ISSN 0134—6388

АКАДЕМИЯ НАУК СССР
НАУЧНЫЙ СОВЕТ ПО КОМПЛЕКСНОЙ ПРОБЛЕМЕ
«КИБЕРНЕТИКА»

ВК

ВОПРОСЫ КИБЕРНЕТИКИ

МЕТОДЫ И АЛГОРИТМЫ АНАЛИЗА БОЛЬШИХ СИСТЕМ

МОСКВА 1988

2. *Корпелевич Г. М.* Экстраградиентный метод для отыскания седловых точек и других задач. — Экономика и матем. методы, 1976, т. XII. вып. 4, стр. 747—756.
3. *Гольштейн Е. Г.* О сходимости градиентного метода отыскания седловых точек модифицированных функций Лагранжа. — Экономика и матем. методы, 1977, т. XIII, вып. 2, стр. 322—329.
4. *Антипин А. С.* Методы нелинейного программирования, основанные на прямой и двойственной модификации функции Лагранжа. Препринт, М.: Всесоюзный НИИ системных исследований, 1979.
5. *Антипин А. С.* Декомпозиционные методы вычисления седловой точки функции Лагранжа и их применение к задачам с блочно-сепарабельной структурой. — Ж. вычисл. матем. и матем. физ., 1986, т. 26, № 1. стр. 150—151.
6. *Бакушинский А. Б.* Некоторые математические проблемы композиционного планирования. В сб. М.: Труды ВНИИСИ «Проблемы оптимизации в системных исследованиях», вып. 8, 1978, стр. 37—47.

УДК 519.63

ТЕХНОЛОГИЯ СОГЛАСОВАНИЯ НЕКОТОРЫХ АЛГОРИТМОВ С АРХИТЕКТУРОЙ ВЕКТОРНО-КОНВЕЙЕРНОЙ ЭВМ

О. С. Бацуков, И. М. Коротаев, С. А. Кутасов, М. А. Цепков

(Москва)

Рассматривается задача конструирования программ, выполняющихся с предельной скоростью на векторно-конвейерной машине (ВКМ). Оптимальное планирование вычислений достигается за счет согласования информационной структуры (ИС) алгоритма с архитектурой ВКМ типа *CRAY*. Подход иллюстрируется на примере решения одной из задач газовой динамики методом крупных частиц. Отмечаются «узкие» места в архитектуре ВКМ, «реконструкция» которых могла бы увеличить производительность и упростить программирование.

§ 1. Введение

В работе приводится краткий анализ возможных подходов к организации оптимальных вычислений на векторно-конвейерных ЭВМ.

Создаваемые в настоящее время супер-ЭВМ предназначены в первую очередь для решения выдвигаемых практикой задач, требующих быстродействия на грани возможностей вычислительной техники. Поэтому такая характеристика программ и алгоритмов, как быстродействие выходит по своей значимости на первое место.

Описывается методика конструирования программ с пре-

дельным быстродействием, основанная на структурном анализе алгоритмов [1] и на согласовании алгоритма с архитектурой ВКМ [2]. Для получения программ на ассемблере используется разработанный авторами препроцессор с языка векторной машины (см. § 6).

Подход демонстрируется на примере решения одной из задач аэрогидродинамики методом крупных частиц [3] на ВКМ типа ЭВМ *CRAY*-1. Приводятся характеристики полученной программы.

Результаты работы могут распространяться и на супер-ЭВМ других типов, т. к. архитектура ВКМ типа *CRAY* является носителем основных концепций структур супер-ЭВМ — **параллельность и конвейеризация.**

Выбор для целей иллюстрации метода крупных частиц обусловлен стремлением к общности рассмотрения, т. к. метод крупных частиц можно отнести к классу неоднородных методов, как и метод потоков, рассматриваемый в работе [5]. Такие методы, вообще говоря, считаются неэффективными при решении задач на ВКМ. Однако, в работе будет показано, что применение некоторых эвристик позволяет обеспечивать проектную производительность машины, такую же, как и для однородных схем.

Приведенный материал основан на опыте конструирования следующих программ предельного быстродействия для ВКМ: решения задач аэрогидродинамики методами потоков [5], крупных частиц, сеточно-характеристическим; алгоритмов, использующих прогонку; решения систем обыкновенных дифференциальных уравнений; задач приближения элементарных функций.

§ 2. Анализ возможных подходов к достижению высокой производительности на супер-ЭВМ

Проблемы, возникающие при использовании супер-ЭВМ как в математике, так и в программировании, были осознаны еще в 70-х годах. Их общий анализ содержится, например, в работе [6]. Впечатляющим подтверждением трудности освоения супер-ЭВМ явился тот факт, что за первые 5 лет эксплуатации ЭВМ *CRAY*-1 в США, быстродействие на программах из библиотек алгоритмов, компилируемых с фортрана, составляло 10-20% от проектного быстродействия 80 *Mgflops* (при предельном — 240 *Mgflops*).

Один из путей преодоления трудностей реализуется в настоящее время посредством внедрения векторных версий фортрана [7]. Однако рассчитывать на получение высокого быстродействия можно только в тех случаях, когда структура алгоритма хорошо соответствует архитектуре ЭВМ, что, естественно, случайное и не так уже часто реализуемое событие.

