Kpi-best

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 № 8

з курсу “Паралельні та розподілені обчислення - 2” на тему:

“Програмування для ПКС з ЛП. Бібліотека MPI”

Виконав: студент ІІІ курсу групи ІО‑01

Корнієнко Дмитро

Перевірив: доцент кафедри ОТ Корочкін О.В.

Київ – 2013

Технічне завдання

Розробити програму для розв’язання в ПКС з ЛП (структура на рис. Рисунок 1) математичної задачі: .



Рисунок – Структура ПКС з ЛП

Виконання роботи

Етап 1. Побудова паралельного алгоритму

1. ,

;

1. .

Етап 2. Розроблення алгоритмів роботи кожного процесу

Задача T1

1. Прийняти  від задачі T4.
2. Передати  задачі T2.
3. Прийняти  від задачі T2.
4. Обчислення .
5. Передати  задачі T2.
6. Прийняти  від задачі T2.
7. Обчислення .
8. Передати  задачі T2.

Задача T2

1. Прийняти  від задачі T1.
2. Передати  задачі T3.
3. Прийняти  від задачі T3.
4. Передати  задачі T1.
5. Обчислення .
6. Прийняти  від задачі T1.
7. Обчислення .
8. Прийняти  від задачі T3.
9. Обчислення .
10. Передати  задачі T5.
11. Прийняти  від задачі T5.
12. Передати  задачі T1.
13. Передати  задачі T3.
14. Обчислення .
15. Прийняти  від задачі T1.
16. Передати  задачі T3.

Задача T3

1. Прийняти  від задачі T6.
2. Прийняти  від задачі T2.
3. Передати  задачі T2.
4. Обчислення .
5. Передати  задачі T2.
6. Прийняти  від задачі T3.
7. Обчислення .
8. Прийняти *МА2Н* від задачі Т2.
9. Передати  задачі T6.

Задача T4

1. Введення .
2. Передати  задачі T1.
3. Передати  задачі T7.
4. Передати  задачі T5.
5. Прийняти  від задачі T5.
6. Обчислення .
7. Передати  задачі T5.
8. Прийняти  від задачі T5.
9. Обчислення .
10. Передати задачі T5.

Задача T5

1. Прийняти  від задачі T4.
2. Передати  задачі T6.
3. Прийняти  від задачі T6.
4. Передати  задачі T4.
5. Обчислення .
6. Прийняти  від задачі T4.
7. Обчислення .
8. Прийняти  від задачі T6.
9. Обчислення .
10. Прийняти  від задачі T2.
11. Обчислення .
12. Прийняти  від задачі T8.
13. Обчислення .
14. Передати  задачі T2.
15. Передати  задачі T8.
16. Передати  задачі T4.
17. Передати  задачі T6.
18. Обчислення .
19. Прийняти  від задачі T4.
20. Передати  задачі T6.

Задача T6

1. Введення .
2. Передати  задачі T3.
3. Передати  задачі T9.
4. Прийняти  від задачі T5.
5. Передати  задачі T5.
6. Обчислення .
7. Передати  задачі T5.
8. Прийняти  від задачі T5.
9. Обчислення .
10. Прийняти  від задачі T5.
11. Прийняти  від задачі T3.
12. Прийняти  від задачі T9.
13. Виведення результату.

Задача T7

1. Прийняти  від задачі T4.
2. Передати  задачі T8.
3. Прийняти  від задачі T8.
4. Обчислення .
5. Передати  задачі T8.
6. Прийняти  від задачі T8.
7. Обчислення .
8. Передати  задачі T8.

Задача T8

1. Прийняти  від задачі T7.
2. Передати  задачі T9.
3. Прийняти  від задачі T9.
4. Передати  задачі T7.
5. Обчислення .
6. Прийняти  від задачі T7.
7. Обчислення .
8. Прийняти  від задачі T9.
9. Обчислення .
10. Передати  задачі T5.
11. Прийняти  від задачі T5.
12. Передати  задачі T7.
13. Передати  задачі T9.
14. Обчислення .
15. Прийняти  від задачі T7.
16. Передати  задачі T9.

Задача T9

1. Прийняти  від задачі T6.
2. Прийняти  від задачі T8.
3. Передати  задачі T8.
4. Обчислення .
5. Передати  задачі T8.
6. Прийняти  від задачі T8.
7. Обчислення .
8. Прийняти *МА2Н* від задачі Т8.
9. Передати  задачі T6.

