Journal of the Korean Data & Information Science Society
2013, **24**(5), 975–987

http://dx.doi.org/10.7465/jkdi.2013.24.5.975
한국데이터정보과학회지

Rhipe를 활용한 빅데이터 처리 및 분석[†‡]

고영준[1] · 김진석[2]

[1]제주테크노파크 디지털융합센터 · [2]동국대학교 정보통계학과

접수 2013년 6월 26일, 수정 2013년 7월 21일, 게재확정 2013년 8월 3일

요 약

최근 Hadoop은 빅데이터의 저장, 처리 및 분석을 위한 표준시스템으로 인식되고 있으며, 많은 빅데이터 관련 시스템들이 Hadoop에 기반하여 구축되고 있다. 또한 R은 다른 소프트웨어와의 연동이 쉽고 다양한 분석 라이브러리들을 탑재하고 있어서 Hadoop 환경하에서 빅데이터의 분석을 위한 공통 분석 플랫폼으로 여겨지고 있다. 본 논문에서는 Hadoop 환경에서 분산 데이터 처리를 위한 R패키지인 Rhipe를 소개하고 빅데이터를 이용한 병렬 다중회귀분석을 위해 MapReduce 프로그램을 작성하는 방법을 예시하였다. 또한 시뮬레이션을 통해 기존의 대용량처리를 위한 R 분석패키지인 ff와 bigmemory와의 연산속도를 비교하였으며, 데이터의 크기가 커짐에 따라 Rhipe를 이용한 MapReduce 프로그램의 계산속도가 ff와 bigmemory에 비해 우수함을 확인하였다.

주요용어: 맵리듀스, 빅데이터, 하둡, 하둡분산파일시스템

1. 서론

오늘날 인터넷의 보편화, 스마트 기기의 발전, 트위터나 페이스북과 같은 소셜 미디어의 등장으로 인해 다양한 정보와 많은 양의 데이터가 수집되고 있다. IDC에 의하면 기하급수적으로 늘어나는 데이터의 양은 2020년에 이르러 약 35.2 제타바이트에 달할 것으로 전망되고 있다 (Gantz와 Reinsel, 2011). 이렇게 우리는 한대의 컴퓨터에서 처리할 수 없는 다양하고 방대한 데이터를 처리하고 분석해야 하는 빅데이터 시대를 직면하고 있다. 빅데이터가 주목받고 있는 시점에 빅데이터를 바라보는 견해는 다양하다. 이를테면, IDC는 업무수행에 초점을 맞춰 "빅데이터는 다양한 종류의 대규모 데이터로부터 저렴한 비용으로 가치를 추출하고, 데이터의 초고속 수집, 발굴, 분석을 지원하도록 고안된 차세대 기술 및 아키텍처"라고 정의하였다. 또한 McKinsey는 규모적 측면에 초점을 맞춰 "기존 데이터베이스 관리 도구의 데이터 수집, 저장, 관리, 분석하는 역량을 넘어서는 데이터 셋 규모로, 그 정의는 주관적이며 앞으로도 계속 변화될 것이다."라고 하였다 (Manyika 등, 2011). IBM은 데이터는 방대한 규모 (volume), 종류의 다양성 (variety), 데이터 처리 및 분석의 시의성 (velocity)인 3가지 속성 (3V)을 가지고 있으며, 이러한 속성을 가지는 데이터를 빅데이터라고 정의하였다 (Zikopoulos 등, 2012). 또한, Oracle (2012)에서는 빅데이터의 속성을 IBM의 3V에 가치 (value)를 더하여 정의하고 있다.

[†] 이 논문은 2012년도 정부 (교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2010-0023302)
[‡] 이 논문은 제1저자 석사학위논문 (2013)의 일부를 수정, 보완한 것임.
[1] (690-787) 제주특별자치도 제주시 중앙로 217, 제주테크노파크 디지털융합센터, 전임연구원.
[2] 교신저자: (780-714) 경상북도 경주시 석장동 707, 동국대학교 정보통계학과, 조교수.
E-mail: jinseog.kim@gmail.com

이처럼 다양한 분야에서 빅데이터를 바라보는 시각은 해당분야 혹은 이해관계에 따라 다양하지만 빅데이터 관련기술이 미래의 핵심기술이 될 것이라는 전망에는 이견이 없어 보인다. 2012년 개최된 세계경제포럼 (다보스포럼)에서는 향후 가장 주목할 만한 기술 중에 하나로 빅데이터 관련 기술을 지목하였고, 이에 따라 빅데이터의 처리 및 분석 기술관련 인력양성이 향후 국가나 기업의 미래 경쟁력을 좌우하는 시대가 도래할 것이라고 전망하였다 (World Economic Forum, 2012).

빅데이터 시대의 도래에 따라 빅데이터를 저장하고 분석하기 위한 기술개발이 이슈가 되고 있다 (Ahn과 Hwang, 2012; Jun 등, 2011). 이 중에서도 빅데이터의 효율적인 저장과 신속한 분석에 필요한 고성능 컴퓨팅 (HPC; high performance computing) 관련 연구가 활발히 진행되고 있다 (Eddelbuttel, 2012). 빅데이터의 저장 및 처리에 대한 대표적인 기술로는 Google에서 개발한 GFS (Google file system)와 MapReduce가 있으며, 이를 기반으로 아파치 재단에 의하여 개발된 Hadoop이 있다. Hadoop은 신뢰할 수 있고 (reliable), 확장이 용이하며 (scalable), 분산 컴퓨팅 (distributed computing) 환경을 지원하는 오픈 소스 (open source) 소프트웨어로 알려져 있다 (White, 2012; http://hadoop.apache.org/). 최근 Yahoo, amazon.com 등 많은 기업에서 빅데이터의 처리를 위해 Hadoop을 활용하고 있으며, 사실상 Hadoop은 빅데이터의 처리를 위한 표준 플랫폼으로 인식되고 있다.

