오디세우스 대용량 검색 엔진을 위한 병렬 웹 크롤러의 구현

(Implementation of a Parallel Web Crawler for the Odysseus Large-Scale Search Engine)

신 은 정 † 김 이 른 † 허 준 석 † 황 규 영 ††
(Eun-Jeong Shin) (Yi-Reun Kim) (Jun-Seok Heo) (Kyu-Young Whang)

요 약 웹의 크기가 폭발적으로 증가함에 따라 인터넷에서 정보를 얻는 수단으로서 검색 엔진의 중요성이 부각되고 있다. 검색 엔진은 사용자에게 최신의 정보를 검색 결과로서 제공하기 위해 웹 페이지를 주기적으로 수집하고 이를 데이타베이스에 저장한다. 웹 크롤러는 이러한 목적으로 웹 페이지를 수집하는 프로그램이다. 대부분의 검색 엔진은 제한된 시간 내에 많은 수의 웹 페이지를 수집하기 위해 다수의 머신을 사용하는 병렬 웹 크롤러를 이용한다. 그러나, 병렬 웹 크롤러의 아키텍처와 세부 구현 방법이 잘 알려져 있지 않기 때문에 실제로 병렬 웹 크롤러를 구현하는 데에 어려움이 많다.

본 논문에서는 **병렬 웹 크롤러**(*parallel web crawler*)의 아키텍처와 세부 구현 방법을 제시한다. 병렬 웹 크롤러는 다수의 머신에서 웹 페이지를 병렬적으로 수집하기 위해 조정자(coordinator) 대리자(agent) 구조의 2-티어(tier) 모델을 사용한다. 조정자/대리자 모델은 각 머신에서 웹 페이지를 수집하기 위한 다수의 대리자들과 이 대리자들을 관리하기 위한 하나의 조정자로 구성된다. 병렬 웹 크롤러는 웹 페이지를 수집하기 위한 *크롤링*(crawling) 모듈, 수집한 웹 페이지를 데이타베이스 로딩 포맷으로 변환하기 위한 *컨버팅*(converting) 모듈, 수집된 웹 페이지의 중요도를 계산하기 위한 *랭킹*(ranking) 모듈로 구성된다. 본 논문에서는 병렬 웹 크롤러의 각 모듈들을 설명하고, 세부 구현 방법을 설명한다. 마지막으로, 실험을 통해 병렬 웹 크롤러의 성능을 평가하였다. 실험 결과, 제안된 병렬 웹 크롤러가 수집해야할 웹 페이지 개수와 머신 개수에 따라 확장 가능함을 보였다.

키워드 : 병렬 웹 크롤러, 대용량 검색 엔진

Abstract As the size of the web is growing explosively, search engines are becoming increasingly important as the primary means to retrieve information from the Internet. A search engine periodically downloads web pages and stores them in the database to provide readers with up-to-date search results. The web crawler is a program that downloads and stores web pages for this purpose. A large-scale search engines uses a parallel web crawler to retrieve the collection of web pages maximizing the download rate. However, the service architecture or experimental analysis of parallel web crawlers has not been fully discussed in the literature.

In this paper, we propose an architecture of the **parallel web crawler** and discuss implementation issues in detail. The proposed parallel web crawler is based on the *coordinator/agent* model using multiple machines to download web pages in parallel. The coordinator/agent model consists of multiple agent machines to collect web pages and a single coordinator machine to manage them. The parallel

· 본 논문은 과학기술부/한국과학재단의 국가지정연구실사업(No. R0A-2007-000-20101-0)에 의해 수행된 연구임

† 학생회원 : 한국과학기술원 전산학과
joyceshin@mozart.kaist.ac.kr
early@mozart.kaist.ac.kr
jsheo@mozart.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
kywhang@mozart.kaist.ac.kr

논문접수 : 2007년 12월 4일
심사완료 : 2008년 5월 1일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제6호(2008.8)

web crawler consists of three components: a *crawling* module for collecting web pages, a *converting* module for transforming the web pages into a database-friendly format, a *ranking* module for rating web pages based on their relative importance. We explain each component of the parallel web crawler and implementation methods in detail. Finally, we conduct extensive experiments to analyze the effectiveness of the parallel web crawler. The experimental results clarify the merit of our architecture in that the proposed parallel web crawler is scalable to the number of web pages to crawl and the number of machines used.

Key words : parallel web crawlers, large-scale search engines

1. 서 론

웹의 크기가 폭발적으로 증가함에 따라 인터넷에서 정보를 얻는 수단으로서 검색 엔진의 중요성이 부각되고 있다[1]. 검색 엔진은 웹 페이지를 수집하고, 수집된 웹 페이지에 대해 색인을 구성한다. 그리고, 이를 사용하여 사용자의 질의에 대한 검색 결과를 사용자에게 전달하는 역할을 한다[2]. 대표적인 검색 엔진으로는 Google[3], Naver[4], Yahoo[5] 등이 있다.

검색 엔진은 사용자에게 최신의 정보를 검색 결과로서 제공하기 위해 웹 페이지를 주기적으로 수집한다[3]. 웹 크롤러는 검색 엔진에서 사용되는 웹 페이지를 수집하기 위한 프로그램이다[6]. 웹 크롤러는 시드(seed) URL로 주어진 웹 페이지를 수집하고, 수집된 웹 페이지에서 새로운 URL들을 추출한다. 그리고, 새로운 URL들에 대해 이전의 과정들을 반복한다[7]. 여기서, 시드 URL이란 최초로 수집할 URL들의 집합이다[8].

최근에는 웹의 규모가 커짐에 따라 하나의 프로세스(process) 또는 스레드(thread)만으로 전체 웹 페이지를 수집하기가 어려워지고 있다[6]. 따라서 대부분의 검색 엔진은 제한된 시간 내에 가능한 많은 수의 웹 페이지를 수집하기 위해 병렬 웹 크롤러를 이용한다[3,9,10]. 병렬 웹 크롤러는 여러 개의 프로세스 및 스레드를 이용하여 대량의 웹 페이지들을 빠르게 수집하기 위한 프로그램이다[6]. 병렬 웹 크롤러는 대량의 웹 페이지들을 빠르게 수집할 수 있어야 하며, 시드 URL로 주어진 웹 페이지들뿐만 아니라 이들과 링크된 웹 페이지들도 함께 수집할 수 있어야 한다.

대표적인 병렬 웹 크롤러에는 Multiple Site Crawlers[7]와 Mercator[11]가 있다. Multiple Site Crawlers는 여러 머신의 프로세스들이 웹 페이지들을 동시에 수집하며, 시드 URL로 주어진 웹 사이트들에 해당하는 웹 페이지만을 수집한다[7]. 따라서, Multiple Site Crawlers는 시드 URL로 주어진 웹 사이트 이외의 웹 페이지들을 수집하지 않는다. Mercator는 단일 머신의 스레드들이 웹 페이지들을 동시에 수집하며, 시드 URL로 주어진 웹 사이트들에 해당하는 웹 페이지들과 이들과 링크된 웹 페이지들을 수집한다[11]. 그러나 Mercator는 단일 머신만을 사용하므로 확장 가능하지 않다.

참고문헌[7]에서는 여러 대의 머신을 사용하여 시드 URL로 주어진 웹 사이트 단위로 웹 페이지를 수집하는 병렬 웹 크롤러를 제안하였으나, 이론적인 동작 방식만을 언급하고 있다. 참고문헌[11]에서는 단일 머신을 사용하여 시드 URL에 해당하는 웹 페이지뿐만 아니라 이들과 링크된 웹 페이지들도 수집한다. 그러나, 여러 대의 머신을 동시에 사용할 경우에 대해서는 언급하지 않고 있다. 따라서, 아직까지 병렬 웹 크롤러의 아키텍처와 세부 구현 방법이 잘 알려져 있지 않기 때문에 실제로 병렬 웹 크롤러를 구현하는데 있어서 어려움이 많다.

본 논문에서는 **병렬 웹 크롤러**(*parallel web crawler*)의 아키텍처와 세부 구현 방법을 제시한다. 병렬 웹 크롤러는 Multiple Site Crawlers와 Mercator의 장점을 하이브리드(hybrid)한 아키텍처를 갖는다. 병렬 웹 크롤러는 다수의 머신을 효율적으로 관리하기 위하여 *조정자/대리자*의 2-티어(tier) 모델을 사용하고 각 머신의 자원을 최대한 활용하기 위하여 대리자들은 멀티프로세스/멀티스레드 모델을 사용한다. 조정자는 대리자들이 수집할 사이트 URL들을 관리하며, 대리자는 웹 페이지를 수집하고 새로운 사이트 URL들을 추출한다. 병렬 웹 크롤러는 1) 웹 페이지를 수집하기 위한 *크롤링* 모듈, 2) 수집한 웹 페이지들을 데이타베이스 로딩 포맷으로 변환하기 위한 *컨버팅* 모듈, 3) 수집된 웹 페이지의 중요도를 계산하기 위한 *랭킹* 모듈로 구성된다. 본 논문에서는 병렬 웹 크롤러의 각 모듈을 설명하고, 세부 구현 방법을 설명한다. 마지막으로, 실험을 통해 병렬 웹 크롤러의 성능을 평가한다[12].

