Суперкомпьютерное моделирование и технологии

Отчёт

Задача для трёхмерного гиперболического уравнения в прямоугольном параллелепипеде. CUDA реализация.

Перминов Андрей Игоревич

Вариант 3

06.11.2021

Содержание

Математическая постановка задачи 3
Численный метод решения задачи 3
Программная реализация 5
Типы граничных условий: 5
Типы стратегий разбиения 5
Особенности параллельной реализации 6
Блочное разбиение: 6
Ленточное разбиение: 6
Разбиение разложением на три сомножителя 6
Получающиеся разбиения 7
Блочное разбиение 7
Ленточное разбиение 7
Разбиение разложением на три сомножителя 7
Графики аналитического и полученного решений 8
Решение при $L_x = L_y = L_z = 1$ 8
Решение при $L_x = L_y = L_z = \pi$ 8
Результаты запусков программы на кластере Polus 9
Lx = Ly = Lz = 1, N = 128, K = 2000, блочное разбиение 9
Lx = Ly = Lz = 1, N = 256, K = 2000, блочное разбиение 9
Lx = Ly = Lz = 1, N = 512, K = 2000, блочное разбиение 9
Lx = Ly = Lz = 1, N = 128, K = 2000, ленточное разбиение 10
Lx = Ly = Lz = 1, N = 256, K = 2000, ленточное разбиение 10
Lx = Ly = Lz = 1, N = 512, K = 2000, ленточное разбиение 10
Lx = Ly = Lz = 1, N = 128, K = 2000, разбиение разложением 11
Lx = Ly = Lz = 1, N = 256, K = 2000, разбиение разложением 11
Lx = Ly = Lz = 1, N = 512, K = 2000, разбиение разложением 11
Выводы 12

Математическая постановка задачи

В трёхмерной замкнутой области

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

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

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

с начальными условиями

$$\begin{cases} u|_{t=0} = \phi(x, y, z) \\ \left.\frac{\partial u}{\partial t}\right|_{t=0} = 0 \\ u(0, y, z, t) = 0 \\ u(L_x, y, z, t) = 0 \\ u(x, 0, z, t) = u(x, L_y, z, t) \\ u_y(x, 0, z, t) = u_y(x, L_y, z, t) \\ u(x, y, 0, t) = 0 \\ u(x, y, L_z, t) = 0 \end{cases}$$

Численный метод решения задачи

Введём на Ω сетку $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$, где

$T = T_0$

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

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

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

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

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:

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

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

Для начала счёта должны быть заданы значения $u^0_{i,j,k}, u^1_{i,j,k}, (x_i, y_i, z_i) \in \omega_h$:

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

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

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

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

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

Программная реализация

Реализована консольная гибридная параллельная (MPI + CUDA) программа, принимающая входные данные в виде аргументов командной строки. Используются следующие аргументы (доступно при вызове с единственным аргументом –help, полностью аналогично предыдущим версиям программы):

- -d – отладочный режим (по умолчанию не используется)
- -Lx – длина параллелепипеда вдоль оси X (по умолчанию 1)
- -Ly – длина параллелепипеда вдоль оси Y (по умолчанию 1)
- -Lz – длина параллелепипеда вдоль оси Z (по умолчанию 1)
- -T – конечное время сетки (по умолчанию 1)
- -N – количество точек пространственной сетки (по умолчанию 40)
- -K – количество точек временной сетки (по умолчанию 100)
- -steps – количество шагов для решения (по умолчанию 20)
- -btx – тип граничного условия вдоль оси X (по умолчанию однородные первого рода)
- -bty – тип граничного условия вдоль оси Y (по умолчанию периодические-численные)
- -btz – тип граничного условия вдоль оси Z (по умолчанию однородные первого рода)
- -s – стратегия разбиения на блоки (по умолчанию блочное, доступно только для MPI версии)
- -on – путь к json файлу для сохранения численного решения (по умолчанию не используется)
- -oa – путь к json файлу для сохранения аналитического решения (по умолчанию не используется)
- -od – путь к json файлу для сохранения погрешности (по умолчанию не используется)
- -o – путь к текстовому файлу для вывода результатов (по умолчанию output.txt)