한편, 빅데이터의 분석을 위해서 Google, Facebook은 분석엔진으로 R을 활용하고 있으며, Oracle, IBM의 Netezza, SAP, Teradata 등에서도 in-memory 혹은 in-database 분석 엔진으로 R을 채택하고 있다. 또한, 통계소프트웨어로 잘 알려진 SAS나 SPSS는 R과의 연동을 위한 인터페이스를 제공하고 있을 뿐만 아니라, Revolution (2011)은 Hadoop과 R을 통합한 시스템을 상용 비즈니스 툴을 제공하고 있다. 이처럼, R은 빅 데이터 분석에서 공통 분석 플랫폼으로 여겨지고 있는 실정이다. 이는 R이 빅데이터 분석과 관련된 다양하고 최신의 라이브러리를 제공할 뿐만 아니라 오픈소스 프로젝트이며 프로그램 언어인 Java, C, Python 등과의 연동이 용이하기 때문이다. 국내에서도 R의 Google 맵 라이브러리를 활용한 통계분석이 시도되는 등 (Han 등, 2012) 앞으로 빅데이터 분석에 R의 활용은 점점 활발해질 전망이다.

본 논문의 목적은 최근 개발된 Hadoop기반 하에서 R을 이용하여 빅데이터 처리 및 분석을 지원하는 패키지인 Rhipe (Guha, 2010)에 대하여 소개하고, Rhipe를 이용하여 병렬화 다중회귀분석을 구현하는 것이다. 본 논문은 다음과 같이 구성되어 있다. 2절에서는 Hadoop의 주요 시스템인 HDFS와 MapReduce를 간단히 소개하고 R과 Hadoop을 연동해주는 Rhipe 패키지에 대하여 설명한다. 3절에서는 Rhipe를 이용하여 병렬화 다중회귀분석을 구현하고, 모의실험을 통하여 기존의 대용량처리를 위한 패키지들과의 연산속도를 비교한다. 마지막은 본 연구의 결론 및 향후 연구과제이다.

2. Hadoop 개요

R 커뮤니티에서는 앞 절에서 설명한 바와 같이 빅데이터 분석을 위해 R에서 Hadoop과 연동하여 통계분석을 하게 해주는 패키지들을 제공한다. 하지만 Hadoop 관련 패키지를 사용하기 위해서는 먼저 Hadoop 시스템이 구축되어야 하고, Hadoop의 주요 구성요소인 HDFS와 MapReduce에 대한 이해가 필수적이다. 본 절에서는 Hadoop 시스템 및 그 구성요소인 HDFS와 MapReduce의 개요를 소개한다.

Hadoop은 대용량 데이터의 분산 저장 및 신속한 처리를 위해 다수의 컴퓨터를 네트워크로 연결하여 하나의 시스템과 같이 사용할 수 있도록 구성한 시스템이다. Hadoop 시스템은 Figure 2.1과 같이 주 컴퓨터 (master)들과 종속 컴퓨터들 (slaves)을 하나의 클러스터로 묶어 이루어져 있으며, 기능적 측면으로는 크게 HDFS와 MapReduce 시스템으로 구성되어 있다. 이 중, HDFS 시스템은 대용량 데이터를 분산 저장하고 처리하는 기능을 수행하며 Hadoop 클러스터의 주 컴퓨터 중 하나를 네임 노드

(name node), 그리고 종속 컴퓨터들을 데이터 노드 (data node)라고 부르고 있다. MapReduce 시스템은 HDFS에 분산 저장된 대용량 데이터를 이용해 병렬 연산을 지원하는 시스템이다. MapReduce 시스템에서는 Hadoop 클러스터의 주 컴퓨터중 하나를 job tracker라 하고 종속 컴퓨터들을 task tracker라고 부르고 있다 (Hedlund, 2011).

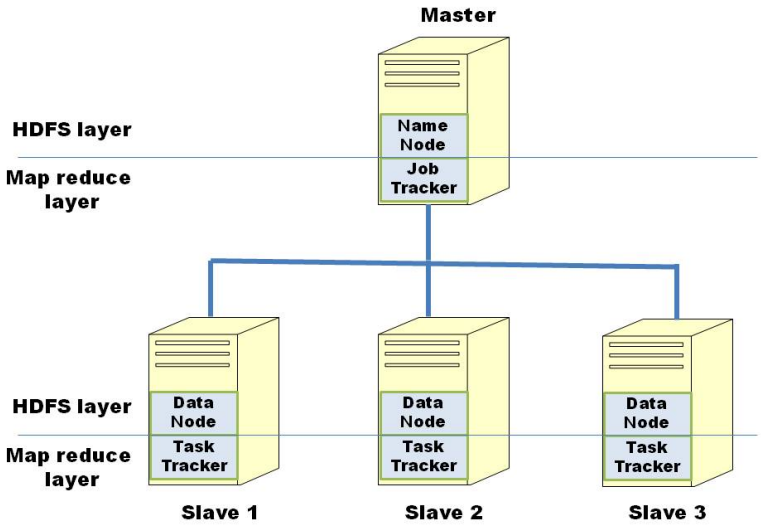

Figure 2.1 Architecture of the Hadoop cluster

2.1. HDFS

HDFS (Hadoop distributed file system)는 범용 컴퓨터들로 구성된 클러스터에서 안전하고 효율적으로 데이터를 저장하기 위한 목적으로 설계되었기 때문에 다음과 같은 특징을 갖는다 (Shvachko 등, 2010; http://hadoop.apache.org/).

