МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1981 год Выпуск 89

УДК 519.683:681.324:681.3.015

О ДИАЛОГОВОЙ СИСТЕМЕ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Н.К. Кербель

Неизбежным следствием развития вычислительной техники является совершенствование методов обработки информации, а также рост числа пользователей "непрофессионалов". Высокая эффективность применения указанных методов должна быть достигнута как путем создания средств их автоматизации, так и путем создания систем обучения пользователя принятию решений по их применению. Описание и анализ функционирования систем подобного рода, используемых при подготовке пользователей ЭВМ, даны в работах [1-3].

Развитие многопроцессорных систем и, в частности, мульти-мини-машинных систем с программируемой структурой, привело к появлению нового объекта - параллельной (п-) программы [4], требующей специальных методов и средств автоматизации программирования, в том числе средств, обеспечивающих совокупное эффективное использование процессоров и памяти. При построении п-программ, ввиду многокритериальности задачи, речь идет о создании все более совершенных средств, о том, чтобы вопрос о распараллеливании вычислений решался в диалоге с человеком, с использованием имеющегося опыта построения п-программ.

В работе предлагается один из подходов к построению диалоговой системы структурного параллельного программирования, предназначенной для обучения пользователя принятию решений по созданию параллельных алгоритмов. Этот подход предполагается реализовать на базе системы типа МИНИМАКС.

1. Стратегия создания параллельных программ базируется на выявлении внутренних связей отдельных компонент задачи и рассмотрении их с точки зрения возможности параллельного выполнения на системе. Поэтому предла-

гается как р а с п а р а л л е л и в а т ь последовательные алгоритмы (программы), так и с о з д а в а т ь новые параллельные алгоритмы с помощью предложенной системы.

Основой при создании параллельных алгоритмов является использование естественной иерархии задачи и свойств вычислительной системы, на которой будет реализован параллельный алгоритм. Иерархия задачи есть не что иное, как представление понятий более высоких уровней в терминах понятий более низких уровней [5].На каждом этапе декомпозиции задача представляется (в диалоге с пользователем) в виде отдельных фрагментов с информационными и управляющими связями между ними. Затем каждый фрагмент рассматривается как отдельная задача, к ней снова применяется декомпозиция и т.д. На каждом этапе декомпозиции можно выделить те фрагменты, которые могут быть выполнены параллельно.

Основой декомпозиции являются (примеры см. с. 33):

- расслоение исходной задачи по данным с последующей привязкой к ним действий;
- расслоение действий на независимые (параллельно выполняемые) с последующей привязкой данных;
- комбинация этих расслоений.

Степень декомпозиции задачи диктуется соображениями целесообразности, которые определяются пользователем. При этом могут учитываться поставленные цели, наличие ресурсов и т.д. Характер декомпозиции зависит от физического смысла задачи. Конечной целью декомпозиции является получение алгоритма задачи, пригодного для распараллеливания.

При декомпозиции в диалоге будет обращаться внимание на два момента.

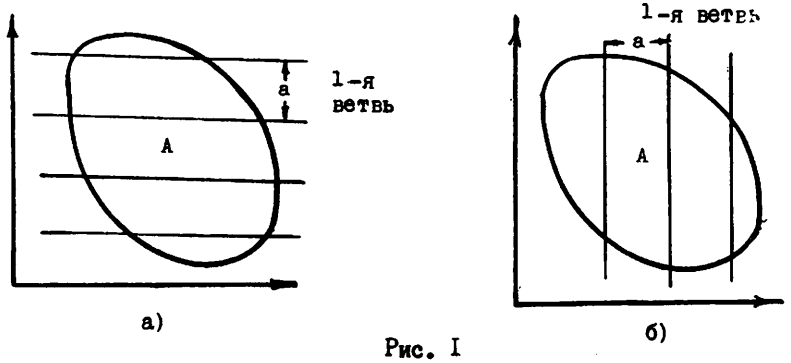
Во-первых, на выявление тех фрагментов или групп фрагментов, алгоритмы которых удовлетворяют условиям автоматического распараллеливания. Например, автоматическое распараллеливание алгоритмов с п е ц и а л ь н о г о в и д а [6] основано на их представлении в виде ориентированных графов без циклов и петель, вершины которых соответствуют элементарным частям алгоритма, а дуги отражают функциональную зависимость между ними; автоматическое распараллеливание п о ц и к л а м основано на отыскании независимых циклов, либо отыскании условий, при которых может быть организовано конвейерное выполнение цикла. Обзор таких методов приведен в работах [7,8].