Второй путь заключается в описании алгоритма на непроцедурном языке, т. е. на языке спецификаций, в котором не содержится явного указания на последовательность операций и на распределение памяти. Например, предлагается язык для решения задач матфизики разностными методами НОРМА [8, 9]. Безусловно, такой подход снимает многие проблемы, возникающие при программировании на процедурных языках. Но компилятор с этого языка кроме реализации обычных для этого компилятора функций должен «уметь» анализировать структуру алгоритма и организовывать вычисления, что требует высокого «интеллекта» от такой системы программирования. Поэтому, даже учитывая достижения в системном программировании по трансляции и оптимизации, трудно в ближайшее время рассчитывать на реализацию компилятора, способного создавать программы с высокой эффективностью. В первую очередь необходим такой теоретический аппарат структурного анализа алгоритмов, который мог бы быть положен в основу алгоритма, реализуемого компилятором.

Наконец, третий подход — «ручной» структурный анализ алгоритма с последующим планированием вычислений [1], согласованием с архитектурой машины [2] и написанием программы на языке ВКМ; язык векторной машины может представлять собой суженный язык ассемблера в нотации, близкой к фортрану. Язык дополнен макроопределениями конструкций фортрана *DO*, *ENDDO*, *DIMENSION* и т. д., так что «ручное» программирование относится к векторным арифметическим операциям и обменам регистров и памяти, т. е. управление наиболее «узкими» местами программы полностью контролируется программистом.

Этот подход может рассматриваться как этап разработки второго подхода в части конструирования программы вместо компилятора с входного языка. Входным языком могут быть либо расчетные формулы в обычном представлении, либо язык типа НОРМА [8], который в смысле корректности описания алгоритма не отличается от расчетных формул. В настоящей работе в рамках второго и третьего подходов предлагается технология согласованного проектирования программ с высоким, близким к предельному, быстродействием.

§ 3. Описание задачи. Расчетные формулы

Для иллюстрации рассматривается следующая задача. Сверхзвуковой поток воздуха набегает на ступеньку (плоская двумерная постановка) — см. рис. 1. В области *ABCGFE* задаются начальные параметры набегающего потока. Область расчета *ABCD* окружается слоем так называемых фиктивных ячеек, в которых ставятся граничные условия. На границах тела *EF* и *FG* ставятся условия непротекания и вся область

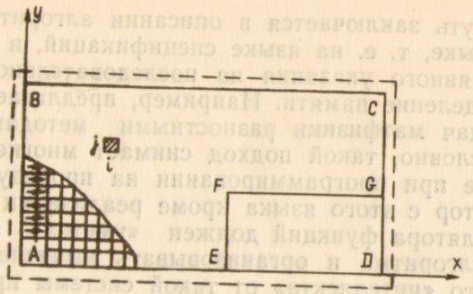

Рис. 1 Постановка задачи. Область расчета

ABCD рассчитывается «сквозным» образом. Рассматриваемый процесс обтекания описывается системой дифференциальных уравнений:

$$\begin{cases} \partial\rho/\partial t + \operatorname{div}(\rho\vec{W}) = 0, \\ \partial\rho u/\partial t + \operatorname{div}(\rho u\vec{W}) + \partial P/\partial x = 0, \\ \partial\rho v/\partial t + \operatorname{div}(\rho v\vec{W}) + \partial P/\partial y = 0, \\ \partial\rho E/\partial t + \operatorname{div}(\rho E\vec{W}) + \operatorname{div}(P\vec{W}) = 0, \\ P = P(\rho, E, u, v). \end{cases} \quad (1)$$

Область численного решения покрывается неподвижной эйлеровой сеткой с параметрами $(\Delta x, \Delta y)$. Центры ячеек обозначаются (i, y). Процесс вычисления состоит из многократно повторяющихся шагов по времени. Расчет каждого такого шага (основного вычислительного цикла) при решении системы (1) методом крупных частиц разбивается на три этапа [3]:

1) эйлеров этап — пренебрегаем всеми эффектами, связанными с перемещениями элементарной ячейки (нет потока массы через границы ячеек), в области расчета действует только поле давления. Решается «укороченная» система уравнений:

$$\begin{cases} \rho\partial u/\partial t + \partial P/\partial x = 0, \\ \rho\partial v/\partial t + \partial P/\partial y = 0, \\ \rho\partial E/\partial t + \operatorname{div}(P\vec{W}) = 0, \\ P = P(\rho, E, u, v). \end{cases}$$

Используется следующая разностная схема:

$$\tilde{u}^n_{i,j} = u^n_{i,j} - \Delta t\,(P^n_{i+1/2,j} - P^n_{i-1/2,j})/(\Delta x\cdot\rho^n_{i,j}),$$

$$\tilde{v}^n_{i,j} = v^n_{i,j} - \Delta t\,(P^n_{i,j+1/2} - P^n_{i,j-1/2})/(\Delta y\cdot\rho^n_{i,j}),$$

$$\tilde{E}^n_{i,j} = E^n_{i,j} - [(P^n_{i+1/2,j}u^n_{i+1/2,j} - P^n_{i-1/2,j}u^n_{i-1/2,j})/\Delta x + (P^n_{i,j+1/2}v^n_{i,j+1/2} - P^n_{i,j-1/2}v^n_{i,j-1/2})/\Delta y]\,\Delta t/\rho^n_{i,j}.$$

На эйлеровом этапе получаем т. н. промежуточные значения компонент скорости и полной энергии $\tilde{u}^n_{i,j}$, $\tilde{v}^n_{i,j}$, $\tilde{E}^n_{i,j}$ для всех ячеек (i, j) на временном слое t^n.

