Задание по курсу
Суперкомпьютерное моделирование и технологии

выполнил:
Артамонов Иван 607 группа

Москва 2024

Содержание

1 Введение

Требуется методом конечных разностей приближенно решить задачу Дирихле для уравнения Пуассона в прямоугольной области области. Задание необходимо выполнить на следующих ПВС Московского университета:

1. IBM Polus

Мой вариант задания 7, используется равномерная сетка и максимум-норма.

2 Математическая постановка дифференциальной задачи

В квадратной области

$$D = (x, y) : -1 < x, y < 1 \setminus (x, y) : 0 < x, y < 1$$

требуется найти функцию $u = u(x, y)$, удовлетворяющую дифференциальному уравнению

$$-\Delta u = 1, (x, y) \in D$$

и дополнительному условию

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

во всех граничных точках D

3 Разностная схема решения задачи

Для решения задачи уравнение Пуассона во внутренних точках аппроксимируется методом конечных разностей. Для решения полученной системы линейных алгебраических уравнений используются

метод скорейшего спуска. Подробные формулы приведены в описании задачи; здесь отмечу только, что в моем варианте задания используется максимум-норма, то есть алгоритм заканчивает работу, если

$$||w^{(n)} - w^{(n-1}|| < \varepsilon$$

где

$$||w|| = \max_{\substack{0<=i<=M \\ 0<=j<=N}} |w(x_i, y_j)|$$

Обозначения соответствуют принятым в описании задания. ε принят равным 0.0001.

4 Описание программы

4.1 Использование OpenMP

Ряд действий в рамках одного процесса, такие как подсчет пятиточечной аппроксимации внутри прямоугольника, присвоенного процессору, или обновление матрицы решения после вычисления всех необходимых коэффициентов в обоих методах, могут быть эффективно распараллелены в помощью нитей OpenMP. Практически везде это реализуется добавлением директивы
`#pragma omp parallel for` Еще одна используемая директива – `reduction`, она нужна для эффективного параллельного вычисления скалярного произведения.

4.2 Использование MPI

Производится разбиение на домены, каждый домен рассчитывается на отдельном процессе.

Использованы асинхронные директивы `Irecv Isend` для пересылки информации, `Gather` для сбора максимальной погрешности

на каждой итерации.

4.3 Использование MPI+OpenMP

Код MPI-программы дополнен OMP директивами, аналогично пункту "Использование MPI"выше.

5 Результаты расчетов

5.1 Проверка последовательной и параллельной версии

Были проверены результаты расчетов последовательной и параллельной версии с сохранением в файл, с дальнейшим запуском файла, считывающего результаты. Проведено сравнение на сетке 40x40 для последовательной и параллельных версий.

Отличий не обнаружено, максимум отличие двух решений двух версий для 1, 4, 16 нитей равен машинному нулю.

Это может быть объясненно тем, в частности, что в данной ситуации нет зависимости результата от порядка вычислений, каждая нить вычисляет произведение своей пары: строчка столбец.

В случае, если бы вычислялась некая общая характеристика, могла бы быть зависимость от порядка суммирования. Но не для случая максимума, как в данной программе, а для случая суммы.

5.2 Сравнение времени вычислений последовательной и параллельной версии

Таблица 1: Время вычислений на Polus

Число OpenMP-нитей	**Размер сетки**	**Время решения, сек**	**Ускорение**
1	80×90	1787	
2	80×90	962	1.85
4	80×90	617	2.89
8	80×90	455	3.92
16	80×90	241	7.41
32	80×90	171	10.45
1	160×180	23	
2	160×180	11	2.0
4	160×180	6	3.8
8	160×180	3.2	7.1
16	160×180	2.1	10.9
32	160×180	1.6	14.3

