https://doi.org/10.31891/2219-9365-2026-85-16
УДК 004.415

ЛУЖЕЦЬКИЙ Володимир
Вінницький національний технічний університет
https://orcid.org/0000-0001-7466-7738
lva.kzi2002@gmail.com
ЗАХАРЧЕНКО Сергій
Вінницький національний технічний університет
https://orcid.org/0000-0003-3977-2908
lva.kzi2002@gmail.com
КИСЮК Дмитро
Вінницький національний технічний університет
https://orcid.org/0009-0003-8514-2007
kneimad@gmail.com

МЕТОД БАЙТ-ОРІЄНТОВАНОГО ГЕШУВАННЯ З ПІДВИЩЕНИМ РІВНЕМ ДИФУЗІЇ ТА СПЕЦІАЛІЗОВАНИЙ MH-ПРОЦЕСОР ДЛЯ ПРИВАТНИХ ХМАРНИХ ІНФРАСТРУКТУР

У роботі запропоновано метод байт-орієнтованого гешування з підвищеним рівнем дифузії, орієнтований на потокову обробку даних у приватних (відомчих) хмарних інфраструктурах з жорсткими обмеженнями щодо латентності та апаратних ресурсів. Метод призначений для виявлення випадкових та нецільових модифікацій даних, забезпечення коректності дедуплікації та індексації блоків у розподілених сховищах. Робота не ставить за мету досягнення криптографічної стійкості проти адаптивного зловмисника, але забезпечує високий рівень дифузії для інженерних задач цілісності та індексації. Метод формує два статистичних масиви: K[v] — частоти входжень байтів, S[v] — позиційні характеристики, та вводить додатковий етап змішування g(K,S) перед редукцією C(·) до геш-значення фіксованої довжини.

На основі методу розроблено спеціалізований MH-процесор з мікроархітектурною структурою PI–PC–PM–DO. Наведено функціональну та структурну моделі, оцінку апаратних ресурсів та експериментальні показники продуктивності для CPU/FPGA/ASIC реалізацій. До основних внесків роботи належать: формалізація байт-орієнтованого потокового методу з підвищеною дифузією, інженерна мікроархітектура MH-процесора для детермінованої апаратної реалізації, експериментальна оцінка throughput, latency, cycles/hash та позиціонування методу як інженерного гешу для задач цілісності/дедуплікації.

Ключові слова: байт-орієнтоване гешування, дифузія, цілісність даних, дедуплікація, апаратний прискорювач, приватна хмара, хмарні сервіси, безпека програмних систем.

LUZHETSKYI Volodymyr, ZAKHARCHENKO Serhii, KYSIUK Dmytro
Vinnytsia National Technical University

BYTE-ORIENTED HASHING METHOD WITH ENHANCED DIFFUSION AND A SPECIALIZED MN PROCESSOR FOR PRIVATE CLOUD INFRASTRUCTURES

This paper proposes a byte-oriented hashing method with an enhanced diffusion level, designed for streaming data processing in private (departmental) cloud infrastructures operating under strict latency and hardware resource constraints. The method is intended for detecting accidental and non-targeted data modifications, ensuring reliable deduplication and accurate block indexing in distributed storage environments. The proposed approach does not aim to achieve full cryptographic resistance against adaptive adversaries; instead, it provides a high level of diffusion tailored for engineering tasks related to data integrity verification and indexing. The method constructs two statistical arrays: K[v], representing byte occurrence frequencies, and S[v], capturing positional characteristics. An additional mixing stage g(K, S) is introduced prior to the reduction function C(·), which produces a fixed-length hash value. Based on this method, a specialized MN-processor with a PI–PC–PM–DO microarchitectural structure has been developed. The paper presents both functional and structural models of the processor, evaluates hardware resource utilization, and provides experimental performance results for CPU, FPGA, and ASIC implementations.

The main contributions of the study include: the formalization of a byte-oriented streaming method with enhanced diffusion; the design of an engineering-grade MN-processor microarchitecture for deterministic hardware implementation; experimental evaluation of throughput, latency, and cycles per hash; and positioning of the method as an engineering hash function for integrity verification and deduplication tasks in constrained cloud environments.

Keywords: byte-oriented hashing, diffusion, data integrity, deduplication, hardware accelerator, private cloud, cloud services, software system security.

