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

# МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.О.11 Параллельные методы и алгоритмы

**Направление подготовки (специальность)**
09.04.01 Информатика и вычислительная техника

**Профиль образовательной программы**
**“**Автоматизированные системы обработки информации и управления**”**

**Форма обучения** очная

**СОДЕРЖАНИЕ**

**1. Тематическое содержание дисциплины**

**1.1. Тема 1: «Параллельная форма алгоритма»(25часов).**

**1.1.1. Перечень и краткое содержание рассматриваемых вопросов:**

1. Концепция неограниченного параллелизма.

Для реализации алгоритма на параллельной системе его следует представить в виде последовательности групп операций. Отдельные операции в каждой группе должны обладать следующим свойством: их можно выполнять одновременно на имеющихся в системе функциональных устройствах. Итак, пусть операции алгоритма разбиты на группы, а множество групп полностью упорядоченно так, что каждая операция любой группы зависит либо от начальных данных, либо от результатов выполнения операций, находящихся в предыдущих группах. Представление алгоритма в таком виде называется параллельной формой алгоритма. Каждая группа операций называется ярусом, а число таких ярусов – высотой параллельной формы. Максимальное число операций в ярусах (число привлекаемых процессов в ярусах) - шириной параллельной формы.

Один и тот же алгоритм может иметь много параллельных форм. Формы минимальной высоты называются максимальными.

Рассмотрим следующие примеры.

Пример 1. Пусть требуется вычислить выражение

(a1a2 + a3a4)(a5a6 + a7a8);

соблюдая лишь тот порядок выполнения операций, который представлен в записи.

Фактически здесь фиксирован не один алгоритм, а несколько, эквивалентных по результатам.

Приведемодинизних:

Данные: a1, a2, a3, a4, a5, a6, a7, a8.

Ярус 1. a1a2, a3a4, a5a6, a7a8.

Ярус 2. a1a2 + a3a4, a5a6 + a7a8.

Ярус 3. (a1a2 + a3a4)(a5a6 + a7a8).

Высота этой параллельной формы равна 3, а ширина 4.

Особенность этой параллельной формы в том, что 4 процессора загружены только на первом ярусе, а на последнем ярусе загружен лишь один процессор.

Концепция неограниченного параллелизма предполагает использование идеализированной модели неограниченной параллельной вычислительной системы.

Неограниченной параллельной вычислительной системой будем называть систему, работа которой производится в течение некоторого количества единиц дискретного времени, называемых тактами, и которая обладает следующими свойствами:

1) система имеет любое нужное число идентичных процессоров;

2) система имеет произвольно большую память, одновременно доступную всем процессорам;

3) каждый процессор за упомянутую единицу времени может выполнить любую унарную или бинарную операцию из некоторого априори заданного множества операций; операции из этого множества будем называть основными;

4) время выполнения всех вспомогательных операций (т.е. операций, не являющихся основными), время взаимодействия с памятью и время, затрачиваемое на управление процессором, считаются пренебрежимо малыми;

5) никакие конфликты при общении с памятью не возникают;

6) все входные данные перед началом работы системы записаны в память;

7) после окончания вычислительного процесса все результаты остаются в памяти.

2. Условия реализуемости алгоритмов.

Реализация алгоритма на вычислительной системе - это реализация базовой программы эквивалентным способом. Поэтому (явно или неявно) должны быть определены:

- функция, устанавливающая взаимно однозначное соответствие между вершинами графа алгоритма и отдельно срабатываниями функциональных устройств вычислительной системы;
- потоки информации между функциональными устройствами системы;
- моменты включения функциональных устройств.

Имеются две основные стратегии при решении этих задач:

- статическая;
- динамическая.

При статической стратегии решение принимается заранее на основе анализа алгоритма и вычислительной системы, а при динамической стратегии решение принимается в процессе реализации.

Динамическая стратегия характеризует так называемые потоковые линии. Иногда используют смешанные стратегии.

