https://www.sworldjournal.com/index.php/swj/article/view/swj25-00-051
DOI: 10.30888/2663-5712.2024-25-00-051

УДК 004.432

OPTIMIZATION OF COMPUTER PROGRAM CODE: A MODEL APPROACH

ОПТИМІЗАЦІЯ КОДУ КОМП'ЮТЕРНИХ ПРОГРАМ: МОДЕЛЬНИЙ ПІДХІД

Krykhivskyi M.V. / Крихівський М.В.
c.t.s., as.prof. / к.т.н., доц.
ORCID: 0009-0000-3285-4308
Bandura V.V. / Бандура В.В.
c.t.s., as.prof. / к.т.н., доц.
ORCID: 0000-0003-3143-0946
Vavryk T.O. / Ваврик Т.О.
ORCID: 0000-0002-0612-0084
Hobyr L.M. / Гобир Л.М.
ORCID: 0009-0007-3176-2314
Ivano-Frankivsk National Technical University of Oil and Gas,
Ivano-Frankivsk, Karpatska, 15, 76019
Івано-Франківський національного технічного університету нафти і газу,
Івано-Франківськ, Карпатська 15, 76019

Анотація. *У роботі розглядається оптимізація коду комп'ютерних програм. Програмний код в сучасному світі є невід'ємною частиною різноманітних технологій, від мобільних додатків до великих корпоративних систем. Він є основний будівельним блоком будь-якої програми. Його продуктивність і ефективність впливають на швидкість виконання та ресурси, які використовуються. З кожним роком збільшується складність програмного забезпечення, що вимагає не лише розуміння програмування, але й вміння ефективно оптимізувати код для забезпечення оптимальної продуктивності та швидкодії. Автори розглядають сучасні підходи до оптимізації коду, включаючи техніки оптимізації швидкодії, використання ресурсів та зменшення футпринту програм. Також розглядається модельний підхід до оптимізації програмного коду як ефективний метод зниження часу виконання та ресурсів, необхідних для виконання програм, за допомогою аналізу та моделювання його роботи. Модельний підхід надає можливість попередньої оцінки впливу різних оптимізацій на продуктивність коду, забезпечуючи таким чином більшу ефективність процесу оптимізації та підвищуючи швидкість розробки програмного забезпечення. Традиційні методи оптимізації, такі як ручна оптимізація чи використання оптимізаторів компіляторів, мають свої обмеження та вимагають значних зусиль від розробників .У статті пропонуємо новий підхід до оптимізації коду програм за допомогою модельного підходу. Наш підхід базується на використанні математичного моделювання для автоматичного виявлення та усунення недоліків у програмному коді з метою покращення його продуктивності та ефективності.*

Ключові слова: *код, комп'ютерна програма, оптимізація, математичне моделювання.*

Вступ

Оптимізація програмного коду – це процес вдосконалення виконання програми шляхом зменшення споживання ресурсів, таких як час виконання, пам'ять і енергія. Оптимізація програмного коду є важливою складовою процесу розробки програмного забезпечення. Підвищення швидкодії та зниження споживання ресурсів може бути досягнуто за допомогою різних методів оптимізації, однак вибір правильного підходу та оцінка його впливу можуть бути складними завданнями.

У роботі [1] виділені рівні оптимізації програмного коду. Оптимізація може відбуватися на декількох рівнях. Відмічено, що більш високі рівні впливають сильніше й їх важче змінити пізніше у проєкті. Тому оптимізація, як правило, проводиться через удосконалення на вищому рівні до нижчого. Рівень дизайну вважається найвищим. Його можна оптимізувати для найкращого використання наявних ресурсів, враховуючи цілі, обмеження та очікуване використання/ навантаження. Наступним рівнем є рівень алгоритмів та структур даних. Враховуючи загальну конструкцію, продуманий вибір ефективних алгоритмів і структур даних (за асимптотичною складністю) і ефективна реалізація цих алгоритмів і структур даних може значно вплинути на продуктивність. За загальними алгоритмами та їх реалізацією на абстрактну машину слідує вибір рівня конкретного вихідного коду і це може мати істотне значення. На найнижчому рівні є написання коду, використовуючи мову асемблер, яка призначена для певної апаратної платформи і може виробляти найбільш ефективний і компактний код, якщо програміст користується повним набором машинних інструкцій.

