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

본 논문에서는 다중 쓰래드 환경에서 동작하는 웹 크롤러를 구현하고 성능을 분석한다. 이 웹 크롤러의 특징은 검색시간을 단축하기 위하여 크롤링, 파싱 및 페이지랭크, DB 저장 모듈을 서로 독립적으로 다른 작업을 수행하도록 구현한 것이다. 크롤링 모듈은 웹상의 데이터를 수집하는 기능을 제공한다. 그리고 파싱 및 페이지랭크 모듈은 수집한 데이터를 파생하고 웹 페이지의 상대적인 중요도를 수치로 계산하여 페이지랭크를 제공한다. DB 인도 모듈은 페이지랭크 모듈에서 구한 페이지랭크를 데이터베이스에 저장한다. 성능평가에서는 다중 쓰래드 환경에서 쓰래드 수와 웹 페이지의 수에 따른 검색 시간을 측정하여 그 결과를 비교 평가한다.
키워드: 웹 크롤러(Web Crawler), 페이지랭크(Page-Rank), 검색(Searching)

I. 서론

월드와이드웹(World Wide Web)은 하이퍼텍스트 문서의 거대한 저장 공간이다. 하이퍼텍스트 문서는 텍스트를 포함하여 웹상의 다른 문서와 하이퍼링크로 연결되어 있다. 오늘날 많은 사용자가 다양한 텍스트 문서를 꾸밀고 있기 때문에 웹 트래픽이 빠르게 증가하고 있다.
웹 트래픽은 1991년과 1994년 기간에 1000배가 증가하였고, 웹서버는 1991~1997년 기간에 기하급수적으로 증가하였다. 또한 뉴스 사이트나 포털사이트 상의 많은 웹 문서들이 반복적으로 변경되기 때문에 웹상에서 최신의 정보를 얻기 위해서는 웹 문서들의 변경을 파악하고, 수집하여 분석하는 과정을 필요하다. 이로 인해 최근 기하급수적으로 증가하는 트래픽을 처리하기 위한 비용 또한 급격히 증가하고 있다. 이와 같은 비용 증가를 줄일 수 있는 하나의 방법은 수많은 웹 페이지의 빠른 시간 내에 분석할 수 있는 도구를 사용하는 것이다. 이러한 도구 중의 하나가 웹 크롤러다. 웹상에서 빠른 시간 내에 유익한 정보를 얻기 위해서는 웹 크롤러와 검색 엔진의 성능이 중요하다.[1][2]

본 논문에서는 다중 쓰래드 환경에서 동작하는 웹 크롤러의 성능을 분석한다. 웹 크롤러는 크롤링 모듈, 파싱 및 페이지랭크 모듈, DB 인도 모듈을 구현한다. 크롤링 모듈은 웹상의 데이터를 수집하는 기능을 제공한다. 그리고 파싱 및 페이지랭크 모듈은 수집한 데이터를 파생하고 웹 페이지의 상대적인 중요도를 수치로 계산하여 페이지랭크를 제공한다. DB 인도 모듈은 페이지랭크 모듈에서 구한 페이지랭크를 데이터베이스에 저장한다.

II. 관련 연구

2.1. 웹 크롤러 구성 및 동작 원리
웹 크롤러는 제한된 자원으로 제한된 시간 내에 웹상에서 최대한 많은 웹 페이지를 자동으로 수집하는 프로그램이다. 웹 크롤러의 기본적인 동작과정은 그림 1과 같다.
III. 웹 크롤러 및 검색엔진 구현

3.1 웹 크롤러 구현

웹 크롤러는 웹 크롤링, 파싱 및 페이지랭크, DB 저장 모듈로 구성된다. 웹 크롤링은 웹상에서 데이터를 수집하는 기능을 제공한다. 웹 크롤링 모듈은 (그림 2)와 같다.

(그림 1) 웹 크롤러의 동작 과정

(그림 1)에서 힐 크롤러는 웹 페이지의 URL관리를 위해 2개의 루트 URL을 저장하고, VisitedURLs를 생성한다. URLVisToVisit 큐는 접속할 URL을 저장하고, VisitedURLs 큐는 접속한 URL을 저장한다. URLToVisit 규의 초기 테이터는 seedURL이다.

seedURL은 사용자가 세운 초기 목록에서 수집한 URL이나, seedURL은 사용자가 직접 특정 주소로부터 구하거나 혹은 수의 암호를 통해 구한다. 초기 데이터는 다양한 종류의 URL이 포함될 수 있고, 특정 주소에 관련된 URL이 포함될 수 있다. 대부분의 검색엔진은 잘 알려진 웹 사이트의 URL을 초기 데이터에 사용한다.

웹 크롤러는 다음과 같은 방법으로 데이터를 수집한다.

