УДК 004.932.2; 004.42; 004.032.24; 528.854.2

ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ NVIDIA ПРИ КЛАСТЕРИЗАЦИИ МУЛЬТИСПЕКТРАЛЬНЫХ ДАННЫХ СЕТОЧНЫМ АЛГОРИТМОМ CCA

Сергей Александрович Рылов
Институт вычислительных технологий СО РАН, 630090, Россия, г. Новосибирск, пр. Академика Лаврентьева, 6, аспирант, e-mail: RylovS@mail.ru

Игорь Алексеевич Пестунов
Институт вычислительных технологий СО РАН, 630090, Россия, г. Новосибирск, пр. Академика Лаврентьева, 6, заведующий лабораторией обработки данных, тел. (383)334-91-55, e-mail: pestunov@ict.nsc.ru

В работе рассматривается реализация сеточного алгоритма кластеризации CCA на графических процессорах NVIDIA с использованием технологии CUDA. Эффективное использование архитектурных особенностей графических ускорителей позволило существенно увеличить быстродействие алгоритма, что делает его удобным инструментом для кластеризации мультиспектральных данных. Приводится анализ производительности алгоритма при работе на графическом ускорителе, на одном и на нескольких ядрах центрального процессора.

Ключевые слова: кластеризация мультиспектральных данных, сеточный подход, параллельные вычисления, CUDA, GPU, графический ускоритель.

NVIDIA GPU FOR MULTISPECTRAL DATA CLUSTERING WITH GRID-BASED ALGORITHM CCA

Sergey A. Rylov
Institute of Computational Technologies SB RAS, 630090, Russia, Novosibirsk, 6 Acad. Lavrentjev ave., postgraduate student, e-mail: RylovS@mail.ru

Igor A. Pestunov
Institute of Computational Technologies SB RAS, 630090, Russia, Novosibirsk, 6 Acad. Lavrentjev ave., head of data processing laboratory; tel. (383)334-91-55, e-mail: pestunov@ict.nsc.ru

The present work explores parallel computation of grid clustering algorithm CCA with CUDA/GPU. Effective use of the graphic processors architecture has significantly increased the performance of the algorithm, which makes it a convenient tool for multispectral image clustering. Computational results are presented, showing the algorithm computing time on the GPU in comparison with single core of the CPU and parallel implementation on the CPU with OpenMP.

Key words: multispectral images, grid-based clustering, parallel computing, CUDA, GPU.

Введение. При решении целого ряда прикладных задач возникает необходимость кластеризации больших массивов данных при отсутствии каких-либо априорных сведений об искомых классах. Например, в задачах обработки мультиспектральных спутниковых изображений, содержащих десятки миллионов

пикселей. В этих условиях целесообразно использовать быстрые непараметрические алгоритмы кластеризации [1].

В настоящее время большинство настольных персональных компьютеров оснащены современными видеокартами, производительность которых в последнее десятилетие стремительно растет. На фоне этого активно развиваются технологии вычислений общего назначения на графических процессорах (ГП), позволяющие использовать их для решения сложных математических задач.

В данной работе рассматривается сеточный алгоритм кластеризации CCA [2] и его реализация с использованием технологии CUDA [3].

Алгоритм кластеризации CCA(*m*,*T*) [2] разработан в рамках комбинации сеточного [4] и плотностного [1] подходов и характеризуется высоким быстродействием и возможностью выделять кластеры сложной структуры.

Для его описания введем следующие определения.

Пусть множество объектов X состоит из векторов, лежащих в пространстве признаков R^d: $X = \{x_i = (x_i^1,...,x_i^d) \in R^d, i = \overline{1,N}\}$, и ограниченных гиперпараллелепипедом $\Omega = [l^1, r^1] \times ... \times [l^d, r^d]$: $l^j = \min_{x_i \in X} x_i^j$, $r^j = \max_{x_i \in X} x_i^j$. Под *сеточной структурой* будем понимать разбиение пространства признаков на *клетки* гиперплоскостями: $x^j = (r^j - l^j) \cdot i / m + l^j$, $i = 0,...,m$, где m – число разбиений Ω по каждой размерности. Под *плотностью клетки* D_B будем понимать количество элементов множества X, попавших в клетку B.

