Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа №4
по дисциплине "Анализ Алгоритмов"

Тема Параллельное программирование

Студент Кишов Г. М.

Группа ИУ7-53Б

Преподаватель Волкова Л. Л.

Москва

2021 г.

СОДЕРЖАНИЕ

Введение

Многопоточность - способность центрального процессора (CPU) или одного ядра в многоядерном процессоре одновременно выполнять несколько процессов или потоков, соответствующим образом поддерживаемых операционной системой. Смысл многопоточности - квазимногозадачность на уровне одного исполняемого процесса. Данный подход не стоит путать с многопроцессорностью, так как многопоточность процессов и потоков совместно использует ресурсы одного или нескольких ядер: кэш-памяти CPU, вычислительных блоков.

В случаях, когда многопроцессорные системы включают в себя несколько полных блоков обработки, многопоточность направлена на максимизацию использования ресурсов одного ядра.

Так как эти два метода являются взаимодополняющими, то их иногда объединяют в системах с несколькими многопоточными ЦП и в ЦП с нескольким многопоточными ядрами.

Цель лабораторной работы: изучение параллельных вычислений на примере нахождения среднего геометрического строк матрицы.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- Исследовать основы параллельных вычислений;
- Привести схемы реализации последовательного и параллельного подсчета среднего геометрического строк матрицы;
- Определить средства программной реализации;
- Протестировать разработанное программное обеспечение;
- Провести сравнительный анализ временных характеристик алгоритмов;
- На основании проделанной работы сделать выводы.

1 Аналитический раздел

В данном разделе будет рассмотрен алгоритм нахождения среднего геометрического строк матрицы и его параллельная реализация.

1.1 Некоторые теоретические сведения

Средним геометрическим [1] ряда положительных чисел называется такое число, которым можно заменить каждое из данных чисел так, чтобы их произведение не изменилось. Другими словами, среднее геометрическое *n* чисел равно корню *n-ой* степени из их произведения.

Формула вычисления среднего геометрического *n* чисел:

$$G(x_1, x_2 \cdots x_n) = \sqrt[1/n]{\prod_{i=1}^{n} x_i} \tag{1}$$

1.2 Алгоритм нахождения среднего геометрического строк матрицы

Пусть дана прямоугольная матрица

$$A_{nm} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}, \tag{2}$$

Тогда средним геометрическим строк матрицы *A* будет называться массив *B*

$$Bn = \begin{pmatrix} b_1 & b_2 & \cdots & b_n \end{pmatrix}, \tag{3}$$

где

$$B[i] = \sqrt[1/m]{\prod_{j=1}^{m} a_{ij}}, \quad (i = \overline{1, n}) \tag{4}$$

Среднее геометрическое *i-ой* строки матрицы *A*.

1.3 Параллельный алгоритм нахождения среднего геометрического строк матрицы

Поскольку каждый элемент массива B вычисляется независимо от других и матрица A не изменяется в процессе, то для распараллеливания алгоритма нахождения среднего геометрического строк матрицы достаточно равным образом распределить строки матрицы A между потоками.

1.4 Вывод

В данном разделе были рассмотрены последовательный и параллельный алгоритмы нахождения среднего геометрического строк матрицы.

В качестве входных данных в программу будут подаваться целочисленная матрица, ее размер (количество строк и столбцов матрицы). На выходе программа будет выдавать массив, который будет содержать средние геометрические каждой строки исходной матрицы. Ограничением для работы программного продукта будет являться то, что размеры матриц должны быть целыми положительными числами, а сами матрицы состоять только из положительных целочисленных значений.

Реализуемое программное обеспечение будет работать в пользовательском и экспериментальном режимах. В пользовательском режиме можно будет ввести матрицу и программа выведет среднее геометрическое строк данной матрицы, в экспериментальном режиме будет проводиться сравнение временных характеристик последовательного и параллельного алгоритма нахождения среднего геометрического строк матрицы при разном количестве потоков.

2 Конструкторский раздел

В данном разделе будут представлены схемы алгоритмов, описаны используемые типы данных, структура программного обеспечения (далее - ПО) и классы эквивалентности.

2.1 Схемы алгоритмов

На рисунках 1-3 приведены схемы реализации алгоритмов последовательного и параллельного нахождения среднего геометрического строк матрицы.

