МИНИСТЕРСТВО НАУКИ и ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине «Параллельные вычисления / Parallel computing»
для магистрантов направления подготовки 09.04.03 «Прикладная информатика» (направленность (профиль) «Управление знаниями»)

Ставрополь, 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данный курс лабораторных работ необходим для обучения практическим навыкам программирования параллельных алгоритмов, приведенных в лекциях "Параллельные вычисления".

За основу для организации параллельных вычислений студентам предлагается ознакомиться с идеей построения параллельной среды программирования PVM (Параллельная Виртуальная Машина).

Лабораторная работа № 1. Основные функции MPI

Цель лабораторной работы: дать представление о построении простых параллельных программ на языке параллельного программирования MPI.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Система MPI является основным средством программирования таких современных высокопроизводительных мультикомпьютеров, как Silicon Graphics Origin 2000, Cray T3D, Cray T3E, IBM SP2 и многих других. MPI работает на самых разных архитектурах мультикомпьютеров, как с распределенной памятью, так и с разделяемой памятью. Кроме того, MPI работает на сетях компьютеров (кластерах) однородных и гетерогенных. Количество компьютеров в сети может быть от одного и более. Важнейшей особенностью MPI является то, что пользователь при написании своих параллельных программ не должен учитывать все эти архитектурные особенности конкретных мультикомпьютеров, поскольку MPI предоставляет пользователю виртуальный мультикомпьютер с распределенной памятью и с виртуальной сетью связи между виртуальными компьютерами. Пользователь заказывает количество компьютеров, необходимых для решения его задачи, и определяет топологию связей между этими компьютерами. MPI реализует этот заказ на конкретной физической системе. Ограничением является объем оперативной памяти физического мультикомпютера. Таким образом пользователь работает в виртуальной среде, что обеспечивает переносимость его параллельных программ. Система MPI представляет собой библиотеку средств параллельного программирования для языков C и Fortran 77.

Одной из целей, преследуемых при решении задач на вычислительных системах, в том числе и на параллельных, – является эффективность. Эффективность параллельной программы существенно зависит от соотношения времени вычислений ко времени коммуникаций между компьютерами (при обмене данными). И чем меньше в процентном отношении доля времени, затраченного на коммуникации, в общем времени вычислений, тем больше эффективность. Для параллельных систем с передачей сообщений оптимальное соотношение между вычислениями и коммуникациями обеспечивают методы крупнозернистого распараллеливания, когда параллельные алгоритмы строятся из крупных и редко взаимодействующих блоков [2–8]. Задачи линейной алгебры, задачи, решаемые сеточными методами и многие другие, достаточно эффективно *распараллеливаются крупнозернистыми методами*.

MPMD - модель вычислений. MPI - программа представляет собой совокупность автономных процессов, функционирующих под управлением своих собственных программ и взаимодействующих посредством стандартного набора библиотечных процедур для передачи и приема сообщений. Таким образом, в самом общем случае MPI - программа реализует MPMD - модель программирования (Multiple program - Multiple Data).

SPMD - модель вычислений. Все процессы исполняют в общем случае различные ветви одной и той же программы. Такой подход обусловлен тем обстоятельством, что задача может быть достаточно естественным образом разбита на подзадачи, решаемые по одному алгоритму. На практике чаще всего встречается именно эта модель программирования (Single program - Multiple Data) [1,12].

Последнюю модель иначе можно назвать моделью *распараллеливания по данным.* Кратко, суть этого способа заключается в следующем. Исходные данные задачи распределяются по процессам (ветвям параллельного алгоритма), а алгоритм является одним и тем же во всех процессах, но действия этого алгоритма распределяются в соответствии с имеющимися в этих процессах *данными*. Распределение действий алгоритма заключается, например, в присвоении разных значений переменным одних и тех же циклов в разных ветвях, либо в исполнении в разных ветвях разного количества

витков одних и тех же циклов и т.п. Другими словами, процесс в каждой ветви следует различными путями выполнения на той же самой программе.

Указанные модели, помимо поддержки на языковом уровне, поддерживаются архитектурами таких самых современных суперкомпьютеров как: ASCI RED (более 9000 Pentium PRO/200 объединены в единую систему, имеет быстродействие около 3,2 Tflops), и ASCI WAIT (8192 - , имеет быстродействие 12,2 Tflops), Cray T3D, Cray T3E, IBM SP2.

Лабораторное задание

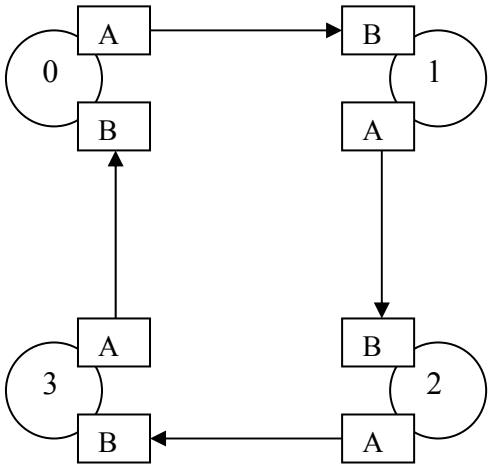
1. Изучить теоретический материал данной лабораторной работы.
2. Выполнить задания

Задание 1. Программирование конвейера.

Ветвь 0 пересылает некоторые данные ветви 1, ветвь 1 принятые данные передает ветви 2, ветвь 2 принятые данные передает ветви 3 и т.д. по цепочке увеличения номеров. И, наконец, ветвь 0 принимает пересылаемые данные от ветви size-1 и выводит на экран свой номер и принятые данные. Т.е. пересылка информации по кольцу компьютеров. В этом примере программу алгоритма нужно написать настраиваемой автоматически на размер вычислительной системы, т.е. алгоритм не должен зависеть от числа компьютеров и программа должна запускаться на любом допустимом количестве компьютеров.