1) seedURL은 URLToVisit에 추가한다.
2) URLToVisit에서 하나의 URL을 로드한다.
3) 해당 URL의 웹 페이지를 다운로드한다.
4) 수집한 웹 페이지를 디스크에 저장하고, VisitedURLs에 URL을 추가한다.
5) 수집한 웹 페이지로부터 URL들을 추출하고, VisitedURLs에 포함되지 않은 URL들을 URLToVisit에 추가한다.
6) 마지막으로 URLToVisit에 URL이 존재하는 동안 2)~6) 과정을 반복한다.

웹 크롤러는 다중 빌드 비율을 최대화하기 위해 다중 프로세스와 쓰레드를 이용하여 웹 페이지를 수집한다. 이때 다중 빌드 비율은 시간 내에 수집된 웹 페이지의 비율을 의미한다. 웹 크롤러는 대량의 웹 페이지를 퍼져서 수집할 수 있고, seedURL로 주어진 웹 페이지뿐 아니라 이들과 링크된 웹 페이지들도 수집할 수 있다.3

페이지랭크는 웹의 하위페이지 구조를 기반으로 웹 페이지의 상대적인 중요도를 측정하여 순위로 나누어 계산하는 페이지랭크 알고리즘을 계산한다. 4

Kamvareta는 페이지랭크를 로그와 볼록으로 나누어 계산하는 페이지랭크 알고리즘을 계산한다.5

로컬 페이지랭크는 블록내의 사용자가 블록B 내에 있을 때 웹 페이지 P에 있음을 확률을 나타낸다. 계산은 블록내의 웹 페이지를 대상으로 블록내의 수를 확률을 나타내는 그래프를 구하여 기존의 페이지랭크 알고리즘에 대입한다. 블록 페이지 랜징크는 사용자가 블록 B에 있을 확률을 나타낸다. 블록 B의 페이지랭크는 B의/가에 계산한다. 여기서 B는 블록 B에 속하는 웹 페이지의 수이고, N은 전체 웹 페이지 수를 의미한다.

사용할 uint8*의 내terminated 를 검색할 경우 1개의 url 수신 // 만약 앎을 체크하면 이상 발생할 내terminated 이라는 신호를 // MainThread에서 검출
if(URLToVisit.empty())
    nowPage = URLToVisit.pop();
else
    send "no job" signal;

(그림 2) 웹 크롤링 모듈

파싱은 수집한 데이터를 http 헤더 정보와 하위웹크로 url의 실제 내용으로 분리한다.5 http 헤더에서 http 헤더의 정보, 성공 여부가 반환하면 파싱하여 원하는 정보를 얻는다. 헤더의 위치 정보가 성공하면 데이터를 수집하고, 리턴리스트 데이터는 헤더의 location 단의 url을 파싱하여, 재귀가 가능하도록 한다. 이때 http 프로토콜로 진단된 헤더의 charset에 따라 데이터를 수집한다.

본 논문에서는 메타 태그를 분석하여 얻은 charset을 Mysql의 기본 charset인 EUC-KR로 변환하여 (그림 3) 파싱-페이지랭크 모듈의 입력으로 저장한다.

(그림 3) 파싱- 페이지랭크 모듈
(그림 3) 파생 모듈은 데이터를 입력 받아 테그를 파악하고, 테그의 content 부분과 하이퍼링크 테그의 url을 추출하는 단계로 구성되어 있다. 메모리 관리를 위해 동일한 메모리 공간을 공유하고, 처음부터 메모리 공간을 반납하고 재활당 받는 형식으로 구현한다.

파생 결과의 content는 페이지 링크 파일을 통하여 링크 값을 보정한다. 페이지 링크 파일은 링크 수 보정, 키워드 값 보정, 타입 보정으로 구성된다. 링크 수 보정은 파생한 데이터의 하이퍼링크 URL의 유입성 체크를 하는 동시에 호출된 링크의 빈도수를 구한다. 해당 페이지가 링크된 수는 해당 페이지의 중요도를 측정하는 요소 중 하나다. 키워드 값은 이미 입력된 키워드 콜에 존재하는 키워드와 대조하여 매칭된 키워드의 같은 보정치와 키워드 빈도수를 고려하여 보정한다. 키워드 콜은 내용을 분류하는 기준이 된다. 키워드 콜은 내용을 분류하는 기준이 된다. 타입 보정은 각 사이트의 메인페이지의 주요 정보를 누락 된 url과 각 인터넷의 홈페이지 등에 좌표 보정치를 준다. 링크 수 보정, 키워드 값 보정, 타입 보정에서 산출된 링크 보정치에 개별 보정치를 합산하여 크롤러에 반영한다.