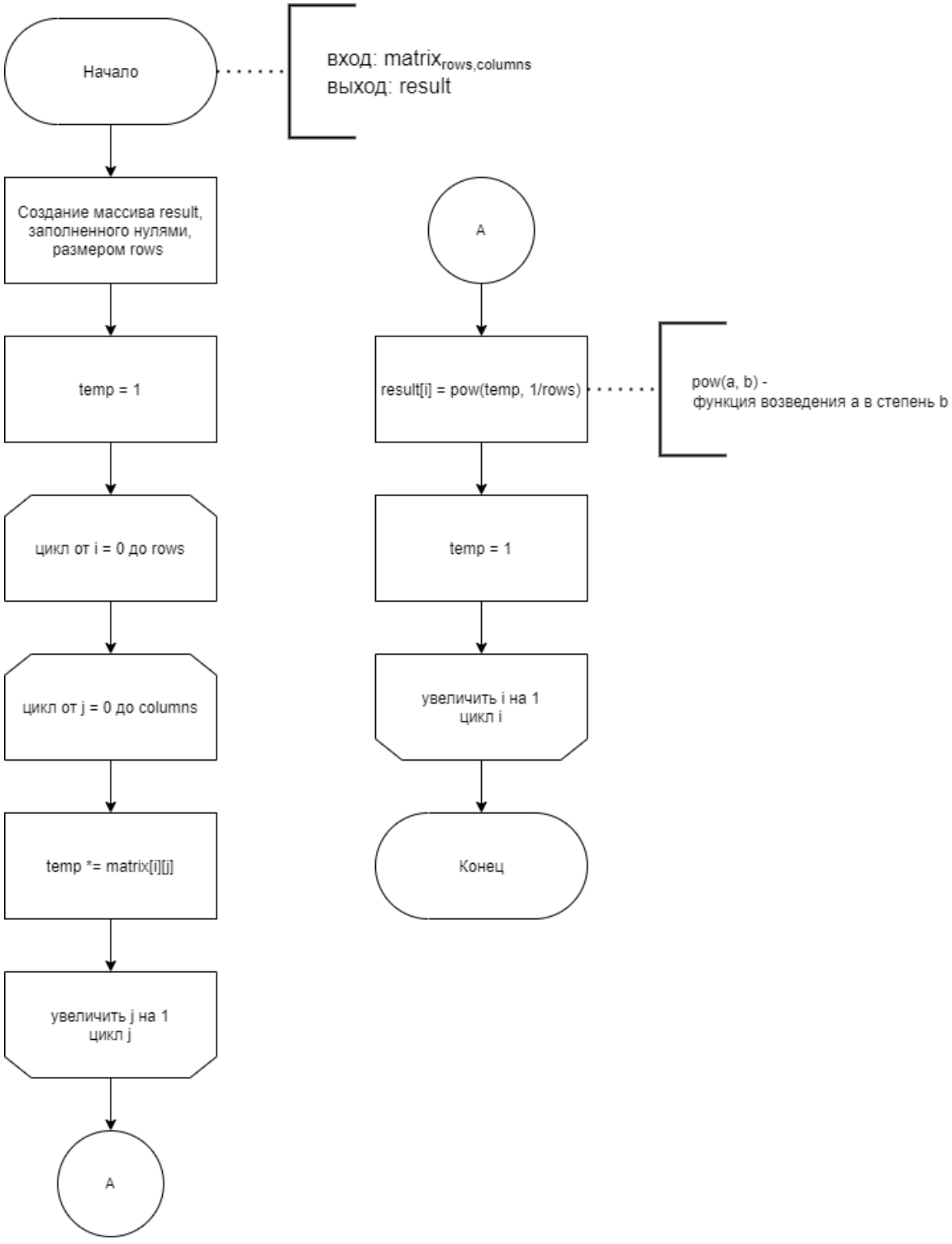

Рис. 1 — Схема последовательного алгоритма нахождения среднего геометрического строк матрицы

