ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКОГО УСКОРИТЕЛЯ ДЛЯ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ ДАННЫХ

С. В. Жабинский
Разработчик в ООО "Информационно-технологический альянс", магистр технических наук

ООО «Информационно-технологический альянс», Республика Беларусь
E-mail: sergey.zhabinskii@gmail.com

Abstract. This paper describes how general-purpose computing on graphics processing units (GPGPU) can be used in tasks of intelligent data processing using Data Mining algorithms. Results of practical experiments are given to analyze performance of the approach.

Появление технологий интеллектуального анализа данных (или Data Mining) связано, в первую очередь, с необходимостью анализа тех объёмов информации, которые накапливаются в современных хранилищах данных. Возможность использования алгоритмов математической статистики и машинного обучения для данных задач открыли новые возможности для аналитиков и исследователей.

В последние годы сильное развитие получили технологии параллельной обработки информации на графических процессорах [1, 2]. Данные процессоры имеют SIMD архитектуру и изначально ориентированы на обработку компьютерной графики, большая часть операций в которой представляет собой выполнение одинаковой операции одновременно над разными данными.

Архитектура компьютера и схема передачи данных при вычислении на GPU представлена на рисунке 1.

Время выполнения операции в системе можно разбить на две основные группы:

1 время, необходимое для обработки данных на CPU или GPU;

2 время, необходимое для передачи данных между компонентами системы.

Перед началом любых вычислений данные должны быть загружены из HDD в оперативную память компьютера (DRAM). При вычислении на CPU, данные передаются непосредственно из основной памяти в CPU. При использовании GPU, обмен данными происходит по следующей схеме: DRAM – CPU – память GPU – вычислительные элементы GPU.

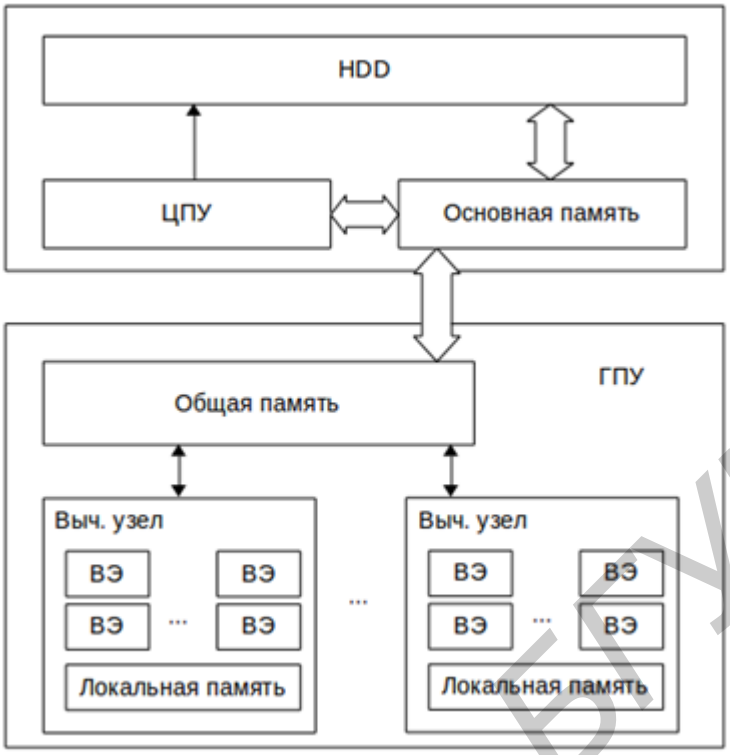

Рис. 1. Архитектура компьютера с GPU и обмен данными между его компонентами

Возникает вопрос о том, насколько существенно влияние времени обмена данными на общее время вычислений. Несмотря на то, что, как правило, при использовании GPU можно добиться более высокой скорости вычислений, главным недостатком является задержка при загрузке данных в память графического ускорителя. Поэтому необходим анализ каждой отдельной задачи. Необходимо сравнивать время на загрузку данных в память GPU и время, которое удалось сэкономить за счёт использования параллельных вычислений.

Рассмотрим основные технологии, использующиеся при GPGPU вычислениях.

В настоящий момент существует несколько аппаратно-программных архитектур для универсальных вычислений на графическом ускорителе. Основные из них: Compute Unified Device Architecture (CUDA) [3], Open Computing Language (OpenCL) [4], Microsoft DirectCompute. Данные платформы позволяют разработчику абстрагироваться от использования графических примитивов и оперировать привычными понятиями и типами данных.

