DOI 10.31891/2307-5732-2022-315-6-24-27
УДК 007

БАРНА Андрій
Національний університет «Львівська політехніка»
https://orcid.org/0000-0002-6692-7496
e-mail: andrii.o.barna@lpnu.ua
КАМІНСЬКИЙ Роман
Національний університет «Львівська політехніка»
https://orcid.org/0000-0002-0563-5748
e-mail: kaminsky.roman@gmail.com

АНАЛІЗ ЕФЕКТИВНОСТІ МЕТОДІВ РОЗБИТТЯ ПОТОКУ ДАНИХ ДЛЯ СИСТЕМ ДЕДУБЛІКАЦІЇ ДАНИХ

В роботі наведено результати порівняння ефективності методів розбиття потоку даних: класичного TTTD та нового CB-TTTD, які використовуються в системах дедублікації даних.
Ключові слова: дедублікація, фрагментування, хешування.

BARNA Andrii, KAMINSKY Roman
Lviv Polytechnic National University

ANALYSIS OF THE EFFICIENCY OF DATA CHUNKING METHODS FOR DATA DEDUBLICATION SYSTEMS

There is a significant increase in the amount of data that needs to be stored worldwide. More and more companies are turning their attention to deduplication systems, which effectively increase data warehouse volume and reduce storage costs. Deduplication not only reduces the overall amount of information in storage but also reduces the load on networks by eliminating the need to retransmit duplicate data. In this work, we considered the stages that any deduplication system includes, namely chunking, hashing and indexing, mapping. The effectiveness of deduplication systems primarily depends on the choice of the method of dividing the data stream at the chunking stage. We considered the classic Two Threshold Two Divisor (TTTD) method, which is widely used in modern deduplication systems. This method uses Rabin's fingerprint to find the hash of the substring value. The formula for calculating the hash for the first substring and the formula for calculating the rest of the substring are given. Another method we investigated is Content Based Two Threshold Two Divisor (CB-TTTD) - it uses new hash functions to fragment the data stream, and the corresponding formulas for calculating the first and each subsequent substring are given. To test the effectiveness of these two methods, we developed a test deduplication system, implemented these two fragmentation methods, and tested their performance on two sets of text data. We have modified these methods with the addition of a new string-splitting condition based on the content specification of the data we tested. The results of a comparison of the work of classical and modified methods are given. Using metrics to compare the efficiency of data fragmentation methods, we obtained experimental data, based on which we can make conclusions about the feasibility of using CB-TTTD as an alternative to TTTD in new deduplication systems. The obtained data can be used in the development of new highly efficient data deduplication systems and to improve old solutions
Keywords: deduplication, chunking, hashing.

Постановка проблеми

Зараз у світі спостерігається вибух обсягу цифрових даних, про що свідчить значне зростання вимірюваного обсягу збережених даних. Для прикладу, у 2010 та 2011 роках з 1,2 цетабайта до 1,8 цетабайта 1 відповідно [1], у 2020 році, становило вже 44 зетабайти [2, 3]. Тож управління сховищем, яке є економічно ефективним, стало важливим завданням в епоху великих даних. Дослідження робочого навантаження, проведені американською транснаціональною корпорацією Dell, EMC (Richard Egan, Roger Marino & John Curly the E, M&C in EMC) та Microsoft, свідчать про те, що приблизно 50% і 85% даних є надлишковими в основних і резервних сховищах даних.

Згідно з недавнім дослідженням Міжнародної корпорації даних (IDC), майже 80% опитаних корпорацій зазначили, що вони використовують технології дедублікації у своїх системах зберігання даних, які ефективно збільшують обсяг зберігання даних та скорочують витрати на їх зберігання [4]. Дедублікація даних не тільки робить збереження даних більш ефективним, але й зменшує навантаження в мережі, усуваючи зайві дані в мережевих середовищах з низькою пропускною здатністю.

Загалом будь-яка система дедублікації включає в себе три етапи: (фрагментація, хешування та індексування і етап зіставлення).

Система дедублікації розбиває вхідний потік даних на безліч «фрагментів» даних, для яких обчислюється хеш-сигнатура (наприклад, SHA-1), і виявляє повторювані за допомогою певного методу порівняння. Системи дедублікації видаляє повторювані фрагменти та зберігає або передає лише одну копію з них для досягнення мети економії місця на сховищі або пропускної здатності мережі. З іншого боку, використання такої системи зменшує швидкість резервного копіювання та задіює багато ресурсів центрального процесора.

