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»
ДИСЦИПЛИНА «Анализ алгоритмов»

Лабораторная работа № 5

Дисциплина Анализ алгоритмов
Тема Конвейер
Студент Боренко А. Д.
Группа ИУ7-52Б
Оценка (баллы) _____
Преподаватель Волкова Л.Л.

Москва — 2022г.

Содержание

Введение

Конвейер - это машина непрерывного действия, служащая для перемещения сыпучих, кусковых, штучных и др. грузов.

В программировании конвейерная обработка - это один из способов совмещения операций, при котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции. Первый способ - параллельные вычисления - уже был приведен раннее. В этой работе будет рассмотрен конвеер.

Целью данной работы является разработка и исследование конвейерных вычислений. Задачами данной лабораторной являются:

1. реализация последовательного алгоритма нахождения среднего арифметического матрицы;;
2. реализация конвейерной обработки на примере шифрования строк;
3. сравнительный анализ шифрования строк конвеерной обработкой и последовательным алгоритмом;
4. описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчетно-пояснительная записка к работе.

1. Аналитическая часть

В данном разделе будет рассмотрено понятие конвеерной обработки. Так же будут описаны алгоритмы применяемые в работе для шифрования строк.

1.1 Конвейерная обработка

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 исполь зовать 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.

1.2 Алгоритмы шифрования строк

Идея шифрования строк - преобразование их для скрытия от посторонних. В то же время, те которым предназначается информация способны ее дешифровать. В данной работе будут рассмотрены следующие алгоритмы: шифр Цезаря, шифр XOR, шифр Атбаш.

Шифр Цезаря

Имеется ключ от 0 до 26 (для латинского алфавита), каждая буква смещается на значение ключа.

Шифр Xor

Строка разбивается на отдельные символы и каждый символ представляется в бинарном виде. Далее с каждым символом применяется операция XOR с ключом. Результатом является зашифрованная строка.

Шифр Атбаш

Каждому i-ому символу алфавита ставится в соответствие символ алфавита (n-i), где n - размер алфавита.

1.3 Вывод аналитической части

В данной работе стоит задача реализации конвейерных вычислений. Были рассмотренны особенности построения конвейерных вычислений.

2. Конструкторская часть

В данном разделе представлены схемы алгоритмов. Так же будут описаны пользовательские структуры данных, приведены классы эквивалентности для тестирования реализуемого ПО.

2.1 Схемы

На рисунке 2.1 показана траектория заявки.

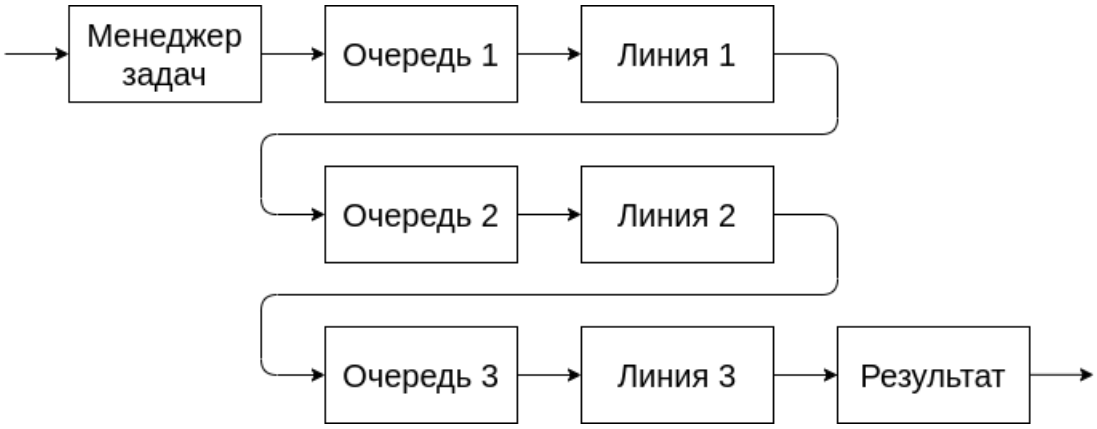

Рис. 2.1: Траектория заявки

На рисунке 2.2 показан принцип работы конвейера. После того, как первая заявка будет обработана на первом этапе, она поступит на второй, а на первый этап придет 2 заявка, таким образом две линии будут работать параллельно. Когда первая заявка будет обработана на 2 этапе, она попадет в 3 линию конвеера. Как толко первая линия обработает 2 заявку, она получит третью заявку.

