УДК 519.68

Л.С. КИРКОРОВА, С.И. КИРКОРОВ

Научно-производственное предприятие «МедиаСкан», Республика Беларусь

УЛУЧШЕНИЕ ЛОКАЛЬНОСТИ КОМПОЗИЦИИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ЯЗЫКА ADA

В данной статье рассмотрена композиция параллельных алгоритмов и улучшение локальности алгоритмов: каждое гнездо циклов (каждый алгоритм) отображается на параллельный компьютер сам по себе. Исследуется локальность данных, определяемых в одном, а используемых в другом гнезде циклов. Выводятся условия, позволяющие выбирать в этих гнёздах циклы, приводящие к согласованию входных/выходных данных при композиции составляющих сложного алгоритма. Рассматриваются два иллюстрационных примера – алгоритм перемножения трёх матриц и алгоритм метода матричной прогонки.

Ключевые слова: *язык Ada, параллельный алгоритм, гнёзда цикла, тайлинг.*

Введение

Есть множество задач, которые по различным причинам должны быть решены быстро. Поэтому во многом делается упор на компьютеры с несколькими ядрами, в частности – суперкомпьютеры. Параллельные вычисления (параллельная обработка) – это использование нескольких или многих вычислительных устройств для одновременного выполнения разных частей одной программы (одного проекта) [1, 2]. Основная цель параллельных вычислений – это уменьшение времени решения задачи. Единственным универсальным языком программирования имеющем международный стандарт и поддерживающим параллельные алгоритмы строго стандартными средствами является Ada. Выбор языка Ada – подход для приложений критичных к безопасности и выполнения требований информационной безопасности [3, 4].

В данной работе рассмотрена композиция параллельных алгоритмов и улучшение локальности алгоритмов: каждое гнездо циклов (каждый алгоритм) отображается на параллельный компьютер сам по себе.

1. Математический аппарат параллельных вычислений

Формальный аппарат, используемый в методах статического распараллеливания программ, описан в [2,5,6].

Пусть в гнезде циклов имеется K операторов и используется L массивов данных. Простые переменные будем считать массивами размерности 0. Область изменения параметров гнезда циклов для оператора S_β будем называть **областью итераций** и обозначать V_β. Обозначим $N \in Z^e$ – вектор внешних переменных, где e – число внешних переменных.

Вхождением (l,β,q) будем называть q-ое вхождение массива a_l в оператор S_β. Для таких вхождений также будем использовать обозначение (a_l, S_β, q). Через Q обозначим множество вхождений (l,β,q). Индексы элементов l-го массива данных, встречающегося в операторе S_β и относящегося к q-тому вхождению элементов этого массива в оператор, будем выражать аффинной функцией $\bar{F}_{l,\beta,q} : V_\beta \to W_l$ вида

$$\bar{F}_{l,\beta,q}(J) = F_{l,\beta,q}J + G_{l,\beta,q}N + f^{(l,\beta,q)},\ J \in V_\beta, \quad (1.1.1)$$

где $F_{l,\beta,q} \in Z^{v_l \times n_\beta}$, v_l – размерность q-го массива данных, n_β – число циклов, окружающих оператор S^β, $N \in Z^e$, $G_{l,\beta,q} \in Z^{v_l \times e}$, $f^{(l,\beta,q)} \in Z^{v_l}$, $(l,\beta,q) \in Q$. Если l-й массив имеет размерность 0 (т. е. является простой переменной), то будем считать $\bar{F}_{l,\beta,q}(J)$ константой. Область изменения индексов -го массива будем обозначать W_l ($W_l \in Z^{v_l}$).

Реализация (срабатывание, выполнение) оператора S_β при конкретных значениях вектора параметров цикла J называется **операцией** и обозначается $S_\beta(J)$. Выполнение всех операций, зависящих от J, называется J-Й (итерацией J).

Введём определение **зависимости между операциями**. Операция $S_\beta(J)$, $J \in V_\beta$ зависит от операции $S_\alpha(I)$, $I \in V_\alpha$ (обозначение: $S_\alpha(I) \to S_\beta(J)$), если:

1) $S_\alpha(I)$ выполняется раньше $S_\beta(J)$;

2) $S_\alpha(I)$ и используют один и тот же элемент какого-либо массива $S_\beta(J)$, (т.е. $\bar{F}_{l,\alpha,p}(I) = \bar{F}_{l,\beta,q}(J)$ для некоторых (l,p,q), и, по крайней, мере, одно из использований есть переопределение (изменение) элемента;

3) между операциями $S_\alpha(I)$ и $S_\beta(J)$ этот элемент не переопределяется.