Задание 2. Программирование кольцевых сдвигов данных

Все ветви одновременно пересылают некоторые свои данные ветвям с номерами на единицу большими, чем передающие ветви. Т.е. пересылка информации вдоль кольца компьютеров, см. рисунок ниже. В этом примере программу алгоритма нужно написать настраиваемой автоматически на размер вычислительной системы, т.е. алгоритм не должен зависеть от числа компьютеров и программа должна запускаться на любом допустимом количестве компьютеров.



.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся

выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Методы распараллеливания и модели программ, поддерживаемые MPI.
2. Свойства MPI, обеспечивающие переносимость параллельных программ и независимость от вычислительных систем.
3. Пояснить основные принципы выполнения ниже указанных функций.
4. Команда компиляции MPI–программ.
5. Команда запуска MPI–программ.
6. Действия системы по команде mpirun .

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

Лабораторная работа № 2. Виртуальные топологии.

Цель лабораторной работы: дать представление о построении простых параллельных программ на языке параллельного программирования MPI.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Для освоения программирования Декартовых топологий и освоения совмещенных функций приема-передачи данных.

В примере выполняется сдвиг данных соседним ветвям вдоль обоих координат компьютеров на топологии двумерный тор на один шаг, т.е. все ветви параллельной программы одновременно передают данные соседним ветвям, например, в сторону увеличения значений вдоль одной и вдоль другой координаты компьютеров.

```
#include <mpi.h>
#include <stdio.h>
#define DIMS 2
int main(int argc, char** argv)
  { int        rank, size, i, A, B, dims[DIMS];
    int        periods[DIMS], sourc1, sourc2, dest1, dest2;
    int        reorder = 0;
    MPI_Comm   tor_2D;
    MPI_Status status;
    MPI_Init(&argc, &argv);
  /* Каждая ветвь узнает общее количество ветвей */
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  /* и свой номер: от 0 до (size-1) */
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    A = rank;
    B = -1;
  /* Обнуляем массив dims и заполняем массив periods для топологии
"двумерный тор" */
      for(i = 0; i < DIMS; i++) { dims[i] = 0; periods[i] = 1;
}
  /* Заполняем массив dims, в котором указываются размеры решетки */
    MPI_Dims_create(size, DIMS, dims);
  /* Создаем топологию "двумерный тор" с коммуникатором top2D */
    MPI_Cart_create(MPI_COMM_WORLD, DIMS, dims, periods,
reorder,
                                                        &
tor_2D);
  /* Каждая ветвь находит своих соседей вдоль координат, в направлении
бо□льших значений
   * координат */
    MPI_Cart_shift(tor_2D, 0, 1, &sourc1, &dest1);
    MPI_Cart_shift(tor_2D, 1, 1, &sourc2, &dest2);
  /* Каждая ветвь одновременно передает свои данные (значение переменной
A) соседней ветви
   * с бо□льшим значением координаты и принимает данные в B от
соседней ветви
   * с меньшим значением координаты вдоль "кольца". */

    MPI_Sendrecv(&A, 1, MPI_INT, dest1, 2, &B, 1, MPI_INT,
sourc1, 2,
```

```
                                                        tor_2D,
&status);
         printf("rank = %d  B=%d\n", rank, B);
         MPI_Sendrecv(&A, 1, MPI_INT, dest2, 2, &B, 1, MPI_INT,
sourc2, 2,
                                                        tor_2D,
&status);
```

/* **Каждая ветвь печатает свой номер (он же и был послан соседней ветви) и значение переменной**

* **В (ранг соседней ветви)** */

```
         printf("rank = %d  B=%d\n", rank, B);
```

/* **Все ветви завершают системные процессы, связанные с топологией top_2D и завершают**

* **выполнение программы** */

```
         MPI_Comm_free(&tor_2D);
         MPI_Finalize();
 return 0;
}
```

Лабораторное задание

Параллельное умножение матрицы на вектор на топологии "кольцо".

Разработать алгоритм написать и отладить параллельную программу умножения матрицы на вектор в топологии "кольцо": A x B = C, при условии, что количество строк матрицы и элементов вектора нацело не делится на количество компьютеров. Например, матрица A размером [22x22] и вектор B размером [22] на четырех компьютерах.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Для чего нужны виртуальные топологии?
2. Как задаются "декартовы" топологии в MPI?
3. Как задаются топологии "граф" в MPI?
4. Какой метод распараллеливания используется в параллельных алгоритмах умножения матрицы на вектор и матрицы на матрицу с использованием MPI?
5. Как распределяются элементы матрицы и вектора при умножении матрицы на вектор на топологии "кольцо"?
6. Как распределяются элементы обоих матриц при умножении матрицы на матрицу на топологии "кольцо"?
7. Как лучше представить в памяти компьютера вторую матрицу, при умножении матрицы на матрицу, для ускорения времени решения задачи?

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

Лабораторная работа № 3. Введение в OpenMP

Цель лабораторной работы: получение представления о OpenMP. **Формируемые компетенции:** ПК-3; ПК-10; ПК-12

Теоретическая часть

Интерфейс OpenMP задуман как стандарт для программирования на масштабируемых SMP-системах (SSMP,ccNUMA, etc.) в модели общей памяти (shared memory model). В стандарт OpenMP входят спецификации набора директив компилятора, процедур и переменных среды. Примерами систем с общей памятью, масштабируемых до большого числа процессоров, могут служить суперкомпьютеры Cray Origin2000 (до 128 процессоров), HP 9000 V-class (до 32 процессоров в одном узле, а в конфигурации из 4 узлов - до 128 процессоров), Sun Starfire (до 64 процессоров).

