

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»**

**МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ
ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ**

Б1.В.17 Параллельное программирование

Направление подготовки *09.03.01 Информатика и вычислительная техника*

Профиль образовательной программы *Автоматизированные системы обработки информации и управления*

Форма обучения *заочная*

СОДЕРЖАНИЕ

1.	Конспект лекций
1.1.	Лекция № 1 <i>Цели, задачи и проблемы параллельных вычислений</i>
1.2.	Лекция № 2 <i>Моделирование и анализ параллельных алгоритмов</i>
1.3.	Лекция № 3 <i>Принципы разработки параллельных алгоритмов и программ</i>
2.	Методические материалы по проведению практических занятий.....
2.1.	Лабораторная работа № ЛР-1 <i>Архитектура высокопроизводительных ЭВМ</i>
2.2.	Лабораторная работа № ЛР-2, 3 <i>Моделирование и анализ параллельных алгоритмов</i>
2.3.	Лабораторная работа № ЛР-4, 5 <i>Принципы разработки параллельных алгоритмов и программ</i>
2.4.	Лабораторная работа № ЛР- 6 <i>Средства разработки параллельных программ</i>
2.5.	Лабораторная работа № ЛР-7 <i>Технология программирования OpenMP</i>
2.6.	Лабораторная работа № ЛР- 8 <i>DVM система разработки параллельных программ</i>

1. КОНСПЕКТ ЛЕКЦИЙ

1.1. Лекция № 1 (2 часа)

Тема: «Цели, задачи и проблемы параллельных вычислений»

1.1.1. Вопросы лекции:

Ограничения максимальной производительности однопроцессорных ЭВМ. Постоянное возникновение задач, потребности которых превышают возможности однопроцессорных ЭВМ. Параллельные и распределенные вычисления и их техническая основа – вычислительные кластеры, ГРИД-системы и суперкомпьютеры. История развития параллельных вычислительных систем, виды параллелизма. Основные проблемы использования параллельной обработки данных.

1.1.2. Краткое содержание вопросов:

Наиболее распространенной технологией программирования для кластерных систем и параллельных компьютеров с распределенной памятью в настоящее время является технология MPI. Основным способом взаимодействия параллельных процессов в таких системах является передача сообщений друг другу. Это и отражено в названии данной технологии – Message Passing Interface (интерфейс передачи сообщений). Стандарт MPI фиксирует интерфейс, который должен соблюдаться как системой программирования на каждой вычислительной платформе, так и пользователем при создании своих программ. MPI поддерживает работу с языками Фортран и Си. Полная версия интерфейса содержит описание более 125 процедур и функций.

Интерфейс MPI поддерживает создание параллельных программ в стиле MIMD (Multiple Instruction Multiple Data), что подразумевает объединение процессов с различными исходными текстами. Однако писать и отлаживать такие программы очень сложно, поэтому на практике программисты, гораздо чаще используют SPMD-модель (Single Program Multiple Data) параллельного программирования, в рамках которой для всех параллельных процессов используется один и тот же код. В настоящее время все больше и больше реализаций MPI поддерживают работу с так называемыми "нитеями".

Поскольку MPI является библиотекой, то при компиляции программы необходимо прилинковать соответствующие библиотечные модули.

После получения исполнимого файла необходимо запустить его на требуемом количестве процессоров. После запуска одна и та же программа будет выполняться всеми запущенными процессами, результат выполнения в зависимости от системы будет выдаваться на терминал или записываться в файл.

1.2. Лекция № 2 (2 часа)

Тема: «Моделирование и анализ параллельных алгоритмов»

1.2.1. Вопросы лекции:

Модели параллельных вычислительных процессов. Концепция неограниченного параллелизма. Модели многопроцессорных систем с общей и распределенной памятью. Модель конвейерной системы. Модель алгоритма в виде графа "операнд - операции". Представление алгоритма в виде графа потока данных. Расписание параллельных вычислений. Показатель временной сложности алгоритма. Оценка времени выполнения алгоритма для паракомпьютера (предельное распараллеливание) и для систем с конечным количеством процессоров. Зависимость оценок от топологии графа алгоритма и необходимость оптимизации структуры графа. Способы получения оптимального расписания вычислений.