Стаття надійшла до редакції / Received 10.01.2026
Прийнята до друку / Accepted 14.02.2026
Опубліковано / Published 05.03.2026

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

У приватних хмарних інфраструктурах, зокрема у відомчих та військових сегментах, операції контролю цілісності, індексації об'єктів і дедуплікації виконуються над значними потоками даних у режимі,

наближеному до реального часу. Такі задачі виникають під час реплікації, резервного копіювання, синхронізації вузлів, журналювання подій та формування індексів об'єктів. У цих сценаріях вузли зберігання та периферійні сервери часто працюють в умовах обмежених обчислювальних і енергетичних ресурсів, що висуває підвищені вимоги до ефективності алгоритмів обробки даних.

Класичні криптографічні геш-функції сімейств SHA-2 та SHA-3 забезпечують високий рівень криптографічної стійкості та формально доведені властивості односторонності і стійкості до колізій. Проте їх багатораундова структура, складні нелінійні перетворення та значна кількість арифметичних операцій призводять до підвищеної латентності, складності апаратної реалізації та збільшеного енергоспоживання.

Для задач дедуплікації, індексації та технічного контролю цілісності в приватних хмарах такі властивості не завжди є визначальними, натомість критичними стають інші характеристики: лінійна обчислювальна складність O(L), детермінована та прогнозована латентність, регулярність мікроархітектури, можливість глибокої конвеєризації та ефективної апаратної реалізації у вигляді спеціалізованого прискорювача.

У цій роботі запропоновано байт-орієнтований метод гешування з підвищеним рівнем дифузії, орієнтований на потокову обробку даних у приватних хмарних інфраструктурах. Запропонований підхід не позиціонується як альтернатива стандартизованим криптографічним геш-функціям для задач електронного підпису або формального криптографічного захисту, а формує окремий клас інженерних потокових методів, оптимізованих для сервісів контролю цілісності та дедуплікації. Метод побудовано на формуванні статистичних характеристик байтового потоку з подальшим етапом змішування та регулярної редукції.

Такий підхід дозволяє інтегрувати сервіс гешування в багатошарову архітектуру приватної хмари та реалізувати його у вигляді спеціалізованого МН-процесора з конвеєрною організацією обробки, що зменшує навантаження на центральні процесори вузлів і підвищує енергоефективність системи в цілому.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Мета роботи — розробити метод гешування, що працює в потоковому режимі з лінійною складністю O(L) від довжини повідомлення L; має підвищений рівень дифузії (суттєва зміна геш-значення при малих змінах вхідних даних); придатний до регулярної апаратної реалізації; допускає інтеграцію у сервіси перевірки цілісності та дедуплікації приватної хмари.

Місце запропонованого рішення у приватній хмарній інфраструктурі

На рис. 1 наведено рівні інтеграції сервісу гешування та апаратного прискорювача у стек приватної хмари.

