

```
1 //-----
2 // 2D_Jacobi_parallel_strips.cpp
3 //-----
4 #if (defined __linux__ || (defined _AIX)
5     #include <sys/types.h>
6     #include <sys/stat.h>
7     #include <unistd.h>
8 #elif (defined _WIN32) || (defined _WIN64)
9     #include <conio.h>
10    #include <direct.h>
11 #endif
12
13 #include <mpi.h>
14 #include <stdlib.h>
15 #include <time.h>
16 #include <iostream>
17 using namespace std;
18
19 #define N 20
20 #define iterations 10000000
21 //-----
22 int main(int argc, char* argv[])
23 {
24     int i, j, k, rank, size, region, row_start, row_end;
25     double my_time, A[N + 2][N + 2], B[N + 2][N + 2];
26     MPI_Datatype my_type1, my_type2;
27
28     MPI_Init(&argc, &argv);
29     MPI_Barrier(MPI_COMM_WORLD);
30     my_time = -MPI_Wtime();
31
32     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
33     MPI_Comm_size(MPI_COMM_WORLD, &size);
34     if (rank == 0) {
35         printf("There are %d processes.\n", size);
36         fflush(stdout);
37     }
38
39     MPI_Type_vector(N + 2, 1, 1, MPI_DOUBLE, &my_type1);
40     MPI_Type_commit(&my_type1);
41     MPI_Type_vector(N, 1, 1, MPI_DOUBLE, &my_type2);
42     MPI_Type_commit(&my_type2);
43
44     region = N / size;
45     row_start = region * rank + 1;
46     row_end = row_start + region - 1;
47
48     if (N > 20 || N % size != 0) {
49         if (rank == 0)
50             printf("The number of processes should divide and be less than %d!\n", N);
51         goto end;
52     }
53
```

```
54 printf("Process %d: row_start %d, row_end %d.\n", rank, row_start, row_end);
55 fflush(stdout);
56
57 for (i = 0; i < N + 2; i++)
58     for (j = 0; j < N + 2; j++) {
59         A[i][j] = 0;
60         B[i][j] = 0;
61     }
62
63 if (rank == 0) {
64     srand((unsigned)time(NULL));
65     for (i = 0; i < N + 2; i++)
66         for (j = 0; j < N + 2; j++)
67             A[i][j] = rand() % 100;
68 }
69
70 if (rank == 0)
71     for (k = 1; k < size; k++)
72         for (i = 0; i < N + 2; i++)
73             MPI_Ssend(&A[i][0], N + 2, my_type1, k, k, MPI_COMM_WORLD);
74 else
75     for (i = 0; i < N + 2; i++)
76         MPI_Recv(&A[i][0], N + 2, my_type1, 0, rank, MPI_COMM_WORLD,
77                 MPI_STATUSES_IGNORE);
78
79 if (rank == 0) {
80     cout << "\nStarting values of a matrix A:\n";
81     for (i = 0; i < N + 2; i++) {
82         for (j = 0; j < N + 2; j++)
83             printf("%6.2f ", A[i][j]);
84         cout << endl;
85     }
86 }
87 fflush(stdout);
88
89 for (k = 0; k < iterations; k++) {
90     for (i = row_start; i <= row_end; i++)
91         for (j = 1; j < N + 1; j++)
92             B[i][j] = 0.25 * (A[i - 1][j] + A[i + 1][j] + A[i][j - 1] + A[i][j + 1]);
93
94     for (i = row_start; i <= row_end; i++)
95         for (j = 1; j < N + 1; j++)
96             A[i][j] = B[i][j];
97
98     // nasledovna komunikacia je separatne vykonavana pre parne a neparne
99     // horizontalne pasy, aby sa zabranilo deadlocku
100 if (rank % 2 == 1)
101     MPI_Ssend(&B[row_start][1], 1, my_type2, rank - 1, 0, MPI_COMM_WORLD);
102 else
103     if (rank < size - 1)
104         MPI_Recv(&A[row_end + 1][1], 1, my_type2, rank + 1, 0, MPI_COMM_WORLD,
105                 MPI_STATUSES_IGNORE);
106
```

```
107     if (rank % 2 == 1) {
108         if (rank < size - 1)
109             MPI_Ssend(&(B[row_end][1]), 1, my_type2, rank + 1, 1, MPI_COMM_WORLD);
110     }
111     else {
112         if (rank > 0)
113             MPI_Recv(&(A[row_start - 1][1]), 1, my_type2, rank - 1, 1,
114                 MPI_COMM_WORLD, MPI_STATUSES_IGNORE);
115     }
116
117     if (rank % 2 == 0) {
118         if (rank < size - 1)
119             MPI_Ssend(&(B[row_end][1]), 1, my_type2, rank + 1, 2, MPI_COMM_WORLD);
120     }
121     else {
122         if (rank > 0)
123             MPI_Recv(&(A[row_start - 1][1]), 1, my_type2, rank - 1, 2,
124                 MPI_COMM_WORLD, MPI_STATUSES_IGNORE);
125     }
126
127     if (rank % 2 == 0) {
128         if (rank > 0)
129             MPI_Ssend(&(B[row_start][1]), 1, my_type2, rank - 1, 3, MPI_COMM_WORLD);
130     }
131     else {
132         if (rank < size - 1)
133             MPI_Recv(&(A[row_end + 1][1]), 1, my_type2, rank + 1, 3, MPI_COMM_WORLD,
134                 MPI_STATUSES_IGNORE);
135     }
136 }
137
138 if (rank > 0)
139     MPI_Ssend(&(A[row_start][0]), region, my_type1, 0, rank, MPI_COMM_WORLD);
140 else
141     for (k = 1; k < size; k++)
142         MPI_Recv(&(A[region * k + 1][0]), region, my_type1, k, k, MPI_COMM_WORLD,
143             MPI_STATUSES_IGNORE);
144
145 if (rank == 0) {
146     cout << "\nFinal values of the matrix A:\n";
147     for (i = 0; i < N + 2; i++) {
148         for (j = 0; j < N + 2; j++)
149             printf("%6.2f ", A[i][j]);
150         cout << endl;
151     }
152 }
153
154 end:
155 my_time += MPI_Wtime();
156 if (rank == 0)
157     printf("\nTime = %10.6f s\n", my_time);
158
159 fflush(stdout);
```

```
160     MPI_Finalize();  
161  
162     return 0;  
163 }
```