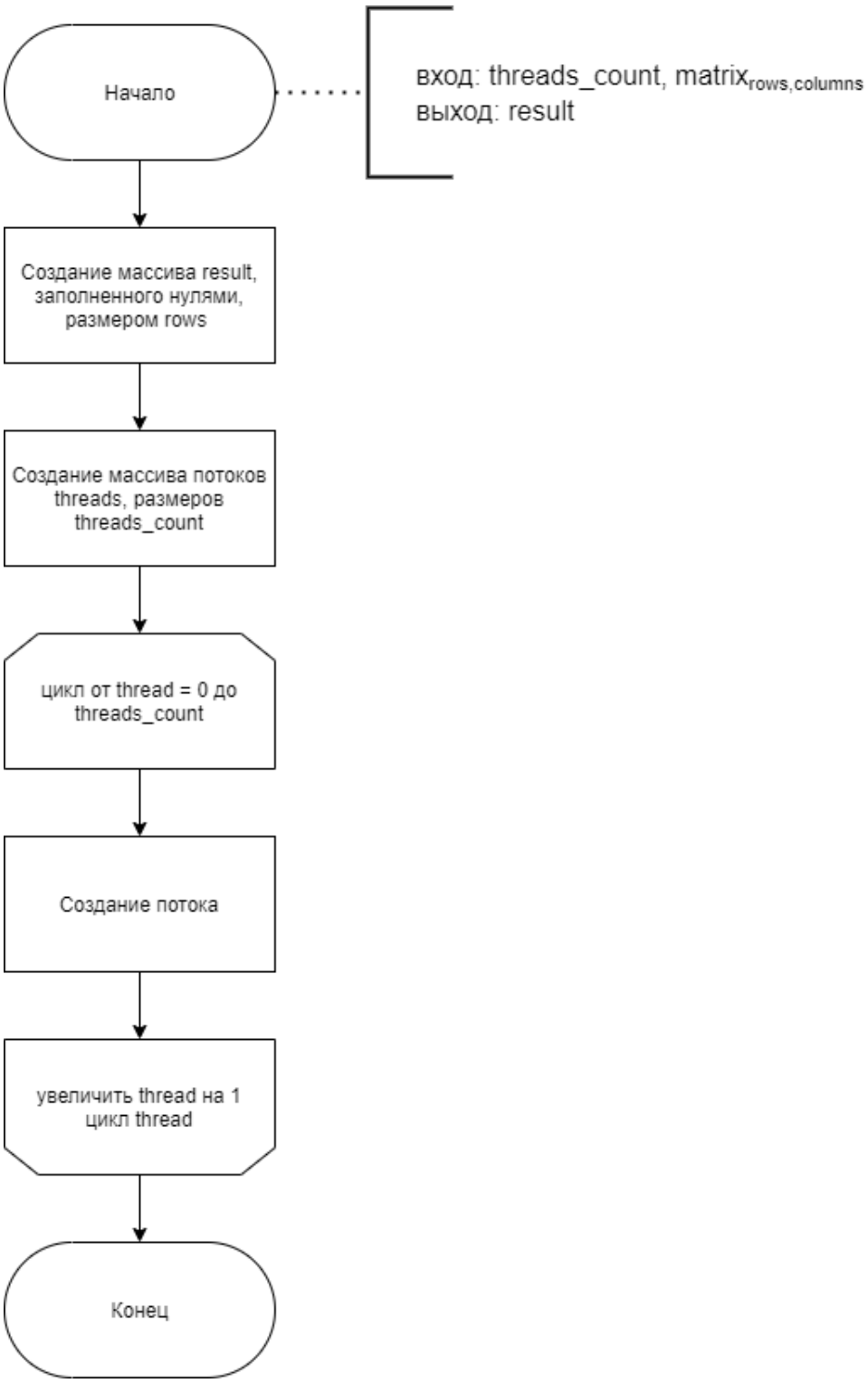

Рис. 2 — Схема параллельного алгоритма нахождения среднего геометрического строк матрицы

