УДК 519.6

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ КЛАСТЕРИЗАЦИИ БОЛЬШИХ ОБЪЕМОВ ДАННЫХ

Рожнов Иван Павлович, кандидат технических наук,
старший научный сотрудник научно-исследовательской лаборатории, Институт информатики и телекоммуникаций
Сибирский государственный университет науки и технологий им. академика М.Ф. Решетнёва, Красноярск, Россия
e-mail: ris2005@mail.ru
Казаковцев Лев Александрович, доктор технических наук, доцент,
профессор кафедры «Информационные технологии и математическое обеспечение информационных систем», Институт экономики и управления АПК
Красноярский государственный аграрный университет, Красноярск, Россия
e-mail: levk@bk.ru
Резова Наталья Леонидовна,
доцент кафедры «Информационно-управляющих систем», Институт информатики и телекоммуникаций
Сибирский государственный университет науки и технологий им. академика М.Ф. Решетнёва, Красноярск, Россия
e-mail: natalyakl@yandex.ru

Аннотация. Для сокращения времени расчетов при решении задач автоматической группировки большого объема данных в агропромышленном комплексе предложены параллельные алгоритмы с жадной агломеративной эвристической процедурой, адаптированные к архитектуре CUDA.
Ключевые слова: алгоритмы кластеризации, агропромышленный комплекс, CUDA, CPU, GPU, GPGPU, GH-VNS.

PARALLEL ALGORITHMS FOR CLUSTERING LARGE DATA

Rozhnov Ivan Pavlovich, candidate of technical sciences,
senior researcher of Research Laboratory, Institute of Informatics and Telecommunications
Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia
e-mail: ris2005@mail.ru
Kazakovtsev Lev Aleksandrovich, doctor of technical sciences, associate professor,
professor of the department of “Information Technologies and Mathematical Support of Information Systems”, Institute of economics and AIC management
Krasnoyarsk state agrarian university, Krasnoyarsk, Russia
e-mail: levk@bk.ru
Rezova Natalya Leonidovna, associate professor of the department of “Information and Control Systems”, Institute of Informatics and Telecommunications
Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia
e-mail: natalyakl@yandex.ru

Abstract. To reduce the computation time when solving problems of automatic grouping of large data in the agro-industrial complex, parallel algorithms with a greedy agglomerative heuristic procedure are proposed, adapted to the CUDA architecture.
Key words: clustering algorithms, agro-industrial complex, CUDA, CPU, GPU, GPGPU, GH-VNS.

Агропромышленный комплекс (АПК) - крупнейший межотраслевой комплекс, объединяющий отрасли экономики, направленные на производство и переработку сельскохозяйственного сырья и получения из него продукции, доводимой до конечного потребителя. Это совокупность отраслей экономики страны, включающая сельское хозяйство и отрасли промышленности, тесно связанные с сельскохозяйственным производством, осуществляющие перевозку, хранение, переработку сельскохозяйственной продукции, поставку ее потребителям, обеспечивающие сельское хозяйство техникой, химикатами и удобрениями, обслуживающие

сельскохозяйственное производство [1]. АПК играет существенную роль в удовлетворении потребностей населения.

Тем не менее, не смотря на важность АПК в развитии государства, для него характерны невысокая техническая укомплектованность и недостаточные темпы развития инфраструктурной базы в сфере информационного и научного обеспечения [3].

Распоряжением Правительства РФ от 12.04.2020 № 993-р утверждена «Стратегия развития агропромышленного и рыбохозяйственного комплексов Российской Федерации на период до 2030 года», в которой указано, что развитие цифровой экономики и цифровых технологий будет способствовать повышению эффективности работы АПК.

В АПК возможно широкое использование таких сквозных цифровых технологий, как:

- «искусственный интеллект» – для создания экспертных систем по почвам, системам удобрений, по распознаванию болезней и выработке рекомендаций по средствам защиты;

- «большие данные» – для анализа историй полей, физического и химического состава почв, сделок с земельными участками и маркетинговых исследований поставок сельскохозяйственной техники и оборудования, удобрений, средств защиты, оценки влияния рационов кормления на продуктивность сельскохозяйственных животных различных пород, а также почвенных и погодных условий на урожайности сортов сельскохозяйственных культур [3].

Одним из перспективных направлений в аналитике больших данных и искусственного интеллекта является кластерный анализ [1, 6].

В настоящее время используется большое количество алгоритмов кластеризации объектов. Основным отличием жадных агломеративных эвристических методов кластеризации является то, что они, представляя собой алгоритмы локального поиска в некоторой окрестности известного решения, выбирают в качестве следующего улучшенного решения тот вариант, который дает наибольшее уменьшение целевой функции (наибольшее увеличение – в случае максимизации).

