**О Т Ч Е Т**

о выполнении лабораторной работы № 4

МОСКВА 2020

Оглавление

[1 Задание 3](#_Toc38878360)

[2 Теоретическая справка 4](#_Toc38878361)

[3 Спецификации разработанных программ 5](#_Toc38878362)

[3.1 Программное средство загрузки данных на Disco DDFS 5](#_Toc38878363)

[4 Алгоритм решения задачи 6](#_Toc38878364)

[5 Тестирование программных средств 7](#_Toc38878365)

[5.1 Проверка корректности реализованных приложений 7](#_Toc38878366)

[5.2 Проведение экспериментов 10](#_Toc38878367)

[5.3 Результаты экспериментов 10](#_Toc38878368)

[5.4 Анализ полученных результатов 11](#_Toc38878369)

# Задание

1. В качестве исходных данных для работы использовать файлы, формируемые генератором исходных данных, разработанным в рамках лабораторной работы № 1.

2. Разработать формат данных для размещения исходных данных в файловой системе DDFS MapReduce-кластера Disco.

3. Разработать алгоритм размещения исходных данных в файловой системе DDFS MapReduce-кластера Disco.

4. Реализовать разработанный в пункте 3 алгоритм в виде сценария на языке программирования Python с использованием API-интерфейса доступа к файловой системе DDFS.

# 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 загрузки данных на Disco DDFS

Интерфейс созданного сценария реализован следующим образом:

python load2DDFS.py --input inputFilesName --tag-name tagName –-metrics metrics.csv

где load2DDFS.py – имя файла разработанного сценария; --input – ключ, определяющий следующий аргумент командной строки как общую часть для имени группы входных файлов, обрабатываемых данных; inputFilesName – общая часть для имени группы входных файлов; --tag-name – ключ, определяющий следующий аргумент командной строки как общую часть имени ссылок (tag), под которыми будут размещены загружаемые данные; tagName – общая часть имён ссылок (tag), под которыми будут размещены загружаемые 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.

# Алгоритм решения задачи

Алгоритм решения задачи был разработан при помощи DDFS Python API.  
Для решения задачи использовались функции: disco.ddfs.DDFS("disco://localhost") – для подключения к бд:

push - для загрузки данных

list – для просмотра текущих тегов  
 Для поиска файлов с указанным префиксом использовалась библиотека re. Каждый файл в текущей директории сравнивался с маской, переданной в флаге input.

Затем формировался список объемов входных файлов при помощи регулярных выражений. Для каждого объема был создан тег с префиксом, заданным в ключе tag-name.

Далее данные были загружены в DDFS командой рush. Для каждой «порции» данных была измерена скорость их загрузки при помощи модуля time. Время загрузки было размещено в файле, указанном в параметре metrics.

