УДК 004.8

К. О. Гончак
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ У СЕРЕДОВИЩІ DJANGO REST FRAMEWORK

Вінницький національний технічний університет

Анотація

Розглянуто розробку програми реалізації паралельної обробки запитів у середовищі Django REST Framework. Розглянуто питання аналізу предметної області, поставлено задачу на виконання, обґрунтовано засоби розробки програмної частини на мові Python, а також архітектурні особливості Django REST Framework. Розроблено математичну модель паралельної обробки запитів, побудовано UML-діаграми класів та активності програмного забезпечення, обґрунтовано вибір моделі реалізації паралелізму та оптимізовано алгоритми обробки API-запитів складності O(n). Створено програмну реалізацію паралельної обробки набору обчислювальних завдань різної складності та проведено тестування роботи API. Програмна реалізація дозволяє збільшити швидкодію, продуктивність та надійність обробки вхідних запитів. Одержані результати можливо використовувати у різноманітних алгоритмах та програмних засобах для підвищення ефективності взаємодії задач у веб-сервісах

Ключові слова: *паралельні обчислення, Django REST Framework, API, паралелізм, обробка запитів*.

Abstract

The development of a program for implementing parallel query processing in the Django REST Framework environment. The issues of domain analysis are considered, the task is set, the means of developing the software part in Python are justified, as well as the architectural features of the Django REST Framework. A mathematical model of parallel query processing is developed, UML diagrams of classes and software activities are constructed, the choice of a parallelism implementation model is justified, and algorithms for processing API queries of O(n) complexity are optimized. A software implementation of parallel processing of a set of computational tasks of varying complexity has been created and the API has been tested. The software implementation allows for increased speed, productivity, and reliability of processing incoming requests. The results obtained can be used in various algorithms and software tools to improve the efficiency of task interaction in web services...

Keywords: *parallel computing, Django REST Framework, API, parallelism, query processing.*

Вступ

Актуальність теми зумовлена швидким розвитком веб-технологій та потребою у створенні програмних продуктів, здатних виконувати паралельні обчислення в умовах високого навантаження. У світовій та вітчизняній практиці простежується тенденція переходу від послідовних обчислень до розподілених і паралельних моделей, що дозволяють значно підвищити продуктивність сервісів та забезпечити стабільність їх роботи. Існуючі програмні рішення часто не забезпечують достатньої ефективності при обробці великої кількості одночасних запитів або мають обмежену масштабованість, що підкреслює необхідність створення нового програмного продукту з покращеними характеристиками.

Метою роботи є розробка програмного продукту на мові Python із використанням Django REST Framework, який забезпечує ефективну паралельну обробку вхідних API-запитів, оптимізує швидкодію системи та створює можливість масштабування.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань: проаналізувати існуючі підходи до паралельної обробки запитів у веб-системах; дослідити архітектуру Django REST Framework та її можливості щодо організації паралельних обчислень; побудувати математичну модель паралельної

обробки запитів та визначити оптимальні алгоритми обчислення; реалізувати програмний модуль обробки API-запитів із використанням механізмів паралелізму; провести тестування програмної реалізації для оцінки швидкодії та коректності роботи; оцінити ефективність створеного програмного продукту та можливості його подальшого застосування.

Виклад основного матеріалу

Останні десятиліття паралельні та високопродуктивні обчислювальні системи стали ключовою складовою розвитку програмного забезпечення в більшості галузей науки, техніки та інформаційних технологій. Зростання обсягів даних, інтенсивність обчислень та поширення веб-сервісів, що обробляють тисячі одночасних запитів, визначають необхідність застосування методів паралельної обробки інформації. Паралельні обчислення забезпечують можливість суттєвого підвищення продуктивності системи та скорочення часу виконання обчислювальних завдань [1].