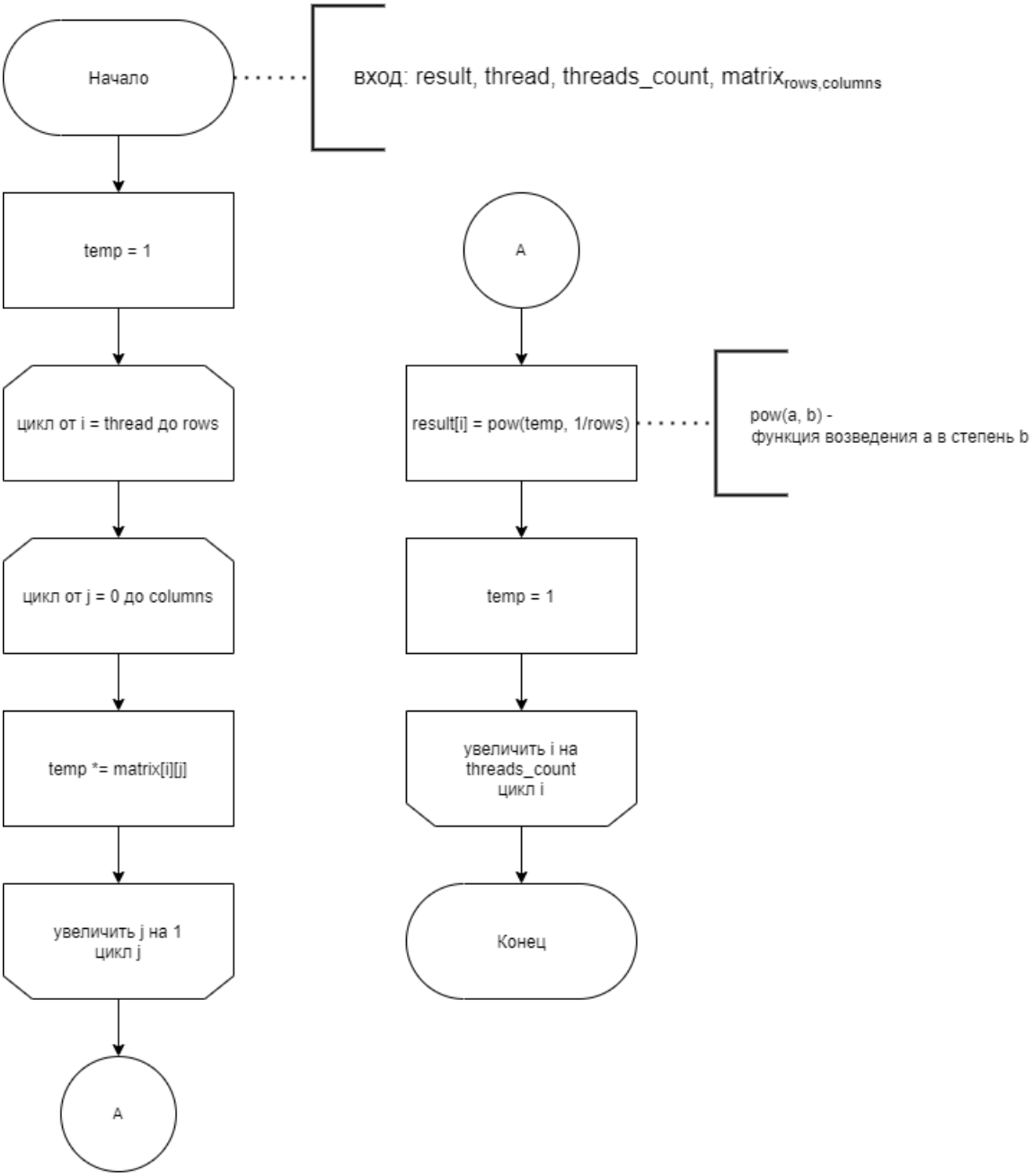

Рис. 3 — Схема создания одного потока

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие структуры данных:

1. количество строк матрицы - целое число типа int;
2. количество столбцов матрицы - целок число типа int;
3. матрица - двумерный массив типа int;
4. результирующий массив - одномерный массив типа int;
5. массив потоков - одномерны массив типа thread.

2.3 Структура ПО

ПО будет состоять из двух модулей:

1. *main.cpp* - модуль, содержащий весь служебный код;
2. *algos.cpp* - модуль, содержащий код алгоритмов нахождения среднего геометрического строк матрицы.

2.4 Способы тестирования и классы эквивалентности

При тестировании алгоритмов была выбрана методика тестирования черным ящиком. Были выделены следующие классы эквивалентности:

1. Матрица заполненная случайными положительными числами;
2. Матрица, заполненная построчно равными элементами;
3. Матрица, содержащая один элемент;
4. Пустая матрица.

2.5 Вывод

В данном разделе были построены схемы алгоритмов нахождения среднего геометрического строк матрицы, описаны используемые типы данных, структура ПО, способы тестирования и классы эквивалентности.

3 Технологический раздел

В данном разделе приведены средства реализации и листинги реализаций алгоритмов и тестирование.

3.1 Средства реализации

Для реализации программ был выбран язык программирования C++ [2]. Данный язык был выбран поскольку я ознакомился с ним из курса Объектно-ориентированного программирования, а также он предоставляет инструменты для замера процессорного времени и удобной работы с потоками.

