Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет

По курсу: «Суперкомпьютерное моделирование и технологии»

«OpenMP»

Цирунов Леонид Александрович

группа 628

29 октября 2024 г.

Математическая постановка задачи

В трехмерной замкнутой области

$$\Omega = \left[0 \le x \le L_x\right] \times \left[0 \le y \le L_y\right] \times \left[0 \le z \le L_z\right]$$

Для $0 < t \le T$ требуется найти решение $u(x,\ y,\ z,\ t)$ уравнения в частных производных

$$\frac{\partial^2 u}{\partial t^2} = \triangle u$$

С начальными условиями

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

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

При условии, что на границах области заданы однородные граничные условия первого рода

$$\begin{aligned} u(0, y,\ z,\ t) &= 0, & u(L_x,\ y,\ z,\ t) &= 0, \\ u(x,\ 0, z,\ t) &= 0, & u(x,\ L_y,\ z,\ t) &= 0, \\ u(x,\ y,\ 0, t) &= 0, & u(x,\ y,\ L_z,\ t) &= 0 \end{aligned}$$

Либо периодические граничные условия

$$\begin{aligned} u(0, y,\ z,\ t) &= u(L_x,\ y,\ z,\ t), & u_x(0, y,\ z, t) &= u_x(L_x,\ y,\ z,\ t), \\ u(x,\ 0, z,\ t) &= u(x,\ L_y,\ z,\ t), & u_y(x,\ 0, z, t) &= u_y(x,\ L_y,\ z,\ t), \\ u(x,\ y,\ 0, t) &= u(x,\ y,\ L_z,\ t), & u_z(x,\ y,\ 0, t) &= u_z(x,\ y,\ L_z,\ t), \end{aligned}$$

Численный метод решения задачи

Введем на Ω сетку: $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$

$T = T_0$

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

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

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

Через ω_h обозначим множество внутренних, а через γ_h - множество гранитных узлов сетки $\bar{\omega}_h$.

Для аппроксимации исходного уравнения с начальными условиями воспользуемся следующей системой уравнений:

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

Где $\triangle_h$ - семиточечный разностный аналог оператора Лапласа:

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

Для начала счета должны быть заданы значения $u_{i,j,k}^{0}$ и $u_{i,j,k}^{1}$, $(x_i, y_j, z_k) \in \omega_h$.

$$u_{i,j,k}^{0} = \varphi(x_i, y_j,\ z_k), \qquad (x_i, y_j, z_k) \in \omega_h.$$

$$u_{i,j,k}^{1} = u_{i,j,k}^{0} + \frac{\tau^2}{2} \triangle_h \varphi(x_i, y_j,\ z_k)$$

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

Из варианта №8 следуют следующие формулы:

$$u_{analytical} = \sin(\frac{2\pi}{L_x}x)*\sin\left(\frac{4\pi}{L_y}y\right)*\sin\left(\frac{6\pi}{L_z}z\right)*\cos(a_t*t),$$

$$a_t = \pi\sqrt{\left(\frac{4}{L_x{}^2} + \frac{16}{L_y{}^2} + \frac{36}{L_z{}^2}\right)}$$

Граничные условия:

$$\text{П: } u(0, y, z, t) = u(L_x, y, z, t), \quad u_x(0, y, z, t) = u_x(L_x, y, z, t),$$
$$\text{П: } u(x, 0, z, t) = u(x, L_y, z, t), \quad u_y(x, 0, z, t) = u_y(x, L_y, z, t),$$
$$\text{П: } u(x, y, 0, t) = u(x, y, L_z, t), \quad u_z(x, y, 0, t) = u_z(x, y, L_z, t)$$

Алгоритм численного решения:

1. Вычисление граничных значений u^0 и u^1.
2. Вычисление u^0 внутри области: $u^0_{i,j,k} = \varphi(x_i, y_j, z_k)$
3. Вычисление u^1 внутри области: $u^1_{i,j,k} = u^0_{i,j,k} + \frac{\tau^2}{2} \triangle_h \varphi(x_i, y_j, z_k)$
4. Вычисление K – 1 раз u^{n+1}:
   $u^{n+1}_{ijk} = 2u^n_{ijk} - u^{n-1}_{ijk} + \tau^2 \triangle_h u^n$

Программный метод решения задачи

Программная реализация состоит из 4 файлов: ***main.cpp***, ***equation_solution.cpp***, ***equation.h*** и ***Makefile***.

Файл ***main.cpp*** содержит основную функцию получающую входные значения и запускающую решение задачи, а также две функции для сохранения результатов:

- ***dump_grid_to_CSV*** - сохранение погрешности или сетки, полученной численным или аналитическим способом, в файл в формате CSV (для дальнейшей визуализации);
- ***save_statistics*** - вывод в файл информации о времени работы решения и максимальной погрешности вычисления.

На вход программа получает 3 или 6 значений в зависимости от переданного ***L_type***. Описание параметров в порядке передачи:

- ***N*** - размер сетки по одной координате (конечный размер N^3);
- ***threads_num*** - количество потоков, которые будут задействованы;
- ***L_type*** - тип значений L по разным координатам, принимает значения: *1*, *pi*, *custom.*

В случае *1* и *pi*, значения L по всем координатам равны числу соответственно. Если ***L_type*** передано значение *custom* , в этом случае необходимо передать значения L по каждой координате: L_x, L_y, L_z.

Файл ***equation.h*** содержит объявления типов, класса сетки с функцией получения индекса элемента в линейном массиве описывающем сетку, а также функцию $u_{analytical}$ для вычисления аналитических значений точек сетки.