Важливим аспектом паралельної обробки є синхронізація доступу до спільних ресурсів, таких як база даних, кеш або спільні змінні у пам'яті. Для вирішення цих проблем використовуються різні методи синхронізації, такі як м'ютекси, семафори, монітори та атомарні операції. Сучасні мови програмування та фреймворки надають вищорівневі абстракції, які спрощують роботу з потоками та паралельними операціями, зменшуючи ймовірність помилок та підвищуючи надійність програмного забезпечення. Завдяки паралельній обробці API-запитів системи можуть ефективно працювати під високим навантаженням, забезпечувати швидкі відповіді користувачам та масштабуватися відповідно до зростаючих потреб без значного зниження продуктивності [2–3].

У роботі розроблено паралельну програмну систему, що реалізує обробку вхідних API-запитів у середовищі Django REST Framework з використанням багатопоточності та потокового поділу завдань.

Перший компонент — модуль приймання запитів — відповідає за отримання вхідних даних, їхню первинну перевірку та передачу задач у чергу для подальшої обробки.

Другий компонент — модуль попередньої обробки — виконує лінійну обробку елементів масиву: фільтрацію, нормалізацію, серіалізацію та інші підготовчі операції.

Третій компонент — модуль аналітичних обчислень — працює паралельно з використанням пулу потоків або процесів, виконуючи трудомісткі операції: перетворення рядків, статистичні обчислення, агрегацію результатів та формування підсумкової структури відповіді.

Для взаємодії між цими підсистемами застосовано механізм синхронізованих черг та виконавчих пулів (ThreadPoolExecutor/ProcessPoolExecutor), які забезпечують коректну передачу даних і виключають конфлікти при конкурентному доступі. Додатково використано атомарні змінні та внутрішні примітиви Python для контролю завершення обчислень та координації роботи потоків.

Математичне моделювання системи виконано з урахуванням потокового поділу даних. Пропускна здатність кожного паралельного модуля визначалась кількістю елементів, що обробляються за одиницю часу, а загальна продуктивність системи оцінювалася за законом Амдала. При збалансованому навантаженні досягалося майже лінійне прискорення, оскільки незалежні підзадачі рівномірно розподілялися між робочими потоками [4–6].

Для підвищення швидкодії реалізації використано механізми конкурентності Python (concurrent.futures) та можливості Django REST Framework щодо асинхронної та паралельної обробки запитів. Оптимізацію алгоритму досягнуто шляхом:

- розподілу обчислень між потоками за допомогою пулу ThreadPoolExecutor;
- зменшення кількості синхронних операцій у межах API-виклику;
- застосування локальних буферів для мінімізації доступів до спільних ресурсів;
- можливості масштабування шляхом зміни кількості потоків без зміни логіки API..

Програмну частину реалізовано мовою Python на базі Django REST Framework, що забезпечило структурованість API, простоту інтеграції паралельних обчислень та можливість масштабування системи відповідно до навантаження. Для візуалізації архітектури створено UML-діаграму компонентів та UML-діаграму активності, які наочно відображають взаємодію між модулями та логіку побудованого паралельного алгоритму.

Програмно реалізовано REST-API, здатний обробляти довільні набори вхідних даних у паралельному режимі та описано всі компоненти коду [7].

Аналіз результатів тестування програми виконано на наборах даних різного розміру (табл.1).

Таблиця 1 – Порівняння часу виконання програми з різними розмірами вхідного масиву та кількістю потоків

Розмір масиву	Потоки	Запитів	Середній час, с	Мін час, с	Макс час, с	Успішних
100	4	5	0.045	0.043	0.048	5/5
100	8	5	0.038	0.035	0.041	5/5
500	4	5	0.185	0.180	0.192	5/5
500	8	5	0.162	0.158	0.167	5/5
1000	4	5	0.365	0.350	0.380	5/5
1000	8	5	0.310	0.298	0.322	5/5

Результати тестування підтвердили не лише коректність реалізованого алгоритму, а й високу ефективність обраної моделі паралелізації. При збільшенні розміру вхідного масиву від 100 до 1000 елементів час обробки зростав лінійно, без різких стрибків або деградації продуктивності. Кількість потоків також суттєво впливала на тривалість обчислень: збільшення числа воркерів від 4 до 8 забезпечило додаткове прискорення, особливо помітне на масивах розміру 500 та 1000 елементів. Така поведінка узгоджується з моделлю потокового паралелізму та демонструє баланс між накладними витратами на створення потоків і продуктивністю CPU-bound задач.

