УДК 004.896.3

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ ДАНИХ В ПАРАЛЕЛЬНІЙ СИСТЕМІ УПРАВЛІННЯ БАЗАМИ ДАНИХ

Варчук Д.А.[1)], **Пришляк О.В.**[2)], **Виноградова Д.Р.**[3)], **Дементьєв Р.В.**[4)], **Марценюк М.О.**[5)]
Західноукраїнський національний університет
[1)]магістрант;[2)]аспірант;[3-5)]студент

I. Постановка проблеми

У сучасному інформаційному суспільстві одним із ключових феноменів, що має значний вплив на область обробки даних, є великі дані [1,2]. У зв'язку з розвитком сучасних технологій, таких як соціальні мережі, електронні бібліотеки, геоінформаційні системи та інші, виникає широкий спектр додатків, в яких накопичуються неструктуровані дані.

Системи управління базами даних (СУБД), побудовані на основі реляційної моделі даних залишаються основним і найбільш популярним інструментом для управління даними на сьогоднішній день. Ефективна обробка та аналіз великих сховищ даних вимагає використання паралельних СУБД на високопродуктивних обчислювальних системах[1]. Паралельні СУБД будуються на основі концепції фрагментного паралелізму, що передбачає розбиття таблиць баз даних на горизонтальні фрагменти, які можуть оброблятися незалежно на різних вузлах багатопроцесорних систем.

II. Мета роботи

Метою дослідження є розробка математичного забезпечення для нечіткої кластеризації даних в паралельній системі управління базами даних.

III. Метод нечіткої кластеризації даних в паралельній СУБД

Нечітка кластеризація даних у СУБД виконується за допомогою інтеграції алгоритму Fuzzy c-Means (FCM) [2] в паралельну СУБД відповідно до наступного сценарію. Вихідні дані, проміжні результати обчислень і вихідні дані алгоритму подаються у вигляді реляційних таблиць, які входять до бази даних зі спеціально розробленою схемою. Обчислення реалізуються у вигляді запитів SQL до відповідних таблиць. Таблиці бази даних піддаються горизонтальній фрагментації та розподіляються по вузлах кластерної обчислювальної системи. Кожен екземпляр паралельної СУБД запускається на власному вузлі і обробляє власний фрагмент таблиці.

У подальшому описі використовуються такі позначення:
$d \in N$ - розмірність простору об'єктів, що кластеризуються; $l \in N : 1 \le l \le d$ - номер координати об'єкта, що кластеризується; $n \in N$ - кількість об'єктів, що кластеризуються; $X \subset R^d$ - множина об'єктів, що кластеризуються; $i \in N : 1 \le j \le k$ - номер об'єкта; $x_i \in N$ - і-й об'єкт; $k \in N$ - кількість кластерів; $i \in N : 1 \le j \le k$ - номер кластера; $C \subset R^{k \times d}$ - матриця, що містить центри кластерів (матриця центроїдів); $c_j \subset R^d$ - центр кластера j; $x_{il}, c_{jl} \in R$ - координати об'єктів x_i, c_j відповідно; $U \subset R^{n \times k}$ - матриця ступенів приналежності, де число $u_{jj} \in R$ показує ступінь належності об'єкта x_i кластеру j і виконуються такі властивості:

$$\forall i, j[u_{ij}] \in [0,1] \forall i \sum_{j=1}^{k} u_{ij} = 1 \quad (1)$$

$\rho : R^d \times R^d \to R_+ \cup 0$ - функція відстані, яка використовується для визначення близькості об'єкта x_i та центроїда c_j (параметр алгоритму);

$m \in R : m > 1$ - показник нечіткості цільової функції (параметр алгоритму);

J_{FCM} – цільова функція.

Алгоритм FCM виконує мінімізацію цільової функції J_{FCM}, яка має такий вигляд:

$$J_{FCM}(X, k, m) = \sum_{i=1}^{N} \sum_{j=1}^{K} u_{i,j}^m, \rho^2(x_i, c_j) \quad (2)$$

Нечітке розбиття вихідної множини об'єктів досягається за допомогою мінімізації цільової функції (2). На кожній ітерації відбувається оновлення матриці центроїді C та матриці степенів належності U за такими формулами:

$$\forall i, lc_{jl} = \frac{\sum_{i=1}^{N} u_{i,j}^{m} x_{il}}{\sum_{i=1}^{N} u_{i,j}^{m}} \quad (3)$$

$$u_{i,j} = \sum_{t=1}^{K} \left(\frac{\rho^2(x_i, c_j)}{\rho^2(x_i, c_t)} \right)^{\frac{2}{1-m}} \quad (4)$$

Нехай s - номер ітерації, $u_{i,j}^{s}$ та $u_{i,j}^{s+1}$ елементи матриці U на s-му та $(s+1)$-му кроках відповідно, $\varepsilon \in (0,1) \subset R$ - порогове значення зупинки алгоритму.

