УДК 004.032.26

ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В АЛГОРИТМАХ ОБУЧЕНИЯ И РАБОТЫ ИСКУССТВЕННОЙ НЕЙРОННОЙ СЕТИ

Ширма А.А.[1], Чулюков В.А.[1]

[1]*ФГБОУ ВПО «Воронежский Государственный педагогический университет», Воронеж, Россия (394043, Воронеж, ул. Ленина, 86), e-mail: chul_130451@mail.ru*

Рассмотрено использование параллельных вычислений на многоядерных центральном (CPU) и графическом (GPU) процессорах для повышения быстродействия работы искусственной нейронной сети (ИНС) в системе фильтрации шума. В некоторой степени многопроцессорные системы, как и ИНС, копируют структуру мозга. Предварительные результаты показывают 1,5-кратное повышение производительности при обработке изображений и 4-кратное при обучении ИНС. Таким образом, использование параллельных реализаций алгоритмов для обучения и работы ИНС способно уменьшить количество времени, необходимое для обработки данных. Для расчетов использованы центральный процессор Intel Core i5-2400 с 4 ядрами и графический адаптер NVIDIA GeForce GTX 460 с 336 ядрами. В качестве ИНС использован многослойный перцептрон. Тестирование проводилось в MATLAB с установленным Parallel Computing Toolbox.

Ключевые слова: нейронная сеть, изображение, параллельные вычисления

USE OF PARALLEL ALGORITHMS FOR LEARNING AND WORK ARTIFICIAL NEURAL NETWORK

Shirma A.A.[1], Chulyukov V.A.[1]

[1]*Voronezh State Pedagogical University, Voronezh, Russia (394043, Voronezh, Lenin Str., 86), e-mail: chul_130451@mail.ru*

The usage of parallel computing on multi-core central (CPU) and graphics (GPU) processors to improve performance of an artificial neural network (ANN) in a noise filtering. To some extent, multiprocessor systems copy brain structure. Preliminary results showed 1.5 times performance improvement in image processing and 4-fold at training the ANN. Thus, the use of parallel implementations of algorithms for learning and working ANN can reduce the amount of time required to process the data. For the calculations used CPU Intel Core i5-2400 with 4 cores and graphics card NVIDIA GeForce GTX 460 with 336 cores. As ANN used a multilayer perceptron. Testing was conducted in MATLAB to set Parallel Computing Toolbox.

Keywords: neural network, image, parallel computing

Введение

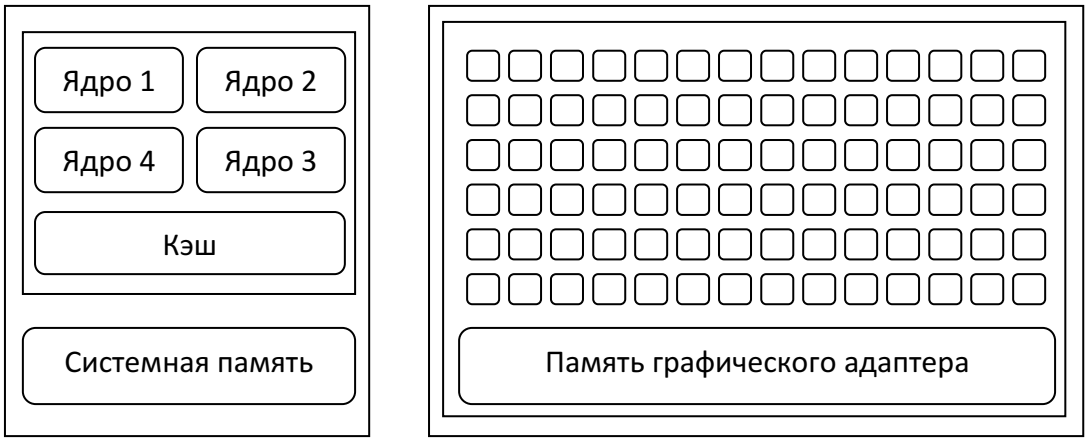
В последнее время разработчики центральных процессоров (CPU) для увеличения производительности и решения проблем с тепловыделением пошли по пути добавления вычислительных ядер вместо увеличения тактовой частоты самого процессора [6]. Каждое ядро может обрабатывать независимый поток инструкций, что позволяет операционной системе и отдельным программам выполнять несколько задач одновременно на нескольких «медленных» ядрах, вместо того чтобы последовательно выполнять несколько задач на одном быстром ядре. Большинство коммерчески доступных процессоров в настоящее время включают от 2 до 8 вычислительных ядер.

