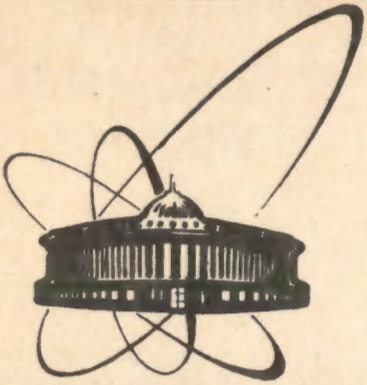


объединенный
институт
ядерных
исследований
дубна

P11-91-426

Л.Грегушова

ИССЛЕДОВАНИЕ МЕЖСЕГМЕНТНЫХ СВЯЗЕЙ ПРИ РАСПАРАЛЛЕЛИВАНИИ БОЛЬШИХ ПРОГРАММ

Направлено в журнал "Программирование"

1991

1. Введение

Интенсификация внедрения вычислительной техники как в область управления технологическими процессами, так и в область исследования физических объектов приводит к необходимости создания высокопроизводительных проблемно-ориентированных вычислительных систем (ВС), обеспечивающих высокий темп обработки информации.

В данном случае возникает необходимость в эффективной реализации многократно применяемых алгоритмов к периодически изменяемым на входе ВС потокам данных. Этому требованию отвечают мультиконвейерные вычислительные системы (МКВС), макроструктура которых изображена на рис. 1. Для обеспечения высокого темпа обработки информации конвейер мультипроцессоров осуществляет конвейерную обработку параллельных ветвей программы [1].

В результате появляется возможность параллельно-конвейерной реализации больших программ за пределами ресурсов одного вычислителя. Однако время вычисления программных сегментов Φ_i может быть недопустимо большим, что не позволяет увеличивать темп обработки данных.

Расширение класса решаемых задач и увеличение темпа их обработки возможно при детальном анализе и эквивалентном преобразовании их структуры с учетом циклических участков исходных программ. Так как в настоящее время не существует единого подхода, позволяющего осуществлять параллельно-последовательную конвейеризацию заданного класса программ, необходимо рассмотреть следующую проблему:

определить и формализовать типы связей между программными сегментами, назначенными к вычислению на МКВС с распределенной памятью с целью уточнения условий распараллеливания вычислений.

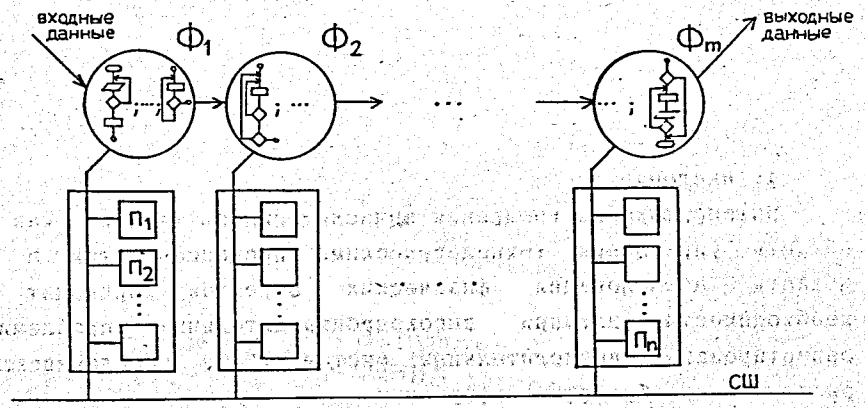


Рис.1. Макроструктура МКВС, ориентированная на реализацию фаз $(\Phi_i)_{i=1}^{m}$, СШ - системная шина, Π_n - n- й процессор.

2. Основные понятия

Под программным сегментом Φ_i будем понимать некоторую выделенную последовательность операторов O_{i_k} входного языка (напр. FORTRAN). Иллюстрацией этого понятия является рис.2. Для каждого сегмента определены наборы $In(\Phi_i)$ и $Out(\Phi_i)$ входных и выходных данных соответственно и $|\Phi_i| = n_i$, где n_i - число операторов O_{i_k} сегмента Φ_i; $k = \overline{1,n_i}$.

