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

Название: Конвейеры вычисления

Дисциплина: Анализ алгоритмов

Студент ИУ7-54Б _____ Л.Е.Тартыков
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Л.Л. Волкова
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Содержание

Введение

Разработчики архитектуры компьютеров издавна прибегали к методам проектирования, известным под общим названием "совмещение операций при котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции. Этот общий метод включает два понятия: параллелизм и конвейеризацию. Хотя у них много общего и их зачастую трудно различать на практике, эти термины отражают два совершенно различных подхода. При параллелизме совмещение операций достигается путем воспроизведения в нескольких копиях аппаратной структуры. Высокая производительность достигается за счет одновременной работы всех элементов структур, осуществляющих решение различных частей задачи.

Конвейеризация (или конвейерная обработка) в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые ступенями, и выделении для каждой из них отдельного блока аппаратуры. Так обработку любой машинной команды можно разделить на несколько этапов (несколько ступеней), организовав передачу данных от одного этапа к следующему. При этом конвейерную обработку можно использовать для совмещения этапов выполнения разных команд. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд. Конвейерная обработка такого рода широко применяется во всех современных быстродействующих процессорах.

Целью данной работы является получение навыка организации асинхронной передачи данных между потоками на примере конвейерной обработки информации.

Задачи данной лабораторной работы:

- Выбрать и описать методы обработки данных, которые будут сопоставлены методам конвейера;
- описать архитектуру программы, а именно какие функции имеет главный поток, принципы и алгоритмы обмена данными между потоками;

- реализовать конвейерную систему, а также сформировать лог событий с указанием времени их происхождения, описать реализацию;
- интерпретировать сформированный лог.

1 Аналитический раздел

1.1 Постановка задачи

Конвейер – машина непрерывного транспорта [1], предназначенная для перемещения сыпучих, кусковых или штучных грузов.
Конвейерное производство - система поточной организации производства на основе конвейера, при которой оно разделено на простейшие короткие операции, а перемещение деталей осуществляется автоматически. Это такая организация выполнения операций над объектами, при которой весь процесс воздействия разделяется на последовательность стадий с целью повышения производительности путём одновременного независимого выполнения операций над несколькими объектами, проходящими различные стадии. Конвейером также называют средство продвижения объектов между стадиями при такой организации[2]. Появилось в 1914 году на производстве Модели-Т на заводе Генри Форда[3] и произвело революцию сначала в автомобилестроении, а потом и во всей промышленности.

1.2 Параллельное программирование

Параллельные вычисления — способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (одновременно).

При использовании многопроцессорных вычислительных систем с общей памятью обычно предполагается, что имеющиеся в составе системы процессоры обладают равной производительностью, являются равноправными при доступе к общей памяти, и время доступа к памяти является одинаковым (при одновременном доступе нескольких процессоров к одному и тому же элементу памяти очередность и синхронизация доступа обеспечивается на аппаратном уровне). Многопроцессорные

системы подобного типа обычно именуются симметричными мультипроцессорами (symmetric multiprocessors, SMP).

Перечисленному выше набору предположений удовлетворяют также активно развиваемые в последнее время многоядерные процессоры, в которых каждое ядро представляет практически независимо функциони рующее вычислительное устройство.

Обычный подход при организации вычислений для многопроцессорных вычислительных систем с общей памятью – создание новых параллельных методов на основе обычных последовательных программ, в которых или автоматически компилятором, или непосредственно программистом выделяются участки независимых друг от друга вычислений. Возможности автоматического анализа программ для порождения параллельных вычислений достаточно ограничены, и второй подход является преобладающим. При этом для разработки параллельных программ могут применяться как новые алгоритмические языки, ориентированные на параллельное программирование, так и уже имеющиеся языки, расширенные некоторым набором операторов для параллельных вычислений.

Широко используемый подход состоит и в применении тех или иных библиотек, обеспечивающих определенный программный интерфейс (application programming interface, API) для разработки параллельных программ. В рамках такого подхода наиболее известны Windows Thread API. Однако первый способ применим только для ОС семейства Microsoft Windows, а второй вариант API является достаточно трудоемким для использования и имеет низкоуровневый характер [5].