Принято на этой стадии исследования идеализировать ситуацию, предполагая, что система не может сработать и выдать неправильный результат (т.е. результат, относящийся к другому алгоритму): либо система срабатывает и реализует алгоритм, либо система не срабатывает. Этот подход на самом деле во многих случаях весьма далек от истины: например, при решении той или иной вычислительной задачи обычно предполагается, что указанные в алгоритме арифметические действия выполняются точно; однако, при реальных вычислениях компьютерная система реализует "машинную арифметику", так что результат фактически относится к реализации другого алгоритма (причем реализуемый алгоритм, как правило, остается неизвестным).

3. Функциональные устройства как составляющие параллельной вычислительной системы.

Предположим, что вычислительная система состоит из функциональных устройств и работает по тактам, т. е. каждая операция начинается или заканчивается только в фиксированные равноотстоящие моменты времени. Для простоты полагаем, что все рассматриваемые функциональные устройства синхронизированы, т.е. могут включаться лишь в упомянутые моменты времени.

Временной интервал между соседними моментами времени будем считать единичным и называть его тактом.

Функциональное устройство называется простым, если время выполнения операции на нем (число тактов) определено априори и никакая следующая операция не может начаться раньше окончания предыдущей.

Функциональное устройство называется конвейерным, если время выполнения на нем любой операции заранее определено, но следующая операция может начать выполнение через один такт после начала выполнения предыдущей.

Предполагается, что выполнение любой операции происходит за целое число тактов.

Замечание. Основное отличие между простым и конвейерным функциональным устройством состоит в том, что простое функциональное устройство использует все свое оборудование для выполнения одной операции, а конвейерное - распределяет свое оборудование между несколькими операциями, к выполнению каждой из которых оно приступает последовательно, обычно не закончив выполнение предыдущих. В некоторых

случаях конвейерное функциональное устройство можно рассматривать как особое устройство распараллеливания, точнее - "диагонального" распараллеливания.

### 1.2. Тема 2: «Параллелизм при обработке информации» (20 часов).

#### 1.2.1. Перечень и краткое содержание рассматриваемых вопросов:

1. Конвейерные вычисления.

Наиболее эффективным способом параллельной обработки вычислений является конвейерный. Для упрощения обсуждения конвейеризации вычислений условимся о следующем:

- на первом этапе откажемся от рассмотрения смыслового содержания операций, выполняемых отдельными функциональными устройствами;
- рассмотрим по возможности минимальное число характеристик конвейерных вычислений;
- ограничимся наиболее важными задачами.

Будем предполагать, что имеется три этапа обработки данных:

- короткий подготовительный этап, определяющий настройку конвейерного устройства, связанный с нестандартными вычислениями;
- длинный, в котором происходит основная обработка;
- короткий, заключительный, определяющий вывод полученных результатов и выполняющий нестандартные вычисления в конце процесса.

В связи с этим считаем, что на втором этапе:

- данные обрабатываются однозначно;
- отсутствует поиск и подкачка данных;
- фиксирована коммутация функциональных устройств.

Кроме того, будем считать, что:

- имеется полная или почти полная загруженность оборудования конвейерного устройства;
- результаты обработки любого функционального устройства могут быть переданы для дальнейшей обработки без обращения к общей памяти;
- все функциональные устройства имеют одинаковую номинальную производительность.

2. Векторные вычислительные машины.

Многие вычислительные методы широко используют операции с векторами, которые характеризуются однотипностью и естественной параллельностью арифметических действий. Поэтому вычислители, ориентированные на быстрое выполнение векторных операций, традиционно называют векторными.

Векторные операции представляют собой совокупность независимых скалярных операций над координатами векторов с согласованными номерами. Поскольку операции однотипные, то для их реализации естественно применить конвейерные функциональные устройства.

Особенностью ситуации состоит в том, что на конвейерное функциональное устройство нужно быстро подавать данные и быстро считывать результаты на каждом такте. Это достаточно трудно осуществлять, если конвейерные функциональные устройства взаимодействуют с общей памятью, поэтому векторные вычислители снабжаются сверхбыстрой памятью, с которой в основном взаимодействуют упомянутые функциональные устройства.

Эту сверхбыструю память можно считать промежуточной между функциональными устройствами и основной (сравнительно медленной) памятью.