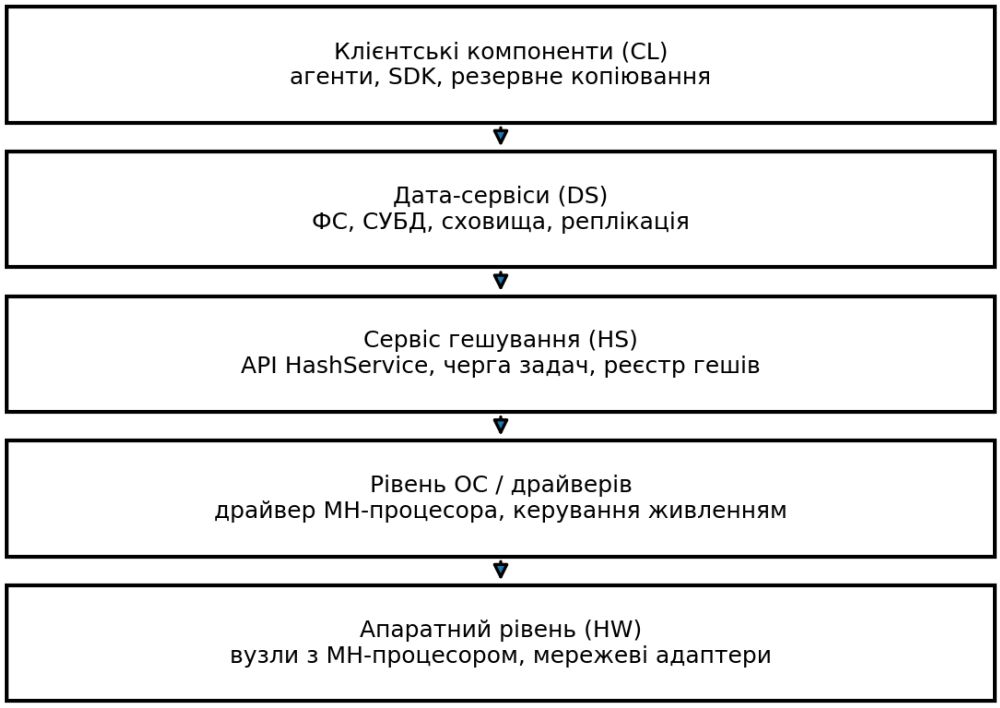

Рис. 1. Рівні інтеграції сервісу гешування та МН-процесора у приватну хмару

Формалізація методу байт-орієнтованого гешування з підвищеним рівнем дифузії

Нехай вхідне повідомлення задано як послідовність байтів

$$M = (m_1, m_2, \dots, m_L), m_i \in \mathrm{B} = \{0, \dots, 255\}.$$

Обробка здійснюється в однопрохідному потоковому режимі. У процесі обробки формується дві статистичні характеристики, що відображають частотну та позиційну структуру повідомлення.

Частотна характеристика

Для кожного значення байта $v \in$ Ввизначається кількість його входжень у повідомленні:

$$K[v] = \sum_{i=1}^{L} \mathbf{1}_{\{m_i = v\}}, v \in \text{B}, \quad (1)$$

де $\mathbf{1}_{\{.\}}$— індикаторна функція.

Вектор $K = (K[0], K[1], \dots, K[255])$ формує частотний профіль повідомлення.

Позиційна характеристика

Для кожного $v \in$ B додатково визначається позиційна характеристика:

$$S[v] = \sum_{i=1}^{L} i \cdot \mathbf{1}_{\{m_i = v\}}, v \in \text{B}. \quad (2)$$

Вектор $S = (S[0], S[1], \dots, S[255])$акумулює позиційно зважену інформацію про розташування байтів. Таким чином, пара (K, S)формує статистичний опис повідомлення у просторі розмірності 512.

В апаратній реалізації елементи $K[v]$зберігаються у 32-бітних лічильниках, а $S[v]$— у 64-бітних акумуляторах, що забезпечує відсутність переповнення для практичних довжин потоків.

Обидва варіанти використовують лише операції, придатні до регулярної апаратної реалізації (XOR, додавання за модулем 2^n, циклічні зсуви, опційно множення).

Застосування певної операції до цих двох масивів:

$$\text{H} = g(\text{K}, \text{S}). \quad (3)$$

До вхідних масивів K та S застосовується один з варіантів функції g:

$$g = \{g_0, g_1, \dots, g_{255}\}$$

1) дописування коду s_n до коду k_n gn(kn, sn) = $h_n^* = k_n \,||\, s_n$;
2) звичайне арифметичне множення 32-бітних кодів на 64-бітні коди gn(kn, sn) = $h_n^* = k_n \cdot s_n$.

$$\text{H} = g(\text{K}, \text{S}) = (h_0^*, h_1^*, \dots, h_{255}^*),$$

Масив H складається з 256 елементів довжиною 96 біт (12 байтів):

$$h_n^* = \left(b_{n,11}, b_{n,10}, \dots, b_{n,0}\right), (n = 0 \div 255).$$

Редукційна функція ущільнення

Після формування масиву Hзастосовується редукційна функція ущільнення

$$h = C\,(\text{H}). \quad (4)$$

Функція ущільнення C передбачає виконання таких дій.

Масив H перетворюється у масив $\text{H}^{(256)}$ з 256 однобайтних елементів:

$$\text{H}^{(256)} = \left(h_0^{(256)}, h_1^{(256)}, \dots, h_{255}^{(256)}\right),$$

де

$$h_n^{(256)} = \left(\sum_{i=0}^{11} b_{n,i}\right) mod\ 2^8. \quad (5)$$

Далі, масив $\text{H}^{(256)}$перетворюється у масив $\text{H}^{(128)}$аналогічно як у методі 1, і так далі до досягнення розміру масиву в 32 байти.

$$\text{H}^{(128)} \rightarrow \text{H}^{(64)} \rightarrow \text{H}^{(32)}.$$

Результатом обчислень є $h = \text{H}^{(32)}$.