1.2.2. Краткое содержание вопросов:

При разработке параллельных алгоритмов решения задач вычислительной математики принципиальным моментом является анализ эффективности использования

параллелизма, состоящий обычно в оценке получаемого *ускорения* процесса вычисления (сокращения времени решения задачи). Формирование подобных оценок ускорения может осуществляться применительно к выбранному вычислительному алгоритму (*оценка эффективности распараллеливания конкретного алгоритма*). Другой важный подход может состоять в построении оценок максимально возможного ускорения процесса получения решения задачи конкретного типа (*оценка эффективности параллельного способа решения задачи*).

В данном разделе описывается модель вычислений в виде графа "операции-операнды", которая может использоваться для описания существующих информационных зависимостей в выбираемых алгоритмах решения задач, приводятся оценки эффективности максимально возможного параллелизма, которые могут быть получены в результате анализа имеющихся моделей вычислений. Примеры использования излагаемого теоретического материала приводятся в 4 разделе пособия.

1. Модель вычислений в виде графа "операции-операнды"

Для описания существующих информационных зависимостей в выбираемых алгоритмах решения задач может быть использована модель в виде графа "операции-операнды" [18] (дополнительная информация по моделированию параллельных вычислений может быть получена в [3, 16, 23, 26, 28]). Для уменьшения сложности излагаемого материала при построении модели будет предполагаться, что время выполнения любых вычислительных операций является одинаковым и равняется 1 (в тех или иных единицах измерения); кроме того, принимается, что передача данных между вычислительными устройствами выполняется мгновенно без каких-либо затрат времени (что может быть справедливо, например, при наличии общей разделяемой памяти в параллельной вычислительной системе). Анализ коммуникационной трудоемкости параллельных алгоритмов выполняется в 3 разделе пособия.

Представим множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного графа

$$G = (V, R),$$

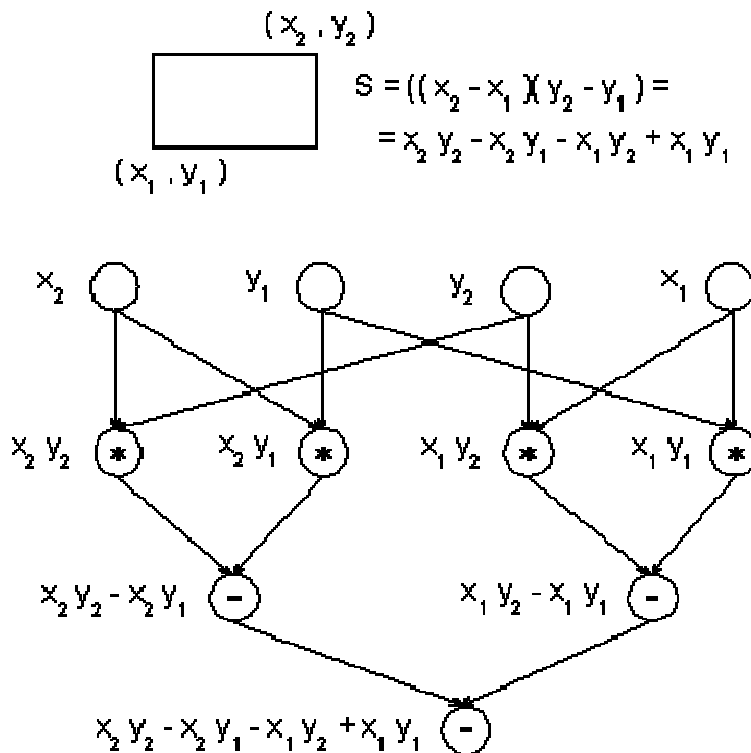


Рис. 2.1. Пример вычислительной модели алгоритма в виде графа "операции-операнды"

где $V = \{1, \dots, |V|\}$ есть множество вершин графа, представляющее выполняемые операции алгоритма, а R есть множество дуг графа (при этом дуга $r = (i, j)$ принадлежит графу только, если операция j использует результат выполнения операции i). Для примера на рис. 2.1 показан граф алгоритма вычисления площади прямоугольника, заданного координатами двух углов. Как можно заметить по приведенному примеру, для выполнения выбранного алгоритма решения задачи могут быть использованы разные схемы вычислений и построены соответственно разные вычислительные модели. Как будет показано далее, разные схемы вычислений обладают разными возможностями для распараллеливания и, тем самым, при построении модели вычислений может быть поставлена задача выбора наиболее подходящей для параллельного исполнения вычислительной схемы алгоритма.

