УДК 004.3

Жабина В.В.,
Каспич И.В.

МОДЕЛЬ ПОТОКОВОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ С ПАРАЛЛЕЛЬНЫМ ФОРМИРОВАНИЕМ КОМАНД

Национальный технический университет Украины «КПИ»

Предложена имитационная модель системы, управляемой потоком данных, на базе нескольких сред формирования команд. Модель является комплексом программных средств, позволяющих подготавливать данные для задачи, выполнять моделирование хода вычислений и формировать данные для его анализа. Разработанный комплекс программных средств позволяет упростить и ускорить подготовку задач для потоковой системы

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.п. Ускорение вычислений в этом случае можно добиться путем распараллеливания вычислений. Использование статических средств подготовки параллельных задач (например, языков параллельного программирования, библиотек передачи сообщений) не всегда позволяет выявить скрытый параллелизм [1 - 4]. Кроме того, возникают трудности, связанные с необходимостью учета топологии системы, формированием и распределением заданий между вычислительными узлами системы в ручном режиме с учетом их загрузки и минимизации пересылок данных.

Одним из эффективных подходов к ускорению реализации мелкозернистых алгоритмов является динамическое распределение операций между вычислительными узлами системы с использованием модели вычислений, управляемых потоком данных [5 - 10]. Распределение операций в этом случае может быть реализовано автоматически в процессе вычислений на аппаратном уровне. Имея структурные особенности, потоковые системы используют общую модель вычислений. Они содержат одну общую среду формирования команд (СФК) и несколько вычислительных модулей (ВМ), связанных коммуникационными средствами (КС). Подготовка задачи осуществляется на основе графа. Операция для i-й вершины графа описывается информационным словом, которое называют актором (*actor*). Акторы связаны между собой только по данным. Каждой дуге графа соответствует поток данных. Из соответствующих элементов актора и данных в СФК формируется команда, которая выполняется в свободном ВМ или помещается в очередь, если нет свободных ВМ.

Для организации СФК используется ассоциативная память или ее эмуляция с применением других технических средств. При определенных условиях, например, большой длительности выполнения операций одна СФК может обеспечить такую интенсивность потока команд, при которой ВМ не будут простаивать. Однако современная элементная база позволяет использовать различные способы ускорения преобразования данных. В частности, для реализации потоковой модели вычислений могут быть использованы ПЛИС, которые содержат вычислительные ядра, модули памяти и средства коммуникации между ними. При большой частоте тактирования современных ПЛИС основную задержку вычислений вносит общая СФК, построенная на основе памяти. Простое дублирование СФК приводит к недостаткам статических методов распараллеливания. В этом случае

необходимо в ручном режиме распределять данные между разными СФК. Фактически это приводит к недостаткам статических средств распараллеливания процессов, то есть необходимо учитывать топологию системы, выделять параллельные ветви в алгоритмах, присваивать идентификаторы данным, которые определяют их принадлежность к определенным ветвям алгоритмов.

Структурная организация потоковой системы

В работах [11 - 13] показана возможность автоматического назначения идентификаторов акторам и данным для потоковых систем с несколькими СФК.

Организация системы поясняется рис. 1.

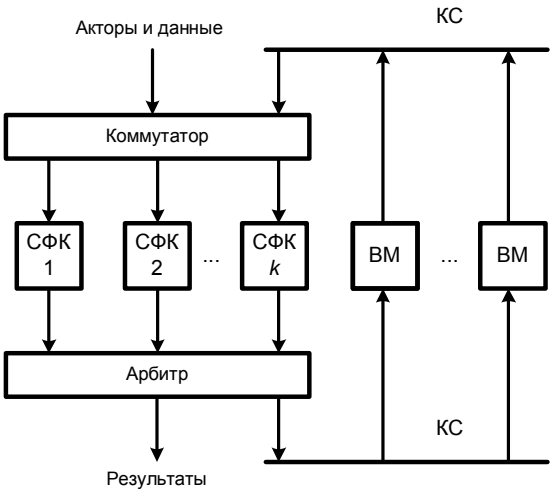

Рис. 1. Структура потоковой системы

Система может иметь любое число ВМ. Количество СФК должно быть равно $k = 2^j$ ($j = 1,2,3,...$). Данные для задачи подготавливаются компилятором автоматически на основе графа задачи.

