НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №6

з дисципліни **«**Програмування паралельних комп’ютерних систем**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Корочкін О. В.

Київ – 2015 р.

Тема: Програмування для комп’ютерних систем зі спільною пам’яттю.

Розробити програму для розв’язання в ПКС зі СП математичної задачі: MA = (B∙C)∙MO + α∙(MT∙MR)

Мова програмування: Java

Засоби організації взаємодії: монітори



Структурна схема ПКС

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. vi = BH\*CH, *i* =
2. v = v + vi, *i* =
3. MAH = vi∙MOH + αi∙(MTH∙MRi)

Спільний ресурс: α, v, MR

**Етап 2. Розроблення алгоритмів роботи кожного процесу**

Зручним варіантом реалізації є написання єдиного алгоритму для всіх задач. Кожна задача має свій ідентифікатор tid (tid=).

|  |  |
| --- | --- |
| **Задача Тi (i=)** | КД |
| 1. Якщо tid = 1, Введення MT, α |  |
| 2. Якщо tid = 1, **Сигнал** про завершення вводу. |  |
| 3. Якщо tid = 3, Введення С, MO |  |
| 4. Якщо tid = 2, **Сигнал** про завершення вводу. |  |
| 5. Якщо tid = 6, Введення B, MR |  |
| 6. Якщо tid = 6, **Сигнал** про завершення вводу. |  |
| 7. **Чекати** завершення вводу в інших задачах |  |
| 8. Обчислення 1: vi = BH∙CH |  |
| 9. Обчислення 2: v = v + vi | КД |
| 10. **Сигнал** про завершення обчислення 2. |  |
| 11. **Чекати** завершення обчислення 2 в інших задачах |  |
| 12. Копіювати αi := α, vi = v, MRi = MR | КД |
| 13. Обчислення 3: MAH = vi∙MOH + αi∙(MTH∙MRi) |  |
| 14. Якщо tid !=6, **Сигнал** про завершення обчислення 3 |  |
| 15. Якщо tid = 6, **Чекати** на завершення обчислення 3 в інших задачах |  |
| 16. Якщо tid = 6, Вивести MA |  |

**Етап 3. Розроблення структурної схеми взаємодії задач**



Етап 4. Розроблення програми

1. **package** pro2\_lab6;

4. */\*\**
5. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
6. *\* \**
7. *\* Laboratory work #6. Java. Monitors \**
8. *\* \**
9. *\* Task: MA = (B\*C)\*MO + α\*(MT\*MR) \**
10. *\* \**
11. *\* @file CalculateUtils.java \**
12. *\* @author Kuzmenko Volodymyr \**
13. *\* @group IO-21 \**
14. *\* @date 17.04.2015 \**
15. *\* \**
16. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
17. *\*/*
18. **public** **class** CalculateUtils {


22. **public** **static** Vector inputVector(**int** value) {
23. Vector vector = **new** Vector(Executor.N);
24. **for**(**int** i = 0; i < vector.size(); i++) {
25. vector.**set**(i, value);
26. }
27. **return** vector;
28. }

31. **public** **static** Matrix inputMatrix(**int** value) {
32. Matrix matrix = **new** Matrix(Executor.N);
33. **for**(**int** i = 0; i < matrix.size(); i++) {
34. **for**(**int** j = 0; j < matrix.size(); j++) {
35. matrix.**set**(i, j, value);
36. }
37. }
38. **return** matrix;
39. }

42. **public** **static** **void** outputVector(Vector vector) {
43. **if**(vector.size() <= 12) {
44. System.**out**.**print**(vector.toString());
45. }
46. }

49. **public** **static** **void** outputMatrix(Matrix matrix) {
50. **if**(matrix.size() <= 12) {
51. System.**out**.**print**(matrix.toString());
52. }
53. }

56. **private** **static** Matrix add(**final** Matrix left, **final** Matrix right,
57. **final** **int** id) {
58. **int** l = (id - 1) \* Executor.H;
59. **int** r = id \* Executor.H;
60. Matrix result = **new** Matrix(left.size());
61. **for**(**int** i = l; i < r; i++) {
62. **for** (**int** j = 0; j < left.size(); j++) {
63. result.**set**(i,j, left.**get**(i,j) + right.**get**(i,j));
64. }
66. }
67. **return** result;
68. }

