**О Т Ч Е Т**

о выполнении лабораторной работы № 5

МОСКВА 2020

Оглавление

[1 Задание 3](#_Toc38890011)

[2 Теоретическая справка 4](#_Toc38890012)

[3 Спецификации разработанных программ 5](#_Toc38890013)

[3.1 Программное средство обработки данных на кластере Disco 5](#_Toc38890014)

[4 Алгоритм решения задачи 6](#_Toc38890015)

[5 Тестирование программных средств 7](#_Toc38890016)

[5.1 Проверка корректности реализованных приложений 7](#_Toc38890017)

[5.2 Проведение экспериментов 10](#_Toc38890018)

[5.3 Результаты экспериментов 10](#_Toc38890019)

[5.4 Анализ полученных результатов 10](#_Toc38890020)

# Задание

1. В качестве исходных данных для работы использовать записи, сформированные в файловой системе DDFS в рамках выполнения лабораторной работы № 4.

2. Разработать MapReduce-алгоритм решения выбранного варианта задания с учетом формата входных данных формируемого сценарием, реализованным в рамках лабораторной работы № 4.

3. Реализовать разработанный в пункте 2 алгоритм в виде сценария на языке программирования Python с использованием API-интерфейса доступа к вычислительным ресурсам MapReduce-кластера. Разработанный сценарий должен за один запуск обрабатывать несколько групп входных данных и измерять время обработки для каждой группы.

# Теоретическая справка

Поисковый индекс — структура данных, которая содержит информацию о документах и используется в поисковых системах. Индекси́рование, совершаемое поисковой машиной — процесс сбора, сортировки и хранения данных с целью обеспечить быстрый и точный поиск информации. Создание индекса включает междисциплинарные понятия из лингвистики, когнитивной психологии, математики, информатики и физики. Веб-индексированием называют процесс индексирования в контексте поисковых машин, разработанных, чтобы искать веб-страницы в Интернете.

Популярные поисковые машины сосредотачиваются на полнотекстовой индексации документов, написанных на естественных языках. Мультимедийные документы, такие как видео и аудио и графика, также могут участвовать в поиске.

Метапоисковые машины используют индексы других поисковых сервисов и не хранят локальный индекс, в то время как поисковые машины, основанные на кешированных страницах, долго хранят как индекс, так и текстовые корпусы. В отличие от полнотекстовых индексов, частично-текстовые сервисы ограничивают глубину индексации, чтобы уменьшить размер индекса. Большие сервисы, как правило, выполняют индексацию в заданном временном интервале из-за необходимого времени и затрат на обработку, в то время как поисковые машины, основанные на агентах, строят индекс в масштабе реального времени.

# Спецификации разработанных программ

## Программное средство обработки данных на кластере Disco

Интерфейс созданного сценария был реализован следующим образом:

python computeInDisco.py --input tagName --metrics metrics.csv   
--output result-file

где computeInDisco.py – имя файла разработанного сценария; --input – ключ, определяющий следующий аргумент командной строки как общую часть имени ссылок (tag), под которыми в файловой системе DDFS размещены исходные данные; tagName – общая часть для имени группы входных файлов; --tag-name – ключ, определяющий следующий аргумент командной строки как имя ссылки (tag), под которой будут размещены загружаемые данные; tagName – общая часть имён ссылок (tag), под которыми в файловой системе DDFS размещены исходные данные; --metrics – ключ, определяющий следующий аргумент командной строки как имя файла, содержащего результаты измерения времени; metrics.csv – имя файла, содержащего результаты измерений в формате CSV: tagName – общая часть для имени группы входных файлов; --output – ключ, определяющий следующий аргумент командной строки - общую часть имени, под которым будут размещены выходные данные; result-file – общая часть имён, под которыми будут размещены выходные данные.

# Алгоритм решения задачи

Программа computeInDisco требует дополнительный параметр – префик имени выходных файлов. Ее результатом является набор выходных файлов с результатами разбиения, и файл, содержащий время обработки каждого из входных файлов.

В функции main обрабатываются параметры программы. Затем считывается сформированная генератором «порция» данных. Для каждой «порции» данных выполняются вызовы функции run из модуля Job.

job\_1 = Job().run(input=input\_file, map=build\_subline\_index, reduce=merge\_map\_sub)

job\_2 = Job().run(input=input\_file, map=build\_symbol\_index, reduce=merge\_map\_sym)

Для этого требуется подключить данный модуль:

from disco.core import Job, result\_iterator

Функция build\_subline\_index (map) - работа, которую нужно произвести с данными. merge\_map\_sub (reduce) - функция, которая объединяет результат обработки данных и сводит их к необходимому типу данных.

