REPORTS OF NATIONAL ACADEMY OF SCIENCES
OF THE REPUBLIC OF KAZAKHSTAN
ISSN 2224-5227
Volume 5, Number 5(2014), 5 – 10

UDC 519.683; 519.684

PARALLEL ALGORITHMS FOR CLUSTERIZATION PROBLEMS WITH USING THE GPU

B. Baizhanov, N. Litvinenko
baizhanov@hotmail.com, n.litvinenko@inbox.ru

Institute of mathematics and mathematical modelling of the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan, Almaty, Kazakhstan

Key words: Parallel algorithm, GPU, cluster analysis, multithreading, single linkage method.

Abstract. This article describes the problem of cluster analysis using single linkage method for problems with up to 2 million records and up to 25 indexes. Due to large volumes of data for solving the problem we used computing power of graphic processors. Parallel algorithm was developed in C++ in Microsoft Visual Studio 2010 for complex use of CPU multithreading and parallel data processing with using the GPU. This development may have applications in various areas of science, for example, in genetics, biology, sociology and other.

УДК 519.683; 519.684

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ В СРЕДЕ С ГРАФИЧЕСКИМИ ПРОЦЕССОРАМИ ДЛЯ ЗАДАЧ КЛАСТЕРНОГО АНАЛИЗА

Б.С. Байжанов, Н.Г. Литвиненко
(Институт математики и математического моделирования КН МОН РК, Алматы, Казахстан)

Ключевые слова: Параллельные алгоритм, графические процессоры, кластерный анализ, мультипоточность, метод ближайшего соседа.

Аннотация. В данной статье описывается решение задачи кластеризации по методу ближайшего соседа (МБС) для задач с объемами данных до 2 млн. записей и количеством полей до 25. В связи с большими объемами данных для решения задачи используются вычислительные мощности графических процессоров. Параллельный алгоритм разработан на языке C++ в среде Microsoft Visual Studio 2010 под комплексное использование средств мультипоточности центрального процессора и возможностей параллельной обработки данных средствами графического процессора. Данная разработка может иметь приложение в различных отраслях науки, например, в генетике, биологии, социологии и других.

Работа выполнена при поддержке гранта 0741/ГФ МОН РК

Введение. Кластеризация – это разбиение объектов одной природы на несколько групп так, чтобы объекты в одной группе были в каком-то смысле схожими, а объекты из разных групп значительно отличались друг от друга. Под схожестью обычно понимается близость объектов друг к другу в соответствии с выбранной метрикой. Кластеризация может применяться в любой области науки, где необходимо исследование статистических или экспериментальных данных.

Метод ближайшего соседа является одним из наиболее часто используемых при решении задач кластерного анализа. Расстояние между двумя кластерами определяется как расстояние между ближайшими их представителями. На первом этапе работы алгоритма рассчитывается матрица расстояний между объектами. На каждой итерации в матрице расстояний ищется минимальное значение, соответствующее расстоянию между двумя наиболее близкими кластерами. Найденные кластеры объединяются, образуя новый кластер. Процедура повторяется до тех пор, пока не будут объединены все кластеры.

В данной работе рассматривается задача с большим количеством данных – до 2 миллионов записей и до 25 показателей. На каждом шаге ведется поиск двух кластеров, расстояние между которыми будет минимальным с последующим слиянием их в один. Вычисление расстояний между объектами определяется какой-либо из общепринятых метрик, в данном случае

$$\rho_{ab} = \sqrt{\sum_j \left(x_{j,a} - x_{j,b}\right)^2}$$

За расстояние между кластерами P и R возьмем

$$L_{PR} = \min_{\substack{a \in P \\ b \in R}} (\rho_{ab})$$

Вычисление расстояний между объектами, но в особенности между кластерами представляет собой достаточно дорогую по времени операцию. Приблизительный подсчет времени показал, что задача будет считаться очень долго. Поэтому при решении данной задачи были задействованы методы, позволяющие значительно сократить время счета.

Для решения задач по методам кластеризации по методу МБС в данной работе используются вычислительные мощности графических процессоров. Это сделано для того, чтобы иметь возможность решать задачи с объемами данных до двух миллионов записей и количеством показателей до 25. Однако даже значительное увеличение вычислительной мощности рабочей станции не позволяет решать такие задачи. Чтобы решить данные проблемы, мы отказываемся от иерархических идей построения алгоритмов.

