МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Кафедра "Программное обеспечение вычислительной техники и автоматизированных систем"

Технология MPI

Методические указания к практическим работам по дисциплине "Технологии высокопроизводительных вычислений"

Ростов-на-Дону

20 г.

Составитель: к.ф.-м.н., доц. Габрельян Б.В.

УДК 512.3

Технология MPI: методические указания – Ростов н/Д: Издательский центр ДГТУ, 20 . – с.

В методической разработке рассматриваются вопросы поддержки параллельного программирования в технологии MPI. Даны задания по выполнению лабораторной работы. Методические указания предназначены для магистрантов направления 09.04.04 – «Программная инженерия».

© Издательский центр ДГТУ, 20

Цель работы: Ознакомление с технологией MPI.

I. Основы MPI.

Интерфейс передачи сообщений (Message Passing Interface – MPI) предназначен, прежде всего, для систем с разделяемой памятью и представляет собой программный интерфейс для систем параллельных вычислений. Поддерживаются языки программирования Си/C++ и Fortran. Мы в дальнейшем будем рассматривать только реализацию для Си/C++. Классическим примером использования MPI является вычислительный кластер, узлы которого состоят из автономных вычислительных устройств с собственным процессором (или процессорами), памятью и другими ресурсами. Программа размещается на каждом узле кластера и, по команде с одного из этих узлов, запускается на выполнение. Во время работы процессы, размещенные на разных узлах, взаимодействуют друг с другом обмениваясь сообщениями. Процессы, выполняющие некоторую общую задачу можно объединить в группу. Каждый процесс в группе имеет свой ранг (номер). Ранг это целое число, которое может принимать значение от нуля до N-1, если N – число процессов в группе. Среду, обеспечивающую поддержку передачи и посылки сообщений, обеспечивает некоторая логическая конструкция MPI, называемая коммуникатором. Есть два вида коммуникаторов, обеспечивающие взаимодействие процессов внутри группы или взаимодействие между двумя группами.

В стандарте MPI предопределены некоторые типы, например, MPI_Group, MPI_Comm. Вообще, почти все имена типов и функций MPI начинаются с префикса MPI_.

Функция int MPI_Group_size(MPI_Group group, int *size) возвращает количество процессов в группе.

int MPI_Group_rank(MPI_Group group, int *rank) возвращает ранг того процесса в группе group, в котором вызвана эта функция.

Изначально определена группа MPI_COMM_GROUP, в которой можно определять другие группы. С этой группой связан коммуникатор MPI_COMM_WORLD. Группу можно создавать так, чтобы она не была связана с коммуникатором, тогда затем нужно будет явно обеспечить эту связь, либо же можно при создании коммуникатора создать новую группу потоков.

Связь уже существующей группы с коммуникатором осуществляет функция

int MPI_Comm_group(MPI_Comm comm, MPI_Group *group).

Над группами можно выполнять операции объединения, пересечения, разности.

int MPI_Group_union(MPI_Group group1, MPI_Group group2, MPI_Group *newGroup)

int MPI_Group_intersection(MPI_Group group1, MPI_Group group2, MPI_Group *newGroup)

int MPI_Group_difference(MPI_Group group1, MPI_Group group2, MPI_Group *newGroup)

Функция int MPI_Group_free(MPI_Group *group) освобождает объект group и назначает ему значение MPI_GROUP_NULL (признак ошибки).

int MPI_Comm_size(MPI_Comm comm, int *size) возвращает количество процессов в коммуникаторе comm.

int MPI_Comm_rank(MPI_Comm comm, int *rank) возвращает ранг запрашивающего процесса в коммуникаторе comm.

int MPI_Comm_create(MPI_Comm comm, MPI_Group group1, MPI_Comm *newComm) создает новый коммуникатор, а функция

int MPI_Comm_free(MPI_Comm *comm) уничтожает указанный коммуникатор.

MPI требует начальной инициализации системы до ее использования и финализации при завершении работы. Для этого используются функции

int MPI_Init(int *argc, char ***argv) и

int MPI_Finalize().

Поэтому общей схемой для MPI-программы является следующая

```
#include "mpi.h"


int main(int argc, char *argv[]) {
    MPI_Init(&argc,&argv);
    …
    MPI_Finalize();
```

```
    return 0;
}
```

II. Посылка и прием сообщений.

Сообщения могут пересылаться между двумя процессами, так называемый протокол "точка-точка" (point-to-point) либо между многими процессами внутри группы.

Операции точка-точка могут быть блокирующими или неблокирующими, буферизованными, синхронными или ориентированными на состояние.

1) Блокированная передача данных

Посылка сообщения

int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dstRank, int tag, MPI_Comm comm), здесь buf – буфер, содержащий передаваемые данные, количество передаваемых данных в буфере, datatype – тип данных в буфере, dstRank – ранг процесса, которому передаются данные), tag – тэг сообщения, comm – коммуникатор.

