DOI: 10.34185/1991-7848.itmm.2025.01.122

ОПТИМІЗАЦІЯ ПРОЦЕСУ ПОШУКУ СЕГМЕНТІВ ПАМ'ЯТІ: МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ПОВНОТИ ІНФОРМАЦІЙНОГО ВМІСТУ ПРИ ФІКСОВАНИХ ОБ'ЄМАХ ПАМ'ЯТІ

Мітіков М.Ю.

Дніпровський національний університет імені Олеся Гончара, аспірант, Україна

Анотація. *У роботі досліджено проблему надмірного використання оперативної пам'яті в програмних системах, зокрема при зберіганні даних у вигляді масивів System.Byte[]. В умовах обмежених ресурсів і високих навантажень постає необхідність ефективного використання доступної пам'яті без залучення додаткових обчислювальних потужностей. Запропоновано математичну модель, що базується на аналізі знімків пам'яті, та методику визначення масивів з низькою щільністю інформації, які є кандидатами на стиснення. Для зменшення обсягу зайнятої пам'яті використано алгоритм LZ4, що забезпечує збереження інформаційного вмісту при зменшенні фізичного розміру масивів. Дослідження реалізовано на платформі .NET із використанням бібліотеки ClrMD для програмного аналізу знімків пам'яті. Проведено серію експериментів на знімках промислових систем, які підтвердили ефективність запропонованого підходу: зменшення обсягу пам'яті до 45% для окремих масивів без втрати даних. Результати свідчать про доцільність впровадження подібних рішень для підвищення продуктивності систем і скорочення витрат на інфраструктуру. Науковий внесок полягає в розробці нової метрики оцінки ефективності стиснення та рекомендаційної системи для виявлення надмірного споживання пам'яті в керованих середовищах виконання.*

Ключові слова: *оптимізація пам'яті, алгоритми стиснення, масив байтів (byte[]), знімок пам'яті, надмірне використання пам'яті, математичне моделювання, ClrMD, LZ4, управління ресурсами, високонавантажені системи.*

Вступ. Зростання обсягів даних і вимог до продуктивності спричиняє актуальність оптимізації використання оперативної пам'яті в програмних системах. Більшість досліджень зосереджуються на виявленні витоків пам'яті, тоді як проблема надмірного, але формально допустимого споживання залишається недостатньо дослідженою. Зокрема, масиви System.Byte[], які зберігають інформацію у вигляді повторюваних значень, можуть займати значний обсяг пам'яті.

Мета роботи – розробити математично обґрунтований підхід до оптимізації використання оперативної пам'яті в програмних системах шляхом

виявлення та стиснення об’єктів типу System.Byte[], зберігаючи повноту інформаційного вмісту. Практична цінність математичної моделі полягає в можливості створення інформаційної системи, яка на основі аналізу знімків пам’яті дозволяє оцінити потенціал зменшення споживання пам’яті без масштабування інфраструктури, шляхом застосування ефективних алгоритмів стиснення (зокрема, LZ4).

Постановка проблеми. Традиційні моделі веб-розробки часто супроводжуються складністю керування інфраструктурою, високими витратами на підтримку серверів і складністю масштабування додатків у відповідь на зміну навантаження. Розробники стикаються з проблемами швидкості розгортання, надмірними витратами на невикористані ресурси та складністю забезпечення високої доступності додатків. Саме тому постає потреба у використанні безсерверних технологій, таких як Azure Functions, для створення гнучких, масштабованих і економічно ефективних веб-додатків.

Основною метою дослідження є аналіз можливостей Azure Functions для створення веб-додатків з використанням технологій .NET, оцінка їх переваг порівняно з традиційними серверами та визначення найкращих практик використання безсерверних архітектур.

Огляд літератури. В роботі проаналізовано сучасні дослідження та публікації щодо використання безсерверних архітектур у веб-розробці. Особливу увагу приділено роботам, що висвітлюють переваги Azure Functions, їх інтеграцію з .NET, а також питання продуктивності, масштабування та безпеки. Дослідження показують, що безсерверні рішення забезпечують значне скорочення часу на розробку, зменшення витрат на підтримку та підвищення надійності веб-додатків.

Дослідження. У ході дослідження було виконано аналіз технології Azure Functions у контексті створення веб-додатків з використанням .NET. Особливу увагу приділено вивченню механізмів автоматичного масштабування, особливостей інтеграції з іншими сервісами Azure (наприклад, Azure Storage, Cosmos DB, Service Bus) та можливостей забезпечення безпеки і авторизації

через Azure Active Directory. Було реалізовано експериментальний додаток для практичної перевірки переваг і обмежень цієї технології.

Аналіз та результати. Результати дослідження продемонстрували, що використання Azure Functions та платформи .NET дозволяє значно скоротити час на розробку і розгортання веб-додатків, а також підвищити їх доступність завдяки автоматичному масштабуванню. Було виявлено, що інтеграція Azure Functions з іншими сервісами Azure забезпечує гнучкість архітектури та простоту управління ресурсами. Водночас, дослідження вказало на важливість правильного вибору підходів до безпеки та авторизації для забезпечення захисту додатків.

