Отчет о выполненном практическом задании по курсу “Суперкомпьютерное моделирование и технологии”

Студентки 627 группы

Рагозиной Полины

(3 поток)

Постановка задачи

В трехмерной замкнутой области $\Omega = [0 \le x \le Lx] \times [0 \le y \le Ly] \times [0 \le z \le Lz]$ 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 $u_{t=0} = \varphi(x, y, z)$, $\frac{\partial u}{\partial x}_{t=0} = 0$, при условии, что на границах области заданы следующие граничные условия:

$u(0, y, z, t) = 0,$

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

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

$u_y(x, 0, z, t) = u_y\big(x, L_y, z, t\big),$

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

$u_z(x, y, 0, t) = u_z(x, y, L_z, t).$

Численное решение

На Ω вводим сетку $\omega_{h\tau} = \overline{\omega}_h \times \omega_\tau$, где

$$T = T_0,$$

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

$\overline{\omega}_h = \big\{\big(x_i = ih_x, y_j = jh_y, z_k = kh_z\big), i, j, k = 0,1, \dots, N, h_x N = L_x, h_y N = L_y, h_z N = L_z\big\}, \omega_\tau = \{t_n = n\tau,\ n = 0,1, \dots, K, \tau K = T\}.$

Через ω_h обозначим множество внутренних узлов решётки, а через γ_h - граничных.

Аппроксимация - через систему уравнений:

$\frac{u_{ijk}^{n+1} - 2u_{ijk}^n + u_{ijk}^{n-1}}{\tau^2} = \Delta_h u^n,\ \big(x_i, y_j, z_k\big) \in w_h,\ n = 1,2, \dots K - 1,$

где

$\Delta_h u^n = \frac{u^n_{i-1,j,k} - 2u^n_{i,j,k} + u^n_{i+1,j,k}}{h_x^2} + \frac{u^n_{i,j-1,k} - 2u^n_{i,j,k} + u^n_{i,j+1,k}}{h_y^2} + \frac{u^n_{i,j,k-1} - 2u^n_{i,j,k} + u^n_{i,j,k+1}}{h_z^2}$Зная u^0_{ijk} и u^1_{ijk}, можно посчитать аппроксимированные значения остальных u^n_{ijk}. Получим их так:

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

Последовательная реализация

Последовательная программа реализована очевидным образом. Производится итеративный проход в виде 4 вложенных циклов по матрице чисел с плавающей точкой$K \times N^3$, после каждой итерации поиск максимальной погрешности (эти значения хранятся в массиве длиной K). Так как вычисление значения в каждой ячейке использует значения предыдущих слоёв, все вычисления во время одной итерации независимы друг от друга.

На дальнейших этапах было решено отказаться от матрицы $K \times L^3$ в пользу матрицы $3 \times L^3$, так как в процессе вычисления в каждый момент времени требуется только текущий слой и два предыдущих. В момент t текущий слой имеет индекс t%3, а предыдущие, соответственно, (t-1)%3 и (t-2)%3. В конце каждой итерации слои "смещаются".

OpenMP

Параллельная OpenMP-программа получена простым добавлением omp-прагм: `#pragma omp for collapse(3)` в цикле обновления массива и omp-редукцией `#pragma omp for collapse(3) reduction(max:my_max)` при поиске максимальной погрешности.

MPI

Трёхмерная матрица на каждой итерации для удобства последующего добавления CUDA представлена в одномерном виде. В этой части работы была использована топология "3D решётка", таким образом если процессы составляют решётку $a \times b \times c$, то каждый процесс работает с частью решётки задачи размером $\frac{N}{a} \times \frac{N}{b} \times \frac{N}{c}$. Однако, чтобы избежать неоднозначности адресации, было решено создавать матрицу размером $3 \times N_{max}^3$, где N_{max}максимальное из значений $\frac{N}{a}, \frac{N}{b}, \frac{N}{c}$. Адресация при этом выполняется так: $w[t][x][y][z] = arr[t\%3][z + y \cdot N_{max} + x \cdot N_{max} \cdot N_{max}]$. Помимо этого, на каждом процессе также создаётся 36 массивов-буферов размером N_{max}^2 - по два массива на отправку и приём данных в направлениях "от меньшего номера в топологии к большему" и "от большего к меньшему" в каждом из трёх измерений для каждой из трёх матриц. При добавлении нового значения в граничную ячейку решётки такое же значение добавляется в соответствующий буфер передачи. Затем на каждом этапе производится асинхронный обмен. Данные из буфера получения затем используются при подсчёте значений на новой итерации.

Кроме того, на каждой итерации каждый процесс считает наибольшую погрешность и из них выбирается наибольшая с помощью MPI_Reduce.

MPI+OpenMP

Гибридная программа получена добавлением omp-прагм к циклам заполнения ячеек и поиска погрешности.

MPI+CUDA

К MPI-версии также добавлена работа с GPU. Вместо прохода по циклу значения на каждой итерации хранятся и высчитываются на устройстве. На каждой итерации на хост передаются только полученные значения буферов отправки, а после асинхронного mpi-обмена данными хост передаёт устройству значения буферов приёма.

Максимальная разность на каждой итерации находится с помощью атомарной операции сравнения и MPI_Reduce.

Таблица результатов