УДК 519.688

І. П. Дробязко, В.М. Оберемчук

Підвищення продуктивності кластеризації даних

Запропоновано алгоритм кластеризації, який дозволяє підвищити продуктивність кластеризації даних за рахунок використання обчислювальних можливостей сучасних графічних процесорів з підтримкою технології CUDA та багатоядерних центральних процесорів.

The algorithm for clustering, which improves the performance of data clustering through the use of computational power of modern GPUs with CUDA technology and multi-core CPUs.

Ключові слова*: кластеризація, підвищення продуктивності, сучасні графічні процесори, CUDA, багатоядерні центральні процесори.*

Вступ

Для рішення багатьох прикладних задачах, які потребують обробки значних обсягів даних, доцільно використовувати кластеризацію даних. Під кластеризацією розуміють розбиття заданої вибірки об'єктів на підмножини (їх називають кластерами) таким чином, щоб кожний кластер складався зі схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися.

Цілі кластеризації можуть бути різними в залежності від особливостей конкретної прикладної задачі:

Зрозуміти структуру безлічі об'єктів, розбивши його на групи схожих об'єктів. Спростити подальшу обробку даних і прийняття рішень, працюючи з кожним кластером окремо.

Скоротити обсяг збережених даних у разі надвеликої вибірки, залишивши по одному найбільш типовому представникові від кожного кластера.

Виділити нетипові об'єкти, які не підходять до жодного з кластерів.

Алгоритми кластеризації характеризуються тим, що виконують поділ набору елементів векторного простору на заздалегідь відому кількість кластерів і мають досить високу обчислювальну складність, яка експоненційно зростає зі збільшенням кількості та розмірності вхідних даних та збільшенням кількості кластерів. Тому важливою задачею є підвищення продуктивності кластеризації.

В роботах [1], [2] розглядаються алгоритми кластеризації, які використовують технологію MPI. У роботі [3] розглядається алгоритм кластеризації, який використовує фреймворк MapReduce. Запропоновані у цих роботах алгоритми дозволяють підвищити продуктивність кластеризації великих обсягів даних при їх застосуванні на обчислювальних кластерах. Але для актуальної задачі підвищення продуктивності кластеризації даних, що зберігаються у корпоративних сховищах даних, організація та застосування таких кластерів вимагає надмірних витрат.

Метою даної роботи є розробка алгоритму кластеризації, що використовує обчислювальні можливості сучасних графічних процесорів, які підтримують технологію CUDA, та багатоядерних центральних процесорів для підвищення продуктивності кластеризації даних, які зберігаються у корпоративних сховищах.

Алгоритм кластеризації

При кластеризації виконується розподіл набору елементів векторного простору на заздалегідь відому кількість кластерів шляхом мінімізації дисперсії на точках кожного кластера:

$$D = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - m_i)^2 ,$$

де k — число кластерів, S_i — отримані кластери ($i = 1,2,...,k$) і m_i — центри мас векторів $x_i \in S_i$.

Спочатку виконується вибір початкових центрів мас кожного з k кластерів. Цей вибір здійснюється випадковим чином.

Далі виконується розподіл векторів на кластери у відповідності з тим, який з центрів мас виявився ближчим за обраною метрикою.

$$S_i^{(t)} = \{x_j : \left\| x_j - m_i^{(t)} \right\| \le \left\| x_j - m_{i^*}^{(t)} \right\|,$$
$$\forall i^* = 1,...,k\}$$

При використанні графічного процесора розподіл виконується паралельно для кожного елемента з вхідного набору даних. Всі елементи даних поділяються на частини приблизно однакового розміру, які оброблятимуться незалежними потоками. Для цього потрібно виконати наступні кроки:

1. Визначення необхідної кількості ітерацій.
2. Синхронізація потоків.
3. Визначення початкового елементу даних потоку.
4. Визначення чергового кластера.
5. Якщо номер потоку менше кількості вимірів, тоді перейти на крок 6, інакше — на крок 7.
6. Завантаження значення центру мас кластера за відповідним виміром.
7. Синхронізація потоків.
8. Визначення відстані між елементом даних і кластером.
9. Якщо відстань мінімальна, тоді перейти на крок 10, інакше — на крок 11.
10. Збереження номера кластера.
11. Синхронізація потоків.
12. Якщо визначені відстані до центрів мас усіх кластерів, тоді перейти на крок 13, інакше — на крок 4.
13. Збереження номера кластера, відстань до якого мінімальна.
14. Визначення наступного елементу даних потоку.
15. Зменшення кількості ітерацій.
16. Якщо кількість ітерацій дорівнює нулю, тоді перейти на крок 17, інакше — на крок 4.
17. Синхронізація потоків.

Далі виконується обчислення центру мас для кожного кластера, отриманого під час виконання попередніх кроків.