Каждая порция данных обрабатывается параллельно и независимо функциями build\_subline\_index и build\_symbol\_index. По окончанию обработки словари, полученные в результате работы функций, сливаются в один общий (для строкового и символьного индексов - соответственно).

# Тестирование программных средств

## Проверка корректности реализованных приложений

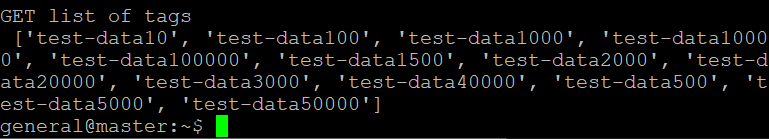


Рисунок 1 – Созданы теги в результате ЛР №4



Рисунок 2 – Запуск обработки данных

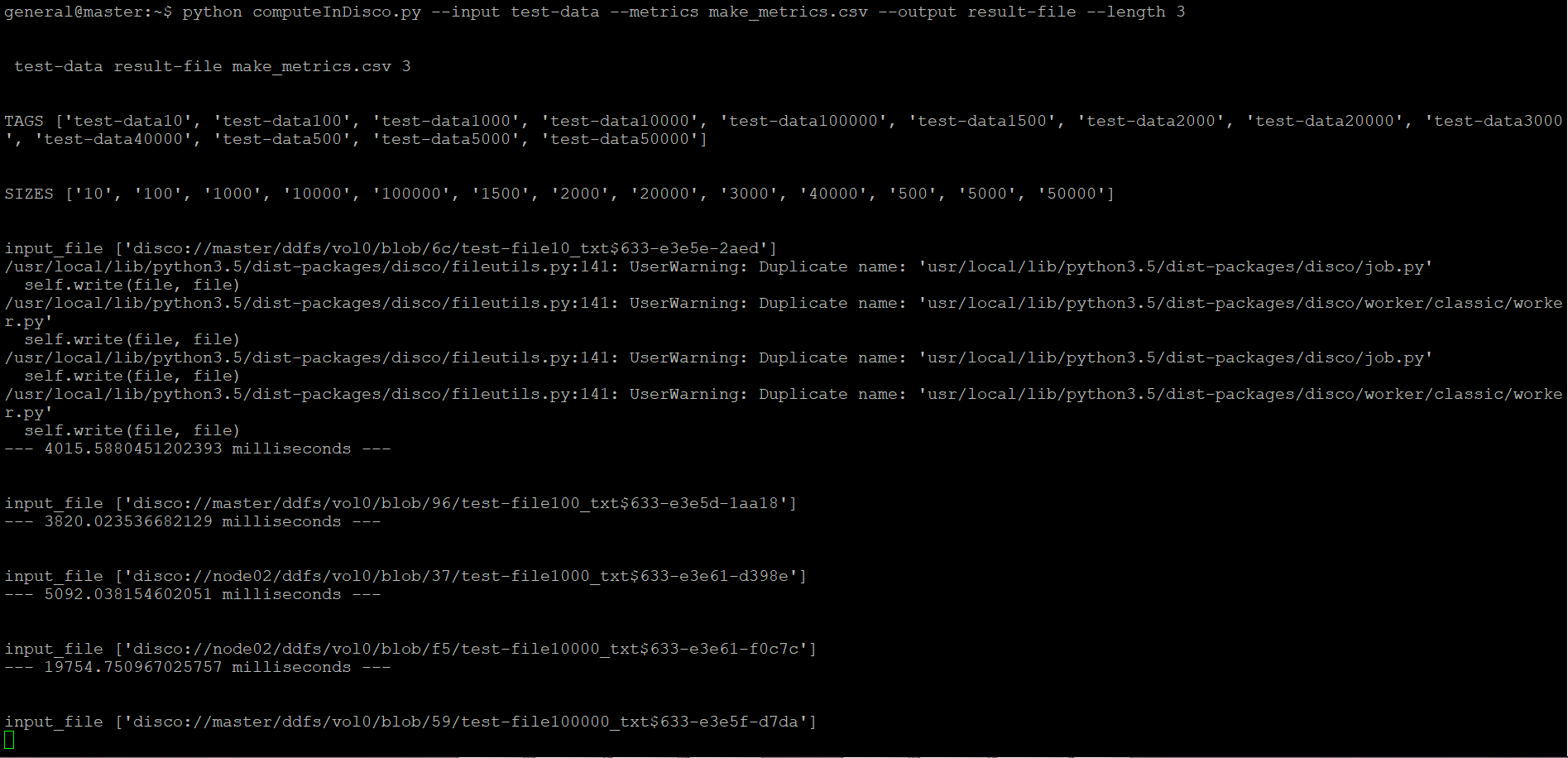


Рисунок 3 – Процесс обработки данных



Рисунок 4 – Процесс обработки данных WebAPI

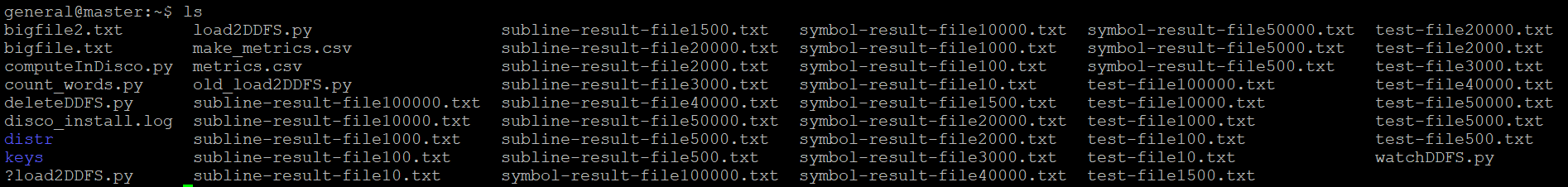


Рисунок 5 – Выходные файлы

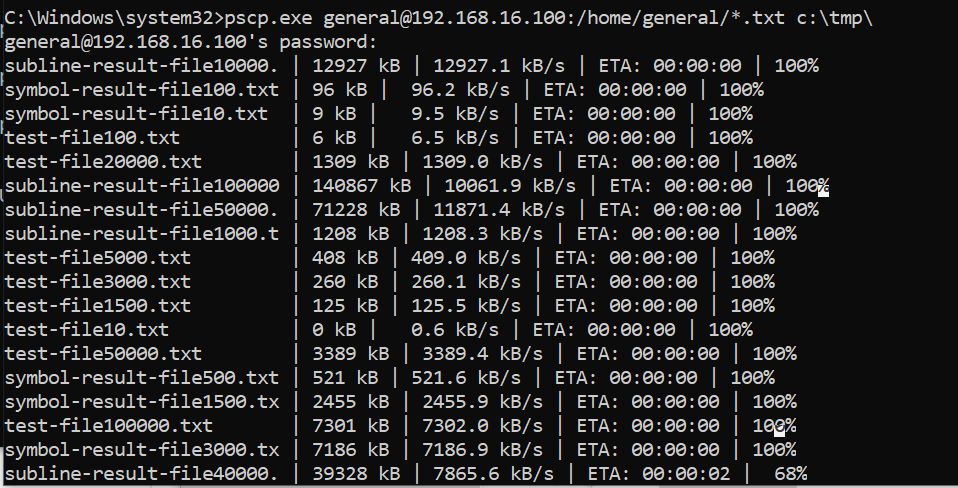


Рисунок 6 – Копирование результатов на realhost

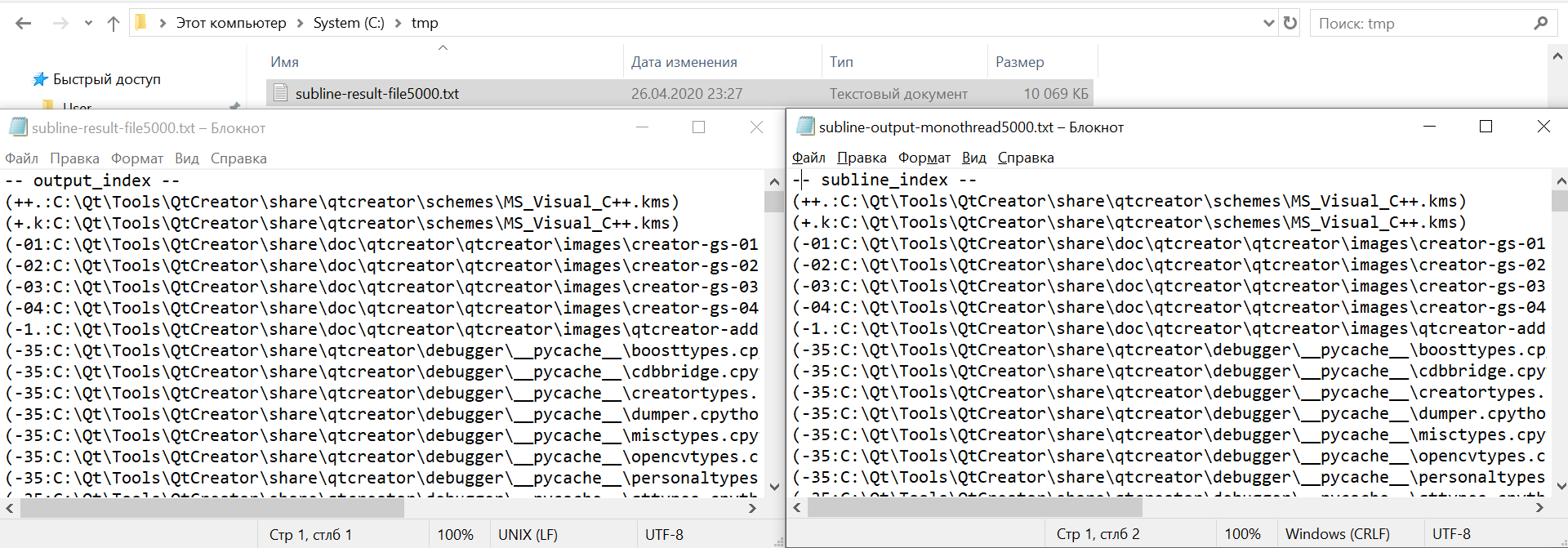


Рисунок 7 – Cравнение subline результатов disco (слева) с multithread (справа), объем 5000, начало файла

