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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Системы параллельной обработки данных»

Тема: Группы процессов и коммуникаторы

Студент гр. 0303 _____ Калмак Д.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург
2024

Цель работы.

Целью работы является написание параллельной программы MPI с группой процессов и коммуникаторами, изучение зависимостей времени работы программы от числа процессов и построение сети Петри.

Задание.

1. Написать параллельную программу, где в каждом процессе дано целое число N, которое может принимать два значения: 0 и 1 (имеется хотя бы один процесс с N = 1). Кроме того, в каждом процессе с N = 1 дано вещественное число A. Используя функцию MPI_Comm_split и одну коллективную операцию редукции, найти сумму всех исходных чисел A и вывести ее во всех процессах с N = 1.

   Указание. При вызове функции MPI_Comm_split в процессах, которые не требуется включать в новый коммуникатор, в качестве параметра color следует указывать константу MPI_UNDEFINED.
2. Краткое описание выбранного алгоритма решения задачи и листинг программы.
3. Нарисовать сеть Петри для MPI программы.
4. Распечатка результатов и времени работы программы на разном количестве процессов.
5. Построить графики зависимости времени выполнения программы от числа запущенных процессов.
6. Построить графики ускорения/замедления программы.

Выполнение работы.

1-2. С использованием языка программирования C++ написана параллельная программа MPI с использованием групп процессов и коммуникаторов. Для работы с MPI была включена библиотека mpi.h. Для

инициализации среды MPI используется MPI_Init. MPI_Comm_size позволяет получить общее количество процессов, а MPI_Comm_rank – ранг текущего процесса. Каждый процесс имеет переменную N, которая может принимать 0 или 1, что регулируется четностью процесса. Объявляется переменная group_Comm для хранения нового коммуникатора, который будет создан для группы процессов. Для процессов, которые в переменной N имеют значение 1, создается переменная A, которая имеет случайное вещественное значение от 0.0 до 100.0. Для этого используется шаблонный класс uniform_real_distribution<double> и генератор псевдослучайных чисел на основе алгоритма Mersenne Twister: чтобы процесс генерации был более случайным, для генератора случайных чисел используются: std::random_device rd, чтобы получить случайное число, которое обеспечивает более высокую степень случайности и используется для инициализации std::mt19937 gen(rd()). Затем происходит расщепление на подгруппу с новым коммуникатором group_Comm. Это позволяет сделать функция MPI_Comm_split(MPI_COMM_WORLD, N, ProcRank, &group_Comm), которая расщепляет процессы группы из коммуникатора MPI_COMM_WORLD по color равному 1, то есть все нечетные процессы, и под управлением нового коммуникатора group_Comm. Процессы, которые в переменной N имеют значение 0, так же взаимодействуют с функцией MPI_Comm_split(MPI_COMM_WORLD, MPI_UNDEFINED, ProcRank, &group_Comm), поскольку все процессы старого коммуникатора должны вызвать эту функцию. Процессы с N = 0 не должны иметь отношение к новому коммуникатору и суммирующей подгруппе, поэтому в качестве color указано MPI_UNDEFINED, что позволяет указать процессам в качестве нового коммуникатора значение MPI_COMM_NULL. Поскольку вывести сумму A должны все процессы коммуникатора group_Comm, используется функция MPI_Allreduce(&A, &sum_A, 1, MPI_DOUBLE, MPI_SUM, group_Comm), которая записывает в переменную sum_A каждого процесса коммуникатора

group_Comm сумму значений A, по одному у каждого процесса, типа MPI_DOUBLE с помощью оператора MPI_SUM, указывающего на операцию суммирования. После этого каждый процесс из суммирующей подгруппы помечает коммуникатор group_Comm для удаления. Программа представлена в листинге 1. Пример работы программы представлено в листинге 2.

