Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Лабораторна робота №6

з дисципліни «Паралельне програмування -2. Програмування для паралельних систем»

Виконав

студент групи ІП-31

Кобилинський Дмитро

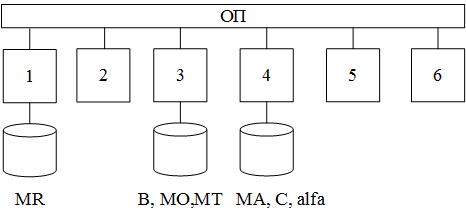
Київ 2016

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

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

Мова програмування: Java

Засоби організації взаємодії: монітор, fork/join



**Етап 1. Побудова паралельного алгоритму**

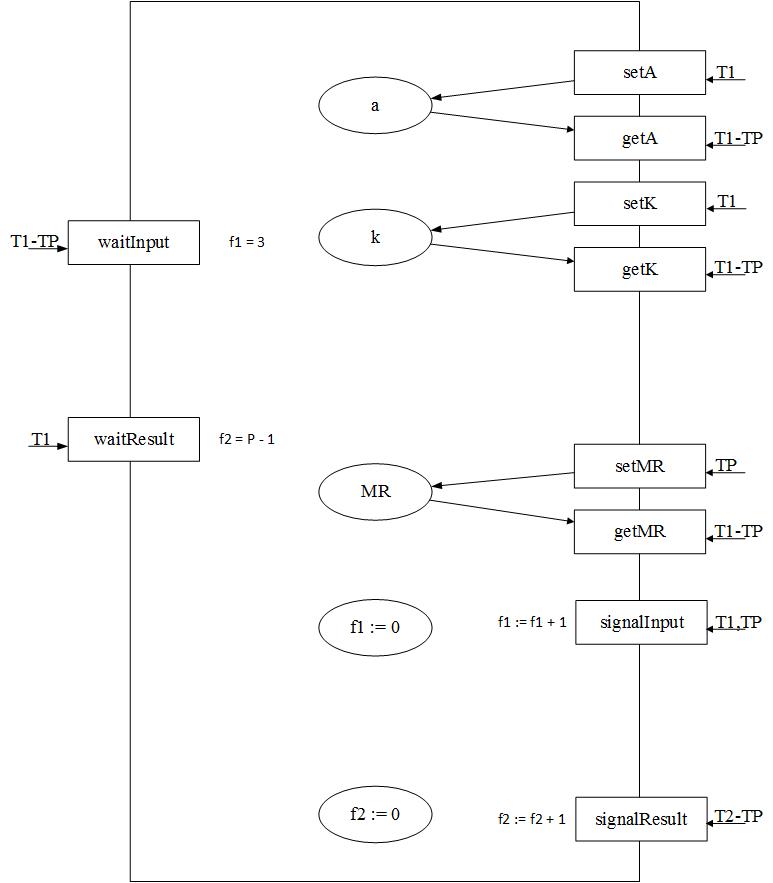
1. ki = BH\*CH, *i* =
2. k = k+ki, *i* =
3. MAH = α ∙MOH + k∙(MTH∙MR)

