Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный технический университет
имени Н. Э. Баумана» (национальный исследовательский университет)

Дисциплина: «Анализ алгоритмов»

Отчет по лабораторной работе №5

Тема работы:
«Конвейерная обработка данных»

Студент: Васюков А. В.
Группа: ИУ7-52Б
Преподаватели: Волкова Л. Л.,
Строганов Ю. В.

Москва, 2019 г.

Содержание

Введение

Имеется большое количество важных задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем.

Параллелизм может значительно ускорить решение таких задач. Особое место среди систем подобного рода занимает конвейерный обработчик. Он прост для понимания, ведь принцип его работы основан на реальном механизме непрервного действия. [3]

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

Задачи работы:

1) изучить алгоритм конвейеризации;
2) реализовать конвейерную систему;
3) провести журналирование событий в конвейерной обработке;
4) описать и обосновать полученные результаты в отчете о лабораторной работе, выполненного как расчётно-пояснительная записка.

1 Аналитический раздел

В данном разделе будет рассмотрен алгоритм конвейеризации.

1.1 Описание алгоритмов

1.1.1 Алгоритм конвейерной обработки данных

Конвейеризация — это техника, в результате которой задача разбивается на некоторое число подзадач, которые выполняются последовательно. Конвейерный подход позволяет создавать эффективные параллельные реализации обычных неспециализированных алгоритмов. [4]

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. [5]

Каждое звено конвейера выполняет следующие действия:

- получить данные,
- обработать данные,
- послать данные следующим звеньям.

Выводы

Рассмотрен алгоритм конвейерной обработки данных, выделены его ключевые моменты.

2 Конструкторский раздел

В разделе приводятся схемы выбранных алгоритмов сортировки, производится их теоретический сравнительный анализ.

2.1 Разработка алгоритмов

На рис. 1 приведена схема общая схема конвейерной обработки.

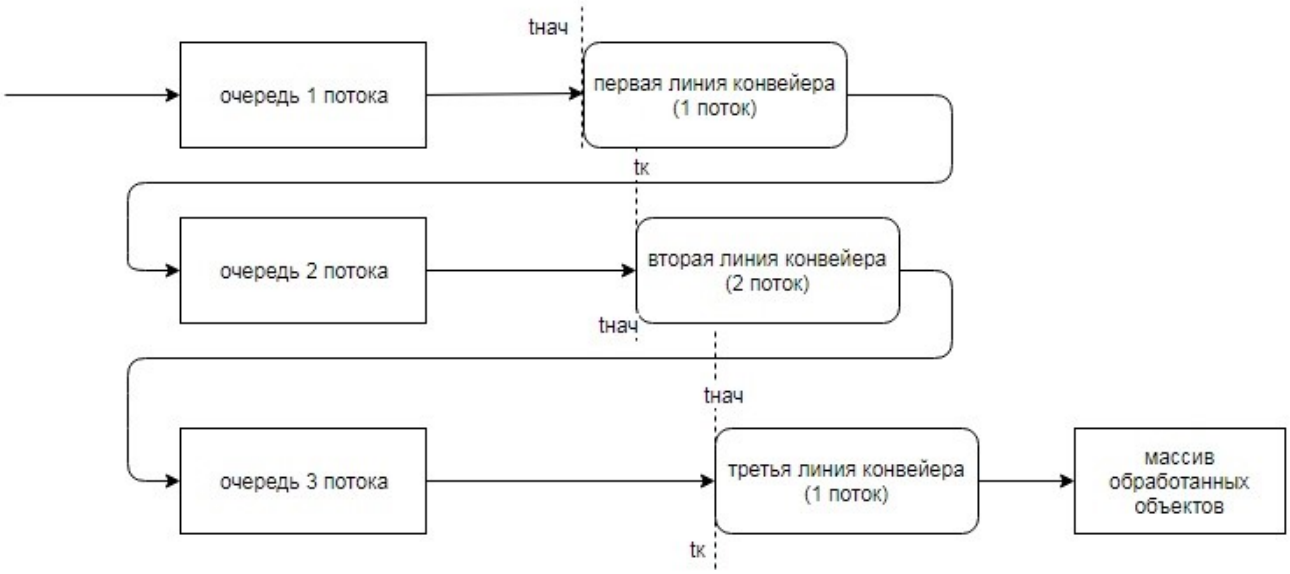

Рис. 1: Схема работы конвейера

Псевдокод главной функции:

```
1. Создать 3 потока, мьютекса и очереди
2. Заполнить массив объектов значениями от 1 до n
3. Добавить элементы массива в очередь 1
4. Применить join ко всем потокам
5. Записать информации из массива результатов в лог-файл
```