Аналогичные разработки. Пакеты для статического анализа получили широкое распространение на современном рынке. Существующие статические пакеты можно поделить на универсальные (предлагают широкий спектр статистических методов для решения различных задач без деления на предметные области, например STATISTICA, Minilab и др.) и специализированные (число используемых статистических методов, как правило, ограничено конкретными предметными областями, например STADIA, statit и др.). Вычислительные мощности универсальных статических пакетов не всегда удовлетворяют нуждам проектов, а использование профессиональных пакетов, где допустимые объемы обрабатываемых данных, как правило, значительно выше, требует больших финансовых затрат. К тому же эти пакеты имеют различные ограничения в использовании, так как были разработаны под конкретные нужды, зачастую сложны в использовании, для них отсутствует специализированная литература. В виду объемности нашей задачи, и сложности использования каких либо готовых программных продуктов, целесообразнее решать задачу своими средствами.

Постановка задачи. Имеется достаточно большой набор данных, характеризующий некоторое множество объектов или процессов. Набор данных может содержать до двух миллионов записей, и до 25 полей, описывающих характеристики объекта. Требуется разработать алгоритм для задачи кластеризации по методу МБС. Иерархические подходы при составлении алгоритма здесь не приемлемы ввиду больших объемов данных. При построении алгоритма необходимо ориентироваться на архитектуру графических процессоров и разумно распределить задачи между

центральным и графическим процессором.

Итак, требуется разбить данное множество объектов на кластеры, содержащие схожие объекты. За расстояние между объектами a и b возьмём обычное Евклидово расстояние

$$\rho_{ab} = \sqrt{\sum_j \left(x_{j,a} - x_{j,b}\right)^2}$$

За расстояние между кластерами P и R возьмем

$$L_{PR} = \min_{\substack{a \in P \\ b \in R}} (\rho_{ab})$$

При построении кластеров может возникнуть необходимость учитывать некоторые из дополнительных условий:

- Ограничения на предельно допустимое количество объектов в кластере.
- Ограничение на минимальное количество кластеров.
- Ограничение на максимальный размер диаметра кластера.
- Ограничение на максимальное расстояние между кластерами. Если расстояние между двумя ближайшими кластерами на очередной итерации стало больше некоторого наперед заданного числа, работа по построению кластеров заканчивается.

В данной работе мы не рассматриваем вопросы, связанные с корректной подготовкой исходных данных. Это отдельная большая проблема, которая часто решается исходя из специфики исследуемой прикладной задачи.

Для решения объемных по вычислениям и данным частей желательно использовать мощные вычислительные средства графических процессоров.

Описание алгоритма. Ясно, что вычислительных мощностей центрального процессора недостаточно для решения данной, чрезвычайно объемной задачи. Необходимо привлекать вычислительные мощности графического процессора. Вначале определим, когда, с какими задачами должен работать центральный процессор, а какие задачи должны выполняться на графическом процессоре. Архитектура графического процессора, в основном, заточена под технологию SIMD, процессор имеет очень много вычислительных ядер, обладает достаточно большой глобальной памятью и очень быстрой, но небольшой, распределяемой памятью. Центральный процессор больше сориентирован под многозадачность, то есть, по сути, под технологию MIMD. Он намного гибче в использовании, но и значительно медленнее. Значит объемные по вычислениям места, достаточно хорошо ложащиеся под технологию SIMD необходимо выполнять на графическом процессоре, а различные управляющие, организующие функции поручить центральному процессору. Там же должны выполняться небольшие по объему вычислений задачи и задачи, плохо ложащиеся под технологию SIMD.

Перечислим задачи, которые целесообразно выполнять на графическом процессоре:

- Определение расстояний между всевозможными парами объектов.
- Определение расстояний между всевозможными парами кластеров.
- Различные виды сортировок.

Задачи, которые целесообразно выполнять на центральном процессоре:

- Построение заданий для различных потоков, выполняющихся на центральном процессоре.
- Построение различных заданий для графических процессоров.
- Обработка результатов вычислений, полученных на графических процессорах.
- Обработка результатов вычислений, полученных в различных потоках на центральном процессоре.
- Объединение кластеров.
- Вычисление различных вспомогательных величин.

Распределение потоков на центральном процессоре и на графическом процессоре принципиально разное. Переключение между потоками на центральном процессоре – дорогая операция, и если потоков много, время на переключение потоков может значительно превысить время реальных полезных вычислений. На центральном процессоре потоков не должно быть

много. Практика показывает, что потоков должно быть втрое больше числа физических ядер процессора. Для графических процессоров ситуация принципиально другая. Переключение между потоками на графическом процессоре, здесь их принято называть нитями – операция дешевая, и ограничений на количество потоков здесь нет. Количество ядер на графическом процессоре велико, и переключения будут достаточно редки.

Исходя из вышесказанного, и некоторых других соображений опишем алгоритм решения задачи.

Пусть **K1** – количество записей, **K2** – количество ядер процессора, **K3** – количество потоков, **K4** – количество полей в записи.

