УДК 004.2

ЖАБИНА В.В.

ПАРАЛЛЕЛЬНОЕ ФОРМИРОВАНИЕ КОМАНД В ПОТОКОВЫХ СИСТЕМАХ

Предложен метод ускорения вычислений в системах, управляемых потоком данных, при реализации алгоритмов с ветвлениями, основанный на динамическом формировании параллельных потоков команд. Показана возможность идентификации слов акторов и данных на основе графа задачи в процессе компиляции для потоковых систем с несколькими средами формирования команд.

Method of computation acceleration in data-flow systems for realization of algorithms with branches and loops is proposed. It is based on dynamic parallel instruction flows compilation. There is shown a possibility to automatically identify actor and data words during compilation on the base of task graph for data-flow systems with several instruction formation units.

Введение

При решении задач управления и моделирования в реальном времени возникает необходимость реализации вычислительных алгоритмов с мелкозернистой структурой. В качестве примера таких алгоритмов можно указать алгоритмы интерполяции функций, расчета траектории объектов в многомерном пространстве. Ускорения вычислений в этом случае можно добиться распараллеливанием вычислений на уровне операций, а также аппаратной реализацией операций. Время решения параллельных задач во многом определяется эффективностью распараллеливания вычислений.

Большинство из современных технологий параллельного программирования относятся к средствам статического распараллеливания процессов [1]. Задачи распараллеливания в этом случае решаются на этапе разработки программ. При статическом анализе алгоритмов не всегда удается выявить параллельные ветви, то есть скрытый параллелизм, что объясняется недостатком информации о динамике процессов в системе.

Одним из перспективных подходов, позволяющих устранить ряд недостатков статического планирования, является разработка средства динамического распараллеливания вычислений. В этом случае назначение заданий на вычислительные узлы осуществляется системой в процессе решения задач. Такой подход дает возможность достичь большей степени параллелизма, так как позволяет выявить параллельные ветви, которые возникают непосредственно в процессе вычислений.

Одним из подходов для динамического распределения заданий между вычислительными узлами является использование модели вычислений, управляемых потоком данных (потоковой модели). Распределение операций между вычислительными узлами в этом случае может быть реализовано автоматически на аппаратном или микропрограммном уровне.

Реализация данной модели вычислений для реализации мелкозернистых алгоритмов может быть обеспечена применением ПЛИС, которые содержат вычислительные ядра, модули памяти и средства коммуникации. Использование такой элементной базы и технологии SoC (System on chip) дают возможность эффективной реализации параллельных вычислений на уровне операций.

Поскольку динамическое распределение заданий осуществляется средствами самой системы, то важной задачей является уменьшение непроизводительных расходов времени на этот процесс.

Методы организации вычислений, управляемых потоком данных

К одним из первых работ в области организации вычислительных систем, управляемых потоком данных (потоковых систем), можно отнести работы [2-6]. Имея структурные особенности, такие системы используют общую модель вычислений. Потоковые системы содержат несколько вычислительных модулей (ВМ) (устройств обработки информации) и среду формирования команд (СФК), связанных коммуникационными средствами (КС).

В системах, управляемых потоком данных, команды выполняются не в заданной программой последовательности, а при наличии готовых данных (наличии всех операндов), то есть определяющим в данном случае является не порядок выполнения команд, а доступность

данных для команды.

Подготовка вычислений осуществляется на основе графа, каждой i-й вершине которого соответствует операция (функция), а каждой дуге – данные.

Операция для i-й вершины графа описывается информационным словом, которое называют актором (actor). Актор описывается кортежем

$$A_i = \langle I_i, F_i, N_i, T_i \rangle, \quad (1)$$

где I_i – идентификатор (уникальное имя) данного актора; F_i – функция преобразования данных (код операции); N_i – имя актора, для которого i-й актор подготавливает операнд, а T_i – совокупность признаков этого операнда.

Акторы связаны между собой только по данным. Каждой дуге графа соответствует слово данных

$$D_i = \langle I_i, Q_i, N_i, T_i \rangle, \quad (2)$$

где Q_i – значение операнда.

