Journal of the Korean Data & Information Science Society
2015, **26**(5), 1155–1166

http://dx.doi.org/10.7465/jkdi.2015.26.5.1155
한국데이터정보과학회지

RHadoop을 이용한 빅데이터 분산처리 시스템[†]

신지은[1] · 정병호 [2] · 임동훈[3]

[123]경상대학교 정보통계학과

접수 2015년 7월 2일, 수정 2015년 9월 3일, 게재확정 2015년 9월 22일

요 약

기하급수적으로 증가하는 대용량 데이터를 저장, 분석하는데 기존 방식으로는 거의 불가능하여 이를 가능케 해 주는 기술이 바로 하둡이다. 최근에 R은 하둡기술을 활용하여 분산처리에 기반한 빅데이터 분석 엔진으로 활용되고 있다. 본 논문에서는 R과 하둡의 통합환경인 RHadoop을 이용하여 실제 데이터와 모의실험 데이터에서 다양한 데이터 크기에 따라 병렬 다중 회귀분석을 구현하고자 한다. 또한, 제안된 RHadoop 플랫폼의 성능을 평가하기 위해 기본 R 패키지의 lm 함수, bigmemory 상에서 유용한 biglm 패키지와 처리 속도를 비교하였다. 실험결과 RHadoop은 데이터 노드가 많을수록 병렬처리로 인해 빠른 처리속도를 보였고 또한 대용량의 데이터에 대해 다른 패키지들보다 빠른 처리속도를 보였다.

주요용어: 병렬회귀분석, 빅데이터, 하둡, R, RHadoop.

1. 서론

2012년 미국의 리서치 전문업체인 가트너 (Gartner)는 빅데이터를 '데이터 양 (volume)이 많고, 데이터 형태가 다양 (variety)하고, 데이터 속도 (velocity)가 빠른 데이터'로 정의하였다 (Beyer와 Laney, 2012). 위키피디아 (wikipedia)에서는 빅데이터를 '기존 데이터베이스 관리 도구로서 데이터 수집, 저장, 관리, 분석 영역을 넘어선 대량의 정형 또는 비정형 데이터 집합'으로 정의하였다 (Manyika 등, 2011).

모바일 회사인 Tech Spartan에 의하면 2014년 기준 인터넷에서 1분 동안 생산되는 데이터량은 페이스북에서는 600,000건의 로그인이 되고, 트위터에서는 433,000건의 메시지가 트윗 (tweet)되고, 구글에서는 419만건의 검색이 이루어지고, 유튜브에서는 306시간 분량의 비디오가 업로드되고, 애플 아이튠즈에서는 50,200개 앱이 다운로드되고, 아마존에서는 80,000건 거래가 이루어지고, 이메일은 136,319,444개 전송된다고 하였다 (Tech Spartan, 2014). 오늘날, 이처럼 대용량 데이터 시대를 맞이하여 IBM, HP, SAP, MS, EMC, 오라클 같은 대형 IT 벤더 회사들은 빅데이터 솔루션을 개발 중에 있다.

빅데이터 시대가 도래하면서 새로운 처리/분석 패러다임이 요구되고 있고 미국중심으로 통계엔진인 R이 기업용 분석 플랫폼으로 확산되고 있다. R은 이미 구글, 페이스북, 야후, 아마존 등 닷컴 기업의 분석 플랫폼으로 사용 중에 있다. 특히, 오라클, IBM, 테라데이터 등 빅데이터의 고성능 분석을 추구

[†] 이 논문은 2011년도 정부 (교육부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (No.2011-0010089).
[1] (660-701) 경남 진주시 가좌동 900번지, 경상대학교 정보통계학과, 석사과정.
[2] (660-701) 경남 진주시 가좌동 900번지, 경상대학교 정보통계학과, 박사과정 수료.
[3] 교신저자: (660-701) 경남 진주시 가좌동 900번지, 경상대학교 정보통계학과, 교수 및 RINS.
E-mail: dhlim@gnu.ac.kr

