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 №4 по дисциплине «Анализ Алгоритмов»

Тема Параллельные вычисления на основе нативных потоков

Студент Смирнов И.В.

Группа ИУ7-52Б

Преподаватель Волкова Л. Л., Строганов Д.В.

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В общем случае поток исполнения представляет собой последовательность инструкций, выполняемых на выделенном процессорном ядре и управляемых планировщиком операционной системы. Потоки могут быть приостановлены или заблокированы в процессе выполнения. Они создаются внутри процесса и совместно используют его ресурсы, такие как оперативная память и дескрипторы файлов. Такой механизм организации потоков называется нативными потоками [1]. Нативные потоки обеспечивают эффективное использование системных ресурсов и позволяют выполнять несколько задач параллельно в рамках одного процесса, что существенно повышает производительность приложений.

Цель лабораторной работы — сравнить основные принципы последовательных вычислений с параллельными на основе нативных потоков. Для достижения поставленной цели необходимо выполнить следующие задачи:

— описать входные, выходные данные, а также преобразования входных данных в выходные;

— реализовать два алгоритма для загрузки контента из $HTML$—страниц: последовательный и параллельный с использованием нативных потоков;

— протестировать разработанные алгоритмы по методологии черного ящика;

— описать пример работы программы на конкретном случае;

— выполнить сравнительный анализ последовательного и параллельного алгоритма по времени выполнения в зависимости от количества обрабатываемых страниц.

1 Входные и выходные данные

Входные данные: базовый URL—адрес веб-сайта, с которого будут загружаться страницы; режим работы — последовательный или параллельный; количество страниц для загрузки в последовательном режиме или количество потоков (от 1 до 16) в параллельном режиме.

Выходные данные: файлы, каждый из которых содержит ссылки на рецепты блюд конкретной страницы с указанного веб-сайта.

2 Преобразование данных

В интерфейсе программы выбирается один из двух режимов: последовательный или параллельный. По нажатии кнопки «Начать парсинг» программа считывает базовый URL—адрес, а также количество страниц или количество потоков, в зависимости от выбранного режима, из полей ввода. Программа загружает $HTML$—контент страниц с указанного веб-сайта, а затем сохраняет загруженные страницы в соответствующую директорию ($seqfiles$ для последовательного режима и $parfiles$ для параллельного). Также программа выводит в консоль сообщения о ходе выполнения и возможных ошибках.

3 Пример работы программы

На рисунках 3.1 — 3.2 представлен пример работы программы в последовательном режиме. В данном случае программа последовательно обрабатывает 4 страницы сайта $https://food.ru/recipes/zakuski$; с каждой страницы программа сохраняет ссылки на рецепты в отдельный файл в директорию $seqfiles$. Содержимое одного из файлов представлено на рисунке 3.3. Сообщения, связанные с парсингом, программа выводит в консоль.

