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 №4
по дисциплине «Анализ алгоритмов»

Тема Параллельные вычисления на основе нативных потоков

Студент Звягин Д.О.
Группа ИУ7-53Б
Преподаватель Волкова Л.Л., Строганов Д.В.

Москва, 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **4**

1 Требования к разрабатываемому ПО **5**

2 Разработка ПО **6**

3 Примеры работы ПО **10**

4 Тестирование ПО **11**

4.1 Описание исследования 11

4.2 Вывод 12

ЗАКЛЮЧЕНИЕ **13**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **14**

ВВЕДЕНИЕ

Параллелизм — это возможность выполнения нескольких процессов одновременно [1].

С точки зрения программирования, параллелизм — это возможность систем производить вычисления одновременно.

В тех ситуациях, где это возможно и обоснованно, использование параллельных вычислений может привести к улучшению временных характеристик программ.

Такими случаями могут быть, например, задачи в которых нужно произвести сложную обработку нескольких независимых сущностей, или задачи, где возможно производить вычисления в ожидании ввода/вывода для других элементов программы [2].

В современных системах параллельные вычисления разделяют на два типа [2]:

— потоковая параллельность;

— параллельность, основанная на процессах.

Целью данной работы является разработка программного обеспечения, извлекающего веб-страницы онлайн ресурса chiefs.kz.

Для достижения этой цели нужно выполнить следующие задачи:

— рассмотреть структуру страниц с рецептами ресурса chiefs.kz;

— разработать ПО, выполняющее извлечение веб-страницы;

— разработать ПО, выполняющее эту задачу параллельно;

— реализовать разработанное ПО;

— проанализировать полученные реализации по временным характеристикам.

1 Требования к разрабатываемому ПО

К разрабатываемому ПО предъявляется несколько требований:

— на вход подаются ссылки на веб-страницы с рецептами с ресурса http://chiefs.kz/;

— на выход подаются html файлы с содержанием веб-страниц.

На момент обращения к ресурсу http://chiefs.kz/ (21 декабря 2024 г.), веб-ресурс выглядит, как сайт с содержимым-заглушкой. На нём доступно всего 4 рецепта, поэтому дополнительных программ для получения списка рецептов не требуется.

Для замеров временных характеристик, ссылки будут использованы повторно, а содержимое страниц будет загружаться заново.

2 Разработка ПО

Для реализации алгоритмов был выбран язык программирования c++ стандарта 14882 (c++20) [3]. Этот язык достаточен для выполнения работы, так как его стандартная библиотека предлагает средства для работы с нативными потоками.

Для доступа к интернет ресурсам была использована библиотека libcurl [4]. Она предоставляет интерфейс для работы с интернет запросами, а также предоставляет средства для получения и расшифровки ответов.

Поскольку libcurl — изначально библиотека для языка Си, приходится самостоятельно создавать объект CURL и освобождать из под него память. Для гарантии отсутствия утечек и упрощённой загрузки страниц, был написан класс-обёртка над библиотекой libcurl, интерфейс, которого представлен в листинге 2.1, а реализация — в листинге 2.2.

Листинг 2.1 — Интерфейс класса-обёртки над библиотекой libcurl

```
#pragma once //                                                    ,

#include <curl/curl.h>
#include <string>

using namespace std;

class CurlWrapper {
public:
  CurlWrapper();
  ~CurlWrapper();

  string get_html(const string &url);

private:
  CURL *instance;
};
#include <sqlite3.h>
```

