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

Егоров Илья Георгиевич
группа 627

Суперкомпьютерное моделирование и технологии

Многопоточная реализация с использованием OpenMP решения гиперболического уравнения в частных производных на трехмерном прямоугольном параллелепипеде

ОТЧЁТ

Москва, 2025, 23 Октября

Содержание

1 Описание задания и программной реализации

1.1 Постановка задачи

В трёхмерной замкнутой области $\Omega = [0, L_x] \times [0, L_y] \times [0, L_z]$ для $t \in [0, T]$ требуется найти решение $u(x, y, z, t)$ задачи

$$\begin{cases} u_{tt}(x, y, z, t) = a^2 \Delta u(x, y, z, t) \\ u(x, y, z, 0) = \phi(x, y, z) \\ u_t(x, y, z, 0) = 0 \\ u(0, y, z, t) = u(L_x, y, z, t) = 0 \\ u(x, 0, z, t) = u(x, L_y, z, t) \\ u(x, y, 0, t) = u(x, y, L_z, t) = 0 \end{cases} \tag{1}$$

где $u(x, y, z, t) = sin\frac{\pi x}{L_x} sin\frac{2\pi y}{L_y} sin\frac{3\pi z}{L_z} cos(a_t t)$, $a_t = \frac{\pi}{2}\sqrt{\frac{1}{L_x} + \frac{4}{L_y} + \frac{9}{L_z}}$ — точное аналитическое решение, $\phi(x, y, z) = u(x, y, z, 0) = sin\frac{\pi x}{L_x} sin\frac{2\pi y}{L_y} sin\frac{3\pi z}{L_z}$ и $a^2 = \frac{1}{4}$.

1.2 Численный метод решения задачи

Пусть заданы $N, K \in \mathbb{N}$, тогда пусть $h_x = \frac{L_x}{N}$, $h_y = \frac{L_y}{N}$, $h_z = \frac{L_z}{N}$, $\tau = \frac{T}{K}$ и $\omega_{h\tau} = \overline{\omega_h} \times \omega_\tau$, где

$\overline{\omega_h} = \{(x_i, y_j, z_k) | x_i = ih_x, y_j = jh_y, z_k = kh_z, i, j, k \in \overline{0, N}\}$

$\omega_\tau = \{t = n\tau, n \in \overline{0, K}\}$

Пусть $\omega_h = int\overline{\omega_h}$, т.е. внутренние узлы сетки.

Пусть $u^n_{ijk} = u(x_i, y_j, z_k, t_n), \phi_{ijk} = \phi(x_i, y_j, z_k)$

Тогда

$$\begin{aligned} &u^n_{0jk} = u^n_{Njk} = u^n_{ij0} = u^n_{ijN} = 0, && i, j, k \in \overline{0, N}, n \in \overline{0, K} \\ &u^n_{i0k} = u^n_{iNk}, u^n_{i1k} = u^n_{iN+1k}, && i, k \in \overline{0, N}, n \in \overline{0, K} \\ &u^0_{ijk} = \phi_{ijk} = \phi_{ijk} = \phi(x_i, y_j, z_k), && (x_i, y_j, z_k) \in \omega_h \\ &u^1_{ijk} = u^0_{ijk} + \frac{a^2\tau^2}{2}\Delta_h \phi_{ijk} && (x_i, y_j, z_k) \in \omega_h \\ &u^{n+1}_{ijk} = 2u^n_{ijk} - u^{n-1}_{ijk} + \Delta_h u^n_{ijk} && (x_i, y_j, z_k) \in \omega_h \end{aligned}$$

Здесь Δ_h — семиточечный разностный аналог оператора Лапласа:

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

Будем рассматривать случай, когда $L_x = L_y = L_z = L$ и, соответственно, $h_x = h_y = h_z = h$. Для устойчивости решения требуется

$$\tau < h$$

Или же

$$\gamma = \frac{\tau}{h} < 1$$

Тогда при заданных L, N, K можно положить $T = \tau K = \gamma h K = \frac{\gamma L K}{N}$.

1.3 Программная реализация

Программа реализована на языке C++ (c++11/c++20). Для записи решения написан шаблонный класс *Grid4D*, который абстрагирует взаимодействие с памятью и индексированием. Также написаны функциональные классы *AnalyticalFunction* для инкапсуляции вычисления значения аналитической функции, *AnalyticalFunctionGrid* — обёртка, делающая то же самое, но в терминах узлов сетки, и *InitialFunctionGrid* — для вычисления начальных условий. Для непосредственного решения написан класс *Solver* с методом *solve*, возвращающий решение, — объект класса *Grid4D*, значение ошибки/-погрешности и времени, затраченного на вычисление решения и ошибки.