Авторами роботи [2] розглянуто проблему підвищення ефективності програмного забезпечення обчислювальних систем інтегрованих у комп'ютерні мережі шляхом автоматичного визначення «вузьких місць». Вони формалізували задачу підвищення часової ефективності програмного забезпечення відповідно до стандарту ISO9126, побудували модель процесу роботи за допомогою графів, розробили схему системи автоматизованого підвищення ефективності програмного забезпечення обчислювальних систем на основі зменшення впливу «вузьких місць» і розкрито принципи її роботи.

У роботі [3] було проведено детальних аналіз та моделювання усіх етапів розробки програмного продукту із використанням понятійного апарату теорії множин та представлена математична модель, яка дозволили сформувати повну та чітку уяву про обсяги робіт, виконання яких лежить в основі успішної реалізації програмних систем. Запропонована авторами модель дає змогу охарактеризувати окремі етапи розробки продукту та аналітично описати життєвий цикл програмного забезпечення, що є передумовою для пошуку шляхів автоматизації окремих процесів підтримки прийняття управлінських рішень.

Модельний підхід до оптимізації програмного коду базується на створенні математичних моделей роботи програми та аналізі їх продуктивності при різних варіантах оптимізації. Цей підхід передбачає наступні кроки:

1) Аналіз роботи програми. Спочатку необхідно ретельно проаналізувати роботу програми та виявити частини коду, які займають найбільше часу виконання або використовують найбільше ресурсів.
2) Створення математичних моделей. Наступним кроком є створення математичних моделей роботи програми, які дозволяють передбачити її продуктивність при різних вхідних параметрах та оптимізаційних методах.
3) Експерименти з оптимізацією. Після створення моделей можна провести серію експериментів з застосуванням різних методів оптимізації та порівняти їх вплив на продуктивність програми.

4) Вибір оптимального варіанту. На основі результатів експериментів обирається оптимальний варіант оптимізації, який надає найбільше покращення продуктивності програми при мінімальних витратах.

Модельний підхід до оптимізації коду комп'ютерних програм має переваги і недоліки (рисунок 1).

Рисунок 1 - Переваги та недоліки модельного підходу

Джерело: авторська розробка

Основний текст

Оптимізація програмного коду – це процес вдосконалення його швидкодії та продуктивності. Це один із ключових аспектів розробки програмного забезпечення, особливо в умовах постійного росту обсягів даних та складності програмних продуктів. Для оптимізації програмного коду застосовуються різні моделі, які допомагають аналізувати та передбачати його продуктивність при різних оптимізаційних заходах. Розглянемо деякі основні математичні моделі, які використовуються в цьому контексті:

1. Аналітичні моделі. Ці моделі базуються на математичних рівняннях та статистичних методах для аналізу продуктивності програми, можуть використовуватися аналітичні моделі для прогнозування часу виконання коду в залежності від обсягу вхідних даних або параметрів оптимізації. Аналітичні моделі в оптимізації коду стали невід'ємною складовою цього процесу, дозволяючи розробникам здійснювати обґрунтовані вибори та досягати більшої ефективності.

Аналітичні моделі в оптимізації програмного коду стають дедалі важливішими для сучасних розробників програмного забезпечення. Вони

допомагають забезпечити оптимальну продуктивність та ефективність програм, що є ключовими факторами у конкурентному світі IT-індустрії.

2. Емпіричні моделі. Емпіричні моделі в оптимізації програмного коду – це підходи, що базуються на спостереженнях та експериментах з реальним кодом. Вони дозволяють розробникам отримувати практичні рекомендації щодо вдосконалення продуктивності програми на основі конкретних даних. Ці моделі базуються на експериментальних даних, отриманих під час тестування програми з різними параметрами оптимізації. Емпіричні моделі можуть допомогти визначити ефективність конкретних оптимізаційних методів на практиці. Розглянемо деякі з найбільш ефективних емпіричних моделей оптимізації коду.