1.3. Лекция № 3 (2 часа)

Тема: «Принципы разработки параллельных алгоритмов и программ».

1.3.1. Вопросы лекции:

Использование распространенных языков программирования и коммуникационных библиотек и интерфейсов. Традиционные последовательные языки и распараллеливающие компиляторы, проблема выделения потенциального параллелизма последовательных программ. Специальные комментарии и директивы компилятору.

1.3.2. Краткое содержание вопросов:

В настоящее время все чаще возникает потребность в решении сложных вычислительных задач, многие из которых допускают возможность параллельных вычислений. В качестве аппаратных средств для решения таких задач используются однопроцессорные системы, в основе которых находятся механизмы временного разделения обработки параллельных потоков (квазипараллелизм) или многопроцессорные системы.

До последнего времени, в связи с высокой стоимостью аппаратной части компьютеров, возможность работать с многопроцессорными системами была доступна

лишь ограниченному числу разработчиков. Несколько лет назад, с появлением многоядерных процессоров появилась возможность обеспечить реальный параллелизм на базе одного процессора, что вызвало огромный интерес к этим задачам широких кругов разработчиков программного обеспечения (ПО). Многоядерные процессоры, распределяя работу и задачи между независимыми ядрами, обеспечивают более высокую производительность при меньшем энергопотреблении. С точки зрения разработчика ПО, для реализации параллельных алгоритмов вычислений необходимо выполнить декомпозицию поставленной задачи на отдельные процессы (потoki). Такая процедура получила название распараллеливания алгоритмов.

Определенные проблемы разработки параллельных приложений связаны с тем, что стандартные средства разработки ПО, как правило, не поддерживают программирование параллельных алгоритмов. Это вызывает необходимость использования специализированных пакетов или библиотек, обеспечивающих поддержку применяемых приложений. Такие средства разрабатывались, как правило, в рамках академических коллективов, и преимущественно относятся к некоммерческому, свободно распространяемому ПО. Вместе с тем, наряду со свободно распространяемыми пакетами или библиотеками существуют и коммерческие средства разработки параллельных приложений.

2. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ПРОВЕДЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

2.1. Лабораторная работа № 1 (2 часа).

Тема: «Архитектура высокопроизводительных ЭВМ».

2.1.1. Вопросы к занятию:

Классификация многопроцессорных вычислительных систем. СуперЭВМ. Многопроцессорные вычислительные комплексы (МВС). Многомашинные вычислительные комплексы. Сети ЭВМ. Систематика Флинна. Потоки данных (команд). Конкретизация видов многопроцессорных систем: векторные компьютеры, симметричные мультимикропроцессоры (SMP), массивно-параллельные компьютерные системы (MPP), кластеры.

2.1.2. Краткое описание проводимого занятия:

2.1.2.1. Ответы на вопросы лабораторной работы.

2.1.2.2. Проведение текущего контроля успеваемости

Задания для проведения текущего контроля успеваемости

1. Типовые топологии схем коммутации.
2. Аппаратная реализация и программная эмуляция топологий.
3. Классификация многопроцессорных вычислительных систем.
4. Систематика Флинна. Потоки данных (команд).
5. Виды многопроцессорных систем и кластеров.
6. Модели параллельных вычислительных процессов.
7. Конвейерные и векторные вычисления. Процессорные матрицы.

Многопроцессорные вычислительные системы с общей и распределенной памятью.

8. Концепция неограниченного параллелизма. Компьютер с неограниченным параллелизмом (паракомпьютер).

9. Модели многопроцессорных систем с общей и распределенной памятью. Модель конвейерной системы.

1. Микроядро обеспечивает планирование и диспетчеризацию задач, а также осуществляет их:

- +а) взаимодействие
- b) независимость друг от друга
- c) условную зависимость
- d) разграничение

2. Концепция многозадачности (псевдопараллелизм) является существенной для системы реального времени с:

- +а) одним процессором
- b) двумя процессорами
- c) тремя процессорами
- d) несколькими процессорами

3. Потоки существуют в одном контексте процесса, поэтому переключение между потоками происходит:

- +а) очень быстро
- b) очень медленно
- c) быстро
- d) медленно

2.2. Лабораторная работа № 2, 3 (4 часа).

