Отчет по второму заданию практикума, вариант № 2-2.

Димов Илья Николаевич, 620 группа

1 Формулировка задания

Рассмотрим трехмерное гиперболическое уравнение в области: $\Omega = [0 \le x \le L_x] \times [0 \le y \le L_y] \times [0 \le z \le L_z]$:

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

$$u|_{t=0} = \phi(x, y, z) \tag{2}$$

$$\frac{\partial u}{\partial t}|_{t=0} = 0 \tag{3}$$

и граничными условиями:

$$u(0, y, z, t) = 0, \qquad u(L_x, y, z, t) = 0 \tag{4}$$

$$u(x, 0, z, t) = 0, \qquad u(x, L_y, z, t) = 0 \tag{5}$$

$$u(x, y, 0, t) = u(x, y, L_z, t), \qquad u_z(x, y, 0, t) = u_z(x, y, L_z, t) \tag{6}$$

где $\phi(x, y, z) = u(x, y, z, 0)$, а u:

$$u(x, y, z, t) = sin(\frac{\pi}{L_x}x) \cdot sin(\frac{\pi}{L_y}y) \cdot sin(\frac{2\pi}{L_z}z) \cdot cos(a_t \cdot t + 2\pi), a_t = \pi\sqrt{\frac{1}{L_x^2} + \frac{1}{L_z^2} + \frac{4}{L_z^2}} \tag{7}$$

Требуется реализовать численный метод, аппроксимирующий задачу (1) - (6), с использованием технологий распараллеливания MPI и OpenMP. Разбиение замкнутой области $\overline{\Omega}$ по процессам следует сделать трехмерным. Полученную реализацию требуется исследовать на масштабируемость на платформах IBM BlueGene/P (MPI+OpenMP) и Polus (MPI).

2 Численный метод

Введем в замкнутой области $\overline{\Omega}$ равномерную сетку по пространству и времени $\omega_{h\tau} = \overline{\omega}_h \times \omega_\tau$:

$\overline{\omega}_h = \{(x_i = ih_x, y_j = jh_y, z_k = kh_z), i = 0, 1, ..., N_x - 1, j = 0, 1, ..., N_y - 1, k = 0, 1, ...N_z - 1,$
$h_x(N_x - 1) = 1, h_y(N_y - 1) = 1, h_z(N_z - 1) = 1\},$
$\omega_\tau = \{t_n = n\tau, n = 0, 1, ..., K, \tau K = T\}.$

Для аппроксимации уравнения (1) используем разностное уравнение:

$$\frac{u_{ijk}^{n+1} - 2u_{ijk}^n + u_{ijk}^{n-1}}{\tau^2} = \Delta_h u_{ijk}^n \tag{8}$$

$$1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1,\ 1 \le k \le N_z - 1,\ 1 \le n \le K - 1,$$

где

$$\Delta_h u^n_{ijk} = \frac{u^n_{i-1,j,k} - 2u^n_{i,j,k} + u^n_{i+1,j,k}}{h_x^2} + \frac{u^n_{i,j-1,k} - 2u^n_{i,j,k} + u^n_{i,j+1,k}}{h_y^2} + \frac{u^n_{i,j,k-1} - 2u^n_{i,j,k} + u^n_{i,j,k+1}}{h_z^2}.$$

С помощью (8) можно явно найти значения на (n+1)-м шаге через значения на предыдущих слоях во внутренней части сетки. Уравнение (8) аппроксимирует (1) со вторым порядком точности по пространству и времени, так как все вторые производные аппроксимируются стандартным трехточечным приближением второго порядка.

Начальное условие (2) аппроксимируется точно за счет аналитического решения.

В уравнениях (6) при вычислении разностного аналога оператора Лапласа считается, что индекс k зациклен, то есть: $z^n_{i,j,-1} = z^n_{i,j,N_z-2}$.

3 Программная реализация

Реализация схемы была выполнена на языке $C++$ с использованием технологий MPI и OpenMP.

Для распределения рассматриваемой области по процессам было реализовано трехмерное разбиение области на блоки-параллелепипеды. Разбиение по процессам реализовано следующим образом: число блоков по осям x, y, z последовательно домножается на множители в разложении числа доступных процессов, после чего число точек пропорционально распределяется по процессам. Таким образом 8 процессов дадут число блоков (2, 2, 2), 16 процессов дадут разбиение (4, 2, 2), а 10 процессов (2, 5, 1). Нумерация блоков идет по направлениям XYZ.

Т.к. на Polus предлагается использовать те же сетки $128^3, 256^3, 512^3$, но число процессов (10, 20, 40) не является делителем числа точек, предусмотрено разбиение на неодномерные блоки за счет увеличения размера первых блоков. Рассмотрим на примере: пусть дана сетка $128 \times 128 \times 128$ и число процессов 5. Тогда по осям XYZ число блоков будет (5, 1, 1). По YZ каждый (единственный) блок будет иметь 128 точек, а по оси X блоки будут размеров (26, 26, 26, 25, 25).

