https://doi.org/10.33778/kcsa.2019.19.3.021

분산형 병렬 크롤러 설계 및 구현

장 현 호*, 전 경 식*, 이 후 기**

요 약

기관이나 단체에서 관리하는 홈페이지 수가 증가하면서 그에 따른 웹 어플리케이션 서버나 컨테이너도 그에 상응하게 증가한다. 웹 어플리케이션 서버와 컨테이너의 웹 서비스 상태를 점검하는데 있어서 사람이 일일이 원격지에 있는 물리적인 서버에 터미널을 통해 접근하거나 다른 접근 가능한 소프트웨어를 사용하여 접근한 뒤 웹 서비스의 상태를 점검하는 것은 매일 반복하기에는 매우 번거로운 일이다. 이전에 연구되었던 크롤러관련 연구에는 크롤링에 따른 발생 데이터 처리에 관한 언급이 찾아보기 힘들다. 크롤러에서 데이터베이스에 접근하여 데이터를 저장하는데 있어서 데이터 손실이 발생한다. 본 연구에서는 크롤링 기반의 웹 어플리케이션 서버 관리에 따른 점검 데이터를 손실 없이 데이터화 하여 저장하는 방안을 제시하였다.

Distribute Parallel Crawler Design and Implementation

Hyun Ho Jang*, kyung-sik, jeon*, HooKi, Lee**

ABSTRACT

As the number of websites managed by organizations or organizations increases, so does the number of web application servers and containers. In checking the status of the web service of the web application server and the container, it is very difficult for the person to check the status of the web service after accessing the physical server at the remote site through the terminal or using other accessible software It. Previous research on crawler-related research is hard to find any reference to the processing of data from crawling. Data loss occurs when the crawler accesses the database and stores the data. In this paper, we propose a method to store the inspection data according to crawl - based web application server management without losing data.

Key words : crawler, web Crawler, server

접수일(2019년 2월 28일), 수정일(1차: 2019년 3월 19일), 게재확정일(2019년 3월 28일)

* 숭실대학교/IT정책경영학
** 건양대학교/사이버보안공학과(corresponding author)

1. 서 론

인터넷이 발달하면서 각 기업이나 단체에서는 관리해야 할 웹 어플리케이션 및 홈페이지 수가 증가함에 따라 사람이 일일이 상태를 확인하고 점검하는 관리가 어려워지고 있다. 이를 효과적으로 관리하기 위해서는 Hypertext transfer protocol (HTTP)의 상태 값을 확인하여 정상, 지연, 단절, 오류(웹 페이지 프로그램 오류)를 점검하는 것을 사람이 일일이 확인하기 보다는 자동화된 프로그램으로 효과적인 웹 어플리케이션 서버의 상태를 확인하고 관리될 수 있도록 한다.

HTTP 서버는 요구하는 메시지를 수신하여 처리한 뒤 그 결과를 알려주는 세 자리 정수로 처리번호가 있다. 이를 상태코드라 하며 Reason-Phrase에는 Status-Code에 대한 간단한 설명문이 포함될 수 있다. 표준으로 정의된 상태코드 이외에 확장 가능하다.[1][2][5]

웹상의 다양한 정보를 수집하는 소프트웨어적 도구로서 스파이(spider), 검색엔진(Search engine), 봇(bot), 지능형 에이전트, 크롤러(crawler)이라고 하는 것이 있다. 이 소프트웨어 도구로 HTTP protocol을 이용하여 HTTP 서버의 상태를 점검하고 관리를 용이하게 해줄 수 있다.[3]

여러 HTTP Server의 상태정보를 수집하기 위해서는 스레드(thread)나 프로세스(process)기반의 병렬 크롤러(crawler)를 이용하여 제한된 시간 내에 수많은 정보를 수집하고 저장하고 있다. 하지만 수집된 정보의 처리속도가 빠르기 때문에 데이터베이스(DBMS)의 저장 처리 속도가 다중 스레드간의 처리 속도보다 느려 데이터 손실이 발생하는 경우가 있다.

참고문헌에서는 여러 대의 머신 또는 병렬 웹 크롤러를 제안하고 있으나 수집된 데이터에 대한 처리가 파일 또는 데이터베이스(DBMS)를 이용하고 있는지도 명확하지 않다.