Во-вторых, на правильный, эффективный выбор метода решения отдельных фрагментов, учитывающего явным или неявным образом свойства вычислителя, на котором эта задача будет реализована. Иллюстрацией этому может служить п-программа внутренней сортировки массива из N целых неотрицательных чисел [9], которая производит упорядочение массива по возрастанию. Действительно, распараллеливание по циклам уже имеющегося последовательного алгоритма, основанного, например, на пузырьковом методе, дает один результат. Переход же к табличному методу (что возможно в связи с наличием в системе большей чем в одной ЭВМ оперативной памяти) и его распараллеливание даст уже другой, более сильный результат.

2. Примеры задач, иллюстрирующих необходимость диалога при построении п-алгоритмов.

2.1. В задачах математической физики часто требуется определить значения функций в заданной области с известными граничными условиями. В зависимости от применяемой декомпозиции эти задачи могут быть распараллелены одним из описанных ниже способов:

- область данных A распределяется горизонтальными или вертикальными полосами между ветвями п-программы [7]; на основе этого однородного распределения задача распараллеливается на идентичные по действиям ветви над различными данными (см.рис.1: а - горизонтальное распределение; б - вертикальное распределение данных);



а) б)

Рис. 1

- область данных A делится на подобласти A_i, отличающиеся различным поведением функций; затем решается задача перевода граничных условий на подобласти, после чего для каждой подобласти исходная задача может быть решена независимо [10] (рис. 2);

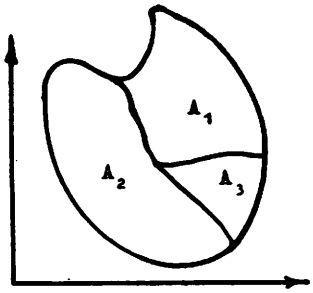


Рис.2

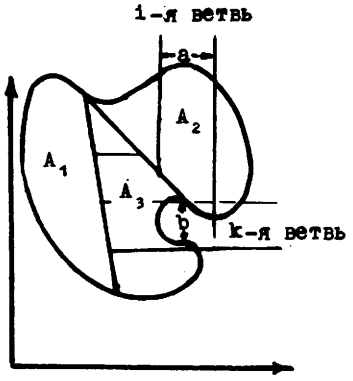


Рис.3

- комбинированный способ, заключающийся в последовательном делении области A на подобласти указанными выше способами (рис.3).

Выбор способа распараллеливания, очевидно, связан с "физикой" задачи, и трудно надеяться на эффективность п-алгоритма, если он создан без диалога с пользователем.

2.2. Автоматическое распараллеливание программ в общем случае требует от пользователя знания содержательного смысла задачи и в связи с этим принятия решения о правильности полученного результата распараллеливания. Это может быть проиллюстрировано на примере упомянутой выше задачи сортировки. Рассмотрим фрагмент последовательной программы внутренней сортировки, основанной на пузырьковом методе, при котором массив чисел упорядочивается по возрастанию:

```
      DO 10 I=1, N-1
      DO 10 J=I+1, N
      IF (M(I)-M(J)) 10,10,20
   20 IR=M(I)
      M(I)=M(J)
      M(J)=IR
   10 CONTINUE
```

После выполнения этой программы массив M (здесь M - массив чисел, состоящий из N элементов) упорядочен по возрастанию. Попробуем теперь написать п-программу сортировки. Распределим массив M по L ветвям. Фрагмент программы 1-й ветви будет выглядеть сле-

дующим образом (функция DEL(N) вычисляет величину [N/L] + 1 для ветвей с относительными номерами не больше остатка от деления N на L , и [N/L] для остальных ветвей, квадратные скобки означают целую часть числа):

```
      N=DEL(N)
      DO 10 I=1,N-1
      DO 10 J=I+1,N
      IF (M(I)-M(J)) 10,10,20
   20 IR=M(I)
      M(I)=M(J)
      M(J)=IR
   10 CONTINUE
```

Теперь в каждой ветви l (l= 1,L) имеется упорядоченный массив чисел из DEL(N) элементов. Для получения конечного результата необходимо произвести слияние полученных массивов. Выбор таких дополнительных процедур ведет к необходимости диалога.