В работе [7] был предложен метод жадных эвристик для задач автоматической группировки на моделях теории размещения, представленный ниже.

Алгоритм жадной агломеративной эвристической процедуры (количество кластеров более 50)

Требуется: начальное количество кластеров K, требуемое количество кластеров $k<K$, $k>50$, начальное решение S, $|S|=K$.

1: Улучшить решение S (если возможно).

пока $K \neq k$

для каждого $i' \in \{\overline{1,K}\}$

2: $S' = S \setminus \{X_{i'}\}$. Вычислить $F'_{i'} = F(S')$, где F – значение целевой функции.

конец цикла

3: Сформировать множество S_{elim} из n_{elim} центроидов, $S_{elim} \subset S$, $|S_{elim}| = n_{elim}$ с минимальными значениями $F'_{i'}$, где $n_{elim} = \max\{1, 0.2 \cdot (|S| - k)\}$.

4: Составить новое решение $S = S \setminus S_{elim}$, $K = K - 1$.

конец цикла

В упрощенном виде «жадный» алгоритм представляет собой некую последовательность этапов, на каждом из которых исследователь получает частичное решение исследуемой задачи, пока не получит полное решение, и алгоритм не закончит свою работу.

Огромное количество вычислений при выполнении описанного метода и большой объем данных избавляют от необходимости в больших кэшах, как при использовании центрального процессора. Однако при расчетах на графическом процессоре алгоритмы, использующие параллельную обработку данных большого объема, показывают очень хорошие результаты [9].

Исполнение параллельной обработки данных на графических процессорах получило большое развитие. Основные производители видеочипов (AMD и Nvidia) разработали соответствующие программно-аппаратные архитектуры: CTM (Close To Metal или AMD Stream Computing) и CUDA (Compute Unified Device Architecture), соответственно. Обе эти архитектуры устранили некоторые из серьезных недостатков предшествующих моделей программирования графических процессоров - они имеют прямой доступ к аппаратным возможностям видеочипов.

Стандарты, использующие платформонезависимую спецификацию OpenGL, кажутся наиболее универсальными, они позволяют использовать один и тот же программный код для видеочипов разных производителей. Зато такие методы значительно менее гибкие и не такие удобные в использовании. Кроме того, они не позволяют использовать специфические возможности

определенных видеокарт в современных вычислительных процессорах - например, такие как глобальная и быстрая разделяемая виды памяти.

Рассмотрим более подробно платформу, предложенную компанией Nvidia. CUDA - это технология на базе программно-аппаратной архитектуры, которая позволяет повысить производительность параллельных вычислений. Параллельные вычисления - это вычисления, при которых процесс разработки программного обеспечения делится на потоки. Потоки обрабатываются параллельно и взаимодействуют между собой в процессе обработки. В основе параллельных вычислений CUDA лежит технология GPGPU.

GPGPU (General-Purpose computing on Graphics Processing Units) - это технология, которая позволяет использовать графический процессор GPU в операциях, которые обычно выполняет центральный процессор CPU, например, в математических вычислениях. С помощью GPGPU можно использовать видеокарту для выполнения неграфических вычислений. При этом графический процессор будет работать не вместо центрального, а в качестве вычислительного блока. CUDA является улучшенной вариацией GPGPU.

Модель программирования CUDA представляет собой специальный упрощенный диалект языка программирования C со своим компилятором командной строки nvcc и библиотеками численного анализа FFT и BLAS для вычислений на GPU [4].

Эта технология поддерживает несколько языков программирования. Среди них Java, Python и некоторые другие.

Основные понятия CUDA:

1. Хост (host) – центральный процессор и его память – запускает различные задания, выделяет память.

2. Устройство (device) – сама видеокарта, графический процессор и его память – выполняет команды центрального процессора.

3. Ядро (kernel) – функция (задание), предназначенная для выполнения на GPU.

4. Пользователь самостоятельно запускает с CPU ядра на GPU.

5. Перед выполнением ядра пользователь копирует данные из памяти хоста в память GPU.

6. После выполнения ядра пользователь копирует данные из памяти GPU в память хоста.

Если говорить максимально упрощенно, то работа алгоритма CUDA выглядит следующим образом:

1. Центральный процессор (хост) выделяет нужное количество памяти и отправляет её графическому процессору.

2. Центральный процессор (хост) запускает ядро и также «делится» им с графическим.

3. Графический процессор выполняет ядро.

4. После обработки данных центральный процессор (хост) принимает результаты.

CUDA имеет преимущества не только перед вычислениями на CPU, но и перед более ранними технологиями вычисления с помощью GPGPU, а именно:

1. Простой в применении интерфейс программирования на C.