하는 IT업체들은 실시간 분석을 위해 메모리나 데이터베이스에서 직접 분석을 실시하는 인-메모리 분석 (in-memory analytics) 혹은 인-데이터베이스 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 기존의 모든 데이터를 메모리에 로딩 후 처리하는 작업방식에서 벗어나 데이터 구조만을 메모리에 로딩하여 처리하는 방식으로 너무 느리다는 단점과 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). R과 Hadoop의 통합환경으로는 Rhipe (Guha, 2010; Hafen 등, 2014; Ko와 Kim, 2013; Jung 등, 2014) 과 Rhadoop 등이 있다. RHadoop은 레볼루션 어낼리틱스 (revolution analytics)에 의해 개발되어 데이터 분석도구인 R과 대용량 처리 시스템인 Hadoop과 연동하여 빅데이터 처리/분석을 수행할 수 있는 오픈소스 솔루션이다. 지금까지 RHadoop에 대한 연구로는 Prajapati (2013), Oancea와 Dragoescu (2014), Harish 등 (2015)이 있고 국내에서는 주로 Park 등 (2013)에 의해 연구가 이루어져왔다. 여기서 Prajapati (2013), Park 등 (2013)은 RHadoop의 가상분산 모드 (pseudo-dstributed mode)에서 회귀분석 (regression analysis)을 포함한 통계 계산 알고리리즘을 맵리듀스로 구현하였다.

본 논문에서는 RHadoop 플랫폼에서 Prajapati (2013), Park 등 (2013)에서 다루지 못한 완전분산 모드 (fully-distributed mode)에서 다중회귀분석 (multiple linear regression analysis)을 통해 데이터 노드의 개수 증가에 따른 처리 속도를 비교 분석하고자 한다. 또한, 제안된 RHadoop 플랫폼의 성능을 평가하기 위해 실제 데이터와 모의실험 데이터에서 기존의 R 함수 lm ()과 bigmemory 패키지 상에서 유용한 biglm 패키지와 처리 속도를 비교하고자 한다.

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

2. R과 Hadoop 개요

R은 통계 계산과 데이터 가시화를 위한 무료 소프트웨어이다. R은 UNIX, 윈도우 그리고 MacOS 등 다양한 운영체제를 지원하며 Java, C, Fortran 등의 프로그래밍 언어와 연동이 가능하여 전세계적으로 많은 사용자 들이 있다. R은 데이터 핸들링을 위한 기능과 데이터 분석을 위한 도구 그리고 데이터 가사화를 위한 강력한 그래픽 함수들을 갖고 있다. R은 25개 표준 패키지와 CRAN 인터넷 사이트 (http://CRAN.R-project.org)로부터 필요한 패키지를 다운로드하여 사용한다. R은 많은 공식적인 통계기구(Todorov, 2010, 2012)에서 정식 통계를 내는 플랫폼으로 사용하고 있을 뿐만 아니라 생물정보학, 금융, GIS 등 많은 분야에서 활용되고 있다 (Oancea와 Dragoescu, 2014).

Hadoop은 대용량 데이터를 처리할 수 있는 소프트웨어로 자바 기반의 오픈소스 프레임워크이다. Hadoop의 핵심기술은 Figure 2.1과 같이 HDFS (Hadoop Distributed File System)와 MapReduce로 구성되어 있다.

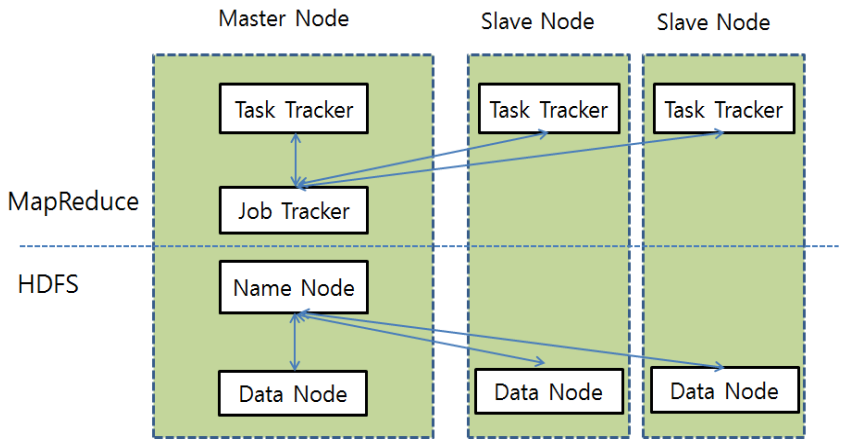

Figure 2.1 Basic architecture of Hadoop

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.

