УДК 681.3

С.Д. ПОГОРЕЛЫЙ, И.Ю. ШКУЛИПА

КОНЦЕПЦИЯ СОЗДАНИЯ ПАРАМЕТРИЧЕСКОЙ СИСТЕМЫ ПРОЕКТИРОВАНИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ И ИХ ПРОГРАММНЫХ РЕАЛИЗАЦИЙ

Ключевые слова: *алгоритм, модифицированные системы алгоритмических алгебр, САА-М-схема, формализация, парадигмы параллельного программирования, процесс, поток, параллелизм, Java Multithreading*.

ВВЕДЕНИЕ

Специфика и функциональная ориентация современных систем параллельной обработки данных, создание новых архитектур (многоядерных микропроцессоров фирм Intel и Sun Microsystems, симметричных мультипроцессорных систем SMP, распределенных и кластерных архитектур и т.п.), программных платформ и технологий не позволяют в полной мере пользоваться существовавшими ранее средствами проектирования параллельных алгоритмов и ассоциированных с ними программ и выдвигают актуальную задачу разработки новых средств.

Алгоритмический этап проектирования параллельных алгоритмов и программ является начальным и наименее исследованным. Он влияет на весь дальнейший процесс разработки и определяет ее результативность в целом. Инструментарий этого этапа проектирования обеспечивают:

- эффективные средства описания параллельных алгоритмов функционирования систем параллельной обработки данных с учетом их специфики;
- возможность формальных эквивалентных преобразований алгоритмов в целях их исследования и анализа;
- средства описания темпоральных (зависящих от времени) и асинхронных фрагментов алгоритмов;
- возможность генерации ассоциированных с исследуемым алгоритмом программ для различных парадигм параллельного программирования [1].

Для формализованного описания алгоритмов на этапе алгоритмического проектирования предлагается использование аппарата модифицированных систем алгоритмических алгебр (САА-М) В.М. Глушкова [2, 3]. САА-М допускают представления любого алгоритма в аналитическом виде. В свою очередь, САА-М-схемы можно формально модифицировать путем применения к ним эквивалентных преобразований, что дает полнофункциональную основу для автоматизированного преобразователя алгоритмов.

Работа посвящена описанию инструментария — системы автоматизированного параметрического проектирования алгоритмов (САПП), предназначенной для преобразования и оптимизации алгоритмов с последующей генерацией программного кода, реализующего соответствующий алгоритм на целевом языке программирования. Идеи автоматизации программирования кластерных систем, которые стали доминантой проводимых в настоящее время исследований, восходят к концепциям, сформированным под руководством Е.Л. Ющенко в отделе автоматизации программирования Института кибернетики.

Задача создания САПП заключается в разработке программного комплекса, допускающего выполнение проектирования, анализа алгоритмов и генерации соответствующих параллельных программ на разных языках программирования в автоматизированном режиме, а также моделирование их работы и сбор результатов.

ОПИСАНИЕ СИСТЕМЫ

Организационно САПП состоит из трех компонентов (рис. 1):

- подсистема преобразования САА-М-схем алгоритмов;
- компилятор схем алгоритмов в программный код;
- подсистема выполнения и сбора статистики.

Рис. 1

Подсистема преобразования САА-М-схем алгоритмов как функциональная составляющая САПП предназначена для преобразования и оптимизации входного алгоритма в целях формирования новой или параллельной версии. Аналитический вид алгоритма базируется на его представлении в виде формализованной САА-М-схемы. В терминах САА-М эквивалентные преобразования — это набор теорем и аксиом, описывающих правила преобразования частей алгоритмической схемы в новые операторные конструкции.

Преобразователь САА-М-схем алгоритмов, как недетерминированный конечный автомат (НКА), задан соотношением $SAATransformer = (Q, E, d, q_0, F)$. Здесь

- Q — конечное множество состояний автомата;
- E — множество входных символов нотации схем алгоритмов;
- q_0 — начальное состояние НКА, определяемое входной САА-М-схемой алгоритма;
- F — конечное множество допустимых состояний автомата, представляющее собой множество схем алгоритмов, полученных путем применения к входной схеме всех эквивалентных преобразований;
- d — функция переходов, аргументами которой являются состояние автомата из Q и входная схема алгоритма из E, а значением — подмножество множества Q.

