http://dx.doi.org/10.5392/JKCA.2012.12.10.001

전문용어 인식 시스템을 위한 분산 병렬 처리 플랫폼 최적화 및 성능평가

Optimization and Performance Analysis of Distributed Parallel Processing Platform for Terminology Recognition System

최윤수, 이원구, 이민호, 최동훈, 윤화묵, 송사광, 정한민
한국과학기술정보연구원 소프트웨어 연구실

Yunsoo Choi(armian@kisti.re.kr), Wongoo Lee(wglee@kisti.re.kr),
Minho Lee(cokeman@kisti.re.kr), Dong-Hoon Choi(choid@kisti.re.kr),
Hwamook Yoon(hmyoon@kisti.re.kr), Sa-kwang Song(esmallj@kisti.re.kr),
Hanmin Jung(jhm@kisti.re.kr)

요약

과학기술 문헌의 전문용어 인식 분야는 지금까지 다양한 통계적 방법론을 사용하여 용어 인식 정확률을 향상시키기 위하여 연구되어 왔다. 하지만 기존의 연구는 단일-코어 또는 단일 머신 상에서 수행되었기 때문에, 폭발적으로 증가하는 문헌들에 대한 실시간 분석 요구를 처리할 수 없는 상황에 직면하고 있다. 본 논문에서는 전문용어를 인식하는 과정에서 병목현상이 발생하는 작업을 '후보용어 추출 과정'의 언어처리 부분과 '용어 가중치 할당 과정'에서 통계정보를 취합하는 부분으로 분류하고, 각 작업을 분산병렬 처리 기반의 맵리듀스 작업을 이용하여 해결하는 전문용어 인식 방법을 구현하고 실험하였다. 실험은 확장성과 분산 병렬 처리 환경 최적화 두 가지로 수행하였고, 첫 번째 실험에서 12개의 노드를 사용하여 분산 병렬 처리하였을 때 단일 머신을 사용한 경우보다 11.27배의 처리속도 향상을 보였다. 두 번째 실험에서 1)기본 환경, 2)복수 리듀서, 3)컴바이너, 4) 2)와3)의 조합에 대하여 수행하였고, 3)컴바이너 사용이 가장 우수한 성능을 보여 주었다. 본 논문에서 구현된 전문용어 인식 시스템은 대용량 과학기술 문헌에 대한 지식 추출 작업 속도 개선에 기여하였다.

■ 중심어 : | 전문용어인식 | 분산병렬처리|최적화 | 하둡 | 맵리듀스 |

Abstract

Many statistical methods have been adapted for terminology recognition to improve its accuracy. However, since previous studies have been carried out in a single core or a single machine, they have difficulties in real-time analysing explosively increasing documents. In this study, the task where bottlenecks occur in the process of terminology recognition is classified into linguistic processing in the process of 'candidate terminology extraction' and collection of statistical information in the process of 'terminology weight assignment'. A terminology recognition system is implemented and experimented to address each task by means of the distributed parallel processing-based MapReduce. The experiments were performed in two ways; the first experiment result revealed that distributed parallel processing by means of 12 nodes improves processing speed by 11.27 times as compared to the case of using a single machine and the second experiment was carried out on 1) default environment, 2) multiple reducers, 3) combiner, and 4) the combination of 2)and 3), and the use of 3) showed the best performance. Our terminology recognition system contributes to speed up knowledge extraction of large scale science and technology documents.

■ keyword : | Terminology Recognition | Distributed Parallel Processing | Hadoop | MapReduce |

접수번호 : #120907-003
접수일자 : 2012년 09월 07일
심사완료일 : 2012년 10월 18일
교신저자 : 윤화묵, e-mail : armian@kisti.re.kr

I. 서 론

과학기술 문헌에 대한 전문용어(이하, 전문용어) 인식에 관한 연구는 정보추출(Information Extraction), 텍스트마이닝(TextMining), 질의응답(Q&A)분야 등의 기반이 되는 연구로서, 특히 생의학 분야에서 많은 연구가 수행되어 왔다. 지금까지 전문용어 인식에 관한 연구는 다양한 통계적 방법론을 사용하여 전문용어 인식률의 성능향상에 목적을 두고 수행되어 왔으며 상당한 수준의 인식률 향상을 보여 주고 있다.

F. Smadia et. al.(1996)[1]은 병렬 코퍼스내에서 용어들의 응집도(Unithood)를 계산하기 위하여 Dice Coefficient를 사용하였고, K. Frantzi et. al.(2000)[2]은 다어절 용어 후보에서 내포된 부분 문자열의 전문성을 측정하기 위해 C-value를 제안하였고, S. K. Song(2011)[3]은 웹 검색 정보를 이용하여 용어 후보에 가중치를 부여하는 방법을 제안하였다.

지금까지는 전문용어 처리에 대한 실시간 처리 요구가 많지 않고, 처리해야 할 데이터의 양이 비교적 적은 관계로 인식속도(또는 인식 처리율)에 대한 연구가 상대적으로 더디게 진행되어 왔다. 하지만, 빅데이터 시대를 맞이하여 처리할 수 없을 정도로 쏟아져 나오는 데이터에 대해 전문용어를 인식한다는 것은 상당히 어려운 일이며, 더욱이 실시간으로 전문용어를 추출한다는 것은 상상하기 힘든 상황에 이르렀다.

