비순환 방향 그래프를 이용한 파이썬 스케줄러의 순환 탐지

신민욱[1], 강승식[2]
[1]국민대학교 소프트웨어학부 학부생
[2]국민대학교 인공지능학부 교수

minwook0106@gmail.com, sskang@kookmin.ac.kr

Detecting Cycles in Python Scheduler using Directed Acyclic Graph

Min-Wook Shin[1]
Seung-Shik Kang[2]
[1]Dept. of Computer Science, Kookmin University
[2]Dept. of Artificial Intelligence, Kookmin University

요 약

본 논문은 비순환 방향 그래프(directed acyclic graph) 로 파이썬 함수들을 호출하는 방법을 다룬다. 이전에 개발하였던 함수 실행 스케줄러에서 동일 함수 별칭으로 다시 호출하면 발생하는 순환 참조 문제를 해결하기 위한 추가적인 연구 개발을 진행한다. 함수를 실행하는 순서를 사전에 파악하여 순환을 탐지하도록 파이썬 3.9버전부터 지원하는 비순환 방향 그래프 기반 위상정렬 패키지로 각 그래프 정점에 함수를 할당하여 그래프가 하나의 시나리오로 동작할 수 있도록 한다.

1. 서론

최근 소프트웨어 설계 관점에서 시스템의 복잡한 요구 사항들을 최적화하고, 실시간으로 여러 코어에 작업을 적절히 분배하는 효율적인 처리 방법이 중요해지고 있다. 이러한 문제를 해결하기 위하여 다양한 스케줄링 기법이 개발되고 있으며, 그래프의 모든 정점들의 순서를 위반하지 않는 비순환 방향 그래프를 이용한 스케줄링 방법도 그 중 하나이다.[1, 2] 비순환 방향 그래프는 각자 정점 별로 실행 순서 간의 의존성을 구분해서 부모 자식의 실행 순서를 명확히 하며, 순환 참조를 감지하여 전체적인 과정을 안정적으로 실행한다.[3] 파이썬 코드로 함수 실행 순서를 정의하는 오프라인 스케줄러 방식을 개선하는 과정으로 서술한다.[4] 아파치 에어플로우 또는 Argo 워크플로우 등의 DAG 형태와 유사하게 의도하였다.[5] 초기에 구현한 단계에서 여러 제한 사항으로 인해 간단한 연결 리스트 수준으로 구현되어 이전에 사용된 함수를 다시 호출할 경우에 순환 참조 현상이 존재한다. 이러한 순환 문제를 해결하기 위해서 함수 스케줄러를 개선하고, 개선된 스케줄러 패키지를 실험하여 성능을 평가한다. 이를 통해 비순환 방향 그래프를 이용한 파이썬 함수들의 실행 과정에서 순환하는 부분을 사전에 탐지하여 전체적인 스케줄러 프로그램을 안정적으로 수행할 수 있도록 제시한다.

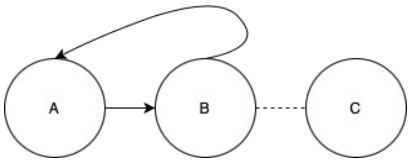
2. 함수 스케줄러 개선

기존 구현 방식은 기본적인 동작이 구현된 베이스 오퍼레이터와 함수를 단일 실행하는 오퍼레이터, 반복적으로 함수를 실행하는 오퍼레이터 등으로 구분되어 있다. 이는 반환 값을 조회하거나, 반복할 횟수를 부여할 수 있으며, 참 거짓 값에 따라서 다음 함수의 실행 여부를 제어하게 된다. 실행할 파이썬 함수를 오퍼레이터 구현 객체가 받아서 바로 함수를 실행하지 않고, DAG 객체에 함수 자체를 파라미터로 넘겨주면서 객체 내부 데이터 필드에 저장하게 된다. 그래프 정점마다 파이썬 함수를 보관한 DAG 객체는 비순환 방향 그래프 특성처럼 파이썬 함수를 부를 수 있는 별칭으로 각 정점을 연결해두어 하나의 시나리오 그래프를 구현하게 된다.

모든 파이썬 함수의 정점들을 연결하여 하나의 시나리오 그래프가 완성되면, 시작하고자 하는 그래프 정점의 함수 별칭으로 DAG 객체에 진입하여 실행하게 된다. 함수가 그래프 정점의 순서대로 순차적으로 실행하면서 함수의 반환 값을 객체에서 확인하여 다음 함수의 실행 여부를 결정하게 된다. 이러한 시나리오 그래프 구조는 연결 리스트처럼 다음에 실행할 함수를 별칭으로 지정하고 있으므로 이전에 사용된 별칭을 다시 연결하면, 의도한 다음 순서의 함수가 아닌 이전에 이미 실행한 함수를 가리키게 된다. 다음 함수로 진입하지 못하고 이전 함수로 되돌아가서 다시 동일한 동작으로 코드를 실행하는 문제가 발생한다.

스케줄러에서 여러 함수가 이어진 상태에서 이전에 사용한 파이썬 함수의 별칭을 다시 다음 함수

별칭으로 지정하게 된다면, (그림 1)과 같이 정점 A에 저장된 파이썬 함수를 부르는 별칭으로 받아들여지므로 의도한 정점 C로 넘어가지 않고 이전 정점 A로 돌아가면서 함수 A와 함수 B 구간을 별도 종료 신호가 없으면 계속 반복하여 실행한다.

