정보 검색용 다중 스레드 한국어 형태소 해석기

최유경[U] 안동언 정성종
전 북 대 학 교 컴 퓨 터 공 학 과
ykchoi@ielahp.chonbuk.ac.kr, {duan, sjchung}@moak.chonbuk.ac.kr

A Korean Morphlogical Analyzer Supports Multi-Threads

Yoo-Kyung Choi[U] Dong-Un An Sung-Jong Chung
Dept. of Computer Engineering, Chonbuk National University

요 약

본 논문에서는 한국어 형태소 해석기에 다중 스레드 기법을 도입하여 다중 처리가 가능하도록 하였다. 기존의 여러 형태소 해석기들은 언어 분석에만 관심이 있었기 때문에 다량의 문서를 동시에 처리하는 기능을 고려하지 않았다. 그러나 형태소 해석기가 정보 검색 시스템 분야에서 사용되기 시작하면서, 다수의 사용자가 대량의 문서를 처리해야 하는 필요성이 생겼다.

스레드 간에는 메모리 영역과 같은 자원을 공유한다. 이러한 특징 때문에 자칫하면 예상치 못한 결과가 야기될 수 있다. 따라서, 다중 스레드 기법을 사용하기 위해서는 스레드의 특징을 고려한 조치가 필요하다. 기존의 한국어 형태소 해석기의 소스 코드를 분석하여 자주 사용되는 전역 변수는 하나의 구조체로 구성하였다. 그리고 이러한 전역 변수와 크기가 큰 지역 변수를 사용할 때 메모리를 동적으로 할당하였다. 또한, 파일에서 입력값을 읽어오거나 파일에 결과값을 쓰는 등 여러 스레드가 접근할 때 값이 변경될 위험이 있는 부분은 조건 변수를 이용하여 동기화 시켰다.

구현된 시스템의 검증을 위하여, 단일 스레드 방식으로 순차적인 처리를 하는 원래의 형태소 해석기와 비교 실험을 실시하였다. 35Kbyte 문서 30개를 처리하는 경우, 다중 처리가 가능한 형태소 해석기가 단일 스레드 방식의 형태소 해석기보다 처리속도가 약 12% 향상되었다.

1. 서 론

인간의 언어를 컴퓨터에서 처리하고자 하는 자연언어 처리 분야에서 형태소 해석기에 대한 연구는 꾸준히 진행되어 왔다. 형태소 해석은 언어 자료를 의미를 가진 최소의 언어 단위인 형태소(morpheme) 정보로 나열하는 것이다.

초기에는 주로 형태소 정보를 해석하는 방법에 초점을 두었기 때문에 실용성과는 거리가 먼 수준에 그쳤고, 1990년대에 들어서 응용 분야가 확대되면서 실용적인 형태소 해석 시스템을 개발하기 시작하였다.[1][2]

특히 인터넷의 발전으로 정보 검색 시스템이 급속히 발달하였고, 일반인들에게도 널리 보급되었다. 따라서 정보 검색 시스템의 기초 작업이라 할 수 있는 색인기의 중요성도 커지게 되었다. 정보 검색 시스템의 색인기로서 기존의 단순한 명사 추출기를 사용하기보다는 형태소 해석기가 본격적으로 도입되는 추세이며, 그 동안의 지속적인 노력으로 형태소 해석의 성능 면에서는 매우 높

은 정확률을 보이고 있다.[3]

그러나, 규모가 점점 확대되어지고 있는 정보 검색 시스템의 현실이 색인 시스템에는 고려되고 있지 않다. 다량의 문서를 처리해야 하거나 다수의 사용자가 이용하는 정보 검색 시스템에서 효율적으로 색인을 하기 위해서는 다중 처리 기능을 지닌 색인 시스템을 도입하는 것이 필요하다.