# Тестирование программных средств

## Проверка корректности реализованных приложений



Рисунок 1 - Генерация данных

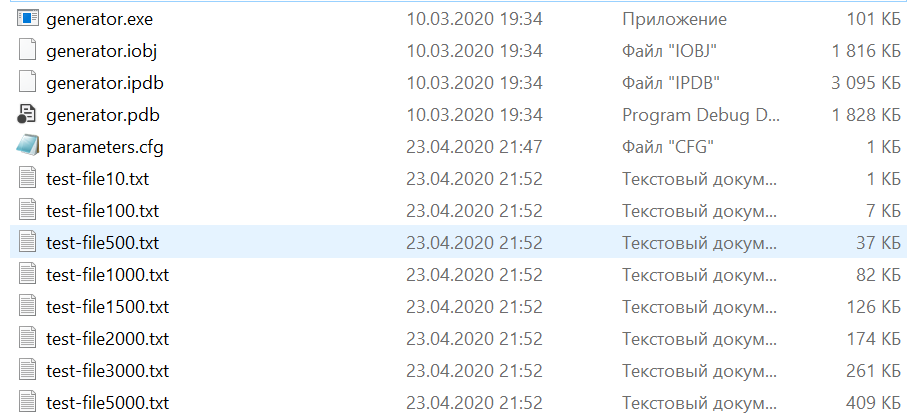


Рисунок 2 - Результат генерации данных

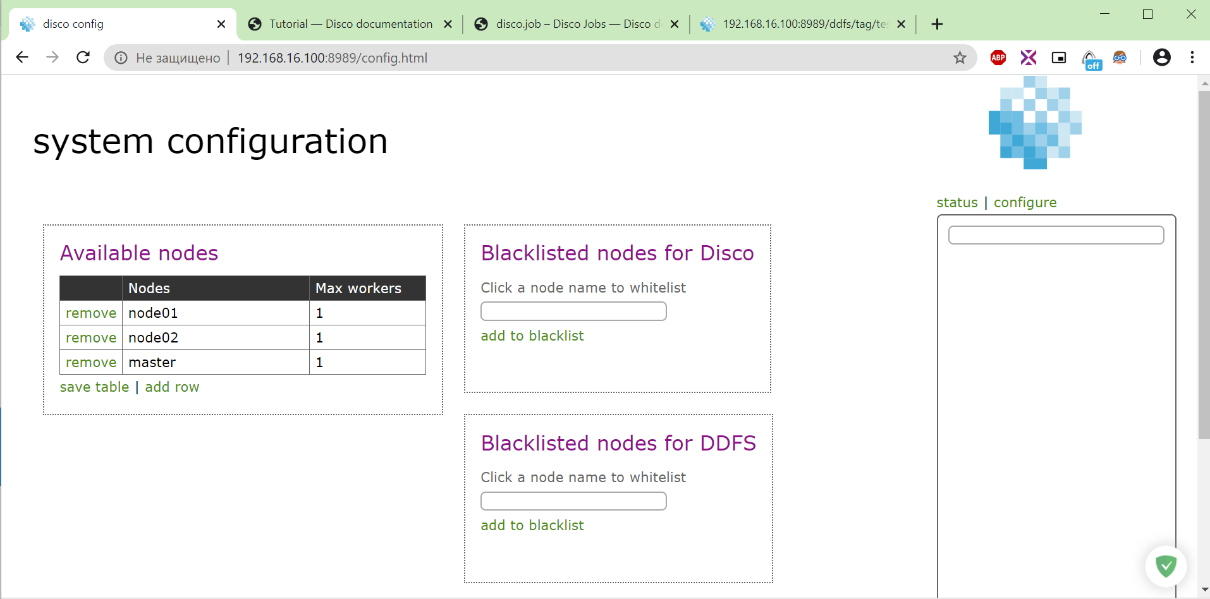
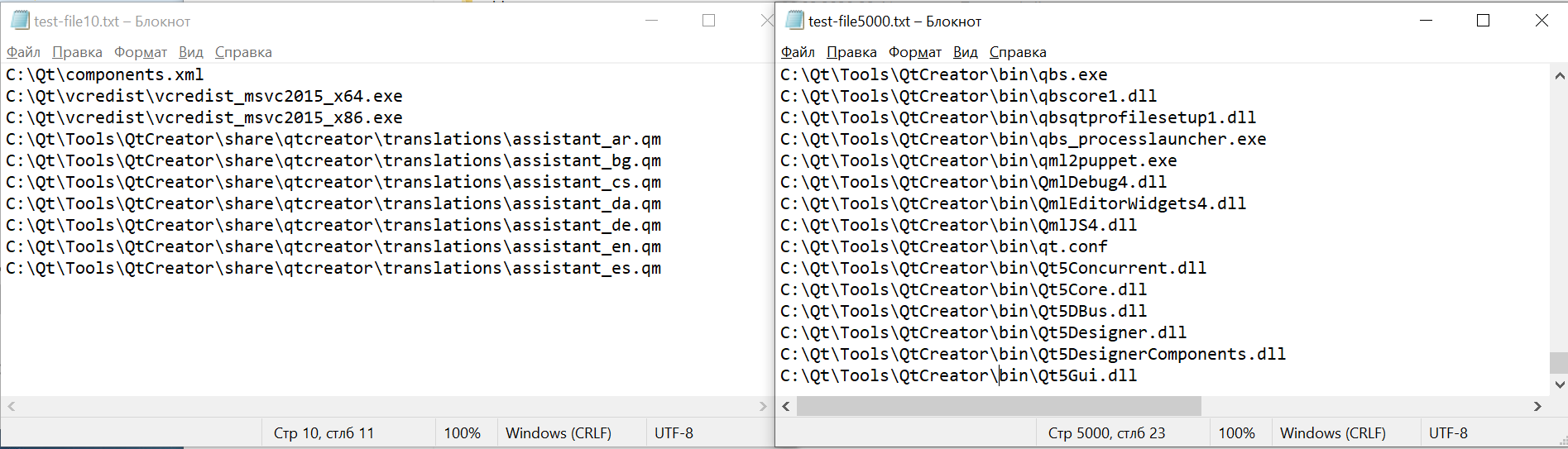


