5. Widger J. and Grosu D. Parallel Computation of Nash Equilibria in N-Player Games, Computational Science and Engineering, IEEE International Conference, 2009. – P. 209–215.

6. Колемаев В.А., Калинина В.Н. Теория вероятностей и математическая статистика: учеб. для вузов. – 2-е изд., перераб. и доп. – М.: Юнити, 2003. – 352 с.

7. Данциг Дж. Линейное программирование, его применения и обобщения. – М.: Прогресс, 1966. – 600 с.

Е.А. Козинов

Нижегородский государственный университет им. Н.И. Лобачевского

РАЗЛИЧНЫЕ ПОДХОДЫ К ВИЗУАЛЬНОМУ ПРЕДСТАВЛЕНИЮ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

В настоящей работе рассматривается одна из актуальных проблем написания параллельных программ – снижение сложности и времени разработки. В настоящее время большая часть процессоров в различных вычислительных устройствах имеют более одного ядра. Во всех крупных компаниях используются многопроцессорные вычислительные системы. Как следствие, эффективная программа должна уметь утилизировать данные ресурсы, т.е. быть параллельной. На создание параллельной программы, как правило, уходит существенно больше времени, чем на разработку последовательной программы. Связано это с тем, что в параллельных программах могут возникать «параллельные ошибки», сложность выявления и устранения которых гораздо выше «обычных ошибок». Для снижения сложности написания параллельных программ может быть применено визуальное программирование. Визуальный язык может помочь программисту построить правильный «каркас» параллельной программы. Сгенерированный «каркас» в данном случае берет на себя всю сложность организации параллельных вычислений, оставив на программиста последовательные участки кода, отвечающие за программируемый алгоритм. Такая схема

создания параллельных программ, как правило, по производительности будет проигрывать коду, написанному полностью вручную, но позволяет сократить время написания и отладки программ. Следовательно, работы в данном направлении являются актуальными.

Краткий обзор визуальных языков. Направление создания визуальных языков параллельного программирования активно и давно развивается. Часть языков обладает общими чертами, с помощью которых языки можно условно разделить на группы. Наиболее интересными представляются следующие группы языков:

- группа, отображающая программы в виде функциональных зависимостей;
- группа, отображающая программы в виде асинхронно выполняемых последовательностей задач;
- группа, отображающая программы через представление их потоков данных.

Визуальные языки, относящиеся к первой группе, содержат два типа элементов: функции и их аргументы. Результат выполнения функции – аргумент, являющийся либо целью алгоритма, либо входом для другой функции. При графическом изображении функциональных языков чаще всего используют древовидные структуры. Корнем дерева является финальная цель алгоритма. Листьями дерева являются входные данные алгоритма и функции. Все остальные узлы соответствуют алгоритму решения задачи. Ярким примером функционального языка является язык из системы «Пифагор» [1]. На рис. 1 представлен параллельный алгоритм суммирования элементов вектора.

Визуальные языки, отображающие асинхронно выполняемые задачи, представляют параллельную программу в виде направленного графа. В качестве вершин графа могут быть использованы один или несколько типов запускаемых задач. Ребра в графе отображают последовательность, в которой задачи должны быть выполнены. В данных языках, как правило, принято, что для задачи запускается столько копий, сколько на нее указывает ребер. Яркими представителями данной группы визуальных языков являются «CODE» [2] и «ГРАФ ПЛЮС» [3].

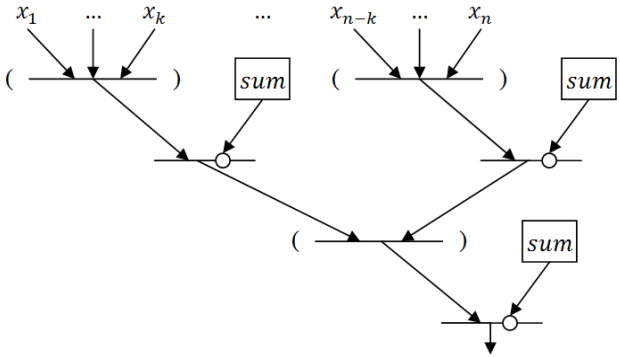

Рис. 1. Визуальная схема параллельного суммирования вектора с точки зрения функционального языка

