The paper describes two systems for generation of computational model using REACTOR-GP code package. The first system, called Constructor of Reactor Computational Model, is meant for computational problems of the reactor shielding. The second system, called Constructor of Core Computational Model, is meant for computational problems of the core of a reactor under design or operation. The first system is universal and generates mesh cells of computational region and their composition on the basis of combinatorial geometry. The second system forms a 3D computational model considering geometric and physical features of most nuclear power reactor cores.

УДК .001.1:621.039

А.В. Воронков; А.С. Голубев; В.А. Крюков; Е.П. Сычугова
(ИПМ им. М.В. Келдыша РАН)

ПАРАЛЛЕЛЬНАЯ ВЕРСИЯ ПАКЕТА РЕАКТОР-ГП

В работе обобщен опыт использования параллельных расчетов для задач расчета ядерных реакторов. В качестве параллельных систем применялись наиболее доступные, кластеры и многоядерные компьютеры. Программными инструментами распараллеливания служили FORTRAN-DVM и FORTRAN-OpenMP. Приведены рекомендации по созданию параллельных программ и характеристики прогона тестовых задач, коэффициенты эффективности распараллеливания на реальных задачах.

Введение

Начиная с 1985 года, все программные разработки по нейтронно-физическим задачам в ИПМ РАН проводятся в рамках модульной идеологии, что позволило объединить все программы в единый комплекс – пакет прикладных программ РЕАКТОР [1], который уже в течение многих лет эксплуатируется в ИПМ РАН и ряде других организаций. Комплекс РЕАКТОР позволяет проводить численный расчет как стационарного состояния, так и кампании ядерного реактора. Счет может проводится в одно-, двух- и трехмерных геометриях, в различных приближениях по энергии (одно-, мало- и многогрупповое), с использованием различных приближений пространственной части уравнения переноса (диффузионное, P1, кинетическое).

В начале 2000-х годов появилась третья версия пакета РЕАКТОР, версия РЕАКТОР-ГП, максимально удовлетворяющая потребностям физика-расчетчика и инженера-проектанта. Этому способствовало то, что пакет РЕАКТОР-ГП был создан в результате тесных контактов и совместных работ с ОКБ «ГИДРОПРЕСС».

Растущие требования к вычислительной мощности со стороны задач реакторной физики, относительная доступность вычислительных кластеров, массовое производство многоядерных процессоров делают все более актуальным и реальным создание параллельных программ для прикладных задач реакторной физики. С целью ускорения расчетов ядерных реакторов была создана параллельная версия основных расчетных модулей пакета РЕАКТОР-ГП. Большинство модулей пакета не требует значительного времени счета и поэтому может работать в последовательном режиме.

Следует отметить, что эти модули по числу и размерам составляют большую часть пакета РЕАКТОР (~90 %). Самыми времязатратными модулями пакета РЕАКТОР-ГП являются модули расчета полей нейтронов и гамма-квантов в многомерных геометриях (~5-10 %). Поэтому именно для этих модулей были в первую очередь распараллелены алгоритмы и программы.

Однако модули как первого, так и второго класса должны работать на одном компьютере в согласованном режиме. Это приводит к тому, что модули первого класса также должны быть модифицированы. Требует переделок и системная часть пакета.

Ниже приводится краткое описание этой работы и результаты по различным методам распараллеливания.

Постановка задачи и основные уравнения

Как отмечалось во введении, самыми времязатратными модулями пакета РЕАКТОР-ГП

являются модули расчета полей нейтронов и гамма-квантов в трехмерной геометрии. В составе пакета РЕАКТОР-ГП функционируют 3 модуля решения уравнений переноса в трехмерной геометрии: модуль KIN3D – перенос в X-Y-Z геометрии, модуль KIN3D6 – перенос в HEX-Z геометрии и модуль KINRTZ – перенос в R-PHI-Z геометрии.

Принципиальной разницы в подходах для распараллеливания этих задач нет. Поэтому для дальнейших исследований был выбран модуль KIN3D. Ниже приведена математическая постановка задачи для этого модуля.

Уравнение переноса нейтронов в общем виде в многогрупповом приближении в некоторой пространственной области E может быть записано в следующем виде:

$$(\vec{\Omega}\cdot\nabla\Psi^g(\vec{r},\vec{\Omega}))+\Sigma_t^g(\vec{r})\cdot\Psi^g(\vec{r},\vec{\Omega})= =S^g(\vec{r},\vec{\Omega}) \quad (1)$$

Правая часть уравнения (1) для нейтронов группы g (g=1,2,...,*NG*) имеет следующий вид:

$$S^g(\vec{r},\vec{\Omega})=\sum_{h=H\min(g)}^{H\max(g)}\int_{\Omega}P^{h\to g}(\vec{r},\vec{\Omega}\cdot\vec{\Omega}')\Psi^h(\vec{r},\vec{\Omega}')d\vec{\Omega}'+ +\frac{1}{4\pi}\chi_p^g\sum_{h=1}^{NG}\nu\Sigma_f^h\bar{\Psi}^h(\vec{r})+\frac{1}{4\pi}Q^g(\vec{r}) \quad (2)$$

где

$$\bar{\Psi}^g(\vec{r})=\int_{\Omega}\Psi^g(\vec{r},\vec{\Omega})d\vec{\Omega} \quad (3)$$

В (1)-(3) использованы следующие обозначения:

$\vec{r}$ – радиус - вектор,

NG – полное число энергетических групп,

$\vec{\Omega}$ – единичный вектор в направлении полета нейтронов,

$\Psi^g(\vec{r},\vec{\Omega})$ – плотность потока в точке $\vec{r}$ в направлении $\vec{\Omega}$ в группе g,

$\bar{\Psi}^g(\vec{r})$ – скалярный поток нейтронов в точке $\vec{r}$, $\Sigma_t^g(\vec{r})$ – полное макроскопическое сечение, $P^{h\to g}(\vec{r},\vec{\Omega}\vec{\Omega}')$ – макроскопическое сечение рассеяния,

$\Sigma_f^h(\vec{r})$ – макроскопическое сечение деления,

ν – число вторичных нейтронов, возникающих при одном акте деления.

