**Теория Параллелизма**

**Отчет:**

Задание 5. «Уравнение теплопроводности на нескольких GPU»

Емельянов Алексей Алексеевич

Группа 21932

Новосибирск 2023

**Цель работы**

Цель данной работы состоит в том, чтобы реализовать решение теплопроводности методом Якоба для двумерной сетки. Произвести оптимизацию предоставленного кода из лекции. Перенести программу на GPU используя CUDA. Распараллеливание на несколько GPU должно производиться с использованием MPI. Операцию редукции (вычисление максимального значения ошибки) в рамках одного MPI процесса реализовать с использованием библиотеки CUB. Подсчет глобального значения ошибки, обмен граничными условиями реализовать в двух вариантах: с использованием MPI. ([ссылка на презентацию](https://classroom.google.com/u/1/c/NTg0Nzg0MTE5Mzgy/p/NTk0MTI0MzYzNDg5?pli=1)).

Для компиляции версии с MPI использовалась:

**mpic++ paralel5.cu -o mpi\_5.pg -O2**

Для данной работы в качестве измерения времени выполнения программы использую команду при запуске программы “*time*” предоставленной на лекции.

Для запуска собранный программы используется следующая команда

*time (имя файлы) (значение ошибки) (размер сетки одной стороны) (максимальное число итераций)*

*time mpiexec -n 1 ./mpi\_5.pg 0.000001 1024 1000000*

*ссылка на гитхаб - https://github.com/EmelyanovAlexey/Paralel/tree/master/task\_5*

**Таблица результатов выполнения программы**

|  |  |  |  |
| --- | --- | --- | --- |
| **CPU** | | | |
| **Размер ячейки** | **Время выполнения** | **Результат ошибки** | **Количество итераций** |
| 128x128 | 1,389s | 9.7835e-07 | 11081 |
| 256x256 | 13,651s | 9.99204e-07 | 37301 |
| 512x512 | 199,024s | 9.99681e-07 | 120361 |
| 1024x1024 | 2254,125s | 9.99989e-06 | 1000000 |

|  |  |  |  |
| --- | --- | --- | --- |
| **CPU MULTI** | | | |
| **Размер ячейки** | **Время выполнения** | **Результат ошибки** | **Количество итераций** |
| 128x128 | 1.364s | 9.93435e-07 | 11081 |
| 256x256 | 4.419s | 9.99204e-07 | 37301 |
| 512x512 | 26.541s | 9.99681e-07 | 120361 |
| 1024x1024 | 42.748s | 9.99989e-07 | 384341 |

|  |  |  |  |
| --- | --- | --- | --- |
| **GPU openacc** | | | |
| **Размер ячейки** | **Время выполнения** | **Результат ошибки** | **Количество итераций** |
| 128x128 | 4.398s | 9.93435e-07 | 11081 |
| 256x256 | 6.660s | 9.99204e-07 | 37301 |
| 512x512 | 8.634s | 9.99819e-07 | 120361 |
| 1024x1024 | 36.752s | 9.99989e-07 | 364621 |

|  |  |  |  |
| --- | --- | --- | --- |
| **GPU cublas** | | | |
| **Размер ячейки** | **Время выполнения** | **Результат ошибки** | **Количество итераций** |
| 128x128 | 0.937s | 9.7074e-07 | 11110 |
| 256x256 | 1.650s | 9.86088e-07 | 37370 |
| 512x512 | 3.876s | 9.98112e-07 | 120392 |
| 1024x1024 | 27.683s | 9.98906e-06 | 364711 |