본 논문에서는 다중 처리가 가능한 한국어 형태소 해석기 모델을 제안한다. 그리고 구현한 시스템을 실제 색인 시스템으로 사용할 때 어느 정도 효율성을 높일 수 있는지 실험을 통하여 검증해 보도록 한다.

다중 처리가 가능한 한국어 형태소 해석기를 구현하기 위하여 다중 스레드 기법(Multi-threads method)을 도입하였다. 다중 스레드 기법은 병렬 처리나 동시 처리가 요구되는 시스템에서 많이 사용되는 기술로, 다수의 프로세스를 생성하여 처리하는 기법에 비하여 여러 장점을 가지고 있다.

본 논문에서는 다중 스레드 기법의 특징을 고려하여 다중 처리가 가능한 한국어 형태소 해석기를 구현하는 과정을 소개한다. 구현한 다중 처리가 가능한 한국어 형태소 해석기를 검증해 보기 위하여 단일 스레드로 순차 처리를 하는 원래의 형태소 해석기와 비교 실험을 실시한다.

논문의 구성은 다음과 같다. 2장에서 스레드의 특징 및 다중 스레드 기법을 사용하여 시스템을 구현할 때의 고려사항에 대하여 제시한다. 3장에서는 다중 처리가 가능한 형태소 해석기의 설계 과정을 소개하고, 구현된 시스템을 제시한다. 4장에서 시스템의 검증을 위한 실험을 실시하고 결과를 제시한다. 마지막으로 5장에서 결론을 맺는다.

2. 다중 스레드 기법의 도입

2.1 다중 스레드 기법

스레드(thread)란 한 프로그램에서 독립적인 하나의 작업 단위 또는 제어 흐름을 의미한다.[4][5]

전통적인 프로그래밍 방식에서는 하나의 제어 흐름만을 사용한다. 이 제어 흐름은 프로그램이 시작되는 부분에서 시작되어 프로그램이 끝날 때까지 하나의 경로만을 따라서 이동한다. 이러한 환경을 단일 스레딩(single-threading)이라 하며, 이 경우에는 한 번에 하나의 작업만이 실행 가능하다.

다중 스레딩(multi-threading)은 한 프로그램에서 이러한 제어 흐름을 동시에 여러 개 생성하는 것을 말한다. 다중 스레딩은 이처럼 여러 개의 제어 흐름을 생성하여 한 프로그램이 다수의 작업을 동시에 할 수 있도록 하는 기술이다.

다중 스레드 기법의 특징을 좀 더 알아보기 위해서 여러 프로세스를 생성하는 기법과의 차이점을 살펴보기로 한다.[4][6]-[13]

fork 시스템 호출을 통해 생성되는 프로세스는 자체적인 변수 및 프로세스 ID, 파일 디스크립터, 가상 메모리 맵과 같은 구조를 모두 포함하는 원본 프로세스의 사본이 된다. 반면에 새로이 생성되는 스레드는 자체적인 스택 및 변수를 가지기는 하지만, 현재 시스템의 상태나 파일 디스크립터 등과 같은 구조를 자신을 생성한 프로세스와 공유하게 된다. 따라서 다중 스레드 프로그램에서는 프로세스의 구조가 여러 스레드들에 의해 공유가 가능하며, 스레드들이 동일한 데이터 공간 내에서 동작하게 된다.

즉, 프로세스의 생성에 비하여 스레드의 생성은 아주 적은 자원만을 필요로 하며, 여러 작업을 동시에 처리 시 시스템의 자원을 비교적 효율적으로 사용할 수 있다. 반면, 자원 공유의 특성 때문에 프로그램 상에서 정교한 타이밍 실패나 의도하지 않은 변수의 공유가 발생하는 경우 원하는 프로그램을 구현하는 것이 실패하게 되는 위험이 있다.