Схема функции переходов приведена на рис. 2, который иллюстрирует обработку входных данных автоматом. На входе имеем схему алгоритма и набор эквивалентных преобразований. Очевидно, что некоторые из преобразований невозможно применить к входной схеме, поэтому из состояния q_0 автомат может перейти либо в состояние q_0 — собственно входную схему алгоритма, либо в состояние q_{11}, представляющее на выходе новую схему алгоритма, который получен путем применения к входной схеме соответству-

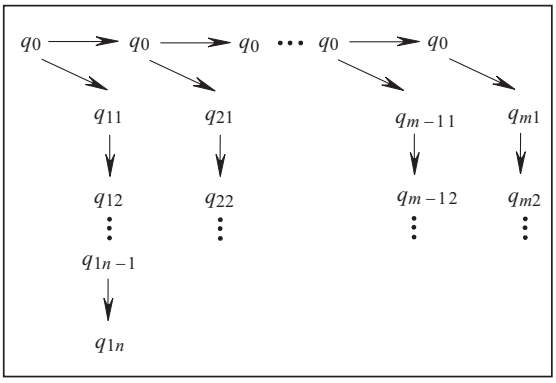

Рис. 2

ющего эквивалентного преобразования. Далее из состояний q_0 и q_{11} автомат может перейти в состояния, соответствующие применению следующих по списку эквивалентных преобразований. Этот процесс будет происходить до тех пор, пока не будут использованы все возможные преобразования и не будет получен ряд соответствующих САА-М-схем алгоритмов. Группа состояний $q_{11}-q_{m1}$ соответствует применению к входной схеме эквивалентных преобразований, а группа состояний $q_{12}-q_{m2}$ — следующего по списку эквивалентного преобразования к группе состояний $q_{11}-q_{m1}$.

Языком данного НКА в общем случае является множество, состоящее из любых последовательностей символов, используемых для записи схем алгоритмов.

Отметим, что некоторые состояния автомата могут быть условно эквивалентными, т.е. соответствовать на выходе одинаковым САА-М-схемам. Это связано с тем, что некоторые эквивалентные преобразования могут быть неприменимы к входной схеме или применение ряда преобразований может дать одинаковый результат. Исключение или частичное исключение таких состояний из результирующего множества должно быть дополнительной частью практической реализации представленного НКА. Трансформация происходит в автоматическом режиме, и пользователь определяет стратегию применения преобразований.

Эквивалентные преобразования САА-М на практике приводят к трансформации части аналитической схемы, соответствующей регулярному выражению, к новой части, которая отвечает другому регулярному выражению. Этот процесс фактически сводится к преобразованию $RL(E) \to RR(E)$, где $RL(E)$ — регулярное выражение над алфавитом E, которое соответствует левой части теоремы об эквивалентных преобразованиях, а $RR(E)$ — регулярное выражение над алфавитом E, которое соответствует правой части теоремы.

Например, для теоремы «Свойство цикла» задается преобразование цикла while с двумя операторами, оперирующими независимыми данными: ${}_{\alpha}\{A*B\} = {}_{\alpha}\{A\}//_{\alpha}\{B\}$. Регулярное выражение для левой части имеет следующий вид:

$$RL(E) = (?[a-z;\ \&;\ /\ ;\ \wedge]?)(\{(?[A-Z;0-9*]*)?[*]?(?[A-Z;0-9*]*)?\}).$$

Это выражение задает конструкцию, в которой сначала следует набор символов $a-z;\ \&;\ /;\ \wedge$, задающий условие цикла, затем открывающаяся фигурная скобка, далее набор символов $A-Z$; $0-9$, задающий первый оператор цикла, знак *, который в терминах САА обозначает композицию операторов, набор символов $A-Z$; $0-9$, обозначающий второй оператор цикла, и закрывающаяся фигурная скобка. Регулярное выражение

$$RR(E) = \$\{CONDS\}\{\$\{OPERS\_LEFT\}\}//\$\{CONDS\}\{\$\{OPERS\_RIGHT\}\}$$

задает правую часть преобразования, где введено обозначение для условия цикла (*CONDS*), первого оператора в цикле (*OPERS_LEFT*) и второго оператора в цикле (*OPERS_RIGHT*).

