동시성으로 작성하는 파이썬 크롤러

김남규* · 강영진** · 이훈재***

*동서대학교 컴퓨터공학부

**동서대학원 유비쿼터스 IT

***동서대학교 컴퓨터공학부

Modern Concurrent Programming for Multicode Environment

Nam-gue Kim* · Young-Jin Kang** · HoonJae Lee***

*Dept. of Information and Communication Engineering, Dongseo University

**Dept. of Ubiquitous IT Graduate School of Dongseo University

***Dept. of Computer Engineering, Dongseo University

E-mail : knq1130@naver.com, rkddudwls55@gmail.com, hjlee@dongseo.ac.kr

요 약

동시성을 보장하는 프로그래밍은 개발자에게 있어서 필수적이다. 이를 사용하지 않는다면 하드웨어 자체의 기술 발전이 있지 않는 한 프로그램의 속도 향상을 기대하기 힘들다. 뛰어난 동시성 코드를 지원하는 프로그래밍 언어로 go, elixir, scala 등이 있다. 수많은 유용한 라이브러리를 지원하는 파이썬 역시 asyncio나 coroutine과 같은 동시성 프로그래밍을 지원하고 있다. 본 지에서는 동시성과 병렬성의 개념을 정의하며, 파이썬에서 동시성 프로그래밍을 작성할 시에 유의해야 할 점에 대해 설명한다. 웹 데이터를 수집하는 크롤러를 동시성 코드로 작성하여 순차, 멀티스레딩 코드로 작성된 프로그램과 성능을 비교한다.

ABSTRACT

Programming that ensures concurrency is essential for developers. If you do not use it, it is hard to expect the speed of the program to improve unless there is technical advancement of the hardware itself. Programming languages that support good concurrency code include go, elixir, and scala. Python, which supports a number of useful libraries, also supports concurrent programming like asyncio and coroutine. This paper defines the concepts of concurrency and parallelism, and explains what to note when writing concurrency programming in Python. The crawler that collects web data is written in concurrent code and compared with programs written in sequential, multithreaded code.

키워드

asyncio, crawler, multithreading, functional programming

Ⅰ. 서 론

순차적인 기능만을 지원하는 기법을 사용하여 코드를 작성하는 것은 한계점이 뚜렷하다. 단순히 계산해도 4개의 일을 순차적으로 처리하는 것과 동시 혹은 병행 처리하는 것은 4배의 성능 차이를 보인다. 멀티스레딩은 이에 대한 해결책으

로 제시되지만 몇 가지 문제점이 존재한다. 객체의 변경 가능성으로 인한 동기화 문제, 처리 시의 사이드 이펙트가 그것들이다. 함수형 프로그래밍을 기반한 동시성 프로그래밍은 비동기를 지원하여 동시성을 보장하며, 불변하는 자료형을 제공하고 루틴에 진입하면 중간에 침범을 불허하여 사이드 이펙트를 제거한다. 본 지에서는 동시성 프로그래밍을 지원하는 언어 중 하나인 파이썬을 사용하여 동시성을 보장하는 크롤러 개발에 대해 기술한다. 2장에서는 동시성과 병렬성의 개념 정리 및 파이썬 언어의 동시성에 대하여 설명하며, 3장에서는 웹 데이터를 수집하는 크롤러를 순차, 멀티스레딩, 동시성 기법으로 작성하고 성능을 비교한다.

II. 파이썬 동시성 프로그래밍

2장에선 동시성과 병렬성의 차이점에 대해 설명한다. 파이썬 언어에서 동시성을 보장하기 위하여 고려해야하는 사항에 대해 기술한다.

