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 Конвейерная обработка данных

Студент Андрич К.

Группа ИУ7И-56Б

Оценка (баллы) _____

Преподаватели Волкова Л.Л.

Москва — 2021 г.

Оглавление

Введение

Имеется большое количество важнейших задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем. Постоянно появляются новые задачи подобного рода и возрастают требования к точности и к скорости решения прежних задач; поэтому вопросы разработки и использования сверхмощных компьютеров (называемых суперкомпьютерами) актуальны сейчас и в будущем. Но пока эти трудности пока что не удается преодолеть. Из-за этого приходится и эти по пути создания параллельных вычислительных систем, т.е. систем, в которых предусмотрена одновременная реализация ряда вычислительных процессов, связанных с решением одной задачи. На современном этапе развития вычислительной техники такой способ, по-видимому, является одним из основных способов ускорения вычислений.

Цель данной лабораторной работы: получить навык организации асинхронной передачи данных между потоками на примере конвейерной обработки информации.

Для достижения поставленной цели требуется выполнить следующие задачи.

1. Выбрать и описать методы обработки данных, которые будут сопоставлены методам конвейера.
2. Описать архитектуру программы, а именно какие функции имеет главный поток, принципы и алгоритмы обмена данными между потоками.
3. Реализовать конвейерную систему, а также сформировать лог событий с указанием времени их происхождения, описать реализацию.
4. Провести тестирование системы.
5. Интерпретировать сформированный лог.

Одной из важнейших идей при создании многопроцессорных систем и при эффективной реализации алгоритмов на этих системах является идея конвейерных вычислений.

1 Аналитическая часть

В данном разделе будут описаны задачи и идея конвейеризации.

1.1 Описание задачи

Конвейеризация – это техника, в результате которой задача или команда разбивается на некоторое число подзадач, которые выполняются последовательно. Каждая подкоманда выполняется на своем логическом устройстве. Все логические устройства (ступени) соединяются последовательно таким образом, что выход i-ой ступени связан с входом (i+1)-ой ступени, все ступени работают одновременно. Множество ступеней называется конвейером. Выигрыш во времени достигается при выполнении нескольких задач за счет параллельной работы ступеней, вовлекая на каждом такте новую задачу или команду.

В конвейере различают r последовательных этапов, так что когда i-я операция проходит s-й этап, то $(i + k)$-я операция проходит $(s - k)$-й этап. На рисунке 1.1 изображена работа конвейера.

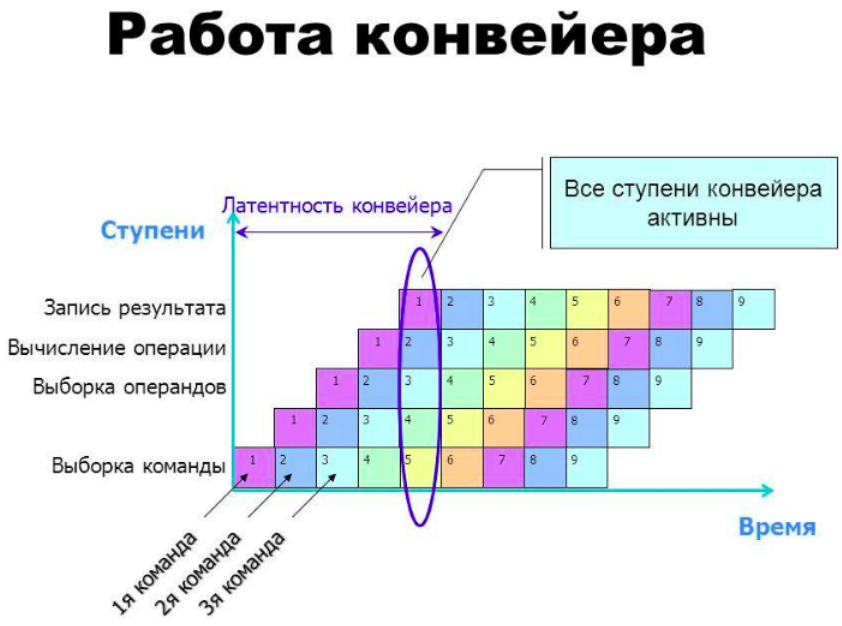

Рис. 1.1: Работа конвейера.

В данной лабораторной работе реализована некая функцию кодирования строки, которая состоит из трех последовательных действий: применения первой функции шифра Цезаря,

применения функции которая меняет регистр буквы и применения функции которая меняет элемент местами с элементом, стоящим на n / 2 от него, где n - размер строки. Если необходимо закодировать какой-то массив строк, то можно использовать конвейерную обработку данных. Таким образом задача будет решена эффективнее, чем при последовательном применении алгоритмов к массиву значений.

