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, искусственный интеллект и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

Отчет по лабораторной работе №5

по курсу «Анализ алгоритмов»

«Конвейерные вычисления»

Студент группы **ИУ7-54Б** _____ **Чепиго Д.С.**

(Подпись, дата) (И.О. Фамилия)

Преподаватели _____ **Волкова Л.Л.**

(Подпись, дата) (И.О. Фамилия)

_____ **Строганов Ю.В.**

(Подпись, дата) (И.О. Фамилия)

***2022** г.*

Содержание

ВВЕДЕНИЕ

Целью данной лабораторной работы является изучение конвейерных вычислений.

При обработке данных могут возникать ситуации, когда один набор данных необходимо обработать последовательно несколькими алгоритмами. В таком случае удобно использовать конвейерную обработку данных, что позволяет на каждой следующей «линии» конвейера использовать данные, полученные с предыдущего этапа. Отдельно стоит упомянуть асинхронные конвейерные вычисления. Отличие от линейных состоит в том, что при таком подходе линии работают с меньшим временем простоя, так как могут обрабатывать задачи независимо от других линий.

Задачи лабораторной работы:

— рассмотреть и изучить асинхронную конвейерную обработку данных;
— реализовать систему конвейерных вычислений с тремя линиями;
— провести сравнительный анализ по времени последовательной и конвейерной реализаций;
— подготовить отчет по лабораторной работе.

1 Аналитическая часть

1.1 Предметная область лабораторной работы

В качестве алгоритма, реализованного для распределения на конвейере, было выбрано абстрактное оформление банковской карты, состоящее из трех этапов:

— генерация данных о владельце карты (ФИО, пол);
— генерация данных о карте (платежная система, номер карты в зависимости от платежной системы, CVV код);
— запись сгенерированных данных в CSV-файл.

Каждый из описанных выше этапов будет выполняться на отдельной ленте.

1.2 Последовательный алгоритм

При последовательном алгоритме создание нескольких банковских карт не может происходить одновременно. То есть, пока новая банковская карта не пройдет все три линии, то новая не начнет обрабатываться.

1.3 Параллельный алгоритм

В случае параллельного для каждой ленты создается отдельный поток. Извлечение и добавление банковских карт осуществляется потоками. Так, в параллельном алгоритме:

— первый поток извлекает карту из первой очереди, она обрабатывается на первой ленте, и поток добавляет ее во вторую очередь;
— второй поток извлекает карту из второй очереди, она обрабатывается на второй ленте, и поток добавляет ее в третью очередь;
— третий поток извлекает карту из второй очереди, и она обрабатывается на третьей ленте;
— каждый поток по завершении вновь извлекает карту из своей очереди, и цикл обработки повторяется в параллельном режиме.

2 Конструкторская часть

Разработка алгоритма оформления банковской карты

На рисунке 1 приведена схема последовательного алгоритма оформления банковской карты.