2) лагранжев · этап — находим за время Δt потоки массы ΔM^n через все четыре границы каждой эйлеровой ячейки. При этом полагаем, что вся масса переносится только за счет нормальной к границе составляющей скорости. Например:

$$\Delta M^n_{i+1/2,j} = \begin{cases} \rho^n_{i,j}\Delta y\cdot\Delta t\,(\tilde{u}^n_{i+1,j} + \tilde{u}^n_{i,j})/2, & \text{если } \tilde{u}^n_{i+1,j} + \tilde{u}^n_{i,j} \geqslant 0, \\ \rho^n_{i+1,j}\Delta y\cdot\Delta t\,(\tilde{u}^n_{i+1,j} + \tilde{u}^n_{i,j})/2, & \text{если } \tilde{u}^n_{i+1,j} + \tilde{u}^n_{i,j} \leqslant 0. \end{cases}$$

3) заключительный этап — находим окончательные значения параметров потока ρ, E, u, v на новом временном слое $t^{n+1} = t^n + \Delta t$:

$$\rho^{n+1}_{i,j} = \rho^n_{i,j} + (\Delta M^n_{i-1/2,j} + \Delta M^n_{i,j-1/2} - \Delta M^n_{i+1/2,j} - \Delta M^n_{i,j+1/2})/(\Delta x\cdot\Delta y),$$

$$X^{n+1}_{i,j} = \rho^n_{i,j}\tilde{X}^n_{i,j}/\rho^{n+1}_{i,j} + \Big(\Delta M^n_{i-1/2,j}\tilde{X}^n_{\binom{i,j}{i-1,j}} + \Delta M^n_{i,j-1/2}\tilde{X}^n_{\binom{i,j}{i,j-1}} - \Delta M^n_{i+1/2,j}\tilde{X}^n_{\binom{i+1,j}{i,j}} - \Delta M^n_{i,j+1/2}\tilde{X}^n_{\binom{i,j+1}{i,j}}\Big)\Big/(\Delta x\cdot\Delta y\cdot\rho^{n+1}_{i,j});\ X(u, v, E).$$

Индексы в круглых скобках выбираются в зависимости от знака ΔM^n на границе ячейки, т. е. в зависимости от направления потока.

§ 4. Структурный анализ алгоритма. Планирование вычислений

Алгоритм можно рассматривать как последовательно-параллельную систему явноопределенных функций. Одна из возможностей ускорения расчета в супер-ЭВМ — параллельная реализация функций параллельными или векторными устройствами.

Параллельные устройства могут одновременно реализовывать **параллельные** или независимые функции, т. е. такие функции, которые несвязны по промежуточным переменным и не являются предшественниками друг друга. Параллельные функции являются **векторными** если они **однородны** и определены на векторном пространстве переменных, такие переменные обычно имеют регулярную структуру в памяти ЭВМ. Векторные функции эффективно реализуются векторными или конвейерными устройствами ВКМ.

Основная цель анализа алгоритма — выявление векторности и взаимной зависимости структур алгоритма для дальнейшего планирования работы векторных и параллельных устройств.

В нашем рассмотрении объект для анализа — информационные структуры (ИС) алгоритма [1, 2]. Каждой вершине ИС относится имя переменной, дуги указывают зависимость переменных.

Выделение в ИС векторных несвязных функций — векторизация ИС — шаг, необходимый для планирования работы векторных и параллельных устройств [5].

Векторная часть ИС $n+1$ шага рассматриваемого в настоящей работе алгоритма решения задачи обтекания методом крупных частиц приведена в ярусно-параллельной форме (ЯПФ) [10] на рис. 2. Все функции на рис. 2 векторные. На ярусе 0 — входные переменные $n+1$ шага, на ярусе 9 — выходные, на остальных — промежуточные. Вершины, обведенные большими

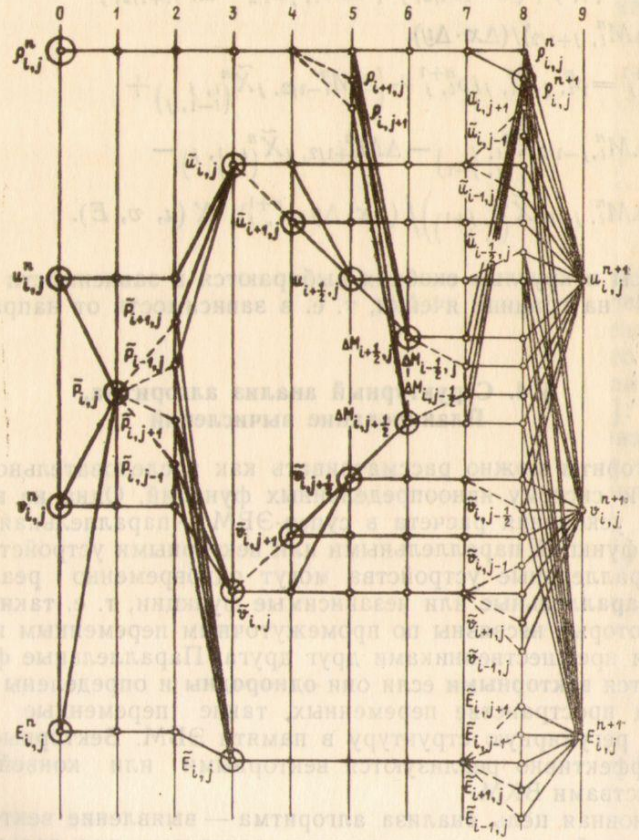

Рис. 2 Информационная структура алгоритма решения задачи обтекания методом крупных частиц

