Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра «Математическое моделирование и программное обеспечение»

ТЕХНОЛОГИИ ВЫСОКОСКОРОСТНЫХ ВЫЧИСЛЕНИЙ

Методические указания по выполнению лабораторных работ для магистрантов специальности 6М070400-Вычислительная техника и программное обеспечение

Алматы 2017

СОСТАВИТЕЛЬ: З.К.Куралбаев Методические указания по выполнению лабораторных работ для магистрантов специальности 6М070400 - Вычислительная техника и программное обеспечение. – Алматы: АУЭС, 2017. – 32 с.

В данной методической разработке приведены теоретические и практические материалы, предназначенные для организации лабораторных занятий по дисциплине «Технологии высокоскоростных вычислений».

Здесь предложены семь лабораторных работ, темы которых охватывают различные разделы данной дисциплины. Рассматриваемые здесь вычислительные задачи обычно возникают при решении различных сложных задач, требующих больших объемов вычислений. При выполнении заданий по этим лабораторным работам магистрант вначале знакомится с популярными способами построения и анализа параллельных вычислений и получают навыки решения вычислительных задач, используемых в научных исследованиях.

Знания и навыки, полученные в результате изучения данной дисциплины, могут быть использованы в дальнейшем магистрантом при выполнении вычислительных работ для научно-исследовательской и экспериментальной работы.

Ил. 3, библиогр. – 15 назв.

Рецензент: Кожамбердиев К.О. к.т.н., доцент кафедры ЭиР

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2016 г.

Введение

Целью включения в учебный план магистерской подготовки дисциплину «Технологии высокоскоростных вычислений» является обеспечение магистрантов, будущих квалифицированных IKT-специалистов, знаниями по решению сложных вычислительных задач, получения навыков разработки параллельных алгоритмов. Изучение дисциплины базируется на знаниях, полученных в результате прохождения подготовки по таким дисциплинам как «Алгоритмизации и основы программирования», «Технологии программирования», «Операционные системы», «Объектно–ориентированное программирование», «Математическое и компьютерное моделирование».

Поиск способов и методов решения вычислительных задач, требующих выполнения большого объема вычислительной работы за малый отрезок времени, потребовал применение параллельного вычисления для сокращения времени выполнения операций. Многие практические задачи связаны с решением задач линейной алгебры таких, как решение системы алгебраических уравнений с многотысячными неизвестными или с матричными вычислениями огромных размеров. Такие задачи возникают при решении краевых задач уравнений математической физики или систем других дифференциальных уравнений в результате их дискретизации. Круг таких задач очень широк к ним приводят многие жизненно важные проблемы.

«Если вначале без точных масштабных расчетов было невозможно проектировать, создавать и эксплуатировать ядерное оружие, самолеты, ракеты и другие образцы военной техники, то сейчас в этом «вычислительно-модельном» фундаменте нуждаются, без преувеличения, все сферы жизнедеятельности современного общества. Томография и компьютерное проектирование лекарств, глобальные компьютерные сети и обеспечение финансовой системы, информационно-телекоммуникационный комплекс, ставшей одной из крупнейших отраслей мировой экономики, многое-многое другое» [1].

Решение таких сложных вычислительных задач, возникающих во многих отраслях науки и в практических исследованиях, невозможно без активного использования компьютерной техники, математических моделей, эффективных вычислительных алгоритмов, больших и надежных программных комплексов. Это потребует применения новых методов организации вычислительного процесса, использующих современные технологии высокоскоростных вычислений, кластерной системы и суперкомпьютеров [2, 3].

Разработка алгоритмов и программ для организации таких процессов имеет некоторые существенные отличия от традиционных технологий программирования, что требует определенную подготовку специалистов в

этом направлении. В связи с такой необходимостью, в учебные планы магистерской подготовки включена дисциплина «Технологии высокоскоростных вычислений», которая предназначена дать будущим IKT-специалистам начальную подготовку по освоению и получению навыков по созданию алгоритмов и программ для решения сложных вычислительных задач.

Для закрепления знаний, полученных в теоретическом обучении, и получения навыков практической реализации их, предусмотрено выполнение магистрантами индивидуальных заданий на лабораторных занятиях. В этих работах предлагаются задания, в которых должны быть разработаны алгоритмы решения задач и программы на одном из языков программирования. На первый взгляд предлагаемые задачи могут казаться очень простыми, для решения которых могут быть использованы известные алгоритмы. Очевидно, это так, если рассматриваются задачи для небольших объемов вычислений. Для больших задач использование традиционных методов программирования потребует затраты огромного времени, что во многих случаях непозволительно.

Каждая лабораторная работа посвящена определенной теме вычислительной математики. В каждой из них определена цель работы, дано задание, приведены методические материалы и ссылки к литературным источникам. Перед выполнением лабораторной работы обучающийся обязан ознакомиться с теоретическими и методическими материалами, отвечать на предлагаемые там контрольные вопросы. При выполнении заданий магистрант должен проявить самостоятельность, использовать знания и навыки, полученные во время изучения других дисциплин специальности, и показать умения разработать алгоритмы и программы параллельных вычислений.

Результаты решения задач в каждом задании должны быть использованы для проведения сравнительного анализа с целью выяснения эффективности каждого из видов алгоритмов. Такой анализ должен выявить преимущества параллельных вычислений перед традиционными методами программирования. По полученным результатам должен быть подготовлен отчет по каждой лабораторной работе согласно утвержденному графику.

1 Лабораторная работа № 1. Алгоритмы параллельного вычисления суммы числовой последовательности

Цель работы: изучение возможных параллельных методов вычисления суммы числовой последовательности, которая может быть рассмотрена как сумма элементов одномерного вектора; знакомство с алгоритмами различных схем суммирования.

Задание и порядок выполнения работы:

1) Изучить способы построения и анализа параллельных методов вычислений суммы числовой последовательности (элементов одномерного массива или вектора).

2) Разработать последовательный алгоритм суммирования.

3) Разработать алгоритм каскадной схемы суммирования.

4) Разработать алгоритм модифицированной каскадной схемы.

5) Составить программы на основе разработанных алгоритмов и выполнить расчеты.

6) Провести анализ возможных способов последовательной и параллельной организации вычислений.

7) Оформить отчет по данной лабораторной работе.

Постановка задачи о вычислении суммы числовой последовательности и методические указания по выполнению работы.

Здесь рассматривается одна из первых задач, для решения которой может быть использован параллельный подход для увеличения скорости вычисления, тем самым уменьшения времени выполнения. Рассматриваемая здесь задача очень простая и алгоритм ее решения легок для понимания. Целью рассмотрения такой задачи является - показать на простом примере возможность использования параллельных вычислений. Для этого здесь, наряду с последовательной схемой суммирования, приведены каскадная и модифицированная каскадная схемы.

