서 공개된 미국 항공기 운항과 관련된 데이터이다 (ASA data expo, 2009). 이 항공기 데이터는 1987년부터 2008년까지 해마다 29개 변수에 대해 조사된 데이터이고 전체 데이터의 행은 123,534,970개이고 데이터의 크기는 12 GB정도이다. 본 논문은 1987년부터 2008년까지의 데이터를 모두 사용하였으며 결측값 (missing values)이 많은 변수를 제외하고 11개의 변수를 사용하였고 Table 4.2은 사용된 데이터 일부를 나타낸 것이다.

Table 4.2 A part of real airline data

Dayof Month	Dayof Week	Dep Time	CRS DepTime	Arr Time	CRS ArrTime	Actual Elapsed Time	CRS Elapsed Time	Arr Delay	Dep Delay	Distance
14	3	741	730	912	849	91	79	23	11	447
15	4	729	730	903	849	94	79	14	-1	447
17	6	741	730	918	849	97	79	29	11	447
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
13	6	655	700	856	849	121	116	0	-5	545
13	6	1251	1240	1446	1437	115	117	9	11	533
13	6	1110	1103	1413	1418	123	135	-5	7	874

Table 4.2에서 사용된 변수 DayofMonth는 월 (1∼12), DayofWeek는 요일 (1∼7), DepTime는 출발시각, CRSDepTime는 실제 출발시각, ArrTime는 도착시간, CRSArrTime는 실제도착시각, ActualElapsedTime는 실제경과시간, CRSElapsedTime는 예정 결과시간, ArrDelay는 도착지연, DepDelay는 출발지연, Distance는 거리를 나타낸다.

(2) 모의실험 데이터

실험에 사용될 모의실험 데이터는 다음과 같이 단계별로 생성한다.

STEP 1: 정규 난수를 원소로하는 $m \times p$ 행렬을 생성한다. 즉, $m \times p$ 행렬의 원소 X_{ij}는 다음과 같다.

$$X_{ij} \sim N(0, \sigma^2), i = 1, \ldots, m, j = 1, \ldots, p$$

여기서 σ=10이다.

STEP 2: STEP 1에서 생성된 $m \times p$ 행렬을 n개 복사하여 $(m \times n) \times p$ 행렬을 생성한다. 즉, $(m \times n) \times p$ 행렬의 원소 X_{ij}는 다음과 같다.

$$X_{ij} \sim N(0, \sigma^2), i = 1, \ldots, m \times n, j = 1, \ldots, p$$

STEP 3: 표준 정규 난수를 원소로 하는 $(m \times n) \times p$ 행렬을 생성한다. 즉, $(m \times n) \times p$ 행렬의 원소 Z_{ij}는 다음과 같다.

$$Z_{ij} \sim N(0, 1), i = 1, \ldots, m \times n, j = 1, \ldots, p$$

STEP 4: STEP 2에서 생성된 행렬 $\{X_{ij}\}$ 과 STEP 3에서 생성된 행렬 $\{Z_{ij}\}$ 을 덧셈 연산하여 얻어진 행렬 $\{X_{ij} + Z_{ij}\}$, $i = 1, \ldots, m \times n, j = 1, \ldots, p$, 로부터 실험 데이터를 얻는다.

본 논문에서는 $m = 5$, $n = 20$, $p = 10$이다.

