



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»

Отчёт по лабораторной работе №5 по дисциплине "Анализ алгоритмов"

Тема Конвейерные вычисления

Студент Варламова Е. А.

Группа ИУ7-51Б

Преподаватели Волкова Л.Л.

Москва — 2021 г.

Оглавление

Введение	3
1 Аналитическая часть	4
1.1 Конвейерная обработка данных	4
1.2 Описание задачи	4
2 Конструкторская часть	5
2.1 Разработка алгоритмов	5
2.2 Используемые структуры данных	7
3 Технологическая часть	8
3.1 Средства реализации	8
3.2 Реализация алгоритмов	8
4 Исследовательская часть	12
4.1 Технические характеристики ЭВМ	12
4.2 Время выполнения алгоритмов	12
Заключение	15
Литература	16

Введение

При обработке данных могут возникать ситуации, когда один набор данных необходимо обработать последовательно несколькими алгоритмами. В таком случае удобно использовать идею конвейерной обработки. Конвейер - механизм организации труда, при котором производство изделия разбивается на стадии, и конкретные работники закрепляются за своими стадиями. Таким образом, обработка делится на линии конвейера. При этом каждая следующая линия обрабатывает данные только тогда, когда предыдущая завершила свою работу над теми же данными.

Здесь также можно заметить, что обработка данных на разных линиях конвейера может быть выполнена параллельно, что значительно улучшит производительность конвейера за счёт уменьшения времени простоя линий.

Целью данной работы является изучение и реализация параллельной и линейной реализаций конвейерной обработки данных. Для достижения поставленной цели необходимо решить следующие задачи:

- изучить конвейерную обработку данных;
- разработать метод конвейерной обработки данных на примере приготовления печенья;
- реализовать систему конвейерных вычислений;
- сравнить параллельную и последовательную реализации конвейерных вычислений на основе собранной статистики.

1 | Аналитическая часть

В данном разделе представлены теоретические сведения о рассматриваемых алгоритмах.

1.1 Конвейерная обработка данных

Конвейер - это механизм, который позволяет обрабатывать однотипные заявки поэтапно. Поэтому конвейер представляет собой некоторое количество лент, на каждой из которых выполняется одинаковая работа над поступающими заявками. В терминах программирования ленты конвейера представлены функциями-обработчиками, выполняющими над неким набором данных операции и передающими их на следующую ленту конвейера. Очевидно, что моделирование конвейерной обработки хорошо сочетается с технологией многопоточного программирования, так как под каждую ленту конвейера может быть выделен отдельный поток.

1.2 Описание задачи

Для моделирования системы конвейерной обработки данных был выбран процесс приготовления печенья в производственных масштабах, состоящий из трех этапов:

- подсчёт массы куриных яиц в мг (вычисление n -ого числа Фибоначчи);
- подсчёт массы масла в мг (возведение в степень)
- подсчёт массы муки в мг (факториал числа).

Вывод

В данном разделе были рассмотрены особенности построения конвейерных вычислений.

2 | Конструкторская часть

В данном разделе представлены схемы рассматриваемых алгоритмов.

2.1 Разработка алгоритмов

На рисунке 2.1 приведена общая схема организации конвейерных вычислений для каждой заявки.

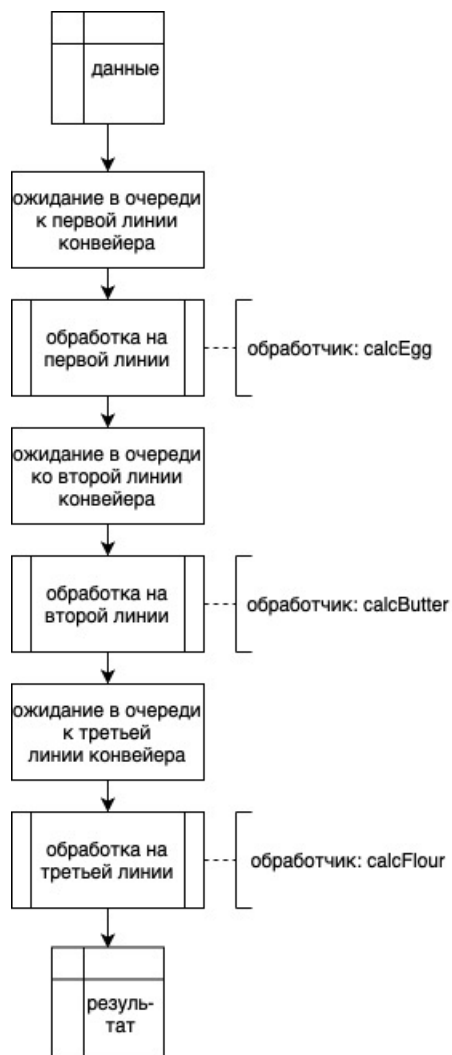


Рис. 2.1: Общая схема организации конвейерных вычислений.