кружками соответствуют **основным** несвязным векторным функциям, остальные — производным несвязным векторным функциям, получаемым из основных без вычислений. Зависимости между основными и производными функциями описываются только **индексными** функциями.

Приведенное графическое описание алгоритма используется для «ручного» анализа.

§ 4.1. Архитектура ВКМ

Планирование вычислений проводится с учетом конкретных особенностей ЭВМ. Структура алгоритма и архитектура машины описываются в одинаковых терминах — при помощи графов ИС.

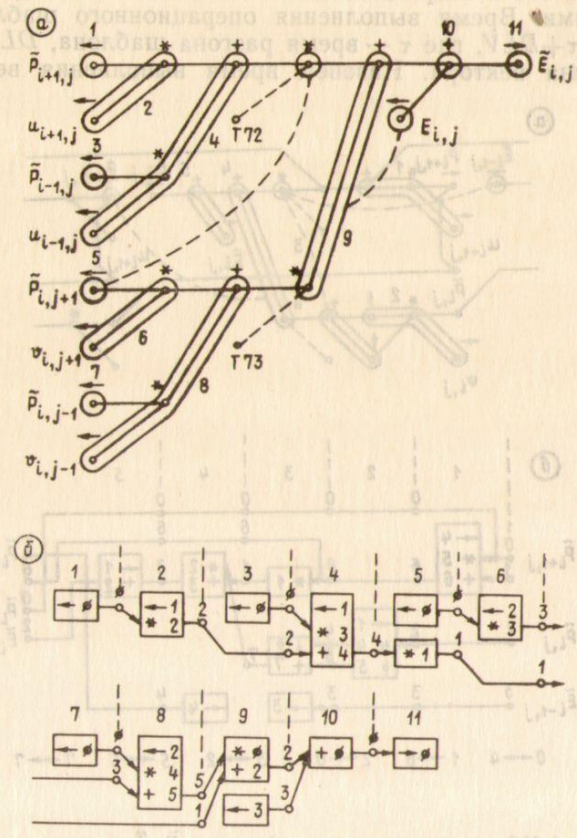

Рис. 3 ИС Функции $E^{n}_{i\,j}$ без применения эвристик

Основные особенности системы команд и архитектуры рассматриваемой абстрактной ВКМ (подробности можно найти в [4]):

— три группы функциональных устройств — векторные, скалярные, адресные — которые могут работать параллельно;

— векторные устройства могут работать с «зацеплением», в таком режиме через каждый такт появляется результат выполнения сразу нескольких операций;

— устройства работают с регистрами — векторными, скалярными и адресными (по 8 регистров в каждой группе).

Назовем векторными операционными шаблонами V-ОШ набор информационных структур элементарных алгоритмов, которые реализуются параллельно или с зацеплением. На рис. 3 и 4 V-ОШ на ИС в Е-разложении [1] обозначены в частях а) овалами, а на ИС в операционных шаблонах в частях б) — прямоугольниками. Время выполнения операционного шаблона составляет $\tau+DLV$, где τ — время разгона шаблона, $DLV=1—64$ $\xi 64$ — длина вектора. Назовем время выполнения векторного

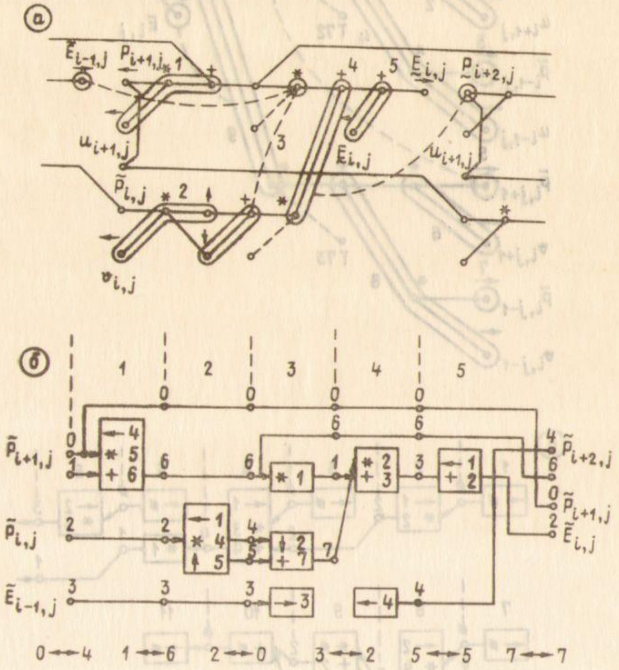

Рис. 4 ИС одного такта цикла функции $\tilde{E}\ \frac{n}{il}$ с оптимальным покрытием ОШ (время выполнения 5 VT)

