Отчёт по лабораторной работе

Задание: A-03 (OpenMP)
Выполнил: Муравьев Илья Владимирович

Условие

Написать программу вычисления матричного выражения:

$\mathbf{A} = \mathbf{B}\ \mathbf{C}^2 + M(\mathbf{C})\ \mathbf{I} + \mathbf{I} + D(\mathbf{B})\ \mathbf{E}$,

где $\mathbf{B}$, $\mathbf{C}$ – квадратные плотные матрицы, элементы которых имеют тип double, причем элементы матрицы $\mathbf{C}$ задаются с помощью генератора псевдослучайных чисел, $\mathbf{I}$ – единичная матрица, $\mathbf{E}$ – полностью заполненная матрица, все элементы которой равны единице, $M(\mathbf{C})$ – среднее значений элементов матрицы $\mathbf{C}$, $D(\mathbf{C})$ – дисперсия элементов матрицы $\mathbf{C}$. Распараллелить эту программу с помощью OpenMP (parallel, task). Исследовать зависимость масштабируемости параллельной версии программы от ее вычислительной трудоемкости (размера матриц).
Проверить корректность параллельной версии.
Проверка закона Амдала. Построить зависимость ускорение:число потоков для заданного примера.

Программно-аппаратная конфигурация тестового стенда

```
OS: Ubuntu 22.04.4 LTS
GCC Version: gcc (GCC) 10.2.0
CPU Model: Intel(R) Xeon(R) E-2136 CPU @ 3.30GHz
Logical Cores: 12
Physical Cores: 6
RAM: 64G
```

Метод

Алгоритм решения:

- вычисление $M(\mathbf{C})$;
- вычисление $M(\mathbf{D})$;

- вычисление $\mathbf{C}^2$;
- вычисление $\mathbf{B}\ \mathbf{C}^2$;
- прибавление $1 + M(\mathbf{C})$ к каждому элементу главной диагонали $\mathbf{B}\ \mathbf{C}^2$;
- прибавление $D(\mathbf{B})$ к каждому элементу полученной на предыдущем шаге матрицы.

Для вычисления произведения матриц используется наивный алгоритм, основанный на определении произведения матриц $(\mathbf{A}\ \ \mathbf{B})_{i,j} = \sum_{k=1}^{n} \mathbf{A}_{i,k}\ \ \mathbf{B}^T_{j,k}$. Правый множитель транспонируется перед вычислением произведения, чтобы эффективнее использовать кэши процессора.

Реализация

Решение реализовано на языке программирования C. При выполнении команды `make` собираются два исполняемых файла:

- `main` – решает поставленную задачу;
- `save_random_matrix` – генерирует случайные квадратные матрицы указанного размера.

При вычислении каждой матричной операции внешний цикл распараллеливается с помощью директивы OpenMP `parallel for`.

Реализация доступна в GitHub-репозитории: https://github.com/IlyaMuravjov/openmp-task-a03.

Данный отчёт сгенерирован из файла Python Notebook: https://github.com/IlyaMuravjov/openmp-task-a03/blob/main/experiment.ipynb.

Эксперимент

В данном разделе приведены результаты экспериментов, призванных ответить на три исследовательских вопроса:

1. Корректно ли разработанная реализация работает в параллельном режиме?
2. Как разработанная реализация масштабируется при увеличении размера матриц?
3. Выполняется ли для разработанной реализации закон Амдала?

Корректность

Результат запуска скрипта проверки соответствия результатов, полученных вручную, и результатов, полученных последовательной версией, для матрицы размера 2×2.

```
Seqential version results match manually calculated results
```

Результат запуска скрипта проверки соответствия результатов, полученных последовательной и параллельной версиями, для матрицы размера 3000×3000.

