Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики

Практикум по курсу «Суперкомпьютеры и параллельная обработка данных»

Разработка параллельной версии программы для релаксации трехмерной матрицы

Работу выполнил
Федоров В. В.
Студент 323 группы
факультета ВМК МГУ

Москва 2020

Постановка задачи и формат данных.

Задача: Реализовать параллельный алгоритм релаксации трехмерной матрицы
Формат ввода: На вход подается единственное число — размер трехмерной матрицы.

Описание алгоритма.

Математическая постановка: Трехмерная матрица размером NxNxN с нулями на границах обрабатывается следующим образом: её элементы обходятся по выбранной координате от 1 до N-2 (нумерация элементов матрицы начинается с нуля), и каждому присваивается среднее из двух его соседей по этой координате. Так выглядит подобный цикл по координате i:

```
for (size_t k = 1; k < n - 1; k++)
   for (size_t j = 1; j < n - 1; j++)
      for (size_t i = 1; i < n - 1; i++)
      {
         A[i][j][k] = (A[i - 1][j][k] + A[i + 1][j][k]) / 2.;
      }
```

Аналогично по координатам j и k. В цикле по координате k считается максимальная разница между исходными и новыми значениями элементов. Данная процедура выполняется до тех пор, пока не выполнится заданное число итераций либо максимальная разность не окажется меньше заданного эпсилон.
Параллельная оптимизация: Для OpenMP оптимизация заключалась лишь в добавлении прагм. Так как в алгоритме присутствует зависимость по выбранной координате, распараллелить можно только два вложенных цикла из трёх.
Анализ времени выполнения: Для оценки времени выполнения программы использовалась функция: `omp_get_wtime()`. Для повышения надёжности экспериментов опыты проводились несколько раз (50).
Верификация: Для проверки корректности работы программы использовалась функция `verify()`, значение которой сравнивалось с её же значением для результата работы оригинального последовательного алгоритма.
Основные функции:

- `int main(...);`
  - В рамках функции осуществляется выделение памяти под матрицу, вызов остальных функций и замер времени.
- `int init(...);`
  - Инициализация элементов матрицы по формуле:
    - A[i][j][k] = 0, если одна из трех координат равна 0 или N — 1;
    - A[i][j][k] = 4 + i + j + k в противном случае.
- `int relax(...);`
  - Функция выполняет одну итерацию последовательного алгоритма релаксации.
- `int relax_parallel(...);`
  - Функция выполняет одну итерацию параллельного алгоритма релаксации.
- `int verify(...);`
  - Функция вычисляет хэш-сумму элементов матрицы по формуле:

$$S = \sum_{i,j,k=0}^{N-1} ((i+1)(j+1)(k+1)A_{ijk}) / N^3$$

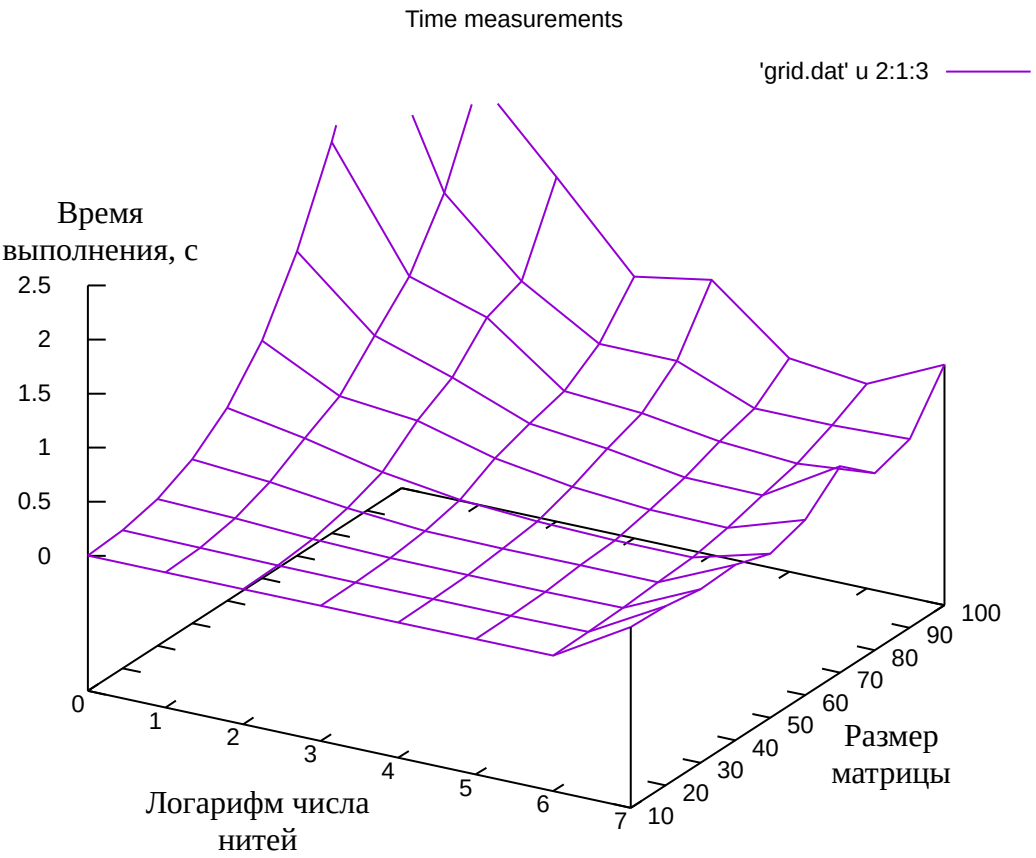
Полностью код можно посмотреть в репозитории на Гитхабе:
https://github.com/P34K1N/skpod

Результаты выполнения.

OpenMP:

Тестирование программы проводилось на системе Polus. Обрабатывались матрицы размеров от 10 до 100 с шагом в 10, число нитей было равно степеням двойки от 1 до 128.

	1	2	4	8	16	32	64	128
10	0.001566	0.001286	0.00124	0.001466	0.001741	0.002546	0.005842	0.423769
20	0.029118	0.018702	0.011385	0.007935	0.008573	0.009965	0.016755	0.410791
30	0.108465	0.08561	0.045735	0.026864	0.024792	0.023147	0.028769	0.357154
40	0.266909	0.212207	0.123911	0.068978	0.05944	0.062319	0.059351	0.375451
50	0.536823	0.409943	0.249276	0.148236	0.102929	0.082376	0.079272	0.268038
60	0.948671	0.591431	0.518568	0.324486	0.214459	0.15641	0.145369	0.371398
70	1.5662	0.943533	0.710464	0.437908	0.359163	0.246163	0.236305	0.661334
80	2.36902	1.28181	1.05514	0.531461	0.480476	0.374889	0.322535	0.388142
90	3.42805	1.84446	1.18439	0.760061	0.756162	0.472098	0.471712	0.496052
100	4.74889	2.67547	1.93661	1.17189	1.29627	0.728205	0.644418	0.976898



Основные выводы.

OpenMP-версия программы при малых значениях N не ускоряется значительно при увеличении числа нитей, а порой даже и замедляется. Только при увеличении N можно заметить значительные улучшения времени работы программы, особенно при малом числе нитей, где время выполнения уменьшается почти пропорционально росту их числа. В целом зависимость объема накладных расходов на одну нить от числа нитей можно выделить на основе графика и объяснить особенностями архитектуры системы Polus:

- 1-8 нитей — программа обрабатывается одним ядром, накладные расходы минимальные;
- 16-64 нитей — программа обрабатывается несколькими ядрами в пределах одного процессора, накладные расходы средние;
- 128 нитей — программа обрабатывается двумя процессорами, накладные расходы большие.