УДК 004.4+519.683+612.13

# ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ГЕНЕТИЧЕСКОГО АЛГОРИТМА, КОНСТРУИРУЮЩЕГО ЦИФРОВЫЕ ФИЛЬТРЫ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ CUDA

***Белобродский В.А.,*** *аспирант факультета компьютерных наук Воронежского государственного университета, e-mail: belobrodsky@yandex.ru*

**Ключевые слова:** параллельные вычисления, технология CUDA, математическое моделирование, алгоритмы обработки данных, биомедицинские сигналы.

*Разработана компьютерная программа, реализующая генетический алгоритм для конструирования цифровых фильтров с применением технологии CUDA. Проведены серии вычислительных экспериментов для экспериментального определения производительности созданной программы, которая является функцией нескольких аргументов. Приведены экспериментальные результаты замера производительности двух версий программ, которые показывает преимущество версии, использующей технологию CUDA, до 5-6 раз по сравнению с программой, не использующей данную технологию. Под производительностью в настоящей статье понимается количество популяций цифровых фильтров, генерируемых программой за 1 час работы.*

## Введение

Несмотря на впечатляющий рост вычислительных мощностей персональных компьютеров, однопоточные программы могут требовать значительного времени, особенно если требуется обработать большой объем входных данных, который свойственен многим зарегистрированным биомедицинским сигналам. Очевидно, что использование такого количества времени на ожидание результата работы неприемлемо, и этот факт делает актуальной проблему поиска наиболее «узких» и ресурсоемких мест алгоритма, с проведением дальнейшей их оптимизации с целью уменьшения времени вычисления. Эффективным способом уменьшения времени выполнения генетического алгоритма является его распараллеливание. Для осуществления этой цели очевидным кажется применение параллельного кластера для проведения вычислений, однако данный подход требует значительных организационных мероприятий и определенной технической подготовки и подчас серьезных материальных вложений.

Начиная с 2006 года появилась возможность ускорить время выполнения не графических вычислений, используя вычислительные мощности видеокарт, за счет предложенной компанией Nvidia технологии CUDA (Cumpute Unidied Device Architecture) [1, 3, 4, 7-9]. В данной работе приводится анализ производительности двух версий программ, реализующих генетический алгоритм, одна из которых использует технологию CUDA.

## Метод создания цифровых фильтров на основе генетического алгоритма

Рассмотрим теоретические положения, которые положены в основу предлагаемого метода конструирования цифровых фильтров для анализа биомедицинских сигналов с использованием идеологии генетических алгоритмов [2]. Пусть имеется поколение функций фильтров $Q^n{}_m(t)$ – его импульсных характеристик, где верхним индексом $n$ обозначен номер поколения, нижним индексом $m$ – номер фильтра внутри поколения, $t$ – дискретное время. При этом фильтр $Q^n{}_m(t)$ имеет нулевое среднее и единичную норму. Выполним операцию свёртки сигнала $E^k{}_x(t)$ (с известными свойствами) с фильтром $Q^n{}_m(t)$:

$$S^n_m(\tau, x, k) = \int Q^n_m(t)\ E^k_x(t-\tau)dt\,. \qquad (1)$$

Здесь $S^n{}_m(\tau, x, k)$ – коэффициенты свёртки сигнала $E^k{}_x(t)$ с фильтром $Q^n{}_m(t)$; $x$ – порядковый номер сигнала в $k$-ой группе.

Для простоты в дальнейшем будем рассматривать две группы $k$ сигналов ($k = k_1, k_2$). Свернем все сигналы каждой из этих двух групп $k_1$, $k_2$ со всеми фильтрами $Q^n{}_m(t)$. Усреднив максимальные модули разницы коэффициентов сверток среди всевозможных пар сигналов, принадлежащих разным группам, можно рассчитать величину $\Phi^n{}_m$, которую будем называть «эффективностью» или «параметром приспособленности» фильтра $Q^n{}_m$ (в литературе также встречается термин «фитнесс» [5]):

$$\Phi^n_m = \frac{\sum_{i=1}^{x^{\max}}\sum_{j=1}^{x^{\max}} MAX\left|(S^n_m(\tau, x^i, k_1) - S^n_m(\tau, x^j, k_2)\right|}{(x^{\max})^2}\,. \qquad (2)$$

Здесь операция *MAX* означает, что из всего одномерного массива, составленного из модулей разницы сверток двух сигналов ( $\left|(S^n_m(\tau, x^i, k_1) - S^n_m(\tau, x^j, k_2)\right|$ ), принадлежащих разным группам, берется только максимальное значение. В дальнейшем будет находиться среднее значение максимального модуля разницы среди всех пар, общее число которых $(x^{\max})^2$, так как здесь используется две группы сигналов по $x^{\max}$ сигналов в каждой. Таким образом, согласно формуле (2), каждому фильтру текущего поколения $Q^n{}_m$ ставится в соответствие определенное число $\Phi^n{}_m$. Очевидно, что фильтры, позволяющие выявить наибольшие различия между исследуемыми сигналами, – это те, которые имеют макси-