Первой фазой синтаксического анализа схемы является проверка корректности записи схемы. Основные условия корректности записи:

- отсутствие в файле схемы любых символов, которые не входят в алфавит принятых условий и обозначений;
- количество левых скобок определенного типа должно быть равно количеству правых скобок того же типа;
- идентификаторы условий и операторов должны соответствовать принятым условным обозначениям; идентификатором считается любая часть схемы, содержащаяся между двумя разделителями (скобки и обозначения операций над операторами и условиями).

В основе реализации подсистемы преобразования САА-М-схем алгоритмов лежит база данных, содержащая регулярные выражения для всех эквивалентных преобразований алгоритмов, а также схемы алгоритмов, которые были введены

и получены путем применения эквивалентных преобразований. Все эквивалентные преобразования сохраняются в базе данных и могут быть изменены в процессе работы. Это повышает гибкость применения преобразований и значительно облегчает процесс добавления того или иного преобразования. Выбор эквивалентных преобразований схем алгоритмов проходит в диалоговом режиме, что позволяет пользователю самому выбирать, какие преобразования применять к выбранной схеме (рис. 3).

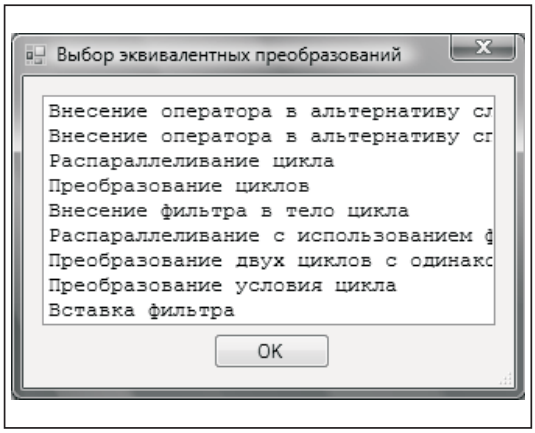

Рис. 3

Подсистема генерации программного кода предназначена для синтеза по САА-М-схемам параллельных алгоритмов программного кода на целевых языках программирования. Программная реализация генератора кода осуществляет синтез программных реализаций алгоритмов языками программирования С с применением процесс-ориентированной, поток-ориентированной и MPI парадигм параллельного программирования, а также на языке Java с использованием парадигмы Java Multithreading.

Генератор кода представляет собой НКА, на вход которого подается САА-М-схема алгоритма с дополнительной конкретизацией используемых условий и операторов, а на выходе получаем программу на выбранном целевом языке программирования с применением заданной парадигмы параллельного программирования

$$CodeGenerator = (Q, E, d, q_0, F),$$

где Q, E, q_0 и d аналогичны обозначениям в соотношении для *SAATransformer*, а F — конечное множество допустимых состояний автомата, которое в общем случае является множеством корректных программ. Схема функции переходов приведена на рис. 4.

Конечные состояния НКА генератора кода соответствуют синтезированным программам на языке С с применением парадигм параллельных процессов, параллельных потоков и MPI, а также на языке Java с применением парадигмы параллельных потоков Java.

В САПП код на основе схемы алгоритма генерируется в три этапа:

- разбор схемы на лексические конструкции — операторы и условия, используемые в алгоритме;
- генерация древовидной схемы алгоритма (дерева разбора);
- генерация кода на языке с применением парадигм параллельного программирования.

