Левин И.И.

СТРУКТУРНО-ПРОЦЕДУРНАЯ ОРГАНИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

НИИ многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета, г. Таганрог

E-mail:levin@mvs.tsure.ru

Научно-технический прогресс неуклонно выдвигает все более высокие требования к производительности вычислительных средств. Это связано с потребностью решения сложных научно-технических задач больших размерностей. Однопроцессорные вычислительные системы уже не справляются с решением большинства таких задач в реальном времени, поэтому для повышения производительности вычислительных систем все чаще используются суперкомпьютеры с параллельной организацией вычислений. Наибольший вклад в развитие вычислительных средств всегда вносили технологические решения, при этом основополагающим параметром быстродействия вычислительных систем являлась тактовая частота элементной базы. Неслучайно поколения вычислительной техники отчетливо связывают с поколением элементной базы. В то же время, начиная с 60-х годов прошлого века, активно развивались параллельные архитектуры и методы организации параллельных вычислений. Параллельные вычисления позволили существенно превзойти уровень производительности вычислительных средств, достигнутых за счет элементной базы [1].

Бурное развитие технологии СБИС и разработка новых поколений микропроцессоров в последнее десятилетие несколько замедлили поиски принципиально новых архитектурных решений. Однако нельзя не признать, что чисто технологические решения в вычислительной технике утратили свое монопольное положение. Так, например, в ближайшей перспективе заметно возрастает значение проблемы преодоления семантического разрыва между аппаратными средствами и методами программирования. Данная проблема может быть решена архитектурными средствами, при этом роль технологии является косвенной (высокая степень интеграции СБИС создает условия для реализации новых архитектурных решений) [2]. Анализ тенденции развития вычислительной техники показывает, что без кардинальной перестройки архитектурных принципов поддерживать интенсивные темпы развития средств вычислительной техники уже невозможно. По самым оптимистическим прогнозам тактовые частоты современных и перспективных СБИС могут быть увеличены в обозримом будущем до

10 ГГц. В то же время, достигнутая степень интеграции позволяет строить параллельные системы, в которых число процессоров может достигать десятков тысяч. В области повышения производительности вычислительных систем резерв технологических решений ограничивается одним порядком. Освоение же массового параллелизма и новых архитектурных решений содержит резерв повышения производительности на несколько порядков. Можно утверждать, что в области исследования архитектуры процессоров и вычислительных систем наблюдается кризис. Тривиальное объединение нескольких тысяч стандартных микропроцессоров не обеспечивает высокой эффективности при решении сложных задач [3, 4, 2].

В настоящее время наиболее распространена мультипроцедурная организация параллельных вычислений. При мультипроцедурной организации вычислений распараллеливание осуществляется по элементам структуры данных и в каждом процессоре обработка ведется по независимой последовательной программе [5, 6]. Программа и данные каждого процессора, как правило, хранятся в его локальной памяти. Для обмена данными между процессорами в системе организуются специальные процедуры.

При разработке программ для многопроцессорных вычислительных систем (МВС) с мультипроцедурным распараллеливанием исходной формой представления задачи является последовательный алгоритм, который распараллеливается по множеству локальных независимых участков. При этом в основном исследуются только информационные взаимодействия смежных частей алгоритма (циклов, процедур и т.п.) и не уделяется должного внимания общей информационной структуре задачи. Высокая реальная производительность систем с мультипроцедурной организацией вычислений достигается при условии, что время обработки информации существенно превышает время выполнения процедур обмена данными, что имеет место при решении на МВС либо множества отдельных задач, либо при решении слабосвязанной задачи. Здесь и далее слабосвязанной задачей мы будем называть задачу, параллельный алгоритм которой требует выполнить пересылок информации существенно меньше (по крайней мере, на порядок), чем количество операций над данными. Сильносвязанной задачей будем называть задачу, в параллельном алгоритме которой количество пересылок информации между процессорами сопоставимо с количеством или больше числа выполняемых операций. Для сильносвязанных задач, класс которых достаточно представителен, имеет место значительное снижение реальной производительности по сравнению с пиковой за счет роста накладного времени (времени, необходимого для организации параллельного процесса).

Существующие параллельные алгоритмы и параллельные программы решения сложных задач на МВС с массовым параллелизмом обеспечивают достаточно низкую реальную производительность вследствие того, что организация параллельных вычислений и процедуры межпроцессорных обменов требуют больше времени, чем непосредственно сами вычисления [4].

Для более детального анализа параллельной программы целесообразно рассмотреть общую структуру задачи – информационный граф алгоритма. В информационном графе задачи вершины соответствуют исходным данным, операциям над данными и результатам информационного преобразования. Дуги информационного графа соответствуют информационным связям алгоритма. В дальнейшем вершины, соответствующие исходным данным и результатам вычислений, мы будем называть информационными вершинами. Вершины, соответствующие операциям над данными, называются операционными вершинами [7]. При описании задачи информационным графом все ее циклические участки итерированы столько раз, сколько должен повторяться этот участок. Если количество итераций неизвестно, то можно представить некоторую итерированную структуру с неизвестным числом повторений [7]. При этом предполагается, что операции алгоритма разбиты на группы, упорядоченные так, что каждая операция любой группы зависит либо от начальных данных алгоритма, либо от результатов выполнения операций, находящихся в предыдущих группах [10, 11]. Максимальная скорость решения задачи в системе будет обеспечена, если создать специализированную систему, структура которой адекватна информационному графу задачи.

В общем виде информационный граф можно представить в виде:

$$G=\langle (X, A_X),(Q, A_Q),(Y, A_Y)\rangle, \qquad (1)$$

где X - множество входных вершин графа,

Y - множество выходных вершин,

Q - множество операционных вершин.

