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

Суперкомпьютерное моделирование и технологии

Решение трёхмерного гиперболического уравнения в прямоугольном параллелепипеде

Вариант 5

Зыкин Ярослав Ильич
628 группа

9 декабря 2022 г.

Содержание

1 Постановка задачи

В трёхмерной замкнутой области

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

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

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

с начальными условиями

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

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

при условии, что на границах заданы периодические граничные условия для переменной x:

$$u(0, y, z, t) = u(L_x, y, z, t),\ u_x(0, y, z, t) = u_x(L_x, y, z, t),$$

и однородные граничные условия первого рода для переменных y и z:

$$u(x, 0, z, t) = 0,\ u(x, L_y, z, t) = 0,$$

$$u(x, y, 0, t) = 0,\ u(x, y, L_z, t) = 0.$$

Требуется программно реализовать численный метод решения поставленной задачи с использованием технологий MPI и OpenMP, а также провести исследование масштабируемости разработанной программы.

Аналитическое решение считается заданным и равно:

$$u_{analytical} = \sin(\frac{2\pi}{L_x}x) \cdot \sin(\frac{\pi}{L_y}y) \cdot \sin(\frac{\pi}{L_z}z) \cdot \cos(\alpha_t \cdot t + 2\pi), \alpha_t = \pi\sqrt{\frac{4}{L_x^2} + \frac{1}{L_y^2} + \frac{1}{L_z^2}}$$

Начальное условие выражается в виде:

$$\phi(x, y, z) = u_{analytical}(x, y, z, 0)$$

Значения константы T выбрано равным 1, $K = 1000$.

2 Численный метод решения

Для численного решения задачи введём на Ω сетку $\omega_{h\tau} = \overline{\omega_h} \times \omega_\tau$, где

$$T = T_0,\ L_x = L_{x_0},\ L_y = L_{y_0},\ L_z = L_{z_0},$$

$$\overline{\omega_h} = \{(x_i = ih_x,\ y_j = jh_y,\ z_k = kh_z),\ i,j,k = 0,1,\ldots,N,\ h_xN = L_x,\ h_yN = L_y,\ h_zN = L_z\},$$

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

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

Для аппроксимации исходного уравнения 1 воспользуемся следующей системой уравнений:

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

Здесь Δ_h – семиточечный разностный аналог оператора Лапласа:

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

Приведённая выше схема является явной – значения u^{n+1}_{ijk} можно явным образом выразить через значения на предыдущих слоях.

Для начала счёта (вычисления u^2_{ijk}) должны быть заданы значения u^0_{ijk} и u^1_{ijk}, $(x_i, y_j, z_k) \in \omega_h$. Для начальных условий имеем:

$$u^0_{ijk} = \phi(x_i, y_j, z_k),\ (x_i, y_j, z_k) \in \omega_h.$$

$$u^1_{ijk} = u^0_{ijk} + \frac{\tau^2}{2}\Delta_h\phi(x_i, y_j, z_k),\ (x_i, y_j, z_k) \in \omega_h.$$

Для граничных условий по x имеем:

$$u^{n+1}_{0jk} = u^{n+1}_{Njk},\ i,f,k = 0,1,\ldots,N.$$

Для граничных условий по y и z имеем:

$$u^{n+1}_{i0k} = 0,\ u^{n+1}_{iNk} = 0,\ u^{n+1}_{ij0} = 0,\ u^{n+1}_{ijN} = 0.$$

3 Программная реализация

Для решения поставленной задачи была написана программа на языке C++ с использованием технологий MPI и OpenMP.

Программа представляет из себя четыре файла:

- equation.cpp – основная вызывающая программа.
- solver.cpp – последовательное решение задачи.
- mpiSolver.cpp – решение задачи с использованием технологии MPI.
- ompMpiSolver.cpp – решение задачи с использованием технологий MPI+OpenMP.

Алгоритм программы:

1. Разбиваем область между процессами (в параллельной версии).
2. Используя формулы из Секции 2 находим значения u^0 и u^1.
3. Передаём необходимые значения последнего слоя соседям.
4. Считаем значения краевых точек по краевым условиям.
5. Высчитываем значения нового слоя во всех внутренних точках процесса.
6. Определяем максимальную погрешность на сетке между посчитанным и аналитическим решением.
7. Повторяем шаги 3-6 для необходимого количества слоёв по времени.