Напротив, другим многоядерным устройством в современном компьютере является процессор графического адаптера. За последние 20 лет они эволюционировали от простых 2D графических процессоров до многоцелевых, программируемых, с высоким уровнем параллелизма, многоядерных с необычайной вычислительной мощностью и очень высокой

пропускной способностью памяти. Теперь они все чаще применяются и для научных расчётов [7].

Стандартный процессор должен обрабатывать много и в большинстве случаев очень разных задач одновременно. Поэтому он должен быть оптимизирован для этой цели – в основном для кэширования важных данных, используемых запущенными процессами, и быстро переключать контекст между ними.

В отличие от центрального процессора, который включает в себя достаточно ограниченное количество вычислительных ядер и оптимизирован для кэширования данных процессов и быстрого переключения между ними, в графическом процессоре их количество исчисляется сотнями (рисунок 1). Графические процессоры в первую очередь ориентированы на распараллеливание манипуляций с данными и только потом на кэширование и управление потоком инструкций [6].



а б

Рисунок 1 – сравнение числа ядер CPU (а) и GPU(б)

На рисунке 2 показан опубликованный компанией NVIDIA график роста производительности CPU и GPU за последние несколько лет в GFLOPS **(**Giga *FLoating-point Operations Per Second*) в секунду [6].

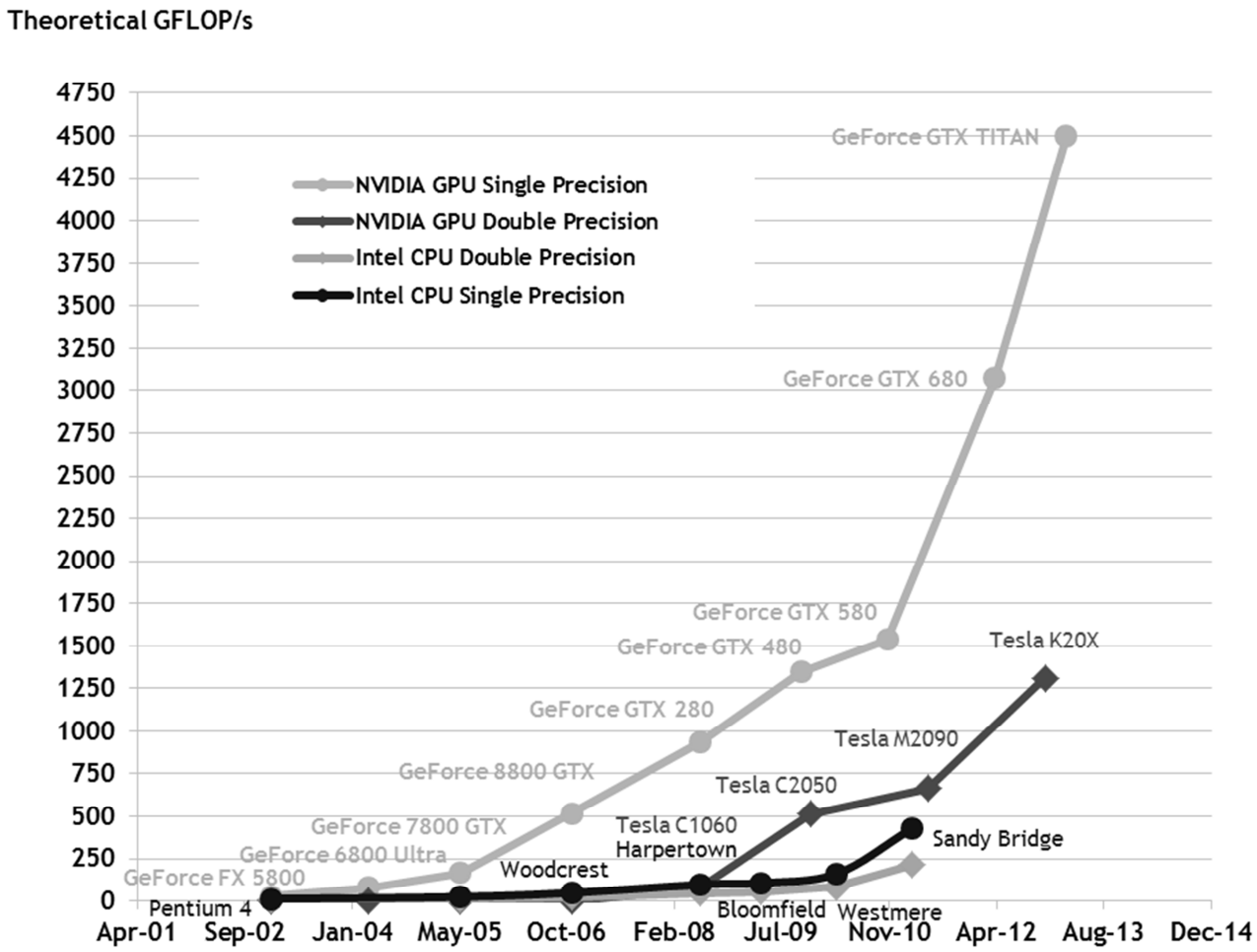


Рисунок 2 – количество операций с плавающей точкой наCPU и GPU

Для распараллеливания вычислительных потоков производитель графических процессоров NVIDIA представил программный интерфейс под названием CUDA (Compute Unified Device Architecture) [7]. Благодаря этой технологии, появилась возможность доступа к набору инструкций графического адаптера и управлению его памятью.

