



МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова



Факультет вычислительной математики и кибернетики

Практикум по учебному курсу

Распределенные системы

ОТЧЕТ

о выполненном задании (MPI_ALLREDUCE)

студента 428 учебной группы факультета ВМК МГУ

Романова Александра Юрьевича

гор. Москва

2022 год

Содержание

1	Постановка задачи	2
2	Алгоритм	3
3	Реализация алгоритма	6
4	Инструкции по запуску	10
5	Временная оценка	11

1 Постановка задачи

В транспьютерной матрице размером 4×4 , в каждом узле которой находится один процесс, необходимо выполнить операцию нахождения максимума среди 16 чисел (каждый процесс имеет свое число). Найденное максимальное значение должно быть получено на каждом процессе.

Реализовать программу, моделирующую выполнение операции *MPI_ALLREDUCE* на транспьютерной матрице при помощи пересылок *MPI* типа точка-точка.

Оценить сколько времени потребуется для выполнения операции *MPI_ALLREDUCE*, если все процессы выдали эту операцию редукции одновременно. Время старта равно 100, время передачи байта равно 1 ($T_s=100, T_b=1$). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

2 Алгоритм

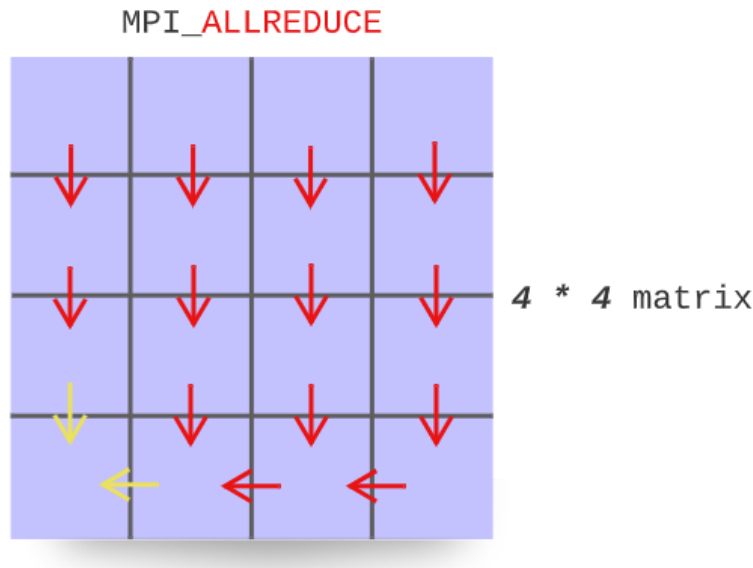


Рис. 1: Сбор максимума в одной ячейке

Рассмотрим матрицу 4×4 , нижняя левая клетка имеет индекс $(0, 0)$, верхняя правая индексируется $(3, 3)$. Наибольшее число будем находить на процессе $(0, 0)$ и затем отправлять его всем остальным процессам. Нахождение максимума на процессе $(0, 0)$ представлено на Рис. 1. После того, как результат максимума будет лежать в ячейке $(0, 0)$, мы отправляем его обратно (фактически производим инверсию стрелок) по всем процессам, как показано на Рис. 2.

Разобьем полученный алгоритм на шаги, которые между собой аналогичны (относительно стрелок на рисунках), далее эти шаги преобразуем в функции:

- "Вперед вниз"

Процессы, находящиеся в координатах $(x, 3)$ (где $x = \{0, 1, 2, 3\}$), отправляют свое число процессам, находящимся в координатах $(x, 2)$. Эти процессы получают число, сравнивают его со своим и отправляют наибольшее из них процессам, находящимся в координатах

