ВАРИАНТ № 1

Моделирование средствами системы MPI двунаправленного обмена данными по кольцевому протоколу

Описание протокола

Кольцевой протокол — это протокол обмена данными между станциями, соединёнными в кольцевую топологию (см. рис.). Рассматриваемый протокол является не более, чем модельным. Основа обмена данными — фрейм, который передаётся от одной станции к другой по кольцу в некотором заранее выбранном направлении (по часовой стрелке (и против нее)).

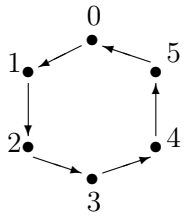

Рис.

Станция с номером 0 имеет специальное назначение — она запускает фрейм в кольцо, т.е. инициирует работу протокола.

Фрейм имеет поля: | CLN | SRC | DST | DATA |, где

- SRC — адрес станции-отправителя фрейма;
- DST — адрес станции-получателя фрейма;
- DATA — передаваемые данные;
- CLN — бит заполненности фрейма.

Постановка задачи

Реализовать средствами системы MPI моделирование обмена данными по кольцевому протоколу между N процессами (станциями), причём считать, что имеется двунаправленное кольцо, в котором передаются два фрейма: один передаётся по часовой стрелке, а другой — против неё.

Для коммуникаций использовать функции неблокирующих передачи и приёма сообщений MPI_Irecv()/MPI_Isend() соответственно, а для ожидания сообщений — функцию MPI_Waitany().

На стандартный вывод должны выводиться сообщения обо всех действиях процесса — о получении фрейма (с указанием направления, содержимого управляющих полей фрейма), об отправке и приёме сообщений и подтверждений о приёме.

Завершение программы должно происходить по прошествию некоторого фиксированного времени, либо после того, как фреймы совершат некоторое фиксированное число обходов кольца.

Алгоритм действий процесса (станции)

1. *Если* номер процесса (станции) равен 0, *то*
   запустить фрейм в кольцо.

2. Ожидать фрейм от любого процесса-соседа.

3. Получить фрейм от процесса-соседа.

   - В случае, когда поле CLN полученного фрейма равно 0, выполнить следующие действия.
     Если есть необходимость (моделируется с помощью датчика случайных чисел) послать сообщение, *то*
     заполнить поле SRC своим номером, поле DST — случайно выбранным номером процесса так, чтобы DST $\neq$ SRC, и поле DATA — тестовыми данными.
   - В случае, когда поле CLN полученного фрейма равно 1, выполнить следующие действия.
     Если поле DST совпало с номером процесса, *то*
     скопировать себе данные из поля DATA.
     Если значение SRC совпало с номером процесса, *то*
     очистить поля SRC и DST и установить CLN равным 0.

4. Отправить фрейм дальше по кольцу. Перейти к шагу 2.

ВАРИАНТ № 2

Реализация средствами системы MPI алгоритма решения уравнения теплопроводности

Описание разностной схемы

Простейшее уравнение теплопроводности в области $[0,T]\times[0,L]=\Omega$ имеет вид:

$$\begin{cases} \dfrac{\partial y}{\partial t}=\dfrac{\partial^2 y}{\partial x^2}+f(t,x) & ((t,x)\in(0,T]\times(0,L)),\\ y(0,x)=\varphi(x) & (x\in[0,L])\\ y(t,0)=\psi_0(t) & (t\in[0,T]),\\ y(t,L)=\psi_1(t) & (t\in[0,T]). \end{cases} \tag{1}$$

Для приближенного решения уравнения (1) производные заменяются на свои разностные аналоги, а область Ω — на сетку (n,j) с шагом τ по переменной t и с шагом h по переменной x. Пусть далее $K=\dfrac{T}{\tau}$, $M=\dfrac{L}{h}$. Подмножество узлов сетки вида $\{(n,j)\,|\,0\le j\le M\}$ будем называть слоем (по времени) с номером n.

В результате получается следующая разностная схема:

$$\begin{cases} \dfrac{u(n+1,j)-u(n,j)}{\tau}= \\ \qquad =\dfrac{u(n,j-1)-2u(n,j)+u(n,j+1)}{h}+F(n,j),\\ u(0,j)=\Phi(j),\\ u(n,0)=\Psi_0(n),\\ u(n,M)=\Psi_1(n),\\ (n,j)\in\{1,\dots,K\}\times\{1,\dots M-1\}=W. \end{cases} \tag{2}$$

Здесь F, Φ, Ψ_0, Ψ_1 — сеточные аналоги функций f, φ, ψ_0, ψ_1 соответственно, т.е., например, $F(n,j)=f(n\tau,jh)$.

Теперь для значения $u(n+1,j)$ можно получить явное выражение:

$$u(n+1,j)=u(n,j)+\tau\Big[\frac{u(n,j+1)-2u(n,j)+u(n,j-1)}{h^2}+F(n,j)\Big]. \tag{3}$$

Из (3) видно, что $u(n+1,j)$ зависит непосредственно только от значений $u(n,j)$ и $u(n,j\pm1)$. Заметим также, что приведенная схема усточива при выполнении соотношения: $\frac{\tau}{h^2}<\frac{1}{2}$.

