Лабораторная работа № 4

Средства MPI для обмена сообщениями

Цель

Ознакомиться со средствами технологии MPI для передачи сообщений между процессами и получить практический навык их использования.

Теоретические сведения

Основным средством коммуникации между процессами в MPI является передача сообщений друг другу. Собственно говоря, MPI — это и есть набор функций для передачи сообщений между процессами.

Все функции MPI имеют схожее название. Для C/C++, к примеру, все функции MPI начинаются с префикса *MPI_* (обратите внимание, что в связи со спецификой языка C, регистр в данном случае имеет значение).

Подробно со всеми функциями и их спецификациями можно ознакомиться в опубликованном стандарте MPI, сопроводительной документации к вашей реализации MPI, а также в учебном пособии Антонова А.С. «Параллельное программирование с использованием технологии MPI».

Примеры программ

Рассмотрим пример простой программы на C++, осуществляющей обмен сообщениями между процессами. Обмен сообщениями сводится к тому, что главному процессу все остальные процессы отправляют следующую информацию: свой номер, общее количество процессов, имя станции, на которой запущен процесс.

Обратите внимание, так как программа написана на C++ (в отличие от предыдущего примера, написанного на C), используется объектная нотация и пространства имен.

C++ Binding:

```
#include "mpi.h"
#include <iostream>

int main (int argc, char *argv[])
{
```

```
 // Объявляем переменные
 int i, rank, size, namelen;
 char name[MPI_MAX_PROCESSOR_NAME];
 MPI::Status stat;
 // Инициализируем подсистему MPI
 MPI::Init(argc, argv);
 // Получаем информацию о процессе s
 size = MPI::COMM_WORLD.Get_size();
 rank = MPI::COMM_WORLD.Get_rank();
 MPI::Get_processor_name(name, namelen);

 if (rank == 0)
 {
  // Находимся в головном процессе
  std::cout << "Hello world: rank " << rank << " of "
    << size << " running on " << name << "\n";

  // Собираем информацию от всех прочих процессов
  for (i = 1; i < size; i++)
  {
    MPI::COMM_WORLD.Recv (&rank, 1, MPI_INT, i, 1, stat);
    MPI::COMM_WORLD.Recv (&size, 1, MPI_INT, i, 1, stat);
    MPI::COMM_WORLD.Recv (&namelen, 1, MPI_INT, i, 1,stat);
    MPI::COMM_WORLD.Recv (name, namelen + 1, MPI_CHAR, i, 1, stat);
    std::cout << "Hello world: rank " << rank << " of "
         << size << " running on " << name << "\n";
  }
 }
 else
 {
 // Находимся в дочернем процессе
 // Необходимо отправить информацию о себе головному процессу
  MPI::COMM_WORLD.Send(&rank, 1, MPI_INT, 0, 1);
  MPI::COMM_WORLD.Send(&size, 1, MPI_INT, 0, 1);
  MPI::COMM_WORLD.Send(&namelen, 1, MPI_INT, 0, 1);
  MPI::COMM_WORLD.Send(name, namelen + 1, MPI_CHAR, 0, 1);
 }
 // Освобождаем подсистему MPI
 MPI::Finalize ();

 return (0);
}
```

C Binding:

```
#include "mpi.h"
#include <iostream>

int main (int argc, char *argv[])
{
  // Объявляем переменные
int i, rank, size, namelen;

char name[MPI_MAX_PROCESSOR_NAME];

MPI_Status stat;

// Инициализируем подсистему MPI
  MPI_Init(&argc, &argv);
```

```
// Получаем информацию о процессе s
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Get_processor_name(name, &namelen);

if (rank == 0) {

   // Находимся в головном процессе

std::cout << "Hello world: rank " << rank << " of "
     << size << " running on " << name << "\n";


// Собираем информацию от всех прочих процессов

for (i = 1; i < size; i++)

{

MPI_Recv (&rank, 1, MPI_INT, i, 1, MPI_COMM_WORLD, &stat);
MPI_Recv (&size, 1, MPI_INT, i, 1, MPI_COMM_WORLD, &stat);
MPI_Recv (&namelen, 1, MPI_INT, i, 1, MPI_COMM_WORLD, &stat);

MPI_Recv (name, namelen + 1, MPI_CHAR, i, 1, MPI_COMM_WORLD, &stat);

std::cout << "Hello world: rank " << rank << " of "
          << size << " running on " << name << "\n";

}

 }

  else

  {
  // Находимся в дочернем процессе

// Необходимо отправить информацию о себе головному процессу

MPI_Send(&rank, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);

MPI_Send(&size, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);

MPI_Send(&namelen, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);

MPI_Send(name, namelen + 1, MPI_CHAR, 0, 1, MPI_COMM_WORLD);

}

 // Освобождаем подсистему MPI

  MPI_Finalize ();


  return (0);

}
```

Задания к выполнению лабораторной работы

Ознакомиться со средствами для обмена сообщениями между процессами, присутствующими в технологии MPI. Теоретический материал взять из учебного пособия Антонова А.С. «Параллельное программирование с использованием технологии MPI», главы: «Основные понятия», «Общие процедуры MPI», «Передача/прием сообщений между отдельными процессами», «Коллективное взаимодействие процессов».