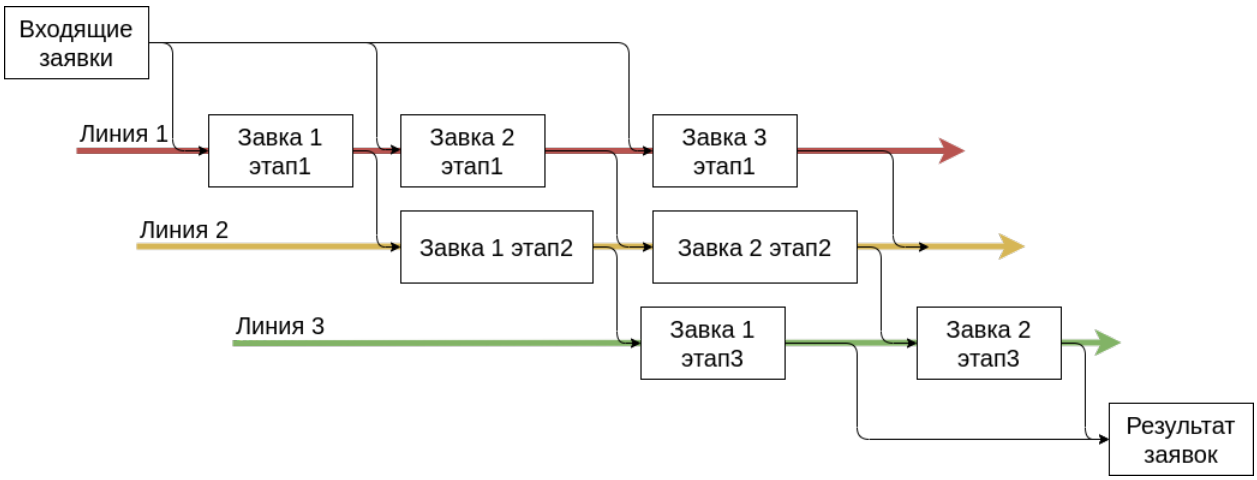

Рис. 2.2: Принцип работы конвеера

2.2 Структуры данных

При реализации приведенных алгоритмов потребуется тип данных: заявка. Она должна содержать следующие поля:

- строка, поступившая на вход,
- строка - результ,
- время поступления в конвейер,
- массив времени этапов (время этапов содержит время начала обработки и конца).

2.3 Тестирование

Для алгоритма шифрования строк можно выделить следующие классы эквивалентности:

Шифр Цезаря

1. символы, где i+key<n,
2. символы, где i+key>n.

Шифр Атбаш

1. символы, где i<n/2,
2. символы, где i>n/2.

Шифр XOR

1. символы, где i xor key < n,
2. символы, где i xor key > n.

2.4 Вывод конструкторской части

На основе данных, полученных в аналитическом разделе, были построены схемы конвеерной обработки, выделены необходимые для реализации структуры данных и методы тестирования.

3. Технологическая часть

3.1 Требования к ПО

Требования к программному обеспечению:

1. на вход подается количество заявок;

2. результат: замеры времени пребывания заявки на этапах и в очередях.

3.2 Выбор языка программирования

Был выбран язык go, поскольку он удовлетворяет необходимым требованиям и удобен программирования параллельных вычислений. Средой разработки выбрана Visual Studio Code.

3.3 Структуры данных

На листинге 3.1 представлено описание структуры заявки.

Листинг 3.1: Структура матрицы

```
type processing_time struct {
  start time.Time
  end time.Time
}
type request_t struct {
  id int
  str string
  start_wait time.Time
  steps[STEPS_COUNT] processing_time
```

```
}
```

3.4 Реализация конвейера

На листинге 3.2 представлена реализация конвейера.

Листинг 3.2: Реализация последовательного алгоритма нахождения среднего арифметического матрицы

```
type procFunc func(string) string

func proc(input <-chan request_t, procID int, f procFunc) <-chan request_t {
  output := make(chan request_t)

  go func() {
    defer close(output)
    for arg := range input {
      arg.steps[procID].start = time.Now()

      //time.Sleep(time.Duration(100) * time.Microsecond)
      //fmt.Println(arg.id, procID, time.Now())
      arg.str = f(arg.str)
      arg.steps[procID].end = time.Now()
      output <- arg //value

          time.Sleep(10*time.Microsecond)
    }
  }()

  return output
}

func setup(n int) <-chan request_t {
  channel := make(chan request_t)

  var req []request_t = make([]request_t, n)
```

```
28
29    for i:=0;i<n;i++{
30      req[i] = make_request(i)
31    }
32
33    go func(req []request_t) {
34      defer close(channel)
35      for i := 0; i < n; i++ {
36        //req := make_request(i)
37        req[i].start_wait = time.Now()
38        channel <- req[i]
39      }
40    }(req)
41    return channel
42  }
43
44
45  func conveyorRun(count int, channel <- chan request_t) request_array_t {
46      //var mutex sync.Mutex
47
48    var steps[STEPS_COUNT]func(string)string
49    steps[0] = CESAR_CODE
50    steps[1] = XOR_CODE
51    steps[2] = ATBASH_CODE
52
53
54    for i:=0;i<STEPS_COUNT;i++{
55      channel = proc(channel, i, steps[i])
56    }
57
58
59    var res_arr_request request_array_t = make_empty_request_array(count)
60
61    var i int = 0
62    for res := range channel {
63      res_arr_request.elems[i] = res
64      i++
```

```
65    }
66
67    return res_arr_request
68 }
```