Рассмотрим достоинства и недостатки каждой из данных платформ. CUDA, являющаяся разработкой компании NVIDIA, обладает высоким быстродействием и хорошей поддержкой, однако ограничивает возможности выбора аппаратной части видеокартами компании NVIDIA. Microsoft DirectCompute, в свою очередь, требует наличия DirectX 10 или 11. В отличие от конкурентов, OpenCL даёт больше свободы в выборе аппаратной платформы.

Данную технологию поддерживают такие компании, как NVIDIA, IBM, Intel, AMD. Написанное с использование OpenCL ПО может запускаться как на GPU, так и на CPU. Это даёт преимущество во время разработки: алгоритм может быть реализован и протестирован практически на любом оборудовании на небольшом объёме данных, а впоследствии запущен без изменений на вычислительном кластере.

Особенностью алгоритмов Data Mining является их высокая вычислительная сложность. При этом, многие алгоритмы могут быть эффективно реализованы с использованием параллельных вычислений. Однако, как было сказано выше, время на загрузку данных в память GPU может негативно сказаться на времени выполнения задачи в целом.

Для тестирования были выбраны следующие задачи: задача кластеризации данных методом k-means (k-средних), и задача обучения однослойного персептрона.

Метод k-means является одним из первых и одним из самых популярных методов кластеризации [5]. Исторически, данный метод предлагался различными авторами в разных формах. Более подробно они рассмотрены в [6].

Для реализации алгоритма k-means был выбран классический алгоритм Ллойда. Для набора наблюдений $[x_1, x_2, x_3, ,x_n] \in R^d$, где R^d – d-размерное пространство, алгоритм пытается найти k центров кластеров $C = [c_1, c_2, , c_k] \in R^d$ которые являются решением задачи минимизации (1).

$$E=\sum_{i=1}^{k}\sum_{j=1}^{n} d\left(c_i,x_{ij}\right) \tag{1}$$

Алгоритм однослойного персептрона относится к алгоритмам с обучением с учителем и позволяет производить линейную классификацию. Принцип работы однослойного персептрона основан на модели функционирования нервной клетки – искусственного нейрона. Однослойный персептрон является нейронной сетью с одним скрытым слоем нейронов. Схема однослойного персептрона представлена на рисунке 2.

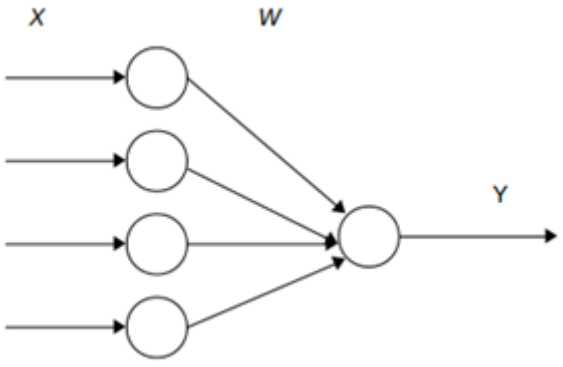

Рис. 2. Схема однослойного персептрона

Для экспериментов использовались следующие конфигурации оборудования:

1 Управляющий процессор – Intel Core i7 2.2 ГГц, сопроцессор – GPU

2 Один узел вычислительного кластера БГУИР [7]: управляющий процессор – 2 x CPU Intel Xeon E5-2650, сопроцессор – 2 x Tesla M2075 6 Gb RAM.

3 В качестве управляющего процессора и сопроцессора – Intel Core i7 2.2 ГГц. В данном случае код для управляющей программы и ускорителя выполняется на одном и том же процессоре. Благодаря OpenCL используется вся доступная вычислительная мощность. Также стоит отметить, что общее время выполнения не включает перенос данных из основной памяти в память ускорителя.

Тестовые данные представляют собой сгенерированные наборы наблюдений различного объёма с разным количеством кластеров. То есть рассматривалось изменение времени выполнения задачи в зависимости от объёма вычислений.

