УДК 519.688

К.т.н. Замятін Д. С., студент Оберемчук В.М.

**Національний технічний університет України
«Київський політехнічний інститут»**

ПАРАЛЕЛЬНА РЕАЛІЗАЦІЯ АЛГОРИТМУ КЛАСТЕРИЗАЦІЇ K-MEANS ЗА ДОПОМОГОЮ БІБЛІОТЕКИ OPENMP

Abstract

Denis Zamyatin, PhD; Vitaliy Oberemchook, student
Parallel implementation k-means clustering algorithm using OpenMP library
This paper concerns the task of parallel implementation k-means clustering algorithm using OpenMP library. The comparative analysis of efficiency of both the parallel and the serial implementations is fulfilled. The experimental study and comparison of the performance of both implementations is investigated.

Вступ

Кластеризація — розбиття заданої вибірки об'єктів на підмножини, що називаються кластерами, так, щоб кожний кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Кластеризація використовується для систематизації великих обсягів інформації, що набуває важливості у зв’язку з зростанням об’ємів корпоративних сховищ документів і розповсюдженням *Web*-технологій.

Алгоритм кластеризації *k-means* є найбільш поширеним алгоритмом кластеризації, який характеризується тим, що виконує розподіл набору елементів векторного простору на заздалегідь відому кількість кластерів і має досить високу обчислювальну складність, яка експоненційно зростає зі збільшенням кількості і розмірності вхідних даних та збільшенням кількості кластерів.

В роботах [1], [2] і [3] розглядається паралельна реалізація алгоритму *k-means* за допомогою бібліотеки *MPI* та фреймворку *MapReduce* для застосування на обчислювальних кластерах. Ці реалізації дозволяють досягти досить високої швидкодії, але вимагають наявності спеціалізованого обчислювального кластера, використання якого передбачає залучення значних фінансових ресурсів.

В роботі [4] виконується розгляд паралельної реалізації алгоритму *k-means* з використанням технології *CUDA* шляхом виконання обчислень за допомогою графічного процесора. Недоліком використання цієї

реалізації алгоритму є необхідність наявності спеціалізованої графічної карти з підтримкою технології *CUDA*.

Таким чином, розглянуті паралельні реалізації алгоритму кластеризації *k-means* вимагають додаткового залучення значних ресурсів, що є недоцільним при використанні у корпоративних сховищах документів. Тому актуальною є розробка паралельної реалізації алгоритму кластеризації *k-means*, яка буде орієнтована на використання найбільш поширених технологій розпаралелювання, зокрема на основі багатоядерних процесорів.

Постановка задачі

Задача роботи полягає в розробці паралельної реалізації алгоритму кластеризації *k-means* за допомогою бібліотеки *OpenMP*, яку пристосовано до використання в системах з багатоядерними процесорами.

Алгоритм кластеризації *k-means*

Алгоритм *k-means* виконує розподіл набору елементів векторного простору на заздалегідь відому кількість кластерів *k*. Дія алгоритму спрямована на мінімізацію 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 $x_i \in S_i$.

Алгоритм *k-means* складається з наступних кроків:

1. Вибір початкових центрів мас кожного з k кластерів. Цей вибір виконується випадковим чином або відповідно до певної евристики.
2. Крок призначення. Вектори розподіляються на кластери у відповідності з тим, який з центрів мас виявився ближчим згідно обраної метрики.

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

3. Крок оновлення. Обчислюється центр мас для кожного кластера, отриманого на попередньому кроці

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

4. Якщо на останній ітерації не відбувається зміни кластерів, то алгоритм завершується, інакше виконується перехід на пункт 2.

Паралельна реалізація алгоритму кластеризації *k-means*

В алгоритмі кластеризації *k-means* можна виділити два кроки, які добре піддаються розпаралелюванню, а саме — кроки призначення та оновлення.

Крок призначення може бути виконаний паралельно для кожного елемента $x_i, i = 1,..., N$ з вхідного набору даних. N елементів даних розподіляються на P частин, приблизно рівного розміру, які буде оброблено за допомогою P незалежних потоків.

Щодо розпаралелювання кроку оновлення, то кожен з P потоків має зберігати і обчислювати локальні параметри для кожного кластера (центр мас кластера і кількість елементів у кластері). Головний потік має накопичувати нові локальні параметри кластерів кожного з P потоків та обчислювати нові глобальні центри мас для кожного кластера.

Оцінка швидкодії

При виконанні алгоритму кластеризації *k-means* основна частина часу витрачається на виконання кроку призначення та кроку оновлення.

Введемо наступні умовні позначення: $T_{посл.}$ — тривалість виконання послідовної реалізації алгоритму; $T_{пар.}$ — тривалість виконання послідовної реалізації алгоритму; $T_{кр.пр.}$ — тривалість кроку призначення; $T_{êð.î í î â.}$ — тривалість кроку оновлення; N — кількість векторів у вхідному наборі даних; D — розмірність векторів у вхідному наборі даних; K — кількість кластерів, на які необхідно розподілити вхідний набір даних; P — кількість потоків, що виконують кластеризацію; $t_=$ — тривалість однієї операції присвоювання; t_a — тривалість однієї арифметичної операції.