Типы граничных условий:

- first-kind (f) – однородные граничные условия первого рода
- periodic-analytical (pa) – аналитические периодические граничные условия

Типы стратегий разбиения

- blocks (b) – блочное разбиение
- tapes (t) – ленточное разбиение
- products (p) – разбиение разложением на три сомножителя ($P = P_x \times P_y \times P_z$)

Для визуализации получаемых решений написан визуализатор (принимает json файлы, генерируемые программой): solve-visualizer

Особенности параллельной реализации

Для распараллеливания вся сетка разбивается на области (также прямоугольные параллелепипеды) в количестве используемых процессов. Операции, производимые с получаемыми параллелепипедами, обрабатываются с помощью CUDA ядер.

Блочное разбиение:

- начнём разбиение с параллелепипеда $[0, N]\ x\ [0, N]\ x\ [0, N]$, выберем начальную ось (X) и запустим рекурсивный процесс
- если текущее количество областей ($size$) равно 1, вернём обрабатываемый параллелепипед
- если размер нечётный, то по текущей оси выберем область $\frac{1}{size}$ и сделаем из неё параллелепипед, и продолжим разбивать область $1\ - \frac{1}{size}$
- по выбранной оси делим область пополам и рекурсивно запускаем для этих подобластей переходя на следующую ось ($X \to Y,\ Y \to Z,\ Z \to X$).

Ленточное разбиение:

- выбираем ось, вдоль которой будет производиться разбиение
- начинаем делить на параллелепипеды по $\max\left(1, \frac{x_{max} - x_{min}}{size}\right)$ элементов

Разбиение разложением на три сомножителя

- Раскладываем число на простые сомножители
- Если получилось менее трёх сомножителей, добавляем единицы до трёх
- В противном случае, пока сомножителей больше 3, заменяем пары чисел их произведением, а затем сортируем последовательность.