Наличие зависимости $S_\alpha(I) \to S_\beta(J)$ означает, что нельзя переупорядочить операции гнезда циклов таким образом, чтобы $S_\beta(J)$ выполнялась раньше $S_\alpha(I)$.

Введём определение лексикографического порядка. Пусть имеются d_1-мерный вектор $(x_1,\ldots,x_{d_1})$ и d_2-мерный вектор $(y_1\ldots,y_{d_2})$; обозначим $d = \min(d_1,d_2)$. Вектор y **лексикографически больше** вектора x (обозначение: $y > \text{lex}^x$), если первая ненулевая координата вектора $(y_1 - x_1,\ldots,y_d - x_d)$ положительна. Вектор y **лексикографически равен** вектору x ($y = \text{lex}^x$), если $(y_1 - x_1,\ldots,y_d - x_d)$=0. Вектор y **лексикографически больше или равен** вектору x ($y \ge \text{lex}^x$), если $y > \text{lex}^x$ или $y = \text{lex}^x$.

Пусть имеется гнездо вложенных циклов. Зависимость между операциями $S_\alpha(I)$ и $S_\beta(J)$ называется **внутриитерационной** (не вызванной циклом), если $I = \text{lex}^J$; если же $I < \text{lex}^J$, то имеет место **зависимость по циклу.** Таким образом, в случае внутриитерационной зависимости $S_\alpha(I) \to S_\beta(J)$ значения параметров всех общих для операторов $S_\alpha(I)$ и $S_\beta(J)$ циклов не отличаются; в случае зависимости по циклу значения параметра фиксированного цикла отличаются, а значения параметров всех более внешних циклов (если они имеются) не отличаются.

Развёрнутым графом зависимостей алгоритма называется ориентированный граф, полученный следующим образом: каждая операция $S_\beta(J)$ порождает вершину $\vartheta_\beta(J)$ графа, каждая зависимость $S_\alpha(I) \to S_\beta(J)$ порождает дугу

$$\left(\vartheta_\alpha(I), \vartheta_\beta(J)\right).$$

Для каждой пары $(\alpha,\beta) \in P$ обозначим через $V_{\alpha,\beta}$ подмножество таких $J \in V_\beta$, что для какого-либо I существует дуга $(\vartheta_\alpha(I), \vartheta_\beta(J))$ в развернутом графе зависимостей:

$$V_{\alpha,\beta} = \{J \in V_\beta \mid \exists S_\alpha(I) \to S_\beta(J)\}. \qquad (1.1.3)$$

Функцию $\bar{\Phi}_{\alpha,\beta} : V_{\alpha,\beta} \to V_\alpha$ такую, что если $S_\alpha(I) \to S_\beta(J)$, $I \in V_\alpha$, $J \in V_{\alpha,\beta} \in V_\beta$, mo $I = \bar{\Phi}_{\alpha,\beta}(J)$ будем называть **функцией зависимостей**.

Функции зависимостей описывают зависимости алгоритма: зная $\bar{\Phi}_{\alpha,\beta}$ и $V_{\alpha,\beta}$ можно для любой операции $S_\beta(J)$ найти все операции $S_\alpha(I)$, от которых $S_\beta(J)$ зависит.

В дальнейшем предполагается, что функции зависимостей являются аффинными:

$$\bar{\Phi}_{\alpha,\beta}(J) = \Phi_{\alpha,\beta}J + \Psi_{\alpha,\beta}N - \varphi^{(\alpha,\beta)}, \qquad (1.1.4)$$

$J \in V_{\alpha,\beta}$, $N \in Z^e$, $\Phi_{\alpha,\beta} \in Z^{n_\alpha \times n_\beta}$, $\Phi_{\alpha,\beta} \in Z^{n_\alpha \times e}$, $\varphi^{(\alpha,\beta)} \in Z^{n_\alpha}$, $(\alpha,\beta) \in P$

Пусть $(\alpha,\beta) \in P$, I, J – векторы одинаковой размерности такие, что существует зависимость $S_\alpha(I) \to S_\beta(J)$. Вектор $d^{(\alpha,\beta)} = J - I$ называется **вектором зависимости**.

Пусть существует зависимость $S_\alpha(I) \to S_\beta(J)$. Номер l первой ненулевой компоненты вектора зависимостей $d^{(\alpha,\beta)} = J - I$ называется **уровнем зависимости**. Если $J - I = 0$, то принимается $l = \infty$.

