Национальный исследовательский университет «МИЭТ»

Отчет о выполненной лабораторной работе №2

По предмету: Технология программирования MPI

На тему:Коллективные операции передачи данных

Цель: изучить основные принципы коллективных операций передачи данных в технологии MPI на примере использования в рамках языка С++.

Выполнила Марина Алина

Группа ПИН-34

Вариант №1

02.03.2022

**Задание.** Модифицировать программу, написанную на Л.Р. №1, так чтобы она работала на основе коллективной передачи сообщений. **Результаты работы сравнить и занести в отчет.**

**Контрольные вопросы**

1.Как происходит передача данных от одного процесса всем?

Достижение эффективного выполнения операции передачи данных от одного процесса всем процессам программы (**широковещательная рассылка** данных) может быть обеспечено при помощи функции MPI:

int MPI\_Bcast(void \*buf,int count,MPI\_Datatype type,int root,MPI\_Comm comm);

2.Как происходит передача данных от всем процессов одному?

Реализация операции редукции при помощи обычных парных операций передачи данных является неэффективной и достаточно трудоемкой. Для наилучшего выполнения действий, связанных с редукцией данных, в MPI предусмотрена функция:

int MPI\_Reduce(void \*sendbuf, void \*recvbuf,int count,MPI\_Datatype type,

MPI\_Op op,int root,MPI\_Comm comm);

3.Какие используются в MPI для синхронизации вычислений?

**Синхронизация** процессов, т.е. одновременное достижение процессами тех или иных точек процесса вычислений, обеспечивается при помощи функции MPI:

int MPI\_Barrier(MPI\_Comm comm);

4.Как организуется неблокирующий обмен данными между процессами?

Список параметров неблокирующих функций содержит весь набор параметров исходных функций и один дополнительный параметр **request** с типом **MPI\_Request** (в функции **MPI\_Irecv** отсутствует также параметр **status**):

int MPI\_Isend(void \*buf, int count, MPI\_Datatype type, int dest,

int tag, MPI\_Comm comm, MPI\_Request \*request);

int MPI\_Issend(void \*buf, int count, MPI\_Datatype type, int dest,

int tag, MPI\_Comm comm, MPI\_Request \*request);

int MPI\_Ibsend(void \*buf, int count, MPI\_Datatype type, int dest,

int tag, MPI\_Comm comm, MPI\_Request \*request);

int MPI\_Irsend(void \*buf, int count, MPI\_Datatype type, int dest,

int tag, MPI\_Comm comm, MPI\_Request \*request);

int MPI\_Irecv(void \*buf, int count, MPI\_Datatype type, int source,

int tag, MPI\_Comm comm, MPI\_Request \*request);

5.Как организуется одновременное выполнение прием и передачи данных?

Достижение эффективного и гарантированного одновременного выполнения операций передачи и приема данных может быть обеспечено при помощи функции MPI:

int MPI\_Sendrecv(void \*sbuf,int scount,MPI\_Datatype stype,int dest, int stag,

void \*rbuf,int rcount,MPI\_Datatype rtype,int source,int rtag,

MPI\_Comm comm, MPI\_Status \*status);