При нахождении значений на слое (n+1) для вычисления значения оператора $\Delta_h u^n$ на границе рассматриваемого блока требуются значения сеточной функции u на слое (n) из соседних 6 блоков. Пересылка требуемых значений проводится с помощью неблокируемых функций MPI_Isend(), MPI_Irecv(): сначала инициализируются приемы и посылки на 6 границах блока по осям x,y,z, затем проводятся подсчеты значений на шаге (n+1) во внутренней части блока, и только потом ставятся функции ожидания приема требуемых значений MPI_Wait() и подсчеты на границах блока. Таким образом, на достаточно густых сетках передача требуемых значений заканчивается раньше подсчетов во внутренней части области, и MPI-процессы не простаивают в ожидании конца пересылок. Также все двойные и тройные циклы в части кода, отвечающей за подсчет значений на новом шаге по времени, распараллелены с помощью директивы #pragma omp parallel for. Для подсчета значений на 0 и 1 шаге по времени какие-либо пересылки не нужны, ведь там решение получается аналитически. Стоит отметить, что при числе процессов 1 соответствующее число блоков тоже 1. В данном случае по оси z осуществляются посылки от процесса себе, а по остальным осям пересылок не происходит.

Для верификации правильности работы программы на каждом шаге считается максимальная L1-норма между расстояниями в узлах сетки и аналитическим решением на соответствующем шаге по времени и пространстве. Подсчет ошибки проводился в 2 этапа. Сначала

на каждом процессе находился максимум модуля отклонения от аналитического решения на блоке, соответствующем данному процессу. После этого в процессе 0 собираются ошибки с помощью функции MPI_Gather(), из них вычисляется максимальная.

Сетка хранится ввиде массива double***. Скорее всего программу можно было бы ускорить, если использовать сетку double*, т.к. тогда данные хранились бы последовательно, но было принято использовать массив double*** ради упрощенной схемы индексации.

4 Запуски на суперкомпьютерах

4.1 Параметры задачи и численного метода

Для тестовых запусков написанной реализации метода была взята задача в кубе $\Omega = [0 < x < 1] \times [0 < y < 1] \times [0 < z < 1]$ и кубе $\Omega = [0 < x < \pi] \times [0 < y < \pi] \times [0 < z < \pi]$. По времени делалось 20 шагов. T и τ рассчитывались для устойчивости из равенства: $\tau = min(h_x, h_y, h_z)/2, T = 20 * \tau$

4.2 Компиляция и запуск программ

На платформе Bluegene/P компиляция и запуск задачи происходили следующим образом:

```
mpixlcxx_r -qsmp=omp ....
mpisubmit.bg -n 64 -w 00:10:00 -e "OMP_NUM_THREADS=2" bin/bluegene 1 1 1 256 256 256 \
--stdout no_omp_out_proc_64_grid_256.txt
```

На платформе Polus:

```
module load SpectrumMPI
mpixlC ...
mpisubmit.pl -p 20 -w 00:05 --stdout 1_20_128.txt bin/polus -- 1 1 1 128 128 128
```

Более подробные параметры компиляции можно посмотреть в Makefile, все параметры запуска можно посмотреть в файлах run_bluegene.sh, run_polus.sh. При компиляции использовался уровень оптимизации по-умолчанию. Запуски происходили в режиме SMP, блоки в программе нумеруются в порядке XYZ.

4.3 Результаты расчетов

Ниже приведены таблицы с результатами расчетов на суперкомпьютерах Bluegene/P и Polus. Ускорение считалось как отношение времени работы ко времени работы реализации на наименьшем числе процессов без использования OMP (с 1 нитью) в рамках данной сетки. Программы с использованием OMP работали с 4мя нитями. Дополнительно стоит отметить, что запуски на системе Polus давали большой разброс по времени, поэтому время работы там является усреднением по 3м запускам.

Ошибка не меняется от числа процессов в рамках одной сетки. На сетке 128^3 ошибка составляет $13 \cdot 10^{-6}$, на сетке 256^2 - 10^{-6}, на сетке 512^3 - 0 (меньше чем вмещает double). Предложенная программная реализация тестировалась в областях единичной длины и длины π. На обеих вариантах работы получились одинаковые погрешности и близкие времена исполнения, т.к. время исполнения зависит от размеров сетки, а не численных параметров задачи.