В случае гнезда вложенных циклов уровень зависимости указывает цикл, по которому имеется зависимость. Для внутриитерационной зависимости уровень зависимости принимается бесконечно большим.

Редуцированным графом зависимостей алгоритма называется ориентированный граф, полученный следующим образом: каждому оператору S_β ставится в соответствие вершина ϑ_β, существует дуга $(\vartheta_\alpha, \vartheta_\beta)$, если для каких-либо I, J существует зависимость $S_\alpha(I) \to S_\beta(J)$ (т. е. существует дуга $(\vartheta_\alpha(I), \vartheta_\beta(J))$ в развёрнутом графе зависимостей).

Редуцированный граф зависимостей даёт сжатое представление развёрнутого графа зависимостей.

Различают зависимости нескольких типов.

Если функция $\bar{F}_{l,\alpha,p}(I)$ встречается в левой части оператора S_α, а $\bar{F}_{l,\beta,q}(J)$, встречается в пра-

вой части оператора S_β, то зависимость является **истинной**. Если $\overline{F}_{l,\alpha,p}(I)$ встречается в правой части S_α, а $\overline{F}_{l,\beta,q}(J)$ – в левой части S_β, то имеет место **антизависимость**. Если обе функции $\overline{F}_{l,\beta,q}(J)$ и $\overline{F}_{l,\beta,q}(J)$ встречаются в левых частях операторов, то зависимость является **зависимостью во выходу**.

Иногда рассматривают **зависимость по входу**:

1) S_α выполняется раньше S_β;

2) S_α и S_β используют как аргумент один и тот же элемент какого-либо массива (т.е. $\overline{F}_{l,\alpha,p}(I)=\overline{F}_{l,\beta,q}(J)$ для некоторых l, p, q, причем обе функции встречаются в правых частях операторов);

3) между операциями S_α и S_β этот элемент не используется.

Антизависимости и зависимости по выходу являются ложными, искусственными.

Локальность при выполнении какой-либо операции означает расположение (хранение) аргумента операции непосредственно перед использованием в памяти с быстрым доступом, где он оказался вследствие участия в более ранней операции.

Обозначим $\rho_{l,\beta,q}=\operatorname{rank}F_{l,\beta,q}$. Рассмотрим базис пространства Z^{n_β} с базисными векторами $u_i^{\perp}$, $1\le i\le\rho_{l,\beta,q}$, u_i, $1\le i\le n_\beta-\rho_{l,\beta,q}$, где u_i — базисные векторы подпространства $\ker F_{l,\beta,q}$, если $\rho_{l,\beta,q}=n_\beta$, то векторы u_i отсутствуют.

Обозначим через $\left(\Phi_{\alpha,\beta}\right)_{\xi_1}$ и $\left(\Psi_{\alpha,\beta}\right)_{\xi_1}$ строки матриц с номером ξ_1; если $\xi_1>n_\alpha$, то примем $\left(\Phi_{\alpha,\beta}\right)_{\xi_1}=0$, $\left(\Psi_{\alpha,\beta}\right)_{\xi_1}=0$, $\varphi^{(\alpha,\beta)}=0$.

Т е о р е м а 1.2.1.1. Пусть элемент массива, связанный с вхождением (l,β,q) в правую часть оператора, используется в некотором виртуальном процессоре. Элемент массива определяется в этом же процессоре, если вхождение (l,β,q) порождает истинную зависимость и выполняются условия

$$\left(\Phi_{\alpha,\beta}\right)_{\xi_1}=\kappa_{\alpha,\xi_1}\kappa_{\beta,\xi_1}e_{\xi_1}^{(n_\beta)},\left(\Psi_{\alpha,\beta}\right)_{\xi_1}=$$
$$\kappa_{\alpha,\xi_1}\left(b^{(\beta,\xi_1)}-b^{(\alpha,\xi_1)}\right), \qquad (1.2.1.2)$$

$$\varphi_{\xi_1}^{(\alpha,\beta)}=\kappa_{\alpha,\xi_1}\left(a_{\alpha,\xi_1}-a_{\beta,\xi_1}\right). \qquad (1.2.1.3)$$

Если условия (1.2.1.2) выполняются, но не выполняются условия (1.2.1.3), то элемент массива определяется в процессоре, координата которого отличается от координаты данного процессора на $a_{\beta,\xi_1}-a_{\alpha,\xi_1}+\kappa_{\alpha,\xi_1}\varphi_{\xi_1}^{(\alpha,\beta)}$. Элемент массива используется в данном процессоре на вхождении (l,β,q) (возможно, меняя свое значение) на итерациях подпространства итераций размерности $\kappa_{l,\beta,q}^{s}$, где $\kappa_{l,\beta,q}^{s}=n_\beta-\rho_{l,\beta,q}^{s}$ [7].

