Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет по второму заданию в рамках курса ”Суперкомпьютерное моделирование и технологии”

Выполнил:

студент 622 группы

Дремин М. В.

Вариант 8

Москва, 2023

Содержание

1. Математическая постановка задачи

Необходимо решить однородное волновое уравнение

$$\frac{\partial^2 u}{\partial t^2} = a^2 \Delta u, \quad \text{где} \quad a^2 = 1$$

С начальным условием

$$u|_{t=0} = \phi(x, y, z),$$
$$\left.\frac{\partial u}{\partial t}\right|_{t=0} = 0$$

И граничными условиями

$$u(0, y, z, t) = u(L_x, y, z, t),$$
$$u(x, 0, z, t) = u(x, L_y, z, t),$$
$$u(x, y, 0, t) = u(x, y, L_z, t),$$
$$u_x(0, y, z, t) = u_x(L_x, y, z, t),$$
$$u_y(x, 0, z, t) = u_y(x, L_y, z, t),$$
$$u_z(x, y, 0, t) = u_z(x, y, L_z, t).$$

2. Численный метод решения

Для решения данной задачи предлагается дискретизировать задачу на фиксированной решетке, а затем использовать метод конечных разностей. Пусть решетка имеет размерность (N_x, N_y, N_z, K). Тогда,

$$\omega_{ht} = \omega_h \times \omega_\tau$$

где

$$\omega_h = \{(x_i = ih_x, y_j = jh_y, z_k = kh_z), i, j, k = 0, 1, ..., N, h_{x,y,z} = \frac{L_{x,y,z}}{N}\}$$

$$\omega_\tau = \{(t_n = n\tau, n = 0, 1, ..., K, \tau = \frac{T}{K})\}$$

Далее будем считать, что

$$N_x = N_y = N_z = N, \quad K = 20$$

$$L_x = L_y = L_z = L, \quad T = 0.4$$

Применив на данной решетке метод конечных разностей получим

$$u_{ijk}^{n+1} = \tau^2 \cdot a^2 \Delta_h u_{ijk}^n + 2u_{ijk}^n - u_{ijk}^{n-1}$$

где Δ_h - семиточечный разностный аналог оператора Лапласа:

$$\Delta_h u_{ijk}^n = \frac{u_{i-1,jk}^n - 2u_{ijk}^n + u_{i+1,jk}^n}{h^2} + \frac{u_{ij-1,k}^n - 2u_{ijk}^n + u_{ij+1,k}^n}{h^2} + \frac{u_{ijk-1}^n - 2u_{ijk}^n + u_{ijk+1}^n}{h^2}$$

Граничные условия переходят в уравнения

$$u_{0jk}^{n+1} = u_{Njk}^{n+1}, \qquad u_{i0k}^{n+1} = u_{iNk}^{n+1}, \qquad u_{ij0}^{n+1} = u_{ijN}^{n+1},$$
$$u_{N+1,jk}^{n+1} = u_{1jk}^{n+1}, \quad u_{i,N+1,k}^{n+1} = u_{i,1k}^{n+1}, \quad u_{ij,N+1}^{n+1} = u_{ij,1}^{n+1},$$

$i, j, k = 0, 1, ..., N.$

Начальные условия переходят в уравнение

$$u_{ijk}^1 = u_{ijk}^0 + \frac{\tau}{2} \cdot a^2 \Delta_h \cdot u_{ijk}^0$$

Значения u_{ijk}^0 инициализируются исходя из значения аналитического решения в точке $t = 0$.

3. Аналитический метод решения

$$u_{analytical} = \sin\left(\frac{2\pi x}{L_x}\right) \cdot \sin\left(\frac{4\pi y}{L_y}\right) \cdot \sin\left(\frac{6\pi z}{L_z}\right) \cdot \cos(a_t \cdot t), \quad a_t = \pi\sqrt{\frac{4}{L_x^2} + \frac{16}{L_y^2} + \frac{36}{L_z^2}}$$

4. Программная реализация

4.1. OpenMP

Код парллельной реализации OpenMP лежит на github

Основные файлы:

- main.cpp - основная программа, реализует численный метод
- grid.hpp - заголовочный файл с реализацией класса Grid (сетка) и операций с ним
- function.hpp - заголовочный файл с реализацией класса Function (аналитической функции решения) и операций с ней

Вспомогательные файлы:

- run.py - запуск задач по сетке
- build.sh - билд программ для получения аналитического и численного решения
- analysis.ipynb - ноутбук для построения визуализаций и таблиц

Запуск решения:

Листинг 1: Пример запуска решения

```
bash build.sh
./main L grid_size
```

Параллелизация основого цикла расчета реализована с помощью

`#pragma omp for reduction(max:max_err),`

которая позволяет параллельно выполнить внешний цикл и взять операцию максимума глобально (по локальным переменным max_err в нитях).