(그림 4)의 DB 연동 모듈은 페이지 링크 파일에서 구한 링크 수를 데이터베이스에 저장한다.[6][7][8]

```c
int SQL_Input(char *url, char *address, char *category, char *content, int rank)
{
    // 동기화를 위한 critical section 시작;
    Mysql 초기화;
    Mysql DB 연동;
    취득성 체크 무리 작성;
    무리 실행;
    취득성 체크 (취득성 체크) 수록 해 변수 종료;
    DB 저장용 무리 작성;
    무리 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 입력하고 그 검색어를 검색 엔진에 검색하여 (그림 6)과 같은 검색 결과를 얻는다. 이 검색 엔진은 url, content에 해당 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

<table>
<thead>
<tr>
<th>HW</th>
<th>CPU</th>
<th>RAM</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>멕티임 IV 3.0Ghz</td>
<td>DDR2 512M</td>
<td>250G</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SW</th>
<th>OS</th>
<th>Script 언어</th>
<th>PHP</th>
<th>DBMS</th>
<th>MySQL 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linux</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

네트워크 | Bandwidth | 100M |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>쓰레드 수</td>
<td>1 ~ 80</td>
<td></td>
</tr>
<tr>
<td>웹 페이지 수</td>
<td>100 ~ 100000</td>
<td></td>
</tr>
</tbody>
</table>

(그림 7) 실험 결과 분석

웹 크롤러의 성능 평가 적도는 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 검색 시간도 증가함을 볼 수 있다. 그리고 웹 페이지 수가 500이상일 때 검색시간이 더욱 급격히 증가함을 알 수 있다. 또한, 썸네일 수가 10일때 페이지 수 증가에 따른 검색시간이 적소감을 볼 수 있다. 따라서 썸네일 수 증가시켜면 검색시간이 줄어들지 않는다는 것을 알 수 있다. 그러므로 동일 썸네일 환경은 네트워크 정체에 따른 검색시간 증가를 막을 수 있는 하나의 방법일 수 있다고 할 수 있다.

썸네일 수를 무한으로 계속 증가시키면 그에 따른 검색시간도 계속 감소하는지를 검증해 본다. 따라서 썸네일 수 증가에 따른 검색시간을 추정하고 결과는 (그림 8)와 같다.

![그림 8: 썸네일 수에 따른 검색시간]

(그림 8)를 살펴보면 썸네일 수가 1일 때 검색시간이 최대임을 볼 수 있다. 하지만 썸네일 수가 5일때는 단일 썸네일 환경에 따른 5배의 성능 향상을 보이고 있다. 그리고 썸네일 수 5~50 사이에서는 썸네일 수의 증가에 따라 검색시간이 감소함을 알 수 있다. 하지만 썸네일 수 50이상에서는 썸네일 수 증가에 따라 검색시간도 비례하여 증가함을 볼 수 있다. 따라서 썸네일 수를 무한으로 계속 증가시키면 해도 그에 따라 검색시간이 계속 감소하지는 않는다는 것을 알 수 있다. 이것은 메모리 용량이 512MB인 시스템의 한계 때문이다. 즉, 각 썸네일이 서로 다른 저장 공간을 사용하기 때문에 메모리가 부족하면 가상메모리로 사용한다. 이제 하드디스크 스토리지 비율이 빠르게 발생함에 따라 검색시간이 급격히 증가하기 때문이다.

V. 결론

본 논문에서 구현한 웹 크롤링은 크롤링, 파싱 및 페이지형장, DB 저장 모듈로 구성된다. 이 웹 크롤링의 성능을 평가하기 위해 PHP 이용하여 웹 응용프로그램 기반의 검색 엔진을 구현한다. 검색 엔진 화면에서 검색어를 입력하면 웹 크롤링의 검색어와 관련된 데이터를 화면에서 수집한다.

성능평가에서는 페이지수와 웹 페이지 수에 따른 검색시간을 측정하고 그 결과를 비교 분석한다. 먼저 썸네일 수에 따른 검색시간 측정 결과를 분석해보면 썸네일 수 증가시 키면 검색시간이 감소한다는 것을 알 수 있다. 즉, 다양한 크롤링 환경이 네트워크 정체에 따른 검색시간 증가를 최소화할 수 있다는 것을 알 수 있다. 하지만 썸네일 수 50이상에서는 썸네일 수 증가에 따라 검색시간이 증가함을 볼 수 있다. 따라서 썸네일 수를 무한으로 계속 증가시키면 해도 그에 따른 검색시간의 감소 효과를 얻지 못한다는 것을 알 수 있다. 이것은 각 썸네일이 서로 다른 저장 공간을 사용하기 때문에 메모리가 부족하면 가상메모리를 사용한다. 이때 하드디스크 스토리지 비율이 빠르게 발생하면 이로 인해 검색시간이 급격히 증가하기 때문이다.

참고문헌

(3) 신순영, “대용량 검색 엔진을 위한 별도 웹 크롤링의 설계 및 구현,” KAIST 석사학위논문, 2007년.
(8) "Authoritative sources in a hyper-linked environment,” http://www.enmh.co.kr/xhtml/bubs and authorities.html.
(9) 이병준, 우남숙, “Advanced UNIX programm ing,” 정보문화사, 2005년.
(10) 하승용, "(목 앞앞, 앞 앞) PHP + MYSQL 프로그래밍,” 그린, 2008년.