The Korean Journal of Applied Statistics (2014) DOI: http://dx.doi.org/10.5351/KJAS.2014.27.7.1171
27(7), 1171–1185

Rhipe Platform for Big Data Processing and Analysis

Byung Ho Jung[a] · Ji Eun Shin[a] · Dong Hoon Lim[a,1]

[a]Department of Information Statistics, Gyeongsang National University

(Received September 30, 2014; Revised December 22, 2014; Accepted December 23, 2014)

Abstract

Rhipe that integrates R and Hadoop environment, made it possible to process and analyze massive amounts of data using a distributed processing environment.

In this paper, we implemented multiple regression analysis using Rhipe with various data sizes of actual data and simulated data. Experimental results for comparing the computing speeds of pseudo-distributed and fully-distributed modes for configuring Hadoop cluster, showed fully-distributed mode was more fast than pseudo-distributed mode and computing speeds of fully-distributed mode were faster as the number of data nodes increases. We also compared the performance of our Rhipe with stats and biglm packages available on bigmemory. The results showed that our Rhipe was more fast than other packages owing to paralleling processing with increasing the number of map tasks as the size of data increases.

Keywords: Big data, R, Hadoop, Rhipe, multiple regression analysis.

1. 서론

2001년 Doug Laney는 방대한 규모(volume), 빠른 속도(velocity), 다양한 형태(variety)를 가진 데이터를 빅데이터라고 정의하였고(Laney, 2001), 위키피디아에서는 “기존 데이터베이스 관리 도구로서 데이터 수집, 저장, 관리, 분석 영역을 넘어선 대량의 정형 또는 비정형 데이터 집합”을 빅데이터라고 정의하였다(Manyika 등, 2011). 구글의 CEO였던 에릭 슈미트(Eric Schmidt)는 2010년 테크노미 컨퍼런스(technomy conference)에서 인류 문명 이래 2003년까지 5 엑사바이트(exa byte)의 데이터를 만들어냈지만, 지금은 2일 마다 같은 양의 데이터가 생산되고 있다고 언급하였다. 오늘날, IBM, HP, SAP, MS, EMC, 오라클 같은 대형 IT 벤더 회사들은 대용량 데이터 시대를 맞이하여 빅데이터 솔루션을 개발하고 있다.

빅데이터 시대가 도래하면서 새로운 처리/분석 패러다임이 요구되는바 미국중심으로 통계엔진인 R이 기업용 분석 플랫폼으로 확산되고 있고, 전통적인 기업용 통계분석 패키지인 SAS와 SPSS를 위협하고 있다. R은 이미 구글, 페이스북, 야후, 아마존 등 닷컴 기업의 분석 플랫폼으로 사용 중이다. 특히, 오라클, IBM, 테라데이터 등 빅데이터의 고성능 분석을 추구하는 IT업체들은 빠른 분석을 위해 메모리나 데이터베이스에서 직접 분석을 실시하는 인-메모리 분석(in-memory analytics) 혹은 인-데

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No.2011-0010089).
[1]Professor, Department of Information Statistics and RINS, Gyeongsang National University, Korea. E-mail: dhlim@gnu.ac.kr

이터베이스 분석(in-database analytics)을 할 수 있는 고성능 컴퓨팅(high performance computing; HPC) 시스템에 R을 기본분석 플랫폼으로 채택하고 있다. R은 위와 같이 많은 활용에도 불구하고 확장성(scalability)이 떨어지는 단점을 갖고 있다(Prajapti, 2013). 따라서, R의 기본패키지로는 제한된 데이터 규모에서 만이 처리되고 작동된다.

R의 사용자들은 대용량 데이터 처리를 위해 여러 가지 방법을 제시하였는데 그 중에서 ff 패키지(Adler 등, 2012)와 bigmemory 패키지(Kane와 Emerson, 2010a, 2010b) 사용은 데이터 전체를 메모리에 로딩하지 않고 데이터 구조만을 메모리에 로딩하여 사용하는 방식으로 처리 속도가 늦고 또한 물리적 메모리 확장의 한계로 인해 대용량 데이터 처리에 한계를 갖고 있다.

Hadoop은 대용량 데이터를 분산처리 할 수 있는 오픈 소스 플랫폼이다(White, 2012; Sammer, 2012). Rhipe은 R과 Hadoop의 통합 환경을 제공하는 대표적인 패키지이다 (Guha, 2010). Rhipe은 미국의 퍼듀 대학교의 Saptarhis Guha에 의해서 처음 개발되었고 데이터 분석도구인 R과 대용량 처리 시스템인 Hadoop과 연동하여 빅데이터 처리/분석을 수행할 수 있는 프로그램이다. Rhipe 관련 주요 연구로는 Lin 등 (2012)은 빅데이터 분석을 위해 D&R(divide and recombine) 방법을 Rhipe 환경에 적용하였고 Prajapati (2013)은 Rhipe을 이용하여 단어 개수(word count) 문제를 다루었고 Hafen 등 (2014)은 PMU(phasor measurement unit) 데이터에서 Rhipe을 이용하여 요약통계량과 빈도분석을 수행하였다. 국내에서는 주로 고영준과 김진석 (2013)에 의해 연구가 이루어져왔다. 고영준과 김진석 (2013)은 모의실험 데이터에서 Rhipe을 이용한 회귀분석을 수행하였다.

본 논문에서는 Rhipe 플랫폼을 이용한 회귀분석을 통해 고영준과 김진석 (2013), Prajapati (2013), Hafen 등 (2014)에서 다루지 못한 Hadoop의 가상분산 모드(pseudo-dstributed mode)와 완전분산 모드(fully-distributed mode)에서 데이터 노드의 개수 증가에 따른 처리 속도를 비교 분석하고자 한다. 또한, 제안된 Rhipe 플랫폼의 성능을 평가하기 위해 기본 R 패키지인 stats와 bigmemory 패키지 상에서 유용한 biglm 패키지와 처리 속도를 비교하고자 한다.