Ефективність ж самої системи дедублікації найбільше залежить саме від вибору методу фрагментації, що безпосередньо впливає на загальний об’єм переданої та збереженої інформації.

Аналіз останніх джерел

Автори роботи [5] провели опитування щодо різних алгоритмів фрагментації даних для дедублікації. Було визначено, що найпоширенішим алгоритмом фрагментації був Two Threshold Two Divisor (TTTD), вони оцінили цей алгоритм за допомогою трьох різних функцій хешування; Rabin Fingerprint, Adler і SHA1, а саме визначили коефіцієнт дедублікації та швидкість роботи кожного з них в тестовій системі.

Найефективніше себе показав TTTD з використанням функції Rabin Fingerprint

TTTD — це алгоритм фрагментації змінного розміру [6], який використовує відбиток Рабіна, щоб знайти хеш-значення підрядка з попередньо визначений розміром вікна (48 байт). Якщо хеш цього підрядка задовольняє умову TTTD, це буде розглядатися як точка зупинки, інакше збільшить розмір вікна на один байт [7].

Нижче наведено формула(1) обчислення відбитка Рабіна для першого підрядка. Потім формула (2), яка використовується для решти підрядка. Вона працює шляхом видалення першого символу з рядка та додавання нового [1].

$$Rabin\left(B_1, B_2, \ldots B_\alpha\right) = \{\sum_{i=0}^{\alpha}\left(B_i * P^{\alpha-i-1}\right)\} Mod\ D \quad 1)$$

$$\left\{\left[Rabin\left(B_i, B_{i+1}, \ldots B_{i+\alpha-1}\right) - B_i * P^{\alpha-1}\right] * P + B_{i+\alpha}\right\} Mod\ D \quad 2)$$

Тут D — середній розмір блоку [5], Bx — ASCII код для символів підрядка, P – просте число, α – це розмір зсувного вікна.

В іншій роботі [8] наведено новий експериментальний метод фрагментування потоку даних Content Based Two Threshold Two Divisor (CB-TTTD) на основі алгоритму TTTD та запропоновано покращити техніку дедублікації шляхом прискорення операція дедублікації та збільшення її коефіцієнта стиснення.

CB-TTTD використовує нові хеш-функції для фрагментування потоку даних. Для кожного символу в першому рядку вікна розміром (36 байт) значення хешу обчислюється за допомогою формули 3. Тоді для кожного з наступних підрядків значення хешу розраховується за допомогою формули 4.

$$FingerPrint\left(B_0, B_2, \ldots B_\alpha\right) = \{\sum_{i=0}^{\alpha-1} Val\left[B_i\right] * 2^{i+1}\} \quad (3)$$

$$New\,FingerPrint\left(B_{i+1}, \ldots B_{i+\alpha+1}\right) = \left[\left\{FingerPrint\left(B_i, \ldots B_{i+\alpha}\right) - Val\left[B_i\right]\right\} \div 2\right] + Val\left[B_{i+\alpha+1}\right] * 2^{i+1} \quad (4)$$

Тут α — розмір підрядка, B1 … Bα: — символи підрядка, Val[Bi]: це значення індексу [Bi] у масиві відбитків. Значення символу, взяте з масиву розміром 256, яке представляє друковані символи та заповнене випадковими числами з (1, 2), щоб створити масив, як на таблиці 1:

Таблиця 1

Масив відбитків

Array[0]	Array[1]	Array[2]	Array[3]	Array[4]		Array[255]
1	2	1	2	1	……..	2

Метою нашої роботи є дослідження нового методу фрагментування даних CB-TTTD для покращення коефіцієнта дедублікації та порівняння його ефективності з методом TTTD.

Виклад основного матеріалу

Для перевірки ефективності методів TTTD та CB-TTTD нами було розроблено тестову систему дедублікації даних та імплементовано обидва методи для етапу фрагментування даних. Для тестування роботи системи ми обрали два набори даних, які містять текстові файли із загальним розміром 580 Мб та 2329 Мб відповідно.