네임 노드와 데이터 노드의 특징

HDFS는 물리적으로 네임 노드로 불리는 컴퓨터와 데이터 노드로 불리는 컴퓨터로 구성되어 있으며, 각각의 역할은 다음과 같다. 네임 노드에서는 HDFS 내의 모든 메타데이터 (metadata)를 관리하고, 클라이언트의 파일 액세스를 조절한다. HDFS 파일은 일반적으로 64M 크기인 블럭 (block) 단위로 분할되고, 각 블럭은 여러 개의 데이터 노드에 중복하여 분산 저장된다. 데이터 노드는 클라이언트의 요청에 의해 읽기 또는 쓰기를 제공하며, 네임 노드의 지시에 따라 블럭의 생성, 삭제 및 복사 기능을 수행한다.

데이터 블럭의 복제

HDFS는 저장된 데이터의 무결성 (data integrity)을 보장하기 위해 데이터 노드에 각 블럭들에 대해서 여러 개의 복제본을 갖는다. 일반적으로 3개의 복제본을 유지하며, 네임 노드는 데이터 노드들과 주기적인 통신을 통해 데이터 노드가 정상적으로 작동하는지를 확인한다. 만약 특정 데이터 노드에 문제가 생기면 네임 노드는 동일한 블럭을 가진 데이터 노드들에게서 해당 복제본을 다른 정상 데이터 노드에 복제하도록 함으로써 복제본 개수를 일정하게 유지한다.

2.2. MapReduce

MapReduce는 컴퓨터 클러스터 환경에서 페타바이트 이상의 대용량 데이터를 병렬 처리하기 위해서 만들어진 프로그램 모델이다. MapReduce는 리스트 형식의 데이터를 처리할 때 사용하는 함수인 map함수와 map작업에서의 결과를 취합하는 reduce함수를 기반으로 구성되어 있다. MapReduce는 사용자가 정의한 프로그램을 job이라는 단위로 관리하며, job은 여러 대의 노드에서 분산처리 되는데, 이때 각 노드에서 수행되는 세부 작업을 task라 부른다. task의 처리과정은 데이터를 여러 개의 데이터 조각으로 나눠서 분산 처리를 한 후 필요에 따라 그 결과를 하나로 모아 처리하고, 최종 결과를 다시 분산 파일 시스템에 저장하는 방식을 따른다. 이 때 분산 처리하는 과정을 map task라고 하고, 각각의 결과를 하나로 모아 처리하는 과정을 reduce task라 한다. map task와 reduce task는 key/value 쌍으로 아래와 같이 6단계를 통하여 병렬 처리하는 기능을 수행한다 (Dean과 Ghemawat, 2008; Duan 등, 2011). Figure 2.2는 map task와 reduce task에서 6단계를 거치는 과정을 그림으로 나타낸 것이다.

1. Split 단계: 입력 데이터에 대하여 일정 크기로 분할하고, 분할된 데이터를 map task로 할당한다.
2. Map 단계: 사용자가 정의한 map함수에 따라 분할 데이터를 key/value 쌍으로 처리하고, key를 기반으로 데이터를 분할한다.
3. Shuffle 단계: 분할된 데이터는 reduce task로 전송이 이루어진다.
4. Sort 단계: key를 기반으로 데이터 정렬이 이루어진다.
5. Reduce 단계: reduce task에 전송된 데이터는 key/value 쌍으로 묶여 사용자가 정의한 reduce 함수에 따라 처리된다.
6. Output 단계: 최종 결과는 사용자가 지정한 HDFS 디렉토리에 저장된다.

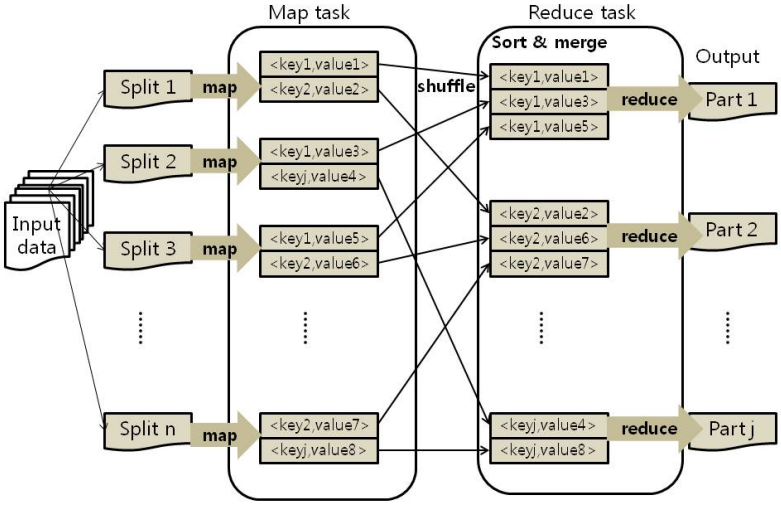

Figure 2.2 Flow of MapReduce process

3. Rhipe

Hadoop과 통계분석 소프트웨어인 R을 연동해주는 Rhipe 패키지의 기능에 대해서 소개한다. 또한, word counting 사례를 이용하여 R에서 빅데이터 분석 프로그램의 작성 방법을 설명한다.