Общедоступный процесс памяти может состоять из множественных потоков, несколько нитей управления, которые имеют общее адресное пространство, но разные потоки команд и раздельные стэки. В простейшем случае, процесс состоит из одной нити. Нити иногда называют также потоками, легковесными процессами, LWP (light-weight processes).. OpenMP основан на существовании множественных потоков в общедоступной памяти, программирующей парадигму.

OpenMP имеет явную (не автоматический) модель программирования, предлагая программисту полное управление по распараллеливанию.

Модель Fork-Join (Ветвление – Объединение)

OpenMP использует Fork-Join модель параллельного выполнения:

Все программы OpenMP начинаются как единственный процесс: главный поток. Главный поток выполняется последовательно, пока не сталкиваются с первой областью параллельной конструкции.

Fork (ВЕТВЛЕНИЕ): главный поток создает группу параллельных потоков.

Инструкции в программе, которые включены параллельной конструкцией области{*региона*}, тогда выполнены параллельно среди различных потоков группы

Join (ОБЪЕДИНЕНИЕ): Когда потоки группы завершают инструкции в области параллельной конструкции, они синхронизируются и закрываются, оставляя только главный поток..

Лабораторное задание

1. Разработать алгоритм написать и отладить параллельную программу умножения матрицы на вектор с использованием распределения работ для параллельных процессов директивой sections.

2. Разработать алгоритм написать и отладить параллельную программу умножения матрицы на вектор с использованием распределения работ для параллельных процессов с непосредственным заданием работы («вручную»).

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру

страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Методы распараллеливания и модели программ, поддерживаемые OpenMP.

3. Как задаются параллельные процессы при умножении матрицы на вектор и матрицы на матрицу в OpenMP?

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

Лабораторная работа № 4. Решение СЛАУ итерационными методами

Цель лабораторной работы: получение представления о решении СЛАУ итерационными методами.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Решение СЛАУ методом простой итерации

Здесь приведены формулы и последовательные программы данного метода.

Дана система линейных алгебраических уравнений:

$$\begin{aligned} &a_{00}x_0+a_{01}x_1+ \dots +a_{0n}x_n = f_0 \\ &a_{10}x_0+a_{11}x_1+ \dots +a_{1n}x_n = f_1 \qquad (1) \\ &\dots\dots\dots\dots\dots\dots \\ &a_{n0}x_0+a_{n1}x_1+ \dots +a_{nn}x_n = f_n \end{aligned}$$

Краткая запись:

$$Ax = f \qquad (2)$$

Корни этой системы методом простой итерации вычисляются по следующей формуле:

$$x_i^{k+1} = x_i^k - \tau * (\sum_{j=0}^{N} a_{ij}x_j^k - f_i).....i = 0,..., N \qquad (3)$$

Здесь верхний индекс обозначает номер итерационного шага, i - номер корня или правой части. τ – итерационный шаг.

Завершение вычислений по условию:

$$\frac{\left\| \sum_{j=0}^{N} a_{ij}x_{ij}^k - f_i \right\|}{\|f\|} < \varepsilon \qquad (4)$$

Замечание.

$$\|f\| = \sqrt{\sum_{i=0}^{N} (f_i * f_i)} \qquad (5)$$

Числитель выражения (4) вычисляется по аналогичной формуле.

Ниже приведена программа для последовательного алгоритма:

```
#include<stdio.h>
#include<sys/time.h>

#define N 100

double A[N][N], F[N], mf, X[N], X1[N], S[N], msf;

main()
 { int i, j;
   long int dt;
   double t, e;
   struct timeval tv1, tv2;

 /* Генерация данных. Здесь задается матрица с элементами равными 1,
  * по диагонали равными 2. Матрица берется хорошо обусловленной.
  * При правой части равной N+1 все корни будут равными 1. */
```

```
  mf = 0;
  for(i = 0; i < N; i++)
   { for(j = 0; j < N; j++)
     (i==j)?(A[i][j]=2):(A[i][j]=1);
    F[i] = N+1;
 /* Сразу вычисляем сумму квадратов элементов вектора F,
  * т.е. подкоренное выражение формулы (4). */
    mf += F[i]*F[i];
   }

/* Задаем шаг t и e. */
  t = 0.01;
  e = 0.00001;

/* Задаем начальное приближение корней. В X1 хранятся значения корней
 * к+1-й итерации. */
  for(i = 0; i < N; i++)
   X1[i] = 0.6;

/* Засекаем время начала вычислений. */
   gettimeofday(&tv1,NULL);

   do
    { for(i = 0; i < N; i++)
/* В X хранятся значения корней к-й итерации. */
       X[i] = X1[i];

      for(msf=0,i = 0; i < N; i++)
       { for(S[i] = 0,j = 0; j < N; j++)
          S[i] += A[i][j] * X[j];
             X1[i] = X[i] - t*(S[i] - F[i]);
 /* Вычисляем сумму квадратов элементов невязки, т.е. подкоренное
  * выражение числителя формулы (4). */
         msf += (S[i] – F[i])*(S[i] – F[i]);
       }
    }
   while(msf/mf > e*e);   /* Проверка условия по формуле (3). */

/* Засекаем время конца вычислений. */
   gettimeofday(&tv2,NULL);
   dt = (tv2.tv_sec - tv1.tv_sec) * 1000000 + tv2.tv_usec - tv1.tv_usec;

/* Выводим на экран время вычислений в секундах */
   printf(" Time= %d\n",dt);
/* Для контроля выводим 4-е первых корня */
   for(i = 0; i < 4; i++)
    printf(" %f\n",X[i]);
 return(0);
 }
```

Обратите внимание на функцию замера времени gettimeofday(&tv,NULL). Эту функцию можно использовать и для параллельных программ.

