МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Суперкомпьютерное моделирование и технологии

Отчет по заданию №4 «Задача для трёхмерного гиперболического уравнения в прямоугольном параллелепипеде»

Вариант №1

студент 628 группы
Гугучкин Егор Павлович

2022 год

1. Математическая постановка 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 $0 \le t \le T$ требуется найти решение $u(x, y, z, t)$ уравнения в частных производных $\frac{\partial^2 u}{\partial t^2} = \Delta u$ с начальными условиями

$$u\,(t = 0) = \phi(x, y, z)$$
$$\frac{\partial u}{\partial t}\,(t = 0) = 0$$
$$u(0, y, z, t) = 0$$
$$u(L_x, y, z, t) = 0$$
$$u(x, 0, z, t) = 0$$
$$u(x, L_y, z, t) = 0$$
$$u(x, y, 0, t) = u(x, y, L_z, t)$$
$$u_z(x, y, 0, t) = u_z(x, y, L_z, t)$$

2. Численный метод решения задачи

Введем на Ω сетку $\omega_{h\tau} = \overline{\omega_h} \times \omega_\tau$, где $T = T_0$,

$$L_x = L_{x0}, L_y = L_{y0}, L_z = L_{z0},$$

$$\overline{\omega_h} = \{(x_i = ih_x, y_j = jh_y, z_k = kh_z), i, j, k = \overline{0, N}, h_x N = L_x, h_y N = L_y, h_z N = L_z\},$$

$$\omega_\tau = \{t_n = n\tau, n = \overline{0, K}, \tau K = T\}$$

Через ω_h обозначим множество внутренних, а через γ_h – множество граничных узлов сетки $\overline{\omega_h}$.

Для аппроксимации исходного уравнения воспользуемся следующей системой уравнений:

$$\frac{u_{i,j,k}^{n+1} - 2u_{i,j,k}^{n} + u_{i,j,k}^{n-1}}{\tau^2} = \Delta_h u^n, (x_i, y_i, z_i) \in \omega_h, n = \overline{1, K-1}$$

Здесь Δ_h – семиточечный разностный аналог оператора Лапласа:

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

Приведенная выше разностная схема является явной – значения $u^{n+1}_{i,j,k}$ на $(n+1)$-ом шаге можно явным образом выразить через значения на предыдущих слоях.

Для начала счета должны быть заданы значения:

$$u^0_{i,j,k}, u^1_{i,j,k}, (x_i, y_i, z_i) \in \omega_h:$$

$$u^0_{i,j,k} = \phi(x_i, y_i, z_i), (x_i, y_i, z_i) \in \omega_h$$

$$u^1_{i,j,k} = u^0_{i,j,k} + \frac{\tau^2}{2}\Delta_h \phi(x_i, y_i, z_i)$$

$$u^{n+1}_{i,j,0} = u^{n+1}_{i,j,N}$$

$$u^{n+1}_{i,j,1} = u^{n+1}_{i,j,N+1}$$

$$i, j, k = \overline{0, N}$$

3. Программная реализация

Реализовано две версии программы: последовательная и параллельная с использованием MPI +OpenMP. В качестве входных аргументов задаются следующие переменные: N – количество точек сетки вдоль одной оси, L – длина сетки вдоль одной оси, *filename* – имя выходного файла. На выходе программа выводит N, число MPI-процессов и погрешность полученного решения.

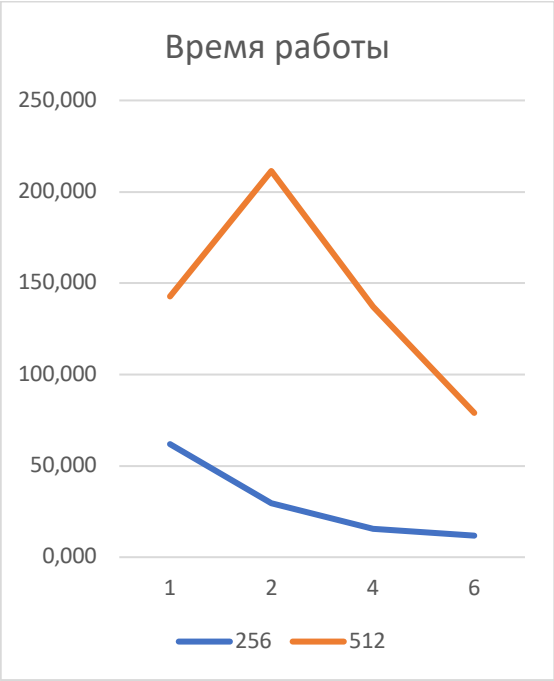
Параллельная версия программы выполнена следующим образом:

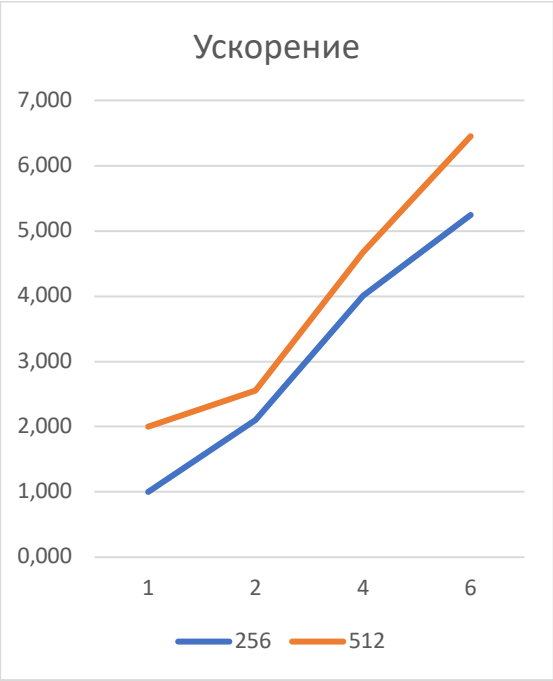
1. Сетка разделяется на *size* блоков, где *size* – число MPI-процессов. Каждому процессу выделяется свой блок.
2. Процессы находят ранги процессов-соседей и вычисляют координаты границ блока.
3. Процессы вычисляют u_0 и u_1 для своего блока.
4. Процессы вычисляют u_i и обмениваются граничными значениями.
5. Итоговая погрешность редуцируется с помощью оператора MPI_Reduce.

4. Результаты расчетов

MPI+OpenMP (128 нитей) программа; L = 1; сравнение с 1-процессной последовательной программой

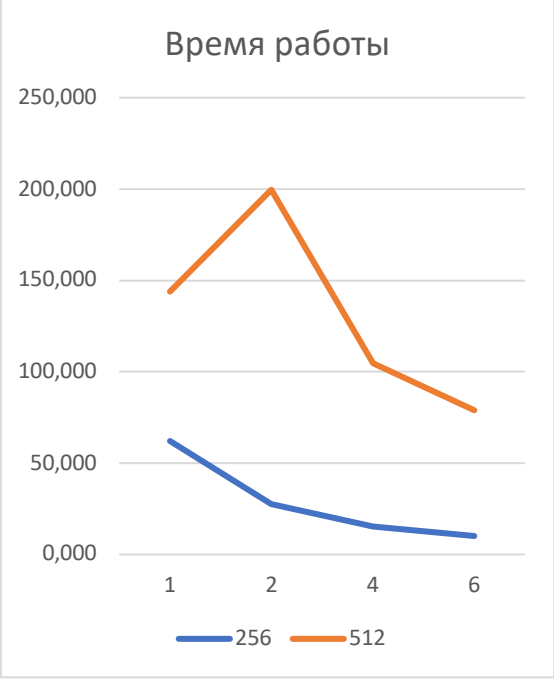
Число MPI-процессов N_p	Число точек сетки N^3	Время решения T	Ускорение S	Погрешность σ
1	256^3	61,881	1,000	5.9588e-08
2	256^3	29,396	2,105	5.9588e-08
4	256^3	15,453	4,005	5.9588e-08
6	256^3	11,798	5,245	5.9588e-08
1	512^3	80,990	1,000	3.96013e-09
2	512^3	181,983	0,445	3.96013e-09
4	512^3	121,522	0,666	3.96013e-09
6	512^3	67,1751	1,206	3.96013e-09