MapReduce 프로그램은 함수의 입력과 출력이 모두 키 (key)와 값 (value)의 쌍으로 구성 되어 있다. 각 함수는 <키, 값> 쌍의 집합을 또다른 <키, 값> 쌍의 집합으로 변환한다. Map 함수는 HDFS에서 불러온 데이터를 가공하여 새로운 <키, 값> 집합을 출력한다. Reduce 함수는 같은 키를 갖는 값들을 묶어 <키, (값1, 값2,· · ·)> 식의 새로운 <키, 값> 쌍의 집합을 만든다. Reduce 함수는 여기에 집계 연산을 수행하여 또 다른 <키, 값> 쌍의 집합을 생성하고 이를 HDFS에 저장한다.

MapReduce에 대한 이해를 돕기 위해 Hadoop에서 흔히 예를 드는 단어세기 (word count) 예를 가지고 Map과 Reduce에서 어떻게 처리되는지 살펴보고자 한다. 여기서 단어세기란 텍스트를 단어별로 나누어 빈도수를 계산하는 것을 말한다. Figure 2.2는 단어세기에 대한 MapReduce 동작과정을 보여주고 있다. Figure 2.2에서 보면 입력데이터는 블럭 단위로 분할 (split)되고 각 블럭은 <키, 값> 쌍으로 표현된다. 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.

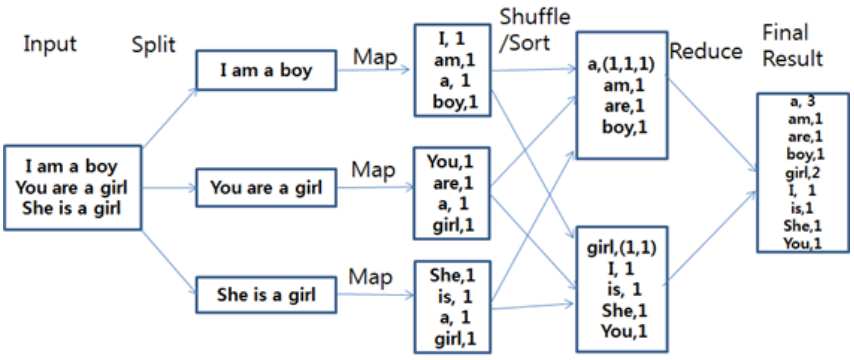

Figure 2.2 MapReduce parallel processing

Hadoop은 설치방식에 따라 가상분산 모드와 완전분산 모드로 구분한다. 가상분산 모드는 Hadoop 데몬 프로세스가 하나의 로컬 컴퓨터 상에서 동작하는 방식으로 네임노드와 데이터노드, 잡 트래커와 태스크 트래커가 모두 하나의 컴퓨터에서 동작한다. 그리고 완전분산 모드는 Hadoop 데몬 프로세스가 여러 대의 컴퓨터로 구성된 클러스터 상에서 동작하는 방식으로 여러 대의 서버에 분산 저장과 분산 연산을 알 수 있는 장점을 갖고 있다.

3. RHadoop 개요

RHadoop은 R과 Hadoop의 통합환경을 제공하는 오픈 소스 프로젝트이다. RHadoop은 Figure 3.1과 같이 3개의 R패키지 즉, rmr, rhdfs 그리고 rhbase로 구성되어 있다. rmr은 R에서 MapReduce 기능을 제공하는 패키지이고 rhdfs는 R에서 HDFS 파일 관리 기능을 제공하는 패키지이고 rhbase는 R에서 HBase 데이터베이스 관리를 제공하는 패키지이다. rmr 패키지는 R 클라이언드 뿐만 아니라 모든 태스크 트랙커 노드에 설치되고 rhbase 패키지는 Thrift 서버를 통해 Hbase에 접근한다. 본 논문에서는 RHadoop의 패키지 중에서 rmr과 rhdfs만을 사용한다.

