УДК 004.415

В.П. Гаврилюк
О.М. Ткаченко

РОЗРОБКА МЕТОДУ БАГАТОПРОЦЕСОРНОГО ЕКСПОРТУ АНІМАЦІЙ З АДАПТИВНИМ РОЗПОДІЛОМ РЕСУРСІВ

Вінницький національний технічний університет

Анотація. *Розроблено інноваційний метод багатопроцесорного експорту анімацій, який використовує адаптивний розподіл системних ресурсів для паралельної обробки кадрів. На відміну від традиційних систем послідовного експорту, запропонований метод забезпечує оптимальне використання багатоядерних архітектур сучасних процесорів через динамічне управління робочими процесами та інтелектуальний моніторинг системного навантаження. Проведено експериментальне дослідження ефективності методу з порівняльним аналізом продуктивності відносно традиційних підходів.*

Ключові слова: *паралелізація, закон Амдала, масштабованість, архітектура SMP, NUMA.*

Abstract. *An innovative method for multiprocessor animation export has been developed, which uses adaptive allocation of system resources for parallel frame processing. Unlike traditional sequential export systems, the proposed method ensures optimal use of multicore architectures of modern processors through dynamic management of workflows and intelligent monitoring of system load. An experimental study of the effectiveness of the method with a comparative analysis of performance relative to traditional approaches has been conducted.*

Ключові слова: *parallelization, Amdahl's law, scalability, SMP architecture, NUMA.*

Фундаментальні принципи паралельних обчислень

Теоретичним підґрунтям багатопроцесорних систем є закон Амдала, який формально описує теоретичні межі прискорення при паралелізації обчислень [1]. Закон визначає максимальне прискорення як $S = 1/(1-P+P/N)$, де P представляє частку коду, що може бути паралелізована, а N означає кількість доступних процесорних ядер [2]. Практичне значення цього закону полягає у визначенні оптимальної стратегії розподілу обчислювальних завдань між процесорами [3].

Ефективність паралелізації визначається як $E = S/N \times 100\%$, що дозволяє оцінити якість використання доступних обчислювальних ресурсів [4]. Ідеальна ефективність 100% досягається рідко через накладні витрати на міжпроцесорну комунікацію, синхронізацію та розподіл даних [5]. Реальні системи демонструють зниження ефективності при збільшенні кількості процесорів через зростання накладних витрат [6].

Масштабованість паралельних алгоритмів характеризується здатністю ефективно використовувати збільшення кількості обчислювальних ресурсів. Сильна масштабованість означає здатність зменшувати час виконання фіксованого завдання при збільшенні кількості процесорів, тоді як слабка масштабованість обробляє більші обсяги даних за сталий час при пропорційному збільшенні ресурсів [7].

Сучасні багатоядерні процесори реалізують архітектуру симетричної багатопроцесорності (SMP), де всі ядра мають рівноправний доступ до спільної пам'яті. Ця архітектура забезпечує ефективний розподіл навантаження між ядрами, але створює потенційні вузькі місця у доступі до пам'яті при високому рівні конкуренції за ресурси [8].

Ієрархічна структура кешів процесора впливає на ефективність паралельних обчислень через локальність даних. Кеші першого рівня (L1) є приватними для кожного ядра, кеші другого рівня (L2) можуть бути спільними або приватними, а кеш третього рівня (L3) зазвичай є спільним для всіх ядер [9]. Оптимальні алгоритми паралелізації враховують цю структуру для мінімізації промахів кешу та максимізації пропускної здатності пам'яті.

NUMA-архітектури (Non-Uniform Memory Access) створюють додаткові перешкоди для паралельних обчислень через нерівномірні затримки доступу до різних банків пам'яті [10]. Ефективні алгоритми повинні враховувати топологію пам'яті та намагатися максимізувати локальність доступу до даних [11].

Статичний розподіл завдань розподіляє роботу між процесорами на основі аналізу структури задачі. Цей підхід ефективний для регулярних завдань з передбачуваним навантаженням, але може призводити до дисбалансу при нерівномірному розподілі складності підзадач.

Динамічне балансування навантаження адаптується до реального розподілу обчислювальної складності через моніторинг продуктивності процесорів та перерозподіл завдань у реальному часі [12]. Алгоритми типу

work-stealing дозволяють незайнятим процесорам отримувати завдання від перевантажених, що забезпечує більш рівномірне використання ресурсів.

