

Московский государственный университет  
имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

## Отчет по практическому заданию

Вариант 20

Выполнил:  
Студент гр. 321  
Астраханцев Дмитрий Андреевич

Москва, 2023

# Содержание

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Введение</b>                                     | <b>2</b> |
| 1.1      | Постановка задачи . . . . .                         | 2        |
| 1.2      | Используемые алгоритмы . . . . .                    | 2        |
| <b>2</b> | <b>Сделанные модификации</b>                        | <b>3</b> |
| 2.1      | Оптимизации без использования OpenMP . . . . .      | 3        |
| 2.2      | Оптимизации с использованием OpenMP . . . . .       | 3        |
| 2.3      | Алгоритм RedBlack3D . . . . .                       | 3        |
| 2.4      | Распараллеливание по гиперплоскостям куба . . . . . | 3        |
| <b>3</b> | <b>Результаты работы</b>                            | <b>4</b> |
| <b>4</b> | <b>Выводы</b>                                       | <b>6</b> |

# 1 Введение

## 1.1 Постановка задачи

Реализовать параллельную версию предложенной программы на языке С с использованием OpenMP для различных входных данных и протестировать работоспособность на различном числе процессоров на суперкомпьютере Polus. Под размером входных данных будем подразумевать число  $N$ , равное размерам трехмерного массива  $A$ .

## 1.2 Используемые алгоритмы

Данную задачу отличает регулярная зависимость по данным. Данный класс задач плохо поддается распараллеливанию, по этой причине были использованы особые алгоритмы, которые нацелены именно на такие задачи:

- RedBlack 3D
- Распараллеливание по гиперплоскостям куба

## 2 Сделанные модификации

### 2.1 Оптимизации без использования OpenMP

Порядок переменных в циклах был заменен с  $k, j, i$  на  $i, j, k$ . Это ускорило работу программы на 25% за счет оптимизации доступа к памяти.

### 2.2 Оптимизации с использованием OpenMP

Т.к. функции `void init(void)` и `void verify(void)` не имеют регулярной зависимости по данным, то в них были добавлены строки

```
1 #pragma omp parallel for collapse(3) default(none) private(i, j, k) shared  
  (A)
```

и

```
1 #pragma omp parallel for collapse(3) default(none) private(i, j, k) shared  
  (A) reduction(+:s)
```

соответственно.

### 2.3 Алгоритм RedBlack3D

Модифицируем алгоритм RedBlack2D для трехмерного случая, высчитывая четность суммы индексов  $(i + j + k) \% 2$  и разбивая вычисления на 2 подцикла. Подробнее реализацию можно посмотреть в файле `openmp_redblack.c`.

### 2.4 Распараллеливание по гиперплоскостям куба

Для данной задачи была написана функция, вычисляющая индексы ячеек, принадлежащих каждой из гиперплоскостей. Данная функция хорошо распараллеливается, однако из-за общей структуры данных она имеет критическую секцию, что замедляет ее работу. Также была переработана функция `relax` так, чтобы обход шел не по индексам, а по элементам гиперплоскости. Подробнее реализацию можно посмотреть в файле `openmp_hyperplain.c`.

### 3 Результаты работы

Результатом работы является тестирование, разработанных программ на суперкомпьютере Polus. Для тестирования было выбрано количество процессоров равное 1, 2, 8, 20, 100 и входные данные  $N=\{200, 500, 700, 1000, 1200\}$ . Из-за ограничения в 15 минут (900 секунд), для некоторых параметров значения так и не были получены. Эти параметры обозначены темно-синим цветом на диаграмме.