Для наглядности реализован визуализатор разбиения: split-visualizer

Получающиеся разбиения

Блочное разбиение

P=2

P=3

P=4

P=5

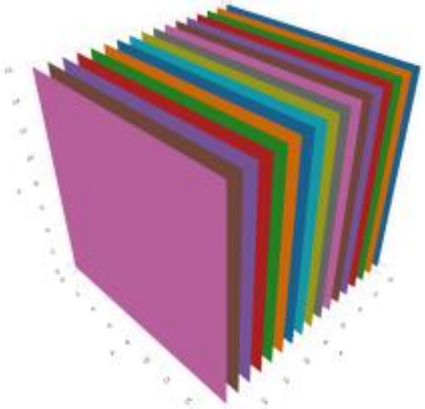
P=10

P=14

Ленточное разбиение

P=2

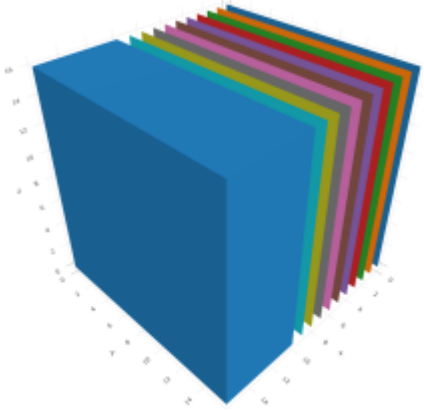
P=10



P=17

Разбиение разложением на три сомножителя

P=6

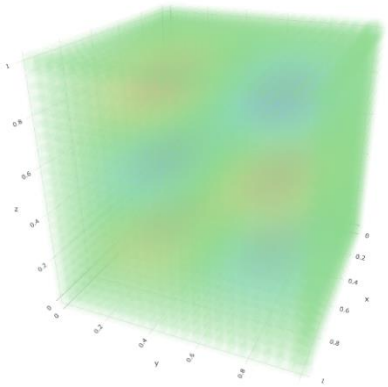


P=11

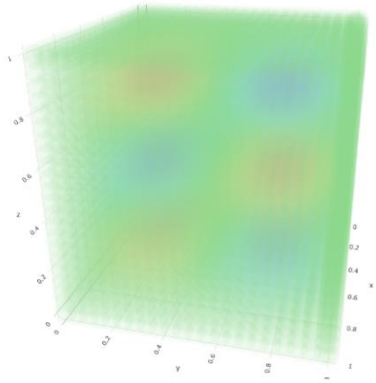
P=18

Графики аналитического и полученного решений

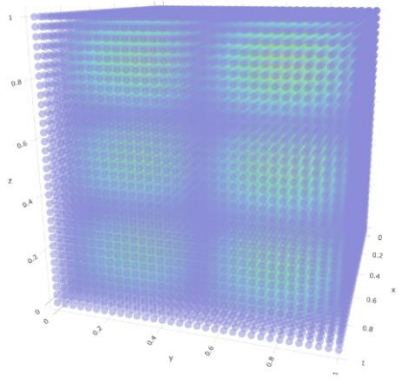
Решение при $L_x = L_y = L_z = 1$



аналитическое

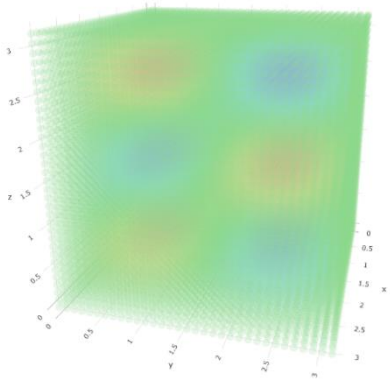


полученное

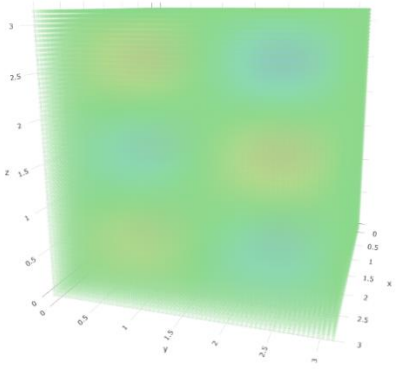


погрешность

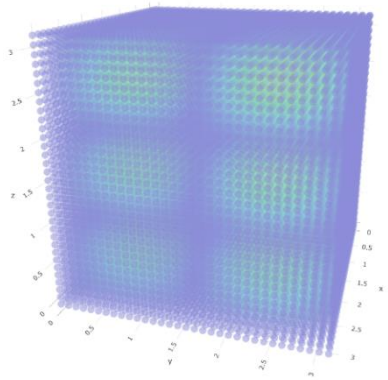
Решение при $L_x = L_y = L_z = \pi$



аналитическое



полученное



погрешность

Результаты запусков программы на кластере Polus

Lx = Ly = Lz = 1, N = 128, K = 2000, блочное разбиение

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	10.8421	1.000	0.00541829	0.625809	1.000	0.00541829	17.325
2	5.56431	1.949	0.00541829	0.491011	1.275	0.00541829	11.332
4	2.80274	3.868	0.00541829	0.235194	2.661	0.00541829	11.917
6	1.95402	5.549	0.00541829	0.23938	2.614	0.00541829	8.163

Lx = Ly = Lz = 1, N = 256, K = 2000, блочное разбиение

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	83.0349	1.000	0.00542011	3.19642	1.000	0.00542011	25.977
2	42.5433	1.952	0.00542011	2.81246	1.137	0.00542011	15.127
4	21.3244	3.894	0.00542011	1.39997	2.283	0.00542011	15.232
6	14.7643	5.624	0.00542011	0.986884	3.239	0.00542011	14.961