Вычислительная архитектура параллельной обработки GPU основана на понятии потокового мультипроцессора и концепции SIMD (Single Instruction and Multiple Data— одна команда и много данных**)**, позволяющей обеспечить параллелизм на уровне данных. Потоковый мультипроцессор (Streaming Multiprocessor) – это многоядерный SIMD процессор, позволяющий в каждый определенный момент времени выполнять на всех ядрах только одну инструкцию. Графические процессоры NVIDIA имеют ряд мультипроцессоров, каждый из которых имеет группу, параллельно работающих, универсальных шейдерных процессоров SP (Shader Processor). Потоковый мультипроцессор обрабатывает данные группой из 32 потоков. Такую группу именуют *warp*.

Цели и методы

Цель данного исследования заключается в повышении производительности алгоритмов обучения и тестирования ИНС путем использования параллельных реализаций алгоритмов

для GPU и измерении прироста производительности. Эти результаты сравниваются с результатами работы алгоритмов на многоядерном CPU.

Механизм общего вычисления с использованием GPU заключается в следующем. Входные данные передаются в GPU как текстуры или значения вершин. Затем в течение ряда проходов визуализации вершинные и пиксельные шейдеры выполняют вычисления, соответственно, координат, цвета для каждой из вершин текстуры и цвета для всех пикселей области, ограниченной этими вершинами.

В случае с использованием ИНС для обработки изображений основная проблема заключается в сложности вычислений на стадии обучения. На нее приходится большая часть времени обработки.

Так как ключевые операции работы с ИНС, такие как вычисление состояния нейрона, его активация, можно свести к матричным, требуется определить представление матрицы весов W, векторов стимулов X и смещений B в формате, доступном для GPU, то есть в виде текстур. Так как GPU оптимизирован для работы с пикселями, цвет которых описывается тремя каналами цвета RGB и одним каналом прозрачности A, то в работе [5] предлагается упаковывать 4 соседних либо по строке, либо по столбцу элемента матрицы в один пиксель. На рисунке 3 схематично изображена текстура размером 8x4 пикселя матрицы весов W размером 13x8 элементов.

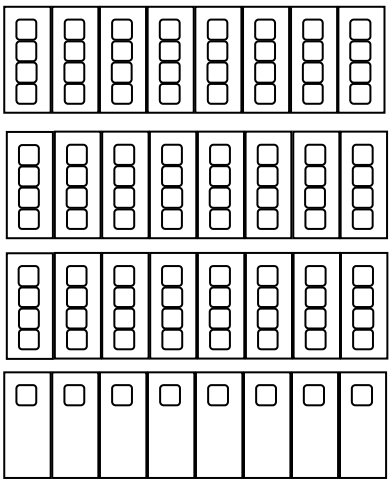


Рисунок 3 – пример текстуры матрицы W

Для вычисления выходных значений Y необходимо произвести вычисление активированных состояний нейронов по следующей формуле:

$$Y = sigmoid(W * X + B)$$

С учетом этого пиксельные шейдеры выполняют вычисление арифметических операций между строкой из W и столбцом из X по указанным координатам текстуры. Количество проходов визуализации, необходимых для вычисления полного результата, зависит от возможностей графического процессора.

При наличии более одного слоя в ИНС вышеописанная процедура повторяется для каждого слоя. В результате значения на выходе предыдущего слоя сохраняется в виде текстуры, которая затем используется в качестве входных данных для следующего слоя.

Результаты

Для тестирования был выбран двухслойный персептрон с топологией, показанной на рисунке 4 [1].