1.2.1 Организация взаимодействия параллельных потоков

Потоки исполняются в общем адресном пространстве параллельной программы. Как результат, взаимодействие параллельных потоков можно организовать через использование общих данных, являющихся доступными для всех потоков. Наиболее простая ситуация состоит в ис-

пользовании общих данных только для чтения. В случае же, когда общие данные могут изменяться несколькими потоками, необходимы специальные усилия для организации правильного взаимодействия.

1.3 Организация обработки

У каждой линии конвейера есть очередь элементов. Когда линия еще активна, но элементов в очереди нет, линия уходит в режим ожидания. По прошествию заданного времени линия проверяет не появились ли новые элементы в очереди. Если очередь не пустая, то нужно получить и обработать элемент, передать его следующей линии, если такая существует.

1.4 Вывод

В данном разделе были рассмотрены основы конвейерной обработки, технология параллельного программирования и организация взаимодействия параллельных потоков.

2 Конструкторский раздел

В данном разделе будут рассмотрена схема работы алгоритма конвейерной обработки, описание архитектуры ПО и схемы рабочего и главного процессов.

2.1 Описание архитектуры ПО

В конвейере 3 основные ленты, содержание их работы описано выше, в аналитической части отчета. Каждой ленте выделен свой поток, в котором она выполняется. В главном потоке (функция main) создаются три рабочих потока: по одному на каждую ленту. Для каждой ленты есть своя очередь, однако с ней могут работать все потоки, поэтому при доступе к элементам очереди необходимо блокировать доступ для других потоков. Для реализации доступа из разных потоков используются мьютексы, по одному для каждой очереди и для результирующего массива. Также в главном потоке генерируется массив входных данных и заполняется первая очередь (уровень 0). Для моделирования ситуации постепенного поступления данных, первая очередь заполняется с задержкой. В рабочих процессах считывается по одному элементу из соответствующей очереди, выполняется вызов обрабатывающих функций, замеры времени и задержка по времени. После обработки текущей задачи, рабочий процесс записывает результат в следующую очередь или в результирующий массив. Также выполняется запись в лог—файл.

2.2 Схема алгоритмов работы конвейерной обработки

На рисунке 2.1 представлена схема алгоритма выполнения главного потока.

