GDB 기반 CPython 디버깅 환경 분석

박신혁 [1], 이정윤 [1], 이하연 [1], 민홍 [2]
[1] 가천대학교 AI·소프트웨어학부 학부생
[2] 가천대학교 AI·소프트웨어학부 교수

quasar27@gachon.ac.kr, cgdideal0227@gachon.ac.kr, qoranreh@gachon.ac.kr, hmin@gachon.ac.kr

Analysis on CPython Debugging Environment based on GDB

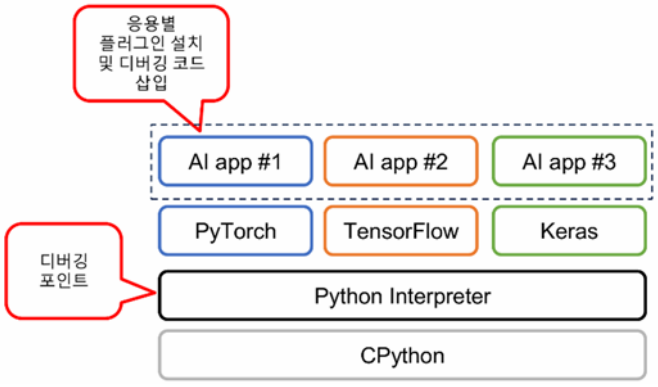
Shin-Hyeok Park[1], Jeong-Yun Lee[1], Ha-Yeon Lee[1], Hong Min[1]
[1]School of Computing, Gachon University

요 약

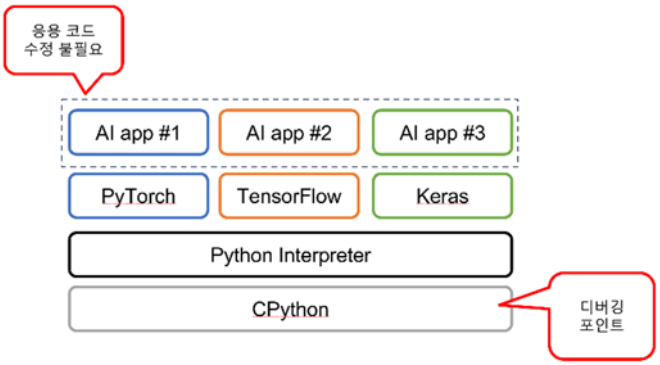
본 연구에서는 GNU Debugger(GDB)를 활용하여 CPython 내부 동작을 심층적으로 분석할 수 있는 디버깅 환경을 구축하는 방법을 제시한다. 기존의 일반적인 CPython 패키지는 최적화 옵션과 디버깅 정보의 부재로 인해 C 레벨에서의 내부 함수 호출, 실행 루프, 변수 추적 등이 제한적이었다. 이를 해결하기 위해 본 연구는 리눅스 환경에서 Mamba 및 Conda-build 를 이용하여 디버깅 정보가 포함된 CPython 을 빌드하고, 맞춤 명령어 세트를 제공하여 개발자가 효율적으로 바이트코드 실행 과정, CPython 함수 호출, 스택 상태를 추적할 수 있도록 하였다. 이러한 환경은 CPython 내부 구조의 이해를 돕고, 파이썬 인터프리터 최적화 및 확장 기능 개발에 유용하게 활용될 수 있다.

1. 서론

파이썬은 다양한 분야에서 널리 활용되는 언어이지만, 내부 동작을 이해하고 최적화하기 위해서는 C 수준에서의 디버깅이 필요하다. 이를 위해 GNU Debugger(GDB)를 활용하면 함수 호출, 변숫값, 실행 루프를 심층적으로 추적할 수 있다. 그러나 일반적인 CPython 패키지는 디버깅 정보의 부재와 최적화 옵션으로 인해 디버깅이 어렵다.



(그림 1) pdb 기반 디버깅.



(그림 2) gdb 기반 디버깅.

따라서 본 논문에서는 리눅스 환경을 기반으로 Mamba 와 Conda-build 를 활용해 체계적으로 CPython 을 빌드하고 디버깅 환경을 구축한다.

2. 요구사항 및 CPython 디버깅 환경 준비

CPython 의 빌드와 디버깅을 위해 필요한 기본적인 것을 갖추어야 한다. 기본적으로 Linux x64 시스템이 필요하다. 또한 CPython 을 빌드하고 디버깅할 가상환경을 제공해 줄 Mamba 또는 Micromamba 가 필요하다. 요구사항이 준비되면 CPython 디버깅 환경을 구축할 수 있다.

