Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# Лабораторна робота №6

# з курсу «Паралельні та розподілені обчислення-2»

**Тема:** Java. Захищений модуль

Виконав: студент групи ІО-93

Глухенько Костянтин Анатолійович

2012 р.

**Завдання**

Математичний вираз: MA=max(Z)(MB\*MC-ME)

Засоби програмування: Java (Захищений модуль)

Структура ПКС:

ЗП

Т1

…

P

P-1

Т2

ME

MC

Z MA

MB

**Виконання**

**Етап 1.** Побудова паралельного алгоритму.

1. ai=max(ZH); i=1,P; H=N/P
2. a=max(a, ai); i=1,P
3. MAH=a\*(MB\*MCH-MEH)

СР: a,MB

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

|  |  |
| --- | --- |
|  | Т1 |
| 1) | Введення Z |
| 2) | Сигнал про введення Z в задачі Т1 |
| 3) | Обчислення a1=max(ZH); |
| 4) | Обчислення a=max(a, a1); |
| 5) | Сигнал про завершення обчислень а в задачі Т1 |
| 6) | Чекати завершення введення в задачах Т2, ТP-1, ТP |
| 7) | Копіювання MB1 = MB |
| 8) | Чекати завершення обчислень а |
| 9) | Копіювання а1 = а |
| 10) | Обчислення MAH=a1\*(MB1\*MCH-MEH) |
| 11) | Чекати завершення обчислень MA |
| 12) | Виведення MA |

|  |  |
| --- | --- |
|  | Т2 |
| 1) | Введення MВ |
| 2) | Сигнал про введення МВ в задачі Т2 |
| 3) | Чекати введення Z в задачі Т1 |
| 4) | Обчислення a2=max(ZH); |
| 5) | Обчислення a=max(a, a2); |
| 6) | Сигнал про завершення обчислень а в задачі Т2 |
| 7) | Копіювання MB2 = MB |
| 8) | Чекати завершення обчислень а |
| 9) | Копіювання а2 = а |
| 10) | Чекати завершення введення в задачах TP-1, TP |
| 11) | Обчислення MAH=a2\*(MB2\*MCH-MEH) |
| 12) | Сигнал про завершення обчислень MA в задачі Т2 |

|  |  |
| --- | --- |
|  | Т3…TP-2 |
| 1) | Чекати на введення Z в задачі Т1 |
| 2) | Обчислення ai=max(ZH); |
| 3) | Обчислення a=max(a, ai); |
| 4) | Сигнал про завершення обчислень а в задачі Т3…TP-2 |
| 5) | Чекати завершення введення в задачах Т2, ТP-1, ТP |
| 6) | Копіювання MBi = MB |
| 7) | Чекати завершення обчислень а |
| 8) | Копіювання аi = а |
| 9) | Обчислення MAH=ai\*(MBi\*MCH-MEH) |
| 10) | Сигнал про завершення обчислень MA в задачі Т3…TP-2 |

|  |  |
| --- | --- |
|  | ТP-1 |
| 1) | Введення MС |
| 2) | Сигнал про введення МС в задачі TP-1 |
| 3) | Чекати на введення Z в задачі Т1 |
| 4) | Обчислення aP-1=max(ZH); |
| 5) | Обчислення a=max(a, aP-1); |
| 6) | Сигнал про завершення обчислень а в задачі TP-1 |
| 7) | Чекати завершення введення в задачах Т2, TP |
| 8) | Копіювання MBP-1 = MB |
| 9) | Чекати завершення обчислень а |
| 10) | Копіювання aP-1 = а |
| 11) | Обчислення MAH=aP-1\*(MBP-1\*MCH-MEH) |
| 12) | Сигнал про завершення обчислень MA в задачі TP-1 |

|  |  |
| --- | --- |
|  | TP |
| 1) | Введення MЕ |
| 2) | Сигнал про введення МЕ в задачі TP |
| 3) | Чекати на введення Z в задачі Т1 |
| 4) | Обчислення aP=max(ZH); |
| 5) | Обчислення a=max(a, aP); |
| 6) | Сигнал про завершення обчислень а в задачі TP |
| 7) | Чекати завершення введення в задачах Т2, TP-1 |
| 8) | Копіювання MBP = MB |
| 9) | Чекати завершення обчислень а |
| 10) | Копіювання аP = а |
| 11) | Обчислення MAH=aP\*(MBP\*MCH-MEH) |
| 12) | Сигнал про завершення обчислень MA в задачі TP |

**Етап 3.** Розроблення схеми монітора.

П

max(ai)

