РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ
2003, том 39, № 3

УДК 519.682.7 + 519.685 + 519.63

В. Э. Малышкин, А. А. Цыгулин
(Новосибирск)

ParaGen – ГЕНЕРАТОР ПАРАЛЛЕЛЬНЫХ ПРОГРАММ, РЕАЛИЗУЮЩИХ ЧИСЛЕННЫЕ МОДЕЛИ

Представлены базисные идеи и алгоритмы генерации параллельных программ для численного моделирования больших задач. Генерация широкого круга программ основана на использовании хорошего вручную разработанного кода, сборочной технологии параллельного программирования, параметризации и макрогенерации. Показано, что сумма этих технологических приемов обеспечивает высокое качество генерируемых параллельных программ и освобождает пользователя системы от программирования процедур синхронизации, динамической балансировки и других, трудных в реализации и отладке, элементов разработки параллельных программ. Подход реализован в системе ParaGen.

Введение. Хорошо известно, что разработка программного обеспечения – сложная проблема, плохо поддающаяся формализации. Это особенно верно для параллельного программирования, так как параллельная программа должна обладать многими специфическими свойствами: настраиваемостью на доступные ресурсы, динамической балансировкой загрузки и т. д. Для преодоления этих сложностей и создания инструмента для конструирования параллельных программ обычно используется сужение предметной области. Мы ограничились программами численного моделирования. Цель работы – построение системы для решения широкого круга задач численного моделирования на прямоугольных сетках с использованием сборочной технологии и элементов синтеза программ. Сборочная технология – результат нашего опыта разработки параллельных программ численного моделирования больших задач: метода частиц в ячейках, многосеточного метода моделирования задач физической химии и т. д.

1. **Общая идея генератора.** Существует много разных подходов к автоматизации параллельного программирования высокого уровня, которые ранжируются от языков параллельного программирования до автоматического синтеза параллельных программ. Генератор ParaGen занимает в этом ряду некоторое промежуточное положение: в нем есть и элементы синтеза программ, и языковые средства программирования. Разработка базируется на простой прагматической идее многократного использования хорошей, профессионально разработанной, программы. Эта идея в общем случае не очень продуктивна, но численные модели, будучи разными содержательно, часто очень схожи по управляющей структуре и структурам данных реализующих

их программ. По этой причине прикладная программа численного моделирования после сравнительно небольших, специально спланированных, модификаций может быть применена для решения содержательно совсем непохожей задачи.

Эта идея и эксплуатируется генератором:

1. Берется хорошая программа численного моделирования. Генератор программ должен быть в состоянии утилизировать эту программу, правильно ее модифицировать с тем, чтобы можно было применять для решения других задач, программы решения которых сходны по структуре.

2. Генератор, используя свою библиотеку процедур и фрагменты кода, специфические для вновь решаемой задачи, которые должен предоставить генератору пользователь, собирает необходимую новую программу [1, 2].

2. **Обобщенная схема программы численного моделирования.** Прежде всего рассмотрим общую схему программы, реализующей численную модель. Реальное физическое пространство представляется в памяти компьютера пространством моделирования (ПМ), т. е. той или иной трехмерной прямоугольной сеткой (прямоугольный параллелепипед на рис. 1), на которой дискретизируются поля (электрическое, магнитное и другие), температура, концентрация, плотность, давление и т. д. Кроме того, ПМ может содержать и другие дополнительные объекты, например частицы [3–5].

Сетка рассматривается как частично упорядоченный набор ячеек (см. рис. 1). Каждая ячейка содержит переменные (электрические и магнитные поля, давление и т. д.), действующие внутри ячейки. Чем больше число ячеек, тем выше точность представления дискретизированных на сетке перемен-