В общем случае алгоритм должен быть параллельным, а также должен использоваться графический процессор. Однако, если данных мало, наиболее эффективным будет однопоточный алгоритм. Будем считать, что если данных меньше 10 тысяч, программа должна работать в однопоточном режиме. Эффект от использования ресурсов графического процессора будет, когда данных достаточно много. Будем считать, что если данных меньше 50 тысяч, программа должна работать в мультипоточном режиме, но без использования ресурсов графического процессора. Если данных будет больше 50 тысяч, будут задействованы все ресурсы рабочей станции.

Опишем вариант параллельного алгоритма с использованием графического процессора.

1. Выбор режима работы. Если ***K1<10001,*** *работает однопоточный режим. Если* ***10000<K1<50001,*** *работает многопоточный режим без привлечения ресурсов графического процессора.* ***Если K1>50000,*** *работает мультипоточный режим, задействованы ресурсы графического процессора.*

Далее необходимо определить оптимальное количество потоков для центрального процессора. Практика показывает, что наиболее эффективно брать количество потоков втрое больше количества физических ядер процессора.

2. Определяем количество ядер центрального процессора на данной рабочей станции ***K2.*** *Вычисляем* ***K3=K2*3.***

Определяем оптимальные размеры блока и сетки для работы на графическом процессоре. Блок и сетку выбираем одномерными. В силу архитектурных особенностей графических процессоров, целесообразно, чтобы количество нитей в блоке было кратно 32, выберем 128. Количество блоков в сетке выберем из следующих соображений. Все данные будут переданы на обработку K3 потокам, т.е. будут поделены на K3 частей. Поскольку минимальное количество записей, при которых начинает работать графический процессор – 50 тысяч, а наиболее распространенные центральные процессоры имеют 4 физических ядра, что определяет 12 потоков, то на каждый поток будет приходиться в данном случае чуть более 4 тысяч записей. Мы определили, что в одном блоке будет 128 нитей, т.е. блок будет обрабатывать 128 записей(объектов). Таким образом в данном случае количество блоков в сетке будет равно

$$KOLBL1 = int\left(\frac{50000}{12 * 128}\right) + 1 = 33$$

Желательно, чтобы все нити в сетке были задействованы. Поэтому каждый поток, кроме последнего должен получить для обработки **33*128=4224** объекта, а последний поток остаток **50000 - 4224*11= 3536** объектов. Этот пример показывает, как будут распределяться данные по потокам и блокам.

3. Определяем количество блоков в сетке и количество объектов в потоке.

$$KOLBL1 = int\left(\frac{K1}{K3 * 128}\right) + 1$$

$$KOLPOT1 = KOLBL1 * 128$$

$$KOLPOT2 = K1 - KOLPOT1 * (K3 - 1)$$

4. Считываем данные и условия расчета в оперативную память **MAS1**.

5. Делим все данные на **K3** *порций (по количеству потоков) согласно пункту 3. Формируем задание для каждого потока. Это пара массивов (***MAS1, MASPJ***).* **J=1,2,3,…,K3**.

6. Подготавливаем **K3** *потоков для работы, передаем им задание.*

7. Каждый из **K3** *потоков формирует задание для графического процессора и инициализирует его работу.*

Далее у нас будет задействовано 2 цикла – внешний и внутренний. Задачи внешнего цикла:

- Синхронизировать работу потоков.
- Организовать корректную работу внутренних циклов на всех потоках.
- Организовать корректную работу графического процессора на всех потоках.
- Обработать результаты работы всех потоков на очередной итерации.

Задача внутреннего цикла для каждого потока следующая (8,9,10,11,12):

8. Каждый поток готовит задание для графического процессора и инициализирует его выполнение.

9. Графический процессор просматривает все пары объектов ***(a,b)****, где* $a \in MASPJ$, $b \in MAS1$, *причем объекты должны принадлежать разным кластерам* ***A*** *и* ***B****. Находит расстояние между центрами кластеров* ***A*** *и* ***B****. Если расстояние меньше некоторой величины, равной удвоенному предельному расстоянию между кластерами, рассчитанному на предыдущей итерации, то кластеры считаются подозрительными и вычисляется расстояние между кластерами* ***A*** *и* ***B****.*

10.Графический процессор анализирует расстояние между кластерами. Если расстояние между кластерами ***A*** *и* ***B*** *входит в 100 наименьших расстояний, определенных в данном задании на данной итерации в данном потоке, запоминаем эту пару кластеров.*

11.Просмотрев все возможные пары объектов ***(a,b)****, графический процессор в конце работы будет иметь 100 или меньше пар кластеров, расстояние между которыми наименьшее для данного потока. Возвращает это множество пар для дальнейшей обработки своему потоку.*

12.Получив результаты работы графического процессора, поток сортирует пары и передает это множество пар на обработку во внешний цикл. Ждет окончания работы текущей итерации всеми потоками.