Пусть задана последовательная программа $\mathcal{P}$ и N есть число выделенных в ней программных сегментов Φ_i, где $i \in \overline{N} = (1,2,\ldots,N)$. Под вычислением сегмента Φ_i будем понимать реализацию последовательности его операторов.

Если Φ_j должен вычисляться сразу после Φ_i, то этот факт обозначим $\Phi_i \lhd \Phi_j$, где $i,j \in \overline{N}$, $i \neq j$. Вычислительным процессом назовем упорядоченную последовательность вычислений программных сегментов, начинающуюся первым сегментом. Эта последовательность при разных реализациях данной программы может отличаться, так как для некоторых Φ_i могут существовать такие Φ_j, Φ_k, что действительно $\Phi_i \lhd \Phi_j$ или $\Phi_i \lhd \Phi_k$ и $i,j,k \in \overline{N}$.

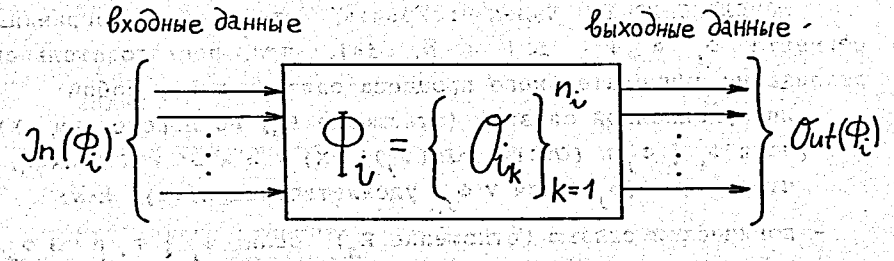


Рис.2. Понятие программного сегмента.

Если вычислительный процесс, при котором вычисляется сегмент Φ_j, проходит или может пройти через сегмент Φ_i, то это будем обозначать $\Phi_i \lhd \Phi_j$. Это значит, что существует цепочка вычислений сегментов $\Phi_{j_1}, \Phi_{j_2}, \ldots, \Phi_{j_n}$; $j_k \in \overline{N}$; что

$$\Phi_i \lhd \Phi_{j_1} \lhd \Phi_{j_2} \lhd \ldots \lhd \Phi_{j_n} \lhd \Phi_j. \qquad (1)$$

Заметим, что если $\Phi_i \lhd \Phi_j$, то $\Phi_i \lhd \Phi_j$ и подавно. Обратное не верно. Обозначим

$$[\Phi_j] = \{ \Phi_i \mid \Phi_i \lhd \Phi_j,\ i \in \overline{N} \}. \qquad (2)$$

Это множество будем называть замыканием сегмента Φ_j.
Если $\Phi_i \in [\Phi_i]$, то Φ_i является циклом.
Если $\Phi_i \in [\Phi_j] \wedge \Phi_j \in [\Phi_i]$, $\Phi_i \neq \Phi_j$; то Φ_i, Φ_j являются частями программного цикла и $[\Phi_i] = [\Phi_j]$.

Достаточным условием параллельного выполнения двух сегментов Φ_i, Φ_j над общей памятью является так называемое условие Бернштейна - Рассела - Нариньяни [2,3]:

$$(Out(\Phi_i) \cap In(\Phi_j)) \cup (In(\Phi_i) \cap Out(\Phi_j)) \cup (Out(\Phi_i) \cap Out(\Phi_j)) = \varnothing. \qquad (3)$$

Для реализации параллельного вычисления сегментов на МКВС с распределенной памятью необходимо ввести понятие слабой связи. Это в дальнейшем позволит нам расширить класс задач, параллельно решаемых на МКВС. Дадим сначала классическое определение связей двух сегментов.

