ПРОЕКТИРОВАНИЕ И ДИАГНОСТИКА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

УДК 681.3.06:681.323

М.С. Тарков

ОТОБРАЖЕНИЕ ПОЛУГРУППОВЫХ ОПЕРАЦИЙ НАД МАССИВАМИ НА ВЫЧИСЛИТЕЛЬНУЮ СИСТЕМУ С ТОРОИДАЛЬНОЙ СТРУКТУРОЙ

Предложен алгоритм отображения полугрупповых операций над массивами на распределенные вычислительные системы с тороидальной структурой, основанный на использовании схемы сдваивания и отображении этой схемы на гиперкуб с последующим *XOR*-вложением гиперкуба в тор. Показано, что предложенный алгоритм дает меньшее время выполнения полугрупповой операции на торе, чем алгоритм, использующий последовательность циклических сдвигов элементов данных.

Ключевые слова: *распределенные многопроцессорные системы*, *гиперкуб*, *тор*, *полугрупповые операции*, *отображение.*

Распределенная вычислительная система представляет собой совокупность пар процессор – память, называемых процессорными элементами (или просто процессорами) и связанных друг с другом сетью линий связи, где каждая линия соединяет два процессора. Такая сеть межпроцессорных соединений описывается графом. В современных суперкомпьютерных распределенных системах в качестве графа сети связи обычно используется трехмерный тор [1 – 4].

Основу многомерного (*k*-мерного) тора образует многомерная «решетка» из $p_1 \times p_2 \times ... \times p_k$ вершин, где каждая вершина имеет метку $(a_1, a_2, ..., a_k)$, $a_i \in \{0,1,..., p_i - 1\}$, $i = 1,2,...,k$. Две вершины $(a_1, a_2, ..., a_k)$ и $(b_1, b_2, ..., b_k)$ являются соседними в решетке, если для некоторого i выполнено $a_i = b_i \pm 1$ и для всех $j \neq i$ выполнено $a_j = b_j$. Многомерный тор образуется «зацикливанием» строк и столбцов решетки. Он описывается графом из $p_1 \times p_2 \times ... \times p_k$ вершин, где вершины $(a_1, a_2, ..., a_k)$ и $(b_1, b_2, ..., b_k)$ являются соседними, если для некоторого i выполнено $a_i = b_i \pm 1 \bmod p_i$ и для всех $j \neq i$ выполнено $a_j = b_j$.

Полугрупповой операцией называют бинарную ассоциативную операцию $\otimes$ [5]. Примерами таких операций могут служить сумма, произведение, конъюнкция, дизъюнкция, исключающее ИЛИ, а также вычисление максимума или минимума. В [5] предложен параллельный алгоритм выполнения такой операции над массивом данных, распределенных по процессорам матричной сети («решетки»), причем результат операции должен получить каждый процессор. Алгоритм сводится к выполнению последовательности циклических сдвигов в линейной сети («линейке») процессоров с выполнением операции $\otimes$ при каждом сдвиге. Такие сдви-

ги выполняются для всех измерений решетки. Этот алгоритм легко переносится на тороидальную сеть процессоров.

В данной работе предлагается альтернативный подход к отображению полугрупповых операций на вычислительные системы с тороидальной структурой, основанный на использовании в полугрупповых операциях схемы «бабочка» [4, 6], отображении этой схемы на гиперкуб с последующим *XOR*-отображением гиперкуба на тор [2, 4]. Показано, что такой подход при использовании отображения [2] дает меньшее время параллельного выполнения полугрупповой операции на торе, чем подход, предложенный в [5].

1. Выполнение полугрупповой операции на решетке и торе с использованием циклических сдвигов данных

Пусть массив $x = (x_1, ..., x_n)$ изначально распределен по одному элементу данных в каждом процессоре двумерной решетки из $\sqrt{n} \times \sqrt{n}$ процессоров, то есть каждый процессор P_{ij} изначально содержит соответствующий элемент $x_{i\sqrt{n}+j}$, $i, j \in \{1, ..., \sqrt{n}\}$. Требуется применить полугрупповую операцию $\otimes$ ко всем исходным значениям массива x так, чтобы каждый процессор получил результат применения операции.

В [5] предложен алгоритм выполнения полугрупповых операций на решетке процессоров, состоящий в выполнении последовательности циклических сдвигов с выполнением операции $\otimes$ при каждом сдвиге:

1. Параллельно в каждой строке *i* выполнить циклический сдвиг данных так, что каждый процессор в строке получает результат $r_i = \otimes_{j=1}^{n^{1/2}} x_{i\sqrt{n}+j}$.

