다중 쓰레드 환경에서 웹 크롤러의 성능 분석

박정우*, 김준호*, 이원주**, 전창호*
*한양대학교 컴퓨터공학과,
**인하공업전문대학 컴퓨터정보과
e-mail : wiltesra@nate.com, haventiger@nate.com,
wonjoo2@inhatc.ac.kr, ch5193@hanyang.ac.kr,

Performance Analysis of Web-Crawler in Multi-thread Environment

Jung-Woo Park*, Jun-Ho Kim*, Won-Joo Lee**, Chang-Ho Jeon*
*Dept. of Computer Science & Engineering, Hanyang University,
**Dept. of Computer Science, Inha Technical College

요 약

본 논문에서는 다중 쓰레드 환경에서 동작하는 웹 크롤러를 구현하고 성능을 분석한다. 이 웹 크롤러의 특징은 검색시간을 단축하기 위하여 크롤링, 파싱 및 페이지랭킹, DB 저장 모듈을 서로 독립적으로 다른 작업을 수행하도록 구현한 것이다. 크롤링 모듈은 웹상의 데이터를 수집하는 기능을 제공한다. 그리고 파싱 및 페이지랭크 모듈은 수집한 데이터를 파싱하고, 웹 페이지의 상대적인 중요도를 수치로 계산하여 페이지랭크를 지정한다. DB 연동 모듈은 페이지랭크 모듈에서 구한 페이지랭크를 데이터베이스에 저장한다. 성능평가에서는 다중 쓰레드 환경에서 쓰레드 수와 웹 페이지의 수에 따른 검색시간을 측정하여 그 결과를 비교 평가한다.

키워드 : 웹 크롤러(Web Cralwer), 페이지랭크(Page-Rank), 검색(Searching)

I. 서 론

월드와이드웹(World Wide Web)은 하이퍼텍스트 문서의 거대한 저장 공간이다. 하이퍼텍스트 문서는 텍스트를 포함하며 웹상의 다른 문서와 하이퍼링크로 연결되어 있다. 오늘날 많은 사용자들이 다양한 웹 문서를 제작하고 있기 때문에 웹 트래픽이 빠르게 증가하고 있다.

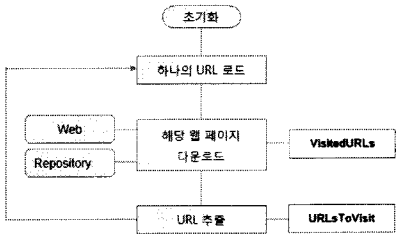
웹 트래픽은 1991-1994년 기간에 1000배가 증가하였고, 웹서버는 1991-1997년 기간에 기하급수적으로 증가하였다. 또한 뉴스 사이트나 포털사이트 상의 많은 웹 문서들이 빈번하게 변경되고 있기 때문에 웹상에서 최신의 정보를 얻기 위해서는 웹 문서들의 변경을 체크하고, 수집하여 분석하는 과정이 필요하다. 이로 인해 최근 기하급수적으로 증가하는 트래픽을 처리하기 위한 비용 또한 급격히 증가하고 있다. 이러한 비용 증가를 줄일 수 있는 하나의 방법은 수많은 웹 페이지를 빠른 시간 내에 분석할 수 있는 도구를 사용하는 것이다. 이러한 도구 중의 하나가 웹 크롤러 이다. 웹상에서 빠른 시간 내에 유익한 정보를 얻기 위해서는 웹 크롤러와 검색엔진의 성능이 중요하다.[1][2]

본 논문에서는 다중 쓰레드 환경에서 동작하는 웹 크롤러의 성능을 분석한다. 웹 크롤러는 크롤링 모듈, 파싱 및 페이지랭크 모듈, DB 연동 모듈로 구현한다. 크롤링 모듈은 웹상의 데이터를 수집하는 기능을 제공한다. 그리고 파싱 및 페이지랭크 모듈은 수집한 데이터를 파싱하고 웹 페이지의 상대적인 중요도를 수치로 계산하여 페이지랭크을 지정한다. DB 연동 모듈은 페이지랭크 모듈에서 구한 페이지랭킹을 데이터베이스에 저장한다.

