УДК 004.023

ВДОСКОНАЛЕНИЙ МЕТОД ДЕДУБЛІКАЦІЇ НА ФАЙЛОВОМУ РІВНІ

Піговський Ю.Р.
Тернопільський національний економічний університет, к.т.н., доцент

I. Постановка проблеми

Дедублікація - це процес усунення дублікатних копій даних. Коли повторюваність (реплікованість) даних є доволі високою, що притаманне серверам резервного копіювання, образам віртуальних машин і репозиторіям сирцевих (source) кодів, дедублікація може зменшити витрати простору і збільшити швидкість обробки даних не на проценти, а навіть на порядки [1].

Заощадження простору є очевидним; підвищення ж швидкості відбувається внаслідок уникнення операцій запису на диск при збереженні дубльованих даних, а також зменшеного обсягу задіяної віртуальної пам'яті за рахунок спільного використання її сторінок багатьма застосунками.

В загальному випадку, дедублікація може розглядатися на одному з трьох рівнів: файловому, блоковому або байтовому. Дедублікація перших двох рівнів має практичний зміст, тоді як недоцільність дедублікації на байтовому рівні обгрунтовано в [1].

Відомим розв'язком задачі дедублікації на файловому рівні є утиліта А. Лопеза fdupes [2], що входить до складу GNU Coreutils. Проте алгоритм її роботи недостатньо ефективний, бо обчилює MD5 хеш кожного файла безумовно, не враховуючи доцільності такого обчислення для конкрентного співвідношення швидкості доступу до файлів та їхнього об'єму.

На блоковому рівні задачу дедублікації розв'язано Дж. Бонвіком [1] в модулі файлової системи ZFS. Дедублікація в ZFS має велику кількість допустимих комбінацій параметрів, зокрема: хешування алгоритмом SHA256, проста побайтова верифікація або хешування алгоритмом Флетчера (fletcher4) з побайтовою верифікацією. При цьому існує проблема у виборі найефективніших налаштувань, оскільки на деяких даних хешування алгоритмом SHA256 може відбуватися повільніше ніж комбінований підхід хешування алгоритмом Флетчера з побайтовою верифікацією і навпаки.

II. Мета роботи

Дану статтю присвячено підвищенню ефективності процедури дедублікації на файловому рівні шляхом побудови теоретичного правила, що дозволило б визначити при яких співвідношеннях швидкості доступу до файлів та їхнього об'єму доцільно проводити хешування, а при яких — ні.

Ефективність теоретичного правила буде перевірено експериментально.

III. Евристичний метод пошуку дублікатів на файловому рівні

Нехай потрібно знайти дублікати на множині файлів $\mathbf{F}$. За допомогою деякого алгоритму побайтового порівняння пари файлів $f_1, f_2 \in \mathbf{F}$ можна визначити чи їхній контент еквівалентний. Припустимо, що номер першого байта m, що відрізняє контенти файлів f_1 та f_2 є рівномірно-розподіленою в діапазоні $1 \le m \le L$ випадковою величиною з матсподіванням $\overline{m} = \frac{1+L}{2}$, тоді математичне сподівання тривалості $T_C(n, L)$ попарного порівняння можна оцінити так:

$$\overline{T_C}(n, L) = (n-1)\, t_{pr} + \frac{n(n-1)}{2}\left(t_{pr} + (1+L) \cdot t_r\right), \qquad (1)$$

де (n–1) — кількість разів, яку виконується опарація відкриття файла в алгоритмі побайтового порівняння; t_{pr} — середня тривалість приготувань до роботи з файлом (обробка структур даних ядра ОС при відкритті і закритті одного файла), t_r — середня тривалість зчитування одного байта.

