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. О. Тояков

Группа: М8О-407Б-18

Преподаватели: К. Г. Крашенинников,

А. Ю. Морозов

# 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.

Вариант № 2: MPI\_Type\_hvector

# Программное и аппаратное обеспечение

Device: GeForce MX250

Размер глобальной памяти: 3150381056

Размер константной памяти : 65536

Размер разделяемой памяти: 49152

Регистров на блок: 32768

Максимум потоков на блок: 1024

Количество мультипроцессоров : 3

OS: Linux Ubuntu 18.04

Редактор: VSCode

Компилятор: nvcc версии 11.4 (g++ версии 7.5.0)

# Метод решения

Задача решается почти также, как и предыдущая. Изменения состоят только в том, что все вычисления выполняются на GPU и запись в файл производится одновременно всеми потоками. Для этого используется произвольный тип данных MPI\_Type\_hvector. Отражаем структуру расположения сетки при разбиении на процессы. Затем указывается размер общей сетки, размер одного блока в зависимости от его номера. Для переноса граничных значений на GPU также пишем несколько ядер.

# Описание программы

#define \_i(i, j, k) – переход из трёхмерной сетки в одномерную для элементов.

#define \_ib(i, j, k) – переход из трёхмерной сетки в одномерную для блоков.

\_\_global\_\_ void calculation() – вычисление параметров сетки.

\_\_global\_\_ void calculate\_diff() – вычисление разности с предыдущей операцией.

\_\_global\_\_ void extractXPart() – ядра, заполняющие буфер для отправки сообщений.

\_\_global\_\_ void putXPart() – ядра, заполняющие данные о границах, пришедших с соседних блоков.

MPI\_Bcast() – передача данных всем процессам.

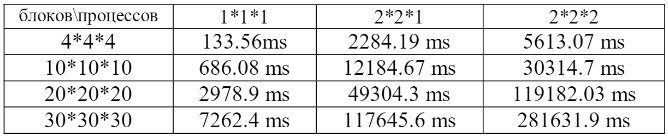
MPI\_Sendrecv – передача данных между процессами.

MPI\_Allreduce – контроль сходимости всех процессов.

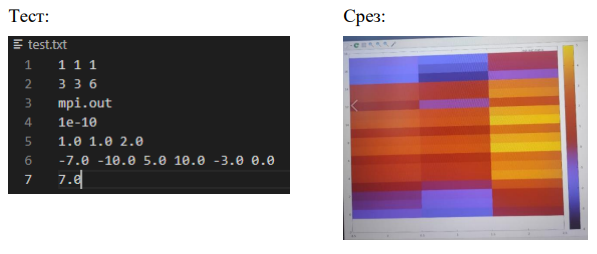
class Processor – класс для хранения всех функций и переменных для решения задачи.

int main() – создание экземпляра класса Processor, а также инициализация и завершение работы MPI.

# тесты производительности



# Визуализация



# Выводы

Выполнив лабораторную работу № 8, я узнал, как можно использовать возможности CUDA и MPI вместе, получая ещё большее распараллеливание задачи, а, следовательно, меньшее время работы. Однако правильное распределение ресурсов по всем процессам является довольно сложной задачей, и если получить ветвление, то производительность сильно уменьшится. В MPI также есть производные типы данных, которые позволяют распределять данные между процессами. Таким образом можно параллелить даже файловый вывод, который в подобного рода задачах обычно занимает существенное время.