Т е о р е м а 1.2.1.2. Если $\rho_{l,\beta,q}^{\Phi}=\rho_{l,\beta,q}^{\Phi,s}$ то на вхождении (l,β,q) фиксированное данное используется только в одном процессоре; если $\rho_{l,\beta,q}^{\Phi}<\rho_{l,\beta,q}^{\Phi,s}$, то данное используется во многих процессорах. На вхождении (l,β,q) фиксированное данное используется в фиксированном процессоре на итерациях подпространства итераций размерности $k_{l,\beta,q}^{\Phi,s}$, где $k_{l,\beta,q}^{\Phi,s}=n_\beta-\rho_{l,\beta,q}^{\Phi,s}$ [7].

Теоремы 1.2.1.1 и 1.2.1.2 позволяют выделить следующие возможные случаи использования данных на вхождении (l,β,q) в правую часть оператора [4].

1. Вхождение (l,β,q) порождает истинную зависимость и выполняются условия (1.2.1.2), (1.2.1.3). Каждый элемент массива, связанный с вхождением (l,β,q) определяется (может быть, многократно) и используется в одном процессоре. Степень многократности использования определяется величиной $k_{l,\beta,q}^{s}$; если $k_{l,\beta,q}^{s}=0$, то использование однократное.

2. Вхождение (l,β,q) истинную зависимость не порождает или не выполняется хотя бы одно из условий (1.2.1.2), (1.2.1.3), выполняются условия $\rho_{l,\beta,q}^{\Phi}=\rho_{l,\beta,q}^{\Phi,s}$, $k_{l,\beta,q}^{\Phi,s}>1$. Каждое данное, используемое на вхождении (l,β,q), требуется доставить в один процессор для многократного использования. Степень многократности использования определяется величиной $k_{l,\beta,q}^{\Phi,s}$.

3. Вхождение (l,β,q) истинную зависимость не порождает или не выполняется хотя бы одно из условий (1.2.1.2), (1.2.1.3), выполняются условия $\rho_{l,\beta,q}^{\Phi}<\rho_{l,\beta,q}^{\Phi,s}$, $k_{l,\beta,q}^{\Phi,s}>1$. Каждое данное, используемое на вхождении (l,β,q), требуется доставить во многие процессоры для многократного использования в каждом. Степень многократности использования определяется величиной $k_{l,\beta,q}^{\Phi,s}$.

4. Вхождение (l,β,q) истинную зависимость не порождает или не выполняется хотя бы одно из условий (1.2.1.2), (1.2.1.3), выполняются условия $\rho^{\Phi}_{l,\beta,q} = \rho^{\Phi,s}_{l,\beta,q}$, $k^{\Phi,s}_{l,\beta,q} = 1$. Каждое данное, используемое на вхождении (l,β,q), требуется доставить в один процессор для однократного использования.

5. Вхождение (l,β,q) истинную зависимость не порождает или не выполняется хотя бы одно из условий (1.2.1.2), (1.2.1.3), выполняются условия $\rho^{\Phi}_{l,\beta,q} < \rho^{\Phi,s}_{l,\beta,q}$ $k^{\Phi,s}_{l,\beta,q} = 1$. Каждое данное, используемое на вхождении (l,β,q), требуется доставить во многие процессоры для однократного использования.

2. Тайлинг

Одним из основных способов уменьшения накладных расходов на использование иерархической памяти является тайлинг [8]. При тайлинге операции алгоритма разбиваются на тайлы, т.е. на множества операций, выполняемых атомарно, как одна единица вычислений.

Пусть в гнезде циклов имеется Θ наборов выполняемых операторов, и в окружении каждого набора есть хотя бы один разбиваемый цикл. Обозначим:

V^{ϑ}, $1 \le \vartheta \le \Theta$, – области изменения параметров циклов, окружающих наборы операторов;

n^{ϑ} – размерность области V^{ϑ}, число циклов, окружающих -й набор операторов;

Установим необходимые и достаточные условия, при выполнении которых множества $V^{\vartheta}_{J^{gl}}$ являются тайлами.