Таким чином, розроблений паралельний метод обробки API-запитів забезпечує::

- підвищення швидкодії та загальної продуктивності веб-сервісу;
- ефективне використання апаратних ресурсів;
- стабільну роботу при високому навантаженні;
- гнучке масштабування.

Подальше вдосконалення може передбачати оптимізацію обробки даних для ще більшого обсягу запитів, розширення API для обробки складніших структур даних, паралельну обробку із динамічним регулюванням кількості потоків.

Висновки

Досліджено програмний комплекс, що реалізує паралельну обробку запитів у середовищі Django REST Framework. Створений веб-сервіс демонструє ефективну організацію паралелізму на рівні обробки API-запитів, забезпечує високу продуктивність та стабільність під час роботи з різними обсягами вхідних даних. Створено модель паралельної обробки набору задач, у якій кожен запит обробляється незалежно за допомогою пулу потоків. На основі цієї моделі реалізовано алгоритм, що виконує паралельну обробку масивів даних зі складністю O(n), використовуючи ThreadPoolExecutor. Також наведено UML-діаграми.

Створено REST-API, здатний обробляти довільні набори вхідних даних у паралельному режимі. Реалізовано модуль parallel_engine.py, який виконує важкі операції у паралельних потоках. Завдяки застосуванню Django REST Framework вдалося забезпечити чітку архітектурну структуру, зручну маршрутизацію, надійну валідацію та масштабованість веб-додатка. Проведено тестування розробленої програми, результати показують лінійне зростання часу виконання та стабільну роботу сервера при великих масивах даних.

Визначено, що можливими напрямами вдосконалення програми може бути: інтеграція асинхронних обчислень (asyncio), адаптація до високонавантажених кластерних систем, підтримка масштабування через Gunicorn/Uvicorn, а також застосування GPU-обчислень або Celery-черг для обробки задач підвищеної складності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мартинюк А. Паралельні обчислення: навчальний посібник. Вінниця: ВНТУ, 2021. 78 с. URL: https://mpa.vntu.edu.ua/fdb/838/Lec_CITCSHI/Tema_2.pdf .
2. Брюханов, В. С., Рейда, О. М. Паралельна обробка запитів у інформаційних системах швидкісної обробки баз даних. ВНТУ, 2022. URL: https://ir.lib.vntu.edu.ua//handle/123456789/39823 .
3. Бессараб, Є. В. Методи обробки запитів в системах паралельних баз даних. Харківський національний університет радіоелектроніки, 2020. URL: https://openarchive.nure.ua/handle/document/18758 .
4. Django Documentation – Asynchronous Support. URL: https://docs.djangoproject.com/en/3.2/topics/async/ .
5. Howik — Boosting Django REST Framework Performance: What Really Works. URL: https://howik.com/optimizing-django-rest-framework-performance/ .
6. Arun Prasher Blog - Leveraging Asynchronous Views in Django REST Framework. URL: https://blog.arunprasher.com/2024/10/leveraging-asynchronous-views-in-django.html .
7. Poespas Blog - Efficiently Handling Large Amounts of Concurrent Requests with DRF. URL: https://blog.poespas.me/posts/2025/03/06/django-rest-framework-concurrent-requests/ .

Гончак Катерина Олександрівна – студентка кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: katagoncak@gmail.com;

Денисюк Валерій Олександрович – канд. техн. наук, доцент, доцент кафедри комп'ютерних наук, Вінницький національний технічний університет, м.Вінниця, e-mail: vad64@i.ua.

Honchak Kateryna Olexandrivna – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: katagoncak@gmail.com;

Denysiuk Valerii Olexandrovich – Ph.D., Assistant Professor, Assistant Professor of the Chair of Computer Science, Vinnytsia National Technical University, Vinnytsia, e-mail: vad64@i.ua