$Q^g(\vec{r})$ – функция распределения внутреннего заданного источника,

χ_p^g – спектр деления мгновенных нейтронов.

На границе Г пространственной области E задаются значения углового потока, входящего внутрь этой области:

$$\Psi^g(\vec{r},\vec{\Omega})\Big|_{\Gamma,\vec{\Omega}\cdot\vec{n}<0}=\Psi_\Gamma^g(\vec{r},\vec{\Omega}) \quad (4)$$

Здесь $\vec{n}$ - внешняя нормаль к границе Г.

Система уравнений (1) – (4) описывает стационарное распределение нейтронов с учетом процесса деления, а также заданных внутренних и граничных источников. Если в правой части (2) член с делением отсутствует ($\nu\Sigma_f^g=0$ для всех g), эта система уравнений описывает стационарное распределение нейтронов в зависимости от заданных источников. Задача на собственное значение K_{ef} (однородная задача) описывается системой уравнений (1) – (4) с нулевыми внутренними и граничными источниками и множителем $1/K_{ef}$ перед членом с делением, где K_{ef} – коэффициент размножения нейтронов.

Уравнение переноса (1) в трехмерной X-Y-Z геометрии имеет следующий вид:

$$\mu\frac{\partial\Psi^g}{\partial x}+\eta\frac{\partial\Psi^g}{\partial y}+\xi\frac{\partial\Psi^g}{\partial z}+ +\Sigma_t^g(x,y,z)\cdot\Psi^g(x,y,z,\mu,\eta,\xi)= =S^g(x,y,z,\mu,\eta,\xi) \quad (5)$$

g=1,2,...,NG;

где направляющие косинусы μ, η и ξ являются координатами вектора направления полета частицы $\vec{\Omega}$: $\xi=(\vec{\Omega}\cdot\vec{n}_z)$, $\mu=(\vec{\Omega}\cdot\vec{n}_x)$, $\eta=(\vec{\Omega}\cdot\vec{n}_y)$, $\mu=\sin\theta\cos\varphi$, $\eta=\sin\theta\sin\varphi$, $\xi=\cos\theta$, $\mu^2+\eta^2+\xi^2=1$. В X-Y-Z геометрии рассматривается полный диапазон изменений параметров: $\theta\in[0,\ \pi]$; $\varphi\in[0,\ 2\pi]$; $\mu,\eta,\xi\in[-1,\ 1]$; $x\in[0,\ x_B]$; $y\in[0,\ y_B]$; $z\in[0,\ z_B]$.

Система уравнений (5) решается в области $D=\{(x,\ y,\ z)\in E,\ \xi\in[-1,+1],\ \mu\in[-1,+1],\ \eta\in[-1,+1]\}$ где пространственная область E – выпуклая и состоит из правильных четырехугольных призм.

Проблемы переноса программ на многопроцессорные кластеры

Перенос последовательных прикладных программ с персонального компьютера на высокопроизводительные многопроцессорные вычислительные системы включает два аспекта. Первый аспект – это перенос последовательной программы с персонального компьютера на МВС в режиме последовательной задачи. Второй аспект – переделка последовательного варианта программы в параллельный. Распараллеливание можно выполнять

различными инструментами. В зависимости от программы и используемых ею исходных данных простой перенос последовательной задачи на другую платформу может оказаться значительно сложнее разработки параллельного варианта программы.

Ниже рассматривается аспект переноса последовательной задачи на МВС

Мультипроцессорные системы имеют обычно в качестве операционной системы сходные разновидности OS UNIX и различающиеся компиляторы с Фортрана и пакеты MPI. Чаще всего OS UNIX имеет меньше средств разработки и отладки программ по сравнению с OS Windows, обычно используемой при последовательном счете на персональных компьютерах. Кроме того, не на всех мультипроцессорных системах сообщения об ошибках при выполнении программы доходят до пользователя в корректном виде.

Для компиляции модулей больших программ на мультипроцессорных системах требуются makefile'ы, тогда как в Visual Studio под OS Windows они создаются автоматически. Список неудобств можно дополнить различием файловых систем Visual операционных систем в плане различия больших и маленьких букв. В OS UNIX большие и маленькие буквы считаются различными, в OS Windows одинаковыми. Служебные байты в конце строк также различаются.

Перенос программ на кластеры заключался в написании makefile'ов для Linux и замене всех имен модулей на нижний регистр. Концы строк в текстовых файлах должны быть преобразованы в формат Linux с помощью какой-либо утилиты типа файлового менеджера FAR или при передаче файлов на МВС с помощью опции передачи.

При компиляции, возможно, придется исправлять какие-либо операторы (чаще всего операторы ввода-вывода) из-за различий во входном языке.