본 논문은 다음과 같이 구성되어 있다. 제 2 절에서는 Hadoop의 주요 시스템인 HDFS와 MapReduce에 대해 간략하게 살펴보고 제 3 절에서는 R과 Hadoop을 연동해주는 Rhipe 패키지에 대해 살펴보고자 한다. 제 4 절에서는 실제 데이터와 모의실험 데이터에서 회귀분석 구현을 통해 데이터 노드의 증가에 따른 Rhipe 플랫폼의 성능을 평가하고 기존의 패키지들과 비교분석한다. 그리고 제 5 절에서 결론 및 향후연구에 대해 논의한다.

2. Hadoop 개요

Hadoop은 대용량 데이터를 처리할 수 있는 소프트웨어로 다수의 컴퓨터를 연결한 클러스터에서 분산 응용 프로그램을 지원하는 자바 기반의 오픈소스 프레임워크이다. Hadoop의 핵심기술은 Figure 2.1과 같이 HDFS(Hadoop Distributed File System)와 MapReduce로 구성되어 있다.

HDFS는 물리적으로 네임 노드(name node)와 데이터 노드(data node)로 구성되어 있고 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. 사용자로부터 요청받은 MapReduce 프로그램은 잡(job)이라는 하나의 작업 단위로 관리되며 잡 트래커는 하둡 클러스터내의 테스크 트랙커들로 나눠서 잡을 실행한다. MapReduce의 병렬처리는 Figure 2.2와 같이 함수의 입력과 출력이 모두 키(key)와 값(value) 쌍으로 구성되어 있다.

Figure 2.2에서 보면 입력데이터는 블럭 단위로 분할(split)되고 각 블럭은 <키, 값> 쌍으로 표현된다.

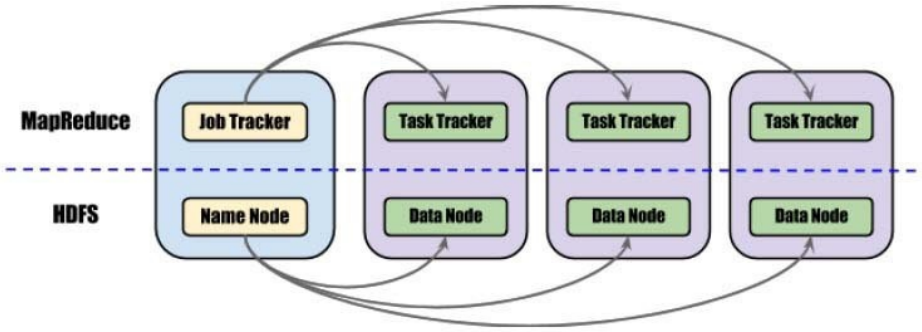

Figure 2.1. Basic architecture of the Hadoop cluster

분할된 각 블럭에 대해 map 함수 수행 후 다음 단계의 작업을 위해 <키, 값> 쌍으로 임시 저장된다. Shuffle / Sort 단계에서는 map 함수의 결과를 키에 따라 섞고 다시 정렬한다. 각 키에 따라 값 들에 대해 병렬로 reduce 함수를 적용한 후 <키, 값> 쌍으로 출력한다.

Hadoop은 설치방식에 따라 독립실행 모드(standalone mode), 가상분산 모드, 완전분산 모드로 구분한다. 독립실행 모드는 Hadoop의 기본 모드로 다른 노드와 통신할 필요가 없다. 따라서 이 모드의 목적은 독립적으로 MapReduce 프로그램을 개발하고 디버깅하는 용도로 적합한 모드이다. 가상분산 모드는 Hadoop 데몬 프로세스가 하나의 로컬 컴퓨터 상에서 동작하는 방식이고 완전분산 모드는 Hadoop 데몬 프로세스가 여러 대의 컴퓨터로 구성된 클러스터 상에서 동작하는 방식으로 분산 저장과 분산 연산의 장점을 갖고 있다.

3. Rhipe 개요

Rhipe은 R과 Hadoop을 연동하여, R상에서 Map/Reduce기법을 적용하여 대용량의 데이터에 대해 통계적 분석이 가능한 R 패키지이다. Rhipe은 데이터 직렬화(data serialization)를 위해 구글에서 제공하는 프로토콜 버퍼(protocol buffer)를 사용한다. 프로토콜 버퍼는 언어 및 플랫폼에서 중립적이고, 구조적 데이터로의 확장을 가능하게 함으로 R에서 제공하는 데이터 타입을 Java, C, python등의 다른 개발언어에서 사용이 가능하게 하고, 다른 언어로 저장된 데이터 형태를 R로 읽어 들이는 역할을 수행한다.

Table 3.1은 본 논문에서 사용 중인 Rhipe의 최신버전 0.73에서 제공되는 HDFS와 MapReduce 관련 함수들이다.

3.1. Map 함수 설정

map 함수는 HDFS상의 데이터를 가공하여 새로운 <키, 값> 쌍의 집합을 출력한다. 아래 코드는 map 스크립트를 나타내고 있다.

