Национальный технический университет Украины

«Киевский политехнический институт»

Лабораторная работа №4

По курсу

Параллельные и распределенные вычисления

Тема:

Ада. Защищенный модуль

Работу выполнил

Студент 3-го курса ФИВТ

Группы ИВ-73

Грубый Павел

Киев-2010

Техническое задание



MA = a×(MB×(MZ + MO×MF)

Этап 1. Построение параллельного алгоритма

MAH = a×(MB×(MZH + MO×MFH)

ОР: a, MB, MO

Этап 2. Разработка алгоритмов работы каждого процесса

|  |  |
| --- | --- |
| **Задача Т1** | Критические участки |
| 1. Ввод a, MB, MZ, MO, MF. |  |
| 1. Сигнал задачам Т2, T3, T4 о завершении ввода a, MB, MZ, MO, MF. |  |
| 1. Копия a1 := a | КУ |
| 1. Копия MB1 := MB | КУ |
| 1. Копия MO1 := MO | КУ |
| 1. Вычисление MAH = a1×(MB1×(MZH + MO1×MFH) |  |
| 1. Сигнал задачe Т3 о завершении вычислений. |  |

|  |  |
| --- | --- |
| **Задача Т2** | Критические участки |
| 1. Ожидание ввода a, MB, MZ, MO, MF в задаче Т1. |  |
| 1. Копия a2 := a | КУ |
| 1. Копия MB2 := MB | КУ |
| 1. Копия MO2 := MO | КУ |
| 1. Вычисление MAH = a2×(MB2×(MZH + MO2×MFH) |  |
| 1. Сигнал задачe Т3 о завершении вычислений. |  |

|  |  |
| --- | --- |
| **Задача Т3** | Критические участки |
| 1. Ожидание ввода a, MB, MZ, MO, MF в задаче Т1. |  |
| 1. Копия a3 := a | КУ |
| 1. Копия MB3 := MB | КУ |
| 1. Копия MO3 := MO | КУ |
| 1. Вычисление MAH = a3×(MB3×(MZH + MO3×MFH) |  |
| 1. Ожидание сигнала от задач Т1, Т2, Т4 о завершении вычислений. |  |
| 1. Вывод MA. |  |

|  |  |
| --- | --- |
| **Задача Т4** | Критические участки |
| 1. Ожидание ввода a, MB, MZ, MO, MF в задаче Т1. |  |
| 1. Копия a4 := a | КУ |
| 1. Копия MB4 := MB | КУ |
| 1. Копия MO4 := MO | КУ |
| 1. Вычисление MAH = a4×(MB4×(MZH + MO4×MFH) |  |
| 1. Сигнал задачe Т3 о завершении вычислений. |  |

Этап 3. Разработка структурной схемы взаимодействия задач



рис. 3.1. Схема взаимодействия задач.

Этап 4. Разработка программы

GNAT GPL 2009 (20090519)

Copyright 1992-2009, Free Software Foundation, Inc.

Compiling: D:\Kache\AdaWorkspace\AdaProtectedUnit\lab4.adb (source file time stamp: 2010-03-13 13:39:02)

1. ----------------------------------------------------------------

2. -- Paralel and distributed computing --

3. -- Laboratory work #4. Ada. Protected unit --

4. -- Func: MA = a\*(MB\*(MZ + MO\*MF) --

5. -- IO-73 Grubyi Pavel --

6. -- 17.03.2010 --

7. ----------------------------------------------------------------

8.

9. with Ada.Text\_Io, Ada.Integer\_Text\_Io;

10. use Ada.Text\_Io, Ada.Integer\_Text\_Io;

11.

12. procedure lab4 is

13. N : integer := 8; --size of structures

14. P : integer := 4; --count of processors

15. H : integer := N / P; --size of piece

16.

17. type TVector is array (1 .. N) of integer;

18. type TMatrix is array (1 .. N) of TVector;

19.

20. MA, MZ, MF : TMatrix;

21.

