ЛУКАШЕНКО В. В., РОМАНЧУК В. А. МОДИФИКАЦИЯ РАБОТЫ АЛГОРИТМА ПРЕОБРАЗОВАНИЯ ОБРАТНОЙ ПОЛЬСКОЙ ЗАПИСИ ПРОГРАММЫ В АБСТРАКТНОЕ СИНТАКСИЧЕСКОЕ ДЕРЕВО

УДК 004.75, ВАК 05.13.18, ГРНТИ 28.25.23

Оптимизация работы алгоритма преобразования обратной польской записи программы в абстрактное синтаксическое дерево

Optimization of the conversion algorithm reverse Polish notation program in abstract syntax tree

В. В. Лукашенко, В. А. Романчук

Рязанский государственный университет имени С. А. Есенина, г. Рязань

V. V. Lukashenko, V. A. Romanchuk

Ryazan State University named for S. Yesenin, Ryazan'

В работе рассматривается вариант реализации вычислительного кластера нейрокомпьютеров. Для реализации основного принципа распределенных вычислений в статье представлен алгоритм разбиения задач, поступающих на выполнение в вычислительный кластер нейрокомпьютеров, на подзадачи. Задача, которой посвящена статья – оптимизация алгоритма преобразования обратной польской записи программы в абстрактное синтаксическое дерево. Для этого в статье предлагается представить поступившую на выполнение в кластер программу в модифицированной постфиксной Польской записи и хранить ее в стеке команд программы. На следующем шаге предлагается получить абстрактное синтаксическое дерево программы, следуя оптимизованному алгоритму перевода модифицированной

The implementation of the computational cluster of neurocomputers is considered in the article. To implement the basic principle of distributed computing, the article presents an algorithm for dividing the tasks included in the computing cluster of neurocomputers into subtasks. The task, which is devoted to the article, is the optimization of the algorithm for converting the inverse Polish program into an abstract syntax tree. For this, the article proposes to present the program entered into the cluster for execution in the modified post-entry Polish record, and store it in the stack of program commands. The next step is to get the abstract syntax tree of the program, following the optimized algorithm to translate the modified Polish postfix record from the command stack to an abstract syntax tree.

постфиксной Польской записи из стека команд в абстрактное синтаксическое дерево.

Ключевые слова: *распределенные вычисления, кластерные вычисления, кластеризация вычислительных ресурсов, нейровычисления, нейрокомпьютеры, кластеризация нейрокомпьютеров, оптимизация, оптимизация алгоритмов.*

Keywords*: distributed computing, cluster computing, clustering of computing resources, neuro computing, neurocomputers, clustering of neurocomputers, optimization, algorithm optimization.*

Введение

В современной научной и производственной сфере крайне актуальна задача использования вычислительных систем параллельной обработки информации ввиду недостаточности вычислительных ресурсов. Сегодня обозначенные задачи решают с недостаточно высокой степенью эффективности масштабируемые параллельные вычслительные кластеры на базе нейрокомпьютеров [1]. В работе рассматривается существующая реализация параллельного кластера на базе нейрокомпьютеров МВ77.07 ЗАО «НТЦ Модуль», технические характеристики и перспективы использования в работе которых, описаны в работе [2–4].

При выборе нетипичной вычислительной архитектуры требуется решить задачу построения модели вычислений. Модель вычислений связывает между собой понятия архитектуры вычислительной системы и модель программирования. В параллельных вычислительных системах отражает полноту взаимодействия процессов. Вопрос разработки модели вычислений рассматривается в работе [5], в ней рассматриваются алгоритмы, разработка которых позволяет реализовать работу модели вычислений параллельного кластера нейрокомпьютеров. Алгоритм разбиения задач на подзадачи, который необходим для декомпозиции поступающих на выполнение в вычислительный кластер задач является одним из таких алгоритмов. Его работа позволяет организовать один из основных принципов параллельных вычислений, а именно – параллелизм. В основной части алгоритма решается задача перевода из обратной польской записи в абстрактное синтаксическое дерево программного кода. Абстрактное синтаксическое дерево – основная структура представляющая процесс передачи данных от одной операции программы к другой. Следует, отметить что скорость работы алгоритма перевода из обратной польской записи в абстрактное синтаксическое дерево крайне низка и требует оптимизации. В связи с этим основной проблемой, поставленной и рассматриваемой в статей является оптимизация работы алгоритма преобразования обратной польской записи программы в абстрактное синтаксическое дерево.