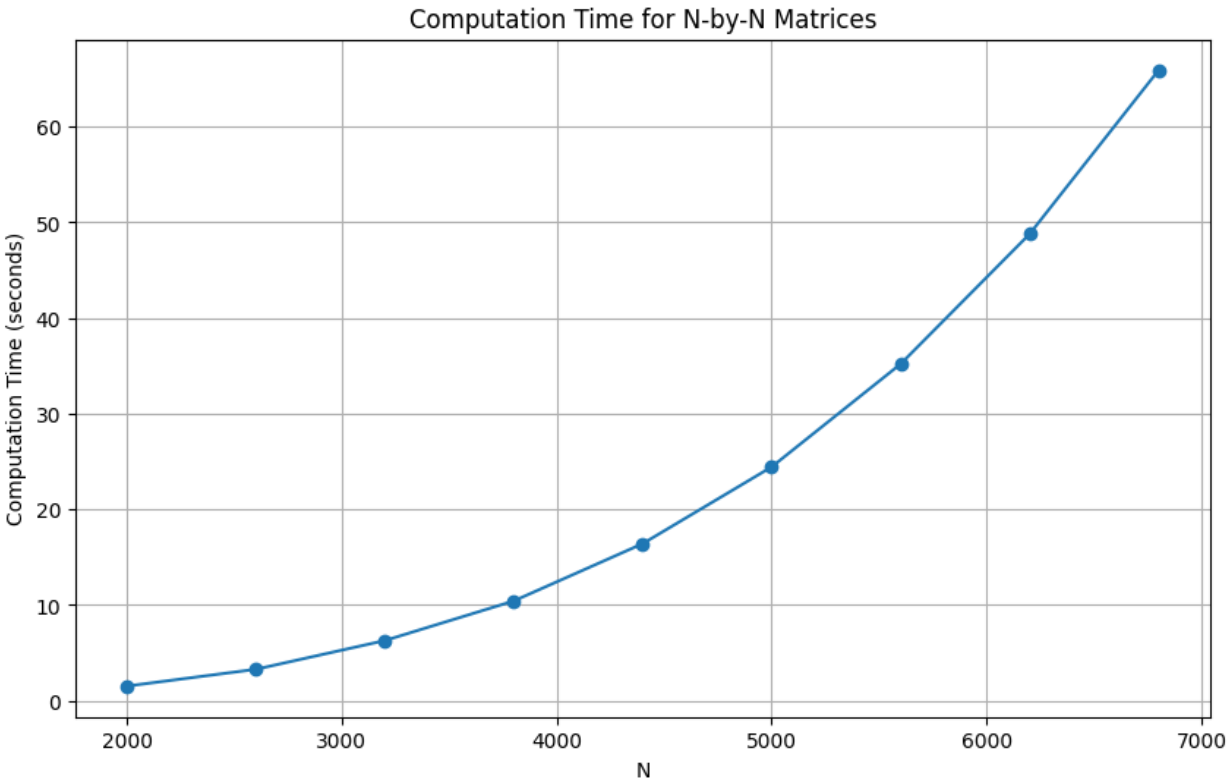
```
Parallel version results match sequential version results
```

Масштабируемость

Здесь и далее время работы указывается без учёта времени генерации входных данных и времени, затраченного на операции ввода-вывода.

Результаты запуска скрипта оценки масштабируемости параллельной версии программы от её вычислительной трудоемкости (размера матриц) для матриц размера $N \times N$.

N	Time (seconds)
2000	1.52
2600	3.27
3200	6.28
3800	10.42
4400	16.41
5000	24.43
5600	35.19
6200	48.77
6800	65.89



