Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

ОТЧЕТ

по курсу
“Суперкомпьютерное моделирование и технологии”

Вариант — 3

Работу выполнил:
студент 628 группы
Сазонов Георгий Владимирович

Математическая постановка задачи

В трёхмерной замкнутой области

$$\Omega = [0 \le x \le L_x] \times [0 \le y \le L_y] \times [0 \le z \le L_z]$$

для $0 < t \le T$ требуется найти решение $u(x, y, z, t)$ уравнения в частных производных

$$\frac{\partial^2 u}{\partial t^2} = a^2 \Delta u, \tag{1}$$

с начальными условиями

$$u|_{t=0} = \varphi(x, y, z), \tag{2}$$

$$\left.\frac{\partial u}{\partial t}\right|_{t=0} = 0, \tag{3}$$

при условии, что на границах области заданы либо однородные граничные условия первого рода

$$u(0, y, z, t) = 0, \qquad u(L_x, y, z, t) = 0, \tag{4}$$

$$u(x, 0, z, t) = 0, \qquad u(x, L_y, z, t) = 0, \tag{5}$$

$$u(x, y, 0, t) = 0, \qquad u(x, y, L_z, t) = 0, \tag{6}$$

либо периодические граничные условия

$$u(0, y, z, t) = u(L_x, y, z, t), \qquad u_x(0, y, z, t) = u_x(L_x, y, z, t), \tag{7}$$

$$u(x, 0, z, t) = u(x, L_y, z, t), \qquad u_y(x, 0, z, t) = u_y(x, L_y, z, t), \tag{8}$$

$$u(x, y, 0, t) = u(x, y, L_z, t), \qquad u_z(x, y, 0, t) = u_z(x, y, L_z, t). \tag{9}$$

Конкретная комбинация граничных условий определяется индивидуальным вариантом задания.

Для численного решения задачи введём на Ω сетку $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$, где

$$T = T_0, \quad L_x = L_{x0}, \quad L_y = L_{y0}, \quad L_z = L_{z0},$$

$$\bar{\omega}_h = \{(x_i = ih_x,\ y_j = jh_y,\ z_k = kh_z) \mid i, j, k = 0, 1, \ldots, N,\ h_x N = L_x,\ h_y N = L_y,\ h_z N = L_z\},$$

$$\omega_\tau = \{t_n = n\tau \mid n = 0, 1, \ldots, K,\ \tau K = T\}.$$

Через ω_h обозначим множество внутренних, а через γ_h — множество граничных узлов сетки $\bar{\omega}_h$.

Для аппроксимации исходного уравнения (1) с однородными граничными условиями (4)–(6) и начальными условиями (2)–(3) воспользуемся следующей системой уравнений:

$$\frac{u^{n+1}_{ijk} - 2u^n_{ijk} + u^{n-1}_{ijk}}{\tau^2} = a^2 \Delta_h u^n_{ijk}, \quad (x_i, y_j, z_k) \in \omega_h,\ n = 1, 2, \ldots, K-1,$$

где Δ_h — семиточечный разностный аналог оператора Лапласа:

$$\Delta_h u^n_{ijk} = \frac{u^n_{i-1,j,k} - 2u^n_{i,j,k} + u^n_{i+1,j,k}}{h_x^2} + \frac{u^n_{i,j-1,k} - 2u^n_{i,j,k} + u^n_{i,j+1,k}}{h_y^2} + \frac{u^n_{i,j,k-1} - 2u^n_{i,j,k} + u^n_{i,j,k+1}}{h_z^2}.$$

Приведённая выше разностная схема является явной — значения u^{n+1}_{ijk} на $(n+1)$-м временном слое можно явным образом выразить через значения на предыдущих слоях.

Для начала счёта (т. е. для нахождения u^1_{ijk}) должны быть заданы значения u^0_{ijk}, u^1_{ijk}, $(x_i, y_j, z_k) \in \omega_h$. Из условия (2) имеем

$$u^0_{ijk} = \varphi(x_i, y_j, z_k), \quad (x_i, y_j, z_k) \in \omega_h. \tag{10}$$

Простейшая замена начального условия (3) уравнением $(u^1_{ijk} - u^0_{ijk})/\tau = 0$ имеет лишь первый порядок аппроксимации по τ. Аппроксимацию второго порядка по τ и h даёт разностное уравнение

$$\frac{u^1_{ijk} - u^0_{ijk}}{\tau} = \frac{a^2\tau}{2}\Delta_h\varphi(x_i, y_j, z_k),$$