Така структура еквівалентна бінарному дереву редукції, що:

1) забезпечує регулярність мікроархітектури,
2) допускає конвеєризацію,
3) дозволяє реалізувати паралельне згортання на FPGA/ASIC.

Алгоритм (псевдокод)

Вхід: потік байтів M, довжина L
Вихід: геш-значення h (32 байти)

1) ініціалізувати K[0..255]=0, S[0..255]=0
2) для i=1..L: v = m_i; K[v]←K[v]+1; S[v]←S[v]+i
3) для v=0..255: H[v]←g(K[v],S[v])
4) h←C(H) (ітеративна редукція до 32 байтів)
5) повернути h

СПЕЦІАЛІЗОВАНИЙ МН-ПРОЦЕСОР: ФУНКЦІОНАЛЬНА МОДЕЛЬ

Для зменшення навантаження на центральні процесори вузлів приватної хмари запропоновано спеціалізований МН-процесор (модуль прискорення хешування), що реалізує описаний алгоритм у режимі потокової обробки. Функціональна модель (рис. 2) відображає взаємодію блоків інтерфейсу приймання (PI), обчислювального контуру (PC), моніторингу/перевірки (PM) та доступу до індексу об'єктів (DO).

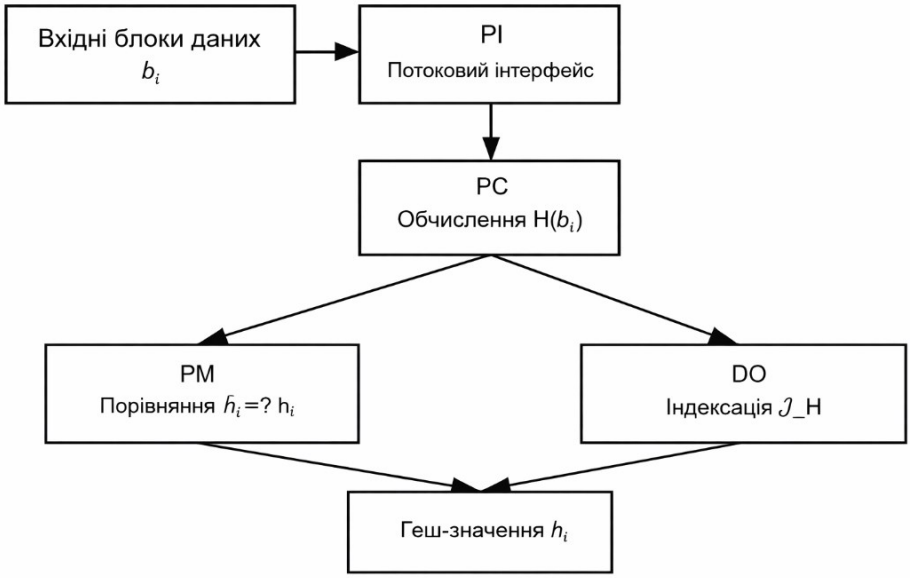

Рис. 2 Функціональна схема використання МН-процесора у сервісах цілісності та дедуплікації

Структурна схема апаратної реалізації

Структурна схема МН-процесора (рис. 3) орієнтована на регулярні операції: декодування байта та оновлення K/S (read-modify-write), модуль змішування g(K,S) з параметризованими зсувами та опційним множенням, редукційне дерево C(·) для формування 256-бітного результату. Керування здійснюється скінченною автоматною логікою (FSM), а обмін з хост-системою — через DMA/FIFO.