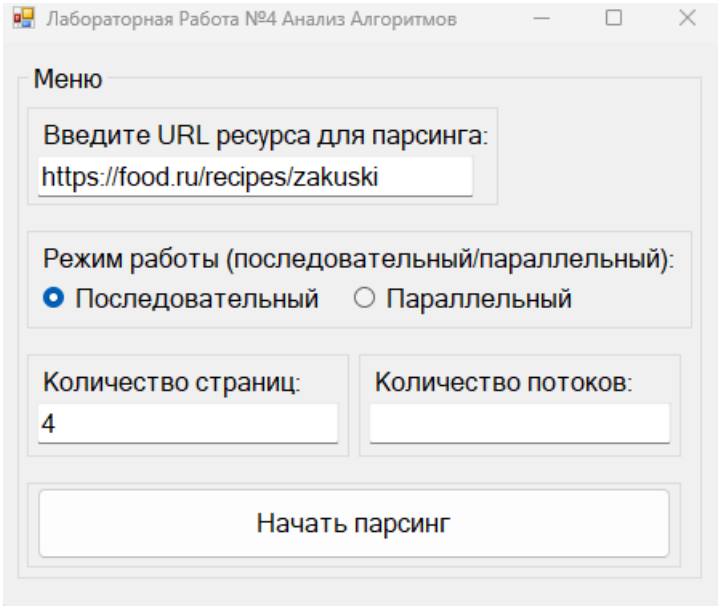

Рисунок 3.1 – Ввод данных в интерфейс приложения

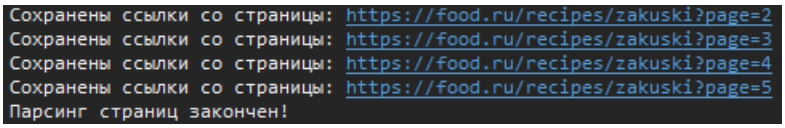

Рисунок 3.2 – Сообщения о ходе выполнения парсинга

На рисунках 3.4 — 3.5 представлен пример работы программы в параллельном режиме. В данном случае программа создает для обработки каждой страницы свой отдельный поток. После создания всех потоков и выдачи им задач главный поток блокируется на время парсинга всех страниц. Каждый поток сохраняет ссылки на рецепты в отдельный файл в директорию $parfiles$. Сообщения, связанные с парсингом, каждый поток выводит в консоль по принципу взаимоисключения [2], так как консоль в данном случае является разделяемым ресурсом.

```
https://food.ru/recipes/228162-liubov-morkov
https://food.ru/recipes/227774-vetnamskii-sendvich-s-farshem-i-ovoshchami
https://food.ru/recipes/226448-bento-boks-s-onigiri-i-japonskim-omletom
https://food.ru/recipes/226447-bento-boks-s-sendvichem-i-salatom
https://food.ru/recipes/228002-tvorozhnye-percytvorozhnye-percy
https://food.ru/recipes/227850-opjata-po-koreiski
https://food.ru/recipes/227933-marinovannaja-kapusta-s-lukom
https://food.ru/recipes/227366-sendvich-s-arbuzom
https://food.ru/recipes/227364-briusselskaja-kapusta-na-zimu
https://food.ru/recipes/227351-pomidory-konfi
https://food.ru/recipes/227347-maslo-s-petrushkoi-i-chesnokom
https://food.ru/recipes/227344-marinovannye-masliny
https://food.ru/recipes/227277-mochenye-jabloki-s-gorchicei
https://food.ru/recipes/224435-tosty-s-tvorozhnym-syrom-i-marinovannymi-cherri
https://food.ru/recipes/227168-ikra-iz-veshenok
https://food.ru/recipes/226502-kartoshka-tornado
https://food.ru/recipes/226346-pashtet-iz-rechnoii-ryby
https://food.ru/recipes/226321-salat-kazachii
https://food.ru/recipes/226306-vengerskii-salat
https://food.ru/recipes/225678-cvetnaja-kapusta
```