У т в е р ж д е н и е 2.3.1. Пусть существует хотя бы одна такая зависимость $S_\alpha(I) \to S_\beta(J)$, $I \in V^{\vartheta^\alpha}_{I^{gl}}$, $J \in V^{\vartheta^\beta}_{J^{gl}}$, что в окружении операторов S_α и S_β имеется некоторое количество $c_{\alpha,\beta}$ общих циклов, имеется хотя бы один общий разбиваемый цикл и равны параметры всех внешних не разбиваемых циклов: $\left(j^{gl}_1,\ldots,j^{gl}_{k_1-1}\right) = (i^{gl}_1 \ldots, i^{gl}_{k_1-1})$, $k_1 \ne 1$. Обобщённый тайлинг является допустимым тогда и только тогда, когда для любой такой зависимости выполняются следующие условия:

$$\left(j^{gl}_{k_1},\ldots,j^{gl}_{c_{\alpha,\beta}}\right) \ge_{lex} (i^{gl}_{k_1}\ldots,i^{gl}_{c_{\alpha,\beta}}) \qquad (2.2.1)$$

$$\text{if}\left(j^{gl}_{k_1},\ldots,j^{gl}_{c_{\alpha,\beta}}\right) = \left(i^{gl}_{k_1},\ldots,i^{gl}_{c_{\alpha,\beta}}\right) \text{then } \vartheta^\beta \ge \vartheta^\alpha \qquad (2.2.2)$$

При отсутствии указанных зависимостей тайлинг допустим [8].

В следующем утверждении 2.2.2 сформулированы удобные для использования достаточные условия корректности тайлинга.

У т в е р ж д е н и е 2.2.2. Пусть существует хотя бы одна такая зависимость $S_\alpha(I(i_1,\ldots,i_{n_\alpha})) \to S_\beta(J(j_1,\ldots,j_{n_\beta}))$, что в окружении операторов S_α и S_β имеется некоторое количество $c_{\alpha,\beta}$ общих циклов, имеется хотя бы один общий разбиваемый цикл и равны параметры всех внешних не разбиваемых циклов: $\left(i_1,\ldots,i_{k_1-1}\right) = \left(j_1,\ldots,j_{k_1-1}\right)$, $k \ne 1$; пусть ϑ^β и ϑ^α — номера наборов операторов, которым принадлежат S_β и S_α соответственно. Обобщённый тайлинг является допустимым, если для любой такой зависимости выполняются следующие условия:

$$j \ge i_k, \quad k_1 \le k \le c_{\alpha,\beta}, \qquad (2.2.3)$$

$$\vartheta^\beta \ge \vartheta^\alpha. \qquad (2.2.4)$$

При отсутствии указанных зависимостей тайлинг допустим [8].

3. Улучшение локальности композиции параллельных алгоритмов

Локальность – это вычислительное свойство алгоритма, отражающее степень использования памяти с быстрым доступом. рассматривается композиция параллельных алгоритмов. Подход для неё следующий: каждый алгоритм (каждое гнездо циклов) отображается на параллельный компьютер сам по себе. Наша задача – получить условия на эти отображения, обеспечивающие хорошую локальность «стыковочных» данных

Пусть у нас имеется зависимость $S_\alpha(I) \to S_\beta(J)$, $I \in V^{\vartheta^\alpha}_{I^{gl}}$, $J \in V^{\vartheta^\beta}_{J^{gl}}$, которую описывает функция зависимостей вида (1.1.4), а именно:

$$\bar{\Phi}_{\alpha,\beta}(J) = \Phi_{\alpha,\beta}J + \Psi_{\alpha,\beta}N - \varphi^{(\alpha,\beta)}. \qquad (3.3.1)$$

Так как данная функция зависимостей отображает $\bar{\Phi}_{\alpha,\beta} : V^{\vartheta^\beta}_{J^{gl}} \to V^{\vartheta^\alpha}_{I^{gl}}$, то будем считать, что координаты виртуальных процессоров, выполняющих операции $S_\beta(J)$ многомерного цикла, задаются функциями вида $I = \bar{\Phi}_{\alpha,\beta}(J)$, $I = \Phi_{\alpha,\beta}J + \Psi_{\alpha,\beta}N - \varphi^{(\alpha,\beta)}$.

