고성능 웹크롤러의 설계 및 구현

(Design and Implementation of a High Performance Web Crawler)

권성호*, 이영탁**, 김영준***, 이용두****
(Chenghao Quan, Youngtak Lee, Youngjun Kim, Yongdoo Lee)

요 약 웹크롤러는 인터넷 검색엔진을 포함한 다양한 웹 응용프로그램에 활용되는 중요한 인터넷 소프트웨어 기술이다. 인터넷의 급격한 성장에 따라 고성능 웹크롤러의 구현이 시급히 요구되고 있다. 이를 위해서는 웹크롤러에 대한 성능확장성에 초점을 둔 연구가 수행되어야 한다. 본 논문에서는 병렬 프로세스 기반 웹크롤러(Crawler)의 성능향상에 필수적인 동적 스케줄링의 구현 기법을 제안한다. 웹크롤러는 웹문서의 수집 성능요구를 만족시키기 위하여 일반적으로 다중 프로세스 기반으로 설계되고 있다. 이러한 다중 프로세스 기반의 설계에서 프로세스 별로 문서수집 대상을 적정하게 선택하여 할당하는 크롤 스케줄링(Crawl Scheduling)은 시스템의 성능향상에 매우 중요한 요소이다. 본 논문에서는 먼저 크롤 스케줄링에 있어 중요한 문제점들에 대한 연구 결과를 제시한 후 공유메모리 기반 동적 스케줄링 지원 기법을 고안, 이를 구현하는 웹 크롤러 시스템 구조(Architecture)를 제안하고 웹 로봇의 수행동작에 대한 분석 결과를 제공한다. 이러한 분석 결과를 기반으로 향후 웹 크롤러의 성능향상을 위한 설계 방향을 제시한다.

핵심주제어: 웹크롤러, 동적 스케줄링, 공유 메모리

Abstract A Web crawler is an important Internet software technology used in a variety of Internet application software which includes search engines. As Internet continues to grow, implementations of high performance web crawlers are urgently demanded. In this paper, we study how to support dynamic scheduling for a multiprocess-based web crawler. For high performance, web crawlers are usually based on multiprocess in their implementations. In these systems, crawl scheduling which manages the allocation of web pages to each process for loading is one of the important issues. In this paper, we identify issues which are important and challenging in the crawl scheduling. To address the issue, we propose a dynamic crawl scheduling framework and subsequently a system architecture for a web crawler with dynamic crawl scheduling support. And we analysed the behaviors of Web crawler. Based on the analysis result, we suggest the direction for the design of high performance Web crawler.

Key Words: Web Crawler, Dynamic Scheduling, Shared Memory

1. 서 론

인터넷의 발전과 더불어 검색엔진[1,2,3]의 발전 또한 커다란 변화를 가져왔다. 인터넷이 처음 등장할 당시에는 텍스트 위주의 정보교환이나 의사전달과 같은 커뮤니케이션의 기능 위주이었기 때문에 검색엔진 또한 단순히 문서를 수집해서 분류한 후 실시간으로 제공하는 것이

* 대구대학교 정보통신공학과 박사과정
** 동국대학교 정보통신공학과 박사과정
*** 대구대학교 정보통신원 정보화개발팀
**** 대구대학교 정보통신공학부 교수

주된 역할이었다. 그러나 현재는 인터넷기술의 급격한 발전과 이용자들의 증가 및 다양한 요구로 인해서 실시간 정보 제공과 이용자의 특성을 분석한 후 이용자가 원하는 정보를 카테고리별로 서비스 할 수 있다.

검색엔진의 근간이 되는 웹크롤러[4,5,6,7]는 인터넷상에 존재하는 웹 문서들을 추적하여 필요한 정보를 수집하는 기술을 말하며 야후와 같은 인터넷 검색시스템 산업, 전자상거래 상품검색 산업, 인트라넷 검색 소프트웨어 산업 등 현재 대부분의 인터넷 산업의 근간이 되는 핵심 기술이다.

인터넷이 계속적으로 성장함에 따라 그 웹 문서의 양이 매우 거대해지면서 기존의 단일시스템 환경 기반의 웹크롤러 기술은 한계에 직면하고 있다. 기존 국내 기술 수준으로는 400만 웹 문서의 데이터를 수집하는 데 약 20일 정도가 소요되고 있으며 데이터 량이 증가할 경우에는 더욱 많은 시간이 소요될 것으로 예상된다 [8,9,10,11]. 향후 5년 이후 국내의 경우 문서 량이 약 1억 페이지가 될 경우 수집에 약 4,600시간이 소요될 것으로 추산되면서 기존의 웹크롤러에 대한 성능향상은 불가피하다.

본 논문에서는 수집 속도를 최대화시킬 수 있는 공유메모리를 이용한 동적 스케줄링 기법과 아울러 상대 시스템과 네트웍 과부하를 막기 위한 동적 부하 분산 기법을 제안한다. 또한 그 구현에 관한 가능성(Feasibility), 정확성(Correctness) 등에 관한 연구를 수행하고 구현된 시스템의 수행 동작 분석에 대한 결과를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 웹크롤러 시스템의 구조와 특징을 살펴보고 3장에서는 실제 구현에 대하여 상세히 설명하고 4장에서는 수행 동작 분석에 대한 결과를 제시한다. 마지막으로 5장에서 결론을 맺는다.

