Михайлова К.В. 6214-020302D

Дополнительное задание по курсу “Операционные системы”

Подключение MPI

Скачивание дополнительных продуктов, для подключения #include<mpi.h> в VS, скачать можно на официальном сайте Microsoft
https://www.microsoft.com/en-us/download/details.aspx?id=57467

Далее необходимо подключить MPI к проекту, в данном видео всё подробно рассказано
https://www.youtube.com/watch?v=dhxW4ZoZQdI&ab_channel=LandLord

После всех этапов можно запустить приведенный ниже код

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


int main(int argc, char* argv[])
{
    int i, j, k, l;
    int* Matr_1, * Matr_2, * Result_matr, * Buffer, * Ans;
    int Size = 3;
    int Rank_proc, Num_procs, Line;


    MPI_Init(NULL, NULL);
    MPI_Comm_rank(MPI_COMM_WORLD, &Rank_proc);//  текущий номер процесса
    MPI_Comm_size(MPI_COMM_WORLD, &Num_procs);//  количество процессов

    Line = Size / Num_procs;// Делим данные на блоки (количество процессов), и основной процесс также должен обрабатывать данные
    Matr_1 = (int*)malloc(sizeof(int) * Size * Size);
    Matr_2 = (int*)malloc(sizeof(int) * Size * Size);
    Result_matr = (int*)malloc(sizeof(int) * Size * Size);
    // Размер кеша больше или равен размеру обрабатываемых данных, когда он больше, чем фактическая часть данных
    Buffer = (int*)malloc(sizeof(int) * Size * Line);// Размер данных
    Ans = (int*)malloc(sizeof(int) * Size * Line);// Сохраняем результат расчета блока данных

    // Основной процесс присваивает матрице начальное значение и передает матрицу N каждому процессу, а матрицу M передает каждому процессу в группах.
    if (Rank_proc == 0) {

        printf("Matr 1:\n");
        for (i = 0; i < Size; i++) {
            for (j = 0; j < Size; j++) {
                Matr_1[i * Size + j] = rand() % 10;
                printf("%4d", Matr_1[i * Size + j]);
            }
            printf("\n");
        }

        printf("Matr 2:\n");
        for (i = 0; i < Size; i++) {
            for (j = 0; j < Size; j++) {
                Matr_2[i * Size + j] = rand() % 10;
                printf("%4d", Matr_2[i * Size + j]);
            }
            printf("\n");
```

```
        }
        // Отправить матрицу N другим подчиненным процессам
        for (i = 1; i < Num_procs; i++) {
            MPI_Send(Matr_2, Size * Size, MPI_INT, i, 0, MPI_COMM_WORLD);
        }
        // Отправляем каждую строку Matr_1 каждому подчиненному процессу по очереди
        for (l = 1; l < Num_procs; l++) {
            MPI_Send(Matr_1 + (l - 1) * Line * Size, Size * Line, MPI_INT, l, 1,
MPI_COMM_WORLD);
        }
        // Получаем результат, рассчитанный по процессу
        for (k = 1; k < Num_procs; k++) {
            MPI_Recv(Ans, Line * Size, MPI_INT, k, 3, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            // Передаем результат в массив Result_matr
            for (i = 0; i < Line; i++) {
                for (j = 0; j < Size; j++) {
                    Result_matr[((k - 1) * Line + i) * Size + j] = Ans[i * Size + j];
                }
            }
        }
        // Рассчитать оставшиеся данные
        for (i = (Num_procs - 1) * Line; i < Size; i++) {
            for (j = 0; j < Size; j++) {
                int temp = 0;
                for (k = 0; k < Size; k++) {
                    temp += Matr_1[i * Size + k] * Matr_2[k * Size + j];
                }
                Result_matr[i * Size + j] = temp;
            }
        }

        printf("Result:\n");
        for (i = 0; i < Size; i++) {
            for (j = 0; j < Size; j++) {
                printf("%4d ", Result_matr[i * Size + j]);
            }
            printf("\n");
        }


        free(Matr_1);
        free(Matr_2);
        free(Result_matr);
        free(Buffer);
        free(Ans);
    }

    // Другие процессы получают данные и после вычисления результата отправляют их в
основной процесс
    else {
        // Получаем данные (матрица Matr_2)
        MPI_Recv(Matr_2, Size * Size, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

        MPI_Recv(Buffer, Size * Line, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        // Рассчитать результат и отправить его в основной процесс
        for (i = 0; i < Line; i++) {
            for (j = 0; j < Size; j++) {
                int temp = 0;
                for (k = 0; k < Size; k++) {
                    temp += Buffer[i * Size + k] * Matr_2[k * Size + j];
                }
                Ans[i * Size + j] = temp;
            }
        }
        // Отправить результат расчета в основной процесс
```

```
        MPI_Send(Ans, Line * Size, MPI_INT, 0, 3, MPI_COMM_WORLD);
    }

    MPI_Finalize();

    return 0;
}
```

Чтобы запустить код на нескольких процессах, нужно открыть командную строку (win+r), далее написать “cmd”. В открытом окне вам нужно написать путь до папки Debug Вашей программы, а далее написать mpiexec –n 4 НАЗВАНИЕВАШЕГОФАЙЛА.exe