ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ ИЗ МИКРО-ЭВМ
(Вычислительные системы)
1983 год Выпуск 97

УДК 519.682:519.683

ПОСТРОЕНИЕ И ЭКВИВАЛЕНТНЫЕ ПРЕОБРАЗОВАНИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ, ПРЕДСТАВЛЕННЫХ НА ЯЗЫКЕ ФОРМАЛИЗАЦИИ

Б.А. Сидристый

При алгоритмизации широкого класса задач, в частности, задач сбора, обработки и отображения учетной информации в автоматизированных системах управления предприятиями удобным языком формального описания алгоритмов решения этих задач является язык формализации, краткое описание которого приводится в данной работе. Языки такого высокого уровня, как язык формализации, в практике программирования наиболее рационально применять в составе средств, обеспечивающих определенную технологию программирования, в частности, технологию получения параллельных алгоритмов для решения задач на вычислительных системах. В работе рассматриваются вопросы перехода от описания последовательных алгоритмов на языке формализации к параллельным, а также вопросы эквивалентных преобразований этих алгоритмов с целью улучшения некоторых показателей, характеризующих их эффективность. В качестве базы для решения указанных вопросов используются модели и методы построения вычислительных систем и параллельных алгоритмов, предложенные в [1] и развитые в работах [2-6].

В рассматриваемом здесь языке формализации для представления числовых и логических зависимостей используются понятия переменной и функции от n аргументов (n = 1,2,...), которые обозначаются обычным образом: a, KM, $F_{i,j}$, LA_{α}. Переменные и функции могут иметь числовые, логические ("истина", "ложь") и словарные (в виде последовательностей из букв и цифр) значения и в соответствии с этим подразделяются на числовые, логические и словарные переменные и функции. В качестве аргументов функций в языке формализации

могут выступать только числовые переменные с целочисленными положительными значениями. Для каждого аргумента x функции устанавливается область его возможных значений в виде $0 \le x \le A$, где A - наибольшее возможное значение аргумента. Кроме переменных этих функций, в описаниях алгоритмов на языке формализации могут использоваться числовые, логические ("истина", "ложь") и словарные константы.

Переменным и функциям могут присваиваться новые значения с использованием операций элементарного и группового присваивания. Операции элементарного присваивания имеют вид: $V \doteq \mathcal{T}$, где V - переменная или функция, значение которой присваивается при выполнении данной операции, а $\mathcal{T}$ - терм, составленный из некоторых исходных переменных и функций, а также конструкций, которые называются связками и операторами. Построение термов подчиняется следующей схеме.

I. Любая константа, переменная или функция является термом, который в этом случае называется элементарным. Переменные, выступающие в качестве аргументов данной функции - терма, называются его свободными переменными, от которых он зависит и которые имеют области возможных значений такие же, как у этих аргументов. Терм является числовым, логическим или словарным в зависимости от типа константы, переменной или функции, составляющей этот терм. Элементарные термы имеют значения, совпадающие со значениями составляющей его константы, переменной или функции.

2. Если $\mathcal{T}$, F - термы, то $\mathcal{T} \circledcirc F$, $\circ\mathcal{T}$ - также термы (здесь $\circledcirc$ - обозначение двуместных числовых, логических и словарных связок таких, как $+, -, \times, /, \vee, \wedge$ и т.д., а $\circ$ - обозначение одноместных связок таких, как $\neg, \sqrt{}$, ln, sin, cos и т.п.). При этом числовые связки могут применяться только к числовым термам, логические к логическим, а словарные к словарным. Свободные переменные термов $\mathcal{T}$ и F являются свободными переменными терма $\mathcal{T} \circledcirc F$ с теми же областями возможных значений, а свободные переменные терма $\mathcal{T}$ являются свободными переменными терма $\circ\mathcal{T}$. Термы $\mathcal{T} \circledcirc F$ и $\circ\mathcal{T}$ для любого набора допустимых значений их свободных переменных имеют конкретные значения, которые вычисляются путем применения к значениям термов $\mathcal{T}$ и F операций, соответствующих связкам $\circledcirc$ и $\circ$.

3. Если $\mathcal{T}(x)$ - терм со свободной переменной x, то $\mathop{\mathcal{B}}_{a \le x \le b} \mathcal{T}(x)$ - также терм (здесь a и b - термы с целочисленными значениями, не

зависящие от x, а $\mathcal{B}$ - обозначение одного из операторов $\Sigma, \exists, \forall$, min и т.д.). Свободными переменными терма $Z = \underset{a \le x \le b}{\mathcal{B}}\ \mathcal{T}(x)$ являются все свободные переменные терма $\mathcal{T}(x)$ и с теми же областями возможных значений, кроме переменной, которая в данном случае называется связанной переменной Z и от которой Z не зависит. Значение Z для конкретного набора возможных значений его свободных переменных вычисляется путем последовательного применения соответствующей оператору $\mathcal{B}$ операции (+ для оператора Σ, V для оператора $\exists, \Lambda$ для оператора $\forall$ и т.д.) к значениям терма $\mathcal{T}$ по всем x таким, что $a \le x \le b$.

4. Если $\mathcal{T}_1, \ldots, \mathcal{T}_n$ - числовые (словарные) термы с одинаковыми свободными переменными, а $R_1, \ldots, R_{n-1}$ - логические термы с одинаковыми свободными переменными ($n = 2,3,\ldots$),то $\mathcal{T}_1$ при R_1 иначе ... иначе $\mathcal{T}_n$ - также терм со свободными переменными термов $\mathcal{T}_i$ и R_i , значение которого равно значению $\mathcal{T}_1$ при R_1 = "истина", значению $\mathcal{T}_2$, если $R_1 \neq$ "истина", но R_2 = "истина" и т.д.