Спільний ресурс: α, k, MR

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

|  |  |
| --- | --- |
| **Задача Т1** |  |
| 1. Введення MR |  |
| 2. **Сигнал** про завершення вводу |  |
| 3. **Чекати** вводу в інших задачах |  |
| 4. Обчислення 1: k1 = BH∙CH |  |
| 5. Обчислення 2: k = k+k1 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. |  |
| 7. **Чекати** завершення обчислення 3 в інших задачах |  |
| 8. Копіювати α1 := α, k1 = k, MR1 = MR | КД |
| 9. Обчислення 3: MAH = α 1∙MOH + k1∙(MTH∙MR1) |  |
| 10. **Сигнал** про завершення обчислення 3 |  |
| **Задача T3** |  |
| 1. **Введення** B,MO,MT |  |
| 2. Сигнал задачам про завершення вводу |  |
| 3. **Чекати** вводу в інших задачах |  |
| 4. Обчислення 1: k3 = BH∙CH |  |
| 5. Обчислення 2: k = k+k3 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. |  |
| 7. **Чекати** завершення обчислення 2 в інших задачах |  |
| 8. Копіювати α3 := α, k3 = k, MR3 = MR | КД |
| 9. Обчислення 3: MAH = α 3∙MOH + k3∙(MTH∙MR1) |  |
| 10. **Сигнал** про завершення обчислення 3 |  |
| **Задача Т4** |  |
| 1. ВведенняC, α |  |
| 2. **Сигнал** задачам про завершення вводу |  |
| 3. **Чекати** вводу в інших задачах |  |
| 4. Обчислення 1: k4 = BH∙CH |  |
| 5. Обчислення 2: k = k+k4 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. |  |
| 7. **Чекати** завершення обчислення 2 в інших задачах |  |
| 8. Копіювати α4 := α, k6 = k, MR6 = MR | КД |
| 9. Обчислення 3: MAH = k4∙MOH + α4∙(MTH∙MR4) |  |
| 10. **Чекати** на завершення обчислення 3 в інших задачах |  |
| 11. Вивести MA |  |
| **Задачі Т2,5,6** |  |
| 1. **Чекати** на завершення вводу в інших задачах |  |
| 2. Обчислення 1: ki = BH∙CH |  |
| 3. Обчислення 2: k = k+ki | КД |
| 4. **Сигнал** задачам про завершення обчислення 2. |  |
| 5. **Чекати** завершення обчислення 2 в інших задачах |  |
| 6. Копіювати αi := α, ki = k, MRi = MR | КД |
| 7. Обчислення 3: MAH = α i∙MOH + ki∙(MTH∙MRi) |  |
| 8. **Сигнал** про завершення обчислення 3. |  |

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



* set\_a – запис в монітор константи *a*;
* get\_a – зчитування з монітору константи *a*;
* set\_k – запис в монітор константи *k*;
* get\_k – зчитування з монітору константи *k*;
* setMR – запис в монітор матриці MR;
* getMR – зчитування з монітору матриці MR;
* signalInput – сигнал про закінчення вводу в потоках Т1, T2, ТР; збільшення флагу f1 на 1;
* signalResult – сигнал про закінчення основних обчислень в потоках Т2 – ТР; збільшення флагу f2 на 1;
* waitInput – очікування сигналу про закінчення вводу в потоках Т1, T2, ТР; перевірка флагу f1 на рівність f1 = 3;
* waitResult – очікування сигналу про закінчення основних обчислень в потоках Т2 – ТР; перевірка флагу f2 на рівність f2 = P - 1;

**Етап 4 . Лістинг коду**

**Main.java**

*/\*  
PP-2  
Lab6 . Java. Monitor. Fork/Join  
Kobylynskiy Dmytro IP-31  
20.04.16  
 \*/***public class** Main {  
  
 **public static void** main(String[] args) {  
 **for** (**int** i = 0; i < 6; i++) {  
 **new** Task().start();  
 }  
 }  
  
}

**DotProduct.java**

**import** java.util.Arrays;  
**import** java.util.concurrent.RecursiveTask;  
*/\*  
PP-2  
Lab6 . Java. Monitor. Fork/Join  
Kobylynskiy Dmytro IP-31  
20.04.16  
 \*/***public class** DotProduct **extends** RecursiveTask<Integer> {  
  
 **private** Vector **A**, **B**;  
 **private static int** *minSize*;  
  
 **public** DotProduct(Vector A, Vector B) {  
 **this**.**A** = A;  
 **this**.**B** = B;  
 }  
  
 **public** DotProduct(Vector A, Vector B, **int** minSize) {  
 **this**.**A** = A;  
 **this**.**B** = B;  
 DotProduct.*minSize* = minSize;  
 }  
  
 @Override  
 **protected** Integer compute() {  
 **int** res = 0;  
 **if** (**A**.size() != *minSize*) {  
 Vector childOfA1 = **new** Vector(**A**.size() / 2);  
 Vector childOfA2 = **new** Vector(**A**.size() / 2);  
 Vector childOfB1 = **new** Vector(**B**.size() / 2);  
 Vector childOfB2 = **new** Vector(**B**.size() / 2);  
  