Беручи до уваги співвідношення (1) можна запропонувати наступне правило: хешування проводити доцільно, коли оцінка його тривалості менша за математичне сподівання тривалості попарного порівняння

$$\mathrm{H}_\alpha(n, L) = \begin{cases} \text{так}, & T_H(n, L) < \alpha\, \overline{T_C}(n, L), \\ \text{ні}, & \text{інакше}, \end{cases} \qquad (2)$$

де $T_H(n, L)$ — тривалість обробки n файлів об'ємом L байт деякою хеш-функцією (хешування) [3]

$$T_H(n, L) = n\,(t_{ph} + L \cdot t_h), \qquad (3)$$

де n — кількість файлів довжиною L байтів; t_{ph} — середній час, що витрачається на підготовку до хешування незалежно від об'єму оброблюваного файлу; t_h — час, що в середньому припадає на хешування одного байту (звісно, що це штучне поняття, бо реальні алгоритми працюють з блоками).

Роль коефіцієнту α в (2) можна пояснити як засіб урівноваження припущень щодо розподілу випадкової величини позиції відмінності при попарному порівнянні файлів.

IV. Чисельні експерименти

На рис. 1 показано тривалість роботи узагальненого алгоритму з використанням співвідношень (1)–(3) для різних значень емпіричного коефіцієнту в діапазоні $0 \le \alpha \le 2$. При $\alpha = 0$ алгоритм зводиться до алгоритму без хешування і триває 16 862 секунди (4 години 41 хвилину), тривалість роботи алгоритму з безумовним хешуванням (при $\alpha = 2$) не надто відрізняється — 16 405 секунд (4 години 33 хвилини), тоді як мінімальна тривалість дедублікації досягається при $\alpha = 0.1$ і складає 15 950 секунд, що на 3% швидше від алгоритму із безумовним хешуванням і на 5% швидше від алгоритму без хешування.

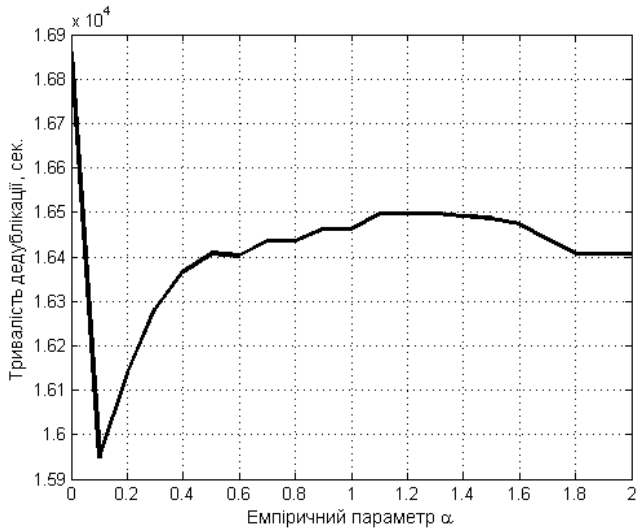

Рисунок 1 - Тривалість роботи узагальненого алгоритму для $0 \le \alpha \le 2$

Такий незначний, на перший погляд, виграш часу, може обернутися суттєвим виграшом при зростанні об'єму, кількості файлів та зберігальних пристроїв у мережах Storage Area Networks.

Висновок

Створено правило доцільності виконання процедури хешування в залежності від кількості, об'єму та швидкості доступу до файлів в задачі пошуку дублікатів контенту. Правило грунтується на теоретичних дослідженнях математичного сподівання тривалості процедур хешування та попарного порівняння.

Розроблене правило дає змогу знайти ті групи файлів однакового розміру, у котрих доцільно проводити хешування з метою підвищення ефективності пошуку дублікатів на файловому рівні. Дедублікація розробленим методом триває 15 950 секунд, що на 3% швидше від алгоритму із безумовним хешуванням і на 5% швидше від алгоритму без хешування.

У наступних дослідженнях планується покращити розроблений метод з метою збільшення виграшу швидкодії і поширити область його застосування на задачу дедублікації на блоковому рівні.

Список використаних джерел

1. Bonwick J. ZFS Deduplication. — 2009 [Електронний ресурс]. — Режим доступу: https://blogs.oracle.com/bonwick/entry/zfs_dedup
2. Lopez A. fdupes(1) — Linux man page [Електронний ресурс]. — Режим доступу: http://linux.die.net/man/1/fdupes
3. Шеховцов В.А. Операційні системи / В. А. Шеховцов // Захист інформації в операційних системах. — К.: Видавнича група BHV, 2005. — Розд. 18. — С. 471–472.