Гібридні підходи комбінують статичний попередній розподіл з динамічними корекціями на основі моніторингу продуктивності [14]. Такі методи особливо ефективні для завдань з частково передбачуваною структурою, де загальний розподіл навантаження можна оцінити заздалегідь, але локальні флуктуації потребують динамічного балансування.

Основною метою дослідження є розробка та експериментальна валідація методу багатопроцесорного експорту анімацій, який забезпечує оптимальне використання обчислювальних ресурсів сучасних багатоядерних систем через адаптивне управління розподілом завдань та інтелектуальний моніторинг системного стану.

Перше завдання дослідження полягає у створенні алгоритму автоматичного аналізу системних ресурсів для визначення оптимальної конфігурації паралельних процесів. Алгоритм повинен враховувати кількість доступних процесорних ядер, обсяг доступної оперативної пам'яті, поточне системне навантаження та специфічні характеристики завдання експорту анімації.

Друге завдання включає проведення комплексного експериментального дослідження ефективності розробленого методу з порівняльним аналізом продуктивності відносно традиційних послідовних підходів. Дослідження повинно охоплювати різні конфігурації завдань та апаратних платформ для оцінки універсальності та масштабованості методу.

Новизна дослідження полягає у розробці комплексного підходу до багатопроцесорного експорту анімацій, який інтегрує методи аналізу системних ресурсів, динамічного балансування навантаження та адаптивного управління процесами. Розроблений підхід відрізняється від існуючих здатністю автоматично адаптуватися до змінних умов системного середовища та характеристик оброблюваних завдань.

Система аналізу системних ресурсів

Першим елементом наукової новизни є розробка алгоритму адаптивного розподілу ресурсів, який динамічно визначає оптимальну кількість робочих процесів на основі багатофакторного аналізу системного стану. Алгоритм враховує не лише статичні характеристики апаратної платформи, але й динамічні параметри включаючи поточне навантаження, температурний режим та доступність пам'яті.

Технічна реалізація методу базується на модулі MultiprocessingAnimationExporter, який забезпечує комплексний аналіз доступних системних ресурсів для визначення оптимальної конфігурації паралельних процесів. Алгоритм цього модуля наведено на рисунку 1.

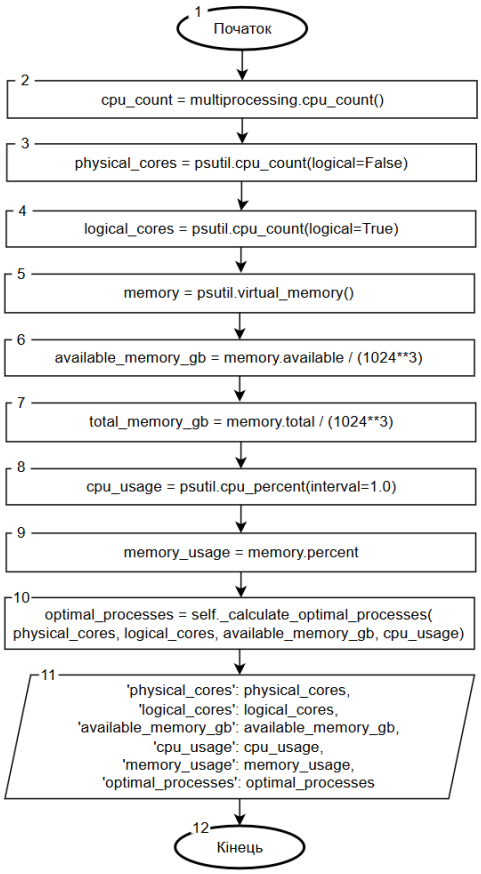

Рисунок 1 — Алгоритм модуля MultiprocessingAnimationExporter

Алгоритм розрахунку оптимальної кількості процесів використовує багатофакторний підхід для визначення конфігурації, що забезпечує стабільну роботу без перевантаження системи. Блок-схема цього алгоритму наведена на рисунку 2.