Тема: «Моделирование и анализ параллельных алгоритмов».

2.2.1. Вопросы к занятию:

Модель параллельных вычислений в виде сети Петри. Основные понятия теории сетей Петри. Использование сетей Петри для описания параллельных вычислений. Основные проблемы параллельных вычислений: синхронизация, взаимоисключение, блокировка (тупики). Модель параллельных вычислений в виде графа "процесс-ресурс". Понятие процесса. Проблемы взаимодействия процессов. Синхронизация параллельных процессов. Аппарат событий. Концепция ресурса. Механизмы взаимоисключения: алгоритм Деккера, семафоры (Дейкстра), мониторы (Вирт). Взаимодействие параллельных процессов посредством механизма передачи сообщений. Механизмы передачи: очереди, почтовые ящики, порты. Понятие тупика и условия его возникновения. Предотвращение тупиков. Алгоритм банкира. Обнаружение тупиков и восстановление состояния процессов.

2.2.2. Краткое описание проводимого занятия:

2.2.2.1. Ответы на вопросы лабораторной работы.

2.2.2.2. Проведение текущего контроля успеваемости

Задания для проведения текущего контроля успеваемости

1. Модель параллельных вычислений в виде сети Петри.
2. Основные проблемы параллельных вычислений: синхронизация, взаимоисключение, блокировка (тупики).
3. Поточковая модель параллельных вычислений.
4. Проблемы взаимодействия процессов. Синхронизация параллельных процессов.
5. Механизмы взаимоисключения: алгоритм Деккера, семафоры (Дейкстра), мониторы (Вирт).
6. Взаимодействие параллельных процессов посредством механизма передачи сообщений. Механизмы передачи.
7. Понятие тупика и условия его возникновения. Предотвращение тупиков. Обнаружение тупиков и восстановление состояния процессов.
 1. Статические алгоритмы планирования используются для:
 - +а) формального доказательства условий предсказуемости системы
 - b) выполнения условий предсказуемости системы
 - c) планирования условий предсказуемости системы
 - d) получения условий предсказуемости системы
 2. Процессы выполняются в соответствии с:
 - +а) приоритетами
 - b) условиями
 - c) задачами
 - d) возможностями
 3. В динамических алгоритмах планирования приоритет присваивается:
 - +а) динамически
 - b) статически
 - c) независимо
 - d) в зависимости от приоритета

2.3. Лабораторная работа № 4, 5 (4 часа).

Тема: «Принципы разработки параллельных алгоритмов и программ».

2.3.1. Вопросы к занятию:

Уровни распараллеливания вычислений. Распараллеливание вычислений на уровне команд, выражений, программных модулей, отдельно выполняемых заданий. Этапы построения параллельных алгоритмов и программ. Выбор параллельного алгоритма. Реализация алгоритма в виде параллельной программы. Построение исполняемой программы для параллельной вычислительной системы. Параллельное исполнение машинной программы. Частные постановки: выбор оптимального алгоритма для конкретной вычислительной системы, нахождение наилучшей топологии вычислительной системы для решения определенной задачи, распараллеливание существующего алгоритма.

2.3.2. Краткое описание проводимого занятия:

2.3.2.1. Ответы на вопросы лабораторной работы.

2.3.2.2. Проведение текущего контроля успеваемости

Задания для проведения текущего контроля успеваемости

1. Модель алгоритма в виде графа "операнд - операции". Представление алгоритма в виде графа потока данных.

2. Расписание параллельных вычислений. Показатель временной сложности алгоритма.

3. Оценка времени выполнения алгоритма для паракомпьютера (предельное распараллеливание) и для систем с конечным количеством процессоров.

4. Способы получения оптимального расписания вычислений.

5. Параллелизм данных и параллелизм задач. Показатель эффективности распараллеливания (ускорение).

6. Эффективность использования вычислительной системы. Способы оценки показателей.

7. Оценка коммуникационной трудоемкости параллельных алгоритмов. Характеристики топологий сети передачи данных.

8. Алгоритмы маршрутизации. Методы передачи данных. Анализ трудоемкости основных операций передачи данных.

9. Передача данных между двумя процессорами сети.

10. Одиночная и множественная рассылка сообщений. Операция циклического сдвига.