Из соответствующих элементов A_i и D_i в СФК формируется команда, которая выполняется в свободном ВМ или помещается в очередь.

Известны различные алгоритмы формирования команд [2-5]. Для организации СФК используется ассоциативная память или ее эмуляция с применением других технических средств.

Компиляторы позволяют автоматизировать подготовку акторов и данных, причем, без учета конкретного числа ВМ в системе.

Если время формирования команд в СФК меньше времени выполнения команд в ВМ, то в разных ВМ одновременно могут выполняться разные команды, за счет чего и достигается распараллеливание операций. Современные технологии реализации ВМ, в том числе, с использованием ПЛИС, позволяют применять аппаратные методы ускорения операций, в результате чего интенсивность формирования команд в СФК становится недостаточной для загрузки нескольких ВМ одновременно.

В связи с этим важной проблемой является увеличение интенсивности потока готовых команд с целью ускорения параллельных вычислений.

Простое дублирование СФК приводит к недостаткам, присущим статическим методам подготовки параллельных программ. Программист должен предварительно разрезать граф задачи на подграфы, предопределить идентификаторы акторов и данных в разных подграфах с учетом связи по данным между ними.

Более эффективным является подход, который позволяет автоматически формировать акторы и данные при наличии нескольких СФК [6, 7]. Система содержит модули, в состав которых входит СФК и ВМ. Модули организуются в кольцевую структуру. Данные циркулируют в такой структуре в поисках своего актора. Команды формируются в разных СФК и выполняются в соответствующих ВМ.

Недостатком такого подхода являются затраты времени на пересылку данных между модулями системы. Кроме того, выход из строя любого модуля приводит к неработоспособности всей системы, поскольку информация передается последовательно от одного модуля к другому.

В работе [8] предложен метод автоматического распределения акторов и данных между несколькими СФК, позволяющий автоматически формировать параллельные потоки команд. Однако указанный метод может быть использован в случае, когда граф задачи не имеет разветвлений потоков данных.

В данной работе предлагается метод параллельного формирования потоков команд, который не требует ручного вмешательства в процесс назначения идентификаторов соответствия акторов и данных различным СФК и может быть использован для алгоритмов, в которых допускается разветвление потоков данных.

Метод формирования параллельных потоков команд

Организация основных компонентов потоковой системы поясняется рис. 1. Система может иметь произвольное число ВМ. Количество СФК должно быть равно $k=2^j$, где $j=1,2,3\ldots$

Формирование акторов и данных осуществляется на основе графа.

Модифицируем форматы объектов (1) и (2). Формат акторов принимает следующий вид

$$A_i = < M_i, I_i, F_i, K_i, L_i, T_i >, \quad (3)$$

где M_i – идентификатор СФК (номер потока команд); I_i – уникальное имя данного актора; F_i – функция преобразования данных (код операции); K_i – количество акторов, для которых i-й

актор подготавливает операнд; $L_i = <I_j, M_j>$ – список имен акторов I_j с идентификаторами M_j, для которых i-й актор подготавливает операнд; T_i – тип данных.

Данные запишем в виде

$$D_i = < Q_i, N_i, L_i, T_i >, \qquad (4)$$

где Q_i – значение операнда; N_i – количество акторов, для которых передается данный операнд.

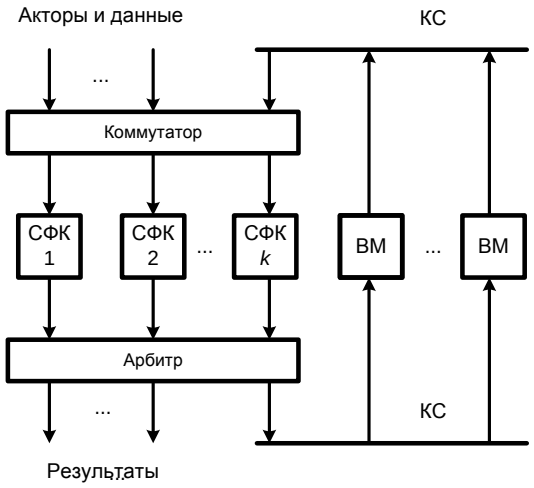

Рис. 1. Структура потоковой системы

Для формирования потоков команд, то есть нахождения значений идентификаторов СФК M_i, предлагается следующий алгоритм.