Етап 3. Розроблення структурної схеми взаємодії задач



Рисунок – Структурна схема взаємодії задач

Етап 4. Розроблення програми

1. /\*\*

2. \* ------------------------------------------------------------------------

3. \* Паралельні та розподілені обчислення

4. \* Лабораторна робота №8. Бібліотека MPI

5. \*

6. \* Завдання: MA = max(MO)\*ME + alpha\*MK\*MR

7. \*

8. \* Виконав: Корнієнко Дмитро

9. \* Група: ІО-01

10. \* Дата: 06.06.13

11. \* ------------------------------------------------------------------------

12. \*/

13.

14. #define WORLD MPI\_COMM\_WORLD

15. #define IGNORE MPI\_STATUS\_IGNORE

16.

17. #include <iostream>

18. #include <mpi.h>

19. #include "functions.h"

20.

21. using namespace std;

22.

23. const int N = 9;

24. const int P = 9;

25. int H;

26.

27. int main(int args, char\* argv[]) {

28.

29. MPI\_Init(&args, &argv);

30. int rank;

31. MPI\_Comm\_rank(WORLD, &rank);

32. H = N / P;

33.

34. cout << "Task " << rank+1 << " started" << endl;

35.

36. int rowsME = H, rowsMA\_MO\_MK = H;

37.

38. switch(rank) {

39. case 0:

40. rowsME = 3\*H;

41. break;

42. case 1:

43. rowsMA\_MO\_MK = 2\*H;

44. rowsME = 2\*H;

45. break;

46. case 2:

47. rowsMA\_MO\_MK = 3\*H;

48. break;

49. case 3:

50. rowsME = N;

51. break;

52. case 4:

53. rowsMA\_MO\_MK = 2\*H;

54. rowsME = 2\*H;

55. break;

56. case 5:

57. rowsMA\_MO\_MK = N;

58. break;

59. case 6:

60. rowsME = 3\*H;

61. break;

62. case 7:

63. rowsMA\_MO\_MK = 2\*H;

64. rowsME = 2\*H;

65. break;

66. case 8:

67. rowsMA\_MO\_MK = 3\*H;

68. break;

69. }

70.

71. Matrix MA(rowsMA\_MO\_MK, N), ME(rowsME, N), MK(rowsMA\_MO\_MK, N), MO(rowsMA\_MO\_MK, N), MR(N);

72. int alpha;

73. int a = INT\_MIN;

74. int aBuf = INT\_MIN;

75.

76. switch(rank) {

77.

78. case 0:

79. // 1. Прийняти ME3H, MR від задачі T4

80. MPI\_Recv(ME.get\_adress(0), 3\*H\*N, MPI\_INT, 3, 0, WORLD, IGNORE);

81. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 3, 0, WORLD, IGNORE);

82.

83. // 2. Передати ME2H, MR2H задачі T2

84. MPI\_Send(ME.get\_adress(H\*N), 2\*H\*N, MPI\_INT, 1, 0, WORLD);

85. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 1, 0, WORLD);

86.

87. // 3. Прийняти alpha, MKH, MOH від задачі T2

88. MPI\_Recv(&alpha, 1, MPI\_INT, 1, 0, WORLD, IGNORE);

89. MPI\_Recv(MK.get\_adress(0), H\*N, MPI\_INT, 1, 0, WORLD, IGNORE);

90. MPI\_Recv(MO.get\_adress(0), H\*N, MPI\_INT, 1, 0, WORLD, IGNORE);

91.

92. // 4. Обчислення a=max(MOH)

93. a = maxValue(MO, 0, H);

94.

95. // 5. Передати a задачі T2

96. MPI\_Send(&a, 1, MPI\_INT, 1, 0, WORLD);

97.

98. // 6. Прийняти a від задачі T2

99. MPI\_Recv(&a, 1, MPI\_INT, 1, 0, WORLD, IGNORE);

100.

101. // 7. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

102. calculation(a, alpha, ME, MK, MR, H, 0, 0, MA);

103.

104. // 8. Передати MAH задачі T2

105. MPI\_Send(MA.get\_adress(0), H\*N, MPI\_INT, 1, 0, WORLD);

106.

