

Отчёт по курсу

«Суперкомпьютерное моделирование и

технологии»

Задание 1

многопоточная реализация явной схемы

второго порядка для 3D уравнения

*Выполнил: Ши Хуэй,
Группа 638(shihuicollapsor@gmail.com)
Дата подачи: 19.10.2025*

Содержание

| | |
|--------------------------------------|---|
| Содержание | 1 |
| 1. Постановка задачи | 2 |
| 2. Формат командной строки | 2 |
| 3. Спецификация системы | 2 |
| 4. Описание алгоритмов | 3 |
| 5. Записи экспериментов и результаты | 6 |
| 6. Заключение | 8 |

1. Постановка задачи

В качестве модельной задачи предлагается задача для трёхмерного гиперболического уравнения в области, представляющей из себя прямоугольный параллелепипед. Индивидуальные варианты заданий отличаются типом граничных условий. Данное уравнение часто применяется в теории тепло- и массопереноса, гидро- и аэромеханике, электростатике. Поэтому поиск решения данной задачи в различных областях является весьма актуальным. Задание необходимо выполнить на следующих ПВС Московского университета: IBM Polus [1].

2. Формат командной строки

Программа принимает параметры через командную строку в следующем формате:

```
xlC++_r -std=c++11 -O3 -qarch=pwr8 -qtune=pwr8 -qsmp=omp -qhot -qipa |  
-o Wave 3dWaveSlover.cpp  
./Wave N K Threads
```

где,

- N — размер сетки по всем осям (принимаем $N_x = N_y = N_z = N$).
- K — число временных шагов (шаг $\tau = T/K$).
- $Threads$ — число потоков OpenMP, используемых при выполнении.

3. Спецификация системы

Тестирование и измерение производительности проводилось на двух вычислительных системах на персональном ноутбуке (MacBook Pro) и на кластере Polus Московского государственного университета.

3.1 Локальная система (ноутбук MacBook Pro M3 Max)

- Процессор: Apple M3 Max (14 ядер: 10 производительных и 4 эффективных)
 - CPU: 14 ядер (10 производительных + 4 энергоэффективных)
 - GPU: 30 графических ядер (интегрированных)
 - Neural Engine: 16-ядерный нейронный процессор
 - 300GB/s memory bandwidth
- Архитектура: ARM64 (Apple Silicon)
- Операционная система: macOS Sonoma 14.x
- Оперативная память: 36 ГБ (унифицированная память UMA)
- Компилятор: clang++ (Apple LLVM 17) с опциями `-std=c++17 -O3 -fopenmp`

3.2 Кластер Polus (МГУ, ВМК)

- Процессоры в узлах: $2 \times$ IBM Power8NVL (по 10 ядер, 8 потоков на ядро, частота до 4.02 ГГц)
- Пиковая производительность CPU: ≈ 580 GFLOPS (двойная точность)

- Общая оперативная память узла: 256 ГБ DDR4-2400 (128 ГБ на сокет)
- Память: 16 каналов на узел, пиковая пропускная способность $\approx 307.2 \text{ GB/s}$
- Графические ускорители: 2 × NVIDIA Tesla P100 (пиковая производительность $\approx 9.4 \text{ TFLOPS}$, не использовались в данной работе)
- Сеть: Infiniband / 100 Gb/s
- Операционная система: Red Hat Enterprise Linux 7.5
- Компилятор: `g++ -std=c++11 -O3 -fopenmp`

4. Описание алгоритмов

Согласно Самарскому А.А. и Гулину А.В. (Численные методы. М.: Наука, 1989, с. 285), трёхслойная явная разностная схема для уравнения колебаний устойчива при $\tau \leq h$ (при $a = 1$); в общем случае условие записывается как $a^* \tau \leq h$.

Рассмотрим волновое уравнение

$$u_{tt} = a^2 \Delta u$$

и трёхслойную явную схему

$$\frac{u^{n+1}_{i,j,k} - 2u^n_{i,j,k} + u^{n-1}_{i,j,k}}{\tau^2} = a^2 \Delta_h u^n_{i,j,k},$$

где

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

1) 1D-вывод

В одномерном случае

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\tau^2} = a^2 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2}, \quad \sigma = \frac{a\tau}{h}.$$

Ищем решение в виде $u_j^n = q^n e^{ijh\phi}$, Подстановка даёт квадратурное уравнение на множитель роста q :

$$q^2 - 2 \left(1 - 2\sigma^2 \sin^2 \left(\frac{h\phi}{2} \right) \right) q + 1 = 0.$$

Требование устойчивости по фон Нейману: $|q| \leq 1$ для всех $\theta = h\phi$.

