Министерство образования и науки Российской Федерации

Московский физико – технический институт

(Государственный университет)

Кафедра радиоэлектроники и прикладной информатики

Численное решение уравнения переноса с использованием технологии MPI

Лабораторная работа № 1

по курсу

Параллельное программирование

Москва 2011

УДК 004.424

Лабораторная работа №1 по курсу : Параллельное программирование.

Составители:

Пальян Р.Л., Гаврилов Д.А., Леус А.В., Филимонов А.В.

Лабораторная работа составлена при поддержке компании Intel.

Московский физико-технический институт

Кафедра радиоэлектроники и прикладной информатики

141700, Моск. обл, г. Долгопрудный, Институтский пер. 9

Эффективность и ускорение параллельных программ

Параллельная программа – это множество взаимодействующих параллельных процессов. Основной целью параллельных вычислений является ускорение решения вычислительных задач. Параллельные программы обладают следующими особенностями:

1. осуществляется управление работой множества процессов;
2. организуется обмен данными между процессами;
3. утрачивается детерминизм поведения в следствие асинхронности доступа к данным;
4. преобладают нелокальные и динамические ошибки;
5. появляется возможность тупиковых ситуаций;
6. возникают проблемы масштабируемости программы и балансировки загрузки вычислительных узлов.

Рассмотрим некоторый последовательный алгоритм решения какой-либо задачи. В нем есть как операции, которые не могут выполнятся параллельно (например, ввод/вывод), так и операции, которые можно выполнять на нескольких процессорах одновременно. Пусть доля последовательных операций в алгоритме равна α.

Время выполнения последовательного алгоритма обозначим T_1. Время выполнения параллельной версии алгоритма на p одинаковых процессорах можно записать следующим образом:

$$T_p = \alpha T_1 + \frac{(1-\alpha)T_1}{p}$$

Ускорением параллельного алгоритма называют отношение времени выполнения лучшего последовательного алгоритмам к времени выполнения параллельного алгоритма:

$$S = \frac{T_1}{T_p}.$$

Параллельный алгоритм может давать большое ускорение, но использовать для этого множество процессов неэффективно. Для оценки масштабируемости параллельного алгоритма используется понятие эффективности:

$$E = \frac{S}{p}.$$

Теоретическую оценку максимального ускорения, достижимого для параллельного алгоритма с долей последовательных операций равной α определяется законом Амдала:

$$S = \frac{T_1}{T_p} = \frac{T_1}{\alpha T_1 + \frac{(1-\alpha)T_1}{p}} \le \frac{1}{\alpha}.$$

Таким образом, если всего 10% операций алгоритма не может быть выполнена параллельно, то никакая параллельная реализация данного алгоритма не может дать

больше ускорение более чем в 10 раз.

Основные понятия технологии MPI

В рамках данной работы изучается параллельное программирование для систем с распределенной памятью. В системах этого типа на каждом вычислительном узле работают процессы, реализующие некоторый параллельный алгоритм. У каждого процесса есть своя собственная область памяти, к которой ни один другой процесс не имеет доступа. Все взаимодействия осуществляются с помощью передачи сообщений между процессами.

Существует множество способов организации передачи сообщений. В данной работе рассматривается разработка параллельных программ с помощью технологии MPI. MPI – программный интерфейс для передачи сообщений между процессами. Стандартизацией MPI занимается организация MPI Forum. Реализации интерфейса MPI существуют для множества различных платформ.

В рамках технологии MPI на каждом вычислительном узле запускается копия параллельной программы. Каждая копия получает ранг – уникальный идентификатор, использующийся для адресации сообщений.

Обмен сообщениями в рамках MPI происходит между процессами, которые относятся к одному коммуникатору. Коммуникатор – это способ группировки процессов. По умолчанию все запущенные процессы попадают в коммуникатор MPI_COMM_WORLD.

MPI предусматривает несколько вариантов обмена сообщениями. Сообщения бывают типа «точка-точка» – между двумя процессами, и «коллективные» – между несколькими процессами одновременно.

Также отправка и прием сообщений бывают блокирующим и неблокирующим (асинхронными). В случае блокирующего обмена передающие и принимающие процессы блокируются до тех пор, пока передача сообщения не завершится. Например, в случае блокирующего обмена типа «точка-точка» отправитель будет приостановлен до тех пор, пока получатель не вызовет функцию получения сообщения и получит его.

Структура программы и основные функции MPI

Структура программы, использующей MPI, выглядит следующим образом:

1) Подключение библиотеки MPI.
2) Инициализация среды MPI.
3) Работа программы, обмен сообщениями.
4) Остановка среды MPI.

Для подключения библиотеки MPI в программе на языке C нужно включить заголовочный файл mpi.h.

Для инициализации среды MPI на каждом вычислительном узле необходимо один и только один раз вызвать функцию MPI_Init(*argc, *argv). После успешной инициализации каждый процесс может узнать свой ранг с помощью MPI_Comm_rank(MPI_COMM_WORLD, *rank). Также можно узнать общее число

процессов в коммуникаторе, вызвав MPI_Comm_size(MPI_COMM_WORLD, *size).

Остановка среды MPI осуществляется вызовом функции MPI_Finalize().

Для обмена сообщениями типа «точка-точка» используется следующий набор функций:

- MPI_Send(buffer, count, type, dst, tag, comm) – блокирующая отправка.
- MPI_Isend(buffer, count, type, dst, tag, comm, request) – неблокирующая отправка.
- MPI_Recv(buffer, count, type, src, tag, comm, status) – блокирующий прием.
- MPI_Irecv(buffer, count, type, src, tag, comm, request) – блокирующий прием.