В зависимости от компилятора Фортрана, длина записи операторов бесформатного ввода-вывода задается либо в байтах, либо в словах. На кластерах mvs50k [6] и rsc4 [7] установлены компиляторы с Фортрана Intel. Компилятор имеет опцию, задающую длину бесформатной записи в словах или в байтах. Компилятор можно использовать также при работе в Windows. Такое совпадение позволяет пускать часть модулей пакета РЕАКТОР на персональном компьютере, а основной счетный модуль после переписи подготовленных данных пускать на кластере. Если задание длины записи на кластере отличается от персонального компьютера, приходится иметь два варианта программ архива, выполняющих ввод-вывод.

Правка программы переноса библиотеки констант через текстовый формат была связана с различием в представлении чисел в памяти компьютера. Персональные компьютеры на основе процессоров Intel используют т.н. Little Endian представление. Числа запоминаются в памяти начиная с младших байтов. Процессоры Power PC (МВС15000 [6]) используют представление Big Endian. Числа запоминаются в памяти, начиная со старших байтов.

Главная особенность модуля KINXYZ при запусках на МВС15000 заключалась в том, что возникали спецзначения Infinity(бесконечность) и NaN(Not a Number) в тех местах программы, в которых на других платформах получались обычные числа.

Как показало изучение этого эффекта, это было связано как с особенностью программы, так и с особенностью МВС15000.

Особенность задачи KINXYZ состоит в том, что в процессе счета образуются очень малые по абсолютной величине числа. Согласно стандарту IEEE четырехбайтовые числа с плавающей точкой представляются до величины $\sim 10^{-38}$ в нормализованном виде. В диапазоне $10^{-38} – 10^{-45}$ – в ненормализованном виде с потерей точности. Менее 10^{-45} – как машинный 0. Потеря точности при сдвиге мантиссы вправо увеличивается и может доходить в крайнем случае (единица в крайнем правом разряде) до 100 % в зависимости от метода округления. Операции с такими числами могут приводить к недостоверным результатам. Так, результат деления двух чисел $\sim 10^{-45}$ может оказаться равным двум вместо единицы.

Большинство процессоров позволяют выполнять операции деления над ненормализованными числами. Процессор PowerPC(МВС15000) при делении на ненормализованное число выдает спецзначение Infinity.

Для обеспечения поведения задачи, аналогичного поведению на процессорах Intel, необходимо завести рабочую переменную double precision dz6 для делителя специально для выполнения деления.

Описанное изменение операций деления позволило получить результаты работы программы KINXYZ на МВС15000, аналогичные результатам на процессорах Intel.

Типы параллелизма

Принято разделять параллелизм на два вида: параллелизм циклов и функциональный

параллелизм. Первый заключается в параллельном выполнении циклов путем распределения его витков между процессорами. Второй пареллелизм заключается в выделении из основного алгоритма максимально независимых подалгоритмов и выполнении их на разных процессорах (функциональный параллелизм).

В задачах реакторной физики чаще встречается пареллелизм первого типа.

Среди параллельных вычислительных систем принято выделять подкласс систем с общей памятью на все процессоры (или на все ядра), SMP-системы (Symmetrical MultiProcessing) и подкласс систем с распределенной памятью MPP (Massively Parallel Processing).

Примером первой может служить персональный компьютер с многоядерным процессором, примером второй – кластер, построенный на основе серверных компонентов. В качестве средства распараллеливания использовался OpenMP для SMP–систем и DVM для MPP–систем.

При написании параллельного варианта программы в первую очередь надо стремиться к избавлению от неявных зависимостей по данным как при использовании параллельных SMP, так и MPP-систем. При переделке последовательной программы в параллельную трудозатраты могут возрасти до нескольких раз.

Ниже описывается опыт создания параллельных программ расчета ядерных реакторов. В качестве типичной задачи выбрана задача расчета защиты реактора в кинетическом приближении с помощью модуля KINXYZ, являющаяся одной из самых времязатратных в составе пакета РЕАКТОР-ГП. В качестве тестовых данных использовались данные защитной задачи реакторной установки СВБР 75/100 с конечно-разностной сеткой 101*101*237 точек и 30 энергетическими группами.

Для тестовых прогонов использовались кластеры СКЦ РАН (MVS15000,MVS50k) [6], ИПМ (RSC4) [7], многоядерный ПК (Четырехъядерный процессор Intel Q6600, 8 Гбайт оперативной памяти, два жестких диска по 500 Гбайт, OS Windows XP x64, Компилятор Intel Fortran 10.0).

Инструменты распараллеливания

В качестве инструмента для MPP – систем применялась доступная относительно давно система создания параллельных программ DVM (Distributed Virtual Memory) [2-5]. Система включает в себя FORTRAN-DVM, конвертор с языка FORTRAN-DVM в язык FORTRAN со вставками MPI, систему поддержки во время выполнения, систему сбора статистики по временам выполнения программы, интерпретаторы команд компиляции и запуска задач на счет, а также систему отладки. DVM – система включает в себя также конвертор с языка C-DVM, который для описываемых задач не использовался. По сравнению с широко распространенной системой распараллеливания MPI(Message Passing Interface) FORTRAN-DVM представляет собой язык более высокого уровня. Конвертор FORTRAN-DVM автоматизирует расстановку вызовов к функциям MPI, избавляя программиста от рутинной работы.

FORTRAN-DVM создавался как аналог HPF (High Performance Fortran), но все основные решения по созданию параллельной программы возлагаются не на компилятор, а на программиста. Главная задача программиста – обеспечить структуру программы, пригодную для параллельного выполнения. Это справедливо и для OpenMP. Первоначальный вариант DVM был рассчитан в, основном, на MPP – системы. Относительно недавно стали широко доступны как аппаратные, так и программные средства разработки параллельных программ для SMP-систем. Это многоядерные процессоры и компиляторы с поддержкой OpenMP [8]. В настоящее время создан вариант конвертора FORTRAN-DVM-OpenMP, рассчитанный на кластеры с многоядерными узлами.