다중 스레드 기법은 단일 프로세서 환경이나 다중 프로세서 환경 모두에서 잘 수행이 된다. 두 가지 경우 모두 CPU를 효율적으로 사용함으로써 처리량을 증가시킬 수 있으나, 특히 다중 프로세서 환경이라면 보다 많은 성능 향상을 기대할 수 있을 것이다.

단일 스레드를 사용하는 전통적인 프로그램에서는 여러 종류의 작업을 수행하기 위해서 복잡한 코드가 필요하였다. 다수의 스레드를 사용하면 이러한 복잡한 코드가 필요없이 구조적이고 단순한 코드로 원하는 작업을 수행할 수 있다.

이러한 장점들로 인해 논문에서는 다중 스레드 기법을 사용하였다. 다중 스레드 기법은 비교적 쉽게 다중 처리가 가능하도록 지원하지만, 이러한 장점을 가지게 한 자원의 공유와 같은 스레드의 특징으로 인해 프로그램 시 세밀한 주의가 필요하기도 하다. 다중 처리가 가능한 형태소 해석기를 구현하기 위해 고려한 사항들을 살펴보기로 한다.

2.2 고려사항

단일 스레드 방식의 한국어 형태소 해석기에 다중 스레드 기법을 도입하기 위해서는 스레드 간의 동작 특성을 고려한 조치가 필요하다.

앞에서 살펴보았듯이, 다중 스레드 시스템에서는 메모리와 같은 시스템 상의 자원을 서로 다른 여러 스레드 간에 공유하게 된다. 또한 스레드들은 거의 동시에 병렬적으로 각자의 작업을 처리한다. 그러므로, 하나 이상의 스레드가 동시에 같은 메모리 영역에 접근하는 경우에는 자칫하면 프로그램에서 예상치 못한 값이 야기될 수도 있다.

시스템 상의 자원이 한정되어 있다는 점 또한 고려해야 한다. 다수의 스레드가 시스템 상의 한정된 자원을 동시에 사용하면서 병렬적으로 작업을 수행하다 보면, 작업의 부하가 커지거나 생성된 스레드의 개수가 증가함에 따라 자원의 사용도가 높아지게 된다. 프로그램이 실행되는 환경에 따라 차이가 있겠지만, 심각한 경우에는 시스템이 다운되는 것과 같은 치명적인 상황이 발생할 수도 있다.

단일 스레드 방식의 기존의 한국어 형태소 해석기[14]에 다중 스레드 기법을 도입하기 위하여 몇 가지 조치를 취하였다.

① 한국어 형태소 해석기에서 사용하는 전역 변수와 크기가 큰 지역 변수에 대한 조정

② 스레드 간의 동기화

위의 사항은 모두 여러 스레드가 동시에 같은 메모리 영역에 접근하거나, 프로그램의 변수들의 크기가 한 스레드가 가질 수 있는 고유의 스택의 크기를 넘어서는 경우에 발생할 수 있는 예기치 못한 결과를 방지하기 위한 것이다.

①에 관한 내용으로서, 형태소 해석기를 분석하여 변수의 사용 현황을 살펴보고, 시스템에서 많이 사용되는 중요 전역 변수들은 하나의 구조체로 구성하였다.

형태소 해석기의 전역 변수에는,

(가) 현재 처리 중인 어절의 사전 정보를 적재하는 변수들

(나) 형태소 해석 시 빈번히 사용되는 스택 관련 변수들

이 많은 부분을 차지하고 있다. 이러한 변수들은 현재 어절의 분석 동안 계속 참조되어지며, 형태소 해석기의 핵심 부분에 속한다고 할 수 있다.

