제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

왕복시간(Round Trip Time)을 고려한 고성능 웹크롤러 설계

정춘호*, 조현태**, 백윤주**
*부산대학교 전자계산학과
**부산대학교 컴퓨터공학과
e-mail : pass@juno.cs.pusan.ac.kr

Design of a High-Performance Web Crawler Considering the Round Trip Time

Chun-Ho Jeong*, Hyun-Tae Cho**, Yun-Ju Beak**
*Dept. of Computer Science, Pusan National University
**Dept. of Computer Engineering, Pusan National University

요 약

인터넷과 정보기술의 급속적인 발전으로 수천 또는 수억에 달하는 방대하고 다양한 정보들이 웹 상에서 존재하게 되었다. 이러한 다양한 정보들 중에서 이용자가 원하는 정보를 제공하기 위해 다양한 검색시스템들이 개발되었다. 웹 크롤러는 검색시스템의 중요한 부분 중의 하나로, 웹 서버를 순회하며 각 페이지에 있는 수많은 정보를 수집하는 프로그램이다. 본 논문에서는 웹 크롤러의 기본요건 중의 하나인 빠른 수집속도를 보장하기 위해 RTT(Round Trip Time) Scheduling 기법을 적용한 URL 스케줄러를 제시하고, 이러한 RTT Scheduling 기법을 이용한 웹 크롤러를 설계한다.

1. 서론

인터넷과 정보기술의 급속한 발전으로 현재 웹 상에 존재하는 정보들은 매우 빠르게 늘어나고 있다. 인터넷 이용자들은 이러한 다양한 정보들 중에 자신들이 원하는 정보를 찾기 위해 검색 시스템에 질의한다. 웹 검색 시스템은 자신이 보유한 다양한 정보들 중에 이용자의 질의에 적합한 문서들을 보여준다. 웹 검색 시스템이 정보를 보유하기 위해서 웹 크롤러를 사용한다. 웹 크롤러는 자동적으로 웹의 하이퍼텍스트 구조를 따라다니며 문서를 수집하고, 재귀적으로 그 문서에 참조되는 다른 문서를 수집하는 식으로 동작되는 프로그램이다[1]. 웹 크롤러는 정보검색 이외에도 전 세계의 웹 서버가 몇 개인지 등을 조사하기 위한 통계 분석 목적, 웹 서버 운영시 데드링크 확인, 수정 등을 통한 유지 보수 목적, 어떤 홈페이지내의 문서를 전부 가져오는 미러링 목적이나, 그 외 여러 가지 복합적인 목적으로도 사용이 가능하다.

이러한 웹 크롤러의 동작을 간단하게 살펴보면, 먼저 초기에 방문하고자 하는 seed URL 을 만들어 주면 크롤러는 해당 URL 의 문서를 받아오고, 그 문서를 분석해서 하이퍼링크를 뽑아내어서 이를 queue 에 저장한다. 그리고 queue 에서 URL 을 하나씩 빼내어 앞서의 작업을 반복하게 된다. 이러한 웹 크롤러의 동작 알고리즘은 단순하지만 방대한 웹의 규모를 고려해 볼 때, 빠른 수집 속도는 필수적이다.

본 논문에서는 웹 크롤러의 문서 수집 속도를 향상시키기 위해 크롤러가 웹 서버에 접근할 때의 timestamp 와 웹 서버로부터의 응답이 도착할 때의 timestamp 를 이용하여 각 웹 서버의 round trip time 를 고려한 URL Scheduling 기법을 제시하고, 이 기법을 이용한 웹 크롤러를 설계한다.

본 논문의 구성은 다음과 같다. 2 장에서는 웹 크롤러의 개념 및 기본 동작 알고리즘, 로봇 배제 규약, URL 스케줄링, 그리고 RTT 에 대하여 기술한다. 3 장에서는 본 논문에서 제시하고 있는 URL 스케줄링 기법을 이용한 URL 스케줄러의 설계에 대하여 구체적으로 기술한다. 4 장에서는 RTT Scheduling 기법을 이용한 웹 크롤러를 설계하고, 마지막으로 5 장에서는

결론 및 향후 연구과제에 대하여 기술한다.