3.5 Тестирование

Модульные тесты

Таблица 3.1 – Тесты

№	Ввод	Шифр Цезаря (key=13)	Шифр Атбаш	Шифр XOR (key a)
1	abc	nop	zyx	yz{
2	zyx	mlk	abc	bap

Тесты пройдены.

3.6 Вывод технологической части

Были реализованы исследуемые алгоритмы, программа прошла тесты и удовлетворяет требованиям.

4. Экспериментальная часть

Оценка качества работы алгоритмов. Экспериментальное сравнение работы различных алгоритмов нахождения среднего арифметического матрицы (зависимость времени выполнения от размерности матриц).

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

1. процессор: Intel® Core™ i3-7100U CPU @ 2.40GHz × 4;
2. память: 11,6 GiB;
3. операционная система: Ubuntu 20.04.1 LTS.

4.2 Примеры работы

На рисунке 4.1 показан пример работы.

```
Hey! Now we will work with conveer.
How many request we will test?
1000
So, we will test conveer on 1000 requests
START
END
Data:
Время в nsec
 Очередь|     min|     max|      av
       1|     488|  202226|    7624
       2|     378|  195068|    3030
    Этап|     min|     max|      av
       1|    3811|  304944|    9256
       2|    3255|  372055|    7470
       3|    3096|  400510|    6418

 Система|     min|     max|      av
        |   13854|  557887|   33800
Show each request log?(1 -yes, other -no)
2
On  10000  request we have results:
Conveer time =  887.510246ms
Lenear  time =  1.176187041s
Draw compare graph?(1 -yes, other -no)
2
```

Рис. 4.1: Пример работы 1

4.3 Замеры времени

Таблица 4.1 содержит лог первых 16 заявок.

Таблица 4.1 – Лог заявок (нсек)

id	go conv	1 step st	2 step en	1 step st	2 step en	1 step st	2 step en
0	0	26079	664732	742436	758880	772739	788061
1	121906	754064	769764	874328	885545	910297	921014
2	771362	909081	919927	1000013	1012542	1138180	1147318
3	920734	1054765	1064300	1224065	1245139	1250378	1254853
4	1064843	1365380	1373496	1383268	1392493	1393559	1397981
5	1382654	1450067	1464613	1474740	1485290	1487019	1496701
6	1498192	1499075	1506823	1540792	1548461	1556604	1564338
7	1498602	1617884	1627640	1630647	1642891	1643533	1650580
8	1651564	1706643	1717353	1722394	1730959	1731787	1736708
9	1737951	1796989	1805167	1809310	1813951	1814601	1819223
10	1820352	1883808	1894320	1903558	1922516	1923622	1930865
11	1902944	1968995	1974614	1978125	1990048	1990820	1995240
12	1978034	2051845	2061071	2065122	2069918	2070621	2074851
13	2076065	2137928	2153802	2163676	2170724	2171611	2178528
14	2162848	2228934	2240335	2244886	2249660	2250362	2254679
15	2255685	2438950	2448361	2456493	2465216	2466067	2471479

Таблица 4.2 содержит минимальное, максимальное и среднее время пребывания заявки в очередях, в этапах и в конвейере.

Таблица 4.2 – Статистика времени пребывания заявок (нсек)

Очередь	min	max	av
1	488	202226	7624
2	378	195068	3030
Этап	min	max	av
1	3811	304944	9256
2	3255	372055	7470
3	3096	400510	6418
Система	min	max	av
	13854	557887	33800

Таблица 4.3 содержит время работы конвейрной обработки и стандартного алго-

ритма.

Таблица 4.3 – Время работы конвейрной обработки и стандартного алгоритма (нсек)

Алгоритм	Время
Conveer	911.49707ms
Lenear	1.184577591s

На рисунках 4.2 и 4.3 показаны графические результаты сравнения исследуемых алгоритмов по времени.