2.3. В определенных ситуациях автоматическое выявление параллелизма в циклах не дает положительного результата, тем не менее задача распараллеливается очевидным образом. Рассмотрим фрагмент программы нахождения корня уравнения $x = f(x) - g(x)$ методом последовательных приближений за N итераций

```
      X =X0
      DO 10 I = 1,N
   10 X = F(X) - G(X)
```

Этот цикл является зависимым [8] и его витки непосредственно не могут быть выполнены параллельно. Однако, введя вспомогательную переменную y , п-программу можно сконструировать достаточно просто:

```
I ветвь                2 ветвь
X=X0                   Y=X0
DO 10 I=1,N            DO 10 I=1,N
X=F(X)                 Y=G(Y)
ПЕРЕДАЧА X             ПЕРЕДАЧА Y
10 X=X-Y            10 Y=X-Y
```

В общем случае введение новых переменных не может быть сделано без диалога с пользователем. Из приведенных примеров следует, что п-программирование требует анализа задачи с точки зрения выявления естественного параллелизма и свойств конкретного вычислителя, рассмотрения возможности распараллеливания на основе разделения действий или разделения данных; а затем - принятия решения

о построении п-алгоритма. Вопрос о распараллеливании того или иного фрагмента решается с помощью диалоговых средств автоматизации п-программирования, но последнее слово при принятии решения и применении рекомендуемых подходов остается за пользователем. Поэтому практически важным становится вопрос эффективного обучения пользователя созданию п-алгоритмов, в частности, организации консультаций со стороны инструментального комплекса, обладающего запасом знаний по современному состоянию параллельного программирования. Таким инструментальным комплексом и представляется описываемая ниже система.

3. Функциональная структура и состав системы для реализации поставленной задачи представлены на рис.4.

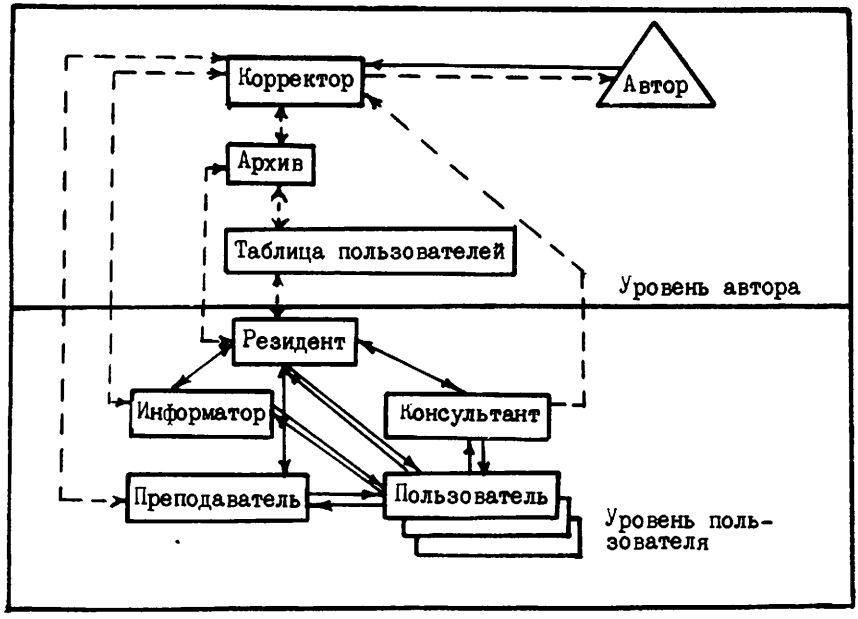


Рис. 4

Структурные единицы (модули) имеют следующий смысл.

Пользователь инициирует работу системы с целью получения им навыков параллельного программирования (в частно -

ности - совета по построению п-алгоритма и написанию соответствующей п-программы). Вспомогательной целью может быть получение справок по теории многопроцессорных систем.

П р е п о д а в а т е л ь обучает пользователя приемам структурного программирования и принятию решений по их применению при написании качественных п-программ. При обучении предполагается использовать алгоритмы, аналогичные разработанным в Ин-те кибернетики АН УССР для систем типа "Педагог" [II].