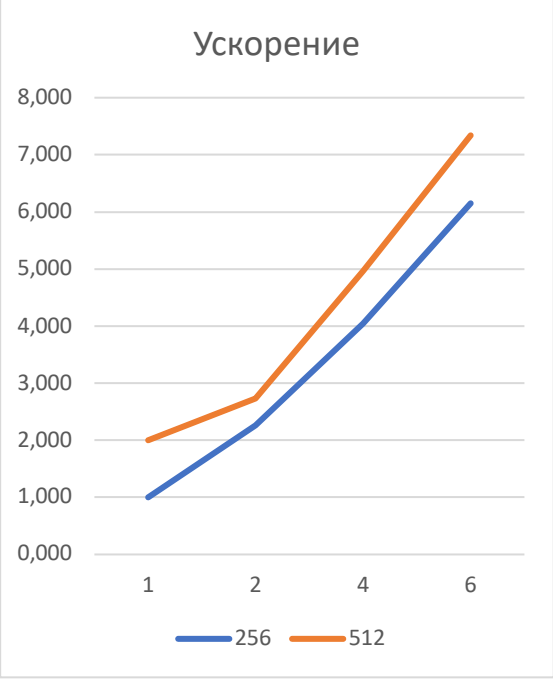




MPI+OpenMP (128 нитей) программа; L = π; сравнение с 1-процессной последовательной программой

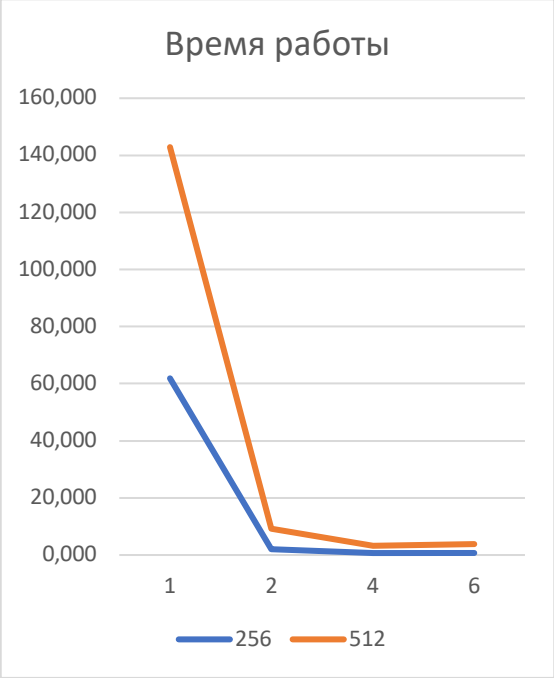
Число MPI-процессов N_p	Число точек сетки N^3	Время решения T	Ускорение S	Погрешность σ
1	256^3	62,084	1,000	7.3784e-09
2	256^3	27,47	2,260	7.3784e-09
4	256^3	15,3425	4,047	7.3784e-09
6	256^3	10,0954	6,150	7.3784e-09
1	512^3	81,881	1,000	1.73213e-09
2	512^3	172,148	0,476	1.73213e-09
4	512^3	89,3822	0,916	1.73213e-09
6	512^3	68,823	1,190	1.73213e-09