이에, 전문용어 처리에 대한 많은 연구가 그 처리 속도의 향상에 초점을 맞추고 있으나, 기존의 전문용어 인식 기술 및 연구가 단일 코어 상에서 수행된다는 제약으로 인해 커다란 인식속도 향상을 기대하기는 어렵다. 따라서 최근에는 단일 코어가 아닌, 독립적이고 병렬적인 처리가 가능한 요소에 대해 다수의 장비를 이용한 분산・병렬처리를 통해 인식 속도 향상을 꾀하고자 하는 방향으로 전문용어 인식 연구가 진행되고 있다.

이와 관련하여 전문용어 인식 과정에서 많은 시간이 소모되는 구문분석을 다중 서버에 분산하여 처리하는 서버/클라이언트 방식에 대한 연구와[4], 전문용어 인식 과정을 그리드 컴퓨팅을 이용하여 분산 처리하는 방법이 제안되었다[5].

본 논문에서는 단일코어 방식의 처리 문제를 해결하고자 최근 분산・병렬 처리 플랫폼으로 많은 관심을 받고 있는 하둡(Hadoop) -분산 파일 시스템(HDFS)과 맵리듀스(MapReduce)- 을 이용하여 전문용어 인식을 위한 최적화된 분산 환경을 구축하고, 이를 기반으로 전문용어 인식 시스템을 개발하였다. 이를 통해, 거대한 양의 과학기술 빅데이터로부터 근실시간적으로 전문용어를 추출할 수 있었고, 또한 유사한 분야에 활용할 수 있는 분산・병렬처리 인프라의 근간을 마련할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서 전문용어 인식방법과 관련된 연구들과 분산・병렬처리 기법인 하둡(Hadoop)을 살펴보고, 3장에서 하둡 플랫폼(맵리듀스)을 이용하여 전문용어 인식 시스템을 설계한다. 4장에서는 제안된 시스템에 대한 확장성을 살펴보고, 환경설정 튜닝 실험을 통하여 최적의 환경을 찾아낸다. 마지막으로 5장에서 결론 및 향후연구에 대하여 논의한다.

II. 관련연구

1. 전문용어 인식방법

전문용어 자동 인식방법은 후보 용어들을 추출하는 '후보 용어 추출(candidate term extraction)'단계와 '용어 가중치 할당(termhood assignment)'단계로 구분된다.

표 1. 후보 용어 추출 패턴 예

저자	추출 패턴	비고
Daille (1994)	(형용사)+(명사)+ (명사)(명사)+	closed filter
Justeson(1995)	명사구+전치사구	open filter
Franzi(2000)	명사구+전치사구	open filter

후보 용어 추출 작업은 일반적으로 문장내의 품사 패턴을 이용하여 추출한다. [표 1]은 후보 용어 추출 패턴 방식을 보여준다. 제한적인 필터(closed filter)를 사용하면 추출된 후보들의 정확률(precision)이 높아지는 반면, 재현율(recall)은 감소하게 된다[6]. 개방적인 필터(open filter)를 사용하면 반대로 재현율이 높아지고 정확률이 감소한다[7].

후보 용어 추출 패턴을 사용하여 추출된 각 후보 용

어에 가중치를 할당하는 출현 빈도 통계 기반 연구로는, 다이스 상관계수(Dice Coefficient)[1], 상호정보(Mutual Information)[8], 구글 정규거리(Normalized Google Distance)[9]등이 있고, 출현 빈도 기반의 가중치 부여 방식에 대한 문제점을 개선하기 위해, 다어절 후보 용어에서 내포된 부분 문자열의 전문성을 측정하기 위해 C-value가 제안되었다[2].

$$C-value(a)=\begin{cases}\log_2|a| \cdot f(a) & |a|=\max \\ \log_2|a| \cdot (f(a)-\frac{1}{P(T_a)}\sum_{b\in T_a} f(b)) & otherwise\end{cases} \quad (1)$$

식(1)은 C-value의 수식을 보여준다. 여기에서 a 는 후보 문자열, $f(a)$는 문서집합 내에서 a의 출현빈도, $|a|$는 문자열의 길이(공백으로 구분되는 단어 수), $f(b)$는 a를 내포하는 후보 문자열의 출현빈도, T_a는 a를 포함하는 용어집합, $P(T_a)$는 T_a에 포함되는 용어의 수이다

예를 들어, “soft contact lens″라는 실제 안과학의 전문용어에 대해 살펴보면, ”soft contact″과 “contact lens″의 두 부분문자열도 전문용어의 후보로 추출되는데, 이 때 ”contact lens″는 단독으로 자주 출현하거나 다른 용어들에 내포되어 출현하는 횟수가 많기 때문에 높은 가중치를 받는 반면, “soft contact″은 ”soft contact lens″외의 다른 용어에 내포되어 출현하는 횟수가 거의 없기 때문에 낮은 가중치를 할당받는다.

2. 분산·병렬 처리기법

분산・병렬 처리 기법은 구글(Google)의 GFS (Google File System)과 맵리듀스(MapReduce)가 있으며[10], 이를 바탕으로 아파치에서 오픈 소스 기반의 하둡 프로젝트를 진행 중이다. 하둡(Hadoop)은 대규모 데이터 처리 분석을 위한 소프트웨어 프레임워크를 제공하는 클라우드 컴퓨팅의 대표적인 기술로서 널리 사용되고 있다. 특히, 하둡은 높은 확장성, 결함탐지, 자동복구 기능이 우수하여 과학기술 분야에서도 도입 및 활용이 증가하는 추세이다.