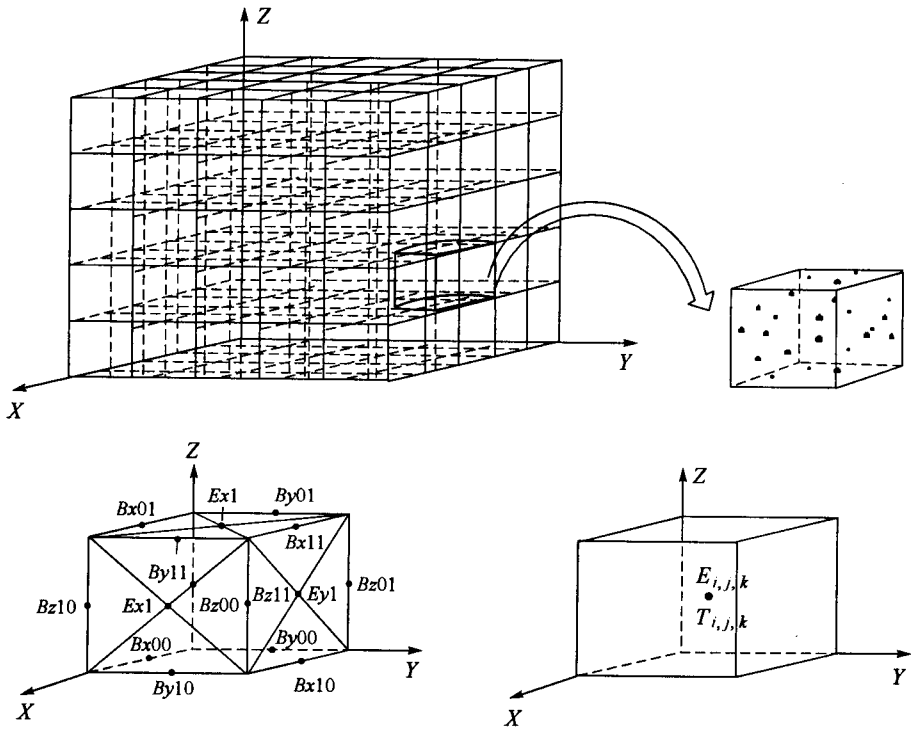

Рис. 1. Пространство моделирования и ячейка сетки

ных. Значения переменных в любой точке внутри ячейки рассчитываются по некоторой интерполяционной формуле.

Каждая ячейка может содержать набор частиц. Каждая частица имеет собственные координаты внутри ПМ. Под действием некоторых сил частицы перемещаются внутри ПМ, и их перемещение может изменять значения переменных ПМ [5].

Процесс моделирования – ряд повторяющихся вычислительных шагов, каждый из которых перевычисляет значения некоторых переменных во всех ячейках ПМ в соответствии с определенным алгоритмом. Перевычисление переменных ячейки на текущем шаге может использовать значения переменных предыдущего шага как этой ячейки, так и соседних ячеек.

Процесс моделирования схематически показан на рис. 2, *a*. Идея распараллеливания рассматриваемых численных моделей заключается в следующем. В соответствии со сборочной технологией [2, 6] ПМ собирается из достаточно малых атомарных фрагментов (ячеек для численных моделей) (рис. 2, *b*). Атомарные фрагменты связаны посредством переменных для передачи данных.

Фрагментированная структура прикладной параллельной программы сохраняется в исходном коде и во время выполнения и обеспечивает возможность организации гибкого и эффективного исполнения собранной программы. Общая идея такова: если атомарные фрагменты достаточно малы, тогда из этих фрагментов (вычислений внутри ячейки) для каждого процессорного элемента (ПЭ) мультикомпьютера собирается одинаковая нагрузка.

Собранная программа выполняется внутри каждого ПЭ мультикомпьютера, в цикле по всем фрагментам ПМ, назначенным на обработку этому ПЭ. При реализации эти фрагменты собираются в большие регулярные блоки, чтобы достичь высокой производительности программы.

Нагрузка ПЭ может измениться в ходе моделирования, и если по крайней мере один ПЭ становится перегруженным, тогда часть атомарных фрагмен-

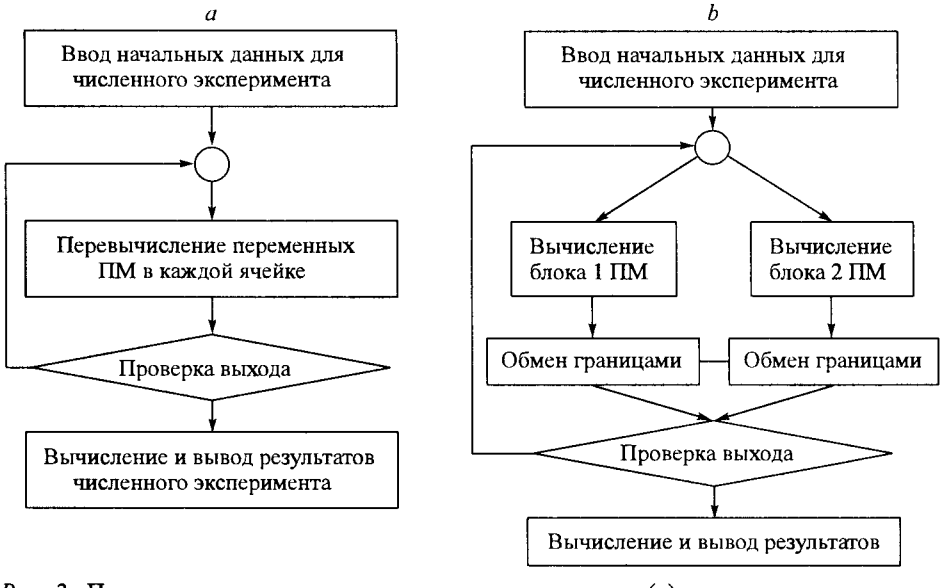

Рис. 2. Процесс моделирования: численное моделирование (*a*), стратегия распараллеливания (*b*)