4.2. OpenMP+MPI

Код парллельной реализации OpenMP+MPI лежит на github

Основные файлы:

- main_mpi.cpp - основная программа, реализует численный метод
- GridManager.hpp - заголовочный файл с реализацией класса GridManager, реализующий распределенные операции над сеткой
- GridSplitter.hpp - заголовочный файл с реализацией класса GridSplitter, реализующий разбиение стеки по блокам
- Function.hpp - заголовочный файл с реализацией класса Function (аналитической функции решения) и операций с ней

5. Результаты расчетов

L	Число OpenMP нитей	Число точек сетки	Погрешность	Время решения	Ускорение
1.0	1	128	0.000324	25.60350	1.000000
1.0	2	128	0.000324	13.18750	1.940668
1.0	4	128	0.000324	8.71851	2.936683
1.0	8	128	0.000324	6.71273	3.814171
1.0	16	128	0.000324	6.84504	3.740446
1.0	32	128	0.000324	5.02725	5.092943
1.0	1	256	0.000021	201.32200	1.000000
1.0	2	256	0.000021	103.22300	1.950360
1.0	4	256	0.000021	58.16300	3.461341
1.0	8	256	0.000021	56.08470	3.589606
1.0	16	256	0.000021	54.99850	3.660500
1.0	32	256	0.000021	47.76840	4.214544
3.0	1	128	0.000042	25.09600	1.000000
3.0	2	128	0.000042	12.78340	1.963171
3.0	4	128	0.000042	10.54620	2.379625
3.0	8	128	0.000042	6.73726	3.726956
3.0	16	128	0.000042	5.91059	4.245938
3.0	32	128	0.000042	4.57800	5.481870
3.0	1	256	0.000003	201.27100	1.000000
3.0	2	256	0.000003	102.87000	1.956552
3.0	4	256	0.000003	58.32650	3.450764
3.0	8	256	0.000003	56.40510	3.568312
3.0	16	256	0.000003	57.69250	3.488686
3.0	32	256	0.000003	47.93380	4.198937

Таблица 1: Результаты расчетов OpenMP

L	Число MPI процессов	Число OpenMP нитей	Число точек сетки	Погрешность	Время решения	Ускорение
1	2	1	128	0.000448599	5.75545	1
1	2	2	128	0.000448599	2.96663	1.94
1	2	4	128	0.000448599	1.61063	3.58
1	2	8	128	0.000448599	0.82054	7.02
1	4	1	128	0.000448599	3.41980	1
1	4	2	128	0.000448599	1.80352	1.90
1	4	4	128	0.000448599	0.97342	3.51
1	4	8	128	0.000448599	0.68773	4.97
1	8	1	128	0.000448599	1.97584	1
1	8	2	128	0.000448599	1.13005	1.75
1	8	4	128	0.000448599	0.75928	2.60
1	8	8	128	0.000448599	0.49524	3.99
3.14159	2	1	128	5.84158e-05	5.81249	1
3.14159	2	2	128	5.84158e-05	3.04218	1.91
3.14159	2	4	128	5.84158e-05	1.60388	3.63
3.14159	2	8	128	5.84158e-05	0.82317	7.06
3.14159	4	1	128	5.84158e-05	3.41059	1
3.14159	4	2	128	5.84158e-05	1.74750	1.95
3.14159	4	4	128	5.84158e-05	0.97346	3.50
3.14159	4	8	128	5.84158e-05	0.67167	5.08
3.14159	8	1	128	5.84158e-05	2.01240	1
3.14159	8	2	128	5.84158e-05	1.04601	1.92
3.14159	8	4	128	5.84158e-05	0.78965	2.55
3.14159	8	8	128	5.84158e-05	0.47161	4.27
1	2	1	256	6.41112e-05	46.16100	1
1	2	2	256	6.41112e-05	23.66490	1.95
1	2	4	256	6.41112e-05	12.53900	3.68
1	2	8	256	6.41112e-05	6.50073	7.10
1	4	1	256	6.41112e-05	26.20390	1
1	4	2	256	6.41112e-05	13.59940	1.93
1	4	4	256	6.41112e-05	7.26778	3.61
1	4	8	256	6.41112e-05	4.32837	6.06
1	8	1	256	6.41112e-05	15.33240	1
1	8	2	256	6.41112e-05	7.89385	1.94
1	8	4	256	6.41112e-05	4.26164	3.60
1	8	8	256	6.41112e-05	3.48889	4.39
3.14159	2	1	256	1.40487e-05	46.05600	1
3.14159	2	2	256	1.40487e-05	23.70740	1.94
3.14159	2	4	256	1.40487e-05	12.21780	3.77
3.14159	2	8	256	1.40487e-05	6.44914	7.14
3.14159	4	1	256	1.40487e-05	26.30580	1
3.14159	4	2	256	1.40487e-05	13.48620	1.95
3.14159	4	4	256	1.40487e-05	7.27542	3.62
3.14159	4	8	256	1.40487e-05	3.65402	7.20
3.14159	8	1	256	1.40487e-05	15.27110	1
3.14159	8	2	256	1.40487e-05	8.17174	1.87
3.14159	8	4	256	1.40487e-05	4.44492	3.44
3.14159	8	8	256	1.40487e-05	3.51408	4.35