Файл ***equation_solution.cpp*** содержит основной алгоритм решения задачи. Содержит функции:

- ***laplace_operator*** - вычисление значения разностного аналога оператора Лапласа;

- ***init*** - вычисление внутренних u^0 , u^1 и граничных начальных точек, необходимых для запуска алгоритма;
- ***run_algo*** - итерация по K – 1 оставшимся временным шагам алгоритма с вычислением значений граничных и внутренних точек на новом шаге алгоритма. На каждом шаге производится подсчет максимальной погрешности численного решения от аналитического с выводом информации в консоль.
- ***solve_equation*** - задается количество потоков, отключается динамическое управление количеством потоков, инициализируются переменные, выполняется запуск алгоритма и ведется подсчет времени алгоритма.

Распараллеливание производится с использованием технологии OpenMP.

Файл ***Makefile*** содержит цели для компиляции и запуска задач. Для запуска на Polus используются команды «***make compile_polus***» и «***make run_all_polus***».

Результаты расчетов

Таблица 1: Результаты при L = 1

Число OpenMP нитей	Число точек сетки N по одной оси	Время решения Polus	Время решения локально M3 arm	Ускорение Polus	Ускорение локально	Погреш-ность
0-Sequential	128	8,77943	0,264147	1,000	2,418	1.4004e-06
1	128	8,69085	0,638651	1,0102	1,000	1.4004e-06
2	128	4,52765	0,325274	1,9391	1,963	1.4004e-06
4	128	2,31081	0,175627	3,7993	3,636	1.4004e-06
8	128	1,32673	0,093504	6,6173	6,830	1.4004e-06
16	128	0,944555	0,082437	9,2948	7,747	1.4004e-06
32	128	0,749173	0,075830	11,7188	8,422	1.4004e-06
0-Sequential	256	62,994	2,175710	1,000	2,234	3.49927e-07
1	256	63,2269	4,861470	0,9963	1,000	3.49927e-07
2	256	32,6417	2,540720	1,9299	1,9134	3.49927e-07
4	256	16,4166	1,364510	3,8372	3,5628	3.49927e-07
8	256	9,30982	0,692482	6,7664	7,0204	3.49927e-07
16	256	6,73194	0,567243	9,3575	8,5703	3.49927e-07
32	256	5,48987	0,502391	11,4746	9,6767	3.49927e-07
0-Sequential	512	484,384	16,5496	1,000	2,427	8.71479e-08
1	512	483,014	40,1724	1,0028	1,000	8.71479e-08
2	512	251,639	20,8492	1,9249	1,9268	8.71479e-08
4	512	126,323	10,7044	3,8345	3,7529	8.71479e-08
8	512	70,0937	5,4583	6,9105	7,3599	8.71479e-08
16	512	44,9736	3,7278	10,7704	10,7765	8.71479e-08
32	512	36,6006	3,7114	13,2343	10,8240	8.71479e-08

Таблица 2: Результаты при L = π

Число OpenMP нитей	Число точек сетки N по одной оси	Время решения Polus	Время решения локально M3 arm	Ускорение Polus	Ускорение локально	Погреш-ность
0-Sequential	128	8,47777	0,264720	1,000	2,3564	1.41974e-07
1	128	8,57208	0,623781	0,9890	1,000	1.41974e-07
2	128	4,43321	0,325018	1,9123	1,9192	1.41974e-07
4	128	2,28357	0,173767	3,7125	3,5898	1.41974e-07
8	128	1,21966	0,093419	6,9509	6,6773	1.41974e-07
16	128	0,966537	0,080144	8,7713	7,7833	1.41974e-07
32	128	0,740379	0,075907	11,4506	8,2177	1.41974e-07
0-Sequential	256	62,5438	2,1346	1,000	2,3695	3.55074e-08
1	256	63,1029	5,0580	0,9911	1,000	3.55074e-08
2	256	32,5949	2,6119	1,9188	1,9365	3.55074e-08
4	256	16,5531	1,3588	3,7784	3,7224	3.55074e-08
8	256	8,86288	0,7093	7,0568	7,1312	3.55074e-08
16	256	6,70346	0,5594	9,3301	9,0416	3.55074e-08
32	256	5,50078	0,5118	11,3700	9,8831	3.55074e-08
0-Sequential	512	480,591	16,8837	1,000	2,4103	8.8744e-09
1	512	481,711	40,6949	0,9977	1,000	8.8744e-09
2	512	248,499	20,8522	1,9340	1,9516	8.8744e-09
4	512	125,171	10,7015	3,8395	3,8027	8.8744e-09
8	512	66,1442	5,4574	7,2658	7,4568	8.8744e-09
16	512	45,7946	3,6415	10,4945	11,1753	8.8744e-09
32	512	37,1287	3,6401	12,9439	11,1798	8.8744e-09

График 1: зависимость ускорения на Polus от количества потоков при L = 1:

