

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего порядка
«Чувашский государственный университет имени И. Н. Ульянова»
Факультет информатики и вычислительной техники
Кафедра вычислительной техники
Дисциплина Параллельное программирование

Лабораторная работа №4.
Параллельное программирование

Выполнил: студ.ИВТ-42-18

Жижайкин К.В.

Проверил: доцент

Ковалев С. В.

Чебоксары, 2021 г.

22)

Задание:

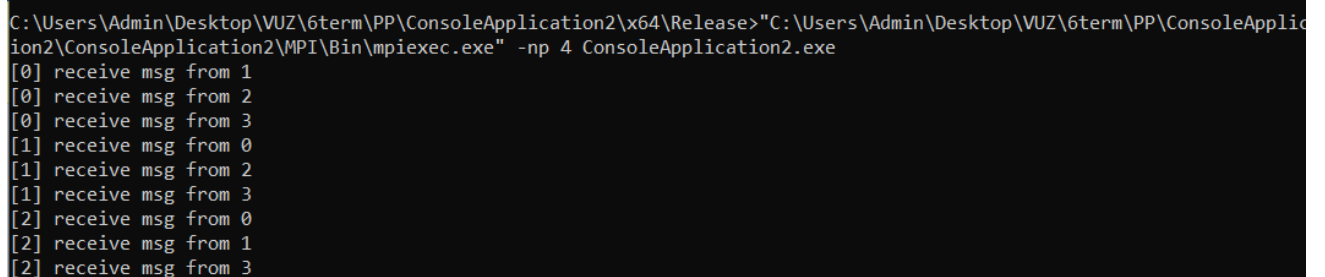
Коммуникации «точка-точка»: схема «каждый каждому» Напишите MPI-программу, реализующую при помощи блокирующих функций послыки сообщений типа точка-точка схему коммуникации процессов «каждый каждому», в которой осуществляется пересылка сообщения от каждого процесса каждому. В качестве передаваемого сообщения используйте номер процесса. Каждый процесс должен вывести на экран все полученные сообщения.

Код:

```
for (auto dest = 0; dest < size; dest++)
{
    buffer = rank;
    if (rank != dest)
    {
        MPI_Isend(&buffer, 1, MPI_INT, dest, 0, MPI_COMM_WORLD, &request);
    }
}

for (auto source = 0; source < size; source++)
{
    if (rank != source)
    {
        MPI_Irecv(&buffer, 1, MPI_INT, source, 0, MPI_COMM_WORLD, &request);
        MPI_Wait(&request, &status);
        std::cout << "[" << rank << "]" receive msg from " << buffer << "\n";
    }
}
```

Результат:



```
C:\Users\Admin\Desktop\VUZ\6term\PP\ConsoleApplication2\x64\Release>C:\Users\Admin\Desktop\VUZ\6term\PP\ConsoleApplication2\ConsoleApplication2\MPI\Bin\mpiexec.exe -np 4 ConsoleApplication2.exe
[0] receive msg from 1
[0] receive msg from 2
[0] receive msg from 3
[1] receive msg from 0
[1] receive msg from 2
[1] receive msg from 3
[2] receive msg from 0
[2] receive msg from 1
[2] receive msg from 3
```

23)

Задание:

Коллективные коммуникации: широковещательная рассылка данных 1. Изучите MPI-функцию широковещательной рассылки данных MPI_Bcast. Напишите MPI-программу, которая в строке длины n определяет количество вхождений символов. Ввод данных должен осуществляться процессом с номером 0. Для рассылки строки поиска и ее длины по процессам используйте функцию MPI_Bcast.

Код:

```
MPI_Init(&argc, &argv);
{
    MPI_Status status;
    MPI_Request request;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0)
    {
        std::cout << "Введите длину строки(до 50)\n";
        std::cin >> leng;
        std::cout << "\n";
        for (auto i = 0; i < leng; i++)
        {
            std::cin >> buffer[i];
        }
    }
    MPI_Bcast(&leng, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&buffer, leng, MPI_CHAR, 0, MPI_COMM_WORLD);
    int count = 0;
    for (int i = rank; i < 26; i = i + size)
    {
        for (int j = 0; j < leng; j++)
        {
            if (buffer[j] == (char)((int)start + i))
            {
                count++;
            }
        }

        std::cout << "[" << (char)(start+i) << "] counts " << count << "\n";
        count = 0;
    }
}
MPI_Finalize();
```