Лабораторное задание

Задание 1. Проработка примеров из пункта "Примеры параллельных программ" этой же лабораторной.

Задание 2. Параллельный алгоритм решения СЛАУ методом простой итерации в MPI

Задание 3. Параллельный алгоритм решения СЛАУ методом сопряженных градиентов в MPI

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Какой метод распараллеливания используется в параллельных алгоритмах решения СЛАУ методами простой итерации и сопряженных градиентов в MPI?
2. Как распределяются элементы матрицы коэффициентов и вектора решений при решении СЛАУ методом простой итерации в MPI?
3. Как распределяются элементы матрицы коэффициентов и вектора решений при решении СЛАУ методом сопряженных градиентов в MPI?
4. Как распараллеливаются алгоритмы решения СЛАУ методами простой итерации и сопряженных градиентов в OpenMP?

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

Лабораторная работа № 5. Решение СЛАУ методом Гаусса

Цель лабораторной работы: получение представления о решении СЛАУ методом Гаусса.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Требуется найти решение системы линейных алгебраических уравнений:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ \dots + a_{1n}x_n &= f_1 \\ a_{21}x_2 + a_{22}x_2 + \ \dots + a_{2n}x_n &= f_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots & \\ a_{n1}x_1 + a_{n2}x_2 + \ \dots + a_{nn}x_n &= f_n \end{aligned}$$

Метод Гаусса основан на последовательном исключении неизвестных. Здесь рассматриваются два алгоритма решения СЛАУ методом Гаусса. Они связаны с разными способами представления *данных* (матриц коэффициентов и правых частей) в распределенной памяти мультикомпьютера. Схемы распределения данных по компьютерам для обоих примеров приведены ниже. Хотя данные распределены в памяти мультикомпьютера в каждом алгоритме по разному, но оба они реализуются на одной и той же топологии связи компьютеров - "полный граф".

Лабораторное задание

Задание 1. Параллельный алгоритм № 1 решения СЛАУ методом Гаусса в MPI

Задание 2. Параллельный алгоритм № 2 решения СЛАУ методом Гаусса в MPI

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Какой метод распараллеливания используется в параллельных алгоритмах решения СЛАУ методом Гаусса в MPI?

2. Как распределяются элементы матрицы коэффициентов и вектора правых частей при решении СЛАУ методом Гаусса алгоритмом № 1?

3. Как распределяются элементы матрицы коэффициентов и вектора правых частей при решении СЛАУ методом Гаусса алгоритмом № 2?

4. Какой из методов Гаусса является более быстродействующим?

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL:

Лабораторная работа № 6. Решение задачи Пуассона в 3D методом Зейделя

Цель лабораторной работы: получение представления о решении задачи Пуассона в 3D методом Зейделя.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Решить уравнение Пуассона

$$\frac{\partial^2\varphi}{\partial x}+\frac{\partial^2\varphi}{\partial y}+\frac{\partial^2\varphi}{\partial z}-a\varphi=\rho(x,y,z) \qquad (1)$$

$$\varphi=\varphi(x,y,z), a>0$$

в области Ω с краевыми условиями 1-го рода

$$\varphi|_G=F(x,y,z)$$

Область решения Ω имеет вид прямоугольного параллелепипеда с размерами X_m x Y_m x Z_m.

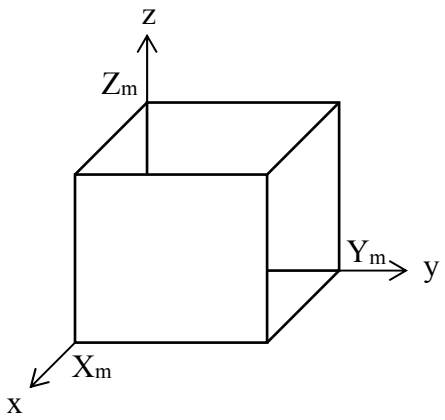


Рис. 6.1. Область решения Ω

В области введена равномерная прямоугольная сетка с шагами:

$$h_x = X_m/I_m, \quad h_y = Y_m/J_m, \quad h_z = Z_m/K_m,$$

где I_m, J_m, K_m - количество ячеек в направлениях x, y, z.

Значения сеточной функции задаются в узлах сетки

$$\varphi_{i,j,k}=\varphi(x_i,y_j,z_k)$$

где $i = 0,\dots,I_m$, $j = 0,\dots,J_m$, $k = 0,\dots,K_m$.

В выбранной сетке апроксимируется уравнение (1):

$$\frac{\varphi_{i+1}-2\varphi_{i,j,k}+\varphi_{i-1}}{h_x^2}+\frac{\varphi_{j+1}-2\varphi_{i,j,k}+\varphi_{j-1}}{h_y^2}+\frac{\varphi_{k+1}-2\varphi_{i,j,k}+\varphi_{k-1}}{h_z^2}-\alpha\varphi=\rho_{i,j,k}$$

i = 1,…,Im-1 , j = 1,…,Im-1 , k = 1,…,Im-1.

