МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Систем и Технологий

Кафедра «Информационные системы»

Дисциплина «Распределенные вычисления и приложения»

**ОТЧЕТ**

**по лаборатороной работе № 3**

Тема Разработка распределенного приложения с использованием фреймворка Spring Boot.Необходимо разработать параллельный вариант алгоритма с применением сервис-ориентированного подхода и фреймворка Spring Boot, замерить время его работы.

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_Кузнецов В. И.\_\_\_\_\_\_\_/

подпись инициалы, фамилия

Курс\_\_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа\_\_\_\_\_\_\_ПИбд-41\_\_\_\_\_\_\_\_\_\_

Направление/специальность\_\_\_09.03.04. Программная инженерия\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель Андреев И.А.

должность, ученая степень, ученое звание

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

Дата сдачи:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_\_г.

Дата защиты:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_\_г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ульяновск

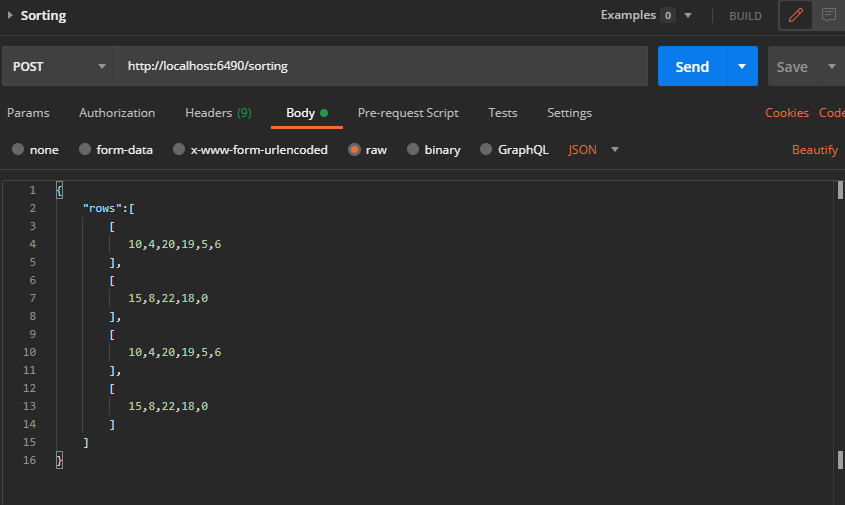
2020 г.

Задание:

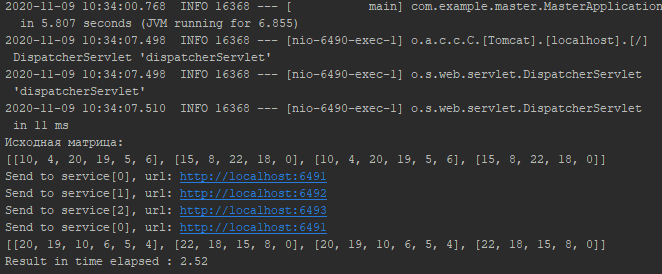
Разработка распределенного приложения с использованием фреймворка Spring Boot.Необходимо разработать параллельный вариант алгоритма с применением сервис-ориентированного подхода и фреймворка Spring Boot, замерить время его работы.

Вариант: отсортировать строки матрицы по убыванию наибольших элементов.

*Входные данные:*

**

*Результат:*

**



Принцип работы такой:

Имеет сервис Master, который принимает запрос и распределяет задания на сервисы исполнители Executor, запущенные на разных портах. Задания отправляются по Http в виде строки матрицы, которую надо отсортировать. Отправка запросов на сервисы исполнители выполняется асинхронно, ответы от сервисов исполнителей, Mater ожидает с помощью ExecutorService и массива объектов Future.

Листинг:

package com.example.master;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.Collections;

@SpringBootApplication

public class MasterApplication {

public static void main(String[] args) {

SpringApplication app = new SpringApplication(MasterApplication.class);

app.run(args);

}

}

package com.example.master.services;

import com.example.master.models.SortingViewModel;

import java.util.\*;

import java.util.concurrent.\*;

public class MainService {

private String[] services = new String[]{

"http://localhost:6491",

"http://localhost:6492",

"http://localhost:6493"

};

public SortingViewModel Sorting(SortingViewModel viewModel) {

System.out.println("Исходная матрица: ");

System.out.println(Arrays.deepToString(viewModel.getRows()));

List<Future<SortingViewModel>> futures = new ArrayList<>();

long start = System.currentTimeMillis();

ExecutorService executor = Executors.newFixedThreadPool(viewModel.getRows().length);

for (int i = 0; i < viewModel.getRows().length; i++) {

int service = i % services.length;

System.out.println("Send to service[" + service + "], url: " + services[service]);

Callable<SortingViewModel> callable = new CallbleServiceRequest(services[service], i, viewModel);

Future<SortingViewModel> future = executor.submit(callable);

futures.add(future);

}

for (Future<SortingViewModel> future : futures) {

try {

SortingViewModel result = future.get();