Листинг 1.

```
#include <iostream>
#include <random>
#include <mpi.h>

using namespace std;

random_device rd;
mt19937 gen(rd());

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);

    int N = (ProcRank % 2 == 0) ? 0 : 1;

    MPI_Comm group_Comm;

    double start, end;
    if (N == 1) {
        double A;
        uniform_real_distribution<double> a_real(0.0, 100.0);
        A = a_real(gen);
        // cout << "Process " << ProcRank << " with A = " << A << endl;

        if (ProcRank == 1) start = MPI_Wtime();
        MPI_Comm_split(MPI_COMM_WORLD, N, ProcRank, &group_Comm);

        double sum_A = 0.0;
        MPI_Allreduce(&A, &sum_A, 1, MPI_DOUBLE, MPI_SUM, group_Comm);

        MPI_Comm_free(&group_Comm);
        if (ProcRank == 1) {
            end = MPI_Wtime();
            cout << "Time " << end - start << endl;
        }

        cout << "Process " << ProcRank << " has the sum of A, which is " <<
sum_A << endl;
    } else {
        MPI_Comm_split(MPI_COMM_WORLD, MPI_UNDEFINED, ProcRank, &group_Comm);
    }
```

```
    MPI_Finalize();
    return 0;
}
```

Листинг 2.

```
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 8 --oversubscribe ./main
Process 1 with A = 63.4233
Process 3 with A = 15.7842
Process 5 with A = 51.0424
Process 7 with A = 41.5479
Time 0.00107466
Process 1 has the sum of A, which is 171.798
Process 3 has the sum of A, which is 171.798
Process 5 has the sum of A, which is 171.798
Process 7 has the sum of A, which is 171.798
```

3. Построена сеть Петри. Сеть Петри представлена на рис. 1 и отражает параллельную программу для четырех процессов, в которой есть группы процессов и коммуникаторы.

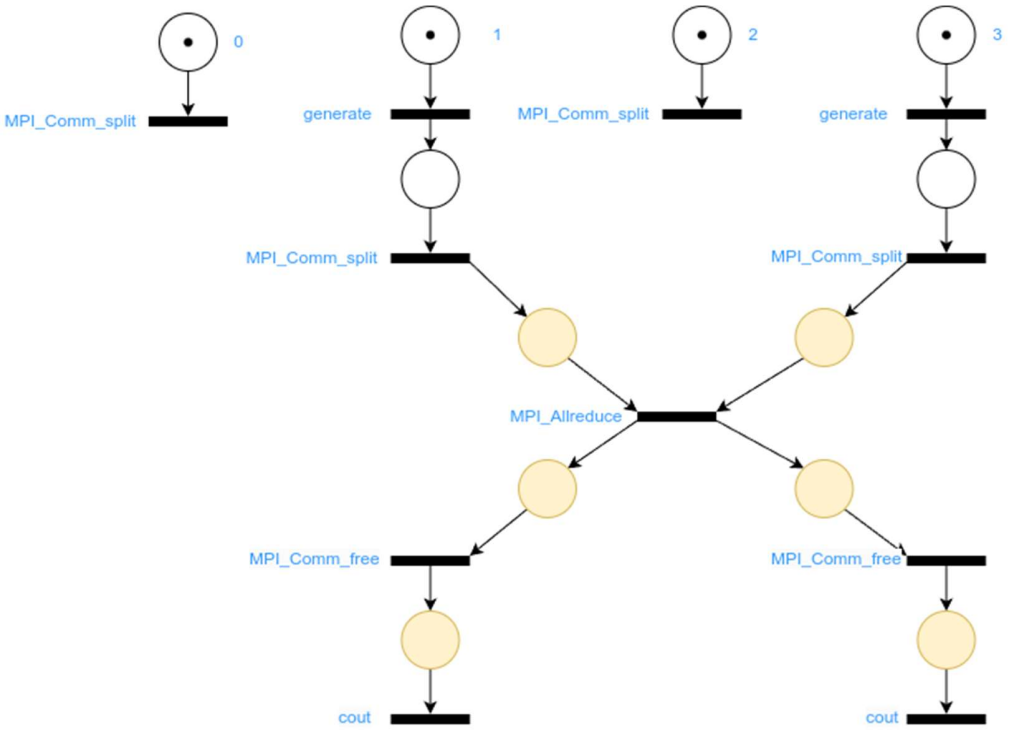

Рисунок 1 – Сеть Петри для параллельной программы с группами процессов и коммуникаторами для четырех процессов

4. Программа была запущена на разных числах процессов: 2, 4, 6, 8, 10, 12, 14, 16 и разным условием для назначения N = 1: каждый нечетный процесс и

только нулевой процесс, – то есть с учетом роста суммирующей подгруппы и без учета. Время выполнения программы в зависимости от количества процессов с каждым нечетным процессом с N = 1 представлено в листинге 3, а с только нулевым – в листинге 4.