107. break;

108.

109. case 1:

110.

111. // 1. Прийняти ME2H, MR від задачі T1

112. MPI\_Recv(ME.get\_adress(0), 2\*H\*N, MPI\_INT, 0, 0, WORLD, IGNORE);

113. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 0, 0, WORLD, IGNORE);

114.

115. // 2. Передати MEH, MR задачі T3

116. MPI\_Send(ME.get\_adress(H\*N), H\*N, MPI\_INT, 2, 0, WORLD);

117. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 2, 0, WORLD);

118.

119. // 3. Прийняти alpha, MK2H, MO2H від задачі T3

120. MPI\_Recv(&alpha, 1, MPI\_INT, 2, 0, WORLD, IGNORE);

121. MPI\_Recv(MK.get\_adress(0), 2\*H\*N, MPI\_INT, 2, 0, WORLD, IGNORE);

122. MPI\_Recv(MO.get\_adress(0), 2\*H\*N, MPI\_INT, 2, 0, WORLD, IGNORE);

123.

124. // 4. Передати alpha, MKH, MOH задачі T1

125. MPI\_Send(&alpha, 1, MPI\_INT, 0, 0, WORLD);

126. MPI\_Send(MK.get\_adress(0), H\*N, MPI\_INT, 0, 0, WORLD);

127. MPI\_Send(MO.get\_adress(0), H\*N, MPI\_INT, 0, 0, WORLD);

128.

129. // 5. Обчислення a=max(MOH)

130. a = maxValue(MO, H, 2\*H);

131.

132. // 6. Прийняти a1 від задачі T1

133. MPI\_Recv(&aBuf, 1, MPI\_INT, 0, 0, WORLD, IGNORE);

134.

135. // 7. Обчислення a=max(a, a1)

136. if(aBuf > a) {

137. a = aBuf;

138. }

139.

140. // 8. Прийняти a3 від задачі T3

141. MPI\_Recv(&aBuf, 1, MPI\_INT, 2, 0, WORLD, IGNORE);

142.

143. // 9. Обчислення a=max(a, a3)

144. if(aBuf > a) {

145. a = aBuf;

146. }

147.

148. // 10. Передати a задачі T5

149. MPI\_Send(&a, 1, MPI\_INT, 4, 0, WORLD);

150.

151. // 11. Прийняти a від задачі T5

152. MPI\_Recv(&a, 1, MPI\_INT, 4, 0, WORLD, IGNORE);

153.

154. // 12. Передати a задачі T1

155. MPI\_Send(&a, 1, MPI\_INT, 0, 0, WORLD);

156.

157. // 13. Передати a задачі T3

158. MPI\_Send(&a, 1, MPI\_INT, 2, 0, WORLD);

159.

160. // 14. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

161. calculation(a, alpha, ME, MK, MR, H, H, 0, MA);

162.

163. // 15. Прийняти MAH від задачі T1

164. MPI\_Recv(MA.get\_adress(0), H\*N, MPI\_INT, 0, 0, WORLD, IGNORE);

165.

166. // 16. Передати MA2H задачі T3

167. MPI\_Send(MA.get\_adress(0), 2\*H\*N, MPI\_INT, 2, 0, WORLD);

168.

169.

170. break;

171.

172. case 2:

173.

174. // 1. Прийняти alpha, MK3H, MO3H T6

175. MPI\_Recv(&alpha, 1, MPI\_INT, 5, 0, WORLD, IGNORE);

176. MPI\_Recv(MK.get\_adress(0), 3\*H\*N, MPI\_INT, 5, 0, WORLD, IGNORE);

177. MPI\_Recv(MO.get\_adress(0), 3\*H\*N, MPI\_INT, 5, 0, WORLD, IGNORE);

178.

179. // 2. Прийняти MEH, MR від задачі T2

180. MPI\_Recv(ME.get\_adress(0), H\*N, MPI\_INT, 1, 0, WORLD, IGNORE);

181. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 1, 0, WORLD, IGNORE);

182.

183. // 3. Передати alpha, MK2H, MO2H задачі T2

184. MPI\_Send(&alpha, 1, MPI\_INT, 1, 0, WORLD);

185. MPI\_Send(MK.get\_adress(0), 2\*H\*N, MPI\_INT, 1, 0, WORLD);

186. MPI\_Send(MO.get\_adress(0), 2\*H\*N, MPI\_INT, 1, 0, WORLD);

187.

