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 *очная*

СОДЕРЖАНИЕ

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 – вычисли-тельные 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-моделъ (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 "нитями".

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

После получения выполнимого 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 час)

Тема: «Архитектура высокопроизводительных ЭВМ»

1.2.1. Вопросы лекции:

Конвейерные и векторные вычисления. Процессорные матрицы. Многопроцессорные вычислительные системы с общей и распределенной памятью (мультипроцессоры и мультикомпьютеры). Схемы коммутации (полная коммутация - общая память, перекрестные коммутаторы, локальные схемы коммутации - общая шина, решетки, кластеры). Анализ возможных схем взаимодействия ветвей параллельных алгоритмов и типовые топологии схем коммутации – кольцо, линейка, решетки, полный граф, гиперкуб, тор, дерево. Аппаратная реализация и программная эмуляция топологий.

1.2.2. Краткое содержание вопросов:

Персональные компьютеры (ПК) появились в результате эволюции миникомпьютеров при переходе элементной базы машин с малой и средней степенью интеграции на большие и сверхбольшие интегральные схемы. ПК, благодаря своей низкой стоимости, очень быстро завоевали хорошие позиции на компьютерном рынке и создали предпосылки для разработки новых программных средств, ориентированных на конечного

пользователя. Это прежде всего - "дружественные пользовательские интерфейсы", а также проблемно-ориентированные среды и инструментальные средства для автоматизации разработки прикладных программ.

Миникомпьютеры стали прародителями и другого направления развития современных систем - 32-разрядных машин. Создание RISC-процессоров и микросхем памяти емкостью более 1 Мбит привело к окончательному оформлению настольных систем высокой производительности, которые сегодня известны как рабочие станции. Первоначальная ориентация рабочих станций на профессиональных пользователей (в отличие от ПК, которые в начале ориентировались на самого широкого потребителя непрофессионала) привела к тому, что рабочие станции - это хорошо сбалансированные системы, в которых высокое быстродействие сочетается с большим объемом оперативной и внешней памяти, высокопроизводительными внутренними магистралями, высококачественной и быстродействующей графической подсистемой и разнообразными устройствами ввода/вывода. Это свойство выгодно отличает рабочие станции среднего и высокого класса от ПК и сегодня. Даже наиболее мощные IBM PC совместимые ПК не в состоянии удовлетворить возрастающие потребности систем обработки из-за наличия в их архитектуре ряда "узких мест".

Тем не менее быстрый рост производительности ПК на базе новейших микропроцессоров Intel в сочетании с резким снижением цен на эти изделия и развитием технологии локальных шин (VESA и PCI), позволяющей устранить многие "узкие места" в архитектуре ПК, делают современные персональные компьютеры весьма привлекательной альтернативой рабочим станциям. В свою очередь производители рабочих станций создали изделия так называемого "начального уровня", которые по стоимостным характеристикам близки к высокопроизводительным ПК, но все еще сохраняют лидерство по производительности и возможностям наращивания. Насколько успешно удаться ПК на базе процессоров 486 и Pentium бороться против рабочих станций UNIX, покажет будущее, но уже в настоящее время появилось понятие "персональной рабочей станции", которое объединяет оба направления.

1.3. Лекция № 3 (2 часа)

Тема: «Моделирование и анализ параллельных алгоритмов»

1.3.1. 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.

