Journal of the Korean Data & Information Science Society
2017, **28**(4), 755–768

http://dx.doi.org/10.7465/jkdi.2017.28.4.755
한국데이터정보과학회지

빅데이터 통합모형 비교분석

정병호[1] · 임동훈[2]

[1]경상남도 도청 [2]경상대학교 정보통계학과

접수 2017년 6월 13일, 수정 2017년 7월 13일, 게재확정 2017년 7월 13일

요 약

빅데이터가 4차 산업혁명의 핵심으로 자리하면서 빅데이터 기반 처리 및 분석 능력이 기업의 미래 경쟁력을 좌우할 전망이다. 빅데이터 처리 및 분석을 위한 RHadoop과 RHIPE 모형은 R과 Hadoop의 통합모형으로 지금까지 각각의 모형에 대해서는 연구가 많이 진행되어 왔으나 두 모형 간 비교 연구는 거의 이루어 지지 않았다. 본 논문에서는 대용량의 실제 데이터와 모의실험 데이터에서 다중 회귀 (multiple regression)와 로지스틱 회귀 (logistic regression) 추정을 위한 머신러닝 (machine learning) 알고리즘을 MapReduce 프로그램 구현을 통해 RHadoop과 RHIPE 간의 비교 분석하고자 한다. 구축된 분산 클러스터 (distributed cluster) 하에서 두 모형간 성능 실험 결과, RHIPE은 RHadoop에 비해 대체로 빠른 처리속도를 보인 반면에 설치, 사용면에서 어려움을 보였다.

주요용어: 빅데이터, 다중 회귀, 로지스틱 회귀, RHadoop, RHIPE.

1. 서론

오늘날 4차 산업혁명이 미래 시대를 열어갈 최대 화두이다. 4차 산업혁명은 인공지능 기반 디지털 기술이 사물과 연결되는 초연결 (hyperconnectivity) · 초지능 (superintelligence) 시대를 가리키는데 빅데이터가 4차 산업혁명의 핵심기술로 부상하고 있다. 따라서 4차 산업혁명의 성패 열쇠는 빅데이터가 갖고 있다고 할 수 있다 (Jee, 2017).

최근 글로벌 IT 시장조사기관 IDC의 보고서에 의하면 올해 세계 빅데이터 및 분석 시장의 규모는 1500억 달러 (약 172조 원)를 넘을 것으로 예상하고, 2020년이 되면 2100억 달러 (약 239조 원)의 규모가 될 것으로 내다보고 있다. 국내 빅데이터 시장 역시 세계 정세에 따라 높은 성장율을 이룰 것으로 예상되는 가운데 빅데이터 기반 분석 및 예측 능력이 금융, 유통, 통신 등 산업 전반에 걸쳐 주요 경쟁력이 될 전망이다 (IDC, 2015).

Hadoop은 대용량 데이터를 분산저장 및 처리할 수 있는 오픈 소스 프레임워크로 지금까지 널리 사용되고 있으나 기본적으로 제공되는 데이터 분석도구의 부재로 인하여 데이터 과학자들은 Hadoop 상에서 데이터 분석을 위한 별도의 도구를 필요로 하고 있다. 물론 Hadoop 기반 대용량 데이터 분석을 위한 아파치 마하웃 (Apache Mahout)과 같은 머신러닝 라이브러리 (machine learnig library)가 있으나 자바언어로 구현되어 있고 또한 군집, 분류, 협업필터링 분야에 국한되어 있어 범용성이 떨어지는 단점

[1] (51154) 경상남도 창원시 의창구 중앙대로 300, 도청, 공무원.
[2] 교신저자: (52828) 경남 진주시 진주대로 501, 경상대학교 정보통계학과, 교수 및 RINS.
E-mail: dhlim@gnu.ac.kr

이 있다. 오늘날 통계 프로그램 언어 R은 구글, 페이스북, 오라클, IBM, SAP 등의 분석엔진으로 채택할 정도로 통계 분석 및 뛰어난 그래픽 기능을 갖고 있으나 대용량 데이터 처리를 위한 스케일 확장성(scalability)을 갖고 있지 않다.

본 논문에서는 R과 Hadoop의 통합모형인 RHadoop과 RHIPE을 이용하여 대용량 데이터에서 다중 회귀 (multiple regression)와 로지스틱 회귀 (logistic regression) 추정을 위한 머신러닝 알고리즘을 Hadoop의 MapReduce 구현을 통해 두 통합모형간 비교분석하고자 한다. R과 Hadoop의 통합모형에는 RHadoop과 RHIPE 외에 여러가지 모형이 있다. 예를 들면, R Streaming 모형, RHive 모형 등이 있다. R Streaming 모형은 MapReduce 작업을 R 언어로 구현한 다음 Hadoop Streaming을 사용하여 Hadoop 명령 라인 (command line)에서 실행하는 방식으로 사용하기 간편한 반면에 클라이언트 측면에서 통합이 아니다. 그리고 RHive 모형은 R과 대용량 데이터 웨어하우스 시스템인 Hive와 연동하여 SQL 언어인 HiveQL를 사용하여 데이터 접근 및 분석을 위한 모형이다.