Конвейер будет состоять из четырех уровней. Обработанные данные передаются последовательно с одного уровня (одной ленты) конвейера на следующий (следующую ленту). Далее на каждом уровне осуществляется обработка данных, занимающая определенное время. Для каждой ленты создается своя очередь задач, в которой хранятся все необработанные строки. На последнем уровне конвейера обработанные объекты попадают в пул обработанных задач.

Уровни конвейера:

- 0 уровень — генерация входных данных в первую очередь;
- 1 уровень(лента) — применение шифра Цезаря к строкам из первой очереди, запись результата во 2 очередь;
- 2 уровень(лента) — применение функции, меняющей регистры символов к строкам 2 очереди, запись результата в 3 очередь;
- 3 уровень(лента) — применение функции, меняющей местами символы в строке по определённом выше закону к строкам 3 очереди, запись результата в пул обработанных задач.

Поскольку запись в очередь и извлечение из очереди это не атомарные операции, необходимо создать их таковыми путем использования мьютексов (по одному на одну очередь) и критических секций, чтобы избежать ошибок в ситуации гонок.

1.2 Вывод

В данном разделе были заданы задачи и описана идея конвейеризации.

2 | Конструкторская часть

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 создан. Также в главном потоке генерируется массив входных данных и заполняется первая очередь (уровень 0). 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.

На рисунке 2.1 изображена схема работы конвейерной обработки.

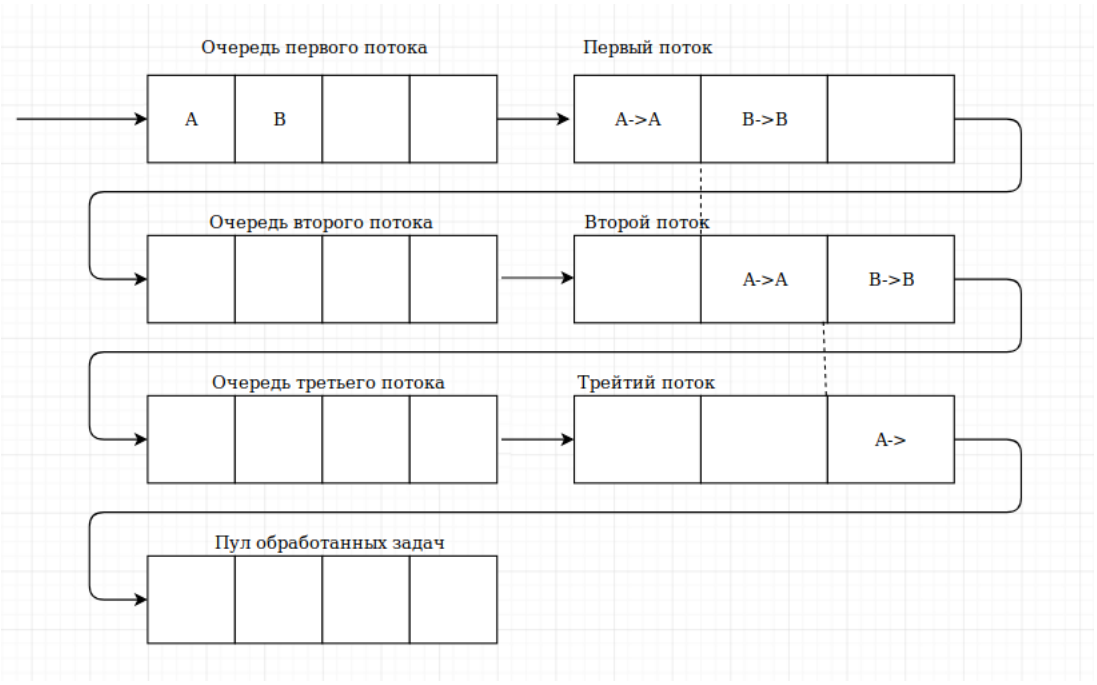

Рис. 2.1: Схема работы конвейерной обработки.

2.2 Схемы алгоритмов работы конвейерной обработки

На рисунках 2.2 и 2.3 показаны схем главного и рабочего процессов.