1.3.2. 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

$G = (V, R)$,

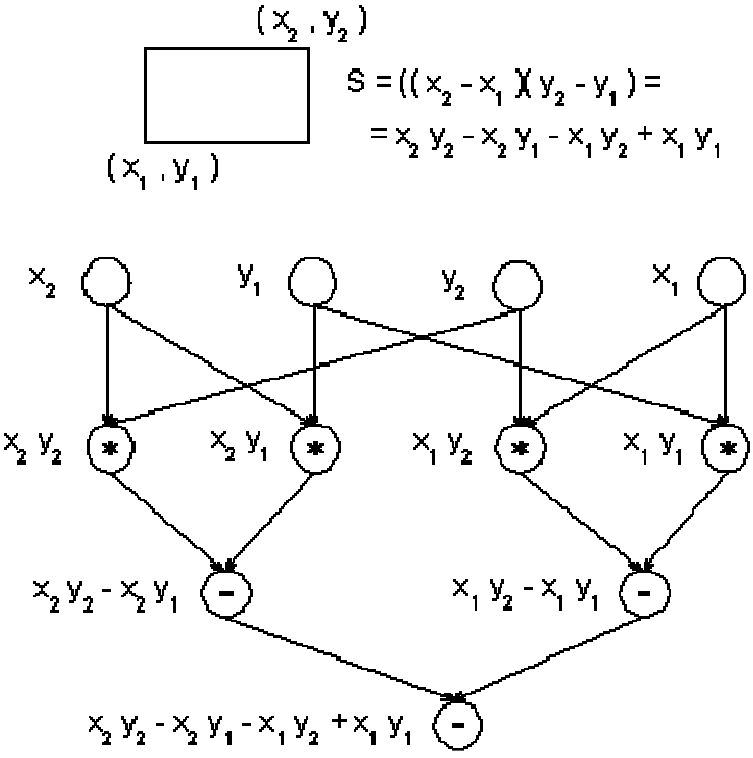

Рис. 2.1. Пример вычислительной модели алгоритма в виде графа "операции-операнды"

где $V = \{1, \ldots, |V|\}$ 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.

1.4. Лекция № 5 (2 часа)

Тема: «Средства разработки параллельных программ».

1.4.1. 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.

1.4.2. 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 (потоки). 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.

1.5. Лекция № 6 (2 часа)

Тема: «Интерфейс передачи сообщений MPI»

1.5.1. Вопросы лекции:

Общие принципы построения и реализации MPI. Разработчики, история создания. Шесть общих функций MPI, коммуникаторы. Функции обмена сообщениями типа «точка-точка»: блокирующий и неблокирующий обмен, синхронные и стандартные посылки сообщений. Предотвращение тупиков.

1.5.2. Краткое содержание вопросов:

MPI – это один из наиболее развитых систем передачи сообщений высокой эффективности, переносимости и надежности параллельных программ.

Цель данной главы – познакомить читателя с основными функциями MPI и привести примеры их использования. Всего в библиотеке MPI около 200 функций, но для того, чтобы создавать сравнительно эффективные параллельные программы, необязательно знать все функции MPI.

В основу главы были положены материалы учебного пособия [30] и MPI - Complete Reference [60]. Все приведенные примеры параллельных программ написаны для компилятора Фортран 77. Программы для компилятора Си будут отличаться лишь синтаксисом самого языка, при этом логика создания параллельных программ останется прежней.

MPI-программа представляет собой набор независимых процессов, каждый из которых выполняет свою собственную программу (не обязательно одну и ту же), написанную на языке Си или FORTRAN. Появились реализации MPI для C++, однако разработчики стандарта MPI за них ответственности не несут. Процессы MPI-программы взаимодействуют друг с другом посредством вызова коммуникационных процедур. Как правило, каждый процесс выполняется в своем собственном адресном пространстве, однако допускается и режим разделения памяти. MPI не специфицирует модель выполнения процесса - это может быть как последовательный процесс, так и многопоточный. MPI не предоставляет никаких средств для распределения процессов по вычислительным узлам. В предыдущей главе описывается технология распределения исполняемого кода параллельного приложения с использованием сервиса NFS операционной системы Redhat Linux. Описываемый в данном методическом пособии стандарт MPI 1.1 не содержит механизмов динамического создания и уничтожения процессов во время выполнения программы. MPI не накладывает каких-либо ограничений на то, как процессы будут распределены по процессорам, в частности, возможен запуск MPI программы с несколькими процессами на обычной однопроцессорной системе.

Для идентификации наборов процессов вводится понятие группы, объединяющей все или какую-то часть процессов. Каждая группа образует область связи, с которой связывается специальный объект - коммуникатор области связи. Процессы внутри группы нумеруются целым числом в диапазоне 0..groupsize-1. Все коммуникационные операции с некоторым коммуникатором будут выполняться только внутри области связи, описываемой этим коммуникатором. При инициализации MPI создается предопределенная область связи, содержащая все процессы MPI-программы, с которой связывается предопределенный коммуникатор MPI_COMM_WORLD. В большинстве случаев на каждом процессоре запускается один отдельный процесс, и тогда термины процесс и процессор становятся синонимами, а величина groupsize становится равной NPROCS - числу процессоров, выделенных задаче. В дальнейшем обсуждении мы будем понимать именно такую ситуацию и не будем очень уж строго следить за терминологией.

1.6. Лекция № 7 (2 часа)

Тема: «Технология программирования OpenMP».

1.6.1. Вопросы лекции:

Последовательные и параллельные нити программы. Организация параллельных секций.

