

ВИКОРИСТАННЯ PYTHON ДЛЯ ПОБУДОВИ РОЗПОДІЛЕНИХ СИСТЕМ У СЕРЕДОВИЩІ КОНТЕЙНЕРИЗАЦІЇ

Дубан Р. М., к.т.н.

Харківський технологічний університет «ШАГ»

Україна, Харків

duban_r@itstep.academy

Анотація. У статті досліджено можливості використання Python для побудови розподілених систем у середовищі контейнеризації, зокрема з використанням асинхронного програмування та підходу Domain-Driven Design. Основні теоретичні результати демонструють ефективність Python у побудові мікро-сервісних архітектур, орієнтованих на високу продуктивність і масштабованість. Практичні експерименти підтверджують надійність інтеграції Python із контейнерами Docker та оркестраційними платформами на зразок Kubernetes.

Ключові слова: розподілені системи; контейнеризація; асинхронне програмування; мікро-сервіси.

Вступна частина. Сучасні інформаційні системи вимагають високої масштабованості, надійності та ефективності у використанні ресурсів. Розподілені системи стали основним підходом для вирішення цих завдань, оскільки вони дозволяють організовувати роботу різних компонентів, що працюють незалежно один від одного. Однак, ізолюваність та забезпечення продуктивної комунікації між компонентами є викликами для багатьох мов програмування та підходів. Контейнеризація (Docker, Kubernetes) дозволяє ізолювати кожен компонент системи, забезпечуючи гнучкість у їх розгортанні та масштабуванні.

Python став популярним інструментом для розробки таких систем через свою гнучкість, простоту у використанні та широку екосистему бібліотек. Проте, деякі підходи для асинхронної обробки запитів можуть мати обмеження щодо продуктивності та відмовостійкості, особливо у системах з великим навантаженням. Недавні дослідження показали, що ефективність можна підвищити за рахунок використання асинхронного програмування та Domain-Driven Design (DDD) для створення мікро-сервісної архітектури веб-сервісів (Seth, 2024).

Метою дослідження є дослідження переваг використання Python для побудови розподілених систем у середовищі контейнеризації з акцентом на асинхронне програмування, архітектуру DDD та мета-програмування.

Основна частина. Асинхронне програмування є ключовим інструментом для побудови розподілених систем, що потребують обробки великої кількості запитів одночасно. Python надає бібліотеки, такі як `asyncio` та `aiohttp`, що дозволяють виконувати операції паралельно без блокування основного потоку. Це забезпечує ефективність і зменшує час очікування під час обробки великих обсягів даних. Наприклад, асинхронний веб-сервіс, створений за допомогою бібліотеки `aiohttp`, здатний одночасно обробляти тисячі запитів (Hattingh, 2020).

DDD допомагає структурувати програмне забезпечення на основі предметної області. Кожен мікро-сервіс відповідає за конкретну частину системи, що забезпечує незалежність під час розробки та масштабування. Використання фреймворків, таких як FastAPI та Flask, дозволяє впроваджувати архітектуру DDD у Python з підтримкою REST API для комунікації

між мікро-сервісами. Це підвищує структурованість коду та полегшує рефакторинг (Zhong et al., 2024).

Контейнеризація стала основою для розгортання розподілених систем, дозволяючи ізолювати кожен сервіс у своєму середовищі. Використання Docker спрощує розгортання та забезпечує стабільність додатків у різних середовищах. Оркестраційні платформи, як Kubernetes, автоматизують розподіл навантаження, управління репліками та відновлення сервісів. Python добре інтегрується з Docker, що дозволяє легко керувати контейнеризованими додатками (Ayyagiri et al., 2024).

Python підтримує мета-програмування, що дозволяє динамічно створювати та модифікувати код під час виконання. Це особливо корисно для розподілених систем, де компоненти можуть змінювати свою поведінку залежно від даних або конфігурацій. За допомогою рефлексії Python дозволяє маніпулювати об'єктами та їх атрибутами у процесі виконання, що підвищує адаптивність системи (Stavytskyi & Voitko, 2022).

У ході експериментів було протестовано декілька варіантів розгортання Python-сервісів у середовищі контейнеризації з використанням Docker і Kubernetes. Асинхронне програмування з використанням бібліотеки *asyncio* та фреймворку *FastAPI* дозволило досягти значного зниження затримок під час обробки запитів, що підтверджувалося метриками часу відповіді та кількістю одночасно оброблюваних запитів. Система підтримувала роботу із сотнями одночасних запитів без суттєвого впливу на продуктивність. Впровадження мікро-сервісної архітектури DDD дозволило легко модифікувати окремі сервіси без впливу на загальну систему, що сприяло гнучкості та масштабованості. Успішна інтеграція з контейнеризаційними рішеннями дозволила розгорнути сервіси швидко й без конфліктів, а оркестрація за допомогою Kubernetes забезпечила автоматичне масштабування та стійкість до відмов.

Висновки. У результаті дослідження було показано, що Python є ефективним інструментом для побудови розподілених систем у середовищі контейнеризації. Використання асинхронного програмування підвищує продуктивність систем, а архітектура DDD дозволяє організувати мікро-сервіси таким чином, щоб забезпечити гнучкість та масштабованість. Контейнеризація і оркестрація спрощують управління системою та підвищують її надійність. Мета-програмування та рефлексія дозволяють динамічно адаптувати поведінку системи, роблячи її більш гнучкою та адаптивною до змін.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Seth, D. (2024). A Domain-Driven Design Approach for Micro-Services using Web Services. *International Journal of Science and Research (IJSR)*, 13(6), 784–790. <https://doi.org/10.21275/sr24610053316>
- [2] Hattingh, C. (2020). *Using Asyncio in Python: Understanding Python's Asynchronous Programming Features*. O'Reilly Media, Inc.
- [3] Zhong, C., Li, S., Huang, H., Liu, X., Chen, Z., Zhang, Y., & Zhang, H. (2024). Domain-Driven Design for Microservices: An Evidence-Based Investigation. *IEEE Transactions on Software Engineering*, 50(6), 1425–1449. <https://doi.org/10.1109/tse.2024.3385835>
- [4] Ayyagiri, N. A., Jain, N. D. A., & Goel, N. O. (2024). Utilizing Python for Scalable Data Processing in Cloud Environments. *Darpan International Research Analysis*, 12(2), 183–198. <https://doi.org/10.36676/dira.v12.i2.78>
- [5] Stavytskyi, P., & Voitko, V. (2022). METHOD OF THE DECLARATIVE METAPROGRAMMING BASED ON DOMAIN-SPECIFIC PROGRAMMING LANGUAGES. *Herald of Khmelnytskyi National University Technical Sciences*, 311(4), 249–255. <https://doi.org/10.31891/2307-5732-2022-311-4-249-255>

Duban R. M.

Using Python for Building Distributed Systems in a Containerized Environment

Abstract. *This paper explores the potential of using Python for building distributed systems in a containerized environment, particularly through the use of asynchronous programming and the Domain-Driven Design approach. Theoretical findings demonstrate Python's effectiveness in developing microservice architectures focused on high performance and scalability. Practical experiments confirm the reliability of Python's integration with Docker containers and orchestration platforms such as Kubernetes.*

Keywords: *distributed systems; containerization; asynchronous programming; microservices.*