Листинг 2.2 — Реализация класса-обёртки над библиотекой libcurl

```
#include "curlwrap.hpp"
#include <format>

namespace {
size_t concatenate(void *data, size_t size, size_t nmemb, void *buf) {
  ((std::string *)buf)->append((char *)data, size * nmemb);
  return size * nmemb;
}
```

```
} // namespace

CurlWrapper::CurlWrapper() { instance = curl_easy_init(); }

CurlWrapper::~CurlWrapper() { curl_easy_cleanup(instance); }

string CurlWrapper::get_html(const string &url) {
  if (!instance)
    throw "curl failed to init";

  string buf;

  curl_easy_setopt(instance, CURLOPT_URL, url.c_str());
  curl_easy_setopt(instance, CURLOPT_ENCODING, "");
  curl_easy_setopt(instance, CURLOPT_WRITEFUNCTION, concatenate);
  curl_easy_setopt(instance, CURLOPT_WRITEDATA, &buf);

  CURLcode rc = curl_easy_perform(instance);

  if (rc != CURLE_OK) {
    throw format("curl failed: {}", curl_easy_strerror(rc));
  }

  return buf;
}
```

Следует заметить, что отдельный объект CURL предназначен для выполнения одного запроса в один момент времени, поэтому использовать один объект класса CurlWrapper в нескольких потоках не получится.

Для реализации параллельного выполнения запросов, был написан класс, который распределяет задачи между несколькими потоками. Его интерфейс и реализация представлены на листингах 2.3 и 2.4 соответственно.

Листинг 2.3 — Интерфейс класса-обёртки для параллельного вызова

```
#pragma once

#include <functional>
#include <queue>
#include <thread>
#include <vector>

using namespace std;
```

```
class ThreadPool {
public:
  ThreadPool(size_t threads) : threadcount(threads) {
    if (threadcount == 0)
      threadcount = 1;
    if (threadcount > thread::hardware_concurrency())
      threadcount = thread::hardware_concurrency();
  }

  void push_to_queue(function<void(void)> f);
  void run();

private:
  void dispatch(function<void(void)> f);

  queue<function<void(void)>> q;
  vector<thread> threads;
  size_t threadcount;
};
```

Листинг 2.4 — Реализация класса-обёртки для параллельного вызова

```
#include "threadpool.hpp"
#include <iostream>
#include <thread>

void ThreadPool::push_to_queue(function<void(void)> f) { q.push(f); }

void ThreadPool::run() {
  while (!q.empty()) {
    // cout << threadcount << '\r';
    if (threadcount > 0) {
      dispatch(std::move(q.front()));
      q.pop();
    }
  }

  for (auto &t : threads)
    t.join();

  threads.clear();
}
```

```
void ThreadPool::dispatch(function<void(void)> f) {
  threadcount--;
  threads.emplace_back([this, f]() {
    f();
    threadcount++;
  });
}
```

Используя эти классы, было написано приложение, которые принимает с клавиатуры количество потоков и запросов, которые необходимо совершить.

Полученные ответы запросов записываются в файлы i.html, где i — порядковый номер запроса, а сами файлы попадают в директорию out.

Дальнейшая обработка файлов будет проводиться в лабораторной работе номер 5.

3 Примеры работы ПО

В качестве примера будет рассмотрена одна из веб-страниц сайта `chiefs.kz`. На рисунке 3.1 изображена веб-страница в браузере.

