Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

Паралельне програмування в мові Java. Монітори

Виконав:

студент групи ІП-22

Сочка О. О.

Перевірив:

Корочкін О. В.

Київ - 2015

**Технічне завдання**



**Етап 1. Побудова паралельного алгоритму**

\* - сортування злиттям

**Спільні ресурси:** q, a, B, MO

**Етап 2. Розробка алгоритмів процесів (задач)**

|  |  |  |
| --- | --- | --- |
| **№** | **Т1** | **ТС/КУ** |
| 1 | Ввід |  |
| 2 | Сигнал іншим задачам про завершення вводу |  |
| 3 | Очікування вводу |  |
| 4 | Очікування вводу |  |
| 5 | Копія a, B, MO | КУ |
| 6 | Підрахунок |  |
| 7 | Підрахунок | КУ |
| 8 | Копія q | КУ |
| 9 | Сигнал іншим потокам про підрахунок q |  |
| 10 | Очікування підрахунку q в інших потоках |  |
| 11 | Підрахунок |  |
| 12 | Очікування завершення підрахунку в інших потоках |  |
| 13 | Вивід |  |

|  |  |  |
| --- | --- | --- |
| **№** | **Т2** | **ТС/КУ** |
| 1 | Очікування вводу |  |
| 2 | Очікування вводу |  |
| 3 | Очікування вводу |  |
| 4 | Копія a, B, MO | КУ |
| 5 | Підрахунок |  |
| 6 | Підрахунок | КУ |
| 7 | Копія q | КУ |
| 8 | Сигнал іншим потокам про підрахунок q |  |
| 9 | Очікування підрахунку q в інших потоках |  |
| 10 | Підрахунок |  |
| 11 | Сигнал потоку T1 про завершення підрахунку |  |

|  |  |  |
| --- | --- | --- |
| **№** | **Т3** | **ТС/КУ** |
| 1 | Очікування вводу |  |
| 2 | Очікування вводу |  |
| 3 | Очікування вводу |  |
| 4 | Копія a, B, MO | КУ |
| 5 | Підрахунок |  |
| 6 | Підрахунок |  |
| 7 | Копія q | КУ |
| 8 | Сигнал іншим потокам про підрахунок q |  |
| 9 | Очікування підрахунку q в інших потоках |  |
| 10 | Підрахунок |  |
| 11 | Сигнал потоку T1 про завершення підрахунку |  |

|  |  |  |
| --- | --- | --- |
| **№** | **Т4** | **ТС/КУ** |
| 1 | Ввід |  |
| 2 | Сигнал іншим задачам про завершення вводу |  |
| 3 | Очікування вводу |  |
| 4 | Очікування вводу |  |
| 5 | Копія a, B, MO | КУ |
| 6 | Підрахунок |  |
| 7 | Підрахунок | КУ |
| 8 | Копія q | КУ |
| 9 | Сигнал іншим потокам про підрахунок q |  |
| 10 | Очікування підрахунку q в інших потоках |  |
| 11 | Підрахунок |  |
| 12 | Сигнал потоку T1 про завершення підрахунку |  |

|  |  |  |
| --- | --- | --- |
| **№** | **Т5** | **ТС/КУ** |
| 1 | Очікування вводу |  |
| 2 | Очікування вводу |  |
| 3 | Очікування вводу |  |
| 4 | Копія a, B, MO | КУ |
| 5 | Підрахунок |  |
| 6 | Підрахунок | КУ |
| 7 | Копія q | КУ |
| 8 | Сигнал іншим потокам про підрахунок q |  |
| 9 | Очікування підрахунку q в інших потоках |  |
| 10 | Підрахунок |  |
| 11 | Сигнал потоку T1 про завершення підрахунку |  |

|  |  |  |
| --- | --- | --- |
| **№** | **Т6** | **ТС/КУ** |
| 1 | Ввід |  |
| 2 | Сигнал іншим задачам про завершення вводу |  |
| 3 | Очікування вводу |  |
| 4 | Очікування вводу |  |
| 5 | Копія a, B, MO | КУ |
| 6 | Підрахунок |  |
| 7 | Підрахунок | КУ |
| 8 | Копія q | КУ |
| 9 | Сигнал іншим потокам про підрахунок q |  |
| 10 | Очікування підрахунку q в інших потоках |  |
| 11 | Підрахунок |  |
| 12 | Сигнал потоку T1 про завершення підрахунку |  |