Результат:

```
C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\WUZ\6term\PP\ConsoleApplication2\x64\Release>"C:\Users\Admin\Desktop\WUZ\6term\PP\ConsoleApplication2\ConsoleApplication2\MPI\Bin\mpiexec.exe" -np 4 ConsoleApplication2.exe
тхфмх фмхмх #ЕЕмх(фм 50)
20
hello world created by mpi
[d] counts 2
[h] counts 1
[l] counts 3
[p] counts 0
[t] counts 1
[x] counts 0
[b] counts 1
[f] counts 0
[j] counts 0
[n] counts 0
[r] counts 2
[v] counts 0
[z] counts 0
[a] counts 1
[e] counts 3
[i] counts 0
[m] counts 1
[q] counts 0
[u] counts 0
[y] counts 1
[c] counts 1
[g] counts 0
[k] counts 0
[o] counts 2
[s] counts 0
[w] counts 1
C:\Users\Admin\Desktop\WUZ\6term\PP\ConsoleApplication2\x64\Release>PAUSE
Для продолжения нажмите любую клавишу . . .
```

24)

Задание:

Коллективные коммуникации: операции редукции 1. Изучите MPI-функцию для выполнения операций редукции над данными, расположенными в адресных пространствах различных процессов, `MPI_Reduce`. Реализуйте программу вычисления числа π , используйте функцию `MPI_Reduce` для суммирования результатов, вычисленных каждым процессом

Код:

```
MPI_Init(&argc, &argv);
{
    MPI_Status status;
    MPI_Request request;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    for (auto i = rank; i < eps; i += size)
    {
        x = (i + 0.5) / eps;
        sum += (4 / (1 + x * x)) / eps;
    }
    float FinallySum = 0;
    MPI_Reduce(&sum, &FinallySum, 1, MPI_FLOAT, MPI_SUM, 0, MPI_COMM_WORLD);
    if (rank == 0)
    {
        std::cout << "pi = " << FinallySum << "\n";
    }
}
MPI_Finalize();
```

Результат:

```
C:\Users\Admin\Desktop\VUZ\6term\PP\ConsoleApplication2\x64\Release>"C:\Users\Admin\Desktop\VUZ\6term\PP\ConsoleApplication2\ConsoleApplication2\MPI\Bin\mpiexec.exe" -np 4 ConsoleApplication2.exe
pi = 3.14159

C:\Users\Admin\Desktop\VUZ\6term\PP\ConsoleApplication2\x64\Release>PAUSE
Для продолжения нажмите любую клавишу . . . █
```

25)

Задание:

Коллективные коммуникации: функции распределения и сбора данных 1. Изучите MPI-функции распределения и сбора блоков данных по процессам MPI_Scatter и MPI_Gather. Напишите программу, которая вычисляет произведение двух квадратных матриц $A \times B = C$ размера $n \times n$. Используйте формулу, приведенную в задании 9. Ввод данных и вывод результата должны осуществляться процессом с номером 0. Для распределения матриц A и B и сбора матрицы C используйте функций MPI_Scatter и MPI_Gather

Код:

```
int rank; // номер процесса в приложении

int size; // кол-во процессов в приложении

int root = 0; // мастер поток

int tag = 0;

// матрицы

int** matrixA = NULL; int** matrixB = NULL; int* arrA = NULL; int* arrC = NULL;

// параллельная область
MPI_Init(&argc, &argv);

MPI_Con«_rank(MPI_COMM_WORLD, &rank);

MPI_Comm_size(MPI_COMM_WORLD, &size);

OutMatrix(matrixB, n, rank);

// рассылка n всем потокам
```

```

MPI_Bcast(&n, 1, MPI_INT, root, MPI_COMM_WORLD);

// разложение строк по потокам int* strA = new int[n];

MPI_Scatter(arrA, n, MPI_INT, strA, n, MPI_INT, root, MPI_COMM_WORLD);
MPI_Barrier(MPI_COMM_WORLD);

for (int i = 0; i < n; i++)
{
    MPI_Bcast(*(matrixB + i), n, MPI_INT, root, MPI_COMM_WORLD);
}

// вычисление строки матрицы C int* strC = new int[n] {0};

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        strC[i] += strA[j] * matrix[j][i];
    }
}

MPI_Gather(strC, n, MPI_INT, arrC, n, MPI_INT, root, MPI_COMM_WORLD);

// вывод матрицы C if (rank == root) {
printf("\nMatrixC: \n");

```

Результат:

```

[0] thread Out Matrix
1 3
4 8
[0] thread Out Matrix
5 4
3 0

MatrixC:
0 - thread where arr =
14 4 44 16

```