Для каждого ϑ, $1 \le \vartheta \le \Theta$ зафиксируем один из параметров циклов $j_{\zeta_{\vartheta\beta}}$. Обозначим $e_{\zeta_{\vartheta\beta}}$ – вектор размера n_β, у которого координата с номером $\zeta_{\vartheta\beta}$ равна 1. Обозначим через $\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta\beta}}$ и $\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta\beta}}$ строки матриц с номером $\zeta_{\vartheta\beta}$; если $\zeta_{\vartheta\beta} > n_\beta$, то примем $\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta\beta 1}} = 0$, $\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta\beta}} = 0$, $\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta\beta}} = 0$.

Поставим в соответствие тайлам $V^{\vartheta}_{J^{gl}}$ функции вида

$$Pr^{\vartheta}\left(J^{gl}\right) = j^{gl}_{\vartheta} . \quad (3.3.2)$$

Нам необходимо определить последовательности зернистых вычислений (т.е. разбитых на тайлы). К одной последовательности отнесём операции тайлов с одинаковыми значениями функции (3.3.2). Данные функции Pr^{ϑ} можно использовать для распределения операций алгоритма между процессорами: организуется вычислительные процесс для выполнения на процессоре с номером $Pr^{\vartheta}(J)$ операций тайлов $V^{\vartheta}_{J^{gl}}$. Задача выбора функций Pr^{ϑ} – задача распределения операций между процессорами – должна быть согласована с задачей распределения массивов данных между процессорами. От степени согласованности распределения операций и массивов данных зависит локальность параллельных реализаций алгоритмов.

Для операторов S_α и S_β функцию (3.3.2) можно записать следующим образом:

$$Pr^{\vartheta^\alpha}\left(I^{gl}\right) = i^{gl}_{\vartheta^\alpha} . \quad (3.3.3)$$

$$Pr^{\vartheta^\beta}\left(J^{gl}\right) = j^{gl}_{\vartheta^\beta} . \quad (3.3.4)$$

В теореме 3.3.1. [6] сформулированы достаточные условия, при выполнении которых мы получаем, что вычисленные данные попадают в именно тот процесс, в котором эти данные нужны, т.е. определение данных и использование данных происходит в одном вычислительном процессе.

Т е о р е м а 3.3.1. Пусть элемент массива a_l определяется на вхождении $(a_l, S_\alpha, 1)$ и используется на вхождении (a_l, S_β, q) в правой части оператора S_β. Определение и использование данных происходят в одном вычислительном процессе, если выполняются следующие условия:

$$\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}} = e_{\zeta_{\vartheta^\beta}} , \quad (3.3.5)$$

$$r^{\vartheta^\alpha}_{\zeta} = r^{\vartheta^\beta}_{\zeta} , \quad (3.3.6)$$

$$\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}} N - \varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}} - m^{\vartheta^\alpha}_{\zeta} = -m^{\vartheta^\beta}_{\zeta} . \quad (3.3.7)$$

Можно сформулировать следствие из теоремы 3.3.1 [9].

С л е д с т в и е 3.3.1. Пусть элемент массива a_l определяется на вхождении $(a_l, S_\alpha, 1)$ и используется на вхождении (a_l, S_β, q) в правой части оператора S_β. Если $\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}} = 0$, то определение и использование данных происходят в одном вычислительном процессе, если выполняются следующие условия:

$$\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}} = e_{\zeta_{\vartheta^\beta}} , \quad (3.3.11)$$

$$r^{\vartheta^\alpha}_{\zeta} = r^{\vartheta^\beta}_{\zeta} , \quad (3.3.12)$$

$$\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}} + m^{\vartheta^\alpha}_{\zeta} = m^{\vartheta^\beta}_{\zeta} . \quad (3.3.13)$$

Таким образом, в данном случае мы получили условия для любого N.

Покажем на двух примерах из пунктов 3.1 и 3.2, что выполняются условия теоремы 3.3.1 и решение, полученное геометрическим путём, верно.

1. Алгоритм перемножения трёх матриц.

Функция зависимостей алгоритма имеет вид:

$$\overline{\Phi_{1,2}}(i,j,k) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} i \\ j \\ k \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} N ,$$

$$V_{l,2} = \{(i,j,k) \in Z^3 | 1 \le i,j,k \le N\} .$$

При анализе алгоритма геометрическим путём [6] мы в первой части алгоритма разбивали цикл по j (уровня вложенности 2), во второй части – цикл по k (уровня вложенности 3). Т.е. у нас $\zeta_{\vartheta^\alpha} = 2$, $\zeta_{\vartheta^\beta} = 3$.

$$\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=2} = (001),\ e_{\zeta_{\vartheta^\beta}=3} = (001) .$$

Т.е условие (3.3.5) выполняется: $\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=2} = e_{\zeta_{\vartheta^\beta}=3}$.