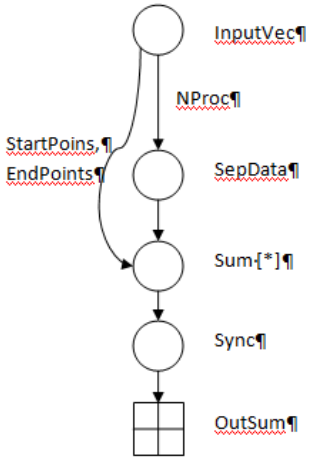

Рис. 2. Визуальная схема параллельного суммирования вектора в виде асинхронно выполняемых последовательностей задач

В зависимости от визуального языка граф может быть дополнен дополнительными графическими элементами, как это сделано в языке «HeNCE» [4]. Отличительной его особенностью является то, что в языке есть специальные графические элементы для представления графа параллельной программы в более компактном виде. К этим графическим элементам относятся обозначение условного исполнения задач, а также репликации цик-

лического и конвейерного исполнения задач в графе. Параллельное суммирование вектора элементов на графическом языке «CODE» [2] может выглядеть, как изображено на рис. 2.

Визуальные языки, отображающие потоки данных, также представляют параллельную программу в виде направленного графа, но смысл вершин и ребер меняется. В данных визуальных языках вершины графа – либо задачи, которые нужно исполнить, либо процессы, которые умеют выполнять заданный алгоритм. Сами задачи могут обмениваться данными. Для отображения передачи данных в языке используются ребра. Как правило, в данных языках считается, что задачи запускаются либо в момент старта программы, либо по готовности всех входящих данных для задачи. К данной группе языков относятся языки «TREPPER»[5] и «GRAPNEL» [6]. В обоих языках вершины графа описывают процессы. Ребра отображают каналы передачи данных. Для каждого процесса есть возможность отдельно описать выполняемую работу. Данные в этих языках могут передаваться через одни и те же каналы, причем передачу данных требуется указывать явным образом. Другие два языка «PGRAPH» [7] и «ЯГСПП» [8] явным образом не привязаны к понятиям процессов или потоков. Под задачей понимается вычислительный элемент, использующий входные данные для создания выходных. Данные при этом передаются автоматически по завершении задачи. Параллельное суммирование вектора элементов на графическом языке, подобном «PGRAPH» [7] и «ЯГСПП» [8], может выглядеть, как изображено на рис. 3.

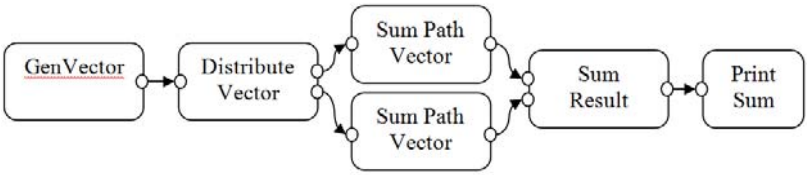

Рис. 3. Визуальная схема параллельного суммирования, отображающая потоки данных

Разрабатываемый визуальный язык. Разрабатываемая визуальная система параллельного программирования относится к последней группе и оперирует тремя основными типами визуальных элементов:

1. **«Задача»** – данный компонент содержит часть логики разрабатываемого алгоритма.

2. **«Порт»** и **«каналы»** – визуальные элементы, отображающие данные, передаваемые между задачами.

3. **«Компонент разделения»** и **«компонент слияния»** данных – необходимые визуальные элементы в параллельном программировании для отображения геометрической декомпозиции данных.

Алгоритм суммирования элементов вектора на разрабатываемом языке представлен на рис. 4.

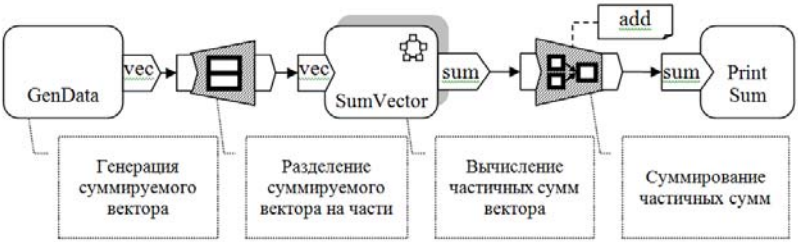

Рис. 4. Визуальная схема параллельного суммирования вектора в виде асинхронно выполняемых последовательностей задач

Одна из основополагающих идей разрабатываемого языка – за счет специальных визуальных элементов добиться компактности, прозрачности передачи данных и масштабируемости получаемого представления параллельных программ.