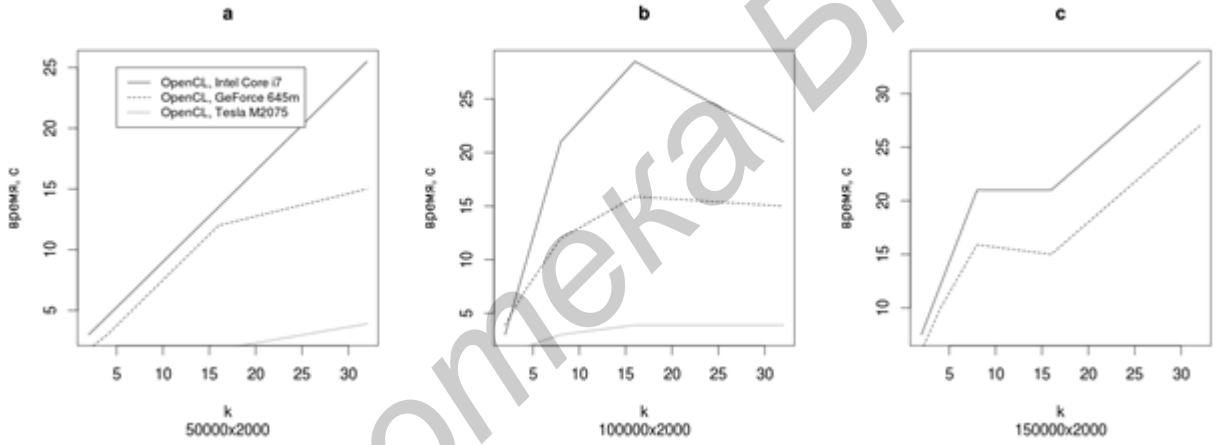

Рис. 3. Зависимость времени кластеризации от количества кластеров для разных объемов данных для предложенной реализации на OpenCL. Количество признаков: 2 тысячи. a) 50 тысяч наблюдений, b) 100 тысяч, c) 150 тысяч

Использовались наборы данных из 50, 100, 150, 300 (только для конфигурации 2) тысяч наблюдений с размером пространства признаков от 1 до 5 тысяч признаков. Кластеризация производилась на 2, 4, 8, 16 и 32 кластера.

На рисунке 3 изображены графики зависимости времени кластеризации от количества кластеров. На графике «c» результаты для Tesla M2075 вплотную прилегают к оси и не видны из-за масштаба графика.

Как видно, при увеличении количества кластеров, время кластеризации ожидаемо возрастает из-за увеличения вычислительной сложности. Однако можно заметить, что при использовании параллельного ускорителя наклон графика значительно более пологий. Изгиб графика в районе k = 16 обусловлен уменьшением количества итераций, необходимых для того, чтобы алгоритм сошёлся.

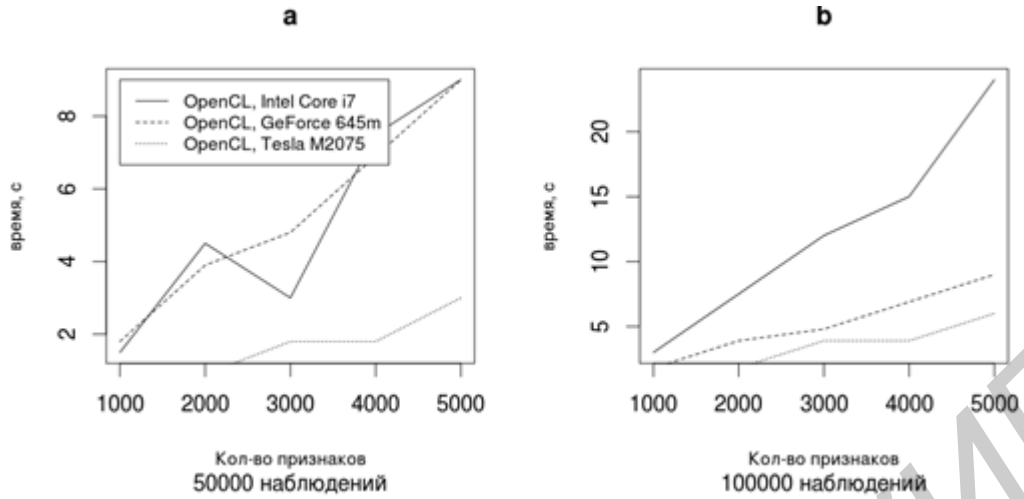

Рис. 4. Зависимость времени кластеризации от изменения размера пространства признаков для предложенной реализации на OpenCL

На рисунке 4 изображена зависимость времени кластеризации от изменения количества признаков. Можно наблюдать такую же зависимость, как и на предыдущих графиках.