Операции группового присваивания имеют вид $V(x,\ldots, y,\ldots,z) \doteq \mathcal{T}$; $a \le x \le b, \ldots, c \le y \le d, \ldots, e \le z \le f$, где $V(x,\ldots,y,\ldots,z)$ обозначает некоторую функцию, в частности, зависящую от аргументов $x,\ldots,y,\ldots,z$, символы a и b обозначают термы, не зависящие от x и имеющие целочисленные значения из области возможных значений x , символы c,d обозначают термы, не зависящие от $x,\ldots,y$ и с целочисленными значениями из области возможных значений y , и наконец, символы e,f обозначают термы, не зависящие от $x,\ldots,y,\ldots,z$ и имеющие целочисленные значения из области возможных значений z . Переменные $x,\ldots,y,\ldots,z$ называются переменными операции группового присваивания, а выражения $a \le x \le b, \ldots, c \le y \le d, \ldots, e \le z \le f$ - условиями, определяющими их области допустимых значений в операции или просто условиями допустимых значений. Операции группового присваивания, в частности,могут иметь только одну переменную операции. Терм $\mathcal{T}$ в операции группового присваивания должен быть таким, что его значение для любого набора допустимых значений переменных операции определяется только этим набором и никак не зависит от других значений $\mathcal{T}$ для других наборов. Выполнение операции группового присваивания заключается в присвоении функции $V(x,\ldots,y,\ldots,z)$ значений терма $\mathcal{T}$ для всех наборов допустимых значений переменных $x,\ldots,y,\ldots,z$.

Алгоритмы на языке формализации представляются в виде вложенных друг в друга блоков. Рассматриваются блоки четырех типов - линейные блоки, блоки с условиями, циклические блоки (или просто циклы) и групповые блоки.

Линейный блок представляет собой последовательность других блоков и операций элементарного или группового присваивания. Выполнение линейного блока заключается в последовательном выполнении составляющих его блоков и операций.

Блоки с условием имеют вид: если Y то [B] либо вид: если Y то [B] иначе [D] , где B и D - блоки, а Y - логический терм. При Y = "истина" выполнение указанных блоков с условием заключается в выполнении блока B. При Y = "ложно" в первом случае ничего не выполняется, а во втором выполняется блок D .

Циклические блоки подразделяются на три типа - с предварительной проверкой условия выхода из цикла, с последующей проверкой такого условия и циклы, управляемые переменной цикла. Циклические блоки первого типа записываются в виде: пока Y Ц [B], где Y - логический терм, называемый условием выхода из цикла, а B - некоторый блок, называемый телом цикла. При Y = "истина" выполняется блок B , после чего снова проверяется условие Y, при Y = "ложь" выполнение цикла заканчивается. Циклические блоки второго типа записываются в виде: Ц[B] до Y. Выполнение таких циклов отличается тем, что условие выхода из цикла Y проверяется после того, как выполнится блок B. Циклические блоки третьего типа имеют вид $Ц_{a\le x\le b}$ [B] либо вид $Ц_{x=a-b}$ [B] , где a и b - термы, не зависящие от x и с целочисленными значениями, x - переменная цикла, а $a \le x \le b$ и $x = a - b$ - условия, определяющие пределы изменения переменной цикла x. Выполнение циклического блока рассматриваемого типа заключается в выполнении блока B при различных значениях x b-a+1 раз, а в случае условия x=a - b значения x должны обязательно перебираться в порядке a, a+1,..., b. Может задаваться любой более сложный порядок, например, x=c-b, a - c - 1, если $a \le c \le b$, и т.п. При a>b выполнение блока B блокируется.

Групповые блоки имеют вид: $Г_{a\le x\le b}$ [B], где a,b и B обозначают то же, что и в случае циклических блоков третьего типа. Отличаются групповые блоки от циклов третьего типа дополнительным условием: результаты выполнения блока B определяются только данным значением x и не зависят от результатов его выполнения при других значениях x, т.е. групповой блок объединяет в себе выполнение,

группы независимых друг от друга ветвей при разных x . Допускается запись следующего вида: $\Gamma\ \underset{\substack{a\le x\le b\\ c\le y\le d}}{}\ [B]$.

Для указания конца выполнения алгоритма используется операция стоп. Алгоритмы, выполняющиеся во времени, называются вычислительными процессами. Вычислительные процессы оформляются в виде: когда Y то [B] либо в виде B, где Y - собственное условие выполнения процесса, представляющее собой логический терм, а B - некоторый блок. В первом случае вычислительный процесс называется вычислительным процессом с собственным условием выполнения. В некоторых случаях он может помечаться его именем в виде: когда Y то [P:B] или P:B, где P - последовательность из букв и цифр.

Вычислительные процессы могут находиться в состоянии выполнения, ожидания и пассивном состоянии. В состоянии выполнения он может находиться только при U = "истина", где U - его условие выполнения. Из этого состояния процесс может перейти в пассивное состояние в результате выполнения в нем операции стоп, либо в состояние ожидания, когда станет выполняться условие $U \neq$ "истина". Из состояния ожидания вычислительный процесс может перейти в состояние выполнения, когда станет выполняться U = "истина". Из пассивного состояния процесс может перейти в состояние выполнения в результате того, что в некотором другом вычислительном процессе выполнилась операция запуска данного процесса.

Операция запуска имеет вид: запуск P , где P - имя запускаемого процесса, либо вид: запуск $\mathcal{P}$, где $\mathcal{P}$ - описание процесса. При выполнении операции запуска формируется условие выполнения U запускаемого процесса следующим образом: $U = Z \wedge Y$, где Z - условие выполнения запускающего процесса, а Y - собственное условие выполнения запускаемого процесса. После выполнения операции запуска начинает выполняться новый и продолжает выполняться запускающий вычислительные процессы.

Структура параллельных алгоритмов и способ их построения в большой степени зависят от определенных свойств вычислительной системы, на которой эти алгоритмы должны выполняться. Будем рассматривать виртуальную однородную вычислительную систему, состоящую из N одинаковых вычислительных машин (ВМ) с номерами 0,..., N - 1 и системы связи между этими ВМ. Виртуальная ОВС может быть реализована на реальной вычислительной системе с помощью необходимых программных средств.