откуда

$$u^1_{ijk} = u^0_{ijk} + \frac{a^2\tau^2}{2}\Delta_h\varphi(x_i, y_j, z_k). \tag{11}$$

Разностная аппроксимация для периодических граничных условий выглядит следующим образом:

$$\begin{aligned}
&u^{n+1}_{0jk} = u^{n+1}_{Njk}, \quad u^{n+1}_{1jk} = u^{n+1}_{N+1,jk}, \\
&u^{n+1}_{i0k} = u^{n+1}_{iNk}, \quad u^{n+1}_{i1k} = u^{n+1}_{i,N+1,k}, \quad i,j,k = 0,1,\ldots,N. \\
&u^{n+1}_{ij0} = u^{n+1}_{ijN}, \quad u^{n+1}_{ij1} = u^{n+1}_{ij,N+1},
\end{aligned}$$

Для вычисления значений $u^0, u^1 \in \gamma_h$ допускается использование аналитического значения u, которое задаётся в программе также для вычисления погрешности решения задачи.

Специфика для Варианта №3

Граничные условия:

- по x: 1-го рода (нулевые)
- по y: периодические
- по z: 1-го рода (нулевые)

Аналитическое решение:

$$u_{\text{analytical}}(x, y, z, t) = \sin\left(\frac{\pi x}{L_x}\right) \cdot \sin\left(\frac{2\pi y}{L_y}\right) \cdot \sin\left(\frac{3\pi z}{L_z}\right) \cdot \cos(a_t t),$$

где

$$a_t = \frac{\pi}{2}\sqrt{\frac{1}{L_x^2} + \frac{4}{L_y^2} + \frac{9}{L_z^2}}, \quad a^2 = \frac{1}{4}.$$

Программная реализация

Реализация задачи состоит из файлов: `main.cpp`, `solution.cpp`, `equation.hpp` и `Makefile`, `run_polus.sh`.

Файл `main.cpp` содержит функцию `main`, которая обрабатывает входные параметры, инициализирует вычисления и сохраняет результаты.

Программа запускается следующим образом: mpirun -np NPROC ./wave3d_mpi N Lx Ly Lz

Здесь:

- N — количество ячеек по каждой пространственной координате (итоговый размер сетки составляет $(N+1)^3$);

- `NPROC` — число MPI-процессов;
- L_x, L_y, L_z — размеры расчётной области по осям. Каждый из параметров может быть задан либо числом (например, `1.0`), либо строкой `pi`, которая интерпретируется как π

Файл `equation.hpp` содержит объявления используемых типов данных, а также класс `Grid`, описывающий геометрию расчётной сетки. Он хранит значения L_x, L_y, L_z, шаги h_x, h_y, h_z, временной шаг $\tau = 10^{-4}$ и предоставляет метод `index(i,j,k)` для преобразования трёхмерных индексов в линейный адрес одномерного массива. Также в этом файле определена функция `u_analytical`, реализующая точное решение для варианта №3:

$$u(x, y, z, t) = \sin\left(\frac{\pi x}{L_x}\right) \cdot \sin\left(\frac{2\pi y}{L_y}\right) \cdot \sin\left(\frac{3\pi z}{L_z}\right) \cdot \cos(a_t t),$$

где

$$a_t = \frac{\pi}{2}\sqrt{\frac{1}{L_x^2} + \frac{4}{L_y^2} + \frac{9}{L_z^2}}, \quad a^2 = \frac{1}{4}.$$

Значение $a^2 = \frac{1}{4}$ также хранится в классе Grid.

Основной вычислительный алгоритм реализован в файле `solution_mpi.cpp` и основан на явной конечно-разностной схеме второго порядка точности (раздел 3 методических указаний). Вычисления распределены между MPI-процессами с использованием **блочного трёхмерного разбиения** области Ω, что минимизирует объём межпроцессорных коммуникаций по сравнению с ленточным разбиением (см. рис. 1 в задании).

- `laplace_operator` — вычисляет разностный аналог оператора Лапласа $\Delta_h u$ по семиточечному шаблону. Внутри каждого процесса используется локальная индексация с учётом *гало-слоёв* (дополнительных призрачных слоёв толщиной 1 узел с каждой стороны). Граничные условия реализуются не в операторе, а отдельно: для осей x и z заданы однородные условия первого рода (формулы (4), (6)), поэтому на внешних границах Ω значение u фиксируется как 0; для оси y — периодические условия (формула (8)), что обеспечивается автоматически декартовой топологией с `periods = {0, 1, 0}` и обменом гало-слоями.

