Ускорение работы алгоритма для решения задачи оптимизации при разработке системы управления мобильными роботами[1]

Захаров Д. Н.[1], Шмалько Е. Ю.[2]

Захаров Дмитрий Никанорович кандидат технических наук,
научный сотрудник;
Шмалько Елизавета Юрьевна кандидат технических наук, научный сотрудник,
Вычислительный центр им. А. А. Дородницына,
Российская академия наук,
Федеральный исследовательский центр «Информатика и управление», г. Москва

Аннотация: *в статье рассмотрены аспекты применения технологии CUDA для ускорения эволюционного алгоритма, решающего задачу оптимизации скалярной функции.*
Abstract: *the article discusses aspects of CUDA technology to accelerate the evolutionary algorithm to solve the optimization problem of a scalar function.*

Ключевые слова: *эволюционные алгоритмы, CUDA, параллельные вычисления.*
Keywords: *evolutionary algorithms, CUDA, parallel computing.*

В настоящее время для решения задач оптимизации применяется несколько методов, в том числе и эволюционные алгоритмы. В данной статье раскрывается применение технологии NVIDIA CUDA для ускорения работы генетического алгоритма.

При разработке системы управления мобильными роботами [1, с. 1] потребовалось решить задачу оптимизации скалярной функции. Цель функционирования системы управления заключается в максимально оперативной обработке поступающей от роботов информации и отправки им новых данных. Применение технологии CUDA и направлено на решение проблемы ускорения вычислений системой управления.

Постановка задачи

Имеется некоторая функция $F(X)$, где X - вектор, удовлетворяющий ограничениям для каждого компонента B.

$$X_{min}(B) \le X(B) \le X_{max}(B)$$

Изменяя вектор X, необходимо максимизировать значение M функции $F(X)$. В нашем случае рассматривается функция:

$$X_1^2 - X_1 * X_2 + X_3 = M. \quad (1)$$

Ограничения заданы в диапазонах:

$$\begin{matrix} 0 \le X_1 \le 5 \\ 0 \le X_2 \le 5 \\ -2 \le X_3 \le 2 \end{matrix} . \quad (2)$$

Решение задачи

Решение поставленной задачи с помощью генетического алгоритма потребовало разработки программы на языке C++. Предпочтение отдано данному языку также по причине того, что NVIDIA CUDA SDK ориентировано на языки C/C++, и отсутствует необходимость в дополнительной разработке модулей для работы на других языках.

Разработанная программа разделена на логические блоки, каждый блок представляет собой операцию генетического алгоритма. Такая структура выбрана для простоты и удобства последующей модернизации алгоритма.

В нашем случае с помощью технологии CUDA создано несколько ядер, реализующих вычисление функции (1) для всей популяции и часть функции селекции, связанных с обработкой всей популяции в цикле. Ввиду особенностей генетического алгоритма такие операции могут выполняться параллельно.

Непосредственно перенос выполнения вышеперечисленных функций на GPU потребовал разработки дополнительных функций для подготовки областей памяти, доступной на CPU и GPU. Технология CUDA не подразумевает использование общей с центральным процессором памяти. Именно этим и обусловлена их разработка.

Вычисления на GPU предполагают возможность использования совместимой пользовательской видеокарты. Проблема состоит в том, что технические характеристики этих видеокарт разные от поколения к поколению и внутри линейки любого модельного года. Для запуска CUDA-функции необходимо указать, сколько нитей в блоке и сколько блоков в сетке будет использовано. При этом необходимо учесть индивидуальные характеристики видеокарты. Параметры выбраны под

[1] Работа выполнена при поддержке РФФИ МК-6277.2015.8.

конфигурацию видеокарты компьютера, на котором разрабатывалась программа. Параметры равны 1024 нити и переменное количество блоков, в зависимости от размера популяции.