ОПРЕДЕЛЕНИЕ 1. Будем говорить, что два программных сегмента Φ_i и Φ_j, $i,j \in \overline{N}$, $i \neq j$, при последовательной реализации вычислительного процесса связаны между собой:

- информационной связью (отношение $R_И$) по переменным (X), если $\Phi_i \{ \Phi_j$ и $(Out(\Phi_i) \cap In(\Phi_j)) = (X) \neq \varnothing$ и $\exists\ x \in X$ такое, что $x \notin Out(\Phi_{j_k})$ для $\forall\ \Phi_{j_k}$, удовлетворяющих (1), $k \in \overline{N}$;
- логической связью (отношение $R_Л$), если $\Phi_i \{ \Phi_j$ и $\exists\ \Phi_k$, $k \in \overline{N}$, $\Phi_k \neq \Phi_j$ такое, что $\Phi_i \{ \Phi_k$ и $\Phi_k \notin [\Phi_j]$.

Выполнение условия (3) говорит об отсутствии информационной и так называемой конкуренционной зависимости двух сегментов при вычислении над общей памятью. Заметим, что если Φ_i, Φ_j не зависят конкуренционно, то $\Phi_i\ \overline{R}_И\ \Phi_j$, но обратное не верно.

Обозначение $\overline{R}_И$ указывает на отсутствие бинарного отношения $R_И$. С учетом особенностей организации МКВС (в частности, распределенной памяти), целесообразно дополнительно ввести в рассмотрение следующие виды связей.

ОПРЕДЕЛЕНИЕ 2. Будем говорить, что два программных сегмента Φ_i и Φ_j, $i,j \in \overline{N}$, $i \neq j$, при последовательной реализации вычислительного процесса связаны между собой:

- непосредственной связью (отношение $R_Н$), если $\Phi_i \{ \Phi_j$;
- конкуренционной связью (отношение $R_К$), по переменной x, если $\Phi_i \{ \Phi_j$ и $x \in (In(\Phi_i) \cap Out(\Phi_j)) \neq \varnothing$;
- связью (отношение $R_Д$), по идентичности переменной x, если $\Phi_i \{ \Phi_j$ и $x \in (Out(\Phi_i) \cap Out(\Phi_j)) \neq \varnothing$.

Заметим, что транзитивное замыкание $R_Н^+$ отношения $R_Н$ совпадает с отношением $\{$. Для нас важным является отношение $(\overline{R_Л})^+$, которое представляет транзитивное замыкание $\overline{R_Л}$. Видно, что $(\overline{R_Л})^+ \subset R_Н^+$ состоит только из линейных цепочек сегментов Φ_k ($k \in \overline{N}$). Обозначим $(\overline{R_Л})^+$ как R_L. На основе информационной связи можно определить следующие, для выяснения возможности параллельной обработки на МКВС, важные отношения.

ОПРЕДЕЛЕНИЕ 3. Два программных сегмента Φ_i и Φ_j, $\Phi_i \{ \Phi_j$ при последовательной реализации вычислительного процесса связаны между собой:

- связью частичной информационной зависимости (отношение $R_З$), если $\bigcup_{k=0}^{n} Out(\Phi_{j_k}) \cap In(\Phi_j) \neq \varnothing$, причем начальный индекс $j_0 = i$;
- слабой связью (отношение $R_С$), если $\bigcup_{k=0}^{n} Out(\Phi_{j_k}) \cap In(\Phi_j) = \varnothing$ и $j = i$.

Необходимо отметить, что $R_С \equiv \overline{R_З}$. Интересно то, что $R_С^+ \subset \overline{R_З^+}$.

3. Условия параллельной обработки

В отличие от классического определения 1, введенные выше связи определены с учетом распределенной обработки информации на МКВС. На основе этих отношений получаем:

ЛЕММА 1. Достаточным условием существования в цепочке $\Phi_i = \Phi_{j_0} \{ \dots \{ \Phi_{j_{n+1}} = \Phi_j$, $\Phi_i\ R_L\ \Phi_j$ сегментов, находящихся в отношении $R_С$, является условие $\overline{R_З^+}$.