Постановка задачи. Пусть дана некоторая числовая последовательность

$$x_1, x_2, x_3, \ldots, x_n. \tag{1}$$

Требуется вычислить сумму данной последовательности:

$$S_k = \sum_{i=1}^{k} x_i = x_1 + x_2 + x_3 + \ldots + x_k, \ k = 1, 2, 3, \ldots, n. \tag{2}$$

Для решения данной задачи предлагается использовать три вида алгоритма и проводить анализ эффективности их.

1. *Последовательный алгоритм*. Обычно для вычисления таких сумм используется известный традиционный метод, который состоит в последовательном суммировании элементов одномерного массива следующих чисел: $\{x_i, i = 1, 2, 3, \ldots, n\}$. Очевидно, этот алгоритм очень простой и не представляет никакой трудности для разработки программы.

Пусть рассматривается алгоритм определения суммы заданных чисел:

$$P = \sum_{i=1}^{k} x_i = x_1 + x_2 + x_3 + \ldots + x_k. \tag{3}$$

Последовательная вычислительная схема суммирования может быть изображена в следующем виде [3]:

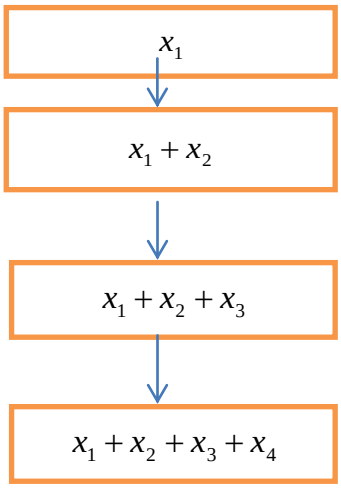

Рисунок 1 - Последовательная вычислительная схема алгоритма суммирования

Алгоритм записывается в виде последовательности следующих действий:

1) Ввод заданной последовательности чисел в память компьютера.

2) Начало цикла $i = 1;\ P = x_i;$ присваивание счетчику цикла i начального его значения и сумме P присваивается первое число x_i.

3) Переход к следующему шагу: $i = i + 1;\ P = P + x_i;$ сумме P прибавляется следующее новое число x_i.

4) Если $i \le k - 1$, то осуществляется переход к пункту С); в противном случае процесс вычисления завершается.

Здесь легко можно догадаться, что простой цикл последовательно повторяется $k - 1$ раз.

2. *Каскадная схема суммирования*. Для ускорения вычислительного процесса следует использовать возможность параллельного выполнения некоторых вычислительных операций. Параллельный процесс в алгоритме суммирования становится возможным только при способе построения процесса вычислений, основанном на использовании ассоциативности операции сложения. При этом необходимо определить информационную независимость операций, подлежащих параллельному выполнению.

Предлагаемый новый вариант суммирования, известный в литературе как *каскадная схема* [3] , состоит в следующем (рисунок 2):

- на первой итерации каскадной схемы все исходные данные разбиваются на пары, и для каждой пары вычисляется сумма значений; для наглядности здесь ограничивались рассмотрением четырех элементов числовой последовательности;

- далее все полученные суммы пар также разбиваются на пары, и снова выполняется суммирование значений пар и т.д., в результате будет получена искомая сумма чисел заданной последовательности.

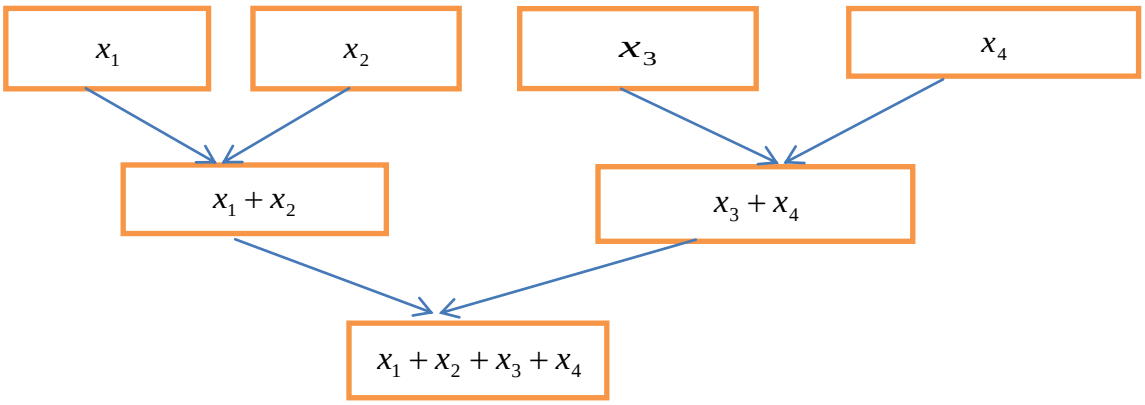

Рисунок 2 - Каскадная схема алгоритма суммирования

Предложенная каскадная схема алгоритма суммирования может быть не единственной; могут существовать различные ее модификации. Одна из них рассмотрена ниже.

3. *Модифицированная каскадная схема*. В данном варианте каскадной схемы все вычисления состоят из двух последовательных этапов суммирования:

- на первом этапе все суммируемые элементы подразделяются на группы, количество которых равно $\frac{n}{\log_2 n}$; в каждой из них содержатся $\log_2 n$ элементов; для каждой из этих групп вычисляется сумма элементов при помощи последовательного алгоритма суммирования; вычисления в каждой группе выполняются независимо друг от друга, что позволяет организовать параллельное выполнение;
- на втором этапе вычисление полученных сумм выполняется методом обычной каскадной схемы.

Схематическое изображение модифицированной каскадной схемы можно представить в виде рисунка 3:

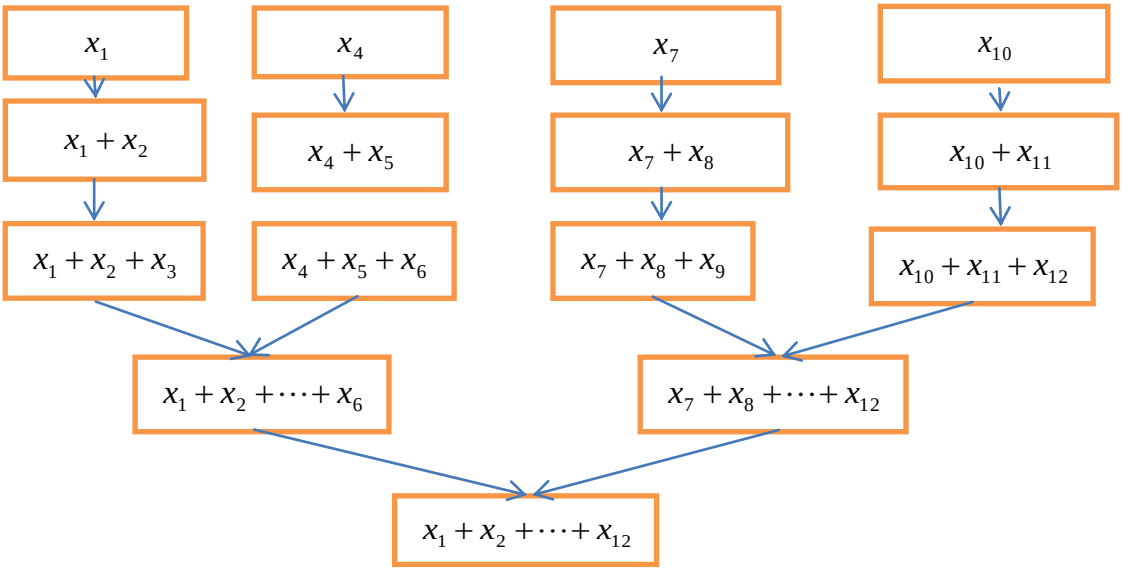

Рисунок 3 – Модифицированная каскадная схема суммирования

Здесь даются некоторые методические указания по выполнению данного задания одним из способов параллельного вычисления. Для этого необходимо выполнение следующих этапов для разработки параллельной программы:

4) Вначале должна быть создана функция ядра, которая будет суммировать элементы вектора. Суммирование производится так, чтобы потоки (нити) не работали с одной и той же памятью. Здесь должна быть использована разделяемая память, которая является памятью общего доступа для потоков внутри блока.

Здесь используется редукция, которая является процедурой, в результате которой из заданного (входного) массива получается массив меньшей размерности. Идея этой процедуры заключается в следующем: каждый поток (нить) складывает два элемента массива; следовательно, количество элементов за один шаг уменьшается два раза. Дальше, таким образом, повторяются эти операции. Следует напомнить, что нить – это непосредственный исполнитель вычислений.

Функция сложения элементов массива (вектора) может быть написана в следующем виде:

```
_global_ void summOFVector(int* a, int* b)
{
_shared_ int data[BLOCK_SIZE];
int tid = thereadIdx.x;
int idx = blockIdxxblockDimx + threadIdx.x;
data[tid] = (idx < N) ? a[idx] : 0;
_syncthread();
 for(int s = blockDimx / 2; s > 0; s = s / 2;
{
if (tid < s) datd[tid]+ = data[tid + s];
– syncthreads();
}
if (tid == 0)
   b[blockIdx.x] = data[0];
}
```

Следует отметить, что для того чтобы программа стала рабочим экземпляром, должны быть записаны заголовки и используемые константы:

```
#include < stdio.h >
#include < stdlib.h >
#define N (100)
#define BLOCK_SIZE 512
```

2) Должна быть создана отдельная функция, в которой будут инициализироваться данные; она может быть написана в следующем виде:

```
void initvector < stdio.h >
{
vecTemp = ( float*)malloc(N * sizeof ( float));
vecX  = ( float*)malloc(N * sizeof ( float));
vecP = ( float*)malloc(N * sizeof ( float));
vecR = ( float*)malloc(N * sizeof ( float));
vecB = ( float*)malloc(N * sizeof ( float));
 for(int i = 0; i < N; i ++)
{
vecX[i] = vecP[i] = vecR[i] = vecTemp[i] = 0.0;
vecB[i] = 1.0 / N;
vecP[i] = 1.0;
}
}
```

3) В результате редукции функцией ядра получается выходной вектор. Здесь должна быть создана функция, которая производит суммирование по

количеству блоков элементов выходного вектора. Очевидно, что здесь выполняется небольшой объем вычислений. В результате выполнения данной функции будет получена сумма всех элементов заданного вектора.

4) В главной части программы производится вызов следующих функций:

- функцию инициализации;
- выделения памяти для переменных;
- копирования в эту память инициализированные переменные;
- функцию, которая вызывает функцию ядра;
- освобождения использованной памяти;
- вывода результата вычислений.

Контрольные вопросы.

1. Какова цель использования параллельного вычисления?
2. Назовите основные показатели эффективности параллельного вычисления.
3. Как происходит редукция элементов одномерного массива?
4. В чем сущность каскадной схемы суммирования перед последовательной схемой?
5. Какое количество операций сложения выполняется при суммировании элементов вектора традиционным (последовательным) способом и параллельным способом вычисления?
6. Каковы преимущества модифицированной схемы суммирования перед обыкновенной каскадной схемой?

Индивидуальное задание: в качестве исходного вектора можно принять любой вектор размерности n.

2 Лабораторная работа № 2. Алгоритм параллельного вычисления скалярного произведения векторов

Цель работы: закрепление навыков разработки алгоритма и программы, полученных в предыдущей работе вычислением скалярного произведения многомерных векторов параллельным методом.

Задание и порядок выполнения работы:

1) Подготовить данные для двух векторов, необходимые для вычисления их скалярного произведения.
2) Ввести эти данные в память.
3) Произвести вычисления через функцию ядра.
4) Вывести полученные данные.

5) Просмотр результатов.
6) Подготовить отчет о выполненной работе.

Некоторые методические указания по выполнению работы.

Следует напомнить, что скалярным произведением двух многомерных векторов является число (скаляр), равное сумме произведений соответствующих компонентов исходных векторов:

$$c = \sum_{i=}^{N} a_i \cdot b_i, \tag{4}$$

где $\vec{a} = \{a_1, a_2, \ldots, a_N\}$ и $\vec{b} = \{b_1, b_2, \ldots, b_N\}$ – исходные векторы.

Традиционный (последовательный) алгоритм также является простым, и он представляет собой суммирование произведений соответствующих компонентов исходных векторов. Параллелизм в данном случае может быть в двух случаях: вначале осуществляется вычисление произведений соответствующих компонентов векторов, в результате чего будет получен массив из N элементов; а затем производится вычисление суммы этих N элементов по каскадной схеме.

Задача о вычислении скалярного произведения двух векторов отличается от предыдущей задачи (лабораторной работы №1), когда рассматривалась задача о вычислении суммы числовой последовательности тем, что здесь рассматриваются вначале попарные умножения вместо попарных сложений, а затем такое же вычисление суммы. Схема алгоритма представлена в виде следующего рисунка:

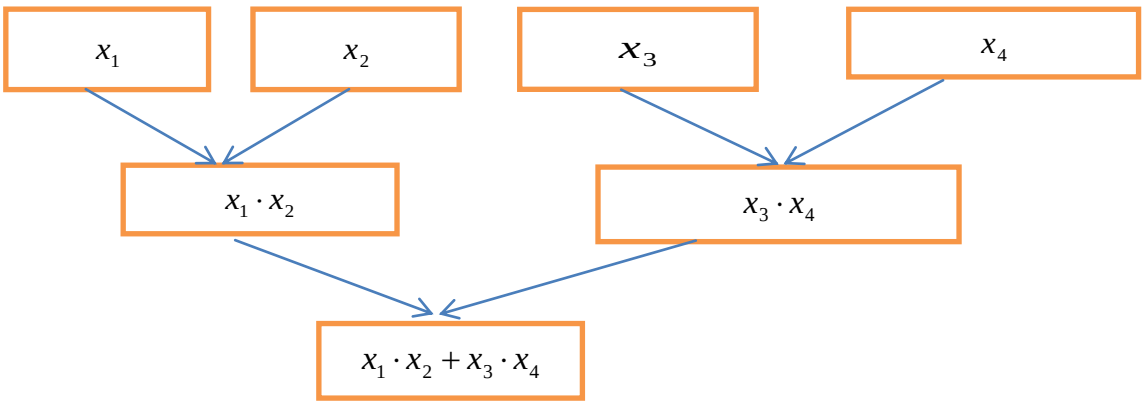

Рисунок 4 - Каскадная схема алгоритма вычисления скалярного произведения векторов

Перед тем как приступить к разработке программы, целесообразно построить схему алгоритма, аналогичную приведенной в предыдущей работе.