Висновки. Azure Functions у поєднанні з .NET надають ефективний інструмент для створення сучасних веб-додатків у безсерверній архітектурі. Це дозволяє розробникам сконцентруватися на створенні доданої вартості застосунків, мінімізувати витрати на інфраструктуру та забезпечити автоматичне масштабування залежно від навантаження.

Перспективи подальших досліджень. Подальші дослідження можуть бути спрямовані на поглиблене вивчення питань безпеки, оптимізації витрат у хмарі та аналіз сценаріїв інтеграції безсерверних рішень з іншими популярними технологіями, такими як мікросервісна архітектура, DevOps та Continuous Integration/Continuous Deployment (CI/CD). Також перспективними є дослідження використання Azure Functions для спеціалізованих доменів, таких як IoT, Big Data та Machine Learning, що дозволить розширити застосування безсерверних технологій у різних сферах.

ЛІТЕРАТУРА / REFERENCE

1. Gregg, B. (2021). Scaling solutions (Sec. 2.7.3, p. 929). In Systems Performance (2nd ed.). Boston, MA: Addison-Wesley.
2. Bentley, J. L. (1982). Writing efficient programs. Englewood Cliffs, NJ: Prentice-Hall.
3. Efficiently and precisely locating memory leaks and bloat. Gene Novark, Emery D. Berger, Benjamin G. Zorn. Dublin : Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2009. 978-1-60558-392-1.
4. Analyzing Data Structure Growth Over Time to Facilitate Memory Leak Detection. Weninger, Markus. Mumbai : Association for Computing Machinery, 2019. 978-1-4503-6239-9.

5. Understanding and Combating Memory Bloat in Managed. Khanh Nguyen, Kai Wang, Yingyi Bu, Lu Fang, and Guoqing Xu. 4, University of California, Irvine : Athens Information Technology, 2018 p., T. 26. https://doi.org/10.1145/3162626.
6. Lee, S. (2015). Mitigating the performance impact of memory bloat [Doctoral dissertation, Georgia Institute of Technology]. Georgia Tech Institutional Repository. http://hdl.handle.net/1853/56174
7. Xia, Q., Ji, H., Zhou, Y., & Kim, N. S. (2025). Hardware-accelerated kernel-space memory compression using Intel QAT. IEEE Computer Architecture Letters, 24(1), 57–60. https://doi.org/10.1109/LCA.2025.3534831
8. Tsai, P.-A., & Sánchez, D. (2019, April). Compress objects, not cache lines: An object-based compressed memory hierarchy. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2019) (pp. 229–242). Association for Computing Machinery. https://doi.org/10.1145/3297858.330400
9. Mitikov, N. Y., & Guk, N. A. (2023). Identification of problems in software operation based on the analysis of memory snapshots. Issues of applied mathematics and mathematical modelling, (18), 171–178. https://doi.org/10.15421/322318 [Ukrainian]
10. Lou, C., Chen, C., Huang, P., Dang, Y., Qin, S., Yang, X., Li, X., Lin, Q., & Chintalapati, M. (2022, July). RESIN: A holistic service for dealing with memory leaks in production cloud infrastructure. In M. K. Aguilera & H. Weatherspoon (Eds.), Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’22) (pp. 109–125). USENIX Association. ISBN 978-1-939133-28-1
11. Ioannis T. Christou, Sofoklis Efremidis. To Pool or Not To Pool? Revisiting an Old Pattern. Marousi : Athens Information Technology, 2018. arXiv:1801.03763.

OPTIMISATION OF THE PROCESS OF SEARCHING FOR MEMORY SEGMENTS: MATHEMATICAL MODELLING AND EXPERIMENTAL EVALUATION OF THE COMPLETENESS OF INFORMATION CONTENT WITH FIXED MEMORY VOLUMES

Mitikov Nikolay

Abstract. *The paper investigates the problem of excessive use of RAM in software systems, in particular when storing data in the form of System.Byte[] arrays. In conditions of limited resources and high loads, there is a need for efficient use of available memory without involving additional computing power. The paper proposes a mathematical model based on the analysis of memory snapshots and a methodology for determining arrays with low information density that are candidates for compression. To reduce the amount of memory occupied, the LZ4 algorithm is used, which ensures the preservation of information content while reducing the physical size of arrays. The study was implemented on the .NET platform using the ClrMD library for software analysis of memory snapshots. A series of experiments on snapshots of industrial systems have been*

conducted to confirm the effectiveness of the proposed approach: reducing the memory size by up to 45% for individual arrays without data loss. The results indicate the feasibility of implementing such solutions to improve system performance and reduce infrastructure costs. The scientific contribution is the development of a new metric for evaluating compression efficiency and a recommendation system for detecting excessive memory consumption in managed runtime environments.

Keywords: *memory optimisation, compression algorithms, byte[] array, memory snapshot, excessive memory usage, mathematical modelling, ClrMD, LZ4, resource management, high-load systems.*