

# Суперкомпьютерное моделирование и технологии

Отчет

Задача для трехмерного гиперболического уравнения  
в прямоугольном параллелепипеде

Лазарев Владимир Александрович

2 вариант

15.12.2021

## Оглавление

Математическая постановка задачи .....	3
Численный метод решения задачи .....	3
Программная реализация.....	5
Результаты запусков программ на различных кластерах .....	9
Выводы.....	13

## Математическая постановка задачи

В трехмерной замкнутой области

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

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

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

## Численный метод решения задачи

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

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

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

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

Через  $\omega_h$  обозначим множество внутренних, а через  $\gamma_h$  — 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  $\Delta_h$  – семиточечный разностный аналог оператора Лапласа:

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

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

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

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

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

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

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

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

## Программная реализация

Реализована гибридная параллельная программа (MPI + OpenMP). Принимает входные данные в виде аргументов командной строки. Используются следующие аргументы:

- $-Lx=$  – 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 128);
- $-K=$  – количество точек временной сетки (по умолчанию 2000);
- $-steps=$  – количество шагов для решения (по умолчанию 5);
- $-omp=$  – количество OpenMP нитей (по умолчанию 1).

Для распараллеливания вся сетка разбивается на области в количестве используемых процессов по следующему алгоритму:

- Начинается разбиение с параллелепипеда  $[0, N] \times [0, N] \times [0, N]$ , выбирается  $X$  как начальная ось;

- Если оставшееся количество областей для разбиения равно 1, возвращается обрабатываемый параллелепипед;
- Если размер нечетный, то по текущей оси выбирается область  $\frac{1}{countOfProcesses}$  и делается из нее параллелепипед, также продолжается разбиваться область  $1 - \frac{1}{countOfProcesses}$ ;
- По выбранной оси делим область пополам и рекурсивно запускаем для этих подобластей переходя на следующую ось  $X \rightarrow Y, Y \rightarrow Z, Z \rightarrow X$ ).