II. 관련 연구

2.1. 웹 크롤러 구성 및 동작 원리

웹 크롤러는 제한된 자원으로 짧은 시간 내에 웹상에서 최대한 많은 웹 페이지를 자동으로 수집하는 프로그램이다. 웹 크롤러의 기본적인 동작과정은 <그림 1>과 같다.



〈그림 1〉 웹 크롤러의 동작 과정

〈그림 1〉에서 웹 크롤러는 웹 페이지의 URL관리를 위해 2개의 큐인 URLsToVisit와 VisitedURLs을 사용한다. URLsToVisit 큐는 접속할 URL을 저장하고, VisitedURLs 큐는 접속한 URL을 저장한다. URLsToVisit 큐의 초기 데이터는 seedURL이다. seedURL은 사용자 요청에 의해 최초로 수집할 URL이다. seedURL은 사용자가 직접 특정 자료로부터 구하거나, 혹은 수작업을 통해 구한다. 초기 데이터에는 다양한 종류의 URL이 포함될 수 있고, 특정 주제와 관련된 URL이 포함될 수 있다. 대부분의 검색엔진은 잘 알려진 웹 사이트의 URL을 초기 데이터로 사용한다.

웹 크롤러는 다음과 같은 절차를 통해 웹 페이지를 수집한다.

① seedURL을 URLsToVisit에 추가한다.
② URLsToVisit에서 하나의 URL을 로드한다.
③ 해당 IP의 웹 페이지를 다운로드한다.
④ 수집한 웹 페이지를 디스크에 저장하고, VisitedURLs에 URL을 추가한다.
⑤ 수집한 웹 페이지로부터 URL들을 추출하고 VisitedURLs에 포함되지 않은 URL들만 URLsToVisit에 추가한다.
⑥ 마지막으로 URLsToVisit에 URL이 존재하는 동안 ②~⑥ 과정을 반복한다.

웹 크롤러는 다운로드 비율을 최대화하기 위해 다중 프로세스와 쓰레드를 이용하여 웹페이지를 수집한다. 이때 다운로드 비율은 시간 당 수집된 웹 페이지의 비율을 의미한다. 웹 크롤러는 대량의 웹 페이지를 빠르게 수집할 수 있고, seedURL로 주어진 웹 페이지뿐만 아니라 이들과 링크된 웹 페이지들도 수집할 수 있다.[3]

페이지랭크는 웹의 하이퍼링크 구조를 기반으로 웹 페이지의 상대적인 중요도를 측정하여 수치로 나타낸 것이다. Kamvaretal은 페이지랭크를 로컬과 블록으로 나누어 계산하는 페이지랭크 알고리즘을 제안하였다.[4]

로컬 페이지랭크는 블록내의 사용자가 블록B 내에 있을 때 웹 페이지 P에 있을 확률을 나타낸다. 계산은 블록내의 웹 페이지를 대상으로 블록내의 웹페이지들의 관계를 나타내는 그래프를 구하여 기존의 페이지랭크 알고리즘에 대입한다. 블록 페이지랭크는 사용자가 블록 B내에 있을 확률을 나타낸다. 블록 B의 페이지랭크는 B_n/N으로 계산한다. 여기서 B_n은 블록 B에 속하는 웹 페이지의 수이고, N은 전체 웹 페이지 수를 의미한다.

III. 웹 크롤러 및 검색엔진 구현

3.1 웹 크롤러 구현

웹 크롤러는 웹 크롤링, 파싱 및 페이지랭크, DB 저장 모듈로 구성한다. 웹 크롤링은 웹상에서 데이터를 수집하는 기능을 제공한다. 웹 크롤링 모듈은 〈그림 2〉와 같다.

```
// 방문할 urlqueue에 url이 있을 경우 1개의 url 수신
// 만약 없을 경우 더 이상 방문할 url이 없다는 신호를
// MainThread에게 전송

if(URLsToVisit.empty())
  nowPage = URLsToVisit.pop();
else
  send "no job" signal;

소켓 생성;
http 프로토콜 쿼리 생성 및 전송;
http 헤더 분석;

if(질의 성공)
  Html문서 수신;
else if(재전송 헤더) {
  재전송 헤더분석;
  url 중복 체크후 urlqueue에 입력;
}else
  서버 접속 불가;
```