Основная часть

Для оптимального разбиения задачи на подзадачи рассмотрим ее с точки зрения алгоритма. В этом случае, каждая, поступающая на выполнение в параллельный кластер задача – есть алгоритм $Al^{(j)}$, реализованный в понятном вычислительной системе виде. Выполнение программы $PR^{(j)}$ на вычислительной машине или группе параллельных вычислительных машин осуществляется посредством четких машинных команд. Тогда, для оптимального разбиения программы на подпрограммы, а затем и их выполнения требуется рассмотреть ее представление с позиции теории компиляции.

С теории компиляции всякая программа представляется в виде $PR^{(j)} = (Gpu, V)$, где $Gpu = (W, E, start, stop)$ – граф потока управления программы, а $V = \{v_1, \ldots, v_k\}$ – алфавит операторов.

Граф потока управления $Gpu = (W, E, start, stop)$ – это ориентированный граф, в котором выделены две вершины start и stop, связанные между собой посредством множеств вершин W и дуг E.

В рассматриваемом графе отдельные операции над операндами, выполняющиеся независимо являются вершинами, абстрактные направления данных, необходимых для выполнения операций – дугами.

Для описания программы исполняющейся на вычислительных машинах традиционной архитектуры с теории компиляции этого достаточно. Для вычислительных машин представленного параллельного кластера нейрокомпьютеров, важен параметр разрядности данных, это связанно с тем, что за один процессорный такт на физическом вычислительном ядре нейрокомпьютера может выполниться до 64 операций, что качественно влияет на процесс параллелизма в кластере, и, в свою очередь, меняет процесс обхода графа потока управления.

Поэтому граф потока управления программы можно представить следующим образом $Gpu = (V, E, start, stop, BitVal)$, где $BitVal$ – множество значений разрядности результатов на каждом шаге выполнения программы.

Для разбиения программы на подпрограммы с позиции компиляции, требуется рассмотреть программу во внутренней форме представления.

Внутренняя форма представления программ – это результат работы синтаксического анализатора. Различают и на практике используют множество внутренних форм представления программ: постфиксная польская запись, префиксная польская запись, синтаксическое дерево, тетрады. Наиболее распространенным способом представления программ во внутренней форме является обратная польская запись [10, 11].

В рассматриваемой работе выбрана обратная польская запись в качестве внутренней формы представления программ. Для записи всей программы в польской форме необходимо применить модифицированый алгоритм представления программы в обратной польской записи.

Модифицированный алгоритм, позволяющий представлять всю программу в постфиксной польской записи, а не только ее логические и математические выражения работает по тому же принципу и был предложен Робертом Седжвиком [17].

Алгоритм перевода из Польской формы в абстрактное синтаксическое дерево

После представления программы в обратной Польской записи необходимо построить граф потока управления программы. Представление программы в расширенной обратной Польской записи хранится и передается в форме абстрактного типа данных класса стек. Такой стек называется стеком команд программы (*Stack*). На следующем шаге необходимо сформировать абстрактное синтаксическое дерево (АСД) программы из стека команд программы. Абстрактное синтаксическое дерево – это ориентированное дерево, где внутренние вершины являются операторами языка, на котором написана программа, а листьями – соответствующие операнды. Листья нижнего слоя – это входные параметры, а, в свою очередь, корень дерева – результат, который должна вернуть программа. Ребра АСД формируются между операторами и соответствующими операндами, требующихся для их работы. Для этого существует алгоритм перевода программы из Польской формы в абстрактное синтаксическое дерево (рис. 1). Следует, отметить что скорость работы алгоритма перевода из обратной польской записи в абстрактное синтаксическое дерево крайне низка и требует оптимизации. В связи с этим рассмотрим оптимизованный алгоритм преобразования обратной польской записи программы в абстрактное синтаксическое дерево.

