УДК 004.056

А.А. Замула[1], Д.А. Семченко[2]

[1]*Харьковский национальный университет имени В.Н. Каразина, Харьков*
[2]*Харьковский национальный университет радиоэлектроники, Харьков*

МЕТОДЫ ПОСТРОЕНИЯ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА ОСНОВЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

Рассмотрены базовые понятия, используемые при программировании параллельных систем. Описан метод генерации Блюма-Блюма-Шуба и возможность его реализации на графическом процессоре. Приведены оценки стойкости данного генератора и предложены решения по увеличению скорости работы генератора. На основе готового решения метода Вихрь Мерсена, входящего в пакет NVIDIA SDK, проведен анализ статистических свойств генератора. Сделан вывод о целесообразности использования графических процессоров при генерации псевдослучайных последовательностей.

Ключевые слова: *генератор, безопасность, графический процессор, метод Монте-Карло, метод генерации Блюм-Блюм-Шуба, многократная точность, псевдослучайная последовательность.*

Введение

Постановка проблемы и анализ существующих технологий. На сегодняшнее время среди всех известных генераторов ПСП, большинство имеют ряд серьёзных недостатков, а последовательности, полученные при помощи таких генераторов, не отвечают основным требованиям, которые выдвигаются к генераторам ПСП для использования в криптографических целях, например, требованиям свойства случайности, равновероятности появления бит в последовательности, независимости, однородности и др. [1].

Программирование параллельных систем в настоящее время активно развивается. Идея распараллеливания вычислений основана на том, что базовая задача разбивается на классы более мелких задач, которые могут быть выполнены одновременно. Существует три основных вида параллельных вычислений: параллельные вычисления на уровне битов; параллельные вычисления на уровне инструкций; параллельные вычисления на уровне данных.

Параллельные вычисления на уровне битов основаны на увеличении размера машинного слова (величина, равная разрядности регистра процессора). Машинное слово определяет следующие характеристики аппаратной платформы: разрядность данных, обрабатываемых процессором; максимальный объем оперативной памяти, напрямую адресуемой процессором; разрядность адресуемых данных (разрядность шины данных).

Переходя от аппаратной реализации к программной, необходимо иметь ввиду, что программа есть некий набор инструкций, выполняемых процессором. Параллельные вычисления на уровне инструкций предусматривают изменение порядка выполнения данных инструкций и их объединение между собой. Выполнение математической операции сразу над всеми элементами массива данных является основным принципом параллельных вычислений на уровне данных.

Широкое применение получил принцип компьютерных вычислений, позволяющий использовать параллельные вычисления на уровне данных (SPMD). Их использование наиболее актуально в вычислениях на GPU (graphics processing unit), в отличие от CPU (central processing unit). Архитектуры, адаптирующие принцип SPMD, такие как CUDA [2] и OpenCL [3], обладают высокими показателями доступности, как с точки зрения пользователя (кросплатформенное программное обеспечение с поддержкой операционных систем Windows, Linux, Mac OS), так и с точки зрения вычислительных возможностей (аппаратная поддержка целочисленных и побитовых операций) и продуктивности (разделяемая между потоками память). Именно поэтому мы будем использовать эти архитектуры для дальнейших исследований.

Основная часть

1. Метод генерации ПСП Блюма-Блюма-Шуба (BBS) и возможность его реализации на GPU

Генератор псевдослучайных чисел BBS представляет собой метод, который можно описать уравнением:

$$x_{n-1} = x_n^2 (\mathrm{mod}\, M),$$

где M является произведением двух простых чисел p и q, которые имеют особые свойства и описываются соотношениями: $p \equiv 3 \bmod 4$, $p = 2p_1 + 1$, $p_1 = 2p_2 + 1$, где p_1 и p_2 – простые числа.