Непустая клетка B_i *непосредственно связна* с непустой клеткой B_j $(B_i \to B_j)$, если клетка B_j обладает максимальной плотностью среди всех клеток, смежных с B_i. Клетки B_i и B_j *непосредственно связны* $(B_i \leftrightarrow B_j)$, если $B_i \to B_j$ или $B_j \to B_i$. Клетки B_i и B_j *связны* $(B_i \sim B_j)$, если существуют $k_1,...,k_l$ такие, что $k_1 = i$, $k_l = j$ и для всех $p = 1,...,l-1$ выполнено $B_{k_p} \leftrightarrow B_{k_{p+1}}$. Введенное отношение связности порождает естественное разбиение множества непустых клеток на компоненты связности $G_1,...,G_S$. *Представителем* компоненты связности G назовем клетку $Y(G)$: $Y(G) = \arg\max_{B \in G} D_B$.

Смежные компоненты связности G_i и G_j связны $(G_i \leftrightarrow G_j)$, если существует цепочка клеток $P_{ij} = Y_i = B_{k_1},...,B_{k_t}...,B_{k_l} = Y_j$ такая, что: 1) для всех $t = 1,...,l-1$: $B_{k_t} \in G_i \cup G_j$ и $B_{k_t}, B_{k_{t+1}}$ – смежные клетки; 2) $\min_{B_{k_t} \in P_{ij}} D_{B_{k_t}} / \min(D_{B_{Y_i}}, D_{B_{Y_j}}) > T$, $T > 0$ – порог объединения. Компоненты G_i и G_j связны $(G_i \sim G_j)$, если существуют $k_1,...,k_l$ такие, что $k_1 = i$, $k_l = j$ и для всех $p = 1,...,l-1$ выполнено $G_{k_p} \leftrightarrow G_{k_{p+1}}$. *Кластером* назовем максимальное множество попарно связных компонент связности.

Алгоритм ССА можно условно разбить на три основных этапа.

1. Формирование сеточной структуры в пространстве признаков. Точки $x_i \in X$ распределяются по клеткам, и вычисляются плотности всех клеток.

2. Разбиение клеток сеточной структуры на компоненты связности $G_1,...,G_S$ и определение их представителей $Y(G_1),...,Y(G_S)$.

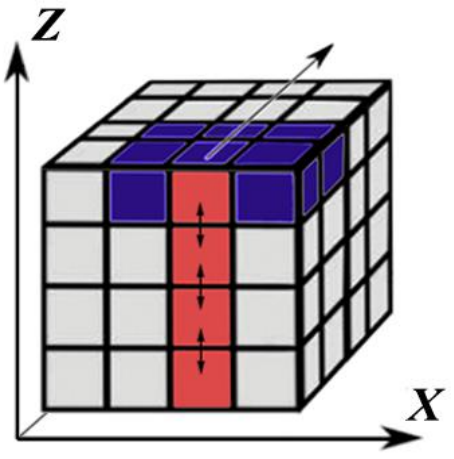
3. Формирование кластеров из выделенных компонент связности.

Реализация алгоритма ССА с использованием архитектуры параллельных вычислений CUDA [3]. Современные ГП содержат тысячи ядер, сгруппированных в блоки под управлением мультипроцессоров. Ядра одного блока выполняют одинаковый набор инструкций, но на различных данных. Каждое ядро имеет небольшое число регистров, а также доступ к ограниченному объему быстрой разделяемой памяти внутри своего блока. Кроме того, всем потокам (исполняемым на отдельных ядрах) доступна медленная глобальная память видеокарты. Синхронизация потоков во время выполнения возможна только в рамках своего блока.

В реализации алгоритма ССА выделяются шесть основных этапов.