Rhipe는 R과 Hadoop을 연동하여 MapReduce 기능을 R콘솔에서 사용 가능하게 하는 인터페이스와 HDFS에 분산 저장되어 있는 파일에 액세스 기능을 제공하는 R패키지로 Guha (2010)가 개발하였다. R과 Hadoop의 연동을 위해서는 서로 다른 환경에서 구조가 다른 데이터의 전송 및 저장을 위한 인터페이스를 필요로 하는 데, Rhipe는 Google에서 제공하는 Google protocol buffer를 이용하여 Hadoop과 통신을 한다. 즉, HDFS에 저장된 데이터를 R로 읽어오거나 R에서 생성한 자료를 HDFS에 저장하는 기능 또는 R에서 작성된 map과 reduce 프로그램을 Hadoop에서 실행토록 하는 역할을 담당한다.

Rhipe를 이용하여 자료분석을 하기 위해서는 먼저 rhinit() 함수를 실행하여 Rhipe와 Hadoop을 연결해야 한다. 이 때, rhinit() 함수의 실행결과가 TRUE 값이면 정상적인 연동이 되었음을 의미한다. Rhipe와 Hadoop이 정상 연결이 되면, Rhipe에서 제공하는 HDFS 및 MapReduce관련 함수들을 이용하여 Hadoop의 여러 가지 기능들을 제어할 수 있다.

HDFS 관련 함수

일반적인 R에서 제공하는 파일 관련 함수를 통해서는 HDFS에 저장된 파일에 접근하는 것이 불가능하다. HDFS 파일에 접근하기 위해서는 Rhipe에서 제공하는 함수를 사용해야 하며, HDFS 관련 함수를 이용하면 HDFS에 저장된 sequence, map 또는 text 형식의 파일에 대한 접근이 가능하다. Table 3.1은 Rhipe에서 제공하는 HDFS 관련 함수와 그에 대한 설명을 요약한 표이다.

Table 3.1 Functions related with HDFS in Rhipe package

Function	Description
rhdel(folders)	Delete HDFS files
rhls(path)	List HDFS files in a specified path
rhget(src, dest)	Copying from the HDFS moves files from the HDFS to a local directory
rhput(src, dest)	Put a local file to HDFS of text format
rhcp(src, dest)	Copy files (or folders) on the HDFS
rhwrite(list, dest)	Write R object of list type to HDFS with sequence format
rhread(files)	Read key/value pairs from the HDFS
rhgetkey(keys, path)	Get value associated with a key in a map file
rhmr(...)	Prepares a MapReduce job for execution

MapReduce의 실행

Rhipe를 통하여 map과 reduce를 실행하기 위해서는 map expression 작성, reduce expression 작성, MapReduce 실행에 필요한 정보가 담긴 R 오브젝트 생성, 마지막으로 MapReduce 실행과 같은 총 4단계의 과정을 거친다. 아래의 Table 3.2는 MapReduce 실행 단계별 관련 R코드를 정리한 표이다. 각 단계별 자세한 설명은 word counting을 예를 들어 설명하겠다.

Table 3.2 MapReduce procedures in Rhipe

procedures	Rhipe codes
1. map expression	map = expression({})
2. reduce expression	reduce = expression(pre = {}, reduce = {}, post = {})
3. Create R object	mrobj = rhmr(map = map, reduce = reduce, inout = c("text", "sequence"), ifolder = input_file, ofolder = output_file)
4. Execute map/reduce	jobid = rhex(mrobj)

1) map expression 작성

map expression은 Hadoop의 map task에 레코드 단위로 분할되어 전송된 데이터를 key와 value쌍으로 맵핑하는 작업을 정의하기 위하여 Rhipe에서 정의한 코드이다. HDFS에서 map task로 전송된 데이터는 다시 R로 전송되어 R 오브젝트인 map.values에 리스트 형태로 저장된다. 아래 코드에서 2, 3번째 행은 리스트 형태의 R 오브젝트를 벡터로 변환한 후 table 함수를 이용하여 각 단어의 빈도를 계산하는 코드이다. 이렇게 계산된 각 단어의 빈도는 6번째 라인의 rhcollect함수를 통해 개별단어와 빈도수를 key, value 쌍으로 하여 Hadoop 시스템으로 전송한다.

```
Map = expression({
                  word = unlist(strsplit(unlist(map.values)," "))
                  word_table = table(word)
                  tname = names(word_table)
                  for(i in 1:length(tname))
                  rhcollect(tname[i], word_table[i])
                })
```