〈그림 2〉 웹 크롤링 모듈

파싱 기능은 수집한 데이터를 http 헤더 정보와 하이퍼링크된 url의 실제 내용으로 분류한다.[5] http 헤더에서 http 쿼리의 질의 성공 여부가 반환되면 파싱하여 원하는 정보를 얻는다. 헤더의 쿼리 질의가 성공하면 데이터를 수집하고, 리다이렉트 데이터는 헤더의 location 단의 url을 파싱하여, 재질의가 가능하도록 한다. 이때 http 프로토콜로 전송된 데이터의 charset에 따라 데이터를 수집한다.

본 논문에서는 메타 태그를 분석하여 얻은 charset을 Mysql의 기본 charset인 EUC-KR로 변환하여 〈그림 3〉 파싱-페이지랭크 모듈의 입력으로 지정한다.

```
//파싱-페이지랭킹
void PasingHTML(char *buf, retpasing *ret)
{
  while(파싱할 문서){
    문자중 new line or tab 은 blank로 전환;
    if('<') {                          // 태그 파싱 시작
      if('--') 주석 체크;
    }
    if(콘텐츠) { 리턴할 ret에 복사; }
    if(태그)   { 태그 버퍼에 복사; }

    if('>') {                          //태그 파싱 종료
      스크립트 형태 예외처리;
      스타일 태그 형태 예외처리;
      url 분석;
      if('href')                       //href라면 url
      {
        if(절대주소) url 저장;
        else {                         //상대주소
          절대주소로 변경;
          url 저장;
        }
      } //if('href')
    } //if('>')
  }//while

  //페이지 내용을 Token화하여 분류
  Tokenizing(Token, Token 수) list;
  return;
}
```

〈그림 3〉 파싱-페이지랭크 모듈

〈그림 3〉 파싱 모듈은 데이터를 입력 받아 태그를 파싱하고, 태그의 content 부분과 하이퍼링크 태그의 url을 추출하는 단계로 구성되어 있다. 메모리 관리를 위해 동일한 메모리 공간을 공유하고, 처음부터 메모리 공간을 반납하고 재할당 받는 형식으로 구현한다.

파싱 결과의 content는 페이지랭크 과정을 통하여 랭크 값을 보정한다. 페이지랭크 과정은 링크 수 보정, 키워드 값 보정, 타입 보정으로 구성된다. 링크 수 보정은 파싱한 데이터의 하이퍼링크 URL의 유일성 체크를 하는 동시에 호출된 링크의 빈도수를 구한다. 해당 페이지가 링크된 수는 해당 페이지의 중요도를 측정하는 요소 중에 하나가 된다. 키워드 값은 미리 입력된 키워드 풀에 존재하는 키워드와 대조하여 매칭된 키워드의 랭크 보정치와 키워드 빈도수를 고려하여 보정한다. 키워드 풀은 내용을 분류하는 기준이 된다.

타입 보정은 각 사이트의 메인페이지인 주소 정보가 누락된 url과 각 민관기관의 홈페이지 등에 차등 보정치를 준다. 링크 수 보정, 키워드 값 보정, 타입 보정에서 산출된 랭크 보정치에 개별 보정치를 합산하여 크롤러에 반환한다.

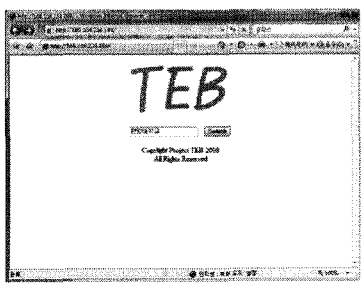
〈그림 4〉의 DB 연동 모듈은 페이지랭킹 모듈에서 구한 랭크 수치를 데이터베이스에 저장한다.[6][7][8]

```
// t1 = url, t2 = address, t3 = category, t4 = content
// rank = inrank값

int SQL_Input(char *t1, char *t2, char *t3, char
                *t4, int rank)
{
  동기화를 위한 critical section 시작;
  Mysql 초기화;
  Mysql DB 연동;

  중복성 체크 쿼리 작성;
  쿼리 실행;
  if(중복성 체크) 중복시 함수 종료;

  DB 저장용 쿼리 작성;
  쿼리 실행;
  Mysql DB 해제;
  return 0;
}
```