Для выполнения условия (3.3.6) выберем $r = r^{\vartheta^\alpha}_{\zeta=2} = r^{\vartheta^\beta}_{\zeta=3}$.

$\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=2} = 0$, поэтому N можно не вычислять и для выполнения последнего условия воспользоваться следствием 3.3.1.

$$\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}=2} = 0,\ m^{\vartheta^\alpha}_{\zeta=2} = 1,\ m^{\vartheta^\beta}_{\zeta=3} = 1 .$$

И проверим условие (3.3.13):

$$\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}=2} + m^{\vartheta^\alpha}_{\zeta=2} = m^{\vartheta^\beta}_{\zeta=3} .$$

$0+1=1$, $1=1$, т.е. условие (3.3.13) выполняется. Значит, определение и использование данных (в нашем примере это элементы массива $c(i,j)$) происходят в одном вычислительном процессе.

2. Алгоритм метода матричной прогонки.

Функция зависимостей алгоритма имеет вид:

$$\overline{\Phi_{4,6}}(i,j,k)=\begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}i\\j\\k\end{pmatrix}+\begin{pmatrix}1\\1\\0\end{pmatrix}M+\begin{pmatrix}0\\0\\1\end{pmatrix}.$$

$$V_{4,6}=\{(i,j,k)\in Z^3 | 1\le i\le M,$$
$$1\le j\le M, 1\le k\le M-1\}.$$

Сначала рассмотрим случай, если в первой части алгоритма разбиваем цикл по j (уровня вложенности 2), во второй части – также цикл по j (уровня вложенности 2). Т.е. у нас $\zeta_{\vartheta^\alpha}=2$, $\zeta_{\vartheta^\beta}=2$.

$\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=2}=(010)$, $e_{\zeta_{\vartheta^\beta}=2}=(010)$, т.е условие (3.3.5) выполняется: $\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=2}=e_{\zeta_{\vartheta^\beta}=2}$.

Для выполнения условия (3.3.6) выберем $r=r^{\vartheta^\alpha}_{\zeta=2}=r^{\vartheta^\beta}_{\zeta=2}$.

Проверим выполнение условия (3.3.7).

$\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=2}=1$, $N=M$.

$\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}=2}=0$, $m^{\vartheta^\alpha}_{\zeta=2}=2$, $m^{\vartheta^\beta}_{\zeta=2}=1$.

И проверим условие (3.3.6):

$\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=2} N-\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}=2}-m^{\vartheta^\alpha}_{\zeta=2}=-m^{\vartheta^\beta}_{\zeta=2}$.

$0\cdot M-0-2\ne -1$, $M\ne 1$, т.е. условие (3.3.7) не выполняется. Значит, данный случай нам не подходит.

Теперь рассмотрим случай, который мы получили при анализе алгоритма геометрическим путём: мы в первой части алгоритма разбивали цикл по i (уровня вложенности 3), во второй части – цикл по k (уровня вложенности 3). Т.е. у нас $\zeta_{\vartheta^\alpha}=3$, $\zeta_{\vartheta^\beta}=3$.

$\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=3}=(001)$, $e_{\zeta_{\vartheta^\beta}=3}=(001)$. Т.е условие (3.3.5) выполняется: $\left(\Phi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=3}=e_{\zeta_{\vartheta^\beta}=3}$.

Для выполнения условия (3.3.6) выберем $r=r^{\vartheta^\alpha}_{\zeta=3}=r^{\vartheta^\beta}_{\zeta=3}$.

$\left(\Psi_{\alpha,\beta}\right)_{\zeta_{\vartheta^\alpha}=3}=0$, поэтому N можно не вычислять и для выполнения последнего условия воспользоваться следствием 3.3.1.

$\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}=3}=-1$, $m^{\vartheta^\alpha}_{\zeta=3}=2$, $m^{\vartheta^\beta}_{\zeta=3}=1$.

И проверим условие (3.3.13):

$\varphi^{(\alpha,\beta)}_{\zeta_{\vartheta^\alpha}=3}+m^{\vartheta^\alpha}_{\zeta=3}=m^{\vartheta^\beta}_{\zeta=3}$.

$-1+2=1$, $2=2$, т.е. условие (3.3.13) выполняется.

Значит, определение и использование данных (в нашем примере это элементы массива $cfb(i,j)$) происходят в одном вычислительном процессе.