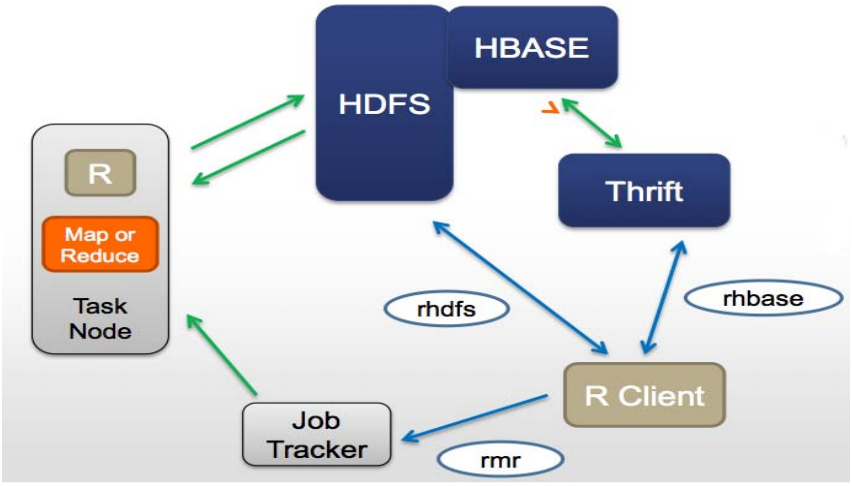

Figure 3.1 RHadoop architecture

Table 3.1은 본 논문에서 사용 중인 RHadoop에서 제공되는 HDFS와 MapReduce 관련 함수들이다.

Table 3.1 Various functions for HDFS and MapReduce operations

Function	Description
hdfs.init ()	Initialize HDFS
keyval (key,value)	Create and extract key-value pairs
mapreduce (map,reduce)	Execute MapReduce job
hdfs.put (ifile, ofile)	Copy files from the local filesystem to HDFS filesystem
hdfs.rm (files)	Delete the directory of file from HDFS
hdfs.ls (files)	List the directory from HDFS
hdfs.read (files)	Read the file from HDFS
to.dfs ()	Write R objects from or to the filesystem
from.dfs ()	Read R objects from the HDFS filesystem

Figure 3.1로부터 Map 함수는 HDFS상의 데이터를 가공하여 새로운 <키, 값> 쌍의 집합을 출력한다. 그리고 keyval() 함수를 이용하여 <키, 값>쌍의 집합을 생성한 후 Hadoop으로 전송한다. 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. MapRedce 실행은 mapreduce() 함수를 이용하여 Map 함수와 Reduce 함수를 실행 시키는 역할을 수행한다.

4. RHadoop 성능 실험 및 평가

4.1. 실험환경

본 논문에서 사용된 실험환경은 Table 4.1와 같이 Hadoop 0.20.0 기반 하에 6대의 컴퓨터 서버를 사용하여 그 중 1대 서버를 마스터 (master) 노드, 나머지 5대 서버를 슬레이브 (slave) 노드로 설정하여 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	4G
		slaves	2G
	Network Interface Card		RealTeck
Software version	OS		14.04LTS
	Java		1.7.0
	Hadoop		0.20.2
	R		3.1.0
	rmr2		3.3.0
	rhdfs		1.0.8
Switch Hub	Cisco Catalyst 2960		

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 반복 측정하여 평균값을 계산하였으며 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. 병렬 회귀분석을 위한 맵리듀스 구현

회귀분석에서 k개의 독립변수 $X_1, X_2, \ldots, X_k$와 종속변수 Y 사이의 관계가 선형관계이면 다음과 같이 다중 선형 회귀모형을 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. 식 (4.1)을 행렬로 표현하면 다음과 같다.

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{4.2}$$

여기서 각각

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

따라서, 최소제곱법에 의해 추정된 회귀계수 $\hat{\boldsymbol{\beta}}$는 정규방정식

$$(\boldsymbol{X^T X})\hat{\boldsymbol{\beta}} = \boldsymbol{X^T Y} \tag{4.3}$$

의 해로 주어진다.