В FORTRAN-DVM, так же как и в OpenMP, директивы распараллеливания задаются, в основном, в виде псевдокомментариев. Программу FORTRAN-DVM и программу OpenMP можно использовать без переделок как последовательную программу, задавая для компилятора соответствующие опции.

Описываемые далее программы пакета РЕАКТОР выполнены как универсальные. Без изменений они могут использоваться на системах, поддерживающих программы OpenMP(например, многоядерных ПК), FORTRAN-DVM (например, кластерах), и FORTRAN-DVM-OpenMP (кластерах с многоядерными узлами).

Исследовавшиеся методы распараллеливания

Применявшийся в задачах пакета РЕАКТОР параллелизм относится к типу параллелизма циклов. В зависимости от того, какие циклы превращаются в параллельные, можно выделить два метода получения параллельной программы.

Основной – распараллеливание циклов по точкам конечно-разностной сетки. Для рассматриваемой кинетической задачи возможно также параллельное выполнение цикла по направлениям.

Распараллеливание по пространству

Под распараллеливанием по пространству будем в дальнейшем понимать такую организацию итерационных циклов по конечно-разностной сетке, когда конечноразностная сетка и соответствующие данные разделяются между процессорами. Каждый процессор обрабатывает ту часть данных, которая была ему выделена.

Основных преимуществ два. Первое преимущество – универсальность. Все рассматриваемые итерационные задачи реакторной физики имеют циклы по конечноразностной сетке и могут быть распараллелены по пространству. Второе преимущество – потенциально большое количество процессоров, используемое при параллельном счете. Большой размер конечно-разностных сеток с тенденцией к росту позволяет делить их на большое число подобластей.

В SMP-системах предоставляется общая память для всех процессоров (ядер) и поэтому проблемы распределения данных не возникает.

В системах типа MPP каждый процессор или группа процессоров (узел) имеет свою локальную память, к которой могут адресоваться только они. Основной принцип программирования для таких систем - принцип собственных вычислений. Из-за локальности памяти всю конечно-разностную сетку и привязанные к ней данные распределяют между процессорами. Принцип собственных вычислений заключается в том, что присвоение значений элементу массива выполняется на том процессоре, который является хозяином данного элемента массива. Распределение данных и обеспечение принципа собственных вычислений в DVM – системе автоматизировано.

Распределять необходимо не все массивы данных. Относительно небольшие массивы можно и нужно хранить в размноженном представлении. Размноженный массив - это такой массив, копия которого хранится в локальной памяти каждого из MPP-процессоров. Для определения массивов, которые необходимо описать как распределенные, требуется проанализировать основной цикл программы и выявить те массивы, которым присваиваются значения в теле основного цикла, занимающего основное время счета.

Принятый в задачах пакета РЕАКТОР способ представления массивов оставляет один вариант их распределения по процессорам. Явным образом задаются два индекса – индекс плоскости IZ и индекс по плоскости, вычисляемый на основе индекса строки L и индекса в строке I.

В рассматриваемой задаче максимальное количество задействованных в расчете процессоров ограничено количеством горизонтальных слоев в расчетной сетке метода конечных разностей. В циклах, объявленных директивой CDVM$ PARALLEL (IZ) ON QF(*,*,IZ) параллельными, каждый процессор обрабатывает не массивы целиком, а выделенные ему блоки плоскостей, для чего полный диапазон индексов IZ подменяется вырезками, соответствующими размещенному блоку.

Пропорционально доле блока в целом массиве уменьшается и работа на параллельных циклах, выполняемых на одном процессоре.

Для создания параллельной программы необходимо знать, где находятся основные циклы, структуру и циркуляцию потоков данных, прежде всего массивов и редукционных переменных в основном цикле (редукционные переменные - это те переменные, значения которых получаются как функции всех элементов массива путем суммирования, умножения, нахождения минимума и максимума).

Исходный последовательный вариант алгоритма кинетической задачи KINXYZ использует детерминированный алгоритм последовательного перебора направлений единичной сферы и последовательный перебор расчетных точек для каждого направления.

На каждой внутренней итерации в каждой расчетной точке вычисляются угловые моменты потока. Каждое направление определяется номером NN направления в квадранте, номером квадранта ID и неявно верхней или нижней полусферой.

Перебор по сетке выполняется сначала сверху вниз (соответствует нижней полусфере), затем снизу вверх (соответствует верхней полусфере). Для элементарного прямоугольника, соответствующего точке (I,L,IZ), вычисляются поток AN, соответствующий направлению NN, и выходящие потоки AX1, AY1, AZ1 через грани по осям X, Y, Z. Выходящие потоки используются как входящие AX0, AY0, AZ0 при расчете соседних точек.

Значения угловых потоков AN для каждого направления NN запоминаются в соответствующих массивах FZ1-FZ8. В процессе расчета угловых потоков используются “новые значения” (входящие потоки) с соседних точек. А именно, при счете сверху вниз используется входящее излучение с верхнего слоя IZ+1,

с предыдущей строки L-1 и от предыдущей точки I-1. При прогоне снизу вверх используется входящее излучение с предыдущего слоя IZ-1, с предыдущей строки L-1 и предыдущей точки в строке I-1. Строго говоря, порядок прохождения строк L и точек в строке , также как и плоскостей, может быть от больших к меньшим).