**Етап 3. Розробка схеми взаємодії задач**



**Етап 4. Розробка програми**

**1 package com.sochka;**

**2**

**3 import java.util.Arrays;**

**4**

**5 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**6 \* Лабораторна робота №5 з дисципліни ПП-2**

**7 \* Lab5. Java. Монітори**

**8 \* ПІБ Сочка Олександр Олександрович**

**9 \* Група ІП-22**

**10 \* Дата 10.04.2015**

**11 \* MA = a\*B\*(MO\*MX) + (Z\*E) \* R**

**12 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**13**

**14 public class Main {**

**15 // Processor setup**

**16 public static final int N = 6;**

**17 public static final int H = N / 6;**

**18**

**19**

**20 // Input and output data**

**21 private static int[] Z, E, R;**

**22 private static int[] A = new int[N];**

**23 private static int[][] MX;**

**24**

**25 // Monitor**

**26 private static final Monitor monitor = new Monitor();**

**27**

**28 static class IO {**

**29 static int readInt() {**

**30 return 1;**

**31 }**

**32**

**33 static int[] readVector() {**

**34 int[] result = new int[N];**

**35 Arrays.fill(result, 1);**

**36 return result;**

**37 }**

**38**

**39 static int[][] readMatrix() {**

**40 int[][] result = new int[N][];**

**41 for (int i = 0; i < result.length; i++) {**

**42 result[i] = new int[N];**

**43 Arrays.fill(result[i], 1);**

**44 }**

**45 return result;**

**46 }**

**47**

**48 static synchronized void output(int[] A) {**

**49 for (int i : A) {**

**50 System.out.print(i);**

**51 System.out.print(' ');**

**52 }**

**53 }**

**54 }**

**55**

**56 static class Monitor {**

**57 private int cntInputs = 0;**

**58 private int cntCalc1 = 0;**

**59 private int cntCalc2 = 0;**

**60**

**61 private int a, q = 0;**

**62 private int[] B;**

**63 private int[][] MO;**

**64**

**65 public synchronized void write\_a(int a) {**

**66 this.a = a;**

**67 }**

**68 public synchronized void write\_B(int[] B) {**

**69 this.B = B;**

**70 }**

**71**

**72 public synchronized void write\_MO(int[][] MO) {**

**73 this.MO = MO;**

**74 }**

**75**

**76 public synchronized void increase\_q(int add) {**

**77 this.q += add;**

**78 }**

**79**

**80 public synchronized void copy\_a\_B\_MO(Task task) {**

**81 task.a = a;**

**82 task.B = B;**

**83 task.MO = MO;**

**84 }**

**85**

**86 public synchronized void copy\_q(Task task) {**

**87 task.q\_cpy = q;**

**88 }**

**89**

**90**

**91 public synchronized void waitInput() {**

**92 while (cntInputs != 6) {**

**93 try {**

**94 wait();**

**95 } catch (InterruptedException e) {**

**96 e.printStackTrace();**

**97 }**

**98 }**

**99 }**

**100**

**101 public synchronized void waitCalc1() {**

**102 while (cntCalc1 != 6) {**

**103 try {**

**104 wait();**

**105 } catch (InterruptedException e) {**

**106 e.printStackTrace();**

**107 }**

**108 }**

**109 }**

**110**

**111 public synchronized void waitCalc2() {**

**112 while (cntCalc2 != 6) {**

**113 try {**

**114 wait();**

**115 } catch (InterruptedException e) {**

**116 e.printStackTrace();**

**117 }**

**118 }**

**119 }**

**120**

**121 public synchronized void signalInput() {**

**122 cntInputs++;**

**123 notifyAll();**

**124 }**

**125**

**126 public synchronized void signalCalc1() {**

**127 cntCalc1++;**

**128 notifyAll();**

