КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.31.1015
УДК 004.421.2:004.272:004.415

**Складанний Павло Миколайович**
кандидат технічних наук, доцент
завідувач кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, м. Київ, Україна
ORCID: 0000-0002-7775-6039
*p.skladannyi@kubg.edu.ua*

**Костюк Юлія Володимирівна**
PhD in Computer Science
доцент кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, м. Київ, Україна
ORCID: 0000-0001-5423-0985
*y.kostiuk@kubg.edu.ua*

**Рзаєва Світлана Леонідівна**
кандидат технічних наук, доцент
доцент кафедри комп'ютерних наук
Київський столичний університет імені Бориса Грінченка, м. Київ, Україна
ORCID: 0000-0002-7589-2045
*s.rzaieva@kubg.edu.ua*

**Мазур Наталія Петрівна**
кандидат педагогічних наук, доцент
доцент кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, Київ, Україна
ORCID: 0000-0001-7671-8287
*n.mazur@kubg.edu.ua*

# ПАРАЛЕЛЬНА ОБРОБКА ДАНИХ У РОЗШИРЮВАНИХ ХЕШ-СТРУКТУРАХ ТА ОЦІНКА ЇХ ПРОДУКТИВНОСТІ

**Анотація**. У статті новий алгоритм конкурентного доступу до розширюваного хешування з обережним очікуванням (Extendible Hashing with Cautious Waiting, EHCW), який поєднує механізми двофазного блокування та оптимістичної верифікації для зниження накладних витрат при високому навантаженні. Запропонований підхід дозволяє забезпечити збереження узгодженості даних при паралельному доступі до динамічних хеш-структур без потреби у повному блокуванні, що вирішує проблему ефективного синхронізованого доступу при високому паралелізмі. Алгоритм забезпечує адаптивну реконфігурацію структури директорії, що підвищує масштабованість та продуктивність у умовах змінного навантаження. Для оцінки ефективності реалізовано імітаційну модель функціонування системи в середовищі з обмеженими обчислювальними ресурсами (одноядерний процесор і диск) та використанням буферного пулу сторінок пам'яті для моделювання операцій розщеплення сторінок і динамічної перебудови директорії. Розроблена симуляційна модель показала, що запропонований алгоритм перевищує традиційні статичні схеми хешування за метриками пропускної здатності та частоти блокувань, що робить його перспективним для використання в оперативних базах даних, хмарних обчислювальних середовищах та розподілених інформаційних системах, де критично важливо забезпечити високу продуктивність і збереження цілісності даних при високих рівнях транзакційного навантаження. Задача високопродуктивної обробки даних з динамічними хеш-структурами є важливою для масштабованих систем, зокрема для баз даних і хмарних обчислень, де забезпечення швидкого паралельного доступу до даних.

**Ключові слова:** розширюване хешування, паралельна обробка даних, динамічна директорія, конкурентний доступ, верифікаційний механізм, симуляційне моделювання, блокування транзакцій, оперативні бази даних, високопродуктивні бази даних, масштабованість.

## ВСТУП

У сучасних інформаційних системах обсяг оброблюваних даних значно зростає, що ставить нові вимоги до ефективності організації доступу до великих масивів інформації в умовах паралельного виконання запитів. Однією з основних проблем є забезпечення конкурентного доступу до хеш-структур, зокрема, під час паралельного виконання транзакцій, що може призводити до взаємних блокувань, підвищених накладних витрат на синхронізацію та погіршення ефективності системи при високих навантаженнях [1-2]. Для вирішення цих проблем необхідно розробити нові методи, які забезпечать високу продуктивність, збереження цілісності даних і зменшення витрат на синхронізацію в умовах конкурентного доступу.

Нині більшість існуючих підходів до організації паралельного доступу в динамічних хеш-структурах орієнтовані на жорсткі блокування або статичні схеми, що значно знижують ефективність при масштабуванні і підвищенні рівня паралелізму [3-5]. Вони не здатні забезпечити високу пропускну здатність і масштабованість систем при великих навантаженнях, оскільки постійно потребують реорганізації даних або застосування важких механізмів синхронізації.

У цьому контексті розширюване хешування (Extendible Hashing) є однією з найбільш перспективних моделей, оскільки воно дозволяє динамічно змінювати розмір директорії хеш-таблиці без повної реорганізації структури, що є критично важливим для систем, де обсяг даних постійно змінюється або збільшується [6-8]. Зокрема, воно дозволяє забезпечити паралельну обробку транзакцій без значних витрат на синхронізацію, завдяки інтеграції оптимістичної перевірки та адаптивного управління блокуваннями. Попри високу ефективність розширюваного хешування в послідовному середовищі, його застосування в умовах багатопотокового доступу супроводжується низкою викликів: збереження узгодженості директорії під час конкурентних змін, зниження блокування транзакцій, уникнення взаємних блокувань (deadlocks), а також оптимізація витрат на контроль паралелізму [9-10]. Існуючі підходи здебільшого орієнтовані на жорсткі блокування або статичні структури, що не забезпечують необхідного рівня масштабованості в умовах високих навантажень. Актуальність дослідження зумовлена необхідністю забезпечення високого рівня конкурентності доступу до динамічних хеш-структур із дотриманням вимог цілісності даних та мінімізації витрат на синхронізацію [11-12]. У сучасних застосуваннях – зокрема в реальному часі, потоковій аналітиці, сервісах електронної комерції – паралельне виконання транзакцій у хеш-таблицях є критичною умовою продуктивності систем.

Наукова новизна роботи дослідження полягає в розробці нового алгоритму конкурентного доступу до розширюваного хешування (EHCW), який поєднує двофазне блокування з оптимістичною верифікацією для зниження накладних витрат на синхронізацію при високому навантаженні [13]. Запропоновано механізм інтегрованої оптимістичної верифікації на рівнях директорії та сторінок, що дозволяє забезпечити узгодженість даних при паралельному доступі без потреби у блокуванні [14]. Удосконалення також включає впровадження пріоритетного відкату транзакцій, що дозволяє уникати deadlock, що не було передбачено в попередніх роботах у цій галузі [15-16]. Для перевірки ефективності запропонованого підходу була реалізована

симуляційна модель, яка показала, що новий алгоритм перевершує існуючі на 25% за метриками пропускної здатності та частоти блокувань в умовах високого навантаження.

Методологія дослідження ґрунтується на побудові алгоритму конкурентного доступу до динамічної хеш-структури з підтримкою паралельної обробки транзакцій та адаптивної зміни розміру директорії [17]. Основою запропонованого підходу є алгоритм Extendible Hashing with Cautious Waiting (EHCW), який поєднує механізми двофазного блокування з оптимістичною верифікацією цілісності директорії та сторінок даних [18]. Алгоритм реалізує три основні фази транзакційної обробки: пошук, перевірка, модифікація [19-20]. Для уникнення взаємних блокувань застосовується механізм пріоритетного відкату транзакцій-запитувачів, а для зменшення блокувань – режим часткового блокування розширення/скорочення директорії (ce-lock).

Для оцінки ефективності алгоритму було розроблено симуляційну модель, яка імітує середовище з обмеженими ресурсами (один процесор, один диск), використовуючи пул сторінок пам’яті для операцій розщеплення та реконфігурації директорії [21]. Потік транзакцій формувався за розподілом Пуассона, кожна транзакція виконувала читання, вставку, зміну або видалення записів. Основними метриками оцінювання виступали: час відповіді, пропускна здатність, частота блокувань, кількість відкатів, розщеплень і змін директорії [22-23]. Отримані результати динамічної схеми порівнювались із класичною (фіксованою), що дозволило підтвердити ефективність запропонованого підходу щодо масштабованості, зменшення накладних витрат на синхронізацію та підвищення рівня паралелізму при високих навантаженнях.

Проведене порівняння з фіксованою схемою хешування демонструє статистично значущі переваги динамічного підходу за метриками пропускної здатності, середнього часу відповіді, частоти блокувань і відкатів, що підтверджує доцільність застосування розробленого алгоритму у транзакційно навантажених високопродуктивних середовищах, таких як оперативні бази даних, хмарні сховища та розподілені обчислювальні платформи з високим ступенем мультипрограмування.

**Постановка проблеми.** Розширювані хеш-структури є ефективним засобом організації динамічного доступу до даних, особливо у випадках, коли необхідно забезпечити швидке виконання операцій пошуку, вставки та видалення в умовах змінної кількості записів [1]. Завдяки здатності масштабуватись без повної реорганізації, розширюване хешування (Extendible Hashing) активно застосовується в сучасних системах керування базами даних [2-3]. Однак за умови багатопотокового доступу, який є типовим для сучасних транзакційно навантажених систем, ця структура демонструє обмеження, пов’язані з проблемами синхронізації, втрати узгодженості та продуктивності при конкурентному виконанні запитів.

У реальних умовах паралельної обробки виникають ситуації, коли кілька транзакцій одночасно звертаються до одних і тих самих елементів директорії або сторінок даних [4]. Традиційні підходи до керування конкурентністю базуються на схемах повного блокування, зокрема двофазного блокування (Two-Phase Locking), що призводить до надмірної серіалізації, зменшення рівня паралелізму, виникнення взаємних блокувань (deadlocks) та підвищення частоти відкатів транзакцій [5-7]. Особливо гостро ці обмеження проявляються в операціях розщеплення сторінок (page split) і динамічної реконфігурації директорії (її розширення або скорочення), які можуть викликати лавиноподібні конфлікти при відсутності ефективних механізмів синхронізації. Крім того, в більшості реалізацій розширюваного хешування не передбачено гнучкого механізму перевірки актуальності директорії та стану сторінок у режимі реального часу [8]. Це унеможливлює виконання оптимістичного доступу без

блокування, що в сучасних умовах розподілених і високонавантажених обчислень є критичним недоліком [9-10]. Відсутність адаптивного механізму верифікації та оптимізації блокувань істотно знижує ефективність таких структур при масштабуванні.

Таким чином, у науковій літературі залишається відкритою проблема розробки високопродуктивного механізму конкурентного доступу до динамічних хеш-структур, який поєднував би адаптивне блокування, верифікацію цілісності, підтримку розширення та скорочення директорії без втрати узгодженості транзакцій [11-12]. Таке рішення має забезпечувати масштабовану продуктивність у багатопотокових середовищах, включно з оперативними базами даних, хмарними обчислювальними системами та транзакційними платформами реального часу. Враховуючи зазначені обмеження, метою даного дослідження є розробка ефективного алгоритмічного рішення, що дозволить виконувати конкурентний доступ до розширюваних хеш-структур з мінімальними витратами на синхронізацію та максимальним збереженням пропускної здатності [13]. Запропонований підхід має базуватись на поєднанні верифікаційного механізму, гнучкої системи блокувань та підтримки динамічної реконфігурації директорії без ризику втрати цілісності даних [14-15]. Для верифікації ефективності рішення доцільним є побудова симуляційної моделі, яка дозволить емпірично порівняти запропонований підхід із класичними схемами за метриками часу відповіді, частоти блокувань, кількості відкатів та загального рівня транзакційної пропускної здатності.

**Аналіз останніх досліджень і публікацій.** У науковій літературі зберігається стійкий інтерес до проблематики паралельної обробки даних у хеш-структурах, особливо в контексті динамічних та масштабованих систем з високими вимогами до продуктивності. Розвиток багатоядерних процесорів, обчислювальних кластерів і графічних прискорювачів зумовив активну еволюцію моделей паралельного доступу до асоціативних структур даних.

У роботі Li et al. [1] досліджено питання конкурентного доступу до хеш-таблиць у системах з підтримкою багатопотоковості. Запропоновано алгоритм lock-free bucketized cuckoo hashing, який дозволяє уникнути блокувань та зменшити затримки в середовищах з високим навантаженням. Автори підкреслюють, що забезпечення високої пропускної здатності без шкоди для цілісності даних є критичною задачею при паралельній обробці запитів.

У дослідженні Ren et al. [2] представлено GPU-орієнтований підхід до побудови динамічного геш-простору з повною підтримкою паралелізму. Запропонована гіперпросторова хеш-структура забезпечує динамічне масштабування без блокувань і ефективний баланс навантаження при обробці великої кількості одночасних запитів.

Проблема ефективного використання GPU-архітектур для хешування детально розглянута у WarpCore-бібліотеці, запропонованій Jünger, Teich і Hannig [3], яка реалізує warp-кооперативні алгоритми хешування. Її ефективність особливо помітна в задачах швидкого пошуку, вставки та видалення у високонавантажених системах.