Исходное значение генератора x_0 является квадратичным вычетом от M. Указанное означает, что существует целое число y, такое, что $y^2 = x_0 \bmod M$. Значение x_0 может быть получено путем выбора случайного сида (seed) s меньшего, чем M/2, но большего чем 0 и удовлетворяющего выражению: $НОД(s, M) = 1$, тогда $x_0 = s^2$.

Генератор BBS имеет периодичность и это означает, что после некоторого числа итераций он будет генерировать x_0 и сгенерированная последовательность будет повторяться. Периодичность может быть увеличена за счет выбора простых чисел p и q, для которых $НОД(p-1, q-1)$ будет как можно меньше.

Функция от натурального числа n - $\lambda(n)$, называемая Carmichael функция [4], определяется как наименьшее натуральное число m такое, что: $a^m \equiv 1 (\mathrm{mod}\, n)$. Возможно рассчитать периодичность, используя функцию $\lambda(\lambda(M))$, однако для того чтобы это было возможно, необходимо, чтобы выражения $p = 2p_1 + 1$ и $p_1 = 2p_2 + 1$ выполнялись для чисел p и q. Генератор BBS дает возможность перехода на любой шаг i в последовательности, если использовать следующее выражение:

$$x_n = x_0^{2^i \bmod \lambda(M)} (\mathrm{mod}\, M).$$

Количественная оценка стойкости генератора может быть выполнена с использованием таких показателей как: энтропия источника ключей H_x, период l_n (длина) повторения ПСП, основа алфавита m последовательности, вероятность перекрытия в пространстве или во времени битов Y_r и Y_μ и др.

Количественная оценка стойкости, проведенная Блюмом [5], позволила установить число бит, которые могут быть получены и использоваться в криптографических целях для каждой итерации генератора. Полученное значение – $\log_2(\log_2 M)$.

Поскольку безопасность генератора основывается на большом числе M, не меньшем чем 1000 бит, что превышает длину регистров любого совре-

менного процессора, поэтому требуется применение методов вычислений с многократной точностью.

Такие методы могут основываться на использовании позиционной системы счисления, где машинное слово некоторого процессора является цифрой в представлении числа, состоящего из некоторого количества машинных слов, а максимальное значение, которое можно хранить в машинном слове, является базой этой системы счисления. Использование алгоритма Монтгомери позволит ускорить выполнение операций умножения и возведения в квадрат, что необходимо при возведении числа в степени по модулю, когда модуль составляет порядка 1000 бит. Также Монтгомери [6] определил способы выполнения модульных операций с использованием операций сложения и умножения.

Операцию умножения, определенную в алгоритме Монтгомери, можно использовать для реализации BBS (Blum Blum Shub) метода на GPU (Graphics processing unit).

2. Метод ПСП Вихрь Мерсена и его реализация на GPU

Для синтеза генератора ПСП, который бы обладал высокими скоростными показателями необходимо, чтобы использовались алгоритмы, основанные на элементарных операциях с двоичными (битовыми) строками. Наряду с генераторами, основанными на регистрах сдвига с линейными обратными связями (LFSR) используются генераторы, основанные на регистрах сдвига с обобщенными обратными связями (GFSR), к числу которых принадлежит и рассматриваемый ниже генератор Вихрь Мерсена [7].

Метод Вихрь Мерсена на выходе генератора получает битовые вектора фиксированной длины посредством следующего рекуррентного соотношения:

$$x_{k+n} = x_{k+m} + (x_k^{upper} \mid x_{k+1}^{lower}) \bullet A,$$

где $n > m$ - фиксированные положительные целые числа; $x_k, k = 0,1,...$ - последовательность битовых векторов фиксированной длины w, равная 32; $x_k^{upper} \mid x_{k+1}^{lower}$ - конкатенация r старших битов слова X_k и $w - r$ младших битов слова X_{k-1}; $(x_k^{upper} \mid x_{k+1}^{lower}) \bullet A$ - произведение конкатенированного битового вектора на битовую матрицу A, выбираемую из следующего соотношения:

$$x \bullet A = \{x_{w-1}, x_{w-2}, ..., x_0\} \bullet \begin{Vmatrix} 0 & 1 & 0 & 0 & & & \\ 0 & 0 & 1 & 0 & \vdots & & \\ 0 & 0 & 0 & 1 & & & \\ & & & & 1 & & \\ & & & \cdots & & \ddots & \\ & & & & & & 1 \\ a_{w-1} & a_{w-2} & & & & a_1 & a_0 \end{Vmatrix},$$