К о н с у л ь т а н т выясняет в диалоге с пользователем возможность получения качественного п-алгоритма (на основе свойств вычислителя и анализа задачи), а затем дает рекомендации по его программированию. При этом будет рассматриваться возможность декомпозиции исходной задачи на подзадачи с целью поиска их аналогов в банке системы, для которых существуют п-алгоритмы. По требованию системы или пользователя могут быть вызваны автоматические распараллеливатели, которые собственными диалоговыми средствами выяснят возможность создания п-алгоритма.

И н ф о р м а т о р выдает справки о составе системы и ее функционировании, информирует пользователя по требуемым вопросам теории систем и программирования.

Р е з и д е н т осуществляет управление системой и предварительную ее настройку на конкретного пользователя. Для этого резидент определяет статус пользователя, принимает решение о вызове отдельных модулей, организует одновременную работу многих пользователей и т.д. По окончании контакта с модулем либо по запросу пользователя управление может быть передано резидентом любому другому модулю системы. Окончание контакта с системой определяется резидентом (в случае конца работы с модулем и отсутствием других запросов у пользователя) либо указывается пользователем посредством специальной директивы.

Т а б л и ц а п о л ь з о в а т е л е й содержит необходимую текущую информацию о пользователях системы, включая историю их обучения.

А р х и в содержит статистическую информацию, касающуюся функционирования системы, а также информацию о пользователях, подлежащую длительному хранению. Содержимое архива служит основой корректировки системы.

К о р р е к т о р осуществляет корректировку системы на основе содержимого архива, вплоть до полной замены содержимого отдельных модулей.

А в т о р инициирует функционирование корректора с целью получения сведений о работе системы для ее модификации.

4. С п о с о б о м в з а и м о д е й с т в и я модулей является диалог: двухсторонний (⇄), односторонний (⇉) и вырожденный (↔ - передача управления и ←- - передача информации). Согласно [II], под двухсторонним понимается диалог, характеризующийся оперативностью, способностью партнеров к управлению и способностью партнеров к самостоятельным действиям. В случае, когда одна из этих характеристик отсутствует, речь идет об односторонем диалоге.

В системе существует два уровня иерархии - уровень автора и уровень пользователя. Группа модулей, образующих уровень автора, осуществляет анализ функционирования всей системы и коррекцию ее отдельных модулей. Модули уровня пользователя осуществляют непосредственное функционирование системы.

У р о в е н ь а в т о р а имеет более высокий приоритет, чем уровень пользователя. По директиве автора требуемая информация, содержащаяся в таблице и архиве пользователей, извлекается корректором. Анализируя эту информацию, автор принимает решение о необходимых изменениях в отдельных модулях. Затем корректор извлекает содержимое затребованных модулей и после проведенной автором модификации направляет обратно. В целях защиты управляющей структуры взаимодействие корректора с резидентом запрещено.

У р о в е н ь п о л ь з о в а т е л я имеет характерную особенность: взаимодействие любых модулей этого уровня, равно как и взаимодействие с модулями другого уровня, возможно только при посредстве резидента. Поэтому, хотя внутри уровня пользователь и обладает высшим приоритетом при обращении к тому или иному модулю, вопрос о возможности передачи управления решается резидентом. После принятия решения резидент инициирует двухсторонний диалог пользователя с другими модулями.

Цели такого диалога различны для различных модулей. Так, двухсторонний диалог "информатор-пользователь" имеет целью пополнение знаний пользователя о предмете беседы, диалог "консультант-пользователь" ведется с целью совместного выяснения возможностей создания п-алгоритма в конкретном случае и, наконец, преподаватель в диалоге с пользователем ставит цель научить последнего приемам параллельного программирования. Степень взаимодействия человека и ЭВМ во всех описанных случаях различна. В первом случае ЭВМ вы-

ступает в роли "справочного бюро", во втором случае - в роли "советчика", в третьем - в роли "наставника". Формально это может выглядеть так.