2.1 동시성과 병렬성

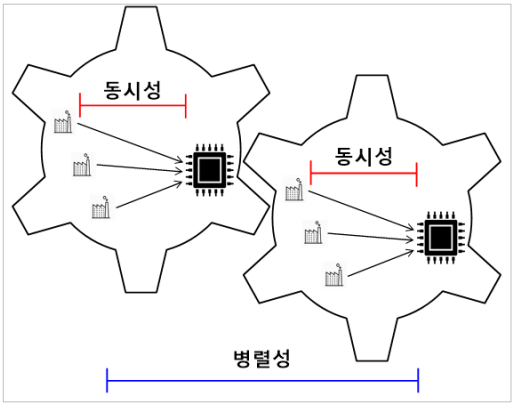

그림 1. 동시성과 병렬성

동시성과 병렬성은 중첩된 의미를 내포한다. 100개의 일을 처리한다고 가정할 때, 하나의 코어에 한정할 경우 동시적으로 처리할 것이다. 100개의 일들이 있는데 하나의 프로세스에서 순차적으로 1개의 일만 처리하는 코어는 없을 것이다. 100개의 일을 4개의 코어에서 나눠서 25개씩 처리할 경우 동시성과 병렬성을 모두 가질 수 있다. 만일 4개의 일을 4개의 코어가 1개씩 나눠서 처리한다면 병렬성만을 보장할 것이다.

즉, 단일 코어에서 여러 일들을 비동기적으로 처리하는 것이 동시성, 여러 코어를 각각 사용하여 병행 동작하는 것을 병렬성이라 명명한다.

2.2 GIL(Global Interpreter Lock)과 I/O

파이썬에서는 전역으로 스레드를 동기화시키는 GIL(Global Interperter Lock)을 지원한다. 다중 스레드를 사용할 때 전역 락이 걸려서 동시에 한 스레드만 실행한다. 동기화는 다중 코어를 사용할 때 프로그램 속도에 있어 치명적이다. 다중 스레드가 순차 실행을 한다면 다중으로 스레드를 사용하는 의미가 없다.

다만, GIL은 본 지의 목적인 크롤러를 만드는 작업과 연관이 적다. cpu 사용이 적고 cpu 연산과 관계없이 동작하는 입출력 연산이 주이기 때문이다. 만일 파이썬에서 다중 코어를 사용한 cpu 연산이 목적인 경우 다중 프로세스를 지원하는 multiprocessing 모듈을 사용해야 정상 속도를 낼 수 있다.

2.3 Asyncio와 비동기

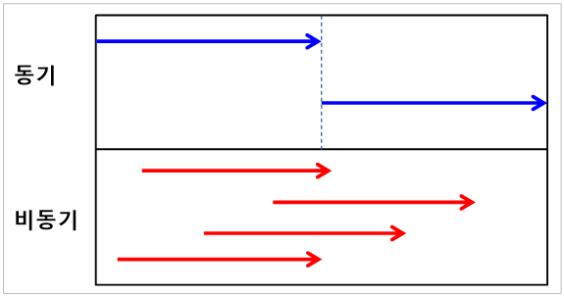

그림 2. 동기와 비동기

동기 프로그래밍은 프로그램 속도에 있어서 암적인 존재다. 빠르고 유효한 프로그램의 구현을 위해서는 비동기 프로그래밍으로 동시성을 보장해야한다.

비동기 프로그래밍은 실행 대상이 실행되고 있어도 블로킹하지 않고 다른 실행 단위에서도 실행이 가능하게 하며, 실행의 종료도 불규칙하다.

파이썬에서 Asyncio는 비동기 프로그래밍을 높은 수준에서 지원한다.

III. 동시성으로 작성하는 크롤러

3장에서는 크롤러를 Asyncio로 작성하는 과정을 소개한다. Asyncio로 작성된 프로그램과 순차, 멀티스레딩 방법으로 작성된 프로그램의 성능을 비교한다.