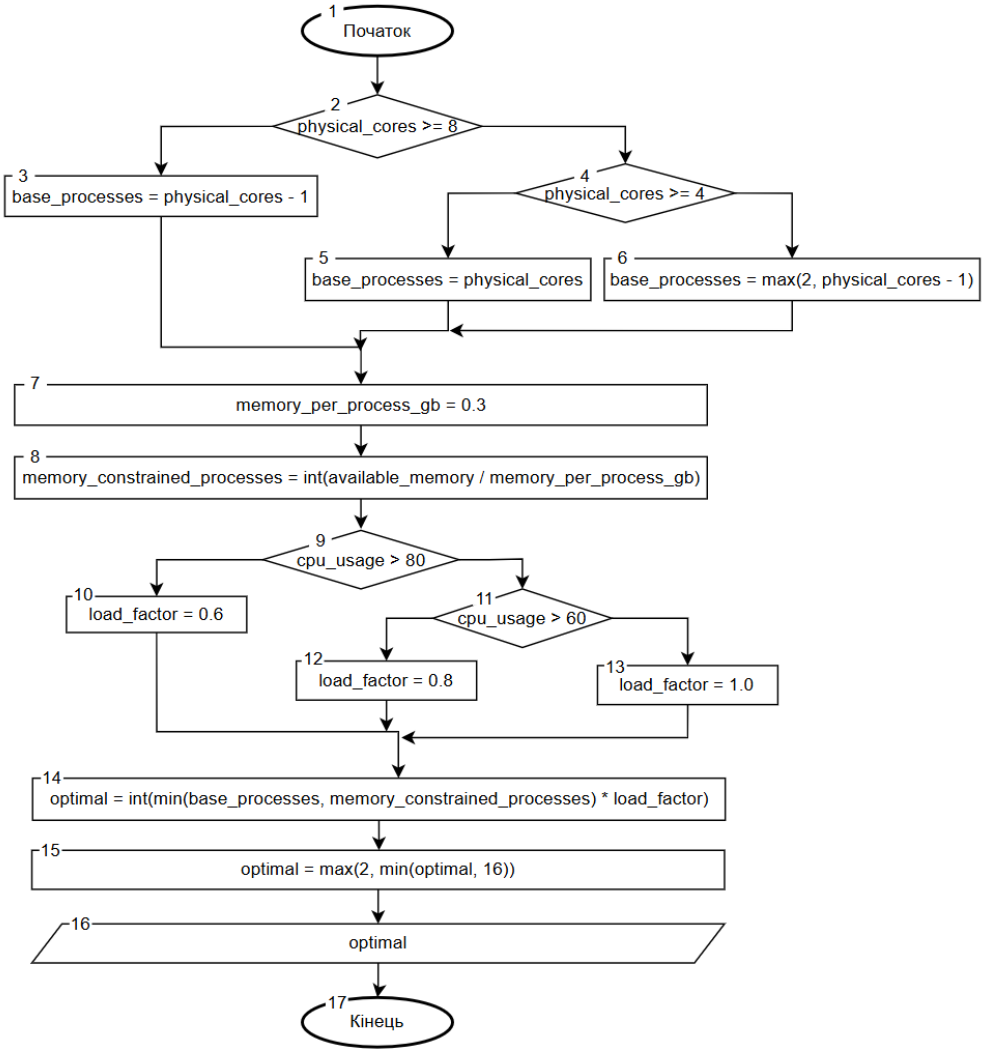

Рисунок 2 — Блок-схема алгоритму розрахунку оптимальної кількості процесів

Функція спочатку визначає кількість фізичних ядер процесора та встановлює базову кількість процесів залежно від апаратної конфігурації. Для систем з менш ніж 8 ядрами використовується консервативний підхід, а для потужніших систем дозволяється більш агресивне використання ресурсів.

Наступним кроком є розрахунок обмежень пам'яті. Функція оцінює, скільки процесів може підтримати система виходячи з доступної оперативної пам'яті, використовуючи норму 0.3 ГБ на процес. Це запобігає перевантаженню системи та використанню файлу підкачки.

Завершальним етапом є аналіз поточного навантаження процесора. Функція застосовує коригувальні коефіцієнти залежно від завантаженості CPU: при високому навантаженні кількість процесів зменшується, при низькому дозволяється повне використання ресурсів.