Доказательство. Оно вытекает из свойств $R_И$, $R_З$, $R_С$. Заметим, что $R_И \subseteq R_З$. В данном случае $R_З$ означает, что два сегмента частично информационно зависимы. В свою очередь, $R_З^+$ означает, что существует цепочка частично зависимых сегментов, соединяющая Φ_i, Φ_j. Тогда $\overline{R_З^+}$ (дополнение транзитивного замыкания частичной информационной зависимости) означает, что в данной цепочке частично зависимых сегментов связь прервана, т.е. существуют два сегмента Φ_{j_k}, $\Phi_{j_{k+1}}$ ($k = 1,2,\dots,n$), которые не связаны частичной информационной зависимостью. Следовательно, $\Phi_{j_k}\ R_С\ \Phi_{j_{k+1}}$, т.е. что и требовалось доказать.

Приведем простой пример задания программных сегментов, не позволяющих для общей памяти вести параллельную обработку. Пусть Φ_1, Φ_2, Φ_3 на рис. 3а являются частью вычислительного процесса. Связи между ними даны в виде графа на рис. 3б. $\Phi_1\ R_С\ \Phi_2$ позволяет на МКВС вычислять эти два сегмента параллельно (предполагаем, что подпрограмма SUM1 не меняет переменные X,Y, а то получаем $\Phi_1\ R_И\ \Phi_2$).

В дополнение к известным из литературы необходимым условиям [2,3] параллельной обработки представляет интерес для МКВС следующее утверждение.

УТВЕРЖДЕНИЕ. Достаточным условием параллельной обработки на МКВС двух сегментов Φ_i, Φ_j, $\Phi_i\ R_L\ \Phi_j$ является выполнение требования

$$\Phi_i\ R_C\ \Phi_j. \qquad (4)$$

Доказательство. Пусть $\Phi_i\ R_C\ \Phi_j$, то по определению R_C для $\forall\ \Phi_{j_k}$ из (1) $\bigcup_{k=0}^{n} Out(\Phi_{j_k}) \cap In(\Phi_j) = \varnothing$, т.е. $\Phi_{j_k}\ \overline{R_И}\ \Phi_j$. Из $\Phi_i\ R_L\ \Phi_j$ вытекает, что Φ_j не зависит логически ни от одного Φ_{j_k} в (1). Покажем, что для параллельного выполнения Φ_i, Φ_j, не препятствуют отношения $R_К$, $R_Д$, возникающие между сегментами Φ_{j_k} (для $k = 0$ будет $j_k = i$) цепочки (1) и сегментом Φ_j. В самом деле:

1. Из отношения $R_К$ следует, что если $In(\Phi_{j_k}) \cap Out(\Phi_j) = x$ и для Φ_{j_k} готова вся входная информация, то он может начать выполняться с Φ_{j_k} параллельно без того, чтобы x передавалось сегменту Φ_{j_k}. Следовательно, для $j_0 = 0$ можно Φ_i, Φ_j выполнять параллельно.

2. Из отношения $R_Д$ следует, что $Out(\Phi_{j_k}) \cap Out(\Phi_j) = x$ и $x \notin In(\Phi_j)$. Так как Φ_j меняет x, выработанное Φ_{j_k}, может быть или а) $\Phi_{j_k}\ R_Л\ \Phi_j$, или б) $x \in In(\Phi_{j_l})$, $l = k+1, \ldots, n$. Случай (а) находится в противоречии с требованием $\Phi_i\ R_L\ \Phi_j$. Случай (б) для отношения $R_К$ уже рассмотрен. Следовательно, Φ_i и Φ_j можно выполнять параллельно. Доказательство закончено.

СЛЕДСТВИЕ. Достаточным условием параллельной обработки двух сегментов $\Phi_i\ R_L\ \Phi_j$ на МКВС является требование, чтобы для $\forall\ \Phi_{j_k}$, $k=0,1,\ldots,n$, цепочки $\Phi_i = \Phi_{j_0} \{ \Phi_{j_1} \{ \ldots \{ \Phi_{j_n} \{ \Phi_j$ выполнялось

$$\Phi_{j_k}\ \overline{R_И}\ \Phi_j. \qquad (5)$$

Доказательство. Требование (5) эквивалентно требованию (4) по определению 3.

