# 쓰레드를 이용한 파이썬 웹 크롤러 구현

양희찬*, 김정길*
*남서울대학교 컴퓨터소프트웨어학과
e-mail: cgkim@nsu.ac.kr

# An Implementation of Python Web Crawler Using Thread

Hee-Chan Yang*, Cheong Ghil Kim*
*Dept. of Computer Science, Namseoul University

요 약

최근 ICT의 발전으로 인해 웹사이트 수가 폭발적으로 증가하고 있으며 인터넷상에는 여러 가지 다양하고 방대한 데이터들이 존재한다. 이러한 데이터를 효과적으로 검색하고 이용하기 위하여 웹 검색 기능은 주요한 역할을 담당한다. 본 논문에서는 웹 데이터를 자동으로 스크랩하여 파일로 저장하는 웹 크롤러를 구현하고 처리 속도 향상을 위하여 쓰레드 기능을 추가하였다. 웹 크롤러 구현을 위한 프로그래밍 언어는 파이썬을 사용하였으며 다양한 크기의 레퍼런스 사이트의 크롤링 속도 비교는 쓰레드 수를 변경함으로 측정하여 성능향상을 확인하였다.

## 1. 서론

최근 ICT의 발전으로 인해 웹사이트 수가 폭발적으로 증가함에 따라서 인터넷상에는 예전보다 더 많은 데이터가 존재한다[1]. 이러한 데이터들을 수집하여 활용하기 위해서는 문서화 또는 파일로 저장하는 작업이 필요하다. 그 과정에서 사용되는 것을 크롤러(Crawler)라고 하며 이는 웹상에 존재하는 많은 수의 웹페이지를 방문하며 각종 정보를 자동으로 수집하는 기능을 수행한다[2].

현재 방대한 데이터가 동적으로 웹에서 생성 수정되는 환경에서 효율적 검색을 위하여 다양한 목적의 웹 크롤러(Web Crawler)가 존재하고 있다[3].

본 논문에서는 오픈소스 기반의 순차적 접근 웹 크롤러를 구현하였으며, 처리 속도의 성능향상을 위하여 멀티 쓰레드 기능을 추가하였다. 성능 검증을 위하여 다른 데이터 크기의 3곳 여행 사이트에 대한 크롤링 성능을 각각의 쓰레드 수에 따라 측정하였다.

## 2. 관련 연구

### 2.1 웹 크롤러

웹 크롤러는 인터넷상에 존재하는 웹 문서들을 추적하여 필요한 정보를 수집하는 기술을 말하며, 구글과 같은 인터넷 검색시스템, 전자상거래 상품검색 등 대부분의 인터넷 산업의 근간이 되는 핵심기술이다[4].

웹 크롤러가 하는 작업을 웹 크롤링(Web Crawling) 혹은 스파이더링(Spidering)이라 부른다. 검색 엔진과 같은 여러 사이트에서는 데이터의 최신 상태 유지를 위해 웹 크롤링한다. 웹 크롤러는 대체로 방문한 사이트의 모든 페이지의 복사본을 생성하며, 검색 엔진은 이렇게 생성된 페이지를 더욱 빠른 검색을 위해 인덱싱한다. 또한, 크롤러는 링크 체크나 HTML 코드 검증과 같은 웹사이트의 자동 유지 관리 작업을 위해 사용되기도 하며, 자동 이메일 수집과 같은 웹 페이지의 특정 형태의 정보를 수집하는 데도 사용된다. 일반적으로 웹 크롤러는 시드(seed)라고 불리는 URL 리스트에서부터 시작하는데, 페이지의 모든 하이퍼링크를 인식하여 URL 리스트를 갱신한다. 갱신된 URL 리스트는 재귀적으로 다시 방문한다[5].

### 2.2 검색 엔진 Bot

웹 크롤러를 이용하는 시스템은 검색 엔진 크롤러가 있다. 대표적으로 Google의 Googlebot[6] 그리고 Bing의 Bingbot[7]이 있다. Googlebot은 Google에서 사용하는 웹 크롤러 소프트웨어이며, Bingbot은 Microsoft에서 Bing을 공급하기 위해 배포 한 웹 크롤링 시스템이다. 각각의 Bot은 웹에서 문서를 수집하여 검색 엔진에 대한 검색 가능한 색인을 작성한다.