Листинг 3.

```
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 2 --oversubscribe ./main
Time 8.2483e-05
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 4 --oversubscribe ./main
Time 9.9176e-05
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 6 --oversubscribe ./main
Time 0.000191157
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 8 --oversubscribe ./main
Time 0.000244726
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 10 --oversubscribe ./main
Time 0.000442378
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 12 --oversubscribe ./main
Time 0.00048142
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 14 --oversubscribe ./main
Time 0.00051697
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 16 --oversubscribe ./main
Time 0.00102507
```

Листинг 4.

```
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 2 --oversubscribe ./main
Time 6.579e-05
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 4 --oversubscribe ./main
Time 0.000101688
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 6 --oversubscribe ./main
Time 0.000132698
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 8 --oversubscribe ./main
Time 0.00016245
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 10 --oversubscribe ./main
Time 0.000251428
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 12 --oversubscribe ./main
Time 0.000345644
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 14 --oversubscribe ./main
Time 0.000551744
cut@cute:~/Документы/cpluscplus/5$ mpiexec.openmpi -n 16 --oversubscribe ./main
Time 0.000631712
```

5. На основе полученных данных при выполнении программы на разных количествах процессов построены графики зависимости времени выполнения программы от числа запущенных процессов с учетом роста суммирующей подгруппы и без учета, которые представлены на рис. 2-3 соответственно.

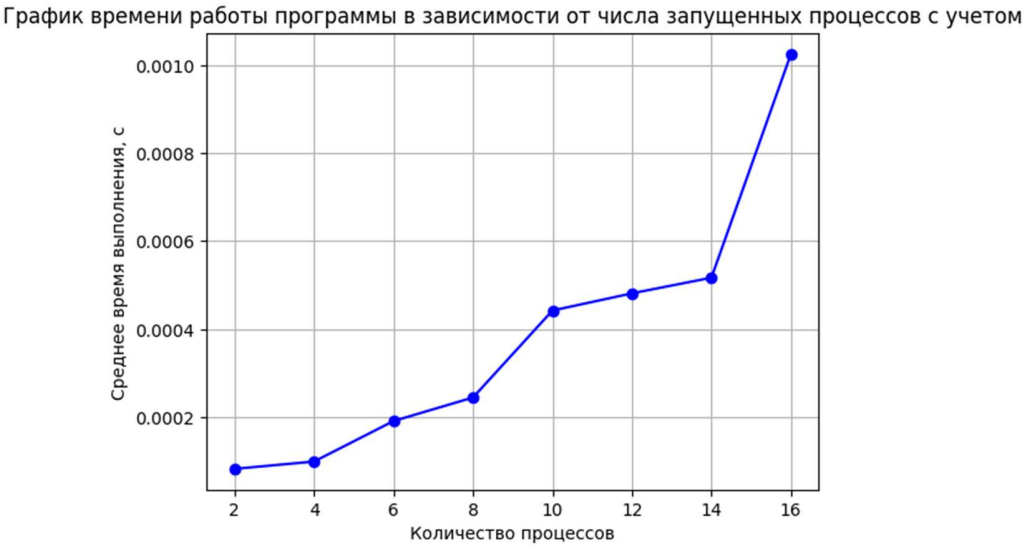

Рисунок 2 – График времени работы программы в зависимости от числа запущенных процессов с учетом суммирующей подгруппы