22. procedure VectorInput (vec : out TVector; x : in integer);

23.

24. procedure VectorOutput (vec : in TVector);

25.

26. procedure MatrixInput (m : out TMatrix; x : in integer);

27.

28. procedure MatrixOutput (m : in TMatrix);

29.

30. -----------------------------------------

31. protected CalcSync is

32. entry WaitForCalc;

33. procedure CalcSignal;

34.

35. private

36. F : integer := 0;

37. end CalcSync;

38.

39. protected body CalcSync is

40.

41. entry WaitForCalc when F >= 3 is

42. begin

43. if N<=8 then

44. MatrixOutput (MA);

45. end if;

46. end WaitForCalc;

47.

48. procedure CalcSignal is

49. begin

50. F := F + 1;

51. end CalcSignal;

52.

53. end CalcSync;

54. ------------------------------------------

55. ------------------------------------------

56. protected InputSync is

57. entry WaitForInput;

58. procedure InputSignal;

59.

60. private

61. F : integer := 0;

62. end InputSync;

63.

64. protected body InputSync is

65.

66. entry WaitForInput when F >= 1 is

67. begin

68. null;

69. end WaitForInput;

70.

71. procedure InputSignal is

72. begin

73. F := F + 1;

74. end InputSignal;

75.

76. end InputSync;

77. ------------------------------------------

78. ------------------------------------------

79. protected Res\_a is

80. procedure seta (x : in integer);

81. function geta return integer;

82.

83. private

84. a : integer;

85. end Res\_a;

86.

87. protected body Res\_a is

88.

89. procedure seta (x : in integer) is

90. begin

91. a := x;

92. end;

93.

94. function geta return integer is

95. begin

96. return a;

97. end;

98.

99. end Res\_a;

100. ------------------------------------------

101. ------------------------------------------

102. protected Res\_MB is

103. procedure setMB (x : in integer);

104. function getMB return TMatrix;

105.

106. private

107. MB : TMatrix;

108. end Res\_MB;

109.

110. protected body Res\_MB is

111.

112. procedure setMB (x : in integer) is

113. begin

114. MatrixInput (MB, x);

115. end;

116.

117. function getMB return TMatrix is

118. begin

119. return MB;

120. end;

121.

122. end Res\_MB;

123. ------------------------------------------

124. ------------------------------------------

125. protected Res\_MO is

126. procedure setMO (x : in integer);

127. function getMO return TMatrix;

128.

129. private

130. MO : TMatrix;

131. end Res\_MO;

132.

133. protected body Res\_MO is

134.

135. procedure setMO (x : in integer) is

136. begin

137. MatrixInput (MO, x);

138. end;

139.

140. function getMO return TMatrix is

141. begin

142. return MO;

143. end;

144.

145. end Res\_MO;

146. ------------------------------------------

147.

148. procedure Process (id, a : integer; MO, MB : in TMatrix) is

149. MX : TMatrix;

150. begin

151. for j in ((H \* id)+1) .. (H \* (id + 1)) loop

152. for i in 1 .. N loop

153. MX (i) (j) := 0;

154. --put(MX(i)(j));

155. -- MO\*MF

156. for k in 1 .. N loop

157. MX (i) (j) := MX (i) (j) + MO (i) (k) \* MF (K) (J);

158. end loop;

159. -- MZ + MO\*MF

160. MX (i) (j) := MX (i) (j) + MZ (i) (j);

161. end loop;

162. end loop;

163.

164. for j in ((H \* id)+1) .. (H \* (id + 1)) loop

165. for i in 1 .. N loop

166. MA (i) (j) := 0;

167. -- MB\*(MZ + MO\*MF)

168. for k in 1 .. N loop

169. MA (i) (j) := MA (i) (j) + MB (i) (k) \* MX (K) (J);

170. end loop;

171. -- a\*(MB\*(MZ + MO\*MF))

172. MA (i) (j) := MA (i) (j) \* a;

173. end loop;

174. end loop;

175. end Process;

176.