본 논문에서는 멀티 HTTP Server의 병렬 크롤링 기법으로 상태코드를 수집하여 저장관리 하는 아키텍처와 세부 구현 방법을 제시한다. 멀티 HTTP Server에 대한 상태정보 수집은 멀티 크롤러(Multi Crawlers)방식을 사용하며, 병렬 크롤링은 스레드 기반의 멀티 크롤러(Multi Crawler)를 분산 배치를 통해 대중의 HTTP Server의 상태정보를 수집하여 일명 데이터 서버로 중앙 데이터 수집에 의한 데이터 저장을 제안한다. 데이터 서버는 TCP/IP기반으로 스레드 기반의 크롤러가 접속하여 집중화된 데이터 서버에 수집한 상태정보를 보내고 저장하는 방식이다.

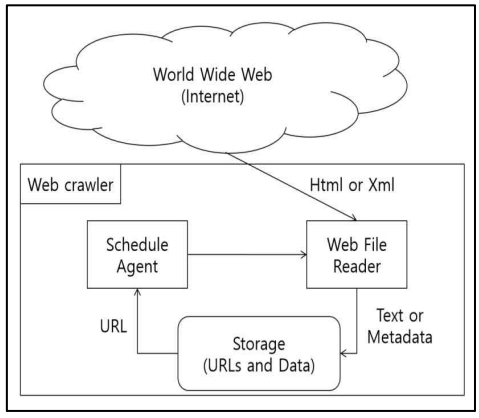
본 논문의 이점은 첫째, 다수의 HTTP Server의 상태정보를 스레드 기반의 병렬처리 웹 크롤러 아키텍처를 제안한다. 둘째, 데이터베이스의(DBMS)의 저장 처리 속도를 보장하기 위한 데이터 서버 아키텍처를 제안한다.

2. 관련 연구

2.1 웹 크롤러

2.1.1 개요

웹 크롤러는 다수의 HTTP Server의 웹 페이지와 상태 정보를 수집하여 저장, 갱신하기 위한 소프트웨어 도구로 사용된다. 웹 크롤러의 목표 아키텍처는 제한된 시간 안에 최대한 많은 HTTP Server의 웹 페이지나 상태 정보를 수집하는데 있다. 프로세스를 활용하여 크롤러를 구현할 경우 컴퓨터의 자원을 낭비하는 결과를 초래할 수 있으므로 스레드를 활용한 크롤러가 여러 사이트의 웹 페이지나 상태정보를 수집하는데 많은 이점이 있다. 아래의 그림은 일반적으로 사용되는 크롤러의 아키텍처를 보여주고 있다.[4][7]



(그림 1) 웹크롤러 시스템

2.1.2 기능

인터넷상에 산재해 있는 웹 페이지를 수집하며, 그 페이지 안에 있는 링크를 추출하고 하위 링크의 웹 페이지 및 이미지, 파일, 멀티미디어등과 같은 정보를 수집한다. 수집된 정보들을 어떤 형태로 저장할 것인지에 따라서 데이터베이스(관계형, 계층형, 메모리형, No-SQL등)나 파일형태로 저장할 수 있다. 이외에 키워드 색인기, 색인어 추출, 파일 색인기, 조회수 랭킹 등에 관한 정보들을 추출하여 데이터베이스화 한다면 검색엔진으로 확장할 수 있다.[6]

2.1.3 동작방식

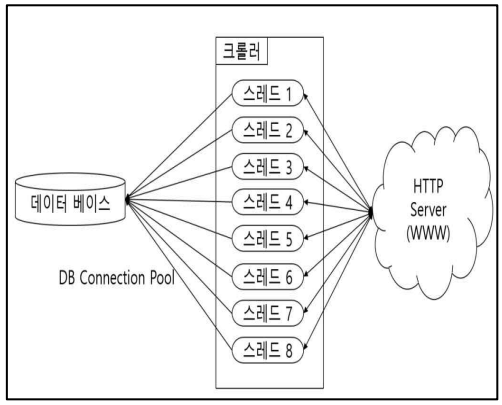
크롤러는 웹 서버를 순회하거나 링크를 따라 가면서 각각의 홈페이지 내에 있는 문자, 숫자, 이미지, 멀티미디어 정보 등 많은 데이터를 수집하는 프로그램으로서 사람들 대신하여 홈페이지의 내용 및 링크를 얻어 주기적(스케줄 에이전트)으로 정보를 수집한다. 웹 크롤러의 다음과 같이 2가지 특징을 가지고 있다.[8][9]