Отсюда следует

$$\left(1 - 2\sigma^2 \sin^2 \frac{\theta}{2} \right)^2 \leq 1 \iff \sigma^2 \sin^2 \frac{\theta}{2} \leq 1.$$

Так как $\sin^2 \frac{\theta}{2} \leq 1$, получаем условие устойчивости: $\sigma = \frac{a\tau}{h} \leq 1$

2) Обобщение на 3D-решётку

Для прямоугольной сетки собственное значение дискретного Лапласса равно

$$\lambda(\theta_x, \theta_y, \theta_z) = -4 \left(\frac{\sin^2(\theta_x/2)}{h_x^2} + \frac{\sin^2(\theta_y/2)}{h_y^2} + \frac{\sin^2(\theta_z/2)}{h_z^2} \right).$$

Повторяя рассуждение и требуя $|q| \leq 1$ для всех θ , получаем достаточное (и в равномерном случае — острое) условие:

$$(a\tau)^2 \left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} \right) \leq 1$$

В коде это проверяется по величине $cfl2 = a2 * (tau*tau) * (1/(hx*hx) + 1/(hy*hy) + 1/(hz*hz))$ и схема считается устойчивой при $cfl2 \leq 1$. Здесь $a2 = a^2$, $\backslash au=T/K$.

1. точное аналитическое решение $U_{analytical}$ в узлах сетки

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

$$a_t = \sqrt{\left(\frac{\pi}{L_x}\right)^2 + \left(\frac{\pi}{L_y}\right)^2 + \left(\frac{2\pi}{L_z}\right)^2}.$$

```
auto askUanalytical = [&](int i, int j, int k, double t) -> double
{
    const double x = i * hx;
    const double y = j * hy;
    const double z = k * hz;
    return sin(PI * x / Lx) * sin(PI * y / Ly) * sin(2 * PI * z / Lz) * cos(at * t + 2 * PI);
};
```

2. Проводим разбиение области Ω между процессами

$$\Omega = [0, L_x] \times [0, L_y] \times [0, L_z].$$

3. Фиксируем временной слой (начиная с 0).

Время: $t_0 = 0$, уровни $n = 0, 1, \dots, K$ с шагом $\tau = \frac{T}{K}$.

4. Используя формулы $u_{ijk}^0 = \varphi(x_i, y_j, z_k)$ и $u_{ijk}^1 = u_{ijk}^0 + a^2 \frac{\tau^2}{2} \Delta_h \varphi(x_i, y_j, z_k)$, при

граничных условиях: по x, y — Дирихле $u = 0$ на $i = 0, N_x$ и $j = 0, N_y$, по z — периодичность $u_{i,j,0} = u_{i,j,N_z}$. Оператор Δ_h аппроксимирован центральными разностями; по z берутся периодические соседи (обёртка индексов). $n \geq 1$

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

5. Обновление u_{ijk}^{n+1}

$$\text{Для } n \geq 1 \text{ используем } u_{ijk}^{n+1} = 2u_{ijk}^n - u_{ijk}^{n-1} \bullet a^2 \tau^2 \Delta_h u_{ijk}^n,$$

где дискретный лапласиан задаётся центральными разностями

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

Границные условия:

по x,y — Дирихле нулевого уровня: $u_{0,j,k}^{n+1} = u_{N_x,j,k}^{n+1} = u_{i,0,k}^{n+1} = u_{i,N_y,k}^{n+1} = 0$;

по z — периодичность: $u_{i,j,0}^n = u_{i,j,N_z}^n$, $k_- = \begin{cases} N_z, & k = 0 \\ k - 1, & \text{иначе} \end{cases}$, $k_+ = \begin{cases} 0, & k = N_z \\ k + 1, & \text{иначе} \end{cases}$.

```
for (int t = 1; t < k; t++) {
```

```
#pragma omp parallel for collapse(3) schedule(static)
```

```
for (int i = 0; i < NX; i++) {
```

```
    for (int j = 0; j < NY; j++) {
```

```
        for (int k = 0; k < NZ; k++) {
```

```
            const size_t id = idx(i, j, k, Ny, Nz);
```

```
            if (i == 0 || i == Nx || j == 0 || j == Ny) {
```

```
                u_next[id] = 0;
```

```
            } else {
```

```
                int k1, k2;
```

```
                if (k == 0) {
```

```
                    k1 = p.Nz - 1;
```

```
                    k2 = k + 1;}
```

```
                else if (k == p.Nz) {
```

```
                    k1 = k - 1;
```

```
                    k2 = 1;}
```

```
                else {
```

```
                    k1 = k - 1;
```

```
                    k2 = k + 1;}
```

```
                const double ijk = u_curr[id];
```

```
                const double i1jk = u_curr[idx(i - 1, j, k, Ny, Nz)];
```

```
                const double i2jk = u_curr[idx(i + 1, j, k, Ny, Nz)];
```

```
                const double ij1k = u_curr[idx(i, j - 1, k, Ny, Nz)];
```

```
                const double ij2k = u_curr[idx(i, j + 1, k, Ny, Nz)];
```

```
                const double ijk1 = u_curr[idx(i, j, k1, Ny, Nz)];
```

```
                const double ijk2 = u_curr[idx(i, j, k2, Ny, Nz)];
```

```
                const double phi = ((i1jk - 2 * ijk + i2jk) / (hx * hx) + (ij1k - 2 * ijk + ij2k) / (hy * hy) + (ijk1 - 2 * ijk + ijk2) / (hz * hz));
```

```
                u_next[id] = 2 * u_curr[id] - u_prev[id] + a2 * tau * tau * phi;
```

```
            }}}
```

```
        swap(u_prev, u_curr);
```

```
        swap(u_curr, u_next); }
```