Профілювання коду. Профілювання – це процес збору даних про виконання програми в реальних умовах. Він включає вимірювання часу виконання окремих фрагментів коду, використання пам'яті та інші метрики. На основі отриманих даних можна виявити «вузькі» місця в коді, де швидкодія або ефективність потребують покращень.

A/B тестування. A/B тестування – це метод порівняння різних варіантів коду для визначення найкращого. Розробники створюють декілька версій коду з різними оптимізаціями, після чого проводять експерименти для визначення найефективнішого варіанту.

Еволюційне програмування. Еволюційне програмування – це метод, який використовує принципи еволюції для оптимізації коду. Він включає в себе генерацію випадкових варіантів коду та оцінку їх продуктивності, "виживають" найкращі варіанти, що дозволяє покращити код без прямого втручання розробників.

Емпіричні моделі в оптимізації програмного коду є потужним інструментом для розробників. Вони дозволяють підходити до оптимізації з практичної точки зору, використовуючи дані та експерименти для досягнення кращих результатів. Завдяки цим моделям, розробники можуть покращувати продуктивність та ефективність свого коду, що є ключовим для успіху в сучасній індустрії програмного забезпечення.

3. Моделі профілювання. Моделі профілювання в програмуванні – це методи, які використовуються для збору та аналізу даних про виконання програми. Основна мета – виявити «слабкі» місця в коді, де зосереджені основні обчислювальні витрати, зайве використання пам'яті чи інші фактори, які сповільнюють роботу програми. Ці моделі використовуються для аналізу використання ресурсів, таких як пам'ять, процесорний час тощо, під час виконання програми. Вони дозволяють виявити "слабкі місця" в коді, які можуть бути оптимізовані.

Види моделей профілювання:

Часове профілювання. Цей підхід вимірює час виконання кожного фрагменту коду, дозволяючи виявити ті, які займають найбільше часу. Також можна виявити цикли або функції, які потребують оптимізації.

Профілювання пам’яті. За допомогою цього підходу вимірюється використання пам'яті кожним елементом програми. Це дозволяє виявити "великі" об'єкти або структури даних, які можна оптимізувати.

Профілювання системних викликів. Таке профілювання дозволяє виявити надмірні системні виклики або інші операції, які можуть бути оптимізовані для підвищення ефективності. дозволяє виявити надмірні системні виклики або інші операції, які можуть бути оптимізовані для підвищення ефективності.

Моделі профілювання для оптимізації програмного коду – це необхідний інструмент для розробників удосконалювати продуктивність та ефективність своїх програм. Вони дозволяють знаходити та виправляти слабкі місця в коді, що призводить до покращення якості програмного продукту та користувацького досвіду.

Моделі використання алгоритмів. Ці моделі аналізують продуктивність різних алгоритмів та їх вплив на загальну швидкодію програми. Вони допомагають вибрати найбільш ефективні алгоритми для конкретного завдання.

Моделі використання алгоритмів для оптимізації коду є потужними інструментами, які допомагають розробникам досягати цілей швидко та ефективно.

4. Моделі машинного навчання. Моделі машинного навчання в оптимізації коду використовуються для прогнозування найоптимальніших параметрів чи структур коду на основі великої кількості даних. Вони дозволяють автоматизувати процес оптимізації та знаходження найефективніших рішень.

Використання алгоритмів у процесі оптимізації коду

Алгоритми пошуку найкращого рішення: Ці алгоритми допомагають знайти оптимальні параметри чи структури для певної задачі. Наприклад, алгоритм генетичного пошуку може використовуватись для знаходження найкращого набору параметрів або оптимального розподілу ресурсів.

Алгоритми класифікації та кластеризації: Ці алгоритми дозволяють групувати схожі частини коду для подальшої оптимізації. Наприклад, кластеризація алгоритмів може допомогти ідентифікувати схожі патерни в коді та застосувати до них спільні оптимізації.