Данный выбор позволил обеспечить более равномерную загрузку GPU, а, следовательно, достичь более высоких показателей производительности за счет оптимальной загрузки потоковых процессоров GPU.

Учитывая вышеупомянутые особенности – при решении задачи с помощью технологии CUDA преимущество перед алгоритмом для CPU может быть получено только после разработки алгоритма, позволяющего учитывать особенности работы с GPU.

На Рис. 1 приведена блок схема CPU версии генетического алгоритма.

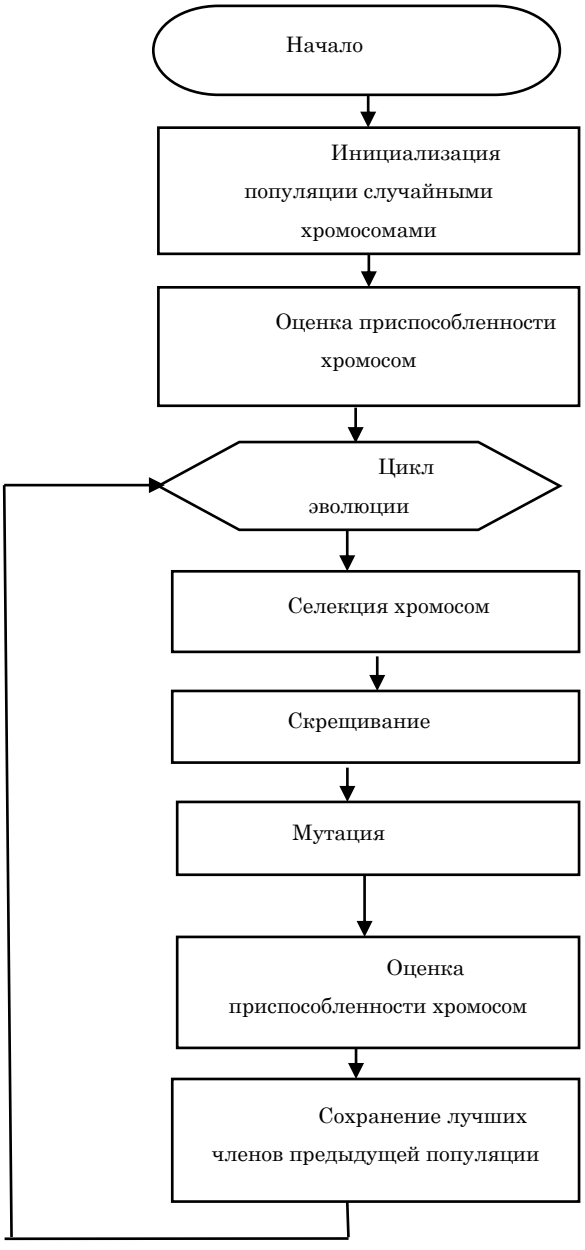

Рис. 1. Блок схема алгоритма для CPU

Для вызова CUDA ядра, вычисляющего значение функции живучести для каждой хромосомы в популяции, необходимо написание функций-оберток, т. к. напрямую вызывать такие функции из кода для CPU нельзя.

Код CUDA ядра для вычисления функции живучести:

```
__global__
void evaluate_cuda(genotype *population) {
```

```
//определение индекса нити
int idx = blockIdx.x * blockDim.x + threadIdx.x;
//вычисление живучести
population[idx].fitness = (population[idx].gene[0] * population[idx].gene[0]) – (population[idx].gene[0] * population[idx].gene[1]) + population[idx].gene[2];
}
```

Результаты проведенных экспериментов показали, что скорость работы алгоритма с применением CUDA оправдана только на популяциях большого размера. Преимущество в скорости начинает проявляться при размере популяции более 2000 хромосом.

Для того чтобы обработать популяцию на GPU, необходим ряд дополнительных действий:

процедуры объявления указателей на области памяти CPU и GPU,

непосредственно выделение памяти на GPU,

копирование данных из CPU в GPU,

вызов CUDA ядра,

копирование результатов обратно в память CPU.

