Московский государственный университет
имени М.В.Ломоносова

ЗАДАНИЕ ПО КУРСУ

«Суперкомпьютерное моделирование и технологии»

Вариант: 4

Студент: Семеняк Г.А.
Группа: 628

Сентябрь 2025 - Декабрь 2025

Москва 2025

[1] Код решения лежит на Гитхабе: https://github.com/twist13227/sctm

Содержание

1 Математическая постановка дифференциальной задачи

В трехмерной 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 однородные граничные условия первого рода

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

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

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

либо периодические граничные условия

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

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

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

Конкретная комбинация граничных условий определяется индивидуальным вариантом задания (см. п. 5).

2 Численный метод решения задачи

Содержание данного пункта основано на материале книги [2]. Для численного решения задачи введем на Ω сетку $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$, где

$$T = T_0,$$

$$L_x = L_{x_0},\ L_y = L_{y_0},\ L_z = L_{z_0}$$

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

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

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:

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

Здесь Δ_h — семиточечный разностный аналог оператора Лапласа:

$$\Delta_h u^n = \frac{u_{i-1,j,k}^n - 2u_{i,j,k}^n + u_{i+1,j,k}^n}{h^2} + \frac{u_{i,j-1,k}^n - 2u_{i,j,k}^n + u_{i,j+1,k}^n}{h^2} + \frac{u_{i,j,k-1}^n - 2u_{i,j,k}^n + u_{i,j,k+1}^n}{h^2}.$$

Приведенная выше разностная схема является явной — значения u_{ijk}^{n+1} на $(n+1)$-м шаге можно явным образом выразить через значения на предыдущих слоях.

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

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

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

$$\frac{u_{ijk}^1 - u_{ijk}^0}{\tau} = a^2 \frac{\Delta_h \varphi(x_i, y_j, z_k)}{2}, \quad (x_i, y_j, z_k) \in \omega_h. \tag{11}$$

$$u_{ijk}^1 = u_{ijk}^0 + a^2 \frac{\Delta_h \varphi(x_i, y_j, z_k)}{2}. \tag{12}$$

Разностная аппроксимация для периодических граничных условий выглядит следующим образом

$$u^{n+1}_{0jk} = u^{n+1}_{Njk}, \quad u^{n+1}_{1jk} = u^{n+1}_{N+1jk},$$
$$u^{n+1}_{i0k} = u^{n+1}_{iNk}, \quad u^{n+1}_{i1k} = u^{n+1}_{iN+1k},$$
$$u^{n+1}_{ij0} = u^{n+1}_{ijN}, \quad u^{n+1}_{ij1} = u^{n+1}_{ijN+1},$$

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

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

Вычисления далее проводятся для следующей аналитической функции:

$$u(x, y, z, t) = \sin\left(\frac{3\pi}{L_x}x\right) \cdot \sin\left(\frac{2\pi}{L_y}y\right) \cdot \sin\left(\frac{2\pi}{L_z}z\right) \cdot \cos(a_t\, t + 4\pi),$$

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

Со следующими граничными условиями:

$$u(0, y, z, t) = 0, \quad u(L_x, y, z, t) = 0, \tag{4}$$
$$u(x, 0, z, t) = u(x, Ly, z, t), \quad u(x, L_y, z, t) = u(x, Ly, z, t), \tag{5}$$
$$u(x, y, 0, t) = u(x, y, Lz, t), \quad u(x, y, 0, t) = u(x, y, Lz, t), \tag{6}$$

3 Программная реализация (MPI+OpenMP)

Гибридная реализация использует двухуровневый параллелизм: распределение данных между узлами кластера через MPI и параллельные вычисления на многоядерных процессорах узла через OpenMP. Трёхмерная декомпозиция области сохраняется (формируемая `MPI_Dims_create`), но каждый MPI-процесс дополнительно использует несколько OpenMP-потоков для обработки своей локальной подобласти. Это позволяет эффективно использовать современные вычислительные архитектуры с иерархией памяти и десятками ядер на узел.