71. **private** **static** Matrix mult(**final** **int** left, **final** Matrix right,
72. **final** **int** id) {
73. **int** l = (id - 1) \* Executor.H;
74. **int** r = id \* Executor.H;
75. Matrix result = **new** Matrix(right.size());
76. **for** (**int** i = l; i < r; i++) {
77. **for** (**int** j = 0; j < right.size(); j++) {
78. result.**set**(i,j, left \* right.**get**(i, j));
79. }
80. }
82. **return** result;
83. }
85. **private** **static** Matrix mult(**final** Matrix left, **final** Matrix right,
86. **final** **int** id) {
87. **int** l = (id - 1) \* Executor.H;
88. **int** r = id \* Executor.H;
89. Matrix result = **new** Matrix(left.size());
90. **for** (**int** i = l; i < r; i++) {
91. **for** (**int** j = 0; j < left.size(); j++) {
92. result.**set**(i, j, 0);
93. **for** (**int** y = 0; y < left.size(); y++) {
94. result.**set**(i, j, result.**get**(i, j) + left.**get**(i, y)
95. \* right.**get**(y, j));
96. }
97. }
98. }

101. **return** result;
102. }
103. **private** **static** **int** mult(**final** Vector left, **final** Vector right, **final** **int** id){
105. **int** l = (id - 1) \* Executor.H;
106. **int** r = id \* Executor.H;
107. **int** result = 0;
108. **for** (**int** i = l; i < r; i++) {
109. result += left.**get**(i)\*right.**get**(i);
110. }

113. **return** result;
114. }
116. **public** **static** **void** operation3(**final** **int** alfai, **final** **int** vi, **final** Matrix MRi, **int** tid ){
117. Matrix MAi = CalculateUtils.add(mult(vi, Executor.MO, tid), mult(alfai, mult(MRi, Executor.MT, tid), tid), tid);
118. **int** l = (tid - 1) \* Executor.H;
119. **int** r = tid \* Executor.H;
121. **for** (**int** i = l; i < r; i++) {
122. **for** (**int** j = 0; j < Executor.N; j++) {
123. Executor.MA.**set**(i, j, MAi.**get**(i, j));
124. }
125. }
126. }
127. **public** **static** **int** operation1(**int** tid){
128. **return** CalculateUtils.mult(Executor.B, Executor.C, tid);
129. }
131. }
132. **package** pro2\_lab6;

135. */\*\**
136. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
137. *\* \**
138. *\* Laboratory work #6. Java. Monitors \**
139. *\* \**
140. *\* Task: MA = (B\*C)\*MO + α\*(MT\*MR) \**
141. *\* \**
142. *\* @file CalculationWorker.java \**
143. *\* @author Kuzmenko Volodymyr \**
144. *\* @group IO-21 \**
145. *\* @date 17.04.2015 \**
146. *\* \**
147. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
148. *\*/*

151. **public** **class** CalculationWorker **implements** Runnable {
152. **private** **int** tid;
153. **private** GeneralResourseMonitor resMonitor;
154. **private** SynchronizationMonitor syncMonitor;
156. **public** CalculationWorker(GeneralResourseMonitor resourseMonitor,
157. SynchronizationMonitor syncMonitor, **int** id) {
158. **this**.resMonitor = resourseMonitor;
159. **this**.syncMonitor = syncMonitor;
160. **this**.tid = id;
161. }