우리의 RHadoop 모형은 Hadoop Streaming의 래퍼 (wrapper)처럼 작동되는 R 패키지 족 (family of R packages)으로서 Park 등 (2013), Oancea 와 Dragoescu (2014), Rotte 등 (2015), Harish 등 (2015), Shin 등 (2015, 2016)에 의해 연구되었고, RHIPE 모형은 Hadoop의 구성요소인 HDFS (Hadoop Distributed File System)와 MapReduce 작업을 관리하는 하나의 방대한 R 패키지로서 Guha 등 (2012), Ko 와 Kim (2013), Lin 등 (2013), Oancea 와 Dragoescu (2014), Hafen 등 (2014), Jung 등 (2014, 2016)에 의해 연구되었다. 지금까지 두 통합모형 RHadoop과 RHIPE간 비교는 단어세기 (word count)와 같은 단순한 작업에서 비교를 제외하고는 거의 이루어 지지 않았다 (Prakash 와 Bejda, 2015).

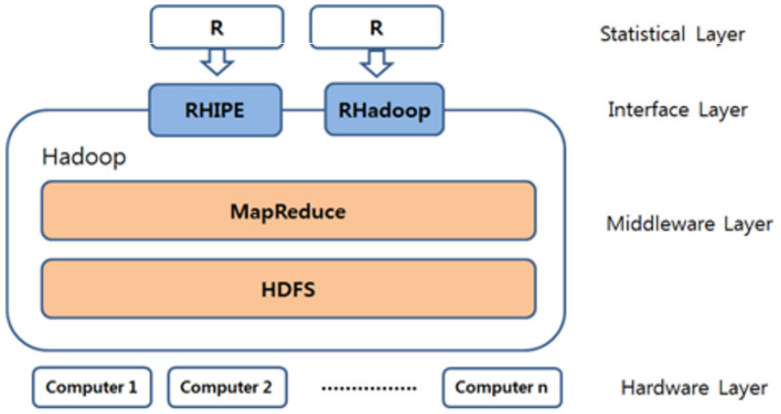

Figure 1.1 Integrated architecture of R with Hadoop

R과 Hadoop의 통합모형의 구조는 Figure 1.1과 같이 레이어 아키텍처 (layered architecture) 형태로 나타낼 수 있다 (Forte, 2015; White, 2012). Figure 1.1의 통합구조는 여러 대의 컴퓨터들로 구성되어 있는 하드웨어 레이어 (hadware layer), HDFS와 MapReduce로 이루어진 Hadoop 프레임워크의 미들웨어 레이어 (middleware layer), R의 통계적 레이어 (statistical layer), 그리고 Hadoop 프레임워크 사이에서 인터페이스 역할을 하는 인터페이스 레이어 (interface layer)로 구성되어 있다. 통계적 레이어에는 R외에 Hadoop과 연동이 가능한 SPSS, SAS와 같은 통계 프로그램이 있으나 이들 프로그램들은 유료이면서 최신 라이브러리를 제공하는데 한계가 있고 또한 Java, C, Python 등과의 연동하는데

어려움이 있다. RHadoop과 RHIPE은 R과 Hadoop사이의 인터페이스 레이어에 속하는 대표적인 통합모형이다 (Oancea 와 Dragoescu, 2014).

본 논문에서 두 통합모형 RHadoop과 RHIPE간 비교분석을 위해 먼저 다중 회귀와 로지스틱 회귀 추정을 위한 분산처리용 클러스터 (distributed cluster)를 구축하고, 실제 데이터와 모의실험 데이터에서 RHadoop과 RHIPE 모형의 확장성을 알아보기 위해 기본 R 패키지에서 회귀 분석에 대한 lm() 함수와 로지스틱 회귀 분석에 대한 glm() 함수 그리고 bigmemory 패키지에서 biglm()함수와 bigglm() 함수와의 처리속도를 비교한다. 그리고 두 모형 RHadoop과 RHIPE 간의 처리 속도를 비교하고자 한다.

본 논문의 구성은 다음과 같다. 제 2 절에서는 R과 Hadoop의 통합모형인 RHadoop과 RHIPE의 개요에 대해 살펴보고, 제 3 절에서는 다중 회귀와 로지스틱 회귀 추정 알고리즘에 대해 살펴보고자 한다. 그리고 제 4 절과 제 5 절에서는 각각 RHadoop과 RHIPE의 실험환경과 성능분석에 대해 살펴보고, 제 6 절에서 결론을 맺고자 한다.

2. R과 Hadoop의 통합모형 개요

2.1. RHadoop 모형

RHadoop 모형은 R과 Hadoop 사이 클라이언트 측면에서 통합을 제공하는 Revolution Analytics 사에 의해 개발된 오프 소스 프로젝트이다. RHadoop은 5개의 R 패키지들로 구성되어 있고 그 중에서 주요 패키지는 rmr, rhdfs, rhbase이다. rmr 패키지는 R과 MapReduce 사이의 인터페이스 역할을 하고, rhdfs 패키지는 R과 HDFS를 연결시켜주는 역할을 하고, rhbase 패키지는 R에서 HBase 데이터베이스 관리하는 역할을 한다. 이들 패키지들은 각 노드에 설치되어 네트워크 통신을 통해 분산처리를 수행한다. RHadoop 설치는 비록 다른 R 패키지들에 의존 (dependency)하지만 매우 간단하다.

다음은 RHadoop 모형에서 MapReduce 프로그램 작성에 대해 단어세기에 대한 예를 가지고 Map과 Reduce로 나누어 설명하고자 한다. 다음은 Map 프로그램을 나타내고 있다.