하둡은 크게 분산 파일 시스템인 HDFS와 분산・병렬 처리 기법인 맵리듀스로 구성된다[11]. HDFS는 [그림 1]과 같이 마스터 노드(Master Node)와 다수의 슬레이브 노드(Slave Node)로 구성된다. 마스터 노드는 분산 파일 시스템의 네임 스페이스를 관리하고 클라이언트의 파일에 대한 접근을 통제하는 네임 노드(Name Node)와 주어진 작업에 대한 스케쥴링을 수행하는 잡 트래커(Job Tracker)로 구성된다.

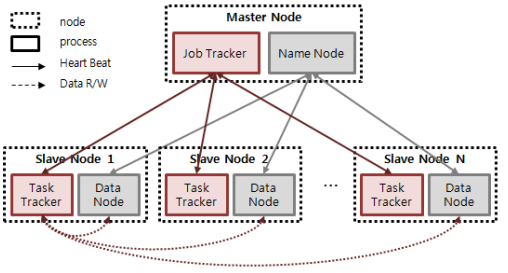


그림 1. 하둡 분산 파일시스템

슬레이브 노드는 노드상의 스토리지를 관리하는 데이터 노드와 잡트래커에 의해 할당된 단위 작업을 수행하는 태스크 트래커(Task Tracker)로 구성된다. HDFS는 사용자의 데이터를 파일로 저장하며, 내부적으로 하나의 파일은 여러 개의 블록으로 분리되어 저장된다.

맵리듀스는 key/value쌍 기반의 데이터 분산・병렬 처리 모델이다. 이는 대규모 분산・병렬 처리에 있어 데이터 증가에 따른 확장성을 제공하고, 노드 간 데이터 이동에 따른 네트워크 트래픽을 최소화하기 위해 구현되었다.

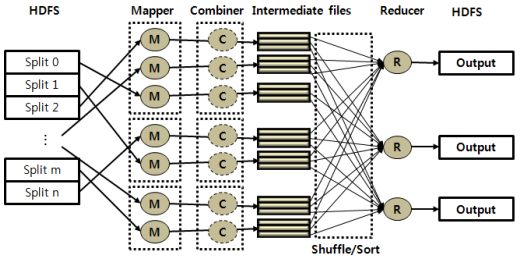


그림 2. MapReduce 과정

[그림 2]와 같이 입력 데이터 소스를 기반으로 맵 태스크를 수행하여 key/value로 구성된 중간 결과를 생성

한다. 이 중간 결과물은 key값에 의해 그룹화되어 리듀스 태스크에 전달된다. 리듀스 태스크는 모든 중간 key 값들을 통합하여 최종 결과물을 내보내게 된다.

맵 태스크에서 수행한 결과에 대한 중간 결과물 생성 시, 컴바이너 사용이 설정되어 있는 경우, 네트워크를 통해 리듀서로 전달되는 중간 결과물의 양을 축소하여 리듀서의 작업시간을 단축시키는 효과를 볼 수 있다. 컴바이너를 사용하도록 설정하였다고 해서 항상 컴바이너가 동작하지는 않는다. 맵 태스크의 중간결과물이 일정 크기 이상 생성되었을 때 동작하도록 되어 있으며, 일반적으로 리듀서와 동일한 역할을 맵 태스크 측에서 미리 수행한다.

III. 하둡 기반 전문용어 인식 시스템

과학기술 문헌으로부터 전문용어를 인식하는 과정에서 가장 많은 시간이 소모되는 작업을 다음과 같이 구분하였다.

작업1) "후보용어 추출 과정"에서 사용하는 언어처리 부분 ([그림 3]의 단계 1)

작업2) "용어 가중치 할당 과정"에서는 추출된 후보용어들의 통계정보를 취합하는 부분 ([그림 3]의 단계 2)

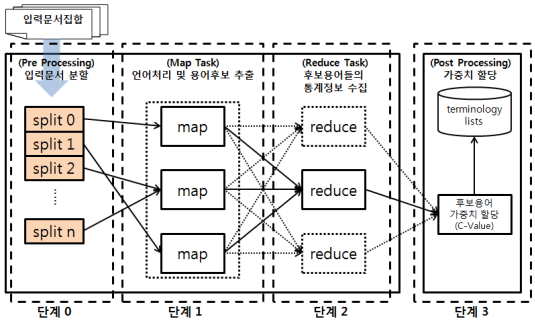


그림 3. 하둡 기반 전문용어 인식 시스템

본 논문에서는 [그림 3]과 같이 맵 과정을 사용하여 작업1)을 분산・병렬 처리하고, 리듀스 과정에서 작업2)를 처리하도록 대용량 과학기술 문헌에 대한 전문용어 인식 시스템을 설계한다. [그림 3]의 단계0은 입력문서 전처리 과정, 단계1과 단계2는 맵리듀스를 이용하여 분산・병렬처리를 수행하는 과정, 단계3은 후보용어에 가중치를 할당하는 과정이다. 여기에서 단계0과 단계3은 분산환경이 아닌 단일 코어[1]에서 동작한다.