2. 관련연구

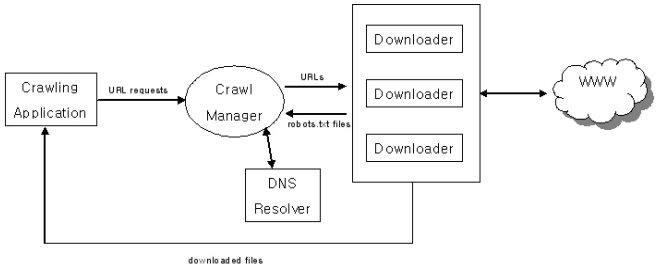
본 장에서는 웹 크롤러의 개념 및 기본 동작 알고리즘, 특정 웹 서버에 부하가 편중되는 것을 방지하기 위한 로봇 배제 표준, RTT, URL 스케줄링 기법에 대하여 기술한다.

2.1 웹 크롤러

웹 크롤러는 웹 서버를 순회하며 원하는 정보를 얻기 위해 웹 상의 문서를 수집하고, 수집된 문서를 분석하고 그 안의 URL 들을 추출하여 다른 URL 로 이동하면서 정보를 가지고 오는 프로그램이다. 이러한 웹 크롤러는 웹 로봇(robot), 웹 스파이더(spider)등으로 불려지기도 한다[1].

웹 크롤러의 기본 동작 알고리즘을 살펴보면 먼저 최초 수집하고자 하는 seed URL 을 이용하여 해당 웹 서버에 접근한다. 웹 서버의 robots.txt 파일을 확인하여 로봇 배제 규약을 분석한 후에 URL 에 해당하는 HTML 문서를 다운로드 한다. 수집된 문서의 중복 체크 후, 문서내의 URL 을 추출해서 URL DB 에 저장하고, 다운 받은 문서는 문서 DB 에 저장하는 과정을 반복 수행한다.

간단한 크롤러의 구성은 (그림 1)과 같다.



(그림 1) 기본적인 크롤러의 구성

2.2 로봇 배제 규약 (Robot Exclusion Protocol)

많은 검색 엔진이 개발되면서 많은 페이지를 보유하고 있는 특정 서버에 여러 개의 크롤러의 접근으로 인해 과도한 부하가 걸리는 문제가 발생하게 되었다. 이러한 문제의 방지를 위한 접근 금지 규약의 필요성이 대두되면서 로봇 배제 규약이 제안되었다[4]. 이것은 각 웹 호스트에 robots.txt 라는 파일을 저장하여 크롤러가 인덱스 할 필요가 없는 문서에 대한 정보를 미리 제공하는 것이다. 따라서 크롤러는 각 호스트로부터 robots.txt 라는 파일이 있는지를 먼저 확인하고 그 파일을 참조하여 CGI 나 임시 문서에 대한 인덱싱을 피할 수 있는 것이다. 따라서 크롤러 설계시 로봇 배제 규약에 대한 부분을 고려해야 한다.

2.3 URL Scheduling

URL 스케줄링은 크롤러의 항해 전략이다. 인터넷에 존재하는 문서는 대부분 하이퍼링크를 이용하여 다른 문서와 연결되어 있는데 인덱싱을 위해서는 하나의 문서에서 출발하여 그 문서 내에 있는 여러 링크를 어떠한 순서로 검색할지 결정해야 한다. 일반적으로 사용되고 있는 URL 스케줄링 알고리즘의 예는 아래와 같다[8].

