**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**Кафедра Информатики**



**Отчет по лабораторной работе №8**

по предмету «КТП»

Выполнил: студент группы БВТ1802

Ткачев Александр Владимирович

Руководитель:

Ксения Андреевна Полянцева

Москва 2020

**1 Цель работы**

Цель работы: изучить работу простейшего веб-сканера.

**2 Задание**

Ускорить работу web-сканера, распределив чтение ссылок в несколько потоков. В командной строке добавляется третий аргумент – количество потоков.

**3 Текст программы**

**Class ScannerApp**

public class ScannerApp {  
 public static void main(String args[]) throws IOException, InterruptedException {  
 URLPool pool = new URLPool(args[0], Integer.*parseInt*(args[1]), Integer.*parseInt*(args[2]));  
 for (int i = 0; i < Integer.*parseInt*(args[2]); i++) {  
 CrawlerTask crawler = new CrawlerTask(pool);  
 new Thread(crawler).start();  
 System.*out*.println("Crawler " + i + " running");  
 }  
 }  
}

**Class URLPool**

package com.company;  
  
import java.util.LinkedList;  
  
public class URLPool {  
  
 private LinkedList<URLDepthPair> m\_Processed = new LinkedList<URLDepthPair>();  
 private LinkedList<URLDepthPair> m\_NotProcessed = new LinkedList<URLDepthPair>();  
 private int m\_Depth;  
 private int m\_Waiting;  
 private int m\_Threads;  
  
 public URLPool(String url, int depth, int threads) {  
 m\_NotProcessed.add(new URLDepthPair(url, depth));  
 m\_Depth = depth;  
 m\_Threads = threads;  
 }  
  
 public synchronized URLDepthPair get() throws InterruptedException {  
 if (isEmpty()) {  
 m\_Waiting++;  
 if (m\_Waiting == m\_Threads) {  
 getSites();  
 System.*exit*(0);  
 }  
 wait();  
 }  
 return m\_NotProcessed.removeFirst();  
 }  
 public synchronized void addNotProcessed(URLDepthPair pair) {  
 m\_NotProcessed.add(pair);  
 if (m\_Waiting > 0) {  
 m\_Waiting--;  
 notify();  
 }  
 }  
  
 private boolean isEmpty() {  
 if (m\_NotProcessed.size() == 0) return true;  
 return false;  
 }  
  
 public void getSites() {  
 System.*out*.println("Depth: " + m\_Depth);  
 for (int i = 0; i < m\_Processed.size(); i++) {  
 System.*out*.println( m\_Depth - m\_Processed.get(i).getDepth() + " " + m\_Processed.get(i).getURL());  
 }  
 System.*out*.println("Links visited: " + m\_Processed.size());  
 }  
  
  
 public void addProcessed(URLDepthPair pair) {  
 m\_Processed.add(pair);  
 }  
  
 public LinkedList<URLDepthPair> getProcessed()  
 {  
 return m\_Processed;  
 }  
  
 public LinkedList<URLDepthPair> getNotProcessed()  
 {  
 return m\_NotProcessed;  
 }  
  
}

**Class URLDepthPair**

package com.company;  
  
import java.util.Objects;  
  
public class URLDepthPair {  
  
 private String m\_Url;  
 private int m\_Depth;  
  
 public URLDepthPair(String host, int depth) {  
 m\_Url = host;  
 m\_Depth = depth;  
 }  
  
 public String getURL() {  
 return m\_Url;  
 }  
  
 public int getDepth() {  
 return m\_Depth;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (obj instanceof URLDepthPair) {  
 URLDepthPair o = (URLDepthPair)obj;  
 return this.m\_Url.equals(o.getURL());  
 }  
 return false;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*();  
 }  
}

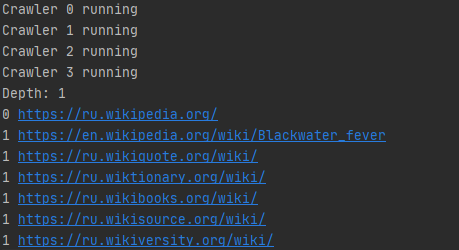
**Class CrawlerTask**

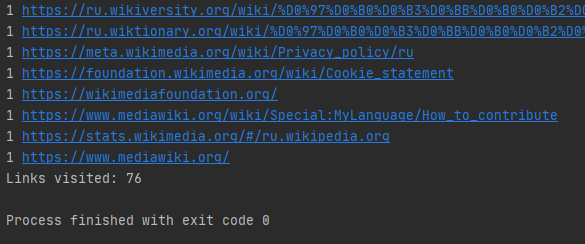
public class CrawlerTask implements Runnable {  
  
 final static int *AnyDepth* = 0;  
  
 private URLPool m\_Pool;  
 // у префикса нету слэша для поддержки https  
 private String m\_Prefix = "http";  
  
 @Override  
 public void run() {  
 try {  
 Scan();  
 } catch (IOException | InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public CrawlerTask(URLPool pool) {  
 m\_Pool = pool;  
 }  
  
 private void Scan() throws IOException, InterruptedException {  
 while (true) {  
 Process(m\_Pool.get());  
  
 }  
 }  
  
 private void Process(URLDepthPair pair) throws IOException{  
 // устанавливает соединение и следует перенаправлению  
 URL url = new URL(pair.getURL());  
 URLConnection connection = url.openConnection();  
  
 String redirect = connection.getHeaderField("Location");  
 if (redirect != null) {  
 connection = new URL(redirect).openConnection();  
 }  
  
 m\_Pool.addProcessed(pair);  
 if (pair.getDepth() == 0) return;  
  
 BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
 String input;  
 while ((input = reader.readLine()) != null) {  
 while (input.contains("a href=\"" + m\_Prefix)) {  
 input = input.substring(input.indexOf("a href=\"" + m\_Prefix) + 8);  
 String link = input.substring(0, input.indexOf('\"'));  
 if(link.contains(" "))  
 link = link.replace(" ", "%20");  
 // избегает использование одной и той же ссылки  
 if (m\_Pool.getNotProcessed().contains(new URLDepthPair(link, *AnyDepth*)) ||  
 m\_Pool.getProcessed().contains(new URLDepthPair(link, *AnyDepth*))) continue;  
 m\_Pool.addNotProcessed(new URLDepthPair(link, pair.getDepth() - 1));  
 }  
 }  
 reader.close();  
  
 }  
  
}

**4 Работа программы**

После добавления потоков поиск стал производиться на порядок быстрее.

Стартовый url https://ru.wikipedia.org/ - Глубина поиска – 1 Количество потоков: 4





Стартовый url <https://ru.wikipedia.org/> - Глубина поиска – 2 Количество потоков: 4