2. Битовые и целочисленные операции проводятся на аппаратном уровне, не требуя переноса алгоритмов в удобный для концепции графического конвейера вид.

3. Не ограничена графическими API.

4. Эффективный взаимообмен между видеопамятью и системной памятью.

5. Память размером 16 Кб на мультипроцессор: её можно разделить на потоки и настроить кэш с широкой полосой пропускания.

Но, как любая технология, CUDA имеет нюансы применения и ограничения:

1. CUDA имеет архитектуру с закрытым исходным кодом, которым владеет NVIDIA.

2. Архитектура работает только на видеочипах компании NVIDIA, начиная с версии GeForce 8.

3. Для выполняемых функций нет поддержки рекурсии.

4. 32 потока - минимально возможная ширина блока.

Однако важно отметить, что видеокарты Geforce очень широко распространены в настоящее время. Кроме того, технология CUDA, являясь кроссплатформенным программным обеспечением, доступна на широко распространенных 32-битных и 64-битных операционных системах Linux, Windows и MacOS X.

Эффективность алгоритма кластеризации k-средних на больших объемах данных падает, в частности когда для нахождения параметра k требуется несколько запусков алгоритма с разным количеством кластеров. Помимо этого, базовая жадная агломеративная эвристическая процедура

подразумевает неоднократный запуск алгоритма k-средних (или любого другого алгоритма локального поиска), и количество этих запусков увеличивается с квадратичной зависимостью от роста числа кластеров. Предлагается применять оптимизированную под GPU стратегию для k-средних, а также приближенную к архитектуре CUDA процедуру исключения кластеров из решения, которая является наиболее вычислительно «дорогим» шагом в жадной агломеративной эвристической процедуре [8].

Для этого Шаг 2 алгоритма был реализован на графическом процессоре.

Реализация Шага 2 базовой жадной агломеративной эвристической процедуры на платформе CUDA

$l = blockIdx.x \times blockDim.x + threadIdx.x$, где l – номер кластера.

Если $l>k$ тогда возврат.

Рассчитать ΔD_l в соответствии с $\Delta D_l = \begin{cases} 0, & C_{i'} \neq l, \\ \left(min_{j \in \{\overline{1,k}\},\, j \neq i'} \left\| A_j - X_j \right\|^2 \right) - \left\| A_j - X_{C_{i'}} \right\|^2, & C_{i'} = l. \end{cases}$

Если ΔD_l >0 то atomicAdd(*sumD*, ΔD_l).

Синхронизировать потоки.

Для вычислений было использовано 512 потоков (число подобрано опытным путем) для каждого блока CUDA, количество блоков рассчитывается как $N_{blocks}=(N+N_{threads}-1)/N_{threads}$. Начальное значение переменной *sumD* равно нулю. Затем для каждого вектора данных запускается алгоритм и вычисляется ΔD_l (рисунок 1).

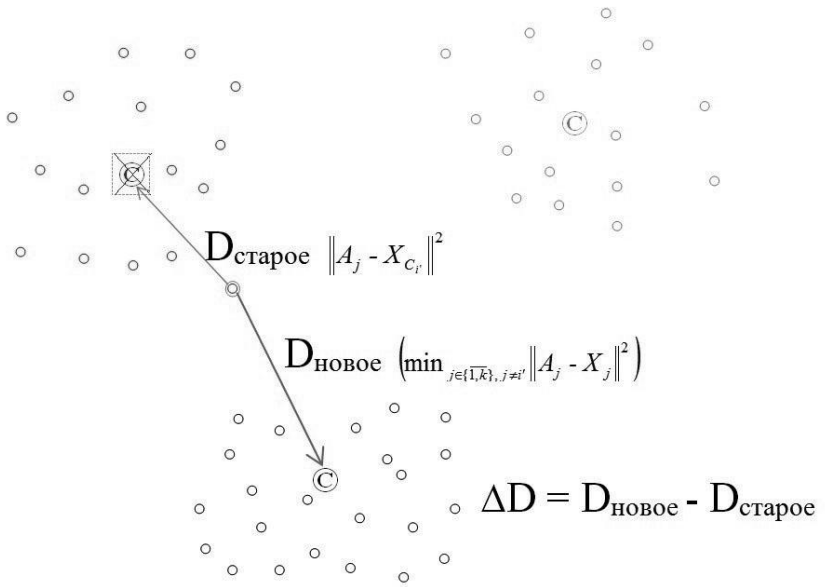

Рисунок 1 - Расчет приращения расстояния ΔD при удалении центроида

Для проведения расчетов использовались наборы данных из репозиториев Clustering basic benchmark и UCI [5]. Для полноты эксперимента были подобраны сеты разного объема из различных областей жизни.