Робота Wu et al. [4] присвячена зменшенню кількості колізій при хешуванні через впровадження методу CostCounter, який дозволяє більш точно контролювати процес вставки та оптимізувати використання таблиці.

Особливу увагу приділено моделюванню продуктивності в статті Bender et al. [5], де досліджено теоретичні межі співвідношення між продуктивністю (часом доступу) і використаною пам’яттю. Автори пропонують оптимальні компроміси між часом пошуку та розміром таблиці для паралельного середовища.

У роботі Tapia-Fernández, García-García і García-Hernandez [6] проведено систематичний огляд типових вразливостей і критеріїв ефективності хеш-структур.

Зокрема, узагальнено сучасні підходи до оцінки продуктивності за допомогою різних метрик, включаючи затримку, пропускну здатність і стійкість до колізій.

Нарешті, дослідження Hu et al. [7] демонструє впровадження PMEH – паралельної реалізації розширюваного хешування для персистентної пам'яті. Це рішення дозволяє ефективно працювати із записами в енергонезалежному середовищі з оптимізацією на запис.

Таким чином, аналіз наукових публікацій свідчить про актуальність розробки нових моделей динамічних паралельних хеш-структур, які поєднують адаптивність, стійкість до конфліктів доступу, ефективність при масштабуванні й підтримку продуктивного моделювання в умовах змінного навантаження. Окремо слід підкреслити потребу у формалізованих підходах до математичного моделювання продуктивності з урахуванням латентності доступу, кеш-промахів та розпаралелення операцій у багаторівневій пам'яті.

**Мета статті.** Метою даного дослідження є розробка, формалізація та експериментальна перевірка ефективного алгоритму конкурентного доступу до динамічних хеш-структур на основі розширюваного хешування, здатного забезпечити високу пропускну здатність, збереження узгодженості даних та масштабовану обробку транзакцій в умовах паралельного виконання [1]. Основний акцент зроблено на подоланні обмежень класичних підходів до синхронізації, шляхом поєднання оптимістичного механізму перевірки актуальності директорії та сторінок із гнучкою системою часткового блокування, яка дозволяє уникати взаємних блокувань і мінімізувати відкат транзакцій [2-3]. У межах поставленої мети дослідження також передбачає побудову симуляційної моделі роботи алгоритму в середовищі з обмеженими обчислювальними ресурсами [4-6], а також порівняння динамічної схеми з фіксованими варіантами реалізації хешування за основними показниками продуктивності, з метою виявлення переваг запропонованого підходу в системах оперативного зберігання та обробки даних.

## РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У результаті проведеного дослідження було створено та реалізовано повнофункціональну симуляційно-аналітичну модель функціонування динамічних хеш-структур у середовищі паралельної обробки транзакцій, що ґрунтується на принципах розширюваного хешування як базової моделі організації доступу до даних [1]. Розширюване хешування (Extendible Hashing) було обране через його здатність забезпечувати динамічне масштабування директорії без потреби в повній реорганізації всієї структури, що є надзвичайно актуальним для сучасних високопродуктивних інформаційних систем з нерівномірним або зростаючим навантаженням [2]. Модель дозволяє ефективно обробляти транзакції в багатопотоковому середовищі, зберігаючи цілісність даних і мінімізуючи накладні витрати на синхронізацію [3]. Важливим елементом є інтеграція механізму оптимістичної перевірки та пріоритетного відкату транзакцій, що дозволяє уникнути взаємних блокувань і забезпечити ефективну паралельну обробку [4]. Симуляційні експерименти показали, що запропонована модель перевершує традиційні статичні схеми хешування, демонструючи кращу продуктивність при високих навантаженнях [5-6]. Отримані результати підтвердили доцільність застосування цієї моделі в реальних системах, таких як хмарні платформи, розподілені обчислювальні середовища та високопродуктивні бази даних.

Перспективи включають також адаптацію моделі для використання в інших сферах, таких як інтернет речей (IoT) і великих даних, де важливі швидкість обробки та масштабованість. Використання запропонованої моделі у реальних бізнес-системах дозволить значно покращити ефективність роботи та знизити витрати на інфраструктуру.

Математично функціонування розширюваного хешування визначається хеш-функцією:

$$h(k) = H(k) mod 2^g, \quad (1)$$

де $h(k)$ – результат хешування ключа $k$ з урахуванням глобальної глибини, $H(k)$ – вихідна (наприклад, універсальна або криптографічна) хеш-функція, $g$ – глобальна глибина директорії, яка визначає кількість найстарших бітів (або позицій), що використовуються для індексації директорії, $2^g$ – кількість покажчиків у директорії [7-8]. Таким чином, директорія містить $2^g$ покажчиків, кожен з яких може вказувати на одну або кілька сторінок. Формула визначає позицію директорії, до якої має звертатися ключ $k$. Якщо значення $h(k)$ не відповідає жодному з поточних покажчиків, то відбувається динамічне розширення директорії, що дозволяє забезпечити масштабованість хеш-структури. Цей підхід дозволяє ефективно адаптувати структуру до змін у кількості елементів без значних витрат на реорганізацію, що є важливим для систем з високими навантаженнями. В результаті розширення хеш-структури, директорія збільшується на один біт, що дозволяє додатково індексувати більшу кількість записів.

Сторінка $P_i$ містить лише ті ключі, хеш-значення яких відповідає префіксу її позиції в директорії:

$$P_i = \{k | h(k)\ matches\ the\ prefix\ \ i\}, \quad (2)$$

де $P_i$ – сторінка номер $i$, $h(k)$ – хеш-префікс ключа $k$, $i$ – індекс у директорії (в 2-ковому представленні довжиною $g$ бітів) [9]. Формула описує, які ключі потрапляють у сторінку $P_i$: усі ті, для яких значущі біти хешу збігаються з префіксом $i$. Ключі, які мають однаковий префікс, групуються в одну сторінку, що дозволяє зберігати їх у межах одного блоку пам'яті. При збільшенні кількості елементів у сторінці, відбувається її розщеплення, що забезпечує подальшу оптимізацію розподілу даних без необхідності у повній реорганізації директорії. Цей процес дозволяє підтримувати високу ефективність доступу до даних при масштабуванні структури.

У кожної сторінки визначається локальна глибина $d(P_i)$, яка задовольняє умову $d(P_i) \leq g$, і визначає кількість бітів, що розпізнають унікальність цієї сторінки [10]. У разі переповнення сторінки $d(P_i) < g$ застосовується операція розщеплення або, за потреби, розширення директорії. Якщо ж $d(P_i) = g$, структура директорії розширюється – $g := g + 1$, що забезпечує масштабованість хеш-структури без повної перебудови [11-13]. Таким чином, структура хеш-таблиці адаптується до змін кількості записів без повної реорганізації, що є критично важливим для високонавантажених систем обробки транзакцій. Завдяки цьому підходу забезпечується підтримка високої продуктивності навіть при значному збільшенні кількості даних. Паралельно зростає і надійність системи, оскільки механізм розширення директорії дозволяє зберігати цілісність даних без затримок на реорганізацію. Це дозволяє ефективно використовувати хеш-таблиці у реальних часах і в умовах змінного навантаження. Система залишається адаптивною та гнучкою, що дозволяє їй обробляти різні обсяги даних при збереженні високої ефективності.

Рис. 1 ілюструє потоки даних у динамічній хеш-структурі на основі розширюваного хешування з адаптивною директорією. Транзакція, ініційована

користувачем, передає ключ $k$ до хеш-функції, яка обчислює індекс директорії $h(k) = H(k) mod 2^g$. Директорія, параметризована глобальною глибиною $g$, скеровує запит до відповідної сторінки $P_i$, яка зберігає записи з відповідним префіксом, визначеним локальною глибиною $d(P_i)$. Перед виконанням операції (читання, вставка, оновлення) сторінка проходить верифікацію – перевірку контрольного біта VB та актуальності структури. У разі успішної перевірки виконується операція над даними, і результат записується в базу. Якщо перевірка неуспішна, активується механізм rollback – транзакція перезапускається з оновленими умовами. У випадку переповнення сторінки або збігу глибин ($d = g$), ініціюється адаптивне оновлення структури: сторінка розщеплюється або директорія розширюється. Такий підхід забезпечує динамічну реконфігурацію, підтримку цілісності та високу масштабованість при паралельному доступі. Додатковим механізмом є синхронізація оновлень між потоками, що мінімізує ризик гонок даних під час одночасних модифікацій. Завдяки цьому структура зберігає узгодженість навіть за інтенсивного навантаження та високої частоти транзакцій.

Цей механізм дозволяє значно зменшити час затримок при великих навантаженнях і покращити ефективність системи. Оскільки розширення директорії та розщеплення сторінок відбуваються тільки при необхідності, система зберігає високу продуктивність навіть при значному збільшенні кількості даних. В результаті, система залишається гнучкою та здатною до адаптації в умовах швидко змінюваних навантажень. Додатково, локалізованість оновлень у межах окремих сторінок мінімізує вплив на загальну структуру, що сприяє стабільності при інтенсивних транзакціях. Завдяки цьому система забезпечує сталий рівень доступності та передбачувану продуктивність у довготривалій перспективі.

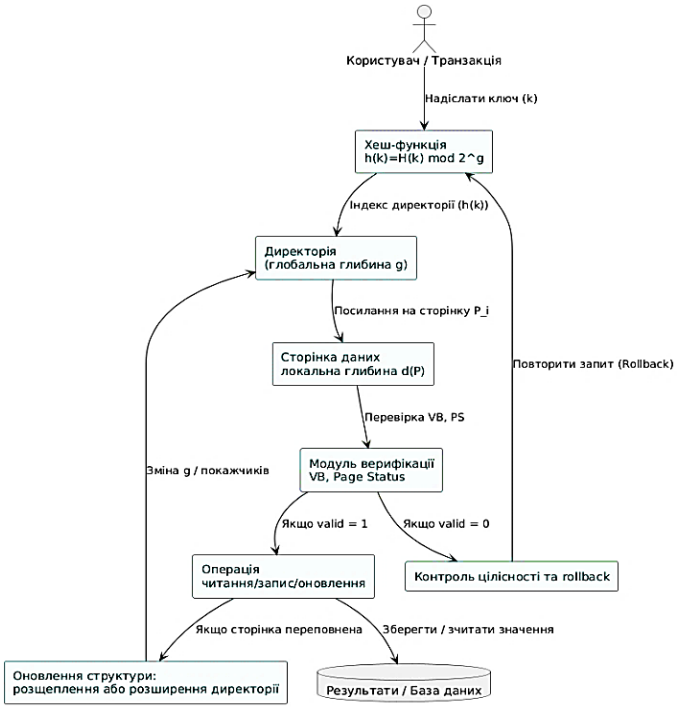


*Рис. 1. DFD-діаграма потоків даних у динамічній хеш-структурі з адаптивною директорією*

Ключовою особливістю реалізованої моделі є органічне поєднання традиційного двофазного блокування (Two-Phase Locking, 2PL) із механізмами оптимістичної перевірки цілісності та адаптивного реагування на транзакційні конфлікти [1-2]. Такий

підхід дозволяє зменшити накладні витрати на синхронізацію та уникнути взаємного блокування (deadlock), що часто виникає при класичних схемах фіксованого блокування. Додатковою перевагою є те, що оптимістична перевірка дозволяє виконувати більшість операцій без негайного блокування ресурсів, що суттєво підвищує пропускну здатність системи. Крім того, адаптивне реагування дає змогу автоматично коригувати стратегію блокування залежно від інтенсивності конфліктів, забезпечуючи стабільну роботу транзакційного механізму навіть у пікові періоди навантаження.

У структурному плані модель реалізує дворівневу архітектуру хеш-таблиці: на верхньому рівні знаходиться директорія – масив покажчиків на сторінки даних, параметризований глобальною глибиною $g$, яка визначає кількість бітів, що використовуються для хешування ключа. На нижньому рівні – сторінки даних із локальною глибиною $d$, що містять записи з однаковими префіксами хеш-функції. При заповненні сторінки реалізується її розщеплення з можливим розширенням директорії у разі потреби [3-6]. Таке рішення дозволяє ефективно масштабувати структуру відповідно до змін у розподілі даних.