Сигнал а

Сигнал Z

Копія а

Чекати а

Чекати Z

BOX1

Т1-TP

Т1-TP

F1=P

Т1-TP

Ф

Т1-TP

F1= F1+1

П

F2=1

Т2-TP

П

Т1

F2= F2+1

Запис MB

Сигнал введення

Сигнал MA

Копія MB

Чекати введення

Чекати MA

BOX2

П

Т2

Ф

Т1-Т6

Т1-TP

F1=3

П

Т2, TP-1, TP

F1= F1+1

F2=Р-1

П

Т1

Т2-ТP

F2= F2+1

**Етап 4.** Розроблення програми**.**

1. //Глухенько Костянтин Анатолійович, група ІО-93
2. //лаб№ 6. Java монітор
3. //MA=max(Z)(MB\*MC-ME)
4. **public** **class** lab6 {
5. **public** **static** **void** main(String[] args) {
6. BOX1 b1 = **new** BOX1(*P*);
7. BOX2 b2 = **new** BOX2(*P*);
8. Task1 t1 = **new** Task1(b1, b2);
9. Task2 t2 = **new** Task2(b1, b2);
10. TaskPm1 tpm1 = **new** TaskPm1(b1, b2);
11. TaskP tp = **new** TaskP(b1, b2);
12. t1.start();
13. t2.start();
14. tp.start();
15. tpm1.start();
16. **for** (**int** i = 3; i < *P* - 1; i++) {
17. Taski t = **new** Taski(b1, b2, i);
18. t.start();
19. }
20. **try** {
21. t1.join();
22. } **catch** (InterruptedException e) {
23. e.printStackTrace();
24. }
25. }
26. **public** **static** **int** *N* = 3000;
27. **public** **static** **int** *P* = 12;
28. **public** **static** **final** **int** *H* = *N* / *P*;
29. **public** **static** **int**[] *Z* = **new** **int**[*N*];
30. **public** **static** **int**[][] *MA* = **new** **int**[*N*][*N*];
31. **public** **static** **int**[][] *MC* = **new** **int**[*N*][*N*];
32. **public** **static** **int**[][] *ME* = **new** **int**[*N*][*N*];
33. }
34. **public** **class** BOX1 {
35. **public** BOX1(**int** P) {
36. **this**.P = P;
37. }
38. **public** **synchronized** **void** max(**int** ai) {
39. **if** (ai > a)
40. a = ai;
41. }
42. **public** **synchronized** **int** Copy\_a() {
43. **return** a;
44. }
45. **public** **synchronized** **void** Signal\_a() {
46. F1++;
47. notify();
48. }
49. **public** **synchronized** **void** Signal\_Z() {
50. F2++;
51. notify();
52. }
53. **public** **synchronized** **void** Wait\_a() {
54. **try** {
55. **while** (F1 != P)
56. wait();
57. notify();
58. } **catch** (InterruptedException e) {
59. e.printStackTrace();
60. }
61. }
62. **public** **synchronized** **void** Wait\_Z() {
63. **try** {
64. **while** (F2 != 1)
65. wait();
66. notify();
67. } **catch** (InterruptedException e) {
68. e.printStackTrace();
69. }
70. }
71. **private** **int** P;
72. **private** **int** F1 = 0;
73. **private** **int** F2 = 0;
74. **private** **int** a = -429496729;
75. }
76. **public** **class** BOX2 {
77. **public** BOX2(**int** P) {
78. **this**.P = P;
79. }
80. **public** **synchronized** **void** Write\_MB(**int**[][] MB) {
81. **this**.MB = MB;
82. }
83. **public** **synchronized** **int**[][] Copy\_MB() {
84. **int**[][] t = MB;
85. **return** t;
86. }
87. **public** **synchronized** **void** Signal\_Input() {
88. F1++;
89. notify();
90. }
91. **public** **synchronized** **void** Signal\_MA() {
92. F2++;
93. notify();
94. }
95. **public** **synchronized** **void** Wait\_Input() {
96. **try** {
97. **while** (F1 != 3)
98. wait();
99. notify();
100. } **catch** (InterruptedException e) {
101. e.printStackTrace();
102. }
103. }
104. **public** **synchronized** **void** Wait\_MA() {
105. **try** {
106. **while** (F2 != P - 1)
107. wait();
108. notify();
109. } **catch** (InterruptedException e) {
110. e.printStackTrace();
111. }
112. }
113. **private** **int** P;