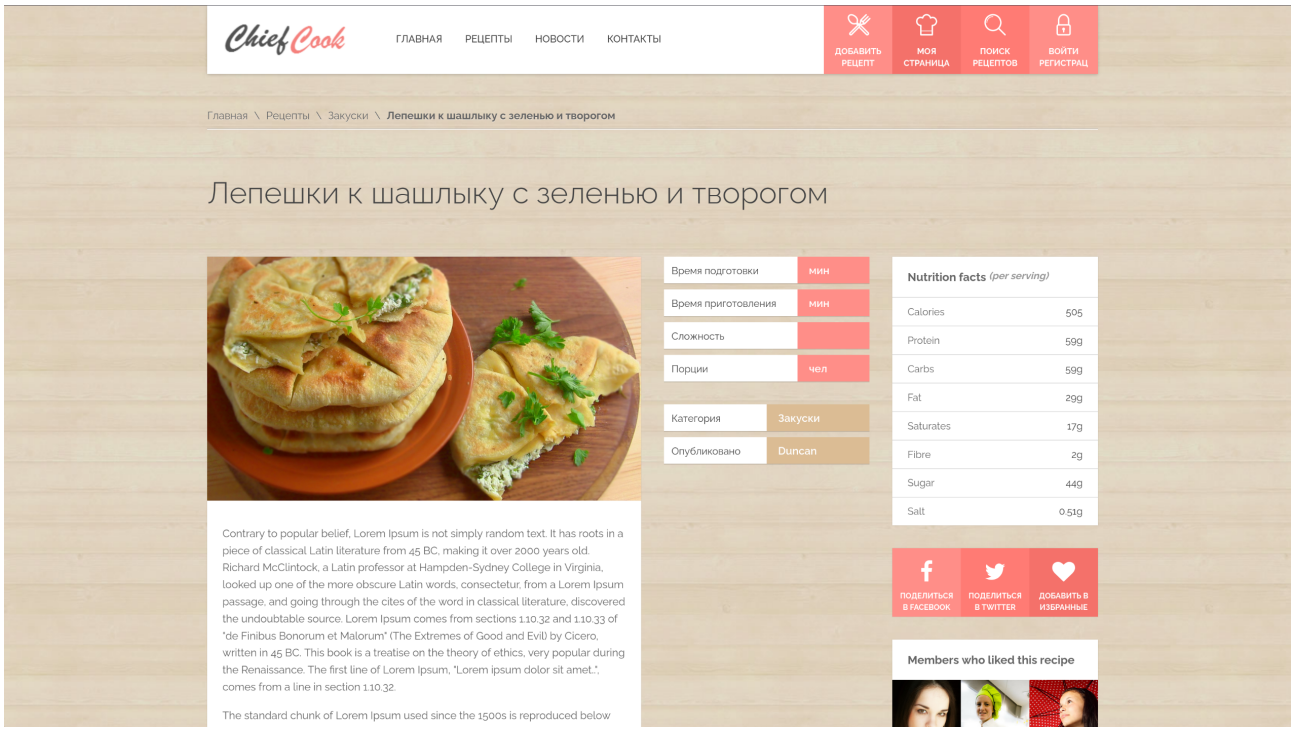

Рисунок 3.1 — Одна из страниц сайта chiefs.kz

Результатом работы алгоритма стал html файл размером 26,3 килобайта.

4 Тестирование ПО

4.1 Описание исследования

Было проведено исследование временных характеристик программы в зависимости от количества потоков при фиксированном количестве загружаемых страниц.

При тестировании, для каждого значения количества потоков было проведено 10 запусков программы, а в результаты записано среднее значение времени.

Результаты тестирования приведены в таблице 4.1, и на рисунке 4.1.

Таблица 4.1 — Таблица зависимости времени скачивания 1024 страниц от кол-ва потоков

Кол-во потоков	Время выполнения (с)
0	217.466
1	223.516
2	120.427
4	68.336
8	31.24
16	21.424
32	21.632
64	21.024

