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»

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

Тема Конвейерные вычисления

Студент Малышев И. А.

Группа ИУ7-51Б

Оценка (баллы) _____

Преподаватель: Волкова Л. Л.

Москва — 2021 г.

Содержание

Введение

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.

Помимо линейной конвейерной обработки данных, существуют параллельная конвейерная обработка данных. При таком подходе все линии работают с меньшим времени простоя, так как могут обрабатывать задачи независимо от других линий.

Целью данной лабораторной работы является изучение и реализация параллельной и линейной реализации конвейерной обработки данных.

В рамках выполнения работы необходимо решить следующие **задачи**:

- изучить конвейерную обработку данных;
- реализовать систему конвейерных вычислений с количеством линий не меньше трёх;
- сравнить параллельную и линейную реализацию конвейерных вычислений;
- сделать выводы на основе проделанной работы.

1 Аналитическая часть

В данной части будут рассмотрены главные принципы конвейерной обработки и параллельных вычислений, а также описание входных и выходных данных, ограничений ПО и функциональных требований к ПО.

1.1 Общие сведения о конвейерной обработке

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.

В терминах программирования ленты конвейера представлены функциями, выполняющими над неким набором данных операции и предающие их на следующую ленту конвейера. Моделирование конвейерной обработки хорошо сочетается с технологией многопоточного программирования - под каждую ленту конвейера выделяется отдельный поток, все потоки работают в асинхронном режиме.

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 [8].

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

Вывод

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

Входные данные:

- количество ковеерных линий;
- минимальная и максимальная задержка;
- действие, совершаемое на конвеерных линиях.

Выходные данные:

- лог работы программы.

Ограничения программы:

- отсутствует интерактивный ввод входных данных: все данные вводятся в коде программы;
- для сообщения конвееру о завершении работы используется специальный элемент.

Функциональные требования к ПО:

- количество ковеерных линий должно быть больше нуля;
- объекты должны последовательно проходить конвеерные линии в заданном порядке;
- конвеерные линии должны завершать свою работу при поступлении специального элемента;
- до завершения работы конвеерная линия должна ожидать поступление новых элементов;
- при параллельной обработке конвеерные линии должны работать каждый на своём потоке.

2 Конструкторская часть

В данном разделе представлены схемы рассматриваемых алгоритмов, а также описание структур данных, способы тестирования, описание памяти для алгоритма и структура ПО.

2.1 Разработка алгоритмов

На рисунке 2.1 приведена схема организации конвейерных вычислений.

Рис. 2.1: Схема организации конвейерных вычислений.

2.2 Описание структур данных

В качестве структуры данных для описания конвеера используется связный список элементов, которые описывают конвеерную линию (класс конвеерной линии). Такая структура данных позволяет изменять размер конвеера во время работы программы, добавляя или удаляя конвеерные линии.

2.3 Способы тестирования

Конвееры делятся по способу обработки данных:

- параллельная обработка;
- линейная обработка.

Для каждого вида конвеера производится отдельное тестирование.

2.4 Описание памяти

Количество памяти, занимаемое конвеером, равно

$$volume = n * (sizeof(Line) + m * sizeof(Args))$$

, где n – число линий конвеера, m – число элементов в очереди линии, $sizeof()$ - операция поучения размера структуры данных в байтах.

2.5 Структура ПО

На рисунке 2.2 представлена диаграмма классов разрабатываемого ПО.