- URL 자체 (도메인, 길이, 등)
- 해당 URL 로 링크되는 페이지의 개수
- 해당 URL 로 링크되는 페이지의 중요도
- 구글의 PageRank 와 같은 URL 의 대중성 (popularity)

2.4 Round Trip Time (RTT)

RTT 는 송신단이 데이터를 전송하여 해당 전송에 대한 응답을 받을 때까지의 소요된 시간을 나타낸다. 본 논문에서는 RTT 를 고려한 URL 스케줄링을 제안한다. 인터넷 상의 수많은 웹 호스트들로의 RTT 가 상이 하기 때문에 이를 고려하여 RTT 가 짧은 호스트의 URL 에 가중치를 주어 URL 스케줄링을 한다.

3. URL Scheduler

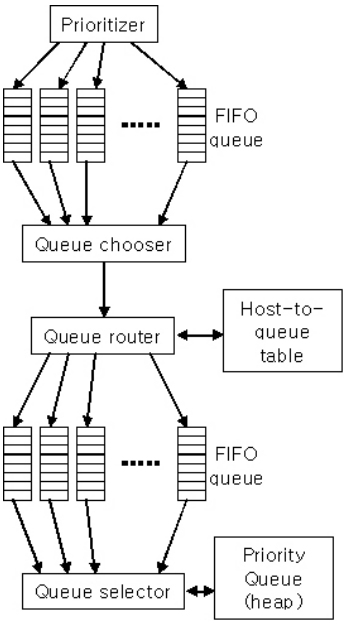
웹 크롤러는 미리 다운로드 할 URL 의 목록을 유지하고 있어야 한다. 그리고 효율적인 문서 수집을 위해 저장되어 있는 URL 스케줄러가 적합한 스케줄링 기법에 의해 스케줄링을 한다. URL 스케줄링을 위해서 속도, politeness, 로봇 배제 표준 등을 고려해야 한다.

대부분의 크롤러는 넓이 우선 탐색 기법을 채택하고 있는데, 이는 FIFO 큐로 쉽게 구현할 수 있다. 이때 같은 호스트 명을 가지는 URL 을 하나의 큐에 포함하는데, 만약 여러 개의 크롤링 스레드가 하나의 큐에서 URL 을 가져와서 문서를 다운로드 한다면 해당 웹 서버에 부하가 가중되게 된다. 이는 특정 웹 서버에 대한 HTTP request 의 수를 제한함으로써 해결 가능하다. 특정 시간에 단지 하나의 스레드가 특정 웹 서버에 접근하도록 하는 것이다. 이를 'weak politeness guarantee'라고 한다. 하지만 각각의 크롤링 스레드가 수 천 만개의 문서를 수집하기 때문에 여전히 'politeness'문제가 존재하는데 이를 위해서 더욱 강력한 politeness guarantee 가 필요하다.

이러한 politeness 문제를 해결하기 위해 본 논문은 속도를 고려한 우선순위 기반의 URL 스케줄러를 제안한다. 이것은 URL 들의 우선순위를 부여하는 prioritizer, FIFO 큐, host-to-queue 테이블, 각 FIFO 큐로의 핸들을 포함하는 힙 구조의 우선순위 큐, 그리고 'strong politeness guarantee'를 위한 부분으로 구성된다. (그림 2)

하나의 URL 이 URL 스케줄러에 들어오면 prioritizer 가 URL 과 다운로드 히스토리에 근거하여 1 부터 k 까지의 우선순위를 부여하고, URL 은 1 부터 k 까지의 우선순위가 부여된 알맞은 큐에 추가 된다. 큐

chooser 는 우선순위에 근거하여 큐를 선택해 이를 큐



(그림 2) URL Scheduler

라우터에 전달하고, 큐 라우터는 해당 URL 을 host-to-queue 테이블을 참조하여 하위의 FIFO 큐에 추가하게 된다. Host-to-queue 테이블은 각 호스트에 대한 큐로의 매핑 정보를 저장한다. 마지막으로 큐 셀렉터에서 호스트에 대한 RTT 정보를 가지는 힙 구조의 우선순위 큐를 참조하여 최종적으로 다운로드 할 URL 을 선택한다.

크롤러가 URL 스케줄러에서 URL 을 획득하는 과정은 (그림 3)과 같다.