이러한 변수들을 포함하는 구조체와 크기가 큰 지역 변수들은 사용 시 동적으로 메모리를 할당해 주도록 하였다. [그림 1]은 이렇게 형성된 구조체의 세부 내용이다.

```
struct MorphAnalysisData {

 /* 어절에 대하여 사전 정보를 적재하기 위한 테이블 */
 int  DICT_TABLE[MAX_DICT_REF][MAX_DICT_REF];
 /* 어절에서 어느 글자열 위치부터 사전 검색을 했는지의 여부 */
 int  if_ref_dict[MAX_LINE];

 int  ANALYSIS_SUCCESS;

 /* 문법 검사의 중간 결과를 저장하는 스택의 구조 */
 unsigned int      STACK_CIT_NUM[MAX_STACK_SIZE];
 unsigned int      STACK_POSI_FROM[MAX_STACK_SIZE];
 unsigned int      STACK_POSI_TO[MAX_STACK_SIZE];
 unsigned int      STACK_CIT_LIST[MAX_STACK_SIZE][MAX_LRC_SIZE][TWO_SIDE];
 unsigned int      STACK_CIT_INDEX;

 /* 문법 검사의 중간 결과를 저장하는 스택의 백업용 구조 */
 unsigned int      _STACK_CIT_NUM_[MAX_STACK_SIZE];
 unsigned int      _STACK_POSI_FROM_[MAX_STACK_SIZE];
 unsigned int      _STACK_POSI_TO_[MAX_STACK_SIZE];
 unsigned int      _STACK_CIT_LIST_[MAX_STACK_SIZE][MAX_LRC_SIZE][TWO_SIDE];
 unsigned int      _STACK_CIT_INDEX_;
};
```

[그림 1] 전역변수의 구조체화

스레드의 구조에는 고유의 스택 영역이 존재하기 때문에 크기가 작은 변수들의 경우에는 여러 스레드가 동작하더라도 값이 잘못 변경되어질 위험이 적다. 그러나 크기가 큰 지역 변수의 경우에는 이러한 영역을 초과할 수 있으므로 메모리 동적 할당 방식으로 변경하여 작업 중 값이 변경되지 않도록 하였다.

여러 개의 스레드가 동작하는 환경에서 각 스레드마다 메모리를 동적 할당하여 사용하면 메모리의 사용량이 증가하게 된다. 따라서 최대한 프로그램을 최적화시키는 작업이 필요하다. 원래의 형태소 해석기의 전체적인 분석을 통하여 필요 없는 루틴을 제거하는 등 수정을 가하였다.

②에 대한 처리는 다중 스레드 프로그램에서 필수적이다. 다수의 스레드들이 함께 동작하고 있으므로, 한 스레드가 값을 변경하고 있는 메모리 영역을 다른 스레드가 동시에 접근하였을 경우 문제가 발생할 수 있다.

형태소 해석기의 코드 중, 파일에서 입력값을 읽거나 파일에 결과값을 쓰는 부분처럼 여러 스레드가 접근시 다른 값으로 변경될 위험이 있는 부분을 찾아 조건 변수를 이용하여 동기화 시켰다.

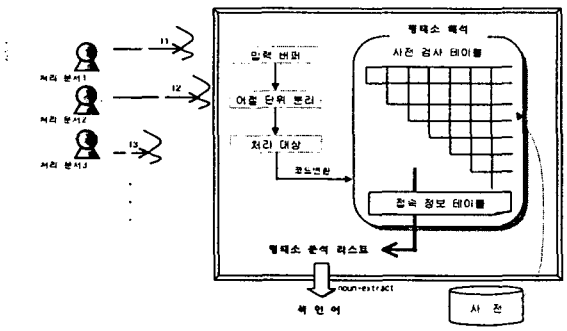
3. 다중 처리가 가능한 한국어 형태소 해석기

다중 처리가 가능한 한국어 형태소 해석기를 구현하기 위하여, 위에서 살펴본 고려사항을 반영하여 기존의 형태소 해석기를 수정하였다.

주요 스레드 라이브러리에는 Win32, OS/2, POSIX의 3가지가 있다.[6] Win32와 OS/2는 각각 NT/Win95와

OS/2 플랫폼에서만 실행되는 제약적인 특성을 띤다. 반면에 POSIX 스레드 라이브러리는 플랫폼의 제약을 거의 받지 않기 때문에 대부분의 UNIX나 Linux 시스템에서 실행이 가능하다. 본 논문에서도 POSIX 스레드 라이브러리를 사용하였다.