2) reduce expression 작성

Hadoop은 map task를 통해 계산된 1차 결과 값을 key를 기반으로 정렬하여 reduce task로 전송한다. 전송된 데이터는 다시 R로 전송되어 key/value쌍의 리스트 타입으로 R 오브젝트인 reduce.values에 저장된다. Rhipe에서 reduce expression은 pre, reduce, post에 해당하는 세가지 인수에 대한 정의를 필요로 한다. 여기서 pre 인수는 reduce task에서 사용될 변수의 초기화를, reduce 인수는 reduce task에서 실행될 계산 작업을 정의한다. post 인수는 reduce 인수에서 정의된 결과 값을 Hadoop으로 전송하는 작업에 대하여 정의한다. 아래 코드의 3번째 라인은 map task에서 1차적으로 계산되어 reduce task로 전송된 reduce.values에서 동일한 key (단어)를 기반으로 value (빈도)의 누적합을 계산하는 코드이다. 이렇게 계산된 key (단어)와 value (누적빈도)는 4번째 라인의 코드를 통해 Hadoop으로 전송된다. Hadoop은 R에서 계산한 최종 결과 값을 전송받아 HDFS에 분산 저장한다.

```
Reduce = expression(
                    pre = { total = 0 },
                    reduce = {total = sum(total, unlist(reduce.values))},
                    post = { rhcollect(reduce.key, total) }
                   )
```

3) R 오브젝트 생성

R에서 MapReduce를 실행하기 위해서는 앞서 정의한 map/reduce expression과 더불어 MapReduce 실행에 필요한 기타 정보가 담긴 R 오브젝트를 필요로 한다. R 오브젝트는 아래의 코드와 같이 rhmr함수를 통해 생성된다.

```
Job<-rhmr(
      map = Map,      # map expression
      reduce = Reduce, # reduce expression
      # HDFS에 저장되어 있는 파일 타입, 분석 후 HDFS에 저장 될 파일 타입
      inout = c("text", "sequence"),
```

```
      # 분석 할 파일(HDFS) path
      ifolder = "/user/stat/text",
      # 분석 후 생성되는 파일(HDFS) path
      ofolder = "/user/stat/word_count"
)
```

4) map/reduce 실행

R에서 MapReduce의 실행은 map과 reduce에 필요한 정보가 담긴 R 오브젝트를 인수로 하는 rhex()함수를 실행함으로써 수행된다. rhex에 의해 최종적으로 실행된 결과는 rhread() 함수를 통해 R 오브젝트로 변환되며, 이 오브젝트는 R콘솔에서 확인할 수 있다. 참고로, Hadoop에서의 MapReduce의 처리 상황은 rhstatus() 함수를 통해 확인할 수 있고, 이를 통해 디버깅이 가능하다.

```
# map/reduce 실행
jobid = rhex(Job)
# map/reduce Job 모니터링
rhstatus(jobid, mon.sec=5, autokill=TRUE, showErrors=TRUE)
# HDFS 파일(결과 파일)을 R에 로드
out = rhread("/user/stat/word_count", type = "sequence")
```

4. Rhipe를 이용한 다중회귀분석

반응값과 입력자료를 $\mathbf{y}$와 X라고 할 때, 최소제곱법을 이용한 다중회귀모형에서의 회귀계수, $\boldsymbol{\beta}$의 추정은 아래와 같다.

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}.$$

Hadoop의 HDFS에서는 데이터 ($\mathbf{y}$와 X)가 k개의 블록으로 분할하여 저장된다. 따라서 Hadoop의 MapReduce 프로그램은 위의 식에서 $X^T X$와 $X^T \mathbf{y}$를 직접 계산할 수 없으며, 아래와 같이 분할 연산 후 합산하여 계산하여야 한다.

$$X^T X = \sum_{j=1}^{k} X_j^T X_j, \; X^T \mathbf{y} = \sum_{j=1}^{k} X_j^T \mathbf{y}_j.$$

위식을 Rhipe의 map expression으로 나타내면 아래와 같으며, 코드에서 마지막 라인은 “Regression”을 key, 그리고 계산 결과인 list(xtx, xty, yty)를 value로 하여 Hadoop으로 전송하기 위한 코드이다.

```
Reg_map = expression({
  # HDFS의 sequence파일을 R의 matrix 오브젝트로 변환
  xx = do.call("rbind", map.values)
  Y = xx[,1]              # Y 벡터 생성
  X = cbind(1, xx[,-1])    # X 행렬  생성
  xtx = t(X)%*%X          # X^tX 계산
  xty = t(X)%*%Y         # X^tY 계산
  yty = as.numeric(t(Y)%*%Y)  # Y^tY 계산
  rhcollect("Regresion", list(XtX = xtx, XtY = xty, YtY = yty))
})
```

아래는 reduce expression이며, map task에서의 부분계산 결과인 reduce.values을 이용하여 $X^TX, X^T\mathbf{y}$ 및 $\mathbf{y}^T\mathbf{y}$의 누적 합을 계산한 후, X^TX의 역행렬과 회귀계수를 추정하게 된다.

```
Reg_reduce = expression(
 pre={XtX = 0; XtY = 0; YtY = 0;}, # 변수 초기화
 reduce={
  for(i in seq_along(reduce.values)){
    XtX = XtX + reduce.values[[i]]$XtX
    XtY = XtY + reduce.values[[i]]$XtY
    YtY = YtY + reduce.values[[i]]$YtY
  }
  # 회귀계수 추정
  ...
  XtX_inverse = solve(XtX) # (XtX)의 역행렬
  Estimate = XtX_inverse %*% XtY # 회귀계수
  ...
```

다중회귀계수의 추정 및 가설검정을 위한 상세 Rhipe 프로그램은 부록을 참조하기 바란다.