Рассмотрим более подробно пример визуальной схемы суммирования элементов вектора представленного на рис. 4. Первая задача «GenData» последовательная и существует для того, чтобы сгенерировать суммируемый вектор. Далее идет специальный визуальный элемент – компонент разделения данных. Методов разделения данных в визуальном языке преду-

смотрено несколько видов. Каждый из видов имеет свое графическое представление. Вид разделения данных визуально отображен на компоненте разделения данных. После разделения данных идет задача специального вида «SumVector». Можно обратить внимание на то, что задача «SumVector» отличается от остальных тем, что у нее есть тень. Тень означает, что задача имеет несколько копий. Предполагается, что количество копий может быть задано в момент исполнения, при этом компонент разделения данных сам понимает, как необходимо распределить данные между копиями. Важно заметить, что код задачи «SumVector» остается последовательным, но пишется для части полученных данных. Данный подход дает схемам масштабируемость и компактность представления. После подсчета частичных сумм с помощью компонента слияния данных вычисляется сумма всех элементов вектора. Как и в случае компонента разделения данных, метод слияния визуально отображен на компоненте слияния данных. В финале с помощью последовательной задачи «PrintSum» результат суммирования может быть отображен пользователю. Также следует заметить, что во всех визуальных элементах видно, какие и куда передаются данные, поскольку все данные предаются через порты.

В данной работе на простом примере рассмотрены различные подходы к визуальному представлению параллельных программ. Также коротко описан новый разрабатываемый визуальный язык, позволяющий описывать сложные параллельные алгоритмы в компактном виде. Компактность графического представления обусловлена наличием ряда введенных компонентов, таких как задачи с несколькими копиями, компоненты разделения и слияния данных. Задачи с несколькими копиями позволяют на схемах не рисовать повторяющиеся задачи и при этом получать более масштабируемые параллельные программы. Масштабируемость достигается за счет того, что часть введенных компонентов параметризировано. За счет изменения параметров можно изменять степень параллельности получаемых программ.

Целью дальнейших исследований является поиск расширений визуального языка, которые позволили бы:

- контролировать данные, с которыми работает пользователь системы;
- создавать и повторно использовать базу визуальных схем параллельных программ;
- используя базу визуальных схем, автоматически выстраивать схемы алгоритмов решения задач, поставленных пользователем.

Список литературы

1. Функциональная модель параллельных вычислений и язык программирования «Пифагор» / А.И. Легалов [и др.] – URL: http://www.softcraft.ru/parallel/fpp/fppcontent.shtml (дата обращения: 27.06.2010)

2. Experiences with CODE and HeNCE in Visual Programming for Parallel Computing / J.C. Browne [et al.] // IEEE Parallel and Distributed Technology. B. – 1994. – Vol. 3, No. 1. – P. 75–83.

3. Востокин С.В. Графический метод проектирования параллельных программ с использованием асинхронной событийной модели вычислений // Вестн. Самар. гос. техн. ун-та. Серия «Физико-математические науки». – 2004. – № 30. – С. 178–183.

4. Graphical Development Tools for Network-Based Concurremt Supercomputing / A. Beguelin [et al.] // Proc. Supercomputing ‘91, Albuquerque, NM. B, 1994. P.435-444.

5. Scheidler C., Schäfers L. TRAPPER:A Graphical Programming Environment for Industrial High-Performance Applications // PARLE '93 Parallel Architectures and Languages Europe. Lecture Notes in Computer Science B. 1993 Vol.694. – P. 403–413.

6. A graphical development and debugging environment for parallel programs / P. Kacsuk [et al.] // Parallel Computing, February, 1997. – Vol. 22. – P. 1747–1770

7. Жидченко В.В. Программный комплекс моделирования и анализа алгоритмов параллельных вычислений: дис. … к.т.н. – Самара, 2007. – 189 с.

8. Программные средства поддержки выполнения граф-схемных программ для кластерных систем / Д.В. Котляров [и др.] // Технологии Microsoft в теории и практике программирования: материалы конф. – Н. Новгород, 2006. – С. 156–159.