1. 입력 문서 처리 (단계 0, 전처리)

단일 프로세서를 이용하여 전문용어를 인식하는 경우에는 입력 문서에 대하여 특별한 처리를 수행할 필요가 없지만, 하둡 기반의 작업에서는 하둡의 특성을 고려하여 입력 문서를 전처리 해야한다. 하둡은 입력 문서를 스플릿(split)이라는 단위로 분할하여 맵 작업을 할당한다. 하둡에서 사용하는 스플릿의 단위는 HDFS에서 디폴트로 사용하는 블록의 크기(64M)와 동일하다.

본 논문에서 사용된 문서 집합은 과학기술 문헌 중 논문 데이터로서, 한 문서가 하나의 파일로 이루어져 있고 평균 1.87KB이다. 이렇게 작은 문서를 하둡을 이용하여 처리할 경우, 처리 시간이 매우 증가하게 된다.

하둡에서 처리하기에 적합한 입력 문서의 크기를 선정하기 위하여, 대상 문서를 통합하여 1M부터 128M까지의 규모로 가공하여 맵리듀스 작업에서 소요되는 시간을 측정해 보았다. 실험을 위해 작성된 맵리듀스 작업은 입력 문서에 대한 문장 분리 모듈을 사용하였다.

표 2. 입력문서크기와 맵리듀스 실행 시간

입력문서 (MB)	map 실행시간(초)	reduce 실행시간(초)	전체시간 (초)
1	493	54	547
2	266	54	320
4	186	75	261
8	130	60	190
16	77	64	141
32	51	73	124
64	56	55	111
128	54	53	107

[표 2]는 입력 문서의 크기에 따라 측정된 맵리듀스 실행 시간을 보여준다. 맵 시간은 입력 문서의 크기에 따라 순차적으로 감소하다가 디폴트 블록크기인 64M 근처에서 일정한 속도를 보여주고 있고, 리듀스 시간은

1) 마스터 노드에서 단계0과 단계3이 수행 된다.

입력 문서의 크기와 상관없이 전체적으로 일정한 속도를 보여주고 있다. 이 결과를 바탕으로 본 논문에서 사용되는 입력 문서의 크기를 HDFS의 블록크기와 동일한 64MB로 결정하였다.

2. 전문용어 후보 추출 모듈 (단계 1, 맵 태스크)

입력된 문서에서 전문용어 후보를 추출하는 전체 작업은 맵리듀스의 맵퍼(mapper)에 의해 수행된다. 각 맵퍼는 스플릿(split) 단위로 입력되는 문서 묶음으로부터 개별 문서를 분리하여 작업을 수행한다.

표 3. 후보 용어 추출 패턴

패턴	후보용어 패턴(정규표현식)	예
0	(JJ+|VB[G|N+)+(NN[S|P]?+)+	(형용사)+(명사)+
1	(NN[S|P])?(NN[S|P])+	(명사)(명사)+

전문용어 후보 추출 과정은 입력된 문서를 문장으로 분리하는 작업으로 시작한다. 다음으로 품사태거를 이용하여 각 문장을 구성하는 단어들에 대한 품사정보를 획득한다. 품사정보로부터 최초 후보용어를 추출하는 과정은 [표 4]와 같다.

표 4. 후보 용어 추출 과정

단계	과정
1	[표 3]의 후보 용어 추출 패턴을 이용하여, 이를 만족하는 최장 후보 용어를 추출한다.
	예) Distributed File System
2	추출된 후보 용어를 정규화하여 정규화된 후보 용어[2]를 생성한다.
	예) distributed file system
3	후보 용어가 3어절 이상인 경우, 이 후보 용어를 구성하는 부분 후보 용어들을 추가적으로 추출한다.
	예) "distributed file", "file system", "distributed file system"

[표 4]에서, 1, 2단계에서 추출된 최장 후보 용어 "distributed file system"은 3어절 이상이므로, 최종적으로 "distributed file", "file system", "distributed file system" 3개의 부분 후보 용어들이 추출된다. 맵 태스크의 최종 출력에서 key값은 이 부분 후보 용어와 이들을 포함하는 최장 후보 용어를 이용하여 [표 5]와 같이 구성된다.

표 5. key/value 쌍 구성

key	value
distributed file@***distributed file system***	1
file system@***distributed file system***	1
distributed file system@***distributed file system***	1

식(1)의 C-value를 구하기 위해서는 부분 용어 후보의 전체 빈도수와 부분 용어 후보를 포함하는 최장 용어 후보들에 대해 단독으로 출현한 빈도수가 필요하다. 이를 위해, 맵퍼는 부분 용어 후보들을 key부분에 "부분 용어 후보@최장 용어 후보"를 할당하고 value 부분에 1을 할당하여, 부분 용어 후보와 최장 용어 후보에 대한 빈도수를 유지하도록 한다.

3. 전문용어 인식 모듈(단계 2, 리듀스 태스크)

전문용어 인식 모듈은 맵 과정에서 출력된 key/value 쌍을 통합하는 리듀스 과정과, 리듀스 과정에서 출력된 최종 출력물을 이용하여 용어 후보에 대한 가중치를 할당하는 과정으로 나누어진다. 맵 과정에서 출력된 key/value쌍은 key값에 의해 정렬된다. 동일한 key에 대한 vlaue들은 목록화되어 특정 리듀서[3]에 전달된다. 리듀서는 [표 6]과 같이 입력된 value목록을 통합하여, 새로운 key/value 쌍을 출력한다.