1. 크롤링 스레드가 힙에서 루트 아이템 선택
2. 일치하는 하위 FIFO 큐에서 URL 선택
3. 크롤링 스레드는 해당 URL의 문서를 다운로드
4. 다운로드가 완료되면 URL을 큐에서 제거
5. 하위 FIFO 큐가 비게 되면 큐 chooser가 상위의 큐를 선택해서 채움
6. 하위 FIFO 큐가 선택된 URL과 호스트 명이 같으면 URL을 추가하고 다르면 host-to-queue 테이블 업데이트 후 URL 추가
7. 스레드는 힙에 RTT와 함께 큐의 핸들 추가

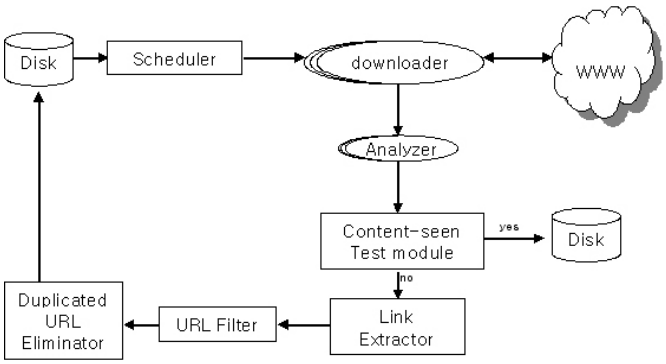
(그림 3) URL 획득 단계

4. 시스템 설계

본 논문에서 제안하는 크롤러는 대용량 인터넷 환경에서 신속한 정보 수집을 목표로 하고 있으며 아래와 같은 특징을 지닌다.

멀티 스레드 기반의 문서 수집으로 문서 수집능력을 향상시킨다. 그리고 로봇 배제 표준과 로봇 설계시에 지켜야 할 원칙을 따름으로써 크롤러를 작동시키는데 사회적, 기술적으로 문제가 발생될 수 있는 소지를 제거한다. 또한 효율적인 스케줄링 정책을 채택한다. 먼저 각각의 URL 에 우선순위를 부여하여 신뢰성 있는 정보 수집을 할 수 있고, 문서 수집 시 각각의 호스트에 대한 RTT 를 반영함으로써 문서 수집 속도의 향상을 이룰 수 있다. 그리고 다른 시스템과 달리 관리자의 개입을 많이 필요로 하는 크롤러에 명령어 상태와 웹에서 작동할 수 있는 인터페이스를 제공하여, 크롤러의 관리에 있어서의 많은 편리성을 제공하도록 한다.

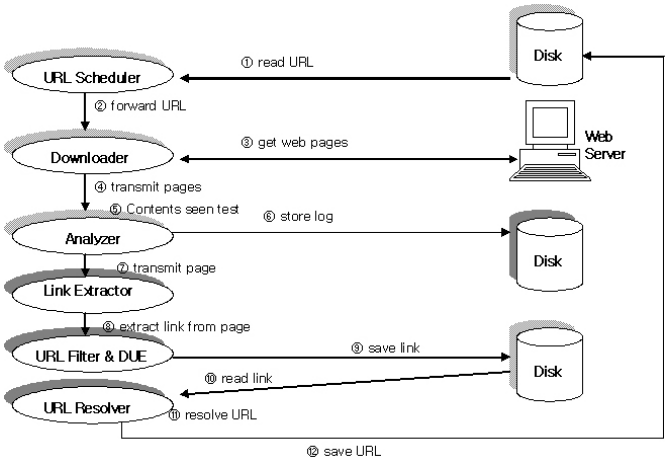
크롤러의 전체적인 구성도는 (그림 4)와 같다.



(그림 4) 크롤러 구성도

본 크롤러는 다운로더, 문서 분석기, 링크 추출기, URL 필터, 중복 URL 제거기, 스케줄러로 구성된다.