Рис. 4 иллюстрирует три возможные ситуации появления

переменной x в сегменте Φ_i, из-за которой $\Phi_i\ R_Д\ \Phi_j \wedge \Phi_i\ R_К\ \Phi_j$ и $\Phi_i \{ \Phi_j$, т.е. $x \in (Out(\Phi_i) \cap Out(\Phi_j))$, но $x \notin In(\Phi_j)$ (иначе $\Phi_i\ R_И\ \Phi_j$). Тогда в ситуациях (а) и (б) информация x в дальнейшем не нужна, если $\Phi_i \{ \Phi_j$, т.к. $x \in Out(\Phi_j)$.

В ситуации (в) сегмент Φ_i не будет зависеть информационно по x от любого Φ_k, $\Phi_k \{ \Phi_i$ и имеет смысл или убрать x из списка $In(\Phi_i)$, или его переименовать.

В ситуации (а) и (б) при $\Phi_i \{ \Phi_j$ и $\Phi_i\ \overline{R_И}\ \Phi_j$ информация x может потребоваться сегментам $\Phi_{j_k} \in ([\Phi_j] \setminus [\Phi_i])$, $j_k \in \overline{N}$ цепочки (1). Тогда Φ_i, Φ_j можно вычислять параллельно, если x, выработанное в Φ_j, достанется каждому Φ_i такому, что $\Phi_j \in [\Phi_i]$.

Если $\Phi_j \{ \Phi_i$, то условия параллельной обработки усложняются. Отношение R_C не является симметричным и, если существует путь $\Phi_j \{ \Phi_i$, то может оказаться $\Phi_j\ \overline{R_C}\ \Phi_i$. Отсюда можно сделать вывод, что отсутствие частичной зависимости далеко не всегда выполняется в циклах, так как здесь отношение конкуренционной связи $\Phi_i\ R_К\ \Phi_j$ может измениться на отношение информационной зависимости $\Phi_j\ R_И\ \Phi_i$. Такие случаи возникают при наличии в циклах простых выходных переменных (не имеющих индексы).

Отметим, почему определение R_C основано на требовании отсутствия логической связи для параллельной обработки сегментов $\Phi_i \{ \Phi_j$: если $\Phi_j \in [\Phi_i]$, тогда конкуренционная связь может превратиться в информационную. Это возможно, когда мы имеем дело с вложенными циклами. Поэтому, если $\Phi_j \notin [\Phi_i]$, то условие $\Phi_i\ R_L\ \Phi_j$ можно не принимать во внимание и ограничиться рассмотрением отношений $\Phi_i\ R_C\ \Phi_j$ для всех цепочек вида (1), для которых $\Phi_i \{ \Phi_j$. В работе [3] дан практический метод определения случаев, когда конкуренционная связь во вложенных циклах с индексными переменными превращается в информационную зависимость.

4. Заключение

Введенное понятие замыкания $[\Phi_j]$ сегмента Φ_j удобно для указания всех сегментов, которые могут повлиять на Φ_j. При конвейеризации вычислительного процесса и определении места вычисления сегмента Φ_j в мультиконвейере необходимо обеспе-