Для выполнения данной работы предлагаются следующие этапы работ:

1) Разработка функции ядра, которая выполняет основные вычисления, т.е. вычисления произведений соответствующих компонентов исходных векторов. Здесь можно использовать программу, приведенную в предыдущей работе (лабораторная работа № 1), где требуется использовать вместо операции суммирования операцию умножения. В результате выполнения данной функции будет получен массив, размерность которого равна количеству нитей. Затем будет вычислена сумма элементов полученного массива. Здесь параллелизм используется дважды: для вычисления попарных произведений компонентов исходных векторов и для вычисления суммы элементов получаемого при этом массива. Для вычисления суммы элементов массива используется способ редукции.

Заголовок программы и используемые константы должны быть показаны:

$\#include < stdio.h >$

$\#define\ i\min(a,b)$

$const\ \text{int}\ N = 60*1024;$

$const\ \text{int}\ threadsPerBlock = 256;$

$const\ \text{int}\ blocksPerGrid = i\min(32,((N + trheardsPerBlock - 1)/threardsPerBlock));$

2) Должна быть создана отдельная функция на хосте, в которой будут инициализироваться данные.

3) В главной части программы производятся инициализация данных, запуск функции и вывод результатов.

Контрольные вопросы.

1. Покажите аналогию между алгоритмами вычисления суммы последовательности чисел и вычисления скалярного произведения векторов.
2. Сколько нитей могут быть в алгоритме вычисления скалярного произведения векторов?
3. Какие будут преимущества в данном случае параллельной программы в сравнении с последовательной?
4. Сколько нитей могут быть в приведенном выше алгоритме ?

Индивидуальное задание: в качестве исходных векторов $\bar{a}$ и $\bar{b}$ можно рассматривать любые векторы размерности n.

3 Лабораторная работа № 3. Алгоритм умножения матрицы на вектор

Цель работы: изучение возможных параллельных методов решения одной из популярных задач линейной алгебры – задачу умножения матрицы на вектор.

Задание и порядок выполнения работы:

1) Изучить способы построения и анализа параллельных методов умножения матрицы на вектор.

2) Разработать алгоритм для каскадной схемы суммирования.

3) Составить программы на основе разработанного алгоритма и выполнить расчеты.

4) Провести анализ параллельной организации вычислений.

5) Оформить отчет по данной лабораторной работе.

Постановка задачи и методические указания по выполнению работы.

Требуется вычислить произведение матрицы A на вектор X. Из курса математики известно, что задача умножения матрицы на вектор определяется следующей формулой:

$$c_i = \sum_{j=1}^{M} a_{ij} \cdot x_j. \qquad (5)$$

Здесь $1 \le i \le N, \quad 1 \le j \le M$;

a_{ij} – элемент заданной матрицы A;

x_j – компонент заданного вектора X;

c_i – компонент вектора C, произведения матрицы A на вектор X.

Процесс вычисления по данной формуле предполагает повторения N однотипных операций по умножению строк матрицы A и вектора X. Получение каждой такой операции включает поэлементное умножение элементов строки матрицы A и вектора X и последующее суммирование полученных произведений. Общее количество необходимых скалярных операций оценивается величиной $2 \cdot M \cdot N$.

Как следует из выполняемых действий при определении произведения матрицы и вектора, параллельные способы решения задачи могут быть получены на основе параллельных алгоритмов суммирования (также как в лабораторной работе №1).

Выбор параллельного способа вычислений осуществляется из анализа информационных зависимостей в алгоритме умножения матрицы на вектор; отсюда следует выбор возможных способов распараллеливания. Из формулы для вычисления произведения матрицы и вектора следует, что:

- операции умножения отдельных строк матрицы на вектор являются независимыми и могут быть выполнены параллельно;

- умножение каждой строки на вектор включает независимые операции поэлементного умножения, и также могут быть выполнены параллельно;

- суммирование получаемых произведений в каждой операции умножения строки матрицы на вектор могут быть выполнены по одному из ранее рассмотренных вариантов алгоритма суммирования (последовательный алгоритм или каскадная схема).

Таким образом, максимально необходимое количество процессоров определяется величиной $M \cdot N$.

Использование такого количества процессоров может быть представлено следующим образом. Множество процессоров P разбивается на n групп: $P_1, P_2, \ldots, P_n$. Каждый процессор является набором процессоров для выполнения операции умножения отдельной строки матрицы на вектор.

В начале операций на каждый процессор группы пересылаются элемент строки матрицы A и соответствующий элемент вектора X. Затем каждый процессор выполняет операцию умножения. В дальнейшем вычисления выполняются по каскадной схеме суммирования.

При реализации параллельного алгоритма целесообразно выделить начальный этап по загрузке используемых процессоров исходными данными. Наиболее просто такая инициализация обеспечивается при помощи одной параллельной операции пересылки данных. При организации множества процессоров может оказаться полезной двухуровневое управление процессом начальной загрузки, при которой центральный управляющий процессор обеспечивает рассылку строк матрицы и вектора к управляющим процессорам процессорных групп $P_i, 1 \le i \le n,$ которые, в свою очередь, рассылают элементы строк матрицы и вектора по исполнительным процессорам.

При использовании n процессоров для умножения матрицы A на вектор X может быть использован параллельный алгоритм построчного умножения. Это означает, что строки матрицы распределяются по процессорам построчно, и каждый процессор реализует операцию умножения какой-либо отдельной строки матрицы A на вектор X.