Особенности решения задачи с использованием технолии MPI

1. Разбиение области между процессами происходит с помощью функций библиотеки mpi MPI_Dims_create и MPI_Cart_create.
2. Соседние процессы (с которыми будут производиться обмены) определяются с помощью фукнции библиотеки mpi MPI_Cart_shift.
3. Первые и последние элементы слоя, хранящиеся на каждом процессе, если это необходимо, заполняются полученными от соседей элементами. Это сделано для сохранения основных вычисляющих функций.
4. Пересылки необходимых данных производятся в асинхронном режиме.
5. Максимальная погрешность вычисляется с помощью функции редукции MPI_Reduce.

Особенности решения задачи с использованием технолии OpenMP

Технология OpenMP используется для распараллеливания вычислений краевых условий и внутренних точек на каждом временном слое. Это возможно, потому что циклы в этих вычислениях не имеют зависимостей по данным.

4 Эксперименты

Для исследования масштабируемости разработанной программы были проведены следующие эксперименты.

- В каждом запуске выполнялось 20 шагов по времени.
- Параметр $L = L_x = L_y = L_z$ принимал значения 1 и π.
- Параметр N принимал значения 128^3 256^3 512^3.
- Программа запускалась в трёх режимах: последовательный, MPI, MPI+OpenMP.
- Для запусков MPI и OpenMP использовались 1, 4, 8, 16, 32 MPI процессов.
- В запусках OpenMP использовались 4 нити.
- Каждый вариант запускался по 5 раз с последующим усреднением результатов.

Результаты экспериментов приведены в таблицах 1 2 3 4 5 6 и на графиках 1 2 3 4 5 6 7 8.

Время работы последовательной версии программы и время работы MPI программы на одном процессе примерно совпадают. Это значит, что на инициализацию инфраструктуры для корректной работы MPI программы занимает достаточно небольшое время.

Во всех случаях MPI программы получились достаточно высокие значения ускорения, что означает хорошую распараллеливаемость алгоритма.

С помощью OpenMP-нитей внутри MPI-процессов было получено дополнительное ускорение, однако оно, в среднем, не превышает 2 (при 4 нитях). Это может означать то, что на пересылку данных тратится достаточное количество времени.

На рисунках 9 10 можно видеть графики аналитической функции, вычисленной функции и погрешности вычисления для двух значений L.

Число MPI-процессов	N^3	**Время решения** T	**Погрешность** δ
1	128^3	2.958	1.6E-06
1	256^3	20.566	3.5E-07
1	512^3	159.878	4.8E-08

Таблица 1: Результаты последовательной программы для Polus при $L = 1$

Число MPI-процессов	N^3	**Время решения** T	**Погрешность** δ
1	128^3	2.894	1.6E-07
1	256^3	20.615	4.0E-08
1	512^3	159.447	9.6E-09

Таблица 2: Результаты последовательной программы для Polus при $L = \pi$

Число MPI-процессов	N^3	**Время решения** T	**Ускорение** S	**Погрешность** δ
1	128^3	2.897	1.000	1.6E-06
4	128^3	0.771	3.759	1.6E-06
8	128^3	0.574	5.047	1.6E-06
16	128^3	0.338	8.565	1.6E-06
32	128^3	0.299	9.686	1.6E-06
1	256^3	21.504	1.000	3.5E-07
4	256^3	5.768	3.728	3.5E-07
8	256^3	3.427	6.276	3.5E-07
16	256^3	2.157	9.972	3.5E-07
32	256^3	1.349	15.943	3.5E-07
1	512^3	160.871	1.000	4.8E-08
4	512^3	42.873	3.752	4.8E-08
8	512^3	24.498	6.567	4.8E-08
16	512^3	13.330	12.069	4.8E-08
32	512^3	7.762	20.725	4.8E-08