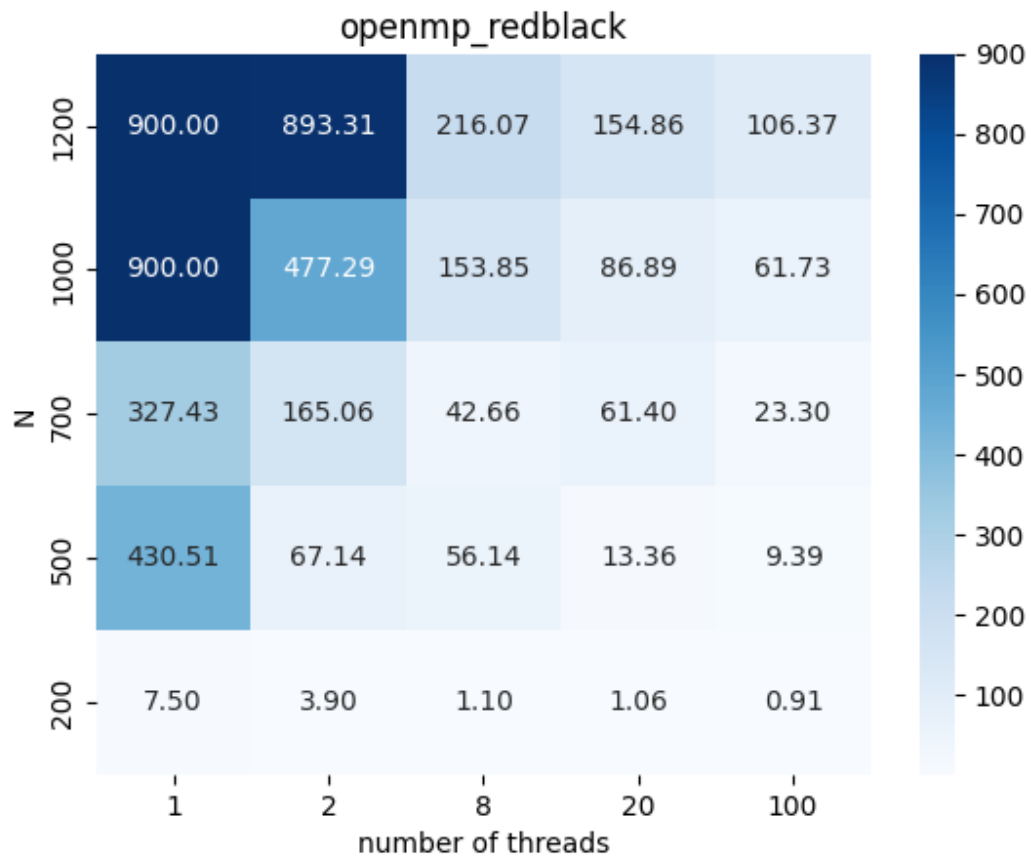


Рис. 1: Алгоритм RedBlack3D

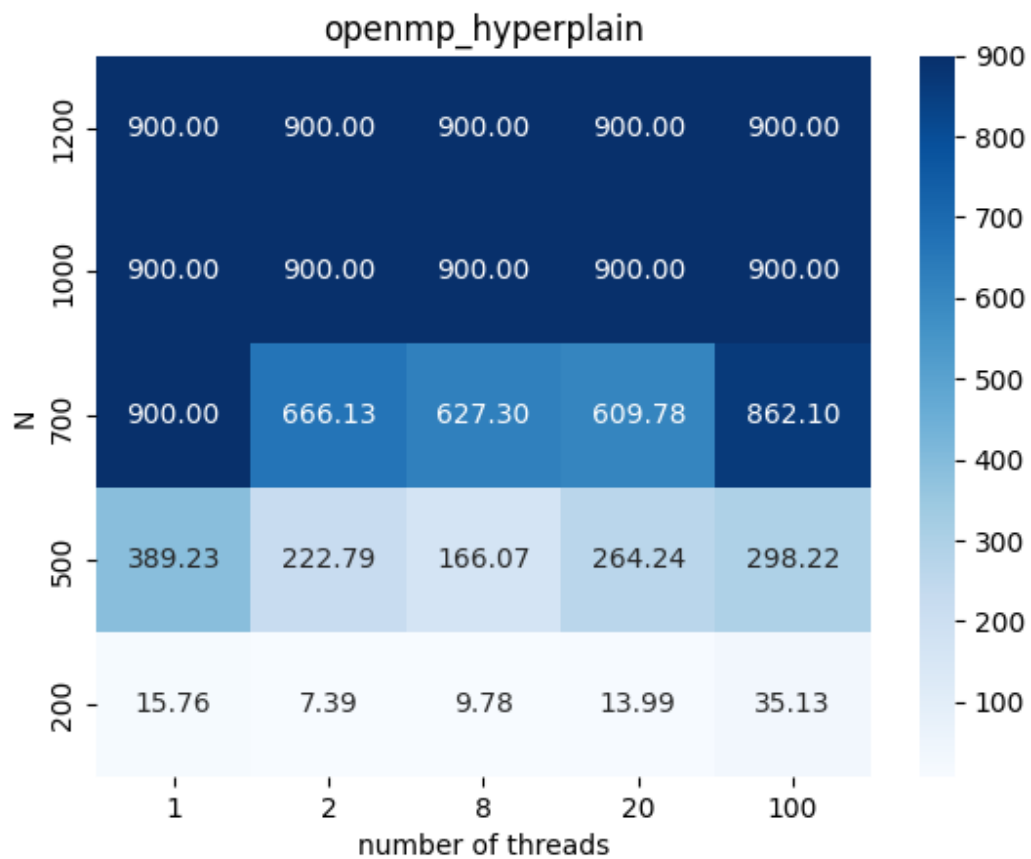


Рис. 2: Распараллеливание по гиперплоскостям куба

Для визуализации данных была использована библиотека **seaborn** для Python 3.

## 4 Выводы

Очевидно, что RedBlack3D алгоритм оказался гораздо быстрее, однако это компенсируется тем, что данный алгоритм в принципе не гарантирует какую-либо точность, а лишь позволяет обеспечить сходимость узкого класса методов.

Можно заметить, что с определенного порога при увеличении числа нитей алгоритм распараллеливания по гиперплоскостям дает худший результат. Скорее всего данный эффект обусловлен наличием в программе разделяемых ресурсов. Таким образом большое число нитей увеличивает расходы на передачу управления между ними, что замедляет работу программы.