2. 제안하는 웹크롤러 시스템

2.1 설계 고려사항

인터넷에서 정보검색 서비스를 제공하려면 일단 웹 문서를 수집해야 한다. 원리적으로는 아주 간단하지만 실제로 구현하기 위해서는 몇 가지 어려운 문제를 해결해야 한다. 우선 끊임없이 변하는 웹에 대한 최신 정보를 제공하기 위해 다운로드 성능을 높이는 "속도 문제"와 한번 다운로드 받은 문서를 다시 다운로드 하지 않게 막도록 하는 "유일성 문제" 그리고 과거에 다운로드 받은 문서가 아직도 유효한지, 혹은 변해서 없어졌는지를 알아내는 "현행화 문제"가 있다. 그 외에도 크롤러배제 표준과 저작권 문제를 해결해야 한다[12,13,14].

웹크롤러 시스템의 설계 및 구현에 있어 기본적으로 다음과 같은 기본 사항을 고려하여야 한다.

웹크롤러의 기본적인 요구사항으로 문서 수집 속도가 빠른 웹크롤러가 가장 성능이 우수한 웹크롤러라고 말할 수 있다. 동일한 네트워크 환경 하에서 웹크롤러가 문서 수집을 할 경우 성능을 향상할 수 있는 방법은 웹크롤러 시스템 내의 처리 속도를 최대화시키는 방법뿐이다. 웹크롤러 시스템의 성능을 향상시키는 방법으로는 일반적으로 멀티 프로세스를 사용한다.

멀티 프로세스로 동작하는 웹크롤러 시스템에서 여러 개의 프로세스가 동시에 한 서버에 문서를 요청한다면 [15,16,17] 상대 시스템과 네트웍에 과부하를 걸리게 한다. 웹크롤러는 상대 시스템과 네트웍의 과부하를 방지하기 위해 동일 시간에 단일 프로세스만 접근하도록 해야 한다[18].

과부하를 방지하기 위한 또 다른 방법으로 배제 규약이 있다. 크롤러배제 규약은 상대 시스템에 계속적인 접근으로 인해 과부하를 주거나, 일시적인 정보, CGI 등의 적당치 않은 부분을 검색하는 것을 막아보고자 로봇배제 표준이 제안되었다[13]. 웹크롤러 웹서버의 입장으로 가장 간단한 방법으로 "robots.txt"파일을 생성하는 것이다.

2.2 제안 모델의 특징

현재 웹크롤러의 문서 수집 속도, 신규URL에 대한 빠른 검색, 등의 요소들이 검색엔진을 평가하는 기준으로 사용되고 있으므로 전술한바와 같이 대용량의 웹 문서를 신속하게 수집하는 웹크롤러 시스템에 대한 연구가 불가피하게 되었다. 본 논문에서는 공유메모리를 이용하여 적재기의 프로세스가 동적으로 URL을 할당하여 문서를 수집할 수 있는 동적 스케줄링 방식을 사용하였고, 문서 적재 시 상대 시스템과 네트웍 과부하 문제를 부하 분산 기술로 해결하였다.

멀티프로세싱문서처리 속도의 향상을 위한 각 모듈들을 다중 프로세스로 처리할 수 있도록 하였다. 또한 공유메모리 데이터의 동기화를 위해 공유메모리에 접근할 때는 항상 락(lock)을 사용한다.

동적 스케줄링각 적재기 프로세스마다 정적으로 URL

을 할당할 경우 각 프로세스간의 종료 시간이 차이가 있으므로 적재기 프로세스가 URL을 동적으로 할당받아 수행할 수 있도록 하였다.

동적 부하 분산한 개 이상의 프로세스가 동일한 호스트에 접근하면 상대 시스템과 네트웍에 과부하가 걸리므로 각 프로세스가 할당받은 URL의 호스트(host)가 다른 프로세스에서 사용하고 있는지 검사하고, 만약 사용하고 있다면 새로운 URL을 할당받아 문서를 수집할 수 있도록 하였다.

본 절의 나머지 부분에서는 제안하는 웹크롤러 시스템의 구성, 동작 모델 그리고 세부구현에 관하여 기술한다.

2.3 제안 모델의 구조

최근 연구에서 웹크롤러의 성능을 판단할 때 가장 먼저 고려해야 할 점은 문서 수집 속도를 최대화하면서 상대 시스템과 네트웍에 과부하를 주지 않는 것이다. 본 논문에서는 이를 위하여 공유메모리를 기반으로한 동적 스케줄링 기법을 제안하여 속도를 향상하였고 동적 부하 분산을 보장한다.