Алгоритми оптимізації функцій: Ці алгоритми використовуються для покращення швидкодії конкретних функцій чи алгоритмів. Наприклад, алгоритми градієнтного спуску застосовуються для оптимізації параметрів функцій.

Машинне навчання та нейронні мережі: Ці технології використовуються для автоматичної оптимізації параметрів чи структур коду на основі великого обсягу даних та попередніх виконань.

5. Моделі оптимізації компілятора. Компілятори є ключовими інструментами у процесі перетворення вихідного коду програми на виконуваний код. Оптимізація компілятора полягає у вдосконаленні швидкодії, ресурсоємності та загальної ефективності згенерованого коду. Ці моделі використовуються для аналізу та оптимізації роботи компілятора, який перетворює вихідний код програми в машинний код. Вони дозволяють виявити можливості для автоматичної оптимізації коду на етапі компіляції.

Модель зниження рівня абстракції (Lowering Model). Ця модель зосереджується на перетворенні високорівневих структур даних і операцій на більш низькорівневі аналоги.

Модель підтримки оптимізації використання ресурсів (Resource Optimization Model). Ця модель спрямована на ефективне використання обчислювальних ресурсів, таких як процесорні потоки та пам'ять. Компілятор може оптимізувати розміщення та управління пам'яттю, а також розподіл обчислювальних задач для максимізації продуктивності.

Модель оптимізації швидкості виконання (Performance Optimization Model). Ця модель спрямована на підвищення швидкості виконання програмного коду. Вона включає в себе різноманітні оптимізації, такі як згортання констант, видалення зайвих операцій, використання специфічних інструкцій процесора та інші техніки для зменшення часу виконання програми.

Модель оптимізації розміру коду (Code Size Optimization Model). Ця модель спрямована на зменшення розміру згенерованого виконуваного коду. Вона включає в себе видалення зайвих даних, згортання коду, використання стиснення та інші методи для зменшення об'єму пам'яті, який використовується програмою.

Математичні моделі допомагають програмістам та розробникам визначати та аналізувати ефективність свого програмного коду, розробляти стратегії оптимізації та удосконалення продуктивності. Оптимізація коду є складним процесом, який включає кілька етапів, від вибору правильних алгоритмів і структур даних до покращення використання ресурсів системи. Ми пропонуємо модель оптимізації коду, яка складається з п'яти основних етапів (рисунок 2):

- Аналіз та профілювання.
- Вибір оптимальних алгоритмів та структур даних.
- Оптимізація коду на рівні мови програмування.
- Мікрооптимізація та використання системних можливостей.
- Тестування та перевірка.

Ця структура демонструє кожен крок оптимізації у вигляді чітко визначених етапів, забезпечуючи логічний і послідовний процес покращення продуктивності коду. Такий підхід допомагає забезпечити оптимальний баланс між продуктивністю та підтримуваністю коду. Переваги запропонованої моделі:

- *Системність*: Охоплює всі аспекти продуктивності, від високого рівня до низькорівневих оптимізацій.
- *Гнучкість*: Підходить для оптимізації коду на різних мовах програмування.
- *Ефективність:* Застосування даної моделі дозволяє значно підвищити продуктивність програмного забезпечення шляхом поетапного поліпшення.

Формально модель можна представити за допомогою математичних формул. Нехай C – початковий код, T(C) – час виконання, M(C) – використання пам’яті, I(C) – кількість інструкцій, що виконуються, E(C) – енергоспоживання. Мета оптимізації полягає в мінімізації цих показників.

Етап аналізу та профілювання: $P(C)=\{T(C), M(C), I(C), E(C)\}$, де P(C) – профіль продуктивності коду C.

Етап оптимізація алгоритмів та структур даних: $C1=A(C)$, де A – функція, що застосовує оптимальні алгоритми та структури даних: $T(C1) \leq T(C)$, $M(C1) \leq M(C)$, $I(C1) \leq I(C)$, $E(C1) \leq E(C)$.