표 6. 리듀서부분 입력 및 출력

입력	file system@***file system***, [1,1,1,...,1] file system@***distributed file system***, [1,1,1 ....,1] file system@***file system api***, [1,1,1,1,1,1]
출력	file system@***file system***, 10 file system@***distributed file system***, 25 file system@***file system api***, 6

[표 6]에서 리듀서의 출력은 key는 "부분 용어 후보@최장 용어 후보"이고, value는 이 key에 대한 빈도수 이

2) 본 논문에서 후보용어는 정규화된 후보용어를 지칭한다.

3) 리듀서가 2개 이상인 경우에 해당한다. 기본환경설정에서는 리듀서가 1개 할당되므로, 맵 태스크의 모든 출력은 하나의 리듀서로 전달된다

다. 이 정보로부터, 용어 후보 "file system"이 단독으로 출현한 횟수는 10, 최장 용어 후보의 부분 용어 후보로 출현한 횟수는 31(25+6), 전체 출현 빈도수는 41, 그리고 자신을 내포하는 최장 용어 후보의 수가 2임을 구할 수 있다. [그림 3]의 단계3에서 이 정보를 식(1)에 적용하여 각 후보용어에 대한 가중치를 할당한다.

IV. 실험

본 논문에서 분산・병렬 처리 기반으로 구현된 전문용어 인식 시스템의 성능을 측정하기 위하여 실험을 2단계로 분리하였다. 첫 번째 실험은 '확장성 실험'으로 분산・병렬 처리를 위한 노드 수의 증가에 따른 확장성을 알아보기 위하여 진행되었고, 두 번째 실험은 '환경설정 튜닝 실험'으로 노드 수를 고정한 상태에서 환경설정에 따른 성능을 비교하였다. 첫 번째 실험은 1절에서, 두 번째 실험은 2절에서 나누어 설명한다.

1. 확장성 실험 (실험1)

본 실험을 위해 과학기술 문헌 중 해외 논문 데이터 초록을 수집하여 20만 건의 문서 집합을 구축하였고, 평균 크기는 1.87KB 였다. 실험에 사용된 클러스터는 [표 7]과 같이 1대의 마스터 노드와 12개의 슬레이브 노드로 구성하였고, 하둡의 기본 환경 설정을 이용하였다.

표 7. 실험에 사용된 클러스터

	CPU(GHz)	MM(GB)	HDD(GB)
Master	2.4(16Core)	16	250
Slaves	2.4(8Core)	8	120
Hadoop	Ver 0.20.203 (기본환경 설정)		

본 논문의 시스템을 이용하여 20만 건의 문서 집합으로부터 전체 409,918개의 최장 후보 용어가 추출되었고, 이 후보 용어로부터 2,413,371개의 부분 후보 용어가 파생되었다. [표 8]은 가장 높은 가중치(C-value)를 할당받은 5개의 후보를 보여준다. [표 8]에서 빈도수는 용어가 출현한 전체 빈도수이고, 긴용어수는 후보 용어가 부분문자열로 내포되어 출현한 최장 후보 용어들의 수이다.

표 8. 후보 용어 Top-5

용어	C-value	빈도수	긴용어수
control system	7,652.23	3,189	1,186
neural network	6,675.59	2,846	870
sensor network	6,687.07	3,429	601
web service	5,891.68	3,523	715
genetic algorithm	3,617.12	2,341	330

본 실험은 1개의 슬레이브 노드를 이용하여 경과된 시간을 베이스라인으로 하고, 슬레이브 노드 수를 늘려가면서 전체적인 수행시간을 측정하였다. [표 9]는 노드 수의 증가에 따른 맵 태스크 수행 시간, 리듀스 태스크 수행 시간, 전체 수행 시간과 확장성을 보여준다.

표 9. 노드 증가에 따른 수행 시간

노드수	Map (sec)	Reduce (sec)	전체 (sec)	확장성 (노드수*전체)
1	108,334	39	108,378	108,378 (1.000)
3	36,174	35	36.193	108,579 (0.998)
6	18,155	34	18,189	109,134 (0.993)
9	12.327	29	12.356	111,204 (0.975)
12	9.616	32	9,648	115,776 (0.936)

맵 태스크 수행 시간은 노드 수의 증가와 비례하여 처리 속도가 빨라지는 반면, 리듀스 태스크 수행 시간은 노드 수와 상관없이 일정함을 보여주고 있다. 이는 하둡 기본 환경이 리듀서 개수를 1개로 설정함으로 인해, 모든 맵 태스크가 종료한 이후에 맵 태스크의 출력이 리듀스 태스크의 입력으로 사용되기 때문이다. 1개 노드를 사용하여 실행된 수행 시간을 베이스라인으로 하고, 12개의 노드를 사용한 분산・병렬 환경에서의 수행 시간을 비교하면, 약 11.27배의 향상된 처리 속도를 보여준다.

전체 성능은 추가되는 노드에 비례해야 하고, 그 비

율이 1이 되었을 때, 완전하게 확장성을 가진다고 말할 수 있다. 즉, 각 노드 별 실험에서 “노드 수 * 수행시간”의 결과가 일정해야 함을 의미한다. 그러나, 병렬 처리되지 않는 부분으로 인해 확장성이 노드 수에 따라 완전하게 비례하진 않는다. 이 시스템에서는 맵 태스크의 결과가 네트워크상에서 리듀스 태스크로 전달되는 작업과 리듀스 태스크 작업이 병렬 처리되지 않는 부분이다.[4)]

[표 9]의 확장성 항목을 살펴보면, 1개 노드를 사용하였을 때와 비교하여 12개의 노드가 추가된 경우에 그 비율이 0.936으로 1에 가까운 값을 보여준다. 이는 새로이 추가되는 노드에 대해서, 본 논문에서 제안한 전문용어 인식 시스템이 유연한 확장성을 가지고 있음을 보여준다.