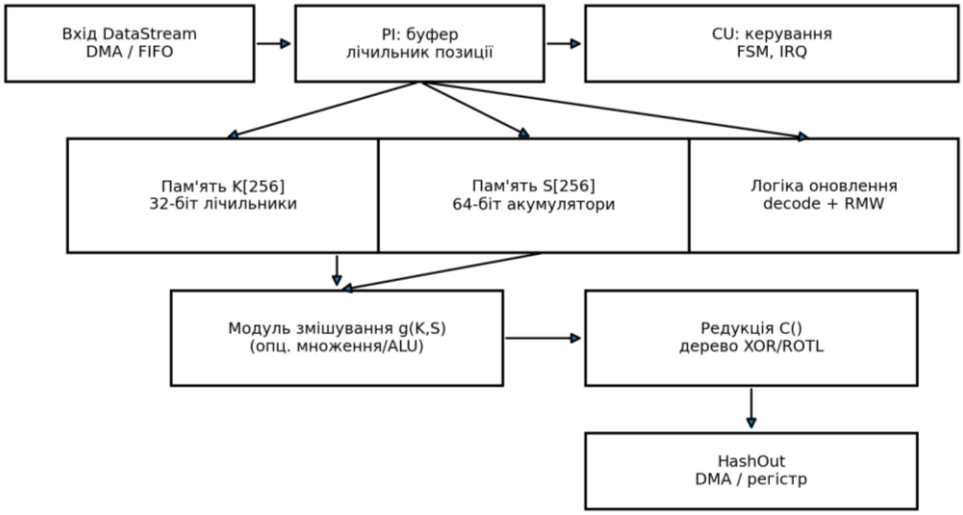

Рис. 3 Узагальнена структурна схема МН-процесора для режиму підвищеної дифузії

Оцінка апаратних ресурсів

Узагальнені мінімальні ресурси пам'яті для зберігання проміжних характеристик наведено у таблиці 1:

Таблиця 1

Оцінка буферів/пам'яті для реалізації методу

Буфер/масив	Розмірність	Розрядність	Обсяг
K[256]	256 елементів	32 біт	8 192 біт (1 024 байти)
S[256]	256 елементів	64 біти	16 384 біт (2 048 байтів)
H[256] (опц.)	256 елементів	96 біт	24 576 біт (3 072 байти)
Разом (K+S)	–	–	24 576 біт (3 072 байти)

Оцінка продуктивності

Експериментальні/розрахункові оцінки швидкодії для реалізацій на CPU, FPGA та ASIC наведено на рис. 4–5 і в табл. 2–3. Для апаратних платформ показано порядок величин пропускної здатності та латентності обчислення геш-значення у режимі підвищеної дифузії.

Отримані результати демонструють:

1) CPU — 177 МБ/с,
2) FPGA — 100 МБ/с,
3) ASIC (оцінка) — 400 МБ/с.

Найвищу пропускну здатність забезпечує спеціалізована ASIC-реалізація, що більш ніж у 2,2 рази перевищує показник CPU та у 4 рази — FPGA. Такий результат пояснюється відсутністю накладних витрат універсальної архітектури, глибокою конвеєризацією та оптимізацією логіки змішування на рівні вентильного опису.

CPU демонструє збалансований показник продуктивності завдяки використанню широких регістрів та SIMD-інструкцій, однак залишається обмеженим архітектурними особливостями загального призначення (керування пам'яттю, багатозадачність, кеш-ієрархія).

FPGA-реалізація забезпечує нижчу пропускну здатність у порівнянні з CPU, що пов'язано з наступним:

1) обмеженням частоти тактування,
2) використанням універсальних логічних блоків,
3) накладними витратами маршрутизації сигналів.

Водночас FPGA залишається гнучкою платформою для прототипування та адаптації алгоритму.

Таким чином, у задачах високошвидкісної дедуплікації та потокової індексації блоків у приватних хмарних інфраструктурах доцільною є ASIC-реалізація, тоді як CPU та FPGA можуть використовуватися на етапах тестування або в середовищах з помірним навантаженням.

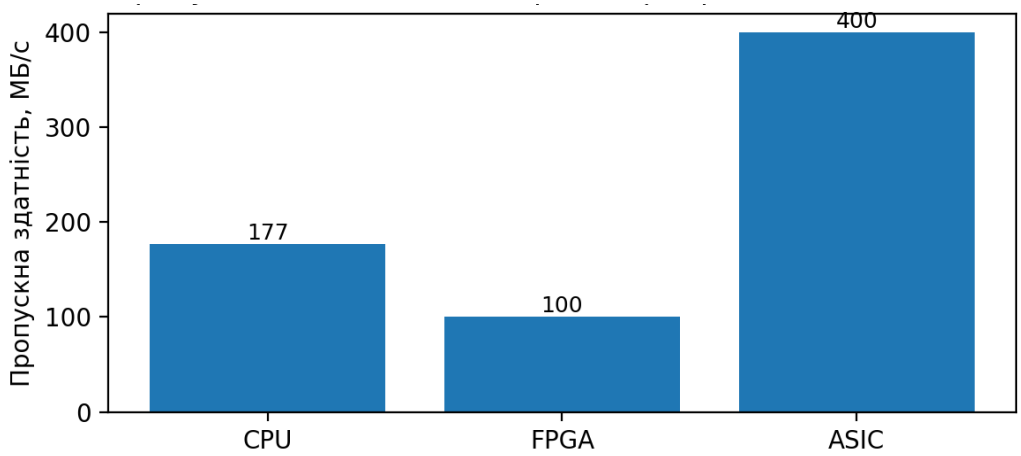

Рис. 4 Пропускна здатність реалізацій (режим підвищеної дифузії)