тов, которые были назначены для выполнения в перегруженном ПЭ, должны «перелететь» в соседний недогруженный ПЭ, уравнивая загрузку мультикомпьютера. Динамическое балансирование загрузки мультикомпьютера основано на такой фрагментации. Конечно, это – только общая идея, в генераторе она реализована должным образом.

3. **Принципы реализации генератора.**

1. *Разработка прикладной программы.* Прежде всего разрабатывается и отлаживается программа численного моделирования. Модель гарантированно должна работать.

2. *Разработка темплейта.* Далее работает профессиональный параллельный программист. Его задача – разобрать последовательную программу на семантические фрагменты, собрать из этих заготовленных и вновь разработанных необходимых фрагментов и параллельных управляющих структур параллельную программу (темплейт), реализующую ту же численную модель. Делается это в том же стиле, как описано в разд. 2.

3. *Разработка скелетона.* Скелетон – это параметризованный темплейт. Из него удалены все фрагменты кода, характеризующие конкретную модель. Свободные позиции содержат описание кода, который должен быть вставлен для реализации другой модели. Скелетон включается в библиотеку генератора, которая может содержать более одного скелетона.

4. *Работа с ParaGen.* Теперь может быть сконструирована новая параллельная программа. Сначала конечный пользователь (математик, физик и т. д.) выбирает подходящий скелетон в библиотеке генератора. Просматривая свободные позиции скелетона, пользователь вставляет в них фрагменты кода, соответствующие проектируемой задаче. Например, вычисления внутри ячейки сетки не включаются в скелетон. Для решения конкретной задачи пользователь должен дать генератору необходимый фрагмент кода. Это очень простой последовательный код. Сгенерированная программа будет обладать всеми необходимыми динамическими свойствами: быть исполнимой на мультикомпьютерах, настраиваемой на доступные ресурсы, обеспечивать динамическую балансировку загрузки.

В соответствии с нашими взглядами на численное моделирование схема реализации модели всегда должна состоять из шага подготовки, главного цикла и шага завершения. В свою очередь, главный цикл состоит из ряда вычислений по всему ПМ.

В процессе разработки нового скелетона семантические фрагменты программы параметризуются, преобразовываются в модули и включаются в библиотеку генератора. Свободные позиции скелетона частично заполнены модулями программы, структурами данных из библиотеки. Обычно скелетон содержит схему управления, структуры данных, распределение данных, балансировку загрузки, процедуры визуализации и вспомогательные процедуры, которые всегда используются в выбранном классе приложений (рис. 3).

Для различных классов приложений формируются различные скелетоны. Все компоненты скелетона объединены при помощи унифицированных интерфейсов. Это означает, что любой компонент скелетона может быть заменен независимо. Библиотека таких компонентов может быть расширена включением компонентов, используемых для решения новой проблемы.

Для описания ПМ и семантических процедур, которые могут быть включены в скелетон, был разработан специальный язык ParaLang, основанный на языке C++. В язык C++ были добавлены несколько специальных типов