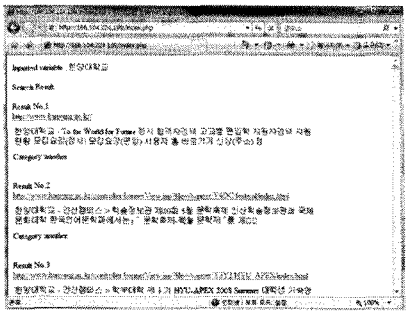
〈그림 4〉 DB 연동 모듈

3.2 검색엔진 구현

본 논문에서는 웹 크롤러의 성능 평가를 위한 검색 엔진을 구현한다. 이 검색 엔진은 PHP 언어를 사용하여 〈그림 5〉와 같이 구현한 웹 응용프로그램이다.



〈그림 5〉 검색 엔진 화면



〈그림 6〉 검색 결과

〈그림 5〉에 검색어를 입력하면 그 검색어를 검색 엔진에 질의하여 〈그림 6〉과 같은 검색 결과를 얻는다.[9][10] 이때 검색엔진은 url, content에 대한 SQL-LIKE 질의로 pagerank 수치가 높은 순서로 검색한다.

Ⅳ. 성능 평가

4.1 실험 환경

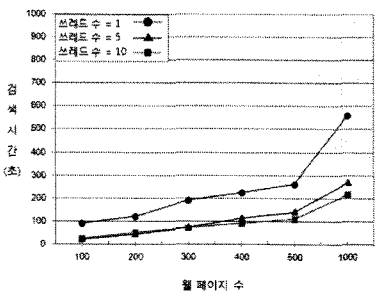
본 논문에서 구현한 웹 크롤러의 성능 평가를 위한 실험 환경은 〈표 1〉과 같다.

〈표 1〉 실험 환경

H/W	CPU	팬티엄 IV 3.0Ghz
	RAM	DDR2 512M
	HDD	250G
S/W	OS	Linux
	Script 언어	PHP 5
	DBMS	MySQL 5
	구현 언어	Visual C++
네트워크	Bandwidth	100M
쓰레드 수		1 ~ 80
웹 페이지 수		100 ~ 100000

4.2 실험 결과 분석

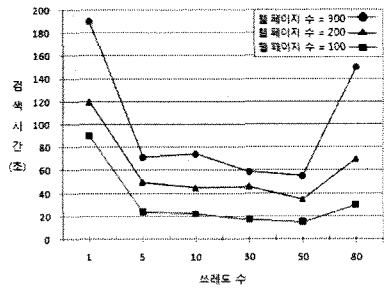
웹 크롤러의 성능 평가 척도는 검색시간이다. 웹 페이지 수의 증가에 따른 검색시간을 측정한 결과는 〈그림 7〉와 같다.



〈그림 7〉 웹 페이지 수에 따른 검색시간

〈그림 7〉를 살펴보면 웹 페이지 수 증가에 비례하여 검색시간도 증가함을 볼 수 있다. 그리고 웹 페이지 수가 500이상 일 때 검색시간은 더욱 급격히 증가함을 알 수 있다. 또한, 쓰레드 수가 10 일 때 페이지 수 증가에 따른 검색시간이 최소임을 볼 수 있다. 따라서 쓰레드 수를 증가시키면 검색시간이 짧아진다는 것을 알 수 있다. 그러므로 다중 쓰레드 환경은 네트워크 지연에 따른 검색시간 증가를 막을 수 있는 하나의 방법이라 할 수 있다.

쓰레드 수를 무한으로 계속 증가시킨다면 그에 따른 검색시간도 계속 감소하는지를 검증해 본다. 따라서 쓰레드 수의 증가에 따른 검색시간을 측정한 결과는 〈그림 8〉과 같다.