Оскільки метод CB-TTTD під час роботи оцінює «вагу» символу в потоці текстових даних, нами було вирішено додати нову умову в систему: символ крапки (‘.’) розглядається як нова умова в додаток до головної та другорядної умови дільника рядка в TTTD. Коли символ крапки знайдено в рядку і за ним слідує символ пробілу чи кінця рядка, той цей параграф розглядається як окремий фрагмент даних. Переваги такої умови проявляються у випадку коли два параграфи тексту розглядаються системою як один фрагмент. Тоді будь які зміни в одному з параграфів будуть мати вплив і на інший. Проте з використанням додаткової умови, ці два параграфи будуть розділені в різні фрагменти, а отже і внесені зміни будуть мати вплив лиш на один з них. Додавання такої умови збільшило коефіцієнт дедублікації та не вплинуло на швидкість розбиття потоку даних на фрагменти, тому що кількість кроків для обчислення межі фрагмента даних не змінилося. На рис.1 зображено вихідний розмір даних після дедублікації з використання оригінального методу TTTD для фрагментування даних та методу CB-TTTD для однакових наборів даних. Отримані результати порівняння роботи методів TTTD та CB-TTTD з використання додаткової умови поділу фрагментів на основі даних з першої вибірки наведені в таблиці 2.

Тестова система дедублікації була запущена на сервері з наступною конфігурацією: Intel I7 CPU, 8 GB RAM, 64 бітна операційна система. Щоб визначити ефективність кожного з методів фрагментування даних були використані наступні метрики:

- розмір даних після дедублікації: показує скільки даних залишилося після того як система видалила повторювані елементи;
- коефіцієнт дедублікації: вказує на ефективність методу дедублікації та обчислюється як відношення розміру даних до дедублікації, до розміру після дедублікації;
- час фрагментування: загальний час за який система розділила потік даних на фрагменти;

- час дедублікації: загальний час роботи системи;
- середній розмір фрагменту: обчислюється як відношення загального розміру даних до кількості фрагментів.

Експериментальні дані наведені в таблиці 3. Вони демонструють що метод CB-TTTD є більш ефективним для фрагментування потоку даних. З використанням цього методу підвищився коефіцієнт дедублікації для малих та великих наборів даних. Час розбиття потоку даних на фрагменти також скоротився.

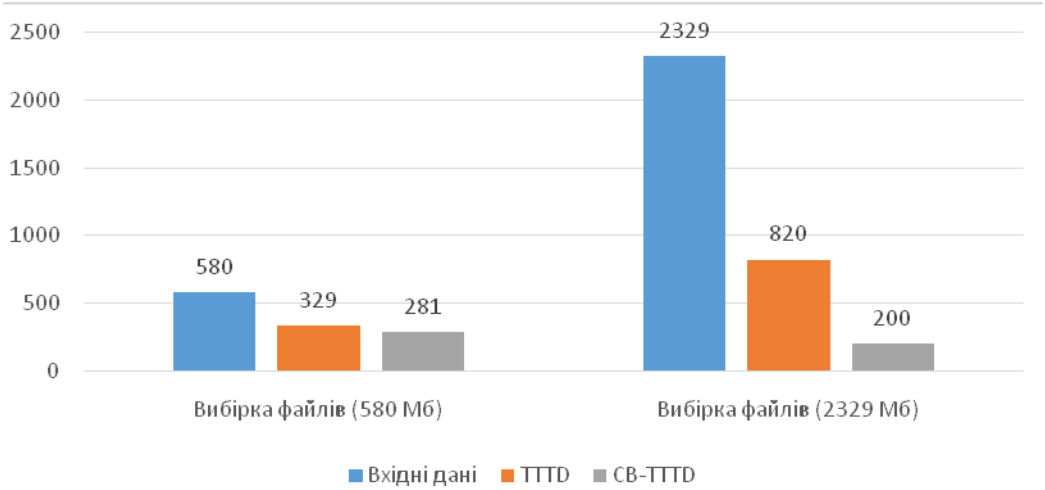

Рис. 1. Розмір вхідних та вихідних даних з використанням TTTD та CB-TTTD у Мб

Таблиця 2

Ефект використання додаткової умови поділу фрагменту для методів TTTD та CB-TTTD

Алгоритм	Кількість фрагментів	Коефіцієнт дедублікації	Розмір метаданих в Мб	Час роботи
TTTD без дод. умови	612862	1.78200	83.1	2305
TTTD з дод. умовою	1543376	1.83175	125	3450
CB-TTTD без дод. умови	654832	2.03643	15.9	440
CB-TTTD з дод. умовою	980152	2.1385	23.2	582