Параллельные алгоритмы для рассматриваемой виртуальной ОВС строятся в виде N ветвей, каждая из которых представляет из себя последовательный алгоритм на языке формализации, предназначенный для выполнения на определенной ВМ в ОВС. Выполнение ветви параллельного алгоритма на соответствующей ВМ заключается в выполнении составляющих ее операций над значениями переменных и функций параллельного алгоритма, находящихся в памяти данной ВМ, и операций системного взаимодействия, осуществляющих обмен этими значениями с другими ветвями алгоритма. При этом значения отдельных функций параллельного алгоритма либо полностью дублируются во всех ВМ системы, либо распределяются между ними по определенному правилу.

Множество всех функций параллельного алгоритма и его переменных разделим на две группы: глобальные и локальные переменные и функции. К первой группе отнесем переменные и функции, значения которых дублируются во всех ВМ, и функции, значения которых распределены между различными ВМ в системе. Особенностью переменных и функций с дублирующими значениями является то, что каждое присвоение им какого-либо нового значения в любой ветви алгоритма обязательно должно сопровождаться последующей пересылкой этого значения во все ВМ системы с целью замены во всех ветвях параллельного алгоритма старого значения рассматриваемой функции или переменной на новое. Ко второй группе отнесем функции и переменные, значения которых в каждой ВМ изменяются локально и независимо от других ВМ. Кроме того, выделим класс системных переменных, используемых для обозначения номеров ВМ, которые также могут быть как глобальными, так и локальными переменными.

Во избежание путаницы локальные функции и переменные будем обозначать с чертой наверху, например, $\overline{b}$, $\overline{CT}_{\overline{a},\overline{b}}$. Системные переменные будем обозначать малыми буквами греческого алфавита. Глобальные переменные и функции с дублирующимися значениями будут обозначаться обычным образом, без черты, например, B_c, КМ, $D_{a,n,\overline{k}}$, а глобальные функции с распределенными значениями между ВМ системы по правилам, которые описываются ниже.

Рассматривается способ распределения значений глобальных функций по одному из ее аргументов. Пусть $Q_{x,\ldots,y,\ldots,z}$ - некоторая функция, значения которой в параллельном алгоритме должны быть распределены по аргументу y. Разобьем множество $y = \{0,\ldots,n-1\}$ возможных значений аргумента y на N подмножеств

$y_0,\ldots,y_{N-1}$ по числу ВМ в виртуальной ОВС так, что $y_0 = \{0,\ldots,n_1-1\}\ldots$, $y_\alpha = \{n_{\alpha-1},\ldots,n_\alpha-1\},\ldots$, $y_N = \{n_{N-1},\ldots,n_N-1\}$, а $n_\alpha = \sum_{0\le\beta<\alpha} \Delta(\beta,n,N)$, где $\Delta(\beta,n,N)$ – количество элементов в подмножестве y_β. Для $\Delta(\beta,n,N)$ примем следующую формулу:

$$\Delta(\beta,n,N) = \begin{cases} n/N + 1 & \text{при } \alpha < r, \\ n/N & \text{при } \alpha \ge r, \end{cases}$$

где / – операция получения целой части от деления n на N, а r – остаток от деления n на N.

Перенумеруем элементы подмножества y_α, начиная с 0, и назовем эти номера локальными значениями аргумента y в α-й ВМ системы. Локальное значение $\bar{y}$ в α-й ВМ определяется следующим образом:

$$\bar{y} = \begin{cases} y & \text{при } \alpha=0; \\ y-n_{\alpha-1} & \text{при } \alpha > 0, \end{cases}$$

где y – исходные (глобальные) значения аргумента. Рассматриваемый способ разбиения множества значений аргумента позволяет однозначно по его глобальному значению y при заданных n и N определить пару $\bar{y},\alpha$, и наоборот, по известным $\bar{y},\alpha$ определить y. В дальнейшем эти операции будут называться операциями определения номера ВМ, определения локального значения и получения глобального значения и будут обозначаться в следующем виде: $\bar{y} = y'$, $\alpha = 'y$, $y = \alpha \circ \bar{y}$. Нетрудно показать, что они определяются следующими формулами:

$$'y = \begin{cases} [y/(q+1)] & \text{при } y < r(q+1); \\ r + [y-r(q+1)]/q & \text{при } y \ge r(q+1); \end{cases}$$

$$y' = \begin{cases} y\%(q+1) & \text{при } y < r(q+1); \\ [y - r(q+1)]\%q & \text{при } y \ge r(q+1); \end{cases}$$

$$\alpha \circ \bar{y} = \begin{cases} \alpha(q+1)+\bar{y} & \text{при } \alpha < r; \\ \alpha q+r+\bar{y} & \text{при } \alpha \ge r, \end{cases}$$

где $q = n/N$, $r = n\%N$, / – операция получения целой части от деления n на N, а % – операция получения остатка от деления.

Глобальные функции с распределенными значениями в ветви α параллельного алгоритма будем обозначать в виде $Q_{a,\ldots,\bar{b},\ldots,'}$, где Q – функция, значения которой распределены по аргументу $\bar{b}$, $\bar{b}$ – аргумент функции, принимающей локальные значения в ветви α.

Если b' - глобальный номер рассматриваемого аргумента, то допускается обозначение $Q_{a,\ldots,b',\ldots,1}$.

Для описания системных взаимодействий между ветвями алгоритма вводятся операции чтения из другой ветви, записи в другую ветвь, трансляционного обмена, рассылки, сборки, конвейерного обмена, дублирования, сортировки и синхронизации его ветвей. Каждая из этих операций, подобно операции присваивания, имеет элементарную и групповую формы. Групповая форма этих операций из элементарной образуется так же, как и в случае операции присваивания, поэтому мы подробно остановимся здесь только на описании элементарных форм рассматриваемых операций.

I. Операция элементарного чтения из другой ветви записывается в виде чт(a,p,b) , где a - функция или переменная рассматриваемой ветви параллельного алгоритма, в которой эта операция употреблена, p - переменная, значение которой - номер ветви алгоритма, b - переменная или функция его p-й ветви. Выполнение операции в рассматриваемой ветви заключается в присвоении a значения переменной или функции b из p-й ветви.