1. На основе исходного графа построить матрицу смежности, показывающую связь между акторами по данным. В качестве номеров строк и столбцов взять имена акторов I_i. или их индексы i. Первому потоку присвоить идентификатор M_0.

2. Обнулить диагональ полученной матрицы.

3. Для формирования параллельных потоков команд начать циклический просмотр новой матрицы смежности. В каждом цикле от первой до последней строки выполнять:

а) если текущая строка не содержит единичных элементов, перейти к следующей исходной строке;

б) если строка с индексом i содержит единичные элементы, то в качестве первого актора в потоке команд взять актор с именем I_i;

в) актор с именем, индекс которого равен номеру столбца первого единичного элемента в текущей строке, добавить в данный поток команд, а все единичные элементы данного столбца обнулить;

д) если новое имя актора в потоке команд уже было добавлено в него ранее, то закончить формирование очередного потока в соответствии с п. 3.ж;

г) в противном случае перейти к строке, номер которой соответствует номеру столбца, определенного в п. 3.в;

д) если строка не содержит единичных элементов, закончить формирование очередного потока команд в соотв. с п. 3.ж;

е) если строка содержит единичные элементы, то продолжить формирование текущего потока в соотв. с данным алгоритмом, начиная с п. 3.б;

ж) полученному потоку присвоить соответствующий идентификатор СФК (если поток первый, то M_0, в остальных случаях M_{i+1}, где M_i – имя предыдущего потока); перейти к следующей строке для формирования следующего потока команд;

4. Если в результате прохода матрицы в ней остались единичные элементы, то проход повторить с п. 3; если единичных элементов нет, то выполнить п. 5.

5. Составить список акторов I_i в соответствии с форматом (3) и учетом связей, заданных матрицей смежности, а также идентификаторов M_i.

6. Аналогичным образом составить список данных D_i.

При выполнении алгоритма реальное число ВМ и СФК в системе не учитывается. В результате выполнения алгоритма формируется максимально возможное число параллельных потоков команд. Если в системе имеется только одна СФК, то идентификаторы M_i системой игнорируется. При наличии нескольких СФК, число которых выбирается по формуле $k = 2^j$ $(j = 1,2,3...)$, реально будет формироваться столько потоков команд, сколько СФК имеется в системе. При распределении объектов (акторов и данных) между СФК учитываются только $\lceil \log_2 k \rceil$ младших разрядов двоичных кодов идентификаторов M_j. Например, для множества идентификаторов $\{M_0, M_1, M_2, M_3, M_4, M_5\}$ соответственно с кодами $\{000, 001, 010, 011, 100, 101\}$ при наличии в системе двух СФК учитывается только один младший разряд кодов, а при четырех СФК – два младших разряда. Соответственно в системе будут формироваться два или четыре потока команд.

Пример формирования параллельных потоков команд

Формирование параллельных потоков команд рассмотрим на примере графа, представленного на рис. 2.

На основе графа формируем матрицу смежности, как показано в табл.1.

В соответствии с приведенным алгоритмом формируем параллельные потоки команд. При первом проходе матрицы получаем последовательность команд

M_0: 1-5-10-14-16-19-21-22.

В табл. 1 рамками отмечены первые значимые элементы в строках при первом проходе матрицы. Затемненные единичные элементы обнуляются при первом проходе.

При дальнейших проходах аналогично получаем:

M_1: 2-6-11-15-17-20;

M_2: 3-7-12;

M_3: 4-8-13-18.

