# РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ ПРГ2 ДЛЯ ПКС ЛП

В роздіді розглянуто розробку та дослідження програми ПРГ2 для ПКС з ЛП.

Математична задача: A = (B\*C)\*E + α\*K\*(MO\*MT)



Мова програмування: Ада

Засоби взаємодії: Ада Рандеву

Структура ПКС ЛП зображена на рис. 3.1. Для отримання індексу потоку Tid з індексів матриці (i,j), де i = 1..P/2, j = 1..2 , потрібно: Tid = i + (j-1)\*D



Рис. 3.1 Структура ПКС ЛП.



* 1. **Розробка паралельного математичного алгоритму**

Згідно з технічним завданням необхідно розробити паралельний алгоритм. Його можна розділити на наступні етапи:

1. , i = 1..P



СР: a, d, K,MO.

Пояснення до використовуваних констант:

*  – розмірність векторів і матриць;
*  – кількість ядер;
* .



* 1. **Розробка алгоритмів процесів**

**Задача :**



1. Введення *E, B, α, MT*.
2. Передати *E, B, α, MT* в *T2,1*.



1. Передати *E, B, α, MT* в *T1,2*.
2. Прийняти від .



1. Передати в .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Передати в .



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Прийняти від .



1. Виведення *A*.

**Задача :**



1. Прийняти *E, B, α, MT* вiд *Tj-1,1*.



1. Передати *E, B, α, MT* в *Tj+1,1*.



1. Передати *E, B, α, MT* в *Tj,2*.
2. Прийняти від



1. Передати в *Tj-1,1*.



1. Передати в .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Передати в .



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Прийняти від .



1. Передати в .



**Задача**



1. Прийняти *E, B, α, MT* вiд *Tj-1,1*.



1. Передати *E, B, α, MT* в *Tj+1,1*.



1. Передати *E, B, α, MT* в *Tj,2*.
2. Прийняти від



1. Передати в *Tj-1,1*.



1. Передати в .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Передати в .



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Прийняти від .



1. Передати в .



**Задача :**



1. Прийняти *E, B, α, MT* вiд *Tj-1,1*.



1. Передати *E, B, α, MT* в *Tj+1,1*.



1. Передати *E, B, α, MT* в *Tj,2*.
2. Прийняти від



1. Передати в *Tj-1,1*.



1. Передати в .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Передати в .



1. Передати в .



1. Передати в .



1. Обчислення



1. Прийняти від .



1. Прийняти від .



1. Передати в .



**Задача**



1. Прийняти *E, B, α, MT* вiд *Tj,1*



1. Прийняти від



1. Обчислення



1. Передати в .



1. Прийняти від



1. Обчислення



1. Прийняти від .



1. Передати в .



**Задача :**



1. Прийняти *E, B, α, MT* від



1. Передати *E, B, α, MT* в *TD,2*.
2. Введення



1. Передати в .



1. Передати в .



1. Обчислення



1. Прийняти від .



1. Обчислення



1. Передати в .



1. Прийняти від .



1. Обчислення



1. Прийняти від .



1. Передати в .



* 1. **Розробка схеми взаємодії процесів**

На основі алгоритму для всіх задач розроблено структурну схему взаємодії задач (рис. 3.2). За допомогою цієї схеми можна наочно спостерігати як саме відбувається пересилка даних за допомогою повідомлень.

Усі задачі логічно розбиваються на декілька типів, як видно з алгоритмів в розділі 3.2.

Розбиття на типи гарантує, що вирішення задачі буде найбільш оптимальним і дорівнюватиме діаметру системи.

Задачі реалізовані у вигляді задачного типу, що об’єднує реалізацію усіх логічних типів, та має входи:

* *DataInput1* – вхід для приймання/передачі *B, E, a, MT*.
* *DataInput2* – вхід для приймання/передачі .



* *DataResult\_di* – вхід для приймання/передачі проміжних значень .



* *DataResult\_d* – вхід для приймання/передачі значень .



* *Result\_A* – вхід для приймання/передачі .



?



Рис. 3.2 Схема взаємодії процесів



* 1. **Розробка програми ПРГ2**

Програма ПРГ2 згідно технічного завдання розроблена на мові програмування Ада.

Програма складається з одніє головної процедури Prg2. Вона в собі містить:

* Визначення констант *N, P, H, D, S*.
* Визначення безрозмірних типів вектора Vector та матриці Matrix.
* Процедури введення/виведення вектора та матриці.
* Процедура обчислення .



* Специфікація та реалізація задачного типу.
* Тип вказівника на задачі для динамічної ініціалізації задач.
* Тип масиву вказівників на задачі.
* Тип матриці вказівників на задачі.