На рисунке 2.2 приведена схема главного потока для параллельной обработки заявок.

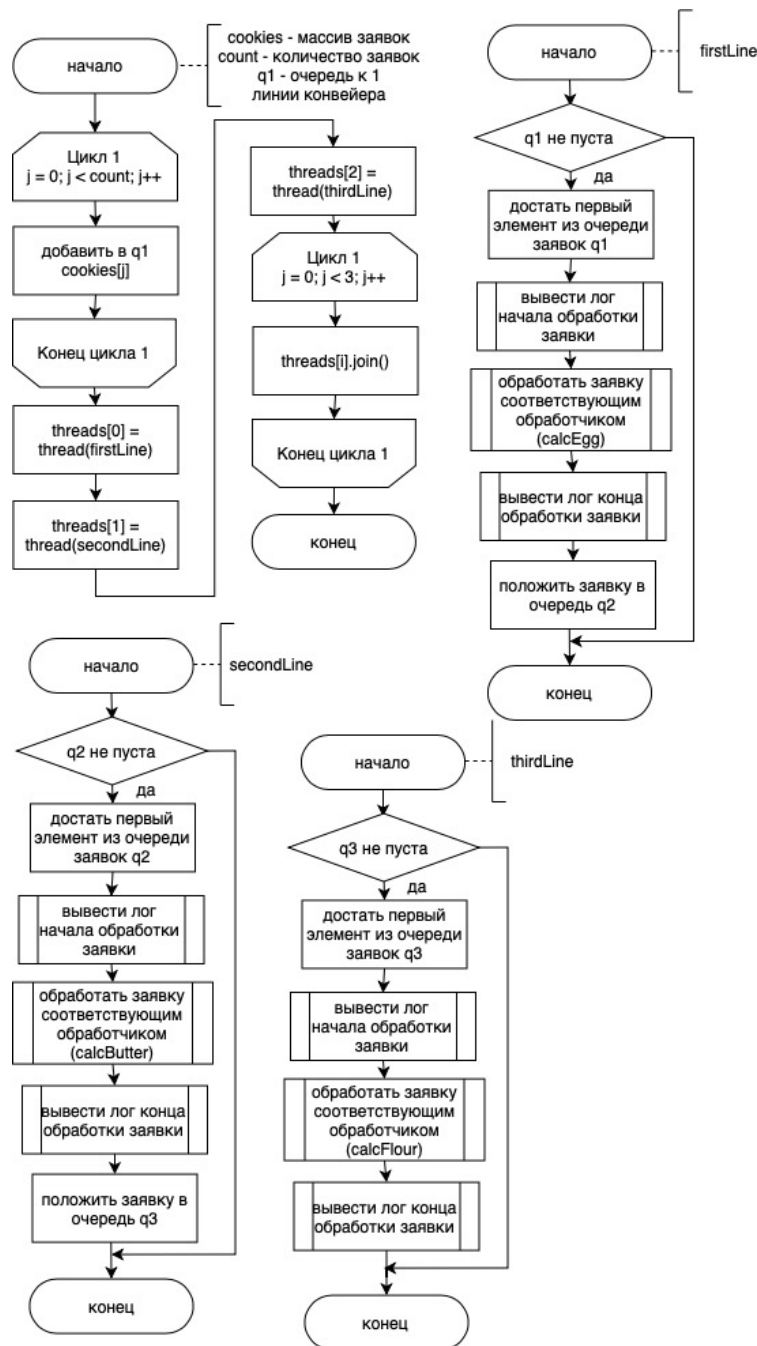


Рис. 2.2: Схема главного потока для параллельной обработки заявок

На рисунке 2.3 приведены схемы каждого типа вычислений конвейера.

