Лабораторная работа №5.

Параллельное и распределенное программирование. Технология MPI.

Название лабораторной работы: РАСПАРАЛЛЕЛИВАНИЕ КВАДРАТУРНОЙ ФОРМУЛЫ. ОПРЕДЕЛЕНИЕ ЧИСЛА π.

Содержание лабораторной работы

1. Обзор способов вычисления числа π. Алгоритмы для квадратурных формул

Число π представляет собой отношение длины окружности к диаметру, однако, вычисление его с использованием этого определения с надлежащей точностью затруднительно.

Число π встречается во многих естественных науках, ибо оно отражает существенные закономерности природы. В некоторых формулах неевклидовой геометрии также участвует число π, но уже не как отношение длины окружности к диаметру (там это отношение не является постоянным).

Число π иррационально и выражается бесконечной непериодической десятичной дробью:

π = 3.141592653589793... (0)

К этому числу, в частности, приводит отыскание пределов некоторых арифметических последовательностей, составляемых по простым законам. Ряд Лейбница сходится (весьма медленно) к числу π/4:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$$

Возможность чисто аналитического определения этого числа имеет принципиальное значение. Существует много способов вычисления числа π. Один из эффективных способов его вычисления — использование определенного интеграла для представления арктангенса с аргументом, равным единице (как известно, arctg 1 = π /4). Итак, имеем

$$\int \frac{dx}{1+x^2} = \text{arctg } x + C,$$

так что

$$\int_a^b \frac{dx}{1+x^2} = \text{arctg } b - \text{arctg } a;$$

принимая во внимание, что, arctg 0 = 0, отсюда получаем

$$\int_0^1 \frac{dx}{1+x^2} = \text{arctg } 1 = \pi/4. \tag{1}$$

Формула (1) позволяет приближенное вычисление числа π /4 заменить приближенным вычислением интеграла $\int_0^1 \frac{dx}{1+x^2}$. Поскольку для любой непрерывно дифференцируемой функции f(x) интеграла $\int_a^b f(x)dx$ можно вычислить приближенно по квадратурной формуле средних прямоугольников:

$$\int_a^b f(x)dx \approx h\sum_{i-1}^{n} f(a + (i - 1/2)h), \tag{2}$$

где

$$h \stackrel{\text{def}}{=} (b-a)/n, \tag{3}$$

то для приближенного вычисления интеграла (1) получаем

$$\int_0^1 \frac{dx}{1+x^2} \approx h\sum_{i=1}^{n} \frac{1}{1+((i-1/2)h)^2}, \tag{4}$$

Кроме приведенной формулы средних прямоугольников можно было бы использовать другие квадратурные формулы: формулу трапеций, формулу Симпсона и т. п. Все квадратурные формулы представляют собой линейную комбинацию значений функции (а иногда — и некоторых ее производных), вычисленных в точках, называемых узлами квадратурной формулы. Коэффициентами таких линейных комбинаций являются некоторые априори заданные фиксированные числа, называемые коэффициентами квадратурной формулы. Напомним квадратурные формулы трапеций и Симпсона (в следующих ниже формулах *h* вычисляется по формуле (3)).

Формула трапеций:

$$\int_a^b f(x)dx \approx h\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(a+ih)\right]. \quad (5)$$

Формула Симпсона:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[\frac{f(a)+f(b)}{2} + 2\sum_{i=1}^{n} f(a+(i-1/2)h) + \sum_{i=1}^{n-1} f(a+ih)\right]. \quad (6)$$

Здесь ограничимся распараллеливанием вычислений для формулы средних прямоугольников, хотя, ввиду только что сказанного, аналогичным образом распараллеливаются и другие квадратурные формулы.

2. Способы распараллеливания квадратурных формул

В предположениях концепции неограниченного параллелизма (т.е. в условиях, когда количество вычислительных модулей параллельной системы и размеры ее памяти столь велики, что не являются ограничением в задаче) естественно было бы на первых четырех тактах параллельной системы вычислить все числа $\frac{h}{1+((i-1/2)h)^2}$, *i*=1,2,...,n, а затем, используя схему сдваивания, вычислить их сумму. В результате потребовалось бы $\lceil \log_2 n \rceil + 4$ тактов параллельной системы (учтен еще такт, требующийся для вычисления величины *h* по формуле (3)).