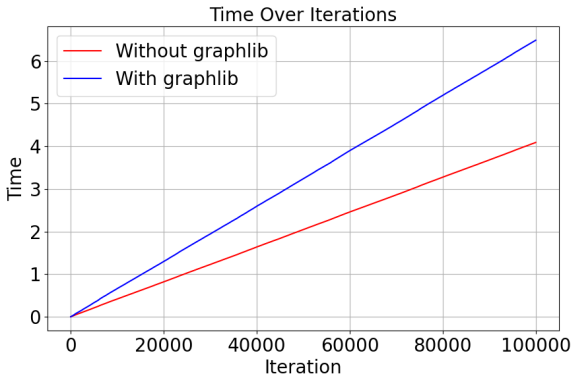


(그림 1) 순환 참조 문제를 표현한 다이어그램

순환 참조 문제를 해결하기 위하여 각 정점을 거쳐 경로를 그릴 수 있는지 직접 확인하여 오류를 검출할 수 있으나, 파이썬 3.9 버전부터 비순환 방향 그래프를 기반으로 한 위상정렬 패키지로 구현할 수 있으므로 해당 파이썬 내장 패키지를 가져와서 도입하였다. 그래프의 특성으로 인하여 함수가 들어있는 그래프 구간에 순환되는 구간이 있으면 Cycle 오류를 발생하게 된다. 기존 스케줄러 코드에서 위상정렬 객체를 도입하여 하나의 DAG 객체에 파이썬 함수를 그래프 정점으로 추가할 수 있도록 변경하고, 순환 참조에 대한 예외 처리를 사전에 포착하도록 개선하였다.

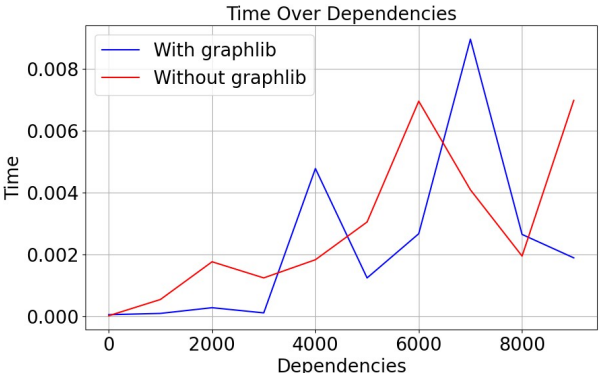
3. 실험 및 성능 평가

기존 스케줄러에서 파이썬에 내장된 위상 정렬 패키지를 도입하여 개선한 코드로 함수의 반복되는 횟수에 따른 시간 차이를 비교하였다. 사전에 지정해둔 파이썬 함수들을 그래프 정점으로 연결한 상태로 실험을 진행하였다. (그림 2)의 파이썬 함수를 기반으로 반복하는 횟수에 따른 시간을 나타낸 그래프처럼 여러 차례 진행하여 규모를 키울수록 사전 검증을 위한 그래프 순회 과정이 추가된 개선 코드가 단순히 연결 리스트로 바로 다음 파이썬 함수를 실행하는 원본 스케줄러 코드보다 느린 현상을 확인하였다.



(그림 2) 함수 반복 횟수에 따른 시간 측정 그래프

하지만, (그림 3)의 특정 파이썬 함수 정점에 무작위 연산을 수행하는 함수들을 하위 정점으로 모두 연결하여 하나의 정점이 의존하는 연결 수를 증가시킨 그래프처럼 순회해야 할 정점과 이동하는 간선의 수가 많아지면 단순 연결 리스트와 그래프 구조가 비슷한 시간으로 수렴하게 되는 현상을 확인하였다.



(그림 3) 함수 의존성의 복잡도에 따른 시간 측정 그래프

4. 결론

비순환 방향 그래프의 특성으로 각자 함수 정점별로 실행 순서 간의 의존성을 구분하고, 실행 순서를 제어할 수 있도록 파이썬 언어에 내장된 위상정렬 패키지로 순환 참조 문제를 해결하였다. 그래프 정점에 함수를 배치하는 방식으로 개선하여 적용한 함수 스케줄러 코드로 실험을 진행하였다. 비순환 방향 그래프로 순회하는 시간을 감수하는 대신에 함수 간 연결이 복잡해지며 발생할 수 있는 오류를 사전에 검증할 수 있었다. 결과적으로 하나의 비순환 방향 그래프 자체가 실행 순서를 보장하는 안정된 파이썬 스케줄러로 동작하였다.

Acknowledgment

본 연구는 2024년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음 (No. 2022-0-00964)

참고문헌

[1] N. Ueter, M. Günzel, G. v. d. Brüggen and J. -J. Chen, "Parallel Path Progression DAG Scheduling," in IEEE Transactions on Computers, vol. 72, no. 10, pp. 3002-3016, Oct. 2023, doi: 10.1109/TC.2023.3280137.

[2] 한종우, 이승수, 박성현, 이창건, "HSFS : 우월성을 이용한 이종 멀티코어 환경에서의 DAG 태스크 스케줄링 알고리즘," 한국정보과학회 학술발표논문집, 서울, pp.1593-1595, 2020.

[3] 박정식, 최상방, "코오스 그레인을 갖는 방향성 비순환 그래프의 선형 집단화," 정보과학회논문지(A), Vol.24, No.7, pp.651-666, 1997.

[4] 김영승, 조현철, 진현욱, "작업 테이블 기반의 실시간 오프라인 스케줄러," 한국정보과학회 학술발표논문집, 여수, pp.1269-1271, 2013.

[5] 김승현, 김영한, "쿠버네티스 환경에서의 작업 자동화를 위한 Argo Workflow 구조 분석," 한국통신학회 학술대회논문집, 전남, pp.910-911, 2021.