Другой возможный способ организации параллельных вычислений может состоять в построении конвейерной схемы для операции умножения строки матрицы на вектор (скалярного произведения векторов) путем расположения всех имеющихся процессоров в виде линейной последовательности. Подобная схема вычислений может быть определена следующим образом. Она может быть представлена как множество процессоров в виде линейной последовательности (рисунок 5):

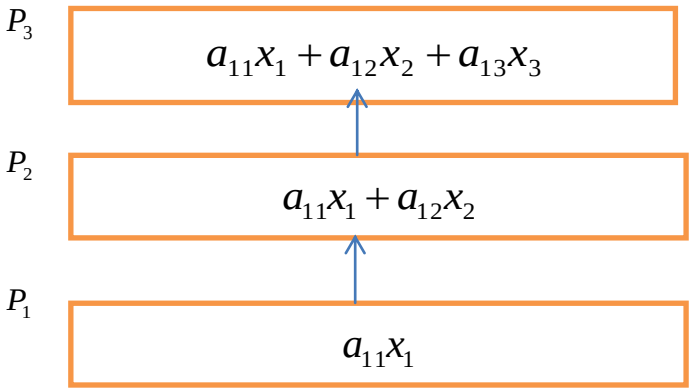

Рисунок 5- Состояние линейного конвейера для операции умножения строки матрицы на вектор после выполнения двух итераций

Каждый процессор P_j, $1 \le j \le n$ используется для умножения элементов $j-$ столбца матрицы и $j-$ элемента вектора X. Выполнение вычислений на каждом процессоре P_j состоит в следующем:

- вызывается очередной элемент $j-$ столбца матрицы;
- выполняется умножение элементов a_{ij} и x_j;
- вызывается результат вычислений S предшествующего процессора;
- выполняется сложение значений $S = S + a_{ij}x_j$;
- полученный результат S пересылается следующему процессору.

При инициализации описанной схемы необходимо выполнить ряд дополнительных действий:

- при выполнении первой итерации каждый процессор дополнительно запрашивает элемент вектора x_j;
- для синхронизации вычислений (при выполнении очередной итерации схемы запрашивается результат вычисления предшествующего процессора) на этапе инициализации процессор P_j, $1 \le j \le n$ выполняет $j-1$ цикл ожидания.

Для однородности описанной схемы для первого процессора P_1, у которого нет предшествующего процессора, целесообразно ввести пустую операцию сложения $S = 0$, $S = S + a_{ij}x_j$.

Контрольные вопросы

1. Какие операции являются параллельными при вычислении произведения матрицы и вектора?
2. Как обеспечивается инициализация данных на начальном этапе загрузки процессоров?
3. Как осуществляется двухуровневое управление процессом начальной загрузки процессоров?

4. Объясните сущность конвейерной схемы для операции умножения строки матрицы на вектор.

Индивидуальное задание: в качестве исходных данных можно рассматривать матрицу A любого порядка n и вектор X размерности n.

4 Лабораторная работа № 4. Произведение двух матриц

Цель работы: изучение возможных параллельных методов умножения матриц.

Задание и порядок выполнения работы:

1) Изучить способы построения и анализа параллельных методов умножения матриц.

2) Разработать алгоритм для каскадной схемы.

3) Разработать алгоритм для модифицированной схемы.

4) Составить программы на основе разработанных алгоритмов и выполнить расчеты.

5) Провести анализ возможных способов параллельной организации вычислений.

6) Оформить отчет по данной лабораторной работе.

Постановка задачи умножения матриц и методические указания по выполнению работы.

Из курса математики известно, что в результате умножения двух квадратных матриц порядка n будет получена также квадратная матрица такого же порядка. Каждый элемент c_{ij} новой матрицы C при умножении матриц A и B определяется с помощью следующей формулы:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad 1 \le i \le n, \quad 1 \le j \le n. \tag{6}$$

Анализ возможных способов параллельного выполнения данной задачи может быть проведен по аналогии с рассмотрением задачи умножения матрицы на вектор. Оставив подробный анализ для самостоятельного изучения, покажем на примере задачи матричного умножения использование некоторых общих подходов, позволяющих формировать параллельные способы решения сложных задач.

Процесс умножения двух матриц порядка n требует для своего решения выполнение большого количества операций, равное n^3 скалярных умножений и сложений. Информационный граф алгоритма при большом размере матриц становится достаточно объемным и, как результат, непосредственный анализ этого графа затруднен. После выявления

информационной независимости выполняемых вычислений могут быть предложены многочисленные способы распараллеливания алгоритма.

Можно предлагать следующий вариант: алгоритм выполнения умножения матриц может быть рассмотрен как процесс решения n независимых подзадач. Каждая из этих подзадач осуществляет умножение матрицы A на столбец матрицы B. Это приводит к более компактному представлению алгоритма, значительно упрощает проблему выбора эффективных способов распараллеливания вычислений, позволяет использовать типовые параллельные методы выполнения операций в качестве конструктивных элементов при разработке параллельных способов решения сложных вычислительных задач.

Другим вариантом при построении параллельных способов выполнения умножения матриц является блочное представление матриц. Пусть рассматривается данный способ организации вычислений более подробно. Здесь предполагается, что количество процессоров равно следующей величине: $Q = k^2$, а количество строк и столбцов матрицы является кратным величине $k = \sqrt{Q}$, т.е. $n = m \cdot k$. Исходные матрицы A и B, а также результирующая матрица C представлены в виде наборов прямоугольных блоков размера $m \, x \, m$. Тогда операцию умножения матриц A и B в блочном виде можно представить следующим образом:

$$\begin{pmatrix} C_{11} C_{12} \dots C_{1k} \\ C_{21} C_{22} \dots C_{2k} \\ . \ . \ . \ . \ . \\ C_{k1} C_{k2} \dots C_{kk} \end{pmatrix} = \begin{pmatrix} A_{11} A_{12} \dots A_{1k} \\ A_{21} A_{22} \dots A_{2k} \\ . \ . \ . \ . \ . \\ A_{k1} A_{k2} \dots A_{kk} \end{pmatrix} \cdot \begin{pmatrix} B_{11} B_{12} \dots B_{1k} \\ B_{21} B_{22} \dots B_{2k} \\ . \ . \ . \ . \ . \\ B_{k1} B_{k2} \dots B_{kk} \end{pmatrix} \quad (7)$$

Здесь каждый блок C_{ij} матрицы C определяется из следующей формулы:

$$C_{ij} = \sum_{l=1}^{k} A_{il} \cdot B_{lj} = A_{i1} B_{1j} + A_{i2} B_{2j} + \dots + A_{ik} B_{kj}, \quad 1 \le i \le n, 1 \le j \le n. \quad (8)$$

Анализ показывает взаимную независимость вычислений блоков C_{ij} матрицы C. Отсюда можно сделать вывод о том, что возможный подход для параллельного выполнения вычислений может состоять в выделении для расчетов, связанных с получением отдельных блоков C_{ij}, разных процессоров. Применение подобного подхода позволяет получить эффективные параллельные методы умножения матриц.

Для разработки программы решения поставленной здесь задачи должны быть выполнены следующие этапы:

1) Ввод данных для расчетов, т.е. ввод элементов исходных матриц.
2) Произвести вычисления с помощью функции ядра.
3) Скопировать вычисленные данные.
4) Вывод результатов.
5) Высвободить используемые ресурсы.

Функция ядра осуществляет умножение матриц. Здесь элементы матрицы представляются как двумерные массивы. Вычисления производятся с помощью приведенной выше формулы.