2. Параллельно в каждом столбце *j* выполняется циклический сдвиг данных так, что каждый процессор в столбце получает результат $s' = \otimes_{i=1}^{n^{1/2}} r_i$, равный желаемому результату $s = \otimes_{i=1}^{n^{1/2}} \otimes_{j=1}^{n^{1/2}} x_{i\sqrt{n}+j}$.

Циклический сдвиг в строке (или столбце) решетки процессоров выполняется так: каждый элемент перемещается от процессора к процессору вправо, пока не попадет в самый правый процессор, далее направление его сдвигов меняется на противоположное (гусеничный алгоритм [5]). Сдвиги в торе отличаются от сдвигов в решетке тем, что, благодаря зацикливанию строк и столбцов, направление сдвига не меняется.

2. Выполнение полугрупповой операции на гиперкубе с использованием схемы сдваивания

Приведенные в предыдущем разделе алгоритмы выполнения полугрупповой операции над массивом данных в торе не являются оптимальными. Эту операцию можно выполнить быстрее, если использовать систему параллельных процессов, структура которой называется «бабочкой» (рис. 1). В этой структуре схема сдваивания [6], являющаяся оптимальной для реализации полугрупповых операций, реализует размножение вычислений с максимальным количеством одновременно выполняемых операций над разными парами аргументов.

Бабочка легко отображается в гиперкуб. Для этого достаточно объединить операции тех процессов, которые не могут выполняться одновременно. На рис. 1 объединяемые процессы (операции) лежат на одной вертикальной линии. На рис. 2 показан полученный в результате слияния процессов гиперкуб. Здесь числа в скобках показывают номер шага взаимодействий между узлами структуры.

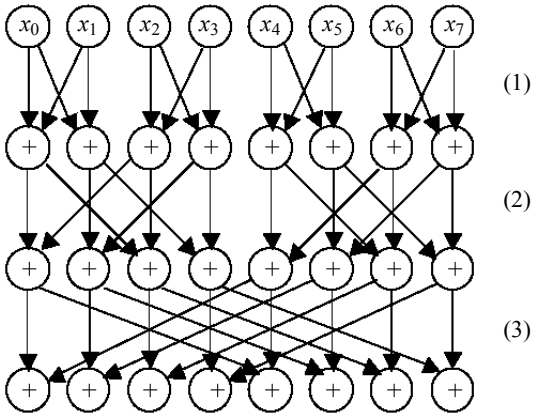

Рис. 1. Бабочка

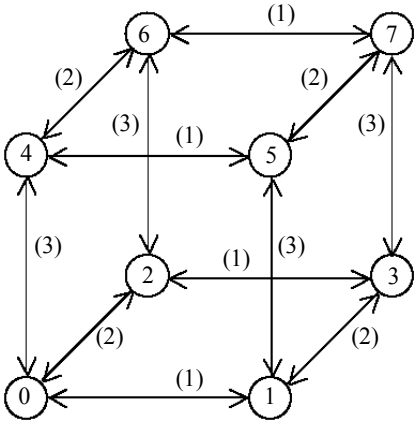

Рис. 2. Гиперкуб, полученный слиянием процессов двоичного дерева или бабочки

3. Вложение гиперкуба в тор

Полученный слиянием процессов бабочки гиперкуб может быть вложен в тор [2, 4]. Способ отображения гиперкуба в тор (*XOR*-вложение) предложен в [2]. Вложение графа G в граф H является инъекцией f вершин G в вершины H. Если граф G не изоморфен подграфу H, то неизбежны растяжения ребер графа G. Растяжением ребра (a,b) графа G на графе H называется расстояние (длина кратчайшего пути) между вершинами $f(a)$ и $f(b)$.

XOR-вложение гиперкуба H_d в k-мерный тор $E_k\left(2^{d_1},...,2^{d_k}\right)$, $\sum_{i=1}^{k} d_i = d$ реализуется следующим образом. Сначала определяются K_j:

$$K_1 = 0,$$
$$K_j = \sum_{i=1}^{j-1} d_i, \quad 1 < j \le k+1.$$

Если граф G является гиперкубом H_d, а граф T – тором, то вершина v графа G отображается в вершину $(m_1,...,m_k) = f_{XOR}(v)$ графа T следующим образом [2]:

$$m_j(i) = v(i + K_j), i \in \left[0, d_j - 1\right], i \ne d_j - 2,$$
$$m_j(d_j - 2) = XOR\left(v\left(K_{j+1} - 1\right), v\left(K_{j+1} - 2\right)\right).$$