다운로더는 스케줄러에서 URL 을 가져와 해당 서버로의 HTTP connection 을 설정하고 URL 검색과 HTTP 헤더를 분석해서 “200 OK”를 확인하고 문서의 바디를 다운로드 한다. 만약 실패하면 다음 URL 을 가져온다.



(그림 5) 크롤러 동작 시나리오

문서 분석기는 전달 받은 문서를 분석하고 content-seen test module 를 통해 다운 받은 문서가 이전의 문서와 동일한지를 검사한다. 검사결과 서로 다른 URL 을 가지지만 문서의 내용이 같으면 이를 대한 정보를

디스크에 저장하게 된다.

링크 추출기는 문서 내의 모든 링크를 추출하여 URL 필터에 전달하고, URL 필터는 먼저 해당 URL 이 다운로드 할 수 있는 URL 인지를 검사해서 HTML 텍스트만 중복 URL 제거기에 전달한다.

중복 URL 제거기는 URL 의 중복 여부를 검사해 중복 되지 않은 URL 만 스케줄러에 전달한다.

스케줄러는 제안된 스케줄링 정책에 의해 다운로드 할 URL 을 다운로더에 전달하는 역할을 담당한다.

전체적인 크롤러의 동작 시나리오는 (그림 5)에 나타나는 것과 같다.

5. 결론 및 향후 연구과제

웹에 산재해 있는 다양한 정보들을 효율적으로 수집하기 위해 웹 크롤러를 이용한 많은 검색 시스템들이 제시되고 있다. 본 논문에서는 제한된 시스템 하에서 신속하게 신뢰성 있는 정보들을 수집하기 위한 스케줄링 정책을 제시하고, 이를 이용한 웹 크롤러를 설계하였다.

크롤러는 제안된 스케줄링 정책에 의해 각 URL 에 우선 순위를 부여하여 보다 신뢰성 있는 정보를 수집 가능하게 하고, 여러 개의 큐를 두어 특정 호스트로의 부하 집중을 1 차 분산시키고, 각 호스트로의 접근 시간 기록을 유지 반영함으로써 크롤러로 인한 호스트의 네트워크 부하를 2 차 분산 시키는 효과를 기대할 수 있다. 또한 웹 호스트로의 왕복시간을 스케줄링에 반영하여 좀더 신속하게 정보수집을 가능하도록 하였다. 웹을 통한 인터페이스를 제공함으로써 쉽게 크롤러를 제어할 수 있도록 한다.

향후 크롤러의 구현을 통해 제안된 URL 스케줄링에 대한 효율성을 기존의 스케줄링 알고리즘들과 비교 분석하고, 좀더 효율적인 스케줄링을 위해 수집된 정보의 신선도를 고려한 round-robin, change-frequency-based, sampling-based[7] 등의 다운로드 정책을 적용에 대한 연구가 필요하다.

참고문헌

[1] V. Shkapenyuk and T. Suel. “Design and implementation of a high-performance distributed web crawler” In Proc. of the Int. Conf. on Data Engineering, February 2002.

[2] S. Brin and L. Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine” Proc. 7th Int’l World Wide Web Conf. (WWW98), ACM Press, New York, 1988, pp. 107-117.

[3] G. L. McLearn. “Autonomous cooperating web crawlers” 2002.

[4] A. Heydon and M. Najork. “High-performance web crawling” Technical report, SRC Research Report, 173, Compaq System Research Center, September 2001.

[5] J. Cho and H. Garcia-Molina. “Parallel Crawlers”. In Proceedings of the Eleventh International World Wide Web Conference, 2002.

[6] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. “Ubicrawler: A scalable fully distributed web crawler” In Proc. AusWeb02. The Eighth Australian World Wide Web Conference, 2002.

[7] J. Cho and A. Ntoulas. “ Effective Change Detection Using Sampling” In Proc. 28th International Conference on Very Large Database(VLDB), September 2002.

[8] http://www.it.lth.se/

[9] 한국전산원 “해외 정부기관 디렉토리 DB 구축 개발완료 보고서” 연구보고서, 1999.