Rhipe에서 구현한 다중회귀분석의 성능을 평가하기 위해서 다른 대용량 데이터의 회귀분석 R패키지들과의 연산속도를 비교하였다. 비교대상 R패키지는 대용량 자료의 처리를 위해 개발된 ff (Adler 등, 2007)와 bigmemory (Kane과 Emerson, 2010a)이다. ff와 bigmemory는 대용량 데이터의 처리만을 지원해 주므로 회귀분석을 위해서는 biglm (Lumley, 2009)과 biganalytics (Kane과 Emerson, 2010b) 패키지를 동시에 이용하여야 한다. 본 실험에서는 우리가 구현한 Rhipe에서의 회귀분석모듈과 ff+bigglm, bigmemory+biganalytics에서의 실험 결과를 비교하였다. 여기서 ff+bigglm, bigmemory+biganalytics의 실험결과는 Park (2013)의 결과를 인용하였다. 모의 실험에서는 자료의 크기를 120M, 240M, 480M, 1.2G, 2.4G로 하였고, 이 때 입력변수의 수는 모두 21개이다. 입력변수 및 종속변수에 대한 자료값은 R에서 표준정규분포에 의한 난수를 적당한 크기로 독립적으로 생성한 후 텍스트 형식으로 저장하였으며, 실험에 사용한 자료의 크기에 따라 결합하여 HDFS 파일로 각각 저장하였다.

모의 실험을 위한 환경을 위해 먼저 1대의 주 컴퓨터와 6대의 종속 컴퓨터들로 Hadoop 클러스터를 구축하였다. 클러스터에 사용된 모든 컴퓨터들의 사양은 듀얼 코어인 CEL-Dual E1400이고, RAM은 1GB이다. 각 컴퓨터에는 Ubuntu 12.04, Java 1.7.0_04, Hadoop 1.0.3 버전을 설치하였다. R은 2.15.1 버전을 설치하였으며, R과 Hadoop의 통신을 위해 Google protocol buffers 2.4.1를 설치하였다. 우리가 구축한 Hadoop시스템에서는 주 컴퓨터는 1대로 지정하였기 때문에 주 컴퓨터는 네임노드이면서 동시에 job tracker의 역할을 담당한다. 이상의 실험세팅에서 동일한 실험을 50번 반복 실시하였으며, Table 4.1은 데이터 크기에 따른 각 패키지별 연산속도를 나타낸 표이다.

Table 4.1 Simulation results: mean and standard deviations of computing times in seconds

Data size	ff+biglm	bigmemory+biganalytics	Rhipe	# of tasks
120M	20.94 (0.10)	20.52 (0.12)	46.81 (2.25)	2
240M	43.92 (0.17)	41.89 (0.20)	44.00 (1.99)	4
480M	91.01 (0.25)	86.18 (0.28)	50.57 (2.37)	8
1.2G	224.09 (0.31)	213.79 (0.60)	63.95 (2.58)	20
2.4G	444.51 (0.58)	Fail	86.19 (2.38)	39
10G	-	-	142.76 (0.64)	161

Table 4.1의 결과를 살펴보면 데이터 크기가 240M까지는 ff+bigmemory를 이용하여 연산하는 것이 빠르지만 그 이상의 데이터는 Rhipe가 빠르게 연산되는 것을 확인할 수 있다. 또한, 데이터 크기가 증가할수록 Rhipe가 ff+bigmemory를 사용하는 것보다 효율적임을 확인할 수 있고, Rhipe는 bigmemory에서 처리하지 못하는 데이터 크기에 대해서도 무난히 처리하는 것을 알 수 있다. 특기할 만한 사항으로는 Rhipe를 이용한 계산에서 데이터의 크기가 120M에서 480M까지 계산시간의 차이는 거의 없는 반면, 1.2G이상의 데이터를 처리하는 시간은 점점 증가 한다는 것을 알 수 있다. 통상적으로 Hadoop에서 task는 HDFS에 저장된 데이터를 약 64M로 분할하여 처리한다. 본 실험은 6개의 듀얼코어 PC를 클러스터로 구축하여 실행하였기 때문에 task tracker가 동시에 처리하는 task의 수는 최대 12개이다. 따라서 동시 처리가 가능한 최대 데이터의 크기는 64M*12=768M까지이다. Table 4.1에서 # tasks는 각 실험에서 task tracker에 의해 처리되는 task의 수를 의미한다. 여기서, 480M (<768M)까지의 작은 데이터를 처리하기 위해 task의 수는 동시처리가 가능한 최대 허용 task의 수인 12보다 작으며, 이들은 동시처리가 가능하다. 이 때문에 480M까지의 데이터를 이용한 분석시간이 크게 변하지 않게 된다. 반면, 데이터가 768M 보다 큰 경우에 job tracker는 12개의 task를 먼저 수행시키고 나머지 작업은 task가 완료된 컴퓨터 (task trackers)에 지시하기 때문에 위의 Table 4.1에서처럼 계산시간은 데이터의 크기에 비례하여 점점 느려진다.

5. 결론 및 향후과제

최근 스마트폰의 발전과 소셜미디어의 등장으로 인해 다양한 정보와 많은 양의 데이터가 수집되고 있고, 이렇게 다양하고 방대한 데이터를 처리하고 분석해야 하는 시대를 빅데이터 시대라고 한다. 빅데이터를 바라보는 견해는 이해관계에 따라 다양하게 제시되고 있지만 빅데이터 관련기술을 개발하기 위한 노력은 여러 분야의 협업을 통해 활발히 이루어지고 있다. 이와 더불어 HDFS와 MapReduce를 기반으로 구성된 Hadoop은 빅데이터의 저장 및 처리을 위한 표준 플랫폼으로 자리잡고 있고, Hadoop 플랫폼 하에서 빅데이터의 분석엔진으로 R이 주목받고 있다.