3.1 Asyncio를 활용한 크롤러

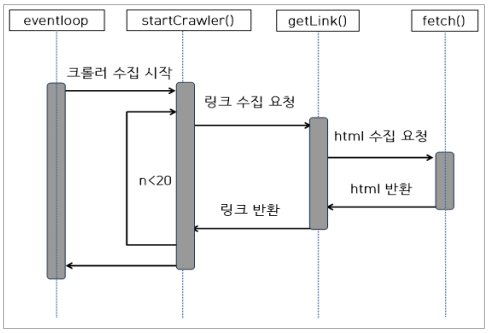

그림 3. 크롤러 시퀀스 다이어그램

<그림 3>는 동시적으로 실행할 일에서 실행될 기능의 순차적 다이어그램이다. eventloop에서 일들을 생성하면 각각의 일 단위에서 웹 데이터 수집이 비동기적으로 이뤄진다.

```
@asyncio.coroutine
def startCrawler(articleUrl):
    links = yield from getLinks(articleUrl)
    for _ in range(20):
        newArticle = links[random.randint(0, len(links) - 1)].attrs["href"]
        links = yield from getLinks(newArticle)
```

그림 4. asyncio 사용

대상 루틴은 비동기를 지원하는 문법을 사용해야 한다. 데코레이터로 '@asyncio.coroutine'과 비동기를 지원하는 대상 앞에 'yield from' 구문을 삽입한다. <그림 4>에서는 getLinks() 함수에 yield from을 넣었는데, 수 안에 있는 코드가 비동기를 지원해야 동시성을 보장할 수 있다.

```
async def getLinks(articleUrl):
    async with aiohttp.ClientSession() as client:
        html = await fetch(client, articleUrl)
        bs_obj = BeautifulSoup(html, "html.parser")
        print(bs_obj.title.get_text())
        return bs_obj.find().findAll("a", href=re.compile("^(/wiki/)((?!:).)*$"))
```

그림 5. 비동기를 지원하는 aiohttp

getLinks 함수 안에서 비동기를 지원하는 aiohttp 모듈을 사용하여 웹의 데이터를 요청한다. 기존의 requests나 urllib 같이 비동기를 지원하지 않는 라이브러리의 사용을 주의해야 한다.

3.2 순차, 멀티스레딩, 동시성 성능 비교

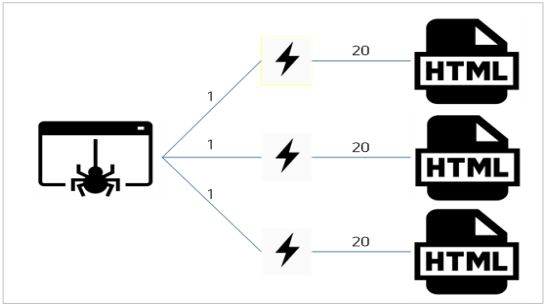

그림 6. 개발할 크롤러 기능도

<그림 6>과 같이 3개의 루트를 지정하고 20개의 웹페이지에서 데이터를 수집한다. 크롤러는 루트 목적지는 지정할 수 있지만 그 이후에는 어떤 사이트로 이동할지 예상할 수 없다. 그 결과 크롤러가 이동한 웹 페이지의 크기, 하이퍼링크의 수에 따라서 성능 차이를 가져올 수 있다.

표 1. 기능별 성능 측정

	방법	평균 시간(n/5)
sequence	순차적 실행	timecheck(5) dt=764.727s -> 152.9454s
multithreading	다중 스레드 생성	timecheck(5) dt=249.262s -> 49.8524s
asyncio	비동기 처리	timecheck(5) dt=228.397s -> 45.6794s

따라서 <표 1>에서는 방법별로 실행을 5번 반복한 후 평균을 내어 성능을 비교한다. 순차 실행은 3배 정도, asyncio는 멀티스레딩보다 약간 빠른 성능을 냄을 확인할 수 있다.