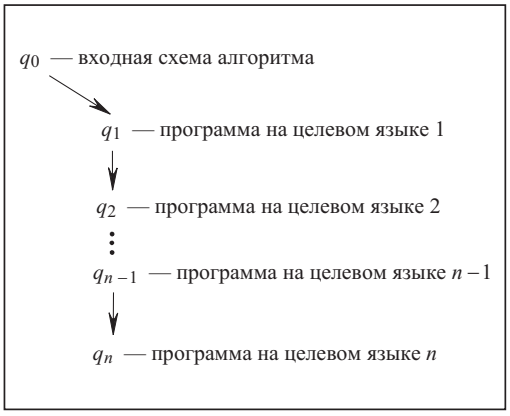

Рис. 4

Все этапы генерации программного кода идут последовательно. Генератор программного кода достаточно параметризован для генерации программ на целевых языках программирования с применением парадигм параллельного программирования. В нем также заложена возможность дальнейшего расширения.

Основным этапом процесса генерации кода является построение

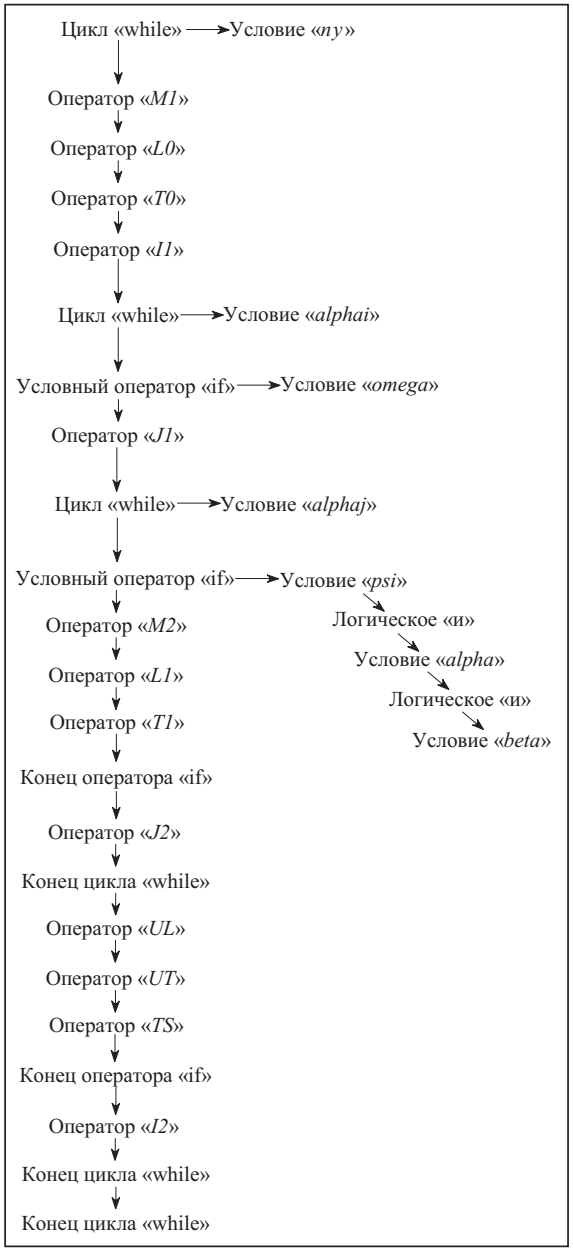

Рис. 5

дерева разбора САА-М-схемы алгоритма [4]. В генератор кода входит синтаксический анализатор, анализирующий входную схему алгоритма и представляющий ее в виде дерева разбора. Дерево разбора схемы фактически задает блок-схему результирующей программы, реализующей заданный алгоритм, без интерпретации операторов и условий, которые были обозначены идентификаторами САА-М при проектировании входной схемы.

Построение дерева разбора САА-М-схемы проиллюстрировано на примере основного цикла алгоритма Прима нахождения каркаса минимального веса графа [5]. САА-М-схема основного цикла алгоритма Прима имеет следующий вид:

$$PrimMainCycle =$$

$$ny\{M1*L0*T0*I1*$$

$$alphai\{omega(J1*alphaj\{$$

$$[psi\&alpha\&beta](M2*L1*T1)$$

$$*J2\}*UL*UT*TS)*I2\}\}.$$

В этой части схемы использованы три вложенных цикла и два оператора альтернативы. Дерево разбора приведенной части схемы изображено на рис. 5.