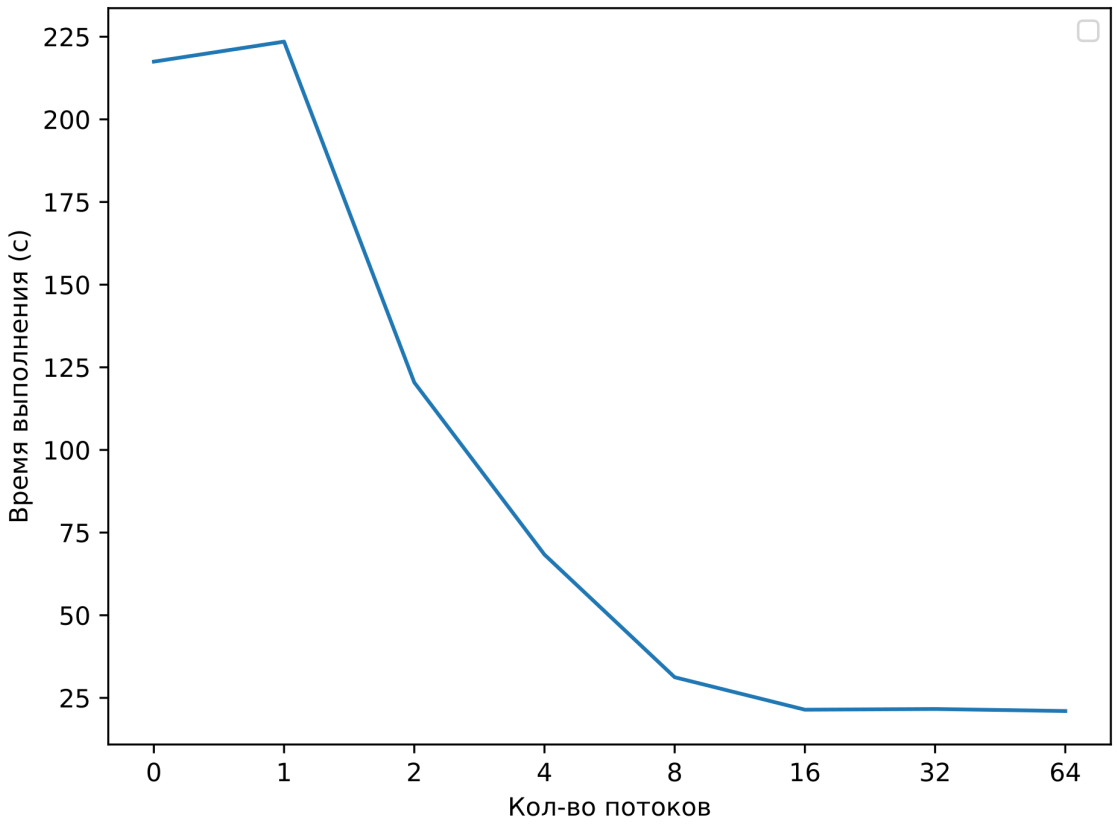

Рисунок 4.1 — График зависимости времени выполнения от кол-ва потоков

4.2 Вывод

В результате измерений было обнаружено, что при загрузке 1024 страниц с ресурса chiefs.kz, увеличение количества потоков, скорость работы программы значительно увеличивается, однако при создании большего количества потоков, чем то, что поддерживается конкретной машиной, прирост скорости прекращается.

ЗАКЛЮЧЕНИЕ

В результате измерений было обнаружено, что увеличение количества потоков, может увеличить скорость работы приложения в несколько раз, однако этот прирост в скорости ограничен количеством потоков, поддерживаемых машиной.

В ходе результате выполнения работы были получены файлы страниц веб-ресурса chiefs.kz в формате html.

Были выполнены следующие задачи:

— была рассмотрена структура страниц с рецептами ресурса chiefs.kz;

— было разработано ПО, выполняющее извлечение веб-страниц;

— было разработано ПО, выполняющее эту задачу параллельно;

— было реализовано разработанное ПО;

— полученные реализации были проанализированы по временным характеристикам.

Цели и задачи лабораторной работы выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. David B. Skillicorn, Domencio Talia Models and languages for parallel computation // Журнал ACM Comput. Surv. // Нью Йорк: Association for Computing Machinery // 1998г. // страницы 123-169

2. Таненбаум Э., Бос Х. Современные операционные системы. // Таненбаум Э., Бос Х. 4-е издание // СПб.: Питер // 2015г. // 1120с.

3. ISO International Standard ISO/IEC 14882:2020(E) [Working Draft] – Programming Language C++ // Geneva, Switzerland: International Organization for Standardization (ISO)

4. Документация библиотеки libcurl (интерфейс для интернет запросов) / [Электронный ресурс] // [сайт] URL: `https://curl.se/libcurl/c/` (дата обращения: 21 декабря 2024 г.)