Разработать распределенную программу, выполняющую обмен сообщениями и провести исследование ее возможностей согласно своему варианту.

Вариант №1

Топология «звезда». Дочерние процессы пересылают пакеты данных заданного объема центральному процессу следующим образом: пакет рассылается каждым дочерним процессом двум соседям, откуда они пересылаются центральному процессу.

Оценить время рассогласованности получения пары пакетов центральным процессом от дочерних.

Вариант №2

Топология «звезда». Каждый дочерний процесс пересылает центральному произвольный пакет данных. Центральный процесс ретранслирует полученный от дочернего процесса пакет всем прочим процессам (в том числе и отправителю).

Оценить время пересылки единицы информации в зависимости от количества дочерних процессов.

Вариант №3

Топология «звезда». Центральный процесс рассылает поочередно дочерним процессам пакет данных произвольной длины. Дочерние процессы, получая от центрального пакет данных, отсылают его назад.

Оценить время пересылки единицы информации в зависимости от размера пакета.

Вариант №4

Топология «звезда». Центральный процесс пересылает всем дочерним процессам произвольный пакет данных, после чего центральным процессом назначается другой, и итерация повторяется.

Отранжировать процессы по скорости работы в качестве центрального процесса.

Вариант №5

Топология «кольцо». Процессы пересылают по кругу пакет данных, длина которого после каждой пересылки увеличивается до некоторого предела.

Оценить время пересылки единицы информации в зависимости от размера пакета.

Вариант №6

Топология «кольцо». Процессы пересылают по кругу в противоположных направлениях 2 пакета данных. Всякий раз, когда оба пакета оказываются в пространстве одного процесса, они увеличиваются в размере

Построить зависимость скорости роста пакетов от количества процессов.

Вариант №7

Топология «кольцо». Процессы пересылают по кругу пакет данных произвольной длины. После того как пакет сделает полный оборот, запускается другой пакет, но уже из другого процесса и итерация повторяется.

Оценить время оборота пакета в зависимости от количества процессов.

Вариант №8

Топология «два кольца». Все процессы условно делятся пополам. Каждая половина процессов пересылает по кругу пакет данных произвольной длины.

Оценить время оборота пакета в зависимости от количества процессов.

Вариант №9

Топология «конвейер». Головной процесс пересылает пакет данных произвольного размера через все процессы замыкающему. После каждой пересылки пакет увеличивается в размере.

Оценить время пересылки единицы информации в зависимости от количества процессов.

Вариант №10

Топология «конвейер». Головной процесс пересылает пакет данных произвольного размера через все процессы замыкающему. Замыкающий процесс, получив пакет данных, пересылает его тем же путем головному.

Оценить время пересылки единицы информации в зависимости от размера пакета.

Вариант №11

Топология «конвейер». Среди процессов выделяется один центральный процесс. Все остальные процессы условно делятся пополам и образуют два конвейера (слева и справа от центрального). Замыкающие процессы с каждой из сторон пересылают пакеты через процессы своего конвейера центральному.

Оценить скорость прохождения пакетов по левому и правому конвейеру в зависимости от размера пакета.

Вариант №12

Топология «две звезды». Среди процессов выделяется два ключевых, все остальные делятся поровну между выделенными в качестве подчиненных. Пакеты пересылаются по следующей схеме: дочерний процесс пересылает пакет своему центральному процессу, который в свою очередь пересылает его соседнему центральному процессу

Оценить скорость прохождения пакетов между центральными процессами в зависимости от количества процессов.

Вариант №13

Топология «две звезды». Среди процессов выделяется два ключевых, все остальные условно делятся поровну между выделенными в качестве

подчиненных. Каждый центральный процесс пересылает пакеты произвольной длины всем своим дочерним процессам.

Оценить время пересылки единицы информации в зависимости от размера пакета.

Вариант №14

Топология «песочные часы». Среди процессов выделяется центральный, а все остальные делятся пополам (будем считать, что одна половина из них располагается справа от центрального, а вторая слева). Каждый дочерний процесс пересылает пакеты данных избранному процессу с противоположной стороны, однако не напрямую, а через центральный процесс.

Оценить скорость передачи единицы информации в зависимости от количества процессов

Вариант №15

Топология «восьмерка». Среди процессов выделяется центральный, а все остальные делятся пополам (будем считать, что одна половина из них располагается справа от центрального, а вторая слева). Процессы с каждой стороны от центрального замыкаются с ним в кольцо. По каждому из полученных колец передается пакет.

Оценить скорость передачи единицы информации в зависимости от количества процессов

Содержание отчета

1. Название и цель работы.
2. Задание к выполнению лабораторной работы согласно варианту.
3. Топология обмена сообщениями между процессами в виде графа, а также словесное ее описание.
4. Программа в виде исходных кодов (с поясняющими комментариями), а также в откомпилированном виде для демонстрации на ЭВМ.

5. Примеры работы программы на тестовых данных. Расчеты выполнить как минимум на 4 разных порциях данных и как минимум на 4 разных количествах вычислителей. Рассчитать коэффициенты ускорения и построить графики зависимости коэффициента ускорения от количества используемых для расчета вычислителей.
6. Результаты исследования программы согласно варианту в виде таблиц или графиков с поясняющими комментариями.
7. Выводы по работе.