Полученная система линейных алгебраических уравнений решается итерационным методом Зейделя:

$$(\frac{2}{h_x^2}+\frac{2}{h_y^2}+\frac{2}{h_z^2}+a)\varphi_{i,j,k}^{m+1}=\frac{\varphi_{i+1}^{m}+\varphi_{i-1}^{m+1}}{h_x^2}+\frac{\varphi_{j+1}^{m}+\varphi_{j-1}^{m+1}}{h_y^2}+\frac{\varphi_{k+1}^{m}+\varphi_{k-1}^{m+1}}{h_z^2}-\rho_{i,j,k}$$

или

$$\varphi_{ijk}^{m+1}=\frac{\frac{\varphi_{i+1jk}^{m}+\varphi_{i-1jk}^{m+1}}{h_x^2}+\frac{\varphi_{ij+1k}^{m}+\varphi_{ij-1k}^{m+1}}{h_y^2}+\frac{\varphi_{ijk+1}^{m}+\varphi_{ijk-1}^{m+1}}{h_z^2}-\rho_{ijk}}{\frac{2}{h_x^2}+\frac{2}{h_y^2}+\frac{2}{h_z^2}+a}$$

Здесь m номер итерации. Напомним, что схема Зейделя неявная.

Входные параметры: Xm, Ym, Zm, Im, Jm, Km , έ, α функция $\rho(x,y,z)$, φ|Ġ. Итерации вести до выполнения условия:

$$\max|\varphi^{m+1}-\varphi^{m}|<\varepsilon$$

Выходные данные:

1) $\max|\varphi^{m+1}-\varphi_{tochnoe}|$

2) функция $\varphi^{m+1}-\varphi_{tochnoe}$

Лабораторное задание

Задание 1. Построение параллельного алгоритма решения задачи Пуассона методом Зейделя в MPI

Задание 2. Запуск параллельного алгоритма решения задачи Пуассона для разных функций

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Какой метод распараллеливания используется для решения задач, использующих в своей реализации сеточные методы?

2. Как можно распределить пространство вычислений 3D на распределенной вычислительной системе?

3. Какой из параллельных алгоритмов решения задачи Пуассона в 3D на ваш взгляд будет более быстродействующим: 1) на линейке компьютеров; 2) на двумерной решетке компьютеров; или 3) на трехмерной решетке компьютеров?

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

Лабораторная работа № 7. Распараллеливание не вычислительной задачи

Цель лабораторной работы: получение представления о распараллеливании не вычислительной задачи.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Решение СЛАУ методом простой итерации

Здесь приведены формулы и последовательные программы данного метода.

Дана система линейных алгебраических уравнений:

$$\begin{aligned} &a_{00}x_0+a_{01}x_1+ \ldots +a_{0n}x_n = f_0 \\ &a_{10}x_0+a_{11}x_1+ \ldots +a_{1n}x_n = f_1 \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n0}x_0+a_{n1}x_1+ \ldots +a_{nn}x_n = f_n \end{aligned} \qquad (1)$$

Краткая запись:

$$Ax = f \qquad (2)$$

Корни этой системы методом простой итерации вычисляются по следующей формуле:

$$x_i^{k+1} = x_i^k - \tau * (\sum_{j=0}^{N} a_{ij}x_j^k - f_i).....i = 0,..., N \qquad (3)$$

Здесь верхний индекс обозначает номер итерационного шага, i - номер корня или правой части. τ - итерационный шаг.

Завершение вычислений по условию:

$$\frac{\left\| \sum_{j=0}^{N} a_{ij}x_{ij}^k - f_i \right\|}{\|f\|} < \varepsilon \qquad (4)$$

Замечание.

$$\|f\| = \sqrt{\sum_{i=0}^{N} (f_i * f_i)} \qquad (5)$$

Числитель выражения (4) вычисляется по аналогичной формуле.

Лабораторное задание

Задание 1. Сортировка множеств. Разработать параллельный алгоритм, написать и отладить параллельную программу сортировки множеств в MPI. (Метод сортировки любой по желанию).

Задание 3. Параллельный алгоритм решения СЛАУ методом сопряженных градиентов локально-оптимальная схема в OpenMP

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см.,

верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Какой метод распараллеливания используется для решения задач, использующих в своей реализации сеточные методы?

2. Как можно распределить пространство вычислений 3D на распределенной вычислительной системе?

3. Какой из параллельных алгоритмов решения задачи Пуассона в 3D на ваш взгляд будет более быстродействующим: 1) на линейке компьютеров; 2) на двумерной решетке компьютеров; или 3) на трехмерной решетке компьютеров?

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

Лабораторная работа № 8. Работа с отладчиком параллельных программ Gepard

Цель лабораторной работы: получение представления о работе с отладчиком параллельных программ Gepard.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Параллельные алгоритмы и программы, как правило, сложнее последовательных. Параллельные программы и отлаживать сложнее. Кроме того, в них возможны и нетипичные для последовательных программ ошибки, вызываемые некорректной синхронизацией процессов или нитей и некорректным использованием средств, обеспечивающих параллелизм. Существенно усложняет отладку параллельных программ их недетерминированное поведение, поскольку оно затрудняет использование привычного приема постепенной локализации ошибок посредством многократной запускова программы под управлением отладчика. Сложность создания хороших средств отладки и анализа параллельных программ является, с одной стороны, следствием проблем разработки параллельных программ, а с другой стороны отладчик параллельных программ представляет собой также параллельную программу, которая должна взаимодействовать с отлаживаемой, также параллельной, что еще больше усложняет проблему.

