N E W S
OF THE NATIONAL ACADEMY OF SCIENCES OF THE REPUBLIC OF KAZAKHSTAN
PHYSICO-MATHEMATICAL SERIES
ISSN 1991-346X
Volume 5, Number 297 (2014), 49 – 53

APPLICATION OF A PARALLEL ALGORITHM FOR SOLVING THE CLUSTERING PROBLEM FOR BIG DATA BY COMPLETE LINKAGE CLUSTERING WITH USING THE GPU

V. Pospelova, G. Litvinenko

Institute of Mathematics and Mathematical Modelling, Almaty, Kazakhstan

Key words: data clustering, big data, complete-linkage clustering, GPU, CPU, SIMD, MIMD.

Abstract. This article discusses the solution to the problem of cluster analysis with data volumes up to 1 million records and the number of fields to 25 by the complete linkage method (CLM) using the computing power of graphics processors. CLM is one of the most difficult techniques for implementation in the program code. Large amounts of computation with this method require significant computational resources. In this paper is developed a parallel algorithm that focuses on the use of hybrid CPU and GPU.

This development may have a certain interest in solving problems with big data in in applied sciences.

УДК 519.683; 519.684

ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ КЛАСТЕРИЗАЦИИ ДЛЯ БОЛЬШИХ ОБЪЕМОВ ДАННЫХ ПО МЕТОДУ ПОЛНОЙ СВЯЗИ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

В. Поспелова, Г. Литвиненко

Институт математики и математического моделирования, Алматы, Казахстан

Ключевые слова: кластеризация данных, большие объемы данных, метод полной связи, графический процессор, центральный процессор, SIMD, MIMD.

Аннотация. В данной статье рассматривается решение задачи кластерного анализа с объемами данных до **1 млн** записей и количеством полей до **25** по методу полной связи (дальше **МПС**) с использованием вычислительных мощностей графических процессоров. **МПС** является одним из наиболее тяжелых методов для реализации в программном коде. Большие объемы вычислений по данному методу требуют привлечения значительных вычислительных ресурсов. В данной работе разрабатывается параллельный алгоритм, ориентированный на гибридное использование центрального и графического процессоров.

Данная разработка может иметь определенных интерес при решении задач с большими объемами данных в прикладных отраслях науки.

В последние десятилетия интеллектуальный анализ данных (Data Mining) получил активное развитие, как новое и перспективное направление в области обработки информации. Основной идеей и отличием Data Mining от классического анализа являются попытки смоделировать поведение человека при решении сложных интеллектуальных задач. Кластеризация данных является частной задачей интеллектуального анализа данных, основной целью которой является объединение в группы объектов, как можно более схожих между собой, и как можно более отличающихся от объектов в других группах [1]. Степень похожести для каждой пары объектов определяется

измерением некоторой меры сходства или «расстояния» между ними. Однако вычисление расстояний между объектами, а, тем более кластерами, представляет собой дорогостоящую по времени операцию. Поиск двух кластеров с минимальным расстоянием между ними осуществляется на каждом шаге, для каждой итерации. Для определения расстояния между кластерами по методу **МПС** требуется вычислить расстояния между всевозможными парами объектов из рассматриваемых кластеров и взять пару с наибольшим расстоянием. Ориентация разрабатываемого программного обеспечения на работу с большими объемами данных заставляют в первую очередь заняться разрешением проблемы временных затрат, так как по приблизительным расчетам для вычисления расстояния между всевозможными парами объектов, к примеру, по Евклидовой метрике, для решаемой задачи нам потребуется около $5 \cdot 10^{20}$ тактов. Это без учета временных затрат на сортировку данных, обозначение кластеров и т.д. Рассчитанное количество тактов означает годы непрерывной работы рабочей станции.

Проведенный обзор современного состояния рынка программных продуктов, предлагающих решение задач кластерного анализа (в основном это статистические пакеты: **STATISTICA, Minilab, STADIA, KNIME** и др.) показывает, что задачи кластерного анализа для больших объемов данных на сегодня являются еще неразрешенными. Дело в том, что в большинстве из рассмотренных пакетов реализованы только самые популярные из методов кластеризации, такие как k-средних, ближайшего соседа, метод Варда. Однако несмотря на свою популярность среди исследователей, связанную, в первую очередь, с высокой скоростью работы и простотой использования, ни один из данных методов нельзя назвать универсальным – на выходе результаты кластеризации одного и того же набора данных по разным методам будут в корне отличаться друг от друга. Поэтому актуальным является разработка алгоритмов кластеризации по различным методам, с дальнейшей их реализацией в программном коде, что предоставит исследователю возможность выбора наиболее оптимального метода для кластеризации конкретного набора данных. На сегодня статистические пакеты предлагают решение задач кластеризации используя иерархические методы, в среднем, для данных в 14 000 – 16 000 записей, в зависимости от характеристик рабочей станции. Этого определенно недостаточно.