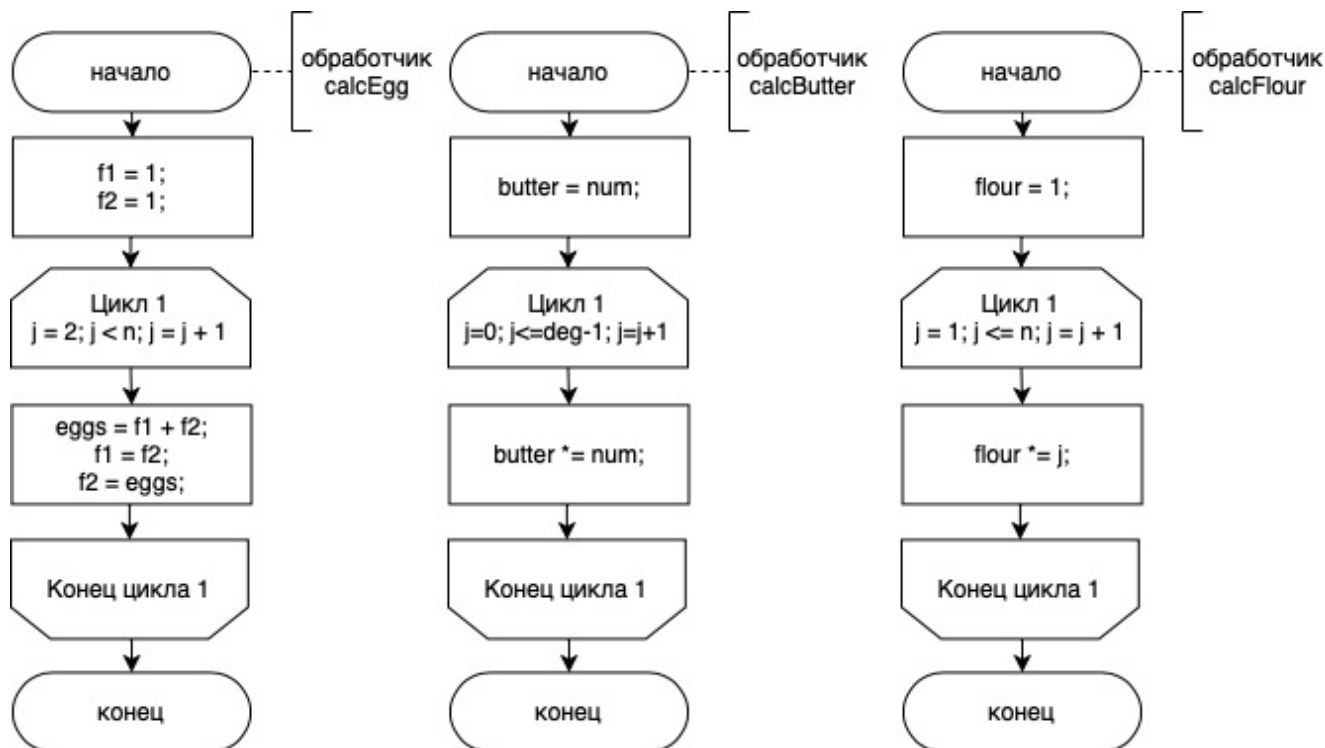


Рис. 2.3: Схемы трёх типов вычислений конвейера

2.2 Используемые структуры данных

Для организации потокобезопасных очередей было решено использовать обёртку над стандартным типом данных очередь. Такая очередь обеспечивает монопольное владение ресурсом каждым потоком, что не позволяет привести к ситуации гонки. Так, каждый вызов методов очереди блокируется мьютексом.

Вывод

На основе анализа принципа работы конвейера была разработана общая схема организации конвейерных вычислений и схемы вычислений на каждой ленте конвейера в соответствии с выбранным для моделирования процессом. Кроме того, были выбраны используемые структуры данных.

3 | Технологическая часть

В данном разделе приведены средства реализации и листинги кода.

3.1 Средства реализации

Для реализации был выбран язык программирования C++. Данный выбор обусловлен наличием инструментов для создания и эффективной работы с потоками. В качестве среды разработки была выбрана среда CLion.

3.2 Реализация алгоритмов

В листингах 3.1 и 3.2 приведены однопоточная и многопоточная реализации конвейерных вычислений, в листинге 3.3 - реализация потокобезопасной очереди, а в листинге 3.4 - реализация обработчиков, представляющих этапы конвейера.