мальные значения $\Phi^n_m$. Они и должны быть допущены к «скрещиванию». «Скрещивание» происходит во временной области по формуле

$$Q_m^{n+1}(t) = (Q_{m_1}^n(t) + Q_{m_2}^n(t))/2 \ . \quad (3)$$

С помощью «скрещивания» создается часть следующего поколения фильтров $Q^{n+1}$. В наших экспериментах – это 50% от всей «популяции» фильтров, а остальные 50% «популяции» заполняются лучшими фильтрами из предыдущего поколения без каких-либо их изменений. Другими словами, ($n$+1)-ое поколение фильтров на 50% состоит из фильтров предыдущего поколения, демонстрирующих наилучшие значения величины $\Phi^n$, и на 50% – из их потомков. Таким образом, «удачный» фильтр будет сохраняться во всём ряду «популяции», что позволяет избежать ситуации, когда фильтр, дающий высокие значения $\Phi^n$ в процессе «скрещивания», потеряет свои свойства. После формирования вышеприведенным образом нового поколения, запускается процедура, определяющая необходимость мутации с помощью логической функции

$$\mathrm{F}(Q^n) = \begin{cases} 1, \eta > 0,5 \\ 0, \eta \le 0,5 \end{cases}, \quad (4)$$

где $\eta$ – доля отсчетов дискретных фильтров, в которых коэффициент вариации $V_i$, определяемый ниже по формуле (5), меньше определенного значения (в наших экспериментах в качестве этого значения мы взяли 0,2).

Коэффициент вариации определяется как

$$V_i = \frac{\sigma(Q^n(i))}{\overline{Q^n(i)}}, \quad (5)$$

где $\sigma(Q^n(i))$ – среднеквадратическое отклонение $i$-го отсчета всех фильтров $n$-го поколения; $\overline{Q^n(i)}$ – среднее значение $i$-го отсчета всех фильтров $n$-го поколения.

В случае, если логическая функция (4) указывает на необходимость «мутации» (возвращает значение «истина»), то у каждого фильтра, созданного в текущем поколении, а не перенесенного из предыдущего, есть определенная вероятность претерпеть одну из следующих «мутаций»: «кроссинговер»; «кроссинговер» и случайный сдвиг фазы; добавление фликкер-шума; добавление фликкер-шума и случайный сдвиг фазы; добавление сине-зеленого шума; добавление сине-зеленого шума и случайный сдвиг фазы; случайный сдвиг фазы. В дальнейшем работа генетического алгоритма представляет собой итерационный процесс, который продолжается до тех пор, пока не сформируется заданное число поколений или не выполнится критерий остановки. Одним из возможных критериев остановки работы алгоритма может быть следующее условие:

$$\left| Max(\Phi^n) - Max(\Phi^{n+1}) \right| < \varepsilon \ , \quad (6)$$

где $\varepsilon$ – порог остановки алгоритма, задаваемый пользователем; $Max(\Phi^n)$ – максимальное значение параметра приспособленности в поколении $n$.

### О выборе критерия для сравнения производительности

Одной из важнейших характеристик вычислительного устройства является его быстродействие, которое для математических расчетов обычно измеряется в количестве операций с плавающей точкой в секунду (flops) [3].

Несмотря на то, что в современных языках программирования, например C#, вводится упрощенный подход для создания асинхронных программ [6], как правило, в обыденной практике, программа не сможет продолжать свои вычисления без завершения других процедур (нитей), так как для их продолжения нужны входные данные, которые еще рассчитываются. Поэтому конечному пользователю важен интегральный показатель быстродействия ресурсоемких программ, как правило, время, измеряемое в секундах, необходимое до завершения вычислительного процесса какой-либо задачи. Программе может понадобиться априори неизвестное количество времени на завершение работы, особенно ввиду того, что неизвестно, сколько поколений потребуется сгенерировать, чтобы сработало условие выхода из итерационного процесса генерации. А каждая конкретная итерация будет длиться в зависимости от исходных данных, а именно от длины сигналов $N$, длины фильтров $M$, количестве сигналов и фильтров ($K$ и $L$ соответственно). Стоит отметить, что количество фильтров в каждом поколении сохраняется постоянным на протяжении всей работы программы. В связи с этим, в настоящей статье приводится сравнительная характеристика и показывается преимущество применения технологии CUDA в единицах производительности программного обеспечения, реализующего генетический алгоритм $W$, под которым будем понимать количество генерируемых «популяций» за 1 час работы. Очевидно, что количество генерируемых за 1 час работы поколений (производительность $W$) является функцией от нескольких переменных – таких, как длина сигналов $N$, длина фильтров $M$, количество сигналов $K$ и количество фильтров $L$.