먼저, pydebug 프로젝트를 클로닝한다[1]. README.md 를 참고하여 가상환경을 활성화[2] 후 conda-build 를 설치한다.[3] conda-build 는 사용자만의 conda 패키지를 만들 수 있게 하는 도구다. 클로닝한 pydebug 디렉토리에 접근한 후 cenv/cpython 에 CPython 소스를 불러오기 위해 git 서브모듈을 가져온다[4]. CPython 을 빌드하기 위한 명령을 실행한다[5]. cenv 에는 CPython 빌드를 위한 재료가 존재하며, CPython 소스코드, 빌드 스크립트, 의존성 정보 등이 포함된다. 빌드를 하기 위해 들어갔던 가상환경을 비활성화하고[6], 새로 빌드한 파이썬이 탑재된 새로운 가상환경을 만든다[7]. 이후 새 가상환경을 활성화한다[8]. 이 가상환경에 설치된 파이썬은 디버깅 정보를 포함하는 것으로써 gdb 와 같은 C 언어 기반 디버거로 디버깅이 가능하다.

3. CPython 맞춤 GDB 확장 명령어

환경이 준비되면 pydebug 디렉토리 내에서 gdb -tui python3 명령어를 통해 GDB 를 통한 CPython 디버깅을 시작할 수 있다. 디렉토리에는 파이썬 디버깅을 도와주는 맞춤 명령어가 제공되며 이를 GDB 내로 불러올 수 있다[9]. 이 명령어들은 GDB Python API 를 사용하여 제작되었다.[1] 이후 제공되는 명령어를 통해 특정 파이썬 파일을 실행하는 CPython 을 디버깅할 수 있다[10]. 맞춤 명령어 중에는 main 함수에서 evaluation loop[11] 까지 가는 과정을 생략해주는 명령어가 있어 GDB 명령어 창에 c 를 입력 후 cc 를 입력하면 evaluation loop 에 빠르게 도달할 수 있다. 이 외에 다음과 같은 명령어가 제공되며 evaluation loop 내에서 사용가능하다.

<표 1> pydebug 에서 제공하는 GDB 확장 명령어

명령어	설명
qi [[s] e]	현재 실행하는 파이썬 바이트코드와 다음 실행될 파이썬 바이트코드를 확인한다. default [-1, 8)
qn [n]	코드를 바이트코드 단위로 진행한다. default=1
qc	qb 로 건 다음 브레이크포인트를 만날 때까지 코드를 진행한다.
qbt	CPython 함수 호출을 백트레이스한다.
qst	CPython value stack 을 확인한다.
qfin	현재 실행 중인 파이썬 함수를 완료한다.
pauto <PyObject*> pautoex <PyObject*>	특정 변수의 자료형과 값을 확인한다.
qb	현재 실행하는 바이트코드에 브레이크포인트를 건다.
pb	qb 로 건 브레이크포인트 목록을 확인한다.

4. 결론

본 연구에서는 리눅스 환경에서 CPython 을 디버깅하기 위한 체계적인 환경 구축 절차와 이를 보조하는 맞춤 GDB 명령어를 제안하였다. 제시된 방법을 통해 기존에 어려웠던 CPython 내부 실행 흐름 추적이 가능해졌으며, 함수 호출 과정, 스택 상태, 바이트코드 단위 실행을 직관적으로 분석할 수 있었다. 이는 CPython 자체를 연구하거나 최적화하려는 개발자뿐 아니라, 파이썬 기반 애플리케이션의 성능 개선 및 오류 분석에도 기여할 수 있다. 향후에는 본 환경을 바탕으로 다양한 파이썬 버전에 대한 지원 확대와 자동화된 디버깅 툴킷 개발이 필요하다. 이를 통해 파이썬 언어와 인터프리터 연구의 생산성과 효율성을 한층 높일 수 있을 것으로 기대한다.

사사문구

이 논문은 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.RS-2025-02216517,재구성형 인공지능 프로세서 SW 프레임워크 기술개발)

참고문헌

[1] Free Software Foundation, "Python API (Debugging with GDB)", GDB Documentation, https://sourceware.org/gdb/current/onlinedocs/gdb.html/Python-API.html#Python-API, 2025, 09, 23

[1] git clone https://github.com/QuasarIIVII/pydebug
[2] mamba activate base
[3] mamba install conda-build -y
[4] git submodule update --init --recursive
[5] conda build cenv
[6] mamba deactivate
[7] mamba create –n <env_name> python=3.12=o0_0_cpython –c local –y
[8] mamba activate <env_name>
[9] source .gdbinit
[10] qinit <python file name>
[11] 파이썬 바이트코드를 실제로 실행하는 부분