УДК 004.3

ЖАБИНА В.В.,
КЕДА А.Ю.

ИССЛЕДОВАНИЕ МОДЕЛИ ПОТОКОВОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ С ПАРАЛЛЕЛЬНЫМ ФОРМИРОВАНИЕМ КОМАНД

Предложена имитационная модель потоковой вычислительной системы с формированием параллельных потоков команд. Исследуется зависимость времени реализации мелкозернистых алгоритмов в зависимости от количества вычислительных модулей, сред формирования команд, а также соотношения числа длинных и коротких операций. Показана возможность автоматического выявления скрытого параллелизма задач.

Simulator of the data-flow system with parallel instruction flows generation is proposed. Dependence of fine grained algorithm realization time on number of processors and instruction generation units, as well as on short and long instructions proportion is investigated. Possibility to discover hidden parallelism is shown.

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 и т. д. Ускорения вычислений в этом случае можно добиться распараллеливанием алгоритмов на уровне операций.

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.

Перспективным подходом, позволяющим 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.

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

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, что сокращает затраты времени.

Потоковая модель вычислений

В системах, управляемых потоком данных (СУПД), команды выполняются не в заданной программой последовательности, а при наличии готовых данных (готовности 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 вершине которого соответствует операция (функция), а каждой дуге – данные.

Операция для каждой вершины графа описывается информационным словом, которое называется актором (actor). Акторы связаны между собой только по данным, которые соответствуют дугам графа.

Из соответствующих элементов акторов и данных в среде формирования команд (СФК) составляется команда, которая выполняется в свободном вычислительном модуле (ВМ) или поступает в очередь.

Для реализации данной модели вычислений могут быть использованы ПЛИС, которые

содержат вычислительные ядра, модули памяти и средства коммутации. Использование такой элементной базы и технологии SoC (System on chip) дают возможность эффективной реализации параллельных вычислений на уровне операций. При большой частоте тактирования современных ПЛИС основную задержку вычислений вносит общая СФК, построенная на основе памяти.

Известны различные способы формирования команд [2-8], эффективность которых определяется достигаемой интенсивностью потока готовых команд. Введение в систему нескольких СФК позволяет формировать параллельные потоки команд. Однако простое дублирование СФК приводит к недостатками статических методов, то есть к ручному распараллеливанию процессов: программист должен предварительно разрезать граф задачи на подграфы, сформировать идентификаторы для акторов и данных, которые приписывают их к определенным СФК.

В работе [9] показан способ автоматической идентификации объектов для систем с несколькими СФК. На основе графа задачи формируется матрица, с помощью которой определяются ветви команд, зависимых по данным (команды из разных ветвей могут выполняться одновременно).

Формат акторов имеет вид

$$A_i =< M_i, I_i, F_i, K_i, L_i, T_i >, \qquad (1)$$

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 определяются кортежем

$$D_i =< M_i, Q_i, N_i, L_i, T_i >, \qquad (2)$$

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

Из объектов (1) и (2) в разных СФК формируются параллельные потоки команд. Каждая команда составляется из объектов, имеющих одинаковые идентификаторы СФК.

Имитационная модель системы

В соответствии с указанным методом построена имитационная модель, позволяющая исследовать зависимость времени вычислений от параметров системы и характеристики алгоритмов.

Модель является комплексом программных средств, позволяющих подготавливать данные для задачи, моделировать ход вычислений и формировать данные для его анализа. Взаимодействие программных модулей системы представлено на рис. 1.