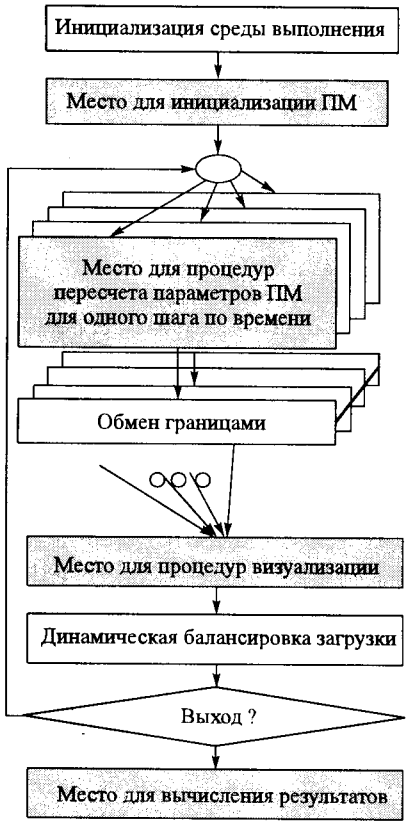

Рис. 3. Схема скелетона

данных и операторов для их описания (для определения сеточных переменных и их свойств: типов, координат и т. д.), а также некоторые специальные операторы (циклы, макросы и т. д.) для описания вычислительного алгоритма. Генератор и язык скрывают от конечного пользователя детали сложной параллельной реализации структур данных, их распределения по процессорным элементам и т. д. Вообще говоря, пользователь не должен знать о том, как реализованы сложные свойства параллельной программы (динамическая балансировка загрузки, доступ к данным и т. д.). Для обеспечения всех необходимых свойств процесса генерации программ ParaGen реализован как макрогенератор с языком периода генерации и древовидной структурой сохраняемой информации.

4. **Схема реализации ParaGen.** Схема реализации и использование генератора ParaGen демонстрируются на примере разработки параллельной версии программы «Кармен» и включения нового скелетона в ParaGen. Программа «Кармен» реализует численную модель явлений, описываемых уравнениями реакции–диффузии [4]. Решение уравнений основано на адаптивном многосеточном методе [3].

4.1. *Численная модель.* В работе рассматривается только вычислительная структура модели, физические и численные аспекты остаются за пределами рассмотрения. Более детальное описание метода может быть найдено в [3, 4, 7, 8].

Уравнения в частных производных, описывающие модель, дискретизируются в соответствии с классическим методом конечных объемов. ПМ собрано из ячеек, каждая из которых содержит среднее значение дискретизируемых функций. Численное решение таких задач обычно требует малого временного и пространственного шагов для достижения приемлемой точности и стабильности. Однако во многих случаях малый пространственный шаг требуется только в небольшой подобласти, например, для решений, имеющих фронт или зоны химической реакции. Это означает, что для обеспечения необходимой точности представления дискретизируемой функции сетка должна быть локально (в некоторых подобластях) сгущена. Степень такого сгущения зависит от начального состояния ПМ и может меняться в процессе моделирования.

В соответствии с многосеточным методом дискретизуемые на мелкой сетке функции могут быть представлены как значения на более грубой сетке плюс ряд уточняющих дифференциалов. Дифференциалы содержат информацию об отличиях значений функций от их усредненного на более грубой

сетке значения, и, если эти различия небольшие, ими можно пренебречь (считать равными нулю) в подобластях, где решение гладкое [3], получая, следовательно, значительно меньшее количество узлов сетки при определении функций с неравномерной регулярностью.

На рис. 4, *a* представлена древовидная структура, представляющая ПМ. Дерево составлено из начальной ячейки, являющейся его корнем, узлов (белые ячейки на рис. 4, *a*) и листьев (серые ячейки), составляющих крону дерева. Для вычисления потока (перемещения энергии или вещества «в» и «из» ячейки) требуется информация от смежных листьев на том же самом уровне измельчения. Поэтому создаются виртуальные листья (заштрихованные ячейки). Они используются только для вычисления потоков и не участвуют в вычислениях, связанных с итерациями по времени. В трехмерном пространстве ячейка на уровне l всегда имеет $2^3 = 8$ потомков на уровне $l+1$.

Сетка первоначально состоит из единственной (начальной) ячейки (рис. 4, *b*), которая определяет физический размер и форму ПМ. Процедура деления, исполненная некоторое количество раз для листьев, производит несколько уровней разбиения ПМ. Обычно после того, как построено пять–шесть уровней разбиения, достигается требуемая точность представления параметров в некоторых подобластях ПМ. Ячейки в этих подобластях прекращают делиться. В подобластях, где дискретизируемые функции изменяются слишком быстро, деление продолжается далее. Таким образом, ПМ имеет различную глубину разбиения ячеек в зависимости от локальных свойств дискретизируемых функций.