Объединение множеств входных дуг A_X, операционных дуг A_Q и выходных дуг A_Y описывает информационные связи в задаче. Для любой дуги $\langle u,v\rangle$, принадлежащей к A_X, вершина $u \in X$, а вершина $v \in Q$. Для любой дуги $\langle u,v\rangle$, принадлежащей к Q, вершина $u \in Q$ и вершина $v \in Q$. Для любой дуги $\langle u,v\rangle$, принадлежащей к Y, вершина $u \in Q$, вершина $v \in Y$.

Если число элементарных процессоров в системе не меньше числа операционных вершин, и все дуги информационного графа задачи (в том числе, обуславливающие связь между информационными и операционными вершинами) можно реализовать в пространственной

коммутационной системе, то информационный граф задачи тривиальным образом реализуется в многопроцессорной системе. При этом наблюдаются строгие соответствия. Операционная вершина графа будет соответствовать процессору, настроенному на выполнение определенной арифметико-логической операции. Входная информационная вершина должна соответствовать сегменту операционной памяти, в которой предварительно помещены исходные данные. Выходная информационная вершина должна соответствовать сегменту оперативной памяти, в котором по окончании процесса решения задачи окажутся результаты вычислений.

Такую организацию вычислений будем называть **структурной организацией** вычислений. На практике обеспечить структурную организацию вычислений не представляется возможным, так для ее реализации и выполнения требуется чрезвычайно большое количество процессоров. В этой связи необходимо сегментировать информационный граф на пересекающиеся подграфы.

Задача, описываемая в виде информационного графа G, может быть представлена в виде множества крупных функционально законченных вычислительных фрагментов P_j, таких, что если существуют вершина $a \in P_i$ и вершина $b \in P_j$, и существует хотя бы одна дуга $\langle a, b \rangle$, то $i \leq j$. Будем называть дуги, соединяющие подграфы, внешними, а дуги, принадлежащие подграфу P_i, внутренними.

Когда ресурсов системы недостаточно для полного покрытия информационного графа задачи, необходимо разработать механизм последовательной реализации графа задачи с помощью аппаратной реализации функционально законченных подграфов. При этом вычислительной структурой может быть как отдельный подграф информационного графа задачи, так и кортеж изоморфных подграфов задачи при условии, что указанные подграфы являются информационно независимыми (могут выполняться параллельно) или непосредственно информационно зависимыми друг от друга (могут выполняться только последовательно).

Вычислительную структуру, которую образуют кортежи изоморфных подграфов, можно преобразовать в специальную программную конструкцию, которую принято называть **кадром**. Можно сказать, что кадр представляет собой аппаратно-реализованный подграф информационного графа задачи, через который следует поток данных. Входной поток данных образуется с помощью преобразования множества входных вершин подграфов, входящих в кортеж, образующих кадр. Аналогичным образом реализуется выходной поток данных.

Механизм последовательного обхода информационного графа задачи кадрами принято называть **структурно-процедурной** программой. В отличие от мультипроцедурной организации вычислений управляющая программа для структурно-процедурной организации вычислений едина для всей многопроцессорной системы, при этом программно-неделимой единицей программы является структурно (аппаратно) реализованный подграф задачи, через который следует поток операндов. Можно сказать, что МВС со структурно-процедурной организацией вычислений являются гибридом фон-неймановской архитектуры и архитектуры потока данных. Такая организация вычислений, с одной стороны, обеспечивает детерминизм выполнения программы, что, в общем случае, не могут обеспечить многопроцессорные системы, построенные по традиционной мультипроцедурной архитектуре. С другой стороны, МВС со структурно-процедурной организацией вычислений обеспечивает высокую эффективность параллельных вычислений на широком классе задач, что будет показано ниже.

Кадр часто описывается как объект [16], определенный в виде тройки $\langle Q, I, O \rangle$, где Q - функциональный граф кадра, вершиной которого являются операционные вершины, соответствующие ЭП, настроенные на выполнение операций (операции ЭП, которые в ходе реализации кадра не изменяются), а также входные и выходные вершины, которые соответствуют информационным каналам. Входной информационный поток представляет собой кортеж независимых потоков $I = \langle I_1, I_2, ... I_n \rangle$, каждому элементу которого I_j взаимно однозначно соответствует входная вершина функционального графа Q A^j. Аналогично, выходной поток представляет собой кортеж независимых потоков $O = \langle O_1, O_2, ... O_n \rangle$, каждому элементу которого O_i взаимно однозначно соответствует выходная вершина графа Q B^j. Каждый элемент входного информационного потока I_j определяет информационный массив, элементы которого последовательно подаются на входную вершину A^j. Каждый элемент выходного информационного потока O_j определяет информационный массив, элементы которого последовательно считываются с выходной вершины B^j. На рис.1 представлено графическое изображение кадра.

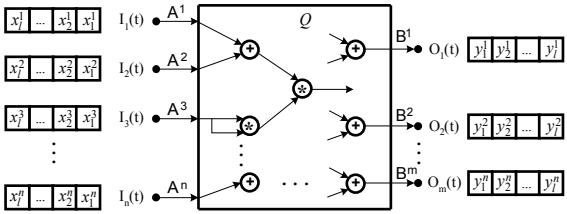


Рис.1. Графическое изображение кадра.

На практике зачастую понимается, что информационные потоки, входящие в кадр, однозначно определены и соответствуют появлению входных данных в заданном временном масштабе, где время определено как физический параметр. Однако такое представление информационных потоков кадра является частным случаем.

Элементарным кадром будем называть кадр, в котором на каждый вход функционального графа кадра подается только один операнд и с каждого выхода снимается только один результат. При этом элементарный кадр определяется тройкой $\langle Q, A, B \rangle$, где A и B - это множества входных и выходных вершин графа.