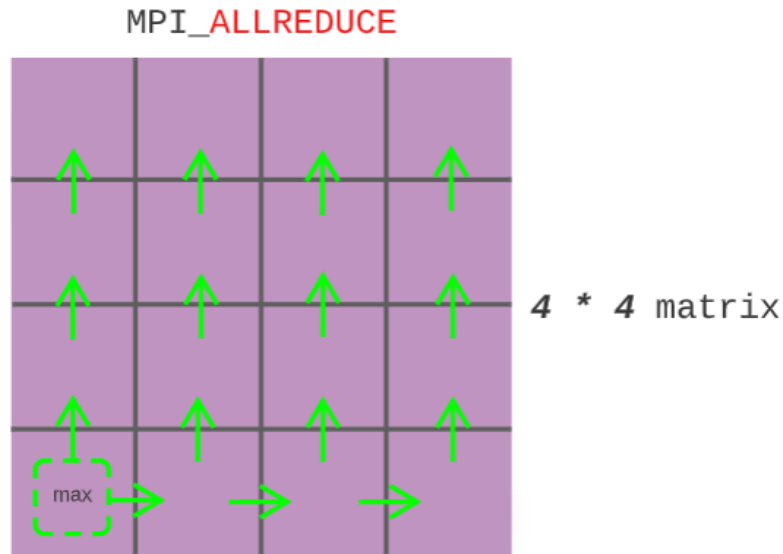


Рис. 2: Рассылка максимума по ячейкам

тах $(x, 1)$. Эти процессы аналогично отправляют наибольшее число процессам, находящимся в координатах $(x, 0)$.

- "Вперед влево"
Процесс, находящийся в координатах $(3,0)$, получает число от процесса с координатами $(3,1)$, сравнивает со своим и отправляет наибольшее из них в процесс с координатами $(2,0)$. Процесс, находящийся в координатах $(2,0)$, получает числа от процессов с координатами $(3,0)$ и $(2,1)$, сравнивает их со своим и отправляет наибольшее процессу $(1,0)$. Процесс с координатами $(1,0)$ аналогично отправляет наибольшее число процессу $(0,0)$.
- "Определение максимума среди 16 чисел"
Процесс $(0,0)$ получает числа от процессов с координатами $(0,1)$ и $(1,0)$, сравнивает их со своим и находим наибольшее. Это число и будет максимумом среди 16 чисел. Теперь нужно разослать это число оставшимся процессам.
- "Обратно вправо"
Из $(0,0)$ полученное число отправляется к $(1,0)$, оттуда в $(2,0)$ и в

$(3,0)$.

- "Обратно вверх"

Из $(0,0)$, $(1,0)$, $(2,0)$, $(3,0)$ полученное число отправляется вверх, то есть в $(0,1)$, $(1,1)$, $(2,1)$, $(3,1)$ соответственно. От этих процессов число попадает в $(0,2)$, $(1,2)$, $(2,2)$, $(3,2)$ соответственно. Далее в $(0,3)$, $(1,3)$, $(2,3)$, $(3,3)$ соответственно.

3 Реализация алгоритма

```
1  #include <mpi.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <stdlib.h>
5
6  #ifndef max
7  #define max(a, b) (((a) > (b)) ? (a) : (b))
8  #endif
9
10 int RecvNumberFrom(int coordinates[2], MPI_Comm cart) {
11     MPI_Status status;
12     int rank, number;
13     MPI_Cart_rank(cart, coordinates, &rank);
14     MPI_Recv(&number, 1, MPI_INT, rank, 0, cart, &status);
15     return number;
16 }
17
18 void ISendNumberTo(int coordinates[2], int number, MPI_Comm cart) {
19     MPI_Request request;
20     int rank;
21     MPI_Cart_rank(cart, coordinates, &rank);
22     MPI_Isend(&number, 1, MPI_INT, rank, 0, cart, &request);
23 }
24
25 int Master(int coordinates[2], int number, MPI_Comm cart) {
26     int up_coordinates[2] = {coordinates[0], coordinates[1] + 1};
27     int right_coordinates[2] = {coordinates[0] + 1, coordinates[1]};
28
29     int up_number = RecvNumberFrom(up_coordinates, cart);
30     int right_number = RecvNumberFrom(right_coordinates, cart);
31
32     int max_number = max(up_number, right_number);
33     max_number = max(max_number, number);
34     printf("Max number %d\n", max_number); fflush(stdout);
35
36     ISendNumberTo(up_coordinates, max_number, cart);
37     ISendNumberTo(right_coordinates, max_number, cart);
38     return max_number;
```

```

39 }
40
41 int DownRightProcess(int coordinates[2], int number, MPI_Comm cart) {
42     int up_coordinates[2] = {coordinates[0], coordinates[1] + 1};
43     int left_coordinates[2] = {coordinates[0] - 1, coordinates[1]};
44
45     int up_number = RecvNumberFrom(up_coordinates, cart);
46
47     int max_number = max(number, up_number);
48
49     ISendNumberTo(left_coordinates, max_number, cart);
50
51     max_number = RecvNumberFrom(left_coordinates, cart);
52     ISendNumberTo(up_coordinates, max_number, cart);
53     return max_number;
54 }
55
56 int DownProcesses(int coordinates[2], int number, MPI_Comm cart) {
57     int up_coordinates[2] = {coordinates[0], coordinates[1] + 1};
58     int right_coordinates[2] = {coordinates[0] + 1, coordinates[1]};
59     int left_coordinates[2] = {coordinates[0] - 1, coordinates[1]};
60
61     int up_number = RecvNumberFrom(up_coordinates, cart);
62     int right_number = RecvNumberFrom(right_coordinates, cart);
63
64     int max_number = max(up_number, right_number);
65     max_number = max(max_number, number);
66
67     ISendNumberTo(left_coordinates, max_number, cart);
68
69
70     max_number = RecvNumberFrom(left_coordinates, cart);
71     ISendNumberTo(up_coordinates, max_number, cart);
72     ISendNumberTo(right_coordinates, max_number, cart);
73     return max_number;
74 }
75
76 int UpProcesses(int coordinates[2], int number, MPI_Comm cart) {
77     int down_coordinates[2] = {coordinates[0], coordinates[1] - 1};
78     ISendNumberTo(down_coordinates, number, cart);