На первом этапе осуществляется вычисление плотностей клеток в формируемой сетке. Этот процесс можно рассматривать как вычисление многомерной гистограммы. Известные подходы для построения гистограммы с помощью CUDA основаны на формировании в разделяемой памяти множества гистограмм для отдельных порций данных с последующим их объединением. Этот подход используется, чтобы сократить задержки произвольного доступа к массиву гистограммы. Однако при вычислении многомерной гистограммы использование разделяемой памяти невозможно из-за ее ограниченного размера (48 КБ). Поэтому был выбран подход, при котором каждый поток обрабатывает один элемент данных и использует атомарную операцию инкремента для заполнения массива гистограммы в глобальной памяти. Эффективность атомарных операций существенно возросла на ГП с современной архитектурой Kepler, что позволяет успешно использовать их в данной задаче.

На втором этапе для каждой клетки выявляется смежная с ней клетка с максимальной плотностью. При реализации в рамках CUDA каждая клетка обрабатывается одним потоком, просматривающим соседние с ней клетки. Для увеличения производительности был использован ряд оптимизаций.



- Использование разделяемой памяти для обмена результатами. Потоки каждого блока выровнены в линию вдоль оси Z, и каждый поток просматривает только соседние клетки с фиксированной координатой z. Затем происходит обмен промежуточными результатами между соседними потоками.

- Отсутствие повторного считывания элементов. Каждый поток последовательно обраба-

тывает группу элементов, смещаясь по оси Y. При этом используются промежуточные максимумы, что позволяет просматривать только 1/3 часть элементов, требуемых для определения максимума в новой позиции.

На данном этапе также определяются представители компонент связности. Их количество и нумерация отслеживаются с помощью счетчика в глобальной памяти через атомарную операцию инкремента.

На третьем этапе выполняется выделение компонент связности. Каждый поток обрабатывает одну клетку, последовательно переходя по ссылкам к смежным клеткам с максимальной плотностью вплоть до достижения одного из представителей компонент связности.

На четвертом этапе формируется матрица связей между смежными компонентами связности. Согласно построению наименьшая плотность между смежными компонентами достигается на их границе. Поэтому при параллельной обработке пар смежных клеток, в случае, если клетки принадлежат различным компонентам, вычисляется значение перепада плотности между компонентами по рассматриваемым клеткам. Если полученное значение превышает заданный порог T, то рассматриваемые компоненты помечаются связными.

На пятом этапе происходит объединение связных компонент в кластеры. В разделяемой памяти формируется массив из компонент связности, содержащий ссылки, ведущие на представителей кластеров (определяется как минимальная по номеру компонента в кластере). На каждой итерации потоки параллельно осуществляют проверку связей от определенной компоненты до всех последующих, формируя соответствующие ссылки. Данная работа разделена на независимые блоки, согласование результатов которых выполняет первый блок.

На заключительном этапе алгоритма точки исходных данных распределяются по кластерам, к которым отнесены содержащие их клетки.

Результаты экспериментальных исследований. В данном разделе представлено сравнение времени работы алгоритма CCA и его отдельных этапов при исполнении на ГП (GeForce GXT 770), одном и четырех ядрах ЦП (Intel Core i7 960, 3.2 ГГц). Параллельная версия алгоритма для ЦП реализована с помощью стандарта OpenMP. Используемые параметры CCA: m=32, T=0.8.

В табл. 1 представлены результаты обработки 5 000 000 трехмерных и четырехмерных случайных данных. На четырехмерных данных значительное время занимает формирование кластеров из компонент связности, что обусловлено их большим количеством в неструктурированных случайных данных. При обработке мультиспектральных изображений, количество компонент связности, как правило, не превосходит 2 000. При этом основное время работы алгоритма занимает первый этап, который эффективно ускоряется с помощью ГП (рис. 1).

В табл. 2 приведено сравнение времени обработки RGB-изображений и четырехканальных спутниковых снимков различного размера.

