URL 분석을 위한 웹 로봇 구현 및 성능분석

정회원 김 원*, 김 희 철**, 진 용 옥***

Implementation and Performance Analysis of Web Robot for URL Analysis

Weon Kim*, Hiecheol Kim**, Yong Ohk Chin*** *Regular Member*

요 약

본 논문은 웹페이지(Webpage)를 효율적으로 수집할 수 있도록 기능별로 나누어서 각 구성 요소간의 의존성이 최소화될 수 있는 멀티에이전트 방식의 「웹 로봇」을 제안하고, 구현된 시험시스템의 성능분석과 우리나라의 웹페이지 언어별 구성비율과 일반문서·멀티미디어 파일 구성비율 등에 대한 분석치를 제시하는 기틀을 마련한 연구이다. 기존의 동일한 자원 환경하에서의 순차적 실행방식의 웹 로봇은 웹페이지 수집기로 활용하기에는 수집 성능의 한계가 존재한다. 즉 웹페이지의 URL은 일시적인 호스트의 다운과 네트워크의 불안정 등으로 연결이 안되는 이른바 "Dead-links" URL이 존재한다. 이 "Dead-links" URL이 많을 경우 웹 로봇은 이로 인하여 HTML 적재시간이 많이 걸리게 된다. 본 논문에서 제안하는 멀티에이전트는 "Dead-links" URL을 독립적으로 처리하도록 하여 성능 향상을 최대화하는 것이다.

ABSTRACT

This paper proposes the web robot based on Multi-Agent which the mutual dependency should be minimized each other with dividing the function each to collect Webpage. In result it is written to make a foundation for producing the effective statistics to analyze the domastic webpages and text, multimedia file composition ratio through performance analysis of the implemented system. It is easy that Web robot of the sequential processing method to collect Webpage on the same resource environment produces the limit of collecting performance. So to speak, Webpages have "Dead-links" URL which is produced by the temporary host down and unstability of network resource. If there are much "Dead-links" URL in the webpages, it takes a lot of time for web robot to collect HTML. The purpose of this paper to be proposed, makes the maximum improvement to extract the webpages to process "Dead-links" URL on the Inactive URL scanner Agent.

Ⅰ. 서 론

인터넷이 계속적으로 성장함에 따라 그 웹페이지의 양이 매우 거대해지면서 기존의 단일시스템 환경 기반의 웹 로봇 기술은 한계에 직면하고 있다. 기존 국내 기술 수준으로는 400만 웹페이지의 데이터를 수집하는 데 약20일 정도가 소요되고 있으며 데이터 양이 증가할 경우에는 더욱 많은 시간이 소요될 것으로 예상된다. 향후 5년 이후 국내의 경우 문서의 량이 약 1억 페이지가 될 경우 수집에 약 4,600시간이 소요될 것으로 추산되면서 기존의 웹 로봇에 대한 성능향상은 불가피하다. 최근의 웹 로봇은 웹페이지(Webpage)의 수집속도를 향상시키기 위하여 여러 개의 URL(Uniform Resource Locator)을 동시에 접근할 수 있는 멀티프로세스 기반의 순차적 실행방식을 채택하고 있다.

* 경희대학교 전자공학과 (wkim@nic.or.kr), ** 대구대학교 정보통신학부 조교수
*** 경희대학교 전파공학과 디지털통신연구실
논문번호 : 010292-1019, 접수일자 : 2001년 10월 19일

그러나, 동일한 시스템 자원 환경하에서의 멀티프로세스 기반의 순차적 실행방식 웹 로봇은 URL의 일시적인 호스트 다운, 네트워크 불안정 및 비활성 웹사이트 등으로 연결이 안되는 이른 바 "Dead-links" URL이 존재한다. 이 "Dead-links" URL이 많을 경우 웹페이지 적재시간이 많이 걸리게 된다.

따라서 본 논문에서는 "Dead-links" URL을 별도로 처리하면서 각 구성 요소간의 의존성이 최소화될 수 있는 멀티에이전트 방식의 「웹 로봇」을 제안하고, 그 구현에 관한 가능성, 성능 등에 관한 연구를 수행하고 수집한 웹페이지 결과를 근거하여 국내 웹페이지 언어별 구성비율, 국내 홈페이지 및 활성도메인의 구성비율 등을 제시하고자 한다.