3. Параллельные суперкомпьютеры.

Потребности решения задач математической физики, обработки больших потоков информации и работы с большими базами данных привели к созданию компьютерных систем, мощности которых поражают воображение. Постоянно обновляемый список TOP500 содержит перечень наиболее мощных современных компьютеров (его можно увидеть в Интернете по адресу www.top500.org); в этом списке компьютеры располагаются в порядке убывания их мощности. Конечно, "мощность" - понятие условное; она определяется с помощью общепринятых специальных тестов, которые позволяют охарактеризовать различные параметры компьютера, и не всегда объективно отражают его свойства при решении той или иной конкретной задачи.

Приведем наиболее впечатляющие примеры суперкомпьютеров и некоторые их параметры.

Векторно-конвейерный суперкомпьютер CRAY T932 имеет 32 процессора, каждый с быстродействием 2 миллиарда операций в секунду 8 гигабайт оперативной памяти, 256 терабайт дискового пространства.

Компьютер T3E1200 фирмы SGI имеет максимальную производительность 891.500.000.000 операций в секунду на пакете LINPACK и содержит 1080 процессоров; пиковая производительность компьютера равна 1.296.000.000.000 операций в секунду.

Компьютер ASCI Red фирмы Intel имеет максимальную производительность 1.338.000.000.000 операций в секунду на пакете LINPACK и содержит 9152 процессора; пиковая производительность компьютера достигает 1.830.400.000.000 операций в секунду.

Компьютер Earth Simulator фирмы NEC имеет максимальную производительность 35.860.000.000.000 операций в секунду на пакете LINPACK и содержит 5120 процессоров; пиковая производительность компьютера достигает 40.960.000.000.000 операций в секунду. Наконец, наиболее мощный современный компьютер BlueGene /L фирмы IBM имеет максимальную производительность 280.600.000.000.000 операций в секунду на пакете LINPACK и содержит 131.072 процессора (см. 29-ю редакцию списка TOP500).

## 1.3. Тема 3: «Переход от последовательных программ к параллельным» (25 часов).

### 1.3.1. Перечень и краткое содержание рассматриваемых вопросов:

1. Технология программирования Open MP.

Исторически сложилось так, что при численном решении задач пользователи пишут программы на последовательном языке. Если оказалось, что без распараллеливания та или иная программа работает не эффективно, то ее можно попытаться распараллелить с тем, чтобы запустить задачу на параллельной системе. Кроме того, для многих пользователей психологически удобнее программирование начинать с последовательной программы. Поэтому естественно задаться вопросом о возможности автоматического распараллеливания программ транслятором. Однако, решение задачи распараллеливания данной программы достаточно сложно.

Для распараллеливания программы требуется:

1) найти участки программы, где распараллеливание возможно,

2) распределить эти участки по вычислительным модулям,

3) обеспечить вычисления правильной синхронизацией.

Особенно трудно выполнить эти требования в случае системы с распределенной памятью. Но даже для системы с разделяемой памятью это достаточно сложно сделать автоматически.

В технологии Open MP за основу берется последовательная программа. Для создания параллельной версии пользователю представляются наборы:

1) директив,

2) процедур,

3) переменных окружения.

Стандарт Open MP разработан для языков Fortran и C (Fortran 77, 90, 95 и C, C++) и поддерживается производителями всех больших параллельных систем. Реализации стандарта доступны в UNIX и в среде Windows NT.

Распараллеливание программы состоит в том, чтобы весь текст разбить на последовательные и параллельные области.

В начальный момент порождается нить-мастер (или основная нить), которая начинает выполнение программы со стартовой точки. Основная нить и только она исполняет все последовательные области секции программы.

Для поддержки параллелизма используется схема FORK/JOIN. При входе в параллельную область основная нить порождает дополнительные нити (выполняется операция FORK). После порождения дополнительные нити нумеруются последовательными натуральными числами, причем основная нить имеет номер 0; таким образом, каждая нить получает свой уникальный номер. Все порожденные нити выполняют одну и ту же программу, соответствующую параллельной области. При выходе из параллельной области основная нить дожидается завершения работы остальных нитей (выполняется операция JOIN), и дальнейшее выполнение программы осуществляется основной нитью.