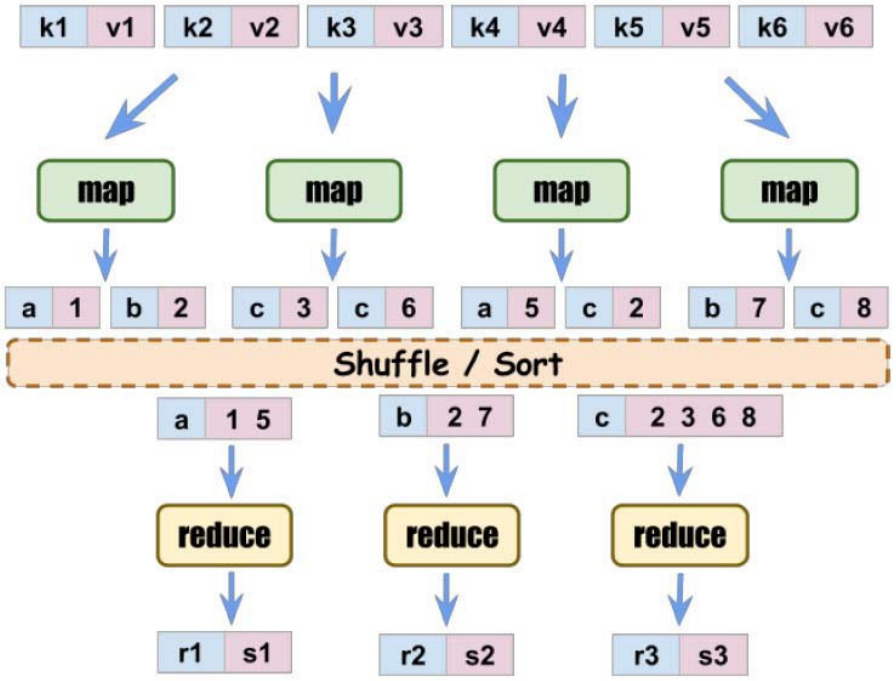

Figure 2.2. Mapreduce parallel processing

map 스크립트

```
map = expression({    do.call“rbind”,lapply(strsplit(unlist(map.values),“ ”))
                      ...
                      rhcollect(key,values)
                      })
```

위 스크립트에서 map 함수는 HDFS에서 불러온 데이터를 리스트 형태인 map.values로 읽어 들인 후 벡터로 변환하여 전처리하거나, 다른 값으로 변환한다. 그리고 rhcollect() 함수를 이용하여 <키, 값> 쌍의 집합을 생성한 후 Hadoop으로 전송한다.

3.2. Reduce 함수 설정

Hadoop은 자동적으로 map 함수를 통해 계산된 결과 값을 키를 기반으로 정렬하여 값들을 모은 후 reduce task로 전송이 이루어진다. reduce 함수는 집계연산을 수행하여 또 다른 <키, 값> 쌍의 집합을 생성한 후 이를 Hadoop에 저장한다. 아래 코드는 reduce 스크립트를 나타내고 있다.

Table 3.1. Various functions for HDFS and MapReduce operations

함수	설명
rhinit()	Rhipe를 초기화한다.
rhcollect(key,value)	<키, 값> 쌍을 생성한다.
rhwatch()	map 함수와 reduce 함수를 R 오브젝트에 생성한다.
rhcp(ifile, ofile)	HDFS상의 파일을 복사한다.
rhdel(file)	HDFS의 파일을 삭제한다.
rhls(path)	HDFD상의 경로에 있는 파일목록을 보여준다.
rhread(files)	files경로에 있는 map, sequence, text파일을 HDFS상으로 읽어 들인다.
rhkill(job)	MapReduce 잡을 멈춘다.

reduce 스크립트

```
reduce = expression(    pre = {          },
                        reduce = {          },
                        post = {         }
                        )
```

위 스크립트에서 reduce 함수는 3개의 인자를 가지고 있다. pre 인자는 변수를 초기화 하고 reduce 함수가 시작할 때 마다 실행되어진다. reduce 인자는 reduce task에서 실제로 실행될 계산 작업을 정의하며 reduce.value가 도착하는 대로 실행되어진다. post 인자는 reduce 인자에서 계산 되어진 결과 값을 Hadoop으로 전송하는 역할을 한다.

3.3. MapReduce 실행

MapReduce 실행은 rhwatch() 함수를 이용하여 map 함수과 reduce 함수를 실행 시키는 역할을 수행한다.

mapreduce 스크립트

```
rhwatch(    map=map 함수, reduce=reduce 함수,
            input=input경로 #문자형일 경우 : input=rhfmt(path-to-input, type='text')
            output=output 경로 #문자형일 경우 : output=rhfmt(path-to-output, type='text')
            )
```

위 스크립트에서 rhwatch() 함수는 map 인자와 reduce 인자, input 인자와 output 인자 등 다양한 인자를 지정한다. map 인자와 reduce 인자에는 사용자가 정의한 map 함수와 reduce 함수를 지정해 주고, input과 output의 경로를 설정할 때는 파일의 데이터 형을 파악하여 경로를 지정하고 문자형(text)인 경우에는 rhfmt() 함수를 이용하여 경로와 데이터 형을 지정해 주어야 한다. 수치형(sequence)일 경우에는 경로만 지정하여 주어도 무관하다.

4. Rhipe 성능 실험 및 논의사항

4.1. 실험환경

본 논문에서 사용된 실험환경은 Table 4.1와 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. Rhipe cluster environment

노드 수	master		1대
	slaves		5대
물리적 성능	CPU		Intel(R)Core(TM)2 Duo CPU E8300@2.83GHz
	RAM	master	4G
		slaves	2G
	Network Interface Card		RealTek
소프트웨어 버전	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)		

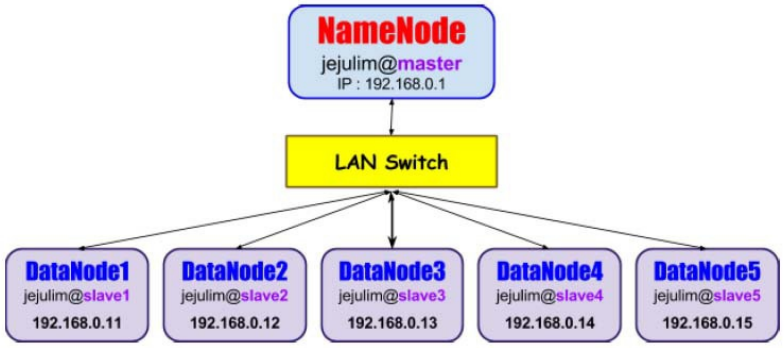

Figure 4.1. Configuration of Hadoop distributed processing system

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(Ciliendo 등, 2007). 여기서 CPU의 유휴상태 판단은 CPU 유휴(idle) 값이 98% 이상 높은 수치를 나타내는 경우로 제한하였다.