Функція повертає оптимальну кількість процесів, яка забезпечує максимальну швидкість обробки анімації без ризику дестабілізації системи. Результат автоматично адаптується до конкретних характеристик апаратного забезпечення та поточного стану системи, гарантуючи ефективну роботу на різних конфігураціях від бюджетних до високопродуктивних робочих станцій.

Система паралельної обробки кадрів

Другим аспектом наукової новизни є створення системи інтелектуального моніторингу ресурсів з можливістю прогнозування потенційних вузьких місць та превентивного корегування параметрів виконання. Система використовує історичні дані про продуктивність для оптимізації майбутніх завдань та попередження критичних станів системи.

Модуль паралельної обробки використовує ProcessPoolExecutor для ефективного розподілу завдань між робочими процесами з підтримкою моніторингу прогресу та обробки помилок.

Блок-схема системи паралельної обробки кадрів наведена на рисунку 3.

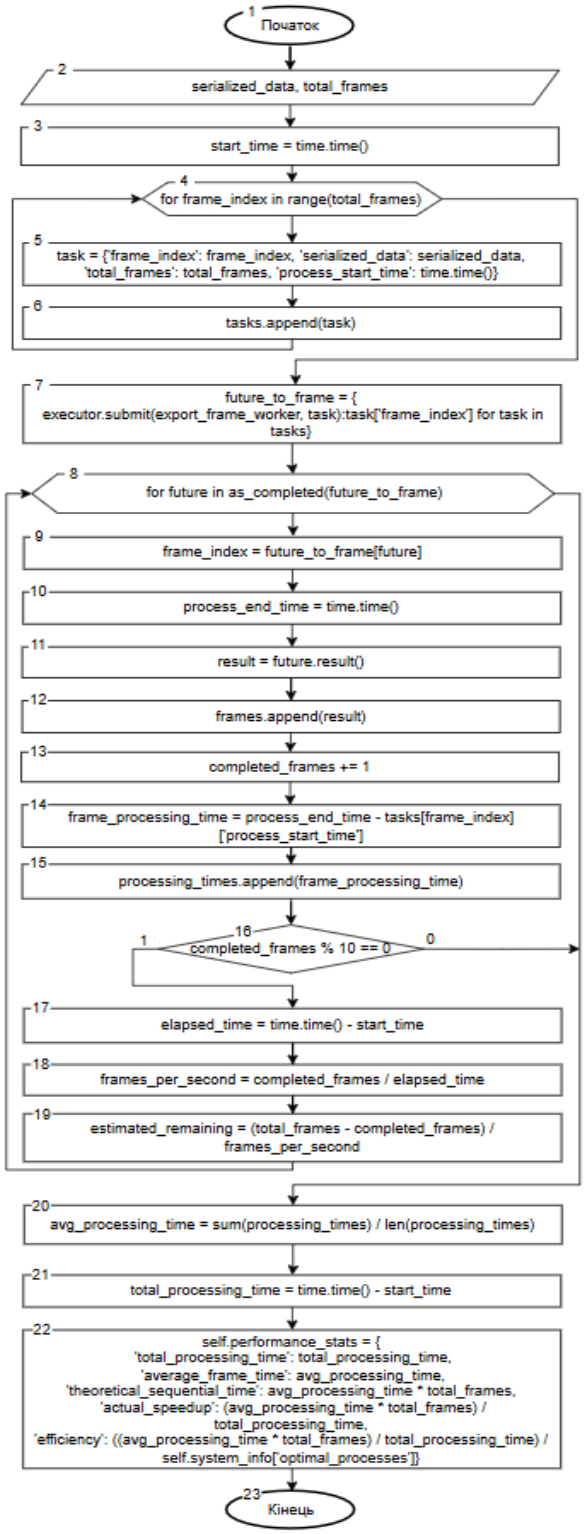

Рисунок 3 - Блок-схема системи паралельної обробки кадрів

Алгоритм ініціалізує вхідні параметри, включаючи серіалізовані дані анімації та загальну кількість кадрів для обробки. Система фіксує початковий час виконання для подальших розрахунків продуктивності та створює структури даних для відстеження прогресу обробки. Ця підготовча фаза забезпечує правильну конфігурацію всіх компонентів системи перед початком ресурсоємних обчислювальних операцій.

Після цього система створює окремі завдання для кожного кадру анімації, включаючи всі необхідні дані та параметри обробки. Завдання додаються до черги для подальшого розподілу між доступними робочими процесами. Такий підхід забезпечує ефективне використання всіх доступних обчислювальних ресурсів та рівномірний розподіл навантаження.