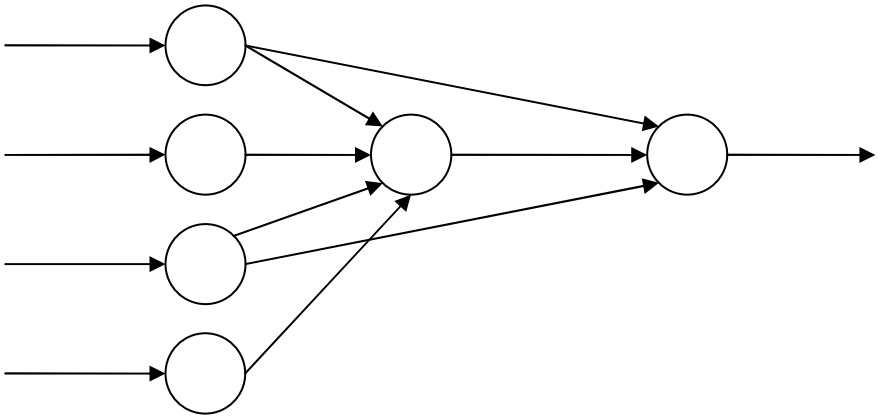


Рисунок 4 – топология двухслойного персептрона

Тестирование производилось на компьютере с процессором Intel Core i5-2400 с таковой частотой 3,1 ГГц и 4 ядрами [3], 4 Гб оперативной памяти и графическим адаптером NVIDIA GeForce GTX 460. Графический адаптер имеет 336 процессорных ядер с тактовой частотой 1350МГц, организованных в 7 мультипроцессоров [2]. По таблице вычислительных возможностей [8] этот адаптер имеет индекс 2.1.

В качестве программной основы использовался MATLAB с Parallel Computing Toolbox.

Обучение ИНС проводилось по методу сопряженных градиентов обратного распространения ошибки Моллера [4]. Временные результаты процесса обучения представлены в таблице 1.

Таблица 1.

Время обучения ИНС на различных процессорах

Процессор	Количество эпох	Время, с.
CPU, 1 ядро	50	34.811
CPU, 4 ядра	50	9.192
GPU	50	25.546

Для тестирования работы ИНС было выбрано изображение размером 255x255 пикселей, показанное на рисунке 5.

Рисунок 5 – изображение для обработки

В таблице 2 показаны временные результаты процесса обработки зашумленного изображения адаптивным нейросетевым фильтром.

Таблица 2.

Время обработки зашумленного изображения на различных процессорах

Дисперсия шума	АНФ (CPUsingle), сек.	АНФ (CPUmulti), сек.	АНФ (GPU), сек.
0.06	0.140	0.823	0.100
0.12	0.143	0.672	0.101
0.17	0.141	0.638	0.100
0.23	0.143	0.619	0.100
0.28	0.141	0.657	0.100
0.34	0.141	0.684	0.101
0.39	0.142	0.628	0.100
0.45	0.141	0.718	0.100
0.50	0.142	0.644	0.102

Выводы

Таким образом, результаты этого исследования показывают, что параллельная обработка в настоящее время способна улучшить производительность ИНС при обучении как минимум в 4 раза и при обработке – в 1,5 раза.

Список литературы

1. Ширма А.А, Чулюков В.А. Нейросетевые модели адаптивной фильтрации полутоновых изображений // Вестник Воронежского института МВД России. – 2010. - №2. – С.185-191.
2. GeForce GTX 460 [Электронный ресурс]. – Режим доступа: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-460 (Дата обращения: 18.07.2013).
3. Intel® Core™ i5-2400 Processor [Электронный ресурс]. – Режим доступа: http://ark.intel.com/ru/products/52207 (Дата обращения: 18.07.2013).
4. Moller M.F. A scaled conjugate algorithm for fast supervised learning // Neural Networks. – 1993. – Vol. 6. – P. 525-533.
5. Moravanszky A., Dense matrix algebra on the GPU // Wordware Publishing [Электронный ресурс]. – 2003. Режим доступа: http://www.shaderx2.com/shaderx.PDF (Дата обращения: 18.07.2013).
6. NVIDIA CUDA Programming Guide [Электронный ресурс]. – Режим доступа: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (Дата обращения: 18.07.2013).
7. NVIDIA CUDA [Электронный ресурс]. – Режим доступа: http://www.nvidia.com/object/cuda_home_new.html (Дата обращения: 18.01.2013).
8. NVIDIA CUDA GPUs [Электронный ресурс]. – Режим доступа: https://developer.nvidia.com/cuda-gpus (Дата обращения: 18.07.2013).
9. Wikipedia. GPU history. – Режим доступа: https://en.wikipedia.org/wiki/Graphics_processing_unit (Дата обращения: 18.07.2013).

Рецензенты:

Сумин В.И., д.т.н., профессор, профессор кафедры управления и информационно-технического обеспечения Федерального казенного образовательного учреждения высшего профессионального образования «Воронежский институт Федеральной службы исполнения наказаний», г. Воронеж.

Астахова И.Ф., д.т.н., профессор, профессор кафедры математического обеспечения ЭВМ Федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Воронежский государственный университет», г.Воронеж.