4.2. 회귀분석을 위한 MapReduce 구현

회귀분석에서 k개의 독립변수 $X_1, X_2, \ldots, X_k$와 종속변수 Y 사이의 관계가 선형관계이면 다음과 같이 다중 선형 회귀모형(multiple linear regression model)을 설정한다.

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k + \epsilon. \tag{4.1}$$

여기서 $\beta_0, \beta_1, \ldots, \beta_k$는 회귀계수(regression coefficient)이고 ϵ는 오차 항을 나타내고 평균 $\mu = 0$이고 표준편차 σ인 정규분포를 따른다고 가정한다. n개의 관찰값에 대해 행렬을 사용하여 표현하면 다음과 같다.

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

여기서 각각

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

이며 $\boldsymbol{\beta}$의 최소제곱 추정치는 다음과 같다.

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y} \tag{4.2}$$

Hadoop의 HDFS에서는 n행의 $\boldsymbol{Y}$와 $\boldsymbol{X}$가 L개의 블럭으로 분할되어 저장된다.

따라서 $\boldsymbol{X} = [X_1^T, X_2^T, \ldots, X_L^T]^T$라고 하면 식(4.2)에서 $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$와 $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y}$ 계산은 아래와 같이 각각의 블럭에 대해 곱셈연산 후 합산하여 계산된다.

$$\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} = \sum_{j=1}^{L} X_j^T X_j, \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y} = \sum_{j=1}^{L} X_j^T Y_j \tag{4.3}$$

식 (4.3)을 Rhipe의 map 함수로 나타내면 다음과 같다.

```
map = expression ({
            datMat  = do.call("rbind", lapply(strsplit(unlist(map.values),","),as.numeric))
                xx  = na.omit(datMat[,c(1,2,3,4,5,6,7)])
              yMat  = as.matrix(xx[,1])
              xMat  = cbind( 1, as.matrix(xx[,c(2,3,4,5,6,7)]))
               XtX  = crossprod(xMat, xMat)
               XtY  = crossprod(xMat, yMat)
                      rhcollect (" ",cbind(XtX,XtY)) })
```

위 map 함수는 HDFS에 저장된 각 블럭에 대해 변수의 개수 k=7인 데이터를 하나의 map task를 생성해 인수 map.values를 사용하여 읽고 결측처리한 후 $X_j^T X_j$와 $X_j^T Y_j$을 계산한다. 그리고 rhcollect() 함수를 사용하여 <키, 값> 쌍을 생성시켜 Hadoop으로 전송한다. 여기서 키는 " "를 사용하여 임의의 값으로 설정하였고 $X_j^T X_j$와 $X_j^T Y_j$의 계산결과를 값으로 설정하였다.

식 (4.3)을 Rhipe의 reduce 함수로 나타내면 다음과 같다.

```
reduce = expression(
            pre  = { sum ← 0 },
         reduce  = { for(x in reduce.values) sum← sum +x },
           post  ={ XtX = sum[, -dim(sum)[2]]
                    XtY = sum[, dim(sum)[2]]
                    betahat = solve(as.matrix(XtX), as.matrix(XtY))
                    rhcollect("B", betahat) })
```

Table 4.2. Variables used in real airline data

번호	변수	설명
1	Month (X_1)	월, 1-12
2	DayofMonth (X_2)	일, 1-31
3	DayofWeek (X_3)	요일, 1 (Monday) - 7 (Sunday)
4	ActualElapsedTime (X_4)	실제 경과시간 (단위: 분)
5	DepDelay (X_5)	출발 지연시간 (단위: 분)
6	Distance (X_6)	비행 거리, 마일기준
7	ArrDelay (Y)	도착 지연시간 (단위: 분)

Table 4.3. A part of real airline data

Month (X_1)	DayofMonth (X_2)	DayOfWeek (X_3)	ActualElapsedTime (X_4)	DepDelay (X_5)	Distance (X_6)	ArrDelay (Y)
10	14	3	91	11	447	23
10	15	4	94	-1	447	14
10	16	6	97	11	447	29
10	17	7	78	-1	447	-2
10	18	1	93	19	447	33
⋮	⋮	⋮	⋮	⋮	⋮	⋮
11	27	5	78	-3	483	-9
11	28	6	84	-1	483	-1
11	29	7	90	2	483	8
11	30	1	81	2	483	-1
11	1	7	55	-2	483	5

위 reduce 함수는 각각의 블럭에 대해 map 함수의 결과인 $X_j^T X_j$와 $X_j^T Y_j$에 대해 누적합을 계산한 식 (4.2)에 의해 회귀계수를 추정한다. 이를 위해 pre 인자에서 변수에 대해 초기화한 후 reduce 인자에서 누적합을 계산하고 post 인자에서 회귀계수를 추정하여 rhcollect() 함수를 사용하여 <키, 값> 쌍을 생성시켜 Hadoop으로 전송한다. 여기서 키는 "B"이고 값은 "betahat"이다.

4.3. 데이터 노드개수에 따른 Rhipe 성능 비교

4.3.1. 실제 데이터 실제 데이터는 2009년 ASA(americal 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 실험에서는 29개 변수 중에서 결측치가 많고 회귀분석에 유용하지 않은 변수를 제외하여 Table 4.2에 있는 7개 변수에 대해 2.33 GB에 해당되는 데이터를 중심으로 실험하였다.

