Задача 33: «Вычислить методом последовательных приближений распределение значений температуры точек верхней грани параллелепипеда размером k \* m \* n, имеющего внутри полость в виде цилиндра. Теплопроводность материала не равна нулю. Нижняя грань параллелепипеда имеет постоянную температуру 0. Одна из сторон верхней грани имеет температуру T1, противоположная - температуру T2.»

Файл для компиляции находится по пути: «cd /lr2/src/»

Для компиляции программы: «/usr/local/cuda/bin/nvcc main.cu -o main.o»

Для запуска: «./main.o»

Программа потребует ввод:

«Run test? (y/n): » - запрос запуска теста блоков;

«Show iteration? (y/n): » - запрос показа всех итераций вычислений для сравнения с CPU вычислениями;

«Show GPU information? (y/n):» - запрос вывода информации о видеокартах;

«Enter size (x,y,z, radius):» - ввод размера параллелепипеда, внутри которого находится цилиндр;

«Enter alpha:» - ввод коэффициента теплопроводности (рекомендуется кэффициента 0.05351 для более успешного распространения температуры);

«Enter eps:» - ввод коэффициента точности;

«Enter maximum of iterations:» - ввод максимума итераций.

После успешного выполнения программа создает файлы «gpumatrux» и «cpumatrix» – ответы «симуляции» распротранения температуры. На экран выводится:

«GPU: Execute time = 0,438463814 took seconds» – время вычисления на GPU.

«CPU: Execute time = 0,546525324 took seconds» – время вычисления на CPU

В общем, можно выделить паралельность в блоках: построения матрицы и итераций.

Стоит также заметить, размер трехмерной матрицы напрямую зависит от мощности железа (для GPU). Распределение не идет на итерации, т.к. они зависимы от предыдущих значений, поэтому в методе реализован линейный перебора элементов. Но реализована паралельная сумма соседей рассматриваемого элемента в матрице: при достижении метода запроса на GPU (как i j k, например, равны (1,1,1)) начинается рассмотр и сумма всех температур соседей, включая нынешнешнюю точку (таким образом рассматриваются коориданты (0,0,0), (0,0,1), (0,1,0)… (2,2,1), (2,2,2)). Также предусмотрена защита от нелегального доступа к памяти или рассмотрения ненужных точек, т.е. заходят ли точка в радиус матрицы или вообще выходят из диапазона (например, если рассматриваемая точка (0,0,0), то не будет переборов, например (-1,0,0) или (2,0,0).

Построение матрицы работает по системе, на нынешний момент, на статичном кол-ве блоков, но динамическом кол-ве потоков, не превышающих максимальное кол-во потоков на блок (1024), также не нарушают Warp-кратность (число 32 на RTX 3050). Но в данном методе есть небольшие недостатоки:

• GPU может просто отказать в выполнении метода на каком-то очень огромном размере матрицы, но, как сказано выше, вычисление зависит от мощности железа;

• Результат заполнения напрямую зависит от размера (блоки могут несовсем правильно передать последовательность при каком-либо размере не кратном 2, что могут нарушить формулу координат, при этом заполнить неправильно какие-то ячейки), поэтому перед началом может вывести сообщение «WARNING: scary input (multiples of 2 are required)!»;

• Время итерации на GPU может быть замедленно из-за синхронизации блоков и потоков, поэтому в каких-то моментах таймер может показывать, что CPU быстрее GPU.

Суммирование соседей происходит на многоблочном, но на 1 поток. Это связано с тем, что нет необходимости многопотока на данный момент, ибо метод получает значение с каждого блока, т.е. с элемента в матрице (можно было просто использовать 1 блок, но многопоток для ускорения решения или несколько блоков и несколько потоков, но оно не особо было необходимо).