Тематичний напрям № 4

Інтелектуальні інформаційні системи, цифрові технології та автоматизація

УДК 004.451

АНАЛІЗ ТЕХНОЛОГІЙ ТА АЛГОРИТМІВ СТИСНЕННЯ ОПЕРАТИВНОЇ ПАМ'ЯТІ ДЛЯ СИСТЕМ ІЗ КРИТИЧНИМ ОБМЕЖЕННЯМ РЕСУРСІВ

Максим Главчев, Дмитро Шеховцов

Національний технічний університет «Харківський політехнічний інститут», м. Харків, Україна

Сучасні тенденції розвитку програмного забезпечення характеризуються постійним та стрімким зростанням вимог до обсягу оперативної пам'яті. Водночас існує широкий клас систем, де фізичне розширення пам'яті є неможливим, технологічно складним або економічно недоцільним. До таких систем належать вбудовані системи (Embedded), пристрої Інтернету речей (IoT), одноплатні комп'ютери, бюджетні ноутбуки, а також хмарні контейнери з жорстко лімітованими ресурсами.

В умовах дефіциту оперативної пам'яті операційні системи традиційно використовують механізм віртуальної пам'яті та свопінгу (swapping) – вивантаження рідко використовуваних сторінок пам'яті на вторинні сховища. Однак, різниця в латентності між оперативною пам'яттю (50-100 нс) та навіть найшвидшими NVMe SSD-накопичувачами (10-100 мкс) складає кілька порядків. Для традиційних жорстких дисків (HDD) ця різниця сягає десятків мільйонів разів. Це призводить до явища "bottleneck" (вузького місця) підсистеми введення-виведення (I/O) і, як наслідок, до критичного падіння продуктивності системи та зависань інтерфейсу.

Ефективною сучасною альтернативою традиційному свопінгу є технології стиснення оперативної пам'яті. Цей підхід базується на парадигмі, згідно з якою сучасні багатоядерні процесори мають значний надлишок обчислювальної потужності порівняно з пропускною здатністю дискової підсистеми. Витрати процесорного часу на компресію та декомпресію даних в оперативній пам'яті повністю компенсуються уникненням повільних дискових операцій.

В операційній системі Linux на рівні ядра реалізовано два основні механізми управління стисненням пам'яті: Zram та Zswap, які мають принципово різну архітектуру (табл.1). Zram функціонує як модуль ядра, що створює віртуальний стиснений блоковий пристрій безпосередньо в оперативній пам'яті. Цей пристрій монтується як простір підкачки. Головна перевага Zram – повна незалежність від дискового простору, що дозволяє системі працювати взагалі без фізичного swap-розділу. Це критично важливо для IoT-пристроїв на базі флеш-пам'яті (SD-карт, eMMC), оскільки значно подовжує ресурс їхньої роботи шляхом зменшення кількості циклів перезапису. Zswap, навпаки, працює як проміжний кеш між основною пам'яттю та традиційним дисковим swap-розділом. Коли сторінка виштовхується з RAM, Zswap перехоплює її, стискає і зберігає у власному пулі в оперативній пам'яті. На диск дані потрапляють лише тоді, коли пул Zswap переповнюється (відбувається виштовхування за алгоритмом LRU) або якщо сторінка не піддається ефективному стисненню.

Таблиця 1 - Порівняльна характеристика технологій Zram та Zswap

Характеристика	Zram	Zswap
Тип реалізації	Блоковий пристрій	Кеш перед swap
Потреба в дисковому swap	Ні	Так
Максимальний розмір	Фіксований	Динамічний (% RAM)
Fallback на диск	Ні	Так
Writeback на диск	Опціонально (з 4.14)	Автоматично
Типове застосування	Embedded, IoT, ChromeOS	Десктопи, сервери
Версія ядра Linux	3.14+	3.11+

Ефективність функціонування систем Zram та Zswap безпосередньо залежить від обраного алгоритму стиснення. Сучасні ядра Linux підтримують алгоритми LZ4, Zstd (Zstandard), LZO та інші. Аналіз їхніх характеристик дозволяє виділити специфіку застосування кожного (табд.2, рис.1):

1. LZ4 – алгоритм, оптимізований для забезпечення екстремально високої швидкості обробки. Швидкість розпакування досягає 4500 MB/s, що робить його безальтернативним вибором для OLTP-баз даних, систем реального часу та сценаріїв, де мінімальна латентність є абсолютним пріоритетом.