4.3. MapReduce 클러스터링 알고리즘

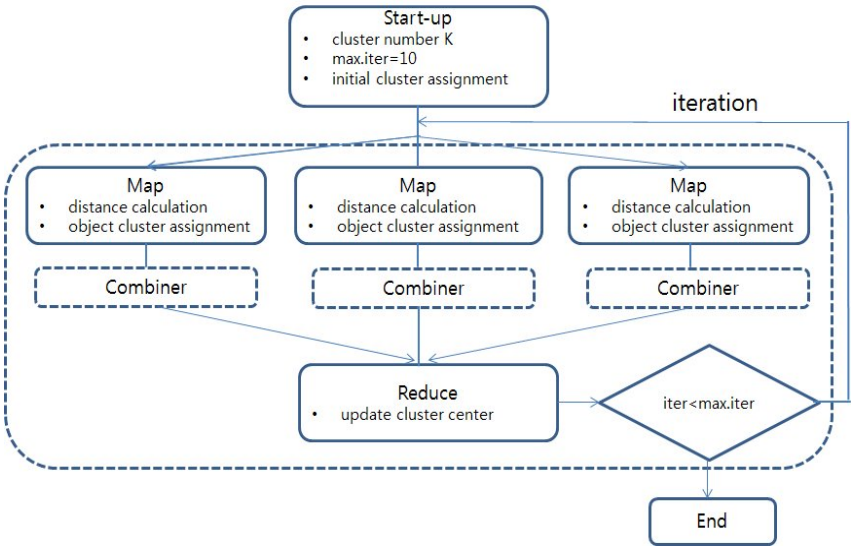

Figure 4.2 MapReduce K-means clustering flowchart

본 절에서는 K-평균 클러스터링을 Map과 Reduce의 단계를 나누어 MapReduce 알고리즘을 구현한다. Figure 4.2은 MapReduce 알고리즘의 순서도를 나타내고 있다. Start-up에서는 군집수 K와 최대 반복횟수인 max.iter =10를 지정하고, 초기개체군집을 임의로 할당한다. Map에서는 거리계산을 통해 개체군집을 할당한다. 그리고 컴바이너 사용여부에 따라 map 함수의 수행결과를 Reduce로 보낸다. Reduce에서는 군집중심을 업데이트한다. 지정된 최대 반복횟수를 만족하면 군집 중심이 변화가 없는 것으로 간주하여 반복을 멈춘다.

```
REQUIRE :
   · Input (key, value) where key = null, and value = objects
   · A set of d−dimensional objects of {x_1,x_2,...,x_n} in each mapper
   · initial set of cluster centers C= {c_1,c_2,...,c_k}
1. M ← {x_1,x_2,...,x_n}
2. for all x_i ∈ M do
3.     if ( C = ∅ ) then
4.      current_centroid ← C
5.     else
6.         for all c_k ∈ C do
7.                dist ← ||x_i − c_k||
8.              if dist < minDist then
9.                 minDist ← dist
10.                index ← k
11.             end if
12.          end for
13.        end if
14.   end for
15. outputlist << (index, (1, object))
16. end
   · output : (key, value), where key is the index of the closest center
    point, and the value is a set of 1 and objects assigned to the index of
    the closest centers.
```

Figure 4.3 Pseudo-code for map function

```
REQUIRE : Input (key, value) where key = index of the closest center,
    and value = a set of 1 and objects assigned to the index of the
    closest centers by the mapper.
1. outputlist ← outputlist from mappers
2. V ← ∅
3. newCentroid ← ∅
4. for all y ∈ outputlist do
5.    index ← y.key
6.    setofobject ← y.value
7.    V_index ← setofobject
8.  end for
9.  for all v ∈ V do
10.    sumofObject, numofObject ← ∅
11.     for all object ∈ V_i do
12.           sumofObject += object
13.           numofObject += 1
14.      end for
15.  end for
16.     outputlist << (index, (numofObject, sumofObject))
17.    newCentroid ← (sumofObject ÷ numofObject)
18. end
 · output : (key, value), where key is the index of the closest center
  point, and the value is a set of numofObject and sumofObject.
```

Figure 4.4 Pseudo-code for reduce function

Figure 4.3은 MapReduce에서 컴바이너를 사용하는 경우 map 함수 의사코드 (pseudo-code)를 나타낸다. Figure 4.3로부터 7번 코드에서 개체와 군집 중심과의 거리를 계산하고 8-11번 코드에서 가장 짧은 거리를 갖는 군집에 개체를 할당하고 있다. map 함수 수행결과 <키, 값>으로 <군집, (1, 개체)>을 출력한다.

Figure 4.4은 MapReduce에서 컴바이너를 사용하는 경우 reduce 함수 의사코드를 나타낸다. Figure 4.4로부터 4-8번 코드에서 입력 데이터 <군집, (1, 개체)>에서 군집에 따라 개체들을 모으고 있고 코드 16에서 군집을 형성하는 개체수와 개체합을 계산한다. reduce 함수 수행결과 <키, 값>으로 <군집, (개체수, 개체합)>을 출력한다. 코드 17에서 새로운 군집의 중심을 계산하고 있다.