Число процессоров N_p	Число точек разбиения N^3	Время работы (с)	Ускорение S	Время работы с OMP (с)	Ускорение с OMP
64	128^3	1.49	1	0.39	3.82
128	128^3	0.75	1.98	0.20	7.4
256	128^3	0.38	3.92	0.11	13.54
512	128^3	0.20	7.44	0.06	24.83
64	256^3	11.91	1	3.07	3.87
128	256^3	5.99	1.98	1.57	7.58
256	256^3	3.00	3.97	0.79	15.07
512	256^3	1.51	7.88	0.40	29.775
64	512^3	95.41	1	24.41	3.9
128	512^3	47.81	1.99	12.29	7.76
256	512^3	23.88	3.99	6.14	15.53
512	512^3	11.97	7.97	3.08	30.9

Таблица 1: Результаты расчетов на суперкомпьютере "Bluegene/P".

Число процессоров N_p	Число точек разбиения N^3	Время работы T (с)	Ускорение S
10	128^3	2.55	1
20	128^3	1.31	1.94
40	128^3	0.65	3.92
10	256^3	15.21	1
20	256^3	8.44	1.75
40	256^3	4.85	3.04
10	512^3	95.95	1
20	512^3	50.99	1.88
40	512^3	25.52	3.75

Таблица 2: Результаты расчетов на суперкомпьютере Polus

Также приведен график погрешности на решении задачи в решетке $\Omega = [0 \le x \le \pi] \times [0 \le y \le \pi] \times [0 \le z \le \pi]$. На данном графике отображена зависимость погрешности от переменной z при x $= \frac{\pi}{2}$ и y $= \frac{\pi}{2}$.

Приведен график погрешности на сетке 128^3 в зависимости от шага для алгоритма. На больших сетках ошибка меньше точности класса double, поэтому графики не приводятся.

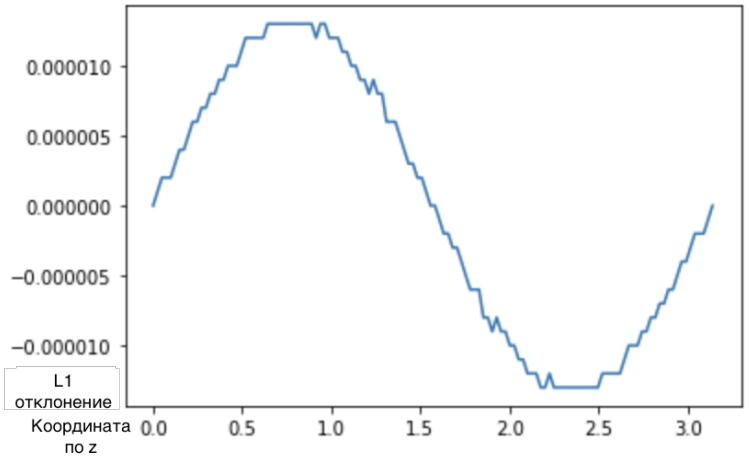

Рис. 1: График ошибки вдоль прямой x $= \frac{\pi}{2}, y = \frac{\pi}{2}$ после 20 шага.

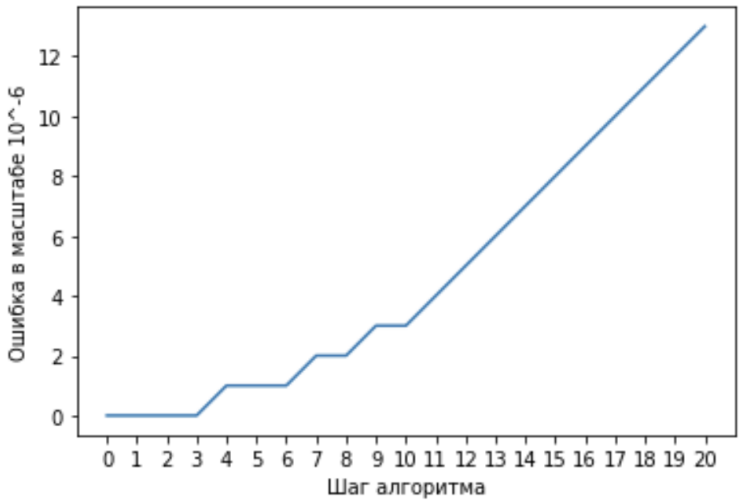

Рис. 2: График абсолютной ошибки в зависимости от шага алгоритма на сетке 128^3.

5 Выводы

По таблицам видно, что алгоритм обладает ускорением близким к линейному по числу MPI-процессов. На системе Polus данная закономерность просматривается на сетках отличных от 256. Несмотря на усреднение по нескольким запускам получить на данной сетке ускорение близкое к линейному не удалось.

Дополнительно стоит отметить, что линейное ускорение может не достигаться из-за блокирующей операции MPI_Gather, которая используется для подсчета ошибки.

Ускорение, связанное с использованием OMP, также почти линейное. Чем больше сетка, тем больше ускорение, из-за того, что организация разбиения цикла по времени становится много меньше, чем само вычисление; однако при увеличении числа процессов сетка уменьшается, что дает обратный результат.