В качестве аргументов программа получат на вход N и T — количество потоков OpenMP. Значение K согласно условиям равно 20, а L —1 и π.

2 Исследование производительности

2.1 Характеристики вычислительной системы

Имя: ПВС «IBM Polus»
Пиковая производительность: 55.84 TFlop/s
Производительность (Linpack): 40.39 TFlop/s
Вычислительных узлов: 5
На каждом узле:
Процессоры IBM Power 8: 2
NVIDIA Tesla P100: 2
Число процессорных ядер: 20
Число потоков на ядро: 8
Оперативная память: 256 Гбайт (1024 Гбайт узел 5)
Коммуникационная сеть: Infiniband / 100 Gb
Система хранения данных: GPFS
Операционная система: Linux Red Hat 7.5

2.1.1 Параллельное ускорение

Дополнительные используемые опции компиляции: *-std=c++11 -O3 -fopenmp*.

На табл. 1 и 2 и рис. 1 и 2 хорошо видно наличие параллельного ускорения. При увеличении числа потоков до 8 включительно виден линейный характер уменьшения затраченного времени, при больших значениях рост уменьшения времени затухает, что логично с учётом того, что каждое ядро "Полюса"имеет ровно 8 потоков.

Thread Number	Node Number per Side	Solve Time (sec)	Acceleration	Error
1	128	7.768	1.000	3.13E-05
2	128	3.873	2.006	3.13E-05
4	128	1.987	3.909	3.13E-05
8	128	1.058	7.339	3.13E-05
16	128	0.716	10.851	3.13E-05
32	128	0.615	12.638	3.13E-05
1	256	59.446	1.000	2.01E-06
2	256	30.056	1.978	2.01E-06
4	256	15.187	3.914	2.01E-06
8	256	7.928	7.498	2.01E-06
16	256	5.298	11.221	2.01E-06
32	256	4.587	12.958	2.01E-06

Таблица 1: Зависимость времени решения от числа нитей процесса для $L = 1$

Thread Number	Node Number per Side	Solve Time (sec)	Acceleration	Error
1	128	7.770	1.000	3.13E-05
2	128	3.877	2.004	3.13E-05
4	128	1.990	3.905	3.13E-05
8	128	1.060	7.332	3.13E-05
16	128	0.717	10.843	3.13E-05
32	128	0.620	12.533	3.13E-05
1	256	58.713	1.000	2.01E-06
2	256	30.539	1.923	2.01E-06
4	256	15.204	3.862	2.01E-06
8	256	7.890	7.442	2.01E-06
16	256	5.041	11.647	2.01E-06
32	256	3.802	15.442	2.01E-06

Таблица 2: Зависимость времени решения от числа нитей процесса для $L = \pi$

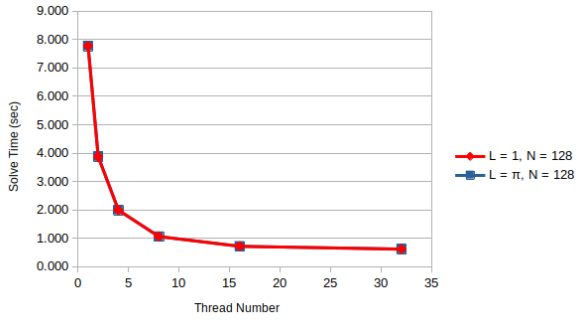

Рис. 1: Зависимость времени решения от числа нитей процесса для $N = 128$

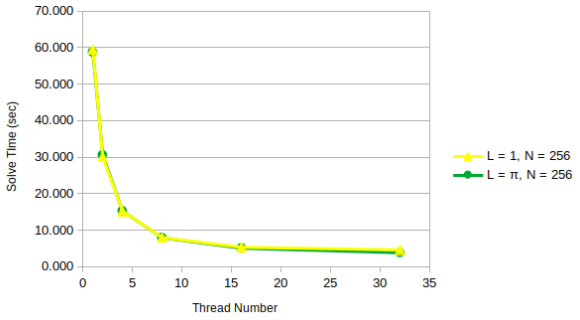

Рис. 2: Зависимость времени решения от числа нитей процесса для $N = 256$