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

위 map 함수는 HDFS에 저장된 <키, 값> 형태의 데이터를 리스트 형태로 읽어서 처리한다. 이 때 데이터는 strsplit() 함수를 사용하여 공백을 기준으로 단어별로 쪼개어 변수에 저장한다. 그리고 각 단어별로 쪼개진 데이터는 unlist() 함수에 의해 벡터 형태로 변환되고 keyval() 함수를 이용하여 <키, 값> 형태로 HDFS에 보내진다. 여기서 <키>는 단어, <값>은 단어의 개수를 나타낸다. 다음은 Reduce 프로그램을 나타내고 있다.

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

위 reduce 함수는 map 함수에 의해 생성된 <키, 값>에 대해 단어의 개수를 알기위해 <키>를 기준으로 <값>들에 대해 합산하고 그 결과를 keyval() 함수를 통해 HDFS로 보내진다. 다음은 mapreduce

함수의 사용 형태이다.

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

위 mapreduce 함수에서 input 인자, output 인자, map 인자, reduce 인자 등 여러가지 인자를 지정한다. 여기서 input 인자는 HDFS 상의 입력 파일을 지정하고 output 인자는 HDFS 상의 출력 파일을 지정한다.

2.2. RHIPE 모형

RHIPE 모형은 원래 D&R (divide and recombine)이라는 분석기법을 통해 대용량의 복잡한 데이터를 작은 데이터 셋 (subset)으로 나누어 (divide) 분석한 다음 전체 데이터에 대해 결과를 재조합 (recombine)하는 방식으로 MapReduce 작업을 수행한다. RHIPE은 각 데이터노드 (datanode)에 대해 R, 프로토콜 버퍼 (protocol buffer)를 설치해야함으로 RHadoop에 비해 다소 설치가 어렵다. 여기서 프로토콜 버퍼는 구조화된 데이터를 직렬화 (data serialization) 할 수 있는 확장형 메커니즘이다.

다음은 단어 세기에 대한 예를 가지고 RHIPE 모형에서 MapReduce 프로그램을 Map과 Reduce로 나누어 설명하고자 한다. 다음은 Map 프로그램을 나타내고 있다.

$$\begin{aligned} map \quad = \quad & expression(\{ \\ & listOfwords \leftarrow strsplit(map.values, split = \text{" "}) \\ & words \leftarrow unlist(listOfwords) \\ & rhcollect(words, 1) \\ & \}) \end{aligned}$$

위 map 함수는 HDFS에 있는 <키, 값> 형태의 데이터를 리스트 형태로 읽어서 처리한다. 그리고 rhcollect 함수는 map 태스크의 결과를 <키, 값> 형태의 데이터로 reduce 태스크로 전송한다. 다음은 Reduce 프로그램을 나타내고 있다.

$$\begin{aligned} reduce \quad = \quad & expression(\\ & pre \leftarrow \{hap = 0\}, \\ & reduce \leftarrow \{hap \leftarrow sum(hap, unlist(reduce.values)\}, \\ & post \leftarrow \{rhcollect(reduce.key, hap)\} \\ &) \end{aligned}$$

위 reduce 함수는 3개의 인자 pre, reduce 그리고 post로 구성되어 있다. 첫번째 pre 인자는 사용된 변수의 초기화하는데 사용되고 두번재 reduce 인자는 실제 처리할 작업내용을 정의하고 세번째 post 인자는 reduce 인자에서 계산 결과를 rhcollect 함수를 통해 HDFS에 전송하는 역할을 한다. 다음은 rhwatch 함수의 사용형태이다.

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

위 rhwatch 함수는 input 인자, output 인자, map 인자, reduce 인자 등 여러가지 인자를 지정한다. map과 reduce 인자는 Map 프로그램와 Reduce 프로그램을 정의한 map 함수와 reduce 함수를 지정해주고, input과 output 인자는 HDFS 상의 입출력 파일의 경로를 지정한다.

3. 관련 알고리즘 개요

3.1. 다중 회귀추정 알고리즘

회귀분석에서 k개의 독립변수 $X_1, X_2, \ldots, X_k$와 종속변수 Y간의 관계가 선형인 경우 다음의 다중 선형 회귀모형 (multiple linear regression model)을 사용한다.

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k + \epsilon, \tag{3.1}$$

여기서 $\beta_0, \beta_1, \ldots, \beta_k$는 회귀계수 (regression coefficient)이고 오차항 ϵ은 서로 독립이고 평균 μ는 0이고 표준편차가 σ인 정규분포를 따른다고 가정한다. n개의 관찰값에 대해 식 (3.1)을 행렬로 표현하면 다음과 같다.

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

여기서

$$\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} 1 & X_{11} & X_{21} & \cdots & X_{k1} \\ 1 & X_{12} & X_{22} & \cdots & X_{k2} \\ & & \vdots & & \\ 1 & X_{1n} & X_{2n} & \cdots & X_{kn} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

이다. 따라서 $\boldsymbol{\beta}$의 최소 제곱 추정치 (least square estimate)는 식 (3.2)와 같다.