### Результаты исследований

Разработанная методика конструирования значений цифровых фильтров, опирающаяся на методологию генетических алгоритмов была реализована на языке программирования Visual C# без использования каких-либо механизмов распараллеливания. Первые вычислительные эксперименты, проведенные с помощью разработанной программы, привели к удовлетворительным результатам, но потребовали значительное время ожидания ($\approx$ 1 суток).

Как было упомянуто выше, для увеличения значений функции производительности $W(N, M, K, L)$ была создана вторая версия программного обеспечения, реализующая данный генетический алгоритм и использующая технологию CUDA для вычислений математической свертки, которая в указанном во введении примере запускалась 100*100*50=500000 раз в каждом поколении, так как на вход программы подавались две группы по 100 сигнлов в каждой, при заданном числе фильтров в каждом поколении, равном 50.

Для проведения сравнительного анализа производительности созданных программ было произведено 4 серии вычислительных экспериментов, в каждом из которых варьировался один из 4 параметров ($N, M, K, L$), от которых зависит функция производительности $W(N, M,$

*K, L)*, а остальные 3 параметра принимались за константу. Значение функции производительности, выраженное в количестве поколений, создаваемых за 1 час работы программы, определялось эмпирическим путем по формуле $W(N, M, K, L)=t^{-1}$, где $t$ – время, за которое программа генерирует одно поколение фильтров.

Это время определялось средствами ОС Windows по времени создания выходных файлов, в которых приводится детальная информация каждого из поколений сразу после их создания. На рис. 1-4 приведены зависимости функции $W(N, M, K, L)$ от значений ее параметров.

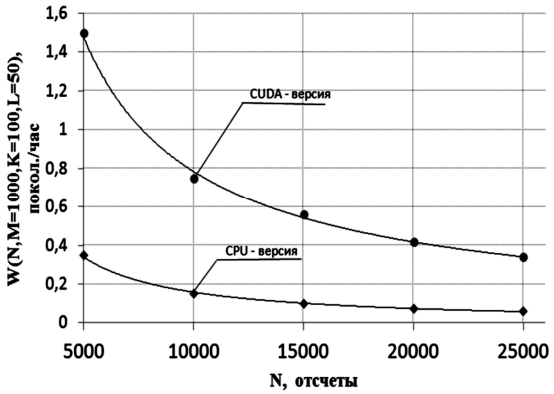


*Рис. 1. Зависимость функции производительности* $W(N,\ M=1000,\ K=100,\ L=50)$ *от длины сигналов* $N$

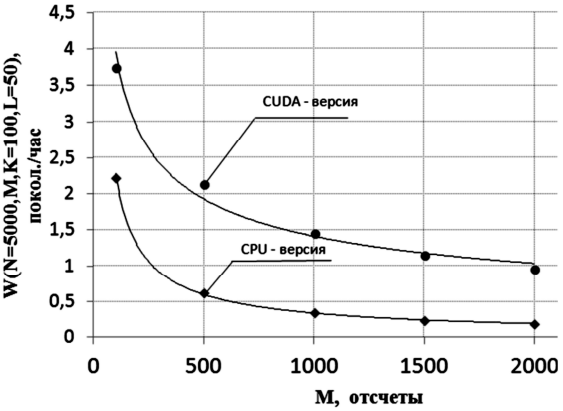


*Рис. 2. Зависимость функции производительности* $W(N=10000,\ M,\ K=100,\ L=50)$ *от длины фильтров* $M$

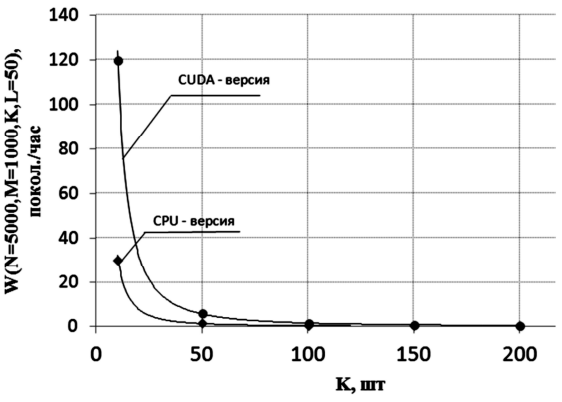


*Рис. 3. Зависимость функции производительности* $W(N=10000,\ M=1000,\ K,\ L=50)$ *от количества сигналов в каждой группе* $K$