Рисунок 3.3 – Содержимое выходного файла

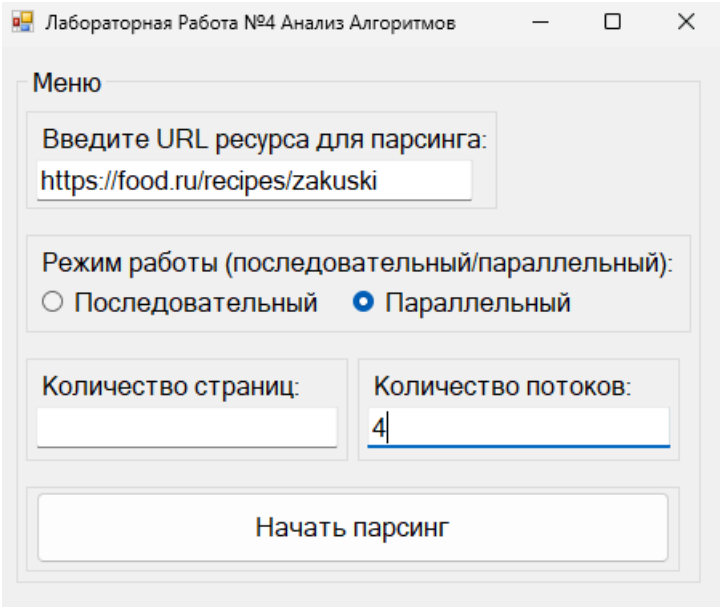

Рисунок 3.4 – Ввод данных в интерфейс приложения

```
Сохранены ссылки со страницы: https://food.ru/recipes/zakuski?page=2
Поток 0x4d88 завершился с кодом 0 (0x0).
Сохранены ссылки со страницы: https://food.ru/recipes/zakuski?page=4
Поток 0x127c завершился с кодом 0 (0x0).
Сохранены ссылки со страницы: https://food.ru/recipes/zakuski?page=3
Поток 0x7610 завершился с кодом 0 (0x0).
Сохранены ссылки со страницы: https://food.ru/recipes/zakuski?page=5
Поток 0x5f64 завершился с кодом 0 (0x0).
Парсинг страниц закончен!
```

Рисунок 3.5 – Сообщения о ходе выполнения парсинга

4 Тестирование

Выполнено тестирование реализованной программы по методологии черного ящика. В таблице 4.1 представлено описание тестов. Все тесты пройдены успешно.

Таблица 4.1 – Описание тестовых случаев

№	Входные данные	Ожидаемый результат	Результат теста
1	Корректный базовый URL, последовательный режим, 5 страниц	Успешная загрузка 5 страниц, сохранение в `seqfiles`	Пройден
2	Пустой базовый URL	Вывод сообщения об ошибке, запрос корректного URL	Пройден
3	Некорректный базовый URL (без протокола)	Вывод сообщения об ошибке, запрос корректного URL	Пройден
4	Корректный базовый URL, параллельный режим, 4 потока	Успешная загрузка страниц, сохранение в `parfiles`	Пройден
5	Корректный базовый URL, параллельный режим, 20 потоков (превышение допустимого числа)	Вывод сообщения об ошибке, запрос корректного числа потоков (1-16)	Пройден
6	Отключенное интернет-соединение	Вывод сообщений об ошибках при загрузке страниц	Пройден
7	Корректный URL, последовательный режим, 0 страниц	Вывод сообщения об ошибке, запрос корректного числа страниц	Пройден
8	Корректный URL, параллельный режим, отрицательное число потоков	Вывод сообщения об ошибке, запрос корректного числа потоков	Пройден

5 Описание исследования

Зависимости времени обработки страниц от количества страниц для последовательного и параллельного алгоритма (с 16 потоками) представлены на рисунке 5.1. Зависимости времени обработки страниц от количества страниц для последовательного и параллельного алгоритма (с 10000 потоками) представлены на рисунке 5.2.