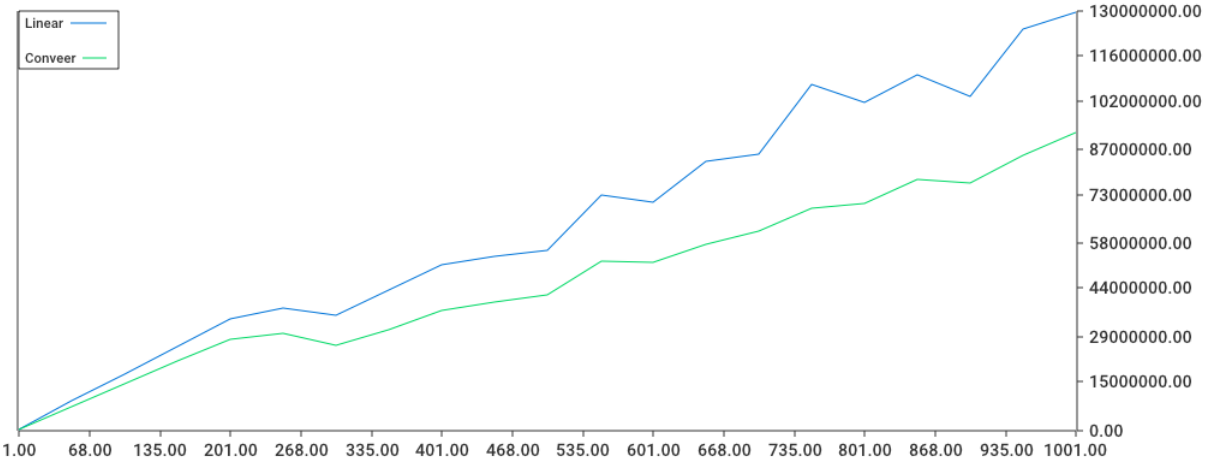

Рис. 4.2: Сравнение конвейерной обработки и последовательного алгоритма

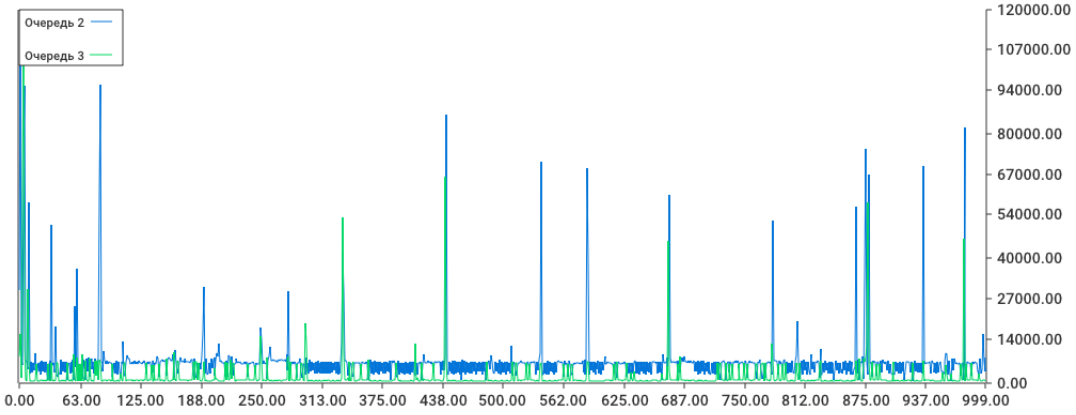

Рис. 4.3: Время пребывания заявок во второй и третьей очередях

4.4 Сравнительный анализ

По результатам экспериментов можно заключить следующее:

- конвейерная обработка быстрее обычной;
- из времени пребывания заявок в 2 и 3 очередях следует, что 3 этап быстрее 2;
- первый этап самый медленный из всех;
- третий этап самый быстрый.

4.5 Вывод экспериментальной части

В данном разделе были рассмотрены результаты работы программы. Из анализа стало ясно, что первый этап - шифрование с помощью шифра Цезаря замедляет работу всей системы. Самым быстрым является шифрование xor.

Заключение

В ходе лабораторной работы достигнута поставленная цель: разработ- ка и исследование конвейерных вычислений и использование их на практике. Также решены все поставленные задачи. Получены практические навыки реализации конвейера. Проведёно исследование конвейерных вычислений и использование их на практике. Экспериментально подтверждены различия в эффективности алгоритмов с указанием лучших и худших случаев. Цель работы достигнута, решены поставленные задачи.

Список литературы

[1] Gratzer George A. More Math Into LaTeX. 4th изд. Boston: Birkhauser, 2007.

[2] Debian – универсальная операционная система [Электронный ресурс]. Режим доступа: https://www.debian.org/. Дата обращения: 20.12.2021.

[3] Linux – Getting Started [Электронный ресурс]. Режим доступа: https://linux.org. Дата обращения: 20.12.2021.

[4] Макконнел. Дж. Анализ алгоритмов. Активный обучающий подход. – М. Режим доступа:https://www.linux.org.ru/. 2017. 267 с.

[5] Go Documentation [Электронный ресурс]. Режим доступа: https://go.dev/doc/. Дата обращения: 20.09.2020.