〈그림 8〉 쓰레드 수에 따른 검색시간

〈그림 8〉을 살펴보면 쓰레드 수가 1일 때 검색시간이 최대임을 볼 수 있다. 하지만 쓰레드 수가 5일 때는 단일 쓰레드 환경에 비해 2배의 성능 향상을 보이고 있다. 그리고 쓰레드 수 5~50 사이에서는 쓰레드 수의 증가에 따라 검색시간이 감소함을 알 수 있다. 하지만 쓰레드 수 50이상에서는 쓰레드 수 증가에 따라 검색시간도 비례하여 증가함을 볼 수 있다. 따라서 쓰레드 수를 무한으로 계속 증가시킨다 해도 그에 따라 검색시간이 계속 감소하지는 않는다는 것을 알 수 있다. 이것은 메모리 용량이 512MB인 시스템의 한계 때문이다. 즉, 각 쓰레드가 서로 다른 저장 공간을 사용하기 때문에 메모리가 부족하면 가상메모리를 사용한다. 이때 하드디스크 스왑 현상이 빈번히 발생함에 따라 검색시간이 급격히 증가하기 때문이다.

V. 결론

본 논문에서 구현한 웹 크롤러는 크롤링, 파싱 및 페이지랭킹, DB 저장 모듈로 구성된다. 이 웹 크롤러의 성능을 평가하기 위해 PHP 언어를 사용하여 웹 응용프로그램 기반의 검색엔진을 구현한다. 검색 엔진 화면에서 검색어를 입력하면 웹 크롤러는 검색어와 관련된 데이터를 웹상에서 수집한다.

성능평가에서는 쓰레드 수와 웹 페이지 수에 따른 검색시간을 측정하고 그 결과를 비교 분석한다. 먼저 쓰레드 수에 따른 검색시간 측정 결과를 분석해보면 쓰레드 수를 증가시키면 검색시간이 감소한다는 것을 알 수 있다. 즉, 다중 쓰레드 환경이 네트워크 지연에 따른 검색시간 증가를 최소화 할 수 있다는 것을 알 수 있다. 하지만 쓰레드 수 50이상에서는 쓰레드 수 증가에 따라 검색시간이 증가함을 볼 수 있다. 따라서 쓰레드 수를 무한으로 계속 증가시킨다 해도 그에 따른 검색시간의 감소 효과를 얻지 못한다는 것을 알 수 있다. 이것은 각 쓰레드가 서로 다른 저장 공간을 사용하기 때문에 메모리가 부족하면 가상메모리를 사용한다. 이때 하드디스크 스왑 현상이 빈번하게 발생하면 이로 인해 검색시간이 급격히 증가하기 때문이다.

참고문헌

[1] 김희철, 채수환, "고성능 웹크롤러의 설계 및 구현(Design and Implementation of a High Performance Web Crawler)," 디지털콘텐츠학회논문지, 제4권, 제2호, 127-137쪽, 2003년.

[2] 김계정, 김민수, 김이른, 황규영, "커뮤니티 제한 검색을 위한 웹크롤링 및 PageRank 계산," 한국정보과학회 2005 한국컴퓨터종합학술대회논문집(B), Vol. 32, No. 1, 1-3쪽, 2005년, 7월.

[3] 신은정, "대용량 검색 엔진을 위한 병렬 웹 크롤러의 설계 및 구현," KAIST 석사학위논문, 2007년.

[4] Soumen Chakrabarti, "Mining the Web: Discovering Knowledge from Hypertext Data," Morgan Kaufmann, 2003.

[5] Bing Liu, "Web Data Mining : Exploring Hyperlinks, Contents, and Usage Data," Springer, 2007.

[6] 구글 페이지랭크(PageRank) 알고리즘, http://www.emh.co.kr/xhtml/google_pagerank_citation_ranking.html.

[7] "The anatomy of large scale search engine," http://www.emh.co.kr/html/google_search_engine.html.

[8] "Authoritative sources in a hyper-linked environment," http://www.emh.co.kr/xhtml/hubs and authorities.html.

[9] 이병준, 우남윤, "Advanced UNIX programm ing," 정보문화사, 2005년.

[10] 하은용, "(꼭 알아야 할) PHP + MYSQL 프로그래밍," 그린, 2008년.