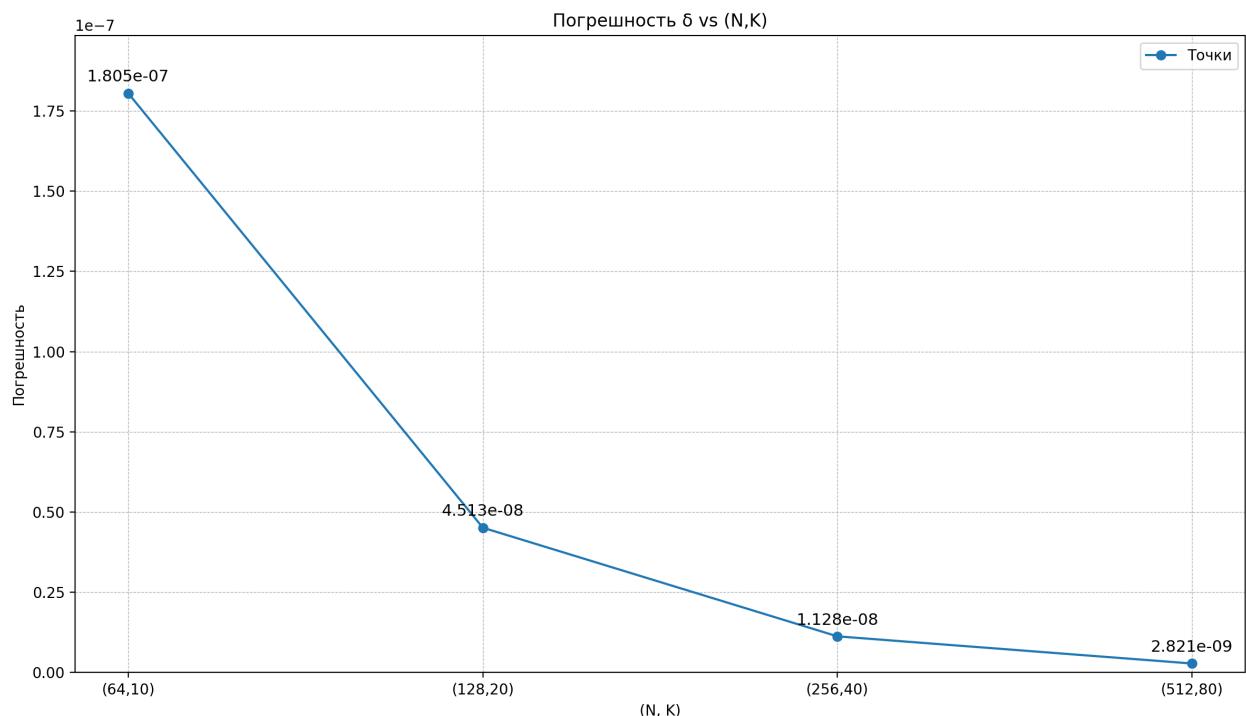
6. Сравнить численное решение с аналитическим на момент $t=T$ и вывести погрешность в норме

$$\delta_{\max} = \max_{i,j,k} \left| u^{\text{num}}_{ijk}(T) - u^{\text{analyt}}_{ijk}(T) \right|.$$

5. Записи экспериментов и результаты

Проводим последовательное измельчение сетки $N_x = N_y = N_z$: $64 \rightarrow 128 \rightarrow 256 \rightarrow 512$ и одновременно удваиваем число шагов K : $10 \rightarrow 20 \rightarrow 40 \rightarrow 80$ (то есть h_x, h_y, h_z и τ уменьшаются вдвое при неизменном CFL).