Таблица 3: Результаты MPI программы для Polus при $L = 1$

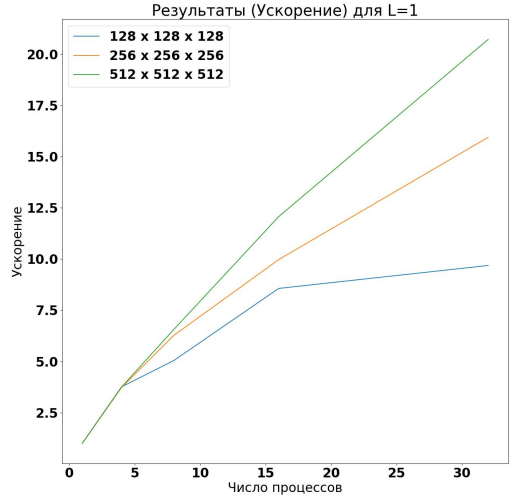

Рис. 1: Ускорение для MPI программы при $L = 1$

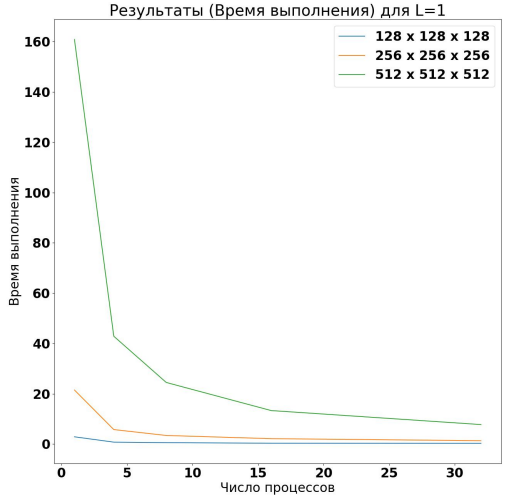

Рис. 2: Время MPI программы при $L = 1$

Число MPI-процессов	N^3	**Время решения** T	**Ускорение** S	**Погрешность** δ
1	128^3	2.758	1.000	1.6E-07
4	128^3	0.770	3.581	1.6E-07
8	128^3	0.535	5.158	1.6E-07
16	128^3	0.348	7.931	1.6E-07
32	128^3	0.331	8.342	1.6E-07
1	256^3	21.196	1.000	4.0E-08
4	256^3	5.764	3.678	4.0E-08
8	256^3	3.320	6.385	4.0E-08
16	256^3	2.034	10.420	4.0E-08
32	256^3	1.252	16.926	4.0E-08
1	512^3	160.316	1.000	9.6E-09
4	512^3	42.798	3.746	9.6E-09
8	512^3	23.956	6.692	9.6E-09
16	512^3	13.034	12.300	9.6E-09
32	512^3	7.788	20.585	9.6E-09

Таблица 4: Результаты MPI программы для Polus при $L = \pi$

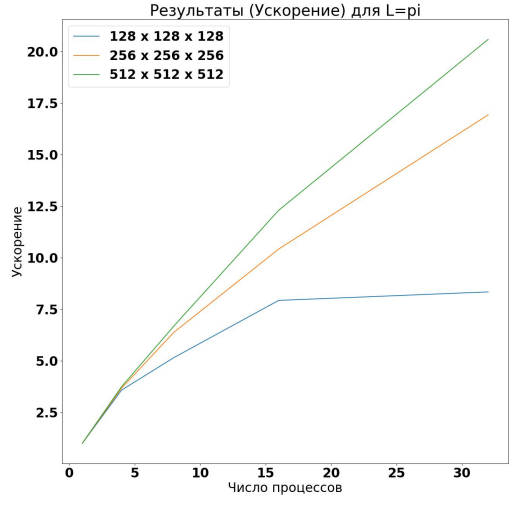

Рис. 3: Ускорение для MPI программы при $L = \pi$

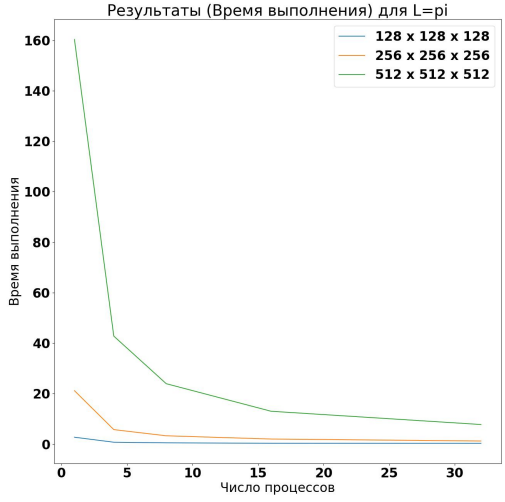

Рис. 4: Время MPI программы при $L = \pi$

Число MPI-процессов	N^3	**Время решения** T	**Ускорение** S	**Погрешность** δ
1	128^3	2.024	1.000	1.6E-06
4	128^3	0.694	2.916	1.6E-06
8	128^3	0.499	4.061	1.6E-06
16	128^3	0.357	5.668	1.6E-06
32	128^3	0.263	7.683	1.6E-06
1	256^3	14.753	1.000	3.5E-07
4	256^3	4.644	3.177	3.5E-07
8	256^3	2.841	5.193	3.5E-07
16	256^3	1.930	7.643	3.5E-07
32	256^3	1.252	11.780	3.5E-07
1	512^3	112.641	1.000	4.8E-08
4	512^3	31.229	3.607	4.8E-08
8	512^3	20.422	5.516	4.8E-08
16	512^3	12.146	9.274	4.8E-08
32	512^3	6.902	16.321	4.8E-08