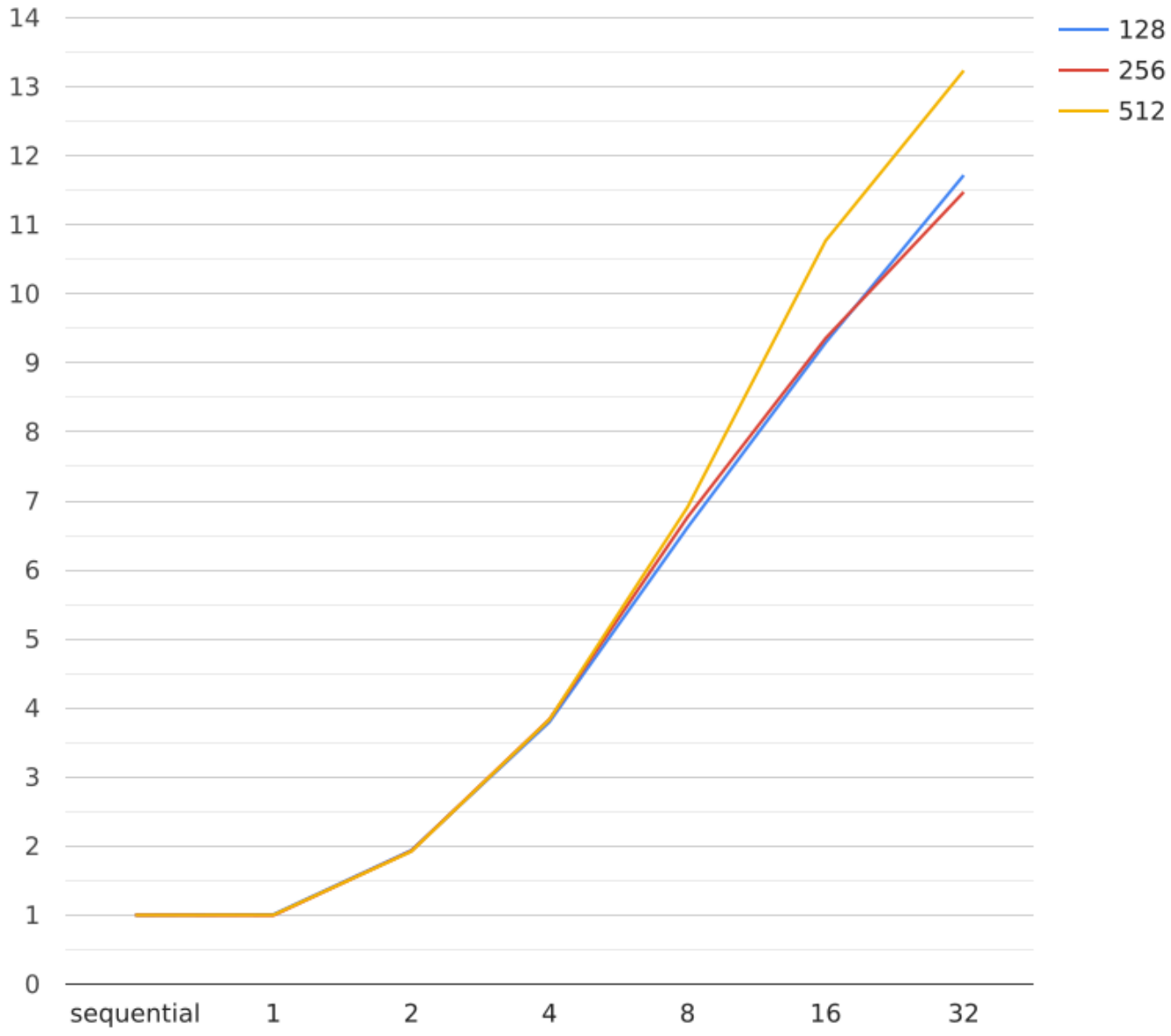


График 2: зависимость ускорения на Polus от количества потоков при L = π:

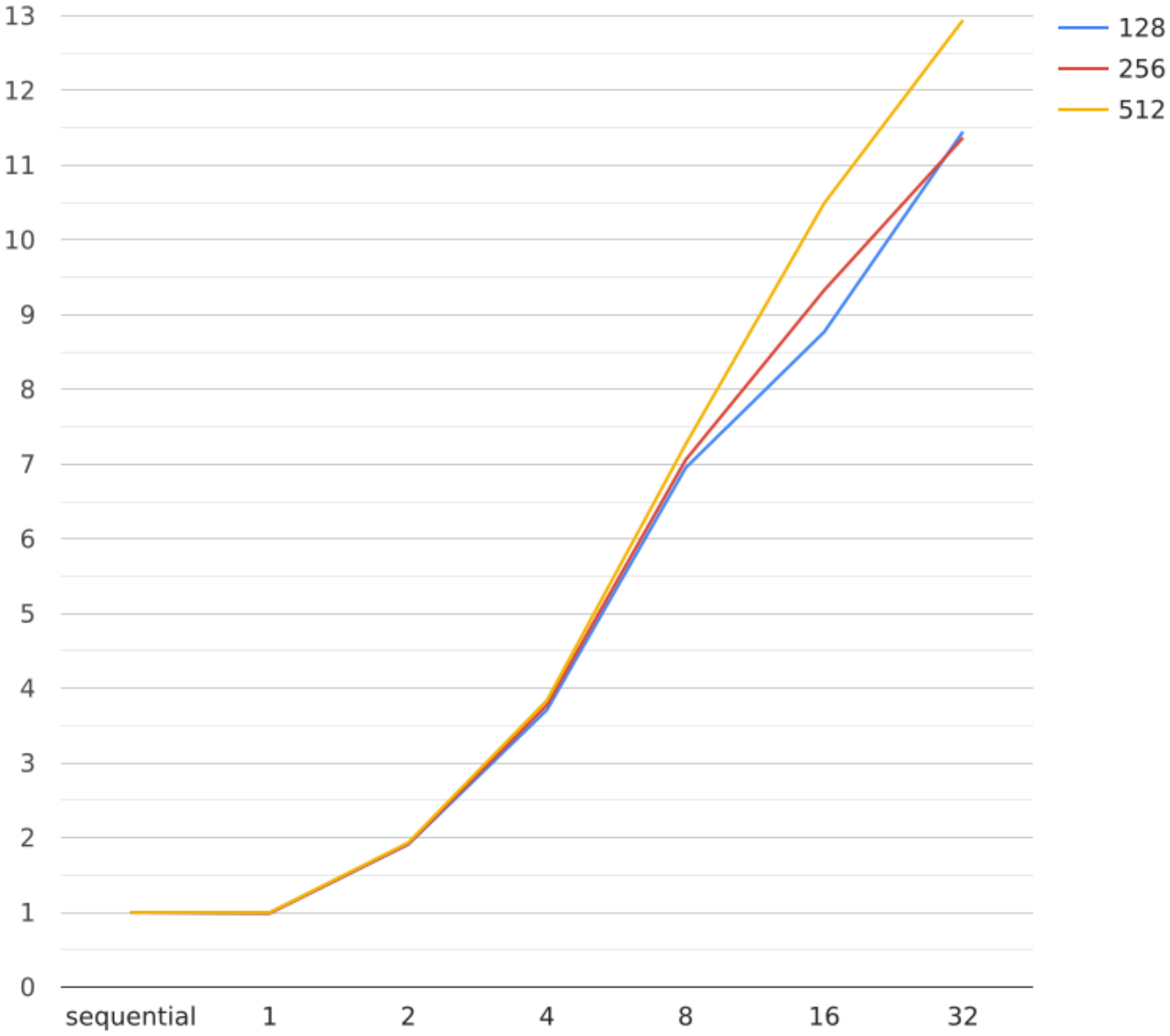


График 3: зависимость ускорения локально на M3 arm от количества потоков при L = 1:

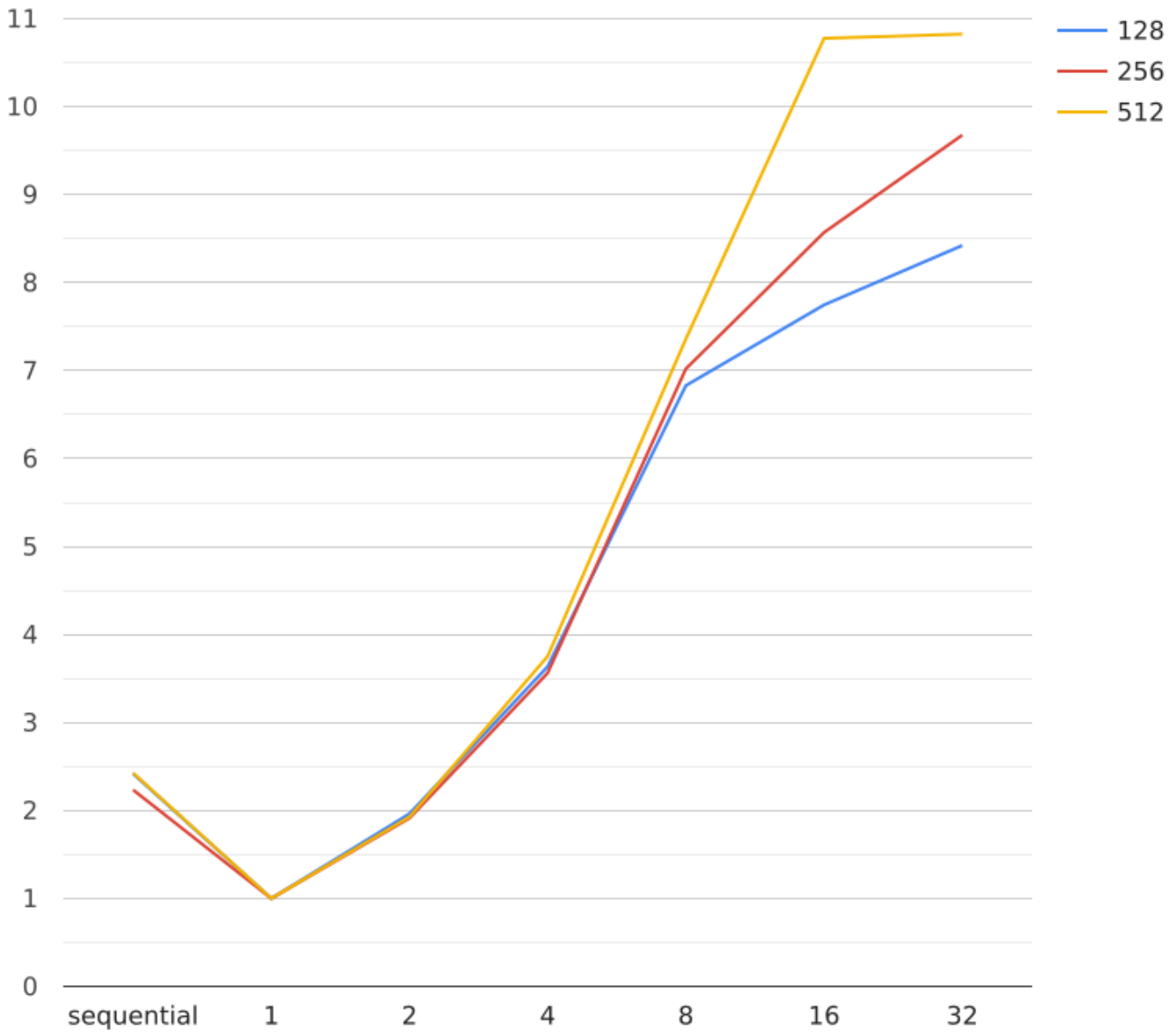
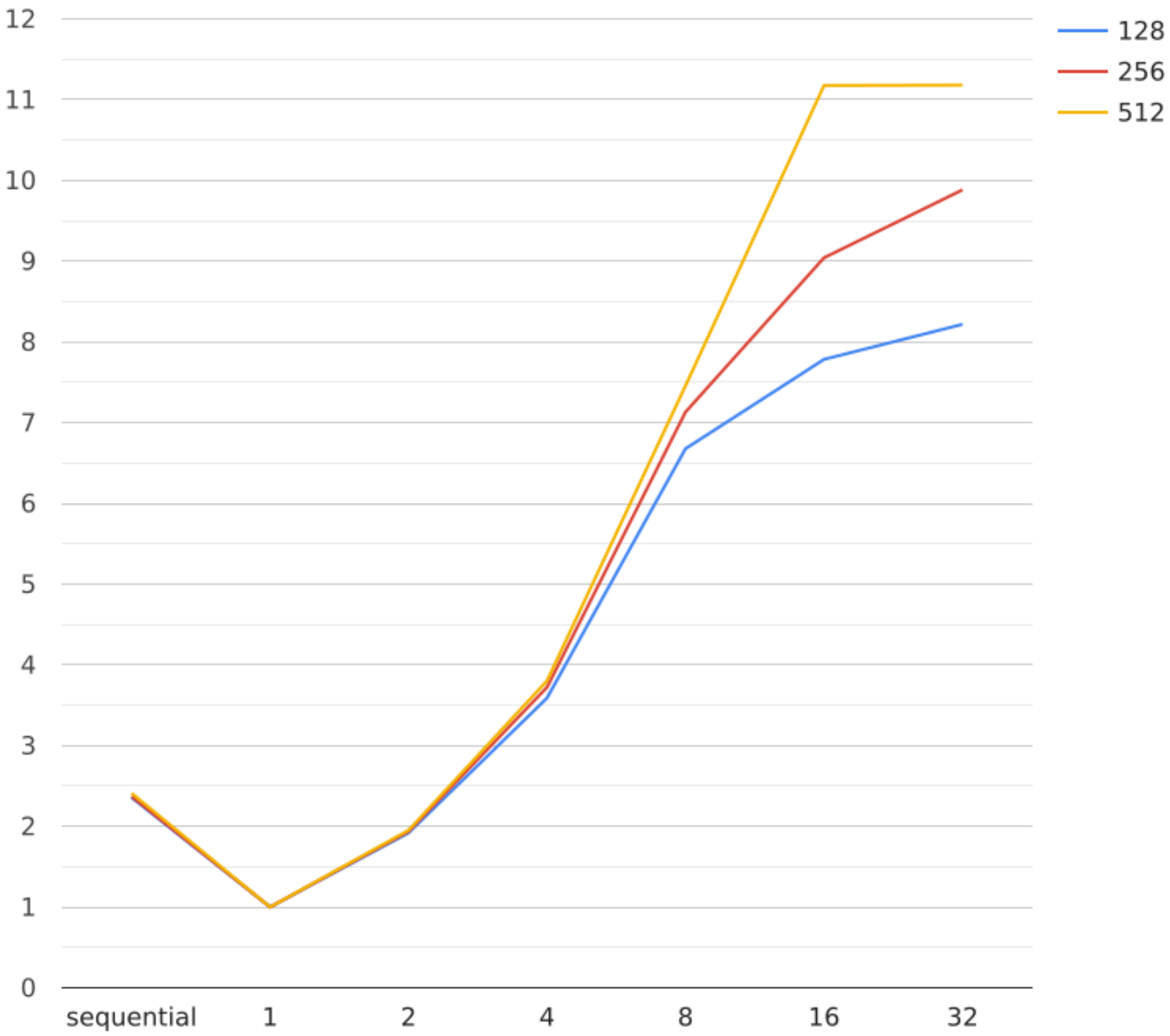