Постановка задачи

Рассмотрим уравнение теплопроводности:

$$\begin{cases} \dfrac{\partial y}{\partial t}=\dfrac{\partial^2 y}{\partial x^2}+\pi[\pi t^2\sin(\pi t x)+x\cos(\pi x t)] & ((t,x)\in(0,1]\times(0,1)),\\ y(0,x)=0 & (x\in[0,1])\\ y(t,0)=0 & (t\in[0,1]),\\ y(t,1)=\sin(\pi t) & (t\in[0,1]). \end{cases} \tag{4}$$

Решением (4) является функция $y(x,t) = sin(\pi x t)$.

Реализовать средствами системы MPI алгоритма приближенного решения уравнения теплопроводности (4) на N процессорах, объединённых в конвейер (см. рис.), и найти максимальную погрешность вычисления.

Заметим, что исходные данные доступны всем процессам с начала их работы.

Для распараллеливания следует разделить сетку по переменной x на отрезки из M/N узлов. На каждом из отрезков значение функции вычисляется отдельным процессом.

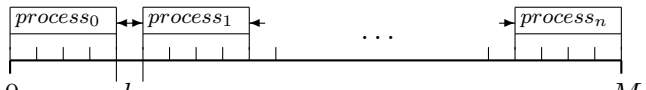

Рис.

Алгоритм действий процесса

1. Вычислить по своему номеру границы отрезка, на котором будут проводится вычисления. Установить номер текущего слоя n в 0.

2. По формуле (3) вычислить значения $u(n+1, j)$ на следующем слое по времени (j изменяется внутри своего отрезка сетки по x), обновить значение максимальной ошибки.

3. *Если* отрезок — не самый левый, *то*
послать соседу слева значение u вычисленное в самой левой точке отрезка.

4. *Если* отрезок — не самый правый, *то*
послать соседу справа значение u, вычисленное в самой правой точке отрезка.

5. *Если* следующий слой — не последний, *то*
увеличить номер текущего слоя и перейти к шагу 2,
иначе
перейти к шагу 6.

6. *Если* номер процесса равен нулю, *то*
получить от всех процессов значение ошибки и выбрать максимальное.

ВАРИАНТ № 3

Реализация средствами системы MPI алгоритма умножения матриц размерности $N \times N$

Постановка задачи

Реализовать средствами системы MPI алгоритм умножения матриц $A = a_{ij}$ и $B = b_{ij}$ ($0 \le i, j \le N-1$, $N \le 3$) за время $O(1)$ на N^3 процессорах. При реализации использовать топологию трёхмерного куба процессов, т.е. процессы должны размещаться в точках с целыми координатами внутри куба $N \times N \times N$. Считается что значения обеих матриц доступны уже в начале работы процессов.

Алгоритм действий процесса

1. Организовать топологию трёхмерного куба и вычислить свои координаты в этой топологии (использовать функции MPI_Cart_create() и MPI_Cart_get()).

2. Вычислить произведение $a_{ik}b_{kj}$.

3. *Если* координаты процесса равны $(i, j, 0)$, *то* получить от всех процессов с координатами (i, j, k), $k \in \{0, \ldots, N-1\}$, вычисленные произведения, просуммировать их и вывести результат. (Для передачи результатов и их одновременного суммирования следует использовать функцию MPI_Reduce(), для выделения подтопологии "колонны" над $(i, j, 0)$ — функцию MPI_Cart_sub()).

ВАРИАНТ № 4

Реализация средствами системы MPI алгоритма параллельной сортировки массива с использованием топологии бинарного дерева

Постановка задачи

Отсортировать массив средствами системы MPI с использованием топологии бинарного дерева процессов (т.е. процессы должны размещаться в вершинах дерева).

Считать, что длина массива равна $k \cdot 2^{n-1}$, где n — глубина дерева, и что копии массива доступны каждому процессу в начале его работы.

Описание алгоритма действий процесса

1. *Если* процесс — вершина-лист, *то*
   отсортировать отрезок массива длины k, начало и конец которого определяются из номера процесса в списке листьев,
   иначе
   получить от процессов-преемников отсортированные части массива и отсортировать их методом слияния.

2. *Если* процесс — не вершина-корень, *то*
   послать полученный на предыдущем шаге массив процессу-предшественнику,
   иначе
   распечатать отсортированный массив.

ВАРИАНТ № 5

Реализация средствами системы MPI алгоритма параллельной сортировки массива с использованием топологии кольцевого конвейера

Постановка задачи

Отсортировать массив средствами системы MPI из $k \cdot n$ элементов на n процессорах (которые нумеруются, начиная с 0 до $n-1$) с использованием топологии кольцевого конвейера (т.е. соединить процессы в кольцо).

Для распараллеливания следует разделить массив на блоки из k последовательных элементов, каждый из которых сортируется отдельным процессом.

Считать, что копии массива доступны каждому процессу в начале его работы.

Алгоритм действий процесса

1. Определить и отсортировать блок массива, соответствующий данному процессу.

2. *Если* номер процесса больше 0, *то*
   получить от предыдущего по номеру процесса часть массива (которая уже отсортирована) и отсортировать её совместно со своей методом слияния.

3. *Если* номер процесса меньше $n-1$, *то*
   послать следующему по номеру процессу отсортированную часть массива,
   иначе
   послать отсортированный массив процессу 0.

4. *Если* номер процесса равен 0, *то*
   получить от процесса $n-1$ отсортированный массив и распечатать его.