Таблица 1

Время работы (в мс) алгоритма CCA на случайных данных (5 млн. элементов)

Этап обработки	$d = 3$			$d = 4$		
	ЦП, 1 ядро	ЦП, 4 ядра	ГП	ЦП, 1 ядро	ЦП, 4 ядра	ГП
1. Формирование сеточной структуры	28.29	45.84	1.07	54.03	41.07	9.56
2. Анализ плотностей смежных клеток	0.85	0.27	0.07	50.39	13.36	0.44
3. Разделение клеток на компоненты	0.09	0.06	0.05	2.71	0.78	0.49
4. Проверка связей смежных компонент	1.52	0.42	0.15	107.53	28.50	3.27
5. Объединение компонент в кластеры	0.48	0.25	0.43	34.76	12.37	2.92
6. Распределение данных по кластерам	6.77	2.66	1.47	21.82	7.9	4.14
Суммарное время кластеризации	**38.00**	**49.50**	**3.24**	**271.24**	**103.98**	**20.82**

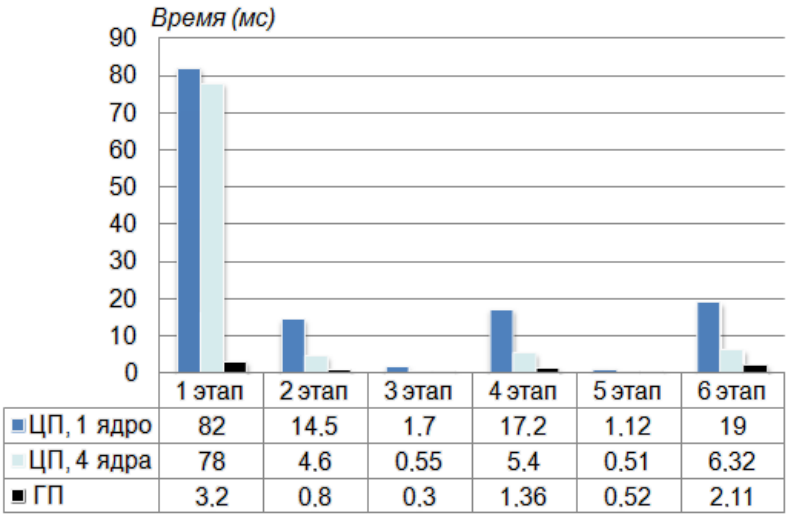


Рис. 1. Сравнение времени работы (в мс) этапов алгоритма CCA при обработке спутникового изображения WorldView-2 (4 спектральных канала, 9 Мп)

Таблица 2

Время работы (в мс) алгоритма CCA на изображениях

	RGB-изображение (3 канала)			Спутниковый снимок (4 канала)				
Размер изображения (млн. пикселей)	5	12.4	115.3	4	9	12.5	24	100
ЦП, 1 ядро	50	87	701	80	144	150	273	931
ЦП, 4 ядра	45	75	619	55	97	130	213	620
ГП	2.6	4.7	38	4.8	7.6	14	21	61

Заключение. В работе рассмотрена реализация алгоритма кластеризации CCA на графических процессорах с использованием технологии CUDA. Результаты экспериментальных исследований показали, что использование ГП позволяет увеличить быстродействие кластеризации мультиспектральных данных в 10 и более раз. Данное исследование показывает перспективность использования графических ускорителей для обработки спутниковых данных.

Работа выполнена при финансовой поддержке РНФ (грант № 14-14-00453) и РФФИ (гранты № 13-07-12202-офи_м и № 14-07-31320-мол_а).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Пестунов И. А., Синявский Ю. Н. Алгоритмы кластеризации в задачах сегментации спутниковых изображений // Вестник КемГУ. – 2012. – № 4/2 (52). – С. 110–125.

2. Куликова Е. А., Пестунов И. А., Синявский Ю. Н. Непараметрический алгоритм кластеризации для обработки больших массивов данных // Конф. «Математические методы распознавание образов»: сб. тр. – Москва: Изд-во MAKS Press, 2009. – С. 149-152.

3. Параллельные вычисления на GPU. Архитектура и программная модель CUDA / А. Боресков, А. Харламов, Н. Марковский и др. – Москва: Изд-во Московского университета, 2012. – 336 с.

4. Ilango M. R., Mohan V. A survey of grid based clustering algorithms // Intern. J. Eng. Sci. and Technology. – 2010. – Vol. 2(8). – P. 3441–3446.