Несложно заметить, что вышеприведенное определение кадра соответствует кортежу элементарных кадров $\langle K_1, K_2, ..., K_l \rangle$, где l - размер информационных потоков на входе (выходе) кадра. Тогда множества входных и выходных вершин элементарного кадра $K_i = \langle Q, X_i, Y_i \rangle$ можно определить следующим образом

$$X_i = I(t_i),$$
$$Y_i = O(t_i).$$

Можно утверждать, что если имеется кортеж элементарных кадров, каждый элемент которого обладает изоморфным функциональным графом, то такой кортеж можно представить в виде исполняемого кадра.

В самом деле, если имеется кортеж $\langle K_1, K_2, ..., K_l \rangle$ элементарных кадров, где $K_j = \langle Q_j, X_j, Y_j \rangle$ и $Q_1 \cong Q_2 \cong ... \cong Q_n$, то можно сформировать кортеж множеств входных вершин $X = \langle X_1, X_2, ... X_l \rangle$, в котором *j*-й элемент кортежа соответствует множеству входных вершин *j*-го элемента кортежа элементарных кадра, и кортеж выходных вершин $Y = \langle Y_1, Y_2, ... Y_l \rangle$, в котором *j*-й элемент кортежа соответствует множеству выходных вершин *j*-го элемента кортежа кадра.

Тогда правило упорядочивания элементарных кадров в кортеж однозначно определяет порядок входных и выходных информационных вершин. Обращаясь к кортежу X и Y по индексу, можно получить требуемый элемент и, следовательно, информационные потоки можно определить как функции обращения к информационным массивам.

Такой подход позволяет значительно расширить применение структурных и структурно-процедурных методов организации параллельных вычислений для задач различных проблемных областей и увеличить эффективность распараллеливания, а, как следствие, и существенно сократить время и стоимость решения задачи на МВС. Таким образом, информационные потоки, поступающие на входы кадровых структур, определяются не временными соотношениями задачи, а информационными зависимостями алгоритма, в которых временные зависимости являются только частными случаями.

Под информационной зависимостью двух подграфов G_1 и G_2 графа задачи G понимается наличие пути в информационном графе задачи, соединяющем вершину $a \in G_1$ и вершину $b \in G_2$. Следовательно, два подграфа G_1, G_2 будут информационно независимыми, если в информационном графе задачи не существует ни одного пути, соединяющего любые вершины, принадлежащие G_1 и G_2 соответственно.

Если имеются два информационных графа G_1 и G_2, которые топологически эквивалентны, то в случае их информационной независимости оба эти подграфа могут быть реализованы в виде единого кадра K.

При реализации двух изоморфных подграфов в виде единственного кадра определяющим является правило упорядочивания их, т.е. порядок реализации. В общем случае $K(G_1, G_2) \neq K(G_2, G_1)$.

Правило упорядочивания подграфов в кортеж однозначно определяет порядок поступления операндов и получения результатов. В этой связи очень важным является соответствие входных и выходных дуг изоморфных подграфов. Данное утверждение можно расширить на произвольное количество информационно-независимых изоморфных подграфов. После упорядочивания множества информационно-независимых изоморфных подграфов в кортеж существует единственное правило преобразования кортежа подграфов в кадр.

Дополнительным условием, при котором кортеж подграфов может быть представлен в виде кадра, является наличие рекуррентных правил отображения дуг подграфов g_i:

$$F_R: (x,a_X)_i \rightarrow (x,a_X)_{i+1},\ i=1, 2, \ldots, N-1, \quad (2)$$

$$F_W: (y,a_Y)_i \rightarrow (y,a_Y)_{i+1},\ i=1, 2, \ldots, N-1, \quad (3)$$

где x_i - множество входных вершин *i*-го графа;

a_{Xi} - множество входных дуг *i*-го графа;

y_i - множество выходных вершин *i*-го графа;

a_{Yi} - множество выходных дуг *i*-го графа.

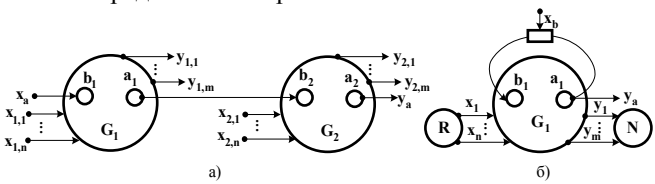
Следует отметить, что изоморфные графы, которые преобразуются в кадр, должны быть реализуемыми, т.е. к моменту инициализации кадра все информационные элементы, соответствующие входным вершинам всех информационных графов, должны быть получены и будут находиться в каналах распределенной памяти в соответствии с функцией чтения.

Полученные выражения можно расширить на наиболее общий случай, когда объединяются информационно зависимые подграфы, но дуги зависимости соединяют непосредственно информационные подграфы без дополнительных операционных вершин.

Если существуют два изоморфных подграфа G_1 и G_2, которые не являются информационно независимыми, при этом существует единственная дуга, начальная вершина которой является $a_1 \in G_1$, а конечной вершиной является $b_2 \in G_2$ и при этом не существует другого пути информационной зависимости между подграфами G_1 и G_2, то подграфы G_1 и G_2 можно упорядочить в кадр, при этом имеется единственное правило упорядочивания $< G_1, G_2 >$, определенное дугой $< a_1, b_2 >$.

В этом случае производится дополнительное преобразование информационного графа. Формируется дуга, соединяющая вершину *a* и вершину *b*. На эту дугу вводится начальный элемент, обеспечивающий реализацию обратной связи. Начальное значение определяется начальной вершиной дуги, входящей в граф G_1.

