

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №5
по дисциплине «ПА»
Тема: Топологии

Студент гр. 8304

Сергеев А.Д.

Преподаватель

Татаринов Ю.С.

Санкт-Петербург

2020

Задание.

Число процессов K кратно трем: $K = 3N$, $N > 1$. В процессах 0, 1 и 2 дано по N целых чисел. Определить для всех процессов декартову топологию в виде матрицы размера $N \times 3$, после чего, используя функцию `MPI_Cart_sub`, расщепить матрицу процессов на три одномерных столбца (при этом процессы 0, 1 и 2 будут главными процессами в полученных столбцах). Используя одну коллективную операцию пересылки данных, переслать по одному исходному числу из главного процесса каждого столбца во все процессы этого же столбца и вывести полученное число в каждом процессе (включая процессы 0, 1 и 2).

Описание алгоритма.

Функция `MPI_Cart_Create` позволяет создать декартову топографию для процессов, а `MPI_Cart_sub` — разделить процессы в разные коммуникаторы, но не по цвету, а в зависимости от топографии. При этом у каждого из полученных коммуникаторов уже будет установлена топография. Для создания используется массив размерностей по каждой из координат топографии и массив чисел, обозначающих образуют ли процессы по этой координате цикл. Для разделения используется массив чисел, обозначающих останется ли эта координата в топографии полученного коммуникатора. Полученные результаты собираются при помощи функции `MPI_Gather`.

Листинг программы.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <mpi.h>

#define true 1
#define false 0

#define array_upper_limit 1000000.0

int main(int argc, char* argv[]) {
    int ProcNum, ProcRank;

    MPI_Init(&argc, &argv);
```

```

MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
srand(time(NULL) + ProcRank);
double start_time = MPI_Wtime();

int n;
if (ProcNum % 3) {
    if (ProcRank == 0) printf("Wrong process number!\n");
    return 0;
} else n = ProcNum / 3;

int a = (int) ((array_upper_limit / RAND_MAX) * rand());
printf("Process %d: number is %d\n", ProcRank, a);

MPI_Comm low_comm, high_comm;
MPI_Cart_create(MPI_COMM_WORLD, 2, (int[2]) {n, 3}, (int[2]) {false, false},
false, &low_comm);
MPI_Cart_sub(low_comm, (int[]) {true, false}, &high_comm);

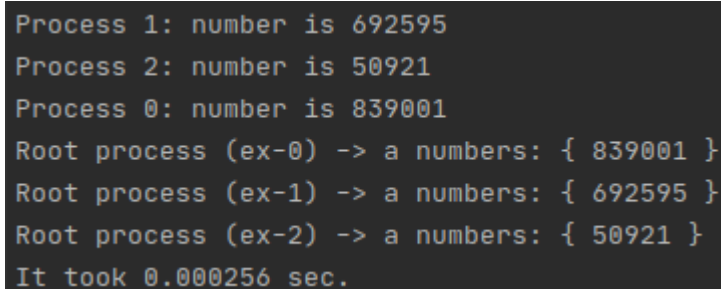
int NProcNum, NProcRank;
MPI_Comm_size(high_comm, &NProcNum);
MPI_Comm_rank(high_comm, &NProcRank);

int *buffer = malloc(NProcNum * sizeof(int));
MPI_Gather(&a, 1, MPI_INT, buffer, 1, MPI_INT, 0, high_comm);
if (NProcRank == 0) {
    printf("Root process (ex-%d) -> a numbers: {" , ProcRank);
    for (int i = 0; i < NProcNum; ++i) {
        if (i != NProcNum - 1) printf(" %d," , buffer[i]);
        else printf(" %d", buffer[i]);
    }
    printf(" }\n");
}
if (ProcRank == 0) printf("It took %lf sec.\n", MPI_Wtime() - start_time);

free(buffer);
MPI_Finalize();
return 0;
}

```

Результаты работы.



