DOI: 10.34185/1991-7848.itmm.2025.01.037

РЕАЛІЗАЦІЯ РОЗПОДІЛЕННЯ ПРОЦЕСУ РОБОТИ ГЕНЕТИЧНОГО АЛГОРИТМУ ШЛЯХОМ ВПРОВАДЖЕННЯ КАНАЛІВ ТА ВУЗЛІВ ОБРОБКИ ПОВІДОМЛЕНЬ

Жадан А. А.[1], Шинкаренко В.І.[2]

[1] *Український державний університет науки і технологій, аспірант, Україна*

[2] *Український державний університет науки і технологій, д.т.н., професор, Україна*

Анотація. *Реалізація генетичного алгоритму шляхом використання архітектурного патерну проектування «Pipes and Filters» вирішує два основних завдання — ефективність роботи та легкість модифікації. Перша досягається за рахунок асинхронної комунікації та можливості масштабування окремих фаз додаючи нові вузли обраного типу. Друге забезпечується низьким рівнем зв'язності компонентів системи між собою, а саме вузли повинні знати лише формат і протокол, на основі яких вини повинні формувати повідомлення з результатом своєї роботи. Сучасні мови та середовища програмування надають великий набір інструментів для реалізації систем подобного виду повністю абстрагуючи від взаємодії з ядром операційної системи. Найбільш провідною мовою програмування для реалізації розподілених обчислень є Golang.*

Ключові слова: *програмне забезпечення, інформаційні технології, генетичний алгоритм, pipes and filters, розподілені обчислення, асинхронне виконання, golang*

Класичним представленням генетичного алгоритму є поетапна обробка поточної популяції відповідно до обраного набору фаз. Реалізація такого підходу складається з набору циклічних операторів та функції обробки. Основним недоліком можна відзначити синхронний принцип роботи. Слід відзначити, що загальна складність вищезазначеного алгоритму за нотацією Ландау [1] є лінійною.

Альтернативним підходом є впровадження багатопотоковості в процес з використанням патерну проектування «Pipes and Filters» [2]. На поточний момент існує багато різновидів даного принципу, але для реалізації генетичного алгоритму найбільш відповідним є «Dump Pipes and Smart Endpoints» [3]. Його основа полягає у тому, що канали пересилання даних відповідають лише за пересилку. Вузли обробки даних виконують всю логіку роботи одного з визначених етапів загального процесу, що відповідає основним принципам реалізації генетичного алгоритму (рис. 1).

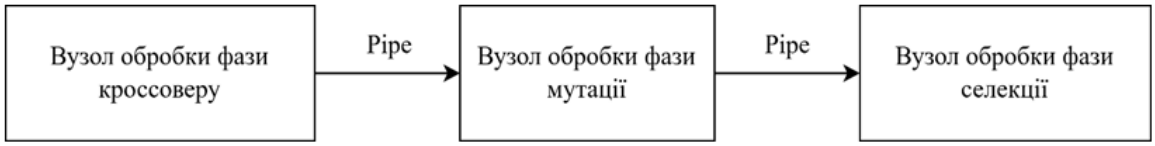

Рисунок 1 — Загальна схема реалізації алгоритму

Класичними елементами програмних реалізацій підходу «Pipes and Filters» на рівні локального середовища є багатопотоковість чи багатопроцесність. В даному випадку кожен вузол обробки виконується на основі окремого об‘єкта ядра. У якості загального алгоритму взаємодії зі об‘єктами даних є алгоритм вирішення задачі Постачальника-Споживача [4].

Основною проблемою багатопотокової реалізації є відсутності можливості до масштабування без зовнішнього керуючого компонента. В такому випадку найпростішим рішенням є визначення пікової кількості необхідних потоків обробки та їх запуск на початку роботи додатку (рис. 2). Основними недоліками є нераціональне використання ресурсів середовища та можливе зменшення продуктивності роботи при неочікуваних сплесках трафіку даних у системі.