2. 환경설정 튜닝 실험 (실험2)

본 실험에서는 노드수를 고정한 상태에서, 하둡의 기본 환경과 컴바이너, 복수개의 리듀서를 사용하였을 경우의 수행 속도를 비교한다. 본 실험의 목적은 맵 태스크에서 생성된 중간 결과물의 처리 작업과 맵 태스크의 최종 결과물이 리듀스 태스크로 전달되는 전송 작업, 그리고 리듀스 태스크에서의 처리 작업에 대한 수행 속도를 각각의 환경 설정에 따라 비교하여 최적의 환경 설정을 획득하기 위함이다.

표 10. 실험데이터 구축 현황

집합	용어수	파일 수
1	20,000,000	20
2	40,000,000	40
3	60,000,000	60
...	....	....
15	300,000,000	300
16	320,000,000	320
17	340,000,000	340

실험1에서 살펴보았을 때, 전체 속도에서 맵 태스크의 수행 속도가 대부분을 차지하고 있다. 본 실험에서는 맵 태스크에서 내부적으로 수행하는 시간을 제외하고, 중간 결과물과 최종 결과물의 처리 속도에 집중하기 위하여, 실험1의 맵 태스크의 최종 결과물을 입력으로 처리하는 맵 태스크를 새로이 구현하였다.

실험1의 맵 태스크에서 출력된 key 중에서 2,000,000개의 독립적인 후보 용어를 이용하여 [표 10]과 같이 실험 데이터를 구축하였다.[5)] 독립적인 2,000,000개의 후보 용어가 각각 10회 출현빈도를 갖도록 20,000,000개의 후보 용어를 20개의 파일로 분리하여 최초 집합을 구성하였고, 이 집합을 계속 복사하여 다음 집합들을 구성하였다. 각각의 집합은 전체 용어수가 20,000,000개씩 증가하지만, 독립적인 용어수는 항상 2,000,000개로 고정되어 있다.

실험2를 위해 실험1과 달리 10개의 슬레이브 노드만을 사용하였다. 실험1에서는 최대 12개의 슬레이브 노드를 사용하였지만, 본 실험에서 사용되는 실험 집합의 파일 수가 모두 10의 배수이고, 맵 태스크 당 1개의 파일을 처리하기 때문에 10개의 노드만을 고정하여 사용하였다.

실험은 (환경 1, default)하둡의 기본 환경, (환경 2, reducer) 복수 리듀서 사용[6)], (환경 3, combiner) 컴바이너 사용, (환경 4, both) 복수 리듀서와 컴바이너 사용, 이렇게 4가지 환경에서 17개의 실험 집합에 대하여 수행하였다.

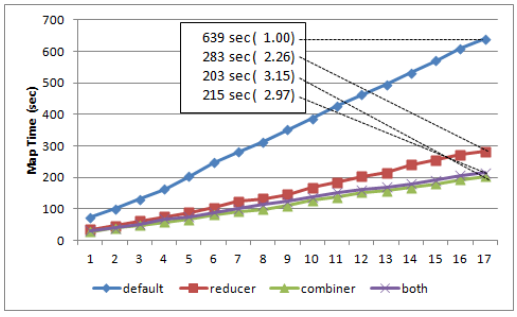


그림 4. 맵 태스크 수행시간

4) 하둡 환경을 조정하여 리듀서 개수를 복수개로 설정하면 분산・병렬 처리 작업이 된다.

5) 최초 실험에서 실험집합을 20개로 구성하였으나, 18번째 집합에 대한 실험에서, 리듀서 작업이 중간에 중지되는 현상이 발생하여 17개로 선정하였다.

6) 본 실험에서는 슬레이브 노드 개수 만큼인 10개의 리듀서를 사용하였다.

[그림 4]는 맵 태스크의 수행 시간을 보여준다. 맵 태스크는 입력 받은 후보 용어에 대하여 출현 빈도 1을 추가하여 출력하는 단순한 작업을 수행한다. 그러므로, 4가지 환경에서 입력되는 데이터의 집합의 규모에 비례하여 수행 시간이 선형적으로 증가함을 보여 주고 있다.

맵 태스크의 중간 결과물은 리듀스 태스크로 전달되기 전에 key값에 의해 정렬되어 최종 결과물을 생성하게 된다.

(환경 1, default)에서는 리듀서가 1개만 사용되므로, 맵 태스크의 중간 결과물 전체가 정렬되어야 하기 때문에 가장 낮은 성능을 보여준다.

(환경 2, reducer)에서는 리듀서 개수를 복수로 설정하였기 때문에, 각 리듀서별로 전달되는 중간 결과물이 분산/병렬되어 정렬되므로, (환경 1)에 비해 2.26배의 성능 향상을 보여준다.

(환경 3, combiner)은 맵 태스크의 중간 결과물이 리듀스 태스크로 전달되기 전에 축소시키는 작업을 미리 수행한 후 정렬되므로, (환경 1)에 비해 3.15배의 성능 향상을 보여준다.

(환경4, both)는 (환경2)와 (환경3)을 조합한 방법으로, (환경1)에 비해 2.97배의 성능 향상을 보여준다.