**129 }**

**130**

**131 public synchronized void signalCalc2() {**

**132 cntCalc2++;**

**133 notifyAll();**

**134 }**

**135 }**

**136**

**137 static class Task implements Runnable {**

**138 protected Monitor monitor;**

**139 protected int t;**

**140 protected int first, last;**

**141**

**142 int a, q\_cpy;**

**143 int[] B;**

**144 int[][] MO;**

**145 Task(Monitor monitor, int t) {**

**146 this.monitor = monitor;**

**147 this.t = t;**

**148 first = H \* t;**

**149 last = first + H - 1;**

**150 }**

**151**

**152 protected void input() {}**

**153**

**154 protected void calc1() {**

**155 int tmp\_q = 0;**

**156 for (int i = first; i <= last; ++i) {**

**157 tmp\_q += Z[i] \* E[i];**

**158 }**

**159 monitor.increase\_q(tmp\_q);**

**160 }**

**161**

**162 protected void calc2() {**

**163 // Calc A**

**164 for (int j = first; j <= last; ++j) {**

**165 for (int i = 0; i < N; ++i) {**

**166 for (int k = 0; k < N; ++k) {**

**167 A[j] += a \* B[i] \* MO[i][k] \* MX[k][j];**

**168 }**

**169 }**

**170 A[j] += q\_cpy \* R[j];**

**171 }**

**172 }**

**173**

**174 protected void output() {}**

**175**

**176 @Override**

**177 public void run () {**

**178 input();**

**179 monitor.signalInput();**

**180 monitor.waitInput();**

**181 monitor.copy\_a\_B\_MO(this);**

**182 calc1();**

**183 monitor.signalCalc1();**

**184 monitor.waitCalc1();**

**185**

**186 monitor.copy\_q(this);**

**187 calc2();**

**188 monitor.signalCalc2();**

**189 monitor.waitCalc2();**

**190 output();**

**191 }**

**192 }**

**193**

**194 static class Task1 extends Task {**

**195 Task1(Monitor monitor) {**

**196 super(monitor, 0);**

**197 }**

**198 @Override**

**199 protected void input() {**

**200 int[] B = IO.readVector();**

**201 Main.MX = IO.readMatrix();**

**202 monitor.write\_B(B);**

**203 }**

**204**

**205 @Override**

**206 protected void output() {**

**207 monitor.waitCalc2();**

**208**

**209 // Output A**

**210 if (A.length <= 24) {**

**211 IO.output(A);**

**212 }**

**213 }**

**214 }**

**215**

**216 static class Task4 extends Task {**

**217 Task4(Monitor monitor) {**

**218 super(monitor, 3);**

**219 }**

**220 @Override**

**221 protected void input() {**

**222 int a = IO.readInt();**

**223 Main.Z = IO.readVector();**

**224 Main.R = IO.readVector();**

**225 monitor.write\_a(a);**

**226 }**

**227 }**

**228**

**229 static class Task6 extends Task {**

**230 Task6(Monitor monitor) {**

**231 super(monitor, 5);**

**232 }**

**233 @Override**

**234 protected void input() {**

**235 int[][] MO = IO.readMatrix();**

**236 monitor.write\_MO(MO);**

**237 Main.E = IO.readVector();**

**238 }**

**239 }**

**240**

**241 public static void main(String[] args) throws InterruptedException {**

**242 Thread[] threads = {**

**243 new Thread(new Task1(monitor )),**

**244 new Thread(new Task (monitor, 1)),**

**245 new Thread(new Task (monitor, 2)),**

**246 new Thread(new Task4(monitor )),**

**247 new Thread(new Task (monitor, 4)),**

**248 new Thread(new Task6(monitor ))**

**249 };**

**250**

**251 for (Thread thread : threads) {**

**252 thread.start();**

**253 }**

**254**

**255 for (Thread thread : threads) {**

**256 thread.join();**

**257 }**

**258 }**

**259 }**