Після формування повної черги завдань алгоритм ініціює паралельне виконання. Кожне завдання передається окремому робочому процесу для незалежної обробки відповідного кадру анімації. Це дозволяє одночасно обробляти множинні кадри, скорочуючи загальний час генерації анімації порівняно з послідовними методами.

Система використовує механізм асинхронного очікування завершення завдань, що дозволяє ефективно використовувати час очікування для обробки вже готових результатів. Кожен отриманий результат додається до загального масиву кадрів, а лічильник завершених завдань оновлюється для відстеження прогресу.

Розраховується середній час обробки одного кадру, поточну швидкість генерації кадрів на секунду та прогнозований час завершення всього завдання. Ці дані використовуються не лише для інформування користувача про прогрес, але й для можливої динамічної оптимізації процесу обробки.

Функція для обробки кадрів

Функція виконує обробку окремого кадру у ізольованому процесі з мінімізацією накладних витрат на серіалізацію та десеріалізацію даних. Блок-схема алгоритму роботи функції наведена на рисунку 4.

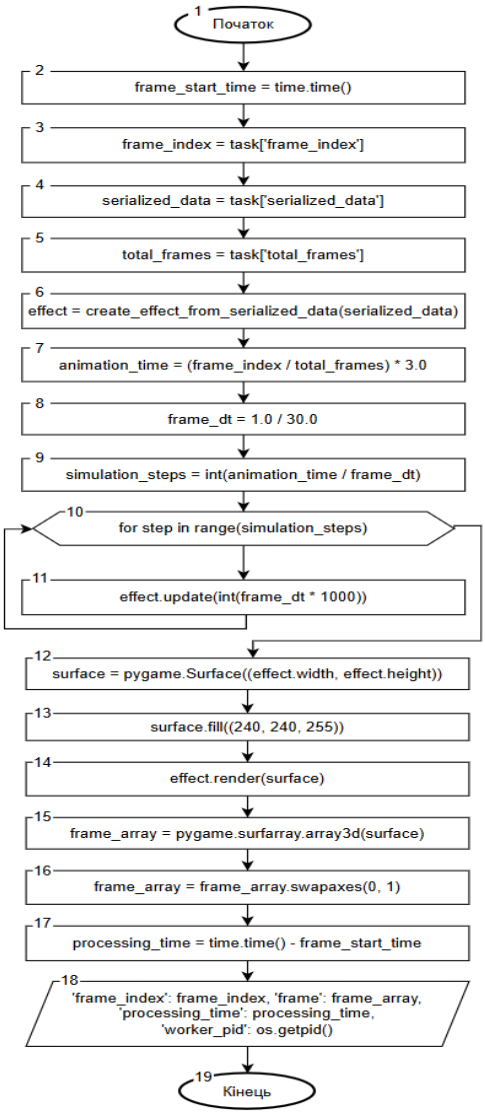

Рисунок 4 — Алгоритм роботи функції обробки кадрів

Алгоритм розпочинає роботу з фіксації часу початку обробки кадру та отримання ключових параметрів з переданого завдання. Система отримує індекс поточного кадру, серіалізовані дані ефекту та загальну кількість

кадрів в анімації. Ці параметри є необхідними для правильного позиціонування кадру в часовій послідовності анімації та забезпечення коректної роботи алгоритмів візуальних ефектів.

Наступним етапом є десеріалізація даних ефекту для створення повнофункціонального об'єкта ефекту в локальному процесі. Система розраховує поточний час анімації на основі індексу кадру та загальної тривалості анімації, що дозволяє точно позиціонувати кадр у часовій шкалі. Додатково обчислюється крок часу для дискретного оновлення симуляції, який визначає точність та плавність анімаційних переходів.

Цикл симуляції який поетапно оновлює стан ефекту від початкового моменту до цільового часу кадру. Система виконує множинні ітерації оновлення ефекту з малим часовим кроком, що забезпечує точне моделювання фізичних процесів та складних анімаційних взаємодій. Такий підхід гарантує коректність симуляції навіть для швидко змінюваних ефектів з високою динамікою.