본 논문에서는 R 커뮤니티에서 제공하는 고성능 컴퓨팅 관련 R패키지, 빅데이터의 저장 및 처리를 위한 대표기술인 Hadoop 그리고 R과 Hadoop을 연동시켜주는 Rhipe에 대하여 소개하였다. 또한, Rhipe를 이용하여 병렬화 회귀분석을 구현하였고, 시뮬레이션을 통해 ff 및 bigmemory 패키지와 데이터 크기에 따른 연산속도를 비교하였다. 시뮬레이션 결과 전반적으로 ff 또는 bigmemory의 연산속도보다 Rhipe의 연산속도가 빠르다는 것을 알 수 있었고, 특히 데이터 크기가 증가할수록 Rhipe를 이용하는 것이 효율적임을 확인할 수 있었다.

빅데이터의 시대를 맞이하여 빅데이터의 저장 및 처리기술과 더불어 분석을 통한 가치창출이 중요시되고 있다. 하지만 기존의 분석 알고리즘은 한 대의 컴퓨터에서 소규모 데이터 분석을 위한 알고리즘으로 빅데이터를 분석하기에 어려움이 있다. 최근 빅데이터 분석을 위하여 Hadoop과 R을 이용한 연구 및 개발이 활발히 진행되고 있지만 다양한 분석들을 시도하기 위해서는 병렬알고리즘에 대한 연구와 프로그램 개발이 필요하다고 판단된다. 본 논문이 Hadoop관련 빅데이터 분석을 필요로 하는 연구자 및 개발자들에게 도움이 되기를 기대한다.

부록 A: Rhipe를 이용한 회귀분석 코드

```
# map expression 작성
Reg_map = expression({
  # HDFS의 sequence파일을 R의 matrix 오브젝트로 변환
  xx = do.call("rbind", map.values)
  Y = xx[,1]              # Y 벡터 생성
  X = cbind(1, xx[,-1])    # X 행렬  생성
  xtx = t(X)%*%X          # X^tX 계산
  xty = t(X)%*%Y         # X^tY 계산
  yty = as.numeric(t(Y)%*%Y)  # Y^tY 계산
  #"Regression"을 key, list(xtx, xty, yty)를 value로 하여 계산 결과를 Hadoop으로 전송
  rhcollect("Regresion", list(XtX = xtx, XtY = xty, YtY = yty))
})
# reduce expression 작성
Reg_reduce = expression(
 pre={XtX = 0; XtY = 0; YtY = 0;}, # 변수 초기화
 reduce={
  for(i in seq_along(reduce.values)){
    # map task에서 계산된 결과인 reduce.values를 이용하여
    # XtX, XtY, YtY의 누적 합 계산
    XtX = XtX + reduce.values[[i]]$XtX
    XtY = XtY + reduce.values[[i]]$XtY
    YtY = YtY + reduce.values[[i]]$YtY
  }
  # 회귀계수 추정
  N = XtX[1,1] # 전체 자료수의 계산
  SUM_Y2 = XtY[1,1]^2 # sum(y)^2
  XtX_inverse = solve(XtX) # (XtX)의 역행렬
  Estimate = XtX_inverse %*% XtY # 회귀계수
  # ANOVA TABLE
  SSE = as.numeric(YtY - t(Estimate)%*%XtY) # SSE
  SST = YtY - SUM_Y2/N # SST
  SSR = SST - SSE # SSR
  p = length(Estimate)-1 # SSR의 자유도
  DF = N-p-1 # SSE의 자유도
  MSR = SSR/p # MSR
  MSE = SSE/(DF) # MSE
  F_statistic = MSR/MSE # F 검정통계량
  P_value1 = 1 - pf(F_statistic, p, DF) # F값의 유의확률
  R_sqr = 1- SSE/SST # 결정계수
  R_adj = 1-((N-1)/(DF))*(1-R_sqr) # 수정된 결정계수
```

```
  # 회귀계수들에 대한 가설검정
  Std.Error = sqrt(diag(XtX_inverse * MSE)) # 표준 오차
  T_value = Estimate/Std.Error # T 검정통계량
  P_value = 2*pt(q=-abs(T_value), df=DF) # T값의 유의확률
  # 계산 결과를 리스트 형태로 변환
  OUT = list(
"Estimate"=Estimate,"Std.Error"=Std.Error, "T_value"=T_value,
"P_value_T"=P_value, "Residual_standard_error" =sqrt(MSE),
"degrees_of_freedom"=DF, "Multiple_R_squared"= R_sqr,
"Adjusted_R_squared" = R_adj, "F_statistic" = F_statistic,
"P"=p, "p-value_F-"= P_value1
    )
 },
# map에서 전달받은 key(Regression)와 reduce task에서
# 계산한 결과 오브젝트(OUT)를 value로 하여 Hadoop으로 전송
 post={ rhcollect(reduce.key, OUT) }
)

ifolder = "/user/stat/data_10GB"  # input data (분석할 데이터의 path)
ofolder = "/user/stat/data_10GB_reg_out"  # output data (분석 결과를 저장할 path)

# map/reduce를 실행하기 위하여 map/reduce  expression와 기타 정보를 포함한
# R오브젝트를 생성하는 함수, 여기서 만들어진 오브젝트는 rhex함수를 통하여 실행됨
z = rhmr(map = Reg_map,  # map expression
           reduce = Reg_reduce,# Reduce expression
           ifolder = ifolder,# 분석 할 파일(HDFS) path
           ofolder = ofolder,         # 분석 후 생성되는 파일(HDFS) path
       # HDFS에 저장되어 있는 파일 타입, 분석 후 HDFS에 저장 될 파일 타입
           inout = c("sequence", "sequence"),
           # reduce task에서 분할하여 작업할 task 수 지정
       mapred = list(mapred.reduce.tasks = 6),
       # Job name
           jobname = "Regression"
     )
# map/reduce 실행
jobid = rhex(z)
# mapreduce 처리상황 확인
rhstatus(jobid, mon.sec=5, autokill=TRUE, showErrors=TRUE)
# 계산결과 확인
out = rhread(ofolder, type="sequence")
```