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T Y}. \tag{3.2}$$

Hadoop에서 HDFS는 데이터 $\boldsymbol{Y}$와 $\boldsymbol{X}$가 L개의 블럭으로 분할되어 저장된다. 따라서 $\boldsymbol{X} = [X_1^T, X_2^T, \ldots, X_L^T]^T$라 하면 식(3.2)에서 $\boldsymbol{X^T X}$와 $\boldsymbol{X^T Y}$는 다음과 같이 L개의 블럭 각각에 대해 곱셈 연산을 수행한 후 그 결과를 합산하여 계산한다.

$$\boldsymbol{X^T X} = \sum_{j=1}^{L} X_j^T X_j, \boldsymbol{X^T Y} = \sum_{j=1}^{L} X_j^T Y_j.$$

3.2. 로지스틱 회귀추정 알고리즘

로지스틱 회귀모형에서 독립변수 $\boldsymbol{X} = (1, X_1, X_2, ..., X_k)^T$에 대해 종속변수 Y가 취하는 값 $\{-1, +1\}$ 중 $Y = +1$일 확률은 다음과 같이 표시할 수 있다.

$$P(Y = +1|\boldsymbol{X}) = g(\boldsymbol{X}^T\boldsymbol{\beta}), \tag{3.3}$$

여기서 $g(Z) = 1/(1 + exp(-Z))$이다.

데이터 셋 $(Y_i, \boldsymbol{X}_i)$, $i = 1, ..., n$,에 대하여 최대 우도 추정법 (maximum likelihood estimation)에 의해 로지스틱 회귀계수 (logistic regression coefficient) $\boldsymbol{\beta}$ 추정을 위한 우도함수 (likelihood function)는 다음과 같다.

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} P(Y_i|\boldsymbol{X}_i). \tag{3.4}$$

따라서 로그-우도함수 (log-likelihood function)는 식 (3.4)에 로그를 취하고 식 (3.3)를 사용하여 다음과 같이 나타낼 수 있다.

$$\begin{aligned} \ell(\boldsymbol{\beta}) &= logL(\boldsymbol{\beta}) \\ &= \sum_{i=1}^{n} log\ g(Y_i Z_i), \end{aligned} \tag{3.5}$$

여기서 $Z_i = \boldsymbol{X}_i{}^T\boldsymbol{\beta}$이다.

Newton-Raphson 알고리즘은 식 (3.5)에 주어진 로그-우도함수에서 다음의 β_k에 대한 1차 기울기 (gradient)와 β_j에 대한 2차 기울기로 부터 얻어진다.

$$\begin{aligned} \frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial\beta_j\partial\beta_k} &= \frac{\partial}{\partial\beta_j}\sum_{i=1}^{n} Y_i X_{ik} g(-Y_i Z_i) \\ &= -\sum_{i=1}^{n} X_{ij} X_{ik} g(Y_i Z_i) g(-Y_i Z_i). \end{aligned} \tag{3.6}$$

H를 Hessian 행렬, 즉, 로그-우도 함수의 2차 기울기 행렬이라 할 때 식 (3.6)은 H의 (j,k)번째 원소 즉, H_{jk}를 나타낸다. 따라서 최적해는 H를 사용하여 다음과 같이 반복을 통해 얻어진다.

$$\beta_k := \beta_k - H^{-1}\sum_{i=1}^{n} Y_i X_{ik} g(-Y_i Z_i).$$

4. RHadoop과 RHIPE의 실험 환경

4.1. 클러스터 환경 구축

Figure 4.1은 본 연구를 위해 구축한 리눅스 기반 분산처리용 클러스터의 구성도를 나타낸다. 이 클러스터는 6대의 컴퓨터를 사용하여 구축하였고 그 중 1대 컴퓨터를 마스터 노드(master node), 나머지 5대의 컴퓨터를 슬레이브 노드(slave node)로 설정하였다.

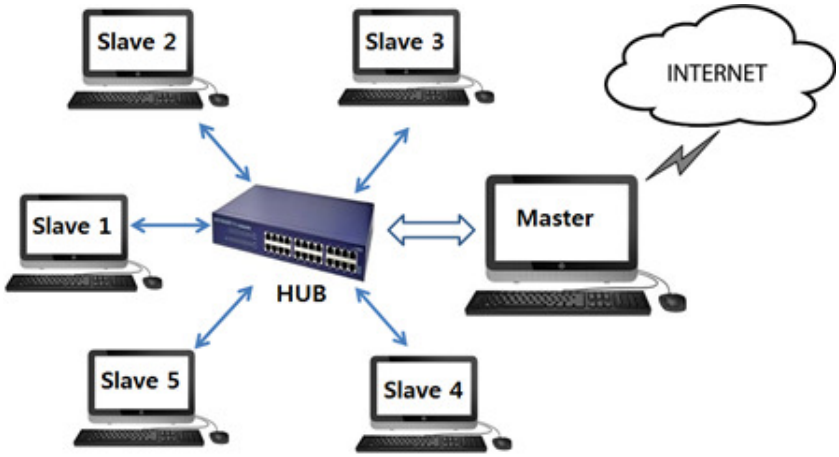

Figure 4.1 Distributed cluster architecture

Table 4.1은 구축된 클러스터의 하드웨어 및 소프트웨어 사양을 나타내고 있다.

RHIPE은 RHadoop와는 다르게 각 데이터 노드에 R을 공유 라이브러리 (shared library)로 설치하고, 효율적인 데이터 전송을 위해 별도의 프로토콜 버퍼를 설치한다.