Листинг 3.1: Однопоточная реализация конвейера

```
1 void Conveyor::run_seq(size_t count) {  
2     for (size_t i = 0; i < count; i++) {  
3         std::shared_ptr< cookies > c(new Cookies);  
4         c->calcEgg(1000000);  
5         c->calcButter(2, 1000000);  
6         c->calcFlour(1000000);  
7     }  
8 }
```

Листинг 3.2: Многопоточная реализация конвейера

```
1 void Conveyor::run_par(size_t count) {  
2     for (size_t i = 0; i < count; i++) {  
3         std::shared_ptr< cookies > p(new Cookies);  
4         cookies.push_back(p);  
5         q1.push(p);  
6     }  
7     this->threads[0] = std::thread(&Conveyor::calcEgg, this);  
8     this->threads[1] = std::thread(&Conveyor::calcButter, this);  
9     this->threads[2] = std::thread(&Conveyor::calcFlour, this);  
10    for (int i = 0; i < 3; i++) {  
11        this->threads[i].join();  
12    }
```



```

13 }
14 void Conveyor::calcEgg()
15 {
16     size_t task_num = 1;
17     while (!this->q1.empty()) {
18         std::shared_ptr<Cookies> c = q1.front();
19         log_current_event(task_num, "Part 1 | Start");
20         c->calcEgg(1000000);
21         log_current_event(task_num, "Part 1 | End");
22         q2.push(c);
23         q1.pop();
24         task_num++;
25     }
26 }
27 void Conveyor::calcButter()
28 {
29     size_t task_num = 1;
30     do {
31         if (!this->q2.empty()) {
32             std::shared_ptr<Cookies> c = q2.front();
33             log_current_event(task_num, "Part 2 | Start");
34             c->calcButter(2, 1000000);
35             log_current_event(task_num, "Part 2 | End");
36             q3.push(c);
37             q2.pop();
38             task_num++;
39         }
40     } while (!q1.empty() || !q2.empty());
41 }
42 void Conveyor::calcFlour()
43 {
44     size_t task_num = 1;
45     do {
46         if (!this->q3.empty()) {
47             std::shared_ptr<Cookies> c = q3.front();
48             log_current_event(task_num, "Part 3 | Start");
49             c->calcFlour(1000000);
50             log_current_event(task_num, "Part 3 | End");
51             q3.pop();
52             task_num++;
53         }
54     } while (!q1.empty() || !q2.empty() || !q3.empty());
55 }
56 }

```

Листинг 3.3: Реализация потокобезопасной очереди

```

1 std::mutex mtx;
2 template <typename type>
3 void SafeQueue<type>::push(type val)
4 {
5     mtx.lock();
6     q.push(val);
7     mtx.unlock();
8 }
9 template <typename type>
10 void SafeQueue<type>::pop()
11 {
12     mtx.lock();
13     q.pop();
14     mtx.unlock();
15 }
16 template <typename type>
17 bool SafeQueue<type>::empty()
18 {
19     bool r;
20     mtx.lock();
21     r = q.empty();
22     mtx.unlock();
23     return r;
24 }
25 template <typename type>
26 type SafeQueue<type>::front()
27 {
28     type val;
29     mtx.lock();
30     val = q.front();
31     mtx.unlock();
32     return val;
33 }

```

Листинг 3.4: Обработчики линий конвейера

```
1 void Cookies::calcEgg(int n) // fib
2 {
3     int f1 = 1, f2 = 1;
4
5     for (int i = 2; i < n; i++)
6     {
7         this->eggs = f1 + f2;
8         f1 = f2;
9         f2 = this->eggs;
10    }
11 }
12 void Cookies::calcButter(int num, int deg) // num ^ degree
13 {
14     this->butter = num;
15     for (int i = 0; i < deg - 1; i++)
16         this->butter *= num;
17 }
18 void Cookies::calcFlour(long n) // factorial
19 {
20     this->flour = 1;
21     for (int i = 1; i <= n; i++)
22         this->flour *= i;
23 }
```

Вывод

В данном разделе была реализована конвейерная обработка данных на примере моделировании процесса приготовления печенья, кроме того, были выбраны средства разработки.