Рисунок 3 - Доступ к кластеру через Web APIРисунок 4 – Обзор данных

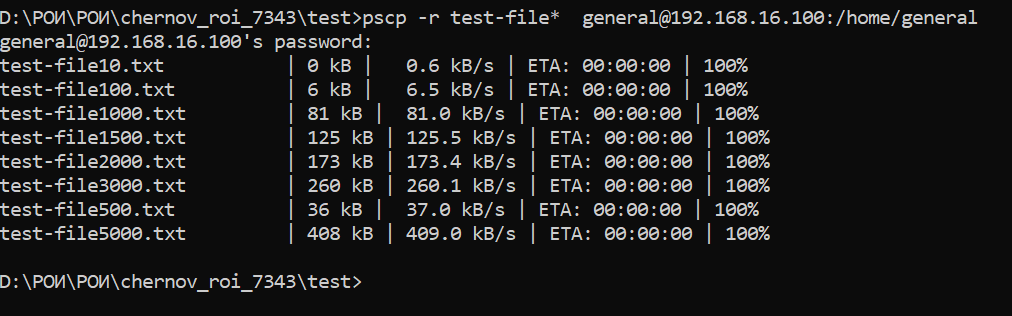
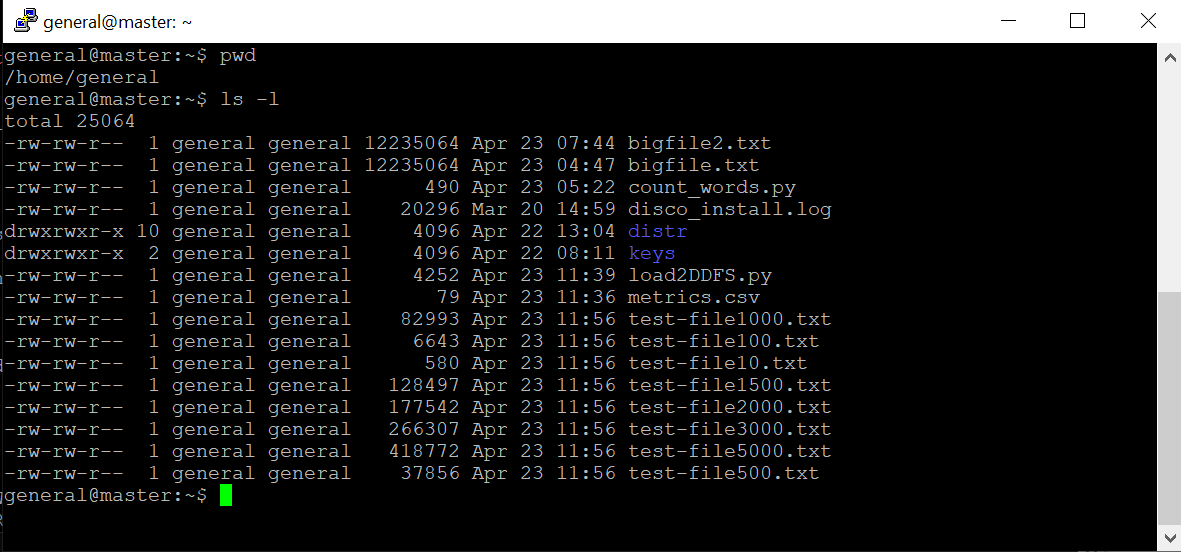
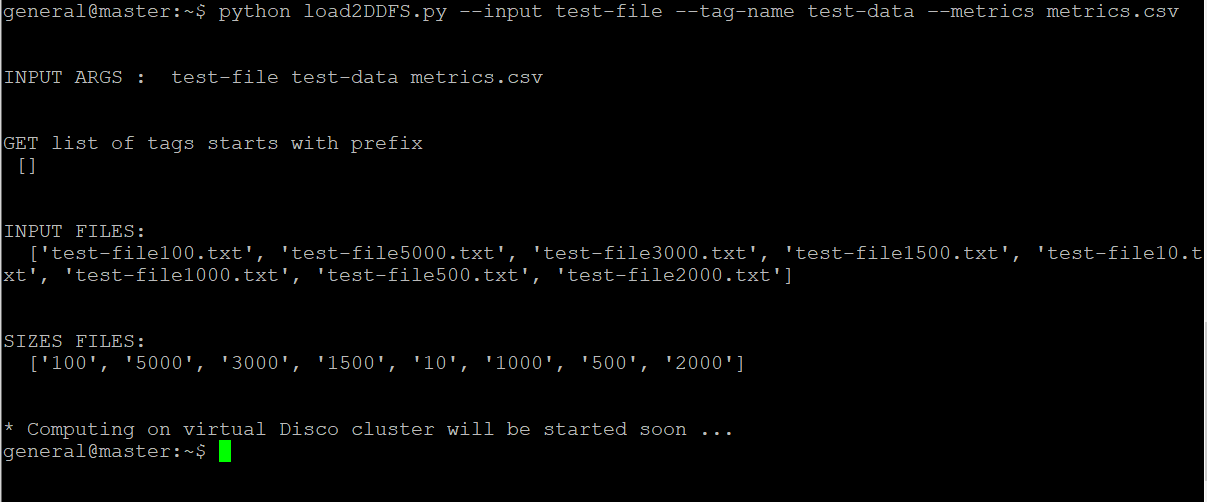
Рисунок 5 – Отправка данных на master