 **for** (**int** i = 0; i < childOfA1.size(); i++) {  
 childOfA1.set(i, **A**.get(i));  
 childOfA2.set(i, **A**.get(childOfA1.size() + i));  
 childOfB1.set(i, **B**.get(i));  
 childOfB2.set(i, **B**.get(childOfB1.size() + i));  
 }  
  
 DotProduct child1 = **new** DotProduct(childOfA1, childOfB1);  
 child1.fork();  
 DotProduct child2 = **new** DotProduct(childOfA2, childOfB2);  
 child2.fork();  
 System.***out***.println(Arrays.*toString*(childOfA1.**array**));  
 System.***out***.println(childOfA1.size());  
 System.***out***.println(Arrays.*toString*(childOfA2.**array**));  
 res += child1.join();  
 res += child2.join();  
  
 } **else** {  
 **for** (**int** i = 0; i < **A**.size(); i++) {  
 res += **A**.get(i) \* **B**.get(i);  
 }  
 }  
 **return** res;  
 }  
}

**Monitor.java**

*/\*  
PP-2  
Lab6 . Java. Monitor. Fork/Join  
Kobylynskiy Dmytro IP-31  
20.04.16  
 \*/***public class** Monitor {  
  
 **private int n**, **p**;  
 **private int a**, **k**;  
 **private int f1**, **f2**;  
 **private** Matrix **MR**;  
  
 **public** Monitor(**int** p, **int** n) {  
 **this**.**p** = p;  
 **this**.**n** = n;  
 }  
  
 **synchronized void** set\_a(**int** a) {  
 **this**.**a** = a;  
 }  
  
 **synchronized void** set\_k(**int** k) {  
 **this**.**k** = k;  
 }  
  
 **synchronized void** setMR(**int** value) {  
 **MR** = **new** Matrix(**n**);  
 **for** (**int** i = 0; i < **n**; i++) {  
 **for** (**int** j = 0; j < **n**; j++) {  
 **MR**.set(i, j, value);  
 }  
 }  
 }  
  
 **synchronized int** get\_a() {  
 **return a**;  
 }  
  
 **synchronized int** get\_k() {  
 **return k**;  
 }  
  
 **synchronized void** getMR(Matrix MR) {  
 **for** (**int** i = 0; i < **n**; i++) {  
 **for** (**int** j = 0; j < **n**; j++) {  
 MR.set(i, j, **this**.**MR**.get(i, j));  
 }  
 }  
 }  
  
 **synchronized void** waitInput() {  
 **if** (**f1** < 3) {  
 **try** {  
 wait();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 **synchronized void** signalInput() {  
 **f1**++;  
 **if** (**f1** == 3) {  
 notifyAll();  
 }  
 }  
  
 **synchronized void** waitResult() {  
 **if** (**f2** < (**p** - 1)) {  
 **try** {  
 wait();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 **synchronized void** signalResult() {  
 **f2**++;  
 **if** (**f2** == (**p** - 1)) {  
 notifyAll();  
 }  
 }  
  
}

**Task.java**

**import** java.util.concurrent.ForkJoinPool;  
*/\*  
PP-2  
Lab6 . Java. Monitor. Fork/Join  
Kobylynskiy Dmytro IP-31  
20.04.16  
 \*/***public class** Task **extends** Thread {  
  
 **private static int** *counter* = 1;  
 **private static** Vector *B*;  
 **private static** Vector *C*;  
 **private static** Matrix *MA*;  
 **private static** Matrix *MT*;  
 **private static** Matrix *MO*;  
  
 **private int id** = *counter*++;  
 **private int p** = 6, **n** = 6, **h** = **n** / **p**;  
  
 **private static** Monitor *monitor*;  
  
 **public** Task() {  
 *monitor* = **new** Monitor(**p**, **n**);  
 }  
  
 @Override  
 **public void** run() {  
  
 **int** copy\_a;  
 **int** copy\_k;  
 Matrix copyMR = **new** Matrix(**n**);  
  
 System.***out***.println(**"Task "** + **id** + **" started!"**);  
 **switch** (**id**) {  
 **case** 1 :  
  