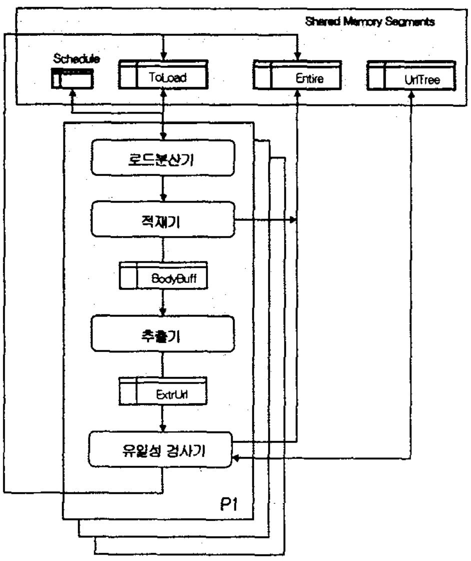


그림 1. 제안 모델 구조

메모리는 크게 공유 메모리와 프로세스 메모리 영역으로 구분된다. 그림1 에서와 같이 공유메모리는 *Entire*, *UrlTree*, *ToLoad*, *Schedule* 버퍼가 있고, 프로세스 메모리는 *BodyBuff*, *ExtrUrl* 버퍼가 있다. *Entire*는 전체 URL 정보 저장하기 위해, *UrlTree*는 유일성 검사를 위해, *ToLoad*는 적재할 URL 정보 저장을 위해 사용되며, *Schedule*는 부하 분산을 위해 사용된다. 각 프로세스는 부하 분산기, 적재기, 추출기, 유일성 검사기로 구성되며, 공유 메모리의 데이터를 처리하게 된다.

2.4 제안 모델의 동작

각 프로세스는 문서 수집을 위해 *ToLoad URL*에서 하나의 URL을 패치한 다음 부하분산을 위해 *Schedule* 버퍼를 사용한다. 그림 2와 같이 각 프로세스의 URL 할당은 다음과 같은 순서로 이루어진다.

① URL 하나를 패치한다.

② 패치한 URL의 호스트가 *Schedule*버퍼에 존재하는지 검사한다.(존재 한다면 현재 해당 호스트에 대해 문서를 수집중이라는 의미이다.)

③ 존재하지 않는다면 *Schedule*버퍼에 호스트 삽입, 해당 URL 문서를 수집한다.

각 프로세스는 문서 수집을 위해 *ToLoad URL*에서 하나의 URL을 패치한 다음 부하분산을 위해 *Schedule* 버퍼를 사용한다.

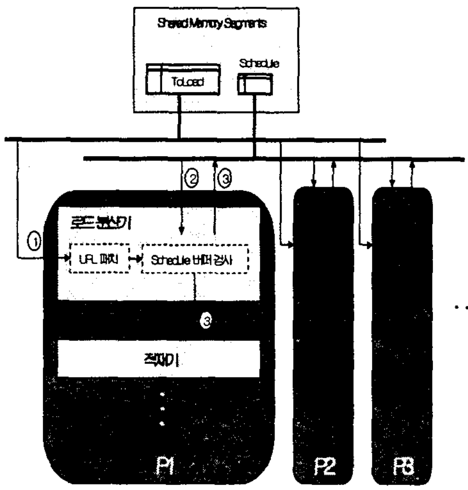


그림 2. 제안 모델 동작

3. 웹크롤러 시스템의 구현

3.1 부하 분산기

한개 이상의 프로세스가 동시에 한 호스트에 접근하게 된다면 상대 시스템과 네트웍에 과부하를 주게 된다. 이를 방지하기 위해 제안하는 시스템에서는 *Schedule* 버퍼를 사용한다. *Schedule* 버퍼는 *char Schedule[process number][256]*로 선언된 2차원 배열이다.

부하 분산기는 *ToLoad*버퍼에서 URL을 하나 패치한 다음 *Schedule* 버퍼에 패치해온 URL의 호스트가 존재하는지 검사한다. 만약 존재하지 않는다면 해당 호스트를 *Schedule*에 삽입하고 URL의 문서를 적재한다. 존재한다면 다시 *ToLoad*버퍼에서 URL을 한 개 패치하고 *Schedule*버퍼와 비교하는 과정을 되풀이한다. 이 과정에서 동일한 호스트는 *Schedule*버퍼에 존재하지 못하게 된다. 즉 동시에 2개 이상의 프로세스가 한 호스트에 접근하지 못하게 된다.

3.2 적재기

적재기는 웹크롤러의 가장 필수적인 모듈로서 URL을 입력으로 하여 해당 URL의 호스트에 소켓 접속한 다음 웹 서버에 URI를 요청하게 된다. 웹 서버는 요청된 URI를 검색한 다음, 검색 결과에 따라 문서의 헤드를 웹크롤러 시스템에 전송해 준다. 웹크롤러는 적재한 헤드를 파싱하여 유용한 문서인지를 검사하고, 유용한 문서이면 본문의 내용을 적재한다. 적재한 문서는 웹크롤러의 로컬시스템에 저장된다.