Рисунок 6 – Результат отправки данных на masterРисунок 7 – Результат запуска скрипта

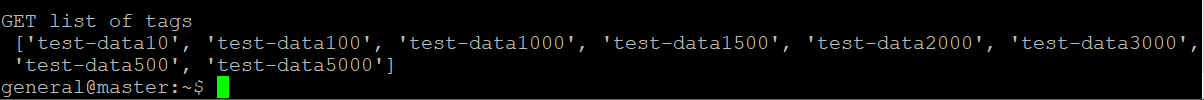


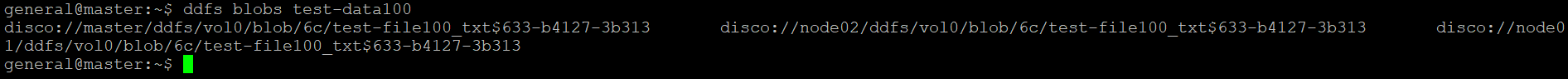
Рисунок 8 – Созданы теги после запуска скрипта

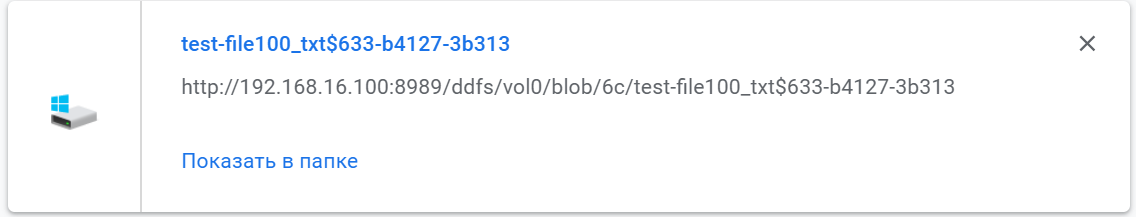
Рисунок 9 – Получение адресов BLOB файлов