Акторы и данные имеют следующий формат:

$$A_i = \langle M_i, I_i, F_i, N_i, T_i \rangle,$$

$$D_i = \langle M_i, I_i, Q_i, N_i, T_i \rangle,$$

где M_i – идентификатор СФК, который назначается компилятором автоматически с учетом связей акторов по данным. I_i – имя актора; F_i – функция преобразования данных; N_i – имя актора, для которого i-й актор подготавливает операнд, T_i – совокупность признаков операнда, Q_i – значение операнда.

Постановка задачи

Целью работы является разработка имитационной модели потоковой системы с параллельным формированием команд на базе нескольких СФК, описание структуры модели и ее возможностей.

Имитационная модель потоковой системы

Модель системы является комплексом программных средств, позволяющих подготавливать данные для задачи, выполнять моделирование хода вычислений и формировать данные для его анализа. Взаимодействие программных модулей системы представлено на рис. 2.

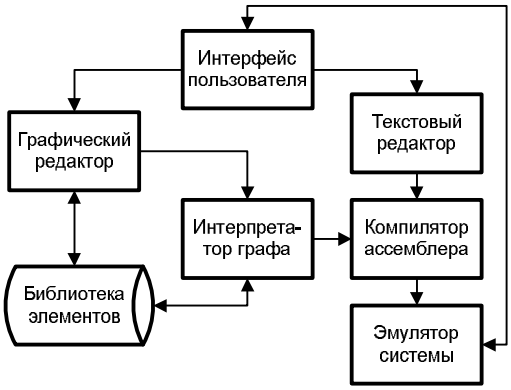

Рис. 2. Схема взаимодействия программных модулей системы

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 – операнды. Кроме того, в редакторе пользователь задает свойства вершин графа. Пример использования графического редактора представлен на рис. 3.

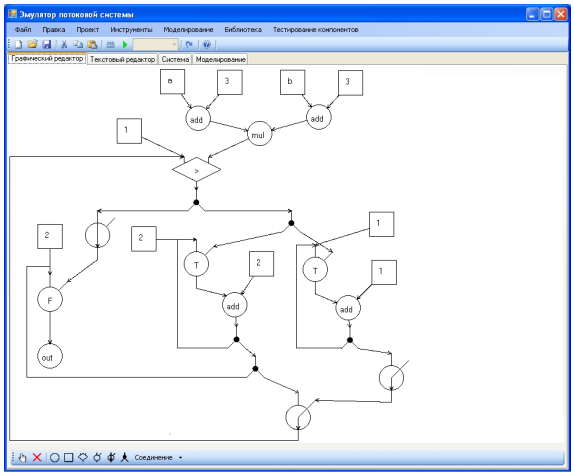

Рис. 3. Пример использования графического редактора

В библиотеке элементов хранятся объекты, которые соответствуют трем типам акторов – арифметическим операциям, командам управления и пользовательским операциям. Пользовательские операции создаются в редакторе графа на основе библиотеки стандартных операций. Каждой пользовательской операции, кроме стандартных свойств операций, соответствует набор записей, которые представляют собой подграф задачи. К стандартным свойствам операций относится форма ее графического отображения, содержимое операции, ее длительность, а также количество аргументов и результатов операции. В памяти граф задачи хранится в виде списочной структуры данных, которая содержит в себе все элементы графа – дуги и вершины. Вершина графа, кроме библиотечной операции, содержит координаты на графическом полотне, состояние актора, а также некоторое множество записей, соответствующие ссылкам на другие вершины графа, с которыми она связанна дугами. Модуль графического редактора передает эту структуру данных в модуль интерпретатора графа для дальнейшей обработки.