다중 처리가 가능한 한국어 형태소 해석기의 설계는 [그림 2]와 같다.



[그림 2] 다중 처리가 가능한 형태소 해석기

[그림 2]에서 T1, T2, T3는 각각 생성된 스레드를 의미한다.

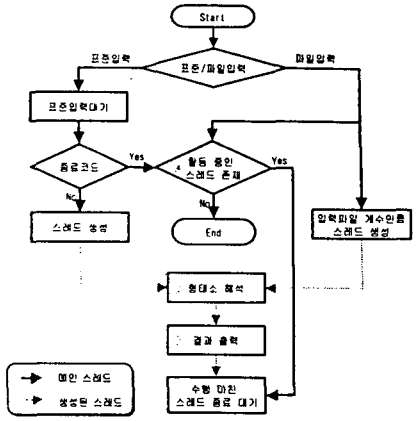
시스템에 입력되는 사용자의 요구나 처리 대상 문서가 존재할 때마다 스레드를 생성한다. 이렇게 생성된 스레드는 각각 자신의 입력에 대하여 형태소 해석을 실시하고, 색인어를 추출한다. 즉, 한꺼번에 처리해야 할 작업이 하나 이상 존재하는 경우에는 처리 작업의 개수만큼 스레드가 생성되며, 스레드들은 동시에 시스템의 형태소 해석부에 접근하여 형태소 해석을 수행하고 결과를 반환한다.

이 때, 색인어 추출에 사용되는 사전들은 메모리에 한 번만 적재되어 여러 스레드에 의해 사용된다. 이것은 스레드 간에 자원을 공유할 수 있는 특성이 있기 때문이며, 이러한 방식으로 메모리 공간과 같은 시스템 자원을 효율적으로 사용할 수 있다.

구현된 형태소 해석기의 입력에 대한 처리 과정을 요약하면 [그림 3]과 같다.

①의 과정에서 새로운 스레드를 생성하는 방법은 POSIX 스레드 라이브러리에서 제공하는 pthread_create 함수를 사용하였다.

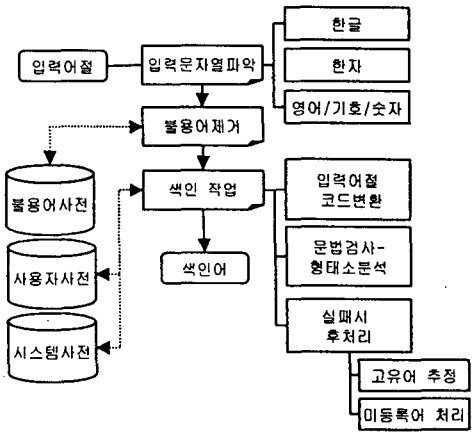
②와 ③의 과정은 생성된 각 스레드는 각자의 입력에 대한 형태소 해석을 독자적으로 수행하며, 작업이 끝나면 그 스레드는 형태소 해석의 결과를 개별적인 파일이나 표준 출력으로 저장한 뒤 종료하는 것이다.



[그림 3] 다중 스레드를 이용한 처리 과정

현재 활동 중인 스레드가 존재하는지의 여부를 검사하는 ④의 과정은 다음과 같은 이유로 필요하다. 스레드들이 여러 개 생성되어 실행되고 있는 다중 스레드 프로그램 환경에서 한 스레드가 작업을 마쳤다고 하여 프로그램 전체가 종료되어서는 안 된다. 즉, 최종적으로 프로그램을 종료할 때에는 실행 중에 있던 모든 스레드들이 작업을 마친 상태여야 한다. 메인 스레드는 이러한 이유로 현재 활동 중인 스레드가 있는지 점검하여 없을 경우에 프로그램을 종료한다. 논문에서는 이러한 역할을 수행하기 위해 POSIX 라이브러리에서 제공하는 join 함수를 사용하였다.

