Аппаратная поддержка некоторых алгоритмов метода Монте-Карло

Яковенко Андрей Богданович, гр. 522

Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра статистического моделирования

Научный руководитель: д.ф.-м.н., проф. Ермаков С.М.
Рецензент: к.ф.-м.н., ассистент Коробейников А.И.

Санкт-Петербург
2011г.

Постановка задачи

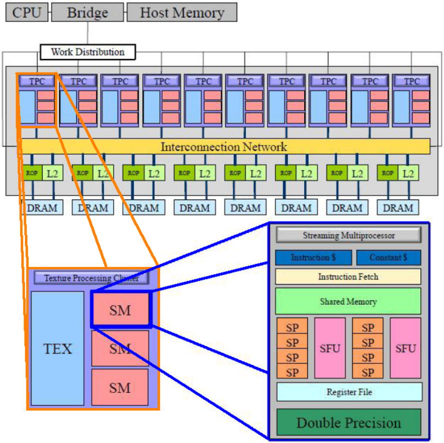
- Метод Монте-Карло является одним из мощнейших методов для численного решения многих задач в областях физики, математики, экономики, медицины и др.
- Как показано в работах С.М. Ермакова, метод Монте-Карло помимо всего прочего интересен тем, что обладает естественным свойством параллелизма как представитель класса п.р.-алгоритмов.
- В данный момент наиболее доступной вычислительной базой, на которой доступны параллельные вычисления, является архитектура GPGPU (General Purpose Graphic Processor Unit).
- В дипломной работе рассматриваются некоторые алгоритмы метода Монте-Карло с точки зрения их параллельной реализации с использованием GPGPU.

Вычисление интеграла.

$$E\left(\frac{f(\xi)}{p\xi}\right) = \int_D f(x)d\mu$$

- Генерация равномерных случайных величин u_i.
- Генерация на основе u_i величин ξ_j.
- Вычисление значений $f(\xi_i)$.
- Вычисление среднего $\frac{1}{N}\sum_{j=0}^{N} f(\xi_j)$.

Видеокарта, поддерживающая технологию CUDA является SIMD сопроцессором и устроена следующим образом:



$U(0,1)$ псевдослучайная величина

- t_ξ — время генерации величины на внешнем устройстве.
- t_ξ^{CPU} — время генерации величины на компьютере.

Тогда ускорение генерации случайной величины можно будет оценить как:

$$P_p = \frac{t_\xi^{CPU}}{t_\xi}$$

- m_ξ — объем памяти, занимаемый случайной величиной.
- B — скорость передачи данных с внешнего устройства.
- t_ξ^{CPU} — время генерации величины на компьютере.

Тогда можно оценить максимальное ускорение генерации случайной величины внешним устройством.

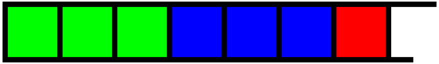
$$P_r = \min\left(\frac{t_\xi^{CPU}}{t_\xi}, \frac{t_\xi^{CPU} \cdot m_\xi}{B}\right)$$

Методы разбиения.

Алгоритм

Инициализация: $C_1^{I_t, I_m} \leftarrow F^{\lfloor \frac{N}{K_m K_t} \rfloor}(Seed)$

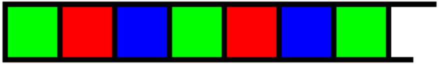
Генерация: $C_{i+1}^{I_t, I_m} \leftarrow F(C_i^{I_t, I_m})$



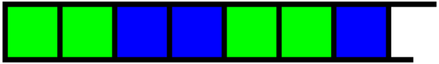
Алгоритм

Инициализация: $C_1^{I_t, I_m} \leftarrow F^{I_t + I_m K_t}(Seed)$

Генерация: $C_{i+1}^{I_t, I_m} \leftarrow F^{K_m K_t}(C_i^{I_t, I_m})$



Комбинированный метод



Алгоритм

Инициализация: $C_1^{I_t,\,I_m} \leftarrow F^{L(I_t+I_m K_t)}(Seed)$

Генерация:

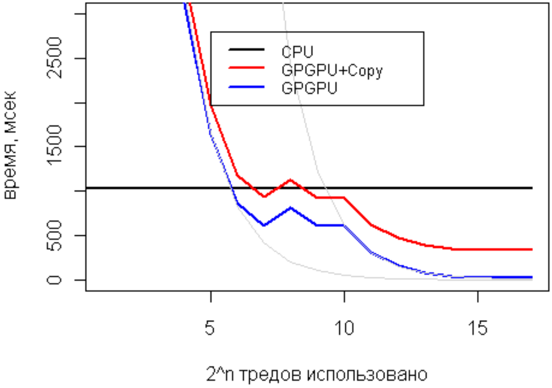
$$C_{L\cdot i+1}^{I_t,\,I_m} \leftarrow F^{K_m K_t (L-1)+1}(C_{i-K_t K_m}^{I_t,\,I_m})$$

$$C_{L\cdot i+2}^{I_t,\,I_m} \leftarrow F(C_{L\cdot i+1}^{I_t,\,I_m})$$

$$\vdots$$

$$C_{L\cdot i+L}^{I_t,\,I_m} \leftarrow F(C_{L\cdot i+L-1}^{I_t,\,I_m})$$

Анализ мультипликативного ДПСЧ



$$P_p = \frac{1057ms}{22ms} = 48, \ P_r = \frac{1106ms}{347ms} = 3$$

Более сложные распределения.

К характеристикам алгоритма генерации псевдослучайных чисел, имеющих заданное распределение можно отнести следующие величины:

- L — количество равномерно распределенных независимых псевдослучайных величин.
- t_a — среднее время вычислений.
- t_u — время генерации псевдослучайной величины.

Тогда общая формула будет выглядеть следующим образом:

$$P_r = \min\left(\frac{t_a^{CPU} + Lt_u^{CPU}}{t_a + Lt_u}, \frac{(t_a^{CPU} + Lt_u^{CPU}) \cdot m_\xi}{B}\right)$$

Простые распределения

Можно рассмотреть класс алгоритмов, что t_a и L являются константами:

Распределение	P_r	C_e
Равномерное (128-bit)	24.8	0.48
Равномерное (32-bit)	5	0.1
Нормальное (128-bit)	25	0.5
Показательное (128-bit)	25	0.5

Где $C_e = \frac{P_r}{P_p}$ — эффективность использования внешнего устройства.

Решение СЛАУ

Был реализован метод Уолкера для параллельного внешнего устройства.

$$P_r(Walker) = P_p(Walker)$$

Следовательно, можно использовать видеокарту для решения СЛАУ методом МК.

$$M_{global} = d(d+1)(m_{a_{ij}} + m_{p_{ij}} + m_{border} + 2m_{index})$$

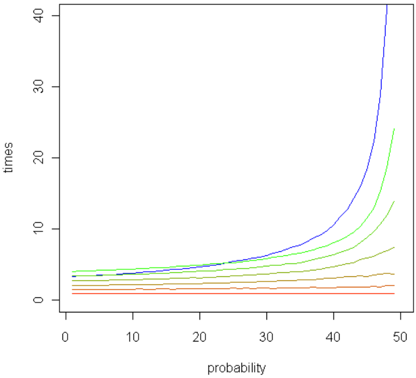
Для имеющегося устройства:

$$d \approx \sqrt{\frac{512Mb}{4b + 4b + 2 \cdot 2b}} \approx 6600$$

Сложные распределения

В общем случае, время t_a так же является случайной величиной.

$$t_\xi^{SIMD} = \max_{i=1}^{K} t_\xi$$



Методы решения проблемы

Алгоритм

- *Пока ξ не подойдет, вычислять ξ.*
- *Записать ξ.*

$$\max_{j=1}^{K} \sum_{i=1}^{N} t_a \leqslant \sum_{i=1}^{N} \max_{j=1}^{K} t_a$$

Алгоритм

- *Вычислить ξ.*
- *Если ξ подходит, записать.*

Аналогично можно «смягчить» генрацию р-я Пуассона.

Заключение

- В работе были рассмотрены возможности GPGPU как внешнего генератора псевдослучайных чисел.
- Были выделены свойства алгоритмов, обуславливающие их эффективную реализацию на GPGPU.
- Была написана программа, реализующая 128-битный мультипликативный генератор и использующие его генераторы нескольких стандартных случайных величин.