본 논문에서 시스템 처리시간은 R의 system.time() 함수에서 제공하는 elapsed time을 가지고 측정하였다. 여기서 elapsed time은 프로세스의 처리시간을 나타내는데 CPU 시간을 의미한다.

Table 4.1 Cluster hardware and software requirements

Hardware requirements	CPU		Intel(R)Core(TM)2 Duo CPU E8300@2.83GHz
	RAM	master	4 GB
		slaves	2 GB
	NIC		RealTek
	Switch Hub		Catalyst 2960 Switches with 1 Gigabit
Software requirements	Operating System		12.04 LTS
	Java		1.7.0
	Hadoop		0.20.2
	R		3.1.0
	RHIPE		0.73.1
	Google Protocal Buffer		2.4.1

4.2. 다중 회귀를 위한 실험 데이터

4.2.1. 실제 데이터

실제 데이터는 2009년 ASA (American Standards Association:미국 표준 협회)에서 공개된 미국 항공편 운항과 관련된 데이터로서 1987년부터 2008년까지 29개의 변수에 대해 얻어졌다 (ASA data expo, 2009). 이 항공 데이터는 123,534,970개의 행으로 구성되어 있고 12 GB에 해당되는 데이터의 크기를 갖고 있다. 실험에 사용된 데이터는 결측값과 회귀분석하는데 부적합한 변수를 제외하여 얻었고 Table 4.2에서 보는 것처럼 7개 변수(여기서 종속변수는 도착 지연시간 (ArrDelay))에 대해 2.33 GB에 해당되는 데이터 크기를 가지고 실험하였다.

Table 4.2 Variables used in real data

Variables	Description
Month	1-12
DayOfMonth	1-31
DayOfWeek	1(Monday) - 7(Sunday)
ActualElapsedTime	in minutes
DepDelay	departure delay, in minutes
Distance	in miles
ArrDelay	arrival delay, in minutes

실험에 사용될 작은 데이터는 원래 데이터 크기 2.33 GB를 기준으로 샘플링을 이용하여 얻었고, 큰 데이터는 원래 데이터로부터 2 ∼ 4배수 복원추출에 의해 데이터 크기 9.32 GB까지 생성하였다.

4.2.2. 모의실험 데이터

실험에 사용된 모의실험 데이터는 식 (3.1)에 주어진 다중회귀 모형으로부터 정규난수 (normal random number)를 발생하여 얻는다. 본 실험에서 $k = 7$인 경우 독립변수 $X_1, ..., X_7$와 종속변수 Y는 다음과 같이 얻어진다.

$$X_j \sim N(10, 100^2), j = 1, \ldots, 7,$$
$$Y \sim N(10, 100^2).$$

실험에 사용된 모의실험 데이터는 실제 항공기 운항 데이터가 정수형 데이터이므로 생성된 실수형 데이터에서 정수 부분만을 취하여 실험에 사용하였다.

4.3. 로지스틱 회귀를 위한 실험 데이터

4.3.1. 실제 데이터

로지스틱 회귀추정에 사용된 실제 데이터는 다중 회귀 분석에서 사용된 똑같은 미국 항공기 운항 데이터를 사용하고자 한다. Wang 등 (2015)은 로지스틱 회귀분석을 하는데 원래 29개의 변수 중 5개의 변수만 사용하였는데 본 연구에서도 똑같은 변수를 사용하였고 종속변수로 사용된 도착지연시간 (ArrDelay)은 다음과 같이 얻었다.

$$\text{ArrDelay} = \begin{cases} 1, & \text{if ArrDelay} > 15, \\ 0, & \text{otherwise.} \end{cases}$$

그리고 4개의 독립변수 중 출발시간 (DepHour)은 원래 데이터의 출발시간 (DepTime)에서 시 (Hour)부분만을 사용하였고, 비행거리 (Distance)는 원래 데이터와 같고 나머지 두개의 변수인 밤 (Night)과 주말 (Weekend)은 다음과 같이 얻었다.

$$\text{Night} = \begin{cases} 1, & \text{if } 0 \le \text{DepHour} \le 5 \text{ or } 20 \le \text{DepHour} \le 24, \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{Weekend} = \begin{cases} 1, & \text{if } 6 \le \text{DayOfWeek} \le 7, \\ 0, & \text{otherwise,} \end{cases}$$

여기서 DayOfWeek는 1 (월요일), 2 (화요일), ..., 7 (일요일) 값을 갖는다.

4.3.2. 모의실험 데이터

로지스틱 회귀모형에서 $k = 4$인 독립변수 $X_1, ..., X_4$는 다음과 같이 표준정규난수로 부터 얻어진다.

$$X_j \sim N(0, 1), j = 1, \ldots, 4.$$

그리고 모형에서 회귀계수 $\beta_0, \beta_1, ..., \beta_4$는 다음과 같다.

$$\beta_0 = 0.70, \beta_1 = 1.00, \beta_2 = 1.30, \beta_3 = 0.25, \beta_4 = 0.05,$$

여기서 β_j, $j = 1, \ldots, 4$들은 Rashid (2008)에서 사용한 회귀계수들이다. 따라서 모의실험에서 사용된 로지스틱 회귀 모형은 다음과 같다.

$$g(Z) = \frac{1}{1 + e^{-Z}},$$

여기서

$$Z = 0.70 + 1.00X_1 + 1.30X_2 + 0.25X_3 + 0.05X_4,$$

$$Y \sim B(1, g(Z)).$$

즉, Y는 이항분포 $B(1, g(Z))$로부터 0과 1인 값을 갖는다.