При кластеризации данных было выполнено по 30 попыток запуска каждого из алгоритмов. Алгоритмы *k*-средних и *j*-средних были запущены в мультистартовом режиме. Результаты работы алгоритмов приведены в таблице 1.

В таблице 1 использованы следующие аббревиатуры и сокращения: GH-VNS - алгоритм автоматической группировки с комбинированным применением алгоритмов поиска с чередующимися рандомизированными окрестностями и жадных агломеративных эвристических процедур.

После всех расчетов для алгоритма *k*-средних по набору данных BIRCH3 без использования технологии CUDA [10] было получено лучшее минимальное значение целевой функции 3.72525E+13, при этом на каждую попытку отводилось по 6 часов. При расчетах с использованием GPU получено минимальное значение целевой функции 3.71973E+13 тем же алгоритмом, но за 1 минуту. Как видно результат при использовании графического процессора получился точнее, а время затрачено на порядки меньше (в приведенном примере в сотни раз).

Таблица 1 - Результаты вычислительных экспериментов

Алгоритм	Значение целевой функции		
	Минимальное	Среднее	Среднеквадратичное отклонение
CPU 6 часов			
k-средних	7.92474E+13	8.31599E+13	3.088140E+12
j-средних	3.76222E+13	3.77715E+13	0.116211E+12
k-GH-VNS1	3.72537E+13	3.74703E+13	0.171124E+12
k-GH-VNS2	4.21378E+13	4.52349E+13	4.333462E+12
k-GH-VNS3	***3.72525E+13***	***3.73745E+13***	***0.074315E+12***
GPU 1 минута			
k-средних	8.18676E+13	8.98255E+13	8.37212E+12
j-средних	5.30805E+13	7.91183E+13	28.2000E+12
k-GH-VNS1	***3.71973E+13***	3.73639E+13	0.18509E+12
k-GH-VNS2	3.73240E+13	3.91485E+13	1.14305E+12
k-GH-VNS3	3.72082E+13	***3.72422E+13***	***0.01998E+12***

Таким образом, было показано, что параллельная реализация алгоритмов кластеризации может применяться при обработки больших объемов данных в агропромышленном комплексе, что позволит сократить время расчетов алгоритмов автоматической группировки в десятки и сотни раз без ухудшения значения целевой функции.

Список литературы

1. Алгоритмы автоматической группировки с повышенными требованиями к точности и стабильности результата / И. П. Рожнов, Л. А. Казаковцев, В. И. Орлов, Д. Л. Михнев ; Сибирский государственный университет науки и технологий им. акад. М.Ф. Решетнева. Москва : Издательский Дом "Инфра-М". 2020. 192 с. (Научная мысль). ISBN 9785160166414.

2. Бурсулая Т. Агропромышленный комплекс / Тенгиз Бурсулая // "Финансовая газета". - 2019. - №19.

3. О долгосрочной стратегии развития агропромышленного комплекса Российской Федерации (подготовлен по итогам заседания Научно-методического семинара Аналитического управления Аппарата Совета Федерации, 19 апреля 2018 года) / [М.О. Орлов и др.]; под ред. начальника Аналитического управления Аппарата Совета Федерации, доктора экономических наук В.Д. Кривова // Аналитический вестник. - 2018. - №10 (699). 122 с.

4. Bhusare B.B. Initialization for K-Means Clustering using Improved Pillar Algorithm / Bhusare B.B., Bansode S.M. // International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 2014. Т. 3. Вып. 4.

5. Dua D. and Graff C. UCI Machine Learning Repository. - Irvine, CA: University of California, School of Information and Computer Science, 2019. - URL : http://archive.ics.uci.edu/ml (дата обращения: 20.09.2021)

6. Jain A.K. Data clustering: 50 years beyond K-means / A.K. Jain // Pattern Recognition Letters. 2010. Т. 31. С. 651-666.

7. Kazakovtsev L.A. Genetic Algorithm with Fast Greedy Heuristic for Clustering and Locagtion Problems / Kazakovtsev L.A., Antamoshkin A.N. // Informatica (Ljubljana). 2014. Т. 38. № 3. С. 229-240.

8. Kazakovtsev L.A. Parallel implementation of the greedy heuristic clustering algorithms / Kazakovtsev L.A., Rozhnov I.P., Popov E.A., Karaseva M.V., Stupina A.A. // IOP Conf. Series: MIP: Engineering. 2019. Т. 537.

9. Luebke D. How GPUs work / Luebke D., Humphreys G. // Computer. 2007. Т. 40. Вып. 2. С. 96–100.

10. Rozhnov I. P. VNS-based algorithms for the centroid-based clustering problem / Rozhnov I. P., Orlov V. I., Kazakovtsev L. A. // Facta Universitatis. Ser. Math. Inform. 2019. Т. 34. № 5. С. 957–972.