НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ
НТУУ «КПІ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ТАВРІЙСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В. І. ВЕРНАДСЬКОГО
ДЕРЖАВНА ЕКОЛОГІЧНА АКАДЕМІЯ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ОБОРОНИ УКРАЇНИ
ІМЕНІ ІВАНА ЧЕРНЯХОВСЬКОГО
НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
ТОВ «ІТЦ ХАЙ-ТЕК БЮРО»
КОМПАНІЯ «E-TRADE HUB LTD.»
МІЖНАРОДНИЙ ІНСТИТУТ ЕКОЛОГІЧНОЇ БЕЗПЕКИ
ОРГАНІЗАЦІЇ БЕЗПЕКИ СПІЛЬНОТИ ЄВРОПИ
ЦЕНТР ЕКОЛОГО-РЕСУРСНОГО ВІДНОВЛЕННЯ ДОНБАСУ
ГО «АСОЦІАЦІЯ ФАХІВЦІВ ЦИВІЛЬНОГО ЗАХИСТУ»

Четверта міжнародна
науково-практична конференція

«Сучасні тенденції розвитку інформаційних систем і телекомунікаційних технологій»

1–2 лютого 2022 р.

Київ НУХТ 2022

УДК 004.9:621.39

Наукові праці Четвертої міжнар. наук.-практ. конф. «Сучасні тенденції розвитку інформаційних систем і телекомунікаційних технологій», 1–2 лютого 2022 р. (Київ, Україна). – К. : НУХТ, 2022. – 226 с.

У працях конференції наведено доповіді за напрямами:

- світові тенденції в розробленні інформаційних систем і телекомунікаційних технологій;
- міжнародні стандарти в галузі інформаційних і телекомунікаційних технологій та кіберзахисту;
- розвиток освіти і науки в галузі інформаційних і телекомунікаційних технологій та кіберзахисту;
- інтернет речей та розвиток його технологій для безпечного суспільства;
- моделювання та симуляція стихійних лих, надзвичайних ситуацій і реагування на них;
- досвід використання інформаційних технологій, безпілотних літальних апаратів і роботів для моніторингу навколишнього середовища, попередження й ліквідації надзвичайних ситуацій природного і техногенного походження;
- неурядові та громадські організації у сфері цивільного захисту.

Праці конференції будуть корисні науковим та інженерно-технічним працівникам, студентам ЗВО та всім, хто цікавиться сучасними інформаційними системами та телекомунікаційними технологіями.

Подано в авторській редакції.

ISBN 978-83-956296-5-5

УДК 004.438:004.272.2

МОЖЛИВОСТІ PYTHON ДЛЯ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ

Костіков М. П.
Національний університет харчових технологій, Київ, Україна
Київський національний університет імені Тараса Шевченка, Київ, Україна
E-mail: MikolaszK@gmail.com

Capabilities of Python for Parallel Programming

Parallel programming is an important concept that allows us to build quick and efficient applications with responsive user interface. Python is one of the most popular programming language that implements this concept as well as many others. The report considers capabilities of Python for parallel programming and specific features of various approaches to its implementation, such as multithreading and multiprocessing.

Паралельне програмування є важливим для прискорення виконання коду, а також для реалізації багатозадачності в програмах і відгуку інтерфейсу користувача без затримок. Цю концепцію реалізовано в більшості сучасних мов програмування, зокрема і в Python, що нині є однією з найпопулярніших мов. Зокрема, згідно з опитуванням компанії HackerRank, Python або вже добре знають, або планують вивчати як наступну мову; крім того, станом на 2020 рік майже половина роботодавців шукали фахівців, які б володіли Python [1]. Розглянемо більш докладно можливості цієї мови та її засобів для написання паралельних програм.

Одними з найпоширеніших підходів до реалізації паралелізму в Python є використання бібліотек *threading*, *multiprocessing* і *asyncio*.

Бібліотека *threading* дає змогу вручну створювати, запускати і проводити всі інші стандартні операції з потоками виконання програми аналогічно до відповідних бібліотек у інших мовах програмування — зокрема, наприклад, *System.Threading* у C#, *thread* у C++ і *java.lang.Thread* у Java. Тож із допомогою *threading* можна розділити програму на незалежні частини та виконувати їх паралельно [2].