Процедурой свертки CONVOL угловые потоки по направлениям FZ1-FZ8 сворачиваются в моменты потока по присоединенным функциям Лежандра, умноженным на косинусы или синусы. Результаты помещаются в массивах RmomNC и RmomNS. Затем с использованием полученных массивов моментов делается следующая итерация в группе.

Главное требование к циклам расчета по просторанству для эффективности распараллеливания состоит в минимизации зависимости по данным. Перебирать расчетные точки в произвольном порядке можно лишь с условием, что входящее излучение для точек по трем измерениям имеет новое значение.

При переходе счета на другой Z слой, расположенный на другом процессоре, необходимо будет передавать значения массива AZ0. Изменим исходный последовательный алгоритм расчета таким образом, чтобы имелась возможность менять порядок расчета точек. Для этого надо будет хранить входящие излучения для большего числа точек

Для большей простоты алгоритма распараллеливания было решено оставить перебор точек по строке (по X) последовательным, а в циклах по L и Z реализовать возможность произвольного перебора строк и плоскостей, что потребовало значительной переделки программы.

Чтобы обеспечить это условие в исходной программе, выполнить следующие изменения. Во-первых, пришлось вынести из подпрограммы ddpn цикл по Z и внести его в подпрограмму wdifpn для объединения с циклами по Y и X. Для возможности выполнять в FORTRAN-DVM проходы строк в свободном порядке, необходимо, чтобы заголовки циклов по Z и Y следовали непосредственно друг за другом (тесно-гнездовой вид). Находившиеся между ними операторы

```
DO 122 IIZ=NZ,1,-1
KV=KVZ(IIZ)
...
VOL=VQ(IIZ)
DO 122 L=1,JMAX
```

были убраны с помощью оператора IF в тело цикла

```
DO 122 IIZ=NZ,1,-1
DO 122 L=1,JMAX
if(L.eq.1)Then
KV=KVZ(IIZ)
...
VOL=VQ(IIZ)
endif
```

Во-вторых, в начале тела цикла значения входящих потоков по Y и Z выбираются из массива AZ0m и присваиваются рабочим переменным AY0 и AZ0. В конце обсчета точки новые значения потоков AY1 по Y и AZ1 по Z запоминаются в массиве AZ0m.

Для распараллеливания межгрупповых переходов в подпрограмме **exsrce** все используемые массивы моментов хранятся в однообразном виде с выделением памяти на слой по Z по максимуму. Кроме того, индексные переменные MZ0, MZ1 вычисляются по формулам, не зависящим от порядка перебора индексов. Например

```
MZ1=ndl1*NTOTAL*(IZ-1),
```

где ndl1 – максимальное количество моментов в точке конечноразностной сетки.

В подпрограмме свертки convol – параметр подпрограммы IBEG, служащий для индексации и модифицируемый в программе, замещен на переменную IBEGPAR. IBEGPAR вычисляется независимым от порядка перебора слоев способом

```
IBEGPAR=IBEG+(IZ-1)*NTOTAL*ndl1
```

Для реализации конвейерного алгоритма в программе KINXYZ представим работу конвейера при проходе снизу вверх. Вместо последовательного перебора всех строк всех “своих” плоскостей заставим первый процессор просчитать IS строк блока всех “своих” плоскостей. Передадим соответствующие входящие излучения для этих IS строк на верхний процессор. Запустим процессор 2 для IS строк “своих” плоскостей. Первый процессор тем временем просчитывает следующую порцию IS строк “своих” плоскостей и отсылает верхнему процессору. Так же поступает и верхний процессор. Вычисления распространяются по конечно-разностной сетке “диагонально”.

Таким способом можно частично совместить работу Nproc= NBLOCK процессоров и добиться ускорения параллельного счета программы. Распространение вычислений показано на рисунке 1. Штриховкой обозначены обработанные строки блоков слоев.

Собственно распределение данных выполняется на FORTRAN-DVM очень просто с

помощью псевдокомментариев. На FORTRAN-DVM достаточно написать

CDVM$ DISTRIBUTE(*,*,*,*,*,block) :: AZ0m и массив будет автоматически распределен по процессорам. Для распределения вычислений в FORTRAN-DVM достаточно перед распараллеливаемым циклом поставить директивы- псевдокомментарии

```
CDVM$ PARALLEL (IZ,L) ON
AZ0m(*,*,L,IZ)
CDVM$+,ACROSS(AZ0m(0:0,0:0,1:0,1:0))
DO 47 IZ=1,NZ
DO 47 L=1,JMAX
```

Ключевое слово ACROSS(AZ0m(0:0,0:0,1:0,1:0)) задает конвейерное выполнение параллельного цикла указанным в предыдущем разделе способом. Пары 1:0,1:0 сообщают конвертору FORTRAN-DVM о прямой зависимости по данным для индексов L,IZ. Вычисления для индексов L,IZ зависят от полученных ранее результатов (входящих потоков) для индексов L-1,IZ-1.

Система поддержки FORTRAN-DVM на этапе параллельного счета программы замерит времена пересылки данных и счета одной строки и выберет шаг конвейера IS.

Временные характеристики двух вариантов параллельной кинетической задачи KINXYZ измерялись для различных участков программы. В частности замерялось время счета вцелом, время счета цикла внешних итераций, время счета основного цикла (перебор направлений).