(환경4)보다 (환경3) 단독으로 사용하였을 때, 가장 좋은 성능을 보여주는 이유는 리듀서를 복수개 사용하여 중간 결과물을 분산/병렬 정렬하는 방법보다, 컴바이너만 사용하여 중간 결과물을 축소하는 방법이 더 효과적이기 때문이다.

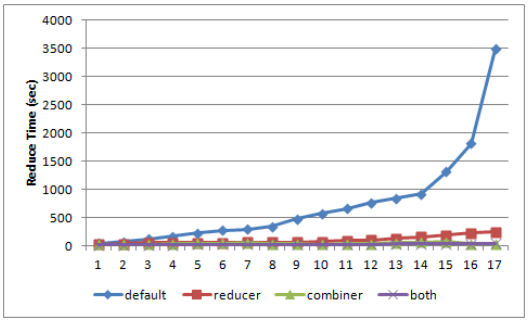


그림 5. 리듀스 태스크 수행시간

[그림 5]는 4가지 환경에서 리듀스 태스크 수행시간을 보여준다. (환경1)에서 15번째 집합부터 처리 속도가 급격히 떨어짐을 보여주고 있다. 이는 정렬될 맵 태스크의 중간 결과물의 양이 메모리에서 처리하기 어려울 정도로 증가하여 페이징이 계속 발생하였음을 의미한다.

[그림 6]은 (환경1)을 제외한 나머지 환경들에 대한 리듀스 태스크 수행시간을 보여준다. (환경2)는 실험 데이터의 규모에 비례하여 처리 시간 또한 증가됨을 보여주는 반면, (환경3)과 (환경4)는 실험 데이터 규모와 무관하게 일정한 처리 속도를 보여준다.

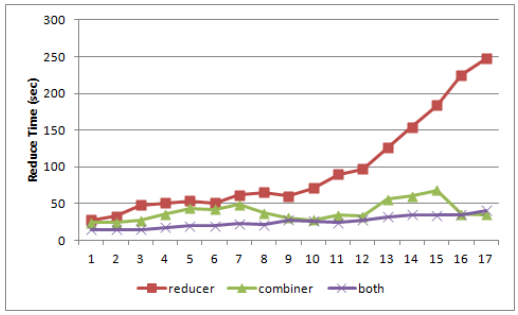


그림 6. 리듀스 태스크 수행시간 (환경1 제외)

이는 본 실험에 사용된 실험 데이터는 규모와 상관없이 독립적인 20,000,000개의 키로 구성되어 있으므로, (환경3)과(환경4)처럼 컴바이너가 사용되었을 때, 맵 태스크의 최종 결과물이 항상 일정한 규모로 생성되어 리듀서 태스크로 전달되기 때문이다.

(환경1)에서, 데이터가 특정 규모보다 커졌을 때, 처리 속도가 급격히 증가하는 현상처럼, (환경2)도 데이터의 규모가 더 커진다면, 동일한 현상이 발생할 것으로 이 실험에서 알 수 있다. 반면, (환경3)과 (환경4)는 데이터내의 독립적인 key의 개수가 일정하다면 데이터의 규모와 상관없이 일정한 처리 속도를 보장함을 알 수 있다.

[그림 7]은 전체 수행 시간을 보여준다. 맵 태스크 수행시간, 리듀스 태스크 수행 시간과 비교하여 보았을 때, 전체적인 성능은 리듀스 태스크의 성능에 종속됨을 알 수 있다.

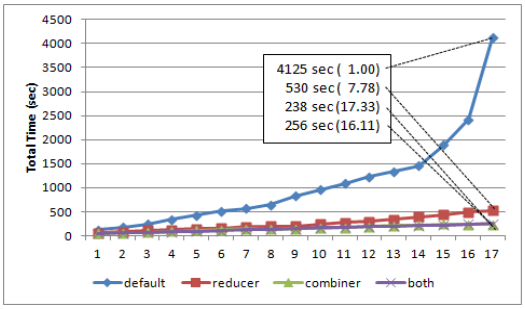


그림 7. 전체 수행시간

17번째 데이터 집합에서 (환경2)는 (환경1)의 7.78배, (환경3)은 17.33배, (환경4)는 16.11배의 성능을 보여주고 있으며, 컴바이너만 단독으로 사용한 (환경3)이 컴바이너와 복수개의 리듀서를 동시에 사용한 (환경4)보다 더 좋은 성능을 보여줌을 알 수 있다. 이는 리듀서의 개수를 확장하여 분산・병렬 처리하는 방법보다, 컴바이너를 사용하여 리듀서로 전달되는 최종 결과물의 양을 줄이는 방법이 노드 간 데이터 전송량과 리듀서 작업량을 모두 감소시키므로 가장 효과적임을 보여준다.

V. 결론

본 논문에서는 대용량 과학기술문서로부터 전문용어를 추출하기 위한 분산・병렬 처리 기반 시스템을 구현하고 성능을 평가하였다.

실험1에서 단일 시스템에서 수행하였을 때와 비교하여 11.27배의 처리 속도 향상을 보여주었고, 노드수의 증가에 따라 성능도 일정하게 향상됨을 보여 주었다. 이는 본 논문에서 제안된 전문용어 인식 시스템이 분산・병렬 환경에 적합하게 설계 되었음을 보여준다.

실험2에서 복수개의 리듀서와 컴바이너의 사용과 관련한 최적화 작업에서, 단독으로 컴바이너를 사용한 경우, 하둡 기본 환경 설정에서 수행된 결과보다 최대 17.33배의 성능을 보여주었다.