4.4. 최적의 군집 수 K 결정

K-평균 클러스터링에서 군집수 K는 사전에 주어져야 한다. 본 논문에서는 Elbow 방법을 이용하여 그래픽에서 휴리스틱 (heuristic)하게 최적의 군집수를 결정한다. 이를 위해 군집에서 오차 제곱합 (sum of squared error; SSE)를 다음과 같이 정의한다.

$$\text{SSE} = \sum_{k=1}^{K} \sum_{x_i \in S_k} ||x_i - c_k||^2$$

여기서 x_i는 i번째 개체이고 c_k는 k번째 군집 S_k의 중심을 나타낸다.

Elbow 방법은 군집수에 대한 SSE의 플롯 (plot)에서 k 값 증가에 대해 SSE의 감소폭이 둔화되는 지점의 군집수를 최적의 군집수로 결정한다.

Figure 4.5(a)와 4.5(b)는 각각 실제 데이터와 모의실험 데이터로부터 기존의 kmeans 함수와 RHadoop 플랫폼을 이용하여 얻은 Elbow 플롯이다. Figure 4.5(a)와 4.5(b)는 각각 4.5GB 실제 데이터와 17GB 모의실험 데이터로부터 랜덤으로 추출한 100MB 데이터에 대한 플롯이다.

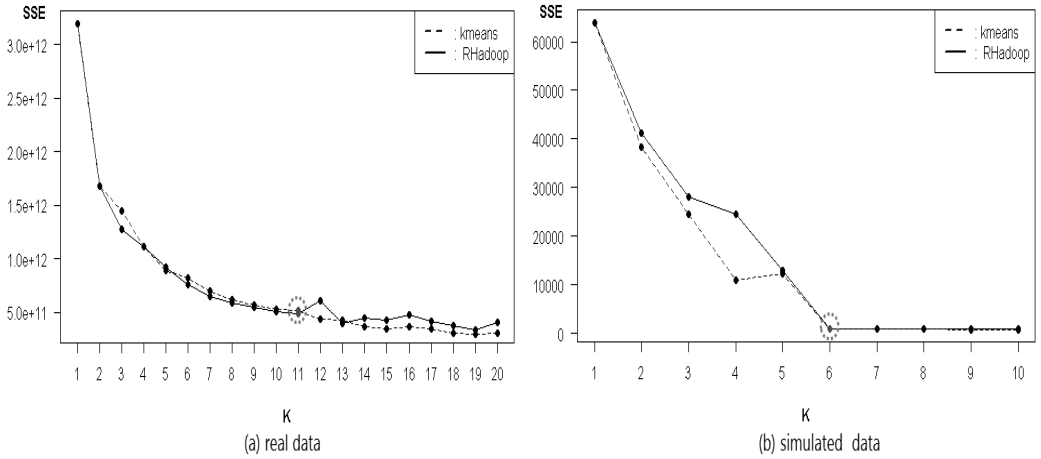

Figure 4.5 Elbow plots using kmeans and RHadoop

Figure 4.5(a)의 실제 데이터에서 kmeans 함수와 RHadoop 플랫폼 모두에서 최적의 군집수가 11이고 Figure 4.5(b)의 모의실험 데이터에서도 kmeans 함수와 RHadoop 플랫폼 모두 최적의 군집수는 6임을 알 수 있다.

4.5. 컴바이너 사용 여부에 따른 RHadoop 성능 비교

본 절에서는 실제 데이터와 모의실험 데이터에서 R의 기본 패키지, bigmemory 패키지 그리고 RHadoop 플랫폼 간의 K-평균 클러스터링에 대한 처리 속도를 비교하고 또한, RHadoop 플랫폼에서 컴바이너의 사용여부에 따라 성능을 비교하였다.

Table 4.3과 Table 4.4는 각각 실제 데이터와 모의실험 데이터의 여러가지 데이터 크기에 따라 R에서 제공하는 kmeans() 함수와 bigmemory에서 제공하는 bigkmeans() 함수 그리고 RHadoop에서 컴바이너 사용유무에 따라 각각의 K-평균 클러스터링의 처리속도를 나타낸 것이다.