Таблица 5: Результаты MPI+OpenMP программы для Polus при $L = 1$

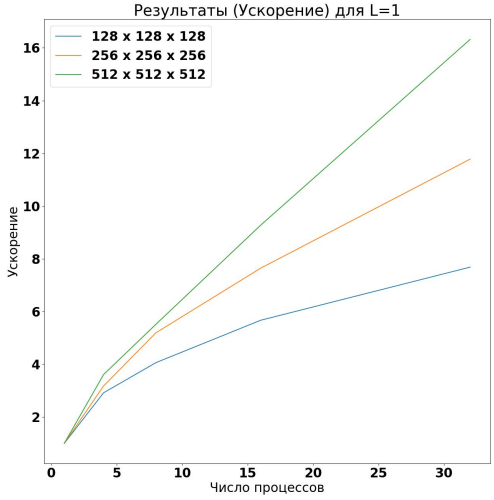

Рис. 5: Ускорение для MPI+OpenMP программы при $L = 1$

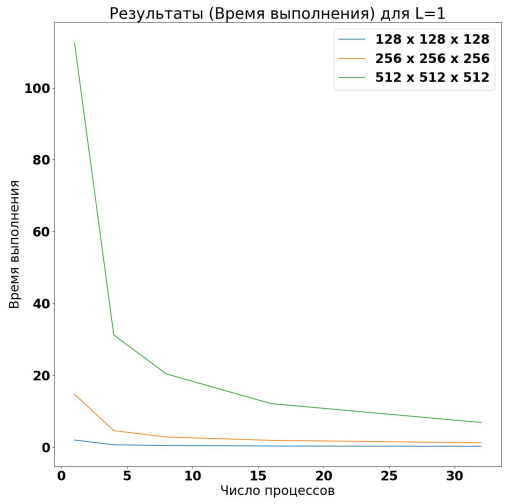

Рис. 6: Время MPI+OpenMP программы при $L = 1$

Число MPI-процессов	N^3	**Время решения** T	**Ускорение** S	**Погрешность** δ
1	128^3	2.042	1.000	1.6E-07
4	128^3	0.715	2.855	1.6E-07
8	128^3	0.517	3.952	1.6E-07
16	128^3	0.341	5.994	1.6E-07
32	128^3	0.252	8.106	1.6E-07
1	256^3	14.778	1.000	4.0E-08
4	256^3	4.331	3.412	4.0E-08
8	256^3	2.848	5.189	4.0E-08
16	256^3	1.720	8.591	4.0E-08
32	256^3	1.285	11.501	4.0E-08
1	512^3	112.334	1.000	9.6E-09
4	512^3	31.792	3.533	9.6E-09
8	512^3	20.752	5.413	9.6E-09
16	512^3	12.212	9.199	9.6E-09
32	512^3	6.950	16.162	9.6E-09