Table 4.3은 Table 4.2에 있는 변수들에 대해 얻어진 데이터의 일부분을 나타낸다. 실제 데이터에서 데이터 크기에 따라 Rhipe의 성능을 비교하기 위해 2.33 GB을 중심으로 작은 데이터는 샘플링기법을 이용하여 얻었고 큰 데이터는 원래 데이터를 2배, 3배수 취하여 얻었다.

본 절에서는 여러가지 데이터 크기에 따른 실제데이터를 가지고 Hadoop의 가상분산 모드와 완전분산

Table 4.4. Comparison of pseudo-distributed and fully-distributed clusters with actual data in terms of computational time

데이터 크기	가상분산 모드	완전분산모드					No.of tasks
		1 Node	2 Nodes	3 Nodes	4 Nodes	5 Nodes	
100 MB (5,181,150 lines)	91.334	86.147	66.030	61.077	61.050	61.044	2
200 GB (10,362,150 lines)	141.331	136.482	80.991	71.194	61.069	61.064	4
300 GB (15,543,450 lines)	186.728	181.768	111.143	76.330	76.001	66.953	5
500 MB (25,905,751 lines)	267.538	257.090	136.238	121.419	71.291	71.035	8
1.00 GB (53,052,912 lines)	516.255	490.425	258.070	192.411	148.508	134.470	16
2.33 GB (123,534,970 lines)	1145.573	1094.745	577.511	395.761	294.356	264.859	38
4.66 GB (247,069,940 lines)	2207.643	2134.667	1100.140	746.305	568.570	468.183	75
9.32 GB (494,139,880 lines)	4439.984	4243.735	2170.253	1430.414	1095.074	874.074	149

모드에서 다중 회귀분석을 수행하는데 Rhipe의 처리속도를 비교하였다. Table 4.4와 Figure 4.2는 데이터 크기에 따라 가상분산 모드와 완전분산 모드에서 데이터 노드의 개수에 따른 처리속도를 나타내는 표와 그림이다. 여기서 표의 데이터 크기에서 괄호 안에 숫자는 데이터 행의 수를 나타내고 오른쪽 열에 있는 "No.of tasks"는 map task의 개수를 나타낸다.

Table 4.4의 실험결과를 보면 먼저, 가상분산 모드과 완전분산 모드 모두에서 데이터 크기에 비례하여 계산시간이 증가하는 것을 알 수 있다. 이것은 데이터의 크기가 클수록 시스템이 처리해야할 map task의 개수가 늘어나면서 MapReduce 실행시간이 그만큼 더 길어지기 때문이다. 그리고 완전분산 모드와 가상분산 모드 두 모드간 처리시간 비교는 예상했듯이 데이터 노드의 개수에 관계없이 완전분산 모드가 가상분산 모드보다 처리 시간이 적게 걸리는 것을 알 수 있다. 또한, 완전분산 모드 내에서 데이터 노드의 개수가 많을수록 같은 크기의 데이터를 처리하는데 걸리는 시간은 점점 줄어드는 것을 알 수 있다. 이것은 완전분산 모드 시스템에서 기본 64MB 입력스플릿 하에서 병렬처리할 수 있는 용량은 '64MB × 데이터 노드 수'인데, 데이터 노드의 개수가 많을수록 병렬처리 가능 용량의 증가로 인하여 처리속도가 빨라졌기 때문이다.

Figure 4.2를 보면, 모든 데이터 크기에서 데이터 노드의 개수가 많을수록 완전분산 모드의 처리속도는 짧고, 가상분산 모드에서 가장 긴 처리시간을 갖고 있음을 알 수 있다.

Table 4.5는 실제 데이터에서 MapReduce 구현을 통해 데이터 크기에 따라 얻어진 추정된 회귀직선식이다.

Table 4.5의 추정된 회귀직선으로부터 Table 4.2에 있는 여러가지 독립변수들로부터 도착지연시간을 나타내는 종속변수 ArrDelay을 추정 및 예측할 수 있다.

4.3.2. 모의실험 데이터 본 실험에서는 Table 4.3에 나와있는 실제 데이터와 유사한 모의 데이터를 얻

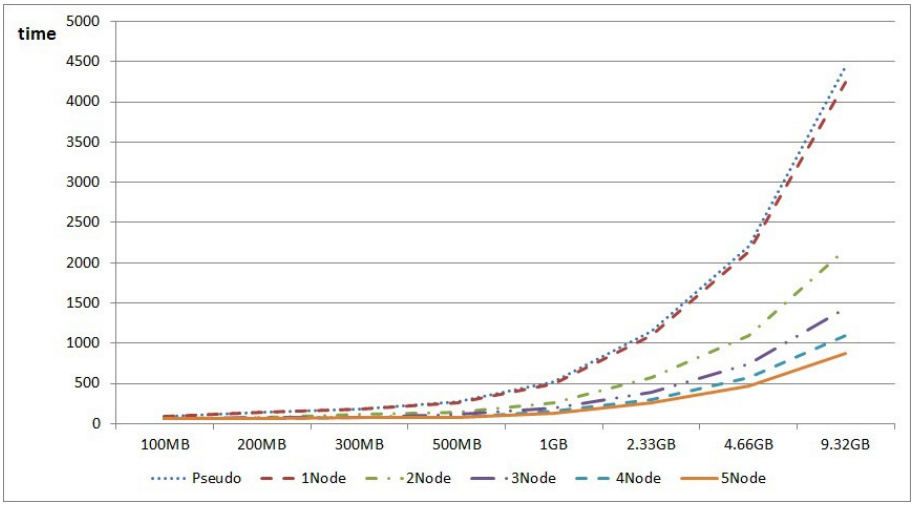

Figure 4.2. Processing speed curves of pseudo-distributed and fully-distributed clusters with actual data