На рис. 5 наведено показники латентності формування геш-значення для кожної платформи. Узагальнені дані подано у табл. 3.

Отримані значення:

1) CPU — 0,36 мс,
2) FPGA — 0,66 мс,
3) ASIC — 0,16 мс.

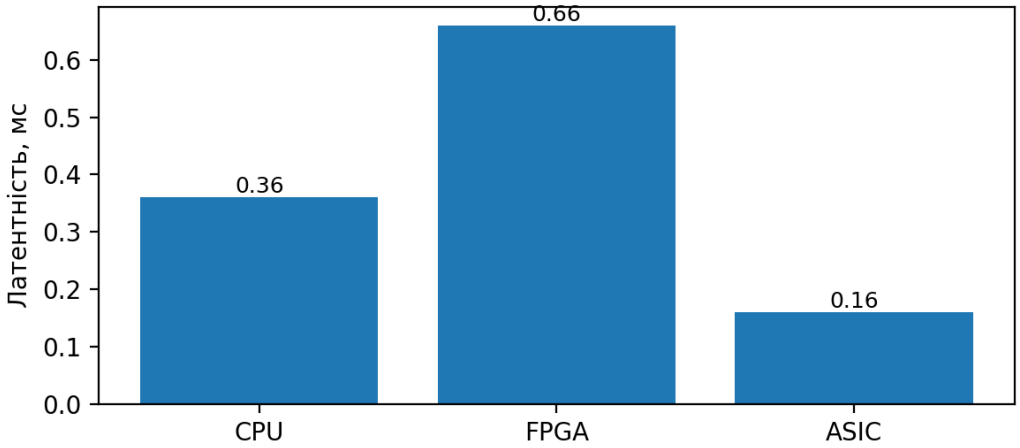

Рис. 5 Латентність реалізацій (режим підвищеної дифузії)

Найменшу затримку демонструє ASIC-реалізація, що обумовлено:

1) фіксованою мікроархітектурою,
2) відсутністю операцій системного керування,
3) мінімізацією міжмодульних затримок.

CPU має помірну латентність, яка залежить від завантаження системи та ефективності кешування. FPGA характеризується найбільшою затримкою через додаткові часові витрати на маршрутизацію сигналів та менш агресивну конвеєризацію.

Зменшення латентності є критичним для систем реального часу, зокрема:

1) систем контролю цілісності в приватних хмарних середовищах,
2) механізмів inline-дедуплікації,
3) індексації блоків у розподілених сховищах.

Таблиця 2

Пропускна здатність обчислення гешу (режим підвищеної дифузії)

Платформа	**Пропускна здатність, МБ/с**
CPU	177
FPGA	100
ASIC (оцінка)	400

Таблиця 3

Латентність формування гешу (режим підвищеної дифузії)

Платформа	**Латентність, мс**
CPU	0.36
FPGA	0.66
ASIC	0.16

Поєднання результатів пропускної здатності та латентності дозволяє зробити такі висновки:

ASIC забезпечує максимальну продуктивність при мінімальній затримці, що робить його оптимальним для інтеграції у високонавантажені хмарні вузли з жорсткими SLA-вимогами.

CPU є універсальним рішенням з прийнятним співвідношенням продуктивність/гнучкість.

FPGA доцільна для дослідних зразків та адаптивних реалізацій, проте поступається за швидкодією.

Отримані експериментальні показники підтверджують доцільність використання спеціалізованого MH-процесора для апаратного прискорення байт-орієнтованого гешування у режимі підвищеної дифузії. Перевага апаратної реалізації проявляється як у throughput (МБ/с), так і у зменшенні latency (мс), що критично для масштабованих приватних хмарних інфраструктур із високою інтенсивністю потокових операцій.