Псевдокод для ленты 1:

```
1. Бесконечный цикл
2.        Заблокировать мьютекс 1
3.        Если очередь 1 не пуста
4.              Извлечь элемент из очереди
5.              Записать информацию о начале работы
6.              Добавить элемент в результирующий массив
```

```
7.              Выполнить функции 1
8.              Добавить элемент в очередь 2
8.              Записать информацию об окончании работы
10.              Добавить элемент в результирующий массив
11.              Если элемент равен n
12.                    Завершить работу
13.        Разблокировать мьютекс 1
```

Псевдокод для ленты 2:

```
1. Бесконечный цикл
2.        Заблокировать мьютекс 2
3.        Если очередь 2 не пуста
4.              Извлечь элемент из очереди
5.              Записать информацию о начале работы
6.              Добавить элемент в результирующий массив
7.              Выполнить функции 2
8.              Добавить элемент в очередь 3
9.              Записать информацию об окончании работы
10.              Добавить элемент в результирующий массив
11.              Если элемент равен n
12.                    Завершить работу
13.        Разблокировать мьютекс 2
```

Псевдокод для ленты 3:

```
1. Бесконечный цикл
2.        Заблокировать мьютекс 3
3.        Если очередь 3 не пуста
4.              Извлечь элемент из очереди
5.              Записать информацию о начале работы
6.              Добавить элемент в результирующий массив
7.              Выполнить функции 3
8.              Записать информацию об окончании работы
9.              Добавить элемент в результирующий массив
10.              Если элемент равен n
11.                    Завершить работу
12.        Разблокировать мьютекс 3
```

Выводы

В разделе представлен псевдокод конвейерной обработки данных. Даны интерпретации главной функции и 3 функций, имитирующих работу лент.

3 Технологический раздел

Здесь описываются требования к программному обеспечению и средства реализации, приводятся листинги программы.

3.1 Требования к программному обеспечению

Запуск: conv.exe [n file s1 s2 s3]

Параметры командной строки:

- n – количество обрабатываемых элементов (по умолчанию 5),
- file – имя файла журналирования (по умолчанию *testconv.txt*),
- s1 – время работы функции для 1 ленты (по умолчанию 1000 мс),
- s2 – время работы функции для 2 ленты (по умолчанию 1000 мс),
- s3 – время работы функции для 3 ленты (по умолчанию 1000 мс).

Выходные данные: файл с записями о событиях в хронологическом порядке.

На рис. 2 приведена функциональная схема программы сортировки массива.