Ⅱ. 관련연구

1. 웹 로봇

웹 로봇은 웹 서버를 순회하며 각 웹페이지에 있는 수많은 정보를 자동으로 수집하는 프로그램으로 HTML(HyperText Markup Language) 페이지를 수집한 후 이를 분석하여 그 안에 포함되어 있는 인링크(In-link) URL들을 추출한 후 추출된 각 URL별로 이동하면서 정보를 수집한다. 1996년 Martijn Koster[1]는 웹 로봇의 주요 용도를 자원 탐색(Resource Discovery), 미러링 관리(Mirroring maintenance), 링크 관리(Link Maintenance), 통계 분석(Statistical Analysis)으로 구별하였다

1.1 동작 알고리즘

문서 수집 과정을 살펴보면 먼저 수집할 문서에 대한 URL 목록을 지니고 있는 URL DB에서 한 개의 URL을 취한 후 그 URL에 대한 호스트에 접속한다. 접속이 성공적으로 이루어지면 웹 서버는 웹 로봇에 URI(Universal Resource Identifier)를 요청하게 된다. 요청된 URI에 대하여 웹 서버는 해당 문서의 헤더를 보내준다[2]. 웹 로봇은 그 헤더 정보를 파싱(Parsing)하여 유용한 문서인지를 검사한다. 유용한 문서일 경우 웹 서버에 웹 문서의 본문을 요청·적재한다. 웹 로봇은 그 웹 문서을 파싱하여 문서 내의 인링크 URL들을 추출해 낸 후에 추출된 URL들을 기 수집한 URL 목록(URL DB)에 대하여 중복검사를 수행한다. 중복검사결과 추출된 URL 중에서 URL DB에 존재하지 않는 URL들만 URL DB에 삽입한다. 웹 로봇은 위의 과정을 반복하며 문서를 수집하게 된다.

1.2 멀티프로세스 기반의 순차적 실행방식

웹 로봇은 네트워크 I/O 트랜잭션(Transaction)을 기반으로 동작하므로 여러 개의 URL에 대하여 동시 수집이 가능하다. 이러한 동시 수집을 위한 웹 로봇은 일반적으로 문서 적재 기능 및 적재된 문서들로부터 신규 URL 추출 기능을 멀티프로세스로 구동시킨다.

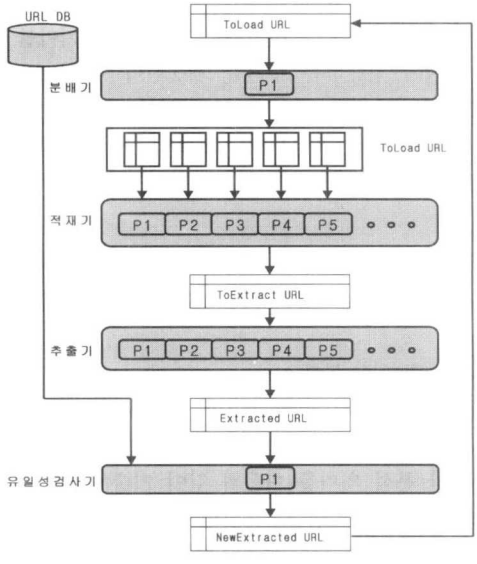

그림 2.1 멀티프로세스 기반의 순차적 실행방식

그림 2.1은 멀티프로세스 기반의 순차적 실행방식 웹 로봇 구조를 보여준다. 본 모델의 기능 모듈은 크게 적재기(Webpage Loader), URL 추출기(Extractor), 유일성 검사기로 구성된다. 웹 로봇의 구동에 있어 적재기와 추출기는 멀티프로세스로서 구동된다. 모듈들의 입출력으로서 사용되는 데이터는 크게 URL DB, ToLoad URL 목록, ToExtract URL 목록, Extracted URL 목록, NewExtracted URL 목록으로 구성된다. URL DB는 웹 로봇이 수집을 완료한 문서들에 대한 URL 목록으로서 데이터베이스 또는 물리적 파일로 구성된다. ToLoad URL 버퍼(Buffer)는 웹 로봇이 수집할 대상 URL들의 목록을 저장한다. ToExtract URL 버퍼는 적재기가 수집한 문서들에 대한 URL 및 해당 웹페이지 파일의 저장경로 등의 정보를 저장하고 있다. Extracted URL 버퍼는 추출기의 출력인 인링크 URL 목록을 저장한다. 마지막으로 New Extracted

URL은 검사를 통하여 중복 및 비신규 URL들을 제거한 비중복, 신규 인링크 URL 목록을 저장한다. 이러한 데이터 저장공간은 메모리 상의 버퍼로서 구현된다.