Будем хранить в *R* текущий считываемый из *Stack* элемент, до тех пор пока он не пуст.

– В случае, когда *R* – идентификатор или константа, его значение считывается из стека и записывается в лист АСД а также в многосвязный список ссылаясь на элемент родитель в дереве, затем переходим к считыванию следующего элемента.

– В случае, когда *R* является бинарным оператором, его действие осуществляется над двумя последующим элементам, которые являются операндам из стека, так формируется левое поддерево АСД. Затем переходим к считыванию следующего элемента.

– В случае, когда *R* является унарным оператором, его действие осуществляется над верхним последующим элементом стека, так формируется левое поддерево АСД. Затем переходим к считыванию следующего элемента.

– В противном случае *R* является *n*-арным оператором, его действие осуществляется над всеми n верхними последующими операндами, так формируется левое поддерево АСД. Затем переходим к считыванию следующего элемента.

На рисунке 1 указаны следующие программные функции, посредством которых реализована работа указанных выше правил:

isEmpty() – контролирует наличие или отсутствие элементов в стеке.

Read(R) – осуществляет считывание очередного элемента из стека.

isConst() – проверка *R* на наличие признаков идентификатора или константы.

operBinary() – проверка *R* на наличие признаков бинарного оператора.

operUnary() – проверка *R* на наличие признаков унарного оператора.

operN() – проверка *R* на наличие признаков n-арного оператора.

AST() – формирование очередного левого поддерева или листа АСД.

Результатом работы программы является абстрактное синтаксическое дерево графа, хранящееся в памяти в виде многосвязного списка с пропусками по Р. Седжвику [17].