Параметры всех этих функции очень похожи:

- buffer – указатель на начало области памяти, откуда будут передаваться (или куда будут приниматься данные).
- count – число элементов в буфере.
- type – тип элемента в буфере. В MPI поддерживаются все основные типы языка C, а также можно создавать произвольные пользовательские типы элементов.
- dst/src – ранг принимающего/передающего процесса.
- tag – метка сообщения. Служит для выделения логического типа сообщений.
- comm – коммуникатор, в рамках которого будет вестись обмен.
- status – указатель на структуру, в которой будет информация о статусе доставки сообщения.
- request – указатель на структуру, в которой будет информация о статусе доставки сообщения.

Для компиляции и запуска программы, использующей MPI, на компьютере должна быть установлена и настроена реализация библиотеки MPI (например, OpenMPI или MPICH2).

Компиляция программы осуществляется с помощью команды:

mpicc -o <название_исполняемого_файла> <имя_исходного_файла>.c

Запуск осуществляется с помощью команды:

mpirun -n <число_запускаемых_процессов> <название_исполняемого_файла> [аргументы]

Практическая часть

Основным заданием данной лабораторной работы является разработка и исследование параллельной программы, осуществляющей поиск численного решения для уравнения переноса:

$$\partial u(t,x)/\partial t + a \cdot \partial u(t,x)/\partial x = f(t,x),\ 0 \le t \le T,\ 0 \le x \le X$$

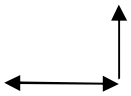
$$u(0,x) = \varphi(x),\ 0 \le x \le X$$

$$u(t,0) = \psi(t),\ 0 \le t \le T$$

Для решения задачи используется равномерная сетка с шагами τ по времени и h по координате. Функция *u(t,x)* рассматривается в точках $t=k\tau$, $x=mh$, $0\le k\le K$, $0\le m\le M$, $T=K\tau$, $X=Mh$.

Для поиска решения в зависимости от индивидуального задания нужно использовать одну из следующих разностных схем:

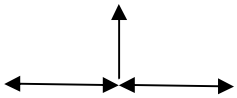
- Явный левый уголок. Шаблон схемы имеет вид:



Разностная схема записывается следующим образом:

$$\left(u_m^{k+1}-u_m^k\right)\tau+\left(u_m^k-u_{m-1}^k\right)h=f_m^k,k=0,\ldots,K-1,m=0,\ldots,M$$

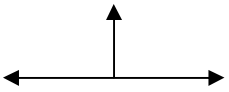
1. Явная четырехточечная схема. Шаблон схемы имеет вид:



Разностная схема записывается следующим образом:

$$\left(u_m^{k+1}-u_m^k\right)\tau+\left(u_{m+1}^k-u_{m-1}^k\right)2h-0.5\tau\left(u_{m+1}^k-2u_m^k+u_{m-1}^k\right)h^2=f_m^k,k=0,\ldots,K-1,m=0,\ldots,M-1$$

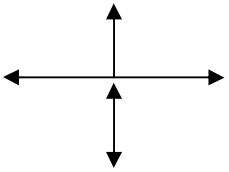
- Явная центральная трехточечная схема:



Разностная схема записывается следующим образом:

$$\left(u_m^{k+1}-0.5\left(u_{m+1}^k+u_{m-1}^k\right)\right)\tau+\left(u_{m+1}^k-u_{m-1}^k\right)2h=f_m^k,k=0,\ldots,K-1,m=0,\ldots,M-1$$

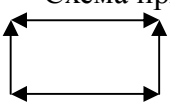
- Схема крест:



Разностная схема записывается следующим образом:

$$\left(u_m^{k+1}-u_m^{k-1}\right)2\tau+\left(u_{m+1}^k-u_{m-1}^k\right)2h=f_m^k,k=0,\ldots,K-1,m=0,\ldots,M-1$$

- Схема прямоугольник:



Разностная схема записывается следующим образом:

$$\left(u_{m-1}^{k+1}-u_{m-1}^k+u_m^{k+1}-u_m^k\right)2\tau+\left(u_m^{k+1}-u_{m-1}^{k+1}+u_m^k-u_{m-1}^k\right)2h=f_{m+1/2}^{k+1/2},k=0,\ldots,K-1,m=0,\ldots,M$$

Задание к допуску:

Написать программу, которая измеряет задержку передачи сообщения между двумя узлами сети с помощью технологии MPI.

Задание к выполнению:

1. Разработать последовательную программу, которая численно решает уравнение переноса по разностной схеме в соответствие с индивидуальным вариантом.
2. Разработать с помощью технологии MPI параллельную программу, которая численно решает уравнение переноса по разностной схеме в соответствие с индивидуальным вариантом.

Задание к сдаче:

Оценить теоретически и измерить ускорение и эффективность полученной реализации в зависимости от числа задействованных процессоров и размера задачи (шага сетки по времени).

Контрольные вопросы:

1. Ускорение и эффективность параллельных алгоритмов.
2. Закон Амдаля.
3. Свойства канала передачи данных. Латентность.
4. Виды обменов «точка-точка»: синхронные, асинхронные. Буферизация данных.
5. Балансировка загрузки: статическая и динамическая.
6. Геометрический параллелизм.
7. Конвейерный параллелизм.

Приложение:

Официальная страница документации MPI Forum:

http://www.mpi-forum.org/docs/docs.html