Вершина b_1 соответствует вершине b_2 (графы G_1 и G_2 изоморфны). Дуги $< x_{b_1}, b_1 >$ и $< a_1, b_2 >$ удаляются из графа кадра. Пример преобразования представлен на рис.2.



а) исходный подграф задачи; б) граф кадра

Рис.2. Пример преобразования связанных информационных графов в кадр.

Если имеется множество изоморфных подграфов $G_1, G_2, \ldots G_n$, являющихся информационно зависимыми только по смежным дугам α_i, соединяющим подграфы G_i и G_{i+1} (начальной вершиной дуг смежности является $a_i \in G_i$, конечной вершиной - $b_i \in G_{i+1}$), то подграфы $G_1, G_2, \ldots G_n$ могут быть преобразованы в кадр.

Если вершина a_1 является соединенной с входной вершиной X_a, то такие подграфы могут быть преобразованы в кадр, причем, правило упорядочивания подграфов определяется путем, связующим подграфы $<G_1, G_2, \ldots G_n>$. Пример такого графа показан на рис.3.

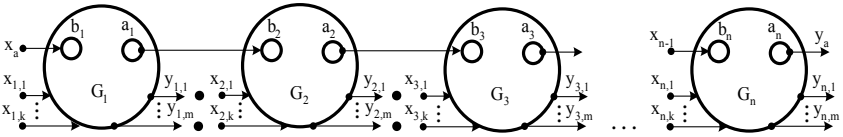


Рис.3. Информационный граф, содержащий изоморфные взаимосвязанные подграфы.

При объединении графа дуга α_i исключается и формируется обратная связь $<a,b>$, инициализируемая начальной вершиной, смежной по дуге α_1 с b_1, т.е. x_a. Структура графа кадра будет такая же, как на рисунке 2б).

Следует отметить, что информационных дуг, соединяющих подграфы, может быть множество (связи могут следовать не только через смежные фрагменты), но все они должны быть описаны функциональной зависимостью между подграфами, согласно выражениям (2) и (3). Количество инициализаций (начальных значений по обратным связям) определяется разностью индексов между индексом начальной вершины $a_i \in G_i$ и конечной вершиной $b_{i+\Delta i} \in G_{i+\Delta i}$.

В более общем случае должна быть выделена функциональная зависимость между начальной вершиной $a_i \in G_i$ и конечной вершиной $b_{f(i)} \in G_{f(i)}$. Причем, связь может быть не только между одиночными подграфами, но и некоторой совокупностью подграфов (слоев) между собой.

Связь между последовательно реализуемыми кадрами может быть только через оперативную память, которая для МВС со структурно-процедурной организацией вычислений должна быть распределенной и многоканальной. Каждой входной дуге a_j кадра должна быть поставлена в соответствие функция чтения r_j, отображающая моменты

времени (такты работы системы) на множество операндов, т.е. обеспечивающая поступление информационных потоков на входы кадра из каналов распределенной памяти. Аналогично, каждой выходной дуге b_j кадра должна быть поставлена в соответствие функция записи w_j, отображающая моменты времени (такты работы системы) на множество результатов, т.е. обеспечивающая сохранение информационных потоков с выходов кадра в каналах распределенной памяти.

Будем называть информационным кадром следующую тройку:

$$K = <R, Q, W>, \tag{4}$$

где R - узел чтения;

Q - структурно-реализованный подграф задачи;

W – узел записи.

Здесь

$$R = \bigsqcup_{j=1}^{N(A)} < a_j, r_j >,$$

где $N(A)$ - мощность множества входных дуг кадра A.

Узел записи представляет собой объединение упорядоченных пар:

$$W = \bigsqcup_{j=1}^{N(B)} < b_j, w_j >, \tag{5}$$

где $N(B_j)$ - мощность множества выходных дуг кадра.

При разбиении информационного графа на подграфы, соответствующие кадрам, необходимо решить ряд математических задач, невыполнение которых может существенно снизить эффективность параллельных вычислений или даже приведет к алгоритмическим конфликтам.

Дополнительным условием разбиения информационного графа задачи на структурно-реализуемые подграфы (фрагменты) является требование, что каждый фрагмент должен быть представлен в виде множества подграфов

$$P_i = \bigsqcup_{j=1}^{N} S_{i,j} \tag{6}$$

таких, что если существуют хотя бы одна дуга $\langle a,b \rangle$ и вершина $a \in S_{ki}$, а вершина $b \in S_{kj}$, то $i \leq j$.

Подграфы S_{ij} будем называть слоями. При этом дуги, соединяющие слои, будем называть межслойными дугами, а дуги, соединяющие операционные вершины одного слоя, будут внутрислойными дугами.

Наконец, каждый подграф $S_{i,j}$ можно представить в виде множества подграфов

$$S_{i,j} = \bigsqcup_{k=1}^{K} g_{i,j,k} \ , \qquad (7)$$

при этом все подграфы $g_{i,j,k}$ должны быть изоморфны друг другу. Подграфы $g_{i,j,k}$ будем называть базовыми подграфами.

Кроме того, выполняются следующие условия.

Внутрислойные дуги должны принадлежать какому-нибудь базовому подграфу. Если существует дуга $\langle a,b\rangle \in S_{i,j}$, то вершина $a \in g_{i,j,k}$ и вершина $b \in g_{i,j,k}$.

Как следствие, не существует дуг, соединяющих вершины, принадлежащие базовым подграфам одного слоя.

Базовый подграф $g_{i,j,k} \in S_{i,j}$ тогда и только тогда, когда существует, по крайней мере, одна дуга $\langle a,b\rangle$ такая, что $a \in q_{ij\text{-}1p}$, $b \in q_{ijk}$.