|  |  |  |  |
| --- | --- | --- | --- |
| **GPU cuda** | | | |
| **Размер ячейки** | **Время выполнения** | **Результат ошибки** | **Количество итераций** |
| 128x128 | 0.465s | 9.0734e-07 | 3040 |
| 256x256 | 0.773s | 9.86088e-07 | 10320 |
| 512x512 | 1.884s | 9.98112e-07 | 339600 |
| 1024x1024 | 25.724s | 9.993e-06 | 1000000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **GPU cuda + MPI** | | | | | |
| **Размер ячейки** | **Время выполнения 1** | **Время выполнения 2** | **Время выполнения 3** | **Время выполнения 4** | **Кол итер** |
| 128x128 | 0.601s | 1.097 s | 1.764 s | 2.097 s | 3040 |
| 256x256 | 1.137 s | 2.004 s | 2.841 s | 3.352 s | 10320 |
| 512x512 | 2.470 s | 6.928 s | 7.492 s | 7.995 s | 339600 |
| 1024x1024 | 27.107 s | 29.642 s | 25.107 s | 22.439 s | 1000000 |
| 2048x2048 | 94.013 s | 76.496 s | 66.157 s | 59.697 s | 1000000 |
| 4096x4096 | 375.523 s | 214.742 s | 162.127 s | 131.531 s | 1000000 |
| 8192x8192 | 1458.143 s | 754.522 s | 596.677 s | 442.317 s | 1000000 |

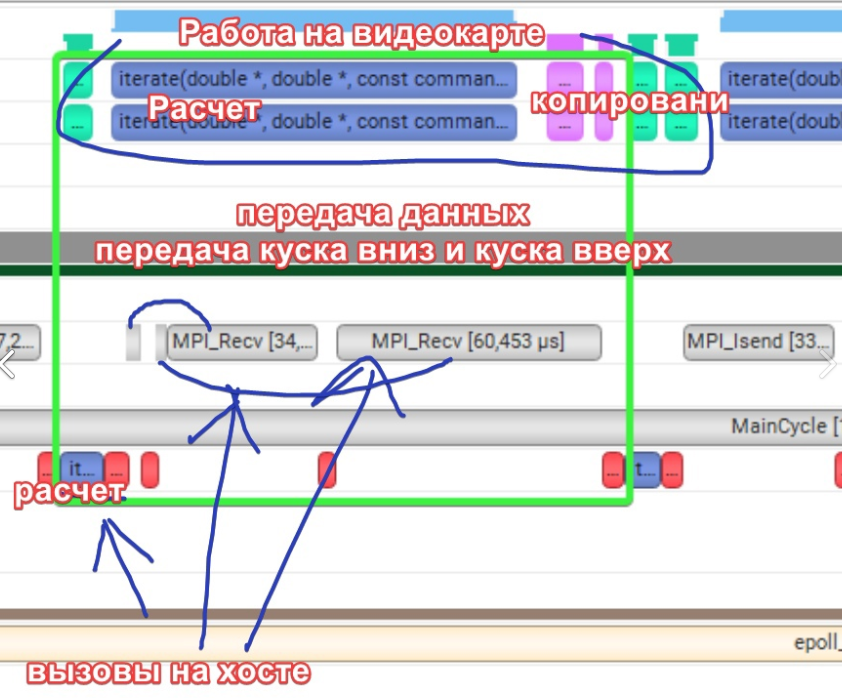
**Графики**

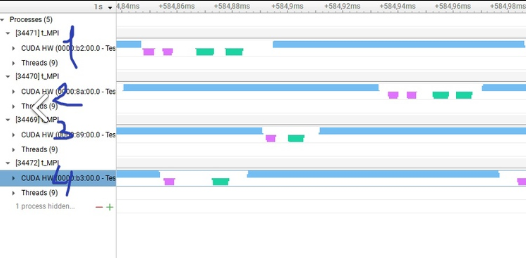
**Результат**

Использование MPI для запуска нескольких процессов мы можем добиться возможности использовать несколько GPU в параллельном режиме. Для маленьких сеток данный способ использовать не целесообразно, так как присутствуют расходы синхронизации.

**Профилировщик**

вычислительный вызов шаг по сетке и передача граничных значений в асинхронном режиме в рамках одного устройства. Проблем по ходу вычисления не происходит так как данные передаются в рамках хоста, а перенос их на устройство происходит после окончания шага.

****

****