114. **private** **int** F1 = 0;
115. **private** **int** F2 = 0;
116. **private** **int**[][] MB;
117. }
118. **public** **class** Task1 **extends** Thread {
119. **public** Task1(BOX1 box1, BOX2 box2) {
120. **this**.box1 = box1;
121. **this**.box2 = box2;
122. }
123. **public** **void** run() {
124. System.*out*.println("Task 1 Started");
125. // 1
126. **for** (**int** i = 0; i < lab6.*Z*.length; i++)
127. lab6.*Z*[i] = 1;
128. // 2
129. box1.Signal\_Z();
130. // 3
131. **int** a1 = -429496729;
132. **for** (**int** i = 0; i < lab6.*H*; i++)
133. **if** (lab6.*Z*[i] > a1)
134. a1 = lab6.*Z*[i];
135. // 4
136. box1.max(a1);
137. // 5
138. box1.Signal\_a();
139. // 6
140. box2.Wait\_Input();
141. // 7
142. **int**[][] MB1 = box2.Copy\_MB();
143. // 8
144. box1.Wait\_a();
145. // 9
146. a1 = box1.Copy\_a();
147. // 10
148. **for** (**int** i = 0; i < lab6.*H*; i++)
149. **for** (**int** j = 0; j < lab6.*N*; j++) {
150. **for** (**int** k = 0; k < lab6.*N*; k++)
151. lab6.*MA*[i][j] += MB1[i][k] \* lab6.*MC*[k][j];
152. lab6.*MA*[i][j] = a1 \* (lab6.*MA*[i][j] - lab6.*ME*[i][j]);
153. **if** (lab6.*MA*[i][j] == -1)
154. **break**;
155. }
156. // 11
157. box2.Wait\_MA();
158. // 12
159. **if** (lab6.*N* <= 8)
160. **for** (**int** i = 0; i < lab6.*N*; i++) {
161. **for** (**int** j = 0; j < lab6.*N*; j++)
162. System.*out*.print(lab6.*MA*[i][j] + " ");
163. System.*out*.println();
164. }
165. **else**
166. System.*out*.println("MA[0][0]= " + lab6.*MA*[0][0]);
167. System.*out*.println("Task 1 finished");
168. }
169. **private** BOX1 box1;
170. **private** BOX2 box2;
171. }
172. **public** **class** Task2 **extends** Thread {
173. **public** Task2(BOX1 box1, BOX2 box2) {
174. **this**.box1 = box1;
175. **this**.box2 = box2;
176. }
177. **public** **void** run() {
178. System.*out*.println("Task 2 Started");
179. // 1
180. **int**[][] MB = **new** **int**[lab6.*N*][lab6.*N*];
181. **for** (**int** i = 0; i < lab6.*N*; i++)
182. **for** (**int** j = 0; j < lab6.*N*; j++)
183. MB[i][j] = 1;
184. box2.Write\_MB(MB);
185. // 2
186. box2.Signal\_Input();
187. // 3
188. box1.Wait\_Z();
189. // 4
190. **int** a2 = -429496729;
191. **for** (**int** i = lab6.*H*; i < lab6.*H* \* 2; i++)
192. **if** (lab6.*Z*[i] > a2)
193. a2 = lab6.*Z*[i];
194. // 5
195. box1.max(a2);
196. // 6
197. box1.Signal\_a();
198. // 7
199. **int**[][] MB2 = box2.Copy\_MB();
200. // 8
201. box1.Wait\_a();
202. // 9
203. a2 = box1.Copy\_a();
204. // 10
205. box2.Wait\_Input();
206. // 11
207. **for** (**int** i = lab6.*H*; i < lab6.*H* \* 2; i++)
208. **for** (**int** j = 0; j < lab6.*N*; j++) {
209. **for** (**int** k = 0; k < lab6.*N*; k++)
210. lab6.*MA*[i][j] += MB2[i][k] \* lab6.*MC*[k][j];
211. lab6.*MA*[i][j] = a2 \* (lab6.*MA*[i][j] - lab6.*ME*[i][j]);
212. }
213. // 12
214. box2.Signal\_MA();
215. System.*out*.println("Task 2 finished");
216. }
217. **private** BOX1 box1;
218. **private** BOX2 box2;
219. }
220. **public** **class** Taski **extends** Thread {
221. **public** Taski(BOX1 box1, BOX2 box2, **int** n) {
222. **this**.box1 = box1;
223. **this**.box2 = box2;
224. **this**.n = n;
225. }
226. **public** **void** run() {
227. System.*out*.println("Task " + n + " Started");
228. // 1
229. box1.Wait\_Z();
230. // 2
231. **int** ai = -429496729;
232. **for** (**int** i = lab6.*H* \* (n - 1); i < lab6.*H* \* n; i++)
233. **if** (lab6.*Z*[i] > ai)