Проте слід розуміти, що паралелізм не завжди означає саме одночасне виконання частин коду. І у випадку з цією бібліотекою особливості інтерпретатора CPython не дозволяють виконувати одночасно більш, ніж один потік Python [3]. Через це програми, написані з допомогою *threading*, не зможуть прискорити свого виконання для ефективного використання наявних ядер процесора. Тож бібліотеку *threading* для Python доцільно застосовувати для досягнення інших корисних ефектів паралелізму. Зокрема це ізолювання частин програми, її структурування й написання зрозумілішого коду, реалізація багатозадачності в рамках програми, незалежного опису та виконання різних дій, прискорення відгуку інтерфейсу користувача тощо.

Натомість для підвищення швидкості виконання програми та відповідно зменшення затрат часу на багатоядерних і багатопроцесорних системах слід або

використовувати інший інтерпретатор Python, або звернутись до інших бібліотек і засобів цієї мови. Саме однією з таких бібліотек і є *multiprocessing.*

Бібліотека *multiprocessing*, як випливає з назви, дає змогу створити кілька окремих процесів для виконання певного завдання [4]. При цьому можливо створити спільну функцію для декількох процесів, яка при їхньому запуску набуває різних параметрів, аналогічно як це зазвичай робиться з потоками в рамках одного процесу. Так само процеси можуть мати доступ до консолі тощо.

Завдяки виділенню окремих процесів під поставлене завдання бібліотека *multiprocessing* дозволяє запустити їх дійсно паралельно, по можливості виконуючи їх на окремих ядрах процесора. Таким чином, ми можемо не лише скористатись усіма перевагами паралелізму, які реалізовано в *threading*, а й отримати бажаний ефект по часу виконання програми завдяки її прискоренню.

Загалом слід зазначити, що реалізація паралелізму через процеси, а не потоки, має свої особливості, які можуть бути як перевагами, так і недоліками. Зокрема під кожен новий процес виділяється окремий обсяг пам’яті, в той час як потоки можуть ділити між собою лише наявну пам’ять у межах одного процесу. Код у різних процесах є більш ізольованим, ніж у потоках, що сприяє захисту ресурсів. У той же час, багатопроцесні програми споживають більше пам’яті, ніж багатопоточні, а різні процеси не можуть використовувати спільну пам’ять, що є доступним для різних потоків у межах спільного для них процесу.

Що стосується *asyncio,* ця бібліотека спрощує паралельне програмування за рахунок можливості писати паралельний код у рамках одного потоку, що дає змогу спростити етапи написання, відлагодження та підтримки коду [5].

Бібліотека *asyncio* добре підходить для операцій введення-виведення та коду для мереж; крім того, вона є основою для фреймворків Python, які забезпечують роботу з веб-серверами, базами даних, розподіленими чергами завдань тощо [6].

Як бачимо, вищезазначені підходи дозволяють повною мірою реалізувати концепцію паралельного програмування в Python, отримуючи ефект як від розбиття коду на незалежні частини, так і від прискорення виконання програм.

Література

1. HackerRank (2020) *2020 HackerRank Developer Skills Report* [online]. URL: https://www.hackerrank.com/research/developer-skills/2020.

2. Geeks for Geeks (2022) *Multithreading in Python. Set 1* [online]. URL: https://www.geeksforgeeks.org/multithreading-python-set-1.

3. Anderson J. (2022) 'An Intro to Threading in Python' [online], *Real Python*. URL: https://realpython.com/intro-to-python-threading.

4. Udacity Team (2020) *What is Python Parallelization?* [online]. URL: https://www.udacity.com/blog/2020/04/what-is-python-parallelization.html.

5. Deitel P., Deitel H. (2019) *Python for Programmers: with Big Data and Artificial Intelligence Case Studies*. London: Pearson Education, 640 p.

6. Python Software Foundation (2022) *asyncio — Asynchronous I/O* [online]. URL: https://docs.python.org/3/library/asyncio.html.