Параллельная реализация алгоритма K-MEANS ++ с использованием технологии OpenCL

Г.А. Максимов, Н.Н. Богословский
Томский государственный университет

K-MEANS является широко используемым методом кластеризации, в ходе которого предпринимается попытка свести к минимуму среднеквадратичное расстояние между точками кластеров и их центрами. Хотя гарантии точности отсутствуют, простота и скорость работы алгоритма становятся приоритетными факторами применения его на практике. Выбирая начальные центры для K-MEANS очень простым вероятностным алгоритмом, можно получить прирост как скорости работы, так и точности [1]. При больших объемах входных данных время, затрачиваемое на выбор центров, становится велико для использования алгоритма в режиме реального времени, что обязывает его распараллеливать. В качестве технологии параллельного программирования используется язык-надстройка для C++ – OpenCL (Open Computing Language).

Введение

Пусть $X \subset R^q$ – конечное множество и k – целое число. Задача кластеризации заключается в разбиении исходного множества X на k подмножеств, состоящих из элементов, схожих по какому-либо признаку. Такие подмножества называются кластерами, а в качестве признака может служить ближайший центр кластера.

Наиболее популярным методом кластеризации является алгоритм K-MEANS (иногда называемый k-средних), предложенный 1950-х годах почти одновременно Г. Штейнгаузом и С. Ллойдом [5]. Суть алгоритма заключается в стремлении минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$\Phi = \sum_{i=1}^{k} \sum_{x_j \in C_i} \left\| x_j - c_i \right\|^2 ,$$

где k – число кластеров; C_i, $i = \overline{1,k}$ – кластеры; c_i – центры масс точек кластеров.

Результат работы алгортма K-MEANS зависит от начальных центров, выбираемых случайным образом, и часто бывает неудовлетворительным. Существует эффективный способ выбора начальных центров

(центройдов) для описанной выше техники кластеризации. Он был предложен Д. Артуром и С. Вассиливицким в 2009 г. и получил название K-MEANS++ [1]. Алгоритм состоит из трех основных этапов:

1. Первый центр выбирается случайным образом из числа входных данных X.
2. Точка $x_i \in X$ становится следующим центром с вероятностью $D(x_i)^2 \Big/ \sum_{x \in X} D(x)^2$, где $D(x) = \min_{c \in C} \|x - c\|$ – расстояние до ближайшего имеющегося центра $c \in C$.
3. Второй шаг повторяется до тех пор, пока не выбрано нужное количество центров.

Выбор центров подобным образом улучшает как скорость работы, так и качество результата K-MEANS [1], но при больших объемах входных данных, которые возникают при обработке изображений, время работы алгоритма K-MEANS++ становится велико, и его сложно использовать в программах, работающих в режиме реального времени. Поэтому необходимо распараллеливать данный алгоритм.

При реализации было решено уйти от вероятностного выбора точки в качестве центра, отдавая предпочтение поиску максимума среди величин $D(x)$. Таким образом, поиск центров представляет собой полный перебор точек среди входных данных, и схема работы алгоритма выглядит следующим образом:

1. Первый центр выбирается случайным образом из числа входных данных X.
2. Точка $x_i \in X$ становится следующим центром, если $D(x_i) = \max_{x \in X} D(x)$.
3. Второй шаг повторяется до тех пор, пока не выбрано нужное количество центров.

Параллельная реализация

Так как параллельную программу планируется запускать на графических процессорах, необходимо учитывать ограничения, возникающие при вычислениях: малый объем оперативной памяти (по сравнению с многопроцессорными вычислительными системами) и значительные временные затраты на копирование данных (на устройство и обратно).

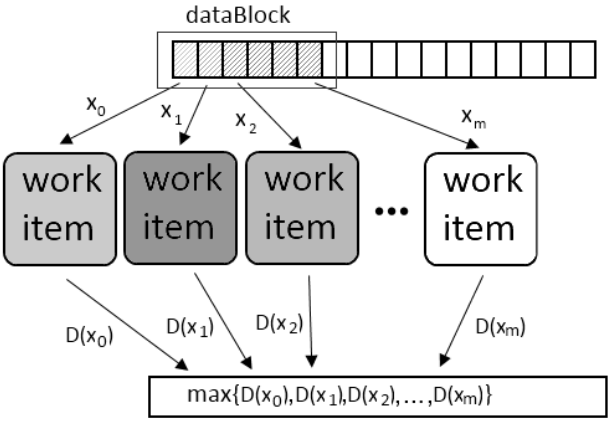

Рис. 1. Схема вычислений параллельной программы

Основная идея параллельной реализации заключается в возможности независимо вычислять величину $D(x)$ для каждой точки $x \in X$ и необходимости делить исходное множество данных X на N непересекающихся подмножеств (блоков) B_n, $n = \overline{1, N}$, размер которых допускает их целостное копирование. Определим подзадачу как отыскание наиболее удаленной от имеющихся центров точки, т.е. выбор одного центра. Для ее решения необходимо последовательно скопировать N блоков данных на устройство. Работу параллельной программы для выбора одного центра можно разделить на следующие основные этапы:

1. Копирование блока B_n в память видеокарты.
2. Параллельное вычисление дистанций $D(x)$ для всех $x \in B_n$.
3. Копирование массива дистанций $D(x)$ в ОЗУ компьютера.
4. Нахождение $D_n(x) = \max\limits_{x \in B_n} D(x)$.
5. Определение максимума среди уже найденных $D_n(x)$.
6. Пункты 1–4 повторяются N раз.

Для решения указанной подзадачи каждая нить (*work-item*) будет обрабатывать одну точку блока входных данных, скопированных на устройство. Схема обработки одного блока данных изображена на рис. 1.

В целях снижения временной стоимости операций копирования задействована закрепленная память (*pinned/non-pageable memory*).

Результаты

Тестирование проводилось с переменным количеством требуемых центров на различных наборах данных на видеокарте Nvidia GTX 460. Среднее время работы (в секундах) последовательного (CPU) и параллельного (GPU) алгоритмов представлено в таблице.

Среднее время работы алгоритма, с, и ускорение

Объем входных данных, количество центров	GPU	CPU	Ускорение
1184, 100	0.4	2.6	6.5
1184, 300	4.4	21.1	4.8
15000, 100	5.2	52.9	10.2
15000, 200	19.3	202.6	10.5
15000, 500	119.7	1352.3	11.3
15000, 1500	1010.7	12213.4	12.1

Модификация K-MEANS++

Была реализована идея еще одного способа поиска центров, ориентированного на графические процессоры и базирующегося на K-MEANS++.

В целях снижения затрат на копирование данных было решено в каждом подмножестве B_n входных данных X, используя K-MEANS++, выбирать $k-1$ точек в качестве центров (первый центр для всех подмножеств общий – c_0). Затем из полученной совокупности снова выбирать $k-1$ центров тем же алгоритмом (рис. 2).

В настоящий момент проводится тестирование эффективности использования центров, выбранных подобным образом как начальных в алгоритме K-MEANS. Однако важно заметить, что получить ускорение работы программы, реализующей данный алгоритм, удастся при достаточно большом наборе входных данных (сотни мегабайт или гигабайты). В ином случае суммарное время выполнения всех операций копирования много меньше времени, необходимого для обработки хотя бы одного блока данных.

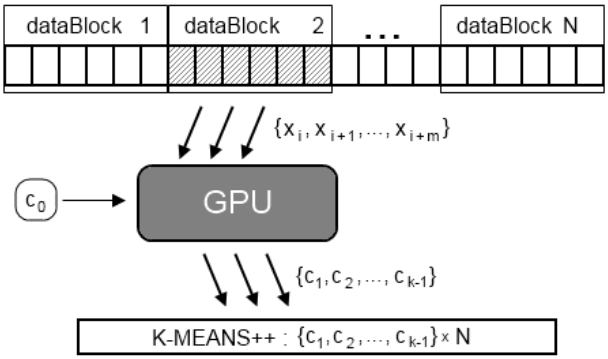

Рис. 2. Модифицированный алгоритм выбора центров

Заключение

Распараллеливание алгоритма K-MEANS++ проводилось с помощью технологии OpenCL, что делает его пригодным для запуска на различных вычислительных и графических устройствах. Ввиду полученного ускорения использование его на практике значительно сократит время выбора центров для K-MEANS, а значит, и время кластеризации в целом.

Литература

1. **Arthur D., Vassilvitskii S**. K-MEANS++: The Advantages of Careful Seeding [Электронный ресурс]. – URL: http://www.stanford.edu/~darthur/kMeansPlusPlus.pdf
2. **Arthur D.** Analyzing and Improving Local Search: K-MEANS and ICP [Электронный ресурс]. – URL: http://www.stanford.edu/~darthur/thesis.pdf
3. **Munshi A., Gaster B.** OpenCL Programming Guide. – USA: Addison-Wesley, 2011. – 603 p.
4. ATI Stream Computing OpenCL Programming Guide. – USA: Advanced Micro Devices, Inc, 2010. – 142 p.
5. **Berkhin P.** Survey of clustering data mining techniques. Techical report, Accrue Software, San Jose, CA, USA, 2002.
6. **Gaster B., Howes L.** Heterogeneous Computing with OpenCL. – USA: Elsevier Inc., 2010. – 288 p.