Процесс моделирования происходит следующим образом. Предварительно в зависимости от начальных условий делением начальной ячейки

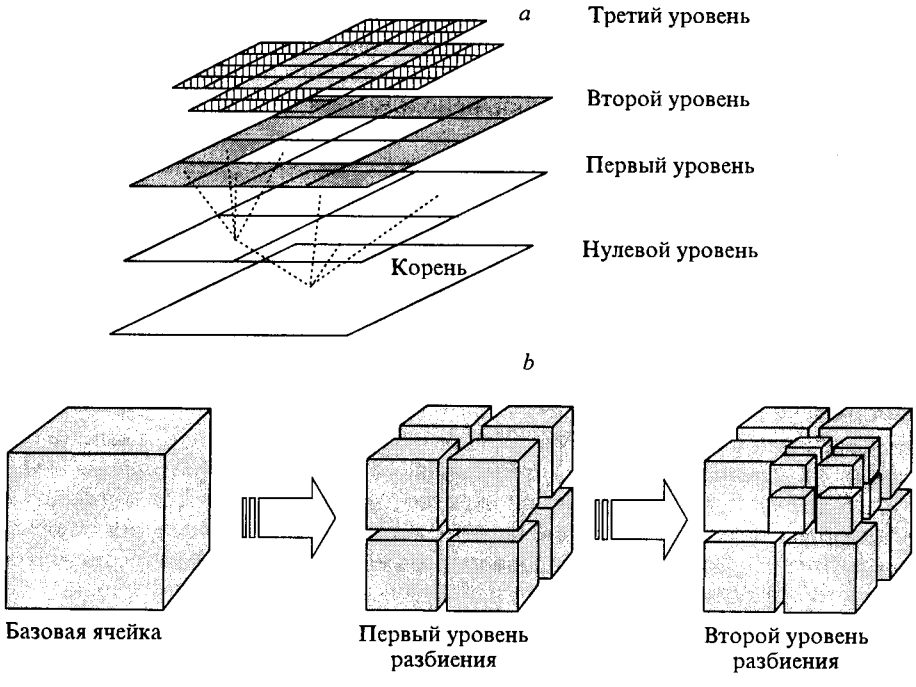

Рис. 4. Пространство моделирования, разбиение базовой ячейки

строится начальное дерево. Затем для каждого шага по времени $t_n = n\Delta t$ пересчитываются параметры ПМ, связанные с химическими процессами; рассчитываются средние значения в виртуальных листьях с использованием значений в смежных ячейках того же уровня и родительского уровня, если необходимо; методом Рунге – Кутта рассчитываются состояния ПМ для времени t_{n+1}. На каждом шаге вычисления производятся только на листьях. После этого рассчитываются локальные ошибки аппроксимации в каждой ячейке, чтобы произвести переразбиение дерева. По завершении цикла вычислений рассчитываются финальные результаты работы программы.

Переразбиение дерева – процедура деления ячеек в областях с большой локальной ошибкой или объединения смежных ячеек внутри подобласти сетки, где функция представлена достаточно точно. Операция переразбиения должна поддерживать структуру дерева, создавая или удаляя виртуальные ячейки таким образом, чтобы сохранялись отношения соседства.

4.2. *Описание последовательной программы.* Последовательная программа «Кармен», реализующая модель, была разработана, отлажена и проверена на персональном компьютере на входных данных небольшого размера [4]. ПМ первоначально представлено единственной ячейкой. В ходе моделирования ячейки рекурсивно делятся, чтобы достичь желаемой точности представления дискретизируемых функций. Каждая ячейка содержит значения физических параметров: температуру, концентрацию реактивов и давление и рассматривается как сетка с одной точкой. В таком контексте ПМ – это набор вложенных сеток. Каждый узел дерева представляет ячейку со всеми ее параметрами и вложенными ячейками. Существует система координат, чтобы перечислять ячейки. Координаты ячейки – это четверка i, j, k, l, где i, j, k – пространственные координаты, а l – уровень разбиения. Начальная ячейка имеет координаты $i = 0, j = 0, k = 0, l = 0$. Система координат ПМ очень важна, потому что с ее помощью определяется понятие смежной ячейки. Смежными считаются ячейки с координатами, отличающимися только одним пространственным компонентом и не более чем на единицу. Для вычисления переменных внутри ячейки используются переменные как той же ячейки, так и смежных ячеек.

Для описания параметров эксперимента существует несколько глобальных переменных: число шагов по времени, порог для переразбиения, вязкость и т. д.

Структура управления была извлечена из программы «Кармен», она выглядит подобно схеме программы (см. рис. 2, *а*, рис. 5), которая состоит из начального шага, цикла итераций по времени и конечного шага. Итерация по времени, в свою очередь, состоит из нескольких подшагов, каждый из которых есть цикл (от корней к листьям или наоборот) по всем ячейкам дерева в зависимости от типа.