3.3 추출기

추출기의 주요 기능은 웹 문서 파싱이다. 웹 문서 파싱의 가장 큰 문제점은 사용자들이 HTML 문법에 맞지 않는 코드를 생성해 놓기 때문에 사용자의 잘못된 코드까지 충분히 고려해야 한다는 것이다. 제안 시스템에서는 이러한 문제를 해결하기 위해 문법에 맞지 않는 코드까지 지원할 수 있도록 구성되어 있다.

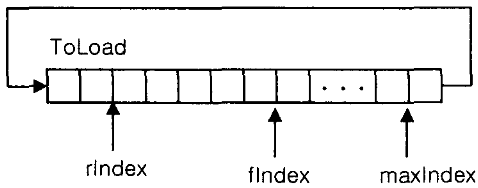


그림 3. ToLoad 원형큐

추출기는 적재기에서 다운로드 받은 *BodyBuff*에서 태그(Tag)만을 추출하여 만든 태그 리스트(Tag List)를 작성한다. 그리고 작성된 태그 리스트에서 웹 문서 참조(Link)를 추출하여 *ExtrUrl*에 삽입한다.

3.4 유일성 검사기

웹크롤러는 한번 수집한 문서를 다시 수집하는 일이 없어야 한다. 수집한 문서는 URL DB에 저장하고 새로운 URL이 추출되면 URL DB를 검색하여 존재하지 않는 URL만 적재기의 입력으로 삽입해야 한다.

웹크롤러는 다중 프로세스가 동작하므로 URL 유일성 검사를 하는 동안 다른 프로세스는 유일성 검사를 하면 데이터 동기화 문제가 발생한다. 유일성 검사를 하기 위해서는 먼저 URL DB에 lock을 걸고, 유일성 검사를 한 다음 다시 락(Lock)을 해제한다. 동일시간에 하나의 URL만 유일성 검사가 가능하므로 전체적인 시간 지연이 발생하게 된다.

제안 시스템에서는 유일성 검사의 시간을 줄이기 위해 다중 루트(Root) 이진 트리 기법을 사용하고 있다. 다중 루트는 해쉬(Hash)로 구성되어 있다. 그러므로 하나의 URL 중복검사를 위해 그 Url의 해쉬 키(Hash Key)를 구한다음 다중 루트에서 해쉬 키에 해당하는 부분만 락이 걸리게 된다. 따라서 해당 URL의 유일성 검사를 위해 하나의 해당 URL의 해쉬 키를 제외한 나머지 루트에서 동시에 중복 검사가 가능하다.

URL DB를 이진트리로 만든 *UrTreeHash*또한 *ExtrNewHash*와 동일한 Hash를 구조를 가지고 있으므로 *ExtrNewHash*의 *n*번째 *Entry*에 락을 거는 것은 *UrlTreeHash*에 *n*번째 *Entry*에 락을 거는 것과 동일하다.

*Hash Root*는 128byte로 구성되어 있고 해쉬 키 생성 알고리즘은 다음과 같다. 추출기에서 추출된 *ExtrUrl*을 입력으로 받아 해쉬 키 생성 알고리즘을 사용하여 *ExtrNewHash*를 만든다. *ExtrNewHash*의 1부터 128번째 *Entry*까지 차례로 lock을 걸고 유일성 검사를 한 다음 락을 해제 한다. 유일성 검사에서 통과된 URL들을 *ProcToBuf*에 저장하고, *ToLoad*에 락을 건 다음 *ProcToBuf*의 Url들을 *ToLoad*에 삽입하고 *ToLoad*의 락을 해제하면 유일성 검사가 끝난다.

```
#robots.txt for lactt.taegu.ac.kr
User-agent: *
Disallow: /admin/
User-agent: NSS.10
Disallow:
```

그림 4. robots.txt 파일 사례

3.5 ToLoad 버퍼 접근 알고리즘

ToLoad 버퍼는 신규로 추출된 Url을 삽입하고, 적재기에서 *ToLoad*에 있는 Url을 다운로딩한다. 그림 3에서와 같이 신규로 추출된 Url을 적재할때는 *fIndex*를 사용하고, Url을 패치할 때는 *rIndex*를 사용하는 원형큐로 구성되어 있다.

3.6 로봇배제 표준

robots.txt에 기술되는 크롤러배재의 포맷은 필드와 값의 쌍으로 기술되며 주석문은 "#"으로 처리한다. 필드는 두 가지로 나눌 수 있는데 에이전트의 이름을 적는 User-agent 필드와, 방문하지 말아야할 URI를 기술하는 Disallow 필드로 구분된다. 또한 빈 robots.txt는 모든 웹크롤러의 접근을 허락한다는 의미이다.

그림 4는 크롤러배제 표준의 사례 파일로서 User-Agent의 이름이 NSS1.0인 웹크롤러를 제외한 모든 웹크롤러는 "/admin/"으로 시작하는 페이지의 접근을 제한한다는 것이다.

로봇배제 규약을 준수하기 위해서는 상대시스템의 웹 문서뿐만 아니라 robots.txt 파일도 다운로드하고 이를 분석하여 해야 한다. 즉 문서 하나를 수집하기 위하여 매번 robots.txt파일을 다운로드 한다면 웹 서버에 많은 부하를 줄 뿐만 아니라 문서수집 속도를 감소시킨다. 이 문제를 해결하기 위하여 호스트에 대한 로봇배제 정보를 캐싱하여 저장한다.