Однако, для достаточно точного вычисления интеграла (1) требуется использовать весьма большие значения n в формулах (2) — (3), и поэтому рассчитывать на то, что количество вычислительных модулей ВС окажется не менее числа π, не приходится. Кроме того, применяя автоматизированные средства

распараллеливания каковым является библиотека, отвечающая стандарту MPI, мы стремимся передать работу по распределению процессов по вычислительным модулям самой вычислительной системе; это не только освобождает нас от рутинной работы, но благодаря стандарту MPI позволяет обеспечить переносимость программы на другую параллельную ВС.

3. Цель работы

Цель данной работы состоит в следующем:

1) дать представление о средствах распараллеливания MPI;

2) научить элементарным способам использования средств MPI при программировании,

3) оценить ускорение работы параллельной версии алгоритма в сравнении с последовательной при использовании многоядерных архитектур,

4) на основе лабораторной работы сделать выводы о зависимости ускорения от начальных параметров.

Постановка задачи: требуется вычислить приближения к числу π, используя различные методы вычисления интеграла (1) (квадратурные формулы средних прямоугольников, трапеций, Симпсона и различные значения параметра n), дать соответствующие варианты программы, численно исследовать ее устойчивость в зависимости от параметра n, искусственно вводя погрешность ϵ в промежуточные вычисления, теоретически обосновать полученные результаты.

4. Описание реализации

На параллельных системах стандарта MPI обычно реализован для алгоритмических языков C, C++ и Fortran, Следующая программа на языке C дает иллюстрацию такого использования для вычисления числа π по формуле (4).

```
1 #include <stdio.h>
2 #include <mpi.h>
3
4 double f(double x)
5 {
6 return 1/(1+x*x);
7 }
```

```
8
9 int main(int argc, char *argv[])
10 {
11 double pi, sum = 0, term, h;
12 int myrank, nprocs, n, i;
13 MPI_Init(&argc, &argv);
14 MPI_Comm_rank(MPI_COMM_WORLD, &myrank) ;
15 MPI_Comm_size(MPI_COMM_WORLD, &nprocs) ;
16 if (myrank == 0)
17   {
18      printf("Number of iterations=") ;
19      scanf("%d", &n);
20      }
21      MPI_Bcast (&n,1,MPI_INT,0,MPI_COMM_WORLD) ;
22      h = 1.0/n;
23      for(i = myrank+1; i <= n; i += nprocs)
24              sum += f(h*(i-0.5));
25      term = 4*h*sum;
26      MPI_Reduce(&term, &pi, 1, MPI_DOUBLE, MPI_SUM,0,MPI_COMM WORLD) ;
27      if(myrank == 0)
28              printf ("Computed value of pi=%lg\n",pi);
29      MPI_Finalize();
30      return 0;
31}
```

Заметим прежде всего, что согласно концепции SIMD (Single Instruction Multiple Data) при использовании стандарта MPI программа копируется во все рассматриваемые параллельные процессы; при этом, конечно, каждый процесс — в зависимости: от своего номера — выполняет специфическую для него работу

Первая строка программы подключает заголовочный файл стандартной библиотеки ввода-вывода языка C; во второй строке подключается заголовочный файл библиотеки mpi.h, реализующей стандарт MPI на языке C, строки с четвертой по седьмую содержат определение интегрируемой функции, а с 9 по 31 — определение функции main, которая и выполняет всю необходимую работу. Дальнейший разбор программы касается функции main.

В 11-й строке описываются вещественные переменные, в 12-й целые. Строки 13, 14, 15 используются для вызова процедур MPI. Рассмотрим вызываемые здесь процедуры подробнее.

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

Процедура MPI_Comm_rank(MPI_COMM_WORLD, &myrank) определяет номер процесса (он присваивается переменной myrank) в группе с коммуникатором MPI_COMM_WORLD. Этот коммуникатор определяет группу всех процессов, запущенных для решения данной задачи. Процессы в любой группе нумеруются целыми числами от 0 до (число процессов)-1.

Процедура MPI_Comm_size(MPI_COMM_WORLD, &nprocs) позволяет определить число процессов в группе с коммуникатором MPI_COMM_WORLD (это число присваивается выходном параметру nprocs). Оно определяется при запуске программы на выполнение при помощи опции —np и не должно превышать имеющегося в системе числа процессоров (в случае кластера 2444 — 9 компьютеров по 2 процессора, т. е. всего не более 18 процессов).