2. 웹페이지 분석

2.1 URL(Uniform Resource Locator)

URL[3]이란, 인터넷상의 자원의 위치를 나타내는 것으로 요약할 수 있다. 네트워크상에서 이용되는 프로토콜을 사용하여 검색할 수 있는 객체, 즉 자원의 물리적인 주소를 나타내며, 다양한 스키마를 통해 자원에 접근하는 수단을 제공한다. 흔히 도메인 이름(Domain Name)이라고 표현되는 World Wide Web 상에서의 프로토콜, 호스트이름, 포트번호와 하부의 디렉토리(Directory)와 파일이름으로 구성된다.

http:// www.nic.or.kr :8000 /english/ index.htm
프로토콜 호스트명 포트번호 디렉토리 파일명

2.2 웹페이지 분석

1996년 UCB(University of California Berkeley)의 Eric Brewer 교수 연구진은 각 웹페이지의 속성에 대한 분석 의미를 다음과 같이 기술하였다.

- Language의 확산분포 : HTML의 확산분포 인지와 널리 사용하지 않은 다른 language의 자연증가에 대한 억제 가능
- 웹페이지 개발 프로그램도구의 향상 : 새로운 웹페이지를 향상시킬 수 있는 프로그램도구에 대한 확산분포와 시험 가능
- 사회학적 분석 가능 : 웹페이지의 컨텐츠 분석을 통하여 흥미있는 사회학적 측면의 고찰 가능

웹 로봇을 이용한 웹페이지 분석의 사례로서 Inktomi Web crawler를 활용하여 수집한 제한된 약 200만개의 웹페이지를 대상으로 UCB에서 Eric Brewer 교수 연구진이 수행한 분석결과를 기술하였다[4]. UCB에서는 Inktomi의 병렬처리 Web crawler와 병렬 웹 인덱스 검색엔진의 두 가지 구성요소를 가진 웹 로봇을 통하여 연구하였으며 웹페이지 추출과 조작은 HTML과 HTTP Objects를 처리할 수 있는 W3C(World Wide Web Consortium) Reference Library 개발도구를 활용하고, 웹페이지 분석을 목적으로 라이브러리(libink)를 개발하여 분석하였다.

UCB의 연구는 URL을 분류 및 추출하여 그 URL에서 최상위 도메인명과 상호 비교하여 종류별로 그 분석결과를 제시한 것이다<표 2.1>. 그러나 우리나라의 웹페이지의 언어별 구성비율과 국내 활성도메인의 구성비율 등의 분석은 이 기법으로 불가능하다.

표 2.1 도메인별 웹페이지의 비율

Domain	웹페이지 개수	점유율
com	516,709	20%
edu	698,616	27%
gov	117,125	4%
net	113,595	4%
mil	14,734	1%
org	89,939	3%
기타	1,064,318	41%
계	2,165,036	100%

Ⅲ. 웹 로봇 및 웹페이지 분석기

1. 웹 로봇의 설계

1.1 착안

일반적인 순차적 실행방식의 웹 로봇은 일시적인 호스트의 다운과 네트워크의 불안정 등으로 연결이 안되는 이른바 "Dead-links" URL의 존재로 수집성능의 한계성을 갖는다. 이 "Dead-links" URL이 있을 경우 웹 로봇은 Socket() 함수로 수집대상의 웹 서버 연결시, 180초의 응답지연이 발생한다. 그러한 이유로 "Dead-links" URL이 많을 경우 대기지연(latency)으로 인하여 웹 로봇의 성능은 점차적으로 떨어질 것이다. 따라서 본 논문에서는 "Dead-links" URL을 별도로 처리하는 멀티에이전트[5][6]를 제안하여 웹 로봇의 성능을 향상시키고자 한다.

1.2 멀티에이전트(Multi-Agent)

Amdahl's 법칙[7]에 따라 순차적 실행비율을 최소화하면 속도향상(Speedup)이 가능하다는 이론과 같이 제안하는 멀티에이전트 웹 로봇의 기본 기능들을 4 개의 독립 에이전트로 분리하여 웹페이지를 적재하는 구조를 갖는다. 이를 위하여 먼저 기존 로봇의 주요 기능에 대한 독립 에이전트로 (1) HTML 적재(Load)에이전트, (2) URL 추출(Extract)에이전트 (3) 활성 URL 검사에이전트, (4) 비활성 URL 검사에이전트를 도입한다. 순차적 실행방식의 적재

기를 HTML 적재에이전트와 활성 URL 검사에이전트, 비활성 URL 검사에이전트로 독립적인 기능을 부여하여 "Dead-Links" URL로 인한 대기지연이 발생하는 기존 방식의 단점을 제거하고 웹 로봇의 성능을 최대화하는 방식이다. 그림 3.1은 멀티에이전트로 구성된 웹 로봇과 국내 웹페이지를 분석하여 통계를 제시할 수 있는 웹페이지 분석기를 보여준다.