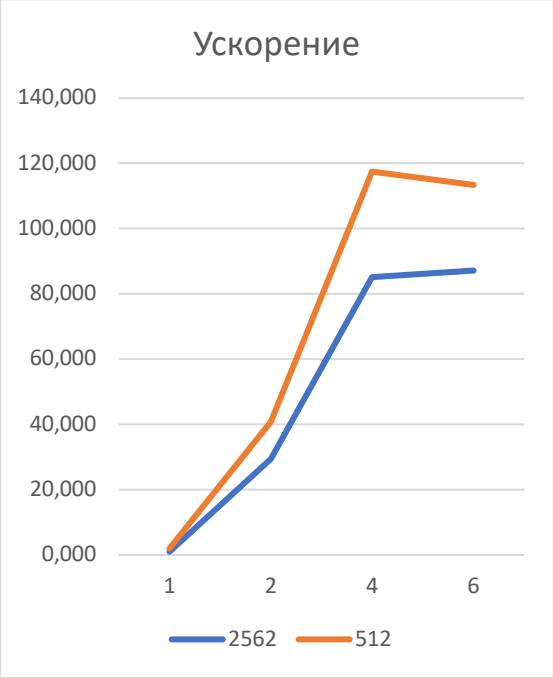




MPI+CUDA программа; L = 1; сравнение с 1-процессной последовательной программой

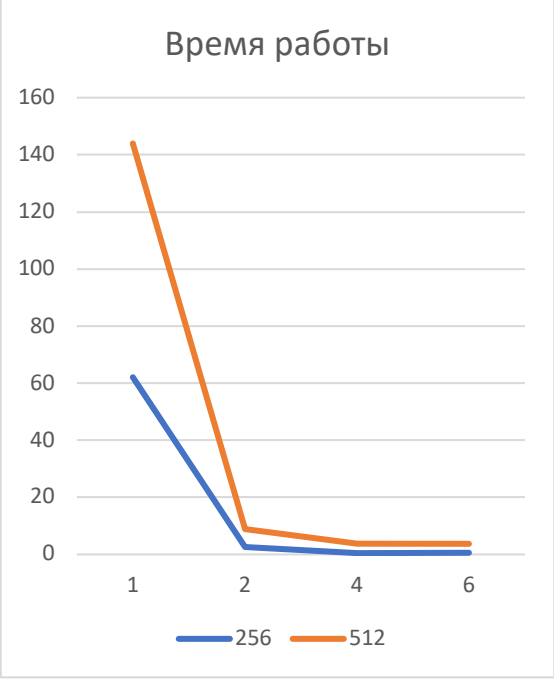
Число MPI-процессов N_p	Число точек сетки N^3	Время решения T	Ускорение S	Погрешность σ
1	256^3	61,881	1,000	5.9588e-08
2	256^3	2,095	29,538	5.9588e-08
4	256^3	0,727	85,124	5.9588e-08
6	256^3	0,710	87,181	5.9588e-08
1	512^3	80,990	1,000	3.96013e-09
2	512^3	7,152	11,325	3.96013e-09
4	512^3	2,501	32,381	3.96013e-09
6	512^3	3,097	26,148	3.96013e-09

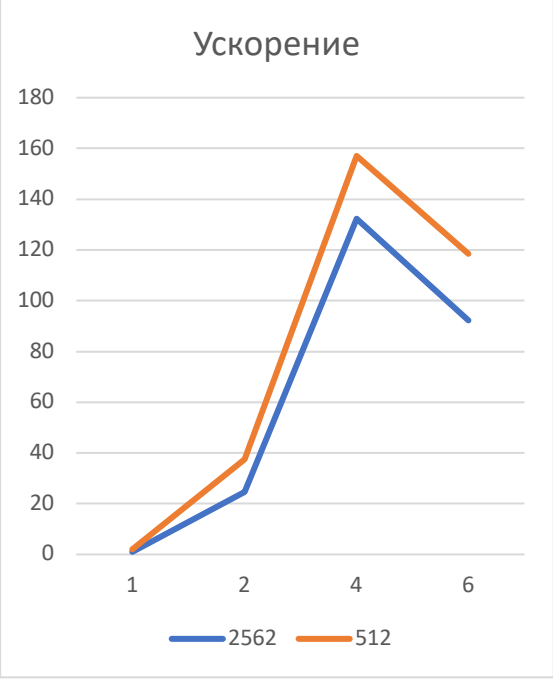




MPI+CUDA программа; L = π; сравнение с 1-процессной последовательной программой

Число MPI-процессов N_p	Число точек сетки N^3	Время решения T	Ускорение S	Погрешность σ
1	256^3	62,084	1,000	7.3784e-09
2	256^3	2,53	24,562	7.3784e-09
4	256^3	0,47	132,299	7.3784e-09
6	256^3	0,672	92,363	7.3784e-09
1	512^3	81,881	1,000	1.73213e-09
2	512^3	6,291	13,015	1.73213e-09
4	512^3	3,307	24,757	1.73213e-09
6	512^3	3,147	26,018	1.73213e-09





5. Выводы

По полученным результатам, можно сделать вывод, что обе реализации: MPI+OpenMP(128 нитей) и MPI+CUDA имеют потенциал для распараллеливания. Для MPI+OpenMP(128 нитей) выгодно использовать 4 и более MPI-процесса, так как при низком числе MPI-процессов, последовательная реализация оказывается быстрее. Реализация MPI+CUDA на полученных данных достигает наибольшего ускорения при 4, 6 MPI-процессов. Если сравнивать обе реализации между собой, то MPI+CUDA лучше распараллеливается по сравнению с MPI+OpenMP(128 нитей).