2. Операция элементарной записи в другую ветвь имеет вид: зп (p,b,a) , где a,b и p имеют тот же смысл, что и в операции чтения. Выполнение операции записи в рассматриваемой ветви заключается в присвоении переменной или функции b из p-й ветви значения переменной или функции a из рассматриваемой ветви.

3. Операция элементарного трансляционного обмена имеет вид: трансляция (a,p,b), где a,b и p имеют тот же смысл, что и в предыдущих операциях. Эта операция предназначена для употребления одновременно во всех или в большинстве ветвей, так как в отличие от двух предыдущих операций описывает коллективное взаимодействие ветвей. Выполнение операции в ветви α заключается в следующем. Вначале сравнивается собственный номер α с номером p (который во всех ветвях, где эта операция встречается, должен быть одинаков). В случае $\alpha = p$ ветвь α - передающая, поэтому осуществляется передача значения b в систему связи виртуальной ОВС и, кроме того, это значение присваивается своей переменной или функции a. В случае $\alpha \neq p$ происходит обращение в систему связи, которая либо сразу выдает значение b (если оно к данному моменту времени передающей ветвью было уже передано в систему связи), либо вначале из передающей ветви читает это значение и только после этого передает его в запрашивающую ветвь.

4. Операция элементарной рассылки так же, как и операция трансляционного обмена описывает коллективное взаимодействие ветвей. В ветви α она имеет вид: рассылка (a,p,b(α o i)),где a и p имеют прежний смысл, а b(α o i) - функция, один из аргументов которой имеет вид α o i, где α - номер ВМ, i - переменная, принимающая локальные значения рассматриваемого аргумента функции b, o - описанная ранее операция получения глобального значения аргумента. Выполняется операция рассылка в ветви α следующим образом. Вначале сравнивается собственный номер α с номером p.В случае α = p ветвь передающая, поэтому осуществляется передача N-1 значений b(β o i) при β = 0,...,α-1,α+1,..., N-1 в систему связи, и одновременно значение b(α o i) присваивается своей переменной или функции a. В случае $\alpha \neq$ p происходит обращение в систему связи, которая, как и в предыдущем случае, обеспечивает передачу в запрашивающую ветвь соответствующего значения b .

5. Операция элементарной сборки также является операцией коллективного взаимодействия ветвей алгоритма. В ветви α она имеет вид: сборка (p,b(α oi),a), где p,a,b(α oi) имеют тот же смысл, что и в предыдущем случае. При выполнении операции в ветви α,если α = p, то ветвь α является принимающей и поэтому она посылает в систему связи запрос на прием информации. Если $\alpha \neq$ p, то значение своей переменной или функции a передается в сеть связи, которая затем обеспечивает передачу его в принимающую ветвь после того, как принимающая ветвь начнет у себя выполнение этой операции и передаст в систему связи запрос на прием информации.Значение a из ветви с номером β присваивается функции b при значении ее аргумента, равном β o i.

6. Операция элементарного конвейерного обмена является операцией коллективного взаимодействия ветвей параллельного алгоритма.В ветви α она имеет вид: конвейер (a), где a - некоторая функция или переменная. При выполнении операции в ветви α значение a выдается в систему связи, которая передает его ветви α + 1 (ветви I, если α = N-1). После выдачи в систему связи этого значения от нее принимается значение a из соседней ветви, которое затем присваивается своей переменной или функции a . Операция конвейер очень эффективна, так как при ее выполнении одновременно осуществляется N обменов.

7. Операция элементарного дублирования является операцией коллективного взаимодействия ветвей рассматриваемого алгоритма. В ветви α она имеет вид: дублирование (a(β o i), b), 0 $\leq \beta <$ N, где

a (βoi) - функция, один из аргументов которой имеет вид βoi , β - номер ветви, i - локальный номер рассматриваемого аргумента, b - функция или переменная, N - число ветвей алгоритма. Результат выполнения операции аналогичен выполнению цикла Ц [трансля - ция (a(βoi), β,b)].
$0 \le \beta < N$

8. Операция элементарной сортировки является операцией коллективного взаимодействия ветвей. В ветви α она имеет вид: сортировка (a(βoi), b(αoj)), $0 \le \beta < N$, либо вид: сортировка (a(αoi), b(βoj)), $0 \le \beta < N$, где использованы известные обозначения. Результат выполнения операции аналогичен выполнению цикла Ц [рассылка (a(βoi),β,b(αoj))] либо цикла Ц [сборка (a(αoi), b(βoj))].
$0 \le \beta < N$ $0 \le \beta < N$

9. Операция синхронизации имеет вид: синхронизация. Выполнение любой ветви на этой операции приостанавливается до тех пор, пока эта операция не встретится во всех остальных ветвях. Таким образом осуществляется синхронизация выполнения всех ветвей алгоритма во времени, когда требуется, чтобы они в определенных точках процесса функционирования выполнение своих операций начинали одновременно. Эта особенность операции синхронизации требует, чтобы выполнялось следующее условие корректности алгоритма: его ветви должны быть устроены так, что если в какой-либо ветви при ее выполнении встречается операция синхронизации, то все другие ветви рано или поздно также должны выйти на эту операцию.

Основными источниками параллелизма алгоритмов на языке формализации являются операции группового присваивания и групповые блоки, так как в этих конструкциях вычисления для различных значений переменных, по которым они образуются, могут осуществляться независимо друг от друга в том числе и одновременно. "Распараллеливание" этих конструкций будем осуществлять по одной из таких переменных следующим образом. Пусть x - одна из переменных данной операции группового присваивания или данного группового блока. Области возможных значений x {0,...,n-1} разбиваются на N частей δ_α, $\alpha = 0,\ldots,N$, так, как это было описано выше при рассмотрении способа распределения значений глобальных функций по одному из ее аргументов. К ветви α отнесем все вычисления для значений $x \in \delta_\alpha$, т.е. таких, что 'x = α.