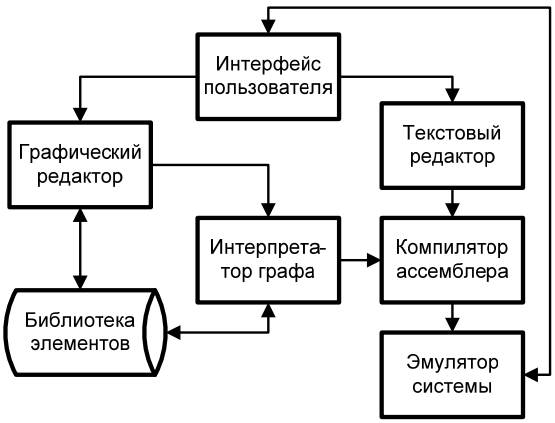

Рис. 1. Схема взаимодействия программных модулей системы

Интерфейс пользователя позволяет выбрать средство ввода алгоритма - графический или текстовый редактор.

Модуль графического редактора предоставляет возможность ввода и редактирования графа задачи, в котором вершинам соответствуют акторы, а дугам графа – данные. Модуль графического редактора передает структуру данных в модуль интерпретатора графа для дальнейшей обработки. В библиотеке элементов хранятся объекты, соответствующие определенному набору функций (преобразования данных и управления).

Модуль текстового редактора позволяет вводить и редактировать программу на специально разработанном ассемблере. Программа хранится в памяти в виде набора строк. Ссылка на программу передается компилятору ассемблера.

Модуль интерпретатора графа выполняет конвертацию графического представления, полученного графическим редактором, в код ассемблера. Для каждой вершины графа интерпретатор генерирует команду ассемблера. Данный модуль, как и текстовый редактор, передает компилятору ассемблера ссылку на программу.

Лексический анализатор представляет собой первую фазу компилятора. Его основная

функция состоит в чтении новых символов и выдаче последовательности токенов, используемых синтаксическим анализатором. Лексический анализатор построен как детерминированный конечный автомат.

Синтаксический анализатор получает строку токенов от лексического анализатора и проверяет, может ли эта строка порождаться грамматикой исходного языка. Он также сообщает обо всех выявленных ошибках. В случае удачного разбора входного потока терминалов синтаксический анализатор передает в модуль эмулятора список акторов и данных.

Эмулятор позволяет отображать состояние компонентов системы на всех стадиях обработки данных, включая процессы ввода данных, этапы формирование команд в СФК, распределение команд между ВМ и устройствами вывода данных. Фиксируется продолжительность этапов обработки данных.

Исследование модели системы

Для сравнительной оценки производительности реальных систем используются тестовые задачи с разной частотой команд, имеющих различную продолжительность выполнения, что учитывает область применения систем. При сравнении учитывается реальное время решения задачи.

На стадии проектирования системы можно сравнивать с помощью моделей по среднему времени выполнения команд $T_{\text{К}} = \sum_i p_i t_i$, где t_i – время выполнения команды i-го типа, а p_i – частота команды в программах. Частоты команд называют также смесями Гибсона. Для оценки продолжительности вичислений используются условные единицы (например, такты).

Для сравнения систем широкого применения на этапе проектирования используют упрощенную формулу $T_{\text{К}} = 0{,}7 p_{\text{К}} t_{\text{К}} + 0{,}3 p_{\text{Д}} t_{\text{Д}}$, где $p_{\text{К}}, t_{\text{К}}$ соответствуют короткой, а $p_{\text{Д}}, t_{\text{Д}}$ – длинной операции [10]. Для проблемно-ориентированных и специализированных систем могут использовать другие частоты команд.

В процессе моделирорвания определялась продолжительность вычислений в зависимости от следующих параметров:

– количество ВМ;

– количество СФК;

– продолжительность выполнения команд.

В качестве длинных рассматривалась команда умножения, а в качестве коротких – сложения.

В качестве тестовой использовалась задача, граф которой представляет бинарное дерево с высотой $h = 6$. Такой граф имеет последовательно-параллельную структуру, что соответствует большинству алгоритмов реализации численных методов при решении задач реального времени. Кроме того, при задании графа легко обеспечить требуемое соотношение длинных и коротких операций.

