УДК 004.43

Д. В. Дейбук

THE FUTURE OF THE PYTHON PROGRAMMING LANGUAGE WITHOUT GIL

Vinnytsia National Technical University

Анотація

У цій статті описано поточний стан конкурентності в Python, з наголосом на обмеження, накладені Глобальним блокуванням інтерпретатора (GIL). Вона досліджує різні моделі конкурентності, такі як threading та multiprocessing, а також бібліотеки, такі як AsyncIO. Стаття висвітлює намагання подолати обмеження GIL та потенціал майбутнього Python без GIL для покращення продуктивності та відповідності вимогам сучасних обчислювальних середовищ.

Ключові слова: Python, конкурентність, GIL

Abstract

This article discusses the current state of concurrency in Python, emphasizing the limitations imposed by the Global Interpreter Lock (GIL). It explores various concurrency models, such as threading and multiprocessing, as well as newer additions like AsyncIO. The article highlights ongoing efforts to overcome the constraints of the GIL and the potential for a GIL-free Python future to enhance performance and meet the demands of modern computing environments.

Keywords: Python, concurrency, GIL

The future of Python, particularly concerning the Global Interpreter Lock (GIL) and its concurrency capabilities, is a subject of significant interest within the development community. This article delves into the current landscape of concurrency in Python, the implications of the GIL, and the anticipated trajectory Python might follow should it move beyond the GIL.

Concurrency in Python is facilitated through several frameworks and modules, each catering to different programming needs. Threading, for instance, allows Python to execute multiple threads concurrently. However, the GIL – a mutex that prevents simultaneous execution of Python byte codes by multiple threads – means that threading is primarily beneficial for I/O-bound tasks rather than CPU-bound tasks. For CPU-bound operations, the multiprocessing module offers a workaround by enabling parallel execution across multiple processors, sidestepping the GIL's limitations.

AsyncIO emerged as a significant addition to Python's concurrency toolkit, introducing a single-threaded, single-process model that uses coroutines for concurrent task management. This model is particularly adept at handling high I/O-bound operations, like network communications or file handling. Additionally, the concurrent futures module simplifies concurrent programming by abstracting the complexities involved in thread and process pool management, providing a more accessible interface for parallel execution.

The presence of the GIL in Python has long been a contentious issue. While it facilitates thread safety within the CPython interpreter and simplifies certain aspects of Python’s implementation, it also constrains the language’s ability to fully utilize multicore processors for parallel execution of CPU-bound tasks. This limitation has started ongoing debate and various proposals for overcoming or eliminating the GIL to unlock true parallel processing capabilities in Python.

PEP 703 proposes several changes to CPython, including modifications to reference counting, garbage collection, memory management, and the integration of deferred and biased reference counting. It also introduces a new build configuration that allows for a version of CPython without the GIL, which can be toggled on or off during compilation. [1]

The Steering Council of Python has accepted PEP 703 with a proviso that the rollout be gradual and disruptive changes be minimized. There is an understanding that if necessary, all changes related to PEP 703 could be rolled back, or even the entire proposal could be reversed. [1]

The implementation of PEP 703 is expected to be a long-term project, involving multiple stages over several years. During this time, the CPython interpreter will transition to make the no-GIL version optional, then supported, and eventually the standard version of CPython. [2]

The quest for a Python without the GIL is not merely a technical challenge; it represents a pivotal evolution in Python's concurrency model. As Python continues to dominate in fields ranging from web development to data science, enhancing its concurrency capabilities is crucial to meeting the demands of contemporary computing environments. The road to overcoming the GIL's constraints is fraught with challenges, yet it offers Python an opportunity to evolve and better serve its diverse and growing user base. This ongoing journey reflects the vibrant dynamism and innovative spirit that characterize the Python community.

REFERENCES

1. PEP 703 – Making the Global Interpreter Lock Optional in CPython | peps.python.org [Електронний ресурс] // PEP 0 – Index of Python Enhancement Proposals (PEPs) | peps.python.org. – Режим доступу: https://peps.python.org/pep-0703/ (дата звернення: 10.02.2024).
2. Python moves to remove the GIL and boost concurrency [Електронний ресурс] // InfoWorld. – Режим доступу: https://www.infoworld.com/article/3704248/python-moves-to-remove-the-gil-and-boost-concurrency.html (дата звернення: 10.02.2024).

Дейбук Денис Валерійович — студент групи 2ПІ-22б, факультет інформаційних технологій і комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: deibukdenys@gmail.com

Науковий керівник: Мельник Марина Борисівна, викладач англійської мови, кафедра іноземних мов, Вінницький національний технічний університет.

E-mail: melnykmary1@gmail.com

Deibuk Denys V. — Student of the Department of information technologies and computer engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: deibukdenys@gmail.com

Scientific supervisor Melnyk Maryna Borysivna – teacher of English, Department of the Foreign Languages, Vinnytsia National Technical University.

E-mail: melnykmary1@gmail.com