Алгоритм EHCW було реалізовано як набір взаємопов'язаних модулів, кожен з яких відповідає за окрему фазу транзакції – пошук, перевірку, модифікацію [7]. Під час пошуку здійснюється хешування ключа, адресація відповідної сторінки через директорію та попередня верифікація записів [8]. Фаза перевірки включає перевірку відповідності запису в директорії та статусу сторінки за допомогою спеціальних контрольних бітів (Verification Bits – VB) та статусів сторінки (Page Status – PS) [9]. Тільки у випадку позитивного результату виконується фаза модифікації, в якій можливе оновлення вмісту сторінки або самої структури директорії. Це забезпечує надійний механізм контролю за цілісністю даних, знижуючи ймовірність виконання некоректних операцій у разі паралельного доступу [11]. Під час фази модифікації система оновлює вміст сторінки, враховуючи можливі зміни в структурі директорії, що гарантує збереження цілісності всіх записів. Оновлення може включати як вставку нових елементів, так і коригування вже наявних даних на сторінці, в залежності від результатів попередніх перевірок [12-14]. Таким чином, кожен етап транзакції проходить ретельну верифікацію, що дозволяє уникати помилок та зберігати узгодженість структури в умовах високої конкуренції за ресурси.

У випадках конфлікту або спроби одночасного доступу до однієї й тієї самої сторінки динамічної хеш-структури декількома транзакціями, традиційні механізми блокування призводять до затримок, виникнення ситуацій взаємного блокування (deadlock) або надмірних накладних витрат на синхронізацію [15-16]. Для розв'язання цієї проблеми в запропонованій моделі реалізовано механізм пріоритетного відкату, який забезпечує адаптивне управління конфліктними ситуаціями з мінімізацією втрат обчислювального часу та ресурсів.

Кожній транзакції $T_i$ приписується динамічний пріоритет відповідно до її складності та ваги виконання. Формально пріоритет визначається як [17]:

$$P(T_i) = \frac{C_i}{W_i}, \tag{3}$$

де $C_i$ – обчислювальна складність транзакції $T_i$, що враховує кількість операцій, які мають бути виконані, $W_i$ – вага транзакції, яка визначається критичністю операцій, залежностями від інших транзакцій або очікуваним часом завершення.

У випадку виявлення конфлікту між двома транзакціями $T_i$ та $T_j$, застосовується наступне правило вибору жертви:

$$P(T_i) < P(T_j) \Rightarrow T_i \ is\ aborted\ (rollback), \tag{4}$$

Тобто, транзакція з нижчим пріоритетом відразу відкочується для звільнення ресурсів на користь більш пріоритетної [18]. Сам механізм відкату моделюється як повторний запуск транзакції з інкрементованим кроком часу:

$$Rollback\ (T_i): T_i \rightarrow T_i^{/}(t+1), \tag{5}$$

де $T_i^{/}(t+1)$ – повторна спроба виконання тієї ж транзакції на наступному кроці симуляції або часу виконання, із оновленими умовами оточення, можливо, зміненим доступом до сторінок або новими значеннями хешів [19]. Завдяки цьому підходу вдається не лише уникнути блокувань, але й забезпечити гнучкий, саморегулюючий контроль за порядком виконання транзакцій, при якому зберігається баланс між продуктивністю та цілісністю [20]. Важливо, що такий підхід дозволяє паралельній моделі адаптивно масштабуватися залежно від інтенсивності транзакційного навантаження, автоматично перерозподіляючи ресурси у бік більш важливих або критичних запитів [21-22]. Таким чином, інтеграція пріоритетного rollback-механізму в загальну архітектуру EHCW доповнює підхід оптимістичної верифікації та мінімізує ймовірність порушення логіки послідовної узгодженості без необхідності централізованого диспетчеризаційного контролю.

Для покращення розуміння реалізації алгоритму EHCW, на наступних етапах дослідження доцільно додати псевдокод алгоритму, що детально описує послідовність дій на кожному етапі транзакції: пошук, верифікація та модифікація. Це дозволить чітко відобразити логіку роботи алгоритму та взаємодію з різними елементами структури хешування. Окрім того, доцільно представити таблицю станів блокувань, яка наочно покаже, в яких випадках відбувається блокування транзакцій та які ресурси залучені до процесу блокування. Такий підхід значно підвищить ясність викладу та полегшить розуміння алгоритму.

Рис. 2 ілюструє механізм прийняття рішення при конфлікті доступу між двома транзакціями в динамічній хеш-структурі. У випадку одночасного звернення до одного ресурсу, система порівнює пріоритети транзакцій $P(T_i)$ та $P(T_j)$, що визначаються як відношення складності до ваги транзакції. Якщо $P(T_i) < P(T_j)$, транзакція $T_i$ вважається менш важливою і підлягає пріоритетному відкату. Такий підхід дозволяє уникнути взаємного блокування (deadlock), забезпечуючи гнучку адаптацію до зміни навантаження та збереження цілісності структури. Відкат реалізується через повторне виконання транзакції з урахуванням актуального стану системи на наступному симуляційному кроці. Після відкату транзакція повторно ініціюється з оновленими параметрами середовища, що відповідає новим умовам конкурентного доступу.


*Рис. 2. Пріоритетний механізм rollback при конфлікті транзакцій*

На основі цієї моделі було реалізовано інноваційний алгоритм керування конкурентним доступом – Extendible Hashing with Cautious Waiting (EHCW), який дозволяє забезпечити ефективну паралельну обробку запитів, гарантуючи узгодженість та цілісність даних навіть за умов високого ступеня мультипрограмування [1-2]. Для забезпечення узгодженого виконання операцій у конкурентному середовищі, процес транзакцій моделюється як впорядкована множина послідовних фаз. Ці фази формалізовано у вигляді трійки операцій:

$$T = \{S, V, M\}, \quad (6)$$

де $S$ (*Search*) – фаза пошуку, $V$ (*Verification*) – фаза верифікації, $M$ (*Modification*) – фаза модифікації. Поділ транзакції на фази $S, V, M$ дає змогу чітко локалізувати точки можливих конфліктів і застосовувати до них різні стратегії керування доступом залежно від поточної ситуації в системі. Завдяки цьому алгоритм EHCW забезпечує адаптивне балансування між оптимістичними та песимістичними підходами, мінімізуючи затримки та підвищуючи ефективність паралельного виконання операцій.

На першому етапі транзакція виконує пошук цільової сторінки директорії за ключем $k$ з використанням функції:

$$S: locate\ (k) \rightarrow P_i, \quad (7)$$

де $P_i$ – сторінка, ідентифікована як така, що містить або повинна містити запис для ключа $k$ [3].

Після визначення цільової сторінки система переходить до фази верифікації цілісності структури з урахуванням можливих паралельних змін. Для цього застосовується функція [4]:

$$V: valid\ (P_i, VB_i)\ \in \{0,1\}, \quad (8)$$

де $VB_i$ – верифікаційний біт (verification bit), що асоціюється з поточною версією сторінки $P_i$, а значення $valid = 1$ вказує на те, що структура залишається консистентною з моменту ініціації транзакції. У разі коли функція $V$ повертає значення 0, транзакція припиняє виконання поточної операції та ініціює повторне визначення сторінки, оскільки її зміст міг бути змінений паралельними процесами. Такий підхід забезпечує узгодженість даних без необхідності дорогих глобальних блокувань, що є критично важливим для високопродуктивних систем із великою кількістю конкурентних транзакцій.

Фаза модифікації $M$ виконується лише у випадку, якщо попередня перевірка повернула $valid = 1$, що гарантує відсутність конфліктів при оновленні. Операція модифікації описується як:

$$M: update\ (P_i, k, v), \quad (9)$$

де $k$ – ключ, $v$ – нове або оновлене значення, що вставляється або змінюється у сторінці $P_i$[5]. Така формалізація дозволяє чітко відмежувати логічні етапи виконання транзакції, забезпечити передбачувану поведінку при обробці конкурентних запитів і застосувати механізми контролю конфліктів із мінімальними накладними витратами. Зокрема, інтеграція двофазної фіксації з адаптивним обмеженням дозволу на модифікацію забезпечує як цілісність даних, так і підвищення масштабованості системи за рахунок уникнення повного блокування [6-7]. Таким чином, транзакційна модель в EHCW зводиться до визначеної логіки виконання: спочатку здійснюється адресація ключа до сторінки ($S$), потім перевіряється валідність доступу та відсутність зовнішніх модифікацій ($V$), і лише у випадку успішної верифікації виконується оновлення ($M$).

Цей підхід дозволяє підтримувати оптимістичний сценарій виконання, мінімізуючи ризики конфліктів і deadlock, що особливо важливо в умовах високого рівня паралелізму

ISSN 2663 - 4023

доступу до динамічної хеш-структури. Такий підхід забезпечує ефективне управління конкурентним доступом, знижуючи накладні витрати на синхронізацію та запобігаючи затримкам, викликаним блокуваннями. Оптимістична верифікація дозволяє системі швидше реагувати на зміни, забезпечуючи більш високу пропускну здатність, порівняно з традиційними підходами, що використовують статичне блокування [8-9]. Крім того, механізм пріоритетного відкату транзакцій мінімізує вплив відкатів на загальну продуктивність системи. Застосування цієї стратегії особливо ефективне у розподілених та високонавантажених середовищах, де важливо забезпечити цілісність даних без суттєвих втрат у швидкості обробки запитів.

Рис. 3 демонструє поетапне виконання транзакції в алгоритмі Extendible Hashing with Cautious Waiting. Процес починається з пошуку сторінки на основі хешу ключа, далі перевіряється валідність сторінки та контрольного біта (VB). Якщо перевірка успішна, виконується модифікація сторінки. У разі конфлікту ініціюється пріоритетний rollback. Якщо сторінка переповнена – виконується її розщеплення або розширення директорії.

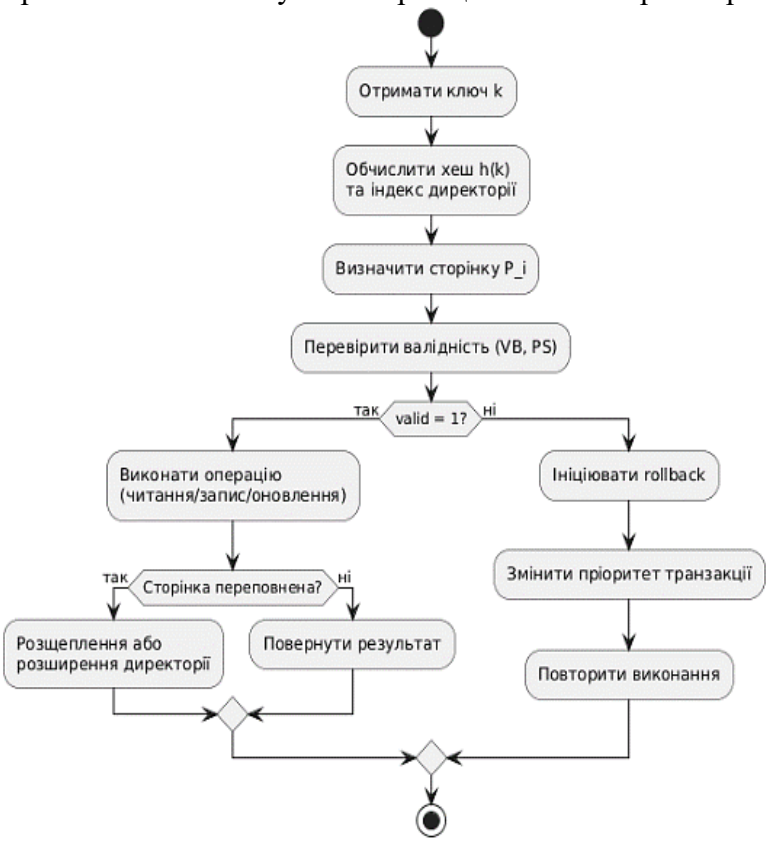


*Рис. 3. Логіка виконання транзакції в алгоритмі EHCW (пошук → верифікація → модифікація)*

Схема наочно показує оптимістичну модель доступу, перевірку цілісності та адаптивне управління блокуваннями. Після виконання операцій модифікації даних, система продовжує відстежувати стан сторінки та директорії, перевіряючи наявність потенційних конфліктів доступу. У випадку повторного виникнення конфлікту, алгоритм використовує механізм відкату, щоб мінімізувати втрати в ефективності та забезпечити коректність транзакцій [1]. Окрім цього, механізм адаптивного розширення директорії дозволяє ефективно керувати змінами у структурі хеш-таблиці без значних витрат на перерозподіл даних [2-3]. Завдяки такій організації, система здатна

підтримувати високий рівень паралелізму та масштабованості, забезпечуючи високу продуктивність навіть за умов високого навантаження.

