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 курсу «Анализ алгоритмов»

на тему: «Организация параллельных вычислений по конвейерному принципу»

Студент ИУ7-53Б _____ Паламарчук А. Н.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Кормановский М. В.
(Подпись, дата) (И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1 Входные и выходные данные 3

2 Преобразование входных данных в выходные 3

3 Примеры работы программы 5

4 Тестирование 6

5 Описание исследования 6

ЗАКЛЮЧЕНИЕ 7

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 8

ВВЕДЕНИЕ

Параллельные вычисления – это использование нескольких или многих вычислительных устройств для одновременного выполнения разных частей одной программы [1].

Цель работы — получение навыка организации параллельных вычислений по конвейерному принципу.

Задачи работы:

— определение структуры html файла, содержащего рецепт;

— разработка программного обеспечения, выполняющего обработку рецептов в конвейерном режиме;

— анализ времен ожидания в очередях и обработки на каждом этапе.

1 Входные и выходные данные

Входными данными являются html файлы с рецептами, загруженные с сайта https://www.gotovim.ru. Каждый файл содержит ровно один рецепт. Выходными данными являются записи характеристик каждой из выполненных задач в базе данных.

2 Преобразование входных данных в выходные

Программа читает данные из файла, извлекает необходимые, записывает извлеченные данные в базу данных. Реализация алгоритма извлечения данных представлена в листинге 2.1.

Листинг 2.1 – Функции разбирают входные данные и извлекают записи

```
void extract_elem(const std::string& page_content, std::string&
   a, const std::string& mask) {
    std::regex elem_regex(mask);
    std::smatch match;
    if (std::regex_search(page_content, match, elem_regex))
        a = match.str(1);
    else
```

```
        a = EMPTY;
}

void extract_set(const std::string& page_content,
   std::set<std::string>& a, const std::string& mask) {
    std::regex set_regex(mask);
    std::sregex_iterator begin =
       std::sregex_iterator(page_content.begin(),
       page_content.end(), set_regex);
    std::sregex_iterator end = std::sregex_iterator();

    std::wstring_convert<std::codecvt_utf8<wchar_t>> converter;
    for (std::sregex_iterator i = begin; i != end; ++i)
        a.insert(converter.to_bytes(clean_string(converter.from_bytes
}

void Task::parse_html() {
    std::string url;
    std::string title;
    std::set<std::string> ingredients;
    std::set<std::string> steps;
    std::string image_url;

    extract_elem(this->page_content, url,
       R"(<link\s+rel=\"canonical\"\s+href=\"(.*?)\"\/>)");
    extract_elem(this->page_content, title,
       R"(<h1[^>]*>(.*?)<\/h1>)");
    extract_set(this->page_content, ingredients, R"(<li
       itemprop=\"recipeIngredient\">(.*?)<\/li>)");
    extract_set(this->page_content, steps, R"(<p
       itemprop=\"recipeInstructions\">(.*?)<\/p>)");
    extract_elem(this->page_content, image_url,
       R"(<img\s+itemprop=\"image\"\s+itemprop=\"[^"]+\"\s+width=\"[^"

    this->url = url;
    this->title = title;
    this->ingredients = ingredients;
    this->steps = steps;
    this->image_url = image_url;
}
```

3 Примеры работы программы

На листинге 3.1 представлен пример работы программы. На вход программа получает три html файла. Каждый файл проходит следующие этапы обработки: чтение, извлечение данных, запись в базу данных.

Листинг 3.1 – Примеры работы программы

```
Enter the number of tasks to process: 3
Task: 1 created
File: utf_8_18268913385945316603.html
Task: 1 added to read queue
Task: 2 created
File: utf_8_15546927078813418238.html
Task: 1 start reading
Task: 2 added to read queue
Task: 3 created
File: utf_8_7650529285466088104.html
Task: 3 added to read queue
Task: 1 end reading
Task: 1 added to extract queue
Task: 2 start reading
Task: 2 end reading
Task: 2 added to extract queue
Task: 3 start reading
Task: 3 end reading
Task: 1 start extracting
Task: 3 added to extract queue
Task: 1 end extracting
Task: 1 added to write queue
Task: 2 start extracting
Task: 1 start writing
Task: 1 end writing
Task: 1 destructed
Task: 2 end extracting
Task: 2 added to write queue
Task: 3 start extracting
Task: 2 start writing
Task: 2 end writing
Task: 2 destructed
Task: 3 end extracting
Task: 3 added to write queue
Task: 3 start writing
```

```
Task: 3 end writing
Task: 3 destructed
```

4 Тестирование

Тестирование программы проводилось на 100 входных html файлов с различными рецептами. Проверка 5 случайных файлов: сравнивались данные, извлеченные программой, с эталонными значениями в html файлах. Тестирование было успешно пройдено.

5 Описание исследования

Технические характеристики устройства:

— операционная система Manjaro Linux x86_64;

— процессор Ryzen 5500U 6 ядер, тактовая частота 2.1 ГГц;

— оперативная память 16 Гбайт.

В ходе исследования были получены следующие характеристики:

— среднее время существования задачи;

— среднее время ожидания задачи в каждой из очередей;

— среднее время обработки задачи на каждой из стадий.

Исследование проводилось для 100 html файлов.

Таблица 5.1 – Среднее значение времен на каждом этапе

Этап выполнения	Среднее время, мс
Ожидание в очереди чтения	70.35
Ожидание в очереди извлечения	1412.39
Ожидание в очереди записи	0.00
Обработка чтения	2.00
Обработка извлечения	29.42
Обработка записи	1.04
Время жизни задачи	1516.97

По результатам проведенного исследования сделан вывод о том, что при организации параллельных вычислений по конвейерному принципу необходимо уделять особое внимание "узким"местам, из-за которых возникают простои остальных частей конвейера в ожидании поступления новых данных.

В данной лабораторной работе извлечение данных вызывает простой, и для улучшения скорости обработки необходимо модифицировать именно эту часть программы.

ЗАКЛЮЧЕНИЕ

Получены навыки организации параллельных вычислений по конвейерному принципу. Цель работы достигнута. Решены все поставленные задачи:

— определена структура html файла, содержащего рецепт;

— разработано программное обеспечение, выполняющее обработку рецептов в конвейерном режиме;

— проведен анализ времен ожидания в очередях и обработки на каждом этапе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Гафаров Ф. М.*, *Галимянов А. Ф.* Параллельные вычисления: учеб. пособие. — Казань : Изд-во Казан. ун-та, 2018. — С. 149.