Некоторые встроенные типы данных MPI указаны в таблице.

MPI-тип	Си-тип
MPI_CHAR	unsigned char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int

MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	
MPI_PACKED	

Прием сообщения

int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int srcRank, int tag, MPI_Comm comm, MPI_Status *status), здесь MPI_Status – это структура с тремя полями MPI_SOURCE, MPI_TAG, MPI_ERROR. count задает максимальное количество принимаемых элементов типа datatype, чтобы узнать, сколько элементов передано на самом деле нужно вызвать функцию

int MPI_Get_count(MPI_Status *status, MPI_Datatype datatype, int *count).

2) Неблокированная передача данных

Посылка сообщения

int MPI_Isend(void *buf, int count, MPI_Datatype datatype, int dstRank, int tag, MPI_Comm comm, MPI_Request *request), здесь request используется для того, чтобы запрашивать состояние связи или ждать ее завершения.

Прием сообщения

int MPI_Irecv(void *buf, int count, MPI_Datatype datatype, int srcRank, int tag, MPI_Comm comm, MPI_Request *request)

Для завершения неблокированной посылки или передачи данных используется функция

int MPI_Wait(MPI_Request *request, MPI_Status *status), а для проверки того, завершена передача или нет – функция

int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status). Если flag не равен нулю, значит операция, заданная параметром request завершилась.

Пример. Программа запущена на нескольких узлах и узел с номером (рангом) ноль посылает сообщение с целым значением 120, а узел с номером 1 принимает его. Коммуникатор задан переменной comm, тэг – в переменной tag.

```
…
int rank;
int data; // данные для пересылки или принимаемые
MPI_Comm_rank(comm, &rank);
if( !rank ) // если rank == 0 послать сообщение
        MPI_Send(&data,1,MPI_INT,1,tag,comm);
else if( rank == 1 ) { // принять сообщение
        MPI_Status status;
        MPI_Recv(&data,1,MPI_INT,0, tag, comm, &status);
}
```

III. Создание и запуск на выполнение MPI-приложений.

Существует множество реализаций стандарта MPI. Наиболее известными являются MPICH2, OpenMPI и Intel MPI. Первые две являются открытыми разработками и могут быть получены, соответственно, на сайтах http://www.mpich.org/, http://www.open-mpi.org/.

Нужно установит какую-нибудь версию реализации MPI. Любая реализация предоставляет набор утилит для создания и запуска MPI-приложений. Для компиляции Си-программ нужно использовать компилятор mpicc, например, если программа записана в файл mpi_prog.c и нужно получить исполняемый файл с именем mpi_prog_exe

```
mpicc –o mpi_prod_exe mpi_prog.c
```

Если программу нужно запустить параллельно на 8 компьютерах вычислительного кластера (или на восьми ядрах единственного процессора) нужно выполнить команду

```
mpirun –n 8 mpi_prog_exe
```

В распределенной вычислительной системе обычно используется запуск с ключом –mashinesfile, через который передается текстовый файл, содержащий сетевые адреса компьютеров кластера. Например, если есть файл comps, содержащий

povtias1.dstu.edu.ru

physics.dstu.edu.ru

povtias2.dstu.edu.ru

команда запуска будет выглядеть так

mpirun –mashinesfile comps –n 9 mpi_prog_exe

На каждом из трех компьютеров кластера будет запущено по три процесса.

IV. Задания.

1. Реализуйте в виде собственных функций следующие алгоритмы численного интегрирования: прямоугольников, трапеций, Симпсона.

2. Создайте программу для тестирования созданных алгоритмов, вычисляющую значение числа π. В процессе тестирования нужно получать результат с разной точностью, замеряя при этом время вычисления.

3. Создайте параллельные версии алгоритмов, использующие возможности MPI.

4. Протестируйте параллельные версии алгоритмов. Измерьте время выполнения операций, сравните с последовательными версиями и сделайте выводы.

V. Контрольные вопросы.

1. Какими преимуществами по сравнению с другими технологиями распараллеливания обладает технология MPI?
2. Каковы ограничения технологии MPI?
3. Как инициализировать MPI?
4. Как послать сообщение в стандарте MPI?

5. Как принять сообщение в стандарте MPI?
6. Какие реализации стандарта MPI Вы знаете?
7. Как задать выполнение приложения на нескольких узлах?

Литература

1. Корнеев В.Д. "Параллельное программирование в MPI", 2-е изд. – Новосибирск: ИВМиМГ СО РАН, 2002. – 215 с.

2. "MPI: A message passing interface standard version 3.0" – University of Tennessee, 1993-1997, 2008, 2009, 2012. – 852 p.

3. Лупин С.А., Посыпкин М.А. "Технологии параллельного программирования". – М.: ИД "Форум"; ИНФА-М, 2011. – 208 р.

4. Антонов А.С. "Параллельное программирование с использованием технологии MPI", учебное пособие. – М.: МГУ, 2004. – 71 с.