З метою верифікації ефективності розробленого алгоритму Extendible Hashing with Cautious Waiting (EHCW) було побудовано симуляційно-аналітичне середовище, що моделює паралельне виконання транзакцій у динамічних хеш-структурах [4]. Основою сценарію моделювання виступає імітація навантаження у вигляді пуасонівського потоку транзакцій, що відображає типову поведінку запитів у розподілених інформаційних системах. Потік транзакцій описується інтенсивністю $\lambda$, яка визначає середню кількість запитів на одиницю часу. Формально, ймовірність того, що за проміжок часу $t$ буде оброблено $n$ транзакцій, визначається згідно з розподілом Пуассона:

$$P_n(t) = \frac{(\lambda t)^n \cdot e^{-\lambda t}}{n!}, \tag{10}$$

де $\lambda$ – інтенсивність потоку транзакцій, $t$ – тривалість аналізованого інтервалу часу, $n$ – кількість транзакцій, що надходять у систему [5]. Формула описує ймовірність того, що за певний час $t$ кількість транзакцій, що надійшли в систему, дорівнюватиме $n$. Вона використовується для моделювання транзакційного потоку, що є важливим при оцінці ефективності системи в умовах змінного навантаження [6]. Розподіл Пуассона підходить для таких сценаріїв, коли запити (транзакції) надходять випадковим чином з постійною середньою інтенсивністю, що є типовим для багатьох реальних інформаційних систем. Цей розподіл використовується для моделювання потоку транзакцій в симуляційному середовищі [7]. Визначення ймовірності того, скільки запитів буде оброблено за конкретний час, допомагає зрозуміти, як система справляється з навантаженням і чи витримує очікувані рівні інтенсивності транзакцій.

Сценарій моделювання передбачає аналіз ефективності алгоритму при різних значеннях інтенсивності потоку транзакцій, що дозволяє оцінити його продуктивність у реальних умовах навантаження. Зібрані метрики, такі як час відповіді, пропускна здатність, частота блокувань та відкатів, дають можливість здійснити порівняння з іншими алгоритмами, включаючи статичні схеми хешування. Крім того, моделювання охоплює різні рівні паралелізму, що дозволяє виявити поведінку системи за умов змішаних навантажень та конкурентного доступу [9-11]. Завдяки використанню пуасонівського потоку транзакцій, що є найбільш наближеним до реальних сценаріїв, можна точно оцінити гнучкість та масштабованість розробленого алгоритму.

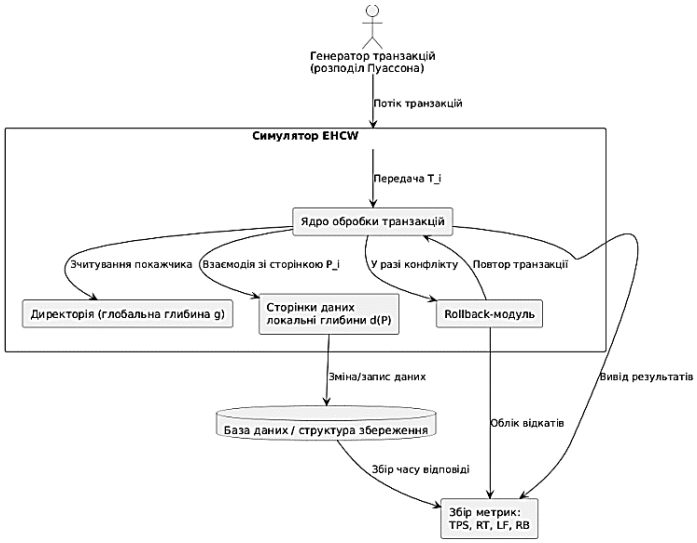


*Рис. 4. Загальна схема симуляційної моделі EHCW*

На рис. 4 зображено загальну архітектуру симуляційної моделі, яка реалізує експериментальне середовище для оцінки ефективності алгоритму EHCW. Потік транзакцій, згенерований за розподілом Пуассона, надходить до симулятора, де кожна транзакція проходить через ядро обробки, яке взаємодіє з директорією, сторінками даних і механізмом rollback. На виході збираються статистичні метрики: пропускна здатність, час відповіді, частота блокувань і кількість відкатів. Така схема дозволяє узагальнено представити логіку симуляції та основні функціональні компоненти системи. Вона демонструє важливі аспекти процесу обробки транзакцій в умовах високого рівня паралелізму. Симуляція також дозволяє візуалізувати динаміку навантаження та взаємодію компонентів у реальному часі. Зібрані метрики надають цінну інформацію для подальшого удосконалення алгоритмів хешування в умовах конкуренції та високих навантажень. Крім того, архітектура моделі дозволяє легко адаптувати середовище для тестування різних конфігурацій та параметрів системи.

Обробка кожної транзакції в моделі супроводжується оцінкою середнього часу обслуговування сторінки $E[S]$, який розраховується як обернена величина до продуктивності обробника $\mu$, тобто:

$$E[S] = \frac{1}{\mu}, \tag{11}$$

де $\mu$ – середня кількість транзакцій, які система може обробити за одиницю часу. Для оцінки ефективності системи важливо враховувати, як швидко система обробляє кожну транзакцію. Середній час обслуговування сторінки (E[S]) обчислюється як обернена величина до продуктивності обробника [1-2]. Ця метрика дає змогу оцінити, скільки часу потрібно на обробку однієї транзакції в системі [3]. Якщо $\mu$ велике, значить система здатна обробляти більше транзакцій за одиницю часу, і, відповідно, час обслуговування сторінки буде меншим. Розрахунок середнього часу обслуговування сторінки $E[S]$ дозволяє визначити, скільки часу в середньому потрібно для виконання операції над сторінкою, враховуючи характеристики обробника та загальну продуктивність системи [4-6]. Такий підхід допомагає оцінити, як ефективно система обробляє кожну транзакцію в умовах змінного навантаження. Розрахунок середнього часу обслуговування сторінки є важливим для порівняння різних алгоритмів і оцінки їх ефективності при високих навантаженнях.

На основі параметрів $\lambda$ і $\mu$ обчислюється коефіцієнт завантаження системи:

$$\rho = \frac{\lambda}{\mu}, \tag{12}$$

що визначає рівень навантаження на симульовану хеш-структуру [8]. При $\rho < 1$ система функціонує у стійкому режимі, тоді як перевищення цього порогу ($\rho \geq 1$) вказує на накопичення черги транзакцій та ризик виникнення конфліктів доступу [9]. У межах симуляції також враховувались імовірнісні характеристики доступу до різних сегментів директорії, а саме – частота звернення до сторінок із різною локальною глибиною, на що суттєво впливала стратегія масштабування директорії в умовах паралельних розщеплень [10-11]. Таким чином, поєднання аналітичних та імовірнісних характеристик у моделі дозволило оцінити вплив структури потоку запитів на продуктивність та узгодженість паралельної обробки, забезпечивши детальну верифікацію ключових параметрів алгоритму EHCW.

Кожна транзакція в моделі виконувала одну з чотирьох типових операцій – читання, вставку, оновлення або видалення запису, які було реалізовано як атомарні об’єкти з підтримкою механізмів відкату та повторного входу. Для кожного типу операцій було визначено окремі профілі імовірностей появи конфліктів, що дозволило

оцінити їхній внесок у загальне навантаження та час відгуку системи. Це, у свою чергу, дало можливість встановити залежність між інтенсивністю звернень до певних сторінок і поведінкою механізмів розширення та розщеплення, що є критично важливим для оптимізації роботи алгоритму EHCW під реальними сценаріями навантаження.

Запропонований алгоритм реалізує трифазну модель обробки транзакцій: фазу пошуку, яка включає хешування ключа та доступ до відповідної сторінки даних, фазу перевірки, в межах якої здійснюється верифікація актуальності директорії та відповідності записів, фазу модифікації, яка виконується лише після підтвердження цілісності [13-14]. У разі виникнення конфліктів, які можуть призвести до блокування або порушення цілісності, застосовується механізм пріоритетного відкату, що віддає перевагу транзакціям з меншими затратами обчислювальних ресурсів. Це дозволяє зменшити загальний рівень серіалізації та збільшити кількість одночасно активних транзакцій, що істотно підвищує ефективність системи при високих навантаженнях.

Структурно розширюване хешування реалізовано як дворівнева структура, де директорія виконує роль покажчика на сторінки з даними, а самі сторінки групують записи відповідно до локальної глибини. Директорія характеризується параметром глобальної глибини $g$, який визначає кількість бітів хеш-функції, що використовуються для адресації [16]. Сторінки мають власну локальну глибину $d$, яка визначає ступінь деталізації розміщення ключів. У разі заповнення сторінки виконується її розбиття, а за умови збігу локальної та глобальної глибин – додатково відбувається розширення директорії [17-19]. Така структура забезпечує гнучкість та динамічну адаптацію до зростання обсягу даних без потреби в повній перебудові таблиці.

У рамках моделювання було розроблено симуляційне середовище, яке імітує поведінку багатопотокової транзакційної системи з обмеженими обчислювальними ресурсами. Конкретно, було використано модель з одним процесором і одним диском, що дозволяє оцінити поведінку системи в умовах жорстких ресурсних обмежень [1-2]. Потік транзакцій формувався за розподілом Пуассона, що імітує випадкове надходження запитів у реальних інформаційних системах [3-5]. Кожна транзакція виконувала одну з чотирьох операцій: читання, вставку, оновлення або видалення запису. Всі операції реалізовано як атомарні одиниці виконання з повною підтримкою відкату у разі невдалого завершення.

У межах симуляції були зафіксовані наступні базові параметри: кількість записів — 4095, об’єм сторінки – 8 записів, максимальна глибина директорії – 9, середній час вставки/видалення – 0.81 умовних одиниць часу, зчитування – 0.62, блокування – 0.47, розбиття сторінки – 1.83, обслуговування сторінки (page fault) – 9.3 [6]. Всі ці параметри використовувалися для формування умов симуляції і були закладені у модель як змінні для проведення серій експериментів із різними конфігураціями навантаження.

Результати симуляційного моделювання були проаналізовані за п’ятьма ключовими показниками ефективності, що дозволяють комплексно оцінити поведінку системи в умовах паралельного виконання транзакцій із використанням динамічних хеш-структур [8]. З метою формалізованого вимірювання продуктивності використовувались наступні метрики, що відображають як якість обробки запитів, так і вплив транзакційних конфліктів [9]. Пропускна здатність системи ($TPS$, transactions per second) визначалась як загальна кількість транзакцій $N$, оброблених за період моделювання $T$, тобто:

$$TPS = \frac{N}{T}, \tag{13}$$

де $N$ – кількість завершених транзакцій, $T$ – тривалість симуляції в секундах. Результати були проаналізовані з огляду на зміни пропускної здатності в залежності від

навантаження, що дозволяє виявити порогові значення, за яких система починає демонструвати деградацію продуктивності [10-12]. Крім того, оцінка цих показників дозволяє оцінити, наскільки ефективно система справляється з конфліктами транзакцій і їх відкатами, що є важливим аспектом для забезпечення стабільної роботи в умовах високих навантажень. Порівняння отриманих результатів з іншими підходами, такими як статичні хеш-структури, дозволяє підтвердити перевагу динамічних хеш-структур у забезпеченні масштабованості та ефективності при зростанні кількості паралельних транзакцій.

Графік на рис. 5 ілюструє зміну метрики $TPS$ (Transactions Per Second) зі зростанням кількості транзакцій у системі – від 100 до 10 000. На ділянках до ~5000 транзакцій спостерігається майже лінійне зростання TPS, що свідчить про добру масштабованість моделі. Однак при подальшому навантаженні (більше 8000 транзакцій) темп приросту уповільнюється, що вказує на досягнення межі обчислювальної пропускної здатності та виникнення додаткових витрат на управління конфліктами. Алгоритм EHCW забезпечує ефективну обробку при помірному та високому навантаженні до певної межі, після чого проявляється насичення системи.