Для примера рассмотрим операцию группового присваивания

$$c_{ij} \doteq \sum_{1 \le k \le 8 g_{ij}} a_{ik} \times b_{kj};\ m \le j \le n,\ p \le i \le q, \qquad (1)$$

где a_{ij}, b_{ij}, c_{ij} - числовые функции, e,m,n,p,q - переменные, а g_{ij} - функция с целочисленными значениями, i,j,k - переменные с областями возможных значений {0,...,I-1}, {0,...,J-1}, {0,...,K-1} соответственно. На значениях переменных e,m,n,p,q и функции g_{ij} наложены условия: $0 \le e,\ 8g_{ij} < K,\ 0 \le m,n < J,\ 0 \le p,q < I$.

Положим, что значения функций a_{ij} должны быть распределены между ветвями алгоритма по аргументу i, значения функций c_{ij}, g_{ij} - по аргументу j, значения b_{kj} - по аргументу j. Значения e,m,n,p,q должны дублироваться во всех его ветвях. "Распараллеливание" рассматриваемой операции будем осуществлять по i. При сделанных предположениях ветвь α параллельного алгоритма ($\alpha = 0,\ldots,N-1$), реализующего последовательный алгоритм (I), может быть записана в виде алгоритма (2):

$$\overline{EI} = \Delta(\alpha, I, N)\ \pi(\bar{p}, p, \overline{EI})\ R(\bar{q}, q, \overline{EI})$$

$$\mathop{\Box}_{\bar{p} \le \bar{i} \le \bar{q}} \left[\mathop{\Box}_{m \le \bar{j} \le n} \left[\begin{array}{l} \underline{\text{рассылка}}\ (\bar{e}, 'j, g_{\alpha o \bar{i},\bar{j}})\ \bar{g} \doteq 8 \times \bar{g}\ \bar{h} \doteq 0 \\ \mathop{\Box}_{1 \le \bar{k} \le \bar{g}} \left[\begin{array}{l} \bar{a} \doteq a_{\bar{i}\bar{k}}\ \ \underline{\text{трансляция}}\ \ (\bar{b}, 'k, b_{\bar{k},\bar{j}}) \\ \bar{h} \doteq \bar{h} + \bar{a} \times \bar{b} \end{array} \right] \\ \underline{\text{сборка}}\ ('j, c_{\alpha o \bar{i},\bar{j}}, \bar{h}) \end{array} \right] \right]. \qquad (2)$$

Здесь локальная переменная $\bar{I}$ в ветви α пробегает допустимые значения, удовлетворяющие локальному условию $\bar{p} \le \bar{I} \le \bar{q}$, которое определяет пределы изменения значений переменной $\bar{I}$ в цикле. Значения $\bar{p}$ и $\bar{q}$ в каждой ветви вычисляются с помощью алгоритмов $\pi(\bar{p}, p, \overline{EI})$ и $R(\bar{q}, q, \overline{EI})$. Алгоритм π имеет следующий вид:

$\bar{p} \doteq p'$ при $\alpha = 'p$ иначе 0 при $\alpha > 'p$ иначе $\overline{EI}$,

а алгоритм R:

$\bar{q} \doteq q'$ при $\alpha = 'q$ иначе $\overline{EI} - 1$ при $\alpha < 'q$ иначе -1,

где $\overline{EI}$ - максимально возможное значение локальной переменной $\bar{I}$ в ветви α, определяемое с помощью ранее описанной операции $\Delta(\alpha, I, N)$. Способ определения $\bar{p}$ и $\bar{q}$ иллюстрируется табл. I и 2.

Таблица I

Значения $\bar{I} = i'$, $\alpha = 'i$ при I = I6, N = 5

i	0	I	2	3	4	5	6	7	8	9	I0	II	I2	I3	I4	I5
$\bar{I}$	0	I	2	3	0	I	2	0	I	2	0	I	2	0	I	2
α	0	0	0	0	I	I	I	2	2	2	3	3	3	4	4	4

Таблица 2

Значения $\bar{p}, \bar{q}, \overline{EI}$ при I = I6, N = I2

α	$\bar{p}$	$\bar{q}$	$\overline{EI}$
0	4	3	4
I	2	2	3
2	0	2	3
3	0	2	3
4	0	-I	3

Для рассматриваемого в табл. I и 2 случая (I = I6, N = 5, p = 6, q = I2) значения $\bar{i}$ соответствующие первой и пятой ветви алгоритма, находятся за пределами, определяемыми условием $p \le i \le q$. Поэтому согласно принятому способу распределения вычислений в этих ветвях не должно выполняться каких-либо операций. Это достигается тем, что значение p в таких случаях полагается больше значения $\bar{q}$, и тогда локальное условие $\bar{p} \le \bar{i} \le \bar{q}$ дает пустое множество допустимых значений $\bar{I}$.

При организации в (2) цикла по $\bar{k}$ необходимо определить верхний предел изменения значений переменной цикла $\bar{k}$ в ветви α в соответствии с глобальным условием, определяющим пределы изменения значений переменной суммирования k: $e \le k \le 8g_{ij}$, кроме того, нужно задать начальное значение, равное 0, переменной $\bar{h}$, в которой последовательно накапливается результат суммирования произведений значений функций a и b для различных $\bar{k}$. Верхний предел $\bar{g}$ изменения переменной цикла $\bar{k}$ в ветви α определяется с помощью алгоритма, имеющего вид: рассылка $(\bar{g}, 'j, g_{\alpha\circ\bar{i},\bar{j}})$ $\bar{g} = 8 \times \bar{g}$, где используется операция системного взаимодействия рассылка, так как в этом случае разными ветвями запрашиваются разные значения функции $g_{\alpha\circ\bar{i},\bar{j}}$, $\alpha = 0, \ldots, N-1$, но находящимися в одной ветви с номером $'j$. В теле цикла по $\bar{k}$ применена операция системного взаимодействия трансляция, так как в этом случае всеми ветвями запрашивается одно и то же значение $b_{\bar{k},\bar{j}}$ из ветви с номером $'\bar{k}$.