Для работы с потоками использовалась библиотека thread [3].

3.2 Листинги

Листинг 1: Функция последовательного нахождения среднего геометрического строк матрицы

```
1  double *find_geometric_mean(matrix_s matrix)
2  {
3      double *result = (double*) calloc(matrix.rows, sizeof(
       double));
4
5      long long int temp = 1;
6
7      for (int i = 0; i < matrix.rows; i++)
8      {
9          for (int j = 0; j < matrix.columns; j++)
10         {
11             temp *= matrix.matrix[i][j];
12         }
13
14         result[i] = pow(temp, 1.0/matrix.rows);
15         temp = 1;
16     }
17
18     return result;
```

```
19 }
```

Листинг 2: Функция создания потоков

```
1  double* parallel_geometric_mean(matrix_s matrix, int
       threads_count)
2  {
3      double *result = (double*) calloc(matrix.rows, sizeof(
       double));
4
5      vector<thread> threads(threads_count);
6
7      for (int thread = 0; thread < threads_count; thread++)
8      {
9          threads[thread] = std::thread(parallel_additional,
       ref(result), matrix, thread, threads_count);
10     }
11
12     for (int i = 0; i < threads_count; i++)
13     {
14         threads[i].join();
15     }
16
17     return result;
18 }
```

Листинг 3: Функция параллельного нахождения среднего геометрического строк матрицы

```
1  void parallel_additional(double *result, matrix_s matrix, int
       thread, int threads_count)
2  {
3      long long int temp = 1;
4
5      for (int i = thread; i < matrix.rows; i += threads_count)
6      {
7          for (int j = 0; j < matrix.columns; j++)
8          {
9              temp *= matrix.matrix[i][j];
10         }
```

```
11
12              result[i] = pow(temp, 1.0/matrix.rows);
13              temp = 1;
14          }
15  }
```

3.3 Тестирование функций

В таблице 1 приведены модульные тесты для функций умножения матриц выше перечисленными методами. Все тесты были пройдены успешно.

Таблица 1 — Тестирование функций нахождения среднего геометрического строк матрицы

Матрица	Ожидаемый результат
$\begin{pmatrix} 2 & 8 & 5 \\ 1 & 10 & 5 \\ 9 & 9 & 3 \end{pmatrix}$	$\begin{pmatrix} 4.30887 & 3.68403 & 6.24025 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 \\ 4 & 4 \end{pmatrix}$	$\begin{pmatrix} 1 & 4 \end{pmatrix}$
$\begin{pmatrix} 8 \end{pmatrix}$	$\begin{pmatrix} 8 \end{pmatrix}$
$\begin{pmatrix} \end{pmatrix}$	Пусто

3.4 Вывод

Были разработаны и протестированы последовательная и параллельная реализации алгоритма нахождения среднего геометрического строк матрицы. Также было проведено тестирование и описаные средства реализации.

4 Исследовательский раздел

В данном разделе будут представлены технические характеристики машины, замер времени выполнения алгоритмов и пример работы программы.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование представлены далее.

— Операционная система: Windows 10;

— Память: 16 GB 3133 MHz DDR4;

— Процессор: Intel® Core™ i7-10700K CPU @ 3.80GHz [4];

— 8 физических и 16 логических ядер.

Тестирование проводилось на компьютере, включенном в сеть электропитания. Во время Тестирования компьютер был нагружен только встроенными приложениями окружения рабочего стола.

4.2 Пример работы программы

Демонстрация работы программы приведена на рисунке 6.

```
Run time analysis? (0 - no, 1 - yes):1

Input matrix rows:1000

Input matrix columns:1000

Time sequential on random array: 0.002619 seconds

Time parallel 2 threads on random array: 0.001660 seconds

Time parallel 4 threads on random array: 0.000910 seconds

Time parallel 8 threads on random array: 0.000697 seconds

Time parallel 16 threads on random array: 0.000697 seconds

Time parallel 32 threads on random array: 0.000995 seconds

Time parallel 64 threads on random array: 0.001905 seconds

Input matrix rows:0

Input matrix columns:0

Input matrix:
Matrix:

Geometric mean of each row:

Process finished with exit code 0
```

Рис. 4 — Пример работы программы.