где $x = \{x_{w-1}, x_{w-2}, ..., x_0\}$,

$a = \{a_{w-1}, a_{w-2}, ..., a_0\}$.

Битовые значения складываются по модулю 2 (т.е. с использованием XOR операции).

Вихрь Мерсена имеет следующий набор параметров:

w - длина слова;

n, m - степень рекурсии и средний терм;

r - точка разделения между x_k^{upper} и x_k^{lower};

a - битовый вектор, нижняя строка матрицы A;

l, u, s, t - параметры сдвига;

b, c - маски, представленные в виде битовых векторов.

Несмотря на большой период генератора, равный $2^{19937} - 1$, и на высокие показатели прохождения тестов DIEHARD [8], что свидетельствует о хороших статистических свойствах, последовательность, сгенерированная таким генератором может быть использована в криптографических целях только совместно с алгоритмами хеширования.

Метод, на основе которого реализован генератор Вихрь Мерсена также требует большого объема памяти для хранения своих состояний на разных стадиях алгоритма. В случаях, когда точность вычислений преобладает над скоростью, имеет смысл использование данного метода генерации. Готовая реализация данного метода содержится в NVIDIA SDK (комплект средств разработки от NVIDIA).

3. Методы параллельных вычислений Монте-Карло

Реализация генераторов псевдослучайных чисел на графических процессорах наиболее применима в контексте симуляции методом Монте-Карло, так как решение задач с применением этого метода достаточно хорошо распараллеливается и, в тоже время, требует значительных вычислительных мощностей для проведения большого количества итераций.

Важнейшим принципом построения методов Монте-Карло есть сведение задачи к расчету математических ожиданий [9]. Для того, чтобы приближенно вычислить некоторую скалярную величину α, надо определить такую случайную величину ξ, что $M[\xi] = \alpha$, тогда вычислив N независимых значений $\xi_1, ..., \xi_N$ величины ξ, можно считать, что $\alpha \approx \frac{\xi_1 + \xi_2 + ... + \xi_N}{N}$.

Численные методы Монте-Карло, используемые для решения математических задач при помощи моделирования случайных величин (для моделирования случайных величин используются генераторы ПСП), требуют больших вычислительных мощностей.

Определим несколько подходов, позволяющих распараллелить методы Монте-Карло.

Первый подход заключается в разделении последовательности на потоки. Пусть p - период используемого генератора псевдо случайных чисел, а n - длина последовательности, такая что $n \le p$. Тогда последовательность длиной n разделяется на последовательности длины $n_1 = n_2 = ... = n_i$ по одной из каждого P потока. Сложность данного подхода заключается в вычислении начального элемента в i-м потоке, что является трудоемкой операцией.

Второй подход заключается в параметризации (использовании различных входных параметров для генератора ПСП), что позволяет использовать в разных потоках разные независимые случайные последовательности. При таком подходе необходимо убедиться в том, что сгенерированные последовательности независимы между собой.

С использованием параллельных вычислений проблема использования больших вычислительных мощностей легко решаема и делает доступной использование методов Монте-Карло. При этом методы Монте-Карло являются достаточно легко программируемыми и позволяют рассчитывать задачи, не доступные для классических численных методов, чем и обосновывается выбор данных методов для дальнейших исследований.