Table 4.3 Comparisons of three packages with actual data in terms of computational time in seconds

Data size (no.of rows)	R	bigmemory	RHadoop	
			Combine=F	Combine=T
100 MB (2,542,600 lines)	35.41	99.902	653.018	562.91
500 MB (12,713,000 lines)	Fail	465.661	951.732	951.028
1.0 GB (25,426,000 lines)	Fail	817.125	1713.941	1697.358
3.0 GB (76,278,000 lines)	Fail	Fail	4084.119	4002.381
4.5 GB (120,748,239 lines)	Fail	Fail	6216.888	6142.324

Table 4.3에서 kmeans() 함수의 경우 100MB까지 처리할 수 있고, bigkmeans() 함수의 경우 1.0GB까지 처리할 수 있었다. RHadoop의 경우 데이터 크기에 상관없이 모두 처리가 가능하였고, 컴바이너를 사용하였을 때가 컴바이너를 사용하지 않을 때보다 처리 속도가 빠른 것으로 나타났다.

Table 4.4 Comparisons of three packages with simulated data in terms of computational time in seconds

Data size (no.of rows)	R	bigmemory	RHadoop	
			Combine=F	Combine=T
100 MB (600,000 lines)	15.492	8.288	414.285	412.622
500 MB (3,000,000 lines)	69.584	38.754	478.855	476.518
1.0 GB (6,000,000 lines)	347.45	83.802	768.305	762.285
3.0 GB (18,000,000 lines)	Fail	279.109	1656.030	1636.168
4.5 GB (27,000,000 lines)	Fail	350.622	2342.785	2314.598
13.5 GB (81,000,000 lines)	Fail	Fail	6628.906	6515.830

Table 4.4에서 kmeans() 함수의 경우 1.0GB까지 처리가 가능하였고, bigkmeans() 함수의 경우 4.5GB까지 처리할 수 있었다. RHadoop의 경우 데이터 크기에 상관없이 모두 처리가 가능하였고, 컴바이너를 사용하였을 때가 컴바이너를 사용하지 않을 때보다 처리 속도가 빠른 것으로 나타났다.

5. 결론

인터넷과 스마트폰의 대중화로 인해 엄청난 양의 데이터 생성으로 기존의 데이터 저장, 관리, 분석 기법으로는 데이터를 처리하는데 한계가 있어 새로운 정보기술의 패러다임이 요구되고 있다.

통계분석이 가능한 R과 빅데이터 처리가 가능한 Hadoop의 통합환경인 RHadoop 플랫폼의 등장으로 분산처리 환경 하에서 대용량의 데이터 처리 및 분석이 가능해졌다.

본 논문에서는 빅데이터 처리를 위한 RHadoop 플랫폼 상에서 Hadoop의 컴바이너를 사용하여 K-평균 클러스터링의 처리속도를 개선하였다. 또한 본 논문은 최적의 군집 수 결정을 위한 Elbow 방법을 MapReduce로 처음으로 구현하였으며, 실제 데이터와 모의실험 데이터를 사용하여 RHadoop 플랫폼의 성능실험을 실시하였다. 우리는 실험결과 다음의 몇 가지 결과를 얻을 수 있었다.

첫째, R의 기본패키지의 kmeans()함수와 bigmemory 패키지 상의 bigkmeans()함수를 RHadoop 플랫폼과 비교 결과, kmeans()함수는 작은 데이터 처리만 가능하고, bigkmeans() 함수는 어느 정도 큰 데이터 처리는 가능하지만 한계가 있는 것으로 나타났다. 하지만, RHadoop은 데이터의 용량에 관계없이 처리가 가능한 것으로 나타났다. 둘째, Elbow 방법을 사용한 최적의 군집 수 결정은 기존의 kmeans 함수 결과와 비슷한 결과를 얻을 수 있었다. 셋째, RHadoop 플랫폼에서 컴바이너를 사용하였을 때 처리속도면에서 성능이 좋은 것으로 나타났다.

References

Anchalia, P. P. (2014). Improved MapReduce k-means clustering algorithm with combiner. *16th International Conference on Computer Modelling and Simulation*, 386-391.