4.3. *Разработка темплейта.* Для разработки параллельной версии программы «Кармен» (темплейта) применялась стратегия распределения ПМ [5, 6, 9]. ПМ темплейта состоит из множества начальных ячеек (базовой сетки), организованных в блок в форме параллелепипеда или нескольких смежных блоков (рис. 6). Основная идея распараллеливания программы «Кармен» состоит в том, чтобы запустить вычисления для каждой начальной ячейки базовой сетки одновременно, каждая из ячеек может быть независимо разделена для достижения желаемой точности представления дискретизируемых функций, порождая дерево. Однако нельзя использовать процедуры построения дерева последовательной программы без изменений, так как теперь от-

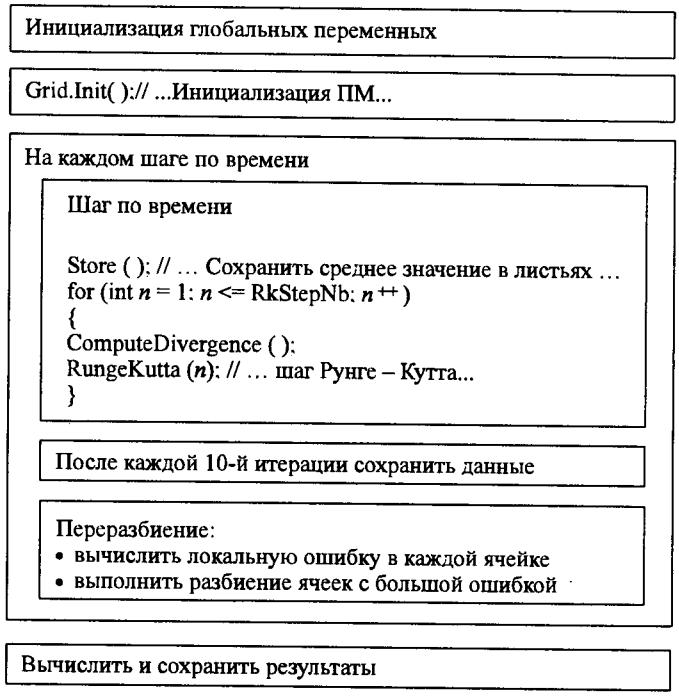

Рис. 5. Структура программы «Кармен»

ношения соседства и условия создания виртуальных ячеек затрагивают деревья, порожденные смежными начальными ячейками, что приводит к проблеме склеивания.

В ходе начальной настройки параллельной программы блоки разбиваются в соответствии с числом ПЭ. Каждый блок содержит дополнительный (граничный) слой виртуальных ячеек, значения сеточных переменных которых копируются из смежных блоков в ходе синхронизации или перевычисляются в соответствии с граничными условиями в случае, если граница блока – это граница ПМ.

Рассмотрим пример набора сеточных переменных, процедуры доступа и их использование.

Набор сеточных переменных: Q – вектор параметров, D – вектор дивергенций параметров, Qs – вектор для временного хранения параметров.

Набор функций для доступа к сеточным переменным:

```
vector & Q(int i, int j, int k, int l);
vector & D(int i, int j, int k, int l);
vector & Qs(int i, int j, int k, int l);
```

Пример использования этих функций:

```
@ForEach (Qs: i, j, k, l)//копирование
      Qs(i, j, k, l) = Q(i, j, k, l);
      Q(i, j, k, l) = Div (Q(i, j, k, l));
@ EndForEach
```

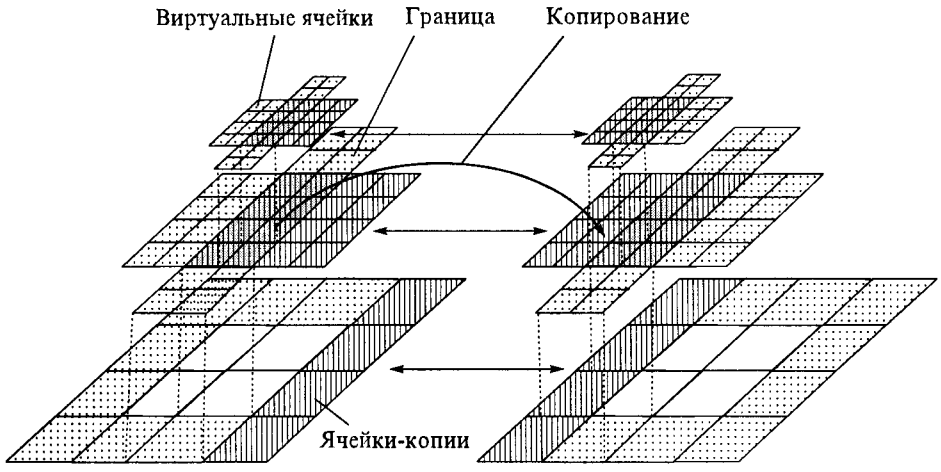

Рис. 6. Декомпозиция пространства моделирования

Формулы вычисления, представляющие соответствующий численный алгоритм, собраны в методах класса блока. Для обеспечения прозрачного распределения данных по процессорным элементам в язык описания алгоритма вычислений ParaLang введены параллельные циклы для обхода ячеек, позволяющие описывать вычислительный алгоритм в близкой к последовательной форме. Доступ к переменным сетки осуществляется с помощью специально разработанных процедур для каждой сеточной переменной.