본 논문의 공헌은 다음과 같다. 첫째, 대용량 검색 엔진을 위한 병렬 웹 크롤러의 아키텍처를 제안한다. 둘째, 병렬 웹 크롤러의 구현 이슈들과 세부 구현 사항을 설명한다. 셋째, 실험을 통해 제안한 병렬 웹 크롤러가 수집해야할 웹 페이지 개수 및 머신 개수에 따라 확장 가능함을 보인다.

본 논문의 구성은 다음과 같다. 제2장에서는 연구 배경으로서 웹 크롤러와 웹 페이지들의 중요도를 계산하기 위한 알고리즘에 대해 설명한다. 제3장에서는 대용량

검색 엔진을 위한 병렬 웹 크롤러의 시스템 아키텍처를 제안한다. 제4장에서는 병렬 웹 크롤러의 구현 방법을 설명한다. 제5장에서는 병렬 웹 크롤러의 성능을 평가한다. 제6장에서는 향후 연구를 제시하고 결론을 내린다.

2. 관련 연구

본 장에서는 관련 연구로서 기존 웹 크롤러와 BlockRank 알고리즘에 대하여 설명한다. 제2.1절에서는 웹 크롤러의 기본적인 동작 방식과 대표적인 웹 크롤러인 Mercator[11]와 Multiple Site Crawlers[7]에 대해 설명한다. 제2.2절에서는 본 논문에서 웹 페이지의 중요도를 계산하기 위해 사용하는 PageRank[13] 방법의 변형인 BlockRank[14]에 대하여 알아본다.

2.1 웹 크롤러

2.1.1 개요

웹 크롤러는 검색 엔진의 로컬 리포지토리(repository) 내부에 있는 웹 페이지들을 저장, 갱신하기 위한 목적으로 사용된다[15]. 웹 크롤러의 설계 목표는 제한된 자원을 이용하여 짧은 시간 내에 최대한 많은 개수의 웹 페이지를 수집하는 것이다[7]. 그림 1은 웹 크롤러의 기본적인 동작을 나타낸다[7]. 그림에서 실선인 화살표는 수집한 웹 페이지 또는 URL의 흐름을 나타내며, 점선인 화살표들은 웹 크롤러의 동작 흐름을 나타낸다. 먼저, 웹 크롤러에서 웹 페이지의 URL을 관리하기 위한 자료구조에 대해 설명하고, 웹 크롤러의 동작에 대해 설명한다.

웹 크롤러는 웹 페이지의 URL들을 관리하기 위해 큐(queue) 자료구조인 *URLsToVisit*와 *VisitedURLs*를 유지한다[7]. URLsToVisit는 다운로드할 URL들을 저장하기 위해 사용하고, VisitedURLs는 이미 다운로드한 URL들을 저장하기 위해 사용한다. 웹 크롤러는 URLsToVisit에 저장한 시드 URL들을 시작으로 웹 페이지를 수집한다. 여기서, 시드 URL은 사용자에 의해 주어진 URL로서 웹 크롤러가 최초로 수집해야할 URL로 정의된다. 시드 URL들은 사용자가 직접 특정 자료로부터 구하거나, 혹은 수작업을 통해 구한다. 시드 URL들에는 다양한 주제와 관련된 URL들이 포함될 수도 있고, 특정 주제와 관련된 URL들만이 포함될 수도 있다. 대부분의 검색 엔진은 잘 알려진 웹 사이트의 URL들을 시드 URL들로 사용한다[7].

웹 크롤러는 다음과 같은 절차를 통해 웹 페이지를 수집한다. 첫째, 시드 URL을 URLsToVisit에 추가한다. 둘째, URLsToVisit에서 한 개의 URL을 가져온다. 셋째, 가져온 URL에 해당하는 웹 페이지를 다운로드하여 수집한다. 넷째, 수집한 웹 페이지를 디스크에 저장하고, 그 웹 페이지의 URL을 VisitedURLs에 추가한다. 다섯째, 수집한 웹 페이지로부터 URL들을 추출한다. 이때, 추출한 URL들 중에서 VisitedURLs에 포함되지 않은 URL들만을 URLsToVisit에 추가한다. 마지막으로, URLsToVisit가 비워질 때까지 앞의 과정을 반복한다.

2.1.2 대표적인 웹 크롤러

지금까지 웹 크롤러의 기본적인 자료구조와 동작 방식에 대해 살펴보았다. 다음으로 대표적인 웹 크롤러인 Mercator[11]와 Multiple Site Crawlers[7]에 대해 설명한다.

Mercator는 단일 머신에서 여러 스레드들이 동시에 웹 페이지를 수집하는 멀티스레드 방식을 사용한다. Mercator는 사용자에 의해 주어진 URL에 해당하는 웹 페이지들을 수집하고 이들 웹 페이지들과 링크된 다른 웹 페이지들도 수집한다. 그림 2는 Mercator의 아키텍처를 나타낸다. URL 관리자는 수집해야할 URL들을 관리하며, 웹 페이지 수집 및 URL 추출기는 웹 페이지를 수집하고 수집된 웹 페이지에서 URL을 추출한다. 여기서, 웹 페이지 수집 및 URL 추출기는 하나의 스레드이다. Mercator는 다음과 같은 절차에 따라 동작한다. 첫째, URL 관리자가 수집해야할 웹 페이지의 URL을 웹

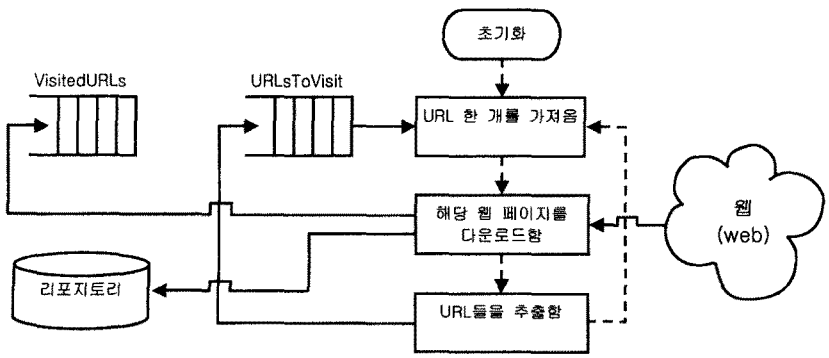

그림 1 웹 크롤러의 기본적인 동작

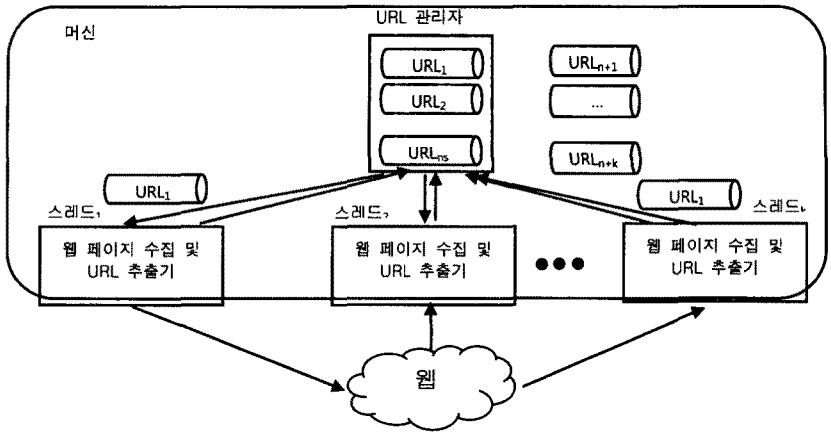

그림 2 Mercator의 아키텍처

페이지 수집기 및 URL 추출기에 전달한다. 둘째, 웹 페이지 수집기 및 URL 추출기는 전달받은 URL에 해당하는 웹 페이지를 웹으로부터 다운로드하고 이 웹 페이지로부터 URL을 추출한다. 셋째, 웹 페이지에서 추출된 새로운 URL을 URL 관리자에 전달한다. 넷째, URL 관리자에 더 이상 수집해야할 웹 페이지의 URL이 없을 때까지 위의 절차를 반복한다.