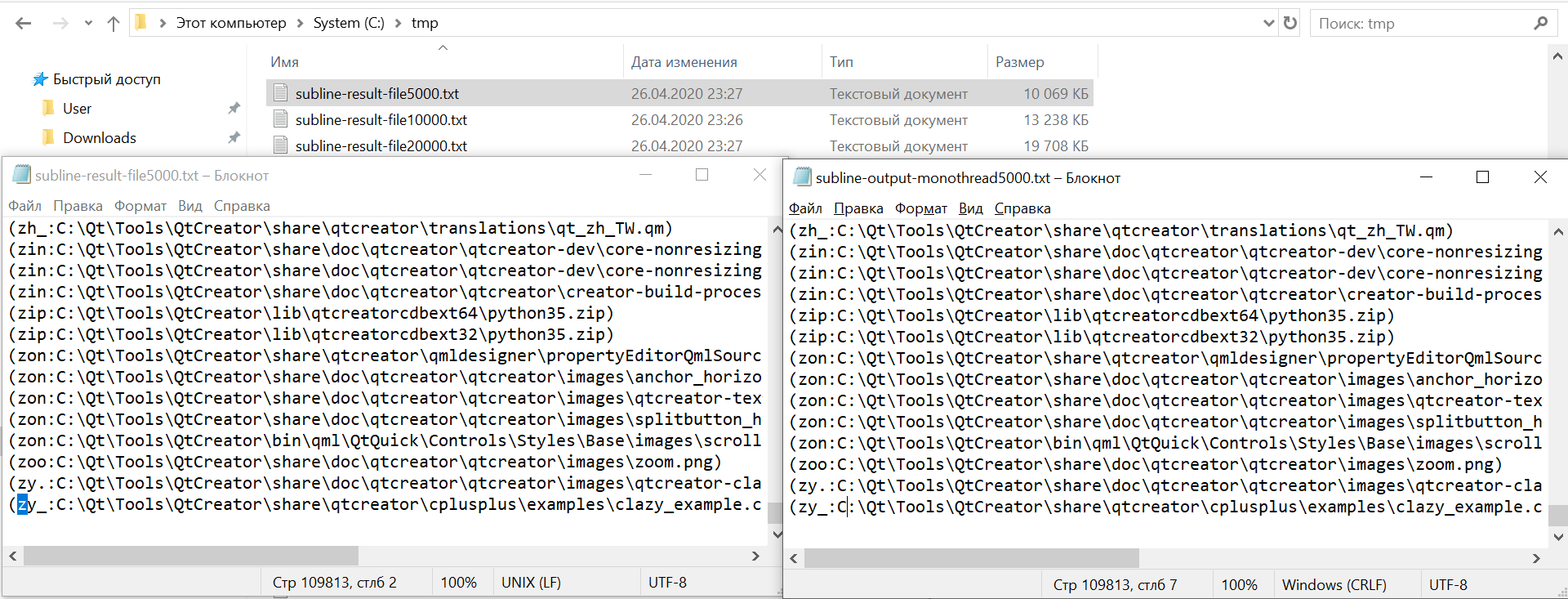


Рисунок 8 – Cравнение subline результатов disco (слева) с multithread (справа), объем 5000, конец файла