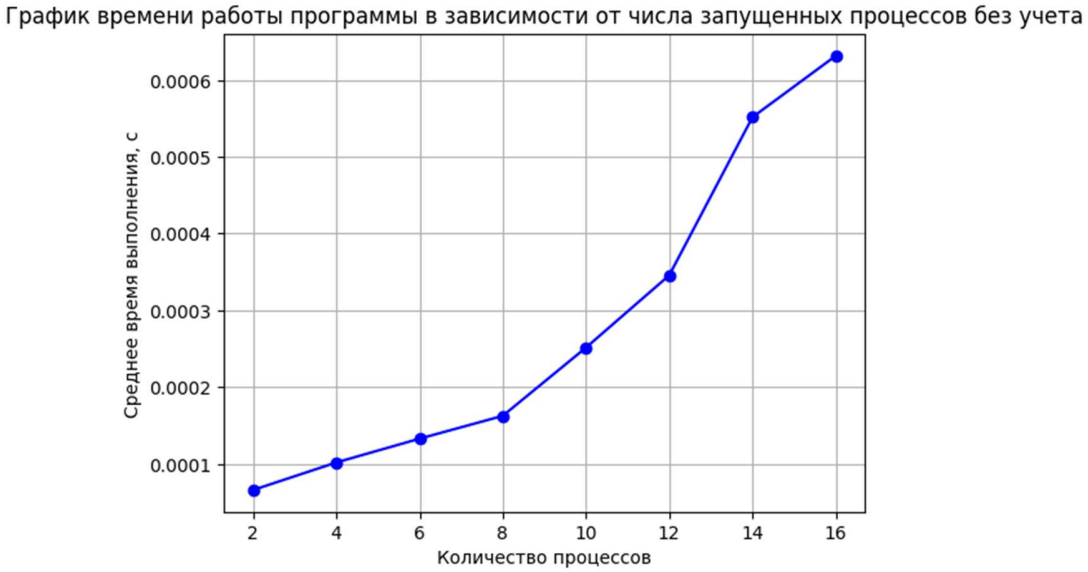

Рисунок 3 – График времени работы программы в зависимости от числа запущенных процессов без учета суммирующей подгруппы

Исходя из графика, представленного на рис. 2, время работы программы увеличивается с ростом количества запущенных процессов. Каждый нечетный процесс имеет значение N, равное 1, что означает его добавление в новую подгруппу нового коммуникатора, которая предназначена для суммирования.

Чем больше процессов, тем больше суммирующих процессов, которые нагружают программу путем создания большей суммирующей подгруппы, большей нагрузки на суммирование MPI и пометками на удаление нового коммуникатора. Исходя из графика, представленного на рис. 3, время работы программы увеличивается с ростом количества запущенных процессов. Только нулевой процесс является частью суммирующей подгруппы, однако остальные процессы все равно взаимодействуют со звеном деления на подгруппы с новым коммуникатором: с помощью функции MPI_Comm_split, только не входят в новый коммуникатор из-за color равного MPI_UNDEFINED, а возвращают MPI_COMM_NULL, – поэтому с увеличением количества процессов нагрузка на разделение возрастает. Из двух графиков, представленных на рис. 2-3, можно заметить, что время выполнения программы при одинаковом количестве процессов меньше при постоянной суммирующей подгруппе – рис. 3, что говорит об уменьшении нагрузки создания новой подгруппы с новым коммуникатором, суммирования и удаления нового коммуникатора. Время выполнения программы может быть разным при одинаковых условиях, что может быть вызвано созданием нового коммуникатора и расщеплением на подгруппы, выполнением суммирования, пометкой удаления нового коммуникатора, а также разным состоянием вычислительной машины.

6. Для построения графика ускорения/замедления работы программы с разным количеством запущенных процессов необходимо разделить время выполнения программы с 2 процессами на время выполнения программы с 4, 6, 8, 10, 12, 14, 16 процессами соотвественно. Графики замедления представлены на рис. 5-6 с учетом суммирующей подгруппы и без соответственно.

