МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра вычислительной техники



Курсовая работа

По дисциплине: «Параллельное программирование»

|  |  |
| --- | --- |
| Факультет: АВТ  Группа: АВТ-118  Студент: Исаков И. В. | Преподаватель: Зеленчук Н.А. |

Новосибирск

2023 г.

**Цель работы:**

Целью данной работы является изучение четырех технологий для параллельного вычисления: OpenMP и MPI. Стоит выделить постановки задач:

1. Создать последовательный алгоритм, который будет выполняться на основном потоке;

2. Создать параллельный алгоритм, который будет выполнять параллельное вычисление;

3. Получить один и тот же результат от двух типов вычислений, замеряя их время и сохраняя в отдельный файл (при огромном выводе).

**Описание алгоритмов:**

1. **Параллельное программирование для систем с общей памятью с использованием технологии OpenMP.**

Задание: Решение системы линейных алгебраических уравнений заданного порядка с заданной точностью.

Последовательный алгоритм:

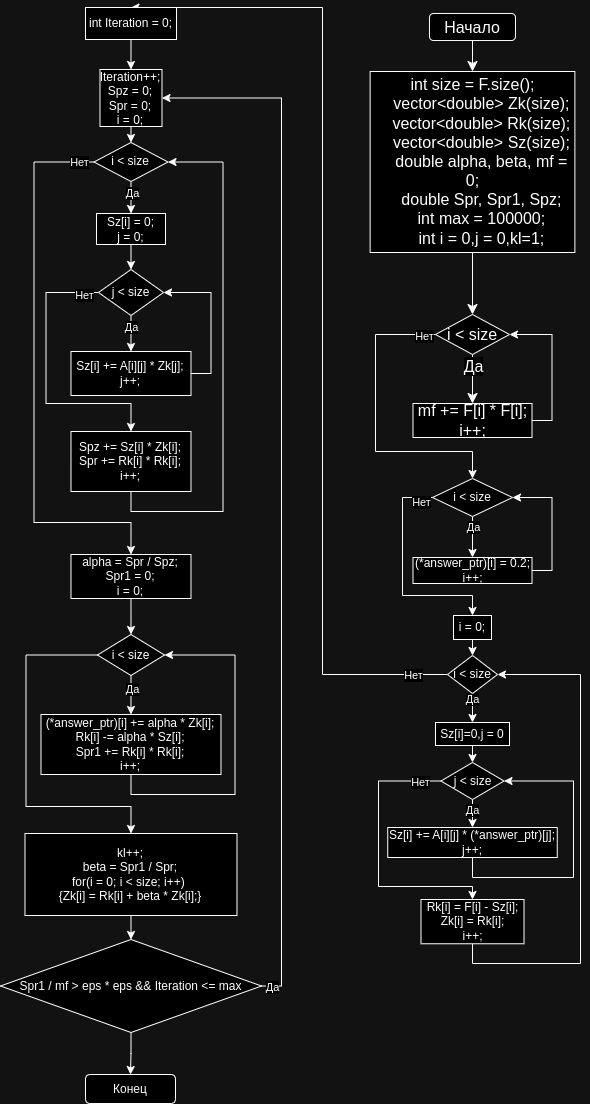


Рисунок 1. Блок-схема последовательного алгоритма.

Параллельный алгоритм:

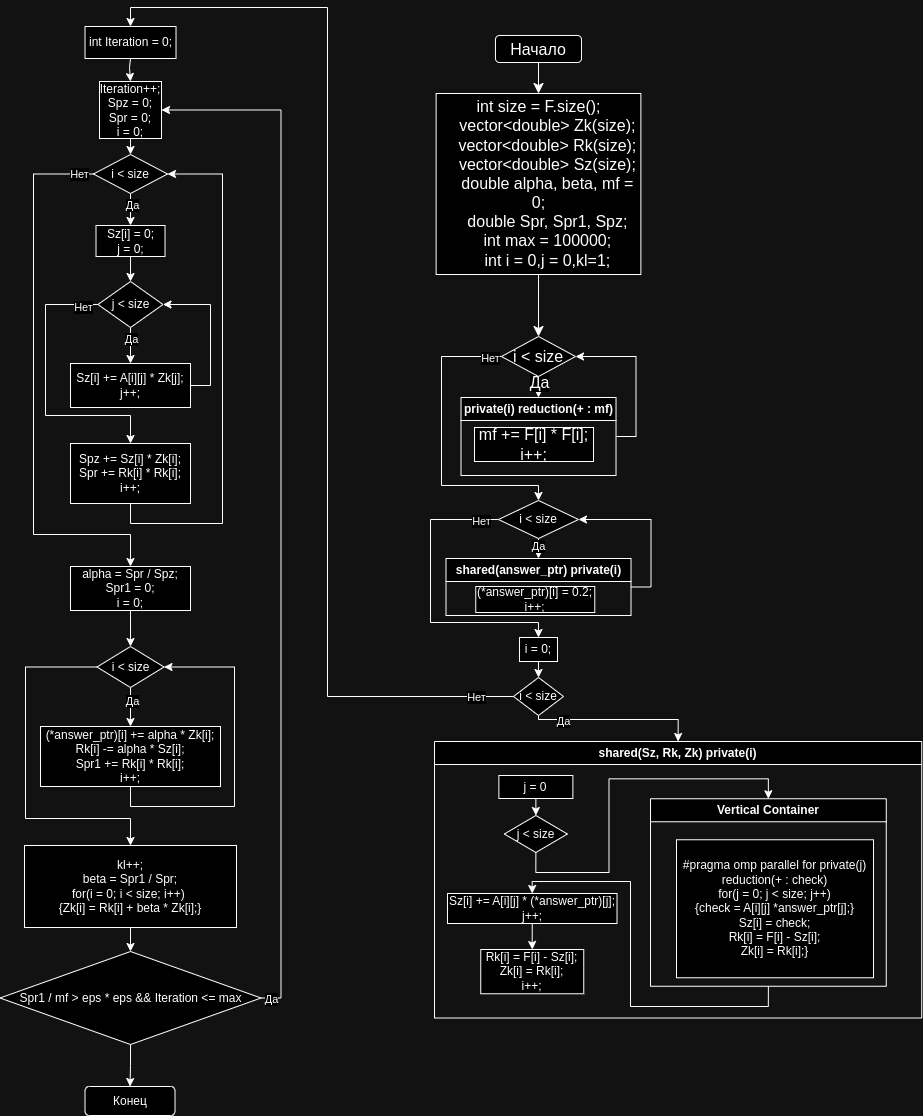


Рисунок 2. Блок-схема параллельного алгоритма.

1. **Параллельное программирование с использованием технологии MPI и библиотеки профилирования MPE.**

Задание: Найти максимальное число, меньшее заданного N, которое может быть представлено как сумма степеней 2, 3 и 4 простых чисел (минимальное такое число есть 28 = 22+23+24)

Последовательный алгоритм:

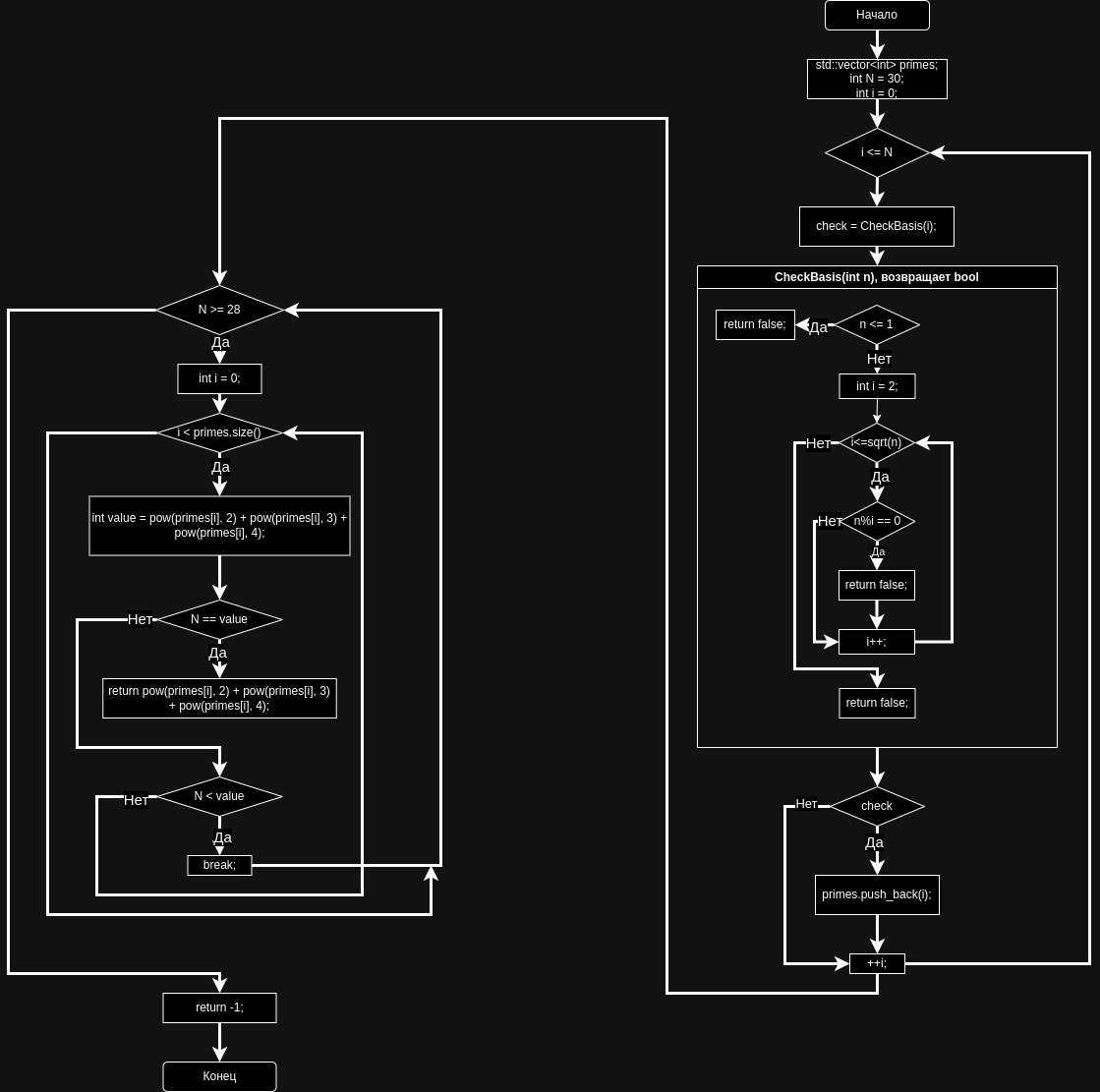


Рисунок 7. Блок-схема последовательного алгоритма.

Параллельный алгоритм:

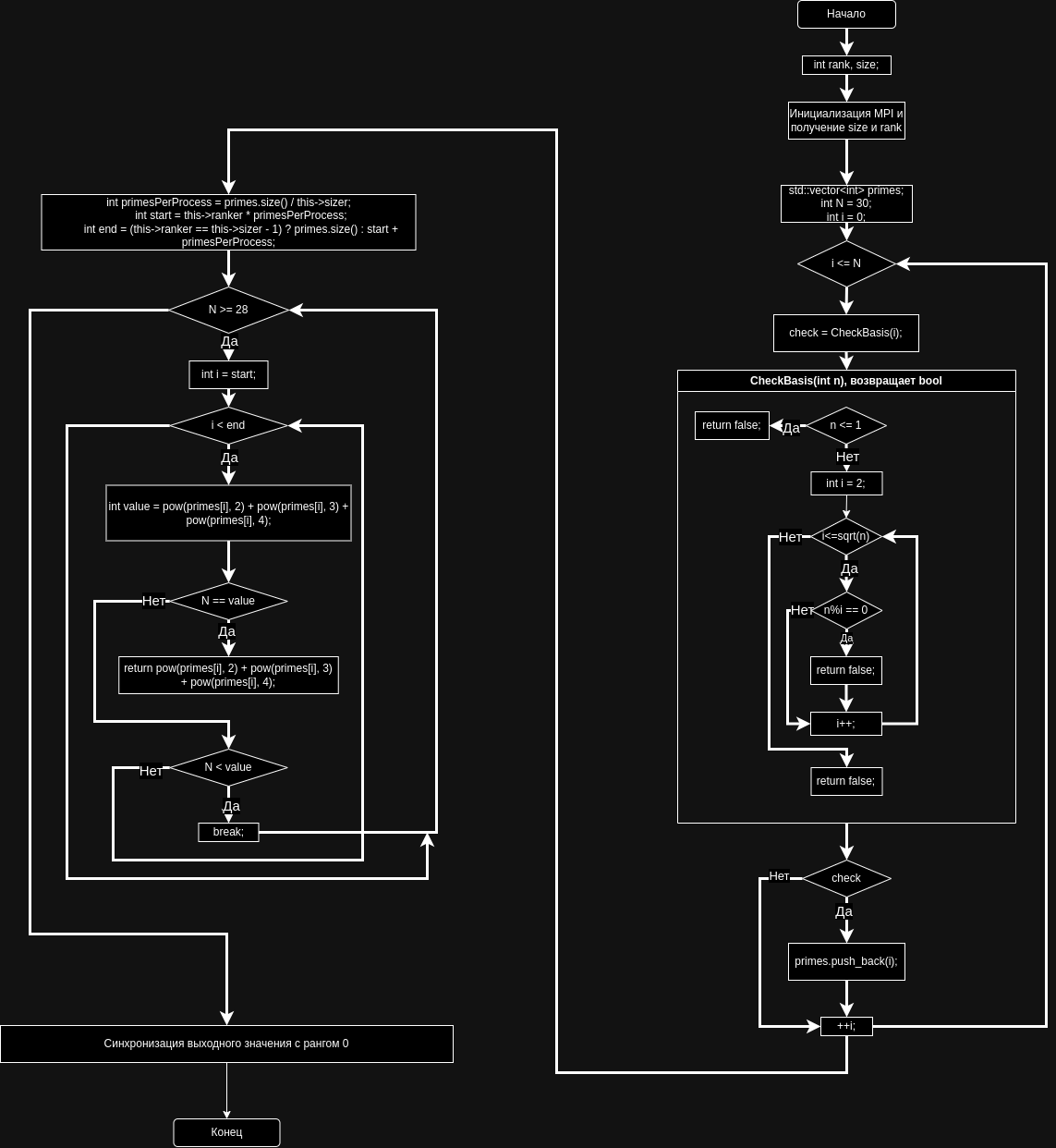


Рисунок 8. Блок-схема параллельного алгоритма.

**Результаты серий вычислений:**

1. Параллельное программирование для систем с общей памятью с использованием технологии OpenMP.

Для сравнения был подан первоначальный ввод данных:

Insert random matrix (y/n): y

Insert size of square matrix: 150

Insert max values: 20

Insert eps: 0.01

Для получения результатов введем разные размеры квадратной матрицы:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер матрицы | 10 | 50 | 100 | 1000 | 2000 | 5000 | 10000 | 20000 |
| Метод | Время в секундах | | | | | | | |
| Линейный | ~0 | ~0 | 0.002 | 0.009 | 0.033 | 0.22 | 0.75 | 4.55 |
| Параллельный | 0.006 | 0.03 | 0.001 | 0.003 | 0.005 | 0.062 | 0.11 | 0.46 |

Как видно, момент прироста ускорения вычисления на размере матрицы ~1000. Это может быть связано с тем, что процессору удобно вычислять линейно на очень маленьких размерах, ведь во время компиляции идет оптимизация процесса. Для параллельного вычисления требуется чуть больше времени на подготовку запуска, распараллеливания, ожидания завершения и вывода результата.

1. Параллельное программирование с использованием технологии MPI и библиотеки профилирования MPE.

Для получения результатов введем разные значения N:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10 | 1000 | 2000 | 3000 | 10000 | 20000 | 30000 |
| Метод | Время в секундах | | | | | | |
| Линейный | 0,43 | 0,65 | 0,27 | 0,15 | 0,19 | 0,27 | 0,58 |
| Параллельный | 0,60 | 0,12 | 0,15 | 0,20 | 0,11 | 0,20 | 0,53 |

В основе параллельного алгоритма лежит получение ранга и размера и выполнение вычисления так, чтобы его каждый ранг имели равное кол-во итераций на процесс.

**Выводы:**

Была произведена работа параллельного вычисления на четырех разных технологиях, где присутствовали вычисления как на GPU, так и на CPU. Можно сделать вывод, что самым легкодоступным был MPI, ведь технология позволяет с легкостью распараллелить алгоритм без особых знаний механизма работы технологии. Но более производительным показал себя OpenMP, алгоритм вычисления был хоть и не самым лучшим (как и само его использование), но он довольно удобен в инструментарии, в отличие от OpenCL, которому приходится собирать ядро для работы.