Недостатком параллельного алгоритма (2) является выполнение большого количества операций системного взаимодействия: на каждое пересылаемое значение - своя операция. Доля времени, которое уходит на обмен информации между ветвями, в таком алгоритме очень

большая. Неприемлемо большой она становится при реализации параллельного алгоритма на распределенных вычислительных системах с использованием метода коммутации пакетов. В этом случае для передачи каждого значения будет организовываться свой пакет.

Алгоритм (2) может быть преобразован в параллельный алгоритм с меньшим количеством операций системного взаимодействия за счет объединения в одной операции множества пересылок. Это можно сделать путем использования эквивалентных преобразований алгоритма (2), связанных с расщеплением каждого из циклов $\coprod_{m \le j \le n}$ и $\coprod_{e \le k \le \bar{g}}$ на два вложенных друг в друга цикла и эквивалентных преобразований, связанных с выносом операций системного взаимодействия за цикловые скобки. В результате применения указанных преобразований получается параллельный алгоритм (3):

$$\overline{EI} \doteq \Delta(\alpha,\ I,\ N)\ \pi(\bar{p},\ p,\ \overline{EI})\ R(\bar{q},\ q,\ \overline{EI})$$

$$\coprod_{\bar{p} \le \bar{i} \le \bar{q}} \left[\coprod_{0 \le \bar{\beta} < N} \left[\begin{array}{l} \overline{EJ} \doteq \Delta(\bar{\beta}, J, N)\ \pi(\bar{m}, m, \overline{EJ})\ R(\bar{n}, n, \overline{EJ}) \\ \underline{\text{рассылка}}\ (\bar{g}_{\bar{j}},\ \bar{\beta}, g_{\alpha o \bar{i}, \bar{j}});\ \bar{m} \le \bar{j} \le \bar{n} \\ \coprod_{m \le \bar{j} \le n} \left[\begin{array}{l} \bar{h}_{\bar{j}} \doteq 0 \quad \bar{g}_{\bar{j}} \doteq 8 \times \bar{g}_{\bar{j}} \\ \coprod_{0 \le \bar{\gamma} \le N} \left[\begin{array}{l} \overline{EK} \doteq \Delta(\bar{\gamma}, K, N)\ \pi(\bar{e}, e, \overline{EK})\ R(\bar{r}, \bar{g}_{\bar{j}}, \overline{EK}) \\ \underline{\text{трансляция}}\ (\bar{b}_{\bar{k}}, \bar{\gamma}, b_{\bar{k}}, \bar{\beta} o \bar{j});\ \bar{e} \le \bar{k} \le \bar{r} \\ \coprod_{\bar{e} \le \bar{k} \le \bar{r}} \left[\bar{a} \doteq a_{\bar{i}, \bar{\gamma} o \bar{k}}\ \ \bar{h}_{\bar{j}} \doteq \bar{h}_{\bar{j}} + \bar{a} \times \bar{b}_{\bar{k}} \right] \end{array} \right] \end{array} \right] \\ \underline{\text{сборка}}\ (\bar{\beta}, c_{\alpha o \bar{i}, \bar{j}}, \bar{h}_{\bar{j}});\ \bar{m} \le j \le \bar{n} \end{array} \right] \right] \qquad (3)$$

Алгоритм (3) можно улучшить, если операцию сборка, а также операцию рассылка совместно с операциями определения $\overline{EJ}$, $\bar{m}$ и $\bar{n}$ вынести за скобки цикла по $\bar{\beta}$. Но в этом случае операции сборка и рассылка превращаются в операцию сортировка:

$$\left. \begin{array}{l} \overline{EJ}_{\beta} \doteq \Delta(\bar{\beta}, J, N);\ 0 \le \bar{\beta} < N, \pi(\bar{m}_{\beta}, m, \overline{EJ}_{\beta});\ 0 \le \bar{\beta} < N, \\ R(\bar{n}_{\beta},\ n, \overline{EJ}_{\beta});\ 0 \le \beta < N, \\ \underline{\text{сортировка}}\ (\bar{g}_{\beta o \bar{j}}, g_{\alpha o \bar{i}, \bar{j}}),\ 0 \le \beta < N;\ \bar{m}_{\beta} \le \bar{j} \le \bar{n}_{\beta}, \end{array} \right\} \qquad (4)$$

$$\coprod_{0\le\bar\beta<N}[\ ],$$

сортировка $(c_{\alpha o\bar i,\bar j},\bar h_{\beta o\bar j})$, $0\le\bar\beta<N$; $\bar m\le\bar j\le\bar n$.

Другой путь уменьшения числа операций системного взаимодействия заключается в применении к (2) перестановки циклов по $\bar I$ и $\bar\beta$ с последующим выносом за скобки цикла по $\bar i$ операции рассылка совместно с операциями определения $\overline{EJ},\bar m,\bar n$:

$$\overline{EI}\doteq\Delta(\alpha,I,N)\ \pi(\bar p,p,\overline{EI})\ R(\bar q,q,\overline{EI}),$$

$$\coprod_{0\le\bar\beta<N}\left[\begin{array}{l}\overline{EJ}\doteq\Delta(\bar\beta,J,N),\ \pi(\bar m,m,\overline{EJ}),R(\bar n,n,\overline{EJ}),\\ \text{рассылка}\ (\bar g_{\bar i\bar j},\bar\beta,g_{\alpha o\bar i,\bar j});\ \bar m\le\bar j\le\bar n,\ \bar p\le\bar I\le\bar q,\\ \coprod\limits_{\bar p\le\bar I\le\bar q}\left[\coprod\limits_{\bar m\le\bar j\le\bar n}[\ ]\right].\end{array}\right] \qquad (5)$$