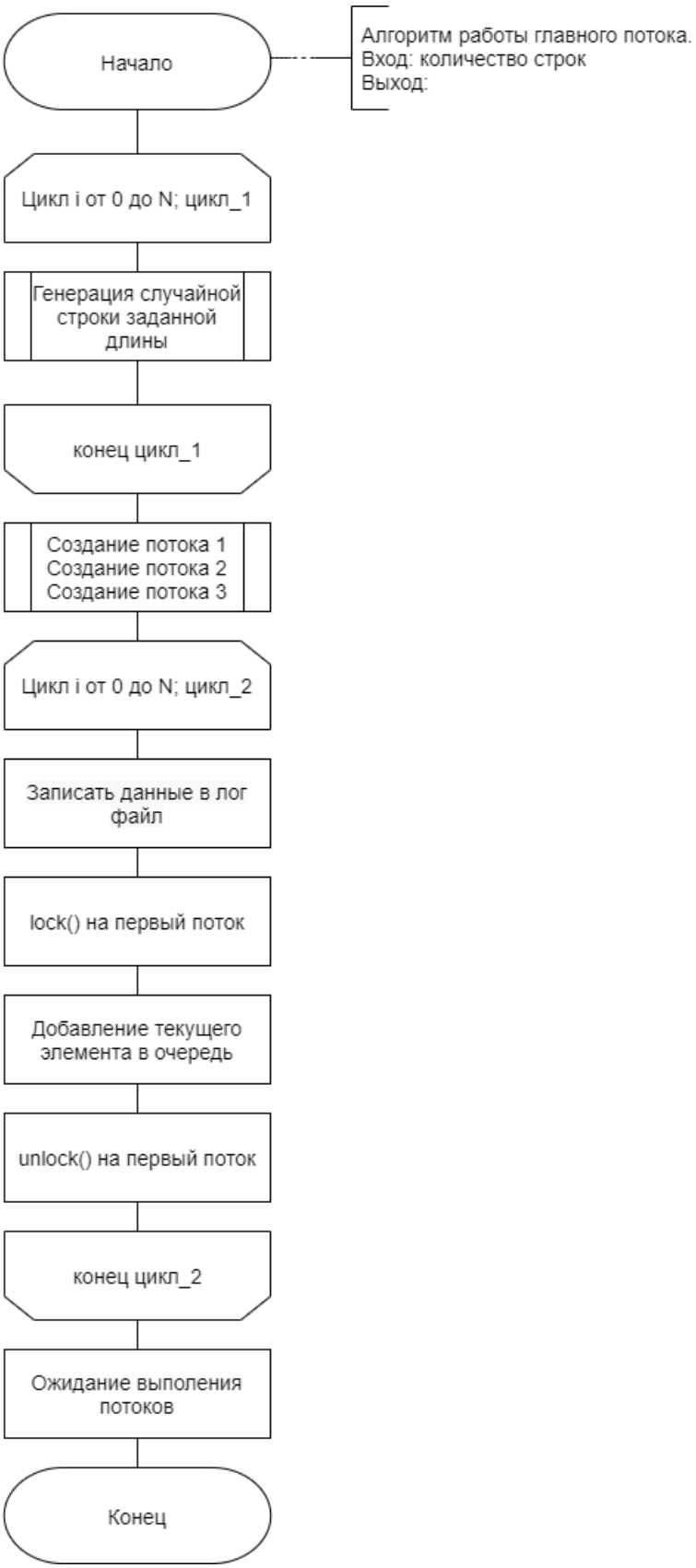

Рисунок 2.1 – Схема алгоритма выполнения главного потока

На рисунке 2.2 представлен схема алгоритма выполнения дочерних потоков.