Lx = Ly = Lz = 1, N = 512, K = 2000, блочное разбиение

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	644.438	1.000	0.00542056	25.6305	1.000	0.00542056	25.143
2	328.795	1.960	0.00542056	21.9411	1.168	0.00542056	14.985
4	170.1	3.789	0.00542056	10.5393	2.432	0.00542056	16.140
6	114.436	5.631	0.00542056	7.13817	3.591	0.00542056	16.032

Lx = Ly = Lz = 1, N = 128, K = 2000, ленточное разбиение

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	10.828	1.000	0.00541829	0.417379	1.000	0.00541829	25.943
2	5.54688	1.952	0.00541829	0.395455	1.055	0.00541829	14.027
4	2.87798	3.762	0.00541829	0.235013	1.776	0.00541829	12.246
6	2.03811	5.313	0.00541829	0.254023	1.643	0.00541829	8.023

Lx = Ly = Lz = 1, N = 256, K = 2000, ленточное разбиение

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	83.024	1.000	0.00542011	3.18848	1.000	0.00542011	26.039
2	42.4604	1.955	0.00542011	2.81574	1.132	0.00542011	15.080
4	21.7445	3.818	0.00542011	1.39985	2.278	0.00542011	15.533
6	14.8906	5.576	0.00542011	0.968354	3.293	0.00542011	15.377

Lx = Ly = Lz = 1, N = 512, K = 2000, ленточное разбиение

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	663.803	1.000	0.00542056	25.4344	1.000	0.00542056	26.099
2	335.254	1.980	0.00542056	22.2357	1.144	0.00542056	15.077
4	169.7	3.912	0.00542056	10.5945	2.401	0.00542056	16.018
6	117.82	5.634	0.00542056	7.512	3.386	0.00542056	15.684

Lx = Ly = Lz = 1, N = 128, K = 2000, разбиение разложением

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	10.8417	1.000	0.00541829	0.41405	1.000	0.00541829	26.185
2	5.61019	1.933	0.00541829	0.398651	1.039	0.00541829	14.073
4	2.96835	3.652	0.00541829	0.234564	1.765	0.00541829	12.655
6	1.97781	5.482	0.00541829	0.193185	2.143	0.00541829	10.238

Lx = Ly = Lz = 1, N = 256, K = 2000, разбиение разложением

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	83.0116	1.000	0.00542011	3.20082	1.000	0.00542011	25.934
2	42.8079	1.939	0.00542011	2.79625	1.145	0.00542011	15.309
4	22.1411	3.749	0.00542011	1.39953	2.287	0.00542011	15.820
6	14.9234	5.563	0.00542011	0.969411	3.302	0.00542011	15.394

Lx = Ly = Lz = 1, N = 512, K = 2000, разбиение разложением

Число MPI процессов (P)	MPI			MPI + CUDA			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
1	625.334	1.000	0.00542056	25.4935	1.000	0.00542056	24.529
2	334.403	1.870	0.00542056	22.2011	1.148	0.00542056	15.062
4	201.045	3.110	0.00542056	10.5416	2.418	0.00542056	19.072
6	112.172	5.575	0.00542056	7.47077	3.412	0.00542056	15.015

Выводы

Задача для трёхмерного гиперболического уравнения в прямоугольном параллелепипеде отлично подходит для распараллеливания. В результате получены программные средства, решающие поставленную задачу как средствами MPI, так и гибридным способом – MPI+CUDA.

Важную роль в MPI распараллеливании задачи сеточного метода играет способ разбиения на блоки. Проведены эксперименты для блочного и ленточного разбиений, а также разбиения путём разложения числа на сомножители. При желании, программу легко дополнить другими способами разбиения. Для этого достаточно добавить новый тип разбиения и описать алгоритм в соответствующем классе.

Использование CUDA для распараллеливания полученной MPI программы более чем целесообразно, о чём свидетельствуют полученные значения ускорений по сравнению с версией без использования CUDA вызовов.