Операцию трансляция, применяя те же преобразования, что и в случае с операцией рассылка в (4), можно вынести за скобки цикла по $\bar\gamma$. В результате преобразования алгоритма (3) она превращается в операцию дублирование:

$$\bar h_{\bar j}\doteq 0\ \bar g_{\bar j}\doteq 8\times\bar g_{\bar j},$$

$$\overline{EK}_{\bar\gamma}\doteq\Delta(\bar\gamma,K,N);\ 0\le\bar\gamma<N\ \pi(\bar e_{\bar\gamma},e,\overline{EK}_{\bar\gamma});\ 0\le\bar\gamma<N;$$

$$R(\bar r_{\bar\gamma},\bar g_{\bar j},\overline{EK}_{\bar\gamma});\ 0\le\bar\gamma<N, \qquad (6)$$

дублирование $(\bar b_{\gamma o\bar k},\bar\gamma,b_{\bar k,\beta o\bar j})$, $0\le\bar\gamma<N$; $\bar e_{\bar\gamma}\le\bar k\le\bar r_{\bar\gamma}$,

$$\coprod_{0\le\gamma<N}[\ ].$$

Операцию трансляция в (3) путем установления в каждой ветви определенного порядка перебора значений $\bar\gamma$ в цикле по $\bar\gamma$, а именно, начиная с $\bar\gamma=\alpha+1$ и кончая значением $\bar\gamma=\alpha$, можно преобразовать также в операцию конвейер:

$$\bar h_{\bar j}\doteq 0\ \bar g_{\bar j}\doteq 8\times\bar g_{\bar j},$$

$$\overline{EK}\doteq\Delta(\alpha,K,N)\ \pi(\bar e,e,\overline{EK})\ R(\bar r,\bar g_{\bar j},\overline{EK}),$$

$$\bar b_{\bar k}\doteq b_{\bar k,\beta o\bar j};\ \bar e\le\bar k\le\bar r,$$

$$\bigcup_{\bar{\gamma}=\alpha+1+N-1,\ 0+\alpha}\left[\underline{\text{конвейер}}\ (\bar{e})\ \underline{\text{конвейер}}\ (\bar{r})\ \underline{\text{конвейер}}\ (\bar{b}_{\bar{\gamma}});\ \bar{e}\le\bar{k}\le\bar{r}\right]$$
$$\bigcup_{\bar{e}\le\bar{k}\le\bar{r}}\left[\bar{a}=a_i,\bar{\gamma}o\bar{k}\quad \bar{h}_j=\bar{h}_j+\bar{a}\times\bar{b}_{\bar{k}}\right].$$

Результатом рассмотренных выше эквивалентных преобразований является уменьшение числа операций системного взаимодействия во время выполнения параллельного алгоритма. Это приводит к уменьшению общего времени его выполнения. Однако этот эффект достигается за счет увеличения в каждой ВМ системы памяти для хранения передаваемой информации из-за замены локальной переменной $\bar{g}$ на локальные функции $\bar{g}_j$ в (3), $\bar{g}_{\beta o\bar{j}}$ в (4) и $\bar{g}_{\bar{\gamma}\bar{j}}$ в (5), замены локальной переменной $\bar{b}$ на функции $\bar{b}_{\bar{k}}$ в (3) и $\bar{b}_{\bar{\gamma}o\bar{k}}$ в (6), замены локальной переменной $\bar{h}$ на локальные функции $\bar{h}_{\bar{j}}$ в (3), $\bar{h}_{\beta o\bar{j}}$ в (4). Поэтому в каждом конкретном случае следует выбирать тот или иной вид параллельного алгоритма в зависимости от того, что нужно экономить память ВМ или время его выполнения.

Эквивалентные преобразования алгоритма, связанные с преобразованием операции трансляция в операцию конвейер, уменьшают время, затрачиваемое на обмен информацией между ветвями за счет большей эффективности операции конвейер и не приводят при этом к необходимости увеличения памяти ВМ. Поэтому такие преобразования являются наиболее желательными.

С целью иллюстрации способа получения параллельного алгоритма при распараллеливании группового блока по его переменной рассмотрим следующий пример такого блока:

$$\Gamma_{0\le i<I}\left[\begin{array}{l} c\doteq a_i\times h+b_i;\ e\doteq a_i\times g+b_i \\ d_{ie}\doteq A_{ie}\end{array}\right], \qquad (7)$$

где a_i, b_i - функции с целочисленными значениями, которые должны быть распределены по аргументу i (область возможных значений аргумента i есть ряд $0,\ldots,I-1$), d_{ie}, A_{ic} - числовые функции, значения которых должны быть распределены по аргументам e и c соответственно, c, e - переменные, которым в алгоритме должны соответствовать локальные переменные $\bar{c},\bar{e}$, и, наконец, h, g - переменные, которым в нем должны соответствовать глобальные переменные с дублирующимися значениями. Алгоритм ветви α ($\alpha = 0,\ldots,N-1$) для (7) имеет вид:

$$\overline{EI} \doteq \Delta(\alpha,I,N)\ R(\bar{q},\bar{I},\overline{EI}),$$

$$\underset{0\le i\le q}{Ц}\left[\begin{array}{l}\bar{a} \doteq a_{\bar{I}}\ \bar{b} \doteq b_{\bar{I}}\ \bar{c} \doteq \bar{a}\times h + \bar{b}\ \ \bar{e} = \bar{a}\times g + \bar{b}\\ \underline{чт}(\bar{f},'\bar{c},A_{\alpha o\bar{I},\bar{c}},)\ \underline{зп}('\bar{e},d_{\alpha o\bar{I}},\bar{g},\bar{f})\end{array}\right].$$

Алгоритм ветви α операции элементарного присваивания зависит от того, какой способ принят для реализации параллельного алгоритма: с дублированием вычислений во всех его ветвях или с вычислением в одной ветви. Например, рассмотрим операцию элементарного присваивания $c \doteq \sum_{0\le j<M} b_j$, где c - переменная, значение которой в алгоритме должно дублироваться, значения функции b_j должны быть распределены по j. Алгоритм ветви α ($\alpha = 0,\dots,N-1$) с дублированием вычислений имеет вид:

$$c \doteq 0\quad \underset{0\le j<M}{Ц}\left[\ \underline{чт}(\bar{b},'\bar{j},b_{\bar{j}})\ c \doteq c + \bar{b}\ \right],$$

а с вычислениями в одной ветви

$$\underline{если}\ \alpha=\gamma\ \underline{то}\ \left[\bar{c}\doteq 0\ \underset{0\le j<M}{Ц}[\underline{чт}(\bar{b},'\bar{j},b_{\bar{j}},)\ \bar{c}\doteq\bar{c}+b]\ \underline{зп}(\bar{\beta},c,\bar{c});\ 0\le\bar{\beta}<N\ \right],$$

где γ - номер ветви, в которой должны проводиться вычисления.