Именно поэтому необходимо найти принципиально новые подходы к решению задач с большими объемами данных. Ясно, что в однопоточном режиме такие задачи не решаются. Как минимум нужен многопоточный режим на центральном процессоре с несколькими ядрами. Значит должен быть параллельный алгоритм. Но и мощностей хорошего центрального процессора так же будет мало. Поиск дополнительных вычислительных ресурсов заставляет нас обратиться к графическим процессорам. Архитектура графических процессоров сориентирована на одновременное выполнение большого количества потоков (нитей). Количество ядер в графических процессорах может быть более тысячи. Потенциал очень велик, но технология SIMD, под которую сориентированы графические процессоры, ограничивает возможности их использования. Мы приходим к выводу, что эффективно использовать мощные графические процессоры вместе с универсальным центральным процессором.

Нам известно, что решением данного вопроса уже занимаются наши российские и зарубежные коллеги, но сведения о характере и стадии их разработок в открытом доступе нам найти не удалось. В связи с вышесказанным актуальным является решение о разработке собственного программного обеспечения с более высокой производительностью, ориентированного на решение объемных задач.

Перейдем к постановке задачи. *Имеется достаточно большой набор данных (до миллиона записей и 25 полей), описывающих некоторое множество объектов или процессов. Необходимо провести разбиение данного множества объектов на кластеры.* Для вычисления расстояний между объектами ***a*** и ***b*** используем Евклидову метрику:

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

За расстояние между кластерами ***P*** и ***R*** возьмем

$$L_{PR} = \max_{\substack{a \in P \\ b \in R}} \left(\rho_{ab} \right).$$

Кластеризация по **МПС** является одной из самых ресурсоемких и времяемких операций. Поэтому наиболее затратные по времени места в алгоритме целесообразно ориентировать на обработку графическими процессорами. Такими местами при **МПС** является нахождение кластеров с минимальным расстоянием между ними и нахождение минимальных расстояний между объектами. Так же для повышения производительности алгоритма целесообразно ввести ряд следующих условий:

- *Первое условие построения кластеров:* построение кластеров прекращается, как только количество кластеров становится меньше заданного числа **N1.**
- *Второе условие построения кластеров:* работа по построению кластеров прекращается, если расстояние между двумя ближайшими кластерами становится больше заданного числа **N2**.
- *Первое условие объединения кластеров:* выбранный кластер не подлежит объединению с другими, если количество объектов в нем больше заданного числа **N3**.
- *Второе условие объединения кластеров:* кластер не подлежит объединению с другими кластерами, если его диаметр превышает значение заданного числа **N4.**

Ввиду больших временных затрат на вычисления при работе с большими объемами данных по стандартным иерархическим методам кластеризации, в данном алгоритме предлагается способ, когда на очередной итерации происходит объединение нескольких пар кластеров, а не одной. Пары выбираются таким образом, чтобы расстояние между кластерами в этих парах было наименьшим среди всевозможных рассматриваемых пар. Перед объединением кластеров из каждой пары в один кластер эти пары сортируются в порядке возрастания по расстоянию между кластерами. Таким образом, мы пытаемся сэкономить на количестве итераций. Вполне возможно, что некоторые из рассматриваемых пар окажутся в одном кластере раньше, чем мы соберемся их объединять. Просто пропустим такие пары кластеров.

При разработке алгоритма для данной метода (МПС) необходимо понять какие задачи должны выполняться на графическом процессоре, а какие на центральном. Графические процессоры – это в основном технология SIMD, что означает очень быстрые вычисления и очень большие неудобства для программиста. Центральный процессор – это технология MIMD. Здесь все наоборот – все очень удобно для программиста, но вычисления происходят значительно медленнее, чем на графическом процессоре. Поэтому будем стараться объемные, но простые задачи направлять на выполнение на графические процессоры, а сложные, но менее объемные задачи – на центральный процессор.

В начале, на первой итерации, мы считаем каждый объект отдельным кластером. Расстояние между кластерами будет равно расстоянию между объектами. На следующих итерациях уже некоторые объекты будут объединены в кластеры, а некоторые кластеры могут по-прежнему представлять из себя только один объект. У нас возникнут следующие основные задачи:

- Подготовка задания для потоков. Задача выполняется одним потоком.
- Расчет расстояний между объектами. Задача выполняется на графическом процессоре.
- Расчет расстояний между кластерами. Задача выполняется на графическом процессоре.
- Сортировка пар выбранных кластеров. Задача выполняется одним потоком.
- Объединение кластеров. Задача выполняется одним потоком.
- Подготовка задания для графического процессора. Задача выполняется каждым потоком центрального процессора.
- Проверка хода выполнения кластеризации объектов, окончания работы. Задача выполняется одним потоком.