※ 멀티 프로세서(Multi-Processor)

데이터 수집 속도를 향상시키기 위해 다수의 프로세스나 스레드를 생성하여 동작시킨다. 산술적으로 다수의 프로세스나 스레드를 생성하면 단일 프로그램보다 대량의 데이터를 수집하는 고성능의 크롤러를 기대하겠지만 네트워크 상황(트래픽 또는 웹페이지 콘텐츠 양), 시스템 메모리 및 CPU 성능 등을 고려하면 성능향상은 단정하기 힘들다.[9]

※ 작업 스케줄링 (Job Scheduling)

크롤러가 방문해야 할 웹 URL이 다수이고 주기적으로 정보를 데이터를 수집하고 갱신처리하기 위해서는 방문 전략에 따른 계획된 스케줄에 의해 동작되어야 한다. 또한 새롭게 수집된 웹 페이지의 sub link를 추출하게 되면 그만큼 더 시간이 소요되기 때문에 각 프로세서나 스레드에 URL정보를 분산 배치해서 제한된 시간 안에 동작될 수 있도록 한다.[9]



(그림 2) 스레드 기반 웹크롤러 시스템

2.2 TCP/IP를 프로그램(소켓 프로그램)

많은 정보통신 기기는 TCP/IP를 기반으로 하여 기간 통신으로 데이터를 주고받을 수 있다. 이를 활용하려면 “통신 규약” 즉 “Protocol(프로토콜)”을 사용하여 한다. 통신 규약은 정보 기기 간의 약속된 데이터 표현식으로서 송수신할 데이터의 형식과도 같다. [1]

여러 프로그래밍 언어들이 Socket 프로그램으로 서버 및 클라이언트 구현에 대한 메서드나 Function을 제공하고 있다.

3. 분산 및 웹 크롤러 설계

본 논문에서는 스레드를 기반으로 하는 병렬처리 웹 크롤러와 데이터 서버를 활용한 분산형 웹 크롤러에 대한 연구 진행하였고, 크롤러를 통해서 수집한 데이터의 손실을 최소화 하는 방식을 소개한다.

3.1 분산 및 병렬 웹 크롤러의 기능별 설계

본 논문에서는 크롤러의 분산처리와 단일 크롤러의 병렬처리에 대한 설계를 골자로 설계하기로 했다. 분산형 크롤러를 구축하기 위해서는 환경설정 내역을 크롤러가 구동 시 입력받아 TCP/IP로 데이터베이스 및 데이터 서버의 정보, 스레드 개수에 대한 정보를 획득하여 구동한다.

3.1.1 구동환경 설계

<표 1> 구동 환경설정

환경설정	설명
데이터베이스 정보	크롤러가 사용할 수 있는 데이터베이스의 IP, Driver, ID, Password, 기타 데이터베이스에 관련된 초기 설정값을 읽어드린다.
크롤러 정보	큐나 스레드 개수 및 데이터 서버 IP, Port, 크롤링 재시도 횟수 등이다.

<표 1>과 같이 크롤러 구동시 데이터베이스 정보 및 크롤러의 환경설정 내역을 파일에 기록하여 구동시킨다. OS별로 프로그램 언어가 환경설정 경로를 읽어드리는 방식이 다르다. 이로 인해서 크롤러 구동 시 OS를 판별하여 환경파일 경로에 있는 설정파일을 읽어드린다.