ОШ векторным тактом (VT). Параллельно с V-ОШ могут выполняться скалярные S-ОШ и адресные А-ОШ операционные шаблоны — такие наборы ИС скалярных и адресных алгоритмов, у которых нет конфликтов по регистрам с V-ОШ, а выполнение завершается до окончания V-ОШ. Пример возможного ОШ, состоящего из V-ОШ, S-ОШ и А-ОШ приведен на рис. 5. Векторные ОШ обычно организуются программно-параллельно или с зацеплением, если нет конфликтов по регистрам. Но есть случай, представленный на рис. 5, когда зацепление должно

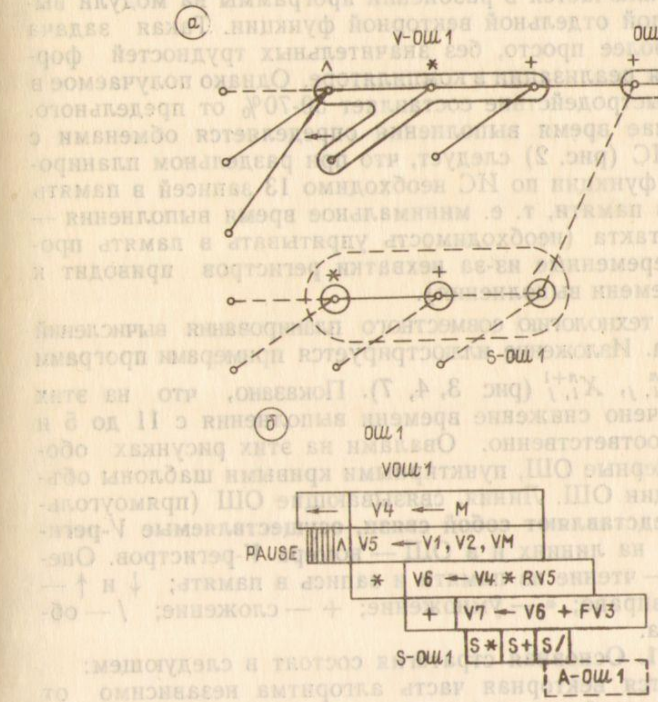

Рис. 5 ИС в Е-разложении сложного операционного шаблона

быть организовано искусственной программной задержкой, набором фиктивных команд, подобранных так, чтобы разгон конвейеров двух векторных устройств заканчивался одновременно. На рис. 5 устройства обмена регистр-память и логическое завершают разгон конвейеров одновременно, если набор команд $PAUSE$ составит 9 тактов. В этом случае устройство умножения будет работать с зацеплением.

ИС всего алгоритма может быть представлена в информационных структурах операционных шаблонов конкретной вектор

ной машины. Как показано в [2, 5] задача согласования алгоритма с архитектурой машины может быть сформулирована как задача отыскания покрытия ИС алгоритма минимальным числом ОШ при ограниченном числе регистров. В этом случае алгоритм будет реализован за минимальное число *VT*.

§ 4.2. Планирование вычислений и задача покрытия

Предложенная в [5] стандартная технология планирования вычислений заключается в разбиении программы на модули вычисления каждой отдельной векторной функции. Такая задача решается наиболее просто, без значительных трудностей формализуется для реализации в компиляторе. Однако получаемое в этом случае быстродействие составляет 50-70% от предельного. В нашем случае время выполнения определяется обменами с памятью. Из ИС (рис. 2) следует, что при раздельном планировании каждой функции по ИС необходимо 13 записей в память и 70 чтений из памяти, т. е. минимальное время выполнения — 83 векторных такта (необходимость упрятывать в память промежуточные переменные из-за нехватки регистров приводит к увеличению времени выполнения).

Рассмотрим технологию совместного планирования вычислений всего алгоритма. Изложение иллюстрируется примерами программ для функций $\widetilde{E}^n_{i,j}$, $X^{n+1}_{i,j}$ (рис 3, 4, 7). Показано, что на этих функциях получено снижение времени выполнения с 11 до 5 и с 17 до 6 *VT* соответственно. Овалами на этих рисунках обозначены конвейерные ОШ, пунктирными кривыми шаблоны объединяются в один ОШ. Линия, связывающие ОШ (прямоугольники в б)), **представляют собой связи, осуществляемые *V*-регистрами.** Цифры на линиях и в ОШ — номера *V*-регистров. Операции ← и → — чтение из памяти и запись в память; ↓ и ↑ — сдвиг влево и вправо; * — умножение; + — сложение; / — обратная величина.

Эвристика 1. Основная стратегия состоит в следующем:

— планируется векторная часть алгоритма независимо от скалярной и адресной частей (рис. 2),

— скалярная и адресная части алгоритма компонуются к векторной встраиванием (по возможности) *S*-ОШ и А-ОШ в *V*-ОШ. Заметим, что помимо повышения эффективности расчета, такой подход позволяет существенно упростить задачу перебора вариантов.

Задача покрытия векторной части решается по ИС алгоритма, представленной в Е-разложении на цепочки связанных между собой элементарных операций [2, 5].