2. Zstd – сучасний алгоритм, що забезпечує найкращий баланс. Завдяки наявності рівнів стиснення (від 1 до 22), він дозволяє гнучко налаштовувати систему. Типовий коефіцієнт стиснення (CR) для Zstd становить 2.8:1, що значно перевершує конкурентів і дозволяє вмістити максимум інформації в лімітовану RAM.

3. LZO – перевірений часом алгоритм, що відрізняється низьким споживанням пам'яті для власних словників та передбачуваною продуктивністю, що зберігає його актуальність для найслабших мікроконтролерів.

Таблиця 2 – Порівняльні характеристики алгоритмів стиснення

Параметр	LZ4	Zstd	LZO
Рік створення	2011	2016	1996
Швидкість стиснення	~780 MB/s	~500 MB/s	~400 MB/s
Швидкість розпакування	~4500 MB/s	~1500 MB/s	~800 MB/s
Типовий коефіцієнт стиснення	2.1:1	2.8:1	2.1:1
Використання CPU	Низьке	Середнє	Низьке
Використання пам'яті	~16 KB	~128 KB	~64 KB
Ліцензія	BSD	BSD	GPL

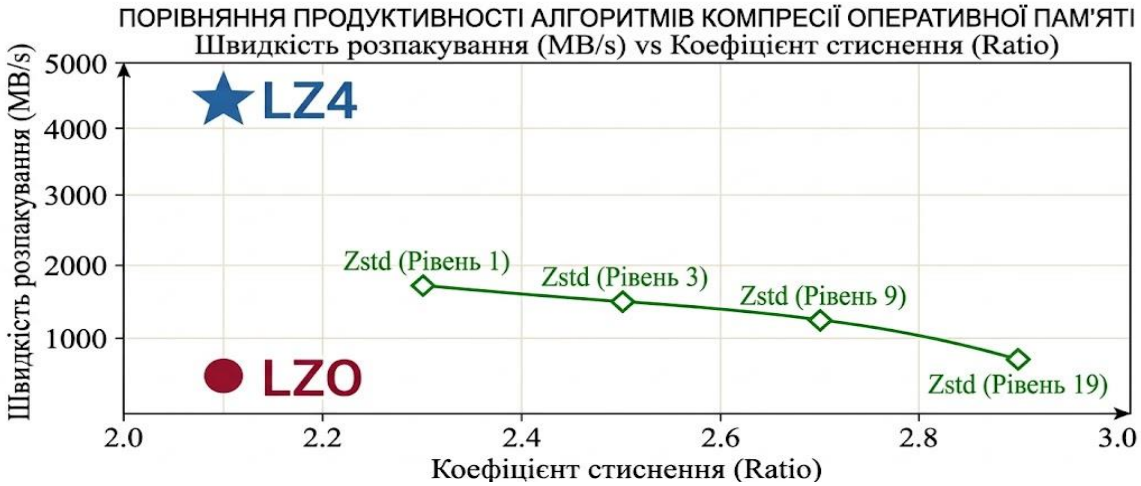


Рисунок 1 - Порівняння алгоритмів за швидкістю та ступенем стиснення

Детальний аналіз предметної області виявив суттєвий недолік існуючих підходів до конфігурації стиснення пам'яті. Більшість сучасних операційних систем (стандартні дистрибутиви Linux, ChromeOS, Android) використовують підхід статичного налаштування. Адміністратор або розробник

ОС обирає один алгоритм (найчастіше LZ4 або Zstd) на етапі завантаження системи, який залишається незмінним протягом усього сеансу роботи незалежно від зміни умов (рис.2).

Такий підхід є неоптимальним для мобільних систем та пристроїв з акумуляторним живленням. По-перше, статична конфігурація не враховує енергетичний стан. Алгоритми з високим ступенем стиснення (наприклад, Zstd на рівні 3 і вище) створюють значне навантаження на CPU (до 85%), що призводить до прискореного розряду батареї. За умов низького заряду доцільно було б автоматично перемикатися на більш енергоефективний LZ4. По-друге, ігнорується температурний режим. У системах з пасивним охолодженням інтенсивна декомпресія спричиняє нагрівання кристала процесора і наступний тепловий тротлінг (thermal throttling), що нівелює всі переваги стиснення.

