**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №01**

**по дисциплине «Параллельные алгоритмы»**

Тема: Обмен сообщениями четных и нечетных процессов

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9382 |  | Иерусалимов Н. |
| Преподаватель |  | Татаринов Ю.С. |

Санкт-Петербург

2021

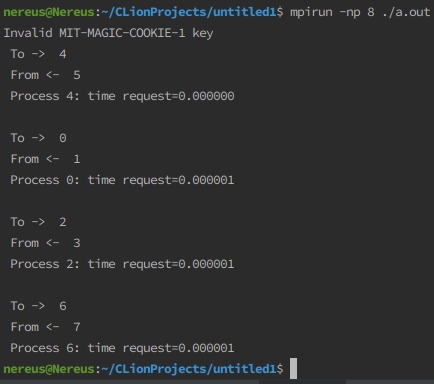
**Цель работы**

Научиться управлять взаимодействием разных процессов, замерять время работы процессов.

**Формулировка задания**

Задание: Напишите программу обмена сообщениями чётных и нечётных процессов. Замерьте время на одну итерацию обмена и определите зависимость времени обмена от длины сообщения.

#include <stdio.h>  
#include "mpi.h"  
int main(int argc, char\* argv[]) {  
 int ProcNum, ProcRank, RecvRank;  
 double start, finish;  
 MPI\_Status Status;  
 MPI\_Init(&argc, &argv);  
 MPI\_Comm\_size(MPI\_COMM\_WORLD, &ProcNum);  
 MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);  
 if (ProcRank % 2 == 0) {  
 start= MPI\_Wtime();  
 if (ProcRank < ProcNum - 1) {  
 printf("\n To ->%3d", ProcRank);  
 MPI\_Recv(&RecvRank, 1, MPI\_INT, ProcRank + 1,MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &Status);  
 printf("\n From <-%3d", RecvRank);  
 }  
 if (ProcRank == ProcNum - 1) {  
 printf("\n Single process %3d", ProcRank);  
 }  
 finish = MPI\_Wtime();  
  
 printf("\n Process %d: time request=%lf\n", ProcRank, (finish - start) / 100);  
 }  
 else  
 MPI\_Send(&ProcRank, 1, MPI\_INT, ProcRank - 1, 0, MPI\_COMM\_WORLD);  
 MPI\_Finalize();  
 return 0;  
}

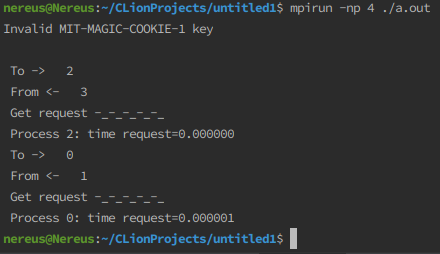
 рис. 1 — результат работы программы на N процессорах

Нечетные процессы отправляют сообщения, а четные – их принимают. Если же процессору не хватает пары то выводится его номер с приставкой Single

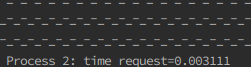
**Разные длины пересылаемых сообщений**

Выполнение программы не зависит от числа процессоров, так как работа происходит попарно. При увеличении длинны сообщения процесс обмена происходит медленнее на незначительное количество времени.

#include <stdio.h>  
#include "mpi.h"  
  
int main(int argc, char\* argv[]) {  
 int ProcNum, ProcRank, RecvRank;  
 double time\_start, time\_finish;  
 int N = 50 000;  
 char text[N + 1];  
 for (int i = 0; i < N; i++)  
 if(i%2==0) text[i] = '-';  
 else text[i] = '\_';   
  
 text[N] = '\0';  
 MPI\_Status Status;  
 MPI\_Init(&argc, &argv);  
 MPI\_Comm\_size(MPI\_COMM\_WORLD, &ProcNum);  
 MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);  
 if (ProcRank % 2 == 0) {  
 time\_start = MPI\_Wtime();  
 if (ProcRank < ProcNum - 1) {  
 printf("\n To -> %3d", ProcRank);  
 MPI\_Recv(&RecvRank, 1, MPI\_INT, ProcRank + 1,  
 MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &Status);  
 printf("\n From <- %3d", RecvRank);  
 MPI\_Recv(text, N + 1, MPI\_CHAR, ProcRank + 1, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &Status);  
 printf("\n Get request %s", text);  
 } if (ProcRank == ProcNum - 1) {  
 printf("\n Single %3d", ProcRank);  
 }  
 time\_finish = MPI\_Wtime();  
 printf("\n Process %d: time request=%lf\n", ProcRank, (time\_finish - time\_start) / 100);  
 }  
 else {  
 MPI\_Send(&ProcRank, 1, MPI\_INT, ProcRank - 1, 0, MPI\_COMM\_WORLD);  
 MPI\_Send(text, N + 1, MPI\_CHAR, ProcRank - 1, 0, MPI\_COMM\_WORLD);  
 }  
 MPI\_Finalize();  
 return 0;

 Рис. 2 — скорость при 10 символах

Разницу в скорости смог получить только на 50 000 символах

Рис. 3 — скорость при 50 000 символах

**Выводы**

В ходе лабораторной работы была написана и модифицирована программа MPI для обмена сообщениями между четными и нечетными процессами. Чем больше длина сообщения, тем больше затрачивается времени на обмен.