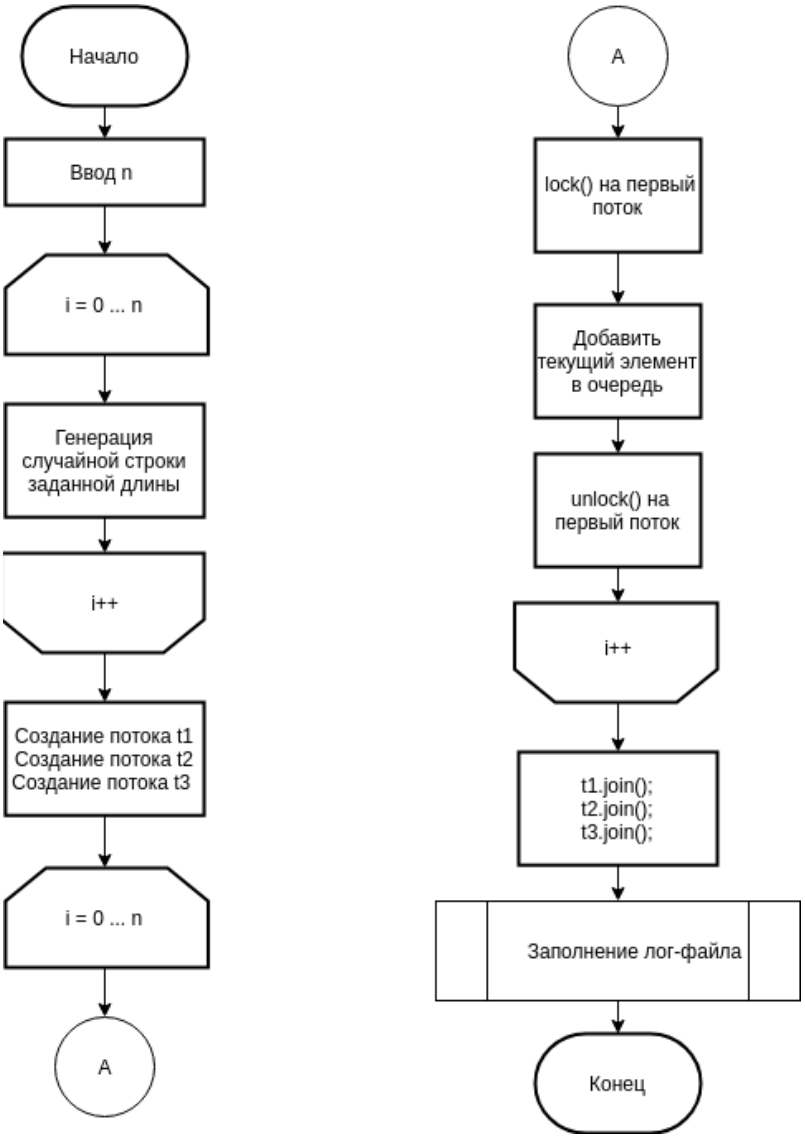

Рис. 2.2: Схема работы главного процесса

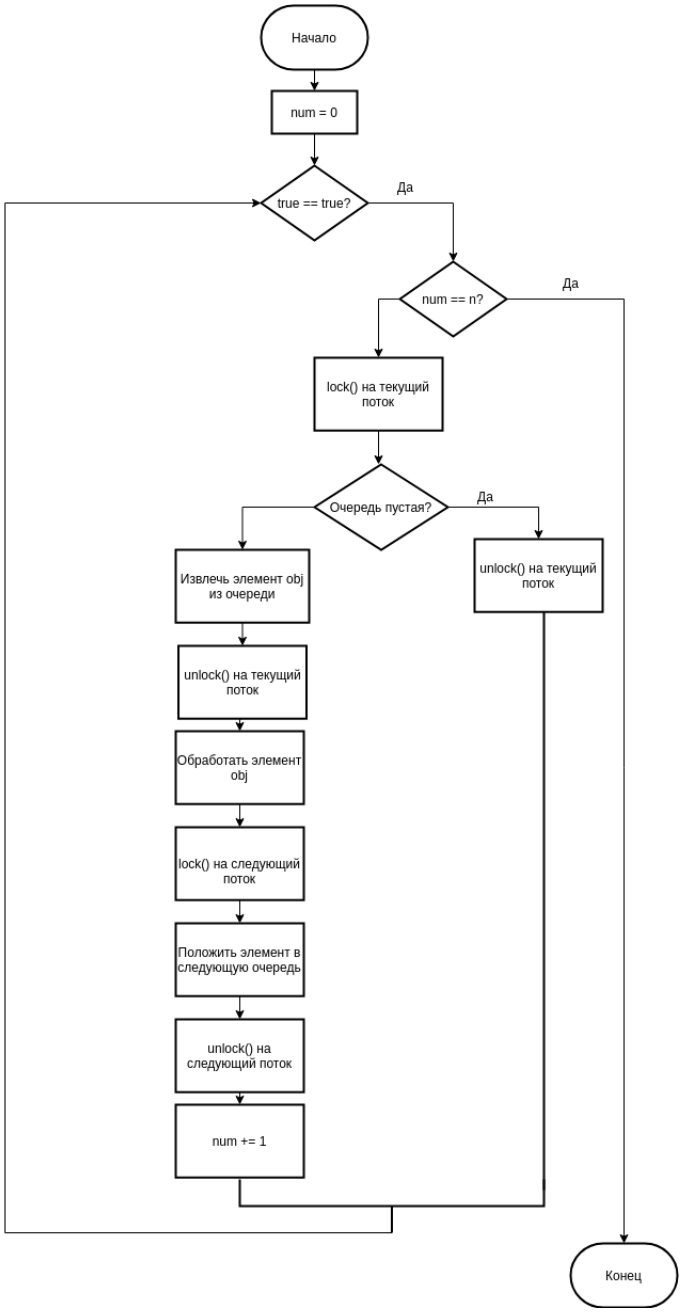

Рис. 2.3: Схема работы рабочего процесса

2.3 Вывод

В данном разделе показаны схемы рабочего и главного процессов.

3 | Технологическая часть

В этом разделе будет обоснован выбор языка програмирования и приведены листинги кода реализованных алгоритмов.

3.1 Выбор ЯП

В качестве языка программирования был выбран C++ так как этот язык удобен для работы с потоками. Для замера времени выполнения использовалась функция *clock*() из библиотеки ctime. Эта функция возвращает количество временных тактов, прошедших с начала запуска программы.

3.2 Реализация алгоритмов

В листингах 3.1, 3.2, 3.3 представлен код рабочих потоков, в листинге 3.4 представлен код главного потока.

Листинг 3.1: Реализация первого уровня конвейера.

```
1  void first_conv() {
2    int num = 0;
3    while (true) {
4      if (num == n)
5        break;
6      m1.lock();
7      if (queue1.empty()) {
8        m1.unlock();
9        continue;
10     }
11     string cur_str = queue1.front().str;
12     int cur_task_num = queue1.front().task_num;
13     timer.add_time(1, clock() - queue1.front().time, queue1.front().
           task_num);
14     queue1.pop();
15
16     clock_t cur_time = clock();
17     m1.unlock();
18     string new_str = caesar(cur_str);
19     m2.lock();
20     timer.add_time(0, clock() - cur_time, cur_task_num);
21
22     queue2.push(Object(new_str, cur_task_num,clock()));
23     m2.unlock();
24     num++;
25   }
26 }
```

Листинг 3.2: Реализация второго уровня конвейера.

```
1  void second_conv() {
2    int num = 0;
3    while (true) {
4      if (num == n)
5        break;
6      m2.lock(); // wait in queue
7      if (queue2.empty()) {
8        m2.unlock();
9        continue;
10     }
11     string cur_str = queue2.front().str;
12     int cur_task_num = queue2.front().task_num;