Далее работает внешний цикл, который делает следующее (13,14,15,16,17):

13.Объединяем все выходные данные (пары) от каждого потока. Находим среди этих пар 100 пар кластеров с наименьшими расстояниями.

14.Сортируем эти 100 пар в порядке возрастания расстояний.

15.Запускаем процесс объединения выбранных пар кластеров.

16.Проверяем признаки окончания работы. Если процесс формирования кластеров закончен – выход, иначе выполняем пункт 18.

17.Делим откорректированные данные на **K3** *порций (по количеству потоков). Формируем задание для каждого потока. Это пара массивов (***MAS1, MASPJ***). J=1,2,3,…,K3.*

18.Переходим к новой итерации с откорректированными данными – пункт 6.

Среда разработки. Программное обеспечение разрабатывалось в среде:

- Программная среда - Microsoft Visual Studio 2010
- Язык программирования - C++
- Среда работы с GPU - CUDA 5.5

Для разработки данного программного обеспечения использовался системный блок, оснащенный:

- Материнская плата - Gigabyte Technology Co., Ltd., Z77MX-D3H, Chipset Intel – наиболее оптимальное решение в соотношении цена-качество;
- Центральный процессор - Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz с поддержкой Hyper-threading, с которым получается 8 виртуальных ядер;
- Графический процессор - NVIDIA GeForce GTX 660 с поддержкой самой современной архитектуры Kepler (CC 3.0);
- Оперативная память - 16384 Mb, Type: DDR3;
- Жесткий диск - 2 Тб.
- Операционная система - Microsoft Windows 7, Ultimate, 32 bit.

Выводы. Кластерный анализ находит применение во многих областях науки, таких как социология, генетика, информатика и т.д. Поэтому решение в данной работе задачи кластеризации

большого объема данных весьма актуально. Выбранный метод кластеризации – метод ближайшего соседа является широко востребованным методом для решения такого рода задач. В нашей работе мы отказались от иерархического способа кластеризации, что позволило нам использовать для решения столь объемной задачи обычную рабочую станцию, без привлечения дорогостоящего оборудования. Комплексное использование средств мультипоточности центрального процессора и привлечение вычислительных мощностей графического процессора значительно сокращает время расчетов. Это обеспечивает возможность рядовым исследователям решать объемные задачи за вполне допустимое время.

ЛИТЕРАТУРА

[1] Бериков В. С., Лбов Г. С. Современные тенденции в кластерном анализе // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы», 2008. — 26 с.
[2] NVIDIA CUDA C programming Guide, May 2013, www.nvidia.com
[3] Фленов М.Е. Библия C# (2-е издание, 2011), Санкт-Петербург, «БХВ-Петербург», 2011
[4] Эхтер Ш, Робертс Д. Многоядерное программирование. «Питер», 2010
[5] Боресков А.В., Харламов А.А., Основы работы с технологией CUDA, Москва, 2010

REFERENCES

[1] Berikov V. S., Lbov G. S. Sovremennye tendencii v klasternom analize // Vserossijskij konkursnyj otbor obzorno-analiticheskih statej po prioritetnomu napravleniju «Informacionno-telekommunikacionnye sistemy», 2008. — 26 s.
[2] NVIDIA CUDA C programming Guide, May 2013, www.nvidia.com
[3] Flenov M.E. Biblija S# (2-e izdanie, 2011), Sankt-Peterburg, «BHV-Peterburg», 2011
[4] Jehter Sh, Roberts D. Mnogojadernoe programmirovanie. «Piter», 2010
[5] Boreskov A.V., Harlamov A.A., Osnovy raboty s tehnologiej CUDA, Moskva, 2010.

КЛАСТЕРЛІК ТАЛДАУ ЕСЕПТЕРІ ҮШІН ГРАФИКАЛЫҚ ПРОЦЕСС ОРТАСЫНДАҒЫ ПАРАЛЛЕЛЬДІК АЛГОРИТМДЕР.

Тірек сөздер: Параллельді алгоритмдер, кластерлік талдау, көп ағындылық К-орташа әдісі, көп ядролы процессорлар.

Аннотация. К-орташа әдіс бойынша кластеризациялық көлемді есептерге, көп ядролы процессорлар үшін параллельді алгоритм дамып, өңделіп жатыр. Бұл алгоритм көп ағындылық әдістерді пайдаланыла отырып, Microsoft Visual Studio 2010 ортасында, C++ тілінде бағдарламалық код жүзінде жүзеге асырылды. Максималды рұқсат етілетін деректердің көлемі: 300 мың жазбаға дейін, 25-ке дейін көрсеткішер санымен.

Поступила 20.09.2014 г.