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

При використанні багатоядерного процесора, всі елементи даних поділяються на частини приблизно однакового розміру. Для кожної частини окремим потоком виконується визначення таких локальних параметрів як розмір кожного кластера та сума значень елементів за кожним виміром. Потім для кожного кластера за допомогою незалежних потоків здійснюється обчислення центру мас кластера з використанням усіх обчислених локальних параметрів. Для цього виконуються наступні кроки:

1. Визначення чергового елементу даних потоку.
2. Визначення кластера, до якого належить елемент даних.
3. Збільшення значення локального лічильника елементів кластера, до якого належить елемент даних.
4. Додавання значення елементу даних до локальної суми значень кластера за черговим виміром.
5. Якщо додавання виконано за усіма вимірами, тоді перейти на крок 6, інакше — на крок 4.
6. Якщо оброблено всі елементи даних потоку, тоді перейти на крок 7, інакше — на крок 1.
7. Визначення наступного кластера з кластерів потоку.
8. Визначення наступних неопрацьованих локальних даних.
9. Додавання значення локального лічильника елементів кластера до глобального лічильника кластера.
10. Додавання значення локальної суми значень кластера до глобальної суми кластера за черговим виміром.
11. Якщо додавання виконано за усіма вимірами, тоді перейти на крок 12, інакше — на крок 10.
12. Отримуємо нове значення центру мас кластера шляхом ділення глобальної суми кластера на значення його глобального лічильника.
13. Якщо додавання виконано для усіх локальних даних, тоді перейти на крок 14, інакше — на крок 8.
14. Якщо оброблено всі кластери потоку, тоді перейти на крок 15, інакше — на крок 7.
15. Визначення кількості кластерів, що змінили значення центру мас.

Далі перевіряється, чи змінилися кластери під час виконання попередніх кроків. Якщо зміни не відбувалися, тоді алгоритм завершується, інакше здійснюється перехід на виконання чергового поділу векторів на кластери.

Оцінка швидкодії

Для оцінки швидкодії кластеризації введемо наступні умовні позначення: *Tmp.* — тривалість виконання кластеризації згідно з традиційним алгоритмом; *Tрозр.* — тривалість виконання згідно з розробленим алгоритмом; *Tкр.пр.* — тривалість поділу векторів на кластери; *Tкр.онов.* — тривалість обчислення центру мас кластерів; N — кількість векторів у вхідному наборі даних; D — розмірність векторів; K — кількість кластерів, на які необхідно поділити вхідний набір даних; P — кількість скалярних ядер GPU; M — кількість ядер CPU; $t_=$ — тривалість однієї операції присвоєння; t_a — тривалість однієї арифметичної операції; $t_з$ — тривалість завантаження одного значення у пам'ять GPU; $t_в$ — тривалість вивантаження одного значення з пам'яті GPU.

Для традиційного алгоритму тривалість поділу векторів на кластери становить:

$$Ткр.пр. = N \cdot (K \cdot (D \cdot (2 \cdot t_{=} + 3 \cdot t_a) + 2 \cdot t_{=}) + 3 \cdot t_{=} + 2 \cdot t_a + D \cdot (t_{=} + t_a));$$

тривалість обчислення центру мас кластерів:

$$Ткр.онов. = K \cdot (D \cdot t_{=} + D \cdot (2 \cdot t_{=} + t_a) + D \cdot (t_{=} + t_a)) = K \cdot D \cdot (4 \cdot t_{=} + 2 \cdot t_a).$$

Для розробленого алгоритму тривалість поділу векторів на кластери становить:

$$Ткр.пр. = N \cdot (K \cdot (D \cdot (2 \cdot t_{=} + 3 \cdot t_a) + 2 \cdot t_{=}) + 3 \cdot t_{=} + 2 \cdot t_a + D \cdot (t_{=} + t_a)) / P + N \cdot D \cdot t_з + K \cdot D \cdot t_в;$$

тривалість обчислення центру мас кластерів:

$$Ткр.онов. = K \cdot (D \cdot t_{=} + (M-1) \cdot (2 \cdot t_{=} + t_a) + M \cdot D \cdot (2 \cdot t_{=} + t_a) + D \cdot (t_{=} + t_a)) / M.$$

Порівнюючи тривалість поділу векторів на кластери і обчислення центру мас кластерів для традиційного і розробленого алгоритму кластеризації отримуємо наступний результат:

$$Трозр. = Ткр.пр._{тр.} / P + N \cdot D \cdot t_з + K \cdot D \cdot t_в + Ткр.онов._{тр.} / M + (M-1) \cdot (2 \cdot t_{=} + t_a)$$

За умови, якщо кількість ядер GPU і CPU більше одиниці, доданки $N \cdot D \cdot t_з + K \cdot D \cdot t_в$ та $(M-1) \cdot (2 \cdot t_{=} + t_a)$, які відповідають витратам часу на виконання поділу даних і синхронізації між потоками, набагато менші ніж $Трозр. - (Ткр.пр._{тр.} / P + Ткр.онов._{тр.} / M)$ — різниця між тривалістю поділу векторів на кластери та обчислення центру мас кластерів для традиційного і розробленого алгоритму. Тобто

$$N \cdot D \cdot t_з + K \cdot D \cdot t_в << << (P-1) / P \cdot Ткр.пр._{тр.}, при\ P > 1$$

$$(M-1) \cdot (2 \cdot t_{=} + t_a) << << (M-1) / M \cdot Ткр.онов._{тр.}, при\ M > 1$$

Таким чином, розроблений алгоритм виконується за менший час і зменшення часу виконання залежить від кількості ядер GPU і ядер CPU.