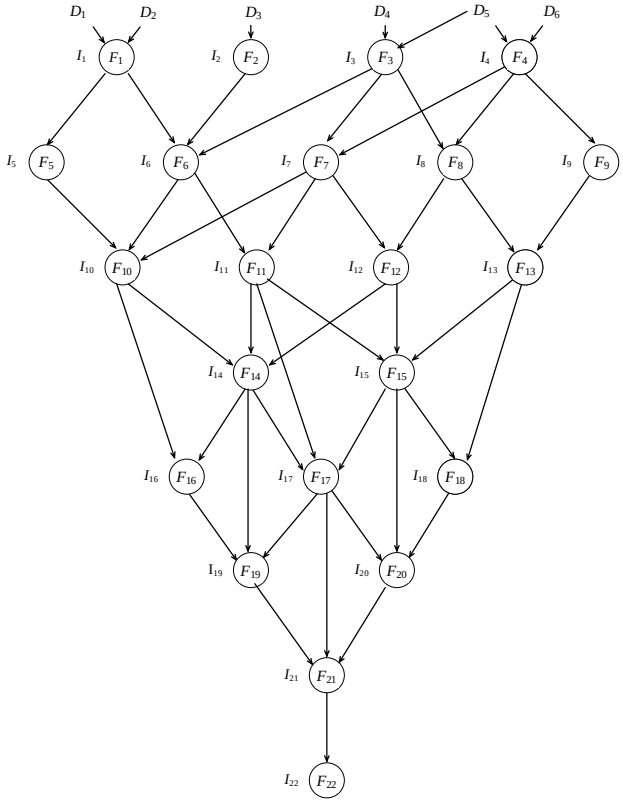

Рис. 2. Граф задачи

Поскольку матрица смежности не обнулена полностью, начинаем обход сначала и получаем последнюю строку потока команд

M_4: 9.

На основе полученных данных описываем каждый актор в соответствии с форматом (3):

$A_1 =< M_0, I_1, F_1, 2, I_5, M_0, I_6, M_1, T_1 >;$

$A_2 =< M_1, I_2, F_2, 1, I_6, M_1, T_2 >;$

$A_3 =< M_2, I_3, F_3, 3, I_6, M_1, I_7, M_2, I_8, M_3, T_3 >;$

$A_4 =< M_3, I_4, F_4, 3, I_7, M_2, I_8, M_3, I_9, M_4, T_4 >;$

$A_5 =< M_0, I_5, F_5, 1, I_{10}, M_0, T_5 >;$

$A_6 =< M_1, I_6, F_6, 2, I_{10}, M_0, I_{11}, M_1, T_6 >;$

$A_7 =< M_2, I_7, F_7, 3, I_{10}, M_0, I_{11}, M_1, I_{12}, M_2, T_7 >;$

$A_8 =< M_3, I_8, F_8, 2, I_{12}, M_2, I_{13}, M_3, T_8 >;$

$A_9 =< M_4, I_9, F_9, 1, I_{13}, M_3, T_9 >;$

$A_{10} =< M_0, I_{10}, F_{10}, 2, I_{14}, M_0, I_{16}, M_0, T_{10} >;$

$A_{11} =< M_1, I_{11}, F_{11}, 3, I_{14}, M_0, I_{15}, M_1, I_{17}, M_1, T_{11} >;$

$A_{12} =< M_2, I_{12}, F_{12}, 2, I_{14}, M_0, I_{15}, M_1, T_{12} >;$

$A_{13} =< M_3, I_{13}, F_{13}, 2, I_{15}, M_1, I_{18}, M_3, T_{13} >;$

$A_{14} =< M_0, I_{14}, F_{14}, 3, I_{16}, M_0, I_{19}, M_0, I_{17}, M_1, T_{14} >;$

$A_{15} =< M_1, I_{15}, F_{15}, 3, I_{17}, M_1, I_{20}, M_1, I_{18}, M_3, T_{15} >;$

$A_{16} =< M_0, I_{16}, F_{16}, 1, I_{19}, M_0, T_{16} >;$

$A_{17} =< M_1, I_{17}, F_{17}, 3, I_{19}, M_0, I_{21}, M_0, I_{20}, M_1, T_{17} >;$

$A_{18} =< M_3, I_{18}, F_{18}, 1, I_{20}, M_1, T_{18} >;$

$A_{19} =< M_0, I_{19}, F_{19}, 1, I_{21}, M_0, T_{19} >;$

$A_{20} =< M_1, I_{20}, F_{20}, 1, I_{21}, M_0, T_{20} >;$

$A_{21} =< M_0, I_{21}, F_{21}, 1, I_{22}, M_0, T_{21} >;$

$A_{22} =< M_0, I_{22}, F_{22}, 1, I_{22}, M_0, T_{22} >.$

Описываем данные в соответствии с форматом (4):