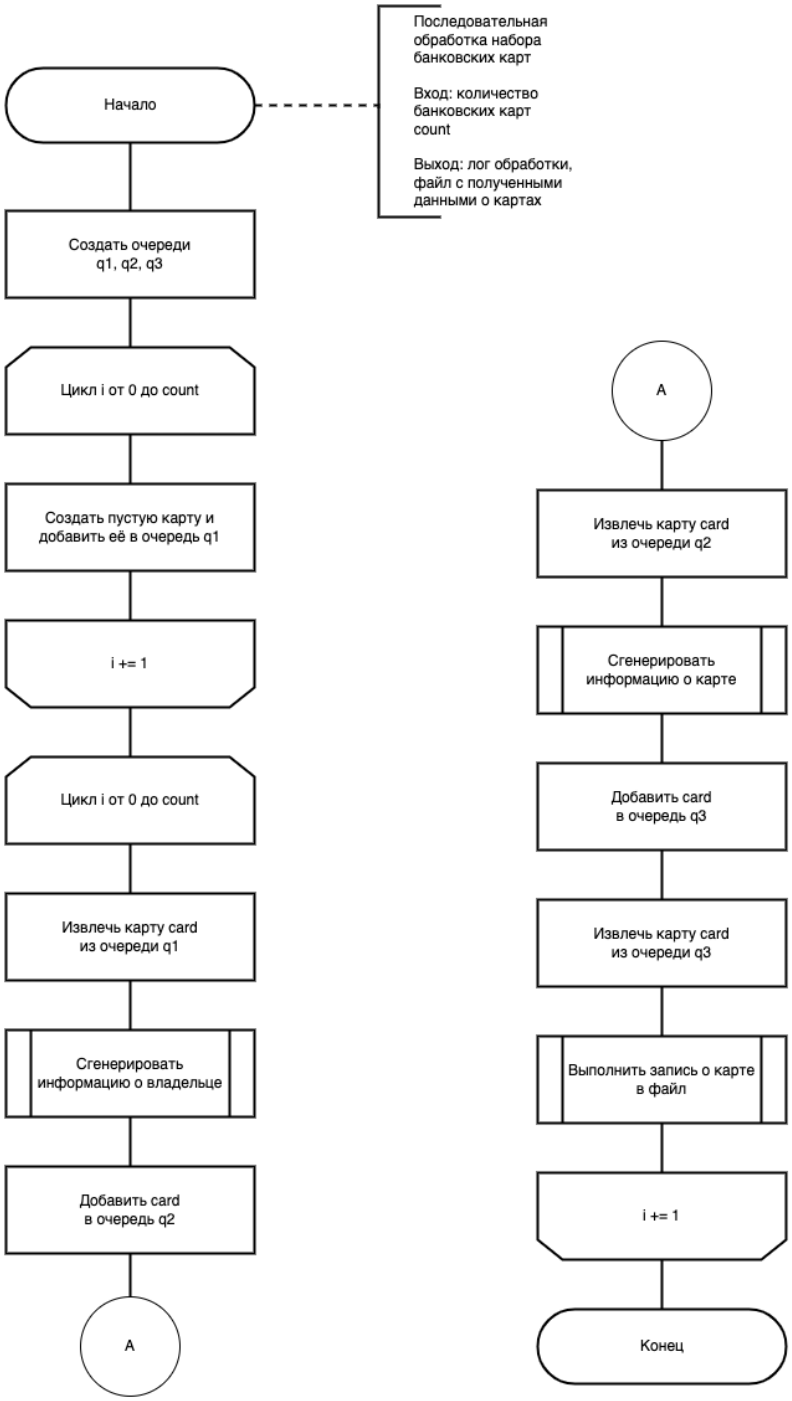

Рисунок 1 – Схема последовательного алгоритма оформления банковской карты

На рисунке 2 приведена схема параллельного алгоритма оформления банковской карты.

Рисунок 2 – Схема параллельного алгоритма оформления банковской карты

3 Технологическая часть

3.1 Требования к программному обеспечению

К программе предъявляется ряд требований:

— на вход подается количество карт, которое надо создать – целое положительное число;

— на выходе – время, затраченное на обработку карт;

— в процессе обработки задач необходимо фиксировать время прихода и ухода карты с линии.

3.2 Средства реализации

В качестве языка программирования для реализации лабораторной работы был выбран C++ – компилируемый, статически типизированный язык программирования общего назначения [1].

Данный выбор обусловлен поддержкой языком парадигмы объектно – ориентированного программирования и наличием методов для замера процессорного времени.

Время работы реализованных алгоритмов было замерено с помощью библиотеки chrono [2].

3.3 Реализация алгоритмов

В листинге 1 приведена реализация последовательного алгоритма оформления банковской карты.

Листинг 1 – Реализация последовательного алгоритма оформления банковской карты

```
1  void linearGenerate(vector<shared_ptr<Card> > &cards,
2                      ofstream &fout)
3  {
4      queue<shared_ptr<Card> > q1;
5      queue<shared_ptr<Card> > q2;
```

```
6       queue<shared_ptr<Card> > q3;
7       cout << "\nПОСЛЕДОВАТЕЛЬНАЯ ОБРАБОТКА\n";
8       for (int i = 0; i < cards.capacity(); i++)
9           q1.push(cards[i]);
10      cout << "\nЛОГ:\n";
11
12      for (int i = 0; i < cards.capacity(); i++)
13      {
14          double result = 0;
15          shared_ptr<Card> tempCard = q1.front();
16
17          auto start = high_resolution_clock::now();
18          tempCard->generateOwner();
19          auto end = high_resolution_clock::now();
20
21          q2.push(tempCard);
22          q1.pop();
23
24          result = chrono::duration_cast<chrono::nanoseconds>
25                  (end - start).count();
26          printf("Заявка: %d  Этап: 1  Начало: %g  Конец: %g\n",
27          i + 1, cur_time, cur_time + result);
28          cur_time += result;
29
30          tempCard = q2.front();
31          start = high_resolution_clock::now();
32          tempCard->generateCard();
33          end = high_resolution_clock::now();
34
35          q3.push(tempCard);
36          q2.pop();
37
38          result =
39          chrono::duration_cast<chrono::nanoseconds>
40          (end - start).count();
```

```
41            printf("Заявка: %d  Этап: 2  Начало: %g  Конец: %g\n",
42            i + 1, cur_time, cur_time + result);
43            cur_time += result;
44
45            tempCard = q3.front();
46            start = high_resolution_clock::now();
47            tempCard->dumpFile(fout);
48            end = high_resolution_clock::now();
49
50            q3.pop();
51
52            result = (chrono::duration_cast<chrono::nanoseconds>
53            (end - start).count());
54            printf("Заявка: %d  Этап: 3  Начало: %g  Конец: %g\n",
55            i + 1, cur_time, cur_time + result);
56            cur_time += result;
57        }
58    }
```