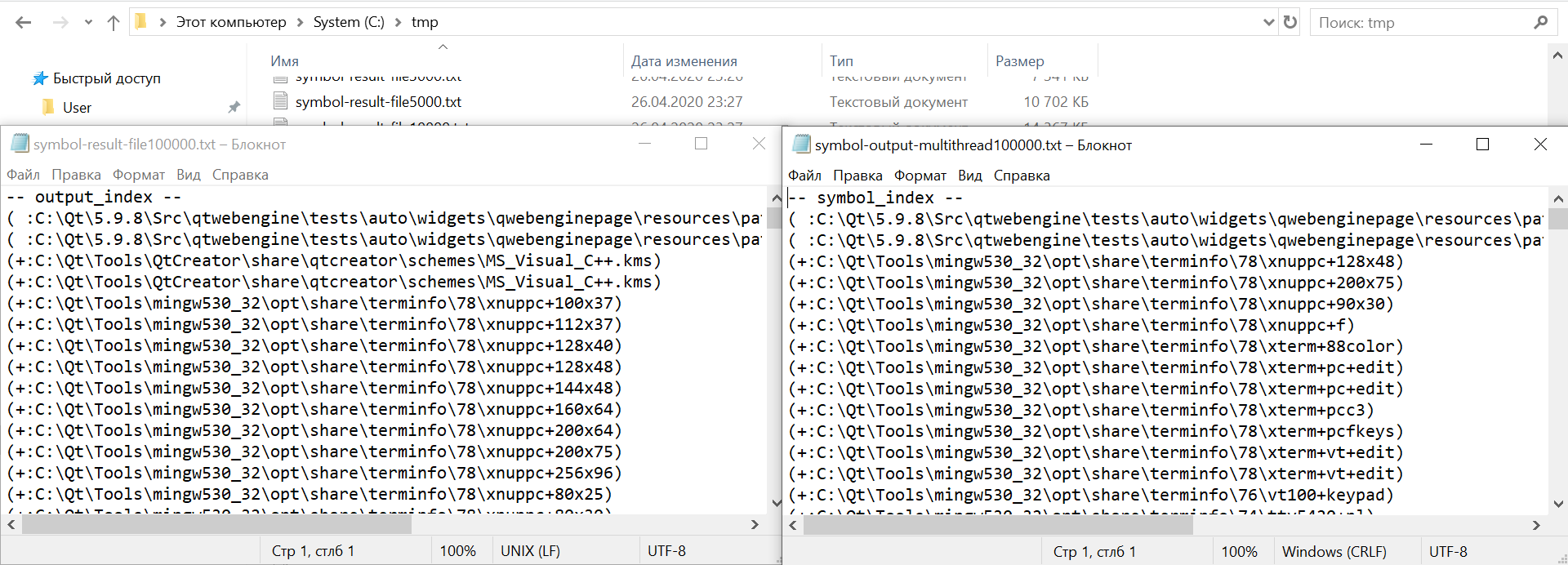


Рисунок 9 – Cравнение symbol результатов disco (слева) с multithread (справа), объем 100000, начало файла

