Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Дальневосточный федеральный университет»

(ДВФУ)

Капитан В.Ю.

Лабораторная работа 2

Процедуры буферизованного и неблокирующего

двухточечного обмена

Владивосток

2022

Двухточечный буферизованный обмен

При передаче сообщения в буферизованном режиме источник копирует сообщение в буфер, а затем передает его в неблокирующем режиме. Выделение буфера и его размер контролируются программистом, который должен заранее создать буфер достаточного размера. Буферизованная передача завершается сразу, поскольку сообщение немедленно копируется в буфер для последующей передачи.

После завершения работы с буфером его необходимо отключить. После отключения буфера можно вновь использовать занимаемую им память, однако следует помнить, что в языке C данный вызов не освобождает автоматически память, отведенную для буфера.

Буферизованный обмен рекомендуется использовать в тех ситуациях, когда программисту требуется больший контроль над распределением памяти.

Двухточечные неблокирующие обмены

Вызов подпрограммы неблокирующей передачи инициирует, но не завершает ее. Передача данных из буфера или их считывание происходит одновременно с выполнением других операций. Завершается обмен вызовом дополнительной процедуры, которая проверяет, скопированы ли данные в буфер передачи. До завершения обмена запись в буфер или считывание из него производить нельзя, так как сообщение может быть еще не отправлено или не получено. Неблокирующая передача может быть принята подпрограммой блокирующего приема и наоборот. Неблокирующий обмен выполняется в два этапа:

1. Инициализация обмена.
2. Проверка завершения обмена.

Для маркировки неблокирующих операций обмена используются идентификаторы операций обмена.

Проверка фактического выполнения передачи или приема в неблокирующем режиме осуществляется с помощью вызова подпрограмм ожидания, блокирующих работу процесса до завершения операции или неблокирующих подпрограмм проверки, возвращающих логическое значение "истина", если операция выполнена.

Подпрограмма MPI\_Wait блокирует работу процесса до завершения приема или передачи сообщения. Функции MPI\_Wait и MPI\_Test можно использовать для завершения операций приема и передачи.

Подпрограммы-пробники

Получить информацию о сообщении до его помещения в буфер приема можно с помощью подпрограмм-пробников MPI\_Probe и MPI\_IProbe. На основании полученной информации принимается решение о дальнейших действиях. С помощью вызова подпрограммы MPI\_Probe фиксируется поступление (но не прием!) сообщения. Затем определяется источник сообщения, его длина, выделяется буфер подходящего размера и выполняется прием сообщения.

**Лабораторная работа**

В заданиях лабораторной работы предлагается дописать предлагаемые фрагменты программ на языках C, написанные с использованием процедур MPICH 1.2.7.

Пропущенные фрагменты обозначены многоточием.

Задание 2.1

Пропущены вызовы процедур буферизованного обмена. Исправить код, откомпилировать и запустить программу. Исправить ошибки в тексте

#include "mpi.h" #include <stdio.h>

int main(int argc,char \*argv)

{ int \*buffer; int murank; MPI\_Status status; int buffsize = 1 int TAG = 0;

… …

if (myrank == 0)

{

buffer = (int \*) malloc(buffsize + MPI\_BSEND\_OVERHEAD);

…

buffer = (int \*) 10;

… } else

{

MPI\_Recv(&bufer, bufsize, MPI\_BOOL, 0, TAG, MPI\_COMM\_WORLD, &status); printf("received: %i\n", bufer);

}

MPI\_Finalize(); return 0;

}

Задание 2.2

Отсутствуют вызовы подпрограмм-пробников. Добавить эти вызовы, откомпилировать и запустить программу. Исправить ошибки в тексте

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h> #include <math.h>

int main(int argc,char \*\*argv)

{ int myrank, numprocs; MPI\_Status status;

int mytag,kk,count\_k,j,\*mas;

…

…

…

printf(" Hello from c process: %d Numprocs is

%d\n",myrank,numprocs);

mytag=150; if(myrank == 0) { j=200; count\_k=1;

kk=MPI\_Send(&j,count\_k,MPI\_INT,1,mytag,MPI\_COMM\_WORLD)

}

if(myrank == 1){ kk=MPI\_Probe(…); kk=MPI\_Get\_count(…); mas=(int\*)malloc(count\_k\*sizeof(int)); printf("getting %d\n",count\_k); kk = MPI\_Recv(…); printf("mas= "); for(j=0;j<count\_k;j++) printf("%d ",mas[j]);

printf("\n");

}

MPI\_Finalize();

}

Задание 2.3

Пропущены вызовы процедур неблокирующих операций обмена. Добавить эти вызовы, откомпилировать и запустить программу. Исправить ошибки в тексте

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

#include <math.h>

int main(argc,argv) int argc; char \*argv[];

{ int myrank, numprocs; int tag, source; int purpose, count; int buffer; MPI\_Status status;

MPI\_Request request;

…

… … tag=1500; source=0; purpose=1; count=1;

request=MPI\_REQUEST\_NULL; if(myrank == source){ buffer=6780;

MPI\_Isend(&buffer,count,MPI\_DOUBLE,purpose,tag,MPI\_COMM\_WORLD,

&request); }

if(myrank == purpose){

MPI\_Irecv(…);

}

MPI\_Wait(&request,&status); if(myrank == source){

printf("processor %d sent %d\n",murank,bufer)

}

if(myrank == purpose){

printf("processor %d receive %d\n",myrank,bufer)

}

MPI\_Finalize();

}