Нижегородский государственный университет им. Н.И. Лобачевского

Факультет Вычислительной математики и кибернетики

Реализация явной разностной схемы решения уравнения теплопроводности

Бастраков С.И.

ВМК ННГУ

bastrakov@vmk.unn.ru

Молодежная научная школа «Суперкомпьютерные технологии и высокопроизводительные вычисления в образовании, науке и промышленности» 5 – 8 ноября 2014 г., г. Нижний Новгород

Постановка задачи

- Рассматривается задача Дирихле для двумерного уравнения теплопроводности:

$$\frac{\partial u(x,y,t)}{\partial t} = \Delta u(x,y,t) + f(x,y,t)$$

$$a \le x \le b, \qquad c \le y \le d, \qquad 0 \le t \le T$$

$$u(x,y,0) = u_0(x,y)$$

ГУ
$$\begin{cases} u(a,y,t) = \varphi_1(y,t),\ u(b,y,t) = \varphi_2(y,t) \\ u(x,c,t) = \varphi_3(x,t),\ u(x,d,t) = \varphi_4(x,t) \end{cases}$$

Постановка задачи

- Введем равномерную сетку с шагами по пространству и времени Δx, Δy и Δt соответственно.
- Обозначим $v^{(k)}_{ij}$ – сеточное значение в точке (x_i, y_j) в k-й момент времени.
- Будем использовать следующую явную разностную схему для внутренних узлов:

$$v^{(k+1)}_{ij} = v^{(k)}_{ij} + \Delta t * \left(f(x_i, y_j) + \frac{v^{(k)}_{i+1j} - 2v^{(k)}_{ij} + v^{(k)}_{i-1j}}{\Delta x^2} + \frac{v^{(k)}_{ij+1} - 2v^{(k)}_{ij} + v^{(k)}_{ij-1}}{\Delta y^2} \right)$$

Постановка задачи

- Будем производить вычисления слой за слоем по времени. Вычисления всех элементов одного слоя по времени независимы.
- Для простоты будем считать, что граничные условия стационарны (не зависят от времени). Это означает, что при программной реализации функции вычисления значений в очередном слое будем вычислять лишь значения во внутренних узлах.

Задание

- Ознакомьтесь с предоставленным проектом CUDA_HeatEquation, убедитесь в том, что он компилируется, запускается и выдает “Test FAILED.”.
- В файле heat_equation.cu реализуйте ядро, которое вычисляет новые значения в одном слое по времени (выполняет работу аналогично функции step в heat_equation.cpp). Не переделывайте функцию step_gpu в ядро – она нужна для других целей.
- Пусть каждый поток вычисляет один элемент, используя двумерные индексы.
- Для вызова функции правой части из ядра создайте аналог функции f из heat_equation.cpp со спецификатором __device__.

Задание

- Реализуйте функцию step_gpu в файле heat_equation.cu, которая вычисляет количество блоков и потоков и запускает ядро. Не меняйте интерфейс функции.
- Рекомендуется использовать блоки фиксированного размера, например, 16 на 16.
- Данные по указателям в аргументах step_gpu уже лежат на GPU, копирования не требуется.
- Убедитесь, что полученный результат совпадает с результатом CPU-версии, в этом случае будет выведено сообщение “Test PASSED.”