Подобное представление графа в виде множества подграфов будем называть секвенцией графа. Секвентированный граф вычислительного процесса решения задачи может быть представлен в виде

$$P_i = \Phi_{i,j}(F_{i,j,k}(g_{i,j,k})), \qquad (8)$$

где $F_{i,j,k}$ – некоторая функция, отображающая дуги базового подграфа $g_{i,j,k}$ на базовый подграф $g_{i,j,k+1}$,

$\Phi_{i,j}$ – функция, отображающая множество дуг слоя $S_{i,j}$ на множество дуг слоя $S_{i,j+1}$.

Множество функций F является описанием параллельного (одновременного) выполнения базовых подграфов. Множество функций Φ определяет информационную зависимость между слоями.

Если в результате выполнения секвенции в подграфе P_m подграфы $g_{i,n,k} \in S_{i,n}$ *и* $g_{i,m,k} \in S_{i,m}$, $k=1, 2,\ldots, K$ изоморфны для всех значений n и m, $n=1, 2,\ldots, N$, $m=1, 2,\ldots, N$ и существуют обобщающие функции отображения базовых подграфов внутри слоя и функции отображения слоев друг на друга, то такое разбиение графа вычислительного процесса решения задачи G на подграфы P_i будем называть функциональным секвентированием. В таком случае каждый подграф P_i может быть представлен в виде

$$P_i = \Phi_i(F_i(g_i)). \qquad (9)$$

При функциональном секвентировании подграфы P_i являются функционально-регулярными, а граф вычислительного процесса решения задачи, представленный в этом виде, будет называться функционально-регулярной формой графа вычислительного процесса решения задачи.

Кажущееся ограничение на возможность представления произвольного графа в функционально-регулярной форме не соответствует действительности. Любой граф вычислительного процесса решения задачи может быть представлен, по крайней мере, в тривиальной функционально-регулярной форме, когда информационный подграф содержит единственный слой, а слой содержит единственный базовый подграф. Тогда фрагмент представляется в виде элементарного кадра. Разумеется, для увеличения эффективности параллельных программ целесообразно осуществлять секвенцию графа вычислительного процесса решения задачи в функционально-регулярную форму таким образом, чтобы мощности множества слоев графа вычислительного процесса решения задачи и множества базовых подграфов слоя были как можно большими, что будет доказано ниже.

Функционально-регулярная форма графа позволяет минимизировать аппаратные и программные затраты при решении задачи на МВС и эффективно преобразовать задачу в кадровую форму, обеспечивающую ее эффективную реализацию. Идеальным является случай, когда граф вычислительного процесса решения задачи G может быть представлен в виде единственного разбиения $G=\Phi(F(g))$. В этом случае задача может быть представлена в виде одного кадра.

При преобразовании кортежа изоморфных подграфов в кадр выполняются следующие преобразования.

1)Объединение множеств входных информационных вершин $X = \bigsqcup_{i=1}^{N} \boldsymbol{x}_i$ заменяется функциональным узлом чтения $R = <N_R, F_R>$, в котором первый элемент - вершина функционального узла, а второй – кортеж функций чтения

$$F_R \subset T \times X, \qquad (10)$$

где T - множество моментов времени t_i, $i=1, 2, \ldots, N$, $t_i < t_{i+1}$.

Элемент кортежа F_R - функция чтения r_j ставит в соответствие моменту времени t множество (группу) входных информационных вершин x. В частности, функция чтения может быть определена следующим образом:

$$r_j = \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ . \\ x_{m,1} \end{pmatrix} \& P(t = t_1) \vee \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ . \\ x_{m,2} \end{pmatrix} \& P(t = t_2) \vee \ldots \vee \begin{pmatrix} x_{1,M} \\ x_{2,M} \\ . \\ x_{m,M} \end{pmatrix} \& P(t = t_M) = \bigvee_{i=1}^{M} \begin{pmatrix} x_{1,i} \\ x_{2,i} \\ . \\ x_{m,i} \end{pmatrix} \& P(t = t_i) \,. (11)$$

Здесь $P(t=t_i)$ – предикат, который равен единице, если высказывание $t=t_i$ истинно. Для функции чтения можно использовать следующую компактную форму записи:

$$r_j = \bigvee_{j=1}^{K} \begin{pmatrix} x_{1,j} \\ x_{2,j} \\ . \\ x_{m,j} \end{pmatrix} \cdot t_j \,, \qquad (12)$$

где "·" - некоторый символ, который означает, что моменту времени $t=t_j$ ставится в соответствие множество информационных вершин x_j. В скобках указаны группы вершин, которые должны быть реализованы параллельно.

В дальнейшем функции чтения и записи могут быть записаны более рационально для их программной реализации.

2)◦Объединение множеств выходных информационных вершин $Y = \bigsqcup_{i=1}^{k} \boldsymbol{y}_i$ заменяется функциональным узлом записи $W=(N_W, F_W)$, в котором первый элемент - вершина функционального узла записи, а второй – кортеж функций записи

$$F_W \subset T_K \times Y. \qquad (13)$$

Аналогично функции чтения функция записи может быть записана в виде

$$w_j = \bigvee_{j=1}^{K} \begin{pmatrix} y_{1,j} \\ y_{2,j} \\ \ldots \\ y_{m,j} \end{pmatrix} \cdot t_j \,. \qquad (14)$$

3) Формируется операционная структура кадра, которая описывается графом $Q=(V_Q, A_K)$. В этом графе V_Q - множество вершин, в которое входят вершина узла чтения N_R, вершина узла записи N_W и множество операционных вершин кадра V^{OP}_Q

$$V_Q = N_R \cup V^{OP}{}_Q \cup N_W, \quad (15)$$

Множество дуг кадра A_K является объединением трех непересекающихся подмножеств:

$$A_K = A_R \cup A_Q \cup A_W. \quad (16)$$

Здесь A_R - множество дуг, начальной вершиной которых является вершина N_R, а конечной вершиной - $v \in V^{OP}{}_Q$; A_Q - множество дуг, начальные вершины - $p \in V^{OP}{}_Q$, а конечные вершины - $q \in V^{OP}{}_Q$; A_W - множество дуг с конечной вершиной N_W и начальной вершиной $v \in V^{OP}{}_Q$.