4 | Исследовательская часть

В данном разделе приведен анализ характеристик разработанного ПО и примеры работы ПО.

4.1 Технические характеристики ЭВМ

Все нижеприведенные замеры времени проведены на процессоре: Intel Core i5, 1,4 GHz, четырёхъядерный, количество логических ядер - 8. Время работы алгоритмов было замерено с помощью chrono [1].

4.2 Время выполнения алгоритмов

На рисунке 4.1 приведено сравнение времени выполнения параллельной и последовательной обработки данных в зависимости от количества входных задач. При этом предполагается, что на лентах конвейера заявки обрабатываются примерно за одинаковое время. В таблице 4.1 приведена статистика выполнения заявок в зависимости от количества заявок: указано среднее время выполнения заявки (время, которое заявка занимала у всех линий конвейера), среднее время ожидания в очереди и среднее время пребывания в системе. В листинге 4.1 приведён лог параллельной обработки восьми заявок.

Таблица 4.1: Таблица времени выполнения параллельной обработки данных (в мс)

Количество задач	t в очереди	t в системе	t обработки
100	213.990	224.430	10.440
200	442.710	453.185	10.475
300	667.347	677.813	10.467
400	907.885	918.395	10.510
500	957.998	967.982	9.984
600	1370.842	1381.335	10.493
700	1587.126	1597.559	10.433
800	1802.399	1812.965	10.566
900	2022.898	2033.367	10.469
1000	2275.780	2286.326	10.546

Листинг 4.1: Лог параллельной обработки восьми заявок

1	Task	1		Part	1		Start		13:09:10.629
2	Task	1		Part	1		End		13:09:10.855
3	Task	2		Part	1		Start		13:09:10.855
4	Task	1		Part	2		Start		13:09:10.855
5	Task	1		Part	2		End		13:09:11. 27
6	Task	1		Part	3		Start		13:09:11. 27
7	Task	2		Part	1		End		13:09:11. 80
8	Task	3		Part	1		Start		13:09:11. 80
9	Task	2		Part	2		Start		13:09:11. 80
10	Task	1		Part	3		End		13:09:11.200
11	Task	2		Part	2		End		13:09:11.253
12	Task	2		Part	3		Start		13:09:11.253
13	Task	3		Part	1		End		13:09:11.307
14	Task	4		Part	1		Start		13:09:11.307
15	Task	3		Part	2		Start		13:09:11.307
16	Task	2		Part	3		End		13:09:11.426
17	Task	3		Part	2		End		13:09:11.480
18	Task	3		Part	3		Start		13:09:11.480
19	Task	4		Part	1		End		13:09:11.532
20	Task	5		Part	1		Start		13:09:11.532
21	Task	4		Part	2		Start		13:09:11.532
22	Task	3		Part	3		End		13:09:11.652
23	Task	4		Part	2		End		13:09:11.704
24	Task	4		Part	3		Start		13:09:11.704
25	Task	5		Part	1		End		13:09:11.758
26	Task	6		Part	1		Start		13:09:11.758
27	Task	5		Part	2		Start		13:09:11.758
28	Task	4		Part	3		End		13:09:11.878
29	Task	5		Part	2		End		13:09:11.931
30	Task	5		Part	3		Start		13:09:11.931
31	Task	6		Part	1		End		13:09:11.986
32	Task	7		Part	1		Start		13:09:11.986
33	Task	6		Part	2		Start		13:09:11.986
34	Task	5		Part	3		End		13:09:12.107
35	Task	6		Part	2		End		13:09:12.161
36	Task	6		Part	3		Start		13:09:12.161
37	Task	7		Part	1		End		13:09:12.217
38	Task	8		Part	1		Start		13:09:12.217
39	Task	7		Part	2		Start		13:09:12.217
40	Task	6		Part	3		End		13:09:12.338
41	Task	7		Part	2		End		13:09:12.393
42	Task	7		Part	3		Start		13:09:12.393
43	Task	8		Part	1		End		13:09:12.449
44	Task	8		Part	2		Start		13:09:12.449
45	Task	7		Part	3		End		13:09:12.566
46	Task	8		Part	2		End		13:09:12.621
47	Task	8		Part	3		Start		13:09:12.621
48	Task	8		Part	3		End		13:09:12.793