Multiple Site Crawlers는 여러 머신의 프로세스들이 동시에 웹 페이지를 수집하는 멀티프로세스 방식을 사용한다. Multiple Site Crawlers는 주어진 사이트 URL에 해당하는 웹 페이지만을 수집한다. 여기서, 사이트 URL은 URL 시작 부분의 호스트 이름과 도메인 이름으로 정의된다. 예를 들어, http://pine.kaist.ac.kr/index.html의 사이트 URL은 http://pine.kaist.ac.kr이다. 그림 3은 Multiple Site Crawlers의 아키텍처를 나타낸다. 사이트 URL 관리자는 수집해야할 사이트 URL들을 관리하고 각 머신의 웹 사이트 수집기는 사이트 URL에 해당하는 웹 페이지를 수집한다. 여기서, 웹 사이트 수집기는 하나의 프로세스이다. Multiple Site Crawlers는 다음과 같은 절차에 따라 동작한다. 첫째, 사이트 URL 관리자는 수집할 사이트 URL들을 각 머신에 전달한다. 둘째, 각 머신의 프로세스는 사이트 URL에 해당하는 웹 페이지를 웹으로부터 수집한다.

지금까지 살펴본 Mercator와 Multiple Site Crawlers의 장점과 단점은 다음과 같다. 먼저, Mercator는 시드 URL로 주어진 사이트 내부의 웹 페이지 뿐만 아니라, 이들과 링크된 사이트 외부의 웹 페이지들도 수집할 수 있다는 장점이 있다. 반면에 단일 머신만을 사용하므로 확장 가능하지 않다는 단점이 있다. 다음, Multiple Site Crawlers는 다수의 머신들을 사용하여 웹 페이지를 병

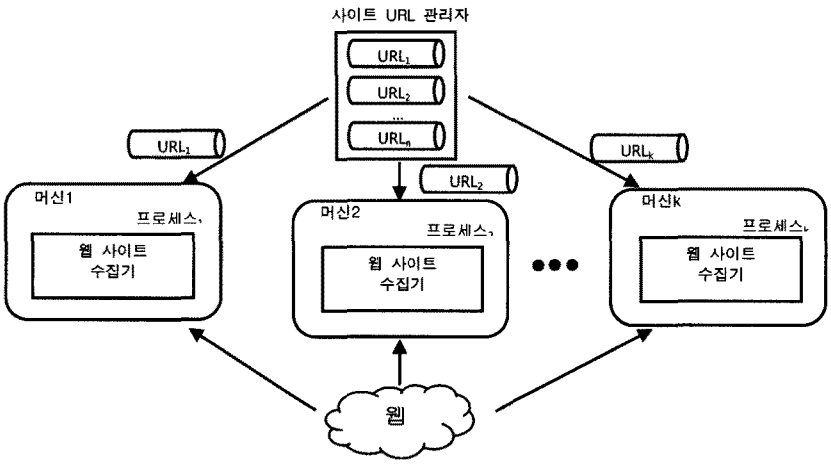

그림 3 Multiple Site Crawlers의 아키텍처

렬적으로 수집하므로 확장 가능하다는 장점이 있다. 반면에 시드 URL로 주어진 웹 페이지와 링크된 사이트 외부의 웹 페이지들을 수집하지 않는다는 단점이 있다. 본 논문에서는 두 웹 크롤러의 장점만을 하이브리드(hybrid)한 병렬 웹 크롤러의 설계와 구현에 대해 제3장과 제4장에서 각각 자세히 설명한다.

2.2 BlockRank

본 절에서는 수집한 웹 페이지의 중요도를 계산하는 방법들 중에서 본 논문에서 제안하는 병렬 웹 크롤러에서 직접적으로 사용하는 방법에 대해 설명한다. 먼저, 웹 페이지들 간의 상대적인 중요도를 측정하는 기본적인 방법인 PageRank[13]를 간단히 설명하고, PageRank의 근사값을 병렬적으로 계산하기 위한 BlockRank [14]에 대해 설명한다.

PageRank는 웹 페이지의 상대적인 중요도를 측정하여 모든 웹 페이지에 랭크(rank)를 부여하는 방법으로 웹의 링크 구조를 기반으로 웹 페이지들 간의 상대적인 중요도를 측정한다[13]. 한 웹 페이지 p의 PageRank $R(p)$는 식 (1)과 같이 계산된다[13]. 식 (1)에서 q는 전체 웹 페이지의 집합 N에 속하는 웹 페이지를 나타내고, $C(q)$는 웹 페이지 q가 링크하고 있는 웹 페이지의 개수 즉, 아웃링크(out-link) 개수를 나타내며, d는 사용자가 웹 페이지 p의 아웃링크를 따라 다른 웹 페이지로 이동할 확률을 나타낸다.

$$R(p) = (1-d) + d\sum_{q \in N} \frac{R(q)}{C(q)} \quad (1)$$

PageRank 알고리즘은 수집된 전체 웹 페이지에 대한 PageRank 값들이 더 이상 변하지 않을 때까지 PageRank의 계산을 반복하므로 수집한 웹 페이지의 개수가 증가할수록 PageRank 계산 시간이 크게 증가한다. 따라서, 대량의 웹 페이지들을 수집하는 병렬 웹 크롤러에서는 웹 페이지들에 대한 정확한 PageRank를 계산하는 대신 PageRank 계산 시간을 단축시키면서 PageRank 값의 근사값을 구하는 방법인 BlockRank[14]를 사용한다.

그림 4는 BlockRank 알고리즘을 나타낸다. BlockRank는 웹 페이지를 여러 개의 블록들로 분할하고, 각 블록에 속하는 웹 페이지들에 대해 PageRank인 로컬 PageRank를 계산한다. 그리고, 각 블록을 하나의 웹 페이지로 간주하고 블록들에 대해 PageRank인 블록 PageRank를 계산한다. 최종적으로, 각 웹 페이지의 로컬 PageRank와 그 웹 페이지가 속한 블록 PageRank를 곱하여 그 웹 페이지에 대한 PageRank의 근사값인 글로벌 PageRank를 계산한다[14].

3. 병렬 웹 크롤러의 설계

3.1 개요 및 아키텍처

병렬 웹 크롤러는 여러 대의 머신을 이용하여 대량의 웹 페이지를 빠르게 수집하기 위하여 *조정자/대리자*의 2-티어 모델을 사용한다. 또한, 각 머신의 자원을 최대한 활용하기 위하여 대리자들은 *멀티프로세스/멀티스레드* 모델을 사용한다. 멀티스레드 방식은 동시에 여러 웹 페이지들을 수집하기 때문에 웹 페이지를 하나씩 수집하는 싱글스레드 방식에 비해 짧은 시간에 대량의 웹 페이지들을 수집할 수 있다는 장점이 있다. 한 개의 스레드가 한 개의 웹 사이트에 속하는 웹 페이지들의 다운로드를 담당하고 이러한 동작을 하는 스레드들은 서로 독립적으로 동작한다. 따라서, 이 스레드들은 서로를 간섭하지 않으며 동시에 여러 웹 페이지를 비동기적으로 다운로드 한다. 멀티프로세스 방식은 싱글프로세스 방식에 비해 더 많은 스레드들을 생성할 수 있다는 장점이 있다. 한 프로세스가 생성할 수 있는 스레드의 개수에 제한이 있기 때문에 더 많은 스레드가 필요한 시점에서 추가로 프로세스를 생성시킴으로써 필요한 만큼의 스레드들을 추가로 생성한다. 그 결과, 한 머신에서

BlockRank 알고리즘 [14]:
입력: 웹 페이지들의 집합
출력: 웹 페이지들의 PageRank
알고리즘:
1. 도메인이 같은 웹 페이지들이 같은 block에 속하도록 웹 페이지들을 분할한다.
2. 각 블록에 대해서 로컬 PageRank를 계산한다.
3. 각 블록을 웹 페이지로 간주하고 블록 PageRank를 계산한다.
4. 각 웹 페이지에 대한 로컬 PageRank 값과 그 웹 페이지가 속한 블록의 블록 PageRank 값을 곱하여 그 웹 페이지에 대한 글로벌 PageRank 값을 구한다.
5. 글로벌 PageRank 값이 변하지 않을 때까지 위의 과정을 반복한다.

그림 4 Kamvar et al. [14]의 BlockRank 알고리즘

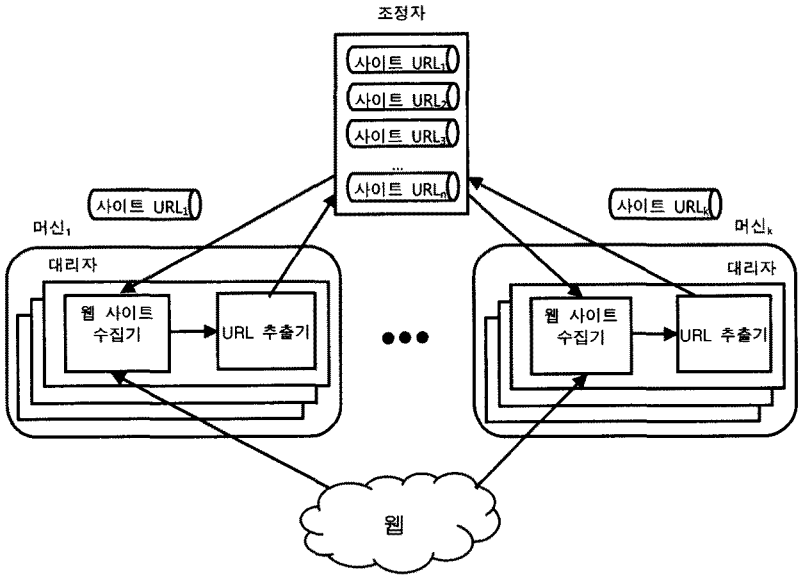

그림 5 병렬 웹 크롤러의 아키텍처

이용 가능한 자원을 최대한 활용할 수 있다.