Table 4.5. Estimated regression equations with actual data

데이터 크기	추정된 회귀직선식
100 MB	$\hat{Y} = -23.4885 + 0.0749X_1 - 0.0076X_2 - 0.0141X_3 + 0.5675X_4 + 0.9955X_5 - 0.0692X_6$
200 MB	$\hat{Y} = -19.5599 - 0.0786X_1 - 0.0044X_2 - 0.0239X_3 + 0.4921X_4 + 0.9993X_5 - 0.0605X_6$
300 MB	$\hat{Y} = -18.6769 - 0.0371X_1 + 0.0009X_2 - 0.0330X_3 + 0.4649X_4 + 0.9993X_5 - 0.0570X_6$
500 MB	$\hat{Y} = -20.4884 - 0.0238X_1 + 0.0026X_2 - 0.0510X_3 + 0.5165X_4 - 0.0637X_5 - 0.0630X_6$
1.00 GB	$\hat{Y} = -21.6326 + 0.0146X_1 + 0.0008X_2 - 0.0675X_3 + 0.5481X_4 - 0.9948X_5 - 0.0666X_6$
2.33 GB	$\hat{Y} = -18.9887 + 0.0232X_1 + 0.0001X_2 - 0.0723X_3 + 0.5435X_4 + 0.9312X_5 - 0.0664X_6$
4.66 GB	$\hat{Y} = -18.9887 + 0.0232X_1 + 0.0001X_2 - 0.0723X_3 + 0.5435X_4 + 0.9312X_5 - 0.0664X_6$
9.32 GB	$\hat{Y} = -18.9887 + 0.0232X_1 + 0.0001X_2 - 0.0723X_3 + 0.5435X_4 + 0.9312X_5 - 0.0664X_6$

기위해 독립변수 $X_1, ..., X_6$와 종속변수 Y들에 대해 평균 $\mu = 10$이고 표준편차 $\sigma = 100$인 정규분포로부터 독립적으로 생성된 난수에서 정수부분만을 가지고 실험에 사용하였다.

Table 4.6과 Figure 4.3은 모의실험 데이터에서 데이터 크기에 따라 가상분산 모드와 완전 분산모드에서 데이터 노드의 개수에 따른 처리속도를 나타내는 표와 그림이다.

Table 4.6의 실험결과를 보면, 모의실험 데이터는 실제 데이터보다 같은 데이터 용량이라도 데이터 행의 수가 많음으로 인해 Table 4.4와 비교하여 처리시간이 짧음을 알 수 있다. 가상분산 모드과 완전분산 모드 비교에서 실제데이터 실험 결과와 비슷한 결과를 얻을 수 있었다.

Figure 4.3에서도 Figure 4.2에서와 같이 데이터 노드의 개수 증가에 따라 처리시간이 감소하는 양상을 보였다.

Table 4.7은 모의실험 데이터에서 MapReduce 구현을 통해 데이터 크기에 따라 얻어진 추정된 회귀직선식을 나타내고 있다.

Table 4.6. Comparison of pseudo-distributed and fully-distributed clusters with simulated data in terms of computational time

데이터 크기	가상분산 모드	완전분산모드					No.of tasks
		1 Node	2 Nodes	3 Nodes	4 Nodes	5 Nodes	
100 MB (4,052,125 lines)	71.098	71.017	55.872	45.908	45.985	45.895	2
200GB (8,104,250 lines)	116.327	116.454	65.907	55.940	45.909	45.904	4
300GB (12,156,375 lines)	146.531	141.314	86.012	60.978	55.953	45.942	5
500 MB (20,260,625 lines)	207.236	201.614	111.165	91.183	60.949	55.617	8
1.00GB (41.492.145 lines)	385.142	369.075	192.463	147.124	101.925	101.792	16
2.33GB (96.615.449 lines)	847.631	825.634	235.384	234.176	228.393	183.150	38
4.66GB (193,230,989 lines)	1650.130	1602.151	821.471	558.964	436.738	350.680	75
9.32GB (386,461,796 lines)	3265.437	3159.687	1619.516	1067.020	822.428	652.731	149

Table 4.7. Estimated regression equations with simulated data

데이터 크기	추정된 회귀직선식
100 MB	$\hat{Y} = 9.9286 - 0.0005X_1 - 0.0011X_2 + 0.0001X_3 + 0.0005X_4 - 0.0001X_5 - 0.0007X_6$
200 MB	$\hat{Y} = 9.9880 - 0.0006X_1 - 0.0003X_2 - 0.0005X_3 + 0.0003X_4 + 0.0005X_5 - 0.0002X_6$
300 MB	$\hat{Y} = 9.9878 - 0.0005X_1 - 0.0005X_2 - 0.0002X_3 + 0.0002X_4 - 0.0004X_6$
500 MB	$\hat{Y} = 10.0015 - 0.0004X_1 - 0.0004X_2 + 0.0002X_3 + 0.0001X_4 + 0.0001X_5 - 0.0003X_6$
1.00 GB	$\hat{Y} = 9.9976 - 0.0004X_1 - 0.0003X_2 - 0.0002X_3 + 0.0001X_4 + 0.0001X_5 - 0.0003X_6$
2.33 GB	$\hat{Y} = 9.9968 - 0.0005X_1 - 0.0003X_2 - 0.0002X_3 + 0.0001X_4 + 0.0001X_5 - 0.0004X_6$
4.66 GB	$\hat{Y} = 9.9964 - 0.0005X_1 - 0.0003X_2 - 0.0002X_3 + 0.0001X_4 + 0.0001X_5 - 0.0004X_6$
9.32 GB	$\hat{Y} = 9.9965 - 0.0005X_1 - 0.0033X_2 - 0.0002X_3 + 0.0014X_4 - 0.0001X_5 - 0.0003X_6$

4.4. 기존 패키지와 Rhipe 성능 비교

Hadoop의 완전분산 클러스터에서 Rhipe의 성능을 평가하기 위해 작은 데이터에서 가능한 내장된 R패키지 stats의 lm() 함수와 bigmemory 패키지 상에서 회귀분석을 위한 biglm 패키지와 처리속도를 비교하였다.