13     timer.add_time(1, clock() - queue2.front().time, queue2.front().
           task_num);
14     queue2.pop();
15
16     clock_t cur_time = clock();
17     m2.unlock();
18     string new_str = upper_lower(cur_str);
19     m3.lock();
20     timer.add_time(0, clock() - cur_time, cur_task_num);
21
22     queue3.push(Object(new_str, cur_task_num, clock()));
23     m3.unlock();
24     num++;
25   }
26 }
```

Листинг 3.3: Реализация третьего уровня конвейера.

```
1  void third_conv() {
2    int num = 0;
3    while (true) {
4      if (num == n)
5        break;
6      m3.lock(); // wait in queue
7      if (queue3.empty()) {
8        m3.unlock();
9        continue;
10     }
11     string cur_str = queue3.front().str;
12     int cur_task_num = queue3.front().task_num;
13     timer.add_time(1, clock() - queue3.front().time, queue3.front().
           task_num);
14     queue3.pop();
15
16     clock_t cur_time = clock();
17     m3.unlock();
18     string new_str = reverse(cur_str);
19     resm.lock();
20     timer.add_time(0, clock() - cur_time, cur_task_num);
21
22     res.push_back(new_str);
23     resm.unlock();
24     num++;
25   }
26 }
```

Листинг 3.4: Основная функция программы.

```
int main() {
  srand(time(nullptr));
  cout << "Enter the number of lines: ";
  cin >> n;
  if (n <= 0) {
    cout << "Incorrect number of lines";
    return -1;
  }

  objvec.resize(n);
  timer.set_size();

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

  start_t = clock();

  thread t1(first_conv);
  thread t2(second_conv);
  thread t3(third_conv);

  for (int i = 0; i < n; i++) {
    m1.lock();
    queue1.push(Object(objvec[i], i, clock()));
    m1.unlock();
  }

  t1.join();
  t2.join();
  t3.join();

  create_log(clock() - start_t);
  return 0;
}
```

3.3 Тестирование программы

На вход подаем 6 строк длиной 50000. Программа выдала файл с логом.