11. Уровни распараллеливания вычислений: команд, выражений, программных модулей, отдельно выполняемых заданий.

1. ...это способ построения программ, концептуальной основой которых являются соответствующая система понятий, стиль мышления и подход к решению задач:

- a) способ программирования
- b) механизм программирования
- c) формализация программирования

+ d) стиль программирования

e) концептуализация программирования

2. ...программирования заключается в том, что алгоритм любой исходной задачи представляется как композиция алгоритмов простых подзадач, последовательно выделенных из исходной задачи:

- + a) модульный стиль
- b) структурный стиль
- c) композиционный стиль
- d) алгоритмический стиль
- e) формализованный стиль

3. При разработке программного модуля целесообразно придерживаться

следующего порядка:

- 1 а) изучение и проверка спецификации модуля, выбор языка программирования
- 2 б) выбор алгоритма и структуры данных
- 6 с) компиляция модуля
- 4 д) шлифовка текста модуля
- 5 е) проверка модуля
- 3 ф) программирование модуля

2.4. Лабораторная работа № 6 (2 часа).

Тема: «Средства разработки параллельных программ».

2.4.1. Вопросы к занятию:

Параллельные языки программирования и расширения стандартных языков. Средства автоматического распараллеливания, параллельные компиляторы. Параллельные предметные библиотеки. Инструментальные системы для проектирования параллельных программ.

2.4.2. Краткое описание проводимого занятия:

2.4.2.1. Ответы на вопросы лабораторной работы.

2.4.2.2. Проведение текущего контроля успеваемости

Задания для проведения текущего контроля успеваемости

1. Этапы построения параллельных алгоритмов и программ.
2. Реализация алгоритма в виде параллельной программы. Построение исполняемой программы для параллельной вычислительной системы.
3. Использование распространенных языков программирования и коммуникационных библиотек и интерфейсов.
4. Распараллеливающие компиляторы, проблема выделения потенциального параллелизма последовательных программ. Специальные комментарии и директивы компилятору.
5. Параллельные языки программирования и расширения стандартных языков.
6. Средства автоматического распараллеливания, параллельные компиляторы.
7. Параллельные предметные библиотеки. Инструментальные системы для проектирования параллельных программ.

2.5. Лабораторная работа № 7 (2 часа).

Тема: «Технология программирования OpenMP».

2.5.1. Вопросы к занятию:

Параллельные циклы. Директивы синхронизации. Классы переменных. Спецификации OpenMP для языков C и C++.

2.5.2. Краткое описание проводимого занятия:

2.5.2.1. Ответы на вопросы лабораторной работы.

2.5.2.2. Проведение текущего контроля успеваемости

Задания для проведения текущего контроля успеваемости

1. Опция `schedule` управляет распределением работы между нитями в конструкции распределения работы цикла.
2. Директива `parallel` создает параллельную область для следующего за ней структурированного блока.
3. Директива `for` для распределения итераций цикла между различными нитями.
4. Если в параллельной области какой-либо участок кода должен быть выполнен лишь один раз, то его нужно выделить директивой `single`.

5. Директива `sections` используется для задания конечного (неитеративного) параллелизма.
6. Директивы `master` выделяют участок кода, который будет выполнен только нитью-мастером.
7. С помощью директив `critical` оформляется критическая секция программы.
8. Директива `barrier` дает всем потокам указание ожидать друг друга перед тем, как они продолжат выполнение за барьером.
9. Директивы `ordered` определяют блок внутри тела цикла, который должен выполняться в том порядке, в котором итерации идут в последовательном цикле.

2.6. Лабораторная работа № 8 (2 часа).

Тема: «DVM система разработки параллельных программ».

2.6.1. Вопросы к занятию:

Использование отладчика и анализатора производительности DVM-программ.
Трудности и перспективы развития МВС и параллельного программирования.

2.6.2. Краткое описание проводимого занятия:

2.6.2.1. Ответы на вопросы лабораторной работы.

2.6.2.2. Проведение текущего контроля успеваемости

Задания для проведения текущего контроля успеваемости

1. Основные возможности системы DVM. Мобильность и эффективность выполнения программ.
2. Директивы распараллеливания DVM-системы. Использование отладчика и анализатора производительности DVM-программ.
3. Перспективы развития МВС и параллельного программирования.