- HTML 적재에이전트 : URL 추출 에이전트가 새로 추출해낸 URL을 인터넷상에서 수집하여 지정된 곳에 저장함과 동시에 수집시의 성공과 실패 여부를 파일에 기록한다.
- URL 추출에이전트 : HTML 적재에이전트가 기 수집한 문서들을 순회 파싱하면서 새로운 URL을 추출한다.
- 활성 URL 검사에이전트 : 기 수집한 URL에서 활성화가 되어있는 URL들을 순회하면서 파일크기 등의 변경과 비 활성화 여부를 체크하여 파일에 저장한다.
- 비활성 URL 검사에이전트 : 기 수집한 URL에서 비 활성화가 되어있는 URL들을 순회하면서 그들의 재 활성화와 삭제여부를 체크하여 파일에 저장한다.

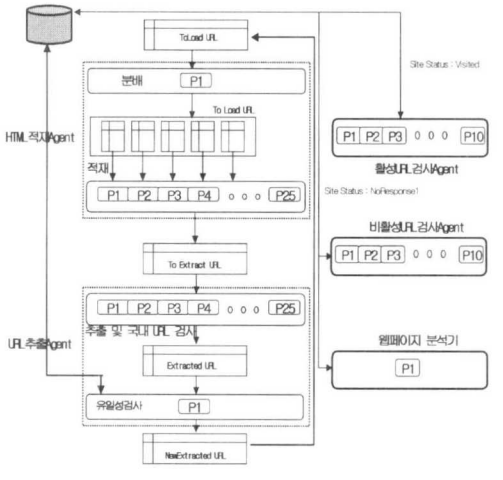

그림 3.1 웹 로봇 및 웹페이지 분석기

에이전트간의 의존성을 최소화하기 위하여 각 에이전트를 제어하고 각 에이전트의 출력과 공통적으로 사용되는 파일에 대해서 수정, 삭제, 추가에 대한 정보만을 각 에이전트로부터 입력받아 처리하는 중재모듈이 존재하며, 중재모듈의 초기 각 에이전트 실행 및 상호 Signal 처리에 관한 주요 소스는 다음과 같다.

```
int Mediator( )
{
Proc_Conf()
 /* 실행 파라미터(Robot.conf)설정에 따라 실행 */
First_Exec_Agent()
while (1){
if (STOP_Signal == 1) { /* 동작중지 : 서브모듈을 중지 */}
CheckChildProcessIsProcessing(Robot_Stat_File,ToLoad_URL_MN)
CheckChildProcessIsProcessing(Robot_Stat_File,HTML_Loader_MN)
CheckChildProcessIsProcessing(Robot_Stat_File,Active_URL_Scanner_M
N)
CheckChildProcessIsProcessing(Robot_Stat_File,Inactive_URL_Scanner_M
N)
}
```

1.3 URL의 현행화

그림 3.2에서 보는 바와 같이 URL은 URL 추출 에이전트에 의하여 Init상태에서 시작하여 HTML 적재에이전트에 의하여 Visited 혹은 NoResponse1 상태로 변화가능하며 활성 URL 검사에이전트에 의하여 Visited 상태에서 NoResponse1상태로 갈 수 있으며 비활성 URL 검사에이전트에 의하여 NoResponse1 상태에서는 Visited상태 혹은 NoResponse2로 변화되며 비활성 URL 검사에이전트에 의하여 NoResponse2 상태에서는 Visited상태 혹은 Dead로 가면서 문서의 생명은 끝난다.