Пусть будут показан один из вариантов функции ядра, которая будет осуществлять умножение матриц:

```
_global _void mult(int*a, int*b, int*c)
{
int bx = blockIdx.x;
int by = blockIdx.y;
int tx = threadIdx.x;
int ty = threadIdx.y;
float sum = 0.0;
int ia = N *128* by + N * ty;
int ib = 128* bx + tx;
for (int k = 0; k < N;k ++)
   sum+ = a[ia + k]* b[ib + k * N];
int ic = N *128* by +128* bx;
c[ic + N * ty + tx] = sum;
}
```

В главной части программы выполняются все указанные выше этапы.

Контрольные вопросы.

1. Что такое информационная независимость, и какую роль играет она в определении параллелизма алгоритма?
2. О каких типовых параллельных алгоритмах идет речь, когда рассматривается процесс умножения матриц?
3. Что такое блочное представление матриц?
4. Почему вычисления блоков результирующей матрицы C являются информационно независимыми?

Индивидуальное задание: в качестве исходных данных можно рассматривать матрицы A и B любого порядка n.

5 Лабораторная работа № 5. Сортировка одномерного массива по возрастанию (убыванию)

Цель работы: изучение возможных параллельных методов организации сортировок одномерного массива.

Задание и порядок выполнения работы:

1) Изучить способы построения и анализа параллельных методов организации сортировок одномерного массива.

2) Разработать последовательный алгоритм.

3) Разработать алгоритм для каскадной схемы.

4) Составить программы на основе разработанных алгоритмов и выполнить расчеты.

5) Провести анализ возможных способов последовательной и параллельной организации вычислений.

6) Оформить отчет по данной лабораторной работе.

Постановка задачи сортировок и методические указания по выполнению работы.

Одной из часто встречающихся на практике задач обработки данных является сортировка одномерного массива. Известно, что такие задачи часто возникают в математической статистике, которые требуют выполнения достаточно большого объема вычислительной работы. Одним из популярных примеров решения таких задач является упорядочение статистического ряда по возрастанию или убыванию. Поэтому сортировка является одной из типовых проблем обработки данных.

Задача сортировки может быть сформулирована в следующем виде. Дана некоторая неупорядоченная последовательность чисел:

$$a_1, a_2, a_3, \ldots, a_n. \tag{9}$$

Требуется эту последовательность преобразовать к виду, в котором элементы ее будут расположены в порядке монотонного возрастания, т.е.

$$a_1 \le a_2 \le a_3 \le \ldots \le a_n. \tag{10}$$

Существуют различные способы решения данной задачи; например, сортировка включением, сортировка Шелла, пузырьковая сортировка и другие. Подсчитано, что для решения данной задачи необходимо выполнить n^2 операций. Это означает, что при достаточно большом значении n объем вычислительной работы будет очень большим.

Анализ способов сортировки показал, что многие из этих способов основаны на применении базовой операции «сравнить и переставить», которая состоит из операции сравнения элементов массива и перестановки их при условии соответствия условию сортировки. Обычно, в традиционных способах программирования, эти операции записываются в следующем виде:

```
for ( i = 1; i < n; i++ )
   for ( j = 0; j < n – 1; j++ )

if (a[i] > a[j]) {
b = a[i];
a[i] + a[j];
a[j] = b;
}
```

Для параллельного программирования эти операции могут быть организованы разными подходами.

Пусть рассматривается следующая ситуация, когда количество процессоров совпадает с числом сортируемых значений, т.е. равно n. В этом случае сравнение значений элементов a_i и a_j, расположенных на процессорах соответственно P_i и P_j, можно организовать следующим образом:

– выполнить взаимообмен имеющихся на процессорах P_i и P_j значений (с сохранением на этих процессорах исходных элементов);

– сравнить на каждом процессоре P_i и P_j получившиеся одинаковые пары значений (a_i, a_j); результаты сравнения используются для разделения данных между процессорами – на одном процессоре (например, P_i) остается меньший элемент, другой процессор (т.е. P_j) запоминает для дальнейшей обработки большее значение из этой пары.

Рассмотренное параллельное обобщение базовой операции сортировки может быть надлежащим образом адаптировано и для случая, когда количество процессоров $(< n)$ является меньшим числа упорядочиваемых значений. В данной ситуации каждый процессор будет содержать уже не единственное значение, а часть сортируемого набора данных, т.е. блоки A_i и A_j. Эти блоки обычно упорядочиваются в самом начале сортировки на каждом процессоре в отдельности при помощи какого-либо быстрого алгоритма (предварительная стадия параллельной сортировки). Далее, следуя схеме одноэлементного сравнения, взаимодействие пары процессоров P_i и P_j для совместного упорядочения содержимого блоков A_i и A_j может быть осуществлено следующим образом:

– выполнить взаимообмен блоков между процессорами P_i и P_j;

– объединить блоки A_i и A_j на каждом процессоре в один отсортированный блок двойного размера (при исходной упорядоченности блоков A_i и A_j процедура их объединения сводится к быстрой операции слияния упорядоченных наборов данных);

– разделить полученный двойной блок на две равные части и оставить одну из этих частей (например, с меньшими значениями данных) на процессоре P_i, а другую часть (с большими значениями соответственно) – на процессоре P_j.

Следует отметить, что сформированные в результате такой процедуры блоки на процессорах P_i и P_j совпадают по размеру с исходными блоками A_i и A_j, и все значения, расположенные на процессоре P_i, являются меньшими значений на процессоре P_j.

Контрольные вопросы

1. В каких практических задачах возникает проблема решения задачи сортировки?

2. Если требуется упорядочение одномерного массива по убыванию, то какое отличие должно быть от сортировки по возрастанию?

3. Для чего проводится сортировка в каждом блоке?

4. Как происходит слияние упорядоченных наборов данных?

Индивидуальное задание: в качестве исходных данных можно рассматривать любой неупорядоченный одномерный массив.

6 Лабораторная работа № 6. Решение систем линейных алгебраических уравнений

Цель работы: получить навыки использования методов параллельных вычислений для решения системы линейных алгебраических уравнений.

Задание и порядок выполнения работы:

1) Изучить возможность использования параллельных вычислений для решения системы линейных алгебраических уравнений и способов построения алгоритмов для их реализации.

2) Выбрать алгоритм для решения данной задачи.

3) Составить программы на основе разработанного алгоритма и выполнить расчеты.

5) Провести анализ других возможных способов параллельной организации вычислений.

6) Оформить отчет по данной лабораторной работе.

Постановка задачи и методические указания по выполнению работы.

Математическое моделирование научных и научно-технических задач в основном приводит к проблеме решения задач, как для обыкновенных дифференциальных уравнений, так и для дифференциальных уравнений в частных производных. Аналитическое решение большинства из этих задач не представляется возможным или приводит к сложным, трудным для практического использования, результатам.

Для решения таких задач обычно используются численные методы, основанные на применении метода дискретизации (метода конечных разностей). В результате такой дискретизации дифференциальные уравнения заменяются системой линейных алгебраических уравнений. С целью достижения высокой точности вычислений, количество «точек» в сеточных уравнениях может достигать очень больших значений. Это означает, что количество алгебраических уравнений в получаемой системе и неизвестных в них будут очень большими.