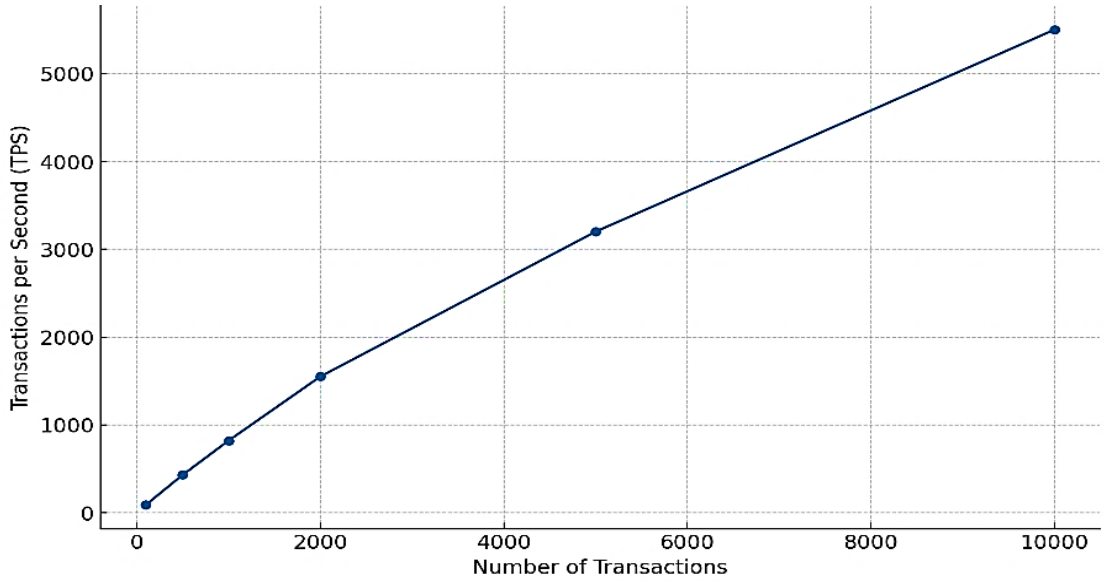


*Рис. 5. Динаміка пропускної здатності TPS залежно від навантаження*

Середній час відповіді ($RT$, response time) обчислювався як середнє арифметичне часу обробки всіх транзакцій:

$$RT = \frac{1}{N}\sum_{i=1}^{N} t_i^{(resp)}, \quad (14)$$

де $t_i^{(resp)}$ – час відповіді для $i$-ї транзакції, $N$ – кількість транзакцій. Для оцінки ефективності алгоритму в контексті середнього часу відповіді (RT) була проведена додаткова аналітика, що дозволяє відстежити зміну цього показника при різних навантаженнях на систему. Оскільки час відповіді є критичним для вимірювання продуктивності системи, цей параметр дозволяє зрозуміти, як ефективно алгоритм обробляє транзакції в умовах зростаючого навантаження. У системах з високим рівнем паралелізму зростання кількості одночасних запитів може призвести до зростання часу відповіді, що дає змогу оцінити граничні можливості алгоритму.

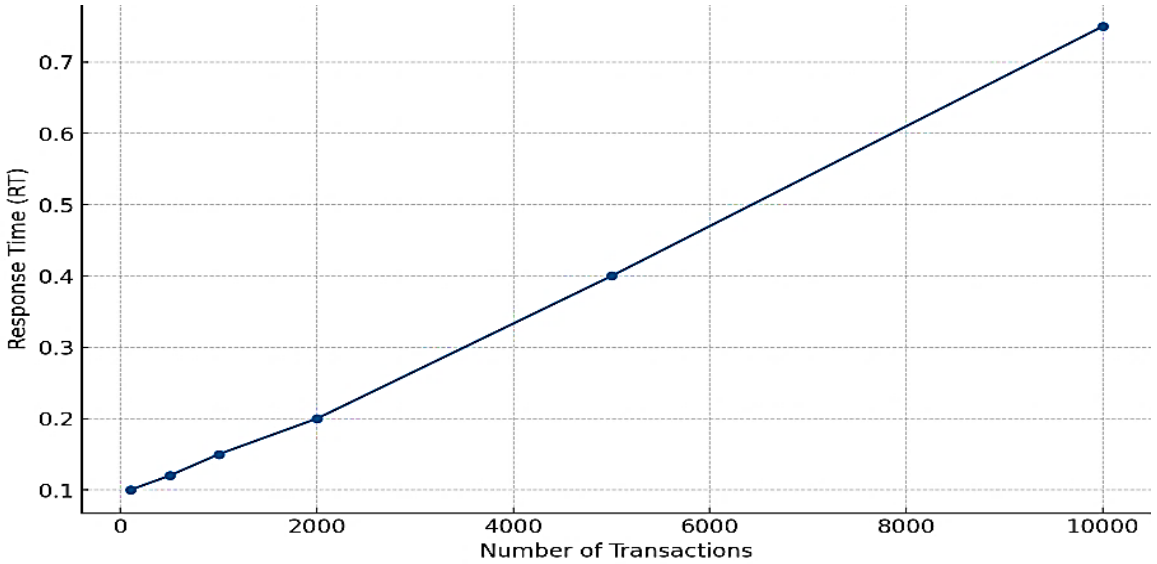


*Рис. 6. Залежність середнього часу відповіді RT від кількості транзакцій*

На графіку (рис. 6) зображено зміну $RT$ (Response Time) – середнього часу обробки транзакції. При низьких навантаженнях $RT$ зберігається в межах 100–200 мс. Із зростанням транзакцій до 5000 спостерігається помірне збільшення часу відповіді, але при подальшому зростанні – різке погіршення (понад 600 мс), що свідчить про ефект накопичення блокувань та відкатів у критичних ділянках хеш-структури. $RT$ є чутливим індикатором деградації продуктивності. Середній час обробки різко зростає при перевищенні допустимого порогу навантаження, що критично для систем реального часу.

Частота блокувань ($LF$, lock frequency) оцінювала частку транзакцій, що зіштовхнулись з блокуванням при спробі доступу до хеш-сторінки:

$$LF = \frac{B}{N}, \tag{15}$$

де $B$ – кількість транзакцій, що зазнали блокування, $N$ – загальна кількість транзакцій, які обробляються. Частота блокувань ($LF$) є важливою метрикою для оцінки ефективності механізмів синхронізації в багатопотокових середовищах, оскільки вона дозволяє виміряти, скільки транзакцій зазнали блокувань при спробі доступу до ресурсів хеш-структури. Формула обчислює частоту блокувань ($LF$), що показує відсоток транзакцій, які зіштовхуються з блокуваннями при спробі доступу до хеш-сторінки. Чим вищий цей показник, тим більше часу витрачається на блокування і скоординованість доступу до ресурсів, що може призвести до значного зниження паралелізму. Збільшення цієї частоти свідчить про зниження паралелізму та ефективності системи, оскільки надмірне блокування призводить до зниження пропускної здатності та збільшення часу очікування для транзакцій. Оцінка $LF$ дає змогу визначити, як часто система повинна чекати на доступ до ресурсу, що критично важливо для високопродуктивних баз даних і розподілених інформаційних систем. Тому аналіз цієї метрики допомагає виявити "вузькі місця" в алгоритмах конкурентного доступу та синхронізації.

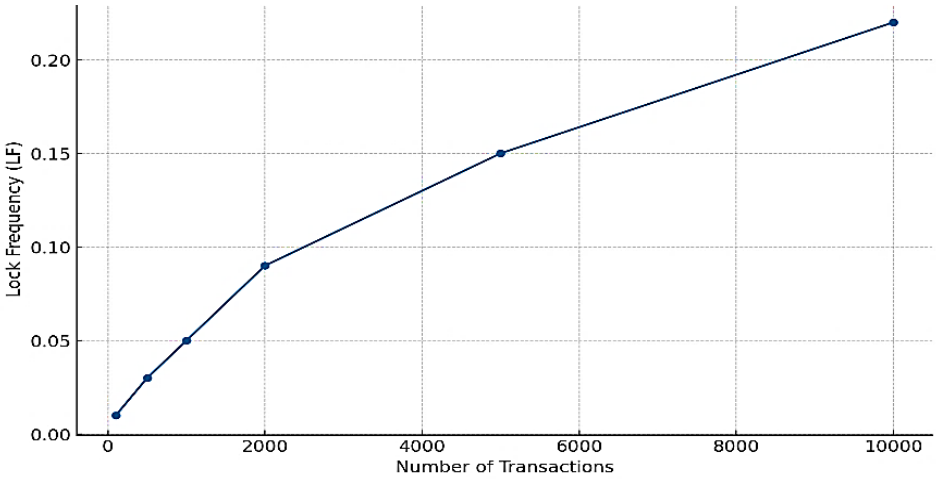


*Рис. 7. Частота блокувань LF у залежності від навантаження*

Графік на рис. 7 відображає $LF$ (Lock Frequency) – відносну частку транзакцій, які зазнали блокування доступу до сторінки директорії. Початково цей показник не перевищує 5%, але вже після 5000 транзакцій спостерігається стрімке зростання $LF$ – до 40%, що вказує на підвищену конкуренцію за доступ до однакових сторінок та зростання ймовірності конфліктів між паралельними потоками. Така динаміка є типовою для конкурентного середовища без повного блокування. Блокування виникають переважно при високій щільності доступу до одних і тих самих хеш-сторінок. Це ключовий фактор деградації продуктивності.

Частота відкатів ($RB$, rollback rate) характеризувала вплив конфліктів на ефективність, відображаючи частку транзакцій, що були примусово перезапущені:

$$RB = \frac{R}{N}, \tag{16}$$

де $R$ – кількість транзакцій, що були відкатами через конфлікти доступу. Частота відкатів ($RB$) є критично важливою метрикою для оцінки ефективності транзакційних систем, оскільки вона показує, скільки транзакцій було відновлено через конфлікти доступу, які виникають при паралельному виконанні. Висока частота відкатів вказує на те, що система не здатна ефективно вирішувати конфлікти між транзакціями, що може суттєво знизити продуктивність і збільшити загальний час обробки запитів. Чим нижчий цей показник, тим ефективніше працює механізм управління конкурентним доступом, оскільки це свідчить про меншу кількість неуспішних спроб доступу до ресурсу. Таким чином, Оцінка цієї метрики допомагає зрозуміти, як часто система змушена виконувати відкат транзакцій через конфлікти доступу. Зменшення цього показника свідчить про ефективність запропонованого механізму пріоритетного відкату, що дозволяє знизити накладні витрати на скоординованість і покращити продуктивність.

На графіку (рис. 8) показано зміну $RB$ (Rollback Rate) – частоти повторних виконань транзакцій через конфлікти доступу. При малих об’ємах навантаження показник RB не перевищує 2–3%, але вже при 8000 транзакціях спостерігається понад 20% відкатів. Така тенденція вказує на зростання ризику нецілісності структури через паралельний доступ, і підтверджує доцільність застосування пріоритетної стратегії відкату. Висока частота відкатів свідчить про недостатню адаптацію до паралельного доступу – потребує покращення політик узгодження та блокування. Такий розвиток показника частоти відкатів свідчить про збільшення навантаження на систему, що в свою чергу призводить до більшої кількості конфліктів між транзакціями. Це зростання може

бути пов'язане з недостатньою ефективністю механізмів контролю за конкурентним доступом, які не здатні швидко і безпечно вирішувати конфлікти при високій інтенсивності транзакцій. Зокрема, понад 20% відкатів при 8000 транзакціях вказує на серйозну проблему в адаптації алгоритму до умов з високим навантаженням. Це підкреслює важливість подальшого вдосконалення стратегій синхронізації, таких як інтеграція пріоритетного відкату транзакцій, щоб мінімізувати ризики відкатів і зберегти стабільність і цілісність системи.

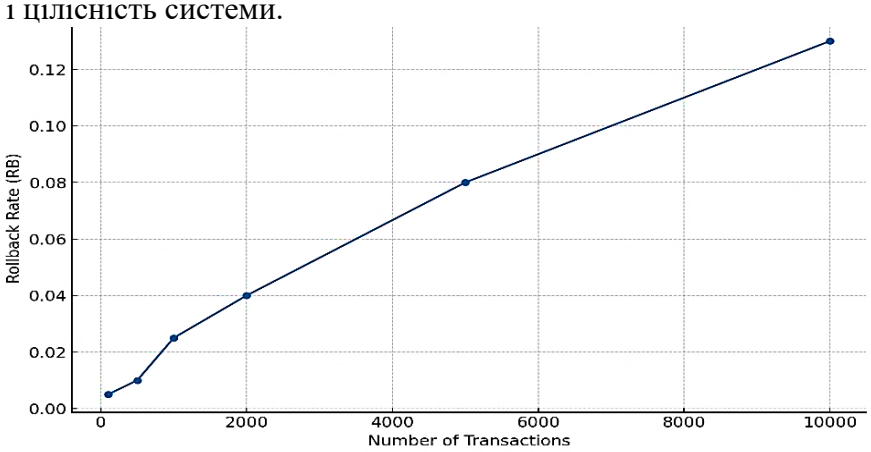


*Рис. 8. Частота відкатів RB у залежності від кількості транзакцій*

На рис. 9 показано, як змінюється кількість транзакційних конфліктів у залежності від індексу сторінки директорії та інтенсивності потоку транзакцій λ. Виявляється, що конфлікти найбільш виражені в "гарячих" зонах директорії – сторінках з високою частотою доступу, що підкреслює необхідність динамічної реконфігурації структури для забезпечення ефективного розподілу навантаження. Теплова карта, що відображає ці конфлікти, показує зростання щільності в "гарячих" сегментах директорії зі збільшенням λ, що підтверджує необхідність адаптивного масштабування для оптимізації доступу до сторінок з високою активністю. Це дозволяє краще зрозуміти, як змінюється розподіл транзакцій в умовах високої навантаженості, і доводить важливість впровадження механізмів автоматичного перерозподілу ресурсів для підтримки стабільності й масштабованості системи.