그림 5는 병렬 웹 크롤러의 아키텍처를 나타낸다. 조정자는 대리자들이 수집할 사이트 URL들을 관리하며, 대리자들은 웹 사이트 내의 웹 페이지를 병렬적으로 수집하고 새로운 사이트 URL들을 추출한다.

병렬 웹 크롤러의 동작 과정은 *크롤링*(crawling), *컨버팅*(converting), *랭킹*(ranking)의 세 단계로 나누어진다. 다음은 각 단계에 대한 간단한 설명이다.

1. 크롤링 단계: 웹에서 웹 페이지를 다운로드한 후, 그 웹 페이지의 내용을 디스크에 저장한다.
2. 컨버팅 단계: 저장된 웹 페이지의 내용으로부터 데이타베이스 스키마에 정의된 각 애트리뷰트의 값을 추출한 후, 이를 데이타베이스 로딩 형식에 맞추어 변환한다.
3. 랭킹 단계: 수집한 웹 페이지의 링크 정보를 이용하여 웹 페이지들 간의 중요도를 계산한다.

3.2 크롤링 단계

병렬 웹 크롤러의 크롤링 단계에서는 여러 대의 머신이 동시에 웹 사이트 단위로 웹에서 웹 페이지들을 빠르게 수집하여 디스크에 저장한다. 이때, 웹 페이지의 중복 수집을 가급적 줄이고 중복 수집된 웹 페이지를 제거한다.

병렬 웹 크롤러는 다음과 같은 이유 때문에 웹 사이트 단위로 웹 페이지들을 다운로드하는 방식을 사용한다. 첫째, 웹 사이트의 규모와 그 웹 사이트에 대한 네트워크 대역폭을 고려하여 그 웹 사이트에 속하는 웹 페이지들의 다운로드 주기를 조절할 수 있다. 둘째, 하나의 웹 사이트 내의 웹 페이지들은 동일한 규칙을 따르기 때문에 The Robots Exclusion Protocol[16]과 같은 프로토콜을 사용하여 수집이 금지된 웹 페이지들을 수집하지 않도록 웹 사이트 단위로 일관되게 규칙을 적용할 수 있다. 셋째, 웹 페이지 단위로 웹 크롤링을 수행하는 방식에 비해 네트워크와 메모리 사용량을 줄일 수 있다. 대부분의 경우 웹 사이트는 여러 개의 웹 페이지들로 구성되기 때문에 웹 사이트의 URL 만으로도 그 웹 사이트와 링크된 웹 페이지들을 한꺼번에 다운로드할 수 있다. 결국, 다운로드할 웹 페이지의 URL들을 별도로 미리 유지할 필요가 없으므로 메모리 사용량을 줄일 수 있고, 조정자와 대리자들 간에 주고 받는 메시지(예, 수집할 사이트 URL들의 집합) 개수가 줄어들기 때문에 네트워크 사용량을 줄일 수 있다.

병렬 웹 크롤러는 여러 머신에서 동시에 여러 웹 페이지를 수집하기 때문에, 동일한 웹 페이지가 중복으로 수집될 수 있다. 웹 페이지가 중복되어 수집되는 경우는 크게 두 가지로 나눌 수 있다. 첫째, 서로 다른 대리자가 동일한 웹 사이트를 중복 수집하는 경우이다. 즉, 웹 사이트의 URL이 다르지만 IP 주소가 같은 경우이다. 웹 사이트의 중복 수집은 서로 다른 사이트 URL들이 동일한 웹 사이트를 가리킬 수 있기 때문이다. 예를 들어, http://organ.kaist.ac.kr(domain name), http://dblab.kaist.ac.kr(nickname), http://143.248.138.163(IP 주소)는 모두 동일한 웹 사이트를 가리킨다. 둘째, 한 웹 사이트 내에서 동일한 웹 페이지를 수집하는 경우이다. 즉, 웹 페이지의 URL이 다르지만 웹 페이지의 내용이 동일한 경우

이다. 웹 페이지의 중복 수집은 서로 다른 URL들이 동일한 웹 페이지를 가리킬 수 있기 때문이다. 예를 들어, http://www.samsungatofina.com/indx.html;sessionid=1, http://www.samsungatofina.com/indx.html;sessionid=2는 동일한 웹 페이지를 가리킨다.

제안하는 병렬 웹 크롤러는 (1) 웹 사이트의 중복 수집을 방지하기 위한 방법과 (2) 웹 사이트 내에서 중복 수집된 웹 페이지를 제거하는 방법을 사용한다. 먼저, 웹 사이트의 중복 수집을 방지하기 위해서 조정자에서 사이트 URL을 IP 주소로 변환한 후, 중복되는 IP 주소를 제거한다. 한 개의 URL은 도메인명 서버(DNS)를 통해 특정 IP 주소로 변환한다. 이때, 서로 다른 여러 개의 URL이 같은 IP 주소를 가질 수 있으므로 조정자에서는 사이트 URL 그대로를 각 대리자에 분배하지 않고, IP 주소로 변환하여 중복을 제거한 후에 분배한다. 다음, 중복 수집된 웹 페이지를 제거하기 위해서 웹 페이지의 내용에 대한 체크섬 값을 구하고, 체크섬 값이 같은 웹 페이지를 제거한다[11]. 본 논문에서는 웹 페이지의 체크섬을 계산하기 위해 secure hash algorithm [17]을 사용한다. 여기서, secure hash algorithm은 텍스트 파일을 입력으로 받아 128-bit의 해시(hash) 값을 생성한다. 해시 값을 사용하기 때문에 서로 다른 웹 페이지가 같은 해시 값을 가질 수는 있지만 그와 같은 경우가 발생할 확률이 매우 낮기 때문에[17] 본 논문에서는 이러한 경우가 발생하지 않는다고 가정한다.

병렬 웹 크롤러의 크롤링 알고리즘은 조정자 알고리즘과 대리자 알고리즘으로 구성된다. 그림 6은 조정자 알고리즘을 나타낸다. 조정자의 크롤링 모듈은 시드 URL을 수집할 사이트 URL의 집합 V_{cr}에 추가한다. 그리고, V_{cr}로부터 대리자가 수집할 사이트 URL들을 대리자에 전달하고, 이를 수집한 사이트 URL들의 집합 V_{co}에 추가한다. 마지막으로, 대리자로부터 조정자로 보내진 새로운 사이트 URL들의 집합 P에 포함된 URL들 중에서 $V_{co} \cup V_{cr}$에 존재하지 않는 URL들을 V_{cr}에 추가한다.

그림 7은 대리자 알고리즘을 나타낸다. 대리자의 크롤링 모듈은 조정자가 전송한 사이트 URL에 해당하는 웹 페이지들을 웹에서 다운로드하여 저장하고, 이들 웹 페이지들의 URL들을 사이트 내에서 수집한 URL들의 집합 A_{co}에 추가한다. 그리고 웹 페이지에서 링크를 추출한 뒤 추출된 링크가 사이트 내부에 있을 경우 수집할 URL들의 집합 A_{cr}에 추가하고, 사이트 외부에 있을 경우 조정자에 전송할 URL들의 집합 A_{new}에 추가한다. A_{cr}이 비워질 때까지 위의 웹 페이지 수집 및 URL 추출 과정을 반복한다. 마지막으로, A_{new}를 조정자에 전달한다.

3.3 컨버팅 단계

병렬 웹 크롤러의 컨버팅 단계에서는 크롤링 단계에서 수집한 웹 페이지로부터 검색 엔진의 데이타베이스 스키마에서 정의된 애트리뷰트들의 값들을 추출한다. 그리고 나서 사용하는 데이타베이스 관리 시스템(DBMS)의 로딩 형식에 맞추어 데이타베이스에 저장할 데이타를 생성한다. 본 논문에서는 검색 엔진으로 오디세우스 DBMS[18]를 사용한다. 오디세우스는 정보 검색 기능을 지원하는 객체 관계형 데이타베이스 관리시스템이다.