При диалоговой отладке и анализе параллельных программ особенно актуальна проблема визуализации получаемых результатов. В отличие от простой последовательной программы, где имеется одна текущая точка выполнения программы, а у переменной - одно значение, в параллельной программе точек выполнения может быть много, сотни или даже тысячи. Значений у переменной также может быть много - у каждого процесса своё. Перед создателями средств отладки и анализа параллельных программ встаёт нелегкая задача представить всю имеющуюся информацию и предоставить средства управления ею с одной стороны в удобном, понятном для пользователя виде, а с другой стороны - дать пользователю исчерпывающую, полную картину поведения исследуемой программы.

Лабораторное задание

Задание 1. Просмотреть MPI-программы: параллельного умножения матрицы на вектор и матрицы на матрицу, запущенных на 4-х и на 8-ми процессорах.

Задание 2. Просмотреть MPI-программу решения задачи Пуассона в 3-х мерной области, запущенной на 4-х и на 8-ми процессорах.

Задание 3. Просмотреть MPI-программу сортировки множеств, запущенной на 4-х и на 8-ми процессорах.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Поведенческие ошибки в параллельной программе.

2. Методы обнаружения поведенческих ошибок.

3. Подходы к описанию частичного поведения.

4. Технические требования, предъявляемые к отладчику ПП ориентированному на отладку поведения ПП.

5. Архитектура отладчика.

6. Отладчик GEPARD, ориентированный на отладку программ для мультикомпьютеров.

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

Лабораторная работа № 9. Параллельное программирование в DVM

Цель лабораторной работы: получение представления о параллельном программированим в DVM.

Формируемые компетенции: ПК-3; ПК-10; ПК-12

Теоретическая часть

Язык C-DVM предназначен для разработки мобильных и эффективных параллельных программ вычислительного характера. Он представляет собой расширение языка Си в соответствии с моделью DVM, разработанной в ИПМ им М.В.Келдыша РАН.

Вычислительные программы для последовательных ЭВМ традиционно создавались на языках Фортран 77 и Си. Для многопроцессорных ЭВМ с распределенной памятью и сетей ЭВМ такие программы в настоящее время, как правило, создаются на языках Фортран 77 и Си, расширенных библиотеками передачи сообщений (PVM, MPI). Разработка таких параллельных программ требует от прикладного программиста гораздо больших усилий, чем разработка последовательных программ, поскольку ему требуется не только распределить данные и вычисления между разными процессорами, но и организовать взаимодействие процессоров посредством обмена сообщениями. Фактически, параллельная программа представляет собой систему взаимодействующих программ, каждая из которых выполняется на своем процессоре. Программы на разных процессорах могут быть совершенно различными, могут различаться лишь незначительно, а могут являться одной программой, поведение которой зависит от номера процессора. Но даже в этом последнем случае, программист обычно вынужден разрабатывать и сопровождать два различных варианта своей программы последовательный и параллельный.

При использовании языка C-DVM программист имеет только один вариант программы и для последовательного и для параллельного выполнения. Эта программа, помимо описания алгоритма обычными средствами языка Си, содержит правила параллельного выполнения этого алгоритма. Эти правила оформляются синтаксически таким образом, что они являются “невидимыми” для стандартных компиляторов с языка Си для последовательных ЭВМ и не препятствуют выполнению и отладке DVM-программы на рабочих станциях как обычной последовательной программы.

Таким образом, язык C-DVM представляет собой стандартный язык Си, дополненный средствами спецификации параллельного выполнения программы:

- распределение элементов массива между процессорами;
- распределение витков цикла между процессорами;
- спецификация параллельно выполняющихся секций программы (параллельных задач) и отображение их на процессоры;
- организация эффективного доступа к удаленным (расположенным на других процессорах) данным;
- организация эффективного выполнения редукционных операций – глобальных операций с расположенными на различных процессорах данными (таких, как их суммирование или нахождение их минимального значения).

Для создания программ на языке C-DVM служит специальная система автоматизации разработки параллельных программ – DVM-система.

DVM-система предоставляет единый комплекс средств для разработки параллельных программ научно-технических расчетов на языках Си и Фортран 77.

Лабораторное задание

Задание 1. Умножение матрицы на вектор. Написать программу в C_DVM и в F-DVM. Алгоритм умножения матрицы на вектор есть в лабораторной работе № 1. Программу запустить на 4-х процессорах. Посмотреть разные варианты отображения элементов вектора на элементы матрицы.

Задание 2. Умножение матрицы на матрицу. C = A * B Написать программу в C_DVM и в F-DVM. Программу запустить на 4-х процессорах. Посмотреть разные варианты отображения, например, элементов матриц A и B на элементы матрицы C.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и программными продуктами: пакет MS Office.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине. Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе. В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Поведенческие ошибки в параллельной программе.
2. Методы обнаружения поведенческих ошибок.
3. Подходы к описанию частичного поведения.
4. Технические требования, предъявляемые к отладчику ПП ориентированному на отладку поведения ПП.
5. Архитектура отладчика.
6. Отладчик GEPARD, ориентированный на отладку программ для мультикомпьютеров.