Темплейт позволяет задавать начальные условия и параметры численных экспериментов унифицированным способом. Начальное состояние ПМ может быть задано при использовании специального подмножества языка С, что позволяет избегать перекомпиляции целевой параллельной программы для каждого эксперимента. Темплейт содержит унифицированную систему для вывода данных, отладочной и служебной информации, а также информации о состоянии ПМ в ходе моделирования. Эту дополнительную информацию можно отображать с помощью специальной программы визуализации данных, например VISA.

Параллельная программа достигает высокой скорости выполнения, если все ПЭ загружены одинаково, иначе происходит резкий спад производительности [5, 10, 11]. Загрузка ПЭ главным образом зависит от числа ячеек, назначенных ПЭ для обработки. В ходе моделирования некоторые ячейки делятся, некоторые объединяются, поэтому даже первоначально однородно загруженный мультикомпьютер может накапливать дисбаланс, и через несколько итераций по времени потребуется перебалансировка. Таким образом, структура управления темплейта и распределение данных среди ПЭ должны динамически изменяться в ходе вычислений.

Комплекс процедур балансировки обеспечивает начальное распределение данных среди ПЭ и динамическую перебалансировку загрузки в ходе вычислений, а также контроль дисбаланса и выбор порога перебалансировки. Информация о загрузке каждого ПЭ собирается параллельно с вычислениями, чтобы уменьшить задержки на межпроцессорные связи.

Для визуализации результатов и контроля процесса моделирования используется специальная программа VISA. Она позволяет контролировать любые параметры модели в процессе вычислений, используя интерфейс

файловой системы и унифицированный протокол обмена данными с генерируемой параллельной программой. Такая программа обеспечивает визуализацию результатов немедленно после их получения из программы моделирования, сбор данных от всех ПЭ и преобразование их к известному формату для других программ визуализации. Дополнительно VISA позволяет управлять начальными параметрами численного эксперимента и контролировать процесс вычислений.

Таким образом, темплейт обладает всеми необходимыми свойствами параллельной программы: настройкой на имеющиеся ресурсы, обработкой адаптивной сетки, динамической балансировкой загрузки и т. д. Кроме того, в результате использования сборочной технологии и языка ParaLang темплейт может быть легко параметризован.

4.4. *Скелетон.* После завершения разработки и отладки темплейта может быть построен скелетон.

Идея. Чтобы создать параллельную программу, способную реализовать класс моделей вместо конкретной, весь специфический для данной модели код должен быть подготовлен к замене кодом другой модели. Все такие фрагменты кода заменены макропрограммами. Эти макропрограммы описывают предопределенные позиции, которые будут заполнены вычислениями (формулами) новых численных моделей. При обработке макроса может происходить диалог с пользователем, чтобы уточнить те или иные параметры. Все фрагменты кода, зависящие от параметров численной модели (в данном случае количество и типы сеточных переменных), должны быть также заменены на соответствующие макропрограммы, которые породят те же самые служебные функции, но для другой модели и с другими параметрами. Также все параметры численной модели должны быть явно описаны. Скелетон – это параметризованный темплейт, реализующий некоторую численную модель, основанную на программе «Кармен». Ядро скелетона включает в себя большое число фрагментов кода программы «Кармен», а также процедуры склеивания и синхронизации. Процесс конструирования скелетона из подготовленного темплейта происходит следующим образом. Определяются параметры скелетона. В конкретном случае параметры скелетона – количество и типы сеточных переменных, их размещения внутри ячейки. Специальные операторы и соответствующие макросы должны быть описаны, например,

```
@ DefGrid ⟨arithmetical type⟩ ⟨name⟩ ⟨placing⟩;

@defmacro @DefGrid %n %n ⟨%d, %d, %d⟩;
  @let SM. grids. %2%. type = %1%;
  @let SM. grids. %2%. xleftoverlap = %3%;
  @let SM. grids. %2%. xrightoverlap = %5%;
  @let SM. grids. %2%. share = %4%;
@endmacro
```

Обычно эти макросы только сохраняют информацию внутри древовидной структуры данных генератора для последующего использования.