- `exchange_halos` — выполняет асинхронный обмен гало-слоями с 6 соседями (3D-стенки) с использованием `MPI_Isend`/`MPI_Irecv` и `MPI_Waitall`. Для направлений x и z обмен происходит только между существующими соседями (на границах Ω сосед отсутствует, что соответствует условиям 1-го рода); для y — обмен происходит всегда, в том числе между крайними процессами вдоль этой оси, благодаря периодичности в декартовой топологии.

- `apply_boundary_conditions` — после обмена гало-слоями на процессах, находящихся на границах Ω по x или z, значения в гало-слоях принудительно устанавливаются в 0, что гарантирует выполнение условий (4) и (6).

- `init` — вычисляет начальные слои u^0 и u^1:

  $$u^0_{ijk} = \varphi(x_i, y_j, z_k), \quad u^1_{ijk} = u^0_{ijk} + \frac{a^2\tau^2}{2}\Delta_h\varphi(x_i, y_j, z_k),$$

  в соответствии с формулами (10) и (12). Для варианта №3 используется $a^2 = 1/4$, а φ берётся из аналитического решения. Граничные узлы инициализируются с учётом типа условий: $u = 0$ по x и z, периодическое продолжение по y.

- `run_algo` — выполняет $K - 1 = 19$ временных шагов (всего 20 слоёв, как требуется в задании), обновляя решение по схеме:

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

  На каждом шаге после вычисления внутренних точек производится обмен гало-слоями и наложение граничных условий.

- `solve_mpi` — управляет выполнением: выделяет память под три временных слоя с учётом гало, запускает `init` и `run_algo`, измеряет время с помощью `MPI_Wtime()`, собирает глобальную максимальную погрешность через `MPI_Allreduce(..., MPI_MAX, ...)` и формирует выходной вектор без гало-слоёв.

Распараллеливание реализовано с использованием технологии MPI. В дальнейшем предполагается реализация с гибридным использованием OpenMP + MPI. Все эксперименты проводились на суперкомпьютере IBM Polus с использованием утилиты `mpisubmit.pl`, что гарантирует корректную загрузку модулей Spectrum MPI и оптимальное распределение процессов по узлам.

Файл `Makefile` обеспечивает сборку программы под суперкомпьютер IBM Polus. Запуск вычислительных экспериментов осуществляется через внешний скрипт `run_polus.sh` и включает:

- размеры сетки $N = 128, 256, 512$;
- два набора геометрических параметров: $L_x = L_y = L_z = 1$ и $L_x = L_y = L_z = \pi$;
- число потоков: 1, 2, 4, 8, 16, 32.

Результаты расчётов

Таблица 1: Результаты расчётов для сетки с $L = 1$ (MPI)

Число MPI процессов	Число точек сетки по одной оси	Время решения, с	Ускорение	Погрешность
1	128	6,70	1	3,42e-4
2	128	3,75	1,79	3,42e-4
4	128	2,04	3,28	3,42e-4
8	128	1,20	5,58	3,42e-4
16	128	0,83	8,07	3,42e-4
32	128	0,51	13,14	3,42e-4
1	256	50,12	1	6,78e-4
2	256	26,09	1,92	6,78e-4
4	256	13,46	3,72	6,78e-4
8	256	7,43	6,75	6,78e-4
16	256	4,55	11,02	6,78e-4
32	256	2,74	18,29	6,78e-4
1	512	388,08	1	1,3e-3
2	512	200,34	1,94	1,3e-3
4	512	105,65	3,67	1,3e-3
8	512	55,95	6,94	1,3e-3
16	512	30,49	12,73	1,3e-3
32	512	19,48	19,92	1,3e-3

Таблица 2: Результаты расчётов для сетки с $L = \pi$ (MPI)

Число MPI процессов	Число точек сетки по одной оси	Время решения, с	Ускорение	Погрешность
1	128	6,67	1	3,48e-5
2	128	3,56	1,87	3,48e-5
4	128	2,07	3,22	3,48e-5
8	128	1,21	5,51	3,48e-5
16	128	0,73	9,14	3,48e-5
32	128	0,47	14,19	3,48e-5
1	256	49,96	1	6,96e-5
2	256	26,24	1,90	6,96e-5
4	256	13,54	3,69	6,96e-5
8	256	7,02	7,12	6,96e-5
16	256	4,70	10,63	6,96e-5
32	256	2,76	18,10	6,96e-5
1	512	386,91	1	1,38e-4
2	512	206,16	1,88	1,38e-4
4	512	116,17	3,33	1,38e-4
8	512	54,08	7,15	1,38e-4
16	512	37,95	10,20	1,38e-4
32	512	18,41	21,02	1,38e-4