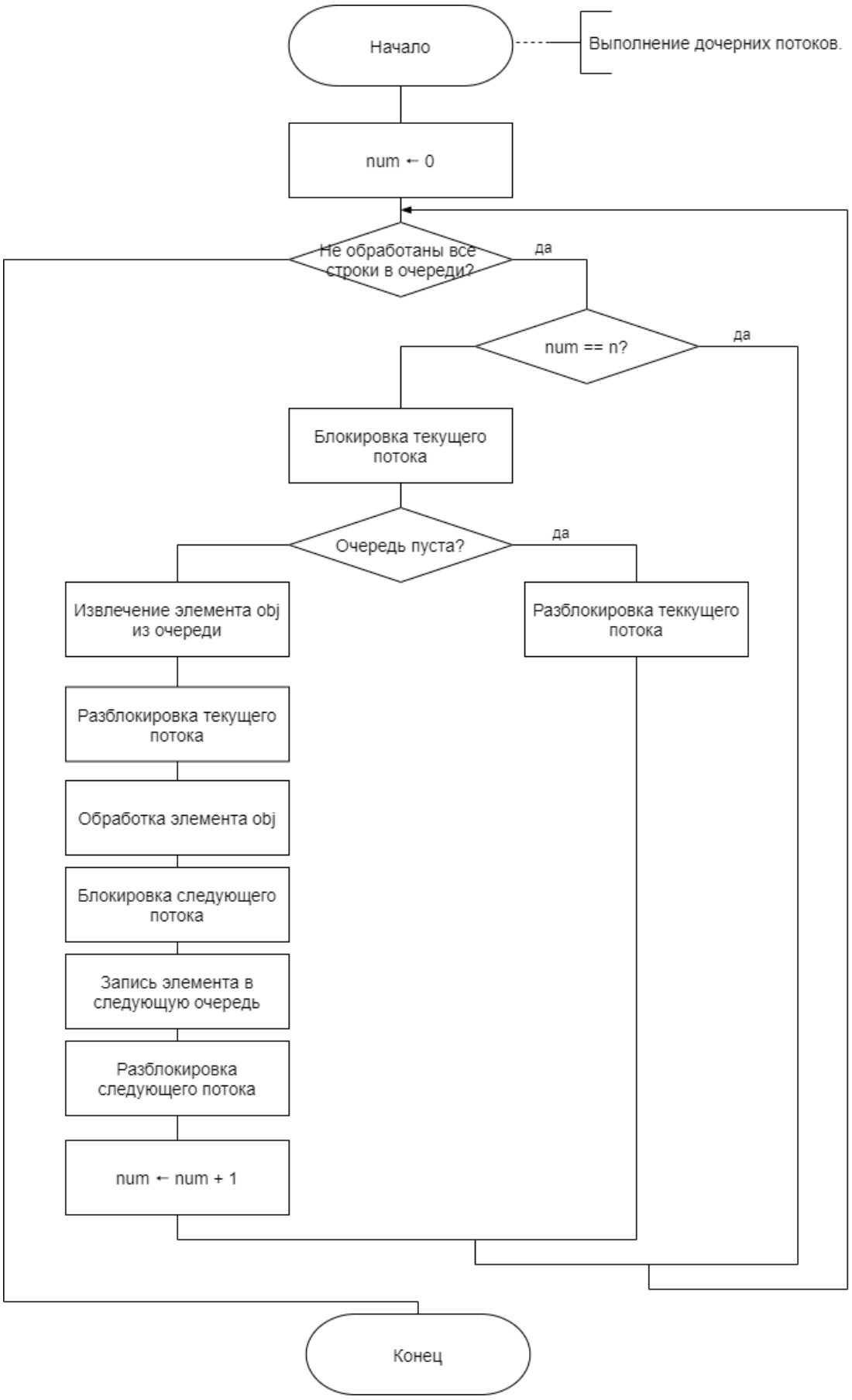

Рисунок 2.2 – Схема алгоритма выполнения дочерних потоков

2.3 Классы эквивалентности

Классы эквивалентности для тестирования:

- пустые строки;
- количество строк на вход конвейера $<= 0$;
- строки из цифр, знаков препинания и других символов;
- корректные входные значения.

2.4 Используемые типы данных

В программе используются следующие типы данных:

- строка - структура типа $input_t\_t$, являющимся переопределением типа std::string;
- очереди имеют тип $std :: queue < input\_t >$;
- переменные потоков thread1, thread2, thread3 - типа $std :: thread$;
- время выполнения этапов обработки каждой матрицы записывается в переменные типа $clock\_t$
- для записи данных в файл используется тип $FILE*$;
- для исключения коллизий во время работы потоков применяются мьютексы $std :: mutex$.

2.5 Вывод

В данном разделе была описана программа с 3 конвейерными потоками были приведены схемы алгоритмов используемых в программе описаны типы используемые в программе и классы эквивалентности для тестовых данных.

3 Технологический раздел

В данном разделе будут рассмотрены средства реализации и представлен листинг кода.

3.1 Требования к програмному обеспечению

Требования к программе при параллельной обработке:

- объекты должны последовательно проходить конвейеры в заданном подядке;
- конвейеры должны работать каждый в своем потоке;
- конвейер должен завершать свою работу при поступлении специального элемента;
- до завершения работы конвейер должен ожидать поступления новых элементов.

3.2 Средства реализации

В качестве языка программирования был выбран C++ т.к. имеется опыт разработки на нем. Важной особенностью языка C++ является его высокая вычислительная производительность наряду с другими ЯП и возможность использования технологии параллельного программирования.[6]. Среда разработки – QtCreator, которая предоставляет умную проверку кода, быстрое выявление ошибок и оперативное исправление, вместе с автоматическим рефакторингом кода и богатыми возможностями в навигации[4]. Время работы было замерено с помощью функции clock() из библиотеки ctime. [2] Для тестирования использовался компьютер на базе процессора Intel(R) Core(TM) i3, 4 логических ядра.

3.3 Листинг кода алгоритмов

В данном пункте представлен листинг кода функций. В листинге 1, 2, 3 представлен код рабочих потоков. В листинге 4 представлен код главного потока.

Листинг 3.1 – Реализация первого уровня конвейера

```
void conveyor1(){
    int num = 0;
    while (1)
    {
        if (num == n)
        break;
        m1.lock();
        if (queue1.empty())
        {
            m1.unlock();
            continue;
        }
        input_t myObj = queue1.front();
        queue1.pop();
        m1.unlock();
        input_t newObj = caesar(myObj);
        m2.lock();
        queue2.push(newObj);
        sleep(1);
        clock_t time = clock();
        log.print(1, newObj, num, time);
        m2.unlock();
        num++;
    }
}
```