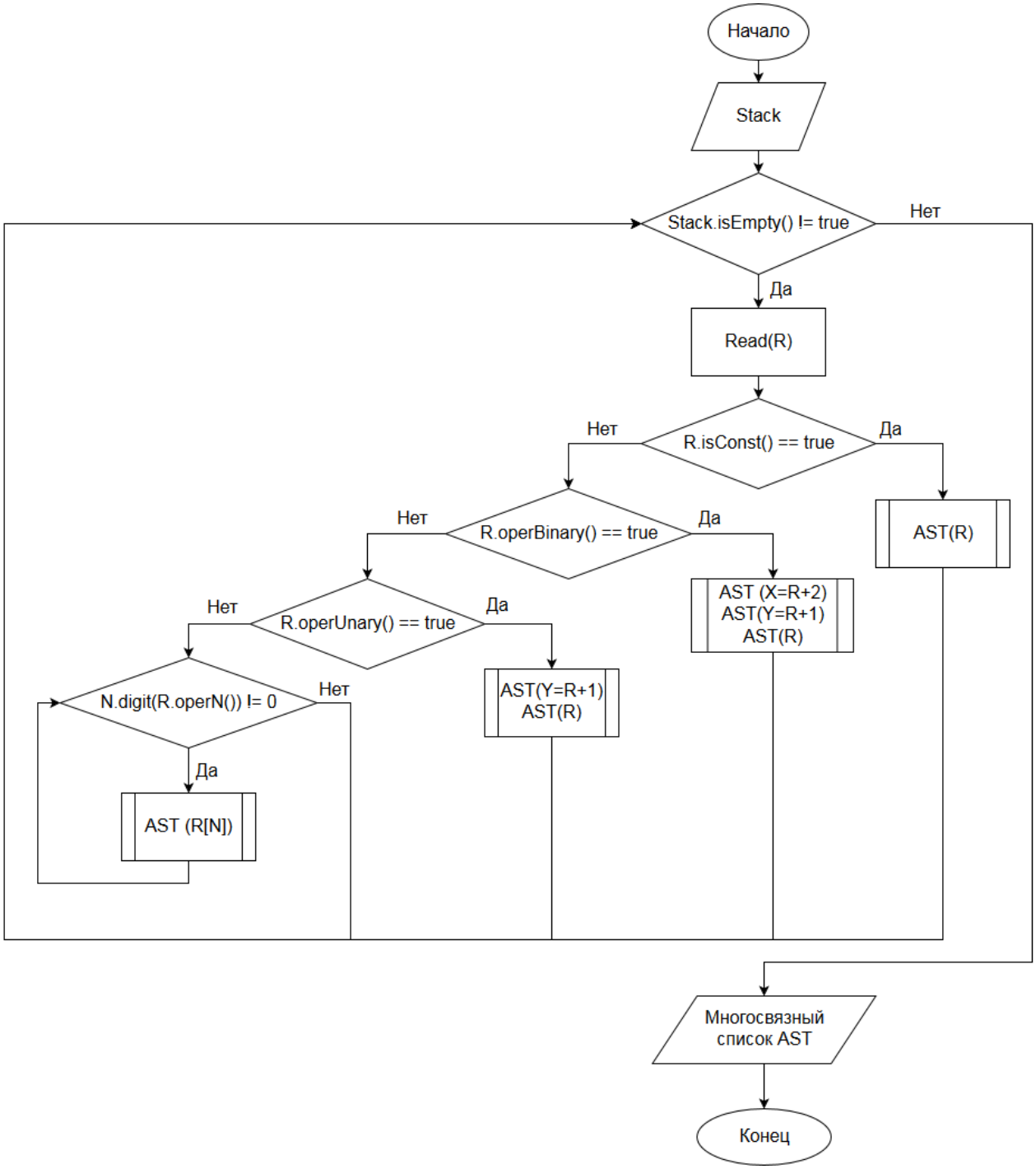

Рисунок 1. Блок-схема работы оптимального алгоритма перевода программы из Польской формы в АСД

Вывод

Таким образом, в статье рассмотрен вариант оптимизации алгоритма перевода программы, все операции которой, записаны в модифицированной обратной польской записи, в абстрактное синтаксическое дерево, который отобразит множество всех путей исполнения программы. Результатом работы данного алгоритма является алгоритмическая структура – многосвязный список, посредством которой осуществляется хранение и передача для дальнейшей обработки результатов представления программы в АСД.

Список литературы

1. Бурцев В. С. Параллелизм вычислительных процессов и развитие архитектуры супер ЭВМ. М. : ИВВС РАН, 1997. 152 с.

2. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. СПб. : БХВ-Петербург, 2002. 608 с.

3. Галушкин А. И. Нейронные сети: основы теории. М. : «Горячая линия Телеком», 2010. 496 с.

4. Топорков В. В. Модели распределенных вычислений. М. : ФИЗМАТЛИТ, 2004. 320 с.

5. Ручкин В. Н., Романчук В. А., Лукашенко В. В. Обобщенная модель вычислений кластера нейрокомпьютеров // Вестник Рязанского государственного университета им. С. А. Есенина. 2015. № 2 (47). С. 146–150.

6. Гергель В. П., Полежаев П. Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Вестник Нижегородского университета им. Н. И. Лобачевского. 2010. № 5–1. С. 201–208.

7. Злобин В. К., Ручкин В. Н. Нейросети и нейрокомпьютеры : учеб. пособие. СПб. : БХВ-Петербург, 2011. 256 с.: ил

8. Коваленко В. Н., Коваленко Е. И., Корягин Д. А. [и др.]. Управление заданиями в распределенной вычислительной среде // Открытые системы. 2001. №5–6. С. 22–28.

9. Комарцова Л. Г., Максимов А. В. Нейрокомпьютеры : учеб. пособие для вузов. М. : Изд-во МГТУ им. Н. Э. Баумана, 2002. 320 с., ил.

10. Коновалов Н., Крюков В. Параллельные программы для вычислительных кластеров и сетей // Открытые системы. 2002. №3. С. 12–18.

11. Корячко В. П. Алгоритм планирования вычислительного процесса в мультипроцессорной вычислительной системе реального времени // Автоматика и вычислительная техника. 1985. № 3. С. 16–18.