검색 엔진에서 사용하는 데이타베이스 스키마의 애트

병렬 웹 크롤러의 알고리즘:
입력: 수집할 사이트 URL의 집합 S
출력: 수집한 웹 페이지들의 집합
조정자 알고리즘:
1. $V_{co} := \{\}$ /* 다운로드한 사이트 URL들의 집합 */
2. $V_{cr} := S$ /* 다운로드할 사이트 URL들의 집합 */
3. FOR $u \in V_{cr}$ DO
   3.1 u를 대리자에 전달한다.
   3.2 $V_{co} := V_{co} \cup \{u\}$
   3.3 $V_{cr} := V_{cr} - \{u\}$
   3.4 만약 대리자로부터 온 사이트 URL들의 집합 P가 있다면 다음을 수행한다.
      FOR $u \in P$ DO /* P: 대리자로부터 온 사이트 URL들의 집합 */
      3.4.1 IF $u \neq V_{co}$ AND $u \neq V_{co}$ DO
            $V_{cr} := V_{cr} - \{u\}$
      3.4.2 $P := P - \{u\}$

그림 6 조정자의 크롤링 알고리즘

대리자 알고리즘:
A_{co} := {} /* 사이트 내부에서 수집한 URL들의 집합 */
A_{cr} := {} /* 사이트 내부에서 수집할 URL들의 집합 */
A_{new} := {} /* 조정자에 전달할 URL들의 집합 */
S : = 조정자가 전송한 사이트 URL들의 집합
1. FOR $u \in S$ DO
   1.1 u에 해당하는 웹 페이지를 수집하여 저장한다.
   1.2 $A_{co} := A_{co} \cup \{u\}$
   1.3 $A_{cr} := A_{cr} - \{u\}$
   1.4 u에 해당하는 웹 페이지에서 추출한 URL들의 집합 Q에 대해 다음의 과정을 수행한다.
      1.4.1 FOR $q \in Q$ DO
         만약 q가 u가 속한 사이트와 같다면 /* 즉, 사이트 내부에 있음 */
            $A_{cr} := A_{cr} \cup (\{q\} - A_{co})$
         그렇지 않다면 /* 즉, 사이트 외부에 있음 */
            $A_{new} := A_{new} \cup \{q$의 사이트$\}$
   1.5 FOR $s \in A_{cr}$ DO
      1.5.1 s에 해당하는 웹 페이지를 수집하여 저장한다.
      1.5.2 $A_{co} := A_{co} \cup \{s\}$
      1.5.3 $A_{cr} := A_{cr} - \{s\}$
2. A_{new}를 조정자로 전송한다.

그림 7 대리자의 크롤링 알고리즘

리뷰트들은 글로벌 애트리뷰트(global attribute)와 로컬 애트리뷰트(local attribute)로 분류된다. 먼저, 글로벌 애트리뷰트는 해당 애트리뷰트의 값을 알기 위해서 모든 대리자들에서 수집한 웹 페이지들의 정보가 필요한 애트리뷰트로 정의된다. 이러한 글로벌 애트리뷰트에는 pageID, siteID, domainID, communityID가 해당된다. pageID는 유일하게 구분되는 각 웹 페이지마다 부여되는 식별자(ID)이므로 이 값을 알기 위해서는 다른 대리자에서 수집된 웹 페이지들의 개수가 필요하다. siteID, domainID, communityID는 각각 사이트 제한 검색, 도메인 제한 검색, 커뮤니티 제한 검색을 위한 ID로서 유일하게 구분되는 각 사이트, 도메인, 커뮤니티마다 부여되는 ID이다[8]. 앞에서 설명한 pageID와 마찬가지로, 이 값들을 알기 위해서는 다른 대리자에서 수집된 사이트, 도메인, 커뮤니티들의 개수가 필요하다. 다음, 로컬 애트리뷰트는 해당 애트리뷰트의 값을 알기 위해서 다른 대리자에서 수집한 웹 페이지들의 정보가 필요 없는 애트리뷰트이다. 이러한 로컬 애트리뷰트에는 웹 페이지의 제목과 설명(description)이 있다.

병렬 웹 크롤러에서의 컨버팅 알고리즘은 조정자 알고리즘과 대리자 알고리즘으로 구성된다. 그림 8은 조정자 알고리즘을 나타낸다. 조정자 알고리즘은 글로벌 애

컨버팅 알고리즘:
입력: 변환할 웹 페이지들의 집합 S
출력: 변환된 웹 페이지들의 집합
조정자 알고리즘:
1. 각 대리자에서 전송받은 사이트 URL들과 웹 페이지 개수를 글로벌 애트리뷰트의 값을 생성한다.
   1.1 유일하게 구분되는 URL에 siteID를 부여한다.
   1.2 각 siteURL의 도메인을 분리하여 유일하게 구분되는 도메인에 domainID를 부여한다.
   1.3 관련된 사이트 URL들에 동일한 communityID를 부여한다.
   1.4 각 대리자가 수집한 웹 페이지의 개수를 이용하여 대리자들 간의 pageID의 범위가 서로 겹치지 않도록 각 대리자가 변환할 웹 페이지의 pageID 범위를 생성한다.
2. 생성된 글로벌 애트리뷰트의 값들을 각 대리자에 전달한다.

그림 8 조정자의 컨버팅 알고리즘

트리뷰트인 siteID, domainID, communityID의 값들을 생성하고, 끝으로 각 대리자에서 수집된 웹 페이지들이 가질 수 있는 pageID의 범위를 생성한다. siteID는 유일하게 구분되는 사이트 URL마다 ID가 부여된다. dominaID는 사이트 URL에서 도메인(domain)을 분리하여 유일하게 구분되는 도메인마다 ID를 부여한다. 또한 communityID는 특정 주제와 관련된 사이트 URL들에 동일한 ID를 부여한다. 다음으로 각 대리자가 수집한 웹 페이지의 전체 개수를 이용하여 각 대리자가 컨버팅할 웹 페이지의 pageID 범위를 생성하고, 생성된 글로벌 애트리뷰트들을 각 대리자에 전달한다.

그림 9는 대리자 알고리즘을 나타낸다. 대리자의 컨버팅 모듈은 글로벌 애트리뷰트 생성에 필요한 정보인 해당 대리자에서 수집한 전체 웹 페이지의 개수를 조정자로 전송한다. 또한, 대리자의 컨버팅 모듈은 로컬 애트리뷰트를 생성하고 DBMS의 로딩형식에 맞추어 웹 페이지를 변환한다. 로컬 애트리뷰트는 웹 페이지의 HTML 테그(tag)를 분석하여 해당 웹 페이지의 제목과 설명을 추출한다. 마지막으로 조정자로부터 전달받은 pageID의 범위 내에서 유일하게 구분되는 웹 페이지마다 ID를 부여한다.

3.4 랭킹 단계

병렬 웹 크롤러의 랭킹 단계에서는 수집된 웹 페이지들의 링크 정보를 이용하여 웹 페이지들 간의 중요도를 계산한다. 병렬 웹 크롤러는 웹 페이지들의 중요도를 계산하기 위해 제2.2절에서 설명한 BlockRank 알고리즘을 사용한다. BlockRank 알고리즘을 사용하는 이유는 여러 대의 머신에서 웹 페이지의 중요도를 병렬적으로 계산함으로써, 랭킹 단계에서 소요되는 시간을 크게 감소시킬 수 있기 때문이다.

그림 10은 병렬 웹 크롤러에서의 랭킹 알고리즘을 나타낸다. 대리자의 랭킹 알고리즘은 도메인이 같은 웹 페이지들이 같은 블록에 속하도록 수집한 웹 페이지들을 여러 개의 블록들로 분할한다. 그리고, 각 블록을 하나의 웹 페이지로 간주하고, 각 블록의 링크 정보를 조정자에 전송한다. 조정자의 랭킹 알고리즘은 대리자로부터 전달받은 각 블록의 링크 정보를 이용하여 블록 PageRank를 계산한 후에 각 블록의 블록 PageRank를 해당 대리자에 전달한다. 대리자는 수집한 웹 페이지들에 대해 로컬 PageRank를 계산한 뒤, 각 웹 페이지의 로컬 PageRank와 그 웹 페이지가 속하는 블록의 블록 Page-

대리자 알고리즘:

1. 수집한 전체 웹 페이지들의 개수를 조정자에게 전송한다.
2. 조정자로부터 글로벌 애트리뷰트들의 값들을 전송 받는다.
3. 로칼 애트리뷰트의 값을 생성하고 데이타베이스 로딩 형식에 맞추어 웹 페이지를 변환한다.
   - 3.1 웹 페이지의 HTML 테그를 분석하여 해당 웹 페이지의 제목과 설명을 추출한다.
   - 3.2 조정자로부터 전달받은 pageID의 범위 내에서 유일하게 구분되는 각 웹 페이지에 pageID를 부여한다.

그림 9 대리자의 컨버팅 알고리즘

랭킹 알고리즘:
입력: 웹 페이지들의 집합
출력: 각 웹 페이지의 랭크
조정자 알고리즘:

1. 각 대리자로부터 전송 받은 블록 정보를 이용하여 블록 PageRank를 계산한다.
2. 각 블록에 대한 블록 PageRank 값을 해당 대리자에게 전송한다.