RHadoop의 HDFS에서는 $\boldsymbol{Y}$와 $\boldsymbol{X}$가 대용량인 경우 L개의 블럭으로 분할되어 여러 개의 데이터 노드에 저장된다. 따라서 $\boldsymbol{X} = [X_1^T, X_2^T, \ldots, X_L^T]^T$ 라고 하면 식 (4.3)에서 $\boldsymbol{X^T X}$와 $\boldsymbol{X^T Y}$ 계산은 다음과 같이 각각의 블럭에 대해 곱셈연산 후 합산하여 얻어진다.

$$\boldsymbol{X^T X} = \sum_{j=1}^{L} X_j^T X_j, \boldsymbol{X^T Y} = \sum_{j=1}^{L} X_j^T Y_j \tag{4.4}$$

식 (4.4)의 $\boldsymbol{X^T X}$와 $\boldsymbol{X^T Y}$의 계산은 RHadoop의 rmr에서 다음과 같이 구현된다.

$$\begin{aligned} XtX = values(from.dfs(\quad & mapreduce(input = X, \\ & map = function(., Xi)\ keyval(1, list(t(Xi)\%*\%Xi)), \\ & reduce = function(., YY)\ keyval(1, list(Reduce('+', YY))), \\ & combine = TRUE)))[[1]] \end{aligned}$$

$$\begin{aligned} XtY = values(from.dfs(\quad & mapreduce(input = X, \\ & map = function(., Xi)\ keyval(1, list(t(Xi)\%*\%Y)), \\ & reduce = function(., YY)\ keyval(1, list(Reduce('+', YY))), \\ & combine = TRUE)))[[1]] \end{aligned}$$

위 XtX와 XtY에서 mapreduce 함수는 map 함수, reduce 함수 그리고 combine 함수로 구성되어 있다. map 함수에서는 HDFS에 저장된 각 블럭에 대해 $X_j^T X_j$와 $X_j^T Y_j$을 계산한다. 그리고 keyval()함수를 사용하여 <키, 값> 쌍을 생성시켜 Hadoop으로 전송한다. 여기서 키값은 1로 동일한 값으로 설정하였고 $X_j^T X_j$와 $X_j^T Y_j$의 계산결과를 값으로 설정하였다. reduce 함수는 각각의 블록에 대해 map 함수의 결과인 $X_j^T X_j$와 $X_j^T Y_j$에 대해 누적합을 계산하여 keyval() 함수를 사용하여 <키, 값> 쌍을 생성시켜 Hadoop으로 전송한다. 그리고 combine 함수는 동일한 키를 갖는 경우 성능 최적화 (optimization)를 위해 사용하고 있다.

우리가 구하고자 하는 $\hat{\boldsymbol{\beta}}$은 식 (4.3)에 의해 R에서 다음과 같이 구현된다.

$$betahat = solve(XtX, XtY)$$

여기서 $betahat$는 $\hat{\boldsymbol{\beta}}$을 나타낸다.

4.3. 데이터 노드개수에 따른 RHadoop 성능 비교

(1) 실제 데이터

실제 데이터는 2009년 미국 규격 협회 (Americal Standards Association; ASA)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 작은 데이터는 샘플링기법을 이용하여 얻었고 큰 데이터는 원래 데이터를 2배, 3배수 취하여 얻었다.

Table 4.2 Variables used in real airline data

Number	Name	Description
1	Month (X_1)	1-12
2	DayofMonth (X_2)	1-31
3	DayofWeek (X_3)	1 (Monday) - 7 (Sunday)
4	ActualElapsedTime (X_4)	actual elapsed time (in minutes)
5	DepDelay (X_5)	departure delay (in minutes)
6	Distance (X_6)	in miles
7	ArrDelay (Y)	arrival delay (in minutes)

Table 4.3은 Table 4.2에 있는 변수들에 대해 얻어진 데이터의 일부분을 나타낸다.

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

Table 4.4는 데이터 크기에 따라 가상분산 모드와 완전 분산모드에서 데이터 노드의 개수에 따른 처리속도를 나타내고 있다. 먼저, 데이터의 크기가 클수록 시스템이 처리해야할 map task의 개수가 늘어나면서 MapReduce 실행시간이 그만큼 더 길어짐을 알 수 있다. 가상분산모드와 완전분산모드 비교는 예상했듯이 완전분산모드가 가상분산모드보다 훨씬 처리속도가 빠름을 알 수 있다. 그리고 완전분산 모드에서 데이터 노드의 수가 많을수록 같은 크기의 데이터를 처리하는데 걸리는 시간이 점점 줄어듬을 알 수 있다.