При использовании дерева грамматического разбора существенно облегчается генерация кода, что связано с исключением применения теории регулярных выражений для преобразования кода САА-М-схемы в программный код на целевом языке программирования. Такой подход также дает возможность генерировать код в зависимости от уровня вложенности того или иного оператора, выполняя форматирование текста программы на этапе генерации, что значительно облегчает чтение и редактирование синтезированной программы.

Подсистема выполнения и сбора статистики САПП предназначена для компиляции файлов программ на целевом языке, исполнения и сбора результатов работы приложений, полученных после компиляции схем алгоритмов в программный код. Она включает также интерфейс пользователя, предназначенный для слежения за процессом преобразования схем и участия пользователя в процессе создания программы алгоритма [6]. В общем случае функционирование этой подсистемы состоит из следующих шагов:

• использование подсистемы преобразования схем алгоритмов для синтаксического анализа входного файла схемы;

• применение к схеме алгоритма эквивалентных трансформаций по выбору пользователя и получение ряда трансформированных схем алгоритмов;

• синтез соответствующих программ на основе полученных схем алгоритмов;

• компиляция полученных программ в исполняемые файлы;

• выполнение программ по требованию пользователя и вывод временных результатов работы программ.

Подсистема выполнения и сбора статистики генерирует сценарии для выполнения программ алгоритмов на кластерах Киевского национального университета имени Тараса Шевченко для платформ MS Compute Cluster Server и Linux [7, 8]. Схема выполнения синтезированных программ приведена на рис. 6.

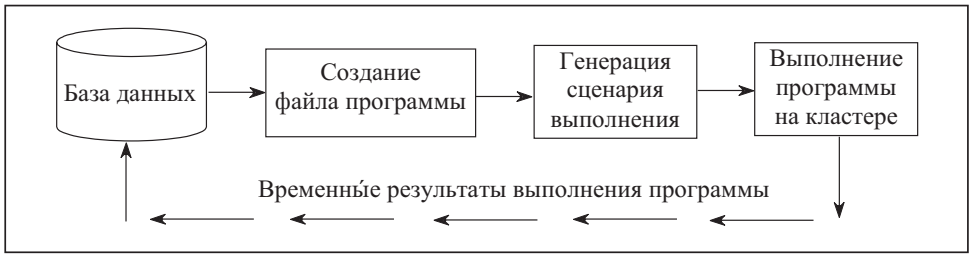

Рис. 6

Подсистема выполнения допускает обобщения на случай выполнения работы нескольких программ. В этом варианте схема работы подсистемы в целом не меняется, за исключением того, что каждый этап проходит в виде циклического выполнения по списку всех программ.

Одна из основных задач подсистемы — генерация сценариев для компиляции программ на целевом языке, их выполнение и сбор результатов работы. Очевидно, для разных аппаратных и программных систем сценарии отличаются, учитывая особенности компиляции и запуска программ на исполнение. В общем случае сценарий исполняется по схеме на рис. 7.