Системы линейных алгебраических уравнений используются также для решения других практических задач, в частности, для описания задач математической экономики и теории управления; они получаются в результате экономико-математического моделирования многих задач экономики, бизнеса и управления. В связи с этим возникает необходимость решения системы линейных алгебраических уравнений, когда количество уравнений и неизвестных в них очень большое. Поэтому актуальным является применение методов высокоскоростных вычислений для решения таких уравнений.

Постановка задачи решения системы линейных алгебраических уравнений общеизвестна из курса математики; здесь требуется решить следующую систему n уравнений с n неизвестными:

$$\begin{pmatrix} a_{11}\, a_{12} \ldots a_{1k} \\ a_{21}\, a_{22} \ldots a_{2k} \\ . \quad . \quad . \quad . \quad . \\ a_{k1}\, a_{k2} \ldots a_{kk} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ : \\ x_n . \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ : \\ b_n \end{pmatrix} \qquad (11)$$

(1)

Здесь система уравнений записана в матричной форме; где матрица:

$$A = \begin{pmatrix} a_{11}\, a_{12} \ldots a_{1k} \\ a_{21}\, a_{22} \ldots a_{2k} \\ . \quad . \quad . \quad . \quad . \\ a_{k1}\, a_{k2} \ldots a_{kk} \end{pmatrix}, \qquad (12)$$

векторы: $X = \{ x_1, x_2, \ldots, x_n \}$, $B = \{ b_1, b_2, \ldots, b_n \}$. Требуется найти значения компонентов неизвестного вектора X, когда заданы матрица A и вектор B.

Из курса «Численные методы» известно, что существуют многочисленные методы решения системы линейных алгебраических уравнений. Для случаев, когда количество неизвестных и уравнений в системе имеет ограниченное значение, существуют программные продукты, и поэтому решение такой задачи не представляет особого труда. Однако для больших систем существует проблема, связанная с временем выполнения вычислений.

Одним из методов решения системы уравнений (1) является так называемый матричный метод. Сущность данного метода заключается в следующем: вначале определяется обратная матрица A^{-1}, а затем умножение «слева» обеих частей уравнения (1) на эту обратную матрицу позволяет определить компонентов неизвестного вектора X:

$$X = A^{-1} \cdot B. \qquad (13)$$

Для решения данной задачи можно воспользоваться тем же методом, который был использован в лабораторной работе № 3.

Очевидно, вначале должна быть определена обратная матрица A^{-1}, для определения которой можно использовать известные правила из курса математики.

Для решения таких популярных задач линейного программирования, возникающих в результате экономико-математического моделирования, используются т.н. преобразования Жордана-Гаусса. Метод решения системы линейных алгебраических уравнений, основанный на преобразованиях Жордана-Гаусса, является одним из эффективных методов. Поэтому предлагается изучить данный метод для определения параллелизма в его алгоритме.

Сущность метода Жордана-Гаусса заключается в следующем: рассматривается расширенная матрица системы уравнений (1):

$$\begin{pmatrix} a_{11}\, a_{12} \ldots a_{1k} \\ a_{21}\, a_{22} \ldots a_{2k} \\ . \quad . \quad . \quad . \quad . \\ a_{k1}\, a_{k2} \ldots a_{kk} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ : \\ b_n \end{pmatrix}; \qquad (14)$$

производятся такие ее преобразования, в результате которых вместо исходной матрицы A будет получена единичная матрица. Из курса «Численные методы» известно, что для этого используются следующие формулы:

$$a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}, \quad 1 \le i \le n; \quad 1 \le j \le n+1; \qquad \text{для} \quad i \ne k; \qquad (15)$$

$$a_{kj} = \frac{a_{kj}}{a_{kk}}, \quad 1 \le j \le n+1; \qquad \text{для} \quad i = k. \qquad (16)$$

В результате преобразований по формулам (4) и (5) будет получена следующая матрица:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ . & . & . & \cdots & . \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_{1n+1} \\ a_{2n+1} \\ a_{3n+1} \\ \cdots \\ a_{nn+1} \end{pmatrix} \qquad (17)$$

В данной матрице, последний столбец получен из последнего столбца матрицы (3) после преобразований; элементы этого столбца являются значениями неизвестных величин, т.е. решением системы уравнений:

$$x_i = a_{in+1}, \quad 1 \le i \le n. \qquad (18)$$

В данной лабораторной работе требуется определить параллелизм в алгоритме решения данной задачи и разработать схему алгоритма.

Контрольные вопросы:

1. Назовите другие методы решения системы линейных алгебраических уравнений.

2. Какому условию должна удовлетворять матрица A системы линейных алгебраических уравнений ?

3. По вашему мнению, какие операции в данной задаче могут быть выполнены параллельно ?

4. Можно ли организовать параллельное вычисление при определении обратной матрицы A^{-1}?

Индивидуальное задание: в качестве примера можно использовать любую систему линейных алгебраических уравнений.

7 Лабораторная работа № 7. Решение краевой задачи для обыкновенного дифференциального уравнения второго порядка

Цель работы: показать возможность использования методов параллельных вычислений для решения дифференциальных уравнений.

Задание и порядок выполнения работы:

1) Изучить возможность использования параллельных вычислений для решения дифференциальных уравнений и способов построения алгоритмов для их реализации.

2) Разработать алгоритм для каскадной схемы.

3) Разработать алгоритм для модифицированной схемы.

4) Составить программы на основе разработанных алгоритмов и выполнить расчеты.

5) Провести анализ возможных способов параллельной организации вычислений.

6) Оформить отчет по данной лабораторной работе.

Постановка задачи и методические указания по выполнению работы

Во многих прикладных задачах управления объектами, физики, механики и инженерных расчетов часто возникает проблема решения краевой задачи для обыкновенных дифференциальных уравнений второго порядка. Решение этой задачи имеет важное значение при изучении различных процессов, особенно динамических процессов, требующих быстрого получения результатов решения задачи. Возникают большие проблемы особенно при решении сложных задач, связанных с управлением быстропротекающими процессами.

Данная задача была предметом изучения различных дисциплин. В отличие от них, в данном случае рассматриваются вопросы, связанные с решением данной задачи с применением технологии высокоскоростных вычислений, в частности, с применением способов параллельных вычислений.

Постановка математической задачи. Пусть даны:

дифференциальное уравнение

$$\frac{d^2y}{dx^2} + p(x) \cdot \frac{dy}{dx} + g(x) \cdot y = f(x) \qquad (19)$$

и граничные условия

$$x = a, \quad y(a) = c, \quad y(b) = d. \tag{20}$$

Здесь a, b, c, d – заданные числа, $p(x)$, $g(x)$, $f(x)$ – известные функции. Требуется найти значения неизвестной функции $y(x)$ в промежутке $a \le x \le b$, удовлетворяющие уравнению (1) и граничным условиям (2).