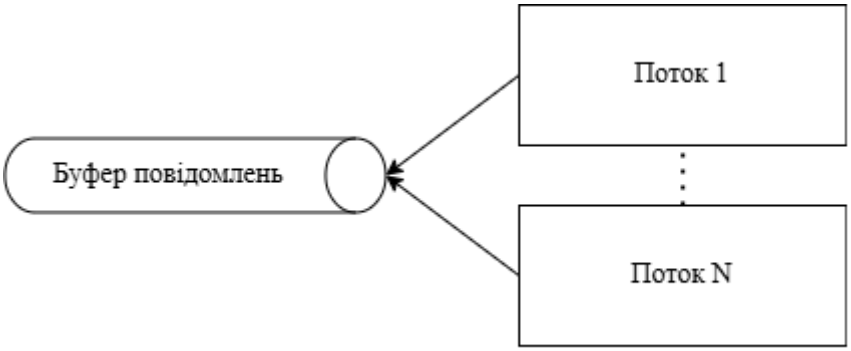

Рисунок 2 — Багатопотоковий підхід

Рішенням зазначених проблем є впровадження компонентів-менеджерів трафіку [5]. Їх основним завданням можуть буди не тільки розподілення навантаження між існуючими потоками обробки, а й запуск нових при недостатності обчислювальних можливостей. Або ж навпаки, зменшувати їх загальну кількість при недостатньому рівні трафіка або в визначені години низької інтенсивності (рис. 3).

У випадку багатопроцесності, окремий процес може повністю покривати функціональність вузла обробки, відповідаючи як за обробку даних, так і за процес масштабування. Слід зазначити, що реалізація даного підходу може відрізнятися базуючись на API обраної операційної системи ЕОМ.

З розвиненням технологій спрямованих на оптимізацію виконання програмних додатків все більше набували популярності рішення для розгалуження процесів обчислення, які надають самі середовища виконання, більшість з яких також підтримує ThreadPool [6]. Для оптимізації кількості використаних потоків, застосовується принцип асинхронного виконання задач [7].

Рисунок 3 — Багатопотоковий підхід з компонентом керування

Найбільш доцільною в питаннях інструментів для паралельних обчислень є мова програмування Golang. У додаток до вище зазначених компонентів вона також надає можливість використовувати канали повідомлень, реалізованих на рівні платформи. Кожен канал представляє собою чергу, яка може використовуватися для пересилання даних між асинхроними задачами. Слід зазначити, що виходячи з типу ітерації, а саме чи є вона найпершою, чи ні, перший вузол в ланцюжку обробки різний. При початку процесу, перший вузол повинен генерувати набір хромосом на основі заданий параметрів. В інших випадках, отриманий набір даних повинен бути лише перетворений відповідно до специфікації формату наступника (рис. 4). Для подальшого збільшення ефективності роботи можливе масштабування окремих елементів алгоритму на архітектурному рівні [8].

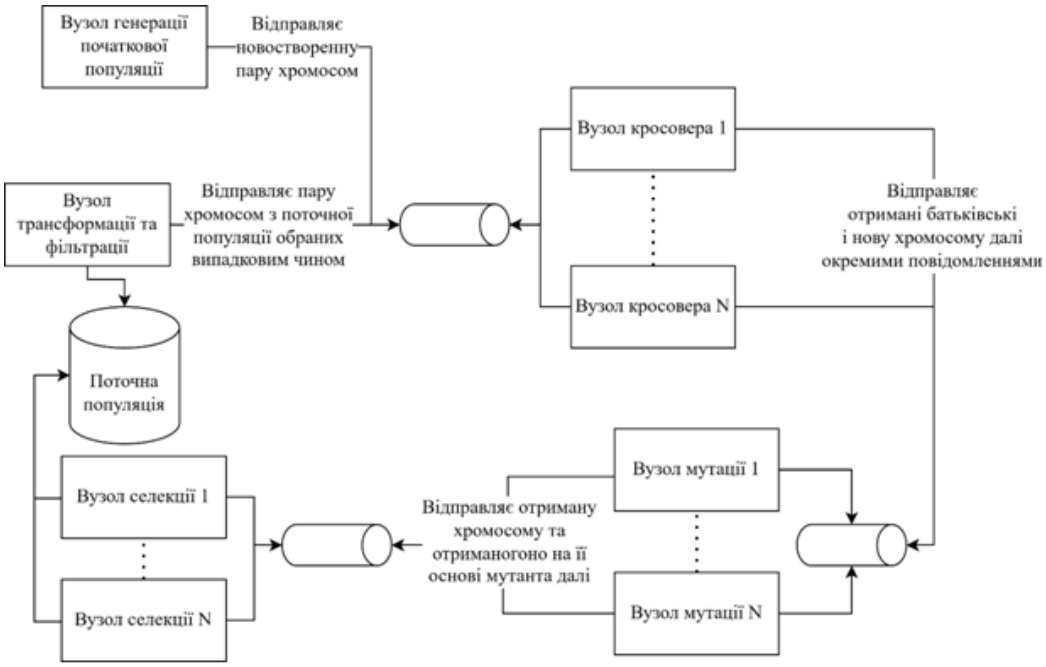

Рисунок 4 — Реалізація генетичного алгоритму інструментами мови Golang

В результаті даної роботи була спроектований та реалізований програмний модуль, який асинхронно виконує генетичний алгоритм на основі асинхронної комунікації між групами вузлів, кожна з яких відповідає за окрему фазу загального процесу. Пересилання даних між вузлами виконується за допомогою буферизованих черг повідомлень.