Граничные условия и структура декомпозиции полностью соответствуют MPI-версии:

- По оси x: условия Дирихле (u=0 на внешних границах x=0 и x=L)
- По осям y и z: периодические условия
- Обмен ghost-слоями выполняется только через MPI (в основном потоке)
- Все OpenMP-потоки работают с уже синхронизированными данными

Параллельная обработка данных

Вычислительно-ёмкие участки кода распараллеливаются с помощью OpenMP:

- Инициализация начальных условий и аналитического решения
- Вычисление дискретного лапласиана на внутренних узлах
- Обновление решения по временной схеме
- Заполнение локальных ghost-слоёв при отсутствии соседей по декомпозиции
- Подсчёт локальной максимальной ошибки

Для вложенных циклов используется директива `collapse(3)`, объединяющая три уровня вложенности в единое пространство итераций:

```
#pragma omp parallel for collapse(3)
for (int i = 1; i <= local_ni; ++i) {
    for (int j = 1; j <= local_nj; ++j) {
        for (int k = 1; k <= local_nk; ++k) {
        }
    }
}
```

Это обеспечивает равномерное распределение нагрузки даже при небольших размерах локальных блоков.

Оптимизация локальных операций

Для заполнения ghost-слоёв в случае отсутствия декомпозиции по некоторой оси (dims[i]=1) используются параллельные циклы:

```
if (dims[1] == 1) {
    #pragma omp parallel for collapse(2)
    for (int i = 0; i < local_ni + 2; ++i) {
        for (int k = 0; k < local_nk + 2; ++k) {
            u[local_index(i, 0, k)] = u[local_index(i, local_nj, k)];
            u[local_index(i, local_nj + 1, k)] = u[local_index(i, 1, k)];
        }
    }
}
```

Такой подход эффективно использует внутренний параллелизм узла для операций, которые в чистой MPI-версии выполнялись бы последовательно.

Синхронизация между уровнями параллелизма

Ключевой принцип гибридной реализации — строгое разделение коммуникаций и вычислений:

1. Все MPI-коммуникации (`MPI_Isend`, `MPI_Irecv`, `MPI_Waitall`) выполняются в последовательной части кода до входа в OpenMP-регионы

2. OpenMP-параллелизм активируется только после полной синхронизации данных через MPI

3. Для агрегации максимальной ошибки используется комбинация локальной OpenMP-редукции и глобального `MPI_Reduce`:

```
double local_error = 0.0;
#pragma omp parallel for collapse(3) reduction(max:local_error)
for (int i = 1; i <= local_ni; ++i) {
}
MPI_Reduce(&local_error, &global_error, 1, MPI_DOUBLE, MPI_MAX, 0, cart_comm);
```

Это исключает гонки данных и гарантирует корректность результатов.

Управление ресурсами

Перед началом вычислений настраивается среда OpenMP:

```
omp_set_dynamic(0);
omp_set_num_threads(omp_get_max_threads());
```

Отключение динамического распределения потоков (`omp_set_dynamic(0)`) обеспечивает предсказуемую производительность, а установка максимального числа потоков позволяет полностью использовать вычислительные ресурсы узла.

Таким образом, гибридная MPI+OpenMP реализация обеспечивает:

- Снижение накладных расходов на коммуникации за счёт уменьшения числа MPI-процессов
- Эффективное использование кэш-памяти за счёт локальности данных в рамках одного узла
- Масштабируемость как на уровне кластера (MPI), так и на уровне многоядерного процессора (OpenMP)
- Гибкость в настройке соотношения MPI-процессов и OpenMP-потоков под конкретную архитектуру

Ниже приведены усреднённые по пяти запускам значения времени, погрешности и ускорения для гибридной реализации.

3.1 Результаты MPI+OpenMP ($L = 1$)

3.2 Результаты MPI+OpenMP ($L = \pi$)

Таблица 1: Результаты MPI+OpenMP, $L = 1$

MPI	OMP	N^3	Время T	Погрешность δ	Var(T)
4	1	128^3	0.863	7.610e-03	0.00678
4	2	128^3	0.490	7.610e-03	0.00476
4	4	128^3	0.410	7.610e-03	0.00826
4	8	128^3	0.239	7.610e-03	0.000736
8	1	256^3	3.466	3.808e-03	0.121
8	2	256^3	2.441	3.808e-03	0.00369
8	4	256^3	1.767	3.808e-03	0.00675
8	8	256^3	1.240	3.808e-03	0.00262

Таблица 2: Результаты MPI+OpenMP, $L = \pi$

MPI	OMP	N^3	Время T	Погрешность δ	Var(T)
4	1	128^3	0.945	8.325e-04	0.00151
4	2	128^3	0.652	8.325e-04	0.00661
4	4	128^3	0.333	8.325e-04	0.00189
4	8	128^3	0.293	8.325e-04	0.00197
8	1	256^3	3.620	4.164e-04	0.01585
8	2	256^3	2.452	4.164e-04	0.00733
8	4	256^3	1.682	4.164e-04	0.00102
8	8	256^3	1.227	4.164e-04	0.000182