1.6.2. Краткое содержание вопросов:

Одним из наиболее популярных средств программирования компьютеров с общей памятью, базирующихся на традиционных языках программирования и использовании специальных комментариев, в настоящее время является технология **OpenMP**. За основу берется последовательная программа, а для создания ее параллельной версии*пользователю предоставляется набор директив, процедур и переменных*

окружения. Стандарт OpenMP разработан для языков Фортран, C и C++. Поскольку все основные конструкции для этих языков похожи, то рассказ о данной технологии мы будем вести на примере только одного из них, а именно на примере языка Фортран.

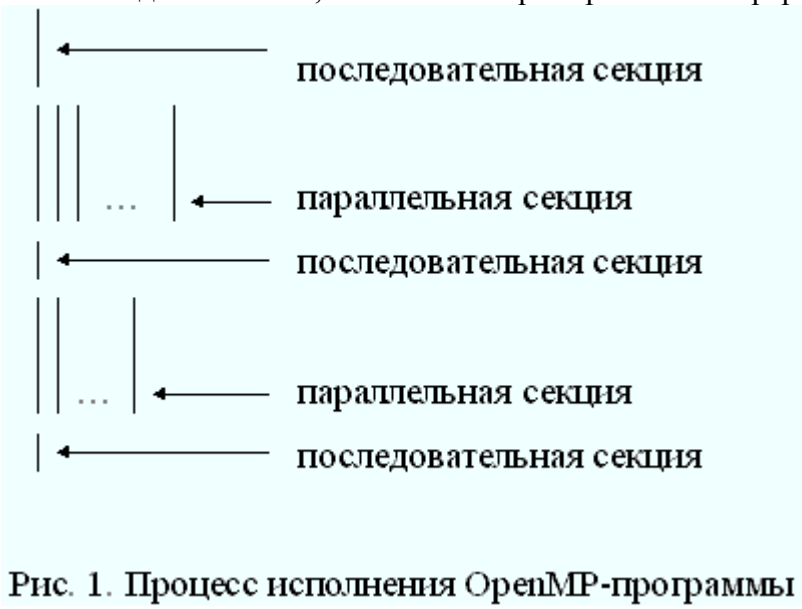

Рис. 1. Процесс исполнения OpenMP-программы

Как, с точки зрения OpenMP, пользователь должен представлять свою параллельную программу? Весь текст программы разбит на последовательные и параллельные области (см. рис.1). В начальный момент времени порождается нить-мастер или "основная" нить, которая начинает выполнение программы со стартовой точки. Здесь следует сразу сказать, почему вместо традиционных для параллельного программирования процессов появился новый термин - нити (threads, легковесные процессы). Технология OpenMP опирается на понятие общей памяти, поэтому она, в значительной степени, ориентирована на SMP-компьютеры. На подобных архитектурах возможна эффективная поддержка нитей, исполняющихся на различных процессорах, что позволяет избежать значительных накладных расходов на поддержку классических UNIX-процессов.

1.7. Лекция № 8 (2 часа)

Тема: «DVM система разработки параллельных программ».

1.7.1. Вопросы лекции:

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

1.7.2. Краткое содержание вопросов:

DVM-система, созданная в Институте прикладной математики им. М.В.Келдыша РАН, позволяет разрабатывать на языках C-DVM и Fortran-DVM параллельные программы для ЭВМ различной архитектуры и сетей ЭВМ. Аббревиатура DVM соответствует двум понятиям: Distributed Virtual Memory и Distributed Virtual Machine. Первое отражает наличие единого адресного пространства. Второе отражает использование виртуальных машин для двухступенчатой схемы отображения данных и вычислений на реальную параллельную машину.

При использовании языков C-DVM и Fortran-DVM программист имеет только один вариант программы и для последовательного и для параллельного выполнения. Эта программа, помимо описания алгоритма обычными средствами языков Си или Фортран 77, содержит правила параллельного выполнения этого алгоритма. Эти правила оформляются синтаксически таким образом, что они являются “невидимыми” для стандартных компиляторов с последовательных языков Си и Фортран и не препятствуют выполнению и отладке DVM-программы на рабочих станциях как обычной последовательной программы.

Программисту предоставляются следующие возможности 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 максимального или минимального значения).

Компилятор переводит программу на языке C-DVM (Fortran-DVM) в программу на стандартном языке Си (Фортран), расширенную функциями системы поддержки выполнения DVM-программ, которая для организации межпроцессорного взаимодействия использует стандартные коммуникационные библиотеки (MPI, PVM, Router).

2. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ПРОВЕДЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

2.1. Лабораторная работа №1, 2, 3 (6 часов).

Тема: «Цели, задачи и проблемы параллельных вычислений».