При реализации кадровых структур в аппаратуре МВС функция чтения преобразуется в функцию адресации чтения R_A и функцию коммутации чтения R_K, а функция записи соответственно в функцию адресации чтения W_A и функцию коммутации чтения W_K. Данные функции описывают процедуры параллельного и последовательного доступа к данным, размещенным в памяти МВС.

Для эффективной реализации параллельных вычислений целесообразно, чтобы память МВС состояла из множества сегментов (данные, требующие параллельного обращения, должны быть расположены в разных сегментах памяти). В результате массивы данных являются двумерными, первое измерение определяет номер сегмента (канала) N_K, а второе – адрес ячейки в канале A_C. Для параллельного обращения к группе данных, размещенных в разных сегментах памяти, требуется задать вектор двумерных адресов $W=\{\langle N_{K1},A_{C1}\rangle, \langle N_{K2},A_{C2}\rangle, \ldots,\langle N_{KL},A_{CL}\rangle\}$.

Одной из основных задач синтеза кадровой формы задачи является размещение данных в сегментной памяти. При этом, исходя из кадровой структуры задачи, а также с учетом конфигурации многопроцессорной системы (количества сегментов памяти, топологии коммутационной системы и количества элементарных процессоров), следует выбрать рациональные варианты размещения данных в памяти, которые исключали бы конфликты параллельного обращения.

Чтобы реализовать бесконфликтное параллельное обращение к каналам памяти многопроцессорной системы, необходимо сформулировать систему ограничений, позволяющих выбрать ограниченное количество рациональных вариантов размещения данных в сегментированной памяти для определенного варианта распараллеливания.

Для этого для каждой задачи в соответствии с выбранным вариантом распараллеливания задачи и конфигурацией вычислительной системы (количеством элементарных процессоров и типом

коммутационной системы) формулируется система реляционных отношений, которая определяет семейство вариантов размещения множества данных, используемых в кадрах задачи. На основе этих ограничений может быть выбран рациональный вариант размещения элементов многомерных массивов в распределенной памяти МВС. С помощью определения порядка перечисления элементов многомерных массивов и варианта размещения формируется функциональная структура каждого кадра задачи (функции чтения и записи). На основании функций чтения и сформированной структуры данных могут быть реализованы рекуррентные выражения для функции адресации, а также выражения для функции коммутации в каждом кадре.

Система ограничений включает в себя фундаментальные ограничения, невыполнение которых не позволяет реализовать выбранный вариант распараллеливания в архитектуре вычислительной системы, а также дополнительные ("мягкие") ограничения, невыполнение которых лишь уменьшает эффективность структурно-процедурной реализации задачи.

Фундаментальное ограничение параллельного доступа можно сформулировать следующим образом. Если функция чтения или функция записи ставит в соответствие моменту времени t два операнда d_1 и d_2, то они должны быть расположены в разных каналах распределенной памяти (вершине d_1 функция F_S должна ставить в соответствие адрес $b_1=\langle N_{K1}, A_{C1}\rangle$, вершине d_2 функция F_S должна ставить в соответствие адрес $b_2=\langle N_{K2}, A_{C2}\rangle$, при этом $N_{K1}\neq N_{K2}$).

Фундаментальное ограничение информационной эквивалентности может быть сформулировано следующим образом. Если функция записи w_i кадра K_i ставит в соответствие моменту времени t_1 операнд d_1, а функция чтения r_j кадра K_j ставит в соответствие моменту времени t_2 операнд d_2, при этом операнды d_1 и d_2 являются информационно-эквивалентными, то они должны быть расположены в одной ячейке памяти. В этом случае функция F_S ставит в соответствие информационным вершинам d_1 и d_2 адрес $b_1=\langle N_{K1}, A_{C1}\rangle$.

Одной из основных задач организации эффективных структурно-процедурных вычислений является задача преобразования информационного графа задачи в кадровую форму таким образом, чтобы обеспечить минимальное время решения задачи, что достигается минимизацией числа кадров и максимизацией размеров потока данных, обрабатываемых в кадрах, и минимизацией времени поступления операнда.

Дополнительное ограничение регулярности накладывается на размещение элементов в сегментированной памяти многопроцессорной

системы таким образом, чтобы программные затраты на реализацию функций чтения и записи были минимизированы.

В соответствии с системой фундаментальных и жестких ограничений разработчик может выбрать единственное решение для размещения данных в сегментной памяти.

Полученная структура данных, а также функции чтения и записи позволяют синтезировать функцию параллельного обращения (функцию коммутации) и функцию последовательного обращения (функцию адресации) в кадрах задачи. Функция коммутации представляет собой отображение моментов времени на множество кортежей, элементами которых являются номера каналов распределенной памяти, к которым требуется параллельное обращение. Если множеству моментов времени соответствует один и только один кортеж каналов распределенной памяти, то такая коммутация называется статической и является наиболее предпочтительной с точки зрения аппаратной процедуры параллельного обращения. Между функцией коммутации и функцией адресации существует неразрывная связь, определяемая структурой данных и функцией чтения (записи). При реализации различных вариантов распараллеливания функция адресации может переходить в функцию коммутации и наоборот - функция коммутации может переходить в функцию адресации. Функции коммутации и адресации используются для программирования процедур обращения к данным и после соответствующей адаптации к системе команд МВС могут быть реализованы ее аппаратными средствами.