형태소 해석부는 원래의 형태소 해석기의 방식을 따른다. 형태소 해석의 주요 처리 과정은 [그림 4]와 같다.



[그림 4] 형태소 해석 흐름도

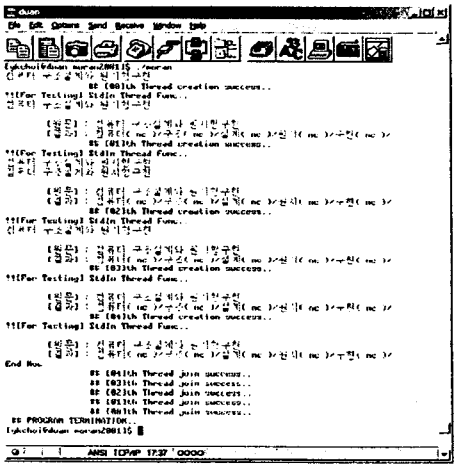
구현된 형태소 해석기의 실행화면은 [그림 5] 및 [그림

6]과 같다. 각각의 그림은 표준 입력에 관한 처리와 파일 입력에 관한 처리 과정 및 그 결과를 보여준다.

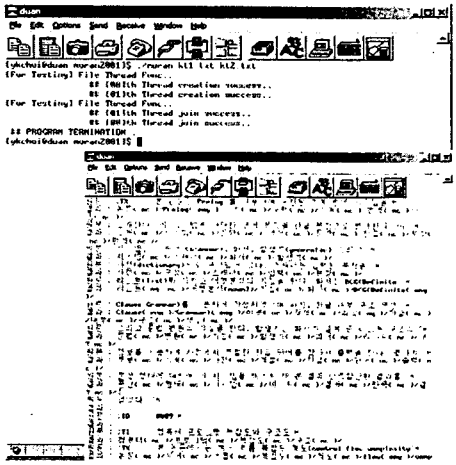
[그림 5]의 경우는 표준 입력으로부터 문장이 입력될 때마다 스레드를 생성하여 형태소 해석을 수행하고 결과를 반환하고 있다.

[그림 6]의 경우 2개의 처리할 파일을 입력받고 있다. 따라서 프로그램 상에서 내부적으로 두 개의 스레드가 생성되어 각각 자신의 입력 파일에 대한 형태소 해석을 실시하고 그 결과를 파일로 저장하게 된다.

두 그림에서 각 입력에 대한 스레드들의 생성 및 처리, 종료 과정을 알아보기 쉽도록 임시로 현재 스레드의 상황을 메시지로 출력하도록 하였다.



[그림 5] 표준 입력에 대한 처리



[그림 6] 파일 입력에 대한 처리

4. 실험 및 평가

구현된 한국어 형태소 해석기가 다중 처리가 가능한지 검증하기 위해서는 다수의 처리 문서가 동시에 시스템에 입력될 수 있는 환경을 만들어야 한다. 이를 위하여 명령어와 함께 처리할 파일을 여러 개 입력받을 수 있도록 하였다. 각 파일은 생성된 스레드 별로 형태소 해석이 되고, 결과 또한 파일로 저장된다.

형태소 해석기가 실행되는 환경은 PentiumⅢ-450MHz 프로세서, 128Mbyte 메인 메모리, Linux RedHat 7.1이 운영체제로 설치된 컴퓨터에서 실행된다.

첫 번째 실험은 생성되는 스레드의 개수는 고정시키고, 파일의 크기를 변화시키면서 실시하였다. 5Kbyte, 10Kbyte, 15Kbyte, 20Kbyte, 25Kbyte, 30Kbyte, 35Kbyte의 문서 각각 5개에 대하여 원래의 시스템과 구현된 시스템의 총 처리 속도를 비교하여 보았다.