Здесь $x(i)$ – i-й бит двоичного представления x. Показано [2], что гиперкуб H_d может быть вложен в тор $E_k\left(2^{d_1},...,2^{d_k}\right)$, где $\sum_{i=1}^{k} d_i = d$, со средним растяжением

$$D = \frac{\left(\sum_{i=1}^{k} 3 \cdot 2^{d_i - 2}\right) - k}{d}. \qquad (1)$$

В табл. 1 и 2 приведены вычисленные по формуле (1) средние значения растяжения D для нескольких значений числа машин $n = 2^d$ при $k = 2,3$ соответственно, $d_i = \frac{d}{k}$, $i = 1,...,k$.

Таблица 1

Среднее растяжение ребер гиперкуба на двумерном торе

n	16	64	256	1024	4096	16384	65536
D	1	1,667	2,75	4,6	7,833	13,5714	23,8750

Таблица 2

Среднее растяжение ребер гиперкуба на трехмерном торе

n	64	512	4096
D	1	1.667	2.75

В общем случае результатом отображения гиперкуба в тор являются пути на торе вместо ребер в гиперкубе. При вложении одного графа в другой возникающие при этом пути могут пересекаться, поэтому кроме растяжений ребер могут возникать конфликты путей (перегрузки каналов связи) на ребрах тора. Эти перегрузки могут привести к увеличению продолжительности межмашинных обменов данными.

Например (рис. 3), пусть вершины A и B одновременно посылают сообщения a и b в вершины A_1 и B_1 соответственно. Если вершина C транслирует сообщение a в вершину D как первое сообщение, то сообщение b будет задержано в вершине C и наоборот. Это и есть перегрузка дуги (канала) (C, D).

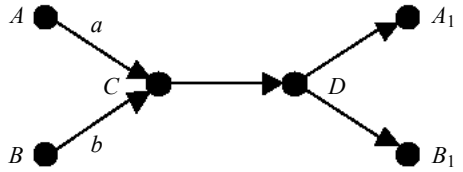

Рис. 3. Конфликт сообщений из вершин A и B на дуге (C, D)

Утверждение 1. *XOR*-вложение гиперкуба в тор не создает перегрузок.

Доказательство. 1. Сначала рассмотрим одномерный тор (кольцо) и произвольное отображение гиперкуба в тор, то есть произвольную нумерацию вершин. Пусть два произвольных пути с различными вершинами-источниками и вершинами-приемниками пересекаются на кольце (т.е. имеют общие ребра) и оба сообщения начинают передаваться одновременно.

Если эти пути направлены противоположно, то перегрузок нет, поскольку каждое ребро используется для одновременной передачи сообщений по двум противоположно направленным дугам (каналам).

Если два пути ориентированы одинаково, как на рис. 4, то перегрузок также нет, поскольку передачи начинаются одновременно. Когда некоторое сообщение поступает в вершину для последующей трансляции, то соответствующая выходная дуга уже свободна, поскольку передача выходного сообщения по ней завершена.

Например, на рис. 4 передача сообщения из вершины B в вершину D не задерживает передачу сообщения из вершины A в вершину C , так как сообщения из A в B и из B в C передаются одновременно.

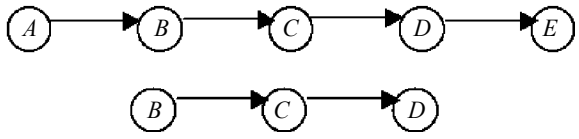

Рис. 4. Одинаково направленные пути (*A*,*B*,*C*,*D*,*E*) и (*B*,*C*,*D*)

2. Рассмотрим общий случай *XOR*-вложения. Взаимодействия между процессами в структуре «бабочка» на d-мерном гиперкубе выполняются за d шагов. На шаге

$$s \in \{1,...,d\} \tag{2}$$

вершина v взаимодействует с вершиной v', если $|v - v'| = 2^{s-1}$.

Рассмотрим стандартное отображение d-мерного гиперкуба в k-мерный тор $E_k(n_1,...,n_k)$

$$n_i = 2^{d_i},\ \prod_{i=1}^{k} n_i = 2^d \tag{3}$$

в виде $f(v) = (p_1,...,p_k)$, где

$$p_i = \left(v \bmod \prod_{j=1}^{i} n_j \right) \operatorname{div} \prod_{j=1}^{i-1} n_j,\ i = 1,...,k, \tag{4}$$

где div обозначает деление нацело.