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

data size	pseudo-distributed mode	fully-distributed mode					No.of map tasks
		1 Node	2 Nodes	3 Nodes	4 Nodes	5 Nodes	
100 MB	71.081	70.434	68.916	61.888	60.919	60.203	2
200 MB	136.577	136.520	81.138	79.709	71.898	70.258	4
300 MB	186.188	185.644	120.166	82.790	81.766	71.881	5
500 MB	272.023	271.780	147.020	130.807	83.792	82.071	8
1.00 GB	542.579	537.926	291.084	219.982	161.972	156.051	16
2.33 GB	1240.452	1230.898	637.452	447.822	343.222	282.808	38
4.66 GB	2446.390	2431.671	1234.653	838.019	648.928	531.151	75
9.32 GB	4858.350	4840.009	2443.033	1640.810	1246.966	1003.031	149

Table 4.5는 실제 데이터에서 데이터 크기에 따라 추정된 회귀직선을 나타내고 있다. Hadoop의 가상분산 모드, 그리고 완전 분산모드에서 데이터 노드 개수와 무관하게 동일한 추정된 회귀직선을 얻을 수 있었다.

Table 4.5 Estimated regression equation with actual data

data size	estimated regression equation
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	$\hat{y} = -18.6769 - 0.0371X_1 + 0.0009X_2 - 0.0330X_3 + 0.4649X_4 + 0.9993X_5 - 0.0680X_6$
500 MB	$\hat{y} = -20.4884 - 0.0238X_1 + 0.0026X_2 - 0.05102X_3 + 0.5165X_4 - 0.0637X_5 - 0.0630X_6$
1.00 GB	$\hat{y} = -21.6326 + 0.01464X_1 + 0.0008X_2 - 0.0675X_3 + 0.5481X_4 - 0.9948X_5 - 0.0664X_6$
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	$\hat{y} = -18.9887 + 0.0232X_1 + 0.0001X_2 - 0.0723X_3 + 0.5435X_4 + 0.9312X_5 - 0.0664X_6$

(2) 모의실험 데이터

본 모의 실험에서는 6개의 독립변수 $X_1, \ldots, X_6$와 종속변수 Y에 대해 실제 데이터와 유사한 데이터를 얻기위해 평균 $\mu = 10$이고 표준편차 $\sigma = 100$인 정규난수로부터 정수부분만을 가지고 실험하였다.

Table 4.6은 모의실험 데이터에서 데이터 크기에 따라 가상분산 모드와 완전 분산모드에서 데이터 노드의 개수에 따른 처리속도를 나타내는 표이다. 실제 데이터에 대한 Table 4.4와 비교했을 때 전반적으로 처리속도가 빠르다는 것을 알 수 있다. 모의실험 데이터에서 추정된 회귀직선을 나타내는 Table 4.7에서도 실제 데이터에 대한 Table 4.5와 같이 Hadoop의 설치방식에 관계 없이 동일한 추정된 회귀직선을 얻을 수 있었다.

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

data	pseudo-distributed	fully-distributed mode					No.of map
size	mode	1 Node	2 Nodes	3 Nodes	4 Nodes	5 Nodes	tasks
100 MB	55.901	53.823	50.405	48.947	48.248	47.824	2
200 MB	98.933	98.583	57.722	57.246	52.485	52.086	4
300 MB	129.105	128.850	87.363	60.278	57.974	50.999	5
500 MB	190.020	189.004	103.758	98.501	59.683	59.178	8
1.00GB	366.435	363.162	189.337	144.143	103.353	103.048	16
2.33GB	838.635	829.714	441.501	313.312	231.297	188.880	38
4.66GB	1640.615	1629.949	821.471	558.964	436.738	350.680	75
9.32GB	3287.266	3241.825	1647.704	1130.506	837.511	687.521	149

Table 4.7 Estimated regression equation with simulated data

data size	estimated regression equation
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 성능 비교

(1) 실제 데이터

Hadoop의 완전분산 클러스터에서 RHadoop의 성능을 데이터 크기에 따라 평가하기 위해 작은 데이터에서 가능한 내장된 R패키지의 lm 함수와 bigmemory 패키지 상에서 회귀분석을 위한 biglm 패키지와 처리속도를 비교하였다.