Выводы

Таким образом, получены достаточные условия отображения параллельных алгоритмов, которые обеспечивают хорошую локальность «стыковочных» данных. Исследована локальность данных, определяемых в одном, а используемых в другом гнезде циклов. Мы показали на двух примерах, что теорема работает и решение, полученное геометрическим путём [9], совпадает с решением, полученным теоретическим путём.

Полученные результаты могут использоваться при автоматизации распараллеливания алгоритмов на языке Ada.

Литература

1. Воеводин, В.В. Параллельные вычисления [Текст] / Воеводин В.В. – СПб.: БХВ–Петербург, 2002. – 600 с.

2. Лиходед, Н.А. Методы распараллеливания гнезд циклов: Курс лекций [Текст] / Н.А. Лиходед. – Мн.: БГУ, 2007. – 100 с.

3. Ada 95 Language Reference Manual ANSI/ISO 8652.1995–std: [Electronic resource]. – http://www.adapower.com/rm95/index.html. – 12.02.2012 г.

4. Киркоров, С.И. Параллельные алгоритмы математических моделей: исследование локальности и применение языка Ada [Текст] / С.И. Киркоров, Л.С. Киркорова // ВІСНИК Харківського національного університету імені В.Н. Каразіна, №863 Серія: Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. – 2009. – Вип. 12. – С. 129 – 142.

5. Lim, A.W. Blocking and array contraction across arbitrary nested loops using affine partitioning [Text] / A.W. Lim, M.S. Liao // Proceedings of the ACM SIGPLAN Simposium on Principles and Practice of Programming Languages. – 2001. – P. 23 – 35.

6. Boulet, P. (Pen)–Ultimate Tiling Integration [Text] / P. Boulet, A. Darte, T. Risset, Y. Robert // The VLSI J. – 1994. – Vol. 17. – P. 33 – 51.

7. Лиходед, Н.А. Характеристика локально–сти параллельных реализаций многомерных циклов [Текст] / Н.А. Лиходед // Доклады НАН Беларуси. – 2010 – Т. 54, № 1. – С. 26 – 32.

8. Лиходед, Н.А. Обобщенный тайлинг [Текст] / Н.А. Лиходед // Доклады НАН Беларуси. – 2011. – Т. 35, №1. – С. 33 – 39.

9. Киркорова, Л.С. Улучшение локальности композиции параллельных алгоритмов [Текст] / Л.С. Киркорова; Магистерская диссертация, Институт подготовки научных кадров НАН Беларуси. – 2011.

Поступила в редакцию 19.03.2012

Рецензент: канд. техн. наук, доцент В.О. Мищенко, Харьковский национальный университет им. В.Н. Каразина, Харьков, Украина

ПОЛІПШЕННЯ ЛОКАЛЬНОЇ КОМПОЗИЦІЇ ПАРАЛЕЛЬНИХ АЛГОРИТМІВ МОВИ ADA

Л.С. Кіркорова, С.І. Кіркоров

У даній статті розглянута композиція паралельних алгоритмів і поліпшення локальності алгоритмів: кожне гніздо циклів (кожен алгоритм) відображається на паралельний комп'ютер сам по собі. Досліджується локальність даних, що визначаються в одному, а використовуються в іншому гнізді циклів. Виводяться умови, що дозволяють вибирати в цих гніздах цикли, що призводять до узгодження вхідних / вихідних даних при композиції складових складного алгоритму. Розглядаються два ілюстраційні прикладу - алгоритм множення трьох матриць і алгоритм методу матричної прогонки.

Ключові слова: мова Ada, паралельний алгоритм, гнізда циклу, тайлинг.

IMPROVING OF LOCALITY OF THE COMPOSITION OF PARALLEL ALGORITHMS OF LANGUAGE ADA

L.S. Kirkorava, S.I. Kirkorau

Composition of parallel algorithms and improvement of locality of algorithms is considered in this article: every nest of cycles (every algorithm) is represented on a parallel computer in itself. The local data defined in one, and used in a different nested loops. We derive conditions that allow you to choose in these nested loops that lead to the harmonization of input / output data when the composition is a complex algorithm. We consider two illustrative examples - the algorithm of multiplication of three matrices and the method of matrix factorization algorithm.

Key words:. Ada Language, parallel algorithm, nested loops, Tiling

Киркорова Любовь Сергеевна – магистр физико-математических наук, ведущий специалист НПП «МедиаСкан», Минск, Республика Беларусь.

Киркоров Сергей Иванович – руководитель НПП «МедиаСкан», Минск, Республика Беларусь.