## 2. Использование стандарта MPI.

К настоящему времени разработаны стандарты на языки программирования высокого уровня, которые позволяют достаточно быстро освоить эффективное использование компьютера и поддерживают переносимость создаваемых программ. Разработаны стандарты на элементную базу, предназначенную для создания компьютеров. В рамки подобного развития укладывается и создание стандартов на параллельные компьютерные системы и на параллельное программирование. Одним из наиболее распространенных стандартов параллельного программирования является стандарт MPI.

Аббревиатура MPI расшифровывается как MessagePassingInterface, что можно перевести как Интерфейс передачи сообщений. Этот стандарт был развит группой MessagePassingInterfaceForum (MPIF). Цель его разработки состояла в создании удобных средств параллельного программирования со свойствами эффективности и гибкости при практическом применении. Группа MPIF с участием более 40 организаций начала свою работу в 1992 году; первая (черновая) версия была опубликована в 1994 году, причем работа велась в Исследовательском центре IBM.

Вторая версия появилась в 1997 году (подробности см. по адресу http://www.mpi-forum.org). К числу требований, которые были предъявлены к результирующей версии стандарта MPI, относятся: 1) реализации для языков C и Fortran77 при использовании различных платформ, 2) удобство использования, позволяющее программисту минимальным образом представлять себе архитектуру параллельной системы, 3) высокая степень переносимости и масштабируемости разрабатываемой пользователем программы.

3. Технология программирования DVM.

Модель Digital Virtualmashine (DVM) положена в основу Fortran-DVM и C-DVM, разработанных в Институте прикладной математики Российской Академии Наук (ИПМ РАН).

При проектировании реализованы следующие принципы.

1. Система базируется на высокоуровневой модели выполнения программы, которая привычна для прикладного программиста.

2. Спецификации параллелизма остаются невидимыми для обычных компиляторов Fortran-77 и C.

3. Языки распараллеливания предлагает программисту модель программирования, близкую к модели исполнения.

4. Основная работа по реализации модели выполняется системой поддержки распараллеливания DVM.

Система DVM базируется на библиотеке MPI (Message Passing Interface) для обеспечения высокой степени переносимости программ. Для повышения надежности работы в настоящее время разрабатывается поддержка DVM-программ с помощью стандарта Open MP. Все DVM-директивы оформлены в виде строк, начинающихся любым из символьных сочетаний CDVM$, *DVM$ или !DVM$. Синтаксис и семантика директив в языках Fortran-DVM и C-DVM практически совпадают.

Модель выполнения DVM-программы описывается следующим

образом.

1. DVM-программа выполняется на виртуальной многопроцессорной системе (MPI-машине, PVM-машине и т.п.).

2. Виртуальная многопроцессорная система представляется в виде многомерной решетки процессоров.

3. В момент запуска DVM-программа начинает свое выполнение сразу на всех процессорах виртуальной вычислительной системы, причем в это же время присутствует единственный поток управления (единственная ветвь).

4. DVM допускает ограниченную иерархию параллелизма:

- на верхнем уровне описывается то или иное число независимых ветвей (задач), которые могут выполняться параллельно (независимые по данным крупные блоки программы);
- в конце ветвей может быть выполнена глобальная операция редукции;
- в каждой ветви могут дополнительно выделяться параллельные циклы.

Никакие другие варианты иерархии параллелизма не допускаются.

5. При входе в параллельную конструкцию поток управления разбивается на ряд параллельных потоков, каждый из которых управляет процессом вычислений на своем процессоре.

6. Все переменные репродуцируются по всем процессорам: в каждом процессоре появляется локальная переменная того же типа и с тем же именем; исключением являются специально указанные "распределенные массивы", способ расположения которых определяется соответствующей директивой.

7. Любой оператор присваивания выполняется по правилу "собственных вычислений": он выполняется тем процессором, на котором распределена переменная, стоящая в левой части оператора присваивания.