2.1.1. Вопросы к занятию:

Закон Амдаля (о существовании последовательных алгоритмов). Закон Мура (о росте производительности последовательных компьютеров). Закон Гроша (о высокой стоимости параллельных систем). Гипотеза Минского (о влиянии потерь на взаимодействие на степень ускорения параллельных вычислений по сравнению с последовательными). Сложность разработки параллельных алгоритмов. Трудоемкость проверки правильности параллельных программ.

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. Каковы ограничения максимальной производительности однопроцессорных ЭВМ.
2. Параллельные и распределенные вычисления и их техническая основа – вычислительные кластеры, ГРИД-системы и суперкомпьютеры.
3. Виды параллелизма.
4. Основные проблемы использования параллельной обработки данных.
5. Закон Амдаля о существовании последовательных алгоритмов.
6. Закон Мура о росте производительности последовательных компьютеров.
7. Закон Гроша о стоимости параллельных систем.
8. Гипотеза Минского о влиянии потерь на взаимодействие на степень ускорения параллельных вычислений по сравнению с последовательными.
9. Схемы взаимодействия ветвей параллельных алгоритмов.

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

Тема: «Архитектура высокопроизводительных ЭВМ».

2.2.1. 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 мультипроцессоры (SMP), массивно-параллельные компьютерные системы (MPP), кластеры.

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. 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:

+a) взаимодействие

b) независимость друг от друга

c) условную зависимость

d) разграничение

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

+a) одним процессором

b) двумя процессорами

c) тремя процессорами

d) несколькими процессорами

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

+a) очень быстро

b) очень медленно

c) быстро

d) медленно

2.3. Лабораторная работа № 6, 7, 8 (6 часов).

Тема: «Моделирование и анализ параллельных алгоритмов».

2.3.1. 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.

2.3.2. 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 параллельных вычислений в виде сети Петри.
2. Основные проблемы параллельных вычислений: синхронизация, взаимоисключение, блокировка (тупики).
3. Потоковая 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 используется для:
+a) 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:
+a) приоритетами
b) условиями
c) задачами
d) возможностями
3. В динамических алгоритмах планирования приоритет присваивается:
+a) динамически
b) статически
c) независимо
d) в зависимости от приоритета

2.4. Лабораторная работа № 9, 10 (4 часа).

Тема: «Принципы разработки параллельных алгоритмов и программ».

2.4.1. 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.

2.4.2. 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. 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 a) изучение и проверка спецификации модуля, выбор языка программирования
2 b) выбор алгоритма и структуры данных
6 c) компиляция модуля
4 d) шлифовка текста модуля
5 e) проверка модуля
3 f) программирование модуля

2.5. Лабораторная работа № 11, 12, 13 (6 часов).

Тема: «Средства разработки параллельных программ».

2.5.1. Вопросы к занятию:

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

2.5.2. 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. 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.

2.6. Лабораторная работа № 14, 15 (4 часа).

Тема: «Интерфейс передачи сообщений MPI».

2.6.1. Вопросы к занятию:

Коллективные функции обмена данных: широковещательная рассылка, функции сбора и рассыпания данных. Функции редукции данных. Создание групп процессов, области связи, коммуникаторы. Обмен данными внутри группы, межгрупповой обмен. Топология обменов. Декартовы топологии. Топологии произвольного графа. Примеры параллельных программ на основе обработки массивов.

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. Общие принципы построения и реализации MPI.
2. Общие функций MPI, коммуникаторы. Функции обмена сообщениями типа «точка-точка»: блокирующий и неблокирующий обмен, синхронные и стандартные посылки сообщений.
3. Предотвращение тупиков. Коллективные функции обмена данных: широковещательная рассылка, функции сбора и рассыпания данных.
4. Функции редукции данных. Создание групп процессов, области связи, коммуникаторы. Обмен данными внутри группы, межгрупповой обмен.
5. Топология обменов. Декартовы топологии. Топологии произвольного графа.
6. Последовательные и параллельные нити программы. Организация параллельных секций.

2.7. Лабораторная работа № 16, 17, 18, 19 (8 часов).

Тема: «Технология программирования OpenMP».

2.7.1. Вопросы к занятию:

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

2.7.2. Краткое описание проводимого занятия:

2.7.2.1. Ответы на вопросы лабораторной работы.

2.7.2.2. 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.

2.8. Лабораторная работа № 20, 21 (4 часа).

Тема: «DVM система разработки параллельных программ».

2.8.1. Вопросы к занятию:

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

2.8.2. Краткое описание проводимого занятия:

2.8.2.1. Ответы на вопросы лабораторной работы.

2.8.2.2. 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.