Данные действия занимают дополнительное процессорное время, которое в случае работы CPU алгоритма расходуется на выполнение других процедур.

Выполнено несколько прогонов разработанной программы с различными вариантами размера популяции.

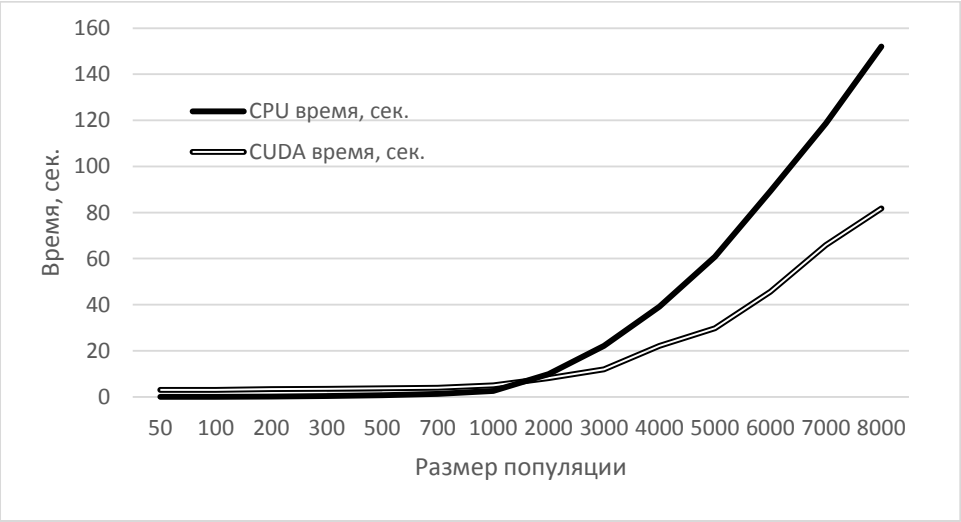

Рис. 2. Время выполнения программы, в зависимости от размера популяции

Результат

Преимущество, полученное от применения технологии CUDA, начинает проявляться после размера популяции в 2000 элементов. На популяциях малого размера алгоритм, выполняющийся на CPU, работает быстрее, т. к. он содержит ряд вспомогательных процедур, направленных на подготовку и пересылку данных по шине PCI-E от процессора к видеокарте.

Выводы

Применение технологии CUDA к решению задач с использованием генетических алгоритмов оправдано, но с учетом размера данных, обрабатываемых алгоритмом. Подобные алгоритмы, с частичной параллелизацией вычислений, успешно применяются [2, с. 2] [3, с. 1] для решения широкого спектра задач.

Литература

1. *Дивеев А. И., Шмалько Е. Ю.* Синтез системы управления мобильным роботом методом интеллектуальной эволюции // НиКСС . 2013. №3. [Электронный ресурс]. Режим доступа: URL: http://cyberleninka.ru/article/n/sintez-sistemy-upravleniya-mobilnym-robotom-metodom-intellektualnoy-evolyutsii (дата обращения: 27.09.2015).
2. *Пугин К. В., Ефимов С. С.* Генетические алгоритмы с частичной параллелизацией в системах с общей памятью на примере задачи коммивояжёра // МСиМ. 2012. № 2 (26). [Электронный ресурс]. Режим доступа: URL: http://cyberleninka.ru/article/n/geneticheskie-algoritmy-s-chastichnoy-parallelizatsiey-v-sistemah-s-obschey-pamyatyu-na-primere-zadachi-kommivoyazhyora (дата обращения: 27.09.2015).

3. *Pospíchal Petr* GPU-Based Acceleration of the Genetic Algorithm. In: Proceedings of the 16th Conference Student EEICT 2010 Volume 5. Brno: Faculty of Information Technology BUT, 2010, pp. 234-238. ISBN 978-80-214-4080-7.