188. // 4. Обчислення a=max(MOH)

189. a = maxValue(MO, 2\*H, 3\*H);

190.

191. // 5. Передати a задачі T2

192. MPI\_Send(&a, 1, MPI\_INT, 1, 0, WORLD);

193.

194. // 6. Прийняти a від задачі T2

195. MPI\_Recv(&a, 1, MPI\_INT, 1, 0, WORLD,IGNORE);

196.

197. // 7. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

198. calculation(a, alpha, ME, MK, MR, H, 2\*H, 0, MA);

199.

200. // 8. Прийняти MA2H від задачі T2

201. MPI\_Recv(MA.get\_adress(0), 2\*H\*N, MPI\_INT, 1, 0, WORLD, IGNORE);

202.

203. // 9. Передати MA3H задачі T6

204. MPI\_Send(MA.get\_adress(0), 3\*H\*N, MPI\_INT, 5, 0, WORLD);

205.

206. break;

207.

208. case 3:

209.

210. // 1. Введення ME, MR

211. ME.fill(1);

212. MR.fill(1);

213.

214. // 2. Передати ME3H, MR задачі T1

215. MPI\_Send(ME.get\_adress(0), 3\*H\*N, MPI\_INT, 0, 0, WORLD);

216. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 0, 0, WORLD);

217.

218. // 3. Передати ME3H, MR задачі T7

219. MPI\_Send(ME.get\_adress(6\*H\*N), 3\*H\*N, MPI\_INT, 6, 0, WORLD);

220. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 6, 0, WORLD);

221.

222. // 4. Передати ME2H, MR задачі T5

223. MPI\_Send(ME.get\_adress(4\*H\*N), 2\*H\*N, MPI\_INT, 4, 0, WORLD);

224. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 4, 0, WORLD);

225.

226. // 5. Прийняти alpha, MKH, MOH від задачі T5

227. MPI\_Recv(&alpha, 1, MPI\_INT, 4, 0, WORLD, IGNORE);

228. MPI\_Recv(MK.get\_adress(0), H\*N, MPI\_INT, 4, 0, WORLD, IGNORE);

229. MPI\_Recv(MO.get\_adress(0), H\*N, MPI\_INT, 4, 0, WORLD, IGNORE);

230.

231. // 6. Обчислення a=max(MOH)

232. a = maxValue(MO, 0, H);

233.

234. // 7. Передати a задачі T5

235. MPI\_Send(&a, 1, MPI\_INT, 4, 0, WORLD);

236.

237. // 8. Прийняти a від задачі T5

238. MPI\_Recv(&a, 1, MPI\_INT, 4, 0, WORLD, IGNORE);

239.

240. // 9. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

241. calculation(a, alpha, ME, MK, MR, H, 0, 3\*H, MA);

242.

243. // 10. Передати MAH задачі T5

244. MPI\_Send(MA.get\_adress(0), H\*N, MPI\_INT, 4, 0, WORLD);

245.

246. break;

247.

248. case 4:

249.

250. // 1. Прийняти ME2H, MR від задачі T4

251. MPI\_Recv(ME.get\_adress(0), 2\*H\*N, MPI\_INT, 3, 0, WORLD, IGNORE);

252. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 3, 0, WORLD, IGNORE);

253.

254. // 2. Передати MEH, MR задачі T6

255. MPI\_Send(ME.get\_adress(H\*N), H\*N, MPI\_INT, 5, 0, WORLD);

256. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 5, 0, WORLD);

257.

258. // 3. Прийняти alpha, MK2H, MO2H від задачі T6

259. MPI\_Recv(&alpha, 1, MPI\_INT, 5, 0, WORLD, IGNORE);

260. MPI\_Recv(MK.get\_adress(0), 2\*H\*N, MPI\_INT, 5, 0, WORLD, IGNORE);

261. MPI\_Recv(MO.get\_adress(0), 2\*H\*N, MPI\_INT, 5, 0, WORLD, IGNORE);

262.

263. // 4. Передати alpha, MKH, MOH задачі T4

264. MPI\_Send(&alpha, 1, MPI\_INT, 3, 0, WORLD);

265. MPI\_Send(MK.get\_adress(0), H\*N, MPI\_INT, 3, 0, WORLD);

266. MPI\_Send(MO.get\_adress(0), H\*N, MPI\_INT, 3, 0, WORLD);

267.