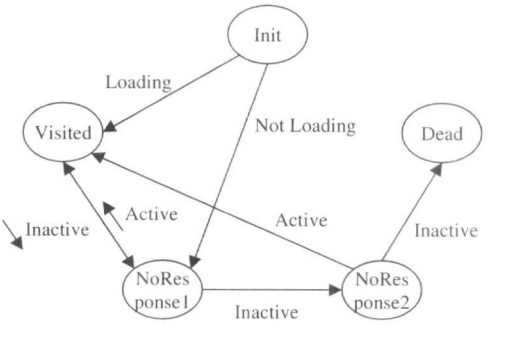

그림 3.2 웹사이트 상태변환

그림 3.3에 보는 바와 같이 활성 URL 검사에이전트는 기 수집한 문서에서 Visted Bit이 설정된 문서만 순회하면서 해당 URL의 헤더부분에서 파일의 생성날짜나 파일의 크기를 추출하여 기존의 값과 비교하여 변경여부를 체크한다. 만약 문서가 변경되

었으면 해당 URL의 Init Bit을 설정함으로서 URL 적재 모듈로 하여금 웹 상에서 다시 적재하도록 하여 최근의 문서를 보유하도록 한다.

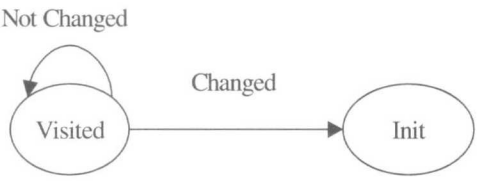

그림 3.3 Status 필드에 의한 상태 변환

2. 웹페이지 분석기

2.1 국내 홈페이지 분석

일반적으로 홈페이지는 www.snu.ac.kr과 같은 URL로 홈페이지 서비스가 되는 경우와 웹서버에 계정이 있는 이용자가 그 계정과 같은 이름의 디렉토리를 만들어 홈페이지를 만들어 서비스하는 경우로 구분된다. 홈페이지를 작성할 때에도 이 디렉토리에 특별한 서브디렉토리(예: html)를 두어 그 안에 필요한 .html 이나 .htm과 같은 파일을 작성할 수 있게 된다. 실제로 URL에서도 Tilda(~)가 붙은 첫번째 depth를 하나의 홈페이지로 본다. 즉 다음은 홈페이지의 하나로 정의된다.

예) http://snu.ac.kr/~wkim, http://202.30.64.22/~wkim

국내 홈페이지 개수를 산출하는 알고리즘은 아래와 같다.

```
Algorithm Homepage_Stat(HtmlSet,HomepageUrlFormSet)
input : HtmlSet  /* 국내의 전체 HTML page */
        HomePageUrlFormSet /* Homepage URL의 형식 */
output: HomepageNum /* 국내의 Homepage 개수 */
{
  for each html in HtmlSet
      if ( url(html) ∈ HomepageUrlFormSet ) then
        HomepagNum = HomepageNum ++;
  return HomepageNum;
}
```

2.2 국내 활성 도메인 분석

국내 인터넷 사용자가 등록하여 활용하고 있는 com, net, org 등과 같은 활성 국제도메인 개수과 활성 국가도메인(.kr) 통계를 분석할 수 있는 기법을 고안한다. 활성 국제도메인과 활성 국가도메인의 개수를 산출하는 알고리즘은 아래와 같다.

① 활성(Active) 국제도메인 개수 : 국내 IP주소 범위에 속하고, 웹서비스 또는 FTP, E-mail 서버를 구동하는 com, net, org, edu 등의 도메인 개수를 의미한다.

② 활성(Active) 국가도메인 개수 : 웹 서비스 또는 FTP, E-mail 서버를 구동하는 .kr의 국가도메인 개수를 의미한다.

```
Algorithm Find_Active_Domain(DomesticIPset, HtmlSet)
input : DomesticIPset   /* 국내 할당 IP set */
        HtmlSet         /* 국내 Html 페이지 */
output : ActiveDomesticDomainSet
        /* 활성 국가도메인 목록 */
        ActiveForeignDomainSet
        /* 활성 국제도메인 목록 */
{
  domainList ← Φ
  for each HTML page, html, in HtmlSet
  /* URL, Email, FTP 주소로부터 domain 추출 */
  begin
   domainListPerPage ← Parse(html);
     Add domainListPerPage to domainList;
  end
   ActiveDomesticDomainSet ← Φ;
   ActiveForeignDomainSet ← Φ;
  for each domain, checkDomain, in domainList;
    begin
     if ( domestic_domain(checkDomain) ) then
        Add checkDomain to ActiveDomesticDomainSet;
     else ip(checkDomain) ∈ DomesticIPset  then
        Add checkDomain to ActiveForeignDomainSet;
    else  /* non domestic-owned domain */
         Ignore checkDomain;
     end
  return ActiveDomesticDomainSet, ActiveForeighDomainSet;
}
```

Ⅳ. 성능분석

1. 시험환경

시험환경은 Linux 플랫폼 상에서 GNU C언어를 사용하여 1대의 DELL 4400 PC 서버을 사용하였다. 서버는 2 개의 PentiumIII 800MHz CPU 와 2 Gbytes 주메모리, 120 Gbytes Hard Disk Array를 정착하고 있다. OS는 Linux RedHat 6.2 버전을 사

용하였다.