# График аналитического и полученного решений

Решение при  $L_x = L_y = L_z = 1$

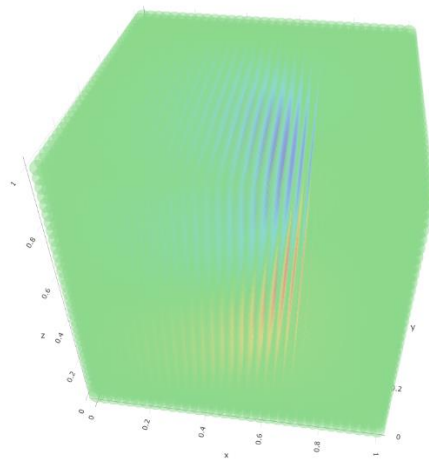


Рисунок 1. Аналитическое решение

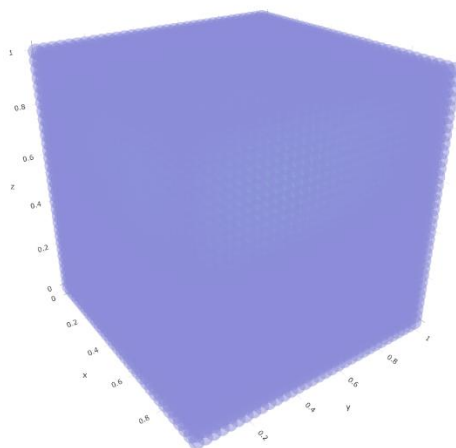


Рисунок 2 Погрешность

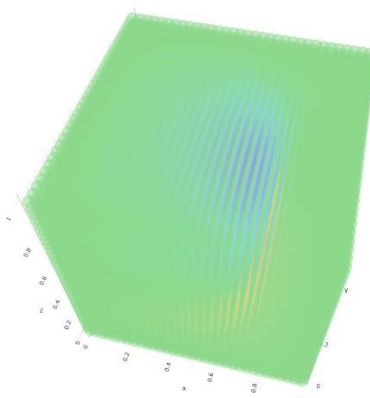


Рисунок 3 Полученное решение

Решение при  $L_x = L_y = L_z = \pi$

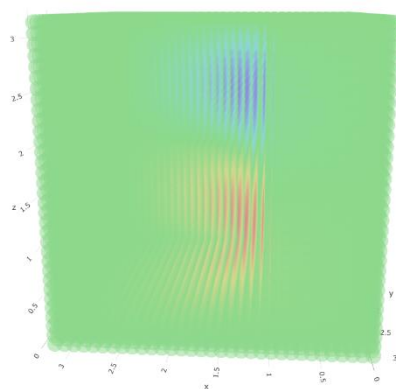


Рисунок 4 Аналитическое решение

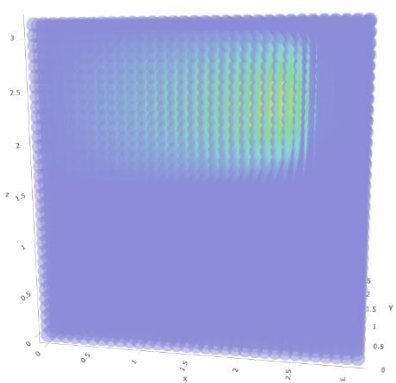


Рисунок 5 Погрешность

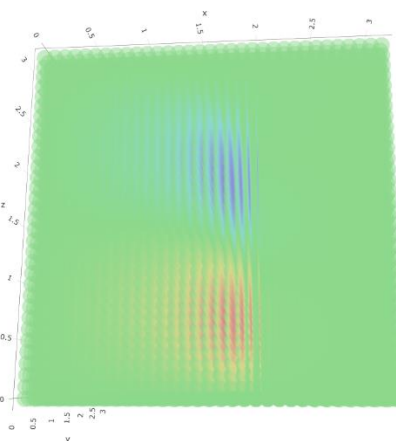


Рисунок 6 Полученное решение



# Результаты запусков программ на различных кластерах

Таблица 1. Результаты расчетов на Blue Gene/P

$L_x = L_y = L_z = 1, N = 128, K = 2000$

Число MPI-процессов	MPI			MPI + OpenMP			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
64	0.88109	1	2.7637e-08	0.664207	1	2.7637e-08	1.3265
128	0.458035	1.9236	2.7637e-08	0.357409	1.8583	2.7637e-08	1.28154
256	0.243407	3.6198	2.7637e-08	0.203291	3.2672	2.7637e-08	1.19733

$L_x = L_y = L_z = 1, N = 256, K = 2000$

Число MPI-процессов	MPI			MPI + OpenMP			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
64	6.92044	1	6.73841e-09	4.95551	1	6.73841e-09	1.3965
128	3.55399	1.9472	6.73841e-09	2.55715	1.9376	6.73841e-09	1.3898
256	1.84311	3.7547	6.73841e-09	1.33835	3.7027	6.73841e-09	1.37715

$L_x = L_y = L_z = 1, N = 512, K = 2000$

Число MPI-процессов	MPI			MPI + OpenMP			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
64	54.9187	1	1.51341e-09	38.4149	1	1.51341e-09	1.4296
128	28.0821	1.9556	1.51341e-09	19.7548	1.9426	1.51341e-09	1.4215
256	14.4007	3.9136	1.51341e-09	10.2014	3.7656	1.51341e-09	1.4116

$L_x = L_y = L_z = \pi, N = 128, K = 2000$

Число MPI-процессов	MPI			MPI + OpenMP			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
64	0.906653	1	2.82401e-09	0.687626	1	2.82401e-09	1.3185
128	0.471202	1.924	2.82401e-09	0.369249	1.862	2.82401e-09	1.276
256	0.249691	3.6311	2.82401e-09	0.209644	3.2799	2.82401e-09	1.191

$$L_x = L_y = L_z = \pi, N = 256, K = 2000$$

Число MPI- процессов	MPI			MPI + OpenMP			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
64	7.13104	1	7.0428e-10	5.14782	1	7.0428e-10	1.3852
128	3.66111	1.9477	7.0428e-10	2.65251	1.9407	7.0428e-10	1.3802
256	1.89584	3.7614	7.0428e-10	1.38544	3.71565	7.0428e-10	1.3684

$$L_x = L_y = L_z = \pi, N = 512, K = 2000$$

Число MPI- процессов	MPI			MPI + OpenMP			Ускорение
	Время решения (с)	Ускорение	Погрешность	Время решения (с)	Ускорение	Погрешность	
64	56.5957	1	1.74311e-10	39.9214	1	1.74311e-10	1.4176
128	28.9268	1.9565	1.74311e-10	20.5028	1.9471	1.74311e-10	1.4108
256	14.8168	3.8196	1.74311e-10	10.572	3.7761	1.74311e-10	1.4015

**Таблица 2. Результаты расчетов на Polus**

**$L_x = L_y = L_z = 1, N = 128, K = 2000$**

Число MPI- процессов	MPI		
	Время решения (с)	Ускорение	Погрешность
10	0.389413	1	2.7637e-08
20	0.156115	2.4943	2.7637e-08
40	0.183659	2.1203	2.7637e-08

**$L_x = L_y = L_z = 1, N = 256, K = 2000$**

Число MPI- процессов	MPI		
	Время решения (с)	Ускорение	Погрешность
10	1.23209	1	6.73841e-09
20	0.878508	1.4024	6.73841e-09
40	0.811804	1.5177	6.73841e-09

**$L_x = L_y = L_z = 1, N = 512, K = 2000$**

Число MPI- процессов	MPI		
	Время решения (с)	Ускорение	Погрешность
10	8.5587	1	1.51341e-09
20	5.54305	1.54404	1.51341e-09
40	4.44931	1.9236	1.51341e-09

**$L_x = L_y = L_z = \pi, N = 128, K = 2000$**

Число MPI- процессов	MPI		
	Время решения (с)	Ускорение	Погрешность
10	0.192021	1	2.82401e-09
20	0.137538	1.396	2.82401e-09
40	0.21486	0.885	2.82401e-09

$$L_x = L_y = L_z = \pi, N = 256, K = 2000$$

Число MPI- процессов	MPI		
	Время решения (с)	Ускорение	Погрешность
10	1.04726	1	7.0428e-10
20	0.852743	1.2281	7.0428e-10
40	0.739732	1.41572	7.0428e-10

$$L_x = L_y = L_z = \pi, N = 512, K = 2000$$

Число MPI- процессов	MPI		
	Время решения (с)	Ускорение	Погрешность
10	7.16041	1	1.74311e-10
20	4.92263	1.4545	1.74311e-10
40	3.83267	1.87314	1.74311e-10

## **Выводы**

Как следует из приведенных выше таблиц, задача для трехмерного гиперболического уравнения отлично подходит для распараллеливания. В результате получены программные средства, решающую поставленную задачу гибридным способом при использовании средств MPI и OpenMP. Важную роль в MPI при распараллеливании задачи сеточного метода играет способ разбиения на блоки. Тонким моментом, которое является «бутылочным горлышком», является передача данных и работа с памятью. При замерах на Polus использование 64 процессов было невозможно в силу технических проблем на кластере.