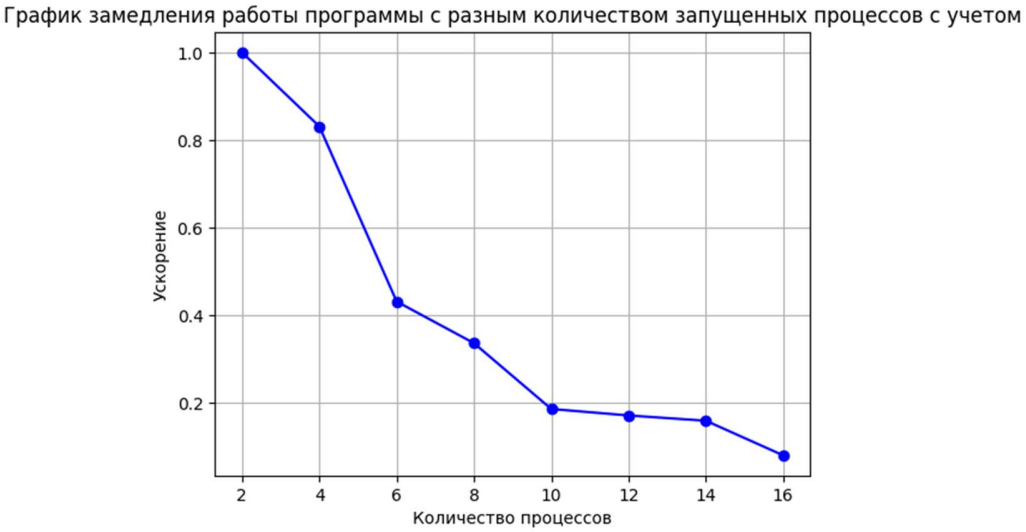

Рисунок 5 – График замедления работы программы с разным количеством запущенных процессов с учетом суммирующих процессов

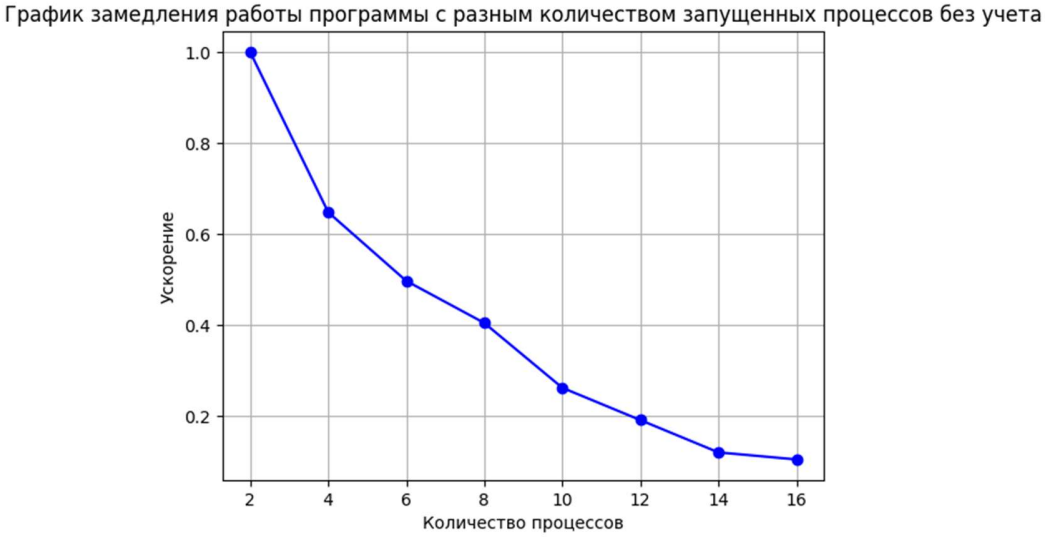

Рисунок 6 – График замедления работы программы с разным количеством запущенных процессов без учета суммирующих процессов

Вывод.

Таким образом, было освоено написание параллельных программ MPI с использованием групп процессов и коммуникаторов библиотеки MPI.

Были изучены зависимости времени работы программы от числа запущенных процессов с учетом роста суммирующей подгруппы и без учета. С

увеличением числа процессов время работы программы увеличивается в обоих случаях. В первом случае на результат влияет увеличение процессов в суммирующей подгруппе с новым коммуникатором, что приводит к увеличению нагрузки как на ее создание, так и на суммирование с освобожденим коммуникатора. Во втором случае на результат влияет обязательное условие запуска функции расщепления процессов из старого коммуникатора на подгруппы с новым коммуникатором, даже если процессы не войдут в новый коммуникатор. На основании полученных данных о времени работы программы построены графики времени работы программы в зависимости от числа запущенных процессов и соответствующие им графики ускорения/замедления работы программы. Для программы была построена сеть Петри.