Рисунок 9 – Загрузка BLOB файла по адресу

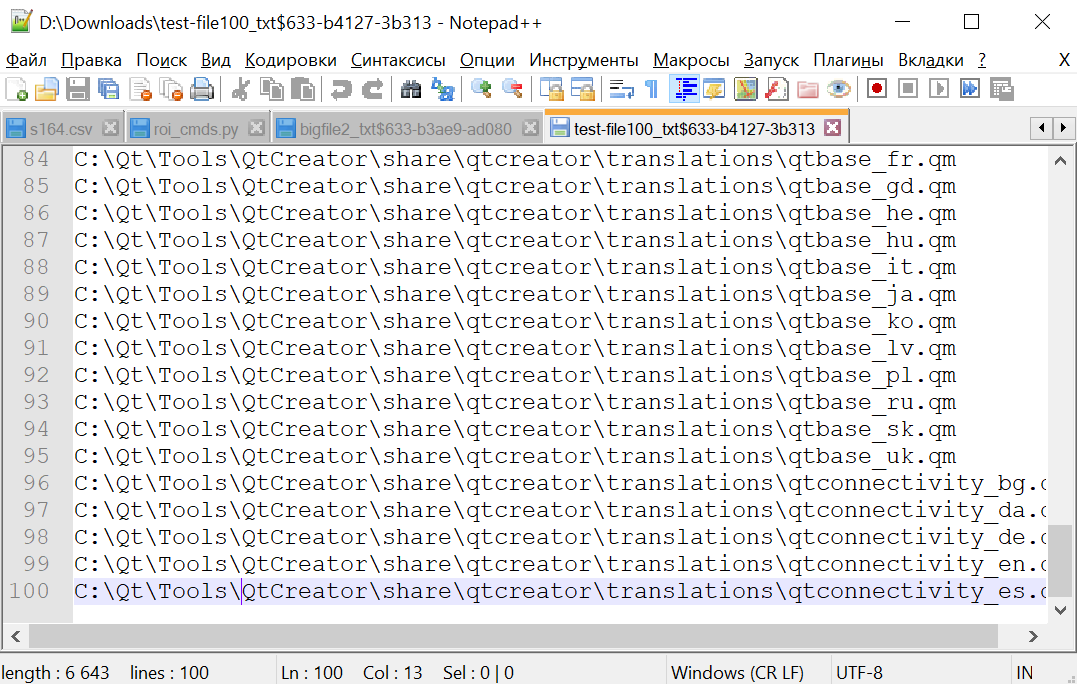


Рисунок 10 – Содержимое BLOB файла

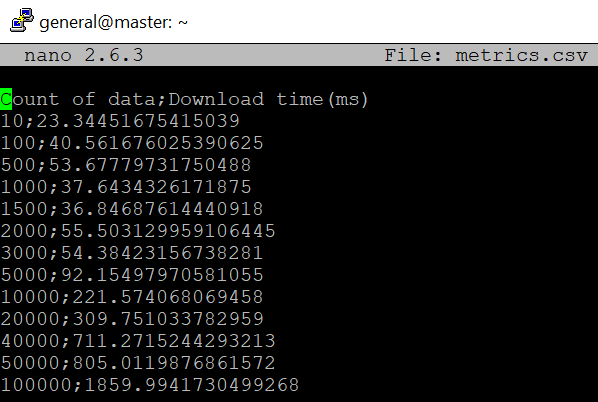


Рисунок 11 – Время загрузки каждой «порции» данных

## Проведение экспериментов

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

## Результаты экспериментов

Результат загрузки данных был представлен в таблице 1.

|  |  |
| --- | --- |
| Объем данных | Время загрузки (мс) |
| 10 | 23.34451675415039 |
| 100 | 40.561676025390625 |
| 500 | 53.67779731750488 |
| 1000 | 37.6434326171875 |
| 1500 | 36.84687614440918 |
| 2000 | 55.503129959106445 |
| 3000 | 54.38423156738281 |
| 5000 | 92.15497970581055 |
| 10000 | 221.574068069458 |
| 20000 | 309.751033782959 |
| 40000 | 711.2715244293213 |
| 50000 | 805.0119876861572 |
| 100000 | 1859.9941730499268 |

Таблица 1 – результат эксперимента

## Анализ полученных результатов

В ходе лабораторной работы было реализовано средство загрузки данных. На скорость загрузки размер файлов и количество ядер и ОЗУ виртуальной машины. Были получены ожидаемые результаты – при посте объема файлов растет время загрузки.

Рисунок 12 – Время загрузки данных