Распределение задач было сделано из следующих соображений.

Так как принципиальным различием между центральным и графическим процессорами является стоимость операции переключения между потоками – в случае с первым она неоправданно высока, и большое количество потоков может сильно затормозить по времени остальные необходимые вычисления. Поэтому количество потоков на центральном процессоре не должно быть минимальным – по некоторым данным, оно не должно превышать утроенного значения числа физических ядер процессора. С потоками на графическом процессоре наоборот не возникает никаких проблем, так как из-за большого количества ядер переключения потоков случаются редко.

Введем следующие обозначения:

- K_1 – количество записей;

- K_2 – количество ядер процессора;
- K_3 – количество потоков;

Далее мы будем рассматривать только случай, когда количество данных, нуждающихся в обработке, требует задействовать все ресурсы графического процессора.

Подготовительный этап вычислений состоит из трех шагов:

- определение оптимального количества потоков для центрального процессора:

$$K_3 = K_2 \cdot 3 = 12.$$

Как уже отмечалось выше, по некоторым практическим данным, оптимальное количество потоков на центральном процессоре должно составлять утроенное количеств ядер процессора. В данном случае учитывая, что наиболее распространенные модели процессоров имеют по 4 физических ядра, получаем, что количество потоков не должно превышать 12.

- определение оптимальных размеров блока и сетки на графическом процессоре:

$$K_{BL1} = \text{int}\left(\frac{50\ 000}{K_3 \cdot 128}\right) + 1 = 33,$$

где K_{BL1} – количество блоков, определяемое как целое значение от отношения минимального количества записей, при которых подключение вычислительных мощностей графического процессора становится целесообразным, к K_3 потокам и количеству нитей в одном блоке, плюс единица.

- определение количества блоков в сетке и количества объектов в потоке.

$$K_{BL1} = \text{int}\left(\frac{K_1}{K_3 \cdot 128}\right) + 1,$$

$$K_{POT1} = K_{BL1} \cdot 128,$$

$$K_{POT2} = K_1 - K_{POT1} \cdot (K_3 - 1).$$

Мы исходим из того, что физически нити одновременно выполняются по **WARP'ам** (32 нити). У нас нет возможности заставить одновременно выполнять нити одного **WARP'а** разные команды. Поэтому количество нитей, передаваемых на обработку графическому процессору, должно быть кратно 32. Чтобы мультипроцессор графического процессора работал эффективно, блок должен содержать несколько **WARP'ов.** Мы взяли 4 **WARP'а,** то есть получили 128 нитей. Конечно, остаток распределяемых данных будет совершенно произвольным, но это будет потеря только в одном месте.

После вычислений всех перечисленных характеристик направляем эту информацию в оперативную память **MAS1**. Все имеющиеся данные необходимо разделить на K_3 потоков и сформировать для каждого потока задание. После получения своего задания, каждый из K_3 потоков должен сформировать задание для графического процессора.

За синхронизацию работы потоков, организацию корректной работы на всех потоках и обработку результатов у нас будет ответственен цикл, условно названный внешним. Так же в его задачу будет входить контролирование работы внутренних циклов. Задачи внутреннего цикла можно расписать следующим образом:

*1. Определение расстояния между центрами кластеров **A** и **B**.* Для определения данного расстояния все пары объектов, принадлежащих разным кластерам, просматриваются графическим процессором. Если найденная величина меньше удвоенного предельного расстояния между выбранными кластерами, найденного на предыдущей итерации, то вычисляется расстояние между кластерами **A** и **B**.

2. Анализ расстояния графическим процессором. Пара кластеров **A** и **B** запоминается, если найденное между ними расстояние является одним из 100 наименьших расстояний, вычисленных в данном потоке на данной итерации.

3. Просмотр расстояний между всевозможными парами объектов. Результатом данного шага является около 100 пар кластеров с наименьшим для данного потока расстоянием. Данные пары передаются своему потоку для дальнейших вычислений.

4. Сортировка результатов графическим процессором. Отсортированные пары будут переданы внешнему циклу. Все потоки ожидают.

Дальнейшая обработка данных совершается внешним циклом.

5. Обработка полученных данных со всех потоков. Выбираем из всех полученных пар со всех потоков 100 пар с наименьшим расстоянием.

6. Сортировка выбранных пар в порядке возрастания расстояний.

