Стандарт параллельного программирования MPI
Простейшие задачи

Следующая программа на языке C++ дает иллюстрацию вычисления числа π по формуле средних прямоугольников (сравните с программой для вычисления π в стандарте OpenMP с использованием параллелизма низкого уровня).

```
1   #include <iostream>
2   #include <mpi.h>
3
4   using namespace std;
5
6   double f(double x)
7   {
8     return 1/(1+x*x);
9   }
10
11  int main(int argc, char *argv[])
12  {
13     double pi, sum = 0, term, h;
14     int myrank, nprocs, n, i;
15     MPI_Init(&argc, &argv);
16     MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
17     MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
18     if(myrank == 0)
19       {
20         cout<<"Number of iterations=";
21         cin>>n;
22       }
23     MPI_Bcast(&n,1,MPI_INT,0,MPI_COMM_WORLD);
24     h = 1.0/n;
25     for(i = myrank+1; i <= n; i += nprocs)
26         sum += f(h*(i-0.5));
27     term = 4*h*sum;
28     MPI_Reduce(&term, &pi, 1, MPI_DOUBLE, MPI_SUM,0,MPI_COMM_WORLD);
29     if(myrank == 0)
30         cout<<"Computed value of pi="<<pi<<endl;
31     MPI_Finalize();
32     return 0;
33  }
```

Строки 15, 16, 17 используются для вызова функций MPI. Рассмотрим вызываемые здесь функции подробнее.

Функция MPI_Init(&argc, &argv) инициализирует библиотеку MPI (ее вызов обязателен перед тем, как начать использование упомянутой библиотеки); передаваемые ей параметры служат для выборки из аргументов командной строки тех, которые относятся к библиотеке MPI.

Функция MPI_Comm_rank(MPI_COMM_WORLD, &myrank) определяет номер процесса (он присваивается переменной myrank) в группе с коммуникатором MPI_COMM_WORLD. Коммуникатор — некий специальный объект, идентифицирующий группу, т. е. набор процессов. Этот коммуникатор определяет группу всех процессов, запущенных для решения данной задачи.

Процессы в любой группе нумеруются целыми числами от 0 до (число процессов)−1.

Функция MPI_Comm_size(MPI_COMM_WORLD, &nprocs) позволяет определить число процессов в группе с коммуникатором MPI_COMM_WORLD (это число присваивается выходному параметру nprocs). Оно определяется при запуске программы на выполнение при помощи опции –np .

В строках 18–22 инициируется ввод числа n слагаемых в сумме квадратурной формулы из процесса с номером 0 (напомним, что в MPI привилегированного процесса не существует: роль корневого процесса может выполнять любой процесс по желанию пользователя).

В строке 23 работает процедура

```
MPI_Bcast(&n,1,MPI_INT,0,MPI_COMM_WORLD),
```

которая производит рассылку информации (в данном случае, числа n) из корневого процесса (т.е. процесса с номером 0) по всем процессам группы с коммуникатором MPI_COMM_WORLD; здесь

первый параметр – указатель на рассылаемые значения (в данном случае рассылается число n),

второй параметр – количество рассылаемых значений (в данном случае их количество равно 1, ибо рассылается одно число n),

третий параметр означает тип рассылаемого значения (здесь должен использоваться тип, определенный в стандарте MPI: в нашем случае это MPI_INT; тип должен быть указан, поскольку, во-первых, он определяет размер соответствующего элемента данных, и, во-вторых, позволяет правильно пересылать данные указанного типа между компьютерами, использующими разные представления для элементарных типов данных),

четвертый параметр – номер рассылающего процесса (в нашем случае его номер равен 0),

пятый параметр – имя коммуникатора группы (поскольку в рассматриваемом случае используется лишь исходная группа параллельных процессов, то здесь должно стоять имя исходного коммуникатора, а именно MPI_COMM_WORLD,

В 24-й строке находится число h. Заметим, что поскольку программа копируется во все параллельные процессы, то каждая переменная присутствует во многих экземплярах под одним именем: каждый процесс имеет эту переменную как свою собственность. Таким образом, значение h вычисляется независимо в каждом процессе.

Дальше (см. строки 25, 26) каждый процесс вычисляет часть квадратурной суммы, выбирая из суммы, фигурирующей в квадратурной формуле, слагаемые, номера которых сравнимы по модулю nprocs с его номером (напомним, что nprocs — общее число процессов). Благодаря такому распределению, количества слагаемых, поручаемых тому или иному процессу хорошо сбалансированы (они отличаются друг от друга не более, чем на одно слагаемое). Таким образом каждый процесс заполняет свою копию переменной sum определенной частью всей суммы, которую необходимо было вычислить.

В 27-й строке полученная процессом сумма умножается на 4h; такое умножение позволяет своевременно «нормализовать» результат вычислений в данном процессе с тем, чтобы по-возможности он находился в середине диапазона представимости чисел с плавающей точкой (после сложения всех полученных сумм нарушение этого правила возможно привело бы к ухудшению точности вычислений за счет сдвига к краю упомянутого диапазона). В каждом процессе результат получается в принадлежащей ему переменной term.

Строка 28 предназначена для сложения (полученных процессами значений переменных с именем term) частей интересующей нас суммы с помощью процедуры MPI_Reduce(&term, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD), где

первый параметр представляет собой адрес обрабатываемых переменных (в нашем случае адрес term),

второй параметр является адресом переменной, которой присваивается результат обработки (в данном случае, адресом переменной pi)

операция обработки представляет собой сложение, а `pi` именем переменной),

третий параметр – число передаваемых данных из каждого процесса (в рассматриваемом случае передается по одному данному – по слагаемому term),

четвертый параметр определяет тип данных, подвергающихся обработке (в данном случае MPI_DOUBLE),

пятый параметр определяет операцию обработки данных (в нашем случае это — операция сложения, идентифицируемая константой MPI_SUM),

шестой параметр указывает номер процесса, в котором сохраняется результат обработки (в данном случае указан процесс с номером 0),

седьмой параметр указывает имя коммуникатора (у нас это — MPI_COMM_WORLD),

Строки 29–30 содержат вывод полученного значения π. Строка 31 содержит вызов функции MPI_Finalize(), чем всегда завершается работа с библиотекой MPI, и строки 32, 33 содержат стандартное завершение функции main.

Задачи

1. В любом редакторе (см. введение) набрать программу, представленную в начале этого файла, и сохранить ее в файле с именем comp_pi.C . Скомпилировать и запустить полученную программу (см. введение). Ввести параметр n и прочитать результат.

2. Исследовать влияние значения n и числа процессов (см. введение) на время работы программы.

3. Написать параллельную программу, вычисляющую число простых чисел в интервале от 1 до n, с использованием стандарта MPI. Исследовать поведение времени выполнения этой программы в зависимости от значения n и числа процессов.