268. // 5. Обчислення a=max(MOH)

269. a = maxValue(MO, H, 2\*H);

270.

271. // 6. Прийняти a4 від задачі T4

272. MPI\_Recv(&aBuf, 1, MPI\_INT, 3, 0, WORLD, IGNORE);

273.

274. // 7. Обчислення a=max(a, a4)

275. if(aBuf > a) {

276. a = aBuf;

277. }

278.

279. // 8. Прийняти a6 від задачі T6

280. MPI\_Recv(&aBuf, 1, MPI\_INT, 5, 0, WORLD, IGNORE);

281.

282. // 9. Обчислення a=max(a, a6)

283. if(aBuf > a) {

284. a = aBuf;

285. }

286.

287. // 10. Прийняти a2 від задачі T2

288. MPI\_Recv(&aBuf, 1, MPI\_INT, 1, 0, WORLD, IGNORE);

289.

290. // 11. Обчислення a=max(a, a2)

291. if(aBuf > a) {

292. a = aBuf;

293. }

294.

295. // 12. Прийняти a6 від задачі T8

296. MPI\_Recv(&aBuf, 1, MPI\_INT, 7, 0, WORLD, IGNORE);

297.

298. // 13. Обчислення a=max(a, a8)

299. if(aBuf > a) {

300. a = aBuf;

301. }

302.

303. // 14. Передати a задачі T2

304. MPI\_Send(&a, 1, MPI\_INT, 1, 0, WORLD);

305.

306. // 15. Передати a задачі T8

307. MPI\_Send(&a, 1, MPI\_INT, 7, 0, WORLD);

308.

309. // 16. Передати a задачі T4

310. MPI\_Send(&a, 1, MPI\_INT, 3, 0, WORLD);

311.

312. // 17. Передати a задачі T6

313. MPI\_Send(&a, 1, MPI\_INT, 5, 0, WORLD);

314.

315. // 18. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

316. calculation(a, alpha, ME, MK, MR, H, H, 0, MA);

317.

318. // 19. Прийняти MAH від задачі T4

319. MPI\_Recv(MA.get\_adress(0), H\*N, MPI\_INT, 3, 0, WORLD, IGNORE);

320.

321. // 20. Передати MA2H задачі T6

322. MPI\_Send(MA.get\_adress(0), 2\*H\*N, MPI\_INT, 5, 0, WORLD);

323.

324. break;

325.

326. case 5:

327.

328. // 1. Введення alpha, MK, MO

329. alpha = 1;

330. MK.fill(1);

331. MO.fill(1);

332.

333. // 2. Передати alpha, MK3H, MO3H задачі T3

334. MPI\_Send(&alpha, 1, MPI\_INT, 2, 0, WORLD);

335. MPI\_Send(MK.get\_adress(0), 3\*H\*N, MPI\_INT, 2, 0, WORLD);

336. MPI\_Send(MO.get\_adress(0), 3\*H\*N, MPI\_INT, 2, 0, WORLD);

337. f

338. // 3. Передати alpha, MK3H, MO3H задачі T9

339. MPI\_Send(&alpha, 1, MPI\_INT, 8, 0, WORLD);

340. MPI\_Send(MK.get\_adress(6\*H\*N), 3\*H\*N, MPI\_INT, 8, 0, WORLD);

341. MPI\_Send(MO.get\_adress(6\*H\*N), 3\*H\*N, MPI\_INT, 8, 0, WORLD);

342.

343. // 4. Прийняти MEH, MR від задачі T5

344. MPI\_Recv(ME.get\_adress(0), H\*N, MPI\_INT, 4, 0, WORLD, IGNORE);

345. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 4, 0, WORLD, IGNORE);

346.

347. // 5. Передати alpha, MK2H, MO2H задачі T5

348. MPI\_Send(&alpha, 1, MPI\_INT, 4, 0, WORLD);

349. MPI\_Send(MK.get\_adress(3\*H\*N), 2\*H\*N, MPI\_INT, 4, 0, WORLD);

350. MPI\_Send(MO.get\_adress(3\*H\*N), 2\*H\*N, MPI\_INT, 4, 0, WORLD);

351.