5. RHadoop과 RHIPE의 성능 분석

5.1. 다중 회귀에서 성능비교

우리는 실제 데이터와 모의실험 데이터에서 다중 회귀 추정을 위한 RHadoop과 RHIPE 모형의 확장성을 평가하기 위해 기존 패키지의 lm() 함수, biglm() 함수와 비교하고, 그리고 두 모형 RHadoop과 RHIPE간의 성능을 비교하였다.

5.1.1. 실제 데이터

Table 5.1는 실제 데이터에서 다중 회귀추정을 위한 모형들의 처리 속도를 나타낸다.

Table 5.1 Running times of models for multiple regression estimation with actual data

Data size	Packages		Integrated models	
(no. of rows)	lm	biglm	RHadoop	RHIPE
100 MB (5,181,150)	58.143	23.491	60.203	61.044
200 MB (10,362,150)	FAIL	47.709	70.258	61.064
300 MB (15,543,450)	-	77.423	71.881	66.953
500 MB (25,905,751)	-	143.700	82.071	71.035
1.00 GB (53,052,912)	-	284.849	156.051	134.470
2.33 GB (123,534,970)	-	708.388	282.808	264.859
4.66 GB (247,069,940)	-	6779.768	531.151	468.183
9.32 GB (494,139,880)	-	FAIL	1003.031	874.074

Table 5.1로부터 lm() 함수는 패키지의 구조적인 문제로 인하여 200 MB에서 처리가 불가능하였고 biglm() 함수는 9.32 GB에서 처리가 불가능하였다. RHadoop과 RHIPE는 모든 데이터 크기에서 처리가 가능하였다. 그리고 제안한 통합모형과 biglm() 함수와의 비교에서 제안한 모형은 데이터가 크더라도 완만하게 감소하는 처리 속도를 보인 반면 biglm() 함수는 급격하게 처리 속도가 늦어지는 것을 알 수 있다. 특히, 4.66 GB에서는 제안한 통합모형보다 최소 12배 이상 처리 속도 차이가 나는 것을 알 수 있다. RHadoop과 RHIPE 비교에서는 예상하였듯이 100 MB에서의 처리속도를 제외하고 모든 데이터 크기에서 RHIPE이 RHadoop보다 빠른 처리속도를 보였다. 그 이유는 RHadoop은 Hadoop Streaming 방식으로 데이터를 처리하는 반면에 RHIPE는 Hadoop Streaming에 의존하지 않고 데이터 직렬화와 같은 별도의 처리 방식으로 인해 효율적인 작업부하 (workload)가 이루어 지기 때문이다.

5.1.2. 모의실험 데이터

Table 5.2는 모의실험 데이터에서 다중 회귀추정을 위한 모형들의 처리 속도를 나타낸다.

Table 5.2로부터 lm() 함수는 300 MB에서 처리가 불가능하였고 biglm() 함수는 9.32 GB에서 처리가 불가능하였다. RHadoop과 RHIPE 비교에서는 Table 5.1의 결과처럼 전반적으로 RHIPE이 RHadoop보다 빠른 처리속도를 보이나, 두 모형간 처리 속도 차이는 Table 5.1에 비해 줄어듦을 알 수 있다. 이것은 똑같은 데이터 크기라 할지라도 실제 데이터와 모의실험 데이터에서 차지하는 행의 수 (no.of rows)가 다르기 때문이다. 예를 들면, 500 MB 데이터 크기에서 실제 데이터의 행의 수는 25,905,751인데 비해 모의실험 데이터의 행의 수는 20,260,625로 실제 데이터의 행의 수에 비해 작아 상대적으로 모의실험 데이터에서 처리속도가 빠르고, 이로인해 두 통합모형간 처리 속도차이도 줄어둔 것으로 생각된다.

5.2. 로지스틱 회귀에서 성능비교

우리는 실제 데이터와 모의실험 데이터에서 로지스틱 회귀 추정을 위한 RHadoop과 RHIPE 통합모

Table 5.2 Running times of models for multiple regression estimation with simulation data

Data size	Packages		Integrated models	
(no. of rows)	lm	biglm	RHadoop	RHIPE
100 MB (4,052,125)	51.197	19.148	47.824	45.895
200 MB (8,104,250)	190.808	36.752	52.086	45.904
300 MB (12,156,375)	FAIL	57.486	50.999	45.942
500 MB (20,260,625)	-	99.551	59.178	55.617
1.00 GB (41,492,145)	-	207.550	100.397	101.792
2.33 GB (96,615,449)	-	485.694	188.880	183.150
4.66 GB (193,230,989)	-	1266.618	359.516	350.680
9.32 GB (386,461,796)	-	FAIL	687.521	652.731

형의 확장성을 평가하기 위해 기존의 패키지의 glm() 함수, bigglm() 함수와 비교하고, 그리고 두 모형 RHadoop과 RHIPE간의 성능을 비교하였다.

5.2.1. 실제 데이터

Table 5.3는 실제 데이터에서 로지스틱 회귀추정을 위한 모형들의 처리 속도를 나타낸다.