Обговорення застосувань у приватній хмарі

Запропонований метод байт-орієнтованого гешування та спеціалізований MH-процесор доцільно розглядати як компонент інфраструктурного рівня приватної хмари, орієнтований на потокову обробку великих масивів даних із детермінованими часовими характеристиками. У контексті архітектури багатошарової системи (CL–DS–HS–OS–HW) геш-сервіс може бути інтегрований як окремий модуль HS із апаратним прискоренням на рівні HW, що дозволяє винести операції формування геш-значень із критичного шляху центрального процесора.

У задачах прийому та первинної валідації великих потоків файлів і об’єктів (edge-вузли, шлюзи, системи резервного копіювання) метод забезпечує лінійну часову складність обчислення $T(L) = O(L)$, де L—

довжина повідомлення у байтах, а також сталі апаратні затримки на етапах змішування та редукції. Це дозволяє прогнозувати латентність у режимі реального часу та забезпечувати гарантовану пропускну здатність при потоковій обробці.

При побудові глобального індексу гешів J_Hдля задач дедуплікації кожному блоку даних b_iставиться у відповідність геш-значення $h_i = H(b_i)$, після чого перевіряється умова належності $h_i \in J_H$. За умови позитивного збігу реалізується логічне посилання на вже збережений блок, що зменшує фізичний обсяг зберігання. Регулярність апаратної структури та можливість конвеєризації дозволяють застосовувати метод у складі контент-визначених схем (CDC), де важливо мінімізувати накладні витрати на обчислення гешів для великої кількості блоків.

З позиції інформаційної безпеки запропонований метод слід трактувати як високопродуктивну інженерну геш-мітку для задач цілісності, індексації та дедуплікації у довірених або контрольованих середовищах приватних хмар. Він не призначений для заміни стандартизованих криптографічних алгоритмів у протоколах автентифікації, електронного підпису або захисту від навмисних колізій. У випадках, де необхідні формально доведені властивості стійкості до колізій чи атак другого прообразу, доцільно використовувати стандартизовані алгоритми (наприклад, SHA-3) або комбінувати запропонований підхід із механізмами HMAC чи цифрового підпису відповідно до політик безпеки організації.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У роботі розроблено метод байт-орієнтованого гешування з підвищеним рівнем дифузії, який зберігає потокову природу обробки та лінійну часову складність, але забезпечує підвищену чутливість геш-значення до структурних змін даних за рахунок комбінування частотних $K[v]$та позиційних $S[v]$характеристик із подальшим нелінійним змішуванням і регулярною редукцією.

Формально геш-функція має структуру h=C(g(K[v],S[v])), де етап формування статистичних масивів виконується за один прохід по повідомленню, а редукційна функція $C(\cdot)$реалізується у вигляді регулярного дерева згортки, придатного до конвеєризації та паралельної реалізації.

Розроблено спеціалізований MH-процесор, що реалізує запропонований алгоритм у регулярній мікроархітектурі з чітким розподілом функціональних блоків (PI–PC–PM–DO). Така архітектура характеризується обмеженим та детермінованим критичним шляхом, підтримує потоковий режим роботи та допускає реалізацію у технологіях FPGA та ASIC. Апаратна структура використовує фіксовані масиви лічильників і акумуляторів, модуль змішування та блок редукції, що забезпечує передбачувану латентність і можливість масштабування шляхом паралельного тиражування обчислювальних модулів.

Наведені оцінки ресурсів пам’яті, пропускної здатності та латентності демонструють потенціал розвантаження центральних процесорів вузлів приватної хмари при виконанні задач контролю цілісності та дедуплікації. Запропоноване рішення формує окремий клас інженерних потокових геш-методів, оптимізованих для апаратної реалізації у сервісних контурах приватних хмарних інфраструктур, та може розглядатися як ефективний інфраструктурний компонент систем обробки та зберігання даних.