Две различных вершины v и v' лежат в m-м одномерном торе (кольце), если $f(|v-v'|)=\left(\underbrace{0,...,0}_{m-1}, p_m, \underbrace{0,...,0}_{n-m}\right)$, $p_m \neq 0$. Покажем, что для $s \in \{1,...,d\}$ существует $m \leq k$, такое, что

$$f(|v-v'|)=f\left(2^{s-1}\right)=\left(\underbrace{0,...,0}_{m-1}, 2^{s-\sum_{i=1}^{m-1} d_i-1}, \underbrace{0,...,0}_{n-m}\right). \quad (5)$$

Из (2) и (3) следует, что существует $m \leq k$, такое, что

$$\prod_{i=1}^{m-1} 2^{d_i} \leq 2^{s-1} < \prod_{i=1}^{m} 2^{d_i} . \quad (6)$$

Из (6) следует, что:

1) для всех $i \in \{1,...,m-1\}$ имеем $2^{s-1} \bmod \prod_{j=1}^{i} 2^{d_j} = 0$. Тогда из (4) следует, что $p_i\left(2^{s-1}\right)=0,\ i=1,...,m-1$;

2) $p_m\left(2^{s-1}\right)=\dfrac{2^{s-1}}{\prod_{i=1}^{m-1} 2^{d_i}}=2^{s-\sum_{i=1}^{m-1} d_i-1}$;

3) для всех $i \in \{m+1,...,n\}$ справедливы равенства $2^{s-1} \bmod \prod_{j=1}^{i} 2^{d_j} = 2^{s-1}$ и $2^{s-1} \operatorname{div} \prod_{j=1}^{i-1} 2^{d_j} = 0$, поскольку $2^{s-1} < \prod_{j=1}^{i-1} 2^{d_j}$ для $i > m$.

Выражение (5) доказано. Из (5) следует, что:

1) для стандартного вложения две любых взаимодействующих вершины гиперкуба принадлежат одномерному тору;

2) две любые пары взаимодействующих вершин либо принадлежат одному и тому же одномерному тору, либо двум непересекающимся одномерным торам. В обоих случаях в соответствии с пунктом 1 доказательства перегрузка каналов отсутствует.

3. Рассмотрим *XOR*-вложение для общего случая. Из (5) следует, что для любых двух взаимодействующих вершин v и v' стандартное вложение имеет вид

$$f(v)=(p_1,...,p_{m-1},p_m,p_{m+1},...,p_k),$$
$$f(v')=\left(p_1,...,p_{m-1},p_m \pm 2^{s-\sum_{i=1}^{m-1} d_i-1},p_{m+1},...,p_k\right),$$

где s,m,k удовлетворяют соотношениям (2),(3),(6).

XOR-вложение изменяет одни и те же биты в компонентах отображений $f(v)$ и $f(v')$, поэтому вложения $f_{XOR}(v)$ и $f_{XOR}(v')$ различаются только m-й компонентой. Следовательно, эти две вершины принадлежат одномерному тору, и для *XOR*-вложения также отсутствуют перегрузки каналов.

Утверждение 1 доказано.

4. Сравнительный анализ времени выполнения полугрупповой операции на тороидальной сети процессоров

Пусть t_w – время передачи элемента данных между соседними процессорами и t_o – время выполнения полугрупповой операции над двумя аргументами. Обозначим T_C – время параллельного выполнения полугрупповой операции на торе с использованием циклических сдвигов данных, а T_{HT} – время выполнения той же операции на гиперкубе, вложенном в тор.

Утверждение 2. $T_C > T_{HT}$.

Доказательство. Для произвольного k и

$$\sum_{i=1}^{k} d_i = d,\ d_i \geq 1 \tag{7}$$

время выполнения полугрупповой операции на k – мерном торе с использованием циклических сдвигов данных

$$T_C = \sum_{i=1}^{k} \left(2^{d_i} - 1\right)(t_w + t_o).$$

Время выполнения полугрупповой операции на гиперкубе, вложенном в k-мерный тор,

$$T_{HT} = (Dt_w + t_o) \cdot d\ .$$

Подставив D из (1), получаем

$$T_{HT} = (\sum_{i=1}^{k} 3 \cdot 2^{d_i - 2} - k)t_w + dt_o.$$

Тогда с учетом (7)

$$T_C - T_{HT} = \left[\sum_{i=1}^{k}\left(2^{d_i} - 1\right) - \sum_{i=1}^{k} 3 \cdot 2^{d_i - 2} + k\right]t_w + \left[\sum_{i=1}^{k}\left(2^{d_i} - 1\right) - d\right]t_o =$$

$$= t_w \cdot \frac{3}{4}\sum_{i=1}^{k} 2^{d_i} + t_o \cdot \sum_{i=1}^{k}\left(2^{d_i} - d_i - 1\right) > 0,$$

поскольку $2^{d_i} - d_i - 1 \geq 0$ при $d_i \geq 1$.