*Метод решения задачи***.** Поставленную задачу (1)-(2) в практических случаях невозможно решить аналитическими методами. Поэтому для решения таких задач используется численный метод, метод конечных разностей. Сущность этого метода подробно рассмотрена во время изучения дисциплины «Математическое и компьютерное моделирование».

Использование метода конечных разностей приводит к решению системы линейных алгебраических уравнений с трехдиагональной матрицей:

$$A_{i\cdot} \cdot y_{i+1} - C_i \cdot y_i + B_i \cdot y_{i-1} = D_i, \quad i = 1,2,3, \ldots, n-1. \tag{21}$$

Здесь $A_i = 1 - \frac{h}{2} \cdot p_i; \quad B_i = 1 + \frac{h}{2} \cdot p_i; \quad C_l = h^2 \cdot g_i; \quad D_i = h^2 \cdot f_i.$

Формула (3) определяет систему алгебраических уравнений относительно неизвестных значений искомой функции y_i. Количество этих уравнений равно $n-1$. Из гранчных условий (2) можно получить два уравнения:

$$y_0 = c; \quad y_n = d. \tag{22}$$

Формулы (3) и (4) представляют собой систему $n+1$ уравнений с $n+1$ неизвестными $y_i, i = 0,1,2, \cdots \ldots, n.$ Если будет решена данная система алгебраических уравнений, то задача будет решена, т.е. будут найдены значения искомой функции $y_i = y(x_i)$.

Для решения системы линейных алгебраических уравнений (3) используется известный метод прогонки, сущность которого заключается в следующем. Решение системы определяется следующей формулой:

$$y_i = \alpha_{i+1} \cdot y_{i+1} + \beta_{i+1}, \tag{23}$$

где α_i, β_i – прогоночные коэффициенты; они пока неизвестные.

Процесс определения этих коэффициентов называется прямой прогонкой; для определения их используются формулы:

$$\alpha_1 = 0; \quad \beta_1 = 1; \quad \alpha_{i+1} = \frac{B_i}{C_i - A_i \cdot \alpha_i \cdot}; \quad \beta_{i+1} = \frac{A_{i\cdot} \cdot \beta_i - D_i}{C_i - A_i \cdot \alpha_i}. \tag{24}$$

Определение значений искомой функции y_i называют обратной прогонкой и для этого используются следующие формулы: сперва вторая формула (4), а затем формула (5).

Алгоритм решения задачи составлен в следующем виде:

- вычисление по известным формулам коэффициентов A_i, B_i, C_i, D_i системы уравнений (5);

- вычисление прогоночных коэффициентов по формулам (6);
- определение искомых значений функции по формулам (4) и (5);
- вывод полученных результатов.

Теперь возникает следующая задача: какие операции должны быть выполнены параллельно ?

Очевидно, что вычисление значений заданных функций $p(x)$, $g(x)$, $f(x)$, коэффициентов A_i, B_i, C_i, D_i информационно независимы, поэтому они могут быть вычислены параллельно. На следующем этапе вычисление коэффициентов прогонки α_i и β_i могут быть вычислены также параллельно. Вычисление значений искомой функции y_i по формуле (5) может быть осуществлено конвейерным способом.

Контрольные вопросы

1. Какая задача называется краевой задачей?
2. В чем сущность конечно-разностного метода решения краевой задачи?
3. Какова погрешность данного метода?
4. В чем заключается сущность метода прогонки?
5. В чем особенность матрицы системы алгебраических уравнений (5)?

Индивидуальное задание: должна быть задана краевая задача для обыкновенного дифференциального уравнения второго порядка; должны быть заданы конкретные функции: $p(x)$, $g(x)$, $f(x)$, также постоянные краевых условий: a, b, c, d.

Требуется определить параллелизм в алгоритме решения данной задачи; составить схему алгоритма параллельного вычисления.

Заключение

В рассмотренных здесь лабораторных работах предложены простые, вместе с тем, актуальные и требующие больших вычислений, задачи. Выполнение заданий в этих работах требует от магистрантов самостоятельного освоения многих вопросов, связанных с определением параллелизма в алгоритмах решения предложенных задач. Рекомендуется использовать для программирования язык Си.

Список литературы

1Будущее прикладной математики: Лекции для молодых исследователей. От идей к технологиям / Под ред. Г.Г. Малинецкого. – М.: КомКнига, 2008. -512 с.

2 Марчук Г.И. Методы вычислительной математики. – СПб.: БХВ-Петербург, 2009. – 469 с.

3 Воеводин В.В., Воеводин Вл. В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004.- 608 с.

4 Корнеев В.Д. Параллельное программирование в MP1.- М.: Издательство «Регулярная и хаотическая динамика», 2003.- 303 с.

5. Гергель В.П. Теория и практика параллельных вычислений. – М.: «Интернет УИТ: Бином», 2013.- 413 с.

6 Антонов А.С. Введение в параллельные вычисления: методическое пособие.- М.: Изд-во МГУ имени М.В. Ломоносова, 2002. - 69 с.

7 Букатов А.А., Дацюк В.Н., Жегуло А.И. Программирование многопроцессорных вычислительных систем. –Ростов-на-Дону: Издательство ООО «ЦВВР», 2003.- 208 с.

8 Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 400 с.

9 Федотов И.Е. Модели параллельного программирования. – М.: «Солон-Пресс», 2012.- 342 с.

10 Барский А.Б. Параллельные информационные технологии.- М.: «Интернет УИТ», 2013. – 504 с.

11 Лацис А.О. Параллельная обработка данных. – М.: «Академия», 2010.- 521 с.

12 Шпаковский Г.И. Программирование для многопроцессорных систем стандарта MPI.-Минск: Изд-во БГУ, 2002. – 325 с.

13 Давыдов В.Г. Технологии программирования С++. –Санкт-Петербург, 2005.- 456 с.

14 Страуструп Б. Программирование: принципы и практика использования С++. Пер. с англ. – М.: ООО «И.Д.Вильямс», 2011. 1248 с.

15 Ластовецкий А.Л., Калинов А.Я., Ледовских И.Н., Арапов Д.М., Посыпкин М.А. Язык и система программирования для высокопроизводительных параллельных вычислений на неоднородных сетях// Труды Института системного программирования. –том 1.– 2000.- С. 5-25.

Содержание

Сводный план 2016г. поз. 148

Зауытбек Куралбаевич Куралбаев

Технологии высокоскоростных вычислений

Методические укзания по выполнению лабораторных работ
для магистрантов специальности
6М070400-Вычислительная техника и программное обеспечение

Редактор Н.М.Голева
Специалист по стандартизации Н.К.Молдабекова

Подписано в печать
Тираж 20 экз.
Объем 1,9 уч. изд. л.

Формат 60х84/16
Бумага типографская № 1
Заказ Цена 938 тн.

Копирально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, Байтурсынулы, 126