Етап оптимізації коду на рівні мови програмування: C2=L(C1), де L – функція, що оптимізує код на рівні мови програмування, яка включає заміну дорогих операцій, усунення надлишкового коду та використання специфічних можливостей мови: T(C2) ≤ T(C1), M(C2) ≤ M(C1), I(C2) ≤ I(C1), E(C2) ≤ E(C1)

Етап мікрооптимізації та використання системних можливостей: C3=S(C2), де S– функція, що застосовує мікрооптимізації та використовує системні можливості (оптимізація використання кешу, розпаралелювання, використання специфічних інструкцій).

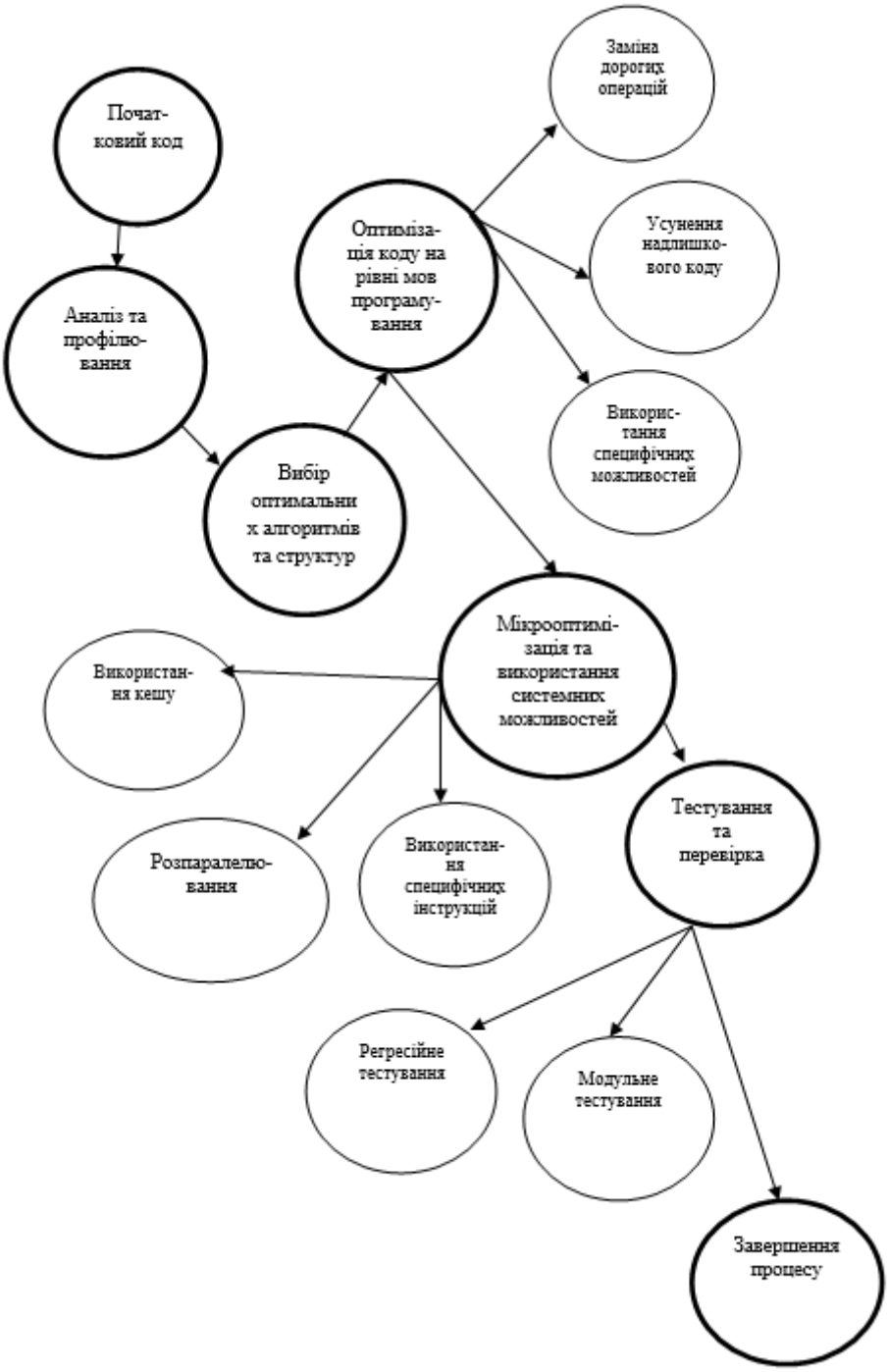

Рисунок 2 – Схема оптимізації коду

Джерело: авторська розробка

Етап тестування та перевірки: T(C3), M(C3), I(C3), E(C3) перевіряються за допомогою тестування. Якщо ∀, T(Ci) ≤ T(Ci-1), M(Ci) ≤ M(Ci-1), I(Ci) ≤ I(Ci-1), E(Ci) ≤ E(Ci-1) Coptimiz = C3. Процес оптимізації можна записати як послідовне застосування функцій: Coptimiz =S(L(A(C)))), де A – оптимізація алгоритмів та структур даних, L – оптимізація на рівні мови програмування, S – мікрооптимізація та використання системних можливостей.

Висновки

Представлена модель оптимізації коду пропонує системний підхід до покращення продуктивності програмного забезпечення. Формалізація процесу за допомогою математичних методів забезпечує чітку і послідовну оптимізацію, охоплюючи всі ключові аспекти від вибору алгоритмів до мікрооптимізацій. Такий підхід дозволяє розробникам ефективно покращувати свої програми, забезпечуючи при цьому стабільну та передбачувану продуктивність. Майбутні дослідження можуть зосередитися на автоматизації кожного етапу оптимізації, розробці інструментів для більш ефективного аналізу та оптимізації, а також на інтеграції цієї моделі в сучасні середовища розробки для полегшення застосування оптимізацій на практиці.

Література:

1. Кукуруза А.О. Методи оптимізації програми / А.О. Кукуруза, Д.П. Павлюк, В.В. Сеник, Б.Ю. Шутко // Збірник тез доповідей VIII Міжнародної науково-технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій», 27-28 листопада 2019 року. – Тернопіль. : ТНТУ, 2019. – Том 2. – С. 53–54.