## 3. 웹 크롤러 및 시스템 설계

### 3.1 크롤러 구조

(그림 1) 웹크롤링의 기본 구성도

위의 (그림 1)은 웹크롤링시 큐를 이용한 스케줄 및 웹 크롤링 방법을 보여준다. 웹크롤러는 데이터를 얻을 웹 사이트의 메인 주소(Seed URL)를 얻어, 일반적으로 FIFO(first input first output) 큐로 구현된다. 하나의 URL에 대한 페이지를 가져오면 그 안에 또 다른 웹 링크가 포함될 수 있으며, 웹크롤러는 이들 페이지도 함께 가져와야 한다. 필요한 경우 웹 링크를 따라가는 깊이를 제한할 수도 있다[8].

### 3.2 크롤러 모듈

파이썬에서 웹 데이터 처리 모듈은 대표적으로 re, Beautifulsoup, 그리고 lxml이 있다[9-11]. 각 모듈은 속도, 사용법에서 차이점이 있다. 실행 속도는 re 모듈이 가장 빠르고 그다음이 lxml, Beautifulsoup 순이다. re 모듈은 정규식을 이용하고 모듈이 C언어로 작성되어 속도가 가장 빠르지만, 정규식을 이해를 수반하기 때문에 사용이 복잡하다. lxml 모듈 또한 내부가 C언어로 구현되어 속도가 빠르며. CSS 선택자 또는 HtmlTag를 통해 데이터를 참조할 수 있어 사용이 편리하다. Beautifulsoup 모듈 또한 HtmlTag를 이용할 수 있고, find() 함수를 제공하여 사용에는 편리하지만, 모듈 내부가 Python으로 작성되어 있어 lxml 모듈이나 re 모듈과 비교하여 상대적으로 속도가 느린 단점이 있다[12].

본 논문에서는 위 세 가지 모듈 중에서 lxml 모듈을 중점적으로 이용하여 웹 크롤러를 구현한다.

### 3.3 콜백함수, 데이터 저장

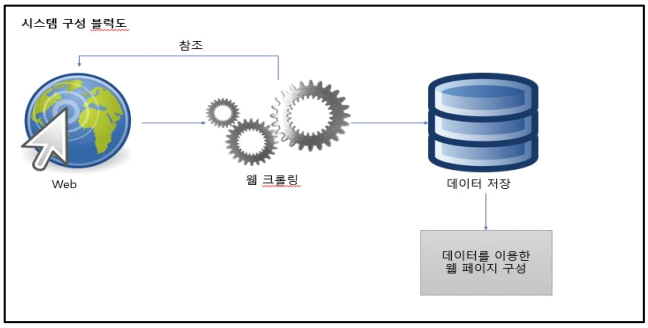
웹페이지에서 URL을 얻어 웹페이지를 다운로드할 때, 필요한 데이터가 있는 웹페이지에서는 그 데이터를 파일 또는 DB에 저장한다. 정보를 제공하는 페이지의 html 구조를 분석한 후, 필요한 데이터를 파일로 저장하는 Callback Function을 만들어서 조건에 맞는 페이지를 탐색할 때 호출하여 해당 페이지의 데이터를 파일로 출력한다.

### 3.4 멀티 쓰레드

멀티 쓰레드는 한 프로세스에서 동시에 두 가지 이상의 루틴을 실행할 수 있게 함으로 다수의 작업이 한 한 프로세스 내에서 병렬 처리가 가능하다. 웹 크롤러에서 멀티 쓰레드를 이용해 병렬 처리를 하면 여러 웹페이지의 데이터들을 동시에 다운로드하게 되어 처리시간을 줄일 수 있다. 웹 크롤러의 멀티 쓰레드 기능은 파이썬의 threading 모듈을 이용하여 구현한다[13].

## 4. 시스템 구현 및 결과

### 4.1 시스템 구조




(그림 2) 시스템 구조 블록도

시스템 구조는 (그림 2)와 같이 앞의 웹 크롤러를 통해 웹 페이지로부터 데이터를 수신하여 파일 또는 DB에 저장하고, 그 데이터들을 이용해 사용자의 목적에 맞게 이용한다.