При хранении моментов для всех групп в памяти нельзя пускать задачу на малом количестве процессоров. По техническим причинам на МВС15000 удалось получить результат, начиная с 16 процессоров. Кроме нехватки памяти сложность состояла с вхождением в счет задач с очень большим временем счета.

Для задачи KINXYZ разработан универсальный заголовок цикла конвейера, работающий как в системах с распределенной памятью для DVM, так и в системах с общей памятью на OpenMP.

Имея программу с основным циклом без зависимости или с минимальной зависимостью по данным, относительно нетрудно, расставляя директивы компилятору, приспособить ее для счета в среде OpenMP. При переходе от MPP-системы к SMP-системе количество служебных операций системы поддержки уменьшится, так как отпадет необходимость пересылки данных между распределенными памятями процессоров.

С другой стороны, если в системе с распределенной памятью все переменные

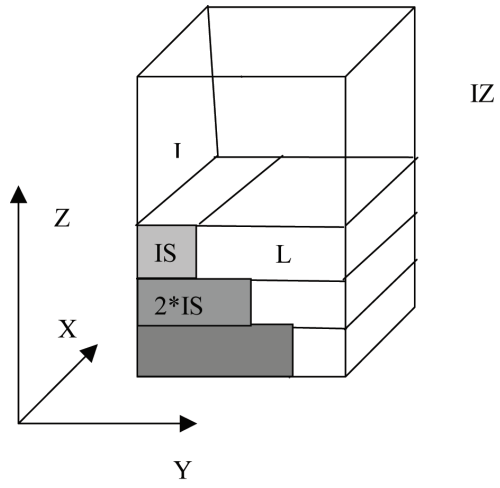

Рис. 1. Работа конвейера.
Проход снизу вверх:
I, L, IZ – индексы соответственно по осям X, Y, Z;
IS – шаг конвейера.

для каждого процессора локальны, то в SMP-системе большинство переменных по умолчанию общи для всех ядер. Общность переменных для всех ядер может вызывать неправильную работу и авосты.

Кроме того, программный код для всех ядер тоже общий. Если все ядра одновременно обратятся к одной подпрограмме, подпрограмма чаще всего выдаст неправильный результат или авост, поскольку все локальные переменные программы будут использоваться всеми ядрами одновременно. Правильную работу программы можно обеспечить, задав компилятору опцию повторной входимости в подпрограмму.

Для правильной работы программы необходимо проанализировать её параллельные участки на предмет использования переменных. Если ядра обрабатывают части общего массива данных или переменная содержит используемые константы, то такие переменные должны быть описаны как shared (разделяемые, общие). Переменные, служащие в качестве индексных и рабочих, требуется описать как private (частные, локальные).

Подпрограммы, вызываемые на параллельных участках, необходимо скомпилировать с обеспечением повторной входимости, включая все подпрограммы в дереве вызова.

Ниже приведен пример параллельного цикла с псевдокомментариями для OpenMP.

```
DO 12 NG = 1,NGROUP
C$OMP PARALLEL DO DEFAULT(NONE)
C$OMP& PRIVATE
(IZ,I,K,KV,KF0,NR,USF)
C$OMP& SHARED (NZ,NTOTAL,QF,FLU
```

Таблица 1

Времена счета и эффективность распараллеливания задачи KINXYZ на МВС15000. Вариант алгоритма с конвейерным распараллеливанием основного цикла. Массивы потоков в памяти.

Кол. процессоров	Время счета (сек)	Эффективность распараллеливания	Ускорение
16	20704	63	10
32	12984	48	15,4
48	13245	32	15,3

```
X,USFRD,USIGF,KARTA,KVZ,NG)
DO 13 IZ=1,NZ
KV=KVZ(IZ)
DO 14 I = 1,NTOTAL
KF0 = KARTA(I,KV)
IF(KF0.GT.600) THEN
      NR=KF0-600
            USF=USFRD(IZ,NR,NG)
            ELSE
            USF=USIGF(KF0,NG)
END IF
DO 15 K=1,6
QF(K,I,IZ) = QF(K,I,IZ) +
USF*FLUX(K,I,IZ,NG)
15 CONTINUE
14 CONTINUE
13 CONTINUE
12 CONTINUE
```

По директиве C$OMP PARALLEL DO DEFAULT(NONE) система поддержки **OpenMP** обеспечит выполнение лежащего ниже цикла, распределяя работу между ядрами. Разные ядра будут выполнять тело цикла с разными непересекающимися наборами значений переменной цикла.

Рабочие переменные KV, KF0, NR, USF должны быть описаны как приватные. Приватными должны быть и индексные переменные IZ,I,K вложенных циклов. Если не писать DEFAULT(NONE), то будут действовать правила умолчания SHARED – PRIVATE для переменных. В частности, индексная переменная параллельного цикла по умолчанию PRIVATE. Чтобы избежать недоразумений, можно отменять действие правил SHARED – PRIVATE по умолчанию, задавая клаузу DEFAULT(NONE) и перечислять явно все задействованные переменные в клаузах SHARED – PRIVATE.

В общем случае, при передаче управления в программе в область, C$OMP PARALLEL, все переменные, отмеченные как PRIVATE, будут размножены в количестве, достаточном для обеспечения каждого ядра своей копией переменной. Значение вновь образованных копий переменной будет в общем случае неопределенным, хотя можно обеспечить и инициацию клаузой FIRSTPRIVATE. Размножаются не только простые переменные, но и массивы, что может вызвать нехватку памяти. Это следует учитывать при написании параллельных участков.