7. Объединение выбранных пар кластеров.

8. Определения стадии процесса формирования кластеров. Если процесс закончен – выход. Иначе переходим к новой итерации с откорректированными данными – с пункта 1, предварительно разделив новые данные на K_3 потоков и сформировав задание для каждого потока.

Для разработки программного обеспечения использовался системный блок следующей комплектации: Gigabyte Technology Co., Ltd., Z77MX-D3H с чипсетом Intel; Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz; NVIDIA GeForce GTX 660; оперативная память 16384 Mb; жесткий диск емкостью 2 Тб. Разработки проводились на операционной системе Microsoft Windows 7, Ultimate, 32 bit, в среде Microsoft Visual Studio 2010 на языке C# с использование CUDA 5.5.

Заключение. Таким образом, для достаточно актуального метода **МПС** использование вычислительных средств центрального и графического процессоров позволит решать достаточно объемные как по вычислениям, так и по объемам данных задачи. Разработка алгоритмов и их программная реализация для различных методов кластеризации (к-средних, МПС, цендроидный метод и др.) даст разработчику прикладных задач инструментарий для исследования своей проблемы.

Работа выполнена при поддержке гранта 0741/ГФ МОН РК.

ЛИТЕРАТУРА

[1] Бериков В.Б., Лбов Г.С. Современные тенденции в кластерном анализе. – Новосибирск: Изд-во Ин-та математики им. С. Л. Соболева, 2008. – 26 с.
[2] Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – 2010. – С. 232.
[3] NVIDIA CUDA C programming Guide. – May 2013, www.nvidia.com
[4] Jain A., Murty M., Flynn P. Data Clustering: A Review // ACM Computing Surveys. – 1999. – Vol. 31, N 3.
[5] Дюран Б., Оделл П. Кластерный анализ / Пер. с англ. Е. З. Демиденко, под ред. и с предисл. А. Я. Боярского. – М.: Статистика, 1977. – 128 с.
[6] Шилдт Г. C# 4.0 полное руководство. – СПб.: Печатный двор, 2011.
[7] Лупин С.А., Посыпкин М.А. Технологии параллельного программирования. – М.: ИД «ФОРУМ»: ИНФРА-М, 2011. – 208 с.

REFERENCES

[1] Berikov V.B., Lbov G.S. Modern trends in cluster analysis. Novosibirsk: Publishing House of the Institute of Mathematics, 2008. P. 26.
[2] Boreskov A.V., Kharlamov A.A., The basics of working with CUDA technology. 2010. P. 232.
[3] NVIDIA CUDA C programming Guide. May 2013, www.nvidia.com
[4] Jain A., Murty M., Flynn P. Data Clustering: A Review. ACM Computing Surveys. 1999. Vol. 31, N 3.
[5] Durand B., Odell P. Cluster analysis. M.: Statistics, 1977. P. 128.
[6] Shildt G. C # 4.0 complete guide. SPb.: Printing house, 2011.
[7] Lupin S.A., Posypkin M.A. Parallel programming technologies. M: Publishing house "FORUM": INFRA-M, 2011. P. 208.

КӨП АҒЫНДЫЛЫҚ ТӘРТІБІНДЕГІ(РЕЖИМІНДЕГІ) КӨЛЕМДІ МАҒЛҰМАТТАРДЫ ТОЛЫҚ БАЙЛАНЫС ӘДІСІ БОЙЫНША КЛАСТЕРИЗАЦИАЛАУ.

В. Поспелова, Г. Литвиненко

Математика және математикалық үлгілеу институты, Алматы, Қазақстан

Тірек сөздер: мағлұматтарды кластеризациялау, үлкен көлемді мағлұматтар, толық байланыс әдіс, графикалық процессор, орталық процессор, SIMD, MIMD.

Аннотация. Берілген мақалада толық байланыс әдіс(ТБӘ) арқылы және есептеуіш графикалық процессорларды қолдана отырып көлемі 1 млн дейін жазбасы және 25-ке дейін жиек көрсеткішері бар есептердің шығару жолдары қарастырылады. ТБӘ бағдарламалық кодта жүзеге асырылуы ең ауыр әдіс болып табылады. Бұл әдіс үлекен көлемді есептеулер үшін айтарлықтай есептеу қорын талап етеді. Бұл жұмыста, орталық және графикалық процессорларды гибридті қолдануларына бейімделген, параллельді алгоритм жетілдірінеді. Айтылмыш зерттеме үлкен көлемді мағлұматтары бар есептерді шығару үшін, ғылымның қосымша тармақтарында белгілі қызығушылықтарды танытуы мүмкін.

Поступила 01.10.2014 г.