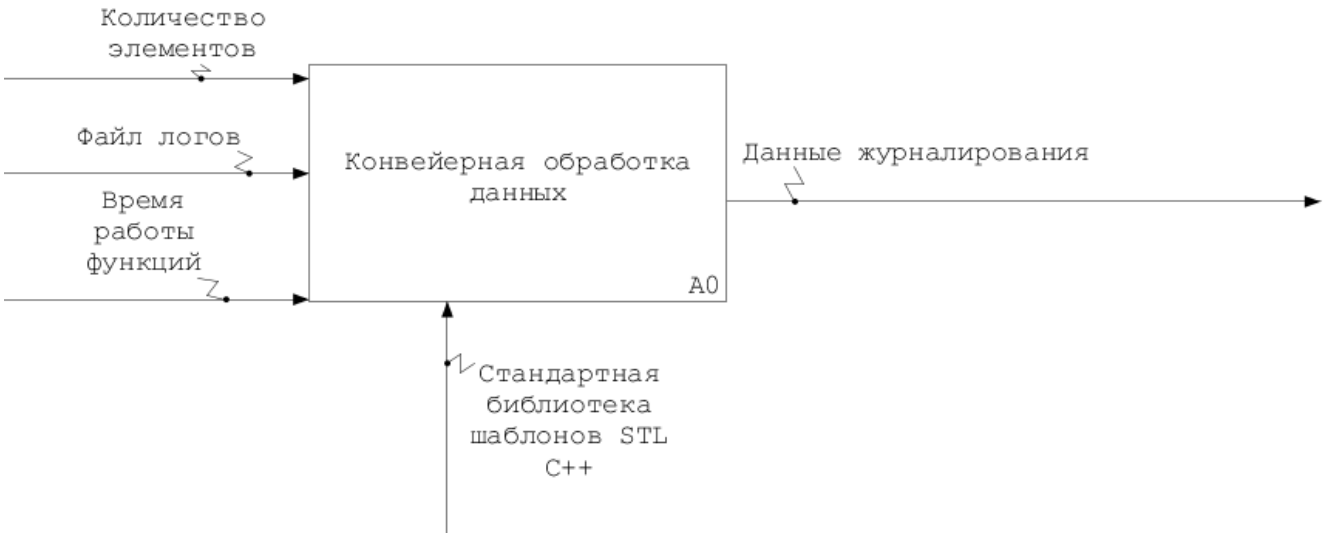

Рис. 2: Функциональная схема программы сортировки массива

3.2 Средства реализации

Программа написана на языке C++ (стандарт C++17), который предоставляет программисту мощные инструменты для реализации различных алгоритмов и является достаточно надежным и эффективным. Для написания использовался редактор исходного кода *Visual Studio Code*, для сборки проекта применена система автоматизации $CMake$ (флаги компиляции $-std = c++17 - Wall - Werror - pedantic$). [6]

Для хранения массива, строк применяются контейнерные классы *std::vector*, *std::string* из стандартной библиотеки шаблонов STL. Кроме того, для реализации конвейера используются *std::thread* и *std::mutex*.

Замер времени производится с помощью функции *GetLocalTime* из библиотеки *windows.h*.

3.3 Листинг программы

Реализованная программа представлена в листингах 1-5.

Листинг 1: Структура объекта

```
struct Obj {
    int index;
    int item;
    SYSTEMTIME t;
    char state;
};
```

Листинг 2: Главная функция

```
void conveyor() {
    std::vector<std::thread> thr(3);

    thr[0] = std::thread(belt_1);
    thr[1] = std::thread(belt_2);
    thr[2] = std::thread(belt_3);

    std::vector<Obj> pool(size_);
    for (auto i = 0; i < size_; ++i) {
        pool[i].item = i + 1;
    }

    for (auto i = 0; i < size_; ++i) {
        m1.lock();
        q1.push(pool[i]);
        m1.unlock();
        Sleep(50);
    }

    for (auto i = 0; i < int(thr.size()); ++i) {
        thr[i].join();
    }
```

```
23
24      get_res();
25      std::cout << "OK!" << std::endl;
26  }
```

Листинг 3: Лента 1

```
1   void belt_1() {
2       while (true) {
3           m1.lock();
4           if (!q1.empty()) {
5               auto obj = q1.front();
6               obj.index = 1;
7               GetLocalTime(&obj.t);
8               obj.state = 'S';
9
10              m.lock();
11              res.push_back(obj);
12              m.unlock();
13
14              q1.pop();
15              Sleep(sleep_1);
16
17              m2.lock();
18              q2.push(obj);
19              m2.unlock();
20
21              GetLocalTime(&obj.t);
22              obj.state = 'F';
23
24              m.lock();
25              res.push_back(obj);
26              m.unlock();
27
28              if (obj.item == size_) {
29                  return;
30              }
31          }
32          m1.unlock();
33      }
34  }
```

Листинг 4: Лента 2

```
1   void belt_2() {
2       while (true) {
3           m2.lock();
4           if (!q2.empty()) {
5               auto obj = q2.front();
6               obj.index = 2;
7               GetLocalTime(&obj.t);
8               obj.state = 'S';
9
10              m.lock();
11              res.push_back(obj);
12              m.unlock();
```

```
13
14              q2.pop();
15              Sleep(sleep_2);
16
17              GetLocalTime(&obj.t);
18              obj.state = 'F';
19
20              m3.lock();
21              q3.push(obj);
22              m3.unlock();
23
24              m.lock();
25              res.push_back(obj);
26              m.unlock();
27
28              if (obj.item == size_) {
29                  return;
30              }
31          }
32          m2.unlock();
33      }
34  }
```

Листинг 5: Лента 3

```
1  void belt_3() {
2      while (true) {
3          m3.lock();
4          if (!q3.empty()) {
5              auto obj = q3.front();
6              obj.index = 3;
7              GetLocalTime(&obj.t);
8              obj.state = 'S';
9
10             m.lock();
11             res.push_back(obj);
12             m.unlock();
13
14             q3.pop();
15             Sleep(sleep_3);
16
17             GetLocalTime(&obj.t);
18             obj.state = 'F';
19
20             m.lock();
21             res.push_back(obj);
22             m.unlock();
23
24             if (obj.item == size_) {
25                 return;
26             }
27         }
28         m3.unlock();
29     }
30 }
```

Выводы

В данном разделе были рассмотрены требования к программному обеспечению, обоснован выбор средств реализации, приведены листинги программы.