При работе с двумерными информационными массивами индексы обоих переменных, в общем случае, зависят друг от друга. В то же время для каждого канала распределенной памяти необходимо выполнить фиксированную процедуру обращения к данным, расположенным в канале, независимо от коммутации. Таким образом, необходимо так преобразовать функции адресации, чтобы они зависели только от номера канала и момента времени обращения к данным, расположенным в канале.

Для организации независимых процедур доступа к ячейкам распределенной памяти в канале необходимо осуществить переход от кортежей функций адресации и коммутации к кортежу функций исполнительной адресации чтения FE_R и кортежу функций исполнительной адресации записи FE_W в каналах распределенной памяти.

C-й элемент er_C кортежа FE_R определяется следующим образом:

$$er_c = \bigvee_{k=0}^{R} \left(Dr_{k,c} \right) \cdot t_k = \bigvee_{k=0}^{R} A_c \left(x^J_{k,\alpha(k,c)} \right) \cdot t_k = ar_{\alpha(k,c)}, \qquad (17)$$

где $Dr_{k,c}$ – адрес чтения в канале распределенной памяти, имеющем логический номер c (c =0,1,…,M-1) в момент времени t_k;

$\alpha(k,c)$ – номер элемента вектора коммутации чтения, значение которого в момент времени t_k равно c.

Аналогично, c-й элемент ew_C кортежа функций исполнительной адресации записи FE_W определяется следующим образом:

$$ew_c = \bigvee_{k=0}^{R} \left(Dw_{k,c} \right) \cdot t_k = aw_{\beta(k,c)} \cdot t_k , \qquad (18)$$

где $Dw_{k,c}$ – адрес записи в канале распределенной памяти, имеющем логический номер c, в момент времени t_k;

$\beta(k,c)$ – номер элемента кортежа функций коммутации записи, значение которого в момент времени t_k равно c.

После преобразования информационных подграфов в кадровую форму задача представляется в виде графа кадров G_K. В данном графе каждой вершине ставится в соответствие кадр $K_k = \langle R_k, Q_k, W_{k,} \rangle$.

Если между подграфами задачи P_k и P_m существует информационная зависимость, описываемая множеством дуг $D_{k,m}$, то при преобразовании графа задачи в граф кадров множество дуг $D_{k,m}$ заменяется дугой информационной зависимости $\langle K_k, K_m \rangle$. Граф кадров может быть, в свою очередь, представлен в параллельной форме. В графе кадров функции чтения и функции записи каждого кадра K_k ставится в соответствие область определения T_k – множество моментов времени $t_{i,k}$.

Структурно-процедурная организация вычислений накладывает на реализацию графа кадра следующее ограничение: в многопроцессорной системе одновременно может быть реализован только один кадр, что обеспечивает фон-неймановский детерминизм программы. При такой организации вычислительного процесса легко преобразовать граф кадров в структурно-процедурную форму. В самом деле, кадр можно рассматривать как некоторый функциональный оператор. Тогда можно выделить повторяемые (итерируемые) участки вычислений и реализовать их в виде циклов. Появляются альтернативные ветки алгоритма и т.д. В целом процесс преобразования задачи из графа кадра в структурно-процедурную программу соответствует синтезу последовательной программы для однопроцессорной ЭВМ, и здесь можно воспользоваться всем опытом подобного синтеза.

После преобразования информационного графа в структурно-процедурную форму граф задачи приобретает окончательные изменения: помимо дуг информационной зависимости появляются управляющие дуги, однозначно определяющие порядок вызова кадровых структур, а

также вершины-решатели [17], определяющие, какая из альтернативных ветвей алгоритма должна быть выполнена. При этом в графе структурно-процедурной формы задачи кадры формируются в особые группы, называемые уровнями.

Для того чтобы кадры *a* и *b* находились в одном уровне, необходимо выполнение двух условий. Во-первых, кадры *a* и *b* должны быть информационно независимы, т.е. не должен существовать путь по дугам информационной зависимости, соединяющий вершины *a* и *b*. Во-вторых, путь по дугам управления, соединяющий кадры *a* и *b*, не должен проходить через вершину-решатель. Таким образом, кадры будут находиться в одном уровне тогда и только тогда, когда они могут быть выполнены одновременно.

Компоненты кадров задачи естественным образом отображаются в структуру МВС ПА. На рис.4 представлена схема отображения кадровой структуры задачи на архитектуру МВС со структурно-процедурной организацией вычислений.

Так, внутренние (операционные) дуги базового подграфа отображаются на множество коммутационных связей, реализуемых в коммутаторе макропроцессора К, множество операционных вершин базового фрагмента – во множество элементарных процессоров макропроцессора (ЭП). Множество входных и выходных дуг базового фрагмента реализуется во множестве коммутационных связей коммутаторов K_1 и K_2.

Массивы данных отображаются на множество адресов распределенной памяти. Управляющая программа смены кадров и процедуры обращения к каналам распределенной памяти осуществляется в контроллере распределенной памяти (КРП).

Таким образом, программист, синтезируя кадры, тем самым создает ряд виртуальных специализированных вычислителей, реализуемых средствами программирования архитектуры многопроцессорной вычислительной системы.

Перенумерация индексов входных (выходных) переменных определяет элементы многомерных массивов, которые являются входными данными (результатами) вычислительного фрагмента. Циклически повторяющиеся вычислительные фрагменты (базовые фрагменты) над элементами многомерных массивов могут быть непосредственно преобразованы в кадры, отображаемые в структуру многопроцессорной системы. Каждый базовый подграф структурно реализуется на множестве элементарных процессоров, соединенных пространственной коммутационной системой. При этом степень структурного распараллеливания задачи определяется количеством

базовых структурных фрагментов, которые могут быть одновременно реализованы в МВС.