대리자 알고리즘:

1. 도메인이 같은 웹 페이지들이 같은 블록에 속하도록 수집한 웹 페이지들을 분할한다.
2. 각 블록을 웹 페이지로 간주하고, 각 블록의 링크 정보를 조정자에게 전송한다.
3. 각 블록에 대해 로컬 PageRank를 계산한다.
4. 각 웹 페이지의 로컬 PageRank 값과 조정자로부터 전송 받은 그 웹 페이지가 속하는 블록의 블록 PageRank 값을 곱하여 그 웹 페이지의 글로벌 PageRank 값을 계산한다.

그림 10 병렬 웹 크롤러의 랭킹 알고리즘

Rank를 곱하여 그 웹 페이지의 최종 PageRank인 글로벌 PageRank를 계산한다.

4. 병렬 웹 크롤러의 구현

본 장에서는 제3장에서 제시한 병렬 웹 크롤러의 자세한 구현 방법에 대해 살펴본다.

4.1 크롤링 모듈

병렬 웹 크롤러의 크롤링 모듈은 조정자의 크롤링 모듈과 대리자의 크롤링 모듈로 구성된다. 조정자의 크롤링 모듈은 대리자들이 수집할 웹 사이트의 URL을 관리한다. 대리자의 크롤링 모듈은 웹 크롤링 프로세스가 생성할 수 있는 스레드의 개수를 조절하면서 웹 페이지를 수집하고 새로운 사이트 URL을 추출한다. 그림 11은 병렬 웹 크롤러의 크롤링 모듈을 나타낸다.

표 1은 조정자의 크롤링 모듈에서 사용되는 자료구조를 나타낸다. TotalToVisitQueue는 수집할 웹 사이트의 URL들을 저장하기 위한 자료구조이고, TotalVisitedQueue는 수집된 웹 사이트의 URL들을 저장하기 위한 자료구조이다.

조정자의 크롤링 모듈은 수집할 사이트 URL들을 관리하는 시드 URL 분배기로 구성된다. 먼저 기존 검색 엔진에 등록된 웹 사이트들을 크롤링하여 구한 시드 URL들을 TotalVisitedQueue에 추가한다. 그리고 도메인명 서버(DNS)로부터 각 시드 URL에 해당하는 IP를 얻어온다. 다음으로 얻어온 IP를 각 대리자에 전달한다. 마지막으로, 대리자가 발견한 새로운 사이트 URL들을 TotalToVisitQueue에 추가한다. 이때, 각 큐에 이미 존재하는 URL은 중복하여 저장하지 않는다.

표 1 조정자의 크롤링 모듈에서 사용되는 자료구조

변수 이름	데이타 타입	설명
TotalToVisitQueue	queue	수집할 URL들을 저장
TotalVisitedQueue	queue	수집된 URL들을 저장

표 2 대리자의 크롤링 모듈에서 사용되는 자료구조

변수 이름	데이타 타입	설명
ToVisitQueue	queue	사이트 내에서 수집할 URL들을 저장
VisitedQueue	queue	사이트 내에서 수집된 URL들을 저장
ToSendQueue	queue	사이트 외부의 URL들을 저장
ContentsTable	hash table	웹 페이지의 체크섬 값을 저장

표 2는 대리자의 크롤링 모듈에서 사용되는 자료구조를 나타낸다. ToVisitQueue는 사이트 내부에서 수집해야할 웹 사이트의 URL들을 저장하고, VisitedQueue는 수집된 웹 페이지의 URL들을 저장한다. ToSendQueue는 사이트 외부의 URL들을 저장하며, ContentsTable은 수집된 웹 페이지의 내용에 대한 128-bit 체크섬 값을 저장한다.

대리자의 크롤링 모듈은 웹 페이지들을 수집하고 웹 페이지 내의 URL들을 추출한다. 대리자의 크롤링 모듈은 크게 웹 페이지 다운로드기(webpage downloader) 모듈, 중복 체크기(replicate checker) 모듈, 링크 추출기(link extractor) 모듈로 구성된다. 웹 페이지 다운로드기 모듈은 조정자의 크롤링 모듈로부터 전달받은 웹 사이트의 IP를 ToVisitQueue에 추가하고, 이들 웹 페이지들을 웹에서 수집한다. 중복 체크기 모듈은 수집한 웹 페이지의 내용에 대해 128-bit 체크섬 값을 구하고 이 웹 페이지의 체크섬 값이 컨텐츠 테이블에 존재하는지를 검사한다. 이때, 체크섬 값이 컨텐츠 테이블에 존재하지 않을 경우, 중복이 없다고 판단하고 새로운 체크섬 값을 ContentsTable에 추가한 후에 이 웹 페이지의 URL을 VisitedQueue에 추가한다. 링크 추출기 모듈은 웹 페이지에서 URL들을 추출하여 사이트 내부의 URL이면서 이미 수집된 URL일 경우 VisitedQueue에 추가

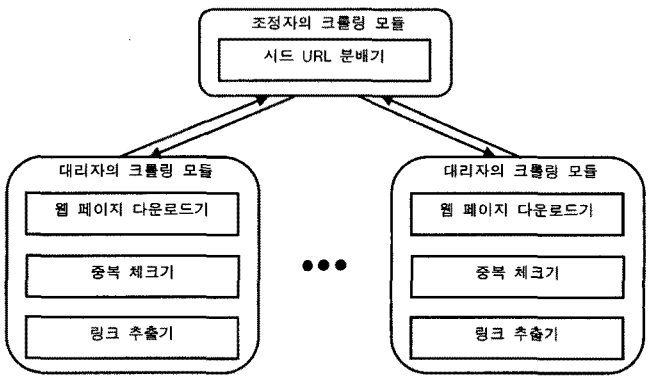

그림 11 병렬 웹 크롤러의 크롤링 모듈

하고, 아직 수집되지 않은 URL일 경우 ToVisitQueue에 추가한다. 만약 사이트 외부의 URL일 경우에는 이 URL을 ToSendQueue에 추가한다. 이때, 조정자에서와 마찬가지로 각 큐에 이미 존재하는 URL은 중복하여 저장하지 않는다.

4.2 컨버팅 모듈

컨버팅 모듈은 세부적으로 조정자의 컨버팅 모듈과 대리자의 컨버팅 모듈로 구성된다. 조정자의 컨버팅 모듈은 글로벌 애트리뷰트에 해당하는 값을 생성한다. 대리자의 컨버팅 모듈은 로컬 애트리뷰트에 해당하는 값을 생성하고 수집한 웹 페이지의 URL을 pageID로 변환한다. 그림 12는 병렬 웹 크롤러의 컨버팅 모듈을 나타낸다.

표 3은 조정자의 컨버팅 모듈에서 사용되는 자료구조를 나타낸다. IDDic은 하나의 사이트 URL에 해당하는 사이트 ID, 도메인 ID, 카테고리 ID를 저장하기 위한 딕셔너리(dictionary)이다.

조정자의 글로벌 애트리뷰트 모듈은 글로벌 애트리뷰트의 값을 생성한다. 글로벌 애트리뷰트에는 웹 사이트, 도메인, 및 클러스터 ID가 해당된다. 먼저, 유일하게 구분되는 각 사이트 URL마다 ID를 부여한다. 도메인 및 클러스터에 대해서도 사이트 URL과 마찬가지 방식으로 ID를 부여한다.

표 4는 대리자의 컨버팅 모듈에서 사용되는 자료구조를 나타낸다. IDDic은 하나의 사이트 URL에 해당하는 사이트 ID, 도메인 ID, 카테고리 ID를 저장하기 위한 딕셔너리이다. PageIdDic은 각 웹 페이지 URL에 해당하는 페이지 번호를 저장하기 위한 딕셔너리이다.

대리자의 웹 페이지 컨버터(webpage converter) 모듈은 로컬 애트리뷰트 값을 생성하고 크롤링 모듈이 수집한 웹 페이지들을 데이타베이스 로딩 형식으로 변환(컨버팅)한다. 웹 페이지는 데이타베이스의 스키마에 맞추어 웹 사이트와 웹 페이지 정보로 나누어 변환한다. 웹 사이트 정보는 각 사이트의 인덱스(index) 페이지의 정보를 이용하여 구한다.

대리자의 링크 컨버터(link converter) 모듈은 링크를 pageID로 변환한다. 이때, URL로 표현된 웹 페이지의 링크를 더 짧은 길이의 pageID로 변환함으로써 수집된 웹 페이지들의 랭킹 계산에 필요한 입력 파일의 크기를 줄인다.

4.3 랭킹 모듈

랭킹 모듈은 조정자의 랭킹 모듈과 대리자의 랭킹 모듈로 구성된다. 조정자의 랭킹 모듈은 블록들의 랭크(rank)를 계산한다. 대리자의 랭킹 모듈은 텍스트(text) 파일을 바이너리(binary) 파일로 변환한 뒤, 웹 페이지들의 로컬 랭크 및 글로벌 랭크를 계산한다. 그림 13은 병렬 웹 크롤러의 랭킹 모듈을 나타낸다.