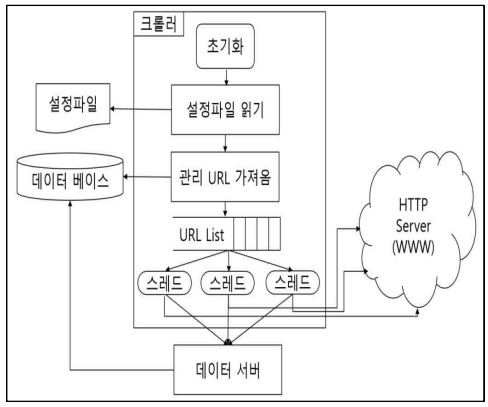
<표 2> 구동 환경설정 파일

설정구분	설정내역
크롤러 설정	crawler.retry.count=3 crawler.queue.maxsize=100 crawler.httpconn.timeout=100 crawler.socket.timeout=3 crawler.queue.names=HTTP_QUEUE crawler.num=1 crawler.logfile.resource=c:/crawler.log. dataserver.ip = xxx.xxx.xxx.xxx dataserver.port=xxxx
데이터베이스 설정	DB_LIST=maria maria.driver_class_name=org.mariadb.jdbc.Driver maria.max_active=10 maria.username=wsis maria.password=wsis1q maria.idleTimeout=1000000 maria.checkoutTimeout=1000000 maria.maxCheckout=1000000 maria.cacheStatements=true maria.url=jdbc:mariadb://xxx.xxx.xxx.xxx:xxxx/db

<표 2>와 같이 크롤러 설정부분과 데이터베이스 접속관련 설정정보를 구분하여 두 개의 파일로 저장한다. 크롤링 retry count는 Http Sever접속 시 단절이나 지연에 관련된 사이트의 재확인 차원에서 설정한다. socket timeout은 Http Server의 접소 지연시간을 설정하여 설정된 시간이 넘어서면 지연되는 사이트로 확인하기 위함이다. crawler num(Number의 약자)는 분산형 크롤러를 구현하기 위함인데, 크롤러 숫자로 구분하기 위해서이다.

3.1.2 병렬 크롤러의 설계

본 논문은 스레드 기반의 병렬 크롤러를 설계하고 크롤러에서 수집되는 정보를 본 논문에서 정의한 데이터 서버로 전송하고 저장하는 것을 설계한다.



(그림 3) 스레드 기반 병렬 웹크롤러 시스템

(그림 3)은 3.1.1 구동환경 설정에서 언급하였듯이 크롤러가 구동하면서 URL을 담는 큐와 데이터베이스 설정정보, 스레드 수를 읽고 구동하면서 수집되는 데이터를 데이터 서버를 통해 저장하는 구조를 도식화 하였다. 크롤러에서 가져오는 URL은 다음과 같이 데이터베이스에 저장된 정보를 가져와 구동된다.

<표 3>은 본 논문에서는 특정 URL이 등록된 내역이 중요한 것이 아니라서 간단한 데이터 정보

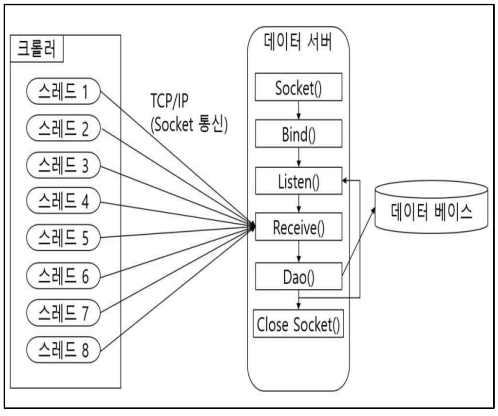
<표 3> URL 등록정보 예시

crawler num	url
1	www.domain name 1
1	www.domain name 2
1	www.domain name 3
2	www.domain name 4
2	www.domain name 5
2	www.domain name 6

를 예시로 정리한다. 크롤러 번호를 기준으로 등록된 URL 도메인 정보를 데이터베이스에서 크롤러가 읽어 들일 수 있도록 하였다.

3.1.3 데이터 서버 설계

본 논문은 소켓 프로그램을 기반으로 하는 데이터 서버를 설계하였다.



(그림 4) 분산 및 병렬 크롤러와 데이터 서버

(그림 4)와 같이 분산 및 병렬 크롤러가 소켓 프로그램 기반으로 데이터 서버에 저장될 데이터를 보내고, 데이터 서버는 보내온 데이터를 받아 저장하는 구조로 한다. 각 스레드는 보낼 데이터가 있을 때 소켓 접속 정보로 접속하여 데이터를 보내며 전송이 완료되면 소켓을 닫는다.