Таблица 6: Результаты MPI+OpenMP программы для Polus при $L = \pi$

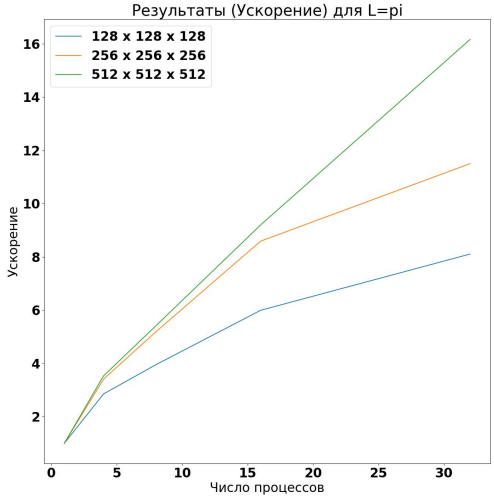

Рис. 7: Ускорение для MPI+OpenMP программы при $L = \pi$

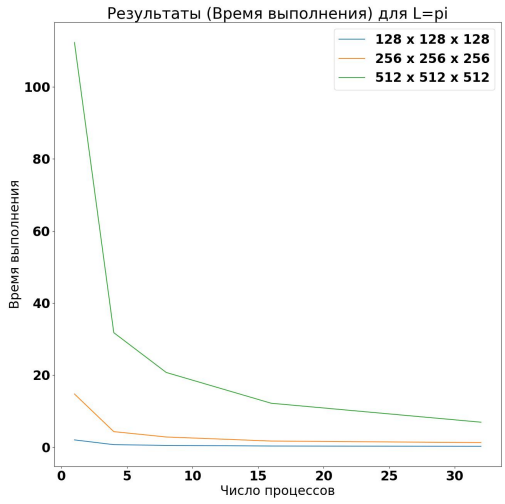

Рис. 8: Время MPI+OpenMP программы при $L = \pi$

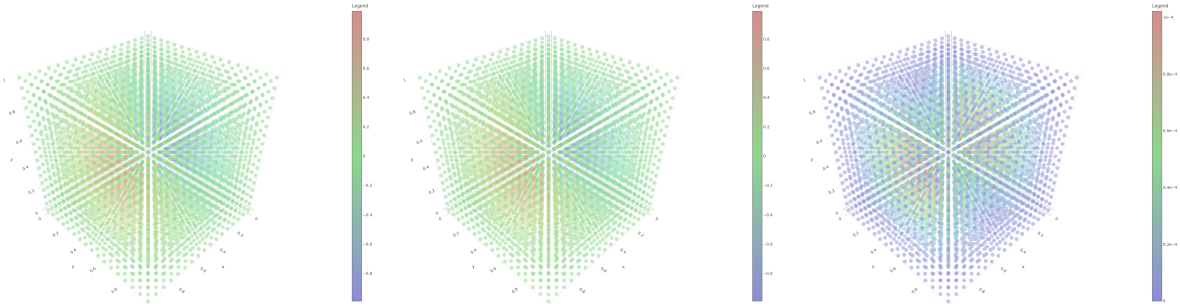

Рис. 9: Численное решение, аналитическое решение и погрешность при $L = 1$

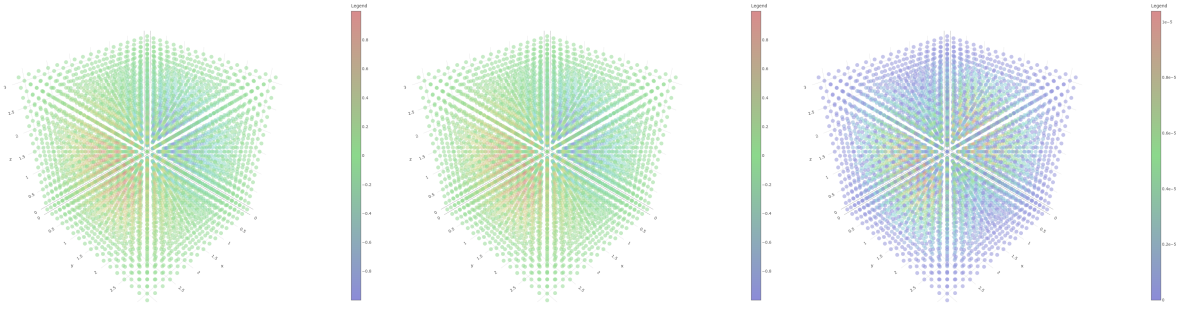

Рис. 10: Численное решение, аналитическое решение и погрешность при $L = \pi$

5 Выводы

В результате работы над поставленной задачей были разработаны программные средства для решения трёхмерного гиперболического уравнения в прямоугольном параллелепипеде с использованием технологий MPI и OpenMP. Разработанная программа показала хорошее ускорение при запуске на суперкомпьютерном кластере IBM Polus.

Результаты экспериментов показали, что с увеличением числа точек сетки позволяет увеличить эффективность работы программы. В этом случае накладные расходы для пересылки данных между процессами уменьшаются по сравнению с вычислительной сложностью задачи. Более того, с увеличением числа точек сетки увеличивает точность, с которой производятся расчёты.

Исходя из всего вышесказанного, можно сделать вывод, что, несмотря на возможные накладные расходы, использование суперкомпьютеров и технологий параллельного программирования позволяет серьёзно ускорить решение трёхмерного гиперболического уравненияв прямоугольном параллелепипеде.