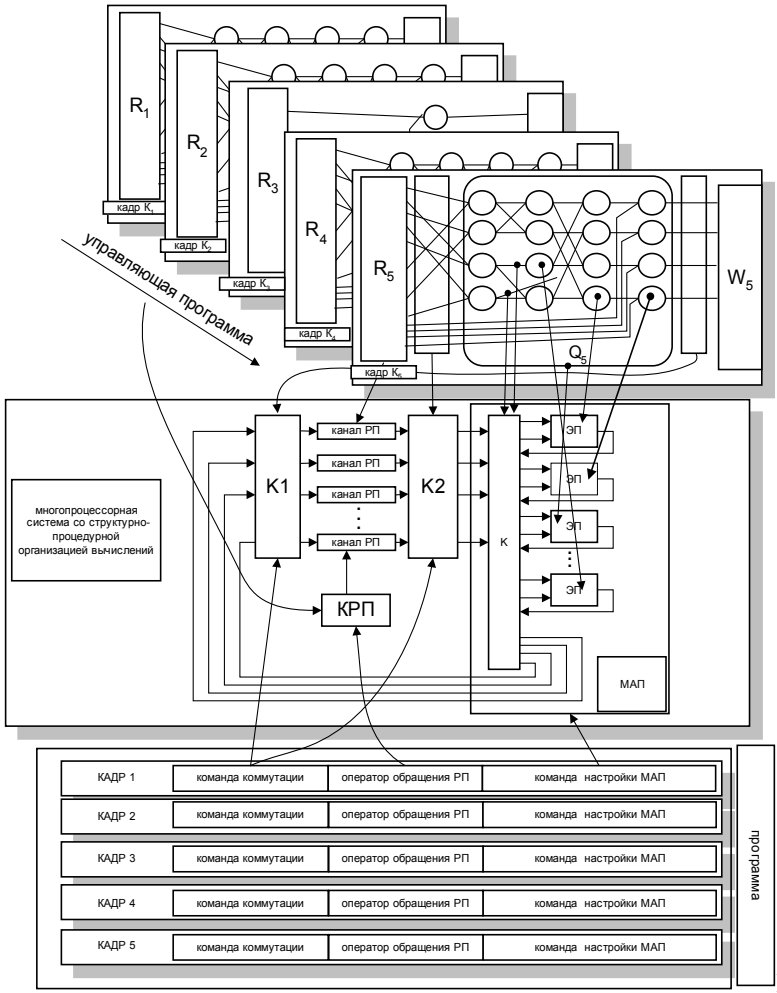


Рис.4. Отображение кадровой структуры задачи на архитектуру многопроцессорной системы.

1. Головкин Б.А. Параллельные вычислительные системы. - М.: Наука, 1980. - 519 с.

2. Мизин И.А., Махиборода А.В. Архитектура самоопределяемых данных в среде взаимодействия открытых систем. // Материалы II-ой

Международной Конференции «Развитие и применение открытых систем», Петрозаводск, 1995.

3. Митропольский Ю. суперкомпьютеры и микропроцессоры. // Электроника: наука, технология, бизнес, 2000. -№2. - С.18-21.

4. Аладышев О.С., Дикарев Н.И., Овсянников А.П., Телегин П.Н., Шабанов Б.М. СуперЭВМ: области применения и требования к производительности. // Известия ВУЗов. Электроника, 2004. - №1. - С. 13-17.

5. Бабаян Б.А., Бочаров А.В., Волин А.С. и др. Многопроцессорные ЭВМ и методы их проектирования. / Под ред. Смирнова Ю.М. - М.: Высшая школа, 1990.

6. Корнеев В.В. Параллельные вычислительные системы. - М.: Нолидж, 1999.

7. Воеводин В.В. Математические модели и методы параллельных процессов. - М.: Наука, 1986. - 286 с.

8. Воеводин В.В. Информационная структура алгоритмов. - М.: Изд-во МГУ, 1997.

9. Воеводин В.В. Отображение проблем вычислительной математики на архитектуру вычислительных систем. // Вычислительная математика и математическое моделирование. Тр. международной конф. - М.: Ин-т вычисл. математики РАН, 2000. - Т.1. - С. 242-255.

10. Воеводин В.В. Параллельные структуры алгоритмов и программ. - М.: ОВМ АН СССР, 1987.

11. Воеводин В.В. Математические основы параллельных вычислений. - М.: МГУ, 1991. - 345 с.

12. Siegel L.J., Siegel H.J., Swain P.H. Performance Measures for Evaluation Algorithms for SIMD-Machines. // IEEE Trans. Software Eng. - V. 8. -№ 4. - Pp. 319-331.

13. Каляев В.А., Левин И.И., Фомин С.Ю. Об оценке эффективности решения задач математической физики на многопроцессорных системах. // Электронное моделирование, 1989. - №6. - С. 11-15.

14. Каляев А.В., Левин И.И. Многопроцессорные системы с перестраиваемой архитектурой: концепции развития и применения. // Наука – производству, 1999. - № 11. - С.11-19.

15. Программирование на параллельных вычислительных системах. / Под ред. Р. Бэба II. - М.: Мир, 1991. - 376 с.

16. Андрианов А.Н., Ефимкин К.Н., Задыхайло И.Б. Непроцедурный язык для решения задач математической физики. // Программирование. 1991. - №2. - С. 80-94.

17. Поспелов Д.А. Введение в теорию вычислительных систем. - М.: Сов. радио, 1972. – 280 с.