Table 4.8은 실제 데이터에서 각 패키지들이 수행가능한 데이터 크기에 따라 계산시간을 나타내고 있다.

Table 4.8로부터 stats 패키지는 데이터 크기가 200 MB에서 연산이 불가능하고, biglm 패키지는 9.32 GB에서 불가능하였다. 그러나, Rhipe 패키지는 biglm 패키지에서 연산이 불가능한 9.32 GB까지 처리가 가능하였다. 데이터 크기 100 MB에서 Rhipe와 stats 패키지 비교에서 Rhipe의 패키지 속도가 약간 늦게 나타나는 이유는 Rhipe은 실제 데이터에서 map task마다 결측치 처리 과정으로 인해 추가적으로 시간이 소비되기 때문으로 사료된다. 우리가 관심있는 Rhipe와 biglm 패키지 비교에서 데이터 크기가 작은 경우는 biglm 패키지의 처리속도가 빠르고, 데이터 크기가 클수록 Rhipe의 처리속도가 훨씬 빠

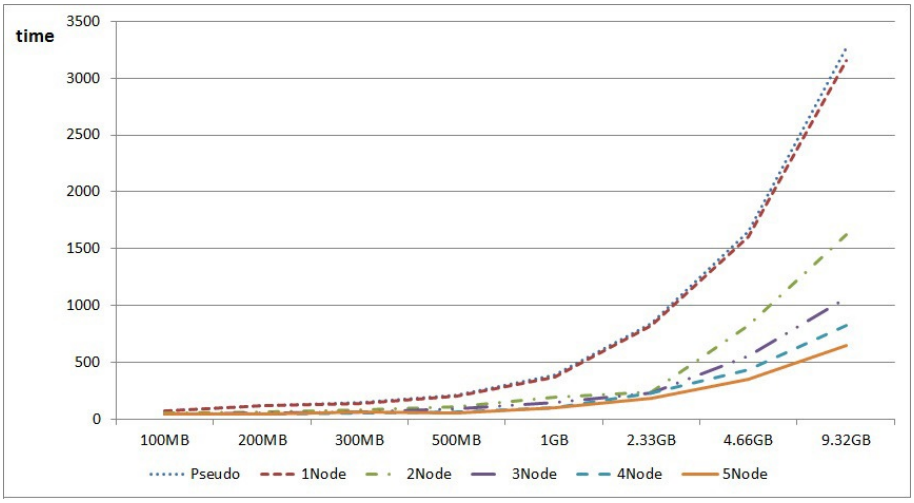

Figure 4.3. Processing speed curves of pseudo-distributed and fully-distributed clusters with simulated data

Table 4.8. Comparisons of three packages with actual data

데이터 크기	stats	biglm	Rhipe	No.of tasks
100 MB (5,181,150 lines)	58.143	23.491	61.044	2
200 MB (10,362,150 lines)	Fail	47.709	61.064	4
300 MB (15,543,450 lines)	-	77.423	66.953	8
500 MB (25,905,751 lines)	-	143.700	71.035	8
1.00 GB (53,052,912 lines)	-	284.849	134.470	16
2.33 GB (123,534,970 lines)	-	708.388	264.859	38
4.66 GB (247,069,940 lines)	-	6779.768	468.183	75
9.32 GB (494,139,880 lines)	-	Fail	874.074	149

름을 알 수 있다. 특히, 2.33 GB에서 Rhipe 패키지는 biglm 패키지보다 약 2.7배 속도 차이가 났으나 4.66 GB에서는 약 14.5배 정도 속도차이가 많이 났다.

Table 4.9는 모의실험 데이터에서 각 패키지들이 수행가능한 데이터 크기에 따라 계산시간을 나타내고 있다.

Table 4.9에 따르면, stats 패키지인 경우 실제 데이터는 200 MB에서 연산이 불가능하였으나 여기서는 300 MB에서 연산이 불가능하였다. 이것은 모의실험 데이터는 실제 데이터보다 데이터 행의 수가 적음

Table 4.9. Comparisons of three packages with simulated data

데이터 크기	stats	biglm	Rhipe	No.of tasks
100 MB (4,052,125 lines)	51.197	19.148	45.895	2
200 MB (8,104,250 lines)	190.808	36.752	45.904	4
300 MB (12,156,375 lines)	Fail	57.486	45.942	5
500 MB (20,260,625 lines)	-	99.551	55.617	8
1.00 GB (41.492.145 lines)	-	207.55	101.792	16
2.33 GB (96.615.449 lines)	-	485.694	183.150	38
4.66 GB (193,230,989 lines)	-	1266.618	350.680	75
9.32 GB (386,461,796 lines)	-	Fail	652.731	149

으로 상대적으로 메모리에 큰 데이터 로딩이 가능했기 때문이다. 그리고 Rhipe 패키지와 biglm 패키지 비교에서 데이터 크기가 클수록 biglm 패키지의 처리속도는 Rhipe패키지의 처리속도보다 훨씬 늦음을 알 수 있다.

5. 결론 및 향후 연구

최근의 스마트 폰과 같은 모바일 컴퓨팅 환경이 빠르게 확산되면서 하루에도 셀 수 없을 정도로 엄청난 양의 데이터가 생성되기도 하고 소멸되기도 한다. 과거에는 쓸모없는 것으로 인식되어 버려졌던 데이터들도 빅데이터 시대를 맞아 기업들은 대량의 데이터로부터 패턴을 추출하고 필요한 정보를 찾아내어 미래 변화를 예측함으로 시장 선도의 기회를 창출할 수 있다.