4. 웹클로러의 수행동작 분석

본 절에서는 앞에서 설명한 웹 크롤러 시스템에 대하여 그 수행동작(Execution Behavior)에 대한 상세 분석결과를 기술한다.

웹 크롤러의 성능은 크게 수집효율(Collection efficiency)과 수집속도(Collection speed)로 구성된다. 수집효율은 수집 가능한 URL에 대하여 얼마나 많은 URL들을 수집하는 가를 나타내며 수집속도는 얼마나 신속하게 수집을 수행하는가를 나타내는 척도이다. 수집효율은 주어진 수집대상 URL에 대하여 수집에 성공한 URL의 비율로, 수집속도는 단위 시간당 수집한 URL의 개수로 나타낼 수 있다.

본 웹 크롤러의 수행동작에 있어 소켓 접속을 위한 타임아웃(Timeout) 크기가 수집효율 및 수집성능에 미치는 영향을 분석하였다. 또한 웹 크롤러의 각 기능 모듈별 수행시간이 전체 수행시간(Turn around time)에서 점유하는 비율을 산출하여 분석하였다.

본 수행동작의 분석에는 실제 실험을 통해서 얻은 데이터, 즉 2절의 웹 크롤러모델을 소프트웨어로 구현한 후 실제 하드웨어 시스템에서 구동시켜 얻은 데이터를 활용하였다. 본 실험에 활용한 하드웨어 시스템은 Hewlett Packard 사의 PC 서버로서 2개의 Pentium III 731MHz CPU와 2G 메모리, 90G Hard Disk Array를 정착하고 있다. OS는 Linux RedHat 6.2 버전을 사용하였다. 본 서버는 10M Ethernet 카드를 장착하고 있으며 인터넷 외부망에 T3 용량으로 연결되어 있는 네트웍에 위치하고 있다.

본 수행동작 분석을 위한 웹 크롤러문서 수집대상으로 국내 대학교 웹사이트 10개를 임으로 선정하였으며 전체 수집 대상 URL의 개수는 약 5만개이며 문서 수집 시에 생성시킨 로그파일에 근거로 하여 분석결과를 산출하였다.

웹 크롤러의 수집은 동일한 네트워크 환경에서 밤 12시부터 같은 조건으로 프로그램을 구동하여 시간별 문서 수집량을 측정하였다.

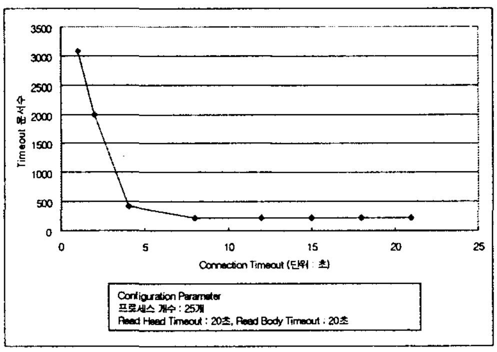


그림 5. 접속요청 시 Timeout 값과 응답실패 문서 수

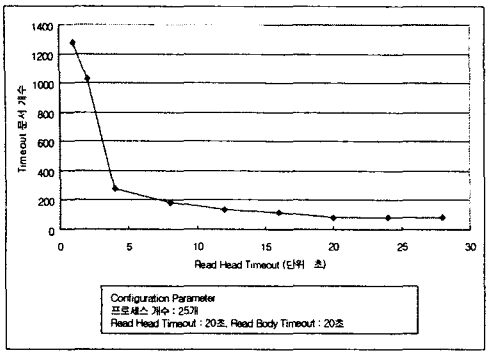


그림 6. 문서헤드 요청 시 Timeout 값과 응답실패 문서 수

4.1 소켓 타임아웃(Timeout) 영향 분석

웹 크롤러의 적재 프로세스는 자료수집 과정에 웹 서버에와 상호동작을 반복·수행한다. 이러한 상호동작은 크게 웹 크롤러으로부터 웹 서버로의 접속요청(Connection request), 문서 헤더 요청(Read head request), 문서본문 요청(Read body request)으로 구성된다.

이러한 요청은 기본적으로 소켓 통신을 기반으로 구현되며 주어진 웹 크롤러의 적재 프로세스 는 요청을 전송한 후 응답이 도달할 때까지 수행을 멈추고 대기 상태에 놓여 있게 된다. 만약 웹 서버 측에서 웹 서버, 해당 호스트, 네트웍 등의 장애로 인하여 정상적인 응답이 가능하지 않은 경우 최대 180초까지의 응답지연이 발생하게 된다. 이러한 경우가 발생할 경우 웹 크롤러의 전체 수집 속도(Collection speed)가 매우 저하된다.