В листинге 2 приведена реализация параллельного алгоритма оформления банковской карты.

Листинг 2 – Реализация параллельного алгоритма оформления банковской карты

```
void parallelGenerate(vector<shared_ptr<Card> > &cards,
                      ofstream &fout)
{
    queue<shared_ptr<Card> > q1;
    queue<shared_ptr<Card> > q2;
    queue<shared_ptr<Card> > q3;
    cout << "\nПАРАЛЛЕЛЬНАЯ ОБРАБОТКА\n";
    for (int i = 0; i < cards.capacity(); i++)
        q1.push(cards[i]);
    cout << "\nЛОГ:\n";
    for (int i = 0; i < cards.capacity() + 1; i++)
    {
        cur_time1.push_back(0);
        cur_time2.push_back(0);
        cur_time3.push_back(0);
    }

    vector<thread> threads(3);

    threads[0] = thread(first, std::ref(q1), std::ref(q2));
    threads[1] = thread(second, std::ref(q1),
                        std::ref(q2), std::ref(q3));
    threads[2] = thread(third, std::ref(q1), std::ref(q2),
                        std::ref(q3), std::ref(fout));
    threads[0].join();
    threads[1].join();
    threads[2].join();
}
```

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялся замерный эксперимент:

— операционная система macOS Ventura 13.0.1 [4];

— память 32 ГБ;

— процессор 2,3 ГГц 4-ядерный процессор Intel Core i7.

Замеры проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, окружением, а также непосредственно замерным экспериментом.

4.2 Пример работы программы

На рисунке 3 представлен пример работы программы. Вводится количество кард, далее выводится лог при параллельной обработки и последовательной.

```
[d.chepigo@d-chepigo ~/D/b/a/l/s/code (develop)> ./a.out
Введите количество карт для изготовления: 5

ПОСЛЕДОВАТЕЛЬНАЯ ОБРАБОТКА

ЛОГ:
Заявка: 1  Этап: 1  Начало: 0  Конец: 5422
Заявка: 1  Этап: 2  Начало: 5422  Конец: 9238
Заявка: 1  Этап: 3  Начало: 9238  Конец: 40370
Заявка: 2  Этап: 1  Начало: 40370  Конец: 42496
Заявка: 2  Этап: 2  Начало: 42496  Конец: 45120
Заявка: 2  Этап: 3  Начало: 45120  Конец: 48227
Заявка: 3  Этап: 1  Начало: 48227  Конец: 50329
Заявка: 3  Этап: 2  Начало: 50329  Конец: 52366
Заявка: 3  Этап: 3  Начало: 52366  Конец: 54831
Заявка: 4  Этап: 1  Начало: 54831  Конец: 56840
Заявка: 4  Этап: 2  Начало: 56840  Конец: 59052
Заявка: 4  Этап: 3  Начало: 59052  Конец: 61701
Заявка: 5  Этап: 1  Начало: 61701  Конец: 63472
Заявка: 5  Этап: 2  Начало: 63472  Конец: 65437
Заявка: 5  Этап: 3  Начало: 65437  Конец: 68004

ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА

ЛОГ:
Заявка: 1  Этап: 1  Начало: 0  Конец: 18632
Заявка: 1  Этап: 3  Начало: 24173  Конец: 46296
Заявка: 2  Этап: 1  Начало: 18632  Конец: 23175
Заявка: 3  Этап: 1  Начало: 23175  Конец: 32798
Заявка: 4  Этап: 1  Начало: 32798  Конец: 34953
Заявка: 5  Этап: 1  Начало: 34953  Конец: 36880
Заявка: 1  Этап: 2  Начало: 18632  Конец: 24173
Заявка: 2  Этап: 2  Начало: 23175  Конец: 26579
Заявка: 3  Этап: 2  Начало: 26579  Конец: 28854
Заявка: 2  Этап: 3  Начало: 26579  Конец: 30588
Заявка: 3  Этап: 3  Начало: 28854  Конец: 31763
Заявка: 4  Этап: 3  Начало: 31763  Конец: 34322
Заявка: 4  Этап: 2  Начало: 28854  Конец: 31505
Заявка: 5  Этап: 2  Начало: 31505  Конец: 34466
Заявка: 5  Этап: 3  Начало: 34466  Конец: 37130
```

Рисунок 3 – Пример работы программы

4.3 Время выполнения реализованных алгоритмов

В таблице 1 представлены результаты замера времени реализованных алгоритмов. На рисунке 4 представлен график зависимости.