### 4.2 멀티쓰레드

```
threads = []
while threads or crawl_queue:
    for thread in threads:
        if not thread.is_alive():
            threads.remove(thread)
    while len(threads) < max_threads and crawl_queue:
        thread = threading.Thread(target=crawlqueue)
        thread.setDaemon(True)
        thread.start()
        threads.append(thread)
    time.sleep(SLEEP_TIME)
```

(그림 3) Thread clawer

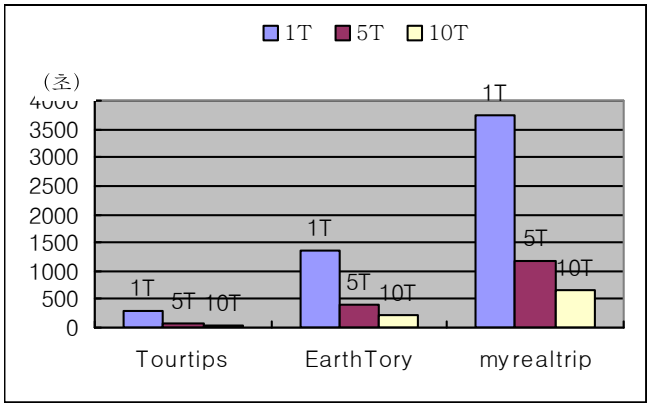
여행데이터를 가지고 있는 규모가 다른 레퍼런스 사이트를 선정하여 각각 쓰레드 수를 적용한 멀티쓰레드 크롤러의 실행 속도를 비교하였다. (그림 3)은 멀티쓰레드의 구현 소스 부분을 보여준다. <표 1>은 테스트 환경을 보이며 <표 2>는 레퍼런스 사이트에서의 실험 결과를 보여준다. 3개의 사이트는 Tourtips, EarthTory, myrealtrip를 이용하였다.

| Item | Spec |
|---|---|
| CPU | AMD Rygen 5 1600 |
| Clock | 3.2GHz |
| Core | 6 |
| Thread | 12 |
| Memory | 16GB |

<표 1> 실험 환경

<표 2> 실험 결과

| 사이트 | 1T | 5T | 10T | |
|---|---|---|---|---|
| Tourtips | 301.9 | 68.04 | 38.04 | Tourtips |
| EarthTory | 1374.91 | 387.52 | 204.79 | EarthTory |
| myrealtrip | 3758.27 | 1173.17 | 643.59 | myrealtrip |




(그림 4) 실험 결과

(T- thread, 단위- 초)

위의 (그림 4)에서 테스트 결과를 보면, 테스트한 3개 레퍼런스 사이트 모두 웹 크롤러의 사용 쓰레드 수가 증가할수록 크롤링 처리 속도가 향상되었으며 데이터 크기가 큰 경우 상대적으로 높은 성능향상의 결과를 보였다.

## 5. 결론

본 논문에서는 오픈소스 기반의 순차적 접근 웹 크롤러를 구현하였으며, 처리 속도의 성능향상을 위하여 멀티 쓰레드 기능을 추가하였다. 성능 검증을 위하여 다른 데이터 치수의 3곳 여행 사이트에 대한 크롤링 성능을 각각의 쓰레드 수에 따라 측정하여 성능 결과를 비교하여 성능향상을 확인하였다.

## 참고문헌

[1] ENISA Threat Landscape Report 2016, Available at: https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2016, 2019, Aug. 29.

[2] 강병일, 황지영, “소셜 네트워크 크롤러 개발을 통한 클러스터링 기반 데이터 통계분석“, 한국컴퓨터종합학술대회 논문집, 2018

[3] Monika Yadav, Neha Goyal, "Comparison of Open Source Crawlers- A Review", International Journal of Scientific & Engineering Research, Volume 6, Issue 9, pp 1544-1551, September 2015

[4] 김성진, “웹 정보탐색행위 모형의 비교분석 연구”, 정보처리학회지, 제21권, 제2호, pp. 211-233, 2004.

[5] Wikipedia, 웹 크롤러 Available at: https://ko.wikipedia.org/wiki/%EC%9B%B9_%ED%81%AC%EB%A1%A4%EB%9F%AC

[6] Google Bot. Available at: https://en.wikipedia.org/wiki/Googlebot

[7] Bing Bot, Available at: https://en.wikipedia.org/wiki/Bingbot

[8] 조완섭, 이정은, 최치환, “웹 크롤러의 수집주기 최적화”, 한국콘텐츠학회 논문지, vol.13, no.6, pp. 30-39, 2013

[9] lxml document Available at: https://lxml.de/index.html

[10] Python module re Available at: https://docs.python.org/3/library/re.html

[11] BeautifulSoup document Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[12] Richard Lawson, "Web Scraping with Python", Packt Publishing, Birmingham, England, 2015

[13] Python module threading Available at: https://docs.python.org/3/library/threading.html?highlight=threading#module-threading