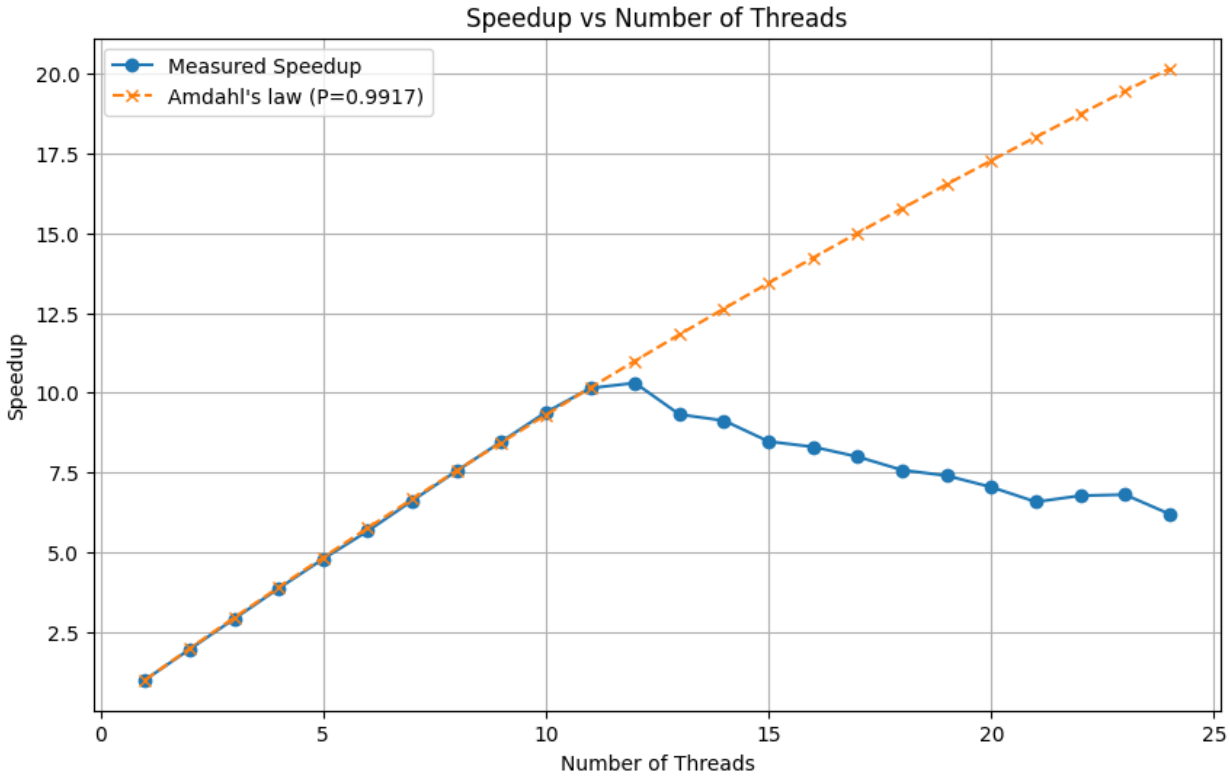

```
Slope of a log-log plot: 3.08
```

Проверка закона Амдала

Результаты запуска скрипта проверки закона Амдала для матрицы размера 3000×3000.

Number of Threads	Measured Speedup	Amdahl's law (P=0.9917)
1	1	1
2	1.96	1.98
3	2.91	2.95
4	3.87	3.9
5	4.79	4.84
6	5.66	5.76
7	6.62	6.67
8	7.55	7.56
9	8.47	8.44
10	9.39	9.3
11	10.14	10.16
12	10.3	11

Number of Threads	Measured Speedup	Amdahl's law (P=0.9917)
13	9.32	11.82
14	9.12	12.64
15	8.47	13.44
16	8.3	14.23
17	7.99	15.01
18	7.58	15.77
19	7.4	16.53
20	7.04	17.27
21	6.58	18.01
22	6.77	18.73
23	6.81	19.45
24	6.2	20.15



Выводы

Эмпирическим путём установлено, что:

- параллельная реализация работает корректно на рассмотренном

примере;
- время работы параллельной реализации растёт примерно пропорционально $N^{3.08}$;
- закон Амдала выполняется для $P = 0.9917$, когда число потоков меньше 12.

Объяснение расхождений с теорией:

- расхождение с теоретической оценкой сложности $\mathcal{O}(N^3)$ объясняется конечностью размера кэшей-процессора: с ростом размера матрицы всё меньшая часть матрицы помещается в кэши, что приводит к замедлению реализации;
- невыполнение закона Амдала при использовании более чем 12 потоков объяснятся тем, что используемый процессор имеет 12 логических потоков;
- небольшое отклонение от закона Амдала при 12 потоках объясняется тем, что часть процессорного времени тратится на выполнение фоновых задач.