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»

Лабораторная работа № 5
по дисциплине «Анализ алгоритмов»

Тема Организация параллельных вычислений по конвейерному принципу

Студент Бугаков И. С.

Группа ИУ7-54Б

Преподаватели Строганов Ю. В., Волкова Л. Л.

Москва, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель данной лабораторной работы: получить навык организации параллельных вычислений по конвейерному принципу.

Задачи лабораторной работы:

1) разработать ПО осуществляющее обработку html страниц полученных от ПО, разработанного в рамках ЛР №4, и сохранение обработанных данных;

2) провести тестирование разработанного ПО;

3) провести замеры среднего времени существования задач, ожидания в каждой из очередей и обработки на каждой из стадий.

1 Входные и выходные данные

Входные данные программы: путь к директории с html страницами полученных в рамках ЛР №4.

Выходные данные: файл, содержащий замеры необходимых времен, json файл, содержащий извлеченные данные в структурированном виде.

2 Преобразование входных данных в выходные

Для реализации указанной задачи был выбран язык C++ [1], т. к. данный язык предоставляет возможность работать с динамической памятью с помощью контейнерных классов.

Для сохранения очередей задач перед каждой стадией конвейера был написан класс, представленный в листинге 2.1. Указанный класс содержит также необходимые средства взаимоисключения для работы нескольких потоков с очередью одновременно.

Листинг 2.1 — Класс очереди задач

```
1  template<typename T>
2  class task_queue {
3  public:
4    void push(const T &elem) {
5      tasks.push(elem);
6    }
7    T pop() {
8      T top = tasks.front();
9      tasks.pop();
10     return top;
11   }
12   bool empty() {
13     return tasks.empty();
14   }
15   condition_variable cv;
16   bool pushed=false;
17   mutex m;
18
19 private:
20   queue<T> tasks;
21 };
```

В листингах 2.2 2.3 2.4 2.5 представлены различные классы для соответствующих стадий задач.

Листинг 2.2 — Класс задачи на этапе чтения файлов

```
class read_task {
public:
  explicit read_task(string &file_1, string &file_2) : filename_1(
     file_1), filename_2(file_2),
  uuid(boost::lexical_cast<string>(
  boost::uuids::random_generator()())) {
    clock_gettime(CLOCK_MONOTONIC, &times["read_queue_push"]);
    cout << uuid << " " << "created\n";
  }
  vector<string> process();
  string get_uuid() { return uuid; }
  map<string, timespec> times;
private:
  string filename_1, filename_2;
  const string uuid;
};

vector<string> read_task::process() {
  setlocale(LC_ALL, "Russian");
  clock_gettime(CLOCK_MONOTONIC, &times["read_queue_start"]);
  std::ifstream file;
  file.open(filename_1, ios_base::in);

  std::string html_content((std::istreambuf_iterator<char>(file)), std
     ::istreambuf_iterator<char>());
  file.close();
  file.open(filename_2, ios_base::in);
  string url;
  file >> url;
  file.close();
  clock_gettime(CLOCK_MONOTONIC, &times["read_queue_pop"]);
  return {html_content, url};
}
```

Листинг 2.3 — Класс задачи на этапе обработки содержимого файла

```
class parse_task {
public:
  explicit parse_task(vector<string> &values, string uuid_, const map<
     string, timespec> &old_times) : times(
  old_times), content(values), uuid(std::move(uuid_)), complete(true) {
    clock_gettime(CLOCK_MONOTONIC, &times["parse_queue_push"]);
  }
  string get_uuid() { return uuid; }
  task_dto process(); //returns JSON
  bool is_complete() { return complete; };
  map<string, timespec> times;
private:
  string search_for_recipe_detail_picture(GumboNode *node);
  vector<map<string, string>> search_for_ingredients(GumboNode *node);
  vector<string> search_for_recipe_steps(GumboNode *node);
  string search_for_title(const string &input);
  vector<string> content;
  const string uuid;
  bool complete;
};

task_dto parse_task::process() {
  clock_gettime(CLOCK_MONOTONIC, &times["parse_queue_start"]);
  task_dto result;
  result.uuid = uuid;
  result.url = content[1];
  GumboOutput *output = gumbo_parse(content[0].c_str());
  result.title = search_for_title(content[0]);
  result.ingredients_json = to_json(search_for_ingredients(output->root
     ));
  result.steps = search_for_recipe_steps(output->root);
  result.image_url = search_for_recipe_detail_picture(output->root);
  if (result.title.empty() || result.ingredients_json == "[]" || result
     .steps.empty())
    complete = false;
  clock_gettime(CLOCK_MONOTONIC, &times["parse_queue_pop"]);
  return result;
}
```