2. 성능분석

웹 로봇의 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. 멀티에이전트의 성능분석 등을 위하여 웹 로봇 문서 수집대상으로 국내 대학교 웹사이트 10개를 임의로 선정하였으며, 전체 수집 대상의 URL은 약 15만개까지만 적재하고, 문서 수집시에 생성시킨 로그파일에 근거로 하여 분석결과를 산출하였다.

2.1 소켓 Timeout 설정값 변화에 따른 성능분석

웹 로봇의 적재 프로세스는 자료수집 과정에 웹 서버와 상호동작을 반복·수행한다. 이러한 상호동작은 크게 웹 로봇으로부터 웹서버로의 접속 요청(Connection request), 문서헤더 요청(Read head request), 문서본문 요청(Read body request)으로 구성된다. 이러한 요청은 기본적으로 소켓 통신을 기반으로 구현되며 주어진 웹 로봇의 적재 프로세스는 요청을 전송한 후 응답이 도달할 때까지 수행을 멈추고 대기 상태에 놓여 있게 된다. 만약 웹 서버 측에서 해당 호스트, 네트워크 등의 장애로 인하여 정상적인 응답이 가능하지 않은 경우 최대 180초까지의 응답지연이 발생하게 된다. 이러한 경우가 발생할 경우 웹 로봇의 전체 수집속도(Collection speed)가 매우 저하된다. 적재 프로세스는 Unix의 알람시그널(Alarm signal)을 기반으로 하여 요청에 대한 Timeout 값을 조절하여 위의 수집성능 저하문제를 해결해야 한다. Timeout 값이 매우 작을 경우에는 네트워크 또는 호스트의 과부하 또는 일시적인 서비스 지연으로 인하여 문서수집 실패를 초래하게 되어 전체문서 수집효율이 저하된다. Timeout 값이 매우 클 경우에는 호스트 및 네트워크 장애로 인한 삭제된 URL에 대한 문서수집 시에 대기 시간이 커지므로 문서 수집속도가 낮아지게 된다. 즉 수집효율과 수집속도 간에는 상충관계(Tradeoff)가 발생한다. 본 실험에서는 웹 로봇으로부터 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 4.1 에서 보는 바와 같이 Timeout 값이 3초 일 때 접속 요청의 경우 3,000개 정도의 문서에 대하여 응답실패가 발생하고 있음을 볼 수 있다.