Після завершення симуляції алгоритм переходить до етапу візуалізації кадру. Система створює поверхню для рендерингу з параметрами, що відповідають розмірам ефекту, заповнює її базовим кольором та викликає функцію рендерингу ефекту. Цей процес трансформує внутрішній стан симуляції у візуальне представлення, придатне для збереження та відображення.

У результаті, згенерований кадр конвертується з внутрішнього формату pygame у стандартний формат масиву NumPy. Система виконує необхідні трансформації координат та кольорових каналів для забезпечення сумісності з подальшими етапами обробки. Алгоритм також розраховує час обробки кадру для моніторингу продуктивності системи.

Дослідження ефективності розробленого методу

Експериментальне дослідження ефективності розробленого методу проводилося на стандартизованій апаратній платформі з процесором Intel Core i5-1135G, що містить 4 фізичних ядра, 8 логічних процесорів та 16ГБ оперативної пам'яті DDR4-3200. Операційною системою слугувала Windows 11 Pro з активованим режимом високої продуктивності.

Тестування включало 7 сценаріїв з різною кількістю кадрів та складністю ефектів. Перший сценарій передбачав експорт короткої анімації тривалістю 1 секунда з 30 кадрами на секунду. Останній сценарій передбачав експорт анімації тривалістю 36 секунд з 30 кадрами за секунду.

Між тестами здійснювалася пауза тривалістю 2 хвилини для стабілізації температурного режиму системи та очищення пам'яті від залишкових процесів.

Результати порівняльного аналізу прискорення запропонованого методу багатопроцесорного експорту анімацій порівняно з традиційним методом зведено у таблицю 1.

Таблиця 1 - Порівняльні результати тестування методів експорту анімацій

Кількість кадрів	Тривалість (сек)	FPS	Коефіцієнт прискорення	Категорія завдання
30	1.0	30	0.26x	Мале
60	2.0	30	0.62x	Мале
90	3.0	30	0.50x	Середнє
120	4.0	30	0.69x	Середнє
180	6.0	30	0.75x	Середнє
720	24.0	30	1.06x	Велике
1080	36.0	30	1.56x	Дуже велике

На основі отриманих результатів можна зробити висновок, що запропонований метод багатопроцесорного експорту анімацій з адаптивним розподілом ресурсів є менш ефективним за традиційні методи експорту анімацій для невеликих та нескладних анімацій, що пояснюється накладними витратами на ініціалізацію паралельних процесів, виділення пам’яті та запуску ядер.

Однак, можемо побачити, що при тривалості анімації в 720 кадрів запропонований метод стає ефективнішим за традиційним, та дає прискорення на 6%. При 1080 кадрів прискорення складає уже 56%. Тобто, доцільним є використання запропонованого методу для великих та тривалих анімацій.

Отже, метод багатопроцесорного експорту анімацій з адаптивним розподілом ресурсів може бути корисним для ігрових застосунків, де візуальна складність та технічні вимоги до графічного контенту постійно зростають. Ігрові застосунки характеризуються особливо високими вимогами до ресурсоємності анімацій, що робить традиційні послідовні методи обробки неефективними для сучасних потреб.

Ігрові анімації відрізняються від інших типів мультимедійного контенту кількома ключовими характеристиками, які суттєво впливають на вимоги до обчислювальних ресурсів. Сучасні ігри використовують анімації з високою частотою кадрів, зазвичай 60 або навіть 120 кадрів на секунду, що автоматично збільшує загальну кількість кадрів у порівнянні з традиційними анімаціями. Крім того, ігрові анімації часто включають складні візуальні ефекти, такі як системи частинок, динамічне освітлення, симуляція фізики та процедурні деформації, які вимагають значних обчислювальних ресурсів для кожного окремого кадру.

Розроблений метод особливо актуальний у контексті поширення багатоядерних процесорів у робочих станціях розробників ігор. Сучасні системи розробки зазвичай оснащуються процесорами з 8, 16 або навіть більшою кількістю ядер, але традиційні методи обробки анімацій не здатні ефективно використовувати цей потенціал. Багатопроцесорний підхід дозволяє максимально використати доступні обчислювальні ресурси, забезпечуючи кращу віддачу від інвестицій в апаратне забезпечення.