Листинг 2.4 — Класс задачи на этапе записи в базу данных

```
class write_task {
public:
  explicit write_task(task_dto data, string uuid_, const map<string,
     timespec> &old_times) : times(old_times),
  uuid(std::move(uuid_)),
  content(std::move(
  data)) {
    clock_gettime(CLOCK_MONOTONIC, &times["write_queue_push"]);
  }
  string get_uuid() { return uuid; }
  bool process();
  map<string, timespec> times;
private:
  const string uuid;
  task_dto content;
};

bool write_task::process() {
  clock_gettime(CLOCK_MONOTONIC, &times["write_queue_start"]);
  sqlite3 *db;
  if (sqlite3_open("/home/ivan/lab.db", &db)) {
    clock_gettime(CLOCK_MONOTONIC, &times["write_queue_pop"]);
    return false;
  }
  string sql_statement =
  "insert into lab values ('" + content.uuid + "', '" + content.
     redmine_id + "', '" + content.url + "', '" +
  content.title + "', '" + content.ingredients_json + "', '" +
  to_json_2(content.steps) + "', '" + content.image_url + "');";
  sqlite3_exec(db, sql_statement.c_str(), nullptr, nullptr, nullptr);
  sqlite3_close(db);
  clock_gettime(CLOCK_MONOTONIC, &times["write_queue_pop"]);
  return true;
}
```

Листинг 2.5 — Класс задачи на этапе логирования значений времен

```
class log_task {
public:
  log_task(string uuid_, const map<string, timespec> &old_times) :
     times(old_times), uuid(uuid_) {
    clock_gettime(CLOCK_MONOTONIC, &times["log_queue_push"]);
  }
  void process();
  map<string, timespec> times;
  const string uuid;
};

void log_task::process() {
  clock_gettime(CLOCK_MONOTONIC, &times["log_queue_start"]);
  clock_gettime(CLOCK_MONOTONIC, &times["log_queue_pop"]);
  ofstream out;
  out.open("logs/" + uuid + ".json", ios_base::out);
  out << "{";
    for (auto it = times.begin(); it != times.end();) {
      out << "\"" << it->first << "\": " << it->second.tv_sec + it->
         second.tv_nsec * NANO;
      if (++it != times.end())
      out << ",";
    }
    out << "}";
  out.close();
}
```

Для передачи необходимой информации о рецепте был написан класс представленный в листинге 2.6.

Листинг 2.6 — Класс для передачи данных о рецепте

```
class task_dto {
public:
  string uuid;
  static const string redmine_id;
  string url;
  string title;
  string ingredients_json;
  vector<string> steps;
  string image_url;
};
```

Для поиска необходимой информации в html файле использовалась библиотека gumbo [2].

3 Тестирование

Для проверки работоспособности было проведено тестирование по методологии черного ящика. Тест представлен в таблице 3.1.

Таблица 3.1 — Тестовые данные программы

Входные данные	Ожидаемые выходные данные	Полученные выходные данные
10.html	Представлены на рисунке 3.1	Совпадают с представленными на рисунке 3.1

Рисунок 3.1 — Пример логируемых данных о задаче

Все тесты были успешно пройдены.

4 Примеры работы программы

На рисунке 4.1 представлен пример полученных данных. Всего получено 149 записей с данными.


```
{
  "id": "b9547ac0-0023-4ff7-9b79-07e11a46ea4a",
  "issue_id": "9203",
  "url": "https://sostra.ru/recipe-book/meal/barbekyu-shashlyk/svinina-v-pikantnom-souse/",
  "title": "Свинина в пикантном соусе",
  "ingredients": [
    {
      "name": "Соус Черный перец (Black Pepper) «Sen Soy Premium»",
      "quantity": "120",
      "unit": "г"
    },
    {
      "name": "Шейка свиная",
      "quantity": "500",
      "unit": "г"
    },
    {
      "name": "Зелень для украшения",
      "quantity": "",
      "unit": ""
    }
  ],
  "steps": [
    "Свинину нарезать на стейки.",
    "Замариновать стейки в соусе «Черный перец» от Sen Soy Premium на 30 - 60 мин.",
    "Затем приготовить на гриле до готовности. Выложить на блюдо и украсить рубленной зеленью."
  ],
  "image_url": "/upload/resize_cache/iblock/89d/550_367_2/275493062.jpg"
},
```


Рисунок 4.1 — Данные о рецепте

Пример логируемых данных представлен на рисунке 4.2.