Список литературы, рекомендуемый к использованию по данной теме:

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Биллиг В. А. Параллельные вычисления и многопоточное программирование. ББК: 32.973. Москва: Национальный Открытый Университет «ИНТУИТ», 2016. С- 311. Дополнительная информация: 2-е изд., испр. http://biblioclub.ru/index.php?page=book_red&id=428948&sr=1

2. Николаев Е. И. Параллельные вычисления: учебное пособие УДК: 004.41 (075.8). ББК: 22.18 я73. Ставрополь: СКФУ, 2016 С- 185. URL: http://biblioclub.ru/index.php?page=book_red&id=459124&sr=1

МИНИСТЕРСТВО НАУКИ ИС ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации самостоятельной работы по дисциплине
«Параллельные вычисления / Parallel computing»
для магистрантов направления подготовки 09.04.03 «Прикладная информатика»
(направленность (профиль) «Управление знаниями»)

Ставрополь
2022

Введение

тема MPI является основным средством программирования таких современных высокопроизводительных мультикомпьютеров, как Silicon Graphics Origin 2000, Cray T3D, Cray T3E, IBM SP2 и многих других. MPI работает на самых разных архитектурах мультикомпьютеров, как с распределенной памятью, так и с разделяемой памятью. Кроме того, MPI работает на сетях компьютеров (кластерах) однородных и гетерогенных. Количество компьютеров в сети может быть от одного и более. Важнейшей особенностью MPI является то, что пользователь при написании своих параллельных программ не должен учитывать все эти архитектурные особенности конкретных мультикомпьютеров, поскольку MPI предоставляет пользователю виртуальный мультикомпьютер с распределенной памятью и с виртуальной сетью связи между виртуальными компьютерами. Пользователь заказывает количество компьютеров, необходимых для решения его задачи, и определяет топологию связей между этими компьютерами. MPI реализует этот заказ на конкретной физической системе. Ограничением является объем оперативной памяти физического мультикомпютера. Таким образом пользователь работает в виртуальной среде, что обеспечивает переносимость его параллельных программ. Система MPI представляет собой библиотеку средств параллельного программирования для языков C и Fortran 77.

1. Общая характеристика самостоятельной работы студента при изучении дисциплины Параллельное вычисление– формирование набора профессиональных компетенций будущего магистра по направлению подготовки 09.04.03 «Прикладная информатика».

2. Задачи освоения дисциплины:

- изучение особенностей архитектуры многопроцессорных вычислительных систем;
- изучение основ параллельного программирования на базе вычислительного комплекса;
- формирование умения составления алгоритмов параллельного программирования для решения типовых задач линейной алгебры.

3. Формируемые компетенции - ПК-3; ПК-10; ПК-12

4. План-график выполнения самостоятельной работы

№ п/п	Раздел дисциплины	Неделя семестра	СРС в часах	Формы текущего контроля успеваемости
1	Лабораторная работа № 1. Основные функции MPI	1.	4	Собеседование по вопросам темы
2	Лабораторная работа № 2. Виртуальные топологии.	2.	4	Собеседование по вопросам темы
3	Лабораторная работа № 3. Введение в OpenMP	3.	4	Собеседование по вопросам темы
4	Лабораторная работа № 4. Решение СЛАУ итерационными методами	4.	4	Собеседование по вопросам темы
5	Лабораторная работа № 5. Решение СЛАУ методом Гаусса	5.	4,5	Собеседование по вопросам темы
6	Лабораторная работа № 6. Решение задачи Пуассона в 3D методом Зейделя	6.	5	Собеседование по вопросам темы
7	Лабораторная работа № 7. Распараллеливание не вычислительной задачи	7.	5	Собеседование по вопросам темы
8	Лабораторная работа № 8. Работа с отладчиком параллельных программ Gepard	8.	5	Собеседование по вопросам темы
9	Лабораторная работа № 9. Параллельное программирование в DVM	9.	5	Собеседование по вопросам темы
		ИТОГО	40,5	

5. Методические рекомендации по изучению теоретического материала

На первом этапе необходимо ознакомиться с рабочей программой дисциплины, в которой рассмотрено содержание тем дисциплины лекционного курса, взаимосвязь тем лекций с лабораторными занятиями, темы и виды самостоятельной работы. По каждому виду самостоятельной работы предусмотрены определённые формы отчетности.

Примерный перечень вопросов для собеседования

Лабораторная работа № 1. Основные функции MPI

1. Методы распараллеливания и модели программ, поддерживаемые MPI.
2. Свойства MPI, обеспечивающие переносимость параллельных программ и независимость от вычислительных систем.
3. Пояснить основные принципы выполнения ниже указанных функций.
4. Команда компиляции MPI–программ.
5. Команда запуска MPI–программ.
6. Действия системы по команде mpirun .

Лабораторная работа № 2. Виртуальные топологии.

1. Для чего нужны виртуальные топологии?
2. Как задаются "декартовы" топологии в MPI?
3. Как задаются топологии "граф" в MPI?
4. Какой метод распараллеливания используется в параллельных алгоритмах умножения матрицы на вектор и матрицы на матрицу с использованием MPI?
5. Как распределяются элементы матрицы и вектора при умножении матрицы на вектор на топологии "кольцо"?
6. Как распределяются элементы обоих матриц при умножении матрицы на матрицу на топологии "кольцо"?
7. Как лучше представить в памяти компьютера вторую матрицу, при умножении матрицы на матрицу, для ускорения времени решения задачи?

Лабораторная работа № 3. Введение в OpenMP

1. Методы распараллеливания и модели программ, поддерживаемые OpenMP.
3. Как задаются параллельные процессы при умножении матрицы на вектор и матрицы на матрицу в OpenMP?

Лабораторная работа № 4. Решение СЛАУ итерационными методами

1. Какой метод распараллеливания используется в параллельных алгоритмах решения СЛАУ методами простой итерации и сопряженных градиентов в MPI?
2. Как распределяются элементы матрицы коэффициентов и вектора решений при решении СЛАУ методом простой итерации в MPI?
3. Как распределяются элементы матрицы коэффициентов и вектора решений при решении СЛАУ методом сопряженных градиентов в MPI?
4. Как распараллеливаются алгоритмы решения СЛАУ методами простой итерации и сопряженных градиентов в OpenMP?