통계엔진인 R은 유연성 있는 통계 프로그래밍 환경을 제공하지만 컴퓨터의 주메모리에 올릴 수 있는 데이터 크기만 처리할 수 있어 대용량 데이터 분석에는 한계를 갖고 있다. Hadoop 또한 대용량 데이터를 분산처리 할 수 있는 환경을 제공하지만 사용자 친밀성, 통계알고리즘 적용, 데이터 시각화 등의 기능을 갖고 있지 않다. R과 Hadoop의 통합환경인 Rhipe 개발로 인해 분산처리 환경 하에서 대용량 데이터 분석이 가능하다.

본 논문에서는 Rhipe을 이용하여 실제 데이터 및 모의실험 데이터에서 다양한 데이터의 크기에 따라 다중 회귀분석을 구현하였다. Hadoop의 가상분산 모드와 완전분산 모드에서 데이터 노드의 개수 증가에 따른 처리 속도를 비교 분석하였다. 그리고 Rhipe 플랫폼의 성능을 평가하기 위해 기존의 stats 패키지, 그리고 bigmemory 패키지 상에서 유용한 biglm 패키지와 처리 속도를 비교하였다.

성능실험 결과, Hadoop의 가상분산 모드와 완전분산 모드에서 데이터 노드의 개수 증가에 따른 Rhipe의 성능은 완전분산 모드에서 데이터노드의 수가 많을수록 같은 크기의 데이터를 분석하는데 걸리는 시간이 점점 줄어드는 것을 알 수 있었다. 또한, 기존 패키지와의 성능비교에서 stats 패키지는 작은 데이터처리만 가능하고 biglm 패키지는 어느정도 큰 데이터 처리는 가능하지만 너무 느리다는 단

점을 갖고 있다. 하지만 Rhipe은 데이터의 크기가 클수록 map task 개수의 증가로 인해 동시 병렬 처리됨으로 다른 패키지들보다 빠른 처리속도를 보였다. 실제 데이터에서 Rhipe 패키지와 다른 패키지와의 속도차이는 모의실험 데이터보다 훨씬 큼을 알 수 있었다. 그러나 Rhipe의 성능 향상을 위해 향후 개선해야할 부분도 있다. Rhipe 성능 실험에서 9.32GB의 모의실험 데이터와 실제데이터를 처리하는데 각각 약 11분과 15분 정도 계산시간이 소요되었다. 물론 Rhipe은 테라, 페타 바이트 급 이상의 빅데이터 처리 또한 가능하나 실시간으로 즉각적인 솔루션을 제공하는 분야에 적용하기 위해서는 속도 향상에 대한 연구가 필요하다.

References

고영준, 김진석. (2013). Rhipe를 활용한 빅데이터 처리 및 분석, *한국데이터정보과학회지*, **24(5)**, 975-987.

Adler, D., Nenadic, O. Zucchini, W. and Glaser, C. (2007). The ff package: Handling large data sets in R with memory mapped pages of binary flat files, UseR2007, http://www.r-project.org/conferences/useR-2007/program/presentations/adler.pdf

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

Ciliendo, E., Kunimasa, T. and Braswell, B. (2007). Linux Performance and Tuning Guidelines, IBM.

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

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

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

Kane, M. J. and Emerson, J. W. (2010a). bigmemory: Manage massive matrices with shared memory and memory-mapped files, Rpackage version 4.2.3.

Kane, M. J. and Emerson, J. W. (2010b). biganalytics: A library of utilities for big.matrix objects of package bigmemory , R package version 1.0.12.

Laney, D. (2001)., 3D Data Management: Controlling Data Volume, Velocity, and Variety. META Group.

Lin, H., Yang, S. and Midkiff, S. P. (2013). *A Parallel R Framework for Processing Large Dataset on Distributed Systems*, DataCloud.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C. and Byers, A. H. (2011). *Big data: The next frontier for innovation, competition, and productivity*, McKinsey Global Institute.

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.

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

빅데이터 처리 및 분석을 위한 Rhipe 플랫폼

정병호[a] · 신지은[a] · 임동훈[a,1]

[a]경상대학교 정보통계학과

(2014년 09월 30일 접수, 2014년 12월 22일 수정, 2014년 12월 23일 채택)

요 약

R과 Hadoop의 통합환경인 Rhipe 개발로 인해 분산처리 환경 하에서 대용량 데이터 분석이 가능해졌다. 본 논문에서는 Rhipe을 이용하여 실제 데이터와 모의실험 데이터에서 다양한 데이터 크기에 따라 다중 회귀분석을 구현하였다. Hadoop의 가상분산 모드(pseudo-dstributed mode)와 완전분산 모드(fully- distributed mode) 구축 시스템 비교에서 완전분산 모드 시스템이 가상분산 모드 시스템보다 처리 속도가 빠르고 데이터 노드의 수가 많을수록 계산 시간이 점점 줄어드는 것을 알 수 있었다. 또한, 제안된 Rhipe 플랫폼의 성능을 평가하기 위해 기본 R 패키지인 stats와 bigmemory 상에서 유용한 biglm 패키지와 처리 속도를 비교하였다. 실험결과 Rhipe은 데이터의 크기가 클수록 map task 개수가 증가되고 동시에 병렬 처리로 인해 다른 패키지들보다 빠른 처리속도를 보였다.

주요용어: 빅데이터, R, Hadoop, Rhipe, 다중회귀분석.

이 논문은 2011년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2011-0010089).

[1]교신저자: (660-701) 경남 진주시 가좌동 900, 경상대학교 정보통계학과 교수 및 RINS.
Email: dhlim@gnu.ac.kr