Исходя из соотношения длительностей выполнения длинных и коротких операций с использованием методов ускорения второго порядка для длинных операций (матричная схема умножения) [11], длительность выполнения короткой операции принималась равной длительности формирования команды в СФК, а длинной – в два раза больше. Такт моделирования принимался равным десяти условным единицам времени.

Для большого класса практических задач считаются весьма устойчивыми частоты коротких и длинных операций соответственно 0,7 и 0,3. При таких значениях частот на рис. 2 сведены результаты моделирования, при котором изменялись такие параметры системы, как количество ВМ (от 1 до 16) и количество СФК (1, 2, 4).

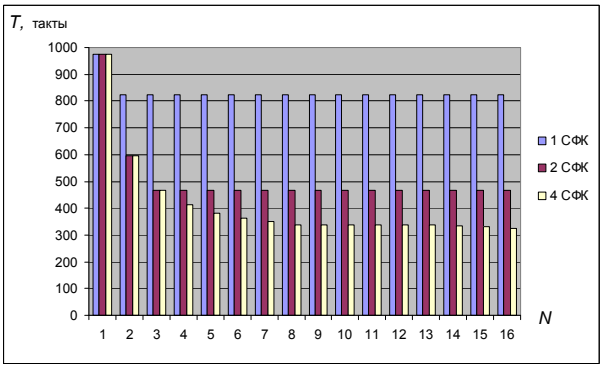

Рис. 2. Изменение времени T выполнения алгоритма при количественном соотношении коротких и длинных операций 7:3 (N – число ВМ)

В специализированных системах, связанных с решением задач цифровой обработки сигналов, аэродинамики, гидроакустики, сейсморазведки, метеорологии, управления быстрыми динамическими объектами, моделирования сплошной среды и ряда других задач, ча-

стота появления длинных операций значительно выше, чем в системах общего применения. При решении вышеуказанных задач усредненное соотношение коротких и длинных операций может изменяться в сторону увеличения частоты длинных операций.

На рис. 3 сведены результаты моделирования бинарного дерева с одинаковыми частотами для длинных и коротких операций при таких же параметрах системы, как в предыдущем случае.

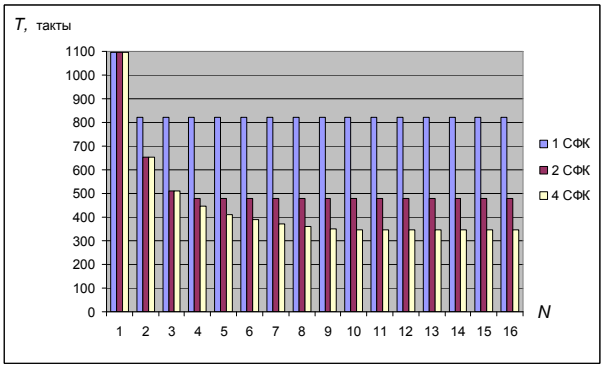

Рис. 3. Изменение времени T выполнения алгоритма при одинаковой частоте коротких и длинных операций (N – число ВМ)

Из приведенных на рисунках диаграмм можно сделать вывод, что при увеличении количества СФК время выполнения алгоритмов может быть уменьшено до определенного предела. Значение этого предела зависит, в первую очередь, от соотношения количества ВМ и СФК в системе, а также от степени параллелизма задачи. Для рассмотренных двух вариантов моделирования видно, что добавление даже одной СФК может улучить временные характеристики системы. Естественно, чем «шире» граф задачи и больше СФК, тем больший выигрыш можно получить в скорости обработки данных.

Для оценки эффективности использования нескольких СФК было рассмотрено изменение коэффициента относительно ускорения вычислений, определяемого по формуле

$$K_{\text{у}} = \frac{T_{1'} - T_2}{T_1} 100\%,$$

где в рассматриваемом случае T_1 – время выполнения алгоритма с одной СФК, а T_2 - время выполнения алгоритма с несколькими СФК.