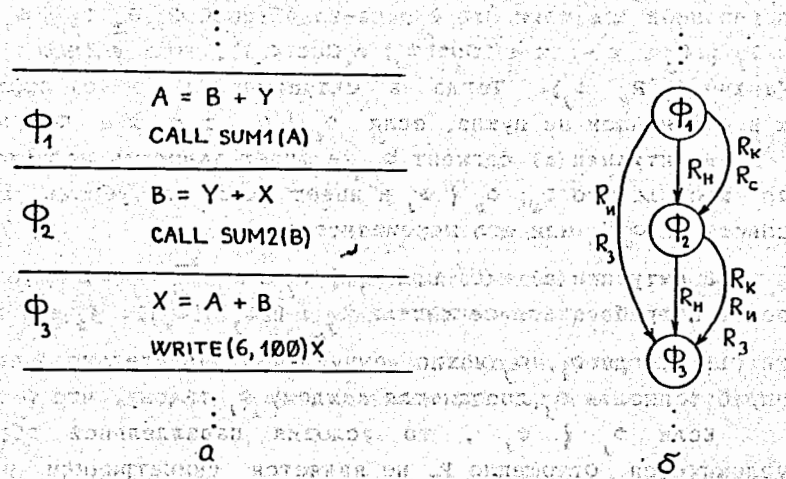


Рис. 3. Пример задания программных сегментов, не позволяющих вести параллельную обработку над общей памятью.

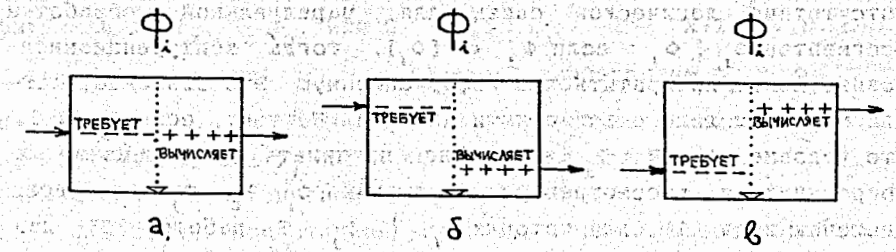
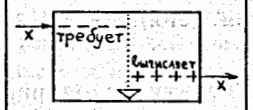
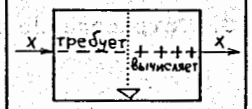
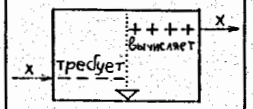
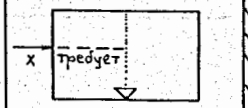
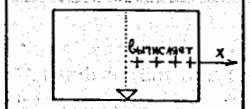


Рис. 4. Места использования и вычисления переменной x сегмента Φ_i. Пунктиром обозначена последовательность операторов сегмента Φ_i (упорядоченная в направлении ↓), штриховой линией обозначены места использования x ($x \in In(\Phi_i)$) и знаками + места вычисления переменной x ($x \in Out(\Phi_i)$).

Таблица 1

Типизация связей сегмента Φ_i с сегментом Φ_j: $\Phi_i \{ \Phi_j$ в зависимости от места использования или вычисления переменной x

случай	Вид сегмента Φ_j / Вид сегмента Φ_i	а)	б)	в)	г)	д)
а		$R_и$	$R_и$		$R_и$	
б		$R_и$	$R_и$		$R_и$	
в		$R_и$	$R_и$		$R_и$	
г				$R_с$		
д		$R_и$	$R_и$		$R_и$	

чить корректное расположение всех сегментов множества [Φ_j] по слоям конвейера. При этом порядковые номера таких слоев не должны быть больше номера слоя, в котором располагается Φ_j.

Возможные области существования слабой связи сегментов в соответствии с введенными понятиями из рис. 4 сведены в табл. 1. Как видно из таблицы, слабая связь между сегментами является распространенной и может появляться практически во всех случаях задания сегментов. Получение этого результата открывает возможность проблемной ориентации МКВС на достаточно широкие классы задач.

Литература

1. Дзегеленок И.И. Мультиконвейерные вычислительные системы на базе микро-ЭВМ.-М.:Изд. МЭИ, 1985.

2. Грегушова Л. Разработка и исследование методов синтеза мультиконвейерных вычислительных структур. Автореферат канд. дисс., Моск. энергет. инст., Москва, 1987.

3. Грегушова Л. Методика распараллеливания циклов для мультиконвейерных вычислительных систем. ОИЯИ, Р11-87-900, Дубна, 1987.

4. Грегушова Л. Синтез структуры мультиконвейерной вычислительной системы. – В сб. работ государственного семинара: "Параллельные системы", апрель 1990, Татранска Ломница, с.34-36. (на словацком языке).

Рукопись поступила в издательский отдел
23 сентября 1991 года