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

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

Москва, 2020

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 №7, 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.

Запись результатов в файл должна осуществляться параллельно всеми процессами. Необходимо создать производный, тип данных, определяющий шаблон записи данных в файл.

Вариант 3. Варианты конструктора типа MPI_Type_hindexed.

Программное и аппаратное обеспечение

Видеокарта	GeForce GTX 1650
Compute capability	7.5
Графическая память	3911 Мб
Разделяемая память	48 Кб
Константная память	64 Кб
Количество регистров на блок	65536
Максимальное кол-во блоков	2147483647*65535*65535
Максимальное кол-во нитей в блоке	1024
Кол-во мультипроцессоров	16
Ядер CUDA	896

Процессор	AMD Ryzen 5 3550H
ОЗУ	8 Гб
ЖД	

Операционная система	Ubuntu 20.04.6 LTS
IDE	VS Code
Компилятор	nvcc V10.1, mpi V3.3.2

Метод решения

Корневой процесс читает входные данные и рассылает их всем остальным процессам при помощи bcasat. Далее начинается сам рассчет.

Каждая его итерация состоит из 3 шагов: обмен граничными условиями, пересчет, вычисление погрешности.

Обмен границами происходит посредством sendrecv. Каждый процесс посылает все свои граничные значения «соседям» и принимает обратно их граничные значения.

Далее для каждой ячейки рассчитывается новое значение на основе предыдущих значений соседних ячеек. При этом все вычисления проводятся на GPU. Так же учтена возможность наличия на машине нескольких GPU. Для этого область, делится по одной из осей (по оси с макисимальным размером) между всеми GPU, доступными на машине.

В ходе рассчета вычисляется максимальная погрешность по всем ячейкам для каждого блока при помощи библиотеки *thrust*. Далее при помощи allreduce вычисляется максимум погрешности уже по всем ячейкам.

Программа продолжает вычисления, пока погрешность не станет меньше заданной пользователем точности.

После проведения расчета каждый процесс самостоятельно записывает данные в файл в сетевой файловой системе. Для пущего удобства создаются два типа данных: строка – тип, массив символов, в который записывается block_size_x элементов в формате %e. Для разметки самого же файла с помощью MPI_Type_hindexed создается сложный тип данных с различными смещениями из строкового типа. Далее одной инструкицей все накопленные данные записываются в файл.

Описание программы

Вся программа реализована в одном файле lab8.cpp. Основная логика реализована прямо в функции main. Сами вычисления по формуле Якоби производятся для всех ячеек блока ядром compute. Так же для удобства чтения и запии границ блока реализованы вспомогательные ядра *(get/set)_(frontback/leftright/updown)_border_kernel*.

Для идентификации сообщений в процессе обмена граничными условиями используется как номер процесса, так и тег т. к. каждый процесс обменивается от 3-6 сообщениями. В качестве тега передается значение LEFT, RIGHT и т. д

Результаты

Тестирование ядер с различными конфигурациями

Область 8x8x8 ячеек с точностью 0.001.

Параметры сетки	Время исполнения
8 гридов 8 блоков	0m 0,268s
64 грида 64 блока	0m 0,250s
512 гридов 512 блоков	0m 0,257s

Лучший результат: 0m 0,250s при 64 гридах, 64 блоках.

Область 64x64x64 ячеек с точностью 0.001.

Параметры сетки	Время исполнения
8 гридов 8 блоков	0m 12,030s
64 грида 64 блока	0m 2,785s
512 гридов 512 блоков	0m 2,410s

Лучший результат: 0m 2,410s при 512 гридах, 512 блоках.

Область 128x128x128 ячеек с точностью 0.001.

Параметры сетки	Время исполнения
8 гридов 8 блоков	2m 27,410s
64 грида 64 блока	0m 23,169s
512 гридов 512 блоков	0m 15,706s

Лучший результат: 0m 15,706s при 512 гридах, 512 блоках.

Сравнение с CPU

Маленький файл, полученный из bmp-файла 800x600 (1,8 мб)
Результат: 0m 0,003s

Средний файл, полученный из bmp-файла 1920x1080 (7,9 мб)
Результат: 0m 6,457s

Большой файл, полученный из bmp-файла 6000x4000 (91,6 мб)
Результат: 3m 3,447s

Выводы

Данный алгоритм по-прежнему может быть использован для решения задачи Дирихле для уравнения Лапласа с граничными условиями 1-го рода в прямоугольной трехмерной области.

Основные сложности в реализации программы возникли в написании обмена граничными условиями между блоками и разделении вычислений на несколько GPU в рамкаха одной машины. В решении данной задачи программа с использованием MPI и CUDA значительно превзошла стандартную реализацию на C++.