Лабораторная работа № 5. Решение СЛАУ методом Гаусса

1. Какой метод распараллеливания используется в параллельных алгоритмах решения СЛАУ методом Гаусса в MPI?

2. Как распределяются элементы матрицы коэффициентов и вектора правых частей при решении СЛАУ методом Гаусса алгоритмом № 1?

3. Как распределяются элементы матрицы коэффициентов и вектора правых частей при решении СЛАУ методом Гаусса алгоритмом № 2?

4. Какой из методов Гаусса является более быстродействующим?

Лабораторная работа № 6. Решение задачи Пуассона в 3D методом Зейделя

1. Какой метод распараллеливания используется для решения задач, использующих в своей реализации сеточные методы?

2. Как можно распределить пространство вычислений 3D на распределенной вычислительной системе?

3. Какой из параллельных алгоритмов решения задачи Пуассона в 3D на ваш взгляд будет более быстродействующим: 1) на линейке компьютеров; 2) на двумерной решетке компьютеров; или 3) на трехмерной решетке компьютеров?

Лабораторная работа № 7. Распараллеливание не вычислительной задачи

1. Какой метод распараллеливания используется для решения задач, использующих в своей реализации сеточные методы?

2. Как можно распределить пространство вычислений 3D на распределенной вычислительной системе?

3. Какой из параллельных алгоритмов решения задачи Пуассона в 3D на ваш взгляд будет более быстродействующим: 1) на линейке компьютеров; 2) на двумерной решетке компьютеров; или 3) на трехмерной решетке компьютеров?

Лабораторная работа № 8. Работа с отладчиком параллельных программ Gepard

1. Поведенческие ошибки в параллельной программе.

2. Методы обнаружения поведенческих ошибок.

3. Подходы к описанию частичного поведения.

4. Технические требования, предъявляемые к отладчику ПП ориентированному на отладку поведения ПП.

5. Архитектура отладчика.

6. Отладчик GEPARD, ориентированный на отладку программ для мультикомпьютеров.

Лабораторная работа № 9. Параллельное программирование в DVM

1. Поведенческие ошибки в параллельной программе.

2. Методы обнаружения поведенческих ошибок.

3. Подходы к описанию частичного поведения.

4. Технические требования, предъявляемые к отладчику ПП ориентированному на отладку поведения ПП.

5. Архитектура отладчика.

6. Отладчик GEPARD, ориентированный на отладку программ для мультикомпьютеров.

6.Методические указания по подготовке к экзамену

Перед экзаменом студенту необходимо полностью выполнить все задания к лабораторным работам. При наличии задолженностей по текущей аттестации по данной дисциплине студент к экзамену не допускается. Экзамен по дисциплине предусмотрен в устной форме по билетам.

Вопросы для проверки уровня обученности

Базовый уровень. Вопросы для проверки знаний:

1. Системы с распределенной памятью. Взаимодействующие процессы.
2. Маршрутизация и управление потоком. Выделение процессов.
3. Реагирующие системы. Системы с бесконечными каналами.
4. Вычисления в анонимных системах. Вычисление булевых функций.
5. Алгоритмы распространения информации.
6. Проверка связности графа. Алгоритм вычисления кратчайших расстояний.
7. Рассмотрение графа "подзадачи – сообщения".
8. Канал передачи данных в параллельных системах.
9. Основные показатели успешности распределения подзадач между процессорами.
10. Распределение подзадач между процессорами.
11. Статическая схема передачи данных.
12. Способы разделения вычислений на независимые части.
13. Необходимые условия для возможности организации параллельных вычислений.
14. Определение и классификация кластера.
15. Классификации Флинна для многопроцессорных вычислительных систем.
16. Определение суперкомпьютерных систем.

Продвинутый уровень. Вопросы для проверки знаний:

1. Классы параллельных задач поддерживает в системах имитационного моделирования.
2. Определений расписания параллельных вычислений.
3. Графы информационных зависимостей в параллельных программах.
4. Задача суммирования последовательности чисел.
5. Определение модели вычислений.
6. Зависимости времен выполнения программ от количества процессоров.
7. Методы отладки распределенных программ.

7.Методическиематериалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения **экзамена** осуществляется в соответствии с Положением о проведении текущего контроля успеваемости и промежуточной аттестации обучающихся по образовательным программам высшего образования в СКФУ.

В экзаменационный билет включаются три вопроса.

Для подготовки по билету отводиться 45 минут.

При подготовке к ответу студенту предоставляется право пользования письменными отчетами по выполненным лабораторным работам.

При проверке практического задания, оцениваются: практическое задание не предусмотрено.

Текущая аттестация студентов проводится преподавателями, ведущими лабораторные занятия по дисциплине в следующих формах: письменный отчет и собеседование.

Допуск к лабораторным работам происходит при наличии у студентов печатного варианта отчета. Защита отчета проходит в форме доклада студента по выполненной работе и ответов на вопросы преподавателя.

Максимальное количество баллов студент получает, если оформление отчета соответствует установленным требованиям, а отчет полностью раскрывает суть работы. Основанием для снижения оценки являются:

- Сдача лабораторной работы не в срок;
- Неполное выполнение.

Отчет может быть отправлен на доработку в следующих случаях:

- Неполное выполнение;
- Слабый уровень представления результатов;
- Не надлежащее оформление.

Критерии оценивания самостоятельного изучения литературы приведены в Фонде оценочных средств по дисциплине "Параллельное вычисление"