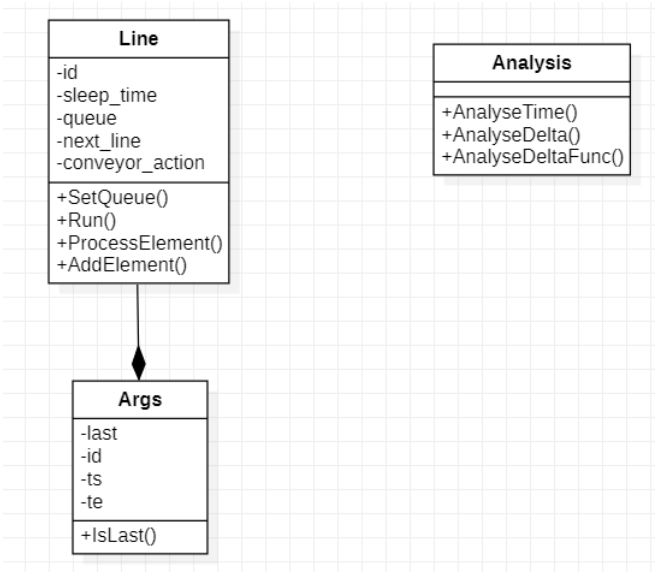

Рис. 2.2: Схема организации конвейерных вычислений.

Line – класс конвеерной линии. Содержит поля номера линии, времени задержки, очереди элементов класса **Args**, ссылки на следующую линию и действия линии.

Args – класс для хранения входных данных для линии. Содержит поля порядкового номера, времени начала и завершения работы линии с очередным элементом и булевое значение, характеризующее является ли данный элемент последним.

Analysis – статический класс для получения результатов анализа работы конвееров.

Вывод

В данном разделе была рассмотрена схема организации конвейерной обработки, а также описание структур данных, способы тестирования, описание памяти для алгоритма и структура ПО.

3 Технологическая часть

Замеры времени были произведены на: Intel(R) Core(TM) i7-4790K, 4 ядра, 8 логических ядер.

3.1 Средства реализации

В качестве языка программирования был выбран C# [9] так как этот язык поддерживает управление потоками на уровне ОС(незелёные потоки). В качестве среды разработки была выбрана Visual Studio. Время работы алгоритмов было замерено с помощью класса Stopwatch. Многопоточное программирование было реализовано с помощью пространства имен System.Threading.

3.2 Реализация алгоритмов

В этой части будут рассмотрены листинги кода (листинг 3.1 - 3.3) реализованых алгоритмов.

Листинг 3.1: Функция для запуска в потоке

```
1  public void Run()
2  {
3      state ret = state.ok;
4      while(ret != state.finish)
5      {
6          ret = ProcessElement();
7          if (ret == state.empty)
8              Thread.Sleep(500);
9      }
10 }
```

Листинг 3.2: Обработка элемента

```
public state ProcessElement()
{
    Args el = null;
    lock(q)
    {
        if (q.Count > 0)
        {
            el = q.Dequeue();
        }
    }

    if (el != null)
    {
        if (el.IsLast())
        {
            if (nextLine != null)
                nextLine.AddElement(el);
            return state.finish;
        }
        conveyorAction(id, el, sleepTime);
        if (nextLine != null)
            nextLine.AddElement(el);
    }
    else
    {
        return state.empty;
    }

    return state.ok;
}
```

Листинг 3.3: Добавление элемента в очередь

```
public void AddElement(Args arg)
{
    lock (q)
    {
        q.Enqueue(arg);
    }
}
```

Вывод

В данном разделе были рассмотрены основные сведения о средствах реализации и листинги кода реализованых алгоритмов.

4 Исследовательская часть

4.1 Сравнительный анализ на основе замеров времени

Был проведен замер времени работы параллельной и линейной обработки данных при разных временах обработки одной линии.

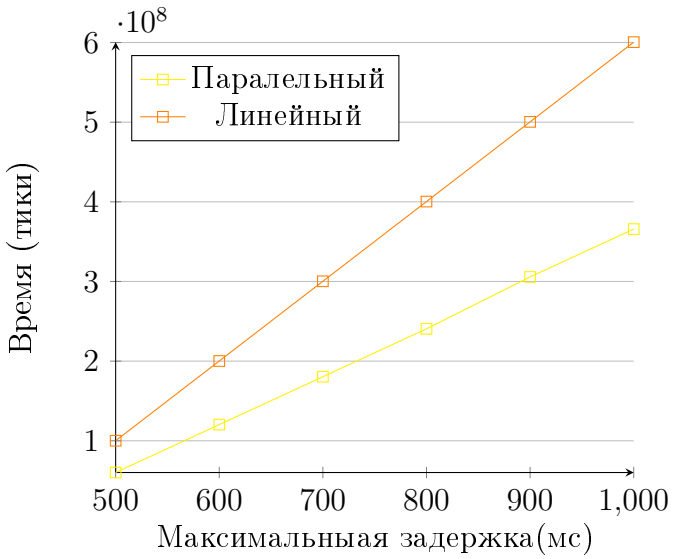

Рис. 4.1: Сравнение времени работы при разных значениях зедержек

На графиках видно, что конвейерная обработка с параллельными потоками в 2.5 раза быстрее, чем такая же линейная с захватом переменных.