Лістинг розробленої програми наведено у додатку В.

Алгоритм роботи задач наведено у додатку Г.

?

**ДОДАТКИ**

**Додаток В**

**Додаток А**

**Лістинг програми ПРГ2**

Source file: ..\kr.adb Fri Apr 23 07:48:40 2015

1 with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;

2 use Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;

3

4 ---------------------------------------------------------------

5 -- Kurs project 3

6 -- A = (B\*C) \* E + alpha \* K \* (MO \* MT)

7 -- Budyashevskiy Andriy IO-22

8 -- 23.04.15

9 ---------------------------------------------------------------

10 procedure KR3LM is

11 N: Integer := 16;

12 P: Integer := 8;

13 D: Integer := P/2;

14 H: Integer := N / P;

15 S: Integer := D / 2 + 1;

16

17 -- StartTime, FinishTime: Time;

18 -- DiffTime: Duration;

19

20 type Vector is array (1..N) of Integer;

21 type Matrix is array (1..N) of Vector;

22

23 ---------------------------------------------------------------

24 -- Enter Vector

25 procedure VectorInput(M: out Vector) is

26 begin

27 for i in 1..N loop

28 M(i) := 1;

29 end loop;

30 end VectorInput;

31

32 procedure VectorOutput(M: in Vector) is

33 begin

34 if ( N < 20 ) then

35 for i in 1..N loop

36 Put(M(i));

37 end loop;

38 New\_Line;

39 end if;

40 end VectorOutput;

41

42

43 procedure MatrixInput(MA: out Matrix) is

44 begin

45 for i in 1..N loop

46 for j in 1..N loop

47 MA(i)(j) := 1;

48 end loop;

49 end loop;

50 end MatrixInput;

51

52 procedure CalculateA(start, endd, d, alpha: Integer; K,E: in Vector; MO, MT: in Matrix; A: out Vector) is

53 MR:Matrix;

54 begin

55 for i in start..endd loop

56 A(i):=0;

57 for j in 1..N loop

58 MR(i)(j):=0;

59 for k in 1..N loop

60 MR(i)(j):= MR(i)(j)+ MT(i)(k) \* MO(k)(j);

61 end loop;

62 A(i):=A(i) + alpha \* K(j) \* MR(i)(j);

63 end loop;

64 A(i):=A(i) + d \* E(i);

65 end loop;

66 end CalculateA;

67

68

69 ---------------------------------------------------------------

70 -- Universal task T\_i

71 task type Task\_I(Ti, Tj: Integer) is

72 entry DataInput1(B,E: in Vector; MT: in Matrix; alpha : Integer);

73 entry DataInput2(C, K: in Vector; MO: in Matrix);

74

75 entry DataResult\_di(dii: in Integer);

76 entry DataResult\_d(d: in Integer);

77

78 entry Result\_A(A: in Vector);

79 end Task\_I;

80

81 ---------------------------------------------------------------

82 -- pointer

83 type Task\_I\_Ptr is access Task\_I;

84

85 -- array of pointers

86 type Tasks\_Array is array (1..2) of Task\_I\_Ptr;

87 type Tasks is array (1..D) of Tasks\_Array;

88

89 tasksAll: Tasks;

90

91 task body Task\_I is

92 start: Integer := (Ti+ (Tj-1)\*D - 1) \* H + 1;

93 endd: Integer := (Ti+ (Tj-1)\*D) \* H;

94

95 Ai, Ci, Bi, Ki, Ei: Vector;

96

97 MOi, MTi: Matrix;

98 di, alphai: Integer;

99

100 begin

101 Put\_Line("task" & Integer'Image(Ti+ (Tj-1)\*D) & " started!");

102

103 if Tj = 1 then

104 -- Entering and reading all T1 data

105 if Ti = 1 then

106 VectorInput(Bi);

107 VectorInput(Ei);

108 MatrixInput(MTi);

109 alphai := 1;

110 else

111 accept DataInput1(B, E: in Vector; MT :in Matrix; alpha: in Integer) do

112 Bi := B;

113 Ei := E;

114 MTi := MT;

115 alphai:=alpha;

116 end DataInput1;

117 end if;

118

119 if Ti < D then

120 tasksAll(Ti +1)(Tj).DataInput1(Bi, Ei, MTi, alphai);

121 end if;

122

123 tasksAll(Ti)(Tj + 1).DataInput1(Bi, Ei, MTi, alphai);

124 --Entering and reading all TP data

125

126 if(Ti = D) then

127 VectorInput(Ci);

128 VectorInput(Ki);

129 MatrixInput(MOi);

130 alphai := 1;