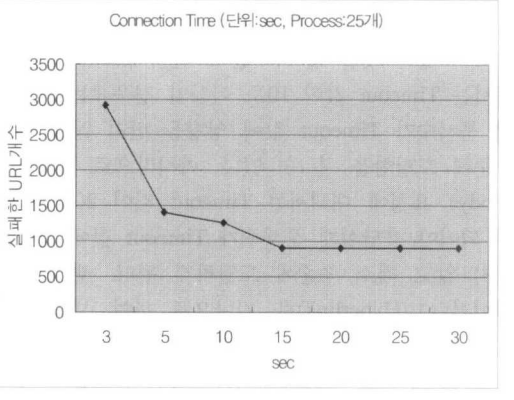

그림 4.1. 접속요청시 Timeout값과 응답실패문서수

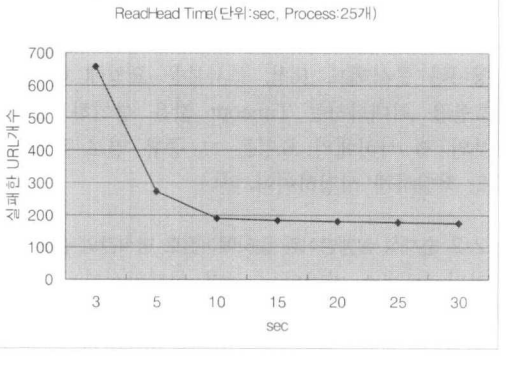

그림 4.2. Head요청시 Timeout값과 응답실패문서수

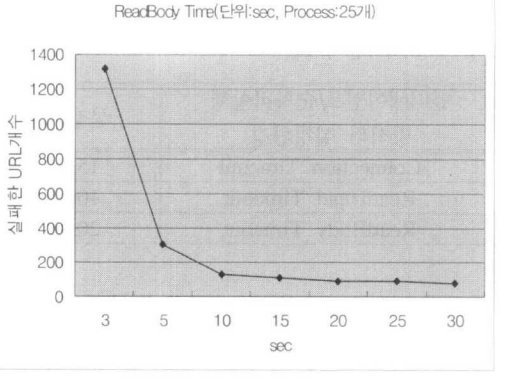

그림 4.3. Body요청시 Timeout값과 응답실패문서수

또한 그림 4.2와 그림 4.3에서 보면 문서헤드 요

청과 문서본문 요청을 비교할 때 Timeout 값이 매우 작은 경우 문서본문 요청의 경우가 문서헤드 요청의 경우보다 약 2배 이상의 응답실패율을 보이고 있음을 볼 수 있다. 수집효율을 최대로 보장할 수 있는 최소 Timeout 값을 살펴보면 먼저 접속(Connection) 요청에 있어서는 Timeout 값을 15초로 이상인 경우에는 응답실패 문서수가 더 이상 감소하지 않는 것을 볼 수 있다. 문서헤드 요청에 있어서는 Timeout 값이 10초 이상일 경우에는 응답실패 문서수가 Timeout 값에 영향을 받지 않는 최소 수준에 도달함을 볼 수 있다. 마지막으로 문서본문(Body) 요청에 대해서는 Timeout 값이 20초 이상이 되어야 응답실패 문서수가 Timeout 값에 영향을 받지 않고 최소 수준에 도달하게 된다. 웹 문서는 하이퍼링크(Hyperlink)를 기반으로 하여 상호 연결되어 있어 주어진 문서에 대하여 응답실패가 발생하면 그 문서에 링크되어 있는 하위 문서들의 수집도 대부분 가능하지 않게 된다.

일반적으로 웹 로봇의 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 로봇의 웹 서버에의 요청은 그 종류 별로 Timeout 값을 적절하게 설정하여야 한다.

2.2 순차적 실행방식과 멀티에이전트 방식간의 성능분석

멀티프로세스 기반의 순차적 실행방식과 본 논문에서 제시하는 멀티에이전트 방식의 웹 로봇의 성능을 비교 분석하였다. 앞에서 분석한 바와 같이 웹 서버로의 문서 요청시 최적화된 Timeout 값을 두가지 방식에 동일하게 표 4.1과 같이 설정하였다.

표 4.1. 최적화된 설정값

멀티에이전트/순차적실행 방식의 설정환경	설정값
Connection Timeout	15sec
ReadHead Timeout	10sec
ReadBody Timeout	20sec

웹페이지의 수집 성능을 나타내기 위하여 시간당 수집문서 개수를 성능평가항목(Performance metric)으로 정의하였다. 약 15만개의 URL을 추출한 분석결과<표 4.2>를 고찰하면 멀티에이전트 방식이 멀티프로세스 기반의 순차적 실행방식보다 평균 약 2.35배의 성능향상을 보이고 있다.

그림 4.4 및 표 4.2와 같이 URL의 수집량의 증가에 따라 본 논문에서 제안한 멀티에이전트 방식의 웹 로봇이 성능향상의 비율을 증가시키는 것은 HTML 적재에이전트에서 수집한 "Active URL"과 "Dead-links URL"에 대해서 각 10개의 프로세스로 구동되는 "활성 및 비활성 URL 검사에이전트"가 별도로 동작하기 때문이다. 즉 멀티프로세스 기반의 순차적 실행방식에서는 적재기(Loader)가 "Active URL"과 "Dead-links URL"의 상태변화에 대해서 반복 검사 및 적재·추출을 수행해야 하기 때문에 수집속도가 떨어지는 것이다. 국내 인터넷의 웹페이지 규모가 약 2,000만개라는 점을 감안하면 변경 및 삭제 또는 일시 중단된 사이트의 증가로 인하여 순차적 실행방식은 그 성능이 저하되나, 멀티에이전트 방식의 활성 URL 검사에이전트 및 비활성 URL 검사에이전트로 인하여 성능저하를 방지하게 된다.