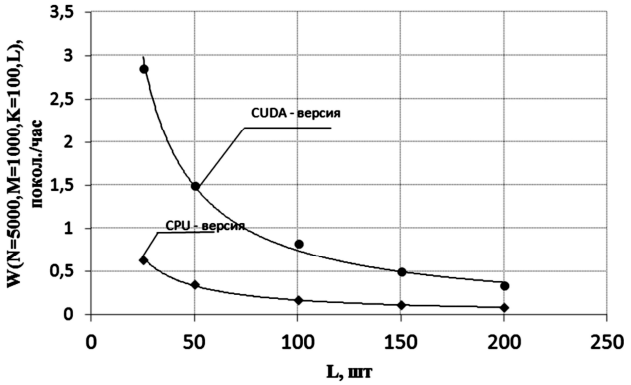


*Рис. 4. Зависимость функции производительности* $W(N=10000,\ M=1000,\ K=100,\ L)$ *от количества фильтров в каждой «популяции»* $L$

Во всех приведенных примерах аппроксимирующие кривые (сплошные черные линии) описывались степенными функциями типа $f=Ax^b$, где показатель степени $b$ был близок к минус единице и принимал значения от -0,989 до -1,098

Стоит отметить, что приведенные результаты имеют некоторую неучтенную систематическую погрешность, которая связана с тем, что в данной реализации генетического алгоритма, проводятся дополнительные преобразования типов, необходимые для запуска процедуры вычисления сверток, такие, как преобразование из типа коллекции List<float> в массив float[] и обратно после вычислений и т.п. Очевидно, что эти неопределенности прямо пропорциональны длине сигнала и количеству сигналов, и увеличивают время выполнения программы использующей CUDA, но несмотря на это, из рис. 1-4 видно её преимущество до 5 раз по величине производительности. Так же рис. 1-4 подтвердили, что наиболее ресурсоемкое место программы было определено верно, так как достигается 5-кратное увеличение производительность CUDA версии.

**Заключение**

Анализ производительности двух реализаций генетического алгоритма, предназначенного для генерации значений цифровых фильтров, наглядно продемонстрировал практическое преимущество внедрения технологии CUDA.

Стоит отметить, что полученное увеличение производительности работы алгоритма соответствует применению технологии CUDA только в одном, самом ресурсоемком месте кода – осуществлении процедуры математической свертки, которая при определенных значениях входных параметров (а именно, 200 – общее число файлов, 200 – число фильтров в каждой популяции) выполняется до $2*10^6$ раз в каждой «популяции».

С другой стороны, по своей сути генетический алгоритм легко может быть подвержен дальнейшему распараллеливанию, например, из-за независимости параметров оптимизации одних фильтров от других можно производить независимые вычисления этих показателей друг от друга. Таким образом, в перспективе остается потенциал для дальнейшего увеличения производительности за счет переноса других расчетных модулей

генетического алгоритма с CPU на GPU вычисления, таких, как «скрещивание», генерация «мутаций» и т.п.

Автор выражает особую благодарность Кургалину С.Д., Туровскому Я.А., Вахтину А.А.

**Литература**

1. Вахтин А.А., Туровский Я.А. Реализация численного вейвлетного преобразования на графических адаптерах архитектуры NVIDIA CUDA // Вестник Воронежского Государственного У ниверситета. Серия: Системный анализ и информационные технологии. – 2012. – №1. – С. 69-72.

2. Туровский А.Я. Создание фильтров для анализа ЭЭГ-состояний на основе генетических алгоритмов // Программная инженерия. – 2014. – №6. – С. 23-28.

3. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. М.: ДМК Пресс, 2010. –232 с.: ил.

4. Электронный ресурс: http://www.nvidia.ru/object/cuda-parallel-computing-ru.html

5. Holland J.H. Adaptation in Natural and Artificial System. – MIT Press, 1992. – 205 с.

6. Snell M., Powers L. Microsoft Visual Studios C#/.

7. Казённов А.М. Основы технологии CUDA.

8. Берилло А. NVIDIA CUDA – неграфические вычисления на графических процессорах (http://www.ixbt.com/video3/cuda-1.shtml)

9. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: введение в программирование графических процессоров. – М.: ДМК Пресс, 2011. – 232 с.

## IMPROVING PERFORMANCE OF GENETIC ALGORITHM THAT CONSTRUCTS DIGITAL FILTERS BY USING OF CUDA TECHNOLOGY

***Belobrodskiy V.A.***

This article describes a computer program that implements a genetic algorithm for designing digital filters using the technology of CUDA. A series of computational experiments to determine the performance of the software, which is a function of several arguments, is performed.

The article shows the experimental results of measuring the performance of two version of software, which show that the version which uses the CUDA technology is 5-6 times more productive. By performance the author understands the number of digital filters populations generated by the program an hour.