Table 4.8는 실제 데이터에서 데이터 크기에 따라 lm 함수, biglm 패키지 그리고 RHadoop의 처리속도를 나타내고 있다. 이 표에 의하면, lm 함수는 메모리 부족으로 200 MB에서 연산이 불가능하였고 biglm 패키지와 RHadoop 비교에서는 300 MB까지는 biglm 패키지가 빠른 처리 속도를 보이나 500 MB부터 데이터 크기가 클수록 RHadoop는 완만한 속도 감소를 보인 반면 biglm 패키지는 현저하게 처리속도가 줄어 듬을 알 수 있다. 특히, 9.32 GB에서 biglm 패키지는 4.66 GB에서 처리속도의 11배 정도 급격한 속도 감소를 보였다. 이것은 biglm 패키지의 거의 연산 불가능한 상태에 해당된다. 그러나 9.32 GB에서 RHadoop은 biglm 패키지처럼 큰 폭의 속도 감소가 없음을 알 수 있다. 이처럼 RHadoop이 biglm 패키지에 비해 빠른 처리 속도를 보이는 것은 biglm 패키지는 대용량의 데이터를 한 대의 서버에서 제한적인 메모리 범위 내에서 동작하는 반면에 RHadoop는 대용량의 데이터를 많은 map task로 쪼개 여러 대의 서버에 분산하여 병렬처리하기 때문이다.

Table 4.8 Comparisons of three packages with actual data in terms of computational time in seconds

data size	lm	biglm	RHadoop	No. of tasks
100 MB (5,181,150 lines)	49.950	24.718	60.203	2
200 MB (10,362,150 lines)	Fail	49.572	70.258	4
300 MB (15,543,450 lines)	-	69.922	71.881	8
500 MB (25,905,751 lines)	-	121.248	82.071	8
1.00 GB (53,052,912 lines)	-	268.792	156.051	16
2.33 GB (123,534,970 lines)	-	722.051	282.808	38
4.66 GB (247,069,940 lines)	-	1461.737	531.151	75
9.32 GB (494,139,880 lines)	-	15956.947	1003.031	149

(2) 모의실험 데이터

Table 4.9는 모의실험 데이터에서 데이터 크기에 따라 lm함수, biglm 패키지 그리고 RHadoop의 처리속도를 나타내고 있다. 먼저, lm함수는 데이터 크기가 300 MB까지는 연산이 가능하고 500 MB부터 연산이 불가능하였다. biglm 패키지와 RHadoop 비교에서는 200 MB까지 biglm 패키지가 빠른 처리속도를 보이나 300 MB부터는 RHadoop가 더 빠른 속도를 보임을 알 수 있다. 실제 데이터에 대한 Table 4.8과 비교에서 모의실험 데이터는 같은 데이터 크기라 하더라도 상대적으로 단순함으로서 biglm 패키지와 RHadoop 처리 속도 간 차이가 실제 데이터보다 크지 않음을 알 수 있다. 결론적으로 말하면, 실제 데이터처럼 다양하고 복잡할수록 RHadoop는 상대적으로 다른 패키지보다 빠른 처리 속도를 가짐을 알 수 있다.

Table 4.9 Comparisons of three packages with simulated data in terms of computational time in seconds

data size	lm	biglm	RHadoop	No. of tasks
100 MB (4,052,125 lines)	43.882	20.064	47.824	2
200 MB (8,104,250 lines)	90.069	37.846	52.086	4
300 MB (12,156,375 lines)	181.443	59.412	50.999	5
500 MB (20,260,625 lines)	Fail	97.936	59.178	8
1.00 GB (41.492.145 lines)	-	198.393	100.397	16
2.33 GB (96.615.449 lines)	-	571.694	188.880	38
4.66 GB (193,230,989 lines)	-	1197.902	359.516	75
9.32 GB (386,461,796 lines)	-	2867.802	687.521	149

5. 결론 및 향후 연구

우리는 스마트 폰과 소셜네트워크서비스 (SNS)의 대중화로 인해 데이터 양이 기하급수적으로 증가하는 빅데이터 시대에 살고 있다. 엄청난 양의 데이터 생성으로 기존의 데이터 저장, 관리, 분석 기법으로는 데이터를 처리하는데 한계가 있어 새로운 정보기술의 패러다임이 요구되고 있다.