데이터 서버는 받은 데이터를 약속된 프로토콜 기반으로 수신되는 데이터를 확인하고 정상적인 데이터를 받았을 경우에만 DAO(Data Access Object)를 통하여 데이터베이스에 접근하여 저장한다.

프로토콜 구조는 아래와 같이 단순 구조이다.

패킷헤더	패킷 데이터

패킷헤더에는 전체 메시지 길이에 대한 내용이 포함되어 있어 잘못된 데이터를 판별할 수 있도록 하였다. 패킷 데이터 부분에서는 크롤러 번호, 도메인 코드, 도메인 HTTP Server 상태코드, 데이터 수집 타임스탬프 등 수집 대상 항목들로 정의하였다. 통신데이터 메시지 기반의 통신방식을 채택하였다.

패킷 메시지 구조는 “패킷길이||크롤러 번호||도메인 코드||도메인 HTTP Server 상태코드||데이터 수집 타임스탬프||....” 이와 같이 정의하였다. 데이터를 구분하여 파싱할 수 있도록 “||”를 구분자로 한다.

4. 분산 및 병렬 웹 크롤러 구현

본 장에서는 3장에서 제시한 분산 및 병렬 웹 크롤러의 자세한 구현방법을 살펴보기로 한다.

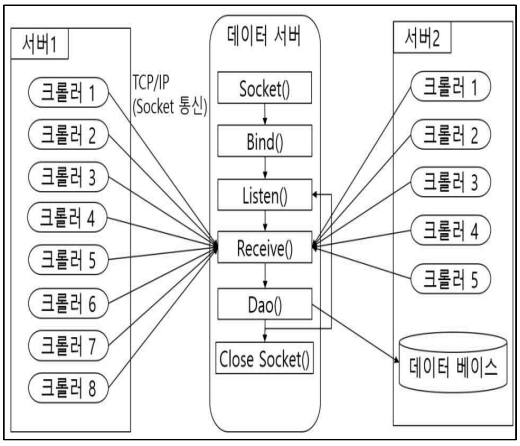
4.1 분산 크롤러

분산 웹크롤러는 물리적인 하드웨어 구성이나 단일 장비에서의 소프트웨어적 분산처리를 구현한다.

<표 4> 크롤러를 구분하는 값 정리

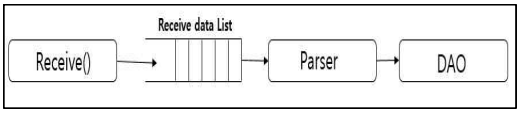
설정 이름	데이터 타입	키	값
num	String	크롤러 번호	1(...4.5)
retry.count	int	접속 재시도 횟수	3
queue.max	int	큐 생성 개수	100
dataserver.ip	String	데이터 서버 IP	xxx.xxx.xxx
dataserver.port	int	데이터 서버 포트번호	xxxx

<표 4>는 분산형 크롤러를 구현하기 위하여 크롤러 번호, 수집대상 URL, 재접속 시도 횟수, URL 저장 큐 생성 수, 데이터 서버 IP 및 포트번로를 설정하는 내역을 정리하였다. 데이터 서버 및 포트 값 중에 x로 표시한 부분은 숫자이다.



(그림 5) 분산 크롤러와 데이터 서버 아키텍처

(그림 5)는 분산형 크롤러를 구현한 내역을 도식화한 아키텍처이다. 물리적인 서버를 다중으로 구성할 수 있고, 각 물리적인 서버에 크롤러 및 데이터서버, 데이터베이스를 설치하여 분산 구축한다.



(그림 6) 수신 데이터 처리

(그림 6)은 데이터 서버가 소켓통신으로 받은 데이터를 큐(Receive data List)에 저장하고 순차적으로 수집대상 데이터에 대한 protocol 정의에 따라 파서를 통해 데이터를 세분화하여 데이터베이스 테이블에 저장한다. 이때 수집된 데이터의 생성 타임스탬프는 크롤러의 각 스레드에서 데이터가 생성될시 발생한 연월일시분초로 하여 발생시킨다.