234. ai = lab6.*Z*[i];
235. // 3
236. box1.max(ai);
237. // 4
238. box1.Signal\_a();
239. // 5
240. box2.Wait\_Input();
241. // 6
242. **int**[][] MBi = box2.Copy\_MB();
243. // 7
244. box1.Wait\_a();
245. // 8
246. ai = box1.Copy\_a();
247. // 9
248. **for** (**int** i = lab6.*H* \* (n - 1); i < lab6.*H* \* n; i++)
249. **for** (**int** j = 0; j < lab6.*N*; j++) {
250. **for** (**int** k = 0; k < lab6.*N*; k++)
251. lab6.*MA*[i][j] += MBi[i][k] \* lab6.*MC*[k][j];
252. lab6.*MA*[i][j] = ai \* (lab6.*MA*[i][j] - lab6.*ME*[i][j]);
253. }
254. // 12
255. box2.Signal\_MA();
256. System.*out*.println("Task " + n + " finished");
257. }
258. **private** BOX1 box1;
259. **private** BOX2 box2;
260. **private** **int** n;
261. }
262. **public** **class** TaskPm1 **extends** Thread {
263. **public** TaskPm1(BOX1 box1, BOX2 box2) {
264. **this**.box1 = box1;
265. **this**.box2 = box2;
266. }
267. **public** **void** run() {
268. System.*out*.println("Task P-1 Started");
269. // 1
270. **for** (**int** i = 0; i < lab6.*N*; i++)
271. **for** (**int** j = 0; j < lab6.*N*; j++)
272. lab6.*MC*[i][j] = 1;
273. // 2
274. box2.Signal\_Input();
275. // 3
276. box1.Wait\_Z();
277. // 4
278. **int** aPm1 = -429496729;
279. **for** (**int** i = lab6.*H* \* (lab6.*P* - 2); i < lab6.*H* \* (lab6.*P* - 1); i++)
280. **if** (lab6.*Z*[i] > aPm1)
281. aPm1 = lab6.*Z*[i];
282. // 5
283. box1.max(aPm1);
284. // 6
285. box1.Signal\_a();
286. // 7
287. box2.Wait\_Input();
288. // 8
289. **int**[][] MBPm1 = box2.Copy\_MB();
290. // 9
291. box1.Wait\_a();
292. // 10
293. aPm1 = box1.Copy\_a();
294. // 11
295. **for** (**int** i = lab6.*H* \* (lab6.*P* - 2); i < lab6.*H* \* (lab6.*P* - 1); i++)
296. **for** (**int** j = 0; j < lab6.*N*; j++) {
297. **for** (**int** k = 0; k < lab6.*N*; k++)
298. lab6.*MA*[i][j] += MBPm1[i][k] \* lab6.*MC*[k][j];
299. lab6.*MA*[i][j] = aPm1 \* (lab6.*MA*[i][j] - lab6.*ME*[i][j]);
300. }
301. // 12
302. box2.Signal\_MA();
303. System.*out*.println("Task P-1 finished");
304. }
305. **private** BOX1 box1;
306. **private** BOX2 box2;
307. }
308. **public** **class** TaskP **extends** Thread {
309. **public** TaskP(BOX1 box1, BOX2 box2) {
310. **this**.box1 = box1;
311. **this**.box2 = box2;
312. }
313. **public** **void** run() {
314. System.*out*.println("Task P Started");
315. // 1
316. **for** (**int** i = 0; i < lab6.*N*; i++)
317. **for** (**int** j = 0; j < lab6.*N*; j++)
318. lab6.*ME*[i][j] = 1;
319. // 2
320. box2.Signal\_Input();
321. // 3
322. box1.Wait\_Z();
323. // 4
324. **int** aP = -429496729;
325. **for** (**int** i = lab6.*H* \* (lab6.*P* - 1); i < lab6.*N*; i++)
326. **if** (lab6.*Z*[i] > aP)
327. aP = lab6.*Z*[i];
328. // 5
329. box1.max(aP);
330. // 6
331. box1.Signal\_a();
332. // 7
333. box2.Wait\_Input();
334. // 8
335. **int**[][] MBP = box2.Copy\_MB();
336. // 9
337. box1.Wait\_a();
338. // 10
339. aP = box1.Copy\_a();
340. // 11
341. **for** (**int** i = lab6.*H* \* (lab6.*P* - 1); i < lab6.*N*; i++)
342. **for** (**int** j = 0; j < lab6.*N*; j++) {
343. **for** (**int** k = 0; k < lab6.*N*; k++)
344. lab6.*MA*[i][j] += MBP[i][k] \* lab6.*MC*[k][j];
345. lab6.*MA*[i][j] = aP \* (lab6.*MA*[i][j] - lab6.*ME*[i][j]);
346. }
347. // 12
348. box2.Signal\_MA();
349. System.*out*.println("Task P finished");
350. }
351. **private** BOX1 box1;
352. **private** BOX2 box2;
353. }