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. При конвертации пользовательских операций происходит обращение к библиотеке элементов и рекурсивный разбор графа таким образом, чтобы вместо одной операции получить набор команд и данных, которые отображают содержимое пользовательской операции. Таким образом, использование пользовательских операций соответствует использованию функций с последующим их раскрытием. Результат конвертации графа, изображенного на рис. 4.

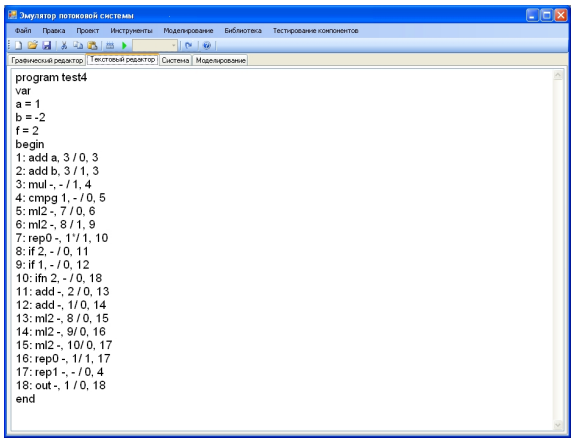

Рис. 4. Результат конвертации графа

Модуль интерпретации графа, так же, как и текстовый редактор, передает в модуль компилятора ассемблера ссылку на программу. Таким образом, компилятор получает унифицированное представление программы независимо от источника получения данных. Компилятор состоит из двух частей: лексического и синтаксического анализаторов.

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. Особенностью такого автомата является наличие для любого входного символа не более одного пере-

хода из каждого состояния. При попадании в некоторое состояние анализатор считывает следующий входной символ и, если имеется переход с меткой, соответствующей этому символу, перемещается в следующее состояние.

Лексический анализатор возвращает терминал на следующем шаге итерации, который представляет собой структуру, соответствующую типу возвращаемого терминала и значению терминала.

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 вектор акторов и операндов.

Эмулятор потоковой вычислительной системы принимает список акторов и данных программы от модуля компилятора ассемблера. Данные из модуля эмулятора системы передаются в виде списочной структуры, которая содержит два типа объектов: задачи и пересылки. Объект задача содержит поля такта, устройства выполнения задачи и длительность. Объект пересылка, кроме такта и длительности, содержит приемник, источник и тип передаваемых данных. Кроме того, конфигурирование параметров потоковой системы для эмуляции происходит в модуле интерфейса пользователя, где настраиваются такие параметры, как количество устройств ввода-вывода, временные задержки, размеры буфера данных и команд, размер памяти, количество вычислительных устройств и время выполнения стандартных операций. Эти параметры передаются с интерфейса пользователя на модуль эмулятора потоковой вычислительной системы, которая обрабатывает все полученные данные в соответствии с алгоритмами функционирования разработанной системы. Затем модуль эмулятора передает результаты моделирования в модуль интерфейса пользователя.

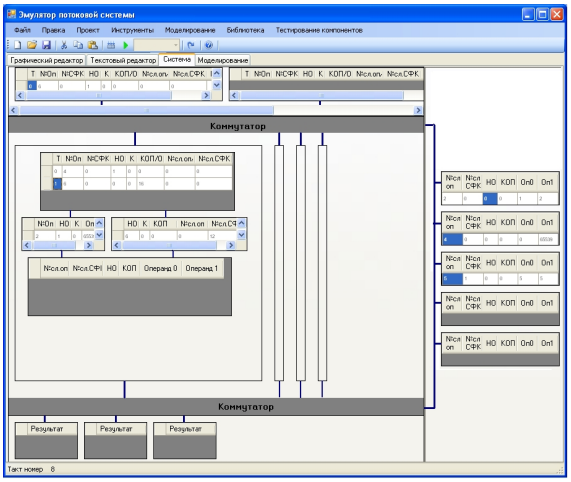

Рис. 5. Состояние компонентов системы во время моделирования

Работу потоковой системы можно наблюдать в режиме система, в котором в интерактивном режиме отображается содержимое всех основных устройств. На рис. 5 изображено одно из состояний системы во время выполнения задачи.