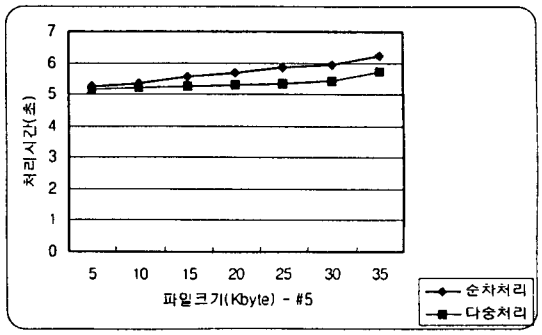
단일 스레드 방식의 순차처리를 하는 원래의 시스템과 구현된 다중 처리가 가능한 시스템에서의 처리 속도는 <표 1>과 같다.

(단위 : 초)

입력 / 시스템	5K 5개	10K 5개	15K 5개	20K 5개	25K 5개	30K 5개	35K 5개
순차처리 (단일스레드)	5.26	5.34	5.54	5.69	5.84	5.94	6.24
다중처리 (다중스레드)	5.16	5.21	5.25	5.30	5.35	5.44	5.71
비고(%)	1.90	2.43	5.23	6.85	8.39	8.42	8.49

<표 1> 실험 결과1

위의 실험 결과를 그래프로 표현한 것이 [그림 7]이다.



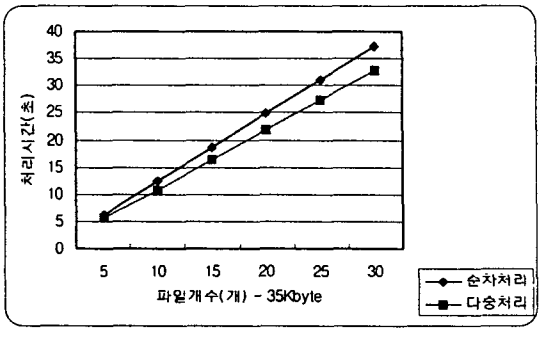
[그림 7] 실험 결과1

두 번째는 일정한 크기의 파일에 대하여 한꺼번에 입력되는 파일의 개수를 증가시키면서 실험하였다. 결과는 아래와 같다.

(단위 : 초)

입력 / 시스템	35K 5개	35K 10개	35K 15개	35K 20개	35K 25개	35K 30개
순차처리 (단일스레드)	6.24	12.44	18.64	24.84	31.04	37.24
다중처리 (다중스레드)	5.71	10.78	16.34	21.79	27.45	32.88
비고(%)	8.49	13.34	12.34	12.28	11.57	11.71

<표 2> 실험 결과2



[그림 8] 실험 결과2

<표 1>과 <표 2>의 비고란은 각각 다중 스레드를 하용하여 다중 처리를 수행하는 형태소 해석기가 단일 스레드를 사용하는 순차처리 방식의 형태소 해석기에 비해 향상된 처리 속도의 정도를 나타낸 값이다.

<표 1>과 <표 2>에서 볼 수 있듯이 두 가지 실험 모두에서 단일 스레드 환경의 시스템보다 다중 스레드를 이용한 다중 처리를 수행하는 형태소 해석 시스템이 총 처리 시간 면에서 이득을 얻고 있다.

또한 [그림 7]과 [그림 8]에서 처리할 작업의 양이 많아지거나 동시에 처리할 작업의 개수가 많아질수록 순차처리 방식의 시스템과 다중 처리를 실시하는 시스템간의 처리 속도의 차이가 전반적으로 커지는 추세임을 알 수 있다.