Table 5.3 Running times of models for logistic multiple regression estimation with actual data

Data size	Packages		Integrated models	
(no. of rows)	glm	bigglm	RHadoop	RHIPE
50 MB (4,056,491)	59.049	59.149	253.771	368.379
100 MB (8,112,982)	FAIL	113.331	353.127	368.101
200 MB (16,225,963)	-	237.454	391.920	418.612
300 MB (24,338,945)	-	370.617	391.494	428.720
500 MB (40,564,909)	-	710.063	442.189	489.254
1.00 GB (83,073,697)	-	FAIL	788.36 2	914.224
1.45 GB (120,748,239)	-	-	1102.506	1268.597
2.90 GB (241,496,479)	-	-	1796.517	2118.884
5.80 GB (482,992,956)	-	-	3469.744	4051.353
11.60 GB (965,985,912)	-	-	6551.920	7673.448

Table 5.3으로부터 glm() 함수는 100 MB에서 처리가 불가능하였고 bigglm() 함수는 1.00 GB에서

처리가 불가능하였다. RHadoop과 RHIPE 비교에서는 RHIPE이 RHadoop보다 빠른 처리 속도를 보일 것이라는 일반적인 예상과 달리 모든 데이터에서 RHadoop이 RHIPE보다 처리 속도가 빠르게 나타났다. 이것은 Table 5.3에서 보는 것처럼 데이터의 크기에 비해 행의 수가 많은 데이터의 특성으로 인해 RHIPE의 프로토콜 버퍼를 통해 변환 및 전송되는데 시간이 소요되기 때문으로 사료된다.

5.2.2. 모의실험 데이터

Table 5.4는 모의실험 데이터에서 로지스틱 회귀추정을 위한 모형들의 처리 속도를 나타낸다.

Table 5.4 Running times of models for logistic multiple regression estimation with simulation data

Data size	Packages		Integrated models	
(no. of rows)	glm	bigglm	RHadoop	RHIPE
50 MB (702,404)	35.375	34.210	370.786	390.176
100 MB (1,404,808)	80.951	79.349	467.189	450.609
200 MB (2,809,616)	179.520	189.758	476.338	400.532
300 MB (4,214,423)	299.717	255.886	503.046	371.197
500 MB (7,024,039)	FAIL	410.107	548.559	452.076
1.00 GB (14,384,671)	-	FAIL	934.792	786.240
1.45 GB (20,908,227)	-	-	1257.910	1120.610
2.90 GB (41,816,454)	-	-	1991.982	1687.665
5.80 GB (83,632,816)	-	-	3793.391	3247.046
11.60 GB (167,265,816)	-	-	7101.415	6221.431

Table 5.4로부터 glm() 함수는 500 MB에서 처리가 불가능하였고 bigglm() 함수는 1.00 GB에서 처리가 불가능하였다. RHadoop과 RHIPE 간의 비교에서는 50 MB 데이터의 크기를 제외하고 RHIPE이 RHadoop보다 빠른 처리속도를 보였다.

6. 결론 및 향후연구

빅데이터가 제4차 산업혁명을 이끌 핵심 아이콘으로 부각되면서 빠른 시간 내에 데이터 수집, 처리, 분석할 수 있는 빅데이터 플랫폼이 요구되고 있다.

Hadoop은 빅데이터의 분산 저장 및 처리 능력을 갖고 있지만 R과 같은 데이터 분석 능력은 갖고 있지 않고 R은 데이터 분석 및 데이터 가시화와 같은 데이터 분석, 조작 능력은 갖고 있지만 데이터 처리를 위한 확장성 (scalability)을 갖고 있지 않다. RHadoop와 RHIPE은 Hadoop과 R의 장점만을 통합한 모형으로 각각의 모형에 대해서는 연구가 많이 진행되었으나 이들 비교 연구는 거의 이루어지지 않았다.

본 논문에서는 두 통합모형 RHadoop와 RHIPE의 비교분석을 위해 분산처리 기반의 클러스터를 구축하고 다중 회귀와 로지스틱 회귀 추정 알고리즘을 MapReduce 프로그램으로 구현하였다. 특히, 로지스틱 회귀추정에서 Gradient Descent 알고리즘을 MapReduce로 구현한 연구는 존재하나 본 연구

에서는 Newton-Raphson 알고리즘을 MapReduce로 처음으로 구현하여 실제 데이터와 모의실험 데이터에서 RHadoop과 RHIPE 모형의 확장성이 용이함을 보였다. 이를 위해 기본 R 패키지에서 다중회귀 분석에 대한 lm() 함수와 로지스틱 회귀분석에 대한 glm() 함수 그리고 bigmemory 패키지에서 biglm()함수, bigglm() 함수와 처리 속도를 비교하였다. 그리고 RHadoop과 RHIPE 모형 간의 처리 성능을 비교하였다.

두 통합모형 성능실험 결과, RHadoop과 RHIPE 모형은 기존의 패키지들에 비해 데이터 크기에 관계 없이 데이터 처리가 가능함을 보였고 RHadoop과 RHIPE 비교에서 RHIPE는 설치와 사용면에서 어려운 점을 갖고 있으나 실제 데이터에서 로지스틱 회귀 추정을 제외하고 RHadoop보다 전반적으로 빠른 처리속도를 보였다.