 *monitor*.setMR(1);  
 *monitor*.signalInput();  
 **break**;  
 **case** 4 :  
 *MA* = **new** Matrix(**n**);  
 *MA*.fill(0);  
 *monitor*.set\_a(1);  
 *C* = **new** Vector(**n**);  
 *C*.fill(1);  
 *monitor*.signalInput();  
 **break**;  
 **case** 3 :  
 *B* = **new** Vector(**n**);  
 *MO* = **new** Matrix(**n**);  
 *MT* = **new** Matrix(**n**);  
 *MT*.fill(1);  
 *B*.fill(1);  
 *MO*.fill(1);  
 *monitor*.signalInput();  
 **break**;  
 }  
 *monitor*.waitInput();  
 ForkJoinPool forkJoinPool = **new** ForkJoinPool(**p**);  
  
 **if** (**id** == 1) {  
 *monitor*.set\_k(forkJoinPool.invoke(**new** DotProduct(*B*,*C*,**h**)));  
 System.***out***.println(*monitor*.get\_k());  
 }  
  
  
 **while**(*monitor*.get\_k() == 0);  
 copy\_a = *monitor*.get\_a();  
 copy\_k = *monitor*.get\_k();  
 *monitor*.getMR(copyMR);  
  
 **for** (**int** i = **h** \* (**id** - 1); i < (**h** \* **id**); i++) {  
 **for** (**int** j = 0; j < **n**; j++) {  
 **for** (**int** k = 0; k < **n**; k++) {  
 *MA*.set(i, j, *MA*.get(i, j) + *MO*.get(i, k) \* copyMR.get(k, j));  
 }  
 *MA*.set(i, j, *MA*.get(i, j) \* copy\_k + copy\_a \* *MO*.get(i, j));  
 }  
 }  
  
 **if** (**id** == 1) {  
 *monitor*.waitResult();  
 System.***out***.println(*MA*);  
 } **else** {  
 *monitor*.signalResult();  
 }  
  
  
  
 }  
}

**Matrix.java**

*/\*  
PP-2  
Lab6 . Java. Monitor. Fork/Join  
Kobylynskiy Dmytro IP-31  
20.04.16  
 \*/***public class** Matrix {  
  
 **private** Vector[] **matrix**;  
  
 **public** Matrix(**int** n) {  
 **matrix** = **new** Vector[n];  
 **for** (**int** i = 0; i < n; i++) {  
 **matrix**[i] = **new** Vector(n);  
 }  
 }  
  
 **public void** set(**int** m, **int** n, **int** value) {  
 **matrix**[m].set(n, value);  
 }  
  
 **public int** get(**int** m, **int** n) {  
 **return matrix**[m].get(n);  
 }  
  
 **public void** fill(**int** value) {  
 **for** (**int** i = 0; i < **matrix**.**length**; i++) {  
 **for** (**int** j = 0; j < **matrix**.**length**; j++)  
 **matrix**[i].set(j, value);  
 }  
 }  
  
 **public int** size() {  
 **return matrix**.**length**;  
 }  
  
 @Override  
 **public** String toString() {  
 String res = **""**;  
 **for** (**int** i = 0; i < **matrix**.**length**; i++) {  
 res += **matrix**[i].toString() + **"\n"**;  
 }  
 **return** res;  
 }  
  
}

**Vector.java**

*/\*  
PP-2  
Lab6 . Java. Monitor. Fork/Join  
Kobylynskiy Dmytro IP-31  
20.04.16  
 \*/***public class** Vector {  
  
 **public int**[] **array**;  
  
 **public** Vector(**int** n) {  
 **array** = **new int**[n];  
 }  
  
 **public void** set(**int** index, **int** value) {  
 **array**[index] = value;  
 }  
  
 **public int** get(**int** index) {  
 **return array**[index];  
 }  
  
 **public void** fill(**int** value) {  
 **for** (**int** i = 0; i < **array**.**length**; i++) {  
 **array**[i] = value;  
 }  
 }  
  
 **public int** size() {  
 **return array**.**length**;  
 }  
  
 @Override  
 **public** String toString() {  
 String res = **""**;  
 **for** (**int** i = 0; i < **array**.**length**; i++) {  
 res += **array**[i] + **" "**;  
 }  
 **return** res;  
 }  
}