References

1. Armbrust M., Fox A., Griffith R. A view of cloud computing // Communications of the ACM. 2010. Vol. 53, No. 4. P. 50–58. https://dl.acm.org/doi/10.1145/1721654.1721672
2. Mell P., Grance T. The NIST Definition of Cloud Computing. NIST SP 800-145. 2011. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf
3. Rogaway P., Shrimpton T. Cryptographic hash-function basics: definitions, implications, and separations // FSE 2004. Springer, 2004. P. 371–388. https://link.springer.com/chapter/10.1007/978-3-540-25937-4_24/
4. National Institute of Standards and Technology. Secure Hash Standard (SHA-3). FIPS PUB 202. 2015. https://csrc.nist.gov/pubs/fips/202/final
5. Bertoni G., Daemen J., Peeters M., Van Assche G. Sponge functions. 2007. https://en.wikipedia.org/wiki/SHA-3
6. Harnik D., Pinkas B., Shulman-Peleg A. Side channels in cloud services: Deduplication in cloud storage // IEEE Security & Privacy. 2010. Vol. 8, No. 6. P. 40–47. https://www.pinkas.net/PAPERS/hps.pdf
7. Meyer D. T., Bolosky W. J. A study of practical deduplication // USENIX FAST. 2012. https://dl.acm.org/doi/10.1145/2078861.2078864
8. Hennessy J. L., Patterson D. A. Computer Architecture: A Quantitative Approach. 6th ed. Morgan Kaufmann, 2019. https://www.academia.edu/3770508/john_L_Hennessy_and_David_A_Patterson_computer_architecture
9. Kuon I., Tessier R., Rose J. FPGA architecture: survey and challenges // Foundations and Trends in EDA. 2008. Vol. 2, No. 2. P. 135–253.

https://www.eecg.toronto.edu/~jayar/pubs/kuon/foundtrend08.pdfhttps://www.eecg.toronto.edu/~jayar/pubs/kuon/foundtrend08.pdf

10. ISO/IEC 27001:2022. Information security management systems — Requirements. 2022. https://www.iso.org/standard/27001

11. Лужецький В. А., Кисюк Д. В., Комаров А. О. Метод та спеціалізований процесор для байт-орієнтованого хешування даних // ІТСМ 2016. С. 123–125. http://itcm.pnu.edu.ua/2016/docs/Luzhetskyi.pdf

12. Лужецький В. А., Савицька Л. А., Кисюк Д. В. Спеціалізований процесор для ущільнення даних // 2016. С. 108–110. https://ir.lib.vntu.edu.ua/handle/123456789/11399?show=full

References

1. Armbrust, M., Fox, A., & Griffith, R. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50–58. https://dl.acm.org/doi/10.1145/1721654.1721672

2. Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing* (NIST SP 800-145). National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf

3. Rogaway, P., & Shrimpton, T. (2004). Cryptographic hash-function basics: Definitions, implications, and separations. In *Fast Software Encryption (FSE 2004)* (pp. 371–388). Springer. https://link.springer.com/chapter/10.1007/978-3-540-25937-4_24

4. National Institute of Standards and Technology. (2015). *Secure hash standard (SHA-3)* (FIPS PUB 202). https://csrc.nist.gov/pubs/fips/202/final

5. Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2007). *Sponge functions*. https://en.wikipedia.org/wiki/SHA-3

6. Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2010). Side channels in cloud services: Deduplication in cloud storage. *IEEE Security & Privacy*, 8(6), 40–47. https://www.pinkas.net/PAPERS/hps.pdf

7. Meyer, D. T., & Bolosky, W. J. (2012). A study of practical deduplication. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*. https://dl.acm.org/doi/10.1145/2078861.2078864

8. Hennessy, J. L., & Patterson, D. A. (2019). *Computer architecture: A quantitative approach* (6th ed.). Morgan Kaufmann. https://www.academia.edu/3770508/john_L_Hennessy_and_David_A_Patterson_computer_architecture

9. Kuon, I., Tessier, R., & Rose, J. (2008). FPGA architecture: Survey and challenges. *Foundations and Trends in Electronic Design Automation*, 2(2), 135–253. https://www.eecg.toronto.edu/~jayar/pubs/kuon/foundtrend08.pdf

10. ISO/IEC. (2022). *ISO/IEC 27001:2022 Information security management systems — Requirements*. https://www.iso.org/standard/27001

11. Luzhetskyi, V. A., Kysiuk, D. V., & Komarov, A. O. (2016). Method and specialized processor for byte-oriented data hashing. In *ITCM 2016* (pp. 123–125). http://itcm.pnu.edu.ua/2016/docs/Luzhetskyi.pdf

12. Luzhetskyi, V. A., Savytska, L. A., & Kysiuk, D. V. (2016). Specialized processor for data compression (pp. 108–110). https://ir.lib.vntu.edu.ua/handle/123456789/11399?show=full