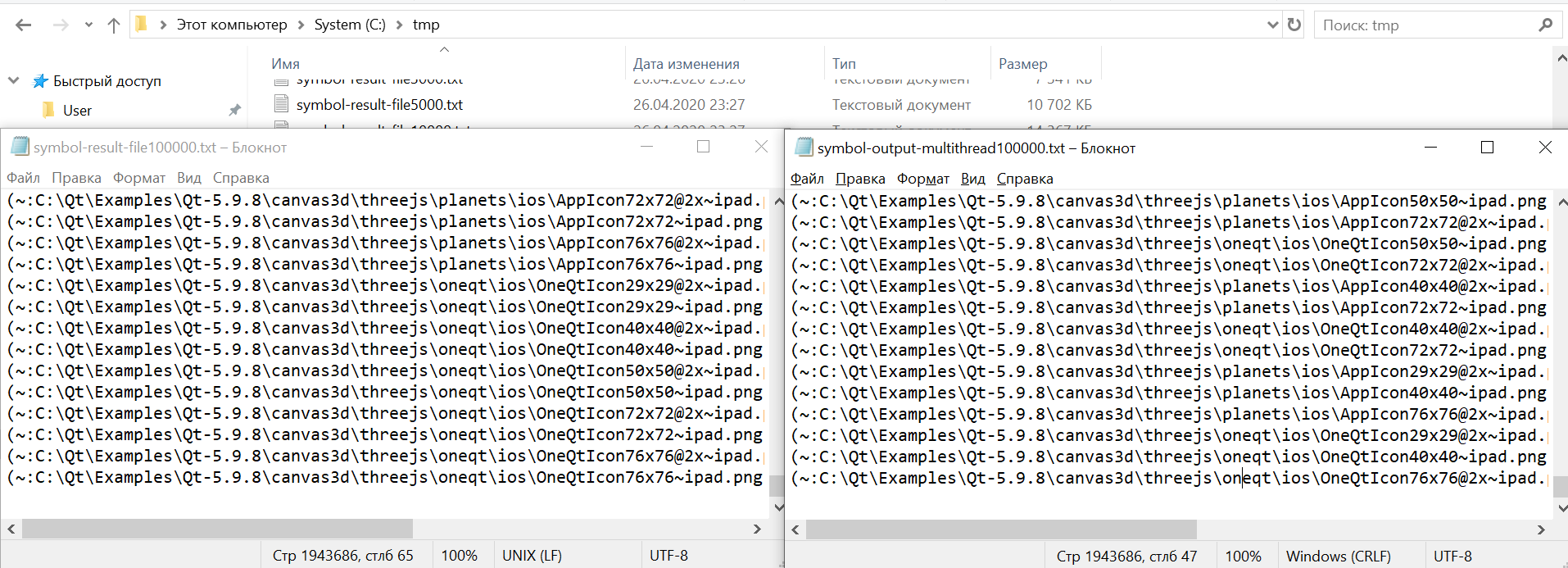


Рисунок 10 – Cравнение symbol результатов disco (слева) с multithread (справа), объем 100000, конец файла

## Проведение экспериментов

Был проведен эксперимент для большого количества данных. Для программы generator был указан путь «C:/Qt» и сгенерирован большой объем входных данных.

## Результаты экспериментов

Результат загрузки и обработки данных был представлен в таблице 1.

|  |  |  |
| --- | --- | --- |
| Объем данных | Время загрузки (мс) | Время обработки (мс) |
| 10 | 23.34451675415039 | 3513.068675994873 |
| 100 | 40.561676025390625 | 3865.5951023101807 |
| 500 | 53.67779731750488 | 4653.383016586304 |
| 1000 | 37.6434326171875 | 5284.315586090088 |
| 1500 | 36.84687614440918 | 6665.890693664551 |
| 2000 | 55.503129959106445 | 9768.404245376587 |
| 3000 | 54.38423156738281 | 10741.441249847412 |
| 5000 | 92.15497970581055 | 8186.914920806885 |
| 10000 | 221.574068069458 | 20657.04655647278 |
| 20000 | 309.751033782959 | 36123.08621406555 |
| 40000 | 711.2715244293213 | 116279.32167053223 |
| 50000 | 805.0119876861572 | 134224.3528366089 |
| 100000 | 1859.9941730499268 | 239447.43084907532 |

Таблица 1 – результат эксперимента

## Анализ полученных результатов

На графике наблюдается близкая к линейной зависимость. Временные результаты могут отличаться, в зависимости от длины подстрок разбиения и заданной директории для построения индекса.

Наблюдается большое отставание от многопоточного алгоритма. Это обусловлено временными задержками клиент серверного взаимодействия (запрос содержимого файла, ответ от сервера), различными конфигурациями оборудования (mapreduce – 4 ядра 12 Гб ОЗУ, dico – 1 ядро, 1 Гб ОЗУ на узел), особенностями виртаулизации.

Рисунок 11 – Временные результаты

Рисунок 12 – Сравнение результатов работы