УДК 004.62

М. А. Благун

ВПЛИВ ВПОРЯДКУВАННЯ ДАНИХ НА ШВИДКІСТЬ ОБРОБКИ ЗАПИТІВ

Вінницький національний технічний університет

Анотація

Розглянуто сучасні підходи до впорядкування даних, спрямовані на підвищення швидкодії обробки інформації в програмних системах.

Ключові слова: впорядкування даних, сортування даних, кеш-пам'ять, Memory Wall, Data-Oriented Design, оптимізація, продуктивність систем.

Abstract

Modern data organization approaches for increasing the speed of data processing in software systems.

Keywords: data organization, data sorting, cache, Memory Wall, Data-Oriented Design, optimization, system performance.

Вступ

На даний момент, швидкість роботи процесорів значно перевищила швидкість роботи RAM-пристроїв. Це призводить до того, що процесор системи навіть з найпотужнішим апаратним забезпеченням змушений очікувати відповідь оперативної пам'яті, втрачаючи при цьому такти CPU. Це явище отримало назву «Memory Wall» [1]. Традиційні методи об'єктно-орієнтованого програмування не завжди враховують особливості роботи кеш-пам'яті сучасних процесорів [2] і, як результат, можуть призводити до хаотичного розташування об'єктів в динамічній пам'яті. Це, в свою чергу, зменшує швидкість обробки великих масивів інформації.

У зв'язку з цим актуальним є дослідження методів зменшення кількості запитів до RAM-пристроїв, які застосовуються для пришвидшення обробки інформації.

Метою даного дослідження є аналіз існуючих рішень для логічного та фізичного впорядкування даних, що дозволяють зменшити час обробки інформації.

Логічне впорядкування даних

Логічне впорядкування даних охоплює методи організації інформації на рівні її структури та способів доступу до неї. На відміну від фізичного впорядкування, яке пов'язане з особливостями розміщення даних у пам'яті комп'ютера, логічне впорядкування зосереджується на формуванні таких структур даних і способів їх зберігання та обробки, що дозволяють зменшити обсяг оброблюваної інформації або скоротити кількість операцій, необхідних для виконання запиту. Використання таких підходів дає змогу підвищити продуктивність програмних систем незалежно від конкретних апаратних засобів та особливостей реалізації пам'яті.

Компонентно-орієнтований дизайн [3] (DoD) – підхід впорядкування даних, що передбачає використання структури масивів замість класичного масиву об'єктів. Відповідно до компонентно-орієнтованого дизайну, дані об'єктів розділяються за типами компонентів та зберігаються в лінійних масивах. Коли система виконує запит, що стосується лише однієї характеристики, вона здійснює обробку лише одного суцільного масиву, уникаючи обробки інших байтів інформації, що стосуються інших властивостей об'єкта.

Сортування для оптимізації алгоритмів пошуку – підхід впорядкування даних, що передбачає попередню обробку даних за певним правилом для подальшої роботи з цими даними, використовуючи певні алгоритми. Гарним прикладом є перехід від лінійного пошуку в несортованому масиві до бінарного пошуку в відсортованому масиві або бінарному дереві, що дозволяє скоротити асимптотичну складність від O(n) до O(log n) [4].

Колоночне зберігання в аналітичних базах даних – підхід, що передбачає впорядкування даних не за рядками, а за колонками. Це дозволяє зчитувати з диску або оперативної пам'яті лише ті атрибути, що фігурують в аналітичному запиті. Оскільки дані колонки є однотипними, вони добре піддаються

компресії, що призводить до зменшення обсягу інформації, яку необхідно передавати по шині пам'яті, що є особливо корисним у Big Data-системах [5].

Фізичне впорядкування даних

Для вирішення проблеми затримки відповіді оперативної пам'яті, сучасні центральні процесори використовують кілька рівнів кеш-пам'яті. У контексті швидкодії доступу до даних в оперативній пам'яті на апаратному рівні, є три ключові поняття: кеш-лінія, промах кешу та попадання в кеш. Механізм кеш-ліній передбачає, що процесор ніколи не зчитує з оперативної пам'яті окрему змінну чи байт. Натомість завжди здійснюється завантаження фіксованого блоку даних, найчастіше розміром 64 байти. Такий блок має назву «кеш-лінія». Якщо дані впорядковані хаотично, ймовірність того, що дані, які потрібно обробити наступними, знаходяться у раніше завантаженій кеш-лінії, суттєво знижується і тому системі потрібно робити новий запит на завантаження з оперативної пам'яті. Це називається «промах кешу». Якщо ж дані впорядковані лінійно та послідовно, завантаження першого елемента при обробці даних, призводить до автоматичного завантаження кількох наступних елементів. Такий випадок має назву «попадання в кеш» [6].