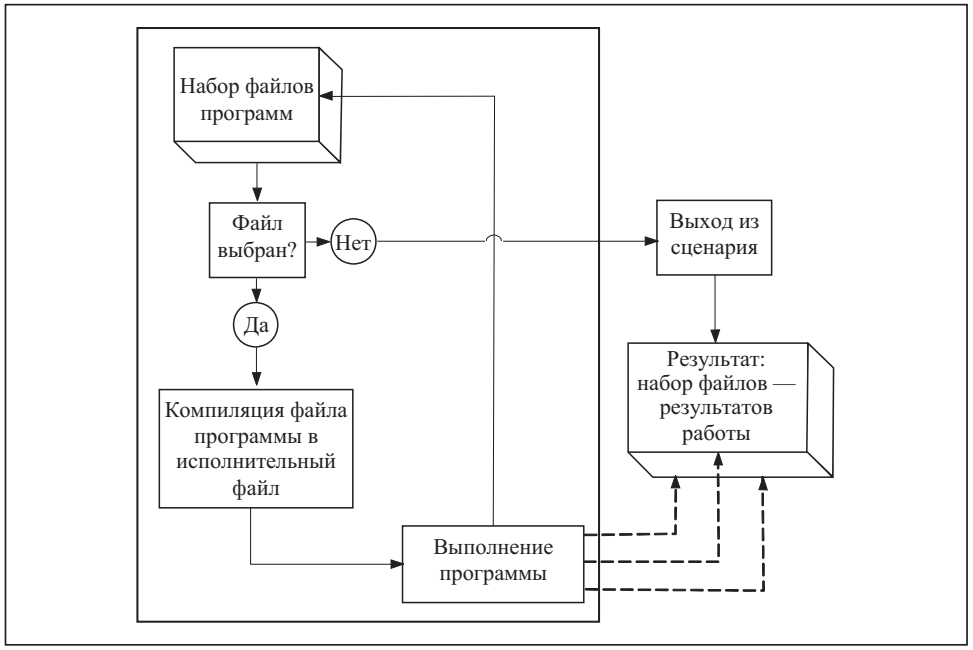

Рис. 7

Выполнение программы, синтезированной согласно алгоритму, идет на аппаратных платформах, без применения моделирующих программных средств. Таким образом, обеспечивается достоверность временны́х результатов выполнения и относительного ускорения работы алгоритма по сравнению с его исходной версией, а также реальная оценка целесообразности выбора преобразованного алгоритма для решения задачи.

Инструментарий разработки. Реализована САПП на платформе MS .Net Framework на языке C#. Как система управления базами данных использована MS Office Access 2007. Платформа MS .Net имеет встроенные мощные средства для обработки регулярных выражений, средства для построения деревьев и работы с базами данных. Тестирование САПП и ее экспериментальные испытания проведены на кластере Киевского национального университета имени Тараса Шевченко для платформы MS Compute Cluster Server.

К перспективам разработки можно отнести реализацию кода на кросс-платформенных технологиях Java, а также применения других СУБД, например MySQL или Oracle, что даст возможность использовать систему на различных аппаратных и программных платформах.

ЗАКЛЮЧЕНИЕ

Сформулирована концепция системы автоматизированного параметрического проектирования параллельных алгоритмов, позволяющая выполнять исследования, анализ и оптимизацию алгоритмов в автоматизированном режиме с минимальным вмешательством со стороны пользователя.

Выполнена программная реализация изложенной концепции. На данный момент САПП находится в стадии опытной эксплуатации.

Создание такого гибкого и мощного инструментария позволяет существенно облегчить работу при создании и исследовании параллельных версий алгоритмов и значительно сократить время, затраченное на анализ и реализацию подходов к оптимизации того или иного алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. Погорелый С.Д. К формализации этапа алгоритмического проектирования микропроцессорных систем // УСиМ. — 1995. — № 6. — С. 3–8.
2. Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К. Многоуровневое структурное проектирование программ. — М.: Финансы и статистика, 1989. — 208 с.
3. Ющенко К.Л. , Суржко С.В., Цейтлін Г.О., Шевченко А.І. Алгоритмічні алгебри. — К.: ІЗМН, 1997. — 480 с.
4. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений: Пер. с англ. — 2-е изд. — М.: Вильямс, 2002. — 528 с.
5. Бойко Ю.В., Погорілий С.Д., Шкуліпа І.Ю. Дослідження паралельних схем алгоритму Прима // Мат. машины и системы. — 2007. — № 2. — С. 77–89.
6. Погорілий С.Д. Програмне конструювання: Підручн. / За ред. О.В. Третяка. — Київ: ВПЦ Київ. ун-т, 2005. — 438 с.
7. Концепція створення гнучких гомогенних архітектур кластерних систем / С.Д. Погорілий, Ю.В. Бойко, Д.Б. Грязнов та ін. // Пробл. програмування. — 2008. — № 2–3. — С. 84–90 (Спец. вип. Мат. міжнар. конф. УкрПРОГ 2008).
8. http://cluster.univ.kiev.ua.

Поступила 08.07.2009