Листинг 3.2 – Реализация второго уровня конвейера

```
void conveyor2(){
    int num = 0;
    while (1)
    {
        if (num == n)
        break;
        m2.lock();
        if (queue2.empty())
        {
            m2.unlock();
            continue;
        }
        input_t myObj = queue2.front();
        queue2.pop();
        m2.unlock();
        input_t newObj = upper_lower(myObj);
        m3.lock();
        queue3.push(newObj);
        sleep(3);
        clock_t time = clock();
        log.print(2, newObj, num, time);
        m3.unlock();
        num++;
    }
}
```

Листинг 3.3 – Реализация третьего уровня конвейера

```
void conveyor3(){
    int num = 0;
    while (1)
    {
        if (num == n)
        break;
        m3.lock();
        if (queue3.empty())
        {
            m3.unlock();
            continue;
        }
        input_t myObj = queue3.front();
        queue3.pop();
        m3.unlock();
        input_t newObj = reverse(myObj);
        resm.lock();
        res.push_back(newObj);
        sleep(1.5);
        clock_t time = clock();
        log.print(3, newObj, num, time);
        resm.unlock();
        num++;
    }
}
```

Листинг 3.4 – Основная функция программы

```
int main(){
    f = fopen("log.txt", "w");
    n = 5;
    objvec.resize(n);

    thread t1(conveyor1);
    thread t2(conveyor2);
    thread t3(conveyor3);
    main_time = clock();

    for (int i = 0; i < n; i++)
    {
        string s = generate();
        objvec[i] = (s);
    }

    for (int i = 0; i < n; i++)
    {
        clock_t tm = clock();
        log.print(0, objvec[i], i, tm);
        m1.lock();

        queue1.push(objvec[i]);
        m1.unlock();
        sleep(2);
    }

    t1.join();
    t2.join();
    t3.join();
    fclose(f);
    printf("That's all folks!");
    return 0;
}
```

Вывод

В данном разделе были рассмотрены основные сведения о модулях программы и листинг кода алгоритмов.

4 Исследовательская часть

В данном разделе будет проведены испытания программы и проведен анализ полученных данных.

4.1 Эксперимент

В результате работы программы был составлен лог:

Листинг 4.1 – Экспериментальные данные

```
1        step: 0 | item: 0 | time: 3 (3) | value:
            HgUeyNFXICSxgWfQUWNOVrKPEGxPRYtwyycpvkfMNmksWrnKcF
2        step: 1 | item: 0 | time: 1056 (1053) | value:
            IhVfzOGYJDTyhXgRVXOPWsLQFHyQSZuxzzdqwlgNOnltXsoLdG
3        step: 0 | item: 1 | time: 2061 (1005) | value:
            TSySAPofZCeUPAOgOyFBPJRIYmeWnrpxjLoAwxhsCXosZvtDnY
4        step: 1 | item: 1 | time: 3073 (1012) | value:
            UTzTBQpgADfVQBPhPzGCQKSJZnfXosqykMpBxyitDYptAwuEoZ
5        step: 0 | item: 2 | time: 4072 (999) | value:
            YFxSnkfUXRBngoktYNouzrdItijayPSmSjFGbAoLYNRdpsTsWD
6        step: 2 | item: 0 | time: 4072 (999) | value:
            iHvFZogyjdtYHxGrvxopwSlqfhYqszUXZZDQWLGnoNLTxSOlDg
7        step: 1 | item: 2 | time: 5083 (1011) | value:
            ZGyTolgVYSCohpluZOpvaseJujkbzQTnTkGHcBpMZOSeqtUtXE
8        step: 3 | item: 0 | time: 5083 (1011) | value:
            hYqszUXZZDQWLGnoNLTxSOlDgiHvFZogyjdtYHxGrvxopwSlqf
9        step: 0 | item: 3 | time: 6083 (1000) | value:
            zoZnWQjVWXRbDGlgUbbrTHspOGgKuDOObxEaFtueZhlUgejQJn
10       step: 1 | item: 3 | time: 7097 (1014) | value:
            apAoXRkWXYScEHmhVccsUItqPHhLvEPPcyFbGuvfAimVhfkRKo
11       step: 2 | item: 1 | time: 8091 (994) | value:
            utZtbqPGadFvqbpHpZgcqksjzNFxOSQYKmPbXYITdyPTaWUeOz
12       step: 0 | item: 4 | time: 8091 (994) | value:
            EQxeJrZAULCHFmiwOIyXUnSsSTkqCzEuMrczjpIneDSRAYmABZ
13       step: 1 | item: 4 | time: 9107 (1016) | value:
            FRyfKsABVMDIGnjxPJzYVoTtTUlrDaFvNsdakqJofETSBZnBCA
14       step: 2 | item: 2 | time: 11106 (1999) | value:
            zgYtOLGvyscOHPLUzoPVASEjUJKBZqtNtKghCbPmzosEQTuTxe
15       step: 2 | item: 3 | time: 14109 (3003) | value:
            APaOxrKwxysCehMHvCCSuiTQphHlVeppCYfBgUVFaIMvHFKrkO
16       step: 2 | item: 4 | time: 17126 (3017) | value:
            frYFkSabvmdigNJXpjZyvOtTtuLRdAfVnSDAKQjOFetsbzNbca
```

```
17          step: 3 | item: 1 | time: 18138 (1012) | value:
               NFxOSQYKmPbXYITdyPTaWUeOzutZtbqPGadFvqbpHpZgcqksjz
18          step: 3 | item: 2 | time: 19150 (1012) | value:
               JKBZqtNtKghCbPmzosEQTuTxezgYtOLGvyscOHPLUzoPVASEjU
19          step: 3 | item: 3 | time: 20163 (1013) | value:
               hHlVeppCYfBgUVFaIMvHFKrkOAPaOxrKwxysCehMHvCCSuiTQp
20          step: 3 | item: 4 | time: 21175 (1012) | value:
               uLRdAfVnSDAKQjOFetsbzNbcafrYFkSabvmdigNJXpjZyvOtTt
```