Судячи з написаного вище, очевидно, що впорядкування даних, при якому стається більше попадань в кеш та менше промахів кешу, є більш сприятливим для швидкодії обробки даних, якщо не враховувати час, необхідний для здійснення впорядкування.

Найбільш розповсюджений приклад фізичного впорядкування даних – це використання суцільних масивів з послідовною обробкою їх елементів. Усі інші підходи так чи інакше зводяться до використання або суцільних масивів, або до користувацького програмного впорядкування даних для потреб системи за допомогою програмного забезпечення для роботи з RAM-пристроями.

Обмеження та компроміси

Впорядкування даних не завжди є доцільним в тій чи іншій системі, оскільки здійснення того чи іншого впорядкування накладає певні обмеження та компроміси:

- Навантаження на операції запису/видалення: підтримка даних у постійно впорядкованому стані може бути ресурсномістким, оскільки будь-яка операція вставки, видалення чи модифікації, вимагає зміщення інших елементів у пам'яті або перебудови індексних структур.
- Додаткові витрати пам'яті: створення допоміжних упорядкованих структур підвищує загальне використання пам'яті системи.
- Ускладнення коду: перехід від інтуїтивно зрозумілих об'єктних моделей до суворого кеш-орієнтованого проєктування ускладнює архітектуру програмного забезпечення та процес його підтримки.

Висновки

У результаті дослідження було проаналізовано основні підходи до логічного та фізичного впорядкування даних, спрямовані на підвищення швидкодії обробки інформації в сучасних інформаційних системах. Встановлено, що організація даних може зменшити кількість операцій, необхідних для виконання запитів, а також скоротити обсяг даних, які потрібно зчитувати та обробляти під час роботи програмних систем.

Дослідження показало, що методи логічного впорядкування даних, зокрема компонентно-орієнтований дизайн, попереднє сортування даних та колоночне зберігання інформації, дають змогу підвищити продуктивність обчислень шляхом оптимізації доступу до необхідних даних та зменшення обсягу надлишкових операцій. Особливої ефективності такі підходи набувають під час роботи з великими масивами даних та аналітичними навантаженнями.

Також встановлено, що фізичне впорядкування даних безпосередньо впливає на ефективність використання кеш-пам'яті процесора. Послідовне розміщення та обробка даних сприяють збільшенню кількості попадань у кеш і зменшенню кількості промахів кешу, що дозволяє знизити вплив проблеми «Memory Wall» та скоротити час доступу до інформації.

Крім того, визначено, що впорядкування даних є компромісом між швидкістю виконання запитів, витратами пам’яті, складністю реалізації та продуктивністю операцій модифікації даних. Тому під час проєктування програмних систем вибір конкретних методів впорядкування повинен здійснюватися з урахуванням особливостей навантаження, характеру даних та вимог до продуктивності системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Scalable Parallel Coset Enumeration: Bulk Definition and the Memory Wall. Gene Cooperman, Victor Grinberg, 2002. URL: https://www.sciencedirect.com/science/article/pii/S0747717102905237
2. Cryptography for Developers, 2007. URL: https://www.sciencedirect.com/topics/computer-science/processor-cache
3. Data-Oriented Design. Richard Fabian, 2018. URL: https://www.dataorienteddesign.com/dodbook.pdf
4. A Practical Introduction to Programming and Problem Solving. Stormy Attaway, 2012. URL: https://www.sciencedirect.com/science/chapter/monograph/abs/pii/B9780123850812000132
5. Joe Celko’s Complete Guide to NoSQL. Joe Celko, 2014. URL: https://www.sciencedirect.com/science/chapter/monograph/abs/pii/B9780124071926000029
6. ARM System Developer's Guide. ANDREW N. SLOSS, DOMINIC SYMES, CHRIS WRIGHT, 2004. URL: https://www.sciencedirect.com/science/chapter/monograph/abs/pii/B9781558608740500137

Благун Михайло Андрійович – студент групи 1ПІ-25м, факультет інформаційних технологій та комп’ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: me.shine.factory@gmail.com

Науковий керівник: ***Ліщинська Людмила Броніславівна*** – д-р техн. наук, професор, професор кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: llb@vntu.edu.ua

Blahun Mykhailo A. – Department of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: me.shine.factory@gmail.com

Supervisor: ***Lishchynska Lyudmyla Bronislavivna*** – Dr. Sc. (Eng.), Full Professor, Professor of Program Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: llb@vntu.edu.ua