Реализация параллельного алгоритма перебора возможна разбиением на подобласти и организацией автономных итераций внутри подобластей на MPP системах. В итерационных циклах по конечно-разностной сетке для вычисления новых значений массивов различных физических величин требуются значения с соседних точек. Если соседние точки хранятся на другом процессоре, необходима

Таблица 2

Времена счета и эффективность распараллеливания задачи KINXYZ на RSC4. Вариант алгоритма с конвейерным распараллеливанием основного цикла. Массивы моментов в памяти.

Кол. процессоров	Время счета (сек)	Эффективность распараллеливания	Ускорение
1	157260	100	1
4	53216	88	3
8	25353	81	6,2
16	13997	66	11,2
32	10239	46	15,3
60	9661	26	16,3
80	9060		17,4

Таблица 3

Времена счета первой группы KINXYZ на Intel Q6600. Вариант с OpenMP.

Кол. процессоров	Время счета (сек)	Эффективность параллельного выполнения (%)	Ускорение
1	3455	100	1
2	1760	98	1,9
3	1180	97	2,9
4	897	96	3,8

их пересылка. Процесс передачи требуемых соседних данных с соседнего процессора в DVM-системе производится автоматически. Принятый в задачах пакета РЕАКТОР способ представления массивов оставляет один вариант их распределения по процессорам, показанный на рисунке 2. В этом примере конечноразностная сетка состоит из 24 слоев и распределена на 3 процессора. Выделены граничные плоскости, копии которых хранятся на двух процессорах. Стрелками показаны направления передачи граничных плоскостей процессором-хозяином соседнему процессору. Граничные плоскости, называемые теневыми гранями, используются при итерационных прогонах по сетке. Явным образом задаются два индекса – индекс плоскости IZ и индекс по плоскости, вычисляемый на основе индекса строки L и индекса в строке I.

В рассматриваемой задаче максимальное количество задействованных в расчете процессоров ограничено количеством горизонтальных слоев в расчетной сетке метода конечных разностей. Циклы нестационарной задачи не имеют зависимостей по данным и легко распараллеливаются добавлением соответствующих директив.

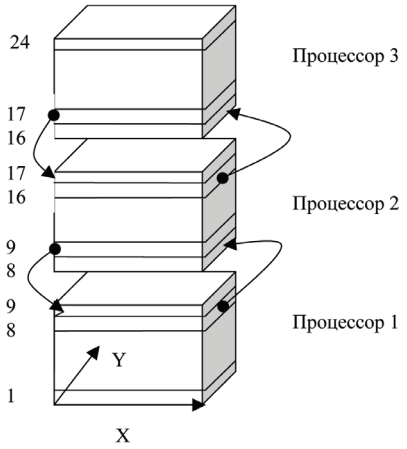

Рис. 2. Распределение конечно-разностной сетки из 24 слоев между 3 процессорами блоками по 8 слоев.

После распределения массивов по процессорам каждый процессор обрабатывает в циклах, объявленных параллельными директивой CDVM$ PARALLEL (IZ) ON QF(*,*,IZ), не массивы целиком, а выделенные ему блоки плоскостей. Для этого полный диапазон индексов IZ подменяется вырезками, соответствующими размещенному блоку.

Пропорционально доле блока в целом массиве уменьшается и работа на параллельных циклах, выполняемых на одном процессоре.

В SMP системах распределять между процессорами надо только вычисления. Эффективность параллельных вычислений уменьшается с ростом межпроцессорного взаимодействия.

В каждой подобласти выполнялось несколько локальных внутренних итераций. Граничными значениями для автономных внутренних итераций служили входящие излучения по оси Z от соседних подобластей. В начале счета эти излучения принимались нулевыми. После автономных итераций производился обмен между подобластями новыми значениями входящих излучений.

Входящее излучение каждой точки конечно-разностной сетки кроме геометрических координат имеет еще два параметра. Для правильной работы программы массив со значениями выходящего излучения AZ0m должен быть значительно большего размера, чем при конвейерной реализации. Вызвано это тем, что каждый перебор по сетке подобласти выполняется при разных сочетаниях независимых параметров, номера направления NN, номера квадранта ID(1-4), индекса полусферы (верхняя, нижняя, присутствует неявно как прогон сверху вниз и снизу вверх). Общее количество индексов массива исходящих излучений оказывается равным шести AZ0m(2,4,Nmax,Imax,Lmax,NZ). Первый индекс соответствует индексу полусферы (верх, низ). Второй индекс соответствует номеру квадранта ID. Третий – номеру направления NN. Последние три индекса – геометрические координаты.

Количество итераций по подобласти выбрано фиксированным для минимизации

Таблица 4

Времена счета первой группы KINXYZ на узле кластера МВС50к. Вариант с OpenMP.

Кол. процессоров	Время первой группы (сек)	Эффективность параллельного выполнения (%)	Ускорение
1	2639	100	1
4	720	92	3,7
8	447	74	5,9

межпроцессорного взаимодействия. Точность оценивается по максимальной разнице массивов моментов Rmomc двух последовательных итераций по всем подобластям.

В начале обсчета группы выполняется одна автономная итерация, после чего процессоры обмениваются входящими потоками AZ0m. В дальнейшем выполняется по 5 автономных итераций. Числа итераций 1 и 5 являются параметрами и могут меняться.

Параллельному варианту задачи с подобластями требуется больше внутренних итераций для достижения той же точности, что для варианта с конвейером. Необходимое количество внутренних итераций также увеличивается с ростом числа подобластей (процессоров).