На відміну від функції відбитку Рабіна, який використовується в методі TTTB, у CB-TTTD збільшилася кількість фрагментів, особливо малого розміру, що дозволяє центральному процесору витрачати менше ресурсів на обробку кожного фрагменту. Варто зазначити, що додаткова умова поділу потоку даних на фрагменти(умова з крапкою) також збільшила кількість фрагментів малого розміру та підвищила коефіцієнт дедублікації без значного впливу на загальний час фрагментування даних системою

Таблиця 3

Порівняння методів TTTD та CB-TTTD

Метрики	Набір даних 1		Набір даних 2	
	TTTD	CB-TTTD	TTTD	CB-TTTD
Вхідні дані(Мб)	580	580	2329	2329
Розмір даних після дедублікації(Мб)	329	281	820	200
Виявлено повторюваних даних (Мб)	251	299	1509	2129
Коефіцієнт дедублікації	1.76	2.07	2.84	11.6
Час фрагментування даних в сек.	730	470	3645	1204
Час дедублікації в сек.	2640	2104	9083	4587
Загальна кількість фрагментів	621861	960091	2468026	2611322
Середній розмір фрагменту	979	633	990	925

Проте, як видно з даних про загальний час роботи системи (дедублікації), через суттєве збільшення кількості фрагментів відповідно збільшився загальний час роботи системи саме на етапах індексування і хешування та зіставлення. Не зважаючи на ріст ефективності на етапі фрагментування даних, потрібно зробити корективи в процесах індексування та хешування фрагментів, їх порівняння, щоб отримати відповідний загальний виграш в продуктивності системи. Отже, наступні кроки у нашій роботі будуть полягати у дослідженні цих двох етапів та пошуку можливих методів вирішення цих часових проблем.

Висновки

При роботі з великою кількістю даних їх ефективне зберігання стає важким завданням. Нами було розроблено тестову систему дедублікації даних та з її допомогою проведено порівняння існуючих методів фрагментування потоку даних. Експериментально доведено, що класичний метод TTTD поступається в ефективності новому CB-TTTD. Другий забезпечує вищий коефіцієнт дедублікації, сам процес фрагментування даних виконується швидше, а метадані, які створюються для фрагментів даних займають менший об'єм пам'яті. Додаткові умови розбиття потоку даних, які були додані до методу CB-TTTD, також покращили коефіцієнт дедублікації без значного приросту в часі роботи, що вказує перспективи підвищення його ефективності. Тестування методів фрагментування даних оцінювалася за допомогою двох різних наборів даних. Отримані результати заохочують продовжувати дослідження процесів розбиття потоків даних для покращення ефективності систем дедублікації.

References

1. Stevenson D., Wagoner N. J. Bargaining in the shadow of big data. Fla. Law Rev vol. 66. 2014. № 5. P. 66.

2. John Reinsel, Gantz Reinsel, David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. IDC iView: IDC Analyze the future. 2012. P. 1-16.

3. Turner, V., Gantz, J. F., Reinsel, D., & Minton, S. The digital universe of opportunities: Rich data and the increasing value of the internet of things. IDC Analyze the Future. 2014. P. 5.

4. Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, Yukun Zhou. A comprehensive study of the past, present, and future of data deduplication. Proc. IEEE vol. 104. 2016. № 9. P. 1681–1710.

5. Fahad A., Abdulsalam H. Evaluation of Two Thresholds Two Divisor chunking algorithm using Rabin fingerprint, Adler, and SHA-1 hashing algorithms. The Iraqi Journal of Science. 2017. № 4. P. 58.

6. Demystifying Data Reduplication: Choosing the Best Solution. 2009. URL: http://www.pexpo.co.uk/contentldownload/20646/353747/file/DemystifyingDataDedupe.

7. Mark W. Storer, Kevin Greenan, Darrell D. E. Long, Ethan L. Miller. Storer Secure Data Deduplication. StorageSS'08. 2008. № 5. P. 1-10.

8. Fahad A., Abdulsalam H. New Techniques to Enhance Data Deduplication using Content based-TTTD Chunking Algorithm. (IJACSA) International Journal of Advanced Computer Science and Applications. vol. 9. 2018. № 5. P. 116.