Из приведенного на рис. 4 графика видно, что использование двух СФК позволяет ускорить вычисления более чем на 40 %, а при 4-х – почти на 60 %.

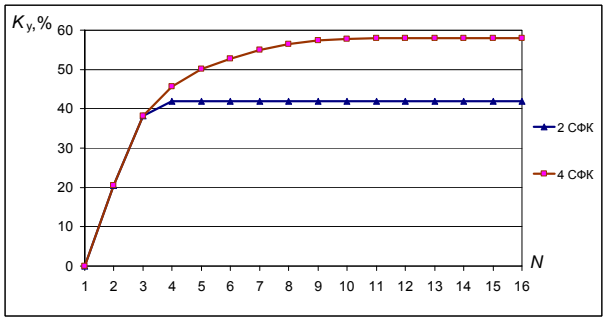

Рис. 4. Зависимость коэффициента ускорения вычислений от числа СФК и количества N ВМ

Скрытый параллелизм задач

На весьма простом примере можно показать, как в системе реализуется скрытый параллелизм задачи, который весьма трудно определить при статическом анализе алгоритма.

Из ярусно-параллельной формы алгоритма (рис. 5) видно, что на каждом ярусе находится не более двух вершин.

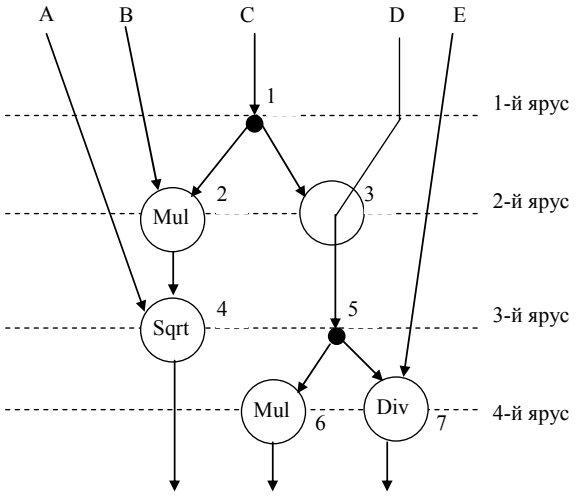

Рис. 5. Ярусно-параллельная форма алгоритма

В результате моделирования получена временная диаграмма (рис. 6), из которой следует, что при наличии в системе трех ВМ одновременно выполняются 3 операции (4, 6, 7). Это связано с разным временем выполнения операций. В частности, операция 5 выполняется быстрее операции 4.

На таком простом алгоритме, зная все времена выполнения операций и затраты времени на пересылки, несложно выделить параллель-

ные операции. Но реальные алгоритмы могут быть намного сложнее, что делает задачу распараллеливания процессов в статике очень трудной. В таких случаях применение СУПД является весьма эффективным.

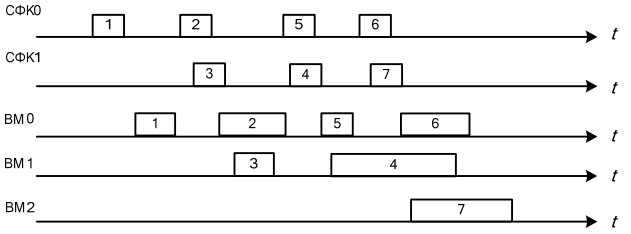

Рис. 6. Временная диаграмма выполнения алгоритма

Оценивалась возможность автоматического обнаружения скрытого параллелизма при решении систем линейных алгебраических уравнений (СЛАУ) методом Гаусса.

Метод Гаусса состоит в последовательном исключении неизвестных до тех пор, пока не останется одно уравнение с одним неизвестным (прямой ход метода). При этом матрица СЛАУ приводится к треугольному виду, где ниже главной диагонали располагаются только нули.

Обратный ход начинается с решения последнего уравнения и заканчивается определением первого неизвестного.

Оценим возможность распараллеливания вычислений при прямом и обратном ходе метода Гаусса.