Для проверки эффективности параллельной реализации алгоритма обучения однослойного персептрона для платформы OpenCL были использованы платформы 1 (CPU + GPU) и 3 (CPU). Графики с временем обучения для данных реализаций представлены на рисунке 5.

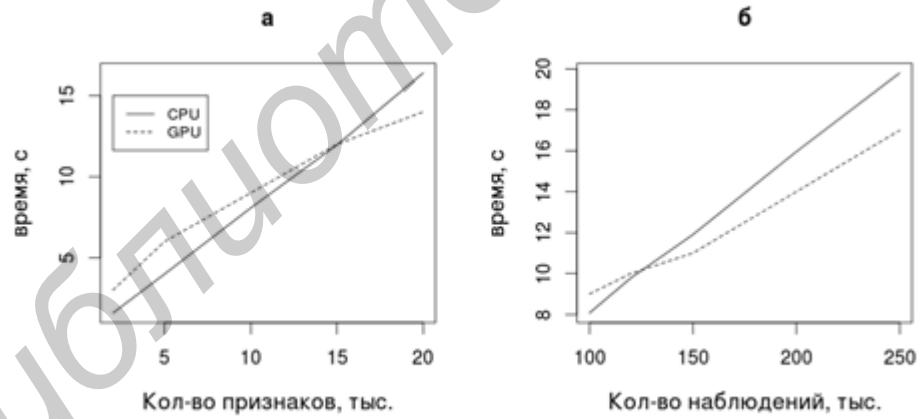

Рис. 5. Время обучения однослойного персептрона

На скорость обучения однослойного персептрона влияют размер обучающей выборки и размер пространства признаков. Тестирование производилось с различными значениями данных параметров для определения масштабируемости реализации, то есть изменение времени обучения от объёма вычислений.

Как видно из графиков, с ростом объёма вычислений, время обучения растёт

практически линейно. Однако, при использовании параллельного алгоритма наклон прямой более пологий. Также следует обратить внимание на точку пересечения графиков. Данная точка показывает пороговое значение объёма данных, при котором эффективность использования реализации для OpenCL становится выше классической реализации для выполнения на CPU.

Таким образом, результаты экспериментов подтверждают эффективность использования для задач Data Mining в качестве ускорителя параллельного графического процессора.

Литература

[1]. Jian, L. Parallel data mining techniques on Graphics Processing Unit with Compute Unified Device Architecture (CUDA) / L. Jian, C. Wang, Y. Liu, S. Liang, W. Yi, Y. Shi // The Journal of Supercomputing. - 2013. - Vol. 64, iss.3 — p. 942-967.

[2]. Keckler, S. GPUs and the Future of Parallel Computing. / S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, D. Glasco // IEEE Micro. - 2011. - Vol. 31, № 5 - p. 7-17.

[3]. Nickolls, J. Scalable Parallel Programming with CUDA / J. Nickolls, I. Buck, M. Garland, K. Skadron // ACM Queue. - 2008 - Vol. 6, № 2 - p. 40-53

[4]. Stone, J. E. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems / J. E. Stone, D. Gohara, G. Shi // Computing in Science & Engineering. - 2010. - Vol. 12, iss. 3. - p. 66-73.

[5]. Bock, H. H. Clustering Methods: A History of k-means Algorithms / H. H. Bock // Selected Contributions in Data Analysis and Classification / H. H. Bock; ed.: P. Brito[et al.]. - 2007. - p. 161 - 172.

[6]. Morissette, L. The k-means clustering technique: General considerations and implementation in Mathematica / L. Morissette, S. Chartier // Tutorials in Quantitative Methods for Psychology. - 2013. - Vol. 9, № 1. - p. 15-24.

[7]. Вычислительный кластер БГУИР [Электронный ресурс]. - Режим доступа: http://www.bsuir.by/ru/kaf-evm/vychislitelnyy-klaster-bguir. - Дата доступа: 11.04.2016. ://www.bsuir.by/ru/kaf-evm/vychislitelnyy-klaster-bguir. - Дата доступа: 11.03.2016.