| N | K | τ | δ_{\max} |
|-----|-----|----------|-----------------|
| 64 | 10 | 1.0E-03 | 1.804968E-07 |
| 128 | 20 | 5.0E-04 | 4.513413E-08 |
| 256 | 40 | 2.5E-04 | 1.128415E-08 |
| 512 | 80 | 1.25E-04 | 2.821075E-09 |



| Число OpenMP нитей N_p | Число точек сетки N^3 | Временной шаг K | Время решения T | Ускорение S | Погрешность δ |
|--------------------------|-------------------------|-------------------|-------------------|---------------|----------------------|
| 1 | 128^3 | 200 | 5.3927 | 1.00 | 4.524370E-08 |
| 2 | 128^3 | 200 | 3.1883 | 1.35 | 4.524370E-08 |
| 4 | 128^3 | 200 | 1.6564 | 2.60 | 4.524370E-08 |
| 6 | 128^3 | 200 | 1.0554 | 4.08 | 4.524370E-08 |

| | | | | | |
|----|---------|-----|---------|-------|--------------|
| 8 | 128^3 | 200 | 0.8656 | 4.98 | 4.524370E-08 |
| 12 | 128^3 | 200 | 0.6029 | 7.15 | 4.524370E-08 |
| 16 | 128^3 | 200 | 0.4852 | 8.88 | 4.524370E-08 |
| 32 | 128^3 | 200 | 0.3669 | 11.79 | 4.524370E-08 |
| 1 | 256^3 | 100 | 14.9999 | 1.00 | 1.130146E-08 |
| 2 | 256^3 | 100 | 8.4123 | 1.78 | 1.130146E-08 |
| 4 | 256^3 | 100 | 4.5121 | 3.32 | 1.130146E-08 |
| 6 | 256^3 | 100 | 3.1768 | 4.72 | 1.130146E-08 |
| 8 | 256^3 | 100 | 2.3660 | 6.34 | 1.130146E-08 |
| 12 | 256^3 | 100 | 1.6975 | 8.84 | 1.130146E-08 |
| 16 | 256^3 | 100 | 1.3644 | 10.99 | 1.130146E-08 |
| 32 | 256^3 | 100 | 1.2351 | 12.68 | 1.130146E-08 |
| 1 | 512^3 | 50 | 59.7022 | 1.00 | 2.818627E-09 |
| 2 | 512^3 | 50 | 32.9394 | 1.81 | 2.818627E-09 |
| 4 | 512^3 | 50 | 17.0780 | 3.49 | 2.818627E-09 |
| 6 | 512^3 | 50 | 11.5020 | 5.19 | 2.818627E-09 |
| 8 | 512^3 | 50 | 9.0251 | 6.61 | 2.818627E-09 |
| 12 | 512^3 | 50 | 6.4151 | 9.31 | 2.818627E-09 |
| 16 | 512^3 | 50 | 5.1447 | 11.65 | 2.818627E-09 |
| 32 | 512^3 | 50 | 3.7617 | 15.92 | 2.818627E-09 |

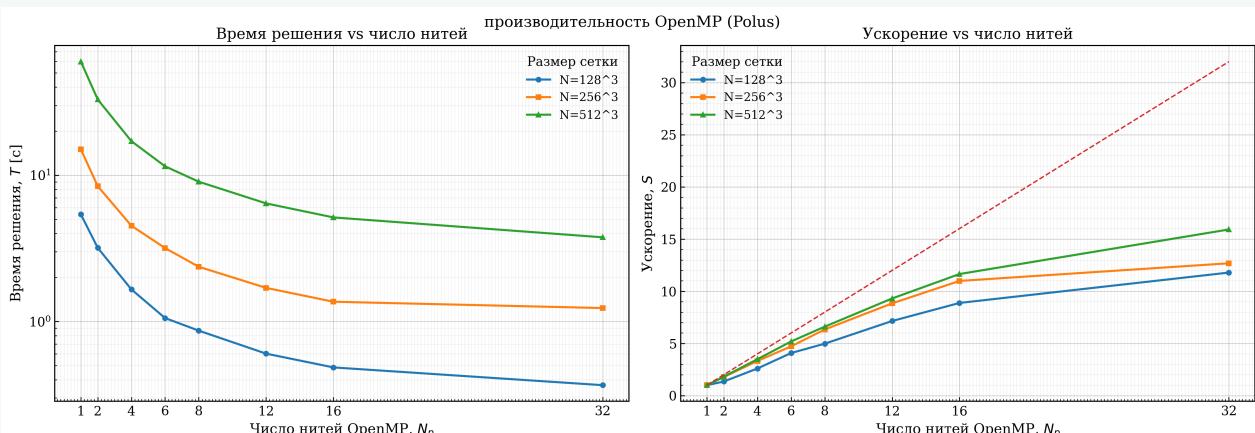


Рисунок Масштабирование OpenMP на кластере Polus.

Слева показано время решения Т в зависимости от числа нитей N_p (логарифмическая шкала по оси Y); справа — ускорение S при тех же параметрах.

6. Заключение

Реализован параллельный OpenMP-вариант решателя для трёхмерного волнового уравнения