ЛІТЕРАТУРА

1. Cormen T. H.; Leiserson C. E.; Rivest R. L. Growth of Functions. Introduction to Algorithms 1990, pp. 23–41.
2. “Pipes and Filters pattern”. Microsoft documentation website. https://learn.microsoft.com/en-us/azure/architecture/patterns/pipes-and-filters (accessed Mar 30, 2025).
3. “Smart endpoints and dumb pipes” AWS official documentation website. https://docs.aws.amazon.com/whitepapers/latest/running-containerized-microservices/smart-endpoints-and-dumb-pipes.html (accessed Mar 30, 2025).
4. Abhishek Sai A. M., Reddy D., Raghavendra P., Kiran G. Y., Rejeenth V. R. Producer-Consumer problem using Thread pool. 3rd International Conference for Emerging Technology – 2022, pp. 1- 5.
5. Islam M. S., Rouff M. A., Threads Scheduling and Load Balancing with Loop Iteration in Multicore Processors: a Case Study with OpenMP. 3rd International Conference on Sustainable Technologies for Industry 4.0 – 2021, pp. 1-6.
6. “The CLR's Thread Pool”. Microsoft documentation website. https://learn.microsoft.com/en-us/archive/msdn-magazine/2003/june/net-column-the-clr-s-thread-pool (accessed Mar 30, 2025).

7. “Asynchronous programming scenarios”. Microsoft documentation website. https://learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming/async-scenarios (accessed Mar 30, 2025).
8. Shynkarenko V. I., Zhadan A. A., “Multiservice architecture of software for stochastic fractal time series forecasting” 2024 IEEE 19th International Conference on Computer Sciences and Information Technologies (CSIT), Zbarazh, Ukraine, в друку.

IMPLEMENTING DISTRIBUTED GENETIC ALGORITHM WORKFLOW WITH CHANNELS AND MESSAGE PROCESSING NODES

Artem Zhadan, Viktor Shynkarenko

Abstract. *The implementation of the genetic algorithm using the architectural design pattern "Pipes and Filters" solves two main problems – efficiency and ease of modification. The first is achieved due to asynchronous communication and the ability to scale individual phases by adding new nodes of the selected type. The second is ensured by a low level of connectivity of the system components among themselves, namely, the nodes should know only the format and protocol, based on which they should form messages with the results of their work. Modern languages and programming environments provide a large set of tools for implementing systems of this type, completely abstracting from interaction with the operating system kernel. The most leading programming language for implementing distributed computing is Golang.*

Keywords: *software engineering, information technologies, genetic algorithm, pipes and filters, distributed computing, asynchronous execution, golang.*

REFERENCE

1. Cormen T. H.; Leiserson C. E.; Rivest R. L. Growth of Functions. Introduction to Algorithms 1990, pp. 23–41.
2. “Pipes and Filters pattern”. Microsoft documentation website. https://learn.microsoft.com/en-us/azure/architecture/patterns/pipes-and-filters (accessed Mar 30, 2025).
3. “Smart endpoints and dumb pipes” AWS official documentation website. https://docs.aws.amazon.com/whitepapers/latest/running-containerized-microservices/smart-endpoints-and-dumb-pipes.html (accessed Mar 30, 2025).
4. Abhishek Sai A. M., Reddy D., Raghavendra P., Kiran G. Y., Rejeenth V. R. Producer-Consumer problem using Thread pool. 3rd International Conference for Emerging Technology – 2022, pp. 1- 5.
5. Islam M. S., Rouff M. A., Threads Scheduling and Load Balancing with Loop Iteration in Multicore Processors: a Case Study with OpenMP. 3rd International Conference on Sustainable Technologies for Industry 4.0 – 2021, pp. 1-6.
6. “The CLR's Thread Pool”. Microsoft documentation website. https://learn.microsoft.com/en-us/archive/msdn-magazine/2003/june/net-column-the-clr-s-thread-pool (accessed Mar 30, 2025).
7. “Asynchronous programming scenarios”. Microsoft documentation website. https://learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming/async-scenarios (accessed Mar 30, 2025).
8. Shynkarenko V. I., Zhadan A. A., “Multiservice architecture of software for stochastic fractal time series forecasting” 2024 IEEE 19th International Conference on Computer Sciences and Information Technologies (CSIT), Zbarazh, Ukraine, in publishing.