} catch (InterruptedException | ExecutionException e) {

System.err.println("task interrupted");

}

}

long finish = System.currentTimeMillis();

float timeElapsed = (finish - start) / 100F;

executor.shutdown();

StringBuilder builder = new StringBuilder();

builder.append("Result in time elapsed : ");

builder.append(timeElapsed);

System.out.println(Arrays.deepToString(viewModel.getRows()));

System.out.println(builder.toString());

return viewModel;

}

}

package com.example.master.services;

import com.example.master.models.SortingViewModel;

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.web.client.RestTemplate;

import java.util.concurrent.Callable;

public class CallbleServiceRequest implements Callable<SortingViewModel> {

private String url;

private int indexRow;

private SortingViewModel resource;

public CallbleServiceRequest(String url, int indexRow, SortingViewModel resource) {

this.url = url + "/sorting";

this.indexRow = indexRow;

this.resource = resource;

}

@Override

public SortingViewModel call() throws Exception {

RestTemplate restTemplate = new RestTemplate();

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

SortingViewModel requestJsonData = new SortingViewModel();

requestJsonData.setRows(new int[][]{

this.resource.getRows()[this.indexRow]

});

HttpEntity<SortingViewModel> request = new HttpEntity<>(requestJsonData);

ResponseEntity<SortingViewModel> response = restTemplate.postForEntity(url, request, SortingViewModel.class);

int[] updateRow = response.getBody().getRows()[0];

this.resource.updateRow(this.indexRow, updateRow);

return this.resource;

}

}

package com.example.master.controllers;

import com.example.master.models.SortingViewModel;

import com.example.master.services.MainService;

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.\*;

import java.util.Arrays;

@RestController

public class MainController {

private MainService mainService = new MainService();

@RequestMapping(value = "/ping", method = RequestMethod.GET)

public String pingPong() {

return "pong";

}

@PostMapping(value = "/sorting", produces = MediaType.APPLICATION\_JSON\_VALUE)

@ResponseBody

public SortingViewModel sorting(@RequestBody SortingViewModel viewModel) {

SortingViewModel response = mainService.Sorting(viewModel);

return response;

}

}

package com.example.executor;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.Collections;

@SpringBootApplication

public class ExecutorApplication {

public static void main(String[] args) {

SpringApplication app = new SpringApplication(ExecutorApplication.class);

app.run(args);

}

}

package com.example.executor.services;

public class SortingService {

private int[][] array;

public String SimpleSort(int[][] matrix) {

this.array = matrix;

//

long start = System.currentTimeMillis();

for (int index = 0; index < array.length; index++) {

array[index] = mergeSort(array[index]);

}

long finish = System.currentTimeMillis();

float timeElapsed = (finish - start) / 100F;

StringBuilder builder = new StringBuilder();

builder.append(toString());

builder.append("Result in time elapsed : ");

builder.append(timeElapsed);

return builder.toString();

}

private int[] mergeSort(int[] input) {

if (input.length <= 1) {

return input;

}

//

int middle = input.length / 2;

int[] left = new int[middle];

int[] right = new int[input.length - middle];

//

for (int i = 0; i < middle; i++) {

left[i] = input[i];

}

for (int i = middle; i < input.length; i++) {

right[i - middle] = input[i];

}

//

return merge(mergeSort(right), mergeSort(left));

}

private int[] merge(int[] right, int[] left) {

//

int size = right.length + left.length;

int i = right.length - 1;

int j = left.length - 1;

int[] sorted = new int[size];

//

for (int k = size - 1; k >= 0; k--) {

if (i == -1) {

sorted[k] = left[j];

j--;

continue;

}

if (j == -1) {

sorted[k] = right[i];

i--;

continue;

}

if (right[i] > left[j]) {

sorted[k] = left[j];

j--;

continue;

}

if (right[i] < left[j]) {

sorted[k] = right[i];

i--;

continue;

}

if (right[i] == left[j]) {

sorted[k] = left[j];

j--;

}

}

return sorted;

}

@Override

public String toString() {

StringBuilder stringBuilder = new StringBuilder();

for (int[] row : this.array) {

for (int element : row) {

stringBuilder.append(element);

stringBuilder.append(", ");

}

stringBuilder.append("\n");

}

return stringBuilder.toString();

}

}

package com.example.executor.controllers;

import com.example.executor.models.SortingViewModel;

import com.example.executor.services.SortingService;

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.\*;

import java.lang.reflect.Array;

import java.util.Arrays;

@RestController

public class MainController {

private SortingService sortingService = new SortingService();

@RequestMapping(value = "/ping", method = RequestMethod.GET)

public String ping() {

return "pong";

}

@PostMapping(value = "/sorting", produces = MediaType.APPLICATION\_JSON\_VALUE)

@ResponseBody

public SortingViewModel sorting(@RequestBody SortingViewModel viewModel) {

System.out.println(Arrays.deepToString(viewModel.getRows()));

sortingService.SimpleSort(viewModel.getRows());

return viewModel;

}

}