Затем численный алгоритм должен быть удален из темплейта и заменен соответствующей макропрограммой. В процессе генерации этот макрос будет обработан и заменен другим численным алгоритмом или разновидностью такого алгоритма или совсем будет не заполнен, например,

```
Void block :: TimeStep (int timestepnm)
 {
 @include @TimeStep (int timestepnm);
 }

@TimeStep (int timestepnm)
   ...
   @for (all Qs)
      Qs(i, j, k, l) = Q(i, j, k, l);
   @endfor
   ...
```

Макрос может быть также параметризован, и несколько макросов могут быть определены для той же самой процедуры, но с другими параметрами, чтобы выбрать лучший из них в процессе генерации.

Затем остальной код темплейта, который может зависеть от параметров модели, параметризуется. Для этого используются макроязык и информация, сохраненная во внутреннем дереве генератора. В нашем случае все процедуры темплейта по управлению сетками параметризованы, для этого используются циклы по всем сеточным переменным и сохраненные параметры каждой из сеток.

Заключение. Разработанный и реализованный генератор параллельных программ в состоянии обеспечить конструирование широкого класса новых высококачественных программ численного моделирования. Наиболее привлекательное свойство генератора – возможность включить в библиотеку хороший параметризованный код и широко его использовать. Генерируемая программа поддерживает и составное пространство моделирования, динамическую балансировку, адаптивные сетки. В генератор ParaGen можно включать новые скелетоны для других классов задач, используя разработанное программное обеспечение и предложенную технику. В частности, различные приложения метода частиц в физике плазмы также подходят для реализации с использованием ParaGen несмотря на значительную несхожесть численных методов. Развивать ParaGen планируется в направлении накопления библиотек скелетонов и готовых фрагментов программ для конструирования новых приложений.

СПИСОК ЛИТЕРАТУРЫ

1. **Вальковский В. А., Малышкин В. Э.** Синтез параллельных программ и систем на основе вычислительных моделей. Новосибирск: Наука, 1988.
2. **Kraeva M., Malyshkin V.** Assembly technology for parallel realization of numerical models on MIMD-multicomputers // Future Generation Computer Systems, Elsevier Science. 2001. **17**. P. 755.
3. **Harten A.** Multiresolution algorithms for the numerical solution of hyperbolic conservation laws // Communs Pure Appl. Math. 1995. **48**. P. 1304.
4. **Roussel O., Schneider K., Tsigulin A., Bockhorn H.** A conservative fully adaptive multiresolution algorithm for parabolic PDEs. Prepr. ICT. Universität Karlsruhe, 2002.
5. **Kraeva M. A., Malyshkin V. E.** Implementation of PIC method on MIMD multicomputers with assembly technology // Proc. of HPCN Europe, 1997, LNCS. Heidelberg: Springer-Verlag, 1997. Vol. 1255. P. 541.

6. **Caretti E., Messina A.** Dynamic Work Distribution for PM Algorithm // http://xxx.lanl.gov/astro-ph/0005512

7. **Dahmen W., Gottschlich-Müller B., Müller S.** Multiresolution schemes for conservation laws // Numer. Math. 2001. **88**, N 3. P. 399.

8. **Cohen A., Kaber S. M., Müller S., Postel M.** Fully adaptive multiresolution finite volume schemes for conservation laws // Technical Report R00009. Paris: Université Pierre et Marie Curie, 2000.

9. **Parashar M., Browne J. C.** On partitioning dynamic adaptive grid hierarchies // Proc. of the 29th Annual Hawaii Intern. Conf. on System Sciences. Hawaii: Maui, 1996. Vol. 1. P. 604.

10. **Corradi A., Leonardi L., Zambonelli F.** Performance comparison of load balancing policies based on a diffusion scheme // Proc. of the Euro-Par'97 LNCS. Heidelberg: Springer-Verlag, 1997. Vol. 1300.

11. **Parashar M., Browne J. C.** The HDDA/DAGH Infrastructure for Implementation of Parallel Structures Adaptive Mesh Refinement // IMA Volumes in Mathematics and its Applications. Heidelberg: Springer-Verlag, 1997.

Институт вычислительной математики и математической геофизики СО РАН,
Новосибирский государственный технический университет,
E-mail: malysh@ssd.sscc.ru,
neolp@mail.ru

Поступила в редакцию
25 марта 2003 г.