Выводы

В контексте базовых арифметических алгоритмов операция умножения может быть легко распараллелена в отличие от операции деления. Распараллеливание операции сложения, можно выполнить аналогично с реализацией сумматоров с переключением переноса.

Выигрыш в производительности при переносе реализации метода BBS (Blum Blum Shub) на GPU (Graphics processing unit) будет только в том случае, если GPU параллельно будет выполнять ряд сопутствующих задач, например, реализация методов Монте-Карло. Если ставить целью непосредственную реализацию рассмотренных в статье алгоритмов генерации, то затраты на построение связи между CPU и GPU будут значительно большие, чем выигрыш от переноса вычислений на GPU.

Список литературы

1. Горбенко І.Д. Навчальний посібник «Захист інформації в інформаційно-телекомунікаційних системах» / І.Д. Горбенко, Т.О. Гріненко. – Х.: ХНУРЕ. – 2003. – 368 с

2. NVIDIA Corporation. What is cuda. URL: http://developer.nvidia.com/what-cuda. [Accessed January 30, 2012].

3. NVIDIA Corporation. OpenCL Programming Guide, 4.0 edition, 2011.

4. Erdős Paul; Pomerance Carl; Schmutz, Eric (1991). "Carmichael's lambda function". Acta Arithmetica 58: 363–385.

5. Leonore Blum, Manuel Blum, and Michael Shub. A simple unpredictable pseudo-random number generator. SIAM Journal of Computing, 15(2):364-383, 1986.

6. Peter L. Montgomery. Modular multiplication without trail division. Mathematics of Computation, 44(170):519{521, 1985.

7. M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Transactions on Modeling and Computer Simulation, 8:3–30, 1998.

8. Marsaglia G.DIEHARD: A battery of tests of randomness. 1996. http://stat.fsu.edu/geo/diehard.html.

9. Дональд Кнут. Искусство программирования / Дональд Кнут // Получисленные алгоритмы. The Art of Computer Programming. – Vol.2. Seminumerical Algorithms. – 3-е изд. – М.: «Вильямс»/ – 2007. – С. 832.

Поступила в редколлегию 18.02.2014

Рецензент: д-р техн. наук, проф. В.А. Краснобаев, Полтавский национальный технический университет. им. Ю. Кондратюка, Полтава.

МЕТОДИ ПОБУДОВИ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ З ВИКОРИСТАННЯМ ГРАФІЧНИХ ПРОЦЕСОРІВ

О.А. Замула, Д.О. Семченко

Розглянуто базові поняття, що використовуються при програмуванні паралельних систем. Описан метод генерації Блюма - Блюма - Шуба і можливість його реалізації на графічному процесорі . Наведено оцінки стійкості даного генератора та запропоновані рішення щодо збільшення швидкодії роботи генератора. На основі готового рішення метода Віхрь Мерсена, що входить в пакет NVIDIA SDK , проведений аналіз статистичних властивостей. Зроблений висновок про доцільність використання графічних процесорів при генерації псевдовипадкових послідовностей.

Ключові слова: *генератор, безпека, графічний процесор, метод Монте-Карло, мето Блюм-Блюм-Шуба, багаторазова точність, псевдовипадкова послідовність.*

METHOD FOR CONSTRUCTING GENERATING PSEUDO RANDON NUMBER GENERATORS BASED ON PARALLEL COMPUTING USING GPUS

A.A. Zamula, D.A. Semchenko

Describes the basic method used in programming parallel systems. Described method for the generation of the Blum-Blum-Shub and the possibility of its implementation on the GPU. The estimations of resistance of this generator and proposed solutions to increase the speed of the generator. On the basis of a turnkey solution algorithm Mersenne Twister included in the package NVIDIA SDK, analyzed statistical properties. Conclusion about the feasibility of using GPUs in generating pseudo-random sequences described in this article.

Keywords: *generator, security, graphics processor, Monte Carlo, Blum-Blum-Shub, multi-precision, pseudo-random sequence.*