Таблица 2: Результаты расчетов OpenMP+MPI

6. Визуализации

Анимированные визуализации можно найти на github

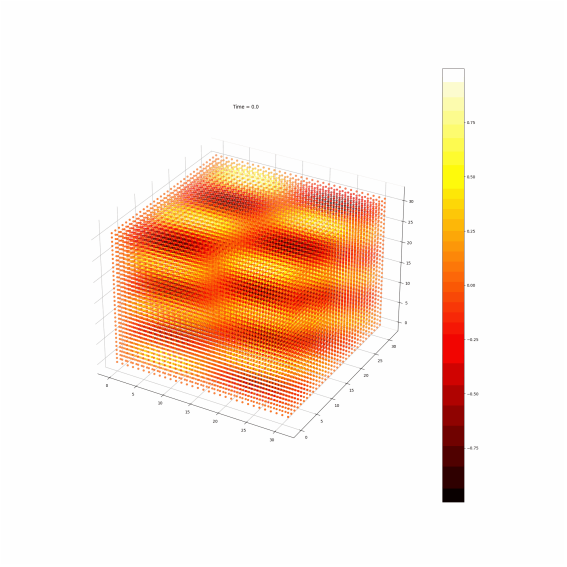

Рис. 1: Визуализация аналитического решения

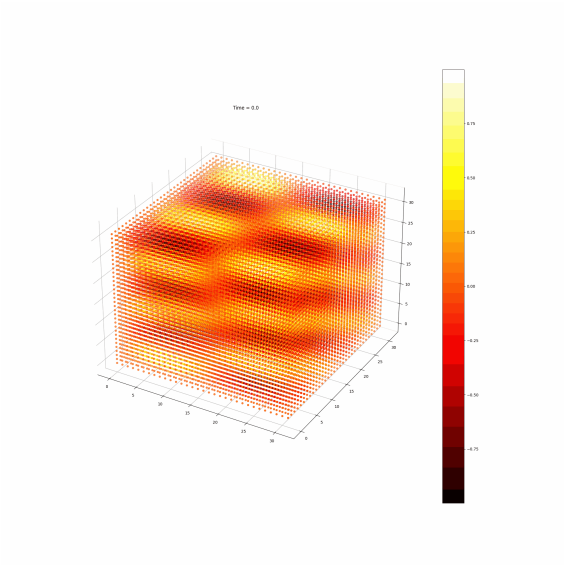

Рис. 2: Визуализация численного решения

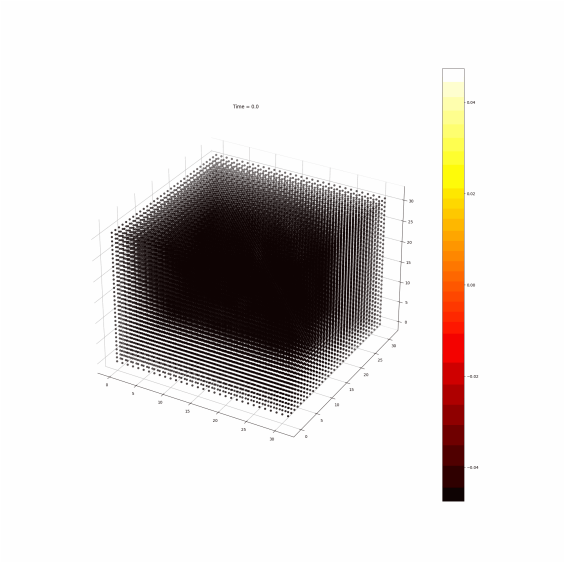

Рис. 3: Визуализация ошибки

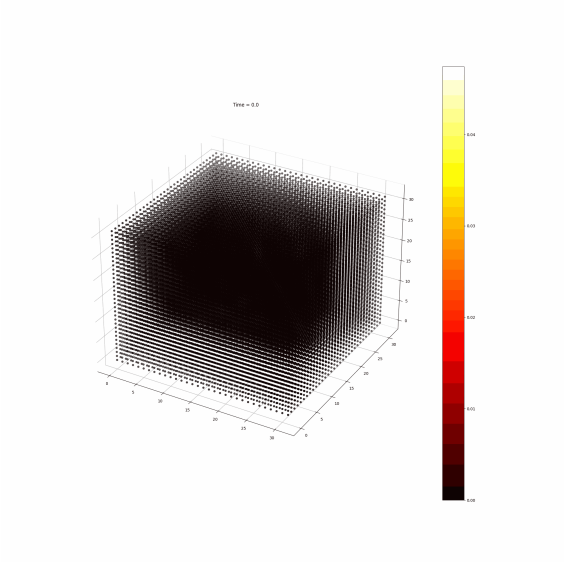

Рис. 4: Визуализация абсолютной ошибки

7. Выводы

Задача для трёхмерного гиперболического уравнения в прямоугольном параллелепипеде отлично подходит для распараллеливания. В результате получены программные средства, решающие поставленную задачу средствами OpenMP и гибридным способом – MPI+OpenMP.