R과 Hadoop의 통합환경인 RHadoop 개발로 인해 분산처리 환경 하에서 대용량 데이터 처리, 분석이 가능해졌다.

본 논문에서는 RHadoop 플랫폼 상에서 모의실험 데이터 뿐 만아니라 실제 데이터에서 다양한 데이터의 크기에 따라 병렬 다중 회귀분석을 구현하였다. RHadoop 플랫폼의 성능을 데이터 노드 증가에 따라 평가하였으며 또한 기존 패키지의 lm 함수, 그리고 bigmemory 패키지 상에서 유용한 biglm 패키지와 처리 속도를 비교하였다.

RHadoop 플랫폼의 성능실험 결과 Hadoop 완전분산 모드에서 데이터 노드의 수가 많을수록 같은 크기의 데이터를 분석하는데 걸리는 시간이 점점 줄어드는 것을 알 수 있었다. 실제 데이터에서는 모의실험보다 RHadoop패키지의 처리속도 증가폭은 훨씬 큼을 알 수 있었다.

또한, RHadoop 플랫폼의 기존 패키지와의 비교에서는 lm 함수는 작은 데이터처리만 가능하고 biglm 패키지는 어느정도 큰 데이터 처리는 가능하지만 너무 느리다는 단점을 갖고 있다. 하지만 RHadoop는 데이터의 크기가 클수록 map task 개수의 증가로 인해 동시 병렬 처리됨으로 다른 패키지들보다 빠른 처리속도를 보였다.

그러나 RHadoop의 사용자 측면에서 성능 향상은 물론 향후 개선해야할 부분도 있다. RHadoop은 여전히 까다로운 정보기술로 인해 범용화의 어려움이 있고 성능 실험에서도 예를 들면, 9.32 GB의 모의실험 데이터와 실제데이터를 처리하는데 각각 약 11분과 16분 정도 소요되었듯이 계산시간의 문제점을 갖고 있다.

References

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

Beyer, M. A. and Laney, D. (2012). *The importance of big data: A definition*, Gartner, Stanford.

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*. Ph. D. 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*, **1**, 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*, 1-34.

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). *bigmemory: Manage massive matrices with shared memory and memory-mapped files*, R package version 4.2.3, `https://cran.r-project.org/package=bigmemory`.

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.

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

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.

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.

Tech Spartan. In *An Internet Minute-2013 VS 2014*, http://www.techspartan.co.uk/features/internet-minute-2013-vs-2014-infographic/, 2014.

Todorov, V. and Templ, M. (2012). *R in the statistical office: Part 2*, Development, policy, statistics and research branch working paper 1/2012, United Nations Industrial Development Organization, Vienna.

Todorov, V. (2010). *R in the statistical office: The UNIDO experience*, Development, policy, statistics and research branch working paper paper 03/2010, United Nations Industrial Development Organization, Vienna.

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

Journal of the Korean Data & Information Science Society
2015, **26**(5), 1155–1166

http://dx.doi.org/10.7465/jkdi.2015.26.5.1155
한국데이터정보과학회지

Big data distributed processing system using RHadoop[†]

Ji Eun Shin[1] · Byung Ho Jung[2] · Dong Hoon Lim[3]

[123]Department of Information Statistics, Gyeongsang National University

Received 2 July 2015, revised 3 September 2015, accepted 22 September 2015

Abstract

It is almost impossible to store or analyze big data increasing exponentially with traditional technologies, so Hadoop is a new technology to make that possible. In recent R is using as an engine for big data analysis based on distributed processing with Hadoop technology. With RHadoop that integrates R and Hadoop environment, we implemented parallel multiple regression analysis with various data sizes of actual data and simulated data. Experimental results showed our RHadoop system was faster as the number of data nodes increases. We also compared the performance of our RHadoop with lm function and biglm packages available on bigmemory. The results showed that our RHadoop was faster than other packages owing to paralleling processing with increasing the number of map tasks as the size of data increases.

Keywords: Big data, Hadoop, parallel regression analysis, R, RHadoop

[†] 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] Master student, Department of Information Statistics, Gyeongsang National University, Jinju 660-701, Korea.

[2] Ph.D. student, Department of Information Statistics, 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