Висновки

Розроблений метод багатопроцесорного експорту анімацій з адаптивним розподілом ресурсів демонструє значні переваги над традиційними послідовними підходами при складних та тривалих анімаціях. Експериментальне дослідження продемонструвало прискорення на 56%.

Ключовими результатами дослідження є розробка алгоритмів автоматичного аналізу системних ресурсів, створення ефективної системи динамічного балансування навантаження та реалізація адаптивного моніторингу продуктивності, які забезпечують високу продуктивність при мінімальних вимогах до ручного налаштування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Amdahl G.M. Validity of the single-processor approach to achieving large scale computing capabilities / G.M. Amdahl // Proceedings of the April 18–20, 1967, Spring Joint Computer Conference. – ACM, 1967. – Pp. 483–485.
2. Quinn M.J. Parallel Programming in C with MPI and OpenMP / M.J. Quinn. – New York: McGraw-Hill Education, 2004. – 322 с.
3. Яровий А.А. Багаторівневі паралельні-ієрархічні системи та їх комп'ютерне моделювання/ А.А. Яровий // Методи та системи оптико-електронної і цифрової обробки зображень та сигналів. – Вінниця: ВНТУ, 2012. – Т. 24, № 2. – С. 50–51.
4. Hennessy J.L., Patterson D.A. Computer Architecture: A Quantitative Approach / J.L. Hennessy, D.A. Patterson. – 5-th ed. – Burlington, MA: Morgan Kaufmann, 2012. – 912 с.
5. Grama A., Gupta A., Karypis G., Kumar V. Introduction to Parallel Computing / A. Grama, A. Gupta, G. Karypis, V. Kumar. – 2-nd ed. – Upper Saddle River, NJ: Pearson Prentice Hall, 2003. – 494 с.
6. Pacheco P.S. An Introduction to Parallel Programming / P.S. Pacheco. – Burlington, MA: Morgan Kaufmann, 2011. – 312 с.
7. Foster I. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering / I. Foster. – Reading, MA: Addison-Wesley, 1995. – 260 с.
8. Tanenbaum A.S., Bos H. Modern Operating Systems / A.S. Tanenbaum, H. Bos. – 4-th ed. – Upper Saddle River, NJ: Pearson, 2015. – 1 136 с.
9. McCool M., Reinders J., Robison A. Structured Parallel Programming: Patterns for Efficient Computation / M. McCool, J. Reinders, A. Robison. – Burlington, MA: Morgan Kaufmann, 2012. – 350 с.
10. Sorin D.J., Hill M.D., Wood D.A. A Primer on Memory Consistency and Cache Coherence / D.J. Sorin, M.D. Hill, D.A. Wood. – San Rafael, CA: Morgan & Claypool, 2011. – 198 с.
11. McKee S.A. Reflections on the Memory Wall / S.A. McKee. – Proceedings of the 1st Workshop on Memory Performance: Dealing with Applications, Systems and Architectures. – IEEE Computer Society, 2004. – Pp. 124–136.
12. Blumofe R.D., Leiserson C.E. Scheduling Multithreaded Computations by Work Stealing / R.D. Blumofe, C.E. Leiserson // Journal of the ACM. – 1999. – Vol. 46, № 5. – Pp. 720–748.
13. Kumar V., Grama A., Gupta A., Karypis G. Introduction to Parallel Computing: Design and Analysis of Algorithms / V. Kumar, A. Grama, A. Gupta, G. Karypis. – Redwood City, CA: Benjamin/Cummings, 1994. – 384 с.
14. Hager G., Wellein G. Introduction to High Performance Computing for Scientists and Engineers / G. Hager, G. Wellein. – Boca Raton, FL: CRC Press, 2010. – 430 с.

Гаврилюк Вадим Павлович **–** студент групи 5ПІ-21б, ФІТКІ, Вінницький національний технічний університет, Вінниця, e-mail: politeh.conference@gmail.com;

Ткаченко Олександр Миколайович **–** *к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: alextk1960@gmail.com.*

Havrylyuk Vadym Pavlovych *– student of group 5PI-21b, FITKI, Vinnytsia National Technical University, Vinnytsia, e-mail: politeh.conferance@gmail.com;*

Tkachenko Oleksandr Mykolayovych *– Ph.D., Associate Professor of the Department of Software, Vinnytsia National Technical University, Vinnytsia, e-mail: alextk1960@gmail.com.*