352. // 6. Обчислення a=max(MOH)

353. a = maxValue(MO, 5\*H, 6\*H);

354.

355. // 7. Передати a задачі T5

356. MPI\_Send(&a, 1, MPI\_INT, 4, 0, WORLD);

357.

358. // 8. Прийняти a від задачі T5

359. MPI\_Recv(&a, 1, MPI\_INT, 4, 0, WORLD, IGNORE);

360.

361. // 9. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

362. calculation(a, alpha, ME, MK, MR, H, 5\*H, 0, MA);

363.

364. // 8. Прийняти MA2H від задачі T5

365. MPI\_Recv(MA.get\_adress(3\*H\*N), 2\*H\*N, MPI\_INT, 4, 0, WORLD, IGNORE);

366.

367. // 9. Прийняти MA3H від задачі T3

368. MPI\_Recv(MA.get\_adress(0), 3\*H\*N, MPI\_INT, 2, 0, WORLD, IGNORE);

369.

370. // 10. Прийняти MA3H від задачі T9

371. MPI\_Recv(MA.get\_adress(6\*H\*N), 3\*H\*N, MPI\_INT, 8, 0, WORLD, IGNORE);

372.

373. break;

374.

375. case 6:

376.

377. // 1. Прийняти ME3H, MR від задачі T4

378. MPI\_Recv(ME.get\_adress(0), 3\*H\*N, MPI\_INT, 3, 0, WORLD, IGNORE);

379. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 3, 0, WORLD, IGNORE);

380.

381. // 2. Передати ME2H, MR задачі T8

382. MPI\_Send(ME.get\_adress(H\*N), 2\*H\*N, MPI\_INT, 7, 0, WORLD);

383. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 7, 0, WORLD);

384.

385. // 3. Прийняти alpha, MK, MOH від задачі T8

386. MPI\_Recv(&alpha, 1, MPI\_INT, 7, 0, WORLD, IGNORE);

387. MPI\_Recv(MK.get\_adress(0), H\*N, MPI\_INT, 7, 0, WORLD, IGNORE);

388. MPI\_Recv(MO.get\_adress(0), H\*N, MPI\_INT, 7, 0, WORLD, IGNORE);

389.

390. // 4. Обчислення a=max(MOH)

391. a = maxValue(MO, 0, H);

392.

393. // 5. Передати a задачі T8

394. MPI\_Send(&a, 1, MPI\_INT, 7, 0, WORLD);

395.

396. // 6. Прийняти a від задачі T8

397. MPI\_Recv(&a, 1, MPI\_INT, 7, 0, WORLD, IGNORE);

398.

399. // 7. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

400. calculation(a, alpha, ME, MK, MR, H, 0, 0, MA);

401.

402. // 8. Передати MAH задачі T8

403. MPI\_Send(MA.get\_adress(0), H\*N, MPI\_INT, 7, 0, WORLD);

404.

405. break;

406.

407. case 7:

408.

409. // 1. Прийняти ME2H, MR від задачі T7

410. MPI\_Recv(ME.get\_adress(0), 2\*H\*N, MPI\_INT, 6, 0, WORLD, IGNORE);

411. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 6, 0, WORLD, IGNORE);

412.

413. // 2. Передати MEH, MR задачі T9

414. MPI\_Send(ME.get\_adress(H\*N), H\*N, MPI\_INT, 8, 0, WORLD);

415. MPI\_Send(MR.get\_adress(0), N\*N, MPI\_INT, 8, 0, WORLD);

416.

417. // 3. Прийняти alpha, MK2H, MO2H від задачі T9

418. MPI\_Recv(&alpha, 1, MPI\_INT, 8, 0, WORLD, IGNORE);

419. MPI\_Recv(MK.get\_adress(0), 2\*H\*N, MPI\_INT, 8, 0, WORLD, IGNORE);

420. MPI\_Recv(MO.get\_adress(0), 2\*H\*N, MPI\_INT, 8, 0, WORLD, IGNORE);

421.

422. // 4. Передати alpha, MKH, MOH задачі T7

423. MPI\_Send(&alpha, 1, MPI\_INT, 6, 0, WORLD);

424. MPI\_Send(MK.get\_adress(0), H\*N, MPI\_INT, 6, 0, WORLD);

425. MPI\_Send(MO.get\_adress(0), H\*N, MPI\_INT, 6, 0, WORLD);

426.