```

Process 1: number is 692595
Process 2: number is 50921
Process 0: number is 839001
Root process (ex-0) -> a numbers: { 839001 }
Root process (ex-1) -> a numbers: { 692595 }
Root process (ex-2) -> a numbers: { 50921 }
It took 0.000256 sec.

```

Рисунок 1 — Результат работы программы для 3 процессов

```

Process 0: number is 244100
Process 4: number is 180645
Process 5: number is 532543
Process 6: number is 899196
Process 1: number is 606351
Process 2: number is 966771
Process 8: number is 613184
Process 7: number is 250781
Process 3: number is 317505
Root process (ex-0) -> a numbers: { 244100, 317505, 899196 }
It took 0.124343 sec.
Root process (ex-1) -> a numbers: {Root process (ex-2) -> a numbers: { 966771, 532543, 613184 }
606351, 180645, 250781 }

```

Рисунок 2 — Результат работы программы для 9 процессов

```

Process 1: number is 146936
Process 9: number is 13090
Process 17: number is 382476
Process 10: number is 371076
Process 12: number is 90093
Process 11: number is 230281
Process 0: number is 790956
Process 3: number is 365132
Process 5: number is 581692
Process 13: number is 950542
Process 4: number is 723329
Process 2: number is 503017
Process 16: number is 26480
Process 6: number is 436489
Process 7: number is 800266
Process 8: number is 659083
Process 14: number is 806081
Process 15: number is 162101
Root process (ex-0) -> a numbers: { 790956, 365132, 436489, 13090, 90093, 162101 }
It took 0.254050 sec.
Root process (ex-2) -> a numbers: { 503017, 581692, 659083, 230281, 806081, 382476 }
Root process (ex-1) -> a numbers: { 146936, 723329, 800266, 371076, 950542, 26480 }

```

Рисунок 3 — Результат работы программы для 18 процессов

```

Process 49: number is 32518
Process 50: number is 393938
Process 38: number is 593778
Process 11: number is 922180
Process 20: number is 641386
Root process (ex-1) -> a numbers: { 972645, 53124, 485124, 562485, 629954, 710271, 784816, 359048, 936776, 6787, 589463, 159545, 733604, 311240, 881446, 462207, 32518, 113340 }
Root process (ex-2) -> a numbers: { 831765, 269113, 842344, 922180, 993904, 564461, 641386, 223778, 293742, 367742, 943134, 22709, 593778, 166422, 744447, 321192, 393938, 469035 }
Root process (ex-0) -> a numbers: { 972645, 53124, 127899, 700097, 922180, 847044, 926483, 498531, 76874, 148717, 722905, 882927, 877574, 953607, 526082, 595484, 682596, 251411 }
It took 0.887369 sec.

```

Рисунок 4 — Результат работы программы для 54 процессов (некоторые строки опущены)