Замечание. Все наборы функций (в том числе состоящие из одной функции) вычисляются в цикле с тем, чтобы рассчитать сетку для всех значений *i* и *j*. За один такт цикла рассчитывается вектор узлов сетки длиной до 64. В этом состоит существенное отличие в организации обхода области расчета от скалярных машин. Возможные варианты обхода представлены

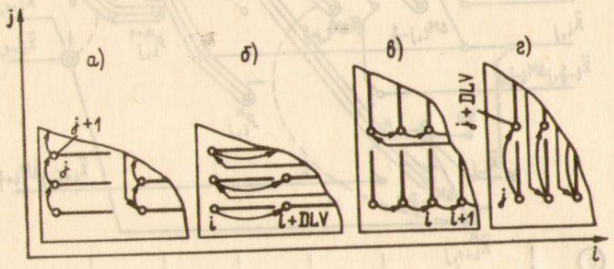

Рис. 6 Схемы возможных обходов расчетной сетки при программировании на ВКМ

на рис. 6. Например, варианту, показанному на рис. 6 соответствует фортраноподобный цикл:

DIRECTION J	*DIRECTION I*
DO 1 *I*=1, *II*	*DO* 1 *J*=1, *JJ*
DO 1 *J*=1, *JJ*	или *DO* 1 *I*=1, *II*
ТЕЛО ЦИКЛА	ТЕЛО ЦИКЛА
1 *CONTINUE*	1 *CONTINUE*

Инструкция *DIRECTION* указывает направление обхода. Здесь и далее фрагменты программ приведены на языке векторной машины, с которого препроцессор порождает команды ассемблера.

Эвристика 2 (рис. 4, 7).

Планирование осуществляется на фрагментах ИС включающих основные векторные функции. Производные векторные функции получаются из основных (например, на рис. 7 производными являются функции $\Delta M^n_{i-1/2,\,j}$, полученные из основных функций $\Delta M^n_{i+1/2,\,j}$) следующими операциями:

— запись-чтение в память;

— хранение на регистре до следующего такта цикла (например при обходе типа 6в) F_i в следующем такте цикла будет иметь смысл F_{i-1});

— использование устройства сдвига — чрезвычайно полезно в разностных задачах, т. к. сокращает время выполнения на

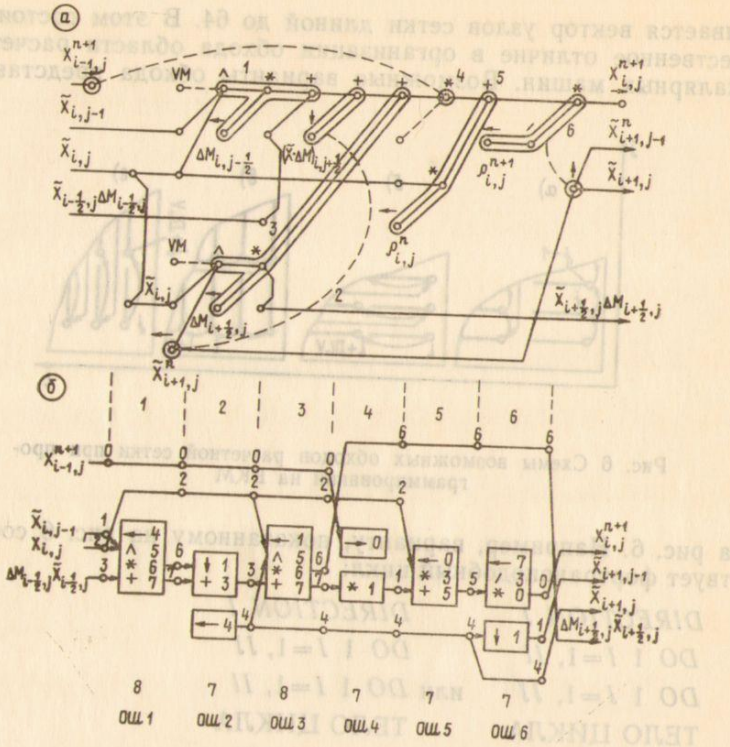

Рис. 7 ИС функции X_{ij}^{n+1} в Е-разложении — а) и в операционных шаблонах — б)

20-30%, но организация такой операции сложна из-за обнуления первого или последнего слова регистра.

Эта эвристика является одним из основных источников увеличения быстродействия за счет уменьшения обменов с памятью.

Распределение быстрых *V*-регистров осуществляется с учетом приемов, приведенных в работах [11, 12] с поправкой на то, что если все производные векторные функции не могут быть получены из основной функции хранением на регистре или устройством сдвига, то основная функция должна быть записана в память.

Эвристика 3 (рис. 7, ОШ1 и ОШ3).

Организация «двойного» зацепления (как и на рис. 5) — наиболее экономичный шаблон с точки зрения занятости регистров.

Эвристика 4 (рис. 4, 7).