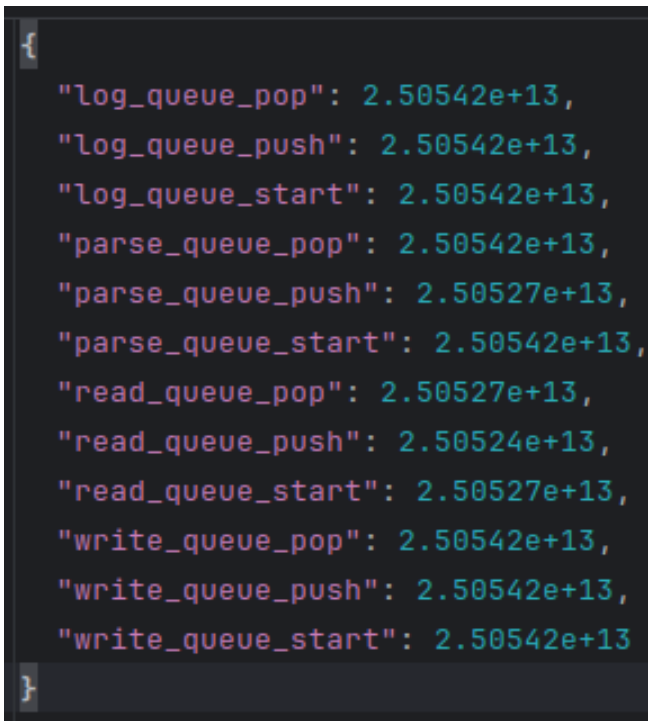

Рисунок 4.2 — Пример логируемых данных о задаче

5 Описание исследования

Исследования проводились на машине со следующими характеристиками:

— процессор Intel(R) Core(TM) i5-10210U, тактовая частота 1.60 ГГц;

— оперативная память: 16 ГБ;

— операционная система: Ubuntu 22.04.4 LTS.

Результаты замеров приведены в таблице 5.1.

Таблица 5.1 — Средние времена этапов выполнения в мкс

Среднее время существования задачи	7805728
Среднее время ожидания в очереди чтения	8645
Среднее время ожидания в очереди обработки	42164
Среднее время ожидания в очереди записи	14
Среднее время ожидания в очереди логирования	14
Среднее время чтения	208
Среднее время обработки	813
Среднее время записи	169
Среднее время логирования	0

В таблице 5.2 приведены метки задач, отсортированные по возрастанию времени, показывающие параллельное выполнение различных этапов конвейера.

Таблица 5.2 — Метки задач отсортированные по возрастанию времени

Время с неопределенного момента в прошлом в мкс	Идентификатор задачи и ее имя
735559	b01fb701-44a6-4e9d-80f2-700106696499 read_queue_push
742961	4a91915e-5ddd-4c2f-bde9-ed82f260607a read_queue_push
753717	b01fb701-44a6-4e9d-80f2-700106696499 read_queue_start
755113	b01fb701-44a6-4e9d-80f2-700106696499 read_queue_pop
755146	b01fb701-44a6-4e9d-80f2-700106696499 parse_queue_push
789426	79af8472-5819-4abb-834a-d56d493c693d read_queue_start
790817	79af8472-5819-4abb-834a-d56d493c693d read_queue_pop
790860	79af8472-5819-4abb-834a-d56d493c693d parse_queue_push
999207	b01fb701-44a6-4e9d-80f2-700106696499 parse_queue_start
1011835	b01fb701-44a6-4e9d-80f2-700106696499 parse_queue_pop
1011868	b01fb701-44a6-4e9d-80f2-700106696499 write_queue_push
1011989	b01fb701-44a6-4e9d-80f2-700106696499 write_queue_start
1017574	b01fb701-44a6-4e9d-80f2-700106696499 write_queue_pop
1017606	b01fb701-44a6-4e9d-80f2-700106696499 log_queue_push
1017681	b01fb701-44a6-4e9d-80f2-700106696499 log_queue_start
1017686	b01fb701-44a6-4e9d-80f2-700106696499 log_queue_pop
1496252	79af8472-5819-4abb-834a-d56d493c693d parse_queue_start
1509341	79af8472-5819-4abb-834a-d56d493c693d parse_queue_pop
1509381	79af8472-5819-4abb-834a-d56d493c693d write_queue_push
1509502	79af8472-5819-4abb-834a-d56d493c693d write_queue_start
1515815	79af8472-5819-4abb-834a-d56d493c693d write_queue_pop
1515844	79af8472-5819-4abb-834a-d56d493c693d log_queue_push
1515915	79af8472-5819-4abb-834a-d56d493c693d log_queue_start
1516937	79af8472-5819-4abb-834a-d56d493c693d log_queue_pop

ЗАКЛЮЧЕНИЕ

Целью данной работы была организация параллельных вычислений по конвейерному принципу с использованием нативных потоков.

В ходе лабораторной работы были выполнены следующие задачи:

1) разработано ПО осуществляющее обработку html страниц полученных от ПО, разработанного в рамках ЛР №4, и сохраняющее обработанные данные;
2) проведено тестирование разработанного ПО;
3) проведены замеры среднего времени существования задач, ожидания в каждой из очередей и обработки на каждой из стадий.

По итогам анализа было выявлено, что поиск данных в html файле занимает наибольшее время среди выполнения этапов конвейера. Соответственно, ожидание в очереди на обработку содержимого файла занимает также максимально время среди времен ожидания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. C++ documentation [Электронный ресурс]. Режим доступа: `https://isocpp.org/std/the-standard`. Дата обращения 24.12.2024.

2. gumbo documentation [Электронный ресурс]. Режим доступа: `https://matze.github.io/clib-doc/gumbo-parser/index.html`. Дата обращения 24.12.2024.