Вхідними даними алгоритму FCM є множина об'єктів $X = (x_1, x_2, ..., x_n)$ кількість кластерів k, ступінь нечіткості m і граничне значення зупинки ε. Алгоритм повертає матрицю степенів належності U.

Множина наборів реалізується за допомогою класу vector зі стандартної бібліотеки класів C++ (Standard TemplateLibrary). Даний клас реалізує масив елементів, що належать одному типу даних та забезпечує операції додавання, видалення елементів, доступ до елемента за його індексом та ітерацію елементів масиву. Кожен із векторів *DASHED* і *SOLID* забезпечує зберігання двох множин наборів алгоритмів: *DashedCircle* і *DashedBox* та *SolidCircle* і *SolidBox* відповідно.

Зберігання двох множин у рамках одного вектора забезпечує менші витрати, ніж виділення одного вектора на кожну множину елементів: для переміщення елемента з одної множини до іншої достатньо змінити значення поля **shape**.

Набір елементів, таким чином, реалізується як структура з наступними основними полями: *Mask* - бітова маска набору; *k*- кількість елементів у наборі; *stop* - лічильник частини транзакцій, в яких перевірено наявність даного набору; *supp* - підтримка цього набору; *shape* - вид цього набору (*BOX* або *CIRCLE*, або *NULL*).

Алг. PDIC(IN $\mathcal{B}$, *minsup*, M; OUT $\mathcal{L}$)

1: $SOLID.init()$; $DASHED.init()$
2: **for all** $i \in 0..m-1$ **do**
3: $I.shape \leftarrow$ NIL
4: $I.mask \leftarrow SetBit(I.mask, i)$
5: $I.stop \leftarrow 0$; $I.supp \leftarrow 0$; $I.k \leftarrow 1$
6: $SOLID.push\_back(I)$
7: **end for**
8: $k \leftarrow 1$
9: $stop_{max} \leftarrow \lceil \frac{n}{M} \rceil$; $stop \leftarrow 0$
10: FIRSTPASS($SOLID$, $DASHED$)
11: **while not** $DASHED.empty()$ **do**
12: $stop \leftarrow stop + 1$
13: **if** $stop > stop_{max}$ **then**
14: $stop \leftarrow 1$
15: **end if**
16: $first \leftarrow (stop-1) \cdot M$; $last \leftarrow stop \cdot M - 1$
17: $k \leftarrow k + 1$
18: COUNTSUPPORT($DASHED$)
19: PRUNE($DASHED$)
20: MAKECANDIDATES($DASHED$)
21: CHECKFULLPASS($DASHED$)
22: **end while**
23: $\mathcal{L} \leftarrow \{I \mid I \in SOLID \wedge I.shape =$ BOX$\}$
24: **return** $\mathcal{L}$

Рисунок 1–Паралельна реалізація пошуку наборів

Розпаралелювання піддаються наступні стадії алгоритму: підрахунок підтримки, знаходження та відкидання наборів-кандидатів, які свідомо не є частими та перевірка завершення перегляду наборів-кандидатів по всій базі даних транзакцій. Підрахунок виконується за допомогою двох вкладених циклів: зовнішній розпаралелюється і виконується за наборами-кандидатами, а внутрішній - за транзакціями.

Висновок

Запропоновано метод нечіткої кластеризації даних за допомогою паралельної СУБД. Виконано проектування схеми бази даних, що дозволяє зберігати вихідні та проміжні дані в реляційних таблицях і виконувати обчислення за допомогою агрегатних функцій SQL. Результати проведених обчислювальних експериментів показують, що запропонований алгоритм для СУБД на великих обсягах даних є більш ефективним у порівнянні з традиційною реалізацією нечіткої кластеризації, що передбачає використання оперативної пам'яті.

Список використаних джерел

1. Рудакевич Т. М. Розширення функціональних можливостей СКБД Postgresql для оптимізації пошуку інформації у файлах баз даних / Т. М. Рудакевич, Г. Г. Цегелик // Вісник Національного університету "Львівська політехніка". – 2010. – № 673 : Інформаційні системи та мережі. – С. 181-194.
2/ Garcia, A.J.; Toril, M.; Oliver, P.; Luna-Ramirez, S.; Ortiz, M. Automatic Alarm Prioritization by Data Mining for Fault Management in Cellular Networks. ElsevierSci. DirectExpertSyst. Appl. 2020, 158, 113526.