3.4 Вывод

В этом разделе обоснован выбор языка програмирования и приведены листинги кода реализованных алгоритмов. Программа прошла тестирование и работает правильно.

4 Исследовательская часть

В данном разделе будут приведены демонстрация работы программы и исследование процессорного времени реализаций алгоритмов.

4.1 Пример работы

Демонстрация работы программы приведена на рисунке 4.1. На вход подаётся количество строк для обработки.

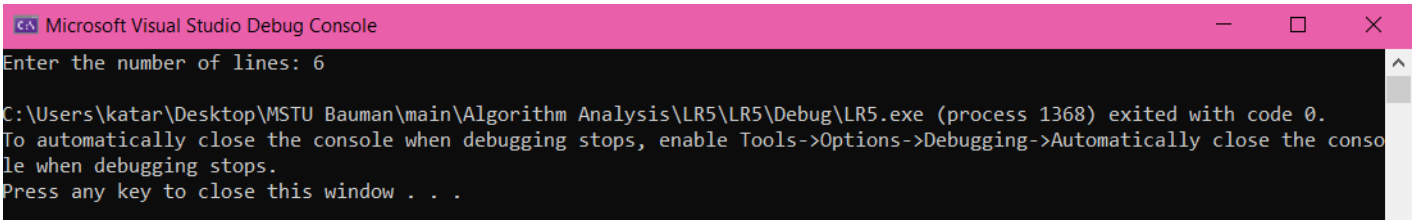

Рис. 4.1: Демонстрация работы программы

В результате работы программы заполняется лог-файл. Содержимое лог-файла приведено на рисунке 4.2. Время замерено с тактах процессора.

Рис. 4.2: Содержимое лог-файла

По лог-файлу можно проследить выполнение каждого этапа задач. Также можно проследить, что обработка выполняются параллельно. Например, второй коневейер не ждёт полного завершения работы первого конвейера прежде, чем начать работу, а начинает выполнение как только в очередь поступает первый элемент.

Если бы использовалась линейная реализация, то проделанная работа заняла бы 36656 тактов процессора, что в 1,5 раза больше, чем время, потраченное при конвейерной реализации.

4.2 Вывод

Конвейерная обработка данных - полезный инструмент, который уменьшает время выполнения программы за счёт параллельной обработки данных. Самым эффективным временем считается время, когда все линии конвейера работают параллельно, обрабатывая свои задачи. Этот метод даёт выйгрыш по времени в том случае, когда выполняемые задачи намного больше по времени, чем время, затрачиваемое на реализацию конвейера (работу с потоками, перекладывание из очереди в очередь и тд).

Заключение

В ходе лабораторной работы была достигнута цель. Был получен навык организации асинхронной передачи данных между потоками на примере конвейерной обработки информации.

Для достижения поставленной цели были выполнены следующие задачи.

1. Выбраны и описаны методы обработки данных, которые будут сопоставлены методам конвейера.

2. Описана архитектура программы, а именно какие функции имеет главный поток, принципы и алгоритмы обмена данными между потоками.

3. Реализована конвейерная система, а также сформирован лог событий с указанием времени их происхождения, описана реализация.

4. Проведено тестирование системы.

5. Интерпретирован сформированный лог.

Эксперементальным путём выявлено, что при большой нагрузке конвейер работает примерно в 1,5 раза быстрее чем линейная реализация, что является показателем эффективности работы конвейера по времени.

Литература

[1] Воеводин В.В. Математические модели и методы в параллельных процессах. М., 1986. 296с.

[2] Корнеев В.В. Параллельные вычислительные системы. М., 1999. 320 с.

[3] Конвейерные вычисления [Электронный ресурс]. Режим доступа: http://www.myshared.ru/slide/674082 Дата обращения: 31.10.2021

[4] Функция clock().[Электронный ресурс]. Режим доступа: http://cppstudio.com/post/561/ Дата обращения 01.11.2021.