표 4.2. 성능분석 결과

실행방식	1H	2H	3H	4H	5H	6H
멀티에이전트	19,681	60,471	110,268	135,310	150,725	151,797
순차적실행	7,240	23,883	36,223	57,452	74,548	96,210
성능향상	2.7배	2.5배	3.0배	2.3배	2.0배	1.6배

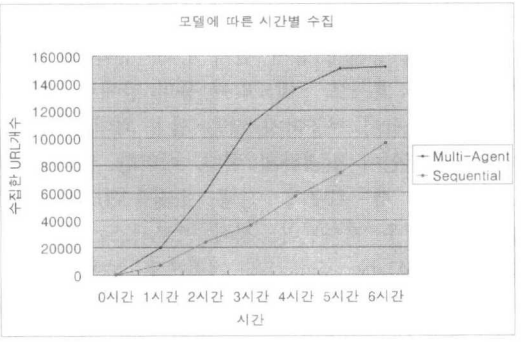

그림 4.4. 성능분석 비교

V. 결 론

실험적으로 멀티에이전트 방식의 웹 로봇이 순차적 실행방식보다 수집속도에서 100% 이상의 성능향상을 보이고 있다는 것을 증명하였다. 나아가 수집대상의 규모가 크면 클수록 더욱 큰 성능향상을 얻을 수 있을 것이다.

그러나 Timeout 값의 선정에 있어서 웹 로봇 수행중에 응답실패율을 관찰하여 상황에 적합하도록

최적의 값으로 동적 조절이 가능한 주어진 시스템 환경하에서의 (1)적응형(Adaptive) Timeout 설정기법에 대한 연구가 요구된다. 또한 HTML 적재에이전트에서 웹 서버를 고려한 (2)Dynamic URL 분배를 할 수 있는 알고리즘 연구가 필요하며, 마지막으로 웹 로봇의 수행시간은 네트워크시간과 컴퓨팅시간으로 분류될 경우 어떤 수행시간이 웹페이지 수집시 많이 점유하는지를 분석하여 해당되는 기능에 대한 성능향상을 연구할 필요가 있다.

참 고 문 헌

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

[2] 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.

[3] Tim Berners-Lee, "Uniform Resource Locators", RFC 1738, IET F, Dec. 1994.

[4] Allison Woodruff, Paul M.akoi, Eric Brewer and Paul Gauthier, "An Investigation of Documents from the World Wide Web", http://www.cs.berkeley.edu /~woodruff/inkotmi

[5] Nicholas Jennings, Katia Sycara and Michael Wooldridge, "Road map of Agent Research and Development : Autonomous Agents and Multi-Agent Systems", vol. 1 pp. 7~38, 1998.

[6] Riechen, Doug, "Intelligent Agents", Communications of the ACM Vol. 37 No. 7, July 1994.

[7] Kai Hwang, "Advanced Computer Architecture", McGraw-Hill, pp111-129, 1993

부록 웹페이지 분석결과

2001. 12월 수집된 웹페이지(18,947,925개)의 분석

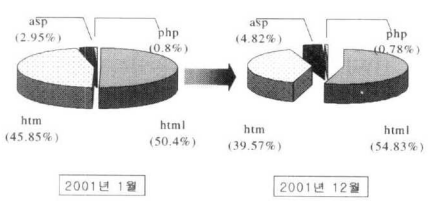

그림 A. 국내 웹페이지 언어별 구성비율

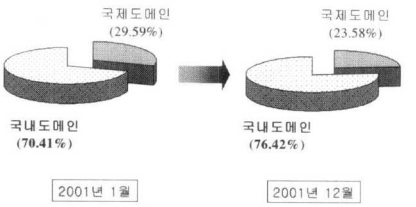

그림 B. 국내 활성도메인의 구성비율

김 원(Weon Kim) 정회원

한국통신학회 논문지, 제26권, 제6A호 참조

김 희 철(Hiecheol Kim) 정회원

1983년 연세대학교 전자공학과 졸업(공학사)
1991년 Univ. of Southern Californi (Computer Eng. M.S.)
1996년 Univ. of Southern Californi (Computer Eng. Ph.D.)
1983년 - 1988년 (주)삼성전자 주임연구원
1996년 - 1997년 (주)삼성SDS 수석연구원
1997년 - 현재 대구대학교 정보통신학부 조교수
<주관심 분야> 병렬처리, 컴퓨터구조, 컴파일러

진 용 옥(Yong Ohk Chin) 종신회원

한국통신학회 논문지, 제26권, 제6A호 참조