427. // 5. Обчислення a=max(MOH)

428. a = maxValue(MO, H, 2\*H);

429.

430. // 6. Прийняти a7 від задачі T7

431. MPI\_Recv(&aBuf, 1, MPI\_INT, 6, 0, WORLD, IGNORE);

432.

433. // 7. Обчислення a=max(a, a7)

434. if(aBuf > a) {

435. a = aBuf;

436. }

437.

438. // 8. Прийняти a9 від задачі T9

439. MPI\_Recv(&aBuf, 1, MPI\_INT, 8, 0, WORLD, IGNORE);

440.

441. // 9. Обчислення a=max(a, a9)

442. if(aBuf > a) {

443. a = aBuf;

444. }

445.

446. // 10. Передати a задачі T5

447. MPI\_Send(&a, 1, MPI\_INT, 4, 0, WORLD);

448.

449. // 11. Прийняти a від задачі T5

450. MPI\_Recv(&a, 1, MPI\_INT, 4, 0, WORLD, IGNORE);

451.

452. // 12. Передати a задачі T7

453. MPI\_Send(&a, 1, MPI\_INT, 6, 0, WORLD);

454.

455. // 13. Передати a задачі T9

456. MPI\_Send(&a, 1, MPI\_INT, 8, 0, WORLD);

457.

458. // 14. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

459. calculation(a, alpha, ME, MK, MR, H, H, 0, MA);

460.

461. // 15. Прийняти MAH від задачі T7

462. MPI\_Recv(MA.get\_adress(0), H\*N, MPI\_INT, 6, 0, WORLD, IGNORE);

463.

464. // 16. Передати MA2H задачі T9

465. MPI\_Send(MA.get\_adress(0), 2\*H\*N, MPI\_INT, 8, 0, WORLD);

466.

467.

468. break;

469.

470. case 8:

471.

472. // 1. Прийняти alpha, MK3H, MO3H від задачі T6

473. MPI\_Recv(&alpha, 1, MPI\_INT, 5, 0, WORLD, IGNORE);

474. MPI\_Recv(MK.get\_adress(0), 3\*H\*N, MPI\_INT, 5, 0, WORLD, IGNORE);

475. MPI\_Recv(MO.get\_adress(0), 3\*H\*N, MPI\_INT, 5, 0, WORLD, IGNORE);

476.

477. // 2. Прийняти MEH, MR від задачі T8

478. MPI\_Recv(ME.get\_adress(0), H\*N, MPI\_INT, 7, 0, WORLD, IGNORE);

479. MPI\_Recv(MR.get\_adress(0), N\*N, MPI\_INT, 7, 0, WORLD, IGNORE);

480.

481. // 3. Передати alpha, MK2H, MO2H задачі T8

482. MPI\_Send(&alpha, 1, MPI\_INT, 7, 0, WORLD);

483. MPI\_Send(MK.get\_adress(0), 2\*H\*N, MPI\_INT, 7, 0, WORLD);

484. MPI\_Send(MO.get\_adress(0), 2\*H\*N, MPI\_INT, 7, 0, WORLD);

485.

486. // 4. Обчислення a=max(MOH)

487. a = maxValue(MO, 2\*H, 3\*H);

488.

489. // 5. Передати a задачі T8

490. MPI\_Send(&a, 1, MPI\_INT, 7, 0, WORLD);

491.

492. // 6. Прийняти a від задачі T8

493. MPI\_Recv(&a, 1, MPI\_INT, 7, 0, WORLD, IGNORE);

494.

495. // 7. Обчислення MAH = a\*MEH + alpha\*MKH\*MR

496. calculation(a, alpha, ME, MK, MR, H, 2\*H, 0, MA);

497.

498. // 8. Прийняти MA2H від задачі T8

499. MPI\_Recv(MA.get\_adress(0), 2\*H\*N, MPI\_INT, 7, 0, WORLD, IGNORE);

500.

501. // 9. Передати MA3H задачі T6

502. MPI\_Send(MA.get\_adress(0), 3\*H\*N, MPI\_INT, 5, 0, WORLD);

503.

504. break;

505. }

506.

507. cout << "Task " << rank+1 << " finished" << endl;

508. MPI\_Finalize();

509.