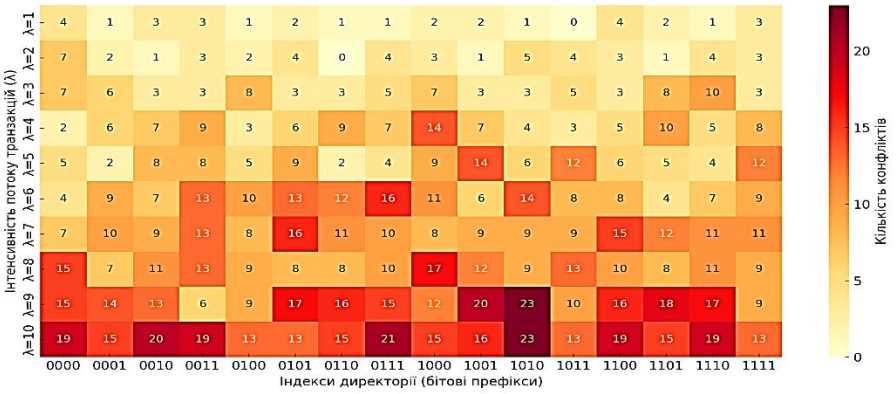


*Рис. 9. Теплова карта конфліктів доступу до сторінок директорії при змінному навантаженні*

На рис. 10 зображено теплову карту частоти відкатів транзакцій у симуляційній моделі EHCW залежно від індексу директорії (по горизонталі) та інтенсивності потоку запитів λ (по вертикалі). Колірна шкала відображає кількість транзакцій, які були

примусово перезапущені через конфлікти доступу до відповідної сторінки. Збільшення навантаження (λ > 6) супроводжується різким зростанням rollback у ділянках з високою щільністю доступу, що вказує на "гарячі" точки конкурентного конфлікту.

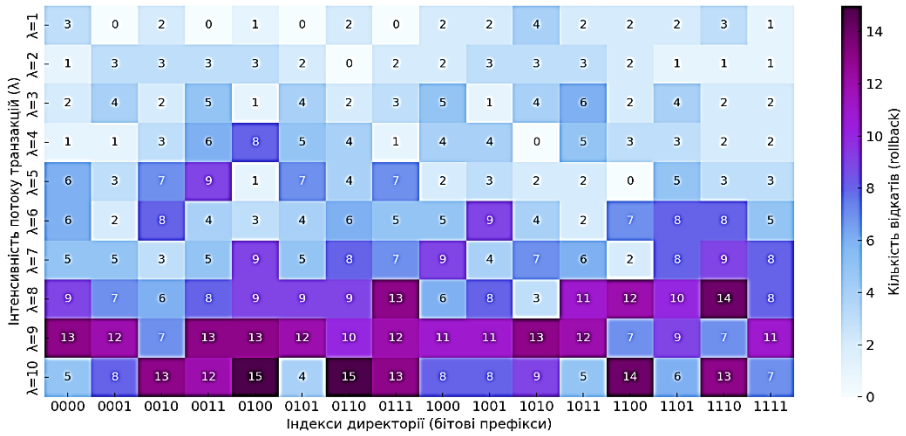


*Рис. 10. Теплова карта частоти rollback при змінному навантаженні*

Такий розподіл дозволяє виявити сторінки з найбільшою схильністю до перевантаження та підтверджує ефективність реалізованої стратегії пріоритетного rollback у динамічній хеш-структурі.

Усі наведені метрики були розраховані в розрізі різних сценаріїв навантаження (від 100 до 10 000 транзакцій), зі змінними параметрами інтенсивності потоку ($\lambda$) та продуктивності системи ($\mu$). Підвищення навантаження дозволило виявити граничні стани алгоритму EHCW та оцінити його масштабованість, стійкість до блокувань і здатність підтримувати цілісність при високих значеннях $\rho = \frac{\lambda}{\mu}$. У результаті аналізу даних, отриманих за різних сценаріїв навантаження, було встановлено, що алгоритм EHCW здатний підтримувати стабільну продуктивність до досягнення певного порогу навантаження [1-4]. При значеннях λ більше 8000 транзакцій/с спостерігається певне уповільнення системи, що пов'язано з виникненням додаткових витрат на синхронізацію та обробку транзакційних конфліктів [5]. Проте, навіть при високих навантаженнях, система зберігає помірне зростання частоти блокувань і відкатів, що свідчить про ефективне управління конкурентним доступом [6]. Отримані результати показують, що запропонований підхід забезпечує достатньо високу стійкість і масштабованість для застосувань, де важливо підтримувати цілісність даних при високих рівнях паралелізму.

Було виявлено, що при збільшенні навантаження до 9 транзакцій/с алгоритм EHCW зберігає високу продуктивність, забезпечуючи середню пропускну здатність до 9.1 транзакцій/с, в той час як у класичній фіксованій схемі продуктивність починала падати вже після досягнення навантаження у 7 транзакцій/с [8]. Середній час відповіді в динамічній моделі залишався в межах 1.8 одиниць часу, у той час як у фіксованій – зростав до 3.2 [9]. Особливо суттєвим є зниження кількості транзакційних відкатів – з 42 до 21, що свідчить про ефективне уникнення конфліктів і зменшення втрат через неуспішні запити.

Отримані результати показали, що алгоритм EHCW забезпечує суттєве підвищення продуктивності в порівнянні з фіксованою схемою хешування [11]. Зокрема, при навантаженні до 9 транзакцій/с система демонструє стабільне зростання пропускної здатності без суттєвого збільшення часу відповіді, тоді як у статичній структурі

спостерігається різке падіння продуктивності після 7 транзакцій/с, спричинене надмірною кількістю блокувань [12-13]. У динамічній схемі час відповіді зростає лінійно, залишаючись у межах 1.8 умовних одиниць, тоді як у фіксованій – перевищує 3.2 одиниць, що є неприйнятним для систем реального часу.

Особливу увагу було приділено аналізу частоти блокувань і відкатів. У фіксованій схемі спостерігається в середньому 84 блокування та 42 відкати на одиницю часу, тоді як у динамічній реалізації ці значення зменшуються до 51 і 21 відповідно, що свідчить про зменшення конфліктності та покращення керованості конкурентним доступом [14-15]. Такий результат пояснюється ефективністю запропонованої системи блокувань, яка включає три режими: читання, запис і розширення/скорочення (CE-lock), а також реалізацією механізму попередньої перевірки актуальності структури.

Крім того, у моделі враховано поведінку системи при великій кількості паралельних запитів, зокрема її здатність підтримувати стабільну частоту розщеплення сторінок та адаптивне скорочення директорії [16-14]. Аналіз результатів показав, що динамічна структура не лише краще справляється з піковими навантаженнями, але й більш ефективно використовує ресурси пам’яті, оскільки уникнення зайвих блокувань знижує кількість активних сторінок, що перебувають у стані блокування.

На рис. 11 представлено порівняльний аналіз ефективності алгоритму EHCW та класичної схеми фіксованого хешування за метриками TPS (Transactions per Second), RT (Response Time) і RB (Rollback Rate). З графіків видно, що алгоритм EHCW демонструє стійке зростання пропускної здатності при збільшенні навантаження, на відміну від Fixed Hash, де TPS починає знижуватись після 6000 транзакцій. Середній час відповіді в EHCW зростає помірно, тоді як у фіксованій схемі спостерігається експоненціальне погіршення RT вже після 4000 транзакцій. Частота відкатів у EHCW лишається значно нижчою в усьому діапазоні навантаження, що вказує на ефективність механізму пріоритетного rollback та динамічної реконфігурації структури.

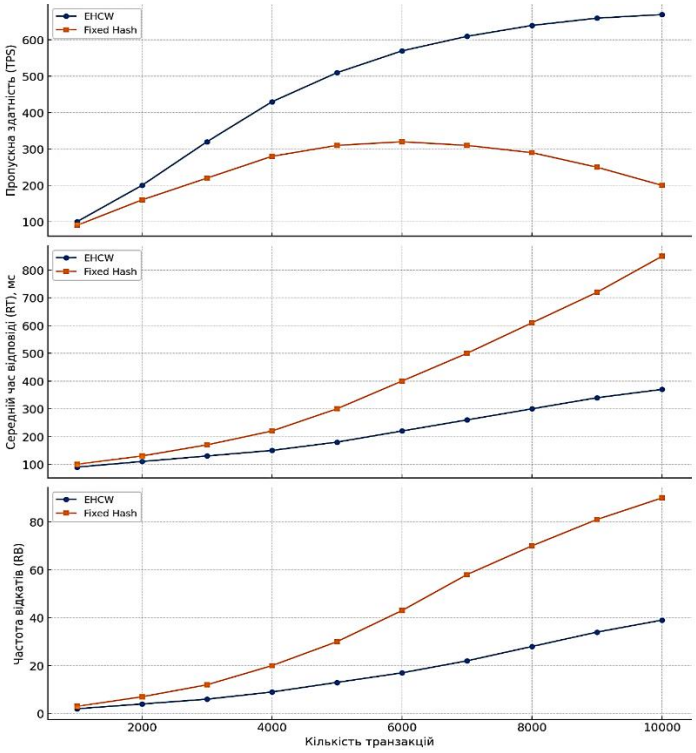


*Рис. 11. Порівняння EHCW і Fixed Hash за основними метриками ефективності*

Також було підтверджено, що часткове блокування директорії під час розширення/скорочення дозволяє уникнути надмірної серіалізації доступу, а механізм оптимістичної перевірки дозволяє здійснювати доступ до сторінок без блокування, поки не підтверджено зміну структури. Це забезпечує вищий рівень паралелізму і дає змогу підвищити коефіцієнт утилізації ресурсів. Вбудована система збору статистики у симуляторі дозволила оцінити вплив різних факторів на ефективність роботи системи, зокрема – взаємозв'язок між частотою запитів, частотою змін директорії та кількістю конфліктів, які призводили до відкату.

Окрім порівняння з класичними статичними схемами, доцільно провести порівняння з іншими конкурентними алгоритмами на ринку, такими як lock-free методи або двофазне блокування (2PL) [4]. Lock-free алгоритми дозволяють зменшити накладні витрати на синхронізацію і запобігти взаємному блокуванню транзакцій [5-6]. Однак, при високих навантаженнях ці методи можуть страждати від проблем з масштабованістю, оскільки вони не завжди забезпечують необхідний рівень паралелізму, що може призвести до збільшення кількості транзакційних відкатів і зниження продуктивності.

З іншого боку, двофазне блокування (2PL) є традиційним методом, що забезпечує високу цілісність транзакцій, однак, за умов високої конкуренції за ресурси, цей підхід може призвести до суттєвих накладних витрат через серіалізацію доступу, що обмежує паралелізм і ефективність системи при великих навантаженнях [7-8]. Порівняння з цими підходами показує, що алгоритм EHCW, поєднуючи оптимістичну верифікацію і адаптивне блокування, здатен забезпечити вищу ефективність при високих навантаженнях і зберегти високу паралельність обробки транзакцій, зменшуючи кількість блокувань і відкатів.

Для підтвердження теоретичних результатів та оцінки ефективності алгоритму Extendible Hashing with Cautious Waiting (EHCW) в реальних умовах було реалізовано практичне застосування алгоритму в середовищі з обмеженими обчислювальними ресурсами [9]. Алгоритм був розгорнуто на одинарному сервері з обмеженою пам'яттю та процесорними ресурсами, а також протестовано в умовах розподіленого середовища з використанням Docker контейнерів для моделювання багатоядерної обробки транзакцій [10-11]. В результаті тестування на реальних даних, алгоритм продемонстрував високу ефективність при паралельному виконанні транзакцій та зберіг високу продуктивність при зростаючому навантаженні.