До сих пор под ИС алгоритма мы понимали однократное изображение расчетных формул. На самом деле ИС алгоритма есть периодическая информационная структура, в которой периодически повторяются имена переменных и операции, изменяются индексы. Ниоткуда не следует, что выполнение цикла следует начинать с первого оператора. Более того, в операционные шаблоны могут включаться функции из различных тактов цикла. Практически всегда экономится два *VT* за счет переноса сброса регистра в память в следующий такт цикла (сброс регистра в память не зацепляется, к сожалению, с предыдущей операцией) — на рис. 4, ОШ3. Кроме того, первый *VT* цикла чаще всего бывает чистой выборкой из памяти без «полезных» операций, если для первой операции необходимо два операнда из памяти. Одну из выборок следует сделать в предыдущем такте цикла (рис. 4, ОШ4). В итоге, как уже отмечалось, в большинстве случаев экономится минимум два *VT*. Выигрыш в этом случае заметен — 20-50%.

Итак, для полной оптимизации необходимо планировать сразу минимум 3 такта цикла. Заметим, что при такой организации существенно усложняется текст программы. Во-первых, возникают текстовые фазы разгона и выхода из цикла. Во-вторых, текст внутреннего цикла дублируется, но с другим распределением регистров (рис. 4, 7). Дублируемые части должны быть разделены инструкцией, осуществляющей проверку на выход из цикла и увеличение счетчика цикла на 1. В нашем языке ВКМ это *NEXT* ⟨имя переменной цикла⟩.

Эвристика 5 (рис. 7).

Реорганизация инструкции *CVMGT* компилятора *CFT* фортрана [7] осуществляется для тех случаев, когда условие для функции смешения по векторной маске *VM* насчитывается в функциях-предшественниках. Организация *CVMGT* требует минимум два *VT* с занятым логическим устройством. Первый такт — формирование векторной маски, второй — собственно смешение векторов. Например, в нашей задаче условия $\Delta M^n_{i+1/2,\,j} \geqslant 0$ эквивалентны условиям $u^n_{i+1/2,\,j} \geqslant 0$ и формируются на ярусе 6 (используются на ярусе 9). В этом случае следует при расчете $\Delta M^n_{i+1/2,\,j}$ содержимое *VM* занести на скалярные регистры и хранить в памяти до яруса 9. На ярусе 9 формирование *VM* можно организовать со скалярных регистров, таким образом экономится 6 *VT* на трех функциях $X^{n+1}_{i,j}$. В случае метода потоков [5] этот прием позволил сэкономить 12 *VT* из 80. Отметим ограничение — функции, в которых формировались значения *VM* и те, в которых они использовались, должны иметь одинаковую схему обхода.

Во многих задачах обтекания, например, в той части течения, где нет возмущений, векторная маска имеет 1-3 бита, отличных

от остальных. В этом случае расчет в точках, соответствующих означенным битам, можно осуществлять на скалярных устройствах параллельно с векторной обработкой. Такая организация может приносить заметную экономию, до десятков процентов.

После планирования вычислений векторной части планируется скалярная часть алгоритма. При этом организуется параллельная работа различных устройств. С этой целью алгоритм преобразуется, по возможности, к параллельному виду. В алгоритме, рассматриваемом в настоящей работе, все скалярные вычисления выполняются параллельно с векторными.

За счет использования эвристик 1-5 и теоретического обоснования технологии согласования алгоритмов с архитектурой ВКМ [1, 2, 5] в рассматриваемой задаче время выполнения сведено с первоначальных 84 *VT* до 36 *VT*, количество умножений при этом составило — 29, сложений — 30, логических операций — 8, 3 операции «обратная величина». Ввиду того, что число операций не может быть более снижено, эффективность алгоритма составляет примерно 83% (это получается из отношения числа операций наиболее загруженного арифметического устройства к общему числу векторных тактов).

§ 5. Тестирование программы

Помимо работ по тестированию программ, обычных для традиционных машин, программы предельной производительности ВКМ проверялись на правильность организации ОШ с помощью потактного временного профилировщика. Полного сохранения ОШ не удается добиться из-за разрывов зацепления в результате перезагрузок буферов команд. Время выполнения рассматриваемой программы увеличилось на 2 *VT* в результате этого эффекта.

С помощью потактного временного профилировщика оценивалось время выполнения программы в тактах. Были получены некоторые оценки быстродействия ВКМ на отдельных модулях, а также на полной программе расчета задачи обтекания. Процедура согласования алгоритмов с архитектурой ВКМ позволила получить программу с высоким быстродействием ВКМ для данной задачи. Среднее быстродействие на всей программе составило примерно 120 *Mflops*. Для получения программы на ассемблере использовался разработанный авторами препроцессор с языка векторной машины.

§ 6. Кодирование программы

Программа составляется по ИС в операционных шаблонах на языке векторной машины. Язык векторной машины включает инструкции фортранного типа: *DO*, *REPEAT*, *DIMENSION*, *ARRAY* (для описания векторных массивов), *IF*, *THEN*, *ELSE*, *ENDIF*. Кроме того, введены инструкции *NEXT*, *STORE*, *READ* (для обменов регистр-память), *DIRECTION* — указатель направления обхода, *PAUSE* — для организации зацепления, *CVMGT*, *VMSTORE*, *VMREAD* — для хранения и загрузки *VM* со скаляров, а также некоторые другие.

Текст арифметических выражений хранится в фортранном виде и на ассемлере, по мере необходимости редактор ВКМ вызывает на экран дисплея тот или иной вариант текста, кроме того в поле комментариев может вызываться таблица соответствия имен переменных и регистров. Таблица может исправляться редактором.