$D_1 =< Q_1, 1, I_1, M_0, T_1 >;$

$D_2 =< Q_2, 1, I_1, M_0, T_2 >;$

$D_3 =< Q_3, 1, I_2, M_1, T_3 >;$

$D_4 =< Q_4, 1, I_3, M_2, T_4 >;$

$D_5 =< Q_5, 2, I_3, M_2, I_4, M_3, T_5 >;$

$D_6 =< Q_6, 1, I_4, M_3, T_6 >.$

Выводы

Предложенный метод формирования параллельных потоков команд в системах, управляемых потоком данных, имеет ряд преимуществ по сравнению с программным управлением параллельными вычислениями. Управляющие слова и данные могут быть сформированы автоматически компилятором на основе графа. Нет необходимости предварительно учитывать число вычислителей в системе и обеспечивать синхронизацию обмена данными между параллельными ветвями алгоритма. Динамическое распределение операций позволяет выявить непосредственно в процессе вычислений скрытый параллелизм, связанный с различными длительностями обработки

данных в различных ветвях алгоритмов. Вычислительные модули могут выполнять операции, относящиеся к различным задачам, в любой последовательности. В связи с этим в системе могут одновременно решаться независимые задачи, причем, нет необходимости синхронизации выполнения задач.

По сравнению с кольцевой организацией потоковых систем предложенный метод позволяет ускорить вычисления за счет сокращения непроизводительных затрат времени на последовательную пересылку данных между модулями системы. Кроме того, это способствует повышению надежности систем.

По сравнению с известным методом формирования параллельных потоков команд предложенный метод расширяет область применения потоковых систем за счет реализации алгоритмов с разветвлениями.

Все это повышается эффективность обработки данных в потоковых системах.

Табл. 1. Матрица смежности

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Список литературы

1. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.
2. *Dennis J. B., Missunas D. P.* A preliminary architecture for basic data flow processor// Proc. 2nd Annual Symp. Comput. Stockholm, May 1975. N. Y. IEEE. – 1975. – P. 126 – 132.
3. *Silva J.G.D., Wood J.V.* Design of processing subsystems for Manchester data flow computer // IEEE Proc. N.Y. – 1981. – Vol. 128, N 5. – P. 218 – 224.
4. *Watson R., Guard J.* A practical data flow computer // Computer. – 1982. – Vol. 15, N 2.– P. 51 – 57.
5. *Hogenauer E.B., Newbold R.F. Inn Y.T.* DDSP – a data flow computer for signal processing/ Proc. Int. Conf. Parall. Process. Ohio, August 1982. N.Y. // IEEE. – 1982. – P. 126 – 133.
6. *Johnson D.* Data flow machines threaten the program counter// Electronic Design. – 1980. – N 22. – P. 255 – 258.
7. Функционально ориентированные процессоры */ Водяхо А.Н., Смолов В.Б., Плюснин В.У., Пузанков Д.В. /* Под ред. *В.Б.Смолова.* – Л.: Машиностроение. Ленингр. отд-ние, 1988. – 224 *с.*

8. Жабина В.В. Повышение эффективности параллельной обработки данных на уровне операций в потоковых системах // Вісник Національного технічного університету України “Київський політехнічний інститут”, Інформатика, управління та обчислювальна техніка,: Зб. наук. пр. – К.: „ВЕК+”. – 2008. – №49. – С. 112-116.

Жабина Валентина Валериевна – аспирант кафедры программного обеспечения компьютерных систем НТУУ «КПИ».