12. Костров Б. В., Ручкин В. Н. Архитектура микропроцессорных систем : учеб. пособие. М. :ТЕХБУК, 2007. 208 с.

13. Bender M. A., Bunde D. P., Demaine E. D. Communicationn Aware Processor Allocation for Supercomputers // Lecture Notes in Computer Science. V. 3608/2005. Berlin : Springer, 2005. P. 1699181.

14. Танненбаум Э., Ван Стен М. Распределенные системы. Принципы и парадигмы. СПб. : Питер, 2003. 877 с.

15. Тель Ж. Введение в распределенные алгоритмы. Пер. с англ. М. : МЦНМО, 2009. 616 с.

16. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. М. : Мир, 1975. 544 с.

17. Роберт Седжвик. Алгоритмы на C++. Фундаментальные алгоритмы и структуры данных. Algorithms in C++. М. : «Вильямс», 2011. 1056 с.

18. Вирт Н. Алгоритмы и структуры данных / Пер. с англ. Ткачев Ф. В. М. : ДМК-Пресс, 2010. 272 с.

List of references

1. Burcev, V. S., *Parallelism of computational processes and the development of architecture in super computers*, Moscow : IAF RAS, 1997, 152 p.

2. Voevodin, V. V., Voevodin, Vl. V., *Parallel computing*, Saint Petersburg : BHV-Peterburg, 2002, 608 p.

3. Galushkin, A. I., *Neural network: basic theory*, Moscow : Hotline Telekom, 2010, 496 p.

4. Toporkov, V. V., *Models of distributed computing*, Moscow : FIZMATLIT, 2004, 320 p.

5. Ruchkin, V. N., Romanchuk, V. A., Lukashenko, V. V., "A generalized computing model of a cluster of Neurocomputers", *Bulletin of the Ryazan state University. S. A. Esenin*, 2015, no. 2 (47), pp. 146–150.

6. Gergel', V. P., Polezhaev, P. N. "The study of scheduling algorithms for parallel tasks cluster computing systems using the simulator", *Bulletin of the Lobachevsky University*, 2010, no. 5–1, pp. 201–208.

7. Zlobin, V. K., Ruchkin, V. N., *Neural networks and Neurocomputers*, tutorial, СПб. : БХВ-Петербург, 2011. 256 с.: ил.

8. Kovalenko, V. N., Kovalenko, E. I., Koryagin, D. A., et al., "Job management in a distributed computing environment", *Open systems*, 2001, no. 5–6, pp. 22–28.

9. Komarcova, L. G., Maksimov, A. V., *Neurocomputers*, textbook for high schools, Moscow : Bauman Moscow state technical university, 2002, 320 p.

10. Konovalov, N., Kryukov, V., "Parallel programs for computing clusters and networks", *Open systems*, 2002, no. 3, pp. 12–18.

11. Koryachko, V. P., "The scheduling algorithm of the computational process in a multiprocessor computing system of real time", *Automation and computer engineering*, 1985, no. 3, pp. 16–18.

12. Kostrov, B. V., Ruchkin, V. N., *Architecture of microprocessor systems*, tutorial, Moscow : TECHBOOK, 2007, 208 p.

13. Bender, M. A., Bunde, D. P., Demaine, E. D., "Communicationn Aware Processor Allocation for Supercomputers", *Lecture Notes in Computer Science*, v. 3608/2005, Berlin : Springer, 2005, p. 1699181.

14. Tannenbaum, Eh., Van Sten, M., *Distributed systems. Principles and paradigms*, Saint Petersburg : Piter, 2003, 877 p.

15. Tel', Zh., *Introduction to distributed algorithms*, translation from English, Moscow : MCFCME, 2009, 616 p.

16. Gris, D., *The design of compilers for digital computers*. Moscow : World, 1975, 544 p.

17. Robert, Sedzhvik, *Algorithms in C++. Fundamental algorithms and data structures,* Moscow : Williams, 2011, 1056 p.

18. Virt, N., *Algorithms and data structures*, translation from English. F. V. Tkachev, Moscow : DMK-Press, 2010, 272 p.