При необходимости может быть использован параллельный способ вычислений терма, если терм в правой части операции элементарного присвоения образован с помощью оператора. Для примера операции $c \doteq \sum_{0\le j<M} b_j$ ветвь α ($\alpha = 0,\dots,N-1$) с параллельным способом вычисления терма имеет вид:

$$\overline{EJ} \doteq \Lambda(\alpha,M,N)\ R(\bar{q},M,\overline{EJ})\ \bar{h} \doteq 0$$

$$\underset{0\le j\le\bar{q}}{Ц}[\bar{b} \doteq b_j\ \bar{h} = \bar{h} + \bar{b}]\ \underline{\text{синхронизация}}$$

$$\underline{если}\ \alpha=\gamma\ \underline{то}\ [\bar{c}\doteq 0\ \underset{0\le\bar{\beta}<N}{Ц}[\ \underline{чт}(\bar{r},\bar{\beta},\bar{h})\ \bar{c}\doteq\bar{c}+\bar{r}]\ \underline{зп}(\bar{\beta},c,\bar{c});\ 0\le\bar{\beta}<N],$$

где γ - номер ветви, в которой осуществляется сложение частичных сумм $\bar{h}$, полученных в каждой ветви, а операция синхронизация используется перед переходом к сложению частичных сумм.

Способ получения ветви α для исходного последовательного алгоритма в целом проиллюстрируем на примере алгоритма решения системы n линейных уравнений $\sum_{0\le k<n} a_{ik}x_k = b_i$, $0 \le i < n$, с n неизвестными x_k методом Гаусса по схеме Жордана:

$$\mathop{\text{Ц}}_{0\le m<n}\left[\begin{array}{l}\text{г}\\ {\scriptstyle 0\le i<n}\end{array}\left[\begin{array}{l} a_{ij} \doteq a_{ij}/a_{mm} \text{ при } i=m \text{ иначе } a_{ij} \times a_{mm} - a_{im} \times a_{mj};\ m\le j\le n \\ b_i \doteq b_i/a_{mm} \text{ при } i=m \text{ иначе } b_i \times a_{mm} - a_{im} \times b_m \end{array}\right]\right],$$

$x_i \doteq b_i$; $0 \le i < n$,

где значения a_{ij}, b_i, x_i при выполнении алгоритма должны распределяться по аргументу i, значение переменной n должно дублироваться, а переменной m в ветвях параллельного алгоритма должна быть сопоставлена локальная переменная $\bar{m}$. Ветвь с номером α (α=0 ,... ...,N-1) рассматриваемого параллельного алгоритма имеет вид:

$\bar{e} \doteq \Delta(\alpha,n,N)\ R(\bar{q},n,\bar{e})$

$$\mathop{\text{Ц}}_{0\le m<n}\left[\begin{array}{l} \mathop{\text{Ц}}_{0\le \bar{i}\le q}\left[\begin{array}{l} \mathop{\text{Ц}}_{m\le j<n}\left[\begin{array}{l} \bar{a}\doteq a_{\bar{i}j} \text{ трансляция } (\bar{k},'\bar{m},a_{\bar{m}j},-)\ \bar{c}\doteq a_{\bar{i}m} \\ \text{трансляция } (\bar{d},'\bar{m},a_{\bar{m}j},-) \\ a_{ij}\doteq \bar{a}/\bar{k} \text{ при } \alpha \circ \bar{i}=\bar{m} \text{ иначе } \bar{a}\times\bar{k} - \bar{c}\times\bar{d} \end{array}\right] \\ \bar{b}\doteq b_{\bar{i}} \text{ трансляция } (\bar{f},'\bar{m},b_{\bar{m}},-)\ b_{\bar{i}}\doteq \bar{b}/\bar{k} \text{ при } \alpha\circ\bar{i}=\bar{m} \\ \quad \text{иначе } \bar{b}\times\bar{k} - \bar{c}\times\bar{f} \end{array}\right] \\ \text{синхронизация} \end{array}\right].$$

С использованием описанных выше эквивалентных преобразований, связанных с выносом операций трансляция за скобки циклов по $\bar{j}$ и по $\bar{i}$, может быть получен более эффективный параллельный алгоритм.

Литература

1. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Универсальные однородные вычислительные системы высокой производительности. -Новосибирск: Наука, 1966. - 308 с.

2. КОСАРЕВ Ю.Г. Распараллеливание по циклам. -В кн.: Вычислительные системы, вып. 24. Новосибирск, 1967, с. 3-20.

3. КОСАРЕВ Ю.Г. О схемах обмена между ветвями параллельных алгоритмов. -В кн.: Вычислительные системы, вып. 51, Новосибирск, 1972, с. 70-76.

4. КОСАРЕВ Ю.Г. Об информационных и пространственно-временных преобразованиях алгоритма. -В кн.: Вычислительные системы, вып. 57, Новосибирск, 1973, с. 149-160.

5. МИРЕНКОВ Н.Н. Структурное параллельное программирование. - Программирование, 1975, № 3, с. 3-14.

6. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на однородных вычислительных системах. -В кн.: Вычислительные системы, вып. 57, Новосибирск, 1973, с. 3-32.

Поступила в ред.-изд.отд.
10 сентября 1982 года