```



```

79
80     int max_number = RecvNumberFrom(down_coordinates, cart);
81     return max_number;
82 }
83
84 int InternalProcesses(int coordinates[2], int number, MPI_Comm cart) {
85     int up_coordinates[2] = {coordinates[0], coordinates[1] + 1};
86     int down_coordinates[2] = {coordinates[0], coordinates[1] - 1};
87
88     int up_number = RecvNumberFrom(up_coordinates, cart);
89
90     int max_number = max(number, up_number);
91
92     ISendNumberTo(down_coordinates, max_number, cart);
93
94     max_number = RecvNumberFrom(down_coordinates, cart);
95     ISendNumberTo(up_coordinates, max_number, cart);
96     return max_number;
97 }
98
99 int main(int argc, char *argv[]) {
100     int size;
101     const int dims[2] = {4, 4};
102     const int periods[2] = {0, 0};
103     MPI_Comm cart;
104
105     MPI_Init(&argc, &argv);
106     MPI_Comm_size(MPI_COMM_WORLD, &size);
107     MPI_Cart_create(MPI_COMM_WORLD, 2 /*ndims*/, dims,
108                     periods, 0 /*reorder*/, &cart);
109
110     int rank;
111     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
112
113     int coordinates[2];
114     MPI_Cart_coords(cart, rank, 2, coordinates);
115
116     int number = rank;
117     printf("(%d, %d): %d\n", coordinates[0], coordinates[1], number); fflush(stdout);
118