Утверждение 2 доказано.

В частности, время выполнения полной полугрупповой операции на k-мерном торе из $\underbrace{\sqrt[k]{n} \times \sqrt[k]{n} \times \ldots \times \sqrt[k]{n}}_{k}$ процессоров с использованием циклических сдвигов данных равно

$$T_C = k(\sqrt[k]{n} - 1) \cdot (t_w + t_o).$$

Время выполнения той же операции на гиперкубе, вложенном в k-мерный тор из $\underbrace{\sqrt[k]{n} \times \sqrt[k]{n} \times \ldots \times \sqrt[k]{n}}_{k}$ процессоров

$$T_{HT} = \log_2 n \cdot (t_o + Dt_w) = \log_2 n \cdot t_o + \left(k \cdot \frac{3}{4}\sqrt[k]{n} - 2\right)t_w.$$

Отсюда $$T_C - T_{HT} = \left[k\left(\frac{\sqrt[k]{n}}{4} - 1 \right) + 2 \right] t_w + \left[k\left(\sqrt[k]{n} - 1 \right) - \log_2 n \right] t_o , \qquad (8)$$

где $k\left(\sqrt[k]{n}-1\right)-\log_2 n \geq 0$ при $n \geq 2^k$. Из (8) следует, что для k-мерного тора из $\underbrace{\sqrt[k]{n} \times \sqrt[k]{n} \times \ldots \times \sqrt[k]{n}}_{k}$ процессоров при $n \to \infty$ выполняется асимптотическое соотношение

$$T_C - T_{HT} = O\left(\sqrt[k]{n}\right) \cdot (t_w + t_o) .$$

5. Заключение

Предложен алгоритм отображения полугрупповых (бинарных ассоциативных) операций над массивами на распределенные вычислительные системы с тороидальной структурой, основанный на использовании схемы «бабочка» и отображении этой схемы на гиперкубическую структуру ВС с последующим *XOR*-вложением гиперкуба в тор. Показано, что *XOR* -вложение гиперкуба в тор не приводит к конфликтам при передаче сообщений (перегрузкам каналов). Получены оценки времени выполнения алгоритма. Показано, что:

1. Несмотря на растяжение ребер гиперкуба при вложении его в тор, предложенный алгоритм дает меньшее время выполнения полугрупповой операции на торе, чем алгоритм, использующий последовательность циклических сдвигов элементов данных в торе.

2. При увеличении числа процессоров ВС (и соответственно числа элементов массива данных) выигрыш по времени выполнения полугрупповой операции на торе при использовании предложенного алгоритма возрастает.

ЛИТЕРАТУРА

1. *Yu H.*, *Chung I-Hsin*, *Moreira J.* Topology mapping for blue gene/L supercomputer // Proc. of the ACM/IEEE SC2006 Conf. on High Performance Networking and Computing, November 11 – 17, 2006, Tampa, FL, USA: ACM Press, 2006. P. 52–64.
2. *Gonzalez A.*, *Valero-Garcia M.*, *Diaz de Cerio L.* Executing algorithms with hypercube topology on torus multicomputers // IEEE Trans. on Parallel and Distributed Systems. 1995. V. 6. No. 8. P. 803–814.
3. *Абрамов С.М.*, *Заднепровский В.Ф.*, *Шмелев А.Б.*, *Московский А.А.* СуперЭВМ ряда 4 семейства «СКИФ»: штурм вершины суперкомпьютерных технологий // Вестник Нижегородского университета им. Н.И. Лобачевского. 2009. № 5. С. 200–210.
4. *Тарков М.С.* Отображение нейросетевых алгоритмов анализа изображений на регулярные структуры распределенных вычислительных систем // Известия Томского политехнического университета. 2008. Т. 313. № 5. С. 101–105.
5. *Миллер Р.*, *Боксер Л.* Последовательные и параллельные алгоритмы: Общий подход. М.: БИНОМ. Лаборатория знаний, 2006. 406 с.
6. *Foster I.* Designing and building parallel programs [Электронный ресурс]. URL: http://www-unix.mcs.anl.gov/dbpp

Тарков Михаил Сергеевич
Института физики полупроводников СО РАН
E-mail: tarkov@isp.nsc.ru

Поступила в редакцию 18 ноября 2011 г.