Полученный файл состоит из записей, отсортированных по времени по возрастанию с начала запуска приложения. В первом столбце указан номер конвейера, во втором столбце — номер текущего элемента, затем время с начала работы программы и время выполнения данного этапа в миллисекундах, а также значение хеша на текущем этапе. По листингу можно проследить выполнение каждого этапа и изменение значений хеша при обработке на каждой из лент. Также можно проследить, что обработка выполняется параллельно. Например, 2 линия не ждет полного завершения работы первой прежде чем начать работу, а начинает выполнение как только в очередь поступает первый элемент.

4.2 Вывод

Так как обработка на каждой линии занимает разное кол-во времени, в листинге можно увидеть, насколько она эффективнее. Если бы мы обрабатывали каждый элемент последовательно то потребовалось бы (1000 + 3000 + 1500) * 5 = 27500 миллисекунд. Однако алгоритм выполнился за 21175 миллисекунд, что дает выигрыш в 6.5 секунд. Также как можно видеть из логов каждый следующий этап обработки зависит от предыдущего, следовательно миномальное время работа алгоритма будет зависить от времени первого потока.

Заключение

В ходе лабораторной работы были выполнены следующие задачи:

- выбраны и описаны методы обработки данных;
- описана архитектура программы, а именно какие функции имеет главный поток, принципы и алгоритмы обмена данными между потоками;
- реализована конвейерная система обработки, а также сформирован лог событий с указанием времени их происхождения, описать реализацию;
- по сформированному логу был сформирован вывод.

Эксперимент подтвердил преимущество по времени данного метода над линейным. Был сделан вывод, что каждый следующий этап обработки зависит от предыдущего, следовательно миномальное время работа алгоритма будет зависить от времени работы первого этапа конвеера.

Список литературы

[1] Меднов В.П., Бондаренко Е.П. Транспортные, распределительные и рабочие конвейеры. М., 1970.

[2] Конвейерное производство[Электронный ресурс] - режим доступа https:/ë/dic.academic.ru/dic.nsf/ruwiki/1526795

[3] Конвейерный метод производства Генри Форда[Электронный ресурс] - режим доступа https://popecon.ru/305-konveiernyi-metod-proizvodstva-genri-forda.html

[4] Qt 5.12. [электронный ресурс], режим доступа: https://doc.qt.io/qt-5.12/index.html (дата обращения: 19.11.2021)

[5] Константин Баркалов, Владимир Воеводин, Виктор Гергель. Intel Parallel Programming [Электронный ресурс], - режим доступа https://www.intuit.ru/studies/courses/4447/983/lecture/14925

[6] C++11. [электронный ресурс], режим доступа: https://en.cppreference.com/w/cpp/11 (дата обращения 12.11.2021)