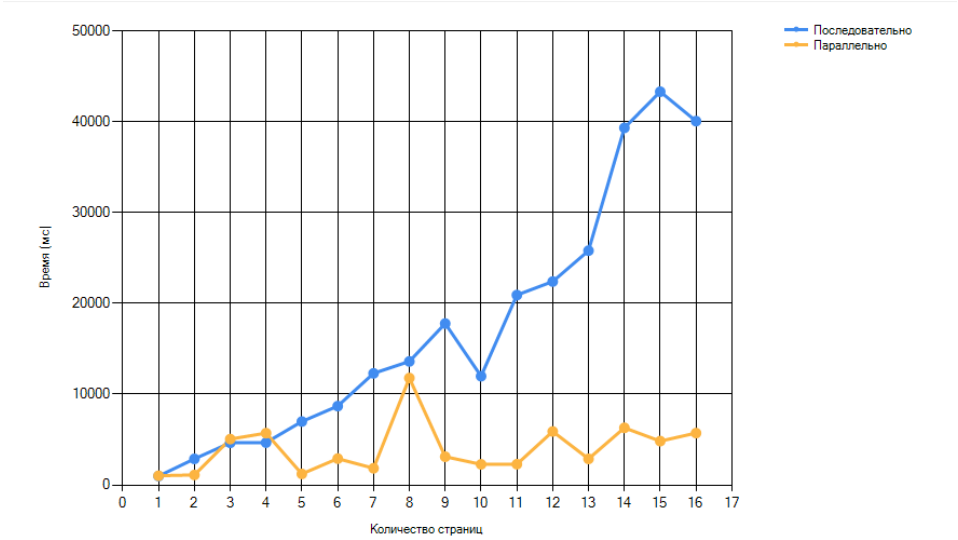

Рисунок 5.1 – Сравнение алгоритмов по времени

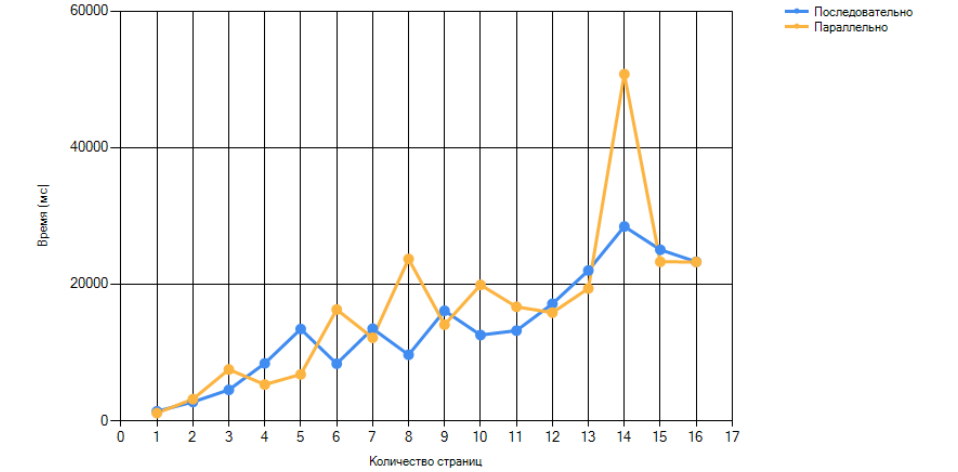

Рисунок 5.2 – Сравнение алгоритмов по времени

В результате исследования было получено, что параллельный алгоритм, использующий 16 нативных потоков, работает быстрее последовательного при количестве страниц большим или равном 5. При обработки 16 страниц параллельный алгоритм работает почти в 8 раз быстрее последовательного. Однако параллельный алгоритм, использующий 10000 нативных потоков, работает при некотором количестве страниц медленнее последовательного. Так как операционная система вынуждена затрачивать ресурсы на поддержку большого количества потоков, то общее время выполнения алгоритма увеличилось.

ЗАКЛЮЧЕНИЕ

Цель работы достигнута: сравнены основные принципы последовательных вычислений с параллельными на основе нативных потоков.

В ходе выполнения данной лабораторной работы были решены следующие задачи:

— описаны входные, выходные данные, а также преобразования входных данных в выходные;

— реализованы два алгоритма для загрузки контента из $HTML$—страниц: последовательный и параллельный с использованием нативных потоков;

— протестированы разработанные алгоритмы по методологии черного ящика;

— описаны примеры работы программы на конкретных случаях;

— выполнен сравнительный анализ последовательного и параллельного алгоритма по времени выполнения в зависимости от количества обрабатываемых страниц.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Threading Model Overview [Электронный ресурс]. URL: `http://justin.harmonize.fm/index.php/2008/09/threading-model-overview/` (дата обращения: 20.10.2024).

[2] Ю. В. Кочержинская. Курс лекций по дисциплине «Теория вычислительных процессов», 2014г.