131

132 else

133

134 accept DataInput2(C, K: in Vector; MO :in Matrix) do

135 Ci := C;

136 Ki := K;

137 MOi := MO;

138 end DataInput2;

139 end if;

140

141 if Ti > 1 then

142 tasksAll(Ti -1)(Tj).DataInput2(Ci, Ki, MOi);

143 end if;

144 tasksAll(Ti)(Tj +1).DataInput2(Ci, Ki, MOi);

145

146 else

147

148 accept DataInput1(B, E: in Vector; MT :in Matrix; alpha: in Integer) do

149 Bi := B;

150 Ei := E;

151 MTi := MT;

152 alphai:=alpha;

153 end DataInput1;

154 accept DataInput2(C, K: in Vector; MO :in Matrix) do

155 Ci := C;

156 Ki := K;

157 MOi := MO;

158 end DataInput2;

159

160 end if;

161

162 --di = B\*C

163 di:=0;

164 for i in start..endd loop

165 di:=di+Bi(i)\*Ci(i);

166 end loop;

167

168 --add bottom di-s to the top

169 if(Tj = 2) then

170 tasksAll(Ti)(Tj -1).DataResult\_di(di);

171 accept DataResult\_d (d : in Integer) do

172 di:= d;

173 end DataResult\_d;

174 else

175 accept DataResult\_di (dii : in Integer) do

176 di:=di + dii;

177 end DataResult\_di;

178 if(Ti > 1 and Ti < D) then

179 accept DataResult\_di (dii : in Integer) do

180 di:=di + dii;

181 end DataResult\_di;

182 end if;

183 if (Ti<S)then

184 tasksAll(Ti+1)(Tj).DataResult\_di(di);

185 elsif (Ti = S) then

186 accept DataResult\_di (dii : in Integer) do

187 di:=di + dii;

188 end DataResult\_di;

189 tasksAll(Ti+1)(Tj).DataResult\_d(di);

190 tasksAll(Ti-1)(Tj).DataResult\_d(di);

191 else

192 tasksAll(Ti-1)(Tj).DataResult\_di(di);

193 end if;

194

195 if(Ti /= S) then

196 accept DataResult\_d(d : in Integer) do

197 di:= d;

198 end DataResult\_d;

199 end if;

200 if (Ti<S ) then

201 if (Ti /=1) then

202 tasksAll(Ti-1)(Tj).DataResult\_d(di);

203 end if;

204 else

205 if (Ti /=D and Ti /=S) then

206 tasksAll(Ti+1)(Tj).DataResult\_d(di);

207 end if;

208 end if;

209

210 tasksAll(Ti)(Tj +1).DataResult\_d(di);

211

212

213 end if;

214 -- Calculatin

215 CalculateA(start, endd,di, alphai, Ki, Ei, MOi, MTi, Ai);

216 ---Put\_Line(" jj= " &Integer'Image(Ti+ (Tj-1)\*D)& "value" & Integer'Image(alphai));

217 if(Tj = 2) then

218 tasksAll(Ti)(Tj -1).Result\_A(Ai);

219 else

220 accept Result\_A (A : in Vector) do

221 for i in 1..N loop

222 Ai(i) := Ai(i) + A(i);

223 end loop;

224

225 end Result\_A;

226 --Put\_Line(" final= " &Integer'Image(Ti+ (Tj-1)\*D)& " value" & Integer'Image(Ai(Ti+ (Tj-1)\*D)));

227

228

229 if(Ti < D) then

230 accept Result\_A (A : in Vector) do

231 for i in 1..N loop

232 Ai(i) := Ai(i) + A(i);

233 end loop;

234 end Result\_A;

235 end if;

236 if(Ti > 1) then

237 tasksAll(Ti - 1)(Tj).Result\_A(Ai);

238 end if;

239

240 end if;

241 if (Ti = 1 and Tj = 1) then

242 VectorOutput(Ai);

243 end if;

244 end Task\_I;

245

246

247

248

249 ---------------------------------------------------------------

250 begin

251 --StartTime := Clock;

252

253

254 for j in 1..2 loop

255 for i in 1..D loop

256 if i > 1 or j > 1 then

257

258 tasksAll(i)(j) := new Task\_I(i, j);

259

260 end if;

261 end loop;

262 end loop;

263 tasksAll(1)(1) := new Task\_I(1, 1);

264 end KR3LM;

265

**Додаток Г**



Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

*НТУУ-КПІ ОТ* 6.050102



Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

*НТУУ-КПІ ОТ* 6.050102



Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

*НТУУ-КПІ ОТ* 6.050102