표 5는 조정자의 랭킹 모듈에서 사용되는 자료구조를 나타낸다. 여기서, source 어레이(array)는 i 번째 반복을 통해 구한 랭크 계산 결과를 저장하는 어레이이고, destination 어레이는 i+1 번째 반복을 통해 구한 랭크 계산 결과를 저장하는 어레이이다. 이와같이 두 가지 어

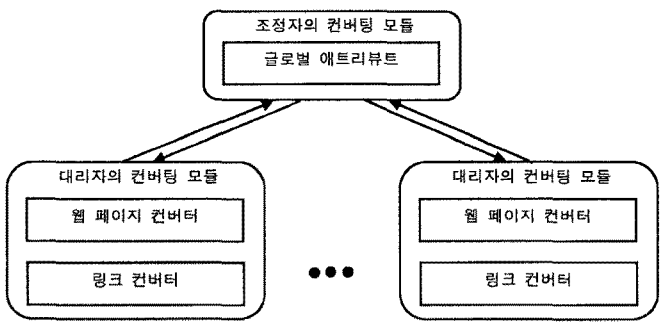

그림 12 병렬 웹 크롤러의 컨버팅 모듈

표 3 조정자의 컨버팅 모듈에서 사용되는 자료구조

변수 이름	데이타 타입	키 (Key)	값 (Value)
IDDic	hash table	사이트 URL	(사이트 ID, 도메인 ID, 카테고리 ID)

표 4 대리자의 컨버팅 모듈에서 사용되는 자료구조

변수 이름	데이타 타입	키 (Key)	값 (Value)
IDDic	dictionary	사이트 URL	(사이트 ID, 도메인 ID, 카테고리 ID)
PageIdDic	dictionary	웹 페이지 URL	페이지 ID

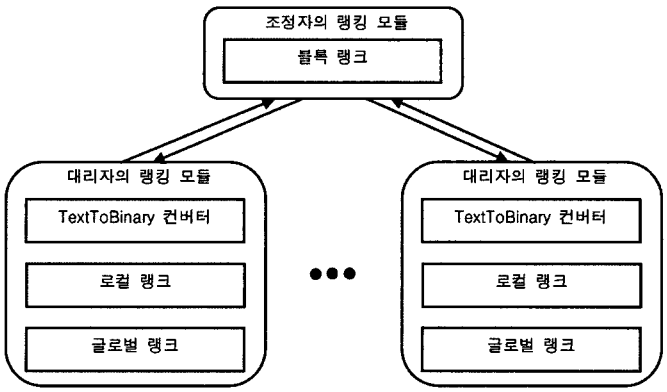

그림 13 병렬 웹 크롤러의 랭킹 모듈

표 5 조정자의 랭킹모듈에서 사용되는 자료구조

변수 이름	데이타 타입	설명
Source	array	i 번째 iteration의 PageRank 값을 저장
Destination	array	i+1 번째 iteration의 PageRank 값을 저장

표 6 대리자의 랭킹 모듈에서 사용되는 자료구조

변수 이름	데이타 타입	설명
source	array	i 번째 iteration의 로칼 PageRank 값을 저장
destination	array	i+1 번째 iteration의 로칼 PageRank 값을 저장

레이 자료 구조를 유지하는 이유는 i 번째 이터레이션을 통해 구해진 랭크 값과 i+1 번째 반복을 통해 구해진 랭크 값의 차이의 총합인 리지듀얼(residual) 값을 구하고, 이 값이 임계값(threshold)보다 작아졌을 경우에 랭크 계산을 종료하기 위함이다.

조정자의 블록 랭크 모듈은 대리자로부터 전달받은 블록 정보를 이용하여 블록들의 랭크를 계산한다. 이때, 블록의 단위로는 웹 사이트 혹은 커뮤니티를 사용한다. 블록들의 랭크를 구하는 방법으로는 제 3장에서 설명한 BlockRank 계산 방법을 사용한다. 즉, 블록을 하나의 웹 페이지로 가정하고 PageRank 알고리즘을 적용시킨다.

표 6은 대리자의 랭킹 모듈에서 사용되는 자료구조를 나타낸다. 여기서, source 어레이는 i 번째 반복을 통해 구한 랭크 계산 결과를 저장하는 어레이이고, destination 어레이는 i+1 번째 반복을 통해 구한 랭크 계산 결과를 저장하는 어레이이다. 이와같이 두 가지 어레이 자료 구조를 유지하는 이유는 앞에서 설명한 조정자의 경우와 동일하다.

대리자의 랭킹 모듈은 세부적으로 TextToBinary 컨버터(converter), 로컬 랭커(local ranker), 글로벌 랭커(global ranker)로 구성된다. 먼저, TextToBinary 컨버터는 pageID로 표현된 한 웹 페이지의 링크를 *Link 자료 구조*로 변환한다. 표 7은 링크 자료 구조를 나타낸다. src는 소스(source) 웹 페이지의 pageID를 나타내고 nLink는 소스 웹 페이지에서 다른 웹 페이지로 나가는 URL의 개수를 나타낸다. 끝으로, dest는 소스 웹 페이지에서 다른 웹 페이지로 나가는 URL들의 어레이(array)에 대한 포인터를 나타낸다. 다음, 로컬 랭커는 블록 내에서 웹 페이지의 랭크를 계산한다. 끝으로, 글로벌 랭커는 로컬 랭크와 블록 랭크를 곱하여 웹 페이지의 글로벌 랭크를 계산한다.

표 7 링크 자료구조

변수 이름	데이타 타입	설명
src	pageID	웹 페이지의 ID
nLink	numPageID	아웃링크의 개수
dest	pageID	아웃링크 어레이(array)에 대한 포인터

조정자의 블록 랭크(rank) 모듈, 대리자의 로컬 랭크 모듈, 글로벌 랭크 모듈은 공통적으로 다음의 과정을 통해 수집한 웹 페이질의 랭크를 계산한다. 먼저, source 어레이(array) 및 destination 어레이의 모든 원소들의 값을 초기화한다. 그리고 나서 source 어레이와 destination 어레이의 랭크 값을 이용하여 리지듀얼(residual)을 구하고 이 값이 임계값(threshold)보다 클 동안 다음의 과정을 반복한다. (1) 먼저 하나의 링크 자료 구조에 기록된 데이타를 읽어온다. (2) 읽어 온 데이타를 이용하여 랭크를 계산한다. (3) 계산된 랭크를 이용하여 리

지듀얼 값을 계산한다. 만약 리지듀얼 값이 임계값 보다 작을 경우에는 source 어레이에 저장된 결과를 파일에 기록한 후에 계산을 종료한다.

5. 병렬 웹 크롤러의 성능 평가

본 장에서는 병렬 웹 크롤러의 성능을 측정하고, 그 결과를 설명한다. 제5.1절에서는 실험 환경과 실험에 사용한 데이타에 대해 설명하고, 제5.2절에서는 실험 결과에 대해 설명한다.

5.1 실험 환경

본 논문에서는 병렬 웹 크롤러의 우수성을 입증하기 위하여 병렬 웹 크롤러의 성능을 측정한다. 본 논문에서는 웹 크롤링을 위한 초기 URL로 상용 검색 엔진인 Naver[4]에 등록된 웹 사이트들을 사용한다. 실험은 총 9대의 펜티엄4 2.0GHz Linux PC(메모리: 512MB)를 사용하여 수행한다.

본 실험에서는 성능 평가 척도로 웹 페이지 수집에 소요된 시간(wall clock time, second)을 사용한다. 수집할 웹 페이지의 개수와 머신 개수를 파라미터로서 변화시키면서 웹 페이지 수집에 소요된 시간을 측정한다. 이 때, 수집할 웹 페이지의 개수를 100,000, 1,000,000, 10,000,000으로 변화시키고, 머신 개수를 1, 2, 4, 8로 변화시킨다.

5.2 실험 결과

그림 14는 수집할 웹 페이지의 개수에 따른 제안한 병렬 웹 크롤러의 성능 측정 결과를 나타낸다. 가로축은 수집할 웹 페이지의 개수를 나타내며, 세로축은 소요 시간을 나타낸다. 이 때, 머신 개수는 아홉 개이다. 그림 14에서 알 수 있듯이, 수집할 웹 페이지의 개수가 증가함에 따라 웹 페이지 수집에 소요되는 시간이 거의 선형으로 증가함을 알 수 있다.