В строках 16-20 инициируется ввод числа n слагаемых в сумме (4) 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 21 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 T; тип должен быть указан, поскольку, во-первых, он определяет размер соответствующего элемента данных, и, во-вторых, позволяет правильно пересылать

данные указанного лица 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 22-й строке находится число h по формуле (3). 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 23, 24) каждый процесс вычисляет часть квадратурной суммы, выбирая из суммы, фигурирующей в квадратурной формуле, слагаемые, номера которых сравнимы по модулю nprocs 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 25-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 26 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), третий параметр — число передаваемых данных из каждого процесса (в рассматриваемом

случае передается по одному данному — по слагаемому теги), 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 27-28 содержат вывод полученного значения π. Строка 29 содержит вызов функции MPI_Finalize(), чем всегда завершается работа с библиотекой MPI, и строки 30. 31 содержат стандартное завершение функции main.

5. Описание лабораторной работы

В лабораторной работе требуется выполнить следующие разделы:

1) Запуск программы.
2) Распараллеливание квадратурной формулы средних прямоугольников.
3) Распараллеливание квадратурной формулы трапеций.
4) Распараллеливание квадратурной формулы Симпсона.
5) Изменение конфигурации (масштабирование) параллельной системы.
6) Построение графиков полученных зависимостей.

1. Запуск программы:
   - в любом редакторе набрать программу, представленную в пункте 4, и сохранить ее в файле с именем comp_pi.c,
   - скомпилировать и запустить полученную программу.
   - ввести параметр n и прочитать результат.
2. Распараллеливание квадратурной формулы средних прямоугольников

Здесь предлагается исследовать точность вычислений и время счета в зависимости от параметра n. Для измерения времени используется способ, описанный во введении. Предупреждение! При исследовании могут получиться весьма неожиданные результаты. После выполнения задания попытайтесь их объяснить.

Порядок действий

1. Измените программу так, чтобы печаталось уклонение результата вычислений от значения числа π, указанного в формуле (0)

2. Для значений n = 2^i, i = 9, 10, 11, 12,….. запустите программу и определите время и точность вычислений; заполните таблицу

Таблица - Квадратурная формула средних прямоугольников

Значение n	Время вычислений, сек.	Точность вычислений
2^9		
….		

3. Распараллеливание квадратурной формулы трапеций

Порядок действий:

1. Измените программу так, чтобы вместо квадратурной формулы средних прямоугольников распараллеливалась квадратурная формула трапеций (см. формулу (5)).
2. Аналогично первому заданию запустите полученную программу для значений и заполните таблицу.

Таблица - Распараллеливание квадратурной формулы трапеций

Значение n	Время вычислений, сек.	Точность вычислений
2^9		
….		

4. Распараллеливание квадратурной формулы Симпсона.

Порядок действий:

- Аналогично предыдущему заданию измените программу так, чтобы вместо квадратурной формулы трапеций распараллеливанию подверглась квадратурная формула Симпсона (см. формулу (6)).
- Аналогично первому заданию запустите полученную программу для значений n,
- заполните таблицу, определите время и точность вычислений.

Таблица - Распараллеливание квадратурной формулы Сипсона

Значение n	Время вычислений, сек.	Точность вычислений
2^9		
….		

5. Изменение конфигурации (масштабирование) параллельной системы

Изменить конфигурацию параллельной системы можно изменением количества вычислительных модулей.

Порядок действий

- Провести следующие варианты экспериментального исследования:

1) сохранить 2 параллельных вычислительных модуля

2) сохранить 4 параллельных вычислительных модуля.

3) сохранить 8 параллельных вычислительных модулей

- В каждом из рассмотренных вариантов конфигурации параллельной системы провести вычисления, указанные в пунктах 1-4.
- По полученным результатам для каждой задачи найти относительное ускорение вычислений как отношение времени T_1, истраченного при запуске на конфигурации 1, ко времени T_i, истраченного при запуске на i-й конфигурации:

$$U_i = \frac{T_1}{T_i} \qquad i = 2, 3.$$

- Результаты оформить в виде таблиц вида (приводится лишь один пример таб-

Таблица - Распараллеливание квадратурной формулы ……

Значение n	Время вычислений, сек.	Точность вычислений	Относительное ускорение, U_i
2^9			
….			

6. Построение графиков полученных зависимостей

Для исследованных случаев построить полигональные графики следующих зависимостей

1) зависимости ускорения от параметра квадратурной формулы,

2) зависимости ускорения от числа процессов,

3) зависимости точности вычислений и времени счета от параметра.

ЛИТЕРАТУРА

1. Мысовских И.П. Лекции по методам вычислений. СПб, 1998. 47

2. Керниган Б., Ригчи Д. Язык программирования С, М. 1992

3. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. СПб, 2002. 400 с

1. Восводин В.В.. Воеводин Вл.В. Параллельные вычисления. СПб, 2002. 608 с.