§ 7. Выводы

Использование технологии согласования алгоритмов с архитектурой ВКМ, основанной на результатах работ [1, 2, 5], а также системы инструментальной поддержки, включающей препроцессор с языка ВКМ, дает возможность получать программы с высоким, близким к предельному, быстродействием. В программе численного решения задачи обтекания методом крупных частиц, рассматриваемой в настоящей работе, удалось полностью векторизовать алгоритм и избежать потерь при расчете условных выражений.

На основе опыта конструирования программ решения различных задач математической физики можно отметить высокий уровень организации архитектуры ВКМ, в первую очередь чрезвычайно удачную сбалансированность векторных, скалярных и адресных устройств. Наряду с малым временем разгона конвейеров (2-16 тактов) это позволяет достаточно полно использовать возможности машины для широкого круга задач, как ни в какой из других известных нам архитектур супер-ЭВМ.

Наиболее «узким» местом в рассматриваемой ВКМ является обмен векторных регистров с памятью. Понимая всю техническую сложность увеличения количества *V*-регистров, мы не удержались от соблазна проэкспериментировать с конструированием программ на гипотетической ВКМ, идентичной рассматриваемой, но с неограниченным числом регистров. Обработка программ указанных выше задач показала, что для данного набора устройств при 16 *V*-регистрах «принудительный» сброс промежуточных переменных приходится 1 раз на 30-40 *VT* при стратегии наиболее полной загрузки устройств. Причем увеличение числа регистров незначительно — на 5-20% сказывается на быстродействии при «ручном» программировании, но зато само программирование существенно упрощается. При автоматической оптимизации должно, по всей видимости, сильнее повыситься и быстродействие.

Еще одно наблюдение заключается в том, что неэффективно — минимум за 2 *VT* — реализуется смешение векторов по

векторной маске (если не удается использовать эвристику 5). Наличие отдельного логического устройства для организации векторной маски позволило бы на неоднородных алгоритмах поднять производительность на 5-20%. Отметим, что в отличие от увеличения числа регистров нет принципиальных технических трудностей, препятствующих созданию такого устройства.

Авторы выражают благодарность Льву Николаевичу Столярову за интересные обсуждения, которые способствовали появлению данной работы.

ЛИТЕРАТУРА

1. *Столяров Л. Н.* Структурный анализ алгоритмов для эффективной организации вычислений. В кн.: Вычислительные системы и автоматизация научных исследований. — М., МФТИ, 1980, с. 23—45.
2. *Столяров Л. Н.* Согласование алгоритма с архитектурой векторно-конвейерной вычислительной системы. — В сб.: Организация вычислений на супер-ЭВМ. — М., МФТИ, 1985, 201 с. (Рукопись депонирована в ВИНИТИ, № 5934-85, 8 июля 1985 г.).
3. *Белоцерковский О. М., Давыдов Ю. М.* Метод крупных частиц в газовой динамике. Вычислительный эксперимент. — М.: Наука, 1982.
4. *Кузюрин Н. Н., Шокуров А. В.* Особенности алгоритмической и программной реализации математических функций на векторно-конвейерных ЭВМ. — В сб.: Вопросы кибернетики-97, «Проблемы организации высокопроизводительных ЭВМ». — М., 1984, с. 1—23.
5. *Бабаков А. В., Бацуков О. С., Белоцерковский О. М., Столяров Л. Н.* О возможности достижения высокой производительности на векторно-конвейерных ЭВМ при решении задач математической физики. — Ж. вычисл. матем. и матем. физ., 1985, **т. 26, № 4, с. 601—613.**
6. *Воеводин В. В.* Математические проблемы освоения супер-ЭВМ. В кн.: Вычислительные процессы и системы. Вып. 2. — М.: Наука, 1985, с. 3—12.
7. *Меткалф М.* Оптимизация в фортране. — Пер. с англ. — М.: Мир, 1985.
8. *Андрианов А. Н., Ефимкин К. Н., Задыхайло И. Б., Поддерюгина Н. В.* Язык НОРМА. — Препринт ИПМ АН СССР, 1982, № 165, 34 с.
9. *Вольдман Г. Ш., Задыхайло И. Б.* Некоторые соображения об определении степени непроцедурности языков программирования. — Препринт ИПМ АН СССР, 1980, № 51, 28 с.
10. *Поспелов Д. А.* Введение в теорию вычислительных систем. — М.: Сов. радио, 1972.
11. *Belady L. A.* A Study of Replacement algorithms for a virtualstorage computer. — IBM Systems Journal, 1966, v. 5, N 2, p. 78.
12. *Horwitz L. P., Kocrp R. M., Miller R. E., Winograd S.* Index register allocation, JACM, 1966 (Jan.), v. 13, p. 43.

СОДЕРЖАНИЕ

Технический редактор *Л. А. Белова*

Сдано в набор 23.03.87 Подписано в печать 16.11.87 Т—20030
Формат бумаги 60×90¹/₁₆ Бум. тип. № 1 Литературная гарнитура
Высокая печать. Усл. печ. л. 8,75 Усл. кр.-отт. 9,0 Уч.-изд. л. 7,96
Тираж 1000 экз. Заказ 1336 Цена 1 р. 20 к.

Производственно-издательский комбинат ВИНИТИ
140010, Люберцы, 10, Московской обл., Октябрьский просп., 403