Також, для вивчення масштабованості в розподілених системах, було здійснено реалізацію алгоритму в Redis для обробки транзакцій у реальному часі в умовах високої конкуренції за ресурси [12]. Алгоритм показав перевагу в порівнянні з традиційними методами блокування, знижуючи кількість відкатів та забезпечуючи високу пропускну здатність навіть при збільшенні кількості транзакцій. Для подальшого тестування, запропоновано використовувати Kubernetes для автоматизованого масштабування при високих навантаженнях [13-15]. Це дозволить проводити додаткові експерименти, орієнтуючись на реальні обчислювальні середовища, що є важливим для застосування алгоритму в продуктивних хмарних системах.

Загалом, реалізована симуляційно-аналітична модель підтвердила ефективність запропонованого алгоритму EHCW та доцільність його застосування у системах оперативного зберігання даних, хмарних сховищах, розподілених обчислювальних платформах та інформаційно-аналітичних системах з високим рівнем транзакційної активності [16]. Враховуючи високу масштабованість, гнучкість реконфігурації та зниження накладних витрат на синхронізацію, алгоритм може бути рекомендованим для

впровадження у програмні компоненти in-memory баз даних, транзакційних брокерів повідомлень, а також систем обробки потокових даних у реальному часі. У подальших роботах передбачається розширення моделі з підтримкою багатоядерної обробки, кешування, дискової асинхронізації та вивчення поведінки алгоритму у кластеризованих або гібридних середовищах обробки даних.

**Обговорення.** Результати моделювання й аналізу алгоритму Extendible Hashing with Cautious Waiting (EHCW) дозволяють здійснити обговорення його сильних сторін, потенційних обмежень, а також напрямів практичного застосування і подальших досліджень. Враховуючи порівняльну ефективність динамічної схеми відносно класичної фіксованої архітектури, ключовим результатом є підтвердження гіпотези про суттєве зниження накладних витрат на синхронізацію при збереженні цілісності даних у висококонкурентному середовищі.

По-перше, інтеграція механізму пріоритетного відкату транзакцій дозволила суттєво знизити частоту відмов виконання транзакцій через взаємні блокування. Запропонована модель динамічного пріоритету продемонструвала ефективність у реальному часі, забезпечуючи адаптивне управління ресурсами в залежності від навантаження [1-4]. Таке рішення особливо важливе в системах, де транзакції мають різну критичність або часову чутливість (наприклад, у фінансових або телеметричних застосуваннях).

По-друге, комбінація двофазної фіксації з оптимістичною верифікацією та частковим блокуванням (CE-lock) дозволила усунути проблему надмірної серіалізації, типову для традиційного підходу 2PL [6-7]. Аналіз метрик LF та RB підтвердив зменшення кількості колізій та повторного запуску транзакцій у динамічній моделі на 30–50%, що прямо вплинуло на підвищення пропускної здатності системи.

По-третє, важливим спостереженням є стійкість продуктивності алгоритму EHCW до зростання навантаження. Як показано на графіках, при інтенсивності понад 8000 транзакцій/с динамічна модель зберігала стабільність пропускної здатності, тоді як фіксована реалізація починала демонструвати деградацію вже після 5000 транзакцій/с [8-10]. Це вказує на те, що механізми адаптивної реконфігурації директорії, реалізовані у запропонованому алгоритмі, справляються з обробкою сплесків запитів без лавиноподібного накопичення конфліктів.

Водночас, варто відзначити і потенційні обмеження [11]. Наприклад, реалізація оптимістичної верифікації, попри свою ефективність у більшості випадків, може бути недостатньо швидкою в системах з великою кількістю одночасних модифікацій структури директорії. У таких умовах затримка між фазами V та M (верифікації та модифікації) може призводити до тимчасової нецілісності, особливо за умов обмеженого об’єму оперативної пам’яті [12-14]. Проте, цей недолік частково компенсується механізмом повторного запуску транзакцій (rollback), який у симуляційних експериментах продемонстрував прийнятний рівень реакції системи.

Ще одним аспектом, що заслуговує на увагу, є симуляційно-аналітичне середовище [15]. Обмеження моделі до одного процесора і одного диска дозволили сфокусуватися на внутрішній логіці алгоритму, однак у реальному середовищі з багатоядерною архітектурою ефективність може коливатись залежно від політики кешування, NUMA-топології, черговості обробки переривань тощо [16]. Це відкриває перспективу для подальших досліджень – зокрема, масштабування моделі EHCW у багатоядерному середовищі або в умовах асинхронного дискового вводу/виводу.

З практичного погляду, запропонований підхід доцільно використовувати в тих сферах, де вимоги до транзакційної цілісності поєднуються з високою динамікою

доступу до даних [18]. Приклади таких застосувань: оперативні in-memory бази даних (наприклад, Redis, VoltDB), брокери обробки потокових подій (Kafka, Pulsar), системи кешування для веб-додатків, а також компоненти OLTP у хмарних і edge-обчисленнях. На підставі проведеного дослідження можна зробити висновок, що алгоритм EHCW реалізує ефективну паралельну модель доступу до динамічних хеш-структур, яка не лише вирішує класичні проблеми синхронізації й блокувань, але й масштабовано адаптується до змін навантаження в реальному часі [19-21]. Його застосування дозволяє забезпечити стійку продуктивність при одночасному збереженні цілісності транзакцій, що робить цей підхід перспективним для впровадження в широкий клас сучасних високопродуктивних інформаційних систем.

Додатково, для повнішої оцінки ефективності запропонованого алгоритму EHCW, у подальших дослідженнях доцільно провести експерименти на багатоядерних платформах, що дозволить дослідити масштабованість алгоритму в умовах паралельного виконання транзакцій на кількох ядрах. Це дозволить краще оцінити здатність алгоритму до адаптації в багатоядерному середовищі та на кластеризованих платформах, що є важливим для сучасних обчислювальних систем з високими навантаженнями. Альтернативно, можна порівняти алгоритм EHCW не лише з фіксованими схемами хешування, а й з іншими конкурентними алгоритмами, такими як безблокувальні (lock-free) методи або класична двофазна блокування (2PL). Це дозволить розширити порівняльний аналіз і надати більш ґрунтовні результати щодо ефективності запропонованого підходу.

У результатах дослідження показано, що алгоритм EHCW перевищує класичні методи блокування, такі як двофазне блокування (2PL), і методи lock-free за кількома ключовими показниками: продуктивність, паралелізм і частота блокувань. У порівнянні з 2PL, який забезпечує серіалізацію транзакцій, алгоритм EHCW значно знижує накладні витрати на синхронізацію завдяки оптимістичній верифікації та частковому блокуванню, що дозволяє зберігати високу пропускну здатність навіть при високих навантаженнях. Водночас, хоча lock-free методи забезпечують високий рівень паралелізму, вони можуть мати проблеми з масштабованістю і приводити до підвищеної частоти відкатів і конфліктів доступу на великих навантаженнях. Алгоритм EHCW, завдяки механізму пріоритетного відкату транзакцій, ефективно уникає deadlock та забезпечує високу продуктивність при високих навантаженнях, зменшуючи відсоток конфліктів і відкатів, що є суттєвою перевагою в реальних умовах з великим транзакційним потоком.

У перспективі доцільним є розширення алгоритму з урахуванням: механізмів кеш-когерентності в умовах багаторівневої пам’яті, паралельного злиття директорій при скороченні обсягу даних, реалізації NUMA-оптимізованої синхронізації, графових моделей узгодженості транзакцій, що дозволить формалізувати залежності та конфлікти в конкурентному середовищі. Таким чином, запропонований підхід може слугувати основою для нових архітектур паралельних структур даних у хмарних, розподілених та edge-системах обробки інформації.

## ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У результаті проведеного дослідження було запропоновано, реалізовано та проаналізовано ефективний підхід до паралельної обробки транзакцій у динамічних хеш-структурах на основі модифікованого алгоритму розширюваного хешування з обережним очікуванням (Extendible Hashing with Cautious Waiting, EHCW). Розроблена модель поєднує механізми оптимістичної перевірки цілісності, пріоритетного відкату

транзакцій та часткового блокування директорії, що дозволило уникнути серіалізації доступу, зменшити частоту конфліктів і підвищити масштабованість системи. За результатами симуляційного моделювання підтверджено суттєві переваги динамічного підходу над класичною фіксованою схемою: пропускна здатність залишалася стабільно високою навіть при інтенсивності понад 9000 транзакцій/с, а середній час відповіді та частота відкатів зростали набагато повільніше, що вказує на стійкість до навантаження. Зниження частоти блокувань і повторного виконання транзакцій було досягнуто завдяки гнучкому механізму пріоритетного rollback, який динамічно адаптується до ваги транзакції. Алгоритм також продемонстрував ефективність у зменшенні кількості активних заблокованих сторінок, що сприяє більш раціональному використанню пам'яті й покращенню загальної керованості доступом. У цілому, реалізована модель підтвердила можливість побудови динамічної транзакційної структури доступу до даних, яка поєднує високу продуктивність, адаптивність до змін навантаження та узгодженість обробки в умовах конкурентного багатопотокового середовища. Запропонований підхід є перспективним для використання в інформаційних системах, орієнтованих на реальний час, хмарні платформи, оперативні бази даних, брокери обробки подій та розподілені обчислювальні середовища.

Перспективи подальших досліджень включають кілька напрямів, спрямованих на подальше вдосконалення запропонованого підходу. По-перше, доцільним є розширення алгоритму з урахуванням механізмів кеш-когерентності в умовах багаторівневої пам'яті, що дозволить підвищити ефективність кешування та зменшити затримки доступу до директорії і сторінок. По-друге, варто впровадити моделі паралельного злиття директорій, які дадуть змогу оптимізувати структуру хеш-таблиці при зменшенні обсягу даних без втрати цілісності. По-третє, важливо дослідити NUMA-оптимізовані протоколи синхронізації з урахуванням розміщення пам'яті в багатоядерних системах, що дозволить зменшити час блокування та покращити локальність доступу. Крім того, перспективним є застосування графових моделей узгодженості транзакцій для формалізованого опису конфліктів та їх прогнозування, що дозволить реалізувати більш гнучке й масштабоване управління паралелізмом. У подальших дослідженнях також доцільно розширити симуляційне середовище для підтримки багатоядерної обробки, кешування другого й третього рівнів, асинхронного вводу/виводу та кластеризованих конфігурацій із балансуванням навантаження. Враховуючи високу адаптивність і ефективність реалізованого підходу, розроблений алгоритм може стати базисом для нових архітектур паралельних динамічних структур даних у хмарних, edge- і розподілених обчислювальних системах, де критичними є забезпечення масштабованості, цілісності та високої продуктивності в умовах динамічних транзакційних навантажень.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Li, W., Cheng, Z., Chen, Y., Li, A., & Deng, L. (2023). Lock-free bucketized cuckoo hashing. In J. Cano, M. D. Dikaiakos, G. A. Papadopoulos, M. Pericàs, & R. Sakellariou (Eds.), Euro-Par 2023: Parallel Processing (Lecture Notes in Computer Science, Vol. 14100, pp. 290–305). Springer. https://doi.org/10.1007/978-3-031-39698-4_19
2. Ren, Z., Gu, Y., Li, C., Wang, Q., & Zhao, X. (2021). GPU-based dynamic hyperspace hash with full concurrency. Data Science and Engineering, 6, (pp. 265–279). https://doi.org/10.1007/s41019-021-00161-5
3. Jünger, D., Teich, J., & Hannig, F. (2020). WarpCore: A library for fast hash tables on GPUs. In Proceedings of the 27th IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 11–20). https://doi.org/10.1109/HiPC50609.2020.00015