Имеем $Ax = b$, где $A = [a_{ij}]$ – матрица размерности $n \times n$, $\det A > 0$, $b = (b_1, b_2, ..., b_n)$.

В предположении, что $a_{11} \neq 0$, первое уравнение системы

$$\sum_{j=1}^{n} a_{ij}^0 x_j = b_i^0,\ i = 1,...,n$$

делим на коэффициент a_{11}, в результате получаем уравнение

$$x_1 + \sum_{j=2}^{n} a_{1j}^1 x_j = b_1^1 .$$

Затем из первого уравнения вычитается каждое из остальных уравнений, деленное на соответствующий коэффициент a_{i1}. В результате эти уравнения преобразуются к виду

$$\sum_{j=2}^{n} a_{ij}^1 x_j = b_i^1,\ i = 2,...,n.$$

Первое неизвестное оказалось исключенным из всех уравнений, кроме первого. Далее предполагаем, что $a_{22}^1 \neq 0$, делим второе уравнение на a_{22}^1 и исключаем неизвестное x_2 из всех уравнений, начиная со второго, и т.д. В результате последовательного исключения неизвестных система уравнений преобразуется в систему уравнений с треугольной матрицей

$$x_i + \sum_{j=i+1}^{n} a_{1j}^i x_j = b_i^i,\ i = 1,...,n$$

.

Из n-го уравнения определяем x_n, из $(n-1)$-го – x_{n-1} и т.д. Последним определяется x_1. Совокупность таких действий называется обратным ходом метода Гаусса.

Реализация прямого хода требует $N \approx 2n^3/3$ арифметических операций. При этом параллельно могут выполняться n^2 операций. При обратном ходе требуется $N \approx n^2$ арифметических операций, причем, параллельно можно выполнять $n-1$ операцию.

При статическом распараллеливании для системы с тремя неизвестными максимальное число ветвей графа равно девяти. Моделирование проводилось с использованием 16 ВМ. Соотношение длительности коротких и длинных операций при моделировании принята такой, как в предыдущих примерах.

Как видно из результатов моделирования (табл. 1), увеличение числа СФК позволило более эффективно загрузить ВМ. При двух СФК одновременно выполняют операции 11 ВМ, а при четырех – 12 ВМ. Скрытый параллелизм связан с разбросом времени выполнения операций на параллельных ветвях программы.

Табл. 1. Результаты моделирования

Количество СФК	**Количество параллельно работающих ВМ**	**Число тактов**
1	8	509
2	11	335
4	12	255

Выводы

Метод автоматического формирования параллельных потоков команд в СУПД дает потенциальную возможность ускорить решение задач, когда интенсивность обработки данных

в ВМ превышает интенсивность формирования команд одной СФК. Акторы и данные автоматически снабжаются идентификаторами СФК на этапе компиляции программы. Нет необходимости выполнять распараллеливание в ручном режиме.

Динамическое распределение операций позволяет выявить непосредственно в процессе моделирования скрытый параллелизм задачи, связанный с различными длительностями обработки данных в различных ветвях алгоритмов, что весьма затруднительно при статическом анализе алгоритма.

Моделирование с использованием симулятора позволяет для рассматриваемых задач определить параметры системы, которые обеспечивают эффективную реализацию целевой функции. Может быть определено время решения задачи при разном количестве ВМ и СФК, что дает возможность в каждом конкретном случае выбрать необходимую конфигурацию системы.

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.
9. *Жабина В.В.* Параллельное формирование команд в потоковых системах // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка: Збірка наукових праць. – К.: Век+. – 2009. – № 50. – С. 113-117.
10. *Лощилов И.Н.* Перспективы роста производительности ЭВМ // Зарубежная радиоэлектроника. – 1976. – №5. – с. 5-25.
11. *Карцев М.А.* Брик В.А. Вычислительные системы и синхронная арифметика. – М.: Радио и связь, 1981. – 360 с.