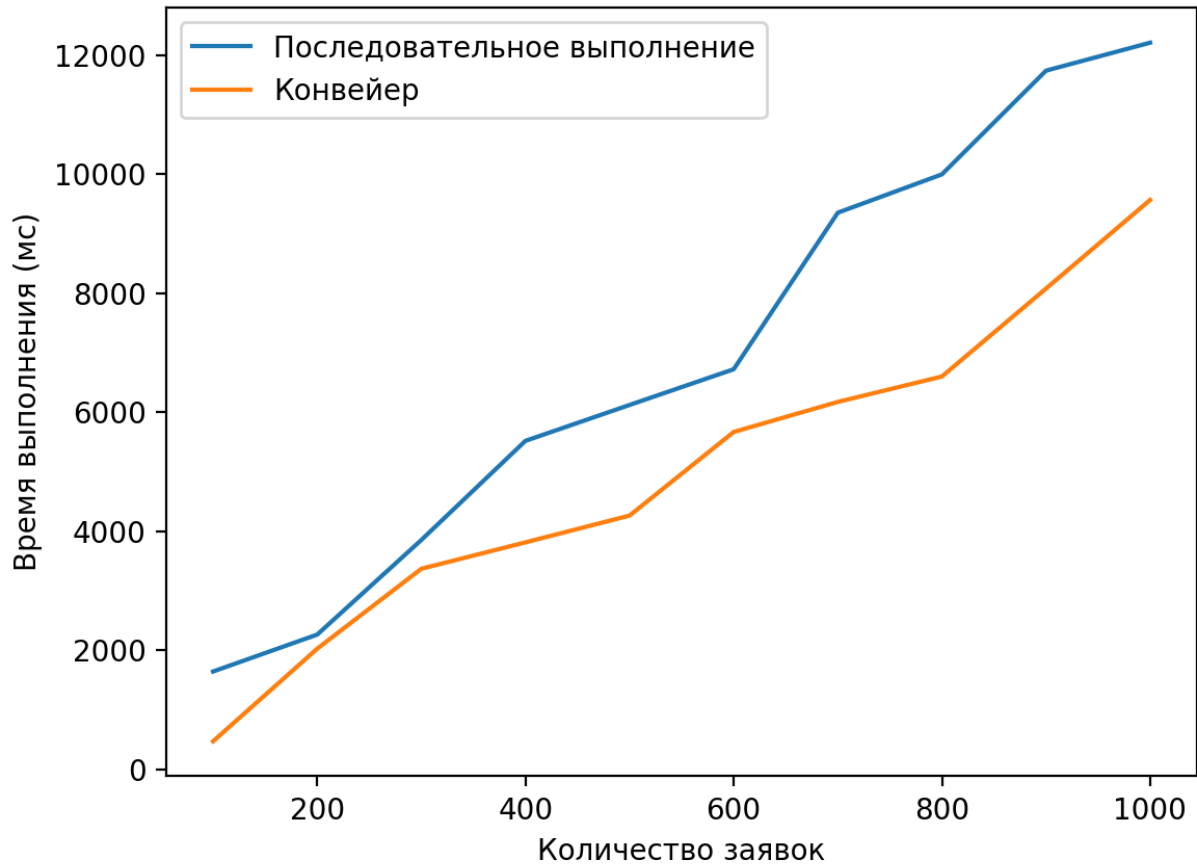


Рис. 4.1: Сравнение времени выполнения параллельной и последовательной обработки данных (в мс).

Вывод

В результате исследования было выяснено, что многопоточная реализация конвейера работает быстрее, чем линейная. Кроме того, с увеличением количества заявок растёт и время ожидания в очереди к линиям конвейера.

Заключение

В рамках данной лабораторной работы были решены следующие задачи:

- изучена конвейерная обработка данных;
- разработан метод конвейерной обработки данных на примере приготовления печенья;
- реализована система конвейерных вычислений;
- было проведено сравнение параллельной и линейной реализаций конвейерных вычислений на основе собранной статистики.

Параллельные конвейерные вычисления позволяют организовать непрерывную обработку данных, что позволяет выиграть время в задачах, где требуется обработка больших объемов данных за малый промежуток времени.

Поставленная цель была достигнута.

Литература

1. Стандартный набор функций для получения значения времени [Электронный ресурс].
Режим доступа: <https://en.cppreference.com/w/cpp/chrono>. Дата обращения: 16.11.2021