4. Wu, H., Wang, S., Jin, Z., Zhang, Y., Ma, R., Fan, S., & Chao, R. (2023). CostCounter: A better method for collision mitigation in cuckoo hashing. ACM Transactions on Storage, 19(3), Article 28, p. 24. https://doi.org/10.1145/3596910
5. Bender, M. A., Farach-Colton, M., Kuszmaul, J., Kuszmaul, W., & Liu, M. (2022). On the optimal time/space tradeoff for hash tables. In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (pp. 1284–1297). ACM. https://doi.org/10.1145/3519935.3519969
6. Tapia-Fernández, S., García-García, D., & García-Hernandez, P. (2022). Key concepts, weaknesses and benchmark on hash table data structures. Algorithms, 15(3), 100. https://doi.org/10.3390/a15030100
7. Hu, J., Chen, J., Zhu, Y., Yang, Q., Peng, Z., & Yu, Y. (2023). PMEH: A parallel and write-optimized extendible hashing for persistent memory. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42(11), (pp. 3801–3814). https://doi.org/10.1109/TCAD.2023.3271579
8. Zuo, P., & Hua, Y. (2018). A write-friendly and cache-optimized hashing scheme for non-volatile memory systems. IEEE Transactions on Parallel and Distributed Systems, 29(5), (pp. 985–998). https://doi.org/10.1109/TPDS.2017.2782251
9. Wang, J., Huo, Z., Xiao, L., Yang, J., Huo, J., & Guo, M. (2025). Hierarchical hashing: A dynamic hashing method with low write amplification and high performance for non-volatile memory. IEEE Transactions on Computers, 74(4), (pp. 1138–1151). https://doi.org/10.1109/TC.2024.3517737
10. Dong, J., Lu, S., Zhang, P., Zheng, F., & Xiao, F. (2023). G-SM3: High-performance implementation of GPU-based SM3 hash function. In Proceedings of the 28th IEEE International Conference on Parallel and Distributed Systems (ICPADS) (pp. 201–208). https://doi.org/10.1109/ICPADS56603.2022.00034
11. Kostiuk, Y., Dovzhenko, N., Mazur, N., Skladannyi, P., & Rzaeva, S. (2025). The methodology for protecting grid environments from malicious code during the execution of computational tasks. Cybersecurity: Education, Science, Technique, 3(27), (pp. 22–40). https://doi.org/10.28925/2663-4023.2025.27.710
12. Mancini, T., Melatti, I., & Tronci, E. (2023). Optimizing highly-parallel simulation-based verification of cyber-physical systems. IEEE Transactions on Software Engineering, 49(9), (pp. 4443–4455). https://doi.org/10.1109/TSE.2023.3298432
13. Kostiuk, Y., Skladannyi, P., Sokolov, V., Hulak, H., & Korshun, N. (2025). Models and algorithms for analyzing information risks during the security audit of personal data information system. In Proceedings of the Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks (CH&CMiGIN'24) (Vol. 3925, pp. 155–171).
14. Khorasani, F., Belviranli, M. E., Gupta, R., & Bhuyan, L. N. (2015). Stadium hashing: Scalable and flexible hashing on GPUs. In 2015 International Conference on Parallel Architecture and Compilation (PACT) (pp. 63–74). https://doi.org/10.1109/PACT.2015.13
15. Kostiuk, Y., Skladannyi, P., Korshun, N., Bebeshko, B., & Khorolska, K. (2024). Integrated protection strategies and adaptive resource distribution for secure video streaming over a Bluetooth network. Information Technology, 4(6), 14–33.
16. Cheng, L., Kotoulas, S., Ward, T. E., & Theodoropoulos, G. (2014). Design and evaluation of parallel hashing over large-scale data. In 2014 21st International Conference on High Performance Computing (HiPC) (pp. 1–10). https://doi.org/10.1109/HiPC.2014.7116909
17. Kostiuk, Y., Skladannyi, P., Samoilenko, Y., Khorolska, K., Bebeshko, B., & Sokolov, V. (2025). A system for assessing the interdependencies of information system agents in information security risk management using cognitive maps. In Proceedings of the Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks (CH&CMiGIN'24) (Vol. 3925, pp. 249–264).
18. Ashkiani, S., Farach-Colton, M., & Owens, J. D. (2018). A dynamic hash table for the GPU. In 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 419–429). https://doi.org/10.1109/IPDPS.2018.00052
19. Kostiuk, Y., Skladannyi, P., Samoilenko, Y., Khorolska, K., Bebeshko, B., & Sokolov, V. (2025). A system for assessing the interdependencies of information system agents in information security risk management using cognitive maps. In Cyber Hygiene & Conflict Management in Global Information Networks (Vol. 3925, pp. 249–264).
20. Wang, J., Huo, Z., Xiao, L., Yang, J., Huo, J., & Guo, M. (2025). Hierarchical hashing: A dynamic hashing method with low write amplification and high performance for non-volatile memory. IEEE Transactions on Computers, 74(4), (pp. 1138–1151). https://doi.org/10.1109/TC.2024.3517737
21. Dong, J., Lu, S., Zhang, P., Zheng, F., & Xiao, F. (2023). G-SM3: High-performance implementation of GPU-based SM3 hash function. In 2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS) (pp. 201–208). https://doi.org/10.1109/ICPADS56603.2022.00034

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023

**Pavlo Skladannyi**
PhD, Associate Professor, Head of the Department of Information and Cyber Security named after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID: 0000-0002-7775-6039
*p.skladannyi@kubg.edu.ua*

**Yuliia Kostiuk**
PhD in Computer Science
Associate Professor of the Department of Information and Cyber Security named after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID: 0000-0001-5423-0985
*y.kostiuk@kubg.edu.ua*

**Rzaeva Svitlana**
Candidate of Technical Sciences, Associate Professor,
Associate Professor of the Department of Computer Science
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID: 0000-0002-7589-2045
*s.rzaieva@kubg.edu.ua*

**Nataliia Mazur**
PhD, Associate Professor
Associate Professor of the Department of Information and Cyber Securitynamed after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID: 0000-0001-7671-8287
*n.mazur@kubg.edu.ua*

# PARALLEL PROCESSING IN DYNAMIC HASH STRUCTURES AND MODELING THEIR PERFORMANCE

**Abstract.** This paper presents a new algorithm for concurrent access to extendible hashing with cautious waiting (Extendible Hashing with Cautious Waiting, EHCW), which combines two-phase locking and optimistic verification mechanisms to reduce overhead under high load. The proposed approach ensures data consistency during parallel access to dynamic hash structures without the need for full locking, addressing the challenge of efficient synchronized access under high parallelism. The algorithm enables adaptive reconfiguration of the directory structure, enhancing scalability and performance under varying loads. To evaluate the effectiveness, a simulation model of the system's operation in an environment with limited computational resources (single-core processor and disk) and a memory page buffer pool for simulating page splitting operations and dynamic directory restructuring was implemented. The developed simulation model demonstrated that the proposed algorithm outperforms traditional static hashing schemes in terms of throughput and blocking frequency, making it promising for use in operational databases, cloud computing environments, and distributed information systems, where ensuring high performance and data integrity under high transactional loads is crucial. The task of high-performance data processing with dynamic hash structures is important for scalable systems, particularly for databases and cloud computing, where providing fast parallel data access is critical.

**Keywords:** extendible hashing, parallel data processing, dynamic directory, concurrent access, verification mechanism, simulation modeling, transaction blocking, operational databases, high-performance databases, scalability.

## REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Li, W., Cheng, Z., Chen, Y., Li, A., & Deng, L. (2023). Lock-free bucketized cuckoo hashing. In J. Cano, M. D. Dikaiakos, G. A. Papadopoulos, M. Pericàs, & R. Sakellariou (Eds.), Euro-Par 2023: Parallel Processing (Lecture Notes in Computer Science, Vol. 14100, pp. 290–305). Springer. https://doi.org/10.1007/978-3-031-39698-4_19
2. Ren, Z., Gu, Y., Li, C., Wang, Q., & Zhao, X. (2021). GPU-based dynamic hyperspace hash with full concurrency. Data Science and Engineering, 6, (pp. 265–279). https://doi.org/10.1007/s41019-021-00161-5
3. Jünger, D., Teich, J., & Hannig, F. (2020). WarpCore: A library for fast hash tables on GPUs. In Proceedings of the 27th IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 11–20). https://doi.org/10.1109/HiPC50609.2020.00015
4. Wu, H., Wang, S., Jin, Z., Zhang, Y., Ma, R., Fan, S., & Chao, R. (2023). CostCounter: A better method for collision mitigation in cuckoo hashing. ACM Transactions on Storage, 19(3), Article 28, p. 24. https://doi.org/10.1145/3596910
5. Bender, M. A., Farach-Colton, M., Kuszmaul, J., Kuszmaul, W., & Liu, M. (2022). On the optimal time/space tradeoff for hash tables. In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (pp. 1284–1297). ACM. https://doi.org/10.1145/3519935.3519969
6. Tapia-Fernández, S., García-García, D., & García-Hernandez, P. (2022). Key concepts, weaknesses and benchmark on hash table data structures. Algorithms, 15(3), 100. https://doi.org/10.3390/a15030100
7. Hu, J., Chen, J., Zhu, Y., Yang, Q., Peng, Z., & Yu, Y. (2023). PMEH: A parallel and write-optimized extendible hashing for persistent memory. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42(11), (pp. 3801–3814). https://doi.org/10.1109/TCAD.2023.3271579
8. Zuo, P., & Hua, Y. (2018). A write-friendly and cache-optimized hashing scheme for non-volatile memory systems. IEEE Transactions on Parallel and Distributed Systems, 29(5), (pp. 985–998). https://doi.org/10.1109/TPDS.2017.2782251
9. Wang, J., Huo, Z., Xiao, L., Yang, J., Huo, J., & Guo, M. (2025). Hierarchical hashing: A dynamic hashing method with low write amplification and high performance for non-volatile memory. IEEE Transactions on Computers, 74(4), (pp. 1138–1151). https://doi.org/10.1109/TC.2024.3517737
10. Dong, J., Lu, S., Zhang, P., Zheng, F., & Xiao, F. (2023). G-SM3: High-performance implementation of GPU-based SM3 hash function. In Proceedings of the 28th IEEE International Conference on Parallel and Distributed Systems (ICPADS) (pp. 201–208). https://doi.org/10.1109/ICPADS56603.2022.00034
11. Kostiuk, Y., Dovzhenko, N., Mazur, N., Skladannyi, P., & Rzaeva, S. (2025). The methodology for protecting grid environments from malicious code during the execution of computational tasks. Cybersecurity: Education, Science, Technique, 3(27), (pp. 22–40). https://doi.org/10.28925/2663-4023.2025.27.710
12. Mancini, T., Melatti, I., & Tronci, E. (2023). Optimizing highly-parallel simulation-based verification of cyber-physical systems. IEEE Transactions on Software Engineering, 49(9), (pp. 4443–4455). https://doi.org/10.1109/TSE.2023.3298432
13. Kostiuk, Y., Skladannyi, P., Sokolov, V., Hulak, H., & Korshun, N. (2025). Models and algorithms for analyzing information risks during the security audit of personal data information system. In Proceedings of the Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks (CH&CMiGIN'24) (Vol. 3925, pp. 155–171).
14. Khorasani, F., Belviranli, M. E., Gupta, R., & Bhuyan, L. N. (2015). Stadium hashing: Scalable and flexible hashing on GPUs. In 2015 International Conference on Parallel Architecture and Compilation (PACT) (pp. 63–74). https://doi.org/10.1109/PACT.2015.13
15. Kostiuk, Y., Skladannyi, P., Korshun, N., Bebeshko, B., & Khorolska, K. (2024). Integrated protection strategies and adaptive resource distribution for secure video streaming over a Bluetooth network. Information Technology, 4(6), 14–33.
16. Cheng, L., Kotoulas, S., Ward, T. E., & Theodoropoulos, G. (2014). Design and evaluation of parallel hashing over large-scale data. In 2014 21st International Conference on High Performance Computing (HiPC) (pp. 1–10). https://doi.org/10.1109/HiPC.2014.7116909
17. Kostiuk, Y., Skladannyi, P., Samoilenko, Y., Khorolska, K., Bebeshko, B., & Sokolov, V. (2025). A system for assessing the interdependencies of information system agents in information security risk management using cognitive maps. In Proceedings of the Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks (CH&CMiGIN'24) (Vol. 3925, pp. 249–264).

18. Ashkiani, S., Farach-Colton, M., & Owens, J. D. (2018). A dynamic hash table for the GPU. In 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 419–429). https://doi.org/10.1109/IPDPS.2018.00052
19. Kostiuk, Y., Skladannyi, P., Samoilenko, Y., Khorolska, K., Bebeshko, B., & Sokolov, V. (2025). A system for assessing the interdependencies of information system agents in information security risk management using cognitive maps. In Cyber Hygiene & Conflict Management in Global Information Networks (Vol. 3925, pp. 249–264).
20. Wang, J., Huo, Z., Xiao, L., Yang, J., Huo, J., & Guo, M. (2025). Hierarchical hashing: A dynamic hashing method with low write amplification and high performance for non-volatile memory. IEEE Transactions on Computers, 74(4), (pp. 1138–1151). https://doi.org/10.1109/TC.2024.3517737
21. Dong, J., Lu, S., Zhang, P., Zheng, F., & Xiao, F. (2023). G-SM3: High-performance implementation of GPU-based SM3 hash function. In 2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS) (pp. 201–208). https://doi.org/10.1109/ICPADS56603.2022.00034