적재 프로세스는 Unix의 알람시스널(Alarm signal)을 기반으로 하여 요청에 대한 Timeout 크기를 조절하여

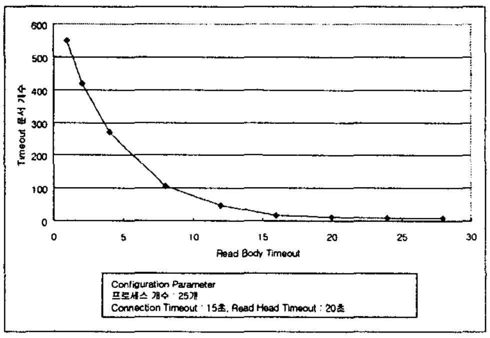


그림 7. 문서본문 요청 시 Timeout 값과 응답실패 문서 수

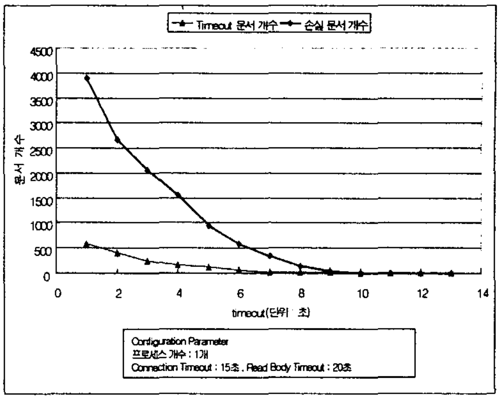


그림 8. Timeout 값과 전체 문서수집 손실양

위의 수집성능 저하문제를 해결해야 한다. Timeout 값이 매우 작을 경우에는 네트웍 또는 호스트의 과부하 또는 일시적인 서비스 지연으로 인하여 문서수집 실패를 초래하게 되어 전체문서 수집효율이 저하된다. Timeout 값이 매우 클 경우에는 시스템 또는 네트워크 장애 시, 삭제된 URL에 대한 문서수집 시에 대기 시간이 커지므로 문서 수집속도가 낮아지게 된다. 즉 수집효율과 수집속도 간에는 상충관계(Tradeoff)가 발생한다.

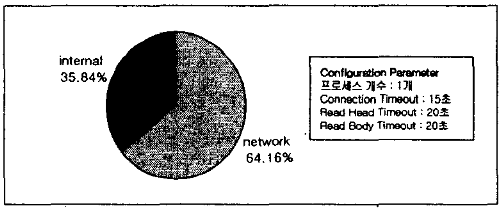


그림 9. 네트워크와 컴퓨팅 시간의 분포

본 실험에서는 위의 웹 크롤러으로부터 웹 서버로 전송하는 각 요청에 대하여 Timeout 값의 크기가 응답실패율에 미치는 영향을 분석하였다.

Timeout 값이 매우 작은 경우 접속 요청에 대한 응답실패율이 높아지게 된다. 실제로 그림 5에서 보는 바와 같이 Timeout 값이 1초 일 때 접속요청의 경우 3000개 이상의 문서에 대하여 응답실패가 발생하고 있음을 볼 수 있다. 또한 그림6과 그림7에서 보면 문서헤드 요청과 문서본문 요청을 비교할 때 Timeout 값이 매우 작은 경우 문서헤드 요청의 경우가 문서본문 요청의 경우보다 약 2배 이상의 응답실패율을 보이고 있음을 볼 수 있다.

수집효율을 최대로 보장할 수 있는 최소 Timeout 크기를 살펴보면 먼저 접속(Connection) 요청에 있어서는

Timeout 값을 8초로 이상인 경우에는 응답실패 문서수가 더 이상 감소하지 않는 것을 볼 볼 수 있다. 문서헤드 요청에 있어서는 그림 5에서 보는 바와 같이 Timeout 값이 약 25초 이상일 경우에는 응답실패 문서수가 Timeout 값에 영향을 받지 않는 최소 수준에 도달함을 볼 수 있다. 마지막으로 문서본문(Body) 요청에 대해서는 Timeout 값이 약 30초 이상이 되어야 응답실패 문서수가 Timeout 값에 영향을 받지 않고 최소 수준에 도달하게 된다.

웹 문서는 하이퍼링크(Hyperlink)를 기반으로 하여 상호 연결되어 있다. 그러므로 주어진 문서에 대하여 응답실패가 발생하면 그 문서에 링크되어 있는 하위 문서들의 수집도 대부분 가능하지 않게 된다. 그림 8은 문서헤드 요청에 Timeout 값의 변동에 따른 전체 수집실패 문서개수 나타내 주는 그래프이다. 본 그래프에 따르면 문서헤드 요청에 대한 응답실패량의 약 8배 정도에 해당하는 문서들에 대하여 수집을 실패하고 있음을 볼 수 있다.

위에서 살펴본 Timeout 값이 문서 수집효율에 미치는 영향에 대한 분석 결과로서 향후 웹 크롤러의 설계에 있어 아래와 같은 점을 고려해야 한다. 일반적으로 웹 크롤러의 Timeout 값은 요청의 종류에 상관없이 동일한 값으로 설정하여 구동시키는 경향이 많다. 그러나 앞에서 살펴본 바와 같이 접속 요청이나 문서헤드 요청, 문서본문 요청에 따라 수집효율을 최대화하는 Timeout 값은 상이하므로 웹 크롤러의 웹서버에의 요청은 그 종류 별로 Timeout 값을 적절하게 설정하여야 한다.