4.2 병렬 크롤러

병렬 크롤러는 소프트웨어로 구현되는 큐와 스레드로 다수의 URL을 병렬로 방문하여 해당 HTTP서버의 상태 값과 이벤트 내역을 수집한다.

<표 5> 병렬 크롤러를 구분하는 값 정리

설정 이름	데이터 타입	설명	값
url_code	int	URL 코드	1(...4.5)
url	String	접속 재시도 횟수	3
depth	int	URL 깊이	1
sub_cnt	int	하위 URL 수	10
url_ip	String	해당 URL IP	xxxx

<표 5>는 스레드에 할당되는 URL에 대한 기본 데이터이다. url_code는 데이터베이스에 저장하여 URL을 코드로 구분하는 key이다. url은 수집하고자 하는 도메인의 정보이다. depth는 1차 또는 그 하위 도메인의 깊이를 나타낸다. sub_cnt는 등록된 URL의 하위 도메인을 분석하여 나온 개수를 나타낸다. url_ip는 등록된 URL에 대한 데이터 수집 시 생성된 해당 HTTP 서버의 IP를 관리한다.

5. 분산 및 병렬 웹 크롤러 실험 평가

본 장에서는 분산 및 병렬 웹 크롤러의 실험을 측정하고 결과를 설명한다.

5.1 분산 및 병렬 크롤러 실험환경

본 논문에서는 크롤링에 사용될 URL은 공공기관 및 교육기관 즉 대학이나 대학원의 대표 홈페이지 URL을 사용한다. 실험은 총 4대의 사설 클라우드를 사용하였다.

<표 6>과 같이 동일한 하드웨어 스펙으로 할당된다. 클라우드 기반에서 실험을 실시하였다. 운영체제는 동일한 버전의 리눅스(CentOS 6.5)

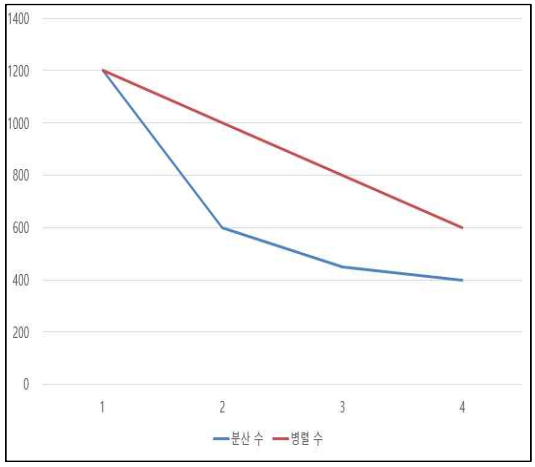
를 사용하였으며, 1대에는 데이터베이스(MariaDB)와 데이터 서버를 구축하였다.

<표 6> 분산 및 병렬 크롤러를 위한 하드웨어

CPU	메모리	HDD	NIC
2.1GHz	16G	500GB	100
2.1GHz	16G	500GB	100
2.1GHz	16G	500GB	100
2.1GHz	16G	500GB	100

5.2 분산 및 병렬 크롤러 실험결과

아래의 (그림 7)과 같이 분산된 크롤러 즉 머신을 늘려가면서 HTTP서버의 상태정보와 데이터를 수집한 결과이다.



(그림 7) 분산 및 병렬 크롤러

한 대의 머신에 1,000개의 URL을 등록하고 처리할 경우 1,200ms의 시간이 소요되며, 머신의 수를 늘려갈 경우 처리 시간이 감소하는 것으로 처리 속도가 개선되는 것을 확인할 수 있다.

위와 같이 동일한 URL 개수로 동일 머신에 병렬로 구성할 경우 크롤링 소요 시간이 다소 높다. 또한 단일 머신에서 크롤러를 병렬로 구성할 경우 CPU 및 메모리 점유율이 높다. 이로 인해 단일 머신에 구성되어 있는 데이터베이스 성능에서 부하를 초래한다.

크롤러를 분산 및 병렬로 구축하여 구동할 경우 URL별 12ms ~ 100ms의 HTTP Server 상태정보 및 데이터를 수집한다. 수집된 데이터를 데이터 서버를 연동하지 않고 데이터베이스에 insert할 경우 하지 동시접근이 발생하여 데이터 누락이 발생한다.