2. Шабатура Ю.В. Нова методика підвищення ефективності програмного забезпечення обчислювальних систем шляхом автоматичного визначення «вузьких місць» / Ю.В. Шабатура, І.М. Штельмах, М.Ю. Шабатура // Наукові праці ВНТУ. – Вип. 2, Вересень 2011. Режим доступу: https://praci.vntu.edu.ua/index.php/praci/article/view/131

3. Кордунова Ю. С. Математичне моделювання процесу розробки спеціалізованих програмних систем безпеко-орієнтованого спрямування / Ю.С. Кордунова, М. Фелтіновскі, О.В. Придатко, О.О. Смотр // Вісник Львівського державного університету безпеки життєдіяльності. – Львів: Вип. 27, 2023. – С.23-31. DOI: 10.32447/20784643.27.2023.03.

Abstract. *The work considers the optimization of the code of computer programs. Software code in today's world is an integral part of various technologies, from mobile applications to large corporate systems. It is the basic building block of any program. Its performance and efficiency are affected by the speed of execution and the resources used. Every year, the complexity of software increases, requiring not only an understanding of programming, but also the ability to effectively optimize the code to ensure optimal performance and speed. The authors review modern approaches to code optimization, including techniques for optimizing performance, resource utilization, and reducing the application footprint. A model approach to program code optimization is also considered as an effective method of reducing execution time and resources required for program execution by analyzing and simulating its operation. The modeling approach provides an opportunity to pre-estimate the impact of various optimizations on the performance of the code, thus ensuring greater efficiency of the optimization process and increasing the speed of software development.*

Traditional optimization methods, such as manual optimization or the use of compiler optimizers, have their limitations and require considerable effort from developers. In the article, we propose a new approach to optimizing program code using a model approach. Our approach is based on the use of mathematical modeling to automatically detect and eliminate flaws in software code in order to improve its performance and efficiency.

Key words: *code, computer program, optimization, mathematical modeling.*

Стаття відправлена: 24.05.2024 г.