ASA Data Expo. (2009). Airline on-time performance, *ASA section on: Statistical computing statistical graphics*, `http://stat-computing.org/dataexpo/2009/the-data.html.`

Ciliendo, E. and Kunimasa, T. (2007). *Linux performance and tuning guidelines*, International Technical Support Organization, IBM, ibm.com/redbooks.

Guha, S. (2010). *Computing environment for the statistical analysis of large and complex data.* Ph. D. Thesis, Purdue University, West Lafayette.

Harish, D., Anusha, M.S. and Dr. Daya Sagar, K. V. (2015). Big data analysis using Rhadoop. *International Journal of Innovative Research in Advanced Engineering*, **4**, 180-185.

Jung, B. H., Shin, J. E. and Lim, D. H. (2014). Rhipe platform for big data processing and analysis, *The Korean Journal of Applied Statistics*, **27**, 1171-1185.

Kane, M. J. and Emerson, J. W. (2010a). *biganalytics: A library of utilities for big.matrix objects of package bigmemory*, R package version 1.0.12, `http://CRAN.R-project.org/package=biganalytics.`

Kane, M. J. and Emerson, J. W. (2010b). *bigmemory: Manage massive matrices with shared memory and memory-mapped files*, R package version 4.2.3, `http://CRAN.R-project.org/package=bigmemory.`

Ko, Y. and Kim, J. (2013). Analysis of big data using Rhipe, *Journal of the Korean Data & Information Science*, **24**, 975-987.

Kodinariya, T. M. and Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal of Advance Research in Computer Science and Management Studies*, **1**, 90-95.

Oancea, B. and Dragoescu, R. M. (2014). Integration R and Hadoop for big data analysis. *Romanian Statistical Review*, **2**. 83-94.

Park, J. H., Lee, S. Y., Kang D. H. and Won, J. H. (2013). Hadoop and MapReduce, *Journal of the Korean Data & Information Science*, **24**, 1013-1027.

Prajapati, V. (2013). *Big data analytics with R and Hadoop*, Packt Publishing Ltd, Birmingham, UK.

Sammer, E. (2012). *Hadoop Operations*, O'Reilly Media, Inc., Sebastopol, CA.

Shin, J. E., Jung, B. H. and Lim, D. H. (2015). Big data distributed processing system using RHadoop. *Journal of the Korean Data & Information Science*, **26**, 1155-1166.

White, T. (2012). *Hadoop: The definitive guide*, O'Reilly Media, Inc., Sebastopol, CA.

Journal of the Korean Data & Information Science Society
2016, **27**(3), 609–619

http://dx.doi.org/10.7465/jkdi.2016.27.3.609
한국데이터정보과학회지

RHadoop platform for K-Means clustering of big data[†]

Ji Eun Shin[1] · Yoon Sik Oh[2] · Dong Hoon Lim[3]

[13]Department of Information and Statistics, Gyeongsang National University
[2]Division of Biological Sciences, Gyeongsang NationalUniversity

Received 7 March 2016, revised 22 March 2016, accepted 23 March 2016

Abstract

RHadoop is a collection of R packages that allow users to manage and analyze data with Hadoop. In this paper, we implement K-Means algorithm based on MapReduce framework with RHadoop to make the clustering method applicable to large scale data. The main idea introduces a combiner as a function of our map output to decrease the amount of data needed to be processed by reducers. We showed that our K-Means algorithm using RHadoop with combiner was faster than regular algorithm without combiner as the size of data set increases. We also implemented Elbow method with MapReduce for finding the optimum number of clusters for K-Means clustering on large dataset. Comparison with our MapReduce implementation of Elbow method and classical kmeans() in R with small data showed similar results.

Keywords: Big data, Hadoop, K-Means clustering, R, RHadoop.

[†] This work was supported by Development Fund Foundation, Gyeongsang National University, 2015.
[1] Master of science, Department of Information Statistics, Gyeongsang National University, Jinju 660-701, Korea.
[2] Professor, Division of Biological Sciences, Gyeongsang National University, Jinju 660-701, Korea.
[3] Corresponding author: Professor and RINS, Department of Information Statistics, Gyeongsang National University, Jinju 660-701, Korea. E-mail: dhlim@gnu.ac.kr