(그림 8) 단순 테스트용 테이블

(그림 8)과 같이 테스트 목적으로 단순 테이블을 생성하여 HeidiSQL을 이용하여 데이터를 입력할 경우 처리 속도는 235 ~ 375ms로 크롤러가 데이터를 수집하는 속도보다 느린 경우가 발생할 경우 PK중복에 의한 데이터 손실이 발생한다.

본 논문에서 제안한 데이터 서버 소켓통신으로 데이터를 집중화하여 연동할 경우 데이터 누락이 발생하지 않았다.

6. 결 론

본 논문에서는 여러 크롤러를 하드웨어적으로 분산하고, 소프트웨어적으로도 병렬처리를 하는데 있어서 단일 하드웨어 또는 단일 크롤러의 성능개선과 수집되는 데이터의 손실을 개선하는데 사용할 모델을 제시하고 그에 따른 설계와 구현에 대한 이해를 돕기 위해 작성하였다.

특정 URL을 기준으로 실험 및 결과를 도출하는데 있어서 해당 도메인에 내포된 서브 URL수에 따라서

성능측정이 상이할 수 있고, 네트워크 트래픽 속도나 환경에 따라서 실험결과는 달라 질수 있으나 여러 번 반복하여 상이한 값이 도출되지 않도록 하였다.

본 논문은 분산 및 병렬 크롤러가 수집하는 데이터를 안정적으로 데이터베이스화 하는데 그 목적에 맞게 설계되고 구현하는 방법을 제시하는데 그 의미가 있다고 할 수 있다.

참고문헌

[1] http://www.tta.or.kr

[2] Berners-Lee, Tim. "HyperText Transfer Protocol". World Wide Web Consortium. Retrieved 31 August 2010.

[3] Curbera Francisco et al., "Unraveling the Web Services Web:An Introduction to SOAP, SDL, and UDDI," IEEE Internet computing, Vol.6 No.2, pp.86-93, 2002.

[4] Castillo, C., "Effective Web Crawling," ACM SIGIR Forum 55, Vol.39, No.1, pp. 55-56, June 2005.

[5] "HTTP/1.1". Webcom.com Glossary entry. Archived from the original on 2001-11-21. Retrieved 2009-05-29.

[6] Heydon, A. and Najork, M., "Mercator: A Scalable, Extensible Web Crawler," In Proc. 2nd Int'l Conf. on World Wide Web, pp.219-229, Dec. 1999.

[7] Tim Berners-Lee. "The Original HTTP as defined in 1991". World Wide Web Consortium. Retrieved 24 July 2010.

[8] V. Shkapenyukn, T. Suel, "Design and Implementation of a High-performance Distributed Web Crawler," In Processings of the 18th International Conference on Data Engineering, San Jose, California, 2002.

[9] 신은정, 김이론, 허준석, 황규영, “오디세우스 용량 검색 엔진을 한 병렬 웹 크롤러의 구현” 정보과학회 논문지 : 컴퓨팅의 실제 및 레터, Vol. 14, No 6, 2008년 8월

〔저자소개〕

장 현 호 (Hyunl-Ho Jant)
2015년 11월 국가평생 교육진흥원 컴퓨터공학 학사
현재 : 숭실대학교 일반대학원 IT정책경영학 박사과정
중부대학교 소프트웨어공학부 겸임교수
관심분야 : AI, ICT,IoT, Information Security, Data Science등
email : jangh2@daum.net

전 경 식 (Kyung-Sik Jeon)
2013년 2월 서울사이버대학교 경영학 전공 학사
2016년 2월 숭실대학교 일반대학원 IT정책경영학과 공학석사
현재 : 숭실대학교 일반대학원 IT정책경영학과 박사과정
관심분야 : 빅데이터, 포렌식, 데이터 분석등
email : jeep3162@gmail.com

이 후 기 (HooKi Lee)
2018년 2월 숭실대학교 일반대학원 IT정책학 공학박사
현재 : 건양대학교 프라임창의융합대학 사이버보안공학과 교수
관심분야 : Cyber Security, Digital Forensic, Data Science 등
email : hk0038@konyang.ac.kr