```

```

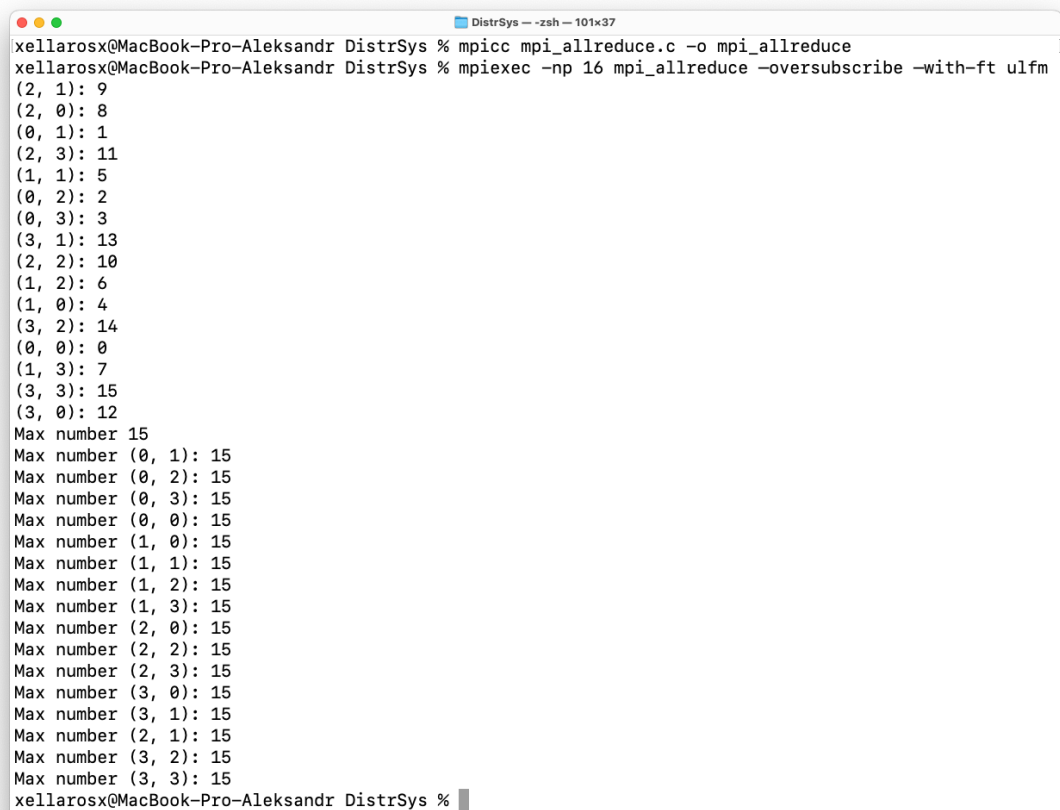
119     int max_number;
120     if (coordinates[0] == 0 && coordinates[1] == 0) {
121         max_number = Master(coordinates, number, cart);
122     } else if (coordinates[0] == 3 && coordinates[1] == 0) {
123         max_number = DownRightProcess(coordinates, number, cart);
124     } else if (coordinates[1] == 0) {
125         max_number = DownProcesses(coordinates, number, cart);
126     } else if (coordinates[1] == 3) {
127         max_number = UpProcesses(coordinates, number, cart);
128     } else {
129         max_number = InternalProcesses(coordinates, number, cart);
130     }
131
132     printf("Max number (%d, %d): %d\n", coordinates[0], coordinates[1], max_number);
133     fflush(stdout);
134
135     MPI_Finalize();
136     return 0;
137 }

```

4 Инструкции по запуску

Для запуска необходимо:

- `mpicc mpi_allreduce.c -o mpi_allreduce`
- `mpiexec -np 16 mpi_allreduce --oversubscribe --with-ft ulfm`



```
xellarosx@MacBook-Pro-Aleksandr DistrSys % mpicc mpi_allreduce.c -o mpi_allreduce
xellarosx@MacBook-Pro-Aleksandr DistrSys % mpiexec -np 16 mpi_allreduce --oversubscribe --with-ft ulfm
(2, 1): 9
(2, 0): 8
(0, 1): 1
(2, 3): 11
(1, 1): 5
(0, 2): 2
(0, 3): 3
(3, 1): 13
(2, 2): 10
(1, 2): 6
(1, 0): 4
(3, 2): 14
(0, 0): 0
(1, 3): 7
(3, 3): 15
(3, 0): 12
Max number 15
Max number (0, 1): 15
Max number (0, 2): 15
Max number (0, 3): 15
Max number (0, 0): 15
Max number (1, 0): 15
Max number (1, 1): 15
Max number (1, 2): 15
Max number (1, 3): 15
Max number (2, 0): 15
Max number (2, 2): 15
Max number (2, 3): 15
Max number (3, 0): 15
Max number (3, 1): 15
Max number (2, 1): 15
Max number (3, 2): 15
Max number (3, 3): 15
xellarosx@MacBook-Pro-Aleksandr DistrSys %
```

Рис. 3: Пример работы программы

5 Временная оценка

Дольше всех будет идти сообщение от (3, 3) до (0, 0). Ему нужно преодолеть 6 пересылок в одну сторону и столько же обратно. На каждую пересылку нужно потратить $(Ts + 4 * Tb)$ времени ($4 = \text{sizeof}(int)$).

Значит в итоге получим:

$$T = 2 * 6 * (Ts + 4 * Tb) = 1248 \quad (1)$$