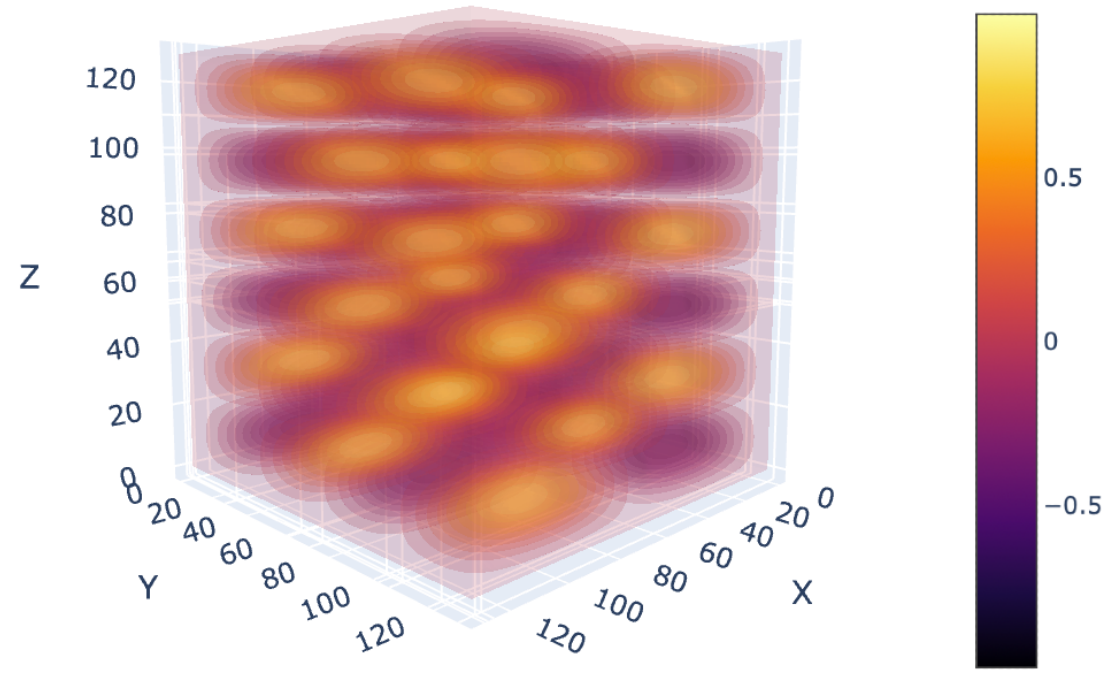
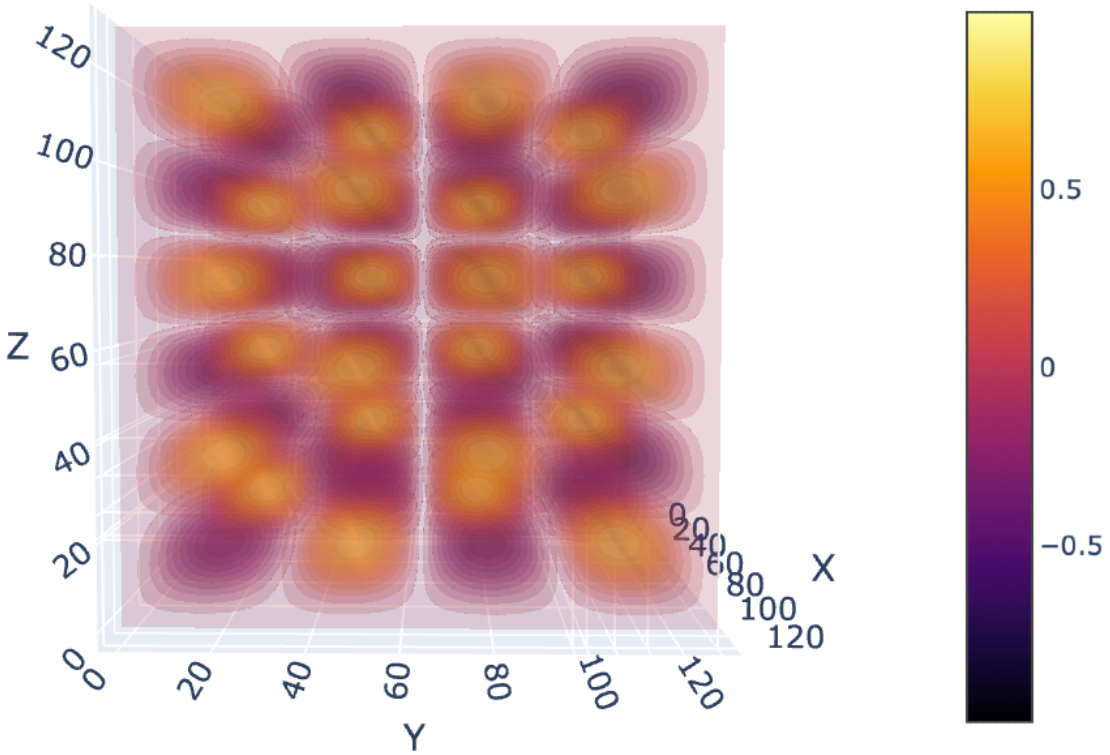