164. @Override
165. **public** **void** run() {
166. System.**out**.println("Thread "+tid + " started.");
168. **switch** (**this**.tid) {
169. **case** 1:
170. *//1. Якщо tid = 1, Введення MT, α*
171. Executor.MT = CalculateUtils.inputMatrix(1);
172. resMonitor.setAlfa(1);
173. *//2. Якщо tid = 1, Сигнал про завершення вводу.*
174. syncMonitor.inputSignal();
175. **break**;
176. **case** 3:
177. *//3. Якщо tid = 3, Введення С, MO*
178. Executor.C = CalculateUtils.inputVector(1);
179. Executor.MO = CalculateUtils.inputMatrix(1);
180. *//4. Якщо tid = 2, Сигнал про завершення вводу.*
181. syncMonitor.inputSignal();
182. **break**;
183. **case** 6:
184. *//5. Якщо tid = 6, Введення B, MR*
185. Executor.B = CalculateUtils.inputVector(1);
186. Matrix MR = CalculateUtils.inputMatrix(1);
187. resMonitor.setMR(MR);
188. *//6. Якщо tid = 6, Сигнал про завершення вводу.*
189. syncMonitor.inputSignal();
190. **break**;
191. }
192. *//7. Чекати завершення вводу в інших задачах*
193. syncMonitor.waitForInput();
194. *//8. Обчислення 1: vi = BH∙CH*
195. **int** vi = CalculateUtils.operation1(tid);
196. *//9. Обчислення 2: v = v + vi*
197. resMonitor.addV(vi);
198. *//10. Сигнал про завершення обчислення 2.*
199. syncMonitor.calcVSignal();
200. *//11. Чекати завершення обчислення 2 в інших задачах*
201. syncMonitor.waitForCalcV();
202. *//12. Копіювати αi := α, vi = v, MRi = MR*
203. **int** alfai = resMonitor.copyAlfa();
204. vi = resMonitor.copyV();
205. Matrix MRi = resMonitor.copyMR();
206. *//13. Обчислення 3: MAH = vi∙MOH + αi∙(MTH∙MRi)*
207. CalculateUtils.operation3(alfai, vi, MRi, tid);
209. **if** (tid == 6){
210. *//15. Якщо tid = 6, Чекати на завершення обчислення 3 в інших задачах*
211. *//16. Якщо tid = 6, Вивести MA*
212. syncMonitor.waitForCalcMA();
213. CalculateUtils.outputMatrix(Executor.MA);
214. }**else**{
215. *//14. Якщо tid !=6, Сигнал про завершення обчислення 3*
216. syncMonitor.calcMASignal();
217. }
218. System.**out**.println("Thread "+tid + " finished.");
219. }
220. }
221. **package** pro2\_lab6;

224. **import** java.util.concurrent.ExecutorService;
225. **import** java.util.concurrent.Executors;

228. */\*\**
229. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
230. *\* \**
231. *\* Laboratory work #6. Java. Monitors \**
232. *\* \**
233. *\* Task: MA = (B\*C)\*MO + α\*(MT\*MR) \**
234. *\* \**
235. *\* @file Executor.java \**
236. *\* @author Kuzmenko Volodymyr \**
237. *\* @group IO-21 \**
238. *\* @date 17.04.2015 \**
239. *\* \**
240. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
241. *\*/*
242. **public** **class** Executor {
243. **public** **static** **int** N = 12;
244. **public** **static** **int** P = 6;
245. **public** **static** **int** H = N / P;

248. **public** **static** Matrix MA = **new** Matrix(N);
249. **public** **static** Matrix MO = **new** Matrix(N);
250. **public** **static** Matrix MT = **new** Matrix(N);
251. **public** **static** Vector B = **new** Vector(N);
252. **public** **static** Vector C = **new** Vector(N);
254. **public** **static** **void** main(String[] args) {
255. System.**out**.println("Lab 6 started");
256. ExecutorService service = Executors.newFixedThreadPool(P);
257. GeneralResourseMonitor resourseMonitor = **new** GeneralResourseMonitor();
258. SynchronizationMonitor syncMonitor = **new** SynchronizationMonitor();



263. **for** (**int** id = 1; id <= P; id++) {
264. service.execute(**new** CalculationWorker(resourseMonitor, syncMonitor, id));
265. }
267. service.shutdown();
268. System.**out**.println("Lab 6 finished");
269. }

272. }
273. **package** pro2\_lab6;
274. */\*\**
275. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
276. *\* \**
277. *\* Laboratory work #6. Java. Monitors \**
278. *\* \**
279. *\* Task: MA = (B\*C)\*MO + α\*(MT\*MR) \**
280. *\* \**
281. *\* @file Matrix.java \**
282. *\* @author Kuzmenko Volodymyr \**
283. *\* @group IO-21 \**
284. *\* @date 17.04.2015 \**
285. *\* \**
286. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
287. *\*/*
288. **public** **class** Matrix {
290. **private** Vector[] array;

293. **public** Matrix(**int** n) {
294. array = **new** Vector[n];
295. **for** (**int** i = 0; i < array.length; i++){
296. array[i] = **new** Vector(n);
297. }
298. }

301. **public** **void** **set**(**int** n, **int** m, **int** val) {
302. array[n].**set**(m, val);
303. }
305. **public** **int** **get**(**int** n, **int** m) {
306. **return** array[n].**get**(m);
307. }