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.

Ahn, C. and Hwang, S. (2012). Big data technologies and main issues. *Communications of the Korea Information Science Society*, **30**, 10-17.

Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, **51**, 107-113.

Duan, S. Wu, W., Wang, B. and Yang, J. (2011). Design and implementation of parallel statistical algorithm based on Hadoop's MapReduce model. *Proceedings of IEEE CCIS2011*, 134-138.

Eddelbuttel, D. (2012). CRAN task view: High-performance and parallel computing with R. http://cranr-projectorg/web/views/HighPerformanceComputing.html.

Gantz, J. and Reinsel, D. (2011). Extracting value from chaos. IDC, http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf.

Guha, S. (2010). Computing environment for the statistical analysis of large and complex data. Purdue University, West Lafayette, http://www.purdue.edu/policies/pages/teach_res_outreach/viii_3_1.html.

Han, K., Park, S., and Ahn, J. (2012). Development of a R function for visualizing statistical information on Google static maps. *Korean Data & Information Science Society*, **23**, 971-981.

Hedlund, B. (2011). Understanding Hadoop clusters and the network. http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/.

Jun, B., Kim, H., Choi, W. (2012). Future society and big data technologies. http://www.dbguide.net/upload/24/201206131339565739711l0.pdf.

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.

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. (2013). *A study on the development of R application for big data analysis*, Master's thesis, Dongguk University. Gyeongju, Korea.

Lee, K., Choi, H., Jung, Y. (2011). Massive data processing and management in cloud computing: A survey. *Journal of KISS: Databases*, **38**, 104-125.

Lumley, T. (2009). biglm: Bounded memory linear and generalized linear models. R package version 0.7.

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

Oracle (2012). Oracle: Big data for the enterprise. *Oracle White Paper*, http://www.oracle.com/us/products/database/big-data-for-enterprise-519135.pdf.

Park, Y. (2013). *A Study on large data analytics and parallel computing with R*, Master's theis, Dongguk University, Gyeongju, Korea.

Revolution (2011). Advanced big data analytics with R and Hadoop. *Revolution Analytics*, http://www.revolutionanalytics.com/why-revolution-r/whitepapers/R-and-Hadoop-Big-Data-Analytics.pdf.

Shvachko, K., Kuang, H., Radia, S. and Chansler, R. (2010). The Hadoop distributed file system. *IEEE 26th Symposium on Date of Conference*, 3-7.

White, T. (2012). *Hadoop: The definitive guide*, 3rd ed., O'Reily Media, Inc., Sebastopol, CA.

World Economic Forum (2012). Big data, big impact: New possibilities for international development. http://www3.weforum.org/docs/WEF_TC_MFS_BigDataBigImpact_Briefing_2012.pdf.

Zikopoulos, P., Eaton, C., Roos, de D., Deutsch, T. and Lapis, G. (2012). *Understanding big data: Analytics for enterprise class Hadoop and streaming data*, McGraw-Hill, http://public.dhe.ibm.com/common/ssi/ecm/en/iml14296usen/IML14296USEN.PDF.

Journal of the Korean Data & Information Science Society
2013, **24**(5), 975–987

http://dx.doi.org/10.7465/jkdi.2013.24.5.975
한국데이터정보과학회지

Analysis of big data using Rhipe[†]

Youngjun Ko[1] · Jinseog Kim[2]

[1]Digital Convergence Center, Jeju Technopark
[2]Department of Statistics and Information Science, Dongguk University

Received 26 June 2013, revised 21 July 2013, accepted 3 August 2013

Abstract

The Hadoop system was developed by the Apache foundation based on GFS and MapReduce technologies of Google. Many modern systems for managing and processing the big data have been developing based on the Hadoop because the Hadoop was designed for scalability and distributed computing. The R software has been considered as a well-suited analytic tool in the Hadoop based systems because the R is flexible to other languages and has many libraries for complex analyses. We introduced Rhipe which is a R package supporting MapReduce programming easily under the Hadoop system, and implemented a MapReduce program using Rhipe for multiple regression especially. In addition, we compared the computing speeds of our program with the other packages (ff and bigmemory) for processing the large data. The simulation results showed that our program was more fast than ff and bigmemory as the size of data increases.

Keywords: Big data, Hadoop, HDFS, MapReduce.

[†] This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2010-0023302).
[1] Associate researcher, Digital convergence center, Jeju Technopark, Jeju 690-787, Korea.
[2] Corresponding author: Assistant professor, Department of Statistics and Information Science, Dongguk University, Gyeongbuk 780-714, Korea. E-mail: jinseog.kim@gmail.com