표 2. 특징 비교

	스레드	속도	사이드 이펙트
sequence	x	느림	x
multithreading	n	빠름	발생
asyncio	1	빠름	없음

n=task의 갯수

<표 2>는 각 방법별 특징을 비교한다. multithreading은 일의 숫자에 따라 스레드를 생성한다. asyncio는 일들을 비동기적으로 단일 스레드에서 처리한다. 따라서 multithreading은 여러 스레드의 동시적인 객체 참조로 인하여 블로킹 이슈가 발생할 수 있다. 반면 단일 스레드의 비동기를 지원하는 asyncio는 정상적인 경우 동시적 객체 참조로 인한 블로킹 이슈가 없으며 그로 인한 사이드 이펙트도 없다. 물론 동시성 프로그래밍이 함수형 프로그래밍 특성을 가지지만 파이썬은 불변 객체만을 데이터 형식으로 지원하지 않음으로 유의할 필요가 있다.

3.3 파이썬 프로그램 성능 측정 방법

파이썬에서 함수는 일급 객체(first object class)로 다양한 기능을 제공한다. 함수 안에서 함수 정의, 함수의 인자로 함수 사용, 반환 값으로 함수를 사용, 변수로 함수를 사용할 수 있다.

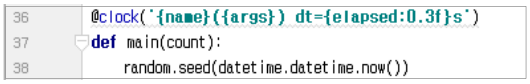

그림 7. 성능 측정 방법

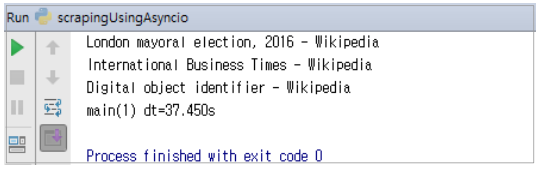

그림 8. 성능 출력 결과

3.1절에서 3가지 방법으로 작성된 프로그램의 성능을 측정하기 위해서 데커레이터를 사용했다. 함수에 데커레이터로 기능을 추가하고 싶은 경우 '@' 키워드를 붙여서 대상 함수 위에 작성해 사용할 수 있다.

<그림 4>의 코드에서는 clock 함수를 데커레이터로 사용했으며 <그림 5>와 같이 함수 실행 종료 시에 대상 함수에서 걸린 시간을 출력하게 작성했다.

Ⅳ. 결론

주어진 작업을 효율적으로 처리할 것인지는 프로그래머의 숙제다. 웹 사이트의 데이터들을 수집하여 파싱하는 작업은 상당한 시간이 걸리는 작업으로 수행 시간을 줄이는데 스레딩, 순차, 동시성과 같은 선택지가 있다. 2장에서는 동시성과 병행성은 중첩되는 의미를 내포하고 있고, 파이썬에서 지원하는 비동기 프로그래밍으로 동시성을 높은 수준에서 구현할 수 있음을 소개하였다. 3장에서는 파이썬에서 지원하는 동시성 기법을 활용하여 웹 데이터를 수집하는 크롤러의 기능을 개선하고 순차, 멀티스레딩 방식을 사용한 프로그램과의 성능을 비교하였다. 덧붙여 성능 측정 시 사용했던 clock 데코레이터 방법에 대해 소개하였다. 기존에 해왔던 방식으로 순차 또는 멀티스레딩을 사용한다면 동시성에서 얻을 수 있는 효율을 얻지 못한다. 본 지에서는 파이썬의 비동기 기법을 소개함과 더불어 비동기 기법을 사용하여 수행하는 프로젝트에서 사용하던 크롤러의 성능을 개선하였다.

감사의 글

이 논문은 2016년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(과제번호: NRF-2016R1D1A1B01011908).

또한 부산광역시에서 지원하는 BB21 과제에서 지원받았음.

참고문헌

[1] http://shop.oreilly.com/product/0636920032519.do : fluent python

[2] http://www.hanbit.co.kr/store/books/look.php?p_code=B7159663510 : 파이썬으로 웹 크롤러 만들기

[3] http://pyrasis.com/go.html : 가장 빨리 만나는 Go 언어