177. -----------------------------------------------------

178. task T1;

179. task body T1 is

180. MB1, MO1 : TMatrix;

181. a1 : integer;

182. id : integer := 0;

183. begin

184. Put\_Line ("T1: start");

185.

186. Res\_a.seta (1);

187. Res\_MB.setMB (1);

188. Res\_MO.setMO (1);

189. MatrixInput (MZ, 1);

190. MatrixInput (MF, 1);

191. InputSync.InputSignal;

192. Put\_Line ("T1: input finished");

193.

194. a1 := Res\_a.geta;

195. MB1 := Res\_MB.getMB;

196. MO1 := Res\_MO.getMO;

197. Put\_Line ("T1: copy finished");

198.

199. Process (id, a1, MO1, MB1);

200. Put\_Line ("T1: calc ended");

201. CalcSync.CalcSignal;

202.

203. Put\_Line ("T1: finished");

204. end T1;

205.

206. -----------------------------------------------------

207. task T2;

208. task body T2 is

209. MB2, MO2 : TMatrix;

210. a2 : integer;

211. id : integer := 1;

212. begin

213. Put\_Line ("T2: start");

214.

215. InputSync.WaitForInput;

216.

217. a2 := Res\_a.geta;

218. MO2 := Res\_MO.getMO;

219. MB2 := Res\_MB.getMB;

220. Put\_Line ("T2: copy finished");

221.

222. Process (id, a2, MO2, MB2);

223. Put\_Line ("T2: calc ended");

224. CalcSync.CalcSignal;

225.

226. Put\_Line ("T2: finished");

227. end T2;

228.

229. -----------------------------------------------------

230. task T3;

231. task body T3 is

232. MB3, MO3 : TMatrix;

233. a3 : integer;

234. id : integer := 2;

235. begin

236. Put\_Line ("T3: start");

237.

238. InputSync.WaitForInput;

239.

240. MB3 := Res\_MB.getMB;

241. a3 := Res\_a.geta;

242. MO3 := Res\_MO.getMO;

243. Put\_Line ("T3: copy finished");

244.

245. Process (id, a3, MO3, MB3);

246. Put\_Line ("T3: calc ended");

247. CalcSync.WaitForCalc;

248. Put\_Line ("T3: finished");

249. end T3;

250.

251. -----------------------------------------------------

252. task T4;

253. task body T4 is

254. MB4, MO4 : TMatrix;

255. a4 : integer;

256. id : integer := 3;

257. begin

258. Put\_Line ("T4: start");

259.

260. InputSync.WaitForInput;

261.

262. MO4 := Res\_MO.getMO;

263. a4 := Res\_a.geta;

264. MB4 := Res\_MB.getMB;

265. Put\_Line ("T4: copy finished");

266.

267. Process (id, a4, MO4, MB4);

268. Put\_Line ("T4: calc ended");

269. CalcSync.CalcSignal;

270.

271. Put\_Line ("T4: finished");

272. end T4;

273. -----------------------------------------------------

274.

275. procedure VectorInput (vec : out TVector; x : in integer) is

276. begin

277. for i in 1 .. N loop

278. vec (i) := x;

279. end loop;

280. end VectorInput;

281.

282. procedure VectorOutput (vec : in TVector) is

283. begin

284. for i in 1 .. N loop

285. put (vec (i), 4);

286. end loop;

287. new\_line;

288. end VectorOutput;

289.

290. procedure MatrixInput (m : out TMatrix; x : in integer) is

291. begin

292. for i in 1 .. N loop

293. VectorInput (m (i), x);

294. end loop;

295. end MatrixInput;

296.

297. procedure MatrixOutput (m : in TMatrix) is

298. begin

299. for i in 1 .. N loop

300. VectorOutput (m (i));

301. new\_line;

302. end loop;

303. end MatrixOutput;

304.

305. begin

306. null;

307. end lab4;

308.

308 lines: No errors, no warnings