При послідовній реалізації алгоритму кластеризації *k-means*, тривалість кроку призначення складатиме

$$T_{кр.пр.} = N \cdot (K \cdot (D \cdot (2 \cdot t_= + 3 \cdot t_a) + 2 \cdot t_=) + 3 \cdot t_= + 2 \cdot t_a + D \cdot (t_= + t_a)),$$

тривалість кроку оновлення —

$$T_{êð.î í î â.} = K \cdot (D \cdot t_= + D \cdot (2 \cdot t_= + t_à) + D \cdot (t_= + t_à)) = K \cdot D \cdot (4 \cdot t_= + 2 \cdot t_à).$$

При паралельній реалізації алгоритму кластеризації *k-means* за допомогою бібліотеки OpenMP, тривалість кроку призначення —

$$T_{кр.пр.} = N \cdot (K \cdot (D \cdot (2 \cdot t_= + 3 \cdot t_a) + 2 \cdot t_=) + 3 \cdot t_= + 2 \cdot t_a + D \cdot (t_= + t_a)) / P,$$

тривалість кроку оновлення —

$$T_{кр.онов.} = K \cdot (D \cdot t_= + (P-1) \cdot (2 \cdot t_= + t_a) + P \cdot D \cdot (2 \cdot t_= + t_a) + D \cdot (t_= + t_a)) / P.$$

Таким чином, порівнюючи тривалість кроків призначення і оновлення при послідовній і паралельній реалізації алгоритму кластеризації *k-means* можна зробити висновок, що паралельна реалізація виконується за час $Tпар. = Tпосл./P + (P-1)\cdot(2\cdot t_{=} + t_a)$, причому зменшення часу виконання залежить від кількості паралельно виконуваних потоків, але разом з цим, використання паралельної реалізації алгоритму також вимагає деяких витрат часу на виконання розподілу даних і синхронізації між потоками.

Дослідження паралельного алгоритму

Експериментальне дослідження проводилося шляхом порівняння часу, необхідного для виконання кластеризації даних використовуючи послідовну і паралельну реалізацію алгоритму кластеризації *k-means* на комп'ютері з процесором *AMD Athlon II X2 265p* 3,30*GHz* і 4ГБ RAM.

Дослідження проводились з розмірністю вхідних векторів рівною 6 і кількістю кластерів рівною 6. Згідно з результатами наведеними на рис. 1 та рис. 2 можна зробити висновок, що при збільшенні кількості векторів збільшується час, необхідний для виконання кластеризації і коефіцієнт прискорення, що рівний відношенню часу виконання послідовної та паралельної реалізації алгоритмів. Таким чином, результати дослідження підтверджують наведені вище оцінки швидкодії.

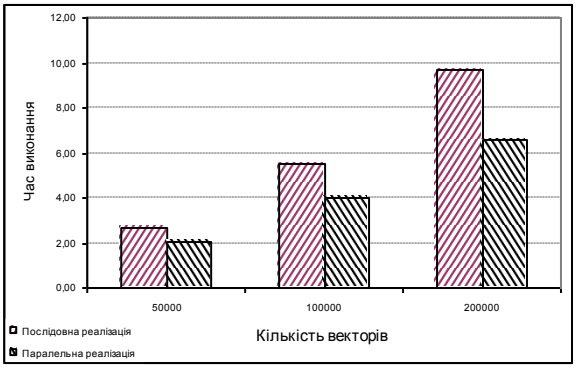

Рис. 1. Час виконання

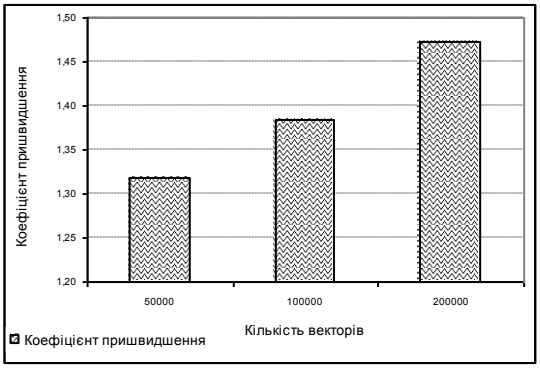

Рис. 2. Коефіцієнт прискорення

Висновки

У даній статті запропоновано паралельну реалізації алгоритму кластеризації *k-means* за допомогою бібліотеки *OpenMP*, що дозволяє підвищити швидкодію у 1,2-1,5 рази в порівнянні з послідовною реалізацією, за рахунок використання багатоядерних процесорів, що є доцільним для виконання кластеризації інформації з корпоративних сховищ документів.

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.

4. *Reza Farivar, Daniel Rebolledo, Ellick Chan, Roy Campbell*. A Parallel Implementation of K-Means Clustering on GPUs // Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 08). – 2008. – P. 340-345.