지금까지 살펴보았듯이, Hadoop기반 플랫폼은 기존의 패키지에 비해 방대한 데이터를 처리할 수 있으나 MapReduce의 일괄처리작업으로 인하여 머신러닝과 같은 반복적인 데이터 처리에는 성능이 떨어지고, 또한 Hadoop의 HDFS는 데이터 노드간의 정보 전달 및 저장하는데 입출력과 네트워크 오버헤드가 발생할 수 있다. 이러한 문제를 해결하기 위해 인메모리 기반인 Spark와 H2O 플랫폼에서 머신러닝에 관한 많은 연구가 진행되고 있고, 속도면에서 기존의 Hadoop기반 플랫폼보다 전반적으로 빠른 처리속도를 보이고 있다.

본 논문에서는 Spark의 MLLib에 수록된 알고리즘이 제한적이고 본 논문에 사용된 알고리즘과 달라 직접 비교할 수 없었으나 향후 이에 대한 비교 연구가 필요하다고 사료된다.

References

ASA data expo. (2009). `http://stat-computing.org/dataexpo/2009/the-data.html`

Davenport, T. (2015). *B. I. G. forum 2015*, Gyeonggi Creative Economy & Innovation Center.

Forte, R. M. (2015). *Mastering predictive analytics with R*, Packt Publishing Ltd, Birmingham, U.K.

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

Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B., Cleveland, W. S. (2012). Large complex data: divide and recombine (D&R) with RHIPE. *Statistics*, **191**, 53-67.

Hafen, R., Gibson, T., Dam, K. K., Critchlow., T. (2014). *Power grid data analysis with R and Hadoop in data mining applications with R*, 1-34.

Harish, D., Anusha, M.S., Dr. Daya Sagar, K.V. (2015). Big data analysis using Rhadoop. *IJIRAE*, **4**, 180-185.

Hilbe, J. M. (2009). *Logistic regression models*, Chapman & Hall/CRC Press.

IDC. (2015). *IDC FutureScape: Worldwide big data and analytics 2016 predictions*, MA, USA.

Jee, Y. S. (2017). Exercise rehabilitation in the fourth industrial revolution. *Journal of Exercise Rehabilitation*, **13**, 255-256.

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.

Jung, B. H. and Lim, D. H. (2016). Learning algorithms for big data logistic regression on RHIPE platform. *The Korean Journal of Applied Statistics*, **27**, 911-923.

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

Liang, S. (2003). *Quantitative remote sensing of land surfaces*, John Wiley & Sons.

Lin, H., Yang, S., Midkiff, S. P. (2013). RABID - A general distributed R processing framework targeting large data-set problems. *IEEE International Congress on Big Data*, Santa Clara, CA, USA.

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., Won, J. H. (2013). Hadoop and Mapreduce. *Journal of the Korean Data & Information Science*, **24**, 1013-1027.

Prakash, L. and Bejda, M. (2015). *Performance analysis for scaling up R computations using Hadoop*, B.S. in Computer Science, The University of Texas at Austin.

Prajapati, V. (2013). *Big data analytics with R and Hadoop*, Packt Publishing Ltd, Birmingham, UK.
Rashid, M. (2008). *Inference on logistic regression*, Ph. D. Thesis, Bowling Green State University.
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.
Shin, J. E., Oh, Y. S. and Lim, D. H. (2016). RHadoop platform for K-Means clustering of big data. *Journal of the Korean Data & Information Science*, **27**, 609-619.
Wang, C., Chen, M. H., Schifano, Wu, J. and Yan, J. (2015). *A survey of statistical methods and computing for Big Data*, Cornell University Library.
White, T. (2012). *Hadoop: The definitive guide*, O'Reilly Media, Inc., Sebastopol, CA.
Rotte, A. V., Patwari, G., Hiremath, S. (2015). Big data analytics made easy with rhadoop. *International Journal of Research in Engineering and Technology*, **4**, 9-15.

Journal of the Korean Data & Information Science Society
2017, **28**(4), 755–768

http://dx.doi.org/10.7465/jkdi.2017.28.4.755
한국데이터정보과학회지

Comparison analysis of big data integration models

Byung Ho Jung[1] · Dong Hoon Lim[2]

[1] Gyeongsangnamdo Provincial Government [2] Department of Information and Statistics, Gyeongsang National University

Received 13 June 2017, revised 13 July 2017, accepted 13 July 2017

Abstract

As Big Data becomes the core of the fourth industrial revolution, big data-based processing and analysis capabilities are expected to influence the company's future competitiveness. Comparative studies of RHadoop and RHIPE that integrate R and Hadoop environment, have not been discussed by many researchers although RHadoop and RHIPE have been discussed separately. In this paper, we constructed big data platforms such as RHadoop and RHIPE applicable to large scale data and implemented the machine learning algorithms such as multiple regression and logistic regression based on MapReduce framework. We conducted a study on performance and scalability with those implementations for various sample sizes of actual data and simulated data. The experiments demonstrated that our RHadoop and RHIPE can scale well and efficiently process large data sets on commodity hardware. We showed RHIPE is faster than RHadoop in almost all the data generally.

Keywords: Big data, multiple regression, logistic regression, RHadoop, RHIPE.

[1] Official, Gyeongsangnamdo Provincial Government, Gyeongsangnamdo, 51154, Korea.
[2] Corresponding author: Professor and RINS, Department of Information Statistics, Gyeongsang National University, 52828, Korea. E-mail: dhlim@gnu.ac.kr