또한 응답실패에 따른 문서의 수집실패는 그 문서 내에 포함되어 있는 인링크 URL에 대한 문서의 수집실패를 초래할 위험성이 있으므로 Timeout값의 선정에 있어서 웹 크롤러수행 중에 응답실패율을 관찰하여 상황에 적합하도록 동적으로 최적의 값으로 변경하여 설정할 수 있도록 하는 적응형(Adaptive) Timeout 설정기법에 대한 연구도 필요하다.

4.2 세부 기능별 성능 분석

Amdahl's 법칙에 따르면 컴퓨터 시스템의 성능향상은 성능에 가장 큰 비중을 차지하고 있는 부분에 우선적인 초점을 두어야 한다. 웹 크롤러또한 수집속도을 향상시키기 위해서는 웹 크롤러의 전체 수집시간에 각 세부 기능이 차지하고 있는 비율을 먼저 파악해야 한다.

웹 크롤러의 수행시간(Turn-around time)은 크게 네트웍 시간과 컴퓨팅 시간으로 구성된다. 네트웍 시간은 웹 크롤러가 네트웍 트랜잭션에 소요하는 시간이며 접속(Connection), 문서헤드 적재, 문서내용 적재에 소요되는 시간을 포함한다. 컴퓨팅 시간은 네트웍 트랜잭션을 제외한 시간을 의미하며 부하분산, URL 추출, 유일성 검사에 소요되는 시간을 포함한다. 향후 웹 크롤러의 수집속도를 향상시키기 위해서는 이러한 세부 기능별 수행시간 점유율의 산출이 필수적이다.

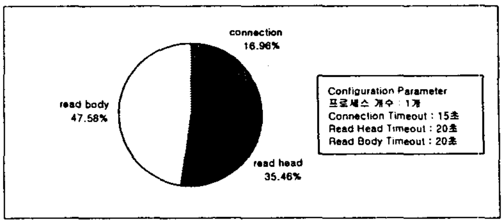


그림 10. 네트워크 시간 내의 세부기능 시간분포

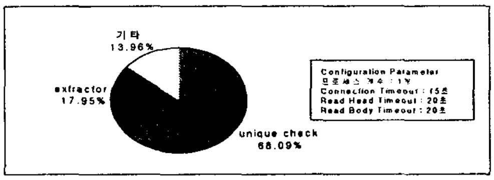


그림 11. 컴퓨팅 시간 내부의 세부기능별 시간 분포

본 실험에서는 웹 크롤러의 수행 시 생성시킨 로그를 분석하여 각 기능별 수행시간 점유율을 산출하였다. 웹 크롤러의 적재기 및 산출기 프로세스는 한 개로 설정·구동시켜 각 기능별 수행시간의 상대 크기를 산출하였다.

먼저 그림 9는 네트웍 시간 대비 컴퓨팅 시간의 비율을 보여준다. 네트웍 시간이 전체 수행시간의 64.2% 컴퓨팅 시간이 36.2%를 점유하고 있음을 알 수 있다. 그림 10은 네트웍 시간의 세부 구성을 보여 준다. 전체 네트웍 시간에서 접속(Connection), 문서헤드 적재, 문서내용 적재에 소요되는 시간은 각각 17.0%, 35.5%, 47.6%를 점유하고 있음을 볼 수 있다. 그림 11은 컴퓨팅 시간의 세부 구성을 보여 준다. 전체 컴퓨팅 시간에서 유일성 검사시간이 68.1%, URL 추출시간이 18%를 점유하고 있음을 볼 수 있다. 본 결과는 유일성 검사시간이 URL 추출시간 보다 3배 이상이 된다는 사실은 매우 의외의 결과라 할 수 있다.

위의 실험 결과를 통하여 볼 수 있는 것은 네트웍시간에서 접속과 문서헤드 적재시간을 합이 문서내용을 적재하는 시간보다 크다는 사실이다. 이러한 결과는 데이터의 전달시간 보다 네트웍 접속에 소요되는 시간이 매우 크

다는 사실을 나타낸다. 그러므로 전체 네트웍 시간을 감소시기 위해서는 문서수집 시에 필요 없는 접속 및 문서 헤드 적재 시도 횟수를 제거할 수 있도록 향후 문서 갱신 및 삭제 여부를 검사하는 문서수집 프로세스로부터 독립 프로세스를 구성하는 웹 크롤러의 설계 방식을 고려할 수 있다.