Для кластеров с ограниченным размером оперативной памяти (МВС1000/RSC4) сделаны варианты программы с параллельным переносом энергетических групп, но с хранением массивов моментов не в памяти, а в архиве и в файлах ОС (отдельный вариант).

Конечной целью распараллеливания является ускорение получения результатов с требуемой точностью. На это влияет процесс замедления сходимости с ростом количества подобластей. С целью выяснения этого влияния были пропущены на разном количестве процессоров обсчеты двух первых групп. Замедление сходимости оценивалось по первой группе.

Для менее 8 процессоров характеристики получить не удалось. Для 8 процессоров задача была выброшена из счета с диагностикой отсутствия памяти после обсчета 1 группы (таблица 5).

В столбце 2 приведены суммарные количества внутренних автономных итераций для первой группы, необходимые для достижения заданной точности. В столбце 3 указано отношение числа требуемых итераций к числу итераций на одном процессоре(без подобластей). Столбец 5 – время обсчета 1 и 2 групп. Столбец 6 – ускорение по сравнению со счетом на одном процессоре без подобластей. Ускорения измерялись для 2 группы.

Для использованных исходных данных вариант с конвейером показывает лучшие результаты.

Распараллеливание кинетической задачи по направлениям

У алгоритма конкретной программы может иметься особенность, которую выгодно использовать при распараллеливании. Такой особенностью кинетической задачи является цикл по направлениям. Цикл по направлениям является основным циклом программы, внутри которого находится цикл прохода по конечно-разностной сетке.

Отличительной особенностью цикла по направлениям является то, что направления полностью независимы друг от друга (цикл без зависимости по данным). Данные, полученные для одного направления, не используются при расчете других направлений.

Таблица 5

Результаты счета 2х групп задачи KINXYZ на разных количествах процессоров. Вариант распараллеливания с подобластями. Кластер МВС15000.

1 Кол. процессоров	2 Внутренних итераций	3 Замедление сходимости	5 Время счета(сек)	6 Ускорение
1	43	1		1
4	46	1,07		
8	51	1,19	2495	6,14
16	71	1,65	1835	8,19
32	101	2,35	1563	10,4

Таблица 6

Времена счета KINXYZ. Распараллеливание по направлениям. Количество направлений 10. Кластер МВС15000.

Кол. проц.	Время счета (сек)	Эффективность параллельного выполнения (%)	Ускорение
5	55288	65	3,25
10	41059	44	4,4

Выполнение такого цикла легко распределить по разным процессорам. Затем полученные на разных процессорах массивы угловых потоков необходимо, просуммировав, объединить.

Преимущество такого способа распараллеливания в минимальности переделок исходной программы. Недостаток такого способа - в малом числе направлений и, соответственно, в малом числе задействованных процессоров. Кроме того, на системах с распределенной памятью (MPP) суммирование больших массивов моментов с разных процессоров занимает заметную долю во времени счета. На персональных многоядерных компьютерах (SMP) такой вариант может представлять интерес.

Полученные результаты (таблица 6) достаточно скромные. Параллельно работает только основной цикл. Цикл переноса, второй по расходу времени, работает последовательно. Относительно долго работает суммирование больших, порядка 100 Мб массивов моментов со всех процессоров.

Заключение

Создана параллельная версия пакета РЕАКТОР. Получены хорошие показатели эффективности параллельного счета основных расчетных модулей пакета. FORTRAN-DVM показал себя удобным средством для автоматизации распараллеливания на кластерах, а OpenMP - на многоядерных ПК. Даны практические рекомендации по созданию параллельных программ. Ускорение счета кинетической защитной задачи KINXYZ на кластере достигнуто до 17 раз на реальных данных на 80 процессорах. На 16 процессорах ускорение порядка 10. На широко доступных ныне ПК с многоядерными процессорами ускорение составляет 3,7 на 4 ядрах и 5,9 на 8 ядрах. В работе обосновывается, что создание параллельных программ требует значительно больших трудозатрат по сравнению с последовательными.

Работа выполнена при частичной финансовой поддержке Российского Фонда Фундаментальных исследований (07-01-00599-а «Комбинированные методы решения трехмерных кинетических уравнений на нерегулярных сетках и параллельные алгоритмы»).

Список литературы

1. А.В.Воронков, В.И.Аржанов, “Принципы построения пакета РЕАКТОР”. Препринт ИПМ РАН № 2, Москва, 1995
2. Konovalov N.A., Krukov V.A., Mihailov S.N. and Pogrebtsov A.A. Fortran DVM - a Language for Portable Parallel Program Development. Proceedings of Software For Multiprocessors & Supercomputers: Theory, Practice, Experience. Institute for System Programming, RAS, Moscow, 1994.
3. DVM-система, (http://www.keldysh.ru/dvm)
4. В.Е. Денисов, В.Н.Ильяков, Н.В.Ковалева, В.А.Крюков. «Отладка эффективности DVM-программ», Препринт ИПМ им. М.В.Келдыша РАН №74, 1998.
5 Н.А. Коновалов, В.А. Крюков, А.А. Погребцов, Ю.А. Сазанов, «C-DVM – язык разработки мобильных параллельных программ», препринт № 86, 1997.
6. http://www.jscc.ru
7. http://www.kiam.ru/MVS/resourses/
8. http://www.openmp.org/

Experience in use of parallel computing for nuclear reactor mathematical simulation is summarized in this paper. The most available clusters and multi-core computers were applied as parallel systems. FORTRAN-DVM and FORTRAN-OpenMP were used as code parallelization tools. The paper provides recommendations for development of parallel codes, characteristics of test problem run and efficiency factors of parallel computing.