4 Исследовательский раздел

В разделе представлены примеры выполнения программы, а также пояснения к некоторым файлам журналирования.

4.1 Примеры работы и анализ файлов журналирования

На рис. 3-6 приведены примеры работы программы.

Обозначения:
№ operation – номер события,
Belt – номер ленты,
Item – значение элемента,
State – состояние обработки (S - начало, F - конец),
Time – время события в формате ч:м:с.мс.

№ operation	Belt	Item	State	Time
1	1	1	S	15:15:54.752
2	1	1	F	15:15:55.753
3	2	1	S	15:15:55.753
4	2	1	F	15:15:56.753
5	3	1	S	15:15:56.753
6	3	1	F	15:15:57.754

Рис. 3: Обработка 1 элемента ($s1 = s3 = s3 = 1000$)

№ operation	Belt	Item	State	Time
1	1	1	S	15:18:12.870
2	1	1	F	15:18:13.870
3	2	1	S	15:18:13.870
4	2	1	F	15:18:18.871
5	3	1	S	15:18:18.871
6	3	1	F	15:18:21.872

Рис. 4: Обработка 1 элемента ($s1 = 1000$, $s2 = 5000$, $s3 = 3000$)

На первых двух рисунках можно наблюдать поэтапную обработку для 1 элемента. Время простоя для каждой ленты соответствует ожидаемой величине.

№ operation	Belt	Item	State	Time
1	1	1	S	15:16:9.6
2	1	1	F	15:16:10.7
3	2	1	S	15:16:10.7
4	1	2	S	15:16:10.7
5	2	1	F	15:16:11.7
6	3	1	S	15:16:11.7
7	1	2	F	15:16:11.7
8	2	2	S	15:16:11.7
9	1	3	S	15:16:11.7
10	3	1	F	15:16:12.8
11	2	2	F	15:16:12.8
12	3	2	S	15:16:12.8
13	1	3	F	15:16:12.8
14	2	3	S	15:16:12.8
15	3	2	F	15:16:13.9
16	2	3	F	15:16:13.9
17	3	3	S	15:16:13.9
18	3	3	F	15:16:14.9

Рис. 5: Обработка 3 элементов ($s1 = s3 = s3 = 1000$)

№ operation	Belt	Item	State	Time
1	1	1	S	15:18:31.121
2	1	1	F	15:18:32.122
3	2	1	S	15:18:32.122
4	1	2	S	15:18:32.122
5	2	1	F	15:18:37.122
6	3	1	S	15:18:37.122
7	1	2	F	15:18:37.122
8	2	2	S	15:18:37.122
9	1	3	S	15:18:37.122
10	3	1	F	15:18:40.123
11	2	2	F	15:18:42.122
12	3	2	S	15:18:42.122
13	1	3	F	15:18:42.122
14	2	3	S	15:18:42.122
15	3	2	F	15:18:45.123
16	2	3	F	15:18:47.123
17	3	3	S	15:18:47.123
18	3	3	F	15:18:50.124

Рис. 6: Обработка 3 элементов ($s1 = 1000$, $s2 = 5000$, $s3 = 3000$)

Такую же картинку наблюдаем при обработке 3 элементов. Записи лог-файлов подтверждают, что конвейер отрабатывает корректно.

Выводы

Программа успешно демонстрирует конвейерную обработку. Журналирование отражает информацию об этапах обработки.

Заключение

В ходе работы выполнено следующее:

1) изучен алгоритм конвейеризации;
2) реализована конвейерная система;
3) проведено журналирование событий в конвейерной обработке;
4) описаны и обоснованы полученные результаты в отчете о лабораторной работе, выполненного как расчётно-пояснительная записка.

Список литературы

[1] Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход.- М.:Техносфера, 2009.

[2] Д. Кнут. Искусство программирования, М., Мир, 1978

[3] Воеводин В.В. Математические модели и методы в параллельных процессах. М., 1986

[4] Погорелов Д.А. Применение конвейерной обработки данных на примере сортировки простыми вставками, М., Образование и наука в России и за рубежом 2019 .- Т. 49 , № 1

[5] Корнеев В.В. Параллельные вычислительные системы. М., 1999.

[6] ISO/IEC 14882:2017 [Электронный ресурс]. – Режим доступа: https://www.iso.org/standard/68564.html, свободный – (27.11.2019)

[7] <chrono> [Электронный ресурс]. – Режим доступа: http://www.cplusplus.com/reference/chrono/, свободный – (20.11.2019)