Пусть два объекта A и B находятся в состоянии двухстороннего диалога, $T = \{\tau_i\}$ - множество тактов диалога. Под тактом диалога понимается выдача сообщения партнеру. Концом такта является передача управления в диалоге. Тот факт, что объекты A и B рассматриваются в такте τ_i, обозначим A_i, B_i. Таким образом, каждому объекту A, B соответствует множество его состояний $\{A_i\}$, $\{B_i\}$, и, говоря о "стратегии A_i", мы будем иметь в виду "стратегию объекта A в состоянии A_i".Пусть для некоторого такта диалога τ_i объект A_i управляет диалогом, тогда B_{i+1} будет управлять диалогом в такте τ_{i+1}. Связь между A и B будем называть с л а б о й с точки зрения объекта A ($A \to B$), если $\forall \tau_i \in T$ стратегия B_{i+1} определяется объектом B, независимо от стратегии A_i, а стратегия A_i определяется стратегией B_{i-1}. В случае, если $\forall \tau_i \in T$ стратегия B_{i+1} определяется стратегией A_i, а стратегия A_i определяется стратегией B_i, то связь между A и B назовем сбалансированной ($A \to B$). И наконец, в случае, если $\forall \tau_i \in T$ стратегия B_{i+1} определяется стратегией A_i, а стратегия A_i определяется A, независимо от B_{i-1}, то связь между A и B будем называть с и л ь н о й с точки зрения объекта ($A \Rightarrow B$).

Например, если A - ЭВМ, а B - человек, с точки зрения ЭВМ слабой связью является связь "информатор-пользователь", сильной - "преподаватель-пользователь" и, наконец, связь "консультант-пользователь" служит примером сбалансированной связи.В случае сбалансированной связи стратегия A и B равнозначна. В приведенном примере стратегия консультанта в какой-то мере имитирует действия квалифицированного пользователя при конструировании п-программы.

Особенностью предлагаемой диалоговой системы является наличие консультанта, который совместно с пользователем принимает решение при создании конкретного параллельного алгоритма.При создании консультанта предполагается использовать элементы теории искусственного интеллекта. Предлагаемая система рассматривается как универсальная, что позволит использовать ее для обучения и принятия решений в широкой области знаний. Конкретный вариант системы предполагается реализовать при построении параллельных программ для вычислительных систем с программируемой структурой.

Литература

I. ДОВГЯЛЛО А.М., НЕБРАТ О.П., ПЛАТОНОВ Б.А. Обучение с использованием вычислительных машин, современное состояние и перспективы. - УСиМ, 1978, № 2, с. 12-20.

2. Two years curricula in computer studies - implementing the guidelines/ Dillman R.W., Anderson W.H., Choper D.L., Lloyd J.M., Simms K.B., Willams J.F. - SIGCSE Bull., 1978, v. 10, N 3, p. 140-150.

3. ABRAHAMS J.R.,De WOLF G. Experience with a computer - assisted training system. - Inform.Process, Proc.IFIP Congr., Toronto,1977, Amsterdam e.a.,1977,p.601-605.

4. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. - Новосибирск: Наука,Сиб.отд-ние,1966.-308 с.

5. ДАЛ У., ДЕЙКСТРА Э., ХООР К. Структурное программирование. Пер. с англ.- М.: МИР, 1975. - 247 с.

6. КЕРБЕЛЬ В.Г. Вопросы мультиобработки и параллельного программирования на однородных вычислительных системах. Автореферат диссертации на соискание ученой степени канд.техн.наук (05.13.13). Новосибирск, Б.и., 1979, 18 с. - В надзаг.: Институт математики СО АН СССР.

7. МИРЕНКОВ Н.Н. Системное параллельное программирование. -Новосибирск, Б.и., 1978. Ч.I - 35 с. Ч.II. - 50 с. (Препринт/ ИМ СО АН СССР: ОВС-05, ОВС-06).

8. МИРЕНКОВ Н.Н., СИМОНОВ С.А. Выявление параллелизма в циклах методом имитации их выполнения. -Кибернетика, 1981, № 3, с.28-33.

9. КЕРБЕЛЬ Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И. Параллельные программы для системы МИНИМАКС. -В кн.: Математическое обеспечение вычислительных систем (Вычислительные системы, вып. 78). Новосибирск, 1979, с. 78-89.

10. Об организации параллельных вычислений и "распараллеливании" прогонки /Яненко Н.Н., Коновалов А.Н., Бугров А.Н., Шустов Г.В. -В кн.: Численные методы механики сплошной среды. Новосибирск, 1978, т. 9, № 7, с. 139-146.

11. Диалог человека и ЭВМ: Основные понятия и определения /Брановицкий В.И., Довгялло А.М., Никитин А.И., Стогний А.Л.-УСиМ, 1978, № 4, с. 3-6.

Поступила в ред.-изд.отд.
28 мая 1981 года