График 4: зависимость ускорения локально на M3 arm от количества потоков при L = π:

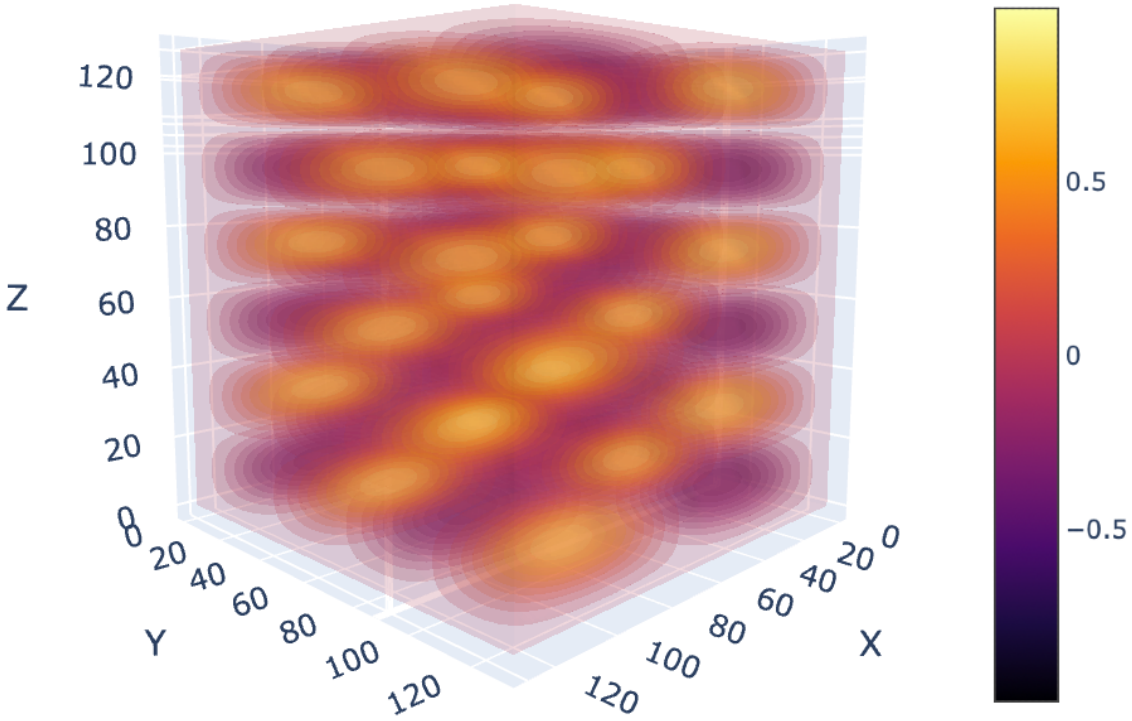


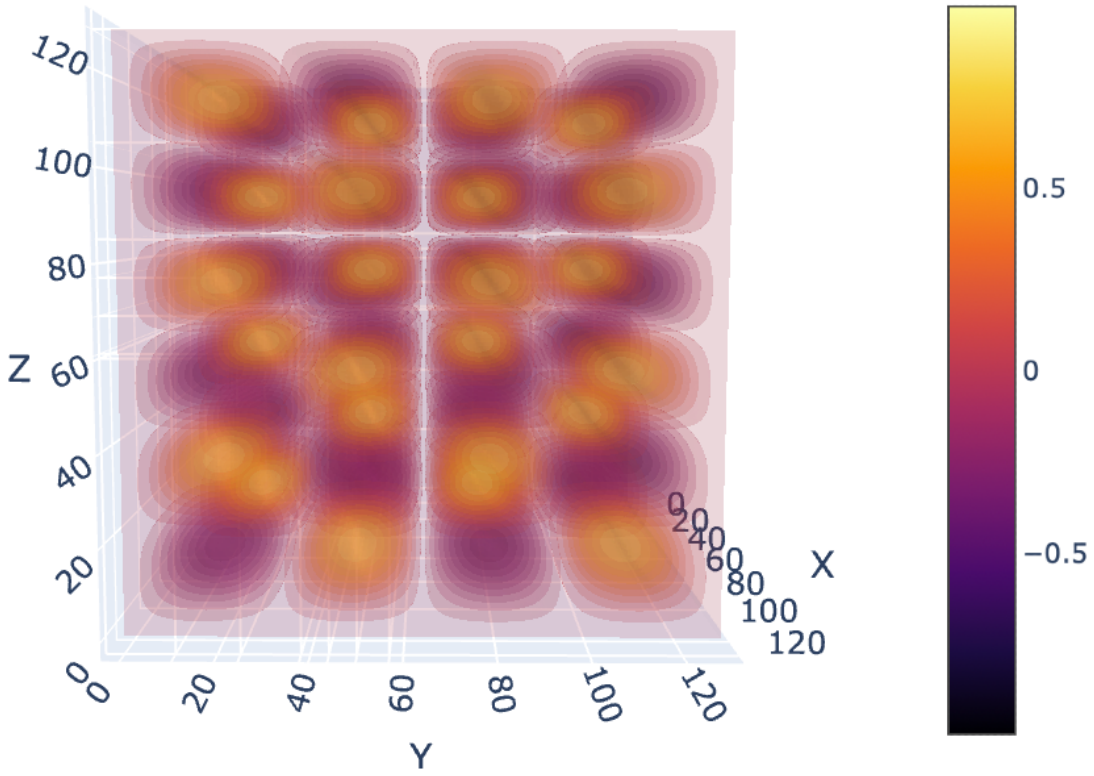
Визуализация сетки, полученной аналитическим решением:



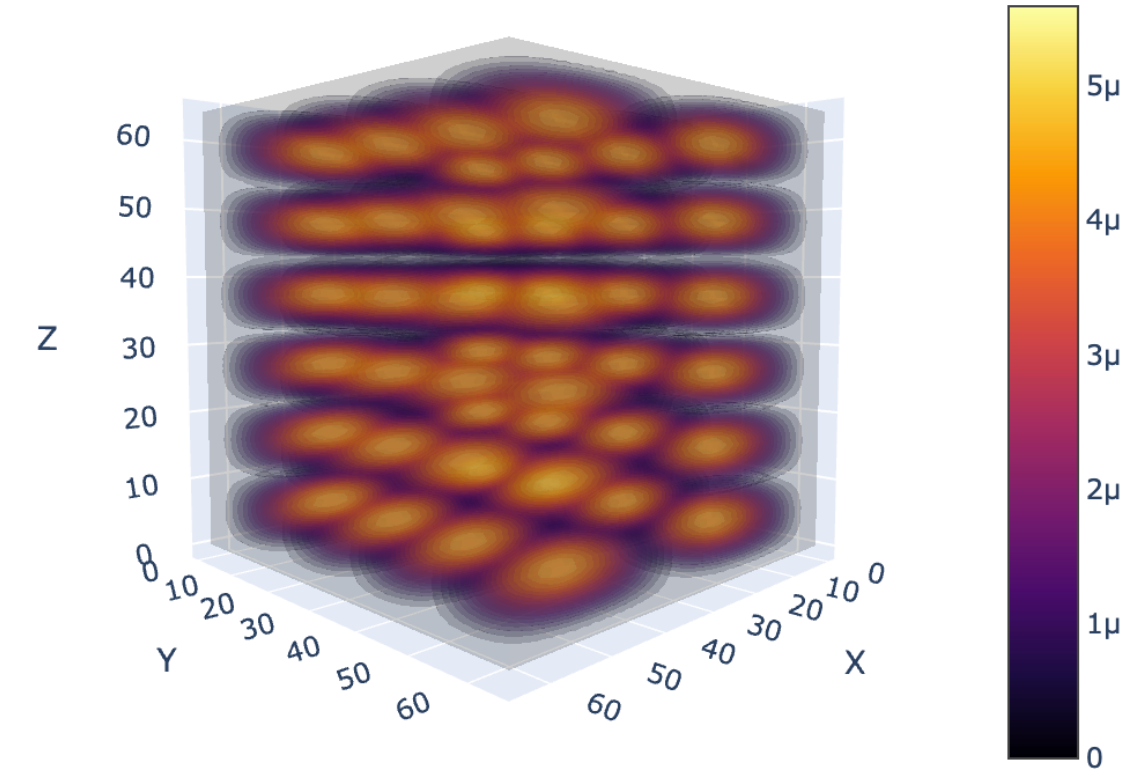


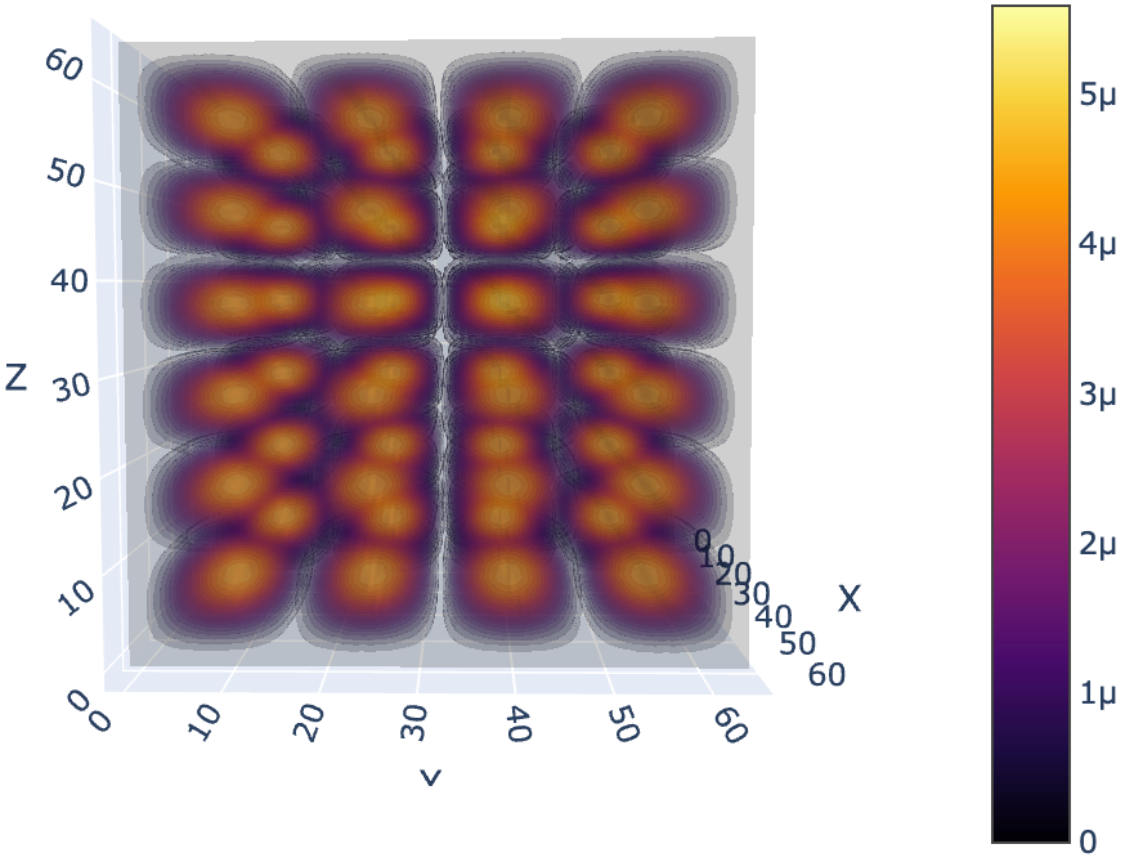
Визуализация сетки, полученной численным решением:





Визуализация сетки погрешности:





Вывод

Задача для трехмерного гиперболического уравнения в области, представляющей из себя прямоугольный параллелепипед, подходит для распараллеливания с помощью технологии OpenMP, позволяя получить ускорение вплоть до 12-13 раз. Причем при большем размере сетки ниже погрешность, а распараллеливание дает немного лучшие результаты по сравнению с мелкими сетками.

Также было замечено, что последовательный код работает немного быстрее, чем код в один поток, это связано с накладными расходами на создание потоков и ожидание их завершения, а также с оптимизациями компилятора.

Локально последовательная программа работает быстрее, чем в 1 и 2 потока, скорее всего это связано с тем, что последовательный код компилятором оптимизируется лучше и более радикально, чем код с параллельными областями, плюс все так же есть накладные расходы на создание и ожидание потоков.