Результаты моделирования отображаются так же в форме диаграммы Ганта (рис. 6), которая позволяет отображать во времени процессы ввода данных из устройств ввода, формирование команд и распределение их между вычислительными блоками, а также устройствами вывода данных.

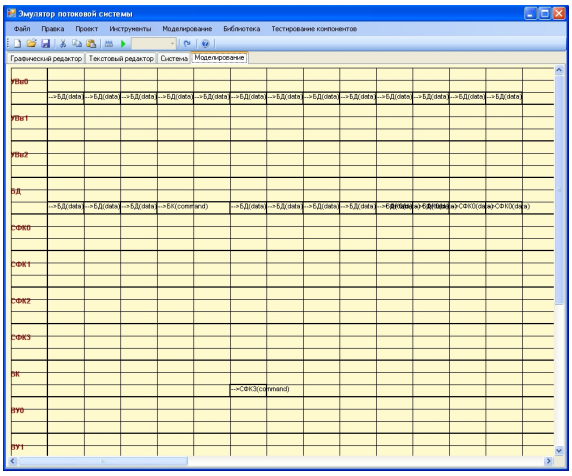

Рис. 6. Фрагмент диаграммы Ганта

Выводы

Разработанный комплекс программных средств позволяет упростить и ускорить подготовку задач для потоковой системы. Подготовка задачи сводится к со-

ставлению графа и разработке библиотеки функций, необходимых для решения задачи. Управляющие слова и данные формируются автоматически компилятором. Нет необходимости предварительно учитывать число вычислителей в системе, выделять параллельные ветви и обеспечивать синхронизацию обмена данными между ними.

Динамическое распределение операций позволяет выявить непосредственно в процессе моделирования скрытый параллелизм задачи, 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.

Результаты моделирования позволяют 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. *Антонов А.С.* Параллельное программирование с использованием технологии MPI: Учебное пособие. – М.: Изд-во МГУ, 2004. – 71 с.

3. *Корнеев В.Д.* Параллельное программирование в MPI. – Изд-во СО РАН, Новосибирск, 2000. – 213 с.

4. *Немнюгин С.А., Стесик О.Л.* Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 400 с.

5. *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.

6. *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.

7. *Watson R., Guard J.* A practical data flow computer // Computer. – 1982. – Vol. 15. – N 2. – P. 51–57.

8. *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.

9. *Johnson D.* Data flow machines threaten the program counter// Electronic Design. – 1980. – N 22. – P. 255–258.

10. 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.

11. *Жабина В.В.* Повышение эффективности параллельной обработки данных на уровне операций в потоковых системах // Вісник Національного технічного університету Украини "КПІ", Інформатика, управління і обчислювальна техника: Зб. наук. пр. – К.: „ВЕК+", 2008. – №49. – С. 112–116.

12. *Жабина В.В.* Параллельное формирование команд в потоковых системах // Вісник Національного технічного університету Украини "КПІ", Інформатика, управління і обчислювальна техника: Зб. наук. пр. – К.: „ВЕК+", 2009. – №50. – С. 113–117.

13. *Жабіна В.В.* Метод паралельного формування команд в потокових системах // Друга наукова конференція магістрантів та аспірантів, присвячена 20-річчю факультету прикладної математики, „Прикладна математика та комп'ютінг ПМК 2010", Зб. тез доп. – К.: Просвіта, 2010. – С. 234–238.

Подано до редакції 30.04.10