```

Process 104: number is 868175
Process 3: number is 166422
Process 65: number is 386639
Process 72: number is 398314
Process 76: number is 829460
Root process (ex-2) -> a numbers: { 311240, 881446, 462207, 32518, 113340, 188011, 757645, 332388, 411771, 485922, 64610, 636299, 211011, 286431, 858811, 434401, 15045, 86814, 662253, 739417, 812932, 386639, 459085, 535260, 614484, 688750, 764470, 836980, 412729, 985789, 567014, 644743, 713022, 789056, 868175, 941077 }
Root process (ex-1) -> a numbers: { 953607, 526082, 595484, 682596, 251411, 830329, 404169, 977926, 52146, 627427, 206332, 778803, 352226, 928354, 504151, 78543, 653385, 725758, 806115, 382134, 453949, 26979, 602088, 677286, 251914, 829460, 407731, 481347, 556036, 129875, 205487, 279564, 353535, 433336, 505662, 580861 }
Root process (ex-0) -> a numbers: { 593778, 166422, 744447, 321192, 393938, 469035, 546415, 617053, 692738, 768426, 346331, 922590, 991117, 570460, 643599, 222848, 295287, 373541, 448550, 21177, 94121, 174806, 744951, 822518, 398314, 972177, 49123, 119859, 696727, 775063, 846131, 924965, 997699, 572469, 647477, 224770 }
It took 2.932336 sec.

```

Рисунок 5 — Результат работы программы для 108 процесса (некоторые строки опущены)

График зависимости времени выполнения программы от числа процессов.

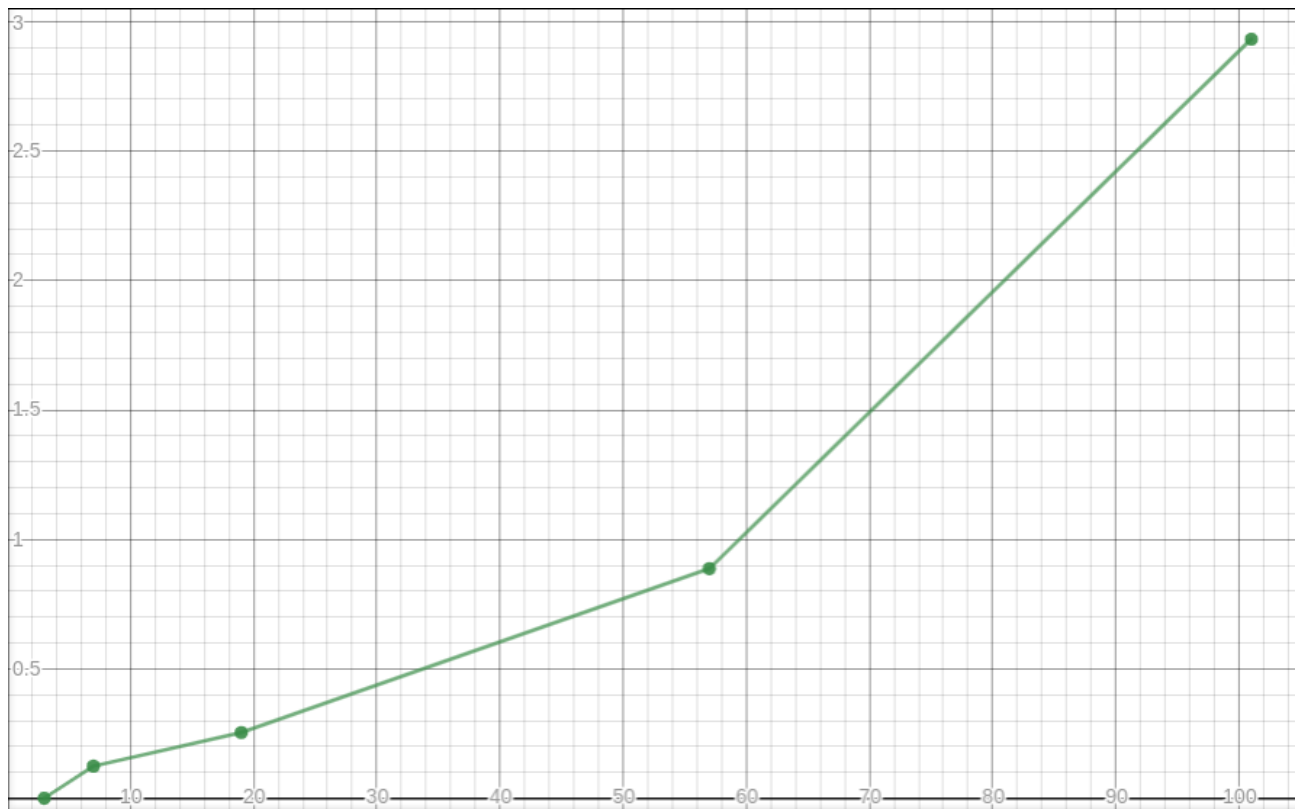


Рисунок 6 — График зависимости времени выполнения программы от числа процессов

Выводы.

Создана программа, собирающая числа из каждого столбца одномерной матрицы. В ходе выполнения работы изучены принципы создания декартовой топологии процессов, деления коммуникатора в зависимости от координат топологии, а также циклического смещения процессов.