5. 결 론

본 논문에서는 웹크롤러의 성능향상과 부하분산을 효과적으로 지원할 수 있는 공유 메모리를 사용한 동적 스케줄링 기법을 제안하였다. 또한 이러한 기법을 적용한 웹크롤러의 설계 및 구현에 관한 연구 결과를 발표하였다. 이러한 구현을 바탕으로 하여 구체적인 수행동작 분석을 수행하였다. 분석결과 기존의 선험적 이해와는 달리 웹 서버로의 요청의 유형에 따라 요구되는 Timeout 값은 상이하다는 사실을 발견할 수 있었다. 그러므로 향후 웹 크롤러의 설계에 있어 각 유형별로 적합한 Timeout 값을 설정하여야 보다 향상된 성능을 얻을 수 있다. 한편 웹 크롤러수행 쓰레드(Execution thread)의 전체 컴퓨팅 소요시간에서 유일성 검사 시간이 68.1%를 차지하고 있다. 따라서 향후 고성능 웹 크롤러설계에 있어서 유일성 검사를 위한 보다 향상된 알고리즘의 개발이 이루어 져야 한다. 이러한 결과는 향후 고성능 웹크롤러의 설계를 위한 방향을 제시할 것으로 기대하고 있다.

참 고 문 헌

[1] 김 한, Design and Implementation of Real-time Contro Search Engine for Local Information Network. Kyungnam University *碩士學位*, 1998, 12.

[2] 김창근, Design and Implementation of Region Administration-Oriented Search Engine Based on Robot Control Policy. Kyungnam University *碩士學位*, 1999, 6

[3] 신동욱, 인터네트 환경에서의 분산정보 검색시스템의 설계 및 구현, *한국과학재단연구과제 961-0911-060-2*, 1998, 충남대

[4] M. Wooldridge and N. Jennings, Intelligent Agents: Theory and Practice. *Knowledge Engineering Review* 10(2), 1995

[5] Riechen, Doug, Intelligent Agents, *Comm. of the ACM* Vol. 37 No. 7, July 1994.

[6] Allison Woodruff, Paul M.Akoi, Eric Brewer and Paul Gauthier. An Investigation of Documents from the World Wide Web 1998.

[7] K. Sparck Jones, A Statistical Interpretation of Term Specificity and Its Application in Retriveal, *JD* 28(1), pp.11-20, 1972.

[8] 한국인터넷 정보센터, http://stat.nic.or.kr

[9] 송관호, 2001 한국인터넷 통계집, *한국인터넷정보센터*

[10] 박성득, 2001 한국인터넷 백서, *한국전산원*

[11] W3 Consortium, http://www.w3c.org

[12] Martijn Koster, ConneXions, Robots in the Web: threat or treat ?, Volume 9, No. 4, April 1995.

[13] Martijin Koster, A proposed standard for Robot exclusion, Published on the WWW at http://web.nexor.co.uk.

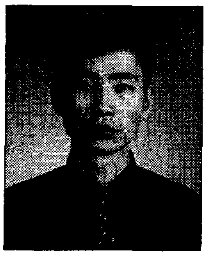
[14] Fielding, Roy. maintaining Distributed Hypertext Infostructures: Welcome to MOMspider's Web. *Proceedings of the First International World-Wide Web Conference*, Geneva Switzerland, May 1994.

[15] R. Fielding, J.Gettys, J.C. Mogul, H. Frystyk and T. Berners-Lee, *RFC 2068* Hypertext Transfer Protocol HTTP/1.1, UC Irvine, Digital Equipment Corporation, MIT, 1997.

[16] N.J. Yeager and R.E. McGrath, Web Server Technology, Morgan Kaufman Publishers, 1996.

[17] T. Berners-Lee, R. Fielding, H. Frystyk, Hypertext Transfer Protocol -- HTTP//1.0, *RFC 1945*, May 1996

[18] Michele Colajanni, et al, Dynamic Load Balancing on Web Servers systems, http://computer.org, May 1999.



권 성 호 (Chenghao Quan)

1992년 (중)연변대학교 전자공학과 졸업
2001년 대구대학교 대학원 정보통신공학과 졸업(공학석사)
2001년 3월 ~ 현재 대구대학교 대학원 정보통신공학과 박사과정
(관심분야 : 웹클롤러, 검색엔진, 컴퓨터 네트워크)

이 영 탁 (Youngtak Lee)

2000년 대구대학교 전자공학과 졸업
2002년 대구대학교 대학원 정보통신공학과 졸업(공학석사)
2002년 3월 ~ 현재 동국대학교 대학원 정보통신공학과 박사과정
(관심분야 : 웹클롤러, 검색엔진, 컴퓨터 네트워크)

김 영 준 (Youngjun Kim)

2000년 대구대학교 전자공학과 졸업
2003년 대구대학교 대학원 정보통신공학과 졸업(공학석사)
2003년 3월 ~ 현재 대구대학교 정보통신원 정보화개발팀
(관심분야 : 웹클롤러, 검색엔진, 컴퓨터 네트워크)

이 용 두 (Yongdoo Lee)

1975년 한국항공대학교 통신학과 졸업(공학사)
1982년 영남대학교 대학원 전자공학과(공학석사)
1995년 한국항공대학교 대학원 전자공학과(공학박사)
1982년 ~ 현재 대구대학교 정보통신공학부 교수
1981년 ~ 1982년 (일)동경대학 전자공학과 객원교수
1991년 ~ 1993년 Univ. of Southern California 교환교수
(관심분야 : 컴퓨터구조, Internet 응용 기술)