510. if(rank == 5) {

511. // Виведення результату

512. cout << MA << endl;

513. system("pause");

514. }

515.

516. return 0;

517. }

1. #pragma once

2.

3. #include "matrix.h"

4.

5. // Повертає max(MT)

6. int maxValue(

7. const Matrix &MT,

8. size\_t start,

9. size\_t end);

10.

11.

12. // MAH = a\*MEH + alpha\*MKH\*MR

13. void calculation(

14. const int a,

15. const int alpha,

16. const Matrix &ME,

17. const Matrix &MK,

18. const Matrix &MR,

19. size\_t H,

20. size\_t offsMA,

21. size\_t offsME,

22. Matrix &MA);

1. #include "functions.h"

2.

3. int maxValue(

4. const Matrix &MO,

5. size\_t start,

6. size\_t end)

7. {

8. int result = INT\_MIN;

9. for(size\_t i = start; i < end; i++) {

10. for(size\_t j = 0; j < MO.cols; j++) {

11. int x = MO.get(i, j);

12. if(x > result) {

13. result = x;

14. }

15. }

16. }

17. return result;

18. }

19.

20. void calculation(

21. const int a,

22. const int alpha,

23. const Matrix &ME,

24. const Matrix &MK,

25. const Matrix &MR,

26. size\_t H,

27. size\_t offsMA,

28. size\_t offsME,

29. Matrix &MA)

30. {

31. int sum;

32. for(size\_t i = 0; i < H; i++) {

33. for(size\_t j = 0; j < MA.cols; j++) {

34. sum = 0;

35. for(size\_t k = 0; k < MA.cols; k++) {

36. sum += MK.get(i + offsMA, k) \* MR.get(k, j);

37. }

38. sum = a \* ME.get(i + offsME, j) + alpha \* sum;

39. MA.set(i + offsMA, j, sum);

40. }

41. }

42. }

1. #ifndef \_MATRIX\_H\_

2. #define \_MATRIX\_H\_ 1

3.

4. #include <assert.h>

5. #include <ostream>

6.

8. class Matrix {

9. public:

10. Matrix(size\_t rows, size\_t cols);

11. Matrix(size\_t N);

12. Matrix(Matrix const &other);

13. ~Matrix();

14.

15. int get(size\_t i, size\_t j) const {

16. assert(i < this->rows);

17. assert(j < this->cols);

18. return this->data[i \* cols + j];

19. }

20.

21. void set(size\_t i, size\_t j, int value) {

22. assert(i < this->rows);

23. assert(j < this->cols);

24. this->data[i \* cols + j] = value;

25. }

26.

27. void\* get\_adress(size\_t element) {

28. return this->data + element;

29. }

30.

31. void fill(int value);

32.

33. const size\_t rows;

34. const size\_t cols;

35.

36. private:

37. int \*data;

38. friend std::ostream &operator<<(std::ostream &ostr, const Matrix &m);

39. };

40. #endif

1. #include "matrix.h"

3.

4. Matrix::Matrix(size\_t N) :

5. rows(N),

6. cols(N),

7. data(new int[N \* N])

8. {}

10.

11. Matrix::Matrix(size\_t rows, size\_t cols) :

12. rows(rows),

13. cols(cols),

14. data(new int[rows \* cols])

15. {}

17.

18. Matrix::Matrix(const Matrix &other) :

19. rows(other.rows),

20. cols(other.cols),

21. data(new int[other.rows \* other.cols])

22. {

23. for(size\_t i = 0; i < this->rows \* this->cols; i++) {

24. this->data[i] = other.data[i];

25. }

26. }

27.

28. Matrix::~Matrix() {

29. delete[] data;

30. }

32.

33. void Matrix::fill(int value) {

34. for(size\_t i = 0; i < this->rows; i++) {

35. for(size\_t j = 0; j < this->cols; j++) {

36. this->data[i \* this->cols + j] = value;

37. }

38. }

39. }

40.

41.

42. std::ostream &operator<<(std::ostream &ostr, const Matrix &matrix) {

43. for(size\_t i = 0; i < matrix.rows; i++) {

44. for(size\_t j = 0; j < matrix.cols; j++) {

45. ostr << matrix.data[i \* matrix.cols + j] << " ";

46. }

47. ostr << std::endl;

48. }

49. return ostr;

50. }