향후 연구로는 용어들 간의 관계를 추출하는 용어간 관계추출 시스템에 대한 분산・병렬 처리 작업에 대한 연구를 통해, 본 논문에서 개발된 용어 인식 시스템과 통합된 분산・병렬 처리 기반 과학기술 지식 추출 프레임워크의 개발이 필요하다.

참 고 문 헌

[1] F. Smadja, K. R. McKeown, and V. Hatzivassiloglou, "Translating collocations for bilingual lexicons: A statistical approach," Computational Linguistics, Vol.22, No.1, pp.1-38, 1996.

[2] K. Frantzi, S. Ananiadou, and H. Mima, "Automatic recognition of multi-word terms: the C-value/NC-value method," International Journal on Digital Libraries, Vol.3, No.2, pp.115-130, 2000.

[3] S. K. Song, Y. S. Choi, H. W. Chun, C. H. Jeong, S. P. CHoi, and W. K. Sung, "Multi-words Terminology Recognition Using Web Search," Communications in Computer and Information Science, Vol.264, No.1, pp.233-238, 2011.

[4] 최성필, 송사광, 정한민, "기술 문헌 분석 테스트베드 툴킷 개발", 한국콘텐츠학회논문지, 제12권, 제8호, pp.13-19, 2012.

[5] 정창후, 최성필, 윤화묵, 최윤수, "그리드 기반의 고성능 과학기술지식처리 프레임워크 개발", 한국콘텐츠학회논문지, 제9권, 제12호, pp.877-885, 2009.

[6] B. Daille, E. Gaussier, and J. Lange, "Towards Automatic Extraction of Monolingual and Bilingual Terminology," COLING-94, 1994.

[7] J. S. Justeson and S. M. Katz, "Technical terminology : some lingustic propertis and an algorithm for identification in text," Natural Language Engineering, Vol.1, No.1, pp.9-27, 1995.

[8] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," Computational Linguistics, Vol.16, No.1, pp.22-29, 1990.

[9] R. Cilibrasi and P. Vitanyi, "The Google Similarity Distance," IEEE Trans. Knowledge and Data Engineering, Vol.19, No.3, pp.370-383, 2007.

[10] S. Ghemawat, H. Gobioff, and S. Leungm, "The Google File System," In Proc. of ACM Symposium on Operating Systems Principles, pp.20-43, 2003.

[11] W. Tom, and C. Doug, *Hadoop:The Definitive Guide*, O'REILLY, 2009.

저 자 소 개

최 윤 수(Yunsoo Choi) 정회원

- 1993년 : 충남대학교 컴퓨터공학과(학사)
- 1995년 : 충남대학교 컴퓨터공학과(석사)
- 1995년 ~ 현재 : 한국과학기술정보연구원 선임연구원

<관심분야> : 정보검색, 텍스트마이닝, 빅데이터

이 원 구(Wongoo Lee) 정회원

- 2000년 : 한남대학교 대학원 컴퓨터공학과 졸업(공학석사)
- 2005년 : 한남대학교 대학원 컴퓨터공학과 졸업(공학박사)
- 2005년 ~ 현재 : 한국과학기술정보연구원 선임연구원

<관심분야> : 데이터베이스, 지식관리, 과학데이터

이 민 호(Minho Lee) 정회원

- 2000년 : 충남대학교 대학원 컴퓨터공학과 졸업(석사)
- 2006년 : 충남대학교 대학원 컴퓨터공학과(박사수료)
- 2001년 ~ 현재 : 한국과학기술정보연구원 선임연구원

<관심분야> : 시맨틱 웹, 빅데이터, 지식관리

최 동 훈(Dong-Hoon Choi) 정회원

- 1983년 : 한국과학기술원 전산학과 졸업(석사)
- 1989년 : Northwestern University 전산학과 졸업 (박사)
- 1992년 ~ 1999년 : 동덕여자대학교 부교수
- 2005년 ~ 현재 : 한국과학기술정보연구원 책임연구원

<관심분야> : 데이터베이스, 바이오인포매틱스

윤 화 묵(Hwamook Yoon) 정회원

- 1997년 : 공주대학교 대학원 전자계산학과 졸업(석사)
- 2008년 : 배재대학교 컴퓨터공학과 졸업(박사)
- 현재 : 한국과학기술정보연구원 정보기술연구실 책임연구원

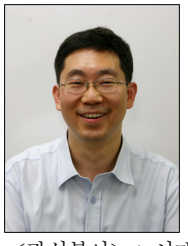
<관심분야> : 데이터베이스, 정보검색, 온톨로지

송 사 광(Sa-Kwang Song) 정회원

- 1999년 : 충남대학교 컴퓨터과학과(석사)
- 2011년 : 한국과학기술원 전산학과(박사)
- 2010년 ~ 현재 : 한국과학기술정보연구원 선임연구원

<관심분야> : 텍스트마이닝, 자연어처리, 정보검색, 시맨틱웹

정 한 민(Hanmin Jung) 정회원



- 2003년 : 포항공과대학교 컴퓨터공학과(박사)
- 2004년 ~ 현재 : 한국과학기술정보연구원 책임연구원
- 2004년 ~ 현재 : 과학기술연합대학원대학교 겸임교수

<관심분야> : 시맨틱웹, 정보검색, 자연어처리, HCI