병렬 웹 크롤러의 머신 개수에 따른 성능 측정 결과는 그림 15와 같다. 가로축은 머신 개수를 나타내며, 세로축은 소요 시간을 나타낸다. 이 때, 수집할 웹 페이지의 개수는 1,000,000개이다. 그림 15에서 알 수 있듯이, 머신 개수가 증가할수록 웹 페이지 수집에 소요되는 시간이 감소하는 것을 알 수 있다. 그러나 머신 개수가 두 배로 증가한다고 해서, 소요 시간이 반드시 반으로 줄어드는 것은 아니다. 그 이유는 여러 대의 머신이 하나의 100Mbps 허브(hub)에 연결되어 있어 네트워크 대역폭

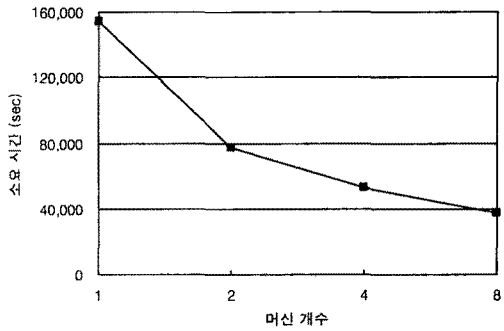

그림 14 수집할 웹 페이지 개수에 따른 소요 시간(머신 개수 = 9)

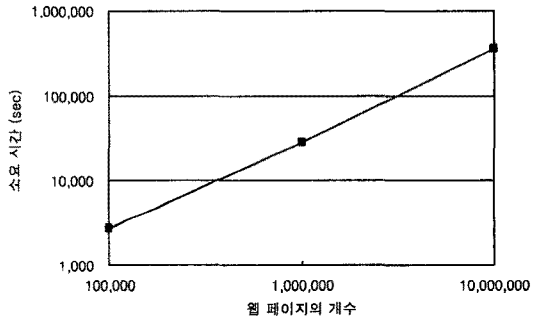

그림 15 머신 개수에 따른 소요 시간(수집할 웹 페이지의 개수 = 1,000,000)

에 제약이 있기 때문이다.

제안한 병렬 웹 크롤러와 제2.1.2절에서 소개한 병렬 웹 크롤러인 Multiple Site Crawler간에는 기능상의 차이만 있고 성능상의 차이는 없다. 제안한 시스템에서 Multiple Site Crawler에 비해 추가된 사이트 URL 추출 과정이 웹 페이지 수집 과정과 비동기적으로 처리될 뿐만 아니라 사이트 URL 확장을 위해 조정자와 대리자 간에 주고 받는 메시지(즉, URL)의 크기도 매우 작다. 그러므로 웹 페이지 수집 성능에 있어서 이 두 시스템들 간의 차이는 거의 없다고 볼 수 있다.

표 8은 머신 개수가 9대일 때, 웹 페이지의 개수에 따라 크롤링 단계, 컨버팅 단계, 랭킹 단계에서 소요된 시간을 나타낸다. 전체 과정에서 소요된 시간 중 크롤링 단계에 소요된 시간은 100,000건 웹 페이지일 경우 92%, 1,000,000건일 경우 90%, 10,000,000건 일 경우

표 8 크롤링 단계, 컨버팅 단계, 랭킹 단계에 소요된 시간(sec)(머신 개수 = 9)

웹 페이지 개수	크롤링 단계	컨버팅 단계	랭킹 단계	전체 단계
100,000	2,750	222	4	2,977
1,000,000	28,710	3,037	33	31,780
10,000,000	356,185	32,000	267	388,452

91%에 달했다. 즉, 크롤링 단계, 컨버팅 단계, 랭킹 단계를 수행하는 데에 소요된 시간 중 크롤링 단계에 소요된 시간의 비율이 가장 높음을 알 수 있다. 따라서 제한된 시간 내에 대량의 웹 크롤링을 수행할 경우 크롤링 모듈의 성능을 향상시키는 방법을 가장 먼저 고려해야 한다.

6. 결 론

본 논문에서는 여러 대의 머신을 사용하여 대량의 웹 문서를 빠르고, 효율적으로 수집하기 위한 병렬 웹 크롤러의 아키텍처를 제시하였다. 병렬 웹 크롤러는 다수의 머신을 효율적으로 관리하기 위해 조정자/대리자 구조의 2-티어 모델을 사용한다. 조정자/대리자 모델은 다수의 머신에서 웹 페이지를 수집하고 새로운 사이트 URL들을 추출하기 위한 다수의 대리자들과 이 대리자들이 수집할 사이트 URL들을 관리하기 위한 하나의 조정자로 구성된다. 병렬 웹 크롤러는 웹 페이지를 수집하기 위한 크롤링 모듈, 수집한 웹 페이지들을 데이타베이스 로딩 포맷으로 변환하기 위한 컨버팅 모듈, 수집된 웹 페이지의 중요도를 계산하기 위한 랭킹 모듈로 구성된다. 본 논문에서는 병렬 웹 크롤러의 이들 모듈을 설명하고, 세부 구현 방법을 설명하였다.

실제 인터넷 환경에서 수집할 웹 페이지의 개수와 머신 개수를 변화시키면서 병렬 웹 크롤러의 성능 평가를 수행하였다. 실험 결과 웹 페이지 수집에 소요된 시간이 수집할 웹 페이지의 개수가 증가함에 따라 점점 증가하고, 머신 개수가 증가함에 따라 감소함을 보였다. 이러한 결과로부터 제안한 병렬 웹 크롤러가 수집할 웹 페이지 개수 및 머신 개수에 대해 확장 가능함을 알 수 있었다.

현재 대부분의 상용 검색 엔진들이 병렬 웹 크롤러를 사용하는 것으로 알려져 있으나 상세한 아키텍처 및 구현 방법이 발표되어 있지 않기 때문에, 실제로 대규모 검색 엔진을 위한 병렬 웹 크롤러를 구현하기에는 어려움이 있다. 본 논문은 이러한 병렬 웹 크롤러의 아키텍처와 세부 구현 방법을 제시하였다는 점에서 큰 의미가 있다고 할 수 있다.

참 고 문 헌

[1] Dong, S., Lu, X., and Zhang, L., "A Parallel Crawling Schema Using Dynamic Partition," In *Proc. Int'l Conf. on Computational Science*, pp. 287-294, 2004.

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

[3] Google, http://www.google.com

[4] Naver, http://www.naver.com

[5] Yahoo, http://www.yahoo.com

[6] Cho, J., Garcia-Molina, H., "Parallel Crawlers," Technical Report, Stanford University, 2001.

[7] Cho, J., Garcia-Molina, H., Haveliwala, T., Lam, W., Paepcke, A., Raghavan, S., and Wesley, G., Stanford WebBase Components and Applications, Technical Report, Stanford University, 2004.

[8] 김계정, 김민수, 김이른, 황규영, "커뮤니티 제한 검색을 위한 웹 크롤링 및 PageRank 계산," *한국정보과학회 한국컴퓨터종합학술대회 논문집*, Vol.32, No.1(B), pp. 1-3, 2005년 7월.

[9] Chau, H., Pandit, S., Wang, S., and Faloutsos, C., "Parallel Crawling for Online Social Networks," In *Proc. 16th Int'l Conf. on World Wide Web*, pp. 1283-1284, Alberta, Canada, May 2007.

[10] Internet Archive, http://www.archive.org

[11] 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.

[12] 신은정, "대용량 검색 엔진을 위한 병렬 웹 크롤러의 설계 및 구현," 석사학위논문, KAIST 전산학과, 2007.

[13] Page, L., Brin, S., Motwani, R., and Winograd, T., The PageRank Citation Ranking: Bringing Order to the Web, Technical Report, Stanford University, 1998.

[14] Kamvar, S., Haveliwala, T., Manning, C., and Golub, G., Exploiting the Block Structure of the Web for Computing PageRank, Technical Report, Stanford University, 2003.

[15] Jie Xu, Qinglan Li, Huiming Qu, and Alexandros Labrinidis, "Towards a Content-Provider-Friendly Web Page Crawler," In *Proc. 10th Int'l ACM Workshop on the Web and Databases*, 2007.

[16] Koster, M., "A Standard for Robot Exclusion," http://www.robotstxt.org/wc/-norobots.html

[17] Secure Hash Standard, http://www.itl.nist.gov/fipspubs/fip180-1.htm

[18] 황규영, 이민재, 이재길, 김민수, 한욱신, "오디세우스/IR: 정보 검색 기능과 밀결합된 고성능 객체 관계형 DBMS", *한국정보과학회 논문지: 컴퓨팅의 실제*, Vol. 11, No.3, pp. 209-215, 2005년 6월.

신 은 정

2001년 3월~2005년 2월 이화여자대학교 컴퓨터학전공 학사. 2005년 3월~2007년 2월 KAIST 전자전산학과 전산학전공 석사. 2007년 3월~현재 LG전자기술원 주임연구원. 관심분야는 정보 검색, 임베디드 시스템

김 이 른
정보과학회논문지 : 컴퓨팅의 실제 및 레터
제 14 권 제 4 호 참조

허 준 석
정보과학회논문지 : 컴퓨팅의 실제 및 레터
제 14 권 제 4 호 참조

황 규 영
정보과학회논문지 : 컴퓨팅의 실제 및 레터
제 14 권 제 4 호 참조