МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «Параллельная обработка данных»

Технология MPI и технология CUDA. MPI-IO

Вариант 3. MPI_Type_hindexed

Выполнил: А.В. Синявский
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы.

Совместное использование технологии MPI и технологии CUDA.
Применение библиотеки алгоритмов для параллельных расчетов Thrust. Реализация метода Якоби. Решение задачи Дирихле для уравнения Лапласа в трехмерной области с граничными условиями первого рода. Использование механизмов MPI-IO и производных типов данных.
Требуется решить задачу описанную в лабораторной работе No7, используя возможности графических ускорителей установленных на машинах вычислительного кластера. Учесть возможность наличия нескольких GPU в рамках одной машины. На GPU необходимо реализовать основной расчет. Требуется использовать объединение запросов к глобальной памяти. На каждой итерации допустимо копировать только граничные элементы с GPU на CPU для последующей отправки их другим процессам. Библиотеку Thrust использовать только для вычисления погрешности в рамках одного процесса.

Входные данные.

На первой строке заданы три числа: размер сетки процессов. Гарантируется, что при запуске программы количество процессов будет равно произведению этих трех чисел. На второй строке задается размер блока, который будет обрабатываться одним процессом: три числа. Далее задается путь к выходному файлу, в который необходимо записать конечный результат работы программы и точность ε. На последующих строках описывается задача: задаются размеры области l_x, l_y и l_z, граничные условия: u_{down}, u_{up}, u_{left}, u_{right}, u_{front}, u_{back}, и начальное значение u^0.

Выходные данные.

В файл, определенный во входных данных, необходимо напечатать построчно значения в ячейках сетки в формате с плавающей запятой с семью знаками мантиссы.

Программное и аппаратное обеспечение

Nvidia GeForce GTX 660

Compute capability: 3.0
Графическая память: 2048MB
Регистров на блок: 65536
Нитей на блок: 1024
Мультипроцессоров: 5
Всего ядер: 960

Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz

Тактовая частота: 3.4 GHz
Кэш-память: 6 MB

Оперативная память
Объём: 8 GB

Жёсткий диск
Объём: 2 TB

Программное обеспечение
OS: Windows 10
IDE: Visual Studio Code
CUDA: v10.2 nvcc

Метод решения

Пусть нулевой процесс считывает начальные данные и отправляет всем остальным процессам. Далее каждый процесс заполняет изначальные данные и начинает вычислительный цикл. В этом цикле процессы сначала обмениваются граничными значениями (необходимо сделать несколько условий по расположению процессов в сетке, чтобы ни один из граничных процессов не повис в вечном ожидании данных), затем для каждой ячейки вычисляют новое значение, параллельно считая максимальную ошибку в рамках процесса. Затем процессы обмениваются друг с другом этой максимальной ошибкой. Максимум из максимальных ошибок по всем процессам – критерий остановки вычислительного цикла для всех процессов.
После остановки вычислительного цикла все процессы записывают результаты вычислений в один общий файл, согласно пользовательскому типу данных.

Описание программы

- Init_data
  Ядро, заполняющее данные на видеокарте
- To_send_forward
  Ядро, заполняющее буферы для отправки границ «вперёд» на видеокарте
- To_send_backward
  Ядро, заполняющее буферы для отправки границ «назад» на видеокарте
- Receive_after_forward
  Ядро, копирующее границы из буферов в основной массив данных
- Receive_after_backward
  Ядро, копирующее границы из буферов в основной массив данных
- Calc
  Основное вычислительное ядро. Считает новые значения сетки и заполняет массив с погрешностями, по которому далее будет найден максимум

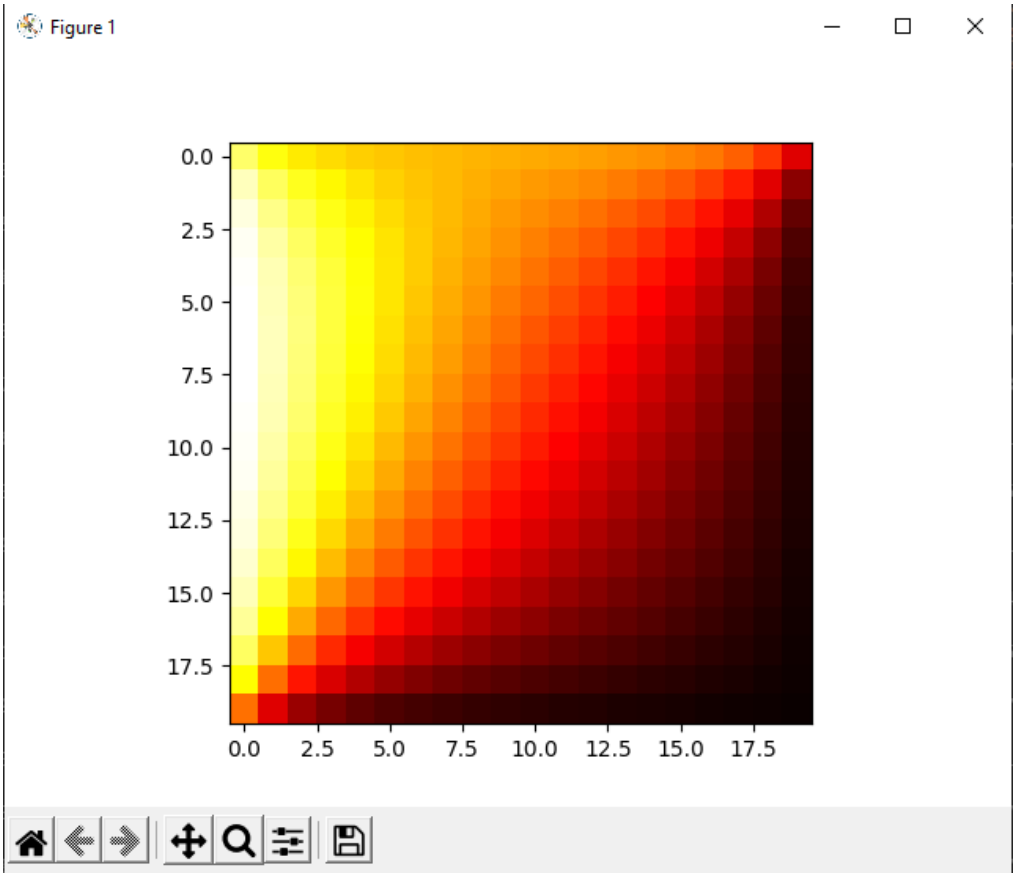
- Main
  Основная функция, исполняющаяся на процессоре. Отвечает за распределение процессов по машинам, работу с памятью, вызовом ядер, вычислением критерия остановки при помощи вызова функций библиотеки thrust. Также в этой функции описывается пользовательский тип данных для записи данных процессами в файл, и собственно происходит сама запись

Результаты

Замеры времени. Общий размер сетки – 20x20x40 ячеек

Число процессов	CPU	GPU
1	1.667767 с	0.742268 с
2	0.837142 с	0.801399 с
16	0.744788 с	0.948677 с
64	1.41726 с	1.035669 с

Визуализация результатов.
Изображение построено по результатам теста
1 1 1
20 20 1
mpi.out
1e-10
1.0 1.0 2.0
7.0 0.0 5.0 0.0 3.0 0.0
5.0



Выводы

Проделав работу, я изучил принципы организации работы с файлами на кластере при помощи технологии MPI, познакомился с функционалом MPI, отвечающим за создание пользовательских типов данных.