4.3 Время выполнения алгоритмов

Был проведен замер времени работы каждого из алгоритмов с помощью библиотеки chrono [5]. Эта библиотека замеряет процессорное время выполнения функции. При выполнении алгоритма проводилось 20 замеров и полученное время усреднялось. В таблице 2 содержаться временные характеристики параллельной реализации алгоритма при размере матрицы 10000x10000 и при разном количестве потоков. В таблице 3 приводится сравнение времени работы последовательной и параллельной реализации алгоритма при 16 потоках на разных размерах матриц.

На рисунке 5 демонстрируется сравнение временных характеристик параллельной реализации алгоритма при размере матрицы 10000x10000 и при разном количестве потоков. На рисунке 6 демонстрируется сравнение времени работы последовательной и параллельной реализации алгоритма при 16 потоках на разных размерах матриц.

Таблица 2 — Время выполнения параллельной реализации алгоритма (в секундах) при размере матрицы 10000x10000

Количество потоков	Параллельная реализация алгоритма
1	0.265104
2	0.133861
4	0.069908
8	0.053910
16	0.046719
32	0.047453
64	0.048590

Таблица 3 — Время выполнения реализаций алгоритмов (в секундах) при 16 потоках

Размер матрицы	Последовательный алгоритм	Параллельный алгоритм
1000x1000	0.002669	0.000737
5000x5000	0.066974	0.009691
10000x10000	0.264854	0.046719
15000x15000	0.616687	0.076846
20000x20000	1.075421	0.137248

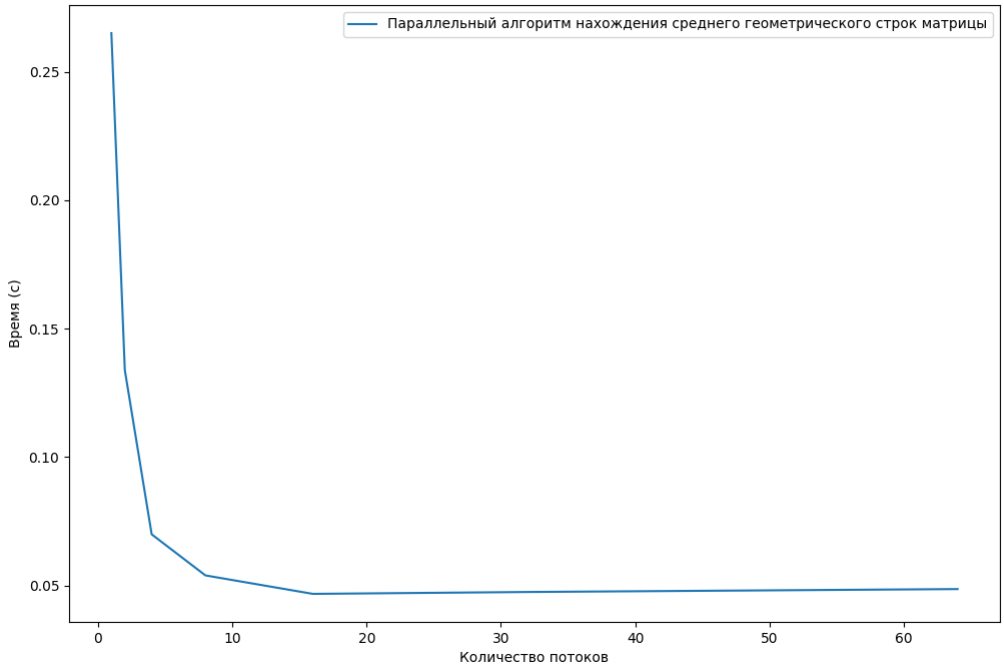

Рис. 5 — Зависимость времени выполнения параллельной реализации алгоритма от количества потоков при размере матрицы 10000x10000