Таблица 2: Время вычислений на Polus

MPI-processes	**OMP-threads**	**Размер сетки**	Число итераций	**[ms]**	**Ускорение**
1	1	80×90	100000	9651	1
2	1	80×90	100000	5812	1.66
2	2	80×90	100000	4324	2.33
2	4	80×90	100000	3939	2.45
2	8	80×90	100000	4200	2.30
1	1	160×180	100000	36611	1
4	1	160×180	100000	10721	3.41
4	2	160×180	100000	7360	4.97
4	4	160×180	100000	6100	6.00
4	8	160×180	100000	6280	5.83

6 График решения на максимальной сетке

График адекватен постановки задачи

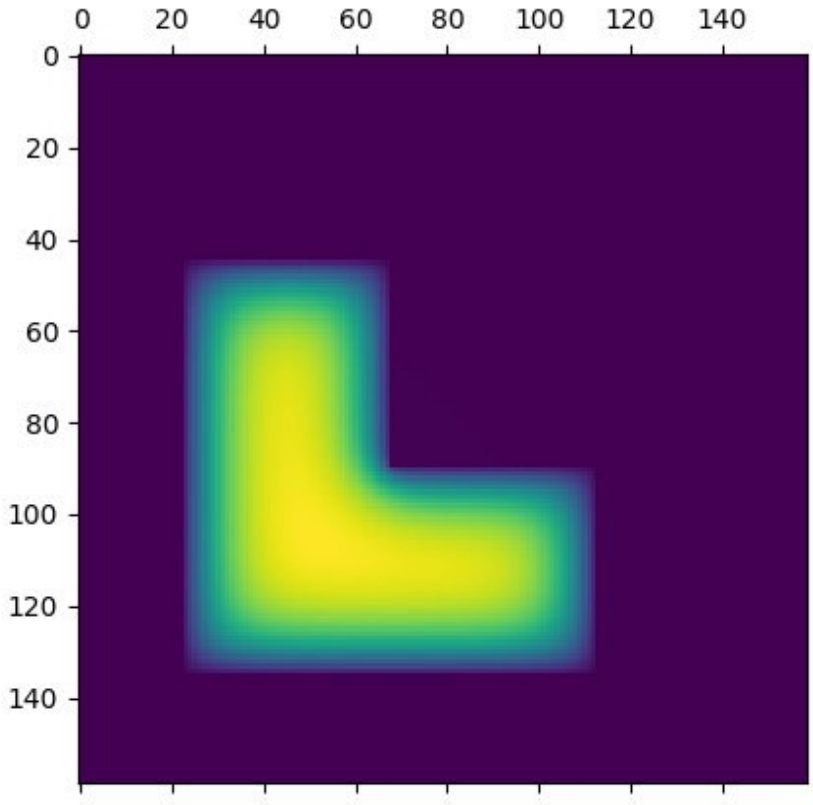

Рис. 1: Enter Caption

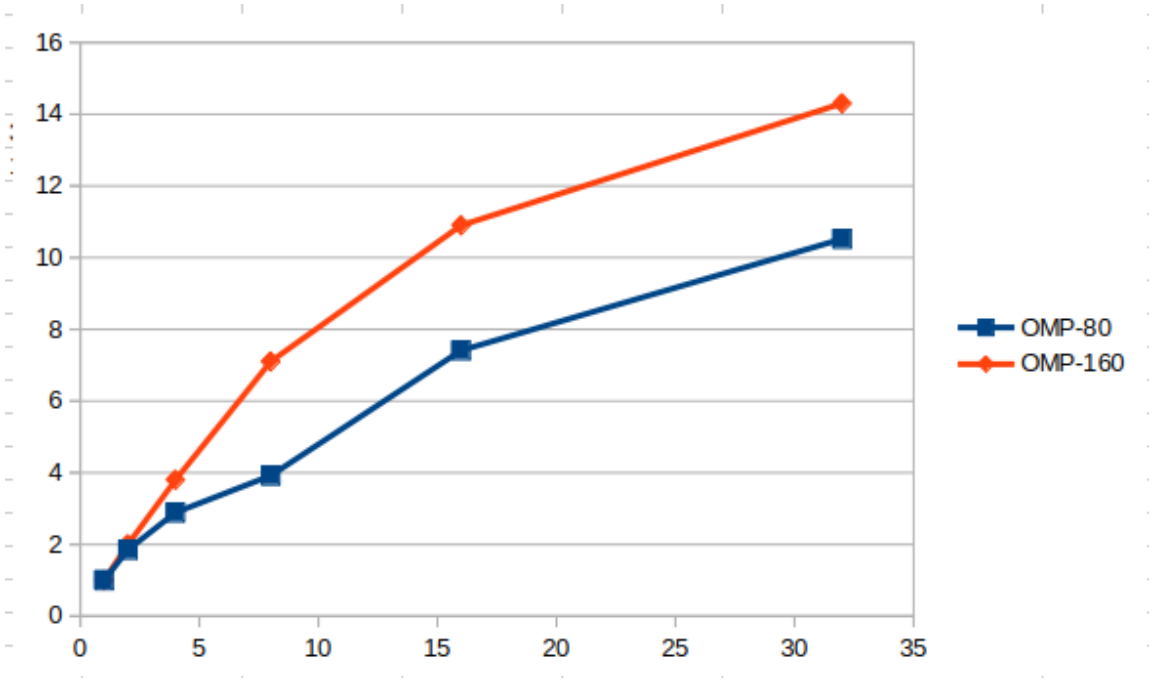

Рис. 2: Графики ускорений OMP

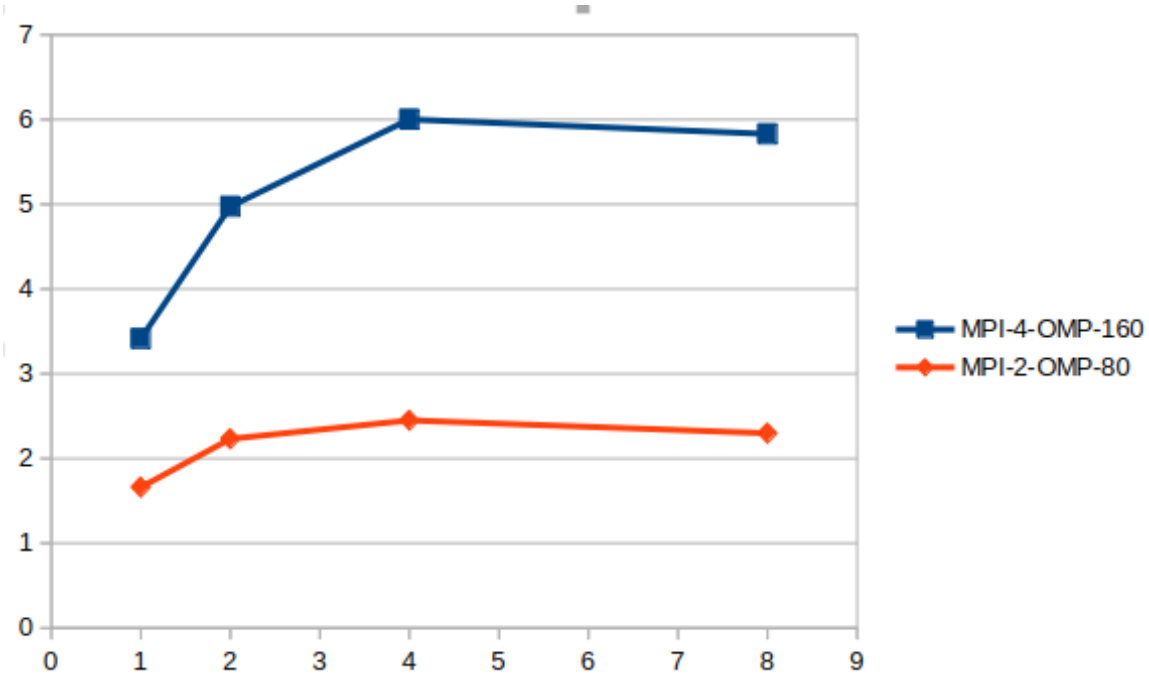

Рис. 3: Графики ускорений MPI+OpenMP