4.2 Тестирование

Для тестирования был выведен лог операций для двух видов конвеерных вычислений в таблицах 4.1 и 4.2.

Таблица 4.1: Лог работы линейной конвеерной обработки

Номер конвеера	Номер элемента	Начало работы (в тиках)	Конец работы (в тиках)
0	0	637754283525565459	637754283555617285
0	1	637754283555617285	637754283585788373
0	2	637754283585788373	637754283615857819
0	3	637754283615857819	637754283646043819
0	4	637754283646043819	637754283676201866
1	0	637754283676201866	637754283696290895
1	1	637754283696290895	637754283716375006
1	2	637754283716375006	637754283736457699
1	3	637754283736457699	637754283756592474
1	4	637754283756592474	637754283776628489
2	0	637754283776628489	637754283796729358
2	1	637754283796729358	637754283816779488
2	2	637754283816779488	637754283836888273
2	3	637754283836888273	637754283856999532
2	4	637754283856999532	637754283877155587

Таблица 4.2: Лог работы параллельной конвеерной обработки

Номер конвеера	Номер элемента	Начало работы (в тиках)	Конец работы (в тиках)
0	0	637754283333576054	637754283363886189
0	1	637754283363886189	637754283393924988
0	2	637754283393924988	637754283423952316
0	3	637754283423952316	637754283454098617
0	4	637754283454098617	637754283484239394
1	0	637754283364352160	637754283384372251
1	1	637754283394399091	637754283414453891
1	2	637754283424735013	637754283444773742
1	3	637754283454882620	637754283475025046
1	4	637754283485184549	637754283505264381
2	0	637754283384681880	637754283404791332
2	1	637754283415082794	637754283435133032
2	2	637754283445230830	637754283465256404
2	3	637754283475337091	637754283495382878
2	4	637754283505423195	637754283525532144

Вывод

По результатам исследования конвейерную обработку нет смысла применять для задач, занимающих мало времени, т.к. в этом случае большая часть времени потратится на ожида-

ние доступа к переменной, дополнительных проверок. Тестирование показало, что конвейерная обработка реализована правильно.

Заключение

В ходе лабораторной работы были изучены возможности применения параллельных вычислений и конвейерной обработки и использован такой подход на практике.

Был проведен эксперимент с разными значениями задержек, который показал, что если первый конвейер тормозит работу, то общее время работы системы линейно зависит от задержки первого конвейера. Данная зависимость распространяется не только на первый конвеер, т. е. конвеерные вычисления тормозятся из-за самого медленного конвеера. Также этот эксперимент показал, что конвейерную обработку нет смысла применять для задач, занимающих мало времени, т. к. в этом случае большая часть времени потратится на ожидание доступа к переменной, дополнительных проверок.

В рамках данной работы были решены следующие **задачи**:

- изучена конвейерная обработка данных;
- реализована система конвейерных вычислений с количеством линий не меньше трёх;
- были сравнены параллельная и линейная реализация конвейерных вычислений;
- сделаны выводы на основе проделанной работы.

Поставленная цель была достигнута.

Список литературы

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

[2] Конвейерное производство[Электронный ресурс] - режим доступа https:/ё/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] И. В. Белоусов(2006), Матрицы и определители, учебное пособие по линейной алгебре, с. 1 - 16

[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] И. В. Белоусов(2006), Матрицы и определители, учебное пособие по линейной алгебре, с. 1 - 16

[7] Le Gall, F. (2012), "Faster algorithms for rectangular matrix multiplication Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012), pp. 514–523

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

[9] Руководство по языку C#[Электронный ресурс], - режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/