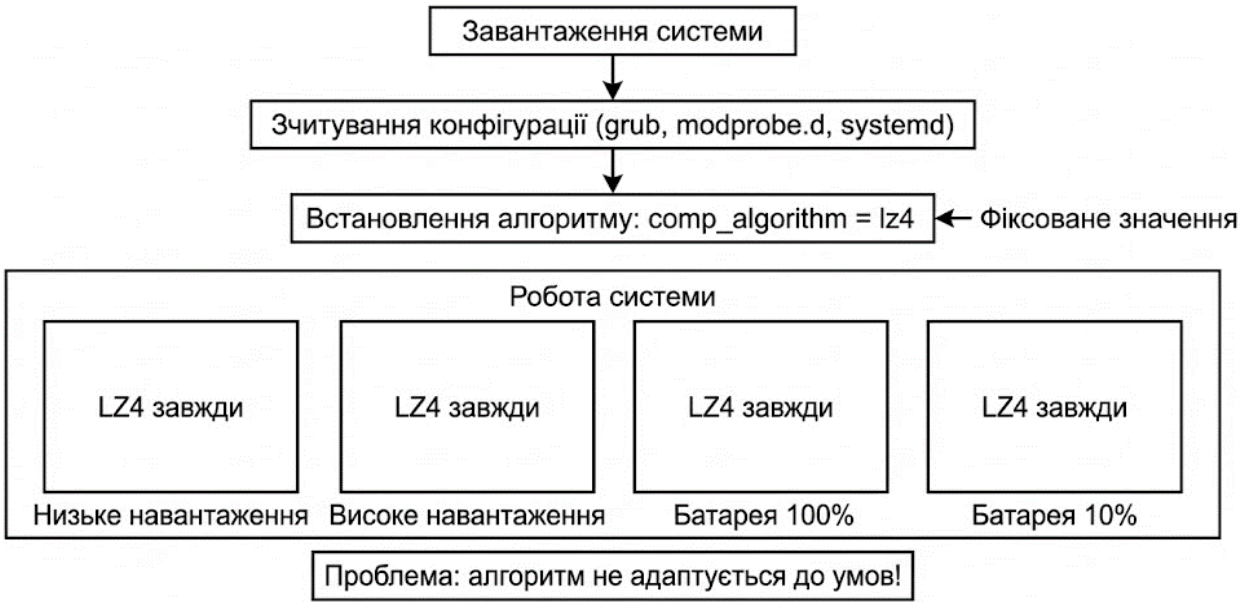


Рисунок 2- Недолік статичного підходу до вибору алгоритму

Висновки. Проведений аналіз доводить, що не існує єдиного універсального алгоритму стиснення оперативної пам'яті для систем з обмеженими ресурсами. Статичний підхід до конфігурації підсистем Zram/Zswap призводить до субоптимальної продуктивності, підвищеного енергоспоживання та ризику перегріву. Вирішенням цієї проблеми є створення адаптивної моделі управління, яка б забезпечувала динамічне перемикання методів стиснення (наприклад, від Zstd до LZ4) в режимі реального часу (на льоту) на основі моніторингу температури CPU, рівня заряду акумулятора та поточного профілю навантаження на систему. Розробка та програмна реалізація такого адаптивного алгоритму становить актуальну задачу для подальших досліджень.

Список використаних джерел

1. Zram: Compressed RAM-based block device. The Linux Kernel documentation. URL: https://www.kernel.org/doc/html/latest/admin-guide/blockdev/zram.html (дата звернення: 04.10.2026).

2. Zswap: Compressed swap cache. The Linux Kernel documentation. URL: https://www.kernel.org/doc/html/latest/admin-guide/mm/zswap.html (дата звернення: 04.10.2026).

3. Collet Y. LZ4: Extremely Fast Compression algorithm. URL: https://lz4.github.io/lz4/ (дата звернення: 04.10.2026).

4. Collet Y. Zstandard: Real-time data compression algorithm. URL: https://facebook.github.io/zstd/ (дата звернення: 04.10.2026).

5. Vahdat A., Lebeck A., Ellis C. Every Joule is Precious: The Case for Revisiting Operating System Design for Energy Efficiency. Proceedings of the 9th Workshop on ACM SIGOPS European Workshop. Kolding, 2000. P. 31–36.