Таблица 1 – Результаты замеров времени реализованных сортировок в наноосекундах

Количество карт	Последовательная	Параллельная
10	102686	57213
20	110647	60790
30	126516	64714
40	137995	69191
50	151877	91153

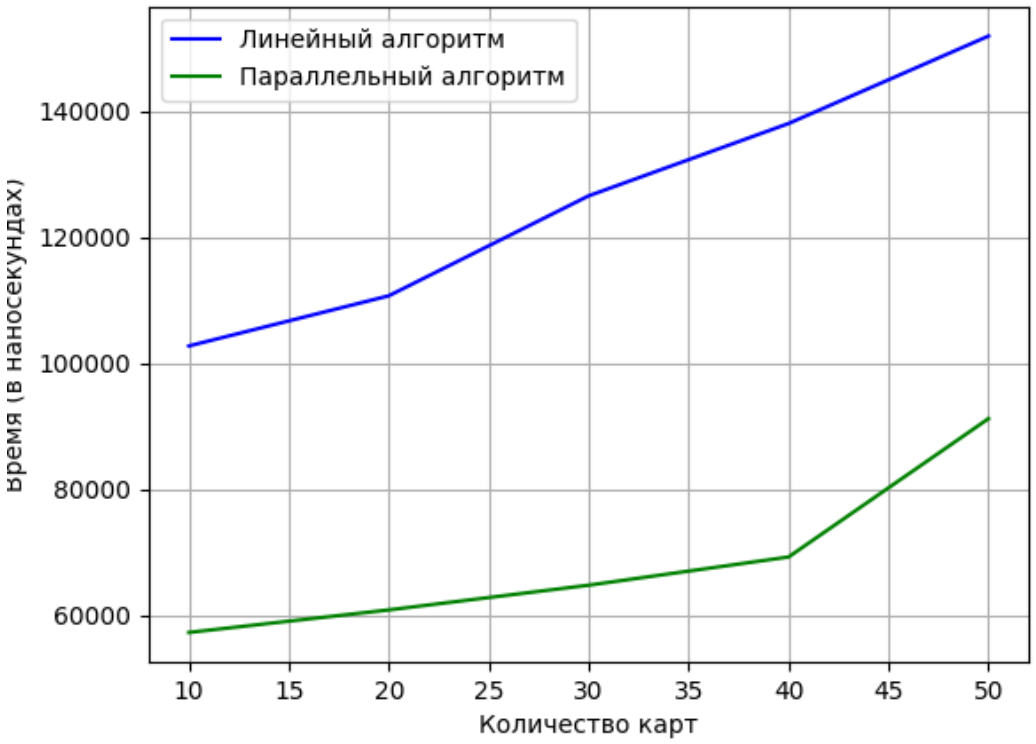

Рисунок 4 – Зависимость времени работы реализованных алгоритмов от количества карт

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы поставленная цель была достигнута: были изучены конвейерные вычисления.

В ходе выполнения лабораторной работы были решены все задачи:

— рассмотрены и изучить асинхронную конвейерную обработку данных;
— реализована система конвейерных вычислений с тремя линиями;
— проведен сравнительный анализ по времени последовательной и конвейерной реализаций;
— был подготовлен отчет по лабораторной работе.

В ходе исследования разницы работы реализованных алгоритмов можно сделать вывод, что параллельная реализация алгоритма работает быстрее в среднем в 2 раза.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Язык программирования C++ [Электронный ресурс]. – Режим доступа: https://isocpp.org/, свободный – (20.10.2022)

2. Стандарт языка C++ [Электронный ресурс]. – Режим доступа: https://isocpp.org/files/papers/N4860.pdf, свободный – (20.10.2022)

3. Стандарт языка Си [Электронный ресурс]. – Режим доступа: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf, свободный – (20.10.2022)

4. Операционная система macOS Ventura [Электронный ресурс]. – Режим доступа: https://www.apple.com/macos/ventura/, свободный – (06.11.2022)