5. 결론

정보 검색 시스템의 색인 작업은 대량의 문서를 대상으로 하며, 일반적으로 오프 라인 상태에서 주기적으로 이루어진다. 본 논문에서는 정보 검색 시스템의 색인기로 많이 사용되는 형태소 해석기에 다중 처리 기능을 추가하여, 색인 작업의 효율성을 높일 수 있는 가능성을 검증해 보고자 하였다.

실험을 수행한 한 가지 예를 들자면 35Kbyte 문서 30개에 대하여 다중 스레드를 사용하는 형태소 해석기가 단일 스레드 방식에 비하여 약 12% 향상된 속도로 문서를 모두 처리하였다.

본 논문에서 실시한 이러한 실험은 다중 처리가 가능한 형태소 해석기가 기존의 순차적인 방식의 형태소 해석기에 비하여, 많은 수의 문서를 시간과 자원을 절약하며 동시에 색인을 할 수 있음을 보여준다.

제안한 다중 처리가 가능한 형태소 해석기는 정보 검색 시스템의 색인기로서 효율적으로 많은 수의 문서를 색인 한다거나 분산 환경 하에서 한꺼번에 처리 요구가 발생하는 경우 등 활용 분야가 넓다고 생각된다.

그러나 다중 스레드 기법을 이용한 프로그래밍에서는 여러 스레드가 메모리와 같은 시스템의 자원을 공유하면서 동시에 동작하기 때문에, 한 스레드가 작업 중인 메모리 영역을 다른 스레드가 변경시킬 수 있는 문제가 발생할 수 있다. 따라서 이러한 가능성이 있는 부분을 예견하여 프로그램으로서 처리해야 하는 주의가 필요하였다. 또한 한 어절에 대하여 형태소 해석을 하는 데에는 해당 어절에 대한 사전 정보 적재 변수, 스택 오퍼레이션에 사용되는 전역 변수들 등 크기가 큰 메모리 영역이 지속적으로 참조된다. 이러한 변수들이 사용될 때 동적 메모리 할당을 사용했기 때문에, 한꺼번에 여러 스레드가 형태소 해석을 수행하게 되는 경우 동시에 사용되는 메모리 사용률이 높아지는 상황이 발생하게 된다.

실제적인 응용 프로그램으로서 제안하는 형태소 해석기를 사용하기 위해서는 효율적인 메모리 기법이 적용되어야 할 것이며, 더욱 많은 실험을 통하여 문제를 발견하고 대처하는 작업이 필요하다.

6. 참고문헌

[1] 마이크로소프트지 특별기획 "한글과 한국어 정보처리 그 성과와 전망", 1999. 10.

[2] 자연언어처리, 홍릉과학출판사, Makoto Nagao 저, 황도삼 외 공역, 1998.

[3] 최신 정보검색 시스템의 특성 및 동향, KOSTI 2000 워크샵 한글정보검색 학술발표 논문집, pp.3-30, 2000.12

[4] Beginning Linux Programming, Matthew, Neil, 정보문화사, 2000
[5] Java Network Programming, 인포북, 2001, Merlin Hughes, Michael Shoffner, Derek Hamner, Umesh Bellur 저, 고려대학교 Syres편저
[6] Multithreaded Programming with Pthreads, 1998, Bil Lewis, Daniel J.Berg, Sun microsystems
[7] http://it.soongsil.ac.kr/
[8] http://fox2000.com.ne.kr/info/thread.html
[9] http://namhae.duksung.ac.kr/~ucpark/c-program/node29.html
[10] http://namhae.duksung.ac.kr/~ucpark/c-program/node30.html
[11] http://cosmos.soongsil.ac.kr/course/os/os1997-2/thread.html
[12]http://cse.sch.ac.kr/~doh/os/hello_world.html
[13]http://unix.or.kr/oldstudy/network/13-1.HTM
[14] 좌우접속정보를 이용한 명사 추출기, 안동언, 한글 및 한국어 정보처리 학술대회 및 제1회 형태소 분석기 및 품사태거 평가 워크숍 논문집, pp.173-178, 1999.10.