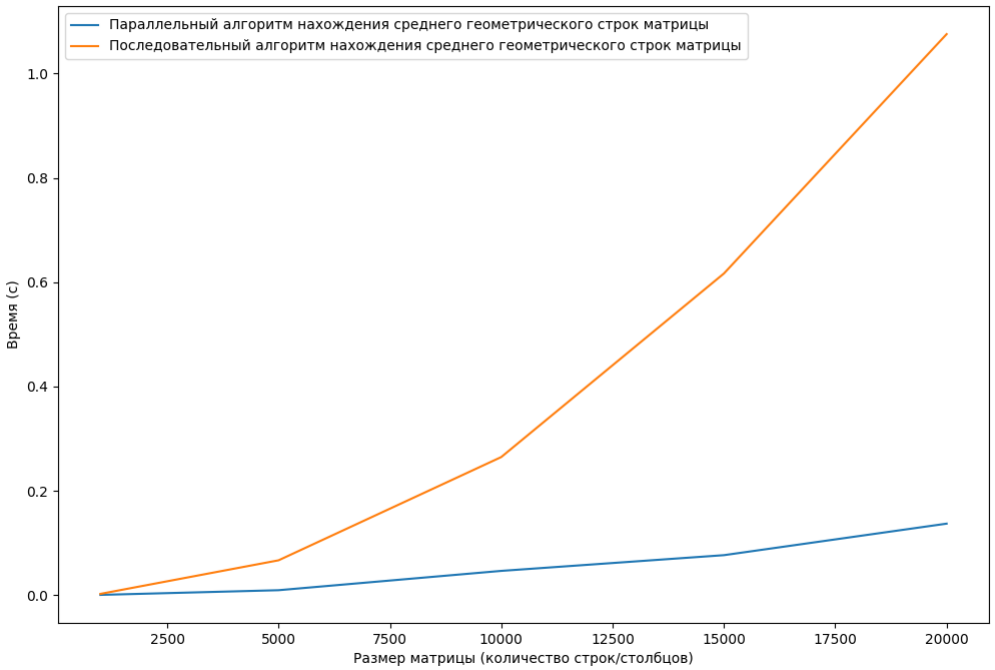

Рис. 6 — Зависимость времени выполнения реализаций алгоритмов от размера матрицы при 16 потоках

4.4 Вывод

В данном разделе были описаны технические характеристики машины, приведен пример работы программы и проанализированы временные характеристики алгоритмов нахождения среднего геометрического строк матрицы.

По результатам экспериментов, можно заметить, что лучшее время наблюдается при 16 потоках (быстрее последовательной реализации в 5.7 раз), поскольку мой процессор обладает 16-ю логичскими ядрами, а следовательно максимально одновременно может быть запущено 16 потоков. При количестве потоков больше 16 замечается остановка прогрессии по уменьшению времени работы алгоритма, так как происходит создание очереди самих потоков (если задать 32 потока, то процессор сможет одновременно выполнять только 16 их них, остальные будут ждать своей очереди). Также стоит заметить, что даже при двух потоках время работы алгоритма уменьшилось примерно в 2 раза, а при четырех потоках, примерно в 4 раза.

Заключение

В результате выполнения лабораторной работы было экспериментально подтверждено различие временных характеристик последовательной и параллельной реализации алгоритма нахождения среднего геометрического строк матрицы.

В результате исследования временных характеристик можно сделать вывод, что пиковый прирост по времени приходится на 16 потоков (быстрее последовательной реализации в 5.7 раз), посколько процессор, на которым выполнялись замеры имеет 16 логических ядер.

В ходе выполнения работы была достигнута цель и выполнены все поставленные задачи:

- Были исследованы основы параллельных вычислений;
- Были приведены схемы требуемых алгоритмов;
- Были определены средства программной реализации;
- Было протестированно разработанное ПО;
- Был проведен сравнительный анализ временных характеристик алгоритмов;
- На основанни проделанной работы были сделаны соответствующие выводы.

Список литературы

1. Среднее геометрическое ряда чисел [Электронный ресурс] Режим доступа: https://umath.ru/calc/srednee-arifmeticheskoe-i-srednee-geometricheskoe-chisel-onlajn/ (дата обращения: 06.11.2021).
2. Бьёрн Страуструп. Язык программирования C++. Специальное издание = The C++ programming language. Special edition. — Бином-Пресс, 2007. — 1104 с. — ISBN 5-7989-0223-4.
3. Библиотека для работы с потоками thread [Электронный ресурс] Режис доступа: https://en.cppreference.com/w/cpp/thread (дата обращения: 06.11.2021).
4. Процессор Intel® Core™ i7-10700K [Электронный ресурс] Режим доступа: https://www.intel.ru/content/www/ru/ru/products/sku/199335/intel-core-i710700k-processor-16m-cache-up-to-5-10-ghz/specifications.html (дата обращения: 06.11.2021).
5. Библиотека Chrono [Электронный ресурс] Режим доступа: https://en.cppreference.com/w/cpp/chrono (дата обращения: 06.11.2021).