Дослідження алгоритму

Експериментальне дослідження розробленого алгоритму кластеризації проводилося шляхом порівняння часу, що необхідний для кластеризації даних згідно з традиційним алгоритмом і згідно з розробленим алгоритмом, на комп'ютері з двоядерним процесором AMD Athlon II X2 265p 3,30GHz, 4ГБ оперативної пам'яті і відеокартою NVIDIA GeForce GT 240, яка містить 12 потокових мультипроцесорів (тобто 96 скалярних ядер) і дозволяє використовувати технологію CUDA.

Дослідження проводились на різних наборах вхідних даних: з кількістю елементів у наборі 40, 400, 4000, 40000, розмірністю елементів 300, 3000, 30000 і кількістю кластерів від 2 до 128. Результати дослідження представлені на рис. 1.

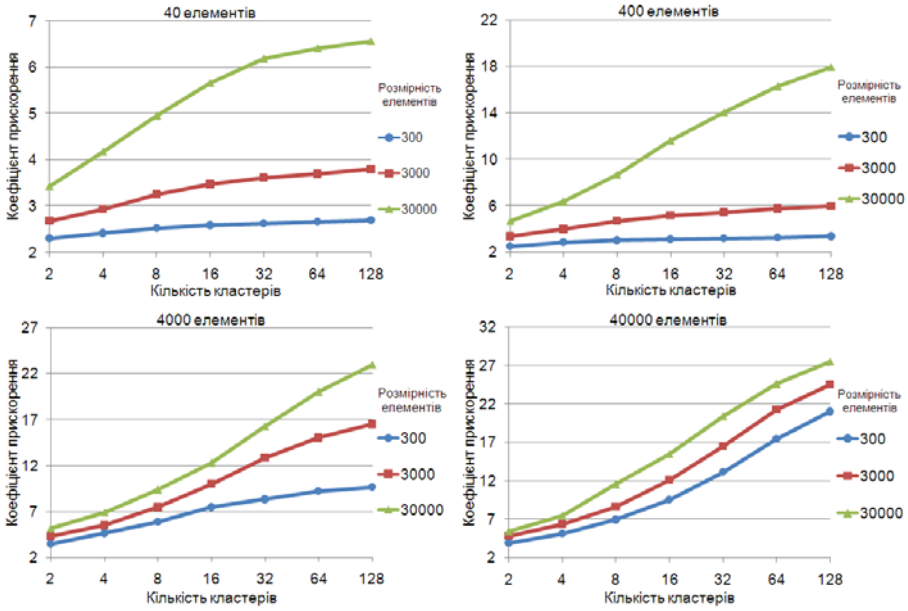

Рис. 1. Результати дослідження

Отримані результати кластеризації дозволяють зробити висновок, що при збільшенні кількості кластерів, елементів і розмірності даних, коефіцієнт прискорення, який характеризує підвищення продуктивності і дорівнює відношенню часу виконання кластеризації даних згідно з традиційним алгоритмом до часу виконання згідно з розробленим алгоритмом, збільшується від 2,3 до 27,5 разів. Таким чином, результати дослідження підтверджують наведені вище оцінки швидкодії.

Висновки

Запропонований у роботі алгоритм дозволяє підвищити продуктивність кластеризації даних у 2,3 — 27,5 разів за рахунок використання обчислювальних можливостей сучасних графічних процесорів з підтримкою технології CUDA та багатоядерних центральних процесорів.

Национальный технический университет Украины «Киевский политехнический институт»

Література

1. *Inderjit S.* Dhillon, Dharmendra S. Modha. A Data-Clustering Algorithm on Distributed Memory Multiprocessors // Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD. – 1999. – P. 17.
2. *Fazilah Othman,* Rosni Abdullah, Nur’Aini Abdul Rashid, Rosalina Abdul Salam. Parallel K-Means Clustering Algorithm on DNA Dataset // Parallel and Distributed Computing: Applications and Technologies. Lecture Notes in Computer Science. – 2005. – Volume 3320/2005. – P. 248-251.
3. *Jian Wan,* Wenming Yu1, Xianghua Xu. Design and Implement of Distributed Document Clustering Based on MapReduce // Proceedings of the Second Symposium International Computer Science and Computational Technology (ISCSCT ’09). – 2009. – P. 278-280.

Поступила в редакцию 18 октября 2012 г.