310. **public** Vector **get**(**int** index) {
311. **return** array[index];
312. }
314. **public** **int** size() {
315. **return** array.length;
316. }
318. **public** String toString() {
319. String res = "";
320. **for** (**int** i = 0; i < array.length; i++){
321. res += array[i].toString();
322. }
323. **return** res;
324. }
325. }
326. **package** pro2\_lab6;
327. */\*\**
328. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
329. *\* \**
330. *\* Laboratory work #6. Java. Monitors \**
331. *\* \**
332. *\* Task: MA = (B\*C)\*MO + α\*(MT\*MR) \**
333. *\* \**
334. *\* @file SynchronizationMonitor.java \**
335. *\* @author Kuzmenko Volodymyr \**
336. *\* @group IO-21 \**
337. *\* @date 17.04.2015 \**
338. *\* \**
339. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
340. *\*/*
341. **public** **class** SynchronizationMonitor {
342. **private** **int** inputFlag = 0;
343. **private** **int** vFlag = 0;
344. **private** **int** MaFlag = 0;

347. **public** **synchronized** **void** inputSignal() {
348. inputFlag++;
350. **if** (inputFlag >= 3) {
351. notifyAll();
353. }
354. }

357. **public** **synchronized** **void** calcVSignal() {
358. vFlag++;
359. **if** (vFlag >= Executor.P)
360. notifyAll();
361. }

364. **public** **synchronized** **void** calcMASignal() {
365. MaFlag++;
366. **if** (MaFlag == Executor.P-1)
367. notifyAll();
368. }

371. **public** **synchronized** **void** waitForInput() {
372. **try** {
373. **if** (inputFlag<3) wait();
374. } **catch** (InterruptedException e) {
375. *// TODO Auto-generated catch block*
376. e.printStackTrace();
377. }
378. }

381. **public** **synchronized** **void** waitForCalcV() {
382. **try** {
383. **if** (vFlag<Executor.P)
384. wait();
385. } **catch** (InterruptedException e) {
386. *// TODO Auto-generated catch block*
387. e.printStackTrace();
388. }
389. }

392. **public** **synchronized** **void** waitForCalcMA() {
393. **try** {
394. **if** (MaFlag<Executor.P-1)
395. wait();
396. } **catch** (InterruptedException e) {
397. *// TODO Auto-generated catch block*
398. e.printStackTrace();
399. }
400. }
401. }
402. **package** pro2\_lab6;
403. */\*\**
404. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
405. *\* \**
406. *\* Laboratory work #6. Java. Monitors \**
407. *\* \**
408. *\* Task: MA = (B\*C)\*MO + α\*(MT\*MR) \**
409. *\* \**
410. *\* @file Vector.java \**
411. *\* @author Kuzmenko Volodymyr \**
412. *\* @group IO-21 \**
413. *\* @date 17.04.2015 \**
414. *\* \**
415. *\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \**
416. *\*/*
417. **public** **class** Vector {
418. **private** **int**[] array;

421. **public** Vector(**int** n) {
422. array = **new** **int**[n];
423. }

426. **public** **void** **set**(**int** index, **int** value) {
427. array[index] = value;
428. }
430. **public** **int** **get**(**int** index) {
431. **return** array[index];
432. }
434. **public** **int** size() {
435. **return** array.length;
436. }
438. **public** String toString() {
439. String res = "";
440. **for** (**int** i = 0; i < array.length; i++){
441. res += " " + array[i];
442. }
443. res += "\n";
444. **return** res;
445. }

448. }

|  |  |
| --- | --- |
|  | package pro2\_lab6; |
|  | /\*\* |
|  | \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* |
|  | \* \* |
|  | \* Laboratory work #6. Java. Monitors \* |
|  | \* \* |
|  | \* Task: MA = (B\*C)\*MO + α\*(MT\*MR) \* |
|  | \* \* |
|  | \* @file GeneralResourseMonitor.java \* |
|  | \* @author Kuzmenko Volodymyr \* |
|  | \* @group IO-21 \* |
|  | \* @date 17.04.2015 \* |
|  | \* \* |
|  | \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* |
|  | \*/ |
|  | public class GeneralResourseMonitor { |
|  | private Matrix MR; |
|  | private int v = 0; |
|  | private int alfa; |
|  |  |
|  | public synchronized void setAlfa(int value){ |
|  | this.alfa = value; |
|  | } |
|  |  |
|  | public synchronized void addV(int value){ |
|  | this.v += value; |
|  | } |
|  |  |
|  | public synchronized void setMR(Matrix value){ |
|  | this.MR = value; |
|  | } |
|  |  |
|  | public synchronized int copyAlfa(){ |
|  | return this.alfa; |
|  | } |
|  |  |
|  | public synchronized int copyV(){ |
|  | return this.v; |
|  | } |
|  |  |
|  | public synchronized Matrix copyMR(){ |
|  | return this.MR; |
|  | } |
|  | } |